ポストペタスケール時代の
メモリ階層の深化に対応する
ソフトウェア技術
(JST-CRESTポストペタ領域 2012-2018)

研究代表者: 遠藤敏夫 (東工大)
主たる共同研究者: 佐藤幸紀 (JAIST)
主たる共同研究者: 緑川博子 (成蹊大)

研究概要

- ポストペタ時代の気象・医療・防災シミュレーションの大規模化・高性能化実現に向けて、メモリウォール問題の悪化が妨げとなる
- システムソフトウェア・アーキテクチャ・アプリにまたがったco-designにより問題解決を図る

異種メモリ階層アーキテクチャを想定

- HMC, NVRAMなど次世代メモリ技術含む

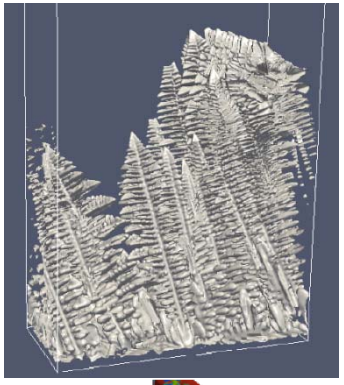
階層活用システムソフトウェアの研究

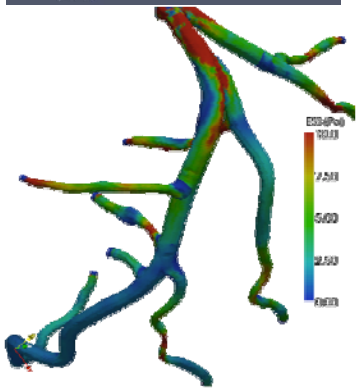
- コンパイラ・ランタイム・メモリ管理の連携によりデータ移動・構造変換・アプリの局所性向上を自動/半自動で実現

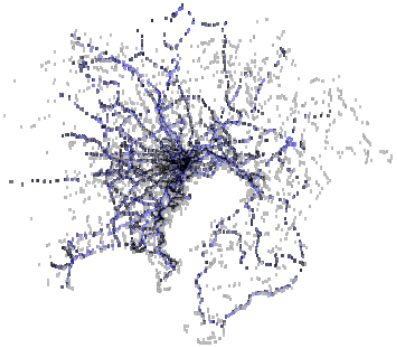
↓

次世代大規模・高性能シミュレーションの実現

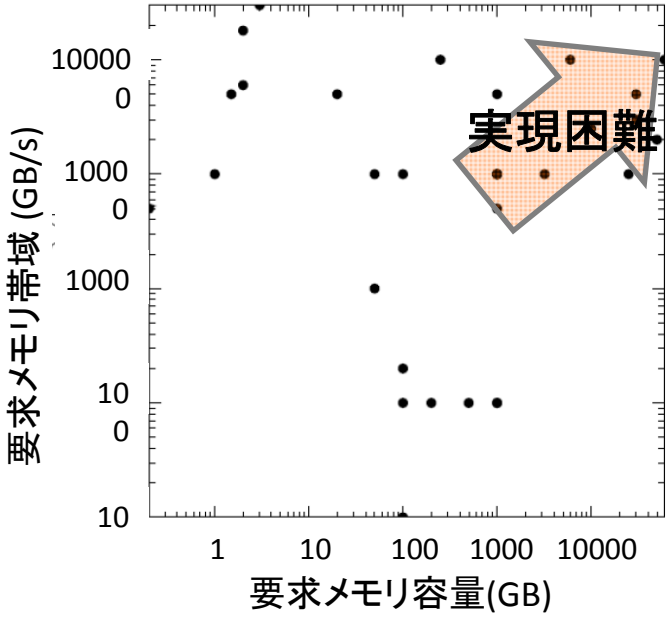
ポストペタ時代に実現が求められる安全・安心社会のためのシミュレーション





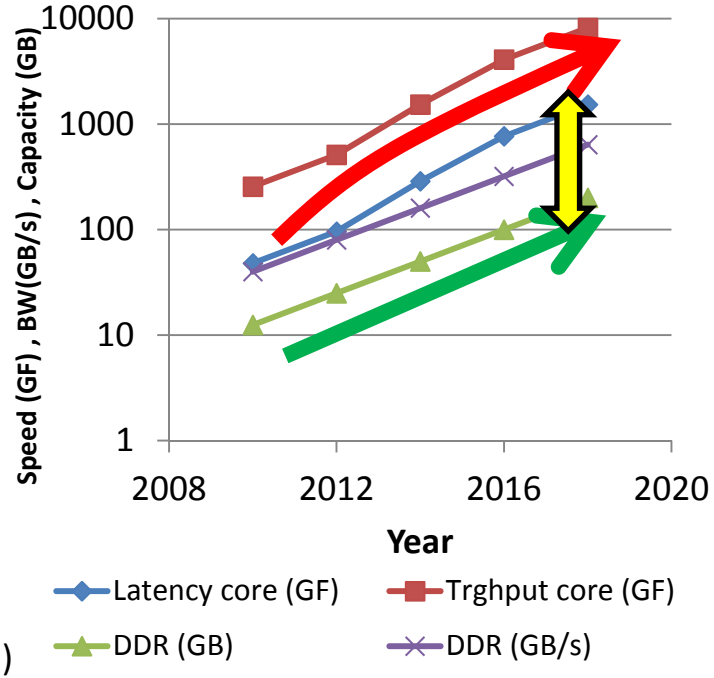


約50種のシミュレーションが要求するメモリ容量・帯域見積もり (100TFlopsあたり)



(計算科学ロードマップ白書の図を一部変更)

プロセッサ性能とメモリ(DDR系)性能予測



各アプリの要求するFlop/s, Byte/s, Byteはバラバラ

エクサスケール時代に向けた メモリウォール問題の悪化

- HPC community is going to Exascale around 2020
- “Memory wall problem” introduces trade-off among “Flop/s”, “Byte/s” and “Byte”

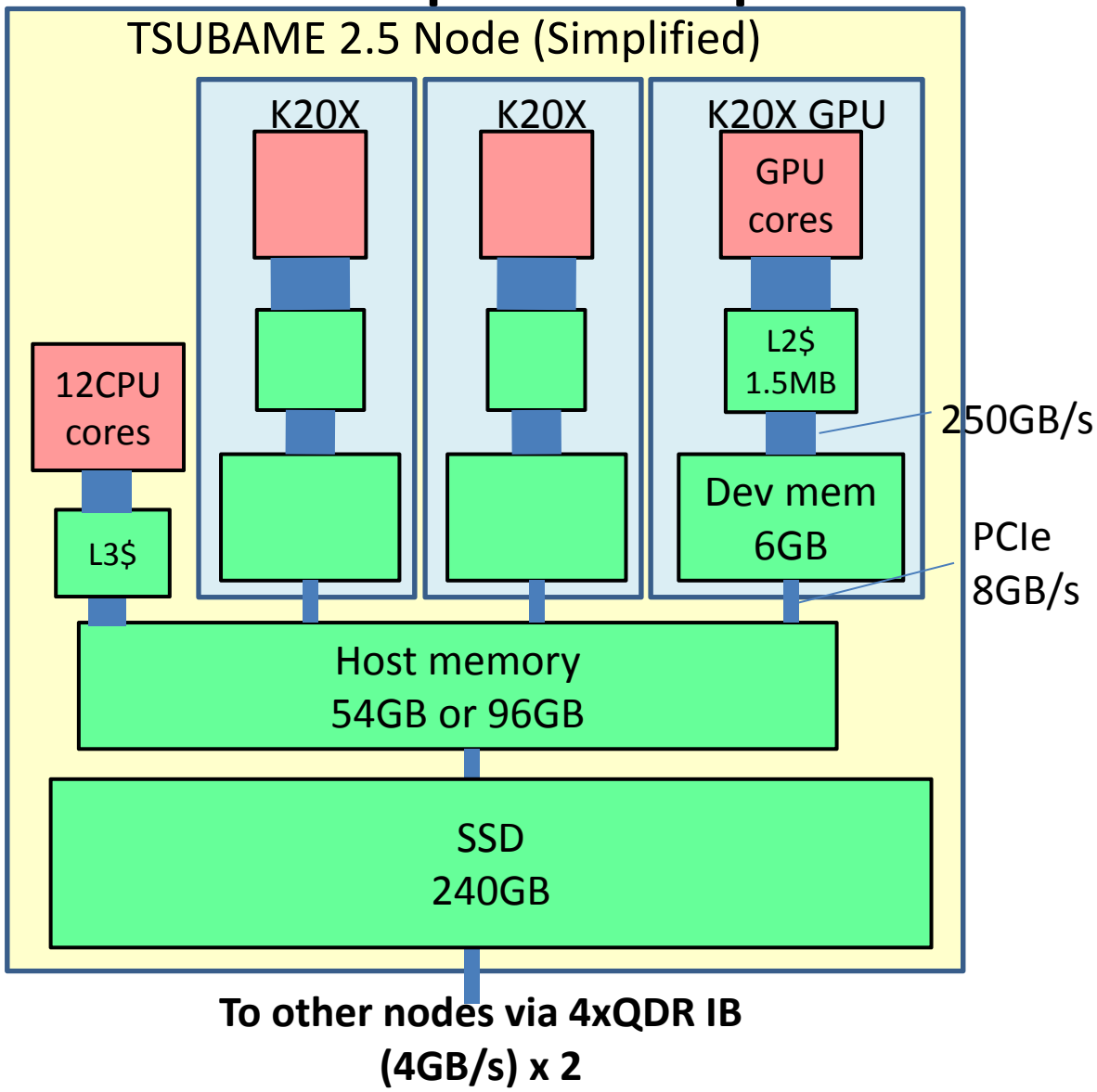
	ASCI white (2000)	BG/P (2007)	FX1 (2008)	M2070 GPU (2011)	K (2011)	K40 GPU (2013)	FX100 (2015)	KNL (2015)
Speed (GF/s)	24	13.6	40	515	128	1430	~1000	~3000
Mem BW (GB/s)	?	13.6	16	150	64	288	~480	~500
B/F =(B/s)/(F/s)	?	1	0.4	0.29	0.5	0.2	0.48	0.16
Mem size (GB)	12	4	32	6	16	12	32	16
B/(F/s)	0.5	0.29	0.8	0.01	0.13	0.008	0.03	0.005
				+ host mem		+ host mem		+ on-card DDR4

演算性能(F/s)もメモリバンド幅(B/s)も重要で、かつ大規模データ(B)を扱いたいアプリを，将来のスパコンでどうやって実現するか？

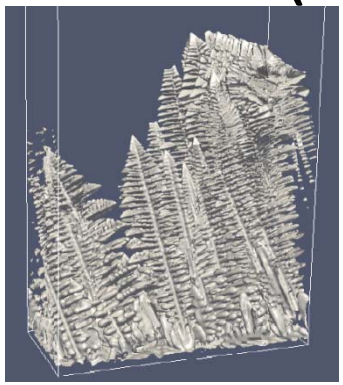
現在のメモリ階層を持つシステム
TSUBAME2.5 GPU Supercomputer

5.7PFlops system has
1408 compute nodes

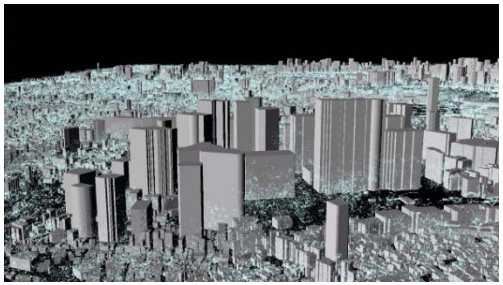
A node has
12 Xeon cores (2.9GHz) &
3 NVIDIA K20X GPUs



メモリ速度・容量ともに必要なアプリケーション例 (TSUBAME2上で稼働中のもの)

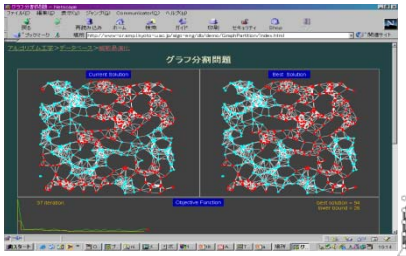


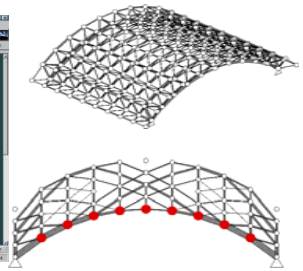
デンドライトシミュレーション
- ステンシル
- 4000GPUで3.4PFlop/s(SP)
- サイズ6GB×4000が限界



都市気流シミュレーション
- ステンシル (LBM)
- 1000GPUで150TFlop/s
- サイズ3GB×1000が限界

GPUメモリ容量が10倍欲しい！

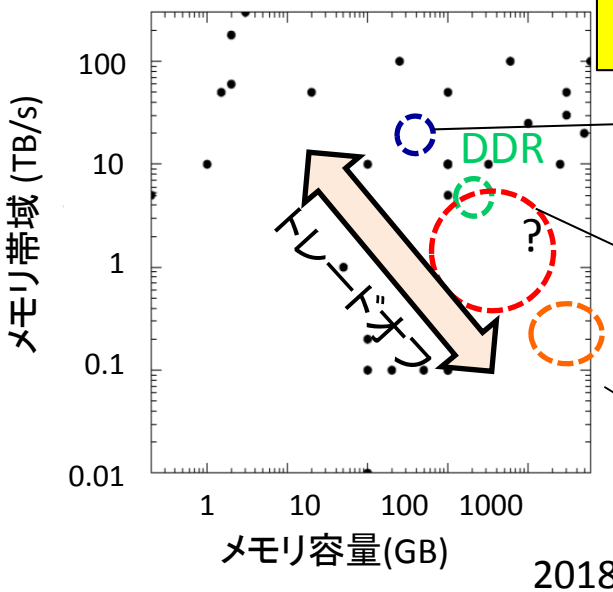




半正定値ソルバーSDPARA
- 行列(密・疎)＋EP
- 4000GPUで1.7PFlops/s(密行列部分、DP)
- ホストメモリ計70TBを利用

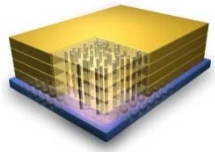
- PCI通信がネック
- ホストメモリ容量も超えたい

将来考えられる想定アーキテクチャ



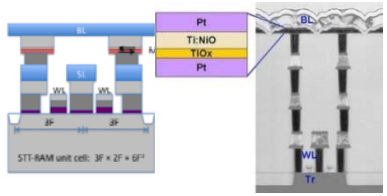
多様化するメモリ階層

Hybrid Memory Cube (HMC)



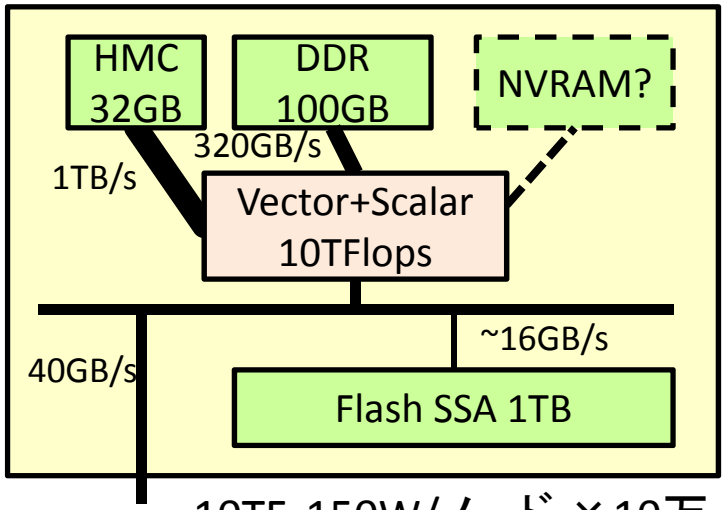
次世代 不揮発メモリ(NVRAM)

- STT-MRAM, ReRAM, FeRAM…



O(GB/s)高速Flashメモリ

2018~2020年ごろの見積もり



10TF, 150W/ノード ×10万

[ノード]

- HMC, DDR, Flash SSAなどの**異種メモリ階層**を持つ
  - メモリ構成自体も研究対象
  - 次世代NVRAMを追加？DDRを置き換え？
- リモートノードのメモリも階層に含まれる

本プロジェクトのアプローチ

Co-design

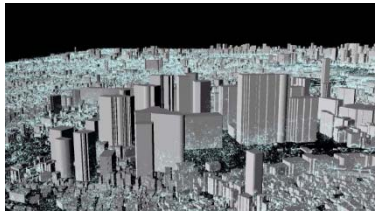
局所性向上・
メモリ最適化技術

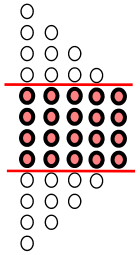
＋

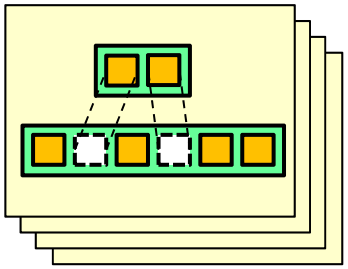
メモリ階層対応
システムソフトウェア

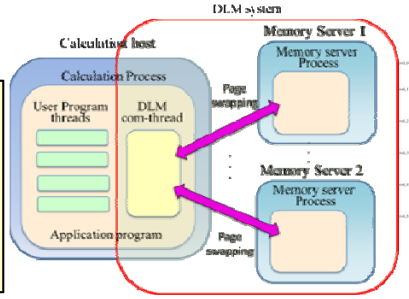
＋

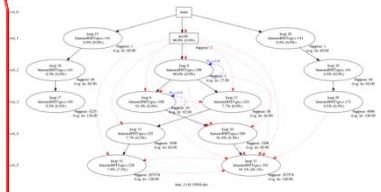
異種メモリ階層利用
アーキテクチャ

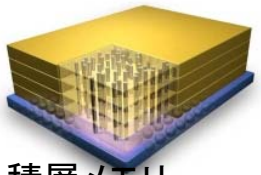






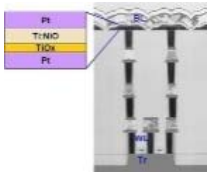


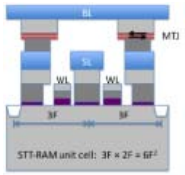




積層メモリ
(HBM/HMC)

PCI-E Flash

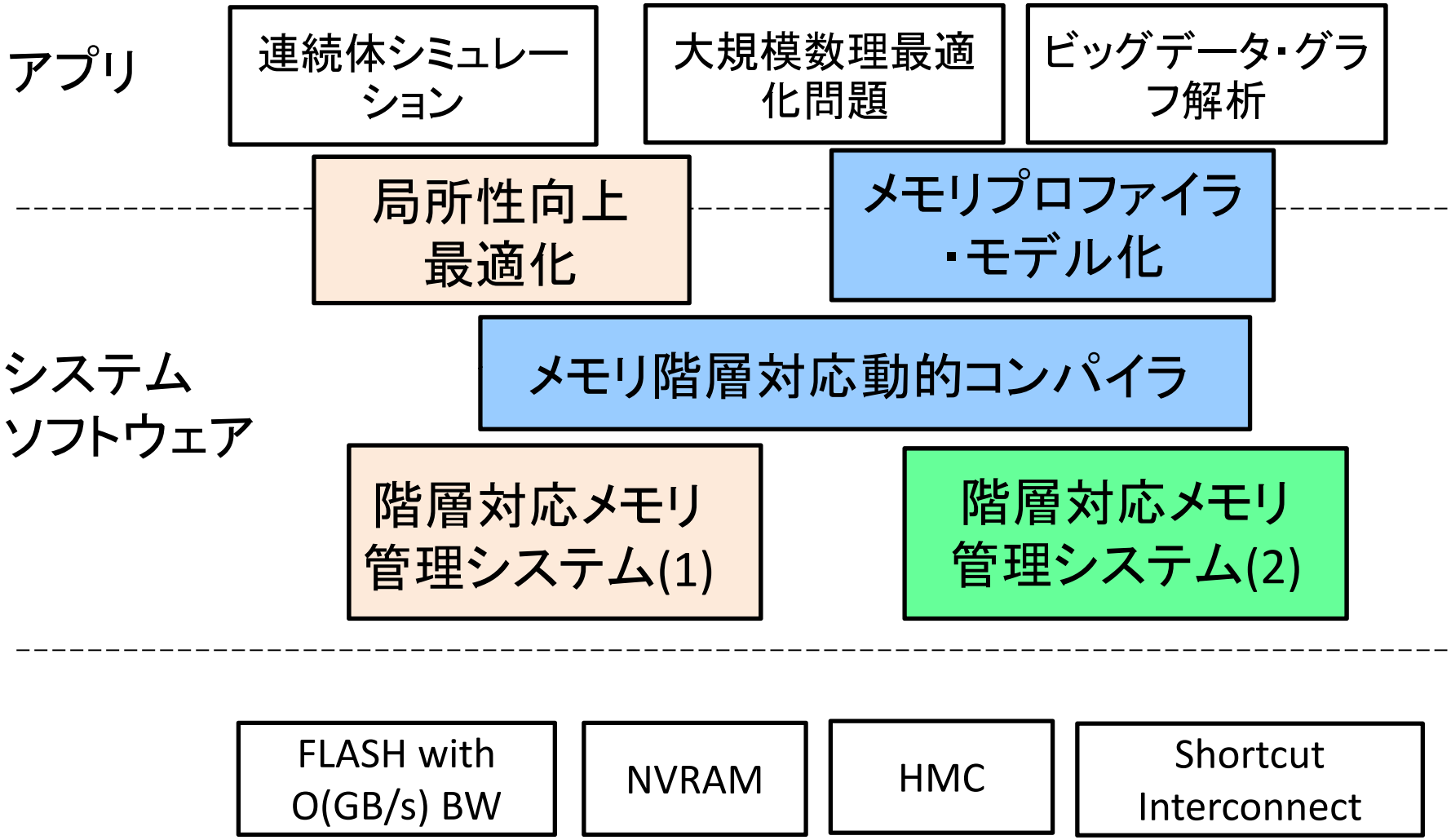




次世代不揮発メモリ
(ReRAM, STT-MRAM, PCM…)

研究のねらい：
トレードオフを持つ異種メモリを有効活用し、大規模・高性能シミュレーションを実現するソフトウェア技術の研究開発
「エクサ時代にO(100PB/s) & O(100PB)」

研究実施体制


アプリ
連続体シミュレーション
大規模数理最適化問題
ビッグデータ・グラフ解析
局所性向上最適化
メモリプロファイラ・モデル化
システムソフトウェア
メモリ階層対応動的コンパイラ
階層対応メモリ管理システム(1)
階層対応メモリ管理システム(2)
FLASH with O(GB/s) BW
NVRAM
HMC
Shortcut Interconnect

研究実施体制

アプリ

連続体シミュレーション

大規模数理最適化問題

ビッグデータ・グラフ解析

研究代表者: 遠藤敏夫 (東工大)

遠藤グループ
- 鯉渕道紘(NII)、佐藤仁、金光浩(東工大)
- 大学院生数名

佐藤幸紀 (JAIST)グループ
- 田中清史、請園智玲、松原裕貴(JAIST)
- 大学院生数名

緑川博子 (成蹊大)グループ
- 甲斐宗徳(成蹊大)
- 丹 英之(アルファシステムズ)
- 大学院生数名

FLASH with O(GB/s) BW

NVRAM

HMC

Shortcut Interconnect

研究実施体制

ポストペタ丸山直也チーム (理研)
- ステンシルアプリ
- Physisフレームワーク

南チーム (九大)
- 京の大規模アプリ

ポストペタ藤澤克樹チーム (九大)
- 数理最適化問題

HPCI FSアプリチーム
- ミニアプリ
- 性能解析ツール

研究代表者: 遠藤敏夫 (東工大)

遠藤グループ
- 鯉渕道紘(NII)、佐藤仁、金光浩(東工大)
- 大学院生数名

佐藤幸紀 (JAIST)グループ
- 田中清史、請園智玲、松原裕貴(JAIST)
- 大学院生数名

緑川博子 (成蹊大)グループ
- 甲斐宗徳(成蹊大)
- 丹 英之(アルファシステムズ)
- 大学院生数名

ビッグデータ松岡チーム (東工大)
- 階層ストレージアーキ

ポストペタ近藤チーム (東大)
- 電力最適化

- Jeff Vetterチーム (ORNL)
- Intel, NVIDIA
- NEC, HP, Fusion-IO・・・
- 東工大TSUBAMEチーム

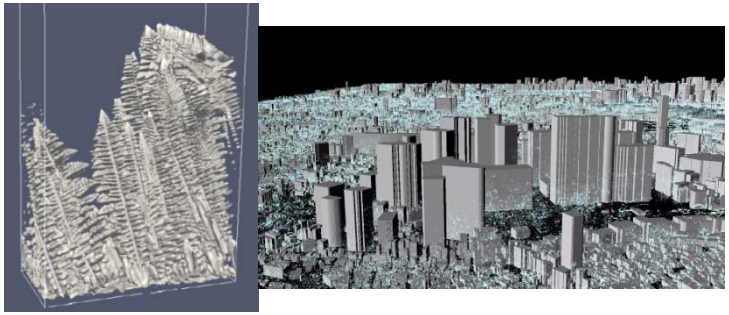
HPCI FSシステムチーム

アプリ局所性向上・メモリ最適化手法 (遠藤G)

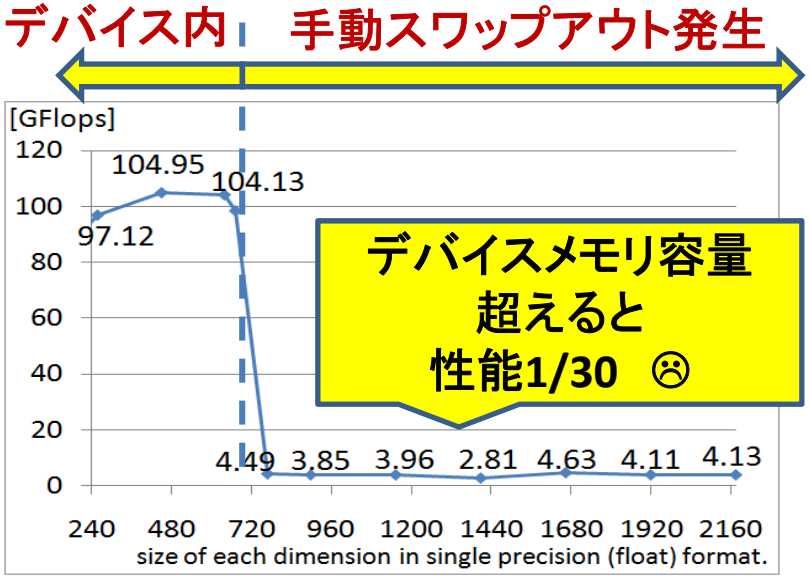
各種アプリ分野の局所性向上手法の提案と、メモリ階層アーキテクチャのモデル化・マッピング

- 当初はステンシル演算に注目
  - メモリ階層の深いアーキテクチャとして、GPUスパコンTSUBAME2を用いる
  - 通常、GPUデバイスメモリ容量(×利用GPU数)にサイズが制限！

  多くのアプリでは、メモリ階層を有効活用できていないと言える



1GPUで3D7点ステンシル



- あえてGPUメモリサイズを超えると、大幅な性能低下！
  - 大規模データをホストメモリに置き、計算したい部分をGPUメモリに「キャッシュ」してから計算

  → ナイーブな実装では、速度1/30に!

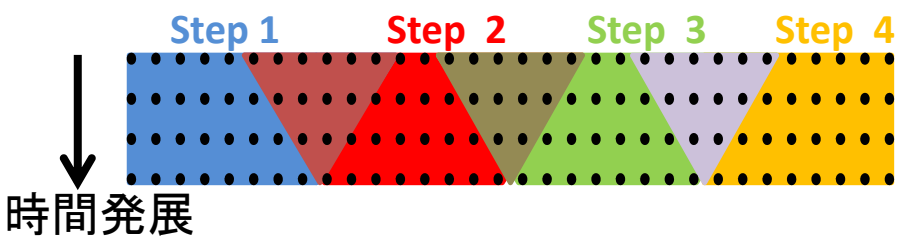
ステンシル計算における局所性向上(1)

- （空間）ブロッキングも有効だが、まだまだ不足

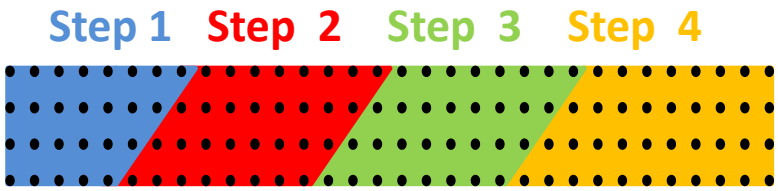
⇒ 時間ブロッキング手法が（古典だが）キーテクノロジーの一つ

[Wolf 91] [Wonnacott 00] [Demmel 12] [岩下13]…

時間ブロッキング (3段)



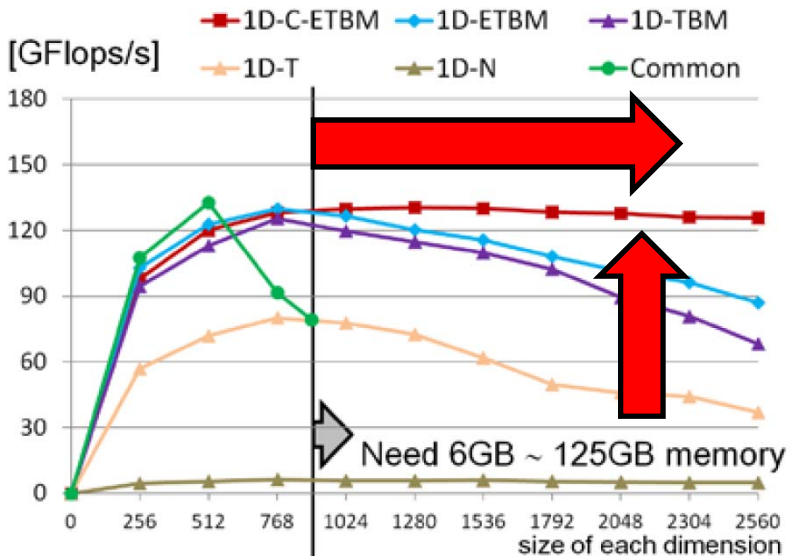
冗長演算削除した場合



- 主にキャッシュ向けに提案された局所性向上手法を、より広いメモリ階層へ適用
  - 大規模データを下位メモリに置き、計算したい部分を上位メモリに「キャッシュ」してから計算
  - 階層間データ移動削減に応用
- さらに，プログラムを激しく手動で最適化することにより、メモリ階層を利用するオーバヘッドを極限まで削減 (Cluster 13, NVIDIA Japan 14, ICHPCA 14など)
  - 部分領域間の境界バッファの配置と階層間移動を最適化

ステンシル計算における局所性向上(2)

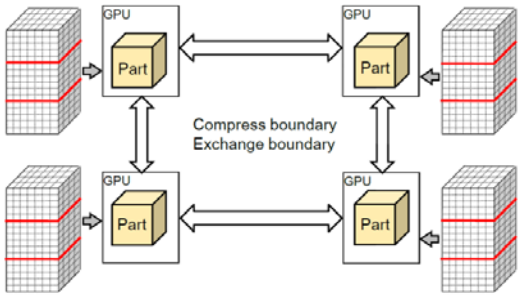
1GPU版7点ステンシル (TSUBAME2.5 (96GB node), 1GPU)

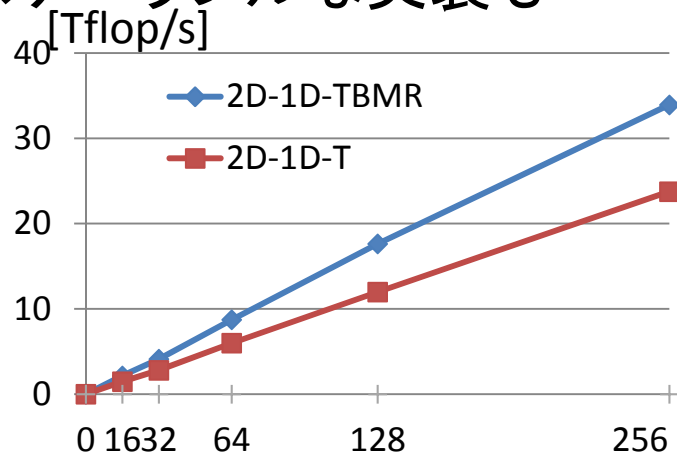


局所性向上により、性能オーバヘッドほぼ無しで、問題サイズを20倍拡大に成功！

6GB → 125GB

マルチGPU向けのスケーラブルな実装も





Weak scalability on TSUBAME2.5
48GB per GPU (> 6GB)

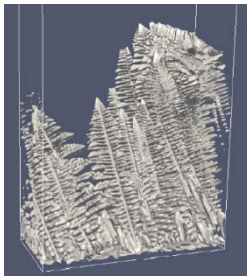
256GPU/256ノードで

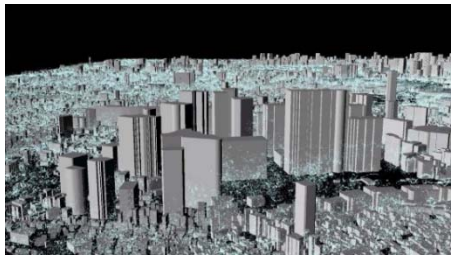
- 34TFlops
- 21TB/s
- 12TB (>3GBx256)

を～120kWで実現

高性能と大規模の両立を、
(人工ベンチでなく)実アプリケーションで達成したい！

ステンシル系の実アプリをまず主対象とする





メモリ階層の
- 高速・小容量メモリ
- 低速・大容量メモリ

を活用したい

本チームの目指す姿

Step 1. 局所性向上・メモリ最適化可能かの定性・定量評価

⇒「メモリ局所性プロファイラ」(佐藤G)

Step 2. ユーザ透過で効率的なメモリ階層間移動

⇒ 階層対応メモリ管理システムにより実現

– GPUメモリ⇔ホストメモリ: 「HHRT」(遠藤G)

– ホストメモリ⇔高速Flash⇔遠隔メモリ: 「DLM」など(緑川G)

Step 3. 局所性向上技術の統合

⇒「動的コンパイレーション技術」(佐藤G)

– 時間/空間ブロッキングに伴うループ変換をどこまで自動化可能か見極め

メモリプロファイラ＋動的コンパイレーション(佐藤G)

アプリのデータレイアウトとループ階層構造をコード実行時に変換し、自動的に局所性向上を行う動的コンパイラツールチェーンの研究開発

ソースコード → コンパイラ → 実行バイナリコード → コード実行 + 動的コンパイレーション → メモリプロファイリング or 透過的チューニング

プログラマによる
- ・アルゴリズム選択
- ・人力での最適化

命令レベルの最適化が中心
- ・レジスタの割り当て
- ・命令選択
- ・細粒度最適化

ランタイムの情報を踏まえたコード変換

入力データを踏まえた様々なシナリオに対応
既存のコンパイラを超える広域最適化

ループ階層構造の最適化

- ループブロッキング
- ループ交換、アンローリング
- ループ融合、ループ分割

データの配置・レイアウト最適化

- 初期配置決定・階層間移動
- タイリング、ループ構造に合わせたデータ分割
- Structure of array 対 Array of structure

実行中の動的な情報

- 膨大なソース中の関数にまたがった局所性
- 動的なワーキングセット変化
- キャッシュ＋各メモリ階層の性能特性

ループのパラメータ制御

body size, # itr,
data transfer,
data dependence

メモリ階層へのマッピング

- キャッシュ、メモリバンド幅の制御
- アクセラレータやスレッド並列への対応

メモリプロファイラ＋動的コンパイレーション(佐藤G)

静的情報 (ソースコード) and/or
動的情報 (runtime exe)

進化するメモリ階層構造の
局所性を最大限活用

局所性を向上させる透過的チューニングと最適化

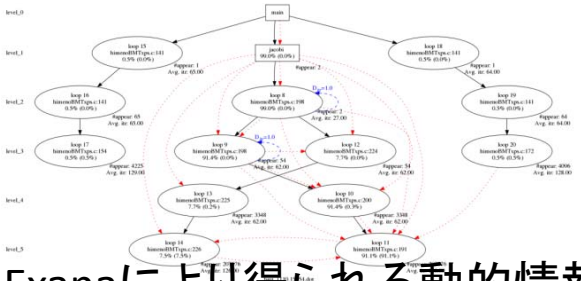
1. ループ構造の最適化
   - loop blocking
   - loop unrolling, loop exchange
   - loop fusion, distribution

2. メモリレイアウトの最適化
   - Initial placement, movement among different hierarchies
   - Structure of array v.s. Array of structure

Exana: EXcution-driven application ANAlysis tool

鍵となる特徴:

1. **メモリアクセス局所性プロファイラの開発**
2. **性能推定のためのメモリモデルとそのコード最適化/チューニング戦略への活用**
3. **バイナリ変換やソース変換によるコード最適化の実現**



Exanaにより得られる動的情報:
- 動的なデータ依存関係
- ループ階層や回転数

メモリアクセスの挙動とランタイム情報

性能チューニングの鍵とは？

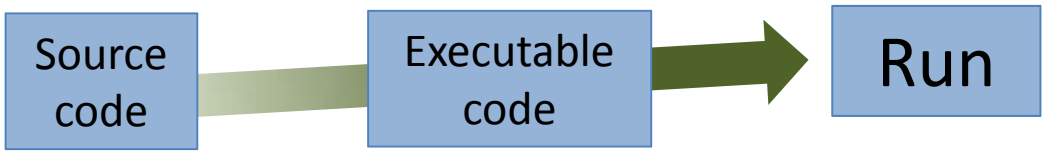
メモリレイアウト
データ構造
ループ階層、回転数

これらは相関があり、以下により制御可能

padding, cache alignment,
loop transformations such as
loop blocking, fusion

これら性能の鍵となる特徴量は最終的には "ランタイム" に決定

メモリの最適化はランタイムの挙動に着目する必要がある



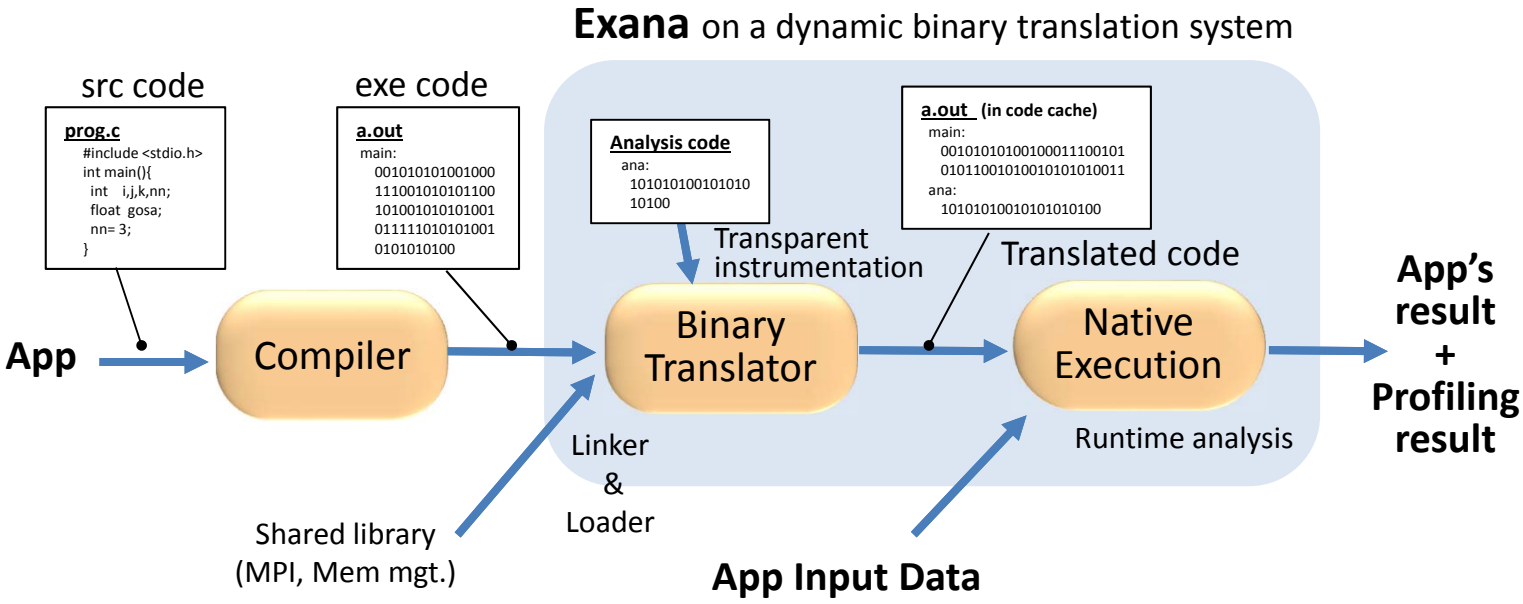
WYSINWYX
[Balakrishnan, ACM TOPLAS2010]
sensitive to compiler and its optimization

Register allocation
Stack memory allocation
Instruction selection

sensitive to input data

underlining HW
OS memory management
runtime resource status
access patterns to array

The Exana tool set



Exana: "An Application Profiling and Optimization Infrastructure" for Accelerating Systems with Deeper Memory Hierarchy

Exanaを用いて本プロジェクトの鍵となる要素技術を実装

1. メモリアクセス局所性プロファイラの開発
2. メモリアクセスパタン解析や性能推定のためのメモリモデル
3. 動的バイナリ変換による透過的メモリチューニング/コード最適化

MemPat + RecPat　：メモリアクセスパターン解析

Binary code
+ input data

Exana

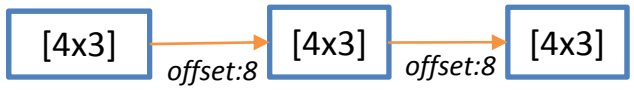
Memory access trace

```
W8@4007ce 7fffa9876810
W8@4007d2 7fffa9876808
W8@4007dc 7fffa9876788
R8@400638 602540
W8@40063e 7fffa9876780
R8@400618 602528
:
```

MemPat

MemPat により以下の4つの基底パターンに分類
(1) Fix, (2) Seq, (3) Fix Stride, (4) Seq Stride

Ex. Seq Stride Access:
R4@4005ac = SeqStr:[4x3](_8_[4x3])*2

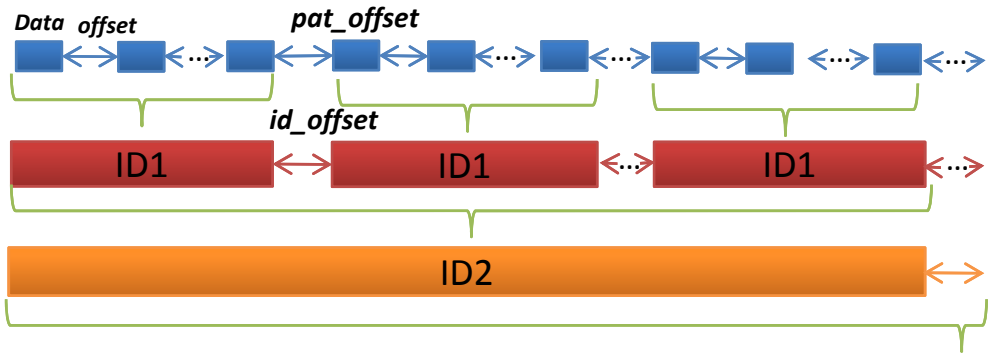


基底パターンに分類できない場合は
複合パターンとしてオフセットとともに保持

R4@4006ab =　{
基底パタン
+offset+　基底パタン
⋮
+offset+　基底パタン
}

RecPat

RecPat は複合パターンが再帰的な出現を検出し、階層パターンとしてパタン間オフセットとともに表現する



Exana tool setの使い方

実行方法： バイナリを実装する際にExanaとつける

% Exana -- ./a.out

出力は static.out、prof.out、memTrace.out、lcctm.datファイル

実装された機能および抽出可能な情報

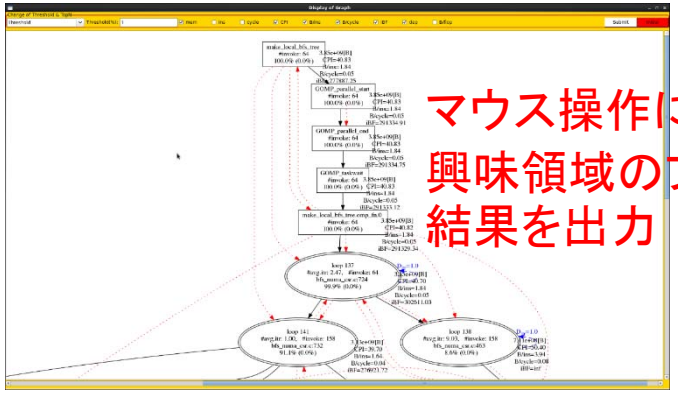
- メモリ参照に関する情報
- ループ領域単位のデータ依存関係 [Sato, IISWC'12]
- ワーキングデータセットサイズ
  各領域のメモリフットプリント [Sato, AMAS-BT'13]
- 区間時間計測 [佐藤, SWoPP'13]
- メモリトレースのモデル化
  パターン化 [松原, SWoPP'13]
- ダイナミックコンパイレーションのプロトタイプ [IWIA'14]

結果のインタラクティブ可視化

% ./vizLcctm lcctm.dat &

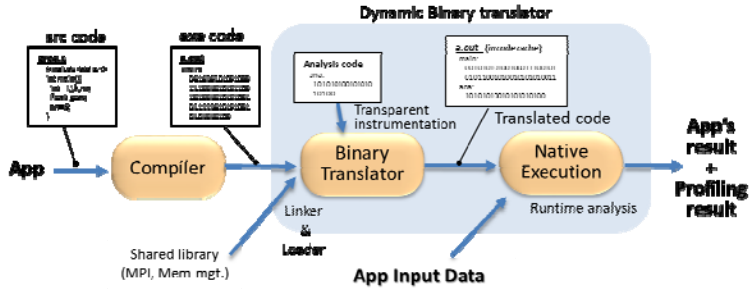
Exana オプションの例
（コマンドのヘルプにて表示）

-mode [default LCCT]
Specify a mode option, CCT, LCCT or LCCT+M
-loopID [default -1]
specify a particular loopID if you focus only on
-apr [default -1]
specify a range of apr count such as N:M
(N: start aprCnt, M: end aprCnt)
-itr [default -1]
specify a particular tripCnt or the range of loop iterations
such as N:M (N: start tripCnt, M: end tripCnt)
-memtrace [default 1]
Turn on memory trace (default, 1) or turn off (0)
-pageSize [default 64B]
A page size of the working data set analysis
[64B, 128B, 256B, 4kB, 64kB]

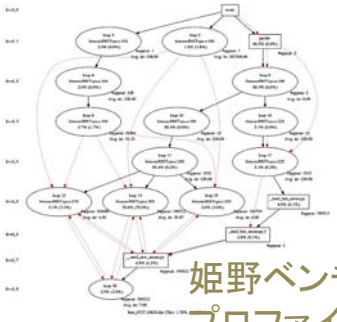


マウス操作により
興味領域のプロファイル
結果を出力

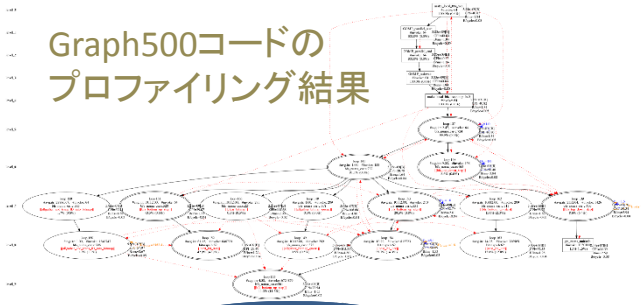
実アプリケーションチューニングのプロファイリング



Exana: EXecution-driven application program ANAlysis Tool



姫野ベンチマークの
プロファイリング結果



Graph500コードの
プロファイリング結果

Exanaによる
プロファイリング

佐藤グループ

コード最適化計画作成
手法にフィードバック

遠藤グループ
テンポラルブロッキング化ステンシルコード

姫野ベンチマークに高度なループ変換の事例としてテンポラルブロッキングの適用

CREST藤澤チーム
グラフ解析処理

Graph500のメモリアロケーションや局所性向上を狙うチューニング手法の適用

理研　南チーム
京向け実アプリ

NICAM-DCをプロファイリングしチューニングにフィードバックすることを検討中

プロジェクトの進行に合わせてツールを公開予定

機能検証・有効性実証に協力していただけるなら先行的にプロトタイプ版を提供

Exanaの実用性

Exana動作実績プラットフォーム

スパコン	Cray XC30
スパコン	SGI Altix UV
スパコン	TSUBAME 2.5
アクセラレータ	Xeon Phi
汎用	x86 Linux

各種コンパイラ、MPI、マルチスレッド共有ライブラリ、動的・静的リンク、バッチジョブ、子プロセスのフォーク、再帰、各種言語（C/C++, Fortran）にも対応

Exanaで検証したアプリ

	アプリ名	備考
Full app	NICAM-DC	理研アプリ
	OpenMX	JAISTアプリ
	数値タービン	東北大FS
	Graph500	藤澤チーム
Mini app	NICAM-DC-min	fiber
	ffvc-mini	fiber
	ccs-qcd-mini	fiber
	modylus-mini	fiber
	CloverLeaf	Mantevo
	CoMD	Mantevo
Benchmark	SEPC CPU 2006	単体コア向け
	NAS Parallel Benchmark	
	HimenoBMT	
	ステンシル	テンポラルブロッキング

Exanaでの解析の実例

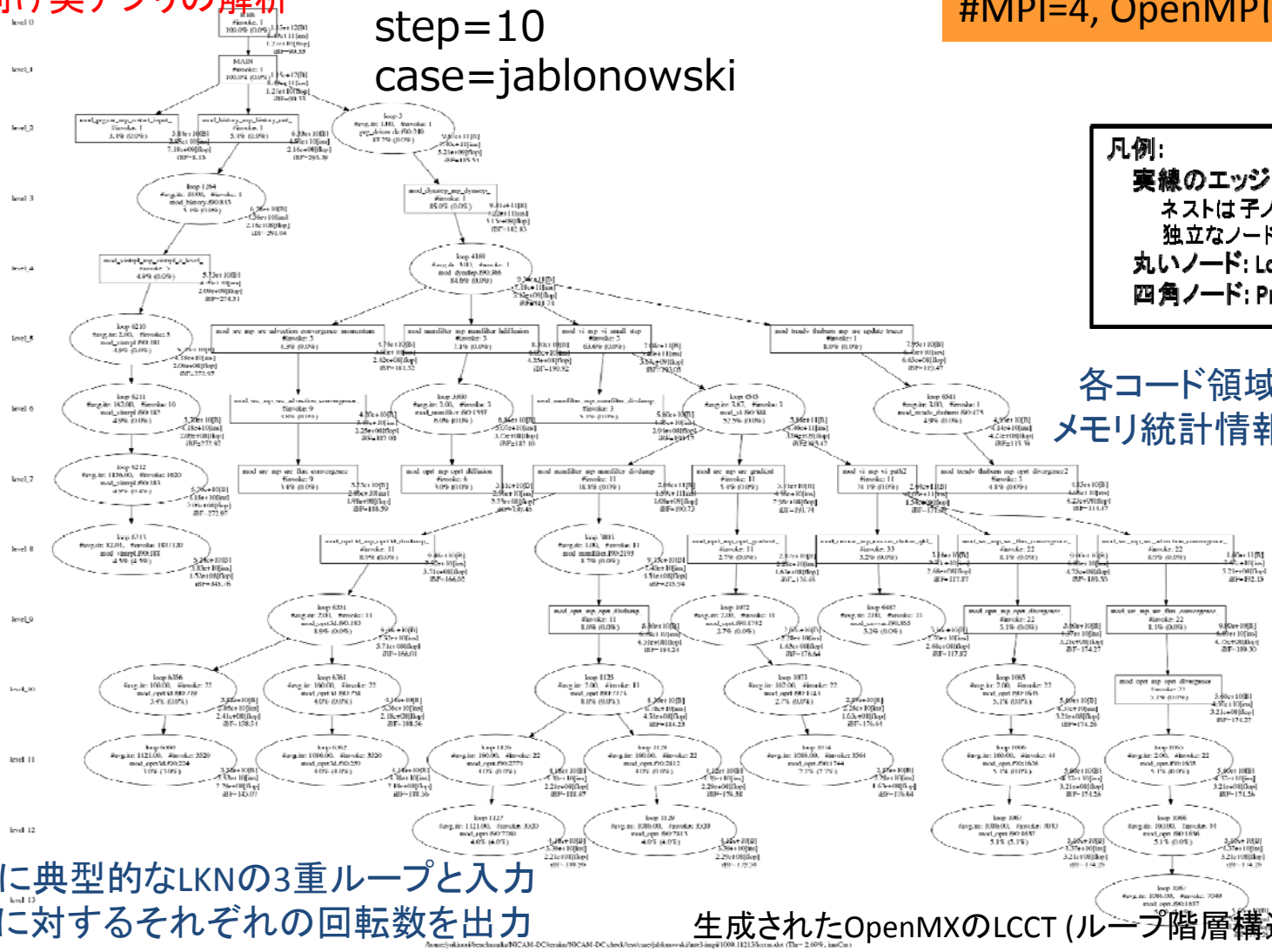
理研　南チームと共同で
の京向け実アプリの解析

NICAM-DC
step=10
case=jablonowski

‘ifortran -g’にてコンパイル
#MPI=4, OpenMPはオフ

凡例:
実線のエッジ: Control flow
ネストは子ノード、
独立なノードは兄弟として表現
丸いノード: Loop
四角ノード: Procedure call

各コード領域のB/F値や
メモリ統計情報を出力可能



NICAMに典型的なLKNの3重ループと入力
データに対するそれぞれの回転数を出力

生成されたOpenMXのLCCT (ループ階層構造)

メモリを介したデータ依存関係の検出

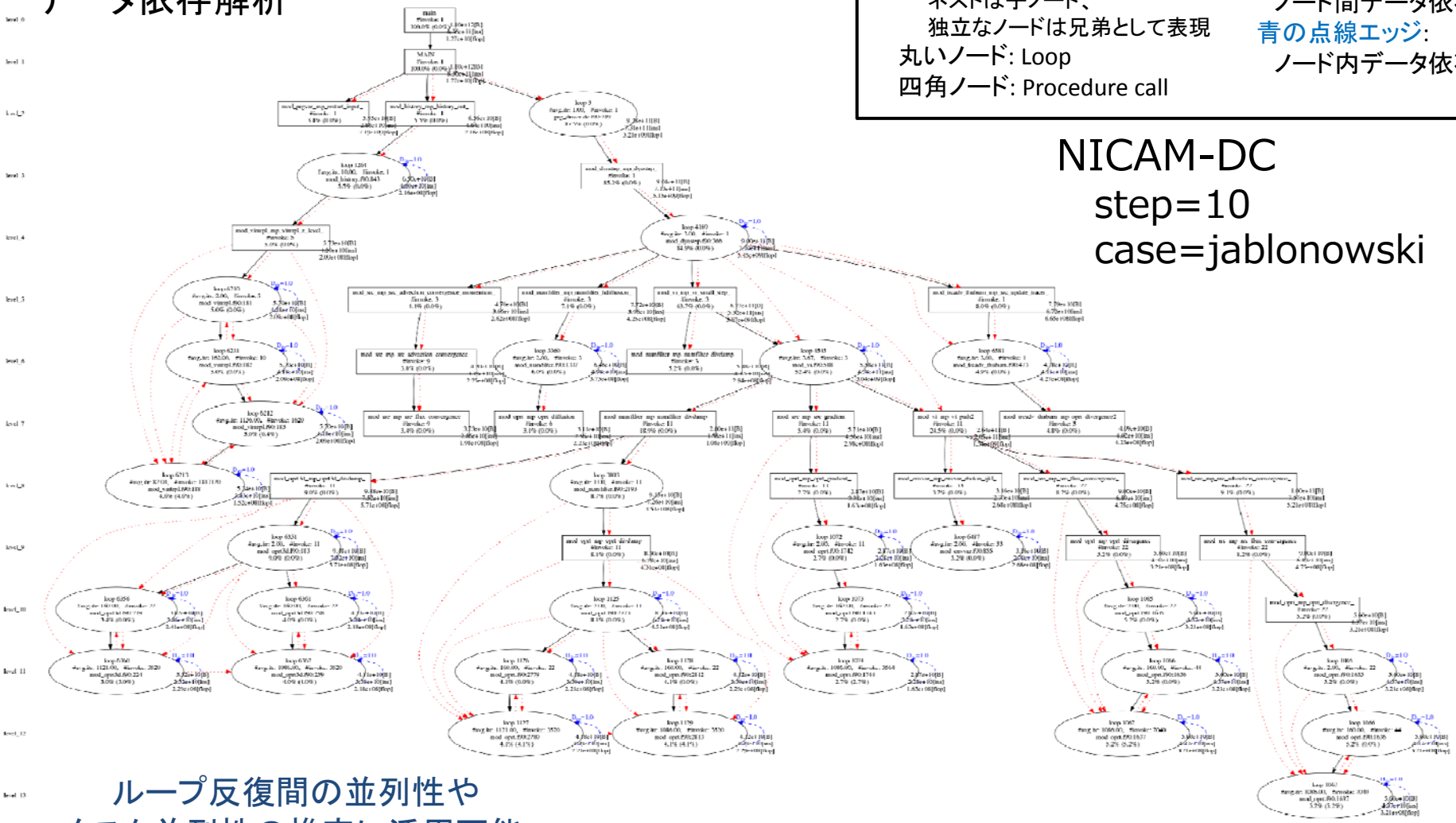
ループ階層構造、関数を単位として
データ依存解析

凡例:
- 実線のエッジ: Control flow
  - ネストは子ノード、
  - 独立なノードは兄弟として表現
- 丸いノード: Loop
- 四角ノード: Procedure call

赤の点線エッジ:
ノード間データ依存

青の点線エッジ:
ノード内データ依存

NICAM-DC
step=10
case=jablonowski



ループ反復間の並列性や
タスク並列性の推定に活用可能

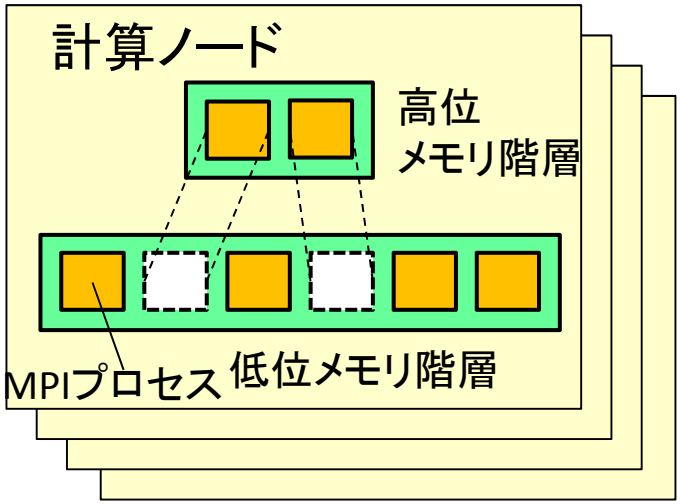
階層対応メモリ管理システム(1) の開発 (遠藤G)

既存の並列アプリ (例: MPI+CUDA)を、少ない変更でメモリ階層に対応させるメモリ管理システムの研究開発

階層対応ライブラリHHRTの開発(Hybrid-Hierarchical RunTime)　[Cluster14]

何をするか:

- MPI + αで記述された既存並列プログラムが、最小限の変更にて動作する
  - 現在のGPU版HHRTは、MPI+CUDAアプリを対象
- 仮想プロセス(VP)によるGPUの多重化
- スワップ機能により、広いメモリ領域を利用可能に
  - ページ単位でなく、プロセス単位のスワップ
  - デバイスメモリ – ホストメモリの階層。将来他階層へ



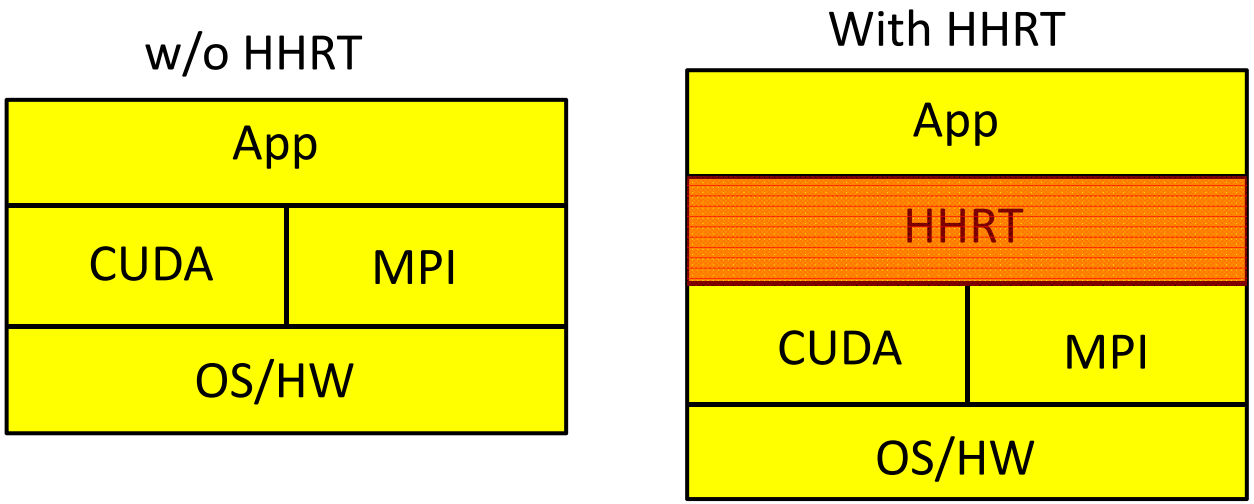
HHRTの実行モデル。
UIUC Charmに近いが、メモリ階層に対応

何をしないか:

- 新しいプログラミングモデルやDSLの提案ではない
- ランタイム自身が局所性向上を行うわけではない

The HHRT Library

- HHRT supports applications written in CUDA and MPI
  - HHRT works as a wrapper library of CUDA/MPI

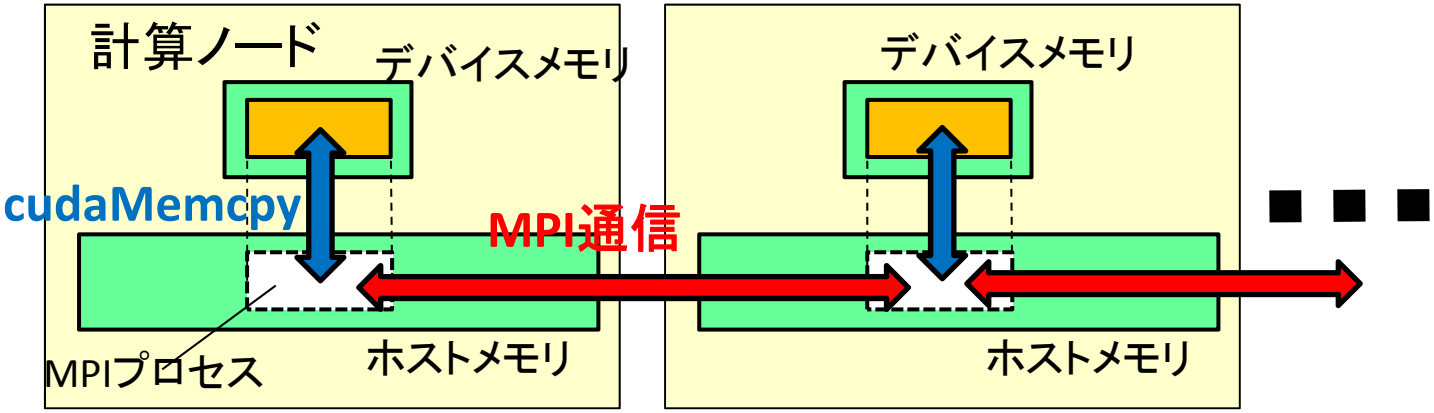


Functions:

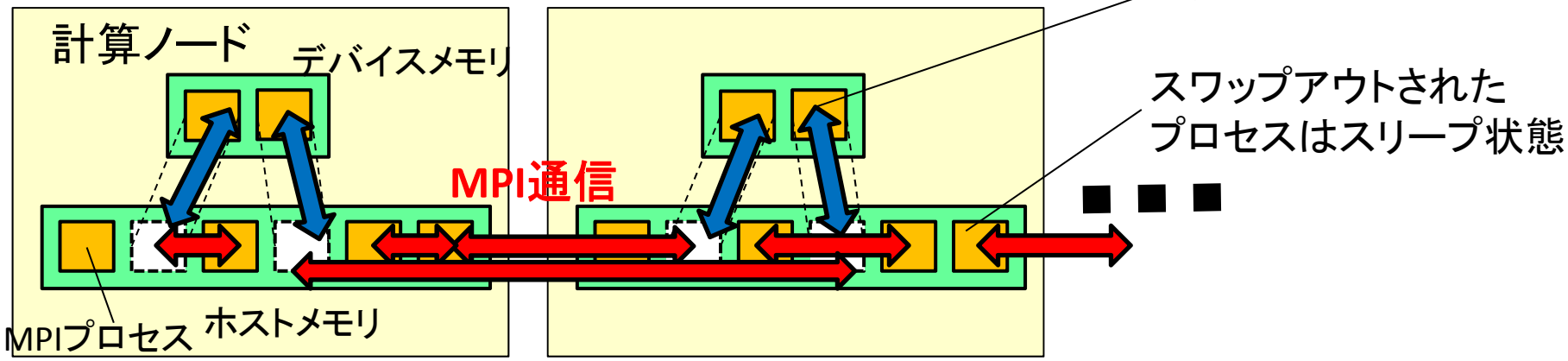
- HHRT supports overprovisioning of MPI processes on each GPU
- HHRT executes memory swapping between device memory and host memory
  - Not “page-wise” swapping, but “process-wise” swapping

HHRT(GPU版)の実行モデル

通常のMPI+CUDA実行モデル



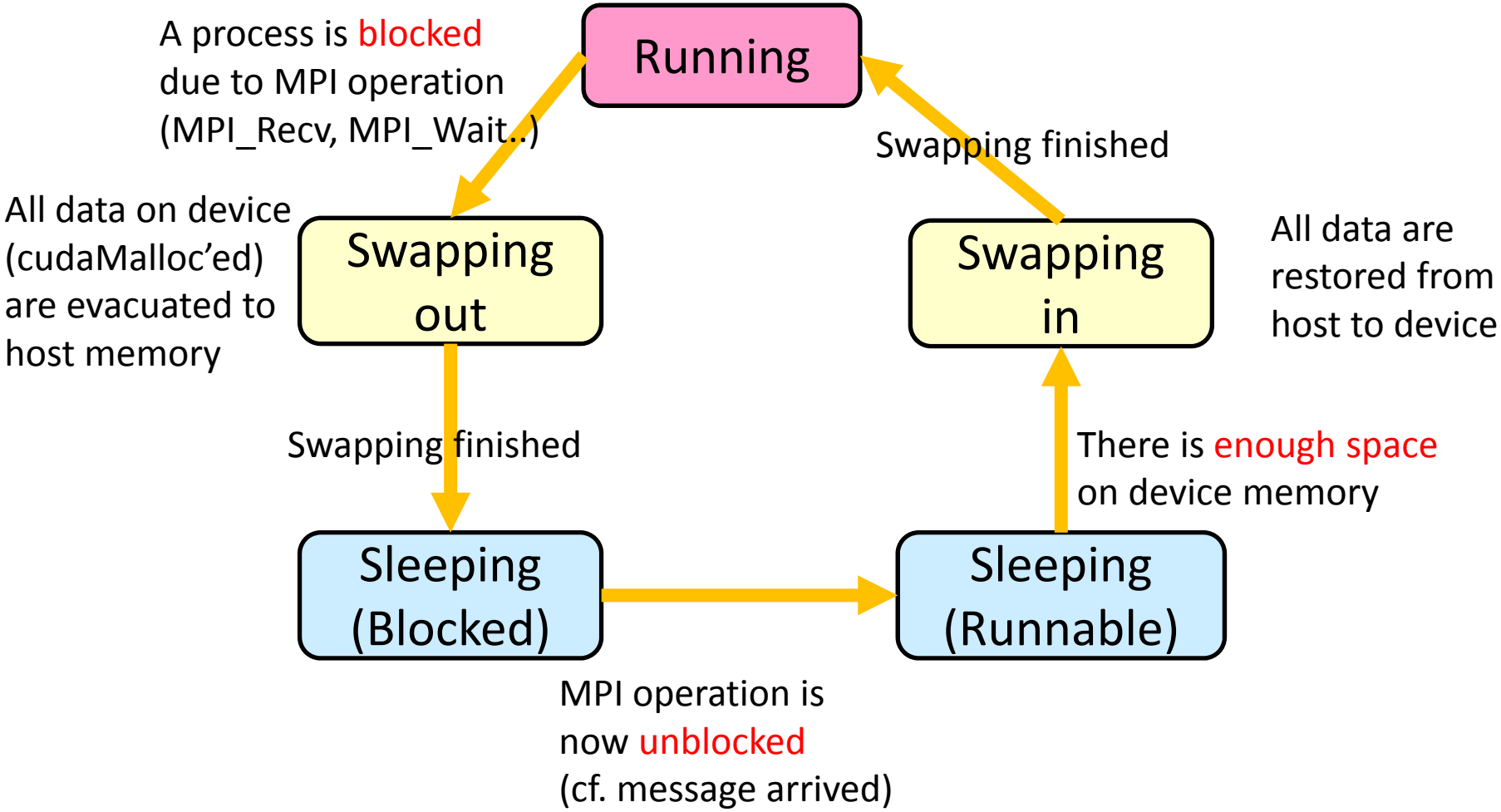
HHRTの実行モデル



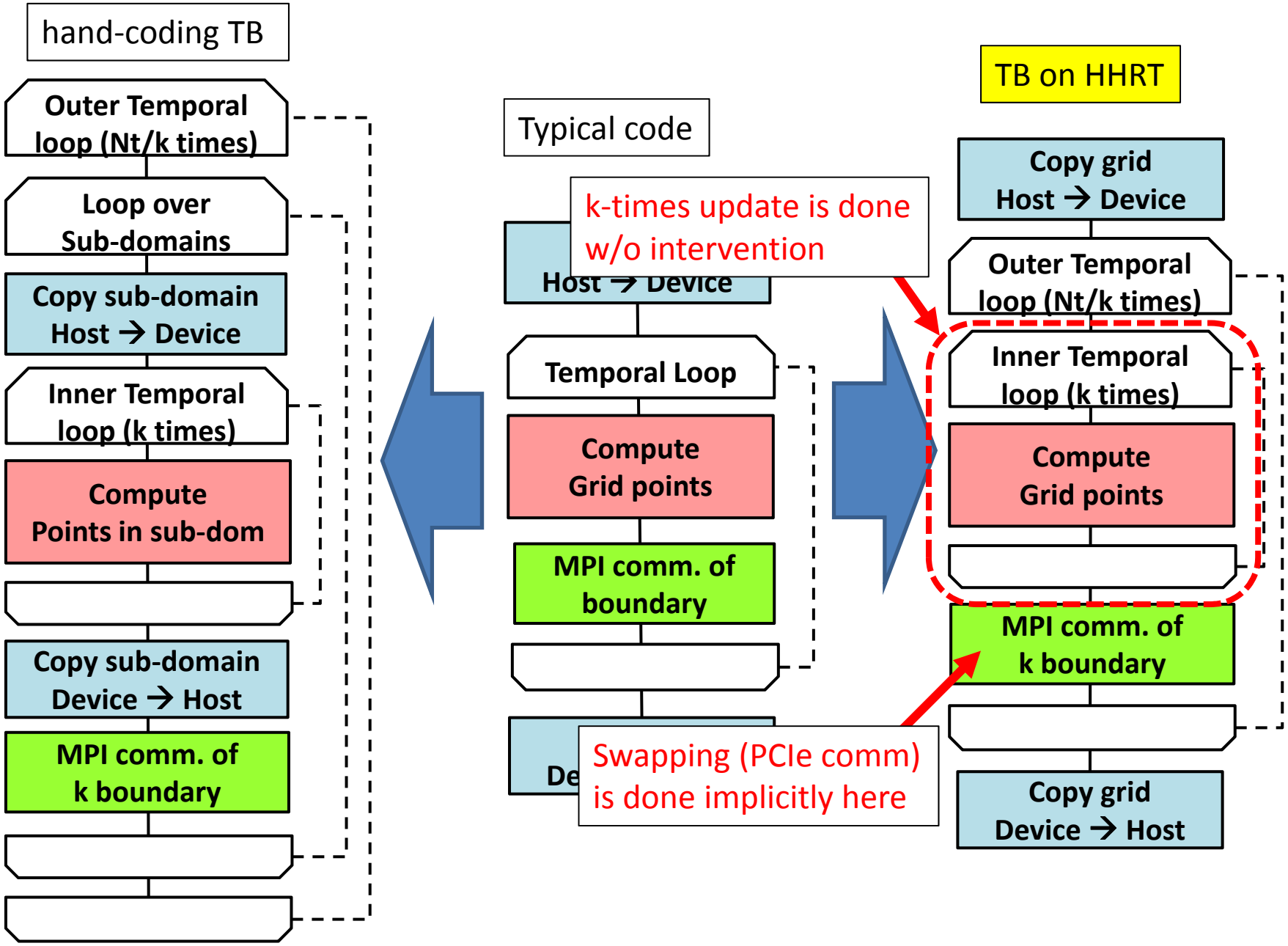
各プロセスは時々(MPIブロック時)スワップアウトされ、スリープする

→ プロセス達がデバイスメモリを時分割することにより、プロセス間合計で容量限界を超える

State Transition of Each Process



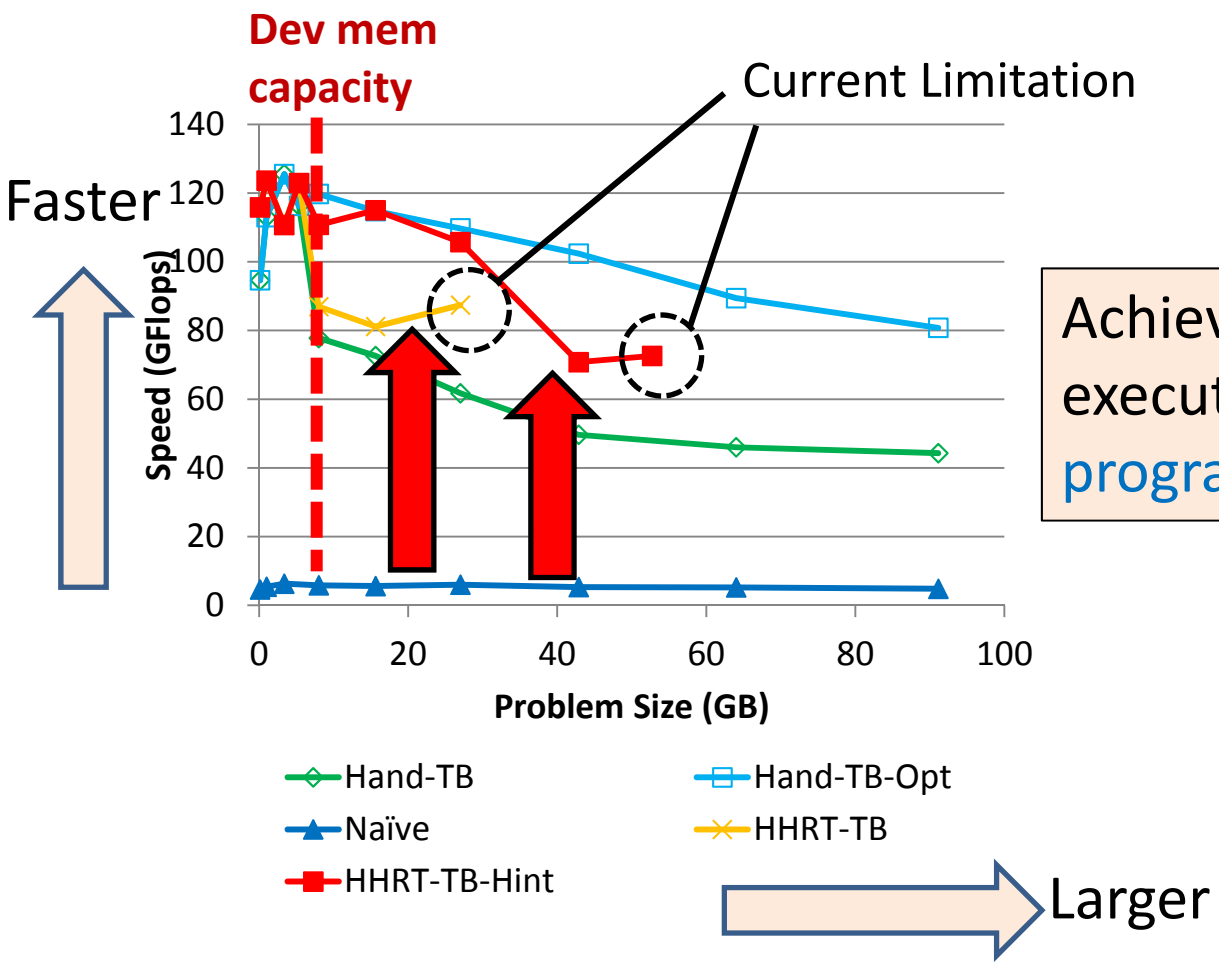
HHRT上でのテンポラルブロッキングの実装



HHRTを用いた7点ステンシルの性能

3D 7point stencil on a TSUBAME 96GB node
A K20X GPU (6GB GPU mem) is used

- SLES Linux 11SP1
- CUDA5.5
- OpenMPI 1.6.3
- gcc 4.3.4



Achieving **fast&(fairly) big** execution with moderate programming cost

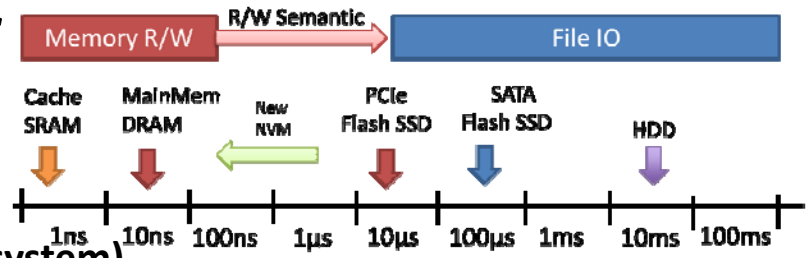
階層対応メモリ管理システム(2)の開発 (緑川G)

垂直方向記憶（記憶階層：メモリ,SSD, HD），水平方向記憶（多ノード連携，局所・遠隔メモリ、共有メモリ・プール）利用するためのシステムソフトウエアの開発

1. 垂直方向記憶階層間のキャッシュ/スワップ方式

- 次世代NVMを省電力・大容量メモリとして利用

  現在利用可能なNVMとして
  PCIeバス接続型Flash SSD、



  最新OSカーネルによる評価 （tastSwap, io subsystem）

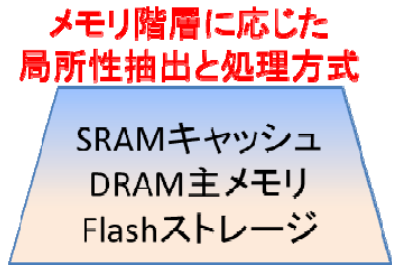
- 物理メモリDRAMサイズを超える大規模ステンシル計算

  アクセス局所性を高めるアルゴリズムを導入

  NVM向けに3方式（aio, mmap, swap)を実装、評価、最適化

- 各種NVMの特性を生かすハイブリッド記憶の利用の模索,調査

  （DRAM + NVM（ReRAM, MRAM) + FLASH + HDD）



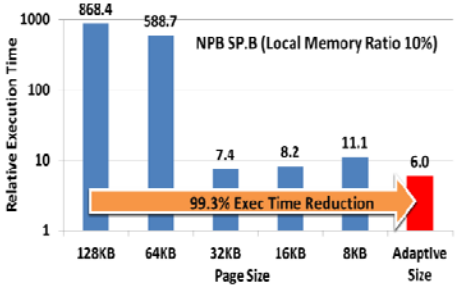
2. 水平方向記憶の連携

- 遠隔メモリサーバ

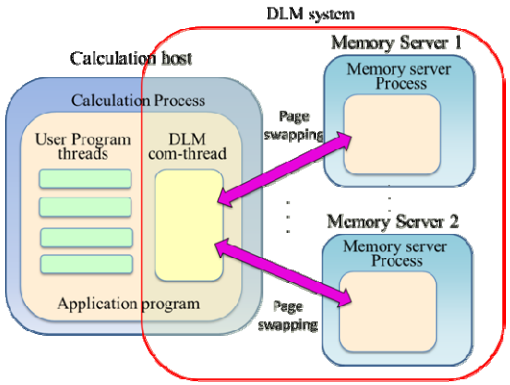
  ノード間通信（MPI3新実装）評価調査

- ページサイズ自動変更機構

  マルチスレッド向けの設計, 実装



自動ページサイズ調整

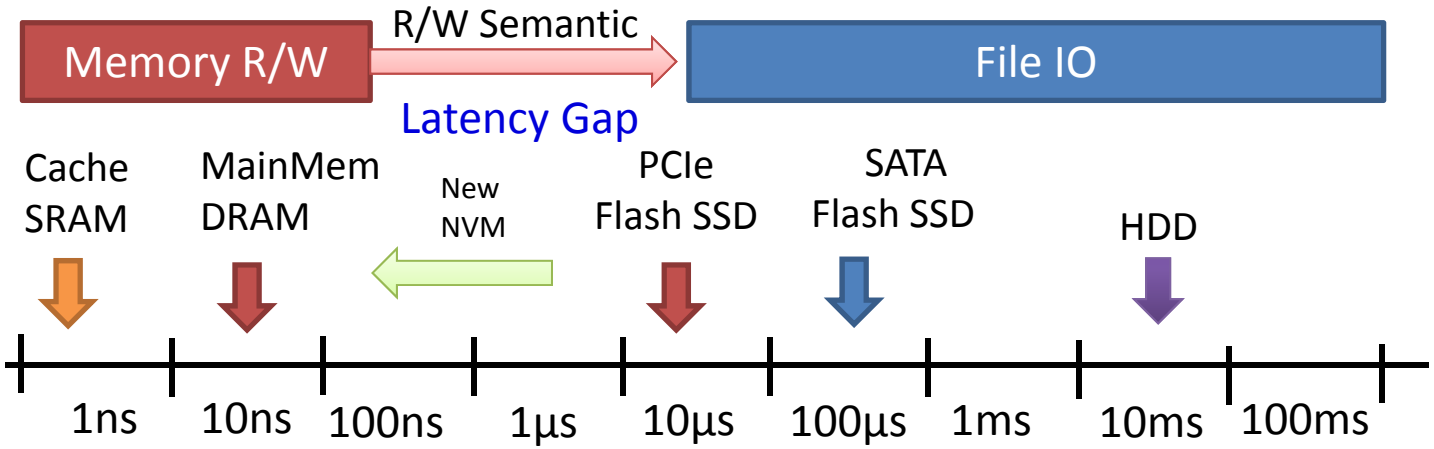


3. 応用プログラム分析によるデータ配置

メモリアクセスログ解析，ページアクセスモニタ, 事前データ配置，実行時データ移動

遠隔メモリサーバ

Out of Core Computation Algorithm extracting Data Access Locality



- The Gap between DRAM and PCIe-Flash SSD

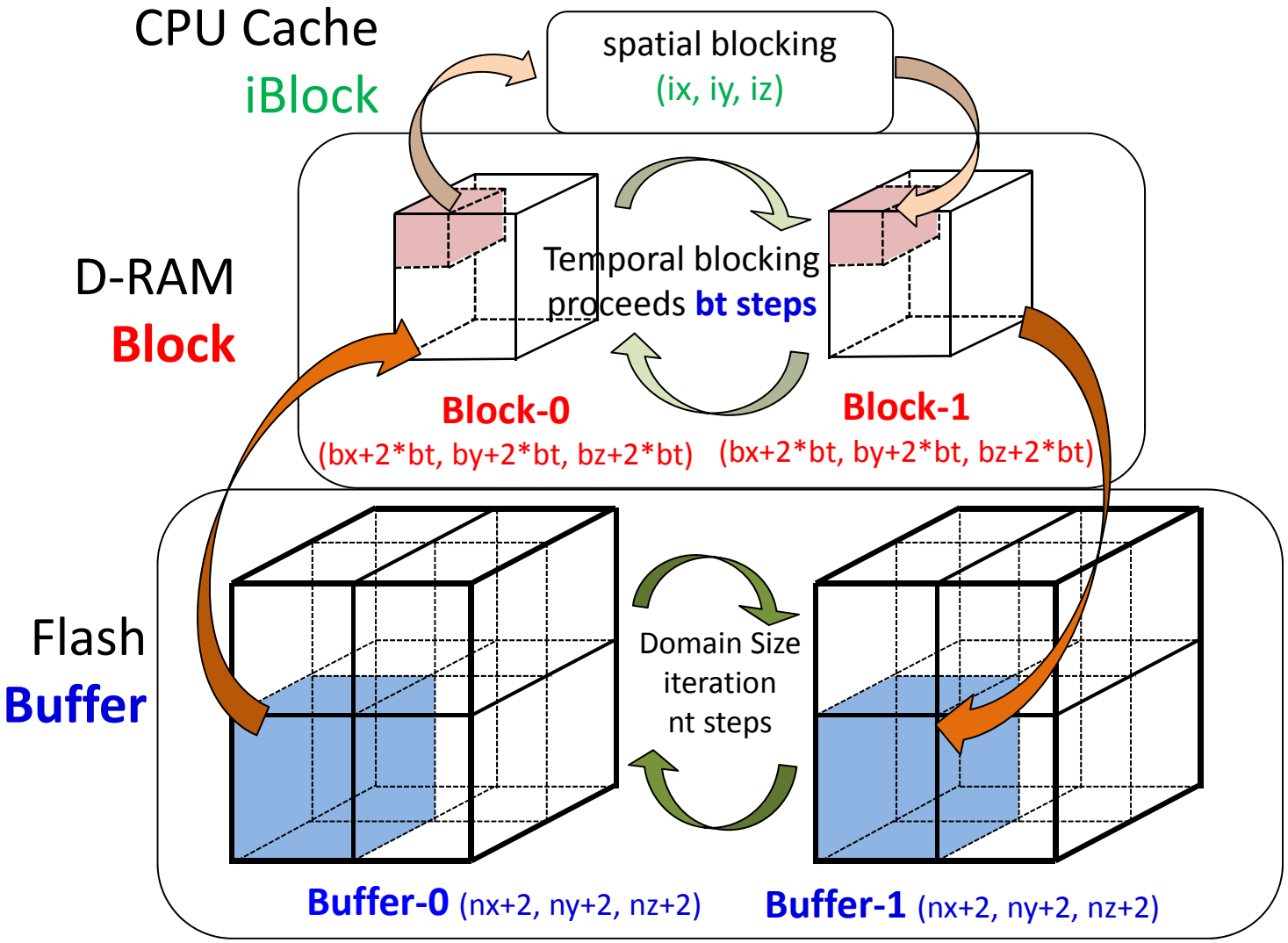
  > the Gap between Cache and Main memory (DRAM)

Stencil Computation

The temporal blocking algorithm is newly designed for DRAM-Flash tier to bridge their latency divide.

copyright H.Midorikawa

Temporal & Spatial Blocking for three-level Memory Hierarchy



copyright H.Midorikawa

Stencil Comp. using Flash SSD

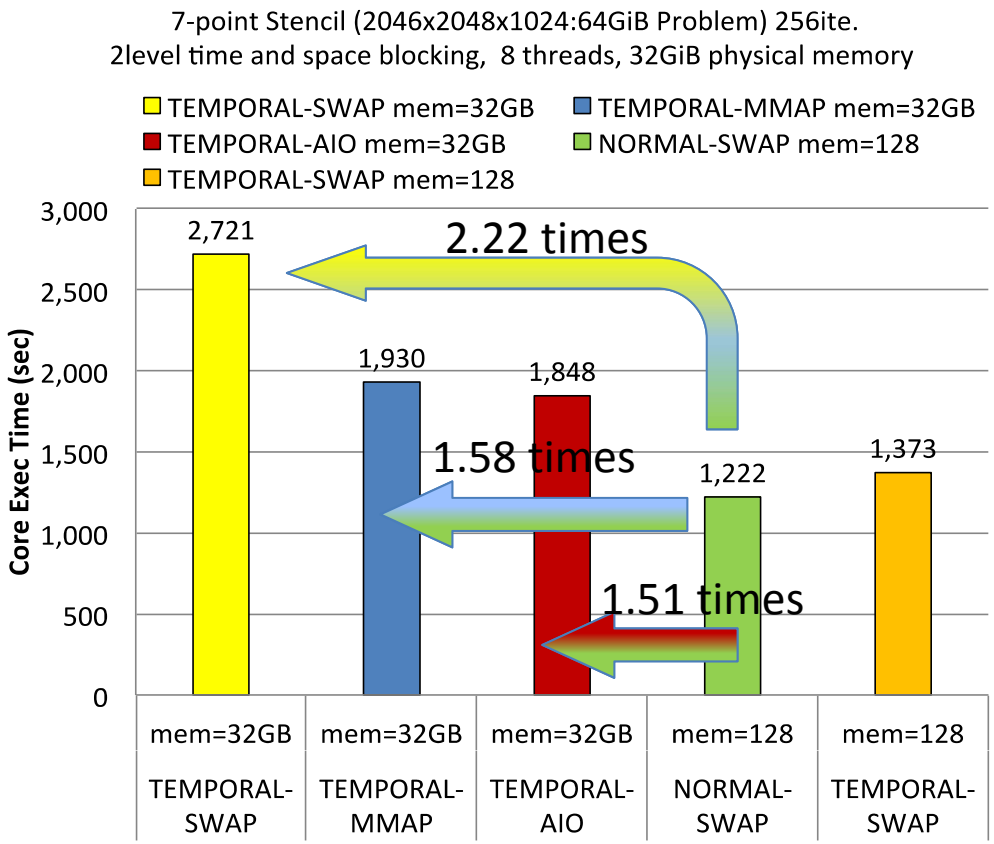
for processing a larger-size problem than physical memory (DRAM) size

Three implementations

- swap method (swap system)
  uses a flash as a swap device
- mmap method (file memory map)
  uses a flash as a ext4 file system
- aio method (asynchronous file io)
  uses a flash as a block device for direct input/output

copyright H.Midorikawa

Preliminary Evaluation
Effect of Temporal blocking



With spatial blocking and temporal blocking

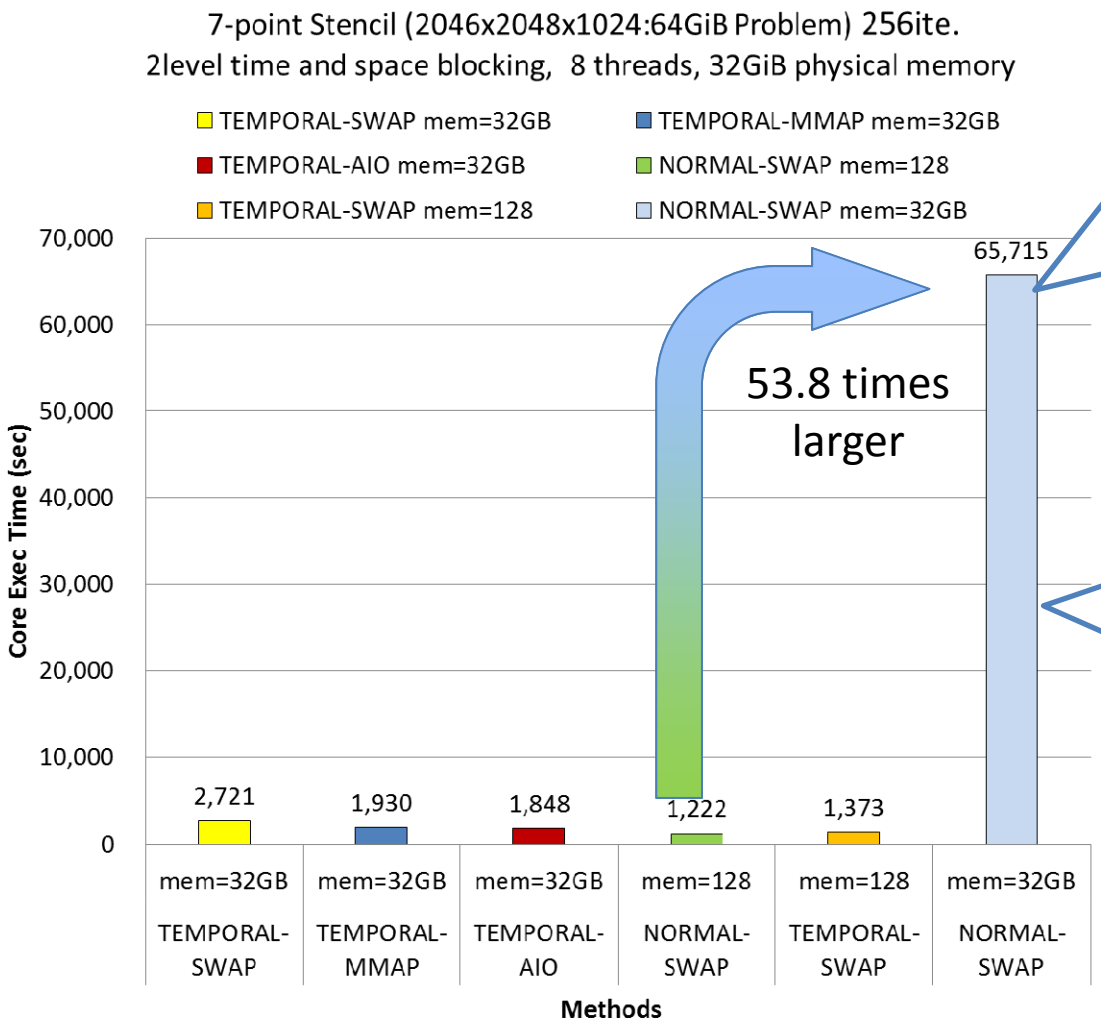
- TEMPORAL-SWAP: 2.22 times larger
- TEMPORAL-MMAP: 1.58 times larger
- TEMPORAL-AIO: 1.51 times larger

7-point Stencil
(2046x2048x1024:64GiB Problem) 256ite.
2level time and space blocking,
8 threads, on 32GiB physical memory

CPU	Xeon E5-2650 2.00GHz x1 (8cores)
Memory	32GiB boot (Total memory 128GiB) 16GiB(DDR3 1600MHz ECC Reg) x 8
OS kernel	3.13.0
Compiler	gcc 4.4.7 20120313 -O3
SwapDevice	ioDrive2 (FusionIO)

copyright H.Midorikawa

Preliminary Evaluation
Result without Temporal blocking



NORMAL-SWAP on 32GB memory with only spatial blocking takes 18 hours 16 minutes, 53.8 times longer time compared with the normal exec. time on 128GB.

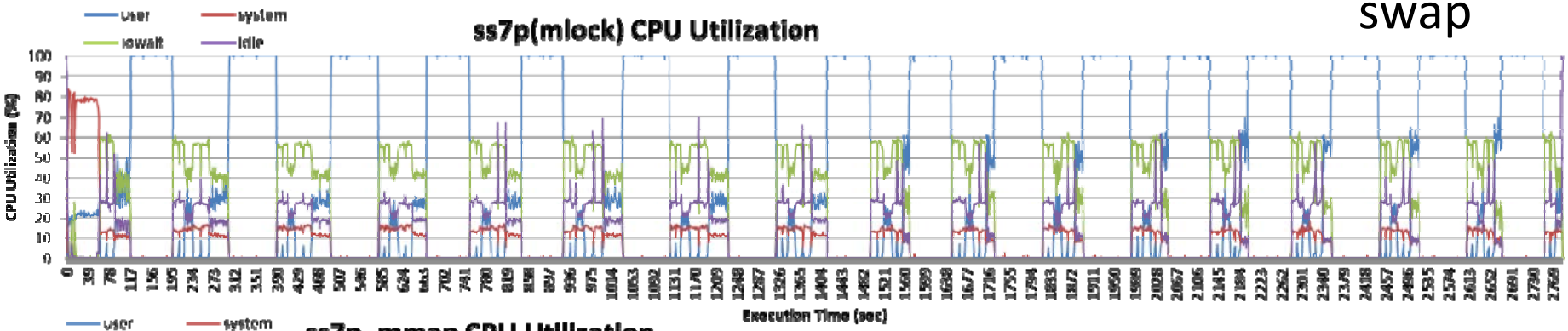
NORMAL-SWAP employs only spatial blocking (32x32) for cache hit

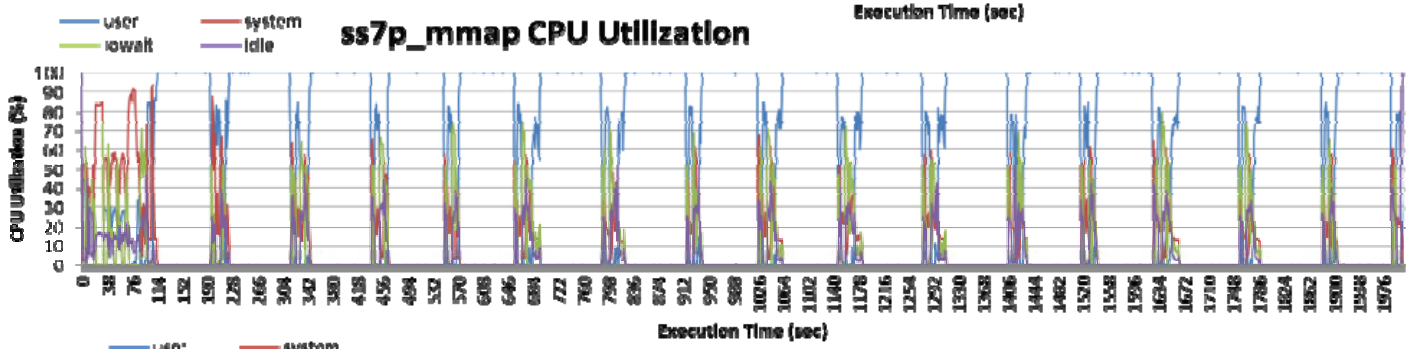
7-point Stencil (2046x2048x1024:64GiB Problem) 256ite.
2level time and space blocking, 8 threads, 32GiB physical memory

copyright H.Midorikawa

Stencil computation
CPU Utilization Profile

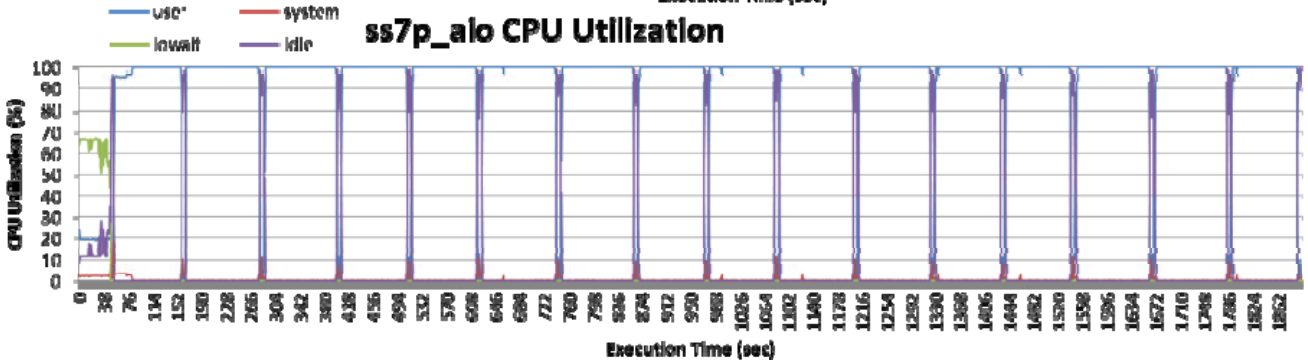
swap





mmap

8 blocks on DRAM
update 2 times



	Total Size(GIB)		Elapsed Time
	read	write	(min.sec)
swap	600.80	124.69	46:24.0
mmap	209.70	156.39	33:45.65
aio	110.25	125.19	31:36.5

aio

TIme

copyright H.Midorikawa

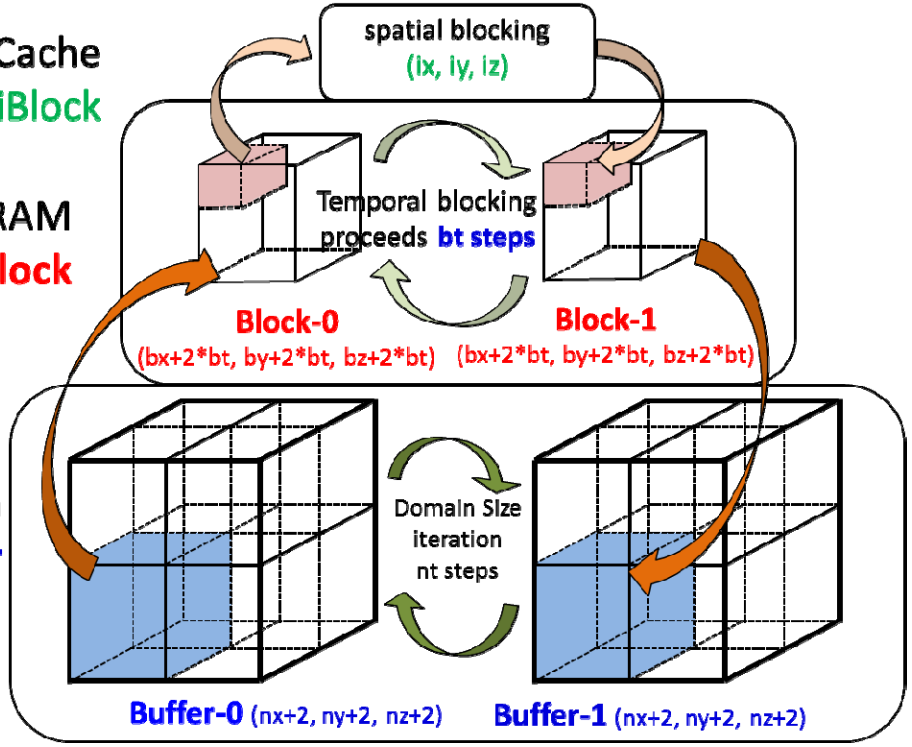
Tuning temporal blocking parameters for higher performance

- Internal block shape
- Thread work share scheme

CPU Cache → iBlock

D-RAM → Block

- Block shape
- Block memory layout



NAND Flash → Buffer

Tuning (parameter select) policy

1. fit block/iblock volume in DRAM/L3 cache size
2. select block shape and a memory layout with device-block-size aligned fashion
   to increase sequential access
   to reduce memory access conflicts by pixel/page padding for block layout
   to increase work share efficiency for iblock shape according to work share scheme
3. select work share scheme to increase core independent and parallel calculation

copyright H.Midorikawa

Runtime Auto Tuning

1. User command parameters

   domain size(nx,ny,nz), Time step(nt), Flash device path

2. Get hardware information
   - get_device_size() device capacity, device block size
   - get_system_info() DRAM size, L3 cache size, # of Cores, # of CPU sockets

3. Calculate optimal blocking sizes by adjusting to underlying hardware
   - Error check (capacity (Flash and DRAM),

     domain lowest-dimension size (device block size multiple) etc.)
   - tune_block_size() returns optimal bx,by,bz, bt for Blocks on DRAM
     - Efficient spatial block size and shape, temporal block size
   - tune_inner_size() returns optimal ix,iy,iz for inner blocks on L3 cache
     - Efficient spatial inner block shape and size

       to increase multi-thread computation performance for z loop parallel

4. Numa-aware computing
   - Blocks are divided into sub-block areas according to # of CPU sockets
   - memory-bind & cpu-bind between each sub-block to each local socket
   - locality-aware sub-block computation on each CPU socket using local cores

copyright H.Midorikawa

実アプリケーションにおける技術統合

対象アプリ：LES都市気流シミュレーション
- 丸山T青木G小野寺直幸氏らによる

東京のビル風などを詳細に計算可能
- ステンシルアプリであり、格子ボルツマン法に基づく
- これまでの最大計算実績は、10km四方を1m格子で
  - 10080x10240x512, 総利用メモリ量11TB
  - TSUBAME2.0 4032GPU利用

  → 600TFlops

(2012 Google ZENRIN)

- コードはC+MPI+CUDAで約19,000行

理想の方針

Step 1. 局所性向上・メモリ最適化可能かの定性・定量評価

⇒「メモリ局所性プロファイラ」(佐藤G)

Step 2. ユーザ透過で効率的なメモリ階層間移動

⇒ 階層対応メモリ管理システムにより実現

– GPUメモリ⇔ホストメモリ: 「HHRT」(遠藤G)

– ホストメモリ⇔高速Flash⇔遠隔メモリ: 「DLM」など(緑川G)

Step 3. 局所性向上技術の統合

⇒「動的コンパイレーション技術」(佐藤G)

理想現状の方針

Step 1. 局所性向上・メモリ最適化可能かの定性・定量評価

⇒「メモリ局所性プロファイラ」(佐藤G)

⇒コード目視により、依存が隣接のみであり、Reductionなどが無いことを確認

Step 2. ユーザ透過で効率的なメモリ階層間移動

⇒ 階層対応メモリ管理システムにより実現

– GPUメモリ⇔ホストメモリ:「HHRT」(遠藤G)

– ホストメモリ⇔高速Flash⇔遠隔メモリ:「DLM」など(緑川G)

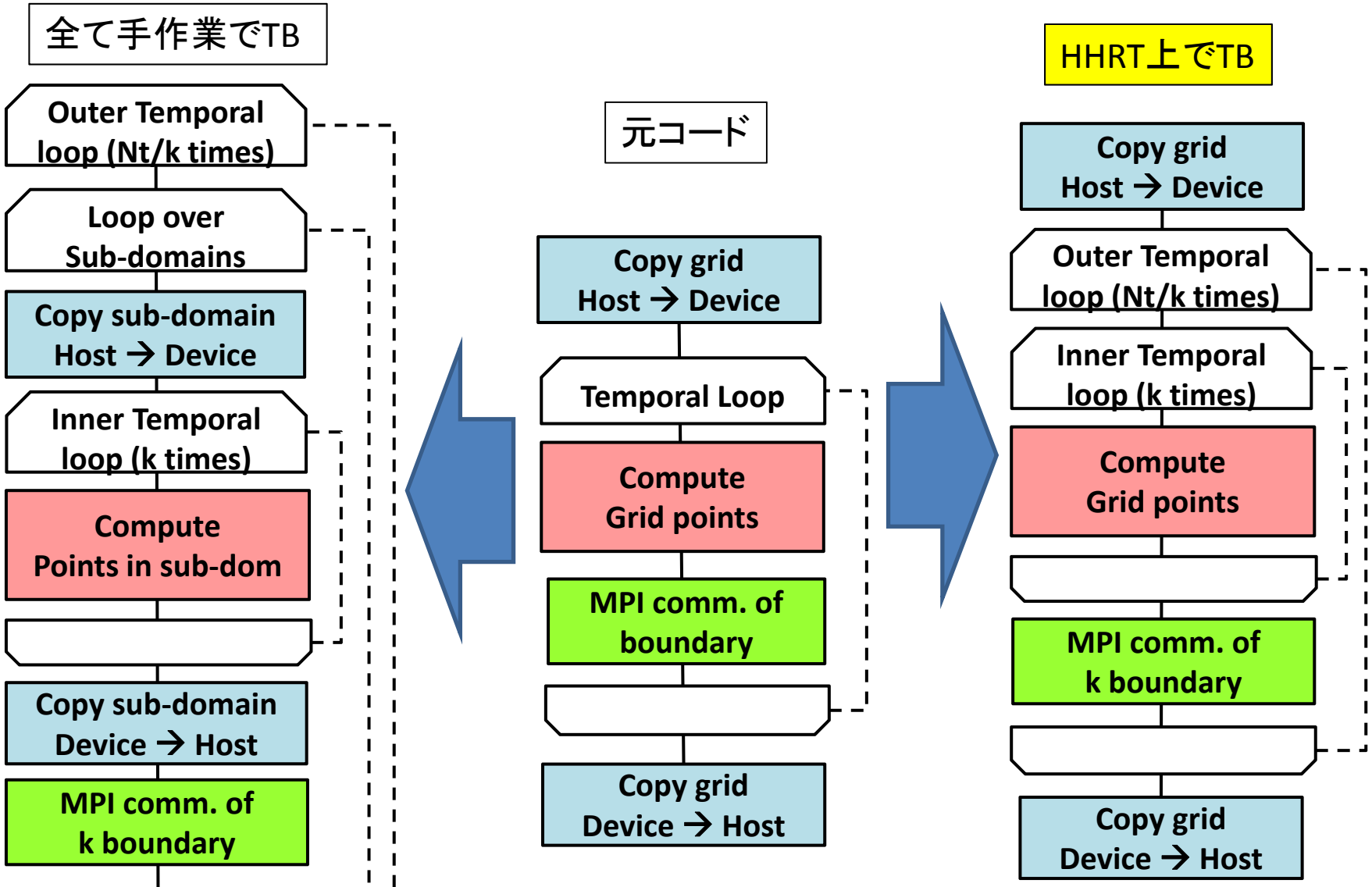
⇒現在利用するメモリ階層は、ホストメモリまで

Step 3. 局所性向上技術の統合

⇒「動的コンパイレーション技術」(佐藤G)

⇒時間ブロッキングのためのループ変換などを手作業で変更。ただしHHRTの実行モデルに基づいており、すべて手作業よりも単純な変更

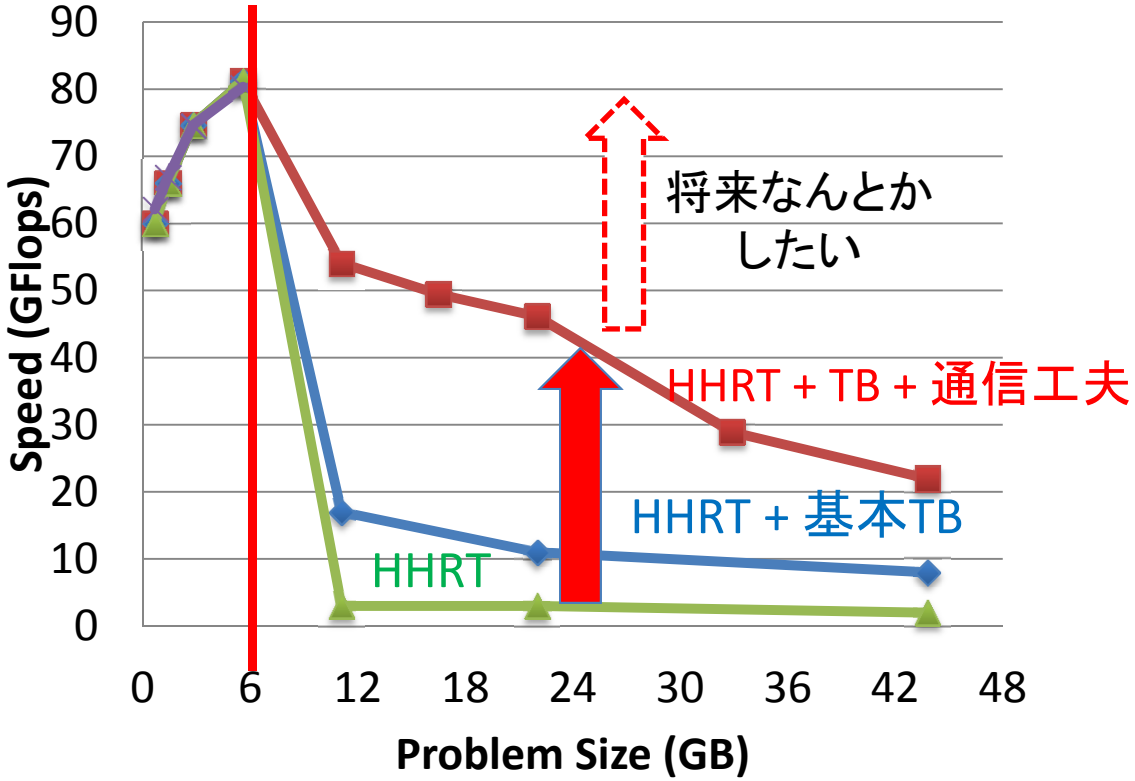
時間ブロッキング(TB)の記述



19,000行のコードに対して約300行の変更
通信最適化のために更に約900行

GPUメモリを超える都市気流の性能(1)

TSUBAME2.5, 1GPU



今のところ、GPUメモリの4倍の問題サイズで、オーバヘッド43%

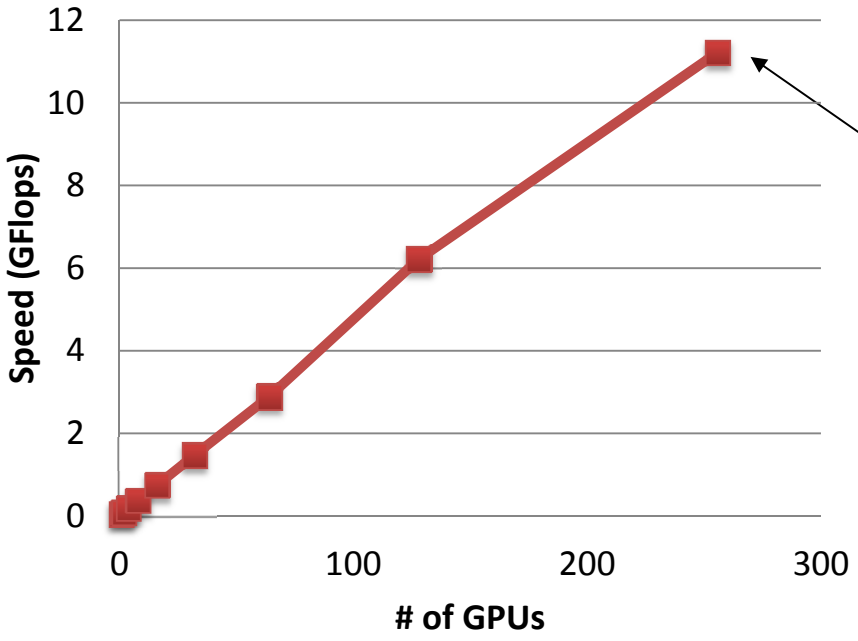
今後の改善方向:

- 冗長計算の除去などの最適化
- HHRTのスケジューリングアルゴリズムの改善
- ブロックサイズ・分割サイズなどの自動チューニング

GPUメモリを超える都市気流の性能(2)

TSUBAME2.5, マルチノード (1GPU per node)

ウィークスケーラビリティ (ノードあたり11GB > 6GB)



性能：11.2TFlops, 5.9TB/s
サイズ：2.8TB

スケーラビリティは良好であり、近日中に
O(1000)GPUで実行予定
課題：プロセス数の増大(#node x ~16)によりMPI自身
がメモリ圧迫

まとめ

- メモリウォール問題の悪化に対し、システムソフトウェア・アーキテクチャ・アプリ分野にまたがったco-designにより問題解決を図る
- <u>次世代気象・医療・防災シミュレーション</u>と<u>次世代メモリ技術</u>とのギャップを埋め、安心安全社会の実現に貢献

  - HMC・NVRAMなどアーキテクチャ分野への要件のフィードバック
  - 局所性向上の自動化・パッケージ化によりアプリ・シミュレーション分野へのフィードバック
  - TSUBAME3.0などポストペタスパコンのデザインへのフィードバック