OpenMP に関する情報

(第 2 版)

平成 25 年 5 月 28 日

§1 依頼内容

★プログラムの概要：

C++、C または FORTRAN で書かれた数値計算プログラム。MPI を使って大型の並列計算機で動くようになっているが、スレッド並列 (SMP) はコンパイラの自動並列化に任せている。

★助言、提案の欲しい問題：

OpenMP を使って現在の計算コードの効率を上げたいが、OpenMP についてほとんど知らないので、これらを使うにあたってお薦めの資料、文献などあれば教えてください。

★使用する計算機（予定も含む）：

Linux-PC, 共同利用されている大型計算機（SR16000, BlueGene, FX10, HA-PACS, SX8, 京など）

★用いる言語：

FORTRAN, C, C++, MPI, OpenMP

§2 コンパイラの診断情報を利用

コンパイラの自動並列化 (スレッド並列) が使える環境では、その機能を利用して、OpenMP 指示行を加えずともスレッド並列化 (MPI と組み合わせればハイブリッド並列化) されたコードを生成することができます。さらに、コンパイル時にコンパイラの診断結果を出力させることにより、ソースコードのどの部分が自動並列化可能であったかを確認することができます。このコンパイル時のログ出力を参考にして、コンパイラが自動並列化可能と診断しなかったけれども並列化可能な部分を見つけ出し、そこへ適切な OpenMP 指示行を挿入することにより、OpenMP 初心者でも比較的容易に OpenMP 指示行を加えていく作業が行えます。アルゴリズムの検討や、実行時の性能情報を取得するツールなどと併用することにより、並列化による性能向上が高いと見込まれる部分から順次 OpenMP 指示行を加えていくことにより、効率よく作業が行えるものと思います。

コンパイル時に診断メッセージを出力するオプションは、例えば、以下のようなものです。

- SR16000(KEK, YITP): `-loglist`
- BleuGene/Q(KEK): `-qsource -qlist`

§3 Parallel Program Generator

スレッド並列 (SMP) のコンパイラの自動並列化機能を利用し、FORTRAN ソースに OpenMP 指示行を挿入する製品があるようです。日立の製品「Parallel Program Generator」。

`http://www.hitachi.co.jp/Prod/comp/soft1/HPC/info/price/index.html`

§4 参考文献・資料

Web から容易に入手できるものを中心に、以下にリストを示します [1–9]。また、ユーザー支援による OpenMP に関する資料・報告書などは、「ユーザー支援事例報告」、「高性能計算の扉」 [10] や、「メンバーの計算経験」に今後情報が載せられていくと思います (例えば文献 [11, 12])。

参考文献

[1] CCS HPC サマーセミナー 2011 [http://www.ccs.tsukuba.ac.jp/workshop/HPCseminar/2011/]

[2] OpenMP 仕様書の日本語訳 [http://www.openmp.org/mp-documents/OpenMP30spec-ja.pdf]

[3] OpenMP.org [http://openmp.org/wp/]

[4] 書籍転載文法からはじめるプログラミング言語 Microsoft Visual C++入門並列処理を行うための基礎知識（Visual C++）――第 13 章 並列処理　マルチスレッドプログラミング（後編）―― [http://www.atmarkit.co.jp/fdotnet/bookpreview/bunpouvcpp_1302/bunpouvcpp_1302_04.html]

[5] NAG による OpenMP 入門ドキュメント [http://www.nag-j.co.jp/openMP/index.htm]

[6] OpenMP [https://computing.llnl.gov/tutorials/openMP/]

[7] POSIX Threads Programming [https://computing.llnl.gov/tutorials/pthreads/]

[8] MPI Documents [http://www.mpi-forum.org/docs/docs.html]

[9] Clay Breshears 著、千住 治郎 訳、「並行コンピューティング技法―― 実践マルチコア/マルチスレッドプログラミング」、O'Reilly Japan (2009).

[10] 高性能計算の扉 [http://www.jicfus.jp/field5/jp/promotion/hpcdoor/]

[11] 平松尚志、「古典場のシミュレーションにおける OpenMP の実装例」　、メンバーの計算経験、ユーザー支援、計算科学の推進、HPCI 戦略プログラム分野 5 [http://www.jicfus.jp/field5/jp/promotion/user/keiken/#2011-04]

[12] 「弾性球間の付着力を考慮した N 体コードへの OpenMP の実装」　、ユーザー支援事例報告、ユーザー支援、計算科学の推進、HPCI 戦略プログラム分野 5、[http://www.jicfus.jp/field5/jp/promotion/user/houkoku/#2012-01]

[13] 根村英克、「ハイペロン相互作用を研究するための格子 QCD による 4 点相関関数を計算する」、メンバーの計算経験、ユーザー支援、計算科学の推進、HPCI 戦略プログラム分野 5 [http://www.jicfus.jp/field5/jp/promotion/user/keiken/#2012-06]

§A ハイブリッド並列化による速度改善の一例

この付録では、すでに MPI 化されている格子 QCD の計算コード[1] に、さらに OpenMP の指示行を加えて計算速度の改善を試みた実例を紹介します。

§A-1 MPI 通信のマルチスレッド化について

MPI によって並列化されているコードでは、実行開始の冒頭部分で、

```
int MPI_Init(int *argc, char ***argv)
```

を呼び出していますが、さらにマルチスレッド化を行いたい場合は、この部分を

[1] 格子 QCD 共通コード (Bridge++) 上でハイペロン相互作用を研究するための 4 点相関関数の計算を実装したもの。 [13]

```
int MPI_Init_thread(int *argc, char ***argv, int required, int *provided)
```

に変更します。3 番目の引数は、マルチスレッド環境での MPI 通信をどのように行いたいかを指定します。具体的に指定できる値は、MPI Documents [8] によると、以下のような値が指定できるようです。(詳しくは MPI の仕様書を参照してください)

- `MPI_THREAD_SINGLE`
  Only one thread will execute.
- `MPI_THREAD_FUNNELED`
  The process may be multi-threaded, but only the main thread will make MPI calls (all MPI calls are funneled to the main thread).
- `MPI_THREAD_SERIALIZED`
  The process may be multi-threaded, and multiple threads may make MPI calls, but only one at a time: MPI calls are not made concurrently from two distinct threads (all MPI calls are serialized).
- `MPI_THREAD_MULTIPLE`
  Multiple threads may call MPI, with no restrictions.

4 番目の引数は、3 番目の引数で要求したマルチスレッド環境での MPI 通信レベルが、実際に実行される計算機上で利用可能かどうかを示す返り値として使われます。例えば、`MPI_THREAD_MULTIPLE` が利用可能な場合には、3 番目の引数に `MPI_THREAD_MULTIPLE` を指定すると、`MPI_Init_thread` を呼び出した後で 4 番目の引数には `MPI_THREAD_MULTIPLE` が返されますが、もしその計算機の実行環境が対応していない場合には、4 番目の引数には `MPI_THREAD_MULTIPLE` 以外のいずれか対応している中で最大のレベルの値が返されます。従って、MPI 並列化されているコードに、さらに OpenMP 指示行を追加してマルチスレッド化を行いたい場合には、まずそのコードを走らせる予定の計算機上で、上に挙げたどのレベルまでが利用可能かを確認した上で、MPI 通信を行っている部分をどのようにマルチスレッド化するかを検討する必要があります。

§A-2 速度改善の一例

高エネルギー加速器研究機構にある BlueGene/Q を利用した場合の例を以下に示します。BlueGene/Q では `MPI_THREAD_MULTIPLE` が利用可能でしたので、マルチスレッドでの MPI 通信のレベルにはこれを使用しています[2]。

この報告書で示す計算速度の改善例では、単純に OpenMP 指示行を入れるだけでなく、マルチスレッド時に性能が上がるようにもとのコードを書き替える作業も行いました。何通りかのスレッド並列化のさせ方を試した中で、いまのところ最も成績が良さそうな場合をここでは紹介しますが、さらに改良の余地があるかもしれません。OpenMP の指示行を入れてマルチスレッド化を進めるにあたって、文献 [9] が参考になりました。

今回は、BlueGene/Q の 32 ノードジョブクラスを使用し、格子サイズ $16^3 \times 32$ においてハイペロンポテンシャルを研究するための Nambu-Bethe-Salpeter 波動関数を計算する時間を、MPI 並

[2] 今回は、マルチスレッド時に通信と演算のオーバーラップによって全体の計算速度が改善されることを期待して、`MPI_THREAD_MULTIPLE` を選択し、MPI 通信のコミュニケータもマルチスレッド化しました。

列およびスレッド並列の並列数を変えながら計算時間を測定しました。今回測定した計算時間は、大まかに、既に解かれているクォーク伝搬関数をインプットとして、単一のバリオン相関関数を計算しその高速フーリエ変換 (FFT) を行う部分 (step-1) と、4 点相関関数を構成するために Wick の縮約を構成する部分 (step-2) からなります。以下で示す計算時間は、step-1、step-2 について、同じジョブを 4 回実行した平均値を示しています。BlueGene/Q では各ノードは 16 コアを持っていて、各コアにつき 4-way までのマルチスレッドが利用可能になっています。MPI のプロセス数とスレッド数は、ジョブスクリプトファイル中で `tasks_per_node` と `OMP_NUM_THREADS` を設定することで指定できます。そこで、まずは表 1 に、`tasks_per_node` を 16 に固定し、`OMP_NUM_THREADS` を $1, 2, 4$ と変えてみた場合に、実行時間がどのように変わるかを示します。スレッド数を増やしていくことによって、実行時間が短縮されていることがわかります。

BlueGene/Q では、ノード当りの最大のスレッド数は $16 \times 4 = 64$ です。そこで、ノード当りの全スレッド数を 64 に固定した上で、`tasks_per_node` と `OMP_NUM_THREADS` の値を、様々に変えた場合の実行時間を表 2 に示します。表の中で最も左側の「64×1」の列が、いわゆるフラット MPI の場合です。今回の例では、フラット MPI よりも MPI プロセス数をコアの数と同じ 16 にして、各プロセス当り 4 スレッドにした場合が最も計算時間が短いという結果になりました。

このように、既に MPI 並列化されているコードに、さらに OpenMP の指示行を入れてマルチスレッド化（ハイブリッド並列化）することにより、大型計算機上での実行時間を短縮できることが確認できました。また、なるべく性能を上げるためには、OpenMP 指示行を単純に機械的に挿入していくだけでなく、より積極的なコードの書き替えが有効なようです。

`OMP_NUM_THREADS`	1	2	4
実行時間 (分:秒)			
Step-1	0:38	0:22	0:14
Step-2	1:09	0:46	0:34
合計	1:47	1:08	0:48

表 1: MPI プロセス数を 16 に固定して、各プロセス当りのスレッド数を $1, 2, 4$ と変えた場合の実行時間。

[`tasks_per_node`]×[`OMP_NUM_THREADS`]	64×1	32×2	16×4	8×8	4×16	2×32	1×64
実行時間 (分:秒)							
Step-1	0:18	0:21	0:14	0:12	0:10	0:10	0:10
Step-2	0:53	0:39	0:34	0:44	1:08	1:39	3:01
合計	1:11	1:00	0:48	0:56	1:18	1:49	3:11

表 2: ノード当りの全スレッド数を 64 に固定し、MPI プロセス数とプロセス当りのスレッド数を様々に変えた場合の実行時間。

、