DEIM Forum 2012 C2-6

広域分散環境における分散ファイルシステム Hadoop のデータ配置手法の提案と実装

百瀬明日香† 小口 正人†

† 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 情報爆発時代を担うストレージシステムとして分散ファイルシステムを採用し，なかでも広域分散環境における性能向上に着目した．対象となるシステムとしては，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である Hadoop Distributed File System を採用した．高遅延環境における基本性能として遠隔ノードを含む Hadoop クラスタでファイルシステム I/O 測定のためのベンチマーク，いくつかの実用的なアプリケーションでの性能評価を行った．ファイルシステム I/O においてはリード性能において特に遅延の影響が謙虚であること，いくつかのアプリケーションにおいて遅延が大きいほど処理時間が増加する相関関係が見られること，また，必ずしも遅延の影響を受けないアプリケーションが存在することなどが確認された．こうした遅延の影響を軽減するため，遠隔アクセス制御手法の提案を行った．ファイルのレプリカ生成時，レプリカの分散先をラック毎に任意の割合に制御することで，明示的データ配置制御を実現した．提案手法を用いた性能測定により，今回採用した転置インデックス作成プログラムについて，遠隔アクセス制御を行うことによってデフォルトよりも性能向上することを確認した．またこうした制御について，遠隔アクセス回避と処理分散のバランスから導かれる，性能上の最適な分散割合が存在することが考察された．

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

A Study about the Remote Data Access Control for Hadoop Distributed File System

Asuka MOMOSE† and Masato OGUCHI†

† Ochanomizu University 2-1-1 Otsuka, Bunkyouku Tokyo 112-8610 JAPAN
E-mail: †momo@ogl.is.ocha.ac.jp, ††oguchi@computer.org

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っている．

こうした分散ファイルシステムのなかでも，本研究では特に分散ファイルシステムにおけるデータ・アフィニティに着目した．データ・アフィニティとはデータとその処理を行う計算ノードの位置を近くなるよう配置することにより，ネットワーク通信量の削減とデータ処理効率の向上が望めるという概念である．データの位置を考慮した処理スケジューリングをデータ・アフィニティ・スケジューリングと呼ぶ．このようなシステムではジョブが常にデータローカルに処理されることで高い並列分散処理性能・フォールトトレーランス性を維持することが知られており，近年特に処理データ量が爆発的に増大していることに伴い重要性を増している概念である．

一方で，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という点から，このようなデータ・アフィニティな分散ファイルシステムの広域分散環境における性能に着目した．こうした実装における運用に際しては，遠隔地へのアクセスによるネットワーク遅延がデータ処理効率に影響を及ぼすことが分かっている [1]．そこで本研究では遠隔ノードへのデータ配置制御手法を提案および実装し，遠隔データアクセス制御を用いて広域分散環境における分散ファイルシステムの実装に関する適切なデータ配置の検討を行った．分散ファイルシステムの実装としてはオープンソースソフトウェア Hadoop

Distributed File System（以下 HDFS）[2] を使用した．

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 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，ファイルシステムである HDFS，分散処理アルゴリズムである MapReduce，データベースである 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．

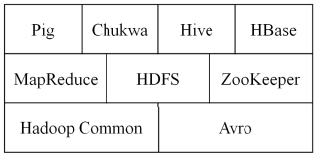

図 1　Hadoop のサブプロジェクト（2010 年 5 月現在）

2.2 Hadoop におけるデータ・アフィニティ

HDFS はデータ格納のマスタである Namenode，スレーブである Datanode と，ジョブ管理を行う JobTracker，個々のジョブを実行する TastTracker などのプロセスからなる（図 2）．Hadoop における特徴的な実装である分散処理アルゴリズム MapReduce では，プログラムを Map，Reduce というフェーズに分けて並列実行するが，このとき各 Mapper，Reducer は TaskTracker に分配され，当該ノードのローカル Datanode に存在するデータブロック（レプリカと呼ぶ）に対して処理を実行する．Hadoop における処理はこうしたデータ・アフィニティなスケジューリングを考慮し得る環境で行われるため，レプリカをどのように配置するかという問題が，計算処理性能などにも大きな影響を及ぼすと考えられる．

3. 実験環境

クラスタ自動構築・管理ツール Rocks [3] を用いて構築したローカルクラスタ上に Hadoop-0.20.2 をインストールし，Hadoop クラスタを構築した．マシンスペックは表 1 に示す通りである．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 2.0MB とし，測定のためのベンチマークには，Hadoop

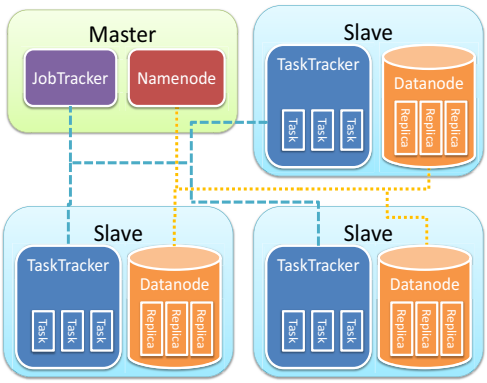

図 2　Hadoop のプロセス

に付属の TestDFSIO プログラムと Map/Reduce 処理をメインに行うベンチマークである自作の転置インデックスプログラムを使用した．

転置インデックスプログラムでは 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．

4. 高遅延環境における基本性能

4.1 実験概要

人工遅延装置 dummynet を使用して高遅延接続を含む模擬的な広域分散環境を構築し，Namenode1 台と Datanode3 台のうち Datanode1 台が遠方に存在すると仮定した環境での測定を行う（図 3）．分散されるファイルのレプリカ数は 2 とした．ただしレプリカがどこに配置されるかは高遅延ノードの物理的な有無にはよらず，デフォルトではノード間で均等になるよう分散配置されることが分かっている．

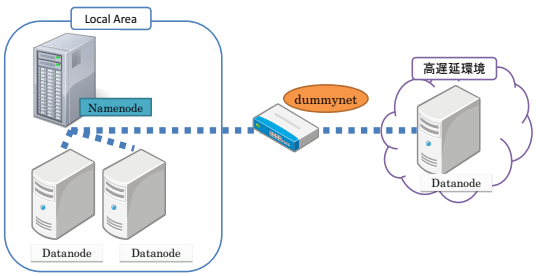

図 3　システム構成

表 1　マシンスペック

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

4.2 TestDFSIO ベンチマーク

TestDFSIO プログラムでは 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，“ Throughput ”は，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．

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 4, 図 5）．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い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．

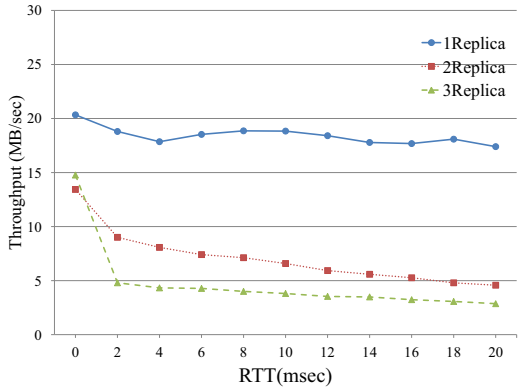

図 4　Write スループット

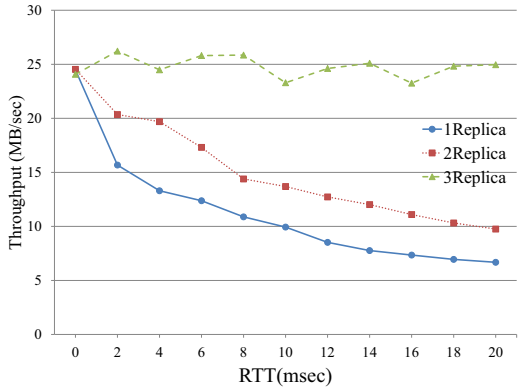

図 5　Read スループット

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．

4.3 転置インデックスプログラム

転置インデックスプログラムを使用し，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 6 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．

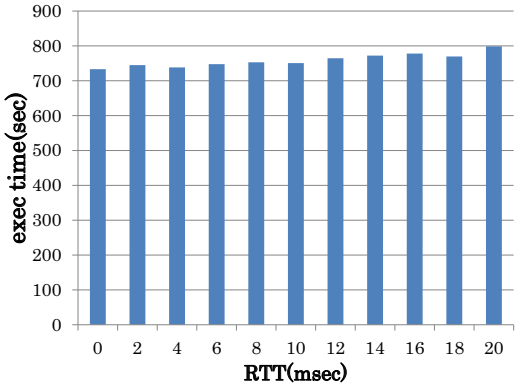

図 6　転置インデックス実行時間

4.4 その他のアプリケーションによる評価

高遅延環境における Hadoop の振舞について，さらにいくつかのアプリケーションで検討を行った．まず RandomWrite プログラムでバイナリデータを生成したときの性能を測定した．100MB のデータを生成するジョブを各ノードに与え, 合計 1GB の書き込みを行った．次に RandomWrite で生成したファイルをソートするプログラムを実行した．

RandomWrite の実行時間は遅延が大きくなってもほとんど変化がないのに対し，Sort の実行時間は RTT の増加に伴い上昇した (図 7，図 8)．TestDFS I/O ベンチマークの結果と同様に，ライトではバッファに遅延が吸収されるのに対して，Sort ではファイルの読出を行っているため遅延の影響が発生していると考えられる．これらの結果から，必ずしも高遅延の影響を受けないアプリケーションも存在することが考察された．

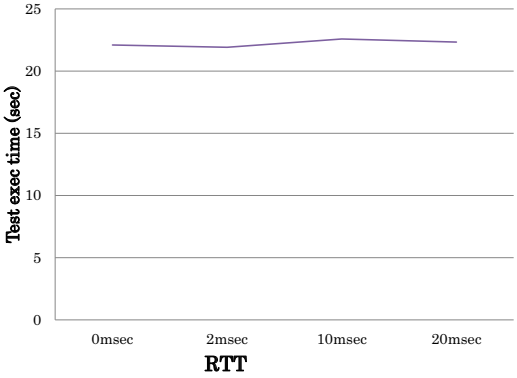

図 7　RandomWrite 実行時間

5.　パケット解析による動作解析

本章ではデフォルト状態の Hadoop において，遠隔アクセス制御が行われていないことを説明する．

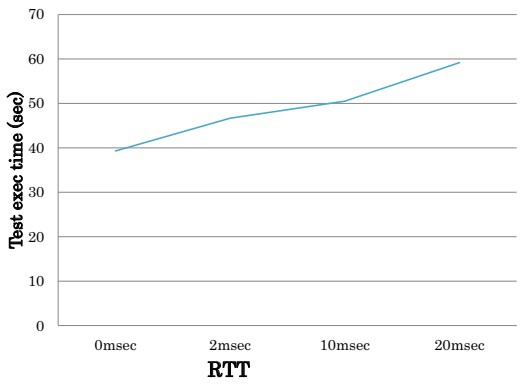

図 8　Sort 実行時間

5.1　システム構成

性能測定と併せて，ファイルシステム内で実際にどのような通信が行われたかパケット解析を用いた調査を行った．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，システム構成は基本性能測定と同様である（図 3）．Namenode のみ Wireshark をインストールし，Namenode と各 Datanode 間で行われたパケット通信を抽出した．

5.2　Datanode 毎の通信量の比較

レプリカ数 1 において性能測定と同様の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 9）．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．

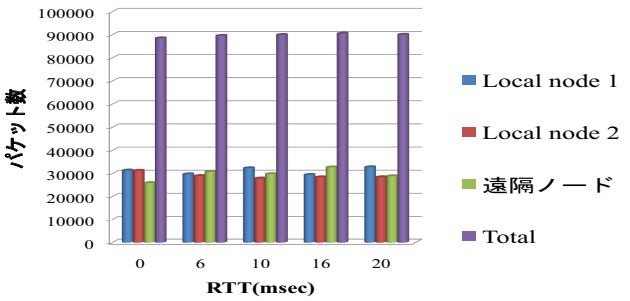

図 9　各ノードとの通信量

続いて同じくレプリカ数 1 においてシーケンシャルライト，リードを交互に 5 回ずつ実行した際の，各ジョブで発生したパケット通信量を調べた（図 10，図 11）．高遅延ノードの RTT=20msec としている．各回毎にパケット数にはばらつきがあり，Datanode3 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．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 Namenode の仕事を一部肩代わりする実装によるものと見られる．ラックを設定した場合はパケット通信量には 50% 程度の変化が見られたが，デフォルトの状態でも 10〜30% くらいの変動があることが分かる．ただしこちらの場合ではローカルエリアのノードと高遅延環境のノードの間で Namenode からのアクセス頻度に差は見られず，通信量の多いノードはランダムに選出されている．

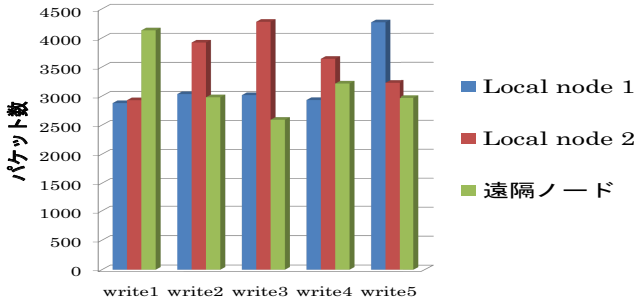

図 10　write 実行時のパケット数

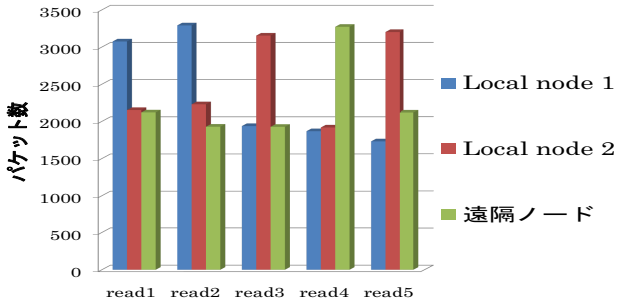

図 11　read 実行時のパケット数

5.3　パケット解析考察

以上の結果から HDFS を高遅延ノードを含む構成で実装すると応答の遅いノードに対しても均等にデータアクセスが行われ，デフォルトの実装ではシステムによるアクセス制御は行われないということが確認された．

6.　遠隔データアクセス制御のための提案手法

基本性能の測定結果より，HDFS クラスタ内に遠隔ノードが含まれることによってシステムの性能劣化が起こる場合があることが分かっている．また前章より，こうした遠隔ノードに対して Hadoop によるアクセス制御などの処理は行われていないことが確認された．そこで以下ではファイルシステムを書き換え，高遅延環境における遠隔アクセス制御がシステムに及ぼす影響についての検討を行う．

6.1　提案手法におけるシステム構成

提案手法では，Hadoop の性能向上のためネットワーク遅延の大きい遠隔ノードへのアクセスを制御することでシステムの性能改善を行うことを目的とする．本手法におけるシステム構成は図 12 の通りであり，ラック単位でのアクセス制御を実装した．

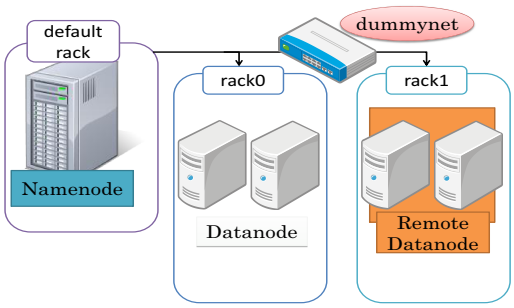

図 12　提案手法のシステム構成

6.2　制 御 手 法

Hadoop のラック設定を利用し，(i)1st レプリカはジョブが現在存在するラックへ配置，(ii)2nd レプリカは指定した確率で Remote rack へ，残りを Local rack へ配置，というデータ配置ポリシーに基づいて HDFS を実装した．この際のブロック配置は表 2）のようになり，Remote rack へのデータ配置を最大で 30%軽減する．この値は，Hadoop の耐故障性維持のため必要最低限のデータ分散であるものとした．このようなデータ配置制御を与えた場合の HDFS 性能について，ベンチマークプログラムによる測定を行う．

表 2　ブロック配置 (Remote rack：Local rack)

Math.random 係数	ブロック比	データ配置
0.5 , 0.5	10：10	50% - 50%
0.4 , 0.6	9：11	45% - 55%
0.3 , 0.7	8：12	40% - 60%
0.2 , 0.8	7：13	35% - 65%
0.1 , 0.9	6：14	30% - 70%

6.3　変更後のブロック配置

実際に上記のように変更した HDFS において，一定量のデータ書き込みを与えた際のブロック配置は表（3）の通りである．このとき Math.Random 係数は 0.8 に指定しており（表 2）で求めた通りの 13:7 の割合でデータ配置制御が行われることが確認された．

表 3　データ書込後のブロック配置 (Remote rack：Local rack)

Rack	Node 比	Blocks
rack0	Datanode1	122
rack0	Datanode2	127
rack1	Datanode3	62
rack1	Datanode4	65

7.　遠隔アクセス制御時の HDFS 性能

7.1　転置インデックスプログラムを用いた性能評価

前章の通り提案手法を用いて実装した HDFS において，遠隔アクセス制御の割合を変化させ，HDFS のプログラム処理性能へ与える影響を調べた．転置インデックスプログラム実行時間を比較すると遠隔ノードへのアクセスを制限した場合の方が，制限を行わない場合よりも概ね性能が良い傾向が見られる（図 13）．デフォルトの HDFS 性能と提案手法を比較した場合，遅延が最も大きくなる RTT200msec において最大で 51.9% の処理時間の減少が見られた．また提案手法を用いた性能同士を比較した場合，遠隔ノードのデータ配置割合をローカルと均等である 50%から最小で 30%まで徐々に減らして行くと，制御 50%に比べ制御 30%の方が平均して 15%程度性能向上することが確認された．今回の測定で採用したアクセス制御割合 30-50%の範囲内では，高遅延になるほど、アクセス制御割合の上昇に伴って，プログラム処理時間が短くなる結果が得られた．

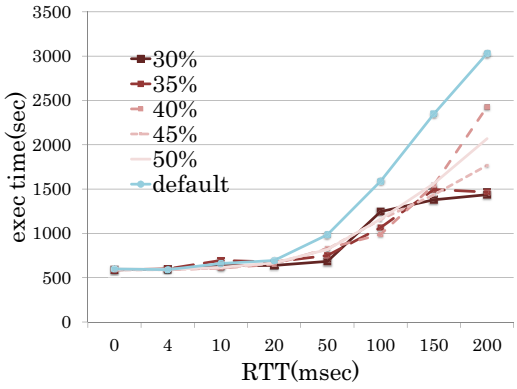

図 13　提案手法適用時の転置インデックス実行時間

この点に関して，既存研究より [1]，こうした振舞は HDFS ライトの結果と似た傾向であることが知られている．図 14，図 15 より遠隔ノードへのアクセスを軽減した場合（optimized rack），ライトではデフォルトよりも平均的にスループットが上昇し，リードでは反対に性能が低下することが分かっている．ここで，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，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．

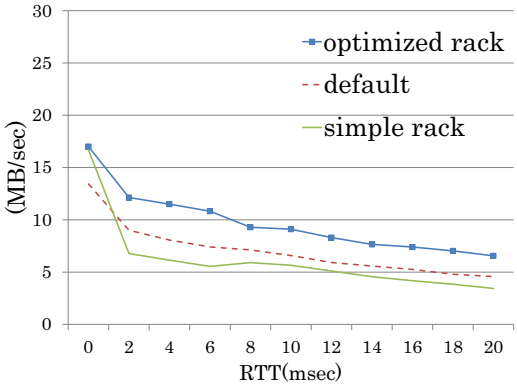

図 14　Write スループット

転置インデックスプログラム処理性能と HDFS ライト性能の傾向に類似が見られた点に関して，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，今回使用したプログラムではラ

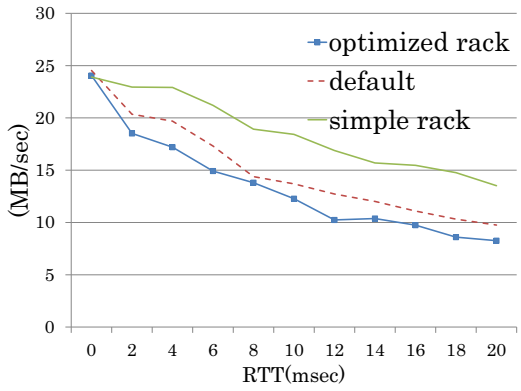

図 15　Read スループット

イト性能がプログラム処理のボトルネックとなる可能性が高いことが推察された．

7.2　提案手法の性能評価まとめ

前節の結果より，本研究の実験環境において提案手法を用いた遠隔アクセス制御を加えることで，転置インデックスプログラム実行時間にデフォルトから最大で 51.9%の差が得られ（RTT200msec 時），RTT0msec から RTT200msec の間では平均して 22.9%程度，遠隔アクセス制御による実行時間の減少が見られた．

なお，このとき RTT20msec は東京-大阪間の往復遅延時間，RTT200msec は東京-米国東海岸間の往復遅延時間を想定している．

8.　まとめと今後の課題

8.1　ま　と　め

情報爆発時代を担うストレージシステムとして分散ファイルシステムを採用し，なかでも広域分散環境における性能向上に着目した．対象となるシステムとしては，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である Hadoop Distributed File System を採用した．

高遅延環境における基本性能として遠隔ノードを含む Hadoop クラスタでファイルシステム I/O 測定のためのベンチマーク，いくつかの実用的なアプリケーションでの性能評価を行った．ファイルシステム I/O においてはリード性能において特に遅延の影響が謙虚であること，いくつかのアプリケーションにおいて遅延が大きいほど処理時間が増加する相関関係が見られること，また，必ずしも遅延の影響を受けないアプリケーションが存在することなどが確認された．

こうした遅延の影響を軽減するため，遠隔アクセス制御手法の提案を行った．ファイルのレプリカ生成時，レプリカの分散先をラック毎に任意の割合に制御することで，明示的データ配置制御を実現した．提案手法を用いた性能測定により，今回採用した転置インデックス作成プログラムについて，遠隔アクセス制御を行うことによってデフォルトよりも性能向上することを確認した．またこうした制御について，遠隔アクセス回避と処理分散のバランスから導かれる，性能上の最適な分散割合が存在することが考察された．

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．

9.　謝　　辞

本研究を進めるにあたり，独立行政法人産業技術総合研究所の竹房あつ子氏，中田秀基氏，高野了成氏，工藤知宏氏には，お忙しいなか貴重なご指導，ご助言賜りましたことを厚く御礼申し上げます．

文　　献

[1] 百瀬 明日香，小口 正人：「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」，電子情報通信学会　DE 研&PRMU 研 (パターン認識・メディア理解研) 共催 6 月第一種研究会，信学技報，Vol.111，No.76，pp.19-24，2011 年 6 月．
[2] Dhruba Borthakur，*HDFS Architecture*，2008 The Apache Software Foundation.
[3] Rocks Cluster：http://www.rocksclusters.org/
[4] Wireshark：http://www.wireshark.org/
[5] 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.
[6] Tom White，玉川竜司，兼田聖士訳：*Hadoop*，2010 O'Reilly Japan，Inc.
[7] Jason Venner：*Pro Hadoop*，2009 Apress.
[8] 建部 修見，曽田 哲之：「広域分散ファイルシステム Gfarm v2 の実装と評価」，情報処理学会研究報告，2007-HPC-113，pp.7-12，2007 年 12 月．
[9] 三上 俊輔，太田 一樹，建部 修見：「広域分散ファイルシステム Gfarm 上での MapReduce を用いた大規模分散データ処理」，SWopp2010，Vol.2010-HPC-126，2010 年 8 月．