Vol.2017-HPC-160 No.26
2017/7/27

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ベンチマーク

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. 本研究では, 並列離散イベントシミュレータ CODES/ROSS を用いて分散メタデータサーバのベンチマークプログラムを実装し仮想的にベンチマークを行うことを目的とする.　評価対象は分散メタデータサーバ PPMDS とし、mdtest を用いたベンチマークをシミュレーションし複数ディレクトリへのファイル作成の評価を行った.

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 [3] 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, シミュレーションによる評価を行いシミュレーションスタディが有効であるという結果を得た. 例えば, PPMDS シミュレータ [1][2] は単一ディレクトリへのファイル操作及びディレクトリ操作のシミュレーションプログラムを実装し, 性能評価を行うことによって性能のボトルネックの要因がネットワーク開始遅延にあるという結果を得た. 本研究では, CODES/ROSS を用いて, PPMDS の mdtest[9] を用いたベンチマークのシミュレーションを実装しツリー状のディレクトリへのファイル操作について性能評価を行うことを目的とする。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, mdtest, 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, PPMDS シミュレータを用いて, 単一ディレクトリへのファイル操作, ディレクトリ操作を行った結果を示す. 第 4 章では, 本研究において実装したシミュレータについて, 設計及び実装について述べる. 第 5 章では, 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 6 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.

2. 背景

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

[1] 筑波大学システム情報工学科
Graduate School of Systems and Information Engineering, 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

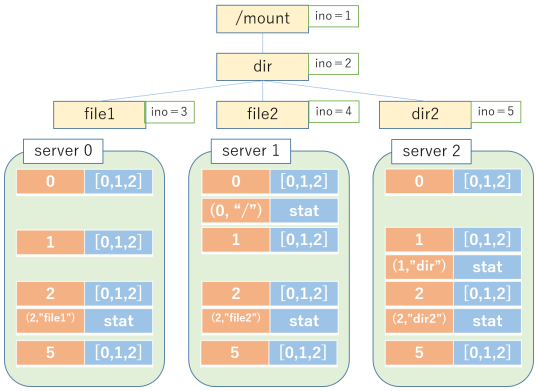

図 1　PPMDS での inode 格納方式 文献 [2] より引用

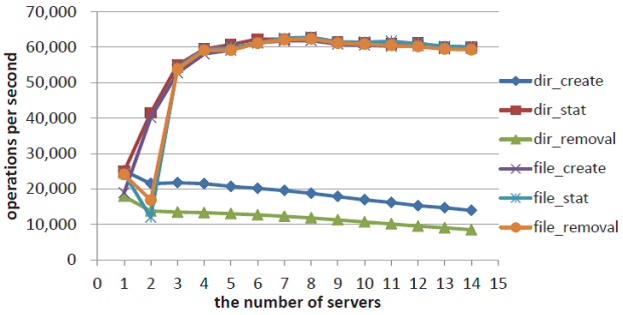

図 2　88 クライアントを用いた並列ファイルシステム操作
文献 [3] より引用

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[4] 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 [5] 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 [3] 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.

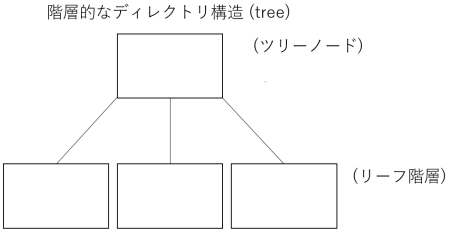

図 3　階層的なディレクトリ構造

2.2 mdtest HPC benchmark

mdtest[9] は MPI を用いて複数タスクによるメタデータベンチマークテストを行う. ファイルとディレクトリの作成, 参照, 削除の性能を測定することができる. mdtest ではツリー状のディレクトリ構造（tree）を作成し作成した tree に対してファイル作成及びディレクトリ作成を行うことによって性能を評価する. mdtest で作成される tree は balanced tree となる. ベンチマークを行うときのパラメータを用いて作成する tree の深さ及び幅、ファイル、ディレクトリ作成数などを指定することができる。また、複数タスクがファイル作成先となるディレクトリの tree 共有するかまたは、独立した tree に作成するかどうかなども指定することができる。 深さ 1, 幅 3 と指定したときのディレクトリの tree を図 3 に示す.

2.3 ROSS: Rensselaer's Optimistic Simulation System

Rensselaer's Optimistic Simulation System [6] 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.

2.4 CODES: Enabling Co-Design of Exascale Strage Architectures

Co-Design of Exacscale Storage System Project [7] 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

図 4　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 (文献 [1] より引用)

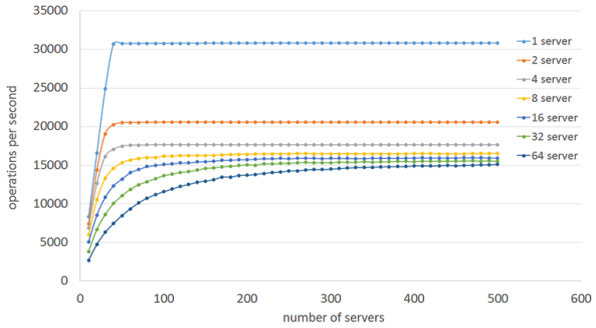

図 5　単一ディレクトリへの並列ディレクトリ作成シミュレーション

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.

3. PPMDS での単一ディレクトリへの並行ファイル, ディレクトリ操作のシミュレーション

PPMDS シミュレータ [1][2] は, CODES/ROSS を使用し, PPMDS での単一ディレクトリへのファイル操作, ディレクトリ操作についてシミュレーションプログラムを実装し, スケーラビリティを評価している.　図 **4**, 図 **5** にシミュレーション結果を示す.

4. mdtest を用いた PPMDS のベンチマークシミュレータ

本章では, mdtest[9] を用いた PPMDS のベンチマークシミュレータについて説明する. 図 **6** に PPMDS シミュレータで再現した評価環境を, 図 **7** に PPMDS ベンチマークシミュレータで再現した評価環境を示す. 今回、mdtest を用いた PPMDS のベンチマークをシミュレータで再現す

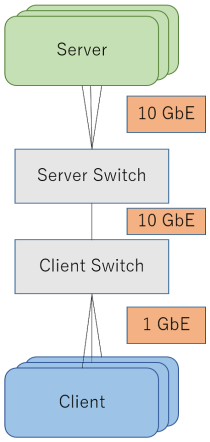

図 6　PPMDS の評価環境 [2] より引用

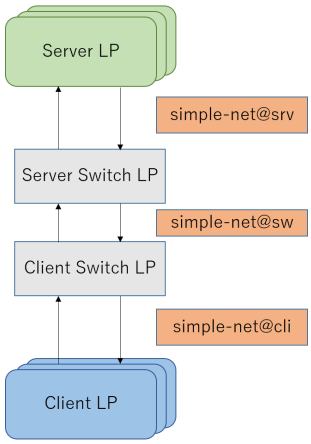

図 7　PPMDS シミュレータで再現した評価環境 [2] より引用

るために、CODES の設定ファイルを用いてファイル作成を行うディレクトリの tree の深さと幅及び、各ディレクトリにいくつのファイルを作成するかを設定できるようにした. ファイル作成シミュレーションを行う前に設定ファイルで指定した値からクライアントごとにツリー状のディレクトリを作成し, クライアントは作成したディレクトリのツリー内のディレクトリに対してファイル作成を行っていく.

4.1 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. Client LP はファイル作成シミュレーションを始める前に、設定ファイルから指定された値にもとづいてディレクトリの tree が作成されるように Server LP に対して、メッセージを送る。　ディレクトリの tree が作成された後、Client LP は tree 内の各ディレクトリに対して、ファイル作成を行っていく。

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 **8** に示す.

KICKOFF

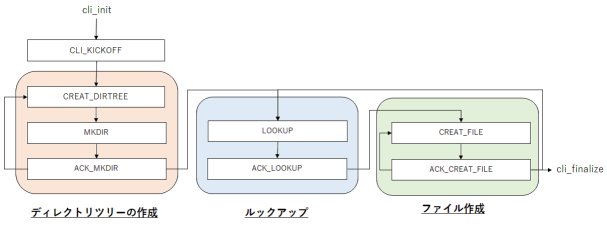

図 8　Client LP のイベントのワークフロー

シミュレーションを開始するためのイベント.

CREAT_DIRTREE 設定ファイルで指定された深さ, 幅のツリー状のディレクトリ構造を作成する.

MKDIR, ACK_MKDIR
ディレクトリ作成のためのイベント、パス名と親ディレクトリの inode 番号からからハッシュ関数を用いて格納先サーバの LP_ID を計算し Server LP にディレクトリ作成のメッセージを送信する. ディレクトリ作成完了のメッセージを ACK_MKDIR イベントで受け取る。　MKDIR イベントを用いてディレクトリの tree を作成し、作成したディレクトリの tree 内のディレクトリそれぞれにファイル作成を行う。

LOOKUP, ACK_LOOKUP
ファイルを作成するディレクトリの inode エントリを Server LP に要求する. ルックアップしたディレクトリの inode は Client LP にキャッシュされる。ルックアップ完了のメッセージを ACK_LOOKUP イベントで受け取り、ルックアップしたディレクトリ inode を用いてファイル作成イベントを実行する。

CREAT_FILE,ACK_CREAT_FILE
ファイル作成を要求するイベントを Server LP に送信する。 ファイル作成完了のメッセージは ACK_CREAT_FILE イベントで受け取り、設定ファイルで指定したファイル数を作成するまでファイル作成イベントを実行する。

4.2　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.

4.3　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.
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.

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

表 2　並列離散イベントシミュレーションに使用した計算ノード

OS	CentOS release 6.9 (Final)
CPU	Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz
Memory	64GB

MKDIR
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 に格納し, 分散先サーバに分散先サーバリストを格納するイベントを送信する。

STORE_SIDLIST
分散先サーバリスト作成要求のメッセージで指定された inode 番号を key と分散先サーバリストを value とし 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.

LOOKUP_INO
Client LP から送信されたルックアップ要求から、Key-Value store 上の inode エントリをルックアップし inode 番号を Client LP にルックアップ完了のメッセージを送信する。

STORE_FILE_INO
Client LP にファイル作成完了のメッセージを送信する.

各 Server LP が KyotoCabinet[8] 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 [5] の手法を用いたトランザクション処理を行う.

4.4　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.

5.　シミュレーション結果

本章では, シミュレータによる PPMDS のベンチマークを実行した結果を示す. 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.　シミュレー

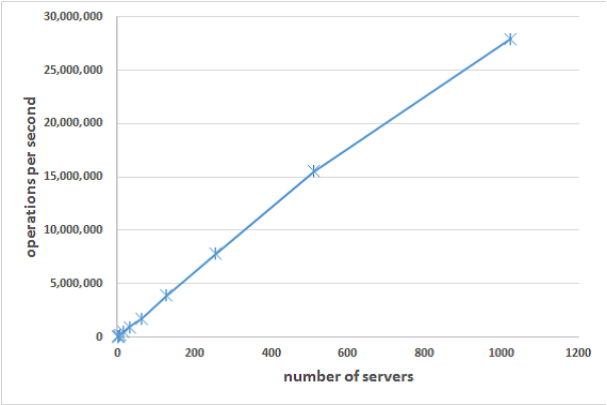

図 9 サーバ数ごとの秒間あたりのファイル作成数の最大性能 (深さ=2, 幅=2) 総ディレクトリ数=7

表 3 サーバ数ごとの秒間あたりのファイル作成数の最大性能 (深さ=2, 幅=2) ディレクトリ数=7

サーバ数	operations/second [ops]
1	30,266 ops
2	60,481 ops
4	120,279 ops
8	236,083 ops
16	465,536 ops
32	915,889 ops
64	1,736,022 ops
128	3,894,045 ops
256	7,800,661 ops
512	15,553,593 ops
1024	27,927,237 ops

ションを並列に実行する際には**表 2** に示す計算機を 5 ノード使用した.

5.1 mdtest でのベンチマークのシミュレーション結果

複数ディレクトリへのファイル作成について PPMDS シミュレータを用いてベンチマークを実行した. 総ディレクトリ数を 7 としディレクトリのツリーの幅と深さを変更しシミュレーションを実行した. ディレクトリのツリーの幅と深さを深さ 2 幅 2, 深さ 1 幅 6 それぞれについてサーバ数ごとの秒間あたりのファイル作成数の最大性能をサーバ数を 1〜1024 サーバの間で変化させ評価した結果を以下の**図 9** **図 10**, **表 3** **表 4** に示す. シミュレーション結果から, PPMDS はディレクトリの構造に関係なくサーバ数に応じて性能が向上すると考えられる. **表 5** に図 10 のシミュレーションを実行したときに設定した 1〜1024 サーバにおけるクライアント数, 1〜512 サーバにおけるシミュレーションの実行時間を示す.

5.2 ディレクトリ構造のファイル作成性能への影響

ディレクトリの木構造の深さ及び幅を変化させファイル作成シミュレーションを実行し, ディレクトリの構造が

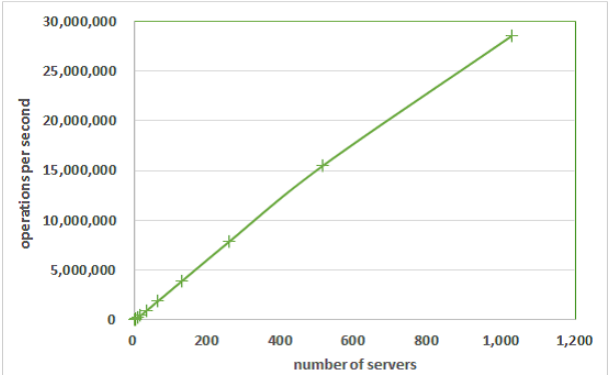

図 10 サーバ数ごとの秒間あたりのファイル作成数の最大性能 (深さ=1, 幅=6) ディレクトリ数=7

表 4 サーバ数ごとの秒間あたりのファイル作成数の最大性能 (深さ=1, 幅=6) ディレクトリ数=7

サーバ数	operations/second [ops]
1	30,266 ops
2	60,614 ops
4	121,303 ops
8	242,878 ops
16	485,890 ops
32	975,807 ops
64	1,959,319 ops
128	3,936,892 ops
256	7,876,963 ops
512	15,539,992 ops
1024	28,523,679 ops

表 5 図 10 の評価で設定したクライアント数, サーバ数及びシミュレーションの実行時間

サーバ数	クライアント数	シミュレーション時間
1	30	0.7621 seconds
2	100	2.7269 seconds
4	200	5.7010 seconds
8	500	11.9868 seconds
16	1000	25.2236 seconds
32	2000	57.0533 seconds
64	4000	119.6023 seconds
128	6000	193.6637 seconds
256	12000	432.0195 seconds
512	24000	1023.7511 seconds
1024	48000	

PPMDS でのファイル作成性能への影響をシミュレーションによって評価した. シミュレーション結果をずに示す. サーバ数を 16, クライアント数を 88 に固定してシミュレーションを実行した. ディレクトリの構造は (深さ:幅) をそれぞれ (0:1),(1:1),(2:2),(3:3),(4:4) としてそれぞれのディレクトリのツリーに対してファイル作成のシミュレーションを行った. シミュレーション結果を**図 11** に示す. シミュレーション結果から, PPMDS ではディレクトリの構造がファイル作成性能へほとんど影響を与えないと考えられる.

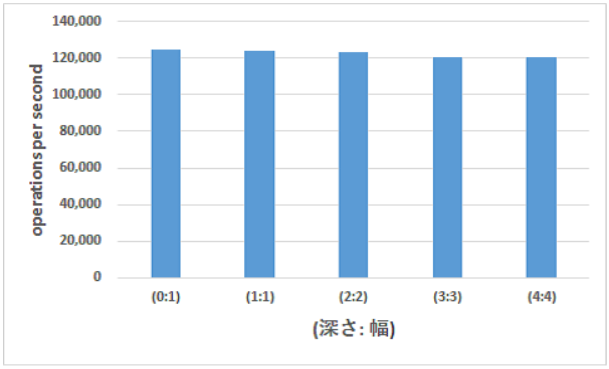

図 11　秒間あたりのファイル作成数

6. 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 でのファイル操作のベンチマークについて, 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, 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. mdtest でベンチマークを場合を再現し, 複数のディレクトリに対してのファイル作成のシミュレーションを行うことによって PPMDS がディレクトリの構造や総ディレクトリ数に関係なくサーバ数を増やしたときに性能が向上するという結果をシミュレーションによって得ることができた.

今後の課題は, 現在 PPMDS のシミュレータはメタデータ操作にしか対応していないためストレージサーバとしての評価を行うことができない. ストレージサーバとしての評価を行うためにファイル作成の要求がサーバに送信されたときストレージサーバにファイルのデータを作成するようなシミュレーションを実装することによってストレージサーバとメタデータサーバを組み合わせたときの性能評価もシミュレーションによって行うことが必要であると考えられる.

謝辞　本研究の一部は，JST CREST「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」，JST CREST「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」および JSPS 科研費 JP17H01748 の助成を受けたものです.

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 PPMDS の評価. 情報処理学会研究報告, Vol. 2015-HPC-152, No.7, pp.1-9, 2015.
[2] 小林淳司, 建部修見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. 情報処理学会研究報告, Vol. 2017-HPC-158, No.10, pp.1-9, 2015.
[3] 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.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] FAL Labs. Kyotocabinet http://fallabs.com/kyotocabinet/
[9] mdtest https://github.com/MDTEST-LANL/mdtest
[10] Shane Snyder, Philip Carns, Robert Latham, Misbah Mubarak, Robert Ross, Christopher Carothers, Babak Behzad, Huong Vu Thanh Luu, Surendra Byna, and Prabhat. Techniques for Modeling Large-scale HPC I/O Workloads. International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS15), 2015.