Rozproszone systemy plików z rodziny AFS

Filip Kalinski

filon@pld.org.pl

Grudzień 2002

Streszczenie

W dokumencie tym zostanie przedstawiony opis rozproszonych systemów plików z rodziny AFS. Należą do niej systemy takie jak AFS-2, AFS-3, Coda i DFS.

Opisany zostanie schemat działania wymienionych systemów plików a także sposób w jaki przeprowadzane są poszczególne operacje na plikach. Pokazane zostaną także różnice pomiędzy poszczególnymi systemami plików typu AFS i porównanie ze standardowym sieciowym systemem plików NFS.

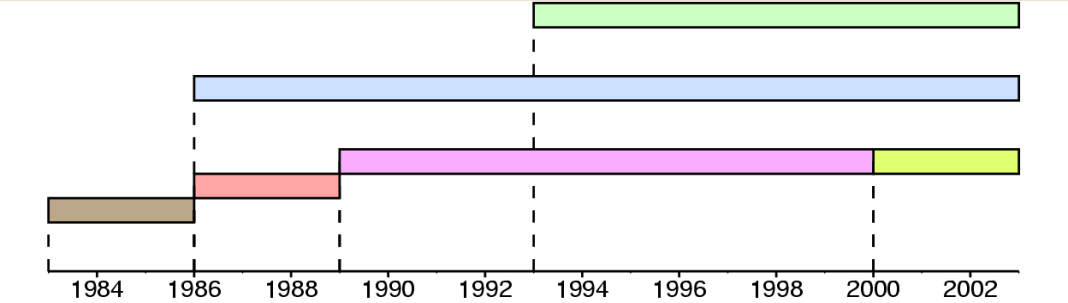
Spis treści

1. Historia

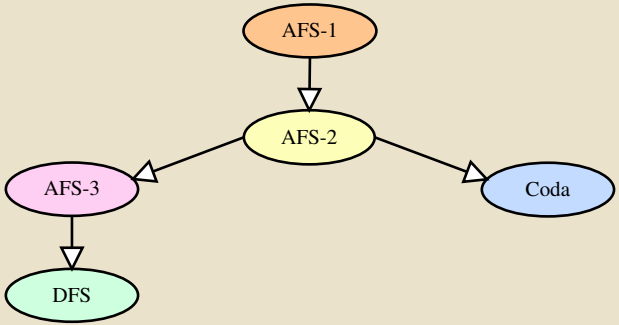
W roku 1983 w Carnegie Mellon University (CMU) w ramach projektu „Andrew” rozpoczęto pracę nad nowoczesnym rozproszonym systemem plików. Od nazwy projektu badawczego wywodzi się rozwinięcie skrótu AFS – „Andrew FileSystem”. Początkowo rozwijany system AFS-1, który został wydany po roku od rozpoczęcia prac W CMU, był jedynie przymiarką przed zaprojektowaniem systemu nadającego się do użycia w praktyce. Zatem, w roku 1986 powstał będący rozwinięciem początkowego projektu – AFS-2. Niedługo po nim zapoczątkowano w CMU pracę nad systemem Coda, który bazując na AFS-2 wprowadził swoje rozszerzenia funkcjonalności. Z AFS-2 wywodził się także jego następca, system AFS-3, który w 1989 został skomercjalizowany przez założoną w tym celu firmę „Transarc Corporation”. Transarc w roku 1998 wykupiony został przez IBM, który dwa lata później wydał AFS-3 z nazwą zmianioną na OpenAFS dostępny „OpenSource”.

W chwili dzisiejszej dostępne i aktywnie rozwijane implementacje AFS to IBM OpenAFS (serwer oraz klient), Arla (implementacje klienta AFS rozwijana w Kungla Tekniska Högskolan) oraz RedHat AFS (klient AFS w jądrze Linuksa, włączony do gałęzi 2.5, na razie nie obsługuje operacji zapisu, etc. . .).

Następujące rysunki ilustrują historię systemów z rodziny AFS i zależności między nimi:



Rysunek 1: Historia



Rysunek 2: Zależności

2. Założenia i konsekwencje

2.1. Cele i założenia

Przy projektowaniu systemu AFS skorzystano ze spostrzeżeń dotyczących najczęstszych schematów użycia plików (badania M. Satyanarayanana z CMU).

Wynikało z nich, że:

- większość plików jest mała i wczytywana w całości
- większość plików używana jest przez tylko jednego użytkownika
- operacje odczytu są znacznie częstsze niż operacje zapisu (średnio w proporcjach 6:1)
- najczęściej dostęp do plików jest seryjny, tzn. w krótkim przeciągu czasu ten sam plik jest używany wielokrotnie

Jako główne cele postawiono sobie:

- wyskoką niezawodość
- bezpieczeństwo systemu
- szybkość działania
- obsługę bardzo dużej liczby użytkowników (przynajmniej 5000)
- przezroczystość dla użytkownika

2.2. Konsekwencje

Z powyższych założeń wynikają podstawowe cechy systemu AFS:

- **Serwowanie plików w całości**
  Zmniejsza liczbę odwołań do serwera, a także umożliwia lepsze wykorzystanie łącz.

  W AFS-3 zmieniono podejście na serwowanie plików w dużych kawałkach (and. *chunk*). Domyślną wielkością kawałka jest 64KB a możliwe jest skonfigurowanie tej wartości aż do 256KB. Katalogi nadal serwowane są w całości.

- **Buforowanie**
  Wszystkie otwierane pliki i katalogi są przechowywane u klienta na dysku. Kolejne operacje odczytu nie wymagają ponownego pobierania pliku z serwera, wykonywany jest jedynie szybki odczyt z lokalnego systemu plików. Jedyną stuacją w której konieczne jest ponowne pobranie pliku jest informacja z serwera o jego modyfikacji. Operacje zapisu także nie przesyłają danych na serwer, dopiero zamknięcie pliku powoduje wysłanie go z powrotem (z wyjątkiem zmian w katalogach, które są przysłane natychmiast).

  Dzięki temu oszczędzany jest nie tylko serwer (co umożliwia mu przyłączenie większej ilości klientów) ale także łącza sieciowe. Jest to kluczowa cecha AFS zapewniająca wysoką skalowalność sytemu.

  Katalogi oraz pliki są buforowane tak jak pobierane – w całości w AFS-2 i Coda a w dużych kwałkach w AFS-3 i DFS.

- **Mechanizm zapewnienia spójności**
  Serwer plików AFS pliku pamięta listę plików otwartych u każdego z klientów. Umożliwia to stosowanie mechanizmu *callbacków*.

  Polega to na tym, że w razie modyfikacji pliku serwer informuje klienta o zmianie, po czym klient pobiera zmodyfikowany plik i uaktualnia go buforze. Zapewnienie spójności danych w przypadku, gdy klient był rozłączony w trakcie rozsyłania informacji o zmianie gwarantuje odpytywanie serwera o aktualność każdego buforowanego pliku za każdym razem po nawiązaniu połączenia.

  Własność pamiętania stanu umożliwia też stosowanie blokad na pliki.

- **Niezawodność**
  W celu zapewnienia niezawodności usługi możliwa jest transparentna dla użytkownika replikacja serwerów. Zapewnia to nieprzerwane działanie systemu a także dystrybuowanie ruchu pomiędzy wiele serwerów.

  W AFS-3 możliwa jest jedynie replikacja plików tylko do odczytu, w Coda i DFS możliwe jest także replikowanie zasobów modyfikowalnych.

- **Bezpieczeństwo**
  Przyjęty został zaawansowany model autentykacji użytkowników polegający na dwustronnym uwierzytelnianiu oparty na protokole Kerberos w wersji IV lub V.

  Drugą własnością mającą służyć zwiększeniu bezpieczeństwa jest autoryzacja poprzez ACL (*acces control list* - listy kontroli dostępu) na poziomie

katalogów. Przemyślane wykorzystanie ACL zarówno zwiększa bezpieczeństwo, jak i ułatwia zarządzanie prawami dostępu. Dodatkowo, użytkownik sam może tworzyć grupy co pozwala na łatwe zarządzanie dostępem do własnych plików.

W systemie DFS zniesiono ograniczenie użycia ACL do katalogów, każdy plik ma w nim własną listę ACL.

- **Globalna przestrzeń nazw**

  Dodatkowo ciekawą cechą AFS (oraz DFS, ale już nie Cody) jest jednolita przestrzeń nazw plików w skali globalnej. Każdy plik ma swoją lokalizację w drzewie katalogów /afs niezależną od tego do jakiego serwera klient się połączył. Zapewnia to przezroczystość dla użytkownika, nie musi on wiedzieć z jakim serwerem się łączy, ani na jakim znajduje się potrzebny mu plik.

3. Implementacja

Podstawową jednostką logiczną w AFS jest komórka (ang. *cell*). Składa się ona ze zbioru serwerów oraz ich klientów. Zbiór plików udostępnianych przez serwery tworzy drzewo katalogów dostępne w tej samej postaci dla każdej maszyny należącej do komórki. U każdego klienta zamontowany system plików AFS znajduje się w podkatalogu `/afs` katalogu głównego lokalnego systemu systemu plików.

Ponadto, możliwe jest wyeksportowanie plików dostępnych w danej komórce do globalnej sktruktury katalogów, tak, że dla każdego klienta AFS, niezależnie od tego z jakiej komórki się połączył pliki te będą dostępne w tym samym miejscu w strukturze katalogów. Powszechnie stosowanym schematem jest umieszczanie katalogów z dajne komórki w katalogu `/afs/<nazwa komórki>`, gdzie nazwa komórki zwykle jest nazwą domeny DNS w której komórka się znajduje, np. `/afs/mimuw.edu.pl`. Lista wszystkich komórek AFS znaduje się w pliku konfiguracyjnym `CellServDB`.

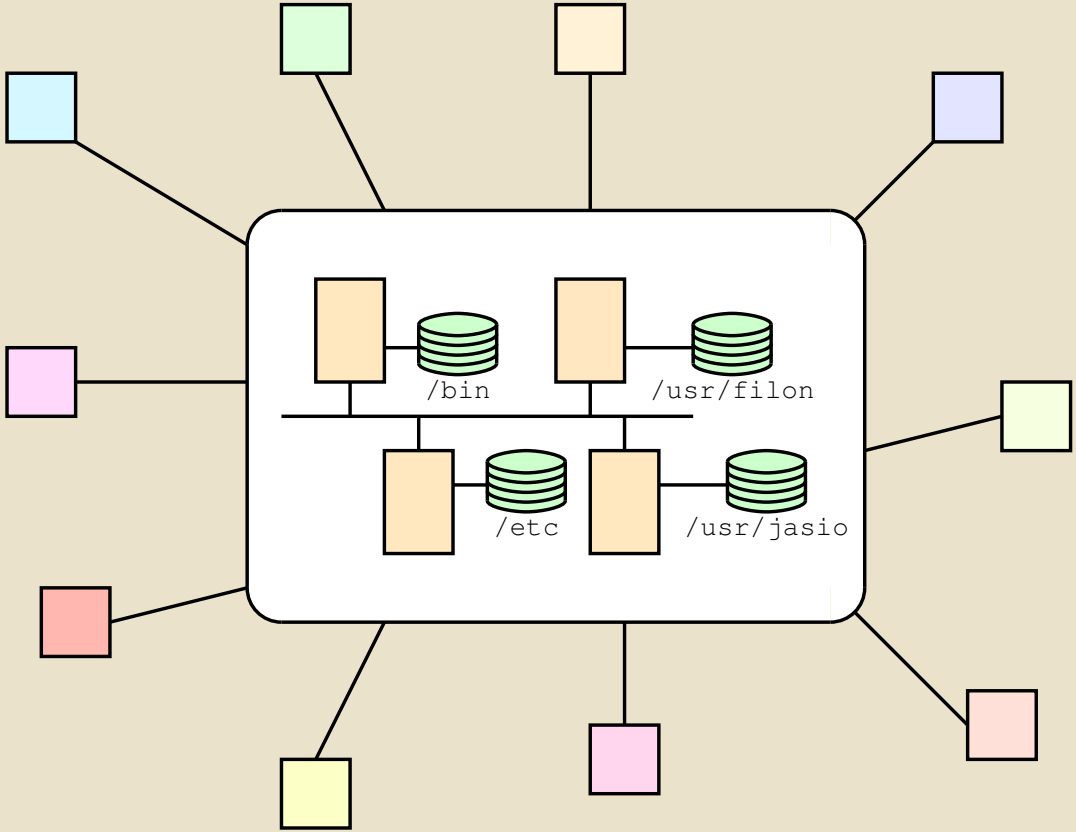
Pliki w AFS grupowane są na mniejsze jednostki zwane wolumenami (ang. *volume*). Wolumen jest zbiorem plików udostępnianych przez serwer (będącym poddrzewem jego drzewa katalogów) i dostępnym dla klienta jako wpis w katalogu danej komórki.

Podział ten jest dla użytkownika przezroczysty, nie musi on wiedzieć, który serwer udostępnia który wolumen, co więcej, wolumen można przenieść na inny serwer w sposób dla klienta niezauważalny. Dodatkowo wolumen może być replikowany, czyli istnieć w kilku kopiach na różnych serwerach. W razie awarii jednego z nich zawartość katalogu odpowiadającemu wolumenowi będzie ser-

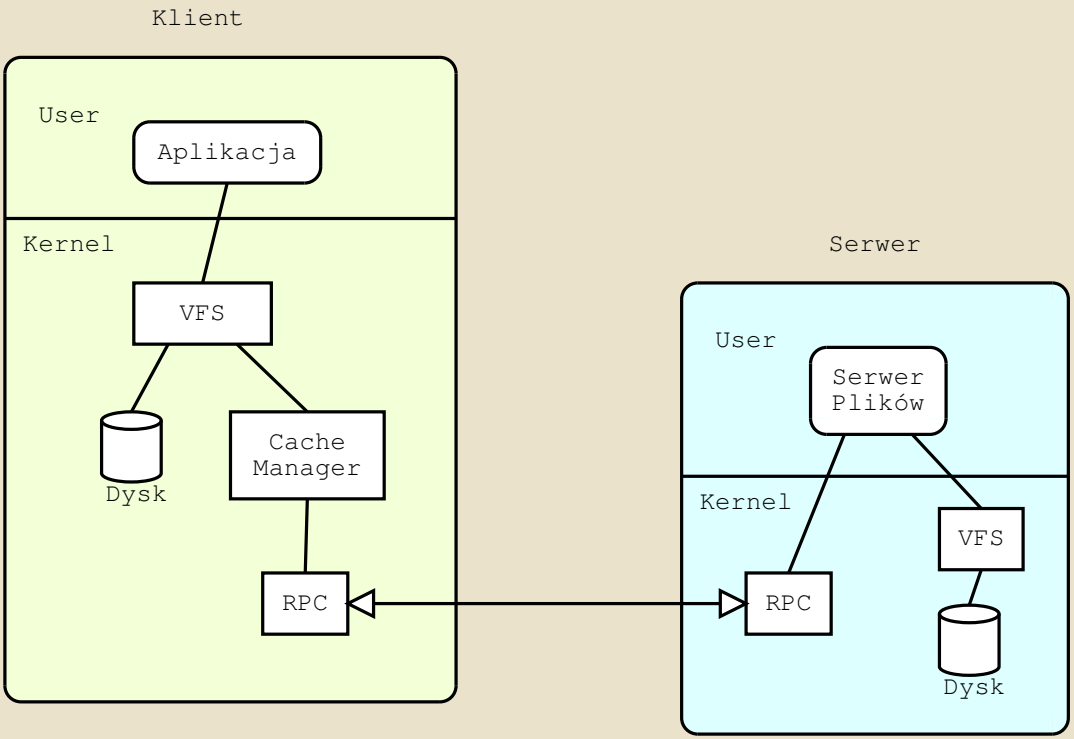
wowana przez pozostałe.

W systemie AFS replikacja nie jest jednak automatyczna. Administrator musi wydać polecenie auktualnianie kopii wolumenu. Tworzona w ten sposób kopia służy jedynie do odczytu, w ten sposób żądania odczytu mogą być rozdzielane pomiędzy kopie. Żądania zapisu nadal przysłane są do głównej kopii wolumenu.

Następujący rysunek przedstawia ogólny schemat komórki AFS:



Komunikacja między klientem a serwerami odbywa się za pomocą protokołu opartego na RPC. Ziluostrować można ją następująco:



3.1. Serwer

W komórce AFS działa wiele serwerów (aplikacji) zajmujących się obsługą usługi serwowania plików. Dla celów wydajności, lub zapewnienia większej niezawodności często działają one na różnych maszynach.

3.1.1. BOS

Zapewnieniem nieprzerwanego działania wszystkich serwerów zajmuje się BOS (*Basic Overseer Server*). Uruchamia on ponownie serwery które „padły" i zachowuje ich plik *core*, a także dba o to, aby wykluczające się serwery nie działały razem.

3.1.2. Ubik

Ubik jest rozproszoną bazą danych. Udostępnia on operacje zapisu i odczytu na rozproszonym pliku. Podstawową cechą Ubika jest transakcyjność, to znaczy zapewnienie, że zmiany dokonywane są tylko w całości. Udostępnia on też mechanizmy synchronizacji stanu bazy danych pomiędzy wieloma działającymi serwerami Ubik przechowującymi jej kopię.

Ubik jest wykorzystywany przez wszystkie pozostałe serwery do trzymania swoich danych.

3.1.3. Zarządzanie wolumenami

Program vldb-server odpowiedzialny jest za udostępnianie informacji o powiązaniach nazw wolumenów z serwerami które je przechowują.

3.1.4. Autoryzacja

Program *ka-server* zajmuje się autentykacją użytkowników. Jest to serwer Kerberos IV zmodyfikowany do wykorzystywania RPC jako mechanizmu po-

bierania biletów i administracji.

Z kolei *ptserver* przechowuje listę użytkowników oraz grup i udostępnia dane innym serwerom.

3.1.5. Serwowanie plików

Serwer plików *fs* serwuje dane dla klientów, pamięta informacje o połączeniach, zarządza callbackami i rozsyła informację o modyfikacji plików.

Volser jest ściśle związany z serwerm plików i pozwala na dodawanie, usuwanie, przenoszenie i modyfikowanie wolumenów. Każda zmiana w serwerze wolumenów jest odpowiednio synchronizowana z pracą serwera plików, tak żeby nie serwował właśnie modyfikowanego wolumneu.

3.1.6. Pozostałe

Serwer backupu (*bu-server*) przeznaczony jest do zarządzania kopiami zapasowymi wolumenów. Kopia zapasowa wolumenu znajduje się na tej samej co on maszynie i tworzona jest codziennie. Umożliwia to użytkownikom odzyskiwania przypadkowo utraconych plików bez ingerencji adminiatratora.

Serwer uaktualniania przeznaczony jest do automatycznego uaktualniania plików konfiguracyjnych oraz programów wykonywalnych na wszystkich serwerach w komórce. Przechowuje on główną kopię tych plików i umożliwia pozostałym serwerom synchronizację z nimi.

3.2. Klient

Po stronie klienta obsługą systemu plików AFS zajmuje się *cache manager*. (zarządca pamięci podręcznej). Jego odpowiedzialnością jest komunikacja z serwerem, buforowanie pobranych plików i zapewnienie integralności i aktualności pamięci podręcznej plików. Buforowane pliki przechowywane są dysku u klienta (zwykle na wydzielonej partycji). Jeśli brakuje miesjca w pamięci podręcznej to zwalniane jest według metody LRU.

Zarządca pamięci podręcznej plików zaimplementowany jest jako moduł jądra systemu operacyjnego i komunikuje się z użytkownikiem poprzez warstwę VFS.

Komunikacja klienta z serwerem wymaga autentykacji. Procedura autentykacji wykonywana jest przy logowaniu się do serwera. Składa się ona z dwóch procedur.

W pierwszej, służącej do zweryfikowania tożsamości klienta i serwera, klient wysyła zaszyfrowaną jego kluczem wiadomość. Serwer deszyfruje wiadomość i odsyła odpowiedź zaszyfrowaną jego kluczem. Jeśli klient potrafi rozszyfrować odpowiedź jest pewny, że serwer jest oryginalny. Z kolei serwer roszyfrowując informację od klienta sprawdził, że klient jest tym za kogo się podaje.

Druga procedura obejmuje trzy strony: klienta, serwer i serwer bilecików. Klient za pomocą pierwszej procedury autentykuje się u serwera bilecików. Ten tworzy nowy klucz, właściwy tylko dla tego połączenia – klucz sesji, za pomocą którego szyfrowane są dalej wiadomości pomiędzy nimi. Przesyła on klientowi „bilecik” zawierający zaszyfrowaną wiadomość dla serwera, zaszyforwaną jego (serwera) kluczem, oraz klucz sesji. Zaszyfrowany klucz sesji zawarty jest także

w bileciku dla serwera.

Przy każdej potrzebie komunikacji z serwerem klient wysyła do niego polecenie zaszyfrowane kluczem sesji razem z bilecikiem. Serwer roszyfrowuje swoim kluczem bilecik, wyciąga z niego klucz sesji, a potem rozszyfrowuje nim wiadomość. Jeśli wiadomość ma sens, serwer uznaje tożsamość klienta i odsyła mu odpowiedź szyfrując ją kluczem sesji.

Klient odbierając wiadomość od serwera odszyfrowuje ją i jeśli ma ona sens przyjmuje ją uznając tożsamość serwera.

Dzięki procedurze wzajemnej autentykacji między klientem a serwerem obydwie strony mają pewność, że nikt się nie podszywa pod drugą z nich, a poza tym, jako że transmisja jest szyfrowana nikt inny nie może przejąć danych.

Opisze teraz krótko schemat działania poszczególnych operacji na plikach (przyjmując pobieranie i buforowanie plików w całości, tak jak w AFS-2 i Coda).

- *open* Operacja open s systemie AFS przebiega następująco:
  1. jeśli żądany plik znajduje się w pamięci podręcznej AFS, jest on otwierany jeśli pozwalają na to prawa dostępu
  2. jeśli nie, do wysyłanie jest żądanie otwarcia pliku do serwera odpwiadającego komórce w której plik się znajduje
  3. jeśli użytownik ma dostęp do pliku, serwer go wsysła, plik zostaje przechowany w pamięci podręcznej i deskryptor zostaje przekazany użytkownikowi
- *read, write, lseek* Operacje read, write oraz lseek operują na kopii pliku znajdującej sie na lokalnym dysku, w pamięci podręcznej AFS. Konsek-

wencją takiej semantyki operacji *write* jest możliwość utracenia zmian w przpadku awarii klienta przed wykonaniem operacji *close*.

- *close, fsync* Jeśli aplikacja wywoła funkcję *close*, zarządca pamięci podręcznej plików wykonuje synchroniczny zapis danych na serwerze przechowującym centralną kopię zapisywanego pliku. Kontrola wraca do aplikacji dopiero o potwierdzeniu przyjęcia danych.

  Przy wykonaniu operacji fsync kontrola wraca do programu dopiero po potwierdzeniu zapisania danych przez serwer na urządzenie pamięci masowej.

4. Pozostałe systemy

4.1. Coda

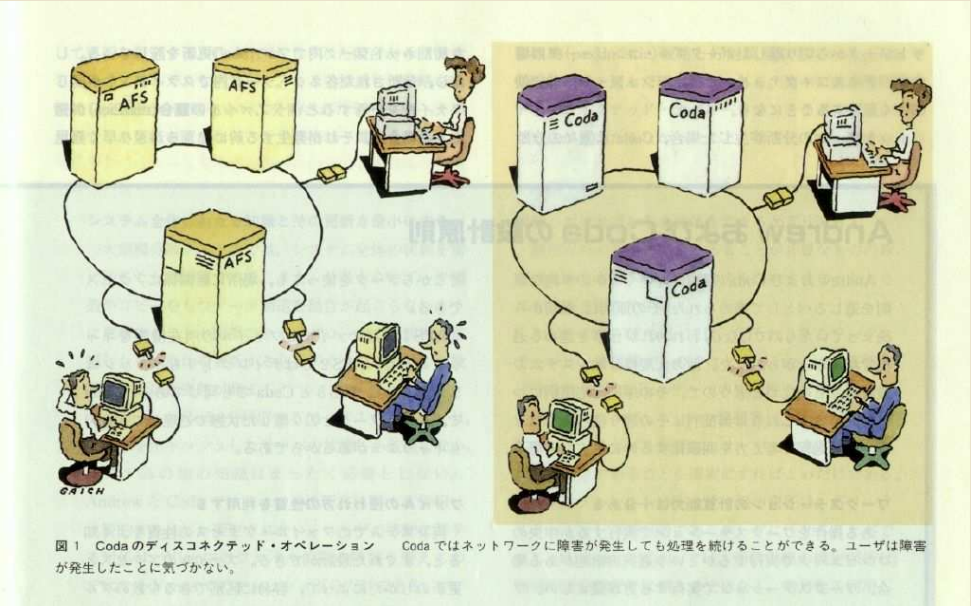
Podstawowe różnice pomiędzy systemem Coda a AFS w wersji 3 to:

- Replikacja wolumenów modyfikowalnych
  W systemie Coda możliwe jest replikowanie także wolumenów modyfikowalnych. Zadaniem klienta jest wprowadzanie zmian na wszystkich dostępnych serwerach. Jeśli dany serwer był niedostepny w czasie modyfkacji, to przy starcie dokonywana jest próba uzgodnienia zawartości pliku.

- Wsparcie dla pracy rozłączanej
  Jeśli połączenie z serwerem zostanie zerwane, pliki znajdujące się w pamięci podręcznej mogą być dalej używane. Przy następnym połączeniu z serwerem system spróbuje uaktualnić pliki na serwerze korzystając z listy zmian. Przyjęte zostało optymistyczne założenie, że współdzielony dostęp r/w do plików jest rzadki. Użytkownik może dodatkowo podać listę plików, które chce, aby zawsze były dostępne w pamięci podręcznej.

- Serwowanie i buforowanie całych plików
  Jest to pozostałość z systemu AFS-2. W systemie Coda pozostawiono ją ze względu na konieczność dostępu do całego pliku w przypadku rozłączenia od serwera.

- Grupowanie operacji
  W systemie Coda operacje zamknięcia pliku są asynchroniczne i dane



Rysunek 3: Coda

mogą być wysyłane na serwer w grupach, w celu lepszego wykorzytsania połączenia.

- Nie ma globalnej przestrzeni nazw plików

Wsparcie dla pracy rozłączanej dorze ilustruje poniższa grafika:

4.2. DFS

System DCE/DFS powstał jako następca systemu AFS-3. W stosunku do AFS-3 zawiera liczne rozszerzenia. Najważniejsze z nich to:

- Listy ACL dla wszytkich plików
  W DFS listy ACL związane są ze wszystkimi plikami, nie tylko z katalogami.
- Blokowanie na poziomie bajtów
  W systemie AFS możliwe było blokowanie dostępu (*locking*) do całych plików. W DFS możliwe jest ustanawianie blokad na poszczególen bajty.
- Replikacja
  W DFS możliwa jest automatyczna replikacja read-only.

5. AFS a NFS

Alternatywnym wobec AFS, bardzo szeroko rozpowszechnionym sieciowym systemem plików jest NFS firmy Sun Corporation.

System AFS jest zaprojetkowany w sposób znacząco różniący się wobec NFS. Główne cechy różniące AFS od NFSv3 to:

- Agresywane cacheowanie
- Pamięć stanu (co za tym idzie, blokady i callbacki)
- Replikacja

AFS charaketeryzuje się lepszym wykorzystaniem łącz sieciowych oraz serwerów, możliwa jest praca znacząco większej liczy klientów połączonych z serwerem. Większe obowiązki spoczywają jednak na kliencie, musi on przechowywać bardzo duże ilości danych. Z tego powodu nie nadaje się tak dobrze jak NFS dla mszyn bezdyskowych.