SSTC
High Performance and Productivity
スケーラブルシステムズ株式会社

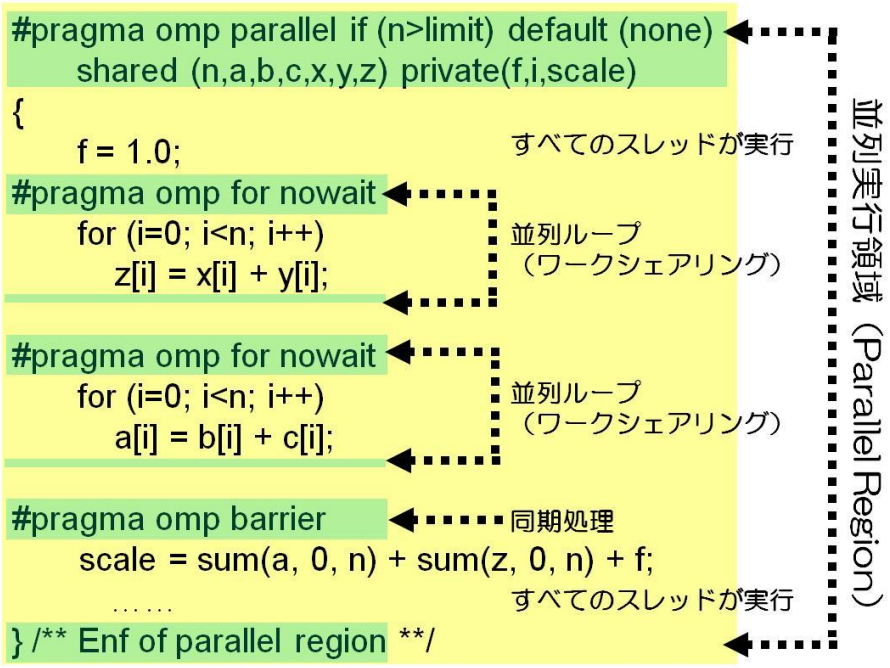
OpenMP プログラムの高速化手法

OpenMP 性能解析ツール ompP の利用について

１．はじめに

OpenMP は、業界標準規格であり、多くの Unix、Linux 及び Windows で利用可能です。Fortran、C/C++の言語規格に準拠しているため、OpenMP を利用することで、プログラムの移植性や互換性を一切、失うことなく並列処理が可能となります。OpenMP を利用したマルチスレッド・プログラミングでは、プログラムに対して、OpenMP　API で規定された指示行を挿入し、並列実行のためのプログラム構造を指定します。OpenMP 指示行を指定したプログラムは、コンパイル時に並列化を指示するオプションを指定してコンパイルすることで、コンパイラは並列化コードを作成します。

Windows や Posix スレッドによる並列処理では、ライブラリ呼び出しによって、陽的にスレッドの生成、管理、同期処理などを行う必要があります。このような陽的なスレッドコントロールを行うプログラミングでは、コード自身の大きな書き直しやプログラム構造の変更などが必要になります。一方、OpenMP での並列処理では、コンパイラに対してプログラムの並列処理を指示することで、より容易な処理を可能とします。OpenMP は、このような並列処理のためのコンパイラ指示行（Pragma）、API 関数、環境変数などが用意されています。



プログラムの並列化に際しては、プログラム自身の並列性を見極めるといったハイレベルのプログラミングが必要になりますが、OpenMP による並列プログラミングでは、プログラム自身に対する変更は比較的少なく、また、その並列処理の適用は段階的に実施することが可能となります。

OpenMP での並列処理では、このプログラム内での並列処理の検討に際して、まず、並列に処理できる領域を見つけることが必要になります。並列に実行可能な領域（複数のスレッドが同時に同じプログラム部分を処理することが可能な領域）を見つけたら、その領域を並列実行領域として、コンパイラ指示行の挿入を行います。コンパイラは、この領域を複数のスレッドで同時に実行するコードを生成します。同時にこの並列実行領域内の処理を分散する方法もコンパイ

ラに指示します。（ワークシェア構文などの指定）

OpenMP では、この並列実行領域内の変数は、ループカウンター変数を除いて、各スレッドで共有されることを前提で処理されます。したがって、この並列実行領域内で、各スレッドが自分のローカル変数（他のスレッドによって更新されない、参照されない）として使用するものについては、指示行に明示的に記載する必要があります。この変数の指定を間違えた場合、OpenMP での並列処理は誤った計算結果を引き起こしますので、注意が必要です。

OpenMP でのプログラミングについては、多くの資料があります[1]。この資料では、OpenMP プログラムでの性能向上を図る手法を、性能解析ツール　ompP[2] を利用しての事例として紹介しています。

[1] http://www.sstc.co.jp/biz/projects/OpenMP.html　OpenMP プログラミング入門&プログラミングトレーニング

[2] http://www.ompp-tool.com The OpenMP Profiler ompP

2. OpenMP 性能解析ツール

OpenMP は、マイクロプロセッサのマルチコア化や並列処理のニーズの高まりによって、より多くの関心を集めるようになっています。しかし、OpenMP プログラムの性能解析については、その標準化インターフェイスなどの規格化は、現時点ではなされていません。商用の解析ツールとしては、インテルの Intel Thread Profiler と Intel Thread Checker[3]などが、代表的なツールであり非常に高度な解析と詳細な分析が可能なツールです。
OpenMP の解析ツールについては、商用製品だけでなく、研究機関が提案し開発したものも数多くあります。

- POMP: OpenMP のための性能解析インターフェイス （ユーリッヒ研究所）
- Opari: ソース変換ツール
- TAU: 解析ツール （オレゴン大学）
- KOJAK: 解析ツール（テネシー大学）

これらのツールの一つとして、ompP があります。ompP はテネシー大学の ICL（Innovative Computing Laboratory）が開発した性能解析ツールで、次のような特徴を持ちます。

- FORTRAN と C/C++をサポート
- ompP は Static Library でのリンク
- Opari を利用して、ソースコード変換を行う
- 非常にシンプルなコマンドインターフェイスでの実装
- 実行時の設定は、環境変数で可能
- 各 OpenMP 構文の実行を実行数と実行時間で表示
- OpenMP の実行時のユーザモードをベースに解析を行う（システムモデルではない）
- PAPI が提供する HW カウンタの利用も可能
- ASCII ベース（テキスト又は CSV 形式）での結果の出力

ompP には GUI のサポートはありませんが、非常にシンプルなコマンドで利用可能であり、OpenMP で並列化されたプログラムがあれば、コンパイル時に ompP 対応にコンパイルを行うだけで利用可能です。出力は ASCII 形式だけになりますが、その出力結果は非常に見やすく有用な情報を多々提供します。

ompP は、GPL ライセンスによって、http://www.ompp-tool.com よりダウンロード可能です。ompP は、Linux だけでなく、Unix プラットフォーム（Solaris や AIX）でも利用可能であり、インテルコンパイラ、PGI、Pathscale、gcc などをサポートしています。

[3] http://www.intel.com/cd/software/products/asmo-na/eng/threading/219785.htm Intel® Threading Analysis Tools

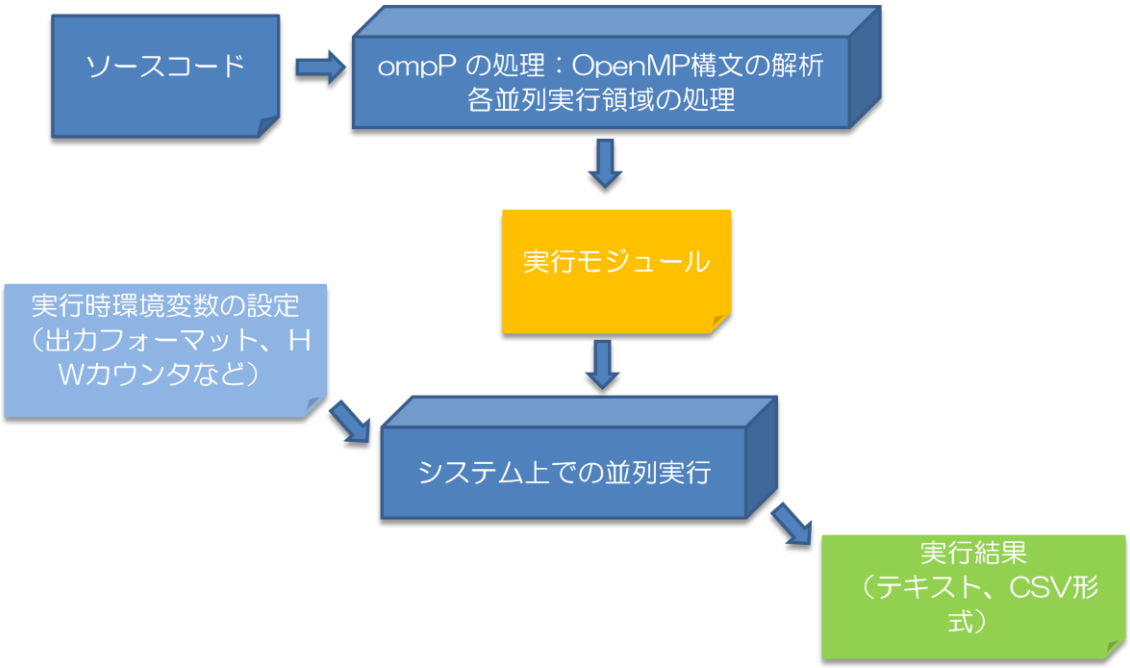


図-1 ompP 処理フロー

プログラムに対して、ompP を利用する場合は、ompP パッケージで提供されるコンパイルスクリプト kinst-ompp 又は kinst-ompp-papi を利用してコンパイルを行います。例えば、次のようなインテルコンパイラのコンパイル時の処理の場合

% icc –openmp foo.c bar.c –o myapp

は、次のようにコンパイルします。

% kinst-omp icc –openmp foo.c bar.c –o myapp

Makefile などで、コンパイルを指定する場合にも、一般に CC = icc と記述する部分を CC = kinst-ompp icc と変更することで、利用可能です。

ompP は、次のような内容で OpenMP プログラムの並列処理に関する解析結果を出力します。

- プログラム実行途中での、プロファイルのダンプ出力
- オーバヘッド解析（個々の並列実行領域とプログラム全体での解析）
- スケーラビリティ解析
- プログラム実行性能属性
- 継続実行プロファイル（プロファイリングとトレーシングの中間的解析）

出力されるレポートには、以下のような情報が提供されます。

- プログラム実行一般情報（実行日時、実行時間、スレッド数、HWカウンタ情報など）
- 並列実行領域概要（領域番号、ソースコードライン）
- フラット領域プロファイル（時間、実行回数、HWカウンタ情報）
- 呼び出しグラフ
- 呼び出しグラフプロファイル
- オーバヘッド解析レポート（４つのオーバヘッドカテゴリ毎の時間と比率）
- 性能に関する分析結果の出力

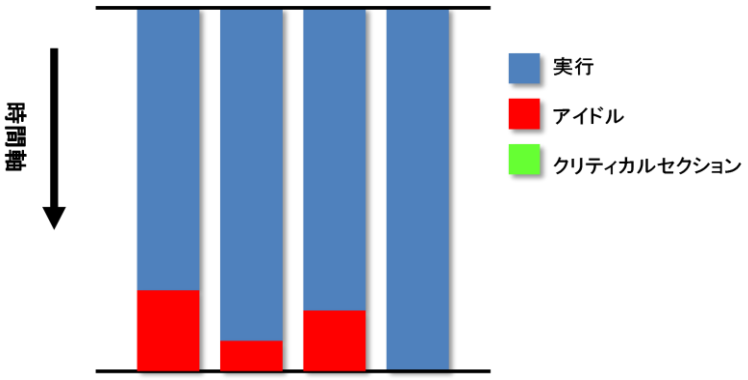
OpenMP プログラムの解析で最も、重要なことはオーバヘッドに関する解析です。プログラムの実行時のオーバヘッドを正しく把握することで、そのオーバヘッドの解消を図ることが可能となります。

3. オーバヘッド解析

ompP の解析では、次の４つのカテゴリのオーバヘッドを分析します。

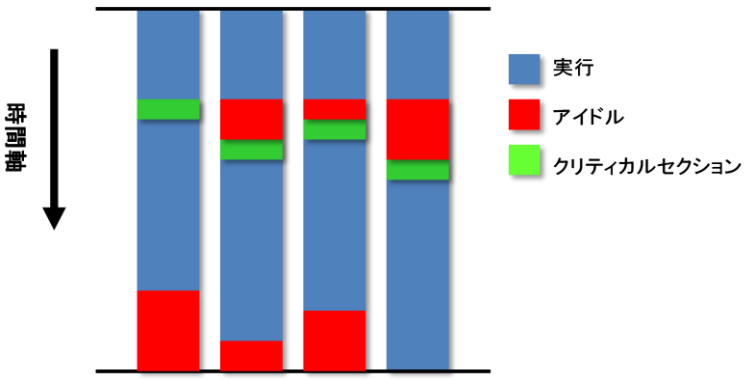
- バランス：ワークシェアや並列実行領域での各スレッドの実行時間にばらつきがあり、スレッドがアイドルする場合

```
#pragma omp parallel
{

#pragma omp for
  for (;;) {
   . . .
  }

}
```



- 同期：並列処理での各スレッドの同期処理に際してのオーバヘッド
- スレッドマネージメント：スレッドの生成や終了処理のための時間などやクリティカルセクションの処理、アトミックアップデートなどの処理に要する時間が大きい場合

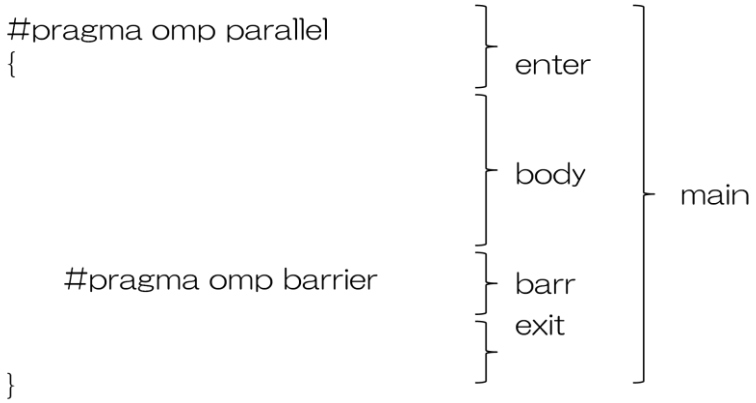
```
#pragma omp parallel
{

#pragma omp critical
  {
      . . .
  }
   . . .
}
```



- 並列度：プログラムの実行に際して、十分な並列度がない場合

OpenMP プログラムは、各並列実行領域を次のように分割して解析を行います。

Main = Enter（開始処理）+ Body （本体処理）+ Barr（同期処理） + Exit （終了処理）



OpenMP の各構文に対して、計測される時間と実行回数のカテゴリを示すと表-1 のようになります。オーバーヘッドは、OpenMP の各構文に対して、その実行カテゴリ毎に評価されます。

	main		*enter*		*body*					*barr*	*exit*	
construct	execT	execC	enterT	startupT	bodyT	sectionT	sectionC	singleT	singleC	exitBarT	exitT	shutdwnT
MASTER	•	•										
ATOMIC	•(S)	•										
BARRIER	•(S)	•										
FLUSH	•(S)	•										
USER REGION	•	•										
CRITICAL	•	•	•(S)		•						•(M)	
LOCK	•	•	•(S)		•						•(M)	
LOOP	•	•			•					•(I)		
WORKSHARE	•	•			•					•(I)		
SECTIONS	•	•				•	•			•(I/L)		
SINGLE	•	•						•	•	•(L)		
PARALLEL	•	•		•(M)	•					•(I)		•(M)
PARALLEL LOOP	•	•		•(M)	•					•(I)		•(M)
PARALLEL SECTIONS	•	•		•(M)		•	•			•(I/L)		•(M)
PARALLEL WORKSHARE	•	•		•(M)	•					•(I)		•(M)

I：バランス　S：同期　L：並列度M：スレッドマネージメント

表-1 OpenMP 構文と解析カテゴリ

ompP の実行では、各並列実行領域での時間、カウントが計測されます。同時に、各並列実行領域のオーバヘッドについても、計測と評価がなされます。以下にサンプルとして、呼び出しグラフ、各並列実行領域のプロファイル、オーバヘッド解析レポートを示します。

```
----------------------------------------------------------------------
----      ompP Callgraph      ----------------------------------------
----------------------------------------------------------------------
  Inclusive  (%)   Exclusive  (%)
 154.54 (100.0%)    0.09 ( 0.06%)           [prime: 4 threads]
 154.45 (99.94%)  149.06 (96.45%)  PARLOOP  +-R00001 prime.c (21-41)
   5.39 ( 3.49%)    5.39 ( 3.49%) CRITICAL     +-R00002 prime.c (34-39) (unnamed)


----------------------------------------------------------------------
----      ompP Flat Region Profile (inclusive data)      ----------
----------------------------------------------------------------------
R00001 prime.c (21-41) PARALLEL LOOP
 TID      execT      execC      bodyT   exitBarT   startupT   shutdwnT
   0     154.45          1      68.96      85.49       0.00       0.00
   1     154.45          1     109.55      44.90       0.01       0.00
   2     154.45          1     134.61      19.84       0.00       0.00
   3     154.45          1     154.45       0.00       0.00       0.00
 SUM     617.82          4     467.58     150.23       0.01       0.00
R00002 prime.c (34-39) (unnamed) CRITICAL
 TID      execT      execC      bodyT     enterT      exitT
   0      5.74    1565927       1.88       2.13       1.73
   1      5.37    1435207       1.77       2.00       1.60
   2      5.30    1393170       1.67       1.91       1.72
   3      5.15    1367151       1.62       1.92       1.62
 SUM     21.56    5761455       6.93       7.95       6.67


----------------------------------------------------------------------
----      ompP Overhead Analysis Report      ----------------------
----------------------------------------------------------------------
Total runtime (wallclock)   : 154.54 sec [4 threads]
Number of parallel regions  : 1
Parallel coverage           : 154.45 sec (99.94%)
Parallel regions sorted by wallclock time:
          Type                          Location      Wallclock (%)
R00001    PARLOOP                 prime.c (21-41)     154.45 (99.94)
                                          SUM         154.45 (99.94)
Overheads wrt. each individual parallel region:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal    (%)  +   Limpar (%)   +    Mgmt (%)
R00001   617.82   164.86 (26.68)    7.95 ( 1.29)  150.23 (24.32)    0.00 ( 0.00)    6.68 ( 1.08)
Overheads wrt. whole program:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal    (%)  +   Limpar (%)   +    Mgmt (%)
R00001   617.82   164.86 (26.67)    7.95 ( 1.29)  150.23 (24.30)    0.00 ( 0.00)    6.68 ( 1.08)
   SUM   617.82   164.86 (26.67)    7.95 ( 1.29)  150.23 (2.30)     0.00 ( 0.00)    6.68 ( 1.08)
```

並列実行領域の各スレッドの各並列実行領域のCPU時間と個々のオーバヘッドの時間とその割合を示しています。

プログラムの実行経過時間と並列実行領域の数を示します。

各並列実行領域のCPU時間と個々のオーバヘッドの時間とその割合を示しています。

4. プログラム事例

ompP でのプログラム解析の事例を示します。ここでは、非常に簡単な素数計算のプログラム（brute force アルゴリズム：総当たりアルゴリズム）を例に示します。プログラムは、実行時に素数を計算する範囲を指定し、その範囲の全ての整数が素数であるかを順次計算して求めています。範囲内の整数の素数判定は、個々に独立して計算出来るため、このプログラムは非常に高い並列性を持ちます。しかし、OpenMP の指示行を適切に指定しない場合には、期待した性能が得られません。ここでは、幾つかの OpenMP での並列化の事例とその OpenMP プログラムの ompP での解析結果を示します。

```
#include <stdio.h>
#include <math.h>

main(int argc, char *argv[])
{
   int i, j, limit;
   int start, end;          /* range of numbers to search */
   int number_of_primes=0;  /* number of primes found */
   int number_of_41primes=0;/* number of 4n+1 primes found */
   int number_of_43primes=0;/* number of 4n-1 primes found */
   int prime;               /* is the number prime? */
   int print_primes=0;      /* should each prime be printed? */

   start = atoi(argv[1]);
   end = atoi(argv[2]);
   if (!(start % 2)) start++;

   if (argc == 4 && atoi(argv[3]) != 0) print_primes = 1;
   printf("Range to check for Primes: %d - %d¥n¥n",start, end);

#pragma omp parallel for private (limit, j, prime)
   for(i = start; i <= end; i += 2) {

      limit = (int) sqrt((float)i) + 1;
      prime = 1; /* assume number is prime */
      j = 3;
      while (prime && (j <= limit)) {
         if (i%j == 0) prime = 0;
         j += 2;
      }

      if (prime) {
            if (print_primes) printf("%5d is prime¥n",i);
#pragma omp critical
        {
          number_of_primes++;
          if (i%4 == 1) number_of_41primes++;
          if (i%4 == 3) number_of_43primes++;
        }
      }
   }

   printf("¥nProgram Done.¥n %d primes found¥n",number_of_primes);
   printf("¥nNumber of 4n+1 primes found: %d¥n",number_of_41primes);
   printf("¥nNumber of 4n-1 primes found: %d¥n",number_of_43primes);
  }
```

このプログラムは非常に単純で、一つの for ループが並列実行領域になります。この for ループ内で、クリティカルセ

クションを設定して、並列実行領域内で、共有変数のアップデートを行っています。

このプログラムを ompP で解析した場合、以下のような結果が得られます。

```
------------------------------------------------------------------
----     ompP General Information     ----------------------------
------------------------------------------------------------------
Start Date      : Tue Mar 11 14:23:51 2008
End Date        : Tue Mar 11 14:26:25 2008
Duration        : 154.51 sec
User Time       : 454.24 sec
System Time     : 13.86 sec
Max Threads     : 4
ompP Version    : 0.6.3
ompP Build Date : Mar  6 2008 14:48:14
PAPI Support    : not available

------------------------------------------------------------------
----     ompP Region Overview     ------------------------------
------------------------------------------------------------------
PARALLEL LOOP: 1 region:
 * R00001 prime.c (21-41)

CRITICAL: 1 region:
 * R00002 prime.c (34-39) (unnamed)

------------------------------------------------------------------
----     ompP Callgraph     ------------------------------------
------------------------------------------------------------------

  Inclusive   (%)   Exclusive   (%)
 154.51 (100.0%)    0.03 ( 0.02%)           [prime: 4 threads]
 154.48 (99.98%)  149.09 (96.50%)  PARLOOP  +-R00001 prime.c (21-41)
   5.39 ( 3.49%)    5.39 ( 3.49%) CRITICAL     +-R00002 prime.c (34-39) (unnamed)

------------------------------------------------------------------
----     ompP Flat Region Profile (inclusive data)     ---------
------------------------------------------------------------------
R00001 prime.c (21-41) PARALLEL LOOP
 TID      execT      execC      bodyT   exitBarT   startupT   shutdwnT
   0     154.48          1      68.89      85.59       0.00       0.00
   1     154.48          1     109.56      44.91       0.00       0.00
   2     154.48          1     134.69      19.78       0.00       0.00
   3     154.48          1     154.48       0.00       0.00       0.00
 SUM     617.91          4     467.62     150.28       0.01       0.00

R00002 prime.c (34-39) (unnamed) CRITICAL
 TID      execT      execC      bodyT     enterT      exitT
   0       5.73    1565927       1.88       2.12       1.73
   1       5.36    1435207       1.76       1.99       1.61
   2       5.29    1393170       1.68       1.88       1.74
   3       5.17    1367151       1.61       1.94       1.61
 SUM      21.55    5761455       6.94       7.93       6.68

------------------------------------------------------------------
----     ompP Callgraph Region Profiles (incl./excl. data)     -------
------------------------------------------------------------------

[*00] prime
[+01] R00001 prime.c (21-41) PARALLEL LOOP
 TID      execT      execC    bodyT/I    bodyT/E   exitBarT   startupT   shutdwnT
```

```
    0      154.48           1       68.89       63.10       85.59        0.00        0.00
    1      154.48           1      109.56      104.13       44.91        0.00        0.00
    2      154.48           1      134.69      129.49       19.78        0.00        0.00
    3      154.48           1      154.48      149.35        0.00        0.00        0.00
  SUM      617.91           4      467.62      446.07      150.28        0.01        0.00

[*00] prime
[+01] R00001 prime.c (21-41) PARALLEL LOOP
[=02] R00002 prime.c (34-39) (unnamed) CRITICAL
 TID      execT      execC    bodyT/I    bodyT/E     enterT      exitT
   0       5.79    1565927       1.82       1.82       2.14       1.82
   1       5.43    1435207       1.70       1.70       2.02       1.70
   2       5.20    1393170       1.65       1.65       1.91       1.65
   3       5.13    1367151       1.72       1.72       1.88       1.53
 SUM      21.55    5761455       6.90       6.90       7.94       6.71


------------------------------------------------------------------------
----     ompP Overhead Analysis Report     ----------------------------
------------------------------------------------------------------------
Total runtime (wallclock)   : 154.51 sec [4 threads]
Number of parallel regions  : 1
Parallel coverage           : 154.48 sec (99.98%)

Parallel regions sorted by wallclock time:
           Type                              Location      Wallclock (%)
R00001   PARLOOP                     prime.c (21-41)     154.48 (99.98)
                                                 SUM     154.48 (99.98)

Overheads wrt. each individual parallel region:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal   (%)  +   Limpar (%)   +    Mgmt (%)
R00001   617.91   164.90 (26.69)    7.93 ( 1.28)  150.28 (24.32)    0.00 ( 0.00)    6.69 ( 1.08)

Overheads wrt. whole program:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal   (%)  +   Limpar (%)   +    Mgmt (%)
R00001   617.91   164.90 (26.68)    7.93 ( 1.28)  150.28 (24.32)    0.00 ( 0.00)    6.69 ( 1.08)
   SUM   617.91   164.90 (26.68)    7.93 ( 1.28)  150.28 (24.32)    0.00 ( 0.00)    6.69 ( 1.08)


------------------------------------------------------------------------
----     ompP Performance Properties Report     ----------------------
------------------------------------------------------------------------

Property P00001 'ImbalanceInParallelLoop' holds for
     'PARALLEL LOOP prime.c (21-41)', with a severity (in percent) of 24.316503

Property P00002 'CriticalSectionContention' holds for
     'CRITICAL prime.c (34-39) (unnamed)', with a severity (in percent) of 1.283167
```

このプログラムの結果では、明らかに各スレッドのバランスが悪いことが示されています。OpenMP での for ループの処理では、デフォルトでは、各ループの実行は、等分に各スレッドに分散されます。素数計算の場合、素数の分布が均等ではないため、このようなループの分割を行うと各スレッドの実行比率は均等にはなりません。
また、プログラムの解析結果では、クリティカルセクションの実行が全体の 1.28%程度もあり、この部分の低減を図る必要があります。

これらの点を改善するために、プログラムを次のように変更します。

- ワークシェアの実行制御を Static から Dynamic に変更
- スレッドでの for ループの分割（Chunk）サイズを１から 100 に変更
- 各スレッドのローカル変数を設定し、各スレッドの計算結果を for ループの終了後にアップデート（クリティカルセクション）

```
#include <stdio.h>
#include <math.h>

main(int argc, char *argv[])
{
   int i, j, limit;
   int start, end;          /* range of numbers to search */
   int number_of_primes=0;  /* number of primes found */
   int number_of_41primes=0;/* number of 4n+1 primes found */
   int number_of_43primes=0;/* number of 4n-1 primes found */
   int prime;               /* is the number prime? */
   int print_primes=0;      /* should each prime be printed? */

   start = atoi(argv[1]);
   end = atoi(argv[2]);
   if (!(start % 2)) start++;

   if (argc == 4 && atoi(argv[3]) != 0) print_primes = 1;
   printf("Range to check for Primes: %d - %d¥n¥n",start, end);


#pragma omp parallel
{
  int numPrimes=0;    /* local number of primes found */
  int num41Primes=0;  /* local number of 4n+1 primes found */
  int num43Primes=0;  /* local number of 4n-1 primes found */

#pragma omp for schedule(dynamic,100)
  for(i = start; i <= end; i += 2) {
    int limit, j, prime;

    limit = (int) sqrt((float)i) + 1;
    prime = 1; /* assume number is prime */
    j = 3;
    while (prime && (j <= limit)) {
       if (i%j == 0) prime = 0;
       j += 2;
    }

    if (prime) {
          if (print_primes) printf("%5d is prime¥n",i);
      numPrimes++;
      if (i%4 == 1) num41Primes++;
      if (i%4 == 3) num43Primes++;
    }
  } // end for

#pragma omp critical
  { // Update global counter values with local values
    number_of_primes += numPrimes;
    number_of_41primes += num41Primes;
    number_of_43primes += num43Primes;
  }
} // end parallel region

   printf("¥nProgram Done.¥n %d primes found¥n",number_of_primes);
```

```
  printf("¥nNumber of 4n+1 primes found: %d¥n",number_of_41primes);
  printf("¥nNumber of 4n-1 primes found: %d¥n",number_of_43primes);
  }
```

プログラムのワークシェアの実行制御に際して、ループの各スレッドの実行に際して、その処理するループカウントを100回毎に分割し、処理の終了したスレッドが順次、処理を行います。また、クリティカルセクションの実行をワークシェアの外側（for ループの外側）で行うことで、スレッド処理のオーバヘッドの低減を図ります。

```
------------------------------------------------------------------------
----     ompP General Information     ---------------------------------
------------------------------------------------------------------------
Start Date      : Tue Mar 11 14:26:25 2008
End Date        : Tue Mar 11 14:28:14 2008
Duration        : 109.26 sec
User Time       : 436.63 sec
System Time     : 0.00 sec
Max Threads     : 4
ompP Version    : 0.6.3
ompP Build Date : Mar  6 2008 14:48:14
PAPI Support    : not available

------------------------------------------------------------------------
----     ompP Region Overview     ---------------------------------------
------------------------------------------------------------------------
PARALLEL: 1 region:
 * R00001 prime2.c (22-54)

LOOP: 1 region:
 * R00002 prime2.c (28-46)

CRITICAL: 1 region:
 * R00003 prime2.c (48-53) (unnamed)

------------------------------------------------------------------------
----     ompP Callgraph     ----------------------------------------------
------------------------------------------------------------------------

  Inclusive   (%)   Exclusive   (%)
 109.26 (100.0%)    0.12 ( 0.11%)           [prime2: 4 threads]
 109.14 (99.89%)    0.00 ( 0.00%) PARALLEL  +-R00001 prime2.c (22-54)
 109.14 (99.89%)  109.14 (99.89%)     LOOP     |-R00002 prime2.c (28-46)
   0.00 ( 0.00%)    0.00 ( 0.00%) CRITICAL     +-R00003 prime2.c (48-53) (unnamed)

------------------------------------------------------------------------
----     ompP Flat Region Profile (inclusive data)     -----------------
------------------------------------------------------------------------
R00001 prime2.c (22-54) PARALLEL
 TID      execT      execC      bodyT   exitBarT   startupT   shutdwnT
   0     109.14          1     109.14       0.00       0.00       0.00
   1     109.14          1     109.14       0.00       0.00       0.00
   2     109.14          1     109.14       0.00       0.01       0.00
   3     109.14          1     109.14       0.00       0.00       0.00
 SUM     436.58          4     436.57       0.00       0.01       0.00

R00002 prime2.c (28-46) LOOP
 TID      execT      execC      bodyT   exitBarT
   0     109.14          1     109.14       0.00
   1     109.14          1     109.14       0.00
   2     109.14          1     109.14       0.00
   3     109.14          1     109.14       0.00
 SUM     436.57          4     436.57       0.00
```

```
R00003 prime2.c (48-53) (unnamed) CRITICAL
 TID      execT      execC      bodyT     enterT      exitT
   0       0.00          1       0.00       0.00       0.00
   1       0.00          1       0.00       0.00       0.00
   2       0.00          1       0.00       0.00       0.00
   3       0.00          1       0.00       0.00       0.00
 SUM       0.00          4       0.00       0.00       0.00

------------------------------------------------------------------------
----     ompP Callgraph Region Profiles (incl./excl. data)     -------
------------------------------------------------------------------------

[*00] prime2
[+01] R00001 prime2.c (22-54) PARALLEL
 TID      execT      execC    bodyT/I    bodyT/E   exitBarT   startupT   shutdwnT
   0     109.14          1     109.14       0.00       0.00       0.00       0.00
   1     109.14          1     109.14       0.00       0.00       0.00       0.00
   2     109.14          1     109.14       0.00       0.00       0.01       0.00
   3     109.14          1     109.14       0.00       0.00       0.00       0.00
 SUM     436.58          4     436.57       0.00       0.00       0.01       0.00

[*00] prime2
[+01] R00001 prime2.c (22-54) PARALLEL
[=02] R00002 prime2.c (28-46) LOOP
 TID      execT      execC    bodyT/I    bodyT/E   exitBarT
   0     109.14          1     109.14     109.14       0.00
   1     109.14          1     109.14     109.14       0.00
   2     109.14          1     109.14     109.14       0.00
   3     109.14          1     109.14     109.14       0.00
 SUM     436.57          4     436.57     436.57       0.00

[*00] prime2
[+01] R00001 prime2.c (22-54) PARALLEL
[=02] R00003 prime2.c (48-53) (unnamed) CRITICAL
 TID      execT      execC    bodyT/I    bodyT/E     enterT      exitT
   0       0.00          1       0.00       0.00       0.00       0.00
   1       0.00          1       0.00       0.00       0.00       0.00
   2       0.00          1       0.00       0.00       0.00       0.00
   3       0.00          1       0.00       0.00       0.00       0.00
 SUM       0.00          4       0.00       0.00       0.00       0.00


------------------------------------------------------------------------
----     ompP Overhead Analysis Report     ----------------------------
------------------------------------------------------------------------
Total runtime (wallclock)   : 109.26 sec [4 threads]
Number of parallel regions  : 1
Parallel coverage           : 109.14 sec (99.89%)

Parallel regions sorted by wallclock time:
           Type                              Location      Wallclock (%)
R00001  PARALLEL                    prime2.c (22-54)     109.14 (99.89)
                                                SUM     109.14 (99.89)

Overheads wrt. each individual parallel region:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal   (%)  +   Limpar (%)   +    Mgmt (%)
R00001   436.58     0.01 ( 0.00)    0.00 ( 0.00)    0.00 ( 0.00)    0.00 ( 0.00)    0.01 ( 0.00)

Overheads wrt. whole program:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal   (%)  +   Limpar (%)   +    Mgmt (%)
R00001   436.58     0.01 ( 0.00)    0.00 ( 0.00)    0.00 ( 0.00)    0.00 ( 0.00)    0.01 ( 0.00)
   SUM   436.58     0.01 ( 0.00)    0.00 ( 0.00)    0.00 ( 0.00)    0.00 ( 0.00)    0.01 ( 0.00)
```

```
------------------------------------------------------------------
----     ompP Performance Properties Report     ------------------
------------------------------------------------------------------

Property P00001 'ImbalanceInParallelLoop' holds for
     'LOOP prime2.c (28-46)', with a severity (in percent) of 0.000585

Property P00002 'CriticalSectionContention' holds for
     'CRITICAL prime2.c (48-53) (unnamed)', with a severity (in percent) of 0.000013

Property P00003 'ImbalanceInParallelRegion' holds for
     'PARALLEL prime2.c (22-54)', with a severity (in percent) of 0.000007
```

このプログラムの結果では、明らかに各スレッドのバランスが改善し、各スレッドの実行時間が均等化しています。また、プログラムの解析結果では、クリティカルセクションの実行時間もほとんどゼロになっています。

これによって、最初のプログラムが、154 秒かかった計算が、109 秒で終了しています。

クリティカルセクションではなく、アトミックアップデートを利用しての処理も可能になります。アトミックアップデートは、クリティカルセクションよりもよりオーバヘッドの少ないオペレーションが可能となるため、for ループ内に指定しても、より少ない時間での処理が可能になります。

```
#include <stdio.h>
#include <math.h>

main(int argc, char *argv[])
{
   int i, j, limit;
   int start, end;          /* range of numbers to search */
   int number_of_primes=0;  /* number of primes found */
   int number_of_41primes=0;/* number of 4n+1 primes found */
   int number_of_43primes=0;/* number of 4n-1 primes found */
   int prime;               /* is the number prime? */
   int print_primes=0;      /* should each prime be printed? */

   start = atoi(argv[1]);
   end = atoi(argv[2]);
   if (!(start % 2)) start++;

   if (argc == 4 && atoi(argv[3]) != 0) print_primes = 1;
   printf("Range to check for Primes: %d - %d¥n¥n",start, end);

#pragma omp parallel
{

#pragma omp for schedule(dynamic,100)
  for(i = start; i <= end; i += 2) {
    int limit, j, prime;

    limit = (int) sqrt((float)i) + 1;
    prime = 1; /* assume number is prime */
    j = 3;
    while (prime && (j <= limit)) {
       if (i%j == 0) prime = 0;
       j += 2;
    }
```

```
    if (prime) {
        if (print_primes) printf("%5d is prime¥n",i);
#pragma omp atomic
      number_of_primes++;
      if (i%4 == 1) {
#pragma omp atomic
        number_of_41primes++;
      }
      if (i%4 == 3) {
#pragma omp atomic
        number_of_43primes++;
      }
    }
  } // end for
} // end parallel region

   printf("¥nProgram Done.¥n %d primes found¥n",number_of_primes);
   printf("¥nNumber of 4n+1 primes found: %d¥n",number_of_41primes);
   printf("¥nNumber of 4n-1 primes found: %d¥n",number_of_43primes);
}
```

この場合の解析結果は次のようになります。クリティカルセクションを使用した場合よりも同期の時間は減りますが、for ループ内での処理のため、オーバヘッドが発生します。この例に示すまでもなく、OpenMP プログラムでは、ループ内での同期処理は、可能であれば避けるべきです。

```
---------------------------------------------------------------------
----     ompP General Information     ---------------------------------
---------------------------------------------------------------------
Start Date      : Tue Mar 11 14:28:14 2008
End Date        : Tue Mar 11 14:30:12 2008
Duration        : 117.35 sec
User Time       : 455.08 sec
System Time     : 13.48 sec
Max Threads     : 4
ompP Version    : 0.6.3
ompP Build Date : Mar  6 2008 14:48:14
PAPI Support    : not available

---------------------------------------------------------------------
----     ompP Region Overview     ---------------------------------
---------------------------------------------------------------------
PARALLEL: 1 region:
 * R00001 prime3.c (22-51)

LOOP: 1 region:
 * R00002 prime3.c (25-50)

ATOMIC: 3 regions:
 * R00003 prime3.c (39-40)
 * R00004 prime3.c (42-43)
 * R00005 prime3.c (46-47)

---------------------------------------------------------------------
----     ompP Callgraph     ---------------------------------------
---------------------------------------------------------------------

  Inclusive   (%)   Exclusive   (%)
 117.35 (100.0%)    0.16 ( 0.14%)           [prime3: 4 threads]
 117.18 (99.86%)    0.00 ( 0.00%) PARALLEL  +-R00001 prime3.c (22-51)
 117.18 (99.86%)  113.91 (97.07%)     LOOP     +-R00002 prime3.c (25-50)
   1.69 ( 1.44%)    1.69 ( 1.44%)   ATOMIC        |-R00003 prime3.c (39-40)
```

```
   0.79 ( 0.68%)    0.79 ( 0.68%)   ATOMIC        |-R00004 prime3.c (42-43)
   0.79 ( 0.68%)    0.79 ( 0.68%)   ATOMIC        +-R00005 prime3.c (46-47)

------------------------------------------------------------------
----     ompP Flat Region Profile (inclusive data)     ----------
------------------------------------------------------------------
R00001 prime3.c (22-51) PARALLEL
 TID      execT      execC      bodyT   exitBarT   startupT   shutdwnT
   0     117.18          1     117.18       0.00       0.00       0.00
   1     117.18          1     117.18       0.00       0.00       0.00
   2     117.18          1     117.18       0.00       0.00       0.00
   3     117.18          1     117.18       0.00       0.00       0.00
 SUM     468.73          4     468.73       0.00       0.01       0.00

R00002 prime3.c (25-50) LOOP
 TID      execT      execC      bodyT   exitBarT
   0     117.18          1     117.18       0.00
   1     117.18          1     117.18       0.00
   2     117.18          1     117.18       0.00
   3     117.18          1     117.18       0.00
 SUM     468.73          4     468.72       0.00

R00003 prime3.c (39-40) ATOMIC
 TID      execT      execC
   0       1.59    1441248
   1       1.58    1441931
   2       1.63    1440521
   3       1.58    1437755
 SUM       6.37    5761455

R00004 prime3.c (42-43) ATOMIC
 TID      execT      execC
   0       0.84     721026
   1       0.82     721084
   2       0.82     719904
   3       0.81     718491
 SUM       3.30    2880505

R00005 prime3.c (46-47) ATOMIC
 TID      execT      execC
   0       0.81     720222
   1       0.81     720847
   2       0.81     720617
   3       0.81     719264
 SUM       3.24    2880950

------------------------------------------------------------------
----     ompP Callgraph Region Profiles (incl./excl. data)     -------
------------------------------------------------------------------

[*00] prime3
[+01] R00001 prime3.c (22-51) PARALLEL
 TID      execT      execC    bodyT/I    bodyT/E   exitBarT   startupT   shutdwnT
   0     117.18          1     117.18       0.00       0.00       0.00       0.00
   1     117.18          1     117.18       0.00       0.00       0.00       0.00
   2     117.18          1     117.18       0.00       0.00       0.00       0.00
   3     117.18          1     117.18       0.00       0.00       0.00       0.00
 SUM     468.73          4     468.73       0.00       0.00       0.01       0.00

[*00] prime3
[+01] R00001 prime3.c (22-51) PARALLEL
[+02] R00002 prime3.c (25-50) LOOP
 TID      execT      execC    bodyT/I    bodyT/E   exitBarT
   0     117.18          1     117.18     113.92       0.00
```

```
   1     117.18          1     117.18     113.89       0.00
   2     117.18          1     117.18     113.89       0.00
   3     117.18          1     117.18     113.92       0.00
 SUM     468.73          4     468.72     455.62       0.00

[*00] prime3
[+01] R00001 prime3.c (22-51) PARALLEL
[+02] R00002 prime3.c (25-50) LOOP
[=03] R00003 prime3.c (39-40) ATOMIC
 TID      execT      execC
   0       1.68    1441248
   1       1.69    1441931
   2       1.69    1440521
   3       1.68    1437755
 SUM       6.75    5761455

[*00] prime3
[+01] R00001 prime3.c (22-51) PARALLEL
[+02] R00002 prime3.c (25-50) LOOP
[=03] R00004 prime3.c (42-43) ATOMIC
 TID      execT      execC
   0       0.79     721026
   1       0.80     721084
   2       0.80     719904
   3       0.79     718491
 SUM       3.18    2880505

[*00] prime3
[+01] R00001 prime3.c (22-51) PARALLEL
[+02] R00002 prime3.c (25-50) LOOP
[=03] R00005 prime3.c (46-47) ATOMIC
 TID      execT      execC
   0       0.79     720222
   1       0.80     720847
   2       0.79     720617
   3       0.80     719264
 SUM       3.18    2880950


------------------------------------------------------------------------
----     ompP Overhead Analysis Report     ----------------------------
------------------------------------------------------------------------
Total runtime (wallclock)   : 117.35 sec [4 threads]
Number of parallel regions  : 1
Parallel coverage           : 117.18 sec (99.86%)

Parallel regions sorted by wallclock time:
           Type                              Location      Wallclock (%)
R00001  PARALLEL                    prime3.c (22-51)     117.18 (99.86)
                                                SUM      117.18 (99.86)

Overheads wrt. each individual parallel region:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal   (%)  +   Limpar (%)   +    Mgmt (%)
R00001   468.73    12.92 ( 2.76)   12.91 ( 2.75)    0.00 ( 0.00)    0.00 ( 0.00)    0.01 ( 0.00)

Overheads wrt. whole program:
          Total        Ovhds (%)  =   Synch  (%)  +  Imbal   (%)  +   Limpar (%)   +    Mgmt (%)
R00001   468.73    12.92 ( 2.75)   12.91 ( 2.75)    0.00 ( 0.00)    0.00 ( 0.00)    0.01 ( 0.00)
   SUM   468.73    12.92 ( 2.75)   12.91 ( 2.75)    0.00 ( 0.00)    0.00 ( 0.00)    0.01 ( 0.00)


------------------------------------------------------------------------
----     ompP Performance Properties Report     ----------------------
------------------------------------------------------------------------
```

```
Property P00001 'FrequentAtomic' holds for
     'ATOMIC prime3.c (39-40)', with a severity (in percent) of 1.357332

Property P00002 'FrequentAtomic' holds for
     'ATOMIC prime3.c (42-43)', with a severity (in percent) of 0.703236

Property P00003 'FrequentAtomic' holds for
     'ATOMIC prime3.c (46-47)', with a severity (in percent) of 0.690009

Property P00004 'ImbalanceInParallelLoop' holds for
     'LOOP prime3.c (25-50)', with a severity (in percent) of 0.000572

Property P00005 'ImbalanceInParallelRegion' holds for
     'PARALLEL prime3.c (22-51)', with a severity (in percent) of 0.000004
```

5. まとめとして

ompP は、研究プロジェクトから開発された OpenMP プログラムの性能解析ツールであり、非常にシンプルなコマンドでプログラムの実行状況の解析が可能です。解析結果の出力は、ASCII 形式のため、リモート環境やバッチ環境でも利用することが可能であり、また、その解析結果は直感的に内容が理解できるものになっています。

OpenMP プログラムは容易にプログラムの並列化が可能ですが、ロードバランスや動機処理のオーバヘッドなどがそのスケーラビリティの障害となります。ompP は、そのようなオーバヘッドや不均衡なロードバランスの状況を的確に解析するツールであり、このツールを利用することでプログラムの開発がより容易になります。

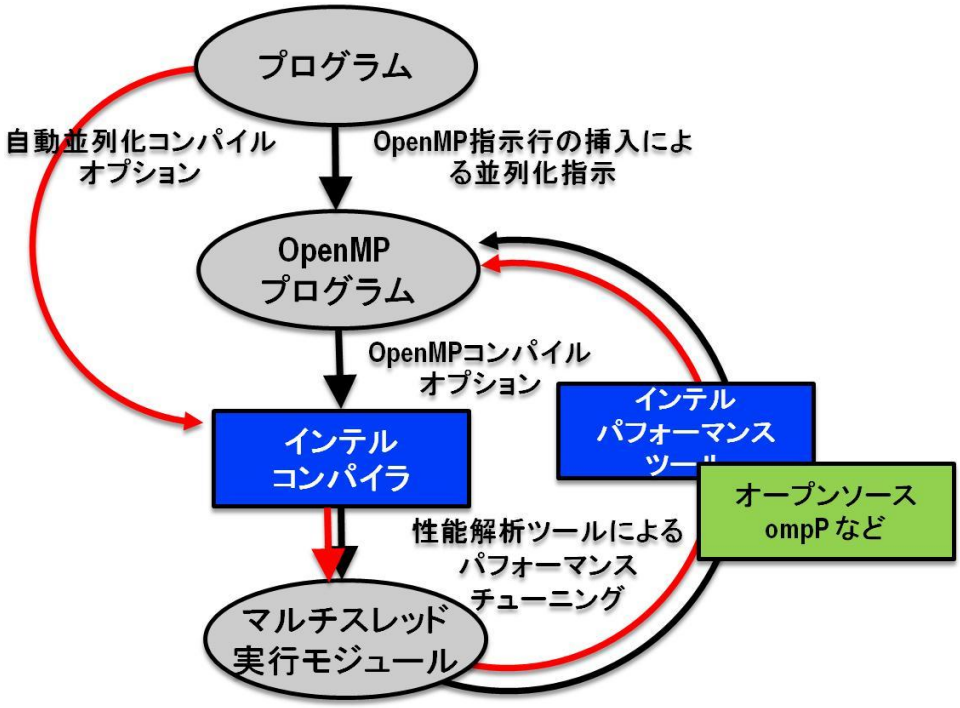


図-2 マルチスレッドアプリケーションの開発フロー

OpenMP のような高い互換性と移植性を備えた API での並列化でも、並列化の適用には、多くの試行錯誤とデバッグ作業が必要になる場合もあります。優れた開発環境とサポートによって、これらの並列化作業は良い容易に実行することが可能となります。

〒102-0083 東京都千代田区麹町 3-5-2 BUREX 麹町 8F

電話:03-5875-4718 FAX:03-3237-7612 www.sstc.co.jp

スケーラブルシステム株式会社では、IT 技術と HPC システムに関する様々な調査レポートを発行しています。

社名、製品名などは、一般に各社の商標または登録商標です。