分散 PC グリッドシステムの開発
Development of Distributed PC Grid System

梅本潤志[†], 榎原博之[†], 大西克実[‡], 森川浩明[‡], 于文龍[†]
Junji Umemoto[†], Hiroyuki Ebara[†], Katsumi Onishi[‡], Hiroaki Morikawa[‡], and Bunryu U[†]

概要：PC クラスタをインターネット（WAN）上に分散配置した分散 PC グリッドシステムを実装し、その評価を行う。分散 PC グリッドシステムは、インターネット（WAN）上に分散配置されている PC クラスタをつなぎ、それらを 1 つのシステムとして機能するものである。PC クラスタ内の各 PC の状態を把握し、PC クラスタ間の情報交換のために、各 PC クラスタにグリッドエージェントと呼ぶ管理サーバを設置する。グリッドエージェントはグリッドポータルの機能も有し、各拠点で計算ジョブの投入やシステム内に存在する資源の状況把握などを行うことができる。

キーワード：グリッド、PC クラスタ、分散配置、システム構築
Key Words:Grid, PC Cluster, Distributed Allocation, System Implementation

1 はじめに

コンピュータのめざましい発展に伴い、家庭や会社で使われている PC でさえ高度な科学技術計算が可能となっている。これらの PC を複数台使えば、ある程度大規模の高速計算が可能であると思われる。さらに、それらの PC は 24 時間稼働しているわけではないため、それらの PC の遊休時間を効率的に利用すれば、安価で大規模計算が可能となる。

著者らが所属する研究室では、停止中や遊休の PC をグリッドシステムの計算サーバとして利用できるシステムを開発した [1]。このシステムは、大学内のコンピュータ演習室などにある LAN に繋がった PC の有効利用を想定しており、グリッドサーバが休止している計算機を見つけ、ネットワークを介して起動させる。さらに、マイグレーション機能によって演習室内でユーザが利用を開始すると計算を行っている仮想計算機をサスペンドさせ、計算内容を他の計算機に移行させる機能を持っている。この機能により、長時間ジョブの実行が可能である。

グリッドシステムの定義は、元来、電力系パワーグリッドに例えられるようにネットワークに接続するだけで、誰もが安価にあるいは無料で計算パワーを利用できるシステムである [2][3]。PC グリッドシステムは、家庭などで使われている PC を使って大規模計算を行うシステムと定義されている。しかし、現在のところ、SETI@home プロジェクト [4] に代表されるように、PC 側がサーバにあるソフトウェアをダウンロードしてバックグラウンドで実行するしくみで、PC 起動中に遊休状態になったとき、そのソフトウェアがサーバからデータを取得し、計算し、実行結果をサーバに返すしくみとなっている。このため、独立した小問題に分割できる問題にしか適用できず、プロセス間で通信を行う並列計算には向かない。

特に PC グリッドシステムは、高性能な PC がその性能をフルに発揮していない点を考慮し、PC の有効活用を目的として、手軽に計算パワーを得ることができるシステムを目指している。そこで、複数の拠点に散らばっている PC や計算資源を 1 つのシステムに統合、管理し、大規模計算のために効率的に利用するシステムが必要となる。これにより、ユーザは資源の情報をそれぞれ調べる必要がなく、より多くの使用できる資源を選択肢として増やすことができ、柔軟に活用することができる。各拠点に、管理や状況把握、通信の仲介を行うサーバを設置することで、各拠点で独立して計算を行うのではなく、それぞれ連携してネットワーク越しに並列計算を行うことができる。また、複数拠点の資源を統合して扱うために、計算機やネットワークを仮想化し利用す

† 関西大学システム理工学部
‡ 大阪市立大学創造都市研究科

ることで、インターネットを介して透過的にかつ動的に資源を扱うことが可能となる。

本研究では、PC クラスタをインターネット（WAN）上に分散配置した分散 PC グリッドシステムの実装とその評価を行う。実装は、関西大学理工学部アルゴリズム工学研究室内と、大阪市立大学学術情報総合センターのそれぞれに拠点を設け、インターネットを介して構築する。本システムでは、インターネット（WAN）上に分散配置された PC クラスタを管理サーバであるグリッドエージェントによってつなぎ 1 つのシステムとして機能させる。グリッドエージェント間で通信し情報を同期することにより、効率の良い計算資源配分と通信負荷の軽減を実現する。また、各グリッドエージェントにはグリッドポータルの機能を持たせ、各拠点で投入した計算ジョブは地点に関係なくかつ効率良く分散実行できる。評価は、分散 PC グリッドシステムの各機能についての性能を評価するために、実装したシステム上で計測を行う。また、その実測結果から得られたデータを用いて、さらに大規模にシステムを構築した際のシミュレーションを行い、システム全体の効率性を評価する。

本研究の構成は次の通りである。2 章では本研究で用いている技術である、仮想化や並列処理システムについて述べる。3 章では、提案手法である分散 PC グリッドシステムについての概要や特徴について述べる。4 章では、分散 PC グリッドシステムの実装や各機能の役割について述べる。5 章では、分散 PC グリッドシステムの評価を行うための計測手法とその結果について述べる。6 章では関連研究について述べ、7 章で結論について述べる。

2 関連技術

本章では、提案システムで利用する技術を中心に説明する。

2.1 SCore

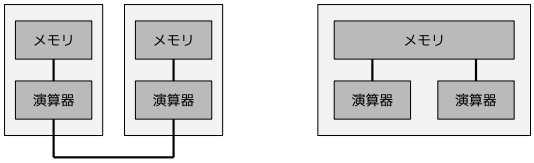
SCore[5] とは、技術研究組合新情報処理開発機構において開発された、PC クラスタのための HPC（High Performance Computing）型システムソフトウェアである。HPC クラスタでは、ネットワーク接続する計算機の台数を増やし、計算処理を各計算機に分散させることで全体の計算処理速度を高速にすることを目的としている。SCore は以下のような特徴をもっている。

- 高性能通信ライブラリ
- 効率の良いコンピュータ管理
- 高いユーザビリティ

SCore は独自に開発された PM2 高性能通信ライブラリを用いて並列計算を行っている。これにより、Ethernet でつながれた小規模クラスタから、Giga bit Ethernet や Myrinet[6] のようなネットワークで構築された大規模クラスタまで、シームレスに利用することが可能となる。OpenMP や MPI、MPC++ といった並列プログラミング環境も、これらの通信ライブラリの上で実装されている。

2.2 MPI

MPI（Message Passing Interface）[7] とは、メッセージパッシング方式に基づいた仕様で、図 1(a) のような分散メモリ型の並列計算を行うために複数のプロセス間でのデータのやりとりに用いられるライブラリである。メッセージパッシング方式とは、あるプロセスから他のプロセスへデータを明示的に送る方法である。各プロセッサ上で並列に動作するプロセスがそれぞれ独立したアドレス空間を持っている分散メモリ型のクラスタ環境では、並列化インターフェースとしてメッセージパッシング方式が用いられている。SCore では MPICH2[8] という形で、MPI が実装されている。



(a) 分散メモリ型　　(b) 共有メモリ型

図 1　分散メモリ型と共有メモリ型

2.3　OpenMP

OpenMP[9] とは、OpenMP Architecture Review Board (ARB) によって業界標準規格に規定された、図 1(b) のような共有メモリ型で並列計算を行うためのライブラリである。MPI ではプロセス間でのデータのやり取りをプログラム中に明示的に記述しなければいけないのに対し、OpenMP では逐次プログラムから段階的に並列化することが可能である。また、OpenMP をサポートしない環境では OpenMP を無効にし逐次処理で実行することができる。さらに、OpenMP で共有メモリ型で並列化されたものを、MPI を使用して分散並列化させることが可能である。

2.4　VMware

VMware[10] は、VMware 社の提供する仮想マシン環境構築用のソフトウェア製品である。製品には、VMware Workstation や VMware Infrastructure (VI)、VMware Server、VMware Player などがある。特に、無償で提供されている VMware Server と VMware Player は、アプリケーションタイプの仮想化ソフトウェアである。アプリケーションタイプとは、Windows や Linux などのホスト OS 上のアプリケーションの 1 つとして動作する。アプリケーションとして入れるだけで、仮想環境の構築が可能なため導入が容易であり、ハードウェアに依存することが少なく、さまざまな OS で動作することができるなどの長所を持つ。しかし、アプリケーションタイプは、仮想化部分がアプリケーションとして動作するため、仮想化のオーバーヘッドが高くなり、ゲスト OS のパフォーマンスは低くなる。

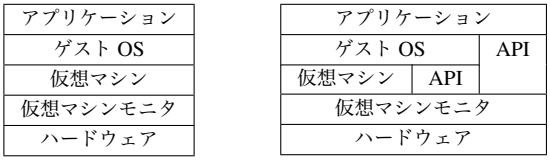
2.5　Xen

Xen[11] は、高い性能と機能を持つハイパーバイザタイプのオープンソースの仮想化ソフトウェアである。ハイパーバイザタイプとは仮想化層にハイパーバイザと呼ばれる小さなプログラムを動かし、すべての OS（ゲスト OS）はそのハイパーバイザ上で動作する。

ハイパーバイザタイプで仮想マシンを構築する上で用いる技術として完全仮想化と準仮想化があり、Xen では完全仮想化と準仮想化のどちらの実装手法も提供している。以下では、完全仮想化と準仮想化について説明する。

- 完全仮想化（Full Virtualization）
  実際のハードウェア用に提供されている OS を修正を加えることなく、そのまま Xen 上のゲスト OS として使用することができる。しかし、OS に修正を加えることなく利用するため、I/O 命令毎にハードウェアエミュレーションを行わなければ ならないため、仮想化のオーバーヘッドが大きくなり、エミュレーションの処理に コストを大きく裂かなければならない。
- 準仮想化（Para Virtualization）
  実際のハードウェアをゲスト OS のデバイスドライバからエミュレートすることで、仮想的なハードウェアを提供することができる。完全仮想化と異なり、ゲスト OS に修正を加える必要があるが、図 2(b) のように仮想化 API を利用することでオー バーヘッドを少なくすることが可能となり、高い性能を発揮することができる。

また、Xen の仮想マシン環境はマルチプロセッサにも対応し、各ドメインに対して資源割り当て量を



(a) 完全仮想化　　(b) 準仮想化

図 2　完全仮想化と準仮想化

動的に変更することができる。メモリ量も同様に動的に割り当てることができるので、資源を効率的に管理することが可能となる。

仮想化ソフトウェアによる仮想マシンの特徴として、仮想マシンはファイル（仮想ハードディスク）として扱われるため、仮想マシンを異なる物理マシン間で移動させる、マイグレーションを行えるといった特徴がある。特に、仮想マシンを動作させたまま移動させるライブマイグレーションという機能を用いることで、仮想マシン上で動作するアプリケーションを停止せず、サービスを提供し続けたまま、動作する物理マシンを変更することが可能である。

2.6　OpenVPN

OpenVPN[12] とは、インターネットや LAN 上に仮想的な VPN（Virtual Private Network）を構築することができるオープンソースソフトウェアである。VPN は、各拠点間に新たに専用のネットワークを構築するよりも容易に導入することができ、また既存のネットワークに追加して構築することができる。VPN の大きな機能として、トンネリングと通信パケットの暗号化が挙げられる。トンネリングによって仮想的に閉じたネットワークを構築し、カプセル化される際に通信パケットを暗号化することで、転送中のセキュリティを高めている。

OpenVPN では、暗号化には OpenSSL[13] を用い、秘密鍵や公開鍵を用いた相互の認証も使用することができる。イーサネット（レイヤ 2）通信を用いてのトンネリングやブリッジ接続、Broadcast 通信の転送も可能といった特徴を備えている。また、複数の OpenVPN のサーバを起動させることで、それぞれ独立したプライベートネットワークを構築することも可能である。

3　分散 PC グリッドシステム

3.1　グリッドシステム

グリッド（Grid）とは、辞書を引くと「格子、碁盤の目、送電網（Power Grid）」など と記述されている。グリッドシステムにおけるグリッドは、電力システムで用いられる送 電網に由来する。広義の意味でのグリッドシステム（Grid System）とは、「コンセントに差し込めばいつでもどこでも必要なだけ電力が得られるように、情報コンセントに接続 するだけで、コンピュータネットワークを介して、誰もが安全に（強固なセキュリティ）、安定して（必要な時に必要なだけ提供可能）、安易に（システム側を気にすること無く利用可能）、ネットワーク上のコンピューティング資源を利用できるシステム」と定義することができる。グリッド協議会のホームページ [14] によると、グリッドシステムの定義（狭義）は、「グリッドは、広域ネットワーク上の計算、データ、実験装置、センサー、人間などの資源 を仮想化・統合し、必要に応じて仮想計算機（Virtual Computer）や仮想組織（Virtual Organization）を動的に形成するためのインフラ」と記されている。

グリッドシステムは、コンピューティンググリッドやデータグリッドなどに分けることができる。

- データグリッド（Data Grid）
  大規模なデータを共有し、データに対するアクセスにグリッド技術を利用することで、物理分野や遺伝子解析分野などで大規模シミュレーションなどの計算を行う。データグリッドでは、扱うデータが大規模なこともあり計算規模が大きいものが多く、高速な処理性能を持つ高性能な計算機により大規模計算を行う。
- コンピューティンググリッド（Computing Grid）
  データ量がさほど大きなものでなくとも、長

時間の計算が必要なジョブがある。これらのジョブを計算資源を効率的に使用して処理する。

3.2 PC グリッドシステム

PC グリッドシステムとは、コンピューティンググリッドの中でも、遊休 PC を有効活用するシステムである。最も有名かつ最大のプロジェクトは、SETI@HOME プロジェクトである。このプロジェクトは、地球外知的生命体探査 (SETI) を行うもので、プエルトリコの天文台で収集された宇宙から届く電波を解析し、人工的に発信されたと思われる電波を検出するプロジェクトである。参加希望のボランティアは、SETI@HOME のホームページからソフトウェアをダウンロードし、PC にインストールすれば、簡単に参加できる。このソフトウェアは、スクリーンセーバーのように PC を利用していないときにデータを取り込み、解析を行う。現在、多くのボランティアを得て、トップクラスのスーパーコンピュータに匹敵するデータ処理を実現している。

PC グリッドシステムは上記の SETI 型グリッドシステムが一般的である。すなわち、 各 PC が自発的に計算処理に参加し、グリッドサーバは単にジョブの投入と結果の収集のみを行う。SETI 型 PC グリッドシステムの研究として、中部電力の曽山らの研究がある [15]。文献 [16][17] のシステムでは、グリッドサーバが能動的に遊休 PC を探索し、遊休 PC にジョブを投入する PC グリッドシステムについて研究している。

3.3 分散 PC グリッドシステム

本研究で実装する分散 PC グリッドシステムは、インターネット（WAN）上に分散配置されている PC クラスタをつなぎ、それらを 1 つのシステムとして機能するものである。PC クラスタ内の各 PC の状態を把握し、PC クラスタ間の情報交換のために、各 PC クラスタにグリッドエージェントと呼ぶ管理サーバを設置する。グリッドエージェントはグリッドポータルの機能も有し、各拠点で計算ジョブの投入やシステム内に存在する資源の状況把握などを行うことができる。

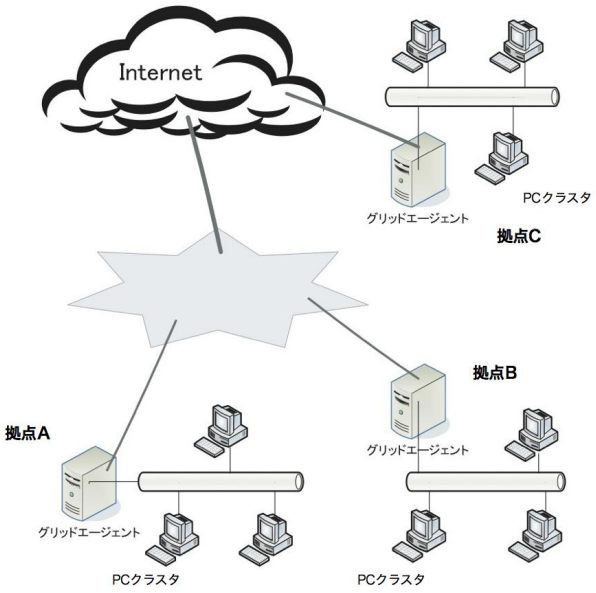


図 3　分散 PC グリッドシステム

具体的には図 3 に示すように、著者らが所属する関西大学システム理工学部アルゴリズム工学研究室内の 2 拠点（拠点 A、拠点 B）と、大阪市立大学学術情報総合センターの 1 拠点（拠点 C）の、計 3 拠点に PC クラスタとそれを管理するグリッドエージェントを設置し、グリッドエージェント間で通信することにより、効率の良い計算資源配分を実現する。

ユーザは、各拠点のグリッドエージェント（グリッドポータル）に計算ジョブを投入する。投入された計算ジョブは、グリッドエージェント間の各拠点の計算資源の空き情報の交換により最も適した環境で実行される。例えば、必要資源がローカルまたは遠隔の単一クラスタで確保できる場合は、ジョブを転送することにより実行し、複数クラスタの資源をあわせることで必要資源が確保できる場合は、仮想ネットワークを利用して複数クラスタを超えた実行により処理が行われる。ジョブは、投入された拠点の PC クラスタで実行されるとは限らず必要な資源を確保できる拠点内で実行される。さらに、1 拠点の PC ク

ラスタ内で資源が確保できない場合は、複数の PC クラスタをまたがった実行も可能である。この実行は、VPN を用いて個別のネットワークを構築するため、プロセス間通信の必要が無いパラメータスイープ型のジョブだけでなく、プロセス間通信が必要なジョブでも実行できる。ただし、複数の拠点での実行は、通信遅延などの問題が起こる可能性があるため、ユーザ側であらかじめ、PC クラスタを分けることができる範囲を指定し、PC クラスタ内通信と PC クラスタ間通信を区分し実行するか、ポータル側で複数の PC クラスタにまたがって実行を許可しない設定にしておく必要がある。

3.4 分散 PC グリッドシステムの特徴

提案する分散 PC グリッドシステムは、PC クラスタ間でのプロセス間通信を実現することにより、PC クラスタを超えた効率の良い計算資源配分が可能となる。さらに、PC の CPU がマルチコア化していることから、コア単位での PC の負荷状況を把握し、コア単位での計算ジョブの投入や PC 単位でのコア分散を考慮した並列計算ジョブの投入を可能とする。また、プロセス間通信は、暗号化によりセキュリティを保証する。

本システムは、仮想マシン上で構築するため、計算ジョブのマイグレーションが可能である。ジョブ実行中の PC を途中で中断させ、他の PC へマイグレーションさせることにより、他のジョブを実行させたり、ユーザが単に PC として利用したりすることが可能である。この機能により、大学などの計算機室の PC をオープン利用時でも計算サーバとして利用することが可能となる。結果として、長時間のジョブも実行可能である。

提案する分散 PC グリッドシステムの特徴を以下に列挙する。

- PC クラスタ
  各拠点が PC クラスタを持ち並列計算処理を行う。PC クラスタは仮想化ソフトウェアによって仮想計算機として構築されるため、動的に計算資源の割り当てを行うことができる。
- グリッドエージェント
  各拠点間の通信や、拠点内の PC クラスタ間での通信、複数クラスタ間での並列計算の通信を仲介する。ユーザのジョブの投入やシステム状況の表示などを行う、グリッドポータルも提供する。
- 仮想ネットワーク
  仮想的に隔離したネットワーク上で安全に通信することで、複数クラスタにまたがるジョブの実行やマイグレーションを行う。ネットワークを仮想的に構築できるため、各拠点内の物理マシンや仮想計算機同士の接続が可能となる。
- マイグレーション
  仮想計算機が仮想化システムによって構築されているため、物理マシン間で仮想計算を移行することができる。長時間にわたるジョブを実行するために、仮想計算機を途中中断することなく他の物理マシンに移行させる。
- スケジューリング
  各グリッドエージェントが保有する資源の空き情報を元に、ジョブに対して資源の最適な割り当てを行う。ジョブの転送や、資源情報の共有はグリッドエージェントが行う。
- グリッドポータル
  ユーザがジョブを投入するためのポータルとして機能する。どの拠点のグリッドポータルからでも、同じ情報を得ることができ、場所に制限されることなくジョブの投入を行うことができる。

これらの特徴を実現するために、次章で実装について述べる。

4　分散 PC グリッドシステムの実装

4.1　システム環境

分散 PC グリッドシステムの構成は、複数のグリッドエージェントがインターネット上に分散配置されており、各グリッドエージェントは Ethernet Switch を介して PC クラスタを構成している。各拠点の物理マシンの構成は表 1 に示す通りである。

表 1　各拠点の構成

	計算機	台数
拠点 A	OPIPLEX755	8 台
拠点 B	eX.Computer	12 台
拠点 C	OPIPLEX755	4 台

分散 PC グリッドシステムのそれぞれの機能の実装について以下に述べる。

4.2　PC クラスタ

各 PC クラスタの物理マシンには、仮想化ソフトウェアである Xen を利用し、準仮想化で仮想マシンモニタと仮想計算機の 2 つの仮想マシンを動作させる。Xen の準仮想化技術を利用することにより、エミュレーションのオーバヘッドを最小限に抑えることができ、各仮想計算機は物理ハードウェア上での動作時と遜色の無い性能を引き出すことが可能となる。Xen によって仮想化された各計算機上で並列計算が実行される。各仮想計算機には、OpenMP や MPI、MPC++ などの並列計算ライブラリが含まれた SCore をインストールして並列コンピューティング環境を構築する。各拠点内の PC クラスタを構成する仮想マシンのイメージは、図 4 に示すようにグリッドエージェントが保有し、適時物理的なネットワークを通して分散ファイルシステムを利用して、各 PC 上で仮想マシンが起動される。物理マシンに対して仮想計算機のイメージそのものを適時転送すると、ジョブを開始するまでの転送時間が大きくなるため、分散ファイルシステムにより仮想計算機の起動時間を短縮させる。

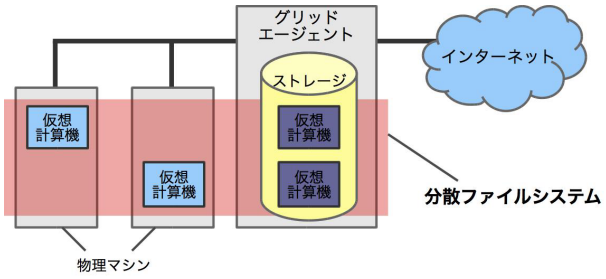


図 4　PC クラスタ

4.3　グリッドエージェント

管理サーバであるグリッドエージェントは、ローカル PC クラスタ内の各仮想計算機の計算資源やジョブの実行をコア単位で管理し、クラスタ内全体の計算資源を効率よく利用できるように計算資源の配分を行う。計算資源を配分するとき、各 PC クラスタ内で仮想計算機がマイグレーションするための空き PC を一定数確保することも考慮に入るようにする必要がある。グリッドエージェント同士は、クライアント-サーバ方式のような上下関係を持たずに、P2P（Peer to Peer）方式同様の対等な関係で密に通信を行い、各 PC クラスタの利用状況などの計算資源情報を常に共有する。グリッドエージェントとグリッドエージェント同士が各 PC クラスタ間の通信を仲介することで、複数クラスタ環境での単一ジョブの実行や計算ジョブの転送、仮想計算機のマイグレーションが可能となる。また、各グリッドエージェントへの外部からの通信は、必要最小限のポート以外の通信は遮断しているため、インターネットへも安全に接続することが可能である。また、グリッドエージェントは各拠点内の PC クラスタを構成する仮想マシンのイメージを保有し、ファイルサーバとしての役割も果たす。グリッドエージェントが仮想マシンのイメージを持つことで、グリッドエージェント間でイメージの受け渡しやイメージへのアクセスが

容易となり、さらにジョブを保留することができるため、インターネットを介したマイグレーションの実現が可能となる。

4.4 仮想ネットワーク

複数クラスタ環境での実行を実現するために、図 5 に示すようにそれぞれの拠点に存在する仮想計算機間を OpenVPN による仮想ネットワークによって接続し、物理的なネットワークとは異なる隔離されたネットワークを新たに構築し通信を行う。

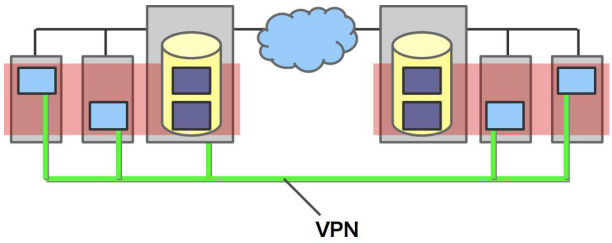


図 5　VPN

複数クラスタ環境で仮想ネットワークを新たに構築することで、物理的なネットワークを通して行われるクラスタ内通信に干渉されずに各計算機間で通信を行うことができる。また、計算ジョブのマイグレーション時には、仮想計算機を動作させる各物理マシンと、仮想マシンのイメージを保有するグリッドエージェント間に、並列計算の通信に用いる仮想ネットワークとは別に新たに仮想ネットワークを構築し、他のジョブの実行を妨げることなく仮想計算機のマイグレーションを行う。

4.5 マイグレーション

Xen のマイグレーション機能を使用することで、ジョブ実行中の仮想計算機を途中中断することなく他の物理マシンに移行させることで、引き続きジョブを行うことができる。図 6 のように、インターネットを介してクラスタ間に構築した VPN を用いることで、インターネットを介して仮想計算機のイメージを共有することができる。これにより、仮想計算機のイメージを持つグリッドエージェントの拠点内だけでなく他のグリッドエージェントが管理する PC クラスタ内の物理マシンへもマイグレーションが可能となる。これにより、システム全体の計算資源を効率的に使用することができる。

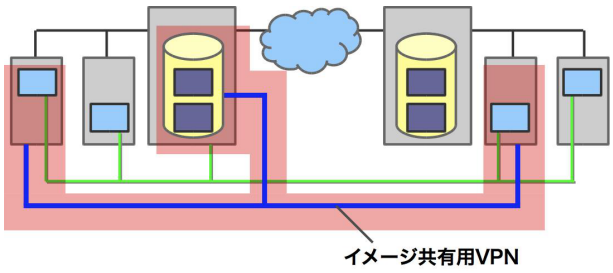


図 6　マイグレーション

したがって、複数の拠点間にまたがって実行している計算ジョブを、単一のクラスタ内で必要資源が確保でき次第、仮想計算機をマイグレーションによって単一の PC クラスタへと集約し、通信による遅延や影響を軽減することができる。仮想計算機のイメージをグリッドエージェントが保有しているため、仮想計算機がマイグレーション後は元の物理マシン上の仮想ネットワークを切り離すことができる。もし起動中の PC がすべて利用中でありマイグレーションできない場合も、使用していない仮想計算機を再起動したり、休止中の PC を Wake on Lan で起動することで新たにマイグレーション先を確保し、ジョブの実行が途中で終了してしまわないようにすることが可能である。

4.6 スケジューリング

ユーザから投入されたジョブは、各グリッドエージェント間で同期している PC クラスタの空き資源情報を元に、資源を確保し実行される。特に指定されない場合、複数の拠点にまたがる実行を許可せず、グリッドエージェントは指定された資源が確保できる拠点が存在するか調べる。投入された拠点以外の遠隔地の拠点で資源が確保できる場合、グリッドエージェントはジョブを遠隔拠点に転送し、遠隔拠点において実行する。しかし、要求された資源を確保で

きない場合は、必要な資源が確保できるまでキューに保存される。

ユーザが複数の拠点間での実行を許可している場合は、要求された資源を複数の拠点で確保できるか調べる。確保できると判明すると、各仮想計算機上で新たに仮想ネットワークを構築しジョブを実行する。

4.7 グリッドポータルとユーザビリティ

各グリッドエージェントは、リモートアクセスだけでなく図 7 に示すような Web ブラウザ上でのポータルの機能を持つ。グリッドポータルでは、使用する計算資源、例えば CPU コア数やメモリ量などを指定してジョブを投入する。グリッドポータルから投入されたジョブは、各グリッドエージェントが管理している PC クラスタの状況に合わせてジョブを実行する PC クラスタを決定し、使用する計算資源を確保する。このとき、ジョブの実行先はユーザがジョブを投入したグリッドエージェントが管理している PC クラスタ以外の遠隔拠点内の PC クラスタで実行されることもある。しかし、計算資源を 1 つの拠点で確保することができず、複数の拠点にまたがってしか実行環境を確保できない場合などは、インターネットを介した実行環境となるため通信遅延が起こる。そのため複数クラスタ環境の使用については、ユーザに許可された場合に限定する。その場合は、ユーザ側で資源に対してのジョブの分割範囲などを指定しなければならない。

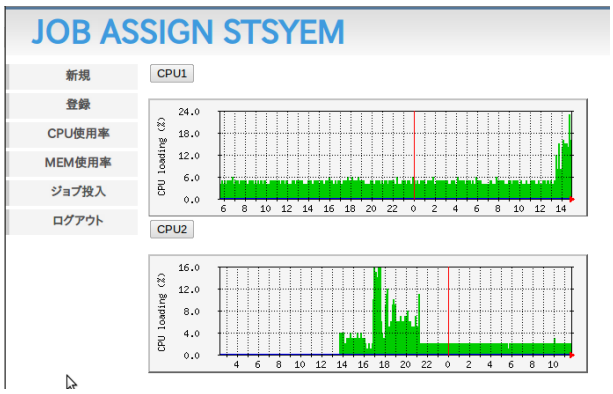


図 7　グリッドポータル

5　性能評価

5.1　性能評価における環境

各拠点内のグリッドエージェント及び各クラスタを構成する計算機の性能は表 2、3 に示す通りで、実装環境は、CentOS 5.6、Xen 3.1.3、SCore 7.0.1、OpenVPN 2.2.0 をそれぞれ用いて実装を行う。

表 2　各グリッドエージェントの性能

	Type	CPU	メモリ
GridAgent A	PE R300	Intel Xeon X5470 3.33GHz	8GB
GridAgent B	DL320G6	Intel Xeon E5606 2.13GHz	8GB
GridAgent C	PE840	Intel Xeon X3220 2.40GHz	8GB

表 3　各クラスタを構成する計算機の性能

	クラスタ	CPU	メモリ
Cluster A	OPIPLEX755 8 台	Intel Core2 E6850 3.00GHz	8GB
Cluster B	eX.Computer 12 台	Intel Core i5 2400 3.10GHz	8GB
Cluster C	OPIPLEX755 4 台	Intel Core2 E6850 3.00GHz	8GB

評価実験を行うシステム構成として、実際にシステム上で考えられる 4 パターンを挙げ、表 4 に示す。

- システム構成 1
  1 拠点：単一クラスタ内での実行
- システム構成 2
  2 拠点：学内 LAN 内の複数クラスタ間での実行

- システム構成 3
  2 拠点：インターネットを介した複数クラスタ間での実行
- システム構成 4
  3 拠点：インターネットを介した大規模環境における複数クラスタ間での実行

表 4　システム構成パターン

	PC 構成	ネットワーク
システム構成 1	OPIPLEX755 8 台	クラスタ内 LAN
システム構成 2	OPIPLEX755 4 台,4 台	学内 LAN
システム構成 3	OPIPLEX755 4 台,4 台	WAN
システム構成 4	OPIPLEX755 3 台,3 台,2 台	WAN, 学内 LAN

システム構成 1 は、通常の PC クラスタでの実行と同じように単一のクラスタ内だけで処理される。システム構成 2 は、2 つの拠点の各仮想計算機上に新たな仮想ネットワークを構築し、2 拠点間で並列処理を行う。システム構成 3 は、システム構成 2 と同様に 2 つの拠点間で構成される。システム構成 2 とシステム構成 3 の違いは、システム構成 2 は学内ネットワーク内などの物理的に近くに存在する 2 拠点を用いて構成されるのに対して、システム構成 3 はインターネットを介した物理的に距離の離れた 2 拠点で構成される。システム構成 4 は、2 拠点の場合と同様にして仮想ネットワークを構築し、多数の拠点間で並列処理を行う。

評価実験では、システム構成 1 は拠点 A の PC8 台、システム構成 3 では拠点 A、C の同性能の PC4 台ずつの計 8 台を使用して計測を行う。またシステム構成 2 とシステム構成 4 では、すべての拠点で同じ性能の PC を用いた同種環境になるように、拠点 A と拠点 B の計算機を配置し直し評価を行う。システム構成 3 は拠点 A、B の同性能の PC4 台ずつの計 8 台を使用し、システム構成 4 では拠点 A、B の PC3 台ずつと拠点 C の PC2 台の計 8 台を用いて、計測を行う。

5.2　評価実験 1：処理性能評価

単一拠点のみでジョブを処理した場合と VPN を用いて複数の拠点間にまたがってジョブを処理した場合において、VPN を利用することによる処理性能に与える影響を調べる。計測には、浮動小数点演算の実効性能を測定するベンチマークとして広く利用されている姫野ベンチマーク [18] を用いる。

姫野ベンチマークは、ポアソン方程式をヤコビの反復法で解く時の主要ループの処理性能を計測する。このベンチマークでは、通常コンピュータの性能を測り、複数の問題サイズが提供されている。サイズが大きくなるごとに扱うデータサイズが大きくなるため、メモリアクセスの性能の影響が大きくなる。本実験では、コンピュータの性能を測るのではなく並列処理について計測を行うため、並列計算機をサポートした MPI 版姫野ベンチマークを用いる。

計測環境は、システム構成 1、システム構成 2、システム構成 3、システム構成 4 の 4 パターンそれぞれで計測を行う。それぞれの環境で、姫野ベンチマークを MPI アプリケーションとして動作させ、処理性能の測定を行う。各性能評価では、問題サイズ L を使用し、10 回計測した平均値を示す。

計測結果を表 5 に示す。

表 5　各環境における処理性能

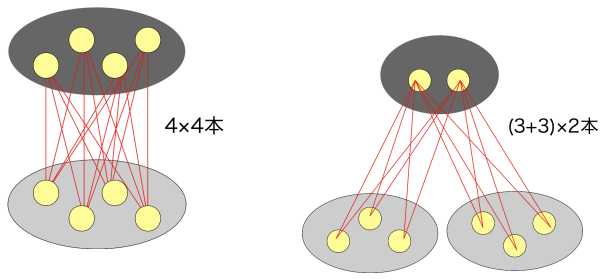
	MFLOPS
システム構成 1	1999.61
システム構成 2	2310.1
システム構成 3	292.44
システム構成 4	552.94

計測結果から、システム構成 1 の単一クラスタで実行した場合と比較して、システム構成 3,4 の VPN を利用し遠隔拠点との複数クラスタにまたがっての

実行時に、処理性能は大きく低下することが分かる。これは、物理的に距離が離れた PC 同士を、仮想的にローカルネットワークとして新たに構築したことにより、物理的な距離が大きく離れた PC 間でのプロセス間通信のネットワーク遅延の影響が大きくなり、性能が低下したと考えられる。このことから、投入されたジョブはできる限り単一クラスタで処理する方が効率的であることが分かる。

また、表 5 のシステム構成 1 とシステム構成 2 を比較すると、仮想ネットワーク上に PC クラスタを構築しているにも関わらず性能が向上している。これは、ジョブ実行中のプロセス間通信の仲介を行うグリッドエージェントが、システム構成 1 では 1 台であるのに対して、システム構成 2 では 2 拠点の 2 台のグリッドエージェントによって処理されるため、負荷が分散されたと考えられる。さらに、物理マシンのグリッドエージェントへの仮想計算機のイメージアクセスの負荷もあり、影響がより顕著に性能に出たと考えられる。

また、表 5 のシステム構成 3 とシステム構成 4 を比較すると、拠点数が増えているシステム構成 4 の方が性能が良くなっている。これは、システム構成 1 とシステム構成 2 の比較結果と同様にグリッドエージェントが 2 台から 3 台に増え負荷が分散されたことに加えて、図 8 に示すような拠点間の通信の数が関係してると考えられる。



(a) 2 拠点　　(b) 3 拠点

図 8　拠点間通信

遠隔地との通信の数は図 8(a) のように、2 拠点の場合拠点間通信は 4 × 4 の 16 本であるのに対して、3 拠点の場合の拠点間通信は (3+3) × 2 の 12 本となる。つまり 3 拠点の場合、近くの 2 拠点との通信頻度が増え、2 拠点における遠隔地との相互の通信頻度よりも 3 拠点における遠隔地との通信頻度の方が少ないためであると考えられる。したがって、複数の拠点が近距離に集中している場合や、離れた拠点が少数である場合は、拠点の分割数が多くなっても十分な性能を得ることができると考えられる。

5.3　評価実験 2：マイグレーションにおける影響

仮想計算機上で複数の拠点間にまたがりジョブが実行されているとき、各拠点におけるクラスタの実行状況によって複数の拠点間に散らばって配置されている仮想計算機を、マイグレーションにより 1 つの拠点に集約することで、VPN による影響を減らすことができる。仮想計算機が異なる拠点のクラスタへマイグレーションする際の、処理性能への影響を調べる。

本実験における仮想計算機のマイグレーションは、ランダムにジョブ実行中には 1 回もしくは 2 回、拠点 C から拠点 A に対してマイグレーションが必ず発生する。ジョブ実行時間と比較してマイグレーション時間が比較的長いため、計測時間の多くに影響を与えることになる。

実験には、評価実験 1 と同様に MPI 版姫野ベンチマークを用いる。また計測環境は、システム構成 3 の拠点 A と拠点 C の 2 拠点を用いて行い、評価実験 1 のマイグレーションを行わなかったシステム構成 3 での結果と比較を行う。計測結果を表 6 に示す。

表 6　マイグレーションがジョブ処理に与える影響

環境	MFLOPS
マイグレーションなし	292.44
マイグレーションあり	3.98

計測結果と評価実験 1 の結果から、VPN を用いた仮想計算機のマイグレーションによってさらに処理性能が低下していることが分かる。しかし、拠点内

に計算資源が不足したとしても、複数クラスタ間でのマイグレーション機能を利用することで、ジョブを引き続き実行することが可能であることが分かる。よって、資源が不足するたびに拠点間をマイグレーションを行うことで、計算時間が長時間に渡るジョブの実行も中断や停止することなく行うことができる。これにより、システム全体の資源を動的に扱うことが可能で、拠点に依存せず常に効率的にすべての計算資源を使用することが可能となる。

5.4 評価実験 3：メタヒューリスティクスアルゴリズムによる評価

本実験では研究室で開発されたメタヒューリスティクスプログラムを用いて、実際の並列プログラムの実用性について評価を行う。このプログラムは、巡回セールスマン問題（Traveling Salesperson Problem:TSP）を解くためのアルゴリズムである。TSP とは、n 個の訪問対象の都市を一度ずつ訪問し初めの都市に戻る巡回路の中で、都市間の移動コストの総和が最小のものになる巡回路を求める問題のことである。このアルゴリズムの特徴は、解情報などを PC 間で共有する上で、複数の PC でグループを形成し、グループ内では頻繁に通信を行い解の改善を行っていく。それに対して、各グループ間での通信頻度は少なくすることで、通信遅延の影響を少なくしている。本実験では、4 グループ、各 2 台をそれぞれの拠点に分けて実行を行う。また解の探索は、各問題の最適解との誤差が 4% 以下の解が見つかるまで行い、探索にかかった時間を計測する。

計測対象の問題は、TSP のベンチマーク問題を配布している TSPLIB[19] より入手し、都市数が 318 の *lin*318、都市数が 575 の *rat*575 の問題を計測に用いる。また計測環境は、システム構成 1 とシステム構成 3 により行う。

各計測結果を表 7、および表 8 に示す。

計測結果から、単一クラスタを用いるシステム構成 1 と比較して、2 拠点間で実行されるシステム構成 3 での処理の方が探索にかかった時間が長く、性能が低下していることが分かる。表 7 および表 8 から、

表 7　メタヒューリスティクスアルゴリズムの計測（*lin*318）

環境	システム構成 1	システム構成 2
経過時間（秒）	70.8	116.9

表 8　メタヒューリスティクスアルゴリズムの計測（*rat*575）

環境	システム構成 1	システム構成 2
経過時間（秒）	339.2	363

システム構成 3 での処理がシステム構成 1 で実行したものと比較して、*lin*318 ではおよそ 60%、*rat*575 ではおよそ 90% の性能を得られていることが分かる。これは、評価実験 1 では 15% の性能しか得られていなかったことと比較すると、大きく改善されている。これは、グループ間通信とグループ内通信の頻度とグループの形成状況が影響したと考えられる。

グループ内通信の通信頻度はかなり多いが、グループが拠点内の PC で形成され複数の拠点にまたがって形成されていないことに加えて、複数の拠点にまたがって形成される場合が多いグループ間での通信は、通信頻度が少なくまた通信量も少ないように作られたアルゴリズムであったことが大きく作用したと考えられる。したがって、インターネットを介した際の通信遅延や、複数拠点間でのジョブの分割を考慮に入れたプログラムである場合、複数拠点間におけるコストを減少させることができ、十分な性能を得ることができることが分かった。

5.5 評価実験 4：ジョブ処理にかかる評価

システム全体においてのジョブ処理にかかる総時間を、シミュレーションにより評価する。

ジョブはランダムな仮想計算機数を使用し、処理にかかる時間もランダムに設定される。このようなジョブを一定数処理するまでにかかる総時間を計測する。シミュレーションでは、システム構成 1 のみ

で実行を行う単一クラスタ環境と、すべてのシステム構成で実行が行える複数クラスタ環境の 2 パターンそれぞれで計測を行う。

複数クラスタ環境では、2 拠点以上の拠点にまたがっての実行を許可しているのに対して、単一クラスタ環境では 1 拠点で必要な計算機を確保する場合のみで実行を許可する。つまり、単一クラスタ環境では、分散された PC クラスタを個々に利用している場合と同様の使用方法である。単一クラスタにおいて、必要な計算機の数を確保できない場合は、ジョブをキューに保存して計算機の空きが出るまで待機する。それに対して複数クラスタ環境では、1 拠点で必要な計算機を確保できない場合、複数の拠点で計算機を確保しシステム構成 3 やシステム構成 4 の場合と同様にしてジョブを実行する。全拠点内の計算機をあわせても必要な計算機を確保できない場合には、ジョブをキューに保存して空きが出るまで待機する。

本シミュレーションにおいて、複数拠点間でのジョブに対する影響などのパラメータは、実測データに基づいて設定を行う。各パラメータを表 9 に示す。

表 9　シミュレーションパラメータ

投入ジョブ数	1000
拠点数 (各拠点)	50
計算機数	10
平均ジョブ処理時間	178
ジョブの平均使用計算機数	6.8
拠点間通信コスト係数	3.5
拠点間負荷コスト	87

表 9 における拠点数と計算機数は、シミュレーションにおけるシステム全体の拠点数と、各拠点の計算機数を示している。投入ジョブ数は、各拠点に一定間隔で投入されるジョブの総和である。平均ジョブ処理時間は、1 つのジョブが実行を終えるまでにかかる時間の平均を示し、ジョブの平均使用計算機数は、1 つのジョブが使用する計算機の数の平均を示している。ジョブの処理時間と使用計算機数は、ジョブ生成時にランダムで設定される。ただし、処理時間の下限値は 40 で上限値は 200、使用計算機数の下限値は 2 で上限値は 10 である。拠点間通信コスト係数は、複数の拠点で仮想ネットワークを用いて複数クラスタ環境を用いた場合のコストであり、この値は評価実験 1 のシステム構成 1 とシステム構成 3 のデータを元に設定している。この値は拠点間での通信遅延にかかる値のため、複数クラスタ環境においての拠点数に依存し、拠点数が n の時の拠点間通信コスト係数は $3.5 \times (n-1)$ となる。拠点間負荷コストは、複数クラスタ環境においてマイグレーションを行った時のコストであり、この値は評価実験 2 のデータを元に設定している。拠点間負荷コストは、ジョブをマイグレーションする際にかかる時間であるため、マイグレーションが発生する場合はそのジョブの処理時間にこのコストの値を加える。マイグレーションを行うジョブは、複数クラスタ環境において投入されたジョブの処理時間が 160 を超え長時間になるものが対象である。また発生条件は、ジョブの終了時間までに実行中の計算機を単一の拠点に集約できるだけの空きができ、かつマイグレーションが完了するまでにジョブが終了してしまわないジョブに限定している。

計測結果を表 10 に示す。

表 10　ジョブ処理にかかる時間

環境	単一クラスタのみ	複数クラスタを許可
経過時間（秒）	2877.3	2521.1

計測結果から、単一拠点での実行のみを許可した場合と比較して複数拠点間での実行を許可した場合の方が、全体のジョブを処理するために必要な時間が短縮されいていることが分かる。これは、複数クラスタ環境では利用可能な資源通信遅延等の影響が発生するが、それらのコストの総和よりも複数拠点にまたがる実行を行う方が、システム全体としては効率的であったということを示している。つまり、各拠点内にそれぞれ少数で使用されていない計算資

源がある場合、それらを統合し利用することで、ジョブの実行時間が短縮され、システム全体をより効率的に使用することに繋がると考えられる。

6 関連研究

仮想クラスタを研究しているものには、Michael Fenn らの研究 [20] がある。この研究では、LAN 内に仮想計算機のイメージを保有する管理サーバと KVM[21] を用いて仮想クラスタを構築している。管理サーバには Condor[22] が導入されており、ジョブのプールやスケジューリングなどを行っている。性能評価では、KVM で実装を行った場合と Xen を用いて実装を行った場合のそれぞれを、物理的な計算機で実行した性能と比較している。性能評価の結果から、Xen と KVM どちらの場合でも物理的な計算機で実行した場合と遜色の無い性能が得られていた。しかし、この研究では計算機として仮想計算機を用いることが前提とされており、柔軟的にシステムを構築できるとは述べられているが、ジョブ実行途中でのマイグレーションについては述べられていない。

自律的にクラウド上にグリッドシステムを構築する手法を研究したものに、Murphy M.A らの研究 [23] がある。この研究では、仮想ネットワークを利用してインターネットを介して仮想クラスタを提供するシステムを提案している。仮想計算機の構築には、KVM と Xen を利用し、それぞれについて実測とシミュレーションによって評価を行っている。しかし、仮想計算機のマイグレーションなどの動的再配置については述べられていない。

大規模計算を行うために、複数の拠点を統合し仮想クラスタとして管理するシステムを実装したものに広測らの研究 [24] がある。この研究では、WAN 上の複数拠点に分散された計算機資源を、仮想ネットワークを用いて単一の仮想クラスタに構築するシステムを実装してる。各計算機は VMware Server により仮想的に構築され、仮想計算機の構築時にパッケージキャッシング機構により動的にパッケージを導入することで、ユーザの要求に合わせたシステムの構築が可能となっている。この研究では、複数の拠点を独立して使用せず、単一のシステムとして用いることを主眼においた研究となっている。また、拠点間をまたいだ仮想計算機の配置や、マイグレーションを利用した仮想計算機の再配置による柔軟性の向上などは今後の検討課題として挙げられている。

また、仮想計算機や仮想ネットワークを利用したクラスタシステムについて研究しているものに、Wang.L[25] や、山形 [26]、中尾 [27] らの研究などもある。

7 おわりに

分散 PC グリッドシステムを利用することによって、インターネット上に分散している複数の PC クラスタをつなぎ 1 つのシステムとして機能させることができ、ユーザは実行先を考慮することなくジョブを投入することが可能になった。そして、グリッドエージェントがそれぞれの PC クラスタの計算資源情報及び利用状況を管理・把握し、グリッドエージェント同士で情報を共有することでジョブの投入先を選定し、複数クラスタをまたぎ統合的に資源管理をすることが可能となり、効率的な計算資源配分をすることができる。また、計算機を仮想環境で構築することから、すべての遊休 PC を計算資源として活用することもでき、マイグレーション機能によって計算ジョブをクラスタ内および遠隔拠点に移動させることが可能となり、長時間ジョブの実行による資源の占有にも対応することができる。ユーザは各サーバにアクセスするのではなく、グリッドエージェントのポータル機能を用いることで、視覚的にシステムの利用状況を把握し、ローカル PC クラスタの利用状況によらず計算資源を確保しジョブを投入することができる。

また、評価実験から仮想ネットワークを用いた複数クラスタ環境では性能が低下することが分かった。しかし、複数クラスタ環境におけるジョブの分割方法や、実行する仮想計算機の配置方法によって、性能の低下を抑えることが可能であることも分かった。また、ジョブ実行中の仮想計算機をマイグレーションすることにより、計算機を単一クラスタへ集

約し再配置させることによって、ジョブの通信オーバヘッドを低減させることができる。これによって、処理に長時間必要なジョブも中断することなく実行し続け、よりよい環境へと変化しながら実行可能となった。

シミュレーション評価実験から、大規模にシステムを構築し大量のジョブが投入された際にも、単一クラスタのみで実行を許可するよりも、仮想ネットワークにより構築された複数クラスタ環境を用いることで迅速に効率的に処理することができることも分かった。したがって、分散配置をユーザが指定したり、非効率的な配置を避けるようにスケジューリングを向上させることによって、クラスタ内で使用されていない計算資源をより効率的に利用できると考えられる。

参考文献

[1] 森川ほか. 仮想計算機を適用した PC グリッドの開発と性能評価 (システム開発論文)(情報・システム基礎). 電子情報通信学会論文誌. D, 情報・システム, Vol. 93, No. 8, pp. 1555–1566, 2010.

[2] The Grid 2, 12 2003.

[3] 合田憲人, 関口智嗣. グリッド技術入門 - インターネット上の新しい計算・データサービス, 12 2007.

[4] SETI@home. http://setiathome.berkeley.edu/.

[5] PC Cluster Consorium. http://www.pccluster.org/.

[6] Myrinet. http://www.myri.com/.

[7] MPI. http://www.mpi-forum.org/.

[8] MPICH. http://www.mcs.anl.gov/research/projects/mpich2/.

[9] OpenMP. http://openmp.org/wp/.

[10] VMware. http://www.vmware.com/.

[11] Xen. http://www.xen.org/.

[12] OpenVPN - Open Source VPN. http://openvpn.net/.

[13] OpenSSL: The Open Source toolkit for SSL/TLS. http://www.openssl.org/.

[14] グリッド協議会. http://www.jpgrid.org/index.html.

[15] 曽山豊. 企業におけるグリッド・コンピューティングの活用とその成果. http://www.jpgrid.org/event/2006/pdf/gw2006_B2-4soyama.pdf, 2006.

[16] 榎原博之, 森川浩明. 仮想計算機を用いた pc グリッドの開発. RCSS ディスカッションペーパー 第 79 号, 2009.

[17] 森川浩明, 榎原博之, 大西克実, 中野秀男. 仮想計算機を用いたジョブマイグレーションの pc グリッドへの適応. 数理モデル化と問題解決（MPS）研究報告, Vol. MPS73, No. 5, pp. 17-20, 2009.

[18] 姫野ベンチマーク. http://accc.riken.jp/HPC/HimenoBMT.html.

[19] TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[20] M. Fenn, M.A. Murphy, and S. Goasguen. A study of a KVM-based cluster for grid computing. In *Proceedings of the 47th Annual Southeast Regional Conference*, p. 34. ACM, 2009.

[21] Main Page - KVM. http://www.linux-kvm.org/.

[22] Condor Project Homepage. http://research.cs.wisc.edu/condor/.

[23] M.A. Murphy, L. Abraham, M. Fenn, and S. Goasguen. Autonomic clouds on the grid. *Journal of Grid Computing*, Vol. 8, No. 1, pp. 1–18, 2010.

[24] 広渕崇宏, 中田秀基, 横井威, 江原忠士, 谷村勇輔, 小川宏高, 関口智嗣. 複数サイトにまたがる仮想クラスタの構築手法. 第 6 回先進的計算基盤システムシンポジウム SACSIS, pp. 333–340, 2008.

[25] L. Wang, G. von Laszewski, M. Kunze, J. Tao, and J. Dayal. Provide virtual distributed environments for Grid computing on demand. *Advances in Engineering Software*, Vol. 41, No. 2, pp. 213–219, 2010.

[26] 山形育平, 高宮安仁, 中田秀基, 松岡聡. グリッド上における仮想計算機を用いたジョブ実行環境

構築システムの高速化. 情報処理学会研究報告, pp. 127–132, 2006.
[27] 中尾, 昌広, 廣安, 三木, 光範, 吉見真聡. 仮想クラスタの構築と性能評価. 同志社大学理工学研究報告, Vol. 50, No. 4, pp. 20–24, 2010.