成蹊大学理工学研究報告
J. Fac. Sci. Tech., Seikei Univ.
Vol.47 No.2 (2010) pp.53-58
（特別研究費に係る論文）

電力・性能の動的な最適化を行う PC クラスタの構築

長名　保範[*1]，栗林　伸一[*2]

Construction of a PC cluster with dynamic power/performance optimization feature

Yasunori OSANA[*1], Shin-ichi KURIBAYASHI[*2]

ABSTRACT： Integrated power and performance optimization of network and computers is an important issue in this cloud computing age. This paper proposes a dynamic relocation method of virtual computing nodes to optimize and reduce power consumption of datacenters. The experiment system consists a network attached storage and 2 host PCs with a quad-core CPU, to run 16 virtual computing nodes. From the early evaluation of the method, the method is expected to enable dynamic optimization of power and performance.

Keywords：PC cluster, virtualization, green computing

（Received September 22, 2010）

１．はじめに

情報通信技術が社会の隅々まで浸透しつつある今日，ネットワーク機器とコンピュータの両方を含めた，情報処理・情報通信技術総体での消費電力軽減が大きな技術的課題となっている。本稿では特に，クラウドコンピューティング技術の普及に伴って急速に増加しつつあるデータセンタの消費電力最適化を視野に入れ，仮想化技術を用いて稼働中のタスクを移動することで動的に消費電力や性能の最適化を図る手法の検討と，その予備評価の結果を示す。

本手法は，マルチコアプロセッサを搭載した物理計算ノード上に複数の仮想計算ノードを配置し，複数の物理計算ノード間で仮想計算ノードを移動することにより，物理計算ノード間の負荷分散を行って性能や消費電力を最適化したり，仮想計算ノードを稼働状態においたまま不要な物理計算ノードを停止することでさらに消費電力を抑制するものである。

２．実験システムの構成

図 1 に，実験システムの構成を示す。物理計算ノードは 2 台で，同じネットワークに接続されたストレージを NFS 経由で mount しており，仮想計算ノードのディスクイメージを共有している。

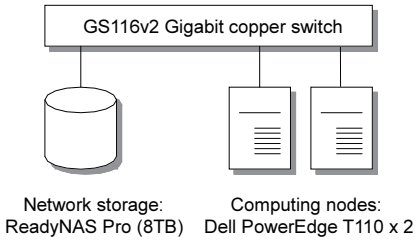

図１　実験システムの構成概要

以下で，構成の詳細を述べる。

2．1　計算ノード

実際の PC である物理計算ノードの上に，仮想化ソフトウェアである VirtualBox OSE [1)] を用いて仮想計算ノードを構成する。

VirtualBox OSE はオープンソースの仮想化ソフトウェアであり，ゲスト OS の userland のコードを Intel アーキテクチャにおける ring 3 でそのまま実行し，通常 ring 0 で実行される kernel space のコードは ring 1 で実行する。必要に応じて逆アセンブルによるエミュレーションが行われる場合もあるが，ring 1 で kernel のコードを可能な限り実行することで，ほぼネイティブにコードを実行することができ，高い実行速度を実現している。

[*1]：情報科学科助教（osana@st.seikei.ac.jp）
[*2]：情報科学科教授

2. 1. 1　物理計算ノード

2 台の物理計算ノードは同じ構成で，以下のようになっている。

- Model: Dell PowerEdge T110
- CPU: Intel Xeon X3440 2.53GHz (4 cores x 2 threads)
- RAM: DDR3 1333MHz ECC, 2GBx4
- OS: FreeBSD 8.1-RELEASE (amd64)

OS はローカルのハードディスクから起動する。これらの物理計算ノードは IPMI [2](Intelligent Platform Management Interface) に対応しており，ネットワーク経由で電源投入などの制御を行うことができる。

VirtualBox OSE を用いて多数の仮想計算ノードを動作させるため，カーネル変数 `kern.ipc.semmni` および `kern.ipc.semmns` を規定値より増加させ，それぞれ 30 と 300 に設定してある。

2. 1. 2　仮想計算ノード

仮想計算ノードは 16 台存在し，すべて以下のような構成になっている。

- Virtualization software: VirtualBox OSE 3.2.8
- CPU: 1 core
- RAM: 512MB
- OS: FreeBSD 8.1-RELEASE (amd64)
- Network: Bridged

ディスクイメージはネットワークストレージ上に置かれており，これを物理計算ノードが NFS 経由で mount することで動作する。初期状態では各物理計算ノードにつき 8 台の仮想計算ノードが起動し，VirtualBox の teleport 機能を用いて動作状態のまま他方の物理計算ノードに移動することができる。

VirtualBox は一般的に GUI から利用されるが，`VBoxManage` などのコマンドを用いてコマンドラインから仮想マシンの操作を行うインタフェイスも用意されており，本稿で扱う実験システムでは 1 台目の仮想計算ノードのセットアップを除き，GUI を使わずに VirtualBox を運用している。

2. 2　ソフトウェア

2. 2. 1　リソース管理ツール

16 台の仮想計算ノードには OpenPBS/Torque [3] を用いて計算ジョブを投入する。OpenPBS/Torque は，PC クラスタ上の計算ジョブを管理するためのバッチ処理システムであり，ジョブのスケジューリングや実行を行う。ユーザはジョブを投入する `qsub`，ジョブを削除する `qdel`，ジョブの状態を参照する `qstat` などのコマンドを使ってジョブを管理することができる。

本稿で取り扱う実験システムでは，仮想ノード 1〜16 のうち，仮想ノード 1 が管理ノード兼用になっており，ここで `qsub` コマンドにより投入されたジョブは，仮想ノード 1ー16 のいずれかで実行されるようになっている。

2. 2. 2　クラスタ管理ツール

ここまでで述べたシステムを，ipmitool や SSH を用いて簡単に管理することのできる管理ツールを開発中である。現在のバージョンでは，

- 物理計算ノードの起動・停止・状態確認
- 仮想計算ノードの起動・停止・状態確認
- 物理計算ノード間での仮想計算ノードの移動

を行うことができる。

３．性能評価

3. 1　計算処理性能

同時に実行するプロセス数や計算ノードの仮想化による性能への影響や，電力効率を測定するために以下のふたつのベンチマークを行った。

- MEncoder [4] による動画エンコード
  - 再生時間: 15 分
  - 解像度: 720x480, 24fps
  - 映像: H264 900kbps
  - 音声: AAC stereo 192kbps
- ClustalW [5] による DNA 配列のマルチプルアライメント
  - データセット: BioBench [6] から取得
  - 430 配列
  - 平均配列長 350bp

以下にこれらの結果を示す。なお，使用した物理計算ノードの CPU が 4 コア×2 スレッド同時実行であるため，仮想計算ノードやネイティブに実行するプロセスの数は最大で 8 とし，以下に示すすべての測定結果は 10 回の測定の平均値である。

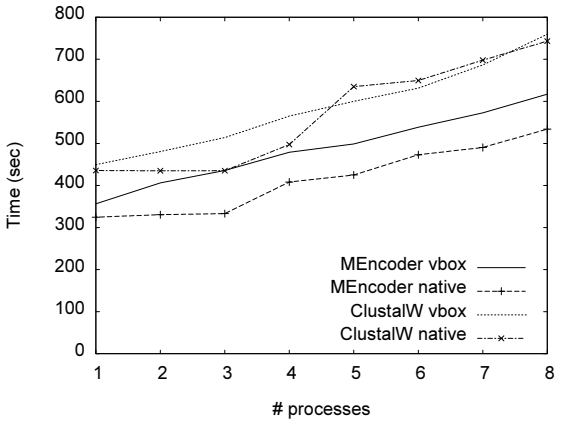

図２　同時実行プロセス数と実行所要時間

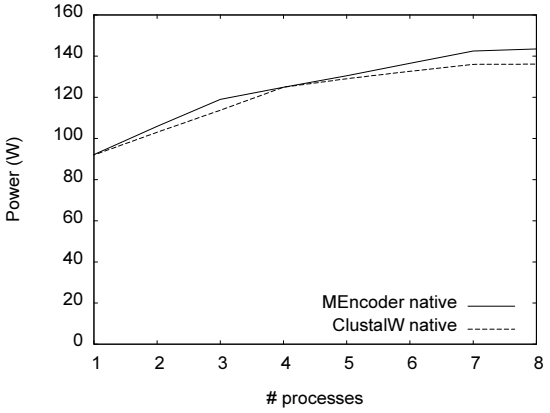

図４　同時実行プロセス数と消費電力

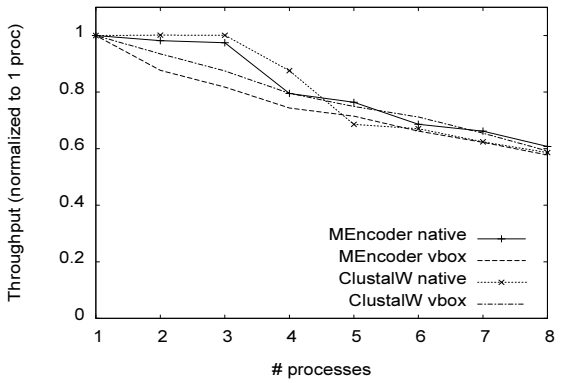

図３　同時実行プロセス数とスループット

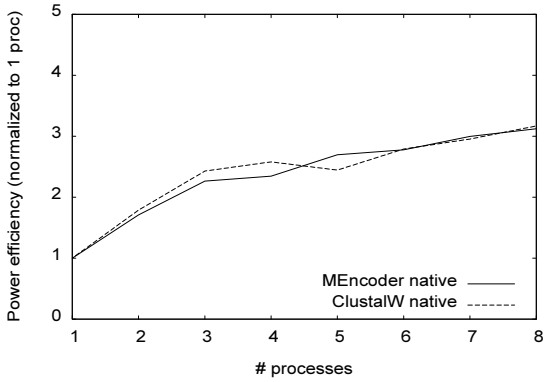

図５　同時実行プロセス数と電力効率

3. 1. 1　実行速度

上記のベンチマークを実行した際の実行時間を図 2 に，1 プロセスの場合の実行速度を 1 とした場合の，複数プロセス実行時の相対実行速度を図 3 に，それぞれ示す。

この結果から，物理計算ノード上でネイティブに実行した場合には，3 プロセスまではほとんど速度の低下が起きず，4 プロセスからスループットが低下し始めるのに対して，仮想計算ノード上での処理はほぼ直線的にスループットが低下することがわかる。ネイティブにコードを実行する場合には，3 プロセス同時の実行までは 4 つのプロセッサコアのうち 1 つを常に OS の入出力に使用することができるのに対して，仮想計算ノードでは常にひとつのプロセッサコアで入出力と計算のすべてを行わなければならないためだと考えられる。

また，ClustalW は比較的ディスクの入出力などの I/O 処理が多いため，仮想計算ノードを用いた場合のオーバーヘッドが大きいこともわかる。

3. 1. 2　電力対性能比

物理計算ノード上で複数の MEncoder および ClustalW を実行した場合の消費電力を図 4 に，実行速度の合計で正規化した電力効率を図 5 に示す。消費電力はすべての CPU コアが稼働する 4 プロセスに達すると増加が若干緩やかになる様子がわかる。また，消費電力の増加よりもプロセス数の増加に伴う処理能力の合計の増加の伸びのほうが速いため，なるべく多くのプロセスを同時実行したほうが電力効率が改善することがわかる。

3. 2　通信性能

MPI などを用いて並列化されたプログラムの実行速度には計算ノード間の通信速度が大きく影響するため，通信速度のベンチマークも実施した。

`dd` および `nc` コマンドを用いて，`/dev/zero` から読み出した 1 ブロック 1024 バイトのデータを転送した際の実効スループットを図 6 に示す。測定は (1) 同一の物理計算ノード上の異なる仮想計算ノード間，(2) 異なる物理計算ノード上の仮想計算ノード間，(3) 物理計算ノード

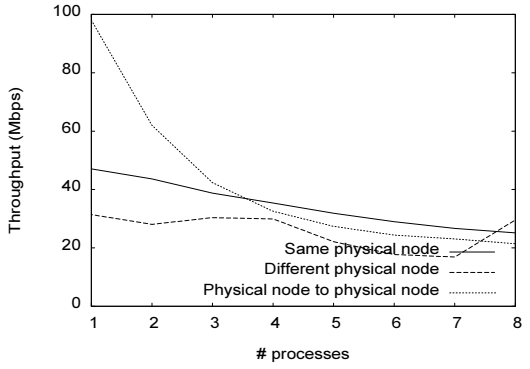

図５　通信スループット

間，の 3 通りで行った。なお，(1) および (2) においては，各仮想計算ノードで実行される送信および受信のプロセスはそれぞれ，最大で 1 つであり，いずれの場合にも最大 8 リンクのデータ転送を行って，1 リンクあたりのスループットを測定した。

この結果から，同一の物理計算ノード上の仮想計算ノード間の通信は，異なる物理計算ノード上のそれよりも 10～30% ほど高速であることと，仮想計算ノード上で 4 リンクを越えるデータ転送を同時に行う場合には物理計算ノード上でネイティブにデータ転送を行った場合と比べて遜色ない転送速度が得られることがわかる。8 リンクの場合には異なる物理計算ノード上のスループットが向上しているが，これはすべての仮想計算ノードの CPU 使用率が 100% に近づき，ホスト OS のスケジューラが各仮想計算ノードを一定の物理 CPU コアに割り当て続けるため，実行効率が向上するためではないかと考えられる。

本実験システムでは動的に仮想計算ノードを物理計算ノード間で移動できるため，並列プログラムの動作環境の動的な最適化を行うことも可能である。

3．3　仮想計算ノードの移動

物理計算ノード間で仮想計算ノードを移動するのに所要する時間は平均して 6.60 秒であった。物理計算ノード間でディスクイメージは共有されており，移動にあたっては CPU コンテキストとメモリの内容を転送すればよいため，1000Mbps の接続で 512MB のメモリの内容を転送するのに最低限必要な時間は約 5 秒であるので，充分に妥当かつ高速に仮想計算ノードの migration が実現できていることがわかる。

４．まとめ

以上の結果から，計算速度を優先する場合には同時に実行するプロセス数は少ないほどよく，電力効率を優先する場合には最低でも物理プロセッサコア数と同数のプロセスを走らせるのが理想的であることがわかる。

また，計算ノードの仮想化に伴うペナルティは主に入出力において発生するものの，少なくとも今回用いたベンチマークの範囲では致命的な性能低下をもたらすものではないことが確認できた。

これに加えて，仮想計算ノードの migration は充分に高速であるため，動的な物理計算ノード間での再配置による性能・消費電力の最適化は有効な手段であると考えられる。

５．今後の課題

現在のクラスタ管理ツールには，各物理・仮想計算ノードの負荷状態を監視して再配置を行う機能が存在しないため，今後はこの機能を開発していく予定である。これには OpenPBS/Torque のジョブの状況や各計算ノードの CPU 負荷，通信量などを一括して監視することが必要になる。

仮想計算ノードの管理についても，現状の静的な構成が必ずしも最適ではない。現状では稼働状態や負荷状態を問わず，同じサイズのメモリを割り当てており，これは物理計算ノード上ではスワップアウトされないため，本当にメモリが必要な仮想計算ノードに多くのメモリ容量を割り当てることができない。さらに，物理あるいは仮想のネットワークインタフェイスを経由した通信は共有メモリを用いた通信と比べて性能が低いため，MPI や pthread を用いて並列化されたプログラムを効率的にハンドリングするためには，仮想計算ノードの CPU 数を 1 に限定せず，必要に応じて増減することも考えられる。

また，今回は LAN 接続され，ディスクイメージを共有する物理計算ノード間での仮想計算ノードの移動が充分に高速であることを示したが，遠隔地間で同様の機能を実現する際にはディスクイメージの共有は現実的でないため，FreeBSD で最近開発された remote replication over TCP/IP の仕組み [7]を応用するなどして，これに対応する手法を開発する必要がある。

参考文献

1）Oracle Corportation: “VirtualBox website”, http://virtualbox.org/

2）Intel Corporation: “Intelligent platform management interface”, http://www.intel.com/design/servers/ipmi/

3）Cluter Resources Inc.: “Torque resource manager”, http://www.clusterresources.com/

4）MPlayer Project web site, http://www.mplayerhq.hu/

5）Larkin MA, Blackshields G, Brown NP, Chenna R, McGettigan PA, McWilliam H, Valentin F, Wallace IM, Wilm A, Lopez R, Thompson JD, Gibson TJ, Higgins DG: “Clustal W and Clustal X version 2.0.” Bioinformatics, Vol.23, pp 2947-2948, 2007.

6）Albayraktaroglu K, Jaleel A, Wu X, Franklin M, Jacob B, Tseng CW, and Yeung D: “BioBench: A benchmark suite of bioinformatics applications.”, In prceedigs of 2005 IEEE International Symposium on Performance Analysis of Systems and Software, pp.2-9, Mar.2005.

7）FreeBSD Project: “HAST - Highly available storage”, http://wiki.freebsd.org/HAST

付　録

A. 物理計算ノードの設定

VirtualBox は System V IPC を使用してフロントエンドと通信を行うため，多数の VirtualBox を同時に実行する場合には，その数以上の System V semaphore を確保しておく必要がある。この値は sysctl コマンドで取得することができるが，変更はできないため，/boot/loader.conf で設定する必要がある。

また，物理計算ノードとして使用した PowerEdge T110 には Ethernet controller として Broadcom BCM5722 (bge ドライバで動作) が搭載されているが，デフォルトの設定では bge ドライバの初期化時に BMC (Baseboard Management Controller) の，Ethernet を経由した通信機能を停止してしまうため，IPMI を利用した電源制御や serial over LAN などの機能を使うためには hw.bge.allow_asf を 1 に設定しておく必要がある。

そのほか，serial over LAN でコンソールを使うための設定を含めると，/boot/loader.conf の記述は以下のようになる。

```
kern.ipc.semmni=40
kern.ipc.semmns=300

ipmi_load="YES"

console="comconsole,vidconsole"
boot_multicons="YES"
hint.uart.0.disabled="1"
hint.uart.1.flags="0x10"
hw.bge.allow_asf="1"
```

VirtualBox のゲスト OS として 64bit 版の FreeBSD を用いるため，BIOS 設定で CPU の VT-x 拡張を有効にしておく必要がある点にも注意が必要である。BIOS ではさらに，console redirection などの設定を行っておけば，電源投入直後から IPMI の serial over LAN 経由での操作が可能である。

B. 仮想計算ノードの作成・複製

VirtualBox は仮想マシンの複製をおこなう機能を持たないが，仮想ハードディスクの複製はサポートされるため，コマンドラインから多数の仮想マシンを作成し，複製したハードディスクを割り当てることで，同様の仮想マシンを多数作成することができる。

たとえば，ハードディスク vmbsd01.vdi を複製して vmbsd02.vdi を作成するならば以下のようにすればよい。

```
% VBoxManage clonehd ¥
    vmbsd01.vdi vmbsd02.vdi --register
```

さらに新しい仮想マシンの仕様を下記のように定める。

- Guest OS: FreeBSD (amd64)
- RAM: 512MB，物理アドレス拡張 off, VRAM 10MB
- Audio: OSS (Open sound system: /dev/dsp)
- Network: bge0 にブリッジ
- PIIX4 compatible IDE controller
  - Primary master: HDD
  - Secondary master: DVD (ドライブは空)

この仕様の仮想マシンを作成してさきほどの仮想ハードディスクを attach するまでの手順は以下のようになる。

```
% VBoxManage createvm ¥
    --name vmbsd02 ¥
    --ostype "FreeBSD_64" --register
% VBoxManage modifyvm vmbsd02 ¥
    --memory 512 --pae off --vram 10 ¥
    --audio oss ¥
    --nic1 bridged --bridgeadapter1 bge0
% VBoxManage storagectl vmbsd02 ¥
    --name "IDE Controller" ¥
    --add ide --controller PIIX4
% VBoxManage storageattach vmbsd02 ¥
    --storagectl "IDE Controller" ¥
    --port 0 --device 0 ¥
    --type hdd --medium vmbsd02.vdi
% VBoxManage storageattach vmbsd02 ¥
    --storagectl "IDE Controller" ¥
    --port 1 --device 0 ¥
    --type dvddrive --medium emptydrive
```

作成した仮想マシンを起動・停止するには,

```
% VBoxManage startvm vmbsd02 ¥
    --type=headless
% VBoxManage controlvm vmbsd02 ¥
    acpipowerbutton
```

のようにする。起動時に“--type=headless”をつけなければ，ディスプレイとキーボードを用いてコンソールにアクセスすることができる。

なお，ディスクイメージその他を共有している状態で teleport 機能により仮想マシンの migration を行う場合には, 同一構成で同じディスクイメージの attach されたふたつの仮想マシンを用意しておき，先に起動したどちらか一方から他方への teleport という形をとることになる点に注意が必要である。