Transactions of JSCES, Paper No.20080009

ディスクレスノードとディスクフルノードが混在するクラスタシステム環境をセットアップできるツールの開発*

Development of PC cluster setup tool for intermingled environment of diskless and diskfull nodes

中尾昌広[1]，廣安知之[2]，三木光範[2]

Masahiro NAKAO, Tomoyuki HIROYASU and Mitsunori MIKI

[1] 同志社大学大学院工学研究科 (〒610-0321 京田辺市多々羅谷 1-3)

[2] 同志社大学工学部 (〒610-0321 京田辺市多々羅谷 1-3)

In this paper, we propose a new setup tool for a PC cluster system called DCAST. DCAST can setup a PC cluster system in an intermingled environment of diskless and diskfull nodes. The unique function of DCAST is the mechanism to design the network of diskfull nodes and diskless nodes. The network is automatically optimized to avoid loads from concentrating on a single diskfull node when the diskless nodes are booted. To verify the utility of this function, we measured the file read access speed and the processing time of applications for cluster, and verified fault-tolerance of the diskless nodes. The results showed that the file read access speed improved by approximately 30%, the processing time improved by approximately 5%, and the number of system failures decreased when the concentration of loads to a single diskfull node was avoided. Therefore, DCAST is useful for constructing large-scale PC clusters.

Key Words: PC Cluster, Clustering, Software, High Performance Computing, Linux

1. はじめに

現在 HPC 分野では，高価なスーパーコンピュータから，多数のパーソナルコンピュータ（以下 PC）を用いた低コストかつ高速な演算処理を行える PC クラスタリングシステム（以下クラスタ）が主流となっている．コモディティ製品の性能向上により，数千〜数万ノードのクラスタの構築が可能になった．しかし，大規模なクラスタの構築が可能になったため，その消費電力と故障率がノード数に応じて大きくなるという問題が発生してきた．

それらの問題点を解決する 1 つの方法として，ノードにハードディスク（以下 HDD）を用いないディスクレスノードを主としたクラスタを構築することが挙げられる．ディスクレスノードはネットワークを通して，別のサーバから Operating System（以下 OS）や計算を行うために必要なファイルを取得し，計算を行う．GigabitEthernet に代表される高速なネットワークの普及により，ディスクレスノードは HDD を積んだノード（以下ディスクフルノード）と変わらない計算能力を発揮できるようになった．

大規模な環境でディスクレスノードを主としたクラスタを構築する場合，1 台のサーバが全てのディスクレスノードに対してルートファイルシステムを提供するのではなく，複数台のディスクフルノードを用意し，それぞれがディスクレスノードに対しルートファイルシステムの提供を行うといった負荷分散の工夫が必要になると考えられる．しかし，その作業は初心者には難しく，また通常のディスクレスノードの構築と比較して手順が増えるため作業者の負担も大きい．

以上のことから，クラスタの大規模化・ディスクレス化に対応したセットアップツールがあれば，有用であると考えられる．そこで我々は，ディスクレスノードとディスクフルノードが混在するクラスタ環境のセットアップを行うツール DCAST(Dynamic Cluster Auto Setup Tool) の開発を行っている．

DCAST は下記のような機能を有する．

- ネットワークを利用したクラスタイメージの自動インストールおよび設定
- ノードの HDD の有無を自動判別
- ディスクレスノードとディスクフルノードの対応

* 原稿受付 2007 年 12 月 18 日, 改訂年月日 2008 年 04 月 02 日, 発行年月日 2008 年 04 月 21 日,

関係の自動設計

ユーザはDCASTを用いることによって，クラスタを構成するノードのHDDの有無を意識することなく，ディスクレスノードとディスクフルノードが混在したクラスタ環境のセットアップを行うことが可能である．

本稿では，DCASTの設計と実装，そしてDCASTの有用性について述べる．特に有用性では，DCASTの主機能である”ディスクレスノードとディスクフルノードの対応関係の自動設計”に注目し，ベンチマークプログラムによる性能測定と，シミュレーションによる耐故障性についての実験を行った．実験では，複数台のディスクレスノードに1台のディスクフルノードがルートファイルシステムの提供を行った場合と，複数台のディスクフルノードが均等にルートファイルシステムの提供を行った場合とを比較し，後者の方が良い結果を得られることを確認した．また，DCASTを用いることで，ディスクレスノードとディスクフルノードが混在するクラスタ環境をセットアップする場合における作業者の負担軽減についての検証実験を行った．その結果，既存のクラスタリングソフトウェアと比較してDCASTが最も作業者の負担が少なく，クラスタ環境をセットアップできることを示した．

2. DCASTの設計と開発

2.1 要請 クラスタを構成する各ノードは単体で動作するコンピュータであるため，それぞれにOSとソフトウェアのインストール作業が必要である．それらの作業に要する労力はノード数が増えるほど大きくなり，かつ設定を間違える可能性も大きくなる．そのためDCASTは下記の要件を満たしている．

- インストール時の対話操作の排除
  通常のOSとソフトウェアのインストールには多くの対話操作が必要である．大規模なクラスタのセットアップを行う場合，繰り返される対話操作は大きな労力となる．このため，DCASTでは対話操作を一切行わないこととし，インストールを自動化することによってクラスタのセットアップ作業を簡易化する．また，CDなどのメディアを用いずに，PXE(Preboot eXecution Environment)ブート[(1)]を利用したネットワークインストールを行うことで，ユーザの負担を減らす．
- モジュールスクリプトの追加機能
  クラスタの利用用途は様々であり，各ノードにおいてインストールしたいソフトウェアが異なる場合や，設定を変える場合がある．これらに対して柔軟に対応するために，ユーザがモジュールスクリプトを作成・追加できる仕組みが必要である．

2.2 DCASTクラスタの構成 DCASTがセットアップを行うクラスタ環境はFig.1のような構成を想定している．

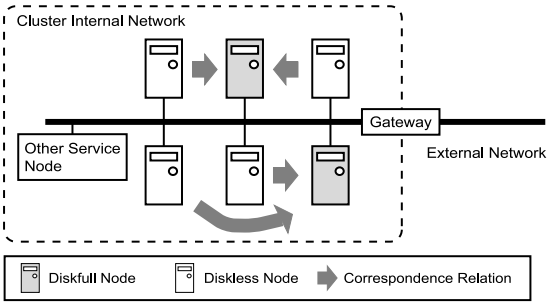

Fig.1 Composition of PC cluster by DCAST

クラスタの内部ネットワークにディスクレスノードとディスクフルノードが存在し，それぞれのディスクレスノードは，いずれかのディスクフルノードとネットワークマウントの関係にある．

なお，一般にクラスタで最もノード数が多いのは計算ノードであることから，DCASTは計算ノードのみのセットアップを目的としている．他のサービスを提供するノード（ユーザ管理，ファイル管理，ジョブスケジューラ等）はすでに存在しているとする．

2.3 DCASTクラスタのソフトウェア環境 クラスタを利用したいユーザが初心者である場合，どのようなソフトウェアをインストールするべきかを判断することは困難である．そのため，DCASTではクラスタでよく用いられる下記のオープンソースソフトウェアのインストールと設定を自動的に行う．

- MPICHを用いた並列計算環境
- NIS（Network Information Service）を用いたユーザ管理
- NFS（Network File System）を用いたファイル管理
- dsh（distributed shell）を用いた分散シェル環境

DCASTは，DCASTをインストールしたマシン（以下DCASTマスタノード）のルートファイルシステムをもとにクラスタイメージを作成し，それを計算ノードに配布することでセットアップを行う．そのため，上記以外のソフトウェアを計算ノードで動作させたい場合，DCASTマスタノードにそのソフトウェアをインストールした後にDCASTを実行することで，セットアップされた計算ノードでそのソフトウェアを利用することが可能である．また，Fig.1中にある他のサービスを提供しているノードと計算ノードとを連携させたい場合も，DCASTマスタノードにその設定を行えば，セットアップされる計算ノードでも同じ設定がなされる．なお，そのサービスに対し各ノードで個別の設定が必要な場合は，2.1節で説明したモジュールスクリプトをユーザが作成することで対応可能になる．

2.4 DCASTの開発 DCASTでは，ディスクレスノードとディスクフルノードの両方を適切な関係でセットアップを行うために，以下のキーコンポーネントを実装している．

- HDD の自動判別機能
  クラスタのセットアップ時にユーザが全ノードの HDD の有無をチェックすることは負担が大きい．そのため，クラスタのセットアップ時に DCAST は全ノードの HDD の有無を自動的にチェックし，HDD がある場合はディスクフルノード，HDD がない場合はディスクレスノードとしてセットアップを行う．
- ディスクレスノードとディスクフルノードの対応関係の自動設計機能
  DCAST は，HDD をチェックした情報を収集し，ディスクレスノードが均等にディスクフルノードに割り当てられるように設計を行う．

DCAST では，上記以外のキーコンポーネントとして grub，rsh，dhcpd，tftpd といった既存のオープンソースソフトウェアを利用している．DCAST はそれらのソフトウェアのインストール，設定を行うことで，複数台の PC をクラスタとして動作させる．この動作の詳細は 3.2 節で説明する．以上のキーコンポーネントを用いることで，ユーザは対話操作を行わずに，ディスクレスノード，ディスクフルノードのセットアップ作業を行うことができる．

DCAST は bash，Ruby といったスクリプト言語を用いて開発した．bash は OS の機能を直接操作するために用いている．また，DCAST は Ruby から bash を呼び出すように作成しているため，例外処理に関する処理も可能になっている．

なお，HDD を積んだ PC であっても，ディスクレスノードとして動作させることも可能にする．この機能を用いることで，例えば平日に利用されている PC を，夜間や休日にのみディスクレスノードとしてセットアップすることで，既存の環境を変更することなくクラスタの計算資源として利用することが可能である [(2)(3)]．

2.5 DCAST の特徴 DCAST はディスクレスノードが均等にディスクフルノードに配分されるようにネットワークマウントの関係を設定する．このことから考えられる利点を下記に示す．

- パフォーマンスの向上
  ディスクレスノードでジョブを実行する際，ディスクレスノードはネットワークマウントしているディスクフルノードからジョブに必要な実行ファイル，ライブラリを取得する必要がある．ディスクレスノードを各ディスクフルノードに均等に配分することにより，ネットワーク負荷とディスクフルノードの HDD の負荷を分散することができる．そのため，均等に配分しない場合と比較してファイル I/O のパフォーマンスが高くなると考えられる．
- 耐障害性の向上
  全てのディスクレスノードが 1 台のディスクフルノードにネットワークマウントしている場合，そのディスクフルノードが故障すると，全てのディスクレスノードが動作不能になる．ディスクレスノードをディスクフルノードに均等に分散することで，計算ノード全体の耐障害性は高くなると考えられる．

これらについては 4 章で検証を行う．

3. DCAST を用いたクラスタ構築手順

3.1 DCAST のセットアップ手順

1. OS と DCAST のインストール
   ユーザはディスクフルノードの内の 1 台に OS と DCAST をインストールする．DCAST では対応 OS として Debian GNU/Linux[(4)] を用いている．Debian GNU/Linux には優れたパッケージ管理機構があり，初心者でも容易にソフトウェアのセキュリティアップデート，パッケージ同士の競合や依存関係の解消を行えるからである．
2. 全計算ノードの MAC アドレスの収集
   セットアップ対象の計算ノードは PXE ブートを用いて起動するため，全計算ノードの MAC アドレスを収集する必要がある．その作業を簡易化するため，DCAST ではネットワーク上に流れるパケットから MAC アドレスを抽出するコマンドを用意している．そのコマンドを利用することで，ユーザは PXE ブートで起動するように設定を行った計算ノードを起動させ，DCAST マスタノードと同じネットワークに接続させるだけで，MAC アドレスの収集を行うことが可能である．
3. モジュールスクリプトの作成（任意）
   各計算ノードに対し個別の設定を行う必要がある場合，ユーザは DCAST の用意するディレクトリに*.masterconfig と*.slaveconfig というファイル名でモジュールスクリプトを配置する．ユーザは*.masterconfig のモジュールに DCAST マスタノードに対する設定を行う命令を記述し，*.slaveconfig のファイルにセットアップを行う計算ノードに対する設定を行う命令を記述する．DCAST は実行直後にそれぞれのモジュールスクリプトを実行させることで，DCAST マスタノードと各計算ノードに対し個別の設定を行うことが可能になる．なお，その際 dcast-parse というコマンドを利用することによって，セットアップを行う計算ノードのホスト名，IP アドレス等を出力できる．このコマンドをスクリプト中で呼び出すことによって，スクリプトを簡潔に記述することが可能である．*.slaveconfig モジュールの例を Fig.2 示す．
4. DCAST の実行
   2 において MAC アドレスが集まった時点で Fig.

```
#!/bin/sh

ROOT=/tftpboot
NIS_SERVER=192.168.1.123

# edit slave's nis files
sed -e 's/compat/nis files/g' $ROOT/etc/nsswitch.conf \
                                        > /tmp/nsswitch.conf

dcast-parse ip ¦ while read IP
do
    echo "ypserver $NIS_SERVER" >> $ROOT/$IP/etc/yp.conf
    echo "+::::::" >> $ROOT/$IP/etc/passwd
    echo "+:::" >> $ROOT/$IP/etc/group
    cp /tmp/nsswitch.conf $ROOT/$IP/etc/nsswitch.conf
done

rm /tmp/nsswitch.conf
```

Fig.2　Example of module script

```
### MASTER CONFIG ###
eth = eth0
ip_address = 192.168.1.1
netmask = 255.255.255.0
network = 192.168.1.0
broadcast = 192.168.1.255
### CLUSTER CONFIG ###
prefix = nova
domain = doshisha.ac.jp
swap = 1024M
### SLAVE CONFIG ###
#NAME          IP ADDRESS        MAC ADDRESS
nova0001       192.168.1.2       00:AB:CD:EF:12:34
nova0002       192.168.1.3       00:AB:CD:EF:12:35
### END OF FILE ###
```

Fig.3　DCAST configration file

3 のような DCAST の設定ファイルが作成される．Fig.3 にあるホスト名やネットワークの設定は，MAC アドレス収集時のコマンドで指定する．ユーザは設定ファイルを確認後，DCAST を実行する．ユーザの行う動作は以上であり，残りのセットアップ作業は全て自動的に進行する．

3.2　DCAST の動作　本節では，DCAST 実行後の DCAST の内部動作について述べる．

1. DCAST マスタノードの設定
   DCAST マスタノードでは，DHCP，TFTP，NFS のサービスが動作している．DCAST は Fig.3 に示した DCAST の設定ファイルから，これらのサーバに対する適切な設定を行う．
2. ルートファイルシステムの準備
   DCAST マスタノードをもとにして，計算ノードにインストールするためのクラスタイメージを作成する．また，全計算ノードが PXE ブートする際必要な NFS マウント用の領域も作成する．
3. 計算ノードの動作
   PXE ブートで起動した計算ノードは，DCAST マスタノードに対し IP アドレスとカーネルを要求する．DCAST マスタノードは DHCP と TFTP の機能により，その要求に答える．次に計算ノードは，DCAST マスタノードに用意されたルートファイルシステムにネットワークマウントし，

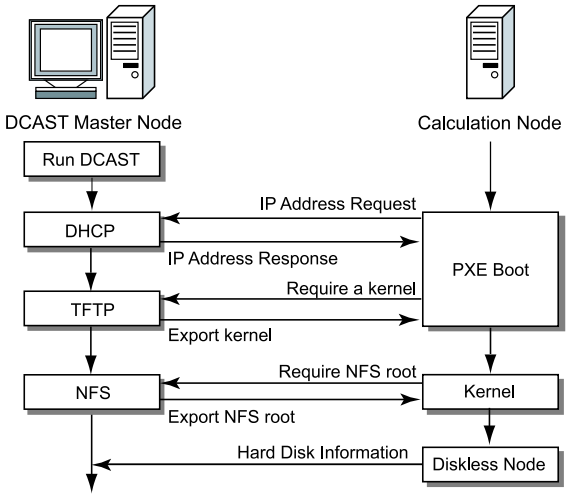

Fig.4　Mechanism of network boot

ディスクレスノードとして起動する．その際に，自ノードの HDD の有無を DCAST マスタノードに通知する．計算ノードがすべてディスクレスノードの場合は，この段階で処理が終了する．これらの動作の概要を Fig.4 に示す．

4. ネットワークマウント構成の自動設計
   DCAST は得られたディスクフルノードとディスクレスノードの数から，各ディスクフルノードに同じ数のディスクレスノードを割り当てるように設計を行う．もし，ディスクレスノードの数がディスクフルノードで割り切れない場合，余ったディスクレスノードを任意のディスクフルノードに 1 台ずつ割り当てる．
5. ディスクフルノードのセットアップ
   DCAST はディスクフルノードに対し，HDD のフォーマット，ファイルシステムの作成，クラスタイメージのインストールを行う．計算ノードがすべてディスクフルノードの場合は，この段階で処理が終了する．
6. ディスクレスノードのセットアップ
   DCAST によってセットアップされたディスクフルノードは，DCAST マスタノードで作成された設定に従い，該当するディスクレスノードがネットワークマウントするように設定を行う．ディスクレスノードはその設定終了後に再起動し，該当するディスクフルノードにネットワークマウントし，セットアップが完了する．

4. 検証

4.1　ディスクレスノードの read 性能の検証　本節では，ディスクフルノードにディスクレスノードを均等に分散させた場合とそうでない場合における，ディスクレスノードのファイルの read 性能の違いについて

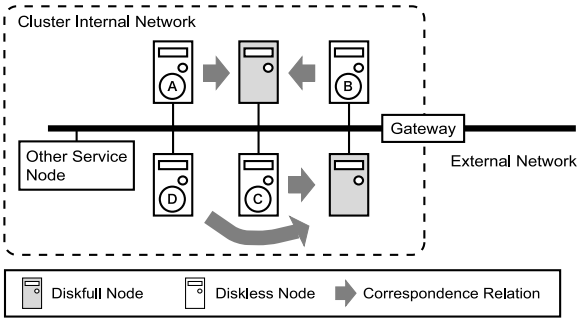

Fig.5　Decentralized composition

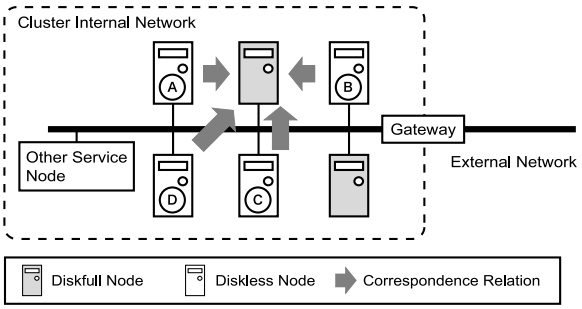

Fig.6　Concentrated composition

検証する．ファイルの write 性能に関しては，クラスタ環境で計算ノードがファイルに書き込みを行う場合は，通常ネットワーク上の共有ファイルサーバに書き込むため，検証は行わない．

検証方法として Fig.5 と Fig.6 の 2 種類の環境を用意し，各ディスクレスノードでベンチマークプログラムを実行させ，測定値に有意な差が生じるかを確かめる．Fig.5 と Fig.6 にあるディスクレスノードのアルファベットはホスト名を表している．

ディスクレスノードのジョブの特徴は，ネットワークマウントしているディスクフルノードから計算に必要なライブラリや実行ファイルを取得するといった，サイズが小さいファイルの read の処理が多い．このことから，ベンチマークプログラムには postmark[5] を使用する．postmark は，主に SMTP サーバ等の能力を測定するために用いられているベンチマークプログラムであり，サイズが小さいファイルを大量に扱う場合のパフォーマンスを測定することが可能である．検証を行った環境を Table 1 に示す．

Fig.5 と Fig.6 の環境において，全てのディスクレスノードで同時に postmark を実行させた．ベンチマークの結果を Fig.7 に示す．横軸はベンチマークを実行したホスト名，縦軸は postmark の出力値である．なお，それぞれの結果は 10 試行の平均値である．

Fig.7 の結果より，均等に分散させた場合の方がそうでない場合と比較して，約 30%性能が高いことがわかった．

4.2　実アプリケーションによるディスクレスノードの性能検証

Table 1　Verification environment

CPU	Pentium III 1GHz(Dual)
Memory	SDRAM 512MByte
HDD (only Diskfull Node)	80GByte 5400rpm
OS	Debian GNU/Linux(3.1)
Kernel Version	2.6.8
Network Interface Card	Intel e100
Switching Hub	Laneed LD-SW08
postmark version	1.51-3
number of postmark operation	5000

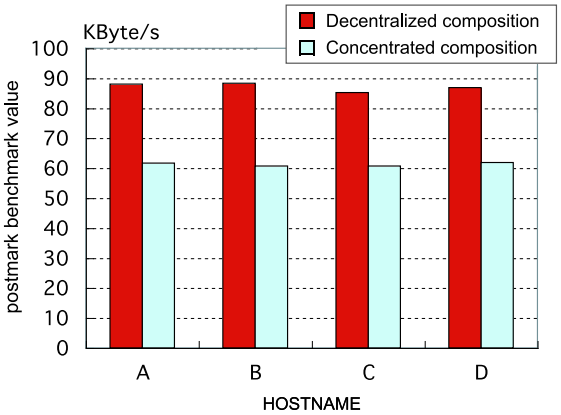

Fig.7　The result of postmark benchmark

本節では，実際にクラスタでよく利用されるアプリケーションを用いて，Fig.5 と Fig.6 のそれぞれ環境における計算速度の違いについて検証する．

クラスタで用いられるようなアプリケーションを大別すると，2 種類に分けることができる．1 つは多くの処理時間を要する大規模計算，もう 1 つは最適化アルゴリズムなどにおいてパラメータを変化させるため，大規模計算と比較して少ない処理時間で終了するが，多くの回数アプリケーションを実行し，かつその実行に必要なファイルを試行毎に読み込む必要のある場合の計算である．

多くの処理時間を要するアプリケーションのベンチマークとして，HPL(High-Performance LINPACK Benchmark)[6] を用いた．HPL は LU 分解による連立一次方程式の解を求めるサブルーチンライブラリであり，世界のスーパーコンピュータのランキングである Top500[7] の性能指標として広く知られている．HPL の出力結果は Flops（Floating point number Operations Per Second）で得られるため，性能検証については Flops の値を用いて比較を行った．また，HPL はパラメータとして行列サイズを指定できるので，いくつかの行列サイズを変化させ HPL の実行を行った．HPL で用いた主要なパラメータと HPL のバージョンを Table 2 に示す．

少ない処理時間の計算を多数行うアプリケーションのベンチマークとして，ga2k[8] を用いた．ga2k は分

Table 2　The parameters of HPL

Matrix size	100, 500, 1000, 2000, 5000
Block size	240
# of processes (P × Q)	2 × 2
Panel factorization	Right-looking
Panel broadcast	1ring
HPL Version	1.20

Table 3　The parameters of ga2k

Population Size	400
Islands	40
Max generations	1000
Crossover rate	1.0
Emigration Rate	0.5
Emigration interval	5
Function	Rastrigin (Dimensions = 10)
ga2k Version	1.4.2

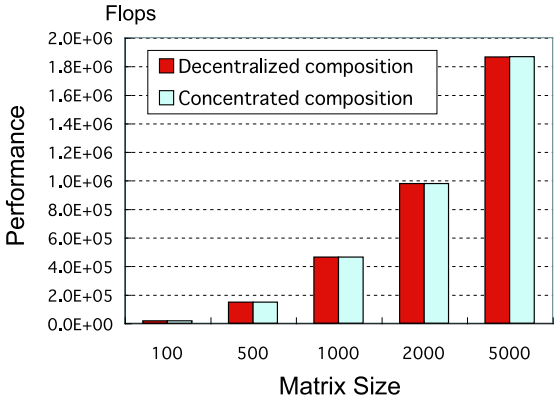

Fig.8　The result of HPL

Table 4　The result of ga2k

Composition	Processing Time (sec)
Concentrated Composition	266.57
Decentralized Composition	281.21

散遺伝的アルゴリズム [9] の実装であり，母集団を複数の計算機に分割して個別に処理させることで，通常の遺伝的アルゴリズムと比較して解精度と解探索速度を向上させることができる．実験では，ga2k を 100 試行実行するのに要した総計算時間の値を用いて比較を行った．ga2k で用いたパラメータと ga2k のバージョンを Table 3 に示す．

HPL と ga2k は共に mpich を用いた並列言語で記述されているので，複数台の計算機を用いた並列処理を行うことができる．HPL と ga2k を Fig.5 と Fig.6 中にあるディスクレスノード 4 台を用いて実行した．HPL の結果を Fig.8 に，ga2k の結果を Table 4 に示す．

Fig.8 より，HPL に関してはどちらの環境においても差は生じなかった．しかし，Table 4 より ga2k に関しては，Fig. 5 の構成の方が Fig. 6 の構成と比較して 5%程度の計算速度向上がみられた．

各ディスクレスノードがそれぞれのアプリケーションを実行するためには，実行ファイルとそれとリンクしている共有ライブラリが必要である．今回の実験では通常のクラスタ環境と同様に，実行ファイルは共有ファイルサーバ（Fig. 5，Fig. 6 中の Other Service Node 内）に置かれている．そのため，Fig. 5 と Fig. 6 の構成における違いは，ディスクレスノードが共有ライブラリを取得するノードが分散しているか，集中しているかである．

読み込む共有ライブラリのファイルの数は，HPL は 6 個，ga2k は 5 個であり，合計ファイルサイズは共に 2MByte 程度であった．そのため，HPL のように多くの計算時間を要する問題に対しては，全計算時間に対する共有ライブラリの取得時間が少ないため，構成の違いによる差は出なかったと考えられる．しかし，ga2k のように実際の処理時間は短いが，多くの試行回数を必要とするアプリケーションに関しては，共有ライブラリの読み込み時間が無視できなくなる．そのため，Fig. 5 の構成の方が計算環境として適していると言える．

4.3　耐故障性の検証　Fig.6 のように 1 台のディスクフルノードに全てのディスクレスノードを振り分けた場合，全ディスクレスノードの故障率はその 1 台のディスクフルノードに依存する．そのため，Fig.5 のようにディスクレスノードを複数のディスクフルノードに均等に振り分けると，ディスクレスノードが同時に動作不能になる可能性が低くなるため，安定したクラスタの管理が行えると考えられる．

それを検証するため，次に述べるシミュレーションを行った．Fig.5 と Fig.6 の環境において，ディスクフルノードの故障率（単位時間にノードが故障する確率）を $a(0 < a < 1)$，ディスクレスノードの故障率を $b(0 < b < 1, a > b)$ とし，10000 単位時間経過させることで，何台の計算ノードが同時に故障するかについて調べた．ただし，故障したノードは，1 単位時間後に復旧するものとした．$a = 0.02, b = 0.01$ の場合の結果を Fig.9 に示す．グラフの横軸は 1 単位時間に同時に動作不能になった計算ノードの数，縦軸は 10000 単位時間中に横軸の項目が動作不能になった回数を表している．

この結果より，半数以上の計算ノードが同時に故障した回数は Fig.6 の構成の方が多く，半数以下の計算ノードが同時に故障した場合は Fig.5 の構成の方が多いことがわかった．

Fig.6 の構成の場合で 5 台の計算ノードが同時に動作不能になった回数が多い原因は，ディスクレスノードが 4 台接続されているディスクフルノードが故障したためである．Fig.5 の場合の構成で 3 台の計算ノードが故障している回数が多い原因は，2 台あるディスクフルノードの内 1 台が故障し，それと関係を持ってい

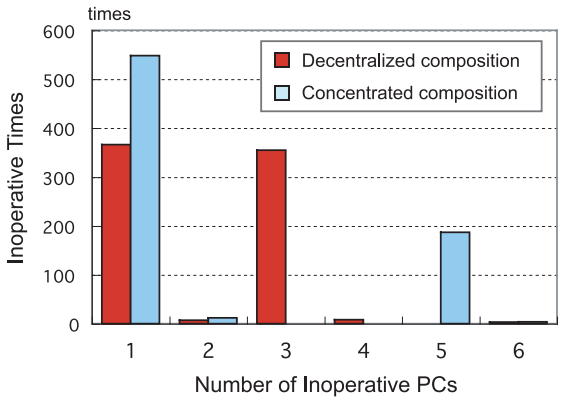

Fig.9　The result of Breakdown simulation

る 2 台のディスクレスノードが動作不能になったからである．

少数の計算ノードが故障した場合はジョブスケジューラなどを用いて動作している別の計算ノードにジョブを割り当てることで対応可能であるが，1 度に大量のノードが動作不能になった場合，ジョブが計算ノードに溜まる可能性が高い．そのため，Fig.5 の方がクラスタ運用上好ましい構成といえる．

4.4　DCAST によるユーザ負担軽減の検討　ディスクレスノードおよびディスクフルノードが混在する環境において，DCAST を用いることでクラスタのセットアップに対する作業者の負担軽減に有効であるかを検証するため，評価実験を行った．

本実験では計算機管理能力が異なる被験者（UNIX 系のサーバ管理経験が 1〜6 年の男女 10 名）に，DCAST，ClusterMatic[10][11] というディスクレスノードのセットアップが行える既存のソフトウェア，手動（Manual）の 3 通りの方法で Fig. 5 に示す構成のクラスタのセットアップを依頼し，作業完了までに要する時間の測定を行った．実験では被験者にマニュアルを配布し，被験者は 1 度それぞれの方法を使ってクラスタのセットアップを経験してから，もう 1 度クラスタのセットアップを行った．本実験では，2 回目のクラスタのセットアップに要する時間を測定した．以下に，各方式におけるディスクレスクラスタの構築手順の概要を示す．

- DCAST
  1. ディスクフルノードの 1 台に OS のインストール
  2. DCAST のインストール
  3. DCAST の設定，および実行
- ClusterMatic
  1. ディスクフルノードの 1 台に OS のインストール
  2. ClusterMatic のインストール
  3. ClusterMatic の設定，および実行（2 ノードのディスクレスノードのセットアップ）
  4. 1 から 3 の作業をもう 1 台のディスクフルノードで行う
- Manual
  1. ディスクフルノードの 1 台に OS のインストール
  2. ディスクレスノードのセットアップに必要なソフトウェアのインストール
  3. ディスクレスノードの設定を行う（2 ノードのディスクレスノードのセットアップ）
  4. 1 から 3 の作業をもう 1 台のディスクフルノードで行う

Table 6　Result of wilcoxon sign rank test

	DCAST and ClusterMatic	DCAST and Manual
critical region $\|z_0\| \geq$	2.33	2.33
Z-value (z_0)	-2.75	-2.75

なお，実験ではディスプレイとキーボードを 2 つずつ用意し，OS のインストール用メディアも 2 つずつ用意した．そのため，例えば，OS のインストールなどの対話作業が少ない手順については，被験者の判断で並列に行ってもよいとした．また，実験では MAC アドレスとノードの対応表を被験者に配布した．その理由は，すべての方式においてユーザはディスクレスノードの MAC アドレスを調べる必要があるが，いずれかの方式で 1 度 MAC アドレスを調べると，その MAC アドレスは他の方式に使用することができるため，各方式に対して平等な時間測定が行えないからである．実験の結果と各方式の平均構築時間，標準偏差，また各被験者のサーバ管理経験年数，ディスクフルクラスタの構築経験の有無を Table. 5 に示す．なお，実験に要する時間は 30 秒単位で計測した．

Table. 5 の結果に対し，Wilcoxon の符号付順位和検定（有意水準 1%の片側検定）[12] を DCAST と ClusterMatic，DCAST と Manual の間でそれぞれ行った．その結果を Table. 6 に示す．Table. 6 より，DCAST は他の手法に比べ，共に有意に短時間でクラスタのセットアップを行えることがわかった．また，各手法の標準偏差の値は DCAST が最も小さいので，DCAST は他の手法に比べ被験者の計算機管理能力に関わらず，クラスタのセットアップを短時間で行えると言える．

また，作業終了後にアンケートを実施し，どの方法が最も負担が少なく，また簡易にクラスタのセットアップが行えたかを尋ねた．その結果，すべての被験者が DCAST が最も負担が少なく，また最も簡易にクラスタのセットアップを行えたという結果になった．さらに DCAST 以外の方法を用いた場合において，どの作業

Table 5　Setting time for Cluster System

Examinee No.	1	2	3	4	5	6	7	8	9	10	Ave.	S.D.
Setup time by DCAST(min)	29.0	33.5	36.5	38.0	29.0	35.0	29.5	31.0	32.0	31.5	32.6	3.0
Setup time by ClusterMatic(min)	34.0	49.0	53.0	62.0	35.0	72.0	49.0	64.5	52.0	70.0	54.1	12.5
Setup time by Manual(min)	63.5	60.0	77.5	99.0	77.0	117.0	91.0	96.5	110.5	88.0	88.0	17.8
Experience of server management(year)	6	3	2	2	1	1	1	1	1	1	-	-
Experience of diskless cluster setup	○	×	×	×	×	×	×	×	×	×	-	-

に負担が大きかったかについてアンケートを実施した．その結果，OS を複数回インストールする必要がある，同じ作業の繰り返しを強いられる，設定ファイルに単純なスペルミスがあった，という回答を得た．DCAST はそれらの作業はすべて自動化されている．以上の結果より，DCAST を用いることで，クラスタのセットアップにおける作業者の負担が少なくなったと言える．

5. 関連研究

クラスタをセットアップするソフトウェアの開発は多数行われている．しかし，DCAST が実現しているようなディスクレスノードとディスクフルノードが混在している環境を自動的にセットアップできるツールはない．

4.4 節で用いた ClusterMatic は，Los Alamos National Laboratory で開発されたディスクレスノードをセットアップできるツールである．DCAST との大きな違いは，DCAST は NFS を用いてディスクレスノードにルートファイルシステムを提供しているのに対し，ClusterMatic はサーバ計算機が用意した単一のブートイメージのみを用いてディスクレスノードは動作する．そのため，ディスクフルノードが 1 台の場合は，DCAST よりも高速にディスクレスノードのセットアップを行うことができる．ただし，ディスクレスノードで用いることのできるアプリケーションは限られており，ディスクレスノードのソフトウェア構成の変更は基本的にできない．また，ディスクレスノードの起動メディアとして CD-ROM を採用しているため，計算ノードの台数分 CD-ROM が必要である．DCAST は PXE ブートに対応しているため，起動メディアは必要ない．

Rocks[13] は OS とクラスタ用ソフトウェアがセットになった起動メディアを用いてクラスタ環境のセットアップを行うツールであり，DCAST と同様に PXE ブートを用いて計算ノードのセットアップが可能である．また，Rocks は，Roll という拡張パッケージを用意しており，ユーザが任意に Roll を導入することで，ジョブスケジューラやバイオインフォマティクスツールのインストールを簡易に行う事が可能である．ただし，Rocks はディスクレスノードのセットアップを行うことはできない．

SCore[14] は DCAST と同様にクラスタのセットアップ直後に並列計算ライブラリなどの設定がなされているが，ソフトウェアの追加や変更を行うことができない．ただし，SCore では各計算ノードが起動時に NFS マウントするサーバの指定を行えるので，利用したいソフトウェアをインストールした NFS サーバを用意することで，そのソフトウェアを利用することが可能になる．

Luice[15] は Lucie メタパッケージという仕組みを有しており，サーバ・計算ノードにインストールするソフトウェアの設定をユーザが任意に変更できる．また，Dolly+[16] を利用した高速ファイル転送機能によって，クラスタイメージのコピーを高速に行える．ただし，Lucie はディスクレスノードのセットアップを行うことはできない．

ParallelKnoppix[17] は Live CD であり，CD-ROM ドライブに CD をいれて起動するのみでディスクレスノードとして動作させることが可能である．その特徴として，起動直後に様々な並列計算用のソフトウェアが利用可能である点，HDD のスワップ領域を自動認識して利用する点が挙げられる．ただし，Live CD にはソフトウェアの変更の手間が大きいという問題点がある．DCAST クラスタでソフトウェアの追加を行いたい場合は，DCAST マスタノードにソフトウェアを追加し，DCAST の再実行を行うのみで対応可能である．

6. まとめと今後の課題

我々はディスクレスノードとディスクフルノードが混在するクラスタ環境のセットアップを行うツール，DCAST の開発を行った．本稿では，クラスタのセットアップ時にディスクレスノードが均等にディスクフルノードとネットワークマウントの関係を設計することで，ディスクレスノードのファイルの read 性能が高くなり，かつ短時間で終了する計算を多数行う場合の総処理時間が短縮することを示した．また，同時に多くのディスクレスノードが動作不能にならないことも示した．さらにディスクフルノードとディスクレスノードが混在する環境をセットアップする場合，DCAST は他の既存のソフトウェアと比較して短時間かつ作業者の負担が少ない操作でセットアップできることを示した．

DCAST の問題点の 1 つに，ディスクフルノードの数が多くなるほど，クラスタイメージの配布に要する時間が増大するということが挙げられる．この問題点を解決するために，Dolly+などを利用した高速ファイル転送や，マルチキャスト通信に対応させる必要がある．

今後 DCAST に追加する機能として，クラスタ稼働

中にディスクフルノードが故障した場合，その故障を検出し，自動的にディスクフルノードとディスクレスノードとの関係を再設計する機能があれば有用であると考えられる．また，計算ノードにインストールするクラスタイメージは，DCAST マスタノードをもとに作られるため，ヘテロな構成のクラスタには対応していない．ヘテロな構成のクラスタに対応させるためには，DCAST マスタノード上で複数のルートファイルイメージを持たせ，計算ノードの種類によって最適なクラスタイメージを配布するなどの工夫が必要である．

7. 謝辞

DCAST の機能の一部は，独立行政法人情報処理推進機構未踏ソフトウェア創造事業からの援助を受けて開発を行いました．関係各位に感謝いたします．

(1) Preboot Execution Environment (PXE) Specification. http://www.pix.net/software/pxeboot/archive/pxespec.pdf

(2) 大熊 俊明，鱸 洋一，小島 義孝，田子 精男，企業内パソコンを活用した分散計算と土留め変形解析への適用，日本計算工学会論文集，2006，Paper No 20060011

(3) 武田 和大，小野 智司，中山 茂，異機種混合並列計算ミドルウェア JSGrid の開発と評価，日本計算工学会論文集，2006，Paper No.20060005

(4) Debian GNU/Linux. http://www.debian.org

(5) Postmark. http://www.netapp.com/tech_library/postmark.html

(6) HPL A Portable Implementation of the High-Performance Linpack Benchmark for Distributed Memory Computers. http://www.netlib.org/benchmark/hpl/

(7) TOP500 Supercomputer Sites, http://www.top500.-org

(8) Hiroyasu Tomoyuki, Mitsunori Miki, PDGA http://mikilab.doshisha.ac.jp/dia/research/pdga/archive/index.html,2002

(9) Reiko Tanese. Distributed Genetic Algorithms. *Proc. 3rd International Conference on Genetic Algorithms*, pp. 434 – 439, 1989

(10) CLUSTERMATIC, Redesigning the Cluster Architecture, A project of the Cluster Research Lab in the Advanced Computing Laboratory at Los Alamos National Laboratory, www.clustermatic.org

(11) 吉瀬謙二, 片桐孝洋, 本多弘樹, 弓場敏嗣, Clustermatic を用いた PC クラスタの構築, Technical Report UEC-IS-2004-3, Version 2004-05-17

(12) 柳川堯，新統計学シリーズ 9 ノンパラメトリック法, 培風館，1982.

(13) Rocks Clusters. http://www.rocksclusters.org

(14) PC Cluster Consortium, http://www.pccluster.org

(15) 高宮安仁, 真鍋篤, 白砂哲, 松岡聡. 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. SWoPP 湯布院 2002, pp. 131-136, 2002

(16) Dolly+ home page. http://corvus.kek.jp/~manabe/pcf/dolly/

(17) ParallelKnoppix. http://idea.uab.es/mcreel/ParallelKnoppix/