Vol.2012-ARC-200 No.10
Vol.2012-OS-121 No.10
2012/5/8

仮想クラスタを構成する複数ディスクイメージの効率的移送手法

森田 大希[1]　市川 昊平[1]　阿部 洋丈[1]　伊達 進[1]　下條 真司[1]

概要：近年，仮想計算機 (VM: Virtual Machine) 技術の発達に伴い，仮想クラスタを異なる拠点間で移送する拠点間移送に関心が高まりつつある．しかし，仮想クラスタの移送では，データサイズの大きい VM ディスクイメージを多数転送する必要があるため，移送時間が長くなることが問題である．そこで，本研究においては仮想クラスタを構成する複数の VM ディスクイメージ間には重複内容が多いことに着目し，これらの重複内容の除去を行うことにより，移送時間の削減を図った．具体的には，重複除去の手法として，ローカル重複除去，多段階重複除去の 2 種類の重複除去手法を提案し，これらを実行するためのプログラムの実装を行った．本研究では，これらの実装したプログラムの処理時間の測定を行い，VM のサイズや個数，使用可能帯域幅などの任意の条件下における仮想クラスタ移送時間の推定式を導出した．この推定式を利用することにより，WAN 実効帯域幅等の条件に応じて，2 種類の重複除去手法，および重複除去を行わずに移送する手法の 3 種類のうち，いずれが移送時間を最短にするものであるかの決定指針を与えることができた．

キーワード：仮想化，仮想クラスタ，マイグレーション，重複除去

An Effective Method of Transferring VM Disk Images Composing a Virtual Cluster

Morita Daiki[1]　Ichikawa Kohei[1]　Abe Hirotake[1]　Date Susumu[1]　Shimojo Shinji[1]

Abstract: Transferring virtual clusters across different sites has been gathering much attention recently due to the development of virtual machine technology. However, it takes a long time to transfer VM disk images composing a virtual cluster across different sites since the total size of multiple VM disk images is very large. In this study, we proposed an efficient transferring method that reduces the size of transferring data by deduplicating multiple VM disk images composing a virtual cluster. In particular, we have developed two deduplication methods, local deduplication and multistage deduplication. We have evaluated the execution time of the developed duduplication programs, and derived estimating equations of transferring time of virtual cluster under arbitrary conditions.

Keywords: virtual machine, virtual cluster, migration, deduplication

1. はじめに

近年，計算機資源を効率的かつ有効に利用するために，仮想計算機 (以下，VM: Virtual Machine) 技術 [1], [2], [3] が注目されている．仮想計算機技術により物理的なハードウェアとは独立した計算環境を動的に構築することが可能となるため，データセンタ内の計算機資源の運用技術として盛んに利用されている．

仮想計算機技術は，HPC(High Performance Computing) 分野においても利用されるようになってきており，その一例として仮想クラスタ技術が挙げられる．ジョブを複数プロセスに分割し，複数の計算機により並列計算を行う計算

[1] 大阪大学
Osaka University

機クラスタを VM で構築することにより，ユーザごとの要望に応じたクラスタ環境を容易に構築できるだけでなく，必要となる計算能力に応じて，計算ノードを容易に追加することもできる．そのため，クラスタ環境をより動的かつ柔軟に準備することが可能となる．

さらに，計算機資源運用のさらなる効率化を目的として，VM を異なるデータセンタ間で移送 [4], [5], [6] することに対する期待が高まっている．VM を拠点間で移送して利用効率の低いデータセンタに集約することにより，データセンタ間の負荷分散が行える．それだけでなく，夜間電力の安いデータセンタへの移送によるの省コスト化 [7] や，日中に人による対応が可能であるデータセンタに VM を移送することで，システムの安定運用 [8] も可能となる．HPC 分野においても，仮想クラスタ [9], [10] の拠点間移送が可能となることで，さらなる計算機資源運用の利便性の向上に対する期待がされている．

しかし，仮想クラスタの移送においては，転送しなければならないデータ量とその転送時間が問題になる．VM を別拠点に移送する際には，VM のハードディスクイメージファイルを移行先拠点に転送する必要があるが，それらは一般に数 GB，数十 GB 以上となり，サイズが大きい．そのうえ，仮想クラスタは一般に多数の VM により構成されるため，仮想クラスタを別拠点に移送する場合，サイズの大きい VM ディスクイメージファイルを多数転送する必要が生じる．その結果，移送には非常に膨大な時間を要し，クラスタ使用の利便性は低下する．

本研究では，この仮想クラスタの拠点間移送時間の問題に着目し，効率的な移送手法を提案することによって，移送時間を短縮することを目的とする．具体的には，仮想クラスタを構成する複数の VM にはアプリケーションや解析対象となるデータが重複して含まれていることが多いことに着目し，これらの重複内容をデータ転送前に除去（以下，重複除去）することにより，転送データサイズの削減を図る手法を提案する．このようにデータの重複に着目し，重複除去により移送時間短縮を行う取り組みは現在までにいくつか報告されている [11], [12]．ただし，現在までに報告されている成果は，単独の物理計算機，もしくは単独のディスクイメージレポジトリ上における複数の VM の移送のみの言及に留まっている．本研究では仮想クラスタでの利用を前提として考え，複数の物理計算機上に複数の VM が配備されている状況を想定し，このような状況下で最適な移送手法を模索する．

以下，2 章では，本研究で想定している環境について説明し，本研究で取り扱う課題について述べる．3 章では，本研究で実装したプログラムについて説明するとともに，重複除去の実装について説明する．4 章では，仮想クラスタの拠点間移送に掛かる総時間を推定するために，本研究で実装した重複除去プログラムを，VM ディスクイメージ間のデータ重複割合や VM の平均ディスクイメージサイズ等のパラメータを変更しながら実行し，その処理時間を測定する．そして，その結果より最適な移送手法に関して議論する．最後に，5 章では本研究で提案した仮想クラスタ移送手法についてまとめるとともに，今後取り組むべき課題について述べる．

2. 重複除去の実装における課題

本節では，本研究において想定している環境ついて説明する．そして，本研究で取り組む問題を明らかにするとともに，問題解決のための課題について述べる．

2.1 想定環境

本研究で想定する仮想クラスタは，複数の物理計算機上で動作する複数の VM により構成されているものとし，VM ディスクイメージもそれぞれの物理計算機上に配備されているものとする．この，複数の VM ディスクイメージ間で重複除去を行って移送先拠点へ転送し，元の VM ディスクイメージに復元するまでの処理を仮想クラスタの拠点間移送とする．

本研究で用いる VM は，OS インストール等の最低限の共通設定がなされたベース OS イメージをもとにして起動され，個々のユーザがそれぞれの目的に応じて追加でインストールしたアプリケーションやデータなどはベース OS イメージからの差分ディスクイメージとして格納されるものとする．そして，ベース OS イメージのデータは移送元と移送先の拠点間で同一のものが既に配備され，各 VM から共有されていることを前提とする．この際，ベース OS イメージ側のデータは読み込み専用として働き，その内容と異なる部分は，差分ディスクイメージの方に書き出される．したがって，多くの VM に共通する OS 部分などのデータが共有されることになるので，大幅なディスクスペースの削減を実現できる．このようなベースイメージを基にした差分ディスクイメージを個々の VM ごとに作成する機能は現在，一般的に実装されており，広く利用されている．本研究で想定する環境の概要を，図 1 に示す．本研究では，具体的にはハイパーバイザとしては KVM を用い，VM ディスクイメージとしては qcow2 形式を利用するものとする．qcow2 形式は KVM 環境下において広く用いられている VM ディスクイメージのフォーマットであり，本研究で利用する，ベースイメージと差分イメージにより VM を動作させる機能を提供している．

以上の想定環境において，仮想クラスタの移送を行う際には，差分ディスクイメージのみを転送することになる．したがって，本研究で重複除去処理を行う対象は，個々の VM における差分ディスクイメージである．HPC 分野においては，データの並列的な解析のために，解析対象となる巨大なデータを複数の計算機で冗長的に保持する場合が

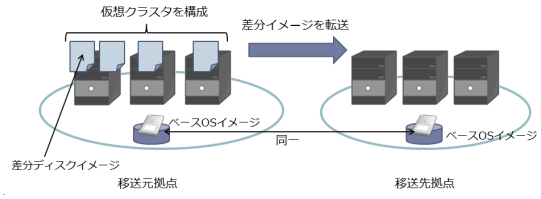

図 1 想定環境
Fig. 1 Assumed Environment.

ある．そのような場合には，差分ディスクイメージのサイズも大きく，かつ，複数の差分ディスクイメージには重複内容が多く含まれる．そのため，重複除去を行うことで転送すべきデータサイズを大きく減らすことができ，仮想クラスタ移送時間を大きく短縮できると考えられる．

2.2 本研究で取り組むべき課題

仮想クラスタのディスクイメージ間で重複除去して移送する際には，重複除去に要する時間と拠点間でのデータ転送時間の間にトレードオフの関係が生じる．つまり，重複除去により転送データサイズを小さくしようとすればするほど，それだけ重複除去に要する時間は長くなってしまう．このトレードオフ関係が存在するため，移送を行いたい仮想クラスタ構成や移送する拠点間の WAN 帯域幅等の環境に応じて，移送時間を最短化ために行うべき重複除去の程度が異なると考えられる．このようなトレードオフ関係を鑑み，本研究では，仮想クラスタの移送を行う手法として，重複除去の程度の異なる以下に示す 3 つの手法を提案する．

- 手法 1: 重複除去無し手法
  仮想クラスタを構成する全ての VM に対して重複除去を全く行わず，ディスクイメージファイルを一つずつ順番に転送する．
- 手法 2: ローカル重複除去手法
  各物理計算機上で動作している複数の VM のディスクイメージを，その各物理計算機ごとに重複除去を行い，重複除去後データを転送する．
- 手法 3: 多段階重複除去
  手法 2 のローカル重複除去により，各計算機毎それぞれで重複除去済みのデータが得られるが，これらの複数重複除去データにはさらに重複内容が含まれる．多段階重複除去は，物理計算機毎に存在する重複除去データを一方の物理計算機から別の物理計算機に LAN 経由で転送し，それらの間で更なる重複除去を実施し，これを繰り返す．

前述の複数の重複除去手法を実装して仮想クラスタ移送時間を短縮するにあたり，本研究では仮想クラスタの移送時間を以下の式 1 により定義し，この式に基づいて各手法の評価を行う．

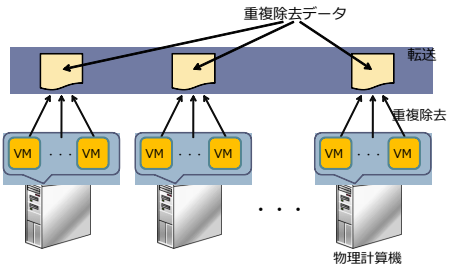

図 2 ローカル重複除去
Fig. 2 Local Deduplication.

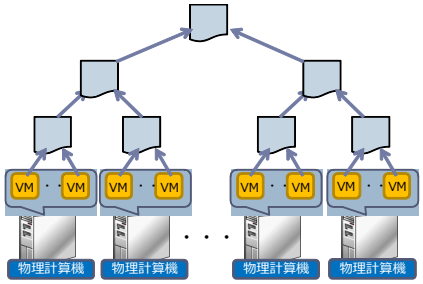

図 3 多段階重複除去
Fig. 3 Multiple Deduplication.

$$T_{total} = T_{dedupe} + T_{trans} + T_{restore} \qquad (1)$$

ここで，T_{total} は仮想クラスタ移送時間で，移送元拠点で仮想クラスタ移送を開始した時点から，移送先拠点において，全ての VM ディスクイメージを重複除去後データから復元完了するまでの時間とする．T_{dedupe} は移送元サイトにおいて重複除去を行う時間，T_{trans} は WAN を経由してデータを転送するのに時間，$T_{restore}$ は移送先サイトにおいて，重複除去後データから複数の VM ディスクイメージに復元するのに要する時間とする．

式 1 の仮想クラスタ移送時間について，前述の通り，重複除去時間 T_{dedupe} と転送時間の間にはトレードオフの関係がある．手法 1 の重複除去を行わない場合では，重複除去および復元を行わないので，T_{dedupe} および $T_{restore}$ は 0 となる．しかし，転送するデータには重複した内容が含まれるためにサイズが大きくなってしまい，結果として T_{trans} は膨大になってしまう．手法 2 のローカル重複除去では，T_{dedupe} は中程度かかってしまい，かつ，移送先拠点でディスクイメージの復元を行う必要があるため，復元時間 $T_{restore}$ がかかってしまう．しかし，重複除去を行うため，T_{trans} は中程度で済む．手法 3 の多段階重複除去手法では，T_{dedupe} は大きくかかってしまい，かつ，復元が必要となってしまうが，重複除去に時間を長く費やした分だけ転送データサイズは小さくなるため，T_{trans} を短くすることができる．これらの手法の違いを表 1 にまとめる．

次に，仮想クラスタ移送時間 T_{total} についてより詳しく分析するため，移送したい仮想クラスタの構成を表すパラメータを以下のように定義する．

表 1 重複除去時間と転送時間のトレードオフ関係
Table 1 The Tradeoff Relation.

重複除去手法	重複除去時間	転送時間	復元の必要性
重複除去無し	0	大	無
ローカル重複除去	中	中	有
多段階重複除去	大	小	有

- VM を動作させる物理計算機の個数：n 個
- 各物理計算機上における VM の個数：m 個
- VM ディスクイメージの平均サイズ：s GB
- 平均サイズ s に対する重複データの割合（重複率）：r
- 移送元拠点内における LAN の実効帯域幅：B_{LAN}
- 移送を行う拠点間の WAN の実効帯域幅：B_{WAN}

これらのパラメータを用いると，重複除去を行った後のデータサイズを求めることができる．$j = 0$ がローカル重複除去を表すとし，j 段階の重複除去を行っうことにより出力されるデータのサイズを S_j とすると，

$$S_j = rs + (1 - r)sm2^j \tag{2}$$

である．したがって，転送時間 T_{trans} は，上記の S_j を B_{WAN} で割った値として求められるため，一意に表すことが可能である．一例として物理計算機内でローカル重複除去を行った場合のデータ転送時間 T_{trans} は，

$$T_{trans} = \frac{\{rs + m(1 - r)s\}n}{B_{WAN}} \tag{3}$$

と表すことができる．

しかし，重複除去時間 T_{dedupe} および復元時間 $T_{restore}$ については，プログラムの実装に依存するものであり，パラメータが分かっただけでは求めることができないため，実際にプログラムを動作させて時間を測定する必要がある．

したがって，本研究では以下の課題に取り込む．まず，提案する重複除去手法を実現する，重複除去プログラム，及び，復元プログラムを実装する．さらに，移送する仮想クラスタの VM ディスクイメージサイズや個数，利用可能帯域幅などの条件に応じた移送時間を推定するため，実装プログラムの実行時間を評価し，得られた結果より最適な仮想クラスタ移送手法の選択指針を構築する．

3. プログラムの実装

本研究では，提案するローカル重複除去手法および多段階重複除去手法を実現するにあたり，2 種類の重複除去プログラムと，1 種類の復元プログラムを実装した．以下，3.1 節では実装したこれらのプログラムの概要を示し，3.2 節では重複除去手法，重複除去及び復元にあたり必要となるデータを格納するためのハッシュテーブル，復元手法の詳細についてそれぞれ説明を行う．

3.1 実装するプログラムの概要

本研究では，以下の 3 種類のプログラムの実装を行う．

1. ローカル重複除去プログラム

ローカル重複除去プログラムは，各物理計算機上に動作する複数の VM のディスクイメージから重複部分を除去して，転送すべきデータサイズを削減するプログラムである．入力としては任意個数の VM のディスクイメージファイルを受け取る．出力として，各 VM ディスクイメージの重複部分とそのディスクイメージ内での位置の対応付けを記録するハッシュテーブルファイルと，重複除去を行った後のデータである重複除去済みファイルを出力する．

2. 多段階重複除去プログラム

多段階重複除去プログラムは，複数のローカル重複除去プログラム，もしくは前段階の多段階重複除去プログラムの出力結果から更に重複除去を行い，データサイズを削減するプログラムである．入力としては，ローカル重複除去プログラム，もしくは前段階の多段階重複除去プログラムからの出力である，ハッシュテーブルと重複除去済みファイルの組み合わせ 1 ペアとし，任意個数のペアを受け取る．そして，複数のハッシュテーブルを 1 つに統合し，複数の重複除去ファイルからは更なる重複除去を行い，1 ペアの重複除去済みファイルとハッシュテーブルを出力する．

3. 復元プログラム

復元プログラムは，ローカル除去プログラム，もしくは多段重複除去プログラムの出力結果から，任意の複数の VM ディスクイメージの復元を行うプログラムである．入力としては，ハッシュテーブルと重複除去済みファイルの組み合わせ 1 ペアと，どの VM を復元するのかの指示を受け取る．出力としては，復元するように指示された複数の VM のディスクイメージを出力する．

3.2 実装の詳細

本研究において，重複除去は以下のようにして行う．重複除去対象となるファイルを先頭からサイズ一定のブロック単位に区切って扱い，ブロックごとに SHA-1 ハッシュ関数を適用することで，ハッシュ値の計算を行う．複数のブロックに対してハッシュ値を比較し，同一であればそれらのブロックは内容が重複しているものとみなす．ハッシュ値の比較により，あるブロックが他のブロックと同一であると分かった場合，ブロックデータをファイルに書き込む際に重複して書き込まれないようにすることで，重複内容の除去が可能となり，転送すべきデータサイズは小さくなる．

前述したように，本研究で扱う VM ディスクイメージのフォーマットは qcow2 形式である．qcow2 ファイルでは，クラスタと呼ばれる一定サイズ単位でデータが扱われる．したがって，本研究ではブロックの単位としてこのクラスタの区切りを用いる．以下に，ハッシュテーブル，ローカ

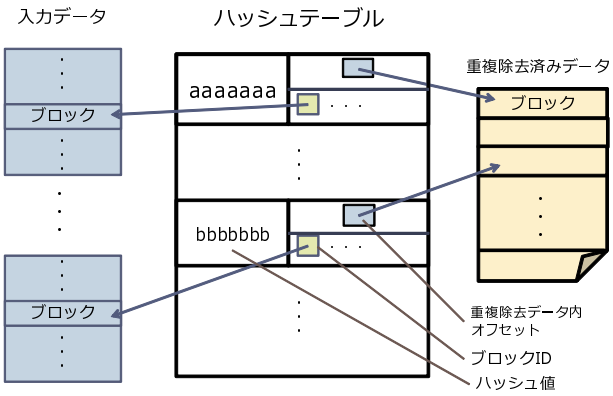

図 4 ハッシュテーブルの実装
Fig. 4 Hash Table.

ル重複除去，多段階重複除去，復元プログラムの実装の詳細についてそれぞれ述べる．

3.2.1 ハッシュテーブルの実装

まず，ハッシュテーブルについて説明を行う．本研究で実装するハッシュテーブルは，あるブロックに対して，
(1) ハッシュ値
(2) ブロック ID
(3) 重複除去済みデータ内オフセット
を関連付けて保存するデータ構造である．ブロック ID とは，ハッシュ値に対応するブロックデータが，重複除去前のどのディスクイメージファイルの何番目のブロックであるかを示すデータである．重複除去済みデータ内オフセットとは，ハッシュ値に対応するブロックデータが，重複除去済みファイル内のどの位置に記録されているかを示すアドレスである．データの中身が同じブロックの場合，ハッシュ値は同じ値となる．したがって，ある一つのハッシュ値に対して，複数のブロック ID を関連付けする必要があるため，ブロック ID はリストとして保持する．実際のブロックデータを書き出す重複除去済みデータは重複したブロックを書き出さないため，一つのハッシュ値に対して一つの重複除去済みデータ内オフセットが関連付けられる．本研究で用いるハッシュテーブルを図 4 に示す．

ハッシュテーブルは図 4 のように，各エントリのキー要素にはハッシュ値がソートされた状態で格納され，キーに対応するエントリには重複除去データ内オフセットとブロック ID のリストが格納される．以上のようにデータを格納することにより，ブロックデータのハッシュ値に基づく重複除去，復元復元が可能となる．

3.2.2 ローカル重複除去プログラム

ローカル重複除去は，以下の手順により動作する．
(1) 処理対象の VM ディスクイメージから 1 ブロック読み込む．
(2) 読み込んだブロックに対して，ハッシュ値計算を行う．
(3) ハッシュテーブルを検索し，既に同じハッシュ値のエントリがないか調べる．
(4) 同じハッシュ値が見つからない場合は，今まで読み込んだブロックと重複がないと判断できるため，重複除去済みファイルに該当ブロックのデータを追記する．そして，ハッシュテーブルに新たにエントリを作成し，読み込み元の VM の情報をブロック ID データとして格納し，重複除去済みファイルに追記したブロックデータのオフセットを関連付ける．
(5) 同じハッシュ値が発見された際には，内容が重複するブロックが既に登録済みと判断できるため，ハッシュテーブルの既存のエントリに対し，ブロック ID データを追記するのみで，ブロックデータは書き出さないようにする．

全ての VM ディスクイメージファイルの全てのブロックに対して以上の手順を行い，ハッシュテーブルをファイルとして出力することにより，ハッシュテーブルと重複除去済みファイルのペアが得られる．

3.2.3 多段階重複除去プログラム

多段階重複除去プログラムは，以下の手順によって動作する．
(1) 入力である複数のハッシュテーブル，重複除去済みファイルペアのうち 1 ペアを選び，これらをメインハッシュテーブル，メイン除去済みファイルとする．それ以外の入力ハッシュテーブル，重複除去済みファイルをサブハッシュテーブル，サブ除去済みファイルとする．
(2) サブ除去済みファイルの一つから，1 ブロック読み込み，対応するサブハッシュテーブルから，ハッシュ値を得る．
(3) メインハッシュテーブルを検索し，既に同じハッシュ値のエントリがないか調べる．
(4) 同じハッシュ値が見つからない場合は，重複除去済みファイルに該当ブロックのデータを追記する．そして，ハッシュテーブルに新たにエントリを作成し，読み込み元の VM の情報をブロック ID データとして格納し，重複除去済みファイルに追記したブロックデータのオフセットを関連付ける．
(5) 同じハッシュ値が発見された際には，ハッシュテーブルの既存のエントリに対し，ブロック ID データを追記するのみで，ブロックデータは書き出さないようにする．

以上の手順を全てのサブハッシュテーブル，サブ除去済ファイルペアに対して行うことにより，最終的にメインハッシュテーブル，メイン除去済みファイルのペアを，重複除去処理の結果として得ることができる．

3.2.4 復元プログラムの実装

復元プログラムは，以下の手順により動作する．
(1) 復元を行う VM ディスクイメージの個数だけ，対応する新規ファイルを作成する．

(2) ハッシュテーブルのエントリを一つ読み出し，重複除去済みデータ内オフセットを読み，該当のブロックデータを読み込む．
(3) さらに，該当ブロックに関連付けられているブロック ID を読み込み，復元先の VM とブロックを書き出すべきアドレスを調べ，該当アドレス位置にデータを書き出す．
(4) ハッシュテーブルの先頭から末尾までの全てのエントリに対して操作 (2)，(3) を行う．
以上の手順により，ハッシュテーブルと重複除去済みファイルからの VM ディスクイメージファイルの復元が可能となる．

ただし，上記手順を単純に実装するだけでは，期待したパフォーマンスが出なかった．ハッシュテーブルの各エントリはハッシュ値の値の順に並んでいるため，その順番にしたがってブロックデータを読み書きする場合，データをランダムリード・ライトすることになったのが原因である．前述の重複除去プログラムの場合は，入力の VM イメージディスクをシーケンシャルに読み込んで，重複除去済みファイルをシーケンシャルに書き出す処理が大半であったため，意識する必要がなかった．しかし，復元プログラムでは同時に複数の VM を復元するため，このような問題を意識する必要がある．そこで，本研究では，手順 (2) において，ハッシュテーブルのエントリを一つずつ読み出すのではなく，一括で読み込み，それを重複除去済みデータ内オフセット順にソートするよう実装を追加した．こうすることで，重複除去済みデータの読み込みをシーケンシャルに保つことが出来き，また個々の VM のブロックデータ書き出しもほぼシーケンシャルに実行することが出来たため，大幅なパフォーマンス向上を実現できた．

4. 評価

本節では，仮想クラスタ移送時間の推定式を導出するため，仮想クラスタを構成する各 VM の平均サイズや重複しているデータの割合（以下，重複率），物理計算機ごとの VM の個数等の環境パラメータ変化させながら実装プログラムの実行時間測定を行った，さらに，導出した仮想クラスタの移送時間の推定式を用いて，いずれの手法を用いて仮想クラスタを移送するのがよいのか選択指針を構築した．

4.1 評価環境

本研究で実装したプログラムの処理時間測定は，以下に示す環境で行った．

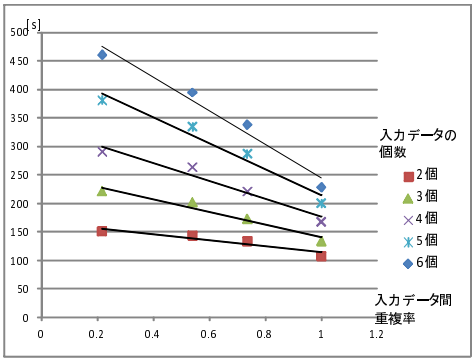

図 5 ローカル重複除去プログラムの実行時間（平均サイズ 5GB）
Fig. 5 Local Deduplication Execution time (5GB).

OS: CentOS release 5.7
Mem: 16GB
CPU: AMD Opteron(tm) Processer2431 800MHz
Cores: 8

4.2 ローカル重複除去プログラムの実行時間

ローカル重複除去プログラムは，複数の VM ディスクイメージを入力とし，重複除去後データおよびハッシュテーブルを出力として動作する．これらのサイズは非常に大きく，プログラムによる処理時間の大半は，入出力処理に費やされていると考えられる．そのため，複数の VM ディスクイメージに対して，重複除去対象の VM の個数 m，データの重複割合 r（以下，重複率）および VM ディスクイメージの平均サイズ s を様々に変えることにより，入出力データサイズを変えながら処理時間の測定を行った．平均データサイズが 5GB の複数 VM ディスクイメージに対する重複率と重複除去時間の関係を図 5 に示す．

図 5 により，重複除去対象となる複数 VM 間のデータ重複率が小さくなるに従い，重複除去に要する時間は線形的に増加することが分かる．このことから，ローカル重複除去に要する時間を $T_ldedupe$ とすると，

$$T_{ldedupe} = ar + b \tag{4}$$

と表すことができる．ただし，a, b は m に依存する定数であるとし，それぞれは図 5 における各直線の傾き，切片を表すものとする．次に，a, b と m の関係を調べる．それぞれの m に対する，a, b の関係をグラフ 6,7 に示す．

グラフ 5 における各直線の傾き a，切片 b もまた，VM の個数 m の増加に対して，線形的に変化することが分かる．このことから，式 4 における a, b は，

$$a = -60m + 69 \tag{5}$$

$$b = -93m - 25 \tag{6}$$

と表すことができる．式 4，5，6 をまとめると，平均サイ

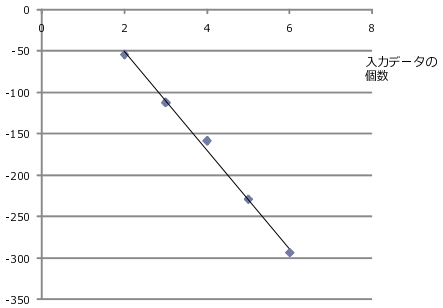

図 6 入力データの個数とグラフの傾き a
Fig. 6 The Relation between the Number of VMs and Lines' Slope.

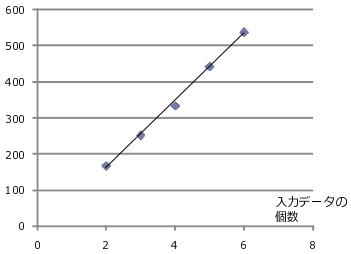

図 7 入力データの個数とグラフの切片 b
Fig. 7 The Relation between the Number of VMs and Lines' Intercept.

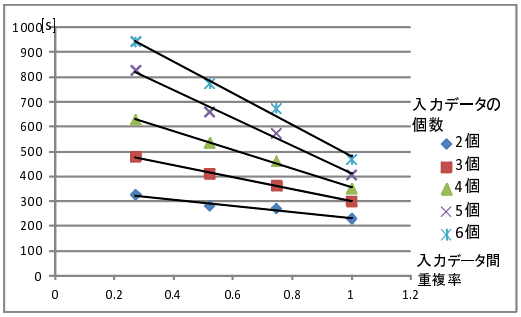

図 8 ローカル重複除去プログラムの実行時間（平均サイズ 10GB）
Fig. 8 Local Deduplication Execution time (10GB).

ズ 5GB の複数 VM の重複除去に要する時間は，

$$T_{ldedupe} = (-60m + 69)r + (-93m - 25) \qquad (7)$$

により推定が可能である．

次に，重複除去対象となる複数 VM の平均サイズ s を 10GB に変えて，同様の実験を行った．得られた結果を図 8 に示す．

得られた結果から，平均サイズが 5GB の場合と同じように，複数 VM 間の重複率の減少に伴い，線形的に重複除去時間が増加することが分かる．よって，平均サイズが 5GB の場合と同様の解析を行うことにより，重複除去対象となる複数の VM の平均データサイズが 10GB の場合，重複除去に要する時間は，

$$T_{ldedupe} = (-145m + 69)r + (-195m - 35) \qquad (8)$$

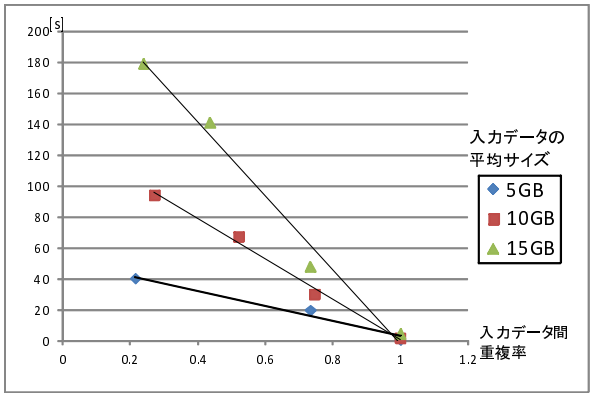

図 9 多段階重複除去プログラムの実行時間

により推定することが可能となる．

式 7,8 および，平均データサイズ s が 0 の時には重複除去時間は 0 となること考えられることから，さらに重複除去時間を s を用いた式にまとめると，ローカル重複除去時間は

$$T_{ldedupe} = \{(-13m + 14)r + (-19m - 3.8)\}s \qquad (9)$$

と推定することができる．

4.3 多段階重複除去プログラムの実行時間

復元プログラムについても同様に，プログラムの処理の大半を入出力処理が占めている．そのため，入力データサイズを変化させるために，入力の組となる重複除去データ 2 つのサイズ平均を変化させ，出力データサイズを変化させるため，2 つの重複除去データに含まれている重複データの割合 r' 変化させて多段階重複除去プログラムの動作時間を測定した．得られた結果を図 9 に示す．

図 9 より，入力データ間のデータ重複率の増加にともなって重複除去に要する時間は線形的に減少することが分かる．入力となる重複除去済みデータ 2 つの平均サイズを s' GB, s' に対するデータ重複率を r' とすると，得られた結果から，重複除去時間の算出と同様の解析を行うことにより，多段階重複除去時間を $T_{mdedupe}$ とすると，$T_{mdedupe}$ は以下の式により推定することができる．

$$T_{mdedupe} = (-19s' + 51)r' + (19s' - 46) \qquad (10)$$

4.4 復元プログラムの実行時間

復元プログラムも同様に，入出力処理が処理時間の大半を占めると考えられる．入力データサイズ，出力データサイズを変化させるために，復元後の VM の個数，復元後の VM 間のデータ重複率，復元後の VM の平均データサイズを変化させながら処理時間の測定を行った．得られた結果を図 fukugentime1,fukugentime1 に示す．

得られた結果から，重複除去プログラムと同様に，復元

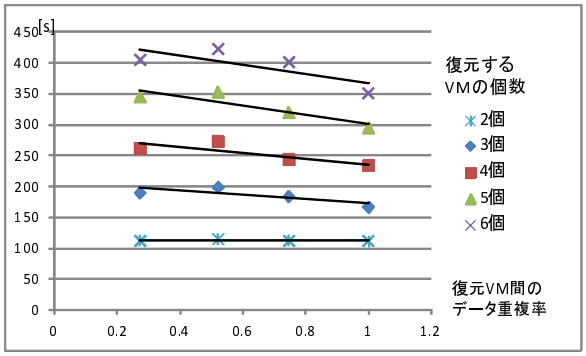

図 10 復元プログラムの実行時間（平均サイズ 5GB）

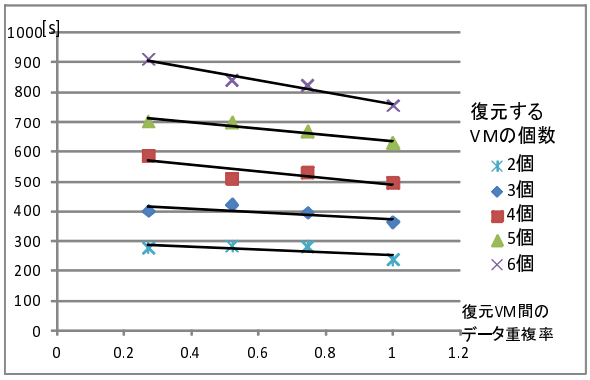

図 11 ローカル重複除去プログラムの実行時間（平均サイズ 10GB）

を行いたい VM の個数に応じて線形的に復元時間は増加することが分かる．重複除去時間の場合と同様に考える．復元される VM の平均データサイズが 5GB の場合には，復元時間 $T_{restore}$ は，

$$T_{restore} = (-23.2m + 41.8)r + (82.7m - 46.4) \quad (11)$$

平均データサイズが 10GB の場合には，

$$T_{restore} = (-48.1m + 86.3)r + (159m - 24.4) \quad (12)$$

これらの式をまとめることにより，平均データサイズ s を用いて，

$$T_{ldedupe} = \{(-4.8m + 8.6)r + (16m - 3.8)\}s \quad (13)$$

により推定することができる．

4.5 仮想クラスタ移送に関する考察

以上で得られた結果を用いて，具体的な仮想クラスタに対していずれの手法が移送時間を最短にするかについて考察を行う．

仮想クラスタ環境の一例として，$B_{LAN} = 1$Gbps, $s = 10$ GB, $m = 3, n = 4, r = 0.5$ とした仮想クラスタの移送時間を考える．WAN の実効帯域幅 B_{WAN} をパラメータとした式として表す．重複除去を行わない場合において，10GB

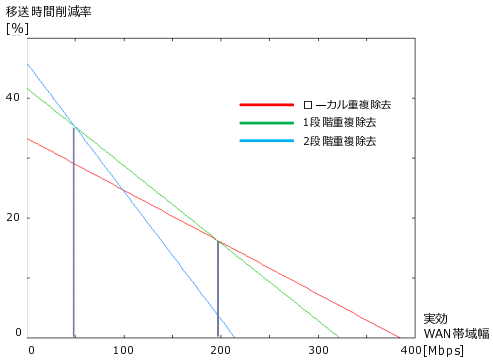

図 12 重複除去無し手法と比較した，各手法の移送時間短縮率
Fig. 12 Migration Time Reduction.

の VM ディスクイメージを 12 個転送することとなるので，仮想クラスタの移送時間は，

$$T_{total} = \frac{120}{B_{WAN}} \quad (14)$$

により表される．次に，ローカル重複除去を行った場合の移送時間の計算を行う．測定データを用いることにより，移送元拠点におけるローカル重複除去に要する時間は 410 秒，移送先拠点における復元時間は 422 秒であることが分かる．また，WAN により転送する総データサイズは，重複除去により 80GB となる．以上の数値を用いて，ローカル重複除去を行う場合の仮想クラスタ移送時間は，

$$T_{total} = 832 + \frac{80}{B_{WAN}} \quad (15)$$

と表わされる．同様にして，仮想クラスタ構成パラメータと実験によって得られた結果を用いて，各手法に対して仮想クラスタ移送時間の計算を順次行う．1 段階の多段階重複除去を行った場合，

$$T_{total} = 1241 + \frac{70}{B_{WAN}} \quad (16)$$

2 段階の多段階重複除去を行った場合，

$$T_{total} = 2051 + \frac{65}{B_{WAN}} \quad (17)$$

以上の式から，重複除去を行わない手法に対して移送時間を何% 短縮できたかを計算し，以下の図 12 に示す．

図 12 により，以下のことがわかる．B_{WAN} が約 380Mbps 以上のときには重複除去を行わない場合に，移送時間が最短となる．B_{WAN} が約 380Mbps 以下のときにはローカル重複除去により，移送時間の短縮が可能となる．B_{WAN} が約 195Mbps 以下のときには 1 段階重複除去により，移送時間は最短となる．B_{WAN} が約 45Mbps 以下のときには 2 段階重複除去により，移送時間は最短となる．

以上のように，実装したプログラムの動作時間の測定を行ったことにより，仮想クラスタ移送時間を短縮できたこ

とを確認することができた．さらに，仮想クラスタの構成が分かった時に，拠点間の実効 WAN 帯域幅によっていずれの移送手法が仮想クラスタ移送時間を最短にするかを求めることが可能となった

5. まとめと今後の課題

本研究では，仮想クラスタをある拠点から別の拠点に移送するのに要する時間を短縮する手法を複数提案，実装するとともに，いずれの手法が移送時間を最短化するかについての考察を行った．具体的には，仮想クラスタを構成する複数の VM には重複するデータが多く含まれることに着目し，重複除去を行うことにより転送データサイズの削減を図った．仮想クラスタの移送手法として，重複除去を行わない手法，ローカル重複除去，多段階重複除去を提案し，これらの手法を実行するためのプログラムを実装した．実装したプログラムを，仮想クラスタ構成パラメータを変えながら実行することにより，プログラムの実行時間特性を調べた．得られた結果から，本研究の提案手法により仮想クラスタ移送時間の短縮が可能になった点を確認した．さらに，複数 VM ディスクイメージの平均サイズやデータの重複割合，使用可能な実効 WAN 帯域幅などの仮想クラスタの動作環境に応じて，仮想クラスタの移送時間を最短化する手法を選択するのがよいかの決定の指針を与えた．

今後の課題として，重複除去処理および復元処理に要する時間のさらなる解析が挙げられる．本研究の実験環境とは大きく異なる物理計算機環境においては，本研究で導出した数式を同様に用いた最適な移送手法決定が行えるとは限らない．しかし，提案したプログラムによる処理時間の大半は，ディスク IO によるものであると分かっている．そのため，重複除去を行う物理計算機のディスク読み込み速度およびディスク書き込み速度と，重複除去，復元処理に要する時間の関連を調べることが課題となる．これらを明らかにすることにより，本研究の実験環境とは大きく異なる物理計算機環境おいても，数式を用いた最適な重複除去手法の選択が可能になると考えられる．

謝辞 本研究の成果の一部は科研費 (23700058) の助成を受けたものである．

参考文献

[1] VMware: VMware Virtualization Software for Desktops, Servers, `http://www.vmware.com/`.
[2] Barham, P., Dragovic, B., Fraster, K., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and The Art of Virtualization, *In Proceedings of The Nineteenth ACM Symposium on Operatingsystems Principles*, Vol. 37, No. 5, pp. 164–177 (2003).
[3] Xen: Xen, `http://xen.org/`.
[4] Clark, C., Fraser, K., Hand, S. and Hansen, J.: Live Migration of Virtual Machines, *In Proceedings of The 2nd conference on Symposium on Networked Systems Design and Implementation*, Vol. 2, pp. 273–286 (2005).
[5] Hirofuchi, T., Ogawa, H., Nakada, H., Itoh, S. and Sekiguchi, S.: A Live Storage Migration Mechanism over WANfor Relocatable Virtual Machine Services on Clouds, *In Proceedings of The 2009 9th IEEE/ACM International Symposium on Cluster Computing and The Grid*, pp. 460–465 (2009).
[6] Ramakrishnan, K., Shenoy, P. and Van der Merwe, J.: Live Data Center Migration across WANs:A Robust Cooperative Context Aware Approach, *In Proceedings of the 2007 SIGCOMM workshop on Internet Network Management*, pp. 262–267 (2007).
[7] Van der Merwe, J., Ramakrishnan, K., Fairchild, M., Flavel, A., Houle, J., Lagar-Cavilla, H. and Mulligan, J.: Towards a ubiquitous cloud computing infrastructure, *The 17th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, IEEE, pp. 1–6 (2010).
[8] Stanoevska-Slabeva, K.: *Grid and cloud computing: a business perspective on technology and applications*, Springer Verlag (2009).
[9] Foster, I., Freeman, T., Keahy, K., Schefner, D., Sotomayer, B. and Zhang, X.: Virtual Clusters for Grid Communities, *In Proceedings of The Sixth IEEE International Symposium on Cluster Computing and The Grid*, Vol. 1, pp. 513–520 (2006).
[10] Keahey, K. and Freeman, T.: Contextualization: Providing One-Click Virtual Clusters, *In Proceedings of the 2008 Fourth IEEE International Conference on eScience*, Vol. 1, pp. 301–308 (2008).
[11] Al-Kiswany, S., Subhraveti, D., Sarkar, P. and Ripeanu, M.: VMFlock: virtual machine co-migration for the cloud, *Proceedings of the 20th international symposium on High performance distributed computing*, ACM, pp. 159–170 (2011).
[12] Deshpande, U., Wang, X. and Gopalan, K.: Live gang migration of virtual machines, *Proceedings of the 20th international symposium on High performance distributed computing*, HPDC '11, New York, NY, USA, ACM, pp. 135–146 (online), DOI: http://doi.acm.org/10.1145/1996130.1996151 (2011).