Vol.2010-DBS-151 No.10
2010/11/12

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に関する一考察

百瀬　明日香[†1]　小口　正人[†1]

情報爆発の現代において大きな障害となってい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，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．また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ことを受けて，遠隔地にファイルのバックアップを保持し自然災害やテロリズムなどによる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．本研究ではこうした分散ファイルシステムについて，高遅延環境を含む実装における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を調査した．

また本研究ではこのような分散ファイルシステムのプラットフォームとしてGoogle社のGoogle File System（以下GFS）を採用し，このオープンソース版であるHadoop Distributed File System（以下HDFS）を実験環境として使用した．

A Study of Remote Data Access for Distributed File System over A Wide Area

Asuka Momose[†1] and Masato Oguchi[†1]

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. In addition, with development of Network Infrastructure such as the Internet, to disperse files over a wide area gradually gain practicality; therefore we aim at DFS implementation over a wide area environment to back up data to prevent large-scale loss of data as in the case of calamity or terrorism. In this research, we have evaluated performance of DFS in a long-latency environment and analyzed packets flowing between nodes, and investigated the characteristic of remote data access in DFS over a wide area. We have adopted Google File System as a platform of the DFS, and used Hadoop Ditributed File System (HDFS), an open source system of Google File System, for experiment.

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・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による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提案している．これは分散ファイルシステムではハードウェアの制約が少なく，高いスケーラビリティが期待できるため，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．また既存のネットワークプロトコルを用いて広域分散環境を構築可能であることから，ファイルレプリケーション手法に着目した実験に焦点を当てている．

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MapReduce，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に採用した．

GFSにおいてファイルはチャンクという単位に分割され，分割したチャンクを複製したものが，異なるチャンクサーバ上に保存される（**図1**）．ファイルを複製する個数をレプリカ数と言い，デフォルトは3である．なおレプリカ数1が，“ オリジナルファイルのみ ”の

†1 お茶の水女子大学
Ochanomizu University

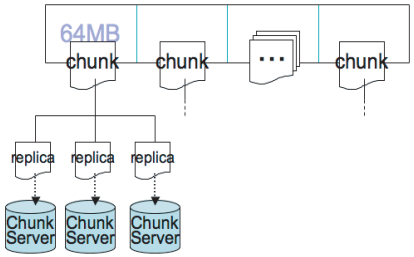

図 1 GFS におけるファイル
Fig. 1 File concept of GFS

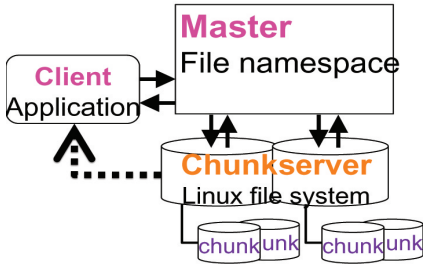

図 2 GFS アーキテクチャ
Fig. 2 GFS Architecture

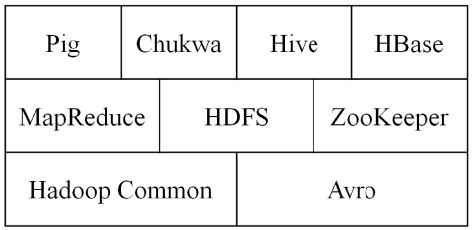

図 3 Hadoop のサブプロジェクト（2010 年 5 月現在）
Fig. 3 The Apache Hadoop Project

状態である．GFS はファイルのメタデータを保持するマスタと実際にデータが格納されるチャンクサーバからなり，ファイルを分散および複製して保存することによって耐故障性と対多クライアントでの高いパフォーマンスを維持している（**図 2**）．

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，MapReduce 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 3**）．本研究ではこのうちの特に HDFS と Hadoop MapReduce を実装したものを実験環境として使用した．

3. 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 1** に示す通りである．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 HDFS におけるマスタ，Datanode とは HDFS 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．

4. 基本性能測定

4.1 実験概要

まず HDFS の基本性能としてローカルエリアのクラスタ環境上で 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，各々の処理効率を測定した．測定は 12 回ずつ行い，最大値と最小値を除いた平均をその台数における平均としている．HDFS 性能測定においては TestDFSIO プログラムで得られるパラメータの一つである“ Throughput ”を参照した．これは分散された各ジョブの単位時間における処理データ量の平均を表している．

4.2 ノード数スケール

ファイルのレプリカ数 1 でノード数を変化させて性能を測定した（**図 4**，**図 5**）．ライト，リードともに各ノードのスループットは台数によらずほぼ一定でありスループットの総和は線形に上昇している．クライアントがシステムから完了報告を受けるまでの所要時間である実行時間はノード数の増加に合わせて減少することも確認されている．これらの結果から，一定のスループットのノードを複数束ねることによってアプリケーションから見たファイルシステム全体のスループットについても，各ノードのスループットの総和と同様にノード数

表 1 マシンスペック
Table 1 Machine Specification

	Machine	CPU	Main Memory	OS
Master node	HP Work-Station xw8200	Intel(R) Xeon(R) @3.6GHz	4.0GB	Linux 2.6.9-55.0.2.Elsmp(CentOS 4)
Slave node	Dell PowerEdge SC1430	Quad-Core Intel(R) Xeon(R) @1.60GHz	2.0GB	Linux 2.6.9-55.0.2.Elsmp(CentOS 4)

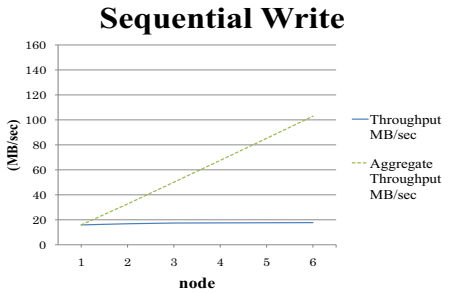

図 4 Write スループット
Fig. 4 Write throughput

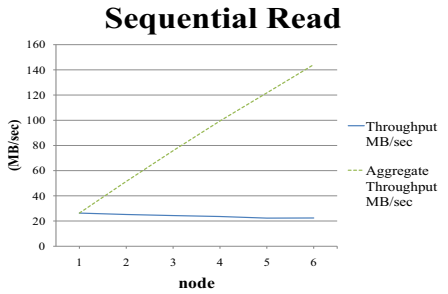

図 5 Read スループット
Fig. 5 Read throughput

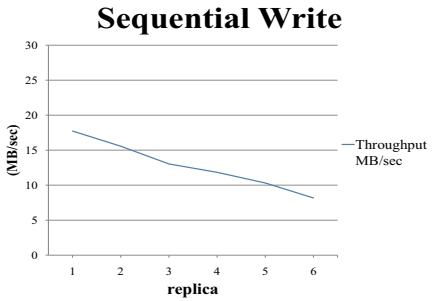

図 6 Write スループット
Fig. 6 Write throughput

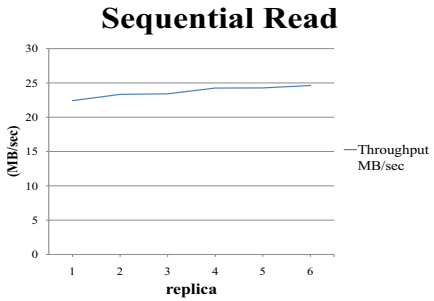

図 7 Read スループット
Fig. 7 Read throughput

に比例して上昇していることが分かる．

4.3 レプリカ数スケール

次にレプリカ数を変化させたときの HDFS 性能についても同様に測定を行った（**図 6**, **図 7**）．レプリカ数の増加につれてライトのスループットは低下している一方，リードでは処理性能がやや向上する結果となった．これはライトでは書き込むデータ量が増加しているのに対し，リードで読み出すデータ量は変化せず，むしろアクセスすべきノード数が減るためであると思われる．すなわち，レプリカ数が少ない場合には，一連の読み出しがいくつものノードにまたがって行われる可能性が高くなるが，レプリカ数が増えると同一のノードで続けて読み出しを行える可能性が増え，処理性能の向上につながったものと考えられる．

4.4 基本性能測定考察

以上の実験結果よりローカルエリアのクラスタ環境における HDFS 性能はノード数増加につれて性能向上することが確認された．これは HDFS が踏襲する GFS に関して，Google 社が自社で行った性能測定の結果とも合致する結果となった[1)]．

5. 高遅延環境における測定

5.1 実 験 概 要

続いて高遅延環境下での性能測定を行った．Namenode1 台と Datanode3 台を使用し，Datanode のうちの 1 台を人工遅延装置 dummynet を介して接続することで，4 つのノードのうち 1 つが遠方に存在すると仮定した環境での測定を行う（**図 8**）．分散されるファイルのレプリカ数は 1 とした．

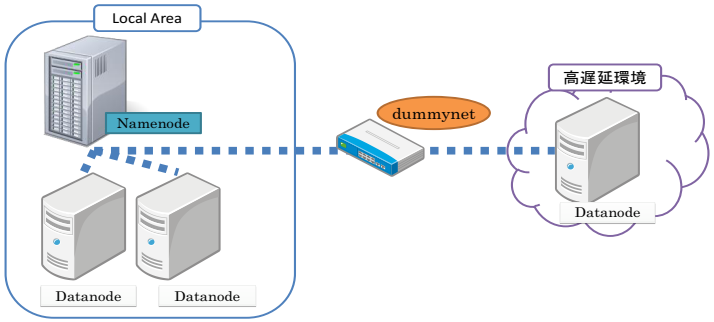

図 8 システム構成
Fig. 8 System Structure

5.2 ノード間距離を考慮しないファイル分散

基本性能測定と同様に，10MB のファイルを 100 個シーケンシャルライトで作成し，それをシーケンシャルリードするプログラムを実行した．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 までと変化させたときの各々のスループットを測定した．また，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．

5.2.1 測 定 結 果

レプリカ数 1 の場合を比較するとライトにおけるスループットはほぼ一定であるのに対

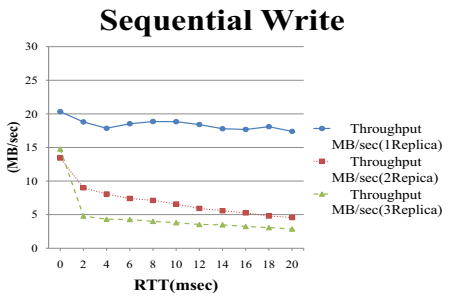

図 9 Write スループット
Fig. 9 Write throughput

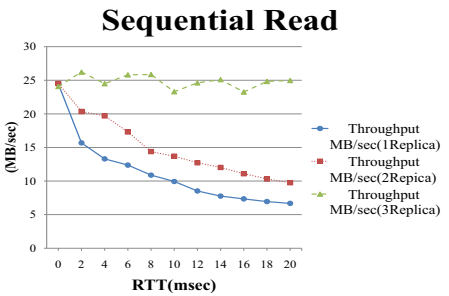

図 10 Read スループット
Fig. 10 Read throughput

し，リードでは大きく低下した（**図 9**，**図 10**）．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 であるため定期的に高遅延環境へのアクセスが行われ，結果スループットが低下したものと考えられる．

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．ライトでは基本性能の場合と同様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．

6. ラックを考慮したファイル分散

Hadoop のラック対応

Hadoop におけるネットワークトポロジは「ラック」という概念で認識される．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．HDFS クラスタ内のネットワークトポロジは，マスタを起点とし各スレーブの位置をラック ID というパラメータで定義付けている．

6.1 単純なラック設定適応時の性能

基本性能測定と同様の実験を，HDFS のラック設定を適応した実装環境で行った．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 11**）．これは物

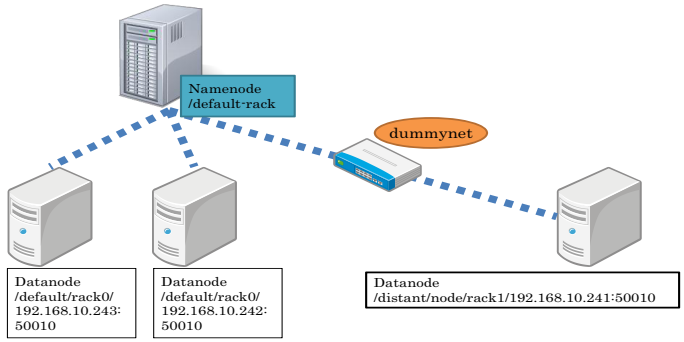

図 11 ラック ID の付与
Fig. 11 Rack ID setting

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．

このときの HDFS 性能をラック設定を行わない場合と比較した（**図 12**，**図 13**）．ライトではスループットが若干低下しているのに対してリードのスループットは上昇している．これは，単一ノードに集中するブロック数が増えたため，リードアクセスでノードを切り替える確率が減少したことによると見られる．この性能向上は，究極的には単一ノードにブロックを集中させた場合にリード性能が向上するのと同等の議論で，遅延の影響にはよらない問題である．

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 2**）．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機能がラックポリシーよりも優先されたことによるものと見られる．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ていることが分かる．

6.2 レプリカ配置ポリシーを考慮したラック設定適応時の性能

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 3**）．近傍ノードを別ラックとして指定することで，間接的に遠隔ノードへのアクセス機会を減らしている．

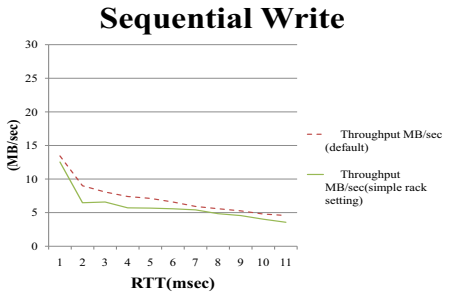

図 12 Write スループット
Fig. 12 Write throughput

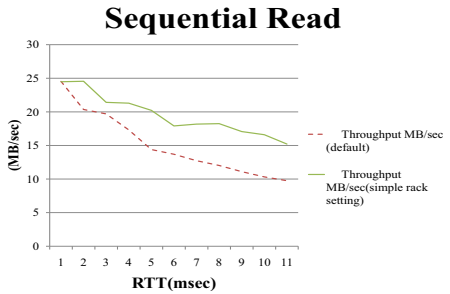

図 13 Read スループット
Fig. 13 Read throughput

このときの性能測定結果を，ラック未設定の場合，ラックを単純に設定した場合と比較する（**図 14**，**図 15**）．ライトでは平均的にスループットが上昇し，リードのスループットは反対に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．リードの性能はブロックが単一ノードへ集中度する度合いが大きいほど向上している．

6.3 レプリカ数 1 でラック設定を適応した場合

比較のため，同様の実験をレプリカ数 1 で行った場合の結果を記載する（**図 16**，**図 17**）．レプリカ数 2 の場合に有効であったラック設定を適応しても，この場合性能はまったく変化

表 2 データ書込後のブロック数
Table 2 Written Block

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

表 3 ラック設定を変更した場合の書込ブロック数
Table 3 Written Block -Considered Rack setting-

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

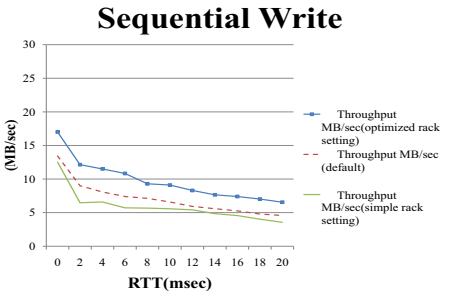

図 14 Write スループット
Fig. 14 Write throughput

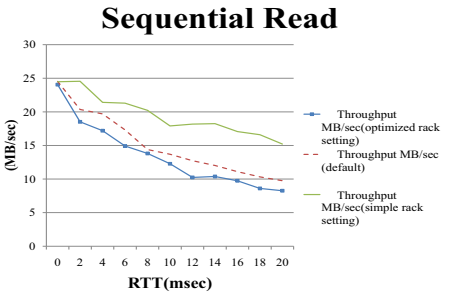

図 15 Read スループット
Fig. 15 Read throughput

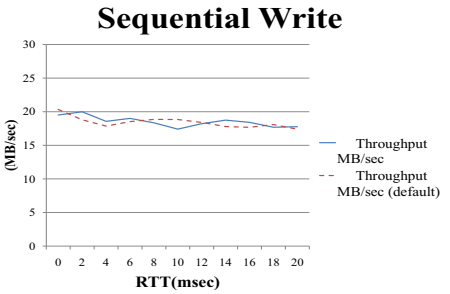

図 16 Write スループット
Fig. 16 Write throughput

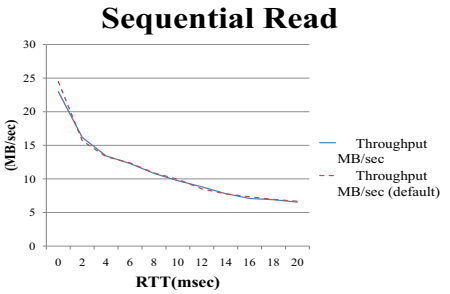

図 17 Read スループット
Fig. 17 Read throughput

しないことが分かる．これはファイル分散の傾向を確認するとより明らかで，レプリカ数 1 の場合はラックを考慮したファイル分散が行われていないことが確認できる（**表 4**）．これらの結果から HDFS がラックを考慮してファイルの分散先を制御するのは，ブロックの複製を別ノードにコピーする瞬間であることが予測される．

6.4 考　　察

以上の実験結果より HDFS において有効なラック設定のためには，必ずしも物理的構造によらず，レプリカ配置ポリシーとノードへの負荷分散の双方を考慮した設定を行うことが必要であることが分かった．その例として近傍ノードへのアクセスを増加させる代わりに遠隔ノードへのアクセスを減らしたところライトにおける若干の性能向上が確認され，レプ

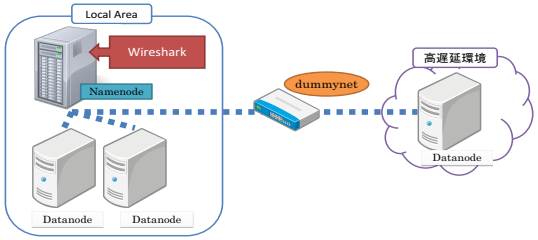

図 18 システム構成
Fig. 18 System Structure

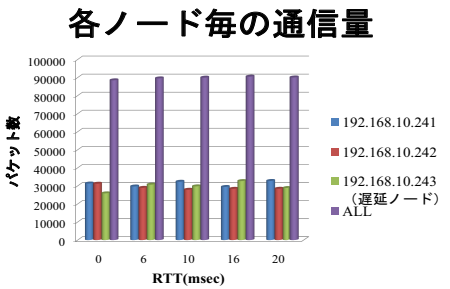

図 19 各ノードとの通信量
Fig. 19 Number of Exchanged Packets among Namenode and each Datanode

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．またラック設定のみを反映した場合ネットワークトポロジを考慮したアクセス制御が行われるのはブロックの複製を分配する瞬間であることから，ノード数とレプリカ数を変化させ，ブロックの分散が様々な頻度で行われる場合の測定が求められる．

7. パケット解析

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・アナライズ・ソフトウェアである Wireshark[4] を使用し，システム構成は図の通りである（**図 18**）．今回は 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．

7.1 ラック設定を適用しないときの通信量

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 19**）． 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

表 4 データ書込後のブロック数（レプリカ数 1）
Table 4 Written Block(1Replica)

	Rack ID	Block	比率（%）
Local Datanode1	/default/rack0	566	36.5
Local Datanode2	/default/rack0	419	27.0
Distant Datanode	/destant/node/rack1	566	36.5

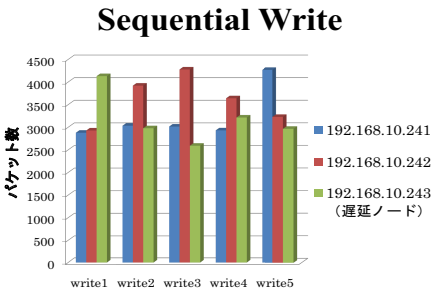

図 20 write 実行時のパケット数
Fig. 20 Exchanged Packets under Write Program

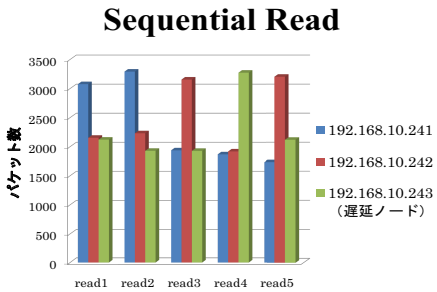

図 21 read 実行時のパケット数
Fig. 21 Exchanged Packets under Read Program

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．

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 20**，**図 21**）．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．

7.2 パケット解析考察

以上の結果から HDFS を高遅延で実装した際，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ということが確認された．なおラック設定適用時の通信量の変化についても今後解析を行う予定である．

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分散ファイルシステム Hadoop に着目し Rocks をインストールしたクラスタ上で HDFS 環境を構築，まず初めに基本性能を測定した．HDFS の性能についてはリード，ライトともにノード数が増えるほど処理効率が上昇することが確認された．続いて人工遅延装置 Dummynet を導入し，広域分散環境を模した高遅延環境下で測定を行ったところ，レプリカ数 1 の場合では特にリード処理において遅延の影響が顕著であり，反対にレプリカ数を増やして行くとライト処理で遅延の影響が大きくなることが分かった．

一方 HDFS のラック設定を適応させて性能測定を行ったところ，単純に物理ノード通りのラック ID を付与した場合では，レプリカ配置ポリシーとの競合で必ずしも性能向上しないことが判明した．この問題に対してはレプリカ配置ポリシーを考慮したラック設定を適用することで性能向上することが確認された．

またパケット・アナライズ・ソフトウェア Wireshark を使用して，Namenode と各 Datanode 間のパケット通信の振舞を調べた．その結果 HDFS では管理者が明示的に設定しない場合，遅延ノードへの書込回避などのアクセス制御は行われず，パケットロスやパケット再転送による偶発的なアクセス制御も発生していないことが分かった．

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 HDFS クラスタを構築するノード数や分散されるレプリカ数の相関から適切なラック設定を見極め，遠隔アクセスを含む広域分散ファイルシステムに有用なアクセス制御を検討する．またこれらの結果から，広域分散ファイルシステムの性能向上のための手法を提案したい．

参 考 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/
5) Tom White，玉川竜司，兼田聖士訳：*Hadoop*，2010 O'Reilly Japan, Inc.
6) Jason Venner：*Pro Hadoop*，2009 Apress.