仮想計算機を用いたグリッド上でのMPI実行環境

立薗真樹[†] 中田秀基[††,†] 松岡聡[†,†††]

近年、大規模なMPIアプリケーションの実行をグリッド上で行うことが求められている。本稿では仮想計算機を用いたグリッド環境上でのMPI実行環境を提案する。仮想計算機Xen上で、MPI実行環境をもつゲストOSごとマイグレーションすることにより、ファイルI/Oや送信過程のメッセージを含めた計算機全ての状態が変更されない、MPIプロセスに透過的なマイグレーションを実現し、実行中のMPIのマイグレーションが正常に行われたことを確認した。さらにVPNを用いたネットワークの仮想化によって、マイグレーション先を異なるネットワークへ拡張し、また動的にマイグレーションを行うシステムの実装によりグリッド環境への対応を実現した。構築した環境における実行性能を評価では、仮想化によって生じたネットワークのオーバーヘッドにより、通信頻度によって大きく性能差が出る結果となった。

MPI Environment on Grid with Virtual Machines

Masaki TATEZONO ,[†] Hidemoto NAKADA [††,†]
and Satoshi MATSUOKA [†,†††]

Recently, a large-scale MPI application is requested to be executed on the Grid. We propose a MPI environment on the Grid with Virtual Machine Monitor Xen. The key idea here is that transparent migration of a MPI process running on a virtual machine would be made possible by moving the underlying virtual machine itself. Furthermore, enabling migration to other networks and implimenting a system which decided to migrate a guestOS automatically, we extend the MPI environment to Grid.On the test bed we constracted, we experimentally show that the overhead due to virtualization is large in network-intensive applications, but CPU-intensive application is performance nearly equal with Native environment.

1. はじめに

現在、並列プログラミングライブラリであるMPIは科学技術計算において高いシェアを誇っており、MPIを用いるプログラムは実行に長時間要するものも少なくない。今日、このような多くの計算力を要求するアプリケーションは、計算資源が単一サイトのみならず、複数サイト間にまたがるグリッド環境での実行が一般的になりつつある。グリッド環境、とくに遊休計算資源利用を目的とした環境においては、特定のユーザーが使用できる計算資源が変化することを想定せねばならなず、長時間にわたってMPIを実行する場合、すべての計算資源が計算終了時まで確保できるとは限らない。通常、MPIプログラムは他のノードとの通信を含んでいる。そのため大部分のMPIアプリケーションでは、並列化された中の一つのプロセスでも失われれば計算全体に影響を及ぼし、場合によっては一から計算をやり直すという状況も考えられる。したがって並列化された計算全体を失わないために、計算続行に問題のあるノードのMPIプロセスを他のノードへマイグレーションすることが必要である。

従来、マイグレーションの実現方法として、計算をプロセス単位で他のノードに移動するプロセスマイグレーションが多く用いられてきた。しかし、プロセスマイグレーションは、PIDの変更や、プロセス以外にファイルディスクリプタ、ソケットなどの保存が必要、かつ中間ファイルの生成時などに問題が生じるなど、実装上の障害が多い。

本稿では、仮想計算機Xen[1)]とVirtual Private Network(以下、VPN)を用いて、低コストなマイグレーションを異なるネットワークにまたがるサイト間で実現可能とし、グリッド環境、特に複数サイトにおいて計算機の遊休時間を利用するような形態に適応したMPI実行環境の構築を提案する。具体的には、プロセスマイグレーションに替わるマイグレーション手段として、仮想計算機Xenを用いて、MPI計算が実行されている仮想計算機上のゲストOSごとマイグ

† 東京工業大学
Tokyo Institute of Technolory
†† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
††† 国立情報学研究所
National Institute of Informatics

レーションを行う。これにより、MPI が実行されている計算機の内部状態に変更が加えられず、MPI プロセスに透過で低コストなマイグレーションが可能となり、実際に動作中の MPI のマイグレーションを行い、計算の継続および正常な終了を確認した。この手法では既存の MPI 実装を用いることができ、グリッド環境におけるユーザー透過な MPI アプリケーションの実行を実現できる。さらに VPN を用いることによって、仮想的にすべての全てのサイトの計算資源を同じネットワークとして扱うことが可能となり、上で述べた Xen によるマイグレーションにおいても複数サイト間で実行可能となる。

この低コストなマイグレーションを用いて、計算資源の監視を行い、遊休計算機へのマイグレーションを動的に行うシステムのプロトタイプ実装を行い、その動作を確認した。

また Xen および VPN を使用することでの計算性能への影響の評価を行った。その結果、通信頻度が低いアプリケーションではその有用性が確認された一方で、ネットワーク通信に起因するオーバーヘッドが大きく、通信頻度が高いアプリケーションにおいて、通常予想されるような通信のバンド幅の不足だけではなく、性能の台数効果が得られないという現象を確認した。その原因として OS 切り替えによるノード間の MPI プロセスの同期が乱れるのが原因であると予想しており、アプリケーションの通信タイミングの取得を行っており、これを Xen の OS 切り替えログと比較して、通信への影響を実証すべく追試実験を行っており、改善の方策を探っていく予定である。

2. Xen による OS マイグレーション

本研究では、MPI のマイグレーションをプロセス単位ではなく仮想計算機のゲスト OS をマイグレートすることで行う。この節では、使用する仮想計算機 Xen について述べる。

2.1 Xen Virtual Machine Monitor

Xen は英ケンブリッジ大学で開発された Virtual Machine Monitor(以下 VMM) であり、ハードウェアのリソースを仮想化し、複数の OS を同時並行的に動作させることが可能となっている。これはハードウェアを完全に仮想化するのでなく、各ゲスト OS が Xen の VMM 上で動作するようにカーネルを改変することによって実現される。現在 Linux2.4 系、2.6 系、NetBSD が安定して動作可能である。

Xen 環境の計算機を起動すると、Domain0 と呼ばれる OS が起動する。この OS もゲスト OS と同様に Linux 等が用いられる。Domain0 から管理ツールによって Domain1,2,3... と複数の OS を順次立ち上げることが出来る。Xen の構造上、Domain0 と Domain1 以降は並列的な位置づけにあるが、本稿では便宜上 Domain0 を「ホスト OS」、Domain1 以降を「ゲスト OS」と呼ぶ。

2.2 ゲスト OS のマイグレーション

Xen の VMM 上のゲスト OS は、他のマシン上で動作する Xen ホストに OS を丸ごとマイグレーションすることが可能である[2)]。このマイグレーション機能は、ホスト OS からマイグレーションを行うゲスト OS、行き先の Xen ホストを指定することによって、特定のゲスト OS を任意のノードへマイグレーションすることが出来る。これは対象ゲスト OS のメモリイメージ、レジスタ内容などの内部状態をネットワーク経由で転送することで実現される。このときディスクイメージの転送は行われないため、マイグレーション先のホストの同じ位置に root ディスクイメージが必要となる。転送先は同一のネットワーク内であれば、マイグレーション後も OS の設定を変更することなくネットワークの利用が可能となる。

3. 仮想計算機を用いた遊休計算機グリッド対応な MPI 環境

3.1 要　　件

グリッド環境上の遊休計算機における MPI の実行を実現するためには、次の要件を満たす必要がある。

(1) MPI 実行中でも必要に応じてマイグレーションが可能
(2) 異なるネットワーク上の資源にも透過的にマイグレーションが可能
(3) ユーザーに環境を意識させない MPI 実行形態の提供

第一項について、遊休計算機の利用では、計算資源がいつ使用可能か不確定であり、遊休状態で無くなった場合に、実行中の計算プロセスは即座にその計算機から退避する必要がある。この場合、状態を保存し再び遊休状態になった場合に再開する、別の遊休状態のマシンへマイグレーションする、別の遊休状態のマシンで計算をやり直す、という選択肢がある。しかし一般的な MPI ではプロセス同士の通信が伴うため、一つのプロセスのみ休止したりやり直すことは計算全体の進捗に影響を与える。したがってこのような場合に MPI プロセスが実行中であっても、低コストなマイグレーションを行うことが必要となる。

第二項について、本稿のターゲット環境としては複数サイト間をネットワークで接続した遊休計算機を中心としたグリッド環境である。したがってネットワークの異なるサイト間においても第一項で述べたマイグレーションを実現する必要がある。また資源の増減がサイト単位で起こる可能性があり、マイグレーションがサイト内で完結するシステムでは対応出来ない。

第三項について、MPI の実行にあたって、複数サイトからの計算資源選択などを不要とし、ユーザーが

特別にグリッド環境での実行を意識することなく、通常のクラスタ環境などと同様に実行できることが必要となり、それを実現するためのスケジューラが必要となる。

3.2 提　　案

上で述べた要件を満たし、グリッド環境上の遊休資源での実行を目的とする MPI 実行環境の提案を行う。

3.2.1 Xen による低コストなマイグレーション

MPI の実行ノード群を、前節で述べた Xen 上のゲスト OS のみで構成することによって、MPI プロセスのマイグレーションの必要性が生じたときに、実行するゲスト OS のマイグレーションを行う。マイグレーション時のゲスト OS のダウンタイムは平均的に数秒程度である。これは一般的な TCP などのタイムアウト時間より短く、また MPI で制御用に用いられる SSH,RSH のコネクションは維持される。

3.2.2 VPN による広域分散資源へのマイグレーション

仮想的なネットワークを提供する VPN を用いることによって、異なるネットワークで運用される複数サイトの計算機を共通のネットワークとして扱うことが出来る。このとき VPN サーバーは、すべてのサイトからアクセス可能な場所に設置し、各サイトの計算機はすべて、この仮想的なネットワークに参加する。図 1 に示すように、Xen 上のゲスト OS はこのネットワークアドレスのみを持ち、マイグレーションによってホストが変更されても継続してネットワークが利用出来る。

3.2.3 ユーザー透過な実行環境

MPI の実行は各ゲスト OS で一般的な MPI 実装を用いて行われる。したがってユーザーの実行に際して特別なライブラリのリンクなどは不要であり、透過的にグリッド環境での MPI 実行が可能となる。また遊休資源の発見・監視を行うことで最適な計算機へ動的にマイグレーションを行う資源スケジューラにより、ユーザーはどのサイトのどの計算機上で自分のジョブが実行されているかを意識する必要がない。

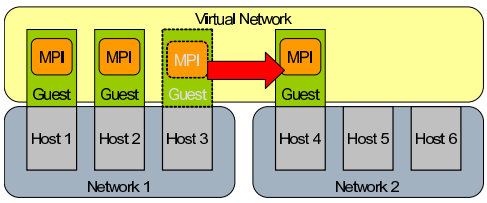

図 1　VPN を利用した他のネットワークへのマイグレーション

4. 仮想計算機を用いたグリッド上での MPI 実行環境の実装

前節で提案した設計にしたがって、Xen3.0 および OpenVPN[3] を用いて、グリッド上の遊休計算機利用へ向けたテストベッドの実装を行い、その妥当性を検証し性能評価を行った。

4.1 OpenVPN の利用

OpenVPN は TUN/TAP デバイスを用いたサーバークライアント型の VPN である。実装環境では各計算ノードのホスト OS が VPN の仮想 NIC を持ち、そこに割り当てられた仮想ネットワークに対して、ゲスト OS のネットワークがブリッジ接続する。これにより、ゲスト OS 同士は異なるサイト間でも同一のネットワーク空間をもち、マイグレーションによりホストが変更されてもネットワーク接続が維持される

また、VPN クライアントの認証には PKI を、通信には任意の暗号化アルゴリズムを用いることが出来るため、グリッド環境におけるサイト間通信においても信頼性を損なうことなく仮想ネットワークが構築可能である。

4.1.1 ゲスト OS のディスクイメージの共有

Xen では、マイグレーションを実行する際の条件として、ゲスト OS の起動時に指定した root ファイルシステムがマイグレーション先にも必要である。root ファイルシステムには物理ディスクの他に、イメージファイルの指定も可能である。提案システムでは、低コストなマイグレーションの実現のため、root ディスクイメージをネットワーク経由ですべてマイグレーション先に転送するのは現実的ではない。そこで、本実装では root ディスクイメージを NFS により共有することによって、各ノードからの同一のアクセスを可能とし、転送を不要としたが、無論 Gfarm[12] などのグリッドファイルシステム等も活用可能である。

4.2 動的なマイグレーションを行う資源監視システム・スケジューラ

低コストなマイグレーションを用いて、任意の条件を満たした場合に他の計算機へのマイグレーションを動的に行うシステムのプロトタイプ実装を行った。システム概要を図 2 に示す。システムを構成する要素は、セントラルマネージャと、各計算ノードのホスト OS で動くデーモンプロセスである。

計算ノードのデーモンプロセスは、ノード上に存在するゲスト OS の CPU 使用率、メモリ使用量等をを収集し、セントラルマネージャに送信する。セントラルマネージャは各計算ノードから集めた情報を管理し、各項目が定められた基準を満たしたノードを遊休ノード、またはビジーノードとする。遊休ノード、ビジーノードがそれぞれ存在し、ゲスト OS の使用メモリ量が総物理メモリを超えない場合、遊休ノードからビ

ジーノードへのマイグレーションの実行を決定する。決定されたマイグレーション情報はビジーノードに送信され、そこでゲスト OS の遊休ノードへのマイグレーションを行うコマンドを実行する。

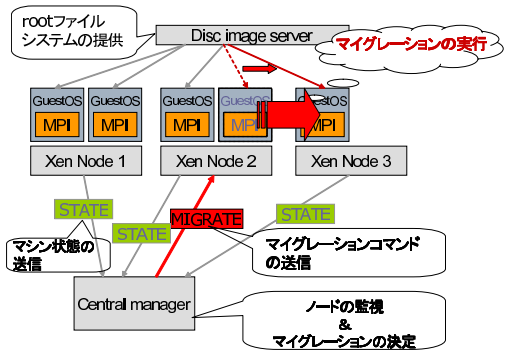

図 2 仮想計算機を利用したマイグレーション可能な MPI 実行環境

4.3 マイグレーション決定のポリシー

実装したプロトタイプシステムでは、マイグレーション目的を負荷分散、遊休計算機利用の二つのケースに分類し、簡単なポリシーを設定した。

4.3.1 負荷分散が目的の場合

負荷分散を目的とした場合、計算中のゲスト OS の数が各計算ノードで均一に近づくようにマイグレーションを行う。

4.3.2 遊休計算機利用が目的の場合

遊休計算機利用は計算機を所有者の利用していない期間に、別の利用者が計算に利用することである。したがって本来の所有者のジョブに影響を与えない利用が原則である。そこで、計算機の所有者はホスト OS もしくは、決まった一つのゲスト OS を使用する。そして、遊休時間機利用者は新たなゲスト OS を立ち上げ利用する。所有者のジョブが再開されたとき、つまりホスト OS または特定のゲスト OS の CPU 使用率が上昇したことを検知した場合、即座に遊休時間利用者のゲスト OS を他の遊休ノードへマイグレーションする。

4.4 スイッチングハブへの対応

スイッチングハブは必要のないパケットの送出を防ぐため、ある MAC アドレスがどのポートへ接続されているかの対応関係を保持している。Xen によるゲスト OS のマイグレーションを行うとゲスト OS の持つ MAC アドレスが瞬時に別のポートに接続される形となり、対応表との不整合が起きる。そこで本システムでは、マイグレーションを行う際に、ゲスト OS から一定時間パケットを送出させることで、スイッチングハブに対応表の再学習を促す方法をとっている。

表 1 PrestoIII クラスタの物理構成

CPU	Opteron 242
Memory	2048MB
Network	1000BASE-T
Kernel	2.6.12
MPI	mpich-1.2.7

5. 評　　価

5.1 評 価 項 目

本節では、Xen および VPN の基礎的な評価、および提案システムの有効性の評価を次の項目について行った。

- Xen での MPI の実効性能
- Xen と OpenVPN を組み合わせた場合の MPI の実効性能
- マイグレーションにかかるコスト
- 動的マイグレーションを行う資源監視システムの有効性

5.2 評 価 環 境

評価には本研究室 PrestoIII クラスタを用いた。各ノードの物理構成を表 1 で示す。用いたマシンはデュアル CPU のマシンであるが、今回の評価では、1 ノードあたり MPI 1 プロセスのみ、Xen 環境では 1 ノードに 1 ゲスト OS とした。MPI は mpich-1.2.7[4]、またベンチマークには MPI の基礎的な性能指標として一般的に用いられる Nas Parallel Benchmarks(以下、NPB)[5]、およびプロトコル依存のネットワーク性能計測を行う NetPIPE[6] を用いた。

5.3 Xen および Xen と VPN での MPI 実行性能

Xen を用いることによるオーバーヘッドを計測するため、通常のクラスタ環境でネイティブなカーネル 2.6.12 を用いた性能と、仮想計算機環境で Xen のカーネル 2.6.12 を用いたゲスト OS 上での性能を比較した。

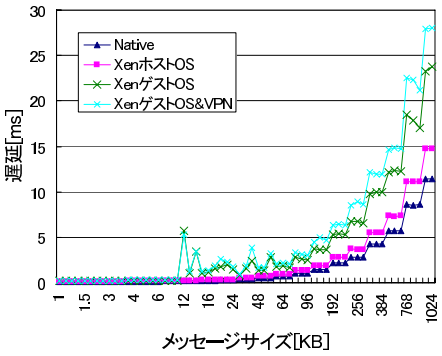

図 3 MPI による ping pong のメッセージサイズと遅延の関係

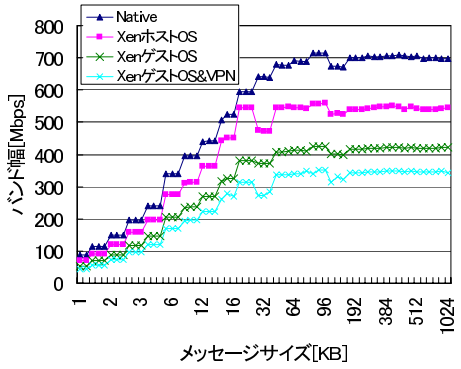

図 4 MPI による ping pong のメッセージサイズとバンド幅の関係

図 3,4 は NetPIPE3.6.2 による MPI のメッセージ Ping Pong において、メッセージサイズと遅延、バンド幅の関係を示したものである。メッセージサイズによらずにネイティブな環境に対してゲスト OS ではほぼ倍の遅延が発生している。さらにこの環境に VPN を使用することで更なる性能低下が見られる。これに対してホスト OS は、ネイティブ環境とゲスト OS の中間の値となっている。ネットワーク性能のオーバーヘッドに関する詳細な議論は次節で述べる。

次にに 4,8,16,32 とノード数を増加させ NPB3.1 の EP,LU,CG を実行した結果が図 5,6,7 の Native,Xen ホスト OS,Xen ゲスト OS の 3 系列である。図 5 に示される EP では各プロセス間の通信はほとんど発生せず、各マシンで計算が行われる。性能は各環境ともにほぼ変わらずローカルでの計算においてはほとんどオーバーヘッドは発生していないと考えられる。図 6,7 に示される、LU,CG では他のプロセスとの通信が行われる。特に CG は頻繁な通信が行われるベンチマークであり、LU では 16 台までスケールしているのに対して、CG、ゲスト OS での実行ではノード数が増えるにしたがって性能が低下し、台数効果が得られていない。この原因に関する議論は同様に次節で行う。

5.4 Xen と VPN を組み合わせによる性能

OpenVPN による仮想ネットワーク上に、Xen のゲスト OS を動作させ、4,8,16,32 とノード数を増加させ NPB3.1 の EP,LU,CG を実行した結果が図 5,6,7 の Xen ゲスト OS&VPN の系列である。Xen のみの時と同様、通信頻度が少ない EP はネイティブに同等の性能となっている。一方で LU では Xen ゲスト OS に比べてスケーラビリティが減少し、16 ノードでから性能低下が始まっている。また CG では Xen ゲスト OS と同様 4 台から、台数を増やすごとに性能は比例して低下し、相対的に Xen ゲスト OS よりも低い値となっている。これらの結果は VPN がネットワークパケットをハンドリングし、カプセル化するという構造

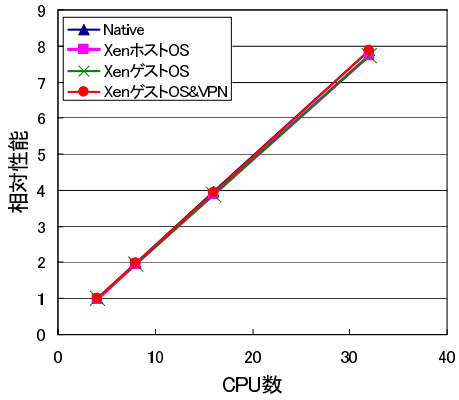

図 5 NPB3.1(EP) におけるスケーラビリティ 4CPU 時を 1 とした相対性能

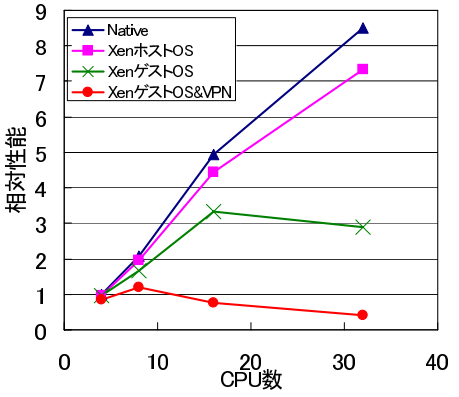

図 6 NPB3.1(LU) におけるスケーラビリティ 4CPU 時を 1 とした相対性能

上、オーバーヘッドが生じるのは確実ではあるが、相対的に性能を低下させるだけではなく、LU においてはスケーラビリティを低下させる結果となっている。

5.5 N-Queen 問題による性能評価

次に N-Queen 問題の計算時間を測定した。使用したプログラム[7] は電気通信大学で作成されたものである。このプログラムは、マスタワーカ型で分割したタスクを動的にワーカに配布する。

結果を図 8 に示す。ワーカを 4,8,16,32 ノードで実行した結果、すべての系列で同様の結果を得た。これから、通信頻度の高くないアプリケーションでは、複数資源での実行によって正しく台数効果を得ることが確認された。

5.6 マイグレーションのコスト

Xen によるマイグレーションが全体の実行時間に与える影響を計測した結果が図 9 である。計測は、NPB2.4(EP,NPROCS=8,CLASS=B) を実行し、実行中に 1,2,4,8 回のマイグレーションを実行した場合の、マイグレーションを行わない場合に対する実行時

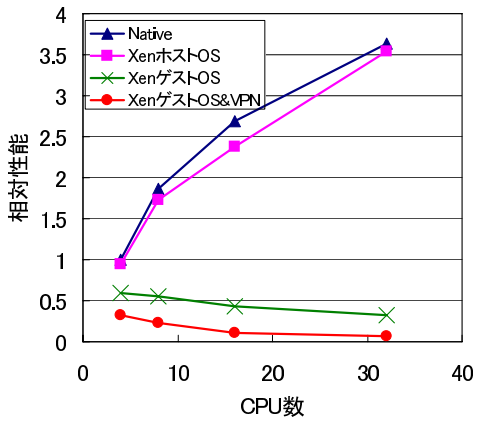

図 7 NPB3.1(CG) におけるスケーラビリティ
4CPU 時を 1 とした相対性能

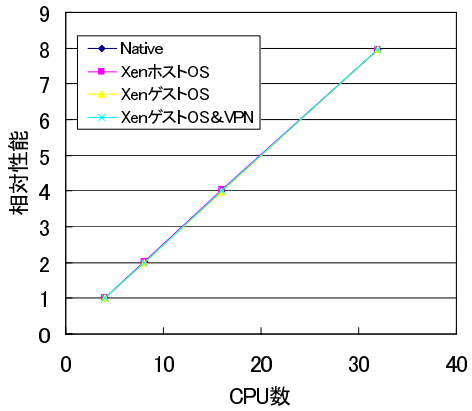

図 8 18-Queen 問題
4CPU 時を 1 とした相対性能

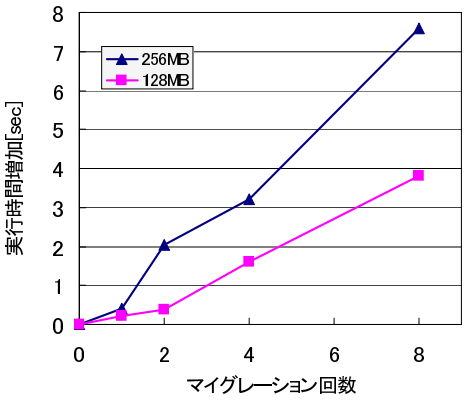

図 9 Xen 上で OpenVPN を使用した場合と Native な Linux の NPB 実行時間の増加

5.7 資源監視システムの有用性

次にプロトタイプ実装をした動的なマイグレーションを行う資源監視システムの評価を行った。ここでは負荷分散を目的としたマイグレーションポリシーを用いた。図 10 のように、実行開始時に Node1 で 1CPU

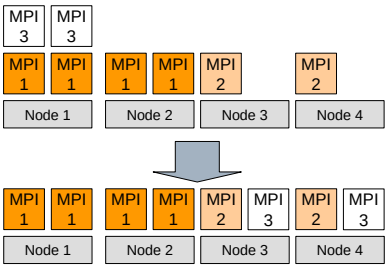

図 10 負荷分散による実行効率の向上

間増加を測定した。この結果から、メモリ量が 256MB の時で 1 回あたりのコストは約 1 秒程度、128MB の時に 1 回あたりのコストは約 0.5 秒程度と考えられ、メモリサイズにオーバーヘッドが比例すると考えられる。

メモリの内容を全て転送する場合、1000BASE-T で約 100MB/sec 程度で転送を行える。しかし、その転送時間以上に図 9 が高速なのは、Xen はゲスト OS のマイグレーション時に、マイグレーション元で稼働中のままメモリ内容をマイグレーション先へ転送し始め、転送後に変更された部分だけ再度転送し、大部分のメモリを送信し終えた後に、マイグレーション元の OS をサスペンドし、残った部分を転送する、という方法をとっている。これによってゲスト OS のダウンタイムを可能な限り短くしマイグレーションに要する時間を短縮してる。

に対して 2 つのゲスト OS が割り当てられている状態になっている。これに対して、本システムは各 Node の稼働中のゲスト OS 数の均一化を行うため、二つのゲスト OS は Node3,Node4 へとマイグレーションされた。

この結果、上で述べたマイグレーションを行う本システム上での実行時間と、ネイティブ環境上で同様のプロセスを実行した場合の実行時間の比較を表 2 で示す。ネイティブな環境と比較しても、Xen 使用によるオーバーヘッド以上に、実行時間の短縮がなされている結果となった。

6. Xen における MPI 実行の評価結果に関する考察

前節での評価により、Xen 上での MPI 実行は通信頻度によりオーバーヘッドが非常に大きくなるという

表 2　ネイティブ環境と提案システムの NPB2.4(LU,CLASS=A) 実行時間 (単位:秒)

	ネイティブ環境	提案システム
MPI-1(NPROCS=4)	137.34	99.89
MPI-2(NPROCS=2)	163.29	188.07
MPI-3(NPROCS=2)	233.68	199.16

結果が得られた。特に CG では台数増加に比例して性能が低下している。ここでは、そのオーバーヘッドの原因についての考察を行う。

6.1　Xen のネットワークに起因するオーバーヘッド

Xen のゲスト OS の外部のネットワークとの通信経路を図 11 で示す。ゲスト OS から計算機外部へパケットが送信される場合、ゲスト OS のフロントエンド・ドライバ、ホスト OS のブリッジ内のバックエンド・ドライバ、ネイティブな Linux ドライバを経て NIC へ到達する。ドライバではメモリコピーが行われるため、通常より 2 箇所多いドライバの存在はオーバーヘッドの原因として考えられる。また Xen 環境上に VPN を用いた場合、図 11 での経路上に TUN/TAP デバイスを経由するため、オーバーヘッドはさらに増大する。

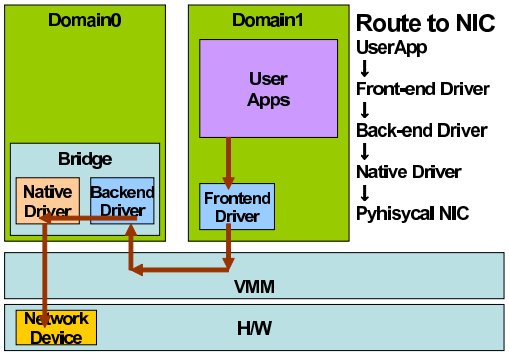

図 11　Xen の内部ネットワーク経路

6.2　ゲスト OS 切り替えと通信タイミングの不整合

単純な遅延のみで比較を行うと Xen 環境はネイティブ環境の 2 倍程度であるが、図 7 の結果はそれ以上のオーバーヘッドが生じたことを示している。上で述べたネットワークの構造的な原因以外に、ゲスト OS の並列仮想化を行うことによって、CPU 時間を複数のゲスト OS でシェアするための OS の切り替えから来るオーバーヘッドが考えられる。これは通信頻度が非常に高いアプリケーションにおいて、OS 切り替えタイミングと通信の時間的粒度が同程度であり、ノード間の通信タイミング、特に同期などに影響を与えていると考え、CG の通信のログ取得を行い、どの部分でのログが有効であるかの試行を行うと同時に、Xen の OS の切り替えタイミングをログとして出力する方法の検討を行っている。

6.3　本環境に適したアプリケーション特性および改善の可能性

評価結果およびこれまでに述べた議論から、Xen の仮想化の基本的なアイデアが MPI のようなネットワーク・インテンシヴなアプリケーションを想定していないと考えられる。しかし、Intel の Virtualization Technology(VT) など、プロセッサ、ハードウェアレベルでの仮想化によって改善の可能性は十分にあり、今後も継続的な調査を続けていく予定である。

またで Xen&VPN 環境での改善の可能性について、OpenVPN をはじめとする TUN/TAP デバイスを用いた VPN では、図 12 のようにネットワークデバイス、ブロックデバイスの二面性をもつ TUN/TAP デバイスのネットワーク側から書き込まれたパケットを一度ブロック側から読み込んで暗号化・カプセル化した上で物理 NIC から送出する。この構造はホスト OS で Back-end ドライバからブリッジを経て Native ドライバへ書き込む Xen の構造と類似している。そこで、VPN プロセスが行う暗号化カプセル化をブリッジが行う構造にすれば、TUN/TAP デバイスのようなドライバが不要になりメモリコピーの回数の減少により Xen&VPN 環境でのオーバーヘッドを削減出来ると予測している。

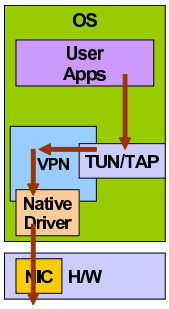

図 12　TUN/TAP を用いた VPN の構造

7.　関 連 研 究

ここでは、本研究に関連すると思われる、グリッド上での MPI 実行、マイグレーションを用いた遊休計算機利用、負荷分散の研究事例を紹介する。

Condor[8] は、遊休状態にある計算資源に対するジョブのスケジューリングを複数行うことで、ハイスループットコンピューティングを実現している。Condor 上で実行されるジョブはチェックポイント/リスタートによる計算プロセスのマイグレーションにより、負荷分散、遊休計算機利用を実現している。しかし、Condor の MPI 実行環境である MPI ユニバースでは上で述べたチェックポイント/リスタートに対応していない。GridMPI は[9] はグリッド環境での実行のための MPI

実装である。サイト内、サイト外間での遅延を考慮に入れた通信の特性に対して最適化を行い、10ms 以下の地理的に比較的近い計算資源上での実行に適している。

Phoenix[10] は東京大学で開発されている、マイグレーションによる動的な資源の追加削除、負荷分散を目的とした並列計算ライブラリである。ユーザー透過な、遅延を考慮した動的な資源の追加と削除が可能となっている。

8. おわりに

本稿では、仮想計算機 Xen、OpenVPN を用いて、複数サイト間に対して低コストなマイグレーションが可能な、グリッドに対応した MPI 実行環境の提案し、テストベッド構築と基礎的な評価を行った。その結果、Xen のゲスト OS 上で N-Queen 問題や NPB の EP などの CPU インテンシヴなアプリケーションではネイティブ環境と同等の性能が得られ本提案の有用性が示せた。一方で CG を実行した結果、台数に比例して性能が低下するという問題が露見した。このオーバーヘッドの原因は、Xen の内部ネットワークの構造的な問題、また Xen の各 OS の切り替えスケジューリングと頻繁な通信タイミングが不整合を生み出している、という 2 点であると考えおり、今後も継続的な調査を行っていく予定である。

また、Xen 環境上で VPN を用いることにより、複数のサイト間で同一のネットワークアドレスを用いることが可能となり、Xen のマイグレーションをグリッド環境のような異なるネットワークアドレスのサイト間で自由に行うことが可能となる。しかし、Xen および VPN を組み合わせた環境では前に述べた Xen のみの環境よりもさらに大きな性能低下が見られた。

最後に今後の課題を挙げる。サイト間をまたがる計算資源にマイグレーションによって MPI プロセスが分散した場合、サイト間の高遅延低バンド幅なリンクによって MPI 実行が著しく性能低下する。これに対処するには MAGPIE[11] のように、通信トポロジーを意識することで、より効果的なマイグレーションが可能となり、サイト間リンクでの通信の頻度を減少させ影響を少なくする事を検討する。

実装面では、構築した環境では NFS によってゲスト OS の root イメージを共有していた。しかしグリッド環境での使用を前提とする場合、Gfarm などのグリッド・ファイルシステムなどのでの共有を行うべきであると考えている。さらにプロトタイプ実装を行った資源監視システムにおいて、マイグレーションポリシーの設定をユーザーが行いやすくするような改善を考えている。

また前節で、今後の課題とした通信頻度の大きなプログラムのオーバーヘッドの原因の調査と、改善方法の模索も継続的に行っていく予定である。

謝辞 本研究の一部は，科学技術振興機構・戦略的創造研究「低消費電力化とモデリング技術によるメガスケールコンピューティング」による。

参 考 文 献

1) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauery, R., Pratt, I. and Warfield, A.: Xen and the Art of Virtualization, *SOSP* (2003).
2) Clark, C., Fraser, K., Hand, S., Hanseny, J.G., July, E., Limpach, C., Pratt, I. and Warfield, A.: Live Migration of Virtual Machines, *NSDI* (2005).
3) : OpenVPN. http://openvpn.net/.
4) : MPICH . A Portable MPI Implementation. http://www-unix.mcs.anl.gov/mpi/mpich/.
5) : Nas Parallel BenchMark. http://www.nas.nasa.gov/Software/NPB/.
6) : NetPIPE. http://www.scl.ameslab.gov/netpipe/.
7) : N-queens Homepage. http://www.yuba.is.uec.ac.jp/kis/nq/index.htm.
8) Litzkow, M., Livny, M. and Mutka, M.: Condor - A Hunter of Idle Workstations, *Proceedings of the 8th International Conference of Distributed Computing Systems* (1988).
9) 石川祐, 松田元彦, 工藤知宏, 手塚宏史, 関口智嗣: GridMPI-通信遅延を考慮した MPI 通信ライブラリの設計, SWoPP 松江 2003 (2003).
10) Taura, K., Endo, T., Kaneda, K. and Yonezawa, A.: Phoenix : a Parallel Programming Model for Accommodating Dynamically Joining/Leaving Resources, *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2003)*, pp. 216–229 (2003).
11) Kielmann, T., Hofman, R. F. H., Bal, H. E., Plaat, A. and Bhoedjang, R. A. F.: MagPIe: MPI's collective communication operations for clustered wide area systems, *ACM SIGPLAN Notices*, Vol. 34, No. 8, pp. 131–140 (1999).
12) 建部修見, 森田洋平, 松岡聡, 関口智嗣, 曽田哲之: ペタバイトスケールデータインテンシブコンピューティングのための Grid Datafarm アーキテクチャ, 情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム, Vol. 43, No. SIG 6 (HPS 5), pp. 184–195 (2002).