将来の演算加速機構について

児玉祐悦
筑波大学計算科学研究センター
筑波大学大学院 システム情報工学研究科

「演算加速機構を持つ将来のHPCIシステムに関する調査研究」

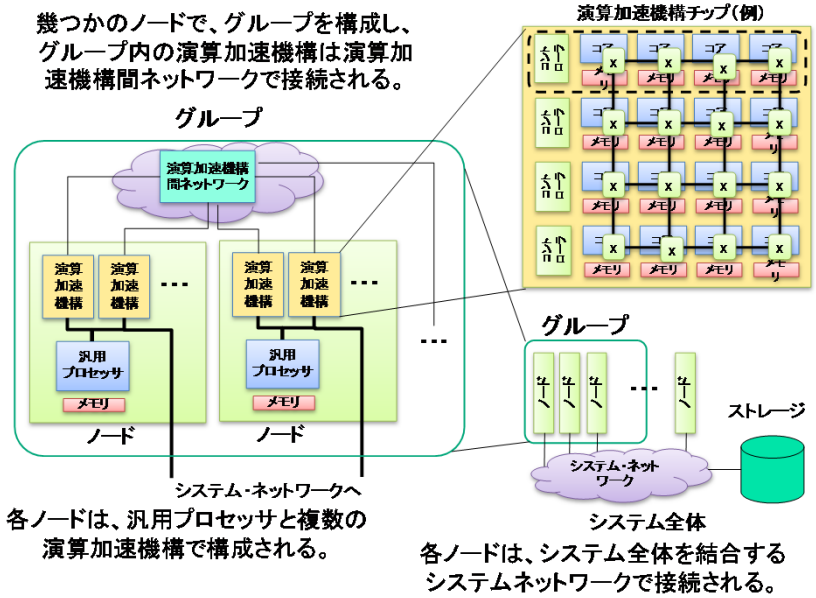
- ナノテクやライフサイエンスの進歩、気候気象予測や地震・防災への対処には計算科学は不可欠かつ有効な手段
  - そのためにはさらなる計算能力が要請されている。
  - 設置面積、消費電力等の制限からノード数の増加による並列システムの性能向上には限界
- ライフサイエンスの分子シミュレーション等、多様な分野で比較的小さい一定サイズの問題の高速化が望まれている(いわゆる強スケーリング)
  - 対応した研究開発の例: ANTON, MDGRAPE-4

電力効率の大幅な効率化と強スケーリング問題の高速化による新たな計算科学の展開を目指して、演算加速機構による並列大規模システムについて調査研究を行う。

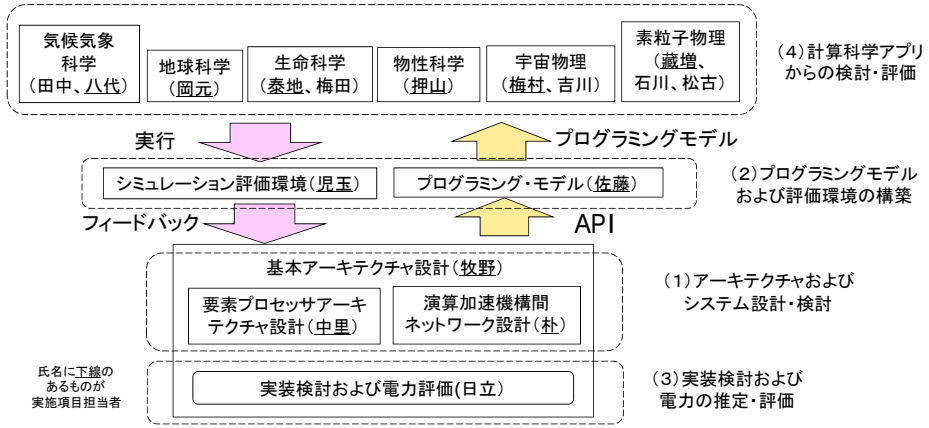
平成23年度文部科学省アプリケーション＆コンピュータアーキテクチャ・コンパイラ・システムソフトウェア合同作業部会において、まとめられた「今後のHPCI技術開発に関する報告書」の中で、分類されたシステム構成のうち**「メモリ容量削減」**および**「演算重視」**のシステムを主な調査研究の対象とする。

多数の演算コアを内蔵したチップによる演算加速機構が汎用プロセッサで構成された並列システムの各ノードに接続もしくは内蔵されているヘテロジーニアスな並列システムを想定

演算加速機構は、多数のスループットコアにより構成。スループットコアは、チップ内ネットワークにより結合される。図に示したものは一つの例。



- 主管事業実施機関: 筑波大学 計算科学研究センター
- 共同事業参画機関: 東京工業大学、理化学研究所、会津大学、日立製作所
- 協力機関: 東京大学、広島大学、高エネルギー加速器研究機構
- 調査研究を、以下の4つの項目に分けて実施
  - (1)アーキテクチャおよびシステムの設計・検討
  - (2)プログラミング・モデルおよび評価環境の構築
  - (3)実装検討および電力の推定・評価
  - (4)計算科学アプリからの検討・評価



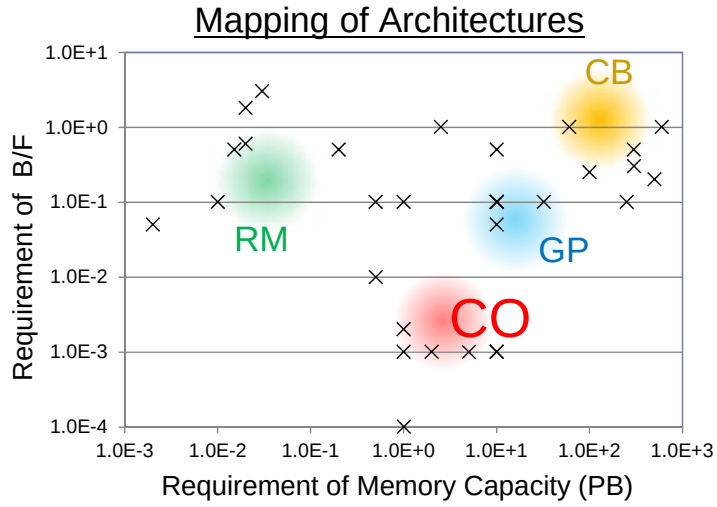
アウトライン

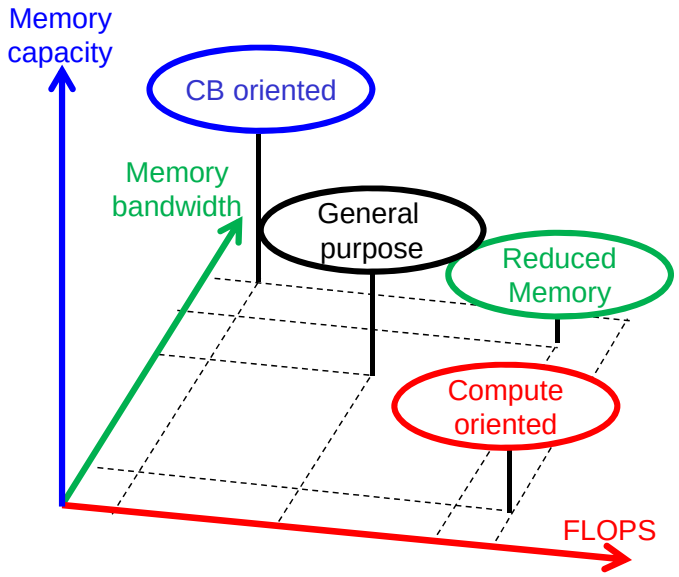
- PACS-GのStraw man（たたき台）アーキテクチャと性能概算
- 電力、面積等からのアーキテクチャの見直し
- プログラミングモデル、性能評価環境の進捗
  - PACS-Gソフト開発用シミュレータ
  - サイクルベースシミュレータ

評価対象アプリケーション

- 計算科学に対する**社会的・科学的課題の達成のために必要なアプリケーション**のうち、本調査研究で対象とするメモリ削減型(RM)および演算重視型(CO)で、ある程度の実行効率が期待できるものの洗い出しを進めている。
  - 生命科学、物性科学における分子動力学計算、生命科学、物性科学、ものづくり分野における第一原理計算、素粒子物理における格子QCD、原子核物理における様々な手法、宇宙物理における粒子シミュレーション、流体計算等(合同作業部会報告より)
- 本調査研究では科学技術計算における**典型的な計算の一つであるステンシル型の並列アプリケーション**について適用可能の検討を進めている
- 以下の5つのアプリのカーネルをターゲットとしてアーキテクチャとのco-designを進めている。
  - 格子QCD　(素粒子分野)
  - 重力多体計算　N-body (宇宙物理分野)
  - 磁気流体コード HMD　(宇宙物理分野)
  - 分子動力学　MD (生命科学)
  - 地震波計算コード(地球物理)
- 現在、検討中:NICAM(気象)、RS-DFT(物性)、FMO(化学)

(合同作業部会報告より抜粋)

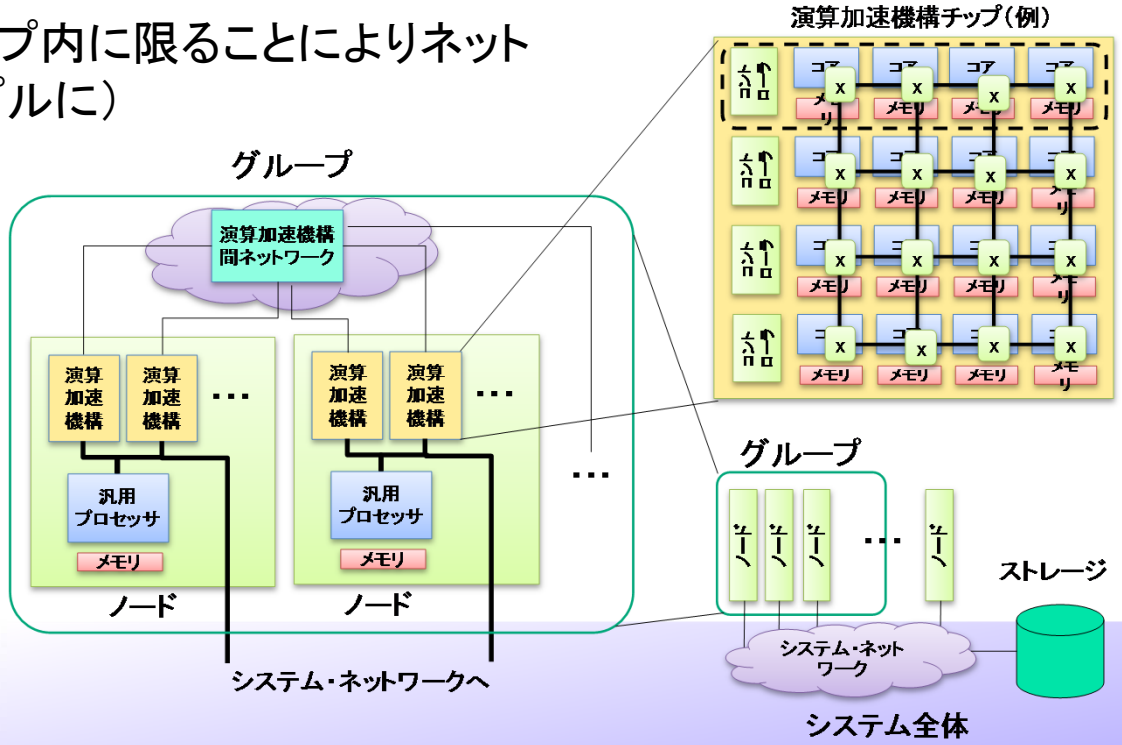




想定システムの基本コンセプト

- 演算器リッチな構成で演算重視型のアプリに対応
- オンチップメモリを演算コアに直結してメモリ削減型のアプリに対応
- 高い電力効率を実現するために、シンプルなSIMD構成
- 適用アプリの拡大を図るためコア間の隣接通信を実現
- アクセラレータ間直接通信機構により、強スケーリングに対応

（ただしグループ内に限ることによりネットワークをシンプルに）



想定している性能パラメータ

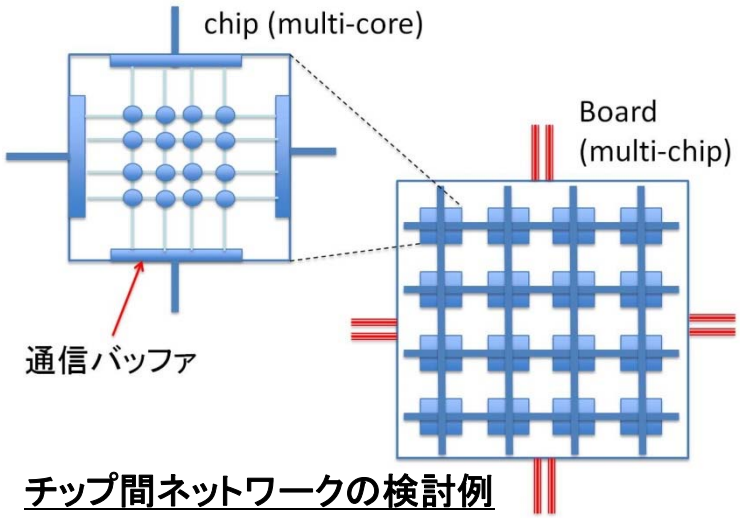
- 2018～2020年のLSIテクノロジーとして、14～10nmを想定。20mm角チップとして、メモリ(SRAM)換算で1GB/チップ。
- チップの基本アーキテクチャは、SIMD
- 演算器とオンチップメモリに1：1に割り当てる
- 64x64=4096PE/chip
- 512MB/chip, 128KB/PE
- 動作周波数は1GHz
- 128bitFMA演算器として倍精度で4GFLOPS/PE, 16TFLOPS/chip
- メモリバンド幅128bit/cycleとして16GByte/s/PE, 4Byte/FLOP
- 隣接間通信：片方向8GB/s
- 電力は250W/チップを目標 （16MW/1EF）
- 32GB, 1TB/sくらいのスタックメモリの接続を検討

GPUとの比較

- メモリバンド幅
  - GPU: NVIDIA K20X (1.31TFLOPS, 250GB/s, B/F:0.2)
  - 想定システム: オンチップメモリ直結　B/F:4
    - スタックメモリを使った場合は、同様の構成のGPUと同程度
- 電力
  - GPU: メモリから演算器へのデータ移動
  - 想定システム：極力オンチップメモリを利用
- スケーラビリティ
  - GPU：PCIe Gen2 x16: 8GB/s, 1port
    - 1GPUを超えた問題の場合、GPU間通信がボトルネックに
    - 他ノードのGPUとはCPU経由のためレイテンシ増大
      - 筑波大ではTCAにより直接通信を可能に
  - 想定システム：グループ内チップ間：2Dトーラスで直結

PACS-G アーキテクチャの概要： ノード間、ホスト間

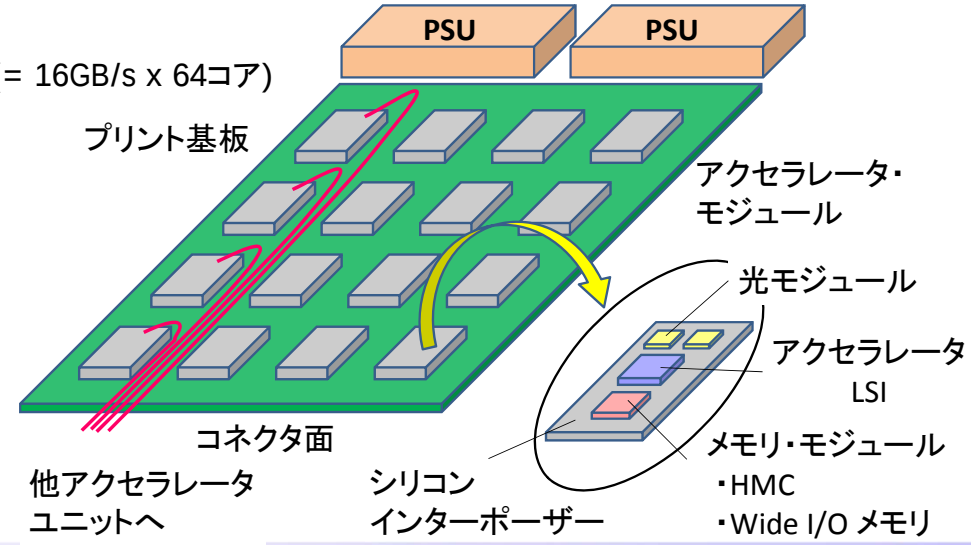
- 2048チップ（グループ）程度をネットワークで直接結合
  - チップ内の2次元メッシュネットワークをボード上およびボード間に展開
  - ボード上のチップ（例えば4x4=16個）間のバンド幅は、チップ内隣接ネットワークの20～40%程度で実現可能（電気配線）
  - ボード間ネットワークは、インタフェースチップからの光コネクションが使えればボード上バンド幅と同等（=チップ内ネットの40%程度）が実現可能。



チップ間ネットワークの検討例

チップ内コア間：
16GB/s (= 1GHz x 8B x 2(双方向)@コア間)→1024GB/s (= 16GB/s x 64コア)
チップ間（ボード内、ボード外）：
200～400GB/s (= 32ch. x 25～50Gbps x 2(双方向))

- 当面、2次元メッシュで考えるが、もう少し高次元の実装も検討。
- 汎用CPUを基本とした超並列システムにアタッチされることを想定
  - ホストとのインタフェースは、PCI Express Gen4 x 16相当の性能を期待



システムの実装イメージの検討例

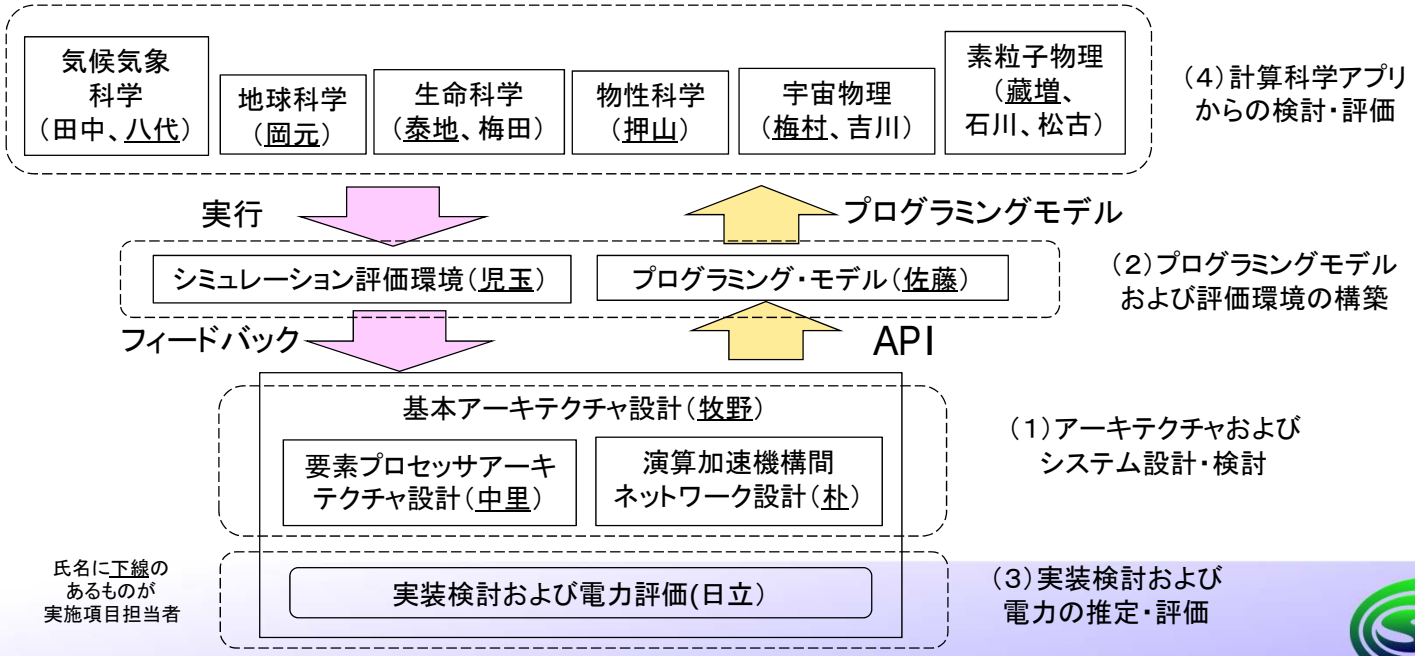
評価アプリによる性能概算

- 計算の1ステップでの演算、メモリアクセス、通信のパターンと発生量を分析し、想定したプロセッサアーキテクチャ、ネットワークアーキテクチャから期待できる性能を推定。現時点では、グループ単位（～2048チップ,32PF）に限定。
  - 全部データがオンチップに乗る場合：演算・メモリ性能　4 B/F, メモリ1TB/group
  - データが積層メモリに乗る場合：演算・メモリ性能　0.05B/F, メモリ32TB/group

アプリケーション	想定問題サイズ	効率・性能	コメント・比較
格子QCD （素粒子物理）	ハドロン多体系 128^4格子	12%～53% 7.9～34.7PF　2048チップ	• オンチップのメモリのみを利用 • 通信レイテンシパラメータの範囲で性能に幅がでる。 • 京では、効率が26%，32768ノード、1.1 PF
磁気流体コード （宇宙物理）	セル数　1984^3	1.89 PF, 22.5% 512チップ(8PF)	• HLL近似リーマン解法、磁場をflux-CT法よる有限体積法。 • グローバルメモリを利用 • 210-220ms/step，Intel Core i7 4096コアで4.5s/step
重力多体計算 （宇宙物理）	814G interaction/sec/chip （単精度、無衝突系）		• 重力計算を演算加速機構で加速。粒子の軌道計算はホスト計算機で行う。オンチップメモリのみ使用 • Intel Xeon E5-2670 の 66.7倍
分子動力学 （MD）カーネル （生命科学）	1セル/コア、1セル（5Å）^{3}，2580原子/コア	3.67PF、15M原子/256チップ，784.4us/ステップ	• 近距離相互作用の直接和計算を計算。 • セルインデックス法（空間座標分割）とハーフシェルスキームを仮定 • 通信ネックになっていないため小規模問題ではさらに高速化可能? • 京では、全ノードで500M原子、4.6PF, 114ms/ステップ
地震波 計算コード （地球科学）	格子サイズ 2048x2048x512	30%、3.5 PF /1024チップ	• 3次元時間領域差分法（FDTD）、空間差分4次精度、時間差分2次精度、弾性体、速度と応力を変数とするスキーム • オンチップのみ。今後、グローバルメモリも検討。 • 現状のGPUクラスタの200 倍程度の計算速度

プログラミングモデル、開発・評価環境

- PACS-Gソフト開発用シミュレータ
  - 命令セットの検討
  - プログラミングモデルの検討
- サイクルベース・シミュレータ
  - チップ内の命令実行をクロックサイクル精度で評価
  - PEアーキテクチャは、GRAPE-DRの構成をベースに拡張



PAGS-Gの基本アーキテクチャ

- 大規模メニーコア

  SIMDアーキテクチャ

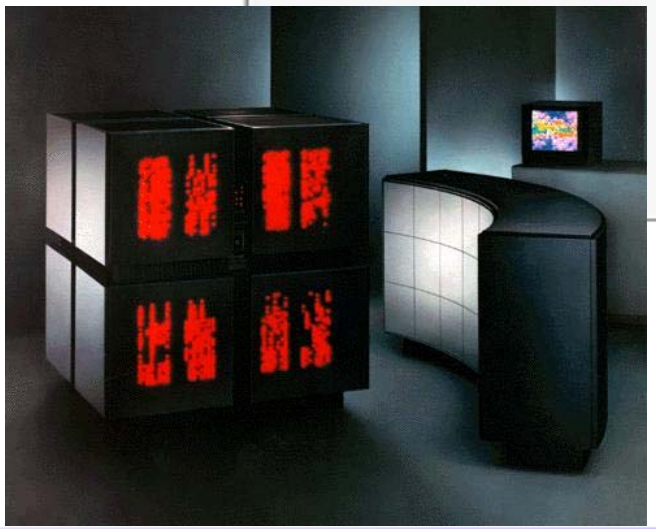
  GRAPE-DR

  +

  コア間ネットワーク通信(ステンシル)

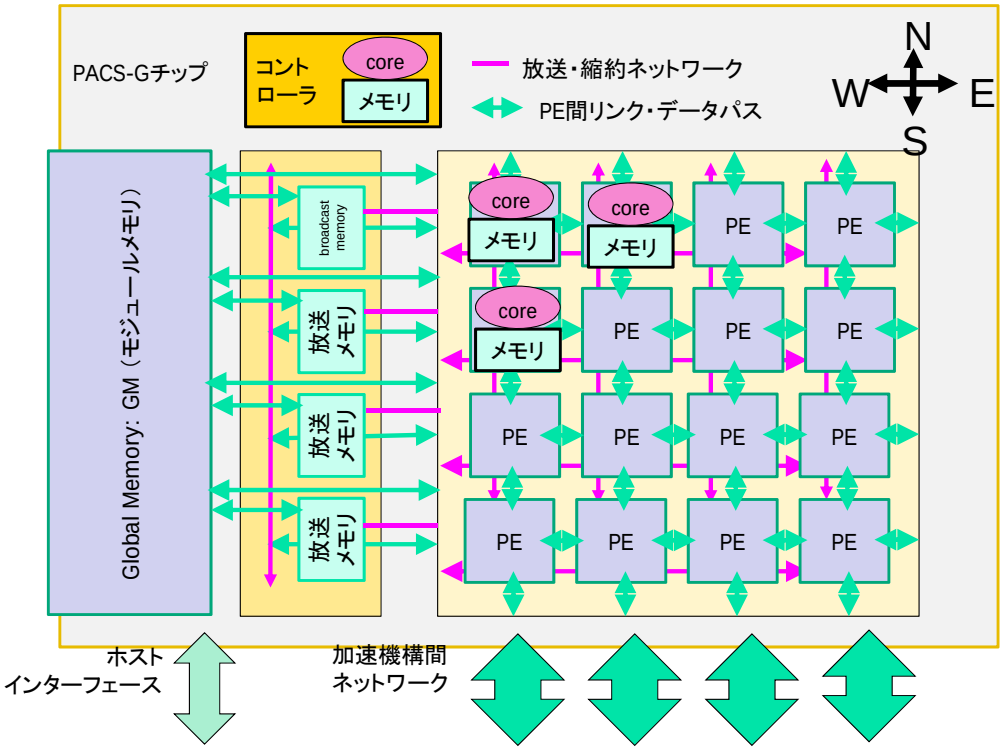
  ≒

  CM (Connection Machine)



CM-2 (1987?)

PACS-Gソフト開発用シミュレータ：PACS-Gの基本構成



- PE　プロセッサエレメント。CPUコアとメモリからなる。PE間は、2次元メッシュネットワークで結合されており、N*M個接続されている。全体がコントローラから制御され、SIMDで動作する。
- 基本命令（演算命令等）は、コントローラとコアで同じ命令が動くものと仮定。基本命令セットは、MIPS-1相当。
- Broadcast Memory (BM) PEの各列に接続されているオンチップメモリ。各列のPEにデータを転送したり、同じデータをブロードキャストする。また、各PEのデータを集約処理をして、書き込む。
- Global Memory (GM) チップに接続されているメモリ（積層メモリを想定）。PEにデータを読み込んだり、PEからデータを書き込むことができる。また、GMからBMにデータを転送することもできる。
- 現在、tiny-Cを使って、簡単なプログラムによりコントローラやPEの評価中

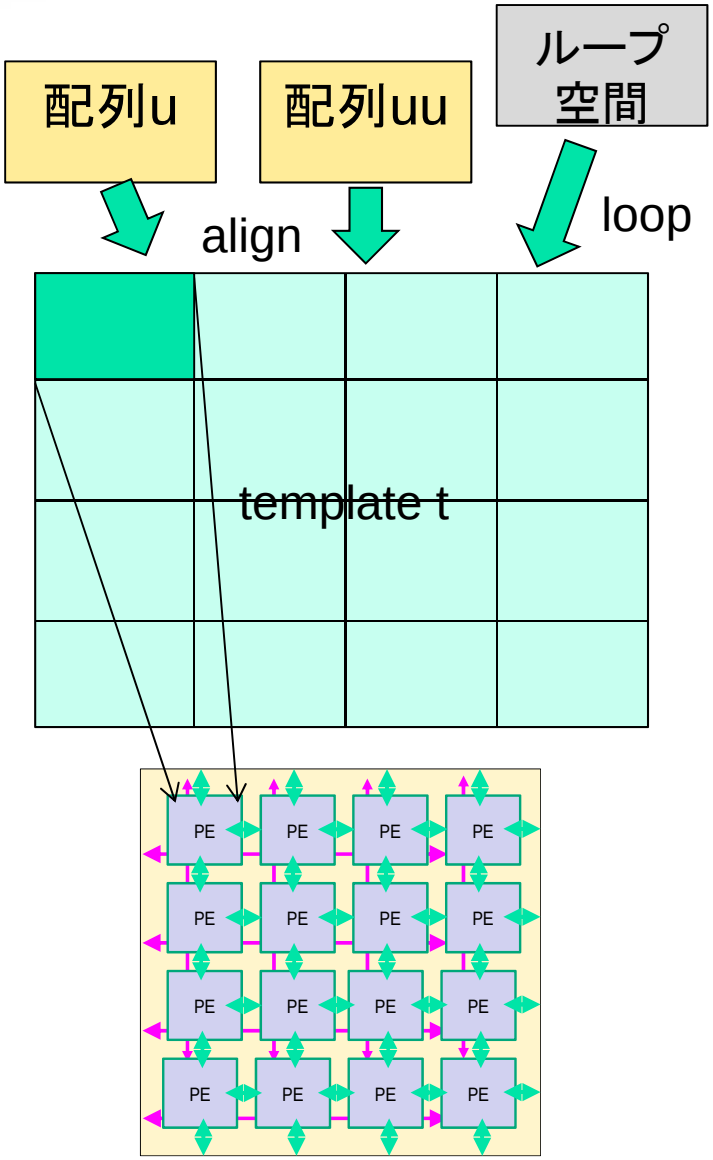
（とりあえず）低レベル・プログラミングモデル

- full setのCコンパイラ (F77も?)
- データ変数に拡張storage classを導入して、アドレスマップを作れるようにする
  - default（何もつけなかった場合）は、PEとコントローラに割り付け。すなわち、PEとコントローラは、同じアドレスマップを持つ（これでOK？）
  - _global_ で、グローバルメモリへ割り当て
  - _bm_ で、BMの割り当て
- PEにSIMDブロックを導入。
  - _pe_block_ { … }
  - ブロックのなかでは、演算命令かif文のみ。
  - これを、SIMD命令に展開
- レジスタ割り当ての品質を上げることは難しいので、変数のstorage classに指定
  - _reg_<レジスタ番号>
  - 指定された変数はレジスタに割り当て
- 転送命令やその他の命令について、intrinsic関数を準備

XcalableMPによるプログラミングの検討

- laplace方程式のソルバーのステンシルのコード
- template指示文で、コア上のtemplate(仮想index空間)を定義
- align指示文で、配列をtemplateに整合させて配置
- ループもloop指示文を使うことにより、iteration空間をtemplateに分割して実行
- block分割しかないため、distribute指示文はなし
- 隣接PEのアクセスは行えるためshadow指示文は不要（ただし、チップ間の場合は必要）

```
#pragma xmp template t(0:XSIZE+2, 0:YSIZE+2)
double u[XSIZE+2][YSIZE+2],uu[XSIZE+2][YSIZE+2];
#pragma xmp align u[i][j] with t(i,j)
#pragma xmp align uu[i][j] with t(i,j)
#pragma xmp loop on t(x,y)
  for(x = 1; x <= XSIZE; x++)
   for(y = 1; y <= YSIZE; y++)
      u[x][y] = (uu[x-1][y] + uu[x+1][y]
              + uu[x][y-1] + uu[x][y+1])/4.0;
sum = 0.0;
#pragma xmp loop on t(x,y) reduction(+:sum)
  for(x = 1; x <= XSIZE; x++)
    for(y = 1; y <= YSIZE; y++)
        sum += (uu[x][y]-u[x][y]);
```



プログラミングモデルからのアーキテクチャの検討

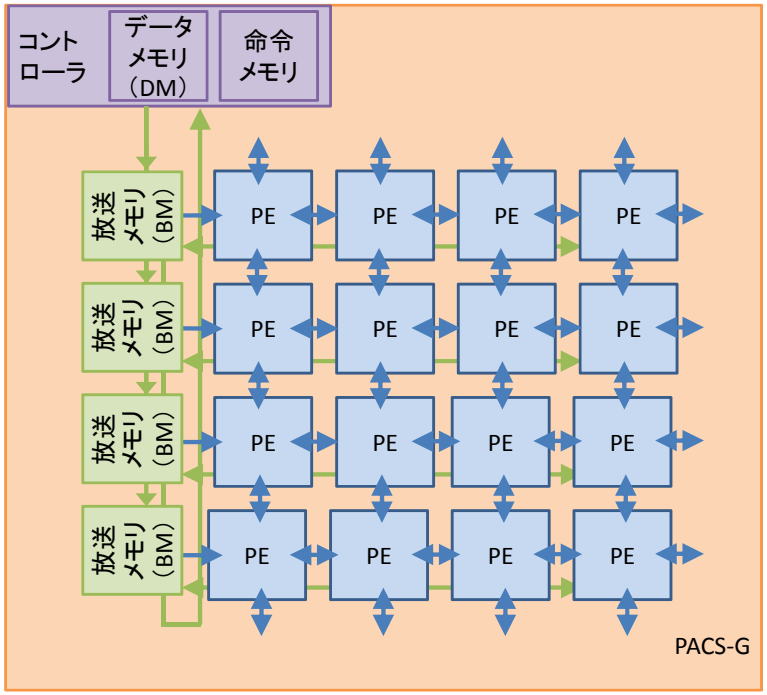
- 他のPEのメモリアクセス(get/put)
  - CM-2ではあった。
  - オンチップネットワークでルーティングが必要
  - SIMDなので、全部が終わるまで待たなくてはならない（PEのなかの計算とはオーバーラップができるが）
- 相対ルーティング
  - 多次元の隣接のアクセス

オンチップネットワークの見直し
⇒2D-HXB, 2D-fat-tree

- チップ間のプログラミングモデル
  - チップ間は、バッファリングして通信する。低レベルは、通信ライブラリ
  - 言語レベルでは、ユニフォームになるようにサポート
- ホストとのプログラミングインタフェース
- OpenCLのサポート
  - OpenCLのグローバルメモリをどこに置くか。
  - 案1）モジュールメモリ(GM)へのランダムアクセスを許す
  - 案2）PEのメモリをグローバルメモリとしてアクセスできるようにする（グローバルなアドレス付を行う）

サイクルベースシミュレータ

- チップ内の命令実行をクロックサイクル精度で評価
  - チップ間通信機構は未実装
  - PEアーキテクチャはGRAPE-DRを拡張
  - PE間隣接通信はPE間の4方向の専用レジスタへの読み書きとして実現
  - GRAPE-DRと同様の放送メモリ
  - ホストとのデータ通信や各PEの命令発行・フロー制御を行うコントローラ



アセンブラプログラムによる評価

- FDTD法による差分スキーム
- 姫野ベンチマーク
- 行列乗算
- N体問題

FDTD法

- アプリ評価でも用いている地震波計算コードのカーネル
- 3次元空間差分法スキーム（時間2次、空間4次）
- 1時間ステップごとに、速度ベクトルの更新と、応力テンソルの更新を繰り返す。
- 1格子あたりの変数は12個の単精度変数
- 1PEあたりn x n x n格子を計算する。袖領域は2層分必要のため、n=8で83KB<128KBで、n=8がオンチップメモリで保持できる最大。
- 3次元格子を、どのように2次元PEにマッピングするか? 16x16x16=4096なので、均等にマッピングすれば1チップで128x128x128の計算がオンチップで可能。

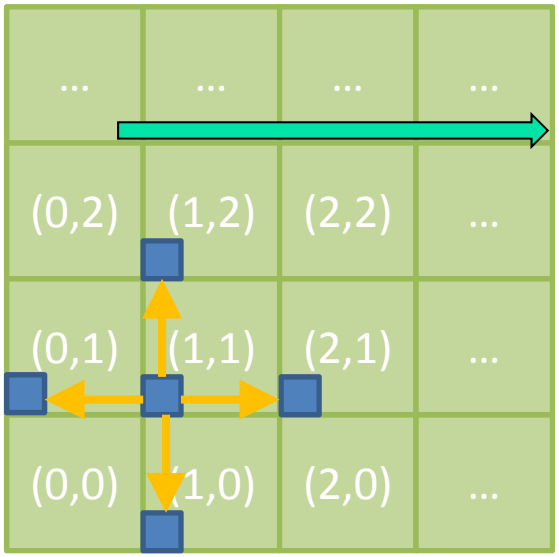
3次元格子の2次元PEへのマッピング

x、y次元の周期境界条件の転送は60ホップだが、2PE間の通信の場合は放送メモリを経由して4GB/sで実現可能。4PEペア間での転送は逐次転送

Z=

15	14	13	12
8	9	10	11
7	6	5	4
0	1	2	3

(x,y)=



- z次元の隣接は1ホップ
- PE毎に転送方向が異なる。
- ナイーブな実装だと条件実行が必要でオーバヘッド大
- 方向をレジスタ値で指定する隣接通信を追加。全PE同時実行可能。
- 設定する値はPE毎に異なる値をメモリに転送しておくことを想定

x、y次元の隣接は4ホップ、バンド幅1/4=2GB/s

FDTD法のアセンブラによる実装

- 元のプログラムはFortranで700行程度
- アセンブラはm4マクロを用いてループやインライン展開を記述
  - データ初期化　435行
  - 袖領域・境界領域通信　194行/変数、include文で各変数を展開
  - 演算 556行
  - 合計　1353行（展開後17153行）
- 1時間ステップの実行時間
  - 演算　26.2us
  - 通信　32.8us
- シミュレーション速度　1.1Kサイクル/秒　（8thread 2.4GHz Xeon)

アプリケーションレベルでの性能見積もり

- 時間ステップあたりの１格子の演算数とメモリ参照数を、元のFortranコードから見積もり、ピーク性能で計算時間を見積る。
  - 速度更新：演算数148、メモリ参照数155
  - メモリ参照ネックであり、各PEのメモリバンド幅は16GB/s、各PEは8x8x8格子を計算するので、演算時間は19.8us （=サイクルベースシミュレーション結果の76%）
- 時間ステップごとに袖領域の通信と、周期境界条件の通信が必要 で、その転送データ量を見積もり、ピーク性能で通信時間を見積もる。
  - 袖領域・周期境界の通信が必要な変数が速度更新で6変数、テンソル更新で3変数
  - 袖領域の通信は、それぞれ8x8x2の領域を6面分隣接PEに転送する（6.9KB）。3次元通信の隣接通信性能を2GB/sとして13.8us
  - 周期境界の通信は、それぞれ8x8x2の領域を4面分両端PE間で転送し合うが、そのペアが4組ある(18.4KB)。2PE間の転送は 4GB/sとして18.4us
  - 合計32.2us（≒サイクルベースシミュレーション結果）
- 演算時間と通信時間を合計して
  - 52.0us, 1.5GFLOPS/PE, 6TFLOPS/chip （37%）

アセンブラプログラムによる評価

- FDTD法による差分スキーム
  - 5.2TFLOPS（単精度）
- 姫野ベンチマーク(Middle:128x128x256)
  - 7.4TFLOPS（単精度）
- 行列乗算
  - Kernel部: 14.9TFLOPS（倍精度）
  - 放送メモリへのデータ転送時間を含めると1TFLOPSに低下
  - 512GB/sのバンド幅を持つスタックメモリを仮定すると12.5TFLOPSに向上

サイクルベースシミュレータの今後の評価

- チップ間ネットワークを含めた複数チップでの評価
- チップ内ネットワークの検討で、2次元隣接以外のネットワーク、例えば4次元ハイパークロスバなどの検討を行っており、それに対応したシミュレータによる評価
- 1命令4サイクルのベクトル演算を、通常の1命令1サイクルの命令とした場合の評価
- 計算と通信をオーバラップしやすくする工夫や、その場合の評価

- ソフトウエア開発用シミュレータとのすり合わせ（マージ？）

- CM-2やMasParといったかつてのSIMD型MPPを再評価し、改良の参考にする

まとめ・課題

- エクサスケール（エクサフロップス）システムを実現するには、演算に特化した演算加速機構が必要。（電力性能50GF/W以上、汎用では難しい）
- 次世代の計算科学を展開するには、strong scalingが必要。
- ポイントは、2018-2020で可能になる、14nm-10nmテクノロジーを用いて、オンチップの計算がどのくらいできるか（16TF/chipの場合のmemory wall問題、…）
- 次のステップはより詳細なプロセッサアーキテクチャの決定、命令レベルシミュレータの開発と、それによるアプリケーション（システム全体）の性能評価である
  - ネットワークトポロジーの検討（現在は、2次元メッシュ）
  - ある程度のプロセッサを演算加速機構に付加することも検討
  - ネットワークのルーティング機能の検討
  - プログラミングモデルとコンパイラの実装
  - 実際のコードを用いた定量的評価、詳細な電力評価（アプリFS mini-app）
  - NICAM（気象）、RS-DFT（物性）、FMO（化学）
- 全体システムの検討
  - ホストとの接続形態の検討
  - システム全体としての耐故障機能
  - I/O等