筑波大学計算科学研究センター
CCS HPCセミナー

並列システム

筑波大学計算科学研究センター

辻美和子

もくじ

- 並列計算機アーキテクチャ
  - 分散メモリ, 共有メモリ, SMP, NUMA...
- 並列処理ネットワーク
- 実システムの紹介

並列計算・並列計算機とは？

- 並列計算機は、複数の演算要素 (Processing Elements, PEs) を同時に利用する計算機システム
  - iPhoneのプロセッサもマルチコアの並列演算機
    - 例：2 Lighting cores, 4 Thunder cores
  - 米国のスーパーコンピュータ「フロンティア」
    - 1 AMD EPYC Milan processor (64 core) / 1 node
    - 4 AMD Instinct MI250x accelerators
    - 9408 nodes
  - 日本のスーパーコンピュータ「富岳」
    - 1 Fujtisu A64FX processor (48-cores) / 1 node
    - 158,974 nodes

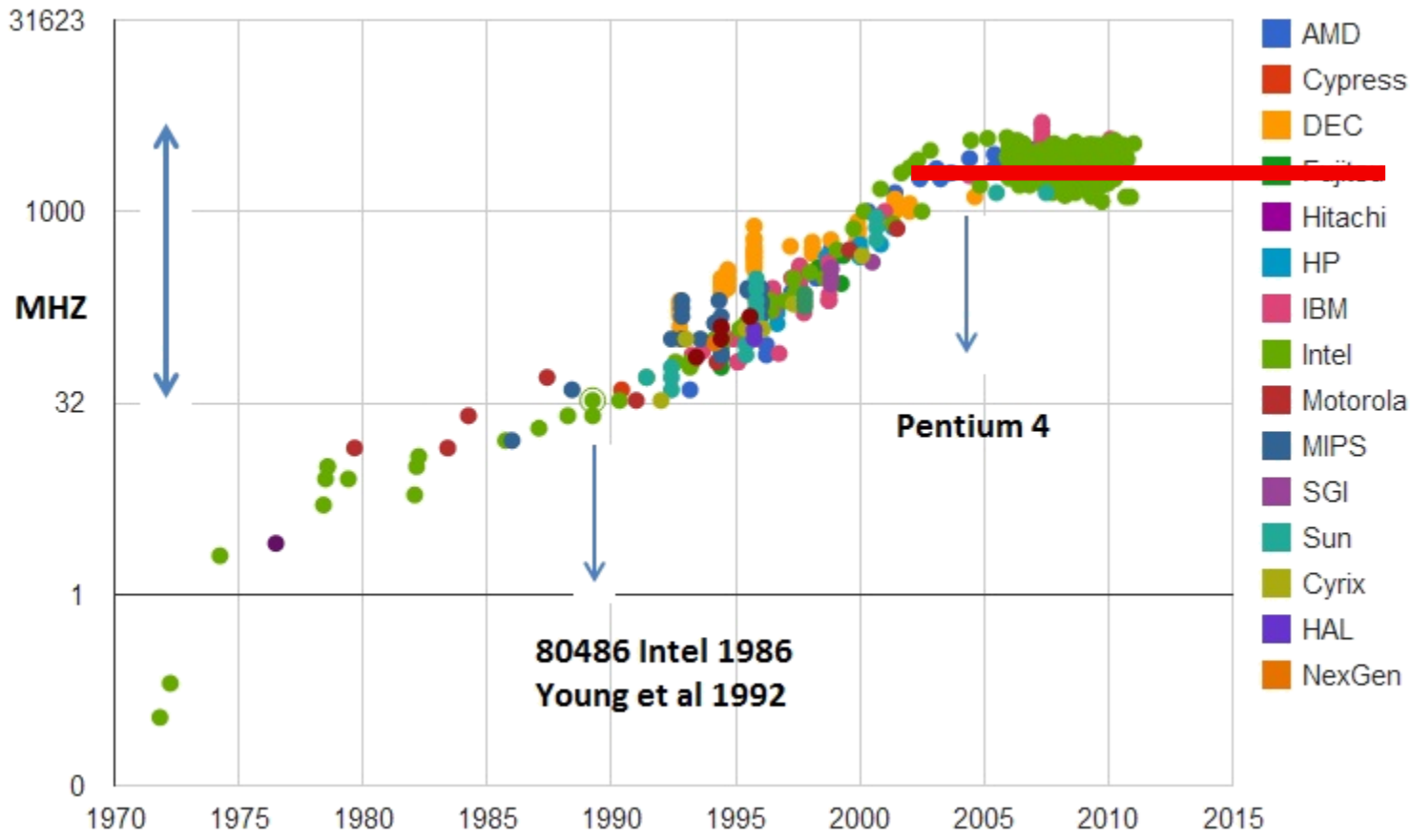
- 並列計算は、複数の計算が同時に実行される計算のやりかた
- ~~数の力でごり押しするのがHPC~~ 性能を求めて並列化するのだ
  - ⇒ 性能、それは、、
    - 解を得る速度
    - 計算の規模の大きさ

どうして並列計算？

- 早くしたいから
  - 周波数をあげることで速度を上げる ⇒ とっくに限界
    - 冷却の難しさ
    - リーケージによる電力ロス
  - 消費電力
    - CPUの周波数に比例し
    - 高周波数は高電圧が必要、CPUの電圧の二乗に比例

    ⇒ 並列計算の場合
    電力は並列数にだいたい比例
- しかし、周波数をあげれば
  - 性能はほぼ確実にあがる ☺
- 並列性を上げると
  - いつも性能があがるとは限らない ☹
  - 電力効率は良い

  ⇒ 地球環境のために並列計算を勉強しよう！

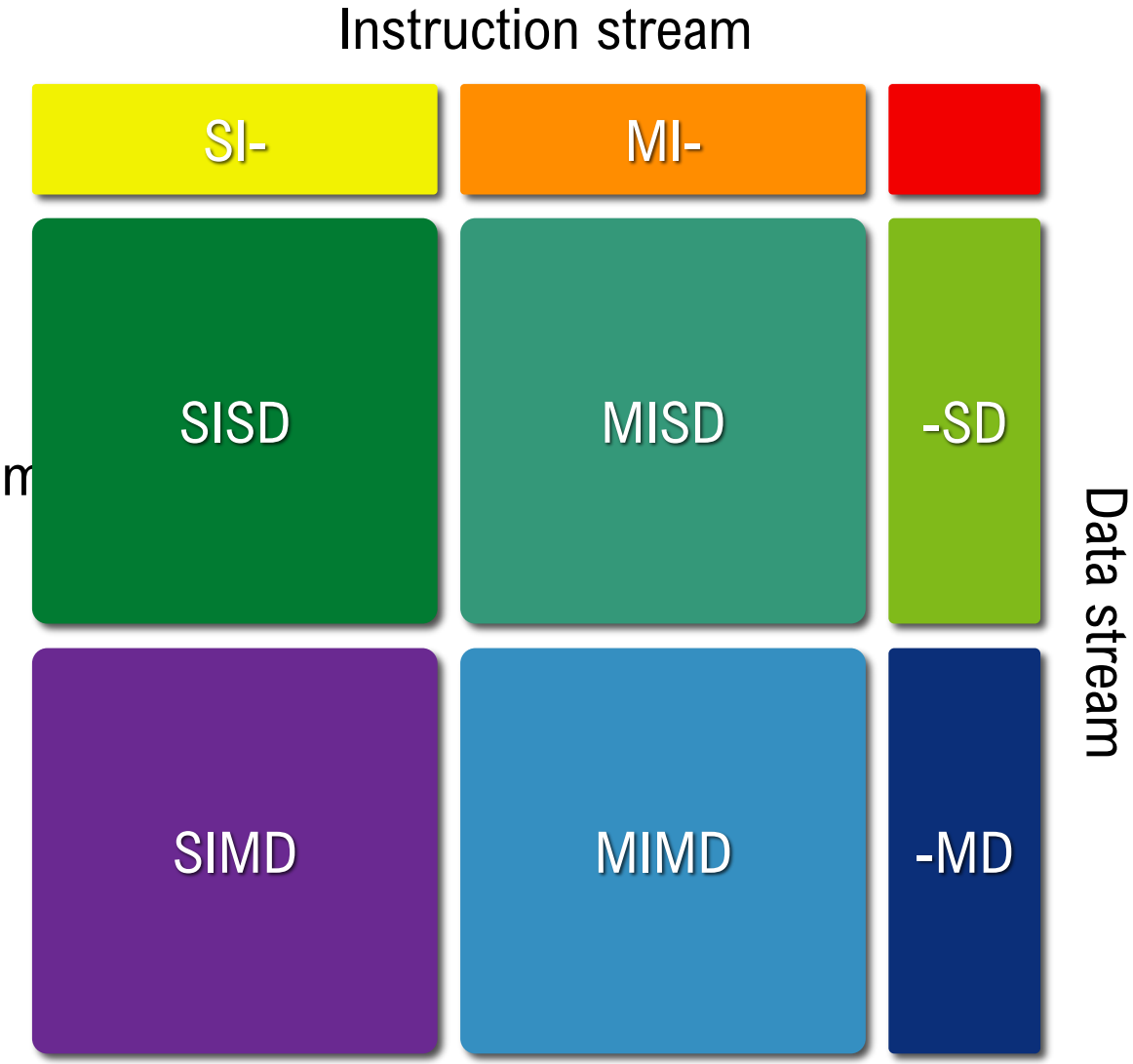


https://en.wikipedia.org/wiki/File:Clock_CPU_Scaling.jp

フリンの分類 Flynn's Taxonomy

- SISD Single Instruction stream, Single Data stream
  - 並列性なし、完全に逐次
- SIMD Single Instruction stream, Multiple Data stream
  - 異なるデータに対して同じ演算を同時に実行
- MISD Multiple Instruction stream, Single Data stream
  - (ほぼない)
  - 異なる演算を同じデータに実行
- MIMD Multiple Instruction stream, Multiple Data stream
  - 異なる演算を、異なるデータに同時に実行

- 近代のHPCシステムは、これらのハイブリッド

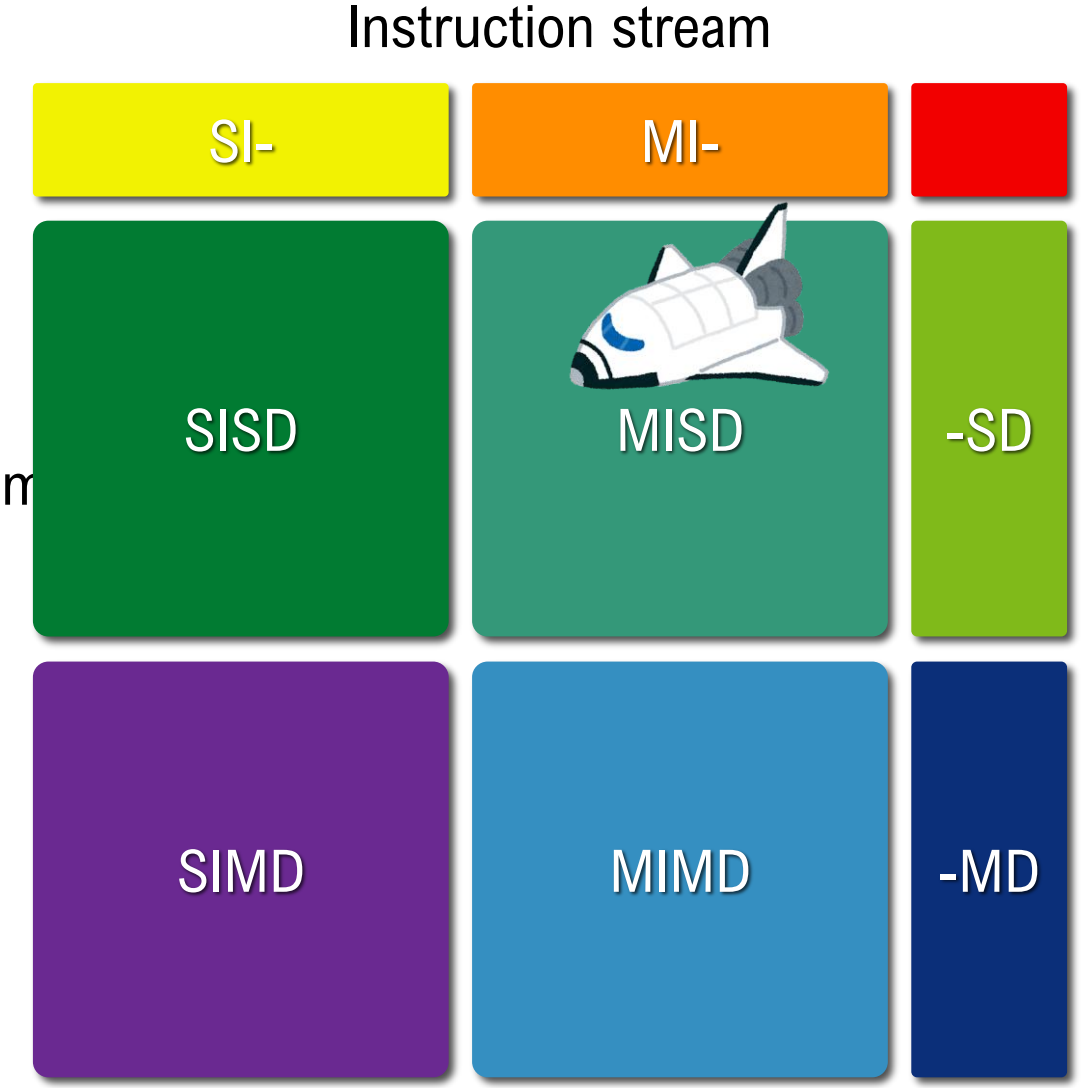


フリンの分類 Flynn's Taxonomy

- SISD Single Instruction stream, Single Data stream
  - 並列性なし、完全に逐次
- SIMD Single Instruction stream, Multiple Data stream
  - 異なるデータに対して同じ演算を同時に実行
- MISD Multiple Instruction stream, Single Data stream
  - (ほぼない)
  - 異なる演算を同じデータに実行
- MIMD Multiple Instruction stream, Multiple Data stream
  - 異なる演算を、異なるデータに同時に実行

- 近代のHPCシステムは、これらのハイブリッド

- 余談：1980年台のスペースシャトルは異なるシステムの「合議制」なMISDシステム



「並列」の種類

- 命令レベルの並列
- SIMD
- 分散並列システム
- 共有メモリシステム

命令レベルの並列性

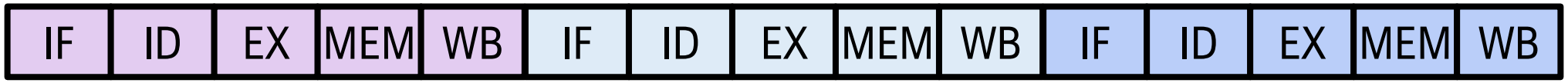
- 命令とは ..
  - プロセッサの1つのオペレーション
  - 命令セットアーキテクチャ instruction set architecture (ISA) により定義される
  - 足す、掛ける、読む、書く、などなど
- 例 (ARMv8.2)

これも命令
変数のメモリ上の位置（アドレス）
を「計算」している

```
//  y[] = y[] + a[]*x[];
ldr     d0, [x0, x3, lsl #3] // load ‘a’
ldr     d1, [x1, x3, lsl #3] // load ‘x’
ldr     d2, [x2, x3, lsl #3] // load ‘y’
fmul    d1, d1, d0           // a*x
fadd    d1, d2, d1           // y+(a*x)
str     d1, [x2, x3, lsl #3] // store y+(a*x)
```

命令レベルの並列性

- 1クロックのうちに複数の命令を実行する
- プログラムは命令のストリームである

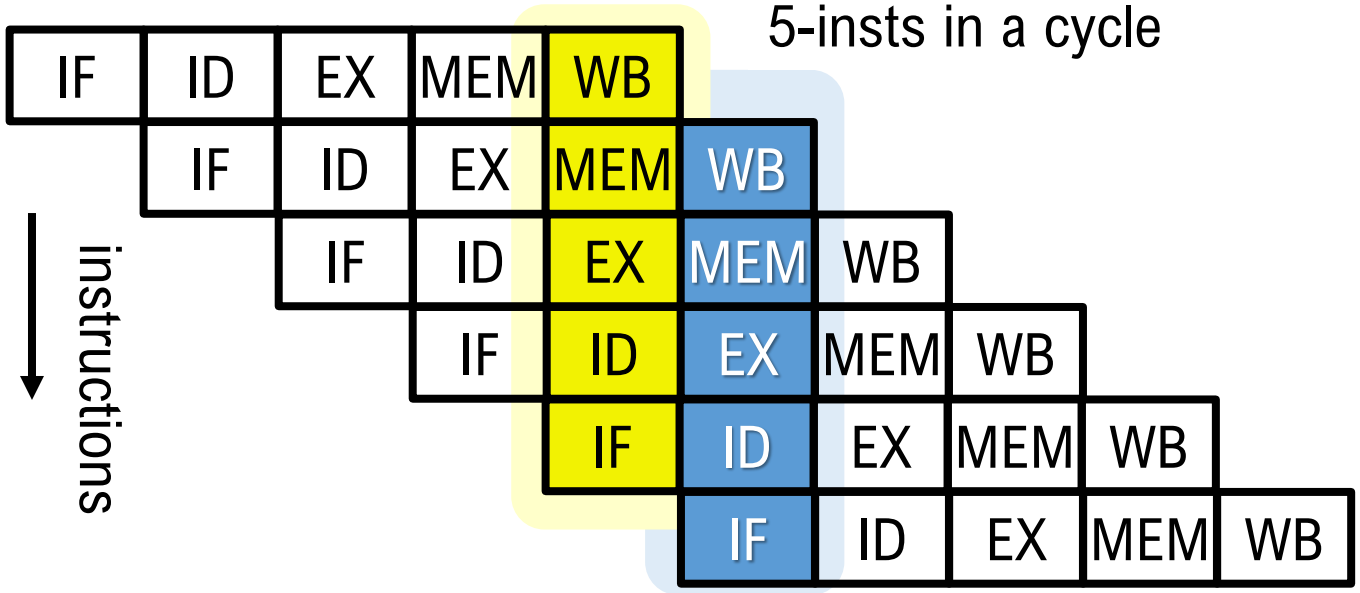


time

parallelization based on “**pipelining**”

time

5-insts in a cycle



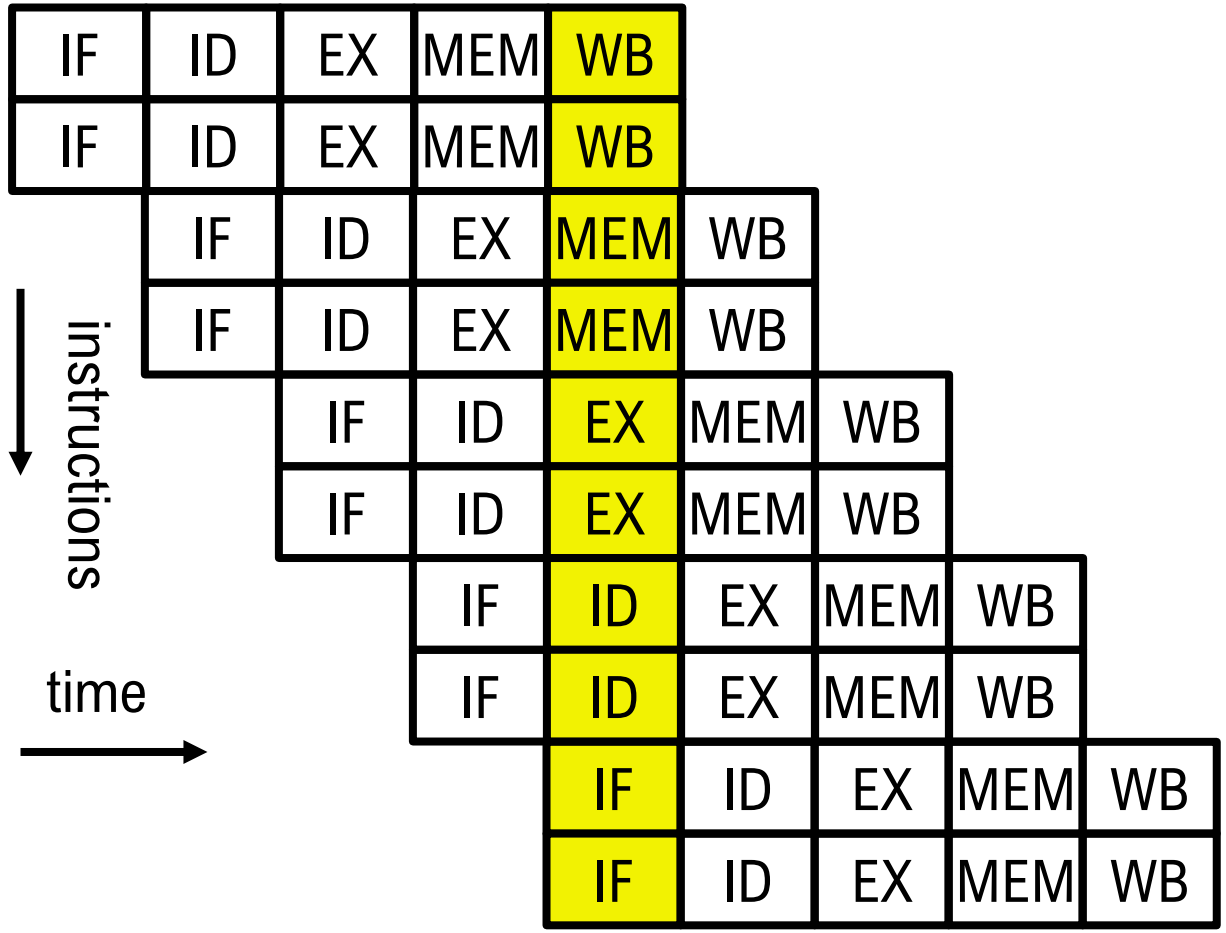
IF: Instruction Fetch
ID: Instruction Decode
EX: Execute
MEM: Memory access
WB: Register Write Back

命令レベルの並列性

- 1クロックのうちに複数の命令を実行する
- プログラムは命令のストリームである
- 近年のプロセッサは、1サイクルに2つ以上の命令を発行する

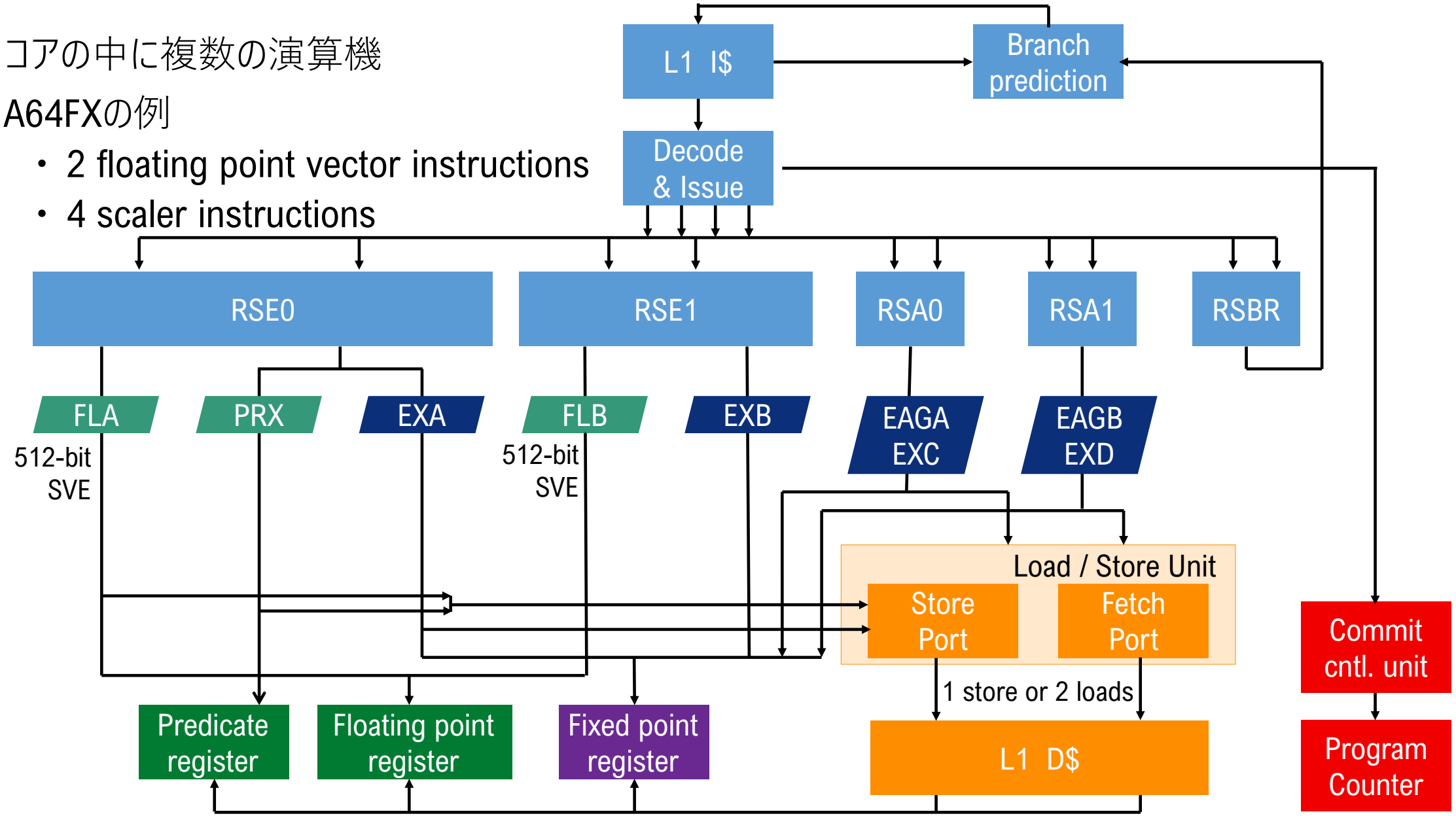
- アウトオブオーダ実行
  - dependency between interactions

  can be taken into account by HW



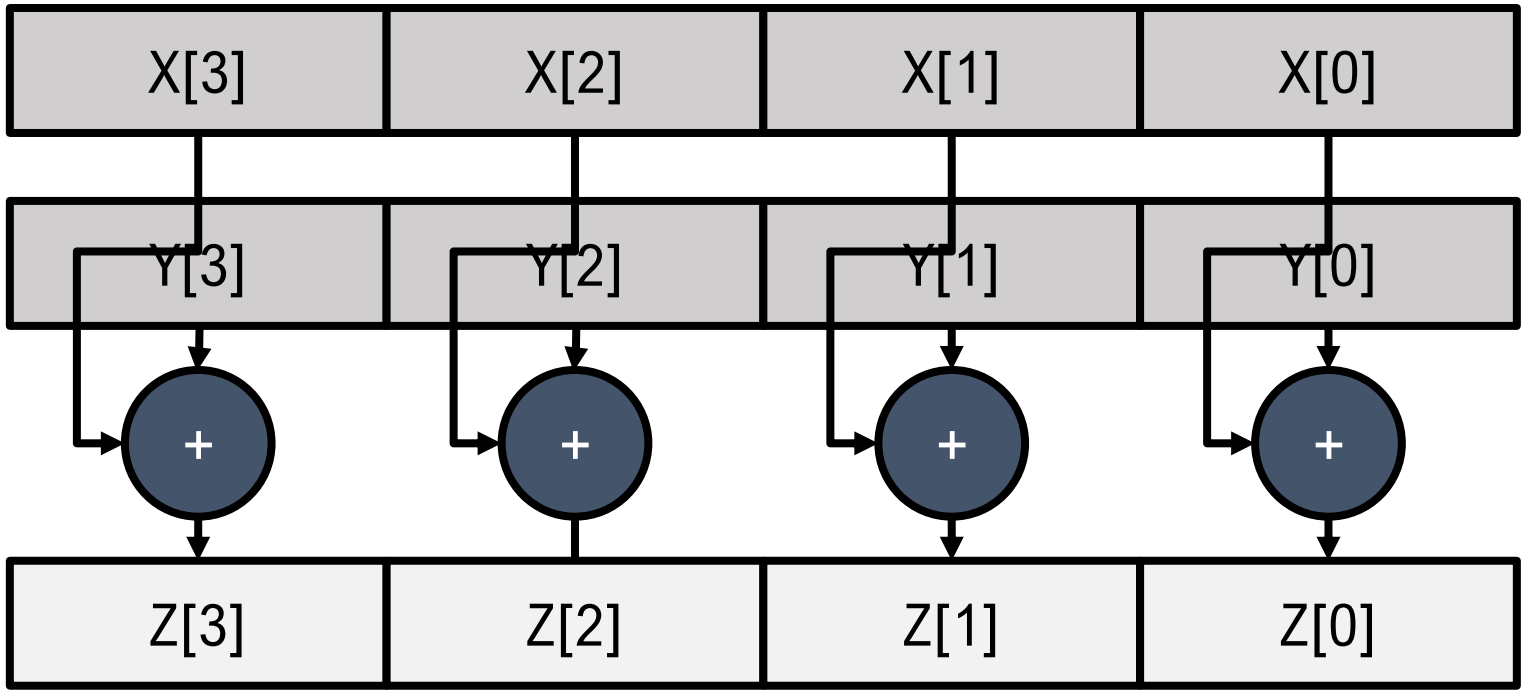
命令レベルの並列性

- コアの中に複数の演算機
- A64FXの例
  - 2 floating point vector instructions
  - 4 scaler instructions



SIMD 命令

- 複数のデータに対して、1つの命令を適用する
  - A64FX (Fugaku)
  - Intel-AVX512 series:
    512-bit SIMD
    8 double precision



SIMD 命令

```
for(i=0; i<n; i++){
    z[i] = x[i] + y[i];
}
```

```
for(i=0; i<n; i+=4){
    z[i+0] = x[i+0] + y[i+0];
    z[i+1] = x[i+1] + y[i+1];
    z[i+2] = x[i+2] + y[i+2];
    z[i+3] = x[i+3] + y[i+3];
}
```

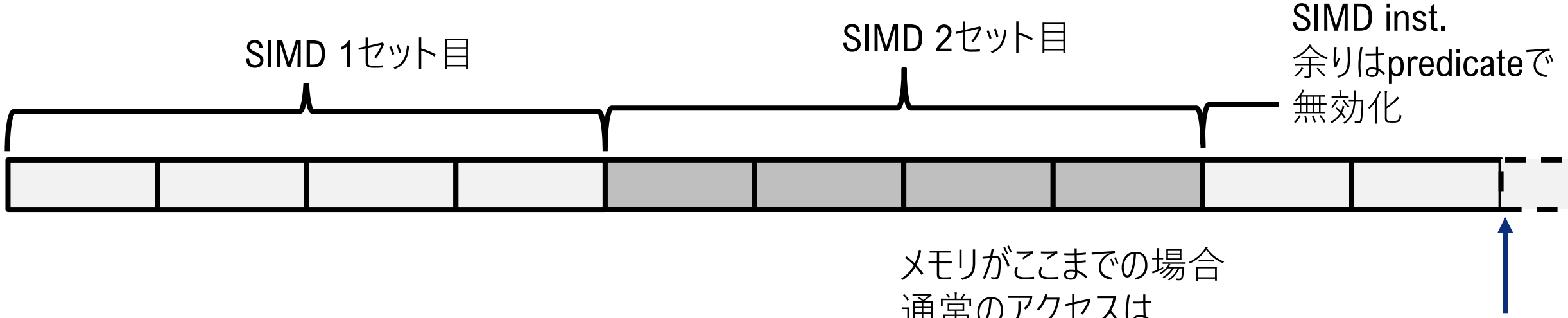
一度に計算

```
//   z[i] = x[i] + y[i];
ld1d    {z0.d}, p0/z, [x0, 2, mul vl] // load x[i],x[i+1], …
ld1d    {z1.d}, p0/z, [x1, 2, mul vl] // load y[i],y[i+1], …
fadd    z2.d, p0/m, z1.d, z0.d        //  add x[i]+y[i], x[i+1]+y[i+1],…
st1d    {z2.d}, p0, [x2, 2, mul v1]   //  store z[i],z[i+1], …
```

SIMD 命令は “**ベクトル**” レジスタに対して作用する。

こんなこともできる SIMD 命令 (1) : プレディケート（マスク）

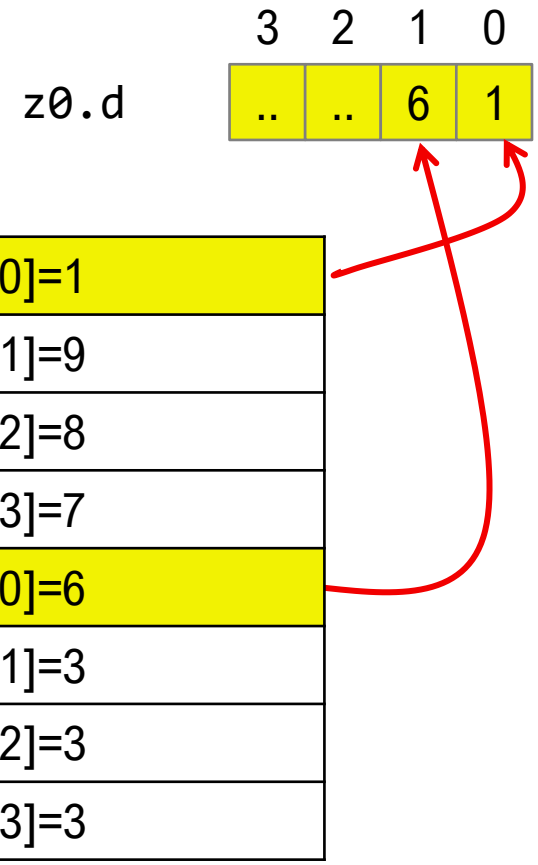
- SIMDが長くなる
  - 計算効率は良いが、小回りが利かない
  - 例：AAarch64+SVEは最大で2048ビットのベクトル長
- 一部のビットのみ操作したい
- Per-lane predication
  - プレディケート (マスク) レジスタで、各レジスタに作用するかしないかを定義できる
- ループの終わりがいい感じに割り切れない場合、「余り」部分のみプレディケートで無効化



こんなこともできる SIMD 命令 (2) : Gather & Scatter

- SIMD命令の対象となるデータはベクトルレジスタ（連続なレジスタ空間）
- 一方で、メモリ上離れたデータに同じ操作を適用したいことがある
- Arm8.2+SVE では **gather load/scatter store** がサポートされている
  - 1つの命令で、連続しないデータをベクトルレジスタに読む
  - 連続するベクトルレジスタを連続しないメモリアドレスに書く

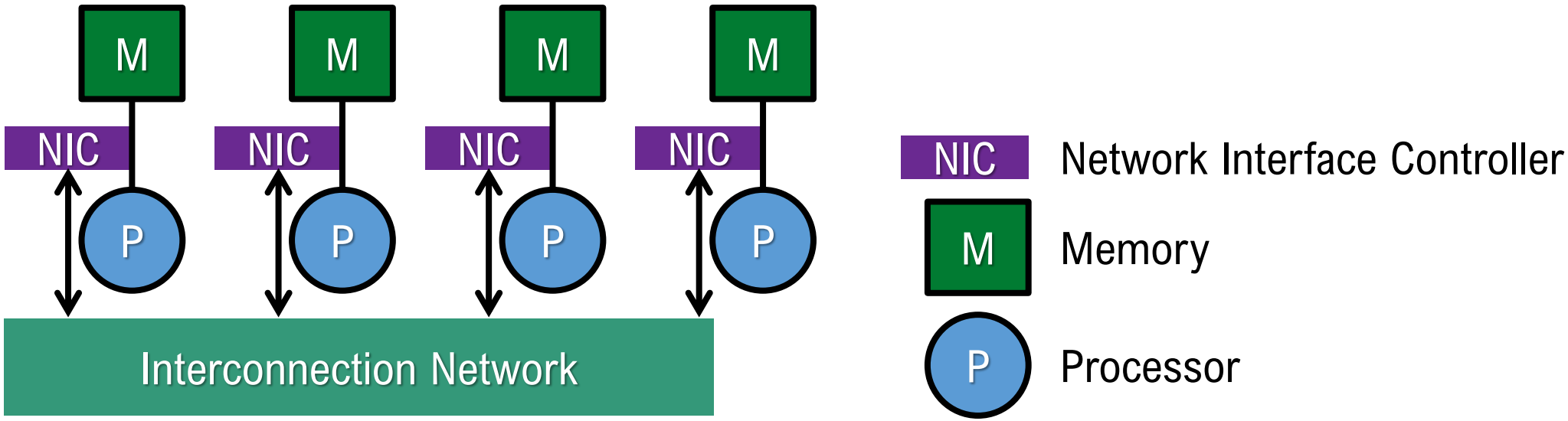
```
   ld1d z0.d, p0/z, [x0, z1.d, LSL #3]
x0    = 0x1000          base address (scaler register)
z1.d = {0, 4, 8, 12}   indexes (SVE register)
LSL #3                 shift
p0/z = {1, 1, 1, 1}    predicate register
z0.d                   destination register (SVE register)
```



- 命令レイテンシは長い
  - A64FXの場合連続するロードの1.6倍以上
- ベクトルレジスタが2つ必要
  - アドレスのリスト
  - データ

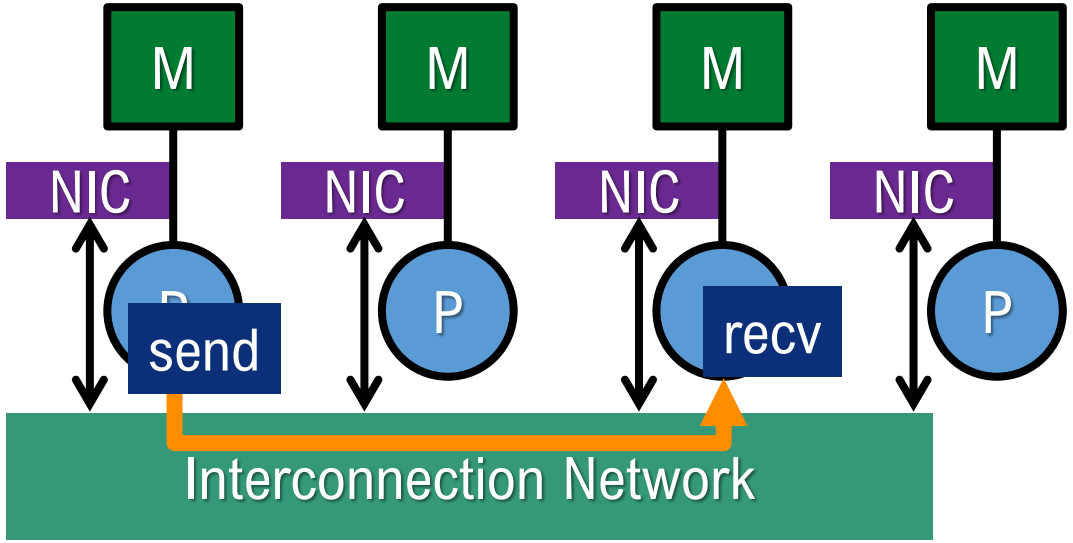
分散メモリシステム

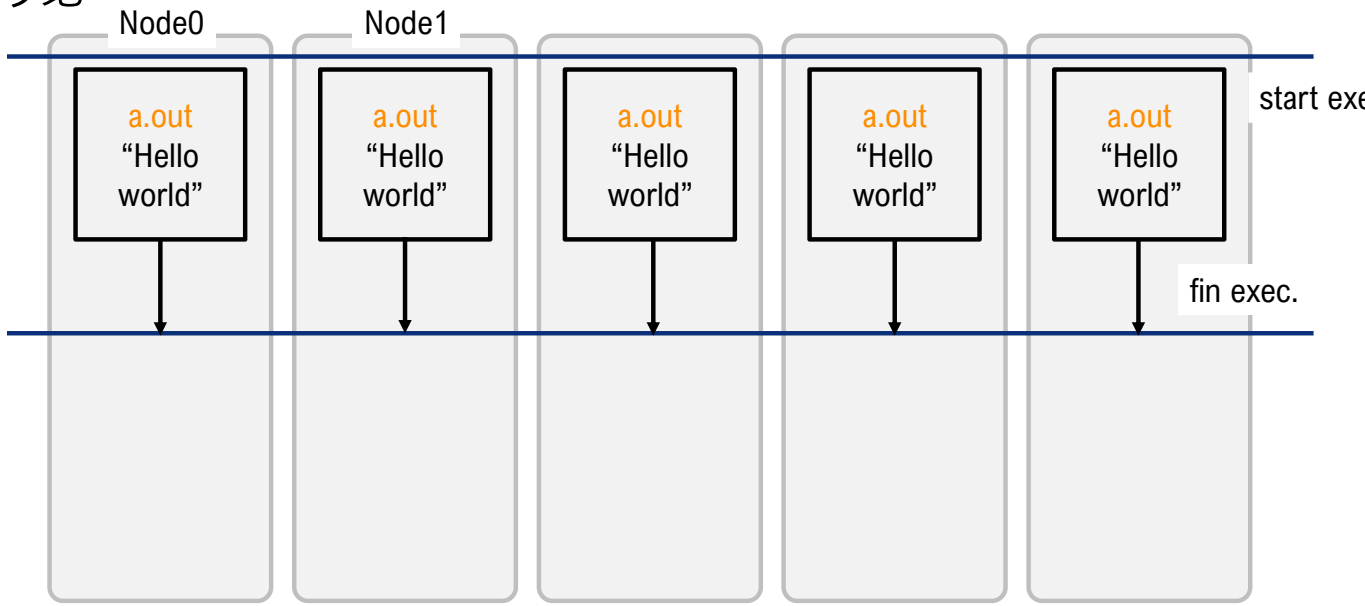
- 各プロセッサがそれぞれローカルメモリ空間を持つ
- メモリは空間は、理論的に、もしくは**物理的**に分散している
- CPUとメモリを持つノードがネットワークで接続されている
- 各プログラムはネットワークを介してデータを交換する
- 利点は拡張性の高さ
  - 大規模な並列システムの効率が可能
  - **Cluster**



分散メモリシステムのプログラミング Message Passing

- 分散メモリシステムでは、各プロセッサはすべてのデータにアクセスできるわけではない
- ローカルでは**ない**メモリにアクセスするためには通信が必要
  - Message passing Interface (MPI)
  - Parallel Virtual Machine (PVM)
  - などなど
- アプリケーション開発者は、陽に並列化を行う必要がある
  - ワークマッピング
  - データ分散
- MPIがデファクトスタンダード
  - 複数のノードに同時にプログラムを起動
  - 「同じプログラム」が基本
  - 別々のことをやらせたいときは if-thenで

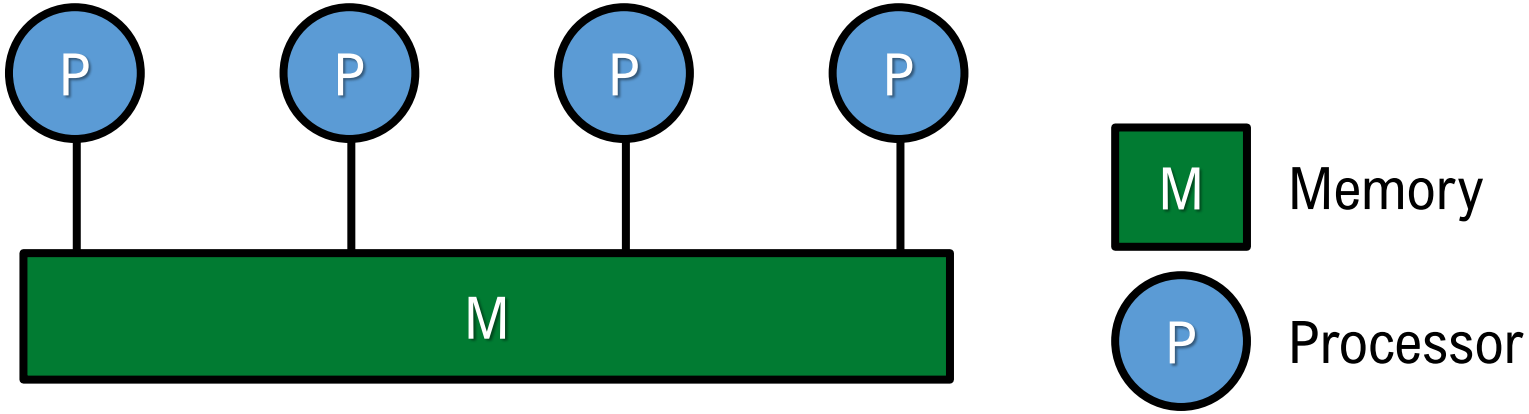




共有メモリシステム

- すべてのプロセッサは単一のメモリアドレス空間を共有する
- 各プログラム（この場合処理単位はスレッドと呼ばれる）どうしは、メモリにデータを書く・読むことでデータを交換する

- 近年のCPUは、1つのプロセッサの中に多数のコアを持ち、コアはメモリを共有する
- 拡張性は低い



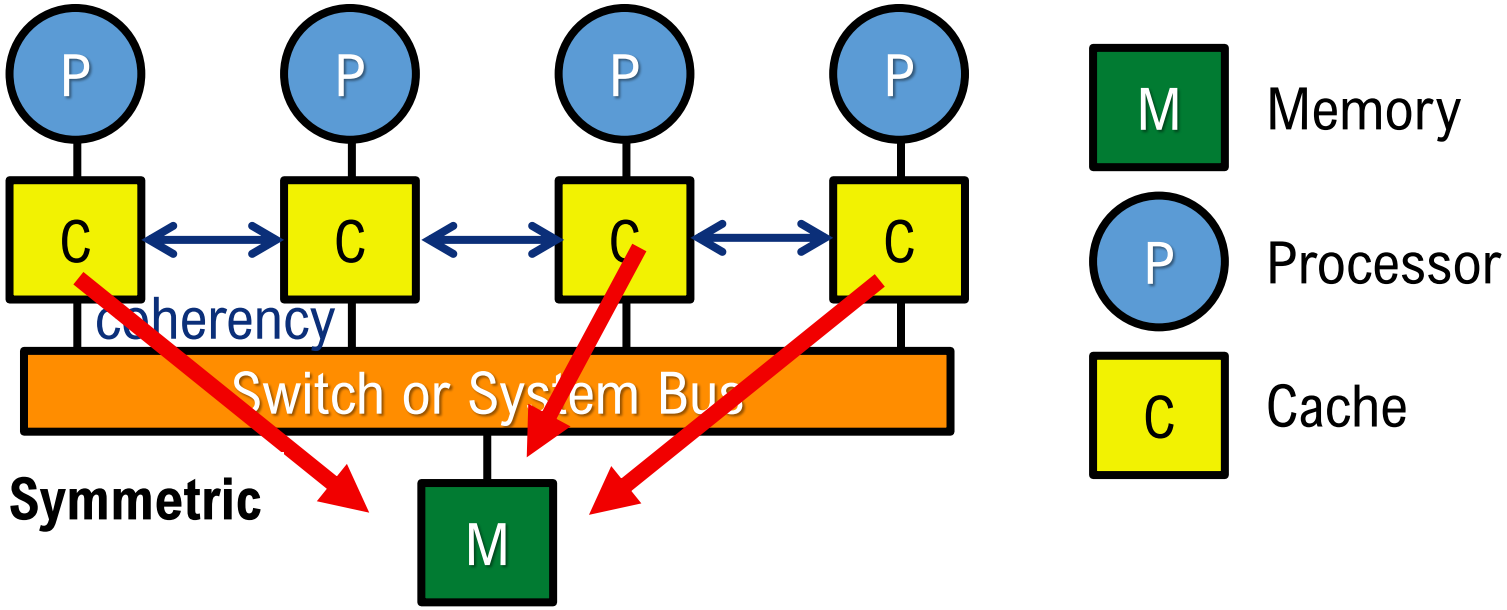
共有メモリシステム:SMP Symmetric Multi-Processor

- 複数のプロセッサが1つの共有メモリに接続される
  - プロセッサは、共有メモリにネットワークスイッチやバスを介してアクセスする
- すべてのプロセッサからのメモリアクセスは均一
  - アクセス時間等を考えずにプログラミングできる
- 昔: キャッシュなし
- 近年: コヒーレントなキャッシュを持つ（複数のキャッシュ間でデータの整合性を保つ）
- 拡張性に限界：コヒーレントキャッシュの実現が難しくなる
- メモリの読み書きが集中すると
  性能が低下する

富士通 HPC2500

日立 SR16000

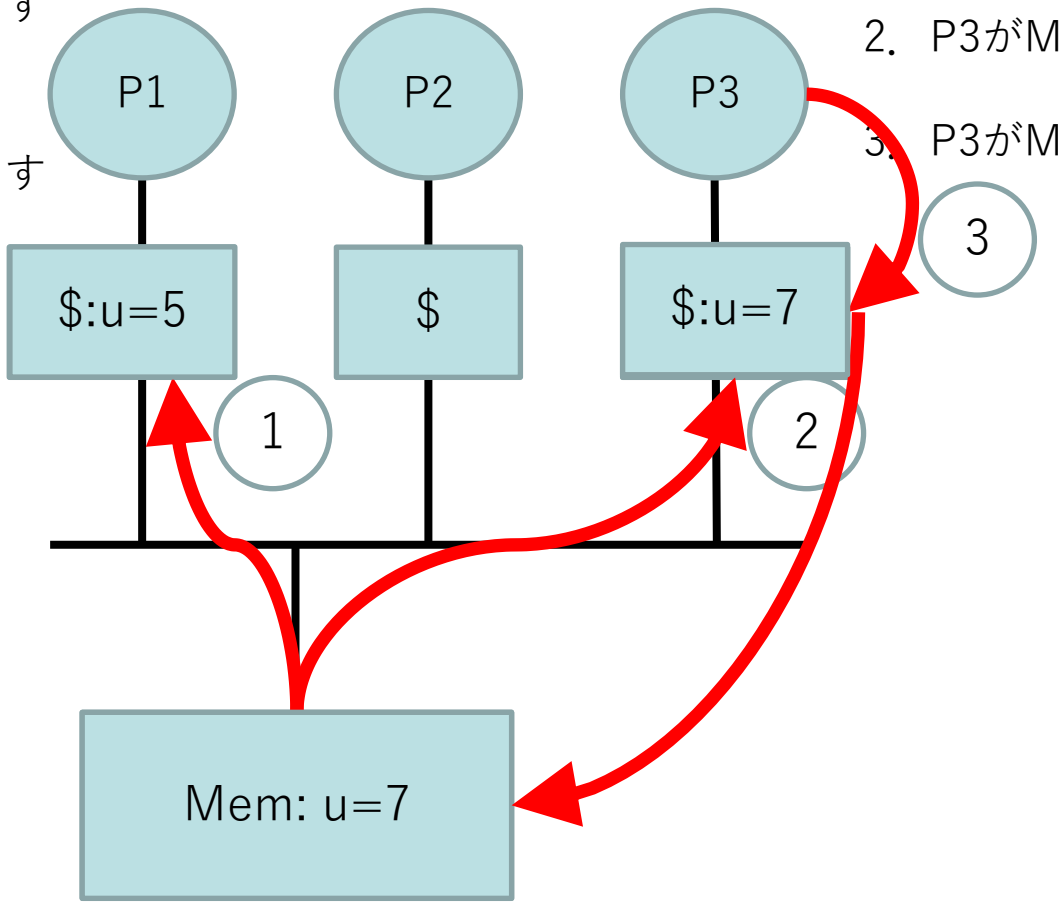
とか、



分散メモリシステム：キャッシュコヒーレントとは？

- コヒーレントでないキャッシュ

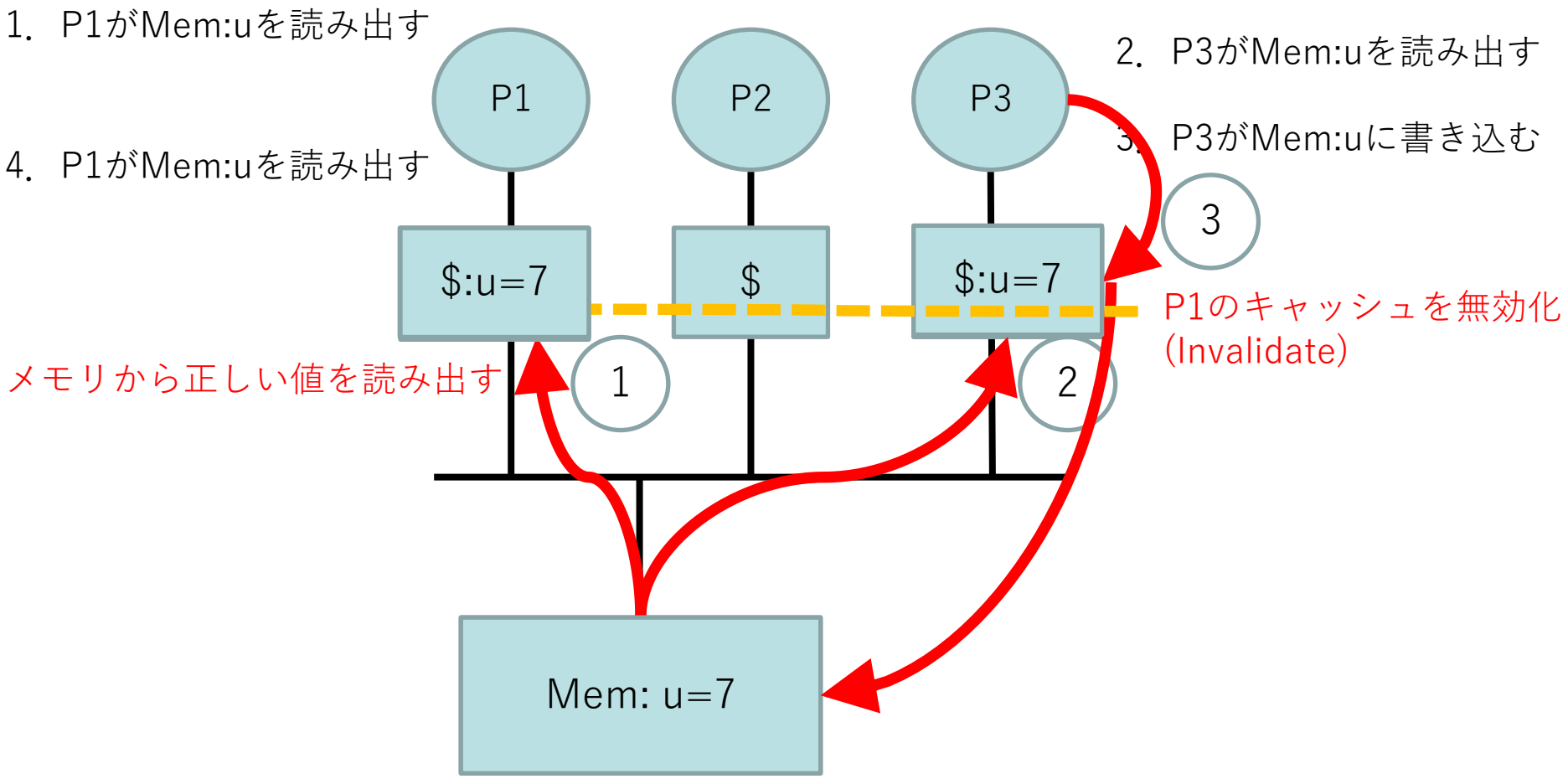
1. P1がMem:uを読み出す
2. P3がMem:uを読み出す
3. P3がMem:uに書き込む
4. P1がMem:uを読み出す



元の値u=5を読み出してしまう
(メモリの最新データは既に
キャッシュにあると思っている)

分散メモリシステム：キャッシュコヒーレントとは？

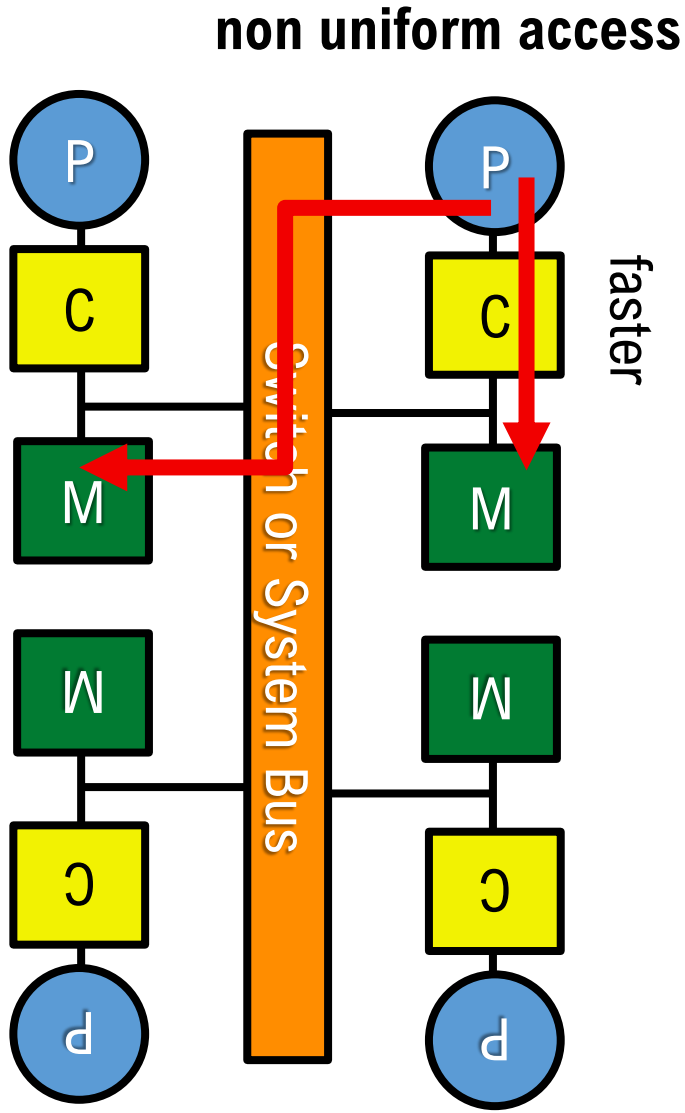
- コヒーレントなキャッシュ



共有メモリシステム: NUMA Non-Uniform Memory Access

- 各CPUがローカルメモリを持つ
- あるCPUはバスもしくはスイッチを介して別のCPUのメモリにアクセスできる
- 非対称、つまり、遠くのメモリにアクセスするときは、ローカルメモリのアクセスよりも時間がかかる
- 必要なところに必要なメモリを配置してあげることで高速化できる

- AMD Opteron Barcelona (2007)



共有メモリシステム：構成の詳細

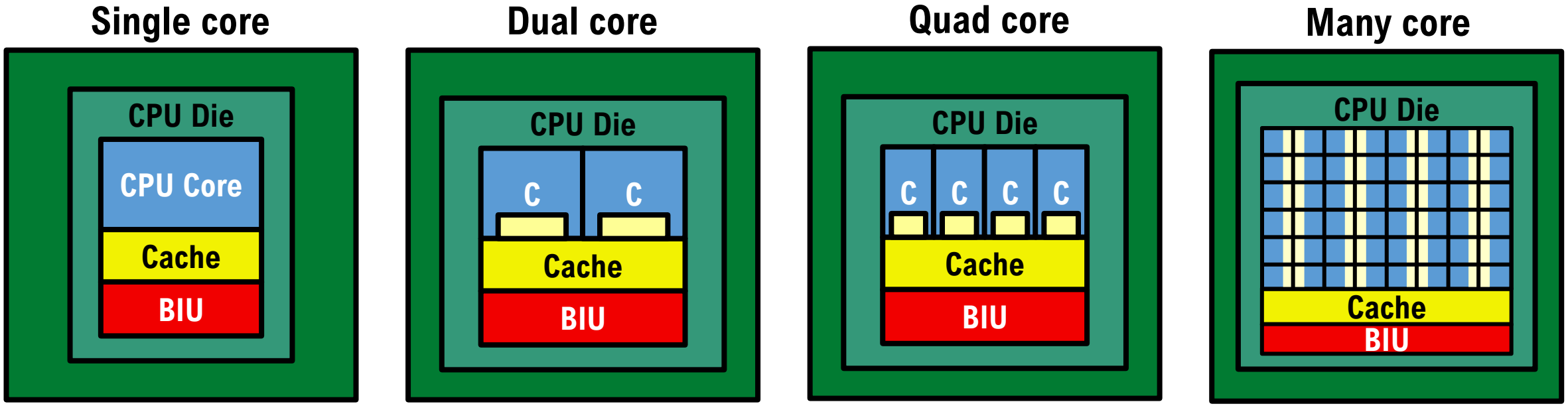
- System scalabilityを確保するため単純バス構造の共有メモリシステムはもはや存在しない
  - bus bottleneck（busは一時には1つのtransactionで占有されてしまう）
  - 複数busを持つシステムもかつてはあった
  - crossbar networkの導入：プロセッサとメモリの結合が実際にはスイッチ結合になっている

- 共有メモリへのアクセス衝突を避けるための工夫
  - memory bank分け：適当なアドレスブロック毎に別のmemory moduleに分散して振り分け
  - coherent cache：各プロセッサは固有のキャッシュを持ち、普段はそのデータを参照する。他のプロセッサによるデータ更新をキャッチし、うまく自分のキャッシュに反映する。
  - NUMA (Non-Uniformed Memory Access)：物理的にはmemory moduleが分散していて、アドレスに寄るメモリへの距離の差が存在する。coherent cacheと共に用いられるのが普通

マルチコア

- コアは演算ユニット
- 2つ以上のコアが1つのプロセッサに入っている：マルチコアプロセッサ
  - 例: 8-cores in SPARC64 VIIIfx processor (京computer)
  - それぞれのコアは独立に動作できる
    - 1つのプロセッサは、複数のコアから、複数の 同じ/異なる 命令を同時に実行できる
    - コアどうしの通信・データ交換は:
    - 通信を介して
    - **共有メモリを介して**

どちらでも可能

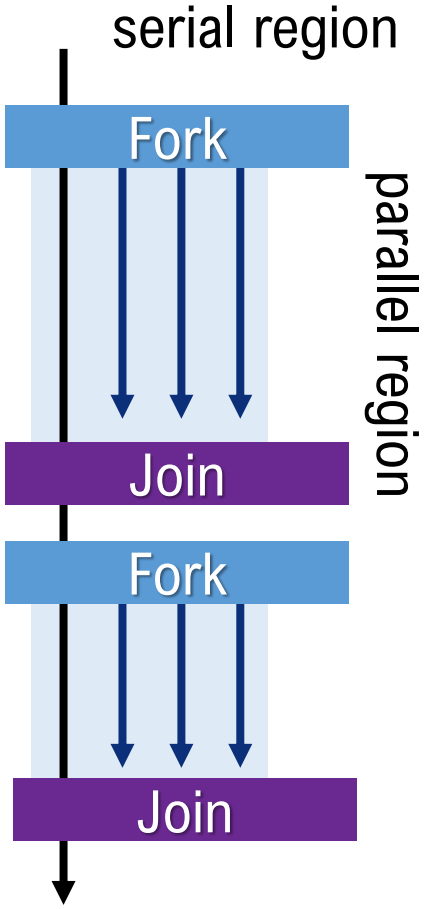


共有メモリシステムのプログラミング: OpenMP

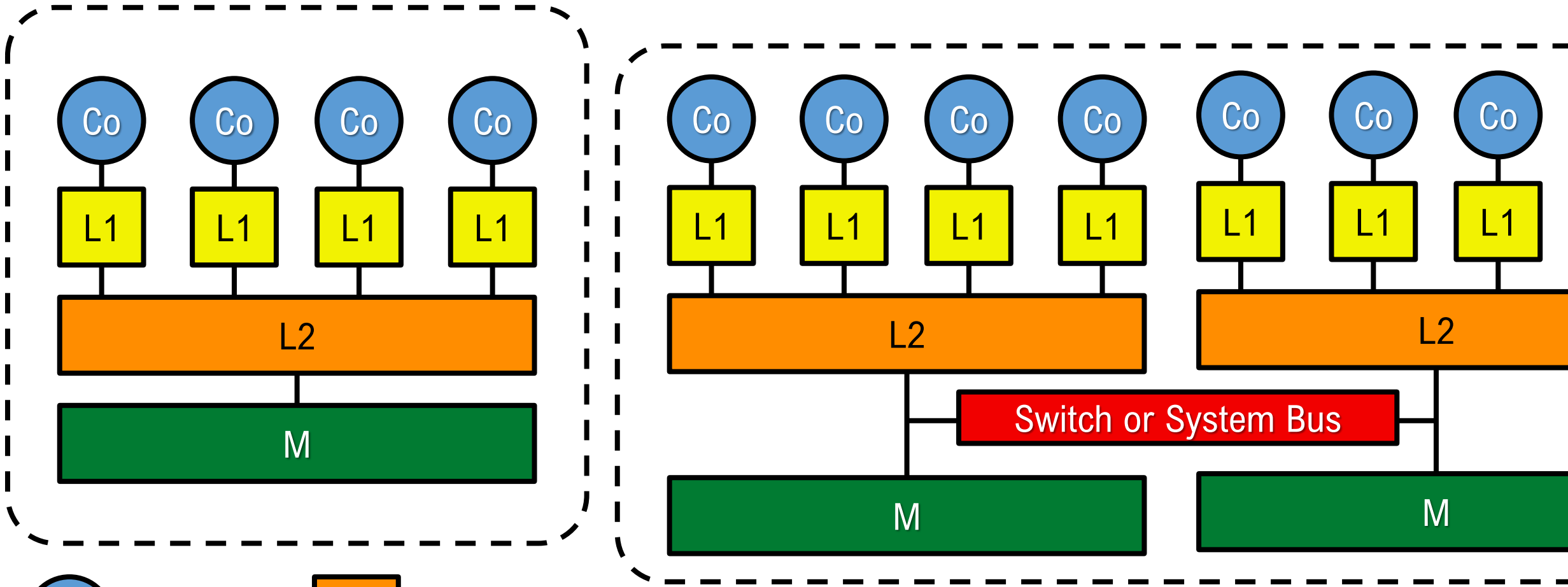
- POSIX Thread等のマルチスレッドプログラミング
- OpenMP: デファクトスタンダード↑を内部的に利用している
- “Fork-Join” 実行モデル
  - 親スレッドが “**Fork**” を呼びだし、新しいスレッドを生成する
    - 親スレッドは演算を継続し、子スレッドも演算を開始する
  - “**Join**” は親子両スレッドから呼ばれ、呼ばれると
    - 子スレッドは終了
    - 親スレッドはすべての子スレッドが終了するのを待つ
    - その後、親スレッドは逐次で演算を継続
- 指示文 (Directive) によるプログラミング
  - 既存のC/C++/Fortranのソースコードの並列化したい部分に特定の指示文を入れる

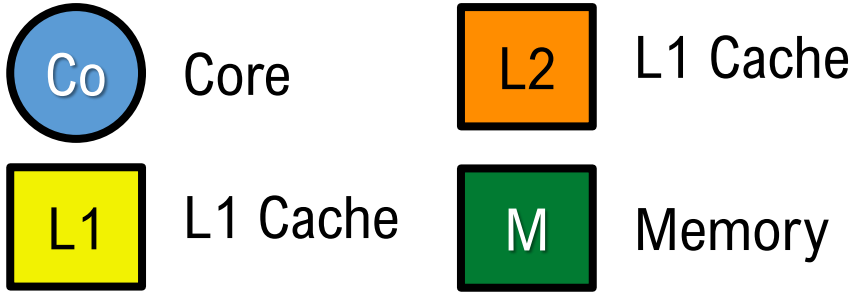
詳しくは午後に。。。

```
PROGRAM TEST
print *,”Thanks”
!$OMP PARALLEL
print *,”Many Thanks”
!$OMP END PARALLEL
END PROGRAM TEST
```



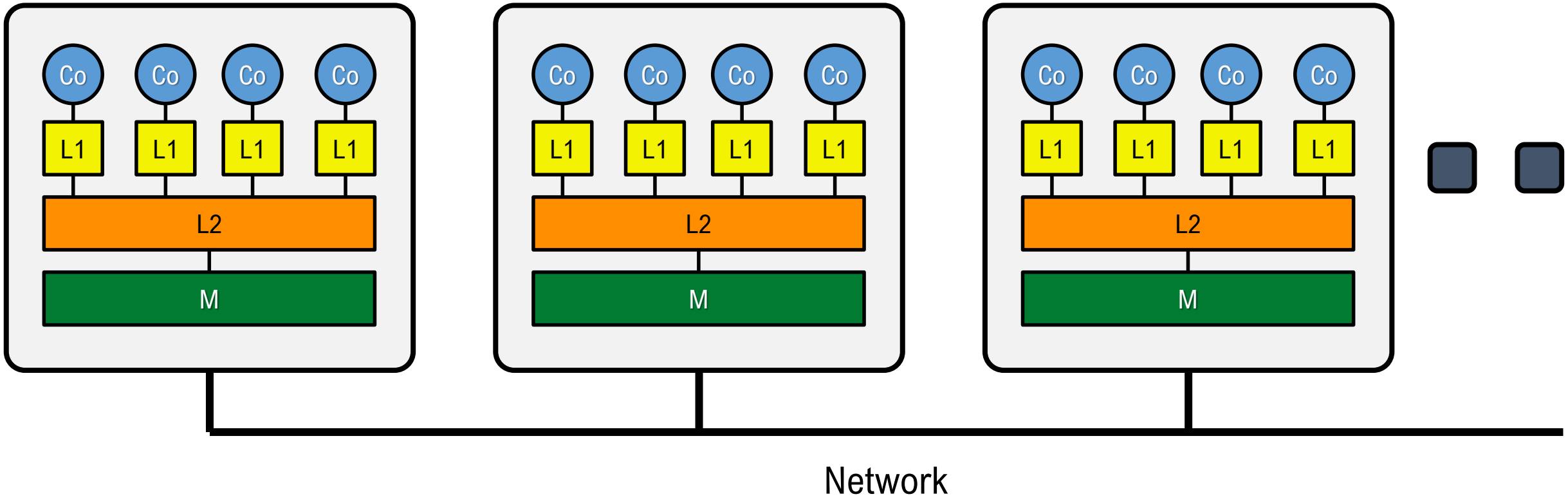
マルチコア： SMP (SMC?) / NUMA





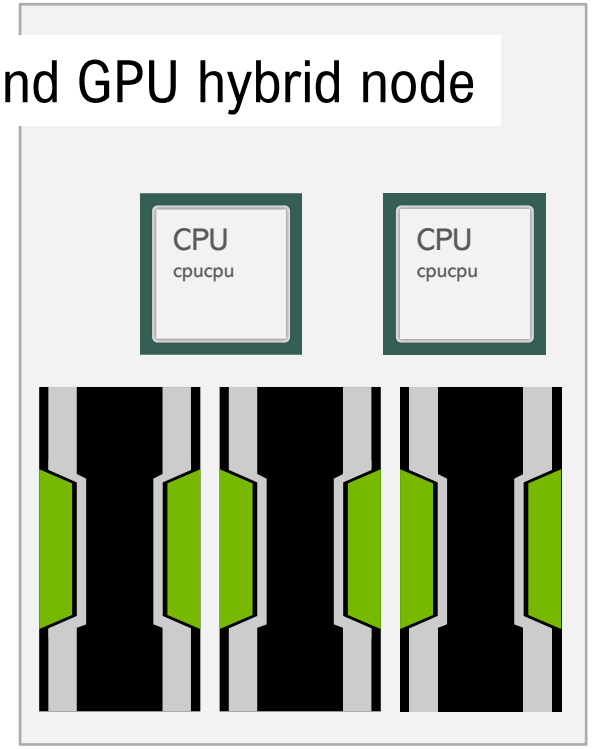
分散並列システムと共有メモリシステムのハイブリッド

- 複数の共有メモリノードをネットワークでつなぐ
- 例：48マルチコアのノードを158,976ノードをつないだスーパーコンピュータ「富岳」



アクセラレータ付きシステム

- 分散メモリ型計算機の各ノードが汎用CPUだけでなく演算性能を加速するハードウェア（アクセラレータ）を伴う
- Graphic processing unit (GPU)
  - もともとはディスプレイに表示する画像をつくるために、特別に設計されたプロセッサ
  - フレームバッファ内の画像生成を高速化するために、メモリを迅速に操作・変換
  - General purpose GPU (GPGPU) と呼ばれる、画像以外の一般的なプログラムで用いられることが提案され、現在ではスタンダードになっている
  - 高速、高電力効率、~~安価、~~安価だった（近年は値上がり
- Field Programmable Gate Array (FPGA)
  - 再構成可能なハードウエア
- 汎用アクセラレータ
  - ClearSpeedなど
- ヘテロジニアスプロセッサ
  - Cell Broadband Engine (CBE)、はじめて1PFLOPSを超えたマシンに搭載
  - プレステにはいってた



マルチコア、メニーコア、GPU、、

	マルチコア	メニーコア	GPU
例	Intel Xeon Platinum 8280	Intel Xeon Phi 7250P	NVIDIA Tesla V100
コア数	28	72	5120
周波数	2.7 GHz	1.2GHz	1.53 GHz
性能	605 GFLOPS	1.1 TFLOPS	7.8 TFLOPS
メモリ容量	1 TB	16 GB	6 GB
メモリ性能	141 GB/s	450 GB/s	900 GB/s
電力	205 W	200 W	300 W
プログラム	C, OpenMP, MPI	C, OpenMP, MPI	CUDA　(C, Fortran)
モデル	MIMD	MIMD	STMD

近代HPC並列システムの概要

- 複数のノードをネットワークで接続
  - 1つのノードに複数のプロセッサ＋アクセラレータ
    - 1つのプロセッサに複数のコア
      - コア内に複数のSIMD演算機
        - 演算がパイプライン化
    - アクセラレータ

} コンパイラがよろしくやってくれる、が、さらなる最適化も可能。

- それぞれの並列アーキテクチャ向けのプログラミング必要
- システム全体を使うためにはハイブリッドプログラミングが必要

- 詳しくは午後のプログラミングの話を聞きましょう

もくじ

- 並列計算機アーキテクチャ
  - 分散メモリ, 共有メモリ, SMP, NUMA...
- **並列処理ネットワーク**
- 実システムの紹介

並列処理ネットワーク（相互結合網）

- 役割
  - 分散メモリアーキテクチャに基づく並列計算機における明示的なデータ交換 (例：MPI通信)
  - CC-NUMAアーキテクチャ (Cache Coherent NUMA)に基づく並列計算機におけるデータ及び制御メッセージの転送
- 特性／分類
  - static (direct) / dynamic (indirect)
  - diameter (distance)
  - degree (number of links)
- 性能指標
  - Throughput (単位時間あたりにどのくらいのデータ量を転送できるか)
  - Latency (転送要求を出してからどのくらいの時間でデータがやって来るか)

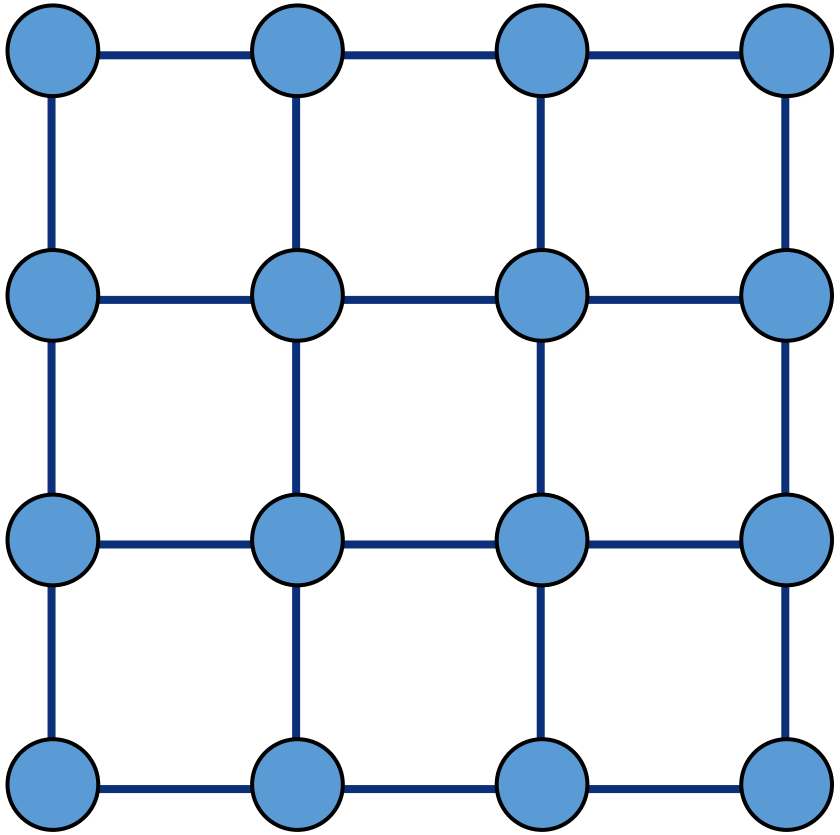
直接網（静的網）

- ノード同士が直接接続されているネットワーク
- ノード以外のスイッチを介さない
- 代表的な直接網トポロジ
  - メッシュ
  - トーラス
  - ハイパーキューブ
  - など

  （メッシュ、トーラス）：1次元、2次元、、、n次元で構成可能
- スイッチを介さずノード同士が直接接続されるため、低遅延かつ高スループット
- ネットワーク直径 （Diameter, ネットワーク内の任意の2つのノード間の最短ホップ数の長さの最大値）は、長くなりがち

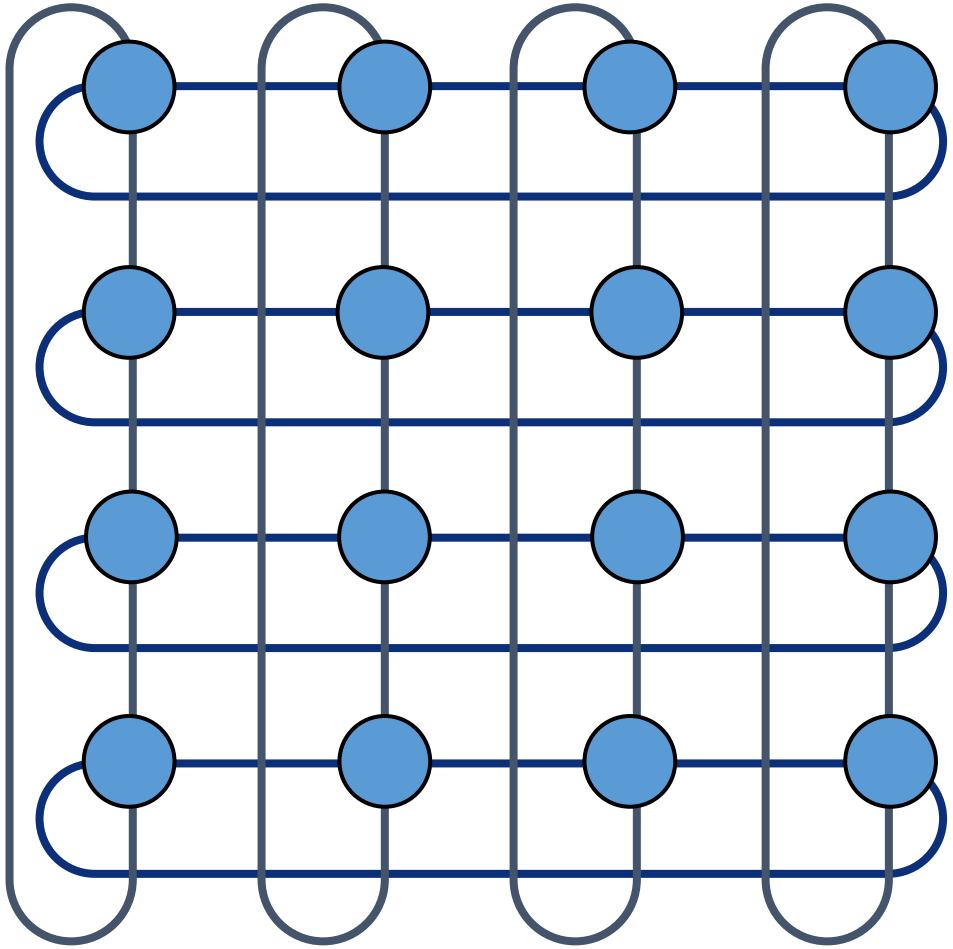
直接網（静的網）メッシュ

- 各ノードが（多くは）格子状の隣接ノードと直接接続
- 直径は n(k-1)



直接網（静的網）トーラス

- メッシュの端を接続してループ状に

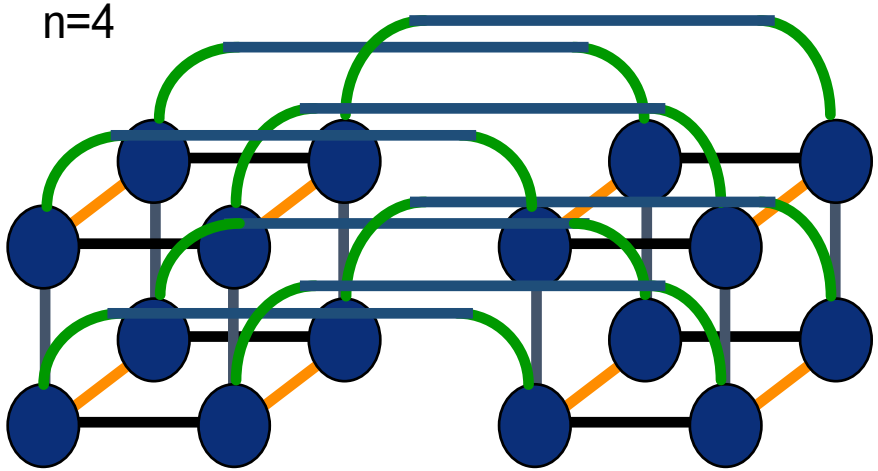


直接網（静的網）ハイパーキューブ

- 2進数表現で1ビット違いのノードどうしが直接接続

- ネットワーク半径は小さめ
  - ノード数 N に対して d = log2 N
- 次元が増えると次数が増える
  - 配線が複雑

2x2x2



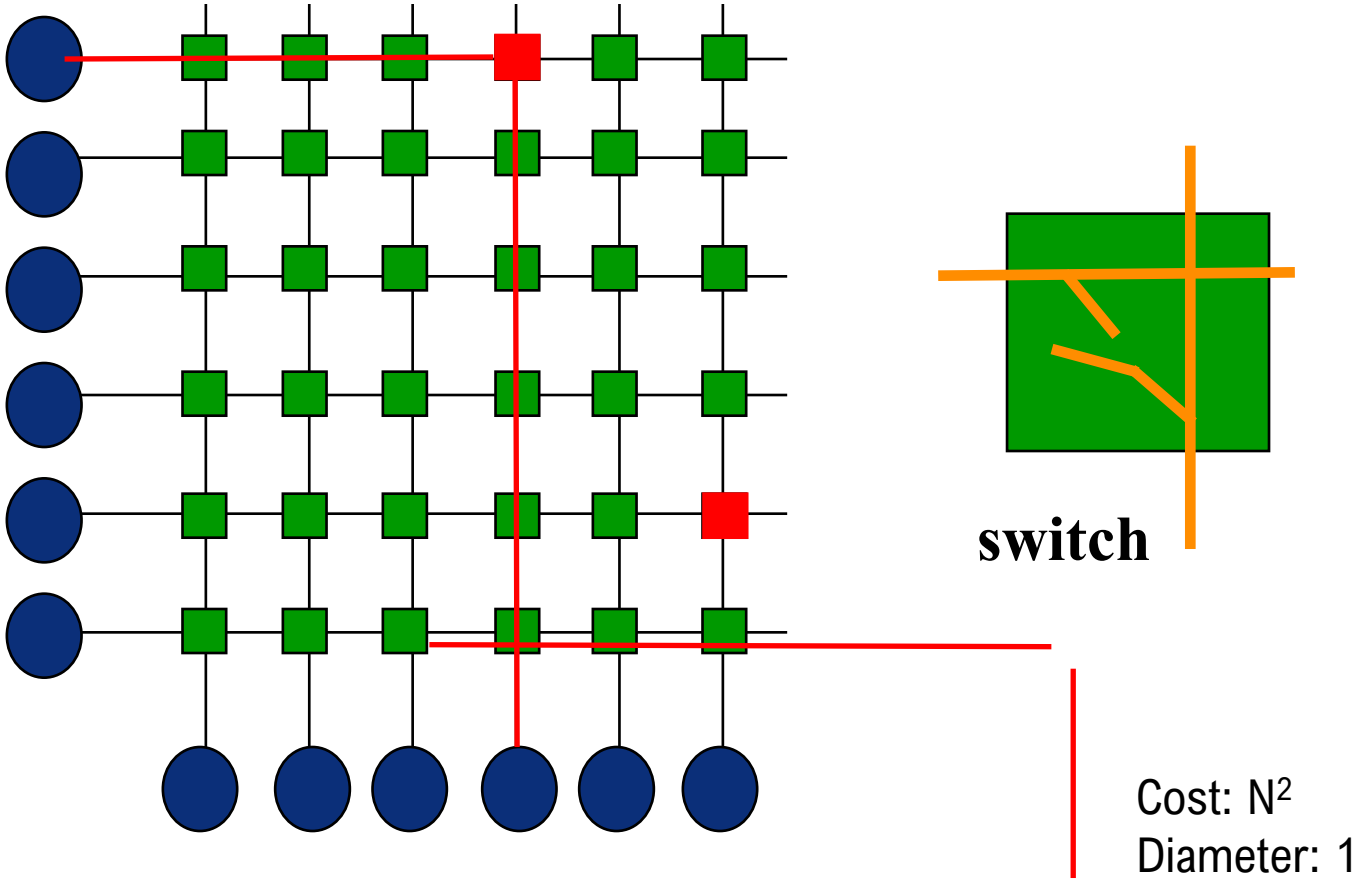
Cost: N (=2^n)
Diameter: n

間接網（動的網）

- ノードからは一般的に1本のリンクのみ（例外あり）
- 各ノードからのリンクを１つ以上のスイッチで結合してネットワークを形成
- スイッチでのルーティングが基本
- 代表的な間接網
  - Crossbar
  - MIN (Multistage Interconnection Network)
  - HXB (Hyper-Crossbar)
  - Tree (Indirect)
  - Fat Tree

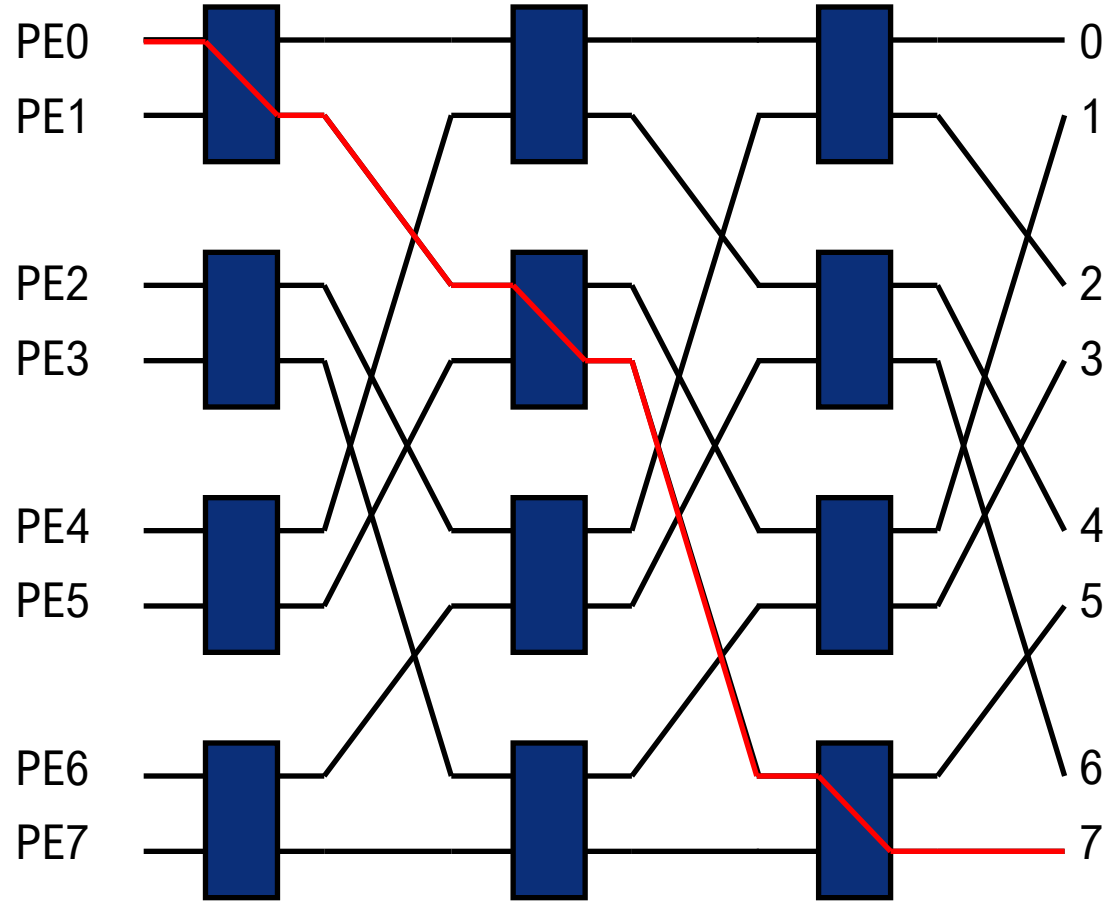
クロスバー Crossbar

- 入力ポートと出力ポートが格子状に並ぶ
- 各交差点（クロスポイント）にスイッチがあり、必要に応じて入力から出力へ接続可能
- 全ノード間通信を1ホップで実現可能
- 規模が大きくなるとコストが増大

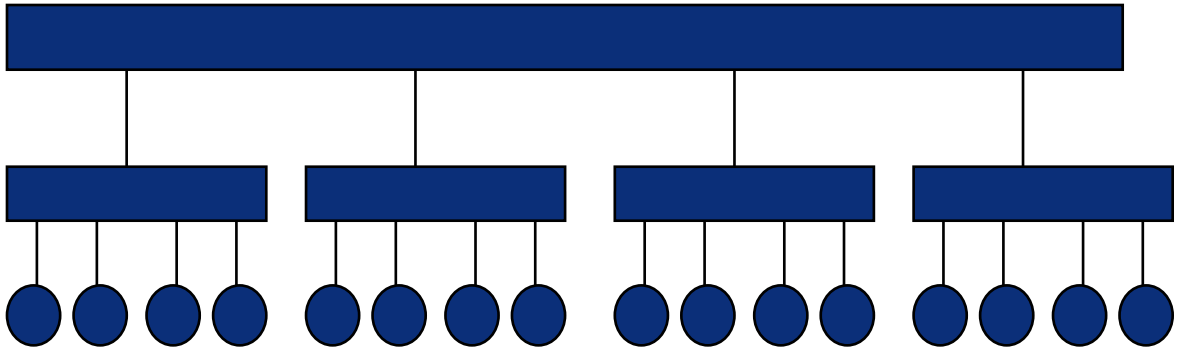


MIN (Multi-stage Interconnection Network)

- 複数段階のスイッチを経由してノード間通信を行うネットワーク構造



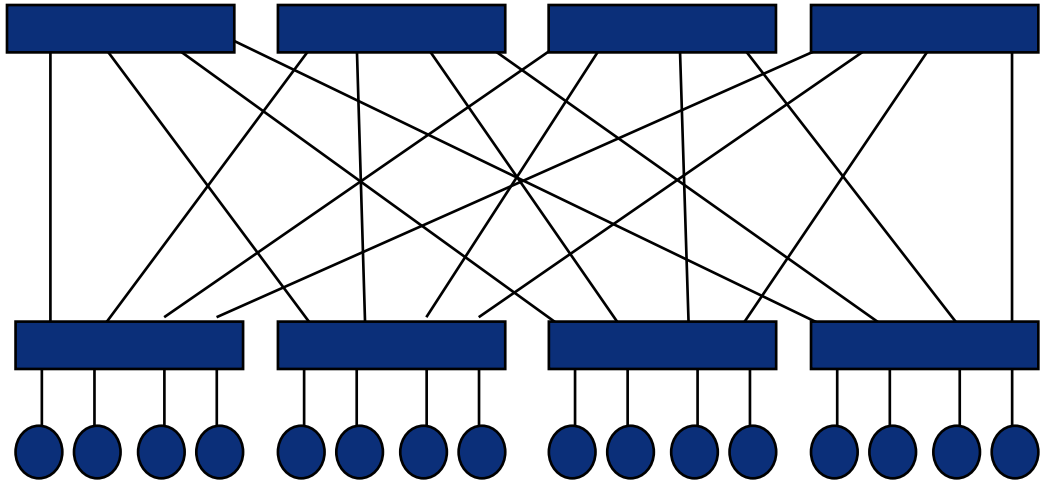
Cost: NlogN
Diameter: logN



Tree

Cost: N/k
Diameter: $2\log_k N$

Fat Tree



Cost: $N/k\log_k N$
Diameter: $2\log_k N$

並列処理ネットワークの性能指標

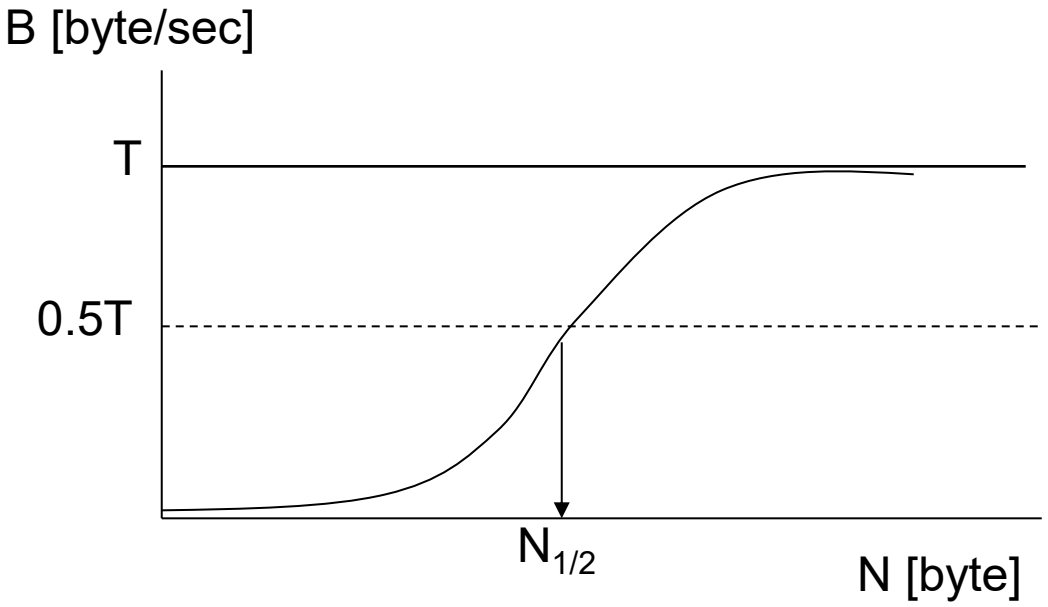
- Throughput（スループット）
  - リンクあるいはネットワーク全体の単位時間当たりのデータ転送性能
  - 単位：[byte/sec]（あるいは [bit/sec] ）

- Latency（遅延時間）
  - 狭義：転送すべきデータの先頭がsourceを出発してからdestinationに到着するまでの時間（ここではこれに従う）
  - 広義：転送すべきデータ全体がsourceを出発してからdestinationに到着するまでの時間
  - 単位：[sec]

ネットワーク転送性能とメッセージ粒度

- ネットワークリンク上で他のメッセージとの衝突がないとする。
  T [byte/sec]のスループットとL [sec]の遅延時間を持つネットワーク上で、*N* [byte]のメッセージを完全に転送し終わるまでの時間*t* [sec]と、有効バンド幅*B* [byte/sec]は以下のようになる

$$t = L + N/T \qquad B = N / t$$



ここで、理論ピークバンド幅(T)の半分の0.5Tの性能が出るメッセージ長を$N_{1/2}$ (N-half「半性能長」)と表す。
理論的には

$$N_{1/2}[byte] = L \times T$$

となる。
$N_{1/2}$は「この長さ以下ではLがdominantで、この長さ以上ではTがdominantである」ことを表し、これが小さい程、短いメッセージの通信に強いネットワークということになる。

もくじ

- 並列計算機アーキテクチャ
  - 分散メモリ, 共有メモリ, SMP, NUMA...
- 並列処理ネットワーク
- **実システムの紹介**

実際の並列計算機概観

- システムの分類
  - MPP (超並列計算機)
    - 筑波大/日立 CP-PACS (SR2201)
    - LLNL/IBM Sequoia
    - ORNL/Cray Titan
    - 理研/富士通 富岳
  - 大規模並列ベクトル計算機
    - NEC 地球シミュレータ
  - スカラ並列計算機（クラスタを含む）
    - 筑波大/日立/富士通 PACS-CS
    - 筑波大・東大・京大/Appro・日立・富士通 T2K
  - アクセラレータ付ハイブリッド計算機
    - LANL/IBM Roadrunner
    - 東工大/HP (SGI) TSUBAME3.0
    - 筑波大/NEC Cygnus
    - NUDT Tianhe-2

TOP500リスト

- 全世界のスーパーコンピュータの性能を１つのベンチマーク性能値で定量化し順位付けを行ったリスト
- Linpack（多次元連立一次方程式のガウスの消去法による直接求解）ベンチマークの性能（FLOPS）
- 毎年6月と11月の２回、リストを更新
  - http://www.top500.org
- １つの数値で順位付けするためわかりやすい
- 問題の特徴として
  - ガウスの消去法のカーネル部分は小規模の行列×行列演算の帰着可能、キャッシュアーキテクチャでのデータ再利用性が非常に高い
  - ネットワーク性能は比較的低くても性能に大きく影響しない
- メモリバンド幅やネットワークバンド幅が比較的低くても性能が出るため、「本当にHPCベンチマークとして適当か」という議論はあるが、現時点では最も知られている
- ほかのベンチマークとしてはHPCGなどもある

Green500

- TOP500の中で、電力あたりの性能（MFLOPS/W) をランク付けしたもの。
- 近年、電力供給が大規模並列計算機のボトルネックの一つといわれており、注目されている。
- 毎年6月と11月の 2 回、リストを更新
  - http://www.green500.org/
- ベンチマークとしてTOP500の値を用いているため、TOP500と同様の問題がある。
- TOP500に入っていることが条件であるが、その電力規模は大きく異なる(10MW – 30kW)。一般的に小電力システムのほうが電力あたりの性能を高めやすいので、単一の指標で良いか議論がある。そのため、大規模運用しているマシンの中で優秀なマシンを特別に表彰したりしている。

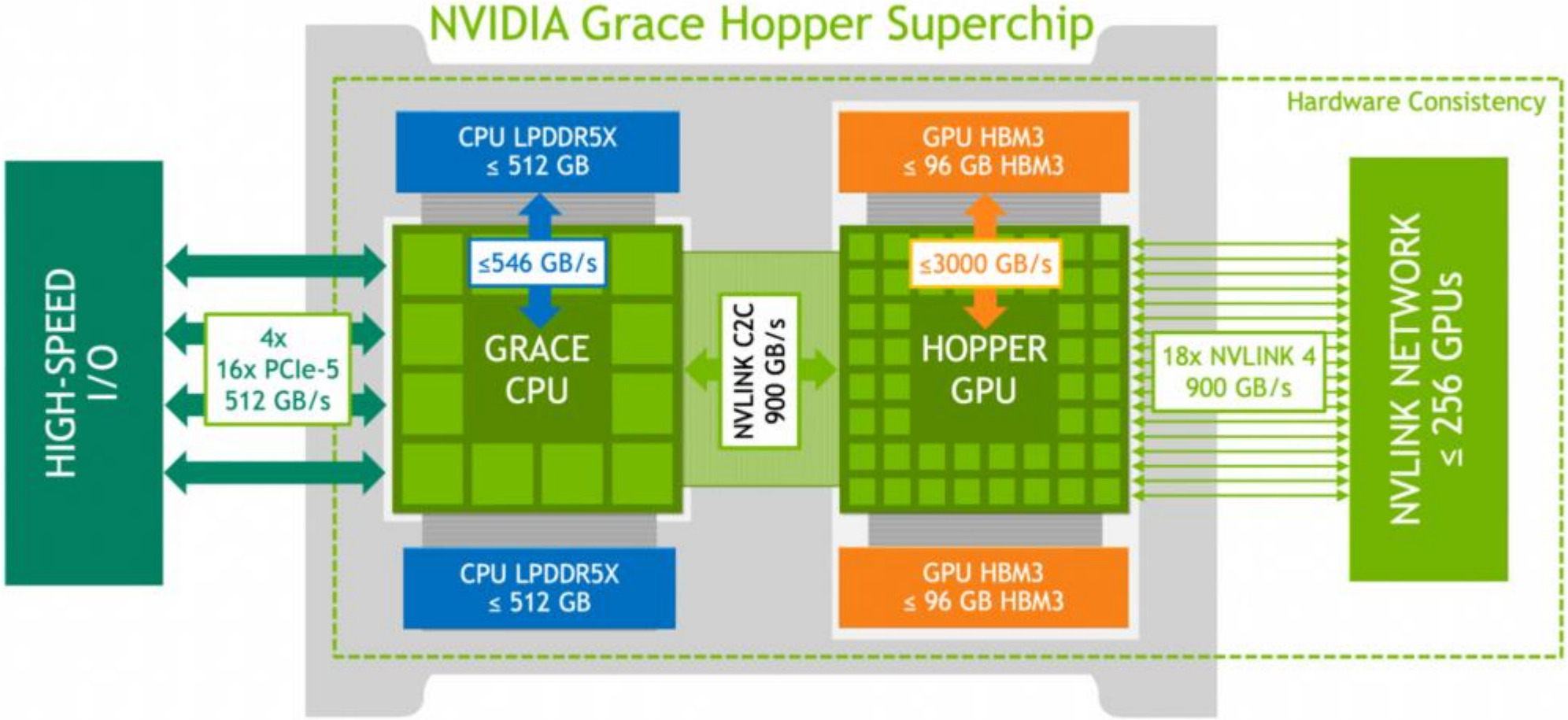
Miyabi

- 最先端共同 HPC 基盤施設スーパーコンピュータシステム JCAHPC 筑波大＆東大
- Fujitsu
- Miyabi-G (GPUあり）と
  Miyabi-C (CPUのみ）がある
- Infiniband NDR200
- TOP500#28 2024/11 (Miyabi-G)
  - 日本の国立大学では最高性能
  - 富岳についで国内2位
  - Peak 72.80 PFOPS
  - HPL 46.80 PFLOPS

https://www.cc.u-tokyo.ac.jp/supercomputer/images/Miyabi.jpg

Miyabi NVIDIA GH200 Grace Hopper Superchip

- CPU (Grace) と GPU (Hopper) を 1つのノードに



https://developer.nvidia.com/ja-jp/blog/nvidia-grace-hopper-superchip-architecture-in-depth/

Specification of Miyabi

		Miyabi-G	Miyabi-C
総理論性能		78.8 PFLOPS	1.29 FPLOPS
総ノード数		1120	190
総メモリ容量		220.0 TiB	23.75 TiB
インターコネクト		InfiniBand NDR (200Gps)	
ネットワークトポロジー		Full-bisection Fat Tree	
共有ファイルシステム	システム名	Luster DDN EXAScaler	
	サーバ (OSS)	DDN400NVX2	
	サーバ(OSS)数	10	
	ストレージ容量	11.3PB	
	ストレージデータ転送速度	1.0TB/s	

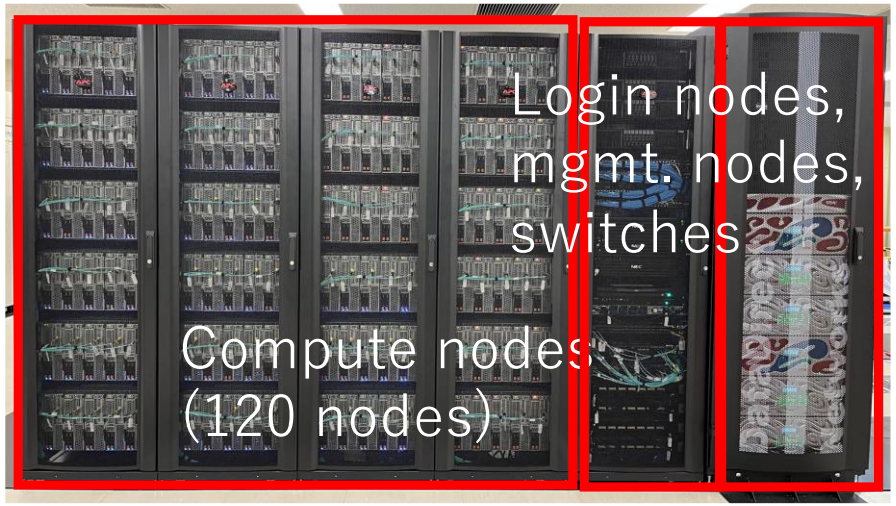
Specification of Miyabi Nodes

		Miyabi-G	Miyabi-C
マシン名		Supermicro ARS-111GL-DNHR-LCC	FUJITSU Server PRIMERGY CX2550 M7
CPU	プロセッサ名	NVIDIA Grace CPU	Intel Xeon Max 9480
	プロセッサ数 (コア数)	1 (72)	2 (56+56)
	周波数	3.0 GHz	1.9 GHz
	理論演算性能	3.456 TFLOPS	6.8096 TFLOPS
	メモリ容量	120 GB	128 GiB
	メモリ帯域幅	512 GB/s	3.2 TB/s
GPU	プロセッサ名	NVIDIA Hopper H100 GPU	
	メモリ容量（単体）	96 GB	
	メモリ帯域幅（単体）	4.02 TB/s	
	理論演算性能（単体）	66.9 TFLOPS	
	搭載数	1	
	CPU-GPU間接続	NVLink C2C Cache-coherent 片方向450GB/s	

Pegasus

- 筑波大学

  - 4th Gen Intel Xeon SP, NVIDIA H100 Tensor Core PCIe GPU , Intel Optane persistent memoryを搭載し,
    ビッグデータとAIを強力に推進する



Parallel File System

Specification of Pegasus

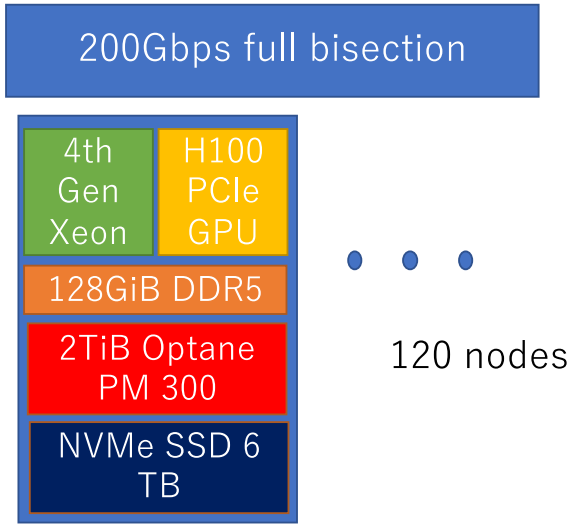
Pegasusの設計目標

- 大容量メモリでかつ高性能ストレージである不揮発性メモリを活用し，大規模データ分析やビッグデータAIを高速化
- 大規模データ分析やビッグデータAIの新規アプリケーション，システムソフトウェア研究の新分野を開拓

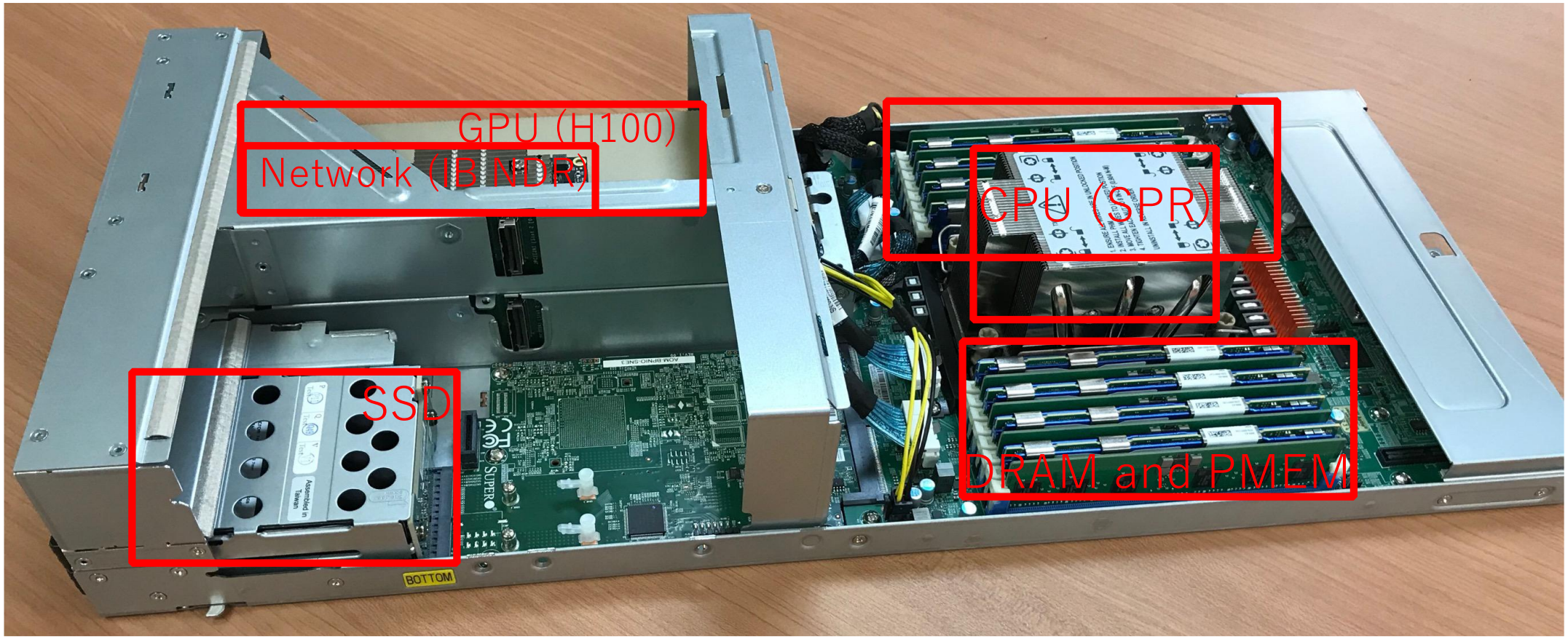
- 2022年第4四半期に導入
- Total Performance
  - 120 nodes, 6.5 PFlops, 240 TiB Pmem
- ノード仕様
  - 3.2 TFlops Intel Xeon Platinum 8468 (codenamed Sapphire Rapids)
  - 51 TFlops NVIDIA H100 PCIe GPU (80GB mem)
  - 128 GiB DDR5 DRAM (282 GB/s)
  - 2 TiB Optane DCPMM 300 series
  - 6 TB NVMe SSD (7 GB/s)
- Interconnection Network
  - NVIDIA Quantum-2 InfiniBand platform NDR200 (200 Gbps) full bisection
- Parallel File System
  - 7.1 PByte DDN EXAScaler (40 GB/s)

NEC LX B1000E Blade Enclosure

NEC LX 102Bk-6



Pegasusのノード



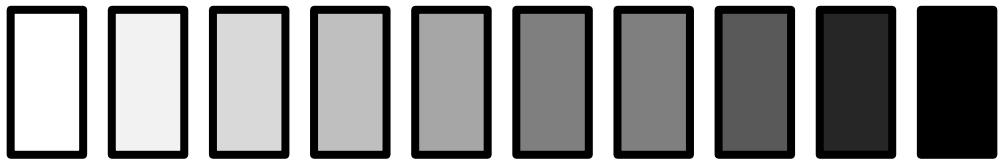
富岳

- 理化学研究所計算科学研究センター
- 158,978のCPUからなる超並列システム
- 2020年11月のTop500を含む4つのランキングで1位

Ranking	Performance	2nd System Performance
TOP500	442.01 PFLOPS	148.6 PFLOPS (3.0)
HPGC	16.00 PFLOPS	2.9 PFLOPS (5.5)
HPL-AI	2.00 EFLOPS	0.55 EFLOPS (3.6)
Graph500	102.05 TTEPS	23.75 TTEPS (4.3)

Specification of supercomputer Fugaku (& K)

Specification of supercomputer Fugaku	
CPU Technology	A64FX based on **Armv8.2-A** SVE
# of Cores in CPU	48+2/4 cores
CPU Clock	2.0 - 2.2 GHz
Theoretical Peak	
Node Cache	
Memory Technology	
Memory Capacity	
Memory Bandwidth	
Network	
# of nodes	



10 racks ≒ K computer

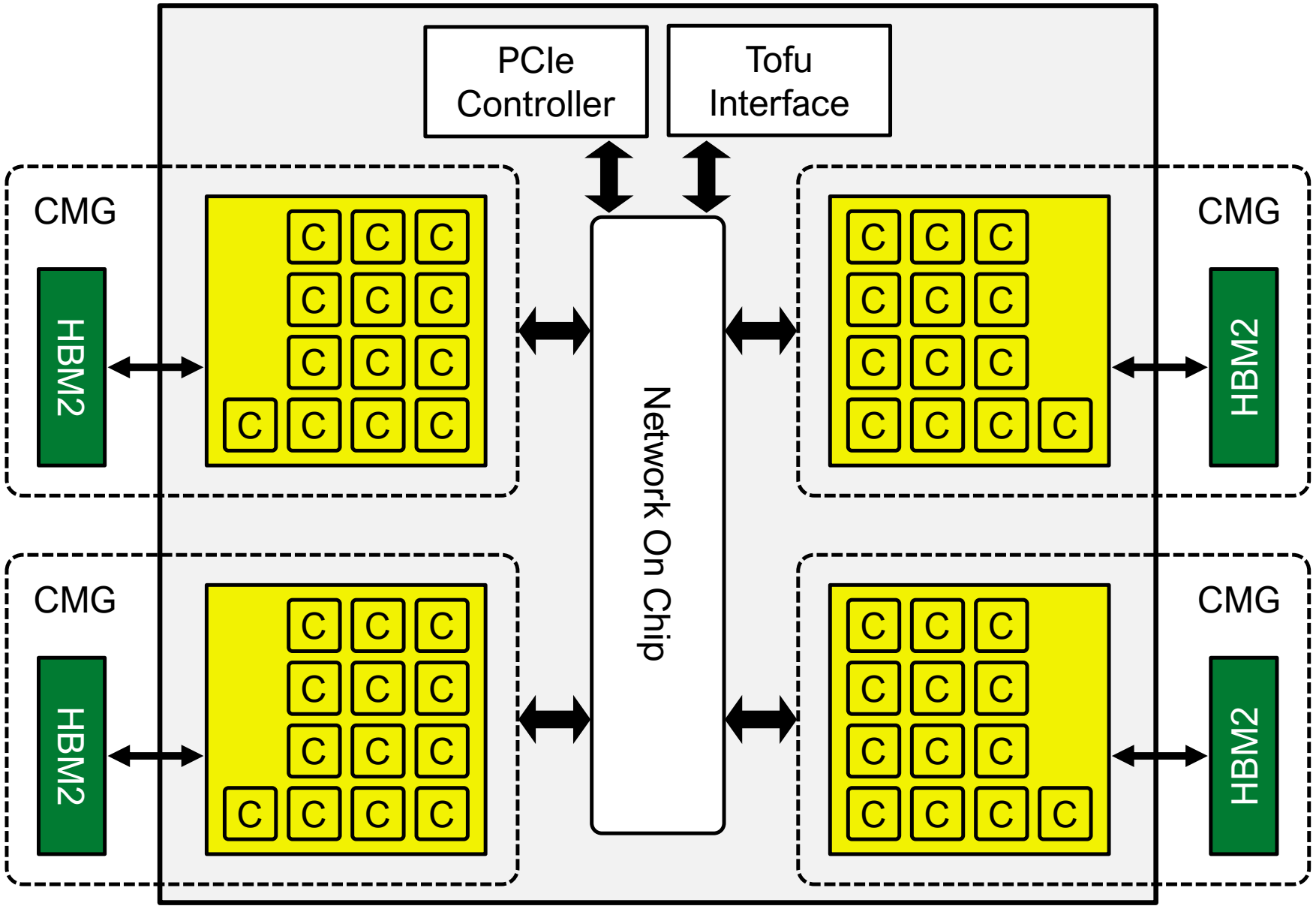
Specification of K computer	
CPU Technology	**SPARC64** VIIIfx
# of Cores in CPU	8 cores
CPU Clock	2 GHz
ak	128Gflops
ne	L1 32 KB / core
	L2 6 MB / node
gy	DDR3
ty	16 GB
th	64 GB/s
rk	Tofu 6-dimentional torus, 5GB/s (bi-direction)
es	82944 (88128, including IO-nodes)

K computer

A64FX: Node Overview

C: Core
CMG: Core Memory Group
HBM: High Bandwidth Memory

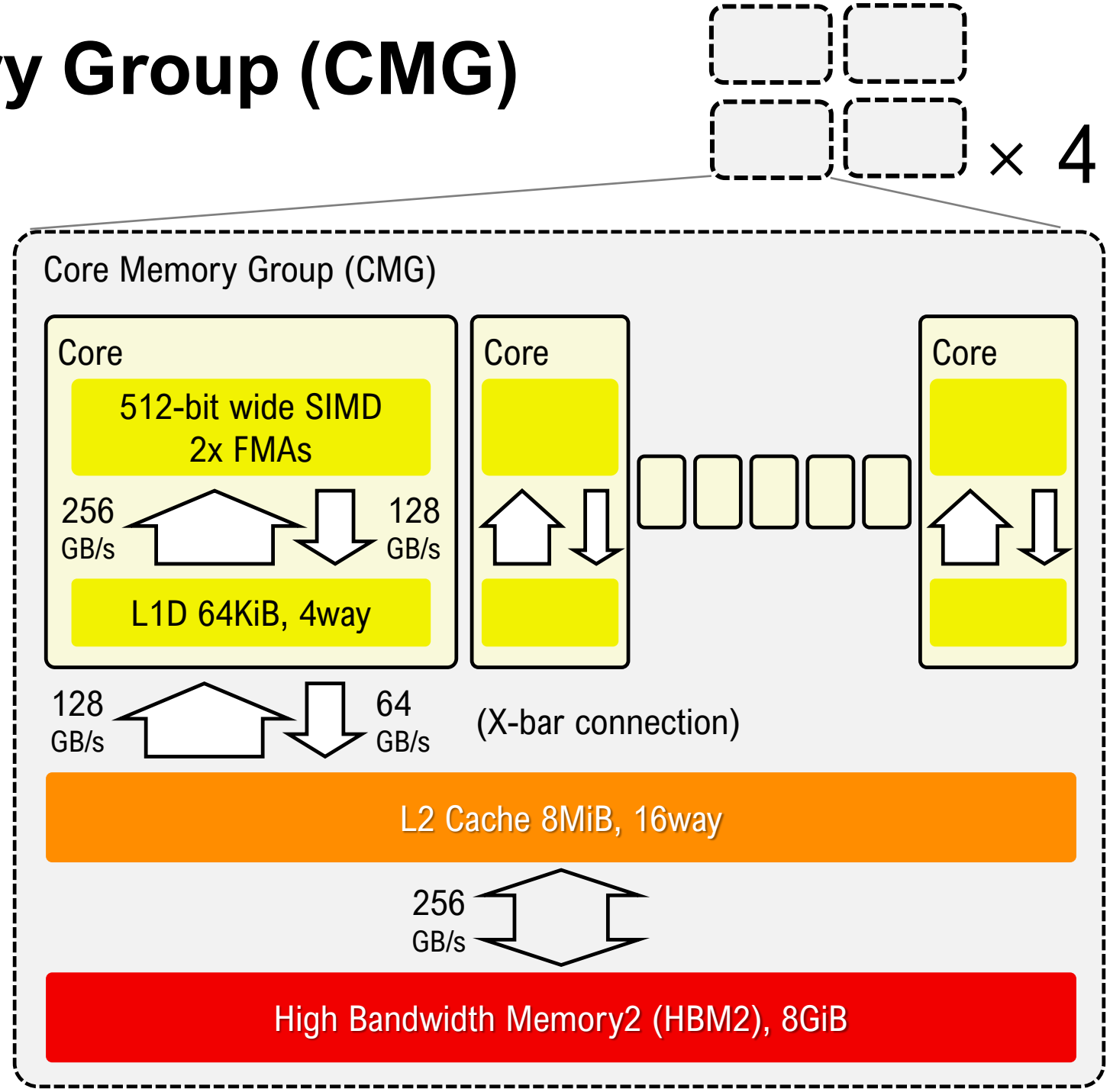
- SVE 512-bit wide SIMD
- FP64/FP32/FP16
- 12+1 cores in a CMG
  - an assistant core in each CMG
- 4 CMG in a Node
  - 48 + 2/4 core
- HBM2 32GiB/Node
  - 8GiB/CMG
  - 1024 GB/s
- Ring Network, 128 GB/s x 2
- TofuD NIC, 6.8 GB/s x 6
  - 6 simultaneously transmit directions



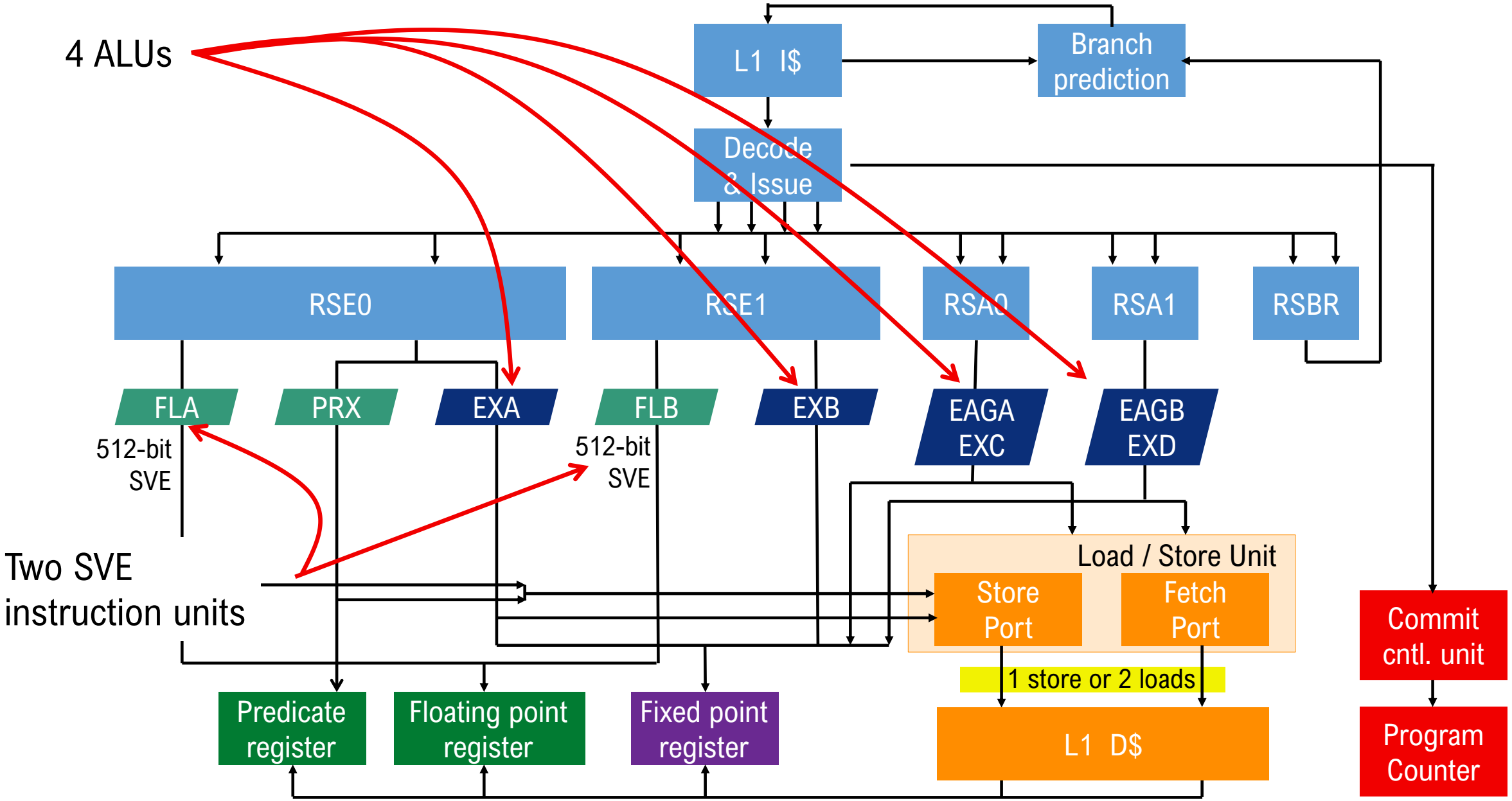
A64FX: Core Memory Group (CMG)

- 2x 512-bit wide SIMD, 4x ALUs, Predicate Operation
- 2x 512-bit wide SIMD load or 512-bit wide SIMD store
- L1D cache (/core) : 64 KiB, 4 ways, “Combined Gather” on L1
- L2D cache (/CMG): 8MiB
  - X-bar connection in a CMG
- 4 CMGs support cache coherency by ccNUMA with on-chip directory (256 GB/s x 2 for inter-CMGs)
- Memory controller for HBM2

※ L1/L2 cache bandwidths are for 2.0GHz



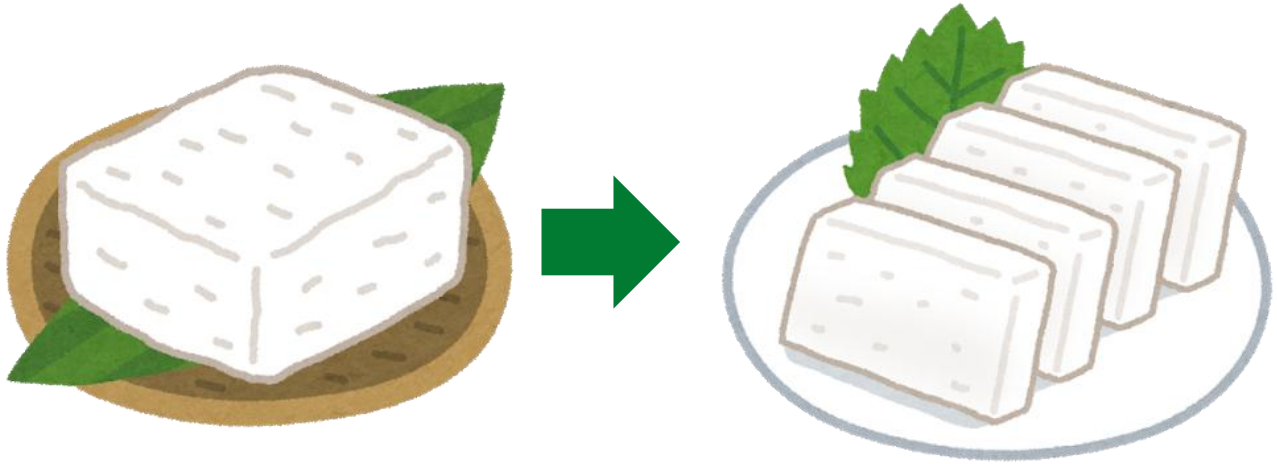
A64FX: Core Overview



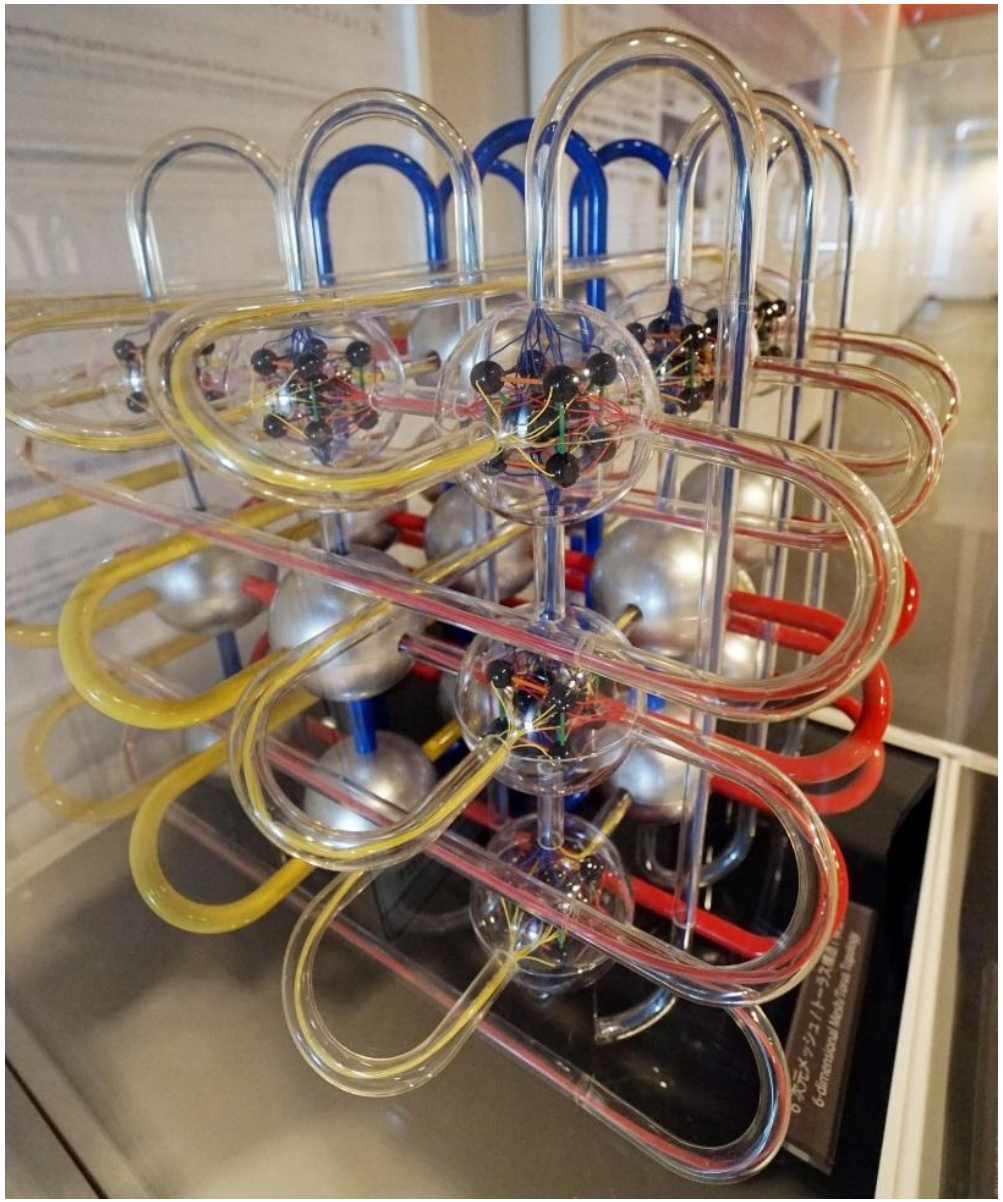
富岳のインターコネクト: Tofu Interconnect D (TofuD)

- Tofu interconnect series
  - 6次元メッシュトーラスハイブリッドネットワーク
  - 任意の**トーラス**トポロジを 48x36x48 の範囲で選択できる

  ※ 通常のトーラスネットワークは切り抜くと**メッシュ**になってしまう



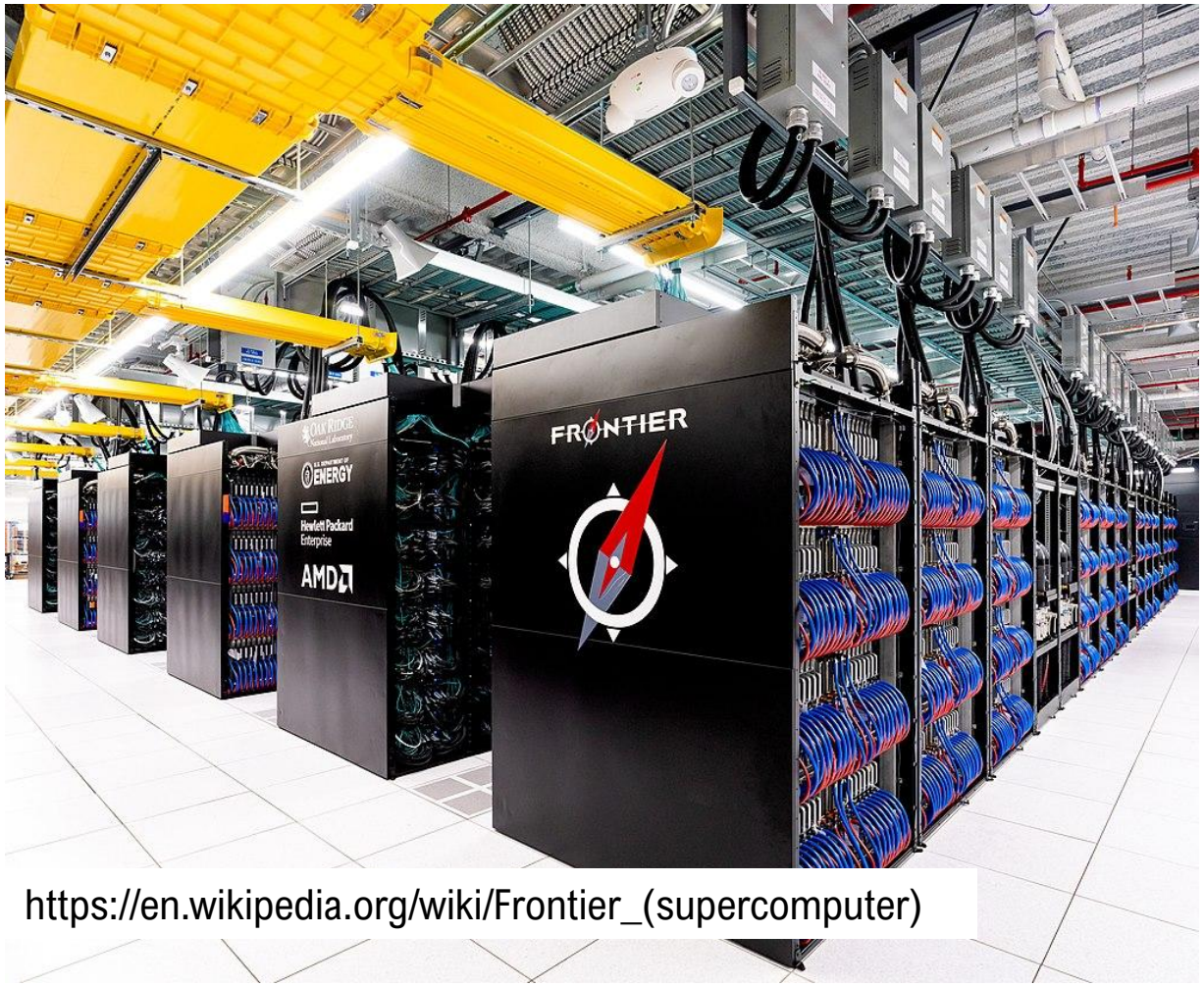
豆腐は切りやすく、切っても切っても豆腐
TofuDはTorus、切ってもTorus



フロンティア Frontier

- オークリッジ国立研究所 (Oak Ridge National Laboratory)
- 1st of the world (June 2022)
- HPE Cray EX system
- node
  - 1 AMD EPYC Milan processor
    - 64 core 2 GHz CPU
  - 4 AMD Instinct MI250x accelerators
- 9408 nodes

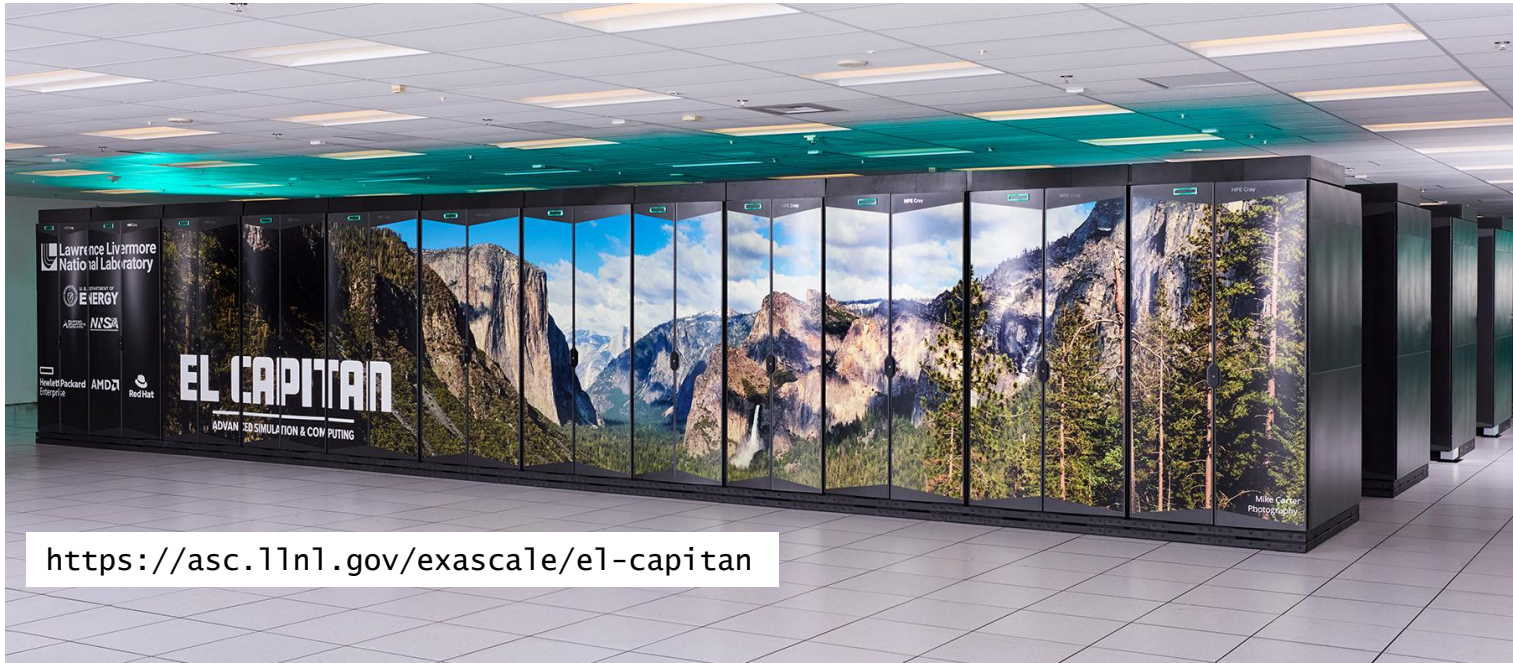
1.102 **Exa** FLOPs (HPL benchmark)
2 Exa FLOPs (theoretical peak)



https://en.wikipedia.org/wiki/Frontier_(supercomputer)

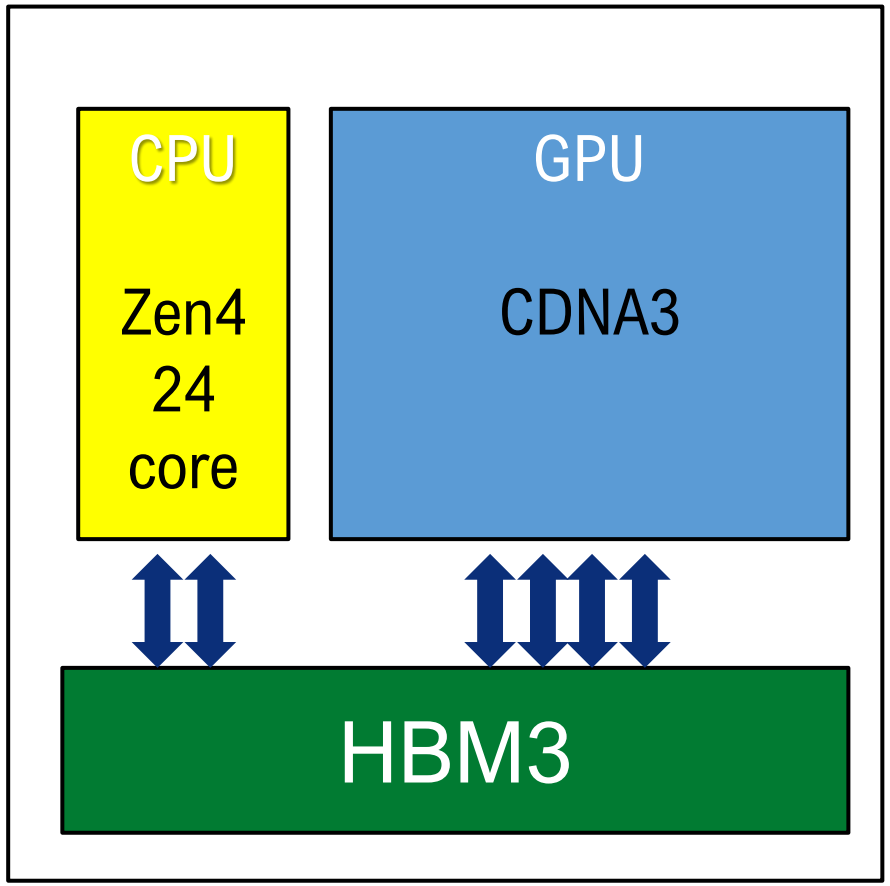
エル・キャピタン El Capitan

- 米ローレンスリバモア国立研究所 (Lawrence Livermore National Laboratory)
- 1st of the world (June 2025)
- HPE Cray EX255a
- node
  - AMD MI300A
    - CPUとGPUがメモリをシェアするNUMAノード
- 44,544 nodes
  1.742 **Exa** FLOPs
  (HPL benchmark)
  2.746 Exa FLOPs
  (theoretical peak)
- Slingshot-11



https://asc.llnl.gov/exascale/el-capitan

エル・キャピタン El Capitan のノード MI300A



- GPU
  - 14,592ストリームプロセッサ
  - 912マトリックスコア
  - 228演算ユニット
    - 倍精度マトリックス(FP64)ピーク性能 122.6TFlops
    - 単精度 (TF32 マトリックス) ピーク性能 490.3 TFLOPs
    - 半精度 (FP16) ピーク性能 980.6 TFLOPs
- CPU
  - 24コア
  - 倍精度浮動小数点理論ピーク演算性能 1.42 TFlops
- これが1ノードに4セットはいっている

並列処理システムの動向

- MPPは徐々に衰退（特定マシンのみ躍進）
- コモディティ化が進む（クラスタの台頭）
  - コモディティなスカラープロセッサ（64bit IA32=x64）
  - コモディティなネットワークとスイッチ
    - InfiniBand (EDR 100Gbps、高級機器だったが徐々に価格低下）
- 全体的に、演算性能：メモリ性能：通信性能のバランスが悪化
  - 演算性能はプロセッサのmulti-core化等により順調に向上
  - メモリ性能（バンド幅）は相対的に低下（プロセッサが速すぎる）
  - 通信性能は段階的に上がっていく（IB等）
  - プロセッサコストはO(N)だがネットワークコストはO(N log N)程度なので相対的にシステム価格を圧迫
  - 結果的に並列処理効率を上げるのが難しくなってきている。より一層のアルゴリズム、ソフトウェア上の工夫が必要。
- Exa FLOPS マシンが続々？搭乗
  - 2022年6月についにエクサフロップスを達成するマシンが登場！
  - 2025年6月現在で3台
  - 性能/電力の100倍程度の向上が必要
  - 1000万並列を効率よく利用できるアルゴリズムの開発

まとめ

- 並列処理システム／アーキテクチャ
  - 逐次プロセッサ（コア）性能の限界により、全体性能は並列処理に頼らざるを得ない
  - 性能を維持しつつ拡張性（scalability）を確保
  - 分散メモリ / 共有メモリ
- 並列処理ネットワーク
  - scalabilityが最も重要
  - 以前はMPP向け、現在はcommodity networkの充実によりfat-treeでかなりの規模が可能
  - 2つの性能メトリック：throughput & latency
- 並列処理システムの実際
  - 2025年6月で，エクサFLOPSを超えるマシンが3台
  - 基本は分散メモリシステムだがmulti-coreの一般化によりハイブリッドが基本
  - アクセラレータが注目されている (GPUがポピュラー)

課題

2024/06のTop500の1〜10位の計算機について、3つ選んで以下の問いに答えよ。ただし、CPUもしくはGPUのいずれかが同じシステムを2つ以上選ぶことができない。

1. 各計算機の最大性能について、コアあたり、チップあたり、ノードあたり、システム全体について示せ。コアあたりについては、その根拠を示すこと。ただしノードがヘテロジニアスな構成の場合は各構成要素についても示すこと。
2. 各計算機のメモリシステム、ネットワークの構成について、その特徴を比較して述べよ。

Q&A