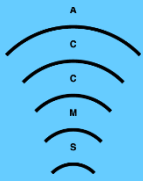


計算科学が拓く世界 スーパーコンピュータは 何故スーパーか

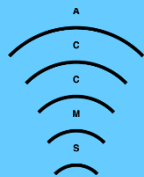
学術情報メディアセンター
中島 浩

- <http://www.cs.kyoto-u.ac.jp/>
提供科目＞計算科学が拓く世界＞前期#1



科目の概要 (1/2)

- **計算科学: 理論 & 実験科学に続く第3の科学**
 - **実施困難・不可能な実験を(スーパー)コンピュータの中で仮想的に実施(=シミュレーション)**
 - **観測困難・不可能な空間** (星の内部, 原子・分子レベル...)
 - **到達困難・不可能な時間** (過去の再現, 未来の予測...)
 - **構築困難・不可能な実験規模** (宇宙, 地球, 日本全土...)
- **科目の内容**
 - **最新の計算科学の研究事例を**
 - **さまざまな分野の第一線研究者が**
 - **なるべく数式を使わずに紹介・解説**



科目の概要 (2/2)

1	中島浩	ACCMS	スーパーコンピュータは何故スーパーか
2	中嶋洋	農・地環	オフロード車両の走行力学について
3	山本量一	工・化工	シミュレーション科学の挑戦～メソスケールの物質科学～
4	木元小百合	工・社基	地盤の変形シミュレーション
5	榎本剛	防災研	スーパーコンピュータを用いた気象の予測
6	青木一洋	医・生命動態	細胞内情報伝達系のシステムズバイオロジーと計算科学
7	竹広真一	数解研	計算機で天体内部の流れを考える
8	安藤耕司	理・化学	生命エネルギーを支える電子とプロトンの移動反応
9	黒瀬良一	工・機械	数値流体力学の役割
10	阿久津達也	化研	バイオインフォマティクスにおける計算科学
11	佐藤啓文	工・分子	化学現象・化学反応の理論～pHから蛋白質まで～
12	村上章	農・地環	(TBA)
13	小山田耕二	教育院	可視化の世界
14	木村欣司	情・数理	世界最長の多項式を計算する～判別式世界記録への挑戦～



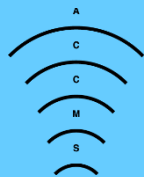
講義の概要

■ 目的

- 計算科学に不可欠の道具**スーパーコンピュータ**が
 - どうスーパーなのか
 - どういうものか
 - なぜスーパーなのか
 - どう使うとスーパーなのか
- について**雰囲気**をつかむ

■ 内容

- スーパーコンピュータの歴史を概観しつつ
- スーパーである基本原理を知り
- どういう計算が得意であるかを学んで
- それについて**レポート**を書く



どのくらいスーパー？ (1/2)



<http://www.aics.riken.jp/jp/k/system.html>



は の**180万倍**も高速

- **速さの単位 = FLOPS (フロップス)**
 = FLoating-point Operations Per Second
 = 浮動小数点演算毎秒
 = 1秒間に実行可能な浮動小数点数の加減乗算回数
- **浮動小数点数**
 - $10^{-308} \sim 10^{308}$ の実数を近似的に(10進16桁精度)表現したもの
 - $2.99792458... \times 10^8$ (m/s), $9.1093829140... \times 10^{-31}$ (kg)
-  11.5 P(ペタ 10^{15}) FLOPS (1.15 )
- ÷  6.4 G(ギガ 10^9) FLOPS (64億)
- = **1,797,120**

どのぐらいスーパー？ (2/2)

-  =  × 180万と  =  × 3
は話が違う

- 同じ土俵で比べるなら

- N700系 : $300\text{km/h} \times 1323\text{人} = 396,900\text{人} \cdot \text{km/h}$
 $\div \text{B767-300} : 880\text{km/h} \times 270\text{人} = 237,600\text{人} \cdot \text{km/h}$
 $= 1.67$ (倍も新幹線は飛行機より高速)

- 180万倍を細かく見ると

-  : $2.0\text{GHz} \times 8 \times 8 \times 88,128$

ここがスーパー

$\div$  : $1.6\text{GHz} \times 2 \times 2 \times 1$

$= 1,797,120$

intel Atom 330

intel Core i7なら
 $3.3\text{GHz} \times 8 \times 6$ も



スーパーコンピュータ (スパコン) とは (1/2)

■ パソコンの数千倍～数万倍の規模・性能を持つ 巨大な超高速コンピュータ

- 世界最大・最高速マシン $\doteq$ パソコン $\times$ 550万

- 京大スーパーコンピュータ $\doteq$ パソコン $\times$ 9万

→ パソコンで1ヶ月かかる計算 = 0.5秒～30秒

(ただしスパコン向きの問題をうまくプログラムしたら)

■ スパコンが高速な理由

- 個々の部品(CPU, メモリなど) $\doteq$ パソコン

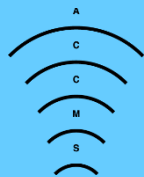
- 非常に多数のパソコン(のようなもの)の集合体

- パソコン = 1～16 CPU

- 京大スパコン = 40,208 CPU

- 世界最高速スパコン = 3,120,000 CPU

- 世界最大規模スパコン = 3,120,000 CPU



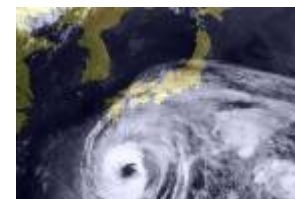
スーパーコンピュータ (スパコン) とは (2/2)

■ スパコンが得意な計算＝大量CPUによる分担計算 ＝超大量のデータを対象とする計算

■ 地球全体の気象・気候・海洋現象の予測

→ 1km^2 あたり1データ

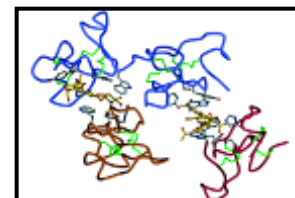
→ データ数 ≒ **5億** (× 高さ方向)



■ 生体物質・化学物質・材料の解析

→ **膨大**な分子・原子数

(e.g. 水 $1\text{ml} = 3.3 \times 1\text{兆} \times 100\text{億}$)



■ 自動車の空力・衝突解析

→ 1mm^2 or 1cm^3 あたり1データ

→ データ数 = **1~10億**

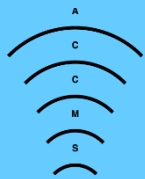


■ Web 文書の解析

(← 自動翻訳用データ作成など)

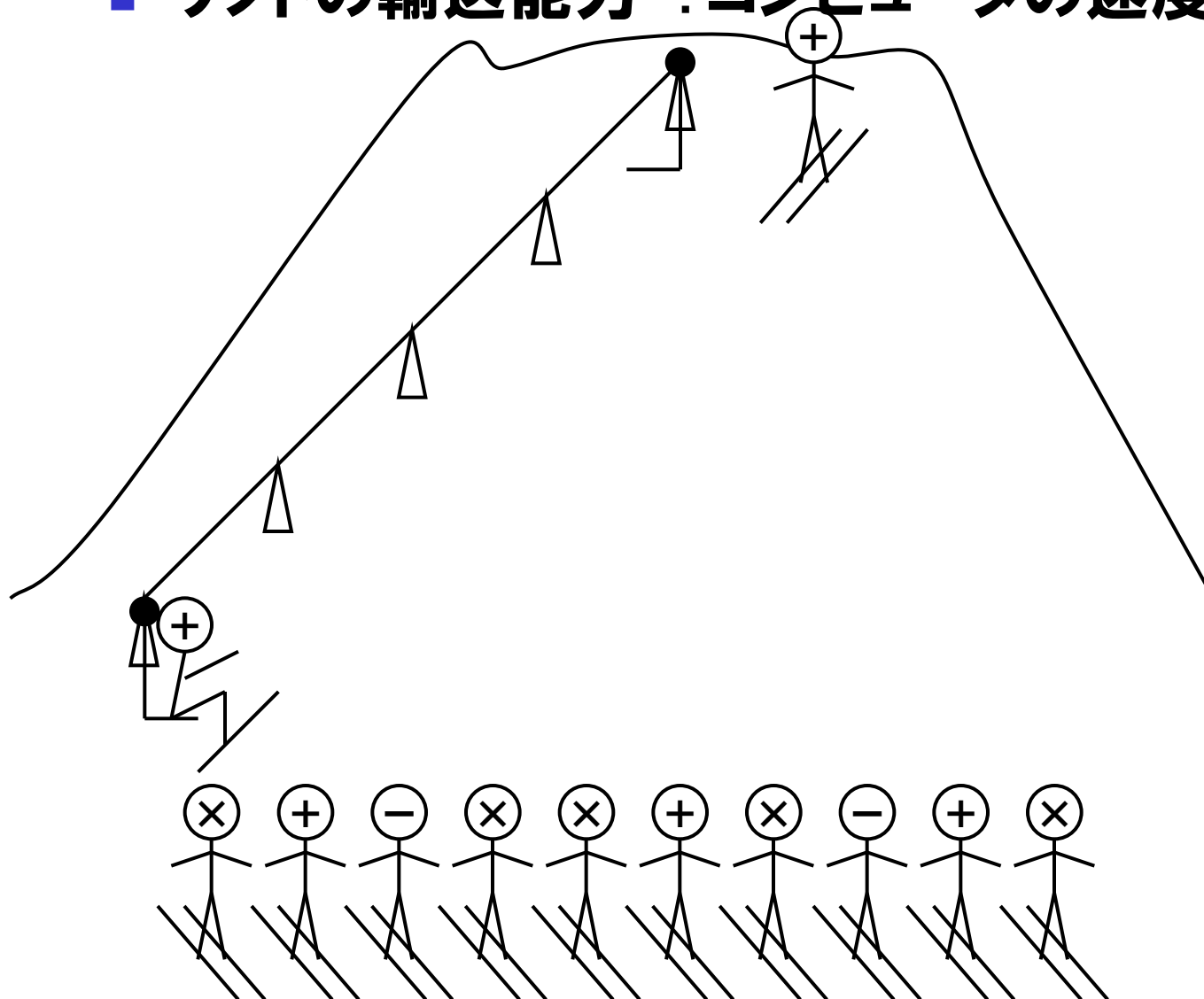
→ 文書数 = **数億～数10億**

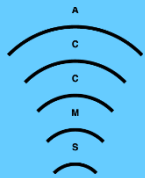




スーパーにする方法

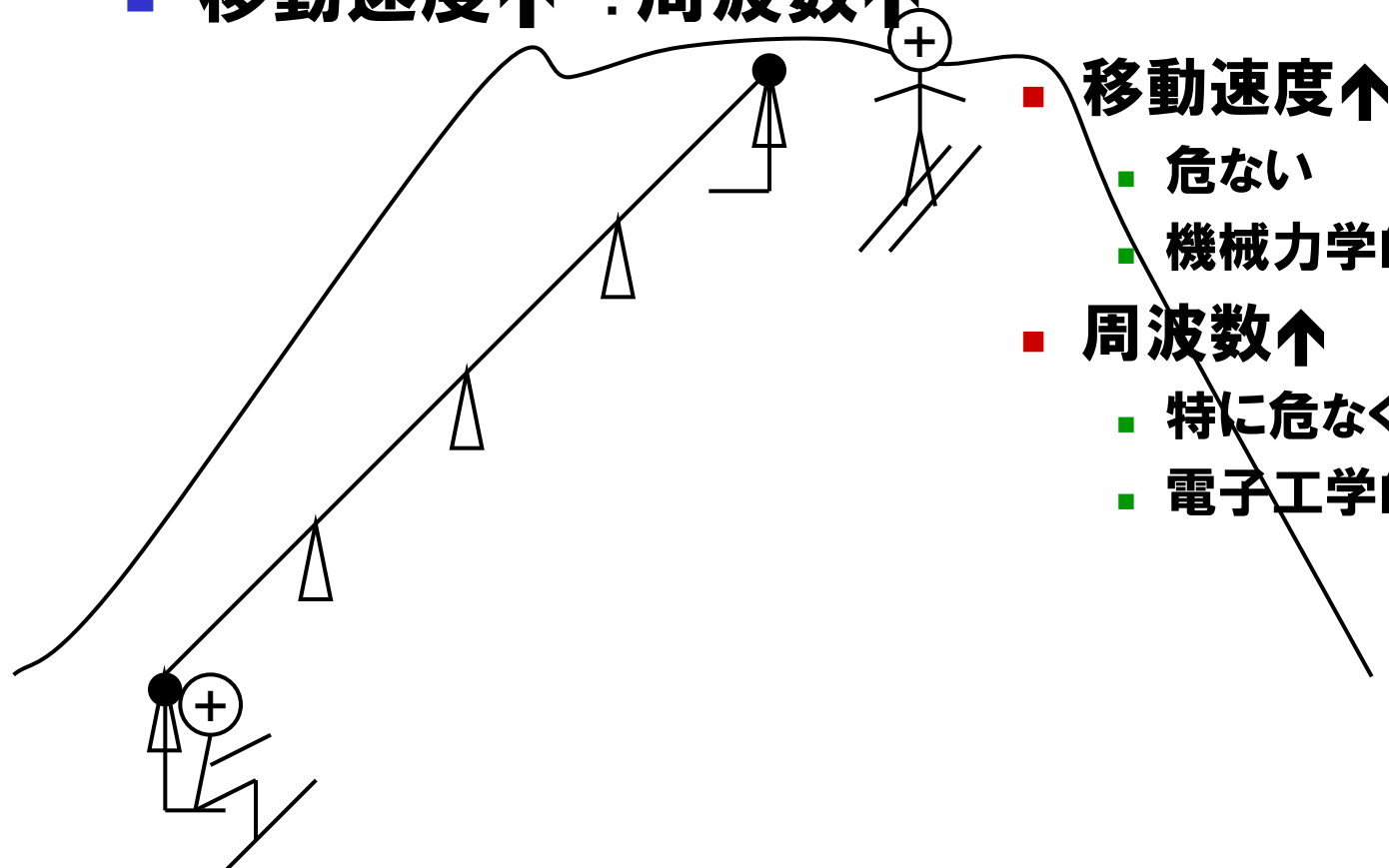
■ リフトの輸送能力 $\doteq$ コンピュータの速度





スーパーにする方法: ~1970

■ 移動速度 $\uparrow$ $\equiv$ 周波数 $\uparrow$

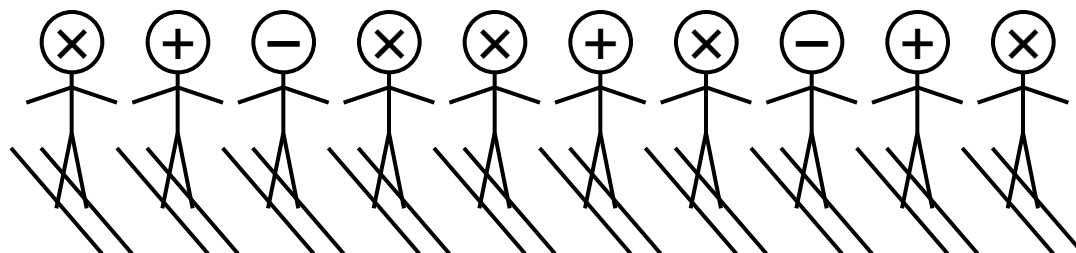


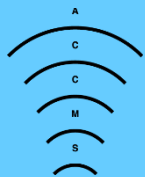
■ 移動速度 $\uparrow$

- 危ない
- 機械力学的に無理

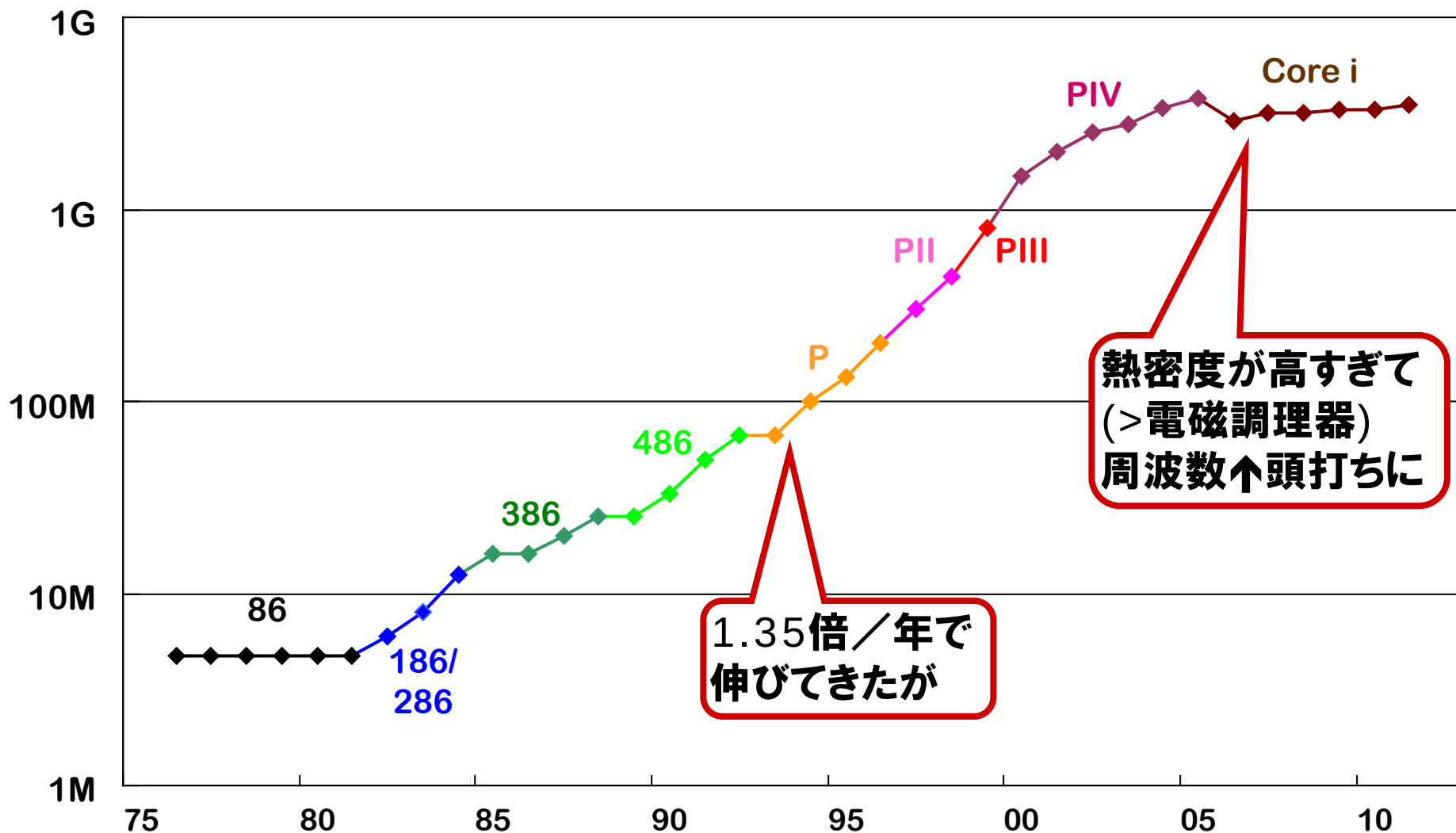
■ 周波数 $\uparrow$

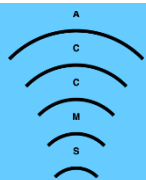
- 特に危なくはない
- 電子工学的に無理ではない?





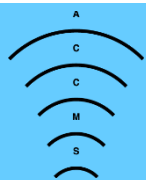
スーパーにする方法：周波数↑の歴史



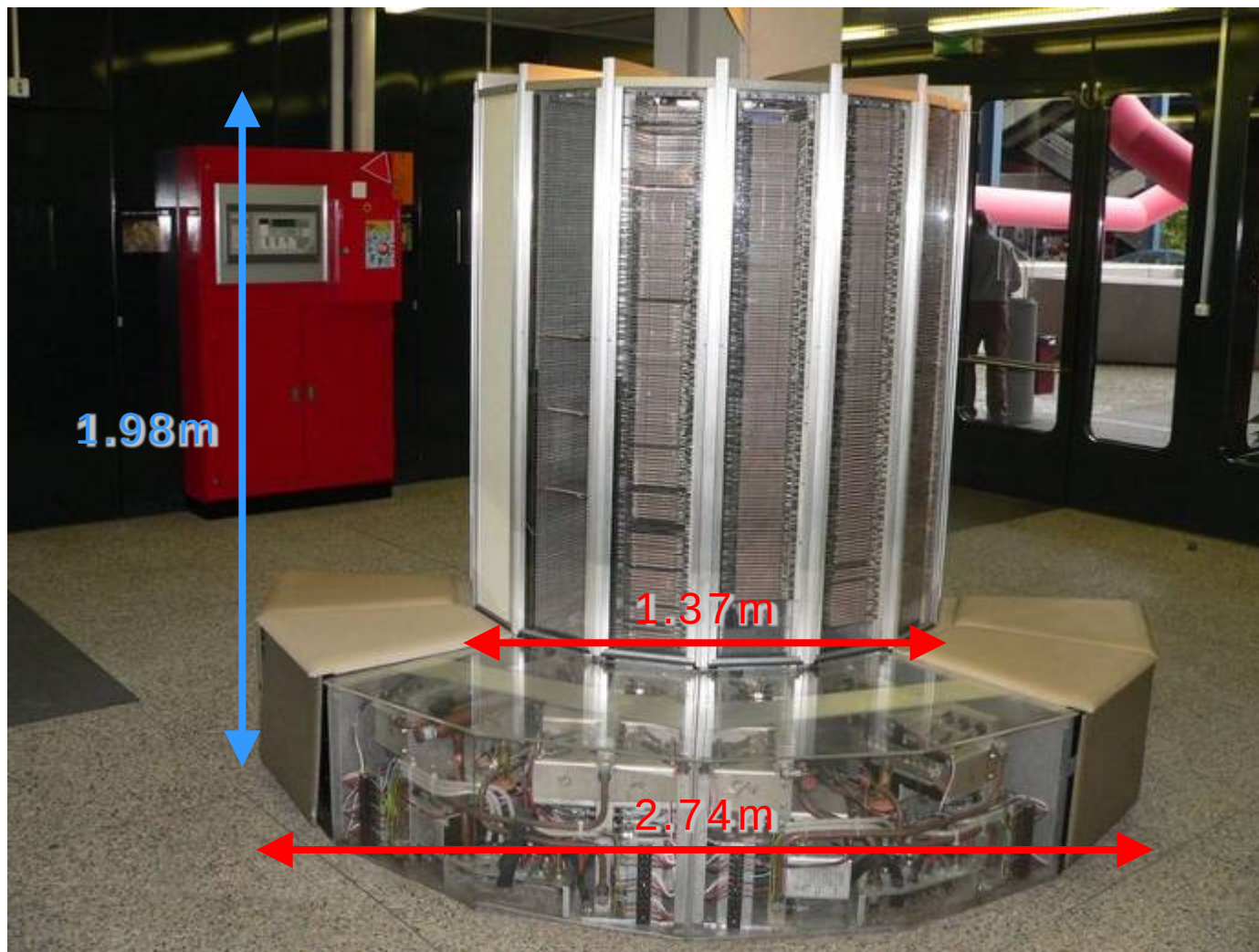


そもそもの始まり:ベクトルマシン (1)

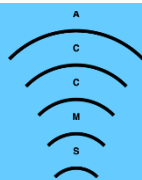
- 1976年:最初のスパコン **Cray-1** 登場
 - 動作周波数=80MHz (< 携帯電話)
 - 演算性能=**160MFlops** (< 携帯電話)
 - 消費電力=115kW
 - **大量の数値データ**(ベクトル)に対する同種演算が得意
- 1976年(中島=20歳)での「スーパー」度
 - 最速@京大(富士通 F230-75) < 5MFlops
 - 最速@京大情報工学科(日立 H8350) < 1MFlops
 - Intel 8086/87(1978/80) $\div$ 50KFlops



そもそもの始まり:ベクトルマシン (2)



source: <http://en.wikipedia.org/wiki/Image:Cray-1-p1010221.jpg>

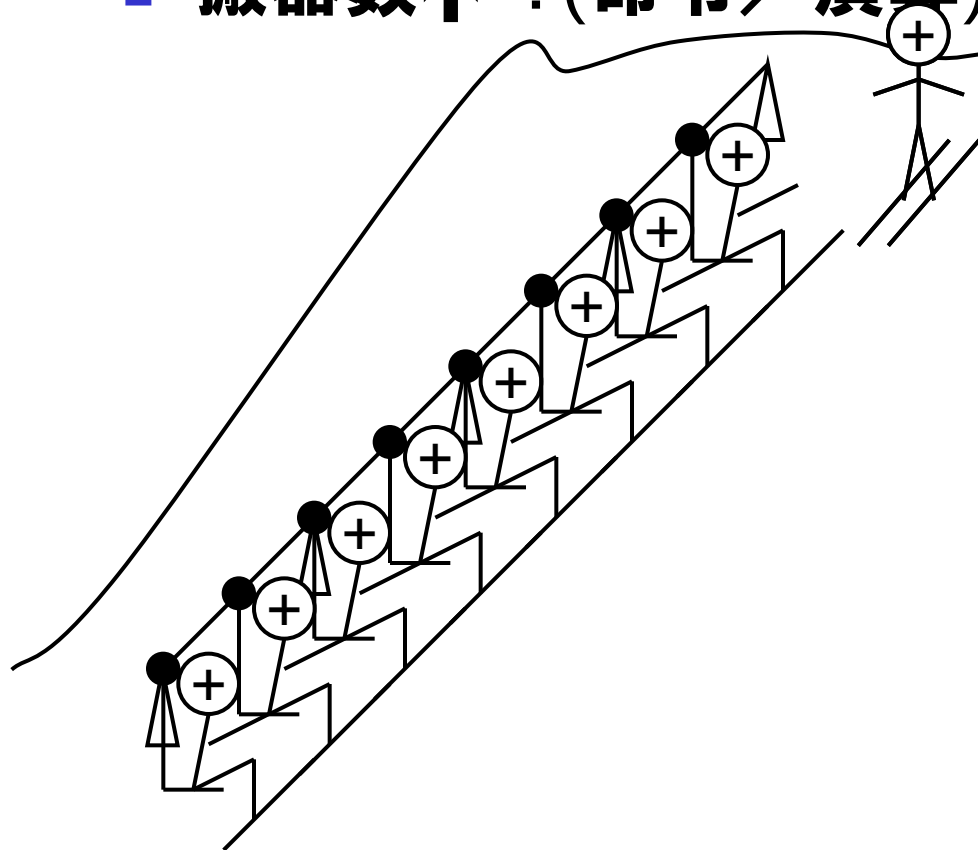


少し話を戻して

スーパーにする方法: 1970~

© 2014 H. Nakashima

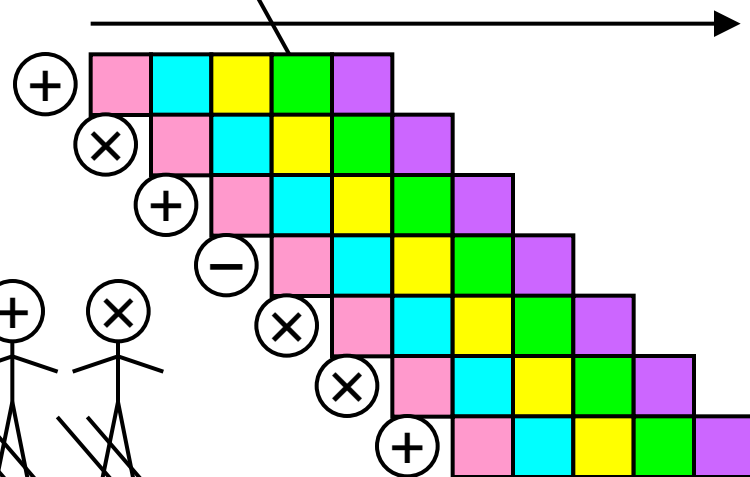
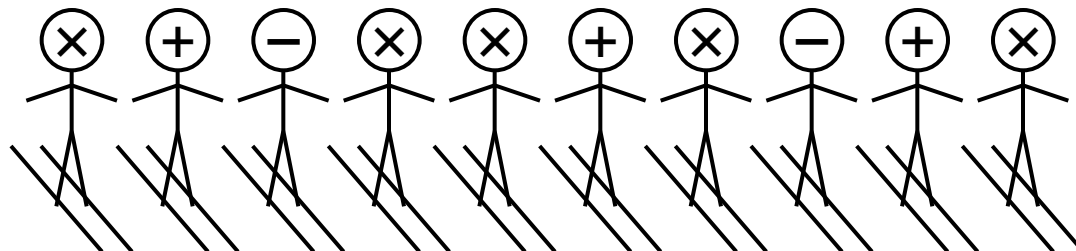
■ 搬器数↑ ≡ (命令 / 演算) **パイプライン**

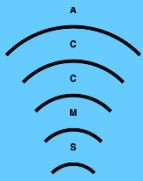


■ $z = x + y$ (加算命令) の手順

- 命令を取ってくる
- 加算だと判る
- x と y を取ってくる
- 加算をする
- 結果を z に入れる

■ これを1つずつずらして行う





スーパーにする方法：ベクトル計算の原理 (1)

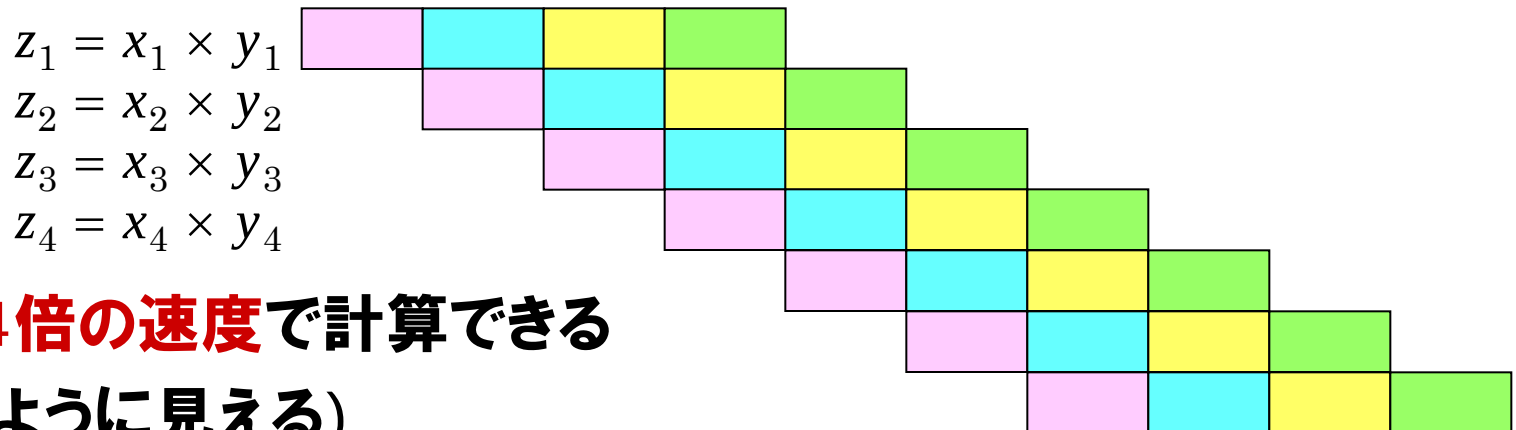
■ 大量数値データの同種演算を高速に行う方法

例： $z_i = x_i \times y_i$ ($i = 1, 2, \dots$)

- 1つの乗算をいくつか(たとえば4つ)の小さい操作に分ける

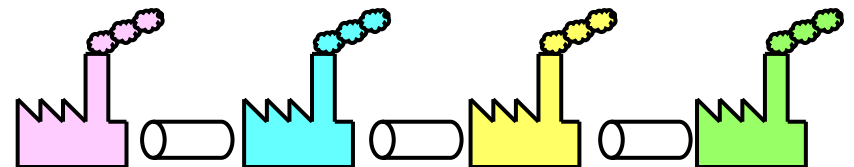
$$z_i = x_i \times y_i$$


- 多数の乗算を1小操作ずつずらして行う



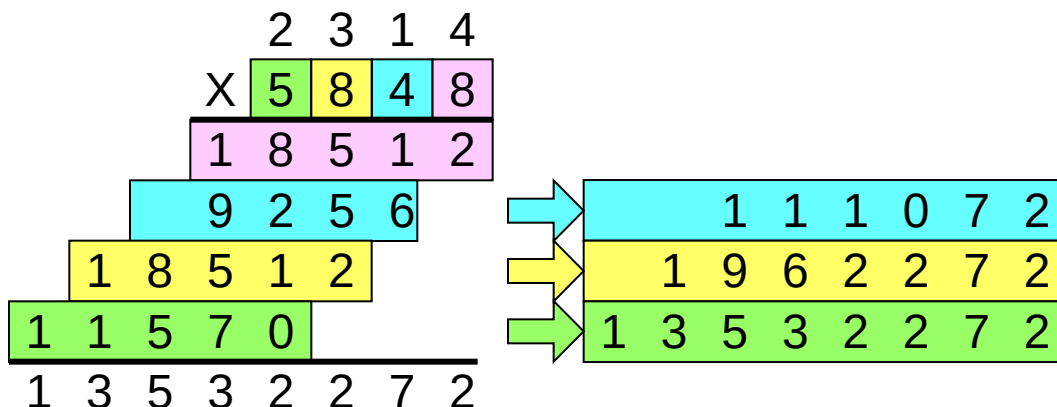
→ **4倍の速度**で計算できる
(ように見える)

→ (演算) **パイプライン処理**



スーパーにする方法：ベクトル計算の原理 (2)

- 乗算を4分割して**ずらす**考え方(たとえ話≠真実)



$$2314 \times 5848 = 13532272$$

$$2314 \times 8 + 2314 \times 40 + 2314 \times 800 + 2314 \times 5000$$

$$7872 \times 6752 = 53151744$$

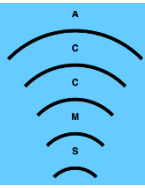
$$7872 \times 2 + 7872 \times 50 + 7872 \times 700 + 7872 \times 6000$$

$$1778 \times 7142 = 12698476$$

$$1778 \times 2 + 1778 \times 40 + 1778 \times 100 + 1778 \times 7000$$

$$8485 \times 1651 = 13843635$$

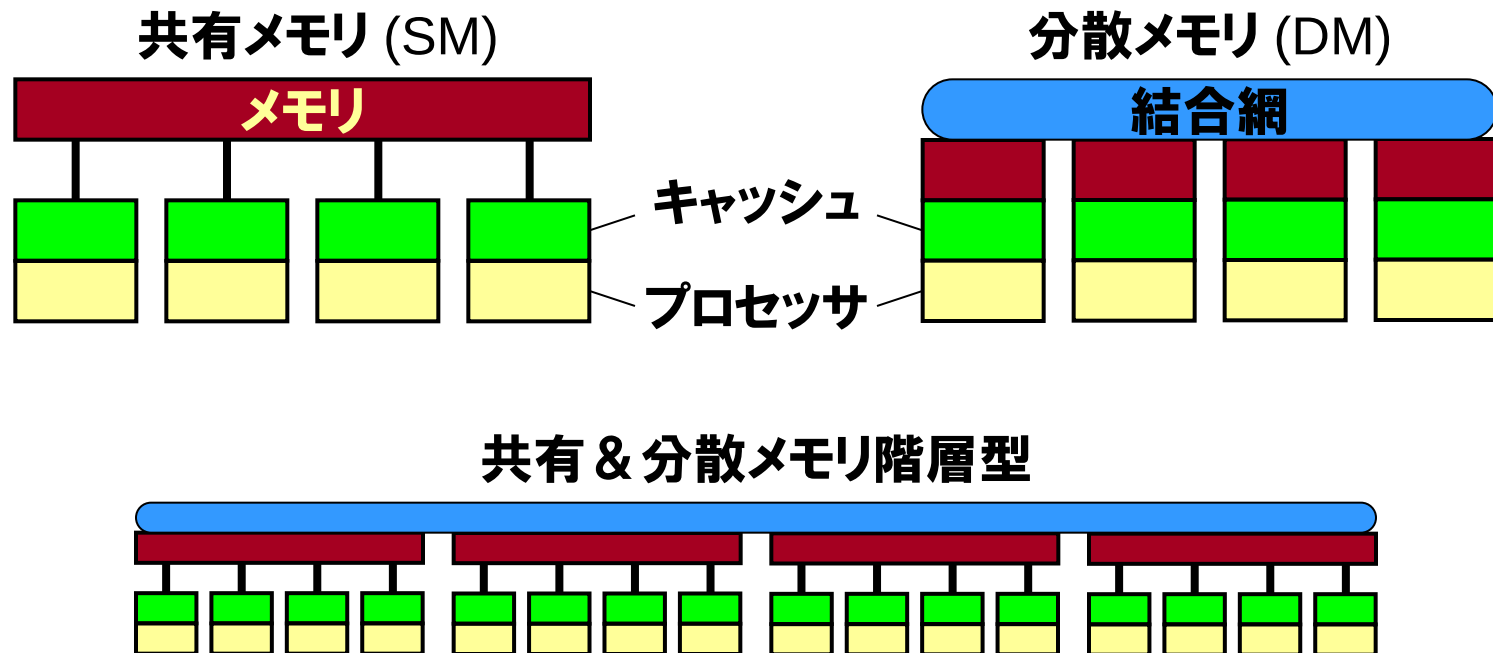
$$8385 \times 1 + 8385 \times 50 + 8385 \times 600 + 8385 \times 1000$$

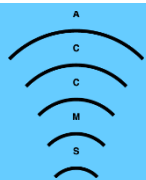


スーパーコンピュータの歴史(に戻って) もう一つの方法: 並列マシン

© 2014 H. Nakashima

- 1980年代: **スカラーマルチプロセッサ台頭**
 - 多数のパソコン(のようなもの)の集合体
 - Sequent Balance : 20 x NS32016 ('84)
 - Intel iPSC/1: 128 x i80286 ('85)





また話を戻して

スーパーにする方法:1990~

© 2014 H. Nakashima

■ 座席数↑ ≡ スーパースカラー / SIMD



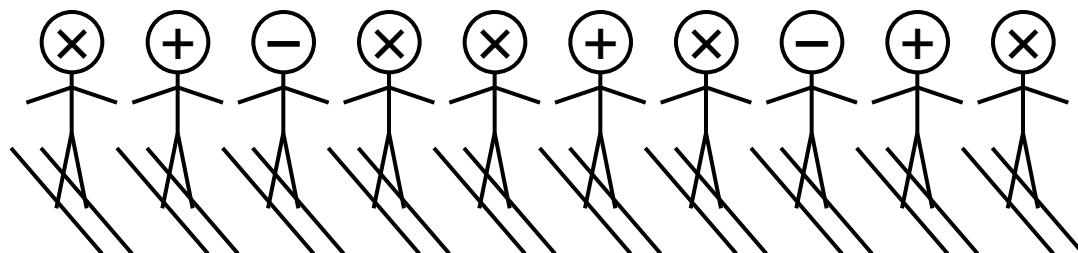
■ $= 2\text{GHz} \times \textcircled{8} \times 8 \times 88128$

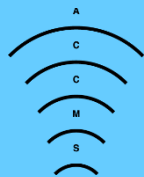
■ 加減算を4つと

■ 乗算を4つが

同時にできる

■ **同時にできる演算って？**

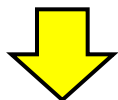




スーパーにする方法：並列演算

■ 3元連立一次方程式

$$\begin{cases} 2x - 3y + 4z = 8 \\ 3x - 4y + 2z = 1 \\ 4x - 2y + 3z = 9 \end{cases}$$



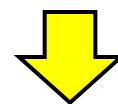
$$\begin{cases} x - \frac{3}{2}y + \frac{4}{2}z = \frac{8}{2} \\ x - \frac{4}{3}y + \frac{2}{3}z = \frac{1}{3} \\ x - \frac{2}{4}y + \frac{3}{4}z = \frac{9}{4} \end{cases}$$



同時にできる除(乗)算

同時にできる加減算

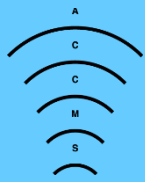
$$\begin{cases} \left(-\frac{4}{3} + \frac{3}{2}\right)y + \left(\frac{2}{3} - \frac{4}{2}\right)z = \frac{1}{6}y - \frac{8}{6}z = \frac{1}{3} - \frac{8}{2} = -\frac{22}{6} \\ \left(-\frac{2}{4} + \frac{3}{2}\right)y + \left(\frac{3}{4} - \frac{4}{2}\right)z = \frac{8}{8}y - \frac{10}{8}z = \frac{9}{4} - \frac{8}{2} = -\frac{14}{8} \end{cases}$$



$$\left(-\frac{10}{8} + 8\right)z = \frac{54}{8}z = -\frac{14}{8} + 22 = \frac{162}{8} \Rightarrow z = 3$$

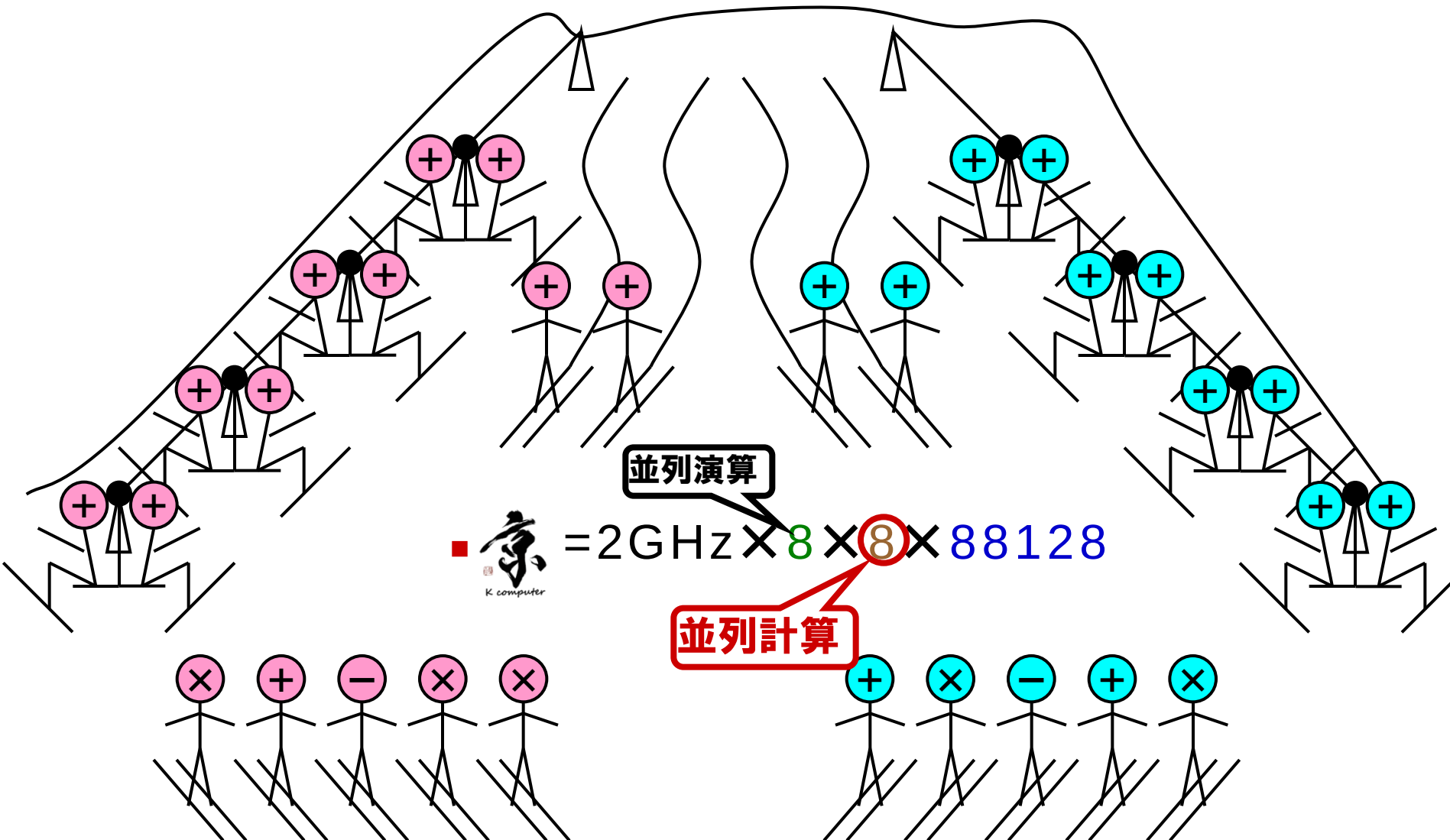
$$y = -22 + 8 \cdot 3 = 2$$

$$x = \frac{8}{2} + \frac{3 \cdot 2}{2} - \frac{4 \cdot 3}{2} = \frac{2}{2} = 1$$



スーパーにする方法:2000~ (1980~)

■ リフト数↑ ≡ マルチコア / 共有メモリ並列マシン

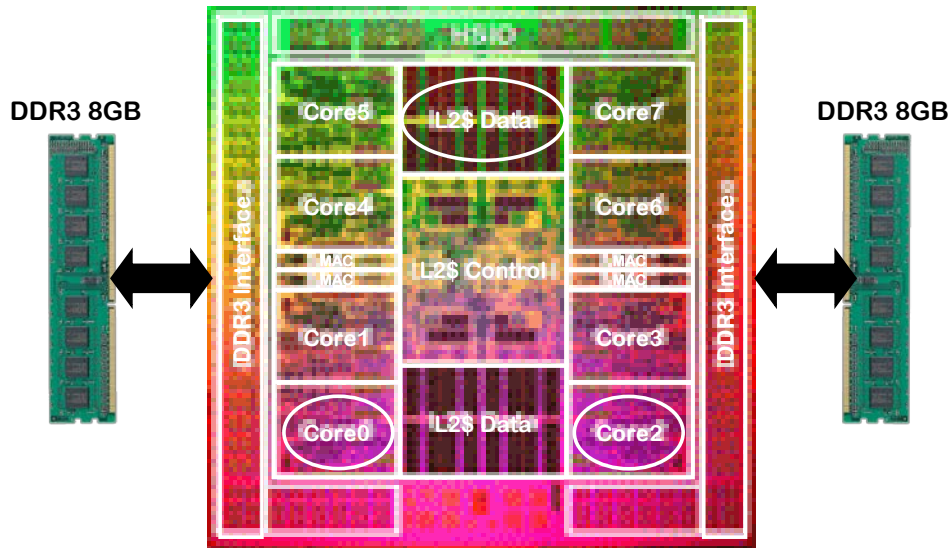




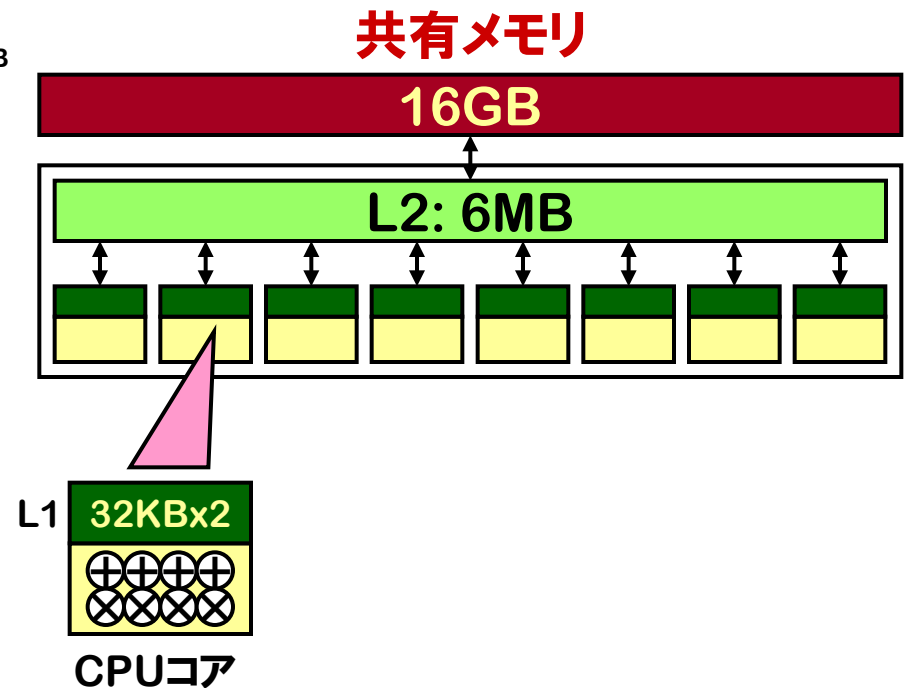
スーパーにする方法：**京** のプロセッサ

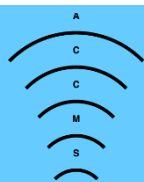
K computer

FUJITSU SPARC 64 VIIIfx



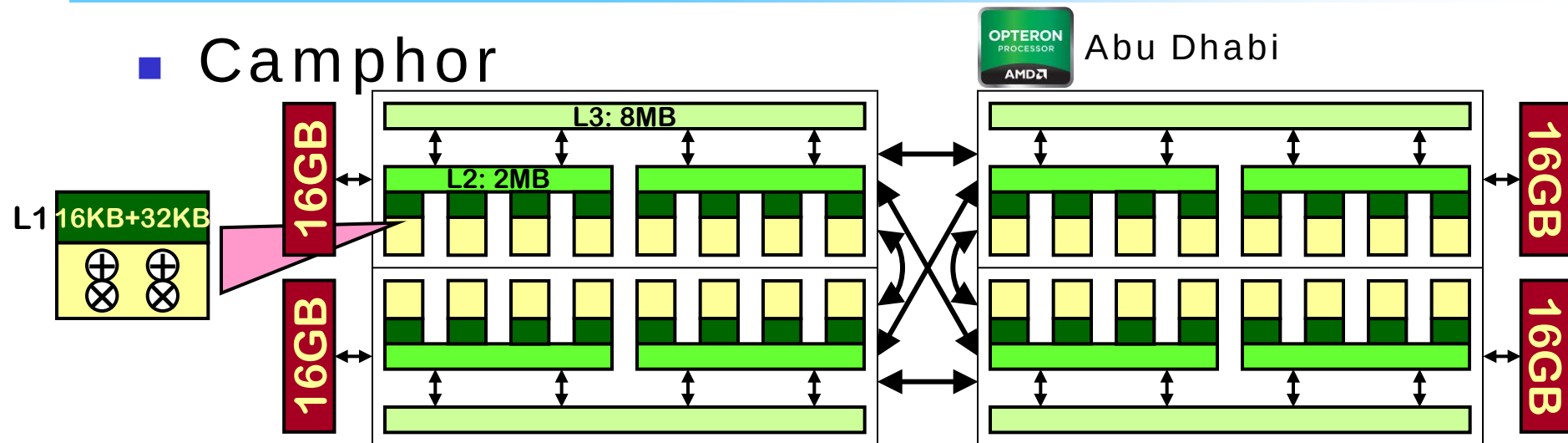
<http://www.aics.riken.jp/jp/k/system.html>



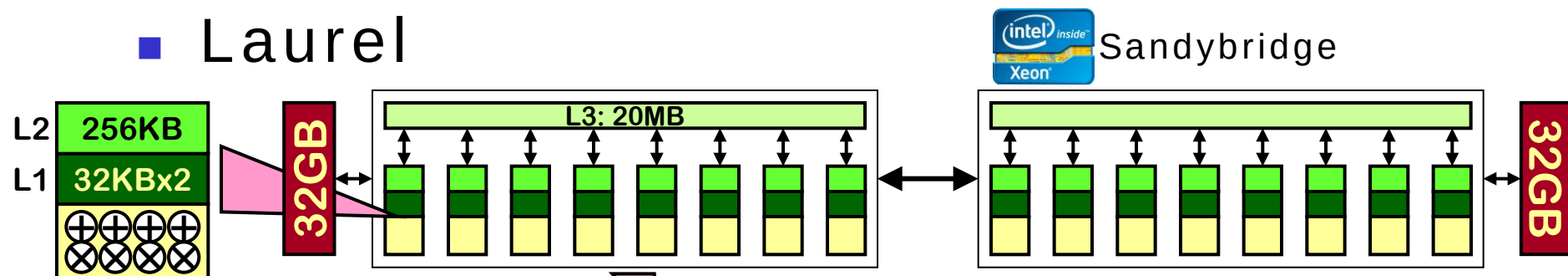


京大スパコンのプロセッサ (1/2)

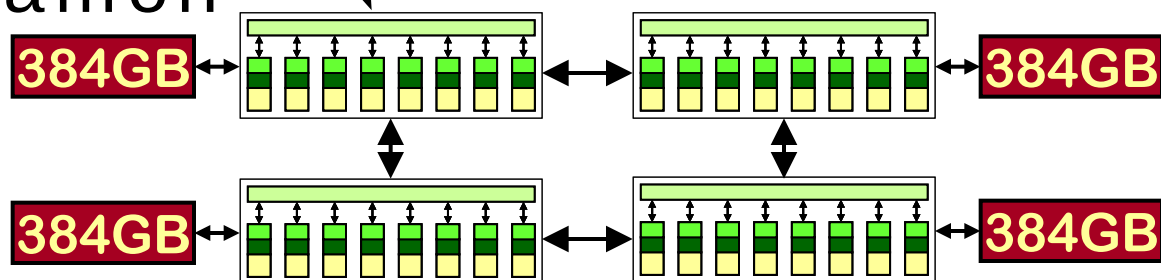
■ Camphor

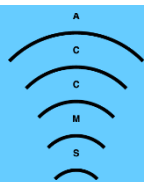


■ Laurel



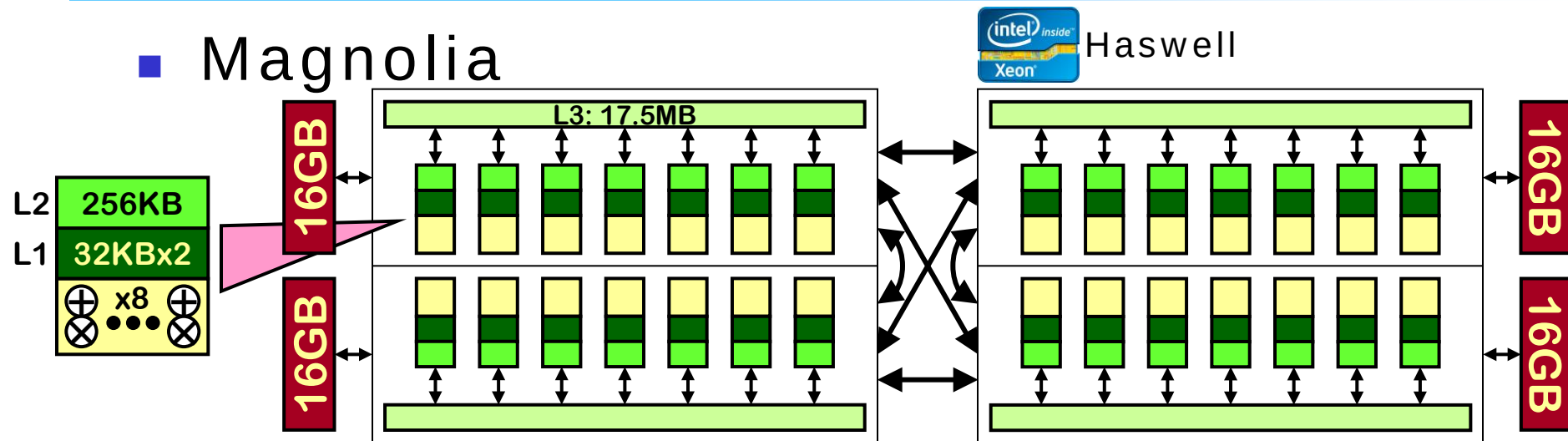
■ Cinnamon



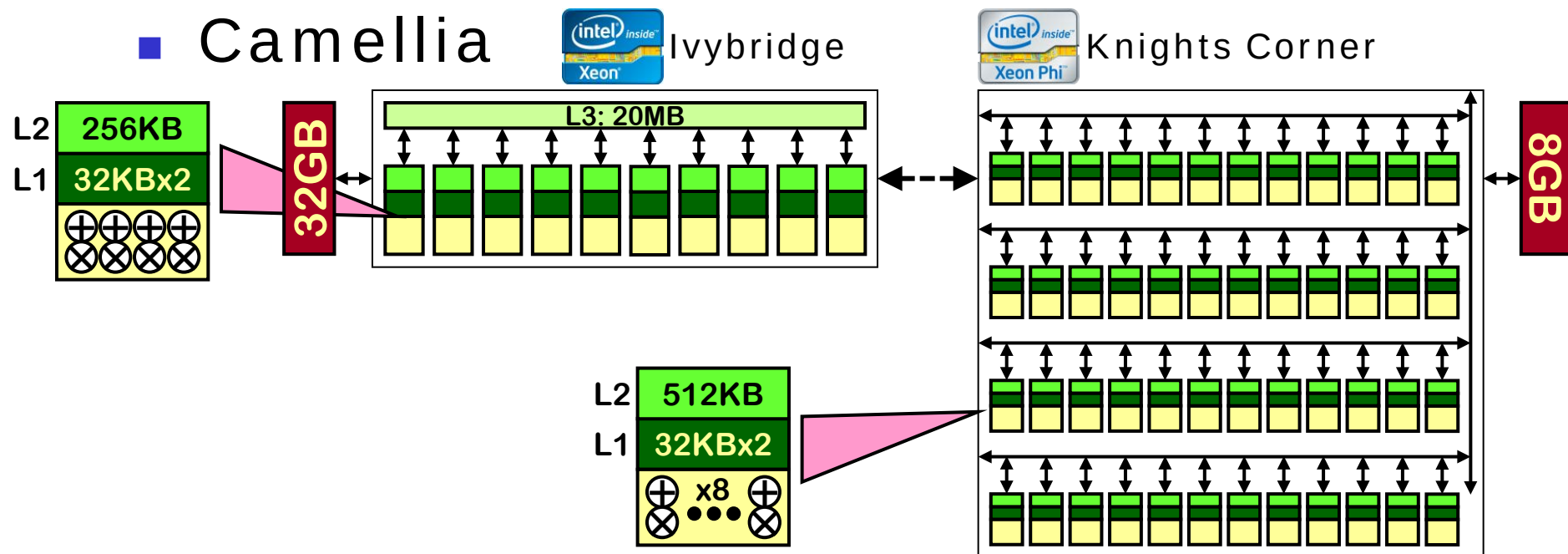


京大スパコンのプロセッサ (2/2)

■ Magnolia



■ Camellia



スーパーにする方法：連立方程式の並列計算

1行目担当の
コアが書いて

$$a'_{1j} = a_{1j} / a_{11}$$

$$a'_{ij} = a_{ij} / a_{i1} - a'_{1j}$$

i行目担当の
コアが読む

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + \cdots + a_{2n}x_n = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 + \cdots + a_{3n}x_n = b_3$$

$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + \cdots + a_{4n}x_n = b_4$$

...

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + a_{n4}x_4 + \cdots + a_{nn}x_n = b_n$$



$$x_1 + a'_{12}x_2 + a'_{13}x_3 + a'_{14}x_4 + \cdots + a'_{1n}x_n = b'_1$$

$$x_1 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 + \cdots + a'_{2n}x_n = b'_2$$

$$x_1 + a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 + \cdots + a'_{3n}x_n = b'_3$$

$$x_1 + a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 + \cdots + a'_{4n}x_n = b'_4$$

...

$$x_1 + a'_{n2}x_2 + a'_{n3}x_3 + a'_{n4}x_4 + \cdots + a'_{nn}x_n = b'_n$$

スーパーにする方法:1980~

■ リフト数↑↑↑↑↑↑↑↑↑↑ ≡ 超並列コンピュータ

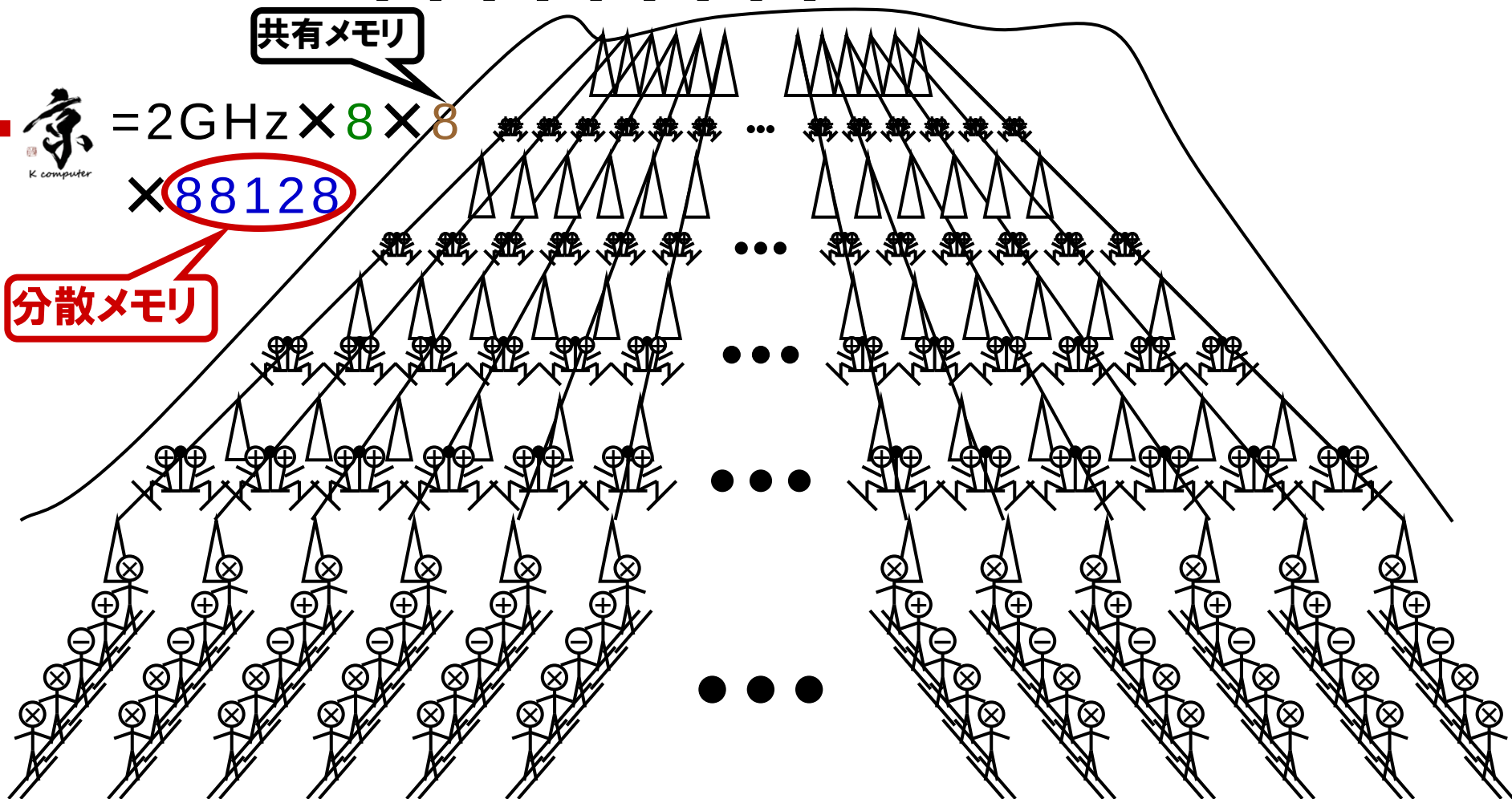
共有メモリ



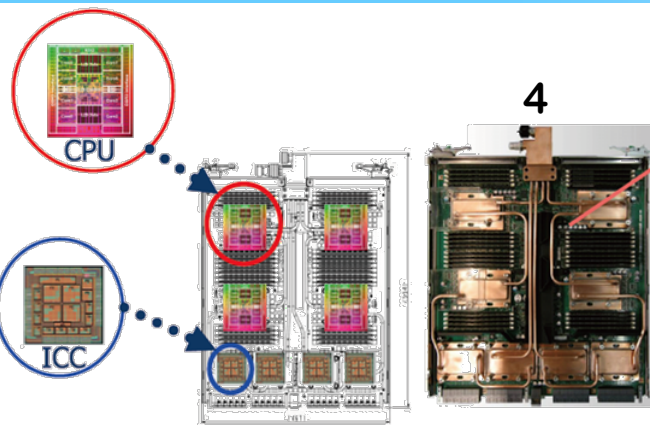
$$= 2\text{GHz} \times 8 \times 8$$

$$\times 88128$$

分散メモリ

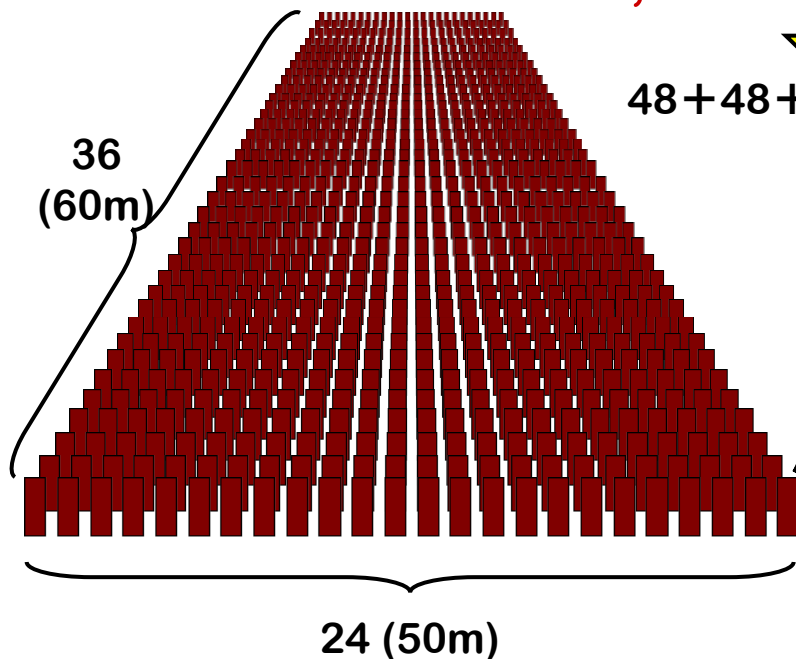


スーパーにする方法：京の全体像



<http://www.aics.riken.jp/jp/k/system.html>

$$102 \times 24 \times 36 = 102 \times 864 = 88,128$$



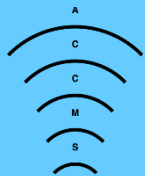
4×12

$$48 + 48 + 6 = 102$$



京計算機室
60m x 50m

京大体育館
56m x 54m

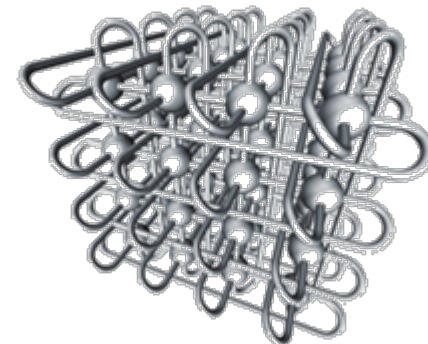
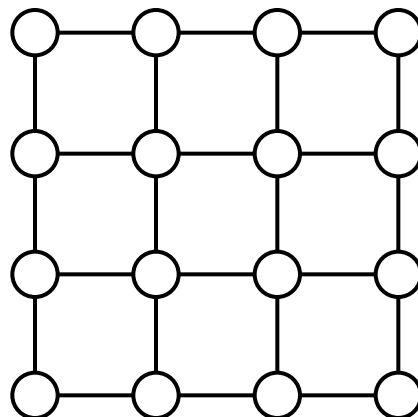


スーパーにする方法：K computer 京 の通信路 (1/2)

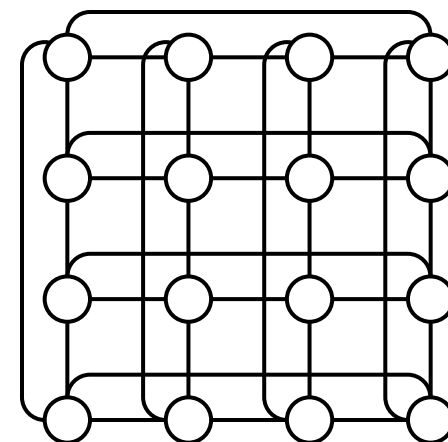
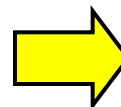
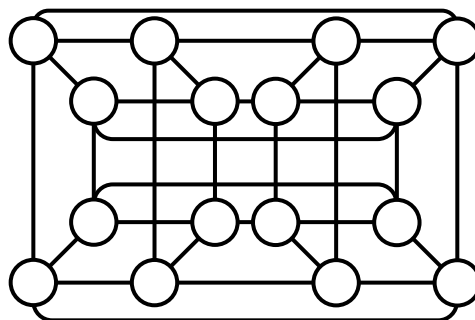
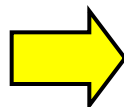
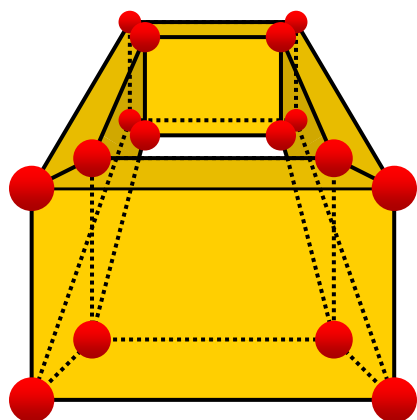
■ 6次元メッシュ／トーラス結合網 Tofu

■ って意味不明～

■ 2次元メッシュ



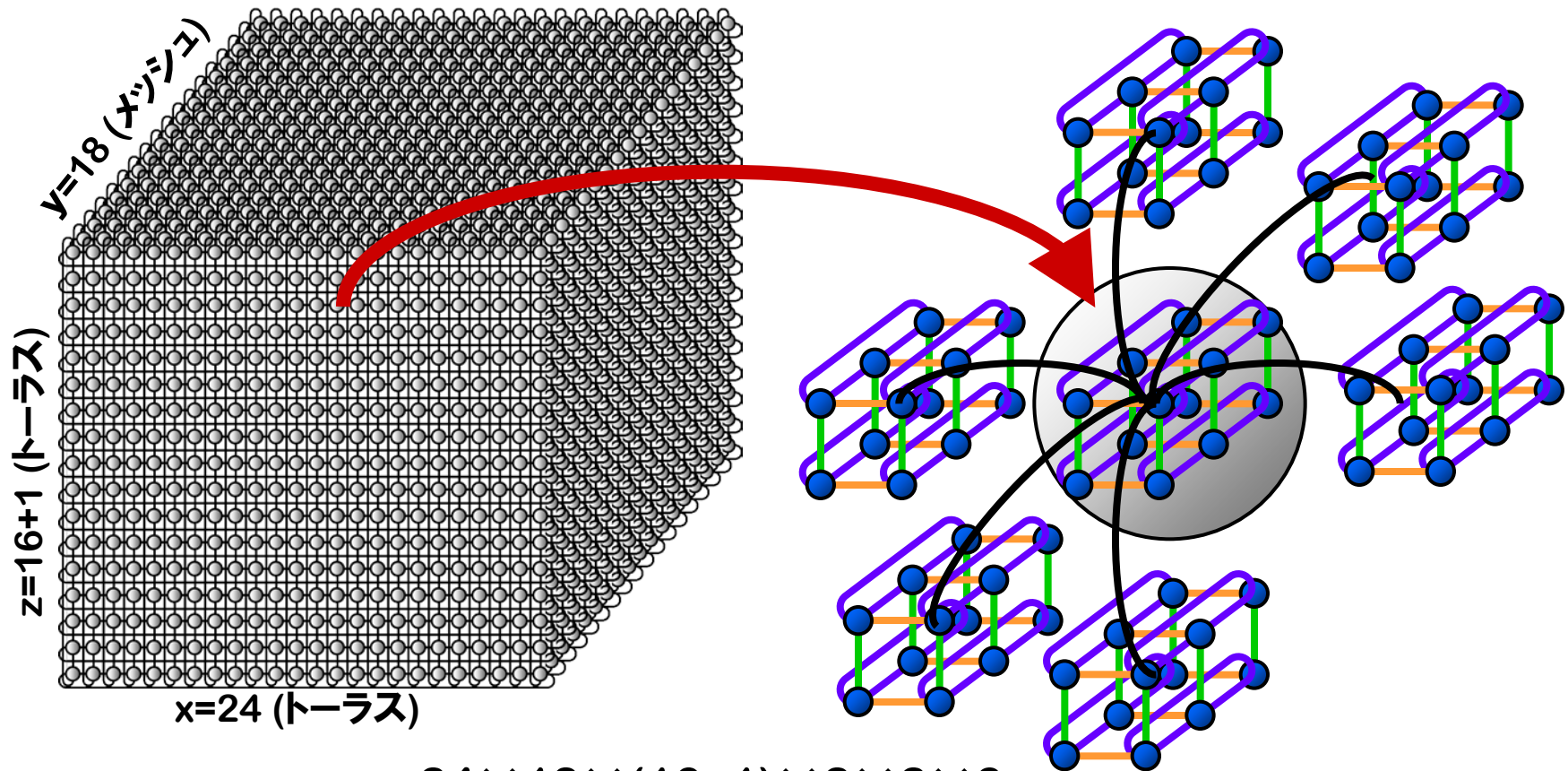
■ 2次元トーラス (ドーナツの表面)



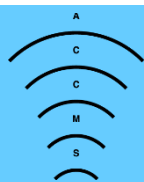


スーパーにする方法：K computer 京 の通信路 (2/2)

■ 6次元メッシュ/トーラス結合網 Tofu



$$24 \times 18 \times (16+1) \times 2 \times 2 \times 3 \\ = 88,128$$



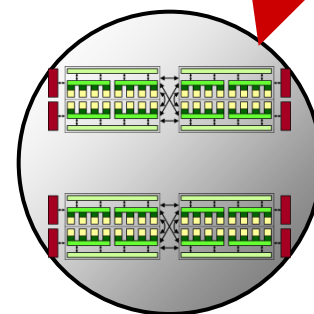
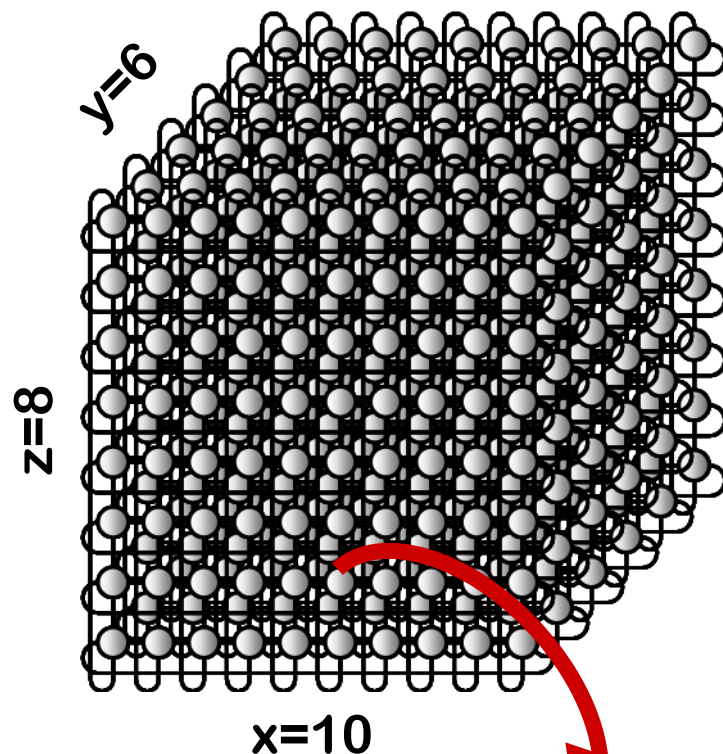
京大スパコンの全体像 (1/4)

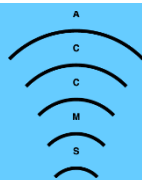
■ Camphor



CRAY XE6

$$2 \times (10 \times 8 \times 6 - 10) = 940$$





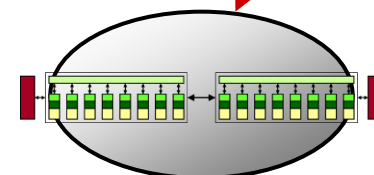
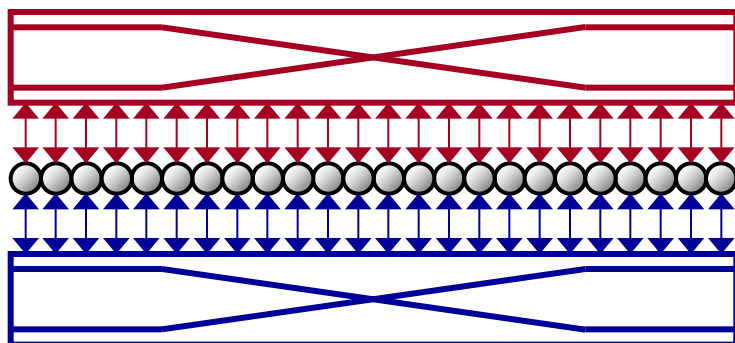
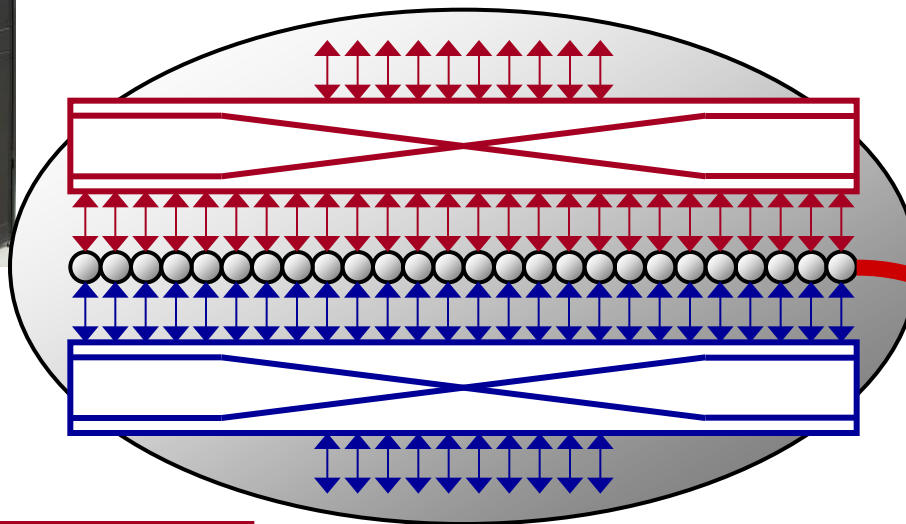
京大スパコンの全体像 (2/4)

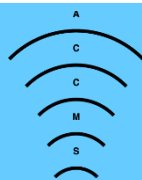
■ Laurel



GreenBlade 8000

$$24 \times 26 - 23 = 601$$



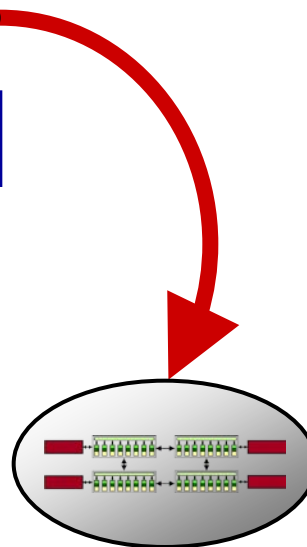
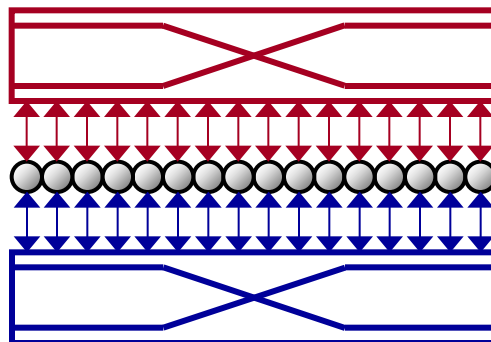


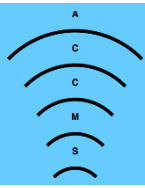
京大スパコンの全体像 (3/4)

■ Cinnamon



2548X



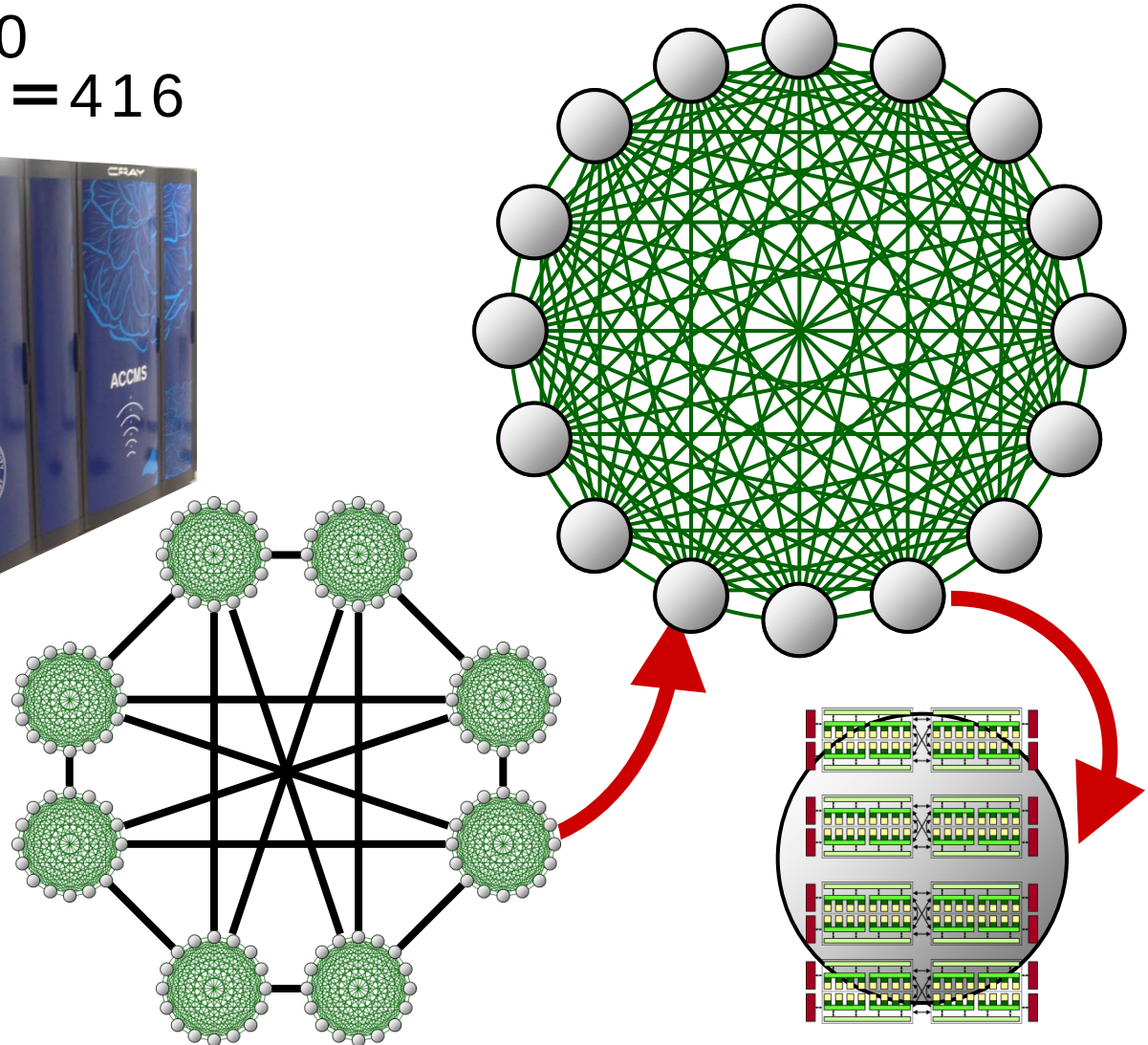


京大スパコンの全体像 (4/4)

■ Magnolia (& Camellia)

CRAY XC30

$$4 \times (16 \times 8 - 24) = 416$$



スーパーにする方法：連立方程式の並列計算

1行目担当の
プロセッサから

$$a'_{1j} = a_{1j} / a_{11}$$

$$a'_{ij} = a_{ij} / a_{i1} - a'_{1j}$$

全てのプロセッサへ
通信 (放送)

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + \cdots + a_{2n}x_n = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 + \cdots + a_{3n}x_n = b_3$$

$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + \cdots + a_{4n}x_n = b_4$$

...

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + a_{n4}x_4 + \cdots + a_{nn}x_n = b_n$$



$$x_1 + a'_{12}x_2 + a'_{13}x_3 + a'_{14}x_4 + \cdots + a'_{1n}x_n = b'_1$$

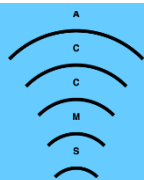
$$x_1 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 + \cdots + a'_{2n}x_n = b'_2$$

$$x_1 + a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 + \cdots + a'_{3n}x_n = b'_3$$

$$x_1 + a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 + \cdots + a'_{4n}x_n = b'_4$$

...

$$x_1 + a'_{n2}x_2 + a'_{n3}x_3 + a'_{n4}x_4 + \cdots + a'_{nn}x_n = b'_n$$



スーパーコンピュータの歴史(にまた戻って)

ベクトル vs 並列

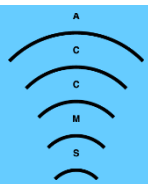
© 2014 H. Nakashima

■ 1990年代:ベクトル並列 vs スカラー並列

■ TOP-10 of @ 1993.6

machine	#proc	Rmax	Rpeak
TMC CM-5	1024	59.7	131.0
TMC CM-5	544	30.4	69.6
TMC CM-5	512	30.4	65.5
TMC CM-5	512	30.4	65.5
NEC SX-3	4	23.2	25.6
NEC SX-3	4	20.0	22.0
TMC CM-5	256	15.1	32.8
Intel Delta	512	13.8	20.5
Cray Y-MP	16	13.7	15.2
Cray Y-MP	16	13.7	15.2

- 巨大で(>100万元)密な連立一次方程式の求解性能に基づく世界中のスパコン順位表
- 1993.6から毎年2回発表(6月&11月)
- Rmax: 求解性能
Rpeak: 理論最大性能
(単位GFlops: 毎秒10億演算)

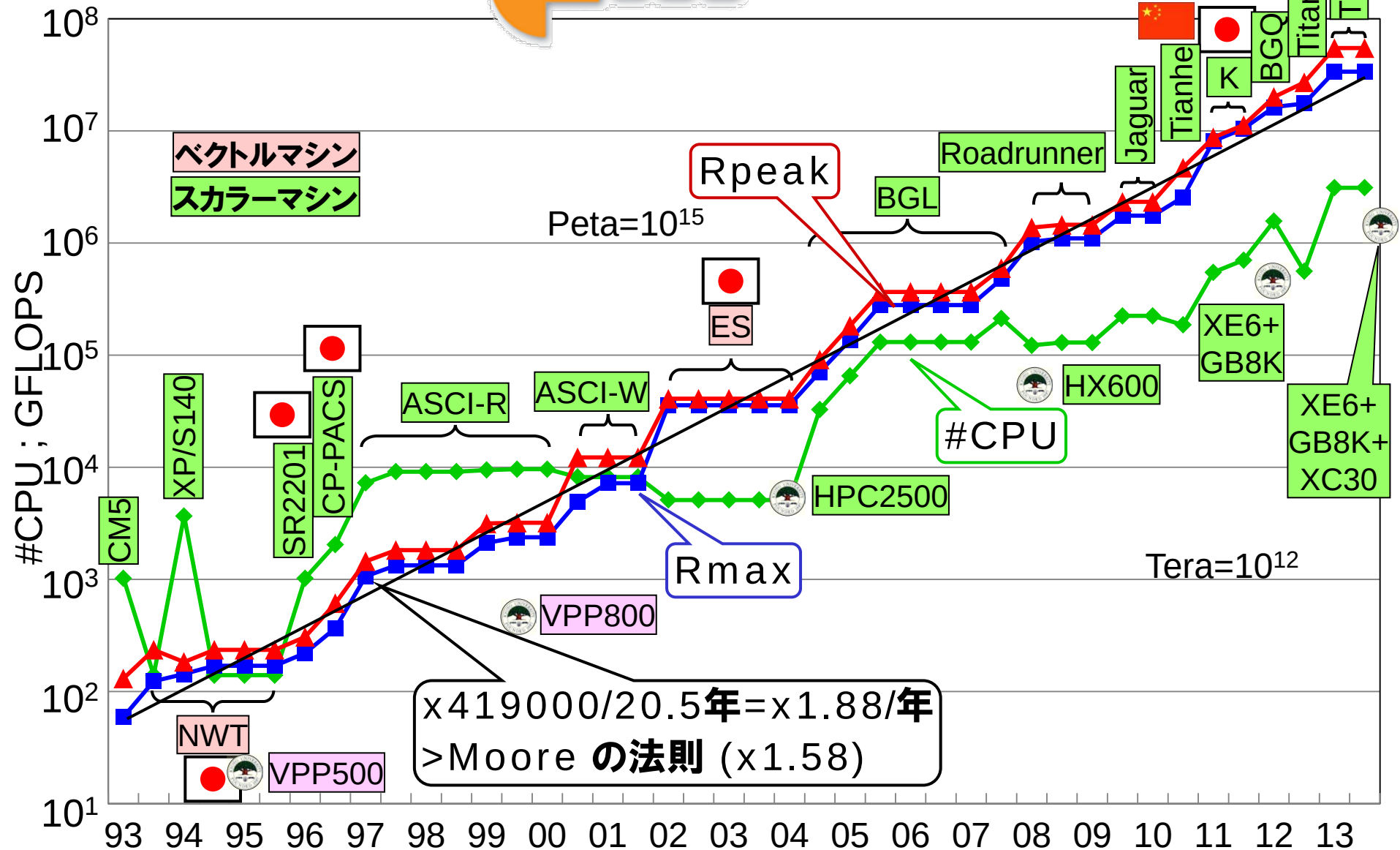


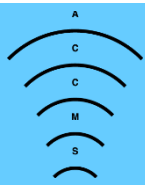
スーパーコンピュータの歴史

© 2014 H. Nakashi



Top 1 of





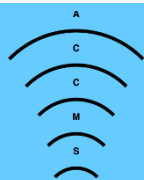
(いきなり&とりあえず) まとめ

■ ベクトルマシン

- 1つの演算を k 個の小さい操作に分割する
- 多数の同種演算を1小操作ずつずらして行う
- k 倍の速度で計算できる(ように見える)
- 大量 ($\gg k$) の同種演算が得意

■ 並列マシン

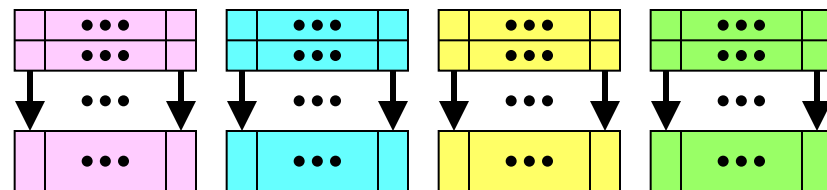
- 多数の同じ(ような)演算を p 個のCPUに分割
 - それぞれのCPUが割当てられた計算をする
 - p 倍の速度で計算できる(ように見える)
 - 大量 ($\gg p$) の同じ(ような)演算が得意
- スパコンは大量の同じ(ような)演算(や処理)が得意



大量同種演算は何でも得意か？ (1/2)

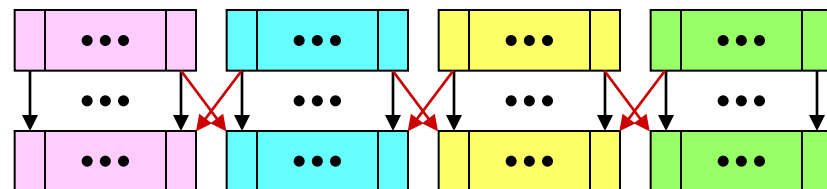
■ 超得意

$$Z_i = x_i + y_i$$



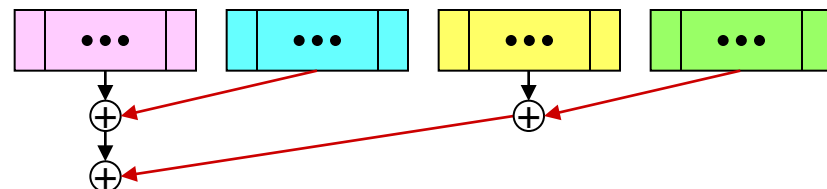
■ 普通に得意

$$Z_i = (x_{i-1} + 2x_i + x_{i+1}) / 4$$



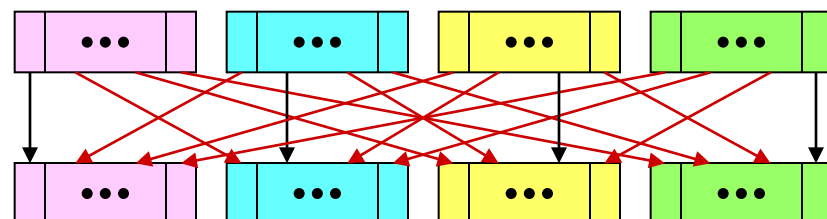
■ 微妙に得意

$$Z = x_1 + x_2 + \dots + x_n$$



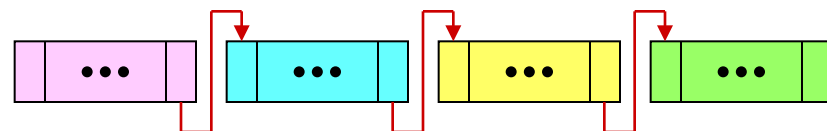
■ 何とかなる

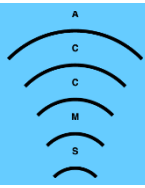
$$Z_i = x_{f(i)} \text{ s.t. } z_1 \leq z_2 \leq \dots \leq z_n$$



■ 全然ダメ

$$z_1 = f(x_1, 0), z_i = f(x_i, z_{i-1})$$



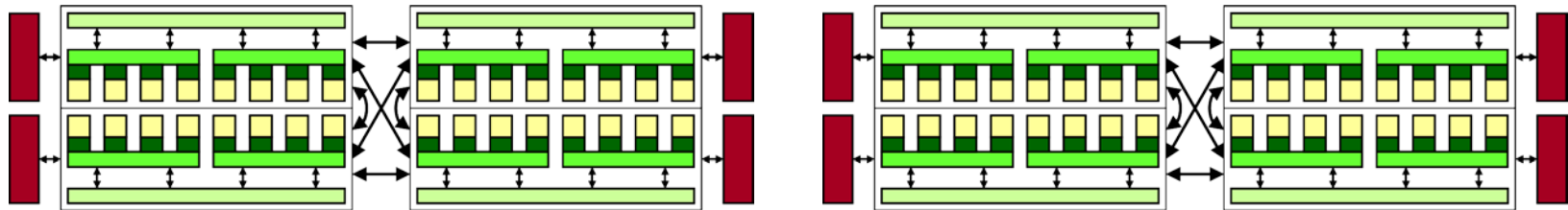


大量同種演算は何でも得意か？ (2/2)

■ 京大スパコン (Camphor) の通信速度

320GFlops

320GFlops



$2\mu\text{sec}$

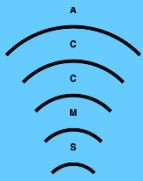


9.3GB/sec
= 74.4Gbit/sec
= 200Mbit/sec $\times$ 372

$21.9\text{TB/sec} = 17.5\text{Tbit/sec} = 200\text{Mbit/sec} \times 874,000$

- 1個の数値(8B)の通信時間 = $2\mu\text{sec}$
= 640,000個分の演算時間
- 10億個の数値(8GB)の通信時間 = 0.86秒
= 2,752億個分の演算時間

京では $\times 1.1$ 億



まとめ & 課題

■ スーパーコンピュータは ...

- 大量の同じ(ような)演算(や処理)が得意
- ただし演算どうしの**依存性が少ない**ことが必要

→ そんな都合のよい問題はあるのか？

■ そこで**レポート課題**

(できればスパコンに適する大規模な)並列計算により高い性能が期待できる**実際的な**問題を一つ挙げ、なぜその問題が並列計算に適するのかを説明せよ。