Vol.2012-HPC-136 No.20
2012/10/4

マルチコアCPU環境における仮想計算機を用いたHadoopシステムの評価

石井 朝葉[1,a)]　金 鎔煥[1]　中村 純哉[1]　大下 福仁[1]　角川 裕次[1]　増澤 利光[1]

概要：Hadoopとは，大規模データの並列分散処理フレームワークである．Hadoopシステムは，大規模データを効率的かつ安全に保持するだけではなく，複数の計算機が効率的に並列処理を行うことで，膨大なデータの高速処理を実現している．近年，仮想化技術が注目を浴びており，Hadoopシステムが仮想計算機で構成される場合が想定される．仮想計算機によるHadoopシステムの運用は，拡張性，耐故障性，運用コストの面で利点を持つ．その一方で，仮想計算機を管理するための仮想マシンモニタがリソースを必要とするため，計算機本来のリソースを仮想計算機が使いきることはできず，性能低下が懸念される．本報告では，仮想計算機から構成されるHadoopシステムと実際の物理マシンから構成されるHadoopシステムとの比較実験を行い，性能の評価・分析を行った．比較実験の結果として，仮想計算機を用いたHadoopシステムは，小さいファイルを大量に扱う場合に有利であること，仮想計算機のリソースで扱える程度の負荷の処理では物理マシンとの処理性能の差が少ないことを確認した．

キーワード：Hadoop，HDFS，MapReduce，マルチコアCPU，仮想計算機

Evaluation of Hadoop system consisting of Virtual Machines on Multi-core CPUs

Abstract: Hadoop is a framework to allow the parallel distributed processing of large data sets. Hadoop provides not only the logical storage system which keeps large data set safe and efficient, but also a software framework for distributed processing of large data sets efficiently on the computer cluster. Recently, virtualization technology is emerging, and actually lots of virtual servers are operating due to many advantages of virtualization. For this reason, it can be assumed that Hadoop system is configured and operated on virtual machines. Virtualized Hadoop system has a lot of advantages, for example, saving operating and/or spatial cost, scalability, fault-tolerance, and so on. However, a virtual machine monitor is required to apply virtualization technology and surely a virtual machine monitor spends a portion of each machine's resource. Therefore, virtualized system cannot exert its full performance. In this paper, we compare the performance between Hadoop system with virtualization and it without virtualization to evaluate and analyze the difference of practical performance. As a result of the experiment, virtualized Hadoop system is advantageous when dealing with a large number of small files and we found that there is no significant performance degradation in the case of processing small-load unit tasks.

Keywords: Hadoop,HDFS,MapReduce,Multi-core CPUs,Virtual Machines

1. はじめに

仮想化とは，CPUやメモリ等のハードウェア内のリソースを，物理的構成にとらわれず統合，分割する技術のことである．近年，この仮想化技術に対する注目が高まっており，仮想化技術に関するIDCの発表 [9] によると，2014年までにサーバの70 %が仮想化環境で運営されると予想されている．仮想化技術は，計算機のリソースを論理的に分割させて複数の計算機として利用することができるため，構成に必要な計算機の数より少ない計算機でクラスタを構成することが可能となる．この特徴は，実際の計算機を購

[1] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
[a)] a-isii@ist.osaka-u.ac.jp

入する費用を節約するだけではなく，計算機を設置する場所のスペースコスト，また計算機運用の電力コストの削減も可能にするため，仮想化技術を取り入れる企業が急増している．更に，仮想化を行うことで各計算機のバックアップ，移動などが容易になり，かつ計算機のリソースを動的に変更させることも可能となる．

また，近年クラウドコンピューティングの浸透が進み，Web コンテンツや IT システムにより生成されるデータ量が急増している．情報社会においてあらゆる形で生成される膨大な量のデータは，既存のデータ処理方法では，保存及び管理が非常に困難である．このように，既存の方法での処理が難しいほどの膨大なデータは BigData と呼ばれ，近年研究課題として注目を浴びている．Google では BigData の処理に対応するため，大量のデータを格納できる分散ファイルシステム [2] と分散処理を自動化するフレームワーク [3] を独自開発した．この Google のファイルシステムとフレームワークの概念を基に開発されたのが，オープンソースの並列分散処理フレームワーク **Hadoop**[1] である．

Hadoop システムの特徴としては，コモディティなマシンで高性能の分散システムを構成可能であること, 台数を増やすことで線形的に性能向上すること, データの複製により高い耐故障性を持つことがあげられる. よって仮想化技術を用いると，少数の物理計算機上で大規模な Hadoop クラスタが構成可能であり，コスト削減やマシン管理の容易さなど仮想化の利点を享受することができる．

しかし，仮想計算機 (VM) を稼働させるためには仮想マシンモニタ (VMM) が介在するため，VMM によるオーバーヘッド [11], [12], [13] の発生は避けられない．このオーバーヘッドの発生により，VM で構築された Hadoop クラスタの性能低下が考えられる．VMware 社の発表 [5] では，VSphere を用いて複数の CPU コアと対応づけられた VM を作成し，その VM で構築した Hadoop システムの性能評価を行なっている．その結果，1 台の物理マシン上に 1 台の VM を構築した場合は平均 4% の処理速度の低下，1 台の物理マシン上に複数の VM を構築した場合は最大 14% の処理速度の向上が見られた．しかし，VMware 社の実験 [5] で用いられた計算機は，複数の高性能プロセッサと膨大なメモリを搭載しており，仮想化技術のためのリソース消費の割合が非常に少ない環境である．また性能評価の際には，タスク数が少なく，ファイルサイズが大きい処理を実行していた．

本報告では，仮想化技術のためのリソース消費の割合が高くなるコモディティなマシンで，ジョブあたりのファイルサイズ，タスク数を変化させた場合の実験を試みる．仮想化ソフトウェア Xen[7] を用いて 1 台の VM が 1 コアを持つ Hadoop システムを構築し，物理マシンのみで構成された Hadoop システムとの比較実験を行った．その結果，ファイルサイズが小さい場合，最大 25% の処理速度の向上が見られた．また，VM で構成された Hadoop システムの処理速度が物理マシンで構成された Hadoop システムよりも遅い場合，1 つのノードが実行する処理を，分散して仮想計算機のリソースで扱える程度の負荷にすることで，性能差が縮まることを確認した．

本報告の構成は以下のとおりである．まず，2 章において Hadoop について説明する．次に 3 章で評価実験環境について説明する．4 章では, 評価実験結果について述べる．最後に 5 章で本報告の結果をまとめる．

2. Hadoop

Hadoop とは，大規模データを効率的に処理するためのオープンソースの並列分散処理基盤である．複数の計算機で並列的に処理を行い，結果を集約することで高速処理を実現している．Apache[6] のプロジェクトとして開発が進められており，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．主な構成要素は，Hadoop Distributed File System(以下 HDFS)[4] と，MapReduce[3] である．

Hadoop システムは，Master・Worker 型のシステム構成であり，システム全体の管理は 1 台の Master ノード，実際の並列処理は複数の Worker ノードが担っている．

2.1 HDFS

HDFS とは，大規模データを効率良く管理・処理するために設計された論理的な分散ファイルシステムである．巨大なファイルを複数の計算機に分割して保持することにより，複数の計算機のストレージを 1 つの巨大なストレージとして扱うことを可能にしている．HDFS では Master ノードの役割を果たすものを NameNode，Worker ノードの役割を果たすものを DataNode と呼ぶ．NameNode はファイルシステム全体の管理を行い，DataNode は実際のデータの格納先となる．HDFS 内では，ファイルは固定長のブロックに分割され管理されており，ブロックを複数の DataNode に複製配置することにより耐故障性を高めている．

2.2 MapReduce

MapReduce とは，大規模なシステムにおいて，膨大なデータを高速で並列分散処理するアプリケーションを作成するためのプログラミングモデルおよびソフトウェアフレームワークである．MapReduce では Master ノードの役割を果たすものを JobTracker，Worker ノードの役割を果たすものを TaskTracker と呼ぶ．JobTracker は，分散処理の指示，管理を行い，TaskTracker は実際に処理を行う．MapReduce による分散処理では，まず JobTracker がクライアントからジョブと呼ばれる分散処理要求を受け取る．

JobTracker はクライアントから受け取ったジョブをタスクと呼ばれる小さい単位の処理に分割し，複数の TaskTracker にタスクの処理を要求する．以降，JobTracker はクライアントからのジョブが全て完了するまで，ジョブやタスクの処理の進捗，タスクの割り当て，TaskTracker の死活などの状況を監視し，管理する．

計算処理は Map フェーズと Reduce フェーズに分かれており，Map フェーズで実行するタスクを map タスク，Reduce フェーズで実行するタスクを Reduce タスクと呼ぶ．Map フェーズでは，map タスクを割り当てられたノードが HDFS に格納された入力データに対して処理を行い，必要なデータを抽出する．抽出されたデータは Reduce フェーズを実行するノードへ転送される．Reduce フェーズでは Map フェーズで抽出された情報の集約を行い，処理結果を得る．

3. 評価実験環境

表 1 計算機環境
Table 1 Specification of Machines.

	Master ノード	Worker ノード
CPU	Intel Core i3(3.07GHz) 2 Cores(HT 対応)	Intel Core i3(3.10GHz) 2 Cores(HT 対応)
RAM	2GB(DDR3)	4GB(DDR3)
HDD	500GB(S-ATA Ⅱ)	500GB(S-ATA Ⅱ)
OS	CentOS 5.7 (Linux Kernel 2.6)	CentOS 5.7 (Linux Kernel 2.6)
台数	1 台	20 台

本実験では，21 台の計算機を用いた．1 台は Master ノード，残りの 20 台は Worker ノードの役割を担っている．この 21 台の物理マシンから構成される Hadoop システムと，20 台の物理マシン上に構築した 40 台の VM を Worker ノードとして持つ Hadoop システムの比較実験を行った．各物理マシンの計算機環境を**表 1** に示す．各マシンは 1Gigabit Ethernet で接続されている．

3.1 仮想化環境

表 2 仮想計算機環境
Table 2 Specification of Virtual Machines.

RAM	1.5GB
HDD	50GB
OS	CentOS 5.7 (Linux Kernel 2.6)

仮想化は，仮想化ソフトウェア Xen[7] を用いて準仮想化方式で行った．使用した Xen のバージョンは Xen3.1.2 である．各 Worker ノード用計算機に対して，計算機のコア数と同数になるよう 2 台ずつ VM を設置した．本実験に用いる計算機は Hyper-Threading 技術 [14] に対応しており，1 つの物理 CPU コアを仮想的に 2 つの CPU コアのように扱える．よって**図 1** のように VM の仮想 CPU 数を 2 つにし，それぞれを同一コアの異なるスレッドに対応づけた．VM の計算機環境は**表 2** に示す．

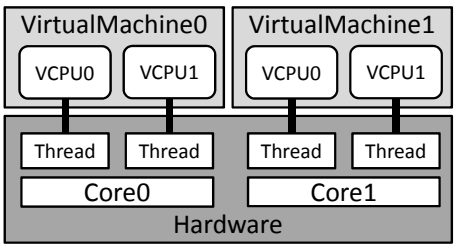

図 1 仮想計算機の CPU 設定
Fig. 1 CPU configuration of the virtual machine.

3.2 Hadoop 環境

実験に用いた Hadoop のバージョンは 1.0.3，Java のバージョンは 1.7.0_01 である．Hadoop システムは耐故障性向上のためデータを異なるマシンに複製配置する．本実験では複製数を 2 とした．しかし仮想化 Hadoop システムでは，同じ物理マシン上の異なる VM にデータが複製配置される可能性がある．その場合，物理マシンの故障により元のデータと複製データが共に失われてしまう．Hadoop では，ラック設定 [8] を行うと 2 つ目の複製は別のラックに配置されるようになるため，仮想化 Hadoop システムでは，同一物理マシン上の VM2 台を 1 つのラックに設定することにより，データの複製が同一物理マシン上に配置されないようにした．また本実験では，HDFS にデータを保存する際のブロックサイズはデフォルト値の 64MB である．Hadoop では MapReduce 処理において，各 Worker ノードに同時に割り当てるタスクの数を設定することが可能である．割り当てるタスクの数はデフォルトでは 2 に設定されている．同時に割り当てる map タスクの数を変化させた予備実験の結果 (**図 2**) より，物理マシン 1 台あたり 4 つの map タスクを割り当てた場合が最も処理時間が短くなったため，同時に割り当てる map タスク数は 4 にする．また，単一コアで構成された仮想化クラスタの実験環境では，ノードあたりの同時に割り当てる map タスク数を 2 つにした．これにより，物理クラスタ，仮想化クラスタ共に，同時に 80 個の map タスクを受け入れることが可能となる．

4. 評価実験

4.1 Hadoop ベンチマーク

評価実験では以下の 3 つのベンチマークを用いた．

(1) Pi

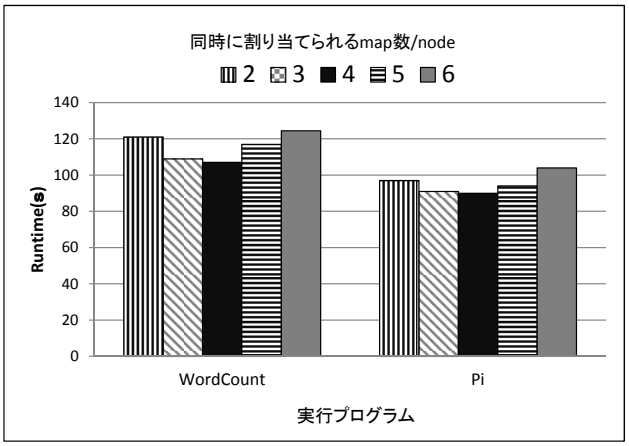

図 2 物理クラスタ実行結果
Fig. 2 Execution result of Physical Machine Cluster

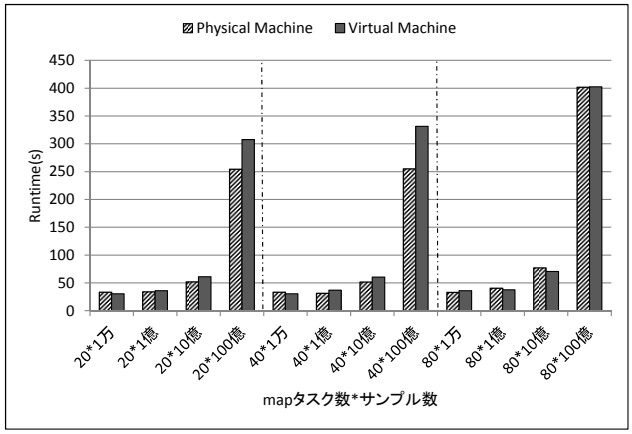

図 3 Pi の実行結果 (1)
Fig. 3 Execution result of Pi(1)

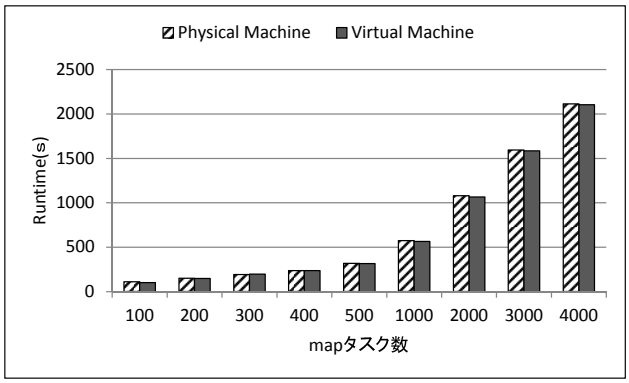

図 4 Pi 実行結果 (2)
Fig. 4 Execution result of Pi(2)

モンテカルロ法によって円周率を求めるプログラムである．計算処理が多く，入出力処理はほとんど無い．計算サンプル数と map タスク数の 2 つの引数をもつ．Map フェーズで平面にサンプル数の数だけランダムにプロットした点のうち円の中に入る数を数え，円周率を求める．Reduce フェーズでは，各 Map フェーズの出力値の平均を算出する．

(2) RandomTextWriter
HDFS にランダムなテキストデータを書き込むプログラムである．RandomTextWriter は，(1) の Pi とは異なり，計算処理は殆ど行わず大量の書き込み作業を行う．各ノードに書き込まれるデータ量は等しいため，ノードの処理量の差はほとんど生じない．Map フェーズでは，データを生成しシーケンシャルファイルとして HDFS に書き込む．Reduce フェーズは存在しない．

(3) WordCount
入力ファイル中の各単語の出現頻度を数えるプログラム．Map フェーズでは，HDFS に格納されている入力ファイルから指定ブロックサイズごとにファイル中に出現する単語を抽出し，Reduce 処理を行うノードに送信する．Reduce フェーズでは，Map フェーズから送信されたファイルを基に単語ごとの出現回数をカウントし結果を HDFS に格納する．WordCount では，少しの計算処理と大量の読み出し作業が行われる．

4.2 実験結果

(1) Pi
サンプル数,map タスク数を変化させて実験を行った．サンプル数を増加させることにより，1 タスクあたりの計算処理による負荷が増加する．map タスク数を増加させることにより，割り当てられた map タスクが完了した際のデータ通信と新たな map タスクの割り当てのためのデータ通信が頻発し，ネットワーク転送量が増加する．

図 3 は，map タスク数 20，40，80 に対してサンプル数を変化させて Pi を実行した際の結果である．map タスク数がコア数と同数の 40 以下の場合，サンプル数の大きな実行では，物理マシンの方が処理時間が短い．Hadoop では,map タスクは各ノードに均等に割り当てられ，割り当てきれなかったタスクは，割り当てられた map タスクが完了したノードに逐次割り当てれらる．よって map タスク数が 20 の場合，worker ノード数が 20 である物理マシンによるクラスタ，worker ノード数が 40 である VM によるクラスタの双方で，異なるノードに 1 つずつ map タスクが割り当てられることになる．物理マシンによるクラスタでは worker ノードである計算機が 20 台のため，全ての map タスクが別の物理マシン上で実行されることになる．一方 VM によるクラスタでは，20 個の map タスクが 40 台の VM に割り当てられるため，同一物理マシン上の VM2 台に map タスクが割り当てられる場合が発生する．このような割り当てが発生した場合，map タスクを 1 つも割り当てられない VM が存在することになる．このように VM によるクラスタでは，クラスタ内

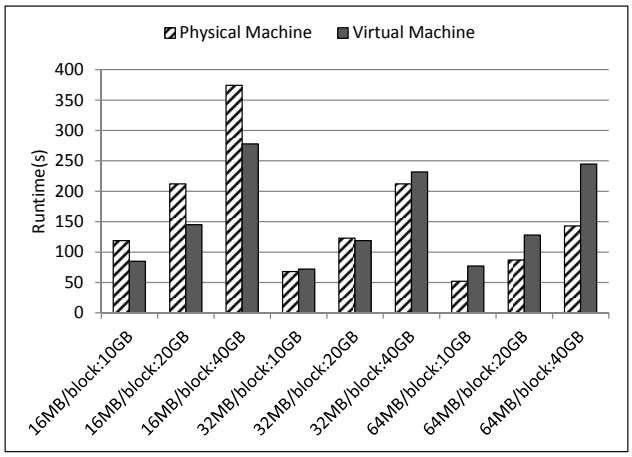

図 5 RandomTextWriter の実行結果
Fig. 5 Execution result of RandomTextWriter

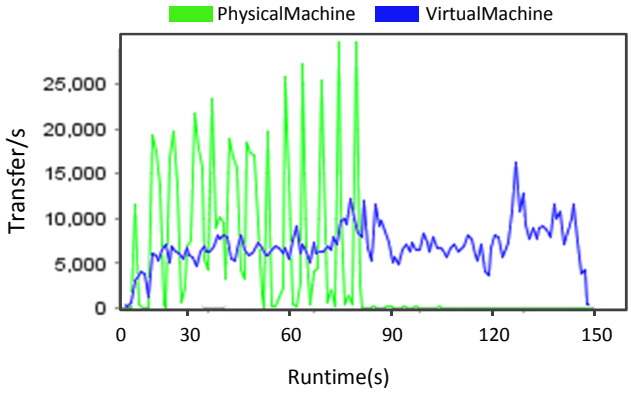

図 6 RandomTextWriter:64MB/block:20GB の実行結果
Fig. 6 Execution result of RandomTextWriter 64MB/block 20GB

の計算機資源を全て利用出来ないため，処理性能の低下につながったと考えられる．また map タスクが 40 の場合，1 つのコアが 1 つのタスクを処理することになり，マシン性能が高い物理マシンの方が有利であるため，性能差が生じたと考えられる．しかし map タスク数が 80 の場合，サンプル数によらず処理時間の差がほぼ無くなっている．

図 4 は，サンプル数を 10 億に固定し，map タスク数を変化させた場合の結果である．map タスク数によらず処理時間の差は殆ど見られない．これにより，1 つのノードに対し演算処理の多いタスクを複数割り当てるような実行では，仮想化によるオーバーヘッドがほぼ無いと考えられる．

(2) RandomTextWriter

RandomTextWriter では 1 つの Map で指定サイズのシーケンシャルファイルを書き込む．よって，あるサイズのデータを書き込む場合，シーケンシャルファイルあたりのサイズが小さいほど，プログラム全体で実行される Map タスク数は増加する．各 Map タスクでシーケンシャルファイルを生成し，HDFS に書き込

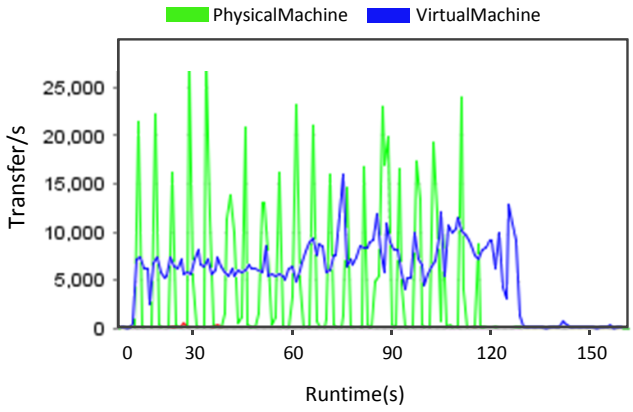

図 7 RandomTextWriter:32MB/block:20GB の実行結果
Fig. 7 Execution result of RandomTextWriter 32MB/block 20GB

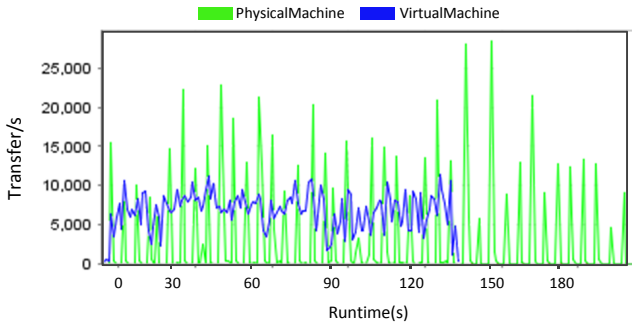

図 8 RandomTextWriter:16MB/block:20GB の実行結果
Fig. 8 Execution result of RandomTextWriter 16MB/block 20GB

んでおり，出力処理が多いプログラムである．

図 5 は，16MB，32MB，64MB のサイズのシーケンシャルファイルを合計入力ファイルサイズが，10GB,20GB,40GB となるように書き込んだ場合の実行結果である．ファイルサイズがブロックサイズと等しい 64MB の場合，物理マシンの方が処理時間が短く，ファイルサイズがブロックサイズに比べ小さくなるにつれて，VM の方が処理時間が短くなっている．

図 6，**図 7**，**図 8** は，64MB，32MB，16MB のサイズのシーケンシャルファイルを 20GB 書き込んだ際の，Worker ノードである計算機の 1 秒あたりの転送数 (デバイスに対する IO リクエスト数) の合計値である．map タスクが割り当てられるとファイルの書き込みを行い，書き込み完了後，次の map タスクの割り当てを待つ．このグラフより，ファイルサイズが小さい場合，物理マシンでは一定量のファイル書き込み後，通信処理を行い，map タスクが割り当てられるのを待機するため，idle 状態が存在している．一方，VM では 2 台の VM が独立して map タスクの実行と通信を行うため，idle 状態がほぼ発生していない．またファイルサイズが大きい場合，物理マシンは，ファイルサイズが小さい場合と比較してブロック数が減少したた

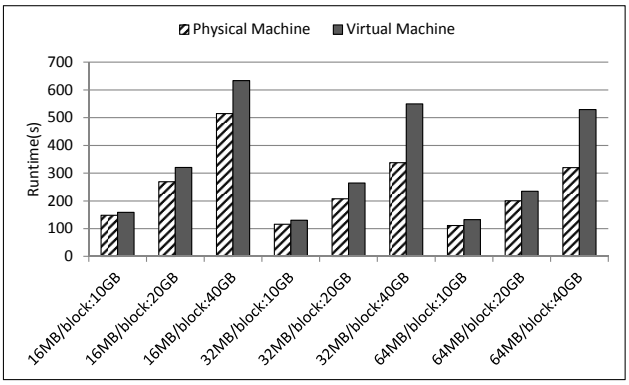

図 9 WordCount の実行結果:Reduce タスク数 1
Fig. 9 Execution result of WordCount.Number of Reducer 1.

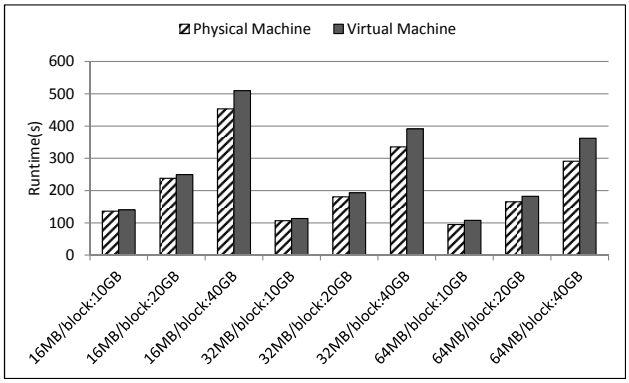

図 11 WordCount の実行結果:Reduce タスク数 2
Fig. 11 Execution result of WordCount.Number of Reducer 2.

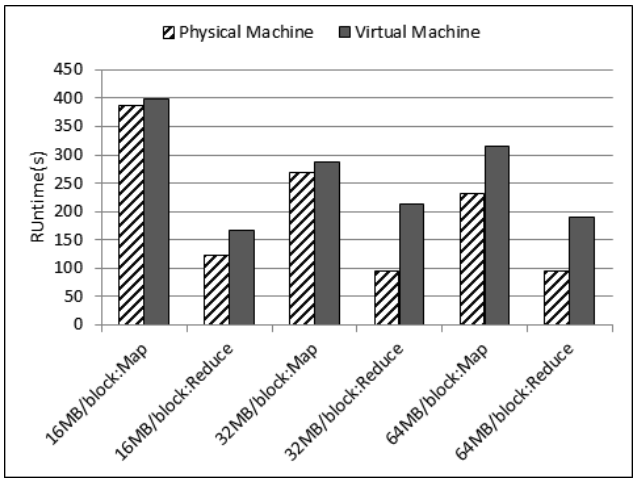

図 10 入力ファイルサイズ 40GB
Map フェーズ，Reduce フェーズの実行時間:Reduce タスク数 1
Fig. 10 Execution time of Map phase and Reduce phase.Input file size 40GB,Number of Reducer 1.

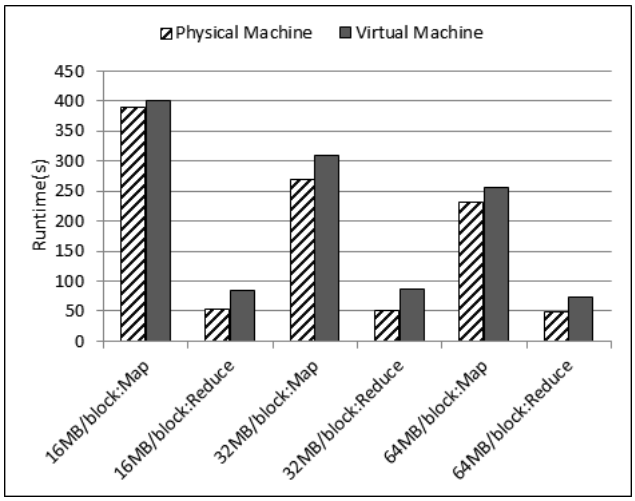

図 12 入力ファイルサイズ 40GB
Map フェーズ，Reduce フェーズの実行時間:Reduce タスク数 2
Fig. 12 Execution time of Map phase and Reduce phase.Input file size 40GB,Number of Reducer 2.

め idle 状態になる頻度が少なくなり，かつファイルサイズが大きく連続したディスク領域に書き込み可能なため最大転送量が大きくなっていることがわかる．以上より，ファイルサイズが小さい場合は VM が，ファイルサイズが大きい場合は物理マシンが優れていると考えられる．

Hadoop システムでは，一般的なファイルシステムと比べデフォルトで 64MB と，非常に大きなブロックサイズを採用しており，大きなファイルの処理に適した設計がされている．よって小さなファイルを扱う場合，処理性能が低下する問題 [15] がある．この問題に対する対策として，小さなファイルを統合して 1 つのファイルとして扱う方法などがあり，本実験が示す VM の利用も有用な方法のひとつであると考えられる．

(3) WordCount

WordCount はデータの読み出し，処理，データの格納というワークフローを持っており，データの入出力が頻繁に発生するプログラムである．**図 9** は，RandomTextWriter の出力に対し，Reduce タスクを 1 つにして WordCount を行った際の実行結果である．Reduce タスクが 1 つのため，1 台のマシンで全ての Map タスクの処理結果を集約し，処理を行っている．図 9 の結果より，VM に比べ物理マシンの方が処理時間が短くなっていることがわかる．40GB の入力ファイルを処理する際の Map フェーズの実行時間，Reduce フェーズの実行時間を**図 10** に示す．図 10 より，VM と物理マシンでは，Map フェーズよりも Reduce フェーズの処理時間に差が出ていることがわかる．Reduce タスクが 1 つの場合，1 台のマシンで全ての Map タスクの処理結果を集約し，処理を行うため，1 台あたりのマシン性能の高い物理マシンの方が実行時間が短くなったと考えられる．そこで Reduce タスクを 2 つに増やし，Reduce タスク 1 つあたりの処理負荷を減らして再度同じデータに対して WordCount を実行した．その結果を**図 11** に示す．結果として，VM と物理マシン共

に処理時間が短くなり，VM と物理マシンの差が小さくなっている．この時の 40GB の入力ファイルを処理する際の Map フェーズの実行時間，Reduce フェーズの実行時間を**図 12** に示す．Reduce タスクが 1 つの場合と比べて，VM と物理マシンでの Reduce フェーズの実行時間の差が短くなっている．これより，タスクあたりの負荷を軽減することで，VM での性能低下を防ぐことができることがわかる．

5. まとめ

本報告では，マルチコア CPU を持つ物理マシン上にコア数と同数の VM を作成し，その VM を用いて Hadoop システムを構築した．仮想化 Hadoop システムと，物理マシンから成る Hadoop システムの比較実験を行った結果，仮想化 Hadoop システムは小さいファイルを大量に扱う場合に有利であること，VM のリソースで扱える程度の負荷の処理では物理マシンから成るシステムとの処理性能の差が少ないことを確認した．

謝辞

本研究の一部は，日本学術振興会科学研究費助成事業 (基盤研究 (B)22300009, (C)24500039, 若手研究 (B)23700056, 挑戦的萌芽研究 24650012) によるものである．

参考文献

[1] Apache Hadoop Project, http://hadoop.apache.org/
[2] Ghemawat S., Gobioff H. and Leung S.T.: The Google file system, ACM SIGOPS Operating Systems Review (2003).
[3] Dean J. et al.: MapReduce: Simplified data processing on large clusters, Communications of the ACM (2004).
[4] K. Shvachko et al.: The Hadoop Distributed File System, IEEE 26th Symposium on Mass Storage Systems and Technologies (2010).
[5] VMware.: A Benchmarking Case Study of Virtualized Hadoop Performance on VMware vSphere 5 (online), http://www.vmware.com/files/pdf/techpaper/VMW-Hadoop-Performance-vSphere5.pdf (2011).
[6] The Apache Software Foundation.: http://www.apache.org/
[7] Barham P., Dragovic B., Fraser K., Hand S., Harris T., Ho A., Neugebauer R., Pratt I. and Warfield A.: Xen and the art of virtualization, ACM SIGOPS Operating Systems Review (2003).
[8] Tom White: Hadoop The Definitive Guide, O'Reilly (2009).
[9] IDC: Worldwide Market for Enterprise Server Virtualization to Reach $19.3 Billion by 2014, http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22605110 (2011).
[10] Big data, http://en.wikipedia.org/wiki/Big_data
[11] Cherkasova L. and Gardner R.: Measuring CPU overhead for I/O processing in the Xen virtual machine monitor,Proceedings of the annual conference on USENIX Annual Technical Conference (2005).
[12] Menon A., Santos J.R., Turner Y., Janakiraman G.J. and Zwaenepoel W.: Diagnosing performance overheads in the Xen virtual machine environment, Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments (2005).
[13] Oracle Corporation.: Oracle VM, Performance Evaluation of Oracle VM Server Virtualization Software(online), http://www.oracle.com/us/026997.pdf (2008).
[14] Intel Hyper-Threading Technology, http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html
[15] Tom White: The Small Files Problem, cloudera, http://www.cloudera.com/blog/2009/02/the-small-files-problem/ (2009).