PMlibを用いた計算性能測定と性能可視化手法

三上 和徳[1,a)]　小野 謙二[2]

概要：オープンソースライブラリ PMlib を用いた HPC アプリケーションの性能評価手法について報告する。HPC アプリケーションの計算性能において、コンピュータシステムの仕様上の最大性能とアプリが実際に達成する実行性能との違いが頻繁に指摘される。この違いを議論する場合には、演算器の並列性やメモリ階層における局所性確保などのハードウエアの動作特性を中心に評価する視点と、アプリケーションがソースプログラムレベルで要求する数値計算上の計算量とコンピュータシステムが実際に実行する命令に基づいた計算量との違いを評価する視点の双方が必要である。
PMlib は数値計算上の計算量を明示的に測定する機能と、HWPC が記録する計算量を測定する機能とを有し、両者の差を定量的に評価することを可能とする。また HWPC 測定時には内部で PAPI 低レベル API を利用し、一般に選択と解釈が容易ではない各種ハードウエアイベント統計情報をカテゴリ分けして HPC アプリ利用者が評価しやすい情報として選択出力する。出力情報としてはアプリ実行中に蓄積された統計情報を時間平均化した標準レポートに加え、経過時間軸に沿った動的な情報の出力も可能であり、計算性能の時系列挙動を可視化する Web ブラウザパッケージとの連携利用が可能な構成となっている。本報告では PMlib を用いた HPC アプリの性能評価手法を説明し、Intel Xeon および富士通 FX100 上での評価事例を紹介する。

1. はじめに

計算科学アプリケーションの開発および利用の各局面において、利用する HPC システム上で性能評価作業を実施することが頻繁に行われる。これは主に、アプリケーションの性能特性を把握した後、ソフトウエア上の最適化を実施して高速処理を実現するための可能性を探るために行われる。このような性能評価作業が頻繁に実施される背景として、HPC システムの仕様上の最大性能とアプリケーションが実際に達成する実行性能との差が、アプリの種類によっては非常に大きいことがある。

最大性能と実行性能との違いに関して演算器の並列性やメモリおよびキャッシュ階層構成における局所性などのハードウエアの動作特性と関連づけられて多くの研究が行われている。

一方、アプリケーションがソースプログラムレベルで要求する数値計算上の計算量と、コンピュータシステムが実際に実行する命令を　HWPC（ハードウエア性能カウンタ）で測定した計算量との間でもしばしば大きな乖離がある。この事は性能評価において重要な問題点として考慮されなければならない。

例えば除算計算などはコンピュータシステムが実際に実行する命令ベースでの（HWPC による見かけ上の）実行性能が非常に高く表示されるが、数値計算上（ソースプログラムで計上した）の計算量に基づく性能は一般に低い。

PMlib は数値計算上の計算量を明示的に測定する機能と、HWPC が記録する計算量を測定する機能とを有し、両者の差を定量的に評価することを可能とする。HWPC 測定には内部で PAPI 低レベル API を利用し、一般に選択と解釈が容易ではない各種ハードウエアイベント統計情報をカテゴリ分けして HPC アプリ利用者が評価しやすい情報として選択出力する。出力情報としてはアプリ実行中に蓄積された統計情報を時間平均化した標準レポートに加え、経過時間軸に沿った動的な情報の出力も可能であり、計算性能の時系列挙動を可視化する Web ブラウザパッケージとの連携利用が可能な構成となっている。本報告では PMlib を用いた HPC アプリの性能評価手法を説明し、Intel Xeon および富士通 FX100 上での評価事例を紹介する。研究の目的

1.1 計算科学的観点での性能

計算量という用語を用いる場合、計算科学アプリケーションの開発者は Fortran 言語や C++言語などで記述されたソースプログラム上で計上される数値計算量をアプリ

1　理化学研究所 計算科学研究機構
2　九州大学 情報基盤研究開発センター
a)　kazunori.mikami@riken.jp

ケーションの計算量として認識することが多い。計算科学的観点での計算性能とはこの計算量を計算経過時間で割った値として定義される。

Perf = ... みたいな式にする必要あり？

HPC アプリケーションにおいて所与の計算を効果的に達成するために計算時間の短縮をめざす場合、必要な計算量を削減するアルゴリズムの採用を試みることが多いが、そのような場合は主として計算科学的観点での計算量が対象とされてきた。

1.2 システム評価的観点での性能

アプリケーションをコンピュータシステム上で実行する段階では、ソースプログラムを言語コンパイラを通して生成した実行プログラムの機械語命令列としてスケジューリング処理することになる。

ソースプログラムでは単純な計算式であっても、実際に実行されるのは複数の命令列であり、演算（例えば浮動小数点演算）量という基準で測定した場合、計算科学的観点での計算量と比較して多くなる。

計算科学的観点で計算評価を行う場合、ソースプログラムで記述される平方根・三角関数といった数学関数ではそれぞれに計算の「重たさ」が異なることは良く認識されているが、

四則演算や最小最大などの基本的な演算の「重たさ」も相当に異なることはしばしば看過されがちである。

が実際に実行する命令ベースでの（HWPC による見かけ上の）実行性能が非常に高く表示されるが、数値計算上（ソースプログラムで計上した）の計算量に基づく性能は一般に低い。

HWPC を基にした実行性能 (FLOPS)、実効性能（特定のシステム・アーキテクチャ・言語処理系 SW での性能に主眼がおかれる

システム評価的観点で計算性能をとらえた場合、

同一プログラムの計算性能あるいは実行性能がの基準が観点によって異なる

1.3 次行こう！

性能情報の可視化　TRAiL

静的な統計情報の可視化ジョブの経過時間で平均化された値

動的な性能挙動の可視化ジョブの経過時刻に沿った過渡的な挙動プロセス毎の過渡的な挙動プロセス間の計算負荷バランスの挙動

本報告ではまず前段の、、、についていくつかの評価事例を通して観点の違いによって評価量の測定値が大きく異なることを示したい。

性能測定値を時刻歴可視化することによって得られる、性能挙動の理解に与える効果については、後続報告を行う計画である。

オープンソースライブラリ PMlib [2]。オープンソースライブラリ PMlib [1]。

{
（１）PMlib とはどのようなツールか
・オープーンソース、C++クラスライブラリ、言語インタフェイス
・計算量の測定方法
・・ソースレベルでの明示的な指定、
・・・組み込み時の他ツールとの連絡（橋本ツール例）
・・HWPC による採取
・・・HWPC のサポート状況
・測定した計算量の蓄積方法
・・逐次合算
・・時刻歴記録
・出力
・・テキストレポート
・・汎用トレースフォーマット（OTF）出力
}

2. PMlib の位置づけと関連研究

性能評価に関するこれまでの研究と PMlib の位置づけ

{
アプリケーションの計算性能モニター用のライブラリ
オープンソース　http://avr-aics-riken.github.io/PMlib
主な用途
タイマー、計算負荷の概要把握、HWPC への簡易アクセスなど
利用方法
アプリのソース中に測定区間を指定。終了時に統計情報を出力
利用のモード
アプリに常時組み込み、プロダクション利用することを想定
性能測定・性能モデル化の支援にも期待
C++と Fortran に対応する API
特徴
インストールが簡単。講習会での実績平均 10 分
テキストレポートを基本とするコンパクトなツール
利用のための学習バリアが低い
}

Linpack の FLOPS はソースで直接定義

rooflinemodel

OS やコンパイラなどの処理系に依存

オープンソース性能統計ツール類　Gprof: 簡易機能、コンパイラに制約 Scalasca: 高機能、Score-P 共通インフラ PAPI : HWPC へのアクセス他 : 多数 ベンダーツール

X86 系 Intel PGI Sparc 系富士通 Oracle Power 系

測定可能な情報

既存ツールはほぼ全てがサンプリングあるいはトレーシングによる測定で、性能の根拠は HWPC 情報を利用している

簡単さ

ベンダー製品の性能計測・統計ツール○豊富な機能が統合化されたインタフェイス○パッケージの完成度が高く、インストールが容易○ 詳しいドキュメント、ベンダーによるサポート、安心感△一般に高価格で相当の習熟期間が必要△選択できるツールがシステム毎に制限される

オープンソースの性能計測・統計ツール○各ツール毎に特徴・機能が明確○様々なシステムへの移植可能○価格の心配が不要△ユーザーインタフェイスが個性的△ものによりインストール・利用がそれなりに大変

3. PMlib を用いた性能測定方法

（2）PMlib でどのよう情報を得るか・汎用タイマーとしての機能・Fortran/C++で表記される数値計算上での演算量・コンピュータが実行する命令ベースでの HW 演算量

発表スライドでは CCA/EBT、OTF、HWPC(PAPI) を説明。そこに力点をおくとよい

（3）PMlib でどのような分析ができるか・演算の種類毎に処理の「重たさ」を評価・・浮動小数点四則演算、平方根や三角関数などの基本計算・・・ー＞コンピュータが見かけ上発揮する実行性能と、実際の数値計算性能との定量的な比較ができる・・・ー＞アプリのコーディングの指針に使える・HWPC ベースでのアーキテクチャの有効利用状況・・使いやすい用にグループ化した統計出力・・SSE/AVX 系命令の実行比率・・・ー＞ SIMD 機構の有効利用の確認・・キャッシュヒット率・・・ー＞コンパイラがどれだけ賢くコンピュータの能力を引き出しているか評価ができる・・・ー＞コンパイラオプションの選択、指示行の指定などで、どのような効果が現れるか

4. PMlib を用いた性能測定と分析例

・基本演算カーネル（＋、＊、＊＋、%、SQRT など）・ミニアプリ１例（NTChem かな）・ピーク性能に対する実効性能比を裏付けてみる

Skylake の例で示すといい

ルーフラインモデルを示し、実測との比較を見るとよい

以下の Listing1 は PMlib API を利用する Fortran ソースプログラムの例である。

Listing 1

```
1  program main
2  call f_pm_initialize (nWatch)
3  call f_pm_setproperties ("Koko!" icalc, iexcl)
4  call f_pm_start ("Koko!")
5  call mykernel (msize,n,a,b,c)
6  call f_pm_stop ("Koko!", fops, ncall)
7  call f_pm_print ("", isort)
8  call f_pm_printdetail ("", ilegend, isort)
9  end
```

4.1 基本的な演算性能の評価

4.2 STREAM 性能の評価

STREAM ベンチマーク [3] はコンピュータシステムのメモリバンド幅を測定するためのツールとして広く用いられている。STREAM は変数配列の積和算（TRIAD）など、メモリ read/write 処理が主要な負荷となる計算式の性能をアプリケーションプログラムレベルで測定出力する。したがってその結果は「計算科学的観点」で評価された性能である。この STREAM プログラムを「システム評価的観点」で評価するとかなり様相が異なって来る。

まず STREAM を Skylake サーバ上で実行した場合の出力結果を示す。Intel コンパイラのデフォルトオプションを用いている。STREAM Fortran プログラム OpenMP スレッド並列版を 1CPU 上で 8 スレッド実行した場合、20 スレッド実行した場合について示す。

Skylake サーバされた HWPC ベースでの PMlib を用いてメモリ階層での実測イベントベースによるデータ移動の状況を示す。図

測定に用いたプラットフォーム Intel Skylake CPU 搭載サーバの構成と性能諸元表を以下に示す。

富士通 prime HPC FX100 構成と性能諸元表を以下に示す。

Intel コンパイラにはオプションが多数あり、中でも特にメモリバンド幅に関係が深いと思われる以下のオプションを組み合わせて比較した結果を図 1 に示す。

この図は仮置きで Ivybridge の結果。Skylake の結果と差し替えること。

STREAM:Intel compiler で Cache Eviction Level オプションの指定効果はほとんどなかった

5. 謝辞

京補助金についての謝辞

参考文献

[1] Ono Kenji. Pmlib 開発リポジトリ.
[2] Ono Kenji. Pmlib 公開 web ページ.
[3] John D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, December 1995.

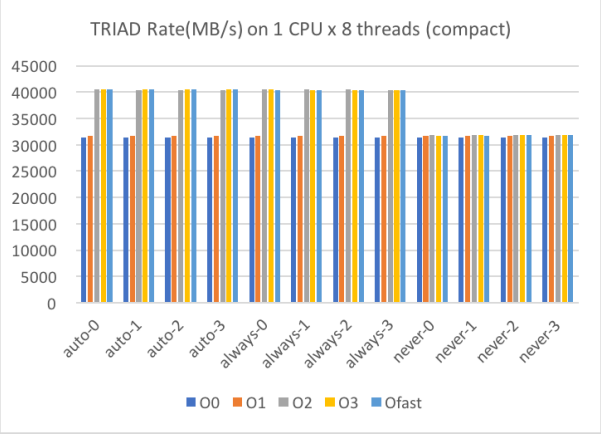

図 1 stream-ivy-compact-1cpux8