TOHOKU UNIVERSITY

Cyberscience Center

NEC \Orchestrating a brighter world

2020 年度
並列プログラミング入門 I
（並列化概要）

２０２０年　１２月　９日

東北大学サイバーサイエンスセンター
日本電気株式会社

本資料は，東北大学サイバーサイエンスセンターとＮＥＣの共同により作成された.
無断転載等は，ご遠慮下さい.

- 並列化概要編
- OpenMPプログラミング編
- MPIプログラミング編

並列化概要

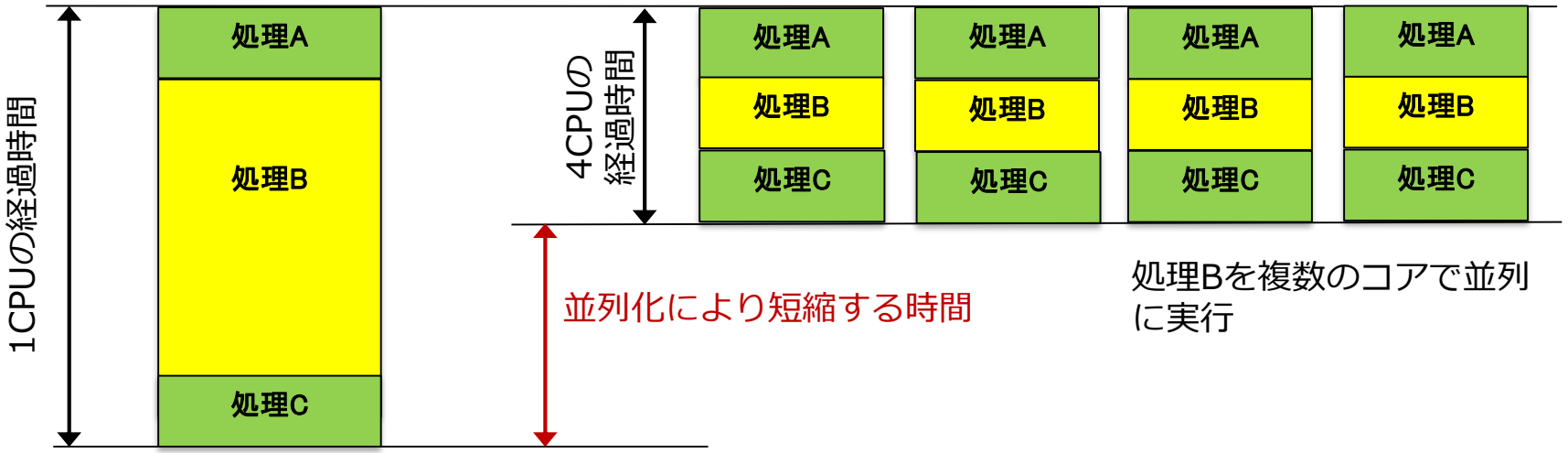
並列化とは

並列処理・並列実行

- 仕事（処理）を複数のコアに分割し，同時に実行すること

並列化

- 並列処理を可能とするために，処理の分割を行うこと



並列化の効果

行列積プログラム

```
      implicit real(8)(a-h,o-z)
      parameter ( n=7680 )
      real(8) a(n,n),b(n,n),c(n,n)
      real(4) etime,cp1(2),cp2(2),t1,t2,t3
      do j = 1,n
        do i = 1,n
          a(i,j) = 0.0d0
          b(i,j) = n+1-max(i,j)
          c(i,j) = n+1-max(i,j)
        enddo
      enddo
      write(6,50) ' Matrix Size = ',n
   50 format(1x,a,i5)
      t1=etime(cp1)
      do j=1,n
        do k=1,n
          do i=1,n
            a(i,j)=a(i,j)+b(i,k)*c(k,j)
          end do
        end do
      end do
      t2=etime(cp2)
      t3=cp2(1)-cp1(1)
      write(6,60) ' Execution Time = ',t2,' sec',' A(n,n) = ',a(n,n)
   60 format(1x,a,f10.3,a,1x,a,d24.15)
      stop
      end
```

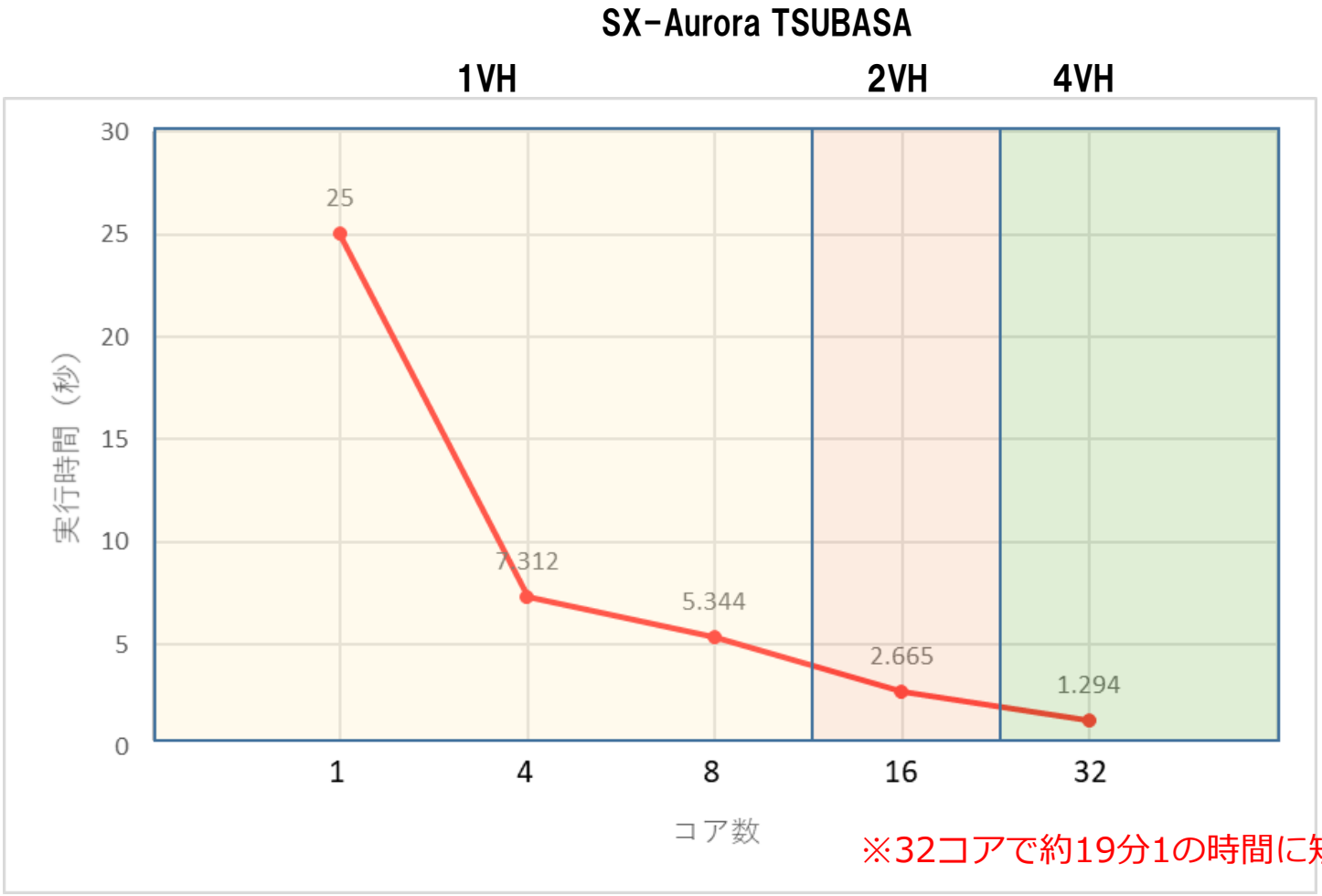
- SX-Aurora TSUBASA　1コアの実行時間は25秒

```
 Matrix Size =  7680
  Execution Time =     25.000 sec  A(n,n) =     0.768000000000000D+04
********  Program  Information  ********
  Real Time (sec)                          :            24.995688
  User Time (sec)                          :            24.994095
  Vector Time (sec)                        :            24.992846
  Inst. Count                              :          28907976960
  V. Inst. Count                           :           7080207403
  V. Element Count                         :        1812533093390
  V. Load Element Count                    :         905969664256
  FLOP Count                               :         905969664150
  MOPS                                     :            91517.439002
  MOPS (Real)                              :            91509.617724
  MFLOPS                                   :            36248.138469
  MFLOPS (Real)                            :            36245.040625
  A. V. Length                             :              256.000000
  V. Op. Ratio (%)                         :               99.045716
  L1 Cache Miss (sec)                      :                0.000371
  CPU Port Conf. (sec)                     :                0.000000
  V. Arith. Exec. (sec)                    :               11.712283
  V. Load Exec. (sec)                      :               13.280471
  VLD LLC Hit Element Ratio (%)            :               49.993697
  FMA Element Count                        :          452984832000
  Power Throttling (sec)                   :                0.000000
  Thermal Throttling (sec)                 :                0.000000
  Memory Size Used (MB)                    :             1860.000000
```

- 複数のコアを用いることで実行時間の短縮が可能に

並列化の効果

- 並列化により複数のコアを利用し，実行時間を短縮
- MPIを用いることで，複数のノードが利用可能(さらに時間短縮)



※32コアで約19分1の時間に短縮

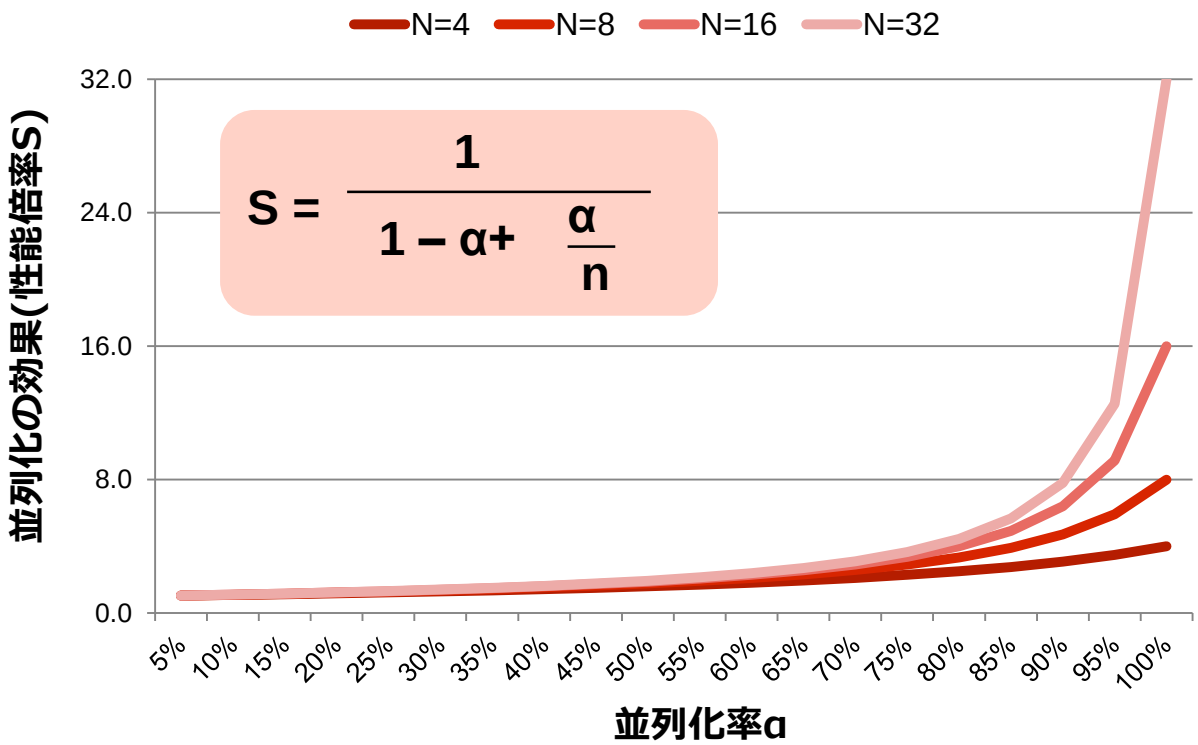
並列化の効果

- 並列に実行可能(あるいは効果のある)部分と並列に実行不可(あるいは効果のない)部分を見つけ，並列に実行可能部分を複数のコア(CPU)に割り当てる.
- できるだけ多くの部分を並列化の対象としなければ，コア数に応じた効果が得られない.

$$\text{並列化率}\alpha = \frac{\text{並列化対象部分の処理時間}}{\text{全体の処理時間}}$$

(並列化の対象部分と非対象部分の時間の合計)

並列化の効果



並列化率と並列化の効果の関係(アムダールの法則)

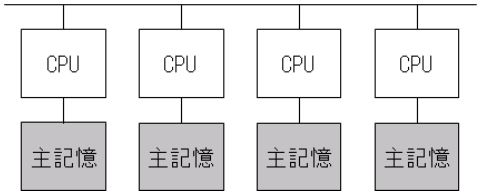
- Nはコア(CPU)数
- 並列化率が100%から下がるにしたがって性能倍率は急速に低下する

並列化率100%はあり得ない(データの入出力で必ず逐次処理発生)が，可能な限り100%に近づかなければ並列化の大きな効果は得られない

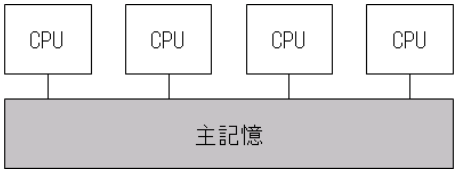
並列処理モデル

コンピュータアーキテクチャに応じた処理の分担(分割)のさせ方によって幾つかの並列処理がある

1. 分散メモリ並列処理
   - LX406 Rz-2(マルチノード)
   - SX-Aurora TSUBASA(複数VE)



2. 共有メモリ並列処理
   - LX406 Rz-2(シングルノード)
   - SX-Aurora TSUBASA(1VE)



- MPI(Message Passing Interface)は分散メモリ並列処理のための並列手法
- OpenMPは共有メモリ並列処理のための並列手法

並列処理モデル

並列処理モデルの特徴

	共有メモリ並列処理	分散メモリ並列処理
メモリ空間	すべてのコアが同じメモリ空間をアクセス可能	ネットワーク上のすべてのメモリ空間をデータ通信によりアクセス可能
利用可能な並列化手法	自動並列化 OpenMP並列化 MPI並列化	MPI並列化 (※1)
利用可能なSX-Aurora TSUBASAのコア数	8コア(1VE)	2,048コア(32VH)
利用可能なSX-Aurora TSUBASAのメモリ容量	48GByte	約12TByte

※1：共有メモリ内は共有メモリ並列処理，分散メモリ間は分散メモリ並列処理のハイブリッド並列処理が可能

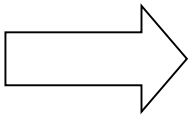
並列化の対象

物理現象

空間的並列性

現象の並列性

プログラム化



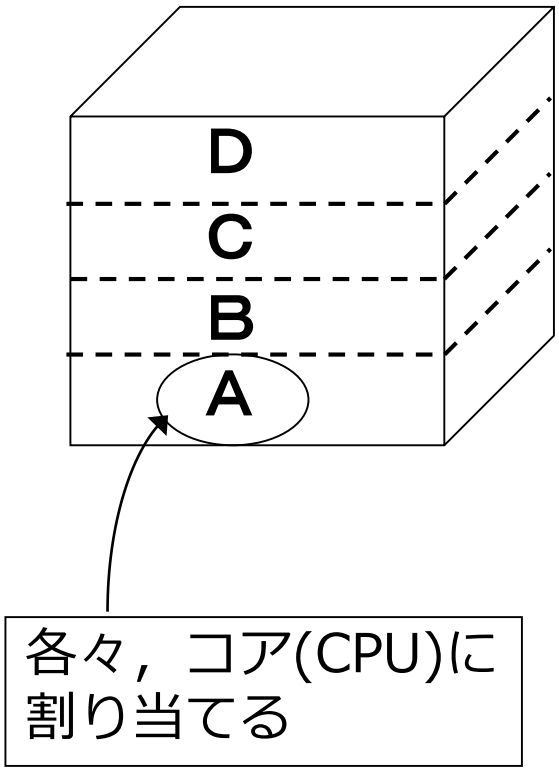
プログラム

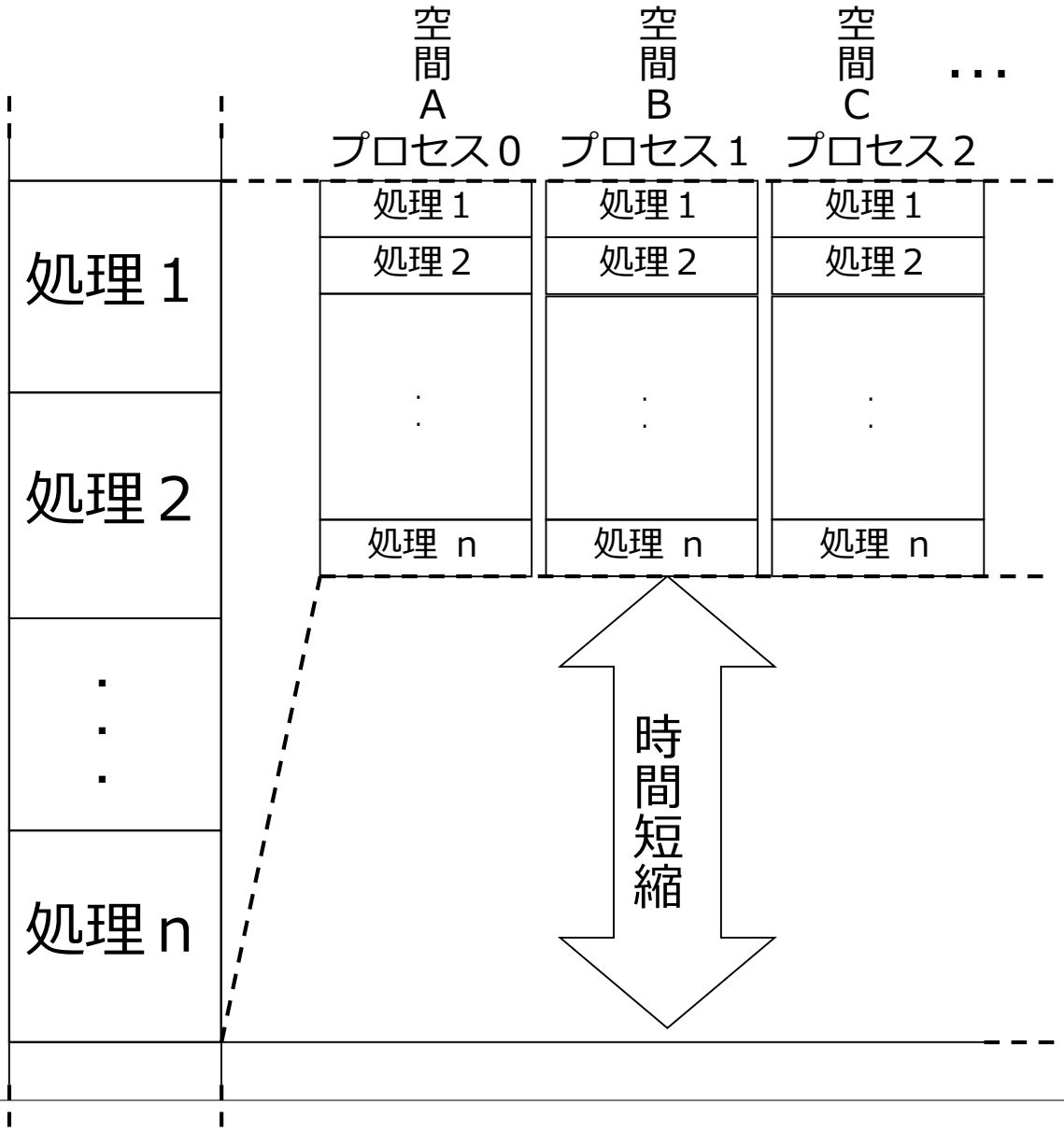
空間による並列化
（**領域分割法**）

処理による並列化

空間による並列化（イメージ）

領域分割法



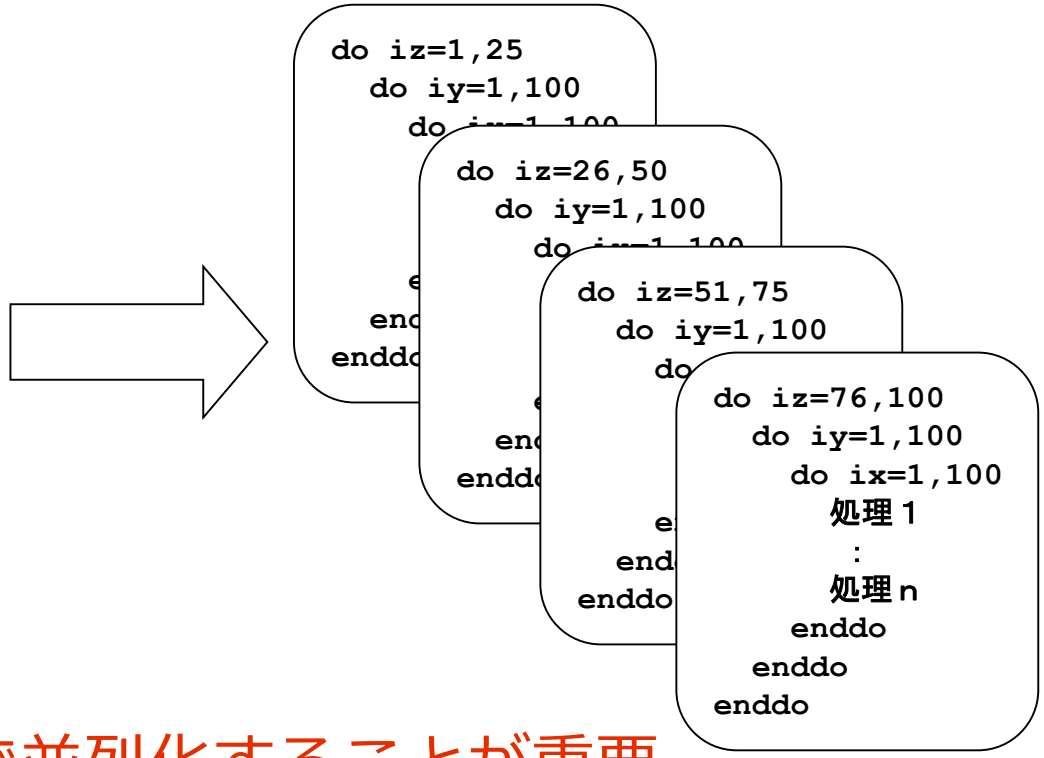


空間による並列化の例

DOループ（FORTRAN）単位での並列処理

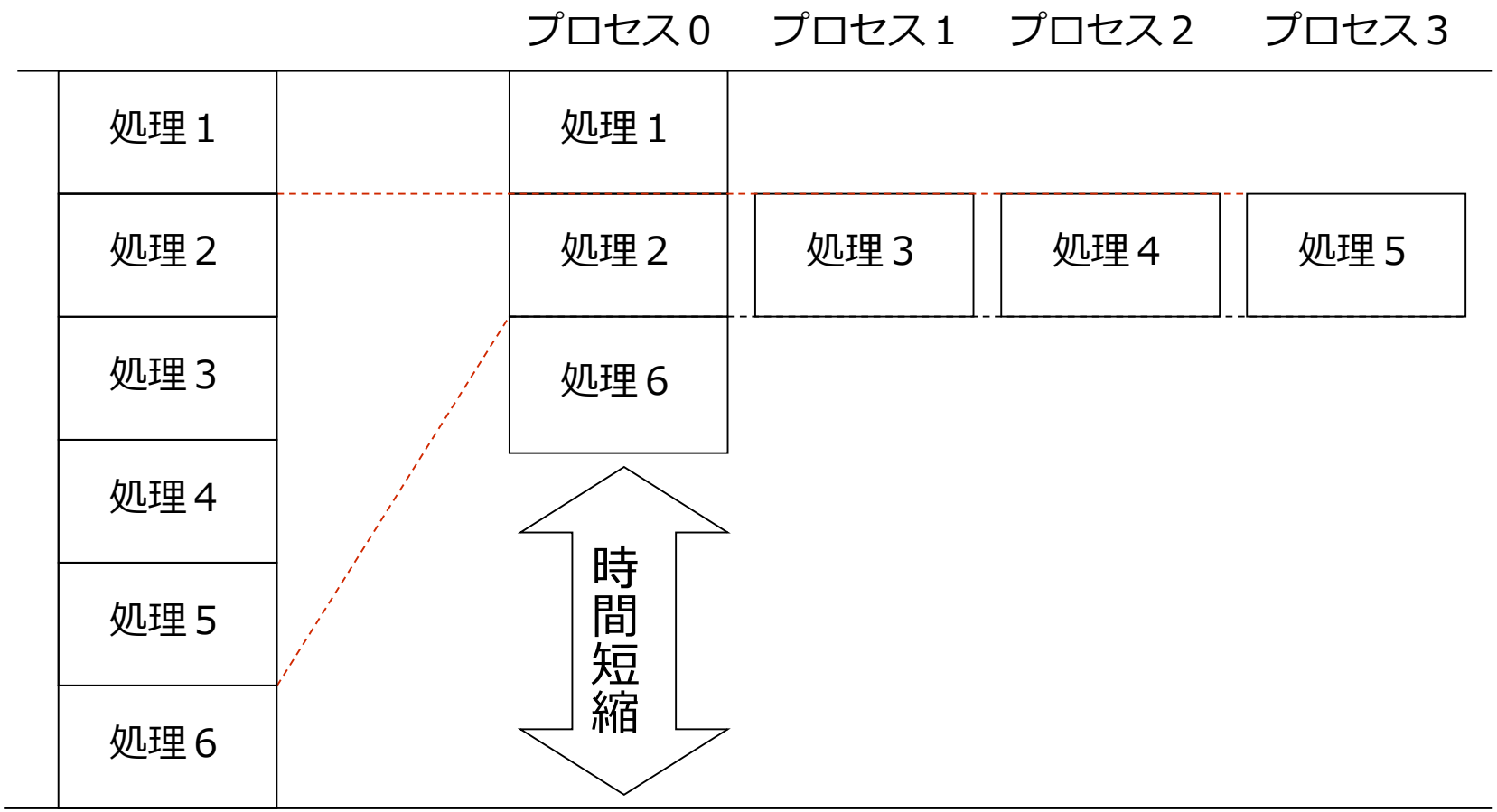
例）領域分割法

```
do iz=1,100
  do iy=1,100
    do ix=1,100
      処理１
        :
      処理ｎ
    enddo
  enddo
enddo
```



より外側のループで並列化することが重要

処理による並列化（イメージ）

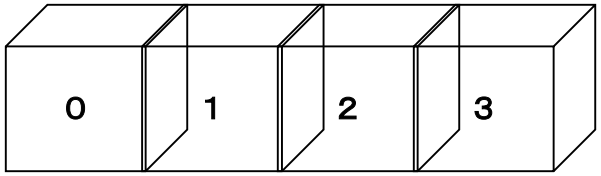


空間分割の分類

① ブロック分割

- 空間を分割数の塊に分割する

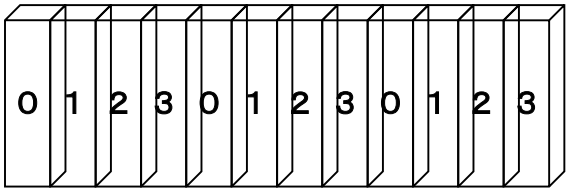
例）４分割



② サイクリック分割

- 帯状に細分し，巡回的に番号付ける

例）４分割

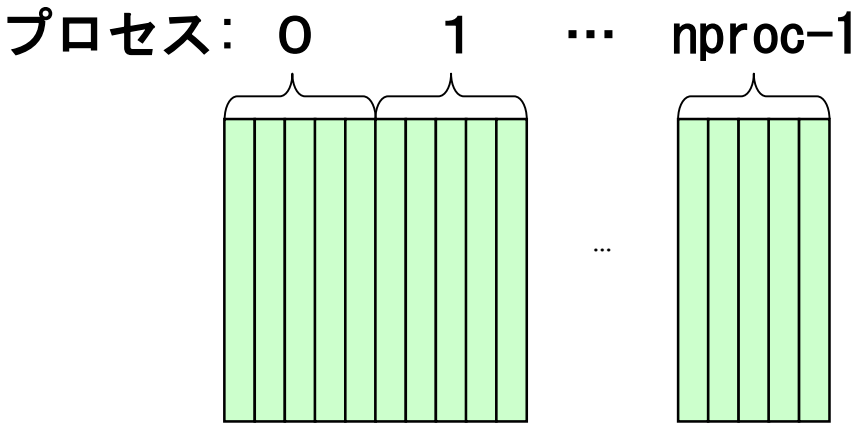


ブロック分割

処理量が均等なループを分割する場合

```
do i=1,100
...
enddo
```

繰り返し数をプロセス毎に均等に割り当てる




```
do i=1,25
...
enddo
```

```
do i=26,50
...
enddo
```

```
do i=51,75
...
enddo
```

```
do i=76,100
...
enddo
```

サイクリック分割

処理量が不均等なループを分割する場合

```
do i=1,n
・・・
enddo
```

繰り返し数をプロセス毎に
巡回的に割り当てる

```
common /commpi/ myrank,nprocs
do i=myrank+1,n,nprocs
 ・・・
enddo
```

プロセス：０ １ ２ ３ ０ １ ２ ３