成蹊大学理工学研究報告
J. Fac. Sci. Tech., Seikei Univ.
Vol.48 No.1 (2011) pp.23-30

WAN 接続クラスタ群をメモリ資源とする大容量メモリ提供システム

鈴木　悠一郎*1，緑川　博子*2

A Virtual Large Memory System on WAN-connected Clusters

Yuichiro SUZUKI*1, Hiroko MIDORIKAWA*2

ABSTRACT : Emerging 64bitOS drastically enlarges an available memory address space and opens the door to new applications using very large data. In this background, Distributed Large Memory System: DLM was designed for sequential applications requiring a large amount of memory. It gives us very large virtual memory using remote memories distributed over nodes in a cluster. This paper proposes newly designed WAN-based DLM, which is an extension of the LAN-based DLM. It provides an automatic cluster nodes allocation for suitable calculation/memory servers for user requests, as well as easy job entry interface via portal site.

Keywords : Cluster Computing, Large Memory, Remote Memory, Distributed Memory, WAN-Connected Clusters

（Received April 6, 2011）

１．はじめに

64bitOS の普及により大容量アドレス空間が利用可能となってきている。現実装の 48bit でも 256TB のアドレス空間が利用可能である。しかし，1 台に搭載できる物理メモリにはコストや，物理的な制限がある。通常，OS は物理メモリを超えるサイズのデータを扱う場合に，ローカルハードディスク上のスワップ領域にページ単位でスワップすることで，物理メモリサイズを超えた仮想メモリ空間を実現する機構を搭載している。しかし近年，ローカルハードディスクを越える通信性能を持つネットワークが出現しはじめたことから，ハードディスクではなく，クラスタ上の遠隔マシンのメモリを仮想メモリ空間の一部として利用する遠隔メモリページングの研究がされるようになってきた。しかし，ローカル物理メモリサイズを超えるような大規模データを扱うプログラムの実行に，ハードディスクのスワップ領域を利用することは非常に低性能なため，実用的ではない。このため，多くの場合,逐次プログラムを MPI などの並列プログラムに変換し，クラスタの複数のノードへデータを分割して処理を行っている。

しかし，これを行うには，逐次プログラムとして設計されたプログラムもすべて並列プログラムに変換する必要があり，多くの科学技術シミュレーションを行う科学者にとって，その開発，デバッグ，動作の正当性検証などは，煩雑であるばかりか，本来，シミュレーション対象のモデル化や解析に費やすべき時間を大幅に損失することにもなっている。また，扱うシミュレーションモデルが複雑で並列化が困難な場合や，他人が作成した既存プログラムやライブラリを利用したプログラムでは，並列化自体が困難な場合も存在する。このため，並列処理実行による速度向上がなくても，逐次プログラムのまま実行ができるシステムが必要とされている。

そこで我々は，C 言語で記述された逐次プログラム向けの遠隔メモリページングソフトウェアである分散大容量メモリシステム(DLM : Distributed Large Memory 以降 DLM)を開発してきた[1][2]。DLM は，ユーザによる並列化のための再設計を必要とせず，逐次プログラムに最小限の変更を加えるだけで，ユーザには見えない形で，クラスタの遠隔ノードのメモリへページスワップ処理を行い，ローカル物理メモリサイズを超える大規模データを扱うプログラムの実行を実現する。

我々は，WAN で接続したクラスタ群の中から，メモリ資源として適切なクラスタ，メモリサーバノード，計

*1：理工学研究科理工学専攻情報科学コース修士学生
*2：情報科学科 助教（midori@st.seikei.ac.jp）

算ノードを自動選定して DLM を利用することが可能なクラスタ自動選定システムを構築した[5]。また，直接的なクラスタ接続環境がないユーザのパソコンからも DLM を使用したプログラムの実行ができるような，ポータルサイトを提供する Web サーバも構築している[5]。

本報告は，WAN 接続クラスタ群の中からユーザのメモリ要求を満たし，かつできるだけ負荷が少ないクラスタやノードを自動選択して，ユーザが大容量メモリを使う逐次プログラムを容易に実行できる環境を実現したので，報告する。

２．分散大容量メモリシステム DLM

DLM は遠隔メモリページングシステムを OS とは独立のユーザレベルソフトウェアとして構築している。このことから，OS カーネルの変更を必要とせず，可搬性が高く，汎用オープンクラスタなど，ユーザの権限に制限がある環境でも利用が可能となっている。

DLM のシステム構成は，図 1 のように計算ノードとメモリサーバノードの 2 つに分かれている。計算ノードでは，2 つのスレッドが動いており，1 つはユーザプログラムを実行する計算スレッド，もう 1 つは他のメモリサーバとの通信をする通信スレッドがある。メモリサーバノードでは，計算ノードのローカルメモリに入りきらないデータをページ単位で格納し，必要に応じて計算ノードに提供するメモリサーバプロセスが動作している。

計算ノードとメモリサーバノードの間では，OS とは別の DLM 独自のページ単位でスワップが行われており，ページサイズは DLM で独自に設定されている。

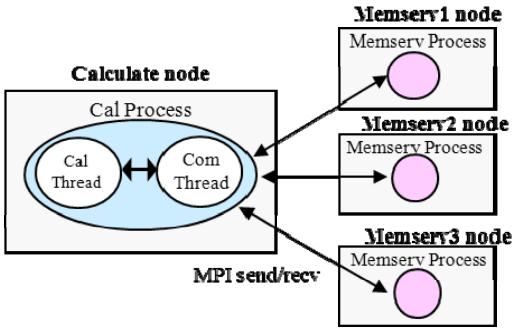

図 1　DLM システム構成

ユーザは，逐次コードに図 2 の①～⑤のような追加を行う。①は，DLM 関数を利用するための DLM のヘッダーであり，②は DLM の初期化関数である。①の dlm_startup 関数が呼ばれると，図 1 のように，遠隔のメモリサーバノードにメモリサーバプロセスを生成し，計算プロセスとの通信を確立する。すなわち，ユーザには逐次プログラムの実行として見えるが，実際には，計算プロセスと複数のメモリサーバプロセスの並列プログラムとして DLM は実行されている。⑤dlm_shutdown は DLM の終了関数で，プロセス間の通信を閉じ，すべてのプロセスを終了させる。ローカルメモリが不足する場合に遠隔メモリへデータを展開したい大容量のメモリを確保・解放する際には，malloc，free と同様な関数③dlm_alloc，④dlm_free を用いて，メモリの動的割り当てと解放を行うことができる。ユーザは，逐次プログラムにこれらの関数呼び出しを加えるだけで，並列プログラムであることを意識することなくクラスタノードに分散したメモリを仮想的な大容量メモリとして使うことができる。

図 2 下部に，通信に MPI を用いる DLM システムの場合のコンパイル例と実行コマンド例を示す。この例では，DLM が MPI プログラムとして実行されるので，mpicc と mpirun を用いている。実行時に指定している memfile は，メモリサーバノード名と各メモリサーバで DLM が利用可能なメモリ量を指定するためのファイルである。この例では図 1 に対応しており，メモリサーバプロセス 2 個と計算プロセス 1 個の 3 プロセスの並列プログラムとして実行される。

また，DLM コンパイラ[6]を利用することにより，静的な配列宣言についても，ユーザはソースプログラムの簡単な変更だけで DLM を使用することもできる。

記述例 prg.c

```
#include <stdio.h>
#include <dlm.h>                                  ①
#define N 1<<30      // 1G(*4B=4GB)
int main(int argc,char *argv[]){
  int *a;
  /* DLMの初期化 */
  dlm_startup(&argc,&argv);                       ②
  /* DLMの動的メモリ割り当て */
  a = (int *)dlm_alloc(sizeof(int)*N);            ③

  逐次コード

  /* DLMで確保したメモリの解放*/
  dlm_free(array);                                ④
  /* DLMの終了 */
  dlm_shutdown();                                 ⑤
  return 0;
}
```

コンパイル例

```
gcc prg.c –l dlmmpi
```

実行コマンド例

```
mpirun -np 4  prg -- -f memfile
```

memfile例

```
# ホスト名     メモリサイズ
calhost        2048   //2GB
memhost1       2048   //2GB
memhost2       4096   //4GB
```

図 2　DLM プログラム例

2. 1　マルチクライアント向け DLM(DLM-M)

DLM には 2 つのタイプがあり，図 1 のような構成となっている DLM-S と，図 3 のような構成となる DLM-M[3][4]がある。DLM-S は，シングルクライアント向けとなっていて，DLM-M はマルチクライアント向けとなっている。

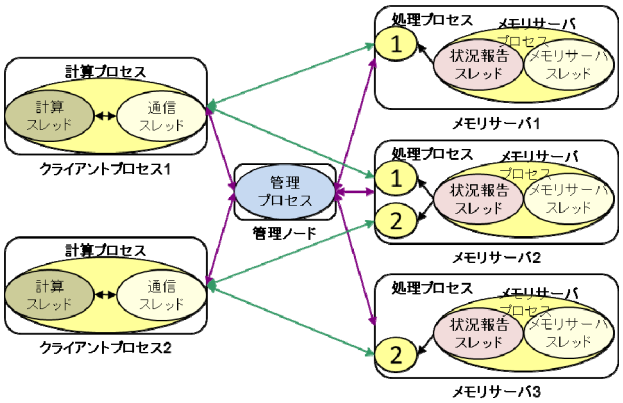

図 3　DLM-M System

DLM-M は，ユーザのクライアントプロセスが動作する計算ノードと，メモリを提供するメモリサーバ，メモリサーバをユーザへ割り振る管理プロセスからなる。システムの管理者は，常駐型となっているメモリサーバプロセスと管理プロセスを事前に立ち上げておき，ユーザのクライアントプロセスは管理プロセスへアクセスすることによりメモリサーバが割り振られる。図 4 に管理者が DLM-M を起動する際のコマンド例と，メモリサーバノードの利用するメモリ量の設定ファイル例 dlmm_admin.conf を示す。この際，管理者はメモリサーバノードごとに DLM が利用できるメモリ量の上限を設定することができる。

DLM-M起動コマンド

```
$dlmm_admin dlmm_admin.conf
```

dlmm_admin.conf例

```
memhost0   24576   // 24GB   メモリサーバ0
memhost1   24576   // 24GB   メモリサーバ1
memhost2   20480   // 20GB   メモリサーバ2
```

図 4　管理者コマンド例

ユーザは，図 5 のように DLM-S のメモリサーバの指定の代わりに，管理プロセスが稼働している管理プロセスノードを指定する。指定方法は，--のあとがオプションとなっていて，-s の後に管理プロセスがあるホスト名，-m でプログラム全体で利用するメモリ量，-l で計算ノードで利用するローカルメモリ量をそれぞれ指定する。また，メモリサーバの提供の際には管理プロセスは，一つのメモリサーバへの負荷が集中することがないように負荷分散を行う。負荷分散の方法は，クライアントからの問い合わせに対し，各メモリサーバが現在サービスしているクライアント数を考慮して負荷が分散するようにメモリサーバを割り当てる。1 つのクライアントプログラムが複数のメモリサーバを用いている場合には，それぞれのメモリサーバのクライアント数を 1 とするのではなく，そのクライアントプログラムが使用する全メモリサイズに対する，各メモリサーバでそのクライアントプログラムが利用するメモリサイズの比を乗じた重み付けクライアント数を用いる。一つのメモリサーバの負荷計算には，このメモリサーバを利用している複数のクライアントのそれぞれの重み付けクライアント数の総和が用いられる。

ユーザプログラム実行コマンド例

```
$user_program ･･･s admin_host -m 10000 -l 4000
```

図 5　DLM-M ユーザプログラム実行コマンド例

3. 大容量メモリ提供システム DLM-WAN

大容量メモリ提供システム DLM-WAN システムは，従来の DLM-M を新たに WAN 接続クラスタ群環境へ拡張したもので，「クラスタ自動選定システム」とユーザのジョブ投入インターフェースを担う「web サーバ」の 2 つの部分から構成される。

クラスタ自動選定システムでは，従来の DLM-M に対応するシステムを各クラスタ内に置き，新しく WAN 全体を管理してクラスタやノードの自動選定を行う「WAN 管理プロセス」を導入した。

さらに，web サーバがポータルサイトを提供しており，ユーザは，どこからでもポータルサイトを通してジョブ投入を行える。これにより，WAN 接続されたクラスタ群のメモリ資源を利用して，大容量のメモリ空間を使用する逐次プログラムを容易に実行することができる。

システムの全体を，図 6 に示す。右側がクラスタ自動選定システム，左側がユーザへのポータルサイトを示す。図中，Cluster1 にある W は WAN クラスタ群全体を管理するノードを示し，クラスタ自動選定システムの WAN 管理プロセスが動作している。すべてのクラスタには，図 3 の DLM-M システムの管理プロセスを本システム用に拡張した「LAN 管理プロセス」が動作する

ノードAが1つあり，さらにそれぞれ，複数の計算ノードCとメモリサーバノードMを有している。一方，Webサーバは，自動選定システムのWAN管理プロセスと通信を行い，ユーザからのジョブの投入処理等を行っている。

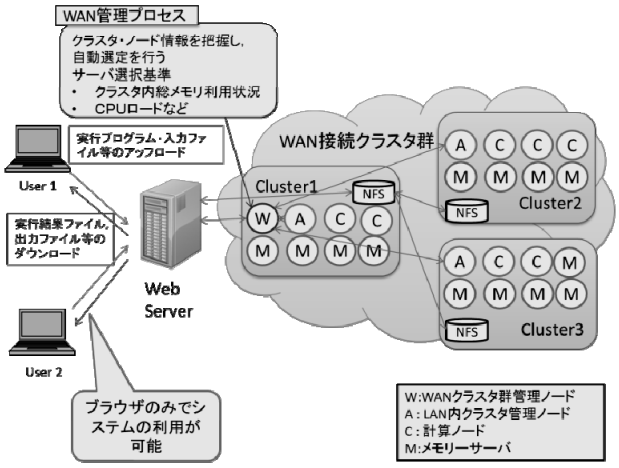

図6　システム全体図

4．クラスタ自動選定システム

4．1　システム概要

ユーザは，クラスタ自動選定システムを使用することにより，複数のクラスタから負荷が少ないクラスタを自分で探す必要がなくなる。

クラスタ自動選定システムは，WAN接続クラスタ群で，大容量のメモリ空間を必要とする逐次プログラムを実行するにあたって，メモリ資源として条件の良いクラスタとその中のノードを自動選定する。

WAN接続クラスタ群環境でのクラスタ自動選定システムを図7に示す。これは，3クラスタが接続されている例であるが，クラスタ内の詳細なプロセス構成は，2クラスタについては省略している。

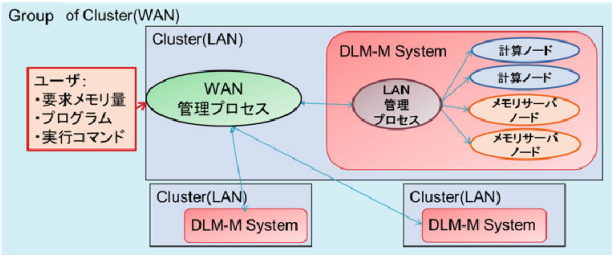

図7　クラスタ自動選定システム

WAN管理プロセスの起動には，図8のようにLAN管理プロセスがあるホスト名をリストにしたファイルhost.confを指定する。WAN管理プロセスは，ユーザからのジョブ投入があった時，このリストにあるホストで動く各クラスタのLAN管理プロセスに問い合わせ，担当するクラスタ内のメモリサーバノードや計算ノードの情報を取得する。

WAN管理プロセス起動コマンド

```
$ WANadmin host.conf
```

host.conf設定例

```
hongo200.logos.ic.i.u-tokyo.ac.jp
hosei000.intrigger.cis.k.hosei.ac.jp
huscs000.iic.hokudai.ac.jp
```

図8　WAN管理プロセス起動例

一方，LAN管理プロセスは，DLM-Mの管理プロセスが拡張されており，メモリサーバプロセスから定期的に送られてくるメモリ使用状況, CPU負荷状況だけでなく，必要に応じて計算ノードのメモリ，CPU負荷状況も取得できるようになっている。

LAN管理プロセスの起動時には，図9に示すようなメモリサーバノード，計算ノードのリストとそれぞれにおけるDLM利用可能メモリ量の上限値を示すファイルを指定する。すなわち，図4のDLM−M起動コマンドにおけるdlm-admin.confのファイルとして，新たに，計算ノードの情報も含めた図9のファイルを指定する。

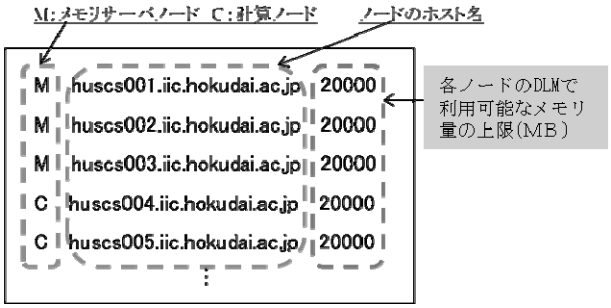

図9　LAN管理プロセス設定ファイル例

また，計算ノードとメモリサーバノードのメモリ使用状況とCPU負荷情報としては，Linuxが提供している以下の情報を用いている。

- CPU負荷状況：topコマンドのCPU load average
- メモリ使用状況：/proc/meminfo内のMemFreeとInactiveの値

MemFree は，ノード内の利用されていないメモリ量であり，Inactive は，ノード内の解放予定のメモリ量である。よって，この値の合算値が，実際の「利用可能メモリ量」となっている。

WAN 管理プロセスは各 LAN 管理プロセスから情報を取得し，その情報をもとに自動選定を行った後，ユーザのプログラムや入力ファイルなどを，選定先クラスタ内の選定計算ノードへ転送し，ユーザからの実行コマンド通りにプログラムを遠隔実行する。この DLM−WAN システムをコマンドベースで利用する場合のユーザコマンド例を図 10 に示す。--以降がオプションとなっていて，例では，-w の後に WAN 管理プロセスがあるホスト名，-m の後にユーザが要求するメモリ量(MB)，-u の後にユーザ名を入力するようになっている。

```
$DLM-WAN user_program -- -w WANadmin_host
                         -m 10000
                         -u username
```

図 10　自動選定システムを使用するユーザコマンド例

4. 2　稼働環境

大容量メモリ提供システムの対象環境は，DLM 以外の他のユーザのアプリケーションが同時に動作しており，各クラスタ，各ノードの状態は常に変動し，利用できる資源の量は変化するものである。

本システムは，以下の環境を持つ WAN で接続されたクラスタ群で利用できる。

A) クラスタ間でのユーザアカウントは同一。
B) 同一クラスタ内のノードはホームディレクトリが共通。
C) クラスタ内の各ノードはグローバル IP を持ち，クラスタ間で各ノードへの遠隔アクセスが可能。

この条件を満たす実行環境として，本報告では，多くの大学などのクラスタを結合した分散コンピューティングシステムである InTrigger(図 11)[7]を用いてシステムの構築を行った。InTrigger は，全国 17 拠点 23 クラスタ(2011 年 3 月現在)が WAN で接続されており，システムの条件を満たし，複数のユーザが同時にインタラクティブな利用が可能となっている。

4. 3　自動選定ポリシー

クラスタと計算ノード，メモリサーバノードの選定基準は以下のとおり行う。これは，計算ノード内の利用可能メモリ量が多いところを優先的に割り当てるようにしている。遠隔メモリページングの特性上，ユーザプログラム実行時に計算ノード内のメモリ量をなるべく多く使えるようにするほうが，遠隔ノードへスワップする頻度が下がり，高速に処理できるためである[1]。これをローカルメモリ率優先モードと呼ぶ。ローカルメモリ率とは，ユーザプログラムが利用する全体のメモリ量に対する計算ノードで使用するローカルメモリの割合である。残りの不足分はメモリサーバの遠隔メモリを利用する。

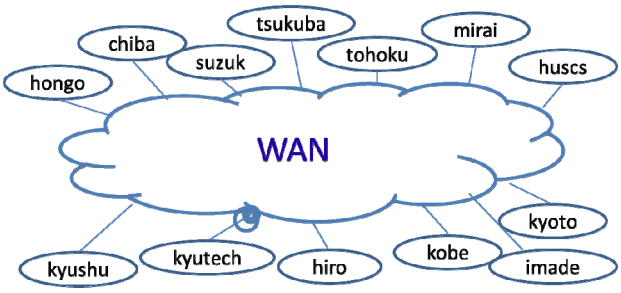

図 11　InTrigger 概念図

クラスタの選択ポリシーは以下の通りである。

A) ユーザプログラムを，ローカルメモリのみで実行できる計算ノードを持つクラスタを優先的に選択する。すなわち，ユーザの要求メモリ量より，ノードの利用可能メモリ量(MemFree)が多く，CPU の load が低いクラスタを選択する。MemFree のみで条件を満たすノードがない場合は，解放される予定のメモリ量(Inactive)との合算値で選択する。
B) 計算ノードのローカルメモリ量で要求メモリ量を満たせない場合は，遠隔ノードメモリを利用する DLM-M システムでプログラムの実行を行う。1 クラスタ内の全メモリサーバの MemFree と Inactive の合算値(利用可能メモリ量)も加え，ユーザのメモリ要求量を満たすクラスタのうち，利用可能メモリ量が低いクラスタから選択する。

また，このほかにクライアント数均一化モードも設定可能である。これは，1 つのクラスタに複数のクライアントプログラムが集中しないようにするモードである。DLM では，同一クラスタ内のクライアント数が増えるほど，遠隔メモリアクセス時のクラスタ内通信が増加する。そこでジョブをクラスタ間で分散し，通信負荷を軽減するモードを提供している。

この選定ポリシーでは，クラスタ間の担当クライアント数が同じ場合，ローカルメモリ率優先モードのポリシーで選定し，同じでない場合は，クライアント数が少ないクラスタから割り振る。

５．稼動実験

5．1　実験環境

実験環境は，InTrigger 内の 3 クラスタ(hongo，hosei，huscs)を使用した。この 3 クラスタはどれも，クラスタ内の各ノードが 10Gbit Ethernet の高速ネットワークで繋がれている。

実験は，自動選定システムを通して 8 つのジョブの投入を前述した 2 つのモードで行った。

各クラスタは，計算ノードを 4 ノードに設定してある。また，メモリサーバは各クラスタで以下のように設定した。

- hongo：9 ノード(物理メモリ合計 260GB)
- hosei：7 ノード(物理メモリ合計 140GB)
- huscs：5 ノード(物理メモリ合計 100GB)

ジョブは，今回使用するクラスタの計算ノードにおいて，メモリの使用状況によって，ノード内の利用可能メモリ量で足りる場合と足りない場合がでる値として，15GB を必要とする姫野ベンチマーク[8]の EXLARGE サイズを使用した。

5．2　実験結果

ローカルメモリ率優先モードでの実験の結果の一例を図 12 に示す。○の中の数字は job が割りつけられた順番である。この結果から自動選定ポリシーどおり割り振られたことが確認できた。この例では，huscs クラスタへのジョブの割りつけがされなかった。これは，huscs クラスタ内のノードで DLM 以外のアプリケーションが実行されていたことにより，ローカル(計算ノード)での DLM 利用可能メモリ量が少なくなっており，CPU 負荷も高くなっていたため，DLM が利用可能なメモリ量が少なくなり，計算ノードとして選択されなかった。

次に，クライアント数均一化モードでの実験結果の一例を図 13 に示す。クラスタの状態は実行結果 1 と同時期に行い，ほぼ同じ状況で行われた。やはり hongo クラスタ・hosei クラスタに比べて huscs クラスタが混んでいたため，hongo，hosei クラスタに優先的に割り当てられている。しかし，huscs クラスタにも均一にクライアントが割り当てられているのが分かる。

各ジョブの実行時間の比較も，2 つのモードで行った。その結果，混んでいた huscs クラスタの実行時間が長くなったため，平均実行時間は huscs クラスタを使用していない実行結果 1 のほうが速くなった。今回のようにクラスタ間の負荷が著しく不均衡な場合は，ローカルメモリ率優先モードで実行したほうが良い結果が出た。

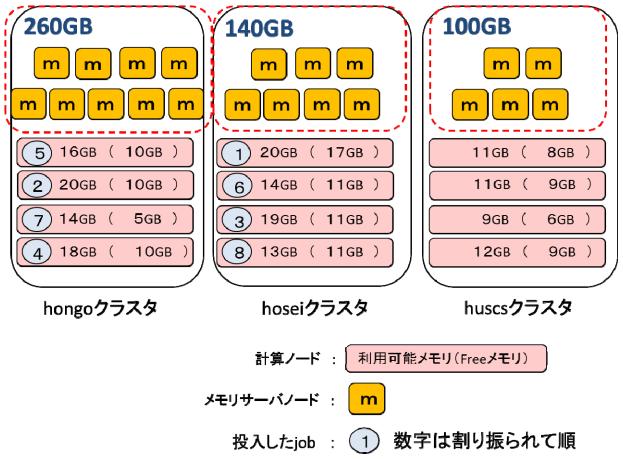

図 12　ローカルメモリ率優先モードにおける実行例

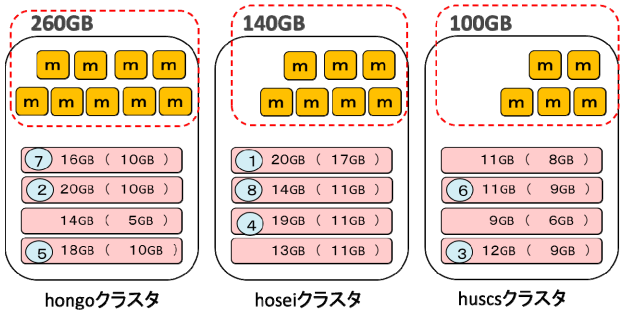

図 13　クライアント数均一化モードにおける実行例

６．Web インターフェース

Web サーバを介して自動選定システムを使用できる Web インターフェースを構築した。以下に，インターフェースの説明をする。

自動選定システムには，自動でクラスタとノードを選定する Autorun モードと，現在のクラスタとノードの状況を表示させ，ユーザがクラスタとノードを選択する Interactive モードがある。

図 14 は，Autorun モードのプログラム等のアップロード，ユーザの設定，実行のページである。このページでは，ユーザは，プログラムの実行に必要なファイルをアップロードし，実行に必要となるステータス(必要メモリ量，実行コマンドライン，必要に応じて再コンパイル指定と入出力ファイル等の指定)を設定，実行ボタンをクリックすることによって，ジョブが実行される。

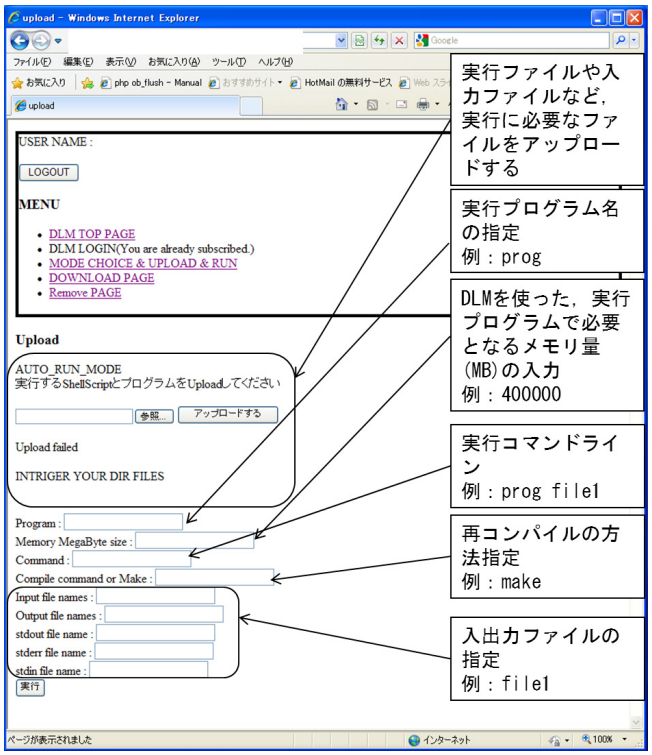

図 14　アップロードとジョブ実行用ページ

図 15　Interactive モードにおける情報表示ページ

図 15 は，自動選定を使わない Interactive モードのページである。ユーザは，表示されるクラスタ内のノード情報(メモリ利用状況や CPU 負荷など)から，システムを実行するクラスタと計算ノードを指定することができる。この図では，2 クラスタ，計算ノード合計 3 ノードの図となっているが，クラスタ，計算ノードが増えると，大量の情報が表示されることとなり，その中からクラスタとノードを選ぶことになる。Autorun モードを使うと，このような煩雑な操作をすることなく実行が可能となっている。

図 16 は，ダウンロードページとなっていて，ユーザのディレクトリ内のファイル名が表示され，ファイル名を指定することによってダウンロードできる。これにより，ユーザは，実行した際の出力結果ファイル等をダウンロードすることにより，プログラムの実行結果を知ることができる。

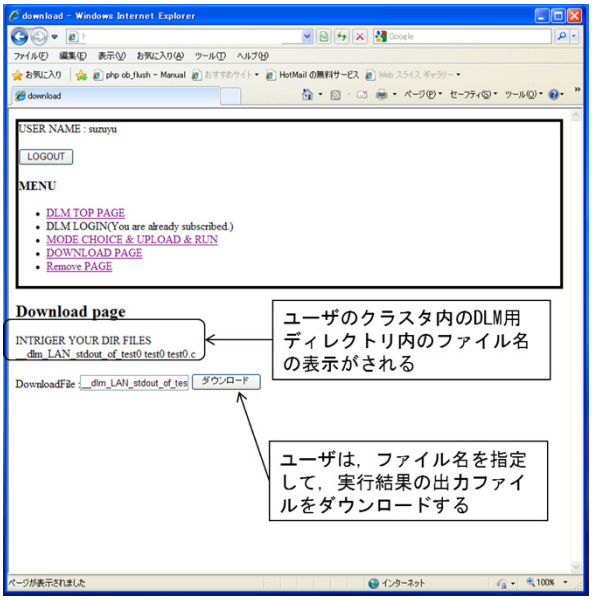

図 16　ダウンロードページ

7. まとめ

本システムにより，ユーザは大容量メモリを必要とする逐次プログラムを，WAN 接続クラスタ群環境での各クラスタ内の資源の変化を意識せずに，CPU 負荷が低く，メモリの豊富なクラスタで実行が可能となった。さらに，ポータルサイトへアクセスすることにより，クラスタへの直接な接続環境がない場合でも，どこからでも利用が可能となった。

今後の課題としては以下のことがあげられる。選択モードの追加や改良，インターフェースの改善，バッチキュー方式の実装などである。さらに，計算ノードとメモ

リサーバノードは起動時に固定となっているが，計算ノードとメモリサーバノードの追加や削除なども，動的に行えるような環境の構築や，使用状況によっては計算ノードをメモリサーバノードに，またはその逆のように役割の変更を行えるような環境の構築も考えている。

参考文献

1) 緑川博子，齋藤和広，佐藤三久，朴 泰祐：“クラスタをメモリ資源として利用するための MPI による高速大容量メモリ”，情報処理学会論文誌，コンピューティングシステム，Vol.2, No.4, pp.15-36, (2009.12)
2) H. Midorikawa, K.Saito, M.Sato, T.Boku : “Using a Cluster as a Memory Resource : A Fast and Large Virtual Memory on MPI”, Proc. of IEEE cluster2009, 2009-09, Page(s): 1-10 (Digital Object Identifier 10.1109/CLUSTR. 2009. 5289180)
3) 齋藤和広，緑川博子，甲斐宗徳：“マルチクライアント向け分散型大容量メモリシステム DLM-M の設計と実装”，FIT2008 論文集，C－003，pp.199-200，（2008.9）
4) 三浦　望，緑川博子，甲斐宗徳：“クラスタをメモリ資源として利用するための動的メモリ提供システムの提案”，情報科学技術フォーラム FIT2009，FIT 論文集,B-029, pp.421-422，（2009.9）
5) 鈴木悠一郎，緑川博子：“WAN 接続クラスタをメモリ資源として利用するためのメモリサーバ自動選定システム”，情報処理学会 第 73 回全国大会，論文集，3A－7 （2011.3）
6) 吉村　礎，緑川博子，甲斐宗徳：“ローカルメモリを越える大容量データを扱う逐次処理のための C コンパイラ”，情報科学技術フォーラム FIT2010，FIT 論文集,B-026 , pp.335 -336，（2010.9）
7) 田浦健次郎：“InTrigger：オープンな情報処理・システム研究プラットフォーム”，情報処理 Vol.49 No.8, pp939-944, (2009.8)
8) Himeno Benchmark web site [Online] , http://accc.riken.jp/HPC/HimenoBMT.html (2011)