プライベートクラウド環境における分散ファイルシステムの試作
―サーバ機能の委譲を考慮したクライアント側システム―

2009SE017 筑紫嵩大　2009SE107 加藤卓
指導教員：宮澤元

1　はじめに

近年，クラウドコンピューティング（以下クラウド）が様々な場面で用いられている．クラウドとは，データセンターなどに置かれたサーバ群がインターネットなどのコンピュータネットワークを介して，アプリケーションやストレージなどのサービスを提供するコンピュータの利用形態である．クラウドには大きく分けて，パブリッククラウドとプライベートクラウドの 2 つの形態が存在する．パブリッククラウドは，インターネットを経由することによって不特定多数のユーザに広くサービスを提供するオープンなクラウドである．一方で，プライベートクラウドは，企業内や学校内など限定された環境で利用されることが多く，利用できるリソースも制限されている．また，パブリッククラウドと異なりサーバのみならずクライアントも組織の管理下に置かれている．

我々はプライベートクラウド環境において効率的に動作する分散ファイルシステムを開発している．本分散ファイルシステムでは本来サーバが担っている機能の一部をクライアントへ委譲することで，サーバの負荷の低減を実現する．プライベートクラウドではクライアントもシステムの管理下に置かれており，クライアントの信頼性が高いためこのようなサーバ機能の委譲が可能である．

本分散ファイルシステムのクライアントでは，様々な性能のコンピュータが利用されると考えられるが，性能の低いクライアントにサーバ機能を委譲すると処理能力や容量の不足などが起こりうる．こういった利用環境の違いに対して，いつどのような場面でサーバ機能をどの程度委譲させればよいか，また，クライアントへの影響は自明でない．

そのため，本研究では，サーバ機能をどのように委譲するかに関する基礎データを得るために，本分散ファイルシステムのクライアント側システムを実装するとともに，2 種類のクライアントを用いて実験を行い，委譲機能の自明でない点を明らかにする．

2　本分散ファイルシステムの全体像

本分散ファイルシステムは，クライアント側システム，メタデータサーバ，ストレージサーバから構成される．本章では，それらの概要および役割について述べる．

2.1　クライアント側システム

クライアント側システムは，ユーザが実際に使うクライアントホスト上で実行される．クライアント側システムの主な役割は，アプリケーションのファイルアクセス要求に応じてメタデータサーバに通信を行い，メタデータサーバからファイルについての情報を得，それを元にストレージサーバへ通信を行いファイルへアクセスすることである．

2.2　メタデータサーバ

メタデータサーバは，ストレージサーバ内に保存されているファイルのメタデータの管理を行う．クライアント側システムからのファイルアクセス要求に応じ，inode 番号などのメタデータをクライアント側システムへ伝えたり更新したりする．

2.2.1　メタデータサーバスレーブ

クライアントが複数存在する場合，クライアントからの要求の増大によってメタデータサーバに負荷がかかってしまう．それを低減するため，クライアントホスト上にメタデータサーバ機能の一部を委譲されるメタデータサーバスレーブを設置する．それに対し，通常のメタデータサーバをメタデータサーバマスタと呼ぶ．

2.3　ストレージサーバ

ストレージサーバは，ファイルデータの保存を行う．クライアントからの要求に応じ，指定されたファイルデータの読出し及びクライアントから送信されたファイルデータの保存を行う．ストレージサーバは複数台稼働しており，ストレージサーバ間でレプリケーションを取っている．

2.3.1　ストレージサーバスレーブ

クライアントの数が増えるとストレージサーバの負荷が増大する．特に書き込みの場合，レプリケーションを行っているのでそれだけストレージサーバに書き込むデータ量が増え，それに伴いストレージサーバの負荷が増大する．それを低減するため，クライアントホスト上にストレージサーバ機能の一部を委譲されるストレージサーバスレーブを設置する．それに対し，通常のストレージサーバをストレージサーバマスタと呼ぶ．

3　クライアント側システムの設計

クライアント側システムは，アプリケーションのファイルアクセス要求を受け取り，メタデータサーバ，ストレージサーバと通信を行い，ファイルアクセス要求を満たす．本節では，分散ファイルシステム内のクライアント側システムの設計について述べる．

3.1　サーバとの通信

本節では，クライアント側システムとそれぞれのスレーブサーバを含めた場合のメタデータサーバ，ストレージサーバとの通信について述べる．

3.1.1　スレーブなし時の通信

スレーブがない場合，通信を行うのはクライアント側システム、および外部のメタデータサーバ，ストレージ

サーバである．ユーザアプリケーションからのファイルアクセス要求に応じて，クライアント側システムがメタデータサーバへ，open や unlink などのファイル操作コマンドを送信する．このとき，要求するファイルのパス名や新規作成フラグなども送信する．成功時はパス名に対応する inode 番号が，失敗時はエラーメッセージが返される．次に，クライアント側システムはストレージサーバへ接続し，読み込みの場合は inode 番号を送信し，ファイルデータを受け取る．書き込みの場合は inode 番号，部分リスト，ファイルデータを送信し，保存の成否を受け取る．

3.1.2 メタデータサーバスレーブ存在時の通信

メタデータサーバスレーブが存在する場合，クライアント側システムからのファイル要求に対して初回接続時は，メタデータサーバマスタへ接続を行い，要求するファイルのメタデータ情報を問い合わせる．メタデータサーバマスタはそれに対し，該当するメタデータ情報を保存しているメタデータサーバスレーブの情報をクライアントへ返す．情報を受け取ったクライアント側システムは，そのメタデータサーバスレーブへ接続を行い，要求したファイルのメタデータ情報を取得する．2 回目以降の接続時は，クライアント側システムは前回接続時の情報を保持しているので，直接メタデータサーバスレーブに接続を行って，メタデータ情報を入手することが出来る．

3.1.3 ストレージサーバスレーブ存在時の通信

ストレージサーバスレーブが存在する場合，クライアント側システムはストレージサーバスレーブに接続し，レプリケーションレベルと同じ数のストレージサーバすべてに，ストレージサーバスレーブが直接書き込みを行う．これにより，ストレージサーバ同士でレプリケーションを行う必要はない．

3.2 キャッシュ管理

本分散ファイルシステムでは，キャッシュを利用してメタデータサーバ，ストレージサーバとの通信効率の向上を図っている．本節では，キャッシュ管理のアプローチを述べる．

3.2.1 キャッシュの取得

本分散ファイルシステムのクライアント側システムでは，ストレージサーバに保存されているデータにアクセスする際，自身のキャッシュフォルダへキャッシュを保存する．これにより，次回以降，同じデータにアクセスする場合，自身のキャッシュを参照する．

3.2.2 キャッシュの更新

本分散ファイルシステムでは 2 通りのアプローチでキャッシュ一貫性保持を行っている．

- クライアントオフライン時のキャッシュ一貫性保持

クライアントがオフライン時に他クライアントがファイルを更新すると，キャッシュが最新であるという保証をすることが出来ない．そのため，オフライン時のキャッシュ更新の手法として，キャッシュに最新フラグを設ける．新規にキャッシュを取得すると，そのキャッシュに最新フラグが立てられる．このフラグが立っている場合，クライアントはサーバに問い合わせることなくそのキャッシュを使用できる．クライアントがオフラインになった場合，次回クライアント起動時に，キャッシュの最新フラグを全てオフにする．最新フラグがたっていないキャッシュを読み込もうとした場合，クライアントはメタデータサーバに，キャッシュが最新かの問い合わせを行う．これにより，クライアントは使用する 1 つのキャッシュに対しメタデータサーバと 1 回の接続をするだけでキャッシュが最新かどうかの保証を得ることが出来る．

- ファイル更新時のキャッシュ一貫性保持

クライアントがオンラインの場合，クライアント側システムにキャッシュ削除サーバを設置することでキャッシュの削除を行う．他クライアントがファイルを編集した場合，そのキャッシュが更新されたことがメタデータサーバへ通知され，メタデータサーバからキャシュ削除サーバへキャッシュ削除命令が送られる．キャッシュ削除サーバはメタデータサーバから当該キャッシュの削除命令を受信した場合，キャッシュフォルダから当該キャッシュを削除する．

4 クライアント側システムの実装

クライアント側システムは，FUSE サーバとして実装されている．アプリケーションに特別なライブラリをリンクしたり，カーネルに手を加えたりすることなく，アプリケーションのファイルアクセス要求を受け取ることができる．

4.1 FUSE

FUSE(Filesystem in Userspace)[1] とは，ファイルシステムをユーザ空間で動作するプロセスとして実装するための仕組である．ここでは FUSE サーバに定義されている関数がどのような処理をしているかを述べる．

- open

FUSE の open はシステムコールの open に対応する関数である．関数内では，ローカルキャッシュがあるかないかの確認を行う．ローカルキャッシュが存在しない場合は，メタデータサーバに open を行う処理要求としてファイル名を送信し，inode 番号を受信する．受信した inode 番号をストレージサーバへ送信することでファイルのデータを受信し，ローカルキャッシュを作成する．ローカルキャッシュが存在した場合は，そのローカルキャッシュが最新かどうかの確認を行う．最新かどうかの確認は最新フラグとメタデータサーバへの問い合わせで行う．

- flush

FUSE の flush はシステムコールの close と対応する関数である．我々のシステムではファイルを close した際にファイルに書き込みが行われていれば，ストレージサーバへの書き込みが行われる．そのため関数内では，そのファイルに書き込みが行われているかどうかの確認がされ

る．この書き込みが行われているかどうかの確認は write の際に立てられるフラグによって決定される．メタデータサーバに更新 close を行う処理要求として書き込み後のサイズを送信する．最後にフラグをクリアする．書き込みが行われていない場合は，ストレージサーバとの通信は行わず，メタデータサーバへ close 処理要求を送信する．

4.2 メタデータサーバの決定

クライアント側システムはメタデータサーバスレーブが存在する場合，メタデータサーバマスタかどのメタデータサーバスレーブに接続するかを決定しなければならない．ここではその決定方法について述べる．

4.2.1 メタデータサーバリスト

クライアント側システムは host list というメタデータサーバリストを持っている．メタデータサーバリストにはメタデータサーバスレーブのホスト名とそのメタデータサーバスレーブが担当しているディレクトリが書き込まれている．クライアント側システムはこのメタデータサーバリストを使い接続するメタデータサーバを決定する．

4.2.2 メタデータサーバの決定方法

クライアント側システムはメタデータサーバに接続を行う前にまずメタデータサーバリストを参照する．パス名と一致するメタデータサーバスレーブが存在する場合，そのメタデータサーバスレーブに接続する．パス名と一致するメタデータサーバスレーブが存在しない場合，メタデータサーバマスタに接続する．

メタデータサーバマスタに接続した場合，メタデータサーバマスタへ送ったパス名がメタデータサーバスレーブに委譲してある範囲であると，メタデータサーバマスタからホストとパスの組が返ってくる．クライアント側システムは返ってきた組をメタデータサーバリストに書き込み，メタデータサーバスレーブへ改めて接続する．メタデータサーバマスタへ送ったパス名がメタデータサーバスレーブに委譲してある範囲でないと，通常の処理が行われ確認通知が返される．

4.3 ストレージサーバの決定

クライアント側システムはストレージサーバスレーブが存在する場合，どのストレージサーバマスタかストレージサーバスレーブに接続するかを決定しなければならない．ここではその決定方法について述べる．

4.3.1 ストレージサーバの決定方法

クライアント側システムはストレージサーバに接続する場合，inode 番号にハッシュをかけることで，接続するサーバを決定する．ストレージサーバスレーブを持つかどうかは，システム起動時に設定ファイルを読み込むことで静的に決定し，ストレージサーバスレーブを持つ場合は，そこに接続する．

5 実験

今回実装したクライアント側システムを用い，スレーブ無し，メタデータサーバスレーブのみ，ストレージサーバスレーブのみ，両スレーブありの 4 つの状態での実験を行った．実験ではノート PC とデスクトップ PC の 2 種類のコンピュータを使用している．表 1 に使用したコンピュータの仕様を示す．クライアントにはノート PC とデスクトップ PC の 2 種類を使用した．メタデータサーバマスタにはノート PC，ストレージサーバマスタにはデスクトップ PC を使用した．

表 1 実験で使った 2 種類のコンピュータの仕様

	ノート PC	デスクトップ PC
CPU	Intel(R) Core(TM)2 Duo T8100 2.10GHz	Intel(R) Core(TM) i7-2600 3.4GHz
memory	4GB	8GB
OS	Vine Linux 4.2	Ubuntu server 11.10
ネットワーク	1000BASE-T Ethernet	1000BASE-T Ethernet

5.1 書き込み

5.1.1 所要時間

スレーブの有無で書き込みの所要時間に変化が生じるかを調べるために，cp コマンドを用い，ローカルファイルシステムに存在するサイズ 1GB のファイルデータを本分散ファイルシステムに書き込んだ時の所要時間を計測する実験を行った．

実験結果を図 1 に示す

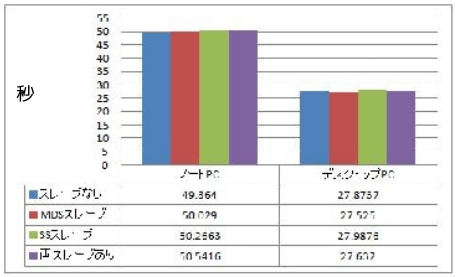

図 1 ノート PC とデスクトップ PC の書き込み時所要時間

結果から，書き込みの所要時間の平均はノート PC とデスクトップ PC ともにストレージサーバスレーブの有無に関係無くほとんど一定である．これは cp の処理にストレージサーバスレーブが関与が少ないためだと考えられる．

5.1.2 ロードアベレージ

コンピュータがどの程度の性能を持っている時にスレーブを起動させるのが最適かを調べるために，cp コマンドを用い，ローカルファイルシステムに存在するサイズ 1GB のファイルデータを本分散ファイルシステムに書き込んだ時のロードアベレージを計測する実験を行った．

実験結果を図 2 に示す．

スレーブ無しとメタデータサーバスレーブの結果，ストレージサーバスレーブと両スレーブありの結果がほぼ同じである．これはメタデータサーバスレーブが書き込みへの処理にほとんど関与しないためだと考えられる．ロード

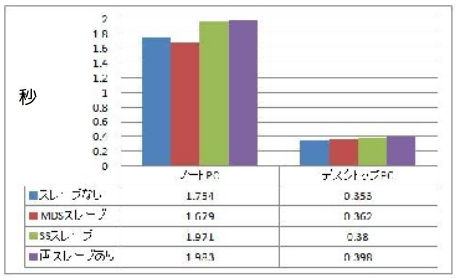

図 2　ノート PC とデスクトップ PC の書き込み時ロードアベレージ

アベレージの平均は，スレーブ無しのノート PC が 1.75，デスクトップ PC が 0.35 であり，ストレージサーバスレーブありのノート PC が 1.97，デスクトップ PC が 0.38 である．ノート PC のロードアベレージの差が 0.22 であるのに対して，デスクトップ PC のロードアベレージの差は 0.03 とデスクトップ PC の方は差が少ない．この結果からストレージサーバスレーブを用いるとノート PC のようなコンピュータではロードアベレージが上がる可能性がある．

5.2　読み込み

5.2.1　所要時間

スレーブの有無で書き込みの所要時間に変化が生じるかを調べるために，cat コマンドを用い，本分散ファイルシステムに存在するサイズ 1GB のファイルデータを読み込んだ時の所要時間を計測する実験を行った．

実験結果を図 3 に示す

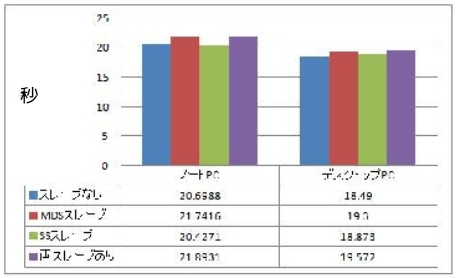

図 3　ノート PC とデスクトップ PC の読み込み時所要時間

所要時間の平均はほとんど一定だが，ノート PC とデスクトップ PC ともにメタデータサーバスレーブがあると数秒遅くなっている．メタデータサーバスレーブに接続すると，メタデータサーバリストを参照するという処理が増える．それが原因で，所要時間が増加したと考えられる．

5.2.2　ロードアベレージ

コンピュータがどの程度の性能を持っている時にスレーブを起動させるのが最適かを調べるために，cat コマンドを用い，本分散ファイルシステムに存在するサイズ 1GB のファイルデータを読み込んだ時のロードアベレージを計測する実験を行った．

実験結果を図 4 に示す．

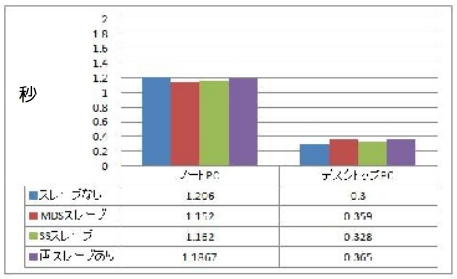

図 4　ノート PC とデスクトップ PC の読み込み時ロードアベレージ

スレーブ無しとストレージサーバスレーブの結果，メタデータサーバスレーブと両スレーブありの結果が同じである．これはストレージサーバスレーブが読み込みの処理に関与しないためだと考えられる．ロードアベレージの平均は，スレーブ無しのノート PC が 1.2，デスクトップ PC が 0.3 であり，ストレージサーバスレーブありのノート PC が 1.15，デスクトップ PC が 0.35 である．ノート PC とデスクトップ PC の両方で平均の差が少ない．

6　おわりに

我々はプライベートクラウドで効率的に動作する分散フィルシステムを開発している．本分散ファイルシステムはメタデータサーバ，ストレージサーバの負荷を軽減するためクライアント側システムに各サーバの機能の一部を委譲することが出来る．実装したクライアント側システムを用い，ノート PC およびデスクトップ PC をそれぞれクライアントホストとして用いる 2 つの異なった環境で実験を行った．実験の結果から，スレーブサーバが有効である場合とそうでない場合があることが分かった．メタデータサーバスレーブを用いた場合，処理時間およびロードアベレージの変化が少なく，メタデータサーバ機能の委譲が有効であると考えらる．また，ストレージサーバスレーブを用いた場合，処理時間はほぼ一定であるが，ノート PC のロードアベレージの平均の差がデスクトップ PC と比べ上昇した．従って，ストレージサーバスレーブを用いる場合，クライアントの負荷状況に注意する必要があると考えられる．

今後の課題として，タブレット端末など今回実験に用いたコンピュータより更にリソースの限られたコンピュータを用いた実験を行った際のスレーブの有効性の確認，および実験から得た基礎データを元に，より効率的な通信，サーバの負荷軽減を目標としたシステムの改良が必要である．

参考文献

[1] FUSE: Filesystem in Userspace (http://fuse.sourceforge.net/)