Rozproszone systemy plików
AFS i Coda

Konrad Witkowski

22 stycznia 2003

Spis treści

1 System plików AFS

1.1 Wprowadzenie

Andrew jest rozproszonym środowiskiem obliczeniowym opracowanym w Carnegie-Mellon University jako uniwersytecki system obliczeniowy. Wykorzystuje się w nim system plików Andrew (Afs - *Andrew File System*) jako sposób dzielenia informacji przez użytkowników.

1.2 Właściwości

1.2.1 Cechy ogólne

Afs jako system rozproszony musi spełniać podstawowe założenia projektowe systemu rozproszonego. Czyli:

- przezroczystość dostępu - dostęp do plików rozproszonych odbywa się przez ten sam interfejs co do plików lokalnych. Umożliwia to programom użytkownika działanie na plikach zdalnych bez konieczności ich rekompilacji czy przeładowania
- przezroczystość położenia - użytkownik powinien widzieć jednolitą przestrzeń nazw plików; zmiana fizycznego położenia pliku nie powinna powodować zmiany ścieżki dostępu do pliku
- przezroczystość współbieżności - zmiany w pliku dokonane przez jednego użytkownika nie powinny zaburzać działań innych użytkowników na pliku
- przezroczystość awarii - zapewninie poprawnego działania serwera podczas awarii klienta oraz zapewninie poprawnego działania klienta przy problemach komunikacyjnych z serwerem
- przezroczystość wydajności - programy klienckie powinny być wykonywane z zadowalającą wydajnością pomimo określonych w pewnym przedziale obciążeń serwera
- przezroczystość sprzętu i systemu operacyjnego - niezależność systemu rozproszonego od sprzętu na którym działa oraz od systemu operacyjnego wykorzystywanego przez system rozproszony
- bezpieczeństwo - zapewnienie dostępu do zasobu tylko użytkownikom do tego uprawnionym

1.2.2 Cechy specyficzne

- skalowalność - powinna istnieć możliwość stopniowego rozszerzania usług na coraz większą liczbę komputerów działających przy coraz większych obciążeniach;skalowalność została przyjęta za najistotniejszy cel projektowy systemu AFS
- plikowe pamięci podręczne - zastosowanie podręcznych pamięci *cache* na komputerach klienckich

- usługi całoplikowe - serwery AFS przesyłają do maszyn klienckich całe pliki, a nie ich części

1.3 Dlaczego taka konstrukcja ?

Przy projektowaniu systemu AFS kierowano się kilkoma obserwacjami dokonanymi w środowiskach akademickich.

- większość plików jest czytana i zapisywana przez jednego użytkownika jest więc duża szansa na utrzymanie w pamięci podręcznej klienta aktualnej kopii pliku przez długi czas
- większość plików wykorzystywanych przez użytkowników jest mała
- dominującymi są operacje czytania nie pisania
- dostęp do plików jest najczęściej sekwencyjny
- odwołania do plików są skumulowane tzn. jeżeli odwołanie nastąpiło niedawno jest duża szansa ,że niedługo się powtórzy

1.4 Przykład działania

1. Gdy proces użytkownika w komputerze klienckim odwoła się do pliku dzielonego pliku systemu, sprawdzana jest aktualność kopii lokalnej w pamięci podręcznej. Jeżelin kopia nie jest aktualna to odnajdywany jest serwer przechowujący poszukiwany plik i wysyłane jest do niego zamówienie.
2. Kopia przesłana przez serwer jest zapamiętywana lokalnie i otwierana u klienta.
3. Wszystkie następne operacje *read* i *write* są wykonywane lokalnie.
4. Gdy proces użytkownika wykona operację *close* wówczas lokalna kopia zostaje przesłana z powrotem do serwera (jeśli wystąpiły w niej zmiany).

1.5 Realizacja w architekturze klient-serwer

Oprogramowanie systemu AFS składa się z dwu części zrealizowanych w architekturze klient-serwer.

Część oprogramowania, nazywana Vice, to proces poziomu użytkownika działający na serwerze. Nazwa Venus została przypisana procesowi użytkownika wykonywanemu w każdym komputerze klienckim.

W systemie AFS pliki są pogrupowane w tomy (*volumes*). Tom zawiera poddrzewo plików i katalogów powiązanych ze sobą logicznie. Przykładem może być katalog domowy komputera podłączonego do sieci studenckiej. Pliki użytkowników są podmontowywane w katalogu

/home danego komputera przez co użytkownik może widzieć swoje pliki na każdej maszynie podłączonej do systemu.

Proste usługi plikowe są w AFS realizowane przez serwery Vice. Obsługą hierarchicznej struktury katalogów zajmuje się zbiór procesów Venus na komputerach klienckich.

Wszystkie pliki w dzielonej przestrzeni plików systemu są identyfikowane przez 96-bitowe, jednoznaczne identyfikatory. Procesy Venus zajmują się tłumaczeniem nazw ścieżek na te identyfikatory.

32 bity	32 bity	32 bity
Numer tomu	Uchwyt pliku	Niepowtarzalny kwalifikator

Numer tomu - wyodrębnienie tomu w którym znajduje się dany plik

Uchwyt pliku - (standard NFS) wyróżnia plik w tomie

Niepowtarzalny kwalifikator - zapewnia, że te same identyfikatory nie będą używane ponownie

Serwery Vice przyjmują zlecenia od procesów Venus używając do tego tylko ww. identyfikatorów. Dlatego proces Venus musi sam uzyskać identyfikator pliku w drodze wielokrotnego przeszukiwania plików katalogowych serwera.

1.6 Zachowanie spójności pamięci podręcznej

Gdy serwer dostarcza do klienta kopię pliku zostaje dołączona do niej informacja o tzw. obietnicy powiadomienia (*callback promise*). Zobowiązuje to proces Vice do informowania wszystkich zainteresowanych procesów Venus o ewentualnych zmianach dokonywanych na pliku przez innych klientów. Obietnice powiadomienia,przechowywane lokalnie z plikami, mogą mieć dwa stany : ważny,nieważny. Gdy serwer dostaje zlecenie aktualizacji pliku wówczas powiadamia o tym wszystkie procesy Venus którym to „obiecał”. Proces Venus powiadomiony o aktualizacji ustawia znacznik obietnicy na nieważny.

Proces Venus sprawdza ważność obietnicy przy każdorazowym wykonaniu operacji *open*. Gdy znacznik jest nieważny oznacza to,że trzeba ściągnąć aktualną wersję pliku z serwera.

Opisane rozwiązanie nie gwarantuje jednak współbieżnych aktualizacji. Jeżeli plik zostanie odczytany przez klientów i po jakimś czasie zapisany to przetrwa tylko ostatnia aktualizacja.

Jeżeli komputer klienta ulega awarii (proces Venus próbuje zachować jak najwięcej danych już sprowadzonych z serwera) to po ponownym uruchomieniu proces Venus musi sprawdzić aktualność wszystkich (tylko aktualnych) obietnic powiadomienia. Może się bowiem zdarzyć,że nieaktywny klient nie otrzyma sygnału aktualizacji od serwera. Z tego względu,po uruchomieniu,proces Venus odpytuje serwer o aktualność wszystkich wyglądających na aktualne plików.

W systemie AFS zastosowano także sposób zachowania spójności pamięci. Proces Venus odpytuje także serwer o ważność pliku który jest otwierany z pamięci podręcznej ,a którego ważność nie jest potwierdzona od czasu T (T jest ustalony systemowo - zazwyczaj kilka minut). Ma to na celu zabezpieczenie przed różnego rodzaju awariami komunikacyjnymi które mogły doprowadzić do utraty komunikatów zawiadamiających.

W prototypowym systemie AFS do zapewniania spójności pamięci podręcznej zastosowano mechanizm znaczników czasu. Przy każdej operacji *open* proces Venus musi odpytać serwer o aktualność pliku. Takie rozwiązanie powoduje duże obciążenie serwera nawet przy samych operacjach odczytu.

Rozwiązanie oparte na zawiadomieniach wymaga jednak od serwerów plikowych przechowywania informacji na temat klientów którym obiecano zawiadomienia dotyczące odpowiednich plików. Wykazy tych zawiadomień muszą więc przetrwać awarię systemu, dlatego też są one pamiętane na dyskach serwerów i uaktualniane przy pomocy niepodzielnych operacji.

1.7 Tomy

Każdy serwer zawiera kopię bazy danych informacji o położeniu tomów plików. Baza ta odwzorowuje nazwy tomów w nazwy serwerów plików na których te pliki aktualnie są dostępne. Przy przenoszeniu tomu mogą powstawać chwilowe niedokładności, lecz nie mają one wpływu na system gdyż na serwerze z którego tom się wywodzi są informacje gdzie dany tom można odnaleźć obecnie.

1.8 Zwielokrotnienia do czytania

Tomy zawierające pliki, które są w częstym użyciu , lecz rzadko są modyfikowane (np. katalogi */bin* czy */usr/bin* systemu Unix) mogą zostać zwielokrotnione na kilku serwerach, ale (!!!) jako tomy przeznaczone tylko do czytania. W systemie nadal istnieje jedna kopia pliku, którą wolno czytać i zapisywać. Wszelkie uaktualnienia będą dokonywane za pomocą tej jednej kopii. W przypadku korzystania z plików zwielokrotnionych w ten sposób w bazie danych o ich położeniach są wszystkie serwery natomiast wybór zależy od obciążenia i ew. innych czynników.

1.9 Bezpieczeństwo

Komórka - to zbiór maszyn serwerów i klientów które dzielą ustawienia konfiguracyjne. Oprócz maszyn do komórki są także przypisani użytkownicy. Użytkownicy mogą mieć konta w wielu komórkach.

Podczas logowania do komórki w której użytkownik posiada konto, procesy Vice i Venus muszą ustalić autentyczność użytkownika. Proces Venus wysyła hasło użytkownika, a proces Vice sprawdza jego zgodność z autentycznym. Jeżeli hasła się zgadzają użytkownik jest zatwierdzony, a jego proces Venus dostaje znacznik autentyczności.

1.10 Wydajność

Ponieważ podstawowym celem projektowym systemu AFS jest skalowalność, szczególną troskę przykłada się do wydajności pracy z wieloma użytkownikami. AFS korzysta z transmisji masowych pomiędzy klientami,a serwerem. Pliki są przesyłane w porcjach po 64KB. Zastosowanie tak dużych pakietów minimalizuje wpływ opóźnień sieciowych optymalizując wykorzystanie

łącza. Niektóre testy systemu wykazują (przy pewnych standardowych warunkach) ,że obciążenie serwera wynosiło 40% podczas gdy w podobnych warunkach serwer oparty na systemie NFS wykazywał obciążenie 100%. Ponadto należy wziąć pod uwagę fakt,że serwery NFS są wbudowane w jądro natomiast system AFS działa jako procesy poziomu użytkownika.

2 System plików Coda

2.1 Wprowadzenie

System plików Coda jest następcą systemu AFS. Został stworzony na bazie AFS,a ponadto w zależeniach projektowych przyjęto kilka nowych punktów opracowanych na bazie doświadczeń z systemem AFS oraz nowych warunków pracy systemu rozproszonego.

2.2 Założenia projektowe systemu

Mimo ,iż system AFS sprawdzał się bardzo dobrze w warunkach uniwersyteckich zaczęto zauważać kilka niedociągnięć tego systemu które mogłyby znacznie poprawić jakość usług.
Główne wymagania stawiane systemowi Coda jako spadkobiercy AFS:

- Zauważalna stała się zwężona forma zwielokrotnień ograniczonych do plików tylko do czytania.
- Ulepszeń wymagało tolerowanie uszkodzeń przez usługi AFS. Skala systemu AFS powodowała ,że jednego dnia występowało kilka awarii.
- Powstał także nowy problem którego system AFS nie rozwiązywał w żaden sposób - użytkowanie komputerów przenośnych. Czyli wszystkie pliki potrzebne użytkownikowi powinny być przechowywane w pamięci podręcznej komputera.

System Coda miał spełniać wszystkie te wymagania określane mianem stałej dostępności do danych. Celem systemu Coda było udostępnienie użytkownikowi magazynu danych na serwerze,a jednocześnie w sytuacjach awaryjnych polegania na zasobach własnych.

2.3 Realizacja

W przeciwieństwie do systemu AFS, Coda przechowuje tomy do odczytu i zapisu na wielu komputerach. Zwiększa to przepustowość systemu, a jednocześnie poprawia tolerowanie uszkodzeń. Rozbudowany został sposób magazynowania informacji w pamięci podręcznej komputerów-klientów umożliwiający dostęp do zasobów podczas odłączenia od sieci.

Zbiór serwerów przechowywujący kopie tomu plików nazywany jest grupą tomów pamięci (*volume storage group* - VSG). Klient chcący otworzyć plik w takim tomie może uzyskać dostęp do pewnego podzbioru grupy VSG - AVSG (*available* VSG). Skład tej grupy zmienia się w zależności od dostępności serwerów.

Mechanizm powiadamiania stosowany w systemie Coda jest podobny do tego z AFS. Dochodzi jednak potrzeba uaktualniania wszystkich kopii pliku znajdujących się na wszystkich serwerach w grupie AVSG (przy aktualizacji są propagowane zgłoszenia).

Jeżeli AVSG jest pusta to dochodzi do operacji „odłączonej”. Może to być spowodowane problemami komunikacyjnymi,ale także świadomym odłączeniem stacji od systemu. Operacja taka może być wykonywana przy założeniu,że w pamięci podręcznej komputera istnieją pliki

konieczne do jej realizacji. Z tego też względu stacje klienckie muszą współpracować z systemem aby wygenerować listę plików potrzebnych klientowi na czas odłączenia.

W założeniach systemu Coda przyjęto,że pliki przechowywane w pamięci podręcznej klienta są mniej znaczące niż pliki przechowywane na serwerze (są lepszej jakości). Z tego też względu kopie przechowywane w pamięci podręcznej można uważać za zdatne do użytku pod warunkiem ich okresowego uaktualniania.

2.4 Spójność pamięci podręcznej

W systemie Coda używana jest optymistyczna strategia zwielokrotniania. Pozwala ona na wykonywanie zmian w plikach podczas podziału sieci lub w operacjach odłączonych.

Zastosowany mechanizm polega na przypisywaniu każdej wersji pliku wektora wersji systemowej (*Coda version vector* - CVV) oraz znacznika czasu. Wektor ten utworzony jest z liczb całkowitych odpowiadających licznikom (lub tylko oszacowaniom) liczb zmian wykonanych na pliku utrzymywanym w danym serwerze. Informacje zawarte w CVV mają dać wystarczające informacje aby powstałe niespójności można było rozwiązać automatycznie (po powrocie do normalnego stanu systemu) lub ,jeżeli automatyczna procedura zawiedzie, użytkownik zostanie o tym powiadomiony. Automatyczne usuwanie niespójności działa tylko gdy wszystkie elementy wektora na jednym stanowisku są większe od elementów na innym stanowisku.

Gdy następuje zmiana zawartości pliku, proces Venus wysyła do każdego serwera z AVSG komunikat o uaktualnieniu z zawartością pliku i bieżącym rekordem CVV. Proces Vice odbiera komunikat i sprawdza czy CVV jest spójny z jego aktualnym. Jeżeli nie jest spójny to proces Vice uaktualnia plik i CVV zwracając potwierdzenie pozytywne. Proce Venus oblicza wtedy nowy wektor zwiększając liczniki dla wektorów które odpowiedziały pozytywnie i rozsyła nowy wektor do członków AVSG.

Ponieważ CVV jest rozsyłany tylko do członków grupy AVSG,a nie do VSG to każdy serwer będzie zawierał dokładny licznik tylko dla samego siebie. Pozostałe liczniki będą z reguły zaniżone.

2.5 Przykład

Rozważmy grupę VSG=A,B, komputery klientów f1 i f2 oraz plik P.

1. Początkowo wektory CVV w VSG są równe [1,1,1].

2. Klient f1 otwiera plik P i wykonuje operację zapisu (i *close*) wysyłając komunikat o aktualizacji tylko do swojej grupy AVSG=A.

3. Klient f2 otwiera plik i wykonuje operację aktualizacji dwa razy. Wysyła komunikat o aktualizacji do swojej grupy AVSG=B.

4. Po usunięciu awarii sieci okazuje się,że serwer A ma CVV=[2,1],a serwer B ma CVV=[1,2]. Taka sytuacja oznacza sprzeczność i konieczność ręcznej aktualizacji pliku.

5. Gdyby jednak klient f2 nie dokonywał zmian na pliku to po przywróceniu osiągalności można automatycznie zaktualizować zawartość serwera B.

2.6 Dostęp do kopii pliku

Przy wyborze serwera z którego mamy dostać plik możemy kierować się różnymi wskazówkami (bliskość fizyczna, małe obciążenie). Po wybraniu serwera otrzymujemy od niego kopię pliku i musimy sprawdzić czy jest to kopia aktualna rozsyłając zapytania do wszystkich innych serwerów. Jeżeli kopia jest aktualna to OK,w p.p. wybieramy serwer który ma najnowszą kopię pliku i ponawiamy zlecenie.

2.7 Spójność pamięci podręcznej

Z konstrukcji systemu wynika,że u każdego klienta proces Venus musi wykrywać następujące sytuacje:

- udostępnienie poprzednio niedostępnego serwera
- zaginięcie poprzednio dostępnego serwera
- utracenie zawiadomienia

W tym celu proces Venus klienta wysyła co każde T (kilka 10) sekund komunikat próbny do wszystkich serwerów w kompletnych grupach VSG. Odpowiedzi przychodzą tylko od serwerów dostępnych.

Jeżeli proces Venus dostał odpowiedź od serwera poprzednio niedostępnego to unieważnia wszystkie obietnice zawiadomień plików z tego serwera.

Jeśli proces Venus nie dostanie odpowiedzi od serwera poprzednio dostępnego to (poza modyfikacją AVSG) nie robi nic (chyba,że utracony serwer był wybrany wtedy usuwa wszystkie obietnice powiadomień z tego serwera).

Przypadek uaktualnień niezauważonych przez serwer. W odpowiedzi na komunikaty próbne serwery odsyłają procesowi Venus wektory CVV. Jeżeli wykryta zostanie niezgodność między wektorami to oznacza,że niektóre serwery z AVSG mają nieaktualne niektóre pliki. Venus przyjmuje tu założenie pesymistyczne i unieważnia obietnice zawiadomień wszystkich plików pochodzących z danego tomu.

2.8 Działanie w odłączeniu

System Coda został zaprojektowany z myślą o działaniu na komputerach klienckich czasowo odłączonych od sieci. Aby to zrealizować konieczne jest jednak przechowywanie plików systemu na dysku twardym komputera-klienta. Jeżeli klient ma dostateczną ilość miejsca przyznaną systemowi to możliwe jest przechowywanie wszystkich plików na jego dysku. Jeżeli jednak miejsca jest mało to należy wybrać tylko niektóre pliki. W tym celu stworzono narzędzi pomagające systemowi ustalić które pliki trzeba przechowywać na dysku,a które do pracy w

odłączeniu nie są potrzebne. Ponadto system Coda daje użytkownikowi możliwość określenia priorytetów dla plików przechowywanych w pamięci podręcznej.

Gdy działanie w rozłączeniu się kończy system Coda rozpoczyna proces ponownego scalania plików z pamięci podręcznej z ich kopiami na serwerach. Dla każdego zmienionego pliku Venus dokonuje ciąg operacji aktualizujących,aby doprowadzić do identyczności kopii w pamięci podręcznej i serwerach.
Jeżeli podczas operacji aktualizacji wykryty zostanie problem którego nie można automatycznie usunąć wówczas klient dostaje o tym powiadomienie, a kopia jego pliku jest przechowywana na serwerze w tymczasowych tomach pomocniczych. Takie podejście wynika z filozofii przyjmującej,że kopie na serwerze są ważniejsze niż kopie na konkretnych maszynach klienckich.

2.9 Wydajność

Wydajność systemu Coda jest porównywalna z wydajnością systemu AFS. Przy niewielkiej liczbie użytkowników systemy zachowują się prawie identycznie. Zwiększenie liczby użytkowników pociąga za sobą znaczny spadek wydajności systemu Coda wobec systemu AFS.