スーパーコンピュータは何が「スーパー」なのか？

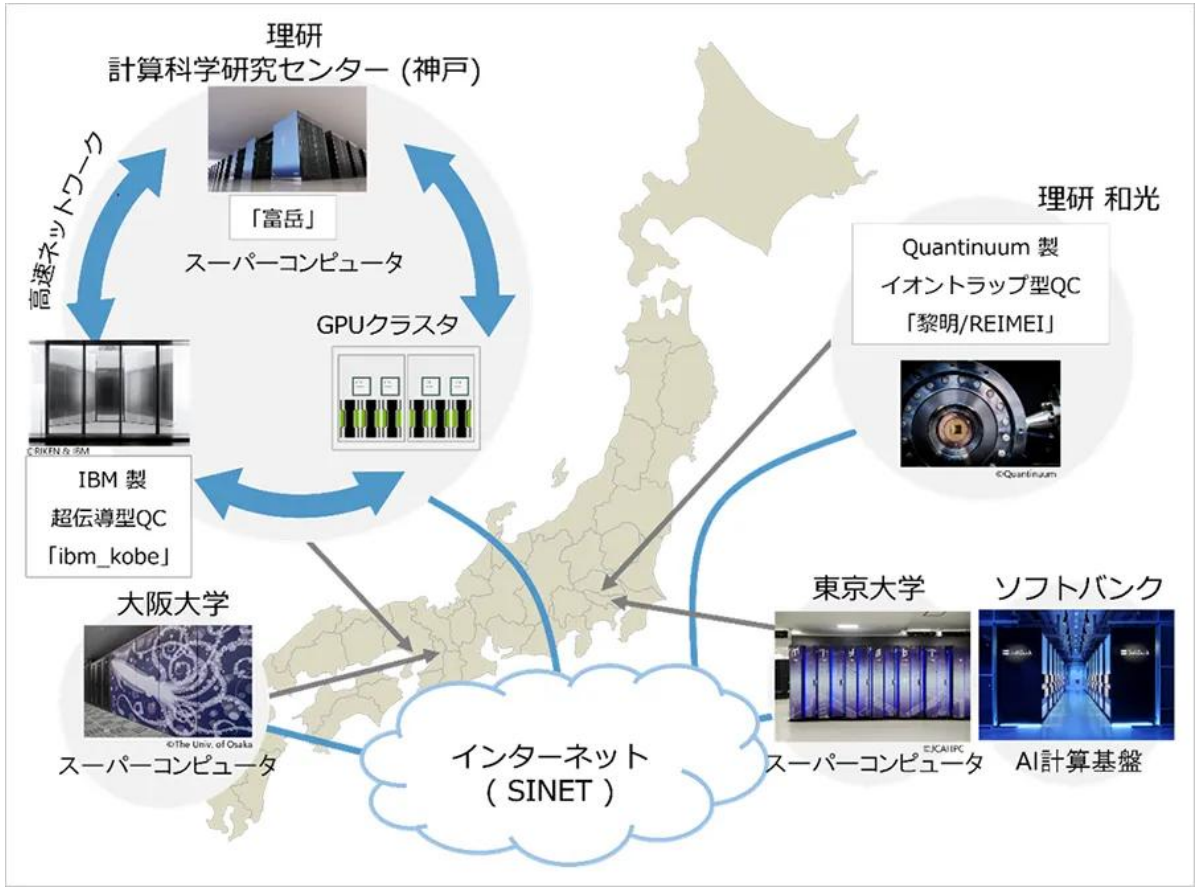
追手門学院大学
理工学部　数理・データサイエンス学科
幸谷　智紀（こうや　とものり）
https://kd-tklab.na-inet.jp/

概要

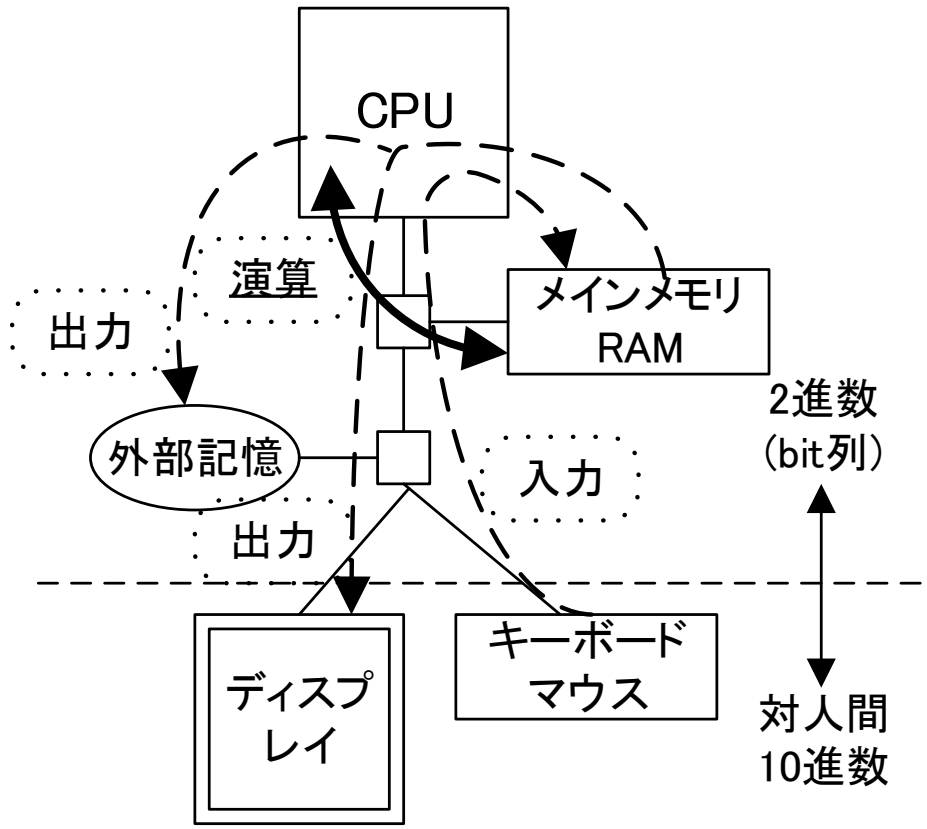
- コンピュータの基本
  - 効率的な「アルゴリズム」の重要性
  - 事例１）自然数の乗算
  - 事例２）連立一次方程式の直接解法
- 「並列化」による高速化
  - 事例３）自然数の乗算の並列化
  - 事例４）連立一次方程式の直接解法の並列化
  - スーパーコンピュータの中身
  - Top500とLinpackテスト
- まとめ

AIの主力は「スーパーコンピュータ」

- 「ソフトバンクのAI計算基盤と理化学研究所の量子コンピュータの接続を開始」
- https://www.softbank.jp/corp/news/press/sbkk/2025/20250929_01/



コンピュータの基本(1/3):ハードウェアの概要

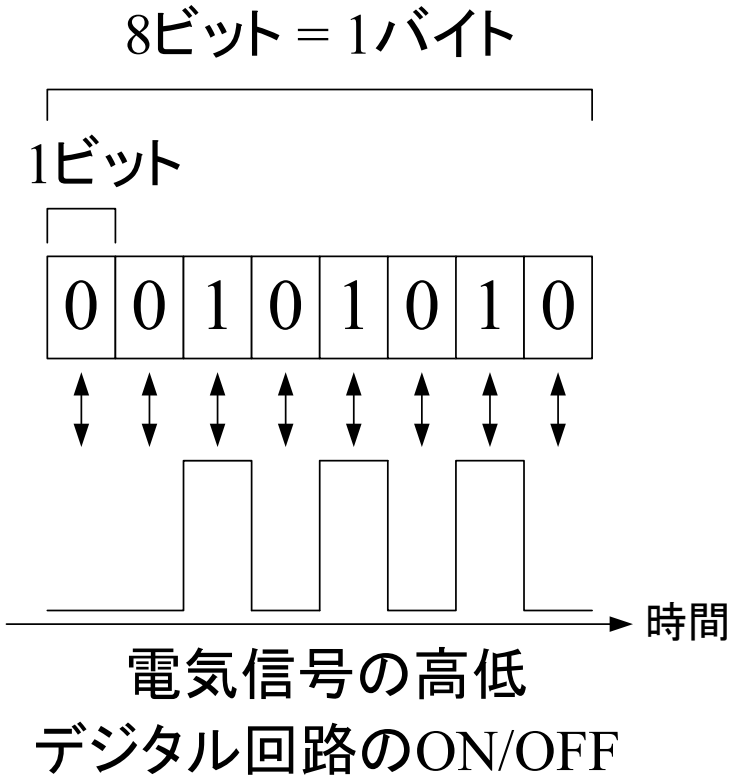


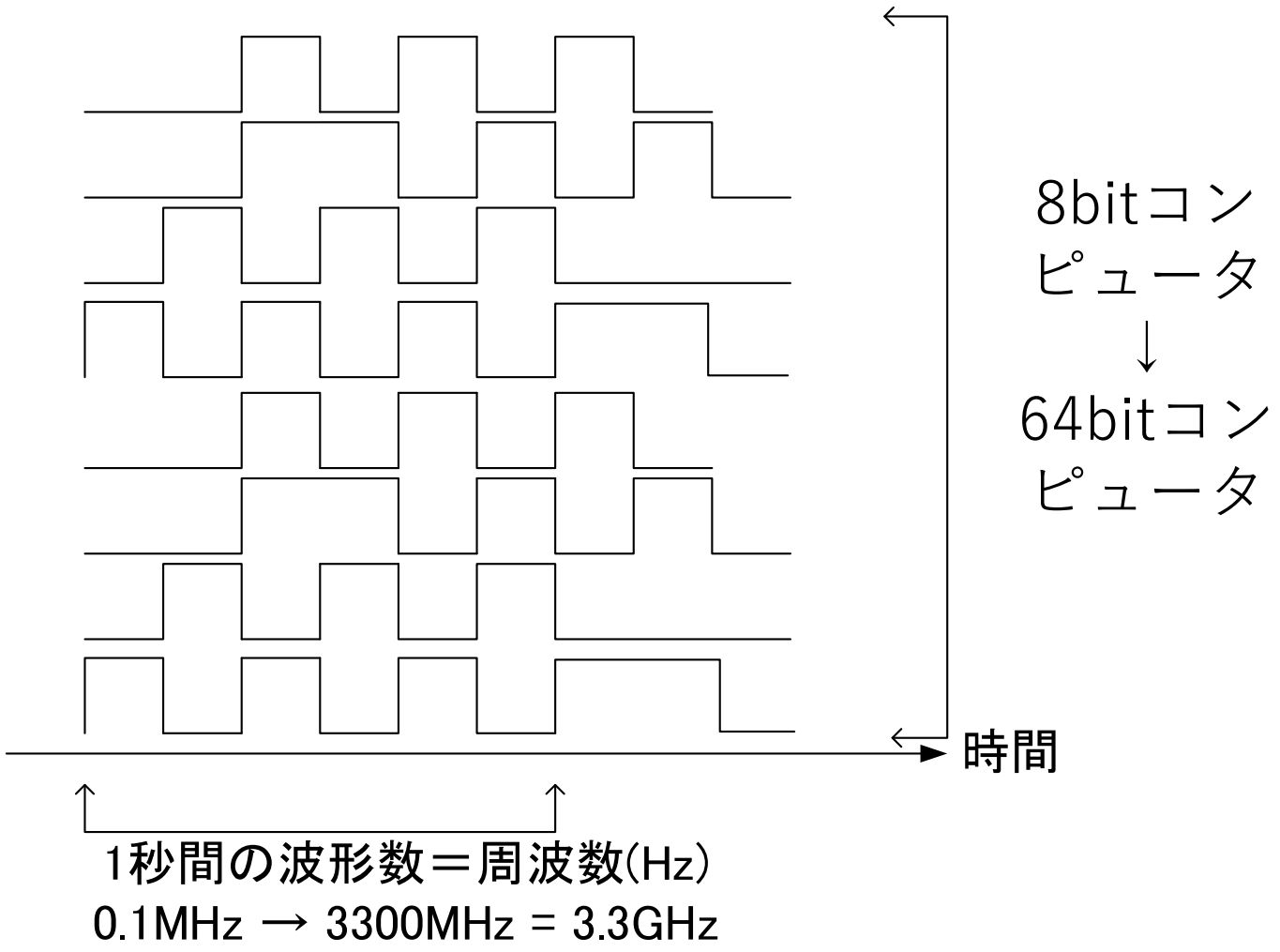
☜ハードウェア(HW)の概念図

- データの入力・出力（表示・保存）
- 処理されるデータは２進数として表現され，メインメモリに蓄えられる
- メインメモリに置かれたデータはCPUから読み込まれ・処理され・結果を書き戻される

→処理手順（アルゴリズム）をプログラムで指示しなければコンピュータは動かない

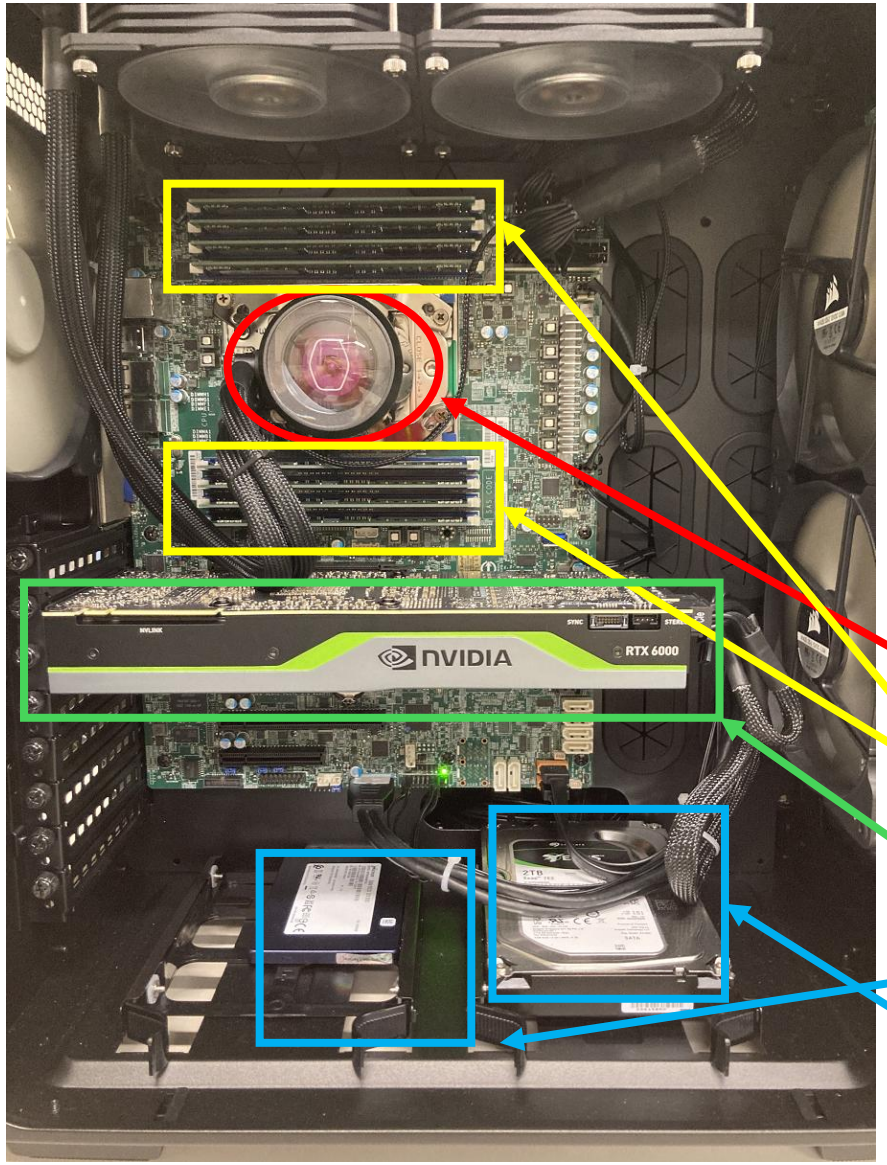
コンピュータの基本(3/3):高速化の基本原理





動作周波数を増やす
→電力量が増大（困難）
→並列化する

コンピュータの基本(2/3):HWの例



2020～2022年度　科研費で購入

「マルチコアCPU向けに最適化された高速任意精度線形計算ライブラリの開発」

CPU: AMD EPYC 7402P 24コア, 2.8GHz, 180W
RAM: 128GB
GPU: NVIDIA Quadro RTX6000 24GB
SSD: 480GB
HDD: 2TB

自然数の乗算：筆算

1234×8765

乗算4回×4= 16回

$= 1234 \times 5$

$\rightarrow (1 \times 5) \times 10^3 + (2 \times 5) \times 10^2 + (3 \times 5) \times 10 + 4 \times 5 = 6170$

$+ (1234 \times 6) \times 10$

$\rightarrow (1 \times 6) \times 10^4 + (2 \times 6) \times 10^3 + (3 \times 6) \times 10^2 + (4 \times 6) \times 10 = 74040$

$+ (1234 \times 7) \times 10^2$

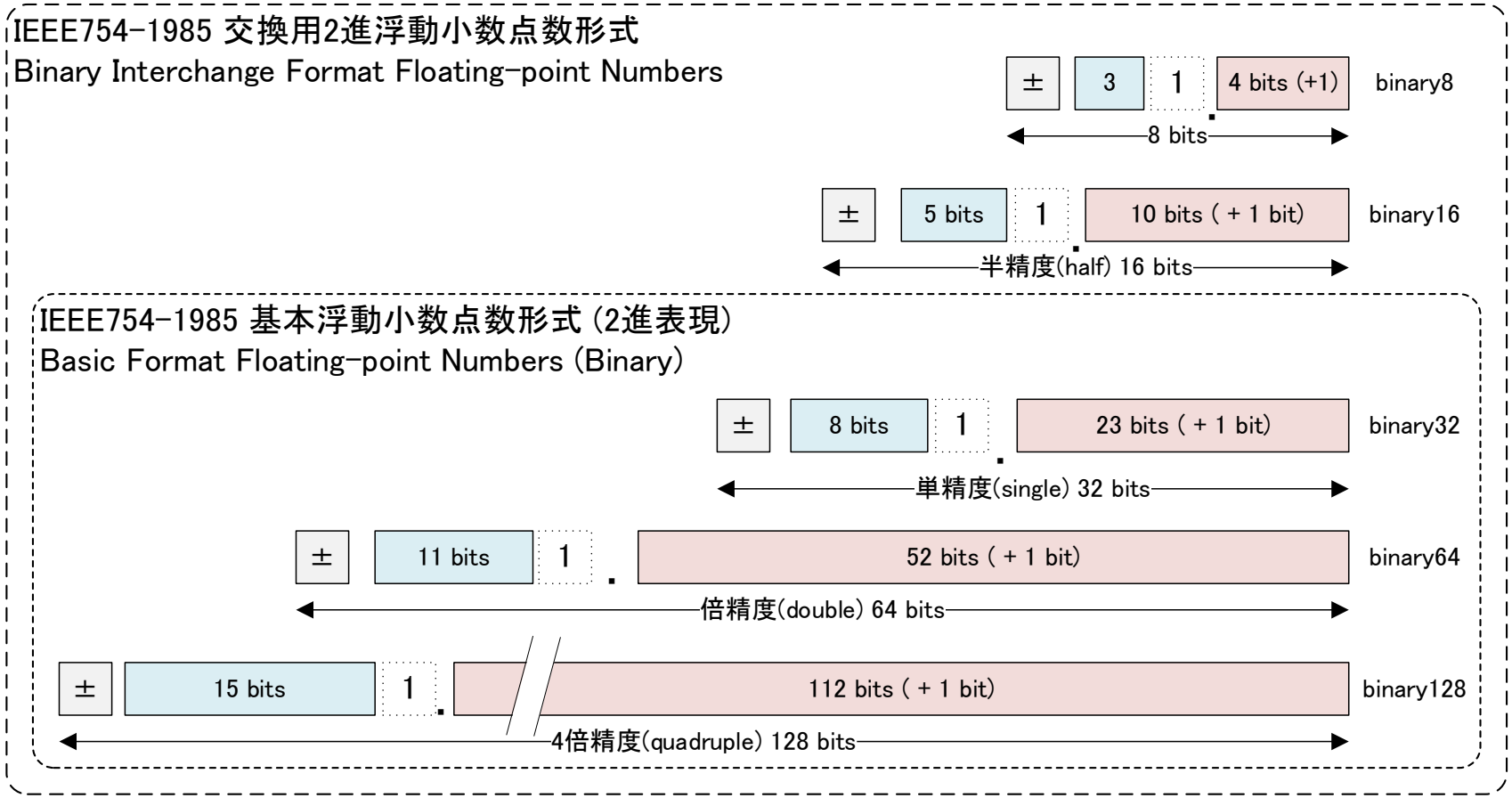
$\rightarrow (1 \times 7) \times 10^5 + (2 \times 7) \times 10^4 + (3 \times 7) \times 10^3 + (4 \times 7) \times 10^2 = 863800$

$+ (1234 \times 8) \times 10^3$

$\rightarrow (1 \times 8) \times 10^6 + (2 \times 8) \times 10^5 + (3 \times 8) \times 10^4 + (4 \times 8) \times 10^3 = 9872000$

$= 10816010$

浮動小数点数形式



ハードウェアサポートのある標準浮動小数点数

- IEEE754 binary16(半精度), binary32(単精度), binary64(倍精度)
- AIではbinary8等も登場

5. 事例３：連立一次方程式を解く

$$\begin{cases} 3x_1 - x_2 = -1 & \cdots ① \\ -6x_1 + 3x_2 = 2 & \cdots ② \end{cases}$$

①×2＋②

$$\begin{array}{rl} & 6x_1 - 2x_2 = -2 \\ +) & -6x_1 + 3x_2 = 2 \\ \hline & x_2 = 0 \end{array}$$

$x_2 = 0$を①に代入

$$3x_1 = -1$$

$$x_1 = -\frac{1}{3}$$

(答) $\begin{cases} x_1 = -\frac{1}{3} \\ x_2 = 0 \end{cases}$

- 左辺の未知数が１次式で記述できる複数式の方程式を連立「一次」方程式と呼ぶ
- 実用上重要な方程式
  - 微分方程式に基づくシミュレーション
  - 建築物の構造解析

  等々・・・
- 未知数は大量にあるケースが多い

→短い計算時間で求めるにはどうすればよいか？

LU分解と前進・後退代入(2×2の場合)

係数行列 A $\begin{bmatrix} 3 & -1 \\ -6 & 3 \end{bmatrix}$ $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 解$\mathbf{x}$ $= \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ 定数ベクトル$\mathbf{b}$

1. LU分解

$$\begin{bmatrix} 3 & -1 \\ -6/3 & 3-(-6/3)\times(-1) \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix}$$

$$\rightarrow L = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}, U = \begin{bmatrix} 3 & -1 \\ 0 & 1 \end{bmatrix}$$

2. 前進代入

$$L\mathbf{y} = \mathbf{b} \Leftrightarrow \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$\Leftrightarrow \begin{cases} y_1 = -1 \\ -2y_1 + y_2 = 2 \end{cases}$$

$$\rightarrow \begin{cases} y_1 = -1 \\ y_2 = 2 + 2y_1 = 0 \end{cases}$$

3. 後退代入

$$U\mathbf{x} = \mathbf{y}$$

$$\Leftrightarrow \begin{bmatrix} 3 & -1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

$$\Leftrightarrow \begin{cases} 3x_1 - x_2 = -1 \\ x_2 = 0 \end{cases}$$

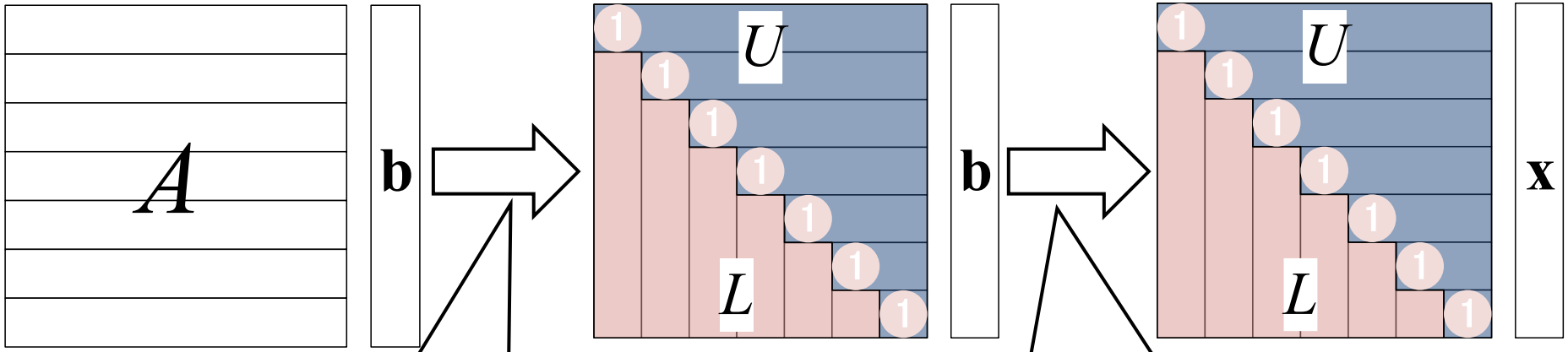
$$\rightarrow \begin{cases} x_1 = \dfrac{-1 + x_2}{3} = -\dfrac{1}{3} \\ x_2 = 0 \end{cases}$$

(答) $\begin{cases} x_1 = -\dfrac{1}{3} \\ x_2 = 0 \end{cases}$

規模の大きなLU分解と前進・後退代入

$$A\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

係数行列 A　　解$\mathbf{x}$　　定数ベクトル$\mathbf{b}$



LU分解
n^3に比例した計算量

前進・後退代入
n^2に比例した計算量

CPUの内部処理の並列化

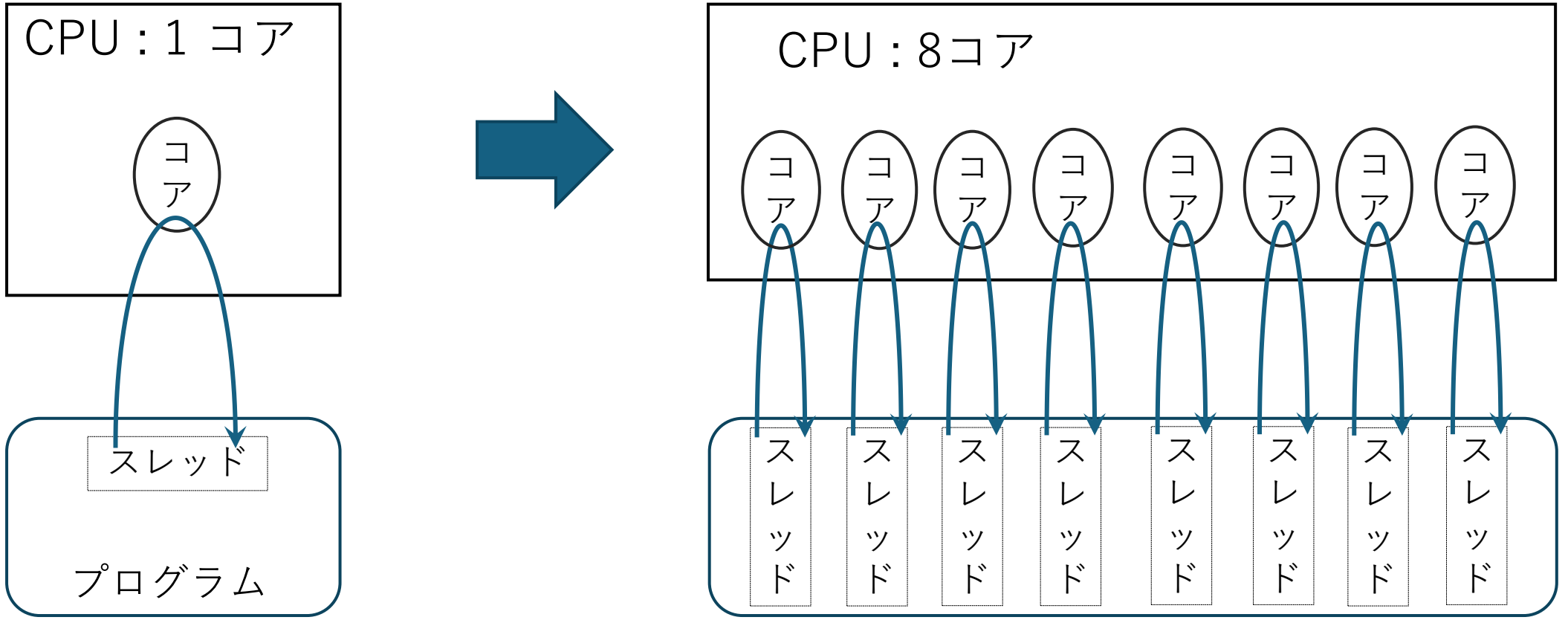
- 機械語：一命令(Instruction)でレジスタからのデータを読み取って処理
- SIMD (Simultaneous Instruction Multiple Data): 複数のデータをまとめて処理する機械語の一命令→内部処理の並列化
- 機械語のまとまり：ISA (Instruction Set Architecture)
- ISAはCPUの種類によって分かれる
  - x86アーキテクチャ: IntelのCoreシリーズ，AMDのRyzenシリーズなど
  →SIMD: AVX(128bit), AVX2(256bit), AVX-512(512bit)
  - Armアーキテクチャ: AppleのMシリーズ，QualcommのSnapdragonシリーズ，Raspberry Pi等
  →SIMD: Arm Neon(128bit), SVE2(256～512bit)

CPUマーケット状況

用途分野	x86 系シェア	Arm 系シェア	備考／変化傾向
PC／ノート	**約 80〜90 %**	**約 10〜20 %**	ノート市場では、Arm（特に Apple の M シリーズや Windows-on-Arm の採用）が徐々にシェアを拡大中。（Tom's Hardware）
サーバ／データセンター	**約 85〜95 %（従来） → 最近は若干後退**	**数 % → 最近では 10〜15 %程度**	Arm 系サーバ向けCPUの採用がハイパースケーラー企業（クラウド事業者など）で進んでおり、Arm 側も “データセンターでのシェア 50 % を目指す” という宣言も出している。（Reuters）
モバイル／スマートフォン／タブレット	**極めて小さい（ほぼゼロに近い）**	**圧倒的多数（90-95 % 以上）**	モバイル分野は Arm が事実上支配。x86 によるモバイル用途はほぼ限定的。（ウィキペディア）
組込み／IoT	**多様（ARM、RISC-V、特殊アーキテクチャが混在）**	**主導的**	組込み用途では Arm や他の低消費電力アーキテクチャ（RISC-V など）も多く使われ、x86 のシェアは限定的。

ChatGPT5(Auto)調べ，URLは確認済み

マルチコアの時代



消費電力　70～110W

処理時間　8時間

消費電力　150～200W

処理時間　1時間

今のスマホはマルチコアCPU搭載コンピュータ

- Apple iPhone 16e https://www.apple.com/iphone-16e/specs/

A18

A18 chip

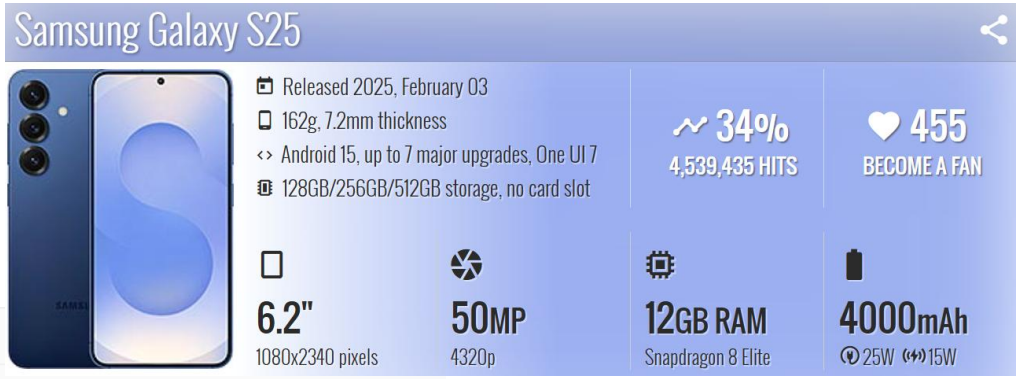
6-core CPU with 2 performance and 4 efficiency cores

4-core GPU

16-core Neural Engine

Hardware-accelerated ray tracing

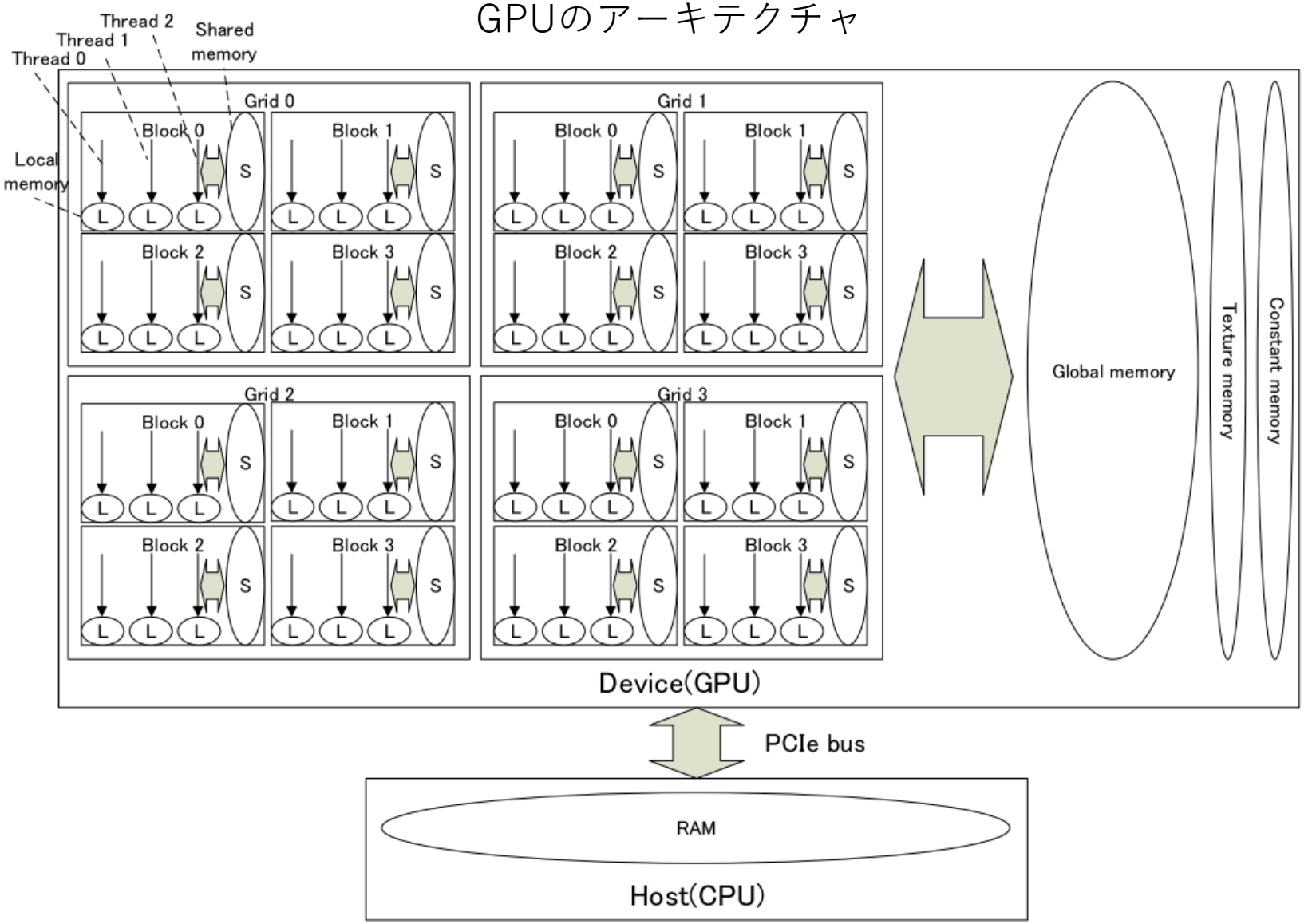
- Samsung Galaxy 25 https://www.gsmarena.com/samsung_galaxy_s25-13610.php

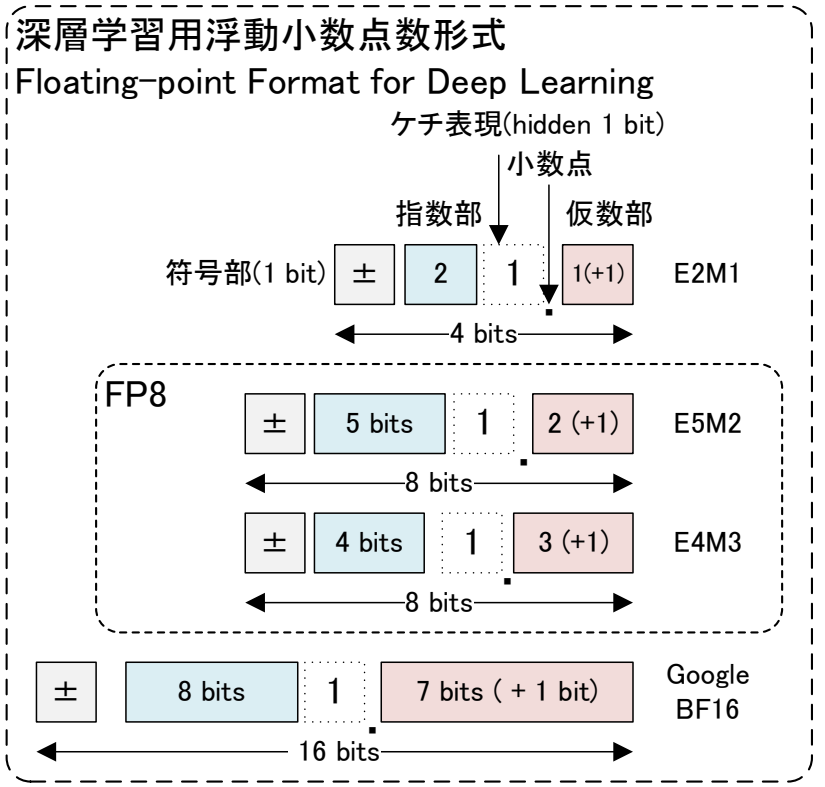


PLATFORM		
	OS	Android 15, up to 7 major Android upgrades, One UI 7
	Chipset	Qualcomm SM8750-AC Snapdragon 8 Elite (3 nm)
	CPU	Octa-core (2x4.47 GHz Oryon V2 Phoenix L + 6x3.53 GHz Oryon V2 Phoenix M)
	GPU	Adreno 830 (1200 MHz)

現在の環境：マルチコアCPU/GPU搭載PC

GPUのアーキテクチャ





自然数の乗算：筆算の並列化

乗算4回×4= 16回
→最大1/4の手間減

1234×8765

$= 1234 \times 5$ Thread1

$\rightarrow (1 \times 5) \times 10^3 + (2 \times 5) \times 10^2 + (3 \times 5) \times 10 + 4 \times 5 = 6170$

$+ (1234 \times 6) \times 10$ Thread2

$\rightarrow (1 \times 6) \times 10^4 + (2 \times 6) \times 10^3 + (3 \times 6) \times 10^2 + (4 \times 6) \times 10 = 74040$

$+ (1234 \times 7) \times 10^2$ Thread3

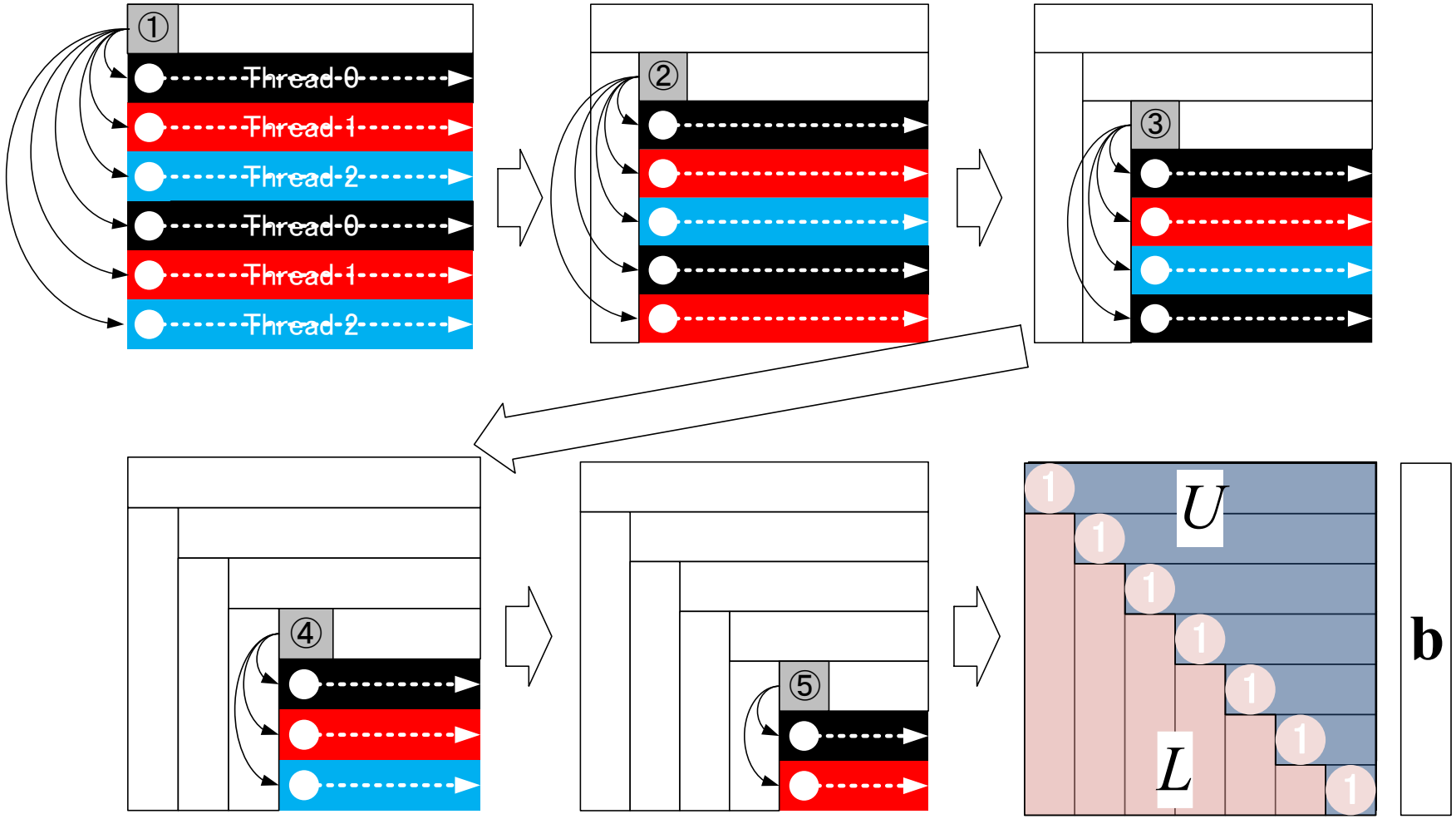
$\rightarrow (1 \times 7) \times 10^5 + (2 \times 7) \times 10^4 + (3 \times 7) \times 10^3 + (4 \times 7) \times 10^2 = 863800$

$+ (1234 \times 8) \times 10^3$ Thread4

$\rightarrow (1 \times 8) \times 10^6 + (2 \times 8) \times 10^5 + (3 \times 8) \times 10^4 + (4 \times 8) \times 10^3 = 9872000$

$= 10816010$

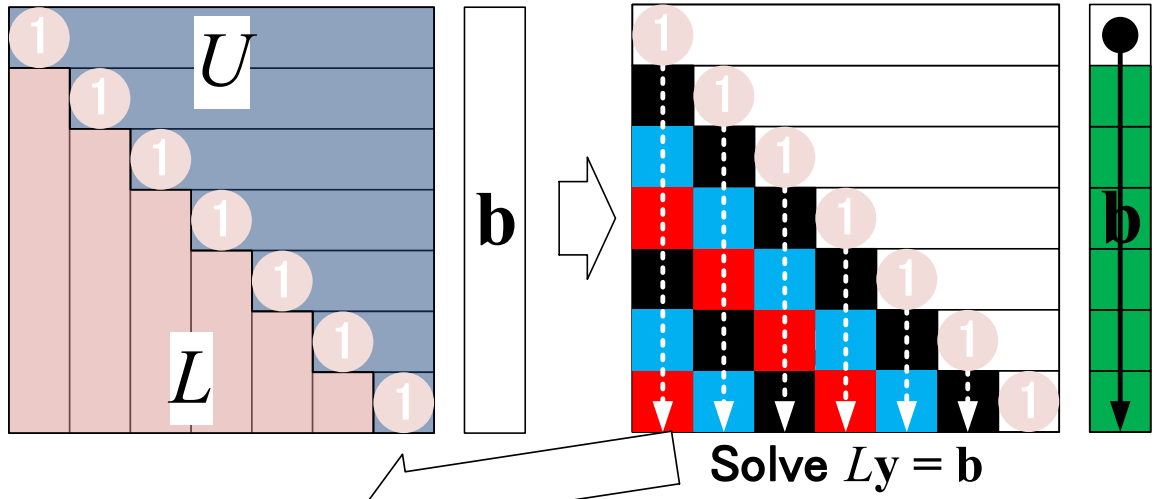
LU分解の並列化



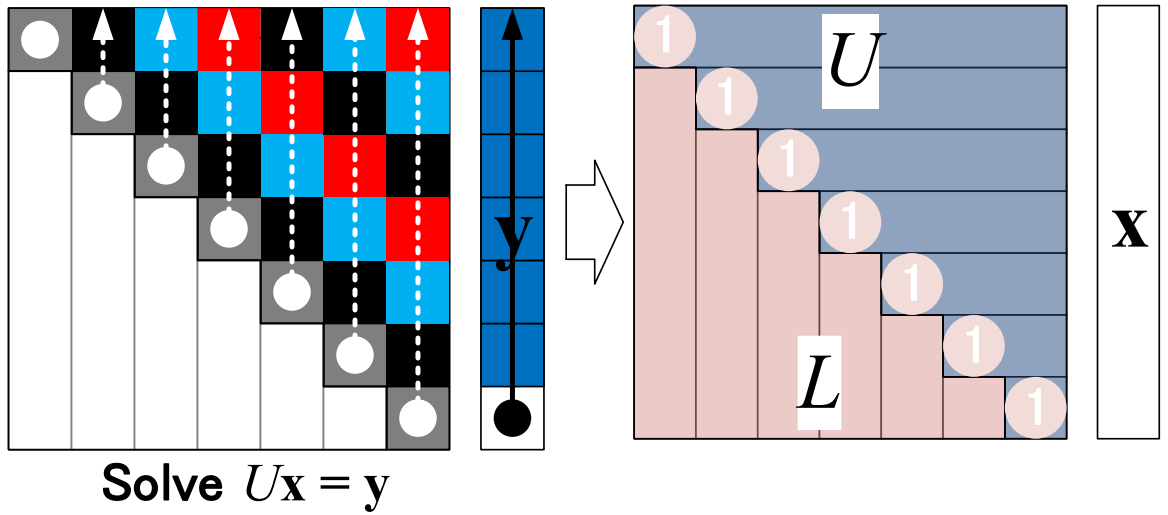
- 行（横方向）単位でスレッドに計算を割り当てる
- 計算が進むにつれて，並列化できる余地が減る

前進・後退代入の並列化

前進代入



後退代入

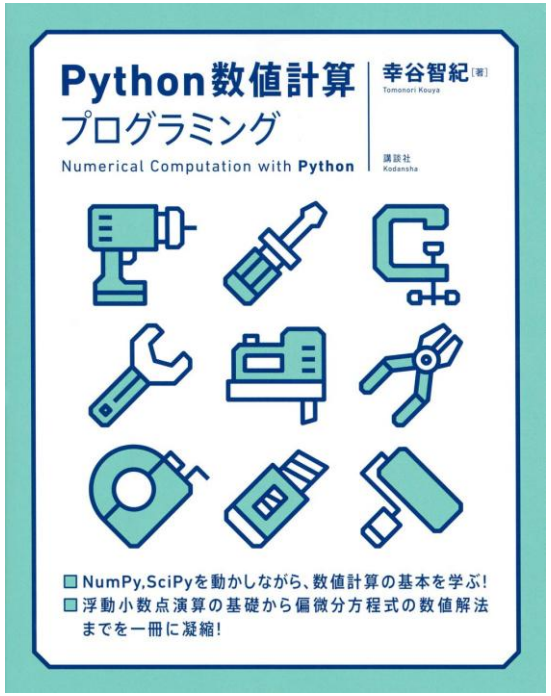


- スレッド割り当ては市松模様に
- LU分解より計算量が少ないので並列化の効果も少ない

Pythonによるデモ

- 3種類の連立一次方程式の解法の比較：計算時間（短いほど良い）と解の正確さ（有効桁数は長いほど良い）
- https://github.com/tkouya/inapy/blob/master/chapter07/linear_eq.py

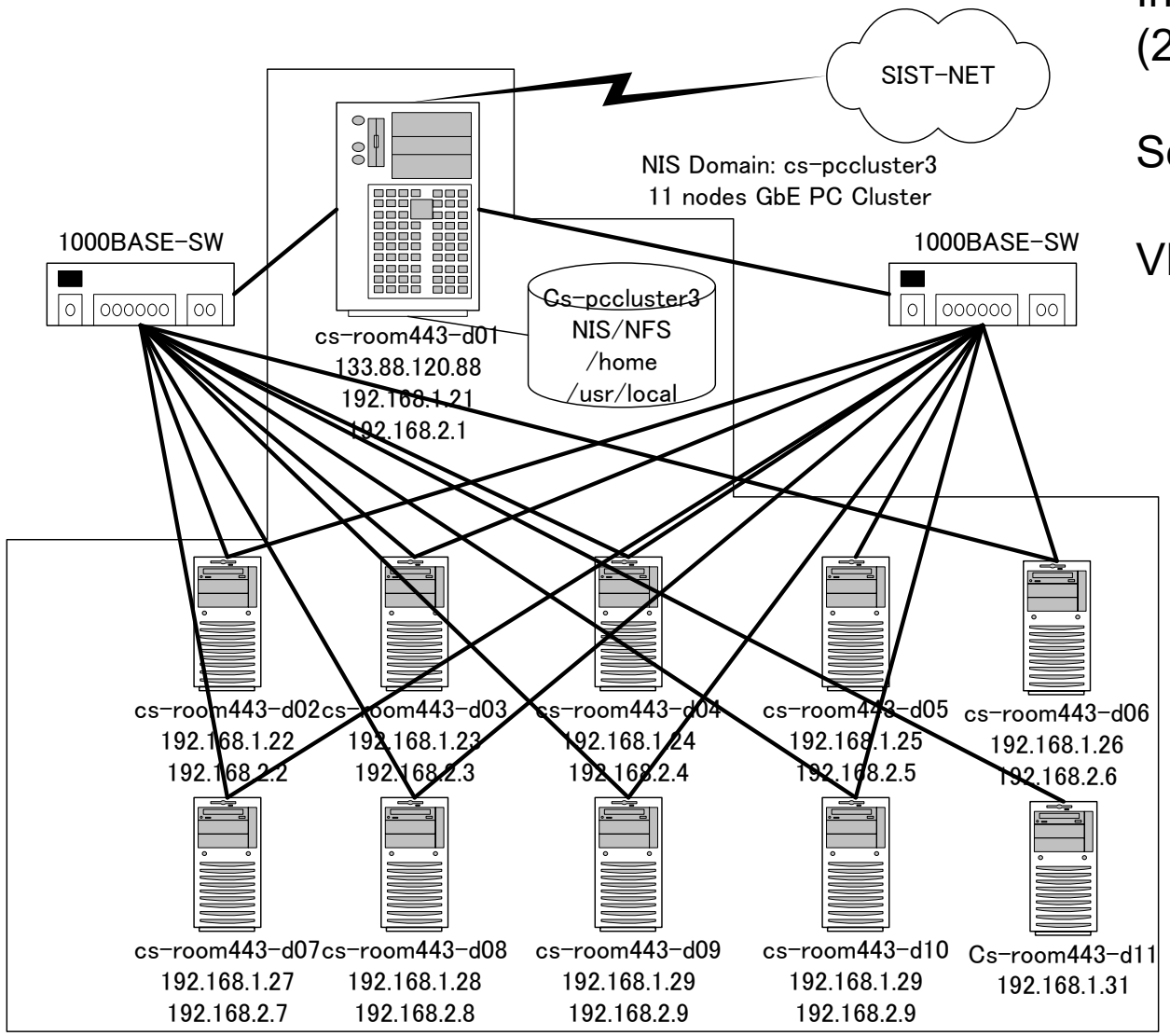
```
正方行列サイズ dim = 2000
方法1: time =          1 s, -log10(relerr(vec_x)) =       10.6 dec.digits
方法2: time =      0.301 s, -log10(relerr(vec_x)) =       11.2 dec.digits
方法3: time =      0.278 s, -log10(relerr(vec_x)) =       11.2 dec.digits
PS C:\Users\tkouy> & C:/Users/tkouy/AppData/Local/Programs/Python/Python31
2025/python/linear_eq.py"
正方行列サイズ dim = 5000
方法1: time =       3.26 s, -log10(relerr(vec_x)) =       10.7 dec.digits
方法2: time =       1.75 s, -log10(relerr(vec_x)) =       11.7 dec.digits
方法3: time =       1.18 s, -log10(relerr(vec_x)) =       11.7 dec.digits
PS C:\Users\tkouy> & C:/Users/tkouy/AppData/Local/Programs/Python/Python31
2025/python/linear_eq.py"
正方行列サイズ dim = 10000
方法1: time =       20.5 s, -log10(relerr(vec_x)) =       10.6 dec.digits
方法2: time =       9.69 s, -log10(relerr(vec_x)) =       11.1 dec.digits
方法3: time =        6.8 s, -log10(relerr(vec_x)) =       11.1 dec.digits
```



PCクラスタの例(2010年) @ 526実験室

- Pentium D + Windows 7 or Scientific Linux 6 ・・・ 11台

526実験室のネットワーク環境・・・自作



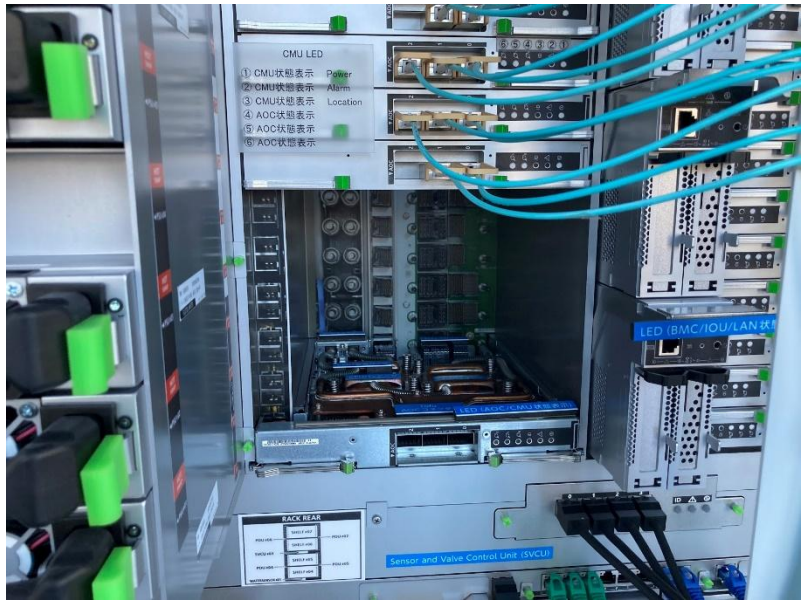
Intel® Pentium D 820 (2.8GHz) & Core i3/i7

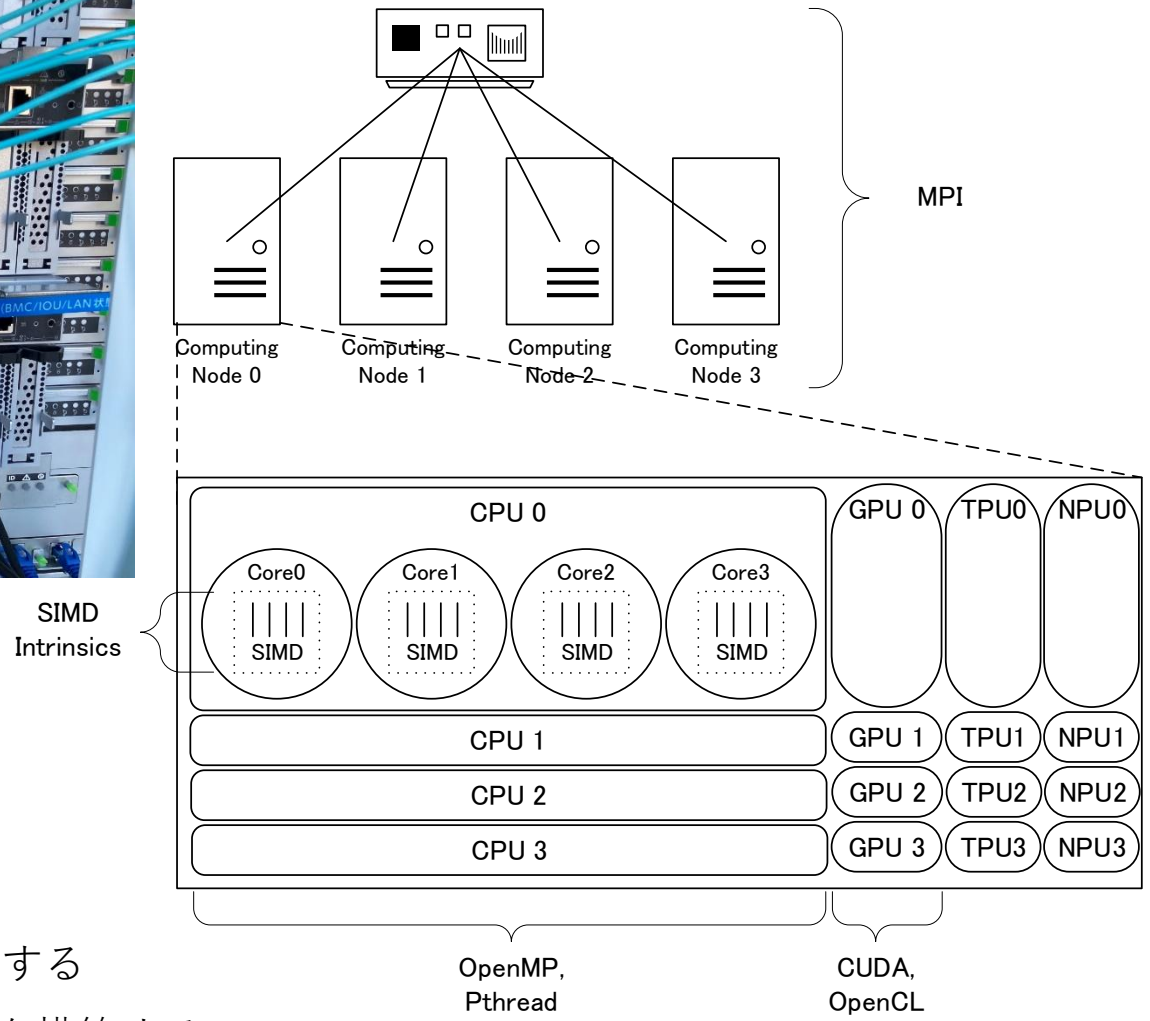
Scientific Linux 6 x86_64

VMware Server

富岳@理研・神戸(1/2)

富岳@理研・神戸(2/2)





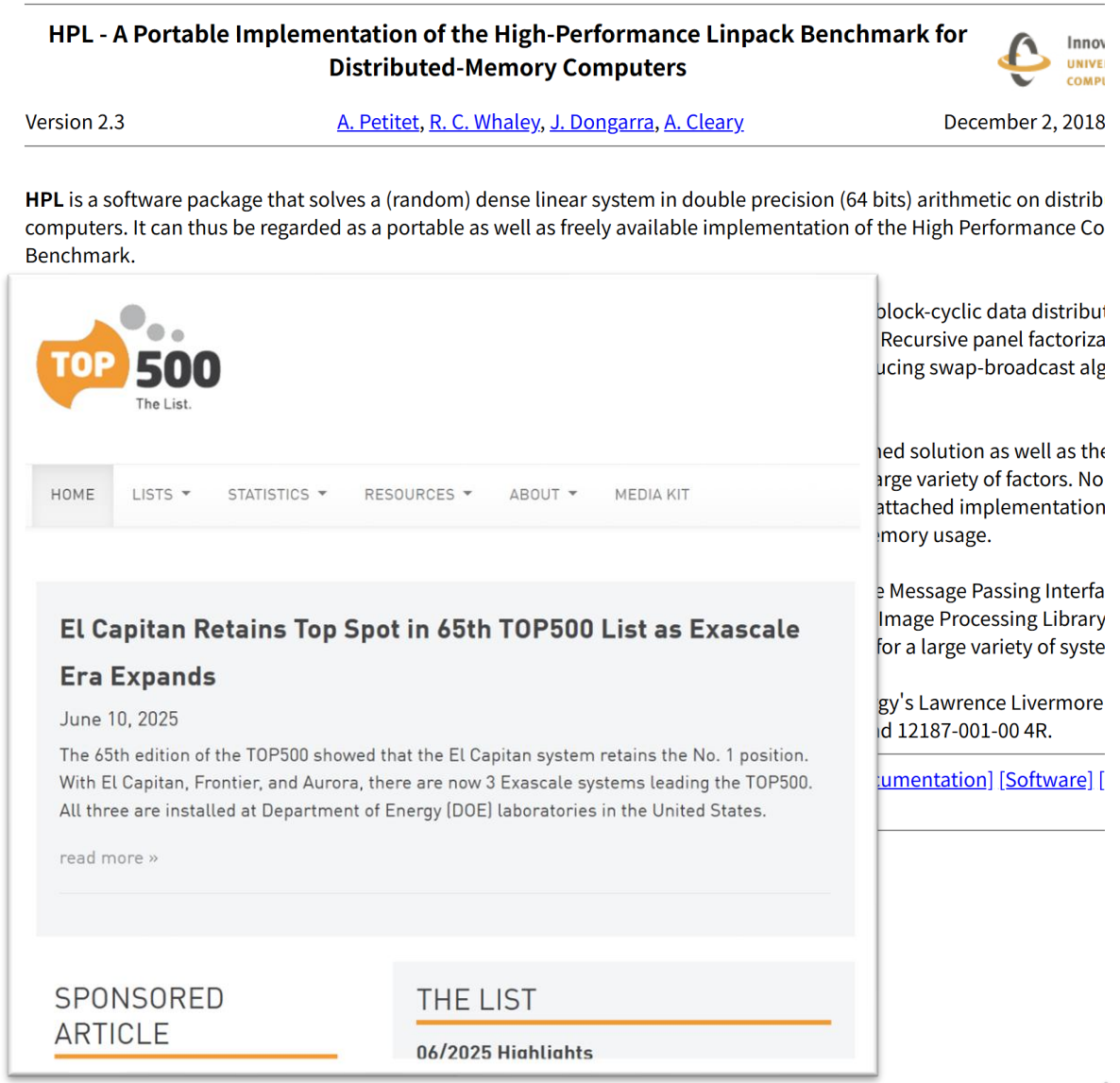
- マルチコアCPU，メニーコアGPUを複数積んだマシンを１ノードとする
- 各ノードをネットワークで繋いで巨大なコンピュータネットワークを構築する

→データの分散方法も考慮してプログラムを作る必要がある

LinpackテストとTop500(1/2)

- Linpackテスト・・・　並列LU分解を使って　連立一次方程式を解く https://www.netlib.org/benchmark/hpl/

- Top500 https://top500.org/
- 年２回（６月と１１月）にLinpackテストによるスーパーコンピュータの性能を評価する
- スーパーコンピュータの特性に合ったチューニング（プログラムの改良）を行うのが普通



Linpackテストと Top500(2/2)

2021年6月の順位

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646

1. 日本
2. アメリカ
3. アメリカ
4. 中国
5. アメリカ

2025年11月の順位

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	793.40	930.00	13,088
5	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	

1. アメリカ
2. アメリカ
3. アメリカ
4. ドイツ
5. アメリカ
→7. 日本の富岳

スーパーコンピュータの性能競争

- https://top500.org/statistics/perfdevel/
- Top500に登場するスーパーコンピュータの速度(Flop/s = 1秒当たりの計算数)
- 最近でも，5年で10倍に伸びている

まとめ

- スーパーコンピュータは何が「スーパー」なのか？

1. CPUのコア内部
    1. 64-bitの2進数を同時に処理
    2. SIMD処理ができる
2. CPUに複数のコアを保持（マルチコアCPU）
3. 1ノードに多数のマルチコアCPUと複数のGPUを持つ
4. 沢山のノードを高速ネットワークで結合

→大量の「並列処理」を行って短時間に多数の計算処理を行う

→AIの学習計算に有利