Prezentacja AFS

Andrzej Kurach

13 stycznia 2003

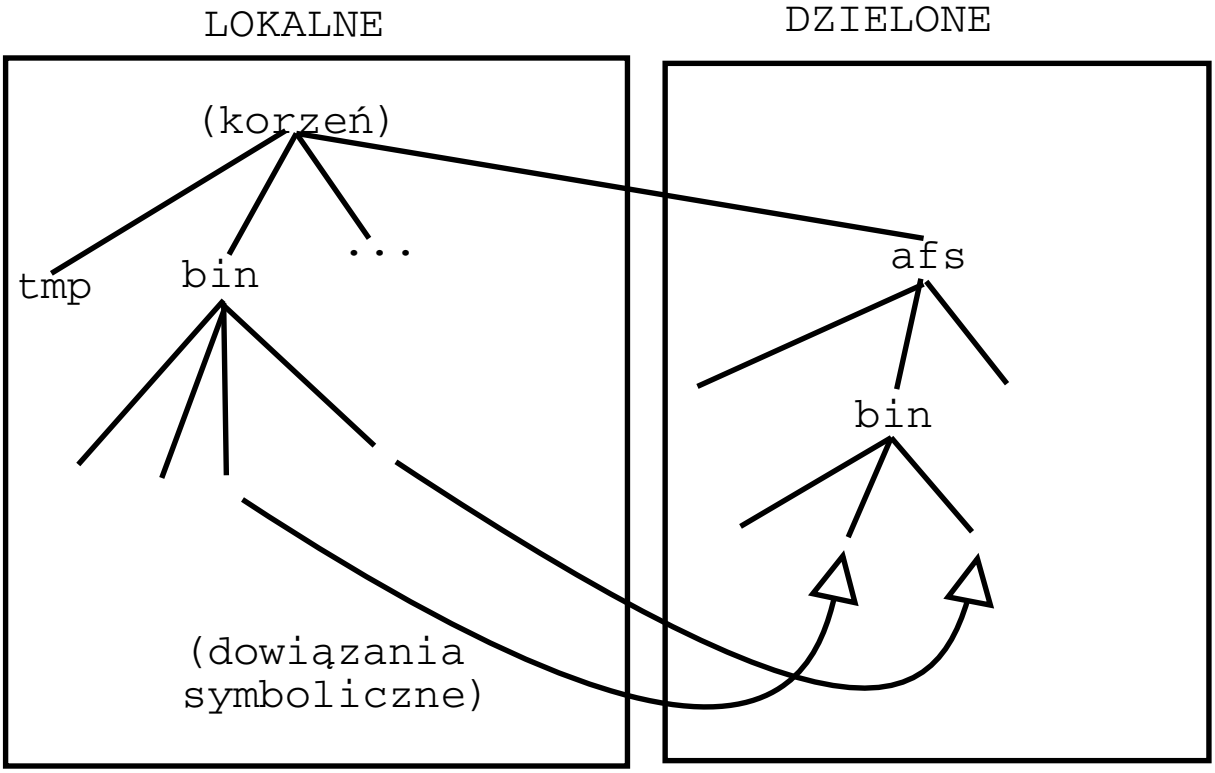
1 AFS

1.1 Wprowadzenie

- **Andrew** jest rozproszonym środowiskiem obliczeniowym opracowanym w Carnegie-Mellon University (CMU) jako ogólnouniwersytecki system informacyjno - obliczeniowy.
- System plików Andrew (ang. ***Andrew File System***), nazywany jest w skrócie **AFS**, został zaprojektowany dla środowiska Andrew. AFS umożliwia dzielenie informacje na dużą skalę (przy projektowaniu, zakładano, ze Andrew będzie pracować na 5000-10000 stacji roboczych).

Skoncentruje się głównie na omówieniu AFS-2, jest to pierwsza „fabryczna" implementacja AFSu. Dla uproszczenia skoncentruje się tylko na AFS zrealizowanego w sieci stacji roboczych i serwerów działających pod nadzorem systemu operacyjnego UNIX.

1.2 Wprowadzenie cd.

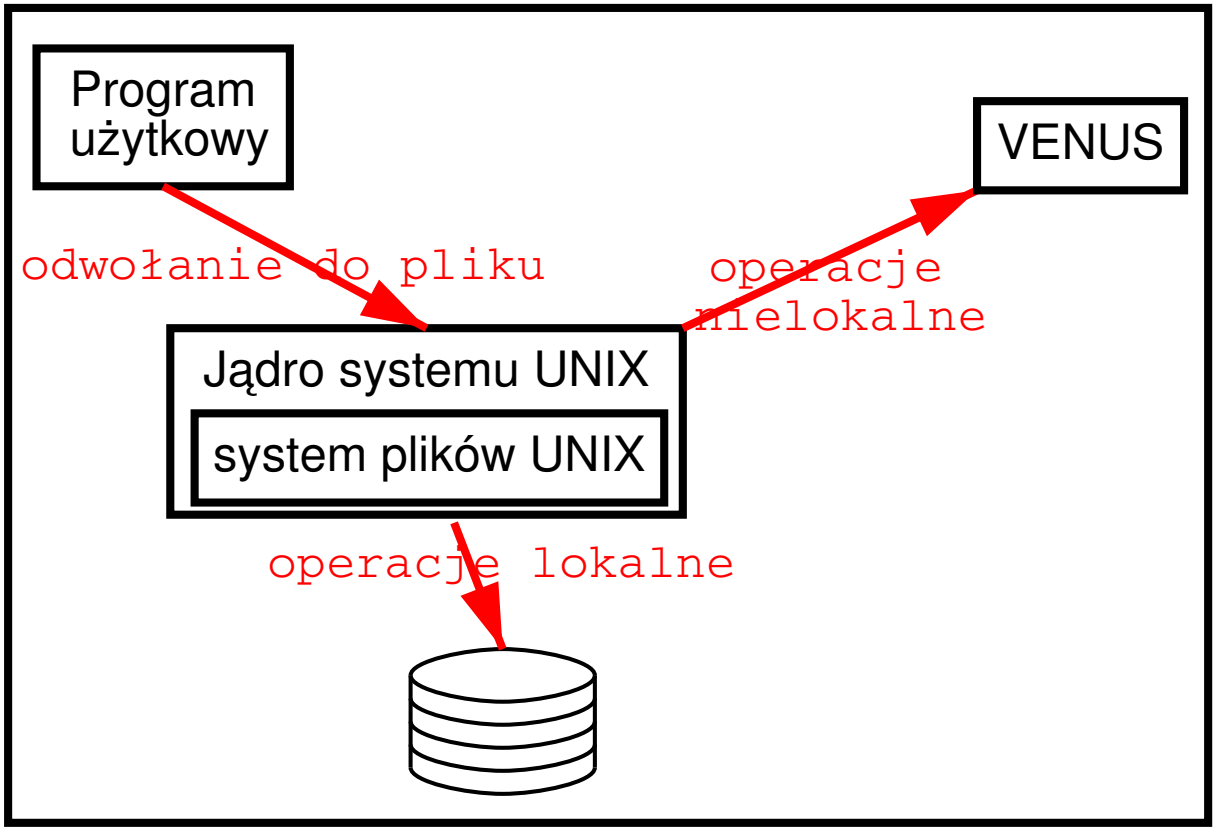


Przestrzeń plików w AFS ma strukturę drzewiastą. Użytkownik widzi jednolita przestrzeń nazw plików. Ścieżki dostępu do plików powinny być takie same, niezależnie od miejsca logowania (przezrczystość dostępu). Drzewo posiada korzeń, do którego jest podmontowany wierzchołek drzewa AFS (najczęściej podmontowany pod ścieżką */afs*).

Skalowalność - zadawalająca działanie w obecności większej liczby aktywnych użytkowników (w porównaniu do innych systemów rozproszonych)

Nietypowe właściwości AFSu (w stosunku do NFS):

- Usługi są *CAŁOPLIKOWE*. Serwery przesyłają cale zawartości plików.

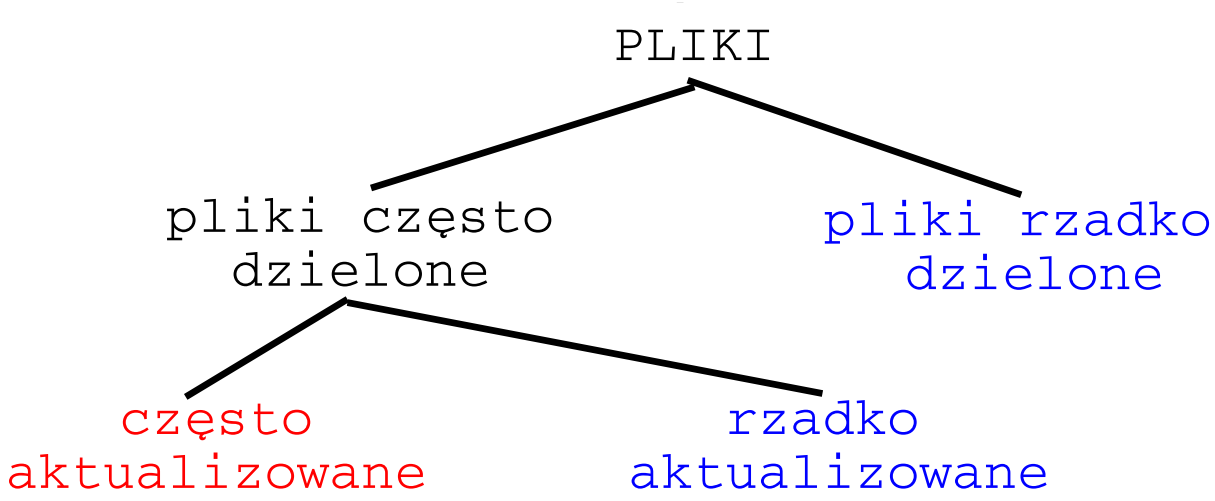


- *CACHE*, czyli **plikowa pamięć podręczna**. Po przesłaniu kopii pliku do komputera klient zapamiętuje plik w pamięci podręcznej na lokalnym dysku. Pamięć ta powinna przechowywać kilkaset najczęściej ostatnio używanych przez dany komputer plików średniej wielkości. Lokalne kopie plików maja pierwszeństwo przed kopiami w czasie operacji *open*.

Dodatkowo istotny wpływ na kształt AFS miał fakt, iż system plików został skonstruowany w środowisku akademickim i dla systemu działającego na uniwersytecie. Oto najważniejsze obserwacje tam zauważone:

- pliki są małe - większość jest mniejsza niż 10KB
- operacje czytania są częstsze od zapisywania (około 6 krotnie)
- dostęp sekwencyjny jest typowy; losowy rzadkością
- większość plików jest czytanych i zapisywanych przez jednego użytkownika; a nawet jak plik jest dzielony to na ogól tylko jeden użytkownik zmienia jego zawartość.
- odwołania do plikowi są skumulowane; jeżeli odwołanie do pliku nastąpiło niedawno to istnieje duża szansa kolejnego odwołania

1.3 Przeznaczenie AFSu



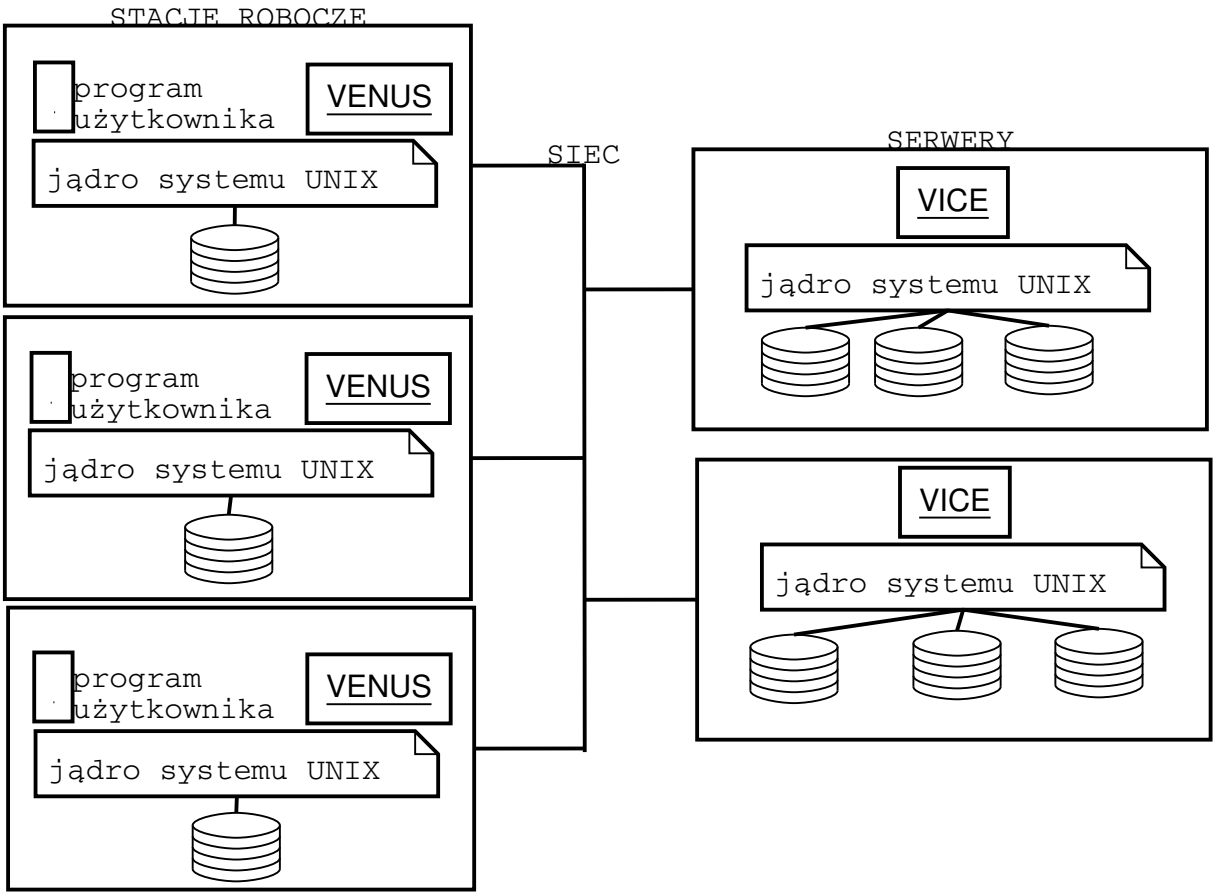
Pliki **dzielone i rzadko aktualizowane** to głównie pliki zawierające kod poleceń, bądź biblioteki systemu UNIX. AFS sprawdza się w tego typu plikach, gdyż pliki z *Cache* 'a rzadko będą aktualizowane.

Pliki **rzadko dzielone** to głównie indywidualnych użytkowników. W tym przypadku możemy liczyć, że aktualizowane kopie znajdują się w podręcznej pamięci.

Pliki **dzielone i czesto aktualizowane** to głównie pliki bazodanowe. Twórcy systemu AFS wykluczyli ze zbioru swych celów udogodnień dla aplikacji bazodanowych. Stwierdzili, że problem ten powinien zostać rozwiązany osobno. Dodatkowo dochodzi utrudnienie związane z transakcjami CAŁOPLIKOWYMI.

1.4 Realizacja podstawowych operacji

1.4.1 Podstawy

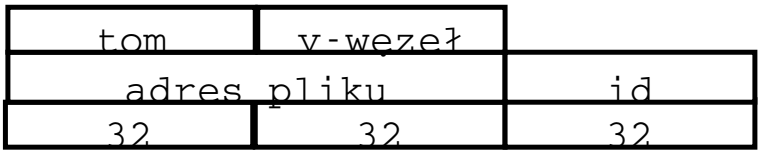


Dla AFS funkcjonuje klasyczny model komunikacji klient-serwer, czyli skalda się z dwu części, z których każda istnieje jako proces systemu UNIX:

1. **VICE** proces serwera
2. **VENUS** ewentualnie ***Cache Manager*** to proces klienta

Zadaniem VENUS jest tłumaczenie nazw plików na stosowne fid-y. VICE udostępnia klientom podstawowa funkcjonalność usług plikowych.

1.4.2 FID



FID składa się z następujących części:

1. ***Wolumenu***, inaczej **tomu**, tzn. z postawowej jednostki składowej AFSu. W systemie Andrew były to na ogół pliki pojedynczego klienta. Kilka wolumenów może stworzyć jedna strefę dysku. Koncepcja zespalania wolumenów jest podobna do Uniksowego montowania, z ta różnica ze w uniksie montuje się całą strefę dyskowa (a w AFS niekoniecznie). Wolumeny:

    - są kluczowa jednostka administracyjna
    - identyfikują i lokalizują pliki

2. ***v-wezel*** jest używany jako indeks do tablicy zawierającej i-wezly plików pojedynczego wolumenu.

3. ***Wyróżnik*** (ang. ***Uniquifier***) to jednoznacznie wyznaczony ciąg znaków dla każdego pliku (unikalny identyfikator)

W systemie plików AFS pliki identyfikuje się nie po nazwach i ścieżkach dostępu, ale po unikalnym identyfikatorze. Informacje o lokalizacji wolumenu jest przechowywana w ***bazie danych o lokalizacji wolumenów***, ktorej repliki znajdują się w każdym systemie obsługi.

Osiągamy w ten sposób:

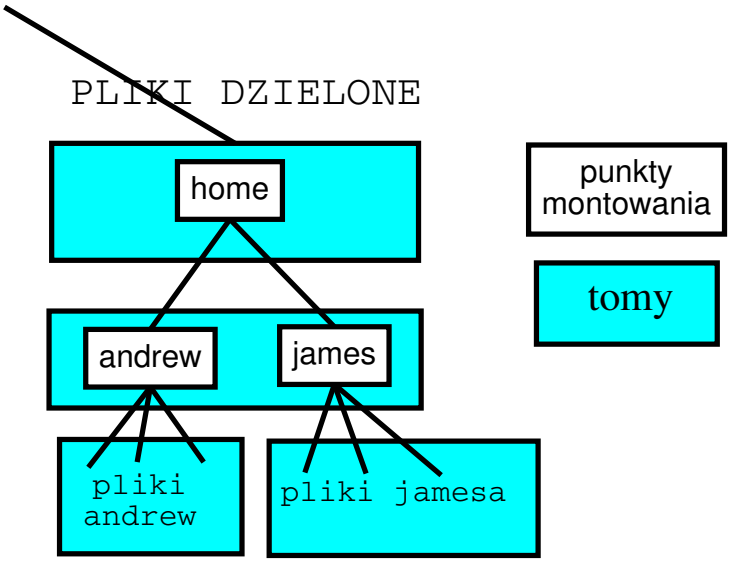
- **przezroczystość lokalizacji** - nazwa pliku nie daje jakiejkolwiek wskazówki odnośnie do fizycznego miejsca przechowywania pliku
- **niezależności lokalizacji** - nazwy pliku nie trzeba zmieniać wtedy, gdy plik zmienia swoje fizyczne umiejscowienie

1.4.3 Podstawowe operacje

Proces użytkownika	**Jądro systemu UNIX**	**VENUS**	**SIEC**	**VICE**
open(plik, tryb)	Jeśli plik odnosi się do pliku w dzielonej przestrzeni plików, to przekazuje zamówienie do procesu VENUS	Sprawdza wykaz plików w lokalnej pamięci podręcznej. Jeśli pliku tam nie ma albo jego znacznik jest nieważny, to zamawia plik u tego serwera VICE, który jest opiekunem tomu zawierającego dany plik.		

Proces użytkownika	**Jądro systemu UNIX**	**VENUS**	**SIEC**	**VICE**
	Otwiera plik lokalny i zwraca jego deskryptor do programu użytkowego.	Umieszcza kopie pliku w lokalnym systemie plików, wpisuje jego nazwę lokalna na wykaz plików przechowywanych w pamięci podręcznej i zwraca nazwę lokalna do systemu UNIX	→ ←	Przesyła do stacji roboczej kopie pliku i obietnice zawiadomienia. Rejestruje obietnice zawiadomienia.
read(deskr, Bufor, długość)	Wykonuje zwykła uniksowa operacje czytania kopii lokalnej			
write(deskr, Bufor, długość)	Wykonuje zwykła uniksowa operacje pisania kopii lokalnej			
close(deskr)	Zamyka kopie lokalna i powiadomienia proces VENUS o zamknięciu pliku.	Jeśli kopia lokalna, został zmieniona, to wysyła kopie do tego serwer VICE, który jest opiekunem pliku.	→	Zastępuje plik przesyłanymi danymi i wysyła zawiadomienia do wszystkich innych klientów mających obietnice zawiadomienia o tym pliku.

1.5 Tomy



Tom jest podobny do katalogu, składa się :

- plików
- punktów montowania

Tomy są niewidoczne dla użytkownika. Przechodząc po drzewie katalogów nie jest w stanie okreslic w jakim tomie się znajduje. Co więcej nie powinien odczuć „przejście”przez punkty montowania.

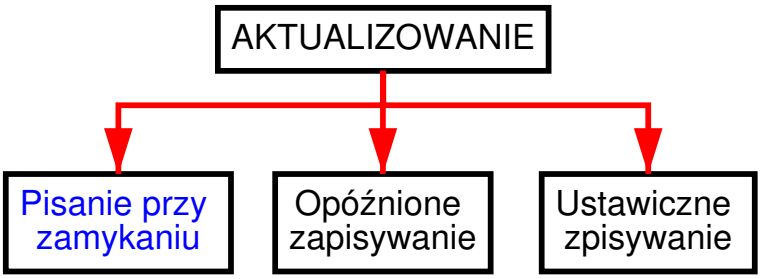
Tomy mogą być ograniczone, ten prosty mechanizm umożliwia administratorowi w prosty sposób ograniczając rozmiar katalogu domowego użytkownika. Tom powinien się zmieścić na jednej partycji.

W punktach montowania przechowuje się nazwy plików. Dzięki ***„bazie danych lokalizacji wolumenów”*** umożliwiono przenoszenie plików pomiędzy serwerami (bądź partycjami).

W AFS można:

- przenosić tomy
- replikować tomy. Kopiowanie tomów na kilka serwerów w celu zwiększenia dostępności (jest jedna kopia główna, służąca od zapisu reszta tylko do odczytu. Są pewne problemy ze spójnością danych)
- robić kopie zapasowe

1.6 Aktualizacja

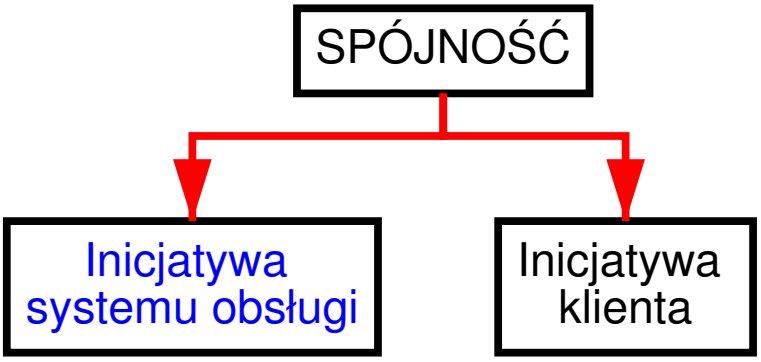


Zapisywanie danych występuje tylko podczas wykonywania *close*. Polityka ta:

Nie Sprawdzi się	**Sprawdza się**
• dla plików otwieranych na krótki okres czasu • rzadko modyfikowanych	• dla plików otwieranych na długie okresy czasu i czesto modyfikowalnych

Zaletą **zapisywania przy zamykaniu** jest niezawodność, gdyż nie zapisane dane nie giną przy awarii maszyny użytkownika (w przeciwieństwie do opóźnionego pisania) oraz jest to rozwiązanie o dużej wydajności (w przeciwieństwie do ustawicznego pisania)

1.7 Spójność



W system Andrew zaimplementowano semantykę sesji, tzn.:

- wynik operacji pisania otwartego pliku nie jest natychmiast widoczny dla innych użytkowników mających równocześnie otwarty ten sam plik.
- po zamknięciu pliku zmiany, które były do niego prowadzone będą widoczne tylko w następnych sesjach. Otwarte pliki nie będą odzwierciedlać tych zmian.

Na planowanie dostępu do plików nie nałożono prawie żadnych ograniczeń. Gdy VENUS otrzymuje plik od VICE załącza obietnicę zawiadomienia (***callback promise***) o zmiannie w pliku. Obietnica jest przechowywana w VENUS razem z plikiem. Obietnica ma dwa stany : **WAŻNY** i **NIEWAŻNY**.

Gdy serwer otrzyma nowa wersje pliku, wysyła zawiadomienie każdemu z klientów, którzy posiadali *callback promise*. *Cache Manager* ponownie ściąga aktualna wersje pliku wraz z nowym callback promise. Na wszelki wypadek co określony okres czasu (stała systemowa) dochodzi do sprawdzenia aktualnego stanu pliku (Mechanizm ten wprowadzono w celu zachowania spójności pliku, który to mogł zostać zdezaktualizowany z powodu awarii sieci).

Opisany mechanizm nie sprawdzi się w przypadku, gdy dany plik będzie modyfikowany przez kilku użytkowników jednocześnie. Bez dodatkowego oprogramowania użytkownik nie zostanie poinformowany o kolejnych modyfikacjach. Zatem zapamięta zostanie wersja pliku, która została zapisana ostatnia.

1.8 Awarie

1.8.1 Klienta

- awaria klienta uniemożliwia dostęp do AFS
- po ponownym uruchomieniu wszystkie pliki z *Cache'a* są odnawiane (mogło dojść do utraty komunikatów callback promise)

1.8.2 Serwer

- są niedostępne tomy znajdujące się na serwerze
- pliki replikowane są nadal dostępne (za wyjątkiem plików przeznaczonych do replikowanych zapisu)
- pliki niereplikowane i bez kopii zapasowych sak chwilowo niedostępne

1.9 Bezpieczeństwo

Po podaniu hasła użytkownik jest identyfikowany. VENUS otrzymuje stosowny identyfikator uwierzytelniajacy użytkownika. Każda wymiana informacji jest poprzedzona wzajemna identyfikacja serwera i klienta. Hasło klienta jest wykorzystywane do zakodowania wiadomości.

2 Coda

2.1 Wprowadzenie

System plików Coda jest spadkobiercą systemu AFS, spełniający dodatkową funkcjonalność. Projekt Cody został zrealizowany w CMU pod przewodnictwem Satyanarayanana. Wymagania dotyczące Cody zostały oparte na doświadczeniach uzyskanych dzięki eksploatacji AFSu.

1. zawężona forma zwielokrotnień (tylko tomy do czytania)

2. niezadawalająca odporność na awarie lub planowane odłączanie serwerów

3. nieuwzglednienie użytkowania komputerów przenośnych

Te trzy wymagania określa się czasem *constant data availability*.

Coda zachowuje:

- skalowalność AFSu

- emulacje semantyki UNIXa

2.2 Zwielokrotnienie

2.2.1 Wstęp

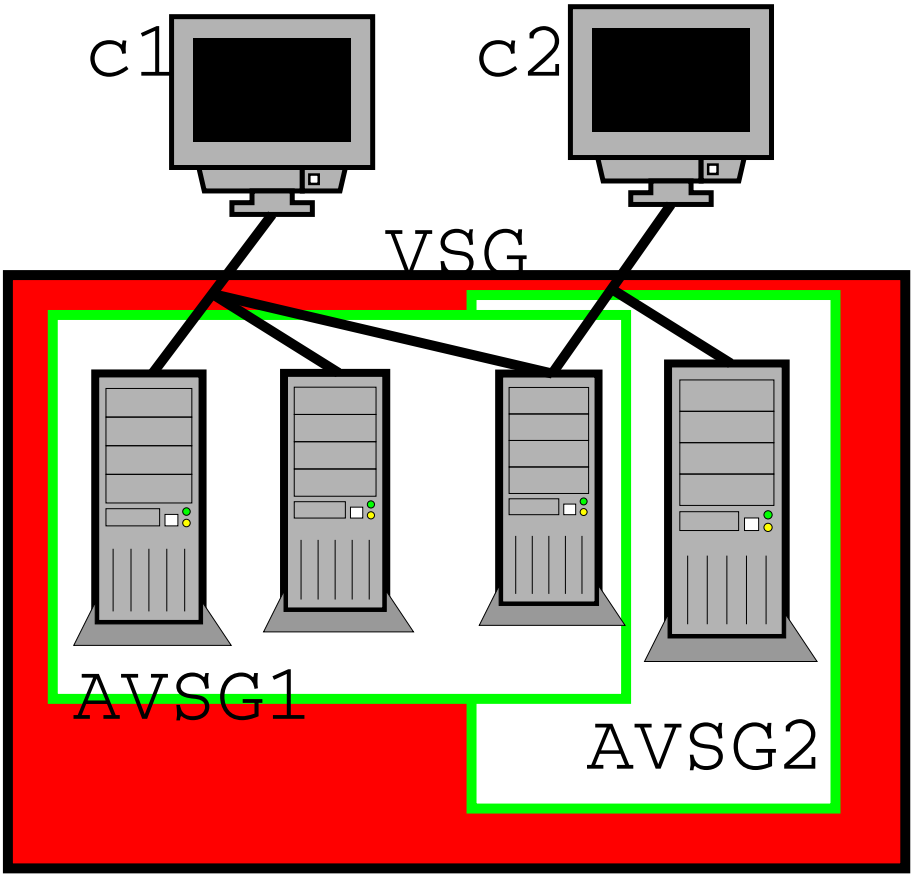
W przeciwieństwie do AFSu, projekt Coda zakłada zwielokrotnienie tomów plików w celu:

1. osiągniecia większej przepustowości

2. wyższego stopnia tolerowania uszkodzeń

Ponadto Coda korzysta ze specjalnego mechanizmu umożliwiającym stacjom roboczym działanie po odłączeniu od sieci (dzięki kopii plików w stacjach roboczych)

2.2.2 implementacja



Zbiór serwerów przechowujących kopie tomu plików nazywa się grupa tomów pamięci (ang. **volume storage group**, w skrócie **VSG**). Klient, który chce otworzyć plik może uzyskać dostęp do pewnego podzbioru vsg, nazywanego grupa dostępnych tomów pamięci (ang. **available volume storage group**, w skrócie **AVSG**). Skład grupy AVSG zmienia się w zależności od dostępności serwerów lub ich eliminowania wskutek awarii ich samych lub sieci.

Dostęp do plików jest podobny do AFSu. Podręczne kopie plików dostarcza stacjom roboczym klientów ktorys z serwerów grupy ASVG. Klienci sak powiadomiani za pomocą obietnic zawiadomień. Podczas operacji *close* kopie zmienionych plików są rozgłaszane równolegle w całej grupie AVSG.

Jeżeli grupa dostępnych tomów staje się pusta to następuje odłączenie (ang. *disconnected operation*). Może to wynikać z powodu:

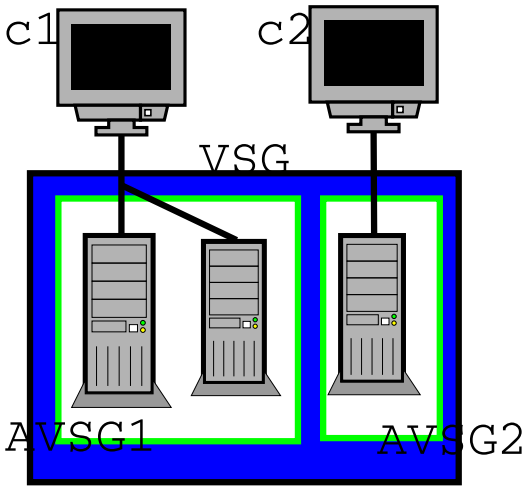
1. umyślnego odłączenia klienta
2. awarii

Jeżeli klient posiada wszelkie potrzebne mu pliki w pamięci stacji roboczej to może dalej kontynuować prace odłączony od systemu. Istnieje narzędzie zapisujące chronologię użycia plików w czasie pracy w podłączeniu do sieci. Tworzy wykaz daje podstawę do okreslania potrzeb na czas działania w czasie odłączenia.

Kopie plików pozostające w pamięci podręcznych są zdatne do użytku o ile przywraca się im co pewien okres czasu ważność. W przypadku działań odłączonych integracja danych nastąpi dopiero po kolejnym podłączeniu do sieci. W najgorszym przypadku może to wymagać ręcznej interwencji, aby pozbyć się niespójności i konfliktów.

CVV (ang. **Coda version vector**) składa się z liczb całkowitych odpowiadającym ilościom aktualizacji tomu na każdym z serwerów.

2.2.3 Przykład



Rozważmy ciąg zmian wykonywanych na pliku F w tomie zwielokrotnionym na trzech serwerach: S_1, S_2, S_3. Grup VSG dla pliku F jest zbiorem S_1, S_2, S_3. Plik jest zmieniany przez dwóch klientów C_1, C_2, ale z powodu awarii klient C1 ma dostęp do serwerów S_1, S_2 (dla C_1 AVSG= S_1, S_2 , zaś klient C2 dostęp do serwerów S_3 (dla C_2 AVSG=S_3).

1. CVV dla wszystkich serwerów jest równe [1,1,1]
2. C_1 otwiera plik F, zmienia jego zawartość i zapisuje. Wektory dla serwerów S_1, S_2 ulegaja zmianie na [2,2,1]
3. C_2 tworzy dwa procesy otwierające F, modyfikujące do i zapisujące. Wektor CVV dla S_3 zmienia się na [1,1,3]
4. po przywróceniu działania sesji, dochodzimy do konfliktu, gdyż wektory [2,2,1][2,2,1][1,1,3] nie można „jednocześnie” posortować po każdej zmiennej. Konflikt trzeba rozwiązać ręcznie.

Jeżeli nie nastąpi punkt 3 to [1,1,1] $\leq$ [2,2,1] $\leq$ [2,2,1] , czyli możemy uzgodnić nowa wersje bez potrzeby ręcznych zmian. Wersja pliku F z serwera S_1 lub S_2 zostanie skopiowana do serwera S_3.

Jeżeli nie da się posortować plików po każdej zmiennej jednocześnie, to plik ten zostaje odznaczony jako „nieoperatywny”, a właściciela informuje się o powstałym konflikcie. W przypadku katalogów możemy usunąć sprzeczności automatycznie (jedyne operacje to wstawianie i usuwanie wpisów katalogowych).

Gdy następuje zamkniecie pliku proces VENUS posyła do każdego elementu ze zbioru AVSG komunikat uaktualniajacy wraz z jego nowym CVV. O ile zmieniono zawartość pliku VICE zwraca odpowiedz pozytywna. Proces VENUS zbiera informacje od serwerów, które odpowiedziały na komunikat pozytywnie. VENUS wysyła stosowne wektor AVSG dla każdego serwera.

2.2.4 Podsumowanie

Zalety zwielokrotnień:

- zwiększona odporność na awarie, wystarczy do pobrania danych jeden działający serwer zawierajacy dany tom.
- obciążenia sieciowe wynikające z zamówień klientów można podzielić na wszystkie serwery, zwiększając wydajność systemu

2.3 Problemy z aktualizacja

Oto gwarancje jakości w systemie Coda:

- **Otwarcie pliku oznacza jedna z możliwości**:
  1. AVSG niepuste, plik aktualny
  2. AVSG niepuste, plik przestarzały co najwyżej o stalą systemowa i jest w *Cache'u*, przez ostatnie sekundy stracono zawiadomienia
  3. AVSG niepuste, ale plik jest w *Cache'u*
- **Nieudane otwarcie pliku oznacza**
  1. AVSG niepuste, ale są konflikty
  2. AVSG puste i pliku nie ma w *Cache'u*
- **Udane zamkniecie**
  1. AVSG niepuste i plik zaktualizowano
  2. AVSG jest puste
- **Nieudane zamkniecie**
  1. AVSG niepuste i wykryto konflikt

Aby zrealizować te gwarancje VENUS musi wykrywać następujące zdarzenie w czasie najdalej T sekund od ich wystąpienia (T stalą systemowa):

- powiększenie grupy AVSG (wskutek udostępnienia poprzednio niedostępnego serwera)
- zmniejszenie grupy AVSG (z powodu wycofania uprzednio dostępnego serwera)

- utracenie zawiadomienia

W razie sytuacji wyjątkowych powodujących niespójność danych na lokalnej pamięci podręcznej, VENUS przyjmuje założenie pesymistyczne i unieważnia obietnice zawiadomienia do wszystkich plików określonego tomu (mimo ze cześć plików może być aktualna).

W Codzie istnieją specjalne pliki lub partycjach (RVM), które są odpowiednikiem „bazy danych lokalizacji wolumenów" z AFSu. Przekazywane są tutaj wszystkie potrzebne dane o transakcjach nie ma możliwości częściowego lub błędnego modyfikowania bazy.

2.4 Wydajność

Satyanarayanan i in. porównują wydajność systemu Coda i AFS za pomocą testowych obciążeń w obecności od 5 do 50 użytkowników. Zauważono:

- Coda bez zwielokrotnień i AFS działają ze zbliżoną wydajnoscia
- Coda z 3 zwielokrotnieniami dla 5 użytkowników działa o ok 5% mniej wydolnie
- Coda z 3 zwielokrotnieniami i 50 użytkownikami spowodował zwiększenie czasu wykonania o odpowiednio 70% i 16% (w stosunku do 5 użytkowników).

Różnice te tylko częściowo można przypisać kosztom związanym ze zwielokrotnieniami za część różnic w działaniu odpowiada tez odmienne zestrojenie implementacyjne.

3 Podsumowanie AFS i Coda

NFS	AFS	Coda
• pierwszy oferowany handlowo produkt • dobre działanie dla systemów średniej wielkości • elastyczność przy montowaniu	• dobrze skalowalny • spójny • chwilowa utrata uaktualnień • zwielokrotnienia tomów do czytania	• działa dobrze dla stacji przenośnych • zwielokrotnienia • zwiększona odpornosc na błędy
• brak ogolnie rozumianego zwielokrotnienia • koszty administrowania • ograniczona skalowalność • problemy wydajnościowe	• ale juz nie do pisania • problemy z awariami • nie dla baz danych	• trochę ręcznej roboty w przypadku konfliktów • juz nie tak szybki