広域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の影響に関する一検討

百瀬　明日香 †　　小口 正人 †

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（以下 DFS）に注目し，特に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対応できるような信頼性の高い DFS 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解析した．

A Study about the effects of Remote Data Access for Hadoop Distributed File System over a Wide Area

Asuka Momose†　　Masato Oguchi†

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. 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.

1. はじめに

情報爆発の現代において大きな課題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などの問題に対して，汎用のハードウェアを用いて高度な集約処理を行う DFS に注目が集まっている．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 DFS 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．

1.1 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 はファイルのメタデータを保持する Master と実際にデータが格納される ChunkServer というノードからなり，ファイルを分散・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実現している．

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

† お茶の水女子大学
Ochanomizu University

2. 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に Hadoop-0.18.3 をインストールし，1 台を Hadoop Namenode，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．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．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．

3. 高遅延環境における性能測定

3.1 実験概要

高遅延接続を含む広域分散環境を想定して，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 1）．分散されるファイルのレプリカ数は 1〜3 とした．

3.2 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 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，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 まで変化させたときの各々のスループットを測定した．レプリカ数 1 の場合を比較すると，ライトではバッファが遅延を吸収しスループットがほとんど変化しないのに対し，リードではレプ

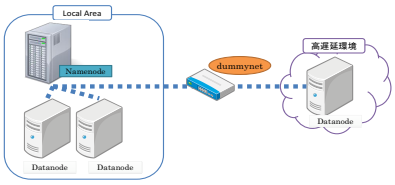

図 1: システム構成

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ていることが分かる（図 2，図 3）．

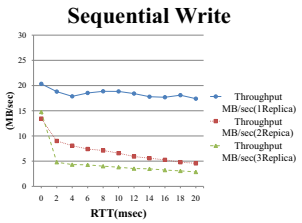

図 2: Write スループット

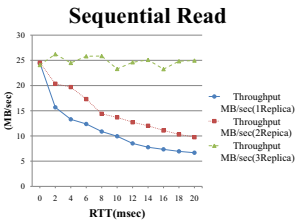

図 3: Read スループット

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することが確認された．

4. 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，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り，HDFS クラスタ内の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ることが可能である．

4.1 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．ローカルクラスタの Datanode2 台に/default/rack0，遠隔ノードの Datanode に/distant/node/rack1 というラック ID を付与し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 1）．これは Hadoop の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るものと見られ，これによって単純なラック設定では単一の遠隔ノードへのアクセスはむしろ増加することが分かる．

表 1: 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

4.2 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

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 2）．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．

表 2: 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 setting）を，ラック未設定の場合（default），ラックを単純に設定した場合（simple rack setting）と比較する（図 4，図 5）．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する度合いが大きいほど向上した結果になったと考えられる．

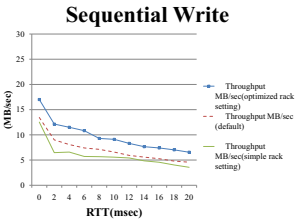

図 4: Write スループット

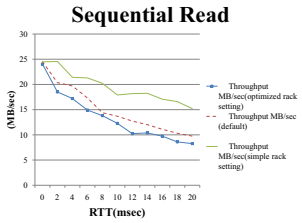

図 5: Read スループット

5. まとめと今後の課題

5.1 まとめ

広域分散環境を模した高遅延環境下で HDFS の性能測定を行ったところ，レプリカ数 1 の場合，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．HDFS のラック設定を適応させて性能測定を行うと，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，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．

5.2 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．

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/