メニーコア型スーパーコンピュータ向け高効率Particle-in-Cell計算手法の開発（の現状と課題）

三宅 洋平*[1]、寸村 良樹[1]、木倉佳祐[1]、中島 浩[2]

1. 神戸大学，2. 京都大学

*y-miyake@eagle.kobe-u.ac.jp

H29年度STEシミュレーション研究会

高性能計算 (HPC) 技術の重要性

- ポスト「京」もしくはエクサスケール計算システムの性能を最大限に引き出す並列計算アルゴリズムが必要

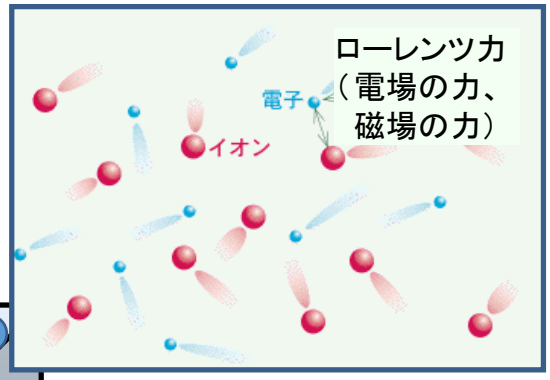
- 次世代超並列計算機システムの並列度 ©2016 Nakashima

	#node	#core	#superscalar	Total parallelism
30 PF system	10,000	100-200	32	$32\text{-}64 \times 10^6$
1 EF system	100,000	150-300	64	$1\text{-}2 \times 10^9$

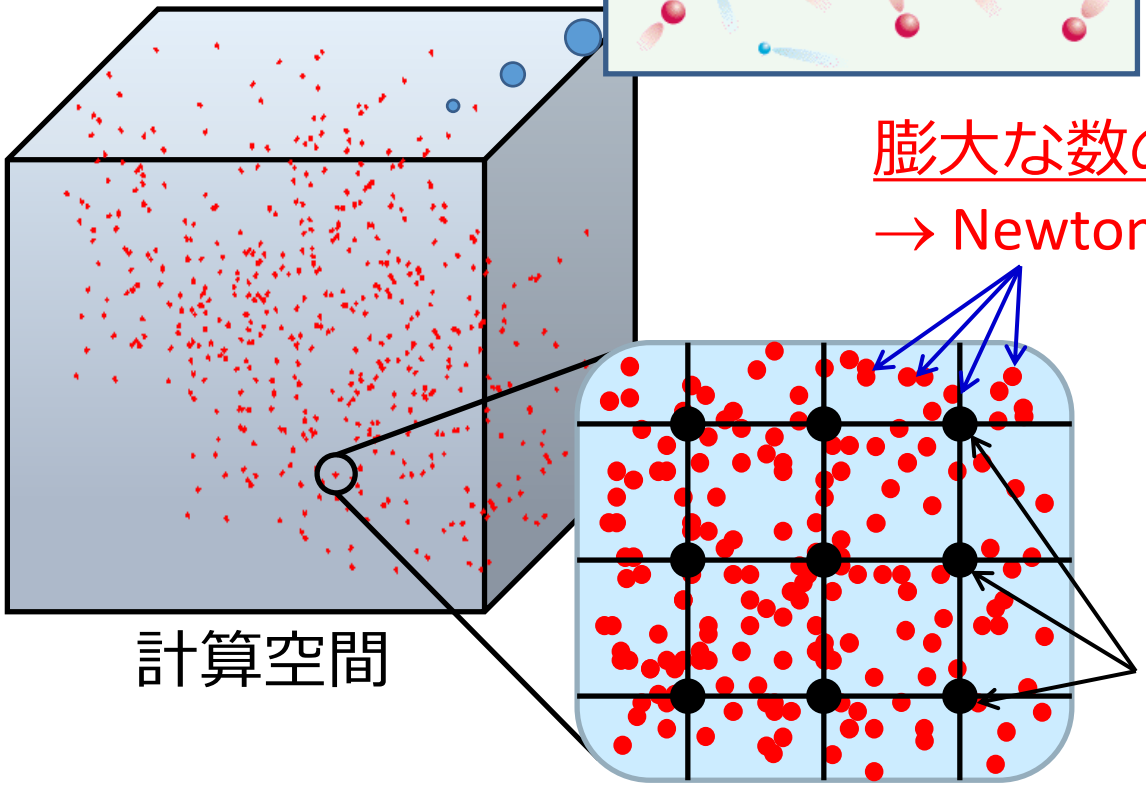
- アーキテクチャ階層毎の並列化・最適化
  - 分散メモリ(プロセス)：負荷分散・通信最適化
  - 共有メモリ(スレッド)：競合回避アルゴリズム
  - SIMD：制御フロー・データ参照の規則化

⇒ 高性能計算技法の探求

プラズマ粒子シミュレーション（Particle-In-Cell, PIC法）



膨大な数のマクロ荷電粒子
→ Newton運動方程式

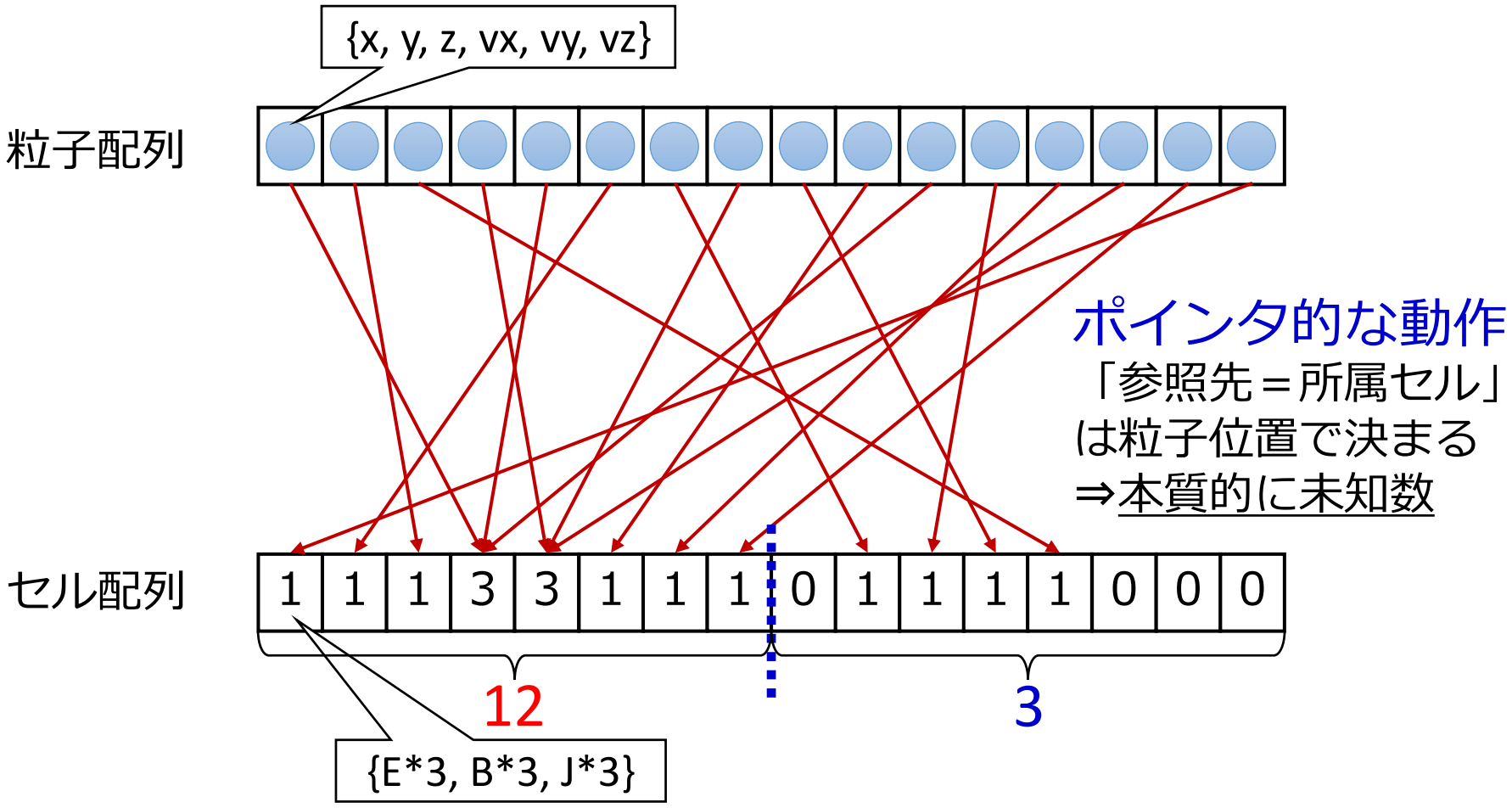


計算空間

格子点上の電磁場
→ Maxwell方程式や
流体方程式

マクロ荷電粒子と格子点上電磁場の相互作用演算

PIC計算のデータ並列処理上の難しさ

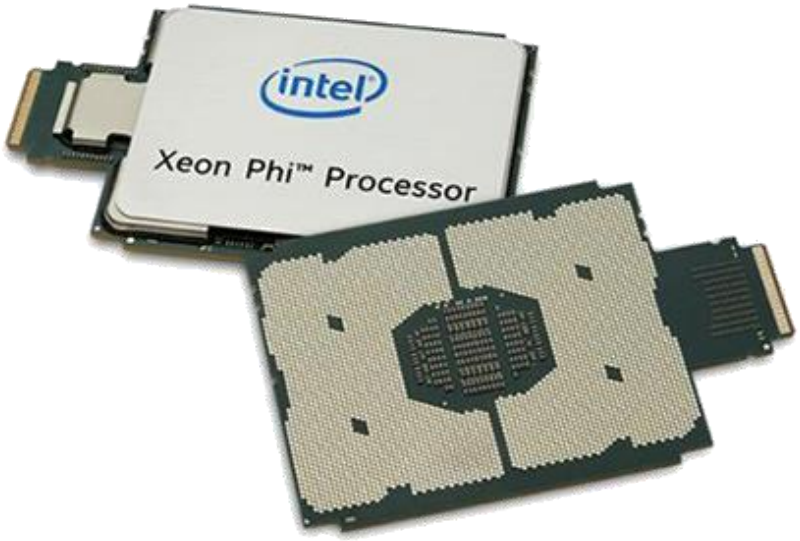


問題点１．粒子を前から順に処理 ⇒ セル配列への参照パターンはばらばら

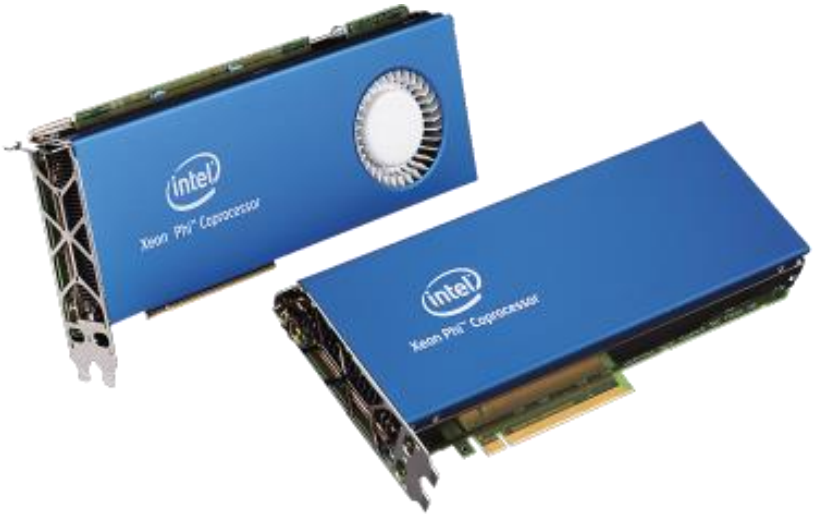
問題点２．分散メモリ並列時に以下の問題：

例えば領域で均等分割 ⇒ 分割された小領域内にいる粒子数がばらばら

メニーコアプロセッサ：Xeon Phiシリーズ



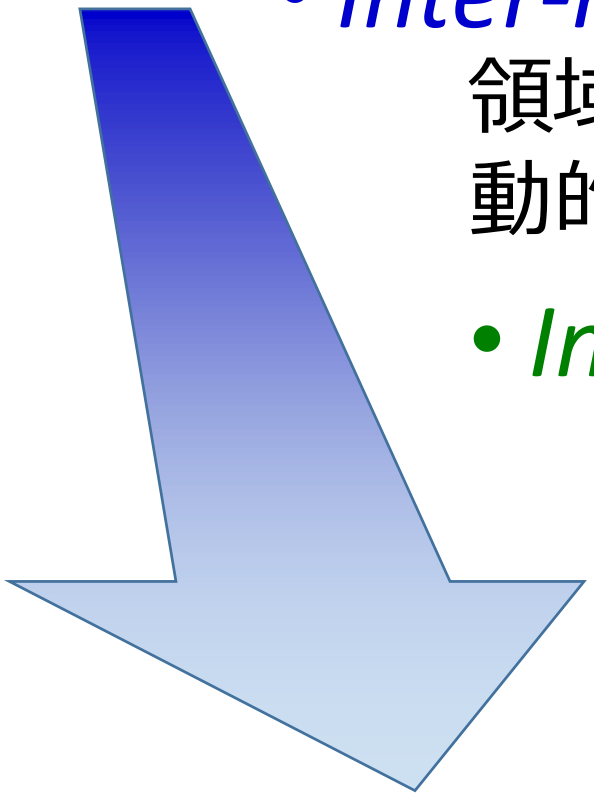
Host processor



PCIe Co-processor

- MIC (Many Integrated Core)に基づくコプロセッサ
- x86互換のアーキテクチャ
- 60～70のコア数 × 4-way Hyper-Threading
- 512bit幅のSIMDベクトルユニット
- (第2世代Knight Landing) ホストプロセッサもしくはコプロセッサとして動作

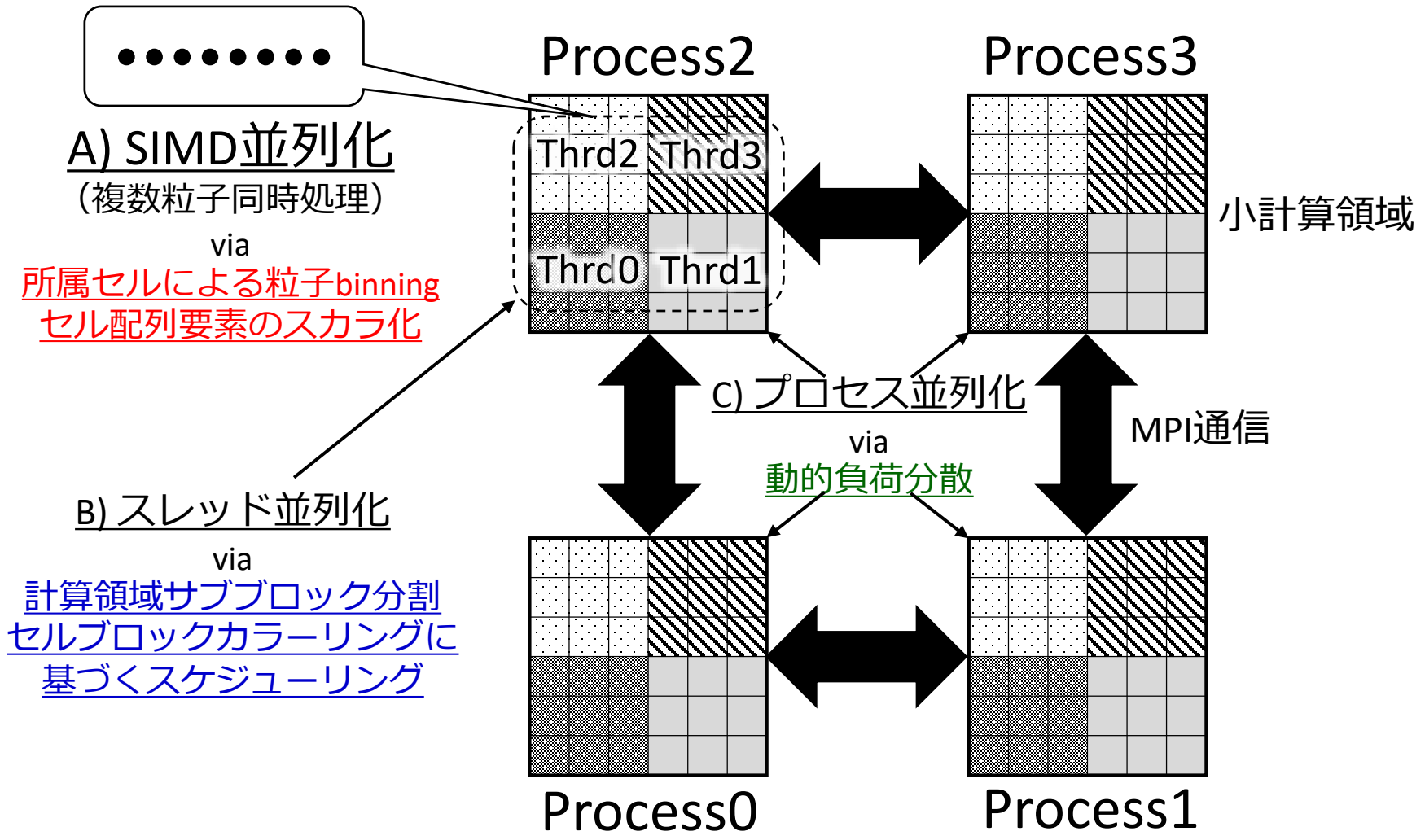
階層的超並列PIC計算の実現に向けたロードマップ



- *Inter-node parallelism*
  領域分割方式の分散メモリ並列
  動的負荷分散
- *Intra-node parallelism*
  10^2スケール共有メモリ並列
- *Intra-core parallelism*
  SIMD演算機構の有効活用

本格的「階層的並列時代」に備える意味でメニーコアプロセッサは良いインスタンス（練習台）

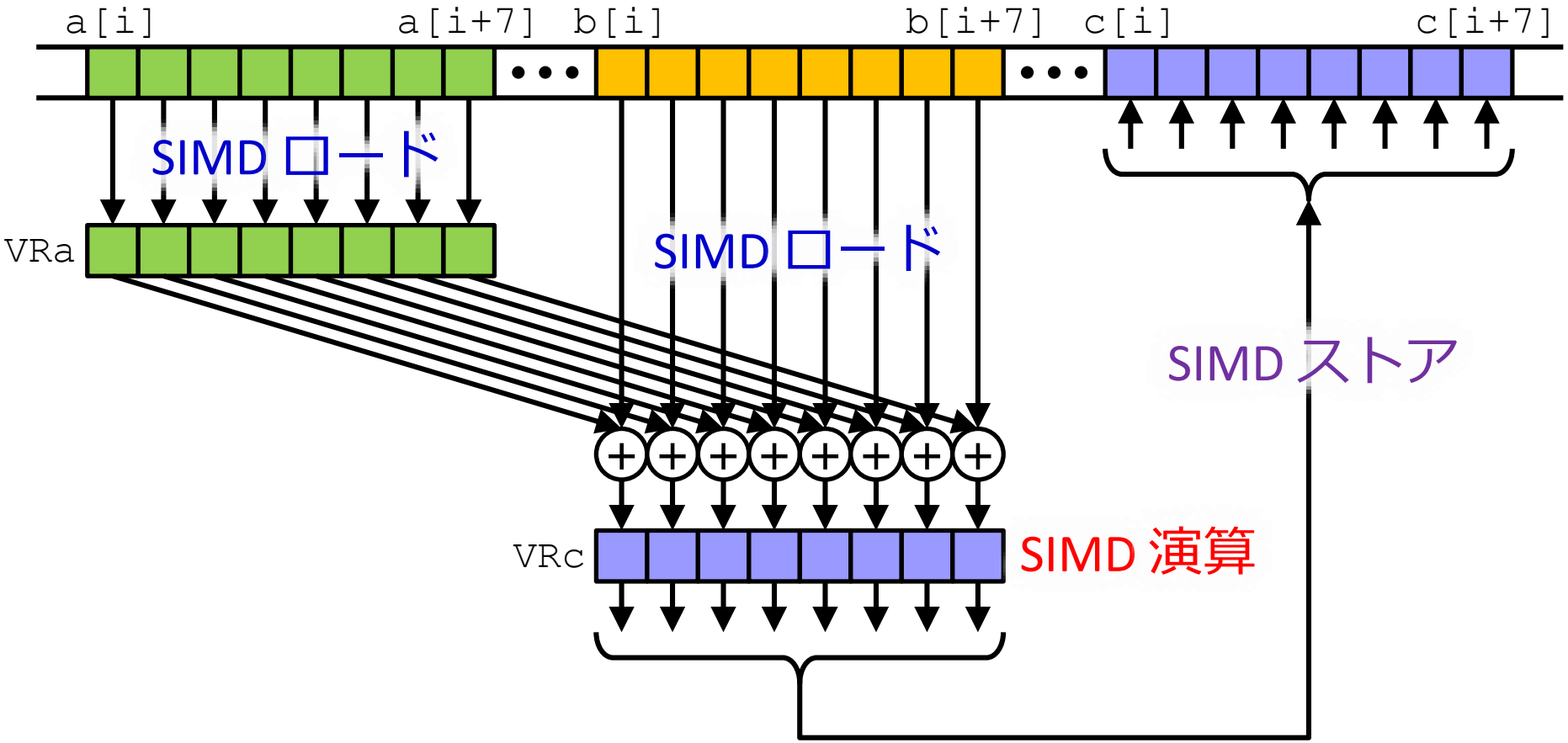
PICプラズマ計算の階層的並列化／最適化



PICプラズマ計算のSIMD最適化

参照・制御の規則性：SIMD効用の“源泉”

- **double a[],b[],c[];**
  for(i=0;i<n;i++) c[i]=a[i]+b[i];



3種類のSIMD処理があることに注意

PIC電流計算の場合：参照規則性の欠如

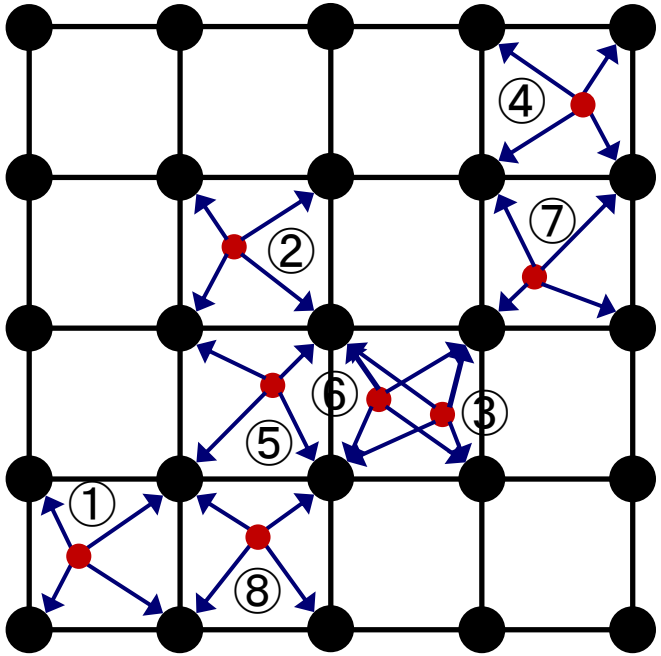
$$J(\delta(x_p)) \mathrel{+}= qv_p$$

各粒子が
作る電流

電流配列

粒子所属セルの
頂点を求める関数

粒子位置
（未知数）



➔ <u>電流密度配列への間接参照</u>

SIMD との相性

1. 粒子がランダムに配列 ➔ J を random access ➔ SIMD化困難／不能
2. 粒子をセル座標でソート ➔ 「それが compiler に伝わらないと」ランダムに見える ➔ SIMD化困難／不能

ではどうするか？ ➔ ①粒子binning

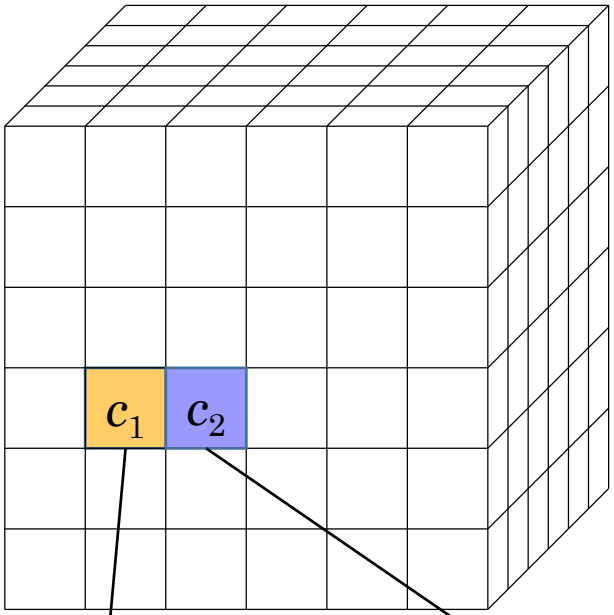
セル（電磁場、電流）の3D-AOS
(Array of Structure)

[Nakashima, CEE, 2015;
Nakashima et al., IPDPS, 2017]

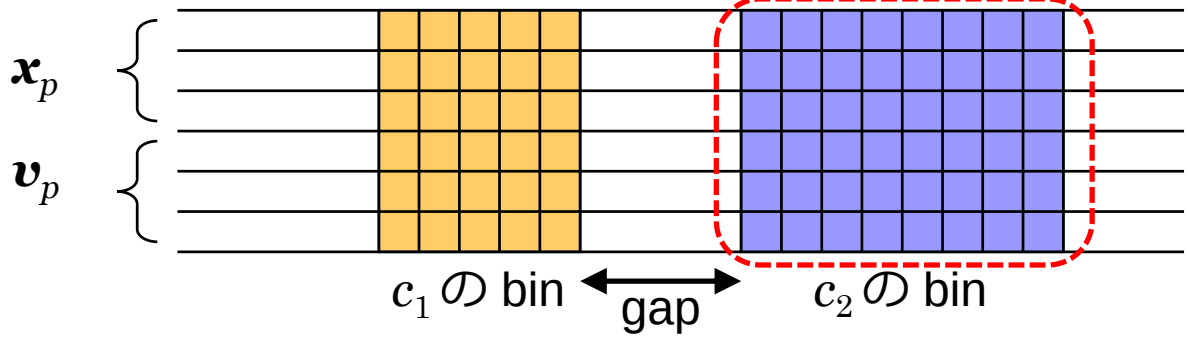
効率的なSIMD演算を実現

参照パターンは規則的

∀p: δ(x_p) ➔ c_2の頂点



粒子の1D-SOA
(Structure of Array)



これで十分？ ➔ 実際には”NO”

ではどうするか？ ➔ ②セル要素スカラ化

電流計算

$J(\delta(x_p))$ += qv_p, p = i+1,...,i+8

は、たとえ実行時に

$\delta(x_{i+1}) = \delta(x_{i+2}) = \ldots = \delta(x_{i+8})$

…であってもコンパイラは（コンパイル時点では）SIMD不可

そこでセル要素「$J(\delta(x_p))$」をスカラ変数化

$j = J(\delta(x_p))$, p = i+1

j += qv_p, p = i+1,...,i+8

$J(\delta(x_p))$ += j

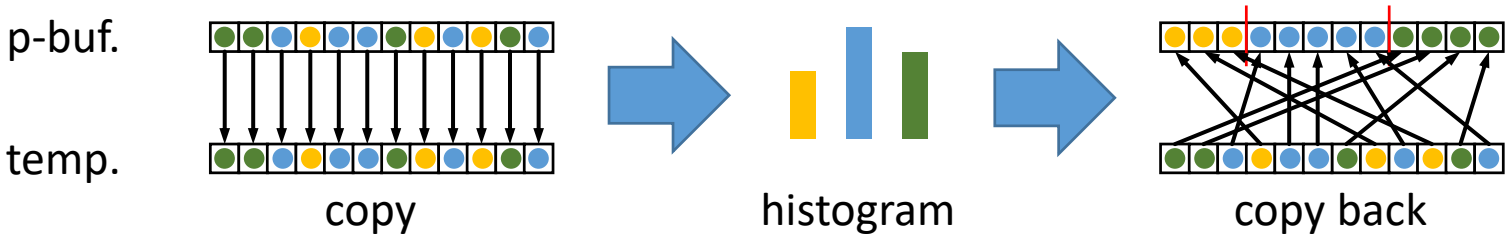
…であればコンパイラは「j += qv_p」をSIMD化できる

＊particle pushのためのE/B-field参照にも利用可能

粒子binning：一見、高コストだが…

粒子ビニング？ 粒子ソーティング？

1. ソーティング
   - （例えば）粒子位置座標に関して、全ての粒子が昇順もしくは降順に隙間なく並ぶように配置
   - 様々なアルゴリズム。例：分布数えソート

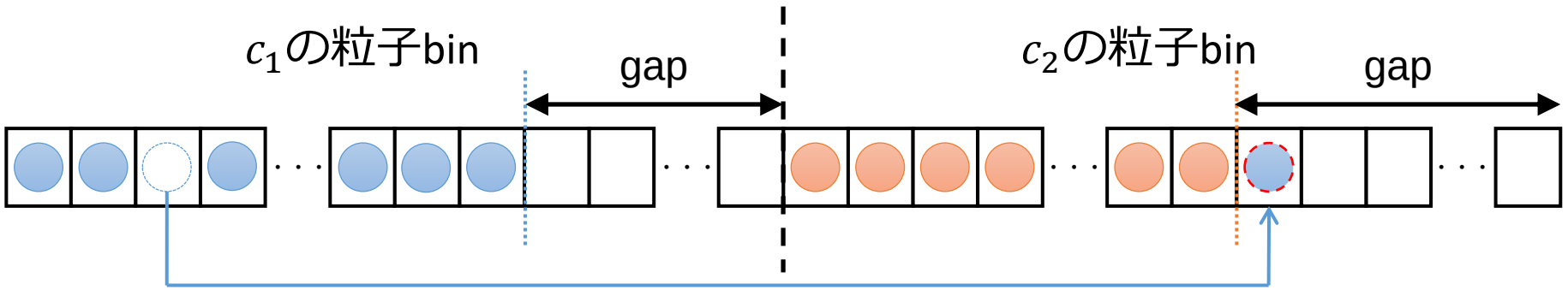


   - 全粒子データが移動対象。長大な一時バッファが必要

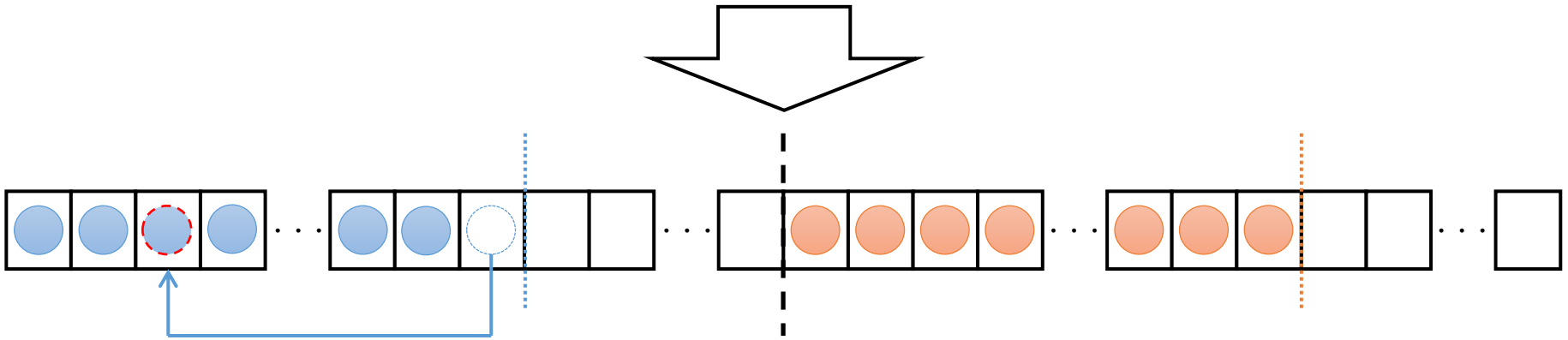
➔ 基本的に高コスト。毎ステップの実施は避けたい。

2. ビニング
   - 所属セルで粒子をグルーピング。グループ間に隙間を許す。
   - 一般に粒子シミュレーションでは、1時間ステップにセル境界をまたぐ粒子は全体の一部 ➔ うまく利用できないか？

on-the-fly方式に基づく粒子binning



①移動先のセルの“gap”の先頭番地にデータ移動

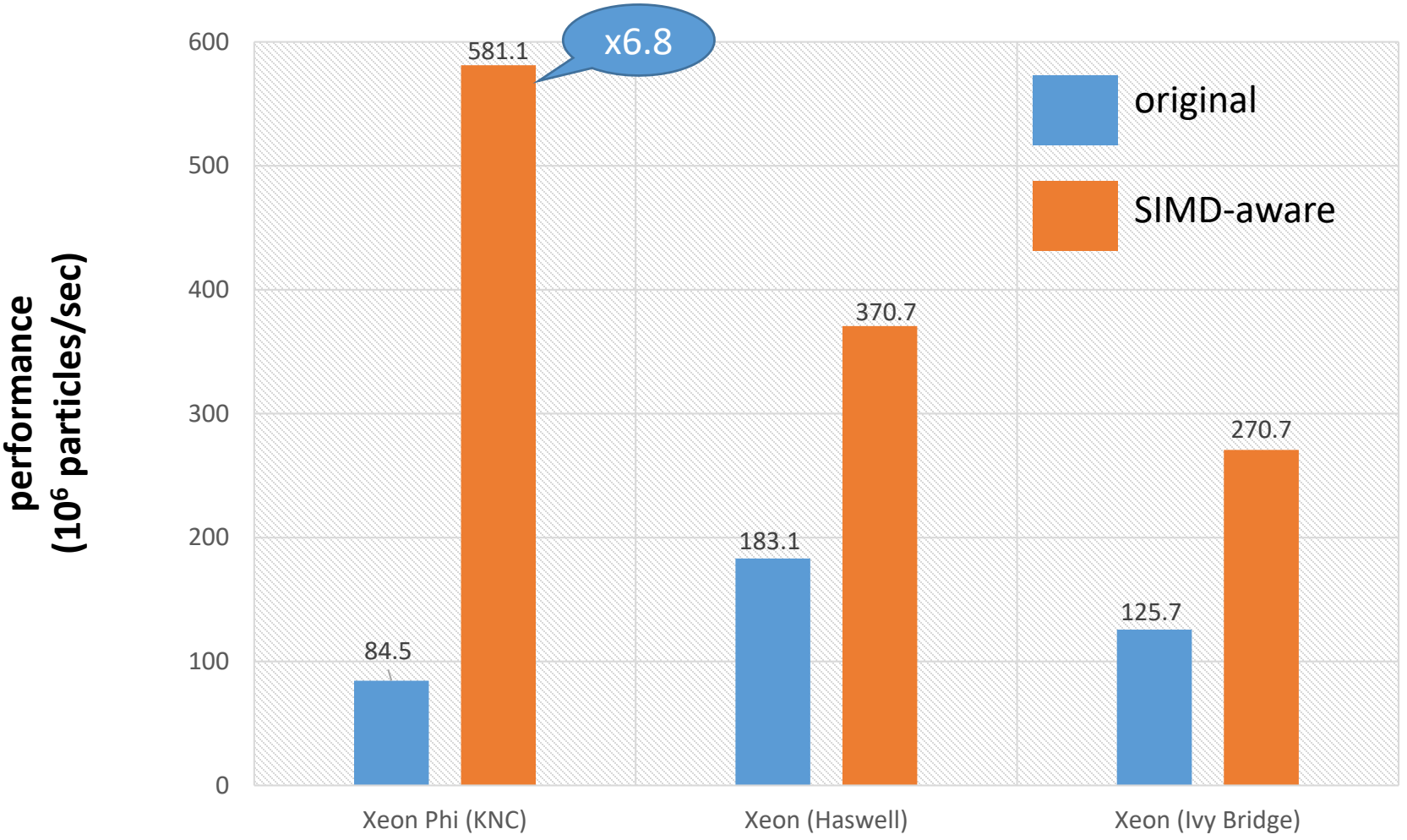


②移動元のセルにできた空白は、末尾の粒子で穴埋め

- セル内の粒子の順序は任意なので、これでオーケー
- 実際にセル境界をまたいだ粒子のみ、データ移動対象となる

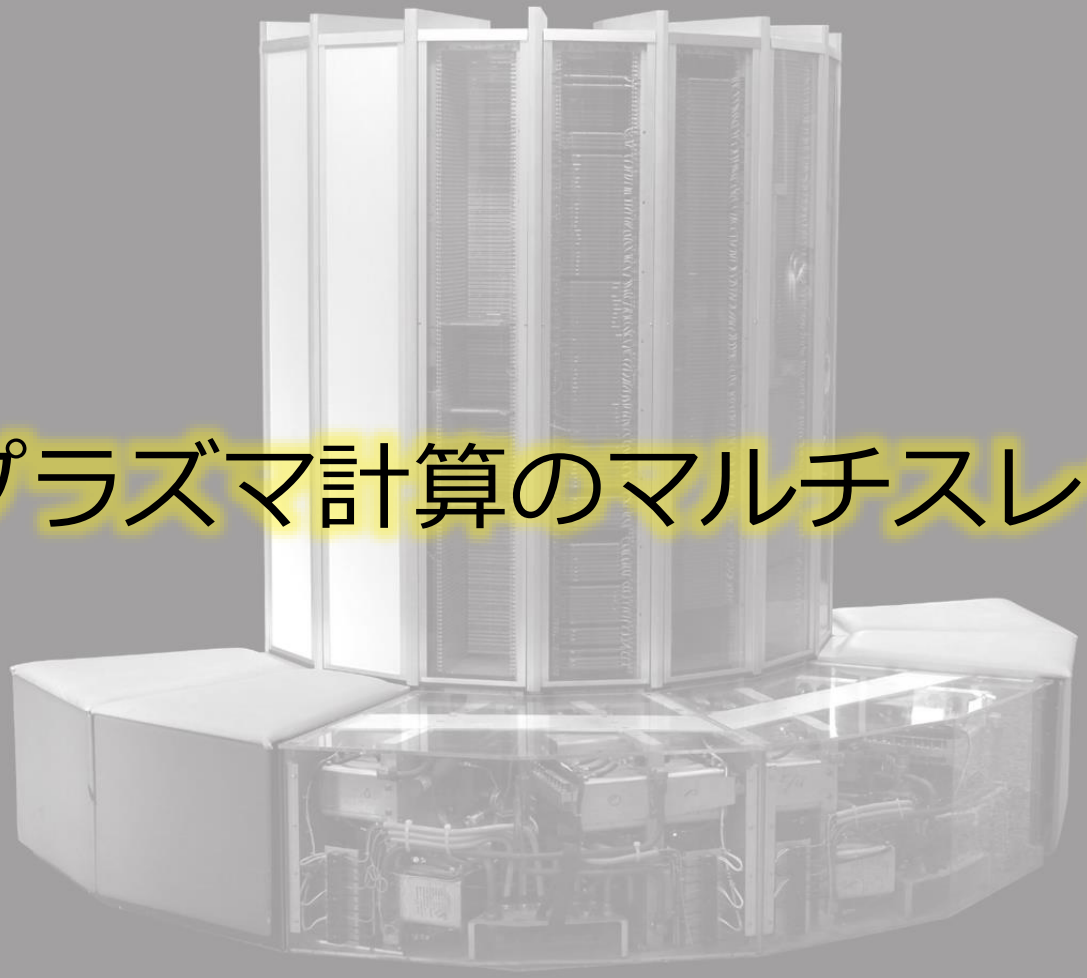
➔ gapが枯渇した(overflow)時の対処は少し面倒

SIMD最適化の効果 (シングルノード計測)



- メニーコアシステムで効果大
- （粒子binの枯渇など）例外的事象の取り扱いが煩雑

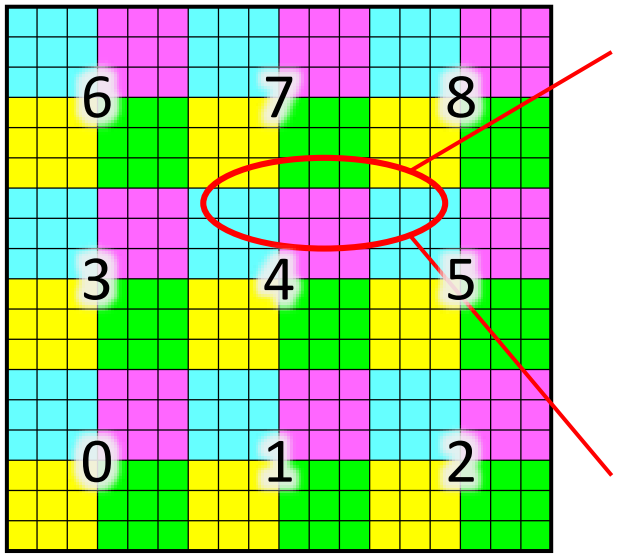
PICプラズマ計算のマルチスレッド化

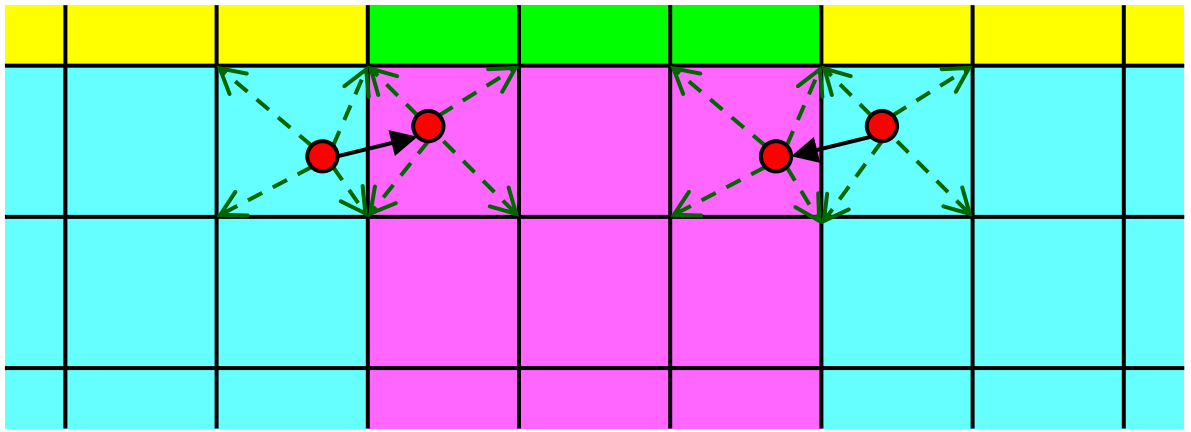


10^2並列度マルチスレッディング

1. プロセス領域内をセルブロックで分割
2. セルブロックを 2^k 色(k 次元分割の場合)で色分け
3. 同色のセルブロックをマルチスレッド実行

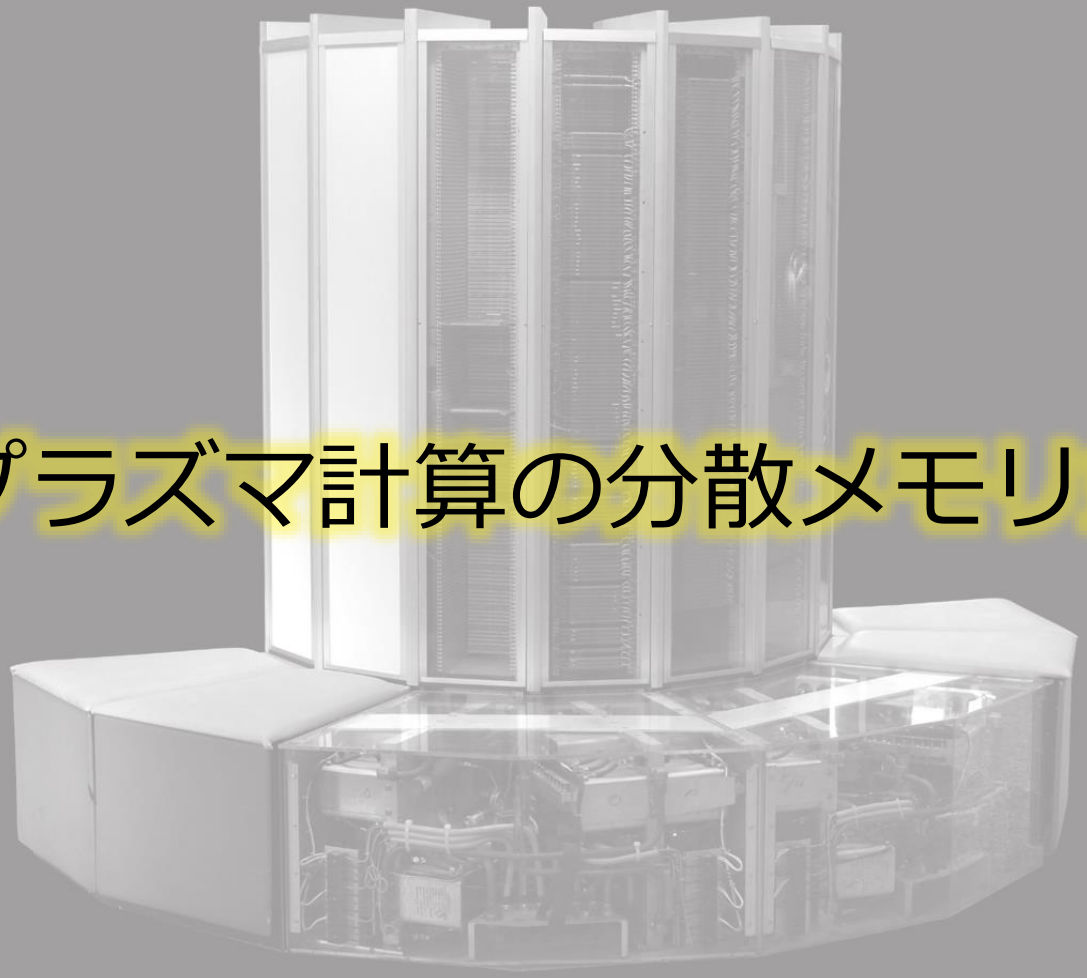
➔ 複数スレッド間の「アクセス衝突」を回避



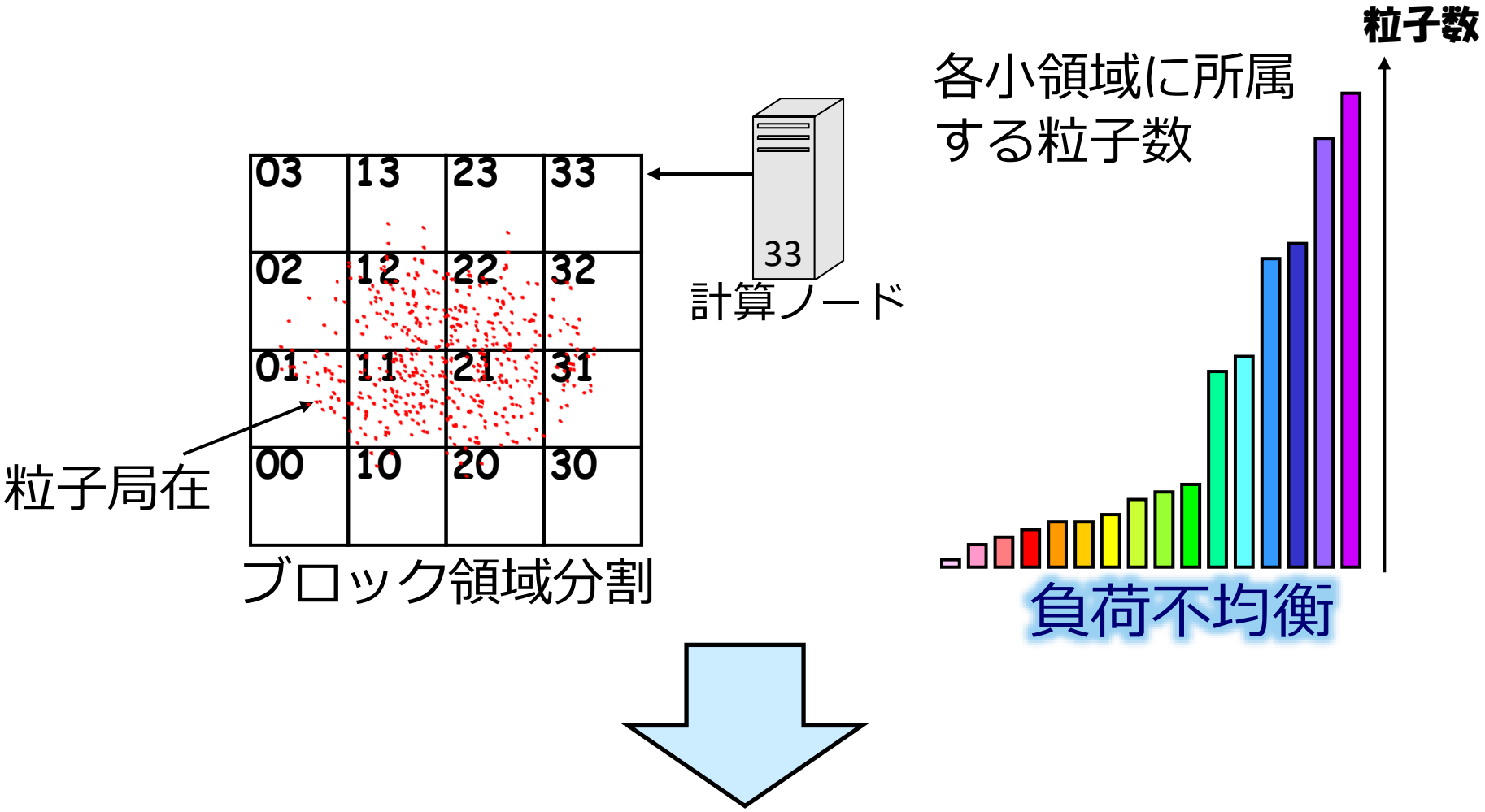


電流・速度計算、粒子binningの並列化に有効

PICプラズマ計算の分散メモリ並列化



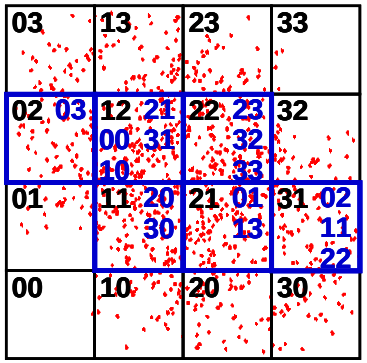
領域分割に基づくプロセス並列

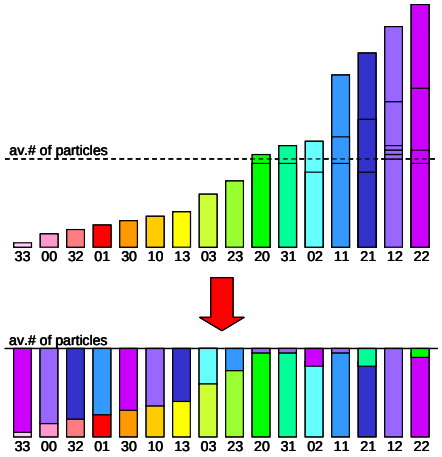


動的負荷分散技法OhHelp

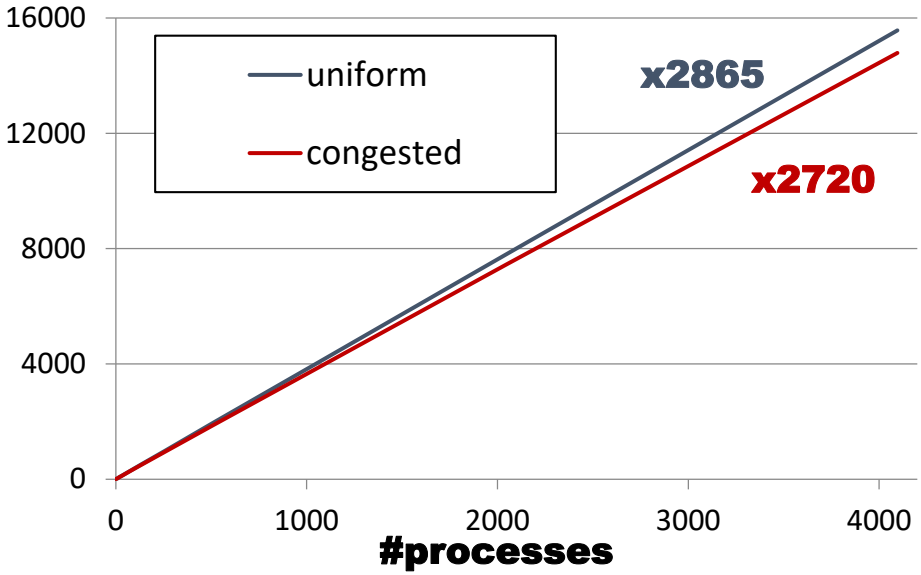
➢概要

- 極端な粒子不均一にも対応した動的負荷分散手法





- 各プロセスは静的に割り当てられた1次領域の他に、粒子数の多い2次領域を担当
- 1プロセス領域に粒子が集中するような状況でも、10^3プロセス以上のスケーラビリティを実現



次世代スパコン向けPICコード開発進捗

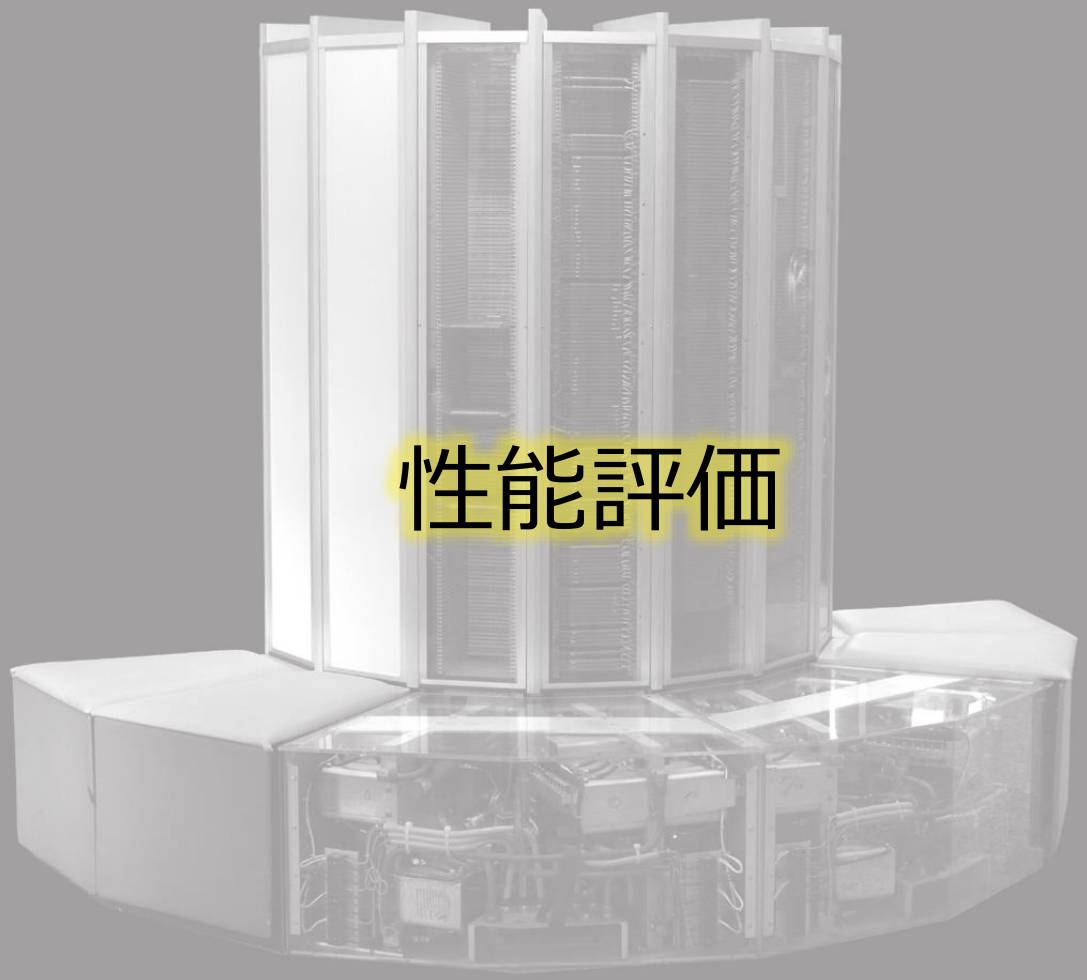
キーコンセプト（プロセス並列：OhHelp、スレッド並列：セルブロック多色順序付け、SIMD：粒子ビニング）を実用コードに採用する上での、「実装上の様々な問題（と解決法）」

例えば…（すべて話すと要3時間）

- 低コスト粒子ビニング実装 ⇒ On-the-fly粒子ビニング
- 粒子ビンオーバーフローへの対処 ⇒ 粒子ビンIn placeシフト
- 粒子ビンオーバーフロー頻度低減のための工夫
  - ⇒ 粒子ビン間gapの再配分法
  - ⇒ オーバーフロー粒子を一時的に保持するバッファ
- スレッド間粒子負荷不均衡の低減

解決法の提案：Nakashima et al., IPDPS, 2017など

コード開発の現状：一部の技法は実用コードに対しては実装中



性能評価

(ミニアプリによる) 性能評価：セットアップ

評価環境

- Cray XC40(KNL), XC30(KNC), XC30(Haswell), XE6(AMD) @京大 ACCMS
- 最大64 node × 最大64コア
- Hyper Threading: HT=1~4 ⇒ HT=2が大体best

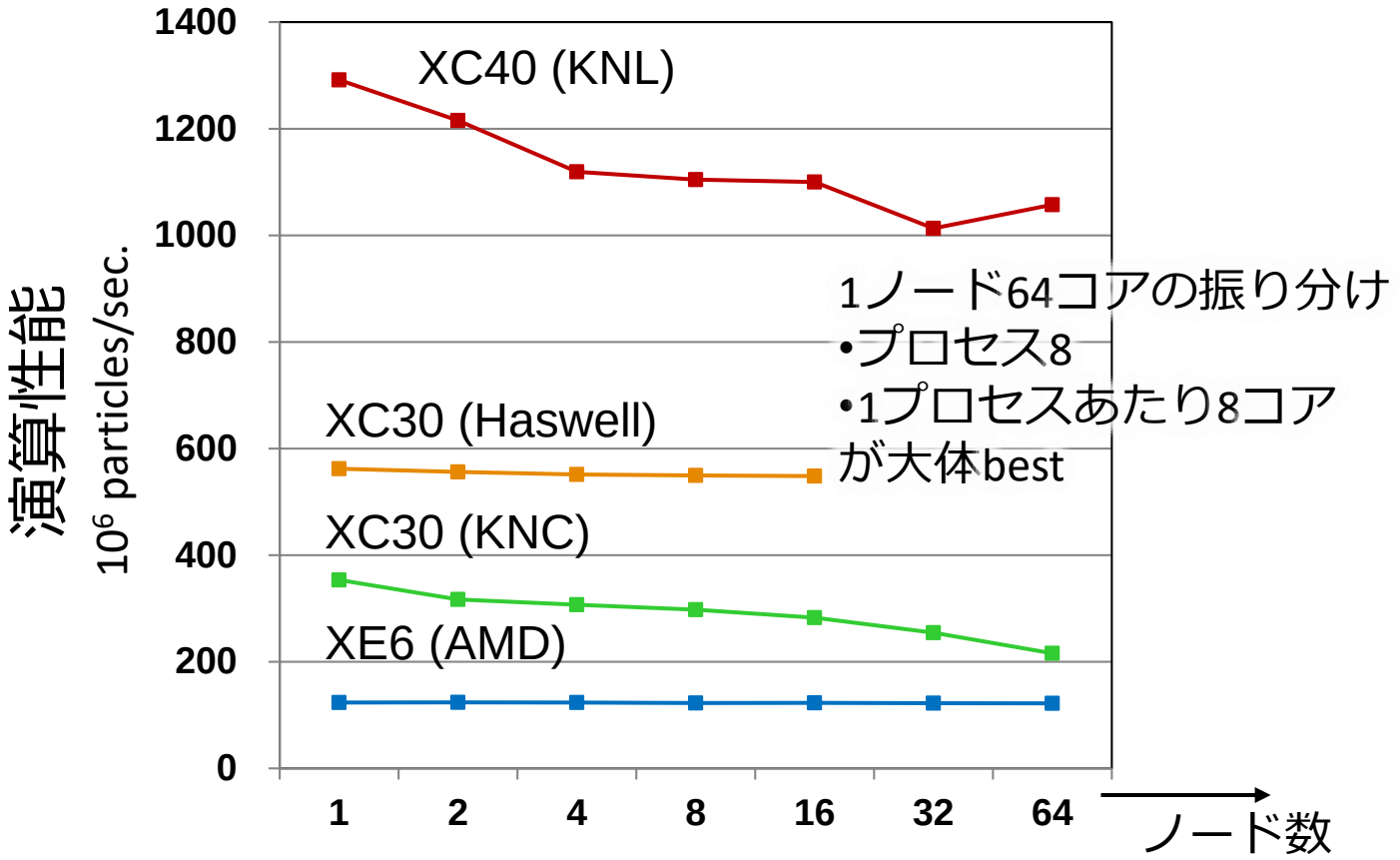
粒子密度・速度・分布

- 平均粒子数 / セル = 128
- 全粒子が x 軸に沿って (1/64)grid / step のドリフト運動
- シミュレーションステップ数 = 2000
- 均一分布: 負荷完全均衡，オーバーフロー僅少
- 不均一分布: 1/4 の空間内で均一分布
  ➔ 大きな負荷不均衡 & 負荷変動

Weak scaling：粒子均一分布

[Nakashima et al., IPDPS, 2017]

ノードあたりの演算性能（⇒フラットなら並列効率100%）



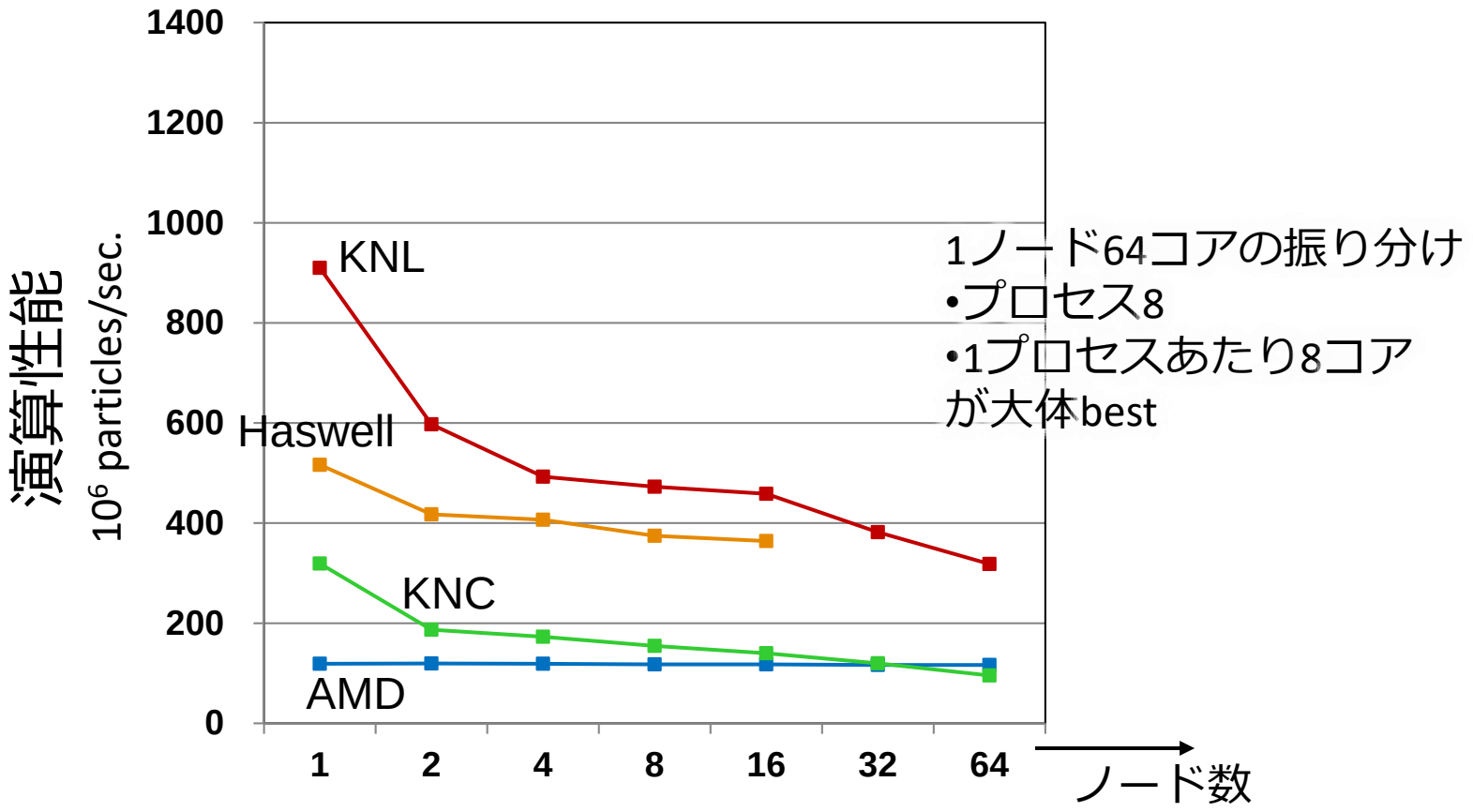
	peak TFlops
KNL	3.05
Haswell	1.03
KNC	1.01
Abu Dhabi	0.32

- マルチコアプロセッサ：非常に良い並列性能
- メニーコアプロセッサ：許容範囲ながら改善の余地あり

Weak scaling：粒子不均一分布

[Nakashima et al., IPDPS, 2017]

ノードあたりの演算性能（⇒フラットなら並列効率100%）

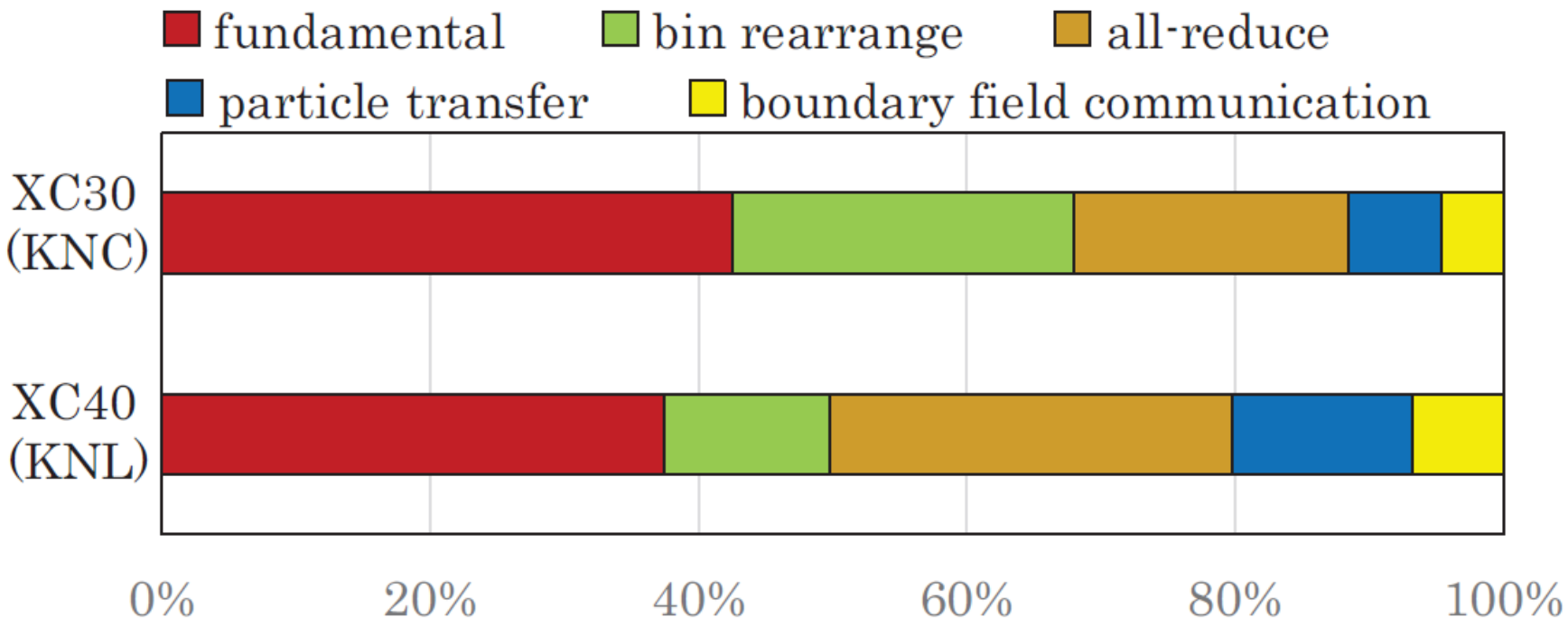


- 特にメニーコアプロセッサで性能劣化が顕著

⇒ ノード間通信性能がボトルネック

実行時間breakdown分析

[Nakashima et al., IPDPS, 2017]



電流のAll-reduce計算（動的負荷分散OhHelpに必要）がネックに

まとめ

メニーコア時代のHPC物理計算

SIMD活用のため、不規則な制御フロー・データ参照を排除

PICプラズマ計算の高効率メニーコア実装法の探求

1. SIMD: 粒子binning、セル配列要素スカラ化で参照規則化
2. マルチスレッド：セルブロック分割＆色付け
3. マルチノード：OhHelp

➔ 性能はまずまずだが，予想以上に P 大 & 不均一性能低

諸悪の根源 = KNC/KNL のMPI性能不足

課題

プロダクションコードの作成、ポアソンソルバーの実装

物理的シミュレーションsetupでの性能評価

演算・通信オーバーラップ（気が進まないが…）