AI 向けプロセッサの開発と数値シミュレーション

牧野淳一郎

神戸大学/Preferred Networks

第 10 回 DML Lecture **2025/7/4**

話の構成

- これまでの計算機の進歩と現状の困難
- 「ポストムーア」
- 計算機の進歩の方向
- AI 向けプロセッサの方向
- MN-Core について
- 重力多体シミュレーション(時間あれば、、、)
- まとめ

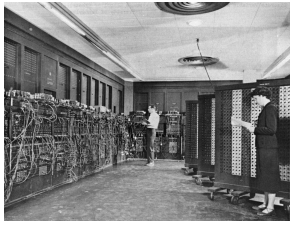
牧野はどういう人か？

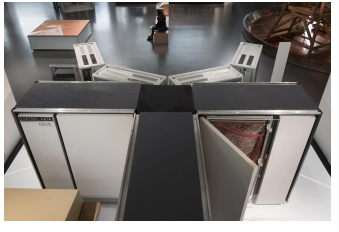
- 元々、銀河系や星団の、コンピュータシミュレーションを使った研究をしていた (今もしている)
- 「シミュレーション専用計算機」の開発を 1990 年くらいから始める
- 2004 年から、汎用並列計算機開発にもかかわる
- 2011 年から、理研の「ポスト京」プロジェクトにも
- 2016 年から、 AI ベンチャーの Preferred Networks と共同で深層学習向けプロセッサを開発
- 2020 年 6 月、完成したプロセッサ MN-Core を使った MN-3 が「Green500」ランキングで世界一位に
- 2023 年 11 月から Preferred Networks 兼任 (クロスアポイントメント)

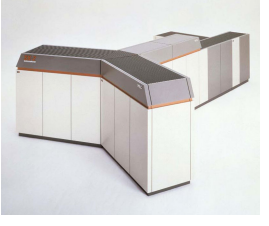
これまでの計算機の進歩

1940年代から2010年代までの70年間、ほぼ10年で100倍。何故そのような指数関数的進歩を長期に続けたのか？
基本的な理由:

- 使うスイッチ素子が高速になった
- 使うスイッチ素子が小型、低消費電力になって、沢山使えるようになった
- 使うスイッチ素子が安くなって、沢山使えるようになった(スパコンの物理的大きさは70年代が最小。そこまで段々小さくなって、そこからまた大きくなった)






過去のスパコンの進化

何故計算機はどんどん速くなったのか？
基本的な理由:

- 使うスイッチ素子が高速になった
- 使うスイッチ素子が小型、低消費電力になって、沢山使えるようになった
- 使うスイッチ素子が安くなって、沢山使えるようになった(スパコンの物理的大きさは70年代が最小。そこまで段々小さくなって、そこからまた大きくなった)

それらとアーキテクチャの関係は？今後は？という話を以下で。

素子の高速化

- といっても、真空管でもそれなりに速かった
- スイッチング速度が重要でないわけではないが、配線を信号が伝搬する速度のほうが昔から重要
- 昔は信号はほぼ光の速さでつたわった
- 最近の LSI 上の配線は非常に細く (抵抗が大きく)、キャパシタンスを充電しないといけないことによる RC(抵抗とキャパシタンス) 遅延のため、信号が伝わる速度は光速度よりはるかに低い
- 太い配線に大電流を流せば速いが、大量の電力消費になる

素子の小型化

- 真空管 → トランジスタ →IC という進化は 1970 年代までは重要
- サイズだけでなく、消費電力が下がることが重要
- 80 年代から重要になったのは CMOS LSI の微細化。10 年でサイズが 1/10 になる
- CMOS 素子では(2000 年くらいまでは)微細化すると電圧を下げることができ、消費電力が下がり、速度は向上した。**いわゆる CMOS スケーリング。**
- 2000 年頃からは電圧が下がらないので、電力はちょっと下がるが速度は上がらなくなった。**いわゆる CMOS スケーリングの終焉。**

- そろそろ微細化も困難になってきた。また、トランジスタの構造・製造工程が複雑になり、微細化するとかえって価格上昇するようになった。**いわゆるムーアの法則の終焉。**

CMOS スケーリングって何？

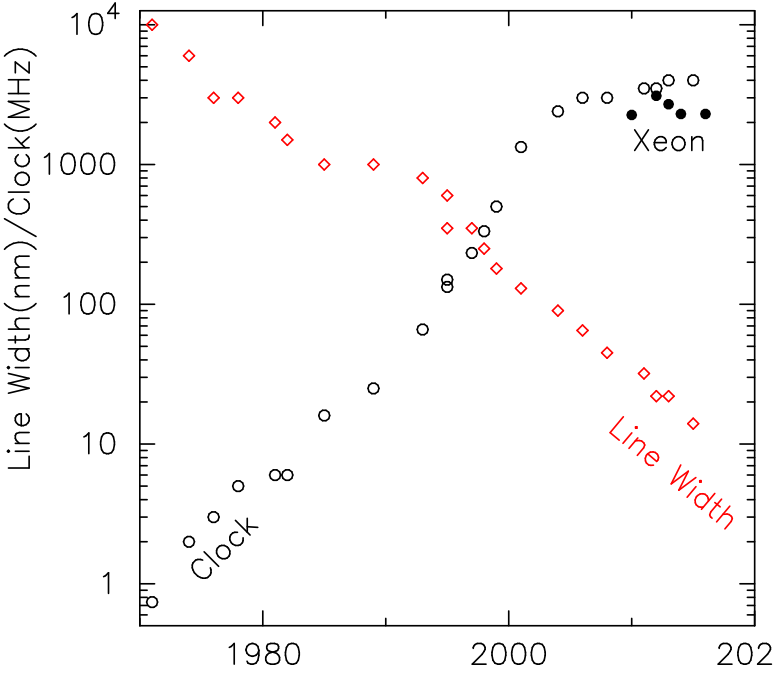
「Dennard Scaling」とも

トランジスタのサイズ (3 次元的にすべて)、電源電圧を $1/k$ にし、不純物濃度を k 倍にすると、トランジスタの速度は k 倍、スイッチングあたりの消費電力は $1/k^3$ になる

これが成り立つなら、

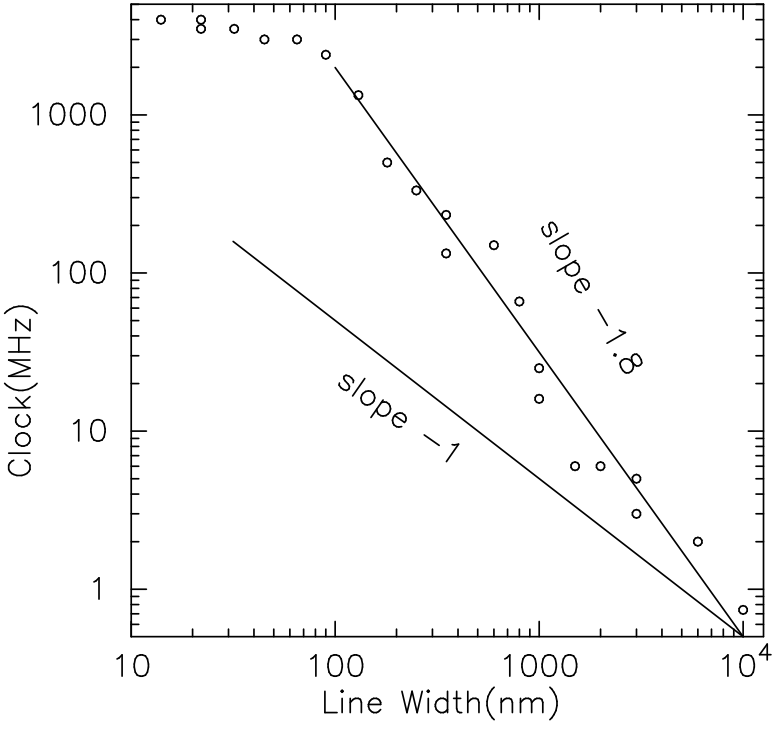
動作周波数 $\propto 1/$線幅
動作電圧 $\propto$ 線幅
性能 $\propto 1/$線幅^3 (電力、面積一定で)

現実は



40 年間の Intel マイクロプロセッサの線幅 (トランジスタサイズ) と動作クロック
線幅は 40 年間で 3 桁縮小。クロックは 25 年間で 3 桁上昇、そのあと停止。(コア数の多い Xeon は段々クロック下がる)

線幅とクロック



線幅とクロックの関係

90nm までは

クロック $\propto$ 線幅$^{-1.8}$

そのあとはほぼクロック一定

(Xeon は下がる、、、、)

90nm までのクロック上昇は理屈より速い=無理していた。

つまり

- ムーアの法則 (トランジスタサイズは時間の指数関数) は 2015 年くらいまでは成り立っていた (今は成り立たってない)
- いわゆる CMOS スケーリング (速度が線幅に反比例) は 2000 年くらいまでしか成り立たっていない

何が起こったのか？

- 90nm までは電圧をあまり下げず、パイプライン段数を増やすことでクロックをあげた
- これは消費電力増やすので、1 チップ 100W になったところで限界
- そこからは、コア数やコア内演算器数を増やすことで性能向上
- 最近はそれも難しくなってきて再びクロック上げる方向も

計算機アーキテクチャの進化を振り返ってみる

- 1976 年まで: スカラー計算機。最後は CDC 7600
- 1976 年から 1992 年まで: ベクトル (共有メモリ並列含む) 計算機。Cray-1 から C-90 まで
- 1993 年から 2008 年まで: (マルチコア含む) マイクロプロセッサの分散メモリ並列計算機 Cray T3D から、、、Red Storm あたりまで
- 2008 年から: GPU アーキテクチャのアクセラレータとマイクロプロセッサの組合せ: IBM RoardRunner (Cell なので GPU じゃないけど)

大体 15 年毎にアーキテクチャが大きく変わった。GPU の次はまだでてきてない

アーキテクチャの変化が必要な理由

基本的な理由: そのアーキテクチャでは、半導体の進歩を有効に利用できなくなる

- **半導体技術の変化**
- **アーキテクチャのスケーラビリティの限界**

スカラー計算機→ベクトル計算機

- スカラー計算機の進歩: 増えるトランジスタを「1 つの演算器を速くする」に (CDC 6600 は複数のパイプライン化されていない乗算器をもっていたが、7600 てパイプライン演算器に)
- 主記憶は磁気コアで、半導体よりはるかに遅い
- CDC 7600 (S. Cray の設計) あたりが最後
- S. Cray はこの後、4 プロセッサ並列の CDC 8600 を開発していたが、完成前に CDC やめて Cray Research を設立。ベクトルの Cray-1 を開発する

スカラー計算機では

- IC の開発で使えるゲート数が 1 演算器を超えて大きくなった
- 半導体メモリの開発でメモリが非常に高速になった

ことを生かせなかった

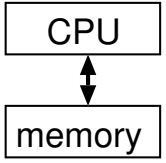
ベクトル計算機→分散メモリ並列計算機

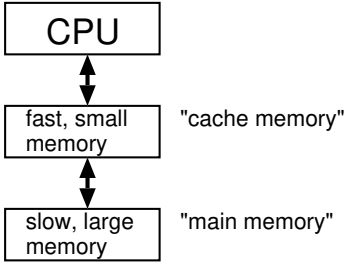
- ベクトル計算機の進歩: 1 プロセッサのパイプライン数や主記憶を共有するプロセッサの数を増やす
- パイプライン数、プロセッサ数に比例以上にメモリへの配線の数が増える。ある程度 (64 パイプラインあたり) で限界
- パイプライン化した 1 プロセッサが 1 チップにはいるようになる (Intel 80860)
- 1 演算器でも、プロセッサチップと外付けメモリの間のバンド幅が不足するようになり、キャッシュが必須になる。
- 多数の小さなプロセッサとメモリの組合せの間を細いネットワークでつなぐ構成が有利になる。ベクトル計算機よりはるかに少ない配線で高い性能になる。
- 初期の代表: Cray T3D。但しネットワークが太い設計でそのうち破綻。

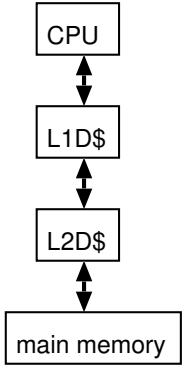
マイクロプロセッサ超並列→GPU

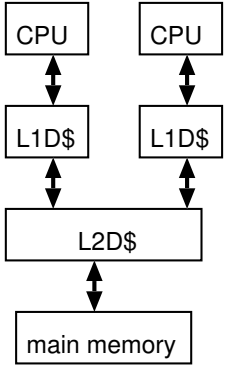
- 1 チップの中に沢山プロセッサがはいるようになる
- これは実はシステムレベルでのベクトル計算機の限界と同じ。外部メモリへのバンド幅もチップ内のバンド幅も不足する。
- マルチ (メニー) コアプロセッサでは、キャッシュコヒーレンシを維持した階層キャッシュとオフチップの共有メモリ
- キャッシュコヒーレンシの維持のための電力が支配的になる
- GPU ではキャッシュコヒーレンシを緩和したり、捨てたりで若干改善
- 本質的な解決ではない

キャッシュとは？







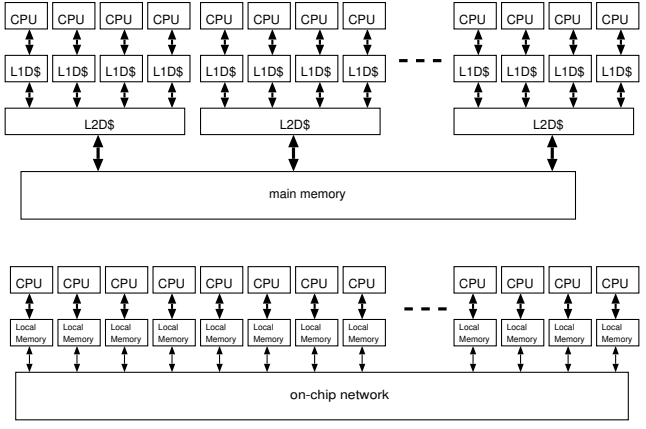


一時的にデータを置く小さいメモリ
何を置くかはハードウェアが勝手に決める=ソフトウェアからみるとメインメモリをアクセスしているだけ。
データがキャッシュにあると速い
複数コアになると非常に複雑

ある CPU が自分のキャッシュに書いたものが他の CPU が元々のそのデータのアドレスを読んだ時にちゃんと読める必要がある (キャッシュコヒーレンシ)

GPU→???

- コヒーレンシを諦めても、オフチップメモリや階層キャッシュで物理的に遠くにデータ移動させること自体が性能の制限になる
- と書いた以上、階層キャッシュを諦めて、なるべく物理的に遠くにデータ移動させないようにすることが必要になるはず
- つまり、１チップの中でも、多数の独立なプロセッサに分かれた、チップ内分散メモリにいく必要がある



何故チップ内の横方向のデータ移動が制約になるか？

配線遅延・配線消費電力の問題
微細化しても、「単位長さ辺りのキャパシタンス」はかわらない、ところが、「単位長さあたりの抵抗」は配線幅の二乗に反比例して増える:

- **配線長がスケールしても配線遅延は減らない**
- **配線長一定だと遅延は二乗で増える。電力は減らない**

(最近だと、トランジスタ密度は上がっても配線はそんなに細くならないので配線が短くなれば遅延・電力が減る。長いと電力も遅延も減らない)

数字をいれてみると

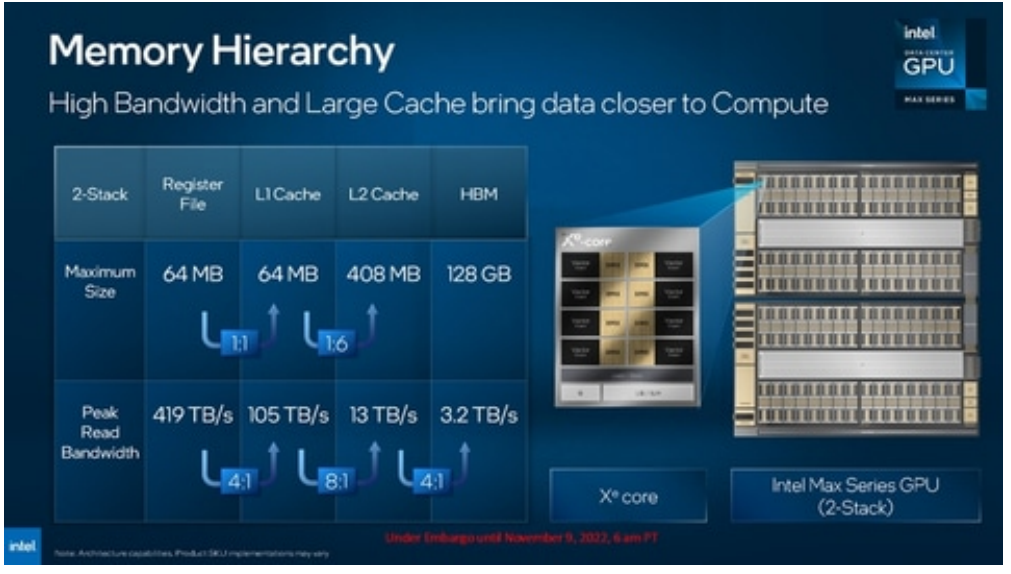
- 10cm ($10^5 \mu\mathrm{m}$) の配線は大体 20pF のキャパシタンスがある。これは微細化してもかわらない (線と平面の間のキャパシタンス)。電圧 1V だと、この線は 10pJ/bit を消費する
- DDR5 メモリは 20pJ/bit くらい。電圧 1.3V とか。
- LPDDR5 と GDDR6x: 10 pJ/bit くらい。配線が DDR5 のモジュールより短くて電圧も低い
- HBMx: 3-4 pJ/bit。概ね 25mm くらいまで配線短くなっている

最先端の GPU だと、メモリアクセスだけで消費電力の半分くらいになる
64bit のデータを 10 cm 動かすと 640pJ。最新の GPU だと 64 ビット浮動小数点演算 1 つに 20pJ くらい。データの横方向移動の消費電力はものすごく大きい。

Intel GPU の例

B/F: 1 演算の間に何バイトのデータを移動できるか

B/F の値



レジスタ: 8? (普通 16 いる。行列乗算器だと 8 はありえる)

L1: 2

L2: 0.25

メモリ 0.06

非常に小さい

それでも電力消費が大きい原因になっている

設計目標は 1.6GHz 動作だったが、熱の問題で 900MHz 程度の動作になった模様。

最新の GPU の状況

	FP64 性能 (TF)	L2 バンド幅 (B/F)	メモリバンド幅 (B/F)
V100	7.8	0.32	0.12
A100	19.5	0.25	0.08
AMD MI250X	47.9	0.27	0.07
H100	34	0.17	0.10
Intel MAX	52(31)	0.25	0.06

- L2 バンド幅が極端に小さくなっている。 0.2 前後だと L2 にのるアプリケーションでもメモリアクセスリミットなら実行効率は 5% 程度。
- CPU に比べてメモリバンド幅・演算性能の絶対値は高いが、キャッシュでも B/F は低く、アプリケーション実行効率が高くならない。

ポストGPUアーキテクチャ

ポストGPUアーキテクチャでは

- チップ内部の演算器が物理的に共有するオフチップメモリはもたない
- チップ内部の演算器が物理的に共有するキャッシュはもたない

必要がある。これは1990年代前半の共有メモリベクトル並列から分散メモリ超並列への移行と同じ。

つまり、チップ内で分散メモリ超並列を実現すればよい。

但し、普通にオフチップDRAMをつけたのでは駄目(バンド幅が足りない)

システムレベルアーキテクチャとチップレベルアーキテクチャの進化の類似

システム	チップ	
CDC 7600 まで	Intel 80860 まで	1 パイプ
Cray 1 から T90	Intel KNL まで	複数パイプ共有メモリ
Stanford DASH/SGI Altex	GPUs?NUMA CPUs?	分散共有メモリ
Cray T3D から	???	分散メモリ

分散メモリシステムへのチップレベル対応物がまだない。

ここはどうなるか？の話の前に深層学習向けプロセッサと MN-Core の話を。

深層学習向けの計算機

- ニューラルネットワークの１層は「行列ベクトル積」
- 画像認識で主流の「畳み込みネットワーク」では「行列行列積」
- 計算精度はあんまりいらない。最近は８ビット表現、さらに４ビット表現も使われる
- トランスフォーマーでも「行列行列積」が主要な計算。但し、行列の幅はバッチサイズ=推論ではあんまり大きくできない。また、一部の計算はバッチサイズに依存せずに行列ベクトル積=非常に高いメモリバンド幅を必要とする。

現在の最先端

- NVIDIA H100 (2022 年発表)
- 16 ビットで 4×4 の行列同士の乗算の専用回路、8 ビットだとその 2 倍の性能になるプロセッサ
- 1 チップで 1 Pflops 程度を実現。(「富岳」の 1 チップは 11Tflops なので、100 倍近く速い)
- 通常の 64 ビットだと 50Tflops 程度。
- 今年 B300 がでた。性能 4.5 倍だがプロセッサダイが 2 個はいっていて電力も 2 倍近い。

以下、Preferred Networks でやっている話を

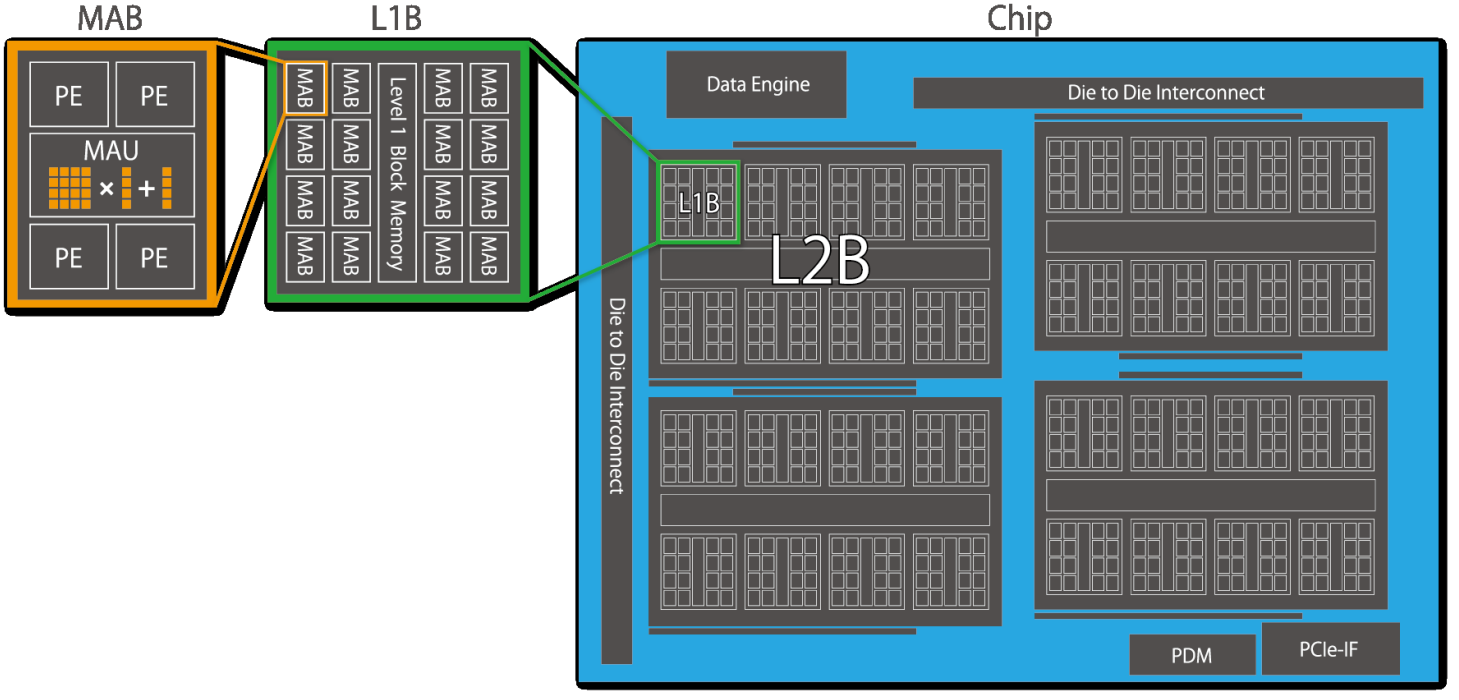
MN-Core

- PFN と神戸大学で開発した深層学習向けプロセッサ
- 深層ニューラルネットワーク (DNN) の学習で世界最高の電力あたり性能を実現
- 2020 年に完成
- FP16 (ライク) フォーマットでピーク 524TF
- 電力消費 (チップレベル) 500W 以下、1.2TF/W
- 同じ半導体技術の NVIDIA V100 の 2.5 倍以上の電力あたり性能を実現した

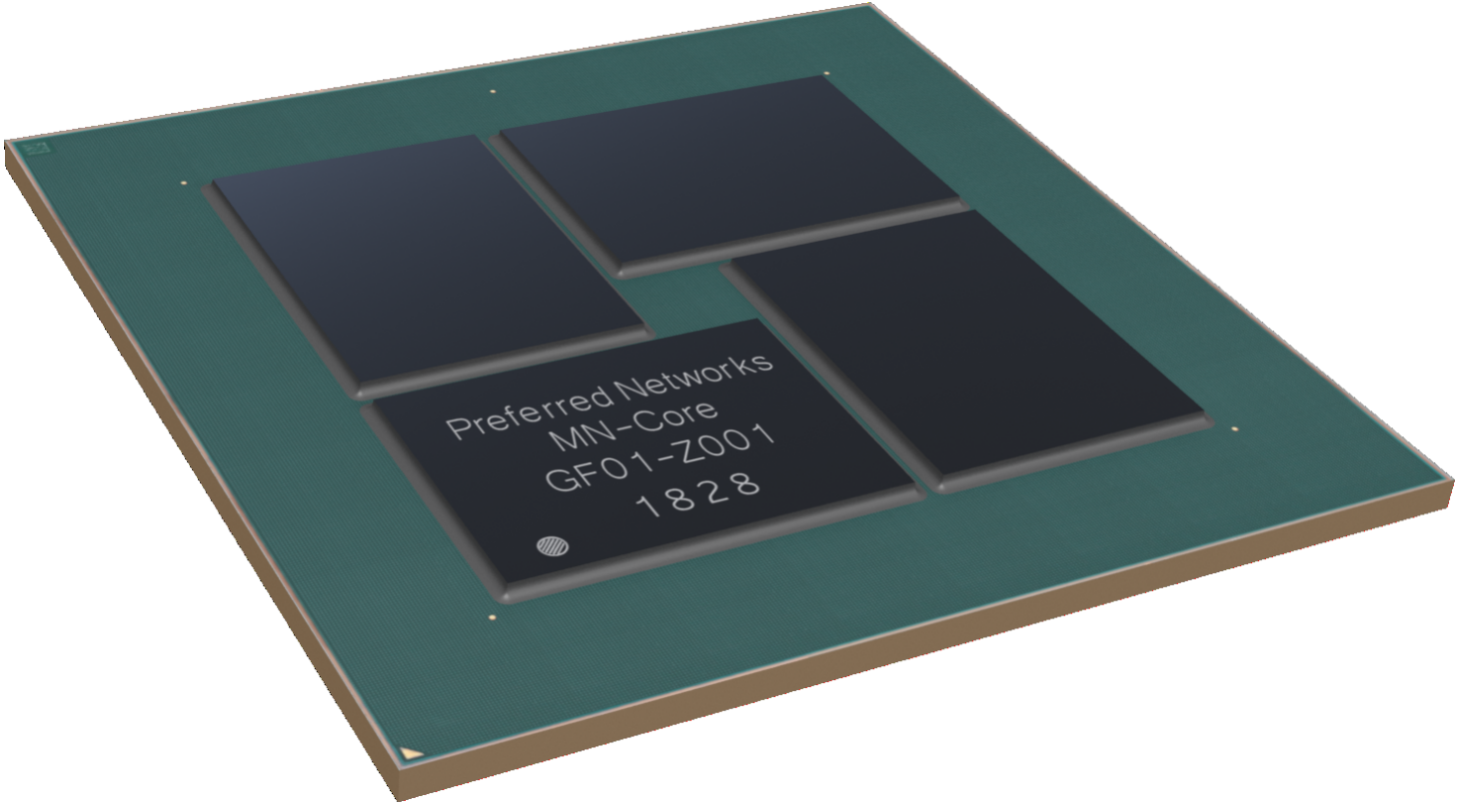
MN-Core 概要

- 1 カード: 1「モジュール」
- 1 モジュール: 4 ダイ MCM
- 1 チップ: PCIe (gen4, x16), xDDRx メモリ (内緒) 4 “Level-2 放送ブロック” (L2Bs)
- 1 L2B: 8 L1Bs
- 1 L1B: 16 MABs (Matrix Arithmetic Blocks)
- 1 MAB: 4 Processor Elements。これに対して 1 つ「行列乗算ユニット」
- FP64:FP32:FP16 性能比は 1:4:16
- カード全体が SIMD で動く。命令ストリームは 1 つ。

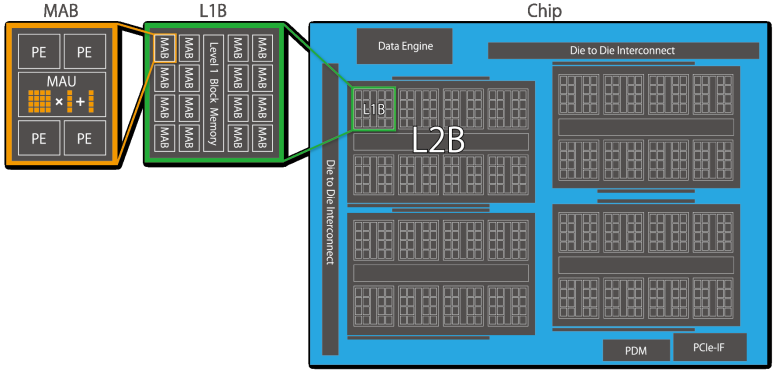
MN-Core構成図


MAB
L1B
Chip
PE
PE
MAU
PE
PE
MAB
Level 1 Block Memory
Data Engine
Die to Die Interconnect
L1B
L2B
Die to Die Interconnect
PDM
PCIe-IF

MN-Core



チップ階層構造



- トップレベル: 4 チップ内に 4 個(合計 16 個) の L2B(レベル 2 放送ブロック)
- L2B の中: 8 個の L1B(レベル 1 放送ブロック)
- L1B の中: 16 個の MAB(行列演算ブロック)
- MAB の中: 1 つの MAU(行列演算ユニット) と 4 個の PE(プロセッシングエレメント)

MN-Core: 1 ボードに PE 8192 個。MAB 2048 個。ポスト富岳の現在の案は PE あたりの性能を 2 倍に強化、クロック 5 倍、PE 数 1.5 倍くらい。

チップ階層構造続き

- 概念的には全体が同期して単一の命令流を処理する。 L2B 以下は実際にクロック同期。チップ間を含めて L2B 間は (周波数は同じだが) 命令実行にずれがあることを許す。
- チップ毎に DRAM ブロックがある。DRAM は L2B とブロック転送でデータ交換できる。放送、個別転送、チップ内分配等いくつかのモードがある。

通常のアーキテクチャとの違い(1)

- キャッシュ階層がない、ローカルメモリだけのアーキテクチャ(DRAM はオフチップでデータ転送命令でアクセス): メモリアクセスの電力消費が小さい
- ロード・ストアアーキテクチャではなく、メモリとレジスタの区別はポート数だけ: 演算に対する「無駄な」メモリアクセスを削減
- 複数のメモリ要素を ISA として見せる: 面積・電力の大きいマルチポートレジスタファイルをなくす
- 大規模 SIMD アーキテクチャを採用: コア間の同期オーバーヘッドをなくし、細粒度並列化を可能に
- 放送・縮約をチップ内ネットワークレベルでサポート: 細粒度並列の性能向上に大きく貢献

通常のアーキテクチャとの違い(2)

- 現在の通常の CPU では、レジスタリネーミングと OoO 実行によって、最内側ループに本来ある並列性を実行時に回復する必要がある。これは、通常はアーキテクチャレジスタの数が不足し、コンパイラによるアンロールだけではパイプラインを埋められないからである (「京」は例外)
- MN-Core ではアーキテクチャレジスタ数の制限がないため、レジスタリネーミングが不要となる。
- さらに、固定長ベクトル命令の採用により、OoO 実行自体が不要になっている。直前の命令の実行結果が必ず利用可能だからである。
- かなり単位の大きな SIMD 実行をすることで、非常に長い命令語を許しており、レジスタのアドレス長等の制限が不要になっている。

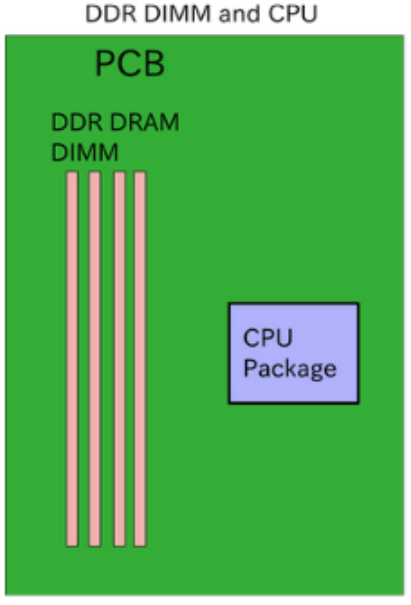
とはいえ、、、

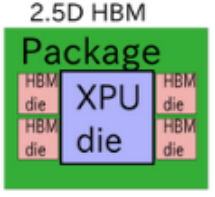
- 大容量の DRAM は依然としてチップ外部。このバンド幅はあまり大きくない。
- DRAM 転送バンド幅をあまり必要としないアルゴリズム・アプリケーションでないと性能がでない。 CNN はそうだが LLM はそうではない。
- 「真の」オンチップ分散メモリとはいいがたい。

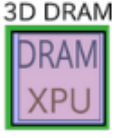
「真の」オンチップ分散メモリは？

- HBM は DRAM の平面的実装としてはほぼ究極。プロセッサダイのすぐ隣に DRAM ダイがある。
- それでも、電力消費はプロセッサダイと DRAM ダイの中を横に動く分、下のインターポーザの中を横に動く分が大きい。また、どうしても共有メモリアーキテクチャになる。
- 分散メモリにするは、DRAM ダイをプロセッサダイの上にのせて、さらに面的な配線で演算器とその直上(下？)の DRAM メモリブロックを接続すればよい。

どうするべきか?

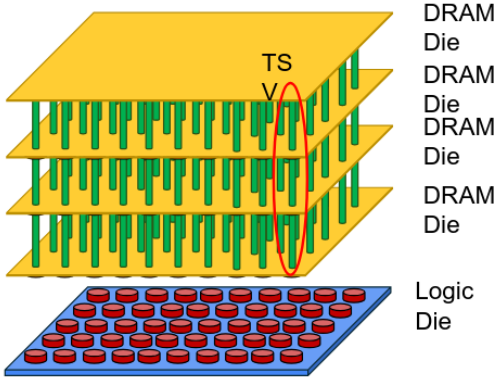






3D DRAM、つまり、DRAM をプロセッサダイの上に載せる (か、下に置く) 必要がある。
メモリ階層を変える必要もある。

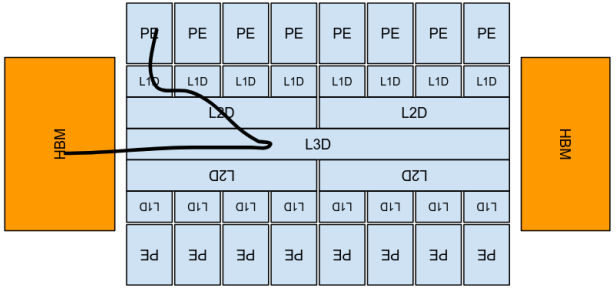
3D 積層メモリ

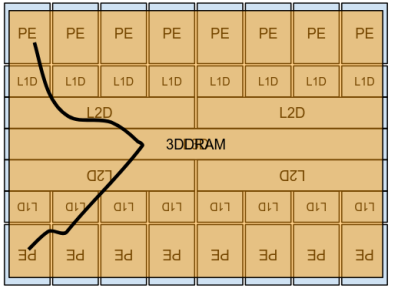


複数の DRAM ダイとロジックダイを非常に多数の TSV とパッドでつなぐ
マイクロバンプか「ハイブリッドボンディング」が使われる
いくつかの (3 大ベンダでない、、、)DRAM ベンダが技術開発している。
台湾: Powerchip, Winbond, Nanya
中国: CXMT, YMTC, XMC.

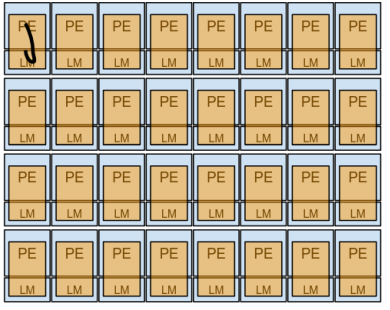
コンピュータアーキテクチャの歴史で、主記憶バンド幅が大きくジャンプする 2 回目の (最後かもしれない) 機会
最初: 磁気コアメモリから半導体メモリへの変化。

積層メモリと共有メモリ、分散メモリ





共有メモリだとデータ移動距離があんまり減らない。



分散メモリだとデータ移動距離を最小化できる。

つまり: 意味がある「オンチップ分散メモリアーキテクチャ」を考えられる。

システムレベルアーキテクチャとチップレベルアーキテクチャの進化の類似

システム	チップ	
CDC 7600 まで	Intel 80860 まで	1 パイプ
Cray 1 から T90	Intel KNL まで	複数パイプ共有メモリ
Stanford DASH/SGI Altex	GPUs?NUMA CPUs?	分散共有メモリ
Cray T3D から	オンチップ分散メモリプロセッサ	分散メモリ

分散メモリシステムへのチップレベル対応物が存在する (かも)

3D 積層って本当に上手くいく？

- 歴史的には何度か試みられては消えた。最近の例: Wide I/O。
- Wide I/O が消えた理由: HBM にバンド幅で負けた。LPDDRx にコストで負けた。電力でのメリットがあんまり大きくなかった。
- つまり、当時のロジックの性能とメモリバンド幅要求ならまだ HBM や LPDDR でよかった。
- 現在は電力・バンド幅の要求がずっと厳しくなり、3D 積層以外の解がなくなってきている。
- また、ハイブリッドボンディングで非常に多数の接続を高い信頼性で実現できるようになった。熱の問題も改善された。
- つまり、3D 積層の必要性が高くなり、技術的にも現実的になった。

原理的な限界は？

- HBM 等でも、 DRAM 側の現状の設計は DDRx の DRAM と同等。クロックがはいり、FIFO やセレクタを通してデータがでてくる。部分書き込みのためのデータ保持回路等もある。
- DRAM チップ内でみても、実際の DRAM セルやセンスアンプの消費電力は結構小さい (O’Connor et al. Fine-Grained DRAM: Energy-Efficient DRAM for Extreme Bandwidth Systems, MICRO-50)
- ハイブリッドボンディングでは非常に大きなパッド数が可能であるので、回路構成から再検討が可能。
- 0.1-0.2 pJ/bit くらいまではそのうちいけそう。

実装上の懸案事項

一杯ある。

- HBM みたいに DRAM スタックできるし、 TSV middle とかですごく沢山パッドだせるんだけど、DRAM スタック上につんで冷却できるか？
- face up でロジックダイ使って電源とか信号線引き出せるか？
- DRAM スタック下においたら、DRAM スタック通して信号とか電源だせるか？

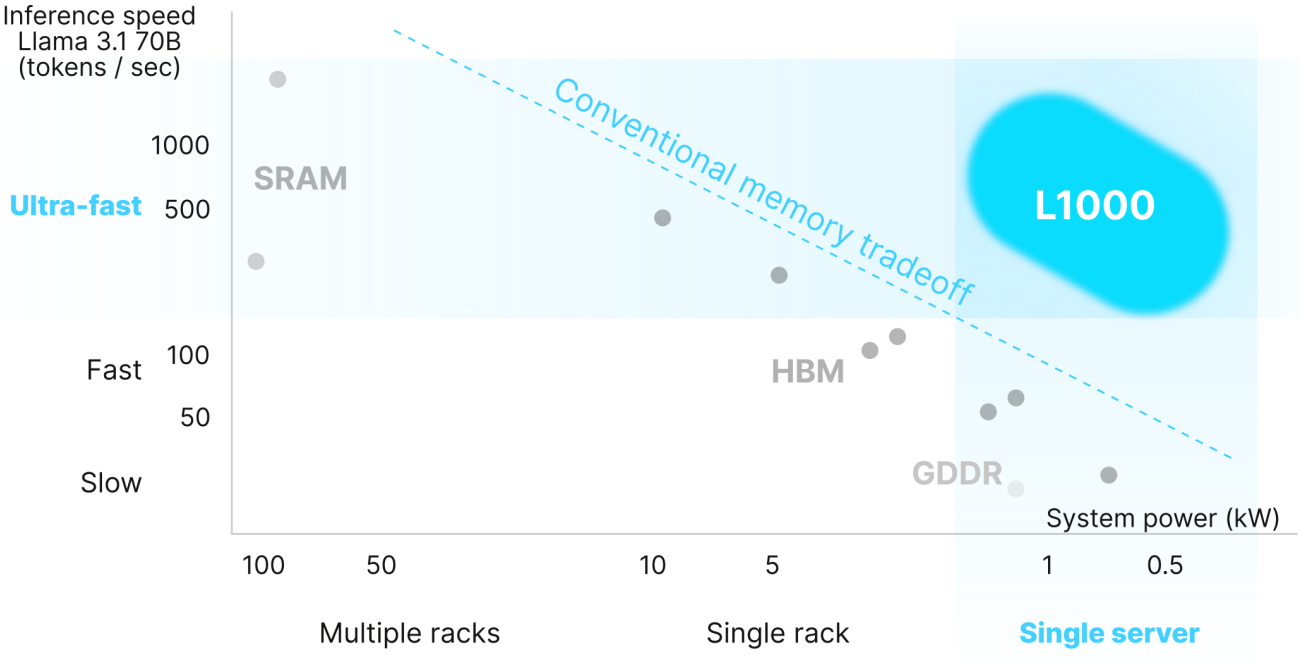
とはいえ、解決できない問題という感じでもない。物理的に金属配線でつながるので微細化できれば数が増えてバンド幅があがる。(磁界結論や容量結合はスケールしない)

MN-Core アーキテクチャと 3D DRAM

- MN-Core アーキテクチャはチップ全体が SIMD アレイ。それぞれの PE がローカルメモリをもって、階層的なネットワークで結合。
- 3D DRAM は、基本的には各 PE に DRAM ブロックがつく。例えば 16MB とか。
- 1980 年代風の超並列 SIMD プロセッサとほぼ同じ。ネットワークはさぼっている。概ね同じようにプログラミングもできる。
- つまり: ほぼ HPF (OpenACC ともいう) でプログラムできる。(但し、DRAM とオンチップメモリの違いを意識する必要はある)

PFN の計画

2026 年に、現在の GPU 等に比べて同じ電力で 10 倍の LLM 推論性能をもつ MN-Core L1000 を利用可能にする
電力性能だけでなく価格性能比でも世界最高を実現する



https://mn-core.com

他社の AI むけプロセッサは？

- 昨年の HotChips 24: 口頭発表 24 個のうち 14 個が AI アクセラレータ。
- Tenstorrent Blackhole、SK Hynix PiM、Blackwell、Sambanova SN40L、Intel Gaudi 3、AMD MI300X、FuriosaAI RNGD、AMD(Xilinx) Versal、Onyx (Stanford の研究)、Meta MTIA、Tesla Dojo、Cerebras CS-2、MS MAIA、PFN MN-Core 2
- Blackhole、SN40L、Onyx、MTIA、Dojo、Cerebras、MAIA: 2D オンチップネットワークに MIMD コア+行列演算ユニット。
- Gaudi、RNGD: 大きな行列乗算器もつ独自アーキテクチャ(Groq も)
- SK Hynix は GDDRx メモリチップの中に演算器いれるもの
- Versal は FPGA、あとは Nvidia, AMD, PFN

アーキテクチャの特徴

- 階層キャッシュはもうないというのは Nvidia と AMD の GPU 以外では共通。
- AI プロセッサアーキテクチャはほとんどが MIMD 2D メッシュ。メッシュの辺のところに HBM つける。
- どうやってプログラムを書くのか?というととりあえずとても大変そう。
- AI 用プロセッサは FP64 ない。 FP32 も絶滅しつつある。Nvidia、AMD の GPU も FP64 の性能が相対的に下がってきている。(MN-Core は異なる精度間で演算器を共有しているので比較的 FP32, FP64 の性能を維持しやすい)
- HBM なのはみんな同じ。次世代では HBM4 ではなくカスタム HBM(チップ間接続に UCIe 等を使う) がでてくるかも。 3D DRAM には時間がかかりそう。

まとめ

- 計算機の進歩について、「ポストムーア」は今度は本当。シリコン半導体技術の限界は確実にきている。
- しかし、計算機設計という観点から見ると、問題はそこではなくて、メニーコア CPU や GPU の設計が半導体技術の進歩を上手く利用できなくなってきていること。これは 30 年前にシステムレベルでの共有メモリ計算機が無理になってきたのと同じ。
- GPU アーキテクチャは階層キャッシュ構造から離れるのは難しいように見える。
- AI 向けプロセッサはキャッシュなし、ローカルメモリの方向。
- MN-Core は、さらに大規模 SIMD と階層的オンチップネットワークで高いピーク性能と AI でも汎用 HPC でも高い実行効率を実現。
- さらに、 3D DRAM を有効に利用して DRAM バンド幅をあげることが重要になる。我々はこれにも取り組んできている。

重力多体シミュレーションの話

大自由度で非常に広いタイムスケールがでてくる系の例、ということで興味があるかも、ということで。

多体シミュレーションの役割

- 観測の解釈 — 理論+モデル計算
  知られている物理法則から天体現象を理解する
- もうちょっと原理的な問題
  カオス
  熱力学

そもそもどんな対象を考えるか？

大抵の天文学的対象: 自己重力系
一つの星: 重力とガスの圧力がつりあう
それよりも大きい構造: 重力を構成要素の運動エネルギーで支える

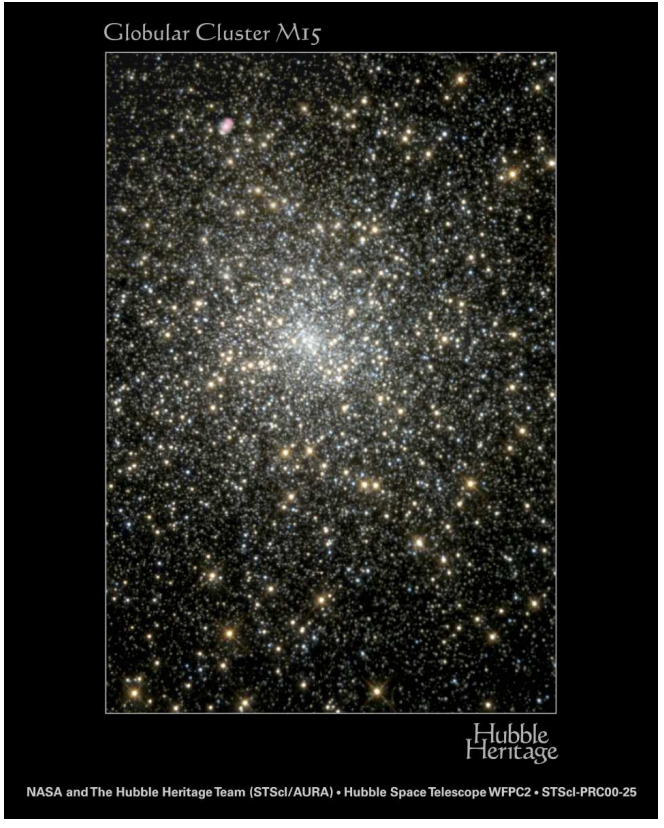
- **太陽系**
- **星団**
- **銀河**
- **銀河団**
- **宇宙の大規模構造**

例: 星団

- 球状星団
- その他、丸い星団

丸い: ある程度熱力学に緩和していることを意味する。(円盤銀河なんかとは違う)
しかし、熱平衡状態ではない(自己重力系には熱平衡状態は存在しない)

球状星団 M15



地球からの距離 10kpc (1pc: 3 光年くらい)
星の数: 200 万くらい
質量: 太陽の 50 万倍
半径 (質量の半分がある): 4 pc

球状星団 G1

アンドロメダ (M31) にある、局所銀河団で最大の球状星団

質量:
見積もり (a) 1.5×10^7 太陽質量 (Meylan et al. 2001)
見積もり (b) 8×10^6 太陽質量 (Baumgardt et al. 2003)
2倍の違い: 力学モデルの違いから。

球状星団を研究する理由

沢山あるが、、、

- 銀河でもっとも古い星の集団 — 球状星団がどうやってできたかは銀河自体がどうやってできたかと関係
- 出来てから星形成やガスとの相互作用が基本的にはない — 理想的な「自己重力質点系」に近い
- より複雑な銀河中心等を理解するための鍵
- ブラックホール連星 (重力波で観測される) が形成される場である可能性 (2010 年代以降に非常にホットな研究テーマに)

もうちょっと原理的な問題

重力だけで相互作用する質点の集まりには一体なにが起きるのか？

粒子数 $N = 2$: ニュートンが解決
$N > 2$ ポアンカレが「一般には解析解はない」ことを示した

球状星団：$N \sim 10^6$
銀河: $N \sim 10^{11}$
なにが起きるか？
数値計算なら、初期条件を与えれば答はでる。

解く方程式 — 重力多体系の基礎方程式

もとの方程式自体はもちろん、各粒子の運動方程式

$$\frac{d^2 x_i}{dt^2} = \sum_{j \neq i} G m_j \frac{x_j - x_i}{|x_j - x_i|^3}, \tag{1}$$

数値計算は基本的にはこれを使う

何故多体問題を直接数値計算するのか？

もうちょっと違う方法

- 解析的ななんか
- 分布関数に対するフォッカープランク方程式

現実問題として、、、

- 解析的には解けない
- フォッカー・プランク近似は正しいとは限らない
- 多次元ではフォッカー・プランク方程式は計算量的に解けない

で、多体問題を直接数値積分すれば、「原理的には」なんでもわかるはず。

数値計算の方法等

この種の問題：基本的に

- より大粒子数で
- より正確な

計算をすることで、「新しいことがわかる」（こともある）。

↓

どうやって今までより「良い（大きく、正確な）」計算ができるようにするかが問題

「良い」計算をする方法

基本的な事実: 速く計算できればそれだけ良い計算が可能になる。(例は時間に余裕があればいくつかあげたい)
それしか方法がないわけではないが、「速く計算できるようにする」のは極めて重要な方法。

- **計算法を改良する**
- **速い計算機を買う**
- **速い計算機を作る**

計算法

原理的には、多体シミュレーションはとっても単純：
運動方程式

$$\frac{d^2 x_i}{dt^2} = \sum_{j \neq i} G m_j \frac{x_j - x_i}{|x_j - x_i|^3}, \tag{2}$$

を数値積分するだけ。

右辺を計算するプログラム： 2 重ループで 10 行くらい
時間積分： なにかルンゲクッタとか適当なものを使えばいい

というだけで話が済めばいいけれど、もちろん世の中はそんなに簡単ではない。

何が問題か？

- 計算精度の問題：２粒子の近接散乱、自己重力による構造形成 — 時間刻みをどんどん短くしないとちゃんと計算できなくなる。積分時間が長いので高精度の公式を使いたい。
- 計算量の問題： 右辺の計算量が $O(N^2)$ — N が少し大きくなるとすぐに計算時間が現実的ではなくなる

というわけで、どんな方法を使っているかという話を簡単に。

計算法 — 時間領域

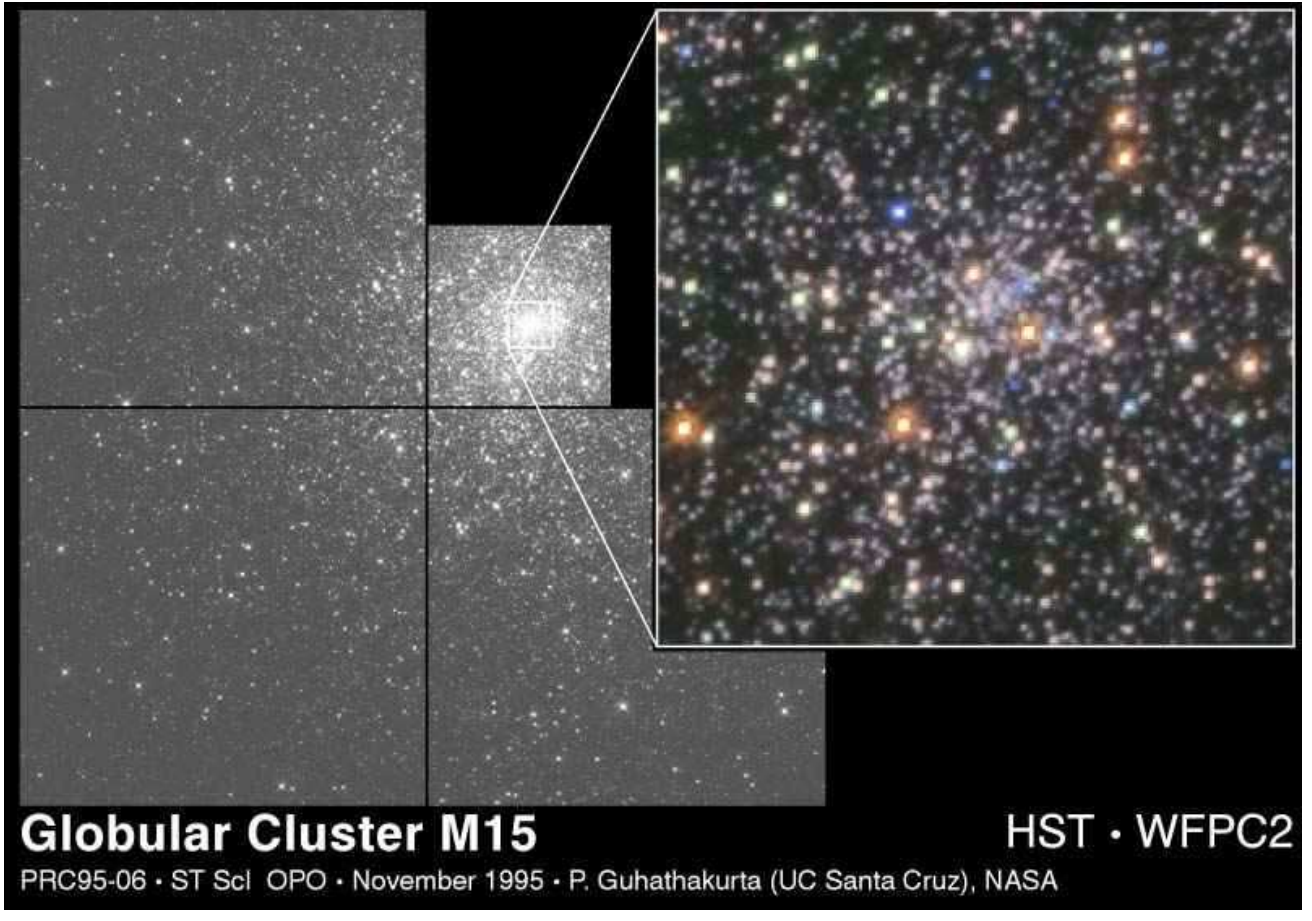
算数としては、単に常微分方程式の初期値問題の数値解。
ナイーブに考えると、いろんな公式がライブラリであるので、それを使えば済みそうな気がする。
それだけでは済まないのが問題。
済まない理由:

- 粒子によって非常に大きく軌道のタイムスケールが違うことがある
- 連星とかそういったものができる

軌道タイムスケールの問題

- 構造形成による効果
- 一様な系でも起きる問題

構造形成による効果: 球状星団の例



構造形成による効果(続き)

- 「コア崩壊型」星団 — 星の数密度が中心までべき (半径の -1.8 乗) で増加
- 中心にブラックホールがある？

星の運動のタイムスケール: 中心に近いほど短くなる。
等温カスプ (密度が半径の -2 乗に比例): タイムスケールの分布がべき乗になる。

一様な系でも起きる問題

重力が引力であるために確率的には２つの粒子がひじょうに近付くような近接散乱が起こる
インパクトパラメータが 0 に近い２体衝突: 非常に短い時間刻みが必要
これは重力多体系特有の問題: 相互作用が引力で、しかも距離 0 で発散するため。
例えば分子動力学計算ではこういう問題はおこらない。

計算量への影響

単純な可変時間刻みでは計算量が大きくなりすぎる。

理由: どちらの場合も、タイムステップの分布がべき乗的なテイルをもつようになる。

粒子数が増えるに従って、タイムステップが短くなる。
構造形成の効果: 最悪 $O(N^{1.3})$
2体衝突の効果: $O(N^{1/3})$ 程度

対応:

- 粒子毎に時間刻みをバラバラに変化させる。(独立時間刻み)
- 2体衝突、連星は座標変換して扱う。

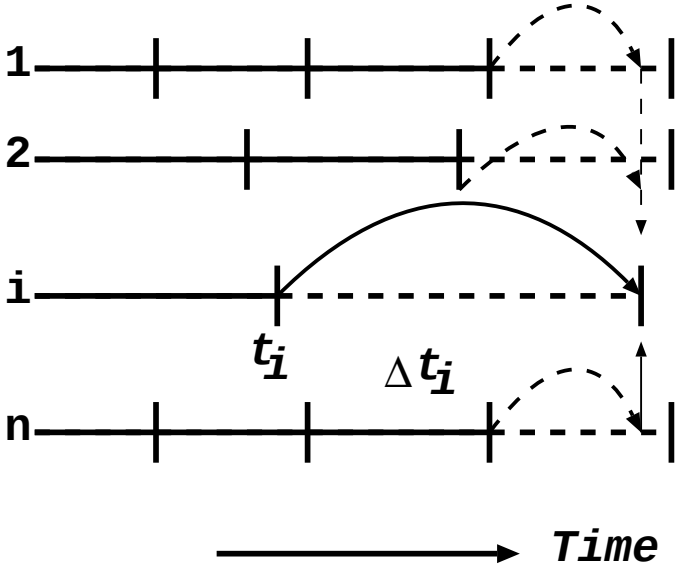
独立時間刻みの原理

粒子毎にばらばらの時刻 t_i と時間ステップ Δt_i を与える

1. $t_i + \Delta t_i$ が最小の粒子を選ぶ。
2. その粒子の軌道を新しい時刻まで積分する。
3. その粒子の新しい時間刻みを決める。
4. ステップ 1 に戻る。

ある粒子の時刻 $t_i + \Delta t_i$ で他の粒子の位置が高精度で必要:
予測子・修正子型の公式を使う。

独立時間刻み

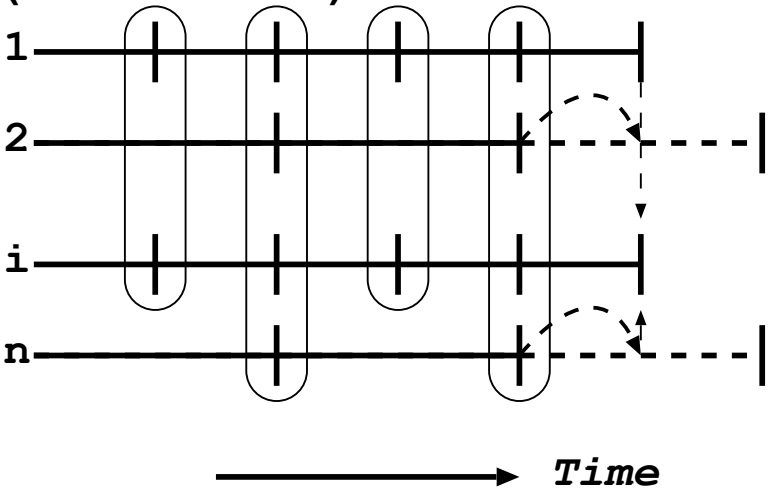


(Aarseth 1963)

- 各粒子にそれぞれ時刻と時間刻みを与える
- 「イベント駆動」時間積分 — $t_i + \Delta t_i$ がもっとも小さい粒子が積分される

ブロック時間刻み法

(McMillan 1986) ベクター／パラレル計算機のための改良



- 時間刻みを 2^{-k} に制限する。
- $t_i + \Delta t_i$ が同じ（Δt_i は違ってもよい）粒子は同時に積分される。

$O(N_c^{2/3})$ 個の粒子（N_c は高密度コアのなかの粒子数）を並列計算

— そんなに大きな数ではない。

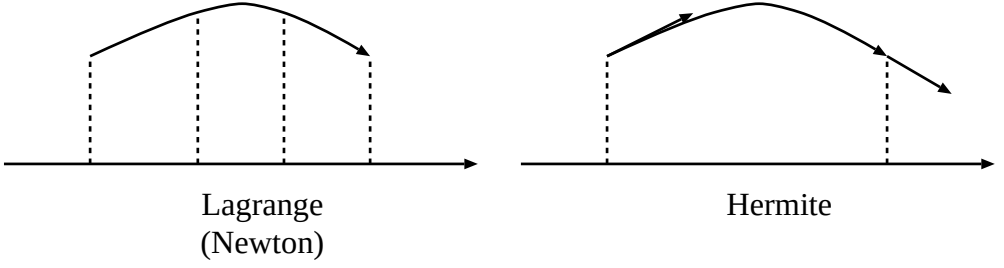
時間積分公式に対する要請

- 高次の予測子が必要(他の粒子の位置が必要)
- 可変時間刻みが必要
- 積分区間の途中で加速度を計算するような方法は使えない。
  - 線形多段階法は使える
  - ルンゲ・クッタは使えない
  - シンプレクティック法は単純には使えない

時間積分公式

次数：「4次くらいが適当」(JM 1990)。
もっと高いほうが良い: (Nitadori and JM 2007)

- **Aarseth scheme (Aarseth 1963): 可変刻みのアダムス法、PECモード、4次、2階の方程式用。**
- **Hermite scheme (JM 1990): ラグランジュ補間（ニュートン補間）の代わりに、加速度の一階時間導関数も使ってエルミート補間を構成。**



高次エルミート

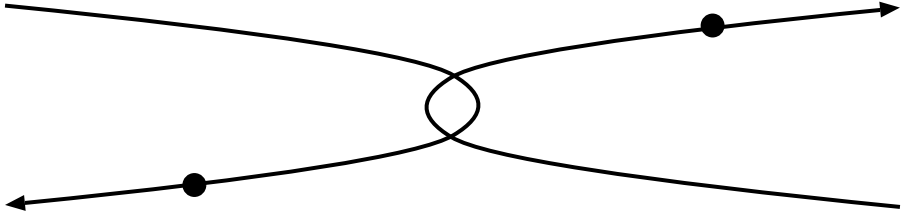
Nitadori and JM 2007

- ２階導関数まで直接計算してエルミート補間: ６次公式
- ３階導関数まで直接計算してエルミート補間: ８次公式
- 予測子は前のステップの値を使って構成

近接遭遇

連星に限らず、２つの星の距離が極端に近づくと計算が破綻する(相互作用ポテンシャルが発散するので数値誤差も発散する)。なんらかの対処が必要。

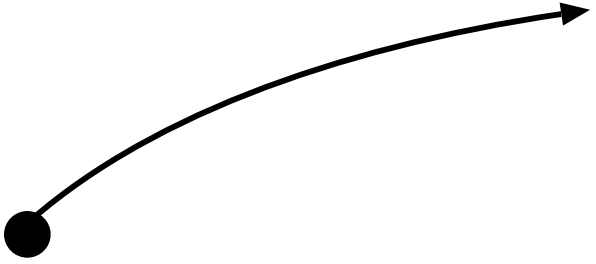
原理的には “straightforward”: ２体が十分近づいたら、解析解(またはそれ+摂動)でおきかえればいい。



具体的には

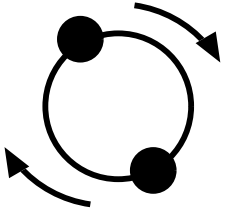
- ２つの粒子が近付いたかどうかをチェックする
- 近付いていたら、その２粒子の重心および相対位置（内部運動）を新しい変数として時間積分する
- 相対運動に対して、回りの粒子からの摂動が十分に小さければ数値積分しないで解析的に解く
- ２つの粒子が十分離れたらその重心と相対位置から元の粒子を復元する

連星



普通の星: 他の星全体が作るポテンシャルの中を運動

連星: 2つが重力的に結合
できかた:



- 生まれる時に連星 (結構多い)
- 3体相互作用

連星と計算量

連星は軌道周期が短い
連星の熱力学: ほぼ「等温」のバックグラウンドに埋めこまれた自己重力系
一旦周りよりも「温度」が上がると、周りに熱を与えて無制限に温度が上がる。
(3体相互作用作用でエネルギーを与える)
周りと温度が等しい = 軌道長半径 $\sim$ 系の大きさ$/N$
軌道周期が他の星の $1/N$
軌道周期が他の星の $1/1000N$ くらいまでは系内にいられる。
こんなのを計算してたら日が暮れるどころではない。

連星の扱い

十分コンパクトな連星で、回りの粒子からの摂動が無視できるほど小さい場合: 要するに単に摂動を無視すればよい

どれくらい摂動が小さいなら無視していいか? —目的による
エネルギー: 断熱不変量
角運動量: そうではない

角運動量について正確な結果を得るのは現状では難しい

弱い摂動の扱い

まともにやるなら、軌道要素で表して、外力を軌道平均して、、、

画期的に安直な方法 (Mikkola and Aarseth 1996)

- 摂動論的扱いができる範囲内で連星の軌道周期を遅くする
- 摂動項は、実時間での軌道要素の変化率が同じになるようにスケールして運動方程式に入れる。

摂動論の範囲内では摂動論と同じ結果を、軌道を直接数値積分して得ることができる。
(平均運動共鳴は扱えない)

うまくいかないケース

階層的な３重星: 連星の外側をもう一つ星が回っているようなもの

- 安定条件は数値的にしかもとまってなく、しかも幅がある。
- 共鳴が効くので時間のスケーリングはできない

いろいろ研究中

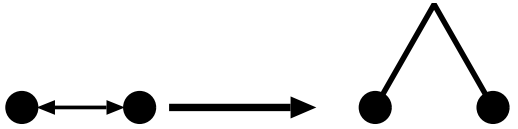
２体以上の系の扱い

例えば３体の系を重心運動と内部運動に分離するにはどうするか？

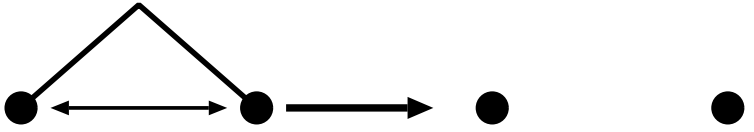
- 「3体」として認識、それ用の正則化
- ２体+1体としては認識、再帰的に扱う

今のところ広く使われているコードは前者
後者のコードもそれなりに動くようになってきている

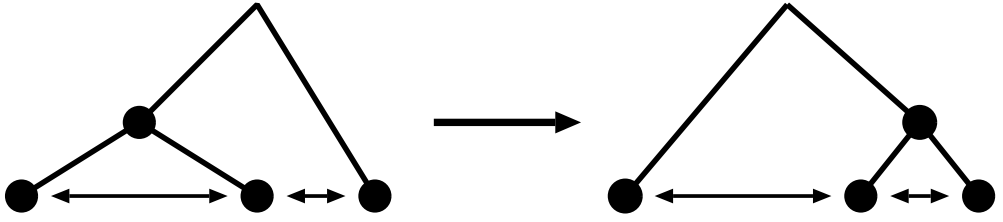
構造の組み換え



$r < r_{close}$



$r > \gamma r_{close}$



計算法 — 空間領域

運動方程式の右辺をどうやって評価するか？という問題。

以下、独立時間刻みのことはとりあえず棚上げにして話をすすめる。

広く使われている方法： Barnes-Hut treecode
有名な方法： 高速多重極法

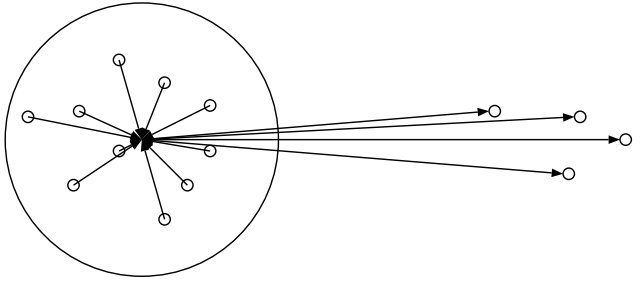
どちらも 1980 年代後半に発表

ツリー法、FMMの基本的発想

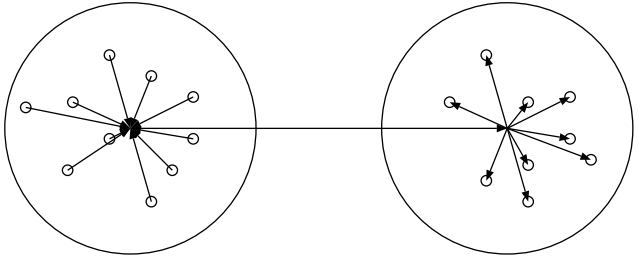
遠くの粒子からの力は弱い

↓

まとめて計算できないか？



Tree



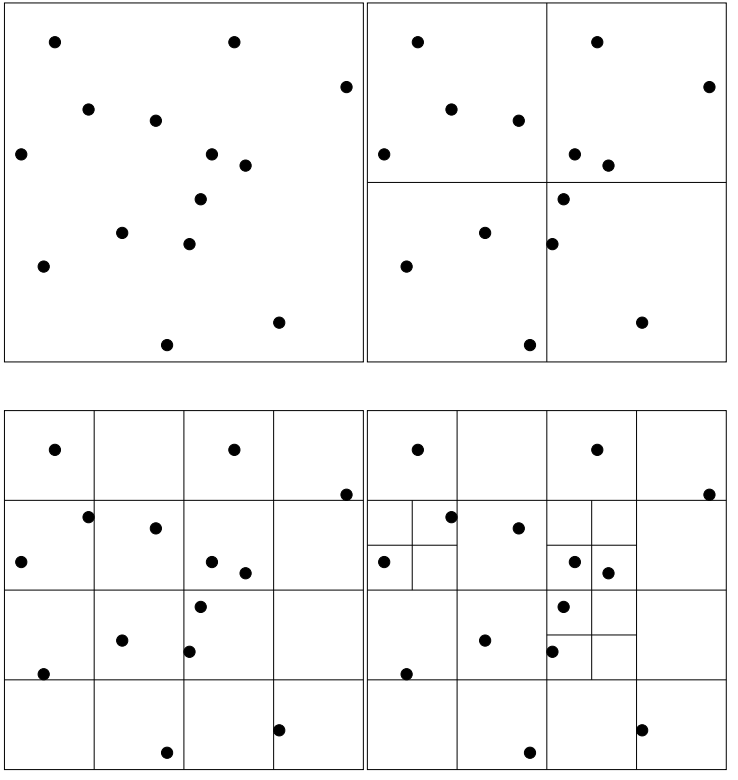
FMM

- ツリー：力を及ぼすほうだけをまとめて評価
- FMM：力を受けるほうもまとめて評価

どうやってまとめるか？ — ツリー法の場合

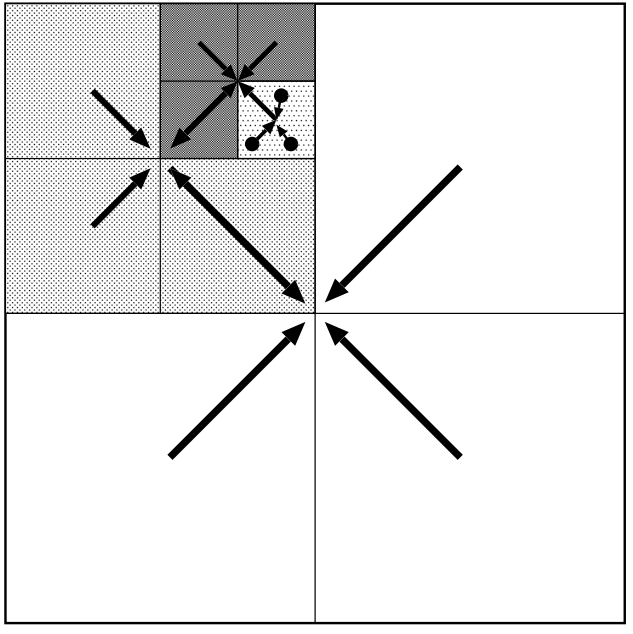
階層的なツリー構造を使う。

- まず、全体が入るセルを作る
- それを再帰的に 8 (2 次元なら 4) 分割する
- 中の粒子がある数以下になったら止める（上の例では 1 個）



多重極展開の構成

まず、ツリーの各セルのなかの粒子がつくるポテンシャルの多重極展開を計算する。

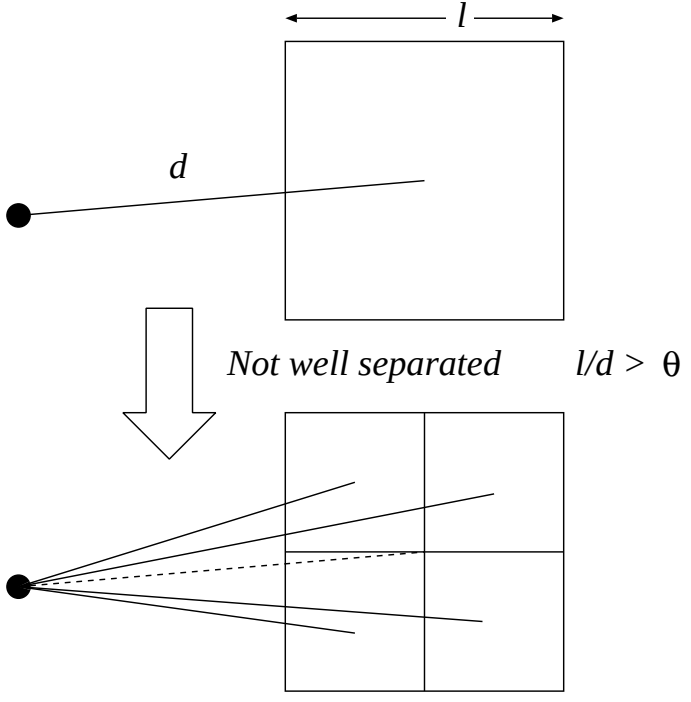


- 最下層のセル: そのなかの粒子が作るポテンシャルを多重極展開
- それ以外：子セルの多重極展開の展開中心をシフトして加算

下から順に計算していけばよい。計算量は $O(N)$。展開をシフトする式はかなり複雑。

ツリー法での力の計算

再帰的な形に表現すると格好がいい。



- **十分に離れている:重心（あるいは多重極展開）からの力**
- **そうでない：子ノードからの力の合計**

系全体からの力 = ルートからの力

ツリー法の効果

- 計算量のオーダーが $O(N^2)$ から $O(N \log N)$ に減る。
- 現状では、例えば「京」とか富岳で 100-1000 億粒子が 1 ステップ 1 分とか。
- もしも直接計算したら、、、1 ステップが年単位。

独立時間刻みとツリー法の組合せ

- 原理的にはこれが望ましいに決まっている
- 研究も昔からある。McMillan and Aarseth 1993 とか
- (牧野は 1987 年あたりに色々やったけど論文書いてない)
- あまり上手くいっていなかった

問題点:

- 昔の殆どの実装は、一番短いステップ毎にツリーを作り直す。そうすると、ツリーを作る時間が全部になって速度があがらない。
- 部分的にツリーを作り直すとかも試みられているが、特に並列化と組み合せるとコードが複雑になりすぎて手に負えない。

並列化向けのアプローチ

例えばこんなのを扱いたい
銀河中心近くでの高密度星団の進化

- 星団を銀河に埋め込む
- 星団内は精度が欲しい
- 銀河はまあ適当でもよい
- BRIDGE scheme (Fujii et al 2006)
- P^3T scheme (Oshino et al 2011)

BRIDGE

MVS (混合変数シンプレクティック) に類似

単純な陽的シンプレクティック: ハミルトニアンを運動エネルギーとポテンシャルに分解、交互に積分

MVS: 惑星の運動を、太陽の回りのケプラー運動とそれ以外の相互作用に分解。

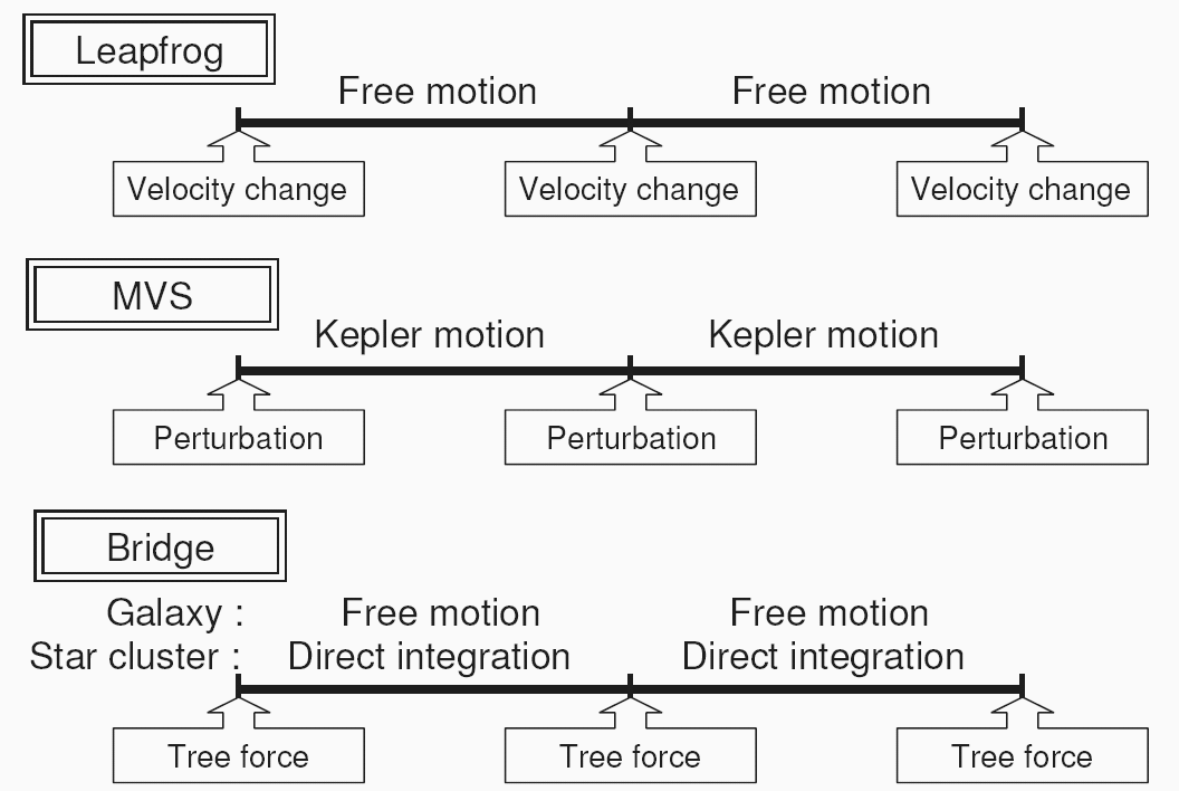
我々の方法: ハミルトニアンを

- 運動エネルギーと星団内ポテンシャル
- それ以外のポテンシャル

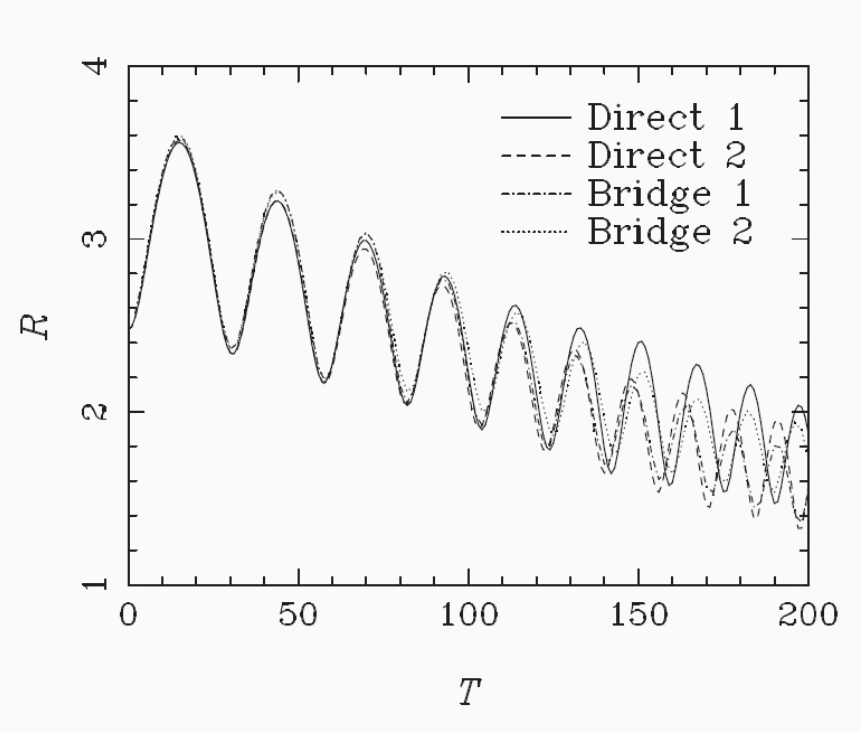
に分解、前者を (シンプレクティックでない) 独立時間刻みで高精度に積分

BRIDGE (Bridge is for Realistic Interactions in Dense Galactic Environment)

How does it work?



Test result



- $N = 100\mathrm{k} + 2\mathrm{k}$
- **Similar model as in Fujii et al. 2996**
- **Two runs: different random seeds**
- **Results agree well.**
- **Energy error: dominated by the parent galaxy.**

もうひとつの方法

BRIDGE は銀河+星団１つだと素晴らしく上手くいくが、限界もある

- 星団複数だと計算大変
- 実際に銀河の中で星団が生まれてくるような現象には使いにくい

というわけで、

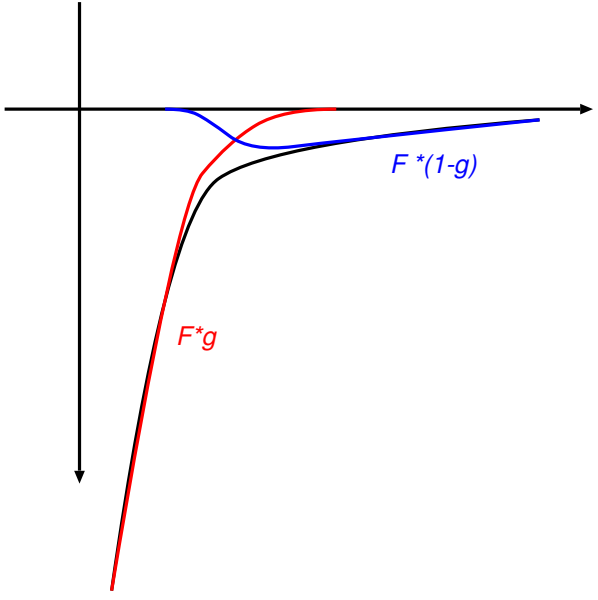
- 粒子の種類によってではなく、距離でわけたらどうか？

具体的には

2粒子間の重力を形式的に2つのタームに分ける。

$$F_{ij} = -Gm_i m_j \frac{r_{ij}}{|r_{ij}|^3} = F_{ij}(1 - g(|r_{ij}|) + F_{ij}g(|r_{ij}|)$$

こんな感じに分解



- $F * g +$ 運動エネルギー を独立時間刻みで高精度に積分
- $F * (1 - g)$ はツリー+リープフロッグで積分(こっちはシンプレクティック)
- g はコンパクトサポートで、積分公式に必要な回数だけ微分できる必要あり(スプラインを使う)

現状

- ツリー法を使った粒子法の並列化フレームワーク FDPS (Iwasawa et al. 2016) を使って、実用になるシミュレーションコードができてきた。
- PETAR (Wang et al. 2020): 球状星団むけ
- GPLUM (Ishigake et al. 2020): 惑星形成シミュレーション向け
- 富岳とかを使うと100万体を超えるシミュレーションが日常的にできるようになってきた。

こちらのまとめ

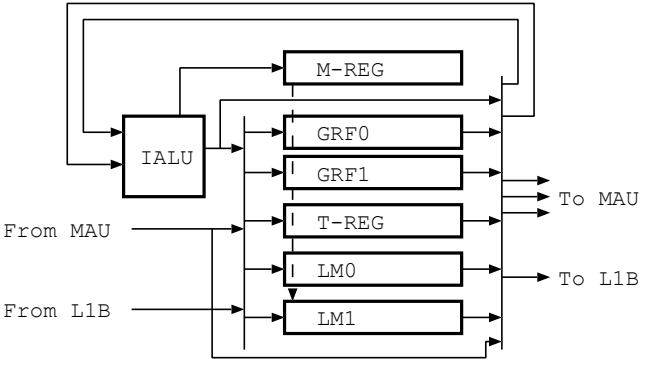
- 重力多体系、特に球状星団のような質点系に近い系の数値計算の方法論を概観した。
- 独立時間刻み、連星系の扱い、ツリー法とのハミルトニアン分割によるハイブリッド等のいくつかの方法が使われるようになった。
- 並列化フレームワークとも組合せてかなり大規模なシミュレーションができるようになってきた。

予備スライド

詳細構造

- PE(プロセッシングエレメント)
- L1B(レベル１放送ブロック)
- L2B(レベル２放送ブロック)
- 全体

PE(プロセッシングエレメント)



- 演算器は IALU と MAU
- MAU は倍精度、単精度、半精度精度の行列ベクトル積を実行
- L2B の中: 8 個の L1B(レベル 1 放送ブロック)
- L1B の中: 16 個の MAB(行列演算ブロック)

- バスみたいに書いてますが実際の回路はマルチプレクサ
- GRF は (MN-Core では) 1R1W 2 ポート。LMx はシングルポート。
- T-reg: 補助レジスタ。1R1W で 1 ベクトル分
- LMx (ローカルメモリ) は 2048 64 ビット語 x 2、 GRF は 512 語
- ベクトル長 4 の固定長ベクトル命令。

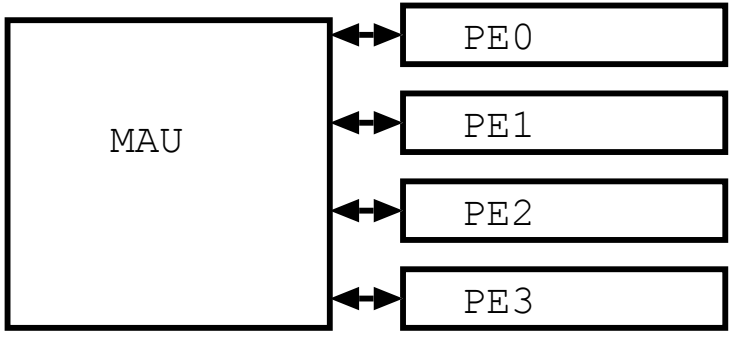
PE(プロセッシングエレメント) 続き

- 直前の命令の演算結果を入力オペランドにとることが可能 (フィードバック)
- ベクトル命令でのアドレッシングは固定、連続アドレス、ストライドアクセスが可能 (ベクトルレジスタもストライドアクセス可能)
- T-レジスタを使った間接アドレスも可能
- アドレスはベースアドレスレジスタで修飾
- IALU の演算結果から大小比較、一致比較等のフラグベクトルを生成して M レジスタに格納可能
- M レジスタの値でメモリ、レジスタに書き込むかどうかを制御

PE(プロセッシングエレメント) 続き

- ロードストアアーキテクチャではなく、LMx の読み出し結果を直接演算器の入力にできる。また、演算器の出力をメモリに直接格納できる。複数のユニットに格納もできる。
- B/F が極端に大きい。(ベクトル演算モードに対して 8 ある)。このため、ローカルメモリにデータがあればほぼ理論ピーク性能がでる。
- この機能と、演算結果フィードバック、T レジスタを最大限利用することで、通常のアーキテクチャに比べてレジスタファイルのハードウェア規模、消費電力を大きく削減している。また、スーパースカラー実行制御、OoO 実行のための回路も不要となる。=高い電力性能の実現に大きく貢献

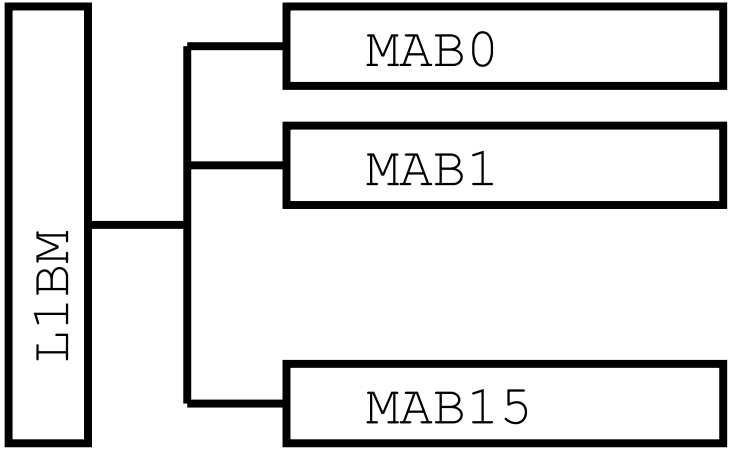
MAB(行列演算ブロック)



- PE4 個が 1 つの行列乗算器を共有
- サイクル毎に行列ベクトル積を実行。
  d= A*b +c
- MAU は行列レジスタをもつ

- 行列レジスタは複数あり、演算中に別の面への書き込みが可能
- 転置しながら格納も可能
- 倍精度、単精度、半精度行列に対応。サイクルあたり演算数は 32, 128, 512
- ベクトル演算 (PE から見ると 64 ビットスカラー演算) モードもある
- 倍精度、単精度、半精度行列乗算器は同一の乗算器の構成変更で実現 (特許取得済): 他社の回路に比べると同一性能で回路規模が半分以下

L1B(レベル1放送ブロック)



- MAB 16 個が 1 つの L1BM(レベル 1 放送メモリ) に接続。
- L1BM から読み出されたデータは放送される (各 PE/MAB での書き込み制御で単一の PE/MAB に送ることもできる)。
- 分配モードも一応ある (あんまり速くならない)

- 各 PE から読み出したデータは、縮約して L1BM に書き込むことができる。単一の PE/MAB を指定した読出しもできる。分配の逆操作 (結合) も一応ある。
- PE 間の直接結合はない。

L1Bの特色

- 明示的な放送/縮約モードがあるため、複数PEを使った細粒度並列処理を低オーバーヘッドで実行できる。
- GPU が非常に苦手とする reduction をオーバーヘッドを気にしないで使うことができる。
- 例えば比較的小さな行列乗算を MAB に分散させることができる。総和が遅いと、行列行列積 A*B で B を縦に切る並列化をすると、各MAB で A は全体が必要になる。総和が速いと、Bを横に切る並列化ができ、Aを分散させられる。また、単一の行列ベクトル積の低オーバーヘッドの並列化ができる。推論(LLMでも)で非常に効率をあげやすくなる。
- (PE 命令と同期した、4サイクル毎のデータ転送命令で制御)

L2B(レベル２放送ブロック)

- L1B ８個に対して L2BM(レベル２放送メモリ) が１つ。
- L1B 内と同様な放送・縮約・分配・結合・個別読出しを L1B に対して行う。
- データ幅は広くしている。
- いくつかの L1B 同士の直接データ交換モードがある。
- PE 命令と同期した、４サイクル毎のデータ転送命令で制御。

全体

チップ 4 個を全体としてみると

- L2B 16 個、DRAM インターフェース 4 個、PCIe インターフェース 4 個がある。
- PCIe インターフェースはバッファメモリ (PDM) を持つ
- PDM-DRAM、 PDM-L2B、 DRAM-L2B で、放送・縮約を伴うデータ転送を実行可能。
- このレベルのデータ転送命令は非同期で実行される。PE 命令とはお互いにタグ待ちができる。

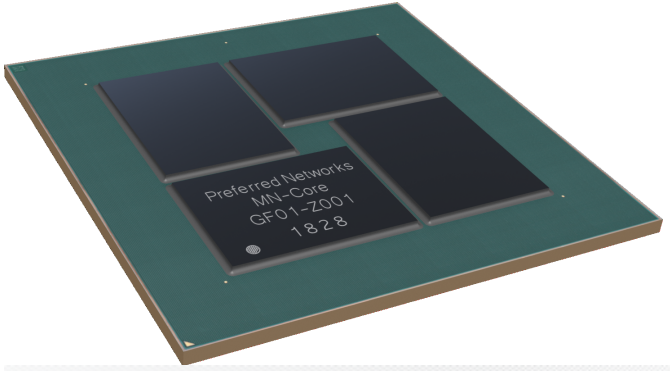
PE 命令の概要

PE 命令は

- 各演算の動作モード、入力セレクト、各メモリユニットのアドレス、LM では read/write モード、入力セレクト、各メモリユニットの入力マスクレジスタ番号、書き込みマスクレジスタ番号等
- L1BM-PE, L2BM-L1BM の転送モード、転送アドレス等

を指定する。

MN-Core/MN-3 system



The GREEN 500 CERTIFICATE

MN-3 - MN-Core Server, Xeon 8260M 24C 2.4GHz, MN-Core, RoCEv2/MN-Core DirectConnect

Preferred Networks, Japan

is ranked

No. 1 in the Green500

among the World's TOP500 Supercomputers

with 21.11 GFlops/Watt Linpack Power-Efficiency

on the Green500 List published at ISC 2020 Digital Conference, June 22nd, 2020

Congratulations from the Green500 Editors

Wu-chun Feng
Virginia Tech

Kirk Cameron
Virginia Tech

PFNのスパコン「MN-3」が世界1位に、消費電力性能ランキングのGreen500で

岡林 凜太郎 日経クロステック／日経コンピュータ

2020.06.23

PR

Preferred Networks（PFN）のスーパーコンピューター「MN-3」が2020年6月22日（欧州時間）、スーパーコンピューターの消費電力性能ランキング「Green500」で世界1位を獲得した。HPC（ハイ・パフォーマンス・コンピューティング）に関する国際会議「ISC 2020 Digital」が同日ランキングを発表した。

PFNのスーパーコンピューター「MN-3」
（出所：PFN）
[画像のクリックで拡大表示]

MN-3はPFNが独自開発した深層学習用プロセッサー「MN-Core」を使ったスーパーコンピューターだ。PFNのスーパーコンピューター「MN-2」の後継機で、2020年5月に運用を始めた。160個のMN-Coreを搭載し、1ワット当たり21.11ギ

MN-Core 2と後継

- MN-Core2 2023年完成。販売開始した。
- TSMC 7nm。比較的小さいチップで 400TF と MN-Core 並みの性能を実現
- 後継の開発もスタート
  - Samsung 2nm。完成時点で世界最高レベルの性能を目標
  - LLM 推論向けの開発も開始
- ポスト富岳向け検討もしてきたが、直近のものには採用されていない。

MN-Core 2 のソフトウェア

- MNSDK: 機械学習向け
  - PyTorch 記述から ONNX 経由でコード生成
  - 既存の PyTorch コードから小さい修正で動作。ネットワークの記述等は同じ
- HPCSDK: 汎用 HPC 向け
  - OpneCL 方言 (メモリモデルが違う)、OpenACC 方言を用意

MN-Core 2 のアプリケーション性能

	MN-Core 2	A100
GCN(PFN internal use)	5.41TF(FP32)	3.17TF
ResNet50 training	77TF(FP16)	33.2TF(BF16)
ResNet50 Inference	154TF(FP16)	33.7TF(BF16)
HIMENO benchmark	9.03TF(FP32)	0.634TF
OpenFDTD	0.655TF(FP32)	0.488TF

- AI ワークロードでは同程度のピーク性能の A100 の 1.5-5 倍の性能。並列化効率が高いことを実証。
- 差分法系のアプリケーションやベンチマークでも高い性能。但し、OpenFDTD は temporal blocking を実装して DRAM アクセスを減らした成果。姫野ベンチはオンチップメモリにはいるサイズ。

MN-Core の特徴

- チップ内の演算コアがそれぞれメモリをもち、コア間がツリーネットワークでつながる「チップ内分散メモリアーキテクチャ」
- キャッシュはない。外部 DRAM は共有される形で接続されるが、各コアがアドレスをだすのではなくデータ移動命令でツリーノードとの間でブロック転送
- チップ内分散メモリアーキテクチャと従来のアーキテクチャの中間
- 従来型プロセッサより高い性能と実アプリケーションでの高い実行効率を実現
  - SIMD 動作と、総和、放送をサポートする低レイテンシチップ内ネットワークによるオーバーヘッドの小さい並列化
  - 大きな PE ローカルメモリによる効率のよい DRAM アクセスの低減
  - 高い PE ローカルメモバンド幅 (ベクトル演算に対して B/F=8) によるカーネル部分の高効率実行

接合技術の現状

基礎となる技術:

- **接合: マイクロバンプ or ハイブリッドボンディング**
- **TSV: TSV-middle 等のファインピッチ技術**

マイクロバンプ

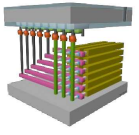
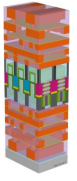
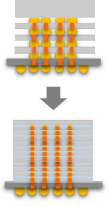
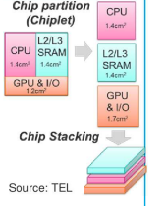
- 基本的にハンダボールや C4 バンプの延長の技術
- 現状で利用可能なピッチ: $\sim 40\mu\mathrm{m}$、Intel MAX GPU では $37\mu\mathrm{m}$。
- 将来的には $10\mu\mathrm{m}$ を切るといわれている。
- HBM3e までの各社の HBM で採用
- 現状では $1\mathrm{mm}^2$ あたり 600 本程度なので、 $800\mathrm{mm}^2$ のチップだと全面において 50 万本。 50TB/s 程度までなら実現可能
- 複数積層だと TSV があるエリアを集中させたいのでもう 1 桁程度高密度にしたい
- ダイ間を樹脂で充填するので、熱抵抗が大きい。(現在の HBM で大きな問題)
- CoW/CoC なので、イールドの問題は少ないが加工費用が高価

ハイブリッドボンディング

- ダイ表面の Cu パッドを直接接合。
- 最初の実用化はソニーの センサー
- 現状で多くのベンダで利用可能なピッチ: $\sim 5\mu\mathrm{m}$
- 将来的には $1\mu\mathrm{m}$ を切るといわれている。
- 現状では $1\mathrm{mm}^2$ あたり 4 万本程度なので、 $800\mathrm{mm}^2$ のチップだとエリアの 2.5% でも 100 万本。 1Gbps で 1Pbit/s、つまり概ね 100TB/s が実現可能。
- ダイ間に樹脂がはいらないので、熱抵抗が小さい。
- Wafer-on-Wafer なので加工費用は安価 (BSPDN でも同様な技術が使われる) が、イールドの問題はある。1Gbit DRAM のイールドは 90-95% といわれており、それを前提にした冗長設計が必須になる。

技術の現状

貼り合わせ接合適用例

Application	CIS	NAND	DRAM	Logic		Advanced Package	
	BSI	3D NAND	3D DRAM	Backside PDN	Sequential CFET	HBM	Disaggregation / Chiplets
Stacking Device	Pixel + (Peripheral) + Logic	Cell + Peripheral	Cell + Peripheral	Logic + Bare Si	Logic + Logic	DRAM ( ) ⋮ DRAM ( ) + Logic	A B C
Bonding	W-W (Ox Fusion) Cu Hybrid	W-W Cu Hybrid	W-W Cu Hybrid	W-W Ox Fusion	W-W Ox Fusion	D-W u Bump → Cu Hybrid	D-W / D-D Cu Hybrid
3D I/O Pitch	3um →1um	1um →0.5um	1um →0.5um	Sub um (nTSV)	Sub um (nTSV)	40um →25um	10um →1um
Structure			 Source: TEL 3D DRAM structure				
Status	HVM	R&D~HVM	R&D	R&D	R&D	R&D	R&D~HVM

次世代デバイスへ向けて、貼り合わせ接合技術の導入が拡大

東京エレクトロンの IR 資料 2025/2 サブミクロンピッチの WoW、TSV は実用範囲