Vol.2009-ARC-186 No.6
Vol.2009-HPC-123 No.6
2009/11/30

スーパーコンピュータ TSUBAME 上での MapReduce の実現

佐藤 仁[†1] 小西 史一[†1] 山本 泰智[†2]
高木 利久[†2] 松岡 聡[†1,†3]

TSUBAME 上で Hadoop を実行するためのツール「Tsudoop」を開発した．Tsudoop は，既存システムの構成や運用方針の変更をすることなく，TSUBAME 上のジョブスケジューラである n1ge や Lustre ファイルシステムなどと協調して動作して Hadoop 実行環境を構築し，ユーザの MapReduce アプリケーションを実行する．予備実験として，このツールを用いて，生物医学系の学術論文を対象にした書籍情報データベースである MEDLINE に対してテキストの全文検索を行うアプリケーションを実行した．その結果，1 ノード (16 コア) での実行と 32 ノード (512 コア) での実行とを比較して 14 倍の性能向上を示し，TSUBAME のような高速な共有ファイルシステムやジョブスケジューラが存在するような計算環境でも，MapReduce アプリケーションの実行が可能なことを確認した．

MapReduce Implementation on the TSUBAME Supercomputer

Hitoshi Sato [†1] Fumikazu Konishi [†1]
Yasunori Yamamoto [†2] Toshihisa Takagi [†2]
and Satoshi Matsuoka [†1,†3]

We developed a program, called “Tsudoop”, which creates a MapReduce application execution environment on the TSUBAME supercomputer. Tsudoop enables application users to be able to run Hadoop-based MapReduce applications on TSUBAME without any modification in existing facilities and operation policies. We executed a MapReduce application, which conducts full-text search operations to a MEDLINE bibliographic database for life sciences and biomedical information, by using 32 nodes with 512 cores via Tsudoop and confirmed 14 times speedup compared with the execution by using a single node with 16 cores. We demonstrate an example of Hadoop deployment in a computing environment in cooperation with a high-speed shared filesystem and a batch job scheduling system.

1. はじめに

近年，情報技術の発達により，人類の取り扱うデータ量が爆発的に増加している．例えば，Google 社では 1 日に 20 ペタバイト以上のデータに対してクラスタ計算機を用いた並列分散処理を行っていると報告されている[1)]．科学技術計算の分野においてもこの傾向は当てはまり，高エネルギー物理学，生物学，天文学などの分野においても大規模なデータに対して並列分散処理を行う試みが広く行われている．

このような大規模データに対する並列分散処理を行うためのプログラミングモデルとして MapReduce[1)] が注目されている．MapReduce は，分散した key-value のペアデータに対して統一的な操作を並列で適用し，データへのアクセスの局所性を考慮したスケーラブルなデータ処理を実現する技術である．データ処理のプロセスを Map，Shuffle，Reduce の 3 つのフェーズに分解し，Map フェーズで入力データとなる key-value ペアから中間データとなる key-value ペアを生成し，Shuffle フェーズで同じ key に対して value のリストを生成し，Reduce フェーズで中間データを Shuffle することにより得られた key と value のリストから最終出力となる key-value のペアデータを生成するというものである．典型的な例では，ウェブのデータ解析などの処理に適用され，数千台規模の大規模なクラスタ計算機においてノード数に応じたスケーラビリティを得ている[1)2)]．また，様々な機械学習アルゴリズムへ MapReduce を適用し，有効であるという事例も報告されている[3)]．

東京工業大学学術国際情報センター (東工大 GSIC) では，2006 年からスーパーコンピュータ TSUBAME を運用しているが，年々，ユーザから大規模データ処理への要望が高まっており，その中で MapReduce 処理を実行したいというものも挙がっている．そのため，我々は，既存の MapReduce システムの実装 の中で最も普及している Hadoop[4)] の TSUBAME への適用の検討をはじめた．Hadoop での MapReduce 処理は，Hadoop Distributed File

†1 東京工業大学
Tokyo Institute of Technology
†2 ライフサイエンス統合データベースセンター
Database Center for Life Science
†3 国立情報学研究所
National Institute of Informatics

Vol.2009-ARC-186 No.6
Vol.2009-HPC-123 No.6
2009/11/30

System (HDFS) と呼ばれる共有ファイルシステムの上で Hadoop MapReduce と呼ばれる MapReduce 処理システムが動作することで行われる．HDFS は，各計算ノードのローカルストレージを束ねて一つの共有分散ファイルシステムを構成し，ストレージのシングルシステムイメージを提供する．Hadoop MapReduce は，典型的なマスタ・ワーカの構成をしており，JobTracker と呼ばれるマスタが MapReduce ジョブの受付やスケジューリングを担い，TaskTracker と呼ばれる複数のワーカが実際の Map・Reduce タスクを実行する．一方，TSUBAME は，計算ノードあたりのローカルディスクの容量が非常に少ないため使用することが現実的ではなく，その代わりに，高速な共有分散ファイルシステムである Lustre[5] が存在し，かつ，TSUBAME 上でのジョブスケジューリングを司るスケジューラ n1ge が存在するなど，Hadoop が前提としている状況とはいくつか異なる．また，既存のシステムの構成や運用方針の大きな改変は望ましくないという要請もある．このため，Hadoop をそのまま TSUBAME へ適用することは難しい．さらに，性能面の特性も明らかではない．

そこで，我々は，これらの問題を避け，TSUBAME 上で Hadoop を実行するためのツール「Tsudoop」を開発した．Tsudoop は，既存システムの構成や運用方針の変更をすることなく TSUBAME 上のジョブスケジューラである n1ge や Lustre ファイルシステムなどと協調して動作して Hadoop 実行環境を構築し，ユーザの MapReduce アプリケーションを実行する．予備実験として，このツールを用いて，生物医学系の学術論文を対象にした書籍情報データベースである MEDLINE に対してテキストの全文検索を行うアプリケーションを実行した．その結果，1 ノード (16 コア) での実行と 32 ノード (512 コア) での実行とを比較して 14 倍の性能向上を示し，TSUBAME のような高速な共有ファイルシステムやジョブスケジューラが存在するような計算環境でも，MapReduce アプリケーションの実行が可能なことを確認した．

2. Hadoop

Hadoop は，Apache Software Foundation にてオープンソースで開発が進められている，並列分散環境で MapReduce による大規模データ処理を行うためのソフトウェアコレクションである．Hadoop での MapReduce 処理は，共有分散ファイルシステムである HDFS 上で Hadoop MapReduce という MapReduce 処理システムが動作することで行われる．Google の MapReduce システムにインスパイアされているため類似した部分[1),6)] が多く含まれるが，ここでは構成要素の中心である HDFS と Hadoop MapReduse の概要について述べる．詳細については，Hadoop のホームページ[4)] を参照されたい．

2.1 HDFS

HDFS は，コモディティなハードウェアで構成されたクラスタ計算機上で大規模データを扱うことに特化した並列共有分散ファイルシステムである．典型的なマスタ・スレーブの構成をしており，NameNode と呼ばれるマスタがファイルシステムのディレクトリツリーやファイルのメタデータなどファイルシステムの名前空間に関する情報を管理し，DataNode と呼ばれる複数のスレーブが実際のファイルのデータを管理する．ファイルは 1 つ以上の一定のサイズ (デフォルトでは 64MB) の断片に分割され，各計算ノードのローカルストレージへ分散されて格納されるが，実際には統合した一つのイメージとしてみえる．また，データの耐故障性を実現するために，ファイル断片は異なる DataNode へ一定数複製される．

2.2 Hadoop MapReduce

Hadoop MapReduce は，HDFS 上で MapReduce 処理を行うためのフレームワークである．典型的なマスタ・ワーカの構成をしており，JobTracker と呼ばれるマスタと TaskTracker と呼ばれる複数のワーカからなる．MapReduce のプログラミングモデルでは，データ処理のプロセスを Map，Shuffle，Reduce の 3 つのフェーズに分解し，Map フェーズで入力データとなる key-value ペアから中間データとなる key-value ペアを生成し，Shuffle フェーズで同じ key に対して value のリストを生成し，Reduce フェーズで中間データを Shuffle することにより得られた key と value のリストから最終出力となる key-value のペアデータを生成する．JobTracker は，この一連の MapReduce ジョブの受付やスケジューリングを担い，TaskTracker は実際の Map 及び Reduce タスクを実行する．HDFS 上での利用を前提とし，タスクはできるだけデータの近くに割り当てられるため，局所性を生かし，ネットワークへのデータ転送量を極力抑えた効率的な I/O を実現する．

3. TSUBAME

東工大 GSIC では，2006 年からスーパーコンピュータ TSUBAME の運用を開始した．2009 年 10 月の時点では，ピーク性能 163TFlops，Linpack 性能 87TFlops，10,000 を超えるコア数，合計 20TB を超えるメインメモリ容量，合計 1.6PB のストレージ領域と，データ・インテンシブアプリケーションを実行するのに適した大規模計算システムである．ここでは，まず，Hadoop を実行する上で，TSUBAME の構成で関連の深い部分についての概要を述べ，その後，Hadoop を TSUBAME へ適用する際の問題点を示す．

3.1 構　　成

TSUBAME は，655 台の計算ノードがストレージサーバと InfiniBand(一部 Gigabit Eth-

Vol.2009-ARC-186 No.6
Vol.2009-HPC-123 No.6
2009/11/30

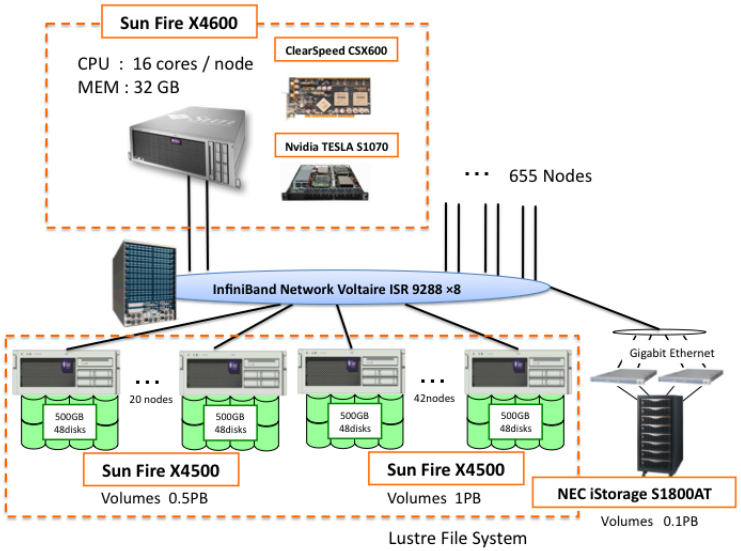

図 1 TSUBAME の構成図

ernet) ネットワークにより接続された構成をしている．TSUBAME 上でのジョブの実行はバッチジョブスケジューリングシステムを介して行う．図 1 に TSUBAME 全体の構成図を示す． 以下では，各構成要素の概要を述べる．

計算ノード

計算ノード (Sun Fire X4600) は，2.4 GHz Dual-Core AMD Opteron 880 を 8 個持ち，計 16 個の CPU コアが 32GB のメインメモリを共有する[*1]．ローカルストレージとして，32GB の HDD が 2 台存在し，RAID1 を構成しているが，主に，OS やシステムソフトウェアなどを格納するために用いられる．また，Host Channel Adapter(HCA) を 2 つ持ち，InfiniBand ネットワークを経由して，他の計算ノード及びストレージサーバへ接続される．さらに，一部の計算ノードは，ClearSpeed アクセラレータ (CSX600) 及び Nvidia TESLA アクセラレータ (S1070) へ接続されている．主要な I/O スロットは PCI-X 及び PCI-Express 1.0×8 であり，ここに HCA や各種アクセラレータが接続されている．OS は，64bit 対応の SUSE Linux Enterprise Server 10 SP 2 である．

ストレージサーバ

主に，NEC iStorage S1800ST と Sun Fire X4500 の 2 種類のストレージサーバが存在する．前者は，NFS により 0.1PB の home 領域を提供している．計算ノードへは Gigabit Ethernet を介して接続される．後者は，ストレージサーバ (Sun Fire X4500) 計 62 台から構成され，Lustre 並列共有分散ファイルシステムにより合計 1.5PB の work 領域を提供している．各ストレージサーバには HDD が 48 台接続され，24TB の領域を提供しており，計算ノードへは InfiniBand ネットワークを介して接続される．Lustre は，Sun を中心にオープンソースで開発が進められている並列分散ファイルシステムである．HDFS と同様に，典型的なマスタ・スレーブの構成をしており，Meta Data Server(MDS) と呼ばれるマスタがファイルシステムのディレクトリツリーやファイルのメタデータなどファイルシステムの名前空間に関する情報を管理し，Object Storage Server(OSS) と呼ばれるスレーブに接続された Object Storage Target(OST) が実際のファイルのデータを格納する．各ストレージサーバは，この OSS,OST に相当する．ファイルは 1 つ以上の一定サイズの断片に分割され，OST 上に分散されて格納される．TSUBAME のデフォルトの設定では，ファイルは 1 つの断片で構成され，分割サイズは 1MB となっている．

InfiniBand ネットワーク

各計算ノードからは 2 本，Lustre を構成する各ストレージサーバからは 1 本の 10Gbps SDR InfiniBand により，288 ポートの Voltaire ISR9288 スイッチに接続される．スイッチ間は，24 本の InfiniBand により接続される．

バッチジョブスケジューリングシステム

TSUBAME では，インタラクティブによるジョブの実行も可能であるが，他のジョブの影響を受けることなく，計算資源を大規模に効率良く利用するために，バッチジョブスケジューリングシステムを介したジョブの実行が推奨されている．ジョブスケジューラは，Sun N1 Grid Engine (N1GE) を基盤に，TSUBAME 用にカスタマイズした n1ge を用いている[*2]．n1ge は，基本的には，N1GE と同一であるが，TSUBAME の構成 (例えば，キューの構成，インストールされサポートしているアプリケーション，など) に沿ったコマンドラインオプションを提供していたり，ユーザの課金情報を収集していたり，といくつか点で異

⋆1 一部，これとは構成が異なるノードが含まれるが，大部分の構成はこのとおりである．

⋆2 本稿では，Sun N1 Grid Engine を大文字の N1GE と表記し，TSUBAME 用にカスタマイズしたものを小文字の n1ge と表記する．

Vol.2009-ARC-186 No.6
Vol.2009-HPC-123 No.6
2009/11/30

なる．

3.2 Hadoop を適用する際の問題点

TSUBAME 上で Hadoop を実行する場合，TSUBAME の構成が Hadoop が前提としている状況とはいくつか異なっており，また，既存システムの構成や運用方針の大きな改変は望ましくないため，Hadoop をそのまま適用することは難しい．具体的には，以下の点が問題になる．

- HDFS は計算ノードのローカルストレージを束ねて一つの共有分散ファイルシステムを構成するが，TSUBAME は計算ノードのローカルストレージの容量が非常に少ないため使用することが現実的ではなく，その代わりに，高速な共有分散ファイルシステム Lustre が存在する．
- Hadoop MapReduce では，JobTracker というスケジューラが MapReduce ジョブのスケジューリングを行うのに対して，TSUBAME はシステム上でのジョブスケジューリングを司るスケジューラ n1ge が存在する．
- HDFS の NameNode，DataNode，及び，Hadoop MapReduce の JobTracker，TaskTracker を起動する際，現在の実装では，計算ノードに直接 ssh を行いプロセスを起動しようとするが，TSUBAME では計算ノードに対して ssh を行うことは推奨されていない (禁止されている)．
- HDFS の NameNode，DataNode，及び，Hadoop MapReduce の JobTracker，TaskTracker は，現在の実装では，プロセスとして起動された後デーモンとなるが，これは，ジョブスケジューラの管理化を外れてゾンビジョブとして残る要因となるため推奨されていない (禁止されている)．

4. TSUBAME への Hadoop の適用

TSUBAME 上で Hadoop を実行するためには，3.2 節で述べた問題点を解決しなければならない．ここでは，まず，問題点を解決するための手法を述べ，その後，それらの解決手法を用いて TSUBAME 上で Hadoop 環境を構成し MapReduce を実行するために開発したツール「Tsudoop」の概要について述べる．

4.1 問題点の解決

Hadoop と n1ge を協調して動作させることに関しては，既に Hadoop と n1ge の基盤となっているオープンソース版の実装の Sun Grid Engine とを協調して動作させた事例[7] がある．我々の手法も，基本的方針としては，この事例を踏襲する．3.2 節で述べたの問題点を回避して，TSUBAME 上で Hadoop を実行するためには，具体的には，以下のことを行えばよい．

- HDFS を用いて計算ノードのローカルストレージを束ねた共有ファイルシステムを構成せず (すなわち，NameNode，DataNode のプロセスを起動せず)，MapReduce ジョブのデータの入出力は Lustre ファイルシステムに対して直接行う．
- MapReduce ジョブの実行環境はジョブスケジューラである n1ge を介して構成する．すなわち，MapReduce ジョブを実行する計算ノードをあらかじめジョブスケジューラである n1ge を介して確保する．
- HDFS を構成しないので，n1ge 経由で JobTracker と TaskTracker のプロセスだけを起動し，Hadoop MapReduce を構成する．この際，計算ノードに対して直接 ssh を行わずに起動する必要がある．
- 起動した JobTracker と TaskTracker のプロセスをデーモン化させない．

我々は，上記の事項を考慮して TSUBAME 上で Hadoop 環境を構成し，MapReduce を実行するためのツール「Tsudoop」を開発した．次節ではその概要について述べる．

4.2 Tsudoop

Tsudoop は，TSUBAME 上で Hadoop 環境を構成し，MapReduce を実行するためのツールである．図 2 に全体の Tsudoop の実行の流れを記す．

Tsudoop は，図 3 のように，コマンドラインから実行する．*script* には，ユーザが Hadoop 環境で実行したいコマンドを記述したシェルスクリプトを与える．例えば，WordCount.jar プログラムを実行したい場合は，図 4 ように記述する． *option* は，n1ge で提供されているオプション (投入するキュー　*-q*，課金グループ *-g*，実行時間 *-rt*，など) や，Hadoop の環境構成に関わるオプション (n1ge により何台の計算ノードを確保するか，1 台の計算ノードでいくつのコアを使うか，など) を受け付ける．

Tsuboop は，コマンドにより起動された後，まず，Hadoop 環境を構成する計算ノードの確保を行う．N1GE では，並列環境 (PE) を構成することで，計算ノードを同時に複数台確保することが可能であるが，現状の n1ge では，限られたアプリケーション (MPI, OpenMP, Gaussian Linda, GAMESS DDI) のための並列環境しかサポートされていないため，Tsudoop では，GXP[8] を用いて並列環境を確保している．GXP は，並列分散環境を効率的に利用するためのシェルである．基本的な機能としては，ssh などの遠隔ログインコマンドを用いて多数のノードに同時にログインを行いセッションを維持し，それらのノードに対して一度の操作で一斉にコマンドを送りプロセスを起動する，というものである．GXP では，

Vol.2009-ARC-186 No.6
Vol.2009-HPC-123 No.6
2009/11/30

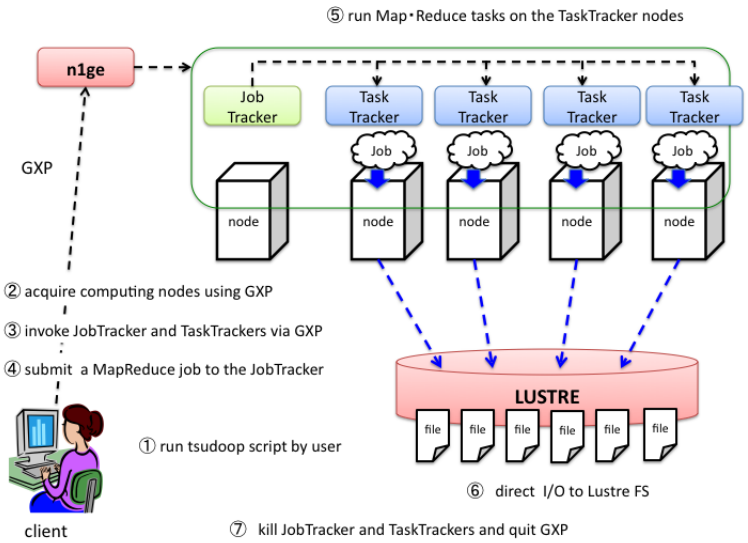

図 2 Tsudoop の実行の流れ

```
$ tsudoop ⟨option⟩ script
```

図 3 tsudoop コマンドの実行

```
#!/bin/sh

hadoop  jar  WordCount.jar  org.myorg.WordCount  /... input  /... output
```

図 4 シェルスクリプトファイル (*script*) の内容の例

バッチジョブスケジューリングで管理されたノードに対して，ジョブスケジューラ経由でノードを確保し，それらのノードに対して，ssh を用いずにソケット経由で，一度の操作で一斉にコマンドを送りプロセスを起動する，という機能がある．Tsudoop ではこの機能を用いている．

計算ノードを確保した後，これらに対して，JobTracker，TaskTracker のプロセスを起動する．N1GE では qrsh というジョブスケジューラ経由でコマンドを起動する機能があるが，n1ge ではサポートされていないため，GXP を用いて JobTracker，TaskTracker を起動する．hadoop コマンドに，jobtracker 及び tasktracker を引数に与えて，直接起動することで，デーモン化を避けることができる．

JobTracker，TaskTracker のプロセスを起動した後，tsudoop コマンドの引数として与えられた *script* を実行する．図 4 の例では，JobTracker に MapReduce ジョブ (WordCount.jar) を依頼し，TaskTracker が各計算ノードで Map・Task タスクを実行する．データの入出力は Lustre 上の/...input や/...output のディレクトリの中に行われる．

MapReduce ジョブの終了後，Tsudoop は，JobTracker，TaskTracker のプロセスを kill し，GXP を終了することで，構築した Hadoop 環境を終了する．

5. 予 備 実 験

TSUBAME へ Hadoop を適用した際の有効性を確認するために，簡単な予備実験を行った．以下では，その概要について述べる．

5.1 概　　要

Tsudoop を用いて，Hadoop を用いて実装された実際のユーザのアプリケーションを実行した．対象としたアプリケーションは，生物医学系の学術論文を対象にした書誌情報データベースである MEDLINE から単語の n-gram(n 個の連続する単語の集合) を取得しその頻度表を得る，というテキストの全文検索処理を MapReduce として実装したものある．Map フェーズで，MEDLINE データベース中のファイルから並列に n-gram の頻度を取得し，Reduce フェーズでそれらを集計する．データ処理の対象するファイルの数は 593 個，総サイズは 15GB，平均 24MB である．ファイルは全て Lustre 上に置かれ，4 つのファイル断片に分割する設定とした．Lustre の I/O 性能は，24MB のサイズのファイルを対象にした場合で，write で 180MB/s，read で 576MB/s であった．スケーラビリティを確認するために，アプリケーションは，ノード台数を 1〜32，ノードあたりのコア数を 1〜16，と変化させて実行した．Map タスクの数は，システム側で自動的に決定した数に従い 790 個程度とし，Reduce タスクの数は，文献[9] を参考にし，0.95 × (ノード数) × (ノードあたりに割り当てるコア数)，と決定した．Hadoop のバージョンは，0.20.0 を使用した．

5.2 実 験 結 果

図 5 に，ノード数に対する MapRedcue の実行時間の関係を表した結果を記す．x 軸は MapReduce ジョブが使用したノード台数を表し，y 軸は MapReduce ジョブが実行しはじ

Vol.2009-ARC-186 No.6
Vol.2009-HPC-123 No.6
2009/11/30

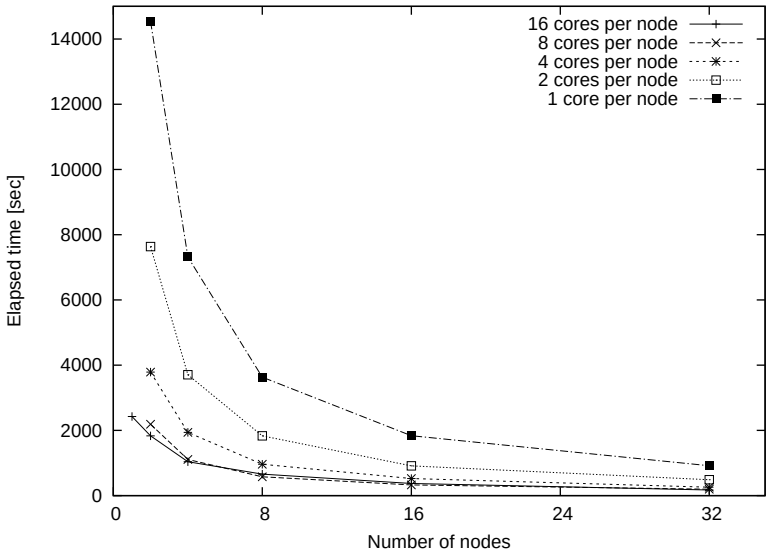

図 5 ノード数に対する MapReduce ジョブの実行時間

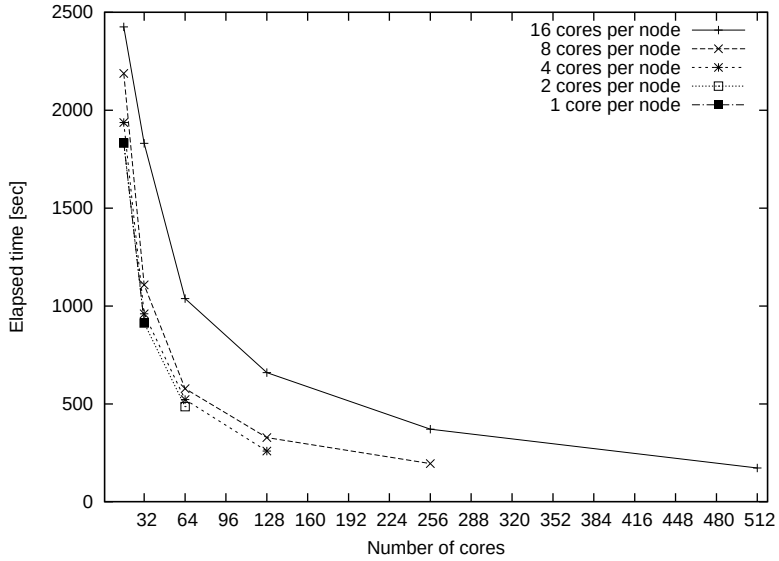

図 6 コア数に対する MapReduce ジョブの実行時間

めてから終了するまでの経過時間 (実行時間) を表す．ここでは，MapReduce ジョブが使用したノードあたりのコア数が 1〜16 のものを記している．ノードあたりのコア数を 16 としたとき，1 台のノードで実行したときの実行時間が 2426 秒であるのに対して，32 台のノードの場合の実行時間は 173 秒となり，14 倍の性能向上がみられた．また，この傾向は，ノードあたりのコア数の割り当て方によらずに同様であった．このことから，ノード台数の増加に応じてスケールすることが伺える．ノードあたりのコア数の割り当て方によって同じノード台数を用いたときの実行時間に差異がみられたが，これは，MapReduce ジョブが使用する総コア数が変わるためである．例えば，ノード台数が 16 であるときノードあたりのコア数を 1 と割り当てると全体で 16 コアを使用するが，コア数を 16 と割り当てると全体で 256 コア使用する．つまり，今回対象としたアプリケーションでは，同じノード台数であれば，コア数を多く使用した方が高い性能を示す．しかし，単にコア数を多くすれば，高い性能を示すわけではない．ノードあたりのコア数の割り当てを 8 としたものと 16 としたものでは，後者の方がコア数が多いものの，ほぼ同様の性能を示している．同様の性能を示すのであれば，できるだけ計算資源を使用しない方が運用上は望ましい．このため，MapReduce ジョブの実行環境の資源選択 (ノード数，ノードあたりのコア数などの決定) に課題があることがあることが伺える．

次に，コア数に対する MapReduce の実行時間の関係を表した結果を図 6 に記す．x 軸はコア数を表し，y 軸は MapReduce ジョブの実行時間を表す．ここでも，ノードあたりのコア数の割り当てを 1〜16 と変えたものを記している．図 6 より，同じコア数を使用するのであれば，ノードあたりのコア数の割り当て方を少なくした方が高い性能を示した．64 コアを用いて MapReduce ジョブを実行した場合を例にとると，ノードあたりのコア数を 2 と割り当てたときの実行時間は 486 秒であるのに対し，コア数を 16 と割り当てたときの実行時間は 1038 秒であり，2.1 倍の開きがあった．この主な原因は，今回の実験では，1 つのプロセスあたりに使用するノードのメインメモリのサイズを 4GB と設定して行っており，ノード上で競合が発生していたため，だと考えている．しかし，ノード上で競合が発生していないような，ノードあたりの割り当てたコア数が 1〜8 のときでも，実行時間の結果に若干の開きがみられるため，ファイルシステムレベルでの競合など他の要因もあると考えられるが，これらの詳細な解析は今後の課題である．

Vol.2009-ARC-186 No.6
Vol.2009-HPC-123 No.6
2009/11/30

6. おわりに

Hadoop を TSUBAME 上へ適用する際の問題点 (具体的には，ジョブスケジューラである n1ge や Lustre ファイルシステムとの協調など) を指摘し，それらの問題点を避けて TSUBAME 上で Hadoop を実行するためのツール「Tsudoop」について述べた．予備実験として，このツールを用いて，生物医学系の学術論文を対象にした書籍情報データベースである MEDLINE に対してテキストの全文検索を行うアプリケーションを実行した．その結果，1 ノード (16 コア) での実行と 32 ノード (512 コア) での実行とを比較して 14 倍の性能向上を示し，TSUBAME のような高速な共有ファイルシステムやジョブスケジューラが存在するような計算環境でも，MapReduce アプリケーションの実行が可能なことを確認した．

今後の課題としては，MapReduce ジョブの I/O 性能の詳細な解析や MapReduce ジョブ実行環境の資源選択手法の開発，システムやアプリケーションの性能チューニング，などをはじめとして，さらに大規模なデータを用いたアプリケーションの実行や他の異なるアプリケーションへの適用，などが挙げられる．

謝辞 本研究の一部は科学研究費補助金特定領域研究 (18049028) と GCOE プログラム「計算世界観の深化と展開」の補助による．

参考文献

1) Dean,J. and Ghemawat,S.: MapReduce: simplified data processing on large clusters, *Communications of the ACM*, Vol.51, No.1, pp. 107 – 113 (2008).
2) Scaling Hadoop to 4000 nodes at Yahoo!, `http://developer.yahoo.net/blogs/hadoop/2008/09/scaling_hadoop_to_4000_nodes_a.html` (2008).
3) Chu,C.-T., Kim,S.K., Lin,Y.-A., Yu,Y.Y., Bradski,G., Ng,A.Y. and Olukotun,K.: Map-Reduce for Machine Learning on Multicore, in *Advances in Neural Information Processing System 19*, pp. 281 – 288 (2006).
4) Hadoop, `http://hadoop.apache.org`.
5) Lustre, `http://www.lustre.org`.
6) Ghemawat,S., Gobioff,H. and Leung,S.-T.: The Google File System, in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 96–108, Bolton Landing, NewYork (2003).
7) Integrating Hadoop into SGE, `https://rc.usf.edu/trac/hadoop/wiki/SGEIntegration`.
8) GXP parallel/distributed shell, `http://sourceforge.net/projects/gxp/`.
9) Partitioning your job into maps and reduces, `http://wiki.apache.org/hadoop/HowManyMapsAndReduces`.