VMwareによる仮想Gridコンピューティング環境の性能検証

TL031257　岩熊 寛樹

2月7日提出

目 次

第 1 章　はじめに **3**

第 2 章　Grid コンピューティングについて **4**
2.1 Grid コンピューティングとは 4
2.2 Grid コンピューティングにおける問題点 5
2.2.1 通信速度の問題 5
2.2.2 Grid ミドルウェアの問題点 6
2.2.3 Grid コンピューティングの研究・開発における問題点 6

第 3 章　VMware について **7**
3.1 仮想化とは 7
3.1.1 PC エミュレーション 7
3.2 VMware とは 7
3.2.1 ホスト OS・ゲスト OS 8
3.3 VMware の機能 10
3.3.1 仮想化環境の独立性 10
3.3.2 VMwareTools について 10
3.3.3 クローン作成機能 10
3.3.4 スナップショット機能 11
3.3.5 各種リソースの設定機能 11
3.3.6 仮想ネットワーク設定機能 11
3.4 VMware の Grid コンピューティングへの転用 13

第 4 章　Sun One Grid Engine について **14**
4.1 SGE とは 14
4.1.1 マスターホスト 14
4.1.2 実行ホスト 14
4.1.3 sge_qmaster 15
4.1.4 sge_schedd 15
4.1.5 sge_execd 15
4.2 SGE の導入 15
4.2.1 Fedora Core4 のインストール 16
4.2.2 NFS の設定 17

4.2.3 SGE のインストール 17
4.2.4 MPICH のインストール 20
4.2.5 SGE から MPICH を利用できるように設定する 21

第 5 章 VMware における SGE インストールの検証 22
5.1 検証内容 22
5.2 VMware を使用した SGE インストールの手順の考察 23
5.2.1 実験環境 23
5.2.2 クローン元となるホストのインストール作業 23
5.2.3 Grid コンピューティング環境拡張時の作業 24
5.2.4 SGE インストール時間に関する考察 24
5.3 仮想 Grid コンピューティングの他の環境への移植に関する考察 26
5.3.1 仮想 Grid コンピューティング環境を移植する際の問題点 26
5.3.2 仮想 Grid コンピューティング環境を移植する際の問題解決 27

第 6 章 VMware の設定変更に伴う仮想マシンの性能への影響検証 36
6.1 実験環境 36
6.2 リソース設定変更に伴う影響 37
6.2.1 実験目的 37
6.2.2 使用したベンチマークソフト 38
6.2.3 実験方法 41
6.2.4 実験結果 41
6.2.5 考察 47
6.3 ネットワーク環境の設定変更に伴う影響 49
6.3.1 実験環境 49
6.3.2 使用したベンチマークソフト 50
6.3.3 実験内容 51
6.3.4 実験結果および考察 51

第 7 章 VMware の時間的・資源的コスト削減に対する総評 57

第1章　はじめに

近年、Grid コンピューティングとよばれる技術が注目されてきている。しかし、この Grid コンピューティングはまだまだ発展途上の技術であり、多くの研究・開発の余地を残している。しかし、Grid コンピューティングの研究・開発を行うにはいくつかの問題をクリアしなければならない。(詳しいことは第 1 章に述べる。)

本論文では、それらの問題解決の手段として仮想化と呼ばれる技術が使えるのではないかと考え、実際に仮想化技術を実現するソフトウェアの 1 つである VMware を用いて、仮想 Grid コンピューティング環境を構築し、Grid コンピューティングの研究・開発に対する VMware の有用性の検証を行った。

第2章　Grid コンピューティングについて

2.1　Grid コンピューティングとは

Grid コンピューティングとはネットワークを経由して、分散している複数のコンピュータ資源を仮想的に一つのコンピュータ資源として見なし、統合・制御することで高性能コンピュータ並みの能力を発揮させるというものである。

複数のコンピュータ資源を仮想的に 1 つのコンピュータ資源とみなすと述べたが、実際には複数のコンピュータ資源に対し並列処理を行わせ、各コンピュータ資源より得られた結果を結合することによって高性能コンピュータ並みの処理速度を実現する。

この並列処理を実行させる際に、ユーザーはネットワークの先に接続されているコンピュータがどういった性能を持っているのか、現在の状態はどうなっているのか、どのコンピュータに仕事を割り振れば効率よく処理を実行できるのかといった事柄を意識する必要はない。それらの事柄は Grid ミドルウェアとよばれるソフトウェアが判断し並列処理を実行してくれる。(図 2.1 参照)

また、近年 Grid コンピューティングが注目され始めている理由には、個人が所有するコンピュータ端末の高性能化が背景にある。これらの端末において、一般的なユーザーの使用状況においてはそのリソース（CPU・メモリ・2 次記憶装置といったもの）を完全に使い切ることはまれである。つまり、端末は常にその性能の何割かは遊んでいる状態となっている。

その反面、暗号解読・医療研究いった複雑な処理を必要とするプロジェクトにおいては大規模な高性能コンピュータを必要としているが、高性能コンピュータを導入することはコスト面において大きな問題を抱えている。

そこで、一般ユーザーが抱えている余剰リソースを Grid コンピューティングにより統合・制御することにより仮想的な高性能コンピュータとして、これらの高性能コンピュータを必要とするプロジェクトに転用しようという動きがでてきた。

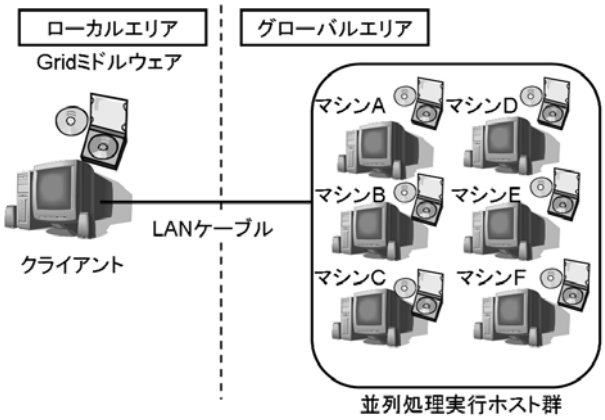

図 2.1: Grid コンピューティングのイメージ

2.2 Grid コンピューティングにおける問題点

前節までの内容を鑑みるに Grid コンピューティングは大筋においてすばらしい技術であるように見えるが、前節までの内容はあくまでも Grid コンピューティングの理論的概要である。実際の Grid コンピューティングは未だに発展途上の技術であり多くの問題を抱えている。以下にその主たる問題点を示す。

- 通信速度の問題
- Grid ミドルウェアの問題点
- Grid コンピューティングの研究・開発における問題点

2.2.1 通信速度の問題

Grid コンピューティングは端末同士の接続方法がネットワーク経由であるがゆえにその通信速度は端末間のネットワーク環境（通信速度・バンド幅といったもの）に依存する。

そのため、通信速度が低かったり、バンド幅が狭ければ一度の通信で送受信できるデータ量が制限されてしまうため、どうしても並列処理の処理速度もあわせて制限されることとなる。

Grid コンピューティングは将来的にネットワークの通信速度・バンド幅などが無限大に成長していくことを前提とした技術である。したがって、現状では理想的な通信速度での運用は不可能である。

2.2.2 Grid ミドルウェアの問題点

2.1 節でも一度触れたが、Grid コンピューティングを実現させるためには Grid ミドルウェアと呼ばれるソフトウェアを用いる。先の説明において Grid ミドルウェアがネットワークの先に接続されたコンピュータの状態を判断し並列処理を実行してくれると説明しているが、現状の Grid ミドルウェアは先の説明通りの処理を実行するには不十分な性能しか保持していない。

Grid ミドルウェアの問題点の例としては、接続されているどのコンピュータに仕事を依頼すれば効率よく並列処理を実行できるかを判断する能力、いわゆるスケジューリングに関する能力が、現状では既存のごく簡単なスケジューリングに基づいた処理はできるものの、実行するジョブに最適なスケジューリングを判断し、実行するといった複雑な処理を行うことは難しい。

このように、Grid コンピューティングがその概念どおりの目的を達成するためには Grid ミドルウェアの性能不足を解消することが必要不可欠である。

2.2.3 Grid コンピューティングの研究・開発における問題点

2.2.2 節をはじめとして、これまでに挙げてきた問題点を解消するために Grid コンピューティングは多くの研究・開発課題を抱えている。しかし、Grid コンピューティングの研究・開発を行うためには、多くのコストを必要とする。ここでは資源と時間のコストについて注目し、その問題点を挙げていくものとする。

まず 1 つ目に、資源的コストに関する問題点であるが Grid コンピューティングの性質上、その研究・開発を行うためには複数の異なる性能を備えたコンピュータを実験環境として用意することが望ましい、それなりの規模を持つプロジェクトであるならば、大きな問題にならないかもしれないが小規模の研究室レベルではその環境を揃えるだけでも大変なコストとなりえる。

2 つ目に、時間的コストに関する問題点である。これまでに度々述べたように、Grid コンピューティング環境を構築するためには Grid ミドルウェアを必要とする。しかも、Grid ミドルウェアは使用するソフトによってまちまちではあるが、概ね Grid コンピューティング環境を構築する全てのコンピュータにインストールする必要性がある。したがって、一度のインストールに時間がかかる Grid ミドルウェアを使用する際には研究を始める準備段階で多くの時間を消費してしまという問題点を抱えている

第3章　VMwareについて

この章では2.2.3節での述べた時間的・資源的コストを軽減するために有用であると考えられる。VMwareと呼ばれる仮想化ソフトについて説明し、なぜ、有用であると考えられたのかを述べる。

3.1　仮想化とは

VMwareのことを知るためには、まず、仮想化と呼ばれる技術について知る必要がある。

本論文内で仮想化といった場合には、1台のハードウェア上で複数のOSを実行できるようにすることを指す。特に今回は仮想化技術のなかでも仮想マシン（ヴァーチャルマシン）を実現するPCエミュレーション型の仮想化ソフト、VMwareを取りあげる。

3.1.1　PCエミュレーション

PCエミュレーションとは1台のコンピュータの資源を何らかの方法を用いて分割し、分割した資源を用いてソフト的にもう1台コンピュータが存在するかのように見せかける技術である。この際に見せ掛け上作成されたコンピュータを仮想マシンと呼び、この仮想マシンは実際のコンピュータと同等の資質を持つことが可能である。

つまり、PCエミュレーションを用いた仮想化とは、1台のマシンを複数台のマシンとして扱えるようにする技術である。

3.2　VMwareとは

VMwareとはVMware社が発売している市販の仮想化ソフトウェアである。PCエミュレーション型のソフトウェアであり、VMwareによってソフト的に再現された仮想マシンは非常に高速に動作するように工夫がほどこされている。

動作環境としてはホストOSとしては、Windows版とLinux版が用意されており、ゲストOSとしてもWincows・Linuxの双方が利用可能である。

3.2.1 ホスト OS・ゲスト OS

- ホスト OS

  ホスト OS とは VMware がインストールされている側の OS のことである。実機 (リアルマシン) にインストールされている OS と言い換えても良い。以降ホスト OS がインストールされているマシンのことをホストマシンと呼ぶ。

- ゲスト OS

  ゲスト OS とは、VMware によって作成された仮想マシン上にインストールされた OS のことである。

以下にコンピュータ内において VMware がどのようにして仮想マシンを構築しているのか、そのイメージを図 3.1 にして示す。

図 3.1 における、アプリケーションとはホスト OS が利用するアプリケーションであり、VMware もその内の 1 つである。

また、点線で囲まれている部分がそれぞれ VMware によって構築された仮想マシンであり仮想マシン内のアプリ 1・2 はそれぞれの仮想マシン上にインストールされているゲスト OS 用のアプリケーションである。

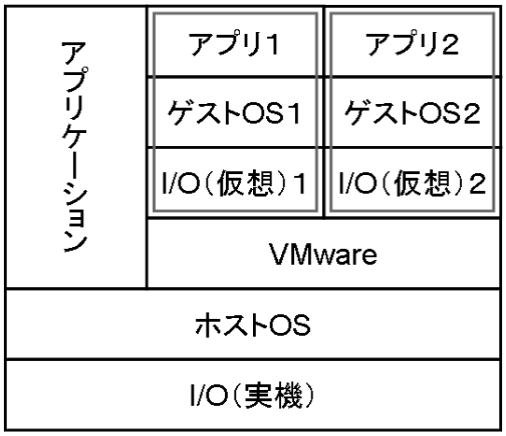

図 3.1: VMware による仮想マシン構築イメージ

さて、文章だけでは VMware が提供しているサービスがどのようなものか掴みずらいと考えらられるため実際に VMware を用いて WindowsXP(ホスト OS) 上で Fedora Core4(ゲスト OS) を起動させている画像を図 3.2 に示す。

ウィンドウとして起動しているアプリケーションが VMware であり、その内部に表示されているのが操作中の仮想マシンの画面である。

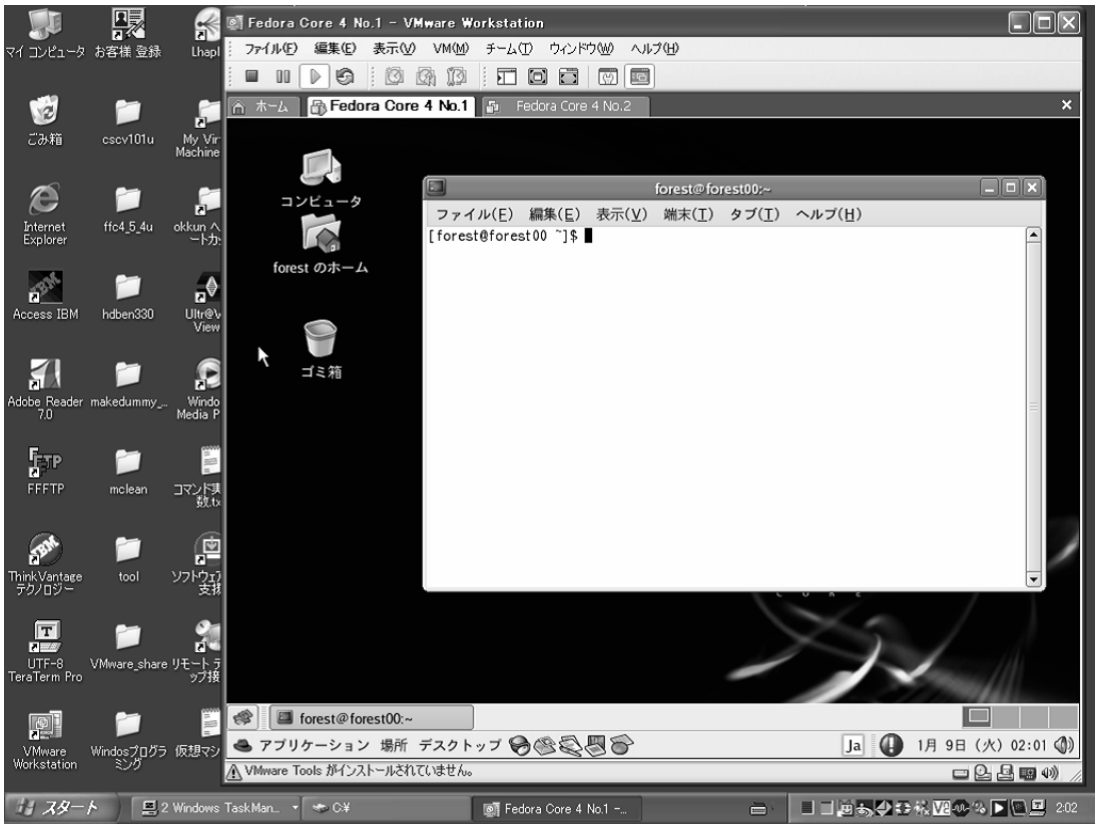

図 3.2: VMware 動作画面

3.3 VMwareの機能

この節ではVMwareの特徴的な機能について説明する。本節内で説明するいくつかの機能を用いることで筆者はGridコンピューティングの研究・開発における資源的・時間的コストを軽減できるものと考える。

3.3.1 仮想化環境の独立性

VMwareによって構築された仮想マシン内で、何らかの問題、例えばウィルス感染やシステムクラッシュが起きたとする。その場合でも、それらの問題による影響は仮想マシン内だけの問題であり、ホストマシンや問題が発生した仮想マシン以外の仮想マシンには全く影響を及ぼすことがない。

つまり、図3.1において点線で囲まれている仮想マシン内部で問題が発生しても枠線内だけのことであり、枠線外部には影響しない。

3.3.2 VMwareToolsについて

VMwareToolsはVMwareにおいてゲストOSを快適に利用するためにVMware社がインストールを勧めているツールである。構築した仮想マシンごとにインストールの必要があり、主な機能としてはホストOSとゲストOS間における共有フォルダの作成を可能にする機能、ドラック&ドロップによるホストOSとゲストOS間のファイルのコピーを可能にする機能。グラフィカルインターフェースの機能向上が挙げられる。

3.3.3 クローン作成機能

VMwareには仮想マシンの複製を作成するクローン機能がある。クローンには、リンククローンと完全クローンの2種類が存在する。

リンククローンとは元となる仮想マシン、オリジナル仮想マシンの仮想ディスク[1]をクローン仮想マシンと共有しつつ、クローンに対して行われた修正は別の仮想ディスクファイルに対して書き込むという方式である。この方式の場合は、仮想ディスクの変更部分だけを差分として持てばよいのでディスク容量の節約となる。また、オリジナルとクローンとで同一の仮想ディスクを用いているためオリジナルとクローンの間ではマシンの独立性は保てない。つまり、共有している仮想ディスク部分で問題が発生した場合、オリジナル・クローンの双方で問題が発生する。

[1] VMwareの仮想マシンの管理方法は、仮想マシン毎に仮想マシンの情報を記憶するフォルダを用意する。そのフォルダ内には仮想マシン内のHDD内容を記憶するための仮想ディスクファイルが存在する。ここで言う、仮想ディスクとはこの仮想ディスクファイルのことである。

完全クローンとは、クローン元となった仮想マシンを完全に複製してしまう。リンククローン方式と違い元となる仮想マシンと同等のディスク容量を必要とするが、作成されたクローンマシンは独立した仮想マシンとして扱うことができるのでリンククローンのような発生した問題の共有が起こることはない。

3.3.4 スナップショット機能

仮想マシンの環境を保存しておく機能である。VMware にはサーバー運用を目的としてものや研究目的に利用するものなのどいつかの種類が存在するが、なかでも今回使用している Workstation(研究・実験用) では複数のスナップショットを作成することが可能である。

スナップショットを作成すると、数 MB〜数百 MB のファイルが作成される。この際のファイルのサイズは保存する環境によってまちまちである。環境を復元する際には VMware のツールバーから項目 [VM] → [スナップショット] → [スナップショットマネージャー] と選択し、表示されるウィンドウ内で復元させたい環境を選択すると、数十秒のうちに選択した環境を復元してくれる。

研究・開発のように環境を頻繁に変更してなくてはいけない場合には大いに役に立ってくれる機能である。

3.3.5 各種リソースの設定機能

VMware には既存の仮想マシンのリソース (メモリ容量、プロセッサの数、仮想ディスクの接続方式、仮想マシンが保持するデバイスなど) を仮想マシンを起動する前に、設定・変更する機能がある。

この機能は性能の異なるコンピュータを複数台準備することが望まれる Grid コンピューティングの研究・開発の環境において、さまざまなリソース設定のマシンを用意する助けとなると考えられる。

3.3.6 仮想ネットワーク設定機能

VMware のネットワーク構築方法には 3 つの方式からの選択が可能となっている。

ブリッジ方式

ブリッジ方式は、イーサネットのレイヤー (レイヤー 2) で、VMware 側の仮想ネットワークとホストマシンの物理ネットワークを接続する。IP アドレ

スよりも下位のレイヤーでの接続であるため、例えば物理ネットワーク側に接続されている DHCP サーバーから仮想マシンに対して動的に IP アドレスを割り当てることができる。あたかも物理的にマシンをもう 1 台追加したように仮想マシンを利用することができる。

難点として、既存の物理ネットワークに影響を及ぼす可能性がある。(図 3.3 参照)

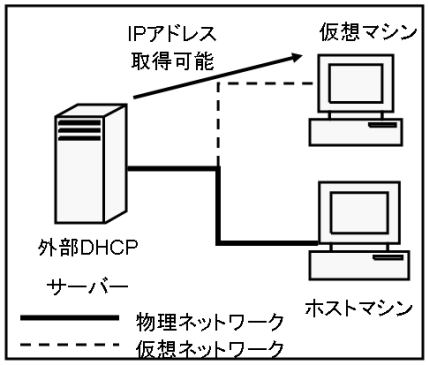

図 3.3: ブリッジ接続イメージ

ホストオンリー方式

ホストオンリー方式とは、仮想マシンとホストマシンの間でだけ LAN で接続される構成である。ブリッジ接続の難点である物理ネットワークへの影響はないが、外部ネットワークへの接続はできない。(図 3.4 参照)

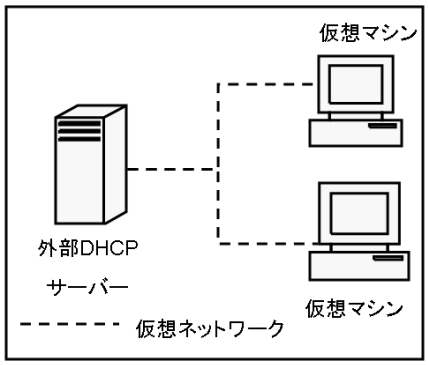

図 3.4: ホストオンリー接続イメージ

NAT 方式

NAT 方式とは、ホストオンリーのネットワークに外部ネットワークへのゲートウェイを接続したような構成である。ちょうど、LAN にインターネット接続用のブロードバンドルーターを追加したようなものでホストマシンで使用している IP アドレスを外部ネットワーク接続時に仮想マシンマシンでも共有して利用可能にした方式である。ただし、NAT 接続は IP レベル (レイヤー 3) であるため、ブリッジ構成のようにダイレクトに物理ネットワークに接続することができない。(図 3.5 参照)

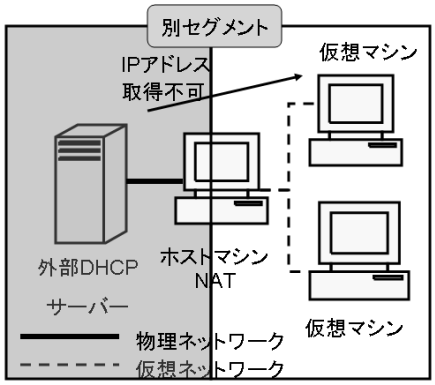

図 3.5: NAT 接続イメージ

仮想ネットワークの通信速度設定機能

仮想ネットワークに関してはホスト間を接続する仮想 LAN ケーブルの通信速度を、100Mbps・1Gbps・無制限のといった値に変更することが可能である。

この機能を用い仮想的に通信速度を無制限にすることで、2.2.1 節で述べた将来的に見込まれている Grid コンピューティングの稼働環境として理想的な環境を提供するこが出来るのではないかと考えられる。

3.4 VMware の Grid コンピューティングへの転用

3.3 節で述べた機能のうちクローン作成機能、各種リソース設定機能・仮想ネットワーク設定機能を用いることで 2.2.3 節で述べたような資源的・時間的コストを軽減することが可能であると考えられる。

第4章　Sun One Grid Engineについて

この章では、本研究において Grid コンピューティング環境を構築するに当たり、利用した Sun One Grid Engine(以下 SGE) の概要、および、実際に SGE を導入する際のインストール手順を説明する。

4.1　SGE とは

SGE はサン・マイクロシステムズが無償で配布している、PC クラスタのジョブ管理のためのミドルウェアであるが、Grid ミドルウェアとしても利用可能である。SGE はユーザーから受け取ったジョブを SGE 内で定義されているジョブの割り当てルールに基づいて、各コンピュータのリソースを効率的に利用できるよう仕事を自動的に分配する機能を持つ。

SGE によって構築される Grid コンピューティング環境下のコンピュータ（ホスト）はその役割から 2 つの種類に分別することができる。マスターホスト（管理・実行依頼ホストとも呼ばれる）と実行ホストである。

また、各実行ホストはマスターホスト上の SGE がインストールされている SGE_ROOT ディレクトリを NFS などによって共有しておく必要性がある。(図 4.1 参照)

4.1.1　マスターホスト

マスターホストは仕事を依頼する側のコンピュータである。マスターホストは実行ホストの状態を監視し、仕事を依頼する際にどの実行ホストに依頼すれば最適にリソースを利用できるか判断し、仕事を各実行ホストに分配する。

4.1.2　実行ホスト

実行ホストは依頼された仕事を実行する側のコンピュータである。マスターホストに対し自己の状態を報告すると共に、仕事を依頼された場合、仕事の実行およびマスターホストに対して結果を報告する処理を行う。

4.1.3 sge_qmaster

sge_qmaster デーモンは、マスターデーモンとも呼ばれ、各実行ホスト（ホスト上の sge_execd）より送られてくる実行ホストの状態の管理および sge_schedd によるスケジューリングに基づき各実行ホストに仕事を依頼する役割をもつ SGE の中核となるデーモンである。

4.1.4 sge_schedd

sge_schedd デーモンは、スケジューラデーモンとも呼ばれ、sge_qmaster より受け取った各実行ホストの状態データを基にどの仕事をどの実行ホストに依頼するかを決定する役割を持つデーモンである。

4.1.5 sge_execd

sge_execd は実行デーモンとも呼ばれ、それが動作しているホストに対する仕事の実行を行う。また、定期的にホストの状態・負荷などのデータをマスターホスト（ホスト上の sge_qmaster）に対して転送する。

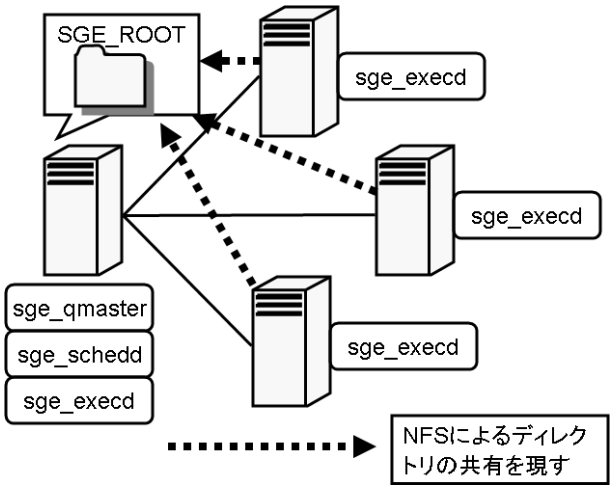

図 4.1: SGE 構成イメージ

4.2 SGE の導入

今回、SGE を導入するのは Linux のディストリビューションの 1 つである Fedora Core4 である。Fedora Core4 をインストール環境に選んだ理由は、

Fedora Core シリーズを VMware のゲスト OS とした際、Fedora Core4 以降のヴァージョンであった場合、VMwareTools のインストールがサポートされておらず VMwareTools インストール後の動作が不安定になってしまったためである。

Grid コンピューティング環境を構築するにあたり SGE を各ホストにインストールするためには次のような作業を行わなければならない。(SGE のインストールには手動で行う方法と自動で行う方法の 2 種類があるが、以下に示す手順は手動によるインストール方法の手順である。)

また、本実験において SGE を導入したのは次の PC 環境 (表 4.1) である。

製品名	IBM 816829I
CPU	Intel(R) Pentium(R)4 3.40GHz
メモリー	SD-RAM512MB(512MB × 1)
グラフィックボード	Intel(r)82945G Chipset Family Graphics chip Accelerate
OS	WindowsXP Professional Ver.2002 Service Pack2
HDD 1	ST3300831AS
HDD 2	ST3300831AS
HDD 3	WDCWD1600JS-08MHBO

表 4.1: 実験環境

() 内は本実験環境における作業に要するおおよその時間である。

1. 各ホストへの Fedora Core4 のインストール（1 時間）
2. NFS の設定（5 分）
3. SGE のインストール（マスターホスト：15 分〜20 分　実行ホスト：5 分〜10 分）
4. MPICH のインストール（5 分）
5. SGE から MPICH を利用できるように設定する (5 分)

1 台のホストに SGE をインストールするためにはおおよそ 1 時間 20 分〜1 時間 30 分程度の時間を要する。

4.2.1 Fedora Core4 のインストール

Fedora Core4 のインストール関しては、注意点のみを記述するものとする。注意点は以下の 3 点である。

- ホスト名の変更および固定 IP アドレスを設定する

- ファイヤーウォールの設定を無効としておく
- アプリケーション選択時に LAM の項目のチェックをはずしておく

ホスト名・IP アドレスの変更は SGE を利用すすために、ホスト名による IP アドレスの判別が必要となるためである。

ファイヤーウォールの設定を無効にするのは、NFS によるディレクトリ共有を行う際にファイヤーウォールによって実行ホストからマスターホストへの接続を拒否することがないようにするためである。

アプリケーション選択時に LAM の項目のチェックをはずすのは、SGE より MPI を利用して並列処理の仕事を依頼しようとする場合に LAM がインストールされていた場合に不具合が発生するためである。

4.2.2 NFS の設定

「/etc/hosts.deny」に全アクセスを拒否するように記述

```
###/etc/hosts.deny###
ALL:ALL
```

「/etc/hosts.allow」にアクセスを許可するホスト名または IP アドレスを記述

```
###/etc/hosts.allow###
ALL:ホスト名
```

NFS のアクセス制御を行う設定ファイル「/etc/exports」を編集。公開するディレクトリとアクセスを許可するホスト名、権限などを記述する。

```
###/etc/exports###
/home/n1admin/n1grid[1]ホスト名 (rw)
```

ブート起動時に NFS を自動起動するように設定する。

```
# chkconfig nfs on
```

4.2.3 SGE のインストール

サン・マクロシステムズより SGE のパッケージを入手した後、Fedora Core4 へのインストールに必要なファイルを SGE_ROOT ディレクトに展開する

[1]本実験環境では SGE_ROOT ディレクトリとして/home/n1admin/n1grid を設定している。設定環境に合わせてこの部分は変更の必要あり

```
$ [2]mkdir /home/n1admin/n1grid
$ cd /home/n1admin/n1grid
$ tar zxvf n1ge-6-0u8-common.tar.gz
$ tar zxvf n1ge-6-0u8-bin-linux24-i586.tar.gz
```

マスターホスト・実行ホスト共通部分

「/etc/services」を編集し、SGE 用のポートを設定する。

```
### /etc/services ###
sge_qmaster 536/tcp
sge_execd 537/tcp
```

「/etc/hosts」にマスターホストおよび実行ホストのホスト名・IP アドレスを登録する。

```
### /etc/hosts ###
# 127.0.0.1 localhost.localdomain 自マシンホスト名[3]
マスターホスト IP アドレス マスターホスト名 マスターホストエイリアス名
実行ホスト IP アドレス 実行ホスト名 実行ホストエイリアス名
```

マスターホストのみ

SGE マスターホストのインストールプログラムを実行する。インストールプログラムは root 権限で実行する。また、SGE_ROOT ディレクトリの所持者は SGE 管理用ユーザーでなければならない。

```
# cd /home/n1admin/n1grid
# ./install_qmaster
```

以上のコマンドを打ち込むと、対話型でいくつかの質問に答えながらインストールを進めていくこととなる。例として本実験環境における各質問への返答例を示す。

[2]SGE をインストール際には、SGE の管理者用ユーザーを作成する必要がある。本実験環境においては n1admin を SGE 管理者ユーザーとして作成した。$ を先頭に添付しているコマンドは SGE 管理者権限で行い、# を先頭に添付しているコマンドは root 権限で行うコマンドである。

[3]127.0.0.1 localhost.localdomain 自マシンホスト名　が既に存在する場合のみコメントアウトする。

[4]2 回目以降は no を選択する

設定名	設定値
Grid Engine admin usera ccount	n1admin
Grid Engine cells	default
Grid Engine qmaster spool directory	no
Verifying and setting file permissions	yes or no[4]
a single DNS domain	yes
a spooling method	berkeleydb
Grid Engine group ID range	20000-20100
add shadow host	no

表 4.2: SGE 対話型インストールの質問に対する返答

SGE の環境設定スクリプトを実行する。

```
$ ./home/n1admin/n1grid/default/common/settings.sh
```

これにより以下で使用する「qconf」などの SGE コマンドのパスの設定等が行われる。

SGE ジョブ投入ホストを登録する。

```
$ qconf -as ホスト名
```

SGE の実行ホストを登録

```
$ qconf -ah ホスト名
```

登録する実行ホストの分だけ実行する。

実行ホストのみ

SGE インストールディレクトリ「SGE_ROOT」を作成し、マスターホスト上の SGE_ROOT ディレクトリを NFS によってマウントし共有ディレクトリとする。

```
$ mkdir /home/n1admin/n1grid
$ mount -t nfs マスターホスト名:/home/n1admin/n1grid /home/n1admin/n1grid
```

ブート起動時に自動的にマスターホストの SGE_ROOT ディレクトリを NFS マウントするようにするために「/etc/fstab」を編集する。

```
### /etc/fstab ###
マスターホスト名:/home/n1admin/n1grid/ /home/n1admin/n1grid nfs defaults 0 0
```

SGE 実行ホストのインストールプログラムを実行する。インストールプログラムは root 権限で実行すること。

```
# cd /home/n1admin/n1grid
# ./install_execd
```

対話型でインストールが行われるが、通常は全ての質問に対して yes または Enter キーの入力だけで良い。

4.2.4 MPICH のインストール

LAM のインストールを行わなかった場合、MPI を利用するために必要な MPICH のインストールも行われないため、MPICH のインストールを手動で行う必要がある。インストール手順は以下の通り。

MPICH のソースを入手する。

$$http://ww-unix.mcs.anl.gov/mpi/mpich1/$$

からソースをダウンロードしてくる。

ソースを展開し、ディレクトリに移動

```
$ tar zxvf mpich.tar.gz
$ cd mpich-1.2.7p1
```

コンパイルする。configure を実行する際に指定するディレクトリは各実行ホストに MPICH をインストールしなくていいように、NFS で共有するマスターホストの SGE_ROOT ディレクトリ下のディレクトリを指定するものとする。

```
$ mkdir /home/n1admin/n1grid/share
$ ./configure –prefix=/home/n1admin/n1grid/share/mpich
$ make
# make install
```

インストールした MPICH のパスが通るように、「/etc/profile.d/mpich.sh」を編集する。

```
### /etc/profile.d/mpich.sh ###
#!/bin/sh
export PATH=/home/n1admin/n1grid/share/mpich/bin:$PATH
```

「/etc/profile.d/mpich.sh」編集後、編集したファイルのパーミッションを設定する。

```
# chmod 755 /etc/profile.d/mpich.sh
```

4.2.5 SGE から MPICH を利用できるように設定する

並列計算環境設定のテンプレートファイル「mpich.template」を編集する。

```
$ cp /home/n1admin/n1grid/mpi/mpich.template  /mpich.template
```

slots の行にある＜ the_number_of_slots ＞を最大計算台数に変更する。start_proc_args、stop_proc_args の行にある＜ your_sge_root ＞を SGE_ROOT ディレクトリに変更する。(以下、太文字部分が変更点である。)

```
### mpich.template ###
pe_name mpich
slots 99
user_lists NONE
xuser_lists NONE
start_proc_args /home/n1admin/n1grid/mpi/startmpi.sh -catch_rsh $pe_hostfile
stop_proc_args /home/n1admin/n1grid/mpi/stopmpi.sh
allocation_rule $round_robin
control_slaves TRUE
job_is_first_task FALSE
urgency_slots min
```

SGE に並列計算環境を追加する。

```
$ qconf -Ap mpich.template
```

クラスタキューを設定する。

```
$ qconf -mq all.q
```

pe_list の行に並列環境 mpich を追加する。また、shell の行を/bin/bash に変更する。

第5章　VMwareにおけるSGEインストールの検証

この章では、第3章において説明したSGEのインストール時間をVMwareを用いることによってどの程度短縮することが出来るのかを、実際に、仮想Grid コンピューティング環境 (仮想 Gird コンピューティング環境のイメージ図を図 5.1 に示す) を構築することによって確認した。また、実験段階において VMware による仮想 Grid コンピューティング環境を他者が保有する VMware 環境に移植するといった利用法も見えてきたため、そちらの利用法に関しても述べる。

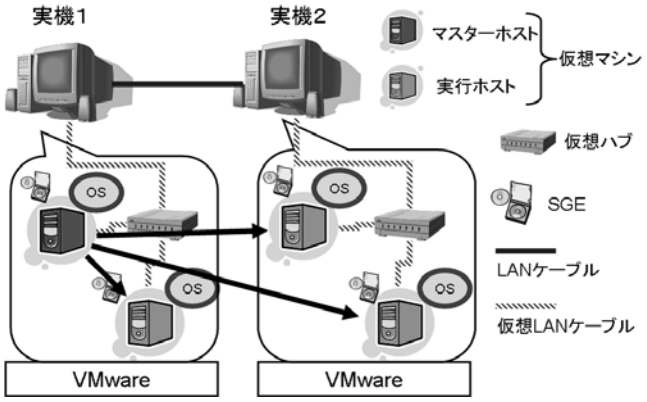

図 5.1: 仮想 Grid コンピューティング環境イメージ

5.1　検証内容

4.2 節において 1 台のホストに SGE を導入するためにはおおよそ 1 時間 20 分〜1 時間 30 分程度の時間を必要とすることを述べた。また、この作業時間は一連のインストール作業にかなり手馴れた上での時間である。筆者が 1 台目のマスターホストをインストールし、諸々の設定を終えるまでには 3 日ほどの時間を要した。

手馴れてしまった後であったとしても、5 台のコンピュータに SGE インストールするには 5 〜 6 時間程度の時間を必要とする。2.2.3 節でも述べたが、

やはりこの時間的コストはGridコンピューティングの研究・開発を行う上で大きな問題となる。

そこでVMwareを用いてマスターホストとしてSGEをインストールした仮想マシンと、実行ホストとしてSGEをインストールした仮想マシンを用意し、仮想Gridコンピューティング環境を拡張する（ホストの台数を増やす）際には、それらの仮想マシンの完全クローンを作成することによってSGEインストールにかかる時間的コストをどの程度軽減できるかを検証する。

加えて、VMwareによって作成した仮想Gridコンピューティング環境を他のVMwareインストール済みの環境に移植する際の検証も行う。

5.2 VMwareを使用したSGEインストールの手順の考察

詳細な手順は4.2節と同一であるためここでは大まかな手順と作業に必要なおおよその時間のみを記述するものとする。

5.2.1 実験環境

本実験は以下に示すPC環境 (表5.1) において、VMware Workstation ver.5.5.1を用いて仮想Gridコンピューティング環境を構築している。

製品名	IBM 816829I
CPU	Intel(R) Pentium(R)4 3.40GHz
メモリー	SD-RAM2G(1G × 2)
グラフィックボード	Intel(r)82945G Chipset Family Graphics chip Accelerate
OS	WindowsXP Professional Ver.2002 Service Pack2
HDD 1	ST3300831AS
HDD 2	ST3300831AS
HDD 3	WDCWD1600JS-08MHBO

表5.1: 実験環境

5.2.2 クローン元となるホストのインストール作業

1. Fedora Core4のインストール（１時間）
2. NFSの設定（５分）
3. SGEインストール前にスナップショットを作成（１分）

4. マスターホストの SGE インストール作業（１５分〜２０分）
5. MPICH のインストール（５分）
6. SGE から MPICH を利用できるように設定する（５分）
7. マスターホストの完全クローンを作成（**[1]）
8. スナップショット機能を用いて SGE インストール前の環境を復元（１分）
9. 実行ホストの SGE インストール作業（５分〜１０分）

以上の作業によってマスターホスト１台と実行ホスト１台を用意することが可能であると考えられる。

5.2.3 Grid コンピューティング環境拡張時の作業

1. 必要なホスト（マスターないしは実行）の完全クローンを作成する。(**[2])
2. マスターホストにおいて、必要に応じて/etc/hosts、/etc/hosts.allow、/etc/exports ファイルの再編集（5 分）

5.2.4 SGE インストール時間に関する考察

上記の手順に従って、仮想 Grid コンピューティング環境を構築しようとしたところ完全クローンはクローン元となる仮想マシンの MAC アドレスまでも元のまま複製してしまうため、仮想 Grid コンピューティング環境内に同一の MAC アドレスを持つ複数のコンピュータが存在するという本来ありえな状態が発生してしまう。そのため完全クローンを作成した場合、仮想 Grid コンピューティング環境を構築することができない。

しかし、完全クローンを作成するのではなくクローン元となる仮想マシンのデータが保存されているディレクトリをコピーした場合、コピーマシンの初回起動時に MAC アドレス等の設定を自動的に変更するか否か選択が可能となるため、コピーマシンを用いることで前述した手順で仮想 Grid コンピューティング環境が構築できることが確認できた。

仮想 Grid コンピューティング環境を構築するために要する時間であるが、マスターホスト 1 台、実行ホスト 1 台の環境を構築するのに要した時間は 1 時間 40 分ほど、マスターホスト 1 台、実行ホスト 2 台の環境を構築するのに

[1] 仮想マシンの HDD 容量により変動する。本実験環境では 10GB で 5 分程度の時間を要した。

[2] 仮想マシンの HDD 容量により変動する。本実験環境においては 10GB で 5 分程度の時間を要した。

要した時間は 2 時間であった、実機を用いてこれらの環境を用意するためには前者 2 時間 40 分、後者 3 時間 50 分程度の時間を必要とするため前者の環境では 40 分程度、後者の環境では 1 時 50 分程度の時間的コストの減少が確認できた。(表 5.2 に同様の内容をまとめる。)

※以下の表中において M はマスターホスト、E は実行ホストを表す。

	M：1 台　E：1 台	M：1 台　E：2 台
Grid コンピューティング環境	2 時間 40 分	3 時間 50 分
仮想 Grid コンピューティング環境	1 時間 40 分	2 時間

表 5.2: SGE インストールに要する時間比較

また、時間短縮の主たる要因として Fedora Core4 のインストールを 1 台目以降は行わなくてもいいこと、SGE のインストールもマスターホスト、実行ホストともに 1 度目以降は行わなくてもいいことが挙げられる。

逆に VMware であるがゆえに起こりうるオーバーヘッドとしてコピーマシンの作成に要する時間、およびコピーマシンの初回起動時の IP アドレスはコピー元となっているマシンと同一のものであるため IP アドレスの変更処理を行うための時間が挙げられる。

これらオーバーヘッドの存在と 1 度目は SGE のインストール時間が必要であるためマスターホスト 1 台、実行ホスト 1 台の環境を用意する段階では仮想 Grid コンピューティング環境を構築する場合と、実機で Grid コンピューティング環境を構築する場合での時間的コストの軽減はあまり見られなかったが、実行ホスト 2 台目以降は SGE インストールの必要がないので大幅な時間的コストの軽減が可能である。

ただし、オーバーヘッドの部分で挙げたコピーマシンの作成時間であるが本実験環境においてコピー元となる仮想マシンの HDD 容量は 10GB としていたため 5 分ほどの時間で済んでいた。しかし、HDD 容量を増やすことによってコピーマシンの作成時間は増大する。Fedora Core4 のインストール時間および SGE インストール時間の合計時間より、コピーマシン作成に要する時間が上回る場合、オーバーヘッドの方が省略可能な処理時間を上回ってしまうため、かって仮想 Grid コンピューティング環境を構築する方が時間的コストを増大させてしまうことも判明した。

そのため、時間的コストの軽減を行うためには 1 台あたりの割り当て HDD 容量をある程度制限する必要がある。この制限は時として Grid コンピューティングの研究・開発の場面において弊害となる可能性がある。

また、コピーマシンの作成時は一切の対話型処理を必要としていないためコピーマシンを作成している間に別の作業を行うことができる実機において

SGE インストール処理を行っている間は対話型処理が随所に存在するためインストールしている間、別の作業を行うことができない。インストール時間短縮のみならず、空いた時間の有効活用という面から見ても時間的コストの軽減を見て取ることができた。

5.3 仮想 Grid コンピューティングの他の環境への移植に関する考察

VMware による仮想 Grid コンピューティング環境はその環境を構築する仮想マシンを VMware をインストール済みの他のコンピュータにコピーすることによって移植することが可能である。出来上がってしまっている Grid コンピューティング環境を譲渡するといったことができるため、こちらも Grid コンピューティング環境を構築する上での時間的コストの軽減につながるのではないかと考えられる。

5.3.1 仮想 Grid コンピューティング環境を移植する際の問題点

仮想 Grid コンピューティング環境を移植する際には、移植先の LAN 環境に合わせて IP アドレス、場合によってはホスト名の変更が必要となる。SGE においてホスト名・IP アドレスを変更する際の注意点は以下の通りである。

- マスターホストにおいて IP アドレスの変更は可能であるが、ホスト名を変更した場合は SGE は正常に機能しなくなる。
- 実行ホストはホスト名・IP アドレスの変更は自由である。

ただし、いずれの場合もマスターホスト・実行ホストにおいて変更されたホスト名・IP アドレスに対する名前解決のための設定変更を行う必要がある。

名前解決の設定を変更するためには以下のような作業が必要である。

マスターホストでの設定変更

- 移植する LAN 環境に合わせて IP アドレスの設定、または、取得
- /etc/hosts、/etc/hosts.allow、/etc/exports ファイルの再編集
- ファイルの再編集に伴う設定の変更を有効にする。

実行ホストでの設定変更

- 移植する LAN 環境に合わせてホスト名・IP アドレスの設定または、取得
- /etc/hosts、/etc/fstab ファイルの再編集
- ファイルの再編集に伴う設定の変更を有効にする。

各ファイルの役割および編集内容は 4.2 節で詳しく説明しているため、ここでの説明は省略する。環境の移植とはまた別問題であるが仮想 Grid コンピューティング環境を拡張する際にもこれらの問題は発生する。

5.3.2 仮想 Grid コンピューティング環境を移植する際の問題解決

5.3.1 において仮想 Grid コンピューティング環境を移植するにあたって問題となるホスト名および IP 変更による各種設定の変更および自動インストールを少ない作業で実現させるために以下に示す 2 つのプログラムを作成した。

ファイル自動編集プログラム

このプログラムは仮想 Grid コンピューティング環境を移植する際に行う。ファイルの再編集作業に必要なデータがマスターホスト・実行ホストのホスト名・IP アドレスだけであることに着目し、それらのデータを最初に対話型処理によって入力することによって後の、ファイルの再編集作業を自動的に行ってくれることを目的としている。

プログラム仕様

作成したプログラムには 4 つのモードを用意しており、選択したモードごとに少しずつではあるがその動作が異なる。用意したモードは以下のようなものである。

- マスターホストにおける実行ホスト追加時に用いるモード（以下モード 0 と記述）
- 実行ホストにおける実行ホスト追加時に用いるモード（以下モード 1 と記述）
- マスターホストにおけるホスト名および IP アドレス変更時に使用するモード（以下モード 2 と記述）

- 実行ホストにおけるホスト名および IP アドレス変更時に使用するモード（以下モード 3 と記述）

以下に各モードにおけるプログラムの動作を示す。

モード 0

1. 標準入力より追加するホストのホスト名・IP アドレス・エアリアス名を文字列として読み込む。最大 100 台分まで読み込み可能、100 台未満で入力を中断する場合は IP アドレス入力時に Ctr+D を入力する。
2. 入力されたデータを元に/etc/hosts、/etc/hosts.allow、/etc/exports 編集時に入力される文字列を作成する。
3. 作成した文字列を 3 つのファイルに書き込む。
4. SGE に対して qconf コマンドを用いて設定を追加したホストを実行ホストとして登録

モード 1

1. 標準入力より追加するホストのホスト名・IP アドレス・エアリアス名を文字列として読み込む。最大 100 台分まで読み込み可能、100 台未満で入力を中断する場合は IP アドレス入力時に Ctr+D を入力する。
2. 入力されたデータを下に/etc/hosts 編集時に入力される文字列を作成する。
3. 作成した文字列を/etc/hosts ファイルに書き込む。

モード 2

1. 標準入力より追加するホストのホスト名・IP アドレス・サブネットマスク・デフォルトゲートウェイ・エアリアス名を文字列として読み込む。最大 100 台分まで読み込み可能、100 台未満で入力を中断する場合は IP アドレス入力時に Ctr+D を入力する。
2. 入力されたデータを下に IP アドレス・サブネットマスク・デフォルトゲートウェイ変更用の文字列および/etc/hosts、/etc/hosts.allow/etc/exports 編集用の文字列を編集する。
3. IP アドレス・サブネットマスク・デフォルトゲートウェイ変更および作成した文字列を 3 つのファイルに書き込む。

4. SGE 自動インストール[3]時にインストール内容を規定するためのファイル（雛形はあらかじめ作成しておく必要あり）に入力情報をもとに必要な情報を追加する。

5. 自動インストール用のバッチ処理用のファイルを sge.sh を生成する。

モード 3

1. 標準入力より追加するホストのホスト名・IP アドレス・サブネットマスク・デフォルトゲートウェイ・エアリアス名を文字列として読み込む。最大 100 台分まで読み込み可能、100 台未満で入力を中断する場合は IP アドレス入力時に Ctr+D を入力する。

2. 入力されたデータを下に IP アドレス・サブネットマスク・デフォルトゲートウェイ変更用の文字列および/etc/hosts、/etc/fstab 編集用の文字列を編集する。

3. IP アドレス・サブネットマスク・デフォルトゲートウェイ変更および作成した文字列を 2 つのファイルに書き込む。

プログラムの使用方法

ソースを展開する。

```
tar xvf sge_program_comp.tar
```

作成されたディレクトリに移動

```
cd sge_program
```

プログラムを root 権限で実行

```
su root
./configur_sge
```

プログラム終了後、仮想マシンを再起動させる。

モード 2 使用時は、再起動後 sge_program ディレクトリに作成されている sge.sh を実行し SGE の自動再インストールを行う。

```
su root
cd sge_program
./sge.sh
```

[3]本プログラムによって自動インストールされる SGE の環境設定は 4.2 節で筆者が例として挙げたインストール環境と同様のものである。

本プログラムのメリット

- 少ないデータの入力で SGE の設定変更に必要な処理を自動で行ってくれる。
- モード 2 は SGE 初回インストール時にも使用可能であり、構築できる環境は一意的ではあるものの構築する Grid コンピューティング環境に組み込まれるホストのホスト名・IP アドレス関連のデータを打ち込むだけでマスターホスト側の SGE インストールができる。

本プログラムの課題

- SGE の自動インストールの際の環境設定を任意で変更できるような機能を追加すること、ただし、その場合は入力する必要があるデータが増えることになるので、新規のモードとして作成する必要があるものと考えられる。
- 各種データの入力の際にユーザーによって不適切な入力がなされた際のエラー処理を一切行わない点を改善する必要がある。
- モードによって入力データに違いがあるため、操作が少々複雑である。
- あくまで、自動化できたのは設定変更に伴うファイルの再編集部分だけなので、ホスト名・IP アドレスの取得、変更した設定を有効にするといった作業も自動化する。

図 5.2 に、ファイル自動編集プログラムの実行画面を記載する。

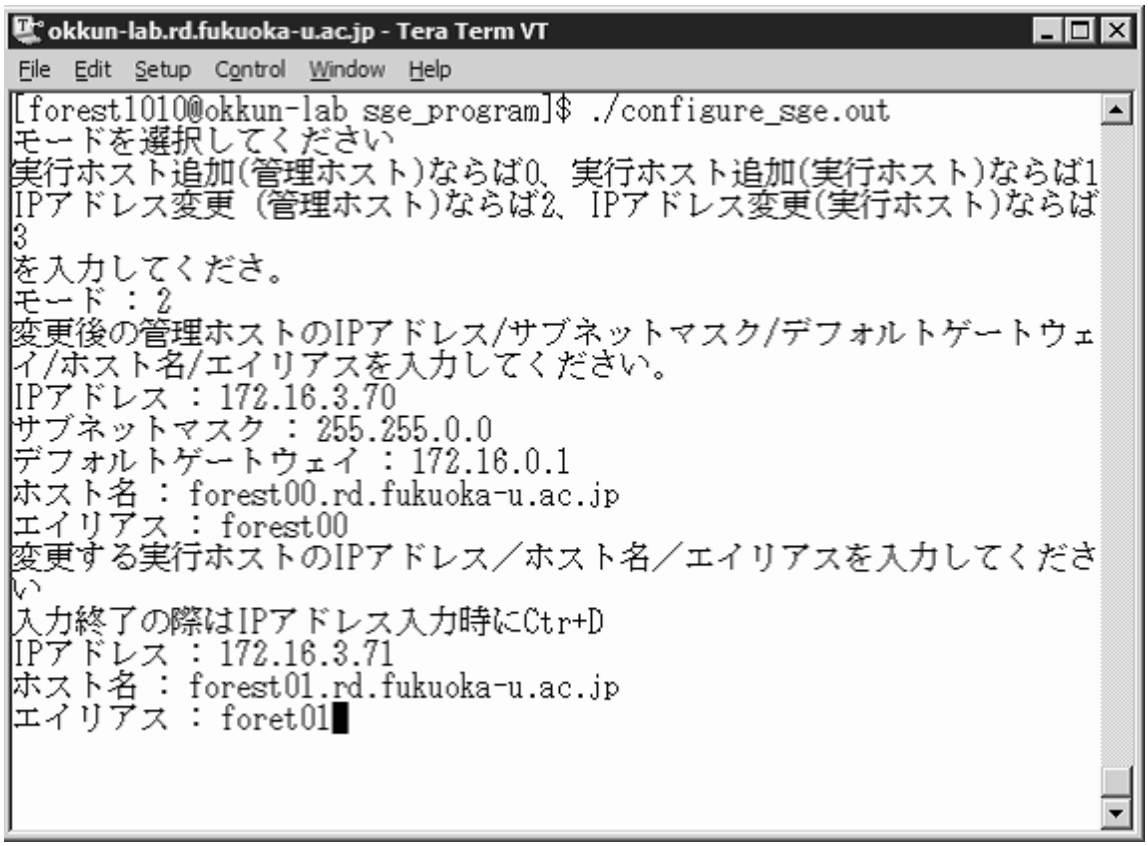

図 5.2: ファイル自動編集プログラム実行画面

実行ホスト自動追加プログラム

このプログラムは先に述べたプログラムの課題を踏まえた上で、再度、作成したプログラムである。

本プログラムでは複雑な操作を必要とせず、また、プログラム終了後に即時利用可能な Grid コンピューティング環境を構築することを目的としている。

プログラム仕様

本プログラムはネットワーク通信を利用したプログラムであり、マスターホスト用のプログラム・実行ホスト用プログラムの 2 種類によって構成される。それぞれのプログラムが行う処理は次のようになっている。(図 5.3、図 5.4 も参照)

マスターホスト用プログラム

1. マスターホストに割り振られている IP アドレスを取得する。
2. マスターホストのホスト名を取得。
3. UDP サーバーを立てる。
4. 実行ホストに割り振られている IP アドレスを取得する。(UDP ブロードキャスト受信)
5. マスターホストのホスト名を基に実行ホストに割り振るホスト名を決定する。
6. 取得した IP アドレスを実行ホストに伝える。(TCP 送信)
7. 実行ホストに割り振るホスト名を実行ホストに伝える。(TCP 送信)
8. 取得した各種データを基にファイル編集に必要なワードを生成する。
9. 生成したワードを用いて/etc/hosts、/etc/hosts.allow、/etc/exports を編集する。
10. SGE に変更 (追加) するホストを登録する。
11. ファイル編集に伴う設定の変更を有効にする。
12. マスターホストにおける設定の変更が全て終了したことを実行ホストに伝える。(UDP 送信)

実行ホスト用プログラム

1. 実行ホストに割り振られている IP アドレスを取得する。
2. 取得した IP アドレスをマスターホストに伝える。(UDP ブロードキャスト送信)
3. TCP サーバーを立てる。
4. マスターホストの IP アドレスを取得。(TCP 受信)
5. マスターホストより割り振られたホスト名を取得する。(TCP 受信)
6. マスターホストより割り振られたホスト名に変更する。
7. 取得した各種データを基に/etc/hosts、/etc/fstab、/etc/sysconfig/network[4]を編集する。

[4]ホスト名の変更に伴い再編集の必要が生じるファイル

8. UDP サーバーを立てる。

9. マスターホストにおける各種設定の終了を確認する。(UDP 受信)

10. ホスト名の変更に伴う、NIC の再起動

11. マスターホスト内の SGE_ROOT ディレクトリを NFS よって共有ディレクトとして扱えるようにする。

12. (実行ホスト側の SGE を自動的にインストールする。)[5]

以上の作業をそれぞれのプログラムが自動的に処理を行い、プログラム終了時に即時利用可能な仮想 Grid コンピューティング環境を構築または再構築してくれる。

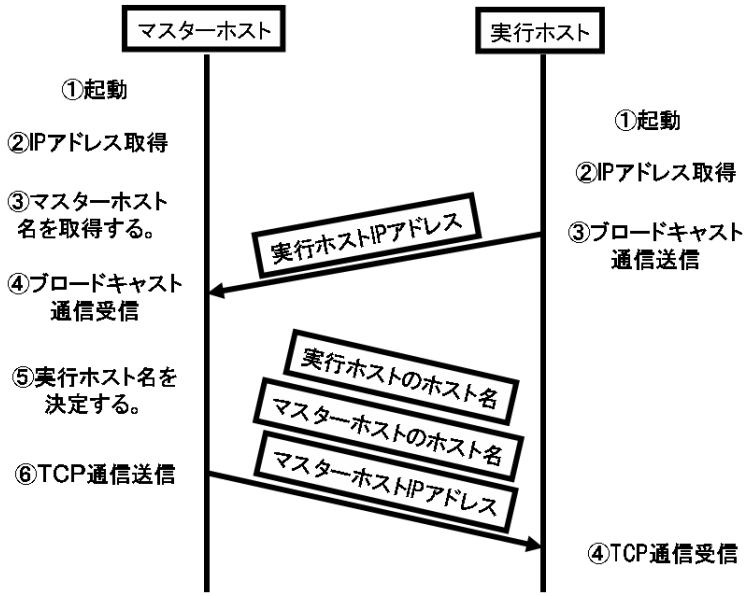

図 5.3: 実行ホスト自動追加プログラムのプロセス その 1

[5] 仮想 Grid コンピューティング環境を移植する際には必要ないが、仮想 Grid コンピューティング環境を拡張する際には必要な処理

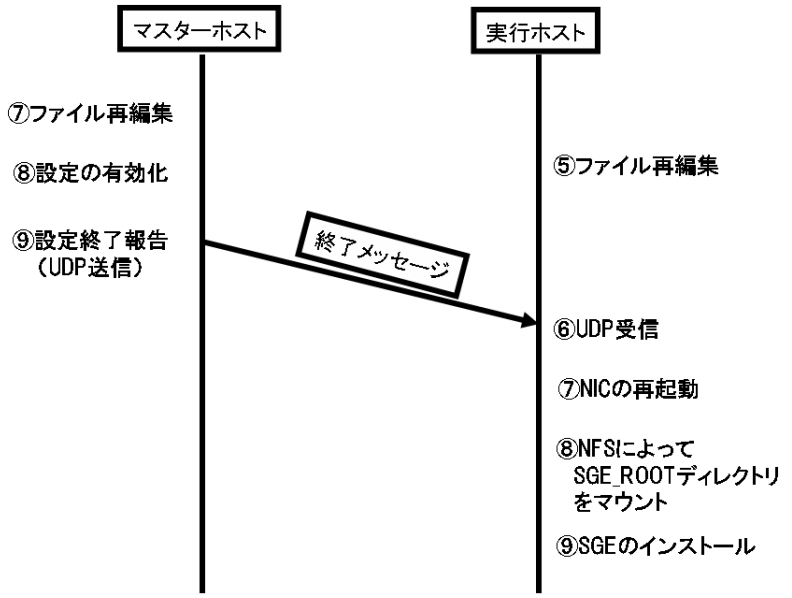

図 5.4: 実行ホスト自動追加プログラムのプロセス その 2

プログラムを使用する際の前提条件

本プログラムを使用する際には以下に示す前提条件を満たしている必要がある。

- マスターホスト・実行ホストにおいて手動ないし、DHCP によって既に IP アドレスが割り振られている。
- マスターホストにおいて SGE(管理ホスト) のインストールが終了している。
- マスターホストにおいて「hostname -f 」「hostname -a」「hostname -d」「exportfs -a」コマンドが実行可能であり、かつ、正常な値が返ってくる。
- マスターホストにおいて NFS が動作している。
- プログラムの実行は root 権限で行う必要がある。

プログラム使用方法

ソースファイルを展開する。

```
tar xvf my_sge_tools_.070120_02.tar
```

展開されたディレクトリに移動し、各種ファイルのコンパイルを行う。

```
cd my_sge_tools
make
```

setSGE_ROOT.out を実行し対話型処理によって SGE_ROOT ディレクトリと SGE 各種デーモンが利用する Port 番号を入力する。

```
./setSGE_ROOT.out
```

setSGE_ROOT.out による設定変更を有効にするため再度 make コマンドを実行

```
make
```

マスターホスト→実行ホストの順でそれぞれのホスト用のプログラムを起動

```
マスターホスト側
./sge_configure_qmaster.out
実行ホスト側
./sge_configure_execd.out
```

自動処理終了後、マスターホスト側のプログラムは内部で無限ループ処理を行っているため外部から終了信号を送信し、プログラムを停止させる。

```
./end.out
```

本プログラムのメリット

本プログラムを用いることで、仮想 Grid コンピューティング環境の移植に伴う設定の変更、または、仮想 Grid コンピューティング環境の拡張に伴う処理を全自動で行うことが可能である。

これにより、Grid コンピューティングの知識・技術に乏しい人であっても、既に出来上がっている仮想 Grid コンピューティング環境を移植することが可能となり、また、移植した環境を拡張する際にも本プログラムを利用するおとで簡単に環境の拡張を行うことが出来る。

また、実行ホストのインストールにかかっていた 5〜10 分程度の時間を数秒に短縮することにも成功した。

本プログラムの課題

- 通信失敗時のエラー処理が不十分であるため、状況次第ではプログラムを強制終了させなくてはならない。
- 同一ホストからの 2 度目以降のアクセスを拒絶することが出来ないため同一ホストの情報を 2 重に登録する可能性がある。

以上の理由により、本プログラムはユーザーの使用方法に依存する部分が多い。

第6章　VMwareの設定変更に伴う仮想マシンの性能への影響検証

この章では、VMware の仮想マシンに対するリソース設定機能、仮想ネットワーク設定機能の影響をベンチマークテストを行うことにより検証し、VMware が Grid コンピューティングの研究・開発における資源的コストの軽減にどの程度有用であるかを考察したいと思う。

6.1　実験環境

本実験では実機と仮想マシンとで実験を行う、ここで言う実機とは以下に示すスペックのコンピュータ (表 6.1) のことであり、仮想マシンとはその実機内に VMware Workstation ver.5.5.1 によって構築する仮想マシンのことである。また、仮想マシンにインストールしているゲスト OS は Fedora Core4 である。

実機のスペック

製品名	IBM 816829I
CPU	Intel(R) Pentium(R)4 3.40GHz
メモリー	SD-RAM512MB(512MB × 1) or SD-RAM2G(1G × 2)[1]
グラフィックボード	Intel(r)82945G Chipset Family Graphics chip Accelerate
OS	WindowsXP Professional Ver.2002 Service Pack2
HDD 1	ST3300831AS
HDD 2	ST3300831AS
HDD 3	WDCWD1600JS-08MHBO

表 6.1: 実験環境

[1]実機のベンチマークテスト時には 512MB、仮想マシンのベンチマーク時は 2G にしている。

6.2 リソース設定変更に伴う影響

仮想マシンのリソース設定を変更しながらベンチマークテストを行い、設定変更に伴う仮想マシンの性能の影響を検証する。変更する設定は以下の通りである。

- スナップショット機能:ON/OFF
- VMwareTools:インストールあり/なし
- CPU 割当て優先度:高/中
- メモリ容量:512MB/1GB
- HDD 事前割り当て:あり/なし

6.2.1 実験目的

スナップショット機能:ON/OFF

スナップショット機能を ON にした状態では VMware が仮想マシンの状態変化を常に監視しているため CPU に多少の負担を掛けている。スナップショット機能を OFF にすることで CPU への負担を軽減し、CPU 処理速度の向上ができると考えられる。その処理速度の向上がどの程度のものであるかを検証する。

VMwareTools:インストールあり/なし

VMwareTools をインストールすることでゲスト OS 上のグラフィック性能が向上するとのことなので、どの程度の性能向上が見られるのか検証を行う。

CPU 割り当て優先度:高/低

VMware 上で動作している仮想マシン上の処理に対する CPU 割り当ての優先度を高めることで、仮想マシンにおける処理速度の向上を図り、どの程度の処理速度の向上が望めるのかを検証する。

メモリ容量割り当て:あり/なし

メモリ容量の増加に伴い、仮想マシンの性能が向上するものと考えられる。設定変更にともなう性能の向上具合を検証する。

HDD 事前割り当て:あり/なし

仮想マシン作成時、事前に仮想マシンの HDD 容量を確保した上で 1 箇所にまとめて全データを保存する場合と、事前に HDD 容量を確保せずデータを記憶するたびに HDD 上の空きスペースに分散して保存する場合とでは、前者の方が HDD アクセス速度を向上するものと考えられる。前者と後者の HDD アクセス速度の差を検証する。

6.2.2 使用したベンチマークソフト

HDBENCH Clone

Linux 上で CPU 演算速度・メモリアクセス・グラフィック性能・ディスクアクセスの性能を検証するためのベンチマークソフト。Linux の X Window を使用してベンチマークを実行する。

HDBENCH Clone 使用方法

ソース（hdbench-1.14.0.tar.gz）をダウンロードしてきた後、ファイルを展開。また、以下に示す操作は root 権限で実行する。

```
tar zxvf hdbench-1.14.0.tar.gz
```

解凍後、インストールに必要なヘッダファイル当の編集を自動的に行う configure を実行する。

```
cd hdbench-1.14.0/
./configure
```

configure 実行後、HDBENC Clone 本体のコンパイルおよび Linux への環境設定を行う。

```
make
make install
```

Fedora Core 上で実行した場合、文字コードの違いから文字化けしてしまうので以下のようなコマンドを入力し実行する。

```
LANG=ja_JP.eucJP hdbench &
```

各項目の単位

- 浮動小数点演算

  1 秒間あたりの浮動小数点演算の計算回数

- 整数演算

  1 秒間あたりの整数演算の計算回数

- メモリ

  データの転送元と先にそれぞれ 2MB づつのメモリを確保し、ダブルワードストリング転送で 32KB のデータを 64 回転送を繰り返す。32KB を転送するごとに計測値は 1 を数え、計測は 3 秒間実施する。

- 短形・円・テキスト

  短形・円は画面上のランダムの位置に指定の図形を描画し、1 秒間の描画回数を計測する。テキストは ”HDBENCH Clone ”と表示し、1 秒間の描画回数を計測する。

- スクロール

  画面全体を 1 ピクセル下にずらしコピーする。Y 座標が 0 ラインに線を引くという処理を、繰り返しスクロールする。50 ピクセルをスクロールするごとに、計測値は 1 を数える。約 3 秒間の測定の後、描画回数を表示。

- IMAGE

  Ximage を使って 1000 枚のスプライトを縦横無尽に動かす。該当の測定値は、1 秒あたりの書き換えたフレーム数である。

- READ

  右側の ”DRIVE ”欄に指定されたパスにテンポラリファイルを作成する。”使用量 ”欄で指定されたサイズまで 512KB ずつ読み込みむ。該当の測定は、1 秒間あたりの転送バイト数である。

- WRITE

  右側の ”DRIVE ”欄に指定されたパスにテンポラリファイルを作成する。”使用量 ”欄で指定されたサイズまで 512KB ずつ書き込む。該当の測定は、1 秒間あたりの転送バイト数である。

NBench

NBench は、一般的な 10 のアルゴリズムを実行し、その処理時間によってシステムの CPU やメモリの処理能力を測定するベンチマークプログラムである。今回は、10 あるアルゴリズムのうち整数演算・浮動小数点演算にかかわる 2 つのアルゴリズムによる結果に着目する。

アルゴリズム	概要
NUMERIC SORT	整数のソート、整数をソートし、整数演算の性能を測定する。
FOURIER	フーリエ級数展開、超越計算や三角関数により浮動小数点演算の性能をテストする。

表 6.2: NBench の計測アルゴリズム

NBench 使用方法

ソース（nbench-byte-2.2.2.tar.gz）をダウンロードし、展開する。

```
tar zxvf nbench-byte-2.2.2.tat.gz
```

NBench 本体のコンパイルを行う。

```
cd nbench-byte-2.2.2/
make
```

実行する際には以下のように入力。

```
./nbench
```

Bonnie

HDD アクセス速度を計測することができるベンチマークソフト。ユーザーが指定サイズのファイルを指定したフォルダに作成し、読み込み速度、書き込み速度を 1 秒間の転送速度として計測する。

Bonnie の使用方法

ソース（Bonnie.tar.gz）をダウンロードし、展開する。

```
tar zxvf Bonnie.tar.gz
```

Bonnie 本体のコンパイル

```
make
```

HDBENCH Clone のディスクアクセス計測方法に合わせるため作成するファイルサイズを 10MB、作成先のディレクトリを/tmp として、実行するには以下のように入力する。

```
e ./Bonne -s 10 -d /tmp
```

オプション ”-s ”の後ろに任意のファイルサイズ（単位は MB）を指定、およびオプション ”-d ”の後ろにファイルを作成するディレクトリを指定する。

6.2.3 実験方法

設定変更を行いつつ、3 つのベンチマークソフトを用いてベンチマークテストを行う。テストは 5 回行うものとし、5 回分の平均値を元に各種設定が仮想マシンに与える影響を検証していくものとする。

6.2.4 実験結果

以下に実機および各設定環境下における仮想マシンのベンチマークテストの結果を示す。なお、各ベンチマークテスト結果の値の単位は 6.2.2 節にて説明した通りである。

実機における各種ベンチマークの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
ALL	282210	285770	281881	321403	317598	297772
CPU　浮動小数点	111604	111615	111670	108892	111480	111052
CPU　整数演算	353174	383431	373447	348884	353167	362421
メモリ	424722	394961	407893	428167	405722	412293
VIDEO　短形	31940	55914	27936	55530	34804	41225
VIDEO　円	43509	43257	43025	44867	46267	44185
VIDEO　テキスト	61469	61251	51562	66033	64799	61023
VIDEO　スクロール	176	176	176	176	176	176
DISK　READ	1747626	1747626	1747626	2097152	2097152	1887436
DISK　WRITE	47880	59578	55480	64329	62415	57936

表 6.3: HDBENCH Clone によりベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
NUMERIC SORT	809.56	804.96	811.04	811.68	810.56	809.56
FOURIER	18340	18340	18388	18388	18404	18372

表 6.4: NBench によるベンチマー (図 5.2 クテスト結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
READ	1739130	1746546	1753124	1733536	1724486	1739364
WRITE	45159	44975	43850	45124	45131	44848

表 6.5: bonnie によるベンチマークテスト結果

仮想マシン スナップショット:OFF VMwareTools:なし 優先度:中 メモリ:512MB HDD 事前割り当て:あり

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
ALL	225734	246786	227833	208353	172373	216216
CPU 浮動小数点	101642	101294	96933	101135	101428	100486
CPU 整数演算	387905	390174	373537	388688	388453	385751
メモリ	344889	318889	340484	304560	353766	332518
VIDEO 短形	26324	34899	36979	24819	25387	29682
VIDEO 円	26331	38918	38384	24368	25942	30789
VIDEO テキスト	30357	45951	43393	37914	26744	36872
VIDEO スクロール	54	54	54	47	54	53
DISK READ	1310720	1497965	1310720	1165084	748982	1206694
DISK WRITE	29127	39718	37854	36921	42974	37319

表 6.6: HDBENCH Clone によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
NUMERIC SORT	735.20	702.56	729.60	722.08	724.64	722.82
FOURIER	16302	17210	17171	17249	16582	16903

表 6.7: NBench によるベンチマークテスト結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
READ	1356471	1322314	1325223	859565	1062240	1185163
WRITE	43211	43670	45039	47690	45842	45090

表 6.8: bonnie によるベンチマークテスト結果

仮想マシン スナップショット:ON VMwareTools:なし 優先度:中 メモリ:512MB HDD 事前割り当て:あり

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
ALL	206814	197827	227559	222380	242406	219397
CPU 浮動小数点	101228	96047	97385	100894	100907	99292
CPU 整数演算	383644	355747	374934	387896	385142	377473
メモリ	307507	325763	317767	309234	312825	314619
VIDEO 短形	26111	36885	38330	24252	25481	30212
VIDEO 円	25524	38965	37589	21503	26154	29947
VIDEO テキスト	32322	46280	43916	30423	30351	36658
VIDEO スクロール	47	55	54	47	47	50
DISK READ	1165084	1048576	1310720	1310720	1497965	1266613
DISK WRITE	26681	30045	54899	38836	45197	39132

表 6.9: HDBENCH Clone によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
NUMERIC SORT	747.04	751.20	753.76	737.14	744.36	746.70
FOURIER	17138	17130	16909	16924	16948	17010

表 6.10: NBench によりベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
READ	1434173	733314	1072700	1003921	1118392	1072500
WRITE	46831	45578	46158	45154	45981	45940

表 6.11: bonnie によるベンチマークテストの結果

仮想マシン スナップショット:OFF VMwareTools:あり 優先度:中 メモリ:512MB HDD 事前割り当て:あり

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
ALL	224804	210884	242404	222165	207535	221558
CPU 浮動小数点	100960	97691	101157	101661	100036	100301
CPU 整数演算	388904	371853	386439	384105	376404	381541
メモリ	308537	308467	306978	301759	309450	307038
VIDEO 短形	23968	36317	24914	25765	29670	28127
VIDEO 円	24391	37398	26007	26992	29729	28903
VIDEO テキスト	36456	46585	31070	35526	31635	36254
VIDEO スクロール	54	54	54	54	54	54
DISK READ	1310720	1165084	1497965	1310720	1165084	1289915
DISK WRITE	54050	45392	49461	35069	33288	43452

表 6.12: HDBENCH Clone によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
NUMERIC SORT	743.52	759.36	755.04	727.20	739.36	744.90
FOURIER	17188	17155	17180	17242	17178	17189

表 6.13: NBench によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
READ	1083597	1003036	1075856	1044578	1086933	1058800
WRITE	47851	46520	46731	47461	46983	47109

表 6.14: bonnie によるベンチマークテストの結果

仮想マシン スナップショット:OFF VMwareTools:なし 優先度:高 メモリ:512MB HDD 事前割り当て:あり

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
ALL	227432	199107	166252	151443	150928	179032
CPU　浮動小数点	104301	104060	102274	103104	104243	103596
CPU　整数演算	392182	399470	390714	395126	392527	394004
メモリ	339326	207640	326671	323059	258744	291088
VIDEO　短形	37056	31183	37074	36715	37074	35820
VIDEO　円	38087	31798	37559	36895	37355	36339
VIDEO　テキスト	45477	34844	45302	45477	45648	43350
VIDEO　スクロール	47	40	54	54	54	50
DISK　READ	1310720	1165084	699050	551332	616809	868599
DISK　WRITE	6628	19859	14423	22121	16831	15972

表 6.15: HDBENCH Clone によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
NUMERIC SORT	753.44	747.68	746.24	752.16	754.56	750.82
FOURIER	17092	17066	17155	17154	16668	17027

表 6.16: NBench によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
READ	1031634	1020631	1023112	594105	751008	884098
WRITE	45417	46811	46272	45184	46449	46027

表 6.17: bonnie によるベンチマークテストの結果

仮想マシン スナップショット:OFF VMwareTools:なし 優先度:中 メモリ:1GB HDD 事前割り当て:あり

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
ALL	206406	159942	162785	221057	151619	180362
CPU　浮動小数点	100093	100369	100525	99751	99889	100125
CPU　整数演算	373454	303738	384848	386561	386176	366955
メモリ	315027	311540	251990	312631	313397	300917
VIDEO　短形	24340	33953	35537	24223	19590	27529
VIDEO　円	24121	36282	36099	23116	24179	28759
VIDEO　テキスト	37354	30239	42870	28628	31078	34034
VIDEO　スクロール	40	54	49	40	53	47
DISK　READ	1165084	748982	748982	1310720	616809	918115
DISK　WRITE	24556	25266	26955	24906	25025	25342

表 6.18: HDBENCH Clone によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
NUMERIC SORT	749.00	739.04	741.76	743.84	719.20	738.57
FOURIER	16773	16908	16949	16581	16393	16721

表 6.19: NBench によるベンチマークテスト結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
READ	563938	1002545	993788	1009065	999609	913789
WRITE	46251	42493	46130	46103	46033	45402

表 6.20: bonnie によるベンチマークテスト結果

仮想マシン スナップショット:OFF VMwareTools:なし 優先度:中 メモリ:512MB HDD 事前割り当て:なし

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
ALL	139428	206722	161117	157386	160547	165040
CPU　浮動小数点	100687	100861	100974	100417	100504	100689
CPU　整数演算	304553	382745	314295	320398	313297	327058
メモリ	308987	306959	302696	307172	305077	306178
VIDEO　短形	31467	24346	35655	34899	35561	32386
VIDEO　円	32636	23605	34070	34316	33960	31717
VIDEO　テキスト	41271	34930	42082	44269	42147	40940
VIDEO　スクロール	54	41	54	54	54	51
DISK　READ	551882	1165034	748982	699050	748982	782786
DISK　WRITE	22745	28649	32363	33288	25890	28587

表 6.21: HDBENCH Clone によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
NUMERIC SORT	737.60	742.56	735.84	733.28	737.28	737.31
FOURIER	17018	16480	16957	16980	17075	16902

表 6.22: NBench によるベンチマークテストの結果

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値
READ	1301970	746900	997856	863260	1521771	1086351
WRITE	46228	46142	46551	41416	42895	44646

表 6.23: bonnie によるベンチマークテスト結果

6.2.5　考察

この実験により判明したことは以下の通りである。

実機におけるベンチマークテストの結果より、HDbench Clone のディスクアクセスの測定結果にある程度のむらが見られることより、この機能に関しては少々信頼性に欠けることが確認できた。

しかし、bonnie における測定結果が信頼性が高いと見なせるものであるので、ディスクアクセスに関しては bonnie の測定結果を用いるものとする。(表 6.3・6.5 比較による)

スナップショットを OFF にすることによる CPU 処理速度の向上は確認できたが、その効果は微々たるものであり、ほぼ性能的に変化はないと考えてかまわない程度である。(表 6.6・6.9 比較および表 6.7・6.10 比較による)

VMwareTools によるグラフィック性能の向上は確認できなかった。しかし、VMwareTools をインストールすることによる利便性の向上は仮想マシンを用いる上では大きな助けとなるこは確認できた。(表 6.6・6.12 比較による)

CPU 割り当ての優先度を高くした場合には、HDBENCH Clone では整数演算・浮動小数点演算ともに若干の性能向上が確認できたが、NBench においては整数演算は性能が向上するものの浮動少数点演算は低下している。したがって、性能の向上は見られるものの、その向上幅はベンチマークソフトによる誤差の範囲内にとどまる程度のものであり、仮想マシン全体から見れば性能的に変化はないと考えてかまわない。

また、CPU 割り当て優先度を高くした場合には CPU の演算速度以外の計測結果が全体的に不安定になっている様に見受けられる。仮想マシン内における演算能力に CPU が使用されるため、ホスト OS における VMware の仮想マシン全体の制御に回る CPU 使用時間が減少したためと考えられる。(表 6.6・6.15 比較および表 6.7・6.16 比較による)

メモリ割り当て容量を増加させることによる性能の変化はほとんど検出することができなった。実験に用いたベンチマークソフトには大容量のメモリを必要とする処理が含まれていなかったためである。メモリ容量に関する実験には今後、見直しの必要がある。(表 6.6・6.18 比較および表 6.7・6.19 比較による)

HDD 事前割り当ての有無によるディスクアクセスの変化は、HDBENCH、Bonnie ともに予想して結果と異なる結果となった、双方の性能がほぼ同一のものであると判断できるような結果になったのだが、事前割りあてを行った場合のほうが体感速度的にも明らかに読み出し・書き込みともに処理速度が速く、仮想マシン全体の動作も滑らかであることから、同程度のディスクアクセスであるとは考えられない。

仮想マシンにおけるディスクアクセスの値は全体を通してみても明らかに信頼性に欠ける計測結果が多々見られる。しかしながら、実機において行ったベンチマークテストにおいては一定した信頼性の高い結果が得られていることからディスクアクセスに関していえば VMware 側にベンチマークソフトに悪影響を与える原因があるものと思われる。(表 6.8・6.23 比較より)

本実験内容では、推測の域は出ないがおそらくホスト OS とゲスト OS による HDD 管理の違いによる差を VMware が吸収する際に乗じるオーバーヘッドが原因ではないかと考えられる。Linux 版 VMware を用いて同様の実験を

行うことで事実確認が行えるものと考えるが本実験環境では確認できなかった。こちらも今後の課題となる。

以上のことからリソース設定機能による仮想マシンの性能への影響は微々たるものであり、リソース設定によって性能を変化させた仮想マシンを複数台用意してもそれは異なる性能を備えたコンピュータ資源と呼ぶことは出来ない。したがって、リソース設定機能は Grid コンピューティングの研究・開発における資源的コストの軽減に有用であるとは言えない。

6.3 ネットワーク環境の設定変更に伴う影響

仮想マシン同士を結ぶ仮想 LAN ケーブルはその通信速度を任意に変更可能である。また、ネットワークの接続方式もブリッジ方式、NAT 方式、ホストオンリー方式の 3 つの方式がある。それらの設定を変更することで仮想マシン間のデータ通信速度にどれだけの影響が出るのかを検証する。

6.3.1 実験環境

本実験は以下に示す PC 環境 (表 6.24) 上に構築した 2 つの仮想マシン (メモリ容量 512MB) を用いて行ったものである。

製品名	IBM 816829I
CPU	Intel(R) Pentium(R)4 3.40GHz
メモリー	SD-RAM2G(1G × 2) or SD-RAM(512 × 1)[2]
グラフィックボード	Intel(r)82945G Chipset Family Graphics chip Accelerate
OS	WindowsXP Professional Ver.2002 Service Pack2
HDD 1	ST3300831AS
HDD 2	ST3300831AS
HDD 3	WDCWD1600JS-08MHBO

表 6.24: 実験環境

また、nttcp ベンチマークテストにおいて実機による実験を行っているが、この際に使用しているコンピュータは上記と同一スペックのコンピュータ 2 台を使用している。なお、この際の LAN の速度は 100Mbps である。

6.3.2 使用したベンチマークソフト

nttcp

ネットワーク性能を計測するベンチマークソフト。tcp、udp、udp マルチキャストでの通信レートを計測することが可能である。

計測方法

リモートホスト上で起動している nttcp デーモンに対して、ローカルホストからデフォルトでは 4KB のバッファを 2024 個、つまり合計 8MB のデータを転送し、転送に要した時間から 1 秒間のデータ転送量を計測する。

nttcp の使用方法

ソース（nttcp_1.47.orig.tar.gz）をダウンロードし、展開する。また、以下に示す処理は root 権限で行うものとする。

```
tar zxvf nttcp_1.47.orig.tar.gz
```

nttcp 本体をコンパイルし、Linux への環境設定を行う。

```
cd nttcp-1.47/
make
make install
```

実行方法は、まずデータを転送するリモートホスト上で nttcp デーモンを起動し、データ受信の準備を行う。

```
ssh リモートホスト名
Password 入力
cd nttcp-1.47/
./nttcp -i &
exit
```

オプション ”-i ”はデータを受信する側のホストで nttcp をデーモンとして起動する際につける。

ローカルホスト上からリモートホストに対してデータを送信し、1 秒間のデータ転送量を計測する。

```
./nttcp リモートホスト名
```

また、デフォルトでは転送するデータの総量は 8MB であるが以下のようにオプション設定を行うことで 100MB のデータを転送可能である。

```
./nttcp -s 1024 -n 102400 リモートホスト名
```

オプション ”-s ”の後ろにはバッファのサイズをオプション ”-n ”の後ろには転送するバッファの数を指定する。

6.3.3 実験内容

VMware では 3.3.6 節で説明したように、ホストマシン・仮想マシン間および仮想マシン・仮想マシン間の通信に関して以下のような 6 つの設定を行うことができる。

通信方式

- ネットワーク接続方式をブリッジ接続とした場合
- ネットワーク接続方式を NAT 接続とした場合
- ネットワーク接続方式をホストオンリー接続とした場合

仮想ネットワークの通信速度

- 仮想 LAN ケーブルの通信速度を 100Mbps とした場合
- 仮想 LAN ケーブルの通信速度を 1Gbps とした場合
- 仮想 LAN ケーブルの通信速度を無制限とした場合

以上に示す 6 通りの設定状態において、次の 2 種類の通信速度計測の実験を執り行った。

1. nttcp を用いて 100MB・200MB・300MB・1GB のデータを転送し、データ送信終了までかかる時間を計測する。
2. FTP を用いて 100MB・200MB・300MB・1GB のファイルをコピーし、処理にかかる時間を計測した。なお、計測結果として FTP の転送終了時に表示される時間を用いるものとする。

6.3.4 実験結果および考察

以下には前述した各実験の実験結果を記載する。また、実験結果の単位は全て秒 (s) である。

仮想ネットワーク通信速度変更に伴う、nttcp ベンチマークテスト

仮想ネットワークの通信速度を変更した場合には、nttcp の結果を見てもらえば分かるとおり、確かな通信速度の変化が見て取れる。(図 6.25〜6.28 参照)

これだけ、値に変化が見られるのならば十分性能が異なる資源とみなすことも可能であると考える。

ただし、実機において nttcp ベンチマークテストを行った際には通信終了までの時間はほぼ同一の値が得られていたが、仮想マシン上ではやや値が不安定になっている。

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	10.70	10.81	10.62	11.13	10.88	10.83	77.45
1Gbps	6.33	7.48	7.08	6.78	7.03	6.94	121.24
無制限	7.91	6.65	5.98	5.85	5.44	6.37	128.84
実機	8.91	8.91	8.91	8.91	8.90	8.91	94.16

表 6.25: nttcp によるデータ総量 100MB 転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	19.81	21.45	20.82	21.21	21.87	21.03	79.84
1Gbps	20.41	14.32	16.60	17.04	12.11	16.10	107.74
無制限	12.31	11.30	10.71	11.99	12.33	11.73	143.48
実機	17.82	17.82	17.82	17.82	17.81	17.82	94.15

表 6.26: nttcp によるデータ総量 200MB 転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	30.25	31.76	33.02	30.95	31.87	31.57	79.79
1Gbps	21.28	21.07	21.41	21.74	22.56	21.61	114.35
無制限	16.36	16.97	17.88	19.38	17.91	17.70	142.52
実機	26.73	26.73	26.73	26.72	27.73	26.93	94.16

表 6.27: nttcp によるデータ総量 300MB 転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	106.44	103.26	105.82	106.73	103.81	105.21	79.73
1Gbps	67.20	74.97	75.20	78.86	83.55	75.96	111.04
無制限	58.79	66.08	68.31	76.04	72.95	68.43	123.53
実機	89.10	89.09	89.09	89.10	89.09	89.09	94.16

表 6.28: nttcp によるデータ総量 1GB 転送に要する時間

仮想ネットワーク通信速度変更に伴う、FTP ファイル転送実験

FTP ファイル転送実験の結果を見てみると、nttcp ベンチマークテストの結果同様、通信速度の設定した値に従って変化していることが分かる。(図 6.29〜6.32 参照)

しかし、FTP ファイル転送時には nttcp ベンチマークテストで見受けられたほどの大きな値の変化ではない。

この 2 つの実験における大きな違いは転送時に分割されているデータのサイズである。nttcp では 1 つのデータサイズは 1KB に固定しているが、FTP においてはそれを大きく上回るデータサイズで分割されたファイルの転送が行われている。

このことから、FTP ファイル転送実験時にはボトルネックが起こっているために、nttcp ほどの値の変化が見られなかったのではないかと考えられる。

したがって、VMware における通信速度の変更にはバンド幅を制御するような処理は含まれていないものと考えられる。

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	12	12	12	12	12	12	67.84
1Gbps	12	11	11	11	11	11	71.68
無制限	12	11	10	10	11	11	76.96

表 6.29: FTP による 100MB ファイルの転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	28	24	24	24	25	25	65.92
1Gbps	27	23	24	23	22	24	69.12
無制限	24	21	23	21	21	22	74.24

表 6.30: FTP による 200MB ファイルの転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	39	37	36	36	38	37	66.24
1Gbps	37	32	34	37	32	34	72.00
無制限	32	31	34	31	34	32	75.68

表 6.31: FTP による 300MB ファイルの転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
100Mbps	130	130	130	120	130	128	64.80
1Gbps	120	120	120	110	120	118	69.12
無制限	110	110	100	110	120	110	74.24

表 6.32: FTP による 1GB ファイルの転送に要する時間

通信方式変更に伴う、nttcp ベンチマークテスト

通信方式の変化に伴う、nttcp ベンチマークテストの結果を見てみると僅かながら値に違いが見られる。(図 6.33〜6.36 参照)

しかし、送信データ総量 100MB・200MB・300MB・1GB の各場合において最短時間で転送が終了している通信方式がまちまちであることからこれらの違いは誤差の範疇内であるものと考えられる。

したがって、nttcp ベンチマークにおいては通信方式の違いによる通信速度の変化は見られなかった。

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	7.20	5.33	6.80	5.68	6.40	6.28	132.61
ホストオンリー	5.48	7.63	5.34	6.32	7.11	6.38	134.20
NAT	7.85	6.19	6.69	5.49	6.83	6.61	125.75

表 6.33: nttcp によるデータ総量 100MB 転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	12.06	13.63	11.63	12.39	15.90	13.12	129.47
ホストオンリー	12.70	12.27	13.26	12.43	11.80	12.49	134.48
NAT	12.62	10.94	10.69	13.10	11.92	11.85	142.39

表 6.34: nttcp によるデータ総量 200MB 転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	16.91	18.56	18.39	18.49	18.77	18.22	138.17
ホストオンリー	21.66	16.24	20.92	18.17	19.19	19.24	132.22
NAT	22.71	24.72	22.24	17.72	17.14	20.91	122.91

表 6.35: nttcp によるデータ総量 200MB 転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	67.08	73.29	74.95	68.33	61.16	68.96	122.07
ホストオンリー	73.74	72.86	78.72	78.98	62.66	73.39	115.10
NAT	67.14	60.58	64.09	63.58	64.48	63.97	131.26

表 6.36: nttcp によるデータ総量 1GB 転送に要する時間

通信方式変更に伴う、FTP ファイル転送実験

通信方式の変更に伴う、FTP ファイル転送実験の結果を見てみるとブリッジ・NAT に関してはほぼ同様の結果であるが、ホストオンリー方式において僅かながら短時間で通信が終了していることが分かる。(図 6.37〜6.40 参照)

ただし、ホストオンリーにおける時間の短縮も微々たるものであり、性能の異なる資源としてみなせるかという見方からすれば、通信方式の変更による資源的コストの削減は見込めないと考えられる。

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	8	11	11	12	13	11	75.04
ホストオンリー	9	10	12	11	10	11	78.40
NAT	11	11	10	11	12	11	73.76

表 6.37: FTP による 100MB ファイル転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	23	22	23	24	25	23	70.24
ホストオンリー	26	22	22	21	22	23	72.48
NAT	25	23	25	25	24	24	67.52

表 6.38: FTP による 200MB ファイル転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	32	35	33	34	37	34	72.00
ホストオンリー	35	33	31	32	34	33	74.56
NAT	35	34	34	35	35	35	70.72

表 6.39: FTP による 300MB ファイル転送に要する時間

	1 回目	2 回目	3 回目	4 回目	5 回目	平均値	Mbps
ブリッジ	120	120	120	120	120	120	67.69
ホストオンリー	110	110	110	100	110	108	75.84
NAT	120	120	120	120	130	122	68.16

表 6.40: FTP による 1GB ファイル転送に要する時間

第7章　VMwareの時間的・資源的コスト削減に対する総評

5章からも分かるとおり、VMwareは次の2つの用法からGridミドルウェアのインストール時間の短縮を図ることができる。

- 新規に環境を構築する際には初期の段階でインストールを終えた仮想マシンのコピーを作成し仮想Gridコンピューティング環境を拡張することによってOSおよびGridミドルウェアのインストール時間を削減できる。
- 既に他者が構築してしまった仮想Gridコンピューティング環境を貰い受ける形で、自身に一切の知識・技術がない状態でも仮想Gridコンピューティング環境を構築することができる。ただし、この場合にはホスト名・IPアドレスの問題がついて回るためこの問題をごく簡単な操作で解決してくれる補助ツールを作成するなどして環境を受け取る側、受け渡す側の知識・技術の差を埋める必要性がある。

これら、2点を鑑みるに時間的コストの削減という面で見ればVMwareは大いに役立つツールとなるだろう。

資源的コストの面について考えてみると、6章からも分かるように各種設定の変更による仮想マシンの処理速度の向上というものは微々たるものである。したがって、リソース設定の変更により、仮想マシン実態の性能をユーザーの思うとおりに操ることは不可能である。

仮想ネットワーク通信速度設定機能に関してもある程度ユーザーの意図したとおりに通信速度を制御できることが確認できたが、計測結果にある程度のむらが見られる。通信速度関連の研究・開発の性質から考えると、やはりこの不安定性は問題である。

加えてディスクアクセスのベンチマーク結果に悪影響がでた用に今回の実験では取り扱わなかった類のベンチマーク実験においてもそういった問題が発生しないとは限らない。

したがって、資源的コストの削減の面からVMwareを見た場合、単純に1台のコンピュータを複数台のコンピュータとして扱う、すなわち実験に必要なコンピュータの頭数を揃えるだけならば効果があるといえるが、Gridコン

ピューティング環境を構築する上で望まれる異なる性能を持ったコンピュータを揃えるといった面では効果が期待できない。

以上の点を踏まえて VMware の Grid コンピューティングの研究・開発における総評を述べると、Grid コンピューティング環境上で動作するソフトウェアの開発・動作確認といった大まかな動作さえしっかりしていればよいという場面では非常に有効であるが、反面、今回行ったベンチマークのように細やかなデータの採取や動作検証を必要とするような場面ではまだまだ力不足であると言える。

謝辞

最後に、本研究に取り組むにあたり、多大なる指導をしてくださった奥村勝助教授ならびに、プログラム作成にあたり、土台となる技術等を指導していただいた同研究室の先輩に心よりの感謝を申し上げます。

参考文献

1. 「仮想化技術完全攻略ガイド」インプレスジャパン社発、平初・宮原徹
   等著

2. 「第 1 回グリッドトレーニング SGE と Ganglia のインストールと利用」
   Grid Consoritum Japan 発

3. 「Claris のわがまま　おもちゃ箱」
   （http://claris.hp.infoseek.co.jp/apply/bentch/bentch.html）

4. 「ファイヤープロジェクト」
   （http://www.fireproject.jp/pc_cluster/benchmark/communication/nttcp.html）

5. 「PC クラスタ管理システム Sun Grid Engine の概要」
   （http://mikilab.doshisha.ac.jp/dia/research/report/2004/0817/001/report20040817001.html）

6. http://www.turbolinux.co.jp/solution_service/
   library/features/softwaredesign_200401_amd64.html#LINK