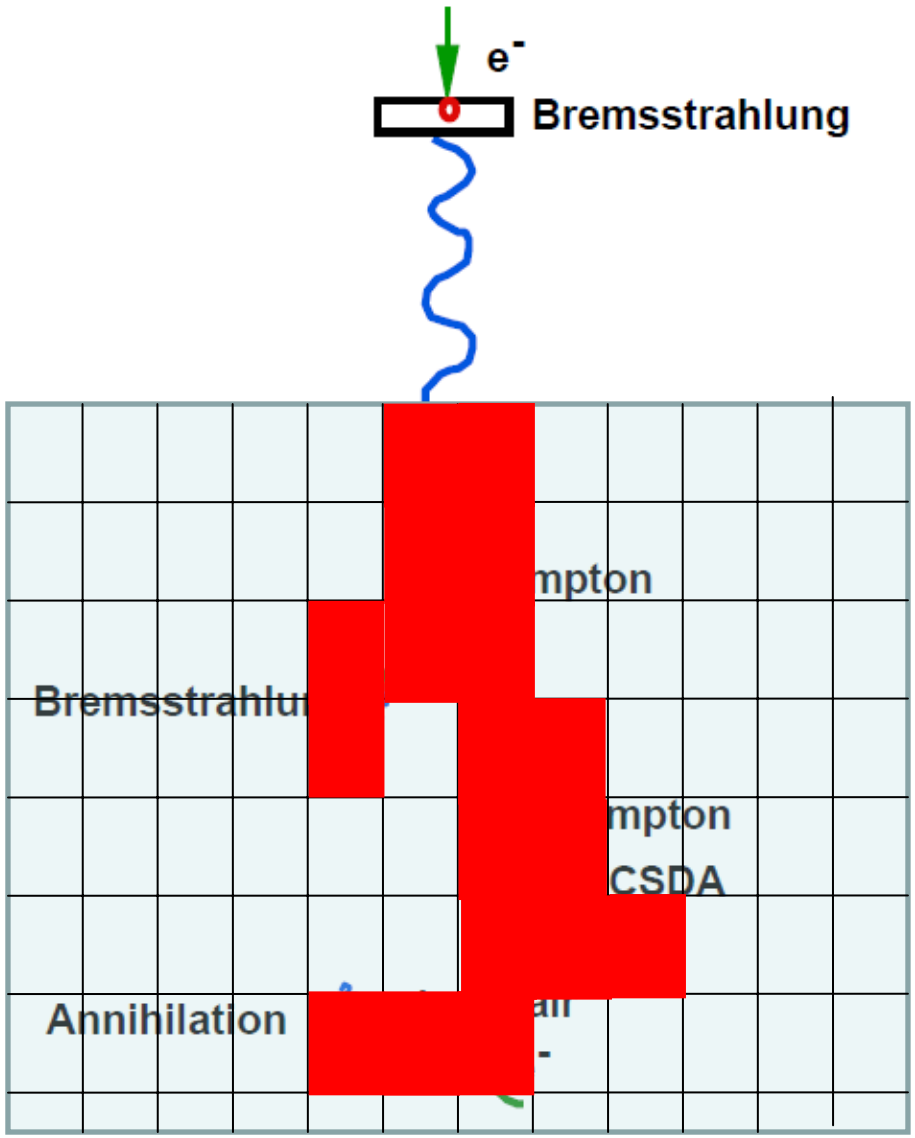
MPIを利用した並列計算

京都大学大学院医学研究科
画像応用治療学・放射線腫瘍学
石原 佳知
y.ishi@kuhp.kyoto-u.ac.jp

MC計算と高速化

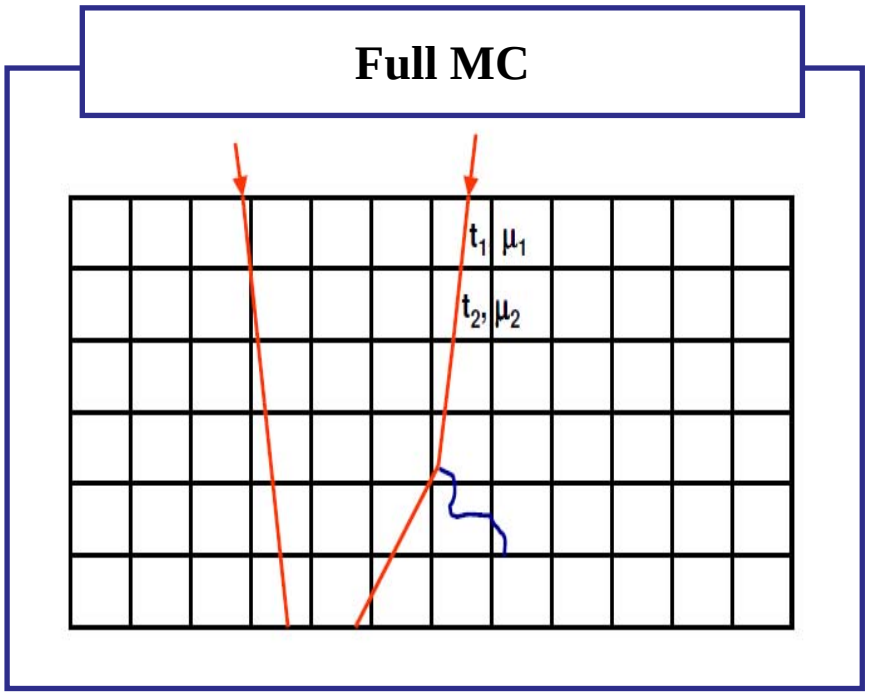


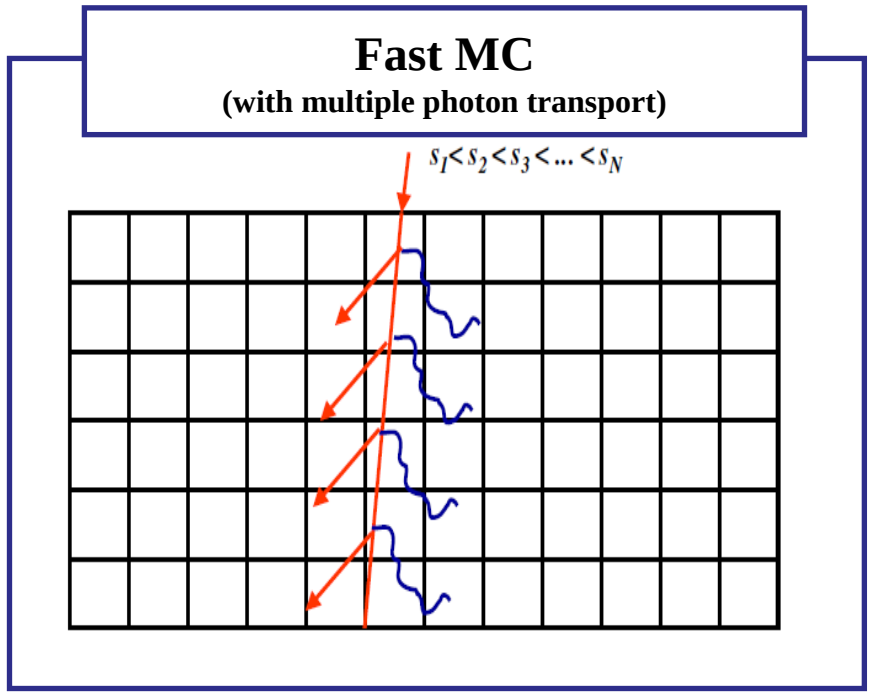
- MC計算では統計誤差を満たす(目的となるregionのイベントを増やす)ため反復計算(複数回の粒子移動計算)を行う
- この反復計算のため計算終了まで膨大な計算時間が必要である
- 計算時間短縮のため様々な高速化方法が存在する

計算高速化手法

- Splitting
- STOPS (Simultaneous Transport of Particle Sets)
- Russian roulet
- History repetition
- Range rejection
- Recycling
- Parallel Processing ?

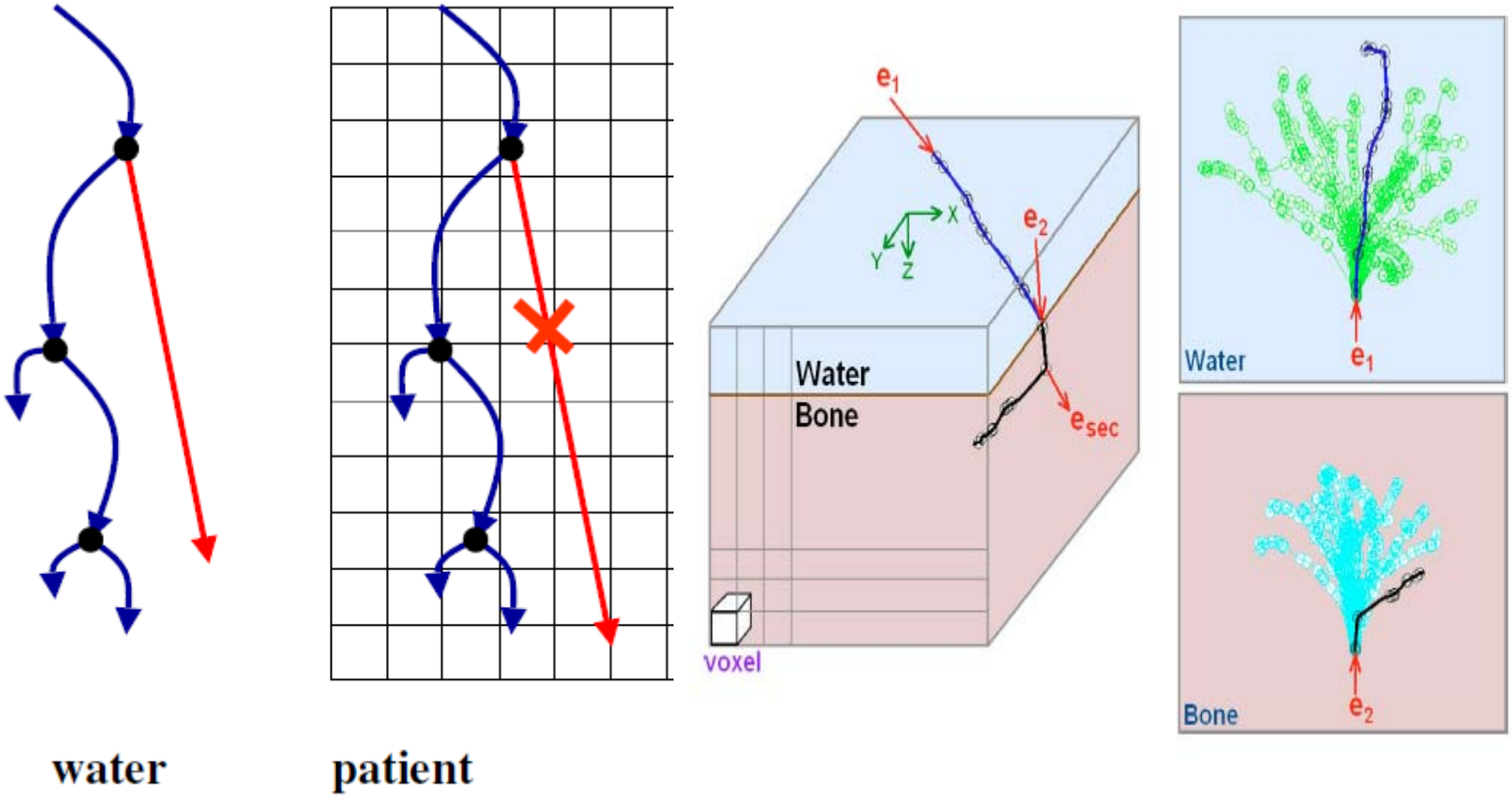
計算高速化手法





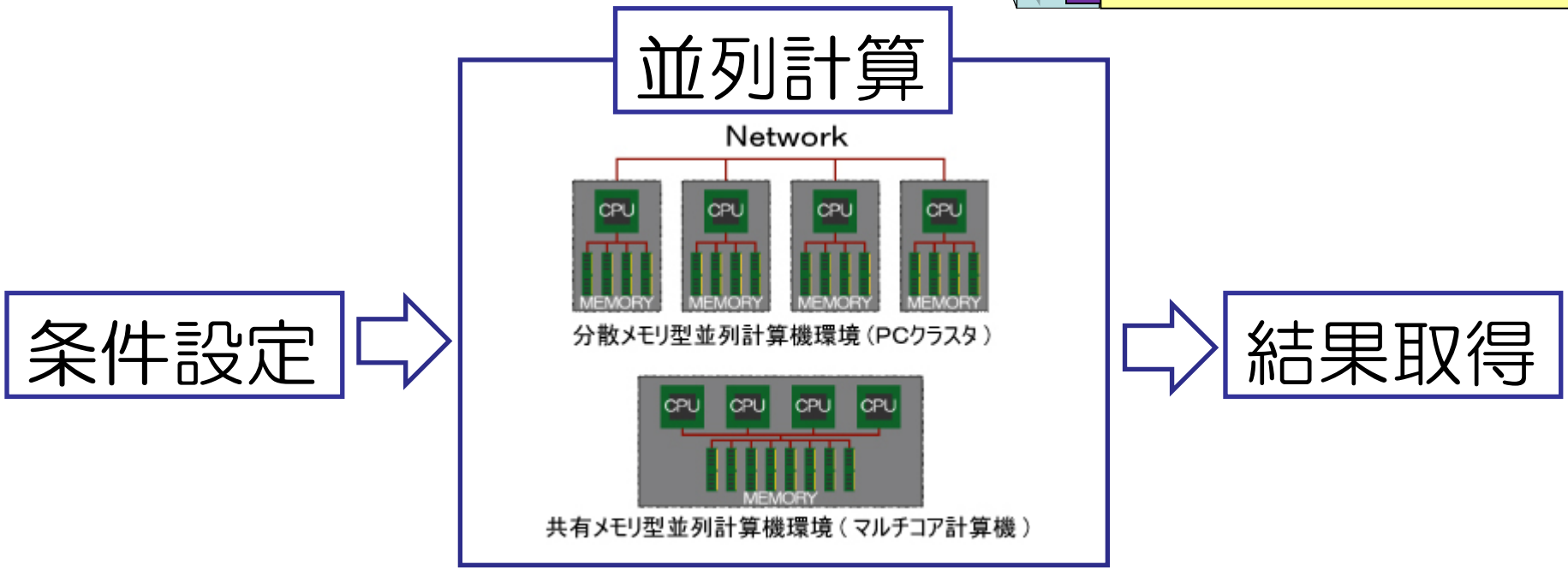
- 強制的に粒子の発生確率を大きくし、
  少ない粒子で大量の相互作用を発生させる

計算高速化手法



- 事前に取得した粒子データをそのまま適用

計算高速化手法



- 処理を複数のCPUに分散し計算時間を短縮
- Core数がN個であれば計算時間は1/Nとなる
- MC計算においては単純にN倍の高速化

計算高速化手法

- 殆どの高速化手法は物理的な歪が生じるために計算精度が落ちる可能性がある

- 並列計算においては計算機さえあれば精度を落とすことなく高速化が可能である
  - MPI (Message Passing Interface)
  - HPF (High Performance Fortran)
  - PVM (Parallel Virtual Machine)
  - Linda

MPI

- 分散メモリ型の並列計算機で2つ以上のプロセス間でのデータをやりとりするために用いるメッセージ通信操作の仕様標準
- この方式を使ったプログラムを共有メモリ型の並列計算機で実行することも可能
- MPI は新しいプログラミング言語ではなく、C または Fortran から呼び出すサブプログラムのライブラリである

MC計算を並列化するために必要な処理

- 並列化用乱数の導入
  - 各coreにおいて異なった乱数でなければならない
  - 並列計算専用乱数の導入
  - 例：SPRNG (http://sprng.cs.fsu.edu/)
- 統計誤差の設定
  - History-by-historyによる評価
  - Walters B R B, Kawrakow I and Rogers D W O, 2002, History by history statistical estimators in the BEAM code system, *Med. Phys.* ***29 2745–52***
- 並列化部分の設定

並列化部分の設定

- .fファイルにMPIのコードを組込む
- MPIの開始と終了部分を設定

#include <sprng_f.h>;　並列化開部分の設定
#include <sprng_f.h>;　並列化開部分の設定
の設定
ZE(MPI_COMM_WORLD,NPROCS,IERR);
MM_WORLD,NPROCS,IERR);
NK(MPI_COMM_WORLD,MYRANK,IERR);
WORLD,MYRANK,IERR);
ORLD,MYRANK,IERR);
RR);
R);
L MPI_FINALIZE(IERR);並列化終了部分の設定
MPI_FINALIZE(IERR);並列化終了部分の設定
MPI_FINALIZE(IERR);並列化終了部分の設定

MPI組込みの概略 (使用するのはこれだけ!)

MYRANK: プロセス番号のための変数, NPROCS: 総プロセス数のための変数

↑このふたつの変数を定義する

#include <sprng_f.h>; 並列化開部分の設定

CALL MPI_INIT(IERR); 変数の初期化

CALL MPI_COMM_SIZE(MPI_COMM_WORLD,NPROCS,IERR);

総プロセス数の取得

CALL MPI_COMM_RANK(MPI_COMM_WORLD,MYRANK,IERR);

自分のプロセス番号の取得

****各ノードにおけるファイル・変数の設定****

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR); プロセスの同期

****並列化計算(粒子輸送計算)****

CALL MPI_REDUCE(A(MYRANK,I),ASUM(I),1,MPI_REAL,MPI_SUM,
0,MPI_COMM_WORLD,IERR); データの結合 (例:outputで使用)

CALL MPI_FINALIZE(IERR);並列化終了部分の設定

初期化

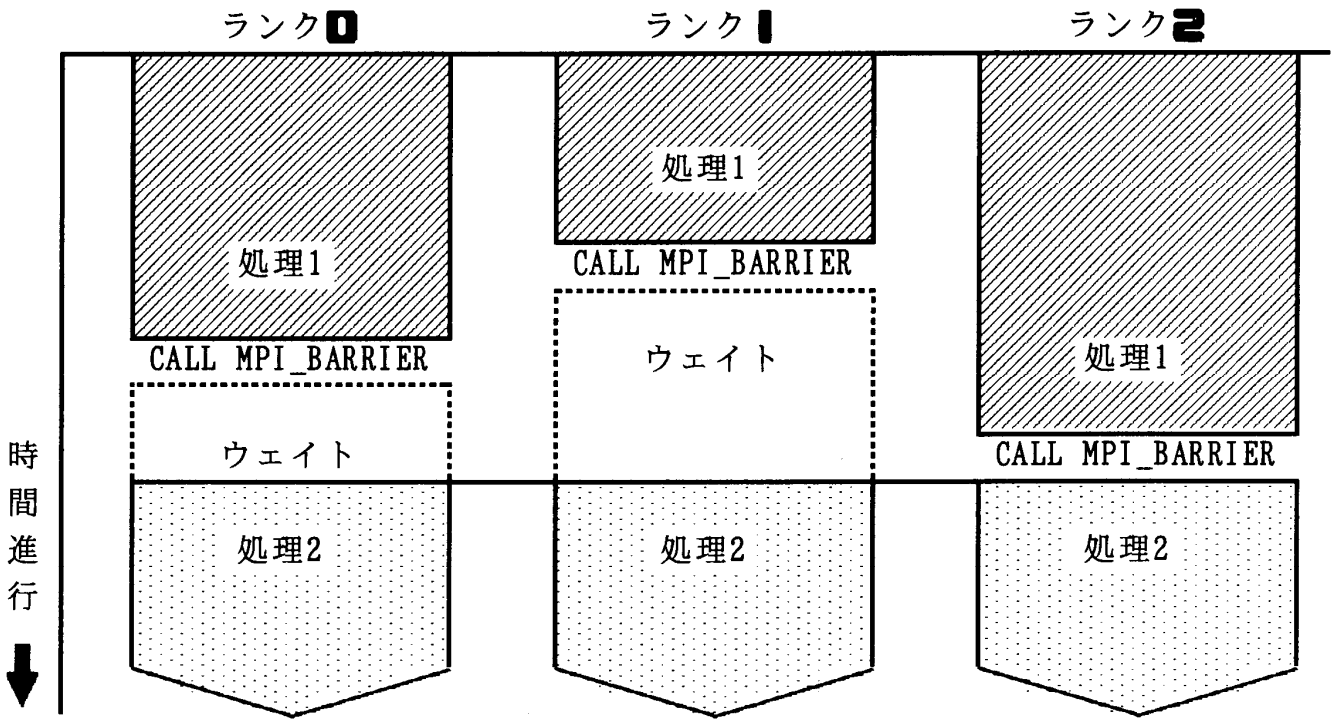
#include <sprng_f.h>; 並列化開部分の設定
CALL MPI_INIT(IERR); 変数の初期化
CALL MPI_COMM_SIZE(MPI_COMM_WORLD,NPROCS,IERR);
総プロセス数の取得
CALL MPI_COMM_RANK(MPI_COMM_WORLD,MYRANK,IERR);
自分のプロセス番号の取得

- MPIの開始部分の設定
- この4文はそのまま使用する

処理の同期

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR);

図1

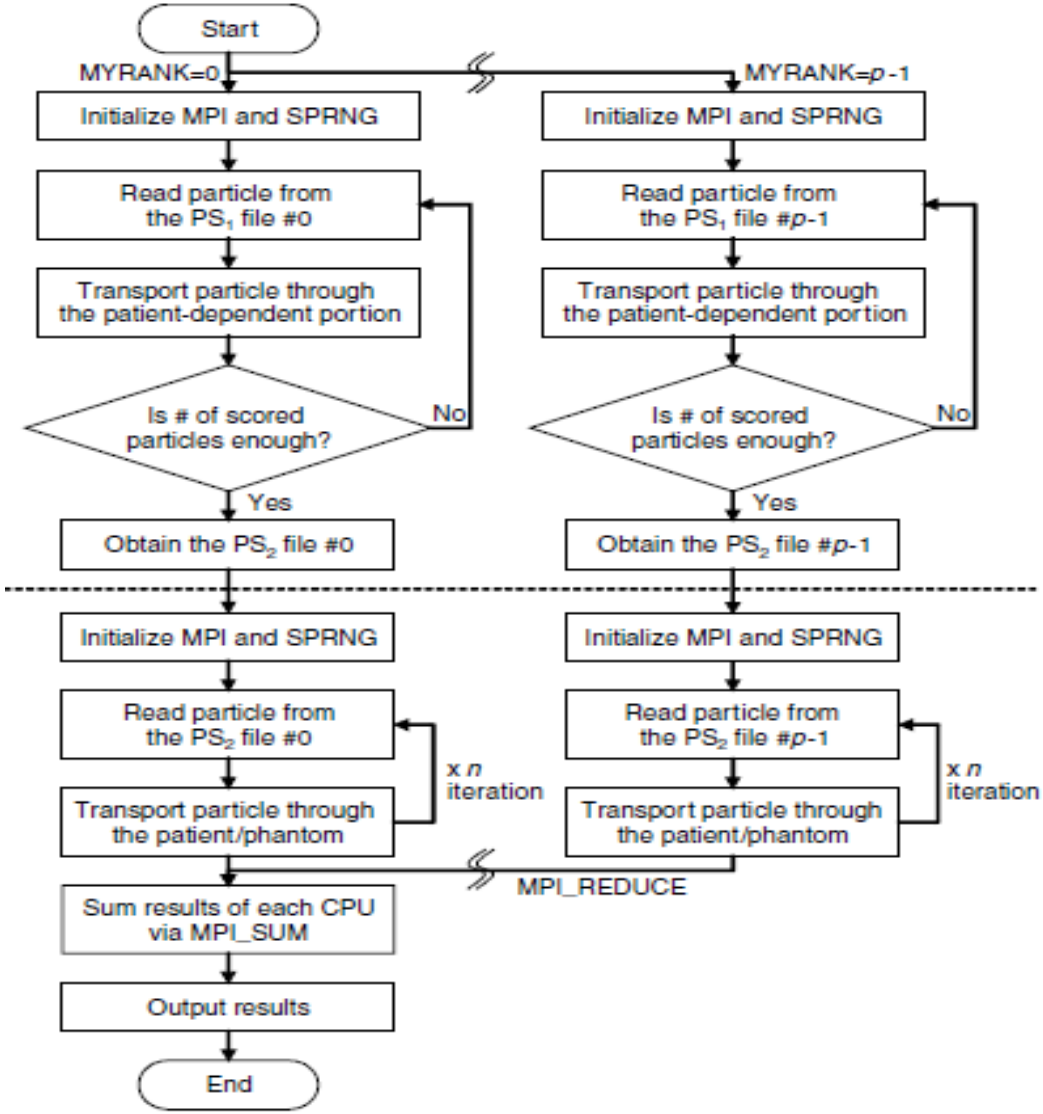


データの結合


```
CALL MPI_REDUCE(A(MYRANK),ASUM,1,MPI_REAL,MPI_SUM,
  0,MPI_COMM_WORLD,IERR);
```

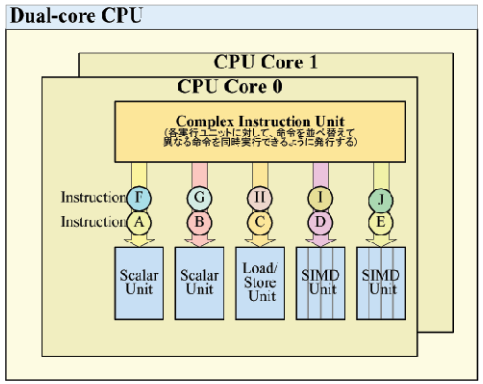
- 各プロセスでの結果A(MYRANK)を0番のプロセスにおいてASUMの変数にreal型(MPI_REAL)として累積(MPI_SUM)する

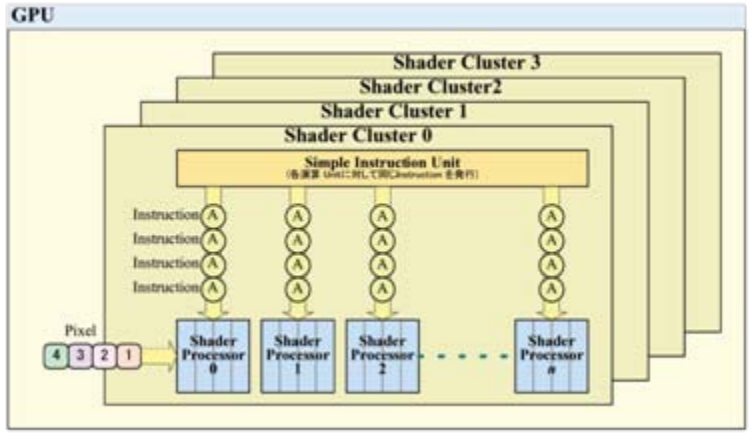
MPI計算例概略図

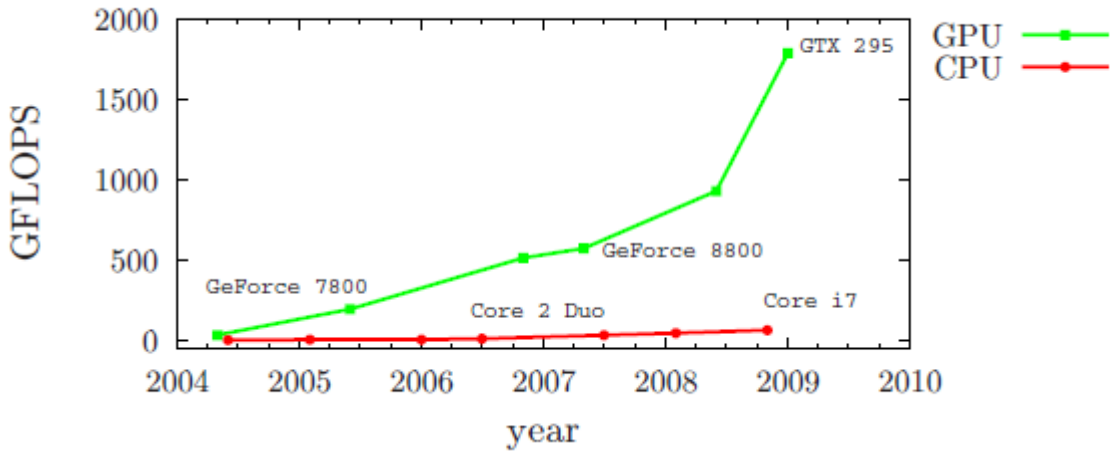

Start
MYRANK=0
MYRANK=p-1
Initialize MPI and SPRNG
Initialize MPI and SPRNG
Read particle from the PS1 file #0
Read particle from the PS1 file #p-1
Transport particle through the patient-dependent portion
Transport particle through the patient-dependent portion
Is # of scored particles enough?
No
Is # of scored particles enough?
No
Yes
Yes
Obtain the PS2 file #0
Obtain the PS2 file #p-1
Initialize MPI and SPRNG
Initialize MPI and SPRNG
Read particle from the PS2 file #0
Read particle from the PS2 file #p-1
x n iteration
x n iteration
Transport particle through the patient/phantom
Transport particle through the patient/phantom
MPI_REDUCE
Sum results of each CPU via MPI_SUM
Output results
End

その他の並列化 (GPU)

CPUとGPUの命令実行の違い







その他の並列化 (GPU)

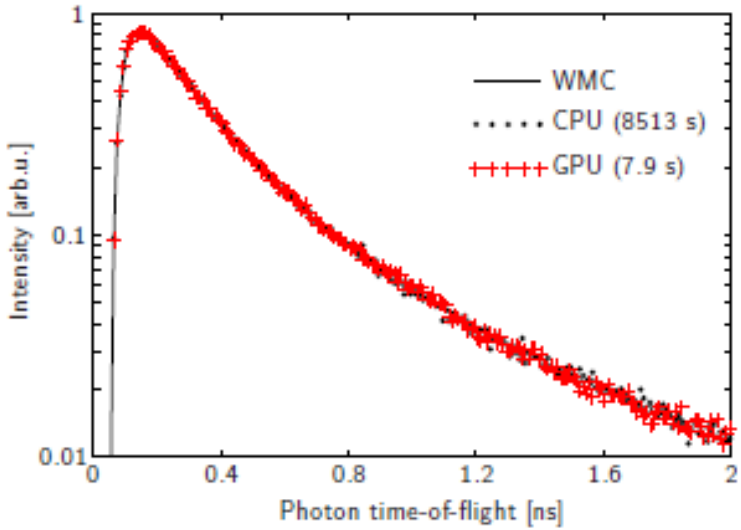


Fig. 1 TOF histogram of three different MC simulation methods. WMC (simulated using double precision) is used for reference. The WMC results comprise 2,133,796 detected photons (building the database took 21 h using all four cores of an Intel Core 2 Quad 2.4 GHz). The GPU and CPU results (both simulated using single precision), which should be statistically equivalent, both comprise about 326,000 detected photons out of about 13.8×10^6 full photon paths simulated. The simulation times were 7.9 s for the GPU (a NVIDIA 8800GT) and 8513s for the CPU (an Intel Pentium 4 HT 3.4 GHz), i.e., the GPU proves to be 1080× faster. Comparing the CPU and GPU results to the WMC result validates the MC code for both implementation and confirms that single precision is sufficient as long as proper care is taken during calculations.

まとめ

- 並列化計算の分野では異なった処理をいかに効率よく並列するかが焦点、つまり反復(同一)処理が多いMCは並列化に最適
- 並列化では粒子輸送過程での物理的な歪が生じず精度を担保しやすい
- 並列化のための乱数、統計誤差の評価の組込みが必須
- MPIは並列化の中でも確立された手法であり資料等の入手が容易(http://accc.riken.jp/HPC/training.html)