

# 高遅延環境における分散ファイルシステム Hadoop の 遠隔データアクセス特性の評価

百瀬明日香<sup>†</sup> 小口 正人<sup>†</sup>

<sup>†</sup> お茶の水女子大学 〒112-8610 東京都文京区大塚 2-1-1  
E-mail: †momo@ogl.is.ocha.ac.jp, ††oguchi@computer.org

あらまし 情報爆発の現代において大きな障害となる企業や個人のデータ処理の負荷とストレージコストの増加の問題に対して、汎用のハードウェアを用いて高度な集約処理を行う分散ファイルシステムに注目が集まっている。なかでもファイルの一部が遠隔地に存在するシステムなどに見られる近接/遠隔ネットワークを併用する広域分散ファイルシステムの運用は、インターネット回線の高速化などネットワーク網の発展に伴い近年その需要を拡大している。本研究ではこうした分散ファイルシステムの実装として Hadoop を取り上げ、高遅延環境において遠隔データアクセス頻度を制御した際の性能評価やパケット解析を行うことによって、広域分散ファイルシステムにおける遠隔データアクセスの特性に関する検討を行う。

キーワード 情報爆発, 分散ファイルシステム, Hadoop, 遠隔データアクセス制御

## Characterization of Remote Data Access for Hadoop Distributed File System over a long-latency environment

Asuka MOMOSE<sup>†</sup> and Masato OGUCHI<sup>†</sup>

<sup>†</sup> Ochanomizu University Otsuka 2-1-1, Bunkyo-ku, Tokyo, 112-8610 Japan  
E-mail: †momo@ogl.is.ocha.ac.jp, ††oguchi@computer.org

**Abstract** Under the Info-plosion age, Distributed File System (DFS), on which intensive transaction is executed with commodity machines, is receiving attention to meet the increase of data management's load and storage costs for business and individuals. Because of the development of Internet connection, there is more demand for DFS implementation over a wide area environment, such like a system using local network and remote network together to hold a part of data at remote server. In this research, we have adopted Hadoop Distributed File System(HDFS), and controled frequency of remote access. And we search characterization of remote data access for HDFS over a long-latency environment, evaluating performance of DFS in a long-latency environment and analyzing packets flowing between nodes.

**Key words** info-plosion, Distributed File System, Hadoop, Remote Data Access

### 1. はじめに

情報爆発の現代において企業や個人のデータ処理の負荷とストレージコストの増加が大きな問題となっている。このような問題に対してコモディティなハードウェアを用いて高度な集約処理を行う分散ファイルシステムに注目が集まるとともに、インターネット回線の高速化などネットワーク網の発展に伴い、広域環境におけるファイル分散が現実化しているという素地が存在する。こうした現状から本研究では広域分散環境において遠隔地にファイルのバックアップを保持し、近接および遠隔ネットワークを併用することで自然災害やテロリズムなどの大

規模なデータ損失にも対応できるような信頼性の高いシステムの運用に着目した。一方でこうしたシステムの運用に際しては、遠隔地へのアクセスによって発生するネットワークの遅延のため、データ処理効率の低下が予想される。本研究ではこのような遅延の影響に関して、遠隔データアクセス頻度に焦点を当てその振舞を調べた。

### 2. 分散ファイルシステム

本研究では近年のストレージ負荷問題の解決手法として分散ファイルシステムを採用している。これはハードウェアの制約がなく高いスケーラビリティを持つため、システムの規模によ

らず様々な場面での運用が見込まれる点を評価したものである。こうした分散ファイルシステムの中でも、特に Web 時代対応のデータインフラサービスとして、多対多アクセスに特化したシステムに着目した。Web サービスにおいては単一ユーザあたりのデータ通信量・トラフィック確保よりも、莫大なアクセスに対してトータルスループットを維持することが優先されるという特徴がある。

### 2.1 Google File System

Google File System(以下 GFS)は現在 Google 社で実用的に用いられている分散ファイルシステムであり、基盤となる GFS, 分散処理アルゴリズムである Map/Reduce, データベースである Bigtable から構成される [1]。大量のデータを全世界的に並行処理する、大規模広域分散ファイルシステムの実用例として本研究の研究対象として取り上げた。Web サービス処理に特化した Map/Reduce アーキテクチャを採用している。

### 2.2 Hadoop Distributed File System

本研究では分散ファイルシステムの実装として GFS から着想を得て開発されたオープンソースソフトウェア Hadoop Distributed File System (以下 HDFS) を使用した [2]。Hadoop は Apache Software Foundation で開発されている分散コンピューティング関連のプロジェクト群を指し、コンポーネントの Hadoop Common をはじめとし、GFS に相当する HDFS, Map/Reduce に相当する Hadoop MapReduce, Bigtable に相当する hBase など複数のサブプロジェクトから構成される (図 1)。本研究ではこのうちの特に HDFS と Hadoop MapReduce の実装を実験環境として使用した。

Pig	Chukwa	Hive	HBase
MapReduce	HDFS		ZooKeeper
Hadoop Common		Avro	

図 1 Hadoop のサブプロジェクト (2010 年 5 月現在)

## 3. 関連研究

大規模分散ファイルシステムに関しては GFS, HDFS 以外にも多数の既存研究・実装が存在する。本項では特に大規模システムとして応用が可能であるいくつかの分散ファイルシステムの関連研究の例を挙げる。

### 3.1 Lustre File System

Cluster File Systems, Inc. が開発、配布しているクラスタファイルシステム。Client, Meta DataServer(MDS), Object Storage Targets(OST) の三種のノードから構成され、複数クライアントによる同時実行制御に特化している。HA 構成のメタデータサーバと、FS レベルでのストライプ処理が可能。特徴的な動作として分散ロック機構が挙げられる。ファイルアクセス時 Client, MDS 間の通信にロック機構を持たせ、ロック取得時に以降の動作情報を付加送信することでメッセージ数を削減し性能向上を図るというものである。高品質・高信頼なデータ通信を確保する一方、レプリカ機能などはストレージに依存

した実装となっている。

### 3.2 Gfarm

産業技術研究所他の開発する Gfarm (Grid Datafarm) は、広帯域ネットワークで接続される大規模クラスタを想定したファイルシステムであり、e-Science (大規模シミュレーション, データ解析) 分野の処理に特化している。データローカリティや専用 API による並列データ処理により高速処理を可能とするが、レプリケーションは手動管理する必要があるため、並列アプリケーション記述のための技術が求められる。

## 4. 実験環境

クラスタ自動構築・管理ツール Rocks [3] を用いてインストールしたマシン 7 台のうち、ワーカ 6 台に Hadoop を導入した。マシンスペックは表に示す通りである (表 1)。各ノード上に Hadoop-0.18.3 をインストールし、1 台を Hadoop Namenode 兼 Datanode, 残る 5 台を Hadoop Datanode として使用した。ここで Namenode とは GFS におけるマスタ, Datanode とは GFS におけるチャンクサーバと同様の役割を持つノードである。また分散されるファイルの最小単位であるブロックサイズは 2MB とした。測定のためのベンチマークには、Hadoop に付属の TestDFSIO プログラムを使用した。

## 5. 高遅延環境における測定

### 5.1 実験概要

まず高遅延環境下における基本的な性能の測定を行った。Namenode1 台と Datanode3 台のうち Datanode1 台を人工遅延装置 dummynet を介して接続することで、4 ノードのうち 1 台が遠方に存在すると仮定した環境での測定を行う (図 2)。分散されるファイルのレプリカ数は 1 とした。ただしレプリカがどこに配置されるかは Hadoop によって決められ、これを明示的に指定するインタフェースは用意されていない。後述するラック設定を用いる場合、一般的にはノード間である程度均等になるよう、レプリカが分散される。

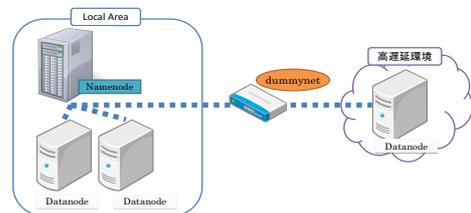


図 2 システム構成

### 5.2 ノード間距離を考慮しないファイル分散

10MB のファイルをシーケンシャルライトで 100 個作成、作成したファイルをシーケンシャルリードしファイルシステム I/O 性能を計測した。ここで、パラメータ“ Throughput ”は、

表 1 マシンスペック

	OS	CPU	Main Memory
Master node	Linux 2.6.9-55.0.2.	Intel(R) Xeon(R)	
	Elsmpt(CentOS 4)	@3.6GHz	4.0GB
Slave node	Linux 2.6.9-55.0.2.	Quad-Core Intel(R)	
	Elsmpt(CentOS 4)	Xeon(R) @1.60GHz	2.0GB

ファイルシステム内部における単位時間辺りのデータ処理量を表している。このとき HDFS のラック設定は適用せず（6 章にて後述）、システムにはノード間距離を考慮しないファイル分散を行わせている。

ローカルエリアと高遅延マシン間の往復遅延時間（以下 RTT）を 0msec から 20msec まで変化させ測定を行った。レプリカ数 1 の場合を比較するとライトスループットは遅延が増加してもほぼ一定であるのに対し、リードでは大きく低下した（図 3、図 4）。ファイルの書込はバッファを介して行われるためライトでは遅延分の差異が出づら形となったのに対して、リードではレプリカ数 1 であるため一定の割合で高遅延ノードへのアクセスが行われ、その結果スループットが低下したものと考えられる。

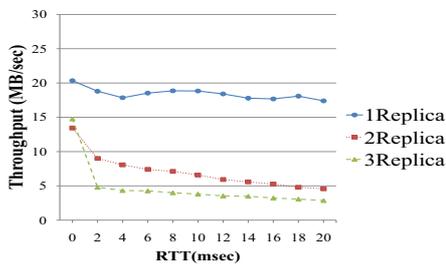


図 3 Write スループット

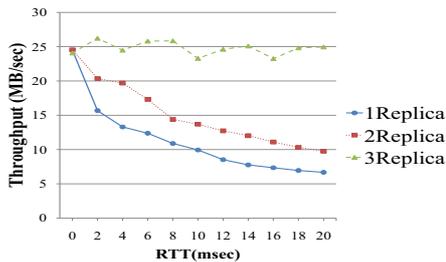


図 4 Read スループット

またこのとき、Namenode と各 Datanode の通信頻度を比較した。解析にはパケット解析ソフトウェアである Wireshark [4] を使用した。性能測定と同様にレプリカ数 1 の状態でシーケンシャルライト、シーケンシャルリードを交互に 5 回ずつ実行したとき、各 Datanode のパケット通信量を調べた（図 5）。

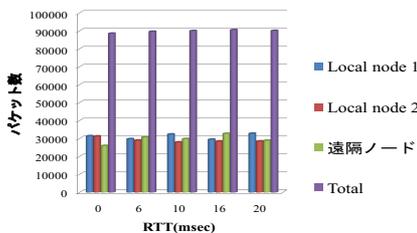


図 5 各ノードとの通信量

3 台の Datanode との合計通信量、高遅延の Datanode の通信量どちらを見ても RTT が変化しても増減が見られず、HDFS で遅延によるパケットロスや再転送が発生していないことが分かる。これはジョブ単位のパケット数を検討した場合も同

様で、各 Datanode との通信量には遅延の有無によらず 10～30% くらいの誤差が見られ、応答の遅いノードに対しても平等にデータアクセスが行われる（図 6、図 7 図は高遅延ノードの RTT=20msec のとき）。Datanode 3 台のうち 1 台特に通信量の増えるノードが発生していることが分かるが、これは Namenode が単一障害点になることを防ぐため Datanode のうち 1 台がプライマリノードとして Namenode の仕事を一部肩代わりする実装による。ラック設定を行わない限りシステムによるアクセス制御は行われないうことが確認された。

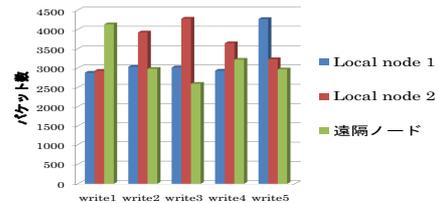


図 6 write 実行時のパケット数

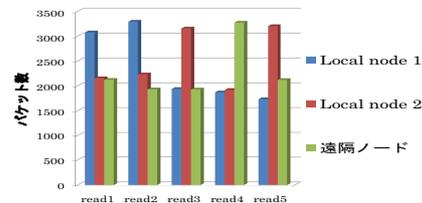


図 7 read 実行時のパケット数

一方、レプリカ数を増加させた場合、ライトのスループットが低下し、リードのスループットは上昇することが確認された。ライトでは書き込むべきデータ量が増加するため性能が劣化し、反対にリードでは高遅延のノードにアクセスする確率が減るためスループットが向上したものと見られる。各ノードへのアクセス頻度はレプリカ数 1 の場合と同様、遅延によらず平等である。

## 6. ラックを考慮したファイル分散

### Hadoop のラックポリシー

Hadoop にはファイル分散制御を明示的に指示できるインタフェースがないため、遠隔データアクセス制御のためにラックポリシーという概念を利用する。クラスタが複数のラックに跨る場合、経由するスイッチやハブが増加する分ネットワーク帯域幅が減少するという前提のもと、ラック間の転送よりもラック内の転送を優先するというのが Hadoop のラックポリシーである。

#### 6.1 ラック設定適用時の基本性能

遅延を含まないクラスタを 2 つのラックに分割した場合のファイルシステム I/O 性能を測定した結果を示す（図 8～図 11）。図 10 と図 11 はそれぞれ図 8 と図 9 の拡大図である。ノード数を 3 台から 6 台までスケールさせ、各々の台数において 2 つのラックを構成するノード数の割合を表 2 のように変化させた。レプリカ数はすべて 2 である。ここでローカルエリアにおける基本性能として、同一レプリカ数でノード数をスケールさ

せた場合 1 台あたりのスループットはノード数によらず一定であること（すなわちシステム全体におけるスループットの合計はノード数に比例して線形増加すること）が分かっている。グラフの横軸は偏度で、これは前ノードのうち何%が片方のラックに含まれているかを表す。各偏度におけるラック毎のノード数は表 2 に示した通りである。

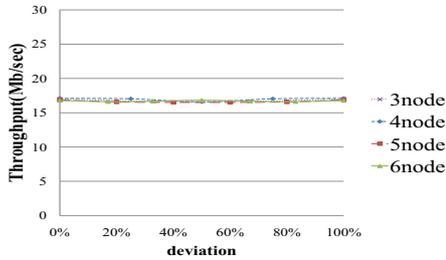


図 8 Write スループット

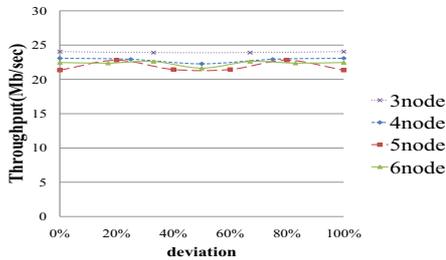


図 9 Read スループット

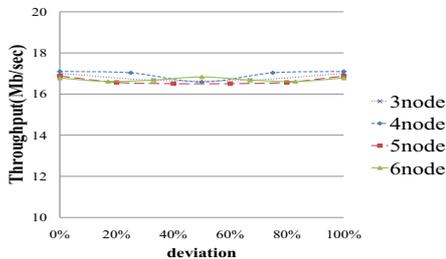


図 10 Write スループット拡大

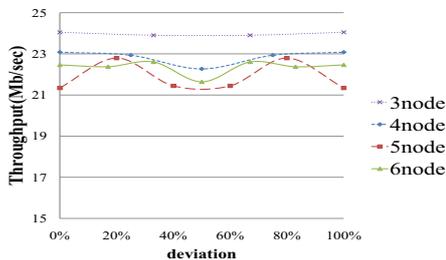


図 11 Read スループット拡大

高遅延を含まない環境において、ライトではラックを構成するノードの偏度によるスループットの変化はほとんど見られないことが分かる（図 8、図 10）。一方リードの場合、例えばノード数 5 では 2 つのラックを 1 : 4 ノードで構成した場合のリードスループットが高く、ノード数 6 で 2 つのラックを 3 : 3 で構成した場合にはスループットが低下していることが分かる（図 9、図 11）。これはリードではブロックが単一ノードへ集中す

るほどアクセス効率が良く、性能向上することによって考えられる。これらの差異は誤差に含まれる程度の値であるが、高遅延ノードを含まないクラスタ内であってもラック設定がファイルシステム I/O に多少の影響を及ぼすことが分かった。

またこのようなラック設定適用時のパケット解析結果を以下に示す（図 12、図 13）。このとき Datanode は 3 台である。

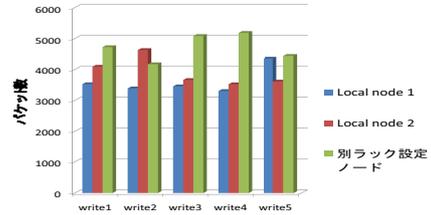


図 12 write 実行時のパケット数

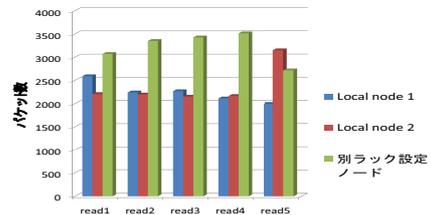


図 13 read 実行時のパケット数

非適用時（図 6、図 7）と比べ、別ラックに設定したノードへのアクセス頻度が上昇している。ノードがプライマリに選出されない場合でも、プライマリノードと同等の頻度でアクセスが行われることが分かる。

## 6.2 高遅延環境における単純なラック設定適用時の性能

続いて高遅延ノードを含む実装環境においてラックポリシーを用いた性能測定を行った。ローカルクラスタに存在する 2 台の Datanode には/default/rack0 というラック ID、遠隔ノードの Datanode には/distant/node/rack1 というラック ID を付与した（図 14）。これは物理的構成を考慮した場合、最も自然に想像されるラック設定である。またレプリカ配置の結果が反映されやすいようレプリカ数は 2 に設定した。

このとき、ファイルシステムに一定量のデータを書込んだ際のブロック数を GUI から確認すると、遠隔ノードへの書込が増加していることが分かる（表 3）。これは Hadoop のレプリカ配置ポリシーとして、複数ラックが存在するとき異なるラックにレプリカを分散して配置することで耐故障性を維持しようとする根源的なレプリカ配置ポリシーがラックポリシーよりも優先されたことにより、ラックの異なる遠隔ノードに対してレ

表 2 ラック毎のノード数

	0%	33%	67%	100%			
3node	0	1	2	3			
RackA のノード数	0	1	2	3			
RackB のノード数	3	2	1	0			
4node	0%	25%	50%	75%	100%		
RackA のノード数	0	1	2	3	4		
RackB のノード数	4	3	2	1	0		
5node	0%	20%	40%	60%	80%	100%	
RackA のノード数	0	1	2	3	4	5	
RackB のノード数	5	4	3	2	1	0	
6node	0%	17%	33%	50%	67%	83%	100%
RackA のノード数	0	1	2	3	4	5	6
RackB のノード数	6	5	4	3	2	1	0

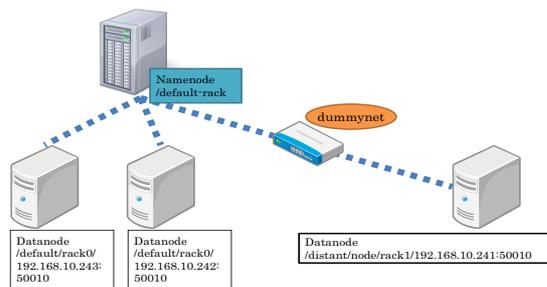


図 14 ラック ID の付与

ブリカの複製が高い確率で配置されたことによる．これによって単純なラック設定においては単一の遠隔ノードへのアクセスはむしろ増加し、性能低下を招くことが予想される．

このような設定における性能測定では、ライトスループットの性能が低下することが確認されている．

表 3 データ書込後のブロック数

	Rack ID	Block	比率%
Local Datanode1	/default/rack0	525	25.3
Local Datanode2	/default/rack0	515	24.9
Distant Datanode	/distant/node/rack1	1030	49.8

### 6.3 レプリカ配置ポリシーを考慮したラック設定適用時の性能

前節の結果を受けて、レプリカ配置ポリシーを考慮したファイル分散としてここでは以下のようなラック設定を適用した(表 4)．近傍ノードを別ラックとして指定することでローカルノードへのアクセスを増やし、間接的に遠隔ノードへのアクセス機会を減らしている．

表 4 ラック設定を変更した場合の書込ブロック数

	Rack ID	Block	比率%
Local Datanode1	/distant/node/rack1	606	50.0
Local Datanode2	/default/rack0	326	26.9
Distant Datanode	/default/rack0	280	23.1

このときの性能測定結果 (optimized rack) を、ラック未設定 (default) の場合、ラックを単純に設定した場合 (simple rack) と比較する (図 15, 図 16)．ライトではレプリカ配置ポリシーを考慮したラック設定を行うと、何も設定を行わない場合より平均的にスループットが上昇している．リードでは反対にラック設定を行わない場合より、レプリカ配置ポリシーを考慮した設定を行った場合の方がスループットが低下する形となっている．ライトではレプリカの分散先に遅延ノードが含まれる確率が減少したことによって性能が向上していることが分かる．リードの性能は偶発的に遠隔ノードのデータを読み出すことが多かった場合に性能が低下したものと考えられる．

### 6.4 考察

以上の実験結果より HDFS においてラック設定を適用し、ラックのノード構成を変化させることでファイルシステム I/O に影響があることが分かった．特にライト性能においては高遅

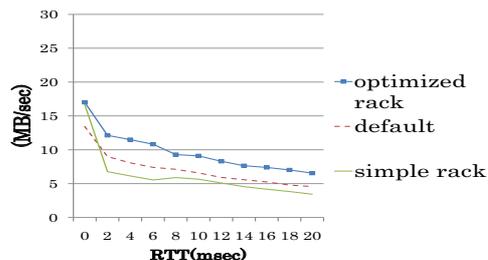


図 15 Write スループット

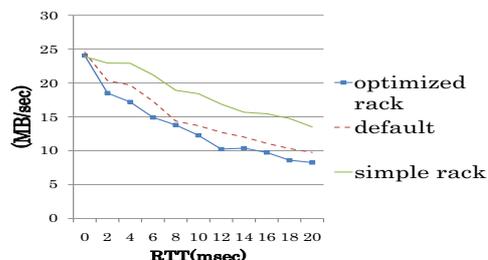


図 16 Read スループット

延ノードへのアクセスを抑えることによって若干の性能向上が確認され、レプリカ配置の制御によって HDFS の性能向上が見込めることが分かった．

## 7. Map/Reduce 性能の測定

### 7.1 転置インデックスプログラム

ここまでファイルシステム I/O の性能評価を行ったが、Hadoop 特有の議論としては広域分散環境が Map/Reduce 性能に及ぼす影響を検討することも重要な課題である．本節ではその一例として、Map/Reduce 処理をメインに行うベンチマークである転置インデックスプログラムを自作し、高遅延環境におけるクライアントアプリケーションの実行時間を測定した．まず URL とテキスト 50,000 行からなる入力データを用意し、Map 処理として文書を N-gram モデルを用いて切り出し、URL をキー、含まれるテキストをバリューとして抽出する．中間処理で (キー、バリュー) データをバリューでソートし、Reduce 処理で重複したキーの削除を行う．なお、N-gram により切り出す文字列の長さは  $n=5$  の場合を採用した．

### 7.2 高遅延環境における基本性能

転置インデックスプログラムを使用し、5 章と同様 3 台の Datanode のうち 1 つが高遅延環境に存在する場合の基本的な処理性能を計測した．クライアントが HDFS にジョブを投げ、その結果が再びクライアントへ返されるまでの所要時間を 12 回計測し、最大最小値を除いた 10 回の試行における平均を採用している．横軸を RTT として実行時間を示した．図 17 より遠隔ノードへの遅延が増加するにつれてプログラム実行時間は増加し、HDFS 内部性能の影響が順当に反映されていることが分かる．

### 7.3 ラック設定適用時の性能

続いて Hadoop ラック設定を転置インデックスプログラムに適用した場合の高遅延環境が及ぼす影響について調べた．同じく Datanode3 台のうち 1 台が遠隔地に存在する構成を想定し、

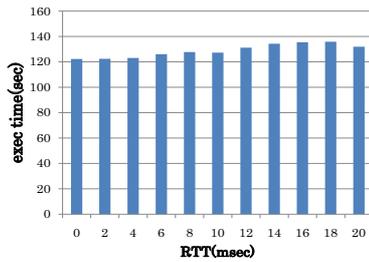


図 17 転置インデックス実行時間

プログラムを実行した際の処理時間を計測した（図 18，拡大図が図 19）。HDFS 性能では単純なラック設定を適用した場合ライトの性能は低下，リードの性能は上昇することが分かっていたが，プログラム実行時間に関しては単純なラック設定を行った場合（simple rack）が最も性能が低い傾向にあることが分かった。またデフォルトの場合（default）とレプリカ配置ポリシーを考慮したラック設定（optimized rack）を比較すると数回の逆転はあるもののレプリカ配置ポリシーを考慮したラック設定を適用した場合の方が実行時間が短い傾向が見られた。

HDFS ライト性能と類似した結果となったが，これはそもそも HDFS 性能ではリードよりライトの方が単位時間辺りの処理データ量スループットが 30～50% 程度低く，ライト性能がプログラム処理のボトルネックになる可能性が高いことによると思われる。今回の実験では個々の処理時間は高々 130 秒前後であったが，より大量のデータを処理する長期的なジョブを検討した場合，この性能差はますます拡大されるものと予想される。

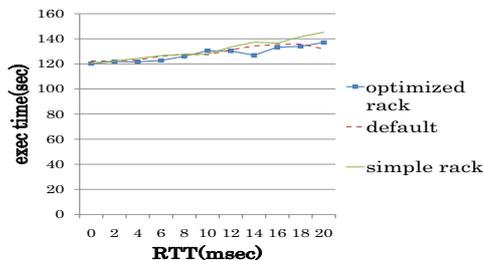


図 18 ラック設定適用時の転置インデックス実行時間

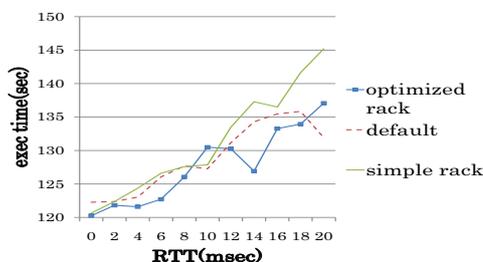


図 19 ラック設定適用時拡大

## 8. まとめと今後の課題

### 8.1 まとめ

ストレージ問題解決手法として HDFS に着目し，高遅延環境下におけるファイルシステム I/O の性能測定・パケット解析

を行った。Hadoop クラスタに高遅延ノードを導入するといくつかの性能劣化が見られ，システムによる自動的な遠隔アクセス回避などは行われないことが分かった。

ノードアクセス制御手法として HDFS ラック設定を適用させて性能測定を行ったところ，遅延ノードを含まない環境においても HDFS 性能に若干の影響があることが確認され，高遅延ノードを含むシステム構成においては遠隔ノードアクセスを制御することでライト性能の向上が確認された。

また実アプリケーション実行時の性能測定の結果，HDFS 性能と同様高遅延環境での性能劣化が確認された。ラック設定を適用した測定の結果では，HDFS におけるライト性能向上がプログラムの実行時間に寄与する傾向が確認された。

### 8.2 今後の課題

今後はより詳しいパケット解析や HDFS のソースコード解析などを行い，高遅延を含む実装での HDFS の振舞いをより詳細に分析する。具体的には遠隔アクセス頻度の最適化を目指し，より複雑なシステム構成モデルを用いた遠隔アクセス制御とその性能評価を行う。また遠隔アクセス頻度が処理性能にもたらす影響に関して多角的な評価を行い，HDFS 性能の総合的な向上のための指針を検討する。これらの結果から広域分散ファイルシステム性能向上のための手法を提案したい。

## 文 献

- [1] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "The Google File System", ACM SIGOPS Operating Systems Review, Vol.37, No.5, pp.29-43, December 2003.
- [2] Dhruba Borthakur, *HDFS Architecture*, 2008 The Apache Software Foundation.
- [3] Rocks Cluster: <http://www.rocksclusters.org/>
- [4] Wireshark: <http://www.wireshark.org/>