計算科学演習Ｉ　（第3回）

並列計算とは

神戸大学大学院システム情報学研究科
横川 三津夫

※ ブラウザ（外部サイトにはアクセスしないように）を立ち上げて
http://exp.cs.kobe-u.ac.jp/wiki/comp_practice/
の「2. 並列計算とは」の項を参照のこと.

講義概要

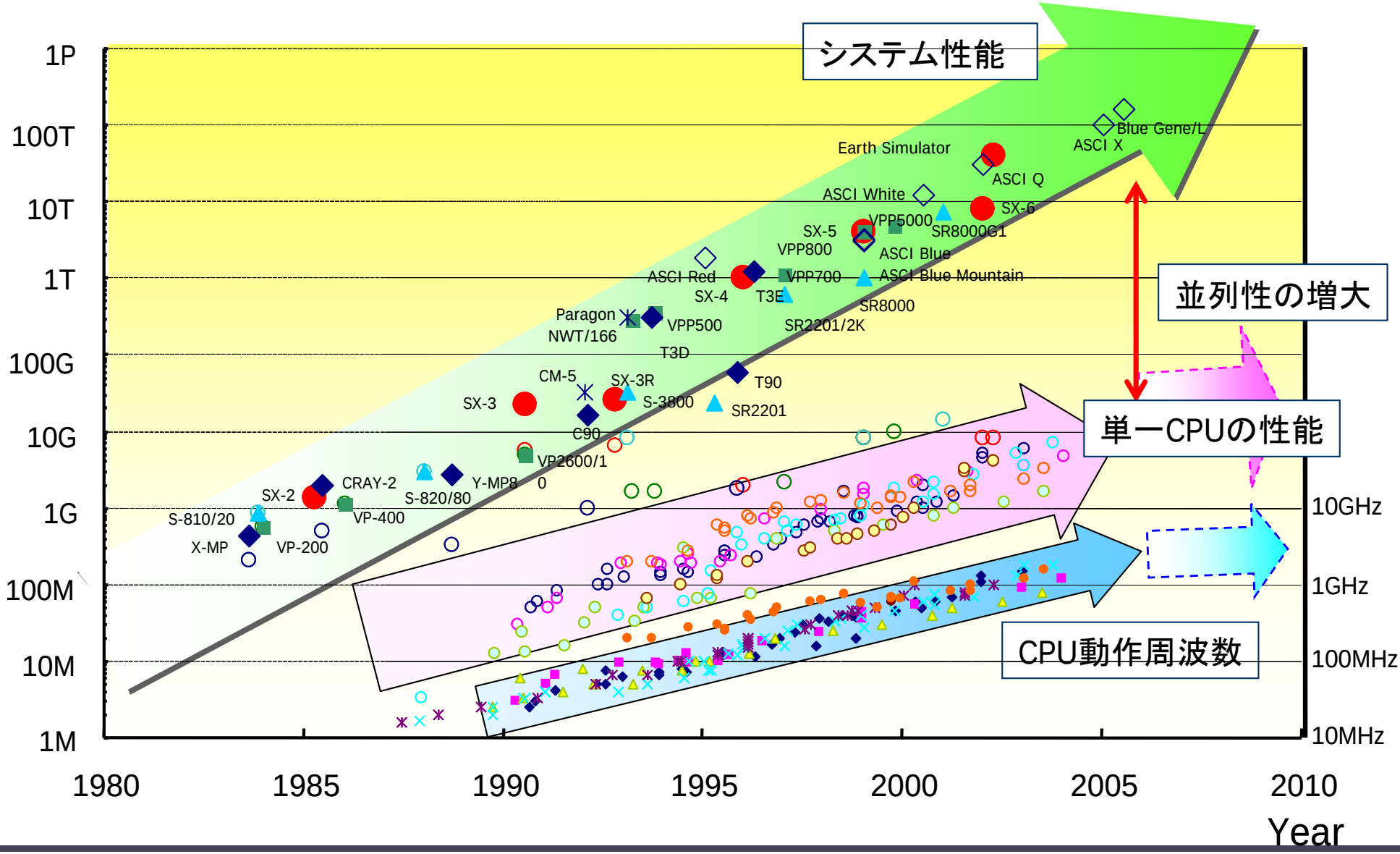
- なぜ並列計算なのか？
- 並列計算機のアーキテクチャ
- 基本的な並列化手法
- 並列計算の性能向上

なぜ並列計算なのか？

- 計算性能の要求がますます増大している.
  - 大規模シミュレーション
  - グラフィックス

- 単一CPUの速度向上の限界
  - CPU製作技術の限界
    - 国際半導体技術ロードマップでは，9nmくらいまで？
  - 信号伝播の限界（光の速度）
    - 1ナノ秒で30cm
  - 電力性能比の向上の限界

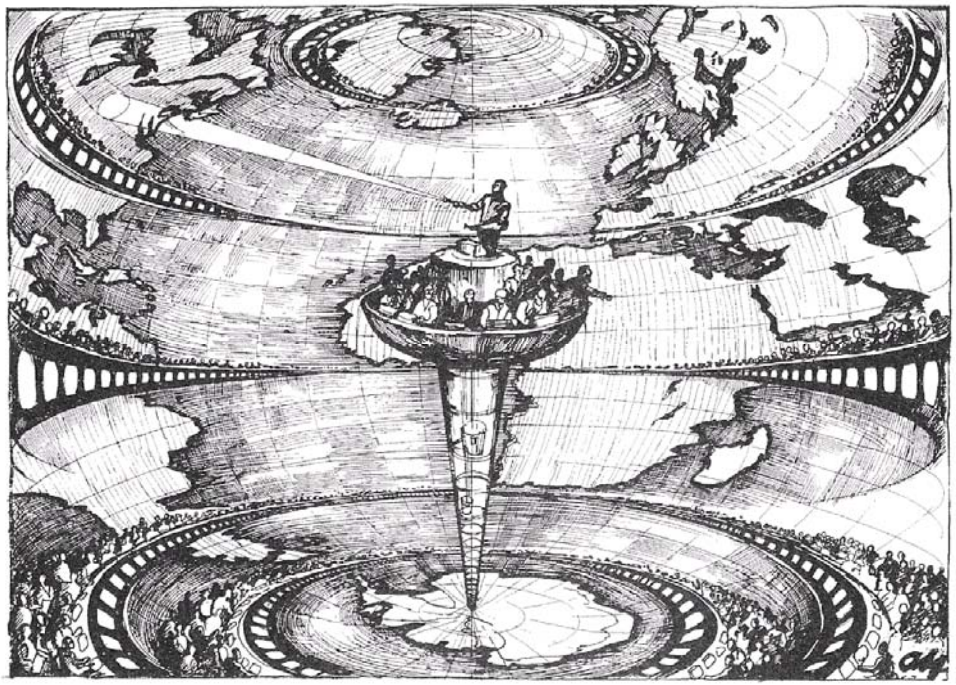
単一CPUとシステム性能の推移



リチャードソンの夢

- 大きな円形劇場に64,000人を集めて，一人一人が地球のある場所の天気を計算し，周りの人と計算結果を交換しながら，全体の計算をすれば、天気予報が出来る.

— by L.F. Richardson, *Weather Prediction by Numerical Process,* Cambridge, University Press (1922)



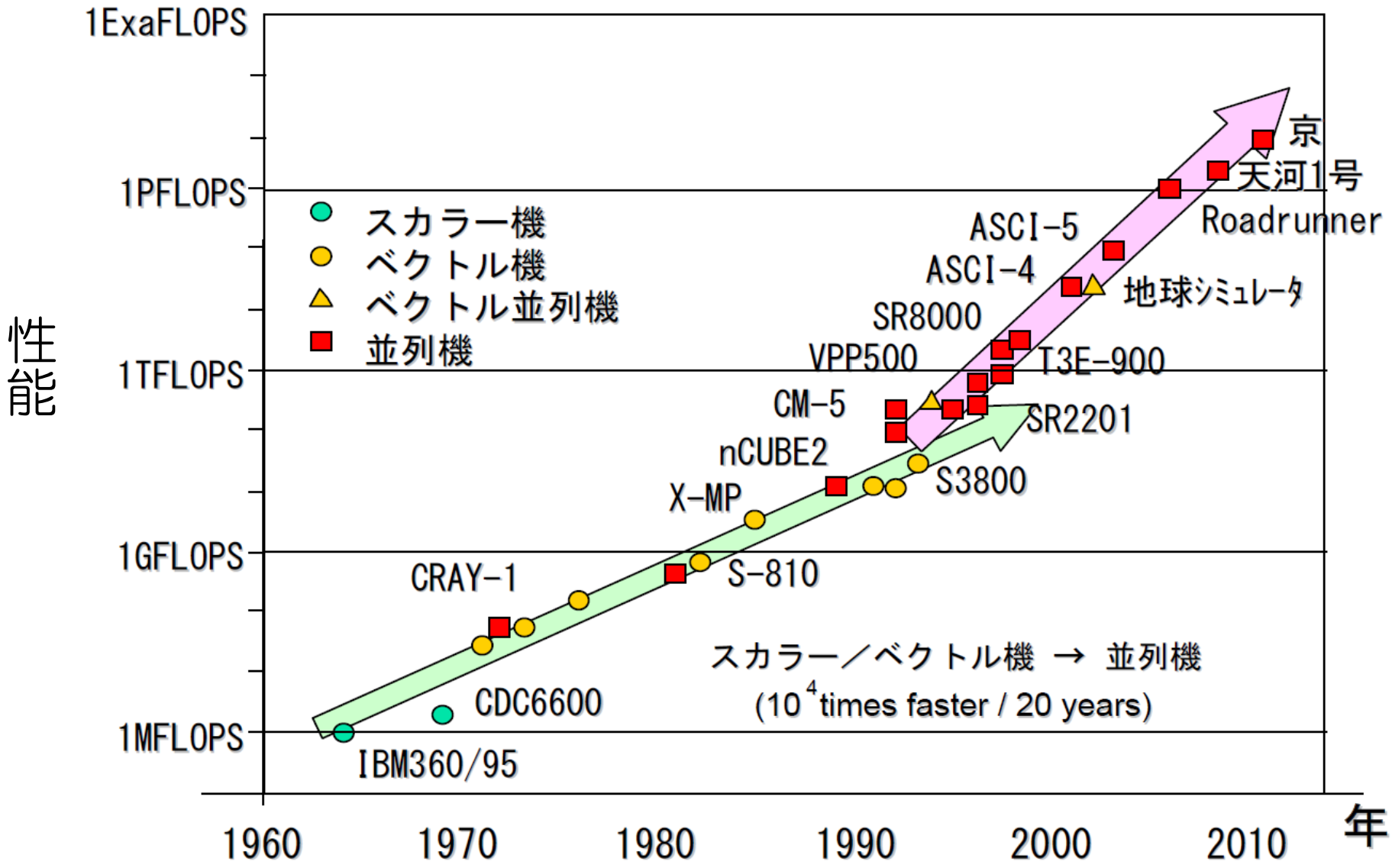
W.J. Kaufmann III and L.L. Smarr, *SUPERCOMPUTING AND THE TRANSFORMATION OF SCIENCE*, Scientific American Library (1993)

並列計算機 ILLIAC IV（イリノイ大学）

http://library.thinkquest.org/18268/History/hist_c_60s.htmより

- SIMDマシン
- 64プロセッサ
- プロセッサあたり
  - 10^4 ECLトランジスタ
  - 2K語（8Mビット）
- 16筐体
- 500MFLOPS
  （パソコンの20分の1）

スーパーコンピュータの性能動向



TOP500によるスーパーコンピュータ性能比較

- TOP500とは．．．
  - 世界のスーパーコンピュータ上位500システムをLINPACKベンチマークにより順位付け（http://top500.org）
    - LINPACKベンチマーク
      - LU分解法により，連立一次方程式の解を求めるベンチマークプログラム
  - 1993年から，LINPACK性能の記録あり
  - 年2回，国際会議でTOP500リストを発表
    - International Supercomputing Conference（ISCシリーズ）：ドイツ開催
    - The International Conference for High Performance Computing, Networking, Storage and Analysis（SCシリーズ）：米国開催

TOP500リストの上位10システム（2013年11月）

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115984	6271.0	7788.9	2325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301
9	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4293.3	5033.2	1972
10	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423

Tianhe-2

Titan

Sequoia

日本のシステム

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,659.9
11	GSIC Center, Tokyo Institute of Technology Japan	**TSUBAME 2.5** - Cluster Platform SL390s G7, Xeon X5670 6C 2.930GHz, Infiniband QDR, NVIDIA K20x NEC/HP	74,358	2,843.0	5,609.4	1,398.6
24	International Fusion Energy Research Centre (IFERC), EU(F4E) - Japan Broader Approach collaboration Japan	**Helios** - Bullx B510, Xeon E5-2680 8C 2.700GHz, Infiniband QDR Bull SA	70,560	1,237.0	1,524.1	2,200
30	Information Technology Center, The University of Tokyo Japan	**Oakleaf-FX** - PRIMEHPC FX10, SPARC64 IXfx 16C 1.848GHz, Tofu interconnect Fujitsu	76,800	1,043.0	1,135.4	1,176.8

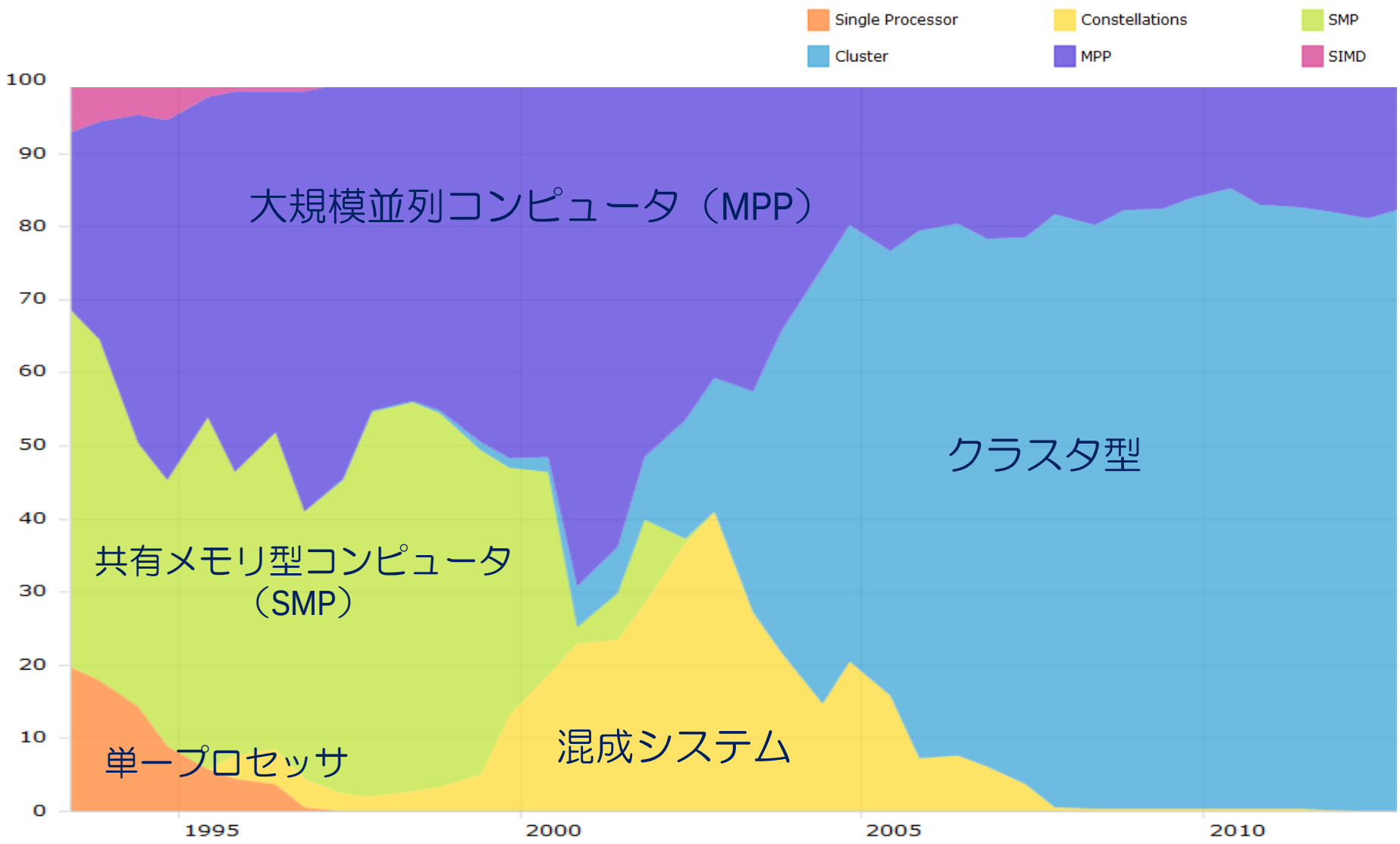
TOP2 ＆ TOP3 のシステム

- Titan（タイタン）
  - 米国オークリッジ国立研究所
  - 主たる目的は，自然科学の幅広い分野での活用
  - 主な仕様
    - ピーク性能：　27.11ペタフロップス
      - CPU: 2.63ペタフロップス
      - GPU:24.48
    - LINPACK性能： 17.59ペタフロップス
    - 計算ノード数：18,688（CPU+GPU）

- Sequoia（セコイア）
  - 米国ローレンス・リバモア国立研究所
  - 主たる目的は，核兵器の性能，安全性，信頼性解析及び予測
  - 主な仕様
    - ピーク性能：　20.13ペタフロップス
    - LINPACK性能： 16.32ペタフロップス
    - 計算ノード数：98,304

https://www.olcf.ornl.gov/2012/12/17/titan-trainers-take-road-trip/
http://www.top500.org/featured/systems/sequoia-lawrence-livermore-national-laboratory

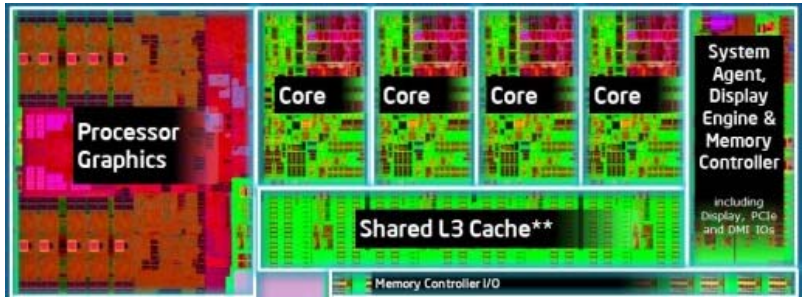
TOP500で見るアーキテクチャの変遷



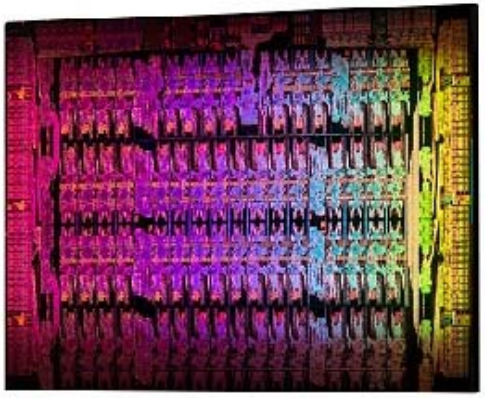
チップレベルでの並列処理

- マルチコアプロセッサ（Multicore processors）
  - チップ上に2～16個のプロセッサコアを搭載したマルチコアプロセッサが一般化
  - チップレベルでの並列計算機

- パソコンでも，並列計算が出来る時代になっている．

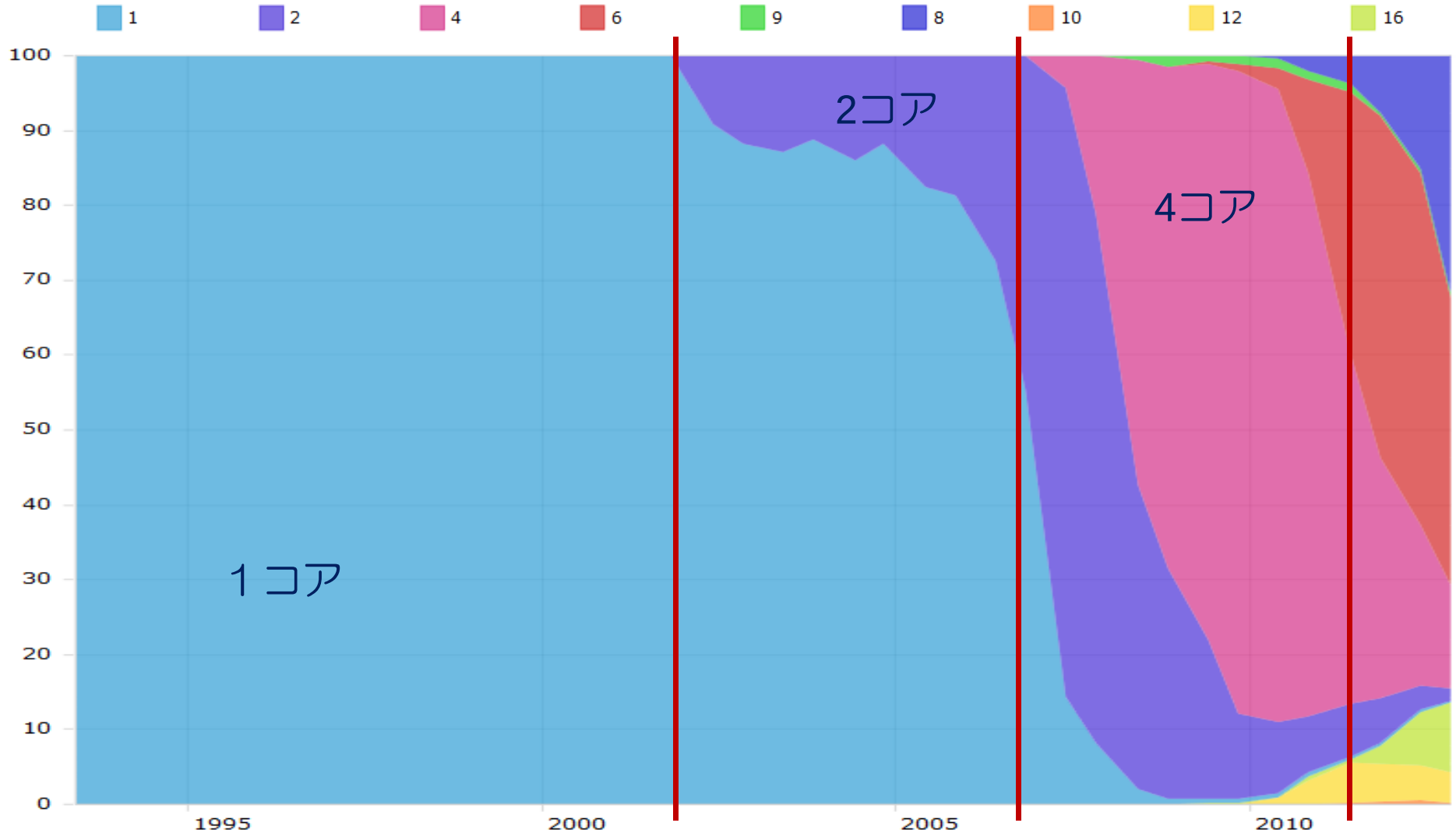


Intel Core i7-4771 (Haswell) プロセッサ
4コア, 22nm Tri-Gate Tr., 177mm^2



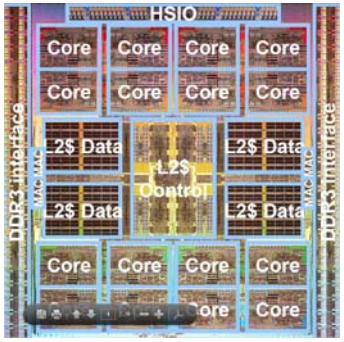
Intel Xeon Phi：MIC (Many Integrated Core) architecture
最大60コア, 22nm Tri-Gate Tr.

TOP500で見るコア数の変遷

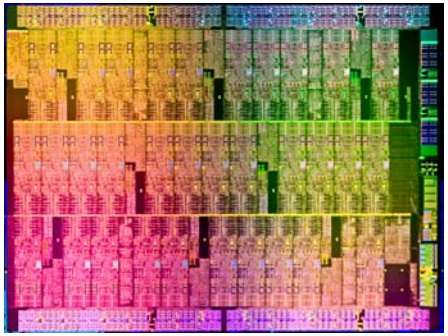


並列計算は必須

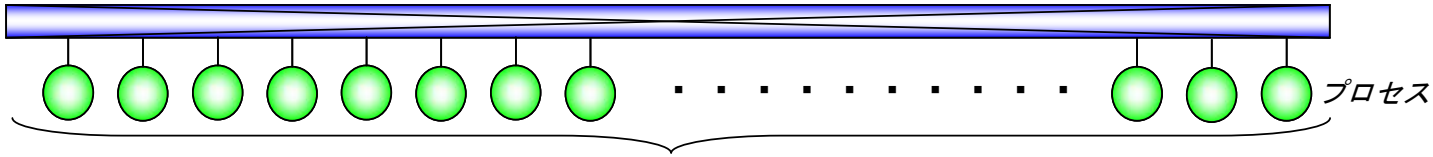
- マルチコア化
  - LSIの上に複数のCPU（コア）が乗っている
    - Intel Core i7 ： 4コア
    - SPARC64 IXfx：16コア
    - Intel Xeon Phi 5110P：60コア



- 並列計算の課題
  - 並列性をどのように使うか？
  - 並列性のあるアルゴリズムとは？
  - どの並列アルゴリズムがいいのか？
    - 評価指標（計算速度，並列化効率）

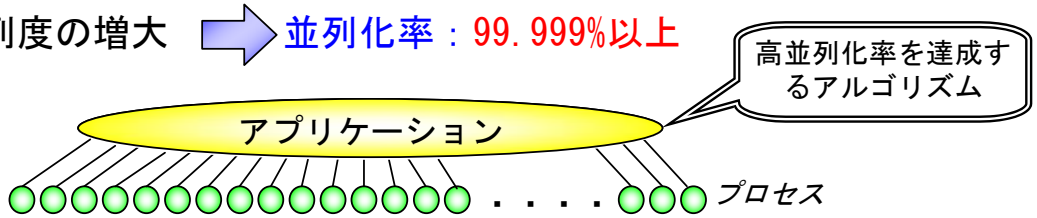


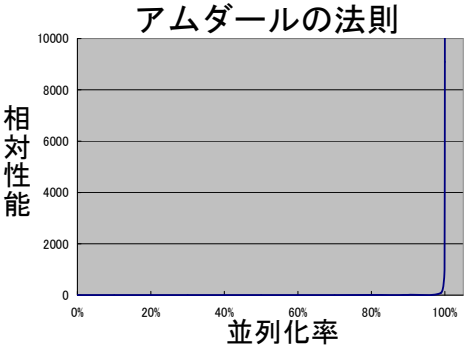
並列動作するプログラム開発の複雑化



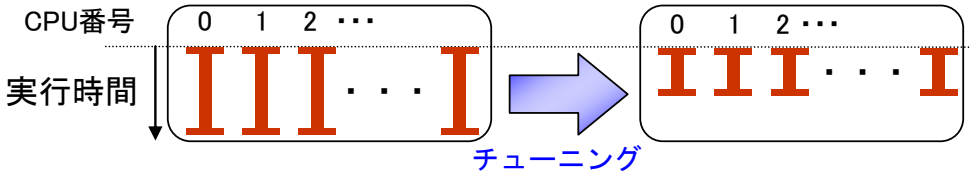
＞ 数万プロセス

①並列度の増大 → 並列化率：99.999%以上



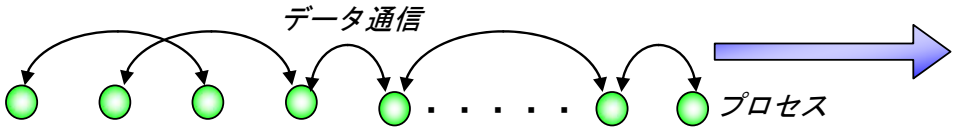


②ＣＰＵ（プロセス）チューニング



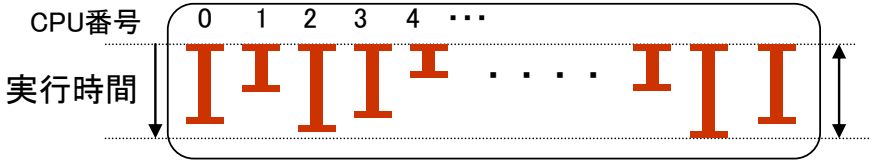
■チューニング手法
- キャッシュ/ソフトウェアパイプライン
- ベクトルチューニング
- ・・・

③通信量削減



データ量/通信回数削減

④ロードバランス



最大実行時間のプロセスで性能が律速

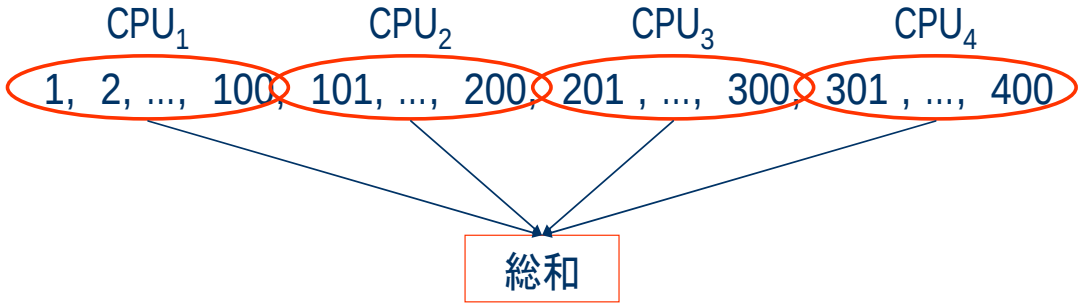
並列プログラムの複雑さ

1から400までの和を求めるプログラム

逐次プログラム

```
real*4  x(400)

do i = 1, 400
  x(i) = i
enddo

s = 0.0
do i = 1, 400
  s = s + x(i)
enddo

write(6,*)  “s= “, s

stop
end
```

4つの部分和（1～100，101～200，201～300，301～400）を4つのCPUで計算し，最後に総和を取る．



MPIによる並列プログラム

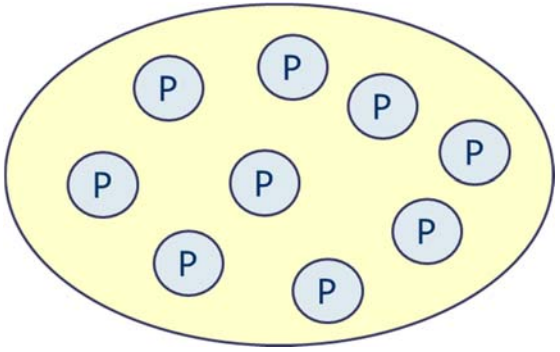
```
parameter (n=400, np=4, masterpid=0)

real*4  x(400)
integer  to_p, from_p, tag, mypid, pnum

call MPI_init (4)
call MPI_comm_rank( MPI_COMM_WORLD, mypid)
call MPI_comm_size( MPI_COMM_WORLD, pnum)

tag = 0
if ( mypid .eq. masterpid) then
  do i = 1, 400
    x(i)  = I
  enddo
  do to_p = 1, 3
    call MPI_send( x(100*to_p+1, 100, MPI_real, to_p, tag,
MPI_COMM_WORLD)
  enddo
else
  from_p = 0
  call MPI_recv( x(1), 100, MPI_real, from_p, tag, MPI_COMM_WORLD,
jdummy)
endif

s = 0.0
do i = 1, 100
  s = s + x(i)
enddo

tag = 1
if ( mypid .ne. masterpid) then
  to_p = 0
  call MPI_send( s, 1, MPI_real, to_p, tag, MPI_COMM_WORLD)
else
  do from_p = 1, 3
    call MPI_recv( ss, 1,  MPI_real, from_p, tag, MPI_COMM_WORLD, jdummy)
    s = s + ss
  enddo
endif

write(6,*)  “s= “, s

call MPI_finalize

stop
end
```

講義概要

- なぜ並列計算なのか？
- 並列計算機のアーキテクチャ
- 基本的な並列化手法
- 並列計算の性能向上

並列計算機のアーキテクチャ

- 並列計算機
  - 複数のプロセッサ（ノイマン・アーキテクチャ）が，何らかの方法で接続されいて，協調して動作する計算機システム
- タイプ
  - SIMDとMIMD（Flynnの分類, 1966年）
    - 命令実行の流れとデータの流れに着目した分類方法
  - 共有メモリ型と分散メモリ型
    - メモリ空間に着目した分類方法
  - ホモジニアス型とヘテロジニアス型
    - ハードウェアの均質性に着目した分類方法



プロセッサ（P），プロセッサエレメント（PE），プロセッサユニット（PU），ノード（Node），コア（core）

SIMDとMIMD

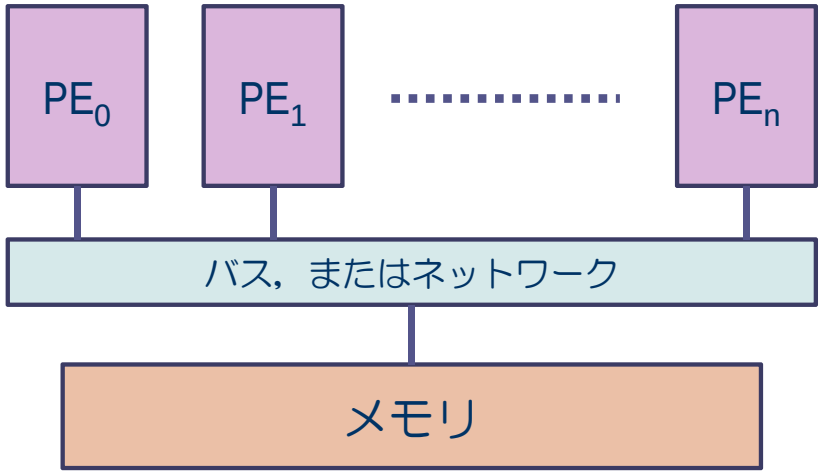
- SIMD（Single Instruction Stream, Multiple Data Stream）型並列計算機
  - 全プロセッサが，それぞれ異なるデータに対し，同じ命令を実行する．
    - 例：初期（～1990年頃）の商用計算機（CM-1など），グラフィックス・プロセッサ
  - 特徴
    - 命令実行の制御回路が1つで済むので，プロセッサ数の増加が容易．
    - 命令実行に柔軟性がないため，用途が限られる．

SIMDとMIMD（続き）

- MIMD（Multiple Instruction Stream, Multiple Data Stream）型並列計算機
  - 各プロセッサ（コア）が，それぞれ異なるデータに対し，異なる命令を実行する.
    - 例
      - 最近の商用並列計算機（π-computer）
      - マルチコアプロセッサ
  - 特徴
    - 各プロセッサに制御回路が必要
    - 命令実行の柔軟性は非常に高い.

共有メモリ型並列計算機

- 複数のプロセッサ（PE）が，単一のメモリ空間を共有
  - どのPEも同じメモリ領域にアクセス可能

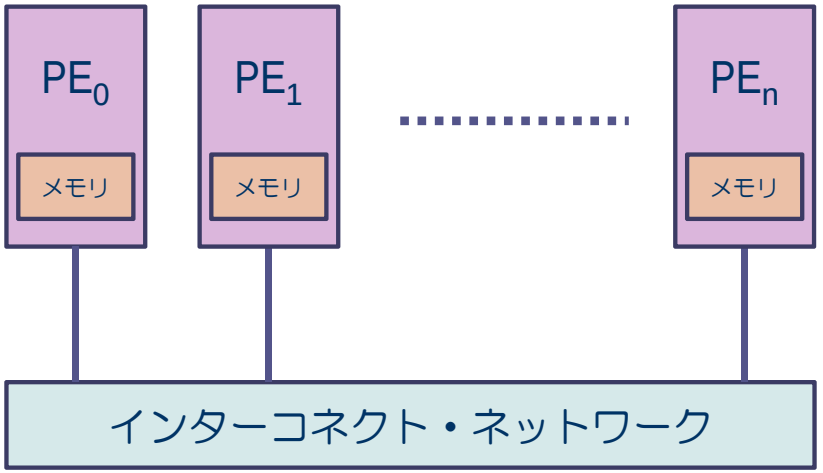


- 特徴
  - メモリ空間が単一なので，プログラミングが容易
  - PEの数が多いと，同一メモリアドレスへのアクセス競合が生じ，性能が低下

- プログラミング技術
  - OpenMP，自動並列化
  - ただし，MPIで実行することも可能

分散メモリ型並列計算機

- 複数のプロセッサがネットワークで接続されており，れぞれのプロセッサ（PE）が，メモリを持っている.
  - 各PEが自分のメモリ領域のみアクセス可能

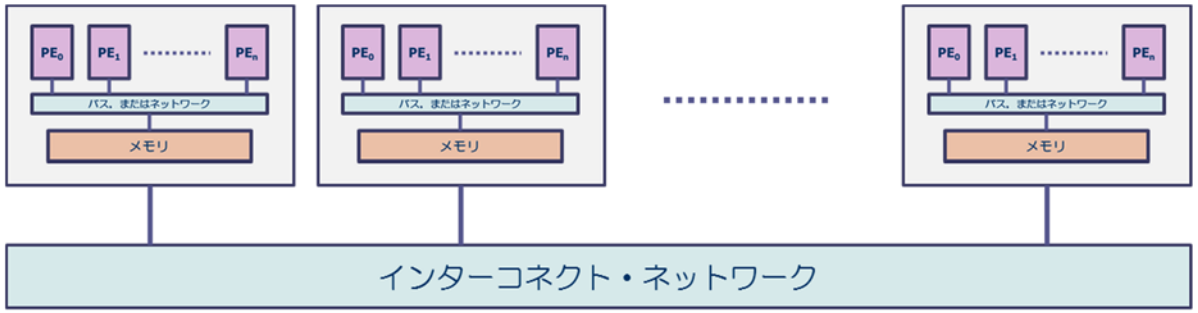


- 特徴
  - 数千から数万PE規模の並列システムが可能
  - PEの間のデータ分散を意識したプログラミングが必要.

- プログラミング技術
  - メッセージ・パシング・インターフェイス（MPI）によるプログラミング

分散共有メモリ型計算機

- 分散メモリ型並列計算機の各PEが，共有メモリ型並列計算機としてみなせるもの
  - 例： SMPクラスタ，マルチコアプロセッサを単一ノードにした分散メモリ型システム



  - 特徴
    - 各ノードの性能を高くできるため，比較的少ないノード数で高性能が達成可能

  - プログラミング手法
    - ノード内部ではスレッド並列，ノード間ではMPIによるプロセス並列のハイブリッド並列化の方法が用いられる.
    - MPIだけのプログラミングも可能（フラット並列）

ホモジニアス型とヘテロジニアス型

- ホモジニアス型の並列計算機
  - 全ノードが同じ構成を持つシステム
  - すべてのタスクが均質である並列化プログラムに向いている.
  - 例：SMPクラスタなど

- ヘテロジニアス型の並列計算機
  - 異なるアーキテクチャ，異なる性能のノードで構成されるシステム
  - 特定の計算部分を高速に実行するプロセッサを付加する場合が多い.
  - 例：グラフィックス専用プロセッサ（GPGPU），MD-GRAPEなど

計算科学演習Ｉで利用する並列計算機

学内LAN

演習用端末
Fujitsu ESPRIMO K552/D

六甲台キャンパス

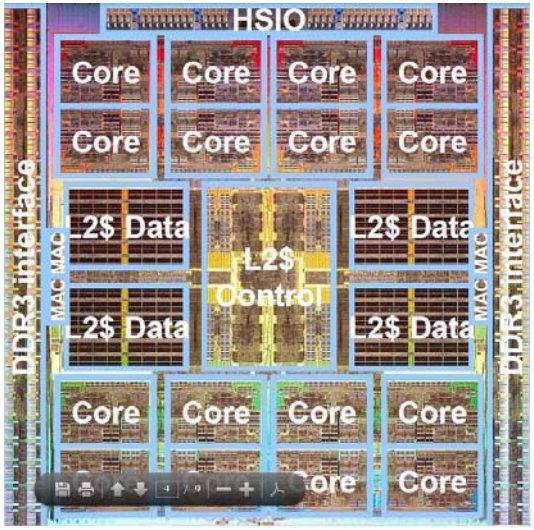
ログインサーバ
Fujitsu Primergy RX300 S6
- CPU：Intel Xeon E5645@2.4GHz, 6コア x 2 sockets
- メモリ 94GB

π-computer上のジョブ
実行はバッチジョブ

π-computer
Fujitsu PRIMEHPC FX10
96ノード，ノードあたり
- CPU：SPARC64 IXfx@1.65GHz, 16コア，211.2GFLOPS
- メモリ： 32GB/ノード

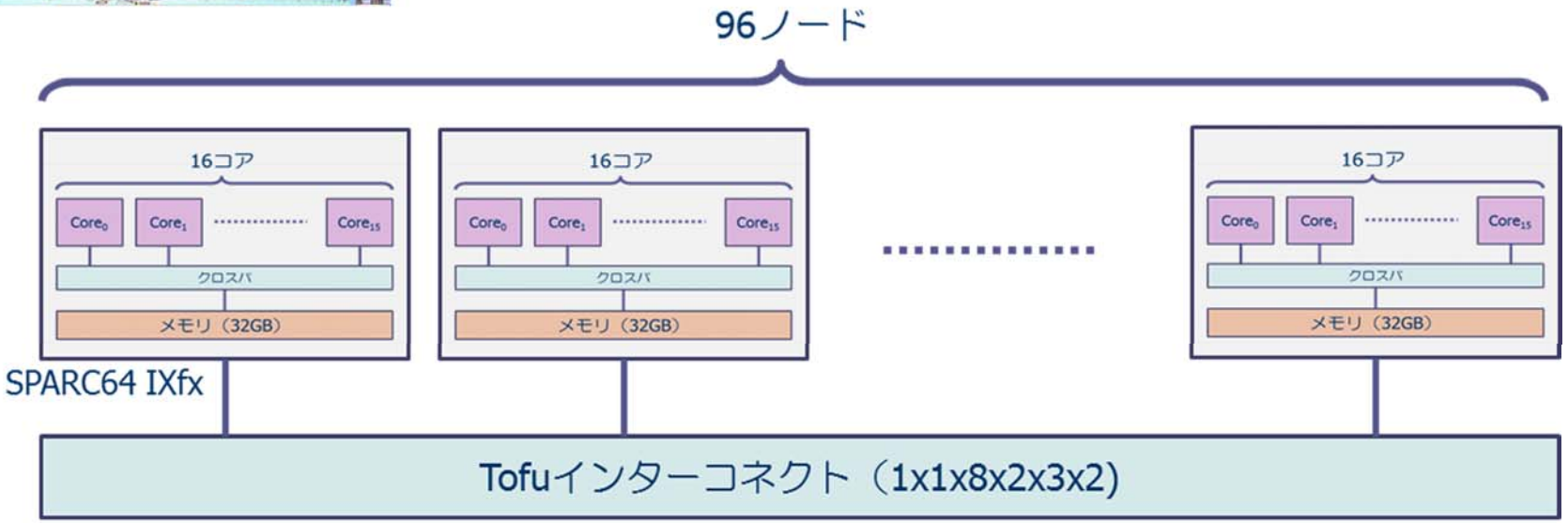
神戸大学統合研究拠点（ポートアイランド）

π-Computer（分散共有メモリ型）



【SPARC64 IXfx （SPARC64 V9 + HPC-ACE）】

- コア数：16コア
- 動作周波数：1.65GHz
- キャッシュ
  - L1-I，L1-D： 32KB/コア，L2：12MB/ソケット
- メモリバンド幅： 85GB/s
- 40nm CMOS，21.9mm×21.9mm



http://www.fujitsu.com/downloads/TC/sc11/sparc64-ixfx-sc11.pdf より

π-Computerと「京」の比較

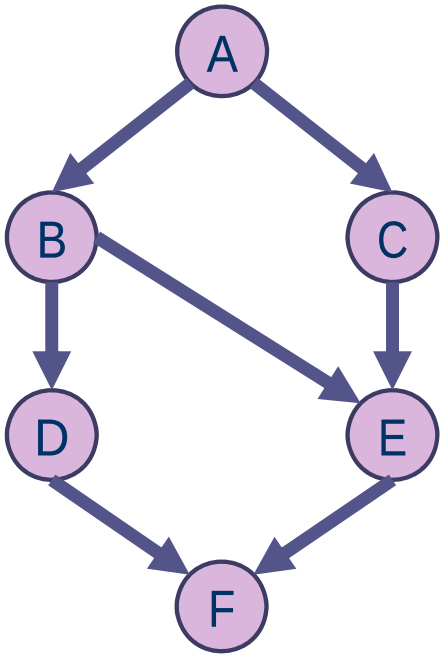
マシン		π-Computer	京
CPU		SPARC64 IXfx	SPARC64 VIIIfx
	コア数	16コア	8コア
	動作周波数	1.65GHz	2.0GHz
	L1キャッシュ（コア毎）	命令キャッシュ 32KiB データキャッシュ 32KiB	←
	L2キャッシュ	12MiB（24way）	6MiB（12way）
ノード	チップ数（LSI数）	1個	←
	性能	211.2GFLOPS	128GFLOPS
	メモリ容量	32GiB	16GiB
	メモリスループット	64GB/s（16コア）	46GB/s（8コア）
ラック	性能	20.3TFLOPS	12.3TFLOPS

講義概要

- なぜ並列計算なのか？
- 並列計算機のアーキテクチャ
- 基本的な並列化手法
- 並列計算の性能向上

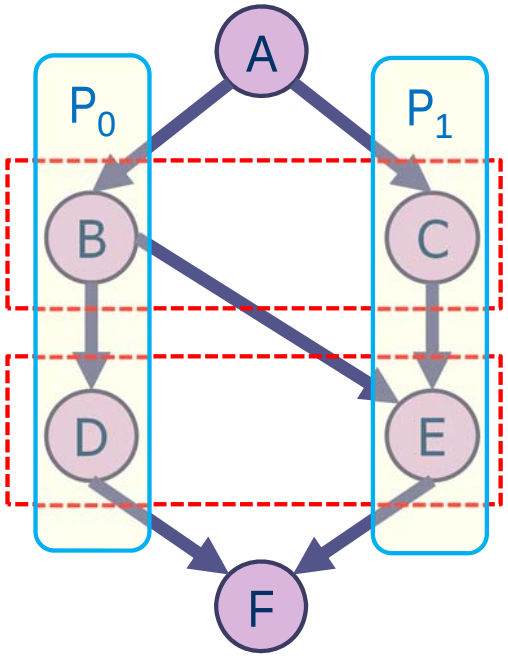
制御フローグラフ

- 処理間の依存関係
  - 2つの処理A, Bに対し，Aの処理を終えないとBの処理が開始できないとき，処理Bは処理Aに依存するという．
- 制御フローグラフ
  - 処理A, B, C, … を丸〇で表し，BがAに依存するとき，AからBへの矢印を描いて作ったグラフ
  - 依存関係にある処理は，順番に実行しなくてはいけないため（並列に実行できないため），処理間の並列性を考える上で有効



基本的な並列化方法：タスク並列化

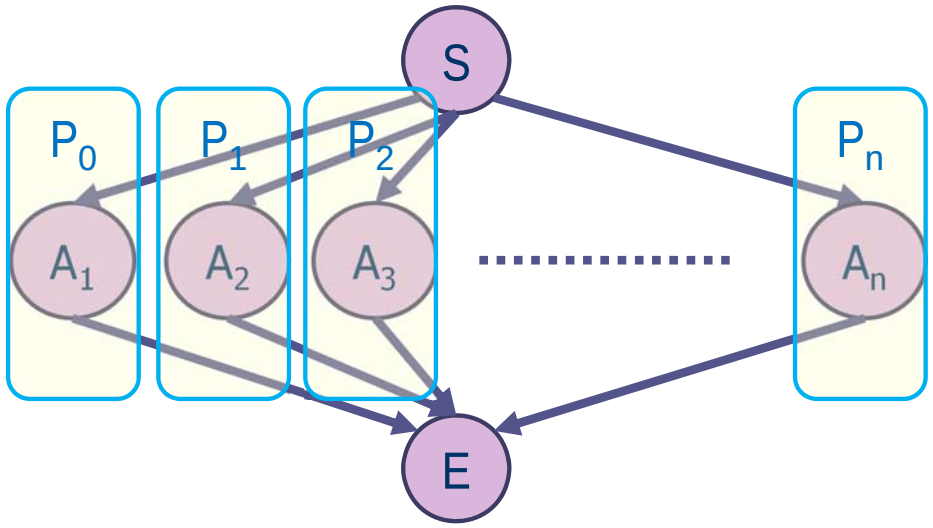
- 異なる処理 A, B, C, … があるとき，同時に実行可能な処理を見出し，別々のプロセッサに処理を割り当てて並列化する方法
  - ひとつのプロセッサがいくつかの処理を行なっても良い．



- BとC，DとEは同時実行可能
- BとC，DとEを別々のプロセッサで実行．
  - ただし，Eの実行には，Bが終了していることが必須．「処理待ち」が発生．

基本的な並列化方法：データ並列化

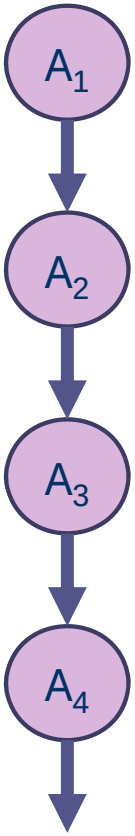
- 多数のデータに対し，独立に同じ処理が行える場合，各データに対する処理を，複数のプロセッサで同時に実行させる方法
  - ひとつのプロセッサが複数のデータの処理をしてもよい．



- n個のデータに対する処理A_1～A_nを，別々のプロセッサで実行する．
- 例：ベクトルの加算
  - 要素ごとに独立に計算可能

並列化できない処理

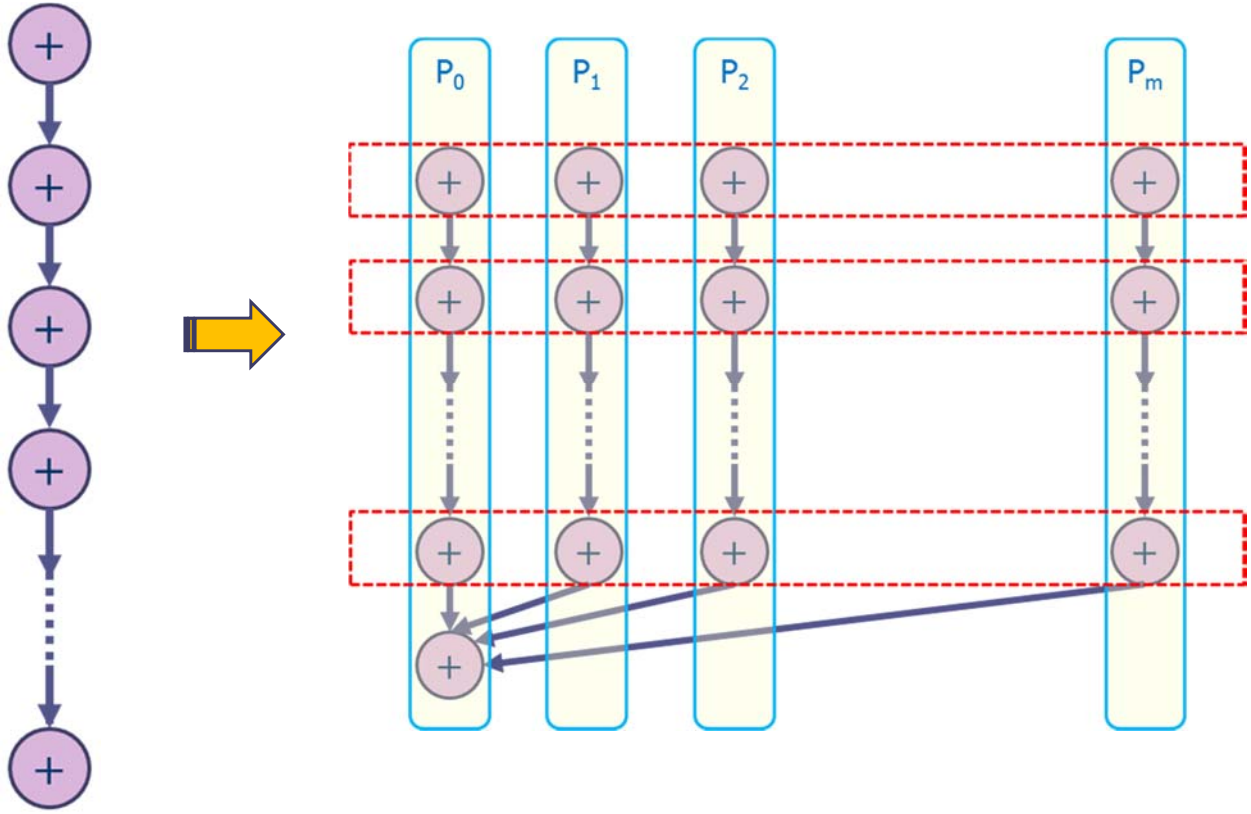
- 逐次的な処理は，同時に実行できない．
  - 処理A_nが終了しないと，処理A_{n+1}が開始できないような場合
  - 制御フローグラフは直線的になる．
- 例
  - 計算の開始や終了時の前処理，後処理
  - データの入出力
- アルゴリズムによっては，結果が変わらない範囲で，計算の順番を変更して並列化できるものがある．



並列計算の例：総和演算

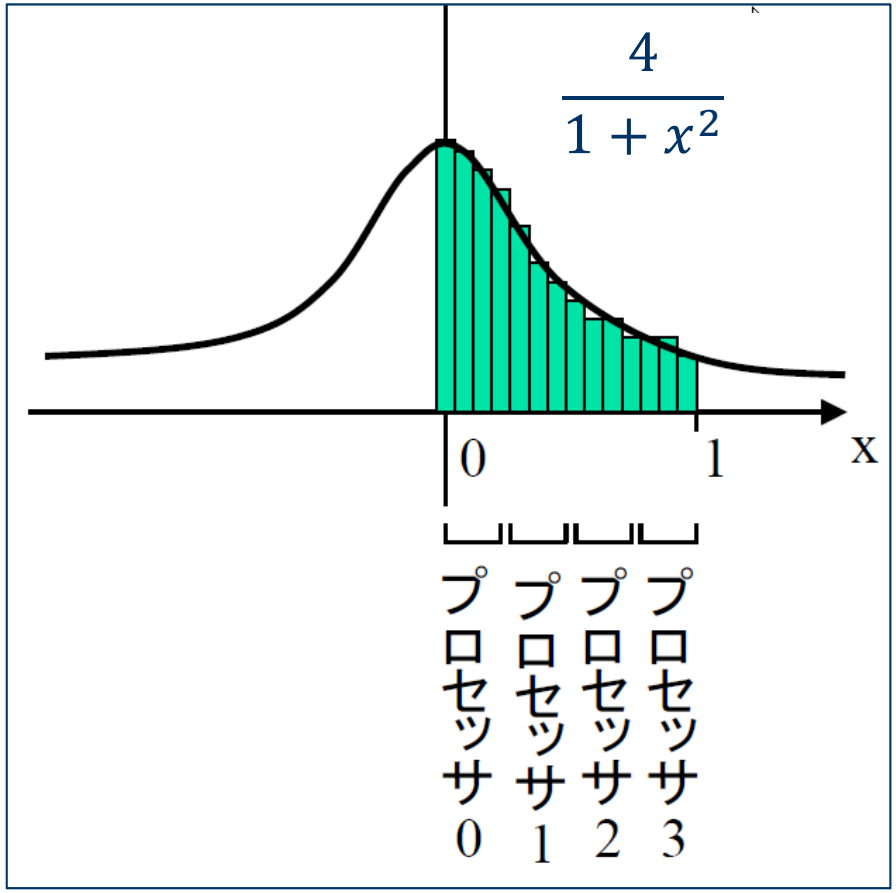
- サイズ n の配列 a[0], a[1], a[2], ..., a[n-1] の要素の和を求める.

- 並列化手法（プロセッサ数：m個）
  - 配列をp個の部分に分割する.
  - 各プロセッサで部分和を求める.
  - 最後に1つのプロセッサで集計する.



総和演算の例

- 数値積分によるπの計算
  - $\pi = \int_0^1 \frac{4}{1+x^2} dx$ をn等分の中点則で計算する



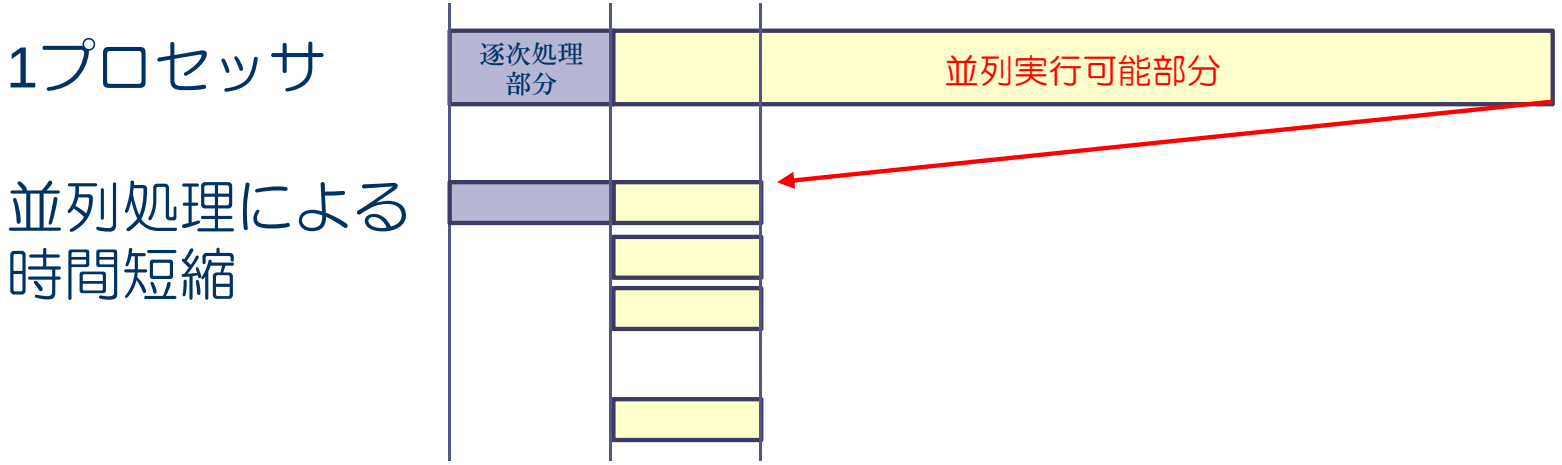
- 並列化
  - n個の長方形を，4個のプロセッサに割り当てる.
  - 各プロセッサは，それぞれの短冊の面積を求め，部分和を計算する.
  - 最後に，プロセッサ0が部分和を足し合わせる.

講義概要

- なぜ並列計算なのか？
- 並列計算機のアーキテクチャ
- 基本的な並列化手法
- 並列計算の性能向上

並列処理による実行時間短縮

- 並列計算の目的：複数のプロセッサを使うことにより，実行時間を短縮すること．
- プログラムには，並行して実行できる部分（並列化可能部分）と，どうしても順番に実行しなければならない逐次処理部分がある．
  - 並列化可能部分の割合が大きいほど，並列実行による時間短縮が期待される．
  - 逐次処理部分の割合が大きいと，並列化の効果は小さい．



並列性能の越えられない壁：アムダールの法則

- 並列化指標：加速率と並列化効率
  - 加速率： プロセッサ 1個，プロセッサ n_p個のときの実行時間をそれぞれ T_1 , T_{n_p}とするとき，$\mathrm{T}_1/\mathrm{T}_{n_p}$
  - 並列化効率： $\mathrm{T}_1/(n_p \times \mathrm{T}_{n_p})$

  ※ 実行時間は何で計るか？ → クロックルーチンがある.

- アムダールの法則
  - 並列化可能部分の割合を α とするとき，加速率の上限は $1/(1-\alpha)$
    - 逐次実行部分が10%もあると，高々10倍にしかならない.
  - 実際には，いろいろな要因により，加速率はさらに低い値になる.

⇒ 並列性能を上げるための努力が必要.

並列性能向上のために

- 並列計算機の性能を引き出すためのプログラムを書かなくてはいけない.
  - プロセッサ間の負荷分散
  - 適切な並列粒度の選択
  - 同期回数の削減
  - プロセッサ間通信の回数と通信量の削減
- 演習の中で触れていく.

出席確認

- piにログインし，以下を実行する.
  - アカウントは，自分のログインID.
  - $はプロンプト

```
$ date > foo （fooという名前のファイルがないことを確認してから）
$ whoami >> foo
$ cat foo | mail -s “Comp:アカウント” yokokawa@port.kobe-u.ac.jp
$ rm foo
```

- アンケートに答えること.