Wprowadzenie do rozproszonego systemu plików Coda

Jacek Pasternak

9 stycznia 2003

1 Wprowadzenie

Coda jest rozproszonym systemem plików opartym na architekturze klient/serwer, wywodzącym się z AFS v. 2. Rozwijany jest w Carnegie Mellon University od 1987 r. Podobnie jak inne rozproszone systemy plików, głównym zadaniem Cody jest udostępnianie grupom komputerów klienckich plików, które są widziane jako część ich drzewa katalogów, ale przede wszystkim utrzymywanie autorytatywnej kopii plików na serwerach. Korzenie AFSu widać w Codzie poprzez wsparcie dla stałego buforowania po stronie klienta, co pomaga zminimalizować czas restartu po stronie klienta i odpowiedzi serwera, dostarczając efektywnej skalowalności. Coda dostarcza wsparcia dla operacji bezpołączeniowych dla komputerów przenośnych. Ta cecha wykorzystywana jest również podczas czasowych niedostępności pewnych serwerów. Zmiany są automatycznie nanoszone podczas ponownego połączenia. Coda potrafi dzielić serwery na grupy. Dane składowane i pobierane są z któregokolwiek serwera z grupy. Coda będzie kontynuować działanie, gdy tylko będzie dostępny jakiś podzbiór wszystkich serwerów. W przypadku powstania różnic z powodu podziału sieci Coda rozstrzygnie różnice automatycznie o możliwie największego zasięgu i pomoże użytkownikowi w naprawieniu tego, co nie może być zrobione automatycznie. Jako projekt akademicki źródła i binarie dla Cody są dostępne na licencji GPL.

2 Główne cechy Cody

Operacje bezpołączeniowe

- reintegracja danych od odłączonych klientów
- adaptacja do szybkości łączy

Radzenie sobie z problemami

- replikacje serwerów do odczytu/zapisu
- rozwiązywanie konfliktów serwer/serwer

- radzenie sobie awariami sieci dzielącymi serwery
- radzienie sobie z rozłanczaniem się klientów

Wydajność i skalowalność

- buforowanie po stronie klienta plików, katalogów i atrybutów
- „write back caching”

Bezpieczeństwo

- autentykacja
- listy ACL

Dobrze zdefiniowana semantyka współdzielenia

Dostępny za darmo kod źródłowy

3 Maszyny

Architektura Cody rozpoznaje 3 typy maszyn: *klienci*, *serwery* oraz *SCM* (*system control machine*). Maszyny klienckie są zwykle jednostanowiskowymi stacjami roboczymi używanymi do dostępu do informacji. Te maszyny nie są zaufane z wyjątkiem czasu trwania sesji logowania. Serwery są bezpiecznymi, zaufanymi maszynami, których celem jest realizowanie zleceń klientów dotyczących współdzielonych danych. Serwery muszą wymagać od każdego użytkownika uwierzytelnienia zanim wyślą mu jakiekolwiek współdzielone dane. Trzecim typem maszyny jest SCM. Jej celem jest udostępnienie wygodniej, „jednopunktowej” administracji systemem. Z logicznego punktu widzenia SCM różni się od innych serwerów, ale fizycznie SCM może również odgrywać rolę zwykłego serwera.

4 Organizacja systemu

4.1 „Coda cell”

Jest to grupa serwerów dzielących jeden zbiór baz danych, z których pobierana jest konfiguracja. Jedna komórka może zawierać od jednego do setek serwerów. Jeden serwer pełni rolę tzw. **SCM**, czyli maszyny kontrolującej system. Tylko on może modyfikować bazy danych, zawierające informacje o konfiguracji, które są dzielone przez wszystkie serwery i propagować takie zmiany do innych serwerów. Obecnie klient Cody może należeć tylko do jednej komórki.

4.2 Pojedyncza przestrzeń nazw

Cała Coda jest widoczna jako pojedynczy katalog `/coda` po stronie klienta (albo jako pojedynczy napęd w Windows). Coda nie posiada różnych exportów bądź zasobów dzielonych, jak robi to NFS czy Samba, które są osobno montowane. W wolumenie `/coda` widoczne są pliki eksportowane przez wszystkie serwery leżące w „Coda cell". Coda automatycznie znajduje serwery i wszystko co potrzebuje wiedzieć klient, to nazwa jednego z serwerów startowych, który udziela informacji jak znaleźć główny (root) wolumen. Z punktu widzenia użytkownika nie widać podziału na serwery. Główną zaletą pojedynczej przestrzeni nazw jest to, że wszyscy klienci mogą być tak samo skonfigurowani i widzieć takie samo drzewo katalogów. Jest to kluczowe zagadnienie dla dużych systemów.

4.3 Wolumeny

Serwery grupują pliki w wolumeny. Wolumen jest zwykle dużo mniejszy od partycji i dużo większy od katalogu. Jeden wolumen jest wyróżniony i jest nim wolumen główny (root volume),który jest montowany w `/coda`. Inne wolumeny znajdują się w podkatalogach `coda` tworząc poddrzewo. Wolumeny zawierają drzewo katalogów z plikami. Mogą także zawierać inne podmontowane wolumeny. Takie punkty montowania to nie to samo co Unixowe punkty montowania bądź napędy w Windows. Coda jest widziana jako jeden napęd (Windows), bądź jeden punkt montowania (Unix). Montowanie w Codzie jest stałe w przeciwieństwie do Unixa, gdzie trzeba je powtarzać za każdym razem, gdy system jest resetowany. Punkty montowania Cody zawierają wystarczająco informacji dla klienta, aby możliwe było znalezienie serwera, który przechowuje pliki zawarte w tym wolumenie. Nie ma więc potrzeby robienia wielokrotnego „look up" jak to ma miejsce w przypadku NFS. Grupa serwerów oferujących dany wolumen nazywana jest „*Volume Storage Group*" (VSG). Ze względu na to, że nie wszystkie serwery danej grupy mogą być dostępne, wyróżnia się tzw. Available VSG (AVSG).

System plików Coda udostępnia 4 różne typy wolumenów. Najprostszym z nich jest „*wolumen niereplikowalny*". Takie wolumeny znajdują się na pojedynczym serwerze i są pod opieką serwera, na którym się znajdują. Serwer współpracuje z procesami **Venus** na maszynach klienckich dając im spójny obraz systemu plików, jednakże kiedy serwer „powiesi się", bądź w jakiś inny sposób będzie niedostępny niereplikowalne wolumeny będą również niedostępne.

W celu częściowego rozwiązania tego problemu Coda udostępnia drugi typ wolumenów: „*replikowalne wolumeny tylko do odczytu*". Ten typ wolumenu ma dokładnie jedną kopię do czytania i pisania, ale może mieć wiele kopii „tylko do odczytu" kontrolowanych przez inne serwery. Zmiany takiego wolumenu dokonywane są na kopii „do odczytu i zapisu", a później dystrybuowane do wszystkich serwerów z kopiami „tylko do odczytu". Replikacja tylko do odczytu stwarza większą dostępność do wolumenów zawierających często pożądanych, ale rzadko uaktualnianych obiektów takich jak „binarie systemowe". Dodatkowo replikacja „tylko do odczytu" jest wykorzystywana przy wykonywaniu kopii zapasowych wolumenów.

Niestety repliki „tylko do odczytu" nie mogą dostarczyć wysokiej dostępności dla wszystkich typów wolumenów np. wolumenów użytkowników. Dlatego Coda udostępnia trzeci rodzaj wol-

umenów: „*replikowalne wolumeny do odczytu i zapisu*”. Takie wolumeny są logicznymi wolumenami, które grupują razem wiele niereplikowalnych wolumenów „do odczytu i zapisu”. Coda dostarcza protokół, który umożliwia replikowalnym wolumenom „do odczytu i zapisu” egzystować na wielu serwerach i być dostępnym nawet wtedy, gdy kilka serwerów jest niedostępnych. Mimo, że replikacja „do odczytu i zapisu” udostępnia wszystko, co replikacja „tylko do odczytu”, jej protokoły są droższe. Dlatego replikacja „do odczytu i zapisu” powinna być używana tylko wtedy, gdy wolumen jest często żądany i szybko się zmienia.

Czwarty typ to *kopie zapasowe*, które są wolumenami tylko do odczytu i wykorzystywanymi kiedy zajdzie taka potrzeba (np. gdy użytkownik skasuje sobie potrzebny plik). Różnica miedzy nimi, a replikowalnymi wolumenami tylko do odczytu jest taka, że kopie zapasowe z żaden sposób nie można zmieniać (jeden z replikowalnych wolumenów tylko do odczytu ma tryb „odczyt - zapis”) oraz są widoczne jako osobne wolumeny (replikowalne wolumeny są swoimi kopiami i wyglądają dla użytkownika jak jeden wolumen).

5 Serwery Cody

Głównym programem działającym na serwerze Cody jest **codasrv**. Jest on odpowiedzialny za wszystkie operacje na plikach, jak również za zarządzanie wolumenami.

Serwer uwierzytelnianie **auth2** przejmuje zapytania od **clog** o tokeny i zmianę hasła z **au** i **cpasswd**. Tylko proces auth2 na SCM może zmieniać bazę haseł.

Wszystkie serwery w komórce Cody dzielą bazę konfiguracji w `/vice/db` oraz otrzymują ją od SCM, gdy zaszły zmiany. Program **updateclnt** jest odpowiedzialny za otrzymywanie takich zmian oraz odpytuje **updatesrv** na SCM żeby zobaczyć, czy coś się zmieniło. Czasami SCM potrzebuje bazę danych z innego serwera aby uaktualnić dzieloną bazę. Robi to poprzez proces **updatesrv** na tym serwerze używając **updatefetch**.

5.1 Składowanie danych

Serwery nie składują, ani nie eksportują wolumenów jako katalogów w lokalnym systemie plików jak robi to NFS i Samba. Coda potrzebuje dużo więcej metadanych do wspierania replikacji serwerów i operacji bezpołączeniowych oraz ma skomplikowany sposób odzyskiwania danych, który jest trudny do przeprowadzenia na lokalnych systemach plików. Serwery Cody przechowują pliki identyfikowane poprzez numery, zwykle wszystkie są w katalogu `/vicepa`. Metadane (właściciele, listy kontroli dostępu, wektory wersji) i zawartość katalogów są przechowywane jako plik RVM, który często jest całą partycją dyskową. Informacje o katalogach, listach ACL i atrybutach plików trzymane są na osobnych partycjach w pliku RVM. Pliki identyfikowane są poprzez trojkę 32 bitowych wartości całkowitoliczbowy nazywaną „Fid” tzn.: „VolumeID”, „VnodeId” i „Uniquifer”. VolumeId identyfikuje wolumen, na którym znajduje się plik. VnodeId jest numerem i-węzla pliku, a Uniquifer używany jest dla zwiększenia rozdzielczości. Fid jest niepowtarzalny w obrębie klastra serwerów Coda.

5.2 RVM

Jest to skrót od *Recoverable Virtual Memory*. RVM to biblioteka oparta na transakcjach używana do uczynienia części wirtualnej przestrzeni adresowej procesu stałej na dysku oraz dokonywania zmian w tej pamięci atomowo w celu uzyskania niezmiennego składowania. Umożliwia utrzymanie danych zawartych w pamięci pomiędzy restartami systemu. Coda używa RVM do zarządzania metadanymi. Te dane są składowane w pliku RVM, który jest mapowany do pamięci podczas startu. Modyfikacje dokonywane w pamięci wirtualnej są również zapisywane do pliku RVM LOG podczas dokonywania transakcji plik LOG zawiera zatwierdzone dane, które nie zostały jeszcze przyłączone do pliku z danymi na dysku. Zwykle Plik RVM jak i LOG trzymane są na osobnych partycjach na dysku.

5.3 Typowa organizacja serwera

Serwery Coda wymagają przynajmniej 3 partycji dyskowych do uzyskania optymalnej wydajności, ochrony danych i zabezpieczenia przed przypadkowym skasowaniem (jedna na RVM LOG, jedna na RVM i jedna ze zwykłym UNIXowym systemem plików przechowującym dane składowane w Codzie). W celu zwiększenia wydajności dobrze jest, gdy RVM LOG jest na osobnym dysku w celu uniknięcia zbytniego ruchu głowic podczas logowanych operacji. Opcjonalnie na osobnej partycji może być `/vice` jak i `/var`. Kiedy potrzebnych jest więcej niż jedno miejsce przechowywania danych na serwerze Coda (domyślnie wszystkie dane składowane są w `/vicepa`) mogą zostać utworzone dodatkowe miejsca (np. `/vicepb`). Poniższa tabelka przedstawia przykładowy podział na partycje.

Partycja	**Cel użycia**	**Punkt montowania**	**Typowy rozmiar**
hda2	System plików użytkownika	`/`	650MB
hda5	System plików „Var”	`/var`	100MB
hda3	System plików „Vice” (tu znajdują się m.in. pliki systemowe)	`/vice`	300MB
hdc1	RVM Log	brak	12MB
sda1	Dane RVM	brak	130MB
sda2	Składowane dane 1	`/vicepa`	1.6GB
sda3	Składowane dane 2	`/vicepb`	1.6GB
sda5	Składowane dane 3	`/vicepc`	1.6GB

6 Klient

6.1 Dane klienta

Składowane są w podobny sposób jak na serwerze. Metadane znajdują się w RVM (zwykle `/usr/coda/DATA`), a zbuforowane pliki przechowywane są wg numerów w `/usr/coda/venus.cache`. Rozmiar pamięci podręczej po stronie klienta jest stały. Za-

wiera kopie plików z serwera. Pozwala na szybszy dostęp klienta do danych oraz pozwala na dostęp do plików kiedy klient nie jest podłączony do serwera.

6.2 Menedżer pamięci podręcznej

Tak jak każdy system plików, aby używać Cody jądro musi obsługiwać dostęp do jej plików. W przypadku Cody udział jądra w zarządzaniu systemem jest minimalny dzięki współpracy z menedżerem pamięci podręcznej **Venus**. Użytkownik zgłasza żądania do jądra, które albo realizuje je natychmiast, albo prosi **Venus** o wsparcie przy obsłudze żądania.
Wszelkie zmiany w plikach są natychmiast odwzorowywane na lokalnie buforowanych kopiach, a także zapisywane w CML (client-modify log). Mechanizm logowania zmian pozwala, aby rzeczywista zmiana plików na serwerach została odwleczona w czasie. Venus propaguje te zmiany za pomocą mechanizmu zwanego *trickle reintegraction*. Kiedy dokonywana jest propagacja, do serwera dostarczany jest prefiks logu wg porządku czasowego. Rozmiar prefiksu jest dostosowywany do przepustowości łączy.
Każda operacja na pliku to pojedynczy rekord w CML. I tak np. operacja `chmod` zapisywana jest w rekordzie `CHMOD`, a operacja `mkdir` logowana jest jako rekord `MKDIR`. Jeśli były wykonywane operacje zapisu na otwartym pliku, po którym nastąpiła operacja `close` zostanie ona zapisana jako `STORE`. Rekordy tego typu są wyjątkowe, ponieważ powiązane z nimi dane zawierają zawartość pliku. Są znacznie większe niż inne typy rekordów. Mogą mieć wielkość kilkunastu kilobajtów, czy megabajtów, podczas gdy inne nie przekraczają kilkuset bajtów.

6.3 Narzędzia

Aby wspomóc obsługę Cody po stronie klienta dostępnych jest kilka małych narzędzi, z czego najważniejszym jest **cfs**. Dostępne są także: **clog** do logowania się do systemu, **codacon** do monitorowania działania menedżera pamięci podręcznej oraz **cmon** udostępniający informacje na temat listy serwerów.

6.4 Magazynowanie

Coda pozwala „doradzać” menedżerowi pamięci podręcznej **Venus** które pliki są krytyczne i powinny być przechowywane w pamięci podręcznej. Użytkownik ma możliwość ustalenia względnej ważności plików poprzez nadawanie im priorytetów. Operacja ta nazywana jest *magazynowaniem* (hoarding). **Venus** utrzymuje wewnętrzną bazę danych o takich plikach. Magazynowanie plików pomaga upewnić się, że będą dostępne podczas operacji bezpołączeniowych.

6.5 „Write-back caching”

W Unixowym modelu wejścia-wyjścia każdy zapis jest od razu widoczny dla czytelników. Coda (i wcześniej AFS) używają innej semantyki. To co jest widoczne dla innych stacji roboczych ma dokładność do zamknięć pliku, zamiast do indywidualnych zapisów. Oznacza to, że zmiany w pliku są widoczne dla innych dopiero po zamknięciu pliku. Coda buforuje operacje zapisu w

CML. Kiedy klient (nazwijmy go klient A) wykonuje operacje zmieniającą plik żąda od serwera tzw. *write-back permit*. Przed udzieleniem go serwer przerywa wszystkie żądania wobec plików wymienionych w zapytaniu. Dalsze modyfikacje mogą być teraz zapisywane w CML, który jest dostarczany do serwera asynchronicznie. Zanim inny klient (klient B) otrzyma dostęp do danych serwer musi unieważnić zezwolenie i odpisuje, aby klient, który je ma zreintegrował wszystkie jego zmiany tak szybko jak tylko może. Klient B sprawdza wersje plików przechowywane na serwerze ze swoimi lokalnymi kopiami i pobiera zmodyfikowane obiekty. Oczywiście ten schemat nie działa dobrze we wszystkich sytuacjach. Np. kiedy wielu klientów współdzieli wolumen, a dokonywane modyfikacje dotyczą tylko kilku plików udzielanie write-back permit będzie fatalne, ponieważ stałe przerywanie operacji i kolejne sprawdzania urosną do dużych rozmiarów. Jednakże dla katalogów domowych użytkowników dzielonych przez niewiele maszyn ten mechanizm może działać całkiem sprawnie.
W ogólności logowanie operacji na plikach zamiast natychmiastowego wykonywania na nich operacji przynoście spore korzyści w wydajności (jak pokazały testy nawet do 6 razy).

6.6 Operacje bezpołączeniowe

Kiedy wszystkie serwery, na których przechowywany jest pewnie obiekt staną się niedostępne, klient będzie używał lokalnej kopii obiektu (jeśli oczywiście takowa została wcześniej stworzona) jako poprawnej repliki oryginalnego zasobu. Przechodzi wtedy w tryb bezpołączeniowy.

Tryb bezpołączeniowy może być wynikiem awarii sieci, bądź też czasowym odłączeniem komputera przenośnego. Tak więc jeśli wszystkie potrzebne pliki zostaną lokalnie zbuforowane użytkownik może odłączyć komputer od sieci i pracować tak jakby był do niej podłączony. Aby nie stracić z pamięci lokalnej potrzebnych nam plików w pracy bezpołączeniowej (**Venus** może uznać, że potrzebne nam pliki nie są na razie potrzebne i usunąć je z pamięci podręcznej) Coda pozwala na nadawanie plikom priorytetów buforowania, co pomaga zachować w pamięci podręcznej interesujące nas pliki.

6.7 „Weak connected mode" (tryb słabego połączenia)

W tym trybie operacje zapisu są opóźniane, jednak odczyt jest pełny (z pamięci podręcznej bądź serwera). Operacja ta jest też zwana „write-disconnected operation" (operacja pisania bezpołączeniowego). Zapisywanie jest wykonywane wtedy co jakiś czas w tle. Tryb ten jednak zwiększa szanse powstania konfliktów. Jest używany gdy:

1. słabe połączenie z siecią (poniżej 50KB/s)

2. użytkownik zażyczył sobie tego trybu. Ma wtedy możliwość ustalenia jak często będzie dokonywana reintegracja danych. Może ustawić sobie częstsze niż standardowe 15 minut uaktualnianie (np. co 5 sekund). Przydatne jest też wtedy, gdy użytkownik wykonuje operacje na plikach, których czas życia jest krótki. Wetdy w przeciągu standardowych 15 minut plik może zostać utworzony, zmieniony i skasowany. Wtedy nie ma potrzeby jakiejkolwiek aktualizacji serwera, a taka sytuacja może być łatwo wykryta poprzez skanowanie CML.

Sytuacje, w których klient Cody jest odłączony od pewnych lub wszystkich wolumenów dostępnego serwera:

1. oczekująca reintegracja - kiedy modyfikacje wolumenu zostały przeprowadzone w trybie bezpołączeniowym klient nie podłączy wolumenu, dopóki wszystkie dokonane zmiany nie zostaną zintegrowane. Oczywiście reintegracja się nie powiedzie bez właściwych tokenów uwierzytelniających. Co więcej reintegracja jest odkładana w czasie, dopóki są jakieś obiekty w konflikcie.

2. prawa dostępu - klient może się również rozłączyć, kiedy serwer stwierdzi błąd w operacji, która według klienta powinna być poprawna. Powodem tego są błędy uwierzytelniania, bądź sprzeczność pomiędzy danymi zbuforowanymi lokalnie przez klienta, a tymi przechowywanymi na serwerze.

3. utracone połączenie - czasami klient nie otrzymuje odpowiedzi z dostępnego serwera i oznacza go jako „martwy". Oczywiście spowoduje to odłączenie wolumenu, jeśli ostatni serwer zostanie utracony. Raz na 5 minut klient automatycznie weryfikuje połączenia ze wszystkimi znanymi serwerami i w ten sposób może przywrócić utracone połączenie. Taka czynność może być również wykonana na polecenie użytkownika.

6.8 Uwierzytelnianie plików

Kiedy Coda wykryje, że serwer po pewnym czasie nieobecności jest znowu osiągalny, sprawdza, czy lokalne pliki przed użyciem, aby się upewnić, czy są to najnowsze wersje. Coda porównuje oznaczenia wersji trzymane wraz z każdym plikiem z wersjami przechowywanymi na serwerze. Dokładniej to z plikiem trzymany jest wektor wersji zawierający po jednej liczbie całkowitej dla każdego serwera z VSG przechowującej dany plik. Kiedy serwer *i* otrzyma od klienta komunikat zmiany pliku zwiększa *i* - tą wartość wektora wersji. Klient informuje serwer *i* o zmianach w wektorze wersji przez inne serwery, skutkiem czego wektory wersji mogą być zgodne. W przypadku awarii serwera wektory wersji pozwalaja na porównanie wersji i w większości przypadków system może automatycznie wybrać aktualną wersję pliku.

6.9 Przykładowa operacja na pliku

Załóżmy, że wydajemy polecenie: `cat /coda/tmp/abc` aby wyświetlić zawartość pliku przechowywanego w Codzie. Cat wykona kilka wywołań systemowych w odniesieniu do tego pliku. Jądro stwierdza, że jest to plik Cody i przekazuje jego obsługę do odpowiedniego modułu. Moduł Cody utrzymuje swój lokalny cache ostatnich zapytań do VFSu (Virtual File System). Jeśli zapytanie nie może być zrealizowane przy pomocy tej pamięci, to jest przekazywane do menedżera pamięci podręcznej Venus (poza jądrem). Ten z kolei sprawdza dyskowy cache klienta w poszukiwaniu pliku `tmp/abc`. Jeśli plik jest znaleziony, Venus odpowiada jądru, które z kolei przekazuje wynik programowi wołającemu (w tym wypadku „cat"). Kiedy jądro przekazuje zadanie otwarcia pliku po raz pierwszy, Venus pobiera cały plik z serwerów używając mechanizmu zdalnego wołania procedur (RPC). Następnie składuje plik w pliku pamięci

podręcznej. Jest to zwykły plik na dysku i mogą być na nim wykonywane zwykłe operacje (co oczywiście nie jest zalecane). Jeśli plik jest otwarty po raz drugi nie będzie pobierany z serwerów, a wykorzystana zostanie właśnie jego lokalna kopia. Pliki katalogów, jak również wszystkie atrybuty są buforowane przez Venus.

Możliwe jest więc wykonywanie operacji na plikach bez połączenia z serwerem, o ile oczywiście istnieje on w pamięci podręcznej. Kiedy plik zostanie zmieniony i zamknięty Venus uaktualnia zawartość serwerów przesyłając im nowy plik. Inne operacje takie jak tworzenie katalogów, usuwanie plików lub katalogów itp. są również propagowane na serwery. Kiedy Venus chce dostać się do obiektu na serwerze, musi wcześniej znaleźć „VolumeInfo" dla wolumenu zawierającego plik. Czynność ta jest wykonywana jednokrotnie i dostęp do innych plików z danego wolumenu jej już nie wymaga. Ta informacja zawiera listę serwerów i numery lokalnych wolumenów na każdym serwerze, który przechowuje dany wolumen. Dla plików komunikacja z serwerami w VSG jest „jednokrotnego odczytu, wielokrotnego zapisu", tzn. plik jest czytany z pojedynczego serwera w VSG, a zmiany są propagowane na wszystkie dostępne serwery w VSG (AVSG).

7 Rozwiązywanie konfliktów

Ponieważ Coda replikuje różne części zasobów po pewnym czasie mogą pojawić się konflikty pomiędzy serwerami. Konflikt powstaje, kiedy ten sam obiekt jest uaktualniany w różnych częściach sieci. Np. załóżmy, że plik jest dostępny na serwerze A i B. Teraz serwery przestają się nawzajem widzieć, a dwóch użytkowników w różnych częściach sieci modyfikuje ten plik. Kiedy podział sieci się zakończy plik będzie w konflikcie na 2 serwerach. Taka sama sytuacja może się zdarzyć, gdy użytkownik zakończy pracę w trybie bezpołączeniowym i ponownie przejdzie w normalny tryb.

Coda gwarantuje wykrycie konfliktu przy pierwszym dostępie do obiektu kiedy oba serwery są dostępne. Kiedy zostanie wykryty konflikt, Coda próbuje uruchomić automatyczne rozwiązywanie konfliktów. W prostych przypadkach konflikty są rozwiązywane na drodze automatycznej, a cały proces jest przeźroczysty dla użytkowników z wyjątkiem oczywiście pewnego opóźnienia w dostępie do pliku.

Jednakże w trudniejszych przypadkach automatyczne rozwiązywanie konfliktów zawodzi, a obiekt oznaczany jest jako *w konflikcie*. Systemowe odwołania do obiektu, który jest w konflikcie

zawodzą z takim samym błędem jak podczas czytania wiszących, tylko do odczytu linków symbolicznych (linków do plików, które nie są w danym momencie odstępne). Konflikt musi zostać rozwiązany przez użytkownika z odpowiednimi prawami dostępu do obiektu. Aby ułatwić użytkownikom naprawianie konfliktów Coda udostępnia narzędzie naprawcze.

8 Bezpieczeństwo

Zagadnienie bezpieczeństwa w Codzie można podzielić na trzy części:

1. **Uwierzytelnianie i bezpieczne połączenia** – Coda zawiera pakiet RPC2, który dostarcza mechanizm do bezpiecznego uwierzytelniania klientów na serwerach i na odwrót oraz ustanawiania szyfrowanego kanału pomiędzy nimi. Kluczowym elementem tego schematu są hasła dla użytkowników Cody.

2. **Kontrola dostępu** – Pliki na serwerach Cody są chronione poprzez listy ACL. Takie listy udostępniają prawa dostępu dla użytkowników oraz ich grup.

3. **Kopie zapasowe** – Aby zwiększyć bezpieczeństwo składowania danych Coda oferuje system tworzenia kopii zapasowych.

8.1 Prawa dostępu do plików i listy ACL

Coda udostępnia podobną semantykę praw jak UNIX. *Acces control list* (ACL) kontroluje dostęp do katalogów dla użytkowników i całych ich grup. ACL przyporządkowuje każdy element członkowi pewnej domeny ochrony praw. Prawa użytkownika są zdeterminowane przez prawa grup do których on należy pośrednio bądź bezpośrednio. Poza ACL ustawiane są jeszcze 3 bity właściciela pliku, które są wykorzystywane do zaznaczenia możliwości pisania, czytania i wykonywania. W Codzie dostępne są następujące uprawniania oraz ich kombinacje:

- **r** *Read* – Z tym prawem użytkownik może czytać dowolny plik w katalogu.
- **l** *Lookup* – Lookup umożliwia przeglądania informacji o statusie plików w katalogu np. żeby zobaczyć jakie pliki się w nim znajdują.
- **i** *Insert* – Umożliwia użytkownikowi tworzenie nowych plików bądź podkatalogów
- **d** *Delete* – Użytkownik może usuwać pliki albo podkatalogi
- **w** *Write* – Umożliwia zapisywanie istniejących plików w katalogu
- **a** *Administer* – Pozwala użytkownikowi na wprowadzanie zmian do ACL

Dostępne są też prawa odwrotne, tzn. można powiedzieć, że dana grupa użytkowników nie posiada określonego prawa.

8.2 Uwierzytelnianie użytkowników

Coda zarządza uwierzytelnianiem i autoryzacją poprzez tokeny wymagając od użytkowników zalogowania się. Podczas procesu logowania klient otrzymuje klucz sesji zwany tokenen w zamian za poprawne hasło. Token jest związany z identyfikatorem użytkownika. Jego ważność wygasa po ok. 25 godzinach. Po tym czasie wymagane jest ponowne zalogowanie się. Obecnie jest to uid użytkownika, który się loguje. Jeśli token nie jest obecny domyślnie przyjmowany jest dostęp anonimowy, którego prawa dostępu występują na każdej liście kontroli dostępu i zwykle są minimalne.

8.3 Uwierzytelnianie i bezpieczne połączenia

Głównymi składnikami tego mechanizmu uwierzytelnianie i ustanawiania bezpiecznych połączeń są:

1. *Szyfrowanie z tajnymi kluczami dzielonymi*: przy obsłudze protokołów w wielu miejscach przyjmowane jest domyślnie, że obydwaj uczestnicy połączenia dzielą „tajny" klucz umożliwiając nadawcy szyfrowanie danych, a odbiorcy ich odszyfrowanie. Oczywiście dzielone klucze nie mogą być przesyłane przez sieć w postaci jawnej.

2. *Ustanawianie bezpiecznych połączeń*: obejmuje mechanizm do ustanawiania bezpiecznych połączeń jeśli dwie strony dzielą klucz i publiczny „identyfikator klienta". „Bezpieczne" oznacza uwierzytelnione i zaszyfrowane.

3. *Protokół uwierzytelniania*: protokół umożliwiający generowanie klucza dzielonego, zwanego „kluczem sesji" lub „kluczem uścisku dłoni", używanym pomiędzy klientem, a serwerem plików, bazującym na komunikacji z *serwerem uwierzytelniania*. Wynikiem sesji uwierzytelniania jest otrzymanie przez klienta klucza sesji bazującego na dostarczonym poprawnym haśle. Początkowo ten klucz jest znany tylko klientowi i serwerowi uwierzytelniania.

Serwer uwierzytelniania i serwer plików (znajdujące się zwykle na jednej fizycznej maszynie, ponieważ operacja uwierzytelniania jest stosunkowo rzadka) dzielą między sobą tajne klucze, dzięki czemu mogą bezpiecznie komunikować się z klientem. W ten sposób serwer plików może ustanowić bezpieczne połączenie z klientem.

8.3.1 Szyfrowanie

W Codzie do szyfrowania używany jest system RPC2_XOR. RPC2 pozwala na połączenie z serwerem na 4 sposoby:

OpenKimono: bez uwierzytelniania i szyfrowania

AuthOnly: uwierzytelnianie ale bez szyfrowania

HeadersOnly: uwierzytelnianie oraz nagłówki pakietów RPC zaszyfrowane

Secure: uwierzytelnianie oraz pakiety RPC zaszyfrowane

8.4 System kopii zapasowych

W miarę, jak rozwijała się Coda, pojawiła się potrzeba na godny zaufania system archiwizowania danych z dużą pojemnością i przy minimalnych stratach na dostępności usług oferowanych przez Codę. „Jednooperacyjny" system kopii zapasowych okazał się nie do pogodzenia z Codą,

ponieważ powodowałby długie czasy przestojów serwerów podczas wykonywania kopii zapasowych. Dlatego został stworzony system trójfazowy.

8.4.1 Klonowanie

W pierwszej fazie zwanej „*klonowaniem*” wykonuje się zamrożenie dla każdego wolumenu tworząc klon tylko do odczytu replikowalnego wolumenu, a następnie jest on odmrażany. Umożliwia to kontynuowanie innych operacji podczas wykonywania obrazu. Po zakończeniu tej fazy przywracane są normalne usługi „do odczytu i zapisu” bez obawy o uszkodzenie danych z powodu wykonywania operacji na aktywnym systemie plików.

8.4.2 Zrzucanie do plików dysków na maszynie obsługującej kopie bezpieczeństwa

Ta faza składa się z przekonwertowania klonów „tylko do odczytu” na obrazy dysków przechowywane na jako zwykłe pliki na przeznaczonej do tego maszynie. Zrzucenie może być pełne, podczas którego wszystkie pliki są przekazywane lub przyrostowe w którym przekazywane są tylko te pliki i katalogi, które zmieniły się od ostatniego udanego archiwizowania. Pomiędzy pełnymi zrzutami wykonuje się kilka zrzutów przyrostowych. Przydaje się to, gdy tylko pewien podzbiór wolumenów wymaga pełnej archiwizacji, zmniejszając ilość danych składowanych „offline” oraz zmniejsza obciążenie sieci. Jednakże przyrostowe zrzuty możliwe są tylko dla wolumenów replikowalnych. Dla wolumenów niereplikowalnych możliwe są tylko pełne zrzuty.

8.4.3 Zapisywanie na nośniku

Ostatnią fazą jest zapisanie wszystkich plików ze zrzutami znajdujących się na lokalnej partycji na media archiwalne takie jak taśma. W tej fazie może być użyty dowolny, standardowy system archiwizacji.

Tak więc tworzenie kopii wygląda następująco:

1. tworzenie klonu tylko do odczytu
2. zrzucanie klonu na lokalny dysk maszyny obsługującej kopie zapasowe
3. zapisanie zrzuconych danych na odpowiednie nośniki

Przywracanie wygląda następująco:

1. odczytanie właściwego pełnego o przyrostowych zrzutów z nośników archiwalnych
2. połączenie pełnego i przyrostowego zrzutu do czasu, od którego chcemy odzyskać dane
3. przywrócenie w pełni zintegrowanej kopii zapasowej do systemu Coda (wolumeny kopii zapasowych)

W praktyce częstym powodem odzyskiwania danych jest przypadkowe usunięcie pliku przez użytkownika. Dlatego jeśli plik był usunięty nie później niż 24 godziny wcześniej można go odzyskać z wolumenów kopii zapasowych dostępnych bezpośrednio w systemie Coda. W przeciwnym wypadku zwykle będzie konieczne odzyskanie danych z nośników archiwalnych.

9 Źródła

Powyższe materiały stanowią jedynie przegląd tego, jak wygląda i funkcjonuje Coda. Dodatkowe informacje na temat działania, instalacji i użytkowania systemu można znaleźć pod adresem: www.cs.cmu.edu/afs/cs/project/coda/Web/docs-coda.html Dostępne są tam również prace naukowe na temat różnych aspektów Cody i ogólnie rozproszonych systemów plików.