情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2015-HPC-152 No.7
2015/12/16

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 PPMDS の評価

桐井 祐樹[1,a)]　建部 修見[2,b)]　Robert Ross[3,c)]

概要：HPC 分野において，大規模データ処理の需要は年々増加しており，分散システムソフトウェアの大規模化は今後も進んでいくと考えられる．実システムを利用した性能評価においては，コンポーネントの構成変更が柔軟に行えないという問題や，利用できるノード数に限りがあるため性能限界の測定が困難であるといった問題が発生しうる．この問題を解決するために，シミュレータを利用し，性能に関わるパラメータやコンポーネントの構成を柔軟に変更可能にする性能評価の手法が考えられる．本研究では，並列離散イベントシミュレーションフレームワーク CODES/ROSS を用いた大規模分散システムのシミュレーションを提案する．今回，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 のシミュレータを実装し，大規模環境における分散メタデータサーバのスケーラビリティ評価を行った．本稿では，本提案に基づくシミュレーションスタディの有効性と，今後の課題について議論する．

1. はじめに

分散システムの開発において，実験を行い性能評価を行うことは非常に重要であり，欠かすことのできないものである．分散システムの性能は，システムの構成や利用する分散プロトコルによって大きく左右される．そのため，構成やパラメータなどの実行条件を変え，最適な設計を決定したいという需要がある．一般的な分散システムでは，サービスを提供するサーバの数を増やすことにより性能をスケールアウトさせ，より多くのリクエストを処理することを可能にする．システムのスケーラビリティは，実運用に向けてハードウェアやネットワーク資源の調達を行う際にも考慮しなければならない重要な指標である．

分散システムの性能評価の手法は，評価環境上で対象のシステムを実際に動作させる手法が一般的であるが，この手法は前述の評価項目の十分な評価を妨げる要因となりうる．最適な設計を決定するためには，パラメータを変更したソフトウェアを各ノードに配置することや，システム構成を変更する必要がある．これらの変更を評価の度に行う必要があり，効率が悪く非常に手間がかかるという問題が発生する．また，評価環境上にクライアントとして利用可能なノード数が十分に確保できない場合，サーバの性能を飽和させるだけのリクエストを発行することができないという問題が発生する．システムの性能限界を知ることができないため，サーバの分散化によるスケーラビリティの変化について，十分な評価が行えないという問題につながる．

ハイパフォーマンスコンピューティングの分野において，大規模データ処理の需要は年々増加している．今後も分散システムの大規模化は進むと予想され，前述の問題のために実環境での性能評価を行うことは困難になってゆくと思われる．そのため，実環境を用意することなく仮想的に性能を見極める手法が求められる．

本研究では，並列離散イベントシミュレーションによる大規模分散システムの性能評価について考察する．大規模分散システムの一例として，平賀ら [1] 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 PPMDS を，シミュレーションによる評価対象とした．PPMDS のスケーラビリティ評価においては，評価環境上に確保したクライアント数が少ないために，サーバ性能を飽和させることができていないという問題がある．我々は，並列離散イベントシミュレーションフレームワークを用いたシミュレーションプログラムを実装し，シミュレーションスタディにより大規模環境における PPMDS のスケーラビリティを評価した．

[1] 筑波大学 情報学群 情報科学類
College of Information Science, School of Informatics, University of Tsukuba
[2] 筑波大学システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba
[3] Mathematics and Computer Science Division, Argonne National Laboratory
[a)] kirii@hpcs.cs.tsukuba.ac.jp
[b)] tatebe@cs.tsukuba.ac.jp
[c)] rross@mcs.anl.gov

以下，第 2 章では，シミュレーションによる性能評価に取り組んでいる関連研究について述べる．第 3 章では，予備知識として，今回シミュレーション対象として選択した PPMDS と，並列離散イベントシミュレーションについて解説する．第 4 章では，我々が提案する並列離散イベントシミュレーションフレームワークを用いたシミュレーションについて，関連技術及びシステムのシミュレータの構築手法について述べる．第 5 章では，本研究で実装したメタデータサーバシミュレータ PPMDSsim の概要と，設計及び実装について述べる．第 6 章では，単一ディレクトリへのメタデータ操作のスケーラビリティについてシミュレーションによる結果を示し，実システムとの結果の比較を行う．第 7 章では，まとめと今後の課題について述べる．

2. 関連研究

シミュレーションを用いた大規模分散システムの評価は，今日までに様々な試みが行われてきた．特に P2P システムで用いられるオーバレイ・ネットワークは，近年システムの規模が急速に拡大しており実環境でのアルゴリズム評価が困難であるため，シミュレーションによる評価が必要不可欠となっている．OverSim [2] や Overlay Weaver [3] が代表的である．

OverSim は，P2P ネットワークのシミュレータであり，離散イベントシミュレータ OMNet++ [4] をベースに開発されている．分散ハッシュテーブルのアルゴリズム Chord によるネットワークを，最大で 100,000 ノードの規模でリアルタイムでシミュレーションすることが可能である．

Overlay Weaver は，オーバレイ・ネットワークを構築するためのツールキットである．構造化オーバレイのアルゴリズムを少ないコード量で実装でき，シミュレーションのために実装したプログラムを実ネットワーク上でも動作させることが可能である．また，Overlay Weaver は分散環境エミュレータを提供しており，1 台の計算機上で数万のノードの動作をエミュレートすることができる．これらのネットワークシミュレータは，ネットワークに関する評価に特化したものであり，一般的な並列分散処理を扱うものではない．

杉野らによる MapReduce を用いた大規模分散システムのシミュレーション [5][6] は，P2P ファイル共有プロトコルである Gnutella の通信シミュレーションを行うものである．シミュレーションの実行基盤に汎用分散処理システムを用いることにより，高いスケーラビリティと耐故障性が期待できる．文献 [6] では，時刻管理アルゴリズムとして Time Warp [7] を導入することで，高効率かつ高速なシミュレーションを実現している．

YARNsim [8] は，並列離散イベントシミュレーションを用いた Apache YARN のシミュレータであり，開発には後述する ROSS 及び CODES が用いられている．YARN を用いたシステム設計とパフォーマンスチューニングを行う際の総合的なシミュレーションプラットフォームを提供するものである．また，既存の MapReduce シミュレーションシステムで対応されていない，プロトコルレベルの詳細な HDFS モデルを提供する．シミュレーションシステムは並列実行でき，大規模なシナリオの元でのシミュレーションが可能である．シミュレーションシステムと実システムとの Hadoop ベンチマークテストの結果は，殆どのテストケースにおいて誤差 10% 未満であり，高い精度でのシミュレーションを実現している．

3. 予備知識

3.1 分散メタデータサーバ PPMDS

本節では，シミュレーションの対象とした PPMDS [1] の概要について述べる．PPMDS は，分散ファイルシステムのための分散メタデータ管理システムであり，ポストペタスケールの高性能システムをターゲットとしている．

PPMDS は，inode データ構造を単純な Key-Value 形式で表現する手法を導入している．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られるメタデータを格納する．この Key-Value ペアは，スケーラブルな分散 Key-Value store 上に格納される．ディレクトリ内のファイルをルックアップするには，ディレクトリの inode 番号を用いて Key を範囲検索することで，ディレクトリに含まれるファイルの inode エントリを得ることができる．メタデータの分散化のために，Key-Value store には inode エントリの他に，inode エントリの分散先サーバリストが格納される．これは，あるディレクトリに属する inode エントリが，どのサーバに保存されているのかを表すサーバ ID のリストであり，ディレクトリの inode 番号が Key となっている．ファイルの保存や参照などの操作を行う際は，まずファイルの格納先ディレクトリの分散先サーバリストを取得し，その後リスト内のサーバに対してファイル操作の要求を送信する．これらのメタデータ格納手法を用いることにより，従来のメタデータ管理システムで性能ネックとなっていた，同一ディレクトリへのファイル作成，参照，削除で高い並列性を実現した．

このメタデータ管理手法を用いると，ファイル作成，参照，削除などの操作は単一サーバの Key-Value の操作で実行可能なため高い並列性が実現される一方，ディレクトリの削除など複数のサーバにまたがる Key-Value の操作が必要な操作もある。複数のサーバにまたがる操作は，それぞれの操作がお互いに影響を与えず，失敗した場合はシステムに反映されないというアトミック性が要求される．PPMDS では，ダイナミックソフトウェアトランザクショナルメモリを応用したノンブロッキングトランザクションを，Key-Value store 上に実現する熊崎ら [9] の手法を採用

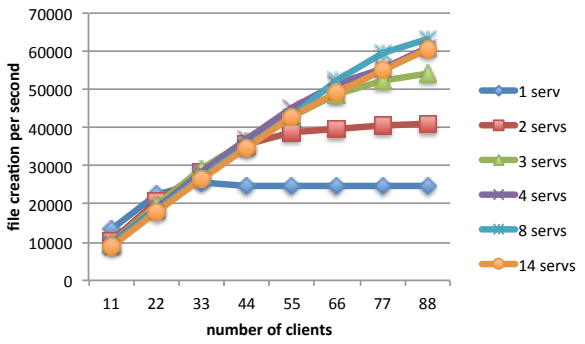

図 1 PPMDS による単一ディレクトリへの並列ファイル作成結果（文献 [1] より引用）

し，一部拡張を加えた上で導入している．これにより，グローバルなロックやオペレーションの直列化をすることなく，分散環境上でのアトミックなメタデータ操作を実現可能にした．

文献 [1] より引用した PPMDS の単一ディレクトリへの並列ファイル作成の結果を，図 1 に示す．複数のメタデータサーバを利用することで，秒間のファイル作成数が増えており，スケーラビリティの向上が見られる．しかし，評価環境で用いられているクライアント数（最大 88 クライアント）では，4 サーバ以上のケースでシステムの性能を飽和させることができていない．この評価結果より，現状では分散メタデータサーバの性能限界が判明しておらず，スケーラリビリティ評価が十分ではないと考えられる．そのため，より多くのクライアントノードを導入し，ファイル操作のリクエスト数を増やした上で，検証を行う必要がある．

3.2 並列離散イベントシミュレーション

離散イベントシミュレーションは，システムの動作のある瞬間に着目し，その瞬間に発生するイベントがもたらすシステムの状態変化を追跡するものである．シミュレータ上では，現実世界の時刻と同様に仮想的な時刻 (Virtual Time) が用意される．発生する各イベントには，イベントが生起した仮想時刻がタイムスタンプとして付与されている．イベントを発生時刻順に処理することで，システムの内部状態の変遷を追跡し，システムの挙動をシミュレーションすることができる．

並列離散イベントシミュレーションは，離散イベントシミュレーションを並列処理技術を用いて並列実行することで，実行時間の高速化や高効率化を図るものである．離散イベントシミュレーションの並列化では，シミュレーション対象のモデルに内在する並列性を元に，シミュレーションモデルを Logical Process (以下 LP) に分割して並列実行する．システムを構成する個々のノードやネットワーク機器などは，個別の LP としてモデリングされる．LP は，お互いにメッセージをやりとりすることで通信し合う．サーバへのジョブの到着やノード間のネットワーク通信は，LP 同士によるメッセージのやりとによって表現される．

並列離散イベントシミュレーションにおけるイベント処理の手法には，保守的 (conservative) な手法と楽観的 (optimistic) な手法の 2 種類が存在する．離散イベントシミュレーションでは，発生時刻順にイベントを処理する必要があるが，並列化によって LP は各計算ノードに割り当てられ別々にイベント処理を行うため，発生時刻順というイベント同士の因果関係が破られる可能性がある．保守的な手法では，イベントの処理順序に矛盾が発生しないように，シミュレーションシステム全体で同期をとりながらイベント処理を行う．一般的に，並列分散処理における同期のコストは大きいため，システム全体で同期をとることは非常に効率が悪い．そのため，保守的な手法の利用はスケーラビリティを制限する要因となる．楽観的な手法では，イベントの発生順序に矛盾が生じる事を許容し，投機的にイベントを処理する．発生順序に矛盾が生じた場合は，矛盾のない時点まで遡って再実行を行うことで整合性を保つ．この仕組みにより，同期回数を少なく抑えることができるため，楽観的手法は保守的手法より高速なイベント処理が可能である．

4. 提案手法

本研究では，並列離散イベントシミュレーションを用いた大規模分散システムのシミュレータの実装手法を提案する．シミュレーションプログラムの構築には，並列離散イベントシミュレーションフレームワーク Rensselars Optimistic Simulation System（以下 ROSS と表す） [10][11] と，CODES Project [12] が提供するライブラリ群を利用する．これらのフレームワーク及びライブラリを用いることの利点として，以下の 3 点が挙げられる．

1 点目は，大規模な並列シミュレーションを高速に実行することが可能である点である．大規模システムのシミュレーションでは，シミュレーションモデルの大規模化により，非常に大きなメモリ容量を必要とすることが考えられる．単一の計算ノードでのシミュレーション実行は，シミュレーションの大規模化を妨げる要因となりうる．そのため，複数の計算ノードを利用し，シミュレータを並列分散処理システムとして実行できることが望ましい．ROSS は，並列シミュレーションを想定して開発されており，複数の計算ノードを用いた大規模シミュレーションの実行に利用することが可能である．また，並列実行時の同期プロトコルとして，楽観的な時刻同期アルゴリズムである Time Warp をベースとした同期プロトコルが採用されている．並列実行時の同期回数を減らすことができるため，複数台の計算機を利用したシミュレーションにおいて，高いスケーラビリティが期待できる．

2 点目は，シミュレータの開発や実行に，特殊な環境を必要としない点である．ROSS を用いたシミュレーションプログラムは ANSI C を用いて記述するため，プログラムの実装に特殊な環境を必要としない．また，シミュレーションの並列実行には MPI が用いられるため，特殊な並列処理基盤システムなどを用意する必要がなく，一般的なクラスタ環境を活用することができる．大規模並列シミュレーションのための環境を，既存の計算機資源を有効に活用し，容易に構築することが可能となる．

3 点目は，シミュレーションモデルの構築や設定が容易であるという点である．ネットワークやストレージといった，多くのシステムでよく使われるモデルは，CODES Project が提供するライブラリ群により提供される．そのため，開発者はシミュレーションに必要なモデルを何もないところから構築せずに済む．また，シミュレータの設定を行うための機能が提供されており，簡潔な記法による設定ファイルの記述によって，シミュレーション実行時のパラメータを柔軟に変更することができる．

4.1 ROSS

ROSS は，並列離散イベントシミュレーションを，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ためのフレームワークである．シミュレータプログラムは，MPI を利用した並列プログラムとして動作する．ソースコードをコンパイルし得られる実行ファイルは，mpirun などの MPI による並列実行を行うコマンドを利用し実行することが可能である．

ROSS は，並列実行時の同期プロトコルとして，楽観的手法を使用することを想定し開発されている．選択可能な同期プロトコルとして，次の 3 つのモードが用意されている．

sequential シミュレーションは 1 プロセッサで逐次的に実行される．

conservative LP は複数のプロセッサに割り当てられ並列実行される．保守的手法により，イベント処理は発生時刻順に忠実に処理を行う．

optimistic LP は複数のプロセッサに割り当てられ並列実行される．楽観的手法により，投機的なイベント処理を行う．イベントの処理順序に矛盾が生じた場合は，イベントのロールバック処理を行い解決する．

シミュレータの構築は，各 LP 毎にイベントハンドラを用意し，イベントが発生した際にどのような状態変化をもたらすかをプログラムとして記述することによって行われる．optimistic プロトコル利用した並列シミュレーションを実行するためには，通常のイベントハンドラの他に reverse イベントハンドラを実装する必要がある．これは，通常のイベントハンドラで行われた変更を元に戻すための処理を記述するものであり，楽観的手法によりイベントを投機的に実行し，イベントのロールバック処理が必要になった場合に実行される．

4.2 CODES

CODES Project は，米アルゴンヌ国立研究所にて実施されているプロジェクトであり，エクサスケールストレージ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いる．

CODES Project では，ROSS を用いたシミュレータで利用可能なライブラリ CODES-base 及び CODES-net を開発している．これらは，ネットワークやストレージをシミュレートするためのモデルを，抽象化された API として利用可能にしたものである．CODES-base は，ハードウェアのシミュレーションモデルや，プログラム開発の際に便利なユーティリティを提供するライブラリ群である．提供されるユーティリティには，シミュレータのパラメータ設定を，設定ファイルによって行うためのコンフィギュレーションユーティリティや，シミュレータ実行時の各 LP の状態を統計情報として集計する機能などがある．CODES-net は，ネットワークトポロジと通信処理をモデル化したものである．LP 同士でメッセージを送受信する際に，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．現状では，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無く，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．CODES-net で提供される ”simple-net” モデルは，任意のネットワークレイテンシとバンド幅を設定することができ，Ethernet や InfiniBand による通信を再現することが可能である．他に Torus や Dragonfly，P2P ネットワークなど，幾つかのネットワークトポロジがモデル化され提供されている．

これらのライブラリは，ROSS の optimistic プロトコルを使用した並列シミュレーションで利用されることを想定し開発されている．提供される関数には，各モデルやユーティリティの本体の他に，LP の状態変更をロールバックするための関数が存在する．CODES の関数を，LP のイベントハンドラで呼び出し，対応する reverse イベントハンドラ内でロールバック用の関数を呼び出すことで，並列実行可能なシミュレータが容易に構築できる．

5. PPMDSsim

本章では，我々が提案する手法により実装した分散メタデータサーバ PPMDS のシミュレータ PPMDSsim の説明を行う．PPMDSsim は，PPMDS による分散メタデータ管理システムのシミュレータであり，並列離散イベントシミュレーションを用いてシステム全体の挙動を再現するものである．シミュレータの実装には，並列離散イベントシミュレーションフレームワーク CODES/ROSS を用いる．現在は，単一ディレクトリへの並列メタデータ操作の

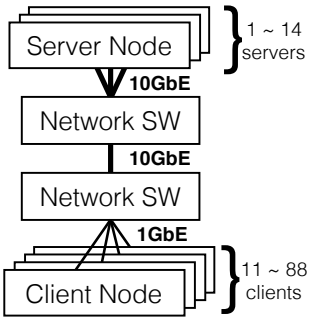

図 2 文献 [1] における PPMDS の実システムの評価環境の構成

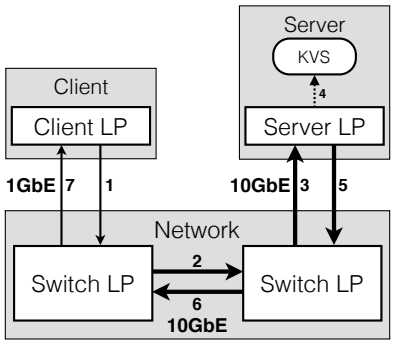

図 3 PPMDSsim のアーキテクチャ

シミュレーションに対応している．

5.1 設計

今回は，文献 [1] の性能評価で利用されたシステム構成を参考に，実際の PPMDS の評価環境を再現する設計を行った．PPMDS の実システムの構成を図 2 に示す．PPMDSsim では，各ノードやネットワーク機器をモデル化するために，それぞれ異なる種類の LP として定義した．PPMDSsim のアーキテクチャを図 3 に示す．各 LP から送信されるメッセージは，図中の数字の順に矢印が指す LP に対し送信される．以下では，ネットワーク通信の表現の手法と，各 LP 毎の動作の詳細について解説する．

5.1.1 ネットワーク通信の表現

図 2 に示すように，実際の PPMDS では Gigabit Ethernet と 10 Gigabit Ethernet の 2 種類のネットワークを利用している．PPMDSsim では，CODES-net が提供する simple-net モデルを利用し，ネットワーク通信のシミュレーションを行った．シミュレータ上で使用するネットワークを，クライアントとスイッチ間，サーバとスイッチ間，スイッチ同士の 3 つの区画に分けて，実際の PPMDS と同様のバンド幅を設定した．シミュレータ上では，各 LP には識別子として LP-ID という番号が割り当てられる．simple-net モデルを用いてメッセージを送信する際，メッセージの送信先は LP-ID によって指定することができる．

PPMDS では，クライアントやサーバ間の処理要求や，パラメータの受け渡しに RPC を利用している．現在，ROSS や CODES には，RPC を再現するための機能が存在しないため，RPC による通信が行われた際の挙動を LP 同士のメッセージ交換によって再現する必要があった．今回は，LP 同士でやり取りするメッセージの内部に，リモートプロシージャやコールバックとして実行されるイベントの種類と，RPC の引数や戻り値として使用するパラメータを含めるという手法をとった．以下では， Client LP から Server LP への RPC リクエストの送信を例に，メッセージ交換の処理手順にについて解説する．送信するメッセージの例をソースコード 1 に示す．

```
struct srv_msg {
    tw_lpid dst_lpid;     // 宛先の LP-ID
    tw_lpid src_lpid;     // 送信元の LP-ID

    /* 宛先で実行するイベント */
    enum srv_event srv_event_type;

    /* コールバックとして実行するイベント */
    int callback_event_type;

    /* 各イベント毎のパラメータ */
    union {
        struct { ... } event1;
        struct { ... } event2;
    } params;
}
```

ソースコード 1 Server LP へ送信するメッセージの例

(1) Client LP は Server LP へ送信するメッセージを構築する．宛先で実行するイベントには，リモートプロシージャとして実行したい Server LP のイベントを指定する．コールバックとして実行するイベントには RPC の戻り値を受け取る際に使用する Client LP のイベントを指定する．引数として渡すデータが存在する場合は，パラメータとしてメッセージ内に含める．
(2) 構築したメッセージを，目的の Server LP に対して送信する．
(3) Server LP はメッセージを受け取り，指定されたイベントを実行する．
(4) イベント実行後，Server LP は Client LP へ返送するメッセージを構築する．宛先で実行するイベントには，送られてきたメッセージに含まれる Client LP のコールバックイベントを指定する．イベントの実行により生されたデータは，パラメータとしてメッセージ内に含める．
(5) 構築したメッセージを，リクエストを発行した Client LP に対して送信する．
(6) Client LP はメッセージを受け取りコールバックイベントを実行する．パラメータに Server LP からの戻り値が含まれる場合はメッセージより取り出し利用する．

5.1.2 Client LP

Client LP は，クライアントノードを表現するための LP であり，主に Server LP へのファイル作成リクエストの送信と，シミュレーションの統計情報の収集と出力を行う．Client LP は，内部状態として，ファイル作成成功回数をカウントする変数を持っている．この変数は，ファイル作成の一連の処理が完了した際にインクリメントされ，最終的な値はシミュレータ終了時に統計情報として出力される．Client LP の実行するイベントの一覧を以下に示す．

REQ_DIR_INODE Server LP に対し，ファイル作成先ディレクトリの inode エントリを要求する．コールバックイベントとして REQ_FILE_CREATION イベントを指定する．

REQ_FILE_CREATION ディレクトリの inode エントリを受け取り，ファイル作成リクエストのためのメッセージを構築・送信する．作成するファイルのメタデータを生成し，ディレクトリの inode エントリの番号と共に Server LP に送信する．格納先サーバの LP-ID は，実際の PPMDS と同様にハッシュ関数を利用した分散によって決定される．

RECV_RESULT ファイル作成の成功リクエストを受け取り，自身の持つファイル作成成功回数のカウンタをインクリメントする．

シミュレータが実行されると，Client LP では初めに REQ_DIR_INODE イベントが実行される．REQ_FILE_CREATION イベントで受け取ったディレクトリの inode エントリは，Client LP 内でキャッシュされ，次回のリクエスト時に再利用される．RECVl_RESULT イベントでは，カウンタの値が指定されたファイル作成数に満たない場合，新たに REQ_FILE_CREATION イベントを実行する．その際は，ディレクトリの inode エントリはキャッシュ済みのものを使用する．

5.1.3 Server LP

Server LP は，サーバノードを表現するための LP であり，Client LP より送信されたメッセージの内容に応じてイベントを実行する．各 Server LP は Key-Value store を持っており，inode エントリの格納と取得を行うことができる．Server LP の実行するイベントの一覧を以下に示す．

GET_INODE inode エントリの取得リクエストを受け取った際に実行される．Client LP より指定されたディレクトリに対応する inode エントリを KVS から取得し，メッセージ送信元の Client LP に対して送信する．

STORE_INODE ファイル作成リクエストを受け取った際に実行される．Client LP により送信されたファイルのメタデータを inode エントリとして KVS に保存する．その後メッセージ送信元の Client LP に対し，ファイル作成の成功メッセージを送信する．

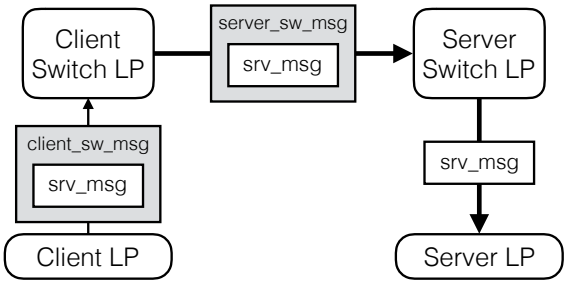

図 4 Switch LP によるメッセージの転送

STORE_INODE イベントでは，inode エントリを KVS へ格納する処理の遅延を再現するために，Client LP へのメッセージ送信を指定した時間遅らせるようにした．遅延時間は，CODES のコンフィギュレーションユーティリティを使い，設定ファイルへ値を書き込むことで変更することができる．

5.1.4 Switch LP

Switch LP は，ネットワークスイッチを表現するための LP である．実際の PPMDS では，図 2 に示すように，それぞれ使用できるネットワークバンド幅の異なる 2 つのネットワークスイッチを使用している．PPMDSsim では，この 2 つのネットワークスイッチを，それぞれ Client Switch LP と Server Switch LP に分けて実装した．

図 4 に，Switch LP によるメッセージ転送の例を示す．メッセージを送信したいノードは，自身が接続されている Switch LP 宛のメッセージ内に，本来送信したいメッセージの本体を格納し送信する．Switch LP 間での転送も同様に，もう一方の Switch LP 宛のメッセージの中に，本来送信したいメッセージの本体を格納し転送する．Switch LP 宛のメッセージには，メッセージの最終的な宛先が Client か Server かを表すフィールドを持っている．メッセージを受け取った Switch LP は，このフィールドの内容を元に，次にメッセージを転送する先を決定する．

5.2 実装

PPMDSsim は，並列離散イベントシミュレーションフレームワークとして CODES/ROSS を用い，C 言語で実装した．各 Server LP は，PPMDS の実システムと同様に Kyoto Cabinet [13] の Key-Value store を持つ．データベースタイプとして TreeDB を使用し，Key の前方一致など，実際の PPMDS と同様の操作が実現できるようにした．シミュレータの設定には，CODES-base が提供するコンフィギュレーションユーティリティを使用し，設定ファイルによりパラメータ変更ができるようにした．

6. 評価

6.1 評価環境

シミュレータによる性能評価を，表 1 に示すマシンで

OS	Linux 2.6.32 (CentOS 6.7)
CPU	Intel E5620 2.40 GHz
Memory	24GB
Network	InfiniBand 4X QDR
ROSS	Rev. 73c2ef1
CODES-base	Ver. 0.4.1
CODES-net	Ver. 0.4.0
MPI	Open MPI Ver. 1.8.8

表 1 シミュレータ実行のための評価環境

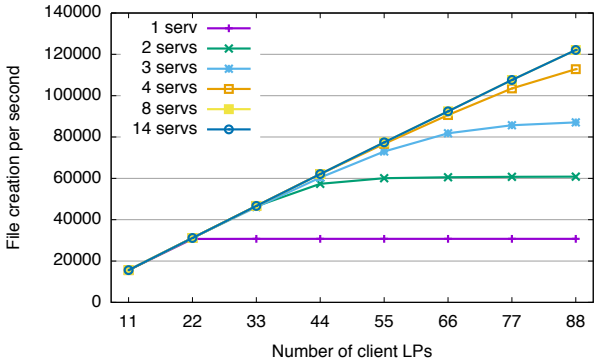

図 5 単一ディレクトリへのファイル作成のシミュレーション結果

行った．ROSS 及び CODES は，本稿執筆時点における最新のバージョンを使用した．LP 数やネットワークのバンド幅などの，シミュレーションの実行に関わるパラメータは，CODES のコンフィギュレーションユーティリティを利用し，設定ファイルの内容を書き換えることにより設定した．パラメータの一覧を表 2 に示す．

6.2 単一ディレクトリへの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行ファイル作成のワークロードについて，PPMDSsim を用いシミュレーションを実行した．実システムの評価で使用された評価環境を再現するために，サーバ LP 数は最大 14 ノード，クライアント LP 数は最大 88 ノードに設定した．図 5 に実行結果を示す．図 1 に示す実システムによる結果と比較すると，性能限界が発生する箇所と，値の変化の様子が概ね再現できていることがわかる．現状では，各ノードの負荷の状況などを完全に再現することはできないため，秒間のファイル作成数の値は実際の結果と完全には一致しない．

次に，実システムで性能限界が判明していない 88 クライアント以降の性能変化を調べるため，クライアントの LP 数を増やしシミュレーションを実行した．図 6 に実行結果を示す．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シミュレーション結果となった．性能限界への到達後は，サーバの性能は飽和し，秒間のファイル作成数の増減は見られなくなった．

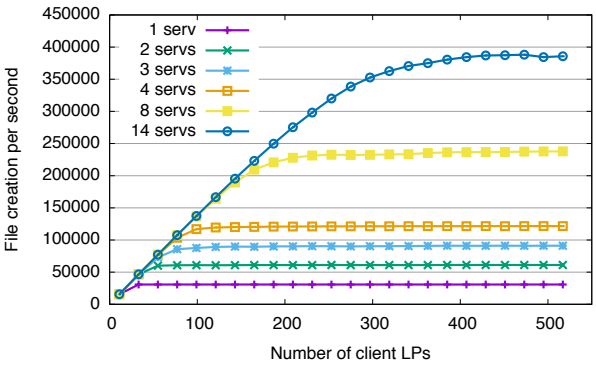

図 6 クライアント数を増加させた単一ディレクトリへのファイル作成のシミュレーション結果

6.3 性能のボトルネックの調査

シミュレータ上で変更可能なパラメータのうち，性能に関わる値を変化させ，システム性能のボトルネックとなる要因が何であるかを調査した．検証では，サーバ台数を 2 ノードに固定し，クライアント数を 11 から 88 ノードへ変化させ，パラメータの値を変化させながらシミュレータを実行した．表 2 に示すパラメータのうち，通信開始遅延 net_startup_ns と，inode エントリの格納完了のメッセージを送信するまでの遅延時間 store_inode_event_latency をそれぞれ変化させた．その他のパラメータは，表 2 に示す値を設定した．

通信開始遅延を変化させた結果を，図 7 に示す．遅延時間が短い場合，性能限界に達するのに必要なクライアント数は多く，性能限界到達時の秒間のファイル作成数は多いという結果になった．特に，遅延時間が 10,000 ns のケースでは，88 クライアントによるリクエストではシステム性能を飽和させることはできなかった．遅延時間を長くするにつれ，性能限界に達するのに必要なクライアント数は少なくなり，性能限界到達時の秒間のファイル作成数も少なくなった．次に，inode エントリの格納処理にかかる遅延を変化させ，同様の検証を行った．結果を図 8 に示す．グラフより，遅延時間が短い場合は，クライアントが増加した際のファイル作成数の増分が増えていることがわかる．性能限界の到達箇所に関しては，遅延時間によらず一定で，ファイル作成数の上限は約 60,000 ops/sec となった．

以上の 2 つの結果より，ファイル作成数の上限に関与し，システムのボトルネックを引き起こす要因がネットワークである可能性があることがわかった．実際の PPMDS の評価においては，ネットワーク構成や使用機器を変更した上での性能評価は行われていない．シミュレーションの結果を踏まえ，今後の実環境での性能評価においては，ネットワーク環境に対する調査検討が必要であると考えられる．

6.4 並列シミュレーションの実行と実行時間の計測

並列シミュレーションに要する実行時間を，プロセス数

パラメータ名	値	備考
packet_size	1500	ネットワークパケットのサイズ (byte)
net_startup_ns	32000	通信開始遅延 (ns)
net_bw_mbps@cli	125	クライアント - スイッチ間のネットワークバンド幅 (MB/sec)
net_bw_mbps@srv	1250	サーバ - スイッチ間のネットワークバンド幅 (MB/sec)
net_bw_mbps@sw	1250	スイッチ - スイッチ間のネットワークバンド幅 (MB/sec)
payload_size	512	メッセージ送信時の想定ペイロードサイズ (byte)
store_inode_event_latency	500000	inode エントリ格納完了のメッセージを送信するまでの遅延時間 (ns)

表 2 PPMDSsim で設定可能なパラメータの一覧

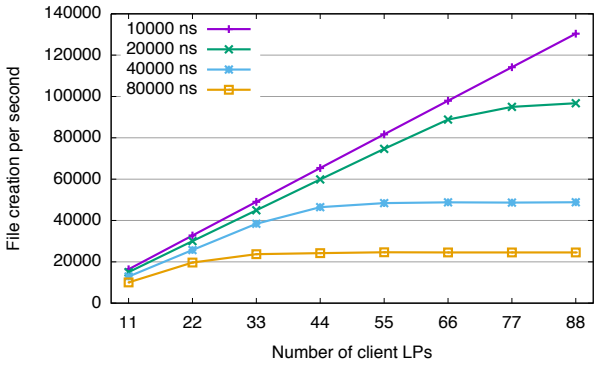

図 7 通信遅延時間ごとの秒間のファイル作成数

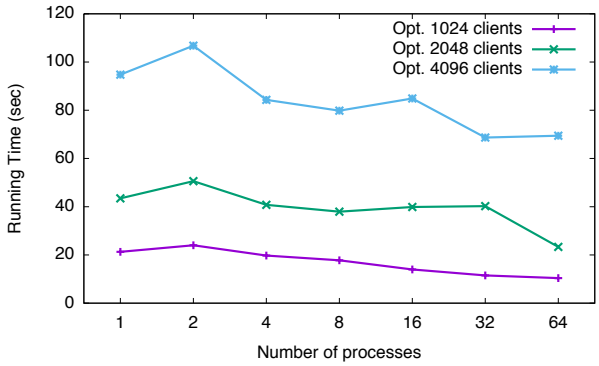

図 9 optimistic プロトコル使用時の並列シミュレーション実行時間

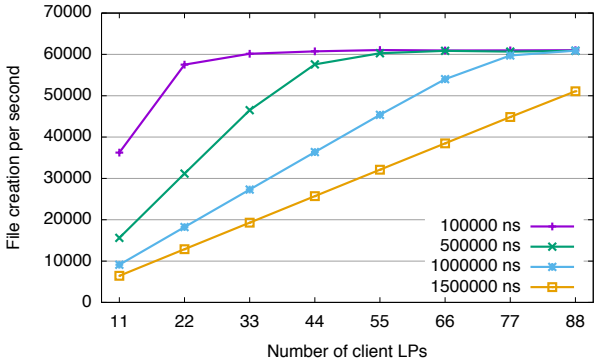

図 8 inode エントリ格納処理にかかる遅延ごとの秒間のファイル作成数

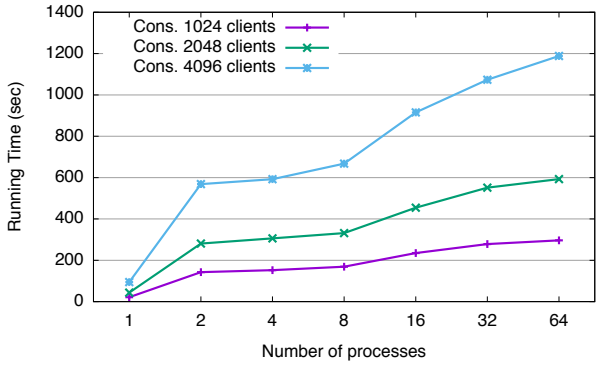

図 10 conservative プロトコル使用時の並列シミュレーション実行時間

を変化させつつ同期プロトコル別に計測した．並列実行時の同期プロトコルには，ROSS で使用できる conservative と optimistic の 2 つのプロトコルを選択した．なお，プロセス数が 1 の場合は並列化は行われないため，同期プロトコルは自動的に sequential が選択される．シミュレータの設定は，サーバ台数を 128 ノードに固定し，クライアントは 1024，2048，4096 ノードの 3 通りの構成を想定し設定した．その他のパラメータは，表 2 に示す値を設定した．評価環境は，表 1 に示す計算機を複数台用意し構築した．

図 9 は，同期プロトコルに optimistic モードを利用し，並列シミュレーションを実行した結果である．グラフの縦軸は，シミュレーションを実行に要した時間を示し，横軸は MPI による並列実行時のプロセス数を示す．プロセス数を 1 から 2 に増やした際に実行時間が長くなっているが，これは並列化に伴うオーバヘッドが導入されたためであると考えられる．以降は，プロセス数を増やすにつれて実行時間は短くなった．図 10 は，同様のシミュレーションについて，同期プロトコルに conservative モードを使用して実行した結果である．こちらは，プロセス数が増えるにつれて，シミュレーション実行時間が長くなっていることが確認できる．

大規模な分散システムのシミュレーションでは，大量のイベントやメッセージを処理するために多くのメモリ領域が必要になる．単体の計算ノードで利用できるメモリ領域は限られるため，複数の計算ノードの利用によって，より多くのメモリ領域が利用出来ることが望ましい．大規模分散システムの評価において，1 台の計算ノードで実行できない規模のシミュレーションを並列化する際には，ROSS の

optimistic プロトコルの利用が有効であると考えられる．

7. まとめと今後の課題

本稿では，大規模分散システムの評価手法として，並列離散イベントシミュレーションを用いた手法について提案を行った．並列離散シミュレーションフレームワーク CODES 及び ROSS を用い，分散ファイルシステムのための分散メタデータ管理システム PPMDS のシミュレータ PPMDSsim を実装した．

評価では，実環境と同じノード構成を想定しシミュレータを実行した．サーバ台数とクライアント数を増加させるに従って性能向上が見られ，サーバ台数が 4 台未満の場合には一定の性能限界を示すというシミュレーション結果が得られた．実環境で十分な評価が行われていなかったサーバ台数が 4 台以上の場合でも，クライアント数を増やすことで一定の性能限界を示すポイントがあることが確認できた．また，性能に関わるパラメータを変更し実行することで，性能のボトルネックについて調査を行い，ネットワークが要因であると考察した．並列シミュレーションの検証では，optimistic プロトコルを使用することで，conservative プロトコルよりも高速なシミュレーションが可能であることを確認した．結果より，1 台の計算ノードで実行できない規模のシミュレーションを並列化する際には，ROSS の optimistic プロトコルの利用が有効であると考えられる．

今後の課題として，以下の 3 点が挙げられる．1 点目は，分散ファイルシステムでの利用を想定し，より多くのワークロードについて評価を行うことである．ファイルシステムにおけるメタデータの操作は，ファイルやディレクトリの作成以外にも，参照や削除が必要となる．また，今回シミュレーション対象にした PPMDS では，メタデータの分散先サーバのリストを Key-Value store に格納している．リストの格納や削除の操作は，操作対象のメタデータに関与する全てのサーバに対して行われるため，並列化した際のオーバヘッドの要因となりうるものである．今後，これらの操作に関するシミュレーションも実行できるようシミュレータを改良し，より多くの観点から分散システムの性能を評価可能にすることが望ましいと考えられる．

2 点目は，並列シミュレーションの高速化及び安定化である．ROSS の optimistic プロトコルの利用でシミュレーションの実行時間の短縮が見られたが，64 プロセスによる並列化で 1 台の計算ノードの実行時間と比較し最大 2.3 倍程度の高速化に留まった．また，図 9 における 16 プロセス使用時の実行時間に見られるように，プロセス数を増やすことによって実行時間が増加するケースも確認された．今後，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査する必要がある．

3 点目は，シミュレーションによる性能評価に取り組む他の類似手法と比較するために，PPMDS 以外の大規模分散システムに関するシミュレーションスタディを行うことである．今後，広く一般的に利用される P2P プロトコルなどをシミュレーション対象とし，同一のワークロードにおけるシミュレーション実行時間の比較や，スケーラビリティの評価を行う必要があると考えられる．

謝辞 本研究の一部は，文部科学省「特定先端大型研究施設運営費等補助金（次世代超高速電子計算機システムの開発・整備等 ）」の一部として実施された文部科学省と米国エネルギー省の合意による「現在および将来の HPC システムのシステムソフトウエアに関する共同研究」，および 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」による．

参考文献

[1] 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, 研究報告ハイパフォーマンスコンピューティング（HPC）, 情報処理学会, 2012-HPC-135, No. 28, 9 pages, 2012.

[2] Baumgart, I.; Heep, B.; Krause, S., OverSim: A Flexible Overlay Network Simulation Framework, Proceeding of IEEE Global Internet Symposium, pp.79-84, 2007.

[3] K. Shudo, Y. Tanaka, S. Sekiguchi, Overlay Weaver: An Overlay Construction Toolkit, Computer Communications, Vol. 31, No. 2, pp.402-412, 2008.

[4] Andras Varga: The OMNeT++ Discrete Event Simulation System, Proceedings of the European Simulation Multiconference (ESM'2001), 7 pages, 2001.

[5] 杉野好宏, 華井雅俊, 首藤一幸, MapReduce による分散システムのシミュレーション, 研究報告マルチメディア通信と分散処理（DPS）, 情報処理学会, 2012-DPS-152, No. 33, 7 pages, 2012.

[6] 杉野好宏, 華井雅俊, 首藤一幸, MapReduce による大規模分散システムのシミュレーション, 信学技報, 電子情報通信学会, NS2013-109, Vol. 113, No. 244, pp. 99-104, 2013.

[7] David R Jefferson. Virtual time. ACM Transactions on Programming Languages and Systems (TOPLAS), 7(3):404425, 1985.

[8] Ning Liu, Xi Yan, Xian-He Sun, Jonathan Jenkins, Robert Ross. YARNSim: Hadoop YARN Simulation System. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 10 pages, 2015.

[9] 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, 研究報告システムソフトウェアとオペレーティング・システム（OS）, 情報処理学会, 2011-OS-118, No. 16, 7 pages, 2011.

[10] Christopher D Carothers, David Bauer, and Shawn Pearce., ROSS: A high-performance, low-memory, modular Time Warp system. Journal of Parallel and Distributed Computing, 62(11):16481669, 2002.

[11] carothersc/ROSS, https://github.com/carothersc/ROSS.

[12] CODES: Enabling Co-Design of Multilayer Exascale Storage Architectures, http://www.mcs.anl.gov/project/codes-enabling-co-design-multilayer-exascale-storage-architectures/.

[13] M. Hirabayashi., Kyoto Cabinet, http://fallabs.com/kyotocabinet/.