Vol.2017-HPC-158 No.10
2017/3/8

並列離散イベントシミュレータを用いた分散メタデータサーバの性能評価

小林 淳司[1]　建部 修見[2]

概要：現在, 分散システムの性能評価の手法としては, 実環境を用いた性能評価が一般的である. しかし, 実環境を用いた性能評価では, 評価を行うたびにシステムの構成や設定を変更したソフトウェアを配置し直す必要があり, 効率が悪く手間がかかるという問題が発生する. また, 評価環境上で利用できるノード数に限りがある場合, サーバに対する十分なリクエスト数を発行することができず, システムの性能限界を知ることができないという問題が発生しうる. そのため, 実環境を用意すること無く, 仮想的に性能評価を行う手法が求められる. そのため, 並列離散イベントシミュレータ CODES/ROSS を用いて仮想的に大規模分散システムの性能評価を行う手法が, PPMDSsim によって提案された.　本研究では, PPMDSsim でサポートされていないディレクトリの作成, 参照, 削除操作のシミュレーションを実装することによって, より多くの観点から分散システムの性能評価を可能にすることを目的とする. 今回, PPMDS でのディレクトリ操作のスケーラビリティについての性能評価を行った. シミュレーション結果からクライント数を増やすことによって, サーバ数にかかわらず一定の性能限界を示すという結果が得られた. 加えて, 現在実際の PPMDS で変更することができないディレクトリ分散先サーバについて, シミュレーション上で変更可能にしディレクトリ分散先サーバの性能への影響を調査した. シミュレーション結果から, 分散先サーバ台数のディレクトリ作成数及びファイル作成数への影響について考察を行った. また, バンド幅, ネットワーク開始遅延の性能への影響について調査を行った. シミュレーション結果から, ネットワーク開始遅延がボトルネックになっている可能性があると考察した. 最後に, 並列シミュレーションの実行時間の改善についての調査を行った.

1. はじめに

分散システムの開発において, 実験によるシステムの性能評価は非常に重要である. システムの構成や使用する分散プロトコルなどは, システムに大きく影響を与える. そのため, これらの条件を変えつつ評価を行い, 最適な設計を見極めたいという需要がある. 分散システムの性能評価の手法としては, 実環境を用いた性能評価が一般的である. しかし, 実環境を用いた性能評価では, 評価を行うたびにシステムの構成や設定を変更したソフトウェアを配置し直す必要があり, 効率が悪いという問題が発生する. そのため, 並列イベントシミュレータ CODES/ROSS を用いて仮想的に大規模分散システムの性能評価を行う手法が, PPMDSsim[1] によって提案された. PPMDSsim は平賀らに [2] よる分散ファイルシステムのための分散メタデータサーバ PPMDS をシミュレーションによる評価対象とし, PPMDS のスケーラビリティについて, シミュレーションスタディによって解明することを目的としている. PPMDSsim はファイル作成, ファイル参照, ファイル削除のシミュレーションプログラムを実装し, PPMDS のファイル操作に関してのスケーラビリティについて性能評価を行っている. 本研究では, PPMDSsim でサポートをしていなかったディレクトリに対する操作を実装し, ディレクトリ作成, ディレクトリ参照, ディレクトリ削除のスケーラビリティを評価した.

以下, 第 2 章では, シミュレーションの対象とした分散メタデータサーバ PPMDS, 並列離散イベントシミュレーション, シミュレーションプログラムの構築に利用した, CODES 及び, ROSS について紹介する. 第 3 章では, 本研究において実装した PPMDS シミュレータについて, 設計及び実装について述べる. 第 4 章では, PPMDS シミュレータによるシミュレーション結果を示し, 実環境との結果比較及び考察を行う. 第 5 章では, 結論として研究のまとめと今後の課題について述べる.

[1] 筑波大学情報学群情報科学類
College of Information Science, University of Tsukuba
[2] 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba

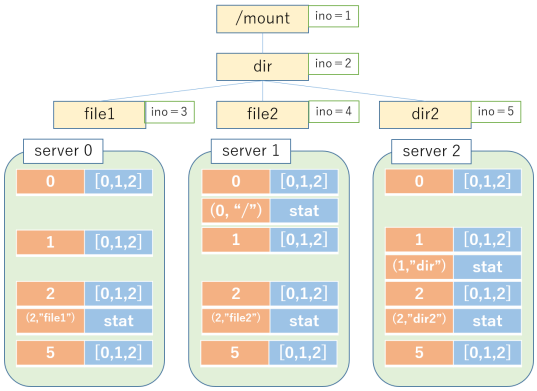

図 1　PPMDS での inode 格納方式

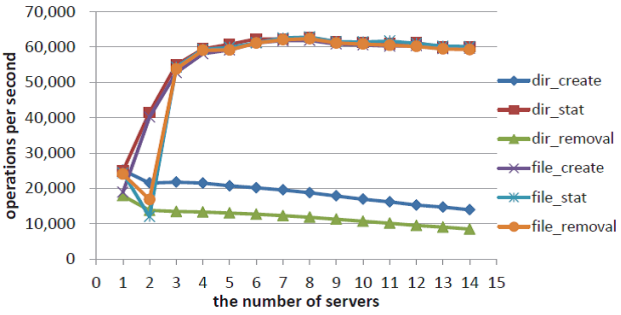

図 2　88 クライアントを用いた並列ファイルシステム操作
文献 [2] より引用

2. 背景と関連研究

2.1 分散メタデータサーバ PPMDS

分散メタデータサーバ PPMDS はメタデータ管理サーバを複数ノードへ分散化を行うことによってスケーラビリティを向上することを目的としている．一般的なファイルシステムではデータ格納場所の管理のため inode と呼ばれるメタデータを保持している．PPMDS では，ディレクトリネームスペースを分散 Key-Value store 上に保持することによって複雑なツリー状のネームスペースをスケーラブルに扱うことを可能にしている．親 inode のエントリ番号と自身のエントリ名を Key として保持し，Value には inode エントリと対応付けされるメタデータを格納する（**図 1**）．メタデータ操作の際，複数のサーバにまたがる Key-Value ペアへの操作にはアトミック性が要求される，PPMDS では Dynamic Software Transactional Memory[3] をベースとした，ノンブロッキングなソフトウェアトランザクションを，Key-Value store 上に実現する熊崎ら [4] の手法を実装することによって実現している．PPMDS は，分散先サーバ情報として分散先サーバリストをディレクトリごとに作成し，inode エントリと同じ Key-Value store 上に格納する，このリストは分散先サーバすべてに保存される．各 inode エントリは親ディレクトリの分散先サーバリストに従い，ハッシュ関数を利用した分散によって，格納先サーバのサーバ id を決定する．文献 [2] より引用した，88 クライアントによる並列ファイルシステム操作の結果を**図 2** に示す．グラフの横軸はサーバ数を表し，縦軸は秒間あたりのオペレーション操作回数を示す．ファイル作成，参照，削除については PPMDS のサーバを増加させるにれて，62,000ops/sec まで性能が上がっている．ディレクトリ操作については，ディレクトリ参照はファイル操作と同様の性能向上が得られた．ディレクトリ作成と削除についてはサーバ台数が増えるにしたがって性能が低下している．これはディレクトリ分散先サーバリストを全サーバに作成するコストが大きいためと考えられる．

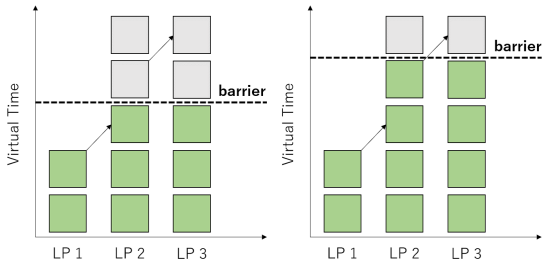

図 3　保守的 (conservative) な手法を用いたときのイベント処理

2.2 並列離散イベントシミュレーション

並列離散イベントシミュレーションとは，離散イベントシミュレーションを並列に実行するためのシステムである．本研究で並列離散イベントシミュレータとして用いた ROSS が採用している，並列離散イベントシミュレーションを実現するための 2 つの手法について述べる．

- 保守的な手法
  LP があるイベントを処理する際，イベントに影響を及ぼす，すべてのイベントが処理されるまでそのイベントの処理を行わず，そのイベントに影響を及ぼすすべてのイベントの処理が終了したことを確認したあとにイベントの処理を行う，よってイベントを処理する順番に矛盾が発生しないことが保証され並列にイベントを処理することを可能にする手法．**図 3** に保守的な手法を用いたときのイベント処理の例を示す．
- 楽観的な手法
  イベントを処理する順番に矛盾が発生することを許容する，矛盾が発生した場合ロールバック処理を行うことによって矛盾が起こる前の状態に戻し，シミュレーションを再度行うことによって並列にイベントを処理することを可能にする手法．**図 4** に楽観的な手法を用いたときのロールバック処理の例を示す．

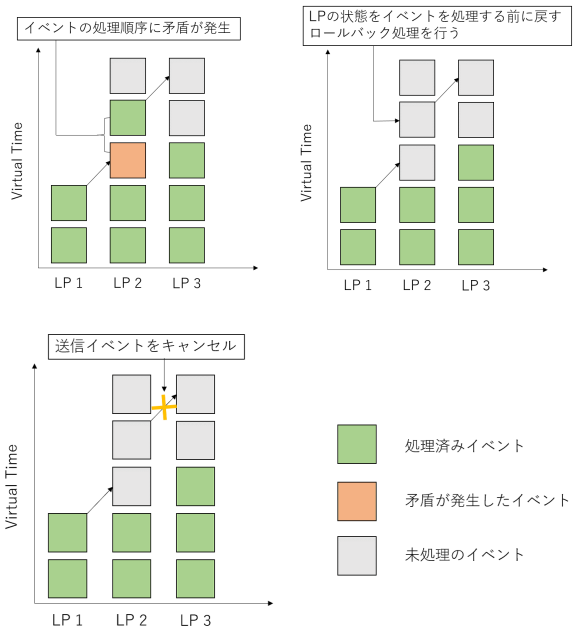

図 4　楽観的 (optimistic) な手法を用いたときのロールバック処理

2.3 ROSS: Rensselaer's Optimistic Simulation System

Rensselaer's Optimistic Simulation System [5] はマルチプロセッサシステムやスーパーコンピュータ上で実行することができる並列離散イベントシミュレータである. ROSS は大規模なシミュレーションモデルを実行することが可能である. シミュレーションプログラムはシステムの構成物を logical processes (LP) の集合として実装することによってモデリングする, そしてそれらの LP がタイムスタンプされたイベントメッセージをやり取りすることによってサーバに新しいジョブが到着したことなどを表現する. ROSS 実行時にオプションとして実行方法 (保守的な手法で実行するか楽観的な手法で実行するかなど) や並列シミュレーションで用いられるパラメータ, KP（カーネルプロセス）数やバッチ数, などを指定することができる.

- KP
  LP は KP 上にマッピングされ, KP はマッピングされた LP で処理されたイベントを, 一つの連続した処理済みイベントリストとして管理する. このリストを用いて保存する LP の状態を決定し, optimistic モードでシミュレーションを実行した際イベントの発生順序に矛盾が生じたときのローバック処理を実現する. よって, ロールバック処理は KP 同士の間で行われる. KP 数を増やすと管理する処理済みイベントリストが増えるため, ロールバック処理のために LP の状態を保存するオーバーヘッドが増えるが KP 上にマッピングされている LP が減るためロールバック処理の影響を受ける LP が減りロールバック数は少なくなる. 逆に KP 数を減らすと, 管理する処理済みイベントリストは少なくなり状態を保存するためのオーバーヘッドが少なくなるが KP 上にマッピングされている LP が増えるのでロールバック処理の影響を受ける LP が多くなりロールバック数が多くなる. そのため実行時間を短縮するためには適切な KP 数を設定する必要がある.

2.4 CODES: Enabling Co-Design of Exascale Strage Architectures

Co-Design of Exacscale Storage System Project [6] は米アルゴンヌ研究所にて実施されているプロジェクトである. エクサスケールアーキテクチャと分散データインテンシブサイエンス環境の設計を探求することを目的としており, ROSS を用いたシミュレータで使用可能なライブラリの開発及び提供を行っている. CODES は幾つかのモジュールからなる. 以下に, PPMDS のシミュレーションの際に用いた主要なモジュールについて述べる.

2.4.1 CODES configuration

LP の構成, パラメータ設定, その他のシミュレーションに必要なパラーメータの設定は構成ファイルを使用し CODES configuration system によって指定する. 構成ファイルのフォーマットには Key-Value ペアをグループ内に定義する形式を採用している.

2.4.2 LP mapping

codes-mapping API はユーザーの LP をグローバル LP ID 上にマッピングする. マッピングはグループごとに行われる . グローバル LP ID を使用することによって LP を一意に指定することが可能になり, グローバル LP ID を使用して LP 同士のメッセージの送受信を行う.

2.4.3 Simplenet model

Simplenet モデルは basic queued point-to-point latency/bandwith ネットワークをモデル化したものでネットワークを介した通信の再現を行う場合に使用する. パケットロスなどの不安定なネットワークを再現する機能はなく, 理想的なネットワーク環境が利用されることを仮定している. Simplenet モデルはネットワーク開始遅延 (ns) とバンド幅 (MB/s) のパラメータを構成ファイルで定義することができる.

2.5 PPMDSsim

PPMDSsim[1] は CODES/ROSS を使用し, PPMDS のファイル作成, 参照, 削除のシミュレーションを実装. シミュレーションにより PPMDS のスケーラビリティを評価している. 単一ディレクトリへの並列ファイル作成シミュレーションの結果を示す. 単一ディレクトリへの並列ファイル作成シミュレーションでは, サーバー LP 数とクライアント LP 数を増加させるに従い, ファイル作成数の上限が向上することが確認されている.

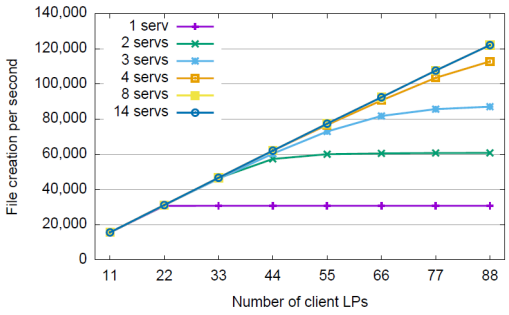

図 5　単一ディレクトリへの並列ファイル作成シミュレーション：実環境の評価を再現させた結果 (文献 [1] より引用)

図 6　単一ディレクトリへの並列ファイル作成シミュレーション：クライアント数を増加させた結果 (文献 [1] より引用)

2.5.1　単一ディレクトリへの並列ファイル作成シミュレーション

単一ディレクトリへの並列ファイル作成のワークロードについて, PPMDSsim[1] を用いてシミュレーションを実行している. 実システムの結果（図 2）と比較するためにサーバ LP 数は最大 14 ノード, クライアント LP 数は最大 88 ノードに設定している. **図 5** に実行結果を示す. 実環境による結果と比較すると, 現状では各ノードの負荷の状況を完全には再現できていないことや, 理想的なネットワーク環境が利用されることを仮定していることから秒間の操作数は完全には一致していないが, 性能限界が発生する箇所と, 値の変化の様子は概ね再現できていることがわかる. 次に, 実システムで性能限界の判明していない 88 クライアント以降の性能変化を調べるために, クライアント LP 数を増やしてシミュレーションを実行している. **図 6** に実行結果を示す. 実行結果からサーバ台数が 4 サーバ以降の場合でも, クライアント数の増加により性能限界が発生する箇所が存在するという結果を示した.

2.5.2　並列シミュレーションの実行時間

シミュレーションを並列に実行したときのシミュレーションの実行時間の結果を**図 7**, **図 8** に示す. 並列シミュレーションに要する実行時間を, プロセス数を変化させつつ同期プロトコル別に計測している. 実行時のプロトコルには, ROSS で使用できる conservative と optimistic の 2 プロトコルを選択している. クライアント数は 1024, 2048, 4096 ノード, サーバ台数を 128 台に設定し, プロセス数を 1〜64 の間で変化させてシミュレーションを実行している. 同期プロトコルに optimistic モードを利用した際は, 並列度を上げることによってシミュレーション時間が短縮されることが示されている. conservative モードを利用した場合, はプロセス数が増えるにつれて実行時間が長くなるという結果を示した. PPMDSsim では並列シミュレーションを実行した際, プロセス数を増加させたときに実行時間が短縮されないケースがいくつか確認された.

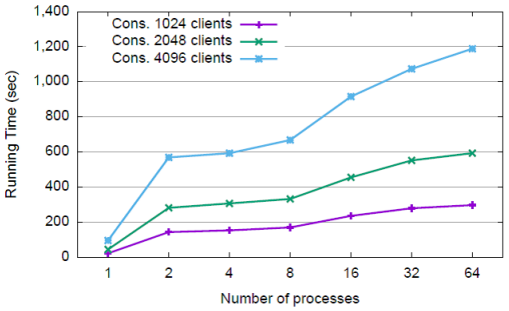

図 7　conservative プロトコル使用時の並列シミュレーションの実行時間 (文献 [1] より引用)

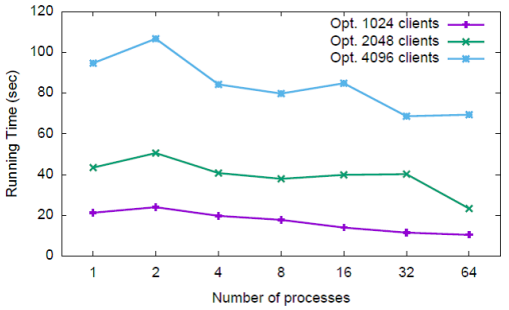

図 8　optimistic プロトコル使用時の並列シミュレーションの実行時間 (文献 [1] より引用)

2.6　関連研究: YARNsim

Apache Hadoop YARN[8] をシミュレーション対象として YARNsim[7] を提案している. YARN のキーコンポーネントを application module, YARN module, MapReduce module, HDFS module としてモデリングし並列離散イベントシミュレーションによって性能評価している. これにより, 開発者がより簡単にシステムの効率を評価すること, そしてシステム構成の違いによるトレードオフの理解を可能にすることを目的としている. YARNsim は並列離散イベントシミュレータ CODES/ROSS を用いて実装されている. Teragen benchmark, Terasort, WordCount を用いた, 実システムとの比較においてもほとんどのケースで誤差の範囲が 10%以内に収まるという高い精度のシミュレーション結果を示している.

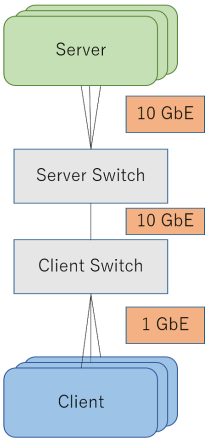

図 9　PPMDS の評価環境

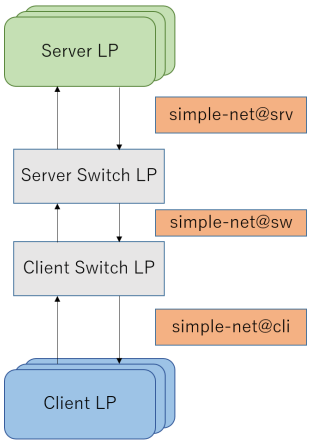

図 10　PPMDS シミュレータで再現した評価環境

2.7　関連研究: Oversim

OverSim[9] はオーバーレイネットワークシミュレーションのフレームワークである．OverSim にプロトコルを実装することによって，オーバーレイネットワークシミュレータとして使用することができる．離散イベントシミュレータ OMNeT++[10] に基づいた C++で実装されている．OverSim はスケーラビリティ，フレキシビリティ，アンダーレイネットワークのモデリング，シミュレーションコードの再利用，統計情報の分析機能，GUI を実装することを仕様として要求している．スケーラビリティを評価するために，P2Psim[11] を比較対象としてシミュレーションを実行し，実行時間とメモリフットプリントについての比較を行っている．10,000 ノードのネットワークのシミュレーションでは OverSim は P2Psim の約 20 倍実行時間が短いという結果を示した．100,000 ノードのネットワークのシミュレーションでは P2Psim が数時間プログラムを実行した後，バッファオーバーフローが原因であると考えられるクラッシュを起こした，しかし OverSim はシミュレーションを一時間以内に完了することができた．

3.　PPMDS シミュレータ

本章では，PPMDSsim でサポートされていなかった PPMDS によるディレクトリ操作をシミュレーションによって再現するためのシミュレータについて説明する．**図 9** に PPMDS シミュレータで再現した評価環境を，**図 10** に PPMDS シミュレータで再現した評価環境を示す．

3.1　Client LP

Client LP は，クライアントノードをモデリングした LP である．Client LP はイベントを Server LP 宛に送信する．また，シミュレーション後に統計情報を収集し出力する．ディレクトリ作成，ディレクトリ参照，ディレクトリ削除の操作に対応している．

Client LP は LP ごとに一意に割り当てられた LP-ID を用いて，”dir_＜ LP-ID ＞_＜通し番号＞” というディレクトリをルートディレクトリに作成する，作成数は設定ファイルで設定する．作成したディレクトリはディレクトリ参照，ディレクトリ削除操作で使用する．Client LP で実行されるイベントを以下に示す．Client LP のイベントのワークフローを図に示す．

CLI_KICKOFF
シミュレーションを開始するためのイベント．

CLI_REQ_DIR_INODE
Server LP にディレクトリ作成先ディレクトリのディレクトリ inode エントリを要求する．Server LP で実行されるイベントとして GET_INODE イベント，コールバックイベントとして CLI_REQ_DIR_CREATION イベントを指定する．

CLI_REQ_DIR_CREATION
親ディレクトリの inode エントリから，ディレクトリ作成のための inode エントリを作成し，inode エントリ名からハッシュ関数を用いて格納先サーバの LP_ID を計算する．Server LP で実行されるイベントを STORE_INODE イベントを指定しメッセージを格納先サーバの LP_ID に送信する．

CLI_RECV_DIR_CREATION_RESULT
Server LP からディレクトリ作成完了のメッセージを受け取り，ディレクトリ作成数を表すカウンタをインクリメントする．

CLI_REQ_DIR_STAT
Server LP にディレクトリ作成イベントで作成したディレクトリの inode エントリを要求するメッセージを送信する． Server LP で実行されるイベントを GET_INODE，コールバックイベントとして CLI_RECV_DIR_STAT を指定したメッセージを作成する．

CLI_RECV_DIR_STAT_RESULT
Server LP からディレクトリ参照完了のメッセージを受け取り，ディレクトリ参照数を表すカウンタをインクリメントする．

CLI_REQ_DIR_REMOVE
ファイル作成イベントで作成した，ディレクトリの削除を要求するメッセージを Server LP に送信する． Server LP で実行されるイベントを DIR_REMOVE，コールバックイベントとして CLI_RECV_DIR_REMOVE_RESULT を指定したメッセージを作成する．

CLI_RECV_DIR_REMOVE_RESULT
Server LP からディレクトリ削除完了のメッセージを受け取り，ディレクトリ削除数を表すカウンタをインクリメントする．

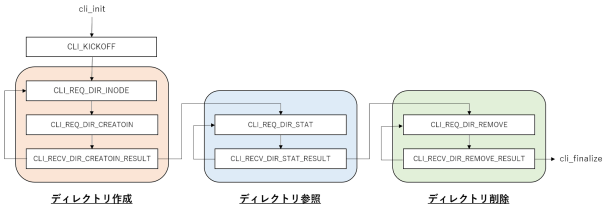

図 11　ディレクトリ操作シミュレーションのイベントのワークフロー

3.2　Switch LP

Switch LP は, クライアントスイッチ, サーバスイッチをモデリングした LP である. メッセージには, 最終的な宛先がクライアントかサーバかを表すフィールドを持つ. Switch LP はこのフィールドからメッセージの送信先を決定する.

3.3　Server LP

Server LP はサーバノードを表現する LP である. Server LP は Client LP から送信されたイベントを処理する. 各 Server LP は Key-Value store を持っており, inode エントリの格納, 取得, 削除を行う.

現在, PPMDS ではディレクトリ分散先サーバはディレクトリの分散先サーバは存在するすべてのメタデータサーバに分散する設定となっているが, シミュレータでは分散先サーバ台数のディレクトリ操作への影響を評価するために設定ファイルから分散先サーバ台数を変更できるように実装を行った. Server LP で実行されるイベントを以下に示す. ディレクトリ作成シミュレーションのワークフローを図に示す.

GET_INODE
Client LP から送信されたメッセージ内で指定されたディレクトリ名の inode エントリを Key-Value store から取得し, Client LP に結果のメッセージを送信する.

STORE_DIR_INODE
Client LP から送信されたディレクトリ作成要求のメッセージで指定されたディレクトリ名の inode エントリと分散先サーバリストを Key-Value store 上に保存する. その後, ディレクトリ分散先サーバ LP に STORE_SIDLIST イベントを送信する.

STORE_SIDLIST
ディレクトリ分散先サーバリストを Key-Value Store に保存する. STORE_SIDLIST_RESULT イベントをディレクトリを作成したサーバに送信する.

STORE_SIDLIST_RESULT
ディレクトリ分散先サーバリストの保存が成功したレスポンスを受け取る. 分散先サーバリストを保存していない Server LP がある場合はその Server LP に STORE_SIDLIST イベントを送信する, すべてのサーバに分散先サーバリストが作成されたことを確認すると, DIR_CREATION_RESULT イベントを作成し自身の LP に送信する.

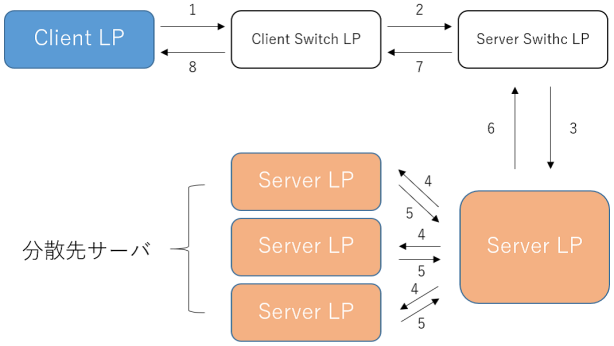

図 12　ディレクトリ作成シミュレーションのワークフロー

DIR_CREATION_RESULT
Client LP にディレクトリ作成完了のメッセージを送信する.

DIR_REMOVE
Key-Value store からディレクトリ inode を削除する, その後ディレクトリ分散先サーバリストを削除するイベントをスケジューリングする.

REMOVE_SIDLIST
Key Value Store からディレクトリ分散先サーバリストを削除する, その後ディレクトリ分散先サーバ LP に REMOVE_SIDLIST イベントを送信する.

REMOVE_SIDLIST_RESULT
ディレクトリ分散先サーバリストの削除が完了したメッセージを受け取る, まだ削除していない分散先サーバリストが存在する場合は REMOVE_SIDLIST イベントを Server LP に送信し, すべてのサーバの分散先サーバリストの削除が終了した場合は, DIR_REMOVE_RESULT イベントを自身の LP に送信する.

DIR_REMOVE_RESULT
Client LP にディレクトリ削除完了のメッセージを送信する.

Server LP でのトランザクション処理は PPMDSsim と同様の方法で実装を行った, 各 Server LP が KyotoCabinet[12] の Key-Value store を持ち, この Key-Value store に対して inode の格納処理や取得要求を行う. Key-Value store への操作は文献 [4] の手法を用いたトランザクション処理を行う. このトランザクション手法を実現するためにのデータ構造と関数を transaction module として実装した. Server LP での Key-Value store への操作は, Kyoto Cabinet が提供するインターフェースを直接使用せずこの transaction module の関数を使用する.

表 1　シミュレータに使用したソフトウェア環境

ROSS	Rev.8af4b41
CODES	Ver.0.5.3
MPI	Open MPI Ver. 2.0.1

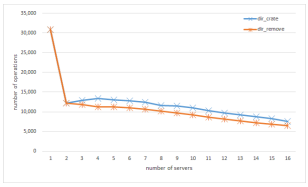

図 13　ディレクトリ作成, 削除

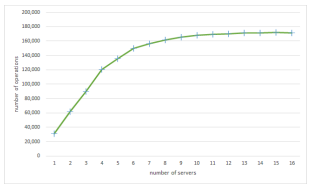

図 14　ディレクトリ参照

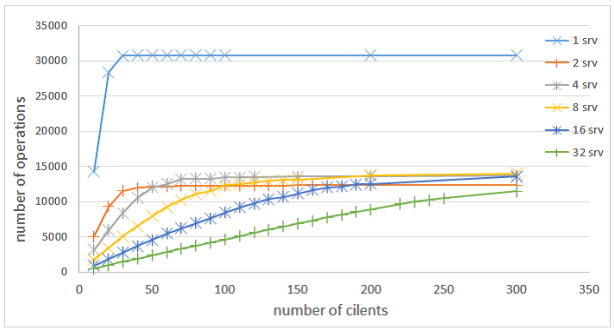

図 15　単一ディレクトリへの並列ディレクトリ作成シミュレーション

3.4　設定ファイル

シミュレータを実行する際に, 設定ファイルを指定する必要がある. 設定ファイルには LP 数, simple-net モデルで用いられるパラメータ, ROSS で用いられるパラメータ, シミュレータ独自のパラメータを記述することができる.. 設定ファイルを使うことにより Client LP 数や Server LP 数などをシミュレーションごとに変更したい場合もプログラムを変更する必要はなく, 設定ファイル内のそれぞれの値を変更することによって Clietn LP 数や Server LP 数などを変更することができる.

4.　シミュレーション結果

本章では, シミュレータによる PPMDS の性能評価を実行した示す. シミュレータ構築及び実行環境として使用したソフトウェアを表 1 に示す.

4.1　実システムでの性能評価をシミュレーションで再現した結果

単一ディレクトリへの並行ディレクトリ操作について PPMDS シミュレータを用いてシミュレーションを実行した. 実環境での評価と同じ環境を再現し比較するために, クライアント数を 88 に固定し, サーバ数を 1〜16 まで変化させ, ディレクトリ分散先サーバ台数を全サーバとしたときのディレクトリ作成, 参照, 削除についてのシミュレーションを行った結果を**図 13** 及び, **図 14** に示す. 実行結果と実システムでの結果を比較すると, シミュレーションでの性能評価, 実システムでの性能評価どちらの結果もディレクトリ作成とディレクトリ削除に関してはサーバ台数が増えるにつれて性能が低下していき, ディレクトリ参照に関してはサーバ台数が増えるにつれて性能が向上していくという結果が得られた.

4.2　単一ディレクトリへの並列ディレクトリ作成のシミュレーション

単一ディレクトリへの並列ディレクトリ作成のシミュレーションについて実システムで性能評価を行っていない 88 クライアント以降の性能についてシミュレーションによる性能評価を行った. クライアント数を 10〜500, サーバ数を 1〜32 に設定し, 分散先サーバ台数を全サーバとしてシミュレーションを行った, 結果を図 15 に示す. シミュレーション結果からサーバ数が 2 台以上のとき, クライント数を増やすことによって, サーバ数にかかわらず一定の性能限界を示すという結果が得られた.

4.3　分散サーバ台数を変更させたときのシミュレーション結果

ディレクトリ分散先サーバ台数を変更させたときの, ディレクトリ作成についてのシミュレーションを行った. ディレクトリ分散先サーバ台数を 4, 8 16 台と変更していきそれぞれの分散先サーバ台数においてディレクトリ作成シミュレーションを実行した. クライアント数を最大 500, サーバ数を 16 台に固定しシミュレーションを実行した. 実行結果を**図 17** にしめす, また分散先サーバ台数によるファイル作成数の変化を**図 16** に示す. 実行結果からディレクトリ分散先サーバ台数が少なくなると, 秒間のディレクトリ作成数が大きくなるという結果を示した. これは, ディレクトリ分散先サーバ台数が少なくなると分散先サーバに分散先サーバリストを作成するコストが減るためだと考えられる. また, 分散先サーバ数が少なくなるとファイル作成数は少なくなる, これはファイルを分散できるサーバが少なくなるためだと考えられる.

4.4　ネットワーク開始遅延及びバンド幅の性能への影響のシミュレーションによる調査

PPMDS のファイル作成シミュレーションを, バンド幅, ネットワーク開始遅延を変更し秒間あたりのファイル作成数が飽和するまでクライアント数を増やしてシミュレーションを実行. 変更する値は InfiniBand を想定し, バンド幅を 10 倍, ネットワーク開始遅延を 1/10 倍とした. シミュレーション結果を図 18 に示す. グラフの横軸をサーバ数, 縦軸を秒間あたりのオペレーション数としてバンド幅だけを変更したもの, ネットワーク開始遅延だけを変更したもの, どちらも変更したものについてのシミュレーション結果と変更を行わなかったものの比較を行った. バンド幅を

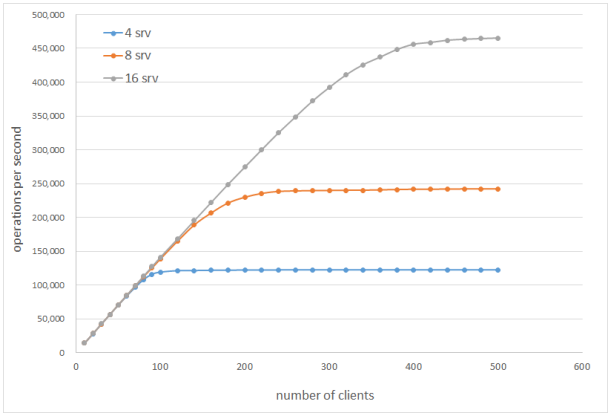

図 16　分散先サーバ台数による秒間のファイル作成数の変化

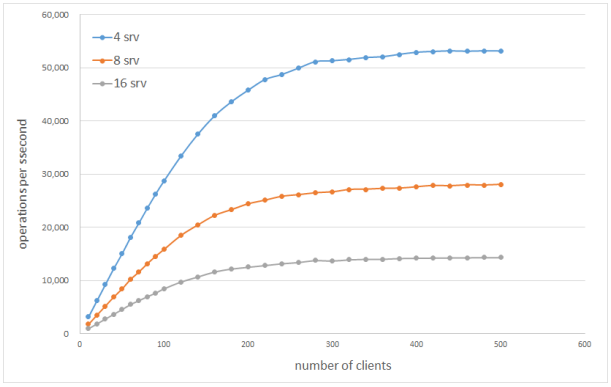

図 17　ディレクトリ分散先サーバ台数を変化させたときのディレクトリ作成シミューレーション

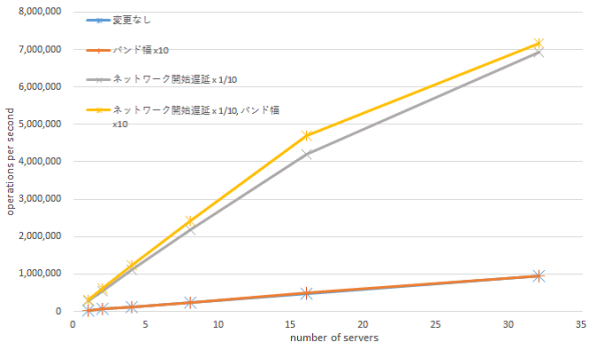

図 18　バンド幅, ネットワーク開始遅延を変更しシミュレーションを実行した結果

変更したときの結果は変更を行わなかったときの結果とほとんど性能が変わらなかった．ネットワーク開始遅延を変更した場合は，秒間あたりの操作数が大きく増えた．よって，シミュレーション結果から PPMDS の性能はネットワーク開始遅延がボトルネックになっている可能性があるということが考えられる．

4.5　並列シミュレーションの実行

PPMDSsim では，ファイル作成の並列シミュレーションを実行しているが，クライアント LP 数が 1024 の条件において 64 プロセスによる並列化で約 2 倍程度の高速化にとどまっていた．また．プロセス数を増やすことによって実

表 2　並列シミュレーションに使用した計算ノード

OS	Linux 2.6.32(CentOS6.7)
CPU	Intel Xeon E5620 2.40GHz x 2 sockets
Memory	24GB
Network	InfiniBand 4X QDR

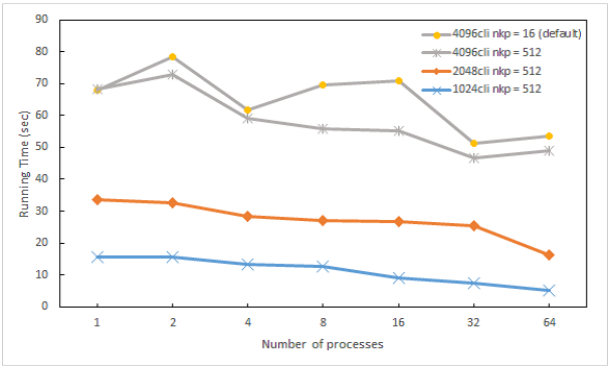

図 19　optimistic モードで実行したときの実行時間

行時間が増加するケースも確認された．シミュレーション並列化の際に，実行時間に影響を及ぼす要因が何であるかを調べる必要があったため，PPMDSsim では変更を行わずに実行していた ROSS での並列シミュレーション実行時のパラメータを変更し並列シミュレーションを実行し実行時間の比較を行った．パラメータを変更しシミュレーションを実行した結果，KP(カーネルプロセス) の値が実行時間に影響しているということがわかった．nkp(number of kernel processes) の値を 512 に変更したときの実行時間を図 19 に示す．実行結果から，プロセス数を増やしたときに逆に実行時間が増えてしまうケースが減った．また実行時間の短縮がみられた．並列シミュレーションを実行する環境は表 2 に示す計算機を 8 台用意し構築した．1〜8 プロセスのときは 1 ノード，16 プロセスのときは 2 ノード，32 プロセスのときは 4 ノード，64 プロセスのときは 8 ノードにプロセスがマッピングされる．

5.　まとめと今後の課題

本研究では，分散メタデータサーバ PPMDS でのディレクトリ操作について，並列離散イベントシミュレーションフレームワーク ROSS 及び，CODES を用いてシミュレーションを実装し，クライアント数サーバ数を増やしたときの性能限界についての評価を行った．また実システムで評価がされていない分散先サーバの性能への影響，バンド幅ネットワーク開始遅延の性能への影響の調査を行った．

今後の課題は，現在サポートしていない readdir などの操作についてもサポートを行い，より多くの観点から性能評価を行うことを可能にすることが望ましいと考えられる．また，現在は単一ディレクトリへのファイル操作，ディレクトリ操作のワークロードのシミュレーションしか実行することが出来ないため，複数ディレクトリへのファイル操作，

ディレクトリ操作などの異なるワークロードのシミュレーションも実行できるようにする必要があると考えられる. 並列シミュレーションに関しては, パラメータを変更することによって実行時間の短縮がみられたが optimistic モードで実行した際, クライアント LP 数が 1024 の条件で 64 プロセスの並列化で約 3 倍程度の高速化に留まっている. 今後より高速化を行う必要があると考えられる.

謝辞 本研究の一部は JST-CREST「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」,「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」,「広域撮像探査観測のビッグデータ分析による統計計算宇宙物理学」による.

参考文献

[1] 桐井祐樹, 建部修見, Robert Ross. CODES/ROSS による分散ファイルシステムのための分散メタデータサーバ PP-MDS の評価. 情報処理学会研究報告, Vol. 2015-HPC-152, No.7, pp.1-9, 2015.

[2] 平賀弘平, 建部修見. ノンブロッキングトランザクションに基づく分散ファイルシステムのための分散メタデータサーバの設計と実装. 情報処理学会研究報告, Vol.2012-HPC-135, No.28, pp.1-9, 2012.

[3] Maurice Herlihy, Victor Luchangco, Mark Moir, William N. Scherer, III. Software transactional memory for dynamic-sized data structures. Proceedings of the twenty-second annual symposium on Principles of distributed computing, pp. 92-101, 2003.

[4] 熊崎, 宏樹, 津邑 公暁, 齋藤 彰一, 松尾 啓志. 分散キーバリューストアを対象としたオブストラクションフリートランザクションの実装情報処理学会研究報告, Vol. 2011-OS-118, No.16, pp1-7, 2011.

[5] J Cope, N Liu, Samuel Lang, C D Carothers, and Robert B Ross. ROSS: A high-performance, low memory, modular time warp system. Proceedings Fourteenth Workshop on Parallel and Distributed Simulation, 2000.

[6] J Cope, N Liu, Samuel Lang, C D Carother, and Robert B Ross. CODES: Enabling Co-Design of Multi-Layer Exascale Storage Architectures. Proceedings of Workshop on Emerging Supercomputing Technologies 2011 (WEST 2011), 2011.

[7] Ning Liu, Xi Yang, Xian-He Sun, J. Jenkins, and R, ROSS. YARNsim: Simulating Hadoop YARN. Proceedings of 15th IEEE/ACM international Sumposium on Cluster, Cloud and Grid Computing (CCGrid), pp 637-646, 2015.

[8] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, Eric Baldeschwieler, Apache Hadoop YARN: Yet Another Resource Negotiator, Proceedings of ACM Symposium on Cloud Computing, p. 16, 2013.

[9] Ingmar Baumgart, Bernhard Heep, Stephan Krause. OverSim: A Flexible Overlay Network Simulation Framework. Proceedings of 2007 IEEE Global Internet Symposium, pp. 79-84, 2007.

[10] Andras Varga. The OMNeT++ Discrete Event Simulation System. Proceedings of the European Simulation Multiconference, pp. 319-324, 2001.

[11] J. Li, J. Stribling, R. Morris, M. Kaashoek, and T. Gil A performance vs. cost framework for evaluating DHT design tradeoffs under churn. Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies. vol. 1, Mar. 2005, pp. 225236.

[12] FAL Labs. Kyotocabinet http://fallabs.com/kyotocabinet/