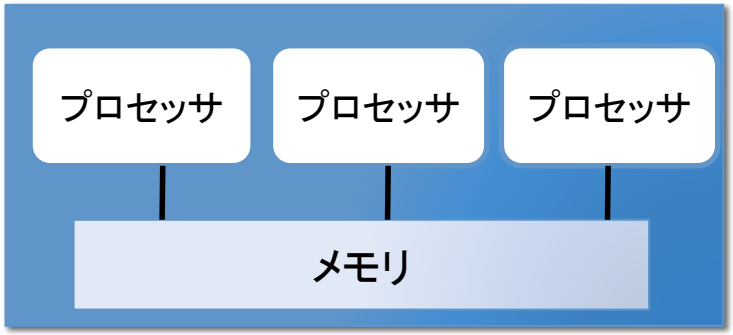
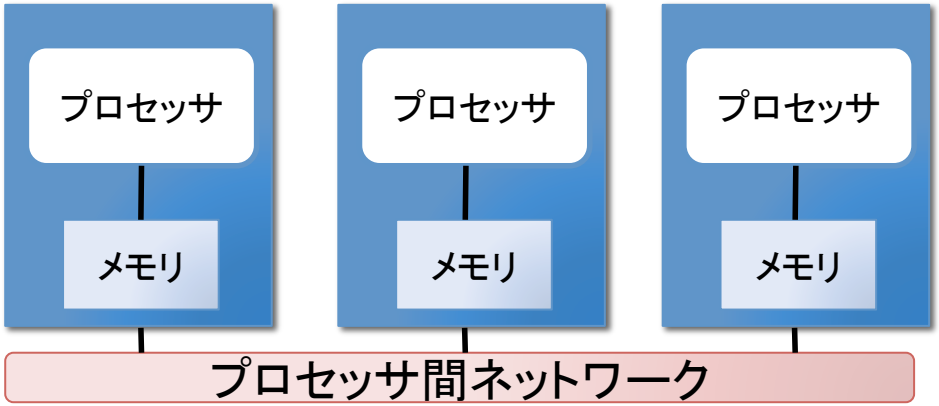
ハイパフォーマンスコンピューティング(4)

プログラムの並列化

- プログラムの並列化には主に2つの種類があります。
  - 共有メモリ型（例: OpenMP）
    - すべてのプロセッサからメモリ上のすべてのデータにアクセスが可能。分散メモリ型の場合に比べて、プログラムを自動的に並列化することが容易。
    - OpenMPでは、いくつかの指示文（特殊なコメント文）を追加するだけで、プログラムの論理的な構造を変化させることなく、並列化が可能。
  - 分散メモリ型（例: MPI）



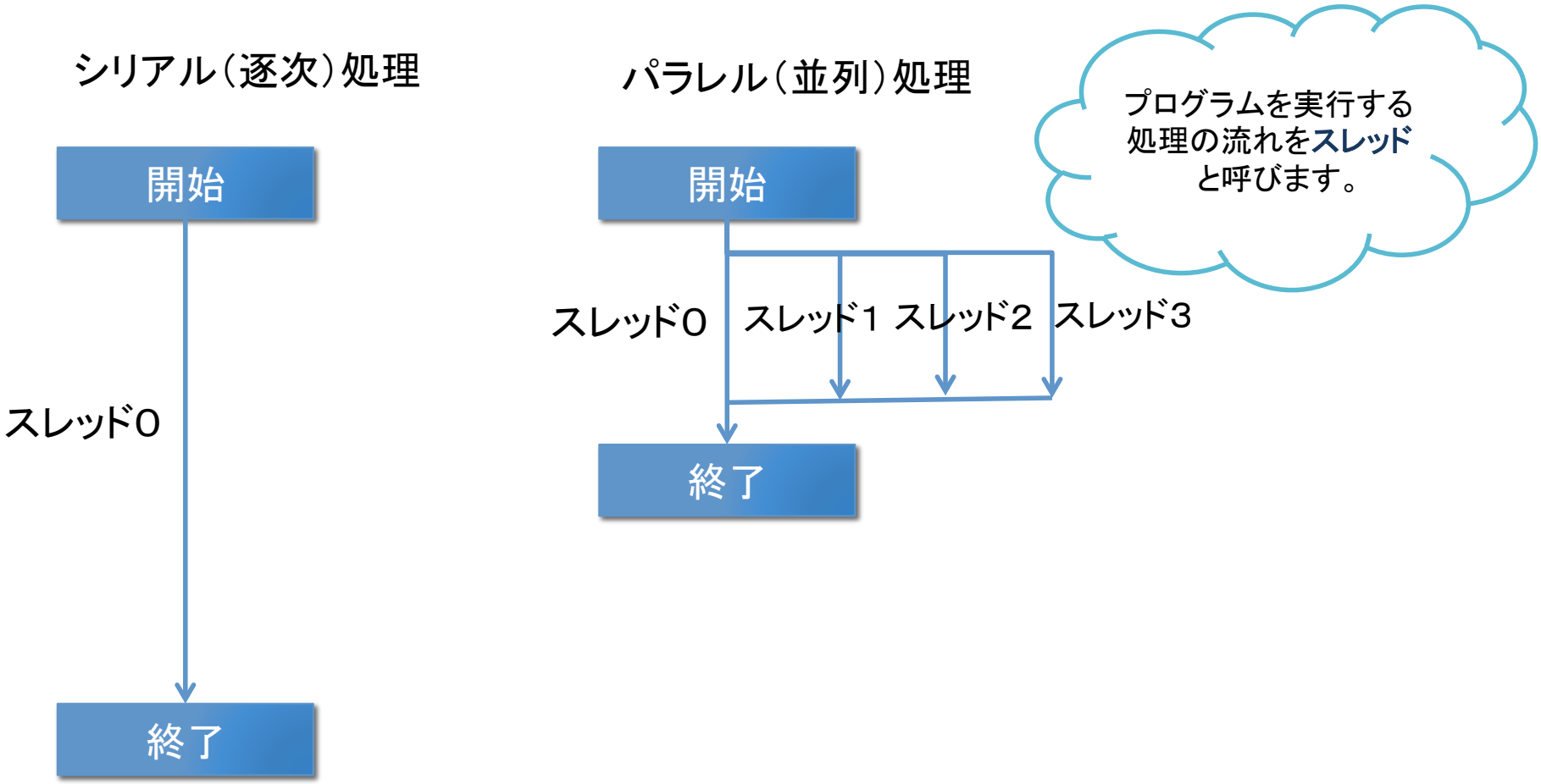
共有メモリ型並列イメージ



分散メモリ型並列イメージ

並列処理とは

- 並列処理するとは、時間的に独立し、物理的に平行して処理を行うこと



OpenMPの基本（C言語）

- #pragma文を利用してコンパイラに並列化の箇所、方法を指示し、並列化を行う。
  - 例）#pragma omp parallel
    { … }
- OpenMPの実行時ライブラリ関数を利用するためにヘッダファイル omp.h を読み込む
  - #include <omp.h>

- gccの場合、-fopenmp　を付けてコンパイルする。
  - gcc　×××.c　–fopenmp
  - iccの場合は、オプションは –openmp とする。

OpenMPの基本（Fortran）

- !$omp文を利用してコンパイラに並列化の箇所、方法を指示し、並列化を行う。
  - 例）!$omp parallel

    ...

    !$omp end parallel

- OpenMPのモジュール omp_lib を読み込む
  - use omp_lib

- gfortranの場合、-fopenmp　を付けてコンパイルする。
  - gfortran　×××.f90　–fopenmp
  - ifortの場合は、オプションは –openmp とする。

OpenMPプログラム例（C言語）

hellp_omp.c

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

int main()
{
  printf("procs = %d\n", omp_get_num_procs());
  printf("max #threads = %d\n", omp_get_max_threads());

#pragma omp parallel
  {
    printf("Hello from %d of %d\n", omp_get_thread_num(), omp_get_num_threads());
  }

  return 0;
}
```

使用可能なプロセッサ数

使用可能な最大スレッド数を表示

実行している各スレッドのスレッド番号

実行しているスレッド数

【OpenMPの環境変数】

export OMP_NUM_THREADS=N もしくは **setenv OMP_NUM_THREADS N**

(N：スレッド数)

⇒ OpenMP並列領域で使用するスレッド数。

値が指定されていない場合は、OSで認識されるプロセッサー数が設定される。

OpenMPプログラム例（Fortran）

hellp_omp.f90

```
program omp
use omp_lib

write(*,*) "proc = ", omp_get_num_procs()
write(*,*) "max #threads = ", omp_get_max_threads()
!$omp parallel
write(*,*) "Hello from", omp_get_thread_num(), "of", &
omp_get_num_threads()
!$omp end parallel

end program omp
```

OpenMPプログラム（OpenMPが動作しない環境でもコンパイル可能なプログラム）

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#endif

int main()
{
#ifdef _OPENMP
  printf("procs = %d\n", omp_get_num_procs());
  printf("max #threads = %d\n", omp_get_max_threads());
#endif

#ifdef _OPENMP
#pragma omp parallel
  {
    printf("Hello from %d of %d\n", omp_get_thread_num(), omp_get_num_threads());
  }
#else
  printf("Hello\n");
#endif
  return 0;
}
```

←OpenMPを使ってコンパイルするかどうかの判別
（gccでは、-fopenmpをつけると _OPENMPが定義される）

←OpenMPを使っていない場合、こちらが採用される。

OpenMPプログラム実行方法（C言語）

- 実行方法（ターミナル）

```
$ gcc  hello_omp.c
$ ./a.out
Hello

$ gcc  hello_omp.c  –fopenmp
$ ./a.out
procs = 4
max #threads = 4
Hello from 1 of 4
Hello from 2 of 4
Hello from 0 of 4
Hello from 3 of 4
```

セクション並列

- プログラムをある程度まとまった処理（セクション）毎に並列化したい場合は、#pragma omp sections/section を利用する。

```
#pragma omp parallel
#pragma omp sections
{
    #pragma omp section
    /* 並列に実行したい処理1 */
    …
    #pragma omp section
    /* 並列に実行したい処理2 */
    …
    #pragma omp section
    /* 並列に実行したい処理3 */
    …
    #pragma omp section
    /* 並列に実行したい処理4 */
    …
}
```

セクション並列

tomp1.c

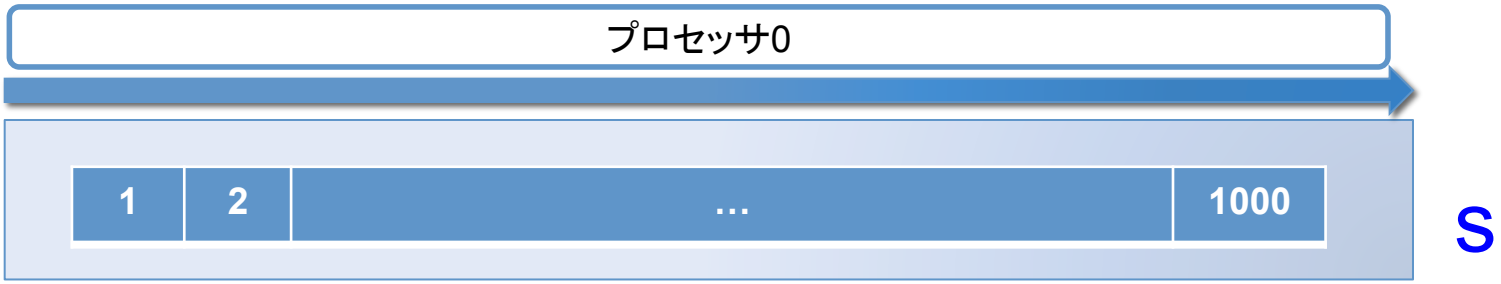
```
#include <stdio.h>

int main()
{
  printf("procs = %d\n", omp_get_num_procs());
  printf("max #threads = %d\n", omp_get_max_threads());

#pragma omp parallel
  {
#pragma omp sections
    {
#pragma omp section
      {
 printf("Section 1 Hello from %d of %d\n", omp_get_thread_num(), omp_get_num_threads());
      }
#pragma omp section
      {
 printf("Section 2 Hello from %d of %d\n", omp_get_thread_num(), omp_get_num_threads());
      }
    }
  }
return 0;
}
```

ベクトル計算の並列化(1)

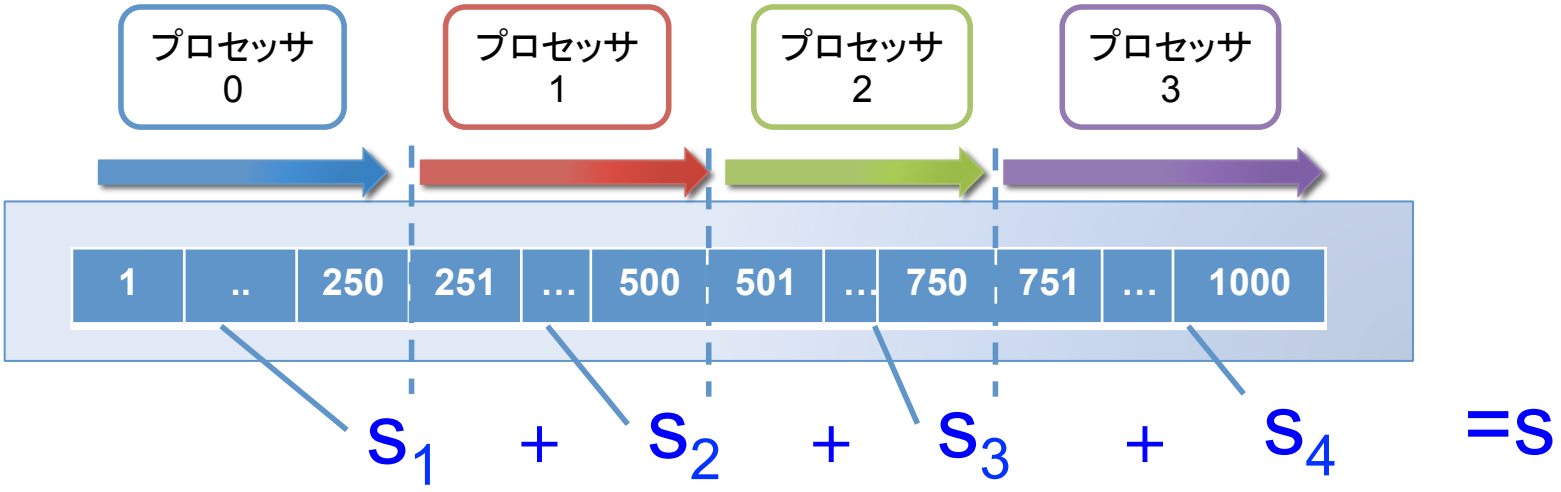
- n個のデータの総和を計算することを考える。

n=1000の例

■逐次処理の場合



■並列処理の場合



ベクトル計算の並列化(2)

- OpenMPでは、スレッド番号を取得し明示的にマルチスレッドプログラミングをすることもできるが、ワークシェアリング指示文を使うことによって、forループなどを簡単に並列化することもできる。
- たとえば、for文を指示文を使って並列化したいときは

```
#pragma omp parallel for
for (...)
```

のようにすれば良い。

最後に結果を足し合わせたりしたい場合は、

```
#pragma omp parallel for reduction(+:s)
for (...)
```

のようにすれば良い。

ベクトル計算の並列化(3)

■スレッド番号を用いて手動で並列化

```
s = 0;
#pragma omp parallel  {
int  ne, istart, iend, i, ss;        ← 各スレッドで別々に宣言される（ローカル変数）

ne = n/omp_get_num_threads();        ← n個を使用可能なスレッド数で割る
istart = ne*omp_get_thread_num();    ← スレッド別の担当部分の割り当て
iend = istart + ne;

ss = 0;
for (i = istart; i < iend; i++) ss += a(i);

#pragma omp atomic     ← atomicな処理にする（複数のスレッドが
s += ss;                 同時にsのロード・ストアをしないようにする）
}
```

■ワークシェアリング指示文による並列化（半自動並列化）

```
s = 0;
#pragma omp parallel for reduction(+:s)
for (i = 0; i < n; i++) s += a[i];
```

時間の計測

- omp_get_wtime()を用いる.
- 型は double precision (Fortran), double (C言語)
- 計測開始と終了時点の2箇所で呼び出しを行い，差が経過時間になる.

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

int main(void){

  double tic, toc;

  tic=omp_get_wtime();
    対象の処理
  toc=omp_get_wtime();
  printf("time(sec)=%f\n",toc-tic);

  return 0;
}
```

演習問題

適当なn個のデータ（nは実行時に指定できるようにする）を配列に格納し、

- 並列化していないもの
- スレッド番号を用いて手動で並列化
- ワークシェアリング指示文による並列化

の計算時間を比較しよう。

（注意）nは、かなり大きくとらないと並列化の効果が現れない場合があります。