高遅延環境における 分散ファイルシステムHadoopの動作解析

百瀬　明日香[†1]　　小　口　正　人[†1]

情報爆発の現代におけ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問題に応える手法として，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．本研究では，自然災害やテロなどの大規模なデータ損失に対応したバックアップが可能である広域分散環境における分散ファイルシステムの運用に着目し，高遅延環境を含む実装における性能評価を行った．またノード間を飛び交うパケットの解析などを行うことでより詳細な分散ファイルシステムの動作解析を行った．これらの結果から実用的な分散ファイルシステム活用手法の確立を目指す．

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．

Analysis of Hadoop Distributed File System over a long-latency environment

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 this research, 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, and evaluated performance of DFS in a long-latency environment. In addition, we analyzed packets flowing between nodes and made much detailed analysis of DFS. As a result, we intend to realize a practical utilize of DFS.

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

情報爆発時代の現代における企業や個人のデータ処理の負荷やストレージコストの問題に応える手法として，コモディティなハードウェアで高度な集約処理を行う分散ファイルシステム (以下 DFS:Distributed File System) に注目が集まっている．本研究では, 自然災害やテロなどの大規模なデータ損失に対応したバックアップが可能である広域分散環境における DFS の運用に着目し, 高遅延環境を含む実装における性能の評価を行うことでより実用的な DFS の活用手法の確立を目指す.

本論文ではまず HDFS のクラスタ環境におけるファイルシステム I/O と Hadoop MapReduce の性能を測定し，評価を行った．始めに基本性能評価としてローカルなクラスタエリア環境における性能測定を行った．また高遅延環境で Hadoop を実装し同様の性能測定を行い，模擬的な広域環境における性能評価を行った．続いてファイルシステム I/O 実行時の各 Hadoop ノード間に飛び交うパケットを収集，解析を行い，性能測定の結果と照らし合わせること，でより詳細な HDFS の動作解析を目指した.

2. 分散ファイルシステム

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]．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

本研究では GFS のオープンソース版である Hadoop Distributed File System(以下 HDFS)

†1 お茶の水女子大学
Ochanomizu University

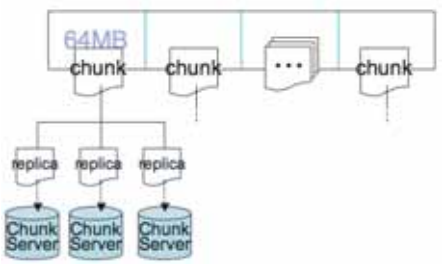

図 1 GFS におけるファイル

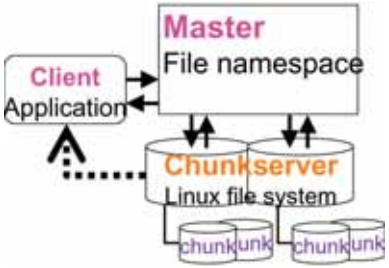

図 2 GFS アーキテクチャ

を使用した [2]．Hadoop は Yahoo Inc. が開発しているオープンソースソフトウェアであり，GFS に相当する HDFS，MapReduce に相当する Hadoop MapReduce，Bigtable に相当する hBase から構成される．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 (図 3)．

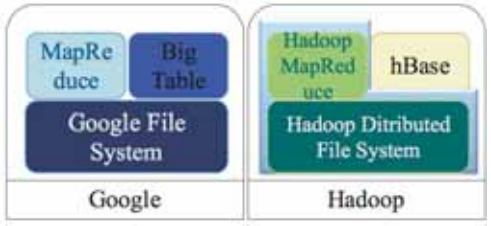

図 3 Google，Hadoop システム相関図

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 CPU:Quad-Core Intel(R)Xeon(R)1.60GHz，メインメモリ:2.0GB,OS:Linux2.6.9-55.0.2.ELsmp(CentOS 4.5) 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 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レプリカ数は 1 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，Sort プログラムを使用した．

4. 基本性能測定

4.1 HDFS 性能測定

4.1.1 レプリカ数 1 の場合

まず HDFS の基本性能の評価を行った．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，各々の処理時間を測定した．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うち，“ Test exec time ” と“ Throughput ”を参照した．Test exec time はクライアントから見たプログラム全体の実行時間であるのに対し，Throughput 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．

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，図 6，図 7)．

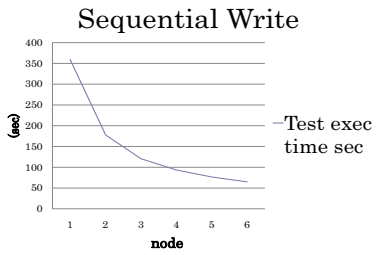

図 4 write 実行時間

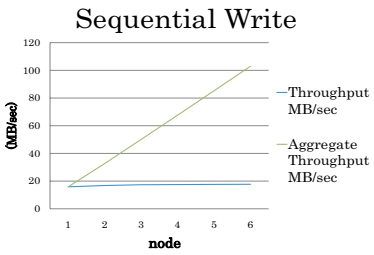

図 5 write スループット

write，read 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．実行時間はノード数の増加に合わせて減少している．このことから，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に比例して上昇していることが分かる．

4.1.2 複数レプリカの場合

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 8，9，

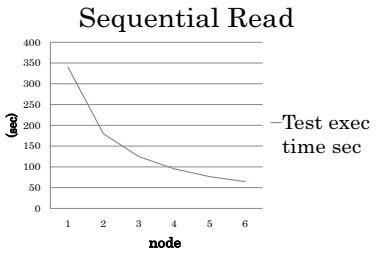

図 6 read 実行時間

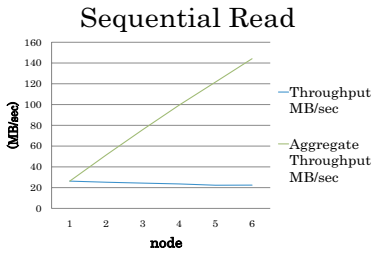

図 7 read スループット

10，11)．レプリカ数の増加につれて write のスループットは低下している一方，read では処理性能がやや向上する結果となった．これは，write では書き込むデータ量が増加しているのに対し，read 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．

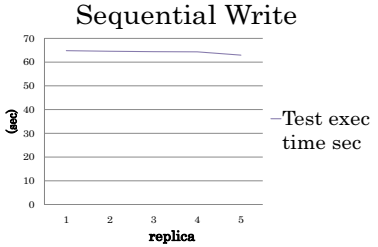

図 8 write 実行時間

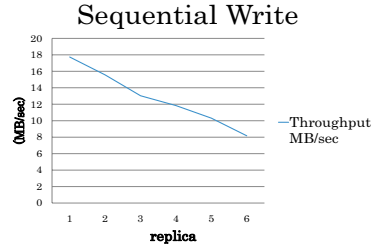

図 9 write スループット

4.2 MapReduce 性能測定

次に MapReduce の基本性能を測定するためソート プログラムを利用した性能評価を行った．ランダムなバイナリデータをソートするプログラムを実行し，実行時間を測定した．実行時間は台数の増加に合わせて減少し，ノード数に比例して性能向上することが確認された (図 12)．

4.3 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いては HDFS 性能，MapReduce 性能ともに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

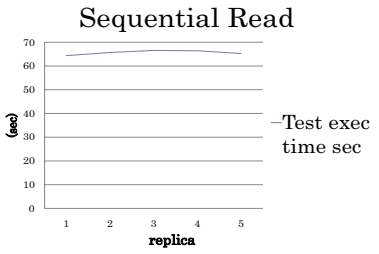

図 10 read 実行時間

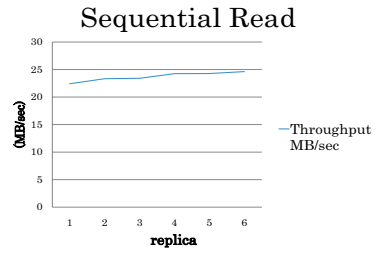

図 11 read スループット

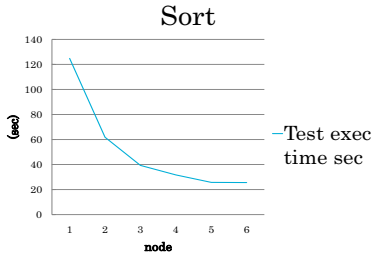

図 12 Sort 実行時間

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．Namenode 兼 Datanode を 1 台と Datanode を 2 台使用し，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，3 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 13)．分散されるファイルのレプリカ数は 1 とした．

5.2 HDFS 性能測定

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

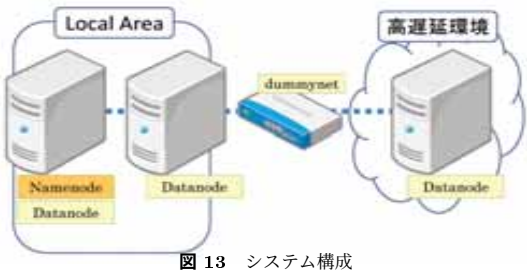

図 13 システム構成

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．write においては実行時間がやや上昇し，スループットは若干低下している (図 14，図 15)．

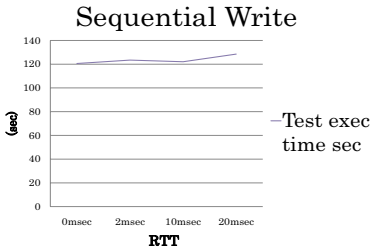

図 14 write 実行時間

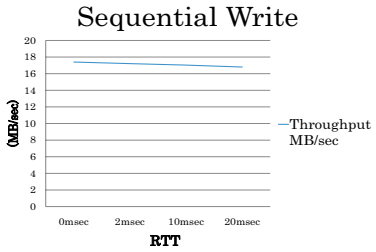

図 15 write スループット

これに対して read ではスループットが大きく低下した (図 17)．ファイルの書込はバッファを介して行われるため遅延分の差異が出づらい形となったのに対して，read 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．また read の実行時間の測定結果では，(i) スループットが低下しているにも関わらず値が一定である，(ii)RTT=20msec において大きな実行時間の減少が見られる，などの振舞が見られ，これらの動きについて後述のパケット解析で詳しく触れる (図 16)．

5.3 MapReduce 性能測定

高遅延環境において基本性能測定と同様のプログラムを実行した結果，実行時間は RTT

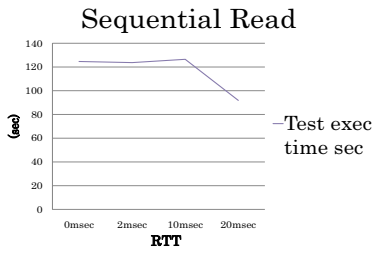

図 16 read 実行時間

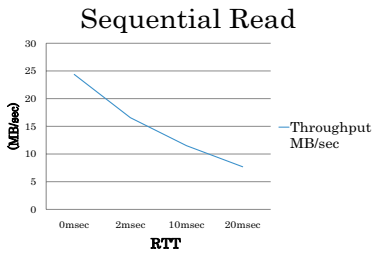

図 17 read スループット

の増加に伴い上昇した (図 18)．ソートプログラム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．

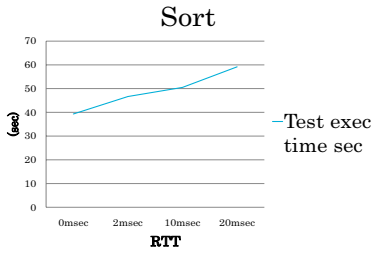

図 18 Sort 実行時間

5.4 高遅延環境における性能測定考察

HDFS 性能に関して，RTT の増加に伴って性能が劣化し，特に read 時におけるスループット低下が顕著であった．また，MapReduce 性能では，RandomWrite の実行時間にはほとんど変化がなかったのに対して，Sort の実行時間が RTT の増加に伴い上昇した．以上から HDFS，MapReduce どちらの性能についても，read アクセスにおいて特に遅延の影響が出やすいということが分かった．

6. パケット解析

6.1 実 験 概 要

前述の性能測定の結果から，HDFS においては各ノードにおける処理性能が低下しても，

クライアント側から見た処理性能は維持される場合が見られることが分かった．これらの事例に関してノード間を飛び交うパケットをキャプチャすることで，より詳細な解析を行った．

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 19)．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のみ抽出した．

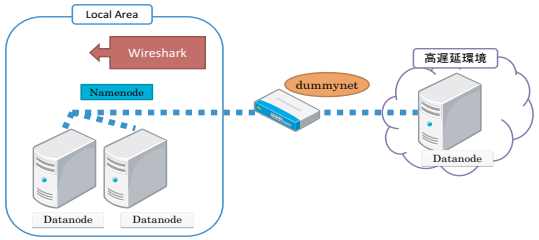

図 19　システム構成

6.2　高遅延環境におけるパケット解析

レプリカ数 1 においてシーケンシャルライト，シーケンシャルリードを MapReduce を通して実行するプログラムを交互に 5 回ずつ実行したとき各回のパケット通信量を調べた (図 20，21)．高遅延ノードの RTT=20msec としている．

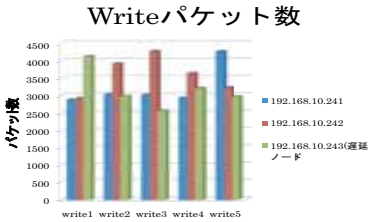

図 20　write 実行時のパケット数

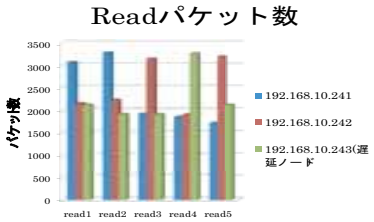

図 21　read 実行時のパケット数

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．ローカルエリアのノード，高遅延環境のノードの間に，Namenode からのアクセス頻度に差は見られない．

また，同様のプログラムを実行したとき，全ノードの合計パケット通信量をグラフにしたもののうち，RTT=0msec，10msec，20msec の結果を以下に示す (図 22，図 24)．

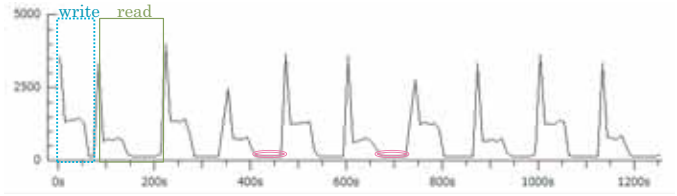

図 22　RTT=0msec におけるパケット IO グラフ

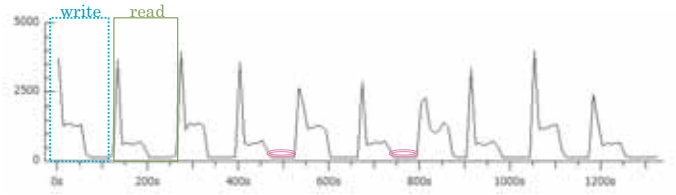

図 23　RTT=10msec におけるパケット IO グラフ

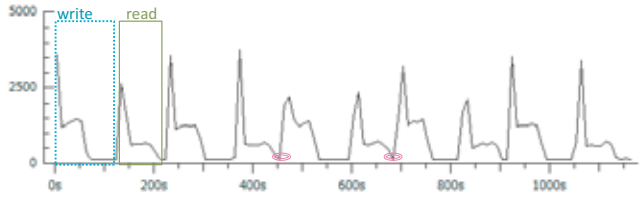

図 24　RTT=20msec におけるパケット IO グラフ

同条件の計測でも各回毎にやや振舞に揺らぎが見られるが，明確な相違点として各 read プロセスの終盤部におけるパケット通信が RTT=20msec の場合では極端に減少し，結果として処理時間全体は短縮されていることが分かる．このプロセスについてより詳細なグラフを示す (図 25，26)．

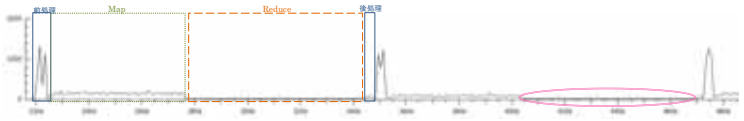

図 25　RTT=0msec におけるパケット IO グラフ詳細

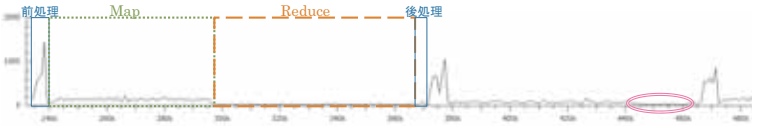

図 26　RTT=20msec におけるパケット IO グラフ詳細

ログファイルの検証から，処理終盤の微量なパケットのやり取りが行われているのは Reduce プロセス時であることが分かっている．このことと Read プログラム実行時 RTT=20msec において RTT=0msec よりもスループットが低下しているにも関わらず実行時間が減少するという性能測定での結果から (5 章参照)，RTT=20msec 時の処理においては Read プログラムにおける Map プロセスの生成結果が減少し，結果として Reduce プロセスの所要時間を減らしているものと見られる．例えば，read において Namenode からは分からないところである map プロセスがファイルの読み出しに失敗してタイムアウトとなり，結果として Reduce プロセスに返ってくる処理が減少したなどの可能性が考えられる．

パケットの振舞とファイルシステムにおける処理との詳細な相関関係については，分析中である．

7. まとめと今後の課題

7.1 ま と め

Rocks をインストールしたクラスタ上で Hadoop 環境を構築し初めに基本性能を測定した．HDFS の性能については read，write ともに台数に反比例して処理時間が減 少していくことが確認された．また MapReduce 性能についても台数を増やすと処理時間が減少することが 確認された．続いて人工遅延装置 Dummynet を導入し，広域環境を模した高遅延環境下で測定を行った．その結果 HDFS 性能，MapReduce 性能ともに，特に read 処理において遅延の影響が顕著であることが分かった．

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 read 処理における実行時間の矛盾に関して read プログラムの Reduce プロセスにおける大きな時間短縮が原因の一つであることが分かった．

7.2 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 Hadoop の振舞を調べ，結果から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”, 2003 Google
2) Hadoop:http://hadoop.apache.org/
3) Rocks Cluster:http://www.rocksclusters.org/
4) Wireshark:http://www.wireshark.org/