小山工業高等専門学校研究紀要
第 4 0 号 (2008) 81 − 90

MPIとSGEによる並列分散処理システムの構築

Parallel and Distributed System using MPI and SGE

南斉 清巳・山口 晃
Kiyomi NANSAI, Akira YAMAGUCHI

1．はじめに

グリッド・コンピューティングは分散処理の一つであり、多数のコンピュータを結びつけて利用することで、ひとつのコンピュータシステムとして利用できるようにするというものである。これにより、一台の場合より高い処理能力を有することが出来る。また、性能の低いPCのみで構築されたグリッドであっても、性能の高いPCに匹敵する効率をあげることも不可能ではない。これを実現するツールとして知られているのが、Sun Grid Engine(以下SGE)である。このツールを用いることでグリッドの構築が容易となり、また、独自の機能であるスケジューリング機能を用いれば、グリッドの資源を効率的に利用できるようになる。

しかし、SGEはグリッド構築を容易にするが、並列計算が行えるわけではない。そのため、並列計算を行う場合は、別のシステムを導入する必要がある。これを実現するのがMPICHである。MPICHは並列計算のための通信ライブラリであり、MPI (Message Passing Interface)の仕様に基づき作成されている。これをSGEで運用することによって、SGEのスケジューリング機能によって効率的な並列計算を行うことが出来る。

本研究では、MPIとSGEとを組み合わせることによって、PC単体で運用する場合よりも効率的なシステムを構築することを目的とする。

*富士通 平成19年度3月 小山高専電子制御工学科卒業

2．ＭＰＩ

MPIとは、メモリ分散方式で並列化を実現するメッセージ通信のための仕様である。この仕様に基づき、ライブラリが作成されている。これは、C言語もしくはFortranから呼びだすことができるため、移植性が良い。また、新しい言語を習得する必要もないため、利用するに当たり負担も少なくて済む。

MPIの主な実装としてMPICHやMPICH-Scoreがある。本研究では、MPICHを用いる。これは、広く使われてきているため、動作が比較的安定しているためである。開発言語としては、C言語を利用した。

MPIは各種コンピュータシステム上で最大のパフォーマンスを引き出すように注意深く設計されており、並列コンピュータシステムにおいて最も広く用いられているメッセージパッシングに基づいているため急速に広範な支持を得ている。MPIの導入により可能性のある効率的なライブラリの記述が可能となり、このライブラリを使用することで簡単に並列処理を行うことが出来るようになる。

プログラムとしてはデータの配置から転送まで全て記載する必要があるため、負担はかかるものの、チューニング次第で最大限の性能を引き出すことも可能である。

3．ＳＧＥ

Sun Grid Engineとは、Sun Microsystems社が

開発した、オープンソースのグリッドコンピューティングを実現するためのツールである。

以下にSGEを利用する利点を記す。

① 使用するリソースの管理が容易になる。
② 資源の空いたリソースを有効活用できる。
③ 多数のジョブを効率よく処理できる。

これらは、SGEの特徴であるスケジューリング機能によってもたらされるものである。スケジューリングについては後述するが、まずはSGEによるグリッドの構造について説明する。

3－1　SGEのホスト

SGEによるグリッドを構築する際、そのリソースとして利用するPCには、いくつかの役割が割り振られる。以下にその役割を挙げる。

(1) マスターホスト

スケジューリングのプログラムが動作しているホストである。システムの中心的な役割であり、全ホストの状態を監視している。基本的に1台のみ使用する。このホストはクラスタ内で最も高い信頼性を要する。プロセッサを複数搭載しているクラスタや、安定したプラットフォームでの構成が望ましいとされる。

(2) 実行ホスト

マスターホストがスケジューリングしたジョブを、実際に実行するプロセスが動作しているホストである。実行ホストの数は決まってはおらず、数が多いほどシステムのパフォーマンスは向上していくとされる。マスターホストも実行ホストとして動作させることも可能であるが、実行ホストとしての処理でマスターホストがビジー状態に陥ることを防ぐためにも、マスターホストにこの役割を与えることは望ましくない。

(3) 管理ホスト

クラスタ内の計算機の管理を行うことを許可するホストである。マスターホストと兼用するかどうか、実行ホストすべてを管理ホストとするかなどを決定する。どこからでも管理出来るようにするか、管理を一括するかなど考え方によってその設定は異なる。

(4) 発行ホスト

ジョブの発行と管理が可能なホスト。発行ホストで発行したジョブを、マスターホストがスケジューリングし、実行ホストがそれを処理するという流れとなる。

このような役割を各PCに割り振ることでグリッドを構築する。

本研究で構築したグリッドは、発行ホストと管理ホストの役割をマスターホストに与えた、マスターホストと実行ホストのみで構成されるものである。以下にその構造図を示す。

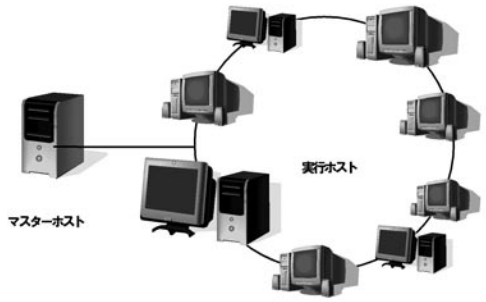

図1　SGEによって構築されるグリッド

図1において、各PCの大きさがまちまちなのは、それぞれの計算能力の大きさを表しているものとする。ここで、SGEによるジョブの処理の流れは次のようになる。

① マスターホストにジョブを投入する。
② ジョブがキューに蓄えられる。ジョブに関する情報（ユーザ、グループ、投入時刻）などが記録される。
③ ジョブが実行可能になると、スケジューラ(SGEの持つ機能)はポリシー（ユーザ指定のスケジュールのパラメータや方式）に従ってジョブを実行ホストに受け渡す。
④ 受け渡す実行ホストは、ぞれぞれの処理能力によって決定される。同時に多数のジョブが投入された場合、図において大きい実行ホストにはその分ジョブを多く受け渡し、小さく描かれているものにはその分少なめに受け渡す、あるいは受け渡さないように決定される。

3－2　スケジューリング

スケジューリングには、ジョブのスケジューリングと、キューのスケジューリングの2つがある。まず、ジョブのスケジューリングについて説明

する。

(1) ジョブのスケジューリング

ジョブのスケジューリングとは、投入されたジョブの実行順序を決定することである。この順序は、次のような方法で決定される。

① ユーザソート

同じユーザが複数のジョブを投入し、その後別のユーザがジョブを投入すると、ユーザごとにジョブのスケジューリングが行われ、1人のユーザのジョブが連続してスケジューリングされないようになること。

② ジョブ優先順位でのソート

ジョブ投入時、オプションとして管理者はそのジョブの優先度を与えることが出来る。この優先度が高いものを優先してスケジューリングすることを指す。

③ ユーザ/／ループのジョブ最大数

任意のユーザやグループに対し、実行可能なジョブの数を制限できる。この制限を超えないように、スケジューラはジョブの優先度を変化させる。

このようにして、ジョブはソートされ、その先頭から順番に実行される。この際、ジョブに要求されるリソースが不足する場合、次のジョブがスケジューリングの対象となる。

次に、キューのスケジューリングについて説明する。

(2) キューのスケジューリング

まず、キューとは、1台以上のホストで同時に実行できるジョブのクラス用のコンテナである。ここから実行ホストはジョブを実行する。

キューのスケジューリングとは、このキューに対してジョブを割り振る際、どのキューにジョブを受け渡すかを決定することである。

キューのスケジューリングは、システムで定義された負荷情報によって実行される。負荷情報には、CPU負荷、IO負荷、空きメモリなどのものがあるが、管理者が更に追加することも出来る。

システム負荷の正規化には、次の式が用いられる。

$$load\_val1[*w1][+|-]load\_val2[*w2][+|-]...$$

ここでload_val1…load_val*n*はn個のパラメータであり、その重みがそれぞれw1,w2,…wnによって与えられる。スケジューラはこの値が大きいものから順にキューをソートする。

また、ホスト間の値のスケーリングを与えることで、異なるホストが同じ値を返した場合にどちらを選ぶかという選択を与えることが出来る。これを用いる場合、同じパフォーマンスを与えるのに必要なコストを考慮することが代表的である。この値は、全体のスケジューリングを行う際にスケーリングされ、上記システム負荷の正規化への更なる係数となる。

以上のようにスケジューラは、ユーザからのジョブの投入があると、以下のような順で動作する。

①. ジョブを優先度、ユーザ制約に基づきソートする。

②. キューを正規化された負荷情報に基づきソートする。

③. ジョブが要求するリソースとの調整を行う。

このようなスケジューリング機能によってシステムを管理することで、処理効率の高いグリッドを構築できるのが、SGEを用いる利点である。

4．システムの構築と設定

MPIを用いた並列分散処理が行える環境を構築し、その環境をSGEによって管理・制御することで、処理の効率化を図る。また、処理効率がどの程度上昇しているかについても調べる。MPIは特にOSに制限はないが、SGEはWindows上では限定された動作しか出来ないという理由から、基本的にLinux上で使用するソフトであるため、本研究ではLinux環境として、無償で提供されているVine Linux3.2及び4.0を用いることにする。

本研究で用いるPCの性能を以下に記す。

(1)マスターホストPC

gmaster.oyama-ct.ac.jp

OS Vine Linux3.2

CPU Intel Pentium Ⅲ 600MHz

メモリ 512MB RAM

(2)実行ホストPC

①grid01.oyama-ct.ac.jp

OS Vine Linux4.0

CPU Intel Pentium4 2.60GHz

メモリ　1024MB RAM

②grid02.oyama-ct.ac.jp
OS　Vine Linux3.2
CPU　Intel Pentium4 2.40GHz
メモリ　512MB RAM

③grid03.oyama-ct.ac.jp
④grid04.oyama-ct.ac.jp
⑤grid05.oyama-ct.ac.jp
⑥grid06.oyama-ct.ac.jp
⑦grid07.oyama-ct.ac.jp
⑧grid08.oyama-ct.ac.jp
OS　Vine Linux3.2
CPU　Intel Pentium4　2.6GHz
メモリ　1024MB RAM

ネットワークは全て100Mbps Fast Ethernet、レイヤ2スイッチで接続している。

上記のPCを用いて、マスターホスト1台、実行ホスト8台で、SGEによるグリッドを構築し、各PCに対しMPICHをインストールした。本研究では、マスターホストに対してgmaster.oyama-ct.ac.jp、実行ホストに対してはgrid**.oyama-ct.ac.jpというホスト名をつけた。

MPICHの関数を用いたプログラムをジョブとして投入するためには、シェルスクリプトで先に設定した並列環境を使用して実行するよう指定する必要がある。

MPICHによって提供される関数を用いたプログラムを実行するよう指定したジョブを、SGEに投入したところ、実際に動作することが確認できた。また、その使用するホストも、その都度変化することから、SGEのスケジューリング機能が働いていることも確認出来た。下の図2は、実際にSGEから実行した並列計算で、その際使用したホストをテキストファイルとして出力させたものである。

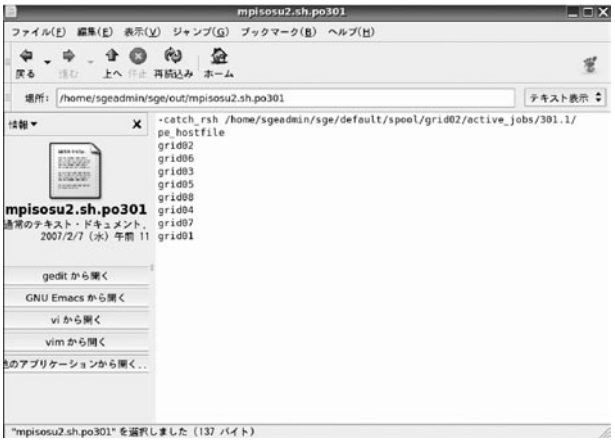

図２　SGEを用いた並列分散処理システムで用いられた実行ホスト

５．実験方法と結果

次に、SGEを用いることのメリットを確かめるため、MPIのみの並列分散処理システムと、SGEを用いた並列分散処理システムとでは、どの程度効率に差が生じるのかを調べることとした。

方法として、単純な計算を繰り返す、素数の判定プログラムを用いて効率を調べることにした。また、各ホストへの負荷をある程度一定にするため、計算数が多くなるものと少なくなるものの2種類を引き渡すようアルゴリズムを考えた。例えば、5台のPCで100までの素数判定をする場合、1番目のPCは1～10と91～100、2番目のPCは11～20と81～90というように、小さい数字と大きい数字の両方を計算させるようにした。

このプログラムで､50000までの素数の個数を調べるのに要する時間を計測し、各々の場合の差を比較することとした。

MPIのみで実行する場合、実行ホストを指定する必要がある。今回は､grid01～08まで名前の順番に使用するよう指定した｡【プログラム1】にMPIプログラム実行用のシェルスクリプトを示し､【プログラム２】に今回使用した素数の個数測定プログラムを示す。

【プログラム１】 MIPプログラム実行用シェルスクリプト

```
#!/bin/sh

#$ -pe mpich 7
```

```
/*　ここで、使用する並列環境とプロセス数を指定する。この場合、先に設定した並列環境"mpich"を、
プロセス数７で使用する。*/

/home/sgeadmin/mpich-1.2.7p1/bin/mpirun -np $NSLOTS -machinefile $TMPDIR/machines
/home/sgeadmin/sge/original2
/*　mpi プログラムを実行する命令文である。np オプションは使用するプロセス数を指定するもので、
machinefile オプションは使用するホストを指定するものである。これらは SGE によって操作されるた
め、上のように環境変数を記述する。
最初に mpirun (mpi プログラムの実行コマンド) を書き、上のようなオプションをつけ、最後に実行した
い mpi プログラムを記述する。環境変数の設定がどれか１ホストでも抜けているとエラーが生じてしまう
と思われるため、今回は絶対アドレスで書いている。　*/
```

図３は、上のジョブをqmonのSubmit Jobから投入した場合の画面である。

図３　ジョブ投入時の画面

【プログラム２】　素数の個数測定プログラム

```
#include <stdio.h>
#include <math.h>
#include "mpi.h"
#define N 50000


int main(int argv,char* argc[])
{
        int n,base_n,source;
        int low1,low2,high1,high2;
```

```
int i;
int a;                    //配列の位置
int size,rank;
int low_array[N];         //下位の計算結果格納
int high_array[N];        //上位の計算結果格納
int num;                  //配列の数

int search(int,int,int[]);

double starttime,endtime;

MPI_Status status;

n=N;

MPI_Init(&argv,&argc);    //並列処理の開始手続き

starttime = MPI_Wtime(); //開始時間の記録

MPI_Comm_size(MPI_COMM_WORLD,&size);       //全プロセス数を取得
MPI_Comm_rank(MPI_COMM_WORLD,&rank);       //自プロセスのランクを取得

for(i=0;i<N;i++)          //初期値設定
{
        low_array[i]=0;
        high_array[i]=0;
}
i=0;
a=0;
num=0;

//並列計算開始
if(rank!=0)
{
        base_n=n/((size-1)*2);    //自分の計算する範囲の個数を調べる
        low1=base_n*(rank-1)+1;   //下位の計算範囲の始点
        low2=base_n*rank;         //下位の計算範囲の終点
        high1=n-base_n*rank+1;    //上位の計算範囲の始点
        high2=n-base_n*(rank-1); //上位の計算範囲の終点

        //下方の素数検索
        search(low1,low2,low_array);
        //上方の素数検索
        search(high1,high2,high_array);
```

```
                //送信
                MPI_Send(&low_array,N,MPI_INT,0,0,MPI_COMM_WORLD);
                MPI_Send(&high_array,N,MPI_INT,0,0,MPI_COMM_WORLD);
        }
        else
        {
                //ランク１のプロセスが素数の個数を受け取る
                for(source=1;source<size;source++)
                {
                        MPI_Recv
        (&low_array,N,MPI_INT,source,MPI_ANY_TAG,MPI_COMM_WORLD,&status);
                        while(low_array[i]!=0)
                        {
                                num=num+1;
                                i++;
                        }
                        i=0;


        MPI_Recv(&high_array,N,MPI_INT,source,MPI_ANY_TAG,MPI_COMM_WORLD,&status);
                        while(high_array[i]!=0)
                        {
                                num++;
                                i++;
                        }
                        i=0;
                }
                endtime = MPI_Wtime();              //終了時間の記録
                printf("%dまでの素数の数は%dです。¥n",n,num-1);
                printf("%lf¥n",(endtime-starttime));
        }


        MPI_Finalize();
}



//素数判定
void search(int a,int b,int array[N/2])
{
        int i,j,frag,now;
```

```
        frag=0;
        now=0;

        for(i=a;i<=b;i++)
        {
                for(j=2;j<=i/2;j++)
                {
                        if(i%j==0)
                                frag=1;
                }

                //素数ならば素数の配列に現在の数字を入れる
                if(frag==0)
                {
                        array[now]=i;
                        now++;
                }
                frag=0;
        }
}
```

＜プログラムの動き＞

数字Nまでの素数を判定し、その個数をデータ受信時に測定することで、Nまでの素数の数を調べる。

負荷をある程度分散するため、計算数が多い上位の数字と、計算数が少なくてすむ下位の数字の両方を各ホストに計算させることにする。たとえば、N=100で実行ホスト数6の場合、

Rank0のホスト→他のホストの計算結果の受信と、素数の個数測定

Rank1のホスト→1 ～ 10,91 ～ 100

Rank2のホスト→11 ～ 21,81 ～ 90

…

Rank5のホスト→41 ～ 50,51 ～ 60

というように役割を分担させる。

まず計算範囲を各ランク毎に決定し、下位と上位の2回にわけて計算し、結果を送信する。Rank0となったホストは、それを2回にわけで受信し、素数の個数を計測する。

この結果が次の表1と図4である。測定は3回実行し、その平均値をとった。なお、実行ホスト数1台で実行していないのは、並列計算時に、その計算結果を取りまとめる役目を担うホストが1台必要なため、実行ホスト数1では計算自体が出来なくなってしまうためである。そのため、実行ホスト数2の結果が、1台で計算した時のものとほぼ等しい結果と言える（勿論、通信による遅延などが含まれてはいる）。

この結果を見ると、SGE使用時の方が計算時間が短いことが確認出来る。故に、SGEのスケジューリング機能を利用することで、より効率の良い並列分散処理システムが構築出来ると言えるだろう。

表1　SGE使用時／未使用時の実行時間

実行ホスト数	SGE使用[sec]	SGE未使用[sec]
2	12.41	12.6
3	6.68	6.88
4	5.1	5.17
5	4.38	4.5
6	4.11	4.26
7	4.2	4.31

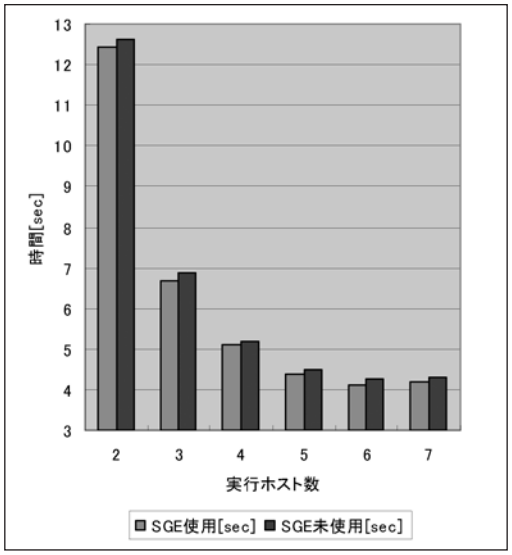

図4　SGE使用時／未使用時の比較

SGEを介してMPI関数を用いた並列計算ジョブを実行する、並列分散処理システムの構築が出来るようになり、また、SGEを用いることによって得られる利点についても確認することが出来た。すなわち、SGEによるスケジューリング機能が働くことで、並列計算を行なうホストを、処理能力が高いものから順に選んでいくため、処理速度がより高速になるということである。

今回の結果では、大きな差は出ていない。本研究では4章で述べたような性能の実行ホストを使用したのだが、その性能に大きな差異はない。このため、SGEにより効率化されたといえども著しい差は出なかったと考えられる。各実行ホストの性能に更に大きな差がある場合は、結果に大きな差が生じると思われる。

SGEを用いた並列分散処理システムの利点は先に挙げた通り、スケジューリングによる効率化が図れるというものがあるが、他にも、GUIを用いた、ジョブや実行ホストの管理が可能であるという点が挙げられる。

実行中に何度かエラーが発生し、特定の実行ホストを使うとエラーが発生するということがあった。このような場合、MPIではマシンファイルをテキストエディタ等で変更しなければ使用しないよう設定することは出来ない。しかし、SGEによって提供されているqmonを用いれば、キュー管理の機能でそのような実行ホストを使用しないようマウスのみで設定出来る上、エラーが発生して実行が停止したジョブを、ジョブ管理機能によって削除することが出来る。

上に挙げたことは、MPIのみでも出来ることも含まれてはいるが、そのためにはファイルの編集など手間がかかる。SGEを用いることでその手間を減らすことが出来るという点は、利点として挙げられるだろう。

問題点としては、前述の通りジョブの実行段階でエラーが発生する場合がある、という点が挙げられる。原因が不明なエラーも発生しているが、たまにタイムアウトエラーも発生していることから、おそらく通信部分で何らかの問題が生じていると思われる。

6．まとめ

MPIとSGEによる並列分散処理システムを構築し、その動作の確認と処理速度の向上を確認することができた。今後は通信エラーの原因を解明することと別のサンプルプログラムにより処理効率の測定を行っていきたい。

参考文献

(1) Grid Engine簡易操作マニュアル・管理マニュアル
http://www.nabe-intl.co.jp/manuals/ge_manual_5.html
(2) N1 Grid Engine 6 Collection
http://docs.sun.com/app/docs/coll/1187.1
(3) gridengine: ホーム
http://gridengine.sunsource.net/
(4) 小山工業高等専門学校・電子制御工学科　卒業論文　原田将徳：グリッドコンピューティング(2005)
(5) 小山工業高等専門学校・電子制御工学科　卒業論文　木村英明：MPIを用いた並列分散処理(2003)
(6) 共立出版　湯淺太一・安村通晃・中田登志之編：はじめての並列プログラミング(1998)