クラスタ向け分散ファイルシステムの広域環境適用時の性能評価

百瀬 明日香　(指導教員：小口 正人)

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および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な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．本論文ではまずクラスタ環境の構築を行い，DFS のクラスタ環境における基本的な性能評価として，ファイルシステム I/O と Hadoop MapReduce の性能を測定した．また高遅延環境で Hadoop を実装し，同様の性能測定を行った．

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 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維持している (図 1)．

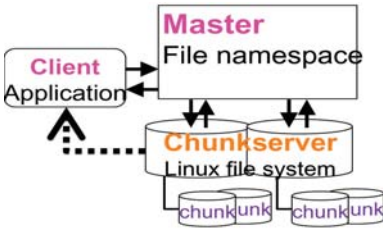

図 1: GFS アーキテクチャ

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) 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，HDFS，Hadoop MapReduce, hBase から構成される．本研究ではこのうちの HDFS と Hadoop MapReduce を実装したものを実験環境として使用した (図 2)．

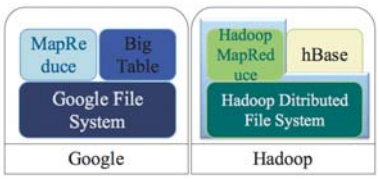

図 2: Google，Hadoop システム相関図

2.3　MapReduce

GFS の主要技術の一つが MapReduce アルゴリズムである．MapReduce は GFS と組み合わせて利用される分散処理技術で，ファイルシステム上で動かすプログラムを作成する際 Map，Reduce という二種類の関数を大枠として用意することで，開発者が分散処理の複雑なアルゴリズムを考慮せずアプリケーションを作成できる分散処理フレームワークである．

まず Map プロセスにおいて入力データであるキーと値のペアを受け取り，新しいキーと値のペアからなるリストを作る. 続いて Reduce プロセスにおいて Map によって作られたデータのうち同じキーを持つ複数の値を受け取り，それらを統合した値を生成する (図 3)．

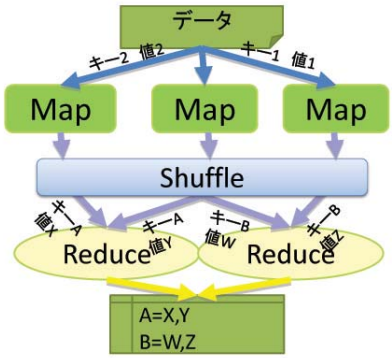

図 3: MapReduce アルゴリズム

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-0.18.3 を導入した．マシンスペックは CPU:Quad-Core Intel(R)Xeon(TM)3.60GHz，メインメモリ:2.0GB,OS:Linux2.6.9-55.0.2.ELsmp(CentOS 4.5) である．

4　基本性能測定

4.1　実験概要

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，すなわち複製は作らない形とした．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．

4.2　HDFS 性能測定

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．Write，Read ともにノード数の増加に合わせてプログラム実行時間が減少することが確認された (図 4, 図 5)．これによりアプリケーションから見たファイルシステム全体の仮想的なスループットはノード数に比例して上昇していることが分かる．

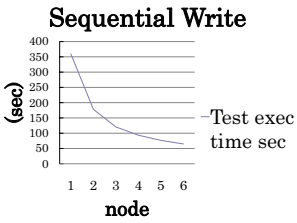

図 4: write 実行時間

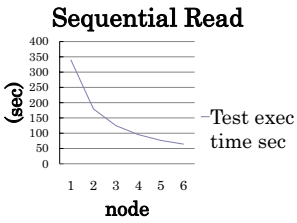

図 5: read 実行時間

4.3 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．まず RandomWriter 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．プログラム実行時間はノード数の増加に合わせて減少した (図 6)．最後に RandomWriter 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．こちらでも実行時間が台数に合わせて減少し，ノード数の増加に比例して性能向上することが確認された (図 7)．

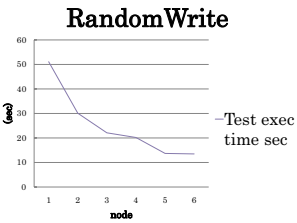

図 6: RandomWrite 実行時間

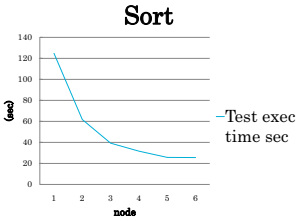

図 7: Sort 実行時間

5 高遅延環境での測定

5.1 実験概要

Namenode 兼 Datanode を 1 台，Datanode を 2 台使用し，そのうちの 1 台を人工遅延装置 dummynet を介して接続した (図 8)．分散されるファイルのレプリカ数は 1 とした．

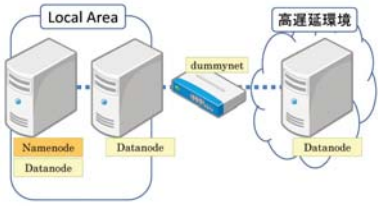

図 8: システム構成

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の往復遅延時間 (以下 RTT) を 0msec → 20msec と変化させたときの各々のスループットを測定した．write のスループットは若干低下し，read のスループットが大きく低下した (図 9，10)．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．

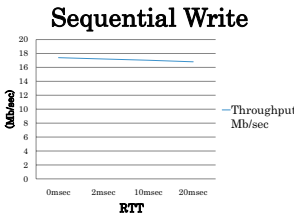

図 9: write スループット

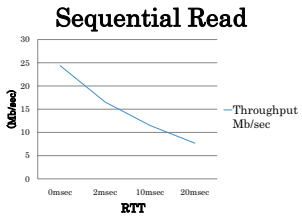

図 10: read 実行時間スループット

5.3 MapReduce 性能測定

基本性能測定と同様のプログラムを実行した．RandomWrite の実行時間は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 11, 12)．HDFS 性能と同様に，write 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．

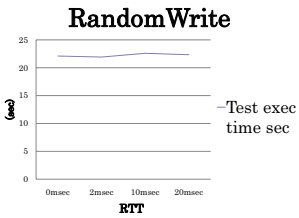

図 11: RandomWrite 実行時間

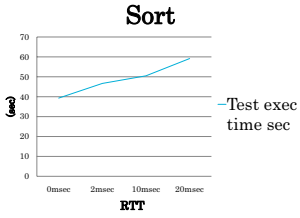

図 12: Sort 実行時間

6 まとめと今後の課題

6.1 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．同じく 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．また高遅延環境で Hadoop を実装し，性能測定を行った．HDFS 性能・MapReduce 性能ともに，write では遅延がバッファに吸収され性能にほとんど変化が見られないのに対して,read アクセス時においてはスループットが大きく低下し，遅延の影響が顕著であることが分かった．

6.2 今後の課題

高遅延環境を含む環境での Hadoop の振舞について，ファイルのロケーションとスループットの関わりを調べる．具体的にはどのようなタイミングで高遅延への書き込みが行われるのか，またリードにおいては高遅延へのアクセスを制御することでスループットにどのような影響があるか測定する．また結果から，性能向上のための手法を検討する．

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] 百瀬明日香，小口正人:“ 分散ファイルシステム Hadoop の広域環境への適用 ”, DEIM2010, 2010 年 3 月発表予定