# Linpackゼミ

資料作成者 : 川崎 考蔵 , 渡辺 彰人
作成日 : 2009 年 8 月 6 日

**ゼミ内容:** クラスタの性能評価を行うベンチマークについて，代表的なリンパックについて理解する，そして Gotoblas，HPL のインストールの方法・実行方法について理解する．

## 1 ベンチマーク

### 1.1 性能をあらわす指標

コンピュータのコスト性能比を調べる際，何らかの指標に基づいて実際に計測し，その結果を比較しなければ，計算能力の客観的な判断ができない．コンピュータの性能を示す指標としては，以下のものがある．

- MIPS（Million Instrucctions Per Second）
  マイクロプロセッサの性能を示す値．1 秒間に何百万回の命令を処理できるかを示す．
- Flops（Floating Operations Per Second）
  1 秒間に何回の浮動小数点演算ができるかを示す．浮動小数点演算とは，浮動小数点数の四則演算である．浮動小数点演算は，科学技術計算において多用される．
- MOPS（Million Operation Per Second）
  ディジタル信号処理を行なうプロセッサなどの性能指標に用いられるベンチマーク．1 秒間に何百万回の命令を実行できるかを示す．

これらの指標は，アーキテクチャや計測に用いるプログラムによって，結果が異なる．一方で，コンピュータの用途は多様であり，同じコンピュータでも問題ごとに性能が異なる．つまり，これらはコンピュータの性能をあらわす絶対的な指標ではない．

### 1.2 ベンチマークの目的

ベンチマークはコンピュータの計算処理能力を測るために，前節のような指標を表すものであるが，上記のような理由からその指標が絶対であるとはいえない．つまり，ベンチマークを用いた計算機の性能の測定は困難である．しかし，ベンチマークの特性を知ることにより，ある特定の問題に対する性能比較を行ったり，ある計算機の得意の問題を調べることはできる．

### 1.3 ベンチマークの種類

ベンチマークは，評価の対象からいくつかの種類に分類できるが，大きくは以下の 2 つに分類される．

1. マクロベンチマーク（総合ベンチマーク）
2. ミクロベンチマーク（機能別ベンチマーク）

マクロベンチマークを用いてシステムの総合的な評価を下し，ミクロベンチマークテストを用いて機能別の評価を下す．このようにベンチマークを使い分けることで，システムの改善すべき点，それによる総合評価への影響などを知ることができ，さらなるシステムの総合的な処理能力を効率的に向上させる手がかりとなる．つまり，マクロベンチマークでは全体の，ミクロベンチマークでは個々のベンチマークを行うことになる．以下に，マクロベンチマーク，ミクロベンチマークについて詳しく示す．

#### 1.3.1 マクロベンチマーク

マクロベンチマークは，システム全体の性能を評価するためのものである．すなわち，マクロベンチマークの実行結果には，CPU 性能，ネットワーク性能，メモリ性能等のミクロベンチマークの集合が総合的に反映される．

#### 1.3.2 ミクロベンチマーク

ミクロベンチマークは，システムのいくつかの機能の性能を測定する．その例としては，以下に挙げるようなものがある．

1. 演算性能ベンチマーク
   MIPS などを用いて，コンピュータの演算性能を測る．
2. グラフィック性能ベンチマーク
   あるグラフィック動作を起こさせ，1 秒間に何回行えるかを計測する．
3. ネットワーク性能ベンチマーク
   単位時間内における処理量，待ち時間，往復時間（RTT）などで測定する．

## 2 Linpack Benchmark

Linpack Benchmark（以下 Linpack）は，米国テネシー大学の J.Dongarra 博士によって開発された LU 分

解にもとづく連立一次方程式の解法プログラムである．もとは，線形代数問題の LU 分解を行うためのサブルーチンの集合であり，ベンチマークにも利用されていると表現するほうが正確である．ベンチマークとしての Linpack は，計算機で頻繁に利用する命令を集めたカーネルベンチマークであったが，実行結果に分散メモリ型並列計算機のためのベンチマークとして HPL が開発された．

Linpack の概要に関して，以下の記述がある．

> Linpack ベンチマークは連立一次方程式を解くことができる．この性能が現実問題において反映されるシステムは 1 台もない．しかしながら，連立一次方程式を解くことによって，特定用途で使用されるシステムの性能を反映する．その問題がとても規則的なので，実行結果はかなり高い．そして，性能の多くは良いピーク性能の良い補正となる．

世界のスーパーコンピュータの性能を競って，その結果をランキングする TOP500 で Linpack が使用されている都合上，本ゼミでは Linpack についておおまかな説明をする．

### 2.1 パラメータ

Linpack には 17 のパラメータが存在する．それらの中で，計測結果に大きく影響を及ぼすものについて説明する．

#### 2.1.1 問題サイズ（N）

問題サイズ（N）は Linpack で解く問題の大きさである．つまり，Linpack では N 次元連立方程式を解くことになる．N は Linpack の実行結果に最も影響を及ぼす．大体において，N が大きくなるほど良い結果が得られるが，N が大きくなるほどメモリ使用量が増える．以下に最適な結果をもたらす N の値について述べる．

1. N は，メモリの 80 %を使用するように設定すると良い結果が得られる．
2. N の 2 乗が係数行列の要素数となる．そして行列の各要素は double 型なので，上の条件を満たす N の値は，以下の式によって求められる．

$$N = \sqrt{(\text{計算対象となる計算機の全メモリ容量 } [byte]) \times 0.8/8}$$

#### 2.1.2 ブロックサイズ（NB）

ブロックサイズ（NB）は，粒度のことである．NB が大きくなると，通信量が減るがロードバランスが悪くなり，NB が小さくなると，通信量が増えるがロードバランスが良くなる．NB の値を 32〜256 にすると，良い結果が得られる．

#### 2.1.3 プロセスグリッド（P，Q）

プロセスグリッド（P と Q）は，問題の行列をそれぞれのプロセスにどのように分割するかを示す．必然的に P と Q の積が実行ノード数となる．P と Q は等しいか P より Q が大きい方が良い．

### 2.2 バリエーション

Linpack を測定するに当たってのルールとして，TPP（Toward Peak Performance）と N=100 と呼ばれる二つのルールがある．TOP500 で用いられているルールは，TPP である．以下に各ルールの特徴に付いて述べる．

#### 2.2.1 TPP（Toward Peak Performance）

TPP では，ソース修正による最適化が許されており，ベストな性能に達するため，ユーザが問題サイズを決めるなど，多くのパラメータを変更することができる．また，コンパイラ等のソフトウェアを最大限に利用することができる．また，問題が規則的なので，結果として得られる性能はかなり高い．このように，Linpack では，各システムの特徴にあわせたチューニングが可能である．

このルールでは，絶対的な性能値とともに，与えられた問題規模において，ハードウェアのピーク性能にどこまで近づけるかが，評価のポイントになる．

#### 2.2.2 N=100

N=100 の基本的なルールとしては，問題サイズを N=100 に固定で，ソースの修正による最適化が許されていないということがある．これは，ベクトル長の短い場合の性能を示し，ハードウェア性能のみならずコンパイラの機能も併せて，総合性能を測定することができる．

## 3 Linpack インストール

Linpack は米テネシー大学の Jack Dongarra 博士が開発した連立 1 次方程式の解法プログラムである．行列計算のライブラリである GotoBLAS と，TOP500 リストで使用されるベンチマークである，Linpack を高度に並列化したプログラム「HPL」(High-Performance Linpack) のインストールの手順を以下に示す．

### 3.1 動作確認環境

本マニュアルにおける動作確認環境を以下に示す．

> * CPU ‥ Intel Xeon E5430 2.66GHz（2 コア）
> * メモリ ‥ DDR2 2GB
> * OS ‥ Debian etch（AMD64 版）

### 3.2 コンパイラのインストール

HPL の実行ファイルを作成するためのコンパイラのインストールを行う．本報告では，代表的な C コンパイラである gcc を利用する事により，HPL 実行ファイルを作成する．

まず，superuser になり，パッケージから gcc をインストールする．

```
$ su # aptitude install gcc
```

GotoBLAS をインストールする際，以下のコンパイラが必要になる場合が存在するため，以下のようにインストールを行う．

```
# aptitude install g77
# aptitude install gfortran
# aptitude install g++
```

これによりコンパイラのインストールは終了である．

### 3.3 GotoBLAS のインストール

GotoBLAS は HPL の実行ファイルを作成する際に利用することのできる数値計算ライブラリである．HPL に利用できる数値計算ライブラリは他にも ATLAS などが存在するが，本報告ではより効率よく計算できるとされている GotoBLAS を利用する．

http://www.tacc.utexas.edu/resources/software/#blas からソースを入手する．その際，レジストレーションが必要になる．

まず，以下のパッケージをインストールする．

```
# aptitude install libf2c2-dev # aptitude install
make
```

次に，入手したソースファイルを展開すし，作成されるディレクトリ GotoBLAS に移動する

```
$ tar xvfz GotoBLAS-1.26.tar.gz
$ cd GotoBLAS
```

次に，Make.rule により make する際の環境を編集する，本環境ではでフォルトのコンパイラである gcc を利用するため特に編集する必要はない．また，GotoBLAS にはより簡単にインストールを行うためのシェルスクリプトが内包されている．本報告ではそのシェルスクリプトを利用することにより実行ファイルを作成する．

```
環境に合わせて Make.rule を編集
$ sh quickbuild.64bit
```

上記を実行すると，libgoto_core2-r1.26.so というファイルが作成される．

### 3.4 mpich のインストール

HPL は並列計算を行うためのプログラムである．そのため，並列計算を行う環境を構築する必要がある．本報告では，並列計算には mpich を利用する．

以下のパッケージをインストールする．

```
# aptitude install libmpich1.0-dev mpich-bin
```

mpich は，多ノードとの通信を rsh により行うため rsh をインストールする．また，rsh により通信を行うノードは/etc/hosts.equiv に記述する．

```
# aptitude install rsh
```

以上により，並列計算環境の構築は終了である．

### 3.5 HPL のインストール

最後に HPL 本体のインストール及び，実行ファイルの作成を行う．

はじめに，以下のように公式サイトからソースファイルをダウンロードし，解凍する．

```
$ wget http://www.netlib.org/benchmark
/hpl/hpl.tgz
$ tar xvfz hpl.tgz
```

次に，解凍した際，作成されるディレクトリ hpl に移動し，実行ファイルを作成する際に利用する変数の設定を行う．設定すべき項目は，利用数値計算ライブラリ (libgoto_core2-r1.26.so) のパス，利用する C コンパイラ，並列計算プログラムの場所等である．各変数を環境に合わせて設定する．

```
$ cd hpl
$ cp setup/Make.Linux_PII_CBLAS .
Make.Linux_PII_CBLAS を以下のように編集
TOPdir = $(HOME)/hpl
LAdir = $(HOME)/GotoBLAS
LAlib = $(LAdir)/libgoto_core2-r1.26.so
HPL_LIBS = $(HPLlib) $(LAlib) $(MPlib) -lm
HPL_OPTS =
CC = gcc
LINKER = gcc
MPdir = /usr/lib/mpich
```

最後に make を行い，実行ファイルを作成する．

```
$ make arch=Linux_PII_CBLAS
```

以上の操作により bin/Linux_PII_CBLAS 以下に xhpl という実行ファイルが作成されていれば成功である．

### 3.6 HPL の実行

実行方法を示していく．mpich では rsh を用いるため，rsh により利用されるノードを記す必要がある．

```
# echo ”localhost” >> /etc/hosts.equiv
```

次に，実行ファイルの存在するディレクトリに移動する．

```
$ cd hpl/bin/Linux_PII_CBLAS
```

最後に，xhpl を実行する

```
$ mpirun -np 4 xhpl
```

以上により，HPL 実行環境構築及び，実行の手順は終了である．