コンピュータアーキテクチャA

天野英晴

hunga@am. ics. keio. ac. jp

コンピュータアーキテクチャAとB

- コンピュータアーキテクチャA
  - 並列コンピュータとそのプログラミングを学ぶ
    1. 並列コンピュータとは？
    2. 共有メモリ型コンピュータ
    3. Open MPの演習
    4. メッセージパッシング型コンピュータ
    5. MPIの演習
    6. GPUと他のアクセラレータ
    7. CUDAの演習
    8. 試験

成績評価

- 4回の座学での演習　5%×４
- 3回のプログラミング演習　10%×３
  - 今回の目的は「体験」なのであまり難しいことはやらない
- 試験　50%
  - 試験はあまり好きではないのだが、理解度を見るためやっときたい
  - 5月28日か6月1日にやる
  - そんなに難しい問題は出ない
  - 過去問がないので厳しいかも。。。。

コンピュータアーキテクチャB

- コンピュータアーキテクチャAとは完全に独立している
- どんどんサマースクール、国際インターンに行って！
  - CPUの高速化テクニックを学ぶ
    1. MIPSのマイクロアーキテクチャの復習
    2. MIPSのVerilog記述の復習
    3. パイプライン処理その1
    4. パイプライン処理の演習
    5. パイプラインハザード
    6. パイプラインハザードの演習
    7. 命令レベル並列処理
    8. 命令レベル並列処理の演習

演習と設計コンテストで成績評価する

並列コンピュータって何？

- CPU＋キャッシュ（あるいはメモリ）＝コアを複数持つコンピュータ
- 同時に処理をすれば、その分速くなる
  - これは当たり前だよね
- 同時に処理すると電力が小さくなる
  - これはちょっと分かりにくいかも→次のページへ
- 2003年以降急速に発達
  - スマフォも2-4コア、PCは4-8コア
  - サーバー、データセンター、スパコンはもちろん
- 現在ほとんどのコンピュータは並列化されている

電力の削減

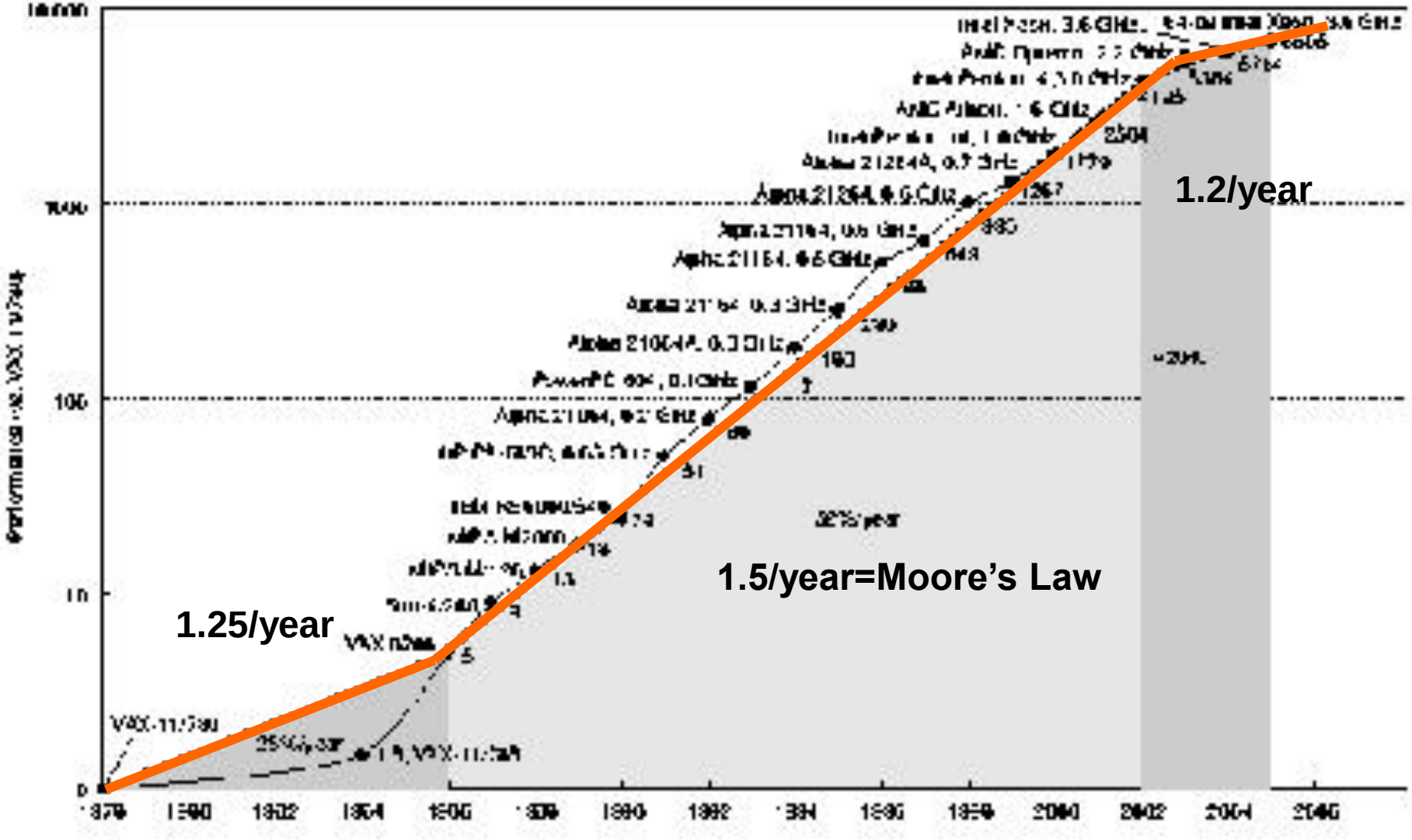
- nプロセッサで並列処理すればn倍速くなる（理想的には、、）
- しかし、nプロセッサを動かすと電力はn倍に！

→これではちっとも電力が減らない（むしろ増えるかも、、）

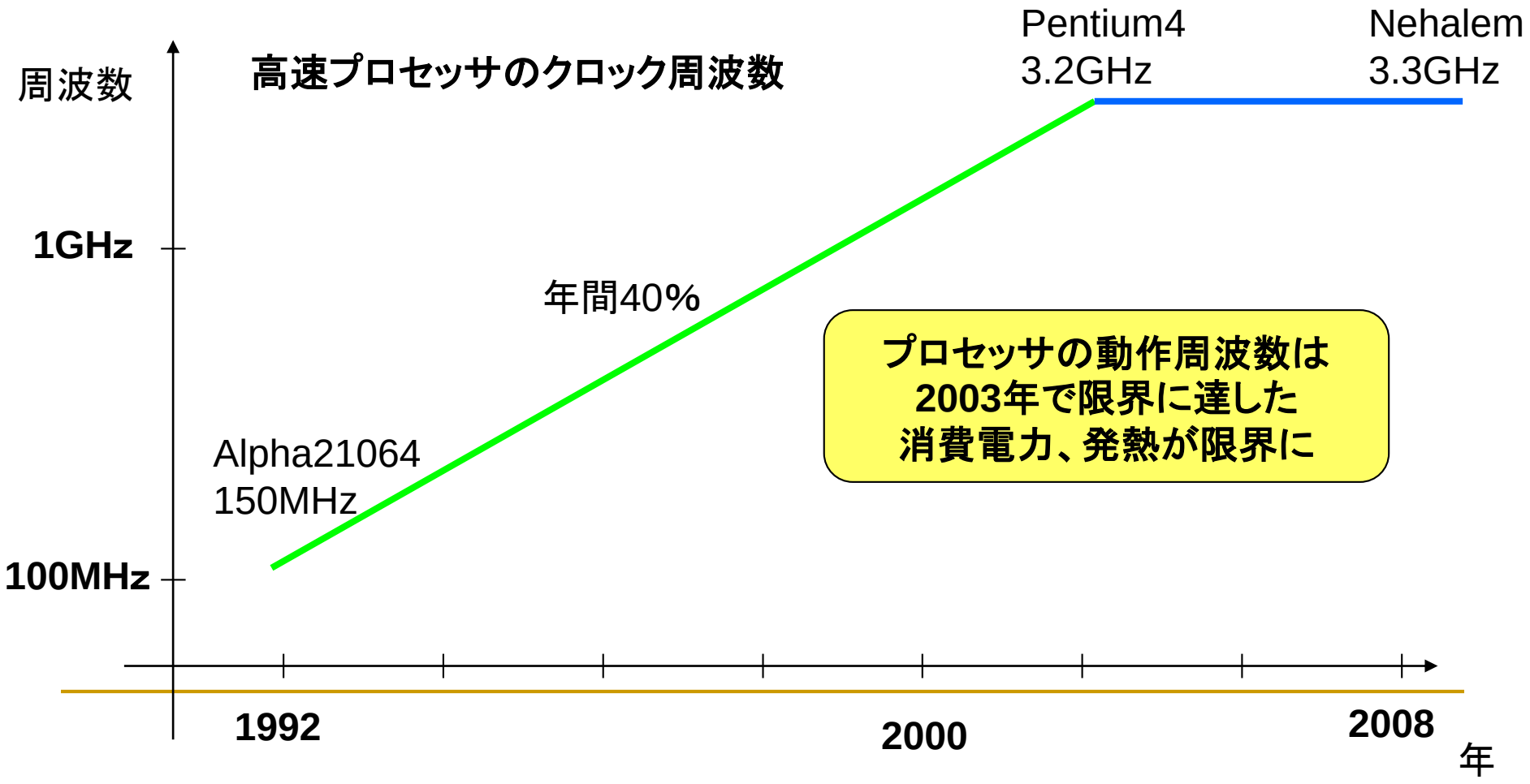
しかし

  - 動的電力　∝　電源電圧の2乗×動作周波数
  - 動作可能な周波数　∝　電源電圧
- 動作周波数を1/nにすれば1/nの電源電圧で動作する（もちろん一定の範囲だが、、、）
- nプロセッサを使って性能をn倍にする代わりに、周波数を1/nにする→電源電圧も1/nにできるので電力は2乗で減る→同じ性能で電力が減る！

コンピュータ性能におけるMooreの法則の終焉



クロック周波数の向上


高速プロセッサのクロック周波数
周波数
1GHz
100MHz
Alpha21064
150MHz
年間40%
Pentium4
3.2GHz
Nehalem
3.3GHz
プロセッサの動作周波数は
2003年で限界に達した
消費電力、発熱が限界に
1992
2000
2008
年

なぜマルチコア、メニーコアになったのか？

- 電力の壁
  - チップに供給し、放熱する電力がクロック周波数の向上で限界を超えた
- メモリの壁
  - CPUのクロックが速くなってもメモリが付いていけず性能が上がらない
- 細かいレベル並列性を生かすことが限界に達した

というか、何で2003年まで並列化されてなかったのか？

- コンピュータを楽に高速にする方法が他にあったから
  - パイプライン処理
  - 命令レベル並列処理
  - プロセスが進めば動作周波数が上がる、電力も下がる→
    スケーリング則
- コンピュータのビジネスモデル
  - 今まで配布されたバイナリ（機械語）レベルのプログラムを高速化できなきゃダメ
  - 互換性が命
  - 並列化は互換性と矛盾はしないが、そのままで速くなるわけではない

スレッドレベル並列性とは？

- 複数のPCに相当する複数の命令の流れ
- 通常のプログラムは単一スレッド
  - プログラマが並列に書く
  - コンパイラが並列化
- データレベル並列性
  - 大量のデータに対して共通の処理を実行
- 要求レベル並列性
  - 大量に到着する独立の処理要求
  - Webサーチ、トランザクション処理

Flynnの分類

- 命令流(Instruction　Stream)の数：M(Multiple)/S(Single)
- データ流（Data　Stream）の数：M/S
  - SISD
    - ユニプロセッサ（スーパスカラ、VLIWも入る）
  - MISD：存在しない（Analog　Computer）
  - SIMD
  - MIMD

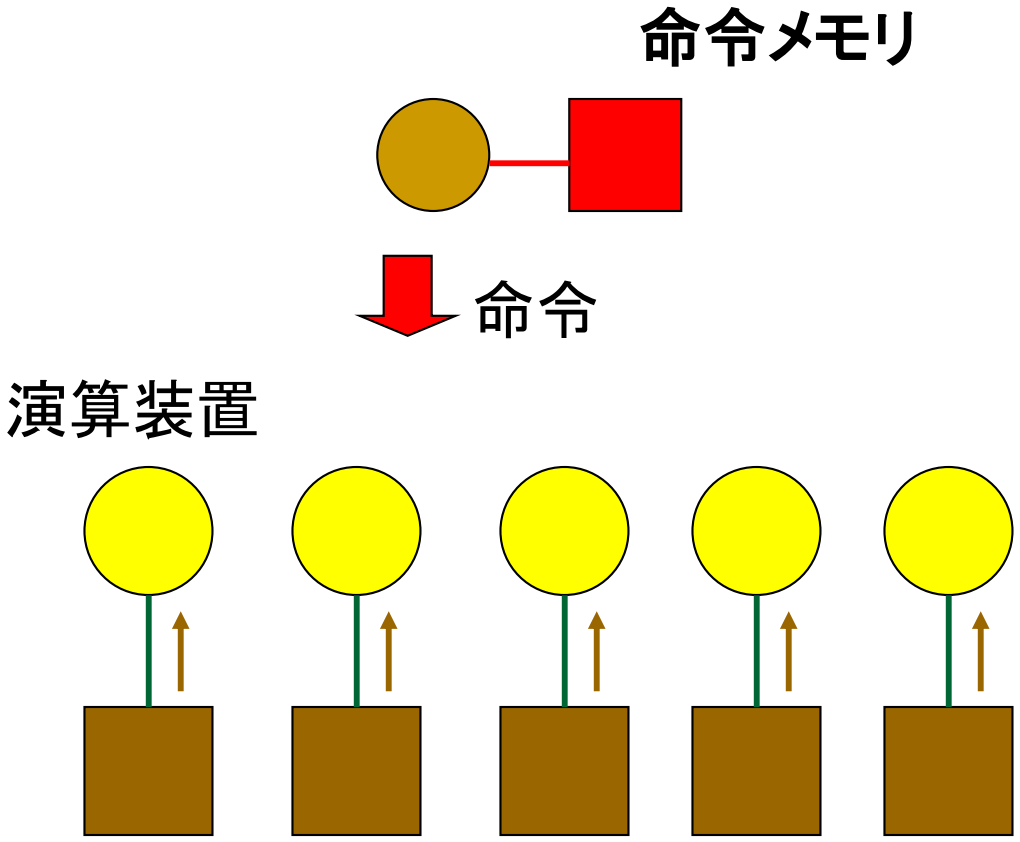
SIMD(Single Instruction stream/Multiple Data streams)の利用

- SIMD命令
  - 64ビット演算器を8ビット×8、16ビット×4などビット方向に切り、同じ演算を実行する
  - ビット単位に切ったロード・ストア命令が必要

  Intel系のCPUが利用している
- SIMDアクセラレータ
  - CPUとは独立に一定の処理のみを高速化する専用プロセッサを設ける
  - GPU（SIMDだけではないが、、）

一人の命令で皆同じことをする

SIMD



半導体チップ内でたくさんの演算装置を動かすには良い方法

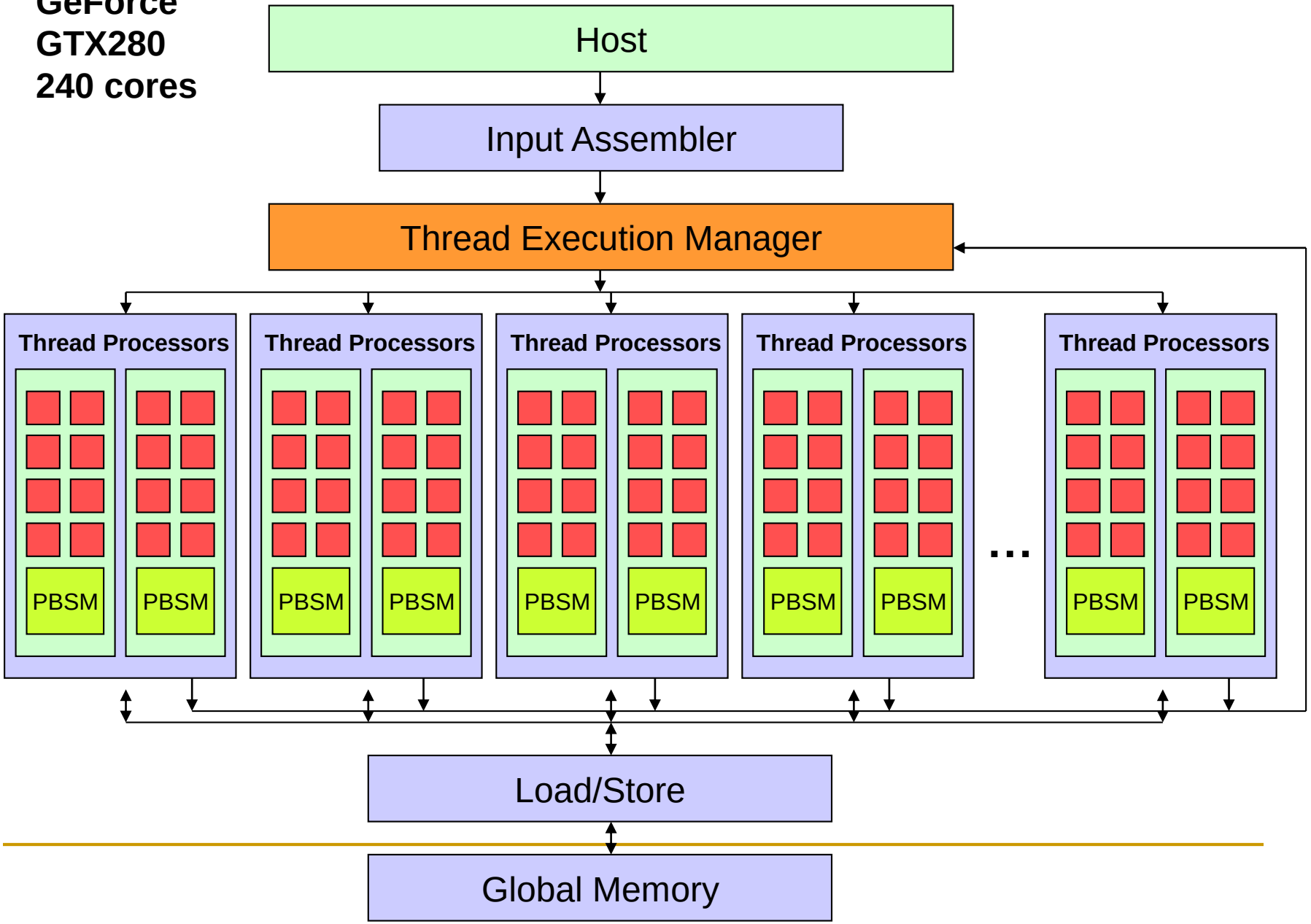
アクセラレータ（普通のCPUにくっつけて計算能力を加速する加速装置）の多くはこの方式

安くて高いピーク性能が得られる

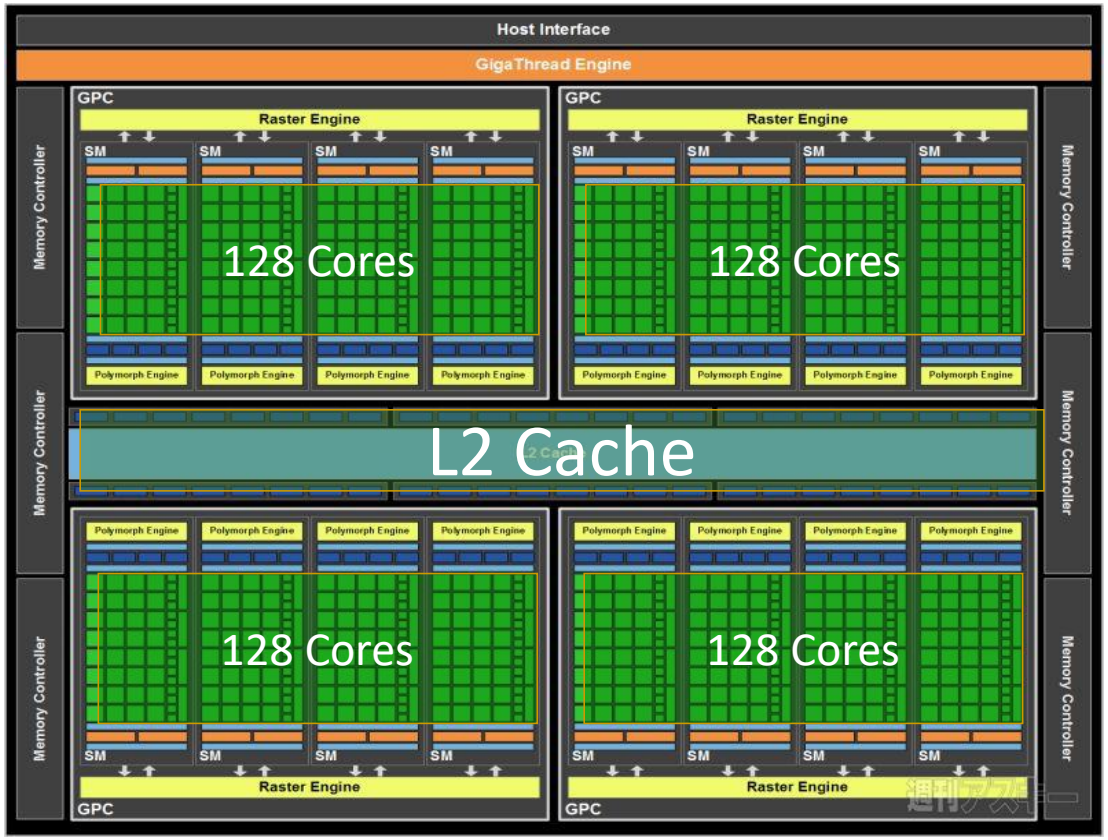
GPGPU：PC用グラフィックプロセッサ

- TSUBAME2.0(Xeon+Tesla, Top500 2010/11 4th)
- 天河一号(Xeon+FireStream, 2009/11 5th)

GeForce GTX280 240 cores



GPU (NVIDIA's GTX580)



128個のコアは
SIMD動作をする

4つのグループは
独立動作をする

もちろん、このチップを
たくさん使う

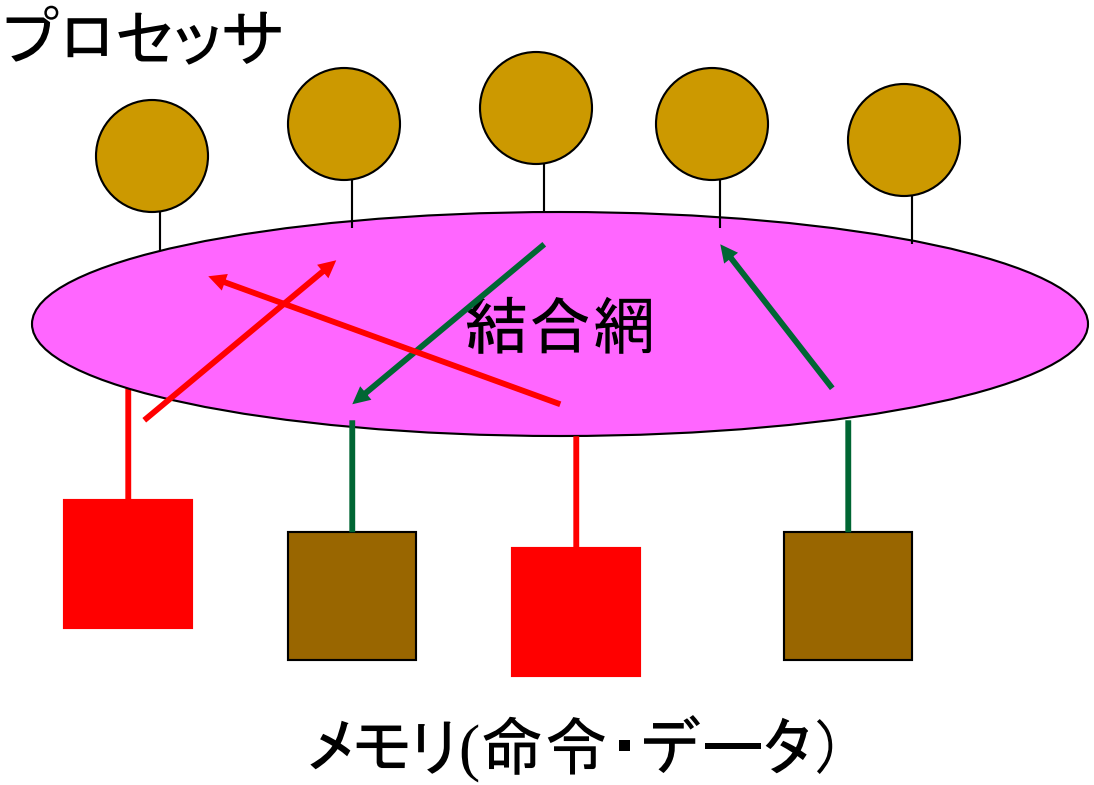
512 GPU cores (128 X 4)
768 KB L2 cache
40nm CMOS　550 mm^2

GPUは高い性能を得る最も簡単な方法

- 終わりの方でアーキテクチャ解説とCUDA演習

MIMD

- 全プロセッサが自分の命令を独立に実行
- 同期が必要
- 汎用性が高い
- 様々な構成法が存在



共有メモリの形態による分類

- UMA(Uniform Memory Access Model)
  - どのプロセッサからでも同様にアクセスすることができる共有メモリを持つ
- NUMA(Non-Uniform Memory Access Model)
  - 共有メモリは持つがレイテンシが異なる
- NORA/NORMA(No Remote Memory Access Model)
  - 共有メモリを持たずメッセージ交換で処理を行う

マルチコア、メニーコア

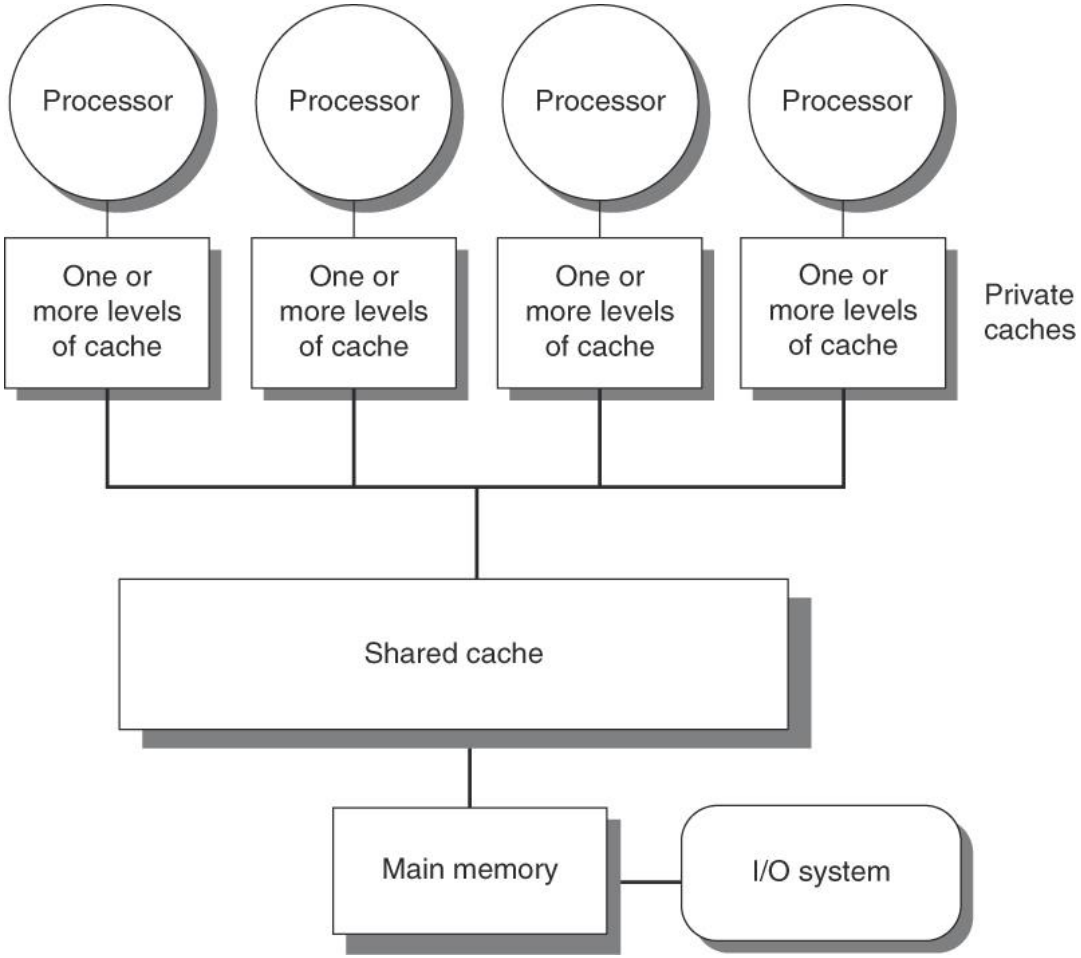
- 単純にチップ内のコア数による視点でFlynnの分類とは関係ない
- マルチコア
  - 2-16くらいまでの数
  - PC、サーバー、スマホのホスト
  - ほとんどの高性能コンピュータはマルチコア
  - MIMDのUMAが多い
- メニーコア
  - 64～数千コア
  - アクセラレータとしてホストと共に使われる
  - SIMD、MIMDのNUMAやNORAが多い
- アーキテクチャの方式による分類は崩壊している
  - GPGPUはSIMDの面が強いが、MIMD、マルチスレッドの技術も使っている
  - 色々な技術を総合的に使うのが最近のアーキテクチャ
  - とはいえ、分類は理解に役に立つ
- マルチコアのチップを複数用いる

  →マルチプロセッサ（古典的な言い方）

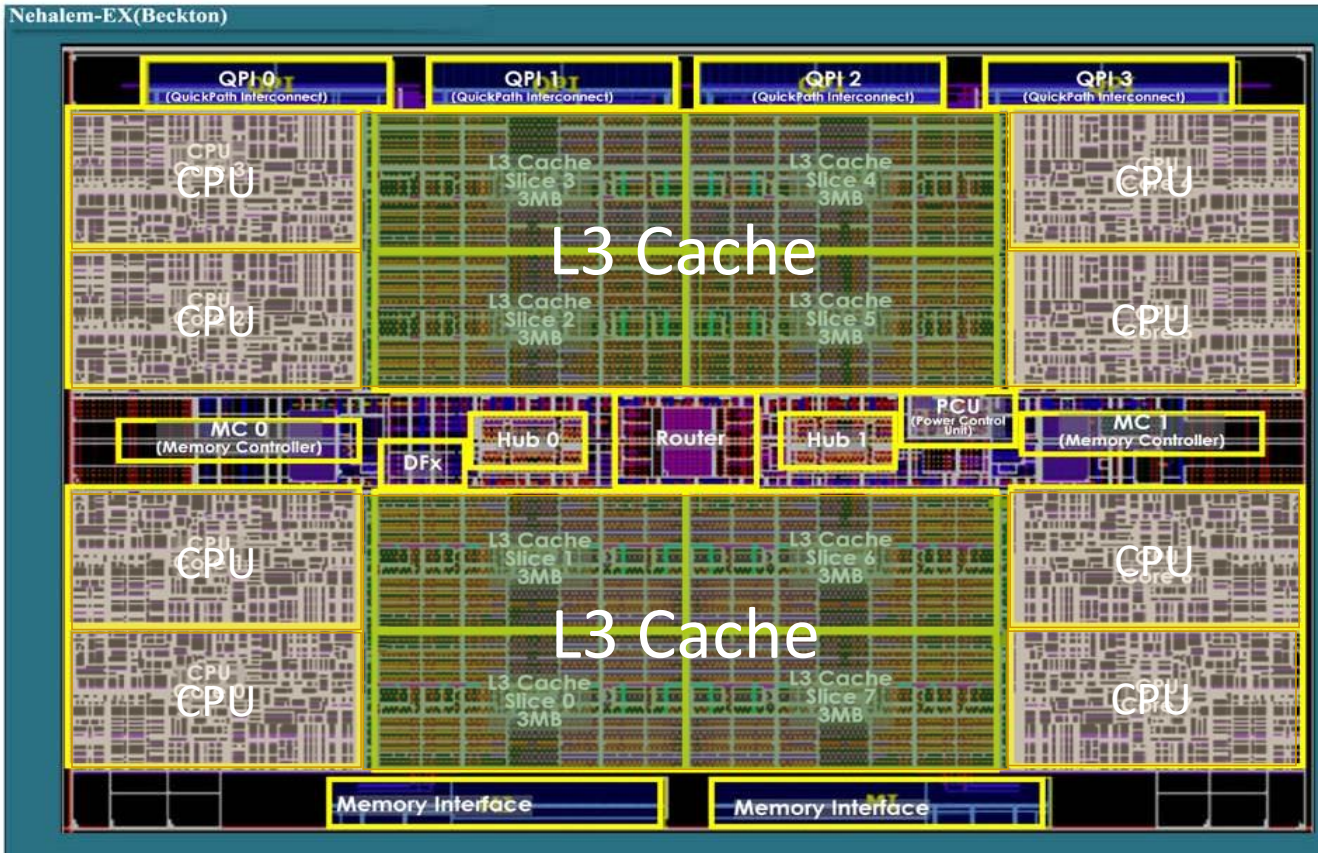
マルチコア、マルチプロセッサ

- ホストCPUとして使われる
- 並列OSが動作する
  - プログラムは共有メモリモデル（OpenMPなど）やメッセージパッシングライブラリ（MPIなど）
    - 同期操作が必要
  - 並列化コンパイラも使われる
  - 要求レベル並列性による独立処理
- 単一のマルチコアから大規模なクラスタまで

典型的なマルチコアシステム



Multi-Core (Intel's Nehalem-EX)



8 CPU cores

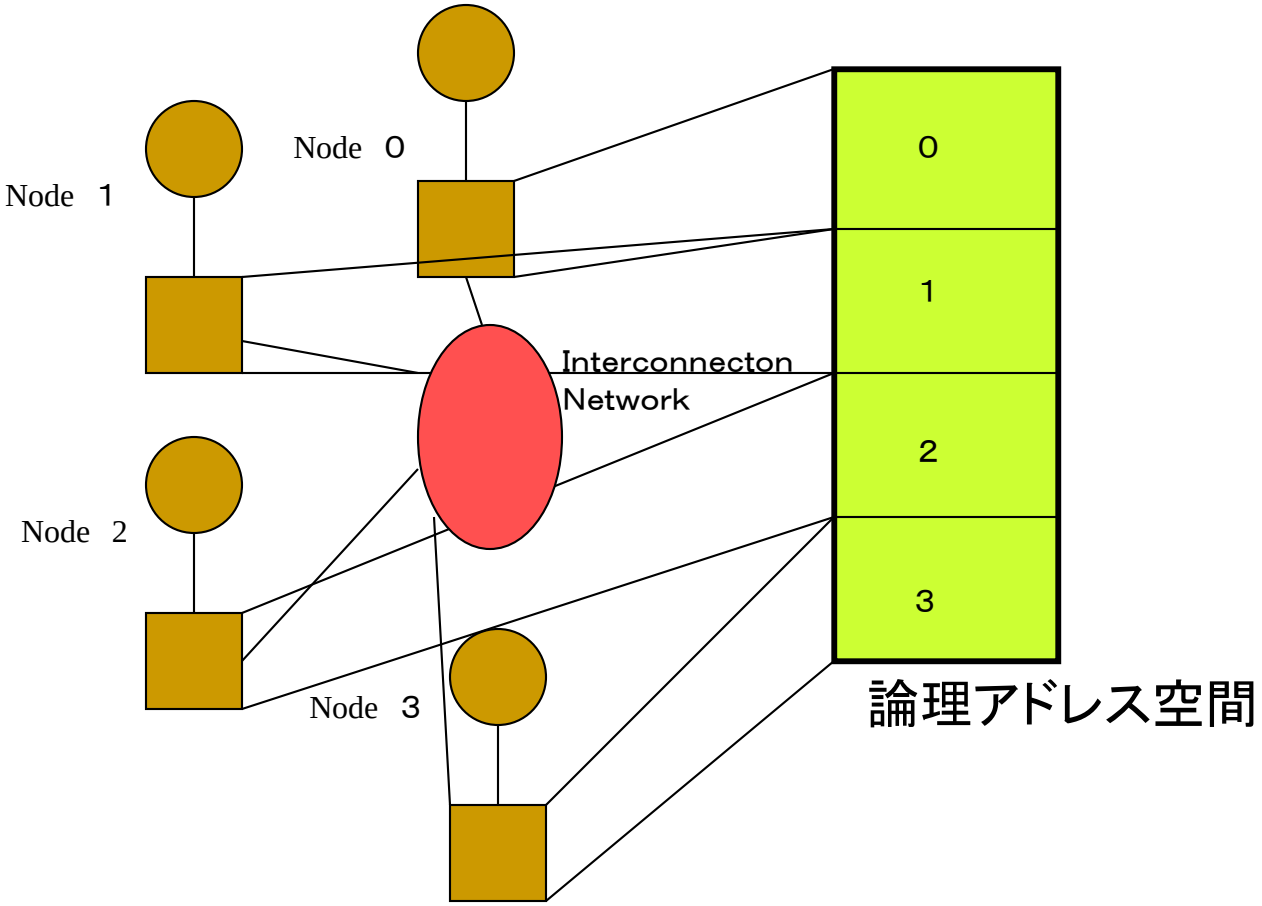
24MB L3 cache

45nm CMOS 600 mm^2

NUMA

- マルチコア＋メモリでノードを形成
- 他のノードのメモリをネットワーク経由でアクセスする
- アドレス変換、キャッシュ制御が必要でハードウェア制御が複雑
- スケーラブル:
  - UMAのプログラムがそのまま移植可能
  - プロセッサ数を大きくした分の性能向上が得られる
- 最近のサーバーはこの構成を取る

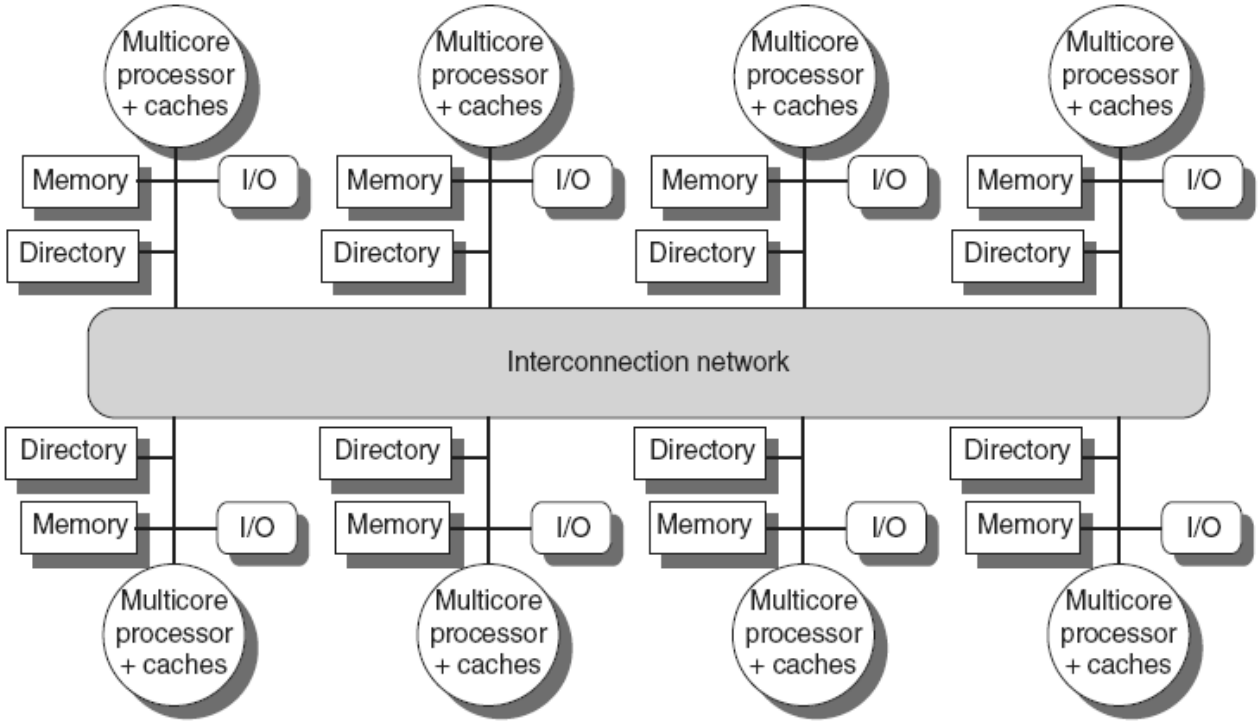
典型的な構成


Node 0
Node 1
Node 2
Node 3
Interconnecton
Network
0
1
2
3
論理アドレス空間

NUMAのシステム例

- L3キャッシュをディレクトリ方式で構成

IBM Power 7
AMD Opteron 8430



スーパーコンピュータ京もNUMAに分類できる

共有メモリ型マルチコア、メニーコアは最も一般的

- 第2回　アーキテクチャを概観
- 第3回　OpenMPによるプログラミング演習

NORA/NORMA

- 共有メモリを持たない
- 交信はメッセージのやりとりで行う
  - MPIが主に使われる
- 接続はGigabit EthernetやInfiniband
- 最近は多出力のスイッチを用いる
  - ハイラディックスネットワーク

データセンターなどで要求レベル並列性を処理

クラスタコンピューティング

第4回にアーキテクチャ解説、第5回にMPI演習


RHiNET-2 cluster

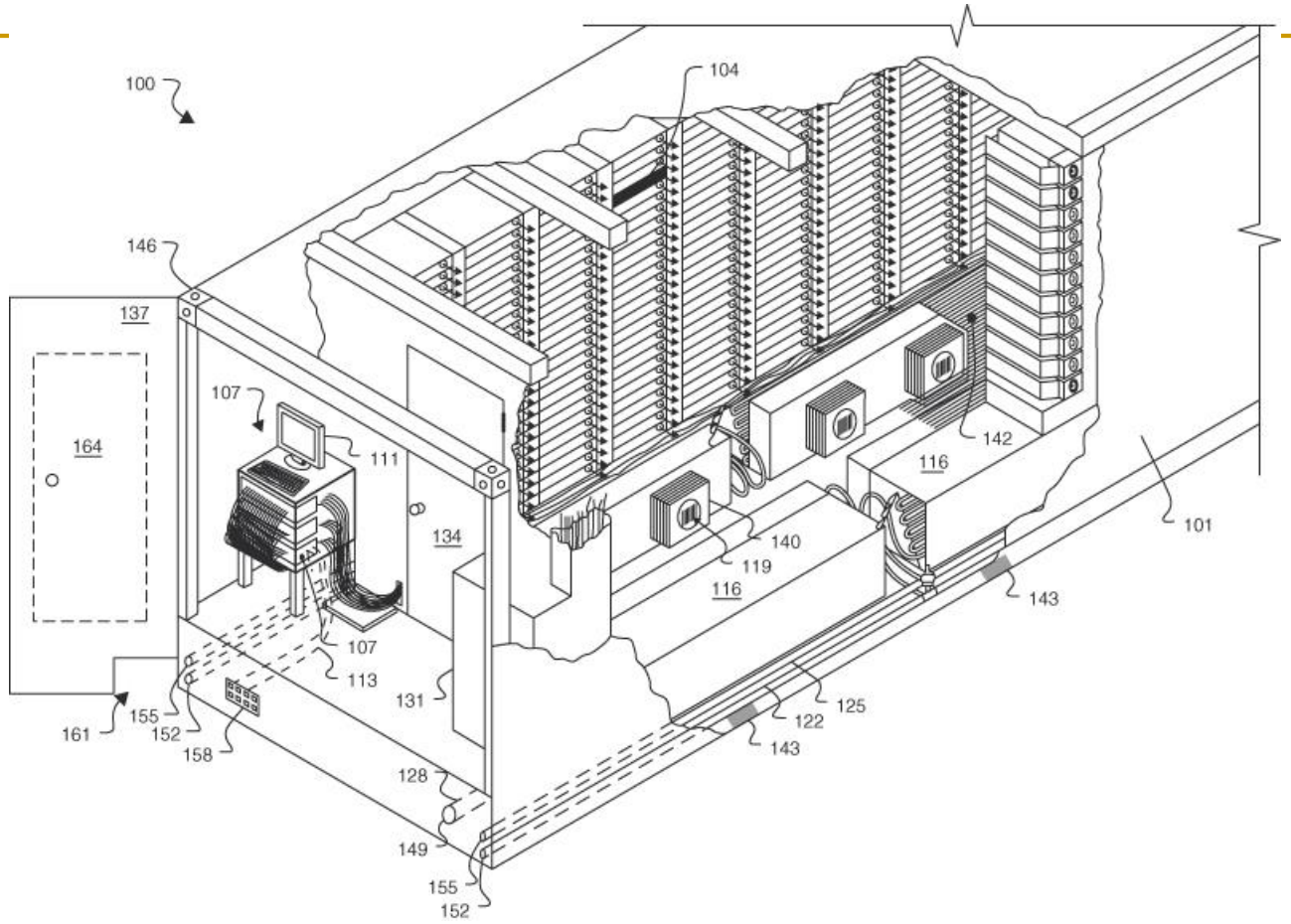


Figure 6.19 Google customizes a standard 1AAA container: 40 x 8 x 9.5 feet (12.2 x 2.4 x 2.9 meters). The servers are stacked up to 20 high in racks that form two long rows of 29 racks each, with one row on each side of the container. The cool aisle goes down the middle of the container, with the hot air return being on the outside. The hanging rack structure makes it easier to repair the cooling system without removing the servers. To allow people inside the container to repair components, it contains safety systems for fire detection and mist-based suppression, emergency egress and lighting, and emergency power shut-off. Containers also have many sensors: temperature, airflow pressure, air leak detection, and motion-sensing lighting. A video tour of the datacenter can be found at *http://www.google.com/corporate/green/datacenters/summit.html*. Microsoft, Yahoo!, and many others are now building modular datacenters based upon these ideas but they have stopped using ISO standard containers since the size is inconvenient.

演習1

- あるコアが最大動作周波数3GHz、電源電圧1.8Vで10W消費する。このコアを10個使って並列処理すると性能が10倍になるので、周波数を300MHzで動作させて同じ性能が得られる。この時、必要な電源電圧が0.8Vである場合、電力をどの程度にすることができるか？