Vol.2011-HPC-129 No.8
2011/3/15

Ceph のメタデータサーバの冗長性の調査

大西 健太[†1] 建部 修見[†2]
西谷 明彦[†3] 大岸 智彦[†3]

本稿では，大規模環境向けの分散ファイルシステムの 1 つである Ceph を対象として，メタデータの入出力性能の測定，およびメタデータサーバの耐故障性の検証を行った．メタデータサーバを 1 台と 2 台の場合でメタデータの作成・読込・削除の性能を測定した結果，作成・読込において大きな差は認められなかったが，削除ではメタデータサーバの増加によって約 5 割から 9 割の低減が生じた．耐故障性の検証ではメタデータサーバを意図的に故障させ，故障前後で得られるメタデータを比較した．その結果，一致する場合としない場合が確認された．

Research for redundancy of Ceph's metadata servers

Kenta Ohnishi,[†1] Osamu Tatebe,[†2]
Akihiko Nishitani[†3] and Tomohiko Ogishi [†3]

In this paper, we evaluate performance of metadata I/O and fault tolerance of metadata servers in Ceph. Ceph is one of a petabyte scale distributed filesystem. First: we determine each amount of time for metadata creation, read and delete operations. These metadata operations are issued from one client. Under 2 metadata servers condition, creations and reads performance are not much difference, but delete are from 50% to 90% lower than under 1 metadata server condition. Second: we compare metadata before and after deliberate metadata server failure. As the result, both congruous and incongruous cases are preserved.

†1 筑波大学情報学群情報科学類
College of Information Science, University of Tsukuba
†2 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†3 KDDI 研究所
KDDI R&D Laboratories

1. はじめに

近年，様々な分野でデータインテンシブコンピューティングに対する需要が高まっている．物理学や天文学などの科学計算の分野では，データの大規模化が進んでおり，例えば CERN の大型ハドロン衝突加速器 (LHC) では年間約 15 ペタバイトのデータを生成している．また，個人向けのクラウドサービスの普及を受けて，今後はエンタプライズへの展開が期待されている．クラウドサービスは，ストレージなどのリソースを集約したデータセンタから，ネットワークを介してアプリケーション等をサービスとしてユーザに提供する仕組みである．

このようなアプリケーションの開発・利用の拡大に伴い，複数のプロセス間でファイルの共有を実現する分散ファイルシステムの重要性はますます増大している．この分散ファイルシステムは，一般的にクライアント・ディスクストレージ・メタデータサーバから構成される．このうち，メタデータサーバは名前空間や i ノードなどのファイル・ディレクトリそのものに関する情報 (メタデータ) を管理している．アプリケーションがファイル・ディレクトリの入出力を行う場合，クライアントはこのメタデータサーバから提供するメタデータを利用して，ファイルデータを格納しているディスクストレージへとアクセスする．現在広く使われている分散ファイルシステムの多くは，複数台のストレージと 1 台のメタデータサーバで構成されている．しかし，データセンタの大規模化が進み，この構成ではメタデータサーバが性能のボトルネックや単一障害点になるといった問題が見られ始めた．

今日のデータセンタに代表される大規模な環境向けに開発された分散ファイルシステムの 1 つとして，Ceph[1)2)] が挙げられる．本研究では，実機のサーバクラスタ上に Ceph を構成し，メタデータの入出力性能の測定とメタデータサーバの耐故障性の検証をそれぞれ行った．

本稿は 6 章で構成される．第 2 章では Ceph，特にメタデータの管理について説明する．第 3 章でメタデータの入出力性能の測定，第 4 章ではメタデータサーバの耐故障性の検証に関する実験とその評価を示す．第 5 章では関連する研究について述べ，第 6 章で結論と今後の課題を示す．

2. Ceph

2.1 Ceph の構成

Ceph は，数百ペタバイト以上の巨大なデータを数十万のユーザが利用する大規模な分散

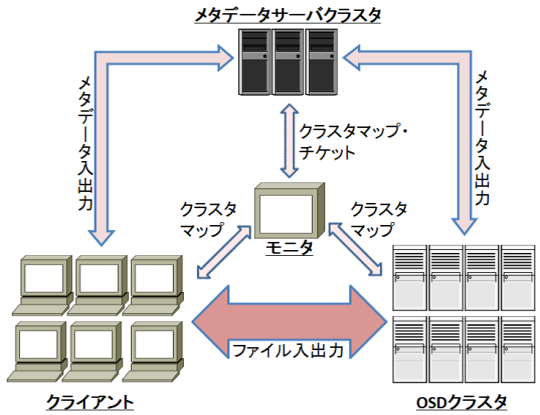

図 1 Ceph の構成

環境を想定した分散ファイルシステムである．したがって，Ceph は高スケーラビリティ・高性能・高信頼性を目指した設計となっている．図 1 に示すように，Ceph はクライアント・OSD・モニタ・メタデータサーバから構成される．

2.1.1 クライアント

クライアントは Ceph を利用する各ホスト上で動作し，POSIX[3] に準拠したインタフェースをユーザへ提供する．これによりユーザアプリケーションはローカルファイルシステムと同等なファイルの入出力が可能となる．クライアントは，メタデータサーバから取得したメタデータを用いて，OSD にファイルデータを入出力する．

またクライアントはカーネルと独立してメタデータを各々キャッシュしており，一貫性を緩める一方で高速なメタデータの入出力を実現している．

なお Ceph は FUSE[4] クライアント以外にも，Linux カーネル 2.6.34 以降で組み込まれたクライアントを利用できる．

2.1.2 OSD

OSD ではファイルデータとメタデータを安定に格納している．

OSD[5] はディスクだけではなく，CPU やネットワークインタフェース，ローカルキャッシュなどによって構成されるデバイスで，固定長のブロック単位ではなく，可変長のオブジェクト単位でデータを管理する．OSD ではオブジェクトの物理的な位置情報，i ノード番号やファイルパスとオブジェクトの対応付け，オブジェクトのバックアップや障害復旧，

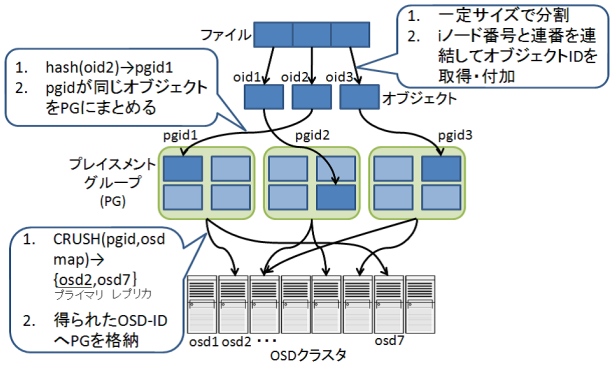

図 2 CRUSH による分散

各オブジェクトの権限の管理など，従来のローカルファイルシステムが提供していた機能を一部有している．ユーザは i ノード番号やファイルパスを与えるだけでファイルデータの入出力が可能となる．

Ceph は RADOS[6] によって OSD クラスタへのオブジェクトの分散や複製，故障検知・復旧を実現している．

図 2 に示すように，ファイルデータ (またはメタデータ) はオブジェクトに分割され，それらを PG(placement group) にまとめて，CRUSH[7] によって OSD へと分散されている．CRUSH を用いることで，各構成要素間で通信することなくオブジェクトの物理的な位置を特定できる．また複数の OSD を指定することで，PG を容易に複製ができる．レプリカ PG の更新はプライマリバックアッププロトコルによって行われる．

OSD の故障は他の OSD やモニタとの ping 等のメッセージ交換によって検知される．故障を検知すると，レプリカ PG を保持する OSD が一時的にプライマリ PG としての役割を引き継ぐ．復旧しなかった場合は，クラスタマップから除外し，その OSD が持っていたプライマリとレプリカの PG を再度分散する．復旧した場合は，最新のレプリカ PG を持つ OSD から更新ログを取得して，プライマリ PG を最新の状態にする．なお，ディスクエラー等の OSD 内での障害に関しては，各 OSD で復旧を行う．

2.1.3 モ ニ タ

モニタはクラスタ内の OSD・メタデータサーバの状態を収集し，追加・除去・故障・復旧などの変更を検知して，最新のクラスタマップをクライアント・メタデータサーバ・OSD へ

提供する．特に OSD の状態ををを表す OSD クラスタマップは CRUSH によるデータ分散で用いられる．またクライアントの起動やクライアントへの認証チケットの発行なども行う．

奇数台であれば複数のモニタで冗長化させることも可能である．

2.2 メタデータの管理

Ceph のメタデータサーバは名前空間を管理し，ファイルデータを格納している OSD の特定に必要な i ノード番号などのメタデータをメモリ上にキャッシュしている．

2.2.1 メタデータの格納

Ceph のメタデータはディレクトリエントリと i ノードからなる．各ディレクトリ内の全ファイルのメタデータは，そのディレクトリの i ノード番号を名前とする単一のディレクトリオブジェクトにまとめられ，通常のファイルデータと同様に CRUSH で OSD へと分散される．

また，図 3 に示すように，OSD はメタデータの更新履歴であるジャーナルを持つ．メタデータが更新されるとメタデータサーバのキャッシュが更新されるとともに，そのメタデータオブジェクトのプライマリを持つ OSD へ更新情報を送信する．プライマリの OSD は更新情報をジャーナルにバッファし，レプリカを持つ OSD へと同じ更新情報を送信して，複製にも更新を反映する．ジャーナルは FIFO で，追い出された更新情報はメタデータを保持するディスクへ反映し，OSD が持つメタデータを更新する．同一のメタデータに対する操作の統合や，ジャーナル中で作成・削除されたメタデータの排除によって，OSD への更新回数を低減させている．またメタデータサーバ故障時にメタデータのキャッシュを復旧する際にも使われる．

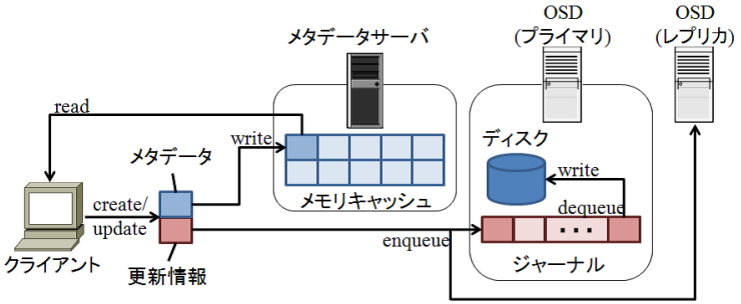

図 3 ジャーナル

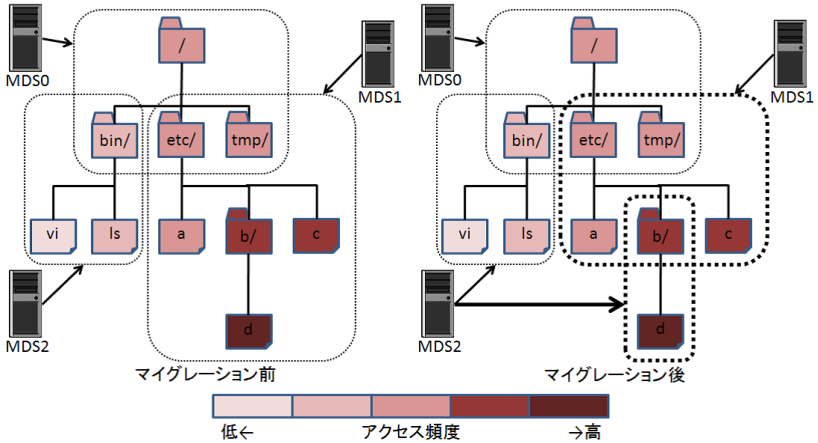

図 4 動的サブツリーパーティショニング

2.2.2 メタデータのキャッシュ

Ceph のメタデータサーバは名前空間をサブツリーに分割して，それぞれメタデータサーバへ割り当てる．各メタデータサーバは割り当てられたサブツリーが含むメタデータをメモリ上でキャッシュする．メタデータサーバはキャッシュする各メタデータの参照・更新回数をカウントしており，メタデータサーバ間でこの値を共有している．メタデータサーバ間でカウンタの値に偏り (すなわち負荷の偏り) があれば，負荷の高いメタデータサーバは管理するサブツリーを分割して，負荷の低いメタデータサーバへディレクトリごとマイグレーションして，負荷を均一化する．これを図 4 に示す．このように，現在の負荷に応じて柔軟に負荷分散している．

動的サブツリーパーティショニングではマイグレーションに伴うオーバヘッドの削減や局所性の利用のためにディレクトリ単位でメタデータサーバへの割り当て方式を採用していた．しかし，単一ディレクトリに集中した負荷を分散するため，ディレクトリ内のメタデータを異なるメタデータサーバへマイグレーションすることも可能であり，これはディレクトリフラグメントと呼ばれる．

2.2.3 故障検知・復旧

メタデータサーバは，他のメタデータサーバやモニタと定期的にメッセージを送受して故障の検知を行っている．モニタは一定時間内にメタデータサーバからメッセージが送られて

こない場合，そのメタデータサーバを故障しているとみなす．

メタデータサーバの回復を一定時間以内に確認できなければ，そのメタデータサーバをクラスタから除外し，モニタは代替のメタデータサーバを 1 台を選ぶ．

代替に選ばれた (または回復した) メタデータサーバは以下の手順で復旧が行われる．

(1) OSD 内のメタデータオブジェクトにジャーナルを適用して最新状態となったメタデータをキャッシュする．
(2) 未コミットのトランザクションを解決する．
(3) クライアントとのセッションを再確立する．
(4) 他のメタデータサーバから複製のメタデータを取得する．
(5) 更新したクラスタマップをマルチキャストして，メタデータサーバクラスタに再加入させる．

3. メタデータの入出力性能の評価

実験では Ceph サーバに 8 ノード (ノード 0 からノード 7)，Ceph クライアントに 1 ノードを用いた．サーバノードの詳細を表 1，クラスタの構成を表 2 にそれぞれ示した．Ceph サーバには Fedora パッケージを用いた．

表 1 Ceph サーバノード

CPU	Intel Xeon 3.00GHz(4 プロセッサ)
メモリ	1 GB
OS	Fedora 13(2.6.34.7)
Ceph サーバ	ceph.x86_64(0.20.2-1.fc13)

表 2 Ceph サーバクラスタ

モニタ		ノード 0
OSD		ノード 0〜ノード 7
メタデータサーバ	1 台の場合	ノード 0
	2 台の場合	ノード 1, ノード 2

上記の環境において，メタデータの作成・読込・削除の性能をそれぞれ測定した．

この実験では Bonnie++[8] を用いた．Bonnie++はファイルデータとメタデータの入出力性能をそれぞれ測定するベンチマークソフトウェアである．この実験ではまず，ディレクトリ毎にファイル名で昇順に，メタデータ (0 バイトのファイル) の作成・読込・削除を実行し，次にディレクトリ毎にランダムで，同様の操作を行う．これをメタデータサーバが 1 台と 2 台の場合で，それぞれ 5 回ずつ測定し，その平均値をメタデータの入出力性能とした．

まず，この Bonnie++を用いた予備実験として Ceph のマイグレーションを検証した．メタデータサーバ 2 台の環境においてディレクトリ数 16，各ディレクトリ内のファイル数 1024 の条件で Bonnie++を実行し，その間にノード 0 のメタデータサーバ (MDS0) とノード 1 のメタデータサーバ (MDS1) がキャッシュしていたメタデータを測定した．

得られた結果を図 5 に示す．105 秒から 1485 秒においてシーケンシャル，1485 秒から 2515 秒においてランダムでの作成・読込・削除を行った．485 秒前後において MDS0 がキャッシュするメタデータが 0 になり，代わりに MDS1 の約 16000 メタデータを新たにキャッシュしている．したがって，485 秒において MDS0 から MDS1 へ約 16000 のメタデータがマイグレーションされていると言える．

グラフを見てわかる通り，作成ではマイグレーションが発生していない．クライアントは作成時点ではメタデータに対して参照・更新を行っていないためである．また削除時においては，一方のメタデータサーバがキャッシュしているほぼ全てのメタデータを，他方のメタデータサーバへマイグレーションしている．Bonnie++の削除はディレクトリ内の全ファイルを削除した後，次のディレクトリを削除するため，ある 1 つのディレクトリの更新カウン

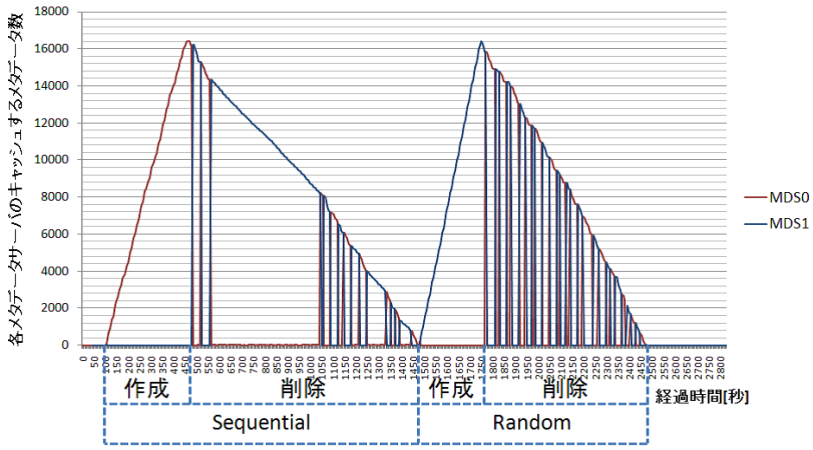

図 5 Bonnie++実行時にキャッシュするメタデータ数

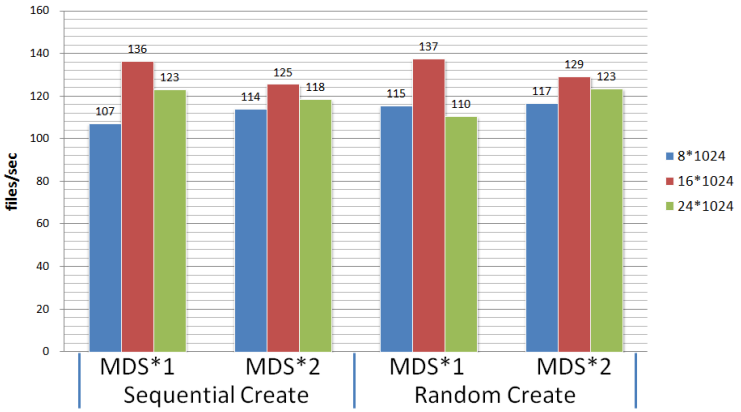

図 6 メタデータ作成性能

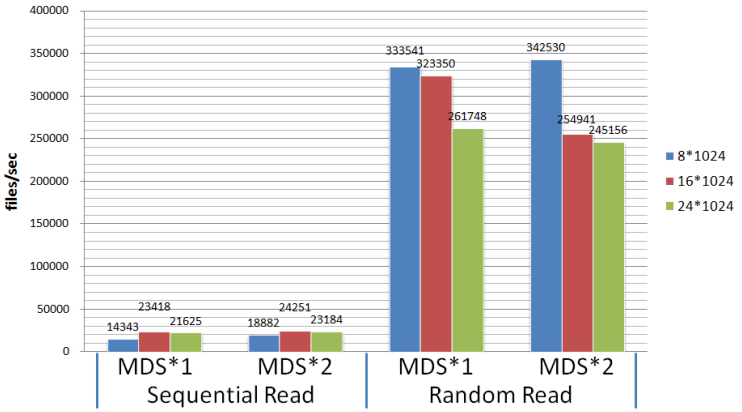

図 7 メタデータ読込性能

タのみが大きくなった．加えて，Ceph のメタデータサーバは局所性を活かすために粗いサブツリーでマイグレーションするため，参照・更新回数がほぼ 0 の他のディレクトリのメタデータとともに，マイグレーションされたと考えられる．

作成の実験結果を図 6 に示す．なお MDS*1，MDS*2 はメタデータサーバ数がそれぞれ 1 台，2 台であることを，また 8*1024，16*1024，24*1024 は，1024 個のファイルを含むディレクトリがそれぞれ 8，16，24 個であることを表している．

メタデータサーバが 1 台と 2 台の場合で，性能に大きな差はなかった．これはメタデータサーバが 2 台の場合でも，クライアントがそのメタデータに対して参照・更新のオペレーションを発行しなかったため，トランザクションは発生せず，したがってクライアントがロックされなかったためと考えられる．シーケンシャルとランダムでも大きな差はなく，ファイル数の推移に対しても性能に大きな変化は見られなかった．この場合に関しても，作成したメタデータに対してジャーナルへのコミットがクライアントと非同期に行われたためと考えられる．しかしファイル数が 16384(16*1024) の場合では他よりも高い性能を常に示しており，最大で約 27%の性能差が見られた．この原因については現在までにわかっておらず，今後検証する必要がある．

次に，読込の実験結果を図 7 に示す．

メタデータサーバの数による大きな性能差は見られなかった．ランダムとシーケンシャルの結果を比較すると，ランダムの方が 10〜23 倍高い性能を得ていることがわかる．Bonnie++ の読込テストは作成直後に行われるため，クライアントのキャッシュにメタデータが残っている場合がある．ランダムの場合では，作成された順とは関係なく読込を行うため，クライアントのキャッシュで必要なメタデータがヒットする可能性がある．しかしシーケンシャルの場合では，作成された順に読込を行うため，常にメタデータサーバとの通信が発生した．この結果，ランダムの方がシーケンシャルよりも高い入出力性能が得られたと考えられる．またランダムの場合において，ファイル数の増加に伴って約 25%の性能低下を示していることがわかる．これは，ファイル数の増加によってクライアントのキャッシュにおけるヒット率が低下したためと考えられる．

最後に，削除の実験結果を図 8 に示す．

削除においては，メタデータサーバを 1 台から 2 台に増やした場合，50〜90%の性能低下が見られた．削除に伴って発生した複製への更新やマイグレーションのトランザクションがこの原因の 1 つとして挙げられる．マイグレーション中のサブツリーやディレクトリフラグメントに含まれるファイルはロックされ，その間に発行された更新要求はそのマイグレーションが完了するまで実行されない．この結果，クライアントの待ち時間が増え，1 台の場合よりも性能が低下したと考えられる．

4. メタデータサーバの耐故障性の検証

次に，メタデータサーバの耐故障性を検証する実験を行った．

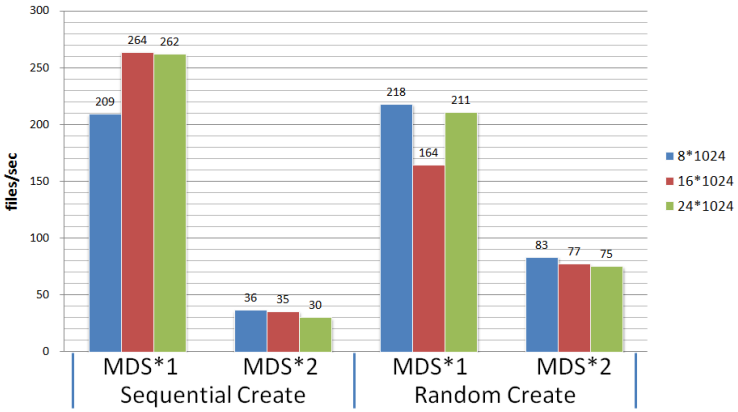

図 8 メタデータ削除性能

ここで言う耐故障性とは，メタデータサーバが故障した場合でも正しいメタデータを取得できることを意味する．そこで，図 9 に示す手順で実験を行った．このように故障前と故障後のメタデータを比較することで，メタデータサーバの耐故障性を検証した．

(1) 3 と同一の環境で，2 台のメタデータサーバを動作させる．
(2) クライアントでこの Ceph をマウントし，適当なファイル・ディレクトリを作成する．
(3) 故障させるノード 1 にマイグレーションされたファイルのメタデータを取得する．
(4) ノード 1 のメタデータサーバのプロセスを強制的に終了させ，意図的に故障させる．
(5) 同一ファイルに対するメタデータを再度取得する．
(6) 2 つのメタデータを比較する．

この実験の結果，正しいメタデータを取得できた場合と，できなかった場合が確認された．できなかった場合を次に示す．(1) と (2) はメタデータサーバの，(3) はマイグレーションの，(4) はモニタのコードの不具合だと考えられる．なお，(3) の不具合は v0.25 で修正が反映されると思われる．

(1) MDS1 の故障にともなって，MDS0 のメタデータサーバプロセスも強制終了する．
(2) MDS1 の故障にともなって，クライアントと MDS0 のコネクションが切断される．
(3) マイグレーションに失敗する．
(4) モニタが MDS1 の故障を検知しても，クラスタマップを更新しない．

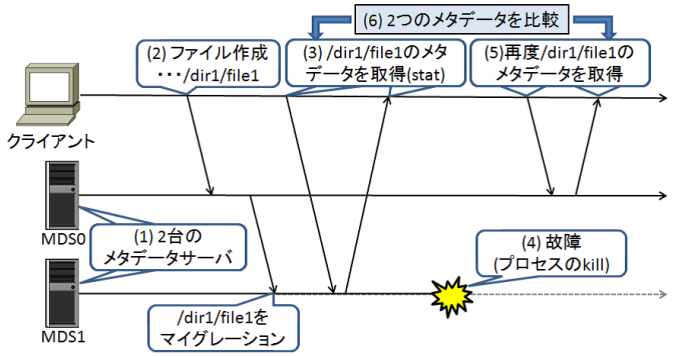

図 9 メタデータサーバの耐故障性の検証実験

5. 関連研究

5.1 XtreemFS

XtreemFS[9] は，Ceph と同様に複数のメタデータサーバと OSD で構成される分散ファイルシステムで，XtreemOS[10] プロジェクトの一環として開発された．

クライアントは，インターネットを介してディレクトリサービスからボリュームをマウントする．このボリュームのファイルやディレクトリは共有することができ，また可用性を高めるために複数の MRC へと分割・複製される．MRC はファイルやディレクトリのメタデータとそれらの複製の位置情報からなり，メタデータサーバはこの MRC をクライアントに対して提供する．

OSD ではファイルデータを保持する．ファイルデータは OSD 間で一部または全部が複製される．複製には読み込み専用と読み書き可能が指定でき，読み書き可能ではマスタースレーブ複製が自動的に行われる．

これらファイルデータとメタデータは，拠点ごとの OSD やメタデータサーバの間で複製が行われる．

インターネットを介して SSL(X.509) でマウントしたクライアントは，POSIX 準拠の API によってファイルオペレーションを発行する．まずメタデータサーバから MRC を取得し，次に複数の OSD から同時に並列して読込を行うことで，高い入出力性能・可用性を実現している．

6. おわりに

本研究では Ceph を対象とした，メタデータの入出力性能の測定と，メタデータサーバの耐故障性の検証を行った．

メタデータサーバ数を 1 台から 2 台に増やした場合，メタデータの作成・読込では大きく性能は変化しなかったが，削除において 5〜9 割の低減が確認された．これは複製の更新やマイグレーションに伴って発生したトランザクションによって，クライアントの待ち時間が発生したためと考えられる．また読込の実験では，クライアントのキャッシュがメタデータの読込性能に大きな影響を与えていることを明らかにした．

メタデータサーバの耐故障性の検証では，メタデータサーバの故障後でも正しいメタデータを取得できた．しかし，メタデータサーバやマイグレーションの不具合によって，取得できない場合が存在することも確認した．

今後の課題として，以下のことが挙げられる．

(1) 複数のクライアントによるメタデータの入出力性能の測定
今回は全て単一クライアントからメタデータの入出力を行った．しかし，実運用環境では複数のクライアントが同時にメタデータの参照・更新を行う．したがって，今後は複数のクライアントから同時にメタデータのオペレーションを発行した場合の入出力性能の測定が必要になると考えられる．

(2) 様々なアクセスパターンによる入出力性能の測定
今回はメタデータの作成・読込・削除を順に行った．しかし，実際はこの順番でオペレーションが発行されるとは限らない．したがって，いくつかの使用環境を想定したアクセスパターンにおける入出力性能の測定が必要になると思われる．

(3) 耐故障性の定量的な評価
今回は，メタデータサーバ故障後に正しいメタデータを取得できるかについてのみ，検証を行った．しかし，メタデータサーバ故障から復旧までに必要な時間や，故障前と故障後や復旧後のメタデータの入出力性能の差などの定量的な評価を行っておらず，今後実施する必要があると思われる．

謝辞 本研究の一部は，文科省科研費特定領域研究情報爆発 IT 基盤 (課題番号 21013005) および文科省次世代 IT 基盤構築のための研究開発「研究コミュニティ形成のための資源連携技術に関する研究」(データ共有技術に関する研究) による．

参 考 文 献

1) Sage Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, Carlos Maltzahn, Ceph: A Scalable, High-Performance Distributed File System, Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI ‘ 06), November 2006.
2) Sage A. Weil, Ceph: Reliable, Scalable, and High-Performance Distributed Storage, Ph.D. thesis, University of California, Santa Cruz, December 2007.
3) POSIX, http://standards.ieee.org/develop/wg/POSIX.html.
4) FUSE, http://fuse.sourceforge.net/.
5) Mike Mesnier, Gregory R. Ganger, Erik Riedel, Object-based storage, IEEE Communications Magazine, August 2003, pp.84-90.
6) Sage A. Weil, Andrew W. Leung, Scott A. Brandt, Carlos Maltzahn. RADOS: A Fast, Scalable, and Reliable Storage Service for Petabyte-scale Storage Clusters. Petascale Data Storage Workshop SC07, November 2007.
7) Sage Weil, Scott A. Brandt, Ethan L. Miller, Carlos Maltzahn, CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data, Proceedings of SC ‘ 06, November 2006.
8) Bonnie++, http://www.coker.com.au/bonnie++/.
9) F. Hupfeld, T. Cortes, B. Kolbeck, E. Focht, M. Hess, J. Malo, J. Marti, J. Stender, E. Cesario, XtreemFS - a case for object-based file systems in Grids, Concurrency and Computation: Practice and Experience, Volume 20 Issue 8 June 2008.
10) XtreemOS, http://www.xtreemos.eu/.