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
—ストレージサーバ機能のクライアントへの委譲—

2009SE010 青木 勇貴　2009SE038 長谷川 浩之
指導教員: 宮澤 元

1　はじめに

近年，コンピューティングリソースの集積に対し，オンデマンドにネットワーク経由でアクセスすることを可能とするクラウドコンピューティングというコンピュータの利用モデルが注目されている．クラウドコンピューティングはサービスの提供範囲に応じて，パブリッククラウドとプライベートクラウドの 2 つの実装モデルに大別できる．パブリッククラウド [1] の特徴として，コンピューティングリソースをインターネット経由で誰でも利用することができ，サービス条項が定められ，管理運用にかかるコストを削減できることが挙げられる．プライベートクラウドの特徴として，閉じた空間で，特定の組織の為だけに運用され，コンピューティングリソースをより自由にコントロールできる．一方で，利用できるコンピューティングリソースに制限が発生することが挙げられる．

我々はプライベートクラウド向けの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，プライベートクラウドの特徴をふまえ，サーバ機能の一部をクライアントに委譲することでサーバの負荷を軽減し，リソースを有効活用できるようにすることを試みる．

本稿では，本分散ファイルシステムのストレージサーバについて述べる．複数のサーバホストで動作し，ファイルのレプリケーションを行うのに加え，クライアントホスト上のスレーブサーバに処理の一部を委譲してサーバの負荷を軽減できる．実装したシステムを用いて実験を行い，スレーブサーバの有効性を確かめる．

2　本システムの概要

ここでは，本分散ファイルシステムの概要について述べる．

2.1　システム全体像

本システムはクライアント側システム，メタデータサーバ，ストレージサーバの 3 つから構成される．

クライアント側システムの役割として，アプリケーションからのファイルアクセス要求の受付，メタデータサーバからの inode 番号等のメタデータの取得，オープンするデータの取得や保存するデータの送信，接続先ストレージサーバの情報のキャッシュといったことが挙げられる．メタデータサーバの役割として，クライアント側システムの要求に応じて，メタデータの送信，検索，追加，削除，更新を行うことが挙げられる．ストレージサーバの役割として，クライアント側システムから受信したデータの貯蔵，クライアント側システムでオープンされるデータの送信，レプリケーションの作成などによる高可用性および耐障害性の実現，そしてストレージサーバ同士での障害検知と復旧が挙げられる．

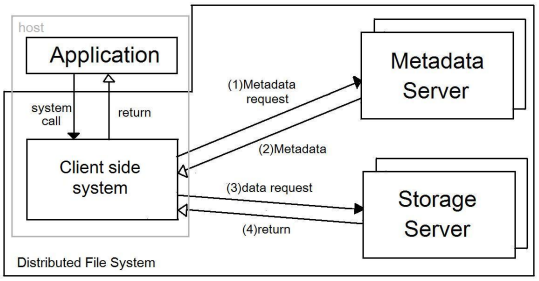

図 1　システムの全体像

図 1 はクライアント側システム，メタデータサーバ，ストレージサーバの間でのデータの流れを示している．

アプリケーションからのシステムコールを受信すると，クライアント側システムは図 1 の (1)–(4) の順にメタデータサーバ，ストレージサーバと通信を行い，アプリケーションに結果を返す．(1) クライアント側システムは最初にメタデータサーバにメタデータ要求を行う．次にクライアントは (2) メタデータサーバから inode 番号を含むメタデータを受信し，そのメタデータを利用して (3) ストレージサーバにデータ要求を行う．ストレージサーバは受信した要求がデータ取得要求であるか，データ保存要求であるかを判断し，それに応じた対応を取る．データ保存要求の場合，必要なメタデータや保存データを受信し，それらを保存した後，(4) 確認通知をクライアント側システムに返す．データ取得要求の場合は，対象となるデータを (4) でクライアント側システムに送信する．

2.2　スレーブサーバ

ストレージサーバとメタデータサーバはそれぞれクライアントホスト上で動作するスレーブサーバに権限や機能の一部を委譲できる．これによって，サーバの負荷を軽減し，リソースを有効活用できることが期待される．

3　ストレージサーバの設計

ストレージサーバは複数のサーバホスト上で稼働し，ファイルの実体を保持する．各サーバホスト上のストレージサーバはクライアント側システムに対するインタフェースを持ち，要求を受け付ける事が出来る．また，ファイルが書き込まれる際は，サーバホスト同士が通信してファイルのレプリケーションを行う．クライアントホスト上にストレージサーバスレーブを置くことでストレージ機能の一部を委譲することも出来る．また，ストレージサーバはハートビートを用いた障害検知や障害復旧の機能も備える．

3.1 ファイルのレプリケーション

クライアント側システムから受信したデータのレプリケーションは (1) から (6) の順に行われる．

(1) プライマリがクライアント側システムからデータを受信する．

(2) プライマリがセカンダリに対して並行にデータを送信する．

(3) データのメモリキャッシュが完了すると，セカンダリがその確認通知を返す．

(4) 全てのセカンダリからメモリキャッシュの確認通知を受け取ると，プライマリがクライアント側システムに確認通知を返す．

(5) データの保存が完了すると，セカンダリがその確認通知を返す．

(6) 全てのセカンダリから保存完了の確認通知を受け取り，かつプライマリ自身での保存が完了すると，プライマリが保存完了と判断して終了する．

上記のレプリケーション方法は Primary-copy Replication[2] と呼ばれ，データの保存を行うストレージサーバのセットに対して順序づけを行う．この順序づけによって優先順位が一番高くなったものをプライマリ，それ以外をセカンダリと呼び，プライマリはクライアントとの通信，自身とセカンダリへの書込について責任を負う．プライマリがスレーブの場合はクライアント側システムがローカルキャッシュとしてデータを保存しているので，自身への保存は行わずに処理を終了する．

3.2 ストレージサーバスレーブ

本ストレージサーバでは，Primary-copy Replication におけるプライマリの負荷をクライアントホストに委譲するために，ストレージサーバスレーブをクライアントホストに置くことが出来る．ストレージサーバスレーブに対して，ファイル読出処理も行う通常のストレージサーバの事をストレージサーバマスタと呼び，これらを区別する必要のない場合は単にストレージサーバと呼ぶ．スレーブを用いてレプリケーションの負荷の一部をクライアントホストに委譲することによって，サーバホストの負荷を軽減出来る．スレーブを作成するかどうかは通信を行うストレージサーバに対する現在の負荷や全ストレージサーバに対する現在の負荷の平均から決定するのが望ましいが，現状では設定ファイルによってスレーブを作成するかどうかを静的に決定している．

スレーブを用いてファイル書込処理を行う場合，マスタのみを用いる場合と異なる点としては，クライアントによって作成される部分リスト (4.1 節参照) に変更が生じる点，レプリケーションを取る個数や処理に変更が生じる点，障害検知や障害復旧を行わない点の三点が挙げられる．

クライアントがスレーブを持つ場合はスレーブがプライマリとなる．この場合，スレーブはレプリケーションの個数としてはカウントされない．

ファイル読出処理については，スレーブを持つ場合でもクライアント側システムが直接ローカルキャッシュにアクセスすることができるので委譲する必要がない．

3.3 障害対策

本ストレージサーバではハートビートを用いた障害対策を行っている．一般に N 台のサーバホスト間において，通常時は自分以外のストレージサーバに対してハートビートを送信し，自分以外のストレージサーバからハートビートを受信する事で互いの生存を確認する．

ストレージサーバスレーブについてはファイル書込処理を代行するだけのものであり，クライアントホスト上で動作するので，障害対策は行わない．

3.3.1 障害検知

各ストレージサーバはハートビートが送信されなくなる事で障害を検知する．一般に N 台のストレージサーバが存在する場合，任意のストレージサーバに障害が発生しダウンすると，ダウンしたストレージサーバを除く N-1 台のストレージサーバはダウンしたストレージサーバから一定時間ハートビートが送信されていない事からそのストレージサーバがダウンした事を検知する事が出来る．

3.3.2 障害復旧

各ストレージサーバはハートビートの送信再開によってダウンしていたストレージサーバの活動再開を検知する．一般に N 台のストレージサーバが存在する場合，任意のストレージサーバが障害から復旧し活動を再開した事は，活動再開したストレージサーバ以外の N-1 台のストレージサーバが復旧したストレージサーバから再度ハートビートを受信する事で検知する．

復旧時のファイルの一貫性は復旧用ファイルを利用して保持される．復旧用ファイルとは，ストレージサーバのダウン中に書込処理が行われたファイルの inode 番号を保持するファイルである．復旧しようとするストレージサーバは，復旧用ファイルの情報を用いてファイルの最新の内容を他のストレージサーバから得る事が出来る．

ストレージサーバスレーブはハートビートを用いた障害対策を行わないので，自身で復旧処理を行う事が出来ない．そこで，ファイル書込処理の発生時にマスタの一つに対して復旧処理を代わりに行うよう依頼する．

4 ストレージサーバの実装

ストレージサーバマスタは TCP サーバとして実装されており，ストレージサーバスレーブはそれを持つクライアント側システムの要求に対して応答する TCP サーバとして実装されている．ファイルの実体はストレージサーバのローカルファイルシステムのファイルとして保持される．ここではストレージサーバリスト，クライアント側システムとのインタフェース，障害対策に対する処理の詳細について具体的に述べる．

4.1 ストレージサーバリストの実装

クライアントの接続先のストレージサーバの決定，レプリケーションを取るストレージサーバのセットの特定，Primary-copy Replication におけるプライマリとセカン

ダリの決定に用いるためにストレージサーバリストを用いる．ストレージサーバリストは，設定ファイルの情報からクライアント側システムで作成される．ストレージサーバリストにはストレージサーバクラスタを構成する全ストレージサーバを持つフルリストとフルリストから作成される部分リストがある．クライアント側システムとストレージサーバとの通信には部分リストが使用される．部分リストの作成にあたっては，接続優先度の順に部分リストに追加される．ストレージサーバスレーブへの通信は，部分リストの先頭に “localhost.localdomain” を追加する事によって実現されている．

4.2 クライアントに対するインタフェース

今回我々が作成したストレージサーバに対するクライアント側システムからの要求には，ファイル読出とファイル書込がある．

4.2.1 ファイル読出処理

クライアント側システムがストレージサーバ上に保存されたファイルを取得するためにストレージサーバに対してファイル読出要求を送信してきた場合，ストレージサーバの処理は以下のように行われる．

(1) クライアント側システムから inode 番号，オフセット，サイズを取得する．取得したサイズの値が-1 の場合はファイル全体を要求している事を表し，ファイルサイズを送信してから要求されたデータを送信する．

(2) 受信した inode 番号をファイル名として持つファイルが存在するかとオフセット，サイズが適切であるかを確認し，その結果をクライアント側システムに通知する．

(3) ファイルが存在し，オフセット，サイズが適切であった場合，該当ファイルを開いて送信データを作成し，クライアント側システムに送信する．

(4) クライアント側システムへの送信が完了すると処理が終了する．

4.2.2 ファイル書込処理

クライアント側システムがストレージサーバに対してファイル書込要求を送信してきた場合，ストレージサーバは以下のように Primary-copy Replication 処理を行う．

(1) クライアント側システムから要求を受信したストレージサーバマスタはプライマリとなり，inode 番号，レプリケーションレベル，部分リスト，オフセット，サイズ，保存データを送信する．

(2) プライマリはセカンダリに対して接続を行い，inode 番号，オフセット，サイズ，保存データを並行に送信する．

(3) データを受信したセカンダリはデータのキャッシュが完了すると確認通知をプライマリに送信する．その後ファイルへの保存を行い，それが完了すると確認通知をプライマリに送信して処理を終了する．保存ファイル名にはクライアント側システムから受信した inode 番号を用い，オフセットからサイズ分上書きされる．また，ファイルが存在しない場合はファイルを作成した後で保存を行う．

(4) プライマリは全てのセカンダリからキャッシュ完了の確認通知を受信するとクライアント側システムに確認通知を送信する．

(5) プライマリは全てのセカンダリから保存完了の確認通知を受信すると自身へのデータの保存を行い，それが完了すると処理を終了する．

クライアント側システムがストレージサーバスレーブを持つ場合，スレーブがプライマリとなる．スレーブがプライマリの場合，セカンダリに送信するデータはクライアント側システムにローカルキャッシュとして保存されているファイルから作成するので，保存データは受信しない．ファイル書込要求を受信したストレージサーバは，部分リストの先頭が “localhost.localdomain” である事から自身がスレーブであると判断する．また，ストレージサーバスレーブはレプリケーションレベルにはカウントされないので，通信を行うセカンダリの数がマスタのみの場合と比べて一台増える．

プライマリがダウンしている場合，クライアント側システムは部分リストの二番目に載っているストレージサーバに対してファイル書込要求を送信する．代替プライマリはダウンしているストレージサーバの活動再開時に復旧を行えるように，クライアント側システムから受信した inode 番号を復旧用ファイルに保存する．プライマリがスレーブの場合，ファイル書込要求を受信するストレージサーバは部分リストの先頭が “localhost.localdomain” かどうかで自身がスレーブかを判断しているので，クライアント側システムはレプリケーションレベルを 1 つ下げ，部分リストの先頭の “localhost.localdomain” を除いて送信を行う．

セカンダリがダウンしている場合，プライマリはダウンしているストレージサーバの活動再開時に復旧を行えるように，クライアント側システムから受信した inode 番号を復旧用ファイルに保存する．プライマリがスレーブの場合，ハートビートを用いた障害対策を行っていないので，ストレージサーバマスタの一つに復旧用ファイルの作成を代わりに行うよう依頼する．

4.3 障害対策の詳細

ハートビートを送受信するために，各ストレージサーバはハートビートのクライアント機能，サーバ機能をそれぞれ持つ．クライアント，サーバともにフルリストを持ち自分以外のストレージサーバと通信を行う．

4.3.1 復旧用ファイルの作成

復旧用ファイルへの復旧情報の記録は Primary-copy Replication において，セカンダリのストレージサーバがダウンしている時にプライマリによって行われる．復旧用ファイルにおいて復旧するファイルを特定する情報としては inode 番号を用いる．どのサーバに対してどのファイルを復旧するかを判断するためにダウンしたストレー

ジサーバのサーバ番号もセットで保存する．任意のストレージサーバの活動再開を検知した各ストレージサーバは復旧用ファイルに記録されている inode 番号をファイル名として持つファイルを活動再開したストレージサーバに対して送信することで復旧を行う．

5　実験

我々の実装したシステムの有用性を確認するために実験を行った．

5.1　実験環境

下の表 1 は，クライアントとサーバとして使用した各 PC の仕様を表す．クライアントとサーバの仕様は同じである．なお，サーバには PC を 2 台用いた．

表 1　クライアントとサーバの仕様

CPU	Intel(R) Core(TM) i7-2600 3.4GHz
memory	8GB
OS	Ubuntu server 11.10
network	1000BASE-T Ethernet

5.2　負荷の実験

ストレージサーバスレーブを用いることにより，サーバの負荷が軽減することを確認するためにサーバの負荷を調べる実験を行った．Primary-copy Replication のプライマリとなるサーバの処理 (スレーブ) をクライアントに委譲した時，そのサーバの負荷が軽減するかどうかを示す．そのためにクライアントとなるホストを 4 台用意し，一斉にサイズ 1GB のファイルを書き込み，負荷をかける．その 4 台のクライアントの中のスレーブを持つクライアントの数を 0，1，2，3，4 台と増やしていき，それぞれの負荷を 10 回ずつ計測する．負荷を計測する方法としては，一斉にファイルを送信した時のロードアベレージの 1 分間の平均値 (以下ロードアベレージ) の最大値をとる．サーバに書き込みを行うときはレプリケーションレベルの 1，2 の時の負荷を計測する．

5.2.1　実験結果

実験の結果を示す．

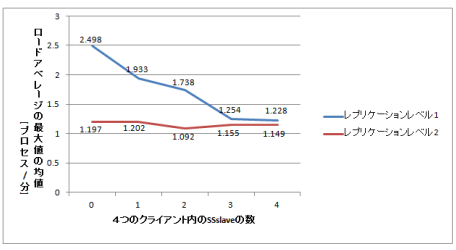

図 2　ロードアベレージの最大値の平均値

図 2 は，サイズ 1GB のファイルを 4 つのクライアントからサーバに一斉に書き込んだ時のロードアベレージの最大値の平均を 4 台中ストレージサーバスレーブ (SSslave) を持つクライアントの数ごとに表したものである．レプリケーションレベルは 1 と 2 である．レプリケーションレベル 1 では，スレーブが 0 と 1 の時にほぼ同じで，2，3，4 の時はわずかに下がっている．レプリケーションレベル 2 では，スレーブの数を増やしていくごとに負荷が軽減されている．スレーブ 0 から 1 の差が 0.565，スレーブ 1 から 2 の差が 0.195，スレーブ 2 から 3 の差が 0.484，スレーブ 3 から 4 の差が 0.026 となっている．

5.2.2　分析

レプリケーションレベル 1 の結果を見ると，スレーブ数 2 の時が一番下がっており，SSslave の数を増やせば増やすほど負荷が下がるようには見えない．従って，レプリケーションレベルが 1 の時，SSslave を用いることによって負荷の軽減はされておらず効果がないように考える．レプリケーションレベル 2 の結果を見ると，スレーブ数が増すごとに負荷が軽減されているが，スレーブを持つクライアントの数を 1 つ増やせば，一定の負荷が下がるわけではなく変則的な下がり方であり，SSslave の数が 3 と 4 ではほぼ同じである．これは，書き込み処理ではレプリケーションレベル 1 であってもある一定の負荷がサーバにかかり，ロードアベレージの値が 1.2 前後以下にならないと考えられる．サーバに一定以上の負荷がかかっている時，SSslave が有効であると考える．

6　おわりに

我々は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開発を行い，ストレージサーバスレーブをクライアント側システムに委譲することによって，ストレージサーバの負荷の軽減を行った．サーバのホスト名のリストを用いたデータの送受信を行い，レプリケーションの作成のためには Primary-copy Replication を用いた．また，障害検知のためにハートビートを用いて，障害復旧を行った．我々の実装したシステムの有効性を確認するために，ストレージサーバスレーブの有無によるストレージサーバのロードアベレージの違いを調べた．その結果，ある程度の負荷がストレージサーバにかかっている場合，ストレージサーバスレーブを用いることでストレージサーバの負荷を軽減できることが分かった．

今後の課題としては，サーバに対して書き込みを行う時，ストレージサーバのリソースの負荷状況に応じた接続先のサーバの選別方法の最適化を行うことと，ストレージサーバの負荷状況に応じてストレージサーバスレーブに動的に委譲を行うことが挙げられる．

参考文献

[1] W. Jansen and T. Grance, “Guidelines on security and privacy in public cloud computing,” in *http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf*, pp. 1–11, 2011.

[2] P. A. Alsberg and J. D. Day, “A principle for resilient sharing of distributed resources,” in *Proceedings of the 2nd International Conference on Software Engineering*, pp. 562–570, 1976.