社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

仮想計算機を用いて負荷分散を行うMPI実行環境

立薗　真樹[†]　　中田　秀基[††,†]　松岡　　聡[†,†††]

† 東京工業大学, 東京都
†† 産業技術総合研究所, 東京都
††† 国立情報学研究所, 東京都
E-mail: †tatezono@matsulab.is.titech.ac.jp

あらまし　長時間に及ぶ実行や遊休計算機上での実行などにおいては、各計算機間の負荷分散が重要である。我々は仮想計算機を用いた負荷分散可能なMPI実行環境を提案する。これはMPI実行環境を仮想計算機ごとマイグレーションさせることにより、MPIプロセスに透過的なマイグレーションを実現する。仮想計算機としてXen [8] を用い、同計算機が持つマイグレーション機能を用いたプロトタイプを実装した。さらに、VPNを用いたネットワークの仮想化によって、異なるサブネットへのマイグレーションを実現した。同プロトタイプを評価した結果、MPI実行時マイグレーションが可能かつ、仮想化によるオーバーヘッドは10%から、通信の頻繁なものでは50%超えることが分かった。 また、同プロトタイプに負荷分散アルゴリズムを実装し、長時間実行されるジョブにおいて実行効率の向上を確認し、本提案の有効性を確認した。

キーワード　MPI, マイグレーション, 仮想計算機, 負荷分散

MPI Environment with Load Balancing using Virtual Machine

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

† Tokyo Institute of Technolory, Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552 JAPAN
†† National Institute of Advanced Industrial Science and Technology, Soto-kanda 1-18-13, Chiyoda-ku, Tokyo, 101-0021, JAPAN
††† National Institute of Informatics, Hitotsubashi 2-1-1, Chiyoda-ku, Tokyo, 101-8430, JAPAN
E-mail: †tatezono@matsulab.is.titech.ac.jp

Abstract　Load balancing is one of the key features for efficient execution of long-running jobs, idle-cycle harvesting, etc. We propose a technique to achieve a load-balanced MPI execution environment using a virtual machine monitor. 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. Our initial implementation of the technique uses a virtual machine monitor called Xen, which has an integrated VM migration ability, and VPN for migration among different subnets. We experimentally show that the prototype successfully achieves runtime migration of MPI processes, and that the overhead due to virtualization ranges from 10% for computation-intensive applications to 50% for network-intensive ones. We also implemented a simple load-balancing algorithm on top of the prototype. Experiments with it suggest that the runtime migration is effective for efficient execution of long-running jobs even with the virtualization overhead.

Key words　MPI, Migration, Load-balancing,VirtualMachine

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いる。MPIで用いられるプログラムは、実行に数時間から長いものでは、数日〜数ヶ月に及ぶものまで存在する。MPIの特徴としては、他のプロセスとの通信を行うため、並列化されたうち一つのプロセスでも失われれば計算全体に影響を及ぼす。この性質から、一度実行された計算は通常、終了するまでその計算機上で実行しなければならない。しかし、それではメンテナンス等も行えず、また一つ

の計算ノードで複数ユーザーの計算プロセスが実行された場合には、競合状態になったプロセスの影響で、全体の計算進捗に影響を及ぼすこともある。これらに対処するためには、計算実行中でも自由に MPI プロセスの他計算ノードへのマイグレーションを行うことが必要であり、これによって負荷の低いノードへ負荷の高いプロセスを持って行き、負荷分散が可能となる。

一方で、マイグレーション先の候補は通常、高バンド幅、低遅延なネットワークで接続された同一サイト内の計算ノードが適当である。しかし、同一サイト内に負荷の低いノードが存在しない場合、他のサイトへのマイグレーションを行うことが出来れば、高負荷なプロセスを同一ノード上で実行するという競合状態に陥らずに済む。

このような自由なマイグレーションを実現できれば同時に、近年広く普及しつつある遊休計算機利用においても MPI を実行することが容易になる。これは、夜間や休日など計算資源の所有者が利用しない時間に、遊休状態にある計算資源を複数台集め、ハイスループットコンピューティングを実現するものである。そのような環境で MPI を実行できれば、高価な計算機を導入せずとも、高い計算能力をもった MPI 実行環境が得られることになる。しかし、遊休計算機での実行は、本来の所有者の利用を優先するという原則があり、所有者の使用開始時には計算プロセスを中断もしくは中止しなくてはならない。

上で述べた負荷分散や遊休計算機利用を達成するために用いるマイグレーションは、従来は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、マイグレーション先でネットワークソケットや PID が変更されたり、プロセス以外にもディスクリプタなどの保存が必要、また中間ファイルの生成時などに問題が生じるなど、実装上の障害が多々ある。

本研究で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 計算を行うゲスト OS ごとマイグレートする方法を提案する。Xen による MPI 実行のオーバーヘッドを測定、MPI アプリケーションの性質により同等から半分程度の性能を示し、これを用いてユーザーが意識することなくマイグレーションの実行が可能とした。また計算ノードの負荷情報を収集し、自動的にマイグレーションを行う、動的負荷分散システムを提案、そのプロトタイプを実装し、マイグレーションによる負荷分散の効果を確認した。さらに広域での MPI 実行に向けて、OpenVPN によるネットワークの仮想化を行い、異なるサブネットにマイグレーションを実行できる環境を構築した。そのオーバーヘッドを測定した結果、頻繁なアプリケーションではオーバーヘッドが非常に高く、特に一部はネイティブな環境に比べて 5%程度の性能であった。

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 Minitor 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、OS カーネルのマシン依存のコードを書き換え VMM 上で動作するようにすることによって実現されている。現在 Liux2.4 系、2.6 系、NetBSD が動作可能である。

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が可能である [5]。このマイグレーション機能は、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 (Ex. $\$xm\ \ migrate\ \ guest1\ \ dist\_host$) 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。この時、求められる条件は、マイグレーション先の同じディレクトリにゲスト OS の root イメージが存在することである。同一のサブネット内であればマイグレーション後も MAC アドレスは変わらないので、ネットワークのコネクションは問題なく動作する。

3. 設　　計

3.1 広域での実行にも対応した MPI 実行環境への要件

広域に分散する計算資源にもマイグレーションが可能で、かつ効率良く MPI を実行できる環境を構築するためには、次のような要件を満たす必要がある。

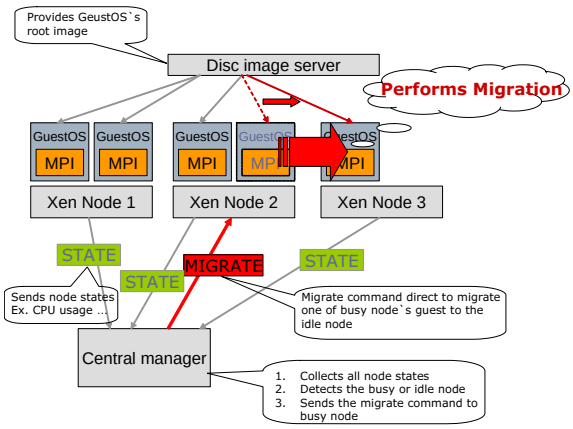

図 1　仮想計算機を利用したマイグレーション可能な MPI 実行環境

- 計算資源の監視　遊休計算機利用、負荷分散という観点から、計算資源を監視し適切にマイグレーション実行の決定、マイグレーション先のマッチングを行わなければならない。
- MPI 実行中でも必要に応じてマイグレーションが可能
計算中のノード上の MPI ジョブでも、必要に応じてマイグレーションを行うことによって、動的な負荷分散、所有者のタスクを優先した遊休計算機利用等が可能となる。この時、ユーザーが意識することなく透過的にマイグレーションが行われる事が理想である。
- 広域に分散する計算資源への対応　マイグレーションに

よってサイト間をまたぐような環境で MPI を実行しなくてはならなくなった時に、MPI 実行を継続して行えるようにネットワークの違いを吸収し、また高遅延なサイト間リンク上で頻繁な通信を行わないような実行が必要になる。

3.2 設計の方針

我々が提案する MPI 実行環境では、すべての MPI プログラムは Xen 上で動作するゲスト OS の上で実行される。また計算資源は、別に用意された集中管理ノードによって、CPU 使用率、メモリの使用量、ゲスト OS の数等の状態が監視されている。

マイグレーションの実行を決定するのも集中管理ノードである。集中管理ノードがマイグレーションの実行およびマイグレーション先を決定すると、マイグレーション元のノードにマイグレーション実行命令が送信され、そのノードで Xen のコマンドによりマイグレーションが実行される。マイグレーションのポリシーは、ノードの CPU 数よりも多くのゲスト OS が CPU を使用しているような場合に、CPU を使用しているゲスト OS が CPU 数よりも少ないノードへのマイグレーションを行う、等である。

また原則的に 1 ゲスト OS あたり MPI 1 プロセスを実行し、ゲスト OS 単位でのマイグレーションの判断を行うことで、プロセスマイグレーションと同等の粒度で、マイグレーションによるメリットが得る事が出来る。

一方、マイグレーション先の選択時に、必ずしも同一サイト内に最適なホストが見つかるとは限らない。そのような場合に、広域ネットワーク越しに接続された他サイトの計算資源もマイグレーション候補とすることが出来れば、より複雑なマイグレーションのシナリオに対応できる。しかし、Xen のゲスト OS は、ホストのネットワークにブリッジ接続参加するので、別ネットワークの Xen 上にマイグレーションした際には、ゲスト上で動く MPI プロセスの持っている IP 情報とゲスト OS の IP が異なり、他のプロセスへ通信が出来なくなる。そこで、ゲスト OS 上に VPN によって仮想的なネットワークを作ることによって、マイグレーションによるゲスト OS の IP に影響を受けないネットワーク空間を構築する。MPI はそのネットワークの IP を使用し実行する。これにより、実行中の MPI であっても他のサイトへのマイグレーションが可能になる。

4. 実　　装

設計の方針に従い、マイグレーションによって負荷分散を行うプロトタイプシステムの実装を行った。

4.1 集中管理ノードによるマイグレーション実行の決定

集中管理ノードが監視する計算ノードの状態は、各ゲスト OS の CPU の使用率と、ノード上の空きメモリ量とした。本研究の対象アプリケーションは MPI であるので、ゲスト OS 間で競合するのは CPU の使用であると考えられる。また空きメモリ量は、新規にゲスト OS を立ち上げる、もしくはマイグレーションによって持ってくる場合、そのゲスト OS が使用する、もしくはしているのと同等の空きメモリがホストになくてはならない。これらの情報は、Xen のゲスト OS 情報を表示するコマンドから収集し、ソケット経由で集中管理ノードに集める。ノードでは次節で述べるマイグレーションポリシーと収集したノード情報に基づきマイグレーションを行うか判断する。マイグレーションの命令は各計算ノードで動く命令受信デーモンが受信し、Xen のマイグレーションコマンドを呼び出す。

4.2 マイグレーションの実行と行き先の決定

マイグレーションの実行と行き先の決定のポリシーは様々なものが考えられるが、今回は以下のようなポリシーで行う。

a） 行き先候補が同一サイト内

行き先候補が同一サイト内の場合、図 2 のように、極力各ノードの動作中のゲスト OS の数が均等になるように行う。特に、図 2 の上は、デュアルプロセッサにもかかわらず、各ノードに 1 ゲストとなるように更にマイグレーションを行う。これは、同じ 2 プロセスも、デュアルプロセッサのマシン 1 台よりも 2 台用いた方が高速に実行できるためである。

b） 行き先候補が他サイト

図 3 のように、同一サイト内に CPU の数より少ないゲスト OS が動作しているようなノードがない場合、他サイトのノードをマイグレーション候補とする。この時はやはり行き先のサイト内で、上の同一サイト内のポリシーを適用する。

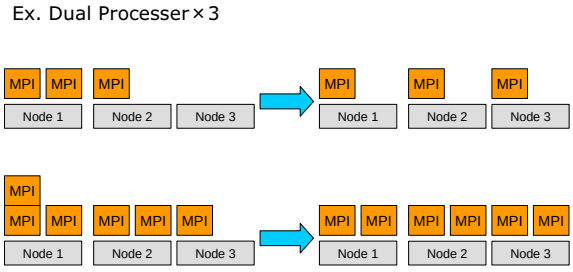

図 2　同一サイト内へのマイグレーションの例

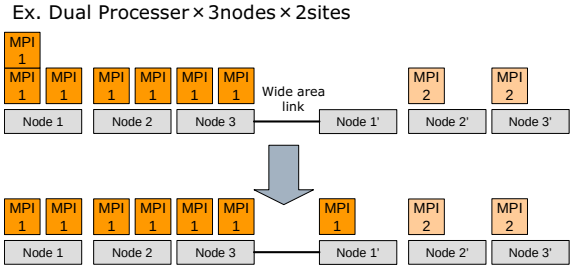

図 3　他サイトへのマイグレーションの例

4.3 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、マイグレーション元と先で同じディレクトリ階層に同じ名前のゲスト OS の root イメージが必要となる。整合性のとれたディスクアクセスを行うために、本研究では、Root 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、同様にマウントするようにした。そのため本システムでは、ネットワークのボトルネックを考え、I/O インテンシヴでない MPI アプリケーションを対象とする。

表 1　評 価 環 境

CPU	AMD Opteron 242
Memory	2048MB
Network	1000BASE-T
OS	XenoLinux 2.6.11-xen0(Xen) or Linux2.4.30(Native)
MPI	MPICH-1.2.6

4.4　スイッチングハブへの対応

各ノードをスイッチングハブで接続している環境でゲスト OS のマイグレーションを行った場合、ゲスト OS の NIC ごと他の Xen ホストに移動する形となる。新しいホストはマイグレーション元のホストとは別のスイッチのポートに接続されているため、そこにブリッジ接続するゲスト OS のポートも変わる事になる。マイグレーション後にゲスト OS が $MPI\_RECEIVE$ 関数などで待機状態になっていた場合、即座にルーティングテーブルの更新が行われない。そこで、マイグレーション直前に、対象となるゲスト OS 内から他のノードへ ping コマンドなどを実行することによ、マイグレーション後も一定時間パケットの送信を行い、直ちにスイッチングハブ内のポート情報を更新し、瞬時のネットワーク通信の回復を行う。

4.5　OpenVPN による広域へのマイグレーション

OpenVPN [3] は、Tun デバイスを用いた SSL 対応の VPN である。これを用いることで、実際のネットワーク空間とは異なったサブネットのネットワーク空間提供され、複数の実サブネットに跨った仮想的なサブネットのネットワークの構成が可能となる。またネットワークデバイスとしてのインターフェイスを提供するので、ユーザーアプリケーション側からは VPN 利用に関して一切意識する必要が無い。

本研究では、図 4 のように OpenVPN を Xen 上で用いることによって、実行中の MPI の他サイトへのマイグレーションを可能とした。

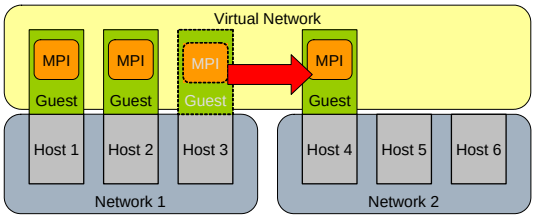

図 4　VPN を利用した他のネットワークへのマイグレーション

5.　評　　価

5.1　評 価 環 境

評価には松岡研 PrestoIII クラスタを用いた。各ノードの主な構成を表 1 で示す。また、ベンチマークには Nas Parallel Benchmark2.4(以下、NPB) [2] を用いた。

5.2　Xen のオーバーヘッドの評価

Xen を用いることによるオーバーヘッドを計測するため、PrestoIII クラスタの通常環境であるネイティブな Linux-2.4.30 カーネル時と、Xen 起動時、カーネル linux-2.6.11-xen0 を用いたゲスト OS 上で、16 プロセス、クラス B の各ベンチマークを使用してその実行時間を比較した結果が図 6 である。通信の頻度が少ない LU,EP はネイティブな Linux に近い値が出ているのに対して、CG,MG はネイティブに対して半分以下のパフォーマンスとなっている。これは Xen のゲスト OS のデバイスはが Domain0 と呼ばれる最初に起動するホスト OS のデバイスドライバを利用する事から生まれるオーバーヘッドと考えられる。実際に図 5 に示すように Domain0 とネイティブな Linux との比較では大きな差は見られず、最大でも 5%程度のオーバーヘッドである。

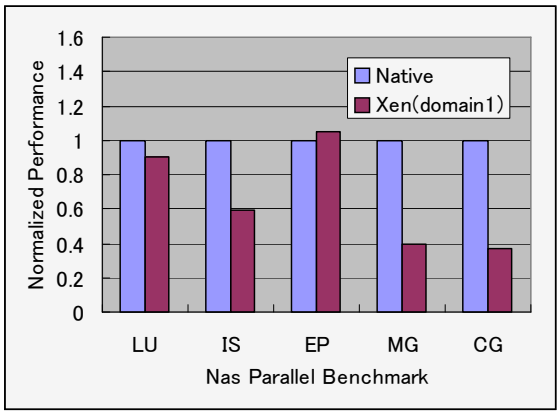

図 5　Xen(domain1) とネイティブな Linux の NPB 実行時間

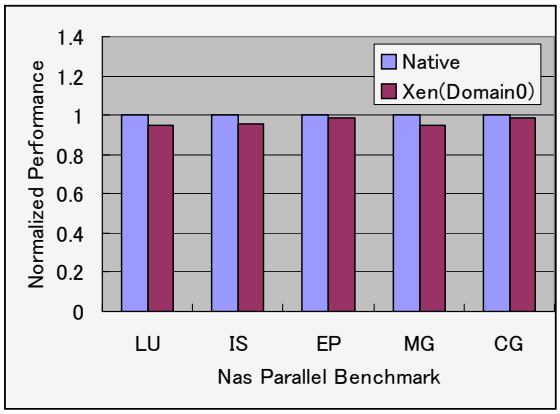

図 6　Xen(domain0) とネイティブな Linux の NPB 実行時間

5.3　マイグレーションのコストの評価

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 7 である。計測は、NPB の EP のクラス B 問題を 8 プロセスで解き、途中に 1,2,4,8 回のマイグレーションを実行した場合の実行時間を測定した。その結果から、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

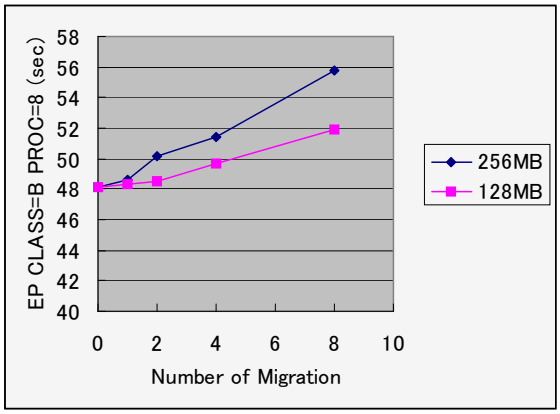

図 7　Xen 上で OpenVPN を使用した場合と Native な Linux の NPB 実行時間

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。この方法によってゲスト 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。

実行時間が長い MPI アプリケーションであれば、この程度のマイグレーションのコストは、無視できる量だと思われる。

5.4　負荷分散システムのプロトタイプの評価

プロトタイプが正しくマイグレーションを行うかを評価するため、図 8 の上の図で示すように、始め Node1 に使用する 8 つの全てのノードを集めて実行を開始後、Node4,Node5 以外の 6 ノードが遊休状態にあると判断され、Node1 のゲスト OS に対してマイグレーションが実行され、図 8 の下の図のようにゲスト OS が再配置された。これによって、ユーザーが使用するノードを意識せずに単一のノードに細分化した MPI が動くゲスト OS を用意することで、システムが適切にゲスト OS の再配置を行うといった実行方法が可能となる。

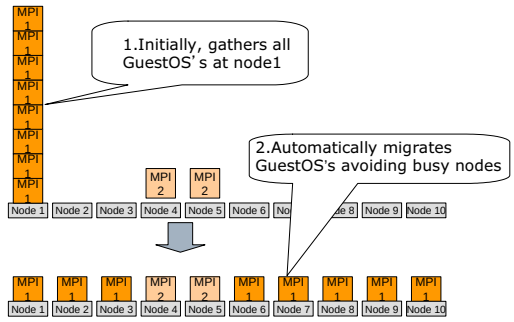

図 8　動的負荷分散の例

また負荷分散による実行効率の向上の評価を行った。図 9 のように、実行開始時に Node1 で 1CPU 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、プロトタイプは各 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な Linux 上で、そのまま実行した場

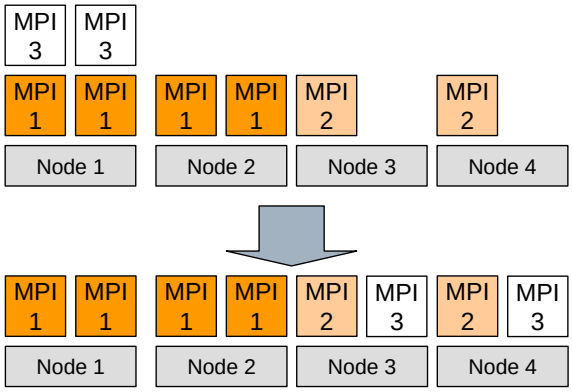

図 9　動的負荷分散による実行効率の向上

表 2　ネイティブな Linux 環境とマイグレーションを行う提案システムでの実行時間の比較　単位:秒

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

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。

5.5　VPN のオーバーヘッドの評価

Xen 上のゲスト OS に OpenVPN による仮想ネットワークを構築し、その上で MPI を実行し、Xen と OpenVPN の使用による、広域へのマイグレーションを行うための呼び評価を行った結果が図 10 である。最も良いもので EP の Native 対

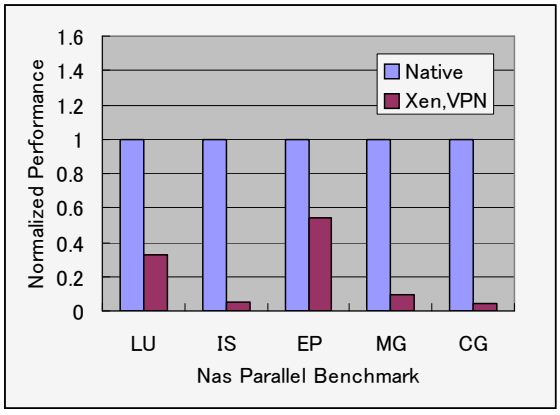

図 10　Xen 上で OpenVPN を使用した場合と Native な Linux の NPB 実行時間の比較

する 50%程度のパフォーマンス、さらに CG ではネイティブな Linux の 5%のパフォーマンスしか出ていない。この結果から、現状の VPN を用いた環境では、通信を頻繁に行うような MPI アプリケーションの実行は現実的ではないと考えられる。しかし、EP のような通信を頻繁に行わないものの場合、マイグレーションを行うことよって遊休計算機利用での実行による高い台数効果によって単一の小規模クラスタでの実行するより良い結果を出すことも可能と思われる。

6. 関連研究

ここでは、本研究に関連すると思われる、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。

ウィスコンシン大学で開発されたジョブスケジューリングシステムである Condor [7] 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。各計算機上では実行中のプロセスに対して、チェックポイントを取り、所有者の復帰時、もしくは障害発生時には他の計算機上で保存されたチェックポイントからのリスタートを行うことによって計算を継続することが出来る。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おらず、他のプロセスと通信する MPI は、1 プロセスのみのやり直し等が出来ず、一度開始された計算は終了するまでその計算機上で実行するしかない。

プロセスマイグレーションを利用した負荷分散の研究例に OpenMosix [10] がある。OpenMosix は、カーネルレベルで動作し、登録された計算ノードの負荷状況を管理し、負荷平準化アルゴリズムを用いて高負荷なプロセスを複数持つノードから、負荷の低いノードへプロセスのマイグレーションを行う。これによって最終的には各計算ノードの負荷が均等に近づけることが出来る。しかし、OpenMosix はスレッドへの対応がなされておらず、また広域にまたがる計算資源を利用することが出来ない。また、マイグレーションはプロセスのみで行われるため、中間フィルの生成も不可能である。

Phoenix [9] は動的な資源の追加削除、負荷分散を目的としたマイグレーションを可能とした並列計算ライブラリである。動的な資源の追加と削除への対応はユーザーが意識することなく行われる。ノード追加時の接続先および削除時の子ノードの接続先は主に遅延をもとに決定される。また問題なく接続されているノードもより良い親の条件が見つかれば接続先を変える。動的に親ノードを変更することによってツリーの末端まで遅延の少ないツリー構造が出来る。また遅延の高いリンクは親ノードの変更によって避けられる。したがって非常に遅延の大きいリンクでなおかつ必ず通る必要のあるリンクは、動的な再構成によって 1 回の通過で済むような構造に収束することが期待できる。

7. おわりに

本稿では、仮想計算機 Xen のゲスト OS マイグレーション、VPN を用いて、複数サイト間に対してマイグレーションを行い、負荷分散や遊休計算機上での実行が可能な MPI 環境の提案をした。そしてそのプロトタイプとして、負荷分散を行うシステムの実装を行い、実行中の MPI のマイグレーションが行われ負荷分散としての効果を得ていることを確認した。さらにその間教条に OpenVPN を用い仮想的なネットワークを構築し、MPI 実行のオーバーヘッドを計測したところ、NPB の CG でネイティブな Linux に対して 5%程度の性能しか出ず、通信の頻繁アプリケーションでの OpenVPN の使用は難しいと予測される。

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 [6] で行われているような、通信トポロジーを意識したマイグレーションを行うことで、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が考えられる。

また現在は、Xen のゲスト OS 上の仮想 NIC はホストの物理 NIC を介して、ホストの実ネットワークにブリッジ参加し、VPN はその上に動いている。従ってオーバーヘッドが非常に大きくなっている。そこで我々は、ゲストのネットワークをホストの NIC を直接使用した VPN のような形で実装できれば、他サイト間のマイグレーションでも IP の変更がなされず、またゲストの仮想 NIC の分オーバーヘッドを減少させることが出来ると考えている。

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．

文　献

[1] *MPICH . A Portable MPI Implementation http://www-unix.mcs.anl.gov/mpi/mpich/.*
[2] Nas Parallel BenchMark. http://www.nas.nasa.gov/Software/NPB/.
[3] *OpenVPN. http://openvpn.net/.*
[4] The Xen virtual machine monitor. http://www.cl.cam.ac.uk/Research/SRG/netos/xen/.
[5] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield, editors. *Live Migration of Virtual Machines.* NSDI 2005, 2005.
[6] Thilo Kielmann, Hofman Rutger F, H, Henri E. Bal, Aske Plaat, and Raoul A.F. Bhoedjang. Magpie:mpi's collective communication operations for clusterd wide area systems. In *roc. Seventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'99)*, pp. 131–140, May 1999.
[7] Miron Livny Michael Litzkow and Matt Mutka. Condor - a hunter of idle workstations,. *Proceedings of the 8th International Conference of Distributed Computing Systems*, pp. p104–111, 1988.
[8] Ian Pratt, Keir Fraser, Steve Hand, and Christian Limpac, editors. *Xen and the Art of Virtualization.* Ottawa Linux Symposium, 2004.
[9] Kenjiro Taura, Toshio Endo, Kenji Kaneda, , and Akinori Yonezawa. Phoenix : a parallel programming model for accommodating dynamically joining/leaving resources. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.*
[10] Esposito Mastroserio Taurino Tortone. Openmosix approach to build scalable hpc farms with an easy management infrastructure.