

# スーパーコンピュータとは

—高速化の基礎—  
(ベクトル化、並列化とは)

2005/8/31 初版  
2008/9/1 第3版

大阪大学レーザーエネルギー学研究中心  
高性能計算機室  
ile-comp@ile.osaka-u.ac.jp

# Contents

---

はじめに	3
1. スーパーコンピュータとは	4
2. 高性能コンピュータの方式	5
3. どうプログラムすればいいか	7
4. ベクトルとは	9
5. 並列化とは	11
6. HPFとは	13
さいごに・参考文献	15

# はじめに

---

このテキストは、プログラミングの経験もあまりない、学部や修士の学生の方を対象に、スーパーコンピュータとは何か、ベクトル化や並列化とはなどのイメージをつかんでいただき、大阪大学サイバーメディアセンターで年2回から3回開催されるベクトル化と並列化の講習会の内容をより理解しやすくなることを目標に作成しています。FORTRAN77を基本としていますが、FORTRAN90やCでも基本は同じです。

Fortranがよくわからないという方は、CMCで開催される「AVS&FORTRANプログラミング入門」の講習会がお勧めです。

2005年8月の講習会のテキストとして使用しましたが、後半にかけて難しいという声をアンケートでいただきましたので、CMCのシステム更新にあわせて修正しました。皆さんの声を参考に、より分かりやすいテキストに改版したいと考えていますので、ぜひアンケートにご協力ください。後日のご連絡も歓迎いたします。

2008年9月1日  
大阪大学レーザーエネルギー学研究センター  
高性能計算機室 福田優子

# 1. スーパーコンピュータとは

「スーパーコンピュータとは」、明確な定義が確立しているわけではなく、時代とともに変化します。あまりに多様化したので、スーパーコンピュータとは呼ばずに、高性能コンピュータ(HPC、High Performance Computer)と呼ばれることが増えています。

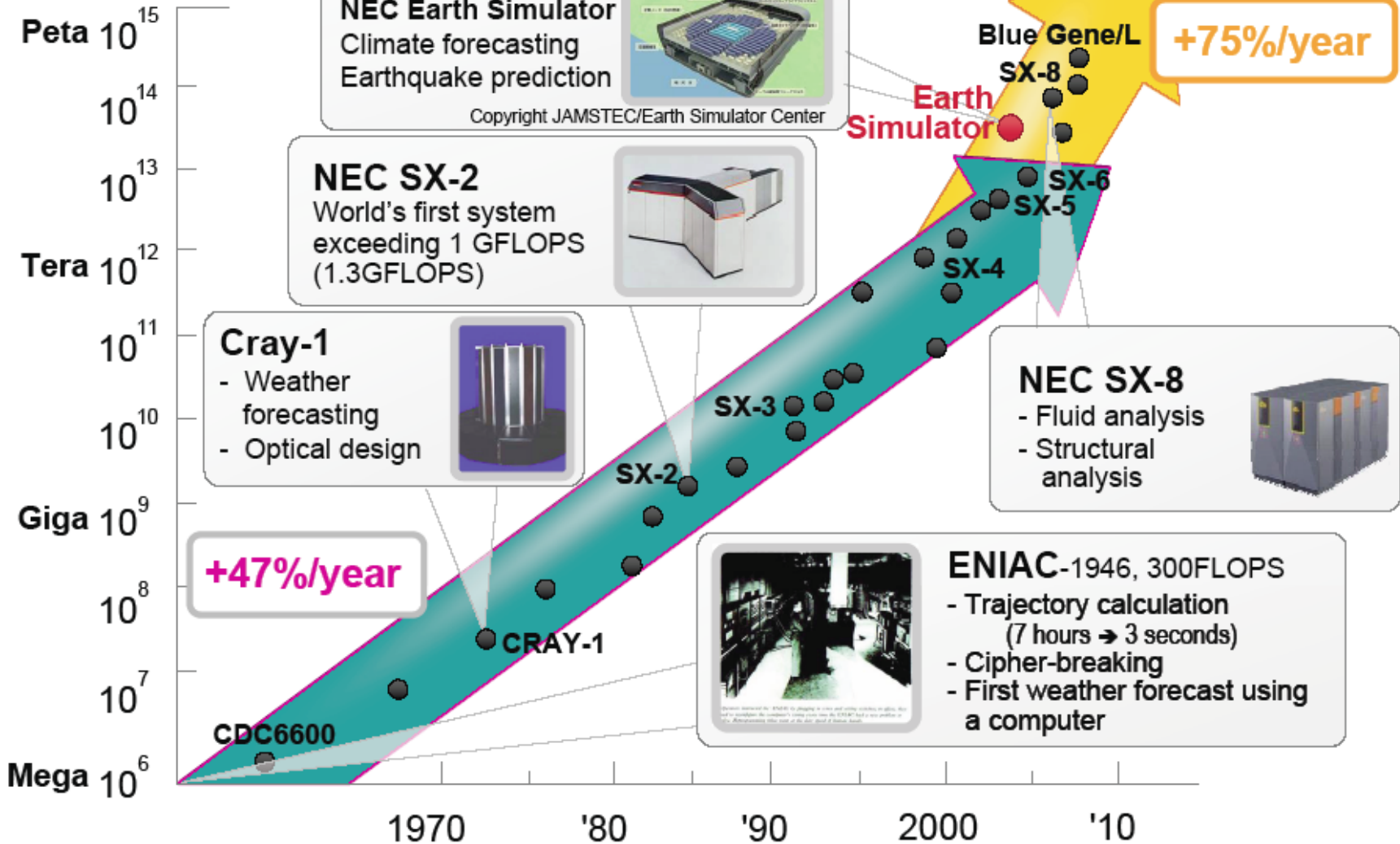
- ・その時代で最も高速な処理能力をもつコンピュータ群の総称
- ・主に科学・技術分野に利用される。

性能としては、GFLOPS, TFLOPSなどの単位が用いられますが、10年以上昔からは、たくさん並べて理論的な性能や、特殊なプログラムでのピーク性能を宣伝する傾向がありますが、SXは簡単なプログラミングで、高い実効性能を得ることができます。

- \* フロップス FLPOS (Floating-Point Operations Per Second)  
浮動小数点演算を1秒間に10億回実行できる

# スーパーコンピュータの性能トレンド

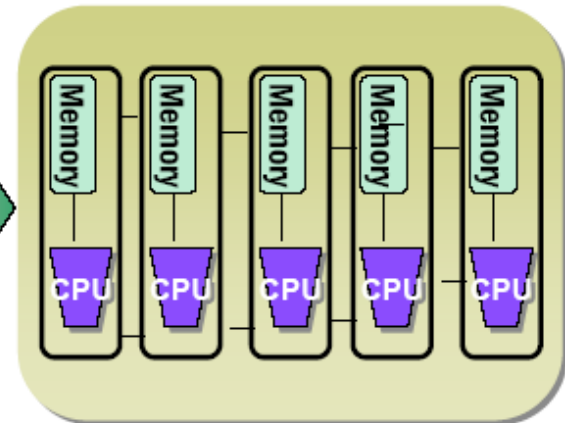
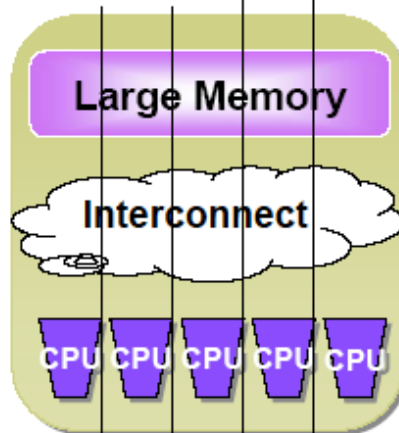
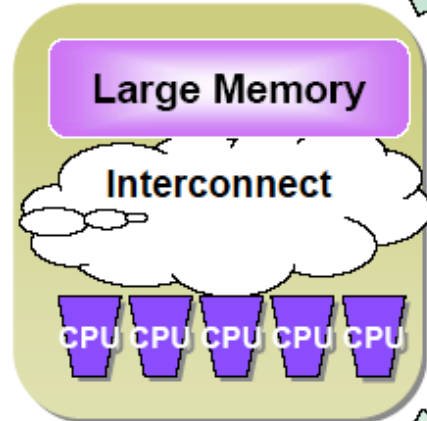
FLOPS (Peak)



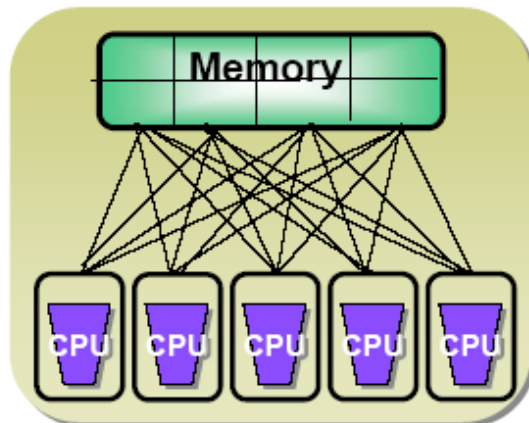
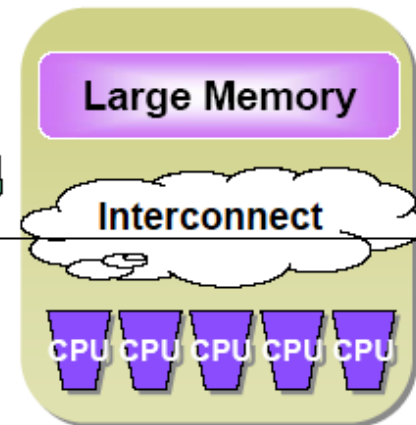
# スーパーコンピュータアーキテクチャ・コンセプト

## 理想的なスーパーコンピュータの実現

理想的な  
スーパーコンピュータ



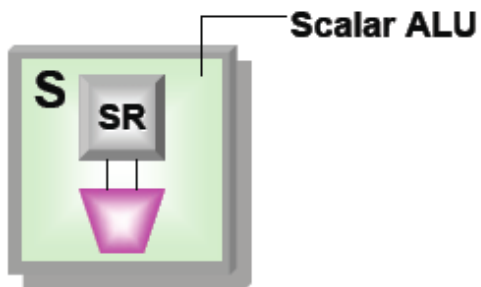
分散メモリ



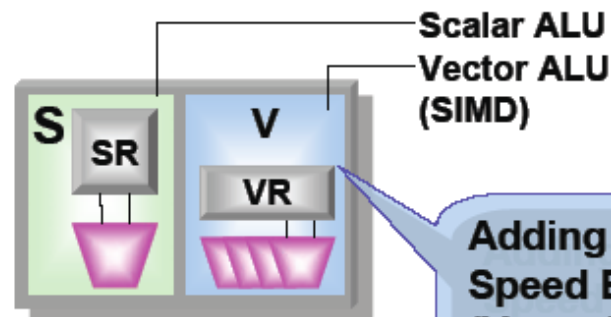
共有メモリ

# Processor Architecture

**SX has been heterogeneous chip configuration**

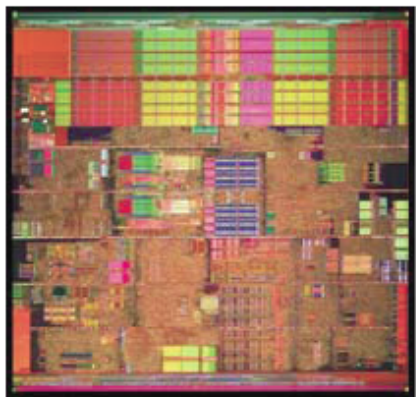


**Scalar Processor**

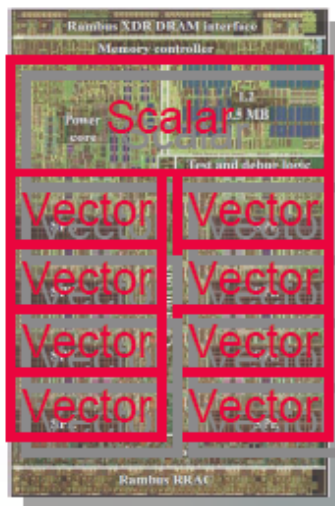


**Vector Processor**

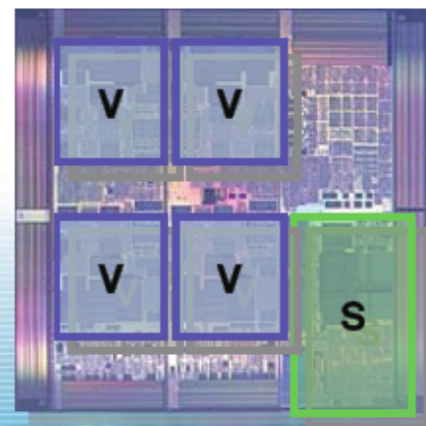
Adding High Speed Engine (Vector)



**Pentium 4**



**Cell**

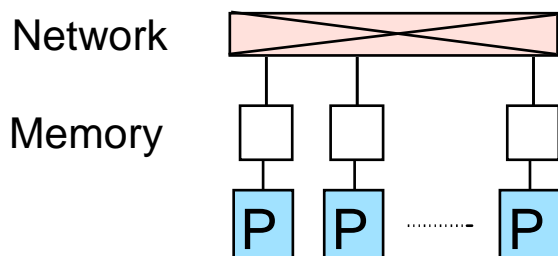


**SX-8**

Empowered by Innovation

**NEC**

## 2. 高性能コンピュータの方式



### 並列分散メモリ機（スカラ）

例 PCクラスタ

- ・完全メモリ分散方式
- ・汎用マイクロプロセッサ使用

特長: 優れたコスト対ピーク性能比

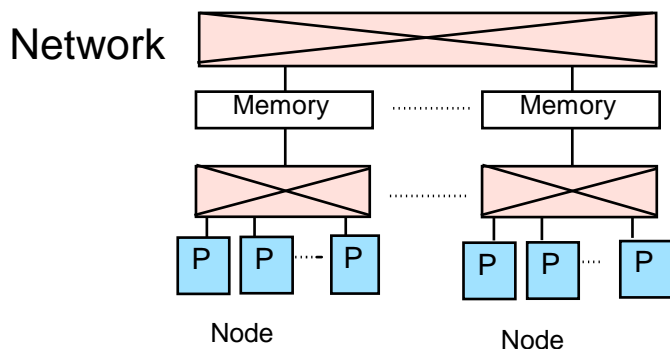
将来の可能性

短所: ピーク性能に比べ実効性能が低い

適合する応用領域が狭い

ハードウェア・ソフトウェアの標準方式がない

完全メモリ分散方式→プログラムが作りにくい



### 共有・分散メモリ機（スカラ）

例 SGI:Altix,富士通:PRIMEPOWER,日立:SR11000

NEC:AsAmA(TX7)

- ・共有・分散メモリ方式
- ・汎用マイクロプロセッサ使用

特長: 優れたコスト対ピーク性能比

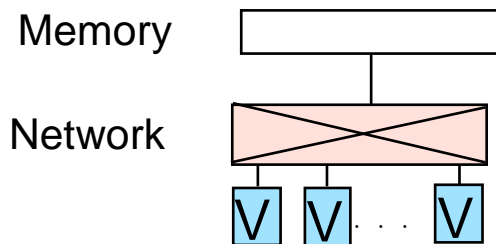
プログラムが作り易い

短所: メモリアクセスが不均一→最適化が困難

超大規模構成→キャッシュコヒーレンス問題あり

## 2. 高性能コンピュータの方式

### 共有メモリ機（ベクトル）



例 NEC: SX-5,6,7,8

・メモリ共有方式（N台Xベクトルプロセッサ）

・専用ベクトルプロセッサ使用

特長：メモリ共有方式ベクトルプロセッサの実績と優れた実効性能

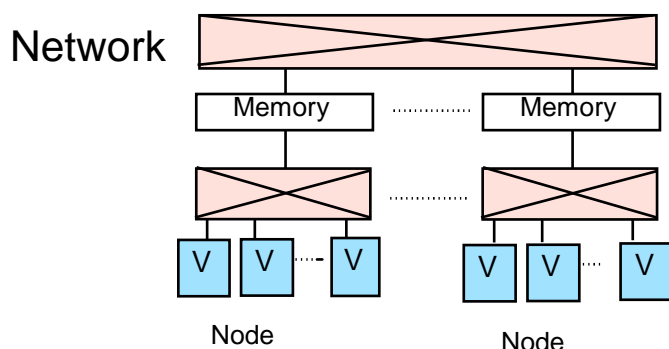
適合する応用領域が広い

標準の開発・運用ツールが完備

メモリ共有方式→プログラムが作り易い

短所：超大規模システムへの拡張性がない

### 共有・分散メモリ機（ベクトル）



例 NEC: SXマルチノードモデル

・メモリ共有 x 分散方式（N台Xベクトルプロセッサ）x Mノード

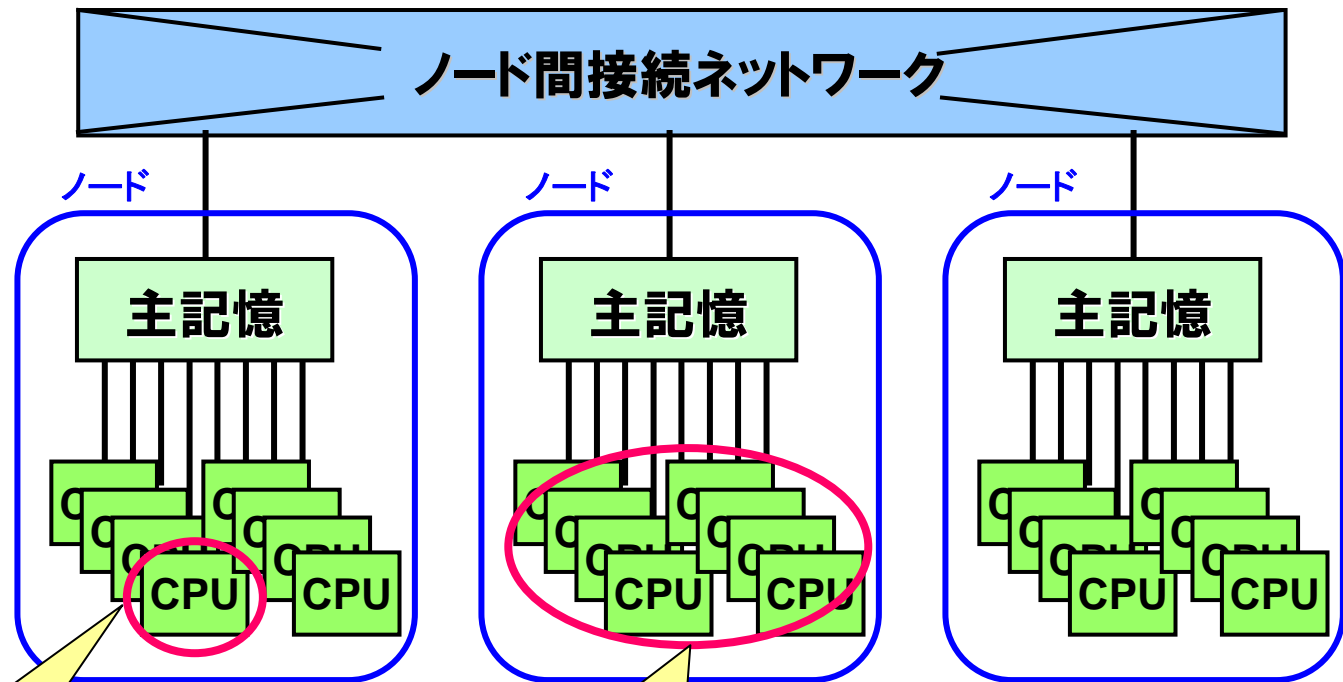
・専用ベクトルプロセッサ使用

特長：ベクトルプロセッサの実績と優れた実効性能  
超大規模システムへの拡張性がある

共有メモリの範囲での使いやすさ

短所：ノード間=メモリ分散→プログラムが作りにくい

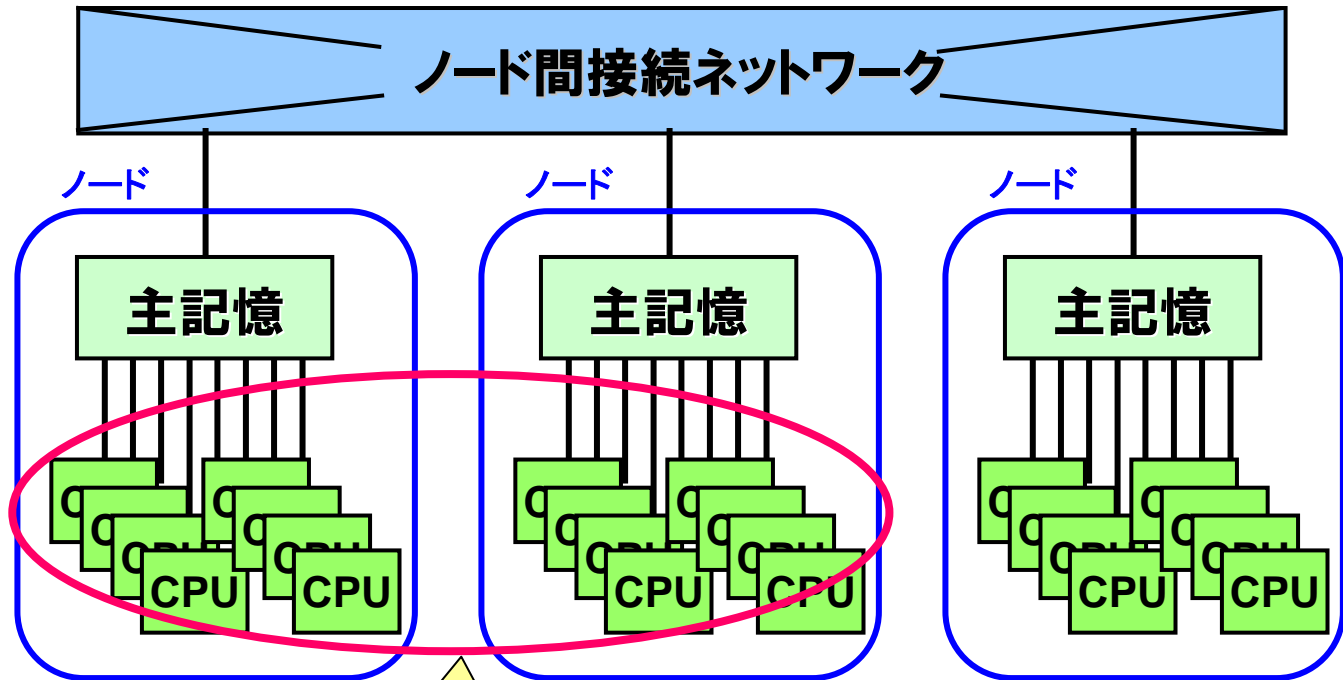
### 3. どうプログラムすればいいか(ノード内共有メモリ)



単一CPUで計算  
(シングルと呼ぶ)  
ベクトル化

ノード内(共有メモリ)の複数CPUで計算  
(ノード内パラレルと呼ぶ)  
(自動並列、openMP、MPI、HPF)

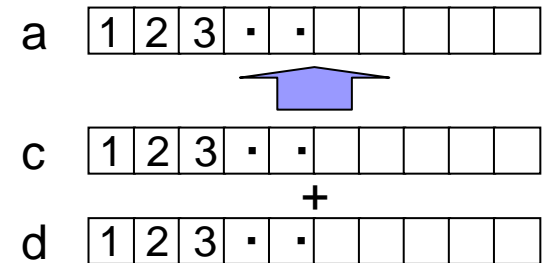
### 3. どうプログラムすればいいか(ノード間分散メモリ)



異なるノードの複数CPUを用いて計算  
(ノードまたがりと呼ぶ)  
(MPI、HPFによるプログラミングが必要)

# 4. ベクトルとは

```
Do i=1,imax  
  a(i) = c(i) + d(i)  
  b(i) = c(i) × d(i)  
enddo
```



スカラー実行  
(逐次処理)

```
a(1) = c(1) + d(1)  
b(1) = c(1) × d(1)  
a(2) = c(2) + d(2)  
b(2) = c(2) × d(2)  
:  
:  
a(imax) = c(imax) + d(imax)  
b(imax) = c(imax) × d(imax)
```

ベクトル実行

```
a(1) = c(1) + d(1)  
a(2) = c(2) + d(2)  
:  
:  
a(imax) = c(imax) + d(imax)  
b(1) = c(1) × d(1)  
b(2) = c(2) × d(2)  
:  
:  
b(imax) = c(imax) × d(imax)
```

## 4. ベクトル化とは

### ベクトルデータ:

行列の行要素、列要素、あるいは対角要素など、規則的に並んだデータ列

**ベクトル命令**:ベクトルデータに対する演算を実行する命令

### スカラーデータ:単一データ

SXではFORTRANSXなどのコンパイラがソースプログラムを解析して、ベクトル命令で実行可能な部分を自動的に検出し、その部分に対してベクトル命令を生成します → 自動ベクトル化(SXでは既定値)

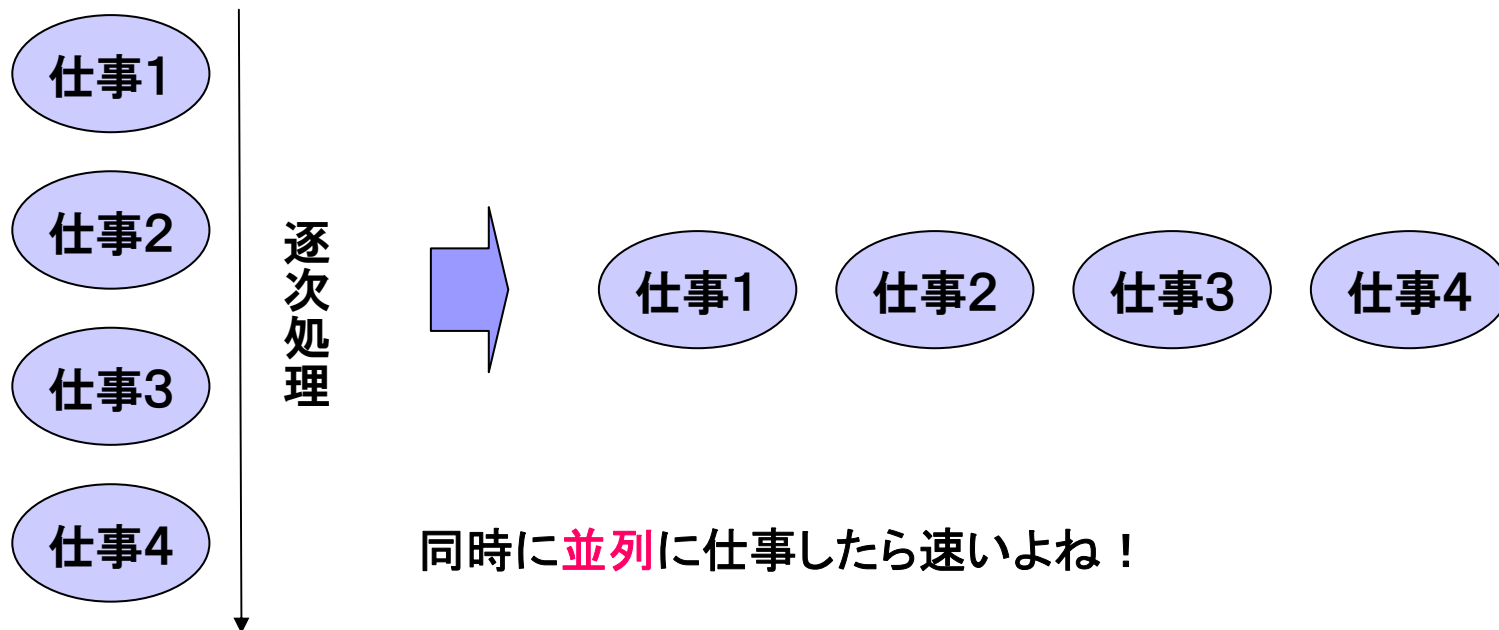
スカラー実行とベクトル実行では、計算の順序が異なる

スカラーとベクトルで計算した場合で結果が異なるかもしれないと判断するとコンパイラは基本的にベクトル化しません。

SXはベクトル化してこそ本領発揮！

→ **ベクトル化できるようにプログラムするのが基本  
(プログラミングも容易)**

## 5. 並列化とは



計算時間そのものは速くならず、仕事を始めてから終わるまでの経過時間を速くしようというもの

共有メモリか分散メモリかによって、プログラミングが異なる。  
SXは共有メモリでも、分散メモリ用の並列プログラムが実行でき、効率がよいのでMPIやHPFをノード内でも用いてもよい。

## 5. 並列化とは

### 自動並列

共有メモリの範囲で、コンパイラが自動的に並列化して高速化を試みる。SXの自動並列は、ベクトル化とともに行われる。並列は利用者が指定しないと行わない。

### MPI (Message Passing Interface)

プログラミング言語ではなく、メッセージ通信のためのライブラリ。FORTRANやC言語などの逐次言語のライブラリとして使用する。最初から並列プログラムとして記述する必要がある。現在、一般的ではある。

### HPF (High Performance Fortran)

FORTRAN言語に最小限の指示文を追加することにより、分散メモリ並列で簡単に高い性能がでることを目指している。指示文はコメントとみなされ、通常のFORTRANでも実行できるので、デバッグやプログラムのメンテナンスが容易。

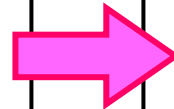
### その他いろいろありますが

これというのではないように思います。

## 5. HPF(High Performance Fortran)とは

最小限の指示文によるデータ分割配置指定で、  
並列計算(計算の分割と通信の生成)を自動的に行うFortranの拡張言語

```
INTEGER A(10),B(10),C(10)
DO I=1,10
  A(I) = C(I)+B(I)
ENDDO
WRITE(*,*) A
END
```



```
INTEGER A(10),B(10),C(10)
!HPF$ DISTRIBUTE (BLOCK) :: A,B,C
DO I=1,10
  A(I) = C(I)+B(I)
ENDDO
WRITE(*,*) A
END
```

### レーザー研での取り組み

- 7月26日27日にレーザー研においてHPF推進協議会の協力でHPF講習会を行いました。
- 殆どの参加者が直ぐにコードを書き換えてHPFで動かすことができました。  
(ベクトル、自動並列の経験のある自分でプログラムを作る研究者、MPIプログラミングはしたくない)

# 5. 並列化とは(MPIとHPFの比 )

プログラムの比較

MPI

```
parameter(n=100)
real a(n), b(n)
call MPI_INIT ( ierr )
call MPI_COMM_SIZE ( MPI_COMM_WORLD, np, ierr )
call MPI_COMM_RANK ( MPI_COMM_WORLD, id, ierr )
if( id .eq. 0 ) then
  read(*,*) a, b
  do i = 1, np-1
    call MPI_SEND ( a, ...
    call MPI_SEND ( b, ...
  end do
else
  call MPI_RECV ( a, ...
  call MPI_RECV ( b, ...
end if
is = ( n / np ) * id + 1
ie = ( n / np ) * ( id + 1 )
aipdt = 0.0
do i = is, ie
  aipdt = aipdt + a(i) * b(i)
end do
call MPI_REDUCE ( aipdt, aipd, ...
if( id .eq. 0 ) write(*,*) 'aipd = ', aipd
call MPI_FINALIZE ( ierr )
stop
end
```

- parameter(n=100)
- real a(n), b(n)
- read(\*,\*) a, b
- aipd = 0.0
- do i = 1, n
- aipd = aipd + a(i) \* b(i)
- end do
- write(\*,\*) 'aipd = ', aipd
- stop
- end

HPF

```
parameter(n=100)
real a(n), b(n)
!HPF$ PROCESSORS proc(number_of_processors())
!HPF$ DISTRIBUTE (BLOCK) ONTO proc :: a,b
read(*,*) a, b
aipd = 0.0
!HPF$ INDEPENDENT, REDUCTION(+:aipd)
do i = 1, n
  aipd = aipd + a(i) * b(i)
end do
write(*,*) 'aipd = ', aipd
stop
end
```

## さいごに

少しはスパコン、ベクトル、並列というものの感触がつかめたでしょうか？  
テキスト作成にあたっては、以下の方々にご協力いただきました。

HPF推進協議会

大阪大学レーザーエネルギー学研究中心 沼波 政倫特任研究員

核融合科学研究所 坂上仁志教授

日本電気(株)

大阪大学サイバーメディアセンター

## 参考文献、マニュアル、URL

1) 大阪大学レーザーエネルギー学研究中心

→ 研究部門紹介

→ 技術部

→ 高性能計算機室

→ グループのホームページへ

→ 公開テキスト

<http://www.ile.osaka-u.ac.jp/research/cmp>

2) 大阪大学サイバーメディアセンター(大規模計算機システムポータル)

[https://portal.hpc.cmc.osaka-u.ac.jp/ouportal\\_hpc/ouportal.jsp](https://portal.hpc.cmc.osaka-u.ac.jp/ouportal_hpc/ouportal.jsp)

3) HPF推進協議会

<http://www.hpfp.org>

# 本日の講習について

「ベクトル化、並列化の基礎」

「パフォーマンスチューニング」

ノード内のベクトル化と自動並列の話です。

MPIやHPFの勉強をしたい方も、まず基本をこの講習会で勉強してください。