KOBE HPC サマースクール 2018（初級）

10. 並列計算性能の評価方法（時間計測関数，バリア同期関数）

演習10-1　MPI_Allreduceの応用：ベクトルの正規化

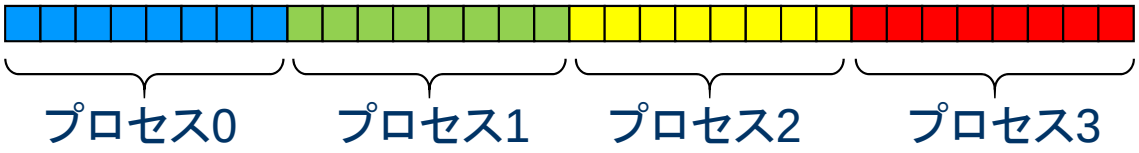
- n次元ベクトル $\boldsymbol{x}$ （double x[N];）の第 i 要素を $i+1$ とする.
  - $x[i] = i + 1 \quad (i = 0, \cdots, n-1)$

- $\boldsymbol{x}$ を正規化したベクトル $\boldsymbol{x}/\|\boldsymbol{x}\|_2$ を求めるプログラム normalize.c の逐次実行動作を確認せよ.
  - 変数は倍精度（double）とし，倍精度で計算する．$\|\boldsymbol{x}\|_2$ は $\boldsymbol{x}$ の各要素の2乗和の平方根である.
  - 正規化されたベクトルの要素は
    $\tilde{x}[i] = (i+1)/\sqrt{n(n+1)(2n+1)/6}$
    であるので（簡単に計算できる），プログラムではこの値と比較することによって，正しく計算ができている（Error = 0.0）ことを確認している.

```
$ cp /tmp/Summer/M-3/normalize.c ./
$ icc -O3 normalize.c
$ ./a.out
Error = 0.000000
```

演習10-2　normalize.cを並列化せよ

- 各プロセスが自分の担当分の要素について，2乗和を計算する.
- 各プロセスの担当する要素（nprocs はMPIプロセス数）
  - `istart = (n/nprocs)*myrank`
  - `iend   = (n/nprocs)*(myrank+1)-1`



- ベクトルの格納方法
  - 各プロセスは長さ n の配列を持ち，そのうち自分の担当部分のみを使う

- 全プロセスの総和を MPI_Allredcue関数で求め，各プロセスはその総和を使って，自分の担当する要素について正規化を行う.
- n=1000 としてプロセス数を1，2，4，8と変えて計算せよ．結果の核にも工夫すること.

時間計測関数，バリア同期関数

計算時間の計測

- 並列計算の目的は，計算時間の短縮にある．
  - 大規模問題を解くためにメモリ容量を増やすことも目的の一つ

- 同じ結果が得られるが，アルゴリズムや書き方が異なったプログラムのうち，どれが一番良いか？

  ⇨ （正しい結果が得られるならば）計算時間の短いものが良いはず．

- 計算時間を計測して比較する．

計算時間を計測する方法

```
double time0, time2;                          計測のための変数を倍精度実数型で宣言する.
      •
      •
MPI_Barrier( MPI_COMM_WORLD )                 MPI_Barrier関数で，計測開始の足並みを揃える.
time0 = MPI_Wtime()                           mpi_wtime関数で開始時刻をtime0に設定

 (計測する部分)

MPI_Barrier( MPI_COMM_WORLD )                 全プロセスで終了の足並みを揃える.
time1 = mpi_wtime()                           mpi_wtime関数で終了時刻をtime1に設定

 (time1-time0 を出力する)                     time1-time0が計測した部分の計算時間となる.
```

MPI_Barrier(comm)　：　バリア同期関数

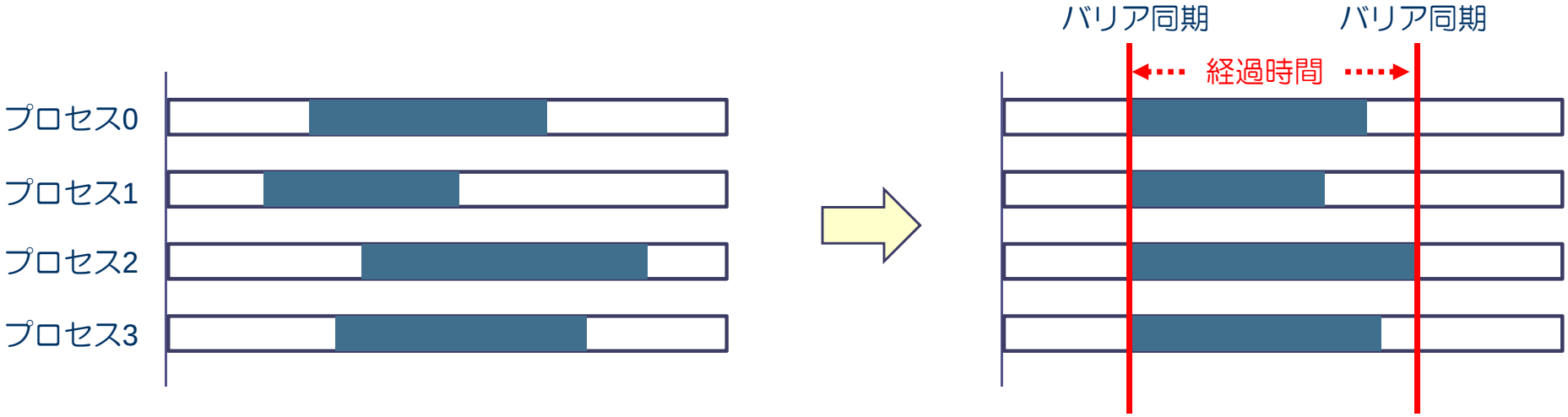
- comm:　　コミュニケータ（例えば，MPI_COMM_WORLD）

var = MPI_Wtime()　：倍精度実数を返す関数（double var;）

時間計測のイメージ

- 各プロセスでの計算時間の測定関数
  - `MPI_Wtime()`
    - ある時点を基準とした経過秒数を倍精度実数型で返す関数

- プログラムのある区間の計算時間の測定
  - プログラムの実行は各プロセスで独立なので，開始時間や終了時間が異なる．
  - ある部分の計算時間の計測では，バリア同期（`MPI_Barrier`）により測定開始と測定終了の足並みを揃えて，計測する．



演習10-3　並列化normalize.c の実行時間計測

- 並列化したベクトルの正規化プログラムの実行時間を計ってみる.
  - 計測範囲は，各プロセスが総和を求めるところの直前から，正規化した直後まで，とする.
- n = 50,000，100,000，500,000 に対して，並列数を1，2，4，8，16 と変化させて時間を計測せよ.
  - 1プロセスの実行時間をT(1)，mプロセスの実行時間を T(m) とするとき，速度向上率 T(1)/T(m) を求めよ（次の表参照）.
  - また，速度向上率のグラフを gnuplotで画け.
    - gnuplotは，Linuxで動作するグラフ描画ソフトウェア.

表：ベクトルサイズと並列数の関係

	n=50,000		n=100,000		n=500,000	
並列数 (m)	計算時間（秒） T(m)	速度向上率 T(1)/T(m)	計算時間（秒） T(m)	速度向上率 T(1)/T(m)	計算時間（秒） T(m)	速度向上率 T(1)/T(m)
1		1.000		1.000		1.000
2						
4						
8						
16						

参考：Fortran版

計算時間を計測する方法

```
real(DP) :: time0, time2                          計測のための変数を倍精度実数型で宣言する.
      •
      •
call mpi_barrier( MPI_COMM_WORLD, ierr )          MPI_barrier関数で，計測開始の足並みを揃える.
time0 = mpi_wtime()                               mpi_wtime関数で開始時刻をtime0に設定

 (計測する部分)

call mpi_barrier( MPI_COMM_WORLD, ierr )          全プロセスで終了の足並みを揃える.
time1 = mpi_wtime()                               mpi_wtime関数で終了時刻をtime1に設定

 (time1-time0 を出力する)                          time1-time0が計測した部分の計算時間となる.
```

`mpi_barrier(comm, ierr)` ： バリア同期関数

- comm: コミュニケータ（例えば，MPI_COMM_WORLD）
- ierr: 戻りコード（整数型）

var = `mpi_wtime()` ：倍精度実数を返す関数