Rozproszony system plików

DFS (ang. **D**istributed File **S**ystem) 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

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 nt fizycznego położenia pliku (np. /server1/dir1/dir2, ale gdzie jest server1?)
  - nazwa pliku oznacza określony, choć ukryty, zbiór bloków dyskowych
  - może ujawniać zależność między składowymi nazwy a komputerami
  - nie jest możliwa automatyczna zmiana położenia pliku
- Niezależność położenia - nazwy pliku nie trzeba zmieniać wtedy, gdy plik zmienia swoje fizyczne położenie
  - lepsza abstrakcja pliku (nazwa określa zawartość, nie położenie)
  - oddziela hierarchię nazw od hierarchii urządzeń pamięci

Schematy tworzenia nazw

1. Nazwa pliku składa się z nazwy komputera macierzystego i nazwy lokalnej; gwarantowana jednoznaczność w całym systemie
2. Zdalne katalogi są montowane w lokalnym katalogu tworząc spójne drzewo katalogów; dostęp przezroczysty jedynie do wcześniej zamontowanych katalogów (np. NFS)
3. Pełna integracja składowych systemów plików
   - jedna globalna struktura nazw obejmuje wszystkie pliki w systemie
   - jeśli serwer jest niedostępny, to pewien zbiór katalogów też staje się niedostępny (np. Locus, Sprite, Andrew)

Semantyka współdzielenia pliku

- 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)

Zdalny dostęp do plików

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

- jeśli potrzebnych danych nie ma w pamięci podręcznej, to sprowadza się ich kopię z serwera
- klient korzysta z kopii przechowywanej w pamięci podr.
- pliki identyfikuje się z kopią główną w serwerze, ale w różnych pamięciach podręcznych w sieci mogą przebywać

2. Problem utrzymania spójności pamięci podręcznych, tzn. zgodności kopii podrzędnych z kopią główną

3. Gdzie przechowywać pliki: dysk serwera, pamięć główna serwera, dysk klienta, pamięć główna klienta

4. Zalety dyskowych pamięci podręcznych
- niezawodność (nie przepadają podczas awarii)
- dane przechowywane w pamięci podręcznej na dysku pozostają tam podczas rekonstrukcji systemu po awarii i nie trzeba ich ponownie sprowadzać

5. Zalety pamięci podręcznej w pamięci głównej
- umożliwiają korzystanie z bezdyskowych stacji roboczych
- krótszy czas dostępu do danych
- 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

6. Aktualizowanie danych w pamięci podręcznej
- ***Natychmiastowe pisanie*** (ang. *write-through*) - przesyła się dane do serwera natychmiast po umieszczeniu ich w pamięci podręcznej. Niezawodne, ale słaba wydajność

- ***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

7. Weryfikacja aktualności danych - czy kopia lokalna w pamięci podręcznej jest zgodna z kopią główną? weryfikację zgodności może zainicjować klient lub serwer

Porównanie obsługi zdalnej i pamięci podręcznej

- Pamięć podręczna pozwala obsługiwać większość żądań zdalnego dostępu tak szybko jak żądania lokalnego dostępu
- Powoduje, że kontakt z serwerem jest rzadszy:
  - mniejsze obciążenie serwera i ruch w sieci
  - większa możliwość skalowalności
- 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)
- Pamięć podręczna sprawdza się lepiej, gdy żądania pisania są rzadkie (gdy częste, duży narzut na utrzymanie zgodności)
- Pamięć podręczna pozwala osiągać korzyści, gdy wykonanie odbywa się na komputerze z lokalnymi dyskami lub dużą pamięcią główną

- Zdalny dostęp na komputerach bezdyskowych i z małą pamięcią główną trzeba realizować poprzez zdalną obsługę

Stanowy (ang. *stateful*) serwer plików

- 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
- 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

- Każde żądanie jest samowystarczalne, więc nie trzeba przechowywać informacji o stanie
- Każde żądanie identyfikuje plik i pozycję w pliku
- Nie trzeba otwierać i zamykać połączenia (zbędne open i close dla pliku)
- Nie trzeba przeznaczać miejsca na pamiętanie informacji o stanie
- Nie ma ograniczeń na liczbę otwartych plików

Różnice między serwerem stanowym i bezstanowym

- 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

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

Tworzenie kopii pliku (ang. *file replication*)

- Zwiększa dostępność i może skrócić czas dostępu
- Umożliwia uniknięcie sytuacji, gdy pojedynczy serwer staje się wąskim gardłem
- Istnienie wielu kopii powinno być niewidoczne na wyższych poziomach; na niższych poziomach kopie muszą się różnić nazwami
- Aktualizacja jednej kopii powinna być przeprowadzona również na pozostałych kopiach
- Kopiowanie na żądanie - czytanie zdalnej kopii powoduje zapamiętanie jej w pamięci podręcznej, a więc utworzenie lokalnej kopii
- 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

N - liczba kopii

N_r - liczba głosów dająca prawo do czytania (*read quorum*)

N_w - liczba głosów dająca prawo do pisania (*write quorum*)

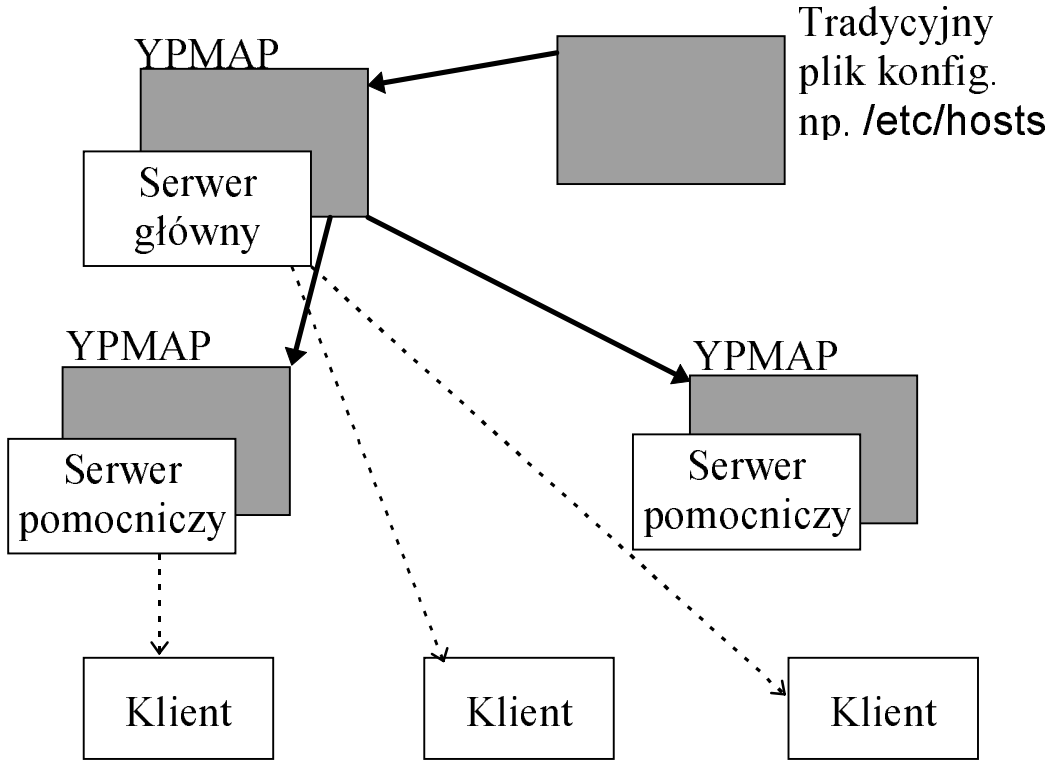
musi zachodzić warunek: $N_r + N_w > N$

np $N = 12$; $N_r = 3$, $N_w = 10$; $N_r = 7$, $N_w = 6$; $N_r = 1$; $N_w = 12$

- głosowanie z duchami (pozornymi serwerami)

NIS (Network Information Service)

Produkt firmy Sun (dawniej znany jako *yellow pages*). Ułatwia zarządzanie dużymi grupami stacji roboczych, tworząc jeden zbiór systemowych baz danych, które grupują informacje przechowywane dotychczas w plikach konfiguracyjnych w każdej stacji. Bazy danych są używane do konfigurowania systemu, mogą być również przeglądane przez użytkowników. Przechowuje pary (*klucz*, *wartość*), np. (*nazwa użytkownika*, *zaszyfrowane hasło*), (*nazwa komputera*, *adres sieciowy*)
Obecnie: **NIS+**



- Dla każdego pliku konfiguracyjnego tworzy się dwie bazy danych; np. dla /etc/hosts → hosts.byname i hosts.byaddr
- Opcjonalnie można utworzyć kopię baz danych na serwerach pomocniczych; pozwala to zmniejszyć obciążenie serwera głównego
- Serwer dla klienta jest wybierany dynamicznie (odpowiada za to demon ypbind wykonywany na komputerze klienta)
- Serwer - tworzy dazy danych z plików konfiguracyjnych poleceniem makedbm (w katalogu /etc/yp)
- Klient - wsparcie na trzech poziomach:

⇒ Polecenia interpretatora: ypmatch, ypcat

⇒ Bazy danych, funkcje dostępu, struktury danych:

group	getgrent(),getgrid(),getgrname()	group
passwd	getpwent(),getpwuid(),getpwnam()	passwd
hosts	gethostent(),gethostbyaddr(),gethostbyname()	hostent
services	getservent(),getservbyport(),getservbyname()	servent
networks	getnetent(),getnetbyaddr(),getnetbyname()	netent
rpc	getrpcent(),getrpcbynumber(),getrpcbyname()	rpcent

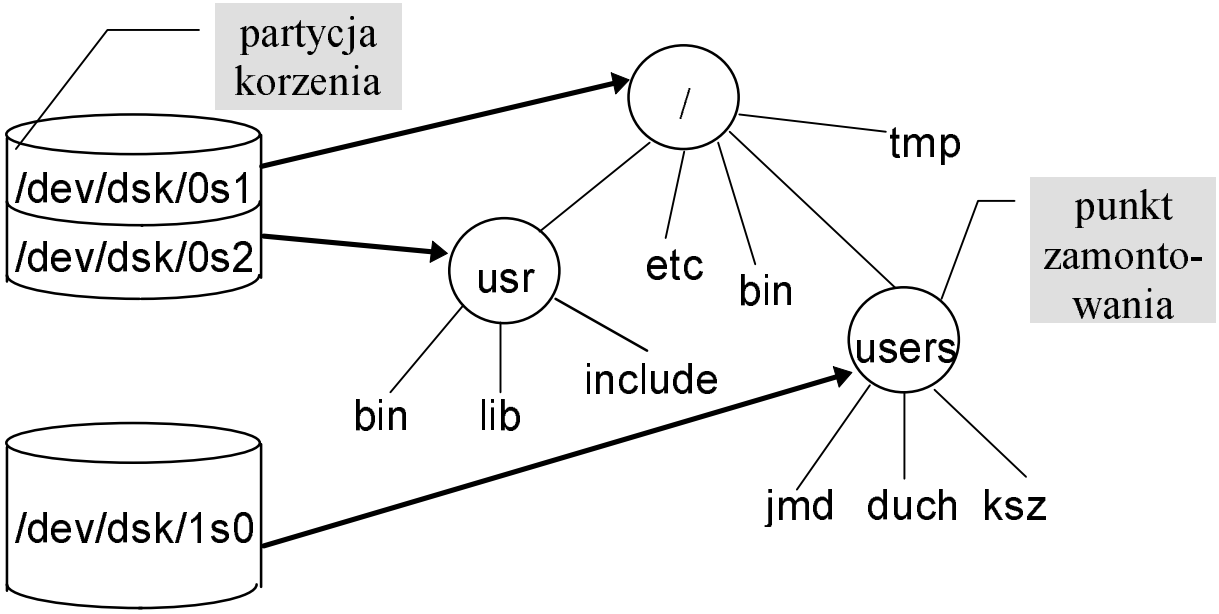
⇒ Funkcje niskiego poziomu: yp_get_default_domain(), yp_match()

NFS (Network File System)

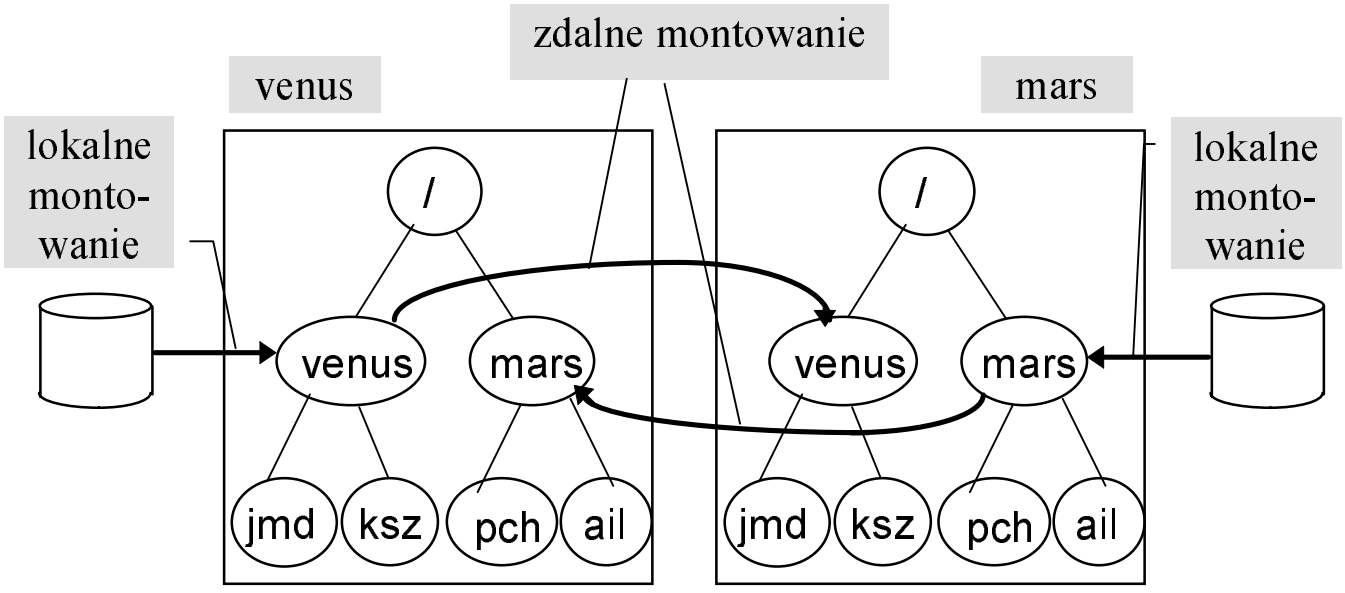
- Produkt firmy Sun Microsystems. Powstał w roku 1984
- NFS dostarcza wspólny system plików komputerom o różnej architekturze i różnych systemach operacyjnych
- Klienci i serwery zwykle pracują w tej samej sieci lokalnej (lecz nie jest to konieczne)
- Każdy komputer może być równocześnie klientem i serwerem NFS
- Każdy serwer NFS eksportuje jeden lub więcej katalogów na potrzeby zdalnych klientów (ich listę zawiera plik /etc/exports)

- Klient uzyskuje dostęp do eksportowanych katalogów montując je. W przypadku stacji bez dysku montuje się zdalny katalog w katalogu głównym
- Różni klienci mogą zamontować te same katalogi i współdzielić pliki

Partycje dyskowe i punkty zamontowania



Zdalne montowanie



Protokoły NFS

⇒NFS korzysta z protokołu RPC, a RPC z XDR, dlatego protokół jest niezależny od konkretnej architektury i reprezentacji danych

⇒Definiowany przez NFS zbiór operacji na plikach nie jest specyficznie unixowy (choć dość bliski); istnieją implementacje NFS dla wielu innych SO

⇒Serwer NFS nie pamięta informacji o otwartych połączeniach → *serwer bezstanowy.*

1.Obsługa montowania

- Klient wysyła komunikat z nazwą katalogu, który ma być zamontowany
- Jeśli nazwa OK i katalog jest eksportowany, to serwer przekazuje klientowi *uchwyt do pliku* (ang. *file handle*), który identyfikuje: typ systemu plików, urządzenie, i-węzeł katalogu itp.
- Kolejne operacje wymagają dostarczenia uchwytu do pliku
- Plik /etc/rc zawiera polecenia montowania wykonywane automatycznie podczas inicjalnego ładowania systemu
- *Automontowanie*: z lokalnym katalogiem jest związany zbiór zdalnych katalogów. Dopiero przy próbie odwołania do zdalnego pliku SO wysyła komunikat do każdego serwera. Montuje ten katalog, którego serwer zgłosi się jako pierwszy

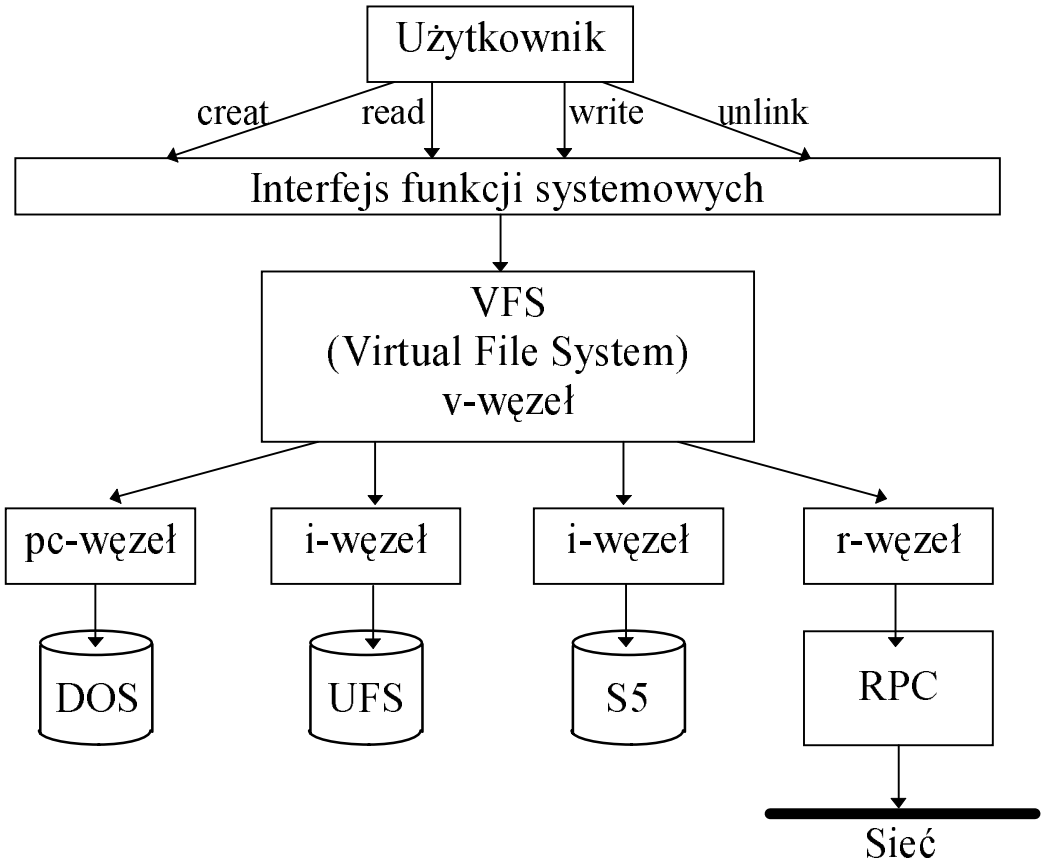
2.Obsługa dostępu do plików

- Dostępna jest większość unixowych operacji na plikach (z wyjątkiem open i close)
- Żeby odczytać plik, klient wysyła komunikat lookup z nazwą pliku → serwer przekazuje uchwyt do pliku.

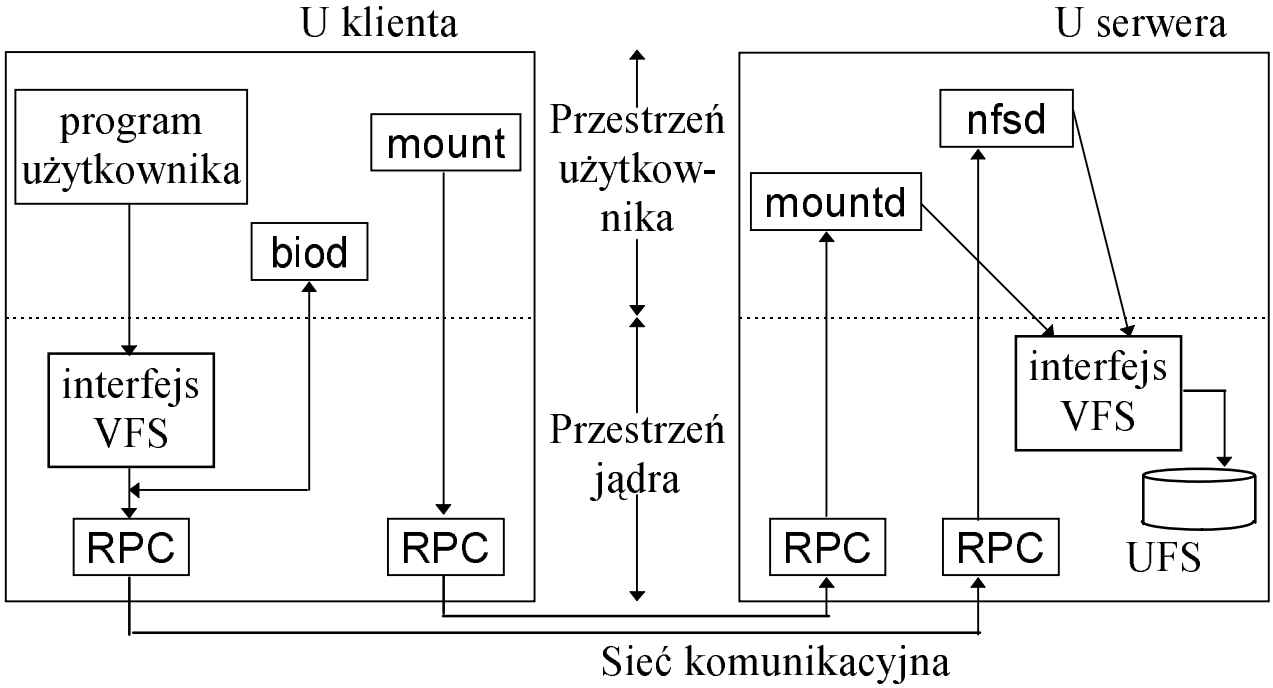
Operacja lookup nie kopiuje niczego do wewnętrznych tablic systemowych serwera

- Komunikat read (write) zawiera uchwyt do pliku, pozycję w pliku i liczbę bajtów. Jest to pełna informacja niezbędna do wykonania operacji.
- NFS przegląda kolejne składowe nazwy ścieżkowej; **dla każdej** wysyła komunikat lookup i dostaje uchwyt do pliku. Pozwala to abstrahować od sposobu zapisu nazwy ścieżkowej w konkretnym SO oraz poruszać się wzdłuż ścieżki do góry. Wydajność można poprawić buforując po stronie klienta uchwyty do często odwiedzanych katalogów
- Możliwość autoryzacji dostępu. Klucze używane do autoryzacji dostępu i inne informacje są przechowywane przez **NIS**

Implementacja NFS



- Warstwa najwyższa obsługuje funkcje open, read, close itp.
- Po analizie składniowej zostaje wywołana warstwa wirtualnego systemu plików (VFS). Przechowuje ona tablicę z jedną pozycją dla każdego otwartego pliku. Jest to *v-węzeł*. V-węzeł definiuje plik jako lokalny lub zdalny. W przypadku zdalnych przechowuje informacje pozwalające na dostęp do pliku
- mount - powoduje nawiązanie kontaktu z serwerem, uzyskanie uchwytu do montowanego katalogu. Wywołuje się funkcję systemową mount z tym uchwytem jako argumentem. Jądro konstruuje v-węzeł i zleca NFS utworzenie *r-węzła*. Każdy v-węzeł wskazuje na r-węzeł w kodzie klienta NFS lub i-węzeł w lokalnym SO.
  Gdy jądro podczas analizy nazwy ścieżkowej pliku natrafi na r-węzeł, to wie, że katalog jest zdalny. Zleca NFS otworzenie pliku. NFS analizuje pozostałe składowe nazwy ścieżkowej i przekazuje uchwyt do pliku
- Na serwerze NFS chodzą dwa demony:
  nfsd (kilka instancji; 4-8, nawet do 50) - odbiera z sieci żądania wykonania operacji NFS i przekazuje je do lokalnego systemu plików. Każdy proces demona obsługuje na raz jedno żądanie
  mountd (jedna instancja) - obsługuje żądanie zdalnego zamontowania
- Po stronie klienta:
  biod (kilka instancji; 4-8) - odpowiedzialny za *czytanie z wyprzedzeniem* i *opóźnione pisanie* w imieniu procesów, które używają zdalnych plików



NFS z punktu widzenia programisty

- Dobra wydajność czytania, gorsza pisania (podczas pisania buforowanie ma miejsce jedynie po stronie klienta; po stronie serwera write jest synchroniczne)
- Inna semantyka niektórych operacji unixowych (np. Unix pozwala na usunięcie otwartego pliku, a w NFS trzeba taką operację specjalnie symulować; w Unixie plik można otworzyć i zablokować, a w NFS jest potrzebny odrębny mechanizm blokowania dostępu)
- Nieefektywne blokowanie rekordów

Ograniczenia NFS

- Klient musi wiedzieć, które systemy plików są eksportowane przez poszczególne serwery
- Przenoszenie systemu plików z jednego serwera do drugiego wymaga powiadomienia wszystkich klientów
- Problemy z buforowaniem danych po stronie klienta