ハイパフォーマンス
コンピューティング 91−23
（２００２．８．２２）

Lucie: 大規模クラスタに適した高速セットアップ・管理ツール

高宮安仁[†] 真鍋篤[†††]
白砂哲[†] 松岡聡[†,††]

コモディティクラスタリングシステムにおける，ノード数規模の急激なスケールアップに伴い，クラスタのセットアップおよび保守コストは増大しつつある。また、各ノードのローカルディスク上に数 GB 単位のデータベースファイルを必要とする、いわゆるデータインテンシブアプリケーションがクラスタ上で用いられつつあるが、インストール時における、大容量データのセットアップに着目したツールは重視されてこなかった。我々は、大規模クラスタ向けセットアップ/ 管理ツールである Lucie を開発している。Lucie では、インストール用メディアを用いないネットワークブート/ インストール機構、および用途に応じたインストーラ自体の再構成といった拡張機構を実現する。また、仮想リング構造による高速データ転送機構によって、耐故障性を保ちつつ、インストール時に高速に数 GB 単位のデータを全ノードへ配布することができる。本稿では、Lucie のインストール性能、およびインストール時の大容量データ配布性能について評価を行った。結果、性能はノード数によらずほぼ一定であり、本研究の将来的な目標であるプラグアンドプレイクラスタリングの基礎技術として有用であることがわかった.

Lucie: A fast installation and administration tool for large-scaled clusters

Yasuhito Takamiya,[†] Atsushi Manabe Satoshi Shirasuna[†††,†]
and Satoshi Matsuoka[†,††]

Rapid increase in the number of nodes for commodity clustering is mandating the handling the potential cost of setup and maintenance clusters as the norm. Moreover, with arising of data intensive applications which requires several GBs of data on each cluster nodes, it is revealed that there were no installation tool aimed at installation-time setup of such large-scaled data. In this paper, we propose a new cluster installation/administration tool called Lucie which allows network boot/installation mechanism with no specific installation media and configurability which allows reconstruction of installer itself on demand. Additionally, using data distribution mechanism with virtual ring topology network, one could distribute several GBs of images to all cluster nodes in installation-time maintaining fault tolerance. Our several benchmarks show that Lucie can install and setup whole cluster in constant time. This result shows that Lucie is scalable and efficient, and could well serve as a basis for Plug-and-Play clustering.

1. はじめに

高性能計算分野において、汎用の PC を Myrinet 等の高性能ネットワークハードウェアで多数連結し、スーパーコンピュータに対して数十倍の価格対性能比を目的とする、PC クラスタリングシステム (以下、クラスタ) が主流となりつつある。近年のネットワークハードウェアやユーザレベル通信技術の進歩によって、数千〜数万ノード超の大規模クラスタはすでに実現可能であると言えるが、実際には以下に挙げるような運用面における問題のため、実現は困難である。

新規にクラスタを運用開始する場合、すべてのノードへ OS やソフトウェアをインストールし、設定を行う必要がある。しかし、通常の単一ノード用インストーラを用いて逐次的に作業した場合、ノード数が増大するにつれ、こうした作業に要する手間は線型に増大する。また、ノード故障時の新品ノードとの交換/ 復旧、ノード間で設定ファイルに不整合が起こった場合の問題個所の特定/訂正作業、および、各ノードへインストールされているソフトウェアについて、セキュリティパッチ適用やバージョンアップといった定期的な更新作業はいずれも、各ノードへのリモートログインによる逐次的な手作業では手間がかかり、作業上の誤りを犯しやすい。

本研究では、大規模クラスタ用インストーラ・管理ツールとして、クラスタ全体を短時間のうちに、ネッ

† 東京工業大学 Tokyo Institute of Technology
†† 国立情報学研究所 NII
††† 高エネルギー加速器研究機構 KEK

トワーク経由でインストールを行うことのできる Lucie[1] を開発した。本稿では、Lucie を用いて、実際に 112 台のクラスタを同時にセットアップした。結果、セットアップに要する時間が 8 分弱とその有効性を確認した。また、大規模データのインストール時配布機構については、1GB のファイルをノード全体へ配布する場合について計測し、台数増加に対して要する時間はほぼ一定であることを確認した。また、Lucie を実際に東京工業大学松岡研究室の Presto クラスタ群[2] の運用に適用し、合計約 500 ノードのクラスタ運用に有用であることを確認した。

2. 要　　請

故障したクラスタノードを復旧する場合、もしくは新たに購入した Vanilla PC をクラスタノードとしてセットアップする場合、クラスタノードとして完全に動作させるためには、1) ハードディスクのパーティション構成 2) ハードディスク上のソフトウェア、OS 構成 3) ソフトウェア、OS の設定情報 4) ソフトウェアで用いられるデータ、についての設定が必要である。

通常の単一ノード用インストーラを用いた場合、1),2) は、いくつかのキーボード入力を行うことによって完了できる操作である。3) については、インストーラを用いて 1) の操作を行った後、ユーザがエディタ等を用いて行う作業である。4) については、通常、ユーザがアプリケーション使用前に、rcp/scp 等を用いて個別に行う作業である。

クラスタ用のインストーラ/管理ツールでは、単一ノード用インストーラにおける逐次的な部分を極力排除する必要がある。たとえば、クラスタのセットアップでは同時に複数ノードのセットアップを完了させる必要があるので、1)〜3) で必要となる、キーボード入力操作は不要でなければならない。また、4) でセットアップされるデータは、たとえば、ホモロジー検索ソフト blast[3] のデータベースファイルなど、数ギガバイトに及ぶ場合があるので、rsync や tar による逐次の単純なコピーではなく何らかのブロードキャスト的なコピー機構を用いる必要がある。また、配送に用いるネットワークデバイスには Myrinet などの高速なデバイスを用い、データのコピー操作自体、1)〜3) の操作と同時に、インストール時に自動的に行われることが望ましい。加えて、インストーラは起動用メディアを必要としないことが非常に重要である。これは、セットアップするノード数が増えるにつれ、準備しなければならないメディアの数も増えるためである。

3. Lucie

上記の要請を満たすためのツールとして、我々は Lucie を開発している。

Lucie システムは、Lucie クライアントへネットワーク情報を送信する BOOTP/DHCP サーバ、NFS 経由でのルートファイルシステムを提供する NFS サーバ、ソフトウェアパッケージを配布する http/ftp サーバ、ブートイメージである pxeboot, Linux カーネルを提供する TFTP サーバから成る Lucie サーバ群と、Lucie クライアント (インストールされるクラスタノード) から成る (図 1)。それぞれのサーバはそれぞれ個別のホストへセットアップすることもできるが、1 台のホスト上ですべてを動作させることもできる。Lucie によるインストールでは、インストール処理の各段階において、Lucie サーバ上の各種設定が Lucie クライアントへ順次送信されることによって行われる。

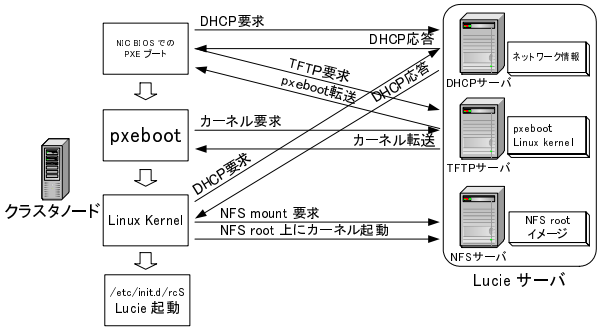

図 1　ネットワークブートによる Lucie の起動

Lucie によるインストールは、次の 2 段階に分けられる。第 1 段階では、ローカルディスクを使わずに NFS 上のディレクトリをファイルシステムとしてマウントし、Linux を起動するディスクレスブートを行う。第 2 段階では、ディスクレスシステム上から Lucie インストーラ本体を起動し、実際のインストール作業が行われる。

3.1 ディスクレスブート

ディスクレスブートでは、Lucie クライアントの NIC BIOS からネットワーク経由でブートイメージを取得し、NFS サーバ上のルートファイルシステムイメージをマウントし、ディスクレス構成の Linux がブートする。

以下にディスクレスブート機構の詳細を説明する (図 1)。PXE/MBA 機構[4),5)] に対応した NIC では、NIC 内の BIOS が起動し、PXE のブート ROM を読み込んだ後、DHCP 要求をブロードキャストし、DHCP/BOOTP サーバより自ホストの IP アドレス、ゲートウェイアドレス等のネットワーク情報を取得する。ネットワーク情報取得後、NIC BIOS は TFTP サーバに対して pxeboot (preboot execution environment boot) を要求し、取得した後 pxeboot を起動する。

pxeboot では TFTP サーバから Linux カーネルイメージを取得する。このカーネルイメージは DHCP/BOOTP が有効になったものである。カーネルイメージ取得後、pxeboot はカーネルを起動する。

起動した Linux カーネルは、DHCP/BOOTP 要

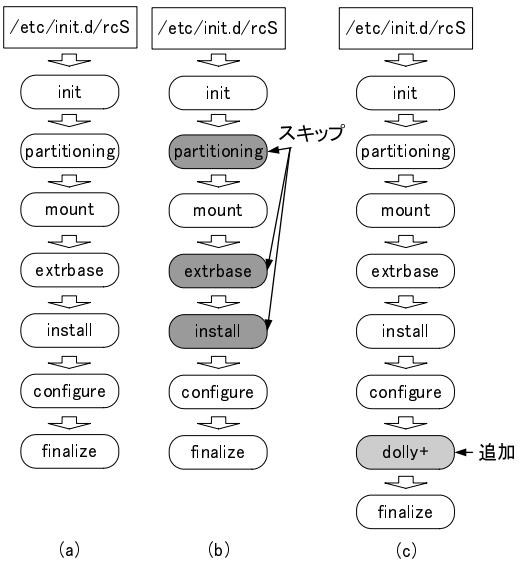

図 2 Lucie インストーラの構成

求によりネットワーク情報、NFS root が存在するホストの IP アドレス、および NFS root のパスを取得する。パスが取得できた場合、Lucie クライアントはこのディレクトリを自ホストの root ファイルシステムとしてマウントし、Linux が起動する。ここまでの機構は一般的なディスクレスシステムで用いられている機構と同様である。

KickStart[6] 等の一般的なインストーラでは、インストーラの実行環境として、メディア上にあらかじめ (圧縮され) 保存された Linux のディスクイメージを用いる。このため、Myrinet 等、Linux 標準添付でないドライバを必要とするハードウェアやソフトウェアをインストール時に使用することは難しい。一方、Lucie ではインストーラの実行環境として NFS root を用いている。この場合、Linux のディスクイメージは NFS サーバ上に通常のファイルシステムとして存在するため、ユーザは、NFS 上のディスクイメージへ追加パッケージをインストールすることによって、インストール中における、任意プロセスの起動、カーネル中への動的なモジュール追加といった設定ができる。

Lucie クライアントはルートファイルシステムをマウント後、/etc/init.d/rcS を起動する。この /etc/init.d/rcS は Lucie 独自のものと置き換わっており、Lucie インストーラを開始する。

3.2 Lucie インストーラ

/etc/init.d/rcS から起動される Lucie インストーラはモジュールに分割されており、ユーザによってあらかじめ指定された順に実行される (図 2)。 主なモジュールの機能を以下に示す。

init Lucie インストーラの環境変数や tty 等の初期化を行う

partition ローカルディスクのパーティショニング、フォーマットを行う

mount ローカルディスクを NFS root の /tmp/target ディレクトリへマウントする

extrbase Linux の最小基本システムを含む tar.gz を/tmp/target にマウントしたローカルディスク上へ展開する。

install /tmp/target へ chroot し、ユーザによって指定された各ソフトウェアパッケージをインストールする。

configure /tmp/target/etc 以下のファイルについて、各ホスト・クラスタに応じた書換えを行う。

finalize インストーラの終了処理を行う。

モジュール実行順序の設定例を図 2 に示す。図 2(a) は通常のインストール、すなわちローカルディスクのパーティショニング、マウント、およびソフトウェアのインストール、/etc ファイルの設定まですべてを行った場合のモジュール実行例である。図 2(b) はパーティショニング、Linux 基本システムのインストール、ソフトウェアパッケージのインストールモジュール実行を省略している。このため、フルインストールに比べて実行時間の短い、/etc の再設定のみを行うインストーラが定義されている。

インストーラの拡張機構として、ユーザは、用途に応じた独自のモジュールを定義し、実行することができる。例として、インストール実行中に Lucie クライアントへリモートログインし、インストールモジュールのデバッグを行いたい場合を考える。ユーザは Lucie サーバ上の NFS root へ ssh/rsh サーバをインストールし、インストーラへ sshd/rshd 起動モジュールを追加することによって、インストール中に sshd/rshd を立ち上げ、リモートログインすることができる。Lucie では、こうしたユーザ定義モジュールを容易に作成するための helper class や script を提供している。これらを用いることによって、モジュールスクリプト中でのパーティショニング処理やソフトウェアパッケージ処理を簡単に記述できる。

3.3 大容量データのノード間配布

Lucie では、インストール時大容量データのノード間配布機構を、dolly+[7] を用いて実現している。

dolly+ は、ディスククローニングソフトウエアの一種である、dolly[8] へ耐故障性機構、高速化などいくつかの拡張を行ったものであり、複数台のホスト間で、ネットワークを介してファイルあるいはディスクイメージを複製する。通常のファイルシステム上のファイルとともに、デバイスやパイプからの順次アクセスファイルを転送することができる。

一般に、このようなクローニングソフトウエアはサーバ・クライアント方式により実現されているが、対象のホスト数が 100 ～ 1000 台以上と多く、また転送するファイルサイズが数 GB と非常に大きい場合、サーバボトルネックが発生し性能が著しく低下する可能性がある。こうした状況に対処するために、dolly+ では

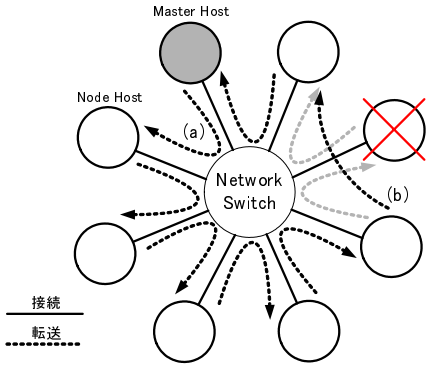

図 3 dolly+ での論理ホストリング機構

対象ホストをマスターホスト (コピーされるイメージを持つホスト) を先頭としたリング状に接続し、直列転送を行う (図 3(a))。こうして一カ所にボトルネックが発生することを回避し、全二重ネットワークスイッチの性能を最大限に引き出している。

転送は直列に Host by Host で行われるが、高速化機構として、ホスト間でのパイプライン転送、およびホスト内での $network \rightarrow memory, memory \rightarrow disk, memory \rightarrow netowrk$ といったマルチスレッディング処理を行っている。

ホストが直列に接続されることから、接続されたホストのいずれか一台でも異常が生じるた場合、転送すべてが停止する可能性がある。dolly+は耐故障性機構として、I/O バッファ監視と転送タイムアウト機構により異常を検出し、異常のあるホストを自動的に論理ホストリングから排除することにより、転送を続行する (図 3(b))。

Lucie インストール時の dolly+ の起動は、dolly+ クライアントを起動する dolly+ モジュールをインストーラ定義へ追加し、NFS root 生成時に追加パッケージとして dolly+ をインストールすることにより実現している (図 2(c))。

3.4 設定ファイル

Lucie の設定ファイルでは、主に 1)NFS root 構築用の設定 2)Lucie インストーラ動作、の 2 種類の設定を行う。

1)NFS root 構築用の設定では、NFS root 環境の Linux のバージョン、ディストリビューション名、root のパスワード、Linux 基本システムの取得方法 (ローカルディスク上の圧縮ファイルもしくは ftp/http)、NFS root へインストールするパッケージリスト等を記述する。2) Lucie インストーラの動作設定では、インストーラの各段階で実行される、各モジュールに対する設定ファイルを記述する。例えば、install モジュールの設定ファイルとして、インストールするソフトウェアパッケージのリスト、また partition モジュールの設定ファイルとして、各パーティションのサイズや用いるファイルシステムの種類を記述する。

```
nisdomain  = "yp-domain.titech.ac.jp"
domainname = "is.titech.ac.jp"
http_proxy = "http://proxy.is.titech.ac.jp:8080/"
subnet     = "255.255.255.0"
# 以下、クラスタ 'prestoI' の設定
prestoI {
        luciesrv =   "192.168.0.2"
        gateway  =   "192.168.0.1"
        dns1     =   "131.112.35.1"
        dns2     =   "131.112.35.2"
        kernelimage =
          "kernel-image-2.4.18_lucie.1_i386.deb"
        # 以下、prestoI クラスタ内のノード群
        psi00 {
             ip  = "192.168.0.3"
             mac = "00022D055293"
        }
        psi01 {
             ip  = "192.168.0.4"
             mac = "00022D055285"
        }
             ...
}
```

図 4 クラスタ構成の定義ファイル (/etc/lucie/machines)

```
#!/usr/bin/env ruby
require 'lucie'
require 'lucie/machines'

Lucie::File::open '/etc/mpich/machines.LINUX' { |f|
  f << "localhost\n"
  Lucie::Config::Machines::hosts(
                Lucie::Config::CLUSTER).each { |h|
    f << h << "\n"
  }
}
```

図 5 MPICH の設定ファイル /etc/mpich/machines.LINUX を生成するスクリプト

図 4 は Lucie でセットアップするクラスタ名、およびそれぞれのクラスタに含まれるノード、NIC の MAC アドレスやネットワーク情報の設定ファイルである。 configure モジュールは、この設定ファイルを読みこみ、ユーザによって指定されたスクリプトを実行し、クラスタノードの /etc ファイルの書換えを行う。例として、図 5 のスクリプトは MPICH 用の設定ファイル/etc/mpich/machines.LINUX をインストール時に生成する。

Lucie では、NFS root を構築するためのコマンドとして lucie-setup を提供している。lucie-setup コマンドは、1) 2) の設定を基にして、NFS や TFTP, DHCP サーバといった Lucie サーバの設定を自動的に行い、Lucie インストーラと NFS root を生成する。

4. 実　　験

図 6 に東京工業大学松岡研究室での Lucie を用いたクラスタ運用形態を示す。図中の (灰色) ホスト lucie, ponta, goucie 上では Lucie が動作しており、それぞれ PrestoII PrestoIII ion クラスタ、PrestoI クラスタ、OBI testbed クラスタを管理している。 今回、実験に用いる環境として、図中の PrestoIII クラスタ (CPU: Athlon MP 1900+ × 2 (SMP), Memory: 768MB, HDD 40GB, OS: Linux 2.4.18 × 255 nodes, Network: 100base-T connected with switching hub, Myrinet2000) を用いた。実験では、dolly+ によるファイル配布性能、および Lucie によるインストール性能を調査した。

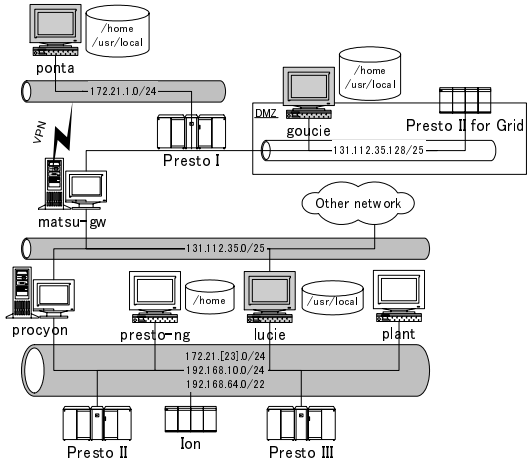

図 6 クラスタ構成図

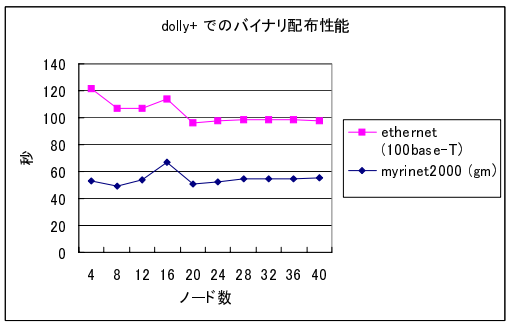

図 7 dolly+ によるファイル配布性能

4.1 dolly+ によるファイル配布性能

図 7 に、ネットワークとして ethernet(100 base-T) または Myrinet(GM[9]) ドライバ使用) を用いた場合の、dolly+ による 1GB のファイル配布性能を示す。なお、dolly+ では、配布終了時にすべてのスレーブノードからマスターノードへ終了通知が送信され、マスターノード上にログを出力する。よって、dolly+ が配布に要する時間として、マスターノード上での配布開始時刻と終了通知受信時刻の差を用いた。グラフより、dolly+ を用いた場合、配布対象のノード数が増加した場合でも要する時間はほぼ一定であり、100base-T では約 110sec/1GB、Myrinet(GM) では 60sec/1GB であることがわかる。このことより、インストーラの dolly+ モジュール中で GM モジュールを動的に Linux カーネルへ読み込み、データ転送に用いるデバイスとして通常の ethernet カードではなく Myrinet デバイスを有効にすることによって、インストール時のデータ配布性能を向上できることがわかる。dolly+ による転送時間がほぼ一定である理由として、dolly+ の仮想リング構造により、通信のボトルネックを回避していることが挙げられる。

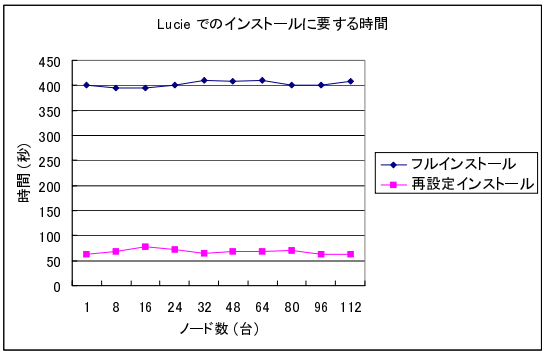

図 8 lucie によるインストール性能

4.2 Lucie によるインストール性能

図 8 に、Lucie を用いてフルインストール、再設定の 2 種類のインストールを行った場合に要する時間とノード数の関係を示す。なお、フルインストール実験では、212 パッケージ、合計約 300 MB (パッケージとして圧縮時の容量) をインストールした。また、すべてのノードについて、同時にインストールを開始するために、TFTP で配布されるインストール用カーネルをハードディスクへあらかじめ書き込んでおき、全ノードをリブートすることでインストーラを開始した。グラフより、Lucie 配布対象のノード数が増加した場合でも要する時間はほぼ一定であり、フルインストールを行った場合では約 400 秒、また再設定のみを行うインストールでは約 60 秒であることがわかる。ノード数の増加がインストール時間へ影響しない理由として、最もボトルネックになりやすい部分である、各クライアントが Lucie サーバへアクセスする時間 (パッケージ取得時、最小構成 Linux のダウンロード時) は高々 30 秒程度であり、インストール時間全体に対する割合が低いためであると考えられる。

5. 関 連 研 究

クラスタのセットアップおよび管理ツールに関する研究は多数行われている。しかし、その大部分は単一ノード用自動インストーラをクラスタへ適用したものがほとんどである。Lucie が達成しているような、インストール用メディアを必要としないネットワークブート/インストール機構、インストーラ自体の拡張機構、および高速データ転送などといった、クラスタに特化した機能を達成したものは少ない。

RedHat KickStart[6] は RedHat 社が配布する自動 Linux インストーラである。特長は、RedHat Linux 標準インストーラでインストールを行うことによって、同様のインストールを行う KickStart 用の設定ファイルを自動的に生成できる点、またネットワークブート/インストールに対応している点である。欠点として、KickStart は単一ノードの自動インストーラであり、

他のクラスタインストーラが達成しているような、複数ノードを単一クラスタとして扱い設定/運用する、といった機能は無い。

NPACI Rocks[10] は Lucie と同様、再インストールによってクラスタノード上のソフトウェアアップグレードや復旧といった操作を簡便に行うことを目的としたツールである。NPACI Rocks では XML ファイルにクラスタの論理構造を記述し、これを kpp と呼ばれるトランスレータを用いて RedHat KickStart の設定ファイルへ変換し、RedHat KickStart の CD-ROM をノード台数分作成することができる。この方式の欠点は、ノード台数分の CD-ROM を準備しなければならない点である。また、基本的にインストーラ自体は KickStart であるので、KickStart 以上の機能は無く、インストーラ自体の拡張や、インストーラが動作する Linux の環境をカスタマイズし、任意のソフトウェアをインストール時に実行するといった機能など、Lucie が実現しているいくつかの機能はない。

FAI[11] は Debian/GNU Linux 用のネットワークインストールツールである。FAI は Rock と同様、ノードを論理グループとして階層グループ化し、グループ単位での透過的な設定を行えるほか、Lucie と同じく、スクリプトによってインストーラ自体を拡張するなどといった細かい設定を行うことができる。欠点として、設定が非常に繁雑であり、PXE や DHCP などの設定ファイルをユーザがすべて手書きする必要がある点が挙げられる。

Oscar[12] は Open Cluster Group によるクラスタ構築/管理ツールである。Open Cluster Group ではグリッドのエンドポイントとしてのクラスタに着目し、クラスタのソフトウェア構成の標準化に取り組んでいる。Oscar を用いることによって、Open Cluster Group 標準互換のクラスタシステムを構築することができる。欠点として、Oscar では決まりきった構成のクラスタを構築することはできるが、柔軟な設定には対応していない。

6. まとめと今後の課題

我々は、大規模クラスタ用の高速セットアップ/管理機構として、Lucie を開発した。本稿ではインストール時のデータ配布機能について、dolly+ を用いて配布対象のノード台数を変えて計測した。その結果、性能は台数によらずほぼ一定であり、40 ノードへ 1GB のデータ配布をする場合、約 110 秒 (100base-T), 約 60 秒 (Myrinet2000) であることを確認した。 また、実際に 112 ノードを Lucie を用いてセットアップし約 400 秒とその有効性を確認した。

現在、Lucie ではノードの問題をすべて再インストールによって解決している。将来的には XML 等によるクラスタ設定の宣言的な記述から Lucie 設定、問題監視・復旧エージェントを生成することにより、いったん Lucie によってインストールを行った後は、エージェントが問題発生の監視/自律的な回復を行うようにする。

Lucie を用いたクラスタの動的な再構築, スケジューリングに関する試みとして, 我々は並列チェックポインティング/マイグレーション機能をもつ Parakeet MPI[13] を開発し、テストベッドとして ion プラグアンドプレイクラスタ[14] を構築している. 将来は計算を止めないノードのプラグアンドプレイ機構として、クラスタ上の並列プロセスを Parakeet MPI により定期的にチェックポインティングし、障害発生時には Lucie を用いて故障ノードを新規インストールノードと交換、マイグレーション/リスタートにより並列プロセスを復活するといった機構についても調査する。

参 考 文 献

1) Lucie web page. http://cluster-team.is.titech.ac.jp/lucie/.
2) Cluster team @ matsuoka lab. web page. http://cluster-team.is.titech.ac.jp/.
3) NCBI blast home page. http://www.ncbi.nlm.nih.gov/BLAST/.
4) Intel wired for management (wfm). http://www.intel.com/labs/manage/wfm/index.htm.
5) 3com support library - dynamicaccess managed pc boot agent (mba) downloads. http://support3com.3com.com/infodeli/tools/nic/mba.htm.
6) Redhat linux kickstart information. http://wwwcache.ja.net/dev/kickstart/.
7) Dolly+ home page. http://corvus.kek.jp/~manabe/pcf/dolly/index.htm.
8) Dolly - a program to clone disks. http://www.cs.inf.ethz.ch/CoPs/patagonia/dolly.html.
9) Myrinet gm software for linux. http://www.myri.com/scs/linux.html.
10) Mason J.Katz, Philip M. Papadopoulos, and Greg Bruno. Leveraging standard core technologies to programmatically build linux cluster appliances. *CLUSTER2002: IEEE International Conference on Cluster Computing*, April 2002.
11) FAI(fully automatic installation) home page. http://www.informatik.uni-koeln.de/fai/.
12) OSCAR: Open source cluster application resources. http://oscar.sourceforge.net/.
13) 高宮安仁, 松岡聡. ユーザ透過な耐故障性を実現する MPI へ向けて. 情報処理学会 電気通信処理学会 並列処理シンポジウム JSPP2002 論文集, pp. 217–224, 2002.
14) ion cluster web page. http://cluter-team.is.titech.ac.jp/ion/index.html.