Analiza właściwości wybranych rozwiązań rozproszonych systemów plików

Karolina KULA, Marcin MARKOWSKI

Politechnika Wrocławska,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
karolina.kula@protonmail.com, marcin.markowski@pwr.edu.pl

STRESZCZENIE: W artykule przedstawiono analizę porównawczą trzech najpopularniejszych rozproszonych systemów plików: Ceph, LizardFS, GlusterFS. Głównym kryterium oceny badanych systemów była ich wydajność, rozumiana jako liczba operacji zapisu i odczytu w jednostce czasu oraz maksymalny transfer danych. Analizę oparto na wynikach licznych eksperymentów przeprowadzonych w opracowanym w tym celu stanowisku badawczym. Część badawczą poprzedzono przeglądem technologii oraz najpopularniejszych istniejących systemów tego typu. W pracy przedstawiono wyniki eksperymentów, ich analizę oraz praktyczne wnioski.

SŁOWA KLUCZOWE: rozproszone systemy plików, wirtualizacja, usługi chmurowe, Ceph, LizardFS, GlusterFS

1. Wprowadzenie

Zakres świadczenia usług informatycznych przez firmy IT stale się poszerza. Rośnie również zapotrzebowanie na zasoby informatyczne. Nie wszystkie technologie stosowane w firmach usługowych dają się skalować. Możliwe jest także, że własne wdrażanie udoskonaleń, zabezpieczeń i redundancji jest zbyt drogie lub ryzykowne. Dlatego wraz ze wzrostem zapotrzebowania na usługi, rozwija się technologia chmur obliczeniowych (ang. *Cloud Computing*), umożliwiająca zdalne świadczenie wszelkiego rodzaju usług dla różnych typów użytkowników (korporacji, małych firm, prywatnych użytkowników). Oferowane usługi to *Infrastructure as a Service, Platform as a Service, Software as a Service*, świadczone są również dedykowane rodzaje usług, np. *Disaster Recovery as a Service*. Niezależnie od tego, jaka usługa będzie świadczona za pomocą chmury, niezmiennie u jej podstawy znajdują się dane. Dane należy gdzieś przechowywać, a dodatkowo zapewnić ich bezpieczeństwo i dostępność. Najniżej w architekturze

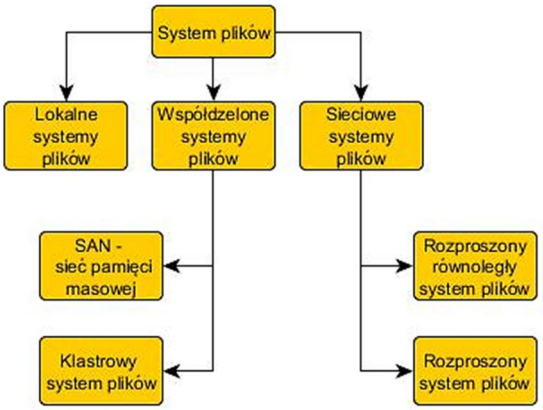
chmury znajdują się fizyczne dyski, natomiast w zasadzie bezpośrednio na dyskach działają systemy plików. Ponieważ dysków zwykle jest bardzo dużo a centra przetwarzania danych znajdują się w różnych fizycznych lokalizacjach (ze względów bezpieczeństwa), systemy plików działające na nich muszą się ze sobą komunikować oraz synchronizować. Systemy dedykowane do takich zadań nazwano rozproszonymi systemami plików (ang. *Distributed File Systems*, DFS).

Zadania stawiane przed rozproszonymi systemami plików są o wiele szersze i bardziej skomplikowanie niż wymagania wobec lokalnych systemów plików. Oprócz samego zarządzania plikami konieczne jest także synchronizowanie działania wielu węzłów (maszyn z dyskami), zapewnienie bezpiecznej komunikacji, a także wydajność i wygoda zarówno procesu obsługi jak i późniejszego administrowania. Z marketingowego punktu widzenia produkt w postaci gotowego DFS musi być elastyczny, aby można było go dostosować do specyficznych wymagań użytkowników. System plików powinno cechować kilka kluczowych właściwości sprawiających, że będzie spełniał oczekiwania [1]:

- Dyspozycyjność – awaria jednego z węzłów nie spowoduje niedostępności klastra (dostęp do danych i funkcji będzie utrzymany).
- Wydajność – system jest w stanie sprostać transferowym wymaganiom użytkowników.
- Skalowalność – właściwość pozwalająca dodawać do klastra nowe węzły zwiększające zasoby bez negatywnego wpływu na wydajność.
- Przejrzystość – żadna operacja wykonywana w systemie plików nie jest widoczna (tj. jest przeźroczysta) dla użytkownika. Operacje mogące mieć wpływ na wydajność i przejrzystość to m.in.: dodanie/usunięcie węzła, obsługa awarii, zapewnianie spójności między replikami.
- Bezpieczeństwo – zapewnienie dostępu wyłącznie dla autoryzowanych użytkowników, bezpieczna komunikacja klaster-klient i węzeł-węzeł, a także zapewnienie dostarczania danych w niezmienionym stanie.

2. Rozproszone systemy plików

Systemy plików nieustannie ewoluują, aby sprostać coraz większej skali potrzeb oraz szeregowi wymagań. Rozwój koncepcji rozproszonego systemu plików, który zapewniałby bezpieczeństwo, skalowalność i dostęp dla wielu użytkowników ma swoje źródło w ,,zwykłym", lokalnym systemie plików. Był on bazą do rozwoju takich technologii jak współdzielony system plików i sieciowy system plików, które następnie ewoluowały. Przykładami pierwszych realizacji zdalnego dostępu do plików były między innymi IFS stworzony przez firmę *Alto Personal Computers* oraz *Datanet file repository* działające w Arpanet, prototypie dzisiejszego Internetu [7]. Rozwój systemów DFS przedstawiono na rysunku 1.



Rys. 1. Rozwój systemów plików

Oprócz gałęzi lokalnych systemów plików (EXT4, btrFS, XFS) na których bazują badane w niniejszej pracy systemy DFS, wyodrębniły się także współdzielone systemy plików oraz sieciowe systemy plików. Przykładem technologii realizującej współdzielony system plików jest SAN (*Storage Area Network*). Jej cechą charakterystyczną jest fakt, że operuje na blokach danych (*Block Level Storage*). Ta właściwość odróżnia współdzielone systemy plików od sieciowych systemów plików (np. NFS, Ceph), które operują na plikach (*File Level Storage*). Jednym z pierwszych pomysłów na realizację rozproszonego systemu plików był Andrew File System (AFS) [8]. Badane w tej pracy systemy należą do gałęzi sieciowych systemów plików. Mimo bardzo wielu koncepcji oraz implementacji tej technologii, istnieje kilka niezmiennych elementów którymi charakteryzuje się każdy rozproszony system plików [2]:

- Mechanizmy replikacji danych. Aby zrealizować podstawową właściwość DFS – dostępność – wprowadzono mechanizmy replikowania danych. Polegają one na dystrybucji kopii wybranych danych na różne serwery. Żądający dostępu klient otrzymuje dostęp do kopii. W zależności od systemu plików istnieją różne strategie rozmieszczania danych (ang. *placement policy*), uwzględniające m.in. geolokalizację serwera czy jego niezawodność.
- Synchronizacja. W systemach DFS należy brać pod uwagę konieczność synchronizacji wszystkich kopii danych. Każda zmiana w kopii musi być uwzględniona oraz zastosowana we wszystkich wystąpieniach.
- Pamięć podręczna (ang. *cache*). Mechanizm stosowany w celu poprawienia wydajności, umożliwiający znaczący wzrost możliwości, ale również wymagający zaimplementowania mechanizmów zapewniających spójność pamięci podręcznej z faktycznymi danymi.

- Mechanizmy rozkładania obciążenia. Jest to właściwość systemu plików umożliwiająca obsługę dynamicznego dodawania lub usuwania serwerów. W razie usuwania, konieczna jest możliwość odzyskiwania danych i przenoszenia ich na inne serwery, a w przypadku dodawania kolejnych węzłów klastra – zdolność przenoszenia danych z obciążonych serwerów na inne, nowo dodane i mniej obciążone.
- Elementy architektury. Każdy DFS musi posiadać elementy odpowiedzialne za obsługę metadanych. Niezbędne są także elementy odpowiedzialne za składowanie danych oraz elementy umożliwiające zarządzanie klastrem.
- Wykrywanie awarii. Awarie w rozproszonych systemów plików, na które składa się bardzo wiele fizycznych urządzeń, są bardzo częste, dlatego wczesne wykrycie i obsługa awarii są kluczowe dla niezachwianego, transparentnego działania systemu plików.
- Interfejsy. Niezbędne jest zapewnienie klientom możliwości wykonywania operacji na danych bez przejmowania się złożoną strukturą systemu. Przykładowe interfejsy to: linia komend (CLI), interfejsy montowane w przestrzeni użytkownika (FUSE) czy webowe API (REST).

W ramach badań przedstawionych w niniejszym opracowaniu przeanalizowano własności następujących rozproszonych systemów plików:

- Ceph [12]. Posiada w pełni zdecentralizowaną architekturę, *Ceph Storage Cluster* składa się z trzech komponentów. Serwer metadanych (*Ceph Metadata Server*) odpowiada za metadane systemów plików zgodnych z POSIX. *Object Storage Device*, jest fizyczną bądź logiczną jednostką przechowywania. W ramach OSD można zunifikować wiele dysków fizycznych. Ostatnim komponentem jest *Ceph Monitor* (MON), monitorujący stan klastra – przechowuje informacje między innymi na temat dostępności węzłów. W ramach klastra może istnieć wiele monitorów oraz serwerów metadanych [2]. Jako jeden z nielicznych rozproszonych systemów plików posiada wsparcie sterowników jądra Linux (*Kerner Driver*). Może być zamontowany jako *File System in User Space* (FUSE), jako *block device* (*Ceph Block Device)* przeznaczony dla zastosowań chmurowych oraz maszyn wirtualnych lub za pomocą interfejsu dostarczonego przez REST API, zgodnego z S3- i SWIFT-API. Ceph dystrybuuje dane w postaci obiektów, używając algorytmu CRUSH (*Controlled Replication Under Scalable Hashing*) oraz mechanizmu *placement groups*. Aby zminimalizować wpływ awarii na dostępność danych skorzystano z mechanizmu replikacji [5]. Ceph umożliwia budowę różnego rodzaju części funkcjonalnych (ang. *backend*) dla OSD. Możliwe jest zastosowanie technologii *filestore, bluestore, memstore*. W zależności od aplikacji mogą one zdecydowanie zwiększać wydajność klastra.

- GlusterFS [13]. System charakteryzuje się architekturą klient-serwer w której brak serwera metadanych. Metadane i dane przechowywane są na dyskach połączonych w klaster. Dostępne tryby działania klastra to replikacja danych lub ich podział na bloki i dystrybucja do różnych lokalizacji (ang. *stripping*). Do lokowania i odnajdywania danych używa funkcji skrótu EHA zapewniającej globalną przestrzeń nazw. EHA konwertuje ścieżkę lokalizacji pliku na unikalną, zunifikowana wartość o stałym rozmiarze. Każdy zasób pamięci posiada przypisany zakres wartości, dzięki któremu wiadomo jakie pliki będzie przechowywał [8]. Takie rozwiązanie problemu metadanych sprawia, że w systemie nie pojawiają się tzw. 'pojedyncze punkty awarii'. Zamontowanie systemu plików w systemie GlusterFS jest możliwe jako NFS, SMB/CIFS, lub *Gluster native client*.
- LizardFS [14]. Programowo zdefiniowany magazyn dyskowy *(*ang. *Software defined storage)* będący pochodną systemu MooseFS. LizardFS przechowuje oddzielnie metadane (drzewa katalogów, nazwy plików, etc.) oraz dane. Metadane obsługuje serwer metadanych, natomiast dane są dystrybuowane pomiędzy serwery klastra (ang. *chunk servers*). Aby zwiększyć ilość dostępnych zasobów, wystarczy dodać kolejne serwery. Dodanie odbywa się bez powodowania opóźnień i może zostać wykonane w dowolnej chwili. LizardFS wspiera trzy rodzaje replikacji: *Standard replica*, *XOR replica* i *EC replica*. Dwie ostatnie bazują na sprawdzaniu parzystości. Podobnie jak pozostałe systemy DFS, LizardFS zapewnia możliwość montowania interfejsu w przestrzeni użytkownika (FUSE). Udostępnia także możliwość montowania jako natywny klient Windows, dzięki czemu system plików jest dostępny jako wirtualny dysk.

Wybór systemów będących przedmiotem badań podyktowany był kilkoma przesłankami. Systemy wybrano spośród rozwiązań o charakterze otwartym (*open source*), kompatybilnych z serwerową dystrybucją systemu Linuks (CentOS). Podyktowane to było dostępnością pełnej wersji systemów oraz faktem, że darmowe oprogramowanie znajduje liczne zastosowania – od instytucji, przez małe firmy do mikroprzedsiębiorstw i użytkowników prywatnych. Wśród otwartych systemów wybrane rozwiązania znajdują się w grupie najczęściej wykorzystywanych. Istotnym kryterium wyboru była również dostępność kompletnej dokumentacji oraz możliwość zbudowania we wszystkich badanych przypadkach porównywalnych konfiguracji rozproszonych systemów plików.

Spośród pozostałych, popularnych rozproszonych systemów plików, znajdujących liczne zastosowania, wymienić należy:

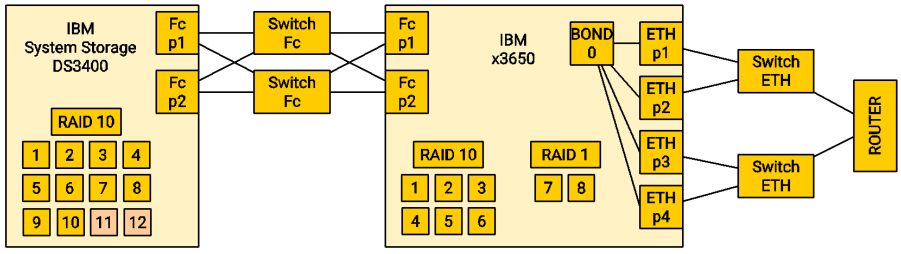
- Google File System. Przeznaczony do użycia w dużych centach danych o dużej intensywności użytkowania. Dostarcza on interfejsy POSIX, sprawne mechanizmy detekcji awarii, zapewnia wysoką przepustowość i optymalizację pod kątem dużych odczytów strumieniowych [3].

- Hadoop Distributed File System. Jest to część *Apache Hadoop Framework*. Bazuje na paradygmacie *Map-Reduce*, mającym za zadanie w wydajny sposób obsługiwać analizę i operacje na dużych zestawach danych. Cechą charakterystyczną jest partycjonowanie zarówno danych jak i obliczeń na wiele maszyn (hostów). Obliczenia te są realizowane równolegle (*paralell computing*) [4]. Dostęp klienta do plików w HDFS jest zrealizowany za pomocą biblioteki *HDFS client*, eksportującej interfejs systemu plików [9].
- Ori File System. OriFS jest bezpiecznym systemem utworzonym na platformy Linux, BSD oraz OS X. Wspiera komunikację *peer-to-peer*, pracę w trybie *offline*, transfer danych poprzez SSH i szybki dostęp z synchronizacją w tle. Daje również kontrolę nad procesem replikacji i rozwiązywaniem konfliktów [10]. OriFS dostarcza interfejsy POSIX oraz interfejs CLI do administrowania [6].

3. Środowisko i metodyka badań

Opracowane środowisko sprzętowe (rys. 2) składa się z serwera wirtualizacji IBM x3650, wyposażonego w dwa procesory Intel Xeon E5420 oraz 32 GB RAM. Magazynem danych jest macierz składająca się z 8 dysków IBM SAS 146 GB 10k włączonych do kontrolera RAID *Adaptec AACRaid*. Posiada ona dwie karty sieciowe Intel i350 włączone do dwóch przełączników wykorzystując połączenia wiązane 802.3ad.

Dodatkowym magazynem danych jest macierz *IBM System Storage* DS3400 wyposażona w 12 dysków 146gib 15k SAS. Do połączenia z serwerem wirtualizacji wykorzystano dwie dwuportowe karty Fibre Channel QLogic ISP2432. Wielokrotne połączenie skonfigurowano przy pomocy *DM-Multipath* wykorzystując algorytm *service-time* do wyznaczania tras *(*ang. *path-selector)*.



Rys. 2. Diagram struktury środowiska badań

System wirtualizacji oparto na platformie zarządzania maszynami wirtualnymi Ovirt [7]. Jest to otwarty, scentralizowany system zarządzania również wirtualnymi sieciami oraz zasobami dyskowymi sponsorowany przez

Red Hat. Ovirt składa się z dwóch komponentów: Ovirt Node oraz Ovirt Engine. Ovirt Node to fizyczny serwer działający pod dystrybucjami Centos, Fedora bądź RHEL wykorzystujący do wirtualizacji hipernadzorcę KVM [6]. Do kontroli zasobów wykorzystywany jest demon VDSM [11], który zarządza procesem tworzenia maszyn wirtualnych, zasobami sprzętowymi oraz czynnościami administracyjnymi. Z kolei Ovirt Engine to fizyczny bądź wirtualny serwer zarządzający zbiorem węzłów. Dostęp dla użytkownika zrealizowany jest poprzez panel webowy napisany w oparciu o *GTW toolkit* działający na serwerze WildFly. Do przeprowadzenia badań, z racji ograniczonych zasobów, wykorzystano możliwość uruchomienia komponentu *Engine* jako maszyny wirtualnej. Taka konfiguracja nosi nazwę *Self-hosted Engine* [11].

W przygotowanym środowisku badawczym każdemu systemowi plików dedykowanych jest pięć maszyn wirtualnych. Początkowo wykonywano badania dla klastrów składających się z dwóch węzłów, następnie powiększono klaster o kolejne trzy i powtórzono badania. Systemem operacyjnym na każdej z maszyn jest CentOs 7, ze względu na dostępność instalacji każdego z systemów plików na tej dystrybucji oraz jej stabilność i dostępność grup wsparcia. Każda z maszyn ma dodatkowo nałożone ograniczenia mające za zadanie zamodelować realne działanie DFS oraz jego obciążenia:

- wydajność dysków danych – ograniczająca prędkość zapisu i odczytu do 50 MB/s,
- przepustowość karty sieciowej ograniczająca transfer do 1 Gb/s.

Przedmiotem badań był następujący problem: Administrator posiada 5 komputerów wyposażonych w 1 dysk systemowy o pojemności 30 GB, 2 dyski o pojemności 10 GB, jeden rdzeń procesora E5420, kartę sieciową 1 Gb/s oraz 2 GB pamięci RAM. Jaki rozproszony system plików należy wybrać, aby osiągnąć maksymalną wydajność?

Badania wykonano dla systemów złożonych kolejno: z dwóch i pięciu węzłów oraz pięciu węzłów zamontowanych za pomocą mechanizmu *kernel driver*. Podczas implementacji systemów zapewniono maksymalne podobieństwa konfiguracji. Nie uwzględniano dodatkowego opóźnienia wprowadzonego przez pośrednie komponenty sieciowe. Każdy z systemów został zamontowany na maszynach klienckich za pomocą modułu jądra FUSE. Jeśli było to możliwe, włączano opcję synchronizowania buforów, niekorzystania z pamięci podręcznej oraz wykonywania operacji w trybie synchronicznym. Każdy z systemów plików badano dla więcej niż jednego tylko trybu działania:

- System Ceph zaimplementowano dla dwóch i pięciu maszyn (węzłów) – znaczone dalej jako *Ceph fuse* $n = \{2 \mid 5\}$ oraz *Ceph kernel driver*.
- LizardFS badano w trakcie pracy na dwóch i pięciu maszynach dla dwóch trybów replikowania – standardowego oraz xor. Dalej oznaczane są one jako *LizardFS xor replica* $n = \{2 \mid 5\}$, *LizardFS standard replica* $n = \{2 \mid 5\}$.

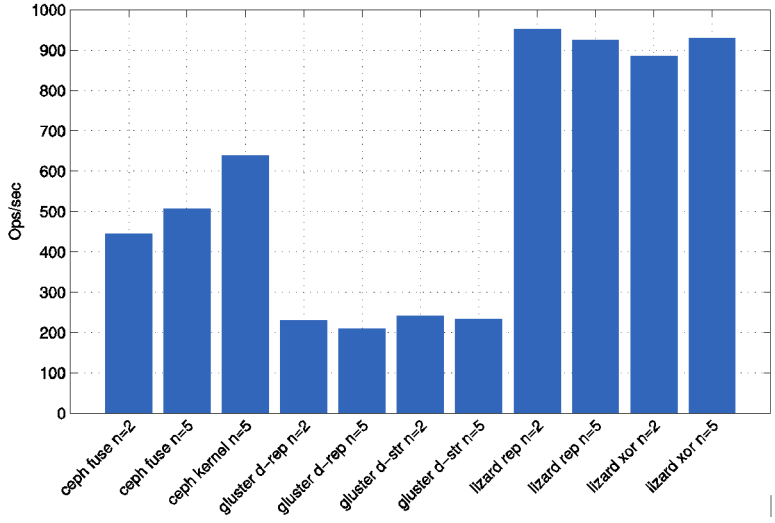
- GlusterFS badano na dwóch i pięciu maszynach, w dwóch trybach działania - *Distributed Replicated Volume* i *Distributed Striped Volume - GlusterFS striped n = {2 | 5}*, *GlusterFS replicated n = {2 | 5}*.

Dzięki dobraniu odpowiedniego sposobu montowania, możliwe jest przebadanie rozproszonych systemów plików za pomocą testów wzorcowych (ang. *benchmark*) dedykowanych dla klasycznych systemów plików. Dzięki ich zastosowaniu możliwe jest zmierzenie wartości określających wydajność każdego z systemów. W tym celu skorzystano z następujących narzędzi:

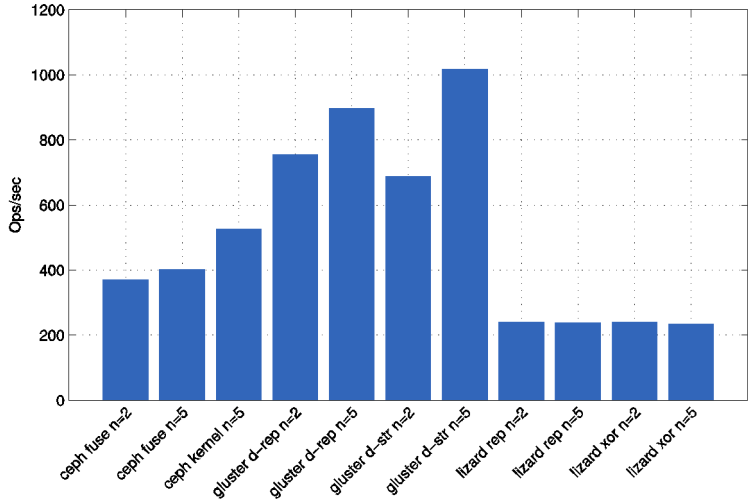
- ***Ioping***. Narzędzie do monitorowania czasu odpowiedzi przy wykonywaniu operacji wejścia/wyjścia na dysku. Rezultatem przeprowadzenia badania jest zależność średniego czasu odpowiedzi, transferu oraz liczby operacji wejścia/wyjścia na sekundę (IOPS) od wielkości wysłanego zapytania.
- ***Iozone***. Może generować i mierzyć rezultaty wielu operacji, takich jak odczyt (*read*), zapis (*write*), ponowny zapis (*re-write*), ponowny odczyt (*re-read*), losowy odczyt (*random read*). Ten test wzorcowy ma za zadanie profilować systemy plików pod kątem radzenia sobie z aplikacjami o różnej charakterystyce zapisu i odczytu i został wybrany ze względu na możliwość badania z wykorzystaniem dużych plików, trybu automatycznego oraz trybu badania z użyciem wielu wątków procesora. Badano następujące zależności:
  - przepustowość w zależności od liczby wątków, rodzaju operacji, wielkości bloku i rekordu,
  - transfer (w [kB/sec]) w zależności od wielkości pliku i wielkości rekordu, wykonywane dla każdej dostępnej operacji.
- ***Bonnie++***. Jest programem do testowania wydajności dysku. Wybrany został ze względu na możliwość badania wydajności operacji na metadanych. Z jego wykorzystaniem przeprowadzono badania: szybkości operacji zapisu i odczytu, liczby operacji wyszukiwania wykonywanych w jednostce czasu, liczby operacji na metadanych wykonywanych w jednostce czasu.
- ***FIO***. *Flexible I/O* jest to generator obciążenia, służący do pomiaru wydajności systemu plików. Generuje on obciążenie w postaci operacji zapisywania i odczytywania danych. Dokonywane są pomiary: sumy operacji I/O, transferu, czasu działania, zużycia CPU i inne.

4. Wyniki badań

Wydajność operacji sekwencyjnego tworzenia i usuwania plików przedstawiono na rys. 3, natomiast wydajność operacji wyszukiwania i obciążenie procesora tą operacją na rysunku 4. Badania przeprowadzono za pomocą narzędzia Bonnie++.



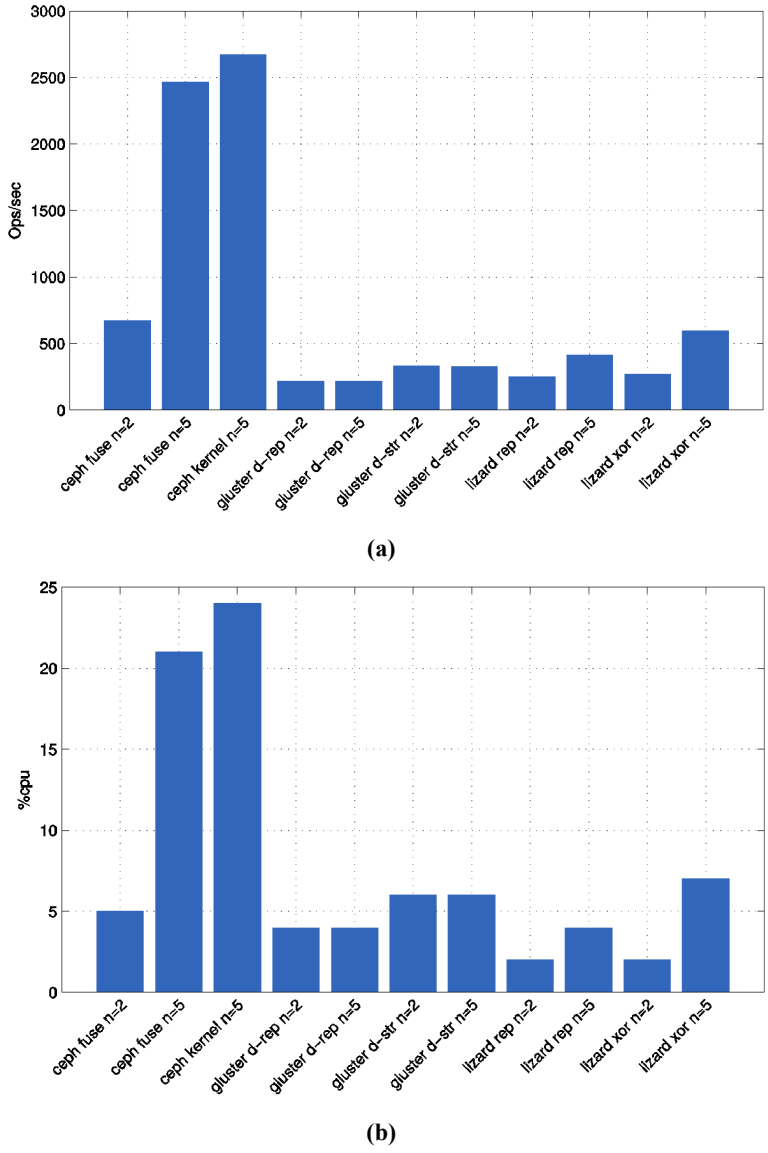
(a)



(b)

Rys. 3. Wydajność sekwencyjnego tworzenia (a) i usuwania (b) plików

System LizardFS wykonuje zdecydowanie najwięcej operacji sekwencyjnego tworzenia plików w jednostce czasu i jednocześnie najwolniej sekwencyjnie usuwa pliki – rysunek 3. Nie stwierdzono istotnej dysproporcji liczby operacji pomiędzy implementacjami o różnej liczbie węzłów. W przeciwieństwie do LizardFS, GlusterFS szybciej usuwa pliki niż je tworzy.



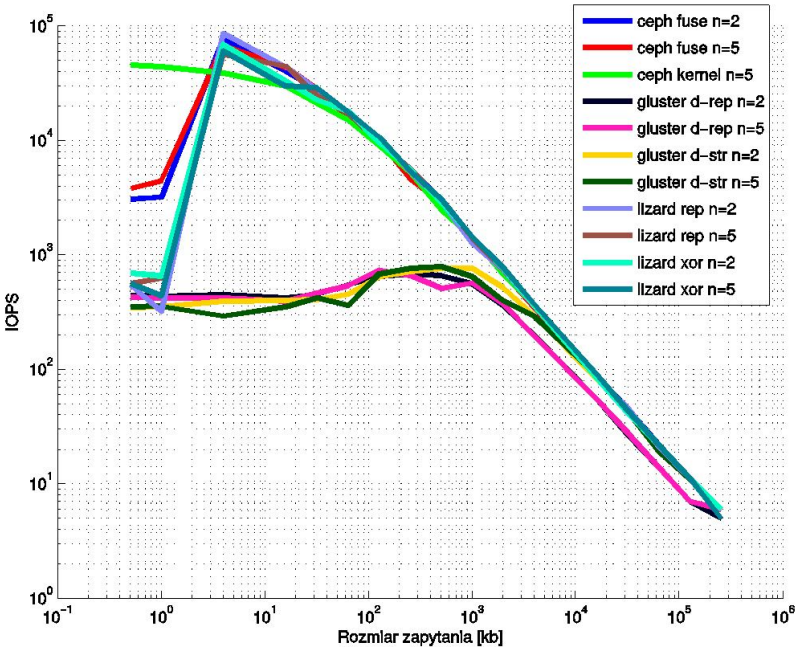
Rys. 4. Wydajność (a) i zużycie zasobów procesora (b) przy operacji wyszukiwania

System Ceph ma najmniejszą ze wszystkich rozbieżność pomiędzy liczbą operacji tworzenia i usuwania, jednak nie jest to różnica pomijalna. Wśród wersji konfiguracji systemu Ceph, najlepiej sprawdza się *Ceph kernel driver*. Wydajność losowego tworzenia plików, pod względem liczby operacji na sekundę nie różni się niemal wcale od sekwencyjnego tworzenia plików dla wszystkich systemów.

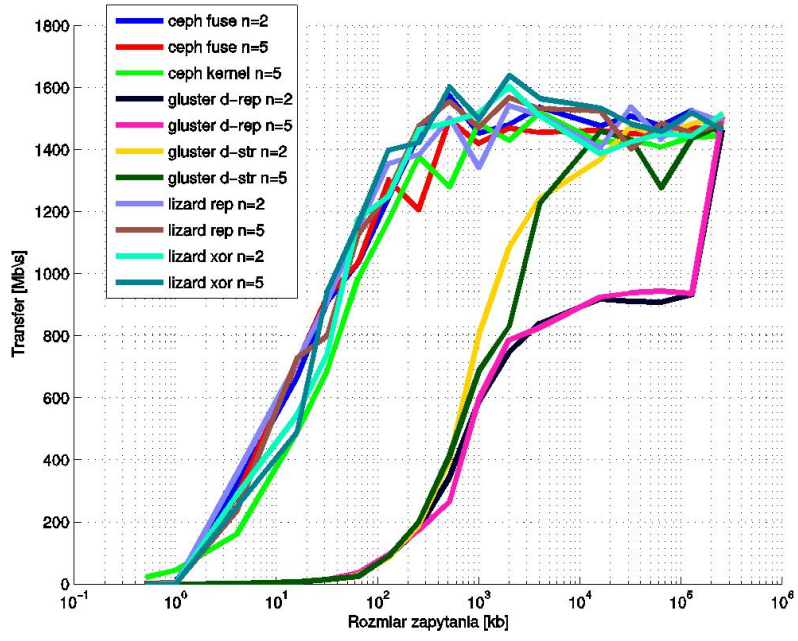
Liczba operacji wyszukiwania na sekundę (rys. 4a) na fizycznym dysku zależy od jego rodzaju, gdyż w przypadku dysków HDD każda operacja wymaga przesunięcia głowicy nad wskazany cylinder. Zużycie procesora (rys. 4b) jest proporcjonalne do liczby wykonanych wyszukań. Najwięcej operacji potrafi wykonać Ceph w konfiguracji *kernel driver*, wśród FUSE również najwydajniejszy jest Ceph. W przypadku tego systemu plików, im więcej węzłów, tym więcej wykonanych operacji i większe zużycie procesora.

Za pomocą narzędzia *Ioping* zbadano zależność liczby wykonywanych operacji wejścia-wyjścia (IOPS) oraz szybkości transferu danych w zależności od rozmiaru zapytań (żądań). Badania przeprowadzono dla dwóch scenariuszy: z wykorzystaniem pamięci podręcznej oraz bez niej. Wyniki eksperymentów przedstawiono na rysunkach 5-7. W badaniu zależności wydajności od rozmiaru zapytania zaobserwować można, że badane systemy, poza GlusterFS, korzystają efektywnie z pamięci podręcznej (rys. 5). Wywnioskować można, że obsługa pamięci podręcznej po stronie klienta jest kosztowniejsza niż w przypadku innych systemów. Dla wszystkich systemów liczba operacji realizowanych w jednostce czasu zaczyna spadać, gdy wielkość zapytania wzrośnie powyżej 100 kb (rys. 5a i rys. 6). Zdecydowanie najwięcej operacji dla wzrastającego rozmiaru zapytania mają systemy *Ceph fuse*, niezależnie od liczby węzłów. Różnią się one od pozostałych systemów o rząd, co jest znaczącą różnicą. Generalizując, w tym badaniu liczba węzłów ma znikomy wpływ na liczbę wykonanych operacji – wartości dla implementacji tych samych systemów w różnych skalach w przybliżeniu się pokrywają. Nie występuje przewaga *Ceph kernel driver*, jest on jedynie niewiele wydajniejszy od LizadFS i GlusterFS.

W badaniu zależności transferu od rozmiaru zapytania (rys. 5b i rys. 7) najlepiej sprawdza się *Ceph fuse*. Dla LizardFS (całość) oraz *Ceph kernel driver* transfer jest niemalże jednakowo niski niezależnie od rozmiaru zapytania. Może to świadczyć nie tyle o powolności systemu, co o braku przechowywania danych w pamięci podręcznej, gdyż ograniczenia nałożone na karty sieciowe pozwalają osiągnąć maksymalny transfer 120 MB/s. Odzwierciedla to wielkość transferu na wykresie dla wspomnianych systemów. Analizując przebieg zależności wielkości transferu od rozmiaru zapytania dla wszystkich implementacji Ceph i GlusterFS, można zauważyć, że jest on zastanawiająco wysoki, wykraczający poza nałożone ograniczenia. Pozwala to stwierdzić, że jest to efekt prawdopodobnie działania pamięci podręcznej po stronie klienta. Tezę tę potwierdza znaczący spadek transferu zarówno dla GlusterFS jak i Ceph, sprowadzający wartość transferu do poziomu osiągniętego przez LizardFS. Dla każdego systemu plików moment zakończenia pracy w pamięci podręcznej jest różny. Powyżej pewnej wielkości zapytania wszystkie systemy plików osiągają wydajność równą przepustowości karty sieciowej.

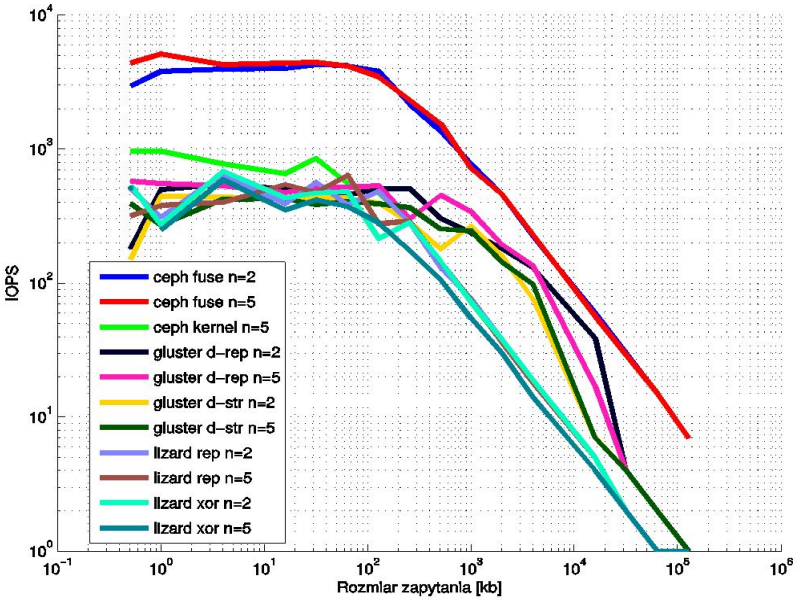


(a)

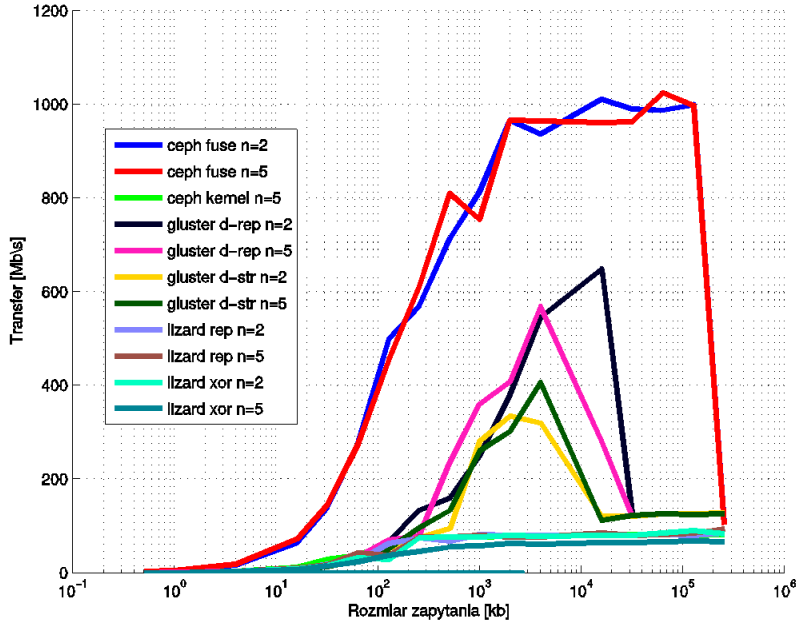


(b)

Rys. 5. Zależność wydajności i transferu od rozmiaru zapytania (z pamięcią podręczną)



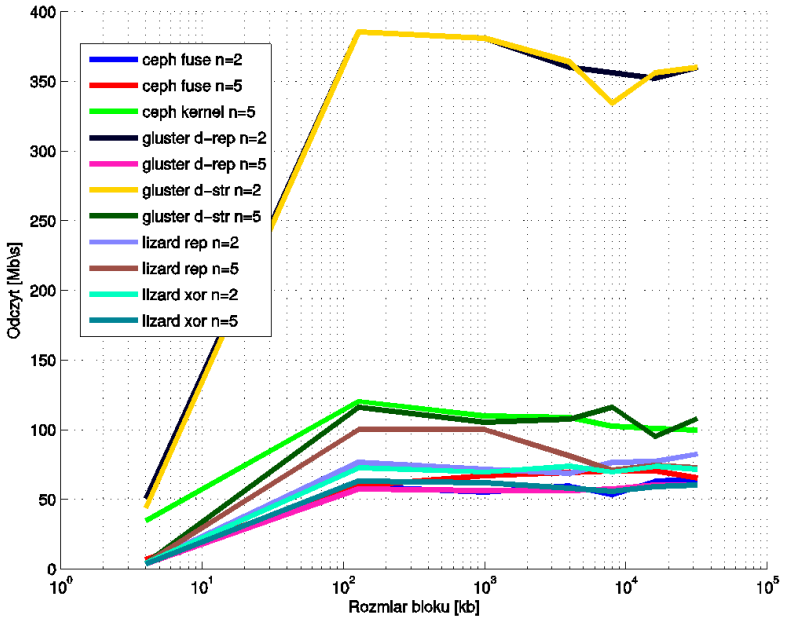
Rys. 6. Zależność wydajności od rozmiaru zapytania (bez pamięci podręcznej)



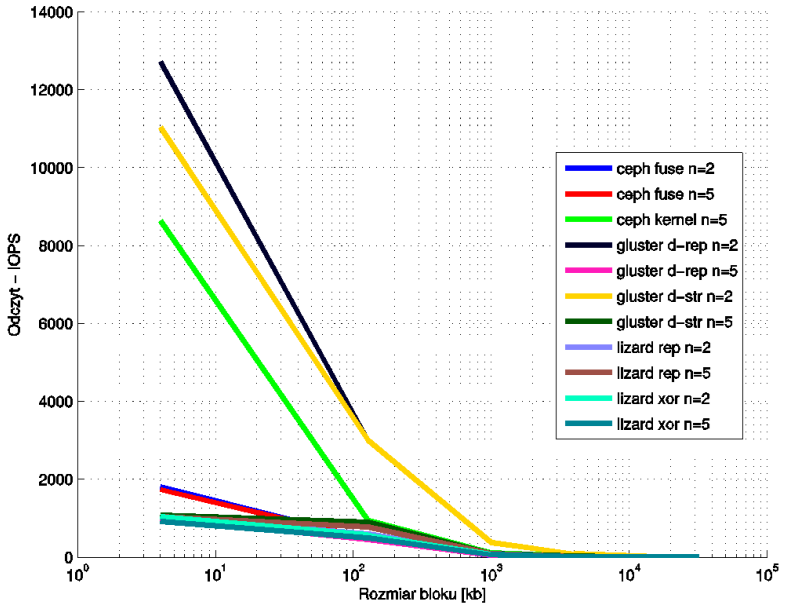
Rys. 7. Zależność transferu od rozmiaru zapytania (bez pamięci podręcznej)

Analizę wpływu wielkości bloków dyskowych (*block size*) na wydajność odczytu i zapisu przeprowadzono z wykorzystaniem narzędzia *FIO*. Wyniki eksperymentów przedstawiono na rysunkach 8 i 9. W badaniu przepustowości odczytu dla operacji losowego odczytu (rys. 8) znacząco lepsze od innych systemów wyniki uzyskano dla *GlusterFS replicated* i *GlusterFS striped*, ale jedynie dla dwóch węzłów. Oczywiście jest to efekt zastosowania pamięci podręcznej, ale trudno wytłumaczyć fakt, czemu to samo zjawisko nie zachodzi dla pięciu węzłów. Wydajność odczytu dla największego badanego rozmiaru bloku nie spada, co oznacza, że nie osiągnięto limitu pamięci podręcznej. Obserwując przebieg wykresu zależności liczby IOPS od wielkości bloku, *GlusterFS replicated* i *GlusterFS striped* nadal wykorzystują pamięć podręczną. Prawdopodobnie używa jej również *Ceph kernel driver*. Pozostałe systemy i ich implementacje utrzymują wydajność odczytu na podobnym poziomie. Interesującą obserwacją jest fakt, że *GlusterFS replicated* i *GlusterFS striped* zachowują się dla operacji odczytu sekwencyjnego niemal identycznie jak dla odczytu losowego. Analizując wyniki badania wydajności zapisu dla operacji zapisu sekwencyjnego i losowego również można zaobserwować znaczące podobieństwa (rys. 9). Jedynie Ceph *kernel driver* odznacza się wydajnością mniejszą o niemal połowę. Największy transfer zmierzono dla *LizardFS xor replica* dla pięciu węzłów, ale prędkość zapisu jest dużo niższa dla dwóch węzłów. Obserwacja ta każe się zastanowić nad istnieniem mechanizmu pamięci podręcznej lub innego rodzaju bufora zapisu po stronie serwera. Dla *LizardFS standard replica* ta zależność również się pojawia, choć nie tak wyraźna, z kolei dla *Ceph fuse* nie występuje. Porównując liczbę operacji wejścia/wyjścia widać znaczące różnice w zachowaniu poszczególnych systemów plików. Przy losowym zapisie najwięcej IOPS wykonują *GlusterFS replicated* i *GlusterFS striped* dla dwóch węzłów, natomiast dla operacji zapisu sekwencyjnego relacja jest odwrotna - najwięcej operacji wykonują *GlusterFS replicated* i *GlusterFS striped* dla pięciu węzłów.

Podczas kolejnych eksperymentów badano zależności szybkości odczytu i zapisu od liczby wątków wykonujących dane operacje. Na rysunku 10 przedstawiono zależność podczas wykonywania badania metodą mieszanego obciążenia (*mixed workload*), polegającą na dokonywaniu zapisów i odczytów w losowych miejscach pliku. Cenną obserwacją jest stwierdzenie, że dla większego rekordu operacje wykonywane są około 10 razy szybciej przez wszystkie systemy plików. Operacja mieszanego obciążenia dla małego rekordu wykonywany jest najszybciej przez *Ceph kernel driver* dla 5 wątków, natomiast dla większego przez *GlusterFS striped n=2*, dla również 5 wątków. Uśredniając, GlusterFS najgorzej radzi sobie dla małego rekordu, za to najlepiej dla dużego.

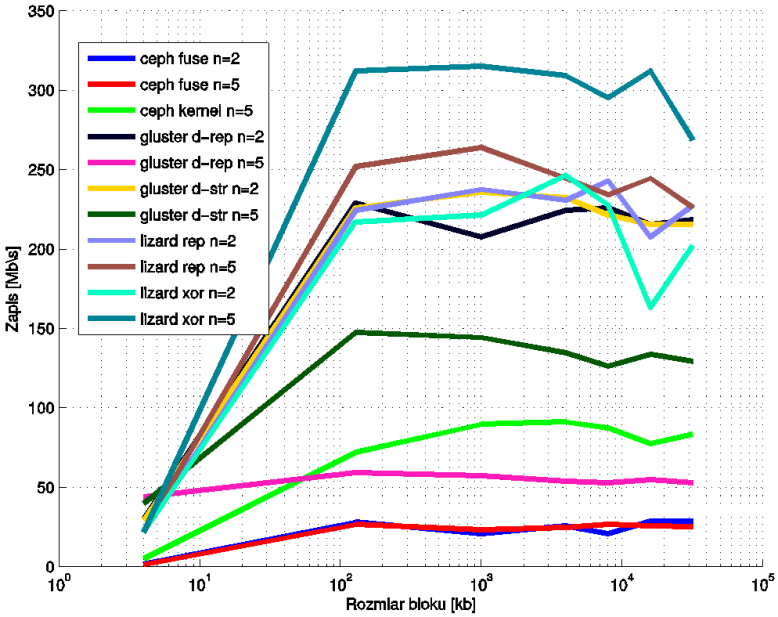


(a)

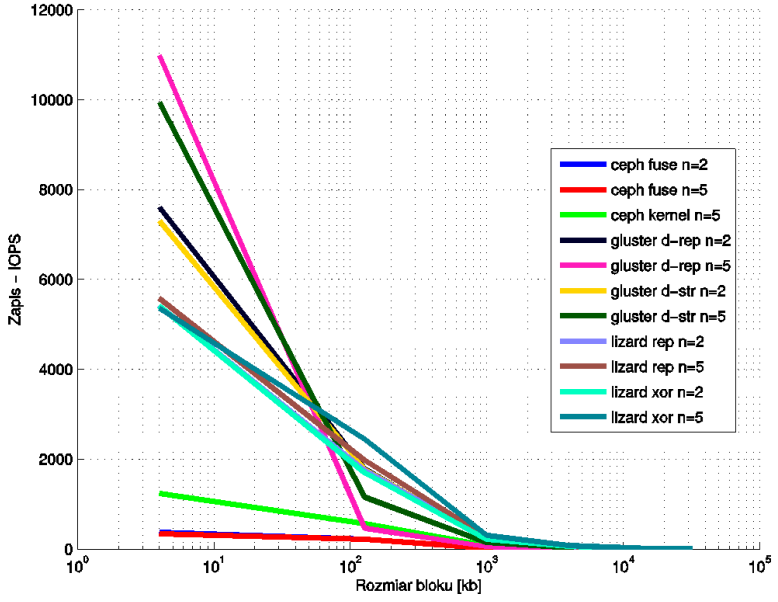


(b)

Rys. 8. Wydajność odczytu w zależności od rozmiaru bloku przy operacji losowego odczytu

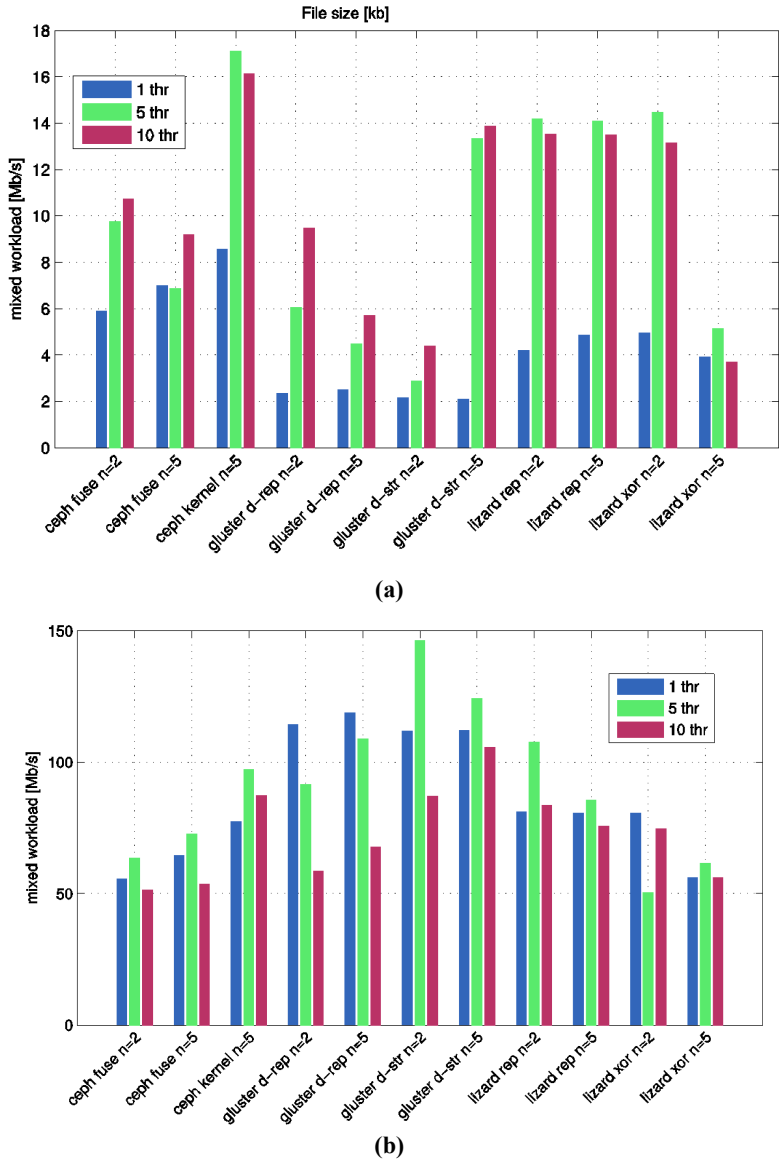


(a)



(b)

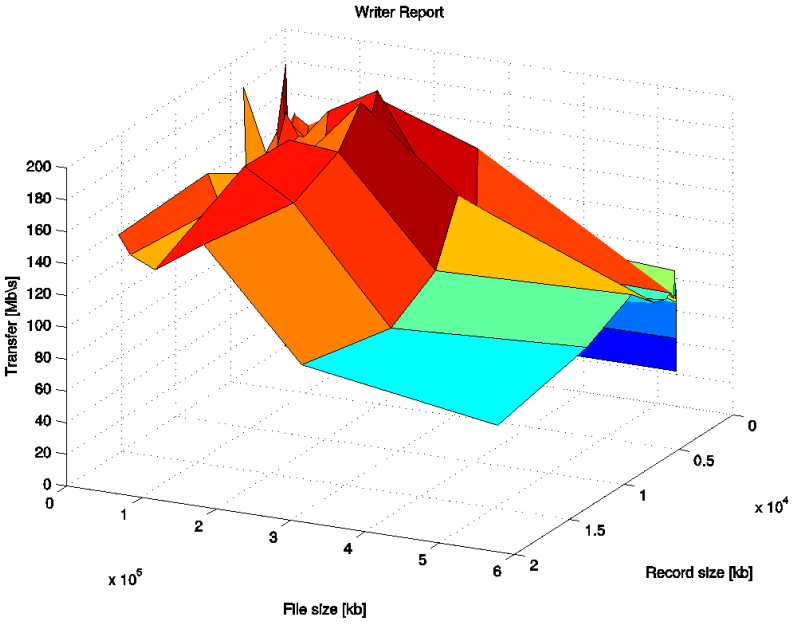
Rys. 9. Wydajność zapisu w zależności od rozmiaru bloku przy operacji sekwencyjnego zapisu



Rys. 10. Wydajność wykonywania operacji mieszanego obciążenia dla rozmiaru pliku 512 kb i rekordu 4 kb (a) oraz wielkości pliku 32 MB i rekordu 16384 kb (b)

Na rysunku 11 przedstawiono zależność transferu od rozmiaru pliku oraz rozmiaru rekordu podczas wykonywania operacji sekwencyjnego zapisu w systemie LizardFS. Zaobserwowano ciekawą zależność – niezależnie od rozmiaru rekordu wyraźnie widać dla jakiego rozmiaru pliku kończy się działanie pamięci podręcznej. Taki spadek cechuje wszystkie badane systemy. Następuje

wyraźny spadek początkowo bardzo wysokiego transferu dla pliku o wielkości $2*10^5$ kb. Następnie, wraz ze wzrostem rozmiaru pliku, transfer stabilizuje się.



Rys. 11. Zależność transferu od wielkości pliku i rekordu dla *LizardFS standard replica n=5* podczas wykonywania operacji sekwencyjnego zapisu

5. Podsumowanie i wnioski

Ceph jest zdecydowanie najbardziej „zrównoważonym" systemem plików, radzącym sobie w zadowalający sposób we wszystkich badaniach. W badaniach za pomocą *Bonnie++* widać że Ceph szybciej, przeciętnie o połowę, od innych systemów wykonuje operację pobrania bloku danych, co może przekładać się w praktyce na szybszy dostęp do danych. W porównaniu do innych systemów stosunkowo powoli wykonuje sekwencyjne operacje tworzenia i usuwania plików. Może to być spowodowane mechanizmem związanym z dziennikowaniem danych i faktem, że sam dziennik nie znajduje się na osobnym, fizycznym dysku SSD. Sama konieczność takiej fizycznej realizacji negatywnie wpływa na odbiór systemu, ale kwestię tę rozwiązuje technologia *backend BlueStore*. W testach wzorcowych *Ioping*, *Ceph fuse* zdecydowanie przoduje, we wszystkich badaniach uzyskując najlepsze rezultaty. Wraz ze wzrastającym rozmiarem zapytania potrafi wykonać najwięcej operacji w jednostce czasu, odpowiada na zapytania najszybciej oraz charakteryzuje się największym

transferem (wszystko, dopóki jest w użyciu pamięć podręczna). Nie uwidacznia się tutaj przewaga konfiguracji *kernel driver*, którego wyniki są podobnie do pozostałych rozproszonych systemów plików.

Cenną obserwacją dotyczącą nie tylko Ceph, ale wszystkich badanych systemów wypływającą z badań za pomocą *FIO* jest brak zysku z mechanizmu odczytu z wyprzedzeniem (ang. *readahead*), polegającego na sekwencyjnym odczytaniu większej ilości danych niż wskazuje zapytanie. Mechanizm ten powinien powodować dużo lepszą wydajność dla operacji sekwencyjnego odczytu. Tymczasem porównując wykresy operacji losowego i sekwencyjnego odczytu prawie wcale się one od siebie nie różnią. Należy jednak zauważyć, że generalnie w testach zapisu i odczytu za pomocą *FIO*, Ceph wypada najsłabiej (IOPS i prędkość). Jak wszystkie systemy, Ceph nie najlepiej radzi sobie z małymi plikami. W Ceph pojawiają się wąskie gardła, uwidaczniające się w operacji zapisu dla dużego rekordu oraz przy mieszanym obciążeniu, choć jest ich relatywnie najmniej. Uśredniając, Ceph osiąga najlepsze wyniki w operacjach wykonywanych wieloma wątkami.

Cechą charakterystyczną LizardFS jest zrównoważona praca niezależnie od liczby węzłów jak i trybu replikacji. Z niemal równą szybkością LizardFS wykonuje operację pobrania i zapisania bloku danych. z czego LizardFS *xor replica* jest odpowiednio wolniejszy i szybszy. W odróżnieniu od GlusterFS, ten system szybko sekwencyjnie tworzy pliki, ale powoli je usuwa. Zużywa również dużo procesora klienta przy okazji operacji sekwencyjnego pobrania metadanych (11%), wykonując stosunkowo mało operacji. W badaniu losowego tworzenia i usuwania plików LizardFS jest najszybszy spośród badanych systemów – wykonuje najwięcej operacji w jednostce czasu. Przy okazji badania zużycia procesora przy losowym pobieraniu metadanych, ponownie zużywa najwięcej zasobów procesora (5%), jednocześnie wykonując najwięcej operacji. LizardFS jest najlepszy w badaniu z użyciem *FIO* w operacjach zapisu (sekwencyjnego i losowego). Badanie za pomocą *Iozone* ujawniło, że w LizardFS również pojawiają się wąskie gardła. Nie ma dużej różnicy między implementacjami o różnych trybach replikowania.

GlusterFS przeważnie radzi sobie porównywalnie jak pozostałe systemy, lecz w mniej zrównoważony sposób - gorzej przy tworzeniu losowych plików, lepiej dla np. operacji zapisu bloku danych. Charakteryzuje się ogólnie niskim zużyciem procesora klienta, lecz jak widać w badaniu liczby operacji wyszukiwania jest to skorelowane z niewielką liczbą operacji wykonanych w jednostce czasu. Szczególnie dużą asymetrię widać w badaniach liczby operacji przy sekwencyjnym tworzeniu i usuwaniu plików. GlusterFS szybko tworzy pliki – szczególnie dla implementacji o 5 węzłach, lecz powoli je usuwa. Stąd można wnioskować, że np. operacja kopiowania dużych plików do innej lokalizacji może być realizowana bardzo powoli. Bardzo efektywny mechanizm pamięci podręcznej ujawnia się we wszystkich badaniach wykonanych za pomocą *FIO*. GlusterFS prezentuje dużą dysproporcję w wydajności wykonywania różnych

operacji. Co więcej, w systemie tym pojawia się najwięcej wąskich gardeł, między innymi w operacjach zapisu, odczytu i mieszanego obciążenia. Zaskakujący jest też bardzo słaby rezultat w badaniu odczytu z wyprzedzeniem, w którym GlusterFS jest dziesiątki razy wolniejszy od pozostałych systemów. Dużo wolniej działa *GlusterFS replicated*, niż *striped*. Cecha ta pokazuje się zarówno przy operacji odczytu jak i zapisu.

Wszystkie badane systemy źle sobie radzą z operacjami na małych plikach. W każdym z nich transfer przestaje w pewnym momencie zależeć od rozmiaru pliku i rekordu, co jest niezmienne względem liczby maszyn na jakich działają systemy. Cechą ogólną jest również wolniejsze wykonywanie operacji zapisu niż odczytu. Wszystkie narzędzia testów wzorcowych pracowały z włączonymi opcjami synchronizowania buforów, bez pamięci podręcznej, z wykonywaniem operacji w trybie synchronicznym, a mimo tego działanie mechanizmów pamięci podręcznej jest bardzo widoczne.

Analizując wszystkie wyniki oraz inne cechy badanych rozproszonych systemów plików, można stwierdzić, że najlepiej do ogólnych zastosowań nadaje się Ceph. Jako system projektowany do zastosowania w dużych centrach komputerowych, jest najwydajniejszy i jednocześnie posiada najwięcej możliwości dopasowywania go do konkretnych zastosowań. W badaniu został zastosowany w możliwie najmniej wydajny sposób - zamontowany za pomocą FUSE, z dziennikiem znajdującym się fizycznie na tym samym dysku, a mimo tego osiąga najlepsze rezultaty. Nadaje się zarówno do małych jak i dużych aplikacji, więc jest to odpowiedź na problem wydajności postawiony w tezie pracy.

Literatura

[1] BŽOCH P., ŠAFAŘÍK J., *State of the Art in Distributed File Systems: Increasing Performance*, 2011 Second Eastern European Regional Conference on the Engineering of Computer Based Systems, Bratislava, 2011, pp. 153-154.

[2] DEPARDON B., MAHEC G. L., SEGUIN C., *Analysis of Six Distributed File Systems*. Research Report <hal-00789086>, 2013.

[3] GHEMAWAT S., GOBIOFF H., LEUNG S-T., *The Google File System*. Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, October 2003, Bolton Landing, NY, USA, pp. 29-43.

[4] SHVACHKO K., KUANG H., RADIA S., CHANSLER R., *The Hadoop Distributed File System*. IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, 2010, pp. 1-10.

[5] WEIL S. A., BRANDT S. A., MILLER E. L., LONG D. D. E., *Ceph: A scalable, high-performance Distributed File System*. Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, 2006, pp. 307-320.

[6] MASHTIZADEH A. J., BITTAU A., HUANG Y. F., MAZIÈRES D., *Replication, history, and grafting in the ori file system.* Proceedings of the 24th Symposium on Operating Systems Principles (SOSP), 2013, pp. 151–166.

[7] SATYANARAYANAN M., *A survey of distributed file systems*. Annual Review of Computer Science, Vol.4, 1989, pp.73-104.

[8] WILLIAMS E. H. ET AL., *The Andrew File System on OS/2 and SNA*. Proceedings of TRICOMM '91: IEEE Conference on Communications Software: Communications for Distributed Applications and Systems, Chapel Hill, NC, 1991, pp. 181-191.

Źródła elektroniczne

[9] HDFS Architecture Guide, https://hadoop.apache.org/.

[10] Strona główna projektu OriFS, http://ori.scs.stanford.edu/.

[11] Self Hosted Engine, http://www.ovirt.org/.

[12] Dokumentacja Ceph, http://docs.ceph.com/docs/master/.

[13] Dokumentacja GlusterFS, http://gluster.readthedocs.io/en/latest/.

[14] Dokumentacja LizardFS, https://lizardfs.com/.

Properties analysis of distributed file systems

ABSTRACT: In this paper we present a comparative analysis of three most popular distributed file systems: Ceph, LizardFS nad GlusterFS. The main evaluation criterion was the efficiency of systems, meant as the number of read and write operation per second and maximum data transfer rate. The analysis is based on results of many experiments conducted in experimental testbed, prepared for this purpose. Experimental part of the paper is preceded with a survey of technology related to most important distributed file systems. The results of experiments have been presented and analyzed, also some practical conclusions have been formulated.

KEYWORDS: distributed file systems, virtualization, cloud computing, Ceph, LizardFS, GlusterFS

Praca wpłynęła do redakcji:17.07.2017 r.