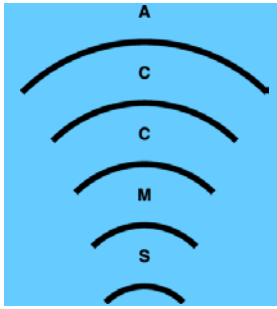



計算科学が拓く世界

# 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**#1-2

# 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~ 4 | 中島浩 | ACCMS | 高性能システムとプログラミング |
| 1 | 中島浩 | ACCMS | スーパーコンピュータは何故スーパーか(1) |
| 2 | 中島浩 | ACCMS | スーパーコンピュータは何故スーパーか(2) |
| 3 | 小山田耕二 | ACCMS | データ分析と可視化 |
| 4 | 木村欣司 | 情・数理 | (偏)微分方程式の差分解析 |
| 05~08 | 牛島省 | ACCMS | 応用計算力学 |
| 9~11 | 大村善治 | 生存圏研 | 地球・惑星・宇宙と計算科学 |
| 12~14 | 阿久津達也 | 化研 | 生命科学 |

# 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

# どのぐらいスーパー？ (1/2)

-  は  の180**万倍**も高速

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

- **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**
  - $10^{-308}$~$10^{308}$**の実数を近似的に**(10**進**16**桁精度**)**表現したもの**
  - **2.99792458...×10$^{8}$(m/s), 9.1093829140...×10$^{-31}$(kg)**
-  11.5 P(**ペタ** $10^{15}$)FLOPS (1.15 京 )

÷  6.4 G(**ギガ** $10^{9}$ ) FLOPS (64**億**)

＝1,797,120

# どのぐらいスーパー？(2/2)

- 京 =  ×180万と  =  ×3
  は**話が違う**
- **同じ土俵で比べるなら**
  - N700系　：300km/h×1323人＝396,900人・km/h
  ÷B767-300：880km/h×　270人＝237,600人・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**
  - 京：2.0GHz×8×8×88,128
  ÷  ：1.6GHz×2×2×1
  ＝1,797,120

ここがスーパー

intel Celeron J3060

intel Core i7なら
3.0GHz×16×10も

# スーパーコンピュータ (スパコン) とは (1/2)

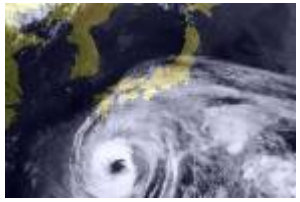
- パソコンの数千倍～数万倍の規模・性能を持つ巨大な超高速コンピュータ
  - 世界最大・最高速マシン ≒パソコン x 1960万
  - 京大スーパーコンピュータ ≒パソコン x 102万

  ➔パソコンで1ヶ月かかる計算＝0.13秒～2.6秒

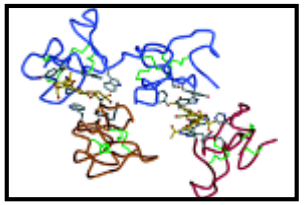
  （ただしスパコン向きの問題をうまくプログラムしたら）
- スパコンが高速な理由
  - 個々の部品(CPU, メモリなど)≒パソコン
  - 非常に多数のパソコン(のようなもの)の集合体
    - パソコン = 1~16 CPU
    - 京大スパコン = 153,512 CPU
    - 世界最高速スパコン = 10,649,600 CPU
    - 世界最大規模スパコン = 10,649,600 CPU

# 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（x 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1ml = 3.3 x 1兆 x 100億）

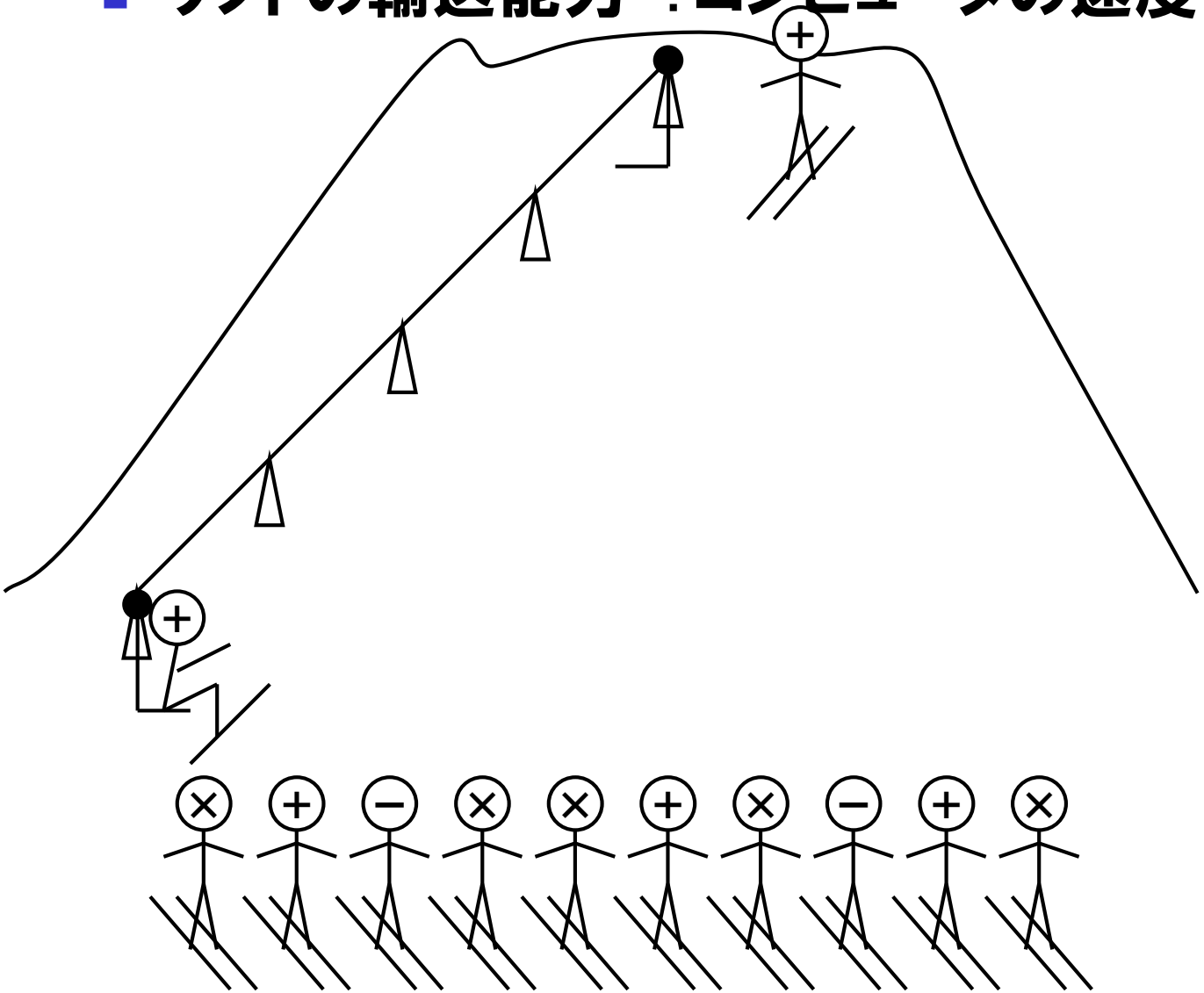


  - 自動車の空力・衝突解析
    → $1mm^2$ or $1cm^3$ あたり1データ
    →データ数＝1~10億

  - Web 文書の解析
    （←自動翻訳用データ作成など）
    →文書数＝数100億～数1000億

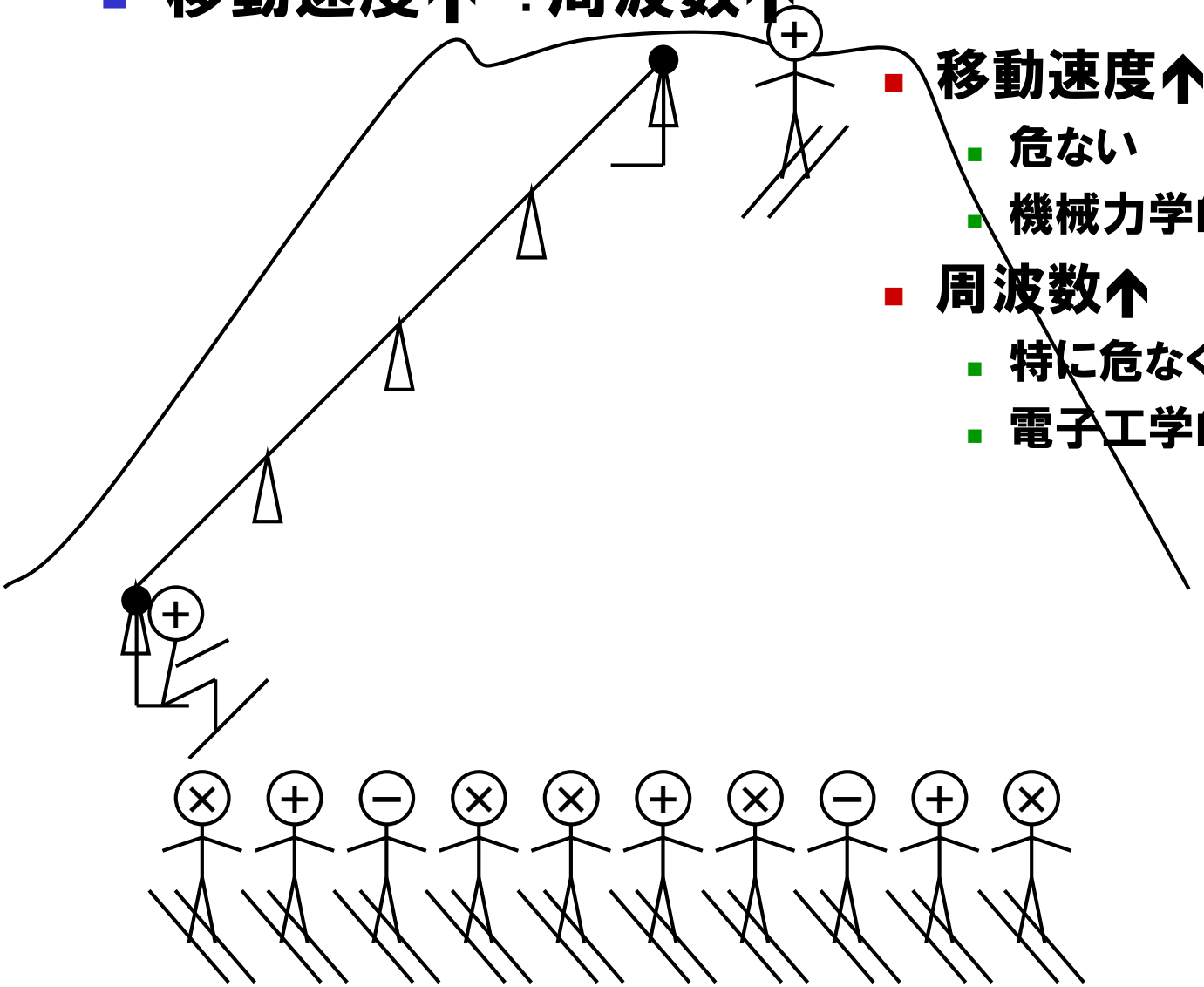
# スーパーにする方法

- リフトの輸送能力≒コンピュータの速度



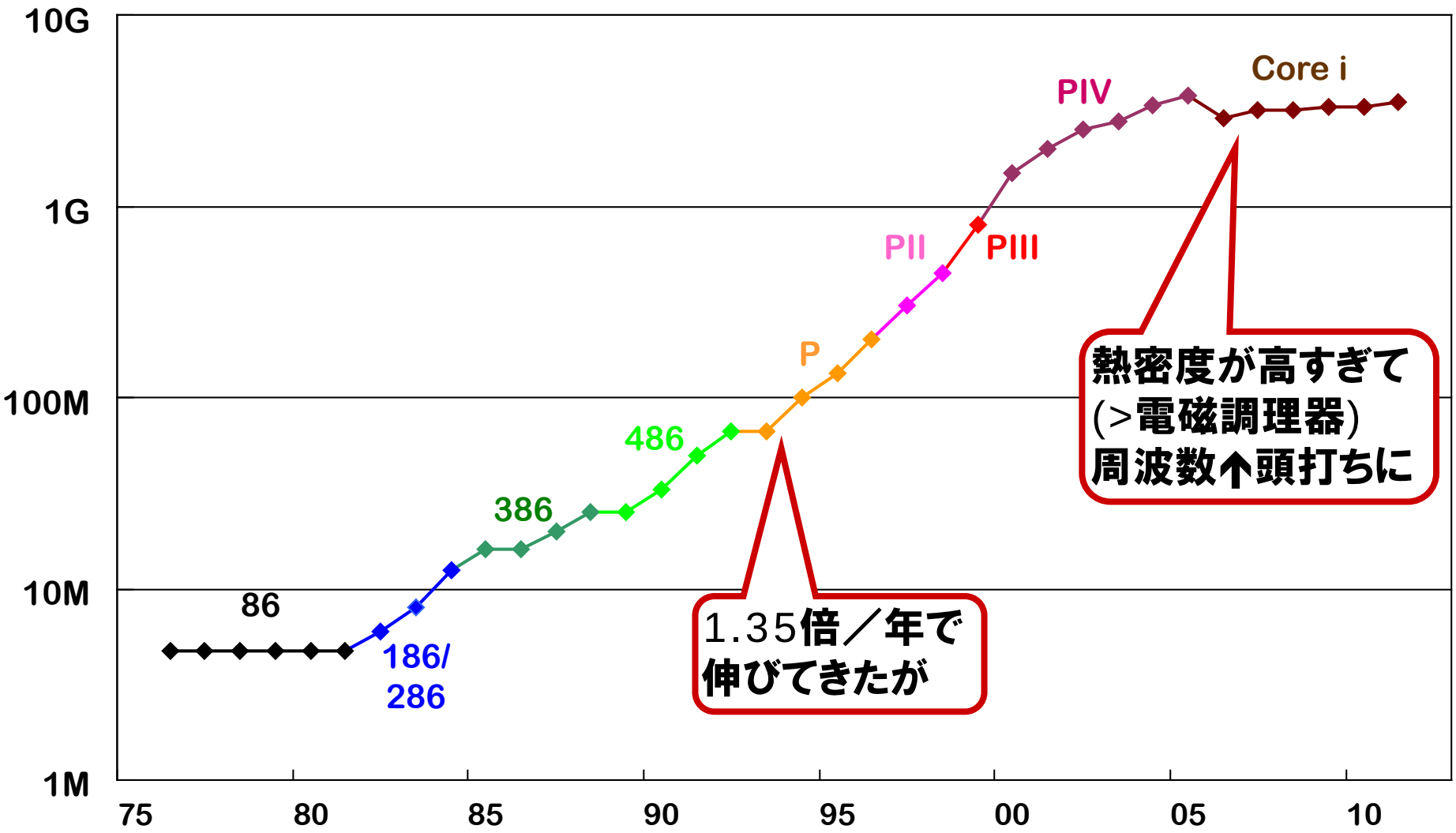
# スーパーにする方法:～1970

- 移動速度↑≒周波数↑



- 移動速度↑
  - 危ない
  - 機械力学的に無理
- 周波数↑
  - 特に危なくはない
  - 電子工学的に無理ではない？

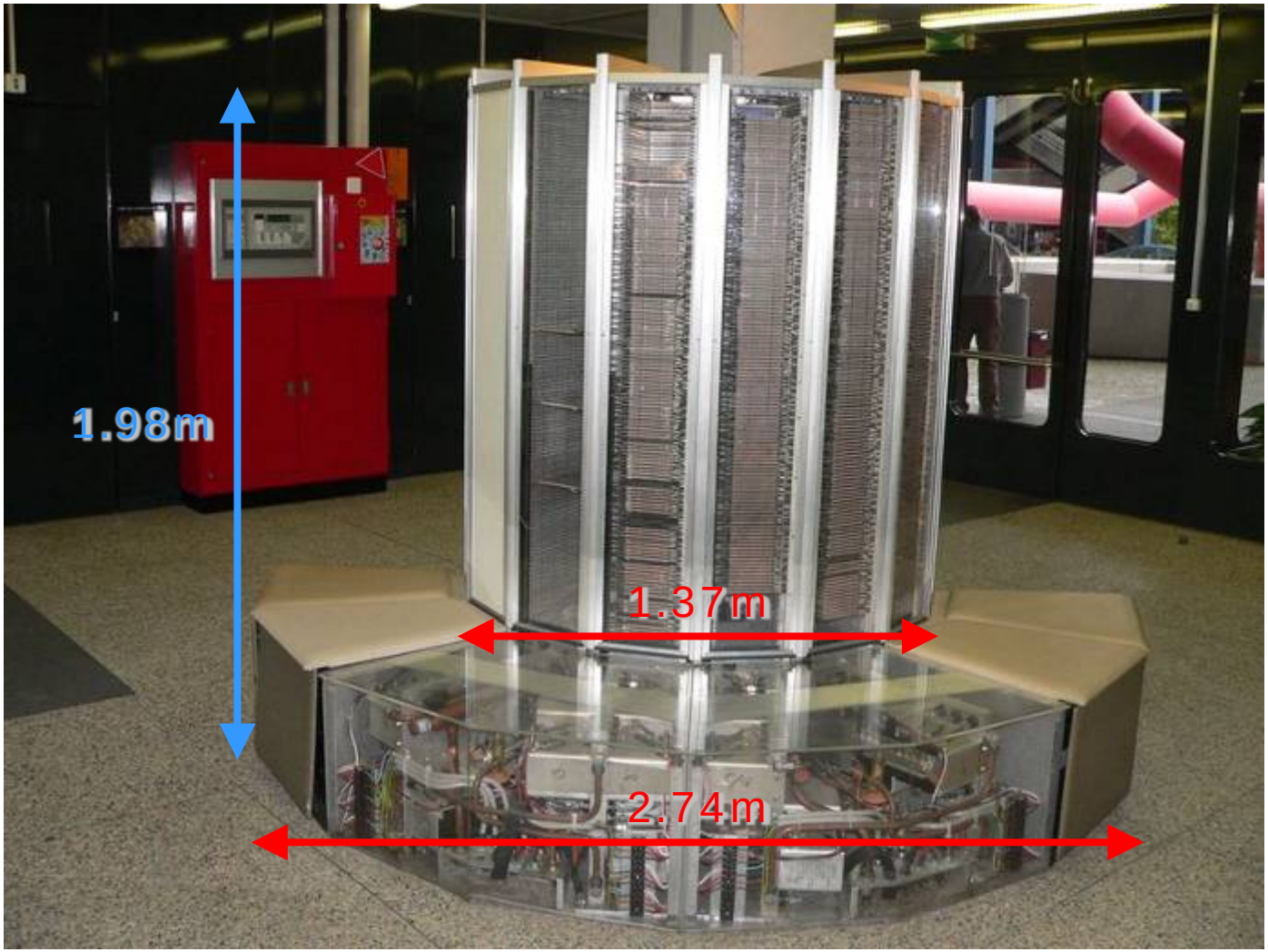
# スーパーにする方法：周波数↑の歴史



# 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) ≒ 50KFlops

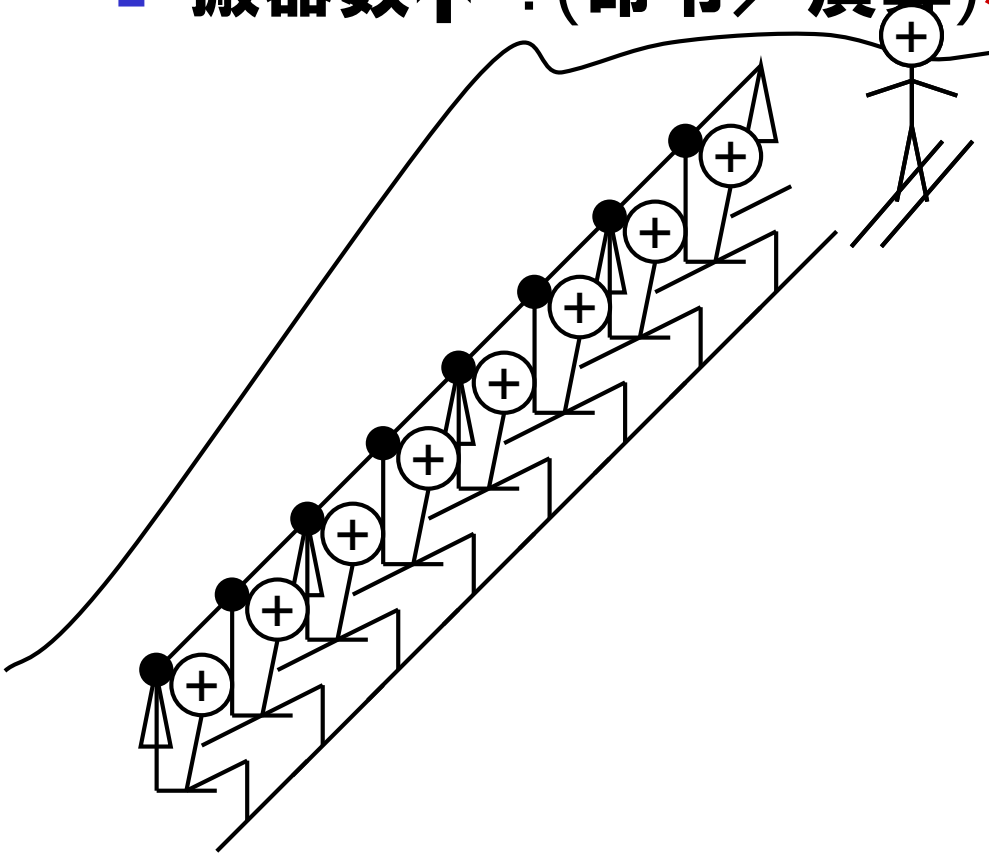
# そもそもの始まり：ベクトルマシン (2)



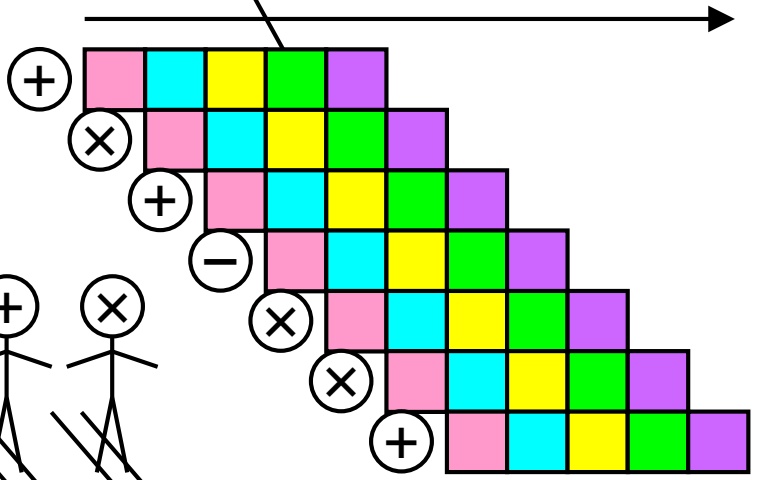

source: http://en.wikipedia.org/wiki/Image:Cray-1-p1010221.jpg

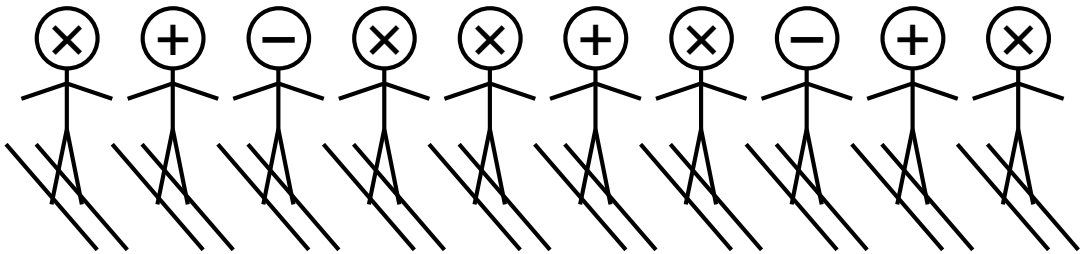
# 少し話を戻して
# スーパーにする方法：1970~

- 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



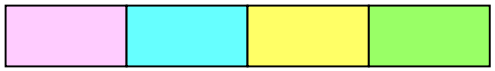


# スーパーにする方法：ベクトル計算の原理 (1)

- 大量数値データの同種演算を高速に行う方法

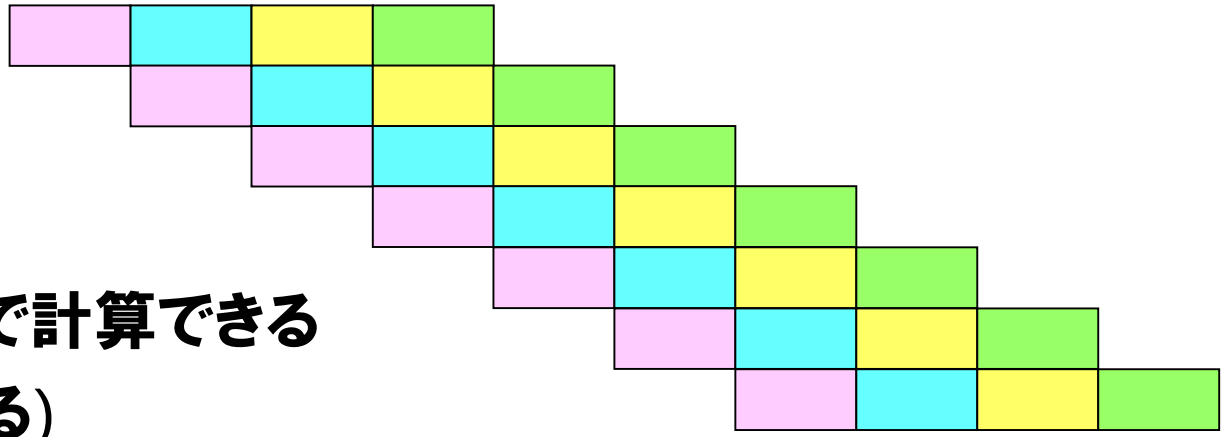
  例：$z_i = x_i \times y_i \quad (i = 1, 2, \ldots)$

  - 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

$z_i = x_i \times y_i$ 

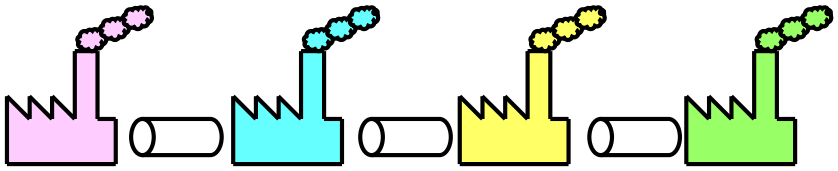
  - 多数の乗算を1小操作ずつ**ずらして**行う

$z_1 = x_1 \times y_1$
$z_2 = x_2 \times y_2$
$z_3 = x_3 \times y_3$
$z_4 = x_4 \times y_4$



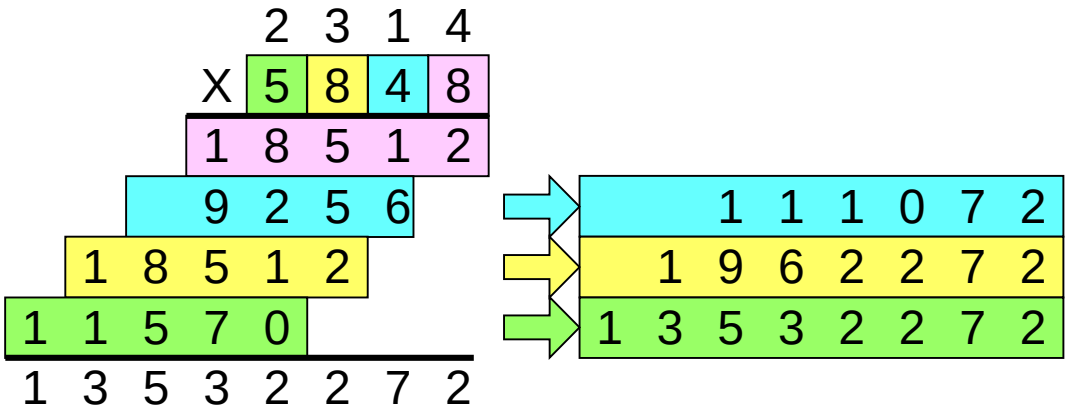
➔ **4倍の速度**で計算できる
(ように見える)

➔ (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)




2314x5848=13532272

2314x8 | +2314x4 | +2314x8 | +2314x5

7872x6752=53151744

7872x2 | +7872x5 | +7872x7 | +7872x6

1778x7142=12698476

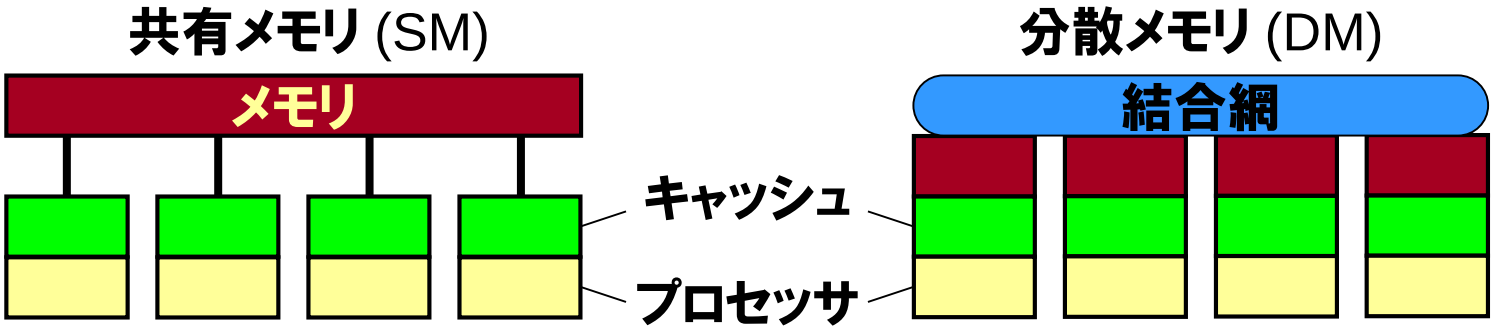
1778x2 | +1778x4 | +1778x1 | +1778x7

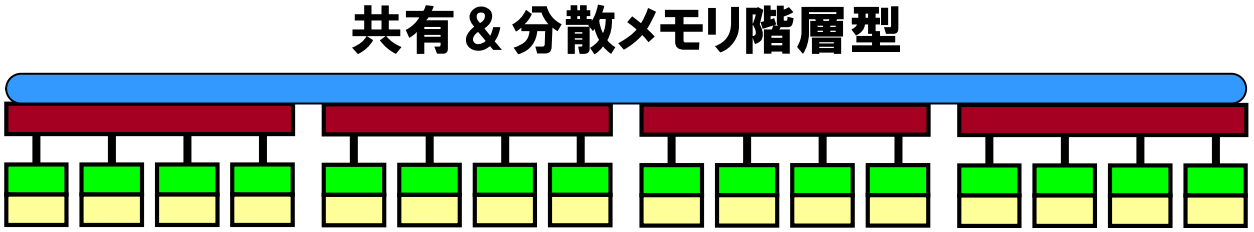
8485x1651=13843635

8385x1 | +8385x5 | +8385x6 | +8385x1

# 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

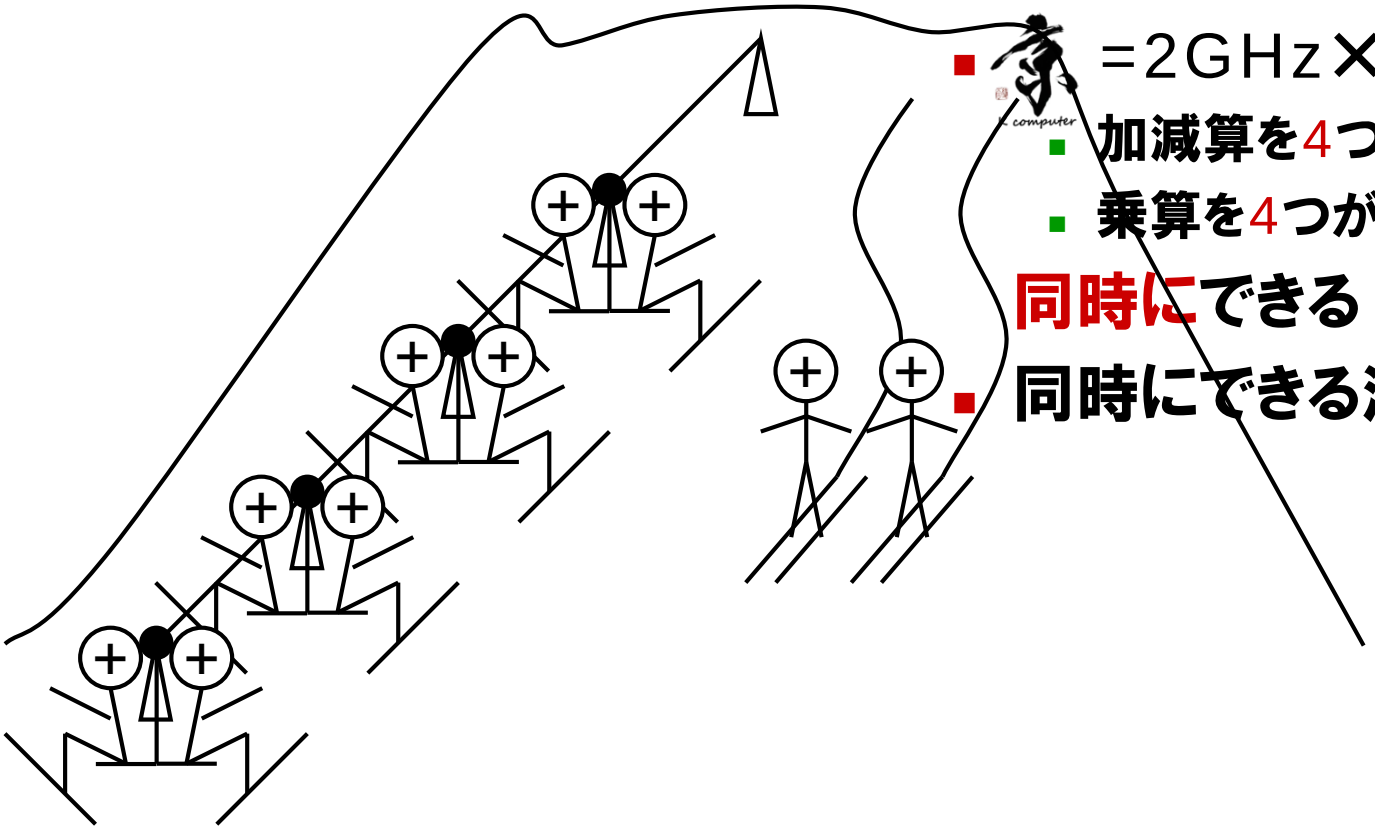
- 1980年代: スカラーマルチプロセッサ台頭
  - 多数のパソコン(のようなもの)の集合体
  - Sequent Balance : 20 x NS32016 ('84)
  - Intel iPSC/1: 128 x i80286 ('85)



# また話を戻して
# スーパーにする方法：1990~

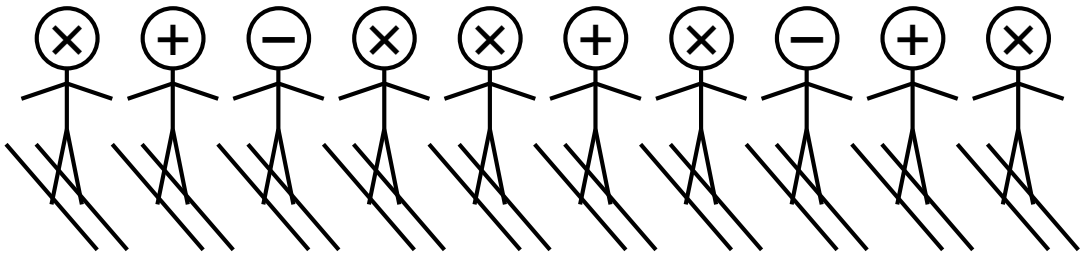
- 座席数↑≒スーパースカラー／SIMD



- =2GHz×8×8×88128
  - 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

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

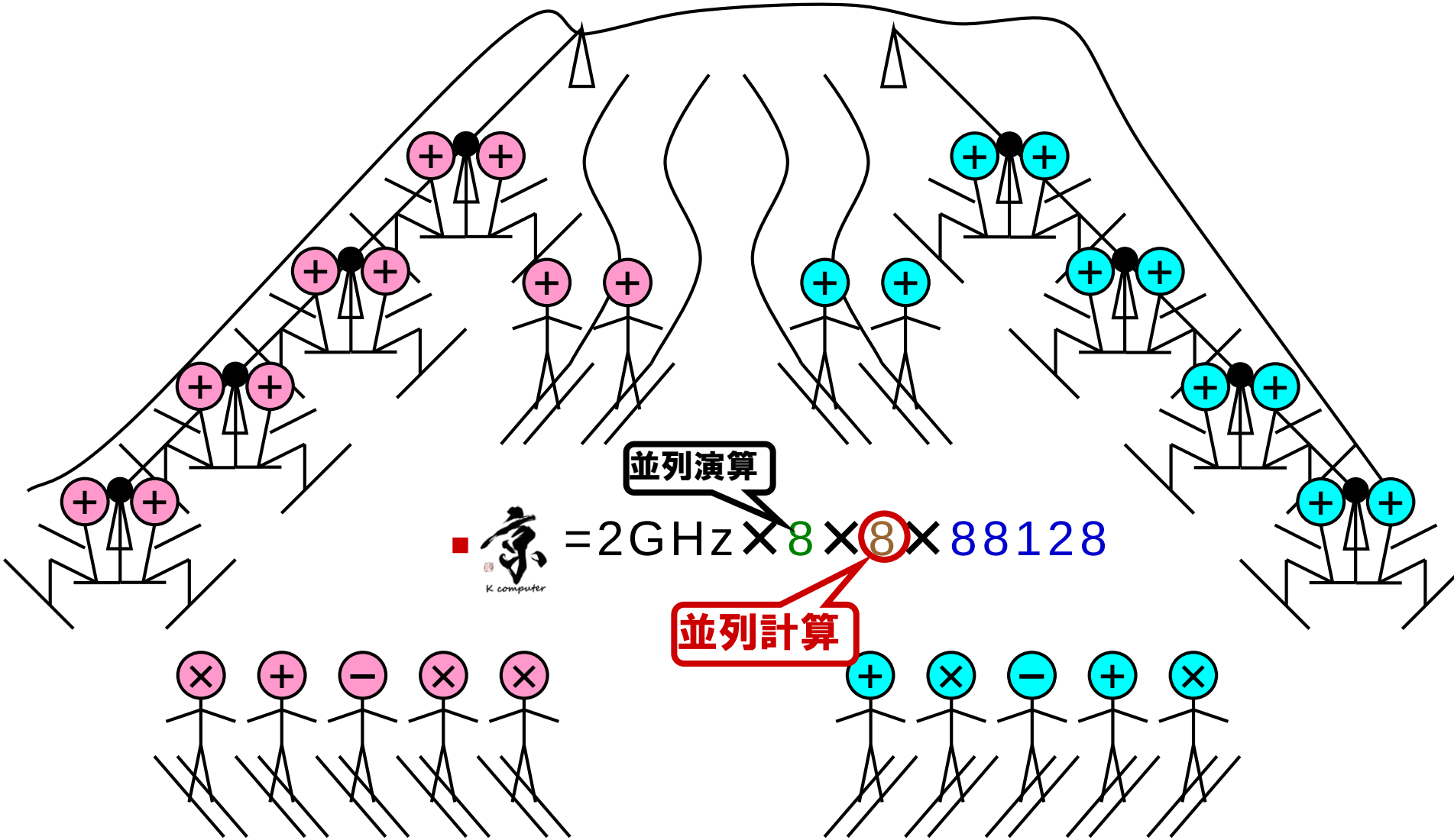
$$\left(-\frac{10}{8}+8\right)z=\frac{54}{8}z=-\frac{14}{8}+22=\frac{162}{8}\Rightarrow z=3$$

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

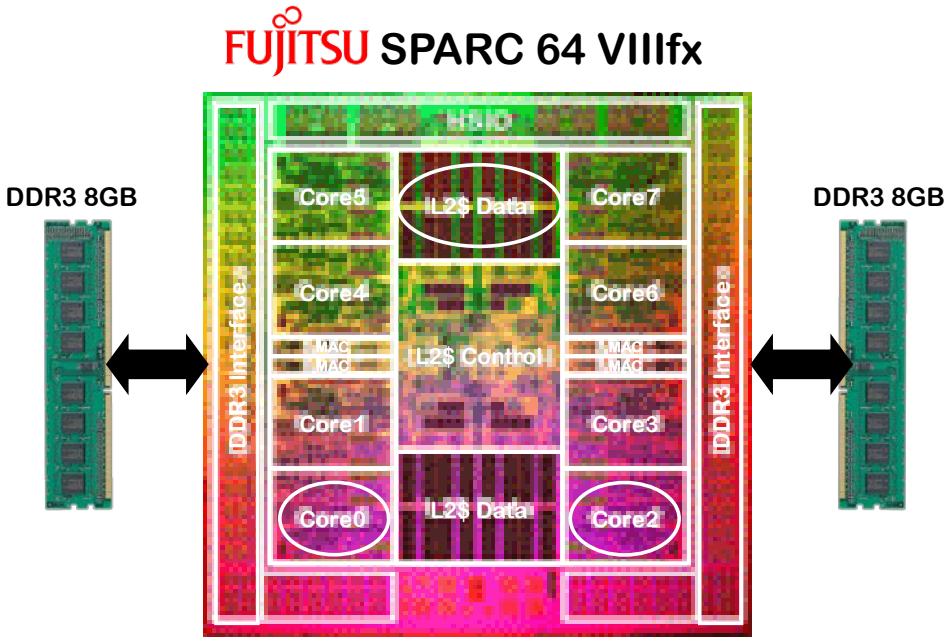
$$x=\frac{8}{2}+\frac{3\cdot 2}{2}-\frac{4\cdot 3}{2}=\frac{2}{2}=1$$

# スーパーにする方法:2000～ (1980~)

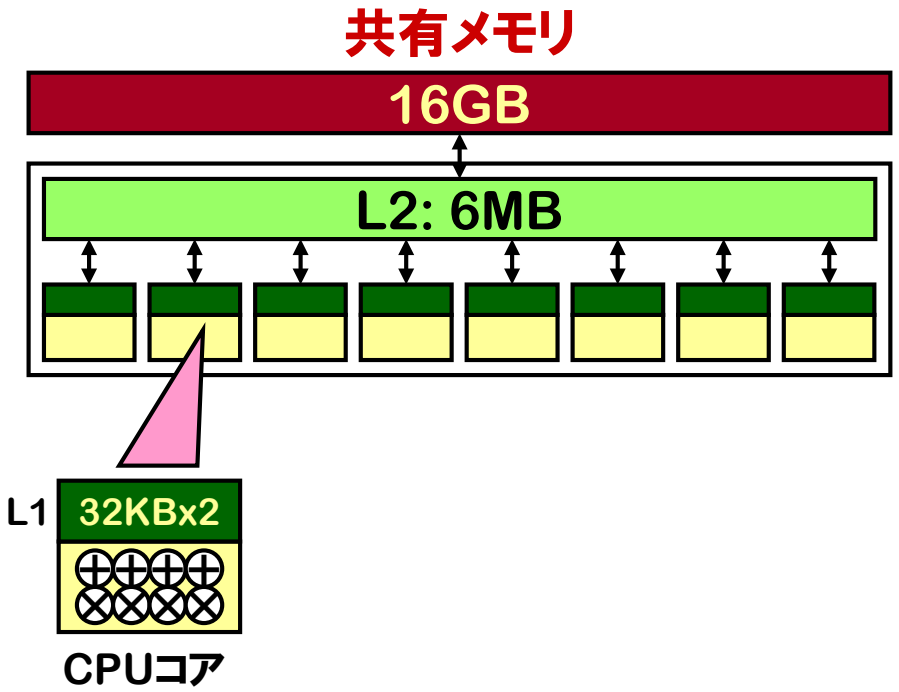
- リフト数↑≒マルチコア／共有メモリ並列マシン



# スーパーにする方法：京 のプロセッサ



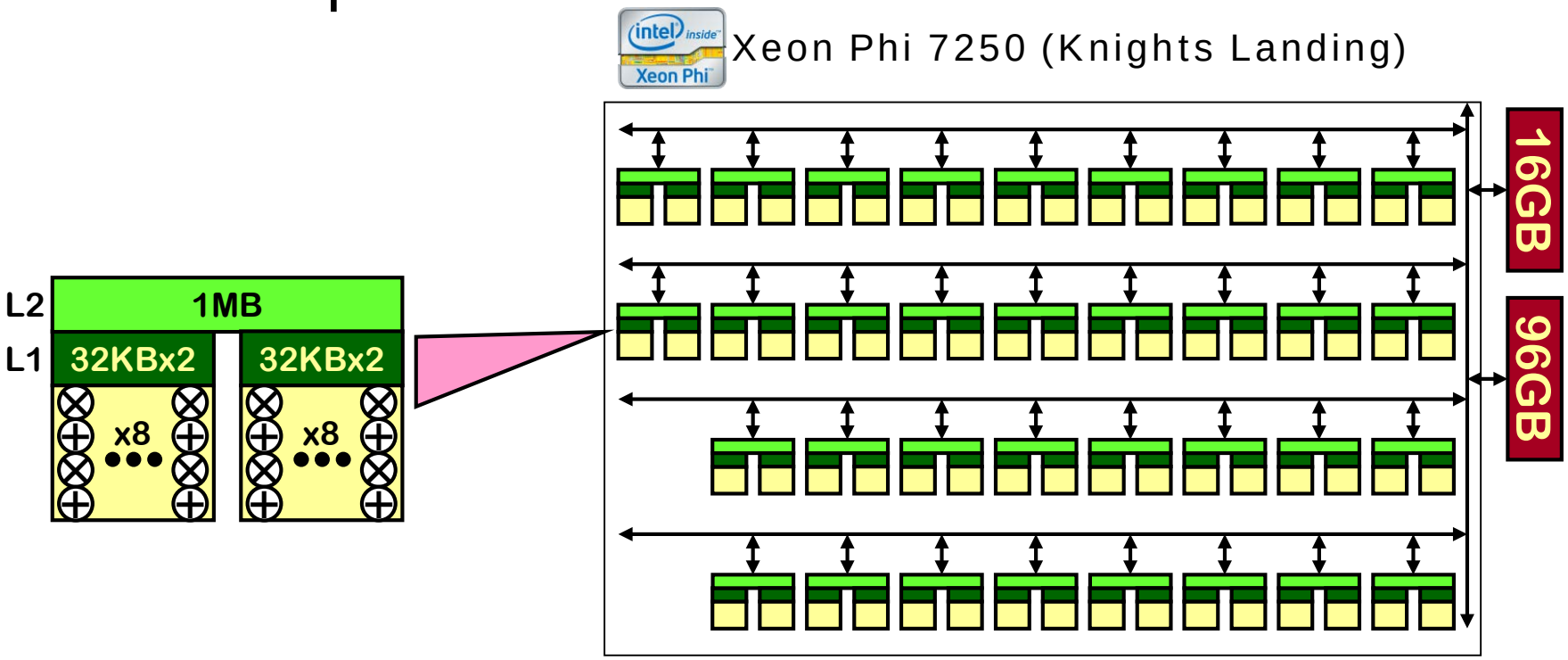

http://www.aics.riken.jp/jp/k/system.html



# 京大スパコンのプロセッサ (1/2)



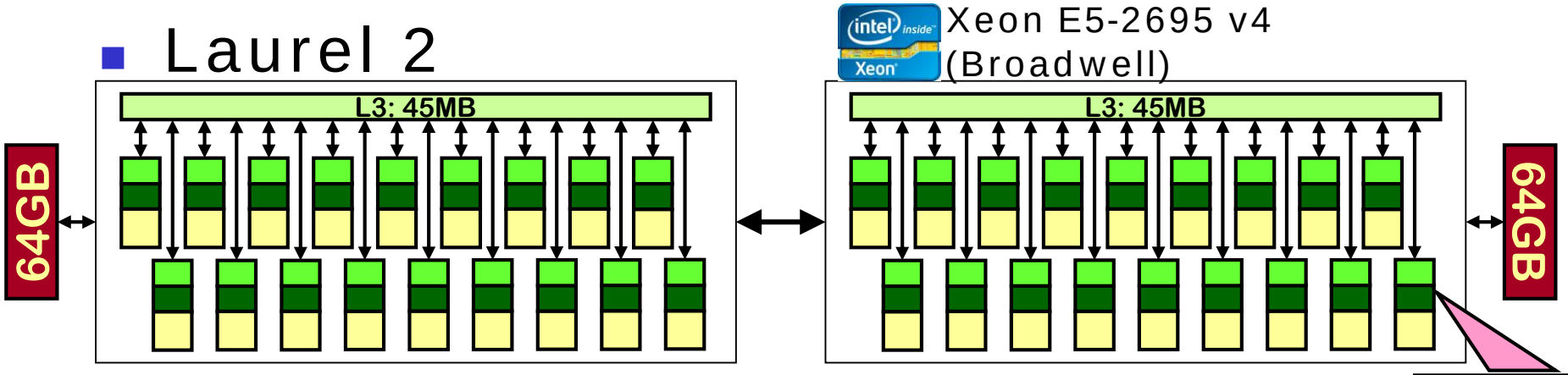
- Camphor 2



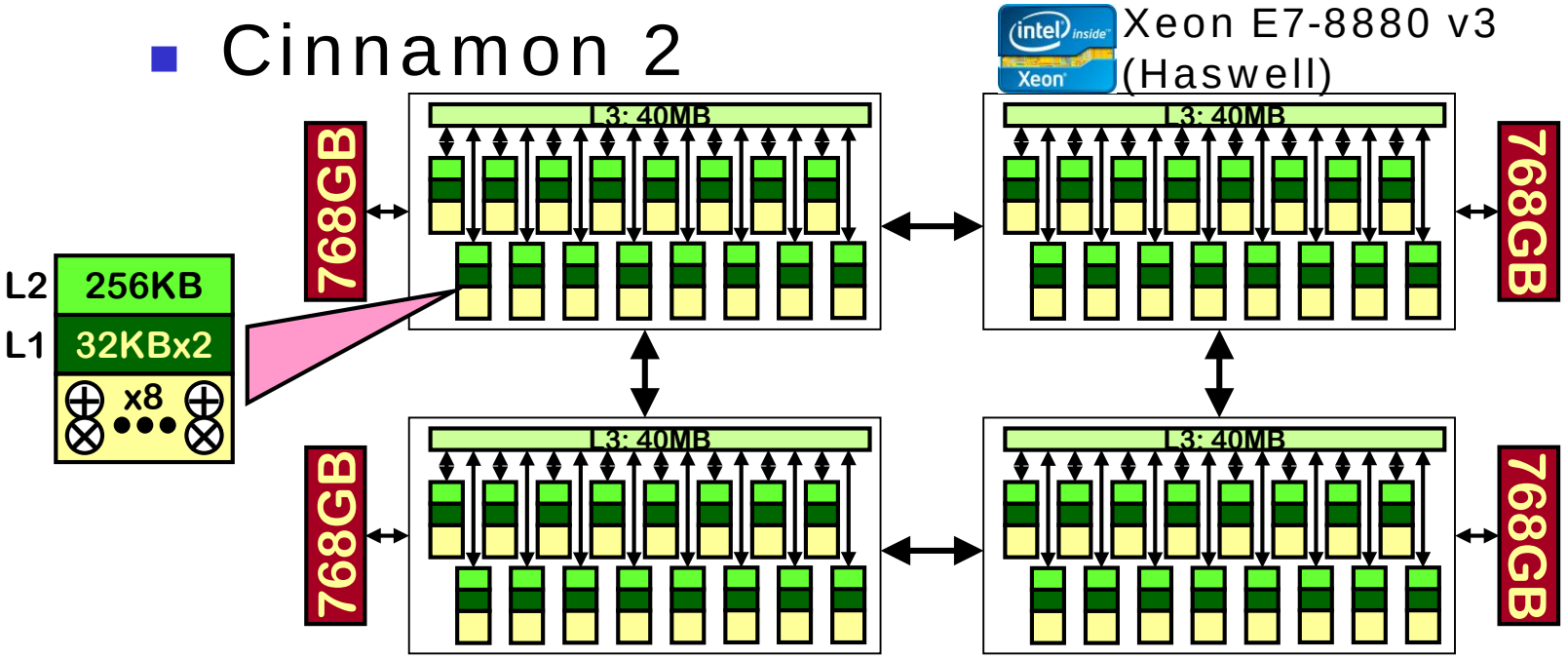
# 京大スパコンのプロセッサ (2/2)



- Laurel 2




- Cinnamon 2



# スーパーにする方法：連立方程式の並列計算

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + \cdots + a_{2n}x_n = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 + \cdots + a_{3n}x_n = b_3$$
$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + \cdots + a_{4n}x_n = b_4$$
$$\cdots$$
$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + a_{n4}x_4 + \cdots + a_{nn}x_n = b_n$$

1行目担当のコアが書いて

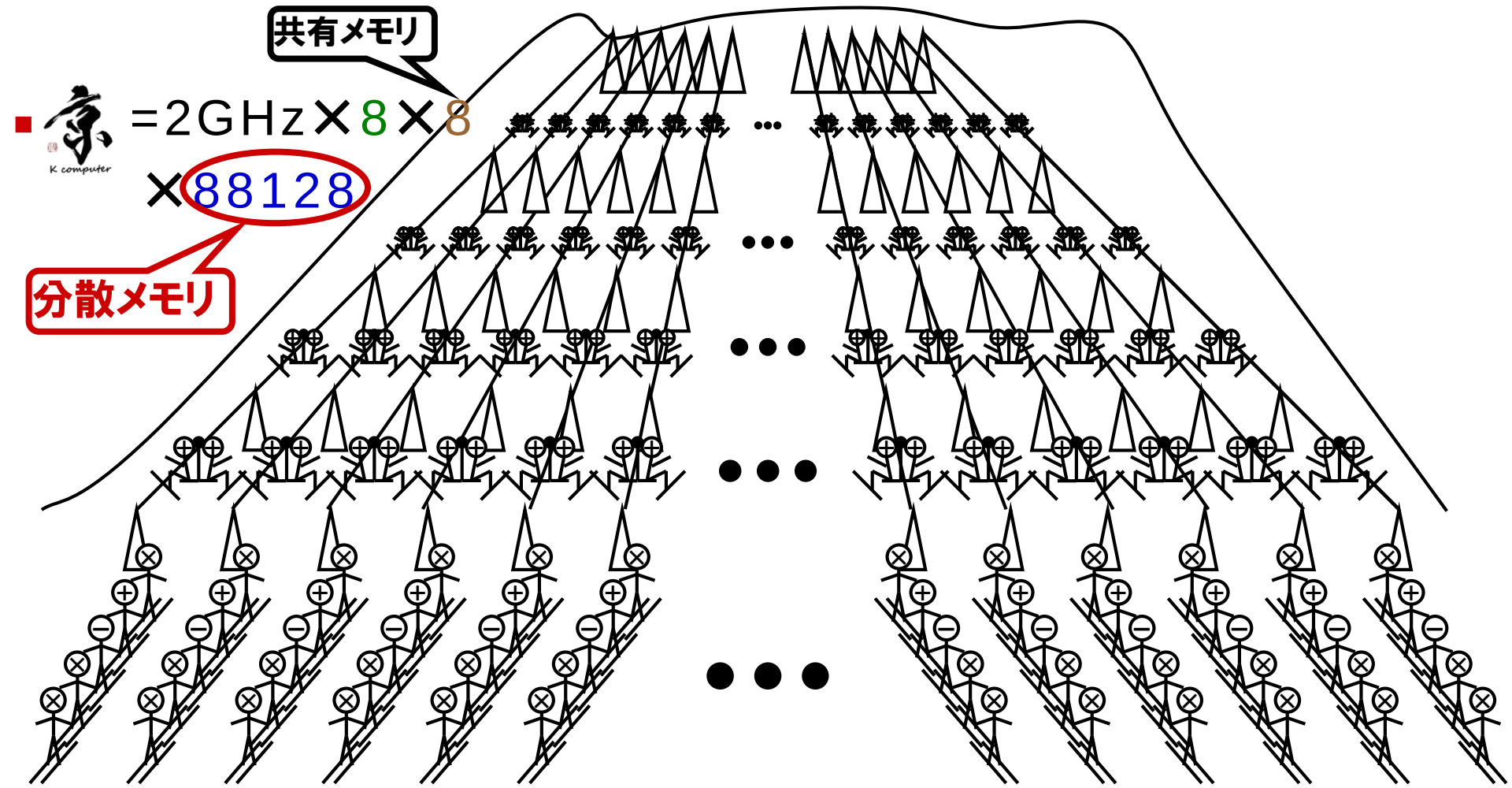
$$a'_{1j} = a_{1j} / a_{11}$$
$$a'_{ij} = a_{ij} / a_{i1} - a'_{1j}$$

i行目担当のコアが読む

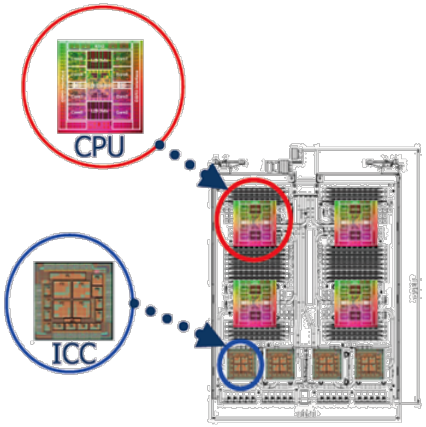
$$x_1 + a'_{12}x_2 + a'_{13}x_3 + a'_{14}x_4 + \cdots + a'_{1n}x_n = b'_1$$
$$x_1 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 + \cdots + a'_{2n}x_n = b'_2$$
$$x_1 + a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 + \cdots + a'_{3n}x_n = b'_3$$
$$x_1 + a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 + \cdots + a'_{4n}x_n = b'_4$$
$$\cdots$$
$$x_1 + a'_{n2}x_2 + a'_{n3}x_3 + a'_{n4}x_4 + \cdots + a'_{nn}x_n = b'_n$$

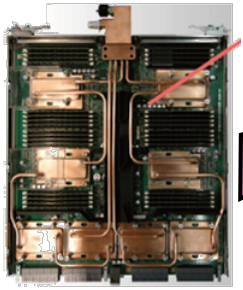
# スーパーにする方法:1980~

- リフト数↑↑↑↑↑↑↑↑↑↑↑↑≒超並列コンピュータ

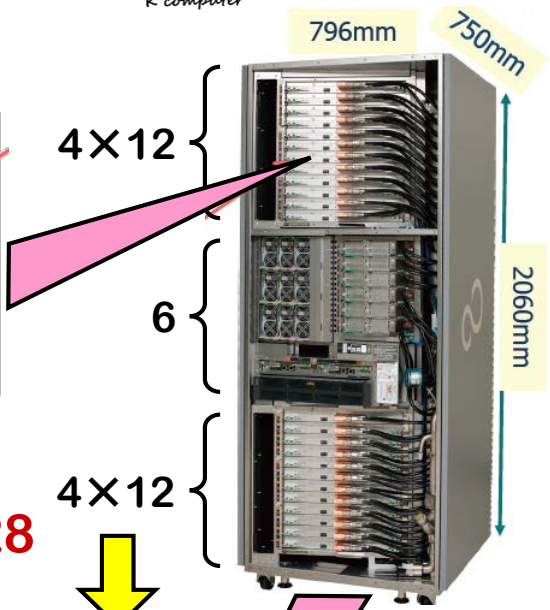


# スーパーにする方法：京 の全体像



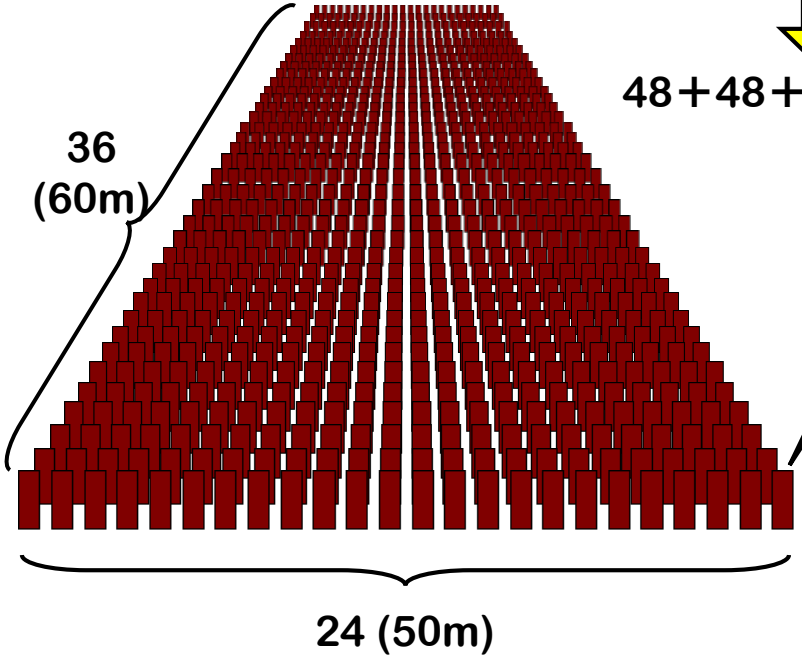

http://www.aics.riken.jp/jp/k/system.html





102×24×36＝102×864＝88,128

48＋48＋6＝102

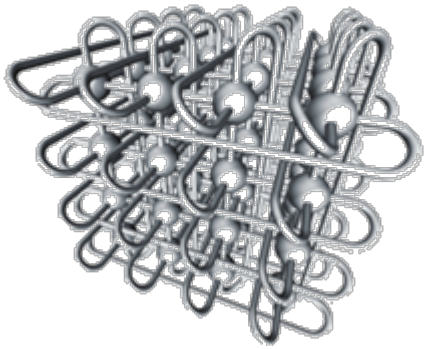



京計算機室
60m x 50m

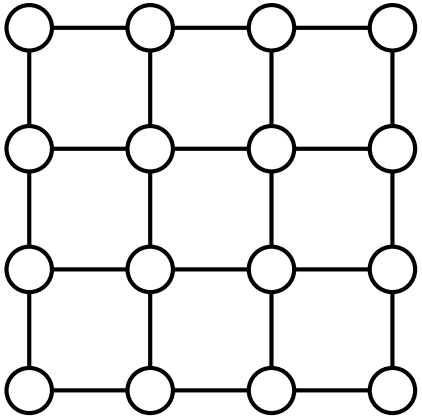
京大体育館
56m x 54m

# スーパーにする方法：京 の通信路 (1/2)

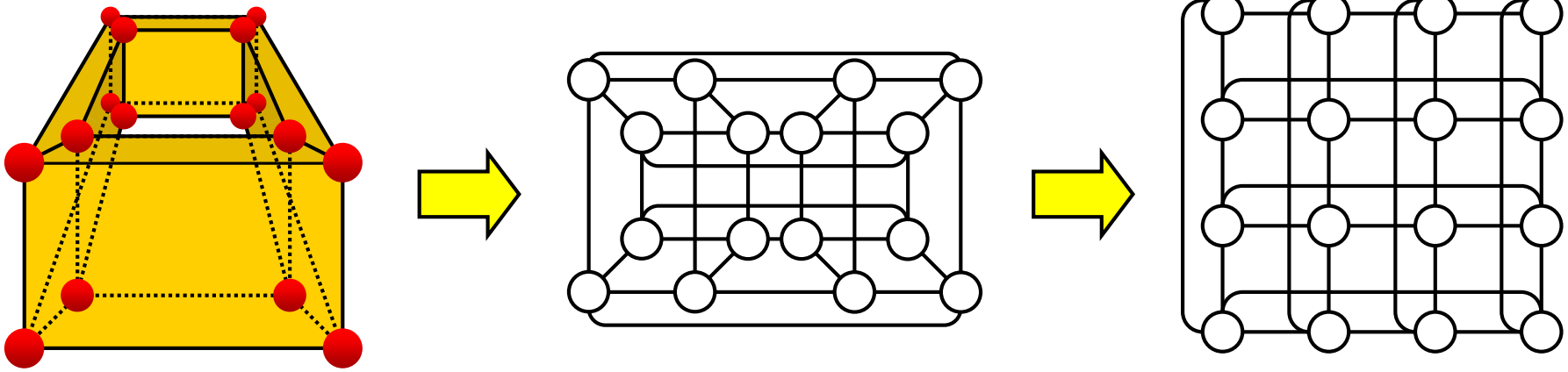
- 6次元メッシュ／トーラス結合網 Tofu
  - って意味不明～



- 2次元メッシュ

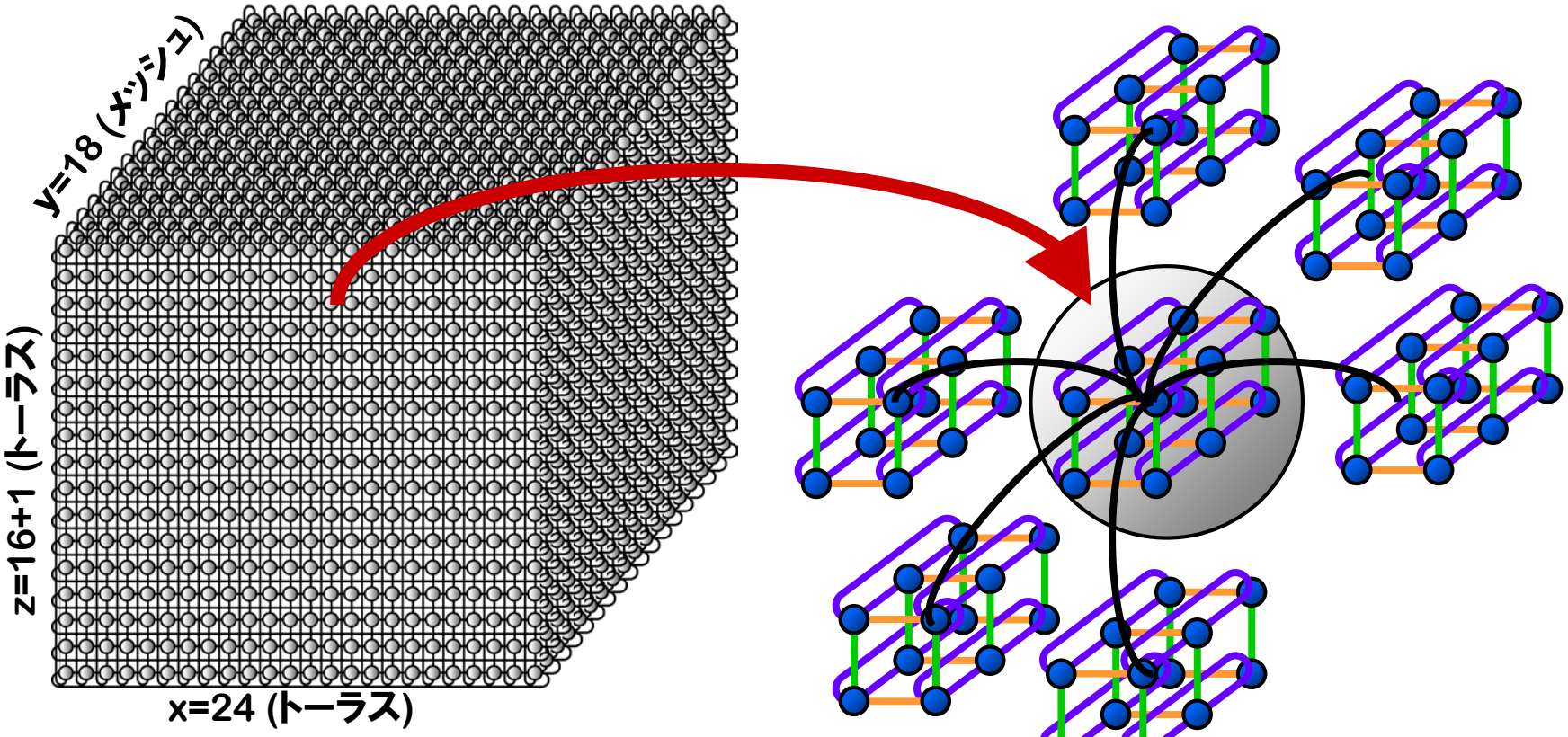


- 2次元トーラス (ドーナツの表面)



# スーパーにする方法：京 の通信路 (2/2)

- 6次元メッシュ／トーラス結合網 Tofu




24×18×(16+1)×2×2×3
＝88,128

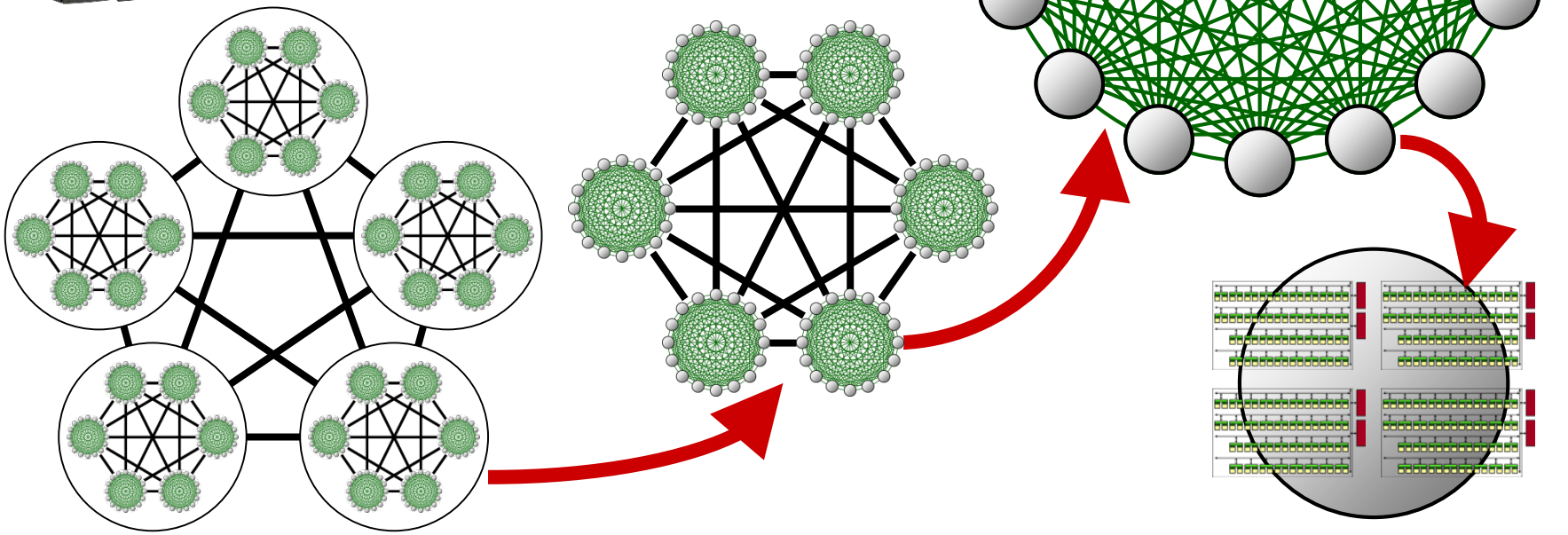
# 京大スパコンの全体像 (1/3)



- Camphor 2

CRAY XC40

4×(16−1)×6×5＝1800

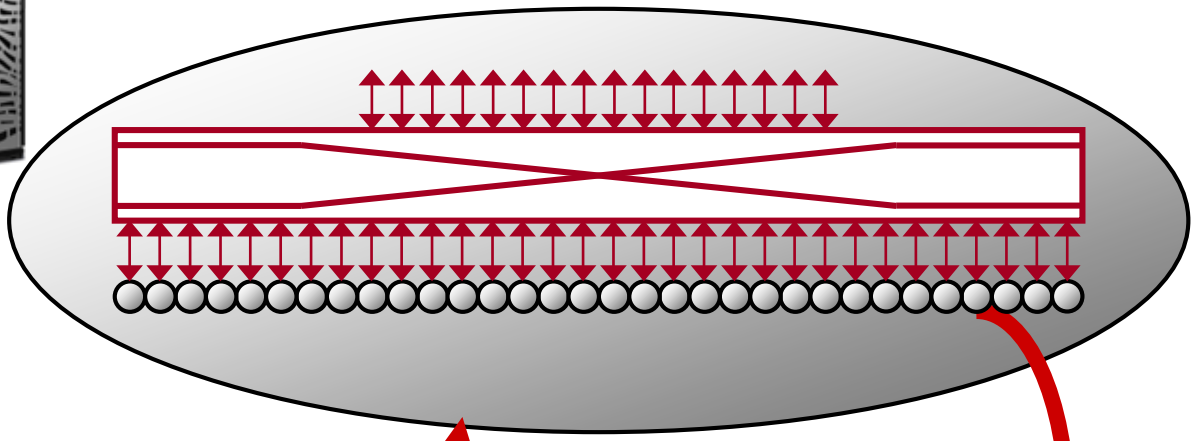
# 京大スパコンの全体像 (2/3)

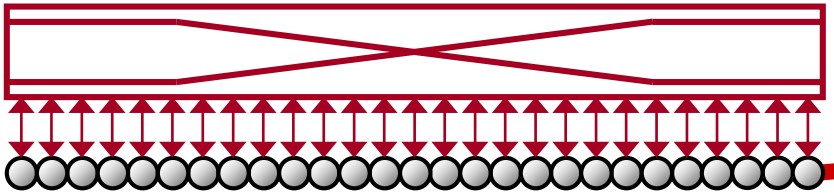


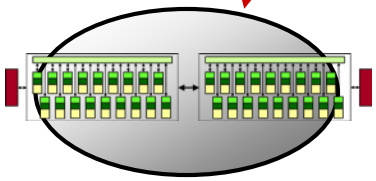
- Laurel 2

CRAY CS400 2820XT

32×27−14=850





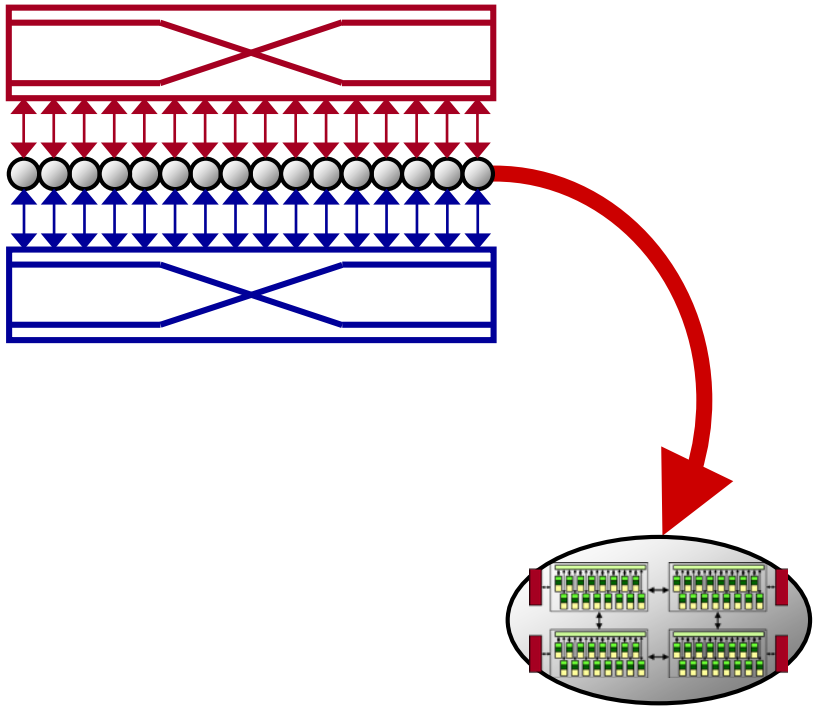


# 京大スパコンの全体像 (3/3)



- Cinnamon 2

CRAY CS400 4820X



# スーパーにする方法：連立方程式の並列計算

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + \cdots + a_{2n}x_n = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 + \cdots + a_{3n}x_n = b_3$$
$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 + \cdots + a_{4n}x_n = b_4$$
$$\cdots$$
$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + a_{n4}x_4 + \cdots + a_{nn}x_n = b_n$$

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)

$$x_1 + a'_{12}x_2 + a'_{13}x_3 + a'_{14}x_4 + \cdots + a'_{1n}x_n = b'_1$$
$$x_1 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 + \cdots + a'_{2n}x_n = b'_2$$
$$x_1 + a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 + \cdots + a'_{3n}x_n = b'_3$$
$$x_1 + a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 + \cdots + a'_{4n}x_n = b'_4$$
$$\cdots$$
$$x_1 + a'_{n2}x_2 + a'_{n3}x_3 + a'_{n4}x_4 + \cdots + a'_{nn}x_n = b'_n$$

# 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

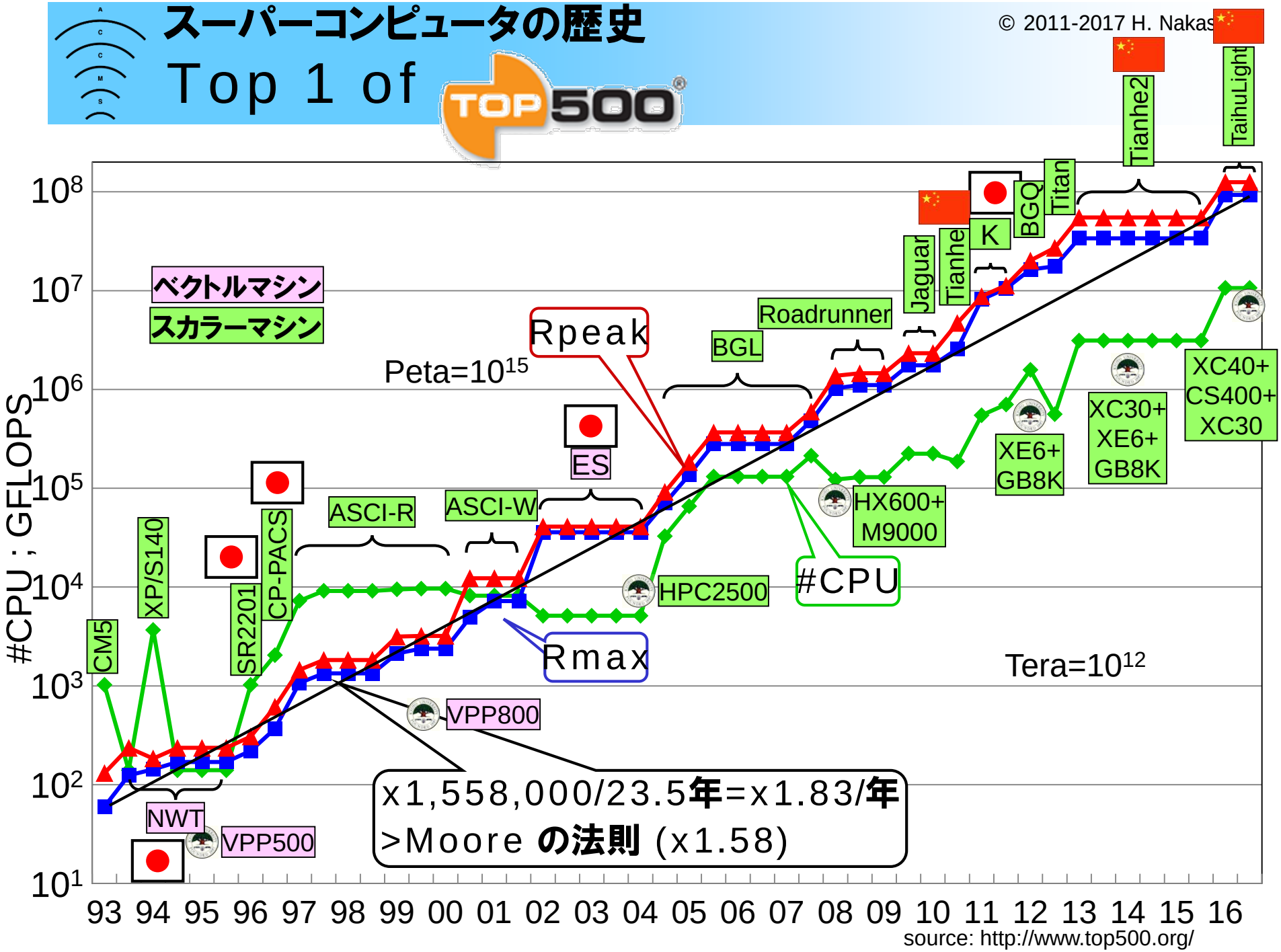
- 1990年代:**ベクトル並列** vs **スカラー並列**
  - TOP-10 of TOP500 @ 1993.6

| 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
# Top 1 of TOP500






source: http://www.top500.org/

# （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**

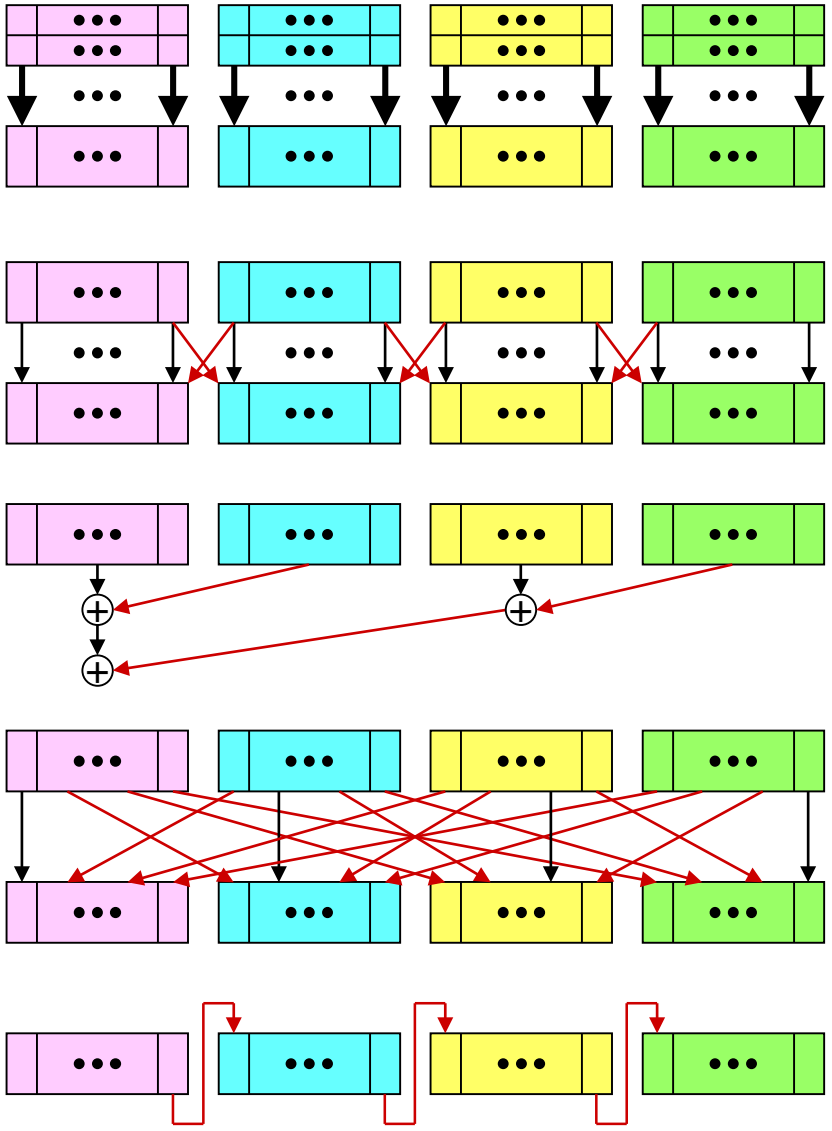
$$z = x_1 + x_2 + \cdots + x_n$$

- **何とかなる**

$$z_i = x_{f(i)} \text{ s.t. } z_1 \le z_2 \le \cdots \le z_n$$

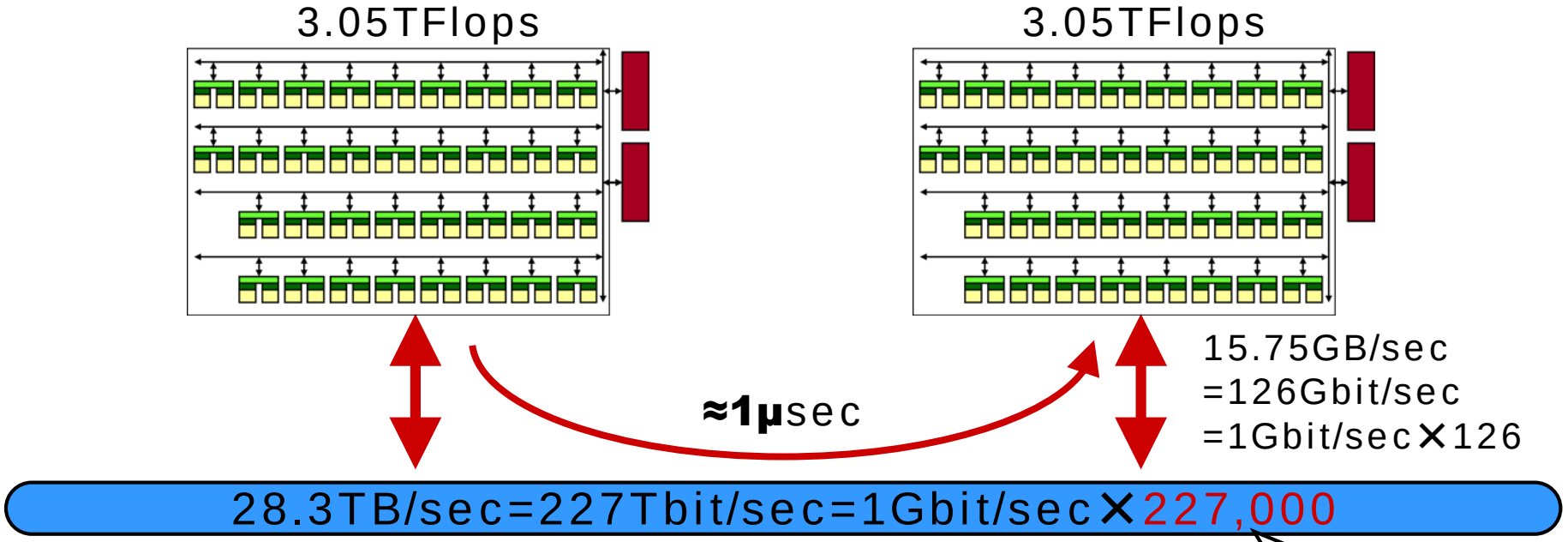
- **全然ダメ**

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



# 大量同種演算は何でも得意か? (2/2)

- **京大スパコン (Camphor 2) の通信速度**




- 1個の数値(8B)の通信時間＝1μsec
  ＝3,050,000個分の演算時間
- 10億個の数値(8GB)の通信時間＝0.51秒
  ＝15,492億個分の演算時間

# 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。