D-032

MapReduce による大規模データの解析環境の構築と評価

Evaluation of Large-scale Data Analysis by MapReduce

村上 卓† 和泉 信生†
Suguru Murakami Shinobu Izumi

1. はじめに

近年，情報システムは独立した計算機で動くものからネットワークを介して提供されるものへと変化している． これに伴いブログや SNS，YouTube，Twitter などのソーシャルメディアが出現し，インターネットには大規模なデータが蓄積され続けている．そこで大規模なデータを効率良く解析することが求められている．しかし，大規模データは読み込むだけでも時間がかかり，1 台のマシンではリソースが足りず扱うことが難しい．従来の解決方法では高性能なコンピュータや，大量のコンピュータを導入し並列，分散処理を行うことで解決していた．高性能なマシンは高価なため容易に購入することはできず，また並列プログラミング[1] は非常に困難な作業である．これらの問題を安価で簡単に解決するためにクラウドコンピューティング（以下，クラウド）が注目されている．

「blogeye[2]」というサイトではクラウドサービスを使い 1 日に 50 万から 70 万ものブログ記事をリアルタイムに分析し，流行語を探るサービスを行っている．また，Google はバックグラウンドにクラウドを実装し，ペタバイトサイズの巨大なデータを MapReduce[3]というモデルを用いて検索エンジンのインデックスデータの作成やアクセスログの統計解析を分散並列処理で行っている． クラウドは様々な形態で提供され，雲のように形を変えながら現在のインターネットサービスに溶け込んでいる．

本研究では MapReduce モデルの実装の一つである Hadoop[4]を用いて大規模データの解析環境を作りその効果について調査を行った．これまでに大規模な環境での報告はされていたが，小規模な環境で有効性を確認するものはなかった．このため，1 から 5 台のコンピュータという小規模な環境で並列分散処理の時間を比較した．この結果，台数の増加によって処理時間が減少することを確認できた．3 台のコンピュータを用いた木村の研究[5]ではパフォーマンスの向上が確認できていなかったが，今回の結果から比較的小規模な環境による並列分散処理は処理時間の短縮に有効であることがわかった．

本論文の構成は次の通りである．第 2 章にクラウドコンピューティングとは何かを述べ，それに伴いクラウドの機能を実装するための Hadoop と MapReduce モデルを説明する．第 3 章に実験準備と MapReduce の実装について述べる．第 4 章に今回の実験結果を，第 5 章で考察を行う．第 6 章でまとめと今後の課題について述べる．

2．研究背景

2.1 クラウドコンピューティング

クラウドコンピューティングとは，データセンター内にある資源をインターネットを介してサービスを提供するものである．ユーザーはネットワークの向こう側にどのような実装がされているか意識することなく，サービスを利用できる（図 1）．

†崇城大学大学院 電気・電子工学専攻

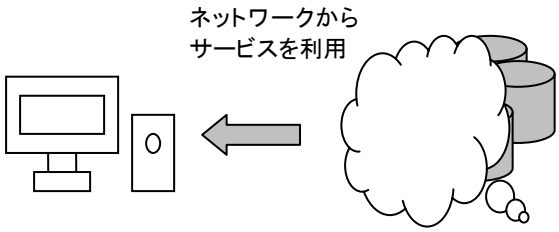

図 1 クラウドコンピューティングの概念

クラウドはサービスの内容で主に以下の 3 種類に分けられる．

- Software as a Service (SaaS)
  Web 中心の Web アプリケーション機能を提供する．Microsoft Online Services，Google Apps など．
- Platform as a Service (PaaS)
  サーバーの OS や BIOS などの環境を提供する．Google App Engine，Windows Azure など．
- Infrastructure as a Service (IaaS)
  ディスクストレージ等のインフラの提供をする．Amazon EC2，Amazon S3 など．

また，クラウドサービス間を連携，補助するサービスも誕生している．

クラウドのサービスでは SaaS とユーティリティコンピューティングを合わせたものがある．データセンターのハードウェアとソフトウェアのことをクラウド，そのサービスに対し”pay-as-you-go”（使っただけ払う）で課金するものをユーティリティコンピューティングと言う．クラウドを利用する利点として，ユーザーはコンピューティングの環境制限を考える必要はなく，利用するクラウドがどのような環境を構築しているか気にすることなく資源を得られる．次に，ユーザーは前もって何かを準備する必要がなく，必要に応じて環境を大きくしていくことができる．最後に短期で開始し，またいつでも終了できることがあげられる．

2.2 MapReduce モデル

Hadoop で用いられている MapReduce モデルおよび，MapReduce プログラミングについて述べる．MapReduce とは Google が開発した大規模データを分散処理するためのフレームワークである．MapReduce を利用したプログラミングでは，Map クラスと Reduce クラスを実装することで大規模な分散処理を行うことができる．MapReduce は次の 3 つの流れで処理される．

- Map
  入力データを受け取り，フィルタリングして情報を抽出する．
- Shuffle
  Map によって作られたデータを整理し，任意の順に並べ替える．
- Reduce
  データをまとめあげ，整理された処理結果を出力する．

2.3 Hadoop

Hadoop は，Apache Software Foundation のプロジェクトの一つであり，分散コンピューティングに関連するサブプロジェクトの集合体である（表 1）．Google の基盤ソフトウェアを基に作られたオープンソースソフトウェアで，分散処理のフレームワークである Haoop MapReduce，分散ファイルシステムの Hadoop Distributed File System（HDFS）を今研究では用いる．

表 1 Hadoop のサブプロジェクト[6]

Pig	Chukwa	Hive	HBase
MapReduce	HDFS		ZooKeeper
Core		Avro	

Hadoop は Master と Slave で構成されている．Master は JobTracker と NameNode，Slave は TaskTracker と DataNode の４種類のデーモンを起動しなければならない（図 2）．NameNode と DataNode は HDFS の機能を受け持ち，JobTracker と TaskTracker は MapReduce 処理を受け持つ．Hadoop ではタスクの管理を Master に，仕事を Slave に割り振る形態で分散処理を実行する．

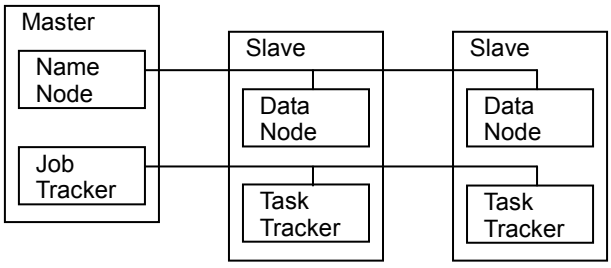

図 2 Master と Slave のデーモンの対応

3. 環境構築と実験

本章では，MapReduce を用いた大規模データ解析を行うための環境構築について述べる．また，分散処理の有効性を確認するための実験内容を述べる．

3.1 環境構築

Hadoop を使用するには，JavaTM 1.6.x（Sun 推奨）と SSH を導入する必要がある．本研究では上記の環境を Mac OS X 上で構築し，Hadoop はバージョン 0.20.1 を用いた．

分散処理を行うにあたり Master から Slave へのプロセスを起動するために SSH でログインする必要がある．そのため，Master の公開鍵を Slave へ事前に配布する必要がある（図 3）．

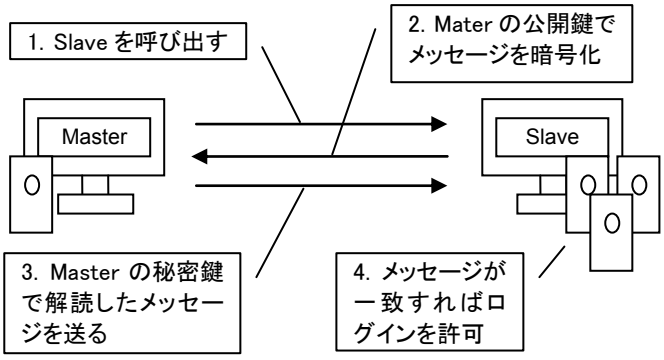

図 3 公開鍵暗号方式による SSH ログイン

また，ホストネームから IP アドレスを逆引きする必要があるので BIND[7]を用いて DNS サーバをローカルネットワークに構築した．

3.2 実験

本研究では， Hadoop を用いた分散処理を 1 台から 5 台のコンピュータで実行し処理時間の比較を行った（表 2）．1 台目を Master として順次 Slave に追加していった．対象データとして映画の平均評価とユーザーの平均評価を求める処理を行うプログラムを Hadoop の MapReduce モデルを用いて記述した．また，比較のため Hadoop を用いず単一のコンピュータで同様の結果を求めるプログラムを Java で作成し，処理時間を計測した．

表 2 コンピュータのスペック

	OS	CPU	Memory
1	Mac OS X 10.6.2	2.53 GHz Intel Core 2 Duo	4GB 1067 MHz DDR3
2	Mac OS X 10.6.2	2.53 GHz Intel Core 2 Duo	4GB 1067 MHz DDR3
3	Mac OS X 10.5.8	2.13 GHz Intel Core 2 Duo	4GB 800 MHz DDR2 SDRAM
4	Mac OS X 10.5.8	2.4 GHz Intel Core 2 Duo	4GB 667 MHz DDR2 SDRAM
5	Mac OS X 10.6.2	2.0 GHz Intel Core 2 Duo	2GB 1067 MHz DDR3

3.2.1 実験設定

実験では Master は Slave を兼用するものとした．また，台数が 1 台の場合は Hadoop の疑似分散モードを使用した． HDFS 上でのファイル複製は 2 に設定している．

3.2.2 実験データ

MovieLens[8]より提供されている映画評価データを使用した．これはオンライン映画推薦サービスである MovieLens のデータセットで，映画をユーザーが 0.5 単位で最高 5 までの評価をしたものである．データサイズは 980[MB]，評価データは次の形式でデータを保持している．

UserID :: MovieID :: Rate

UserID は映画を評価したユーザー，MovieID は評価された映画を示し，Rate は映画の評価を表したものである．入力データを表形式で示す（表 3）．

表 3　入力データ例

UserID	u-1	u-1	u-2	u-2	u-3	u-3
MovieID	m-103	m-102	m-101	m-102	m-101	m-102
Rate	2	3	4	1	2	5

3.3　MapReduce の実装

MapReduce の処理の流れをそれぞれのフェーズごとに述べる．表 2 のデータを使用している．

● Map

Map の処理を示す．Map は入力データを受け取り UserID と MovieID をそれぞれ Key に，Rate を Value として＜Key，Value＞の形で格納する．図 4 では UserID と MovieID を Key に，Rate を Value として格納している．

UserID	u-1	u-1	u-2	u-2	u-3	u-3
MovieID	m-103	m-102	m-101	m-102	m-101	m-102
Rate	2	3	4	1	2	5

⇩

UserID	u-1	u-1	u-2	u-2	u-3	u-3
Rate	2	3	4	1	2	5

MovieID	m-103	m-102	m-101	m-102	m-101	m-102
Rate	2	3	4	1	2	5

図 4　Map 処理の様子

● Shuffle

次に Suffle によって Key でソートされ，同一の Key をもつペア同士が隣り合うよう並べ替える．また，隣り合う同一 Key のペアを束ねて Reduce に渡す（図 5）．Shuffle は Map から Reduce にデータを渡す際，自動的に行われる．

UserID	u-1	u-1	u-2	u-2	u-3	u-3
Rate	2	3	4	1	2	5

MovieID	m-103	m-102	m-101	m-102	m-101	m-102
Rate	2	3	4	1	2	5

⇩

UserID	u-1	u-1	u-2	u-2	u-3	u-3
Rate	2	3	4	1	2	5

MovieID	m-101	m-101	m-102	m-102	m-102	m-103
Rate	4	2	3	1	5	2

⇩

UserID	u-1		u-2		u-3	
Rate	2	3	4	1	2	5

MovieID	m-101		m-102			m-103
Rate	4	2	3	1	5	2

図 5　Shuffle 処理の様子

● Reduce

最後に Reduce によって同一 Key の Value を足し合わせる（図 6）．この時，同一 Key の数をカウントして平均を求めた．＜UserID，平均 Rate＞，＜MovieID，平均 Rate＞の形で出力する．

UserID	u-1		u-2		u-3	
Rate	2	3	4	1	2	5

MovieID	m-101		m-102			m-103
Rate	4	2	3	1	5	2

⇩

UserID	u-1	u-2	u-3
Rate	5	5	7
Count	2	2	2

MovieID	m-101	m-102	m-103
Rate	6	9	2
Count	2	3	1

⇩

UserID	u-1	u-2	u-3
aveRate	2.5	2.5	3.5

MovieID	m-101	m-102	m-103
aveRate	3	3	2

図 6　Reduce 処理の様子

4．実験結果

図 7 に台数による処理時間の推移を示す．縦軸が処理時間，横軸が分散に使ったコンピュータの台数である．正確な処理時間を表 4 に示す．分散台数が増えることによって処理時間の減少が確認できた．また，単一のコンピュータの処理にかかった時間を表 5 に示す．

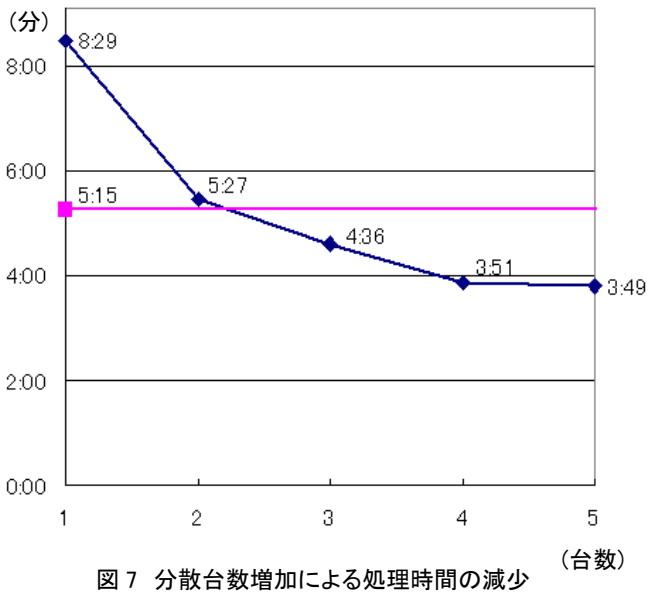

図 7　分散台数増加による処理時間の減少

表 4 分散台数ごとの処理時間

台数	1	2	3	4	5
時間(分)	8:29	5:27	4:36	3:51	3:49

表 5 単一コンピュータの処理時間

時間(分)	5:15

5．考察

実験結果から分散台数を増やしていくことで処理時間が減少することを確認できた．3 台の分散処理で単一コンピュータより速く処理を終えることができ，4 台，5 台で分散処理させた場合も処理時間が減少していることから環境の構築に成功したと言える．

木村の研究では Slave 数が増すほど処理のパフォーマンスが上がると期待されていたが，実験ではそこまでのパフォーマンスの向上を得ることができていなかった．原因として利用したコンピュータの台数が 3 台であることと，ファイルサイズが小さいことがあげられていた．しかし，今回の実験結果から 1GB 程度のファイルサイズであれば Slave 数の増加に伴ってパフォーマンスが上がることを確認できた．また，Hadoop を 1 台で処理させた場合，実メモリの大半をプロセスが使用する状態が発生した．これらのことより比較的少ない台数であっても MapReduce モデルを用いた処理を行う利点はあると言える．

Hadoop では分散処理のためのオーバーヘッドがあるため，分散処理を行わないプログラムに比べ，1 台の処理では大きく時間がかかった．しかし，2 台の分散からは処理時間の減少が確認できた．また，4 台，5 台の間で時間変化が小さいのはマシンの性能が不均一なためか，または今回のデータサイズが十分に大きくなかったなどの原因が考えられる．

6．まとめ

本研究では， Hadoop の MapReduce モデルを用いて大規模データの解析環境を構築した．また，この環境を利用して分散処理を行い有効性の調査を行った．正常に分散処理を行うことができ，大規模なデータを効率的に処理できることが示せた．

今後の課題として，より大規模な環境構築を行い，実用的なデータの解析が行えるアルゴリズムを MapReduce に実装していくことがあげられる．

7．参考文献

[1]Message Passing Interface Forum
http://www.mpi-forum.org/
[2]blogeye.jp
http://blogeye.jp/
[3]MapReduce: Simplified Data Processing on Large Clusters
http://labs.google.com/papers/mapreduce.html
[4]Welcome to Apache Hadoop!
http://hadoop.apache.org/
[5]木村昌史，大学教育系ネットワークにおけるクラウドコンピューティング，情報科学研究，第 18 号
[6]Tom White，Hadoop: The Definitive Guide，O’REILLY
[7]Internet Systems Consortium
http://www.isc.org/software/bind
[8]movielens
http://www.movielens.org/
[9]Jason Venner，Pro Hadoop，Apress
[10]MapReduce: Simplified Data Processing on Large Clusters
http://labs.google.com/papers/mapreduce.html
[11]エヌ・ティ・ティ レゾナント株式会社，株式会社 Preferred Infrastructure，Hadoop 調査報告書
[12]丸山不二夫，首藤一幸 他，雲の世界の向こうをつかむ クラウドの技術