POLITECHNIKA ŁÓDZKA

SYSTEMY ROZPROSZONE

DR INŻ. WŁODZIMIERZ MOSOROW

2007

Spis treści

Wstęp

[17]Systemem rozproszonym nazywamy taki system, w którym przetwarzanie informacji odbywa się na wielu komputerach, często znacznie oddalonych geograficznie (od kilku metrów do dziesiątków tysięcy kilometrów). Przeciwieństwem jest system izolowany lub scentralizowany. Obecnie właściwie wszystkie systemy (poza domowymi komputerami) są rozproszone. Ogromnym katalizatorem rozproszenia systemów jest Internet. Komputer z Internetem można już uważać za system rozproszony.

Tendencja do budowy systemów rozproszonych jest pochodną rozbudowy tanich, szybkich, uniwersalnych i niezawodnych sieci komputerowych.

[17]Popularne architektury rozproszenia:

Klient-serwer: rozproszony system ma wyróżniony węzeł zwany serwerem, oraz szereg podłączonych do niego węzłów zwanych klientami. Związek nie jest symetryczny: serwer wykonuje usługi zlecane przez klientów, nie może im odmówić i nie może im zlecić wykonanie usług.

Klient-multi-serwer: podobnie jak dla architektury klient-serwer, ale istnieje wiele serwerów. Przykładem jest WWW.

Koleżeńska (*peer-to-peer, P2P*): wiele węzłów świadczy sobie wzajemne usługi poprzez bezpośrednie połączenie; nie ma wyraźnego podziału na usługodawców i usługobiorców. Przykładem jest Gnutella, NXOR, w mniejszym stopniu Napster.

Architektura oparta na oprogramowaniu pośredniczącym (*middleware*): nie występuje podział na klientów i serwery. Węzły komunikują się poprzez specjalne oprogramowanie pośredniczące, które zakłada wspólny (przezroczysty dla użytkowników) protokół komunikacyjny. Przykładem jest CORBA (rozproszone obiekty), .NET/COM/DCOM, Java Beans/RMI, SOAP.

Wielkie, międzynarodowe firmy, a często nawet o zasięgu globalnym, w poszukiwaniu zysku dążą do jak najlepszego zaspokajania potrzeb wszystkich swoich klientów. Potrzeby te mogą być diametralnie różne nie tylko na różnych kontynentach, ale nawet na obszarze jednego państwa. W związku z tym wybierane są takie systemy, które gwarantują podejmowanie decyzji w lokalnych oddziałach firm, ale jednocześnie umożliwiają dostarczanie jak najpełniejszej informacji zarządzanej centralnie po to, aby ich działania były jak najbardziej efektywne. Naturalnym wyborem w takich sytuacjach są rozproszone systemy.

Wady i zalety systemów rozproszonych

[18]System rozproszony (ang. *distributed system)* nazywany także systemem luźno powiązanym (ang. *loosely coupled*) to zbiór niezależnych komputerów połączonych siecią komputerową, które są wyposażone w oprogramowanie umożliwiające współdzielenie zasobów systemowych między różnych użytkowników.

W tego typu systemach, oprogramowanie systemowe działa na luźno zintegrowanej grupie współpracujących procesorów połączonych siecią. Obecnie właściwie wszystkie systemy są rozproszone. Przykładami takich systemów są m.in. systemy bankomatów, systemy rezerwacji, systemy pracy grupowej. Ogromnym katalizatorem rozproszenia systemów jest Internet. W systemach tych można zauważyć tendencje do rozdzielania obliczeń między wiele procesorów. W porównaniu ze ściśle powiązanymi systemami, procesory te nie dzielą pamięci ani zegara. Każdy procesor ma natomiast własną pamięć lokalną. Procesory komunikują się za pomocą różnych linii komunikacyjnych, na przykład szybkich szyn danych lub linii telefonicznych. Procesory w systemach rozproszonych mogą się różnić pod względem rozmiaru i przeznaczenia.

[19]Mogą wśród nich być:

- małe mikroprocesory,
- stacje robocze,
- minikomputery,
- wielkie systemy komputerowe ogólnego przeznaczenia.

Na określenie tych procesorów używa się różnych nazw, takich jak:

- stanowiska (ang. *sites*),
- węzły (ang. *nodes*),
- komputery itp. – zależnie od kontekstu, w którym się o nich mówi.

Istotną rolę w systemach rozproszonych odgrywa warstwa pośrednicząca (ang. *middleware*), która pośredniczy w komunikacji pomiędzy komponentami systemów rozproszonych. Przykładami warstwy pośredniczącej są:

- gniazda (ang. *sockets*),
- RPC (ang. *Remote Procedure Call*),
- DCE (ang. *Distributed Computing Environment*),
- CORBA (ang. *Common Object Request Broker Architecture*),
- DCOM (ang. *Distributed Component Object Model*),
- RMI (ang. *Remote Method Invocation*).

[19]Zalety systemów rozproszonych:

- [19]Podział zasobów: Po połączeniu ze sobą różnych stanowisk (o różnych możliwościach) użytkownik jednego stanowiska może korzystać z zasobów dostępnych na innym. Na przykład użytkownik węzła A może korzystać z drukarki laserowej zainstalowanej w węźle B. Użytkownik węzła B może w tym samym czasie mieć dostęp do pliku znajdującego w A. Mówiąc ogólnie, podział zasobów w systemie rozproszonym tworzy mechanizmy dzielonego dostępu do plików w węzłach zdalnych, przetwarzania informacji w rozproszonych bazach danych, drukowania plików w węzłach zdalnych, zdalnego użytkowania specjalizowanych urządzeń sprzętowych (np. odznaczających się wielką szybkością procesorów tablicowych) i wykonywania innych operacji.
- [19]Przyspieszanie obliczeń (ang. *load sharing*): Jeśli pewne obliczenie da się rozłożyć na zbiór obliczeń cząstkowych, które można wykonywać współbieżnie, to system rozproszony umożliwia przydzielenie tych obliczeń do poszczególnych stanowisk i współbieżne ich wykonanie. Ponadto, jeżeli pewne stanowisko jest w danej chwili przeciążone zadaniami, to część z nich można przenieść do innego, mniej obciążonego stanowiska. Takie przemieszczanie zadań nazywa się dzieleniem obciążeń.
- [19]Niezawodność: W przypadku awarii jednego stanowiska w systemie rozproszonym pozostałe mogą kontynuować pracę. Jeżeli system składa się z dużych, autonomicznych instalacji (tzn. komputerów ogólnego przeznaczenia), to awaria jednego z nich nie wpływa na działanie pozostałych. Natomiast, gdy system składa się z małych maszyn, z których każda odpowiada za jakąś istotną funkcję (np. za wykonywanie operacji wejścia-wyjścia z końcówek konwersacyjnych lub za system plików), wówczas z powodu jednego błędu może zostać wstrzymane działanie całego systemu. Ogólnie można powiedzieć, że istnienie w systemie wystarczającego zapasu (zarówno sprzętu, jak i danych) sprawia, że system może pracować nawet po uszkodzeniu pewnej liczby jego węzłów (stanowisk).
- [19]Komunikacja: Istnieje wiele sytuacji, w których programy muszą wymieniać dane między sobą w ramach jednego systemu. Przykładem tego są systemy

okien, w których często dzieli się dane lub wymienia je między terminalami. Wzajemne połączenie węzłów za pomocą komputerowej sieci komunikacyjnej umożliwia procesom w różnych węzłach wymianę informacji. Użytkownicy sieci mogą przesyłać pliki lub kontaktować się ze sobą za pomocą poczty elektronicznej. Przesyłki pocztowe mogą być nadawane do użytkowników tego samego węzła lub do użytkowników innych węzłów.

- Współdzielenie zasobów (ang. *resource sparing*): wielu użytkowników systemu może korzystać danego zasobu (np. drukarek, plików, usług, itp.)
- [17]Otwartość (ang. *openness*): jest ona definiowana jako zdolność systemu do dołączania nowego sprzętu, oprogramowania i usług – najlepiej na platformach sprzętowych i systemach operacyjnych dostarczanych przez różnych dostawców.
- [17]Współbieżność (ang. *concurrency*): w systemie rozproszonym wiele procesów może działać w tym samym czasie na różnych komputerach w sieci. Procesy te mogą komunikować się podczas swego działania.
- [17]Skalowalność (ang. *scalability*): Moc i możliwości przetwarzania może wzrastać w miarę dodawania do systemu nowych zasobów, w szczególności komputerów. W praktyce skalowalność jest często ograniczona poprzez przepustowość sieci oraz (niekiedy) poprzez np. specyficzne protokoły wymiany informacji. Niemniej skalowalność systemu rozproszonego jest nieporównywalnie lepsza w stosunku do systemu scentralizowanego.
- [17]Odporność na błędy (ang. *fault tolerance*): Dostępność wielu komputerów oraz umożliwienie zdublowania informacji (replikacje) oznacza, że rozproszony system jest tolerancyjny w stosunku do pewnych błędów zarówno sprzętowych jak i programowych – np. awaria węzła komunikacyjnego powoduje wygenerowanie innej trasy przepływu informacji.
- [17]Transparentność, przezroczystość (ang. *transparency*): Oznacza ukrycie przed użytkownikiem szczegółów rozproszenia, np. gdzie ulokowane są zasoby lub jak są one fizycznie zaimplementowane, pod jakim systemem pracują, itd. Przezroczystość ma zasadnicze znaczenie dla komfortu działania użytkownika oraz dla niezawodności budowanego oprogramowania. Niekiedy, np. dla celów optymalizacyjnych, użytkownik może zrezygnować z pełnej przezroczystości. Przykładem przezroczystości jest Internet: klikając w aktywne pole na stronie WWW nie interesujemy się, gdzie znajduje się odpowiadająca mu strona, oraz jak i na czym jest zaimplementowana.

Wady systemów rozproszonych:

- [17]Złożoność: Systemy rozproszone są trudniejsze do zaprogramowania i do administrowania niż systemy scentralizowane. Zależą od własności sieci, np. jej przepustowości i czasu transmisji, co utrudnia zaprojektowanie i zrealizowanie wielu algorytmów i procesów przetwarzania.
- [17]Ochrona: Dla systemu scentralizowanego wystarcza w zasadzie strażnik z karabinem. System rozproszony nie może być chroniony w ten sposób, przez co może być narażony na różnorodne ataki (włamania, wirusy, sabotaż, odmowa płatności, itd.) z wielu stron, które trudno zidentyfikować.
- [17]Zdolność do zarządzania: jest ona utrudniona wskutek tego, że konsekwencje różnych działań administracyjnych w systemie rozproszonym są trudniejsze do zidentyfikowania – podobnie z przyczynami sytuacji anormalnych, w szczególności awarii.
- [17]Nieprzewidywalność: system rozproszony jest nieprzewidywalny w swoim działaniu, ponieważ zakłócenia mogą być powodowane przez wiele przyczyn: małą przepustowość i awarię łączy, awarię komputerów, zbyt duże obciążenie danego serwera, lokalne decyzje administracji serwera, itd.; Identyfikacja zasobów: zasoby są podzielone pomiędzy wiele komputerów, w związku, z czym schematy ich nazywania muszą być zaprojektowane tak, aby użytkownicy mogli

zidentyfikować interesujące ich zasoby. Przykładem takiego schematu jest URL znany z WWW.

- [17]Jakość obsługi: odzwierciedla wydajność systemu, jego dostępność i niezawodność. Podlega ona wielu czynnikom, w szczególności, przypisaniu zadań do procesorów, optymalności geograficznego podziału danych, itd.
- [17]Architektura oprogramowania: opisuje ona, w jaki sposób funkcjonalności systemu są przypisane do logicznych i fizycznych komponentów systemu. Wybór dobrej architektury przesądza o spełnieniu kryterium jakości obsługi.

Wady i zalety systemów rozproszonych – zestawienie analityczne [17]

Zalety	Wady
• Podział zasobów: system rozproszony pozwala dzielić zasoby sprzętowe i programowe pomiędzy wielu użytkowników pracujących na różnych komputerach pracujących w sieci. • Przyspieszanie obliczeń (dzielenie obciążenia) • Niezawodność: awaria jednego urządzenia nie uniemożliwiać działania systemu • Komunikacja wzajemne połączenie węzłów za pomocą komputerowej sieci komunikacyjnej umożliwia procesom w różnych węzłach wymianę informacji. • Współdzielenie zasobów: wielu użytkowników systemu może korzystać danego zasobu • Otwartość: podatność na rozszerzenia, możliwość rozbudowy systemu zarówno pod względem sprzętowym, jak i oprogramowania • Współbieżność: zdolność do przetwarzania wielu zadań jednocześnie • Skalowalność: własność systemu polegająca na zachowaniu podobnej wydajności systemu przy zwiększeniu skali systemu • Odporność na błędy: własność systemu polegająca na zdolności działania systemu mimo pojawiania się błędów (np. poprzez utrzymywanie nadmiarowego sprzętu) • Przezroczystość: własność systemu, pozwalająca na postrzeganie systemu przez użytkownika jako całości, a nie poszczególnych składowych.	• Złożoność: systemy rozproszone zależą od własności sieci • Ochrona: konieczność wprowadzania mechanizmów ochrony (dla systemu scentralizowanego w zasadzie wystarczy strażnik z karabinem) • Zdolność do zarządzania: jest utrudniona wskutek konsekwencji różnych działań administracyjnych w systemie tym są trudniejsze do zidentyfikowania. • Nieprzewidywalność: system ten jest nieprzewidywalny w swoim działaniu ponieważ zakłócenia mogą być powodowane przez wiele przyczyn, np. zbyt duże obciążenie danego serwera • Jakość obsługi: odzwierciedla wydajność systemu, jego dostępność i niezawodność. Podlega ona wielu czynnikom, w szczególności, przypisaniu zadań do procesorów, optymalności geograficznego podziału danych, itd. • Architektura oprogramowania: wybór dobrej architektury przesądza o spełnieniu kryterium jakości obsługi.

CZĘŚĆ I. Część teoretyczna

1. Rozproszony system plików

Typowy rozproszony system plików działa w oparciu o model przetwarzania klient-serwer. Sieć może również składać się z kilku serwerów, do których dostęp mają różni użytkownicy sieci. [20] Architektura sieciowa (*peer-to-peer*) pozwala na współdzielenie zasobów komputerowych i usług przez bezpośrednią ich wymianę. Architektura P2P tym różni się od architektury typu klient-serwer, że nie potrzebuje serwera, a do komunikacji potrzebne są dwie lub więcej jednostki, najczęściej komputery PC. Użytkownicy mogą udostępniać innym użytkownikom swoje zasoby, które od tej chwili będą widoczne dla innych użytkowników, tak jak gdyby były ich lokalnymi dyskami. W zależności od przyjętego modelu można stosować różne definicje rozproszonego systemu plików. Do tej pory system był uważany za rozproszony, jeśli klienci mieli dostęp do katalogów znajdujących się na dowolnych serwerach w danej sieci (np. na serwerach znajdujących się w innym gmachu firmy). W takim przypadku jednak użytkownik musiał znać zawartość zasobów poszczególnych serwerów. Współczesne sieciowe systemy operacyjne posiadają dodatkowe możliwości np. usługi katalogowe i replikację – które zmniejszają zaangażowanie użytkowników w czasie dostępu do danych zgromadzonych na różnych serwerach.

Ewolucja systemu NetWare firmy Novell odsłania zjawisko odchodzenia od modelu serwera plików w stronę rozproszonego systemu plików. Poprzednie wersje NetWare dawały użytkownikom dostęp do katalogów na każdym serwerze w danej sieci. Wzbogacenie systemu o usługi katalogowe NDS (ang. *Novell Directory Services*) pozwoliło administratorom sieci na tworzenie hierarchicznych struktur (w tym wypadku drzewa), w których użytkownicy widzą pliki z rozproszonych serwerów w odpowiednich gałęziach. Dzięki temu nie ma już konieczności wyszukiwania serwerów w sieci – wystarczy rozwinąć gałąź drzewa katalogu (ang. *Directory Tree*), obejmującą poszukiwane dane.

1.1. Cechy współczesnych rozproszonych systemów plików

Rozproszony system plików umożliwia klientom korzystanie z zasobów bez względu na ich położenie. Poprzedni model uwzględniał, co prawda, istnienie kilku serwerów, ale użytkownicy mieli problemy z odszukiwaniem żądanych plików w tych zasobach. Usługi katalogowe umożliwiają scalanie i grupowanie zasobów w strukturze hierarchicznej, rozbitej na gałęzie w sposób zgodny z logiką działania przedsiębiorstwa i zgodny z oczekiwaniem użytkowników. Można np. utworzyć gałąź o nazwie Kadra i wrzucić do niej odnośniki do katalogów na wybranych hostach, zawierających dokumenty dotyczące pracowników. Tego rodzaju udogodnienia oferują m.in. systemy DFS (ang. *Distributed File System)* firmy Microsoft i NDS (ang. *Novell Directory Services*) firmy Novell. Współczesny rozproszony system plików powinien również charakteryzować się replikacją danych. Użytkownicy korzystający z danych zgromadzonych na określonym serwerze z różnych miejsc sieci potrzebują replikowania (powielanie) plików z jednego serwera na inny umieszczony bliżej. Replikacja przyczynia się także do zmniejszenia ruchu w sieci, ponieważ rozkłada obciążenie na kilka hostów. Rozproszone systemy plików automatycznie przekazują żądanie dostępu do zbioru do tego serwera, który w danej chwili to żądanie obsłuży najsprawniej.

Kolejną cechą wymaganą dla rozproszonych systemów plików, jest możliwość jednorazowego logowania klienta. Dzięki niej użytkownik nie musi podawać hasła za każdym razem, gdy chce skorzystać z serwera innego niż ten, z którego korzystał dotychczas. Wśród dodatkowych cech wspomnieć należy o szyfrowaniu danych. Jeśli klient przetwarza poufne dane, przechowywane na bezpiecznym serwerze, to dane w czasie przesyłania powinny być zaszyfrowane, zwłaszcza jeśli do transmisji wykorzystujemy sieć Internet.

W każdym systemie umożliwiającym współdzielenie plików, występują problemy jednoczesnego dostępu wielu użytkowników jednego zbioru. Nieodzowne jest

zastosowanie mechanizmów rozwiązujących tą kwestię. Mechanizmy te działają w oparciu o następujące zasady:

- Możliwość jednoczesnego odczytu bez możliwości zapisywania. Wszyscy użytkownicy mogą mieć dostęp do zawartości zbioru, nie mogą jednak dokonywać edycji. Mechanizm prosty w realizacji.
- Kontrolowanie zapisu. Wielu użytkowników otwiera jeden plik, ale tylko jeden z nich ma możliwość zapisu zmienionego dokumentu. Zmiany wprowadzone w dokumencie przez tego klienta najczęściej pozostają niezauważone dla innych użytkowników, oglądających treść danego pliku.
- Jednoczesne zapisywanie. Wymaga intensywnego nadzoru ze strony systemu operacyjnego. Zasada ta daje możliwość jednoczesnego zapisu i odczytu tego samego zbioru przez kilku użytkowników. System zapobiega tu nadpisywaniu danych. Gwarantuje użytkownikom aktualizację na bieżąco informacji. W wielu implementacjach sposób ten może okazać się nie do przyjęcia, ze względu na duże koszty mocy obliczeniowej i obniżenie przepustowości sieci.

Sposób obsługi jednoczesnych zapisów wyróżnia poszczególne systemy plików. Użytkownik żądając od hosta danego pliku powoduje umieszczenie dokumentu w cache'u swojego komputera. Jeśli inny użytkownik zażąda tego samego dokumentu, również w jego stacji umieszczona zostanie kopia. Jeśli obaj klienci modyfikują dokument to istnieją trzy jego wersje; po jednej na stacjach użytkowników i jedna na serwerze. W tym momencie potrzebny jest jeden z systemów zapewniających synchronizację poszczególnych wersji:

- Typ *stateless*. W systemie tego rodzaju serwer nie trzyma informacji o plikach przechowywanych w pamięciach podręcznych stacji końcowych. Klienci są zobowiązani do cyklicznej komunikacji z serwerem i sprawdzania, czy inni klienci zapisali zmienionego pliku. Sytuacja ta prowadzi to do zwiększenia ruchu w sieciach. Sposób ten daje jednak na ogół zadowalające rezultaty w małych sieciach LAN. Przykładem zastosowania klasy *stateless* jest NFS.
- Typ *callback*. W tym przypadku serwer zbiera informacje o działaniach użytkowników i o dokumentach przechowywanych u nich lokalnie w pamięci podręcznej. Użytkownicy są informowani o modyfikacjach plików poczynionych przez innych klientów. Sposób ten eliminuje zwiększenie obciążenia w sieci. Systemem wykorzystującym ten mechanizm jest AFS. Po zapisaniu zmian w zbiorze przez jednego z użytkowników, system „oddzwania" pozostałych klientów i informuje o wystąpieniu zmian.

W pierwotnej wersji obu systemów, pierwszy z nich (*stateless)* był bardziej wydajny, jednak AFS został wzbogacony o mechanizmy ograniczające liczbę informacji przesyłanych do użytkowników o modyfikowanych plikach. Dzięki temu udało się osiągnąć zbliżoną efektywność. Typ *callback* ma jeszcze tę zaletę, że użytkownicy są zapewniani o aktualności pliku, tzn. że jeżeli pliki który jest otwarty przez użytkownika jest związany jest obietnicą „oddzwonienia" i informowania o jego zmianach, to użytkownik ma pewność, że edytowany przez niego plik ma aktualną wersję i będzie aktualny do chwili pozyskania od serwera komunikatu zwrotnego o wprowadzonych zmianach.

1.2. Najczęściej stosowane systemy plików rozproszonych

1.2.1. NFS

NFS (ang. *Network File System*) został pierwotnie opracowany przez firmę Sun Microsystems. Ponieważ Sun opublikował kod protokołu, wiele firm produkujących oprogramowanie włączyło NFS'a do swoich systemów operacyjnych. Tym sposobem NFS stał się standardem na różnych platformach, chociaż nie był przez nikogo nigdy narzucony. Ważną cechą systemu NFS jest bezstanowy serwer. Oznacza to, że serwer NFS eksportujący katalogi nie zapamiętuje żadnej informacji o stacjach klienckich, a zajmuje się wyłącznie operacjami czytania i zapisywania. Klienci mogą być odłączni bez

wysyłania żadnych komunikatów, a serwer nie będzie próbował nawiązywać połączenia. Jest to wyjątek, ponieważ serwery z reguły próbują odzyskać połączenie z klientem. Lokalne drzewo struktury katalogów serwera jest eksportowane do klientów. System NFS niestety nie ma najlepszego modelu bezpieczeństwa, wykluczającego jednocześnie go do stosowania w tej postaci w sieciach rozległych. Systemy plików przechowywane są w bazach danych. Baza ta może być przekazywana przy pomocy NIS lub LDAP. System autofs na podstawie tej bazy danych montuje właściwą hierarchię plików. Model NFS nadaje się do wykorzystania w pracy w klastrze. Jedynym ograniczeniem może być wydajność, ale zależy to od sposobu dostępu do plików. Więcej na http://www.sun.com/.

1.2.2. AFS

AFS (ang. *Andrew File System*) budową przypomina NFS. Stworzono na uniwersytecie Carnegie Mellon. AFS ma przewagę nad typowymi sieciowymi systemami plików, pod względem skalowalności i bezpieczeństwa. Produkcyjne instancje AFS obsługują nawet do 50 tys. klientów. Buforowanie po stronie klienta przyczynia się do zwiększenia wydajności systemu i umożliwia ograniczone funkcjonowania w przypadku awarii serwera lub sieci. Więcej na http://www.psc.edu/general/filesys/afs/afs.html.

1.2.3. OpenAFS

Jest to zarazem sieciowy i rozproszony system plików. Ma możliwość udostępniania plików również w sieciach WAN i oferuje globalny obszar nazw. OpenAFS jest całkowicie wirtualny. Przechowywany jest na replikowanych serwerach danych. Klienci tworzą sobie pamięć podręczną złożoną z ostatnio używanych plików, które były regularnie przekazywane do serwerów danych. Aplikacje korzystają z tych zmagazynowanych danych i pracują nadal nawet podczas awarii serwera danych – muszą jednak poczekać na ponowne uruchomienie serwera w celu zachowania otwartego pliku. AFS wywodzi się z DFS (ang. *Distributed File System*), opracowanego przez firmę kontrolowaną przez IBM. Obecnie DFS nie jest już wspierany i zakończono proces jego rozwoju. Jest to doskonały system plików (z udostępnianiem plików) dla uczelni czy nawet połączeń z całym światem, gdyż jego model zabezpieczeń został bardzo starannie opracowany. Domenę administracyjną lub obiekty sieci nazwano komórkami. Użytkownicy autoryzują się sami – wysyłany jest do nich token o określonym czasie żywotności. Token umożliwia menedżerowi pamięci podręcznej, umieszczonemu na komputerze lokalnym, komunikowanie się z serwerem plików AFS. W OpenAFS administrator może dołożyć dodatkową pamięć masową lub wymienić ją na inną, bez konieczności przerywania pracy w sieci. Ponieważ menedżer pamięci podręcznej znajduje się pomiędzy użytkownikiem i serwerem, przesyłanie danych do innych serwerów może być wykonywane przezroczyście. Istnieje także specjalna wersja OpenAFS o nazwie MultiResident-AFS, która umożliwia automatyczne, nieprzerwane przesyłanie danych nawet do pamięci masowej, która czasowo jest niedostępna (w takiej sytuacji to przesyłanie jest oczywiście wirtualne). System OpenAFS nie jest zalecany, jeżeli potrzebujemy wysokiej przepustowości danych w klastrze. Zaleta pamięci podręcznej, zwiększająca stabilność systemu, jest jednak okupiona wolnym czasem dostępu do serwera plików, działającego jako proces użytkownika.

1.2.4. GPFS

GPFS (ang. *General Parallel File System* – czyli powszechny równoległy system plików) jest produktem komercyjnym firmy IBM. GPFS jest prawdziwym równoległym systemem plików. Dane mogą być rozproszone na wiele dysków, a każdy z węzłów może uzyskać dostęp do tego samego pliku w tym samym czasie. Możliwe są dwie konfiguracje dostępu: sieć SAN lub bezpośrednia pamięć masowa. W konfiguracji SAN każdy węzeł widzi wszystkie elementy magazynu danych udostępnione w GPFS. Pliki zestawiane są z bloków rozproszonych w sieci SAN i są bezpośrednio dostępne dla każdego z węzłów. Konfiguracja bezpośrednia opiera się na szybkiej sieci wewnątrz węzła np. Ethernet lub Myrinet. Bloki z lokalnych dysków twardych są rozpraszane na inne węzły. Pliki zestawiane są przy pomocy bloków pobieranych przez sieć IP z lokalnych dysków twardych dostępnych węzłów. Ten sam model wykorzystywany jest także przez system PVFS. Zarządzanie systemem jest bardzo proste. Polecenia można wydawać z dowolnego

węzła. System ma możliwość podłączania lub odłączania dysków, równomiernej dystrybucji dostępu do danych, zmiany rozmiaru bloku czy liczby możliwych i-węzłów. System nie posiada ograniczenia maksymalnego rozmiaru systemu plików, gdyż umożliwia konfigurację rozmiaru bloku maksymalnie do 1 MB. Na każdy otwarty plik przypada jeden węzeł obsługujący metadane. Wszystkie węzły mogą otrzymać dostęp do tego samego pliku, ale zmiany w metadanych obsługiwane są przez węzeł metadanych. Do węzłów próbujących uzyskać dostęp do danego pliku przekazywane są blokady. Dane zapisane i odczytywane równolegle skalują się liniowo wraz z liczbą dysków i węzłów. Systemy plików GPFS mogą być eksportowane z użyciem AFS lub NFS do serwerów specjalizowanych. Następnie poszczególne węzły obliczeniowe montują wyeksportowane systemy plików. GPFS jest uzależniony od bardzo kosztownej infrastruktury SAN, która umożliwia wysoką wydajność tego systemu. Konfiguracja, w której sieć IP wykorzystywana jest do zestawiania rozproszonych kawałków danych, może stać się szybko wąskim gardłem ograniczającym dostęp do danych.

1.2.5. LUSTRE

[21] [22] Lustre jest obecnie aktywnie rozwijanym systemem. Mimo że powstał on pod szyldem HP, projekt został przekazany na licencji Open Source. Lustre posiada doskonałą dokumentację. Do konfiguracji i logowania wykorzystuje standardy LDAP i XML. Jest to system plikowy (obiektowy). Dane przechowywane w Lustre są traktowane jak obiekty. Obiektami systemu plików są (specjalne) pliki i katalogi. Właściwości, czyli metadane tych obiektów (rozmiar, czas utworzenia, wskaźniki dowiązań symbolicznych czy flagi rezerwowe), przechowywane są na serwerach metadanych (MDS). Metadane przechowywane są oddzielnie w stosunku do rzeczywistej zawartości obiektów. Serwery MDS obsługują tworzenie plików, zmiany ich właściwości i są odpowiedzialne za obsługę obszaru nazw: plik może być odnaleziony poprzez wysłanie zapytania do serwera MDS. Po otworzeniu kluczy MDS, wymianą danych zajmują się Object Storage Targets (OST) (adresaci przechowywania obiektów). Serwer MDS śledzi wymianę danych za pomocą rejestru. Tworzenie i zapisywanie pliku wymaga utworzenia i-węzła na serwerze MDS, który następnie skontaktuje się z OST w celu przydzielenia miejsca na dysku. W celu zwiększenia wydajności alokację można rozproszyć na kilka OST. Przepustowość osiągana w opublikowanych testach jest imponująca. Jednak nadal brakuje dla Lustre pewnych istotnych narzędzi serwisowych dla środowiska produkcyjnego; nie ma narzędzia odzyskiwania systemu plików i nie istnieje na razie możliwość poprawnej pracy serwera MDS pomimo usterek. Oryginalna metoda i prace rozwojowe prowadzone od podstaw stworzyły z systemu Lustre rozwiązanie, które może być bazą potężnego, a jednocześnie eleganckiego systemu.

1.2.6. PVFS

[21] PVFS (ang. *Parallel Virtual File System* – czyli równoległy wirtualny system plików) działa w oparciu o bloki i zapewnia wysoką wydajność aplikacji rozproszonych lub równoległych. Standardowe środowisko aplikacji jest niewielkim klastrem (poniżej 50 węzłów), ale nie ma żadnych ograniczeń. Węzłom wejścia-wyjścia w PVFS udostępniana jest część dysków wewnętrznych. Przestrzeń plików tych dysków jest następnie rozpraszana na cały klaster, a dostęp do niego można uzyskać przez sieć Ethernet, moduł jądra oraz bibliotekę libpvfs zainstalowaną na komputerach-klientach. Klienci mogą być węzłami IO (IOD), a jeden z nich musi być skonfigurowany jako węzeł metadanych (MDS). Pliki rozpraszane są na węzły IO. Po wstępnej wymianie danych zarządzających z serwerem MDS cały ruch danych wewnątrz węzłów IO jest obsługiwany indywidualnie przez klientów przy pomocy biblioteki libpvfs. Dzięki niej aranżowana jest procedura składania plików z rozproszonych elementów. System PVFS obsługuje Myrinet i Infiniband w komunikacji wewnątrzwęzłowej. PVFS nie zawiera obecnie środków dla redundancji danych. Odtworzenie uszkodzonego węzła jest również niemożliwe. Jest to potencjalny problem na poziomie zarządzania, gdyż liczba węzłów zwiększa się z czasem. System PVFS nie jest w stanie przekroczyć ograniczeń protokołu TCP/IP w Linuksie (np. ograniczenia liczby jednocześnie otwartych gniazd systemu). System PVFS musi być zainstalowany na wszystkich węzłach klastra, gdyż nie umożliwia on eksportowania przez

NFS czy AFS. Na podstawie doświadczeń zebranych w systemie PVFS1 stworzono od początki kod PVFS2. Nowa wersja posiada rozszerzenia strukturalne, typu rozpraszanie sterowane przez użytkownika oraz dystrybucja metadanych. Umożliwia to instalację więcej niż jednego kontrolera metadanych, dzięki czemu można uniknąć wad występujących we wcześniejszej wersji.

1.2.7. OpenGFS

[21] [22] OpenGFS, znany także pod nazwą OGFS, stosuje raportowany system plików oparty na blokach, który zapewnia dostęp do zapisu i odczytu wielu węzłom. Okropny kod „zbiorczy" został zmieniony i obecnie w OGFS można korzystać z dowolnego menedżera dysków logicznych. Zalecamy program Elom. Całkiem niedawno zastąpiono też memexp modułem OpenDLM. Poprzednia wersja memexp była głównym „punktem zapalnym" wszystkich problemów i wymagała sporych zasobów obliczeniowych. System OGFS umożliwia rozrastanie się systemów plików oraz dołączanie nowych dysków twardych (poprzez osobny LVM). Uszkodzenia węzłów obsługiwane są przy pomocy odzyskiwania rejestru i izolowania uszkodzonego węzła.

1.2.8. DFS

[23] Od systemu Windows NT v 4 Microsoft wprowadził hierarchiczną rozproszoną wersję systemu plików DFS (ang. *Distributed File System*). DFS umożliwia tworzenie dowolnych struktur hierarchicznych, oraz grupowanie zasobów zlokalizowanych w dowolnych częściach sieci korporacyjnej. Więcej na http://www.microsoft.com.

1.2.9. NCP

[24] NCP (ang. *NetWare Core Protocol*) – pakiet opracowany przez firmę Novell dla systemu NetWare. Bazowym protokołem standardu NCP jest protokół IPX, który służy do przekazywania komunikatów. NCP obsługuje takie funkcje jak: dostęp do plików, blokowanie plików, bezpieczeństwo, śledzenie wykorzystania zasobów itp. Więcej na http://www.novell.com.

Specyfikacja Internetu wymagała opracowania innych protokołów dostępu do plików. W typowym modelu klient po połączeniu z serwerem WWW klient przechowuje kopie strony w pamięci podręcznej. Odczytanie całej strony wymaga częstego łączenia się z serwerem. Osobno pobierany jest tekst dokumentu osobno grafika. Rozproszone systemy plików opracowane na potrzeby sieci Internet są bardziej wydajne, ponieważ umożliwiają pobranie całej strony za podczas jednego połączenia. Najbardziej rozpowszechnionym systemami zbudowanymi na potrzeby Internetu są WebNFS firmy Sun Microsystems oraz CIFS firmy Microsoft.

Pierwszy z nich działa w oparciu o NFS, jest optymalizowany jest do pracy w sieciach Internet i Intranet oraz posiada ulepszone w porównaniu do pierwowzoru mechanizmy dot. Bezpieczeństwa. Produkt Microsoftu powstał na bazie SMB i również poprzez dodanie mechanizmów bezpieczeństwa zoptymalizowany do zastosowań Internecie i intranecie.

Systemy obecnie budowane nie są wszechstronne i trudno znaleźć najlepszy. Żaden z zaprezentowanych nie będzie rozwiązywał wszystkich potrzeb użytkowników. Jednak dzięki dużej różnorodności istnieje możliwość implementacji najlepszego dla naszych potrzeb. Tym bardziej systemy bazujące na pakietach open source, nie są jeszcze najlepiej dopracowane. Lustre nie jest dość skalowalny natomiast system OpenGFS nie najlepiej gospodaruje przestrzenią. Oprogramowanie płatne jest stabilniejsze i bardziej skalowalne.

1.3. Przykład rozproszonego systemu plików

Jako przykład praktycznego rozwiązania problemów rodzących się przy implementacji rozwiązań transakcji rozproszonych można podać rozproszony system plików.

[25] DFS (ang. *Distributed File System*) stanowi rozproszoną implementację klasycznego modelu systemu plików z podziałem czasu, w którym wielu użytkowników współdzieli pliki i zasoby pamięciowe DFS zarządza zbiorami rozproszonych urządzeń pamięci.

Twórcom tego rozwiązania piętrzyły się problemy dotyczące zwłaszcza wzajemnego, jednoczesnego wielodostępu do danych. Każde otwarcie zasobów danych, zbioru dyskowego wiąże się z otwarciem transakcji, która monitoruje jakie działania dokonywane są na zbiorach.

Niedopuszczalnym było by dopuszczenie do sytuacji w której jednocześnie z dwu różnych miejsc dokonuje się edycji tego samego zbioru. Mogło by to powodować zapis sprzecznych informacji.

[25] Nazewnictwo i przezroczystość:

- Nazewnictwo to odwzorowanie między obiektami logicznymi a fizycznymi.
- Przezroczysty DFS ukrywa położenie pliku w sieci.
- W przypadku pliku, którego kopie znajdują się w różnych węzłach sieci, odwzorowanie tworzy zbiór lokalizacji kopii pliku; przezroczysty DFS ukrywa zarówno istnienie wielu kopii, jak i ich położenie.
- Przezroczystość położenia – nazwa pliku nie daje żadnej wskazówki na temat fizycznego położenia pliku (np. //server1/dir1/dir2, ale gdzie jest server1?):
  I. nazwa pliku oznacza określony, choć ukryty, zbiór bloków dyskowych,
  II. może ujawniać zależność między składowymi nazwy a komputerami,
  III. nie jest możliwa automatyczna zmiana położenia pliku.
- Niezależność położenia – nazwy pliku nie trzeba zmieniać wtedy, gdy plik zmienia swoje fizyczne położenie:
  IV. lepsza abstrakcja pliku (nazwa określa zawartość, nie położenie),
  V. oddziela hierarchię nazw od hierarchii urządzeń pamięci.

1.4. [26] Schowki w rozproszonych systemach plików

[26] Schowki w łatwy sposób udało się zaimplementować w rozproszonych systemach plików działających w architekturze klient-serwer (np. NFS, AFS). Każda stacja kliencka może przechowywać często wykorzystywane pliki w lokalnej pamięci dyskowej lub operacyjnej. Powoduje to:

- zmniejszenie ruchu w sieci. Nie ma potrzeby przesyłania całych plików. Pojawiają się ewentualnie pakiety weryfikujące aktualność lokalnej kopii pliku,
- szybszy dostęp do danych. Nie występują opóźnienia wynikające z przesyłania danych siecią,
- zmniejszenie obciążenia serwera,
- większą skalowalność i wydajność systemu plików,
- możliwość pracy w sytuacji, gdy serwer jest niedostępny i wszystkie operacje dokonywane są na plikach znajdujących się w schowku (system Coda).

[26] Schowki powodują także pewne problemy takie jak:

- zachowanie spójności danych wynikające z istnienia wielu kopii,
- pogorszenie bezpieczeństwa danych. Dane znajdują się na komputerach klientów, gdzie użytkownicy mogą mieć inne prawa niż na serwerze, w szczególności mogą mieć prawa super użytkownika. Umożliwia to nieautoryzowany dostęp do danych,
- potrzeba posiadania większej ilości pamięci operacyjnej lub/i dyskowej przeznaczonej na schowek,
- obsługa schowków, która jest dodatkowym obciążeniem systemu plików, może dać skutek odwrotny do zamierzonego i doprowadzić do spadku wydajności. Zastosowanie złej polityki zarządzania schowkiem lub pojawienie się specyficznych

danych może spowodować sytuację, w której klient przed odwołaniem się do pliku sprawdza zawartość nieaktualnego schowka, zwiększając czas dostępu do danych,

- zmiana semantyki operacji na plikach. Jest to konsekwencją rozproszonej natury operacji. Przykładem są zestaw operacji open/close w AFS i operacje write/read w NFS.

1.5. [26] Schematy tworzenia nazw

Nazwa pliku składa się z nazwy komputera macierzystego i nazwy lokalnej; gwarantowana jednoznaczność w całym systemie.

Zdalne katalogi są montowane w lokalnym katalogu tworząc spójne drzewo katalogów; dostęp przezroczysty jedynie do wcześniej zamontowanych katalogów (np. NFS).

Pełna integracja składowych systemów plików:

VI. jedna globalna struktura nazw obejmuje wszystkie pliki w systemie,

VII. jeśli serwer jest niedostępny, to pewien zbiór katalogów też staje się niedostępny (np. Locus, Sprite, Andrew).

1.6. [25] Semantyka współdzielenia pliku

- [25] Semantyka Unixa: system wymusza porządkowanie wszystkich operacji w czasie i zawsze przekazuje najbardziej aktualną zawartość.
- Semantyka sesji: zmiany w otwartym pliku są początkowo widoczne tylko w procesie dokonującym modyfikacji. Inne procesy zauważą zmiany dopiero po zamknięciu pliku.
- Pliki niemodyfikowalne: nie można otworzyć pliku do zapisu, jedynie do odczytu i do tworzenia (zamiast modyfikowania pliku, trzeba utworzyć go od nowa pod tą samą nazwą – ta operacja jest atomowa).
- Transakcje: wszystkie zmiany mają własność .wszystko albo nic. (np. system bankowy).

1.7. [25] Zdalny dostęp do plików

Przechowywanie ostatnio używanych bloków dyskowych w podręcznej pamięci buforowej pozwala zmniejszyć ruch w sieci:

VIII. jeśli potrzebnych danych nie ma w pamięci podręcznej, to sprowadza się ich kopię z serwera,

IX. klient korzysta z kopii przechowywanej w pamięci podręcznej,

X. pliki identyfikuje się z kopią główną w serwerze, ale w różnych pamięciach podręcznych w sieci mogą przebywać.

1. Problem utrzymania spójności pamięci podręcznych, tzn. zgodności kopii podrzędnych z kopią główną.
2. Gdzie przechowywać pliki: dysk serwera, pamięć główna serwera, dysk klienta, pamięć główna klienta.
3. Zalety dyskowych pamięci podręcznych:

   XI. niezawodność (nie przepadają podczas awarii),

   XII. dane przechowywane w pamięci podręcznej na dysku pozostają tam podczas rekonstrukcji systemu po awarii i nie trzeba ich ponownie sprowadzać.
4. Zalety pamięci podręcznej w pamięci głównej:

   XIII. umożliwiają korzystanie z bezdyskowych stacji roboczych,

XIV. krótszy czas dostępu do danych,

XV. pamięci podręczne po stronie serwera są w pamięci głównej niezależnie od tego, gdzie przechowuje się pamięci podręczne klienta; jeśli przechowuje się je w pamięci głównej, to można zastosować pojedynczy mechanizm obsługi pamięci podręcznej po stronie serwera i klienta.

5. [25] Aktualizowanie danych w pamięci podręcznej:

   XVI. natychmiastowe pisanie (ang. *write-through*) – przesyła się dane do serwera natychmiast po umieszczeniu ich w pamięci podręcznej. Niezawodne, ale słaba wydajność,

   XVII. opóźnione pisanie (ang. *delayed-write*) – modyfikacje zapisuje się w pamięci podręcznej i później przesyła do serwera; zawodne

   - wariant: przegląda się pamięć podręczną w regularnych odstępach czasu i wysyła do serwera bloki modyfikowane od ostatniego przeglądania (np. Sprite),
   - wariant (ang. *write-on-close*): dane przesyła się do serwera po zamknięciu pliku. Najlepsze w przypadku, gdy pliki są otwarte długo i często modyfikowane.

6. Weryfikacja aktualności danych – czy kopia lokalna w pamięci podręcznej jest zgodna z kopią główną? Weryfikację zgodności może zainicjować klient lub serwer.

7. [25] Porównanie obsługi zdalnej i pamięci podręcznej:

   XVIII. pamięć podręczna pozwala obsługiwać większość żądań zdalnego dostępu tak szybko jak żądania lokalnego dostępu,

   XIX. powoduje, że kontakt z serwerem jest rzadszy:

   - mniejsze obciążenie serwera i ruch w sieci
   - większa możliwość skalowalności

   XX. narzut związany z komunikacją poprzez sieć jest mniejszy, gdy przesyła się dane dużymi porcjami (pamięć podręczna) zamiast jako szereg odpowiedzi na specjalne żądania (obsługa zdalna),

   XXI. pamięć podręczna sprawdza się lepiej, gdy żądania pisania są rzadkie (gdy częste, duży narzut na utrzymanie zgodności),

   XXII. pamięć podręczna pozwala osiągać korzyści, gdy wykonanie odbywa się na komputerze z lokalnymi dyskami lub dużą pamięcią główną,

   XXIII. zdalny dostęp na komputerach bezdyskowych i z małą pamięcią główną trzeba realizować poprzez zdalną obsługę.

8. [25] Stanowy (ang. *stateful*) serwer plików:

   XXIV. mechanizm:

   - klient otwiera plik
   - serwer odczytuje informacje z dysku, wstawia do pamięci, przekazuje klientowi jednoznaczny identyfikator
   - klient używa tego identyfikatora podczas kolejnych dostępów
   - serwer musi odzyskać pamięć używaną przez klientów, którzy przestają być aktywni

   XXV. zwiększona wydajność:

   - mniej dostępów dyskowych
   - serwer wie czy plik otwarto do sekwencyjnego dostępu i może czytać z wyprzedzeniem następne bloki Bezstanowy (ang. *stateless*) serwer plików

XXVI. każde żądanie jest samowystarczalne, więc nie trzeba przechowywać informacji o stanie,

XXVII. każde żądanie identyfikuje plik i pozycję w pliku,

XXVIII. nie trzeba otwierać i zamykać połączenia (zbędne open i close dla pliku),

XXIX. nie trzeba przeznaczać miejsca na pamiętanie informacji o stanie,

XXX. nie ma ograniczeń na liczbę otwartych plików Różnice między serwerem stanowym i bezstanowym,

XXXI. rekonstrukcja systemu po awarii:

- serwer stanowy gubi całą informację; może ją odtworzyć prowadząc dialog z klientem lub zakończyć rozpoczęte operacje z błędem. Serwer musi wiedzieć, którzy klienci przestali działać w wyniku awarii, żeby odzyskać pamięć zajmowaną przez opis stanu tych klientów
- awaria nie ma wpływu na pracę serwera bezstanowego

XXXII. narzut jaki płaci się za mniej zawodną usługę:

- dłuższe komunikaty z żądaniami
- wolniejsze przetwarzanie żądań
- dodatkowe ograniczenia na projekt DFS (np. trudno zrealizować blokowanie plików)

XXXIII. niektóre .środowiska wymagają usługi z pamiętaniem stanu (np. użycie w Unixie deskryptorów plików i niejawnych pozycji w pliku wymaga przechowywania informacji o stanie).

9. [25] Tworzenie kopii pliku (ang. *file replication*):

XXXIV. zwiększa dostępność i może skrócić czas dostępu,

XXXV. umożliwia uniknięcie sytuacji, gdy pojedynczy serwer staje się wąskim gardłem,

XXXVI. istnienie wielu kopii powinno być niewidoczne na wyższych poziomach; na niższych poziomach kopie muszą się różnić nazwami,

XXXVII. aktualizacja jednej kopii powinna być przeprowadzona również na pozostałych kopiach,

XXXVIII. kopiowanie na żądanie – czytanie zdalnej kopii powoduje zapamiętanie jej w pamięci podręcznej, a więc utworzenie lokalnej kopii,

XXXIX. protokół aktualizacji:

- aktualizacja pliku powoduje wysłanie komunikatu do serwera kopii głównej, który następnie wysyła komunikaty do serwerów kopii podrzędnych (gdy serwer kopii głównej ulegnie awarii, to wszelkie aktualizacje przestają być możliwe),
- głosowanie – klienci muszą otrzymać od serwerów pozwolenie na czytanie lub zapis pliku z wieloma kopiami.

1.8. Podsumowanie

Główną rolą rozproszonych systemów plików jest umożliwienie rozproszenia danych na wielu lokalizacjach fizycznych (serwerach) przy jednoczesnym scaleniu lokalizacji logicznej. Skutkuje to tym, że pliki umieszczone na różnych serwerach plików dla użytkownika otoczenia sieciowego mogą być one widoczne na jednym serwerze plików, tyle że np. w wielu katalogach. Poszczególne katalogi mogą się znajdować wiele kilometrów od siebie mimo że wyglądają że znajdują się na jednym serwerze. Zaletą tego typu rozwiązań jest rozproszenie zasobów po wielu komputerach przy jednoczesnym

łatwym użytkowaniu zupełnie przezroczystym dla klientów. Dzięki temu dane nie są tak narażone na ryzyko awarii jednego serwera oraz pozwala to rozładować obciążenie poszczególnych serwerów poprzez tworzenie łączy wskazujących do różnych komputerów, wtedy klient zostanie skierowany do najbliższego.

Zaletą budowania rozproszonych systemów plików jest:

- Dzielenie zasobów: Możliwość dzielenia zasobów pomiędzy wielu użytkowników pracujących na różnych komputerach w sieci.
- Otwartość: Możliwość dołączania nowego sprzętu, oprogramowania i usług.
- Skalowalność: możliwości przetwarzania wzrasta w miarę dodawania do systemu nowych zasobów
- Tolerancja na błędy: zdublowanie informacji oznacza, że rozproszony system jest tolerancyjny w stosunku do pewnych błędów zarówno sprzętowych jak i programowych. Awaria węzła sieci skutkuje opracowanie nowej trasy transmisji.
- Przezroczystość: Możliwość ukrycia przed użytkownikiem szczegółów rozproszenia, czyli gdzie ulokowane są zasoby lub jak są one fizycznie zaimplementowane, pod jakim systemem pracują, itd.

Wady:

- Złożoność: Systemy rozproszone są trudniejsze do zaprojektowania, implementowania i administrowania niż systemy centralne. Ich prawidłowe działanie jest w dużej mierze zależne od stanu łącz, przepustowości, obciążenia serwera czy od prawidłowego funkcjonowania komputerów. To sprawia, że system taki jest nieprzewidywalny w działaniu i trudniejszy w administracji i ochronie przed atakami.

2. Transakcje rozproszone

2.1. Co to jest transakcja

Transakcja to zbiór operacji na bazie danych, które mogą być wykonane jedynie wszystkie lub żadna, czyli transakcja albo wykona się poprawnie albo wcale. Transakcje są stale używane, ale użytkownik może nie być świadomy ich użycia.

[27] Transakcja umożliwia powrót do stanu sprzed rozpoczęcia jej działania po wystąpieniu dowolnego błędu. Jest to podstawowa technika zwiększenia niezawodności oprogramowania.

Transakcje są tworzone przy zachowaniu następujących czterech własności, które złożone razem zwane są własnościami ACID: [28]

- [A] Niepodzielność (ang. *atomicity*) – w ramach jednej transakcji wykonują się albo wszystkie operacje, albo żadna. Jeżeli jedno z poleceń nie powiedzie się, wszystkie polecenia będące częścią transakcji zakończą się niepowodzeniem.
- [C] Spójność (ang. *consistency*) – o ile transakcja zastała bazę danych w spójnym stanie, po jej zakończeniu stan bazy danych jest również spójny. Zmiany wykonane przez transakcję są spójne z jednego stanu w drugi.
- [I] Izolacja (ang. *isolation*) – transakcja nie wie nic o innych transakcjach i nie musi uwzględniać ich działania, nie zachodzi w reakcję z innymi transakcjami w bazie danych.
- [D] Trwałość (ang. *durability*) – po zakończeniu transakcji jej skutki są na trwale zapamiętane i nie mogą być odwrócone przez zdarzenia losowe. Jeżeli wystąpi awaria zasilania i serwer bazy danych ulegnie awarii, istnieje gwarancja ze transakcja będzie kompletna po ponownym uruchomieniu serwera.

Ze względu na to, że transakcje operować mogą na różnorakich (z punktu widzenia biznesowego) danych, należy przewidzieć jakie powinny ustalać dla modyfikowanych i czytanych danych poziomy dostępności dla innych.

W przypadku gdy podczas wykonywania transakcji, żadne inne transakcje nie mogą modyfikować danych, które są zmieniane przez tę transakcję, aż do czasu gdy użytkownik zdecyduje że zmiany są trwałe. Gdy użytkownik modyfikuje dane zakłada blokady zapewniające wyłączność (ang. *exclusive lock*) danych z którymi pracuje.

Odwrotnie, nie można czytać danych innej transakcji, jeżeli wykonuje ona działania modyfikujące dane. Użytkownik żąda w tej sytuacji blokady współdzielonej (ang. *shared lock*) na tych danych, ale inna transakcja używa blokady wyłącznej na tych samych danych i powoduje że nikt inny nie może czytać tych danych.

Jednakże są sytuacje i dane, które nawet używane przez transakcje mogą być cały czas dostępne dla innych. Można sobie wyobrazić zasób w którym przetrzymujemy pewne współczynniki, stałe, parametry, słowniki wyłącznie czytane w ramach transakcji. Do takich danych w ramach zapewnienia ciągłości dostępu, jeśli mamy pewność co do ich niezmienności – można zastosować odpowiedni sposób dostępu – np. tzw. "brudne czytanie" (ang. *uncommitted read*).

2.2. Wprowadzenie do danych dzielonych

Serwer obudowuje dostępne dla klientów zasoby za pomocą wywoływania operacji. Jedynym sposobem sięgania przez klientów do jego zasobów jest wywołanie jednej z operacji serwera. Zastosowanie wątków w serwerze pozwala klientom na wykonywanie współbieżnych dostępów. Serwer, mający wiele wątków musi zapewnić niepodzielność swoich operacji w sensie skutków, jakie powodują te operacje w obiektach danych serwera.

Serwery mogą poprawić współpracę z klientami, organizując czekanie jednego klienta, jeśli potrzebny mu zasób jest akurat wykorzystywany przez innego klienta. W niektórych

sytuacjach takich jak przekazywanie strumienia obiektów danych do klienta, współpraca klienta i serwera może przypominać bardziej konwersację niż proste wywołanie operacji.

Klient jest procesem podejmującym działanie, natomiast serwer to proces, który czeka na zamówienia klientów i wykonuje to, o co się go poprosi. Z jednej strony należy zapobiegać wzajemnemu zaburzaniu działań klientów, z drugiej strony klientom powinno się umożliwiać użytkowanie serwerów do dzielenia i wymiany informacji. Od niektórych serwerów wymaga się, aby utrzymywały dane na zamówienie klientów przez długie okresy – serwery takie muszą rozporządzać danymi rekonstruowalnymi, to znaczy danymi, które można odtworzyć po uszkodzeniu serwera. Czasami klient może potrzebować wykonywać ciągi powiązanych operacji usługowych w formie niepodzielnych jednostek nazywanych transakcjami. Usługi tego typu mogą być wykonywane przy wykorzystaniu wielu serwerów z danymi podzielonymi pomiędzy sobą. Jako przykład można przedstawić usługi bankowe, które mogą byś oparte na zbiorze serwerów, gdzie każdy serwer utrzymuje konta jednego oddziału.

Omówmy teraz zagadnienia kooperacji między klientami oraz utrzymywania długoterminowych danych w prostym jednoprocesowym serwerze. Przez serwer rozumiemy element systemu rozproszonego zarządzający jednym typem zasobu. Zasoby mogą zależeć od zastosowania, jak na przykład komunikaty poczty elektronicznej, lub rekordy kont bankowych, lub mogą być ogólne – w rodzaju plików, drukarek lub okien.

Serwer obudowuje zarządzane przez siebie zasoby, zezwala na ich tworzenie, sięganie po nie oraz na manipulowanie nimi za pomocą operacji, które mogą być zapoczątkowywane przez klientów. Jedynym sposobem, w jaki, klienci mogą sięgać po obiekty danych serwera, jest wywoływanie jego działań. Proces serwera zawiera w sobie dane określające stan jego zasobów. Wynik działania usługowego zależy od użytego zamówienia, jego argumentów i bieżących wartości obiektów danych serwera. Można powiedzieć, że obiekty danych reprezentujące zasoby zarządzane przez serwer mogą być przechowywane w pamięci ulotnej lub pamięci trwałej. Nawet w przypadku przechowywania ich w pamięci ulotnej serwer może korzystać z pamięci trwałej, żeby przechowywać w niej informacje o stanie zasobów, wystarczające do ich odtworzenia na wypadek awarii procesu usługowego.

Zasoby, zarządzane przez serwer są określone przez zapotrzebowania jego klientów:

- Serwer katalogowy może obejmować nazwy, adresy i inne szczegóły dotyczące grupy ludzi i dostarczać działań umożliwiających poszukiwanie nazw i adresów, ich dodawanie lub zmienianie.
- Na zasoby dotyczące usług wiązania, składają się odwzorowanie nazw usług na porty usług. Gdy serwer rozpoczyna prace staje się klientem łącznika i zaopatruje go w nowe odwzorowanie. Inni klienci łącznika korzystają z jego zasobów, aby lokalizować usługi.
- Zasoby sieciowych usług informacyjnych NIS firmy Sun składają się z odwzorowań kluczy na ich wartości. Odwzorowania mogą zawierać na przykład informacje pliku haseł lub nazwy komputerów sieciowych i ich adresy internetowe. Klienci serwerów NIS sięgają po te zasoby na przykład podczas rejestrowania się w systemie.
- Można zdefiniować serwer utrzymujący terminarzową bazę danych, która może być interakcyjnie przeglądana i uaktualniana przez pewną grupę użytkowników. Użytkownik, chcący umówić się z kimś na spotkanie, zagląda do terminarza danej osoby i wpisuje do niego datę spotkania.

Poniżej zilustrowane zostały poruszane aspekty. Użyto do tego prostych usług katalogowych, które nazwano „Adresowaniem”. Na usługi adresowania składają się następujące operacje na zbiorze zasobów zawierających nazwy i adresy:

```
Szukaj(nazwa)->adres        #Zwraca adres odpowiadający danej nazwie.
DodajAdres(wpis)            #Dodaje nowy wpis zawierający nazwę i adres.
ZmieńAdres(nazwa,adres)     #Zmienia adres skojarzony z nazwą.
UsuńAdres(nazwa)            #Usuwa wpis o podanej nazwie.
```

Dla danej nazwy klient może na przykład wywołać operację Szukaj serwera z usługami adresowymi nazywanymi „Adresowaniem” w sposób następujący:

```
Adresowanie$Szukaj(„Kowalski”);
```

Przedstawimy teraz problem wzajemnego zaburzania działań wykonywanych na zamówienie różnych klientów, które mogą się zdarzać, jeśli serwer nie zostanie starannie zaprojektowany.

2.2.1. Niepodzielne operacje serwera

Załóżmy, że serwer ma więcej niż jeden wątek, to wymaga, żeby jego operacje były niepodzielne, gdyż to umożliwia utrzymanie spójności jego danych. To oznacza, że na wykonanie danej operacji nie mają wpływu operacje wykonywane współbieżnie w innych wątkach. W usługach adresowania na przykład powinno być możliwe spójne wykonanie dwu współbieżnych zamówień Zmień Adres dotyczących tego samego wpisu. W celu wykonania operacji wątek może czytać lub zapisywać wartości pewnych obiektów danych. Żeby zagwarantować niepodzielność każdej operacji, serwer musi zapewnić, że dopóki jeden wątek nie zakończy działania, żaden inny wątek nie może mieć dostępu do tych samych obiektów danych. Można to osiągnąć za pomocą mechanizmów organizowania wzajemnego wykluczania. [29]

2.2.2. Poprawianie współpracy klientów przez synchronizowanie operacji serwera

Serwer może być używany przez klientów do dzielenia pewnych zasobów, takich jak nazwy i adresy. Dzieje się tak w sytuacji, gdy pewni klienci wykonują operacje aktualizacji zasobów serwera, a inni - operacje dostępu do zasobów. Może powstawać sytuacja, że w pewnych usługach operacja zamówiona przez jednego klienta nie może być zakończona do chwili wykonania operacji zamówionej przez innego klienta. Zdarza się to wtedy, kiedy część klientów jest producentami, a część jest konsumentami – wówczas konsumenci będą musieli zaczekać, aż producenci dostarczą potrzebnych towarów. Może do tego dochodzić także wtedy, gdy klienci dzielą jakiś zasób - klienci potrzebujący zasobu będą musieli poczekać na zwolnienie go przez innych klientów.

W sytuacji, kiedy klient-konsument zamawia zasób aktualnie niedostępny, powiedzenie klientowi, aby spróbował nabyć dany zasób później, nie jest zadowalające, jeśli dany zasób jest potrzebny klientowi do dalszego działania. Wprowadziłoby to klienta w stan aktywnego oczekiwania, a serwerowi przysporzyło dodatkowych zamówień (klient przez jakiś czas próbowałby realizacji zamówienia). Mogłoby to również prowadzić do nieuczciwego postępowania, gdyż inni klienci mogliby zrealizować zamówienia, zanim oczekujący klient zdecydowałby się na kolejną próbę. Zalecanym rozwiązaniem jest przechowanie zamówienia przez serwer i zorganizowanie oczekiwania klienta na odpowiedź do czasu, aż inny klient wytworzy zasób. Wówczas działanie konsumenta może być wznowione.

Aby podołać takim zadaniom, serwer musi:

- umieć zawieszać zamówienia, które nie mogą być wykonane natychmiast,
- kontynuować pobieranie zamówień od innych klientów
- wznawiać zawieszone zamówienia, gdy powstaną warunki do ich wykonania.

W celu jednoczesnego obsługiwania zamówień wielu klientów dla każdego nowego zamówienia serwer uruchamia nowy wątek, być może ograniczając liczbę wszystkich możliwych wątków. W sytuacjach, w których wątek nie może skutecznie działać, następuje jego zawieszenie. W przedstawionym wyżej przykładzie dzieje się tak wówczas, kiedy potrzebny zasób nie jest dostępny. Wątek taki jest później wznawiany. Wątek, który nie może kontynuować działania, sam prosi o zawieszenie za pomocą

operacji czekaj (ang. wait). Wznowienie działania zawieszonego wątku może spowodować inny wątek, używając operacji sygnalizuj (ang. signal).

2.3. Dialogi między klientem a serwerem

Istnieją aplikacje, w których zamówienie klienta może spowodować wykonanie przez serwer długiego obliczenia, stopniowo wytwarzającego wiele obiektów danych. Przykładem może być poproszenie serwera bazy danych o zwrócenie wszystkich wpisów o podanym kluczu. Godne zalecenia może tu być przesyłanie przez serwer takich wyników do klienta po kolei, po jednym, co pozwoli serwerowi i klientowi na współbieżne działanie.

W kolejnym przykładzie rozważymy projekt operacji w usługach nazywania i adresowania, podającej szczegóły dotyczące wszystkich aktualnie pamiętanych nazw i adresów. Jeśli serwer przechowuje wielką liczbę nazw i adresów, to będzie lepiej dla klientów, jeżeli będą oni mogli otrzymywać dane jedna po drugiej, małymi porcjami, zamiast uzyskać je wszystkie w jednym komunikacie. Pozwala to klientom na przetwarzanie każdej nazwy i adresu zaraz po ich nadejściu. Współpracę między klientem a serwerem można uznać za, konwersację, podczas której serwer pamięta miejsce, do którego dotarł każdy z klientów. W usługach umożliwiających konwersację klienci potrzebują dwóch nowych operacji:

```
OtwórzKonwersację()->IdKonwersacji#Prośba o podjęcie konwersacji z serwerem
                                  #zwrócony zostaje identyfikator
konwersacji.
ZamknijKonwersację(IdKonwersacji) #Zawiadamia o zakończeniu konwersacji.
```

Na ogół konwersacja dotyczy zasobów serwera. W tym przykładzie usługa adresowania odnosi się do wpisów. Zadaniem serwera jest przekazanie klientowi w odpowiedzi każdego wpisu z nazwą i adresem. Dysponuje on operacją umożliwiającą klientowi zamawianie następnego wpisu lub porcji wpisów:

```
NastępnyAdres(IdKonwersacji) -> wpis REPORTS KoniecCiągu
```

Zwraca szczegóły dotyczące następnej nazwy i adresu; jeśli nie ma więcej wpisów, to operacja zwraca sygnał błędu KoniecCiągu.

Tak, więc klient może wykonać następujące zamówienia, aby pobrać szczegóły dotyczące wszystkich nazw i adresów:

```
IdKonwersacji := Adresowanie$OtwórzKonwersację();
REPEAT
wpis := Adresowanie$NastępnyAdres(IdKonwersacji);
...       (* przetwarzanie wyniku *)
UNTIL OpisBłędu = KoniecCiągu
Adresowanie$ZamknijKonwersację(IdKonwersacji)
```

Każda konwersacja ma identyfikator (IdKonwersacji) zwracany przez serwer na jej początku. Usługi nazewnicze i adresowe będą rozszerzone tak, żeby dla każdego klienta aktualnie prowadzącego z nimi konwersację obejmowały zmienną zawierającą IdKonwersacji oraz odniesienie do następnej nazwy i adresu, których szczegółowe dane mają być zwrócone klientowi. Ilekroć serwer wykonuje operację OtwórzKonwersację, tylekroć zmienna taka zostaje przydzielona, a wskutek wywołania ZamknijKonwersację zmienna ta zostaje zwolniona.

Serwer wielowątkowy mógłby używać osobnego wątku do każdej konwersacji. Uprościłoby to programowanie, ponieważ każdy wątek musiałby pamiętać tylko własne położenie w ciągu pobieranych danych.

2.3.1. Serwery przechowujące stan

W ostatnim przykładzie serwer utrzymuje informacje o wszystkich zamówieniach klientów, którzy aktualnie z nim konwersują. Dla każdego klienta informacje te składają się z identyfikatora konwersacji następnego wpisu. Mogą tu ujawnić się problemy związane z serwerami przechowującymi stan, wynikające z ich wrażliwości na źle zaprojektowanego lub psującego się klienta. Jeśli klienci nie zamykają konwersacji, to po pewnym czasie

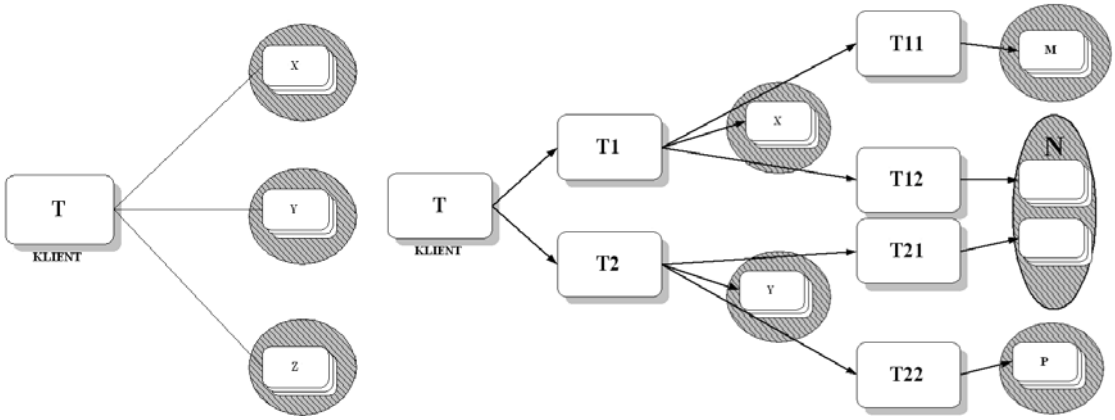
dojdzie do wyczerpania przestrzeni przydzielanej na pamiętanie informacji dotyczących konwersacji. Serwery utrzymujące stan na żądanie poszczególnych klientów przyjmują z reguły pewne ograniczenia czasowe, po przekroczeniu, których przestają przechowywać informacje dotyczące poszczególnych klientów.

2.4. Płaskie transakcje rozproszone i transakcje zagnieżdżone

[30] Transakcja rozproszona to taka, która korzysta z obiektów na różnych serwerach. Możemy wyróżnić jej dwa rodzaje – płaska i zagnieżdżona.

Zakończenie transakcji rozproszonej następuje gdy:

- wszystkie serwery ją zatwierdzą, lub
- wszystkie serwery ją zaniechają.



(a) (b)

Rys. 1. Idea transakcji płaskich (a) i zagnieżdżonych (b).

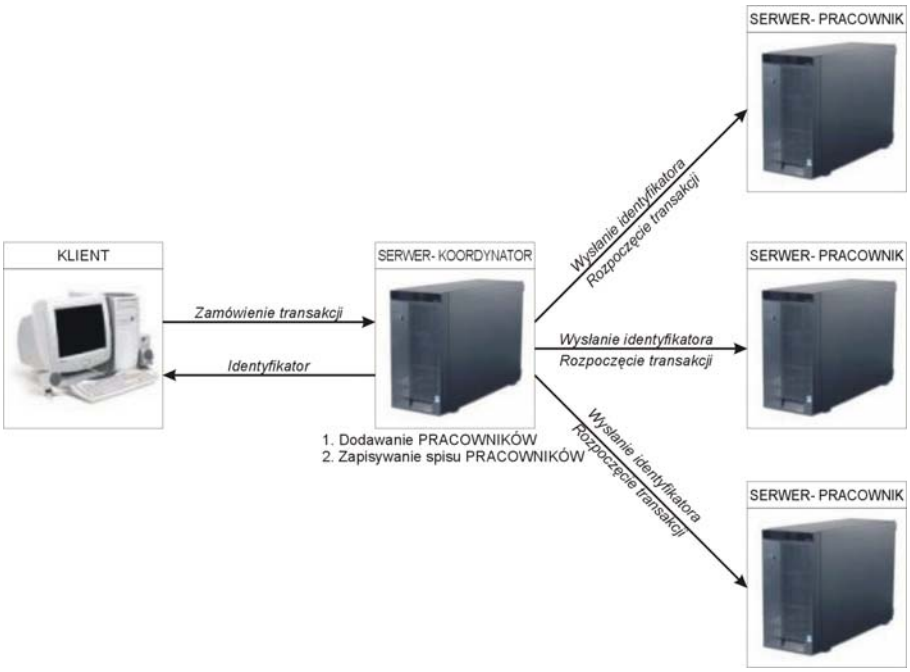
Podczas transakcji rozproszonych jeden z serwerów jest koordynatorem – czyli musi zapewnić ten sam wynik działania wszystkich serwerów, najczęściej stosowany, ze względu na "łatwość" implementacji jest, jest w tym celu „protokół zatwierdzania dwufazowego”.

Na każdym stanowisku działa lokalny koordynator transakcji, odpowiedzialny za kontrolowanie działań rozpoczętych na danym stanowisku (serwerze). Utrzymuje rejestr zdarzeń, zmian na wypadek konieczności rekonstrukcji po awarii.

Centralny koordynator nadzoruje nie poszczególne wykonania działań na serwerach – ale wyłącznie "binarny" sposób odpowiedzi o powodzeniu lub niepowodzeniu lokalnych transakcji od wszystkich lokalnych koordynatorów.

Dopiero po uzyskaniu pozytywnych sygnałów od wszystkich lokalnych koordynatorów – z centralnego wysyłany jest sygnał o możliwości zakończenia całej rozproszonej transakcji – poprzez końcowe zatwierdzenie wyników działań.

Jeśli choć w przypadku jednego z lokalnych serwerów koordynator uzyska sygnał o błędnym zakończeniu transakcji, lub przekroczony zostanie czas oczekiwania – wysyłany jest sygnał o konieczności wykonania operacji przywrócenia stanu sprzed rozpoczęcia transakcji (ang. *rollback*).

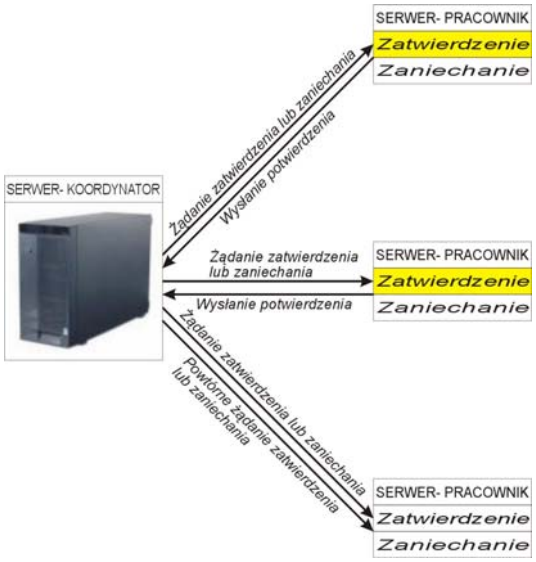


Rys. 2. Rozpoczęcie transakcji.

[30] W przypadku zagnieżdżonych transakcji rozproszonych, operacja na serwerze może wywołać operację na innym serwerze, a ta z kolei może wywołać operację na kolejnym serwerze. Transakcja zagnieżdżona ma strukturę hierarchiczną, przy czym transakcje na jednym poziomie mogą być wykonywane współbieżnie. W przypadku transakcji rozproszonych klient nie musi zdawać sobie sprawy, że ma do czynienia z rozproszoną strukturą danych.

2.4.1. Jednofazowy protokół niepodzielnego zatwierdzania

[31] Serwery wykonujące zlecenia, tworzące część transakcji rozproszonej muszą się porozumiewać w celu koordynacji swoich działań w trakcie zatwierdzania transakcji. Klient rozpoczyna transakcję, wysyłając do serwera zamówienie otworzenia transakcji. Serwer, z którym zostało zestawione połączenie zwraca klientowi jednoznaczny identyfikator transakcji, który składa się z dwóch części: identyfikatora serwera oraz pewnej, niepowtarzalnej w obrębie danego serwera liczby.

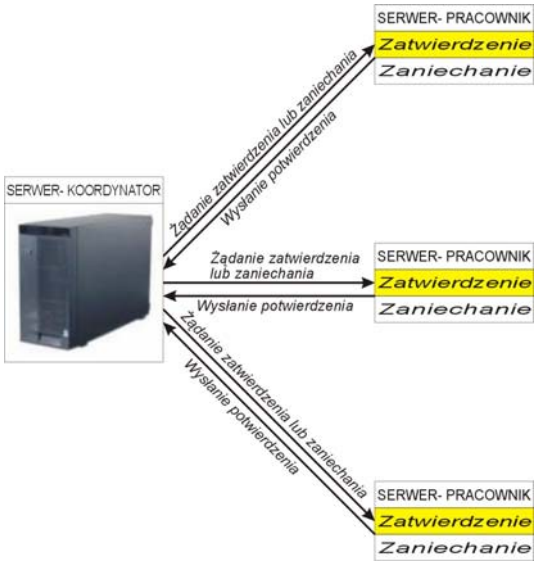


Rys. 3. Jednofazowy protokół niepodzielnego zatwierdzania.

Pierwszy serwer transakcji zostaje jej koordynatorem i jest odpowiedzialny za jej zatwierdzenie lub zaniechanie. Dodatkowo koordynator odpowiada za dodawanie innych

serwerów, które określa się mianem pracowników. Koordynator zapamiętuje spis pracowników, a każdy pracownik zapamiętuje identyfikator koordynatora, co umożliwia im zbieranie informacji potrzebnych do zatwierdzania transakcji.

Niepodzielność transakcji wymaga, aby po dojściu transakcji rozproszonej do końca wykonane były wszystkie jej operacje albo żadna. Transakcja dochodzi do końca, gdy klient prosi o jej zatwierdzenie albo zaniechanie. W najprostszym przypadku koordynator wysyła żądanie zatwierdzania lub zaniechania wszystkim serwerom biorącym udział w transakcji. Żądanie to jest ponawiane dopóty, dopóki wszystkie serwery nie potwierdzą jego spełnienia. Jest to przykład jednofazowego protokołu niepodzielnego zatwierdzania.

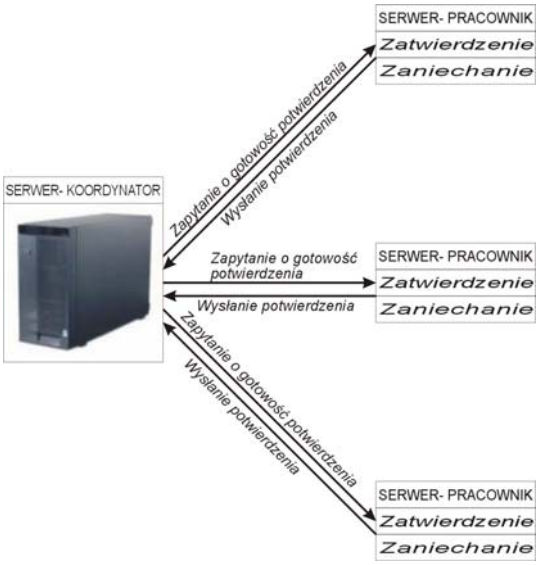


Rys. 4. [31]Jednofazowy protokół niepodzielnego zatwierdzania

Wadą jednofazowego zatwierdzania jest brak możliwości wycofania transakcji przez serwer. Najczęstszą przyczyną powstrzymującą serwer przed zatwierdzeniem transakcji jest sprawa sterowania współbieżnością.

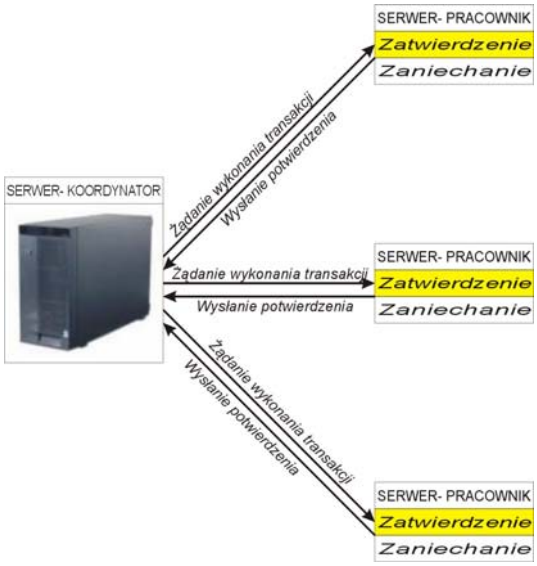
2.4.2. Protokół zatwierdzania dwufazowego

W celu umożliwienia dowolnemu serwerowi zaniechania jego części transakcji, a w konsekwencji do odwołania całej transakcji, opracowano dwufazowy protokół zatwierdzania.



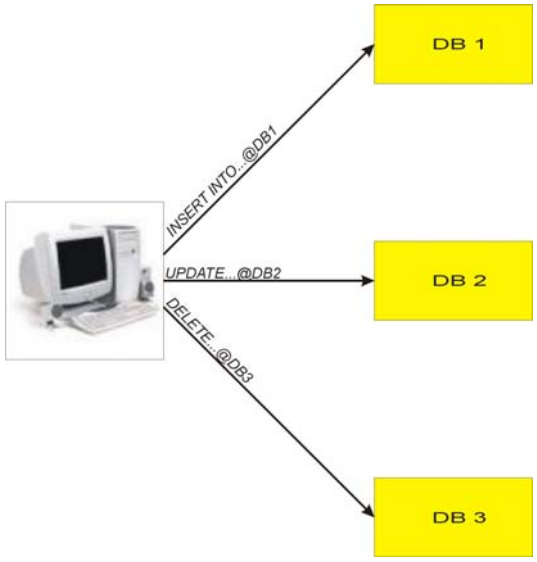
Rys. 5. Dwufazowy protokół niepodzielnego zatwierdzania – faza 1

[31] Dwufazowy protokół zatwierdzania rozpoczyna swoje działanie z chwilą, gdy klient poprosi koordynatora o zatwierdzenie transakcji. W pierwszej fazie działania protokołu koordynator pyta wszystkich pracowników, czy są gotowi do zatwierdzenia transakcji. Jeśli serwer-pracownik jest w stanie zatwierdzić własną część transakcji wysyła do koordynatora potwierdzenie gotowości. Jeśli koordynator odebrał potwierdzenia gotowości od wszystkich pracowników, wysyła do nich żądanie wykonania transakcji, przechodząc tym samym do drugiej fazy wykonywania transakcji. Pracownicy wysyłają do koordynatora potwierdzenia wykonania swojej części transakcji. Następnie koordynator może przekazać klientowi informację o stanie transakcji. W kolejnym kroku pracownicy utrwalają swoje zatwierdzenia, dzięki czemu koordynator wie, kiedy jego informacje przestają być potrzebne.



Rys. 6. Dwufazowy protokół niepodzielnego zatwierdzania – faza 2

Protokół zatwierdzania dwufazowego może zawieść wskutek awarii jednego lub więcej serwerów bądź z powodu przerwy w komunikacji między serwerami. Aby przezwyciężać awarie, każdy serwer przechowuje informacje dotyczące protokołu zatwierdzania dwufazowego w pamięci trwałej oraz włącza się do protokołu postępowanie oparte na odmierzaniu czasu.



Rys. 7. Dane rozproszone

2.4.3. Protokół zatwierdzania dwufazowego dla transakcji zagnieżdżonych

Zewnętrzna transakcja w zbiorze transakcji zagnieżdżonych jest nazywana transakcją szczytową, natomiast pozostałe transakcje są nazywane podtransakcjami. Podtansakcja wywołująca kolejną podtransakcję staje się jej rodzicielką, a wywołaną podtransakcja nazywa się jej potomkiem (dzieckiem).

Gdy podtransakcja kończy działanie, [32] podejmuje niezależną decyzję o tymczasowym zatwierdzeniu lub całkowitym zaniechaniu. Ostateczny wynik tymczasowego zatwierdzenia zależy od transakcji rodzicielskiej, a w końcowym rozliczeniu – od transakcji szczytowej. A zatem gdy transakcja szczytowa kończy działanie musi skontaktować się z serwerami pochodnych podtransakcji, aby zrealizować protokół niepodzielnego zatwierdzania.

Na ogół transakcja szczytowa może zostać zatwierdzona jeśli zostaną zatwierdzone wszystkie jej tymczasowo potwierdzone transakcje pochodne. Te z kolei mogą być zatwierdzone tylko wtedy, gdy dadzą się zatwierdzić wszystkie ich transakcje potomne, itd aż do osiągnięcia bezdzietnych transakcji potomnych. Transakcja zagnieżdżona, przekazuje swój stan transakcji rodzicielskiej. Jeśli transakcja potomna uległa zaniechaniu, to transakcja rodzicielska również ulegnie zaniechaniu, a co za tym idzie, pozostałe transakcje potomne zostaną zaniechane.

Transakcja szczytowa pełni rolę koordynatora w protokole zatwierdzania dwufazowego, a w wykazie pracowników figurują wszystkie serwery podtransakcji w drzewie, które dokonały tymczasowych zatwierdzeń.

Tak jak w przypadku protokołu zatwierdzania dwufazowego transakcji zagnieżdżonych mogą wystąpić problemy z komunikacją między koordynatorem a pracownikami. Dodatkowo może wystąpić problem, gdy tymczasowo zatwierdzona podtransakcja nie zostanie poinformowana o zaniechaniu transakcji rodzicielskiej. Aby zaradzić takim potencjalnym opóźnieniom, każda transakcja potomna będzie po upływie pewnego czasu zadawać pytania transakcji rodzicielskiej o jej stan. Jeśli nie uzyskają odpowiedzi mogą zapytać koordynatora o stan transakcji.

2.5. Sterowanie współbieżnością transakcji rozproszonych

Każdy serwer zarządza zbiorem danych i odpowiada za jego spójność, kiedy uzyskują do niego dostęp transakcje współbieżne. Z tego powodu każdy serwer odpowiada za sterowanie współbieżnością w odniesieniu do jego obiektów danych. Członkowie zbioru serwerów transakcji rozproszonych są wspólnie odpowiedzialni za zapewnianie, iż sposób ich działania jest równoważny szeregowo. Wynika z tego, że jeśli transakcja T poprzedza transakcję U w uporządkowaniu równoważnym szeregowo na jednym serwerze, to obie transakcje muszą mieć ten sam porządek na wszystkich serwerach, z których obiektów danych korzystają zarówno T, jak i U. Aby osiągnąć to samo uporządkowanie na wszystkich serwerach, muszą one uzgadniać kolejność swoich znaczników czasu, które składają się z lokalnego znacznika czasu i identyfikatora serwera. Ze względu na wydajność wymagana jest synchronizacja znaczników czasu na wszystkich serwerach, mimo iż to samo uporządkowanie transakcji można uzyskać bez synchronizacji zegarów fizycznych.

W transakcji rozproszonej [32] każdy serwer nadzoruje blokowanie własnych obiektów danych przy pomocy lokalnego zarządcy blokad. Blokada nie może zostać zwolniona dopóty, dopóki transakcja nie zostanie zatwierdzona lub zaniechana we wszystkich serwerach, na których następuje jej wykonanie.

Ponieważ serwery dokonują blokowania niezależnie od siebie, może zdarzyć się, że różne serwery wymuszą inne uporządkowania transakcji. Jeśli te różne uporządkowania doprowadzą do cyklicznych zależności między transakcjami może nastąpić zakleszczenie rozproszone. Takie zakleszczenia muszą być wykrywane przez serwery. Aby je usunąć

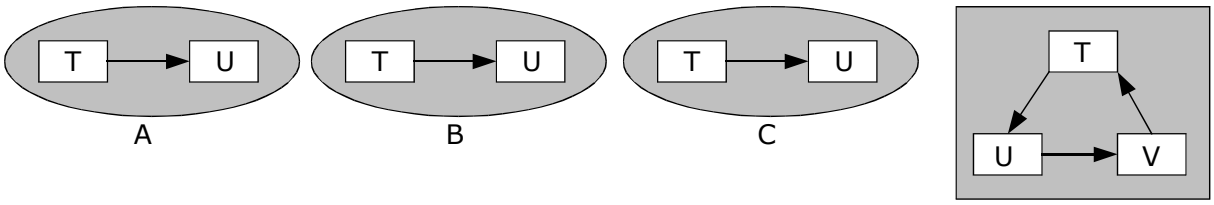
należy odwołać jakąś transakcję i powiadomić o tym koordynatora, który zaniecha tą transakcję.

Aby zapobiec potencjalnym konfliktom między poziomami w transakcjach zagnieżdżonych, nie zezwala się na współbieżne działanie transakcji rodzicielskiej i potomnej. Aby transakcja zagnieżdżona mogła dokonać zablokowania obiektu danych do czytania, wszyscy posiadacze blokad do pisania tego obiektu muszą być jej przodkami. Podobnie, aby transakcja zagnieżdżona mogła dokonać zablokowania obiektu danych do pisania, wszyscy posiadacze blokad do czytania tego obiektu muszą być jej przodkami. Podczas zatwierdzenia transakcji zagnieżdżonej, zablokowane przez nią zasoby przechodzą we władanie transakcji rodzicielskiej. W przypadku zaniechania transakcji blokady zasobów są usuwane.

2.5.1. Zakleszczenia rozproszone [32]

Wydzielonym serwerze może dochodzić do zakleszczeń, gdy do sterowania współbieżnością stosuje się blokowanie. Serwery muszą albo zapobiegać zakleszczeniom, albo je wykrywać i rozwiązywać. Większość schematów wykrywania zakleszczeń działa na zasadzie znajdowania cykli w grafie oczekiwania transakcji. W systemie rozproszonym składającym się z wielu serwerów, do których ma dostęp wiele transakcji, można skonstruować globalny graf oczekiwania z grafów lokalnych. W takim globalnym grafie oczekiwania może pojawić się cykl nie występujący w żadnym grafie lokalnym, co oznacza, że doszło do zakleszczenia rozproszonego. Globalny graf oczekiwania jest przechowywany we fragmentach na wielu serwerach, a zatem znajdowanie w nim cykli wymaga komunikacji między tymi serwerami.

Zakleszczenie, które zostaje "wykryte", lecz którego w rzeczywistości nie ma, jest nazywane zakleszczeniem pozornym. Z takim typem zakleszczenia możemy mieć do czynienia gdy zasób, biorący udział w zakleszczeniu zostanie zwolniony zanim cykl zostanie wykryty. Cykl może zostać wykryty dopiero po zebraniu wszystkich informacji dotyczących związków oczekiwania między transakcjami znajdą się w jednym miejscu. Zanim to nastąpi informacje o zasobach mogą być już nieaktualne.



(a) *(b)*

Rys. 8. Grafy oczekiwania: (a) lokalne, (b) globalne.

Innym podejściem do wykrywania zakleszczeń jest zastosowanie techniki zwanej pogonią za krawędziami lub przecieraniem drogi. W przeciwieństwie do poprzedniej metody nie konstruuje się globalnego grafu oczekiwania, lecz każdy z zaangażowanych serwerów pamięta część jego krawędzi. Serwery próbują znaleźć cykle za pomocą przekazywania komunikatów zwanych próbkami podążających wzdłuż krawędzi grafu w całym systemie rozproszonym. Taki komunikat zawiera związki oczekiwania transakcji reprezentujące drogę w globalnym grafie oczekiwania.

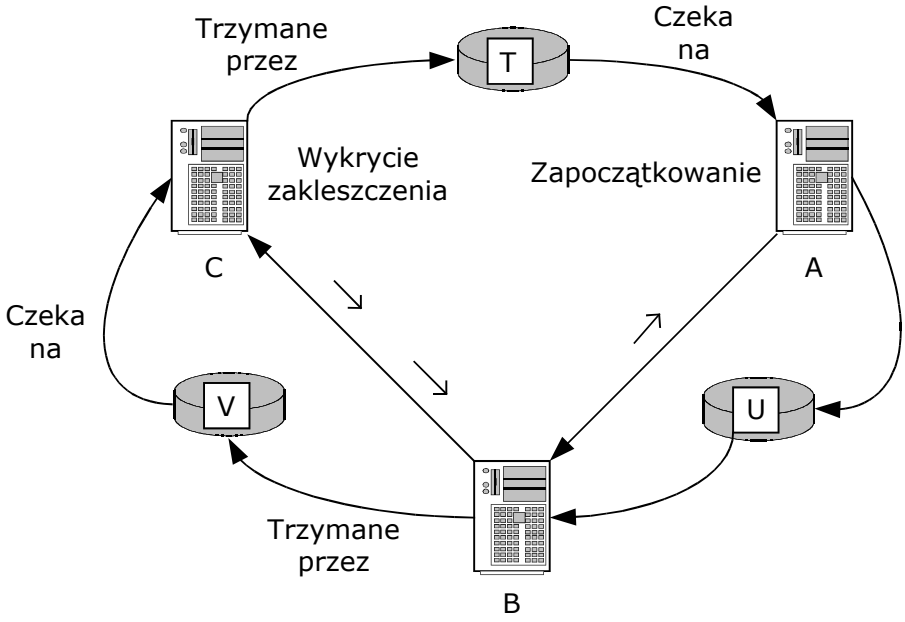
Algorytm pogoni za krawędziami ma trzy etapy: zapoczątkowanie, wykrywanie i rozwiązanie.

Pierwszy etap, czyli zapoczątkowanie, rozpoczyna się, gdy serwer zaobserwuje, że transakcja T zaczyna czekać na inną transakcję U, a U oczekuje na dostęp do obiektu danych na innym serwerze. W takim przypadku serwer wysyła próbkę zawierającą krawędź <T → U> do serwera z obiektem danych, z którego powodu transakcja U jest zablokowana. Jeśli U dzieli blokadę, to próbki są wysyłane do wszystkich posiadaczy tej blokady.

Kolejny etap, czyli wykrywanie polega na odbieraniu próbek i rozstrzyganiu o występowaniu zakleszczenia oraz o miejscu przeznaczenia kolejnych próbek. Dzięki

temu, krawędź po krawędzi, są budowane ścieżki prowadzące przez graf globalny. Przed przekazaniem próbki dalej serwer sprawdza, czy transakcja (np. T), którą właśnie dodał, nie spowodowała, że próbka zawiera cykl (np. <T → U → V → T>). Jeśli tak, to serwer znalazł cykl w grafie i wykrył zakleszczenie.

Po wykryciu zakleszczenia serwer przechodzi do ostatniego etapu – rozwiązania, czyli zaniechania jakieś transakcji, która przerwie zakleszczenie.



Rys. 9. Próbki wysyłane w celu wykrycia zakleszczenia

W powyższym algorytmie poszukiwanie zakleszczenia może się zacząć od każdej z transakcji uczestniczących w cyklu zakleszczenia. Może to spowodować odnalezienie zakleszczenia na kilku różnych serwerach, co z kolei może doprowadzić do zaniechania więcej niż jednej transakcji w cyklu. Aby zagwarantować, że tylko jedna transakcja w cyklu zostanie zaniechana, nadaje się transakcjom priorytety, czyniąc to w taki sposób, by wszystkie transakcje były całkowicie uporządkowane. Jako priorytetów można na przykład użyć znaczników czasu. Po znalezieniu cyklu zakleszczenia następuje zaniechanie transakcji z najniższym priorytetem. Dzięki takiemu rozwiązaniu wszystkie serwery podejmą tę samą decyzję co do tego, której transakcji należy zaniechać.

2.6. Transakcje ze zwielokrotnionymi danymi

Obiekty danych w serwerach transakcyjnych muszą być zwielokrotnione, aby zwiększyć wydajność oraz dostępność. Z punktu widzenie klienta zwielokrotnione usługi transakcyjne powinny wyglądać tak samo jak usługi bez zwielokrotnionych obiektów baz danych. Zwielokrotnione usługi transakcyjne gwarantują, że skutki transakcji wykonywanych przez różnych klientów na zwielokrotnionych obiektach danych będą takie same, jak gdyby wykonywano je po jednej w danej chwili. Ta własność nazywa się szeregowalnością na zasadzie jednej kopii.

Przezroczystość, czyli ukrywanie skutków rozproszenia przed użytkownikami, zwielokrotniania można uzyskać za pomocą osłony, która na ogół jest użytkowym pakietem klienta. Zamówienia klienta mogą być przesyłane do dowolnego z grupy zarządców logicznej kopii obiektu danych. Zarządca kopii, który otrzyma zlecenie wykonania jakiejś operacji na pewnym obiekcie danych, odpowiada za współpracę z innymi zarządcami kopii w grupie, przechowującymi kopie określonego obiektu danych. Po otrzymaniu zamówienia klienta zarządca kopii nie tylko je wykonuje, lecz zanim udzieli klientowi odpowiedzi, kontaktuje się również z niezbędną liczbą innych zarządców kopii.

W różnych schematach zwielokrotnień stosuje się różne reguły odnośnie do liczby zarządców kopii w grupie potrzebnych do udanego zakończenia operacji.

W schemacie "czytaj jedną - zapisuj wszystkie" operacja czytania może być wykonywana przez jednego zarządcę kopii, natomiast operacja pisania musi być wykonana przez wszystkich zarządców kopii w grupie.

W metodzie zwielokrotnianie kopii podstawowej, wszystkie zamówienia klienta są kierowane do jednego serwera podstawowego. Przy takim schemacie można zastosować sterowanie współbieżnością na serwerze podstawowym. W celu zatwierdzenia transakcji serwer podstawowy kontaktuje się z serwerami podporządkowanymi, a potem odpowiada klientowi. Ta forma zwielokrotniania pozwala serwerowi podporządkowanemu na przejęcie w razie konieczności obowiązków uszkodzonego serwera podstawowego.

Z kolei schemat zwielokrotnienia kopii dostępnych zaprojektowano, aby pozwolić na czasową niedostępność części zarządców kopii. W tym schemacie zamówione przez klienta czytanie logicznego obiektu danych może być wykonane przez dowolnego dostępnego zarządcę kopii, natomiast zlecona przez klienta aktualizacja musi się odbyć u wszystkich dostępnych zarządców kopii w grupie, rozporządzających kopią danego obiektu.

W schematach zwielokrotniania należy brać pod uwagę możliwość podziałów sieci, który powoduje rozdzielenie grupy zarządców kopii na dwie lub więcej podgrup. Ich członkowie mogą się kontaktować między sobą w obrębie grupy.

3. Rekonstrukcja danych w transakcjach rozproszonych [32]

Rekonstrukcja danych w systemach rozproszonych i transakcjach rozproszonych nie jest pojęciem trywialnym. Kiedy wiele procesów na raz wykonuje pewne operacje, a jeden lub kilka z nich zawiedzie, ochrona całego systemu jest pojęciem kluczowym. Chcąc zagwarantować, aby dana transakcja w systemie rozproszonym była atomowa, system musi sprawić, żeby wszystkie pojedyncze procesy związane z tą transakcją jednocześnie wykonywały jej zatwierdzenie, albo wycofanie. Odtwarzanie w systemie baz danych oznacza przywrócenie bazy danych do wcześniejszego prawidłowego stanu lub uznanego za prawidłowy. Gwarancją, że baza danych jest odtwarzalna, będzie zadbanie, aby każdy fragment informacji zawarty w bazie mógł być odtworzony z innej informacji przechowywanej nadmiarowo w innym miejscu w systemie. W systemach rozproszonych kontrola odtwarzania opiera się na protokole dwufazowego zatwierdzania lub jego odmianie. [33] Dwufazowe zatwierdzanie jest konieczne w każdym środowisku, w którym pojedyncza transakcja może oddziaływać z kilkoma autonomicznymi menadżerami zasobów.

3.1. Przyczyny występowania błędów danych

Systemy komputerowe mogą zawodzić wskutek wad którejś ze składowych, takich jak procesor, pamięć, urządzenia wejścia-wyjścia, kable czy oprogramowanie. Przez wadę (ang. *fault)* rozumie się niewłaściwe działanie wynikające być może z błędu projektowego, błędu produkcyjnego, błędu programistycznego, fizycznego uszkodzenia, niszczącego wpływu czasu, nieodpowiednich warunków otoczenia, nieoczekiwanych danych wejściowych, błędu operatora i wielu innych przyczyn. Nie wszystkie wady prowadzą (bezpośrednio) do awarii systemu, choć niektóre mogą je powodować.

Wady można ogólnie sklasyfikować jako przejściowe, nieciągłe i trwałe. Wady przejściowe (ang. *transient faults*) pojawiają się i ustępują. Przy powtórzeniu operacji taka wada znika. Jeśli upłynie czas wyczekiwania na transmisje i zostanie ona powtórzona, to prawdopodobnie za drugim razem przebiegnie poprawnie. Wady nieciągłe (ang. *intermittent faults*) pojawiają się i znikają samoczynnie, po czym znów się pojawiają itd. Luźny styk w łączeniu powoduje często wadę nieciągłą. Wady nieciągłe sprawiają mnóstwo kłopotów, gdyż są trudno wykrywalne. Wada trwała (ang. *permanent fault*) nie ustępuje aż do naprawy wadliwej składowej. Przepalone układy, błędy w oprogramowaniu, uszkodzenia głowic dyskowych powodują często wady trwałe.

Celem projektowania i budowania systemów tolerujących awarie jest uzyskanie pewności, że system jako całość będzie działał poprawnie nawet w obecności wad. Jest to cel zupełnie odmienny od celu zwyczajnej inżynierii, w której zakłada się wysoką niezawodność poszczególnych składowych, lecz dopuszcza (a nawet oczekuje), że w przypadku awarii jednej składowej cały system przestanie działać. Wady i usterki mogą występować na wszystkich poziomach: w tranzystorach, kostkach, płytach, procesorach, systemach operacyjnych, programach użytkowych itd. Tradycyjne opracowania z zakresu tolerowania uszkodzeń koncentrowały się głównie na statystyczne analizie wad elementów elektronicznych. W krytycznym systemie rozproszonym często bardziej interesuje nas możliwość uczynienia go zdolnym do przetrwania uszkodzeń składowych (w szczególności – procesora), aniżeli wyeliminowanie prawdopodobieństwa ich powstania. Niezawodność systemu nabiera szczególnego znaczenia w systemie rozproszonym ze względu na obecność wielkiej liczby składowych, co zwiększa szanse na wadliwość którejś z nich.

Bezpieczeństwo danych, bezpieczna, tj. bezawaryjna i ciągła praca są bezwzględnie wymagane wszędzie gdzie chodzi o wysoką niezawodność i szybką pracę. O wyżej wymienione cechy systemów komputerowych troszczymy się szczególnie gdy przy ich pomocy sterujemy ważnymi projektami, obsługujemy wielkie bazy danych. Np. obsługa portu lotniczego, czy łóżka pacjenta w szpitalu i niezwykle niebezpiecznego stanowiska

jakim jest reaktor jądrowy, wymusza od systemu komputerowego szybkie wykrywanie ewentualnych awarii i ich napraw oraz odpowiednich maskowań.

3.2. Technika wektora czasu – rozwiązywanie współbieżnych niepowodzeń w systemach rozproszonych

Teraz opisane zostaną systemy rozproszone pod kątem obsługi uszkodzeń, która dokonuje się w czasie rzeczywistym. Aplikacje rozproszone posiadają strukturę zbioru komunikujących się procesów, działają na wielu różnych procesorach. Ponieważ liczba procesów oraz czas pracy aplikacji w systemie jest duża i ma tendencję do ciągłego wzrostu, wzrasta także prawdopodobieństwo wystąpienia błędu systemu spowodowanego przez dowolny proces. Następstwa pojedynczego błędu mogą być bardzo różne, np. restart całej aplikacji, zawieszenie systemu. Gdy w grę wchodzi struktura rozproszona, kilka czy kilkaset procesorów, z których każdy obsługuje kilka procesów, prawdopodobieństwo awarii kilku procesów jednocześnie lub przynajmniej na nakładających się na siebie w czasie, jest zdecydowanie większe, a sama awaria może doprowadzić chociażby restartu systemu, awarii sprzętu, błędu użytkownika.

[34] Kompletne Odzyskiwanie Procesu (ang. *Complete Process Recovery*), jest jedną z technik naprawy stabilnego stanu sytemu rozproszonego, gdzie pod pojęciem stanu stabilnego rozumiemy pracę bez awarii. Technika ta zapewnia odpowiednie i pełne podejście do sprawy odzyskiwania procesów, ułatwia ich spójną odbudowę, daje właściwą obsługę zagubionych komunikatów oraz stara się maskować awarii przed systemem. Dzięki niej praca systemu powinna trwać bez przeszkód, jakby nie wydarzyło się nic szczególnego. Wymaga ona do naprawy, pojedynczego powtórzenia procesu w odpowiedzi na pojedynczy błąd i ma niską złożoność komunikatów.

3.2.1. Model systemu

Rozważmy model systemu, który składa się z N procesorów, indeksowanych od 1 do N, połączonych w sieć. Procesory nie dzielą pamięci fizycznej oraz globalnego zegara. Procesy komunikują się wyłącznie poprzez wymianę komunikatów bezawaryjnymi portami (kanałami) FIFO. Zakładamy, że warstwa sieci nie posiada punktów kontrolnych, natomiast w przypadku wystąpienia błędu, procesor przestaje funkcjonować bez generowania fałszywych lub błędnych komunikatów. Zakładamy też, że istnieje mechanizm naprawy posiadający możliwość zapisu i odnowy pracujących procesów. Mechanizm timeout, lub jakiś inny, wyszukuje awarie procesorów. Pojedyncze procesy aplikacji są częściowo zdeterminowane. Oznacza to, że ich stan jest funkcją komunikatów, które one otrzymują. Sama aplikacja jako całość może natomiast być niezależna od zmiennych czasów propagacji komunikatów.

Bliższe spojrzenie na obsługę komunikatów:

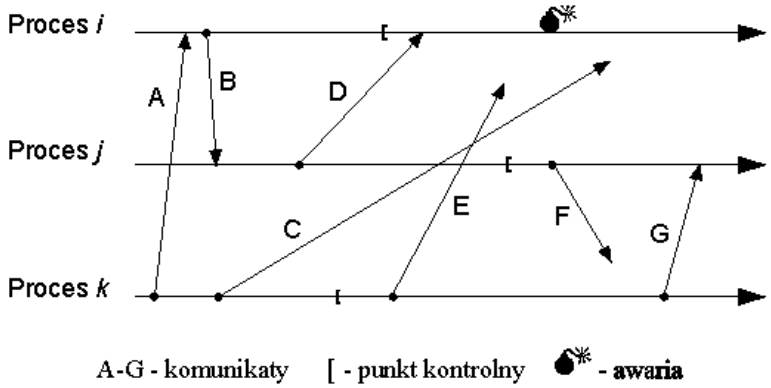
- Typowe rozproszone aplikacje "przeżywają" błędy przez stosowanie protokołów odzyskiwania bazujących na tzw. punktach kontrolnych procesu. Są dwie metody ustawiania punktów kontrolnych: synchroniczna i asynchroniczna.
- [34] W metodzie synchronicznej procesy koordynują się pomiędzy sobą aby upewnić się, czy istnieje spójny globalny stan. Metoda asynchroniczna nie wymaga żadnej koordynacji i dlatego nie ma żadnego wzajemnego oddziaływania podczas kontroli. Jakkolwiek, może się wydarzyć "efekt domina" lub rozległej propagacji, jeżeli punkty kontrolne nie mają spójnych globalnych ustawień.
- Dziennik komunikatów uzupełnia metodę asynchroniczną i pomaga zapobiegać "efektowi domina". Wzbogaca i polepsza metodę punktów kontrolnych. Zapisuje momenty wysyłania komunikatów i ich odbioru.

Zapis przychodzących komunikatów może być dokonywany w dwojaki sposób, w zależności o przyjętej kategorii: pesymistycznej lub optymistycznej. W metodzie pesymistycznej komunikat przychodzący zapisywany jest do pamięci stałej zanim zostanie przetworzony. Taki sposób wnosi jednak duże spowolnienie pracy. Optymistyczny schemat zapisu do dziennika nie od razu zapisuje komunikat do pamięci

stałej, umieszcza go w pamięci ulotnej i okresowo zapisuje grup zgromadzonych w tej pamięci komunikatów na dysk co poprawia szybkość pracy. Niestety komunikaty z pamięci ulotnej mogą być stracone w momencie awarii.

3.2.2. Obsługa komunikatów

Synchroniczne i asynchroniczne punkty kontrolne samodzielnie mogą doprowadzić systemy rozproszone do niespójności, wprowadzając do systemy komunikaty "sieroty" (komunikat z niewykonanymi rodzicami i zapisanymi zmianami w stanach niektórych procesów). Ale nawet z dodatkowym dziennikiem komunikatów, schemat punktów kontrolnych nie gwarantuje, że komunikat utracony, zdublowany lub opóźniony, odtworzony w procesie odzyskiwania, będzie poprawnie obsłużony. Kompletny Proces Odbudowy, daje nie tylko możliwość odbudowy spójności stanów ale także zapewnia obsługę właściwych komunikatów we właściwej kolejności.

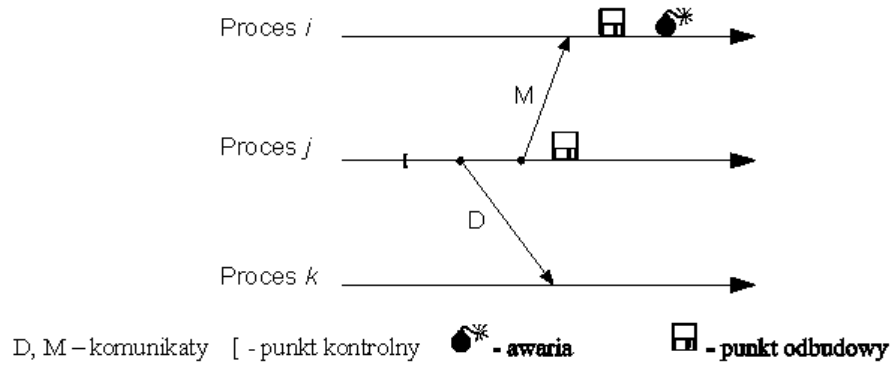


Rys. 10. Stan systemu w chwili awarii procesu i.

Na przykład na rysunku 10, procesy *i*, *j* oraz *k* wysłały wiadomości A-G. Wiadomości A, B, D i G dotarły do miejsca przeznaczenia, czyli innego procesu. Komunikaty C, E i F są ciągle w drodze kiedy procesor spowodował błąd procesu *i*. W początkowej próbie naprawy, przy użyciu ostatniego punktu kontrolnego (zaznaczonego jako "["), można odtworzyć stany procesów A, B i G, ponieważ początek i koniec komunikatu zostały zapisane (zapamiętane), a komunikat G jeszcze się nie rozpoczął, co zwalnia od potrzeby jego odzyskania. Komunikaty A i B nazywamy *normalnymi,* natomiast komunikat G określamy jako *zagubionymi*. Obsługa i odnowienie komunikatów C, D, E i F może stwarzać ewentualne problemy.

Sytuacja tych komunikatów w momencie awarii przedstawia się następująco: opóźniony komunikat C posiada kilka wariantów. Może skończyć się przed odtworzeniem *i*, może skończyć się w czasie odtwarzania lub może zakończyć się już po odtworzeniu *i*. Każda z powyższych możliwości musi być poprawnie obsłużona. Komunikat D posiada zapisaną wiadomość o swoim rozpoczęciu (w punkcie kontrolnym procesu *j*), nie posiada natomiast zapisanej wiadomości o swoim zakończeniu (proces *i* nie posiada punktu kontrolnego z zapisaną informacją o zakończeniu D). Należy zauważyć, że proces *j* nie prześle ponownie komunikatu D, ponieważ komunikat ten został wysłany przed zapisem stanu procesu w punkcie kontrolnym, a system komunikacyjny poprawnie dostarczył D, bez dodatkowego mechanizmu powiadamiania.

Wiadomości E i F są tak zwanymi komunikatami sierotami i stwarzają najpoważniejszy problem. Gdy dotrą one do swoich celów, zostaną odrzucone ponieważ wiadomości i ich wysłaniu nie zostały zatwierdzone. Procesy podczas procesu rekonstrukcji od ostatnich punktów kontrolnych, wygenerują od nowa te komunikaty. Techniki odbudowujące stabilne stany muszą rozróżniać komunikaty typu C od komunikatów typu E i F.



Rys. 11. Dublowanie komunikatów.

W czasie procesu odbudowy może dojść do podwojenia ilości tych samych komunikatów. Na rysunku 11, proces *j* wysłał komunikat D do procesu *k* oraz komunikat M do procesu *i*. Po załamaniu procesu *i* może być on odzyskany do momentu zaznaczonego na rysunku jako dyskietkę (tzw. *recovery point* – punkt odbudowy). Przypuszczalnie, proces *j* może być odtworzony poprzez wykonanie do punktu odbudowy, zrekonstruowanie punktu kontrolnego i odtworzenie dziennika komunikatów. Podczas odbudowy proces *j* wygeneruje od nowa komunikaty M i D, w wyniku czego proces *k* otrzyma komunikat D drugi raz. Wystąpi zdublowanie (podwojenie) komunikatu.

3.3. Podsumowanie

Rozproszone systemy znajdują zastosowanie w coraz większej części naszego życia. Nadzorują lotniska, szpitale, elektrownie i inne strategiczne gałęzie naszego życia. Od ich poprawnego funkcjonowania zależy nieraz życie wielu osób. Już dziś na takie systemy składają się wielkie i rozproszone zasoby sprzętowe, a doświadczenie pokazuje, że jak ze wszystkim, tak i w miarę ich rozwoju będą zwiększać swoje zasoby. Nie mamy tu na myśli już tylko samych zasobów sprzętowych. Działające na nich aplikacje będą coraz bardziej wymagające pod względem szybkości i złożoności obliczeń, będą stawiać coraz trudniejsze zadania, a obok tego będą wymagały zwiększania niezawodności tych systemów. Oczywiście nie jest możliwe skonstruowanie takiego systemu, który byłby całkowicie niezawodny, dlatego systemy rozproszone będą musiały szybko i sprawnie wykrywać i maskować awarie wszystkich swoich składowych sprzętowych i programowych.

Projektanci systemów rozproszonych mają więc przed sobą dwa zasadnicze problemy. Z jednej strony konstruowanie systemów o jak najmniejszej liczbie awarii, w tym celu badają wszelkie możliwe zachowania usług systemu nie tylko pod kątem ich poprawnego działania lecz także przypadki, w których mogą pojawić się awarie. Z drugiej strony, rozwijają ten zakresu usług, który zwiększa odporność systemu na błędy, np. dostępność wielu komputerów w celu wykrywaniu i maskowania możliwych awarii projektowanych usług. Opis sytuacji, w których może dojść do awarii usługi, nazywa się semantyką awarii usługi (ang. *service failure semantics*). Znajomość semantyki awarii może pozwolić na zaprojektowanie nowych usług, mających maskować awarie. Przykładem może być protokół TCP, skonstruowany w celu dostarczenia niezawodnych usług komunikacji strumieniowej za pomocą potencjalnie zawodnych usług datagramowych dostarczanych przez protokół IP.

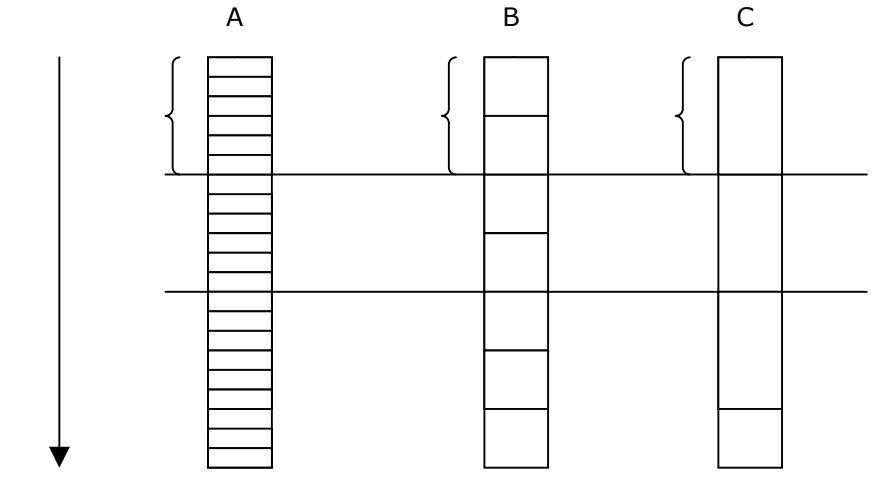
4. Synchronizacja czasu w systemach rozproszonych

Synchronizacja w systemach rozproszonych jest procesem bardziej złożonym niż w systemach scentralizowanych. W zagadnieniach synchronizacji procesów na wielu maszynach nie jest istotna znajomość czasu absolutnego. Istotne jest, aby procesy mogły ustalić kolejność zdarzeń. Z togo powodu pomiar czasu możemy podzielić na dwa elementy:

- pomiar czasu fizyczny – wskazują czas astronomiczny,
- pomiar czasu logiczny – zegary o wzajemnie uzgodnionym czasie, niekoniecznie astronomicznym.

Ponieważ system rozproszony powinien być niewrażliwy na awarie poszczególnych maszyn, to w sytuacji w której główny zarządca systemu ulega awarii, na mamy do czynienia z blokadą całego systemu i oczekiwanie na decyzję zarządcy. Jest to sytuacja niedopuszczalna. Co więcej stworzenie procesu kontrolującego wszystko w systemie rozproszonym powoduje powstanie wąskiego gardła w postaci właśnie tego procesu. Musi on decydować o przydziałach wszystkich zasobów systemowych, przez co może zostać przeciążony w każdym systemie rozproszonym. Jego działanie wymagałoby zastosowanie specjalnych przywilejów i środków technicznych w celu realizacji jego roli (szybsza jednostka centralna, większa ilość pamięci operacyjnej, itp.) Poza tym o efektywności działania systemu rozproszonego decyduje właśnie grupa komputerów o podobnych parametrach, a nie jeden „superkomputer”.

W systemie scentralizowanym czas jest określony jednoznacznie, w rozproszonym nie jest już to prosta sprawa. Zegar systemowy w komputerze to licznik rejestrujący drgania kryształu kwarcu, dla pojedynczego komputera nie Am większego znaczenia czy w ciągu np. 1 sec licznik zlicz[illegible] drgań czy 1001. Ma to jednak duże znaczenie w systemach rozproszonych, przy [illegible]nach składają[illegible]ię z wielu [illegible]tek, zwłaszcza podczas gdy każda jednostka [illegible]sny zegar dzi[illegible] z różną c[illegible]wością.



Rys. 12. Trzy komputery, każdy z własnym zegarem. Zegary działają z różnymi częstotliwościami.

Fakt ten powoduje, iż programowe zegary stopniowo się rozstrajają i wskazują różne wartości przy odczycie. Różnicę we wskazaniach zegarów nazywa się odchyleniem czasu (ang. *clock skew*). Wskutek rozbieżności we wskazaniach zegarów programy oczekujące dokładnych czasów skojarzonych z plikami, obiektami lub procesami, niezależnych od maszyny, na której je zmierzono (tj. niezależnych od użytego zegara) mogą działać błędnie.

Problem wzajemnego wykluczania:

Jeżeli składowe systemu działają niezależnie, ale wykorzystują do działania wspólne zasoby to pojawia się problem kolizji procesów żądających dostępu do tego samego zasobu. Pojawia się także problem w momencie wprowadzania zmian w zasobach poprzez poszczególne procesy, jedne zmiany mogą wykluczać inne.

4.1. Sposoby synchronizacji czasu w systemach rozproszonych

4.1.1. NTP

NTP (ang. *Network Time Protocol*) opracowany został w 1994 roku przez prof. David L. Mills z uniwersytetu w Delaware (USA). Opracował on mianowicie protokół rozgłaszania bardzo dokładnego czasu przez.

[35] W NTP rozróżniamy cztery warstwy:

- [35] komputery warstwy 1 to te, do których jest bezpośrednio podłączony zegar atomowy, bądź odbiornik zegarowego sygnału radiowego lub satelitarnego (obecnie na świecie jest ok. 50 publicznych takich serwerów),
- warstwa 2 (ok. 100 publicznych serwerów na świecie; liczba prywatnych nie jest znana) składa się z komputerów połączonych z tymi z 1. warstwy,
- 3 warstwa jest przeznaczona raczej dla serwerów wydziałowych, które (po jednym na wydziale) rozgłaszają dokładny czas innym komputerom z tego samego wydziału (nie jest to zbyt absorbujące),
- 4 warstwa to samodzielne komputery.

Zauważmy, że nawet przy małej ilości komputerów 1. warstwy takie rozwarstwienie nie obciąża łącz Internetu. Wewnątrz sieci lokalnych dokładność ustawienia czasu względem lokalnego serwera jest rzędu milisekund.

Warto zauważyć, że komputery pracują w różnych strefach czasowych. Wobec tego obowiązującym czasem rozgłaszanym przez serwery jest skoordynowany czas uniwersalny (UTC – ang. *Universal Coordinated Time*). Komputer, którego zegar wewnętrzny nie jest zbyt popsuty, prawidłowo korzystając z NTP powinien chodzić z dokładnością +/-2 ms w zależności od temperatury pomieszczenia, w którym pracuje.

Model NTP został pierwotnie zaimplementowany na systemach, które wywołują przerwanie zegarowe co pewną wielokrotność mikrosekundy, np. Linux – 100 Hz.

Ogólnie rzecz biorąc, demon NTP (np. xntpd) na komputerze, na którym działa manipuluje czasem systemowym w taki sposób żeby aby czas był zawsze monotoniczny, oraz jego wahania nie przekraczały dopuszczalnych w systemie norm. Takie manipulacje umożliwia komenda adjtimex – posługując się nią można np. zmienić szybkość zegara systemowego o +/-10%. Wytyczne demon pobiera poprzez port szeregowy (RS232) z odpowiedniego źródła (najczęściej innego komputera), które wysyła co sekundę sygnał PPS (ang. *pulse per second*). Przy uruchomieniu demona NTP sprawdzamy zgodność czasów. Jeżeli różnią się o więcej niż 1024 s, to trzeba po prostu ustawić czas przez settimeofday(). Jest to możliwe, gdyż sygnały PPS przesyłane są do portu szeregowego w postaci ASCII zawierającej dokładną bieżącą godzinę.

Następnie porównując czasy, jakie upływają między sekundą "komputerową", a sekundą "atomową" przysyłaną z zewnątrz, możemy ustalić, o jaki współczynnik należy przeregulować szybkość zegara systemowego(adjtimex()) .

Gdy odchylenie zegara systemowego zostanie ustabilizowane w granicach +/-128ms, pozostaje tylko co jakiś czas wołać procedurę ntp_adjtime(), która dokona w razie potrzeby niewielkich korekt. Będzie się to odbywało nie częściej niż co 16s, ale nie rzadziej niż co 1024 s. Ta procedura regulacji czasu najlepiej działa jednak dla odstępu między wywołaniami wynoszącego 64 s.

Błąd niezgodności może się teoretycznie wahać w granicach +/-512 ms, ale w praktyce na tym etapie odchylenia większe niż +/-128ms zdarzają się rzadko. W niektórych systemach występuje jednak trudny do usunięcia problem – przerwanie portu szeregowego musi mieć tam wyższy priorytet, niż przerwanie zegarowe. Wówczas sygnał synchronizacji przyjdzie o jedno tyknięcie spóźniony.

4.1.2. Zegary logiczne

Jeżeli synchronizacja zegarów nie musi być absolutna, nie ma konieczności synchronizacji zegarów z globalnym czasem, wtedy konieczna jest jedynie synchronizacja czasu w gronie komputerów realizujących dane zdania. Czas arbitralnie uzgodniony między poszczególnymi komputerami nosi nazwę czasu logicznego. Jeżeli kilka komputerów ustali i zsynchronizuje miedzy sobą określony czas, który nie ma żadnego przełożenia na czas rzeczywisty to jest to czas logiczny.

4.1.3. Algorytm Lamparta [36]

Algorytm umożliwia synchronizację zegarów systemowych w sieci rozproszonej, Lamport wykazał, że synchronizacja zegarów nie musi być absolutna.

Idea algorytmu Lamporta polega na dokonywaniu korekty czasu procesu odbierającego komunikat zgodnie z relacją uprzedniości zdarzeń. Proces odbierający komunikat sprawdza czas nadania tego komunikatu i porównuje ze swoim czasem odbioru. Jeśli czas odbioru jest późniejszy – korekty się nie dokonuje, jeśli czas odbioru jest wcześniejszy od czasu nadania to proces odbierający koryguje swój czas tak, żeby czas odbioru był o jeden impuls późniejszy względem czasu nadania (poprzez dodanie odpowiedniej ilości impulsów).

W celu przedstawienia działania algorytmu Lamparta należy przyjąć następujące zależności :

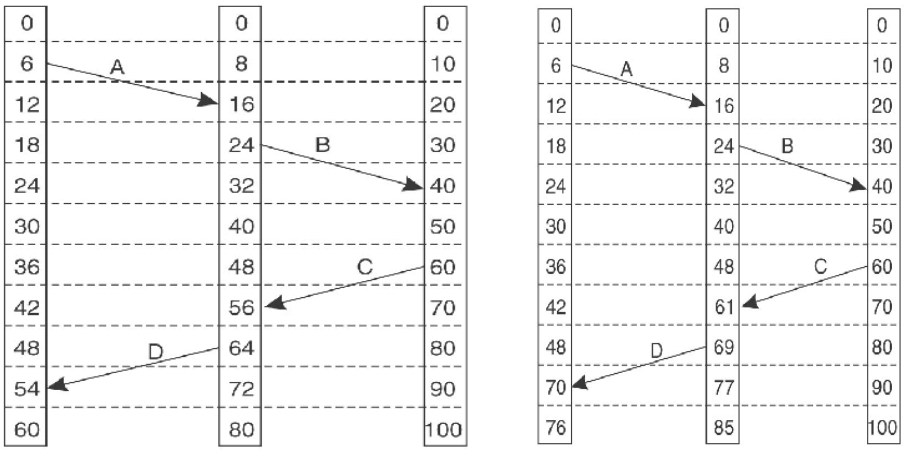
- Jeżeli wyrażenia A poprzedza wyrażenie B (oznaczamy A → B) oznacza to że najpierw zachodzi wyrażenie A, a po jego zakończeniu wyrażenie B. Jeśli A i B są zdarzeniami w tym samym procesie i A występuje przed B, to relacja A → B jest prawdziwa (jeśli A jest zdarzeniem wysłania komunikatu przez pewien proces i B jest zdarzeniem odebrania tego komunikatu przez inny proces, to relacja A → B jest również prawdziwa, ponieważ komunikat nie może być odebrany przed jego wysłaniem lub w tej samej chwili).
- Jeśli dwa zdarzenia A i B występują w różnych procesach, które nie wymieniają między sobą komunikatów to relacja A → B, ani B → A nie jest prawdziwa i bezcelowym jest ustalanie kolejności ich wykonywania.
- Uprzedniość zdarzeń jest relacją przechodnią: jeśli A→ B i B→ C, to A→ C.

Głównym założeniem, tego algorytmu, jest fakt, iż nie jest ważny czas absolutny, a jedynie następstwo zdarzeń poszczególnych procesów. Zatem nie jest ważnym z punktu widzenia pojedynczego procesu czas systemowy, gdyż nie daję on żadnych informacji. Proces w celu synchronizacji musi znać czas systemowy, ale w odniesieniu do czasu systemowego innego komputera.

Pomimo takiego opisu dostrzegamy konieczność opisania sposobu pomiaru czasu, za pomocą, którego każdemu zdarzeniu a będzie można przypisać wartość czasu C(A), co, do której wszystkie procesy byłyby zgodne. Wartości te muszą się charakteryzować tym, że jeżeli A → B, to C(A) < C(B).

Metoda zaproponowana przez Lamporta umożliwia przypisanie czasu wszystkim zdarzeniom w systemie rozproszonym według następujących warunków:

- Jeśli A poprzedza B w tym samym procesie, to C(A) < C(B).
- Jeśli A i B oznaczają nadanie i odbiór komunikatu, to C(A) <C(B).
- Dla wszystkich innych sytuacji, niż z pkt. 1 lub pkt. 2, dla zdarzeń A i B, C(A)≠C(B).



Rys. 13. Zegary bez synchronizacji (a),
zegary zsynchronizowane na podstawie algorytmu Lamparta (b).

4.1.4. Zegary fizyczne

[36] Algorytm Lamporta wykorzystuje do działania zegary logiczne, czasami jednak systemy wymagają dokładnej synchronizacji z czasem rzeczywistym. Najprostszym rozwiązaniem tego problemu jest zwielokrotnienie zegarów fizycznych i na podstawie ich wskazań wyznaczenie czasu średniego. Jednak w przypadku systemów o rozległej strukturze pojawia się problem synchronizacji poszczególnych elementów systemu znajdujących się w dużej odległości od siebie.

Aby móc odpowiedzieć na te pytania musimy przeanalizować synchronizację zegarów fizycznych. Jeśli jakaś maszyna jest wyposażona w odbiornik radiowy np. GPS, który zapewnia synchronizację z satelitą (błąd wzorca czasu w GPS jest na poziomie 340ns dla zastosowań cywilnych, a 200ns dla wojskowych) to celem staje się zsynchronizowanie z nią wszystkich innych maszyn. Jeśli żadna z maszyn nie ma takiego odbiornika, to wszystkie maszyny prowadzą własne pomiary czasu, natomiast celem staje się utrzymanie wzajemnych relacji czasowych tych maszyn. Żeby to zrealizować należy skorzystać ze specjalnego modelu. Przyjmijmy, iż każda maszyna wyposażona jest w czasomierz powodujący H przerwań na sekundę. Po wyzerowaniu czasomierza dochodzi do przerwania, które zwiększa licznik przerwań o wartość 1. W ten sposób zegar programowy zna obecną datę/godzinę. Przyjmijmy oznaczenie aktualnego czasu takiego zegara jako C, oraz przy założeniu, iż czas UTC wynosi t to wartość na maszynie „i" wyniesie Ci(t). Dla idealnego przypadku mielibyśmy Ci(t)=t dla każdego i, lub jednoznaczny opis, iż dCi/dt=1.

Rzeczywiste czasomierze nie generują przerwań dokładnie H razy na sekundę. Czasomierz o liczbie przerwań H = 60 powinien teoretycznie wytwarzać 216000 impulsów na godzinę. W praktyce błąd względny otrzymywany dla współczesnych układów czasomierzy wynosi około 10^{-5}. Oznacza to, że poszczególne maszyny mogą wykazywać od 215998 do 216002 impulsów na godzinę. Oznacza to, że jeśli istnieje stała p taka, że

$$1 - p < \frac{dC}{dt} \leq 1 + p$$

to można powiedzieć, że czasomierz działa zgodnie z założeniami. Stała p określana przez wytwórcę nosi nazwę maksymalnego współczynnika odchylenia (ang. *maximum drift rate*).

Jeżeli wskazania na dwóch zegarach odbiegają od czasu UTC w przeciwnych kierunkach, to po upływie czasu Δt od synchronizacji tych zegarów odstęp między ich wskazaniami wyniesie 2Δt. Jeśli projektanci systemu chcą zagwarantować, że żadne dwa zegary nie będą się nigdy różniły o więcej niż j, to zegary takie trzeba będzie poddawać (programowo) resynchronizacji, co najmniej co j/2p sekund.

4.1.5. Algorytm "Cristiana”

Algorytm dla systemów w których przynajmniej jedna z maszyn jest wyposażona w odbiornik GPS (maszynę z odbiornikiem nazwijmy serwerem czasu) a pozostałe wymagają z nią zsynchronizowania.

Każda maszyna wysyła okresowo komunikat do serwera czasu z pytaniem o bieżący czas. Serwer odpowiada możliwie jak najszybciej, za pomocą komunikatu zawierającego bieżący czas – C_{UTC}. Gdy nadawca otrzyma odpowiedź, wtedy w pierwszym przybliżeniu może ustawić swój zegar na C_{UTC}.

Jednak z tym algorytmem są związane dwie kwestie, poważnym problemem jest ograniczenie, aby czas nigdy nie płynął wstecz. Jeśli zegar nadawcy jest szybki, to wartość C_{UTC} będzie mniejsza niż bieżąca wartość czasu C(u) nadawcy. Zwykłe zastąpienie jej czasem C_{UTC} spowodowałoby poważne problemy.

Zmianę tego rodzaju należy wprowadzać stopniowo. Jedna z możliwych dróg wygląda następująco. Załóżmy, że czasomierz ustawiono tak, aby generował 100 przerwań na sekundę. W normalnych warunkach każde przerwanie dodaje do wartości czasu 10 ms. Podczas zwalniania podprogram obsługujący przerwanie dodaje za każdym razem tylko 9 ms, aż do wykonania całej poprawki. Zupełnie podobnie zamiast wykonywać jednorazowo skok w przód, można stopniowo przesuwać zegar, dodając 11 ms przy każdym przerwaniu.

Kolejnym problemem jest niezerowy czas zużywany na to, by odpowiedź serwera czasu dotarła do nadawcy. Co gorsza, opóźnienie takie może być duże i zmienne w zależności od obciążenia sieci. Aby określić ten czas należy dokonać pomiaru czasu pomiędzy wysłaniem zapytania (Tz) a otrzymaniem odpowiedzi (To) (pomiar jest dokonywany na tej samej maszynie dlatego będzie w miarę dokładny).

Gdy nie ma innych informacji, najlepszym oszacowaniem czasu przenoszenia komunikatu jest wielkość $\frac{(\text{czas otrzymania odpowiedzi - czas wysłysła zapytania})}{2}$.

Po nadejściu odpowiedzi w celu oszacowania bieżącego czasu serwera wartość czasu w komunikacie można zwiększyć o tę wielkość. W celu zwiększenia dokładności możemy doliczyć także czas potrzebny serwerowi na obsługę wysyłania komunikatu z czasem (czas od momentu jego pobrania do momentu wysłania z serwera). Jest faktem, że istnieją systemy tak zaprojektowane, że komunikaty od A do B. wędrują w nich innymi drogami niż komunikaty od B do A, mając w obu kierunkach różne czasy przenoszenia, jednak nie zakładamy tu takiej opcji.

W celu poprawienia dokładności Cristian zaproponował wykonywanie nie jednego, lecz serii pomiarów. Pomiary, w których różnica Tz – To przekroczy pewną wartość progową, są pomijane jako ofiary zagęszczeń w sieci i wobec tego niegodne zaufania. Oszacowania dokonane na podstawie pozostałych prób można uśrednić, otrzymując lepszą wartość. Można też uznać, że komunikat, który wrócił najszybciej, jest najdokładniejszy, bo przypuszczalnie napotkał najmniejszy ruch po drodze, więc najlepiej reprezentuje czysty czas przenoszenia.

4.1.6. Algorytm z Berkeley [37]

Serwer czasu w algorytmie Cristiana jest pasywny. Inne maszyny okresowo zadają mu pytania o czas. Jego działanie sprowadza się do udzielania im odpowiedzi. W systemie UNIX z Berkeley przyjęto dokładnie przeciwne założenie (Gusella i Zatti). Tutaj serwer czasu jest aktywny i odpytuje okresowo każdą maszynę o jej aktualny czas. Na podstawie tych odpowiedzi oblicza czas średni i nakazuje wszystkim innym maszynom, przesunąć zegary na ten nowy czas lub zwolnić ich chód, aż do osiągnięcia określonej redukcji. Jest to metoda odpowiednia dla systemu, w którym żadna maszyna nie ma odbiornika radiowego GPS lub równoważnego do niego.

4.1.7. Algorytmy uśredniania

Obie z opisanych powyżej metod są mocno scentralizowane, co – jak wiadomo – ma wiele wad. Znane są również algorytmy zdecentralizowane. Jedna z klas zdecentralizowanych algorytmów synchronizacji zegarów działa na zasadzie podziału czasu na przedziały resynchronizacji o stałej długości. i-ty przedział rozpoczyna się w chwili To +iR i trwa do chwili To + (i+ l)R, gdzie To jest ustalonym momentem w przeszłości, a R jest parametrem systemowym. Na początku każdego przedziału każda maszyna ogłasza bieżący czas jej zegara. Ponieważ zegary na różnych maszynach nie chodzą dokładnie z tą samą szybkością, ogłoszenia te nie będą nadawane jednocześnie.

Po nadaniu ogłoszenia maszyna włącza lokalny czasomierz i rozpoczyna zbieranie innych ogłoszeń, które nadchodzą po pewnym czasie S. Po nadejściu wszystkich ogłoszeń wykonuje się algorytm, który oblicza na ich podstawie wartość nowego czasu. Najprostszy algorytm polega na obliczeniu średniej z danych otrzymanych od wszystkich innych maszyn.

Możemy także zmodyfikować trochę algorytm odrzucając ekstremalne czasy nadejścia pakietów (ochrona systemu przed wadliwymi zegarami), oraz próbując oszacować czas przejścia pakietu synchronizacyjnego przez sieć. Oszacowanie takie można wykonać na podstawie znanej topologii sieci lub mierząc czas przenoszenia echa próbnego komunikatu.

Tym niemniej musimy uzmysłowić sobie fakt, iż w celu synchronizacji musimy znać architekturę systemu, przez co ograniczamy możliwość jego szybkiej rekonfiguracji, bez potrzeby przerabiania metod synchronizacyjnych.

4.1.8. Zwielokrotnione zewnętrzne źródła czasu

Systemy wymagające wyjątkowo dokładnej synchronizacji z czasem UTC można zaopatrzyć w wiele odbiorników GPS lub innych źródeł sygnałów UTC. Jednakże wskutek wewnętrznych niedokładności źródeł czasu oraz zmiany warunków atmosferycznych na drodze sygnału najlepsze, co może zrobić system operacyjny, to ustalenie zakresu (przedziału czasu) mieszczącego wartości UTC (możliwość zastosowania logiki rozmytej).

Problemem trudnym do ominięcia jest zjawisko, iż procesory nigdy nie dostają pakietów czasu natychmiast. Co gorszą, opóźnienie między nadaniem a odebraniem zależy od długości kabla oraz liczby wrót, które pakiet musi przebyć, różnych dla każdej pary (źródło UTC, procesor). Może się tu także zaznaczyć wpływ innych czynników, takich jak opóźnienia wynikłe z kolizji, do których dochodzi, gdy wiele maszyn próbuje transmitować w sieci Ethernet w tej samej chwili. Ponadto, jeżeli procesor jest zajęty obsługą poprzedniego pakietu, to może nawet nie spojrzeć na pakiet czasu przez sporą liczbę milisekund, wprowadzając w uzyskany pomiar czasu dodatkową niepewność.

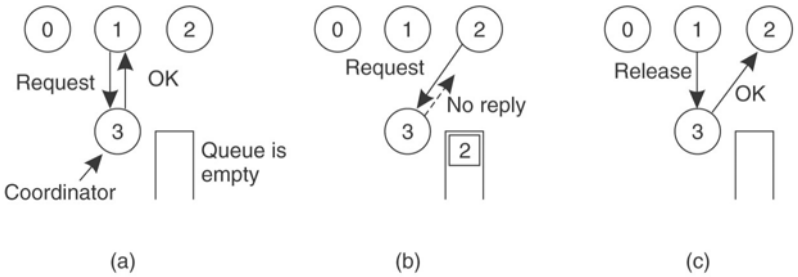
5. Algorytmy wykluczania

[38] Z zagadnieniem tym mamy do czynienia na przykład, gdy dwa równolegle wykonywane procesy korzystają ze wspólnej pamięci. Należy zapewnić wtedy mechanizm koordynujący dostęp do zasobów przez poszczególne procesy.

5.1. Algorytmy wykluczające w systemach rozproszonych

5.1.1. Algorytm scentralizowany

Algorytm oparty na istnieniu jednego koordynatora w danym systemie. Koordynator decyduje o umieszczeniu procesu w kolejce. Wszystkie procesy przeznaczone do realizacji wysyłają komunikat z żądaniem do koordynatora, który, jeżeli w kolejce jest wolne miejsce umieszcza w niej żądanie procesu i wysyła do niego komunikat. Po realizacji proces wysyła komunikat z informacja o jego zakończeniu w wyniku, czego koordynator usuwa go z kolejki i umieszcza tam kolejny oczekujący proces. Algorytm ten wymusza istnienie kilku koordynatorów na wypadek awarii jednego z nich.



Rys. 14. Zasada działania algorytmu scentralizowanego.

Proces 1 wysyła żądanie do koordynatora, ponieważ kolejka jest pusta koordynator wysyła mu odpowiedź i umieszcza go na liście. Następnie (b) proces 2 wysyła żądanie, ponieważ kolejka nie jest pusta nie otrzymuje on odpowiedzi od koordynatora. Dopiero kiedy koordynator otrzyma informacje od procesu 1 (ze się zakończył) wysyła komunikat do procesu 2 (c) i umieszcza go w kolejce.

W systemie istnieje proces będący koordynatorem zasobów. Wymienia on z procesami następujące typy komunikatów:

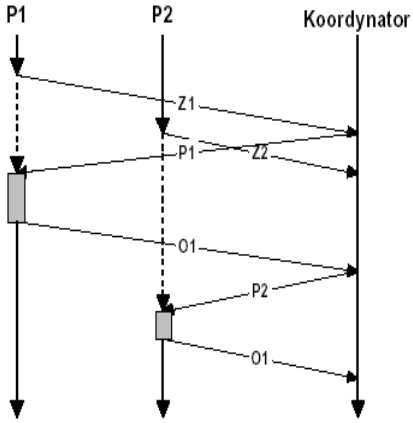
Z- Zamówienie zasobu Proces aplikacyjny -> Koordynator

P- Przedzielenie zasobu Koordynator -> Proces aplikacyjny

O- Oddanie zasobu Proces aplikacyjny -> Koordynator

Proces aplikacyjny działa według algorytmu:

1. Wysyła do koordynatora zamówienie zasobu.
10. Czeka na przydzielenie zasobu.
11. Używa zasobu.
12. Wysyła do koordynatora komunikat o zwolnieniu zasobu.



Rys. 15. Przebieg koordynacji w dostępie do zasobu.

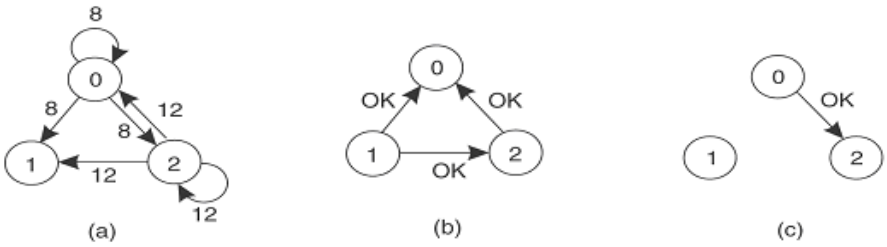
Wadą metody jest wrażliwość na awarie – gdy koordynator ulegnie awarii nie sposób odróżnić braku odpowiedzi od awarii.

5.1.2. Algorytm Ricarda Agrawali wzajemnego wykluczania

Algorytm ten, podobnie jak opisany wcześniej Algorytm Lamporta, wymaga, aby w systemie istniało całkowite uporządkowanie wszystkich zdarzeń, to znaczy dla każdej pary zdarzeń, takich jak komunikaty, musi być jednoznacznie wiadomo, które z nich wystąpiło pierwsze. Algorytm działa w następujący sposób. Proces, który chce wejść do sekcji krytycznej, buduje komunikat z nazwą tej sekcji, swoim numerem i bieżącym czasem. Następnie wysyła ten komunikat do wszystkich innych procesów, teoretycznie – łącznie z samym sobą (rys a). Przyjmuje się, że przesyłanie komunikatów odbywa się niezawodnie, tj. każdy komunikat jest potwierdzany (po wysłaniu komunikatu do siebie automatycznie generowana jest odpowiedź REPLAY).

Jeśli jest dostępna niezawodna komunikacja grupowa, to można się nią posłużyć zamiast indywidualnymi komunikatami. Jeżeli proces otrzyma zamówienie od innego procesu, to działanie, które podejmie, będzie zależeć od jego zamiarów związanych z sekcją krytyczną wymienioną w komunikacie. Można wyodrębnić trzy przypadki:

- [39] Jeśli odbiorca nie jest w sekcji krytycznej i nie zamierza do niej wchodzić, to odsyła do nadawcy komunikat OK (rys b). Po wysłaniu zamówienia z prośbą o pozwolenie wejścia do sekcji krytycznej proces zaczyna czekać, aż wszystkie inne procesy udzielą mu pozwolenia. Gdy skompletuje wszystkie pozwolenia, może wejść do sekcji krytycznej. Opuszczając sekcję krytyczną, proces wysyła komunikaty OK do wszystkich procesów, które ustawił w kolejce, i usuwa je stamtąd.
- Jeśli odbiorca właśnie przebywa w sekcji krytycznej, to nie odpowiada. W zamian ustawia zamówienie w kolejce(rys c). Jeśli nadchodzący komunikat ma mniejszy znacznik czasu, to odbiorca odsyła komunikat OK, natomiast, jeśli jego własny znacznik czasu jest mniejszy, to ustawia zamówienie w kolejce i nie odsyła niczego.
- Jeśli odbiorca chce wejść do sekcji krytycznej, lecz jeszcze tego nie zrobił, to porównuje znacznik czasu nadchodzącego komunikatu z tym, który sam wysłał w komunikacie do wszystkich. Mniejszy znacznik czasu wygrywa.



Rys. 16. Zasada działania algorytmu wzajemnego wykluczania.

Przy odpowiednim zaprojektowaniu systemu, algorytm ten wydaje się prawie optymalny, gdyby nie fakt, iż musimy znać wszystkich uczestników systemu. Czyli albo polegamy na komunikacji UDP jako komunikacji grupowej (zakładając przy tym jej niezawodność), bądź też musimy przetrzymywać gdzieś w systemie informację na temat aktualnej konfiguracji systemu i użyć pewnej komunikacji IP. Zarówno jedno jak i drugie rozwiązanie wydaje się niezbyt fortunne. Komunikacja UDP wcale nie zapewnia dostarczenia pakietów, co jest rzeczą niedopuszczalną w przypadku synchronizacji. Natomiast w komunikacji IP mamy pewien problem, jak nadać w jednej chwili komunikat do kilku/kilkunastu maszyn. Możemy adresować każdą po kolei, tym niemniej nie jest to z pewnością synchronizacja systemu w tym samym czasie, jako iż wysyłamy komunikaty sekwencyjnie. Lepszym rozwiązaniem wydaje się nadawanie w trybie do każdego na raz (broadcast). Tu mamy problem z wszech obecnymi routerami, które nie przepuszczą komunikatów do innych podsieci. Zatem synchronizacja w trybie broadcast jest możliwa tylko w obrębie jednej sieci.

Algorytm charakteryzuje się następującymi cechami:

1. Algorytm zapewnia wzajemne wykluczanie w systemie rozproszonym.
13. Algorytm wymaga, aby w systemie występowało całkowite uporządkowanie zdarzeń -zegary logiczne.

Rodzaje komunikatów:

Z – Zamówienie

P – Pozwolenie

W komunikacie wysyła się:

N – Nazwa zasobu

P – Identyfikator procesu

T – Wartość zegara logicznego

I. Komunikat z zamówieniem wysyłany jest do wszystkich procesów łącznie z samym sobą.

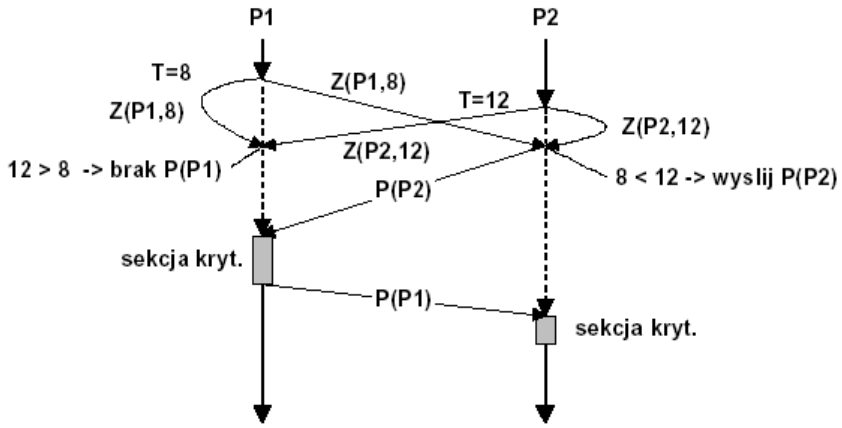
II. Gdy proces otrzyma zamówienie od innego procesu reakcja jest następująca:

1. Gdy proces nie jest w sekcji krytycznej danego zasobu i nie zamierza w nią wejść wysyła P (pozwolenie).
2. Gdy proces jest w sekcji krytycznej to nie odpowiada i ustawia zlecenie w kolejce. Odpowie, gdy wyjdzie z sekcji.
3. Gdy proces zamierza wejść do sekcji krytycznej, ale jeszcze tego nie uczynił porównuje znaczniki czasu otrzymanych komunikatów ze znacznikiem czasu wysłanego komunikatu z zamówieniem. Ten, kto ma mniejszy (wcześniejszy) czas wygrywa.

III. Po wysłaniu zamówień do wszystkich procesów, proces oczekuje zgody od wszystkich procesów. Gdy ja otrzyma może wejść do sekcji krytycznej.

Własności:

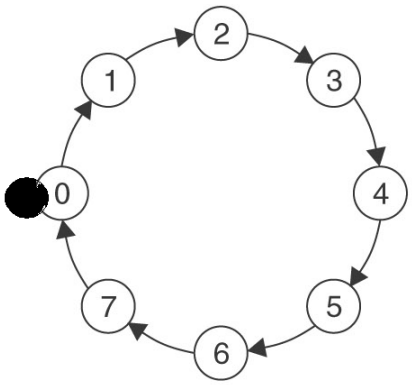
1. Dowodzi się ze algorytm poprawnie realizuje rozproszone wzajemne wykluczanie.
14. Nie występuję zagłodzenie, gdyż zamówienia są uporządkowane czasowo.
15. Awaria jednego procesu powoduje awarie całego systemu.
16. Proces musi być gotowy na odbiór zamówień i odpowiedzi na nie – asynchroniczny model programowania.



Rys. 17. Synchronizacja procesów P1 i P2.

5.1.3. Algorytm pierścienia z żetonem (ang. *Token ring*)

Zasada działania tego algorytmu jest bardzo prosta. Wszystkie procesy w systemie tworzą pierścień. O wejściu do sekcji krytycznej decyduje posiadanie żetonu (specjalny komunikat, tylko jeden w danym pierścieniu).



Rys. 18. Procesy połączone w pierścień logiczny.

Jak widać na rysunku 18 tylko jeden proces posiada żeton (proces 0), może z niego skorzystać tylko jeden raz. Po zakończeniu realizacji musi go przekazać dalej. Jeżeli kolejny proces nie chce wchodzić do sekcji krytycznej przesyła żeton dalej.

Algorytm procesu:

II. Gdy proces chce wejść do sekcji krytycznej:
    1. Oczekuje na żeton.
    17. Gdy otrzyma żeton wchodzi do sekcji.
    18. Po wyjściu przekazuje żeton do następnego procesu w pierścieniu.
IV. Gdy proces nie chce wejść do sekcji krytycznej przekazuje żeton następnemu procesowi.

Wady algorytmu:

1. Duża liczba komunikatów, gdy żaden z procesów nie jest w sekcji krytycznej.
19. Możliwość utraty żetonu – konieczny algorytm elekcji.
20. Konieczny asynchroniczny model programowania.

5.2. Porównanie metod synchronizacji

Algorytm	Liczba komunikatów na transakcje	Opóźnienie wejścia w komunikatach	Możliwe problemy
Scentralizowany	3	2	Awaria koordynatora
Rozproszony	2(n-1)	2(n-1)	Awaria dowolnego procesu
Pierścień z żetonem	od 1 do ∞	od 0 do ∞	Utrata żetonu, awaria procesu

5.2.1. Porównanie algorytmów synchronizacji rozproszonej:

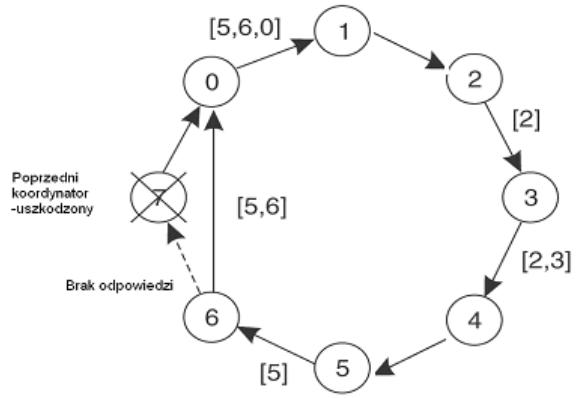
1. Wszystkie algorytmy są wrażliwe na awarie – najmniej algorytm scentralizowany, najbardziej rozproszony i z żetonem.
21. Najmniejsze opóźnienie – algorytm scentralizowany
22. Algorytmy rozproszony i z żetonem wymagają asynchronicznego modelu programowania.

6. Algorytmy elekcji

W wielu algorytmach rozproszonych potrzebny jest proces wyboru nowego koordynatora zarządzającego przebiegiem wykonywania poszczególnych procesów. Algorytmy elekcji służą do określenia, który proces zostanie nowym koordynatorem w momencie utraty poprzedniego. Jeśli Koordynator ulegnie awarii to algorytm musi wybrać proces który posiada największy numer.

6.1. Algorytm pierścieniowy

Algorytm pierścieniowy przeznaczony jest do systemów zorganizowanych w postaci pierścienia w którym łącza są jednokierunkowe.

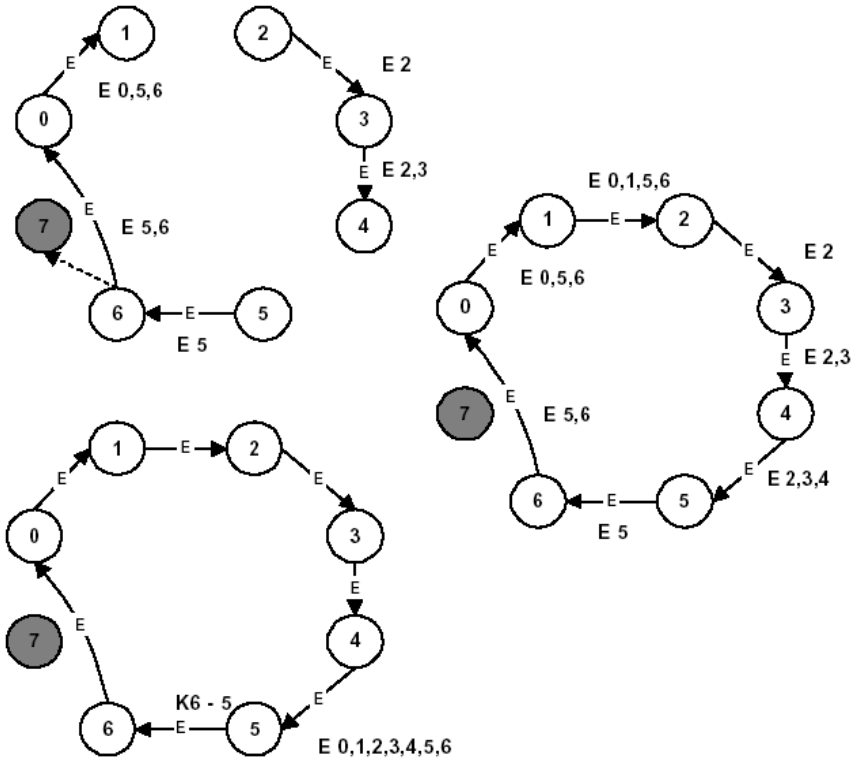


Rys. 19. Zasada działania algorytmu pierścieniowego.

Jeżeli koordynator (proces 7) został uszkodzony i zostanie to wykryte przez jeden lub więcej procesów zostaje rozpoczęta procedura wyboru nowego koordynatora. Każdy proces, który wykrył awarie, wysyła komunikat Election z własnym numerem. Jeżeli proces otrzyma komunikat, który sam wysłał (z jego numerem) oznacza to że został koordynatorem (wysyła komunikat o zwycięstwie do pozostałych procesów). Jeżeli więcej niż jeden proces wyśle komunikat o „wyborach" to zwycięzcą zostanie ten o wyższym numerze.

Założenia:

1. Procesy są uporządkowane w strukturę w kształcie pierścienia.
23. Każdy proces zna swego bezpośredniego następcę i może przesłać do niego komunikat. W systemie przekazywane są dwa typy komunikatów:
    - ELEKCJA, lista odwiedzonych procesów,
    - KOORDYNATOR numer koordynatora ,lista odwiedzonych procesów.
24. Gdy proces zauważy brak koordynatora buduje komunikat ELEKCJA z listą, do której dołącza swój numer. Komunikat wysyła do następnego czynnego procesu.
25. Gdy proces otrzyma komunikat ELEKCJA dołącza do listy swój numer i wysyła komunikat do następnego czynnego procesu.
26. Gdy komunikat powróci do procesu inicjującego elekcje to wybiera on proces o najwyższym numerze jako koordynatora i wysyła komunikat KOORDYNATOR(NrProcesuKoordynatora) z dołączonym swoim numerem.
27. Gdy komunikat KOORDYNATOR okrąży pierścień, usuwany jest z pierścienia.



Rys. 20. Ilustracja działania algorytmu pierścieniowego: proces 7 uległ awarii, proces 6 zostaje koordynatorem.

Do wyznaczenia koordynatora potrzeba n^2 komunikatów.

6.2. Algorytm tyrana

[40] Algorytm tyrana (ang. *bully algorithm*) przeznaczony jest do systemów, w których każdy proces może wysyłać komunikaty do wszystkich pozostałych procesów w celu ustanowienia koordynatora który decyduje o realizacji poszczególnych procesów. Służy do ustalenia następnego koordynatora po awarii poprzedniego.

Rodzaje komunikatów:

- ELEKCJA – podejmij elekcje,
- ANSWER – odpowiedz na ELEKCJA,
- KOORDYNATOR – jestem koordynatorem.

Proces, który zauważył ze koordynator nie odpowiada rozpoczyna elekcje.

Proces elekcji:

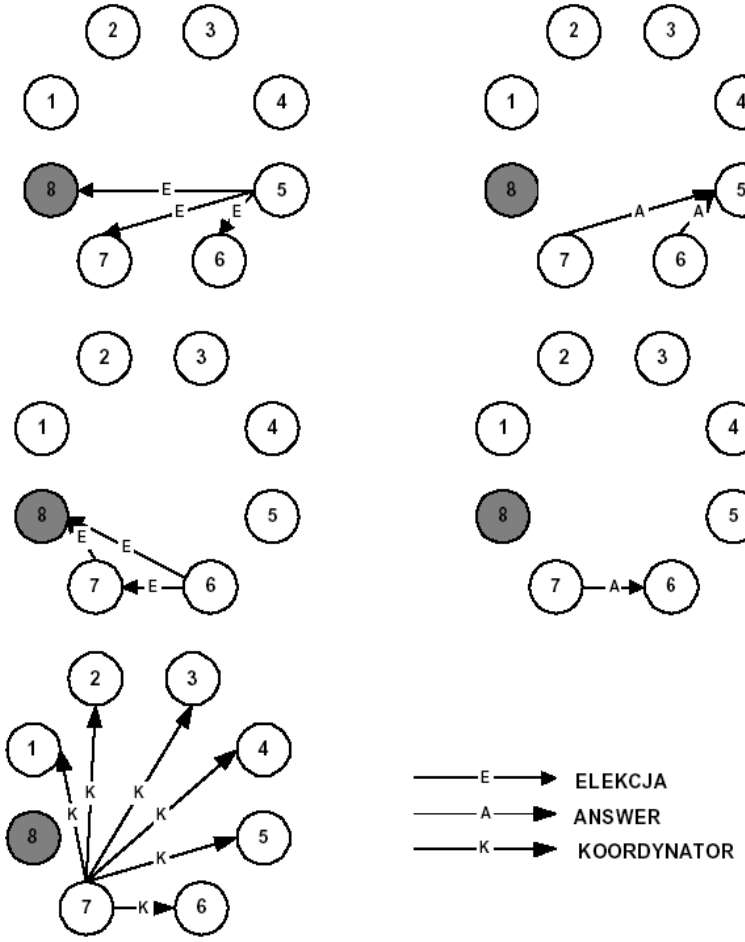
1. Proces P wysyła komunikat ELEKCJA do wszystkich procesów o wyższych numerach.
28. Jeżeli nikt nie odpowie proces P staje się koordynatorem. Wysyła komunikat KOORDYNATOR do wszystkich innych procesów.
29. Jeżeli jakiś proces o wyższym numerze odpowie to rola procesu P w elekcji się kończy. Proces o wyższym numerze dalej prowadzi elekcje.

Proces, który otrzymał komunikat ELEKCJA:

1. Wysyła komunikat ANSWER ze swoim numerem do procesu, od którego dostał komunikat ELEKCJA.
30. Podejmuje proces elekcji o ile jej nie prowadził. W końcu wszystkie procesy z wyjątkiem jednego zaniechają elekcji. Ten proces stanie się koordynatorem.

Przebieg elekcji – proces 5 inicjuje elekcje:

1. Proces 5 zauważa brak koordynatora i rozpoczyna elekcje. Wysyła komunikaty ELEKCJA do procesów P6, P7, P8.
31. Proces P6 otrzymuje komunikat ELEKCJA i wysyła ANSWER do P5.
32. P5 po otrzymaniu ANSWER zaprzestaje elekcji.
33. Proces P6 wysyła komunikaty ELEKCJA do P7 i P8.
34. Proces P7 otrzymuje komunikat ELEKCJA od P5 i P6. Odpowiada im komunikatem ANSWER.
35. Proces P6 otrzymuje od P7 ANSWER i zaprzestaje elekcji.
36. Proces P7 nie otrzymuje żadnego komunikatu ANSWER i zostaje koordynatorem.
37. Proces P7 wysyła komunikaty KOORDYNATOR do P1,...,P6. Do wyznaczenia koordynatora potrzeba n^2 komunikatów.



Rys. 21. Ilustracja działania algorytmu tyrana.

7. Bezpieczeństwo w sieciach rozproszonych

Systemy komputerowe ulegają niekiedy awariom. Wskutek uszkodzenia sprzętu lub oprogramowania programy mogą produkować błędne wyniki lub zatrzymywać się, nie ukończywszy zamierzonych obliczeń.

Są dwa podejścia, na których opiera się projektowanie systemów komputerowych tolerujących uszkodzenia:

a) redundancja sprzętowa – użycie nadmiarowych składowych,

b) odtwarzanie programowe – zaprojektowanie programów do usuwania skutków uszkodzeń.

Aby utworzyć systemy tolerujące awarie sprzętu, do jednego zastosowania „zatrudnia się" często dwa połączone ze sobą komputery, z których jeden działa jako maszyna rezerwowa.

W systemach rozproszonych redundancję można planować z większą precyzją. Zwielokrotnianie może dotyczyć poszczególnych serwerów o istotnym znaczeniu dla nieprzerwanego działania ważnych zastosowań.

Przydział nadmiarowego sprzętu potrzebnego do tolerowania uszkodzeń można tak zaprojektować, by w przypadku bezawaryjnego działania systemu używać tego sprzętu do zadań niekrytycznych. Można na przykład zwielokrotnić bazę danych w kilku serwerach, aby zapewnić, że dane pozostaną dostępne po awarii dowolnego, pojedynczego serwera. Serwery mogą być tak zaprojektowane, aby wykrywały uszkodzenia u swoich kolegów. Gdy w którymś serwerze nastąpi wykrycie uszkodzenia, wówczas klientów skieruje się do pozostałych serwerów. Za pomocą tego sposobu można stosunkowo tanio uzyskiwać tolerowanie niektórych rodzajów uszkodzeń sprzętu w systemach rozproszonych.

Odtwarzanie programowe wymaga zaprojektowania oprogramowania do odtwarzania, czy też przywracania, stanu trwałych danych po wystąpieniu awarii. Ogólnie rzecz biorąc, obliczenia wykonane przez pewne programy mogą być niekompletne w chwili wystąpienia uszkodzenia, a aktualizowane przez nie trwałe dane (pliki i inny materiał przechowywany w pamięci trwałej) mogą utracić spójność.

Niezawodność można zwiększyć, tak projektując system, aby wady i nieprawidłowości jego działania mogły być wykrywane i usuwane. Sprzętowa lub programowa przyczyna niedomagania nazywa się uszkodzeniem lub też wadą (ang. *fault*). System tolerujący uszkodzenia to taki, który potrafi wykryć uszkodzenie i albo w sposób przewidywalny zaprzestaje działania, albo je wykrywa, dzięki czemu użytkownik systemu nie dostrzeże żadnego defektu.

Tolerowanie uszkodzeń jest podstawową cechą systemów rozproszonych. Osiągnięcie tej cechy nie odbywa się automatycznie i zależy od projektu oprogramowania systemu rozproszonego.

7.1. Wady składowych systemu

[41] Systemy komputerowe mogą zawodzić wskutek wad którejś ze składowych, takich jak procesor, pamięć, urządzenia wejścia-wyjścia, kable czy oprogramowanie. Przez wadę rozumie się niewłaściwe działanie wynikające być może z błędu projektowego, błędu produkcyjnego, błędu programistycznego, fizycznego uszkodzenia, niszczącego wpływu czasu, nieodpowiednich warunków otoczenia, nieoczekiwanych danych wejściowych, błędu operatora i wielu innych przyczyn. Nie wszystkie wady prowadzą (bezpośrednio) do awarii systemu, choć niektóre mogą je powodować.

7.1.1. Klasyfikacja wad

[42] Wady przejściowe (ang. *transient faults*) pojawiają się i ustępują. Przy powtórzeniu operacji taka wada znika. Jeśli upłynie czas wyczekiwania na transmisję i zostanie ona powtórzona, to prawdopodobnie za drugim razem przebiegnie poprawnie.

Wady nieciągłe (ang. *intermittent faults*) pojawiają się i znikają samoczynnie, po czym znów się pojawiają itd. Luźny styk w łączeniu powoduje często wadę nieciągłą. Wady nieciągłe sprawiają mnóstwo kłopotów, gdyż są one trudno wykrywalne.

[42] Wada trwała (ang. *permanent fault*) nie ustępuje aż do naprawy wadliwej składowej. Przepalone układy, błędy w oprogramowaniu, uszkodzenia głowic dyskowych powodują często wady trwałe.

Celem projektowania i budowania systemów tolerujących awarie jest uzyskanie pewności, że system jako całość będzie działał poprawnie nawet w obecności wad. Jest to cel zupełnie odmienny od celu zwyczajnej inżynierii, w której zakłada się wysoką niezawodność poszczególnych składowych, lecz dopuszcza (a nawet oczekuje), że w przypadku awarii jednej składowej cały system przestanie działać.

Wady i usterki mogą występować na wszystkich poziomach: w tranzystorach, kostkach, płytach, procesorach, systemach operacyjnych, programach użytkowych itd. Tradycyjne opracowania z zakresu tolerowania uszkodzeń koncentrowały się głównie na statystycznej analizie wad elementów elektronicznych.

Jeśli prawdopodobieństwo uszkodzenia jakiegoś elementu wynosi w danej sekundzie p, to prawdopodobieństwo, że ulegnie on awarii dopiero po upływie kolejnych k sekund wynosi:

$$p(1-p)^k$$

Oczekiwany czas do wystąpienia awarii jest zatem określony wzorem:

$$\text{średni czas do awarii} = \sum kp(1-p)^{k-1}\text{, gdzie } k = 1...\infty$$

Posługując się dobrze znanym równaniem na sumę nieskończoną i rozpoczynając od k=1, mamy:

$$\sum \alpha^k = \alpha / (1-\alpha)$$

gdzie $\alpha = 1-p$. Różniczkując stronami wynikowe równanie względem p i mnożąc przez $-p$, otrzymujemy:

$$\text{średni czas do awarii} = 1 / p$$

Na przykład, jeśli prawdopodobieństwo uszkodzenia w ciągu sekundy wynosi 10^{-6}, to średni czas do wystąpienia awarii wynosi 10^6 sekundy.

7.2. Awarie systemu

W krytycznym systemie rozproszonym często bardziej interesuje nas możliwość uczynienia go zdolnym do przetrwania uszkodzeń składowych (w szczególności – procesora), aniżeli wyeliminowanie prawdopodobieństwa ich powstawania. Niezawodność systemu nabiera szczególnego znaczenia w systemie rozproszonym ze względu na obecność wielkiej liczby składowych, co zwiększa szansę na wadliwość którejś z nich.

Opis sytuacji, w których może dojść do awarii usługi, nazywa się semantyką awarii usługi (ang. *service failure semantics*). Znajomość semantyki awarii usługi może pozwolić na zaprojektowanie nowych usług, mających maskować wadliwe zachowanie usług, którym są one podporządkowane.

System tolerujący uszkodzenia potrafi wykryć awarię i albo psuje się w sposób przewidziany, albo maskuje awarię przed użytkownikami. Usługa tolerująca awarie (ang. *fault-tolerance service*) działa zgodnie z jej specyfikacją pomimo występowania awarii w usługach, od których zależy.

Serwer maskuje (ang. *masks*) awarię w usłudze, która od niego zależy, przez całkowite jej ukrycie lub zamieniając ją na jedno z zachowań wadliwych, które wolno mu uzewnętrzniać. W ostatnim przypadku z reguły zamienia on typ awarii z dotyczącej usług niskiego poziomu na odnoszącą się do wyższego poziomu.

Można wyróżnić:

a) uszkodzenia uciszające,

c) wady bizantyjskie.

[41] Procesor dotknięty uszkodzeniem uciszającym (ang. *fail-silent fault*) po prostu się zatrzymuje i nie odpowiada na dalsze sygnały wejściowe lub produkuje dalsze sygnały wyjściowe – może za wyjątkiem powiadomienia, że już nie działa. Skutki takich zachowań nazywa się również wadliwymi zatrzymaniami (ang. *fail-stop faults*).

Procesor z wadą bizantyjską (ang. *Byzantine fault*), kontynuując działanie, źle odpowiadając na pytania i być może współpracując pokrętnie z innymi wadliwymi procesorami, tworzy wrażenie, że zarówno on jak i pozostałe są w najlepszym porządku. Nie wykryty błąd w oprogramowaniu często objawia się w postaci wad bizantyjskich. Postępowanie w przypadku tych wad jest oczywiście znacznie trudniejsze od postępowania w przypadku uszkodzeń uciszających.

7.3. Porównanie systemów synchronicznych i asynchronicznych

[41] Wady składowych systemów mogą być przejściowe, nieciągłe lub trwałe, a awarie systemu mogą przybierać formę ciszy bizantyjskich intryg. Trzecia, prostopadła oś dotyczy zachowania systemu w pewnym abstrakcyjnym czasie. Załóżmy, że mamy system, w którym, gdy jeden procesor wysyła komunikat do innego, gwarantuje się otrzymanie odpowiedzi w z góry określonym czasie T. Nieuzyskanie odpowiedzi oznacza uszkodzenie systemu odbiorczego. Czas T jest wystarczający do obsługi zaginionych komunikatów (n-krotne, w razie konieczności, ich wysyłanie).

W kontekście badań nad tolerowaniem uszkodzeń system charakteryzujący się tym, że jeśli pracuje – zawsze odpowiada na komunikat w znanym, skończonym przedziale czasu, zwie się synchronicznym. System nie mający tej właściwości nosi nazwę asynchronicznego.

Systemy asynchroniczne są trudniejsze do opanowania od systemów synchronicznych. Jeśli procesor może wysłać komunikat, wiedząc, że brak odpowiedzi w czasie T oznacza awarię obranego odbiorcy, to może podjąć działania naprawcze. W przypadku braku limitu czasu oczekiwania na odpowiedź problemem staje się samo określenie, czy awaria wystąpiła.

7.4. Zastosowanie redundancji

Ogólne podejście do tolerowania uszkodzeń polega na użyciu redundancji. Są możliwe jej trzy rodzaje:

a) redundancja informacji,

d) redundancja czasu,

e) redundancja fizyczna.

Redundancję informacji tworzą dodatkowe bity umożliwiające odtwarzanie zniekształconych bitów. Na przykład do transmitowanych danych można dodać kod Hamminga, aby poradzić sobie z szumami linii transmisyjnej.

Redundancja czasu polega na wykonaniu działania i, w razie potrzeby, na jego powtórzeniu. Jest nam szczególnie pomocna tam, gdzie w grę wchodzą wady przejściowe lub nieciągłe.

[43] Redundancja fizyczna powstaje przez dodanie specjalnego wyposażenia, aby system jako całość mógł tolerować utratę pewnych składowych lub ich wadliwe działanie. Na przykład można wyposażyć system w dodatkowe procesory, by mimo awarii kilku z nich mógł wciąż działać poprawnie. Owe dodatkowe procesory można zorganizować na dwa sposoby: jako aktywne zwielokrotnienie lub jako zasoby rezerwowe. Rozważmy

przypadek serwera. Aktywne zwielokrotnienie polega na tym, że wszystkie procesory są przez cały czas używane jako serwery (równoległe), aby całkowicie ukryć uszkodzenia. W przypadku zasobów rezerwowych używa się tylko jednego procesora, zastępując go rezerwowym, jeżeli ulegnie uszkodzeniu.

W obu strategiach można wyodrębnić następujące zagadnienia:

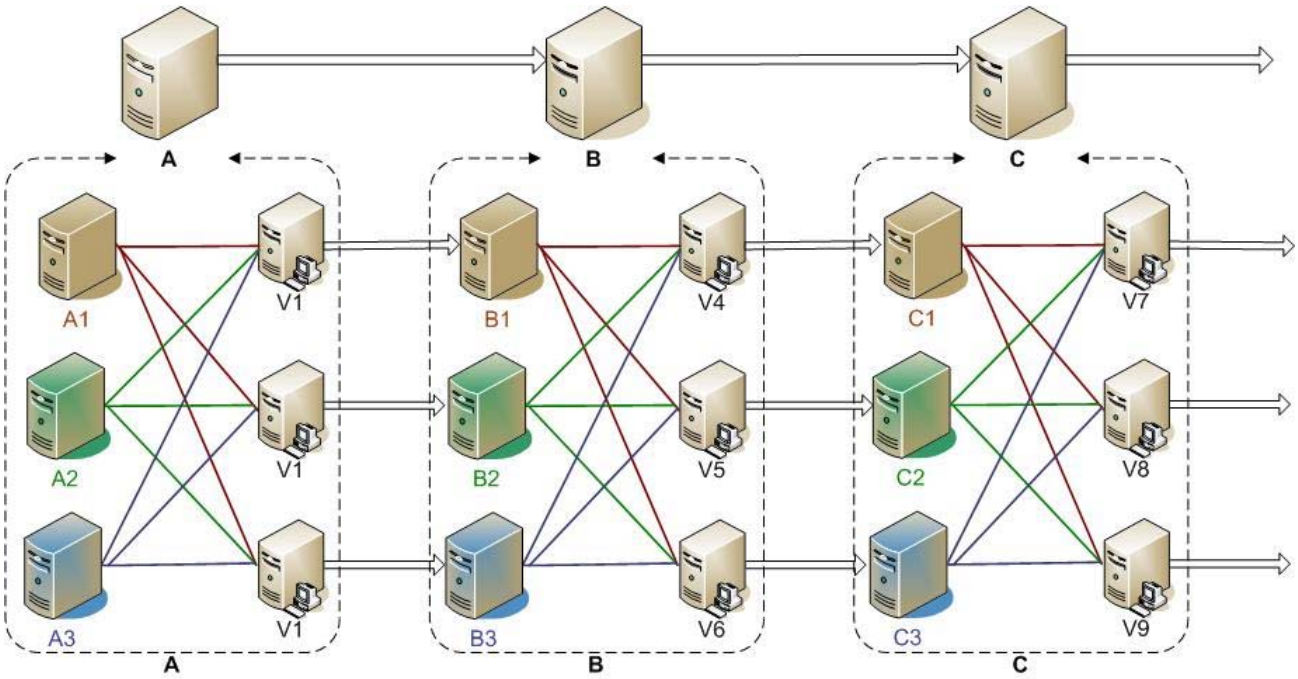
- Wymagany stopień zwielokrotnienia.
- Średnie i najgorsze działanie, gdy nie ma uszkodzeń.
- Średnie i najgorsze działanie, gdy występują uszkodzenia.

Posługując się tymi pojęciami, można wykonywać teoretyczne analizy wielu systemów tolerujących awarie.

7.5. Tolerowanie uszkodzeń dzięki stosowaniu aktywnych zwielokrotnień

[41] Aktywne zwielokrotnienie (ang. *active replication*) jest dobrze znaną techniką zapewniania tolerancji uszkodzeń za pomocą redundancji fizycznej. Aktywne zwielokrotnianie stosuje się od lat w układach elektronicznych w celu zwiększenia ich odporności na uszkodzenia.

Przykładem może być układ potrójnej redundancji modularnej (rysunek 22).



Rys. 22. Potrójna redundancja modularna.

W górnej części rysunku 22 sygnały przechodzą po kolei przez urządzenia A, B i C. Jeśli któreś z nich jest wadliwe, to prawdopodobnie wynik końcowy będzie zły.

W dolnej części rysunku 22 każde urządzenie zostało potrojone. Po każdej części układu występuje potrójny wybierak. Każdy wybierak jest układem z trzema wejściami i jednym wyjściem. Jeśli dwa lub trzy wyjścia są takie same, to wyjście równa się wejściu. Jeśli wszystkie trzy wejścia są różne, to wyjście jest nieokreślone. Projekt tego rodzaju nosi nazwę TMR (ang. *Triple Modular Redundancy* – potrójna redundancja modularna).

Przypuśćmy, że element A2 ulega uszkodzeniu. Każdy z wybieraków V1, V2 i V3 otrzymuje dwa dobre (identyczne) wejścia i jedno oszukane i każdy z nich wprowadza poprawną wartość wyjściową do następnego odcinka. Skutki uszkodzenia elementu A2 zostają w istocie zupełnie zamaskowane, więc wejścia do B1, B2 i B3 są dokładnie takie, jakie byłyby przy niewystąpieniu uszkodzenia.

Zobaczmy co się stanie, gdy oprócz A2 elementy B3 i C1 będą także wadliwe. Również i to zostanie zamaskowane, toteż końcowe trzy wyjścia pozostaną wciąż poprawne.

Z początku może nie być oczywiste, do czego są potrzebne na każdym odcinku trzy wybieraki. Ostatecznie jeden wybierak również mógłby wykryć i przekazać przynajmniej większość sytuacji. Jednak wybierak też jest składową układu i także może być wadliwy. Załóżmy na przykład, że V1 działa błędnie. Wejście do B1 będzie wówczas złe, lecz dopóki reszta działa poprawnie, dopóty B2 i B3 wytworzą takie same wyjścia i każdy z wybieraków V4, V5 oraz V6 wyprodukuje poprawne wyniki dla trzeciego odcinka. Wada elementu V1 nie powoduje innych skutków niż wada powstała w B1. W obu przypadkach B1 wytwarza błędne wyjście, lecz w obu z nich zostaje ono potem przegłosowane na własną niekorzyść.

Aczkolwiek nie wszystkie rozproszone systemy operacyjne tolerujące uszkodzenia stosują technikę TMR. Jest ona bardzo ogólna i powinna dawać dobrą orientację co do tego, czym jest system tolerujący uszkodzenia jako przeciwieństwo systemu, w którym poszczególne składowe są nader niezawodne, lecz którego organizacja nie dopuszcza wad. Technikę TMR można stosować rekurencyjnie, używając jej na przykład do uzyskania dużej niezawodności wewnątrz układu w sposób ukryty dla projektantów, którzy go wykorzystują.

Powracając do tolerowania uszkodzeń w ogólnym rozumieniu aktywnego zwielokrotnienia w szczególności, serwery w wielu systemach zachowują się jak wielkie maszyny skończenie stanowe: przyjmują zamówienia i produkują odpowiedzi. Zamówienia czytania nie zmieniają stanu serwera, natomiast zamówienia pisania powodują jego zmianę. Jeśli zamówienia od każdego klienta prześle się do każdego serwera i wszystkie nadejdą i zostaną przetworzone w tym samym porządku, to po przetworzeniu każdego z nich wszystkie sprawne serwery będą dokładnie w tym samym stanie i dadzą te same odpowiedzi. Klient lub wybierak może połączyć te wszystkie wyniki w celu zamaskowania uszkodzeń.

Istotną kwestią jest, jakie ma być zwielokrotnienie. Odpowiedź zależy od wymaganego stopnia odporności systemu na uszkodzenia. System nazywa się tolerującym k uszkodzeń, jeżeli potrafi przetrwać uszkodzenia k składowych i nadal spełniać swoje założenia. Jeśli składowe (np. procesory) milkną wskutek uszkodzenia, to do zapewnienia tolerowania k uszkodzeń wystarczy, aby było ich k+1. Gdy k z nich po prostu się zatrzyma, wówczas można posłużyć się odpowiedzią pochodzącą z jedynego pozostałego procesora.

Skądinąd, jeśli procesory przejawiają wady bizantyjskie, tzn. kontynuują działanie pomimo niedomagań i wysyłają fałszywe lub przypadkowe odpowiedzi, to w celu uzyskania tolerowania k uszkodzeń potrzeba 2k+1 procesorów. W najgorszym przypadku k wadliwych procesorów może przypadkowo wygenerować tę sama odpowiedź i nawet gdy k+1 pozostałych również wytworzy jednakowe odpowiedzi, klient lub wybierak może po prostu uwierzyć większości.

Teoretycznie można powiedzieć, że system toleruje k uszkodzeń i po prostu dopuścić, by k+1 identycznych odpowiedzi przegłosowywało k identycznych odpowiedzi, jednak w praktyce trudno byłoby wyobrazić sobie warunki, w których można by z pewnością stwierdzić, że k procesorów może się popsuć – lecz w żadnym razie k+1. Zatem nawet w systemach tolerujących uszkodzenia może istnieć potrzeba analizy statystycznej. Podsumowując – aktywne zwielokrotnienie nie jest sprawą trywialną.

7.6. Tolerowanie uszkodzeń dzięki stosowaniu zasobów rezerwowych

Istota koncepcji zasobów rezerwowych polega na tym, że w dowolnej chwili całą pracę wykonuje jeden serwer podstawowy. Jeśli serwer podstawowy ulega awarii, to jego funkcje przejmuje serwer rezerwowy. W idealnych warunkach owo przejęcie powinno przebiegać bez zakłóceń i być dostrzegalne tylko przez system operacyjny klienta, a nie przez programy użytkowe. Podobnie jak aktywne zwielokrotnienia, schemat ten jest

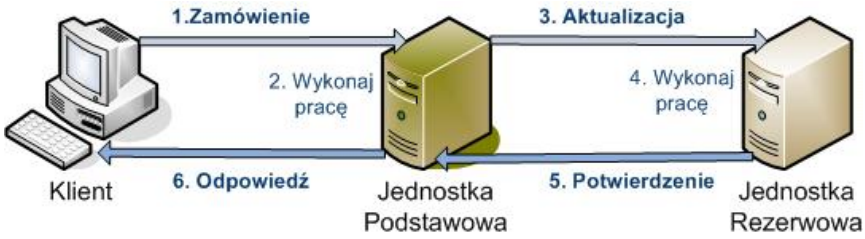
szeroko rozpowszechniony po świecie. Przykładami są: rząd (instytucja wiceprezydenta), awiacja (drugi pilot), motoryzacja (koła zapasowe) itd.

Tolerowanie uszkodzeń uzyskiwane dzięki stosowaniu zasobów rezerwowych ma dwie ważne zalety w porównaniu z efektami stosowania aktywnego zwielokrotnienia:

- jest ono łatwiejsze w czasie normalnego działania, gdyż komunikaty podążają tylko do jednego serwera (podstawowego), a nie do całej grupy. Znika również problem porządkowania komunikatów,
- wymaga ono mniej maszyn, ponieważ w każdej chwili jest potrzebna tylko jedna jednostka podstawowa i jedna rezerwowa. (choć gdy rezerwa zostaje użyta jako jednostka podstawowa, natychmiast staje się potrzebna nowa rezerwa).

Do wad należy zaliczyć złe działanie w przypadku występowania wad bizantyjskich, kiedy jednostka podstawowa fałszywie oświadcza, że działa poprawnie. Złożony i czasochłonny może być także proces przywracania jednostki podstawowej pracy.

Na rysunku 23 przedstawiony jest prosty protokół wykonujący operację zapisu wykorzystujący zasoby rezerwowe.



Rys. 23. Prosty protokół operacji zapisu oparty na wykorzystaniu zasobów

Klient wysyła komunikat do serwera podstawowego, który nie działa, więc komunikat zostaje uaktualniony i wysłany do jednostki podstawowej. Po nadejściu potwierdzenia jednostka podstawowa wysyła odpowiedź do klienta.

Jeśli jednostka podstawowa ulega awarii przed wykonaniem zadania (krok 2), to nie powoduje to żadnego kłopotu. Klient odczeka pewien określony czas, po czym ponowi swoją próbę. Jeśli będzie próbować wystarczająco często, to w końcu dotrze do jednostki rezerwowej i zadanie zostanie wykonane dokładnie jeden raz. Jeśli jednostka podstawowa ulegnie awarii po wykonaniu zadania, lecz przed wysłaniem aktualizacji, to po przejęciu działania przez jednostkę rezerwową i powtórnym otrzymaniu zamówienia, zadanie zostanie wykonane po raz drugi. Jeśli zadanie powoduje skutki uboczne, to może to stanowić problem. Jeśli jednostka podstawowa popsuje się po kroku 4, lecz przed krokiem 6, to zadanie może zostać wykonane trzykrotne: raz przez jednostkę podstawową, raz przez jednostkę rezerwową jako wynik kroku 3 i raz wtedy, gdy rezerwa stanie się jednostką podstawową. Jeśli zamówienia zawierają identyfikatory, to można zapewnić, że zadanie zostanie wykonane tylko dwa razy, lecz zagwarantowanie, że wykona się je dokładnie jeden raz jest trudne lub wręcz niemożliwe.

Teoretycznym i zarazem praktycznym problemem w koncepcji zasobów rezerwowych jest określenie, kiedy powinno nastąpić przejście z jednostki podstawowej na rezerwową. W powyższym protokole jednostka rezerwowa mogłaby wysyłać okresowo do jednostki podstawowej komunikaty "Czy działasz?" Jeśli jednostka podstawowa nie odpowie w ciągu określonego czasu to jednostka rezerwowa mogłaby przejąć działanie.

Jeśli jednostka podstawowa nie uległa uszkodzeniu, lecz po prostu działa wolno to nie ma możliwości rozróżnić tej sytuacji. Istnieje jednak potrzeba zapewnienia, że przy przejmowaniu kontroli przez jednostkę rezerwową jednostka podstawowa rzeczywiście zaprzestaje prób działania w swojej roli. Byłoby najlepiej, gdyby rezerwa i jednostka podstawowa uzgadniały to według jakiegoś protokołu, wszakże trudno jest dyskutować z umarłym.

Najlepszym rozwiązaniem jest mechanizm sprzętowy, za pomocą którego jednostka rezerwowa może wymusić zatrzymanie i uruchomienie jednostki podstawowej.

Można zauważyć, że wszystkie schematy używania rezerw wymagają kłopotliwych do osiągnięcia uzgodnień, podczas gdy aktywne zwielokrotnienie nigdy nie wymaga protokołu uzgodnień (np. TMR).

W odmianie podejścia przedstawiającego protokół operacji zapisu oparty na wykorzystaniu zasobów rezerwowych stosuje się dysk z dwoma portami, wspólny dla jednostki podstawowej i rezerwowej. Gdy w tej konfiguracji jednostka podstawowa otrzymuje zamówienie, wówczas przed wykonaniem czegokolwiek zapisuje je na dysku, gdzie również zapisuje wyniki. Nie potrzebne stają się komunikaty do lub od jednostki rezerwowej. W przypadku awarii jednostki podstawowej jednostka rezerwowa może obejrzeć stan rzeczy, czytając dysk. Wadą tego schematu jest pojedynczy dysk, gdyż w przypadku jego awarii traci się wszystko. Oczywiście ponosząc koszty dodatkowego wyposażenia i kosztem wydajności można zwielokrotnić także dysk i dokonywać wszystkich zapisów na obu dyskach.

7.7. Dochodzenie do uzgodnień w systemach wadliwych

W wielu systemach rozproszonych istnieje potrzeba dokonywania uzgodnień między procesami. Przykładami są wybory koordynatora, decydowanie czy zatwierdzić transakcję, czy nie, podział zadań między pracowników, synchronizacja, itp. Jeśli wszystkie procesory i mechanizmy działają bez zarzutu, to osiąganie takich porozumień jest proste, jednak problem powstaje wówczas, gdy nie wszystko działa poprawnie.

Zasadniczym celem algorytmów rozproszonych uzgodnień jest uzyskiwanie przez wadliwe procesory consensusu w pewnych kwestiach i dochodzenie do niego w skończonej liczbie kroków. Różne przypadki mogą zależeć od parametrów systemowych, takich jak:

1. Czy komunikaty są dostarczane zawsze niezawodnie?
38. Czy procesy mogą ulegać awariom, a jeśli tak, to milknąc, czy też w sposób bizantyjski?
39. Czy system jest synchroniczny czy asynchroniczny?

7.8. Zakończenie

Systemy rozproszone są w pewnym stopniu niewrażliwe na awarie sprzętu, co jest bezpośrednią konsekwencją ich architektury. Uszkodzenie jednej stacji roboczej przerywa pracę jednego użytkownika, lecz nie oddziałuje na usługi świadczone dla innych użytkowników. Nawet awaria serwera nie musi drastycznie wpływać na obsługę użytkowników, jeżeli w systemie działa kilka odpowiedników danej usługi.

W wielu systemach rozproszonych ważne jest tolerowanie uszkodzeń. Można je osiągnąć za pomocą potrójnej redundancji modularnej, aktywnego zwielokrotnienia lub tworzenia zasobów rezerwowych. W przypadku zawodnej komunikacji nie można rozwiązać problemu dwu armii, natomiast problem bizantyjskich generałów można rozwiązać, jeśli sprawnych jest więcej niż dwie trzecie procesorów.

Rozproszone systemy znajdują zastosowanie w coraz większej części naszego życia. Nadzorują lotniska, szpitale, elektrownie i inne strategiczne gałęzie naszego życia. Od ich poprawnego funkcjonowania zależy nieraz życie wielu osób. Już dziś na takie systemy składają się wielkie i rozproszone zasoby sprzętowe, a doświadczenie pokazuje, że jak ze wszystkim, tak i w miarę ich rozwoju będą zwiększać swoje zasoby. Nie mamy tu na myśli już tylko samych zasobów sprzętowych. Działające na nich aplikacje będą coraz bardziej wymagające pod względem szybkości i złożoności obliczeń, będą stawiać coraz trudniejsze zadania, a obok tego będą wymagały zwiększania niezawodności tych systemów. Oczywiście nie jest możliwe skonstruowanie takiego systemu, który byłby całkowicie niezawodny, dlatego systemy rozproszone będą musiały szybko i sprawnie wykrywać i maskować awarie wszystkich swoich składowych sprzętowych i programowych.

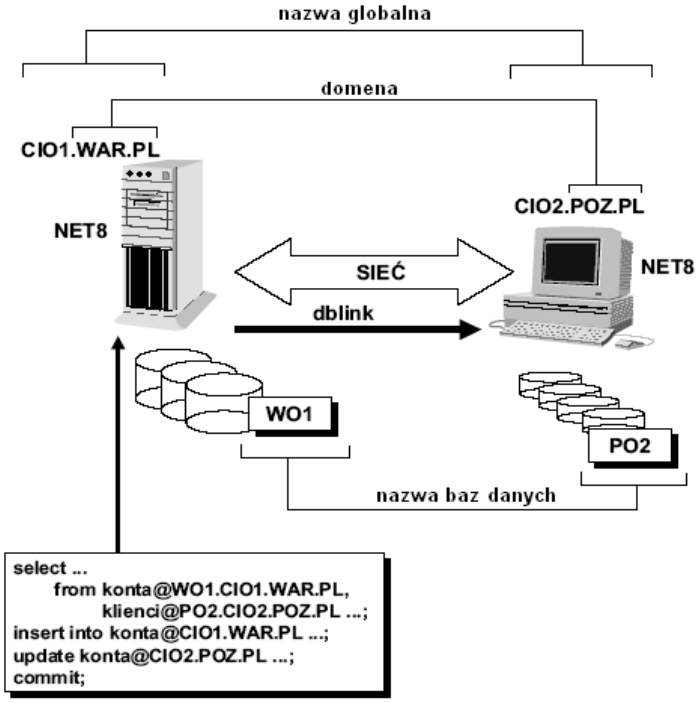
Projektanci systemów rozproszonych mają więc przed sobą dwa zasadnicze problemy. Z jednej strony konstruowanie systemów o jak najmniejszej liczbie awarii, w tym celu badają wszelkie możliwe zachowania usług systemu nie tylko pod kątem ich poprawnego działania lecz także przypadki, w których mogą pojawić się awarie. Z drugiej strony, rozwijają ten zakresu usług, który zwiększa odporność systemu na błędy, np. dostępność wielu komputerów w celu wykrywaniu i maskowania możliwych awarii projektowanych usług. Opis sytuacji, w których może dojść do awarii usługi, nazywa się semantyką awarii usługi (ang. *service failure semantics*). Znajomość semantyki awarii może pozwolić na zaprojektowanie nowych usług, mających maskować awarie. Przykładem może być protokół TCP, skonstruowany w celu dostarczenia niezawodnych usług komunikacji strumieniowej za pomocą potencjalnie zawodnych usług datagramowych dostarczanych przez protokół IP.

CZĘŚĆ II. Praktyczne zastosowanie systemów rozproszonych

1. Systemy rozproszone baz danych

[1] We współczesnym świecie – przy rosnącej z każdym dniem liczbie wielkich korporacji szybko zyskują na znaczeniu systemy rozproszone, które są najlepiej dostosowane do ich wymagań.

Wielkie, korporacje, międzynarodowe firmy, a często nawet o zasięgu globalnym, w poszukiwaniu zysku dążą do jak najlepszego zaspokajania potrzeb wszystkich swoich klientów. Potrzeby te mogą być diametralnie rożne nie tylko na rożnych kontynentach, ale nawet na obszarze jednego państwa. W związku z tym wybierane są takie systemy, które gwarantują podejmowanie decyzji w lokalnych oddziałach firm, ale jednocześnie umożliwiają dostarczanie jak najpełniejszej informacji zarządzanej centralnie po to, aby ich działania były jak najbardziej efektywne. Naturalnym wyborem w takich sytuacjach są rozproszone systemy baz danych.



Rys. 24. Ogólna architektura rozproszonego systemu baz danych.

[2] Systemy rozproszone charakteryzują się pewnymi, sobie tylko właściwymi cechami. Są to:

- konieczność jednoznacznej identyfikacji każdego spośród jego zasobów,
- rozproszony charakter sterowania – każdy węzeł powinien dysponować programowo-sprzętowymi możliwościami realizacji funkcji sterowania siecią, przy czym zwiększone wymagania w stosunku do niezawodności systemu rozproszonego prowadzą do konieczności unikania takich algorytmów sterowania, które bazowałyby na wydzielaniu węzłów uprzywilejowanych w jakiś sposób w porównaniu z innymi węzłami,
- niejednorodność (ang. *heterogeneity*) węzłów sieci – mogą się one różnić np. formatem przedstawienia informacji, oprogramowaniem systemowym i użytkowym.
-

[1] Bardzo ważnym zagadnieniem w technice rozproszonych baz danych jest zapewnienie zgodności danych pomiędzy różnymi fragmentami bazy. Dwufazowy protokół potwierdzeń (ang. *two-phase commit*) polega na tym, że albo transakcja jest poprawnie zrealizowana na wszystkich zasobach biorących w niej udział, albo nie pozostawia po sobie śladów w

żadnym z nich. Wadą takiego podejścia jest jednak mała elastyczność, co uniemożliwia szeroką implementację protokołu przy synchronizacji danych.

Próby efektywnego zaspokojenia potrzeby dostarczania konkretnych danych we właściwe miejsce powodują wybranie jednej z kilku dostępnych technik.

40. Pierwsza z nich polega na tym, że poszczególne fragmenty bazy znajdują się tylko w konkretnych miejscach, natomiast replikowane są jedynie katalogi zawierające informację o położeniu poszczególnych fragmentów. Zalety: brak konieczności synchronizacji wielu kopii oraz zmniejszenie ilości przesyłanych informacji. Wada: trudności w przypadku pracy z dużą ilością danych jednocześnie.
41. Kolejna technika – replikowana baza danych – polega na kopiowaniu całej bazy w jedno lub kilka wyznaczonych miejsc. Dzięki takiemu rozwiązaniu dostęp do danych jest lokalny i unika się konieczności decydowania, jakie fragmenty bazy i gdzie mają zostać skopiowane. Wada: niska efektywność z powodu konieczności składowania ogromnych baz w kilku miejscach oraz dłuższe wyszukiwanie odpowiedzi.
42. Replikowanie na poziomie tabel to następna technika rozpraszania danych, polegająca na kopiowaniu tylko niezbędnych tabel, bez modyfikacji zawartych w nich danych. Stosowana jest tam, gdzie ważne jest oddzielenie danych operacyjnych od aplikacji wspomagających podejmowanie decyzji. Zabronienie użytkownikom uruchamiania zapytań do aktywnych tabel bazy zapobiega konfliktom jednoczesnego dostępu. Trudności mogą pojawić się przy rekonstrukcji logicznej całości bazy danych po ewentualnej awarii systemu.
43. Innym sposobem dystrybucji danych jest tzw. podział danych. Polega on na tym, że dokonuje się podziału i replikacji logicznej tabeli do wielu punktów.
44. Ostatnią techniką jest dystrybucja fan-out, która jest próbą rozwiązania problemu wzrastającej – wraz ze zwiększeniem liczby punktów dystrybucji – trudności zarządzania bazą, co z kolei zwiększa podatność systemu na błędy.

Sposób ten polega na wprowadzeniu kilku pośrednich punktów dystrybucji, dzięki czemu nie trzeba utrzymywać połączenia pomiędzy każdym magazynem danych i każdą zreplikowaną stroną, a wystarczy jedynie utrzymywać połączenia z głównego systemu danych do regionalnych centów przetwarzania, skąd łatwiej jest zarządzać połączeniami do oddziałów terenowych. Dzięki temu ogranicza się liczbę połączeń o dużej szybkości i wysokim koszcie przesłania informacji do punktów dystrybucji.

Rozróżniamy cztery podstawowe typy rozproszonych baz danych:

45. System typu klient-serwer.
46. Jednorodna rozproszona baza danych.
47. Niejednorodna rozproszona baza danych.
48. Federacyjny (wielobazowy) system baz danych.

1.1. Klient-serwer

[3] System typu klient-serwer jest najtańszą rozproszoną bazą danych. Chociaż „klient-serwer" jest bardzo popularnym terminem, nie ma powszechnej zgody co do jego znaczenia.

W praktyce termin „system typu klient-serwer" odnosi się na ogół do lokalnych sieci komputerów PC. Przynajmniej jeden z nich jest wyznaczony do spełniania funkcji bazy danych dla pozostałych komputerów, które działają jako klienci. Baza danych jest przechowywana na serwerze. Interfejs użytkownika oraz narzędzia do tworzenia aplikacji znajdują się na komputerach klientów.

Obecnie komputer pełniący funkcję serwera w tej konfiguracji jest albo serwerem plików, albo serwerem SQL. W wypadku serwera plików zapytanie SQL wyrażone przez klienta spowoduje wysłanie do serwera zapotrzebowania na odpowiednie pliki wymagane do wykonania zapytania. Klient wykona zapytanie, otrzymując odpowiednie dane jako wynik. W wypadku serwera SQL zapytanie SQL zostanie wysłane od klienta do serwera. Serwer wykonuje przesłane zapytanie i w odpowiedzi wysyła tylko dane wynikowe.

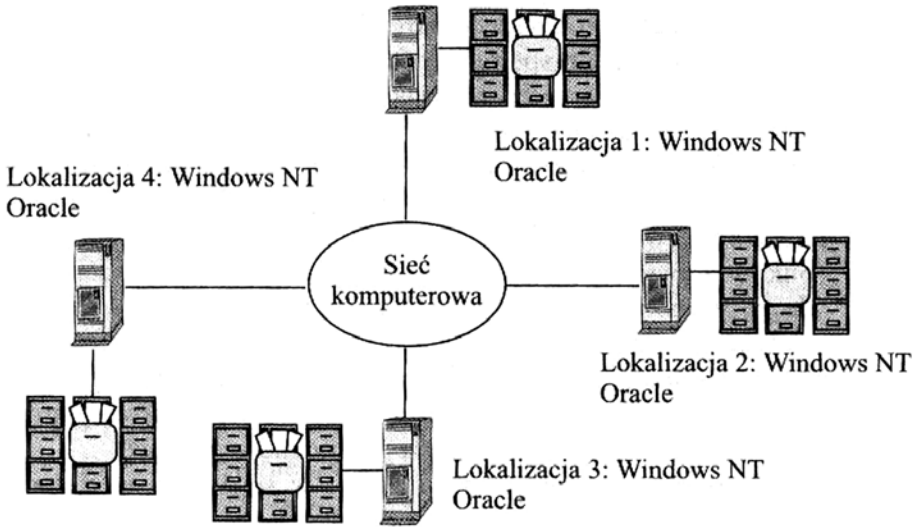
Z powodu zmniejszonego ruchu w sieci wariant z serwerem SQL jest bardziej poszukiwany na rynku.

1.2. Bazy jednorodne

Systemy jednorodnych baz danych mają następujące cechy:

- System Zarządzania Bazami Danych (DBMS) tego samego producenta we wszystkich węzłach.
- Możliwość użycia wbudowanych rozwiązań dla rozproszonych baz danych.

System bazy danych typu klient-serwer dzieli przetwarzanie między klientów i serwer. Systemy rozproszonych baz danych dzielą dane w ramach sieci. Istnieje wiele innych różnic między systemami baz danych typu klient-serwer a rzeczywiście rozproszonymi systemami baz danych. Rozwiązanie klient-serwer możemy zastosować tylko w odniesieniu do systemów składających się z jednego serwera obsługującego wielu klientów. Rozproszone systemy zawierają wiele serwerów. Co więcej, technologia klient-serwer jest stosowana w odniesieniu do krótkich łączy sieciowych. Systemy rozproszone działają na ogół na odległych połączeniach.

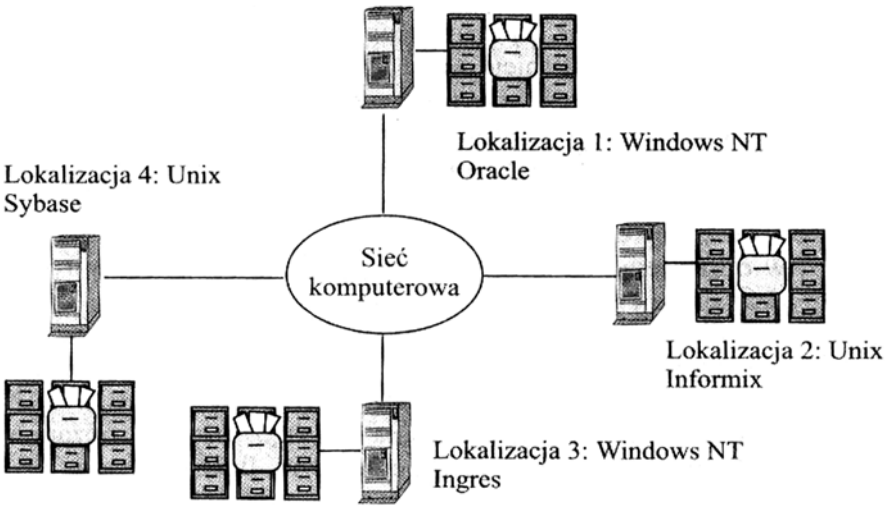


Rys. 25. Przykład jednorodnego systemu baz danych.

W jednorodnej rozproszonej bazie danych dane są rozłożone między dwa lub więcej systemów, każdy oparty na tym samym rodzaju systemu zarządzania bazą danych (np. ORACLE). Na ogół taki rozproszony system działa na tego samego rodzaju sprzęcie pod tym samym systemem operacyjnym. Ilustruje to rysunek 25.

1.3. Niejednorodne systemy baz danych

W niejednorodnej rozproszonej bazie danych konfiguracje sprzętowe i oprogramowania są różne. W jednym miejscu może być ORACLE działający pod Windows NT, w drugim UNIX, a w jeszcze innym Ingres pod Windows NT. Pokazano to na rysunku 26. Obecnie podstawowym sposobem uzyskiwania niejednorodnego systemu jest łącze (gateway). Łącze jest interfejsem z jednego DBMS do drugiego, zwykle dostarczanym przez konkretnego producenta DBMS.



Rys. 26. Przykład niejednorodnego systemu baz danych.

1.4. Federacyjne bazy danych

Federacyjne bazy danych mają następujące cechy:

- względnie niezależne bazy danych,
- niekiedy używane łącznie do wykonania pewnych funkcji,
- z ograniczonym dostępem z zewnątrz poprzez pewien podschemat,
- połączenia między bazami danych typu 1-1.

Federacja, czasem nazywane wielobazowymi, systemy rozproszonych baz danych przypominają polityczny model federacji. Na przykład Szwajcaria jest złożona z pewnej liczby autonomicznych jednostek politycznych. Niekiedy jednostki te zbierają się razem, aby podjąć decyzję na poziomie narodowym.

Federacyjne systemy baz danych składają się z pewnej liczby względnie niezależnych autonomicznych baz danych. Niekiedy zachodzi potrzeba zebrania części lub wszystkich z tych oddzielnych baz danych, aby wykonać wspólną funkcję.

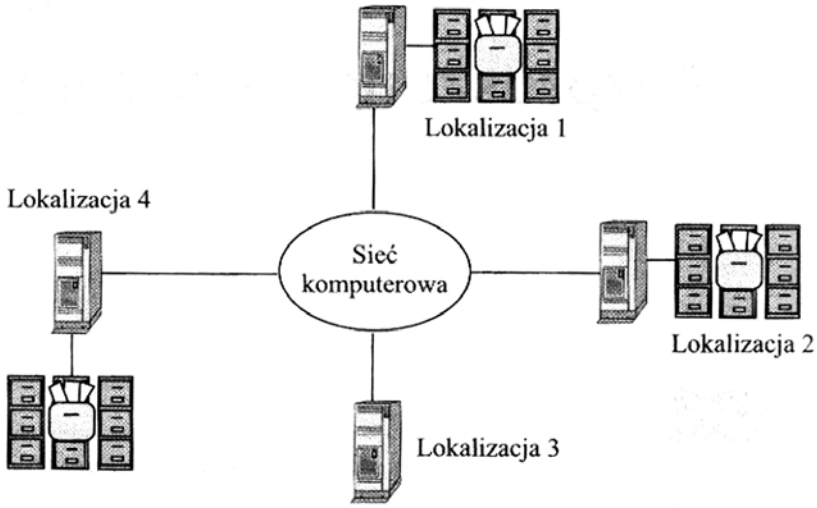
Niektóre aspekty federacji zaczynają się krystalizować z uwzględnieniem połączenia otwartych systemów i standardu dostępu do odległych baz danych. Stanowią one jednak ciągle cel do osiągnięcia, nad którym pracuje wielu producentów systemów baz danych.

[33]Poza tym rozróżniamy:

- Bazy heterogeniczne:
  - Różne DBMS (ang. *Database Management System*) w węzłach,
  - Konieczność stosowania bramek (ang. *gateways*),
  - Problemy współdziałania (ang. *interoperability*):
    - niepełna kompatybilność np. różne dialekty SQL, różne postacie słowników danych itd.,
    - brak kompatybilności np. bazy nierelacyjne, aplikacje spadkowe (ang. *legacy databases*) itd.
  - Niejasna odpowiedzialność producentów.
- Przenośne bazy danych:
  - „Gorsze” wersje DBMS przeznaczone na komputery i inne urządzenia przenośne,
  - Przeznaczone do użycia przez agentów w terenie, bez łączności z centralą baz danych,

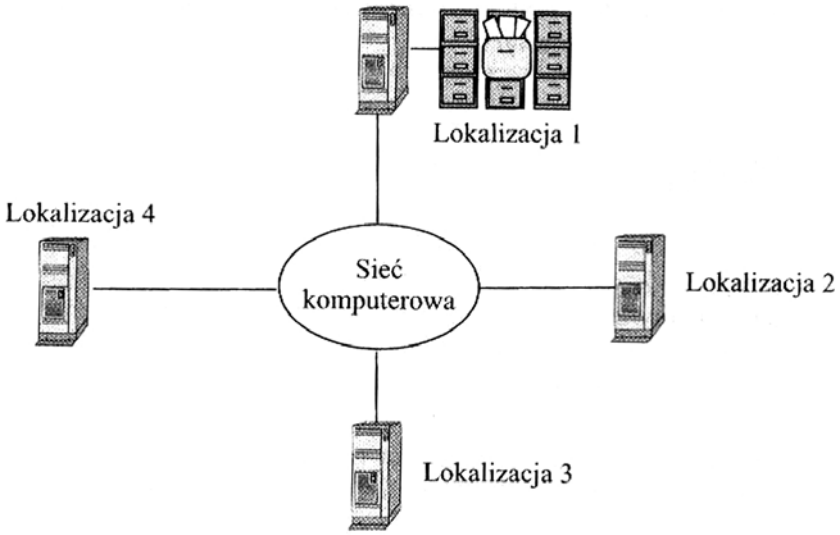
- Wymiana danych z centralną bazą przez specjalnie zorganizowaną replikację.

[4] Rozproszony system baz danych, w którym występuje rozłożenie danych przez ich fragmentację (podział) lub replikację do różnych konfiguracji sprzętowych i programistycznych, na ogół rozmieszczonych w różnych (geograficznie) miejscach przedsiębiorstwa. Rozproszone dotyczy zwykle fragmentacji lub replikacji danych. Fragment danych stanowi pewien podzbiór wszystkich danych całej bazy danych. Replikacja danych stanowi kopię całości lub jakiejś części danych przechowywanych w innej części całej bazy danych.



Rys. 27. Rozproszony system bazy danych.

Rysunek 27 ilustruje ideę systemu rozproszonej bazy danych. Cztery lokalizacje są połączone siecią komunikacyjną. Każda z baz w tych lokalizacjach przechowuje fragment i/lub replikę globalnego systemu rozproszonej bazy danych. Lokalizacja 3 nie przechowuje danych, ale ma dostęp do danych przechowywanych w innych miejscach poprzez sieć komunikacyjną. O rozproszeniu możemy tez mówić w odniesieniu do funkcji.



Rys. 28. System z rozproszeniem przetwarzania.

Rysunek 28 pokazuje system z rozproszeniem przetwarzania. Także tutaj mamy cztery lokalizacje połączone siecią komunikacyjną. Jednakże w tej konfiguracji tylko lokalizacja 1 przechowuje. Lokalizacja 2,3 i 4 działają jako klienci na całej bazie danych, ewentualnie maja własne aplikacje systemów informacyjnych.

Od rozproszonej bazy danych należałoby zwykle oczekiwać, że reprezentuje jeden model danych w przedsiębiorstwie.

[3] Zasadniczym celem systemów bazy danych jest to, aby dla użytkownika wyglądała ona jak jedna, scentralizowana baza danych. Innymi słowy system rozproszonej bazy danych powinien mieć trzy rodzaje przezroczystości:

1. Przezroczystość lokalizacji. Użytkownicy musza wiedzieć, w którym dokładnie miejscu są przechowywane dane. Zatem kierownik, który pragnie się dowiedzieć ilu pracowników zatrudnia biuro w np. Krakowie, nie musi być świadomy tego, że ma do czynienia z rozproszoną bazą danych. Zaletą przezroczystości lokalizacji jest to, że upraszcza programy użytkownika i interfejsu. Dane mogą migrować między lokalizacjami, nie powodując błędów w żadnym z tych programów lub działań. Dane mogą tez migrować wokół sieci w odpowiedzi na zmianę ich użycia lub wymagania dotyczące efektywności.

49. Przezroczystość fragmentacji. Użytkownicy nie muszą wiedzieć, w jaki sposób dane są podzielone. W naszym przykładzie zarządzania kadrami mamy do czynienia logicznie z jedną bazą danych, a fizycznie z trzema jej fragmentami. Dlatego np. kierownik uruchamiający zapytanie np. w Warszawie nie musi wiedzieć, że aby utworzyć łączną listę płac dla firmy, należy wykonać to zapytanie we wszystkich trzech miejscach – np. w Warszawie, Krakowie i Łodzi.

50. Przezroczystość replikacji. Użytkownicy nie muszą wiedzieć, w jaki sposób dane są powtarzane. W przykładowej bazie danych w każdym z trzech biur jest przechowywana kopia informacji o strukturze firmy. Kiedy zachodzi potrzeba okresowej aktualizacji tych danych, użytkownicy nie muszą być świadomi tego, że aktualizacja dotyczy każdego z trzech miejsc. Replikacja jest przydatna, gdyż efektywność rośnie, jeśli aplikacja może działać na lokalnych kopiach, a dostępność jest lepsza dopóty, dopóki co najmniej jedna kopia jest dostępna dla celów wyszukiwania danych.

Kilka cech charakterystycznych systemu rozproszonych baz danych wynika z potrzeby realizacji poniższych celów:

1. Autonomia lokalna. Lokalizacje w systemie rozproszonych baz danych są możliwie najbardziej autonomiczne.

51. Nieopieranie się na lokalizacji centralnej. Wynika to z podstawowego założenia. Jest to cecha pożądana również z tego względu, że oparcie się na siedzibie centralnej mogłoby powodować wąskie gardła w wydajności, a cały system byłby narażony na awarię lokalizacji centralnej.

1.5. Systemy relacyjnych baz danych

Należy podkreślić, że rozwój rozproszonych baz danych nie byłby możliwy bez jednoczesnego rozwoju metod związanych z relacyjnymi bazami danych. Relację bądź tabelę szczególnie łatwo jest podzielić na fragmenty lub wykonać kopię. Proces fragmentacji i replikacji relacji polega po prostu na zastosowaniu odpowiednich operatorów relacyjnej algebry lub relacyjnego rachunku. Fragmentacji pionowej dokonujemy, wykonując rzut na podzbiór kolumn zawierający klucz główny. Odtworzenie oryginalnej relacji polega na zastosowaniu złączenia na podstawie wartości klucza głównego. Fragmentację poziomą uzyskujemy przez zastosowanie operacji restrykcji.

1.5.1. Zalety rozproszenia

[3]Rozproszone bazy danych są bardziej skomplikowane niż scentralizowane bazy danych z powodu dodatkowego czynnika komunikacji sieciowej. Na ogół, procesor komputera działa znacznie szybciej niż urządzenia we-wy, a z kolei operacje we-wy są znacznie szybsze niż komunikacja w sieci. Efektywne zarządzanie przepływem informacji po łączach komunikacyjnych w sieci jest dlatego podstawowym zadaniem systemu rozproszonych baz danych.

Z powodu tych dodatkowych komplikacji rozpraszanie danych nie jest proste. Dlaczego zatem przedsiębiorstwa podejmują się tworzenia rozproszonych baz danych? Niektóre z powodów są następujące:

1. Może być istotne odwzorowanie w systemie danych geograficznego podziału przedsiębiorstwa. Jeśli ma ono charakter narodowy lub ponadnarodowy, to system komputerowy musi uwzględniać podział przedsiębiorstwa na departamenty, oddziały itd.

52. Większą kontrolę nad danymi możemy uzyskać, przechowując je w miejscu, gdzie są one potrzebne. Jeśli biuru w Edynburgu przekażemy odpowiedzialność za dane dotyczące Szkocji, to tylko personel w Edynburgu będzie miał prawo aktualizować te dane.

53. Utrzymanie replikacji danych zwiększa niezawodność systemu. Przechowywanie danych w miejscach, gdzie są one potrzebne, poprawia ich dostępność. W większych bankach bazy danych obsługujące transakcje klientów muszą działać 24 godziny na dobę przez siedem dni w tygodniu. W takiej sytuacji bank nic może sobie pozwolić na zatrzymanie systemu z powodu awarii. Powszechnie używa się dwóch bliźniaczych baz danych: jednej do bezpośredniego wykonywania transakcji, drugiej do utrzymywania wiernej jej kopii. Dane w takich systemach są więc rzeczywiście powielone.

54. Działanie systemu może się istotnie poprawić, jeśli dokonamy prawidłowego rozproszenia danych. Jeśli większość zapytań będzie przekazywana do lokalnej małej bazy danych, to jest duża szansa przyspieszenia dostępu do danych przy operacjach wyszukiwania i aktualizacji.

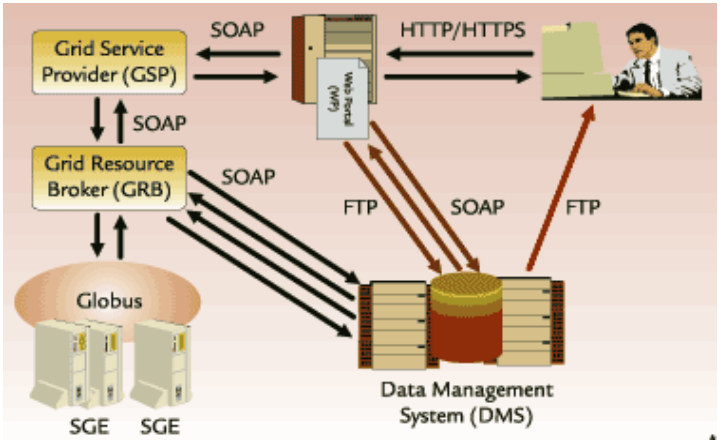
1.6. Podsumowanie

[4] System rozproszonej bazy danych, to system bazy danych, w którym występuje rozłożenie danych przez ich fragmentację lub replikację do różnych konfiguracji sprzętowych i programistycznych, na ogół rozmieszczonych geograficznie miejscach przedsiębiorstwa. Rozproszenie dotyczy zwykle fragmentacji lub replikacji danych. Fragment danych stanowi pewien podzbiór wszystkich danych bazy. Replikacja danych stanowi kopię całości lub części bazy pierwotnej. O rozproszeniu można mówić w odniesieniu do funkcji lub przetwarzania. Z tego powodu do definicji systemów rozproszonych baz danych można włączyć systemy typu klient-serwer. Zasadniczym celem rozproszonego systemu baz danych jest to, aby dla użytkownika wyglądała ona jak jeden scentralizowany system bazy danych: przezroczystość geograficzna, przezroczystość fragmentacji i replikacji.

2. Architektura gridów

[5] Idea zaprzęgnięcia do pracy wolnych mocy obliczeniowych milionów komputerów fascynuje nie tylko naukowców, ale i zwykłych użytkowników pecetów, których miliony wzięły udział w projekcie SETI@home, analizującym nasłuchy kosmosu. Od dojrzałości oprogramowania do zarządzania środowiskiem gridu zależy teraz jego komercyjne zastosowanie w przedsiębiorstwach. Czy kupowanie czasu obliczeniowego stanie się wkrótce bardziej opłacalne od nabywania serwera?

Grid tworzą maszyny liczące połączone w sieć i oprogramowane w taki sposób, by możliwe było współdzielenie zasobów (obliczeniowych, dyskowych) pomiędzy użytkowników gridu. Oprogramowanie gridu umożliwia kierowanie zadań obliczeniowych do wolnych zasobów, koordynowanie równoległej pracy wszystkich maszyn gridu i zapisywanie wyników obliczeń na uwspólnionych zasobach pamięci masowych. Mocną stroną gridów jest możliwość wykorzystania ich do spożytkowania wolnych mocy obliczeniowych maszyn (np. stacji roboczych niewykorzystywanych w nocy).



Rys. 29. Schemat działania gridu w Poznańskim Centrum Doskonałości.

Kolebką gridów są uczelnie. Tam rozwiązani a te przyjęły się najszybciej i korzysta z nich wiele grup badawczych zajmujących się fizyką, chemią i bioinżynierią. Korzyści płynące z doświadczeń związanych z wykorzystaniem gridu szybko dotarły do świata biznesu.

Dotyczy to takich obszarów informatycznych przedsiębiorstw, jak: rozproszone składowanie danych, archiwizacja i udostępnianie aplikacji obliczeniowych, testowanie zabezpieczeń zaawansowanych systemów informatycznych, budowanie eksperymentalnych środowisk testowych dla nowych aplikacji oraz współdzielenie zasobów pomiędzy firmami. Zasoby uczelni połączone w grid mogą być udostępniane firmom potrzebującym mocy obliczeniowych do symulacji pracy projektowanych elementów, modelowania zjawisk fizycznych lub finansowych. Kontynuacja współpracy uczelni z przemysłem technologicznym w zakresie wykorzystania uczelnianych gridów jest na Zachodzie naturalną konsekwencją kooperacji dużych zakładów pracy z ich zapleczem naukowym: lokalnymi uczelniami.

Polska nauka ma dość pokaźny dorobek w pracach nad architekturą grid. W 2002 roku Poznańskie Centrum Superkomputerowo-Sieciowe (PCSS) wspólnie z Sun Microsystems stworzyło Centrum Doskonałości (COE – ang. *Center of Excellence*) firmy Sun. Obecna instalacja gridowa, zarządzana przez PCSS, złożona jest z dwóch serwerów Sun Fire 6800 połączonych w klaster poprzez Polską Sieć Naukowo-Badawczą POL-622/PIONIER z trzecim serwerem SunFire 6800 w Cyfronet AGH w Krakowie. Dodatkowo rozproszona instalacja bazująca na dwóch serwerach dla grup roboczych klasy Sun Fire V880 wspomaga klaster w zakresie replikacji i zarządzania danymi. Stworzone w ośrodkach w Poznaniu i Krakowie oprogramowanie portalowe umożliwiło środowiskom naukowym i badawczym dostęp do zasobów obliczeniowych gridu. Środowisko obliczeniowe Sun zainstalowane w COE działa w konfiguracji klastra, wykorzystując oprogramowanie Sun One Grid Engine.

Centrum Doskonałości powstało w wyniku Projekt Progres. Współpraca poznańskiego ośrodka ze środowiskami gridowymi rozpoczęła się od współdziałania w pracach badawczych dotyczących zarządzania zasobami w gridach. Na konferencji ISThmus w 2000 roku w Poznaniu zostało zawiązane European Grid Forum, na którym spotkali się wszyscy najważniejsi przedstawiciele ośrodków gridowych w Europie. Szefem tego gremium został dr inż. Jarosław Nabrzyski z poznańskiego ośrodka. W ciągu roku amerykańskie Grid Forum i European Grid Forum połączyło się w Global Grid Forum, którego pierwsze spotkanie odbyło się marcu 2001 r. w Amsterdamie. W tym samym czasie poznański ośrodek przedstawił Sunowi propozycję realizacji wspólnego projektu. Projekt został oceniony dobrze i zadecydowano o użyciu łączy optycznych Polskiego Internetu Optycznego PIONIER w instalacji gridowej, łączącej Poznań z Krakowem.

2.1. Wykorzystanie gridów w nauce i biznesie

Wobec pozytywnych ocen projektu przez korporację Sun Microsystems, zainteresowaną nowatorskimi rozwiązaniami w obszarze gridów i portali, pomysłodawcy rozpoczęli jego realizację, kładąc większy nacisk na portalowo-gridową część przedsięwzięcia. Stworzony został portal obliczeniowy dla instalacji serwerów Sun z testowymi aplikacjami bioinformatycznymi.

W listopadzie 2001 r. Centrum podpisało umowę z Sunem, w grudniu – z KBN. W ramach projektu prowadzono prace nad brokerem dla gridu i rozwiązaniami zapewniającymi wysoki poziom bezpieczeństwa w gridzie, a także dodatkową funkcjonalnością systemu operacyjnego Solaris. Ponadto opracowano środowisko portalu umożliwiające łatwy i intuicyjny dostęp do zasobów obliczeniowych przez naukowców-bioinżynierów. Prace nad integracją sprzętu z istniejącą infrastrukturą sieciową dotyczyły instalacji sprzętu, oprogramowania i konfiguracji sieci optycznej. Sun partycypuje w każdej grupie zadań prowadzonych w poznańskim ośrodku. Po zakończeniu prac nad projektem firma prawdopodobnie zdecyduje się na dalsze współfinansowanie projektów w Centrum Doskonałości w okresie półrocznym.

Prace nad gridem prowadzone są także w ramach Programu Ramowego Unii Europejskiej. ACK CYFRONET AGH jest koordynatorem aż dwóch projektów dotyczących gridów: Crossgrid i Gridstart. Gridstart ma harmonizować działania europejskich ośrodków zajmujących się gridem i popularyzować zastosowanie gridów w przedsiębiorstwach, natomiast celem projektu Crossgrid jest upowszechnienie idei gridu w europejskich środowiskach naukowych i przemysłowych oraz wyszukiwanie nowych jego zastosowań. W projekcie Crossgrid bierze także udział Instytut Fizyki Jądrowej w Krakowie.

2.2. Typowo biznesowe wykorzystanie gridów

Firmy Mitsui i WisdomTex we współpracy z Politechniką Tokijską stworzyły technologię wykorzystywaną przez tysiące sprzedawców kosmetyków. Za pomocą lupy i cyfrowego aparatu fotograficznego sprzężonego z telefonem komórkowym kosmetyczka robi zdjęcie fragmentu skóry. Zdjęcie wysyłane jest poprzez e-mail na portal, skąd następnie trafia do analizy. Komputery, głównie firmy Sun, sprzężone w grid, po przetworzeniu obrazu wysyłają na telefon kosmetyczki informację o typie cery klientki, co pozwala na właściwy dobór kosmetyków.

Sun Microsystems udostępnia technologię gridową m.in. przemysłowi motoryzacyjnemu (Ford), farmaceutycznemu (Ribo Targets) i mikroelektronice (Motorola).

IBM traktuje grid bardziej jako ideę niż konkretną architekturę i dlatego wyróżnia typy gridów w zależności od ich zastosowania: dla działów badań i rozwoju, na potrzeby inżynierów i projektantów i do analiz biznesowych. Przy rozwoju gridów IBM współpracuje z partnerami: z Avaki i ISV w zakresie technologii dla przemysłu lotniczego i motoryzacyjnego, z DataSynapse – w sektorze usług finansowych. Zastosowania technologii gridowej, w których partycypuje IBM dotyczą czterech gałęzi przemysłu: lotniczego, motoryzacyjnego, usług finansowych oraz medycyny i farmacji.

Wykorzystanie gridu przez agencje rządowe jest w nomenklaturze IBM jednym z zastosowań przemysłowych (np. SETI@Home).

Inny pogląd na rolę popularnych aplikacji "gridopodobnych" mają przedstawiciele Sun Microsystems. – Napster i SETI były bardzo nośne medialnie, ale ich działalność nie miała wiele wspólnego z rzeczywistymi zastosowaniami oprogramowania gridowego. W ten sposób nie da się tworzyć dużych rozwiązań obliczeniowych Sieci P2P miały duży wpływ na popularyzację idei uwspólnienia zasobów obliczeniowych, ale stosunkowo mały na samą technologię.

Zarówno Sun, jak i IBM wspierają tworzenie gridowego oprogramowania open source. Najbardziej znanym rozwiązaniem jest Globus, wspierany obecnie przez IBM i Microsoft. Sun natomiast przy budowie gridów globalnych wykorzystuje i rozwiązania Globusa, i gridów komercyjnych, np. Avaki. Od kwietnia br. sprzedawane jest profesjonalne oprogramowanie do tworzenia gridów pod nazwą S1GEEE (ang. *Sun ONE Grid Engine Enterprise Edition*). Rozwiązanie, dostępne na platformy Linux i Solaris, kosztuje od 20 do 80 tys. USD (dla uczelni przewidziano 90% rabat).

2.3. Możliwości rozwoju technologii gridowej

Stosowanie gridu pozwala zagospodarować niewykorzystane zasoby zarówno mocy obliczeniowej, jak i zasobów dyskowych. Zasoby dyskowe mogą być współdzielone w obrębie gridu dzięki wykorzystaniu systemów plików: *Andrew File System* (AFS), *Network File System* (NFS), *Distributed File System* (DFS) i *General Parallel File System* (GPFS). Bardziej zaawansowane systemy plików umożliwiają automatyczną duplikację danych, zapewniając redundancję.

Inteligentny scheduler gridu pozwala na podstawie wcześniej ustalonego wzorca postępowania wybrać najlepsze urządzenie do składowania danych dla danego typu zadania. Grid zapewnia równomierne wykorzystanie dostępnych mocy obliczeniowych (ang. *loadbalancing*) i większą niezawodność całego układu obliczeniowego – zadania maszyny, która ulegnie awarii mogą być wykonane przez inną maszynę.

Grid ma jednak także swoje ograniczenia. Przede wszystkim niektóre zadania obliczeniowe i algorytmy nie są skalowalne i nie zawsze zadania obliczeniowe można rozproszyć na większą liczbę procesorów. Problemem może być również wzajemna zależność poszczególnych zadań obliczeniowych wysyłanych na większą liczbę procesorów. Ograniczeniem są wówczas mechanizmy synchronizacji i komunikacji zadań wykonujących się w tym samym czasie. Granice skalowalności wyznacza czasami granica szybkości dostępu do tej samej bazy danych. W zastosowaniach przemysłowych należy brać także pod uwagę ograniczenia zastosowań gridów wynikające z bezpieczeństwa prowadzenia biznesu. W poważnych instytucjach nikt nie zdecyduje się na podłączenie do gridu serwera produkcyjnego, nawet jeśli jego obciążenie jest bardzo małe, bo dostępność usługi biznesowej jest znacznie ważniejsza od dodatkowych korzyści z wykorzystania gridu.

2.4. Równoległe wykonywanie zadań w gridach

Aplikacja przeznaczona do przeniesienia na grid powinna spełniać dwa warunki: musi wykonywać się zdalnie i trafić na maszynę spełniającą wymagania sprzętowe i programowe do obsługi aplikacji. Klasycznym przykładem aplikacji nadającej się do wykonywania w architekturze gridowej jest zadanie wsadowe, składające się z zestawu wejść, które po wykonaniu obliczeń produkują zestaw wyjść. Zadania mogą być wówczas łatwo dystrybuowane w gridzie.

Zrównoleglenie zadań obliczeniowych w gridzie nie jest zadaniem łatwym i nigdy nie odbywa się automatycznie. W niektórych przypadkach problemu nie udaje się rozwiązać przez wydzielenie osobnych zadań obliczeniowych, które można potem dystrybuować na niezależne maszyny (procesory). Większość aplikacji nie wymaga dosłownego zrównoleglenia. Często wystarczy wielokrotne powtórzenie tych samych obliczeń z różnymi parametrami wejściowymi, a do tego zadania grid nadaje się doskonale. Tak jest

w przypadku symulacji zderzeń samochodów: wykonuje się wiele symulacji z różnymi parametrami materiałów, a następnie analizuje wyniki po wykonaniu wszystkich obliczeń.

Do zrównoleglenia obliczeń w gridzie wykorzystuje się standardowe techniki: MPI (ang. *Message Passing Interface*) i biblioteki do obliczeń równoległych, np. PVM (ang. *Parallel Virtual Machine*).

Stworzenie kompilatora, który byłby w stanie wyprodukować kod jednocześnie na wszystkie maszyny gridu to zadanie na przyszłość. Obecnie korzysta się z gotowych, wcześniej przekompilowanych repozytoriów binariów pod różne systemy i architektury. W projekcie Progres użytkownicy mogą korzystać z gotowych aplikacji do rozwiązywania typowych problemów obliczeniowych, dotyczących np. sekwencji DNA czy drugorzędowych struktur białek. W przyszłości będzie też dostępna aplikacja do konstrukcji drzew filogenetycznych. Specyfiką tych problemów obliczeniowych jest ich "ziarnistość".

Identyczne zadania obliczeniowe dla różnych parametrów można wysyłać na poszczególne maszyny wchodzące w skład gridu. W praktyce odbywa się to poprzez przeglądarkę internetową, gdzie użytkownik określa parametry zadania i przekazuje problem obliczeniowy do kolejki zadań. Po pobraniu zadania z kolejki i obliczeniu wyników wszystkie zebrane dane można wizualizować i poddawać analizie. Każda nowa aplikacja musi być przekompilowana pod każdą z platform osobno. Ośrodek w Poznaniu planuje wykonywanie nie tylko obliczeń naukowych, ale także współpracę z przemysłem. Być może w przyszłości na poznańsko-krakowskim gridzie prowadzone będą symulacje zderzeń (tzw. crash testy) autobusów Neoplan.

2.5. Rozwiązywanie problemów z dostępem do serwerów

W terminologii gridowej pojedynczy fragment oprogramowania większej aplikacji wykonywany na maszynie należącej do gridu nazywany jest zadaniem, rzadziej transakcją, jednostką roboczą lub podmisją. Zadania mogą być dystrybuowane i wykonywane w dowolnym punkcie gridu, a także dzielone na mniejsze podzadania. Aplikacja przygotowana jest w taki sposób, by umożliwić rozproszenie i równoległe wykonywanie zadań na wielu maszynach gridu.

Oprogramowanie gridu umożliwia wysyłanie zadań na poszczególne maszyny. W najprostszym przypadku to użytkownik wyszukuje najodpowiedniejszą maszynę (np. najmniej obciążoną) i wysyła tam zadanie. Bardziej zaawansowane gridy zawierają scheduler, który automatycznie wyszukuje najlepszą maszynę do wykonania zadania z kolejki zadań. Zadanie schedulera nie ogranicza się tylko do rezerwacji zasobów. Na bieżąco reaguje on na informacje o dostępności poszczególnych zasobów gridu i kieruje zadania tam, gdzie mogą być najszybciej wykonane. Scheduler zdolny do wymiany zasobów pomiędzy aplikacjami lub zadaniami staje się w ten sposób brokerem zasobów.

Obecnie algorytmy przydziału zadań do konkretnej maszyny uwzględniają zarządzanie zasobami według priorytetów. Z kolejki zadań wybierane jest zadanie o najwyższym priorytecie i wysyłane na węzeł najlepiej odpowiadający potrzebom tego zadania. W schedulerze zaimplementowane są także polityki zarządzania zasobami, zadaniami i użytkownikami. Polityki zarządzania gridem mogą np. uwzględniać ograniczenia w korzystaniu z niego w określonych godzinach, gdy maszyny zajęte są innymi zadaniami. Schedulery muszą na bieżąco reagować na aktualne obciążenie maszyny i kierować zadania na mniej obciążone węzły gridu. Czasami schedulery zorganizowane są hierarchicznie: metascheduler może przydzielać zadania na klaster kilku maszyn będących częścią gridu, a następnie scheduler klastra dystrybuuje zadania na właściwą maszynę. Inteligentne schedulery śledzą wykonanie zadania. Jeżeli komunikacja z węzłem, na którym odbywa się zadanie, została utracona lub minął limit czasu na wykonanie zadania, wówczas zadanie kierowane jest na inną maszynę.

Najważniejszy z informatycznego punktu widzenia zakres prac prowadzonych w Poznaniu dotyczył właśnie zarządzania zasobami gridowymi i stworzenia wielowarstwowego schedulera (metaschedulera). W niższej warstwie zadaniami zarządza Grid Engine i do niego docierają polecenia z warstwy wyższej. Polski ośrodek będzie tworzył oprogramowanie interfejsu

pomiędzy częścią Suna a środowiskiem przygotowywania problemów obliczeniowych. W planach jest także tworzenie middleware'u do komunikacji portalu z schedulerem.

Do osiągnięć poznańskiego ośrodka należy także budowa systemu bezpieczeństwa. Bezpieczeństwo systemu gridowego obejmuje zarówno zabezpieczenie systemu otwartego (dostępnego przez Internet) przed włamaniem, uwierzytelnienie dostępu użytkownika, jak i bezpieczeństwo wewnętrzne w komunikacji usług, realizowane poprzez przyznawanie usługom praw do korzystania z zasobów. Do zabezpieczeń przed włamaniem użyto systemu Valis klasy *Intrusion Detection System* – monitorującego komunikację z systemem i ostrzegającego o potencjalnym zagrożeniu.

2.6. Optymalizacja w systemach gridowych

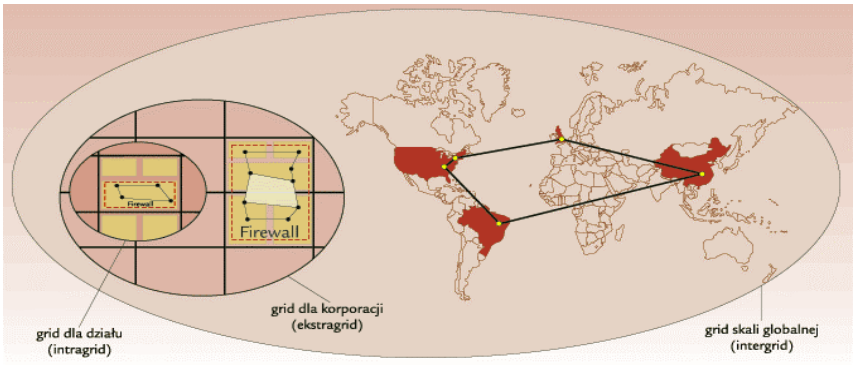
Optymalizacja zasobów gridu w taki sposób, by jak najlepiej wykorzystywać jego potencjał, a zadania wykonywały się możliwie najszybciej, jest problemem złożonym. Na ogół wykorzystuje się algorytmy heurystyczne, czyli takie, które nie dają w sensie matematycznym pewności uzyskania rozwiązania optymalnego.

Niektóre systemy gridowe, np. *scavenging* mają za zadanie spożytkowanie wszystkich dostępnych zasobów w obrębie gridu i kiedy tylko jedna z maszyn zgłasza, że jest za mało obciążona, wówczas węzeł zarządzający przydziela tam zadanie odpowiednie do zgłaszanych wolnych zasobów. *Scavenging* jest zazwyczaj zaimplementowany w taki sposób, by nie zakłócać bieżącej aktywności maszyn, używanych także do innych zadań niż praca w gridzie. W praktyce jednak gridy tworzy się częściej z maszyn przeznaczonych tylko i wyłącznie do tego celu.

Nieodzownym elementem gridu jest oprogramowanie do zarządzania: śledzenia na bieżąco dostępności zasobów i przydzielania zadań wolnym zasobom. Do zadań oprogramowania należy także powiadamianie administratorów o stanie wolnych zasobów i wysyłanie komunikatów alarmowych, jeżeli przekroczone są wartości krytyczne. Najnowsze oprogramowanie do zarządzania gridem potrafi również samodzielnie naprawić niektóre błędy.

Natomiast oprogramowanie umożliwiające podłączenie zasobów maszyny do gridu pozwala m.in. na komunikację z serwerem zarządzającym gridem, przesyłanie zadań do wykonania na maszynę, odbiór wyników i monitorowanie dostępnych zasobów. Zwykle przed przyłączeniem do gridu wykonywana jest procedura identyfikująca i uwierzytelniająca maszynę, choć niektóre gridy nie wymagają specjalnej procedury identyfikującej. Grid dołącza wówczas automatycznie maszynę do puli dostępnych zasobów. Niektóre systemy gridowe mają własny system identyfikacji i login, podczas gdy inne wykorzystują gotowe rozwiązania systemów operacyjnych, na których pracują gridy. Zwykle administrator gridu przydziela prawa wykonywania aplikacji użytkownikom na poszczególnych maszynach.

W przyszłości uruchomieniem aplikacji na gridzie zajmie się *application manager*, system automatyzujący przygotowanie aplikacji od momentu określenia problemu obliczeniowego, poprzez tworzenie kodu aż do serwowania zadań na grid.



Rys. 30. Rozwój technologii gridowej.

2.7. Wspomagające oprogramowanie narzędziowe

Do komunikacji między poszczególnymi elementami gridu i zadaniami służy specjalne oprogramowanie. Poszczególne zadania rozproszone w gridzie mogą komunikować się ze sobą dzięki otwartemu standardowi MPI. Oprogramowanie typu "load sensor" mierzy stopień wykorzystania zasobów (obliczeniowych i pamięci masowej) na poszczególnych węzłach gridu.

Dość popularnym rozwiązaniem jest możliwość wysyłania zadania na grid z każdej maszyny, która należy do gridu. Czasami wykorzystywane do tego celu jest specjalizowane oprogramowanie nazywane oprogramowaniem węzłów podzadań.

Pracownicy poznańskiego Centrum Doskonałości postawili na komunikację z gridem za pośrednictwem portalu. Użytkownik poprzez portal przekazuje dane do tzw. brokera gridowego z informacją o zadaniu (danych wejściowych i pliku wynikowym). Data broker służy do zarządzania danymi, natomiast grid broker wskazuje, gdzie problem będzie obliczany. Poznański broker przekazuje sekwencję danych i problemów obliczeniowych oraz określa lokalizację dla pliku wynikowego. Dzięki temu mechanizmowi użytkownik, korzystając z portalu dostępowego, nie musi znać architektury gridu ani wykonywać poleceń niskiego poziomu uruchamiających zadania.

2.8. Oracle w gridach

Obecność Oracle'a w gridach zapoczątkowało klastrowe rozwiązanie Oracle 9i Real Application Clusters (RAC). Natomiast system bazodanowy przeznaczony dla gridów to baza Oracle 10g oraz Oracle Application Server 10g. Potrzeby przetwarzania kratowego w bazie Oracle 10g realizowane są poprzez rozbudowane funkcje adaptatywnego samozarządzania zasobami z optymalizacją obciążenia, technologie klastrowe (Real Application Clusters 10g) oraz zarządzanie pamięciami masowymi. Oprogramowanie middleware Oracle Application Server 10g umożliwia pracę istniejących aplikacji i serwerów aplikacyjnych oraz usług WWW w systemach gridowych, zaś Enterprise Manager 10g z Oracle Grid Control pozwala na monitorowanie i zarządzanie infrastrukturami gridowymi Oracle'a z jednej konsoli administracyjnej. Popieraną przez Oracle'a alternatywą jest przetwarzanie gridowe, w którym wysoką wydajność systemów bazodanowych można osiągnąć daleko mniejszym kosztem (rzędu 2,4 tyś. USD na procesor i 860 USD/1 GHz przy realizacji z procesorami Intel Xeon 2,8 GHz) na serwerach blade z procesorami Intela, pracujących w środowisku Linuksa. Oto kilka przykładów dużych instalacji gridowych Oracle'a:

- Electronic Arts (18 klastrów linuksowych, 100-150 tyś. jednoczesnych użytkowników).
- CERN (tworzona największa instalacja obliczeniowa na świecie: 3,3 tyś. serwerów, petabajty danych dotyczących cząstek elementarnych).
- Oracle University (240 serwerów linuksowych dla 6 tyś. słuchaczy) i Oracle Outsourcing (ponad 500 serwerów i 80 TB danych).

Na uwagę zasługuje zdecydowanie krótsze podnoszenia systemów gridowych z awarii, porównując czasy ponownego udostępnienia zasobów – rzędu minut w środowisku Oracle 9i – z czasami poniżej 12 sekund dla Oracle 10g. Baza Oracle 10g ustanowiła już rekord wydajności jako pierwsza przekraczając barierę 800 KtpmC. Na serwerze HP Integrity Superdome z 2 procesorami Itanium 1,5 GHz, uzyskano 824164 tpmC przy współczynniku *price to performance* 8,28 USDApmC.

2.8.1. Grid computing

[6] W sensie ogólnym grid computing oznacza przetwarzanie danych, traktowane jako usługa użyteczności publicznej. Inaczej mówiąc, dla klienta nie jest ważne, gdzie są przechowywane jego dane ani który komputer wykonuje zlecenie. Klient powinien mieć możliwość zamówienia informacji lub obliczeń oraz otrzymania wyników w żądanym zakresie i terminie. Można to porównać do dostaw energii elektrycznej. Odbiorca nie wie,

gdzie znajduje się generator ani jak jest zbudowana sieć – po prostu zamawia energię elektryczną i ją otrzymuje. W koncepcji tej chodzi więc o to, aby przetwarzanie danych stało się usługą użyteczności publicznej, powszechnie dostępnym artykułem handlowym.

Taki obraz przetwarzania danych jako usługi użyteczności publicznej jest oczywiście postrzegany „od strony klienta". Natomiast koncepcja grid computing widziana „od strony usługodawcy" oznacza alokację zasobów, współużytkowanie informacji oraz konieczność zapewnienia wysokiej dostępności. Alokacja zasobów gwarantuje, że każdy, kto potrzebuje lub żąda zasobów, otrzymuje to, co mu jest potrzebne, a ponadto pozwala ona uniknąć sytuacji, w której zasoby pozostają niewykorzystane, a zlecenia czekają na realizację. Dzięki współużytkowaniu informacji użytkownicy i aplikacje mają do nich dostęp we właściwym miejscu i czasie. Wysoka dostępność jest niezbędna, ponieważ wszystkie dane i środki przetwarzania muszą być zawsze do dyspozycji, podobnie jak zakład energetyczny musi stale zapewniać dostawę energii elektrycznej. Obecny czas sprzyja tej technologii. Istnieje kilka przyczyn, które łącznie sprawiają, że jej rozpowszechnienie jest nieuniknione.

We współczesnych przedsiębiorstwach ogromne znaczenie ma ograniczanie kosztów. Firmy starają się zmniejszać wydatki oraz zwiększać efektywność procesów i systemów. Osiągnięcie tego umożliwia właśnie technologia grid computing. Pozwala ona lepiej wykorzystywać zasoby przedsiębiorstwa poprzez konsolidację sprzętu oraz eliminowanie obszarów, w których komputery nie są w pełni obciążone. Przedsiębiorstwa mogą tworzyć scentralizowane pule przetwarzania i przydzielać zasoby komputerowe do realizacji zadań priorytetowych.

Wszyscy producenci sprzętu dostarczają komputery typu blade (lub zapowiadają ich dostarczanie). Komputery takie oferują podobną moc obliczeniową, jak systemy w architekturze symetrycznej wieloprocesowości (SMP), ale przy znacznie niższych kosztach oszczędności sięgają nawet 80%. Komputery typu blade można łatwo łączyć w zespoły, dla użytkowników komputerów jest to zwykła codzienność.

2.8.2. Zalety technologii grid computing w porównaniu z superkomputerami

Siatki i superkomputery nie wykluczają się wzajemnie. Superkomputer może być jednym z zasobów siatki. Potrzeba lepszego wykorzystania superkomputerów była jednym z pierwszych motywów podjęcia prac nad technologią grid computing. Technologia ta oferuje rozwiązania alternatywne wobec superkomputerów, takie jak zespoły niedrogich serwerów usługowych typu blade. Zapewnią one dostęp do znacznie większej mocy obliczeniowej przy wielokrotnie niższych kosztach. Problem polega na opracowaniu oprogramowania, które pozwoli efektywnie wykorzystywać zespół serwerów typu blade. W przypadku baz danych mowa jest tu oczywiście o najnowszym produkcie Oracle 10g.

2.8.3. Długoterminowe korzyści z technologii grid computing

Po stronie serwera grid computing zmienia sposób, w jaki przedsiębiorstwa postrzegają zasoby. Zamiast kupować sprzęt do określonych zastosowań, będą one przy zakupach uwzględniać całość potrzeb, a następnie przydzielać zasoby poszczególnym aplikacjom na żądanie lub według ustalonych reguł. Pozwoli to efektywnie wykorzystywać zasoby firmy, a ponadto umożliwi znaczne zmniejszenie kosztów projektowania, wdrażania i zarządzania. Dzięki temu kierownictwo będzie mogło decydować o sposobie wykorzystania zasobów przedsiębiorstwa oraz wprowadzać szybkie zmiany w zależności od bieżących potrzeb.

Dla użytkownika, czyli klienta, grid computing oznacza, że nie musi on się już interesować, w jaki sposób i gdzie są wykonywane obliczenia. Nie musi wiedzieć, kiedy może wykonać określone czynności lub zadać określone pytania. Wyobraźmy sobie na przykład, że suszarkę do włosów można włączyć dopiero po uruchomieniu dużego generatora usytuowanego na skraju miasta, albo że telewizję można oglądać pod warunkiem zgaszenia światła w pokoju! Gdyby tego rodzaju zjawiska wystąpiły w sieci energetycznej, byłby to poważny problem omawiany w krajowych mediach, natomiast pełną automatyzację zadań związanych z zarządzaniem bazą danych Oracle. W ciągu

ostatnich kilku lat interdyscyplinarny zespół złożony z ponad 200 programistów i projektantów z firmy Oracle zajmował się opracowywaniem infrastruktury i narzędzi pomagających urzeczywistnić tę wizję prewencyjnego, automatycznego zarządzania bazą.

Część z narzędzi i udogodnień, jakie powstały w wyniku tych prac, znanych jest z wersji OracleSi i Oracle9i, na przykład wznawialne transakcje, czy dodatkowe pakiety do Oracle Enterprise Managera. Oczywiście wersja 10g wniesie takich udogodnień znacznie więcej. Automatyzacja codziennych, rutynowych zadań związanych z zarządzaniem bazą danych Oracle odciąży administratorów, pozwalając wykorzystać ich doświadczenie i czas w lepszy, ze strategicznego punktu widzenia korzystniejszy dla firmy sposób. Wykorzystywanie dobrze opłacanych pracowników technicznych do realizacji podstawowych zadań związanych z monitorowaniem systemu przynosi firmom i centrom przetwarzania danych raczej nikłe korzyści. Baza danych powinna się sama automatycznie stroić i serwisować – nie powinna wymagać interwencji administratora, chyba że zdarzy się jakaś sytuacja wyjątkowa. Tego właśnie spodziewają się projektanci systemów podejmując decyzję o wyborze bazy danych. W im większym stopniu baza danych może się sama serwisować, tym mniej potrzeba interwencji i tym niższy jest całkowity koszt posiadania systemu w czasie całego cyklu użytkowania. Ważne jest, aby zmniejszyć złożoność systemu, zachowując jednocześnie jego elastyczność. Oczywiście dla rozwiązań specjalizowanych, wymagających strojenia odbiegającego od schematów założonych przez procedury autotuningu. pozostaje możliwość klasycznej zmiany parametrów pracy bazy danych.

3. Systemy rozproszone w Polsce

3.1. System Informatyczny Narodowego Funduszu Zdrowia

[7] System Informatyczny NFZ został zaprojektowany i wdrożony w wersji podstawowej w 1999 roku w 8 regionalnych kasach chorych, tj.: Dolnośląskiej, Lubuskiej, Łódzkiej, Małopolskiej, Opolskiej, Podkarpackiej, Pomorskiej, Śląskiej. Od 1 kwietnia 2003 roku funkcję regionalnych kas chorych przejął Narodowy Fundusz Zdrowia i samodzielne dotąd regionalne kasy stały się oddziałami wojewódzkimi NFZ. System wspomaga pracę na wszystkich szczeblach organizacyjnych NFZ: w centrali, 8 oddziałach wojewódzkich NFZ, 30 delegaturach oddziałów wojewódzkich, 50 biurach terenowych Rejestru Usług Medycznych (RUM) i punktach obsługi. W części wspomagającej ewidencję świadczeń i rozliczanie umów po stronie świadczeniodawców pracuje ponad 18 tys. odbiorców komunikujących się za pośrednictwem sieci rozległej z oddziałami wojewódzkimi NFZ. System Informatyczny NFZ (SINFZ), ze względu na wielkość gromadzonych danych oraz wielość funkcji, sytuuje się wśród największych systemów funkcjonujących w sektorze opieki zdrowotnej.

3.1.1. System w Oddziałach Wojewódzkich NFZ (OW NFZ)

Obecnie na system wspierający pracę oddziału wojewódzkiego NFZ składa się z ok. 40 modułów funkcjonalnych pogrupowanych w podsystemy na które składa się m.in.:

- Zarządzanie Finansami,
- Wspomaganie Struktur,
- Pakiet Świadczeniodawcy,
- Ewidencja Ubezpieczonych,
- Baza Świadczeń.

Podstawowym zadaniem systemu jest zapewnienie kompleksowej obsługi statutowych zadań oddziału, tj.: wspomaganie ewidencji i rozliczanie kontraktów po stronie świadczeniodawców, planowanie i kontrola budżetu, prowadzenie rozrachunków, ewidencja kont bilansowych i pozabilansowych, rozliczanie kosztów, ewidencja zakupów i sprzedaży, ewidencja zatrudnienia, ewidencjonowanie uprawnionych, ewidencjonowanie wykonanych usług i wydanych leków, informowanie uprawnionych o wykonanych dla nich usługach sfinansowanych przez NFZ, ewidencjonowanie świadczeniodawców, ewidencjonowanie ofert, przygotowanie cennika usług i przeprowadzenie konkursu ofert, przygotowanie, monitorowanie i rozliczanie umów ze świadczeniodawcami i refundacja leków, prowadzenie gospodarki materiałowej, ewidencja środków trwałych, ewidencja płac, analizowanie wydatków i przychodów, prognozowanie stanu zdrowia populacji objętej ubezpieczeniem oraz dostarczanie informacji statystycznych wymaganych przez Centralę NFZ i Ministerstwo Zdrowia.

Funkcje raportowe i statystyczne w ramach całego systemu, obok funkcji wbudowanych w poszczególnych modułach, wspomaga system raportowy oparty o technologię WWW – Sprawozdawczość Zarządcza. Organizacja ewidencji świadczeń i rozliczeń opiera się na współpracy systemu oddziałowego z Pakietem Świadczeniodawcy, który oparty jest na następujących założeniach: część ewidencyjna realizuje wszystkie założenia Rejestru Usług Medycznych, gwarancją bezpieczeństwa systemu po stronie świadczeniodawców jest licencjonowany, profesjonalny serwer bazy danych. Komunikacja pomiędzy świadczeniodawcą a OW NFZ odbywa się przez sieć rozległą lub nośniki elektroniczne, wersje oprogramowania i zasoby słownikowe są automatycznie aktualizowane z wykorzystaniem sieci rozległej, potwierdzanie przesyłek po stronie NFZ odbywa się automatyczne, obowiązuje jeden standard komunikacji.

- inicjowanie procesu gromadzenia danych odbywa się z użyciem elektronicznego dokumentu ubezpieczenia (autoryzacja danych) – wersja START,

- do sprawozdań rozliczeniowych są kwalifikowane jedynie dane zgromadzone w module ewidencji świadczeń,
- komunikacja z bazą danych OW NFZ jest możliwa tylko przez aplikacje Pakietu Świadczeniodawcy,
- współpraca aplikacji pakietu z oprogramowaniem własnym świadczeniodawcy jest możliwa w oparciu o interfejs udostępniany przez NFZ, pozwalający na wymianę danych przez pliki; dzięki zastosowanym mechanizmom kontrolnym interfejs ten gwarantuje poprawne prowadzenie ewidencji.

System Informatyczny Narodowego Funduszu Zdrowia uwzględnia regionalne uwarunkowania wynikające z dużego zróżnicowania klientów pod względem:

- wielkości i wewnętrznej struktury organizacyjnej (oddziały obsługujące od 1 do 4 mln ubezpieczonych),
- dokumentów stosowanych do autoryzacji świadczeń (karty elektroniczne, książeczki RUM, brak dokumentów),
- wielkości baz danych.

Oprogramowanie funkcjonalne dla oddziału zaprojektowano w architekturze klient-serwer. Część aplikacji określana jako klient uruchamiana jest na stacjach roboczych z systemem MS Windows. Bazy danych obsługiwane są przez transakcyjne, wydajne programy serwerów baz danych, pracujące w oparciu o język zapytań SQL. Moduły oddziałowe pracują na platformie bazodanowej Oracle; a moduły systemu pracujące u świadczeniodawców NFZ, wchodzące w skład Pakietu Świadczeniodawcy, wykorzystują platformę SQLBase.

3.1.2. System w Centrali NFZ

Podstawowym zadaniem systemu obsługującego Centralę NFZ jest wspomaganie analiz oraz monitorowanie działalności świadczeniodawców w skali NFZ. W większości przypadków źródłem danych w Centrali NFZ są systemy oddziałowe; w wybranych elementach przepływ jest odwrotny: centralny system jest źródłem danych dla systemów oddziałowych – zasoby słownikowe systemu. Podstawowym standardem wymiany danych jest format XML. W części dedykowanej dla centrali NFZ system CL wspiera między innymi następujące obszary działalności:

- monitoring zaopatrzenia w przedmioty ortopedyczne w zakresie umów intencyjnych, ewidencji wniosków, w tym wniosków zrealizowanych, wydanych kart zaopatrzenia (bazy danych umów intencyjnych z realizatorami, potwierdzonych do realizacji wniosków, zrealizowanych wniosków i wydanych kart zaopatrzenia zdrowotnego oraz moduł analiz),
-
- analizę w oparciu o Business Objects,
- zarządzanie centralnymi zasobami słownikowymi,
- centralne rejestry: świadczeniodawców, realizatorów, personelu medycznego,
- monitoring udzielonych świadczeń w zakresie ewidencji i rozliczania umów oraz świadczeń (moduł importu umów zawartych z świadczeniodawcami, moduł importu danych o stanie realizacji umowy, moduł importu danych jednostkowych dotyczących realizacji świadczeń oraz moduł analiz świadczeń, oceny dostępności do świadczeń).

Architektura funkcjonalna systemu oparta jest o technologię grubego klienta, a w wersji klient-serwer o technologię cienkiego klienta z wykorzystaniem serwera aplikacji.

Terminy cienki klient (ang. *thin client*) oraz gruby klient (ang. *fat client*) odnoszą się do mocy i jakości przetwarzania po stronie klienta w architekturze klient-serwer.

Model cienkiego klienta: klient posiada niezbyt wielką moc przetwarzania, ograniczoną do prezentacji danych na ekranie. Przykładem jest klient w postaci przeglądarki WWW.

Model grubego klienta: klient posiada znacznie bogatsze możliwości przetwarzania, w szczególności może zajmować się nie tylko warstwą prezentacji, lecz także warstwą przetwarzania aplikacyjnego (logiki biznesu).

Powyższy podział posiada pewną gradację.

- Model cienkiego klienta jest najczęstszym rozwiązaniem w sytuacji, kiedy system scentralizowany jest zamieniany na architekturę klient-serwer. Wadą jest duże obciążenie serwera i linii komunikacyjnych.
- Model grubego klienta używa większej mocy komputera klienta do przetwarzania zarówno prezentacji jak i logiki biznesu. Serwer zajmuje się tylko obsługą transakcji bazy danych. Popularnym przykładem grubego klienta jest bankomat. Zarządzanie w modelu grubego klienta jest bardziej złożone.

Dla rozwiązań w technologii cienkiego klienta wykorzystywany jest serwer aplikacji IBM WebSphere w wersji 5.1 Express Edition oraz rozwiązania firmy Microsoft .NET. Technologia grubego klienta wykorzystuje Centura Builder 2.0 z dostępem do danych przez ODBC dla bazy DB2 oraz dostępem natywnym dla bazy Oracle, oraz technologii Microsoft .NET z dostępem do baz poprzez OleDB lub Native. W zakresie narzędzi raportowo-analitycznych zakłada się wykorzystanie narzędzia Business Object w wersji 6.5 z dostępem do baz za pomocą sterowników ODBC lub sterowników natywnych.

3.2. Kompleksowy System Informatyczny ZUS-u

[8] Prace nad Kompleksowym Systemem Informatycznym ZUS (KSI ZUS) rozpoczęto w październiku 1997 roku, podpisując umowę z generalnym realizatorem inwestycji, firmą Prokom Software SA. Koncepcję rozproszonego systemu funkcjonującego w 55 oddziałach zastąpiono jednym systemem centralnym. Do umowy wprowadzono opracowanie Programu Płatnika oraz możliwość przekazywania dokumentów do ZUS w formie elektronicznej. Dla dokumentów papierowych zaplanowano wprowadzenie Oddziałowych Ośrodków Przetwarzania Informacji, których zadaniem jest pełna automatyzacja procesu przetwarzania dokumentów papierowych do postaci elektronicznej. Przewidziana w aneksie struktura KSI uwzględniała 7 systemów i 254 moduły, których realizację opisano za pomocą ok. 1500 punktów kontrolnych.

3.2.1. Koncepcja systemu

W wyniku prowadzonych prac i przedstawianych przez wykonawcę kolejnych wersji projektu koncepcyjnego infrastruktury techniczno-systemowej systemu, ZUS zaakceptował projekt zakładający realizację funkcji systemu w trzech rodzajach ośrodków:

- 320 Terenowych Ośrodkach Przetwarzania (TOP) – przyjmowanie dokumentów ubezpieczeniowych, wstępna kontrola i weryfikacja. Punkty te pełnią jednocześnie funkcje informacyjne i dystrybucyjne zarówno dokumentów ubezpieczeniowych, jak i Programu Płatnika,
- 37 Oddziałowych Ośrodkach Przetwarzania (OOP) – przyjmowanie dokumentów ubezpieczeniowych (w formie dokumentów papierowych lub elektronicznych), weryfikowanie i przetwarzanie tych dokumentów do postaci umożliwiającej przesyłanie ich poprzez sieć rozległą ZUS do komputera centralnego,
- Centralnym Ośrodku Obliczeniowym (COO) – przetwarzanie centralnej bazy danych (zasilanej danymi z Oddziałowych Ośrodków Przetwarzania poprzez sieć rozległą ZUS) przez komputer klasy Mainframe.

3.2.2. Konfiguracja sprzętu

O skali realizowanego przedsięwzięcia związanego z zakupami sprzętu świadczy również konfiguracja i wydajność kupowanych urządzeń. System OCR ma wydajność dzienną około 100 tys. wczytanych dokumentów (dla porównania – największe systemy OCR zainstalowane w kraju pracują z wydajnością około 20 tys. dokumentów). Firma IBM, w ramach podpisanej umowy, dostarczyła sprzęt klasy Mainframe o największej wydajności

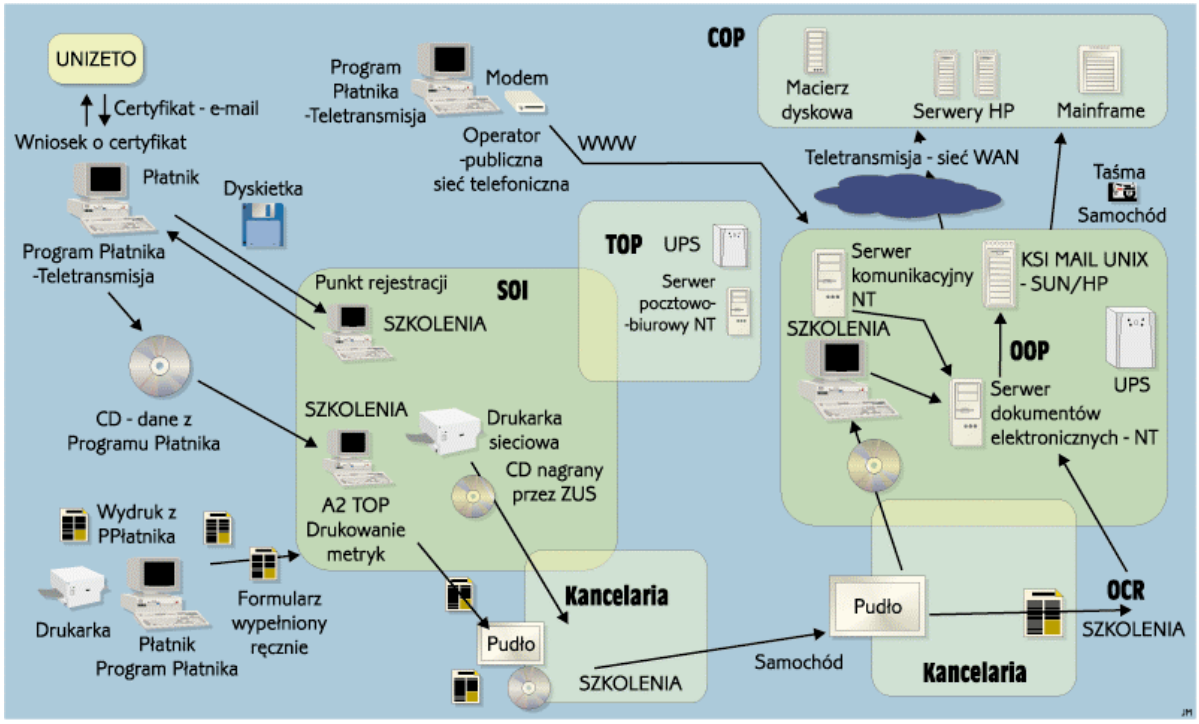
i mocy obliczeniowej nie tylko w Polsce, ale też w Europie Środkowo-Wschodniej. Linia masowych wydruków wraz z linią kopertująco-frankującą ma wydajność 60 mln dokumentów rocznie.

Dla systemów bazodanowych w ośrodkach oddziałowych i w centrali zakupiono 98 serwerów RISC o wydajności od 8000 do 25000 tpmC. Na potrzeby eksploatowanego systemu operacyjnego Windows NT zakupiono 436 serwerów CISC.

Z tym wszystkim wiązała się ogromna praca nad zaplanowaniem i realizacją procesu szkolenia kadry techniczno-eksploatacyjnej do pracy w nowym, zupełnie nie znanym środowisku (system OS390, UNIX, INFORMIX, Windows NT).

Uruchomiono infrastrukturę w 37 Oddziałowych Ośrodkach Przetwarzania. W Centralnym Ośrodku Obliczeniowym pracuje centralny komputer IBM klasy Mainframe oraz baza danych. Testowanie pierwszych modułów aplikacji na tym komputerze rozpoczęło się 1 marca 1999 r. Operator telekomunikacyjny (TP SA) zestawił łącza w 170 jednostkach terenowych (w tym dla wszystkich ośrodków obliczeniowych), co dało możliwość przesyłania danych z OOP do COO i komunikacji z komputerem centralnym. Uruchomiano również pocztę elektroniczną łącząca poszczególne OOP, a następnie kolejne jednostki TOP z COO. Ułatwia to proces komunikowania się ośrodków z centralą na etapie wdrażania systemu i nowej struktury funkcjonalnej jednostek terenowych. W ciągu kilku miesięcy przygotowano infrastrukturę techniczno-programową i wdrożono nowoczesny bezpieczny system przekazywania dokumentów ubezpieczeniowych drogą elektroniczną, wykorzystujący techniki i technologie zapewniające poufność (techniki szyfrowania symetrycznego) oraz integralność i niezaprzeczalność informacji (techniki szyfrowania asymetrycznego z kluczem publicznym służące m.in. do tworzenia podpisu cyfrowego). Konsekwencją zastosowania technologii szyfrowania z kluczem publicznym było powołanie Centrum Certyfikacji, wystawiającej płatnikom, uczestnikom elektronicznej wymiany informacji, certyfikaty klucza publicznego. [45]Sprawnie funkcjonujący KSI ZUS jest także jednym z integralnych elementów głębokich przemian społecznych w Polsce, dzięki któremu może być budowany system nowoczesnego społeczeństwa informacyjnego. ZUS, jako pierwsza instytucja w Polsce, umożliwił obywatelom internetowy kontakt z urzędem państwowym. Upowszechnienie elektronicznej formy rozliczeń z ZUS stanowiło pionierskie wdrożenie infrastruktury klucza publicznego w naszym kraju i przetarło szlak dla innych rozwiązań tego typu. ZUS docenia korzyści płynące z zastosowania Internetu. Dlatego naturalnym kierunkiem rozwoju systemu będzie udostępnienie płatnikom wglądu do konta, wymiany pytań i odpowiedzi oraz weryfikacji i korekty danych przeprowadzanych drogą elektroniczną. Dzięki temu ZUS zamierza umożliwić płatnikom i ubezpieczonym pełny wgląd w rozliczenia, które ich dotyczą. Doświadczenie pokazuje, że można w ten sposób uzyskać oszczędność czasu i pracy (zarówno ZUS jak i płatników) przy wyjaśnianiu wszelkich problemów i niejasności. Konieczność osobistego kontaktu płatnika składek z urzędem powinna być ograniczana do niezbędnego minimum.

[8]O słuszności podjętych decyzji świadczy duże zainteresowanie bezpiecznym przekazem dokumentów elektronicznych. Do 15 lutego 1999 r. wydano blisko 2000 certyfikatów klucza publicznego, a dokumenty zgłoszeniowe do ubezpieczenia społecznego w postaci elektronicznej przekazało ponad 800 płatników składek. Zainteresowanie wdrożeniem analogicznych rozwiązań wyraziło wiele innych instytucji. Pełną ochronę ma także informacja wewnątrz Kompleksowego Systemu Informatycznego ZUS. Prace w tym zakresie, podjęte równolegle z projektowaniem całego systemu, idą w kierunku budowy systemu bezpieczeństwa opartego na szerokim wykorzystaniu kryptograficznych kart mikroprocesorowych, budowy bezpiecznych kanałów przesyłowych oraz kontroli dostępu do wszystkich zasobów systemu informatycznego.



Rys. 31. Schemat przetwarzania dokumentów zgłoszeniowych

[46]Obecnie eksploatowany KSI ZUS, oprócz jednej z największych na świecie instalacji komputerowej klasy mainframe, obejmuje kilkaset instalacji komputerowych w 2 ośrodkach centralnych, 42 ośrodkach oddziałowych i 13 ośrodkach przetwarzania danych, w których utrzymywane jest ponad 270 instancji baz danych. W oparciu o te zasoby realizowanych jest 70 typów procesów przetwarzania użytkowego (ścieżek przetwarzania) wykorzystujących ponad 450 aplikacji. W efekcie około 30 tysiącom pracowników merytorycznym ZUS (użytkownikom KSI) udostępnianych jest 60 typów usług informatycznych umożliwiających użytkowanie systemu.

3.3. Noe.NET w Centralnym Zarządzie Służby Więziennej

[9] Centralny Zarząd Służb Więziennych podległy polskiemu Ministerstwu Sprawiedliwości odpowiada za prawidłowe wykonywanie zastosowanego tymczasowego aresztowania lub środka przymusu oraz wykonanie orzeczonych wobec osób pozbawionych wolności kar.

Noe.NET stworzony został w celu ułatwienia pracy polskiemu wymiarowi sprawiedliwości obejmującemu służbę więzienną, sądy i prokuraturę oraz współpracującej z nim policji. Obieg informacji o zatrzymanych i więźniach był zbyt wolny, a zarządzanie pobytem i transportem więźniów w oparciu o funkcjonujące procedury mało efektywne. Wprawdzie już w 1994 roku Centralny Zarząd Służb Więziennych powołał do życia system KSITA, będący kartoteką osób skazanych, umożliwiający zbieranie informacji o pozbawionych wolności, jednak wprowadzanie informacji do tego systemu odbywało się na podstawie papierowych formularzy przesłanych do KSITA z zakładów penitencjarnych, co oznaczało duże opóźnienie w aktualizacji danych przechowywanych w systemie w stosunku do rzeczywistości. Dużym wyzwaniem było też przeniesienie informacji z docierających około miliona rocznie formularzy papierowych do bazy danych KSITA. Z danych KSITA do dzisiaj korzysta Komenda Główna Policji oraz podległe jej jednostki niższego szczebla, oraz wiele innych instytucji w kraju. W 1998 roku Służba Więzienna zdecydowała się na wdrożenie komputerowego systemu ewidencji skazanych i tymczasowo aresztowanych NOE. Oprogramowanie pierwotnie umożliwiało rejestrację przyjęć, zwolnień i transportów skazanych oraz historii przebiegu ich pobytu, jednak dane do systemu KSITA przekazywane były tradycyjną pocztą w formie wydruków z systemu. Kolejna wersja systemu o nazwie NOE 2000 umożliwiała wykonywanie wydruków dla sądów i prokurator, sporządzanie dynamicznych zestawień i statystyk oraz wyszukiwanie danych według

podanych przez użytkownika kryteriów. Kolejnym krokiem rozwoju systemów informatycznych Służby Więziennej było stworzenie systemu Noe.NET, umożliwiającego wprowadzanie i dostęp do centralnej bazy danych przez pracowników zakładów karnych i aresztów śledczych, policję, prokuraturę i sądy.

Od grudnia 2003 r. do października 2004 r. zrealizowano kontrakt, wykonując system, który pozytywnie przeszedł testy i w chwili obecnej oczekuje na wdrożenie. Zbudowany został on w technologii cienkiego klienta. Wgląd do systemu może mieć każdy autoryzowany użytkownik posiadający dostęp do komputera z systemem Microsoft Windows i przeglądarką Internet Explorer 6.0 oraz zainstalowanym pakietem bibliotek Microsoft Framework wymaganym do działania Microsoft .NET.

Docelowo dostęp do systemu przewidziany został dla 6000 jednoczesnych użytkowników pochodzących z zakładów karnych, aresztów śledczych i innych instytucji. Przed niepowołanym dostępem do systemu chronią karty bezpieczeństwa, dzięki czemu wgląd w dane możliwy jest dopiero po poprawnej autoryzacji przez czytnik karty. Dodatkowe zabezpieczenia stanowią szyfrowanie SSL oraz dedykowana dla Służby Więziennej wewnętrzna sieć WAN.

3.3.1. Rozwiązanie

Wdrożenie systemu Noe.NET umożliwiającego wprowadzanie i dostęp do centralnej bazy danych przez pracowników zakładów karnych i aresztów śledczych, a także policję, prokuraturę i sądy. Wszystkie dane przechowywane są w scentralizowanej bazie danych a dostęp do systemu możliwy jest poprzez przeglądarkę Internet Explorer.

3.3.2. Oprogramowanie i usługi

Oprogramowanie:

- Microsoft Visual .NET,
- Microsoft Internet Explorer 6.0,
- Microsoft Windows 2000 Professional lub wyższy z Microsoft Framework.

Platforma sprzętowa:

- 2 macierze dyskowe (2 x 14 dysków),
- 2 serwery bazodanowe,
- serwer certyfikatów (Windows 2000 Server, Active Directory, DNS, Symantec AntyVirus Client),
- serwer komunikacyjny (Windows 2000 Server, Active Directory, program komunikacyjny z Systemem Informacyjnym Prokuratur),
- serwer kopii zapasowych (Windows 2000 Server, NET Backup Data Center, Symantec AntiVirus Client),
- 5 serwerów aplikacyjnych (Windows 2000 Server, Application Center 2000, Microsoft IIS 5, Microsoft .NET Framework 1.1, Aplikacja Noe.NET, Symantec AntiVirus Client),
- serwer firewall (Windows 2000 Server, Microsoft ISA 2000 Server, Symantec AntyVirus Client),
- 5 stacji administracyjnych (Microsoft Windows 2000),
- router Cisco 3745 FW/VPN/IDS.

3.4. PKO BP

[10] PKO BP to największy działający w Polsce bank, posiadający najliczniejszą rzeszę klientów i największą sieć placówek. Dla tak dużej instytucji finansowej niezwykle istotna jest windykacja i restrukturyzacja należności, które bank musi odzyskać dla dobra swoich klientów i akcjonariuszy. Egzekutor jest systemem stworzonym samodzielnie przez Bank, opartym o technologię .NET i bazę danych Microsoft SQL Server 2000. Egzekutor powstał

na potrzeby Centrum Restrukturyzacji i Windykacji PKO BP, które zarządza wierzytelnościami trudnymi tego Banku. System jest podstawowym narzędziem dedykowanym zarządzaniu wierzytelnościami trudnymi umożliwiającym poprawę efektywności działania Centrum Restrukturyzacji i Windykacji (CRW).

Pierwsza wersja systemu Egzekutor została opracowana w listopadzie 2001 r. przez Marka Sztukowskiego, pracownika Wydziału Restrukturyzacji i Windykacji Oddziału Centrum w Bytomiu. Było to proste narzędzie mające na celu automatyzację najbardziej pracochłonnych czynności administracyjnych wykonywanych przez pracowników wydziału w odniesieniu do masowych należności detalicznych. W okresie niespełna 1,5 roku od powstania pierwszej wersji Egzekutora, stworzony został system analityczny posiadający odpowiednie atesty bezpieczeństwa oraz zapewniający wszechstronną obsługę windykowanych i restrukturyzowanych należności detalicznych. System ten działał testowo w 5 oddziałach Regionu Śląsko-Opolskiego. Ze względu na standardy obejmujące systemy centralne działające w PKO BP, Egzekutor napisany w technologii „klient-serwer" nie mógł zostać wdrożony we wszystkich oddziałach Banku.

Na początku 2004 r. władze PKO BP podjęły decyzję o powołaniu zespołu projektowego w celu opracowania i wdrożenia systemu Egzekutor w całym Banku. Celem projektu było stworzenie systemu informatycznego dedykowanego dla Centrum Restrukturyzacji i Windykacji i uzyskanie korzyści biznesowych poprzez:

- zwiększenie efektywności działania Centrum Restrukturyzacji i Windykacji PKO BP dzięki odciążeniu pracowników od czynności administracyjno-księgowych i sprawozdawczych, dotąd wykonywanych ręcznie,
- przygotowanie danych księgowych w systemie Zorba 3000 do ich prawidłowej konwersji do Zintegrowanego Systemu Informatycznego.

System Egzekutor miał także wspomagać funkcjonalność Zorby 3000 i automatyzować czynności operacyjne, w szczególności rejestrację podejmowanych działań, rozliczanie przychodów i kosztów, edycję standardowej korespondencji, śledzenie historii kontaktów z klientem oraz analitykę i sprawozdawczość.

W skład grupy projektowej weszli przedstawiciele trzech wydziałów katowickiego oddziału Banku: po jednym z Centrum Restrukturyzacji i Windykacji, Centrum Informatyki oraz sześciu z Centrum Rozliczeniowego Banku. Członkowie tego zespołu są również autorami innych aplikacji wykorzystywanych w Banku oraz są w trakcie tworzenia następnych.

Dla Centrali Banku system Egzekutor stał się niezawodnym źródłem informacji o efektywności procesu restrukturyzacji długów i windykacji należności. W centralnej bazie danych przechowywane są informacje o wszystkich sprawach i wszystkich dłużnikach zalegających z płatnościami. Pozwala to na skoordynowanie postępowania wszystkich oddziałów banku wobec dłużnika, który często zalega z płatnościami w więcej niż jednym oddziale Banku. System monitoruje również sytuację ekonomiczno-finansową i majątkową oraz realizację programu naprawczego dłużnika objętego procesem restrukturyzacji długu. Automatycznie dokonywana jest analiza efektywności poszczególnych trybów odzyskiwania należności pod kątem maksymalizacji odzyskiwanej kwoty oraz monitorowane są działania windykacyjne.

3.4.1. Oprogramowanie i usługi

Oprogramowanie:

- Windows Server 2003,
- SQL Server 2000,
- Visual Studio.NET.

Platforma sprzętowa:

- Serwer WWW: Compaq ProLiant DL580 G2 (8x Processor x86 Family 15 Model 2 Stepping 6 GenuineIntel 2,7GHz, Total Physical Memory 2GB),

- VTB: IBM eServer xSeries 365 – [88625 RX] (8x Processor x86 Family 15 Model 2 Stepping 6 GenuineIntel 2,7GHz, Total Physical Memory 2GB),
- Baza danych: HP ProLiant DL740 G1 (16x Processor x86 Family 15 Model 2 Stepping 2 GenuineIntel 2GHz, Total Physical Memory 20GB).

3.5. Amadeus Polska

[11] Amadeus Polska wdrożył system do obsługi rezerwacji i rozliczeń dla biur podróży oparty na serwerze baz danych Microsoft SQL Server i platformie Microsoft Windows Server. Każda spółka lokalna sieci Amadeus (na świecie jest ich 204) prowadzi własną politykę w dziedzinie obsługi informatycznej, ale z zachowaniem globalnych standardów w dziedzinie formatów danych i komunikacji. Amadeus Polska działa w modelu outsourcingowym. Ponad 2 tys. biur podróży z całego kraju łączy się za pośrednictwem Internetu z aplikacją Centauri/Castor, umożliwiającą rezerwację, sprzedaż oraz zarządzanie ofertą turystyczną.

3.5.1. Rozwiązanie

Aplikację Centauri/Castor stworzyła dla Amadeus Polska firma anixe Polska z Wrocławia. Cały system został wykonany za pomocą narzędzi i technologii Microsoft, w tym przede wszystkim Microsoft SQL Server 2005 oraz Microsoft Visual Studio .NET 2003. Uzasadnieniem wykorzystania SQL Server 2005 była dla anixe Polska także możliwość pisania procedur składowanych w języku C#. Dzięki temu aplikacje nie tracą nic z łatwości zarządzania, a jednocześnie, dzięki immanentnym cechom platformy .NET, zyskują większą wydajność i bezpieczeństwo. Wśród wielu drobnych, acz w przypadku anixe użytecznych, nowości SQL Server 2005 zawiera obsługę fonetycznego zapisu nazw geograficznych.

Obsługa C# okazała się przydatna z jeszcze jednego powodu. Wbudowana w SQL Server 2005 obsługa wyszukiwania fonetycznego (biblioteka Soundex) jest niewystarczająca do poprawnego wyszukiwania nazw miejscowości np. (Warsaw, Warszawa, Varsovie). Z tego względu anixe Polska zdecydowała się skorzystać z zewnętrznego algorytmu Metaphone, który integruje się z procedurami składowanymi pisanymi w C#, dzięki temu oparte na nim wyszukiwanie fonetyczne działa tak, jak funkcja wbudowana w sam motor bazy danych SQL Server 2005. Kolejnym istotnym powodem do wykorzystania SQL Server 2005 w projekcie Castor była dla anixe Polska obsługa zapytań hierarchicznych, służących do odwzorowania hierarchii, np. geograficznej struktury sieci sprzedaży czy struktury organizacyjnej na potrzeby systemu ról czy uprawnień. W przypadku aplikacji Centauri/Castor funkcjonalność ta wykorzystana została do opisu struktury organizacyjnej klientów Amadeus Polska korzystających z systemu.

Aby budować systemy o stopniu komplikacji i skali, co system Centauri/Castor, trzeba dysponować technologią, która ułatwia proces projektowania i tworzenia aplikacji. Microsoft SQL Server 2005 zawiera wiele funkcji, dzięki którym wykonanie aplikacji Castor zajęło jedynie kilka miesięcy. Aplikacja pozwala na skalowanie wraz ze wzrostem potrzeb klienta. Ponieważ system finansowy Castor zbudowany na SQL Server 2005 działa bez zastrzeżeń, w niedalekiej przyszłości planowane jest również przeniesienie części rezerwacyjnej na nową platformę.

Korzyści:

- Bogata funkcjonalność biznesowa stworzona w krótkim czasie,
- Gwarancja skalowalności i niezawodności,
- Wysokiej jakości wsparcie techniczne.

Oprogramowanie:

- Windows Server 2003,
- SQL Server 2005.

4. Systemy rozproszone w krajach Unii Europejskiej

W dzisiejszych czasach bardzo trudno wyobrazić sobie życie bez komputerów dlatego nie dziwi fakt szybkiego tempa komputeryzacji firm i urzędów. Pojedyncze jednostki komputerowe potrafią ułatwić oraz przyspieszyć pracę pracowników. Jednak dopiero odpowiednio zbudowana i zarządzana sieć komputerowa potrafi w znaczący sposób przyczynić się do wzrostu wydajności danej organizacji. Odpowiedni podział zadań, rozłożenie obciążenia nawet ogromna międzynarodowa instytucja może działać szybko i sprawnie. Mieszkańcy Unii Europejskiej mogą poruszać się po wszystkich krajach wspólnoty bez żadnych zaproszeń czy pozwoleń, mogą podróżować do dowolnego kraju okazując tylko dowód osobisty. Transport towarów, których miejsce źródłowe i docelowe leży wewnątrz wspólnoty odbywa się bez opłat celnych. Jednakże towary importowane z krajów nie należących do Unii lub eksportowanych poza granice wspólnoty muszą przejść całą procedurę celną. Do niedawna urzędy celne wszystkie czynności związane z tranzytem załatwiały poprzez wypełnianie szczegółowych formularzy papierowych, które następnie przenoszone były pomiędzy odpowiednimi oddziałami. Sposób ten sprawiał, że procedury celne trwały długo a dodatkowo był on podatny na różnego rodzaju nadużycia. Podczas reformy tranzytu w Unii Europejskiej zrodziła się koncepcja opracowania systemu informatycznego do obsługi Wspólnej Procedury Tranzytowej (WPT) realizowanej na podstawie dokumentu SAD, głównie w transporcie drogowym. Tak właśnie powstał NCTS (ang. *New Computerized Transit System*) czyli Nowy Skomputeryzowany System Tranzytowy.

4.1. Zasada działania NCTS

[12] System NCTS stanowi elektroniczne odzwierciedlenie operacji tranzytowej realizowanej zgodnie z przepisami wspólnego i wspólnotowego tranzytu. Umożliwia on wymianę informacji o operacji tranzytowej w czasie rzeczywistym za pomocą elektronicznych komunikatów, eliminując tym samym dotychczasowy czasochłonny sposób przekazywania danych w formie dokumentów papierowych. Komunikaty wymieniane są pomiędzy urzędami celnymi w obszarze międzynarodowym (ang. *common domain*), w obszarze krajowym (ang. *national domain*), gdzie informacje o operacjach dostępne są z poziomu wszystkich urzędów celnych, a także pomiędzy urzędami celnymi a podmiotami realizującymi operacje tranzytowe w systemie NCTS (ang. *external domain*). Operacja tranzytowa jest od początku, tj. od momentu złożenia zgłoszenia celnego, do końca, tj. do zamknięcia operacji tranzytowej, przetwarzana przez system i może być monitorowana na każdym etapie realizacji procedury, tj. w urzędzie wyjścia, tranzytowym, przeznaczenia oraz poszukiwań i poboru należności. Informacje o wprowadzonych do NCTS danych można uzyskać na każdym etapie realizacji operacji WPT, poprzez wyświetlenie zgłoszenia po wprowadzeniu numeru ewidencyjnego operacji tranzytowej, czyli numeru MRN (ang. *Movement Reference Number*).

Działanie NCTS ma charakter interaktywny i wymaga od użytkownika nie tylko znajomości procedury WPT, ale także sporo uwagi.

4.2. Przeznaczenie systemu

Z systemu NCTS korzystają funkcjonariusze celni podczas przetwarzania zgłoszenia celnego i monitorowania operacji tranzytowej oraz podmioty gospodarcze na etapie składania zgłoszenia celnego, unieważnienia, zwalniania towarów do tranzytu, zakończenia operacji tranzytowej w urzędzie przeznaczenia oraz zamykania operacji w urzędzie wyjścia. System NCTS obsługuje zarówno operacje w procedurze normalnej, tj. gdy towar przedstawiany jest w urzędzie celnym wyjścia lub przeznaczenia oraz operacje realizowane w systemie uproszczeń celnych.

Celem wdrożenia NCTS była pełna komputeryzacja operacji tranzytowej WPT w urzędach celnych: wyjścia, tranzytowych i przeznaczenia. Pełna komputeryzacja oznacza, że operacja tranzytowa jest od początku, tj. od momentu złożenia zgłoszenia celnego, do końca, tj. do zamknięcia operacji tranzytowej, przetwarzana i monitorowana przez system.

4.3. Budowa systemu

System posiada architekturę klient/serwer składającą się z centralnego zlokalizowanego w Łodzi (dla Polski) węzła do zarządzania i administrowania systemem na całym obszarze Polski, oraz sieci. Użytkownikami systemu są funkcjonariusze celni. Interfejs użytkownika czyli aplikacja MCC klient instalowana jest tylko na komputerach w placówkach celnych, zaś firmy chcące korzystać z systemu mogą uzyskać dostęp przez stronę internetową lub pocztę elektroniczną. Komunikację polskiego odcinka systemu z systemami innych krajów oraz centrum w Brukseli umożliwia zainstalowany w Warszawie gateway, (zwany też eurobramką specjalnie skonfigurowany i oprogramowany komputer o dużej przepustowości i mocy), a komunikaty przeznaczone dla zagranicznych urzędów celnych przekazywane są przez paneuropejską Wspólną Sieć Komunikacyjną (sieć CCN/CSI).

4.3.1. Wymagania sprzętowe

Stacja robocza:

- Komputer z procesorem PIII lub szybszym,
- RAM 128 MB (zalecane 256 MB),
- Dysk twardy HDD 100 MB pojemności startowej.

Serwer:

- Komputer z procesorem PIII lub szybszy,
- RAM 256 (zalecane 1 GB),
- Dysk twardy 20 GB,
- System operacyjny serwera Windows 98/2000/XP
- Baza danych oparta o:
  - InterBase FireBird – baza darmowa
  - MS SQL Server 7 lub wyższą – baza komercyjna
  - Oracle 8i lub wyższą – baza komercyjna

4.4. Zasięg działania

[47]System został wdrożony w łącznie 29 państwach Unii Europejskiej i krajach EFTA. Każdy kraj uczestniczący w Projekcie Komputeryzacji Tranzytu realizował wdrożenie systemu NCTS we własnym zakresie zgodnie z architekturą techniczną i funkcjonalną opracowaną centralnie przez Komisję Europejską. Systemy krajowe łączą się ze sobą za pośrednictwem domeny wspólnej opartej o sieć CCN/CSI. NCTS łączy ze sobą około 3 tys. placówek celnych w całej Europie, zapewniając sprawniejszą i szybszą wymianę informacji o przebiegu operacji tranzytowej.

[48]Z systemu może korzystać każdy podmiot gospodarczy, korzystający z aplikacji spełniającej określone warunki techniczne umożliwiające komunikację z interfejsem celnym. Aplikacja taka musi być zgodna ze specyfikacjami technicznymi, które zostały udostępnione firmom. NCTS wykorzystuje identyczny standard komunikacji co system Celina, który z powodzeniem został wdrożony przez polskie służby celne. Dla obu systemów został zaprojektowany, a następnie wdrożony podsystem wymiany komunikatów po stronie administracji celnej oparty na XML

4.5. Komunikaty systemu

W systemie NCTS komunikaty elektroniczne muszą być wymieniane przez firmy korzystające z procedury uproszczonej, czyli upoważnieni nadawcy i odbiorcy. Są oni prawnie zobowiązani do rozpoczęcia wymiany komunikatów z systemem NCTS najpóźniej do końca marca 2004 r., o ile ich kontrolny urząd celny jest podłączony do systemu NCTS.

System NCTS opiera się na wysyłaniu komunikatów. Każdy komunikat jednoznacznie definiuje daną operację. Oto możliwe komunikaty wymienianie między aplikacją MCC systemu NCTS a podmiotami gospodarczymi:

- Od podmiotu do urzędu:
  - IE15 – Zgłoszenie deklaracji celnej
  - IE14 – Wniosek o unieważnienie zgłoszenia
  - IE54 – Wniosek o zwolnienie do tranzytu
  - IE07 – Zawiadomienie o przybyciu towaru do upoważnionego odbiorcy
  - IE44 – Uwagi rozładunkowe
- Z urzędu do podmiotu:
  - IE16 – Odrzucenie zgłoszenia
  - IE28 – Zwolnienie do operacji tranzytowej
  - IE60 – Zawiadomienie o decyzji kontroli w miejscu wyjścia
  - IE09 – Decyzja o unieważnieniu zgłoszenia
  - IE51 – Odmowa zwolnienia do tranzytu
  - IE08 – Odrzucenie zawiadomienia o przybyciu towaru
  - IE25 – Zawiadomienie o zwolnieniu towaru z tranzytu
  - IE43 – Pozwolenie na rozładunek
  - IE62 – Odrzucenie wniosku o zwolnienie towaru do tranzytu
  - IE58 – Odrzucenie uwag rozładunkowych

4.6. Usprawnienia wprowadzone dzięki zastosowaniu systemu rozproszonego

Dzięki rozproszonemu po całej Europie systemowi NCTS możliwe jest:

- szybsze składnie zgłoszeń tranzytowych. System NCTS umożliwia składnie zgłoszeń tranzytowych w formie elektronicznego komunikatu IE15. Także unieważnienia zgłoszenia można także dokonywać w formie elektronicznej,
- zautomatyzowanie zwalniania towarów do tranzytu. Zgłoszenie tranzytowe przetwarzane jest przez system. Wiele działań związanych z przetwarzaniem danych przeprowadzanych będzie w sposób automatyczny,
- sprawniejsze przekraczanie granicy w urzędzie celnym tranzytowym. W momencie przedstawienia towarów na granicy, deklarowany urząd celny tranzytowy dysponuje danymi o operacji tranzytowej, co przyśpiesza proces obsługi towarów w urzędzie tranzytowym. Szybkie wyświetlenie zgłoszenia na ekranie funkcjonariusza nastąpi po wprowadzeniu numeru MRN zgłoszenia do systemu za pomocą czytnika kodu paskowego,
- sprawniejsze zakończenie operacji tranzytowej w urzędzie przeznaczenia. W deklarowanym urzędzie przeznaczenia dane o operacji tranzytowej są dostępne w systemie zanim towary zostaną przedstawione w tym urzędzie. System NCTS wspomaga funkcjonariusza celnego w procesie zamykania operacji tranzytowej,
- zwolnienie zabezpieczenia w urzędzie wyjścia po zakończeniu operacji tranzytowej. Niezwłocznie po zakończeniu operacji tranzytowej w urzędzie przeznaczenia, urząd wyjścia informowany jest komunikatem elektronicznym IE18 o wynikach kontroli i na tej podstawie może zamknąć operację o ile została ona zakończona prawidłowo i zwolnić zabezpieczenie. O fakcie tym informowany będzie podmiot podłączony do systemu NCTS,

- szybkość przeprowadzenia operacji tranzytowej przełoży się na zmniejszenie kosztów obsługi operacji. Szybkie zwalnianie gwarancji wpłynie na poprawę płynności finansowej firmy przez szybszy dostęp do środków,

System NCTS przynosi korzyści także podmiotom gospodarczym stosującym procedurę uproszczoną poprzez przyspieszenie i zautomatyzowanie komunikacji z urzędem celnym. Na niektórych etapach przetwarzania zgłoszenia tranzytowego, jeżeli po uzgodnionym z administracją celną czasie nie nastąpi reakcja funkcjonariusza celnego, kontynuowany jest proces operacji tranzytowej (procesy automatyczne). Podmiot korzystać będzie z tzw. „czasomierzy", określających dokładny czas na reakcję funkcjonariusza. Na przykład „Czasomierz oczekiwania na automatyczne zwolnienie" odmierza czas, który posiada funkcjonariusz w urzędzie wyjścia na podjęcie decyzji o kontroli zgłoszenia. Jeżeli firma nie nadeśle wniosku w przewidzianym czasie zgłoszenie zostaje „zawieszone" w systemie.

4.7. Dodatkowe funkcje i możliwości systemu

[49]W systemie NCTS znajduje się także funkcja automatycznej obsługi zabezpieczeń tranzytowych tzw. gwarancje. Nowa funkcjonalność pozwala na bieżące monitorowanie wykorzystania lub obciążenia złożonego zabezpieczenia, także gwarancji wystawionych i zarejestrowanych za granicą. Funkcjonalność ta jest szczególnie korzystna dla przedsiębiorców posługujących się gwarancjami generalnymi lub zwolnieniami z obowiązku składania zabezpieczenia z uwagi na wbudowany mechanizm saldowania kwoty referencyjnej. Moduł zabezpieczeń, którym jest współpracujący z NCTS Ogólnopolski System Obsługi Zabezpieczeń i Pozwoleń, obsługuje następujące typy gwarancji stosowanych w procedurze tranzytu:

- gwarancję generalną (kod systemowy 1),
- zwolnienie z obowiązku składania zabezpieczenia (kod systemowy 0),
- gwarancję pojedynczą w formie oświadczenia gwaranta (kod systemowy 2),
- gwarancję pojedynczą w formie karnetów (kod systemowy 4),
- gwarancję pojedynczą wielokrotnego stosowania (kod systemowy 9).

Zabezpieczenia gotówkowe (kod 3) obsługiwane są w dotychczasowy sposób, tj. numer poświadczenia złożenia zabezpieczenia w formie wydruku z systemu ZEFIR należy wpisywać w pole "Inny odnośnik do gwarancji".

Aby korzystać z posiadanych gwarancji w sposób automatyczny przedsiębiorcy korzystający z systemu NCTS powinni:

- zgłosić się do właściwych miejscowo dla siedziby firmy izb celnych celem zarejestrowania posiadanych zabezpieczeń i otrzymania numeru systemowego gwarancji GRN (ang. *guarantee reference number*). Numer GRN, stanowiący identyfikator zabezpieczenia, należy wprowadzać do zgłoszenia tranzytowego (komunikat IE 15) w polu "GRN",
- oprócz numeru GRN główny zobowiązany otrzyma ponadto kod dostępu do gwarancji, tzw. kod początkowy lub inicjalny umożliwiający autoryzację użycia gwarancji. Kod ten należy zastąpić kodem/kodami identyfikującymi osoby upoważnione do składania zgłoszeń w imieniu głównego zobowiązanego przy użyciu komunikatu IE26. Kody nadane przez głównego zobowiązanego zostaną zarejestrowane w systemie OSOZ i zastąpią kod początkowy. Kod dostępu wprowadza się do zgłoszenia tranzytowego w polu "KodDostępu".

System NCTS umożliwia obsługę operacji tranzytowej przez osoby nie będące głównym zobowiązanym, ani właścicielem gwarancji. W takich przypadkach firma/przedsiębiorstwo (zazwyczaj agencja celna działająca w imieniu klienta) inicjujące operacje na rzecz swojego klienta musi zostać zarejestrowana w podsystemie Danych Referencyjnych PDR z zakresem uprawnień "Agencja celna", z jednoczesnym zakresem "Export/Import". Zakres uprawnień "Agencja celna" stanowi dodatkową funkcję systemową umożliwiającą składanie zgłoszeń w imieniu głównych zobowiązanych lub osób trzecich (np. osób

fizycznych posługujących się zabezpieczeniem gotówkowym). W przypadku gdy zgłaszający nie jest głównym zobowiązanym ani właścicielem gwarancji wypełnia się pole "Przedstawiciel" w komunikacie IE15.

Dzięki zastosowaniu się do specyfikacji europejskich system NCTS może być łączony z innymi systemami związanymi z tranzytem. W Polsce został on zintegrowany z systemami CELINA i ZEFIR.

4.7.1. System CELINA

[13] System CELINA wspomaga pracę organów celnych nie tylko w zakresie obsługi zgłoszeń celnych, a od 1 maja 2004 r. jego funkcjonalność została rozszerzona. Oprócz obsługi zgłoszeń celnych jest on również wykorzystywany m.in. do rejestrowania i przetwarzania danych pochodzących z przekazywanych organom celnym elektronicznych deklaracji INTRASTAT, tj. deklaracji statystycznych dotyczących obrotów towarowych pomiędzy państwami członkowskimi Wspólnoty.

System CELINA nie jest systemem "sprzężonym" z systemami Unii Europejskiej. Służy on do ułatwienia i przyspieszenia obsługi zgłoszeń celnych. Działa on we wszystkich placówkach celnych (urzędach i oddziałach celnych).

Komunikacja z podmiotami (zgłaszającymi) zapewniona jest poprzez strony webowe CELINA WEB-CEL (dla procedur standartowych) i CELINA OPUS (dla procedur uproszczonych), tworzące tzw. Wrota Celne (ang. *customs gateway*). Ponadto, istnieje możliwość korzystania przez podmioty z poczty elektronicznej lub dyskietki/płyty CD a także możliwość złożenia zgłoszenia w formie dokumentu papierowego, z którego dane są wprowadzane do systemu CELINA przez funkcjonariusza celnego.

4.7.2. System ZEFIR

[14] Prace nad systemem ZEFIR rozpoczęły się w 1998 roku. 23 kwietnia 1998 r. pomiędzy Głównym Urzędem Ceł i firmą Systemy Komputerowe Główka S.A. z Bielska Białej podpisana została umowa na realizację prototypu oraz pilotowe wdrożenie systemu. Odbiór prototypu nastąpił 23 sierpnia 1999 r., natomiast 1 września 1999 r. rozpoczęła się jego próbna eksploatacja w Urzędzie Celnym w Krakowie (obecnej Izbie Celnej w Krakowie), a 1 października w Nowym Targu. ZEFIR od dnia 1 stycznia 2000 r. w pełni funkcjonował we wszystkich jednostkach krakowskiego Urzędu Celnego. Jego ostateczny odbiór nastąpił 19 stycznia 2001 r. Wdrożenie ogólnopolskie rozpoczęto w czerwcu 2002 r., a zakończono w styczniu 2004 r., obejmując swym zasięgiem wszystkie Izby Celne wraz z podległymi jednostkami.

ZEFIR – System Rozliczeń Celno-Podatkowych i Finansowo-Księgowy jest systemem informatycznym kompleksowo obsługującym całość spraw związanych z finansami i rachunkowością Polskiej Służby Celnej. Projekt, wdrożenie oraz bieżące prowadzenie systemu realizowane jest przez funkcjonariuszy Izby Celnej w Krakowie. System Zefir jest pierwszym ogólnopolskim systemem jaki był wdrożony w Służbach Celnych RP. Dzięki modularnej budowie stosunkowo łatwo przeprowadzono integrację z systemem NCTS.

5. Laboratoria wirtualne jako przykłady systemów rozproszonych

[15] Wirtualne laboratorium jest heterogenicznym, rozproszonym środowiskiem, które umożliwia grupie naukowców znajdujących się w różnych miejscach na świecie wspólną pracę nad wspólną grupą projektów. Podobnie jak każde inne laboratorium narzędzia i techniki są specyficzne dla danej dziedziny nauki.

Laboratoria wirtualne są przeważnie zlokalizowane przy dużych centrach naukowych i ośrodkach akademickich. Instytucje te mają znaczny budżet na zakup specjalistycznej aparatury pomiarowej oraz dysponują bardzo szybkimi łączami internetowymi. Pomimo połączenia z pewnymi aplikacjami teleimersji, wirtualne laboratorium nie zakłada a priori potrzeby dzielenia środowiska pracy. Laboratoria wirtualne ułatwiają i przyspieszają kształcenie, wymianę poglądów, prowadzenie wspólnych badań, a w końcu zdalne, fizyczne udostępnienie zasobów aparaturowych centrów naukowych. Idea ta jest szczególnie atrakcyjna dla nauk doświadczanych i technologii, a w szczególności: fizyki, chemii, biologii strukturalnej, medycyny doświadczalnej, radioastronomii czy wreszcie szeroko rozumianej inżynierii. Laboratoria wirtualne są budowane aby umożliwić: zlecanie zadań do wykonania, zarówno eksperymentów rzeczywistych jak i obliczeniowych, równoważenie obciążenia zasobów sprzętowych i ludzkich, rozliczanie czasu i postępu pracy użytkowników, digitalizację wyników pomiarów, komunikację między zespołami naukowców prowadzących powiązane ze sobą badania jak i komunikację z obsługą urządzeń, rezerwację czasu przeprowadzenia doświadczenia. Ważną ideą laboratoriów wirtualnych jest wyrównanie szans osób pracujących w dużych ośrodkach naukowych i tych co pracują poza nimi.

[50]Z punktu widzenia wirtualnego laboratorium a szczególnie narzędzi do komunikacji typu teleimersja krytycznym parametrem jest tutaj opóźnienie. Dlatego też wielodyscyplinarne centrum przetwarzania informacji (w tym, prowadzenia obliczeń) powinno być ściśle powiązane z dostępem do sieci szerokopasmowej. Wskazane byłoby również powiązanie komputerowego systemu szeregowania zadań z usługami rezerwacji przepustowości. Kolejnymi krytycznymi parametrami, z punktu widzenia laboratorium wirtualnego są protokoły multikastowe i niezawodność technologii we współpracy z naturą (specyfiką) eksperymentów w wirtualnym laboratorium, gdzie ludzie, zasoby i obliczenia mogą być bardzo rozproszone. Strumienie danych (informacji) w tych eksperymentach mogą być kombinacją głosu (dźwięku), plików video, danych dostarczanych w czasie rzeczywistym z urządzeń badawczych oraz potężnej dawki danych ze źródeł symulacji i wizualizacji.

Aplikacje muszą zapewniać dostęp do danych z wielu heterogenicznych źródeł informacji. Dla przykładu, mogą one pochodzić bezpośrednio z danych eksperymentalnych opartych o wykorzystanie aparatury naukowej jak również z programów symulacyjnych. Podstawowym źródłem informacji, szczególnie ważnym dla burzliwie rozwijającej się bioinformatyki, która w dobie sekwencjonowania genomów z różnych organizmów, w tym człowieka, staną się systemy pamięci masowych. Mogą one być oparte zarówno o olbrzymie, dedykowane dla celów badawczych bazy danych, posadowione w krajowych centrach komputerowych, jak i te udostępnione przez naukowców z ich personalnych stacji roboczych i PC. Przetwarzanie danych będzie sterowane i monitorowane przez indywidualnego eksperymentatora lub rozproszony, pracujący we własnych laboratoriach zespół badawczy, dzięki możliwościom oferowanym przez laboratorium wirtualne.

Ważnym Każdy z zespołu jest ekspertem od poszczególnego modułu symulacji, analizy danych i wizualizacji. Grupa naukowców musi współdzielić widok symulacji i interaktywnie nią sterować.

Inny przykład o profilu wielodyscyplinarnym, wytwarzający pewien produkt. W tym przypadku firma zajmująca się produkcją dużego i skomplikowanego produktu, takiego jak samolot musi być zdolna do kierowania procesem symulacji interaktywnie z danymi projektowymi zawierających techniczne i wykonawcze specyfikacje. Symulacje i projektowanie mogą wymagać równoczesnego dostępu do setek obliczeń pomocniczych,

które są dostarczane podwykonawców z innego miejsca. Wynik jest wielodyscyplinarną optymalizacją. gdzie większość efektywnych i bezpieczniejszych produktów może być tworzonych zgodnie ze specyfikacją klientów.

Trzecim przykładem jest system przewidywania pogody, który zawiera dane satelitarne, dużą liczbę wejściowych czujników oraz dużą liczbę symulacji dla krótkiego i średniego okresu przewidywania. Różnicą w tym przypadku jest przewidywanie jakości powietrza poprzez wirtualne laboratorium, które łączy model pogody z modelem cyrkulacji oceanów i chemicznymi zanieczyszczeniami z danymi z sensorów umieszczonymi na ziemi i w powietrzu. W każdym laboratorium lokalni naukowcy mogą sugerować, biorąc pod uwagę obecne warunki, kiedy wyłączyć tymczasowo pewne typy produkcji przemysłowej w celu uniknięcia potencjalnego kryzysu związanego z jakością powietrza.

5.1. Architektura laboratorium wirtualnego

Ogólna architektura laboratorium wirtualnego przewidziana jest tak, aby zapewnić dostosowanie jej do różnego typu urządzeń laboratoryjnych. Komponenty wirtualnego laboratorium mogą zawierać różne elementy w zależności od tego, do jakiego typu eksperymentów będzie ono używane. Można oczywiście wyróżnić elementy wspólne, które mogą lub nawet powinny występować w każdej konfiguracji laboratorium i takie, które są specyficzne dla pewnego typu laboratoriów. Systemy laboratoriów wirtualnych mogą współpracować z innymi systemami laboratoriów wirtualnych.

Zazwyczaj cechą wspólną laboratoriów wirtualnych jest dostęp za pomocą Internetu, jak również najczęściej stanowią one pewien rodzaj portalu. Takie rozwiązanie sprawia, że główny warunek wirtualnego laboratorium, jakim jest dostępność z każdego miejsca na ziemi, jest bezwarunkowo spełniony.

Kolejny element stanowi maszyna obliczeniowa (serwer komputerowy) będąca w stanie poradzić sobie z potężnymi symulacjami i redukcją danych. Przykład dotyczy regionalnych centrów obliczeniowych, bardzo szybkich o dużej przepustowości sieci komputerowych, systemów o dużej wydajności w uniwersyteckich kampusach oraz w korporacjach jak i instytucjach rządowych.

Następny element laboratorium wirtualnego stanowią bazy danych, które zawierają informacje specyficzne dla poszczególnych aplikacji takich jak symulacje początkowe, warunki graniczne, obserwacje eksperymentalne, wymagania klienta, ograniczenia produkcyjne podobnie jak rozproszone specyficzne dla aplikacji zasoby takie jak na przykład repozytoria genomu ludzkiego. Zawartość tych baz danych może być modyfikowana dynamicznie jak i mogą to być bazy o architekturze rozproszonej. Należy zakładać, że będą one mogły przechowywać bardzo duże ilości informacji i operować na nich.

W architekturze laboratorium wirtualnego przewiduje się zazwyczaj cztery warstwy: dostępową, gridową, nadzoru i zasobów.

W warstwie dostępowej odbywa się:

- realizacja dynamicznych scenariuszy pomiarowych, zapewniają one realizację i sterowanie zleconym przez użytkownika szeregiem eksperymentów,
- zarządzanie laboratoriami i użytkownikami, wykorzystuje się tu specjalistyczny zestaw narzędzi przeznaczonych użytkownikami i ich profilami,
- prezentacja danych, dotyczy ona zarówno wyników przeprowadzonych eksperymentów umieszczonych w systemie zarządzania bazą danych jak i danych napływających na bieżąco,
- komunikacja miedzy użytkownikami, z wykorzystaniem specjalnych narzędzi przeznaczonych do realizacji tego celu.

W warstwie gridowej realizowane są następujące usługi ogólne:

- w Centrum Certyfikacji, generowanie certyfikatów i sprawdzanie ich ważności, przechowywanie i monitoring praw dostępu, w skład tej warstwy wbudowane są

narzędzia służące do autoryzacji i uwierzytelniania użytkowników podczas procesu logowania,

- szeregowanie globalne, wybór przyrządów laboratoryjnych i serwera obliczeniowego, na którym ma być zrealizowane zlecone zadanie,
- zarządzanie danymi, magazynowanie wyników przeprowadzonych eksperymentów i obliczeń oraz dokumentacji elektronicznej,
- transportowanie danych do maszyny docelowej, kontrola nad przebiegiem tej transmisji,
- bramka do gridu, następuje komunikacja z brokerem gridowym, zapewnia odbiór zadań spoza laboratorium i przesłanie tegoż zadania na konkretny przyrząd.

W warstwie nadzoru realizowany jest szereg usług specyficznych:

- szeregowanie lokalne, jest ono realizowane w obrębie danego urządzenia, z uwzględnieniem parametrów i priorytetów szeregowania,
- monitoring zasobów, kontrola nad wykorzystaniem zasobów oraz aktualnym ich obciążeniem i stanem zadań,
- rozliczanie użytkowników, dostarczanie informacji o wykorzystywaniu zasobów laboratorium przez użytkowników.

W warstwie zasobów pracują:

- przyrządy laboratoryjne, zarówno aparatura jak i oprogramowanie potrzebne do wykonywania eksperymentów,
- serwery obliczeniowe i wizualizacyjne wraz z dedykowanym na nie oprogramowaniem.

Z punktu widzenia komunikacji w laboratorium wirtualnym, możemy z kolei wyróżnić trzy warstwy:

- warstwę interfejsu klienta, warstwa ta odpowiedzialna jest za zapewnienie dostępu do serwera aplikacji z dowolnego miejsca na świecie z wykorzystaniem Internetu, odbywa się to w precyzyjnie określony sposób i jest możliwe dzięki wykorzystaniu specjalnych standardów komunikacji z serwerem aplikacji, ponadto do zadań klienta należy autoryzacja i weryfikacja użytkowników podczas procesu logowania, zapewnienie dostępu do modułów administracyjnych laboratorium jak i umożliwienie pracy grupowej,
- agenta, innymi słowy serwera aplikacyjnego, do jego zadań należy odbieranie zadań od użytkowników, a następnie przekazanie ich do maszyny obsługującej konkretne urządzenie i odesłanie wyników obliczeń do użytkownika. Do zadań agenta należy również zarządzanie zleceniami skierowanymi do urządzeń podłączonych do laboratorium wirtualnego oraz do systemów do symulacji i obliczeń,
- serwera urządzenia, do jego zadań należy odebranie zlecenia od agenta, zlecenie jego wykonania, następnie odbiór wyników i odesłanie ich do agenta.

Jest to architektura zapewniająca pełna funkcjonalność laboratorium i jest odpowiednikiem modelu architektury klient-agent-serwer.

5.2. Narzędzia do współpracy i porozumiewania się

[50]Zaliczamy do nich chat, systemy przekazujące dźwięk, wideokonferencje i teleimersję.

5.2.1. Administrator laboratorium

[51]Administrator laboratorium jest użytkownikiem zarządzającym grupą roboczą pracującą w ramach systemu laboratorium wirtualnego na określonym zestawie urządzeń laboratoryjnych lub też obliczeniowych. Grupę taką określamy skrótowo jako laboratorium. Administrator laboratorium nie posiada możliwości zlecania zadań. Jego

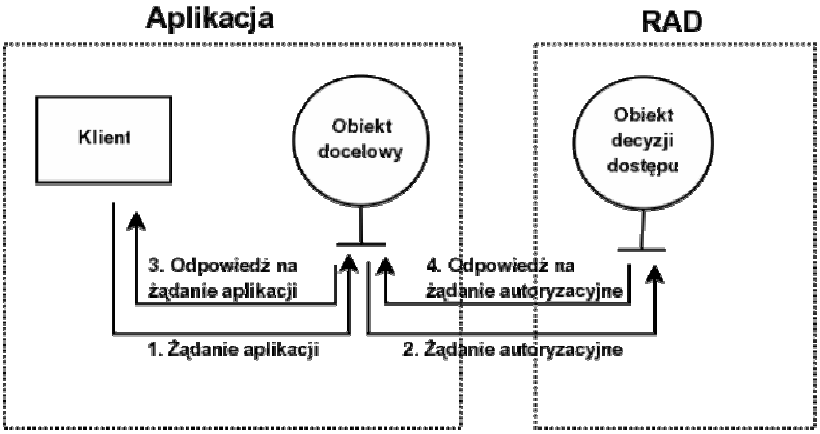
funkcja sprowadza się do czynności czysto administracyjnych . Do celów badawczych musi posiadać osobne konto o profilu użytkownika końcowego.

5.2.2. Administrator systemu laboratorium wirtualnego

Na konto tego administratora loguje się najbardziej uprzywilejowana osoba, czyli administrator systemu laboratorium wirtualnego. Osoba ta ma wpływ na parametry całej instalacji oraz pracę wszystkich laboratoriów (grup roboczych) działających w ramach jednego systemu laboratorium wirtualnego. Administrator systemu ma możliwość dodawania nowych laboratoriów, edytowania parametrów istniejących, blokowania działania takich grup (co pociąga za sobą automatyczną blokada kont wszystkich użytkowników danego laboratorium). Administrator systemu laboratorium wirtualnego zarządza również kontami i profilami administratorów laboratoriów i urządzeń. Ma wgląd w dane rozliczeniowe dotyczące poszczególnych grup roboczych. Może przeglądać pliki logów portali. I wreszcie użytkownik ten zarządza również danymi rzeczywistych instytucji (dodawanie, usuwanie i edycja danych), z którymi związani są poszczególni użytkownicy systemu laboratorium wirtualnego.

5.2.3. Autoryzacja

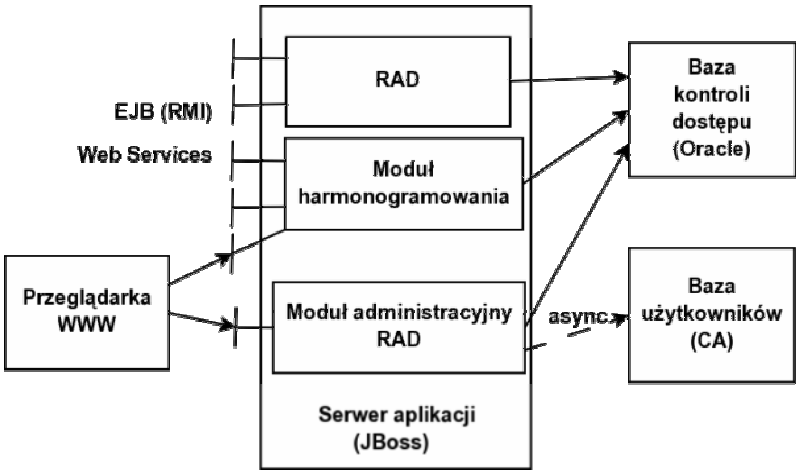
W systemie laboratorium wirtualnego bardzo ważnym aspektem jest opracowanie systemu kontroli uprawnień użytkowników do zasobów systemu. Ma to na celu zapewnić ochronę przyrządów laboratoryjnych przed nieograniczonym dostępem użytkowników, oraz dostarczyć mechanizmów niezbędnych do zarządzania wykorzystaniem przyrządów. Przykładowym rozwiązaniem może być wykorzystanie systemu RAD (ang. *Resource Access Decision*). RAD jest usługą pozwalającą na kontrolę i zarządzanie dostępem do zasobów. Powstał on w ramach prac HDTF (ang. *Healthcare Domain Task Force*) grupy OMG (ang. *Object Management Group*) z nastawieniem na kontrolę dostępu do informacji medycznych.



Rys. 32. Ogólny schemat koncepcji systemu RAD.

Głównymi założeniami modelu RAD są: rozdzielenie logiki autoryzacyjnej (logiki kontroli dostępu) od aplikacji, umożliwienie stosowania różnorodnych polityk kontroli dostępu, maksymalne uproszczenie korzystania z RAD przez aplikacje oraz możliwość zastosowania usługi w systemach informatycznych o dowolnej dziedzinie. RAD ukrywa logikę autoryzacyjną w module zewnętrznym do aplikacji oraz dostarcza standardowego interfejsu umożliwiającego kontrolę dostępu. W celu dokonania autoryzacji, aplikacja żąda decyzji autoryzacyjnej od modułu autoryzacyjnego.

Dla potrzeb systemu laboratorium wirtualnego można wykorzystać implementację RAD.



Rys. 33. Architektura systemu RAD specyficzna dla laboratorium wirtualnego

5.2.4. Opis architektury

Przeglądarka stron WWW służy do przeprowadzania operacji z użyciem interfejsu administracyjnego (zarządzanie użytkownikami, dodawanie zasobów, określanie uprawnień). Interfejs jak i całe zarządzanie realizowane jest poprzez wyodrębniony moduł administracyjny.

Interfejsy RMI oraz Web Services służą do komunikacji z głównym modułem RAD odpowiedzialnym za zarządzanie bazą kontroli dostępu. Za pomocą tych interfejsów realizowane są zapytania autoryzacyjne.

W module RAD zaimplementowane są obiekty funkcji kontroli dostępu (podejmowania decyzji o dostępie do zasobu) nazwane Obiektami Decyzji Dostępu (ADO – ang. *Access Decision Objects*).

Obiekty biorące udział podczas kontroli dostępu to:

- Klient ADO: obiekt żądający dostępu do zasobu (wykonania operacji na zasobie),
- Obiekt *AccessDecision* (Obiekt Decyzji Dostępu): obiekt stanowiący dla klienta ADO interfejs do RAD,
- Usługa dynamicznych atrybutów: usługa udostępniająca dynamiczne atrybuty bezpieczeństwa (dostępne w czasie kontroli dostępu) dla żądanego dostępu,
- Ewaluator polityki: obiekt oceniający czy reprezentowana przez niego polityka zezwala na żądany dostęp.

Zasób chroniony reprezentowany jest w RAD jako struktura *ResourceName* zawierająca identyfikator przestrzeni nazw zasobów (*ResourceNamingAuthority*) oraz listę komponentów wchodzących w skład zasobu (*ResourceNameComponentList*). Komponent reprezentowany jest poprzez parę napisów: nazwa/wartość. Na zasobach chronionych można wykonywać dozwolone dla tych zasobów operacje (np. odczyt, zmiana itp.)

Dla potrzeb systemu laboratorium wirtualnego ogólną funkcjonalność RAD należy rozszerzyć o następujące elementy:

- obsługę protokołu SSL w celu podniesienia poziomu bezpieczeństwa systemu,
- dodatkowe interfejsy Web Services w celu zapewnienia zgodności z modułami laboratorium wirtualnego,
- modyfikacje bazy danych oraz implementacja wykorzystującego ją ewaluatora,
- identyfikację użytkownika na podstawie jego certyfikatu,
- rozszerzenie Bazy kontroli dostępu o czas w którym obowiązuje dane uprawnienie,
- możliwość pobierania listy użytkowników z prawami dostępu dla danego zasobu.

Dodatkowo dla potrzeb Laboratorium Wirtualnego konieczne było zaprojektowanie dodatkowego Modułu harmonogramowania, który pozwalałby na pobieranie czasów dostępności danego zasobu laboratorium (np. urządzenia laboratoryjnego) w przyszłości.

Schemat autoryzacji w systemie laboratorium wirtualnego:

- w portalu, przy uruchamianiu aplikacji SSA sprawdzane jest czy dany użytkownik ma prawo tworzenia oraz zlecania nowych scenariuszy pomiarowych (lub modyfikacji istniejących). W przypadku odpowiedzi pozytywnej wyświetlany jest interfejs SSA i użytkownik może rozpocząć edycję (zakończoną zleceniem scenariusza), natomiast w przeciwnym razie generowany jest odpowiedni komunikat błędu, z informacją jak takie prawo można uzyskać (kontakt do administratora LW).
- Podczas operacji na danych przechowywanych w systemie SZD, RAD odpowiada za określanie praw do ich dostępu. Operacje takie mogą być wykonywane przez wiele modułów. W przypadku każdorazowej próby dostępu (odczyt, zapis, modyfikacja) system autoryzacji określa czy dany użytkownik posiada wystarczające uprawnienia do jej wykonania. Weryfikowane są również prawa dostępu do żądanego pliku danych. Również zewnętrzne systemy, (takie jak GRMS) odwołując się do SZD korzystają z autoryzacji RAD.
- Moduł Szeregowania Globalnego ma za zadanie określić gdzie poszczególne zadania scenariusza pomiarowego trafiają do bezpośredniego wykonania. Dlatego też przed rozpoczęciem przetwarzania, na podstawie informacji z systemu RAD buduje listę urządzeń laboratoryjnych oraz zasobów obliczeniowych, do których użytkownik zlecający zadanie posiada uprawnienia, będących potencjalnymi odbiorcami zadania.
- Do modułu Szeregowania Lokalnego trafiają zadania przeznaczone do uruchomienia na konkretnym urządzeniu. Jego zadaniem jest odpowiednie ich szeregowanie i kierowanie do uruchomienia w określonych momentach. Przy szeregowaniu wykorzystywany jest moduł harmonogramowania RAD, który pozwala określić czy dane urządzenie bądź inny zasób będzie w wybranym momencie dostępne i czy użytkownik będzie miał do tego zasobu prawo dostępu.

5.3. Oprogramowanie

W każdym laboratorium wirtualnym istnieje konieczność wykorzystania specjalizowanego oprogramowaniu służącego do wykonywania symulacji, analizy danych, odkrywania i redukowania oraz wizualizacji. Przykładowym oprogramowaniem wykorzystywanym w laboratoriach wirtualnych jest Virtual Server 2005. Laboratorium wirtualne musi mieć odpowiednie licencje na oprogramowanie, tak jak rzeczywiste laboratoria. Początkowo większość tego oprogramowania była przeznaczona dla maszyn nie podłączonych do sieci. Obecnie bardzo popularny jest proces analizy, w jaki sposób te narzędzia mogą być wkomponowane w heterogeniczną sieć komputerową zawierającą programy, które mogą być skalowane w celu rozwiązywania coraz to nowych problemów.

Ostatnio Microsoft wydał środowiska maszyn wirtualnych Virtual Server 2005 R2. Posiadacze umowy Volume Licence i Software Assurance mogą ten produkt pobrać na stronie licencing.microsoft.com, zaś pozostali mogą czekać na otwarcie innych kanałów dystrybucji już wkrótce lub pobrać 180-dniowe wersje ewaluacyjne. Warto podkreślić ceny, jakie zapowiedział Microsoft na nową wersję wirtualnych serwerów – wersja standard z ograniczeniem do 4 procesorów fizycznych ma kosztować 99$, zaś wersja bez ograniczeń 199$. Virtual Server udostępnia bogate możliwości konfiguracyjne w zakresie składowania danych, sieci i zarządzania. Zamknięty jest przy tym w prostym do użycia pakiecie, instalowanym poprzez przejście 7 prostych kroków kreatora i zarządzanym poprzez konsolę WWW. Virtual Server 2005 R2 umożliwia skuteczną izolację maszyn wirtualnych, podczas gdy mechanizm zarządzania zasobami pozwala zmniejszyć ilość serwerów fizycznych, nawet generujące duże obciążenie. Jako ciekawostkę można podać,

że w wersji R2 formalnie wspierane będą instalacje Linuksa na maszynach wirtualnych, pozwalając konsolidować heterogeniczne środowiska.

[52]Oprogramowanie to umożliwia jednoczesne uruchomienie wielu systemów operacyjnych na jednym komputerze. Obecnie jest ono dostępne jako "release candidate". Producent zapowiedział, że serwer ten pojawi się ostatecznie przed końcem bieżącego roku w dwóch wersjach: Standard i Enterprise. Jedyna różnica pomiędzy nimi to liczba obsługiwanych procesorów. Edycja Standard obsługuje do czterech CPU, Enterprise natomiast do 32.

Aplikacja pracuje na platformie Windows Server 2003. Pozwala na "stworzenie" wirtualnych komputerów, na których instaluje się nowe systemy operacyjne i programy. Obsługiwane są oczywiście Okna, ale nie ma przeszkód, by zainstalować Linuksa. Każdej wirtualnej maszynie można przydzielić aż do 3,6 GB pamięci RAM i przeznaczyć jeden procesor.

Virtual Server służy do tworzenia i testowania aplikacji, przenoszenia starszych programów na nową platformę lub też konsolidacji serwerów. Możliwość migracji przydatna będzie np. wtedy, kiedy korzystamy z aplikacji stworzonych dla Windows NT i mających problemy z działaniem na nowszych platformach. Tę operację w niedalekiej przyszłości uprości tzw. Migration Toolkit, czyli narzędzie służące do tworzenia wirtualnych pecetów z ich fizycznych odpowiedników. Wspomniana wcześniej konsolidacja serwerów polega na przeniesieniu na jedną fizyczną maszynę zasobów wielu komputerów – ułatwia ona zarządzanie i zazwyczaj obniża koszty eksploatacji.

Do uruchomienia Virtual Servera niezbędny jest działający IIS (ang. *Internet Information Server*). Wiąże się to ze sposobem zarządzania komputerami wirtualnymi, które odbywa się z poziomu przeglądarki Internet Explorer z wykorzystaniem specjalnie utworzonej witryny. W pierwszej chwili wydaje się to mało wygodne, ale szybko okazuje się, że rozwiązanie takie ma swoje zalety, np. pozwala wykonywać czynności administracyjne z wielu różnych komputerów bądź też sesji terminalowych. Wirtualne maszyny można bez trudu rekonfigurować, np. dodając im dyski czy karty sieciowe. Niestety, nie obsługują one urządzeń USB, co może być frustrujące i wręcz uniemożliwiające niektóre zastosowania. Bo co można zrobić w takiej sytuacji, kiedy chcemy do logowania wykorzystać na przykład certyfikaty cyfrowe, które są zapisane właśnie na tokenach USB? Virtual Server 2005 to poważny konkurent mających uznaną pozycję na rynku produktów VMWare GSX i ESX Server. Różnice między nimi nie są wielkie. Oprogramowanie Microsoftu jest bardziej skalowalne, gdyż obsługuje do 32 procesorów i umożliwia stworzenie maksymalnie 64 maszyn wirtualnych, ale na razie nie jest dostępne. Konkurencja firmy Microsoft oferuje natomiast obsługę większej liczby systemów operacyjnych i urządzeń USB, jej produkty można zaś obecnie bez trudu kupić.

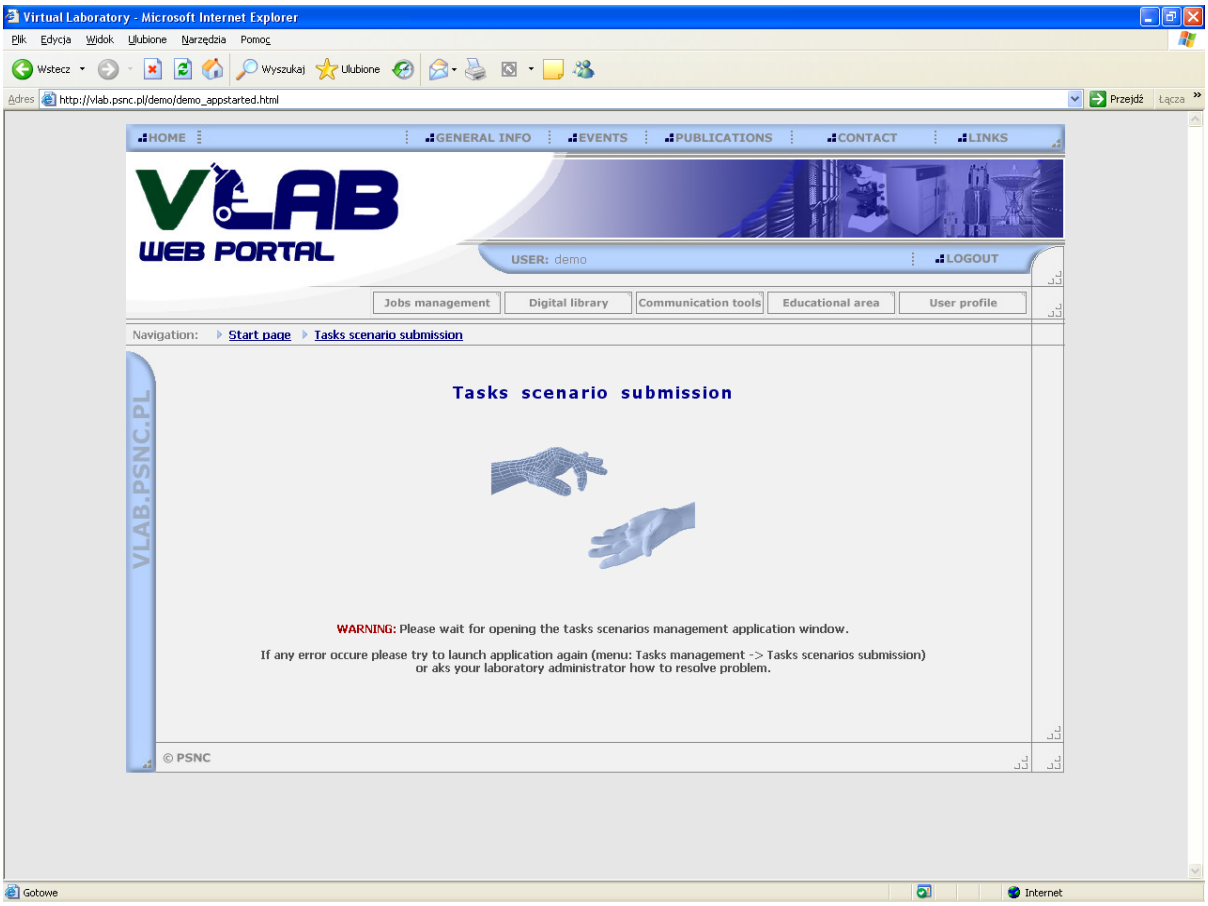
5.4. Krajowe laboratoria wirtualne

5.4.1. VLAB

[16] Laboratorium Wirtualne opracowane w Poznańskim Centrum Superkomputerowo Sieciowym znajduje się w czołówce tego typu systemów na świecie.

System Laboratorium Wirtualnego umożliwia korzystanie z przyrządów laboratoryjnych poprzez Internet. Dzięki takiemu rozwiązaniu również naukowcy znajdujący się poza ośrodkiem badawczym (np. podczas wyjazdów służbowych) oraz uczeni z mniejszych laboratoriów mogą korzystać z bardzo drogich urządzeń pomiarowych takich jak: spektrometr NMR, radioteleskop, mikroskop elektronowy. Właśnie cena tego typu przyrządów powoduje, że są one unikalne i można je spotkać tylko w największych instytutach. Teraz, dzięki Laboratorium Wirtualnemu na ich zakup może złożyć się kilka ośrodków a następnie wspólnie korzystać z ich możliwości. Sama realizacja eksperymentu zazwyczaj nie wystarcza aby uzyskać oczekiwany rezultat. Dane uzyskane podczas badania należy poddać obróbce z wykorzystaniem specjalistycznego oprogramowania. Laboratorium Wirtualne pozwala na połączenie urządzeń laboratoryjnych z serwerami obliczeniowymi i w ten sposób na stworzenie pełnego

scenariusza pomiarowego umożliwiającego przeprowadzenie całego badania w optymalny pod względem czasu i wykorzystanych zasobów sposób.



Rys. 34. Portal systemu VLAB.

Uzyskane w ten sposób dane mogą zostać umieszczone w Naukowej Bibliotece Cyfrowej (NBC) w zorganizowany sposób. NBC oprócz przechowywania danych pomiarowych umożliwia także ich przeszukiwanie, aktualizację, dodawanie referencji do innych tego typu odkryć a przede wszystkim na współdzielenie wyników między badaczami.

Oprócz realizacji eksperymentu i jego obróbki system umożliwia wspólną pracę kilku naukowców z różnych miejsc na świecie nad tym samym problemem. Jest to możliwe dzięki narzędziom do pracy grupowej takim jak: wideo-konferencje, grupy dyskusyjne, chat.

Obecnie do sieci światłowodowej Wirtualnego Laboratorium podłączone są cztery urządzenia: dwa spektrometry oraz dwa radioteleskopy (jeden 32-metrowy w Piwnicach pod Toruniem, drugi w Mexico City). Astronom może na odległość poruszać teleskopem, by zobaczyć interesujący go fragment nieba! W przyszłości podłączone zostaną również mikroskopy elektronowe, a także tomografy, co ułatwiłoby lekarzom stawianie diagnoz. Do miejscowości, gdzie jest tomograf, mógłby jechać tylko pacjent. Znający go dobrze lekarz z lokalnej przychodni badałby go na odległość.

[53]Cena takich przyrządów jak tomograf czy mikroskop elektronowy (każdy z nich kosztuje kilka milionów złotych) powoduje, że są one w Polsce nieliczne i znajdują się tylko w największych placówkach naukowych i medycznych. Dzięki Wirtualnemu Laboratorium na zakup takiego urządzenia może się złożyć kilka instytucji, a potem wspólnie z niego korzystać. Do sieci można podłączyć dowolną liczbę urządzeń – pod tym względem system nie ma żadnych ograniczeń.

Wirtualne Laboratorium jest obsługiwane w języku angielskim. Zaprojektowano je tak, aby każdy, kto chce mieć do niego dostęp, nie musiał instalować na swoim komputerze

dodatkowych narzędzi. Program gwarantuje także bezpieczeństwo – jego użytkownik musi mieć certyfikat, jest wcześniej dokładnie sprawdzany. To są bardzo drogie urządzenia, dlatego do systemu nie może wejść osoba z ulicy. Program to unikat w skali światowej. Jest entuzjastycznie przyjmowany i doceniany na każdej konferencji, na której jest prezentowany. Nad takim oprogramowaniem pracują także naukowcy za granicą.

VLAB jest również członkiem projektu zatytułowanego „Obliczenia wielkiej skali i wizualizacja do zastosowań w wirtualnym laboratorium z użyciem klastra SGI". Projekt ten jest finansowany przez potężny koncern międzynarodowy Silicon Graphics. Ze strony polskiej współpracują ATM, Akademickie Centrum Komputerowe Cyfronet (AGH), Poznańskie Centrum Superkomputerowo-Sieciowe, Wrocławskie Centrum Komputerowo-Sieciowe, TASK oraz Centrum Komputerowe Politechniki Łódzkiej.

Kolejne polskie laboratoria wirtualne udostępniają jedynie applety Javy ilustrujące konkretne zjawiska. Według naszej wiedzy jedynym polskim laboratorium z prawdziwego zdarzenia i jednym z nielicznych na świecie jest VLAB wymienione powyżej. Jak wiadomo koszty budowy takiego laboratorium sięgają kilku a nawet kilkunastu milionów złotych.

5.4.2. Wirtualne Laboratorium Geomatyki

http://netgis.geo.uw.edu.pl/vlg/index.shtml

Wirtualne Laboratorium Geomatyki zostało stworzone przez Wydział Geologii Uniwersytetu Warszawskiego, Interdyscyplinarne Centrum Modelowania Uniwersytetu Warszawskiego oraz Polskie Towarzystwo Informacji Przestrzennej. Podstawę techniczną stanowią cztery serwery: NetGis, InterGis, MapServer i TestBed (GIS – ang. *Geographical Information System* – system informacji geograficznej).

5.4.2.1. NetGIS

Na tej stronie Można tu znaleźć strony dotyczące różnych aspektów teoretycznych i praktycznych geomatyki i zastosowania GIS, a także innych systemów informatycznych wykorzystywanych w przetwarzaniu informacji geoprzestrzennej.

Dostępne zasoby:

- Mapa świata (GoogleMap – opis w dalszej części pracy),
- Nazwy geograficzne na mapie w systemie Unicode,
- Model Polski stworzony przy pomocy programu GRASS (oprogramowanie GNU przeznaczone do tworzenia, przetwarzania i analizy danych geoprzestrzennych) – polecamy pobranie map w formacie TIFF z metadanymi XML . Są to dość duże pliki, jednakże Polska prezentuje się bardzo efektownie,
- GRASS wraz ze szczegółową dokumentacją,
- ASPAR (dane dotyczące systemów hydrologicznych),
- Kilka ciekawych programów z dziedziny geomatyki.

5.4.2.2. TestBed

Serwer ten jest prowadzony przez Komisję Technologii Interoperacyjnych PTIP. Jego zadaniem jest testowanie rozwiązań metodycznych i technologicznych z zakresu geomatyki, a w tym w szczególności: zastosowania języka XML do geoinformacji, do metadanych w zakresie geoinformacji, a także języka GML (ang. *Geography Markup Language*) będącego geomatyczną implementacją języka XML.

5.4.2.3. InterGis

Jest to eksperymentalny serwer WWW przeznaczony interdyscyplinarnym aspektom geoinformacji. Geoinformacja w różnych dziedzinach ma wiele zagadnień wspólnych i w takim przypadku jej modele pojęciowe mogą być uogólnione – można wyznaczyć pewien "wspólny mianownik" mający zastosowanie we wszystkich dziedzinach. W takim ujęciu jest to zagadnienie interdyscyplinarne. Jednak w każdej dziedzinie w zakresie aspektu geomatycznego mamy do czynienia z problemami specyficznymi, występującymi tylko w

tej dziedzinie lub w wąskiej ich grupie – rozwiązywanie tych problemów jest w takim przypadku zadaniem poszczególnych dziedzin.

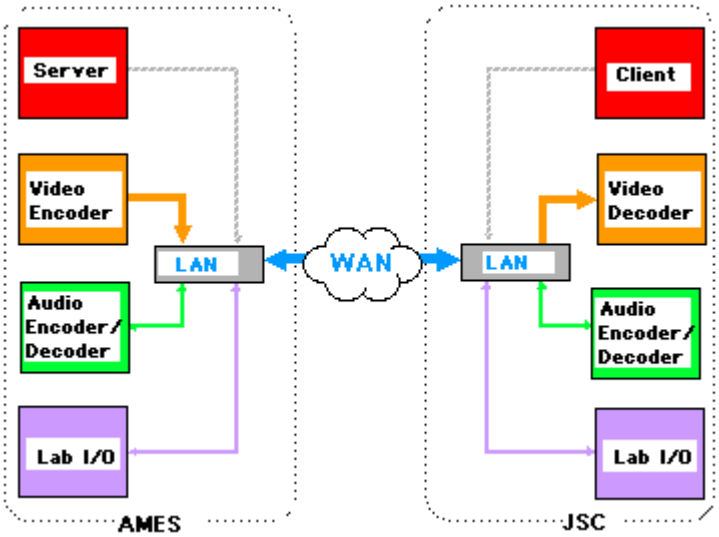
5.5. Zagraniczne laboratoria wirtualne

5.5.1. NASA SimLabs

http://www.simlabs.arc.nasa.gov/vms/virtual_lab.html
(dostęp jedynie z sieci wewnętrznej NASA)

Laboratorium wirtualne NASA, zostało pierwotnie stworzone jako symulator lotu. Wraz z rozwojem systemu powstało wiele aplikacji tj. wirtualne centrum kontroli lotu, symulator aerodynamiczny, tester lotów. Aplikacje te umożliwiają zdalne programowanie lotów oraz wirtualny przelot. W 1997 roku Centrum Kosmiczne Johnsona podłączone do wyrzutni promów kosmicznych wykonało pierwszą symulację lotu kosmicznego.

System posiada architekturę klient/serwer. Dla aplikacji wymagających wysokiej jakości obrazu, zastosowano parę kodeków MPEG-2. Sieć NASA jest podpięta poprzez system AMES do sieci WAN DarwinNet, wspomaganych przez projekt NREN. NASA informuje również, że sieć VLAB jest wykorzystywana do demonstracji osiągnięć pracownikom oraz firmom z nią współpracujących.



Rys. 35. Schemat blokowy struktury laboratorium.

5.5.2. NASA Virtual Labs

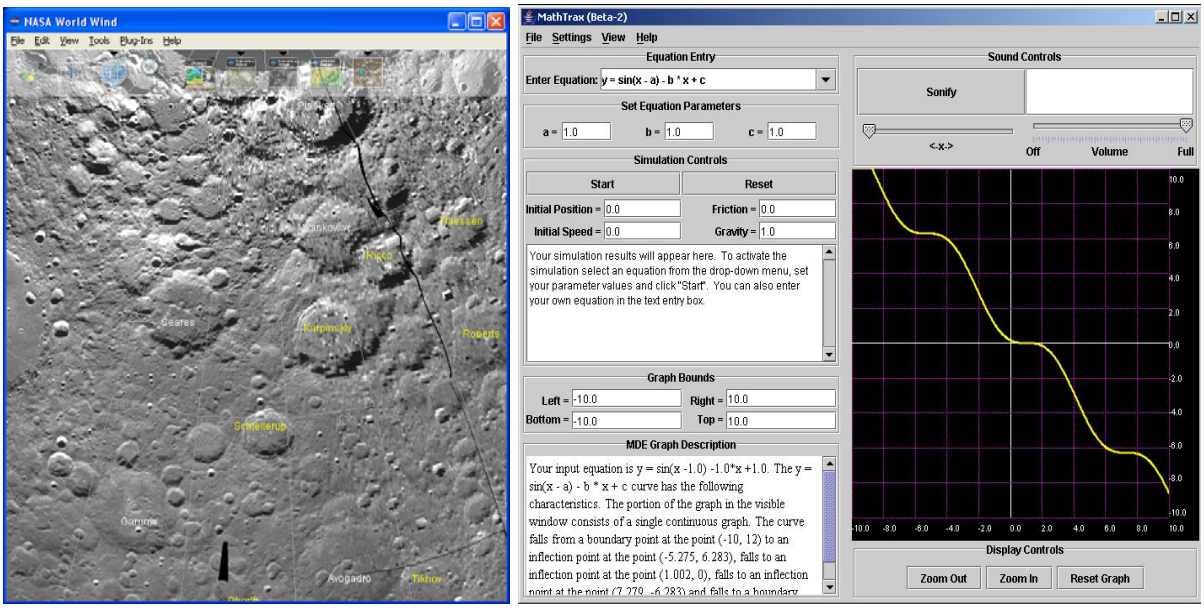
http://learn.arc.nasa.gov/vlab/index.html

NLT (ang. *NASA Learning Technologies*) jest oddziałem mającym na celu publikację osiągnięć NASA. Dostarcza dane w przejrzysty i łatwy sposób.

Wirtualne Laboratorium NASA emuluje dokładnie sposób działania mikroskopu elektronowego SEM PHILLIPS XL-30 ESEM FEG oraz pozwala użytkownikowi obejrzeć kilka przykładów działania (powiększenie do 3600x, focus, jasność, kontrast). W pakiecie zawarty jest również film pokazujący mikroskop podczas pracy. Elementem przykładowym jest system mikro-elektromechaniczny (iMEMS) „Kidney Stone" wyprodukowany przez firmę Analog Devices. Drugim urządzeniem jest mikroskop fluorescencyjny Zeiss Axionvert 100 natomiast układem badanym jest Texas Instuments DLP Chip.

Rys. 36. NASA Virtual Labs.

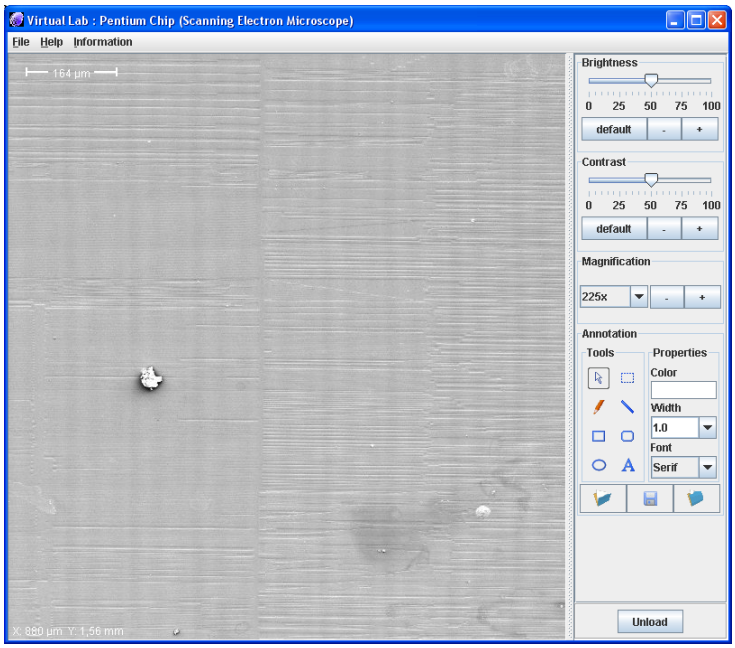
Ponadto ze strony można pobrać również bardzo interesującą mapę ziemi i księżyca wg NASA oraz ciekawe narzędzie matematyczne MathTrax (rysunek 37).



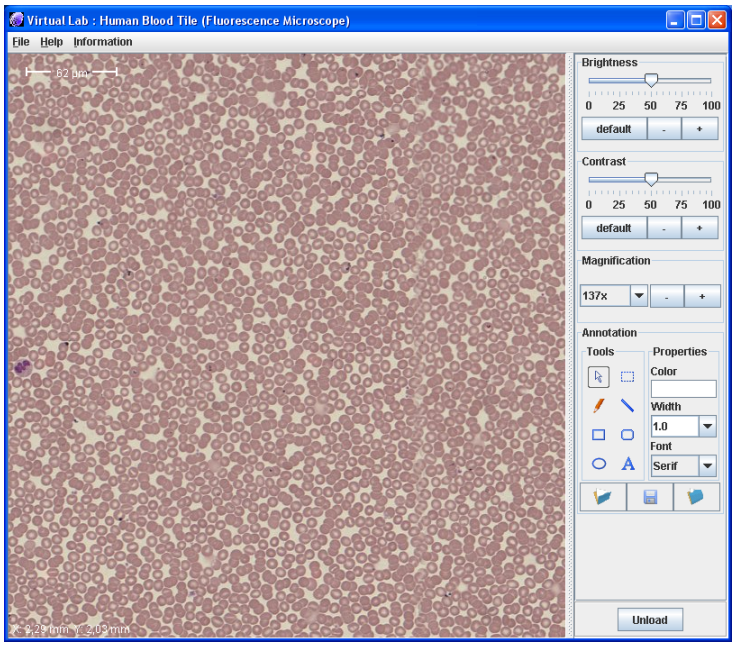
Rys. 37. Przykłady dostępnych informacji w NASA Virtual Labs.

Ze strony http://virtual.itg.uiuc.edu/data/ można pobrać dużą porcję przygotowanych danych dla programu Virtual Lab. Są to pliki liczone w dziesiątkach megabajtów, jednakże autorzy referatu gorąco polecają zapoznanie się z tym materiałem:

- Mikroskop elektronowy:
  - kryształ koloidalny,
  - DLP Chip,
  - zabezpieczenie antypożarowe,
  - mucha,
  - żuk,
  - schemat bramki NAND,
  - struktura potłuczonej filiżanki,
  - uszkodzone jądro CPU Intel Pentium (rysunek 38).
- Mikroskop fluorescencyjny:
  - płytki krwi człowieka (rysunek 39),
  - sensor CCD stosowany w kamerach,
  - alga,
  - histologia aorty człowieka,
  - chip regulujący napięcie,
  - histologia żołądka psa (ang. *esophagus*),
  - histologia jelita cienkiego psa (ang. *jejunum*),
  - histologia śledziony psa (ang. *spleen*),
  - przełyk małpy (ang. *larynx*),
  - histologia nerki człowieka (ang. *kidney*),
  - histologia serca psa,
  - histologia wątroby psa (ang *liver*).



Rys. 38. Obraz uszkodzonego procesora Pentium z mikroskopu elektronowego.



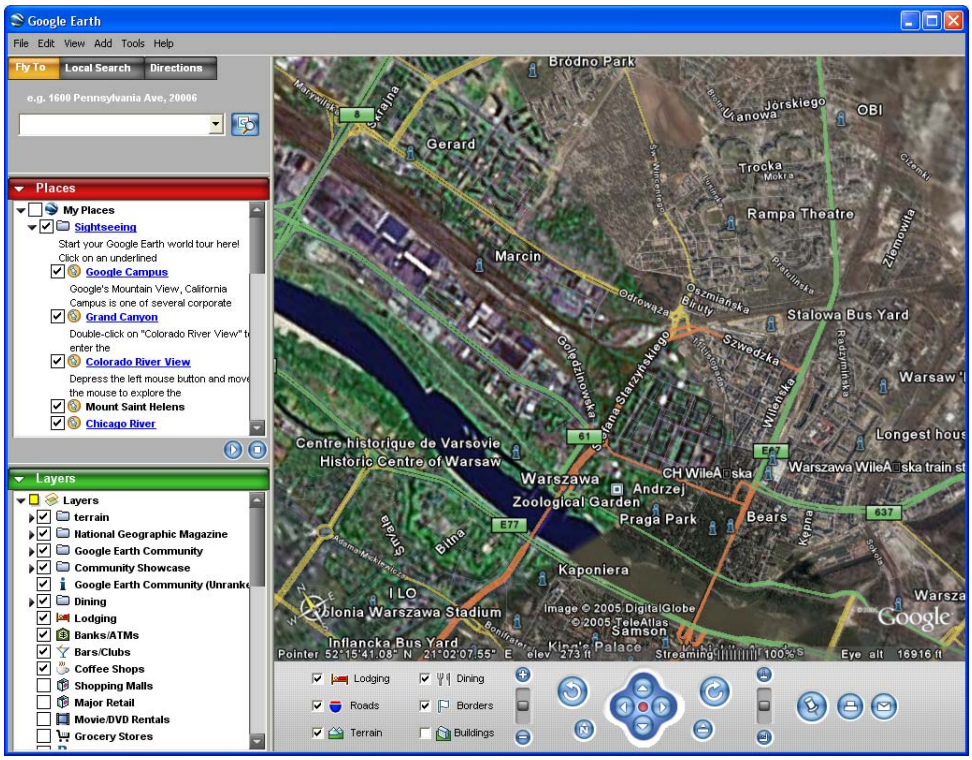
Rys. 39. Obraz płytek krwi człowieka z mikroskopu fluorescencyjnego.

5.5.3. Google Earth

(http://earth.google.com)

[Wikipedia] Google Earth to program do podglądu zdjęć satelitarnych na trójwymiarowym modelu Ziemi, oparty na technologii Keyhole (wykupionej przez Google w październiku 2004 roku). W ramach wersji podstawowej program jest bezpłatny. Wykorzystywane przez niego zdjęcia satelitarne dostępne są również w ramach innej bezpłatnej usługi Google Local.

Za pomocą Google Local można odnaleźć różne instytucje, sklepy oraz inne miejsca publiczne znajdujące się w obsługiwanych krajach (obecnie: Stany Zjednoczone, Kanada, Wielka Brytania) z dokładnością do jednej, pięciu, piętnastu lub czterdziestu pięciu mil. Istnieje również możliwość podejrzenia obrazu z satelity.



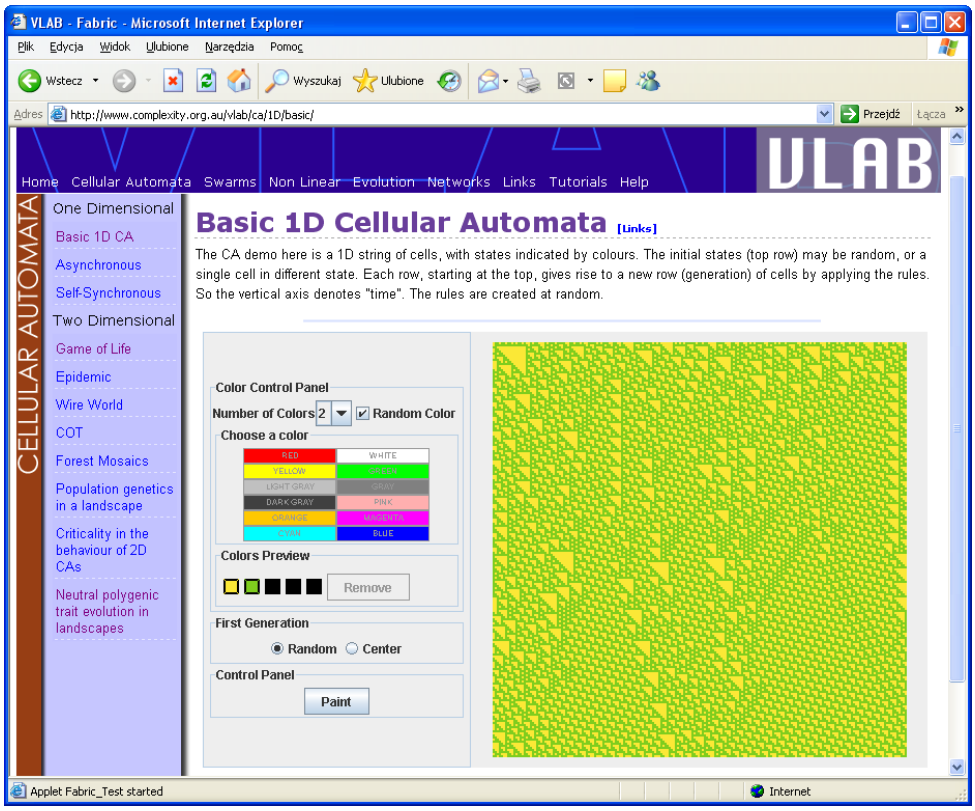
Rys. 40. Zdjęcie dostępne w ramach usługi Google Earth.

5.5.4. Monash University's Artificial Life Virtual Lab (VLAB)

http://www.complexity.org.au/vlab/

Jest to laboratorium stworzone przez Uniwerystet Monasha. Zajmuje się głównie zagadnieniami z dziedziny sztucznego życia. Na stronie znajduje się kilka przykładów ilustrujących algorytmy Alife (ang. *Artifical Life*), przedstawionych w płaszczyźnie jedno i dwuwymiarowej:

- diagram życia komórki,
- życie symbiotyczne kilkunastu organizmów,
- ewolucja i procesy uczenia się,
- sieci.



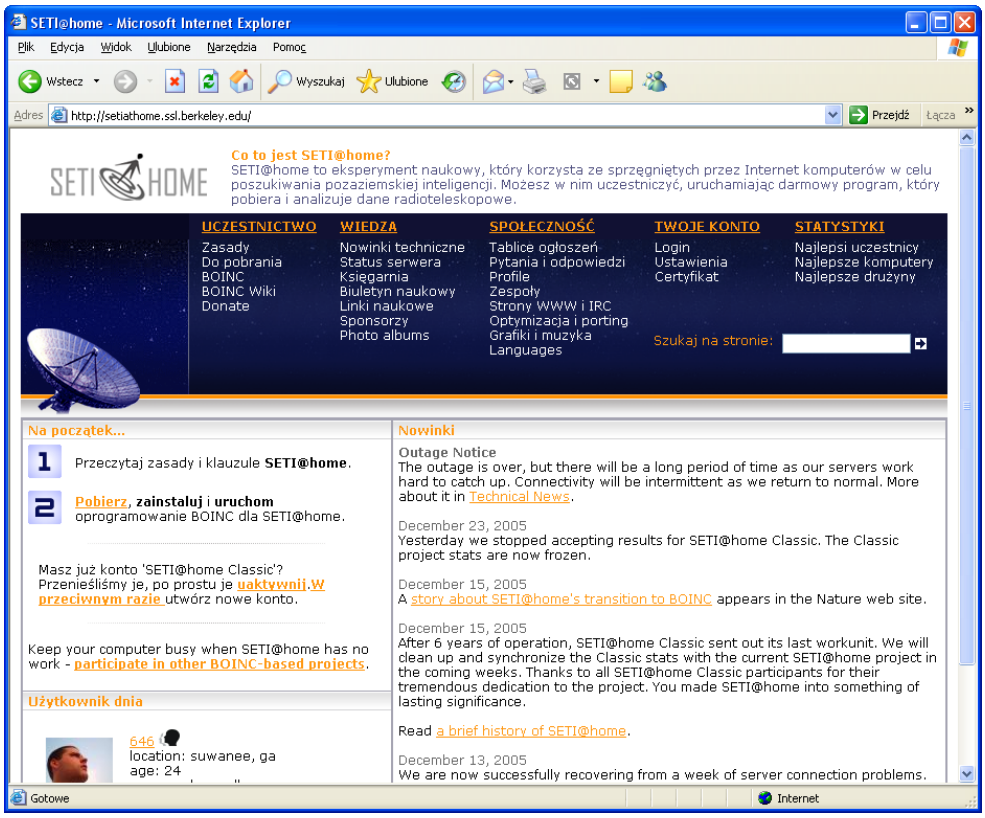
Rys. 41. Przykład ilustrujący algorytm Alife.

5.5.5. Seti@Home

http://setiathome.ssl.berkeley.edu/

SETI@home to eksperyment naukowy, który korzysta ze sprzęgniętych przez Internet komputerów w celu poszukiwania pozaziemskiej inteligencji. Jego celem jest poszukiwanie cywilizacji pozaziemskich przez analizę emisji radiowej kosmosu rejestrowanej przez największe radioteleskopy na Ziemi. Główny system obliczeniowy znajduje się na Uniwersytecie Berkleya w Kalifornii.

Projekt wciąż się rozwija. Sponsorami są również duże międzynarodowe koncerny (m. in. Sun, Nvidia, Quantum, Fuji, HP, O'Reilly). W każdej chwili można się przyłączyć do poszukiwań, wystarczy jedynie zainstalować na swoim komputerze program BOINC, który wykorzystuje określone zasoby komputera do obliczeń.



Rys. 42. Strona informacyjna programu Seti@Home.

5.5.5.1. Podstawowe założenia SETI

Celem programu SETI@home jest wykorzystanie zwykłych domowych komputerów połączonych siecią Internet, dysponujących sumarycznie wręcz nieograniczoną mocą obliczeniową dla potrzeb analizy sygnałów.

Pierwszym etapem prób ustanowienia łączności międzygwiezdnej stał się projekt SETI. Polega on na poszukiwaniu gwiazd, z których emitowane są sygnały wskazujące na istnienie tam inteligentnych form życia pozaziemskiego. SETI nie jest jednak trywialnym zadaniem. Galaktyka Drogi Mlecznej, w której znajduje się Słońce ma średnicę ok. 100 tys. lat świetlnych i zawiera ok. 200 mld gwiazd. Szukanie sygnałów pozaziemskich cywilizacji w całej galaktyce bez odpowiedniej selekcji mogłoby zająć setki lat. Jednakże, trzy proste założenia pomagają zmniejszyć rozmiary tego zadania:

- Większość form życia w naszej jest oparta na chemii węgla, podobnie jak to jest na Ziemi. Koncepcja ta jest dość dobrze uzasadniona naukowo, gdyż tylko węgiel z energetycznego punktu widzenia jest w stanie samorzutnie tworzyć wystarczająco złożone związki chemiczne.
- Niezbędna jest obecność wody w postaci ciekłej. Założenie to wynika z faktu, że tylko w środowisku wodnym możliwy jest złożony metabolizm niezbędny do utrzymania przemiany materii koniecznej dla istot ożywionych.
- Poszukiwania powinny być skoncentrowane na gwiazdach o podobnych parametrach do Słońca należących do ciągu głównego. Bardzo duże gwiazdy mają za krótkie czasy trwania, aby mogły wytworzyć trwałe systemy planetarne, z kolei bardzo małe generują za mało światła i ciepła do powstania życia opartego na chemii węgla.

Około 10% gwiazd w Drodze Mlecznej ma parametry zbliżone do słońca i tylko ok. 1000 z nich znajduje się w odległości mniejszej niż 100 lat świetlnych od Ziemi. Te gwiazdy są głównymi celami programu SETI. Jednakże, istnieje zawsze ryzyko, że powyższe założenia są częściowo błędne i dlatego program SETI obejmuje też wyrywkowe badanie innych obszarów kosmosu.

5.5.5.2. Początki SETI

Współczesny program SETI został zapoczątkowany 19 września 1959 roku. W dwustronicowym artykule zatytułowanym „Searching for Interstellar Communications" opublikowanym w Nature przez dwóch młodych fizyków Philipa Morrisona i Giuseppe Cocconi'ego została opisana możliwość komunikowania się w przestrzeni kosmicznej za pomocą fal radiowych o długości 21 cm (1420 MHz). Wybór akurat tej częstotliwości był podyktowany wieloma racjonalnymi przesłankami:

- Sygnały o zbyt małych częstotliwościach są zbyt silnie tłumione, stąd wydaje się, że komunikacja na długich dystansach jest możliwa tylko w zakresie fal ultrakrótkich. W grę wchodzi więc zakres od 100 do 10000 MHz. Przy częstotliwości poniżej 1400 MHz, sygnały radiowe są silne zakłócane przez emisję generowaną przez wolne elektrony przemieszczające się przez silne pola magnetyczne występujące wokół gwiazd. Z kolei sygnały powyżej 1600 MHz są silnie zagłuszane przez emisje pochodzące od przemian jądrowych zachodzących wewnątrz gwiazd. W grę wchodzi więc tylko zakres 1400 – 1600 Mhz.
- Ze względów energetycznych łatwiej jest generować fale radiowe, o możliwie jak najniższej częstotliwości, a tutaj występuje bardzo charakterystyczny i wąski sygnał emisji wodoru – 1420 Mhz.

Młody radioastronom Frank Drake doszedł do tego samego wniosku i dzięki wsparciu swojego szefa *Otto Struve* (jednego z najwybitniejszych astronomów ubiegłego wieku) mógł do nasłuchów wykorzystać 26-metrowy radioteleskop Narodowego Obserwatorium Radioastronomicznego w Green Bank. Urządzenie skierował ku jednym z bliższych Ziemi, a podobnym do Słońca gwiazdom – *52 tau Ceti* i *18 epsilon Eridani*, odległych od słońca odpowiednio 11,9 i 10,5 lat świetlnych i 8 kwietnia 1960 roku rozpoczął nasłuch.

W ten sposób narodził się jeden z budzących większe kontrowersje programów badawczych a mianowicie program SETI (ang. *Search for ExtraTerrestrial Inteligence* – poszukiwanie pozaziemskich cywilizacji*)*.

W początkach lat 80-tych fizyk z Harvard University, Paul Horowitz zaprojektował analizator widma radiowego specjalnie dla celów projektu SETI. Tradycyjne analizatory stosowane dotąd posiadały bowiem filtry analogowe, które mogły "wycinać" interesujące sygnały i były one w stanie analizować tylko bardzo wąskie zakresy fal radiowych. Horowitz zaadaptował dla celów SETI układy cyfrowej analizy sygnałów stosowane dotąd przez wojsko. W 1981 r. wybudowano pierwszy tego rodzaju analizator o nazwie "Suitcase SETI" (walizkowe SETI), który był w stanie analizować na raz 131 tys. wąskich kanałów radiowych. W 1983 r. "Suitcase SETI" został przyłączony do 25-metrowego radioteleskopu należącego do Harvard University i Smithsonian Institute. Ten projekt został nazwany "Sentinel" i był kontynuowany do 1985 r.

Okazało się jednak, że 131 tys. kanałów to wciąż za mało aby szczegółowo zbadać szum radiowy z kosmosu w rozsądnym przedziale czasu i projekt "Sentinel" został zastąpiony projektem "META" (ang. *Megachannel Extra-Terrestrial Array*). Analizator zbudowany w ramach tego projektu posiadał już zdolność analizy 8 mln kanałów na raz. Szefem projektu "META" był Horowitz, zaś jego sponsorami było Planetary Society oraz Steven Spielberg. Kontynuacja tego projektu o nazwie "META II" rozpoczęła się w 1990 r. z użyciem radioteleskopu położonego w Argentynie – aby przebadać część kosmosu widoczną z południowej półkuli.

Równolegle, w 1985 r. Ohio State University uruchomiło swój własny program SETI, nazwany „Big Ear" (Wielkie Ucho), który został również wsparty przez fundusze z Planetary Society. Rok później władze Berkeley University zdecydowały się uruchomić drugi, równoległy projekt SERENDIP.

Projekt SETI@home wykorzystuje dane pochodzące z innej inicjatywy działającej w ramach programu SETI – z projektu SERENDIP IV (ang. *Search for Extraterrestial Radio Emissions from Nearby Developed Intelligent Populations*). Projekt ten został opracowany na Kalifornijskim Uniwersytecie w Berkeley w celu tzw. pasożytniczego zbierania danych dla programu SETI. Projekt SERENDIP IV korzysta z największego w tej chwili radioteleskopu na świecie – Arecibo znajdującego się na wyspie Puerto Rico. Średnica jego anteny wynosi 305 m. Dane pobierane są z dedykowanego odbiornika radioteleskopu, dzięki czemu projekt SETI może być realizowany w czasie normalnych prac badawczych prowadzonych na tym największym na świecie urządzeniu.

Rys. 43. Radioteleskop Arecibo widziany z wysokości 400 km. Z tego urządzenia pochodzą dane przetwarzane w programie SETI@home

Współczesne projekty radiowego SETI analizuje dane cyfrowo. Ta analiza generalnie wymaga trzech faz:

- obliczenia zmiennej czasowo mocy spektrum danych,
- znalezienia sygnałów kandydackich używając wzoru rozpoznawania mocy widma,
- wyeliminowania z sygnałów kandydackich, tych które są prawdopodobnie pochodzenia naturalnego lub sztucznego.

Większa moc obliczeniowa umożliwia poszukiwania obejmujące większy zakres częstotliwości z większą czułością. Dlatego radiowe SETI ma nienasycony apetyt na moc obliczeniową.

W poprzednich projektach radiowego SETI używano specjalnie stworzonych superkomputerów, umiejscowionych przy radioteleskopie, które wykonywały większość analizy danych. W 1995 r. David Gedye zaproponował stworzenie projektu radiowego SETI używającego do analizy danych wirtualnego superkomputera złożonych z ogromnej liczby komputerów podłączonych do Internetu i on zorganizował projekt SETI@home do zbadania tego pomysłu. SETI@home nie znalazło znaków życia pozaziemskiego. Ale, razem z powiązaniem przetwarzania rozproszonego i projektem przechowywania danych, osiągnęło sukces w zdolności wykorzystania publicznych zasobów obliczeniowych (tak nazwanych, ponieważ zasoby obliczeniowe są dostarczane przez ogół społeczeństwa).

Publiczne zasoby obliczeniowe nie są ani żadnym panaceum, ani „darmową pożywką". Dla wielu zadań, ogromna moc obliczeniowa oznacza ogromną przepustowość sieci, a przepustowość jest z reguły droga lub ograniczona. Ten czynnik ograniczył zakres częstotliwości poszukiwań przez SETI@home, ponieważ zwiększenie zakresu oznaczało większą liczbę bitów na sekundę. Porównując to do innych projektów radiowych SETI, SETI@home obejmuje węższe pasmo częstotliwości, ale dokładniej je analizuje.

5.5.5.3. Projekt SETI@home

Pierwszym wyzwaniem dla SETI@home było znalezienie dobrego radioteleskopu. Idealnym wyborem był radioteleskop Arecibo w Puerto Rico, największy i najczulszy radioteleskop na świecie. Arecibo jest wykorzystywany dla różnych astronomicznych i atmosferycznych badań, i nie mogliśmy otrzymać go na wyłączność w celu długoterminowego użytkowania. Jednakże w 1997 roku projekt SERENDIP Uniwersytetu Berkeley rozwinął technikę pasożytniczego zbierania danych przez drugą antenę w Arecibo. Ponieważ antena główna śledziła ustalony punkt na niebie (pod kontrolą innych badaczy), druga antena przebywa odcinek łuku, który ostatecznie obejmuje cały zakres nieba widoczny przez teleskop. Takie źródło danych może zostać wykorzystane dla przeglądania nieba, które obejmuje miliardy gwiazd.

Na potrzeby SETI@home została przeznaczona część danych źródłowych projektu SERENDIP. W odróżnieniu od SERENDIP dane rozprowadzane są przez Internet. W tamtym czasie Arecibo było połączone z Internetem za pomocą modemu 56Kbps, więc zdecydowano się na zapisywanie danych na usuwalnych taśmach, a potem wysyłanie ich pocztą z Arecibo do laboratorium na Uniwersytecie w Berkeley i rozprowadzanie ich przez tamtejszy serwer.

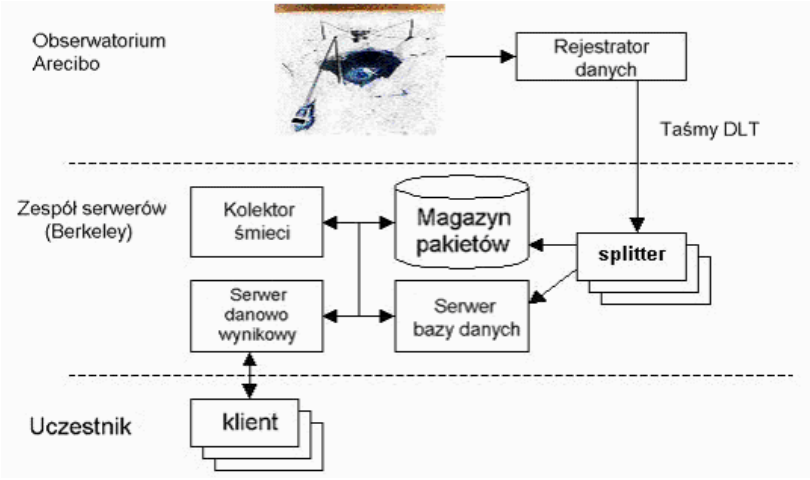
Zdecydowano się na zapisywanie danych z szybkością 5Mbps. To tempo było na tyle niskie, że zapis pojedynczej taśmy był wykonalny w 16 godzin i był możliwy do osiągnięcia przy połączeniu laboratorium z Internetem o szybkości 100Mbps. Z naukowego punktu widzenia było to dostatecznie dużo. Z 1-bitowym próbkowaniem zespolonym dawało to taką szybkość, by uzyskać pasmo o szerokości 2,5MHz – wystarczające do uchwycenia przesunięcia Dopplera dla względnej szybkości powyżej 260 km/s lub o wskaźniku obrotu galaktyk (sygnały radiowe z przesunięciem Dopplera są proporcjonalnymi do prędkości nadajnika względem odbiornika). Tak jak wiele radiowych projektów, SETI skoncentrowano pasmo na linii emisyjnej wodoru (1,42GHz), w środku zakresu częstotliwości, gdzie sztuczne transmisje są zakazane przez traktat międzynarodowy.

Model obliczeniowy SETI@home jest prosty. Sygnał dzielony jest na jednostki robocze o stałej wielkości (ang. *work unit*), które są rozprowadzane, przez Internet, do programów klienckich uruchomionych na licznych komputerach. Program kliencki oblicza wynik (grupa sygnałów kandydackich), zwraca to do serwera i pobiera inną jednostkę roboczą. Nie ma tutaj komunikacji pomiędzy klientami.

W SETI@home występuje nadmiarowość obliczeń: każdy pakiet danych jest przetwarzany wielokrotnie. To pozwala wykryć i odrzucić wyniki z wadliwych procesorów lub pochodzące od złośliwych użytkowników. Nadmiarowość na poziomie od 2 do 3 jest wystarczająca do osiągnięcia celu. Pakiety wytwarzane są w ograniczonym tempie i nigdy nie klient pytający na prace nie jest odsyłany, tak więc poziom nadmiarowości obliczeń wzrasta wraz z liczbą klientów i ich średnią prędkością. Te ilości wzrastają bardzo przez żywotność projektu. Utrzymywany jest poziom nadmiarowości na środku skali zapotrzebowania nakazując klientowi ponowne zrobienie większej liczby obliczeń na próbkę.

Zadaniem tworzenia i dystrybucji pakietów zajmuje się kompleksowy serwer w laboratorium (rysunek 44) Powody dla centralizacji funkcji serwera były w dużej mierze pragmatyczne, dla przykładu zminimalizowaliśmy manipulowanie taśmą.

Jednostki robocze tworzy się poprzez podzielenie 2,5MHz sygnału na 256 przedziałów, każdy szerokości około 10kHz. Następnie każdy taki przedział jest dzielony na 107-sekundowe segmenty, zachodzące na siebie na 20 sekund. To zapewnia, że szukane przez nas sygnały zawierać się będzie zupełnie przynajmniej w jednej próbce. Próbki mają po 350KB – dane wystarczające do utrzymania typowego komputera w zajętości przez około cały dzień, ale wystarczająco małe do ściągnięcia przez powolny modem w kilka minut.



Rys. 44. Dystrybucja danych

Używana jest relacyjna bazy danych do przechowywania informacji o kasetach, próbkach, wynikach, użytkownikach i innych aspektach projektu. Wykorzystywany jest wielowątkowy serwer danowo-wynikowy do rozprowadzania pakietów do użytkowników. Bazuje on na protokole HTTP, więc nawet klient umieszczony za firewallem może się z nim połączyć. Pakiety są wysyłane według zasady: najmniej niedawno wysłanych (ang. *least-recently-sent*).

Program czyszczenia zasobów (ang. *collector garbage*) usuwa pakiety z dysku, oczyszczając dalej znaczniki stanu w ich rekordach bazy danych. Używane są dwie zasady:

- usuwamy pakiety, dla których otrzymaliśmy N wyników, gdzie N jest naszym docelowym poziomem powtarzalności. Jeśli magazyn próbek wypełni się, wytwarzanie pakietów jest blokowana i możliwości systemu spadają,
- usuwamy pakiety, które zostały wysłane M razy, gdzie M jest odrobinę większe od N. To eliminuje wąskie gardło, ale powoduje, że kilka pakietów nigdy nie otrzyma wyniku.

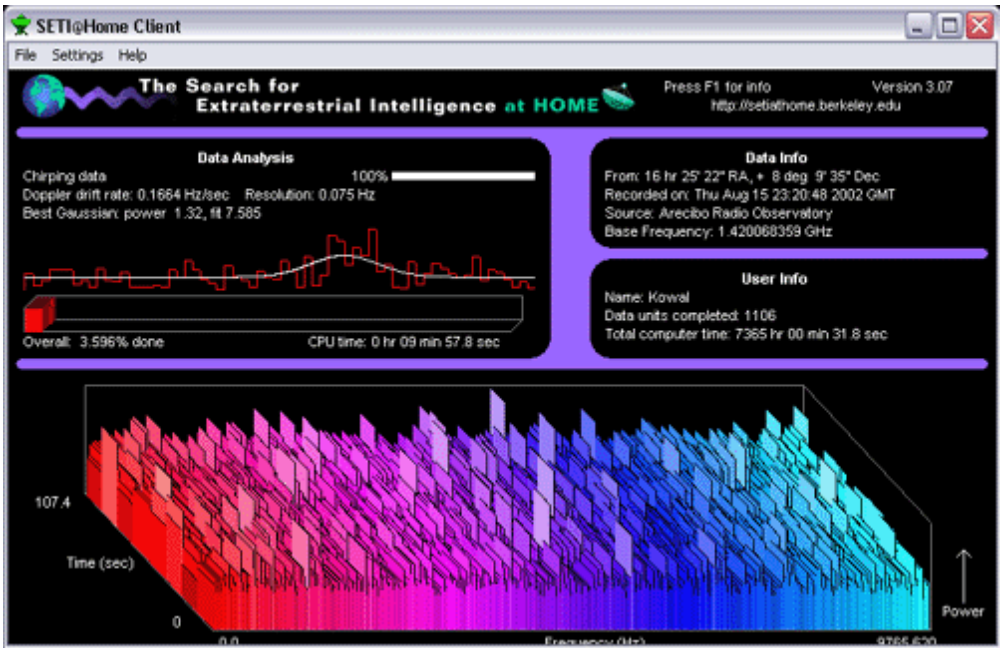
Utrzymywanie systemu serwerowego w ruchu jest najtrudniejszą i kosztowną częścią projektu SETI@home. Źródła uszkodzenia, czy to sprzętu czy oprogramowania są nieograniczone. Zabiegamy o architekturę, która zminimalizuje współzależność pomiędzy podsystemami. Dla przykładu: serwer danowo-wynikowy może być uruchomiony w trybie, gdzie zamiast używania bazy danych do wyliczenia pakietu do wysłania, pobiera tą informację z plików dyskowych. To pozwala rozprowadzać dane, gdy baza danych jest wyłączona.

Program klienta wielokrotnie pobiera pakiety z serwera danowo-wynikowego, analizuje je i zwraca wynik (listę kandydujących sygnałów) do serwera. To wymaga połączenia z Internetem tylko wtedy, gdy komunikujemy się z serwerem. Klient może być skonfigurowany do wykonywania obliczeń, tylko gdy komputer jest niewykorzystywany lub obliczać na stałe z niskim priorytetem. Program okresowo zapisuje stan obliczeń do pliku i wczytuje ten plik przy uruchomieniu, w związku z tym można robić postępy w obliczeniach nawet wtedy, gdy komputer jest często wyłączany.

Pakiety analizowane są pod kątem obliczenia mocy sygnału jako funkcji częstotliwości i czasu, potem szukamy kilku typowych wzorców funkcji mocy: impulsy (krótkie impulsy), sygnały ciągłe (sygnał wąskopasmowy z 20-sekundowa obwiednią Gaussa, odpowiadający wiązce radioteleskopu przechodzącej przez punkt), sygnały pulsujące (ciągłe sygnały pulsacyjne z przypadkową częstotliwością, fazą i cyklem pracy) i trójki lub sygnały potrójne (trzy równomiernie rozłożone impulsy o tej samej częstotliwości; zwykły sygnał pulsujący). Sygnały, których moc i dopasowanie przekraczają progi są zapisywane w pliku danych wyjściowych.

Zewnętrzne pętle różnią się dwoma parametrami:

- wartość przesunięcia Dopplera (ang. *Doppler drift rate*). Jeśli stałoczęstotliwościowy sygnał nadawcy jest przyspieszany względem odbiorcy (np. przez ruchy planetarne), wtedy sygnał otrzymany jest przesunięty w dziedzinie częstotliwości. Sygnały takie mogą być najlepiej wykrawane przez zagubione przesunięcia w oryginalnych danych, następnie przeszukanie sygnału stałoczęstotliwościowego. Przesunięcie nie jest znane, sprawdzamy 31555 różnych przesunięć, obejmując fizyczny zakres prawdopodobnego przyspieszenia.
- rozdzielczość częstotliwościowa (ang. *frequency resolution*). Obejmujemy 15 zakresów rozdzielczości częstotliwościowych od 0,075 do 1220,7Hz. Taki wzrost czułości moduluje sygnał, którego zawartość częstotliwości rozciąga się ponad zakres.

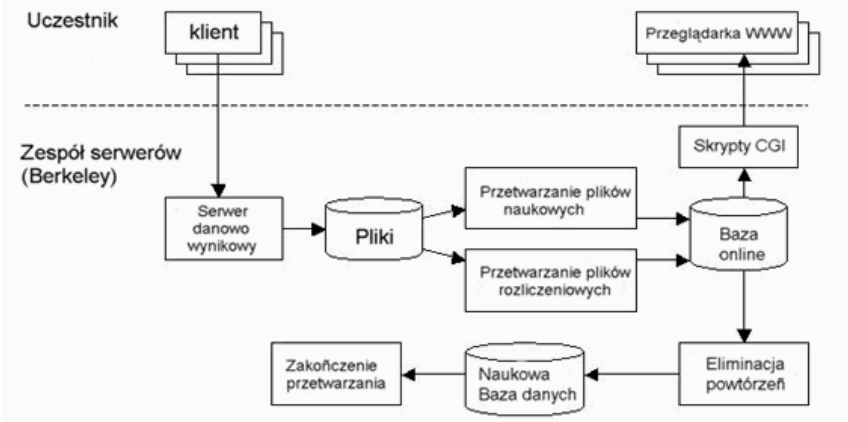


Rys. 45. Widok klienta SETI@home, pokazującego obecne widmo mocy czyli istotę obliczeń (u dołu) i najbardziej dopasowany Gaussian (u góry po lewo).

Program klienta SETI@home został napisany w C++. Składa się ze szkieletu niezależnego od platformy przetwarzania rozproszonego (6423 linie), zaimplementowanych komponentów dla określonej platformy, takie jak biblioteki graficzne (2058 dla wersji UNIX), kod analizujący dane specyficzny dla SETI (6572 linie) i kod graficzny specyficzny dla SETI (2247 linii).

Klient może być przeniesiony na 175 różnych platform. Można go uruchomić jako proces w tle, tak jak aplikacje GUI, lub jako wygaszasz ekranu. Wspierając te różne trybu na wielorakich platformach, używa się architektury, w której jeden wątek odpowiada za komunikacje i obróbkę danych, drugi wątek odpowiada za współdziałanie GUI i trzeci wątek (być może w odrębnym obszarze pamięci) przedstawia grafikę bazując na strukturze ze współdzielonej pamięci.

Wyniki są odsyłane na serwer SETI@home, gdzie są zapisywane i analizowane (rysunek 46).



Rys. 46. Odbieranie i analiza wyników.

Obsługa wyników obejmuje dwa zadania:

- Naukowe: serwer zapisuje dane do plików. Program czyta te pliki, tworzy wynik i zapisuje wynik w bazie danych. W celu optymalizacji przepustowości, kilka kopii programu jest uruchomione w tym samym czasie.

- Ewidencyjne: dla każdego wyniku serwer zapisuje logi wejściowe opisujące użytkownika przesyłającego wynik, jego CPU i tak dalej. Program czyta pliki z logami, gromadzi w pamięci podręcznej, uaktualnia wszystkie istotne rekordy w bazie danych (użytkownik, grupa, typ procesora i tak dalej). Co kilka minut pamięć podręczna jest przepisywana do bazy danych.

Dzięki buforowaniu uaktualnień w plikach, system serwera może funkcjonować w przypadku zaniku prądu lub przeładowania bazy danych.

Ostatecznie, każdy pakiet ma kilka wyników w bazie danych. Program eliminacji nadmiarowości egzaminuje każda grupę powtarzających się wyników – które mogą różnić się w liczbie sygnałów lub parametrach sygnałów – używając przybliżonej jednoznacznej polityki wybiera wynik „kanoniczny” dla tego pakietu. Kanoniczne wyniki są kopiowane do osobnej bazy.

Końcowa faza (ang. *back-end processing*) składa się z kilku kroków. Dla weryfikacji systemu, wybieramy testowe sygnały wstrzykiwane przez radioteleskop. Sztuczne sygnały (RFI) są identyfikowane i usuwane. Szukamy sygnałów o podobnych częstotliwościach i współrzędnych na niebie wykrytych w różnych chwilach. Te „powtarzające się sygnały”, jak również byłe sygnały o dostatecznej wartości, są badane dalej, potencjalnie prowadzący do powtórnego przebadania przez inne projekty radiowego SETI według przyjętego protokołu.

5.5.5.4. Publiczna reakcja na SETI@home

Plan utworzenia SETI@home ogłoszony został w 1998 r. i 400 tys. osób odpowiedziało wciągu następnego roku. W maju 1999 r. udostępnione zostały wersje klienta dla Windows’a i Macintosh’a. W ciągu tygodnia około 200 tys. pobrało i uruchomiło klienta, a liczba ta wzrosła do 3,83 mln osób w lipcu 2002 r. Ludzie z około 226 krajów uruchomili klienta SETI@home z czego prawie połowa była ze Stanów Zjednoczonych.

W 12 miesięcy od uruchomienia, w lipcu 2001 r., uczestnicy SETI@home przetworzyli 221 mln. pakietów. Średnia prędkość przesyłania danych w trakcie tego okresu była 27,36 TeraFLOPS. W sumie obliczenia zabrały $1,7*10^{21}$ operacji zmiennoprzecinkowych, najwięcej z oficjalnie udokumentowanych projektów.

5.5.5.5. Zakończenie

Wielkość publicznych zasobów obliczeniowych zależy od ilości komputerów osobistych posiadających nadwyżkę mocy obliczeniowej. Pomysł używania wolnej mocy obliczeniowej przez przetwarzanie rozproszone był zaproponowany przez projekt obliczeniowy Worm (Xerox PARC), który używał stacji roboczych wewnątrz laboratorium naukowo-badawczym, który później był zgłębiany przez akademickie projekty takie jak Condor.

Wielka skala publicznych zasobów obliczeniowych stałą się osiągalna dzięki rozwojowi Internetu w latach ‘90. Projekt GIMPS (ang. *The Great Internet Mersenne Prime Search*), który szukał liczb pierwszych rozpoczął się w 1996 r. Distributed.net, który demonstruje odszyfrowywanie metodą brute-force wystartował w 1997 r. Ostatnie zastosowania obejmują badanie składu białka (folding@home) i poszukiwanie leku (ang. *The Intel-United Devices Cancer Research Project*).

Kilka projektów jest w trakcie rozwijania uniwersalnych planów wykorzystania zasobów publicznych i innych projektów przetwarzania rozproszonego na dużą skalę. Projekty wspólnie nazwane architekturą Grid rozwijają systemy współdzielenia zasobów pomiędzy akademickimi i badawczymi organizacjami. Prywatne organizacje, takie jak Platform Computing, Entropia i United Devices rozwijają systemy dla przetwarzania rozproszonego i rozproszonego składowania danych zarówno dla zasobów publicznych jak i dla organizacji.

Na ogólniejszym poziomie, przy obliczeniach z wykorzystaniem zasobów publicznych jest aspekt "*peer-to-peer paradigm*" („paradygmat każdy z każdym”), który dotyczy przesuwania funkcjonowania intensywnych zasobów z centralnego serwera do stacji roboczych i domowych komputerów.

Jakie zadania są otwarte na przetwarzanie z wykorzystaniem publicznych zasobów? Jest kilka czynników. Po pierwszy, zadanie powinno mieć wysoki współczynnik wykonywanych obliczeń do ilości danych. Każdy pakiet SETI@home wymaga 3,9 tryliona (10^{18}) operacji zmiennoprzecinkowych, ale wymaga pobrania tylko 350KB i wysłania wyniku o wielkości około 1KB. Taki wysoki stosunek utrzymuje obciążenie serwera ruchem sieciowego na akceptowalnym poziomie i narzuca minimalne obciążenie klienta sieci. Aplikacje takie jak programy do renderingu grafiki wymagają olbrzymiej ilości danych na jednostkę obliczeniową, być może czyniąc je nieodpowiednimi do obliczeń z wykorzystaniem zasobów publicznych. Jednakże obniżenie kosztów użytkowania pasma internetowego złagodzi ten problem a techniki rozgłaszania mogą zmniejszyć koszty, kiedy olbrzymia część danych w jednostce roboczej jest stała.

Po drugie, zadania z niezależną równoległością są łatwiejsze do uchwycenia. Obliczanie próbek SETI@home jest niezależne, więc komputery uczestników nie muszą czekać lub komunikować się z innymi. Jeśli komputer zawiedzie podczas trwania procesu obliczeniowego, próbka jest przesyłana do innego komputera. Aplikacja, która wymaga częstych synchronizacji i komunikacji pomiędzy węzłami paraliżuje używanie bazy sprzętowej takiej jak dostępność współdzielonej pamięci systemów wieloprocesorowych i ostatnio przez programowe klastery obliczeniowe, takie jak PVM. Obliczenia z wykorzystaniem publicznych zasobów, z częstymi przestojami lub brakiem połączenia sieciowego wydaje się nieodpowiednie dla takich aplikacji. Jednakże szeregowanie mechanizmów, które odnajdują i wykorzystują grupy maszyn podłączonych do sieci LAN może wyeliminować te problemy.

Po trzecie, zadania które tolerują błędy są bardziej otwarte na obliczenia wykorzystujące zasoby publiczne. Dla przykładu: jeśli pakiet w SETI@home został przeanalizowany niewłaściwie lub wcale, to na cały projekt wpływa to tylko nieznacznie. Co więcej, przeoczenie jest korygowane w chwili gdy teleskop przegląda ten sam punkt nieba.

Ostatecznie, projekty obliczeniowe z wykorzystaniem zasobów prywatnych muszą przyciągać uczestników. Obecnie jest dosyć podłączonych do Internetu komputerów dla około 100 projektów o wielkości SETI@home, a interesujące i wartościowe projekty będą promowane w takich obszarach jak modelowanie globalnego klimatu i symulacja ekologiczna. Dla przyciągnięcia uczestników, projekt musi tłumaczyć i uzasadniać jego cel oraz musi zapewniać przykucie uwagi obrazem lokalnego i globalnego postępu. Graficzne wygaszacze ekranu są wspaniałym medium prezentacji jak również dostarczają wirtualnego marketingu. Sukces projektów wykorzystujących publiczne zasoby obliczeniowe będzie miał dodatkową korzyść z rosnącej publicznej świadomości naukowej i demokratyzowania się, rozszerzania, oraz alokacji zasobów badawczo-naukowych.

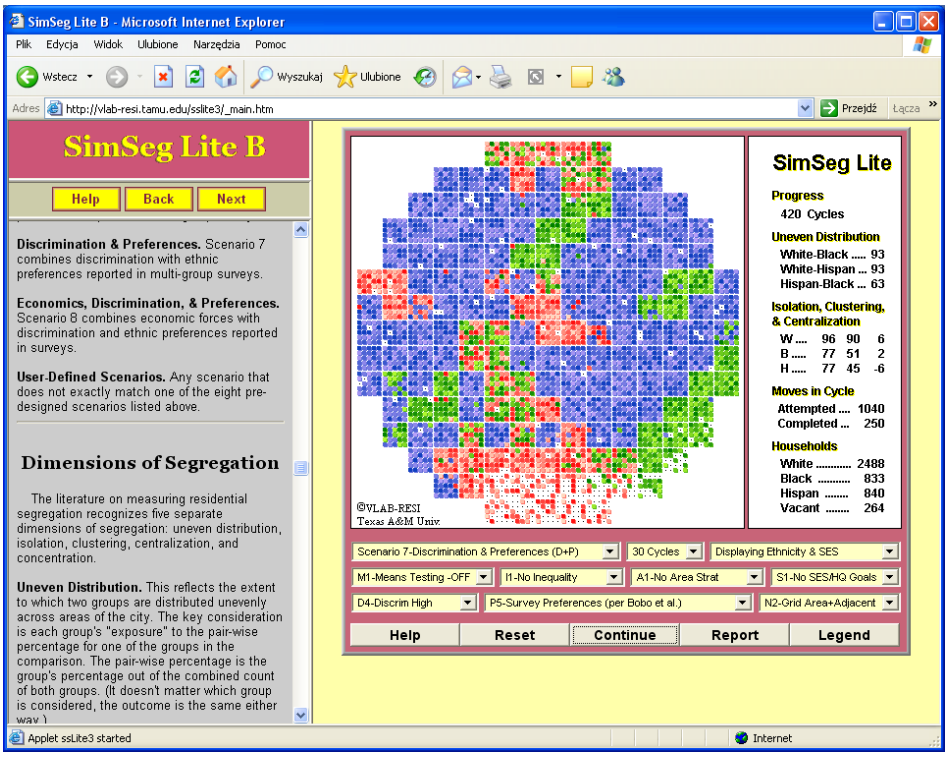
5.5.6. VLAB-RESI

http://vlab-resi.tamu.edu/vlab.htm

Większość funduszy laboratorium VLAB-RESI (ang. *A Virtual Laboratory in Racial & Ethnic Stratification & Inequality*) zostało przyznane przez Fundację Naukowo-Techniczą. Centrum znajduje się na Uniwersytecie Texas, Departamencie Socjologii. Laboratorium zajmuje się głównie ilustracją równomierności rozłożenia poszczególnych warstw rasowych. Jednym z celów jest również pokazanie konkretnych zjawisk socjologicznych przy użyciu dostępnych metod i technik informatycznych. Autorzy również w apletach poruszają temat dyskryminacji rasowej oraz jej wpływu na układ sił w poszczególnych kwadratach (rysunek 47).

Poprzez applet Javy przedstawione są wyniki następujących doświadczeń:

- InterGen – model transmisji nierówności etnicznych w cywilizacji,
- SegMaps –mapy segregacji etnicznych na przykładzie metra w Stanach Zjednoczonych,
- SimSeg – symulacja dynamicznej segregacji rasowej,
- SSW View – prezentacja wyników wygenerowanych przez SimSeg.

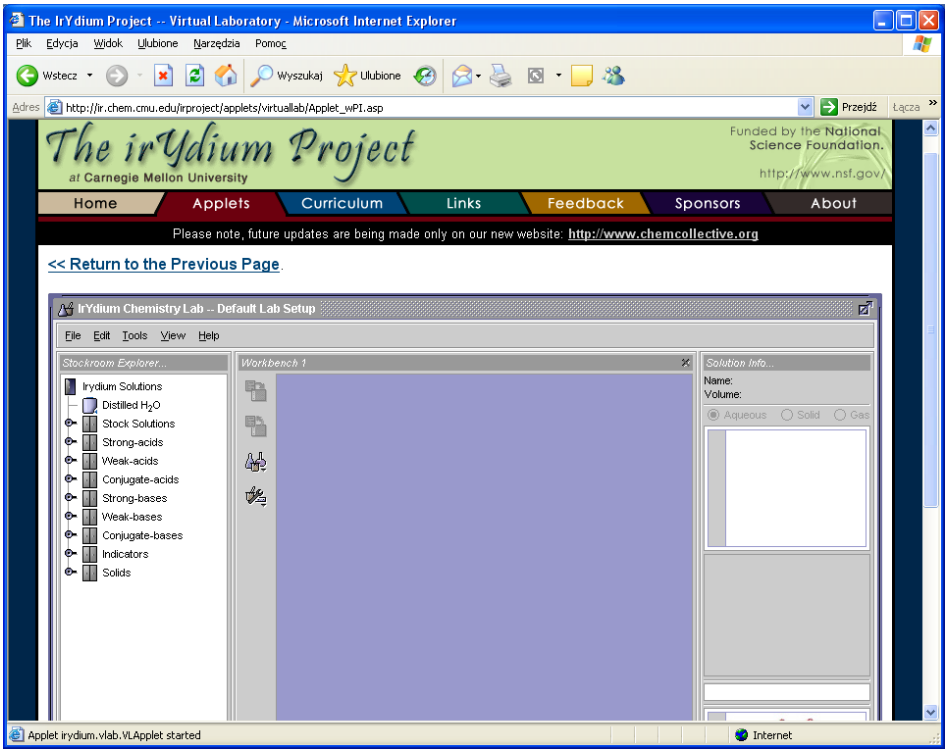


Rys. 47. Symulacja dynamicznej segregacji rasowej.

5.5.7. The IrYdium Project

http://ir.chem.cmu.edu/irproject/applets/virtuallab/

Laboratorium wirtualne zostało stworzone przez Uniwersytet Carnegie Mellon w 1997 roku i jest również sponsorowane przez National Science Foundation. Jest to typowe laboratorium chemiczne, umożliwiające studentom zaobserwowanie zachowania się konkretnych substancji. Dostępnych jest ponad 100 odczynników. Podobnie jak większość przypadków, do komunikacji z użytkownikiem użyto środowiska Java.

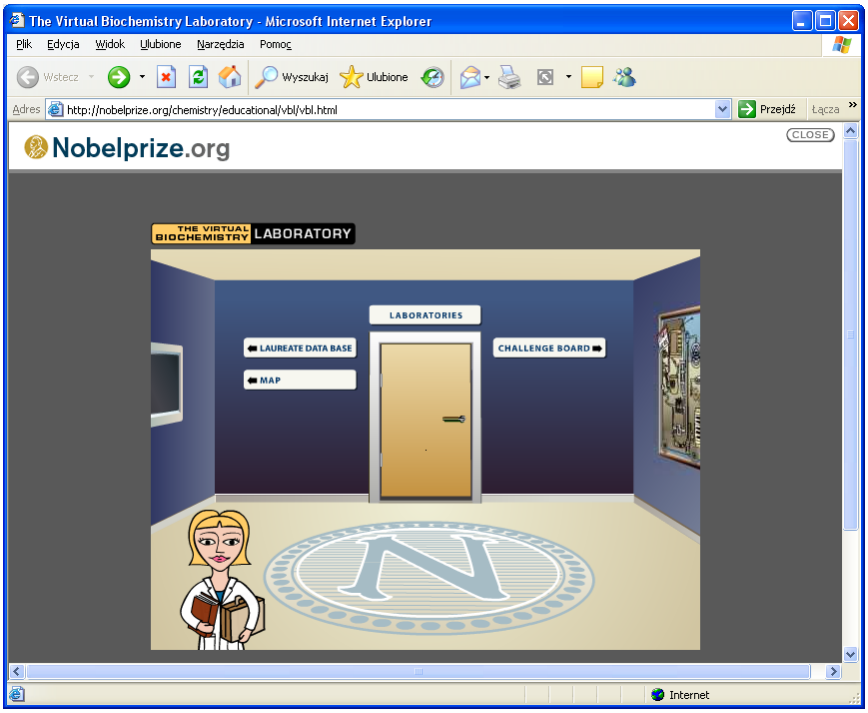


Rys. 48. Środowisko The IrYdium Project.

5.5.8. NobelPrize.org

http://nobelprize.org/chemistry/educational/vbl/vbl.html

Bardzo interesujące, interaktywne wirtualne laboratorium organizacji Alfreda Nobla. Ideą fundacji jest nauka poprzez zabawę.



Rys. 49. Witryna laboratorium NobelPrize.org.

Laboratorium zostało podzielone na kilka pokoi, zwiedzający w każdej chwili może skorzystać z pomocy asystentki, która w dowolnym momencie wyjaśni zasadę działania każdej aparatury pomiarowej. Użytkownik może zaprojektować konkretny eksperyment po czym obejrzeć rezultaty końcowe.

5.6. Podsumowanie

[15] Wirtualne Laboratoria z pewnością mają przed sobą wielką przyszłość. Postrzega się je jako pewne panaceum na problemy związane z zakupem drogich i unikatowych urządzeń czy z dostępem do nich (np. odległość). Obecnie technologie z nimi związane są dopiero w stanie raczkowania. Powstają nowe serwisy umożliwiające wykonanie prostych eksperymentów, w których można sterować niewielką liczbą parametrów. Można zaryzykować stwierdzenie, że obecnie przygotowywany jest grunt pod wirtualne laboratoria z prawdziwego zdarzenia. Tworzy się narzędzia sprzętowe (SGI Reality Center) i software'owe (CAVERN G2, Microsoft) umożliwiające porozumiewanie się i prezentację wyników na odległość. Jednak obecnie większość budowanych laboratoriów to konstrukcje mało zaawansowane, tworzone przez małą grupę ludzi dla potrzeb konkretnego eksperymentu. Brak jest obecnie uniwersalnych narzędzi do tworzenia wirtualnych laboratoriów.

Powstające rozwiązania przeznaczone są do zastosowań w określonym typie laboratoriów. Ich projektanci tworząc je najczęściej uzależniają ich budowę od specyfiki rozwiązywanych problemów. Istnieje potrzeba budowy pewnej struktury, którą będzie można zastosować (przystosować) do budowy każdego typu laboratoriów. [50]Budowa wirtualnych laboratoriów nie powinna się jednak ograniczać tylko i wyłącznie do narzędzi umożliwiających wykonanie eksperymentu ale również powinna zawierać narzędzia umożliwiające współpracę, wirtualne spotkania naukowców z różnych części świata współpracujących nad tym samym problemem. Początkowo mogą to być proste narzędzia typu chat, później mogą się rozwijać w systemy do przekazywania dźwięku, wideo i

ewoluować w kierunku teleimersji. W przyszłości wirtualne laboratoria będą wykorzystywały teleimersję jako sposób wykonania eksperymentu. Można by sobie wyobrazić następujący scenariusz. Naukowiec wchodzi specjalnego pomieszczenia, zakłada odpowiedni strój: rękawice, kombinezon, hełm lub okulary i dzięki teleimersji znajduje się w wirtualnym świecie imitującym laboratorium z urządzeniami potrzebnymi do wykonania odpowiedniego eksperymentu. Dzięki bardzo dużemu stopniowi teleimersji możliwe jest by naukowiec np. włożył odpowiednią próbkę do spektrometru, nastawił wymagane parametry i uruchomił eksperyment oglądając na bieżąco jego wyniki. Po wykonanym eksperymencie będzie można przejrzeć wyniki podobnych eksperymentów np. biorąc z półki wirtualną książkę i porównać je z wynikami otrzymanymi przez siebie. W tym wirtualnym świecie będą możliwe również spotkania naukowców, w czasie których mogłaby się odbywać dyskusja na problemami, które ich nurtują. Pomimo tego, że taki scenariusz brzmi obecnie jak opowiadanie typu science fiction to będzie on wkrótce bardzo prawdopodobny. Jest to tym bardziej realistyczne, że potrzebne do tego celu narzędzia już istnieją (SGI Reality Center Rooms, Cavern). Pewnym problemem okazują się tutaj przepustowości sieci komputerowych oraz moc maszyn obliczeniowych. Wszystko również wskazuje na to, że ograniczenia te wkrótce znikną. Powinniśmy wykorzystać ten czas twórczo by być przygotowanym na wejście nowych technologii.

Literatura

[1] G. Coulouris, *„Systemy rozproszone, Podstawy i projektowanie*"

Źródła

1. PCkurier 20/1998. „Zarządzanie Systemy rozproszone: W całości, a nawet w kawałkach..." Autor: Michał Szulowski

   Adres WWW: http://www.pckurier.pl/archiwum/art0.asp?ID=2302

2. Adres WWW: http://icis.pcz.pl/~roman/index2.html

   Strona "Finite Element Modelling & Parallel Computing"

   Obliczenia i programowanie równoległe - Parallel computing and programming .

   Organizacja procesów obliczeniowych w równoległych i rozproszonych systemach komputerowych.

   Rozdział 1.3. Architektury rozproszonych systemów komputerowych i ich systemy operacyjne

3. Adres WWW: www.oizet.p.lodz.pl/bazy/pliki/f.ppt

   Wydział Organizacji i Zarządzania Politechniki Łódzkiej. Materiały dydaktyczne do przedmiotu "Bazy danych".

4. Adres WWW: http://www.pr.radom.net/labto/bazy%20danych.php

   Komputerowe Laboratorium Teorii Obwodów-POLITECHNIKA RADOMSKA-Wydział Transportu.

   Dział „Dydaktyka – Bazy danych"

5. Adres WWW: http://www.pckurier.pl/archiwum/art0.asp?ID=6076

   PCkurier 9/2003 „Architektura GRID", Autor: Marek Rzewuski

6. Dokument „Grid computing i Oracle 10g w pytaniach i odpowiedziach „

   Adres WWW: www.oracle.com/global/pl/database/GC_QA.rtf

7. Dokument „System Informatyczny Narodowego Funduszu Zdrowia"

   Adres WWW: http://www.sygnity.pl/cms,WERSJA+PL,OFERTA,System+Informatyczny+NFZ,System+Informatyczny+NFZ

8. PCkurier 5/1999 „Kompleksowy System Informatyczny ZUS" Autor: Bolesław Urbański

   Adres WWW: http://www.pckurier.pl/archiwum/art0.asp?ID=2786

9. Dokument: „Wdrożenie aplikacji Noe.NET w Centralnym Zarządzie Służby Więziennej"

   Adres WWW: http://www.microsoft.com/poland/developer/net/rozwiazania/czsw.mspx

10. Dokument: „.NET na dłużników Największy Bank w Polsce wdrożył samodzielnie system do obsługi windykacji w technologii Microsoft .NET i z wykorzystaniem Microsoft SQL Server 2000

    Adres WWW: http://www.microsoft.com/poland/sql/wdrozenia/pkobp.mspx

11. Dokument: „Amadeus Polska wdrożył system do obsługi rezerwacji i rozliczeń dla biur podróży oparty na serwerze baz danych Microsoft® SQL Server 2005 i platformie Microsoft® Windows Server 2003."

    Adres WWW: http http://www.microsoft.com/poland/sql/wdrozenia/amadeus.mspx

12. Serwis Ministerstwa Finansów. Dział Służba Celna

    Adres WWW: http://www.mf.gov.pl/index_wai.php?const=2&dzial=397&wysw=4

13. Serwis Exporter.pl Adres WWW: Dzięki
    http://www.mf.gov.pl/index.php?dzial=396&wysw=2&const=2

14. Serwis Ministerstwa Finansów. Dział Służba Celna
    Adres WWW: http://www.mf.gov.pl/index.php?dzial=574&wysw=2&const=2
15. Dokument „Nowe aplikacje i usługi w środowisku Grid"
    Adres WWW: vlab.psnc.pl/pub/Nowe_Aplikacje_I_Uslugi_W_Srodowisku_Grid.pdf
16. Adres WWW: http://vlab.psnc.pl w dziale „Informacja o projekcie"
17. www.si.pjwstk.edu.pl/dydaktyka/IT_w_biznesie/SYR1.ppt prezentacja Kazimierza Subieta
18. aragorn.pb.bialystok.pl/~wkwedlo/OS1-1.pdf Wojciech Kwedlo
19. http://www.studianet.pl/so_1_izk/wiadomosci%20ogolne/systemy%20rozproszone.htm
20. http://p2p.ort.pl/
21. www.linux-magazine.pl/issue/06/KnowHow_klastry.pdf
22. www.ia.pw.edu.pl/~tkruk/openssi/cluster_afelkner.pdf
23. www.robomatic.pl/?id=enchaslo&idh=324
24. www.robomatic.pl/?id=enchaslo&idh=627
25. students.mimuw.edu.pl/SO/Wyklady-html/13_dfs/13_dfs.html
26. students.mimuw.edu.pl/SR/prace-mgr/gryz/node3.html
27. Wytwarzanie, integracja i testowanie SI
    www.ipipan.waw.pl/.../wyklady/Budowa%20i%20integracja%20systemow%20informatycznych%20BYT%202002/InzOpr07.ppt
28. studia.elka.pw.edu.pl/pub/BADA.A/BADA5.ppt
29. http://calypso.cs.put.poznan.pl/projects/ldsm/LDSM.html
30. http://www.eti.pg.gda.pl/katedry/kask/pracownicy/Michal.Piotrowski/sr-pliki/Transkacje%20rozproszone.pdf
31. http://aragorn.pb.bialystok.pl/~marekg/research/publications/grzes03msc.pdf
32. http://www.eti.pg.gda.pl/katedry/kask/pracownicy/Michal.Piotrowski/sr-pliki/Transkacje%20rozproszone.pdf
33. http://www.ia.pw.edu.pl/~ttraczyk/bd2/bd2_13.pdf
34. http://www.prz.rzeszow.pl/we/katedry/zsc/projekty/26/index.htm
35. http://rainbow.mimuw.edu.pl/SO/LabLinux/PROCESY/PODTEMAT_1/ntp.html
36. http://www.karpaty.edu.pl/algorytm_lamporta
37. http://mars.iti.pk.edu.pl/~bartm/wyklad.pdf
38. http://www.algorytm.org/index.php?option=com_content&task=view&id=20&Itemid=28
39. http://pjwstk.dyski.one.pl:81/public/ftp.pjwstk.edu.pl/zsuski/zso/Czesc%202/07-Koordynacja%20rozproszona.pdf
40. http://students.mimuw.edu.pl/SR-ZSI/Wyklady-html/03_synchro/03_synchro.html
41. http://www.prz.rzeszow.pl/we/katedry/zsc/projekty/26/index.htm
42. http://wazniak.mimuw.edu.pl/images/4/4a/Sr-11-wyk-1.0.pdf
43. http://www.ia.pw.edu.pl/~tkruk/openssi/ha_mnajs.pdf
44. www.is.pw.edu.pl/plik/238/wyklad%204.ppt
45. http://www.prokom.pl/pl/oferta/oferta.php?id=przyklady_zus
46. http://www.prokom.pl/pl/news/news.php?id=389
47. http://www.krakow.uc.gov.pl/ncts.htm
48. www.przemysl.ic.gov.pl/modules.php?name=Downloads&d_op=getit&lid=30

49. http://www.mf.gov.pl/index.php?dzial=397&wysw=4&const=2&PortalMF=40de83b24ff5b233117bcefa8e78dd38
50. http://vlab.psnc.pl/pub/lab_wirt.pdf
51. http://vlab.psnc.pl/modules/arch_labusersmanage.html
52. http://www.chip.pl/arts/archiwum/kts/ktsar_108955.html
53. http://serwisy.gazeta.pl/nauka/1,34150,3078969.html