Rozproszony system plików - CODA

Piotr Miłoś

12 stycznia 2003

1 Wprowadzenie

1.1 Wstęp

Coda - to rozproszony system plików rozwijany przez grupę M. Satyanarayanana na Carnegie Mellon University. Jest to rozwinięcie innego rozproszonego systemu plików AFS. Głównym celem projektantów było zwiększenie niezawodności systemu w stosunku do AFS. A także zwiększenie wydajności i poprawienia dostępu do plików dzielonych. System zaprojektowano dodatkowo z myślą o urządzeniach przenośnych o których wiadomo, że będą odłączane od systemu, tak żeby zapewnić dalsza w miarę bezproblemową pracę.

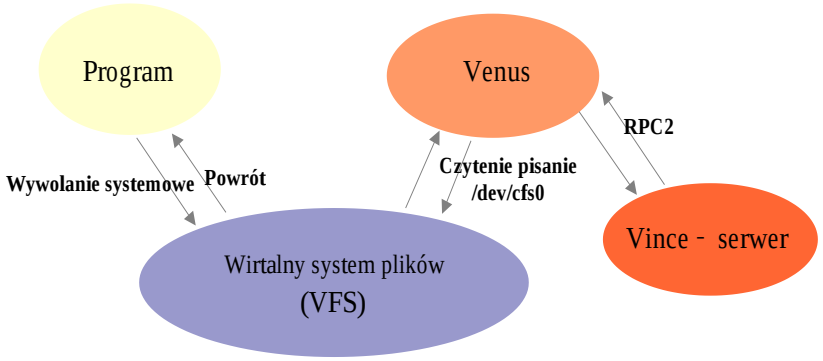
1.2 Podstawowe cechy Cody

- wsparcie dla komputerów przenośnych
  - synchronizacja danych z klientów odłączonych
  - adaptacja szybkości połączenia
  - tryb odłączenia
- szybki powrót po awariach
  - serwery replikujące do czytania i pisania
  - działanie po awrii rozdzielającej serwery
  - obsługa rozłączania klientów
- wydajność i skalowalność
  - stała pamięc podręczna klienta dla plików, katalogów i atrybutów
  - pamięć podręczna zapisywania
- bezpieczeństwo
  - autentykacja za pomocą protokołu podobnego do Kerberosa

  - listy kontroli dostępu (ACL)
  - szyfrowanie
- dostępność kodu źródłowego
- wspomaganie backupów - poprzez zastosowanie maszyny koordynatora backupu

2 Architektura i działanie systemu

W systememi Coda zastosowano architekturę klient serwer. Klient nazywa się Venus, a serwer Vice. Komunikacja między składowymi odbywa się za pomocą protokołu RPC2. Sytuację przedstawia poniższy rysunek.



- Venus - zarząda plikami i pamięcią podręczną na komputrze klienta, kieruje żądania do serwerów
- Vice - zarządza plikami i metadanymi na serwerze

Vice i Venus są zaimplementowne w większości jako zwykłe procesy. W przypadku Linuksa potrzebne są moduły do jądra, ale ich funkcjonalność w większości polega na przekazaniu żądań do procesów użytkownika lub też do zwykłego odczytu z dysku.

W systemie Coda wyróżnione są trzy typy maszyn: orócz klientów, i zwykłych serwerów, występują jeszcze maszyny kontroli systemu (System Control Machine - SCM).

Klienci to zwykle jednostanowiskowe stacje robocze. Serwery nie ufają klientom, odmawiają dostępu do danych, jeśli kelient nie jest w czasie sesji po zalogowaniu użytkownika. Udostępnione dane dostępne w podmontowane są w drzewie katalogów (najczęściej korzeniem tego systemu plików jest katalog /coda).

Serwery są zaufanymi, bezpiecznymi maszynami, które mają dostarczyć dzielone dane użytkownikowi.

Ostatni składnik systemu to SCM (maszyna kontroli dostępu). Jest on punktem kontroli całego systemu. Logicznie SCM nie jest serwerem, choć fizycznie może działać na maszynie, która jest serwerem.

2.1 Serwer - podstawy działania

Serwer systemu Coda przechowuję dzielone dane na lokalnym systemie plików serwera (najczęściej w katalogu /vicepa).

Oprócz tych danych system musi przechowywać pewne metadane potrzebne do prawidłowego działania serwera. Ilość metadanych jest większa niż w większości rozporoszonych systemów plików. Metadane obejmują informacje o właścicielach, listy praw dostępu, wektory wersji i informacje o katalogach (w przeciwieństwie do standardowego zachowania Uniksa, gdzie katalogi to zwykłe pliki z listą zawartych w nich plików), informacje o powielonych obiektach i inne.

Do przechowywania metadanych zaprojektowano specjalny system RVM (recoverable virtual memory) i plik logu. RVM jest systemem transakcyjnym. Operacje, które zostały zatwierdzone, ale jeszcze nie włączone do pliku z metadanymi są przechowywane w logu transakcji. Takie rozwiązanie zapewnia duży poziom bezpieczeństwa i zapewnia szybkie odtworzenie po awarii. Sam RVM i log najczęściej są przechowywane na odzielnej partycji nie obsługiwanej przez żaden system plików, jest to lepsze rozwiązanie niż przechowywanie tych danych w pliku, ze względu na lepszą spójność danych i wydajność.

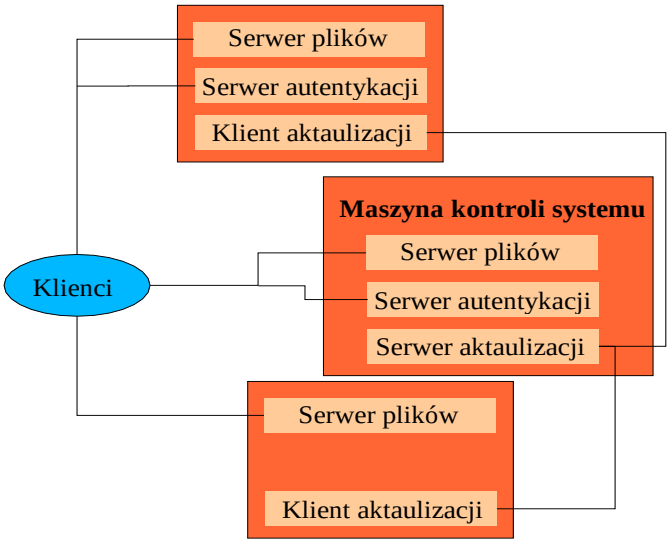
2.2 Serwer - procesy

Poniżej są wypisane i omówione procesy na serwerze i SCM (w ścisłej terminologii systemu Coda SCM nie jest serwerem, ale można patrzyć na niego jak na serwer, jeśli potraktujemy system jako przykład architektury klient-serwer).

- proces serwera plików - proces, z którym bepośrednio łączy się stacja klienta, odpowiedzialny za dostarczenie danych
- proces serwera autentykacji - uruchomiony na wszystkich serwerach proces, który przydziela użytkownikom identyfikatory sesji (po podaniu hasła). Hasła mogą być zmieniane tylko na SCM, na serwerach dostęp do nich jest tylko do czytania. Serwer autentykacji obsługuje też wzajemne uwierzytelnianie serwerów.

- proces uaktualniania - współpracuje z procesem uaktualniania na SCM, jest odpowiedzialny za utrzymywanie aktualnych kopi danych systemowych (np. hasła). Aktualność danych jest sprawdzana co pewnien czas, więc występują pewne opóźnienia.
- proces serwera uaktaulanianie - proces działający na SCM, odpowiedzialny za dostarczenie do serwerów

Przykładowy system Coda może wyglądać tak jak na poniższym obrazku:



2.3 Komórki

Najwyższą komórką organizaji systemu Coda jest komórka. Komórka złożona jest z grupy serwerów mających tą samą bazę konfiguracji. W komórce jest jeden wyróżniony serwer SCM (maszyna kontroli systemu), który jako jedyny może zmieniać konfigurację. Jego zadaniem jest rozprowadzanie zmian konfiguracji inne serwery. W obrębie komórki, każdy serwer ma unikalny numer z zakresu 1-126 i 128-254. (Numer -1, 0 są zarezerwowane do obsługi sytuacji błednych, a numer 128 do identyfikacji "powielonych tomów"). Adresy serwerów i ich identyfikatory są przechowywane na SCM. Aby dodać nowy serwer do komórki należy zrobić odpowiedni wpis na serwerze (w pliku /vice/db/servers).

W chwili obecnej klient Cody może należeć tylko do jednej komórki.

2.4 Tomy

Podstawową jednostką (z punktu widzenia serwera) danych w systemie Coda to tom (volume). Na partycjach przeznaczonych dla Cody, przechowywane są

pliki zgrupowane w tomy. W pojedynczym tomie zawarte jest jedno drzewo i zachowana jest normalna struktura katalogów. Jeden tom musi być w całości zapisany na jednej patycji. Tomy często związane są z logicznym zarządzaniem systemem plików (np. tom zawierający katalogi użytkowników). Dzięki temu są wygodną i łatwą do zarządzania jednostką z punktu widzenia administratora systemu (np. na poziomie tomu ustalane są limity wykorzystania dysku).

W typowych zastosowaniach serwer zawiera kilkaset-kilka tysiecy tomów.

Tom może być zamotowany w dowolnym podkatalogu innego tomu. Montowanie odbywa się na serwerze (za pomocą specjalnych narzędzi) i jest w ten sposób tworzone drzewo tomów, które przekłada się bezpośrednio na drzewo katalogów widzianych przez klienta.

Samo motowanie ma troche inny przebieg niż standardowe motowanie systemów plików, procedura montowania stwarza katalog, w którym ma być moutowany tom (w przeciwieństwie do Uniksa, gdzie katalog jest musi istnieć wcześniej), w przypadku, gdy katalog już istnieje motowanie zakończy się błędem. Katalog stworzony w ten sposób jest przez klienta widziany jak każdy inny katalog. W odróżnieniu od Unixa, takie montowanie jest trwałe, czyli pozostaje po zrestartowaniu systemu.

Jeden z tomów jest oznaczony jako korzeń systemu plików i po zmotowaniu u klienta on jest widziany bezpośrednio w katalogu /coda.

Każdy tom ma nazwę i przypisany numer VolumeId, jednoznacznie go identyfikujący w obrębie komórki.

Serwery systemu plików mogą powielać dane. Zwiększa to bezpieczeństwo (gdy padnie jeden serwer dane mogą być pobrane z innego) i wydajność, ale powoduje dodatkowe problemy ze spójnością danych. Powielanie danych odbywa się na poziomie tomów. Z każdym z tomów związana jest lista serwerów na których znajdują się kopie danego tomu (VSG - Volume Storage Group). Każdy klient przechowuje dla danego tomu listę listę dostępnych serwerów z listy VSG, czyli listę AVSG (Available Storage Group).

Komunikacja serwerów z klientem odbywa się na następującej zasadzie. Przy otwieraniu pliku sprawdzane są wersje na serwerach z listy AVSG. Mogą zdażyć się dwa przypadki:

- dane, są spójne - być może część serwerów przechowuje nieaktualne dane, ale system może automatycznie rozpoznać, które serwery przechowują najnowszą wersję, spośród tych serwerów wybierany jest jeden (np. najszybszy) z którego pobierane są dane. Jeśli były serwery z nieaktualnymi danymi to inicjowana jest operacja uaktualnienia.

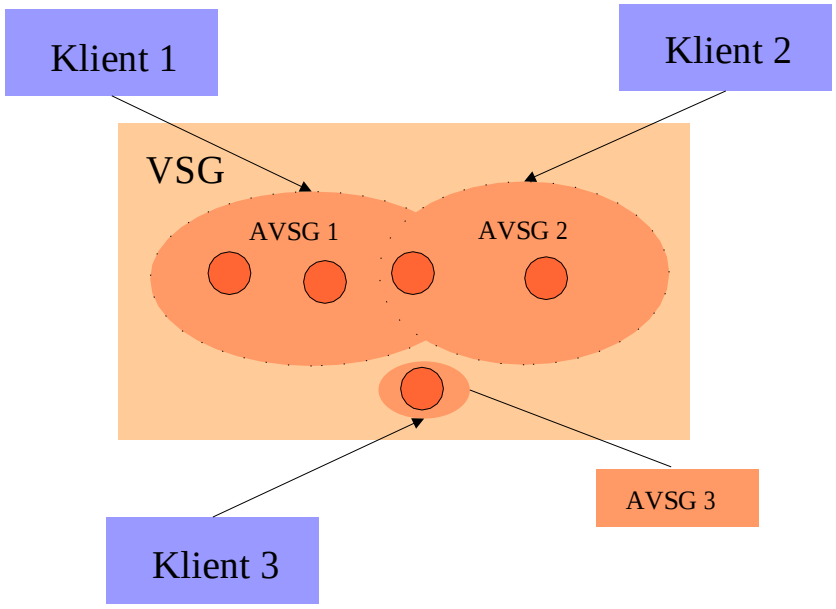
- dane nie są spójne - wśród wersji przechowywanych przez serwery z listy AVSG występuje konflikt, który nie może być rozwiązany. Otwarcie pliku kończy się niepowodzeniem. Odkryty problem jest zgłaszany użytkownikowi i do dalszej pracy potrzebna jest jego interwencja lub interwencja administratora.

Informacje o wszystkich tomach w danej komórce przechowywane są na SCM.

Typy tomów i działania na nich:

Typ tomu	Gdzie można zapisywać	Gdzie można odczytywać	Występowar
nie powielony	na serwerze odpowiedzialnym za tom	na serwerze odpowiedzilnym za tom	r
powielony	na serwerach z VSG	na serwerach z VSG	t
backup	nigdzie	na serwerze odpowiedzialnym za tom	r

Przykładowy schemat dostępu do tomu (klient 1 ma dostęp do serwerów z grupy AVSG1, a klient 2 do serwerów z grupy AVSG2, a klient tylko do jednego serwera).



Na tym rysunku szczególnie łatwo zauważyć możlwość powstania konfliktu wersji. Jeśli nawet założyć, że żaden konflikt nie powstanie w wyniku działania "klienta 1" i "klienta 2", to łatwo zauważyć, że "klient 3" nie widzi co się dzieje w serwerach grupy AVSG1 i AVSG2. Jeśli zarówno "klient 3" i np. "klient 2" zmienią te same pliki, a potem w wyniku zmiany warunków sieciowych nastąpi powiększenie grup AVSG x do całego VSG, to przy kolejnym dostępie zostanie wykryty konflik.

Łatwo zauważyć, że możliwość powstania konfliktu jest konsekwencją nierówności AVSG i VSG. Jest to świadomy krok projektantów, mający na celu zwiększenie dostępu (na powyższym rysunku klient 3 nie miałby prawdopodbnie dostępu do danych).

2.5 Przestrzeń nazw

Wszystkie pliki udostępniane przez Codę mają w obrębie jednej komórki jednoznaczną nazwę, niezależnie od komputera i użytkownika, który się łączy. Ujmując to abstrakcyjnie, można powiedzieć, że użytkownik przyłącza sie do sytemu Coda, a nie do konkretnego serwera. Jeśli system się zmieni (np. dołączone zostaną nowe tomy lub serwer), zmiany w przestrzeni nazw są automatycznie przenoszone do klienta.

Nazwa ta nie wskazuje w żaden sposób, gdzie są umiejscowione pliki, co więcej pliki mogą znajdować się w na wielu serwerach. (tak więc zrealizowana jest przezroczystość położenia i przezroczystość zwielokrotnienia).

Wewnątrz systemu plik jest reprezentowany przez trójkę liczb: pierwsza to identyfikator tomu, druga to identyfikator pliku wewnątrz tomu, a trzecia to dodatkowa informacja przechowywana przez system (np. jeśli tom jest powielony).

2.6 Podstawowy schemat działania

Po przekazaniu przez jądro żądania otwarcia pliku do Venus po raz pierwszy, jest on w całości pobierany z serwera i zapisywany na lokalnym dysku (najczęściej w katalogu /usr/coda/venus.cache). Następnie większość operacji dostępu do pliku odbywa się bez odwoływania do Venus, są one obsługiwane przez lokalny system plików. Venus zarządza również przechowywaniem w pamieci podrecznej katalogów i atrybutów plików. Po zamknięcu pliku, jeśli jego zawartość została zmieniona, zostaje on wysłany na serwer.

Taki schemat działania opiera sie na obserwacji, że pliki znacznie częściej są otwierane tylko do czytania niż do zapisywania, większość plików jest mała i bez problemu może być przechowywana w pamięci podręcznej i odwołania do pliku są skumulowane (tj. jeśli użytkownik raz skorzystał z pliku to prawdopodobnie zrobi to ponownie).

Wyróżnione są dwa tryby pracy:

- podstawowy opisany wyżej, w przypadku zmiany pliku i jego zamknięcia informacja o tym jest natychmiast wysyłana na serwer, w przypadku, gdy serwer jest niedostępny (np. z powodu awarii sieci), funkcja systemowa zwraca błąd
- tryb odłączony, w tym przypadku system zakłada, że komputer jest odłączony od sieci (np. komputer przenośny pracownika) i brak połączenia z serwerem jest czymś normalnym. Zamiast wysyłać zmodyfikowany plik na serwer jest on zapisywany w "logu modyfikacji klienta" (Client Modification Log). Po połączeniu wszystkie pliki zapisane w CML są synchronizowane z serwerem.

2.7 Schemat działania pamięci podręcznej

W celu zwiększenia wydajności w systemie Coda stosuje się rozbudowany system pamięci podręcznej.

System ten jest zarządzany przez zarządcę pamięci podręcznej (Cache Manager). Z chwili, gdy na stacji klienckiej otwierany jest plik, znajdujący się w systemie Coda system sprawdza, czy aktualna kopia znajduje się już w pamięci. Jeśli nie to pobierany jest cały plik z jednego z serwerów na liście AVSG tomu do którego należy plik i umieszczany w pamięci podręcznej. Od tej chwili operacje na pliku przeprowadzane są lokanie (np. następne otwarcie tego samego pliku będzie lokalne). Następny kontakt z systemem następuje w chwili zamykania pliku. Jeśli został on zmodyfikowany, to jest rozyłany do wszystkich serwerów z AVSG. Zarządca pamięci podręcznej przechowuje pliki do czasu, gdy miejsce zajmowane przez ten plik będzie potrzebne dla nowych napływających plików. Taki sposób działania wynika z założen twórców systemu, że odwołania do pliku są skumulowane.

Jasne jest, że do popawnego działania takiego systemu potrzebne jest dodatkowy mechanizm sprawdzania, czy plik przechowywany w pamięci podręcznej jest poprawny. Można to zrealizować na dwa sposoby, w niekórych systemach plików, przy każdym otwarciu sprawdzana jest aktualność, w Codzie, podobnie jak w AFSie zastosowano mechanizm powiadamiania (callback). Dla plików, które mogą być modyfikowane przez użytkownika, załącza się obietnicę powiadamiania (callback promise). Zarządca pamięci podręcznej traktuje dany plik jako aktualny dopóki nie otrzyma zawiadomienia o jego zmianie. Jeden z serwerów z AVSG wysyła zawiadomienia (breaks callback) po tym jak otrzyma od jakiegoś z klientów zmieniony plik. Przy następnej próbie otwarcia zarządca pamięci podręczniej ponownie pobiera plik z serwera.

Obietnice zawiadomienia muszą być przechowywane na serwerze, dla zabezpieczenia tych danych w razie awrii, stosuje się RVM.

Co ustalony okres czasu aktualność jest sprawdzana na serwerach, aby zminimalizować szkody wywołane przez zaginięcie zawiadomienia.

W systemi Coda prowadzono też eksperymetny polegające na załączaniu obietnicy potwierdzeń, nie tylko do plików, ale w pewnych sytuacjach także do tomów. Takie zwiększenie poziomu na którym są przyznawane obietnice ma na celu zmniejszenie ilości informacji potrzebnych do utrzymania spójności pamięci podręcznej.

System Coda nie zapewnia kontroli współbieżności aktualizacji. W przypadku, gdy wielu użytkowników edytuje plik, a potem go zamykają, to jako aktualna zostanie uznana wersja ostatniego użtykownika.

2.8 Wersje plików

Do utrzymania informacji o wersji wersji stosuje się wektor wersji (CVV - Coda Version Vector). Składa się on z wersji pliku na wszytkich serwerach VSG. Numer wersji jest zwiększany przy każdym udanym (bezkonfliktowym) zamknięciu pliku. Tak więc zaraz po zamknięciu dany plik ma ten sam CVV na wszystkich serwerach ze zbioru AVSG. Oprócz CVV przechowuje się również znacznik czasu.

W przypadku pobierania danego pliku sprawdzany są te wektory. Jeśli jakiś wetor dominuje nad pozostałymi (to znaczy, że na na każdej pozycji numer

wersji jest większy lub równy), to uznajemy, że ten serwer przechowuje aktualną wersję pliku. W przypadku, gdy nie można znaleźć takiego wektora, błąd musi być rozwiązany ręcznie przez użytkownika mającego dostęp do pliku lub administratora.

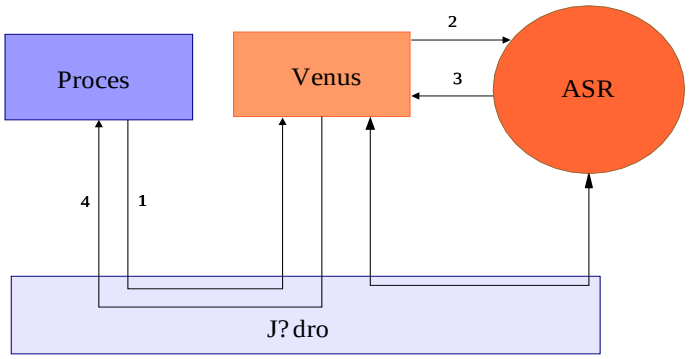
2.8.1 Automatyczne rozwiązywanie konfliktów

Konflikty pojawiające się podczas pracy systemu Coda mogą być rozwiązane automatycznie. Jest to możliwe dzięki zastosowaniu ASR (Application Specyfic Resolver). ASR jest programem, który zna format pliku i w razie wystąpienia konfliktu, w niektórych przypadkach, jest w stanie go rozwiązać.

Przykładem jest program kalendarz, który pozwala wpisywać planowane na dany dzień czynność. Można sobie wyobrazić, że dane całego tygodnia są trzymane w jednym pliku. Jeśli dwoje ludzi zrobiło wpisy na różne dni, to ASR, może to automatycznie zintegrować. Jeśli wpisy były na ten sam dzień to nie jest to możliwe i następuje normalny scenariusz wymagający interwencji użytkownika.

Cechy ASR:

- dla bezpieczeństwa uruchamiany jest na kliencie
- pliki konfiguracyjne są przechowywane na maszynach klienckich i mają łatwy do edytowania format
- w celu poprawego działa ASR, musi na pewnien czas mieć udostępnione różne wersje pliku z różnych serwerów (chwilowa utrata przezroczystości położenia).
- operacja poprawiania, wykonywana jest atomowo



2.9 Praca w trybie odłączonym

Istotną cechą Cody jest wsparcie dla pracy z urządzeniami, które mogą być odłączone od systemu (np. laptopy). Ten tryb pracy nazywa się odłączonym.

W przypadku pracy w trybie odłączonym, ważny jest też dostęp do plików. Najprostszy schemat opierałby się na zachowaniu tylko tych plików, które są w pamięci podręcznej podczas odłączania. Nie byłoby to dobre, gdyż z dużym prawdopodobieństwem użytkownikowi brakowałoby jakiś plików do pracy. Coda rozwiązuje ten problem tworząc listę najczęściej używanych plików (lista ta jest konstruowana automatycznie, ale istnieje też możliwość ręcznego dodawania plików do niej. Udostępniane są też narzędzie, pozwalające łatwo wykryć z jakich udostępnianych plików korzysta dany program "spy"). Pliki z tej listy są regularnie pobierane z serwera, tak że w pamięci podręcznej znajdują się "swieże wersje plików".

Praca w trybie odłączonym powoduje też pewne problemy, zwiększa prawdopodobieństwo wystąpienia konfliktów (bo może się zdarzyć, że dwóch użytkowników pracujących na tych samych danych przez dłuższy czas nie ma żadnego kontaktu). Sposoby rozwiązywania tych konfliktów są takie same jak w przypadku normalnej pracy.

2.10 Sematyka aktualizacji

W systemie Coda są następujące gwarancje aktualności

Po udanym otwarciu wiemy, że jeśli AVSG jest niepuste to plik jest aktualny lub jest nieaktalny co najwyżej o T sekund, przez które był umieszczony w

pamieci podrecznej i starcono zawiadomienia. Jeśli AVSG jest puste to wiadomo tylko, że plik jest w pamięci podręcznej.

W przypadku nieudanego otwarcia są możliwe dwa przypadku, AVSG nie jest puste, ale wystąpiły konflikty, AVSG jest puste i pliku nie ma w pamieci podręcznej.

W przypadku nieudanego zamknięcia wiemy, że AVSG jest niepuste i wtedy plik jest zaktalizowany, lub AVSG jest puste.

W przypadku nieudanego zamknięcia wiadomo, że AVSG jest niepuste, ale wsytąpiły konflikty.

Aby zapewnić działanie powyższego schematu, dla każdego pliku w pamięci podręcznej, klient co T sekund wysyła komunikat próbny do serwerów VSG. Dostaje odpowiedź tylko od serwerów AVSG łącznie z wektorem CVV. W przypadku wykrycia niezgodności aunulowana jest obietnica zawiadomienia dotycząca pliku. Dzięki temu okresowemu odpytywaniu klient wykrywa także zmiany AVSG.

2.11 Bezpieczeństwo

2.11.1 Autentykacja klienta

Podstawowy schemat autentykacji wygląda następująco:

- klient przekazuje swoje hasło
- serwer autentykacji przekazuje klucz sesji
- klucz jest używany przez Venus do ustanowienia bezpiecznego połączenia

Klucz sesji jest przekazany w postaci dwóch żetonów (token), jeden zawiera klucz sesji, czas rozpoczęcia, czas zakończenia i identyfikator użytkownika. Drugi żeton zawiera ponadto pole z wartością losową i jest zaszyfrowany kluczem znanym klientowi.

System Cody umożliwia zastosowanie do autentykacji Kerberosa.

2.11.2 Prawa dostępu do pliku

W chwili dostępu do pliku sprawdzane są uprawnienia do pliku, na podstawie identyfikatrora użytkownika. System uprawnień opiera się na listach konroli dostępu. Jeśli klient nie poda żetonu ze swymi danymi, zakładane jest, że jest to użytkownik anonimowy.

Prawa dostępu przyznawane są grupą użytkowników. Wpis w liście kontroli dostępu odwzorowuje członka domeny zabezpieczeń na zbiór praw. Dla danego użytkownika prawa do danego obiektu są określone przez prawa wszystkich grup do których bezpośrednio lub pośrednio należy.

Coda oferuje rozszerzony (w stosunku do Linuksa) system praw dostępu:

- r - prawo czytania

- l - prawo do czytania katalogu
- i - prawo do tworzenia plików i podkatalogów w danym katalogu
- d - prawo do usuwania plików i katalogów
- w - prawo do pisania do plików w katalogu
- k - flaga utrzymywana z powodów historycznych
- a - prawo zmiany list dostępu

Prawa mogą być pozytywne i negatywne

2.11.3 Szyfrowanie

Komunikacja między klientami i serwerami odbywa się za pomocą protokołu RPC2. Oferuje on następujące poziomy zabezpieczeń:

- wszystkie dane są przesyłane bez szyfrowania
- szyfrowana jest tylko autentykacja
- szyfrowana jest autentykacja i nagłówki procedur RPC
- wszystkie dane są szyfrowane

2.12 Brak dostępu do tomu.

W pewnych okolicznościach maszyna kliencka może nie mieć dostępu do pewnego tomu.

Mogą wystąpić dwie sytuacje:

- zupełny brak dostępu do tomu - podczas synchronizacji danych (użytkownik po rozłączeniu synchronizuje swe dane z serwerem, synchronizacja nie jest zakończona dopóki są jakieś konflikty); brak dostępu (odłączenie klienta następuje jeśli serwer uznaje za błędną, operację uznaną przez klienta za poprawną); brak połączenia (jeśli klient nie uzyskuje szybkiej odpowiedzi na swoje zapytanie o tom, oznacza że dany serwer nie działa, konsekwencją jest oczywiście brak dostępu do tomu;
- tom jest odłączony do pisania - sytuacja ta jest określana również jako słabo połączony. Tom jest oznaczany przez klienta jako słabo połączony, kiedy znacząco spada szybkość połączenia z serwerem. W tym trybie zapisywanie tomów jest przekładane na później, żeby zmniejszyć obciążenie sieci. Operacje czytania działają w normalny sposób. Odłączenie może nastąpić automatycznie, jeśli przesył z serwera spadnie poniżej pewnego poziomu (obecnie 50 kb/s) lub też użytkownik może sam zażądać przejścia w ten tryb (mając na celu zwiększenie wydajności, kosztem ryzyka powstania konfliktów).

3 Coda a cechy rozproszonych systemów plików

Koda realizuje (w mniejszym lub większym stopniu) następujące cechy rozproszonych systemów plików:

- przezroczystość położenia - jest realizowana całkowicie, użytkownik nie posiada wiedzy, gdzie fizycznie znajduje się plik na którym pracuje. Co wiecej plik może znajdować się naprawdę w wielu miejscach.
- przezroczystość dostępu - w przybliżeniu realizowana jest przezroczystość dostępu. Odstępstwa polegają na tym, że użytkownik musi czasem ręcznie rozwiązywać konflikty, ponadto nie jest zrealizowana unixowa sematyka operacji na plikach. Zmiany w plikach nie są widziane przez innych dopóki program piszący nie zamknie pliku (sematyka sesji). W obrębie sesji i stacji roboczej zapewniona jest uniksowa sematyka dostępu do pliku.
- przezroczystość awarii - system Coda jest stosunkowo odporny na awarie, zwielokrotnienie danych zapewnia dalsze działanie systemu mimo, że jego pewne części nie działają. W razie poważniejszej awarii (np. dysków) dzieki zwielokrotnieniu możliwe jest odtworzenie danych. Dodatkowo bezpieczeństwo poprawione jest przez zastosowanie transakcyjnego sytemu RVM.
- przezroczystość wydajności - system Coda zapewnia wygodną pracę nawet w przypadku obciążenia sieci, dzięki kopiowaniu całych plików i wykonywaniu większości operacji na lokalnym dysku. Także na poprawę wydajności całego systemu wpływa zastosowanie odłączania tomów do pisania.
- przezroczystość sprzętu i systemu operacyjnego - system Coda jest może działać na wielu platformach (komercyjne Uniksy, Linux, systemy Windows). Co więcej dostępny jest kod źródłowy co dodatkowo zwiększa przenośność.
- skalowalność - w założeniach projektowych Cody znalazła się skalowalność, dzięki możliwości powielania danych może z nich korzystać wiele stacji roboczych.
- serwer Cody jest serwerem stanowym - zaletą serwerów bezstanowych jest łatwość przywracania do pracy systemów po awarii. W przypadku serwerów stanowych jest to trudniejsze, gdyż serwery przechowywują w pamięci dane, które mogły zostać zniszczone. W przypadku Cody problem ten jest złagodzony przez zastosowanie systemu transakcyjnego RVM.

4 Bibliografia

- G. Coulouris. J. Dollimore. T. Kindberg *Systemy rozproszone podstawy i projekowanie*

- A. Silberschatz. P. Galvin *Podstawy systemów operacyjnych (ogólne informacje na temat rozproszonych systemów plików i pokrewnego do Cody AFS'a).*
- Strona domowa Cody: *http://www.coda.cs.cmu.edu/*