システム評価 東工大 進捗報告

2013/07/09

野村 @ 東工大

目次

- 東工大進捗報告 ～アプリのモデリングに向けて
  - ツール整備状況
  - リポジトリ開発状況
- 性能評価ツールはどう役立つのか
  - GT5Dの事例

ツール整備状況

- 提供いただいたアプリでのツール動作状況の確認
  - 京およびTSUBAMEでScalasca, VampirTraceが適用できることの確認
  - ある程度OK: GT5D, MARBLE, para-TCCI
  - テスト中: pSpatiocyte, FFVC, STATE
  - 以下、続々と着手予定
  - 詳細は本発表の後半で

リポジトリ整備状況

- ベンチマーク結果と実行条件の格納庫
- 開発中
  - 8月末～9月頭に完成予定
  - 現時点でデータの格納機能・検索機能のみ
- デモ

性能評価ツールは アプリ開発でどう役立つのか

- 性能評価ツール(Scalasca, Vampir)をGT5Dに適用して性能モデリングへつなげるデモ
  - 性能評価ツールはモデリングだけではなく、チューニングにも応用できます
  - プロファイラは既存アーキ上での計算量・メモリアクセス量などの計測に使えます

GT5Dカーネルごとの実行時間 (Strong Scaling)

- GT5D自身の実行時間取得機能による表示

[ms]

subroutine costs(msec)	TSUBAME2.0 16MPI_6SMP	TSUBAME2.0 64MPI_6SMP	FX10 Kyushu 16MPI_8SMP	FX10 Kyushu 64MPI_8SMP
init	9,123.09	7,945.24	12,166.82	8,660.45
dn	8.62	2.21	4.56	1.06
field	41.55	26.39	46.56	27.06
drift	19.01	4.59	13.19	3.98
l4dnl	82.29	19.87	184.08	34.22
l4dl	887.29	218.03	1,629.20	186.62
rk	1,802.99	454.89	1,014.74	272.93
bc	262.54	147.40	214.95	94.09
col	615.04	154.97	539.36	134.41
asrc	0.38	0.10	0.17	0.04
out	0.00	0.00	0.00	0.00
1step	3,719.72	1,028.46	3,646.80	754.42
end	0.00	0.00	0.00	0.00

GT5Dカーネルごとの実行時間 (Strong Scaling)

- GT5D自身の実行時間取得機能による表示

分割数 4x4x1x1x4

分割数 2x2x1x1x4

[ms]

subroutine costs(msec)	TSUBAME2.0 16MPI_6SMP	TSUBAME2.0 64MPI_6SMP	FX10 Kyushu 16MPI_8SMP	FX10 Kyushu 64MPI_8SMP
init		7,945.24	12,166.82	8,660.45
dn			4.56	1.06
field	41.55	26.39	46.56	27.06
drift	19.01	4.59	13.19	3.98
l4dnl	82.29	19.87	184.08	34.22
l4dl	887.29	218.03	1,629.20	186.62
rk	1,802.99	454.89	1,014.74	272.93
bc		147.40	214.95	94.09
col				41
asrc				04
out				00
1step				42
end				00

- カーネルごとにマシンにとっての『得手不得手』がある
- Super-Linearが起こっている
  →これを表現する性能モデルを作る

GT5Dカーネルごとの実行時間 (Strong Scaling)

- GT5D自身の実行時間取得機能による表示

[ms]

subroutine costs(msec)	TSUBAME2.0 16MPI_6SMP	TSUBAME2.0 64MPI_6SMP	FX10 Kyushu 16MPI_8SMP	FX10 Kyushu 64MPI_8SMP
init	9,123.09	7,945.24	12,166.82	8,660.45
dn	8.62	2.21	4.56	1.06
field	41.55	26.39	46.56	27.06
drift	19.01	4.59	13.19	3.98
l4dnl	82.29	19.87	184.08	34.22
l4dl	887.29	218.03	1,629.20	186.62
rk	1,802.99	454.89	1,014.74	272.93
bc	262.54	147.40	214.95	94.09
col	615.04	154.97	539.36	134.41
asrc	0.38	0.10	0.17	0.04
out				0.00
1step				754.42
end				0.00

でも、これってGT5Dの機能じゃないの?

Scalasca(プロファイラ)による表示

- GT5Dのように計測用のコードが無くても実行時間の分布を見ることができる
- ハードウェアカウンタによるより詳細な情報も取れる(デモではFP演算数)
  - デモ: para-TCCI, MARBLE
- 京でもx86_64マシンでも動作
  - アーキテクチャ間の比較ができる

Vampir(トレーサ)による表示

- 時系列で何が起こっているのかが分かる
  - Load-imbalanceでどのプロセスが眠っているか
  - それぞれの関数実行中のHWカウンタの推移
  - カーネルの実行時間の推移
  - 通信の関係(Communication Matrix)
    - ただし現状p2p通信のみなのでCollectiveは別途見る必要あり
  - 京詳細プロファイラ並の情報が範囲指定せずに得られる
  - デモ: GT5D (TSUBAMEのFatノードを確保できたら)
    - トレースの解析にはメモリを大量に必要とする
    - プロファイラと適材適所で

性能モデルへの作り方

- Aspenによるモデル例(3DFFT)
  - control main {
    localFFT // in x
    shuffle
    exchange
    localFFT // in y
    shuffle
    exchange
    localFFT // in z
    }
  - kernel localFFT {
    exposes parallelism [n^2]
    requires flops [5 * n * log2(n)] as dp, simd
    requires loads [a * n * max(1, log(n)/log(Z)) * wordSize] from fftVolume
    }

コントロールフロー

各Kernelについて、consistentになるように中身を埋めていけばよい

各Kernelの性能モデル

GT5Dの性能モデル例(イメージ)

- kernel l4 {
  exposes parallelism[x * y * b]
  requires flops[XXXXXX] as dp, simd
  requires loads[YYYYYY] from ZZZZZZ
  }
- kernel rk {
  exposes parallelism[x * y * b]
  requires flops[xxxxxx] as dp, simd
  requires loads[yyyyyy] from zzzzzzz
  }

アプリ開発者の皆様へのお願い(?)

- 性能解析ツールを使うときには、ソースコードをコンパイラ以外のプログラムで処理します
  - 解析用ルーチンの埋め込み
  - 解析範囲(OMPループ等)の切り出し
- また、複数アーキテクチャで動かすので、富士通コンパイラ以外でもコンパイルします
- これらのツールに通す過程でのプログラム改変について、ご相談させていただくこともあるかと思いますが、その際はよろしくお願いいたします。
  - 文法として正しいプログラム、(コンパイラにとって)曖昧性の少ないプログラムを書いていただけると、解析ツールの誤動作が少なくなります

引っかかった実例

- プリプロセッサの影響で$OMP&分の間に空行が出来てしまい、こけた
  - !$OMP& PRIVATE(dxi,dzi)
    !$OMP& PRIVATE(dx2i,dz2i)
    #ifdef AAA
    !$OMP& PRIVATE(fl_wk)
    #endif
    !$OMP& SHARED(if5d,bbs,rg,vx,vy,vz)

    ←ここがpreprocessの結果空行化して、前後のOMP節が切れる→変換ミス
- 多重ループの終わりを1つのcontinue文でまとめたところにOpenMP指示行が入ることで変換ミスが起こった
  - do 10 i1 = 0,kk1
    !$omp parallel do
    do 10 i32 = 0,n3*n2-1
    …
    10 continue

    ←ここの実質2つのcontinueの間に計測用コードを入れたい→変換ミス

今後の予定

- フルアプリでのツール動作確認
  - 随時確認中
- 性能モデルの構築
  - ミニアプリを利用して先行事例を作りたい
- 性能評価リポジトリの作成