PC クラスタ上のスケジューラを利用した MapReduce の実装

山下 尊也

1　はじめに

MapReduce はクラスタなどの並列計算機上で，巨大なデータセットに対し分散並列処理を行うのを支援する目的で，Google によって考案されたソフトウェアフレームワークである [1]．MPI などにも Map や Reduce 関数は定義されているが，これらのわかりやすい関数をフレームワーク化したことが特徴である．すなわち，ユーザはクラスタなどの並列計算機や並列の処理を意識することなくアルゴリズムを検討し，Map 関数と Reduce 関数を定義することで，並列処理を効果的に利用できる．言い換えれば，アルゴリズムを有しているユーザや，アルゴリズムを検討している研究者は，自身のアルゴリズムを Map 関数と Reduce 関数とに定義することができれば，並列処理をたやすく行うことが可能である．そのため，MapReduce の利用は急速に拡大している [2]．

Hadoop は Google の MapReduce および Google File System(GFS) のクローンであり，Java で実装されたフレームワークである [3]．この Hadoop を利用することで，さらに MapReduce を使った分散処理は容易となる．例えば，Amazon.com が開発者向けに提供するクラウドコンピューティング環境 Amazon EC2(Amazon Elastic Compute Cloud) では，この EC2 向け Hadoop ベースの MapReduce がリリースされており [4]，ユーザは Map 関数と Reduce 関数を定義することで Amazon のクラウドコンピューティング環境の資源が利用できることとなる．

このような背景の下，MapReduce に関連した研究も増えてきている．例えば，谷村らは RDF-database のバックエンドに分散ストレージと MapReduce フレームワークを用いた並列データ処理を利用することで，膨大なデータに対する多数の問い合わせに対応したシステムの構築を試みている [5]．

このように，MapReduce フレームワークは，このパターンに合致するアルゴリズムに対しては非常に協力であり，Hadoop を利用することで，簡単にクラスタを利用することも可能である．しかしながら，複数ユーザで利用されているような公共並列計算機では，通常，ジョブの投入はジョブスケジューラを利用するのが通常である．一方で，Hadoop では，ジョブスケジューラを利用する構成となっていない．特に，Windows Server をベースとした，Windows HPC Server を OS とする PC クラスタでは，必ずジョブスケジューラを利用する構成となっているため，Hadoop を用いた MapReduce による分散処理を行うことができない．

これらの問題を解決するために，本研究では，PC クラスタにおいて，ジョブスケジューラを利用する MapReduce フレームワークの実装について提案する．特に，Windows HPC Server を OS とする Windows クラスタを対象とする．基礎的な実装を用意し，簡単な数値実験を行うことで，システムの性能を評価する．

2　MapReduce

MapReduce は，Google の技術者である Jeffrey Dean と Sanjay Ghemawat により提案された，大規模 PC クラスタ上での分散プログラミングのモデルである．

2.1　Map と Reduce

Google では，Map と Reduce という 2 つのプロセスを組み合わせ，大量のデータ処理を PC クラスタ上で行っている [6]．以下では，Map と Reduce のプロセスについて説明する．

2.1.1　Map

Map とは，ひとまとまりのデータから新しい有用なデータへと変換するプロセスである．Fig. 1 に示すように Map では元のデータである A を A’ に変換する．この際，Map は Reduce での処理を考え，< キー，値 > という形で出力する．

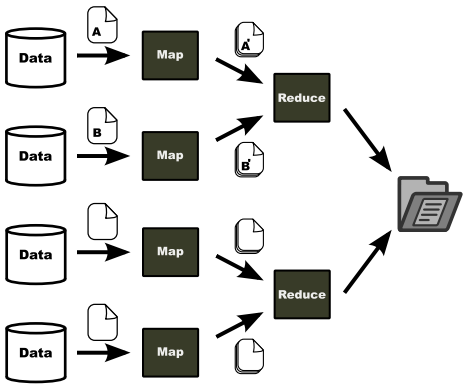

Fig.1　MapReduce の処理

2.1.2　Reduce

Fig. 1 に示すように Reduce では，A’ と B’ の情報からキーを用いて値を最終的に変換したいデータへと変換する．

2.2　Hadoop

MapReduce を実装したオープンソースのシステムとして Apache Hadoop[7]（以下, Hadoop）がある．Hadoop では，Table 1 に示すように，Google による MapReduce に対応し，分散ファイルシステムや分散データベース，分散処理言語などが実装されている．

Table1　Google MapReduce と Hadoop の対応表

	Google MapReduce	Hadoop
分散ファイルシステム	GFS	HDFS
分散データベース	BigTable	HBase
分散処理言語	Sawzall	Pig

2.3　システムの運用例

従来の MapReduce システムでは，Fig. 2 に示すように MapReduce のシステムが各ユーザに対してジョブが平等に割り当てられて実行される．これは，MapReduce のシステムにジョブスケジューラがあり，ジョブを効率よく処理をするため，PC クラスタの各ノードに平等に処理を割り振るためである．

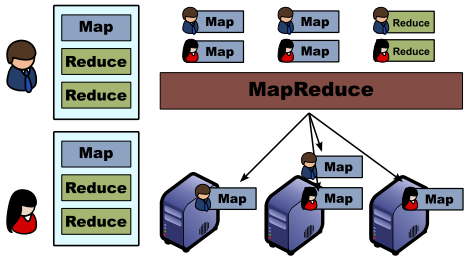

Fig.2　従来の MapReduce システム

しかし，PC クラスタの各ノードに平等に割り振られるため，ユーザに対する割り当てられるノード数を増やすなどの要求に対応することができない．そのため，柔軟にシステムを変えることができない．

3　提案システム

提案するシステムでは，Fig. 3 に示すように PC クラスタに付属するジョブスケジューラを利用することにより MapReduce を実装する．

まず，ユーザは Map と Reduce 用の実行ファイルを作成する．次にそれらの実行ファイルをジョブスケジューラに登録する．その際に，ジョブスケジューラの機能を用いることで，Map の処理がすべて終了した後に，Reduce の処理が開始されるように設定する．

Map では，Reduce に入力するための中間ファイルを生成する．PC クラスタのジョブスケジューラでは，入力可能なファイル数は限られているため，Map は Reduce に入力する前に生成した中間ファイルを 1 つに統合する工夫を行っている．

そして Reduce では，Map から送られてきた入力ファイルをもとにデータ処理を行い，ユーザが要求するデータを出力する．

提案システムの特徴としては，すべての処理はジョブスケジューラが制御しているため，ユーザの使用ノード数，使用時間の制限，ジョブの依存関係などの細かい制御が可能になる点である．

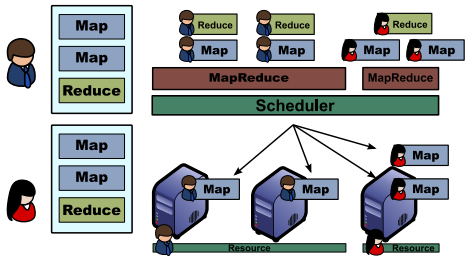

Fig.3　提案する MapReduce システム

4　数値実験

本実験では，従来の MapReduce システムである Hadoop のシステムと提案するジョブスケジューラを利用した MapReduce システムとを比較する．

4.1　実験内容

比較を行うプログラムとして，英文の中から英単語を抜き出し，英単語の出現回数をカウントするプログラムを作成した．対象とする英文は，Project Gutenberg[8] からダウンロードした 90 個の文章ファイルとする．これらの文章ファイルはデータサイズが小さく，PC クラスタでの処理時間が短く比較が難しい．そのため，文章ファイルの内容を 3 回繰り返し，1MB から 8MB の 90 個の文章ファイルを作成した．

これらの文章ファイルから，英単語を抜きだし，英単語の出現回数を数え終わるまでの時間を測定する．

4.2　実験環境

今回使用した機器を Table 2 に示す．ジョブスケジューラを利用した MapReduce システムとして，現在開発中である Microsoft Windows HPC Server V3（以下，HPC Server）を用いた．

ジョブスケジューラは，HPC Server に付属する HPC Job Manager を用い，PowerShell を用いてジョブの投入を行い，MapReduce と同様の処理を行った．

Table2　実験環境

CPU	Intel Xeon E5430 2.66GHz（Quad Core）
Memory	8192MB
HDD	300GB
Ethernet	1 Gigabit Ethernet
ノード数	24
使用プログラミング言語	C#

4.3　結果と考察

実験結果を Fig. 4 に示す．横軸はノード数，縦軸は文章ファイルから英単語を数える処理が行われた時間である．また，速度向上率の結果を Fig. 5 に示す．横軸は，ノード数，縦軸はノード数 1 の処理時間をもとにした速度向上率である．それぞれの結果は 5 回測定を行い，その結果から処理が行われた時間について中央値を取って

扱っている．

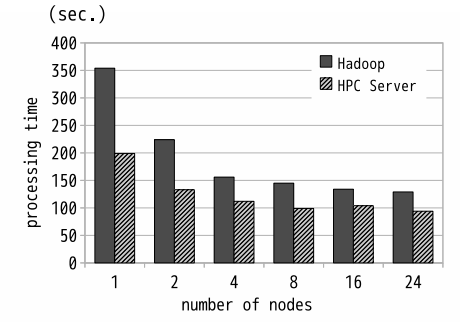

Fig.4　ノード数による処理時間の変化

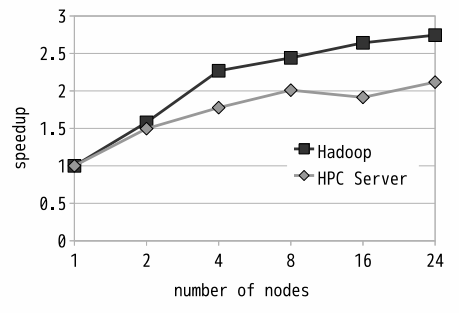

Fig.5　ノード数による速度向上率

Fig. 4 より, Hadoop のシステムと同様にノード数が増えると処理時間が短縮されているのが分かる．また HPC Server では Hadoop に比べ処理の実行時間が少ない．これは，Hadoop では分散ファイルシステムを用いているが，HPC Server では 1 つのノードに対してファイルを書き込む仕様になっている．そのため，HPC Server にはファイルを分散するプロセスがないため，処理時間が短かったと考えられる．

一方，Fig. 5 より今回の対象問題の場合では，ノード数に対し速度向上率があまり向上していない．対象問題では，文章ファイルのサイズに工夫したが，さらに文章ファイルのサイズやファイル数を見直す必要があると考えられる．また，どの部分にボトルネックがあるのかについても詳細に検討が必要である．さらに，提案システムにおいては，Windows HPC Server の提供するジョブスケジューラの提供する API の積極的な利用が必要であろう．

また，提案システムが優れている点として，下記のような使い勝手の良さがあげられる．すなわち，提案システムでは各ノードの使用数などはジョブスケジューラを用いて調整できるのに対し，Hadoop ではノード数を調整するためにはシステムを構築しなおす必要があった．

5　まとめと今後の課題

MapReduce はクラスタなどの並列計算機上で，巨大なデータセットに対し分散並列処理を行うのを支援する目的で，Google によって考案されたソフトウェアフレームワークである．この実装として Hadoop を始めとして種々提案されており，広く利用されている．しかしながら，共用計算機などの場合では，ジョブスケジューラを利用してジョブを投入することが求められるため，これらの実装を利用することができない．そのため，本研究では，PC クラスタのジョブスケジューラを利用した MapReduce のシステムを提案した．提案システムを Hadoop との比較を行った結果，提案するシステムは Hadoop と同様に分散処理可能であり，かつ，資源の利用を細かく決めることが可能であることが確認できた．

一方で，並列化効率や速度向上率は非常に悪く大きな性能向上が望まれる．そのため，今後は，PC クラスタのジョブスケジューラと MapReduce での処理について調査し，ジョブスケジューラの提供する API を効率よく利用することで，より性能の高い MapReduce の実装を提案する．

参考文献

1) MapReduce - Wikipedia. http://en.wikipedia.org/wiki/MapReduce.
2) Google の MapReduce アルゴリズムを Java で理解する @it. http://www.atmarkit.co.jp/fjava/special/distributed01/distributed01_1.html.
3) MapReduce programming with Apache Hadoop - JavaWorld. http://www.javaworld.com/javaworld/jw-09-2008/jw-09-hadoop.html.
4) Amazon Elastic MapReduce. http://aws.amazon.com/elasticmapreduce/.
5) 谷村勇輔，的野晃整，小島　功，田中良夫，関口智嗣：MapReduce における RDF-DB 処理に適したデータ分散格納方法の提案 (HPC-14:分散処理,2008 年並列/分散/協調処理に関する『佐賀』サマー・ワークショップ (SWoPP 佐賀 2008))，情報処理学会研究報告. [ハイパフォーマンスコンピューティング], Vol.2008, No.74, pp. 223–228 (20080729).
6) 西田圭介：Google を支える技術 巨大システムの内側の世界，技術評論社 (2008).
7) Welcome to Apache Hadoop Core! http://hadoop.apache.org/core/.
8) Main Page - Gutenberg. http://www.gutenberg.org/wiki/.