4.2. 実行性能と消費電力のベンチマーク

上智大学　南部 伸孝
富士通株式会社　軽部 行洋、石附 茂

SS 研 BMT の目的、特徴、活用方法、展開方法、BMT 項目、作成指針について述べる。

1. 目的

利用者の視点に立ち、プログラム性能を評価するためのベンチマークとする。さらに、消費電力に関するベンチマークも行えるものとする。

2. 特徴

演算の様子を再現した擬似コードではなく、実際に意味のあるシミュレーションを行い、プログラム全体を評価対象とする。また、各分野のプログラムを用意し、多くの研究者が利用できる。さらに、消費電力も計測し、運用管理者にも利用価値のあるベンチマークとする。

3. 活用方法

以下に示す 5 項目の観点で活用できるようにする。

1) マシンの基本性能の把握
2) 大規模問題のシミュレーション時間の予測
3) マシン毎のプログラム性能の比較
4) 最適な入出力パターンの検討
5) アプリケーションと消費電力の関係

4. 展開方法

SS 研ホームページで公開する。追加プログラムを募集し、随時更新する。

5. ベンチマーク項目

以下の 4 項目について評価できるものとする。

1) 基本性能（CPU，メモリ，ネットワークに関する性能）
2) プログラム性能（実際のシミュレーションプログラムの性能）
3) I/O 性能（ファイル入出力の性能）
4) 消費電力

6. ベンチマークプログラムの作成指針

ベンチマークの作成にあたり、以下の指針を設定する。

・基本性能および I/O 性能について
　1) 公開されている BMT を利用する
・プログラム性能について
　1) メモリサイズ（問題規模）を簡単に変更できること
　2) 入力データは機種依存性の無いテキスト形式とする
　3) コンパイラ依存性が無いこと
　4) 性能指標の考え方に一貫性があること

7. 公開ベンチマークの利用

基本性能および I/O 性能の測定には、以下の**表** 1 で示す一般に公開されているベンチマークを使用する。

表 1. 公開ベンチマーク

項目	アプリ名	公開 URL
メモリバンド幅	STREAM	http://www.cs.virginia.edu/stream
ネットワーク性能	IMB	http://software.intel.com/en-us/articles/intel-mpi-benchmarks
I/O 性能	IOR	http://ior-sio.sourceforge.net
並列特性	NAS-parallel	http://www.nas.nasa.gov/Resources/Software/npb.html

8. 性能指標の考え方

システムエンジニアの観点ではなく、ユーザの観点で評価する。ユーザにとって役立つ情報として、実行時間を念頭に置き、以下の 4 項目を把握できるベンチマークを作成する。

1) どの処理に時間を要しているか
2) 実行時間に占める、演算・通信・入出力の割合
3) 大規模問題の実行時間予測
4) ハイブリッド並列化の有効性

9. 測定項目と評価値一覧

マシンの基礎性能測定から消費電力までを網羅する総合的なベンチマークを準備し、利用者が希望する項目を選択する方式とする。ベンチマークの選択項目を**表 2**2 に示す。また、ベンチマーク項目の内、プログラム性能に関しては、さらに詳細な測定項目があり、プログラムの挙動を把握できるようにする。プログラム性能に関するプログラム挙動把握のための測定項目を表 3 に示す。

表 2. ベンチマーク選択項目

項目	内容	評価値
基礎性能	各種の公開ベンチマークを利用する ・メモリバンド幅（STREAM） ・通信バンド幅（IMB） ・通信レイテンシ（IMB） ・MPI と OpenMP の比較 （NAS-parallel-benchmark を使用し、MPI 版と OpenMP 版を比較する）	バンド幅（byte/s） 遅延時間（秒） 倍率
IO 性能	入出力性能を計測する ・入出力バンド幅	IO バンド幅（byte/s）
プログラム性能	ユーザプログラムの総合性能を評価する ・演算性能（ハイブリッドを含む） ・通信性能（ハイブリッドを含む） ・IO 性能	実行時間（秒） 演算性能（FLOPS） バンド幅（byte/s）
消費電力	消費電力を評価する ・プログラム実行時の消費電力を検出	演算当りの消費電力(w/flop) 時間当りの消費電力(w/hour)

表 3. プログラム挙動把握のための測定項目

項目	内容
コスト分布 (主要処理の実行時間)	実行時間の長い処理をルーチン単位で表示する。 どのルーチンに時間を要しているかを把握できる。
演算・転送・入出力の割合	全実行時間に占める、演算の割合と転送の割合と入出力の割合を表示する。
大規模問題を想定した場合の実行時間予測	小規模問題を実際に実行して、大規模問題の実行時間を予測する。
ハイブリッドの有効性	flat-MPI とスレッド+MPI で実行時間を比較する。

10. 各測定項目の課題と対策案

ベンチマークを作成するにあたり、解決しなければならない課題がある。**表** 4 に、課題と対策案をまとめる。今後、具体的な対処法を決定する必要がある。

表 4. ベンチマーク測定項目の課題と対策案

項目	課題	対策案
基礎性能 IO 性能	一般に公開されているベンチマークを使用するため、以下 3 点の課題がある。 1) インストール方法 2) 実行方法 3) 出力結果の見方	インストールと実行に関しては、実際のコマンド例を示した解説書を作成する。 出力結果の見方として、各出力項目の意味を解説した資料を作成する。
プログラム性能	1) 主要処理の計算内容を解説し、入力データを用意する必要がある。 2) コンパイラ依存性を除去するため、言語仕様に準拠するよう修正が必要。 3) 各ルーチンに実行時間測定用の時間計測ルーチンを挿入する必要がある。 4) ハイブリッド化を実施する必要がある。	1)と 2)に関しては、プログラム提供者に対応して頂く。 3)と 4)に関しては、サブ WG のメンバーで対応する。
消費電力	消費電力の計測方法を調査する必要がある。 また、測定対象として、CPU だけで良いか、ディスクやファンの電力も含めるか等、さらに検討が必要。	有識者に問い合わせ、計測対象と計測方法を詰める。

4.3. ベンチマーク実行例

ベンチマーク構成のイメージを**図** 1 に示す。利用者は、入力パラメータを設定し、制御プログラムを起動する形式を想定している。制御プログラムは、入力データで指定したベンチマークを自動的に起動するように制御するためのプログラムである。

次節で、プログラム性能に関する、ベンチメークの実行イメージを示す。

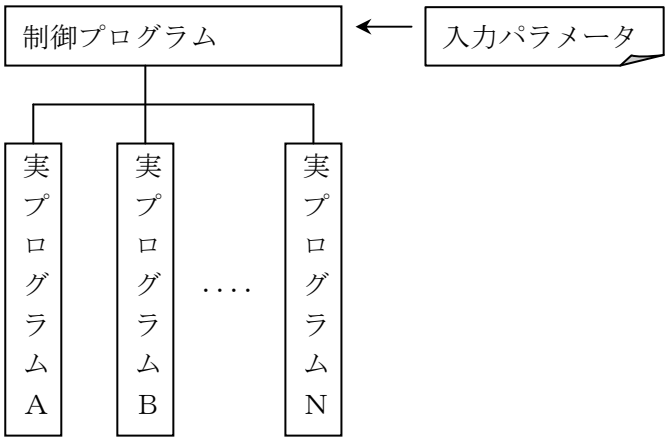

図 1. ベンチマーク構成イメージ

1. 実行例

入力データで、ab_initio_mo を選択した場合の BMT 出力例を**図** 2 に示す。入力パラメータと主要処理の実行時間が出力される。

```
----- ab_initio_mo method -----
-----      [ c6h6 ]       -----
--------------------------------
---- parameter information ----
 number of atom = 12
 number of electron = 42
 number of base function = 36
 threshold in SCF = 1.000000000000000e-05

---- execution information ----
 SCF iteration = 67

------ Time information ------
          normalize: 0.970e-04 (sec)
       one_elec_int: 0.388e-01 (sec)
       two_elec_int: 0.111e+02 (sec)
                SCF: 0.210e+01 (sec)
           file I/O: 0.271e+01 (sec)
 Total execute time: 0.159e+02 (sec)
 -------------------------------
```

図 2. ab_initio_mo 選択時の実行結果例

2. 活用例

1) ベンチマークの実行結果からプログラムの特性を把握する。

図 2 から、シミュレーション対象物質がベンゼン(c6h6)であり、基底関数は 36 という情報が得られる。この場合の主要計算は 2 電子積分であり約 11 秒、ファイル出力も約 3 秒要していることが分かる。

さらに、表 5 に示すような、プログラムの特性を解説書に記述し、入力パラメータと演算数の関係を明らかにすることで、実行時間を予測することも可能となる。

表 1. ab_initio_mo の処理概要

プログラム名	処理内容	評価値	備考
ab_initio_mo	[分子軌道法プログラム] 1 電子積分，2 電子積分，SCF 計算を行う。 2 電子積分で扱う積分タイプは、7 種類である。 (ssss , psss , ppss , psps , pspp , ppps , pppp) 1 電子積分の演算量は、$O(N^2)$となり、 2 電子積分の演算量は、$O(N^4)$となる。 SCF 計算は、$O(N^3)$である。 （N=基底関数の数）	実行時間	言語：Fortran77 行数：3565 step 並列化：無(逐次)

上記の解説を元に、以下の情報を得ることができる。

2) 大規模問題の実行時間予測

主要計算である 2 電子積分は、基底関数の 4 乗オーダの演算となる。よって、

[実行時間 ≒ 11(秒)×(想定する基底関数の個数/36)^4]

と予測することができる。

上記のように、プログラムの特性を把握でき、かつ、異なる問題に対する予測もできる点が、一般的なベンチマークとは異なり、ユーザの視点に立ったものとなっている。

以上