Windows で PC クラスタの構築及びアプリケーション

李　翠敏

1　はじめに

マイクロソフトは，研究開発の分野における新たな発展の働きを助長するとともに，科学者，技術者，そして研究者たちが直面する問題を解消できるようなソフトウェアの開発を進めている．Windows Compute Cluster Server 2003 は導入および操作，ならびに既存のインフラストラクチャやツールとの統合を容易にすることを目標にデサインされている．

本研究では Windows Compute Cluster Server 2003 を用いて，PC クラスタを構築する手法とそのアプリケーショウンの例として流体解析応用について説明する．

2　システムソフトウエア

本研究では，Windows Compute Cluster Server 2003(WCCS) を利用する．これは，PC クラスタのベースになるオペレーティングシステム-Windows Server 2003(CD1) と，PC クラスタの機能を実現するミドルウエア-Compute Cluster Pack(CD2) から構成されている．

- Windows Server 2003 Compute Cluster x64 Edition ：これは Compute Cluster Pack を構築するためのものである．64 ビットのプロセッサーのみ利用可能である．
- Compute Cluster Pack : 並列処理を実現する MPI ライブラリ，計算ジョブを分配するシステム，PC クラスタの管理ツールなどが含まれる．MPI プログラムを開発するための CCP-SDK も含まれる．

3　システム構成

最も単純なシステムは Head Node と Compute Node より構成される．構成トポロジを図 1 に示す．Windows クラスタを構築する要素は Head Node および Compute Node であり，これらの役割を以下に示す．

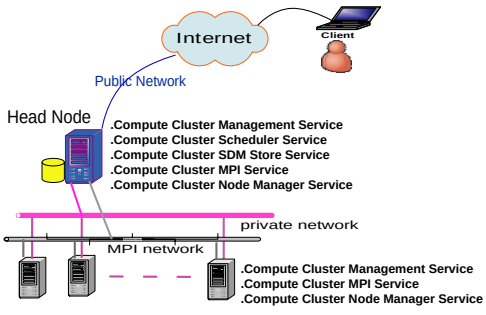

Fig.1　ネットワークトポロジ

- Head Node : PC クラスタのシステム全体を管理するノードであり，このノードからジョブを投入することになる．通常は Head Node と計算ノードは同じドメインでなければならない．本試験で構成する PC の管理に ActiveDirectory を利用している．
- Compute Node : 投入された並列ジョブや分散ジョブを実現するノードである．通常はシステムに複数台設置される．

4　構築前の準備 [1)]

PC クラスタ構築手順を示すの前に本研究で利用するハードウェアとソフトウェアとネットワークについて様々な条件を以下に示す．

1. ハードウェア : Head Ndoe および Compute Node にはいずれも 64 ビット AMD Opteron(tm) Processor 1.79GHz x 2 のプロセッサと 2.0GB メモリのマシンを用いる．自動的に Compute Node をインストルために Remote Installation System (RIS) を用いる．RIS を利用するために Head Node は少なくとも 2 つパーティションを要意する必要である．
2. ソフトウェア : Window Server 2003 x64 (CD1),Compute Cluster Pack(CCP) (CD2).
3. ネットワーク : 本文では単純にするために PC クラスタのネットワークを専用のサブネット内で構築する．

5　構築手順

ここでは Head Node の構築および Compute Node の追加の手順を示す．まず Head Node を構築して，そのあと RIS を用いてクラスタに Compute Node を加える．なお，既存の Windows マシンに CCP をインストールすることで Head Node を構築することが可能であるが，ここでは前節で示した環境のもと，Head Node を構築する．

5.1　Head Node の構築

Head Node ではジョブの管理とクラスタの管理とスケジューリングと資源を提供する．Head Node の構築手順は以下のとおりである．

1. ハードデスクのパーティションは RIS で計算ノードを追加するために少なくとも 2 つパーティション必要である．ここでは C:と D:とする．
2. Windows Server 2003 オペレーティングシステムをインストールする．これは，通常の Windows Server 2003 のインストール手順と同様なので，省略する．
3. Domain Controller を構築する. 図 2 に示すように「Add or Remove a role」をクリクして，Active Directory Installation Wizard で構成する．下記の点を選択して行う．

(a) **Domain controller for a new domain** (default) を選択し，**Next** をクリックする．

(b) **Domain in a new forest** (default) を選択し，**Next** をクリックする．

(c) ドメインの名は任意である，ここでは **haha.com** とする．**Next** をクリックする．後の設定は default として続く．完了したら再起動する．

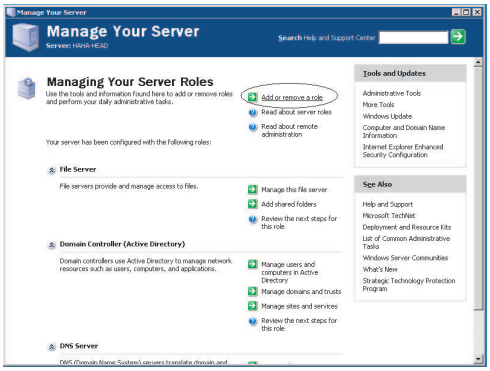

Fig.2　Manage Your Server

4. CD2 からの以下の Compute Cluster Pack をインストールする．CCP のインストールにおいて，インターネットからダウンロードが必要となるためにネットワークの設定を確認する．確認終了後，CD 内の setup.exe を実行する．これより，Wizard 画面が起動するので，指示にしたがってインストールを進める．ここでは Head Node は Compute Node として利用せず，計算は実行しないよう設定する．CCS の Head Node が持つ機能として，Compute Node を自動的にインストールする RIS がある．本システムはこの機能を利用する．RIS は OS と異なるパーティションにインストールする．CD2 には下記の 3 つが含まれており，この順序に一つずつインストールしてゆく．
   (a) Microsoft SQL Server 2000 Desktop Engin(MSDE2000)
   (b) .NET Framework 2.0 RCO
   (c) Microsoft Compute Cluster Pack
   最後に To Do List が表示され，完了する．「Start」をクリックして，図 3 に示す「Compute Cluster Administrator」を見られる．
5. PC クラスタのネットワークの設定を行う．ここでは，ネットワークトポロジを設定する．本研究では，Head Node で NIC を 2 枚利用し，Public Network と Private Network の NIC を選ぶ．

5.2　Compute Node の構築

RIS を用いることで Compute Node の構築は容易である．RIS を利用するために Compute Node はネットワークからリスタートできるようにしておく．これは BIOS で設定する．この後，To Do List で Compute Node を

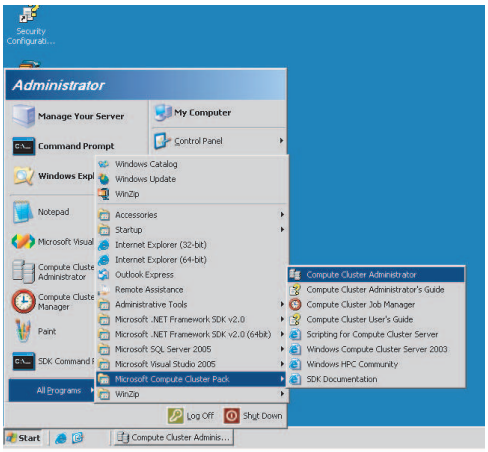

Fig.3　To Do list

クラスタに増加する．手順を以下に示す．

1. To Do List 中の「Add Node」をクリックすると，Wizard 画面が起動させる．
2. 「start RIS」をクリックすると Compute Node の構築が始まる．
3. 追加する Compute Node をリスタートして Compute Node の OS と CCP のインストレーションを行う．インストール途中では Compute Node が再起動される．再起動の後，「Stop RIS」をクリックし，Compute Node は自動的にインストールされる．

完了後，To Do List の Node Management を選択すると Head Node が確認している Compute Node の CPU 数やメモリ容量などのリソース一覧が表示される．Node を選択し，状態を Approve ⇒Resume を設定する．

6　アプリケーション

構築した PC クラスタの計算性能を評価するために，ここでは Linpack[2] と 3 次元流体解析プログラム [3] を用いて試験する．

6.1　HPLlinpack

ここでは問題サイズを 10000，12000，14000，16000 を用いて PC クラスタのメモリを試験する．用いた実行環境を表 1 に示す．

Table1　実行環境

プロセッサ	AMD Opteron Processor 244 1.79GHz　(x 2)
マザーボード	AMD 8131 + 8111
メモリ	2.0GB (x2)
ノード数	2
ソフトウェア	acml-3-5-0-win64.exe (for windows 64bit)
ライブラリ	AMD Core Math Library (ACML) for WindowsR. Built with PGI.

6.1.1　実行方法

Linpack を実行する前に Head Node のコンフィギュレーションを行う．コンフィギュレーションの手法は以下のとおりである．

1. Head Node でディレクトリを作成する．例：c:\PDC
2. このディレクトリを共有する．
3. 環境変数を増加する．図 4 のように設定する．
4. Linpack が必要の「.dat」と「.exe」と「.dll」のファイルを c:\PDC にコピーする．
5. HPL.dat ファイルにおいてパラメータを変更する．
6. 下記のコマンドを実行し，実行結果 HPL.out を出力させる．

```
job submit /numprocessors:4 mpiexec -wdir
\\haha-head\PDC xhpl_acml.exe
```

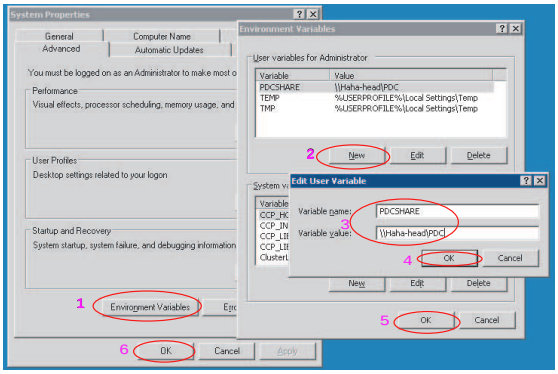

Fig.4　環境変数の設定

6.1.2　パラメータと実行結果

ここではプロセッサ数を 4 用いて PC クラスタの計算性能を検討する．なお，HPL Linpack のパラメータを以下に説明する．用いるパラメータ及び実行結果を表 2 に示す．

- 問題サイズ（N）[4]：一般的に，N の値が大きくなるほど良い結果が得られるが，N の増加に伴いメモリの使用量が増える．
- ブロックサイズ（NB）[4]：ブロックサイズ（NB）は，Linpack で解く問題の粒度である．NB が大きくなると通信量は減少する一方ロードバランスが悪くなる．逆に，NB が小さくなると通信量は増加する一方ロードバランスは良くなる．
- プロセスグリッド（P，Q）[4]:プロセスグリッド（P，Q）は，問題の行列をそれぞれのプロセスにどのように分割するかを示すものである．P と Q の積が実行ノード数となる．P，Q の値は等しい，もしくは P の値より Q の値が大きい方が良い結果が得られる．

Table2　パラメータと実行結果

N	NB	PxQ	Time	Gflop
10000	168	1x1	220.58	3.02
12000	168	1x1	378.99	3.04
14000	168	1x1	591.42	3.09
16000	168	1x1	2640.31	1.03

表 2 内の Time と Gflop を見ると，問題サイズ (N) は大きいほど所要時間が大きくなる，Gflop は問題サイズ 14000 を超えると性能が低下する．これは問題サイズは物理総容量メモリを超えるとディスクにスワップするためと考えられる．

6.2　3 次元流体解析

演算性能の計測には 3 次元流体解析のプログラムを用いた．このプログラムは図 5 のようなファンの解析を行うものである．

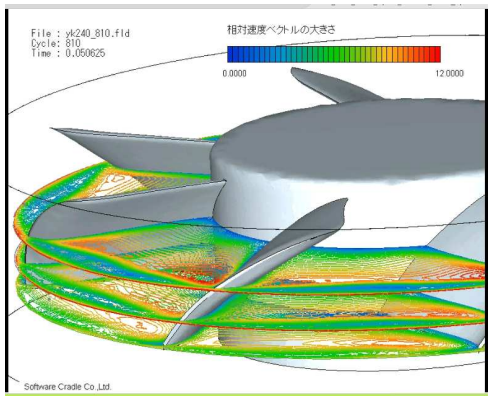

Fig.5　Linpack の実行結果

6.2.1　実行環境

3 次元流体解析プログラムの実行環境は表 3 に示す．

Table3　実行環境

ノード数	Head Nodex1 計算ノード x16
プロセッサ	AMD Opteron Processor 244 (1.8GHz) x2
メモリ	5GB
NIC	Broadcom BCM5704C dual-channel GbE LAN
コンパイラー	Visula C++ 2005
MPI	MS-MPI(1.0.602.1)
行列演算ライブラリ	AMD ACML ライブラリ

6.2.2　実行方法

演算に用いる CPU 数やポルグラムの入力ファイルを指定したテンプレートファイルを作成する．Compute Cluster Job Manager というアプリケーションを利用してファンの流体解析の入力ファイルを選択し，演算に用いる CPU 数を 1,2,4,8,16,32 と変化させながらデータを取得した．

6.2.3　実行結果

解析にかかった時間は表 4 に示す．これらは 2 試行の結果である．

- **速度向上比**
  使用するプロセッサの数を n とすると，1CPU でジョブを実行した時の所要時間 T(1) を，nCPU で

Table4　解析の時間

プロセッサ	1 試行目	2 試行目
1	0.393455	0.39701
2	0.161075	0.161271
4	0.0871414	0.0827469
8	0.0490881	0.0316275
16	0.0325475	0.0316275
32	0.0278122	0.0267708

ジョブを実行したときの所要時間 T(n) で割ったものが速度向上比である．2 試行した速度向上比結果を図 6 に示す．

$$\text{速度向上比} \quad = T(1)/T(n) \qquad (1)$$

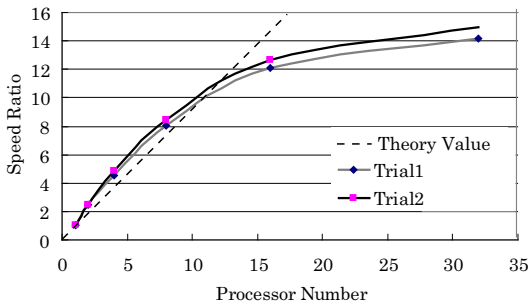

Fig.6　速度向上比

図 6 を見ると，プロセッサ数 2，4，8 の時に速度向上比の値が利用したプロセッサ数を超えている．理想的にはプロセッサ数を n とすれば 1 台の n 倍の速度向上比となるはずであるが，ここでは n 倍以上となるスーパーリニアな結果が得られた．これは一般的に，複数の CPU を利用することによって増えたキャッシュメモリの総容量が原因と考えられている．並列化されたプログラムのサイズがほとんどんキャッシュに収まり，キャッシュが有効利用されたためにスーパーリニアな結果となった考えられた．

- **並列化効率**

並列化効率とはプロセッサの効率的な利用率の目安を示すのもである．2 試行行った計測結果の並列か効率を図 7 に示す．図 7 に見るとプロセッサ数が 2，4，8 のところでは，並列化効率が 100%を超えている．これ以後並列化効率は降下する．

$$\text{並列化効率} \quad = 100 \times \{T(1)/T(n)\}/n \qquad (2)$$

- **オーバーヘッド時間**

式 3 を用いて理想的な速度向上比が取られた時の所要時間 T(1)/n と比較して実際の所要時間 T(n) に含まれるオーバーヘッド時間，T(n)-T(1)/n を計算した．実行結果を図 8 に示す

$$\text{オーバーヘッド時間} \quad = \quad T(n) - T(1)/n \qquad (3)$$

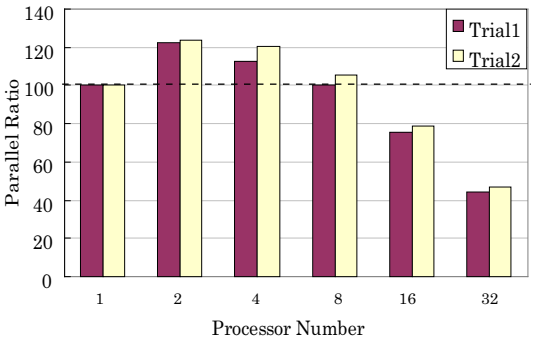

Fig.7　並列化効率

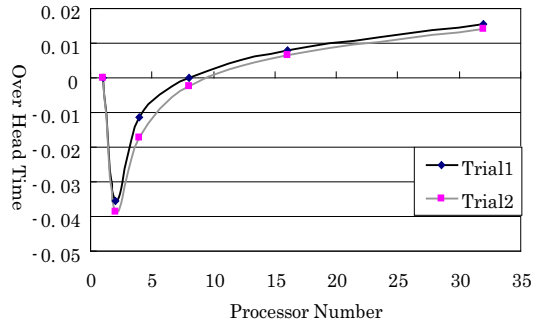

Fig.8　オーバーヘッド時間

図 8 でプロセッサ数 2，4，8 の時実際の時間は理想的な計算時間より少ない．この後プロセッサ数が増えるとノード間の通信などによるオーバヘッド時間が増加する．

7　まとめ

本文は WCCS を用いて PC クラスタを構築手法を説明した．性能を評価するために Linpack を利用して 2 台のノードの総メモリを検証した．また 3 次元流体解析プログラムを利用してプロセッサ数 1,2,4,8,16,32 として速度向上比と並列化効率とオーバーヘッドについて検討した．実行結果に見るとプロセッサ数が 2，4，8 の時速度向上比が理想値以上の結果となり．16 台以上の場合は逆にオーバーヘッドが大きくなり，並列化効率が悪化することが分かった．これはキャッシュサイズによるものと考えられるが，今後検証する必要がある．

参考文献

1) http://technet2.microsoft.com/WindowsServer/en /Library/a651fac4-cf53-4c5b-aa76-a357b81bc9161033.mspx?mfr=true
2) *http://developer.amd.com/acml.aspx*
3) *http://www.cradle.co.jp/*
4) **荒久田 博士, 廣安 知之, 三木 光範** *Opteron* **の** *Linpack Benchmark* **計測** http://mikilab.doshisha.ac.jp/dia/research/report/2003/0702/003/report20030702003.html