解説　計算機システムの性能評価と
プログラムチューニング（前編）

計算機システムの性能評価とプログラムチューニング（前編）

筑波大学電子・情報工学系／計算物理学研究センター　高橋 大介
daisuke@is.tsukuba.ac.jp

計算機システムの性能評価は，情報処理分野の論文を書く上で必要となることが多い．しかしながら，性能評価そのものについての記事はあまり見かけない．本稿では，ハードウェアの性能評価について述べるとともに，各種ベンチマークについて概説し，プログラムのチューニング手法について解説する．

【 性能評価の目的 】

本稿では，計算機システムの性能評価とプログラムチューニングについて解説する．本稿は，前編と後編に分かれている．前編では，性能評価全般およびプログラムのチューニングの基本について述べ，後編ではプログラムのチューニングに関して，より突っ込んだ内容について述べる予定である．

計算機システムを実際に使ってみて，「性能が高いと思って使ってみたけれども，実際は思ったほど性能が出なかった」という経験をしたことのある読者は多いと思う．

これには，大きく分けて2つの理由があるように思える．つまり，

- 「性能が高い」といわれていたのは，その計算機システムの得意なある一面を指したものであり，ユーザが実行しようとした計算には向いていなかった．
- 本当はその計算機システムは高い性能を秘めていたはずであるが，ユーザの使い方に問題があり，その高い性能を引き出せなかった．

といったものである．

このような例は，エンドユーザの購入するPCであれば，そのユーザだけの問題であるが，研究室で購入するワークステーションであるとか，計算機センターに導入するスーパーコンピュータのような場合は，金額や規模が大きいので，大きな問題となる．

効率のよい計算（ハイパフォーマンスコンピューティング，以下HPC）を行うには，高度の知識と経験が必要であることが知られているが，本稿ではその中でも特に関心が高いと思われる，「性能評価」と「プログラムチューニング」について述べることにする．

世の中に計算機が1機種しかなく，今後も進歩がないのであれば，「性能評価」はあまり必要がないといえる．しかし，現実には世界に非常に多くのプロセッサや計算機システムが普及しており，自分の解決したい問題をどの計算機システムが効率よく計算してくれるか，ということをユーザ自身が判断する必要がある．

また，計算機システムの開発者側からは，計算機の性能をハードウェアやソフトウェアの観点から改良していく際には，どうしても「汝自身」を知るために，「性能評価」を行い，改良に役立てる必要がある．

性能評価を行うことで，計算機システムの性能がどの程度のもので，またどういった問題に向いているかということを知ることができる．また，問題が大き過ぎて実行するのに非常に時間がかかる計算に要する時間を，実行する前に予測することができる．

さらに，コストパフォーマンスの高い計算を行うには，計算機システムを使用する際のコストと，性能の両面からユーザは判断することになる．つまり，「価格」も性能のうち，というわけである．

解説　計算機システムの性能評価と
プログラムチューニング（前編）

名称	内　容
EP	乗算合同法による一様乱数，正規乱数の生成
MG	簡略化されたマルチグリッド法のカーネル
CG	正値対称な大規模疎行列の最小固有値を求めるための共役勾配法
FT	FFTを用いた3次元偏微分方程式の解法
IS	大規模整数ソート
LU	Synmetric SOR iterationによるCFDアプリケーション
SP	Scalar ADI iterationによるCFDアプリケーション
BT	5 *times* 5 block ADI iterationによるCFDアプリケーション

表-1　NAS Parallel Benchmarks（NPB）

	名称	n=100	TPP n=1000	Highly Parallel Computing
1位	Fujitsu VPP5000/1	1156MFLOPS	NEC SX-5/16 45030MFLOPS	ASCI White-Pacific 7226GFLOPS
2位	Cray T94（4proc.）	1129MFLOPS	NEC SX-5/8 32570MFLOPS	IBM SP Power3（158nodes）2526GFLOPS
3位	Cray T94（3proc.）	1029MFLOPS	NEC SX-4/32 31060MFLOPS	ASCI Red 2379.6GFLOPS

表-2　LINPACKベンチマークの結果の例

【 性能評価の指標 】

コンピュータの性能を評価する指標としては次のものがある．

•MIPS（Million Instructions Per Second）

*MIPS*とは*CPU*が*1*秒間に何百万回の命令を実行できるかということを表したもので，*CPU*の性能評価を行うときの尺度の*1*つである．たとえば，*1*秒間に*1,000*万回の命令を実行したのであれば，*10MIPS*ということになる．ここで注意したいのは，*MIPS*はあくまでも命令実行回数であるので，アーキテクチャが異なるコンピュータ間の性能比較には適していないということである．コンピュータ間の性能比較に適した*MIPS*値としては，*VAX MIPS*（*Dhrystone MIPS*と呼ばれることもある）が使われることが多い．

•FLOPS（Floating Operations Per Second）

*1*秒間に実行可能な浮動小数点演算の回数を表す単位として，*FLOPS*が使われる．演算回数の測定を*100*万回単位にしたときには*MFLOPS*（*Mega FLOPS*），*10*億回単位なら*GFLOPS*（*Giga FLOPS*），*1*兆回単位なら*TFLOPS*（*Tera FLOPS*）を使う．異なるコンピュータ，アーキテクチャの比較検討に用いられることが多い．

•SPEC

*SPEC*ベンチマーク（後述）の値であり，*SPECint*が整数演算性能，*SPECfp*が浮動小数点演算性能であり，両者を合わせた評価を*SPECmark*で表す．

【 各種ベンチマークの概要 】

ベンチマークテストとは，コンピュータのハードウェア，ソフトウェアの処理性能を評価・比較するためのテストである．典型的な一連の処理を設定し，その実行にかかった時間を測定することにより，評価を行う．ベンチマークテストの結果は，テストのバージョンや，主記憶およびキャッシュの容量，コンパイラの最適化機能などの条件により大きく変わる．ベンチマークテストで優れた結果を得るために，多くのベンダが一般に市販されていないコンパイラや特殊な最適化機能を使うことがあった．このため*SPEC*ベンチマークの最新版では，標準コンパイラによるテスト結果を併記することを義務付けている．

Dhrystone

*Dhrystone*ベンチマーク[1)]はかなり昔から使われている整数演算性能を評価するベンチマークテストの*1*つである．*Dhrystone*ベンチマークは複数の小さなループ処理を主体とし，通常はすべてのループを*L1*キャッシュに常駐させることが可能である．したがって，このベンチマークは*L1*キャッシュを無限大と仮定した場合のプロセッサ性能を表すことになり，*L2*キャッシュの容量やメモリアクセス性能によって特性はほとんど左右されない．

このような理由により，*Dhrystone*ベンチマークはシ


```
double A[N], B[N], C;
for（i = 0; i < N; i++）{
   A[i] = A[i] + B[i] * C;
}
```

図-1　あるループの例

```
double A[N], B[N], C;
for（i = 0; i < N; i += 4）{
   A[i] = A[i] + B[i] * C;
   A[i+1] = A[i+1] + B[i+1] * C;
   A[i+2] = A[i+2] + B[i+2] * C;
   A[i+3] = A[i+3] + B[i+3] * C;
}
```

図-2　ループアンローリングしたループの例

```
double A[N,N], B[N,N], C[N,N], D;
for（i = 0; i < N; i++）{
   for（j = 0; j < N; j++）{
      A[j,i] = B[j,i] + C[j,i] * D;
   }
}
```

図-3　ループの入れ換えが可能なループ

```
double A[N,N], B[N,N], C[N,N], D;
for（j = 0; j < N; j++）{
   for（i = 0; i < N; i++）{
      A[j,i] = B[j,i] + C[j,i] * D;
   }
}
```

図-4　ループを入れ換えたループ

```
double A[N,N], B[N,N];
for（i = 0; i < N; i++）{
   for（j = 0; j < N; j++）{
      A[i,j] = A[i,j] + B[j,i];
   }
}
```

図-5　ブロック化しない場合のループ

```
double A[N,N], B[N,N];
for（i = 0; i < N; i += blocksize）{
   for（j = 0; j < N; j += blocksize）{
      for（ii = i; ii < i + blocksize; ii++）{
         for（jj = j; jj < j + blocksize; jj++）{
            A[ii,jj] = A[ii,jj] + B[jj,ii];
         }
      }
   }
}
```

図-6　ブロック化した場合のループ

ステム全体の性能を正確に表すことができないので，最近はSPECベンチマークが使われることが多い．

またVAX MIPSは，かつて1MIPSマシンとして知られていたDECのミニコンVAX11-780の計算能力（1,767Dhrystones/sec）を元に算出するMIPS値である．ちなみに600MHz程度のクロックのPentiumIIIでも，1,000 VAX MIPS以上の性能が出ている．

Whetstone

Whetstoneベンチマーク[2]は，かなり昔から使われている浮動小数点演算性能を測定するベンチマークテストの1つである．Whetstoneベンチマークは，Dhrystoneベンチマークと同様に，通常はすべてのループをL1キャッシュに常駐させることが可能である．Whetstoneベンチマークはsin，cosなどの関数，整数および浮動小数点数の混合計算，分岐，スカラ変数などが総合的に測定される．コンパイラやハードウェアだけではなく，数学関数ライブラリに重点を置いてテストする．

Whetstoneベンチマークの性能は，MFLOPSで表されることが多い．

Livermore Fortran Kernels

Livermore Fortran Kernels[3]は，リバモアループとも呼ばれ，主にベクトル型のスーパーコンピュータで広く用いられてきたベンチマークテストである．14個のカーネルからなる，リバモア14ループと，24個のカーネルからなる，リバモア24ループがある．

LINPACK

LINPACKベンチマークは，テネシー大学のJack Dongarraによって開発された，浮動小数点演算能力を評価するためのベンチマークテストである（http://www.netlib.org/linpack）．LINPACKは，線形代数のサブルーチンライブラリの名前である．LINPACKベンチマークは，それらのサブルーチンを使って，ガウス消去法を用いて連立1次方程式の解を求めるのに要する時間を測定する．

LINPACKベンチマークにはいくつかのバージョンがあるが，ワークステーションのベンダは通常100×100の行列の倍精度の結果を引用することが多い．

LINPACKは，世界のスーパーコンピュータのランキング「TOP500 Supercomputer」（http://www.top500.org）のベンチマークにも用いられている．

分散メモリ型並列計算機向けのLINPACKのソフトウェアパッケージとして，Petitet，Whaley，Dongarra，Clearyらによって開発された，HPL（A Portable Implementation of the High-Performance LINPACK Benchmark for Distributed-Memory Computers）がある（http://www.netlib.org/benchmark/hpl）．

SPEC

SPEC（The Standard Performance Evaluation Corporation）とは，システム性能評価協会という米国のワークステーションメーカなどが設立した非営利の団体である．コンピュータ・システムの性能評価の標準化を進めている．

そのベンチマークテストの中で，UNIXが動作するコンピュータを主対象にした整数演算ベンチマークがSPECintである．intが整数の意味となる．ちなみに，浮動小数点演算ベンチマークはSPECfpという．最近のSPECベンチマークは多様化しているが，代表的なものとしてはSPECCPU2000等がある．

NAS Parallel Benchmarks（NPB）

NAS Parallel Benchmarks（NPB）[4]は，NASA Ames Research-Centerで開発された，主に並列コンピュータ用のベンチマークテストである．NPBは，**表-1**に示される，5つの

Parallel Kernel Benchmarks（*EP*，*MG*，*CG*，*FT*，*IS*）と*3*つの*Parallel CFD*（*Computational Fluid Dynamics*）*Application Benchmarks*（*LU*，*SP*，*BT*）から構成されており，並列コンピュータの実効性能を知る上で，最近よく使われているベンチマークの*1*つである．

さまざまな計算機におけるベンチマークの結果

*LINPACK*ベンチマークの各部門におけるトップ*3*の計算機と，その性能を**表-2**に示す．表-*2*から分かるように，各部門において，*1*位～*3*位の計算機は異なっていることが分かる．

【 プログラムチューニングの例 】

プログラムチューニングに関して，チューニング手法のいくつかを紹介する．

プログラムのチューニングには，*UNIX*の*prof*コマンドに代表される，プロファイラが有効である．たとえ自分で作成したプログラムであっても，思わぬところに性能向上を阻害するボトルネックがあったりするからである．

また，ベンダ等が提供している，数値計算ライブラリなどを使うのも，パフォーマンスを向上させる上ではきわめて有効であるといえる．

以下に述べるような一般的な最適化手法は，最近の高性能なコンパイラにより，自動的に行われることが多い．

ループアンローリング

ループアンローリングとは，ループを展開することにより，ループのオーバーヘッドを減らす最適化手法である．

図-1のループを，**図-2**のようにループを展開することにより，同じ演算からループのオーバーヘッドを減らし，繰返し回数を減らすことができる．ループを展開することで並列性が表面化され，複数回文の繰返しをまとめて処理できる．なお，図-*2*のループにおいて*N*が*4*で割り切れない際には，余りの繰返しを処理するループが必要になる．

ループアンローリングは有効な最適化手法であるが，あまりにもループを展開し過ぎると，レジスタが足りなくなったり，オブジェクトが大きくなるために命令キャッシュミスが発生することがあるので，注意が必要である．

ループの入れ換え

ループの入れ換えは，主にストライドの大きなメモリ参照による悪影響を軽減する手法である．

図-3のループでは，最内側のループにおいてストライドが*N*となり，実行は遅くなる．ここで，*N*の値が大きくなればなるほど，パフォーマンスは低下することが多い．

そこで，**図-4**のように，ループを入れ換えると，*A*，*B*，*C*は*1*ずつのストライドによるメモリアクセスに入れ換えられるので，パフォーマンスは大きく向上する．

ブロック化

ブロック化はメモリ参照を最適化するための有効な方法である．ブロック化の主な目的は，キャッシュミスをできるだけ減らすという点にある．この方法は，メモリ領域全体を大きなストライドでデータをアクセスする代わりに，隣接するメモリ領域を小さいストライドでアクセスできるようにする．

図-5は，*2*次元ベクトルの加算を行うループである．このループには*A*と*B*の*2*つのベクトルがある．一方は*1*ずつのストライドでアクセスされ，他方は*N*ずつのストライドになる．ループを入れ換えることはできるが，いずれにしても*A*または*B*への配列参照のストライドが*N*になってしまうので，このままではキャッシュミスが頻発してしまう．配列が大きければ，*TLB*（*Translation Lookaside Buffer*，仮想メモリアドレスから実メモリアドレスへの変換キャッシュのことをいう）ミスも発生するので，パフォーマンスは著しく低下する．この場合のメモリアクセスパターンを**図-7**に示す．

ここで，ブロック化という技法を用いることができる．つまり，*A*を数要素，次に*B*を数要素取ってきて，次に*A*を数要素というようにアクセスすることを繰り返すのである．ブロック化を行ったループを**図-6**に，メモリアクセスパターンを**図-8**に示す．ブロック化を行うことにより，メモリアクセスが局所化され，改善されていることが分かる．

【 性能評価を行う際の注意点 】

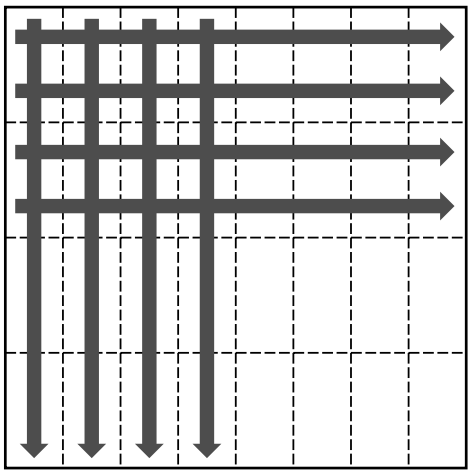

図-7　ブロック化しない場合のメモリアクセスパターン

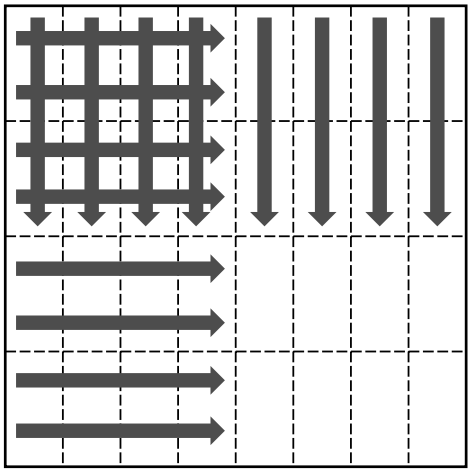

図-8　ブロック化した場合のメモリアクセスパターン