Rozproszone systemy plików, na przykładzie OpenGFS i Lustre

Projekt z przedmiotu RSO

Łukasz Reszka
L.Reszka@elka.pw.edu.pl

15 czerwca 2005

Spis treści

1 Open GFS

System plików OpenGFS, można w jednym zdaniu opisać następująco: System plików OpenGFS jest lokalnym systemem plików, który można rozszerzyć na rozproszony system plików. Można tak powiedzieć, gdyż OpenGFS wymaga, aby wszystkie nośniki danych (dyski, macierze dyskowe), na których będzie operował OpenGFS, były widoczne jako jedno urządzenie, które można adresować w sposób ciągły (bez żadnych luk). OpenGFS, gdy działa jako lokalny system niewiele różni się od systemu plików ext3:

- Każdy obiekt systemu plików jest opisany za pomocą i-węzła, który w przypadku OpenGFS-a nosi nazwę dinode (distributed inode, czyli rozproszony i węzeł).
- Dane zapisane w pliku mogą być zapisane albo bezpośrednio w i-węźle lub w blokach danych, na który wskazuje i-węzeł albo w blokach na które wskazują bloki pośrednie.
- Nazwy plików i katalogów przechowywane są we wpisach katalogowych.
- Jest jurnaling operacji zapisu, na systemie plików.
- Jest specjalny blok (superblok) w wiadomym miejscu, którym przechowywane są informację gdzie znajdują się inne bloki

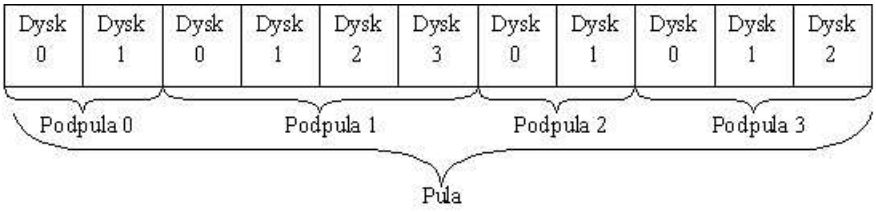
Jednak istnieją pewne różnice w stosunku do ext3:

- Informacja o wolnych blokach nie jest przechowywana w postaci jednaj mapy wolnych bloków. Wolne bloki zostały podzielone na grupy, nazwane grupami zasobów (resource group) i każda grupa ma własną mapę bitów.
- Dla każdego klienta, (komputera który ma załadowany moduł obsługi OpenGFS) prowadzony jest osobny dziennik (jurnal). Dzienniki przydzielane są poszczególnym klientom podczas montowania systemu plików

1.1 Mapowanie przestrzeni dyskowej

OpenGFS do działania potrzebuje dodatkowego modułu, potrafiącego np. wszystkie dyski komputerów w klastrze przedstawić jako pojedyncze urządzenie, widoczne w */dev/costam*, które można adresować poprzez przesunięcie względem jakiegoś ustalonego początku. Innymi słowy, niech jakiś dysk reprezentuje bloki od numeru 0 do n. Jakiś innym dysk musi reprezentować bloki o numerach n+1 do n+m itd. Na podstawie przesunięcia mamy dostęp do kolejnych dysków w klastrze. Sam OpenGFS adresuje bloki tak, jakby były na lokalnym dysku tzn. poprzez urządzanie z katalogu */dev/* oraz przesunięcie. Dlatego aby OpenGFS był systemem rozproszonym potrzebuje pewnego wspomagania w postaci mapowania przestrzeni dyskowej. Jak można przeprowadzić takie mapowanie, przedstawię na przykładzie modułu Pool (pula), który stanowi cześć pakietu OpenGFS (nie jest częścią systemu plików). Dyski lub partycje, z których będziemy tworzyć jeden duży dysk, nazywamy pulą. Dodatkowo w każdej puli możemy wyróżnić pewne grupy dysków tzw. podpule (subpool), które stanowią fragment puli (rysunek 1).

Każda pula ma swój unikalny identyfikator oraz swoją nazwę. Każda podpula ma przypisany swój numer w puli (0 ? liczby popul w puli). Każdemu dyskowi



Rysunek 1: Sposób mapowania dysków w podpule oraz pule.

lub partycji przypisujemy kolejny numer w obrębie podpuli. Na każdym dysku lub partycji, w obrębie puli, zapisujemy powyższe informacje oraz dodatkowo: liczbę wszystkich podpul w obrębie puli, liczbę dysków i partycji w obrębie podpuli oraz rozmiar każdego dysku lub partycji. W momencie ładowania modułu Pool odczytywane te informacje i tworzone jest urządzenie reprezentujące całą pulę. Wszystkie odwołania do niego są mapowane na poszczególne urządzenia w puli.

Podpule zostały wprowadzone aby był możliwy striping na kilka dysków. Każdej podpuli przypisywana jest informacja dotycząca sposobu stripingu.

OpenGFS pozwala na to, aby mapowanie odbywało się za pomocą innego rozwiązania np. EVMS ((ang. *Enterprise Volume Management System*)) lub DM ((ang. *Device Mapper*)).

1.2 System blokad

Kolejnym elementem, który jest konieczny do tego, aby OpenGFS mógł działać, jako rozproszony system plików, jest system blokad. System blokad ma za zadanie:

- Zarządzać i przechowywać LVB (Lock Value Block, czyli bloku danych związanych z blokadą) oraz informacje o blokadach globalnych
- Wykrywać deadlocki
- Wykrywać wygaśnięcie blokady.
- Sprawdzać czy wszystkie węzły działają.
- Powiadomić system plików, że jakiś komputer nie odpowiada.

Oprócz systemu blokad globalnych w samym systemie plików znajduje system blokad lokalnych służących do synchronizacji procesów w dostępie do systemu plików. To właśnie ta część systemu plików komunikuje się z globalnym systemem blokad.

OpenGFS wyróżnia dwa podstawowe rodzaje blokad:

- Blokada potrzebna do odczytu, wówczas dostęp do pliku mają wszyscy, którzy chcą czytać plik.
- Blokada potrzebna do zapisu. Wówczas proces, który chce pisać do pliku otrzymuje dostęp do pliku na wyłączność.

Istnieją jeszcze specjalna blokada, umożliwiająca odtworzenie dziennika, po wygaśnięciu blokady zapisu.

Obiekty, które można zablokować to:

- i ? węzeł (dinode)
- grupa zasobów
- dziennik
- superblok

1.3 Radzenie sobie z błędami

W momencie montowania systemu plików, pewien wyróżniony komputer odtwarza dzienniki wszystkich pozostałych komputerów. Pozostałe komputery mogą odtworzyć tylko swój dziennik. Następnie po odtworzeniu dziennika wyróżniony komputer przydziela wszystkim komputerom dzienniki, w których będą zapisywać informacje o operacjach zapisu. Każda operacja zapisu realizowana jest jako transakcja z jednoczesnym zapisem w dzienniku. O tym że coś się popsuło pierwszy dowiaduje się globalny system blokad, kiedy wygasają blokady, a właściciel blokad nie odpowiada. Wówczas:

- Jeżeli to była blokada na odczyt, a ktoś chce pisać, to system mu na to pozwala. Blokada po prostu zostaje unieważniona.
- W przypadku wygaśnięcia blokady do zapisu, blokada zostaje oznaczona jako wygasła, blokowany jest dziennik właściciela blokady, która wygasła, specjalną blokadą. Następnie odtwarzany jest ten dziennik. Dopiero wówczas następuje unieważnienie blokady na zapis i odwołanie blokady odtworzenia dziennika.

Jeżeli dziennik przeznaczony dla jakiegoś komputera, stanie się niedostępny, to jeżeli istnieje jakiś niewykorzystany dziennik, to staje się nowym dziennikiem danego komputera. Jeżeli wszystkie pozostałe dzienniki są używane, to tworzony jest nowy dziennik z dostępnych niewykorzystanych bloków.

1.4 Grupy zasobów

Nowym rozwiązaniem jakie wprowadzono w OpenGFS są grupy zasobów. Każda grupa zasobów opisuje za pomocą mapy bitowej część bloków przestrzeni ?dyskowej?. W lokalnych systemach plików (ext2 ext3) jest jedna mapa bitowa umieszczona w superbloku. Twórcy OpenGFS-a uznali, że jeśli jest jedna mapa bitowa, a wiele procesów na różnych komputerach chcących zapisać coś do pliku, to taka pojedyncza mapa bitowa stała by się wąskim gardłem całego systemu plików. Dlatego podzielono, tę mapę, na kilka mniejszych części i nazwano grupami zasobów. Grupy zasobów tworzone są w momencie tworzenia systemu plików oraz powiększania systemu plików. W momencie montowania systemu plików, każdemu klientowi zostaje przydzielona grupa zasobów, na wyłączność. Jeżeli grupie zasobów, przydzielonych danemu klientowi, nie ma wystarczającej liczby wolnych bloków, to klient przegląda nie zajęte przez nikogo grupy zasobów, które mają dostateczną liczbę wolnych bloków do przeprowadzenia operacji zapisu. Jeżeli nie znajdzie, to przeszukuje swoją i nie zajęte przez nikogo grupy zasobów w poszukiwaniu takiej, w której zmieszczą się tylko same dane. Jeżeli nie znajdzie to operacja zapisu kończy się niepowodzeniem.

Schemat działania dla pojedynczej operacji (z pominięciem etapu lookup).

1. Wywołanie przez użytkownika funkcji, działającej na systemie plików.

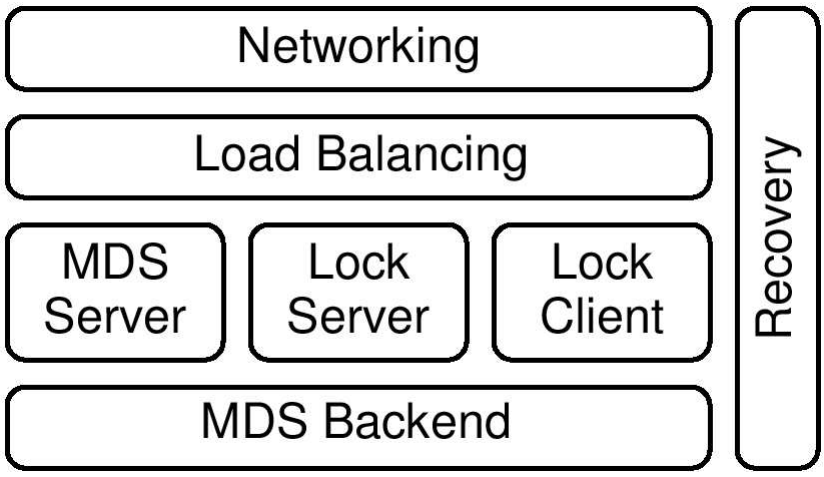
2. System plików blokuje dostęp do pliku albo dla odczytu albo dla zapisu.
3. Jeżeli jest to operacja zapisu, to następuje zapis do dziennika.
4. Jeżeli jest to potrzebne następuje blokada grupy zasobów, a następnie modyfikacja mapy bitowej.
5. Wykonywany jest zapis lub odczyt danych.
6. Jeżeli była to operacja zapisu wymazywany jest wpis z dziennika.
7. Odwoływana jest blokada pliku.

1.5 Zalety OpenGFS:

- prosta koncepcja
- OpenGFS nie potrzebuje żadnej wiedzy o urządzeniach, na których zapisuje dane, o liczbie klientów, o typie sieci łączącej komputery.
- Może korzystać z różnych modułów mapujących przestrzeń dyskową.
- Może korzystać z różnych systemów blokad.

1.6 Wady OpenGFS:

- Podczas tworzenia systemu plików, wszystkie grupy zasobów są zapisywane obok siebie na początku dostępnej przestrzeni adresowej wraz z superblokiem, spisem dzienników i i?węzłem opisującym główny katalog. Prawdopodobnie wszystkie te dane zostaną zapisane na jeden fizyczny dysk. W przypadku jeśli ten dysk stałby się niedostępny, to nie można zrealizować żadnej operacji na systemie plików. System blokad będący częścią pakietu OpenGFS ? memexp, działa w oparciu o pewien centralny serwer blokad. Jeżeli serwer blokad stanie się niedostępny, to również zbyt wiele zrobić się nie da.
- Wyróżnione zostały dwa typy wolnych bloków: bloki na dane i bloki na metadane. Jeżeli brakuje bloków na metadane, to wolne bloki na dane są przerabiane na bloki na metadane, poprzez zmianę bitu w mapie bitowej. Niestety nie zaimplementowano przerobienia wolnych bloków na metadane na bloki na dane.
- Grupy zasobów powodują nienajlepsze gospodarowanie przestrzenią. Wszystkie dane jednego pliku muszą się zmieścić w obrębie jednej grupy zasobów (plik nie może być zapisany jednocześnie w różnych grupach zasobów). Co ciekawsze może istnieć nawet grupa zasobów, z dostatecznie dużą liczbą wolnych bloków, ale jeśli została przydzielona innemu klientowi, to nie możemy z niej skorzystać.



Rysunek 2: Schemat budowy serwera metadanych

2 Lustre

Rozproszony system plików Lustre, jest całkowicie inny niż OpenGFS. System plików Lustre bywa nazywany obiektowym systemem plików, gdyż dane zapisuje w postaci obiektów. W obiekcie oprócz samych danych zapisane są pewne dodatkowe informacje (np. identyfikator obiektu). Jednak większość informacji jest przechowywana w i?węźle, opisującym dany obiekt. Z każdym obiektem powiązany jest przynajmniej jeden i?węzeł, a do opisania jednego obiektu, dowolnej długości, wystarczy jeden i ? węzeł (nie potrzeba żadnych bloków pośrednich). Katalogi w systemie plików Lustre w zasadzie nie różnią się bardzo od katalogów w systemach plików ext2 lub ext3. W systemie plików Lustre można wyróżnić trzy podstawowe elementy:

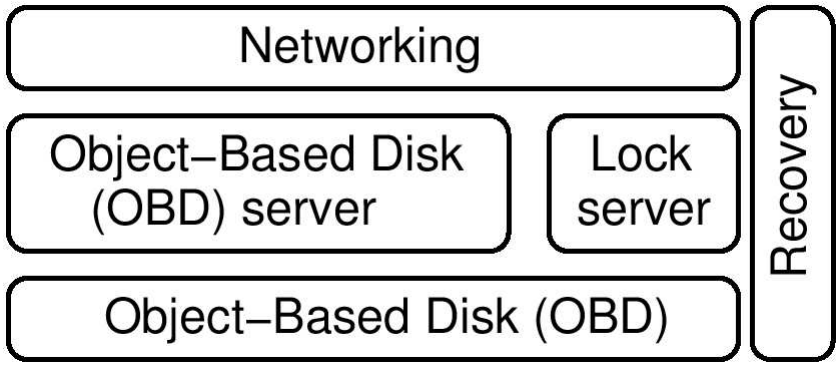
- Serwer metadanych MDS (metadata serwer).
- Magazyn obiektów OST (Object Storage Target)
- Klient systemu plików

2.1 Serwer Metdanych

Podstawowym zadaniem serwera metadanych to przechowywanie i zarządzanie i ? węzłami, katalogami, oraz tablicą LOV (opis niżej). Serwer metadanych, pełni również funkcje serwera blokad. Schematyczną budowę MDS przedstawia rysunek 2.

W skład serwera metadanych wchodzą następujące elementy:

- Networking ? odpowiedzialny za komunikację z innymi elementami systemu plików lustre
- Moduł równoważenia obciążenia (nie ma go w dotychczasowych wersjach)
- MDS Serwer ? serwujący metadane
- Serwer blokad ? umożliwiający dokonywanie blokad, podczas wykonywania operacji



Rysunek 3: Schemat budowy OST

- Klient systemu blokad ? umożliwia MDS skorzystanie z systemu blokad
- Moduł wspomagający odporność na nieoczekiwane zdarzenia.
- Moduł umożliwiający zapisanie danych przechowywanych na serwerze metadanych na lokalnym systemie plików np. ext3

2.2 OST

Zadaniem OST jest przechowywanie obiektów.

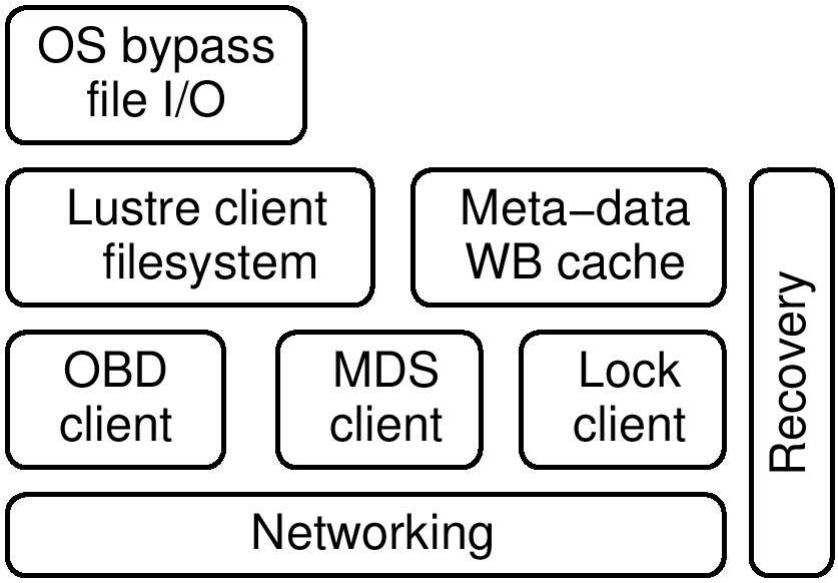
OST (Rysunek 3) składa się z następujących elementów:

- Modułu odpowiedzialnego za komunikację z innymi elementami systemu plików.
- Moduł wspomagający odporność na nieoczekiwane sytuacje
- Serwer blokad umożliwiający blokowanie obiektów lub fragmentów obiektów na danym OST
- OBD - dysk umożliwiający składowanie obiektów. Zazwyczaj jest to pewna nakładka na lokalny system plików np. ext3, umożliwiająca zapis obiektów (zazwyczaj w postaci plików, na lokalnym systemie plików). Z każdym OST musi być powiązany przynajmniej jeden OBD.
- OBD serwer umożliwia dostęp do składowanych na danym OST obiektów.

2.3 Klient sytemu plików lustre (CFS)

W skład oprogramowania klienta (Rysunek 4) systemu plików wchodzą:

- Moduł odpowiedzialny za komunikacje
- Moduł wspomagający ochronę przed nieoczekiwanymi zdarzeniami
- Cache metadanych
- Klient systemu blokad.



Rysunek 4: Schemat budowy klienta systemu plików Lustre

- Klient umożliwiający komunikację z serwerem metadanych
- Klient umożliwiający komunikację z OST. Jest ich tyle ile OST i każdy odpowiada za komunikację tylko z określonym OST.
- Moduł pośredniczący między wirtualnym systemem plików systemu operacyjnego, a systemem plików lustre
- Moduł umożliwiający przyspieszenie operacji wejścia/wyjścia na plikach, poprzez ominięcie systemu operacyjnego (mniej kopiowań z bufora do bufora).

Odczyt i modyfikacje metadanych, przez klienta, odbywa się poprzez komunikację z serwerem MDS. Natomiast odczyt i zapis danych z obiektu, odbywa się podczas bezpośredniej komunikacji z konkretnym(i) OST.

2.4 Uruchamianie Lustre

Na początku należy zainstalować stosowane oprogramowanie dla poszczególnych węzłów (MDS, OST lub klienckie). Kolejnym krokiem jest utworzenie pliku konfiguracyjnego w formacie XML. W pliku tym zapisane są informacje o poszczególnych elementach systemu plików lustre, w szczególności każdemu z tych elementów musi zostać przypisany unikalny identyfikator tzw. UUID, który posłuży później do odwoływania się do poszczególnych elementów systemu. Elementy, którym zostaje nadany UUID to, poza MDS i OST również OBD, ale co ciekawe także modułowi klienckiemu dla MDS, oraz modułom klienckim dla poszczególnych OST.

Plik konfiguracyjny umieszczony jest albo na każdym komputerze, będącego częścią systemu plików lustre, albo w bazie LDAP. Na podstawie informacji zawartych w tym pliku, zarówno każdy klient jak i serwery dowiadują się o pozostałych elementach

systemu lustre. Podczas montowania systemu zbyt dużo się nie dzieje. Właściwie to CFS pobiera z MDS atrybuty katalogu głównego, sprawdza czy ma odpowiednie uprawnienia, i jeśli tak to pobiera wpisy katalogowe z katalogu głównego.

2.5 LOV i odnajdowanie obiektów

Mogłoby się wydawać, iż skoro mam unikalny identyfikator do poszczególnych OST a nawet OBD (UUID), to zapisując UUID w i ? węźle moglibyśmy przechować informację o tym, gdzie znajduje się konkretny obiekt. Niestety, nie jest to dobre rozwiązanie z dwóch względów:

- Lustre dopuszcza migrację danych. Jeżeli zastosowano by to rozwiązanie to po każdej migracji należało by zmodyfikować i ? węzły wszystkich migrujących obiektów.
- Lustre umożliwia zapisanie obiektu na wielu OST. UUID 1000 OST, na których chcemy zapisać obiekt, oraz sposób stripingu mogłaby się nie zmieścić w i ? węźle.

Dlatego wprowadzono deskryptory dla poszczególnych składnic obiektów. W i ? węźle zapisana jest odwołanie do konkretnego deskryptora. Dopiero w deskryptorze zapisane zostają UUID konkretnych OST (lub OBD), na których przechowywany jest dany obiekt. Aby umożliwić striping obiektów na wiele OST, wprowadzono specjalny obiekt LOV (Logical Object Volume). W obiekcie tym zapisane są UUIDy wszystkich OST, na których będzie składowany obiekt oraz sposób stripingu. Każdy LOV ma przypisany unikalny UUID, który jest wpisywany do deskryptora. Aby umożliwić szybkie odnajdywanie konkretnych LOV, umieszcza je się w tablicy. Tablica LOV jest przechowywana na MDS. Aby klient mógł odczytać lub zapisać obiekt potrzebuje: i ? węzła danego pliku, deskryptor wskazywany przez i ? węzeł oraz ewentualnie obiekt LOV. Sposób zapisu plików pomiędzy OST może być zmieniany przez użytkownika na pomocą ioctl.

2.6 System blokad

W Lustre wyróżnia się dwa rodzaje operacji: przeprowadzane na metadanych oraz przeprowadzane na obiektach. Dlatego systemów blokad jest tyle ile serwerów OST +1. Systemy blokad, związane z OST, umożliwiają blokadę dostępu do obiektów na czas realizacji operacji zapisu i odczytu. Natomiast system blokad związany z MDS umożliwia realizację operacji na metadanych.

2.6.1 System blokad związany z operacjami na metadanych

W Lustre operacja lookup nie jest przeprowadzana tak samo dla wszystkich operacji. W Lustre operacja lookup jest wykonywana pod kontem konkretnej operacji. MDS musi znać cel wykonania operacji lookup, dzięki czemu po wykonaniu operacji lookup zostają założone odpowiednie blokady, przez co po zakończeniu lookup mamy pewność, że nikt nie namiesza w danych, na których realizujemy operacje. Z punktu widzenia operacji lookup wyróżniamy cztery klasy operacji:

- Operacje typu namei. Wymagana jest blokada do odczytu katalogu.

- Operacje umożliwiająca odczytanie atrybutów, zapisanych i ? węźle i powiązanych z daną ścieżką dostępu. Operacja ta nie wymaga żadnych blokad, a jedynie zapamiętania atrybutów w czasie wykonywania operacji lookup. Podczas operacji zapisu do obiektu, MDS nie ma informacji na temat aktualnej wartości atrybutów ctime oraz aktualnego rozmiaru pliku. Przed operacją zapisu do obiektu (w czasie lookup), atrybuty, które mogą ulec zmianie, oznaczane są jako nieważne. Klient, aby poznać aktualną wartość tych atrybutów, musi skomunikować się z OST, na którym przechowywany jest dany obiekt.
- Operacje modyfikujące atrybuty obiektu (chmod, chown, utimes). Wymagana jest blokada do zapisu dla konkretnego i ? węzła.
- Operacje modyfikujące informacje zapisane w katalogu (mkdir, rmdir, unlink, create, mknod, rename, symlink). Wymagana jest blokada do zapisu na danym katalogu.

2.6.2 System blokad związany z operacjami zapisu i odczytu obiektów

W tym systemie wyróżniono aż pięć różnych typów blokad na obiekcie:

- Blokada na wyłączny dostęp do pliku (nikt inny nie może ani czytać ani pisać)
- Chroniony zapis (nikt inny nie może zapisywać w tym samym czasie)
- Chroniony odczyt (nikt nie może zapisywać w tym czasie)
- Równoległy zapis (nie ma znaczenia co inni robią)
- Równoległy odczyt (nie ma znaczenia co inni robią)

Trzy pierwsze tryby nie wymagają komentarza. Natomiast dwa ostatnie są dosyć ciekawe. Jeżeli chcemy dostępu w trybie 5 to możemy odczytać dane częściowo zmodyfikowane. W przypadku trybu 4 godzimy się, aby ktoś wspólnie z nami modyfikował plik. Oczywiście gdy dwa różne procesy zaczęłyby modyfikować ten sam obszar obiektu to wynik byłby nieustalony. Tak naprawdę w trybie 4 nie jest możliwe aby dwa procesy modyfikowały ten sam obszar. W trybie 4 godzimy się jedynie aby ktoś inny modyfikował część pliku, którego my nie modyfikujemy. W trybie 5 godzimy się jedynie na to, aby ktoś modyfikował część pliku, której my aktualnie nie odczytujemy. Tryb 4 może być zastosowany tylko jeśli długość pliku nie ulegnie zmianie. Zastosowanie tych trybów może być podczas realizacji obliczeń rozproszonych, gdy obliczenia są prowadzone na olbrzymich macierzach, które nie mieszczą się w pamięci. Aby możliwa była realizacja trybu 4 lub 5 w blokadzie musi zostać zapisana informacja, dotycząca której części pliku dotyczy. Każda blokada zawiera dwie liczby z przedziału [0, 2$\hat{}$64-1], określające fragment obiektu, którego dotyczy. Jeśli druga liczba jest największą z możliwych tzn. 2$\hat{}$64-1, to oznacza, że modyfikowana jest długość pliku. Specjalna blokada w której pierwsza liczba jest równa ?1 (2$\hat{}$64-1), a druga 0, oznacza iż chcemy poznać wielkość aktualnie modyfikowanego pliku.

2.7 Cache

Każdy klient ma swój cache na metadane. Do cache ? u trafiają i ? węzły, wpisy katalogowe, LOV oraz deskryptory. Dodatkowo cache działa na zasadzie opóźnionego zapisu (Write Back), dzięki czemu poszczególne operacje mogą być ze sobą

?sklejane?. Np. Jeżeli wykonujemy polecenie tar ?xf plik.tar, w którym jest bardzo dużo plików, to blokowanie katalogu, na czas tworzenia kilku plików jest lepsze niż blokowanie na czas tworzenia poszczególnych plików. Realizowane jest to w ten sposób, że po otrzymaniu blokady (w czasie lookup) żądania są buforowane przez jakiś czas, a jeśli nastąpi jakieś zdarzenie, które opróżni cache, to przekazujemy te dane dopiero do MDS i po ich uaktualnieniu MDS zdejmuje blokadę.

Przy zwykłej konfiguracji systemu plików Lustre dane odczytywane, lub zapisywane z obiektów są cache ? owane jedynie w OST. Natomiast istnieje możliwość uruchomienia COBD (Cache Object Driver), który cache ? uje dane z OST ? ów. Serwer COBD jest wykorzystywany tylko przy operacjach odczytu, i ma za zadanie odciążyć serwery OST (zmniejszyć obciążenie niektórych OST). Działa to w ten sposób, iż obiekty (lub ich fragmenty) zostają umieszczone na serwerze COBD. Klient chcący odczytać dane z OST, zostaje ewentualnie przekierowany do odpowiedniego serwera COBD. Podczas zapisu obiektu cache, związany z tym obiektem zostaje unieważniony.

2.8 Radzenie sobie z błędami

Rzeczą oczywistą jest iż jeśli serwer metadanych nie będzie działał, to nie będzie możliwe przeprowadzenie żadnej operacji na systemie plików. Ponieważ na razie nie działa równoważenie obciążenia, pomiędzy dostępne serwery MDS więc wprowadzono tymczasowe rozwiązanie. Wprowadzono drugi serwer MDS, który jest wierną kopią pierwszego (wszystkie informacje, które posiada pierwszy, ma też drugi). Jeżeli nie ma żadnych nieoczekiwanych sytuacji, to działa tylko jeden, który przekazuje drugiemu zapisywane informacje. W momencie, kiedy podstawowy serwer MDS stanie się niedostępny, zapasowy MDS przejmuje jego zadania i realizuje operacje na metadanych. Jeżeli podstawowy serwer MDS wróci do działania, to pobiera świeże informacje z zapasowego serwera MDS i przejmuje ponownie obsługę klientów. W przypadku uszkodzenia, któregoś OST/OBD część obiektów staje się niedostępna, co nie przeszkadza przeprowadzać operacji na metadanych lub obiektach umieszczonych w innych OST/OBD. Co prawda istnieje możliwość dublowania OST (podobnie jak serwera MDS), jednak raczej rzadko jest to konieczne. Wszystkie operacje na systemie plików Lustre, przeprowadzane są transakcyjnie (albo się wszystko uda albo nic). Jeżeli transakcja się nie uda, to klient ponawia próbę. Aby wspomóc obsługę transakcji w poszczególnych elementach systemu przechowywanych jest szereg zmiennych:

Same dane lub metadane umieszczone na dysku są chronione poprzez jurnaling lokalnego systemu plików.

2.9 Bezpieczeństwo

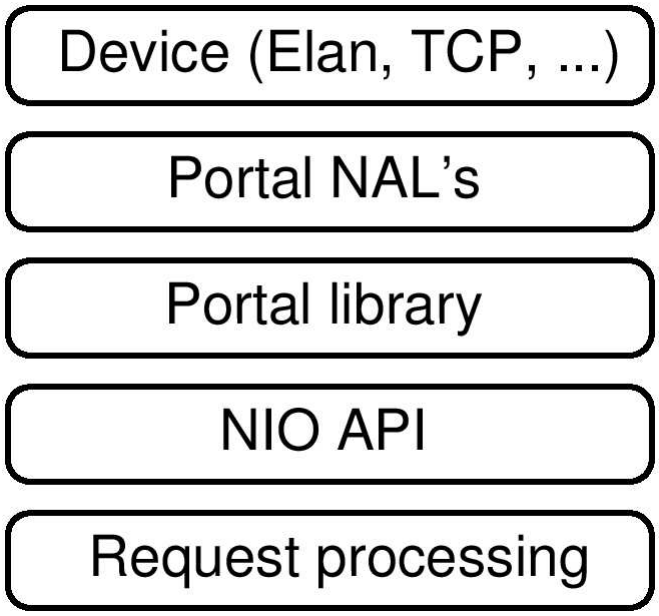
Uprawnienia dostępu do pliku, realizowane są za pomocą list kontroli dostępu.

W najnowszej wersji systemu plików Lustre dołożono elementy, zapewniające bezpieczną wymianę informacji w sieci, pomiędzy elementami systemu plików Lustre. Bezpieczeństwo wymiany informacji w tym rozwiązaniu, opiera się na systemie Kerberos.

2.10 Warstwa komunikacji sieciowej

Opisując system plików Lustre wypada wspomnieć o komunikacji sieciowej, która w lustre jest nietypowa. Stos sieciowy w Lustre wygląda mnie więcej tak i została

Nazwa zmiennej	Opis
numer epoki (epoch number)	Przechowywany jest indywidualnie przez wszystkie elementy zapisujące cokolwiek na dysk. Przy każdym uruchomieniu i odtworzeniu po błędzie, numer ten jest zwiększany. W powiązaniu z danymi o transakcji, umożliwia określenie, które transakcje kwalifikują się do odtworzenia.
numer wcielenia (incarnation number)	Przechowywany przez MDS. Zwiększany przy zmianie konfiguracji systemu plików. Wspomaga odtwarzanie stanu zapasowego MDS.
numer pokolenia (generation number)	Numer połączenia pomiędzy klientem a MDS lub OST. Zwiększany jeśli klient się rozłączy.
identyfikator transakcji (xid)	Monotonicznie rosnący numer żądania, przechowywany przez klienta. Każde żądanie musi mieć unikalny numer dla konkretnego klienta. Służy do określania, które żądani zostały już potwierdzone, a które nie.
numer transakcji (transno)	Przechowywane przez serwer MDS/OST i jest unikalne dla transakcji na danym serwerze. Wykorzystywane do upewnienia się przez klienta, czy transakcje były realizowane przez serwer w odpowiedniej kolejności.
poziom połączenia (connection level)	Służy do określenia co było wysłane przed uszkodzeniem jakiegoś elementu, a co po uszkodzeniu.
xid ostatnio odebranej transacji (last_recv)	Przechowywany przez serwer MDS/OST xid ostatnio odebranej transakcji. Służy do określenia, którą transakcję trzeba odtworzyć.
xid ostanio potwierdzonej transakcji (last_committed)	Przechowywany przez serwer MDS/OST xid ostatnio potwierdzonej transakcji. Służy do określenia, która transakcja została już potwierdzona.



Rysunek 5: Model komunikacji sieciowej

zaczerpnięty z rozwiązań dla RDMA.

W najwyższej warstwie (na rysunku 5 najniżej) komunikacja odbywa się poprzez RPC. Niżej jest warstwa buforująca (NIO). Następnie jest biblioteka portali, które zarządzają połączeniami i przekazywaniem komunikatów. Ciekawostką jest to, że punktami docelowymi dla portali są procesy, a nie komputery. Następnie jest warstwa uniezależniająca portale od sieci, z jaką mamy do czynienia (NAL). A na samym dole są sterowniki sieciowe.

2.11 Realizacja operacji zapisu

1. Oprogramowanie klienckie odbiera od VFS żądanie utworzenia nowego pliku.
2. Klient wysyła żądanie operacji lookup w celu pobrania atrybutów katalogu, w którym ma zostać utworzony plik.
3. CFS weryfikuje uprawnienia danego użytkownika.
4. Klient wysyła żądanie operacji lookup w celu stworzenia nowego pliku.
5. MDS blokuje dostęp do katalogu i zwraca do klientowi dane, potrzebne do utworzenia pliku.
6. Klient wysyła żądanie utworzenia pliku, o konkretnej nazwie, z określonymi atrybutami.
7. MDS komunikuje się z konkretnym(i) OST i prealokuje obiekt, w którym będą zapisywane dane.

8. MDS tworzy nowy wpis katalogowy, nowy i ? węzeł i deskryptor, dla nowo powstałego obiektu. Ewentualnie nowy LOV.
9. MDS oznacza, że nie posiada aktualnych wartości atrybutów size i ctime.
10. MDS odwołuje blokadę, przekazując do klienta wszystkie dane związane ze świeżo powstałym plikiem.
11. CFS wysyła do konkretnego(ych) OST żądanie otwarcia pliku do zapisu, na wyłączność.
12. CFS przesyła kolejne porcje danych, które OST zapisuje.
13. Po zakończeniu zapisu, klient zamyka plik i odwołuje blokadę.
14. CFS przekazuje do MDS aktualny rozmiar i ctime pliku.

2.12 Zalety:

- Jest bardzo elastyczny.
- OS bypass.
- Wysoki poziom bezpieczeństwa dzięki Kerberosowi.
- Możliwość strippingu.
- Wykorzystuje lokalne systemy plików.
- Istnieje support.
- Możliwość współdzielenia pliku przy zapisie.
- Prawa dostępu oparte na ACL.
- Bardzo skalowalny.
- Działa na różnych sieciach (Ethernet, InfiniBand)

2.13 Wady:

- Każdy użytkownik, może sam określać striping.
- Metadane przechowywane w niewielu miejscach.
- Niebanalna konfiguracja.
- Dostępny kod źródłowy zawiera błędy, celowo wprowadzane, aby klient nabył supporu.

3 CFS (Cluster File System)

CFS jest rozwiązaniem dla Open SSI, umożliwiający współdzielenie jednego systemu plików, przez wiele węzłów w klastrze.

3.1 Specyfika Open SSI

CFS jest ściśle zależne w działaniu od Open SSI. Open SSI zapewnia to iż wszystkie dyski lub partycje, znajdujące się w klastrze, są widziane w ten sam sposób, przez wszystkie komputery będące w klastrze. Innymi słowy urządzenie z katalogu /dev/ (np. hda6), zawsze będzie skojarzone z konkretnym urządzeniem fizycznym, w konkretnym węźle, niezależnie od ilości węzłów w klastrze oraz kolejności ich podłączania się do klastra. Ta cecha Open SSI zapewnia, iż każdy węzeł traktuje każde urządzenie w klastrze tak, jakby to urządzenie było fizycznie w tym węźle. W odniesieniu do urządzeń dyskowych możemy sobie wyobrazić, iż jeśli w jednym z węzłów klastra mamy jeden dysk, z lokalnym systemem plików, to pozostałe węzły traktują ten dysk tak, jak gdyby był to ich lokalny dysk z lokalnym systemem plików. Dzięki czemu wystarczy w każdym węźle sterownik lokalnego systemu plików, aby każdy węzeł mógł odczytywać i zapisywać dane ze współdzielonego dysku. Niestety Open SSI, ani sterownik lokalnego systemu plików nie zapewnia, iż dwa węzły jednocześnie, nie będą modyfikować wspólnie tego samego pliku. Dlatego konieczne było wprowadzenie CFS.

3.2 CFS

CFS jest warstwą w jądrze, umieszczoną pomiędzy sterownikami lokalnych systemów plików, a Wirtualnym Systemem Plików (VFS). CFS umożliwia współbieżny dostęp do współdzielonego lokalnego systemu plików z warstwy VFS, a z drugiej strony zapewnia iż sterownik lokalnego systemu plików umieszczony w każdym węźle, nie będzie przeszkadzał innym węzłom w dostępie do współdzielonego systemu plików.

CFS zapewnia dwie rzeczy:

- Synchronizację w dostępie do systemu plików
- Pamięć Cache dla operacji dostępu do systemu plików.

3.2.1 Synchronizacja w dostępie do systemu plików

Synchronizacja jest realizowana w sposób zcentralizowany. Jest jeden wyróżniony węzeł, na którym działa serwer przydzielający tokeny, na wykonanie operacji na systemie plików. Każdy węzeł, który chce wykonać operację na systemie plików, wysyła żądanie do serwera o przyznanie tokenu, a następnie jeśli otrzyma token, może realizować operację na współdzielonym systemie plików. Każdemu tokenowi odpowiada pewien znacznik czasowy, dzięki czemu serwer ma możliwość wykrycia wygaśnięcia blokady. Ponadto zadaniem serwera jest unieważnianie pamięci cache w węzłach, po dokonaniu przez jeden z węzłów operacji zapisu.

W CFS wyróżniono trzy sposoby korzystania z pliku:

- Na wyłączność (tylko jeden proces może wykonywać operacje na danym pliku)
- Dzielenie do odczytu (wiele procesów może jednocześnie czytać dany plik)
- Dzielenie do zapisu (wiele procesów może jednocześnie korzystać z pliku)

3.2.2 Pamięć cache w CFS

Pamięć cache w CFS jest realizowana po stronie klienta. Są trzy rodzaje pamięci cache w CFS:

- lookup cache ? pamięć cache dla operacji lookup
- dir cache ? pamięć cache dla wpisów katalogowych
- page cache

W czasie operacji zapisu, pamięć cache realizuje strategie opóźnionego zapisu (write back). Pamięć page cache podczas operacji odczytu, realizuje strategię czytania z wyprzedzeniem (read ahead).

4 Open SSI i rozproszone systemy plików

Open SSI wspiera zarówno system plików Open GFS jak i Lustre. Open GFS obecnie nie jest częścią pakietu Open SSI. Open GFS jest wspierany przez Open SSI, ale musi zostać zainstalowany jako dodatkowy pakiet. Open GFS wymaga mapowania całej przestrzeni dyskowej w jeden spójny obszar, ukryty pod pewnym wpisem w katalogu /dev/ (pewnym wirtualnym urządzeniem). Ze względu na to iż Open SSI, wszystkim urządzeniom w klastrze, nadaje unikany identyfikator. Dlatego w Open SSI możliwe by było mapowanie urządzeń dyskowych, w ciągły obszar adresowy (wirtualnego urządzenia), przeprowadzane lokalnie na każdym węźle klastra Open SSI. Niestety taki rozwiązanie jeszcze nie istnieje. Obecnie mapowanie odbywa się niezależnie od Open SSI. Odwołania do takie wirtualnego dysku, na którym jest utworzony system plików Open GFS, odbywają się za pomocą sterownika systemu plików Open SSI, ale z pominięciem CFS. CFS nie jest składowany pomiędzy sterownikiem Open GFS a VFS, a rolę CFS przejmuje sterownik systemu plików Open GFS. Pakiet Open GFS zawiera w sobie system blokad, o nazwie memexp, oraz moduł pool służący do realizacji wirtualnego dysku. Obecnie jednak memexp został zastąpiony przez system blokad Open DLM, który obecnie stał się częścią pakietu Open GFS i jest zalecanym systemem blokad. Również moduł pool coraz częściej jest zastępowany przez EVMS (Enterprise Volume Management System
). Jednak na razie EMVS nie jest wspierany przez Open SSI.

W przypadku systemu plików Lustre, nie tworzy żadnych dodatkowych urządzeń. Lustre w specyficzny sposób organizuje dostępne zasoby dyskowe, dzieląc je na OST i MDS. Sposób działania systemów Lustre nie zależy w żaden sposób od Open SSI. Jednak instalując system plików Lustre, dokonywane są pewne zmiany w jądrze. Instalując Open SSI dokonywane są znaczne modyfikacje jądra, które mogą mieć wpływ na zmiany dokonane przez system plików Lustre. Dlatego zalecana jest zainstalowanie najpierw Open SSI, a dopiero później sytemu plików Lustre. Ponadto system plików Lustre musi być w odpowiedniej wersji w stosunku do wersji Open SSI. Ze względu na to Lustre stał się częścią pakietu Open SSI i zalecana jest instalacja systemu plików Lustre, będącego częścią tego pakietu.