SS研HPCフォーラム2015　計算科学の新潮流（フロンティア）

SuirenでのHPL及びアプリ性能について（＋Shoubuの話）

国立研究開発法人理化学研究所
計算科学研究機構
エクサスケールコンピューティング開発プロジェクト
コデザイン推進チーム
研究員　似鳥啓吾

はじめに

- 関係各位の皆様、おめでとうございます！
  - 2015年6月のGreen500リスト
  - Shoubu (RIKEN Wako), Suiren Blue (KEK), Suiren (KEK)でのTop3

Listed below are the June 2015 The Green500's energy-efficient supercomputers ranked from 1 to 10.

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	**7,031.58**	RIKEN	Shoubu - ExaScaler-1.4 80Brick, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband FDR, PEZY-SC	50.32
2	**6,842.31**	High Energy Accelerator Research Organization /KEK	Suiren Blue - ExaScaler-1.4 16Brick, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband, PEZY-SC	28.25
3	**6,217.04**	High Energy Accelerator Research Organization /KEK	Suiren - ExaScaler 32U256SC Cluster, Intel Xeon E5-2660v2 10C 2.2GHz, Infiniband FDR, PEZY-SC	32.59

今日話すことになった経緯

- 2014年11月のTop500/Green500のHPL計測（KEK Suiren）を微力ながら手伝わせていただきました
  - 上司の牧野淳一郎からの紹介（と出張命令）
  - 惜しくも2位でしたが、ハードウェアやLU分解についてとても勉強になりました
- 今回の企画にあたって、KEKの石川正先生経由で小柳先生から指名されました

話の内容

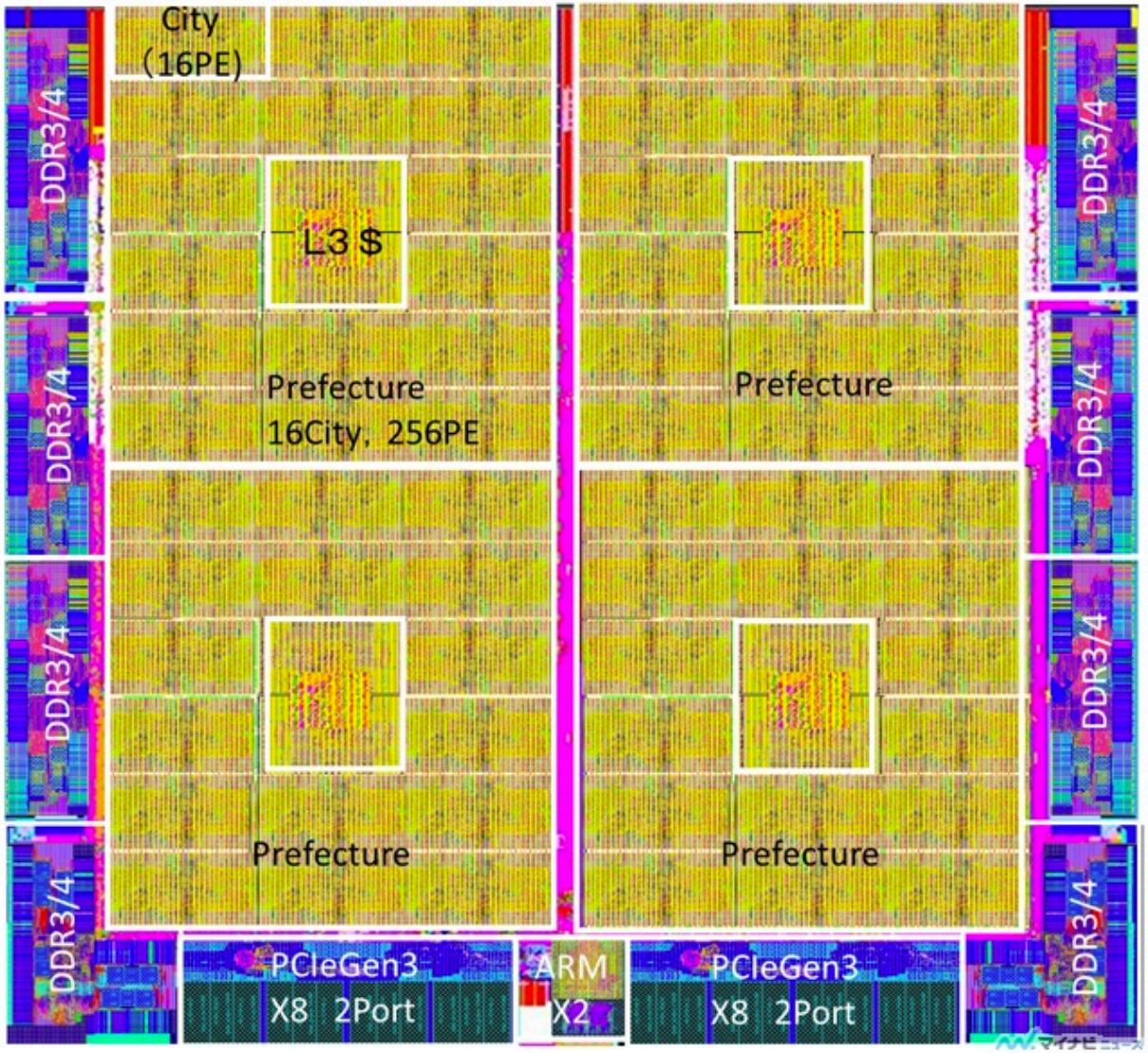
1. ハードウェアの紹介
    1. PEZY-SCチップのアーキテクチャ
    2. Suiren (2014/11)
    3. Suiren BlueとShoubu (2015/06)
2. HPLのチューニング
    1. LU分解の基本と牧野版コード
    2. PEZY-SCでのDGEMM
3. アプリケーション開発の現状紹介
    1. ファインマンループ積分（KEK石川+）
    2. 格子QCD（KEK松古+）
    3. 多倍長ライブラリ（会津大 中里+）

PEZY-SCチップ

- **「国産」1024コアMIMDプロセッサ**
  - TSMC 28nm HPM 21mm×19.6mm
  - 本当の本当に1024コアでMIMD
    - デコーダが1024個、PCが8192個（8-way SMT）
    - 固定長命令のload/storeアーキテクチャ
  - 733MHz駆動で単精度3.0Tflops、倍精度1.5Tflops (IEEE-754準拠)
- **OSが動くようなコアというよりはアクセラレータのような使い方が想定**
  - キャッシュコヒーレンシ無し、割り込み（？）、atomic命令は備える
  - GPUに似ているけど分岐を気にしなくていい、といった感じ

チップ画像

- 16PEでL2を共有するCity
- 16 City, 256PEでL3を共有するPrefecture
- 4 Prefecture, 1024PE/chip



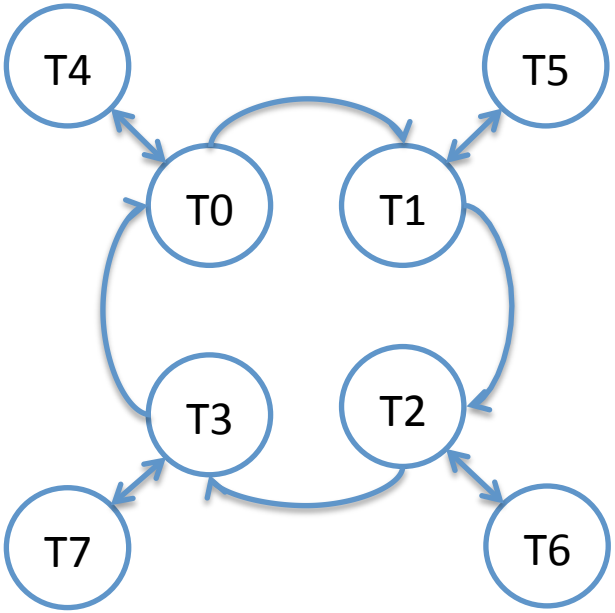
階層キャッシュ

- PE毎に16KiBのローカルメモリと1KiBのL1D（2PEで共有可）
- 16PEのCity毎に64KiBのL2D
- 16 CityのPrefecture毎に2MiBのL3D
- 同様に2K, 32K, 128Kの命令キャッシュ

- コヒーレンシ機構は無い
  - Sync+掃き出し命令で他スレッドのデータが反映される
    - 階層ごとにLevel 0〜Level 5の同期命令
  - 同じラインの別アドレスの書き込みは同期後コンシステント
  - 同一アドレス同一値の書き込みはOK、同一アドレス別値書き込みは不定
    - この辺はOpenMPやGPUと同じ

SMT (8 threads)

- 4つのスレッドがクロック単位でラウンドロビン
  - 4 cycle latencyの命令は直前の結果が待ちなしで使える
  - 命令スケジューリングの簡素化
- さらにそれぞれが「表」と「裏」のスレッドを持つ
  - 表裏の機能は対等
  - ロード命令発行時等に明示的な切り替え

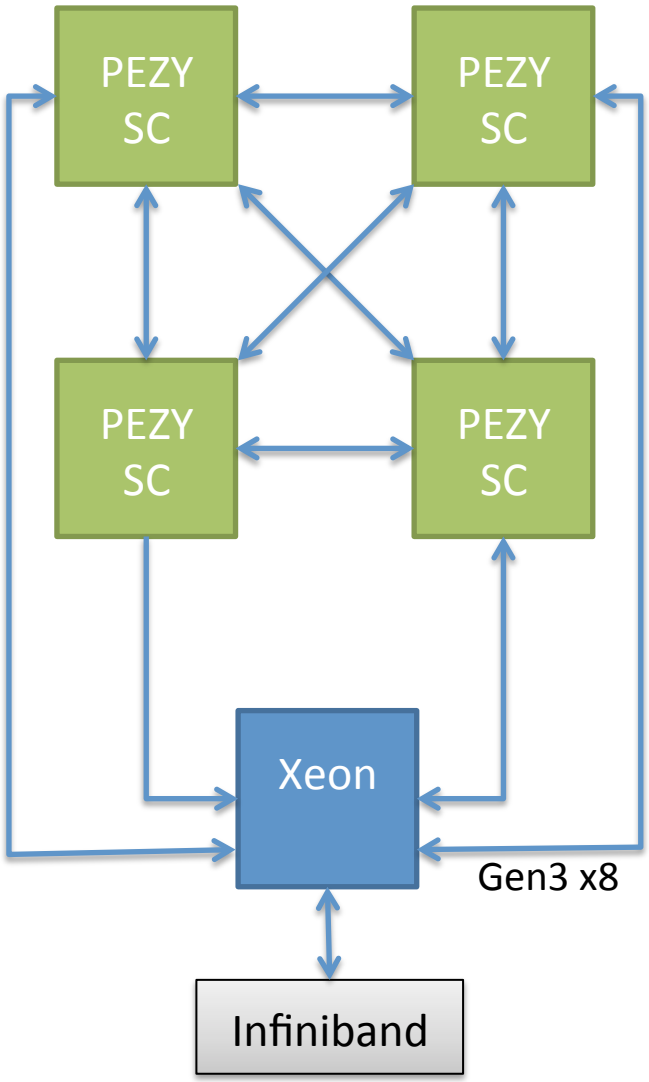


プログラミングモデル

- OpenCLのサブセットとしてPZCLを用意
  - LLVMベース
  - 各スレッドの挙動は普通のC言語＋一部組込み関数で
  - 自分のスレッド番号を見ながらSPMDで
- 感覚的にはGPUと殆ど同じ
  - ただし8192スレッドが全てMIMD動作
    - バラバラに分岐しても構わない
    - ただし命令キャッシュはそんなに大きくない

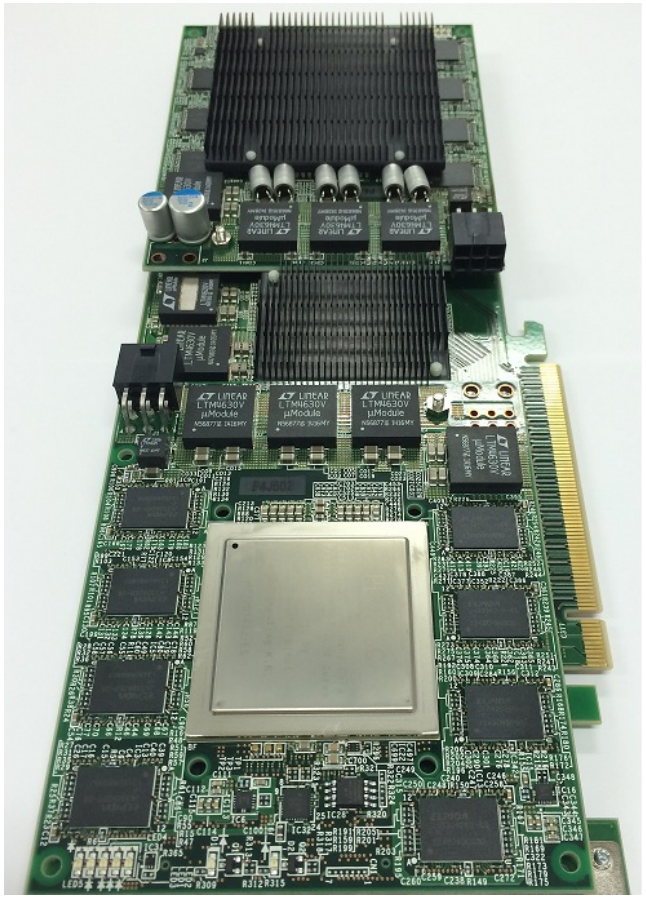
I/O

- PCI express Gen2/3 x8 が4 port
  - ただし2015/06のHPLはGen2接続
  - チップ間直接接続にも使える
- DDR3/4が8ch、合計512-bit
  - 「京」やFX10相当（DIMMではない）
  - GDDR5のGPUより帯域が弱いがその分容量は大きい（16/32 GiB）



ExaScaler-1.5 構想図

Suiren (node)



- PLXスイッチのベースボードを介して PEZY-SCモジュールを２つ搭載
- Gen3 x16をGen2 x16ふたつに分岐
- 各モジュールにDDR3が32GB

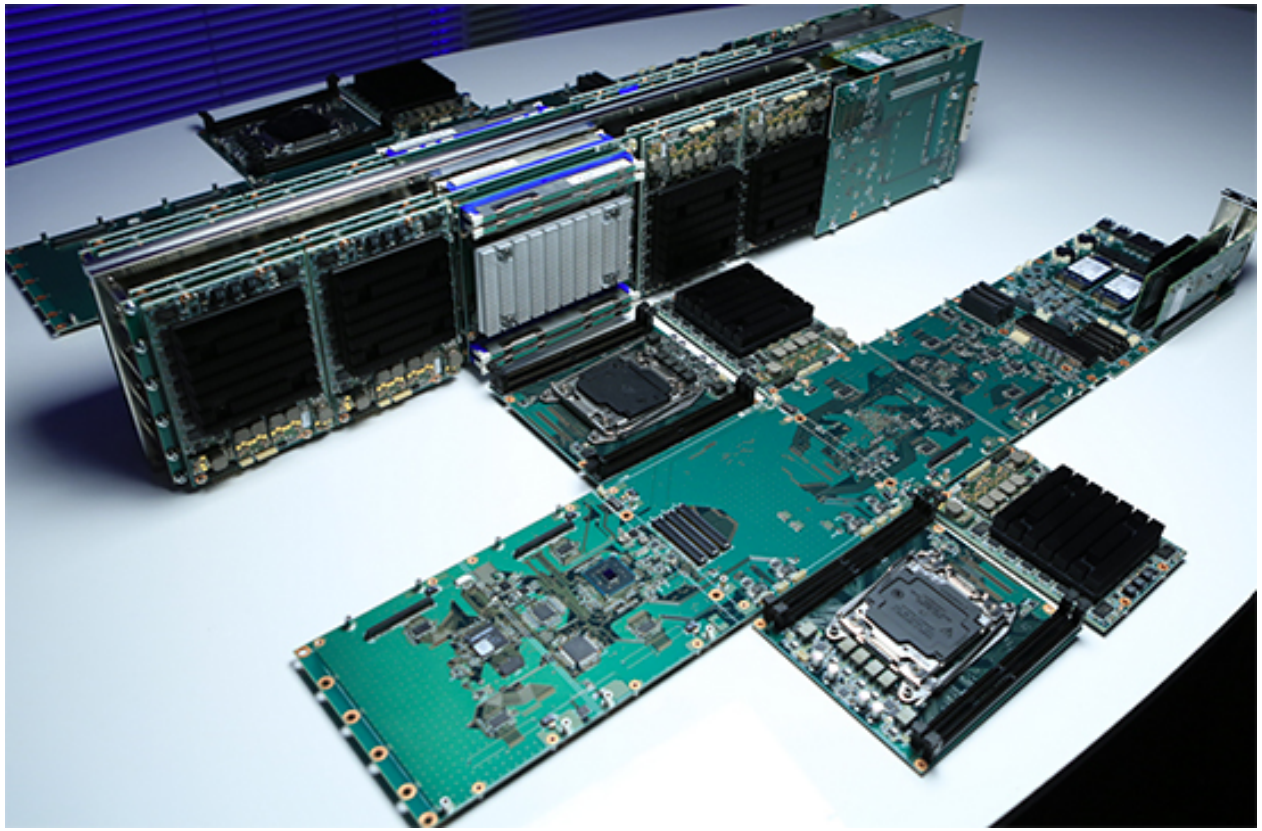
- Supermicroベアボーンの魔改造品
- Dual IvyBridgeに左のカードを4枚搭載
- 総メモリはHost/Device共に256GB

Suiren (system)

- 1U筐体を8つ、液浸タンクに沈める
  - 合計で4タンク、32ノードのシステム
  - IBスイッチはひとつ
- 冷却液はフロリナート（フッ化炭素）
  - 不活性、低揮発性
  - 液浸環境下でもコネクタの抜き差しができる
- 循環させたフロリナートを室外機で冷却

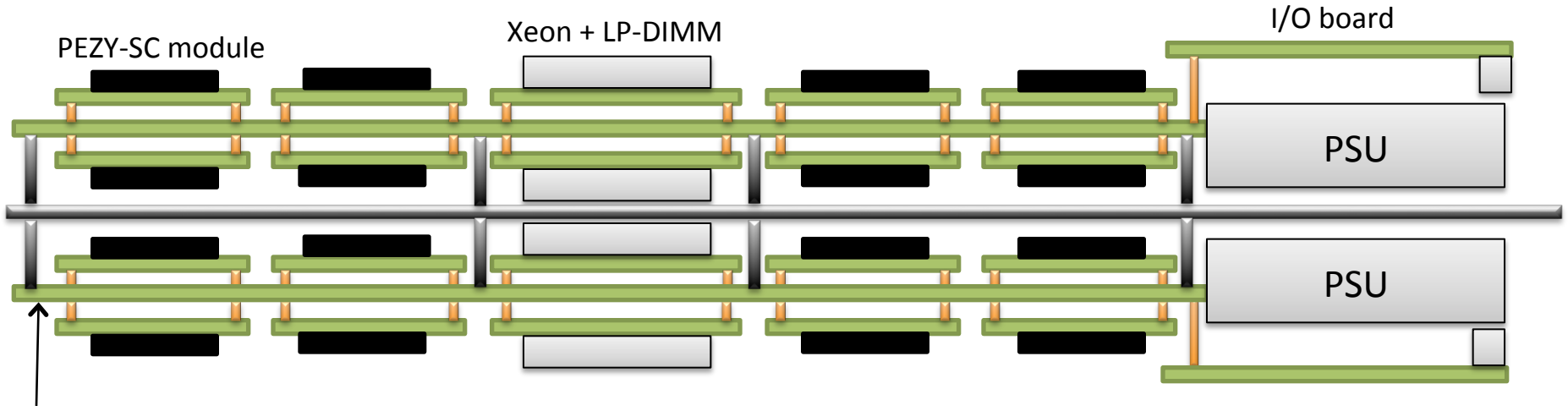
ExaScaler-1.4 (Suiren Blue / Shoubu)

- マザーボードからカスタム制作することで更なる高密度実装
  - フロリナートは大変高価なため
  - １「ブリック」に4 Xeon (Haswell 1 socket/node), 16 PEZY-SC
  - モジュールもDDR4へと改版、DC 1 2Vはネジ給電



ブリックの構造

- 真ん中に補強用のアルミ板
  - 剛性は確保されたが、内側の部品へのアクセスが悪い、、、
- キャリアボード側に、２ノード分のPCI express配線、PCH、BMCチップ等
- IOボード側に、２ノード分のIB-HCA、BMC-LAN、USB、mSATA
- PSUは2ノードで共有



※実際の寸法を再現するものではありません

RIKEN Shoubu （菖蒲）

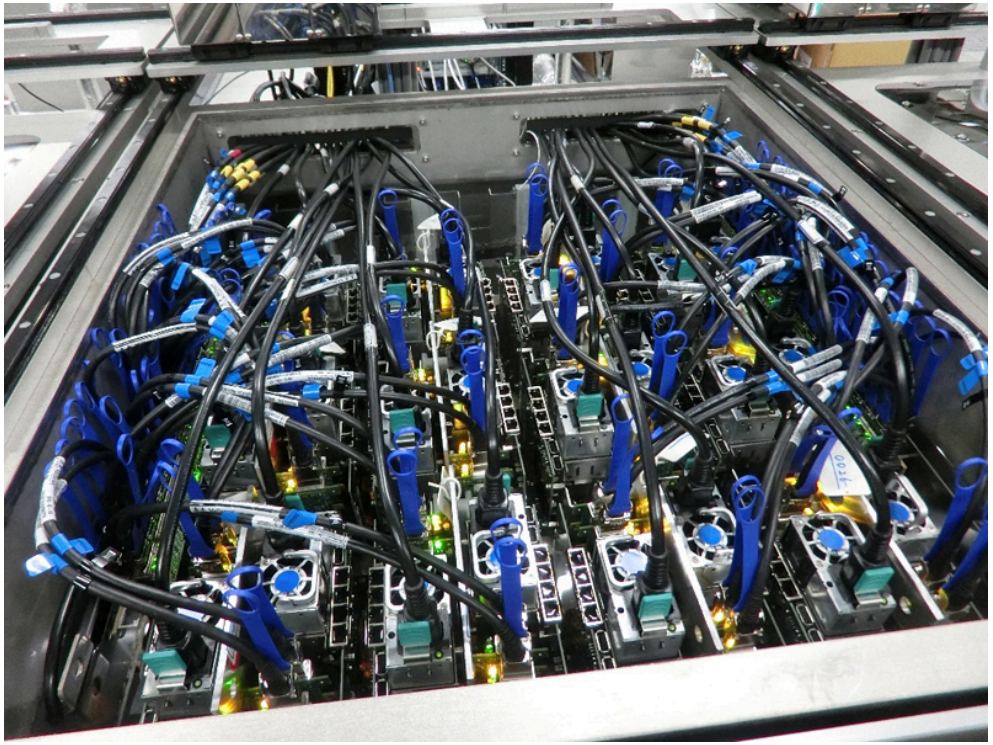
- 5液浸槽、最大80ブリック（320ノード）
  - 2015/06 Top500時にインストールできたのは60ブリック (12 bricks/tank)

FX100
1 PFLOPS

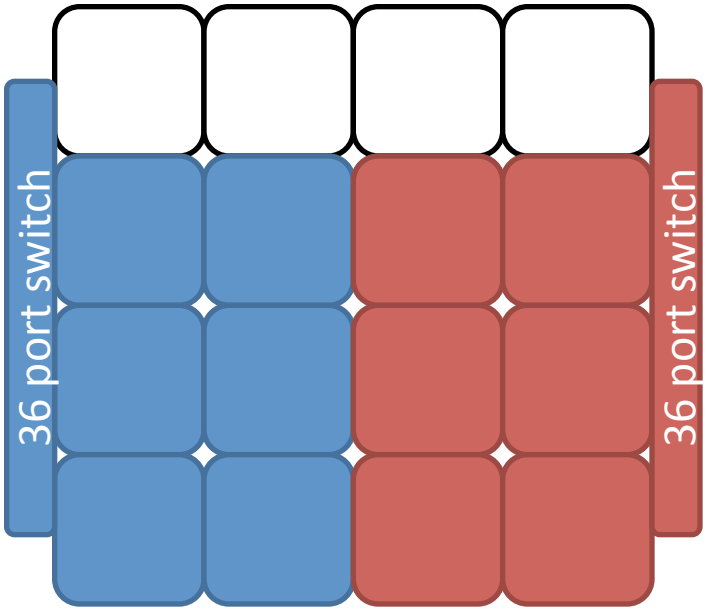
HIOKI
電力測定器

Shoubu network

- FDRで12:24のfat tree
- 36 portのスイッチを分解して液浸化



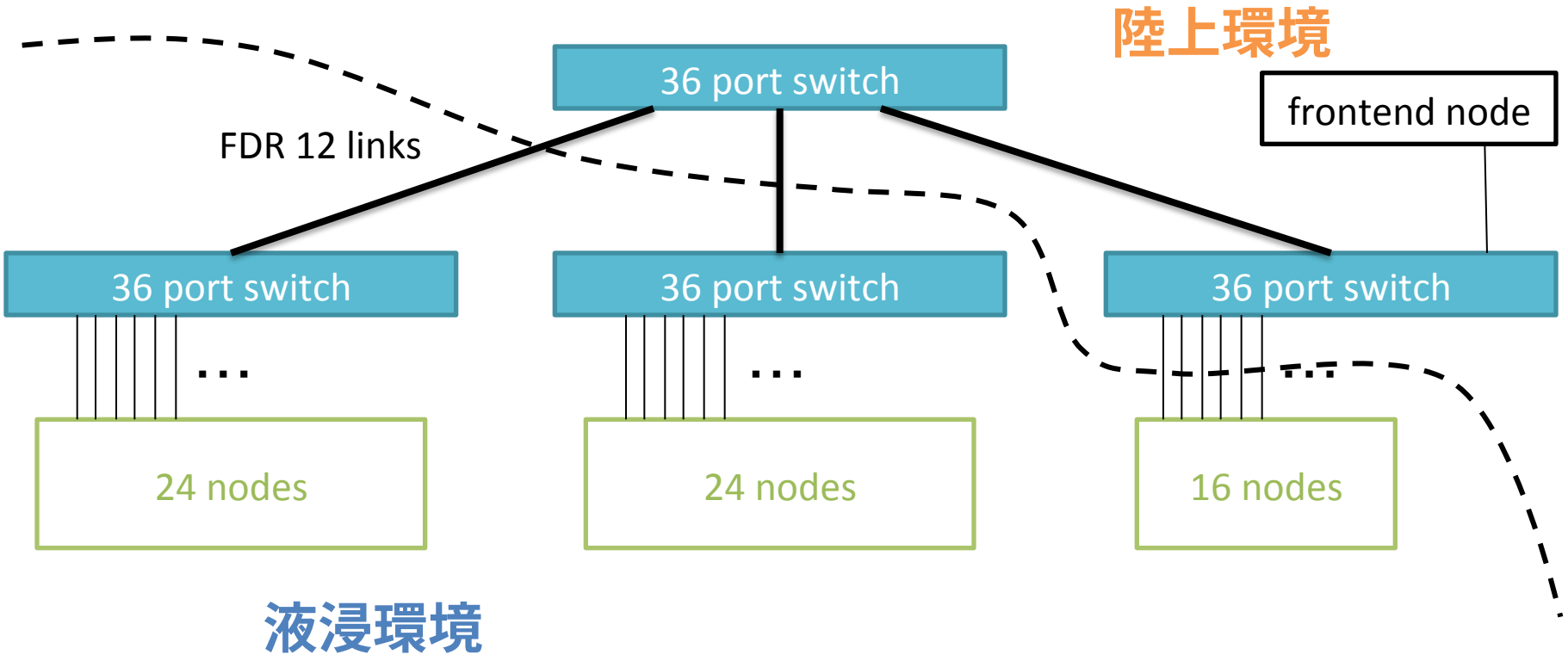
配線してみるとなかなかの混雑状況



右側のスイッチの下にもう
一台スイッチが沈んでいる

Suiren Blue（青睡蓮）fat tree

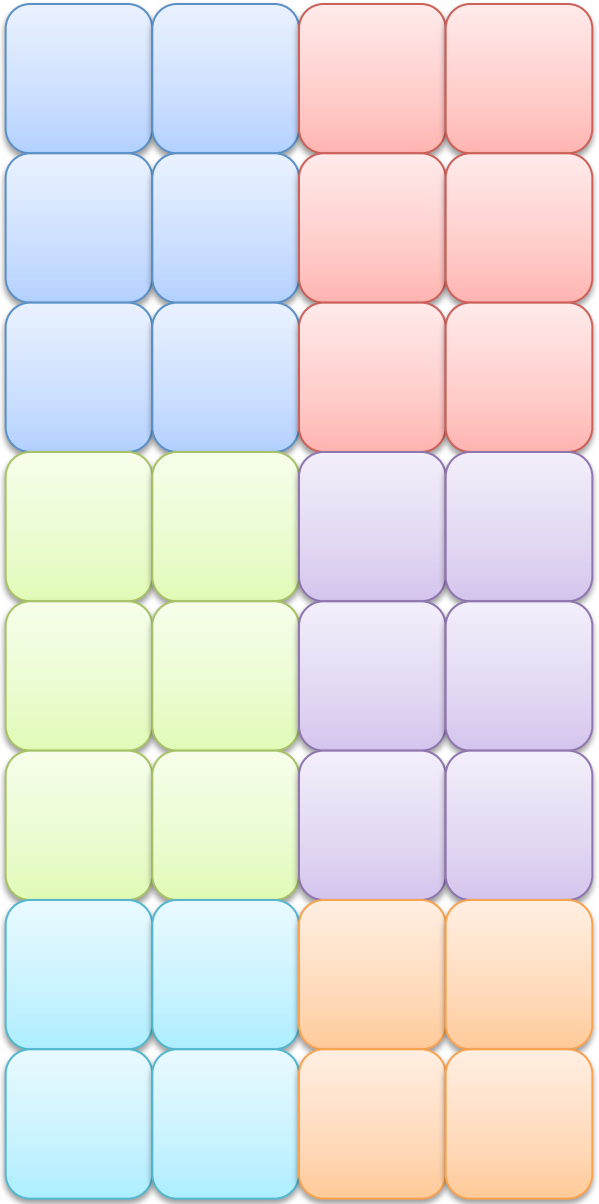
- こちらは1タンクフル実装の64ノード (24+24+16)
- ４タンクのSuirenと同じ256 PEZY-SC



HPLノード割り付け

- Shoubu 128 node 512 MPI proessの場合
  - ノード内の4プロセスは2x2でブロック化
  - スイッチ内24ノードは6x4でブロック化
    - 今思うと4:32のfat treeでもなんとかなったかも
  - ノードやスイッチをまたぐ通信をできるだけ減らす

＝1ブリック4ノード16プロセス



HPLコード

- 牧野先生がGRAPE-DR用に「書き直し」た“lu2”がベース
- 何で書き直しか？
  - 既存のHPLからDGEMMだけオフロードしても全然効率が出ない
  - 通信や行交換の時間が顕著に
  - アクセラレータの比重が大きなシステムでは必ず起こること

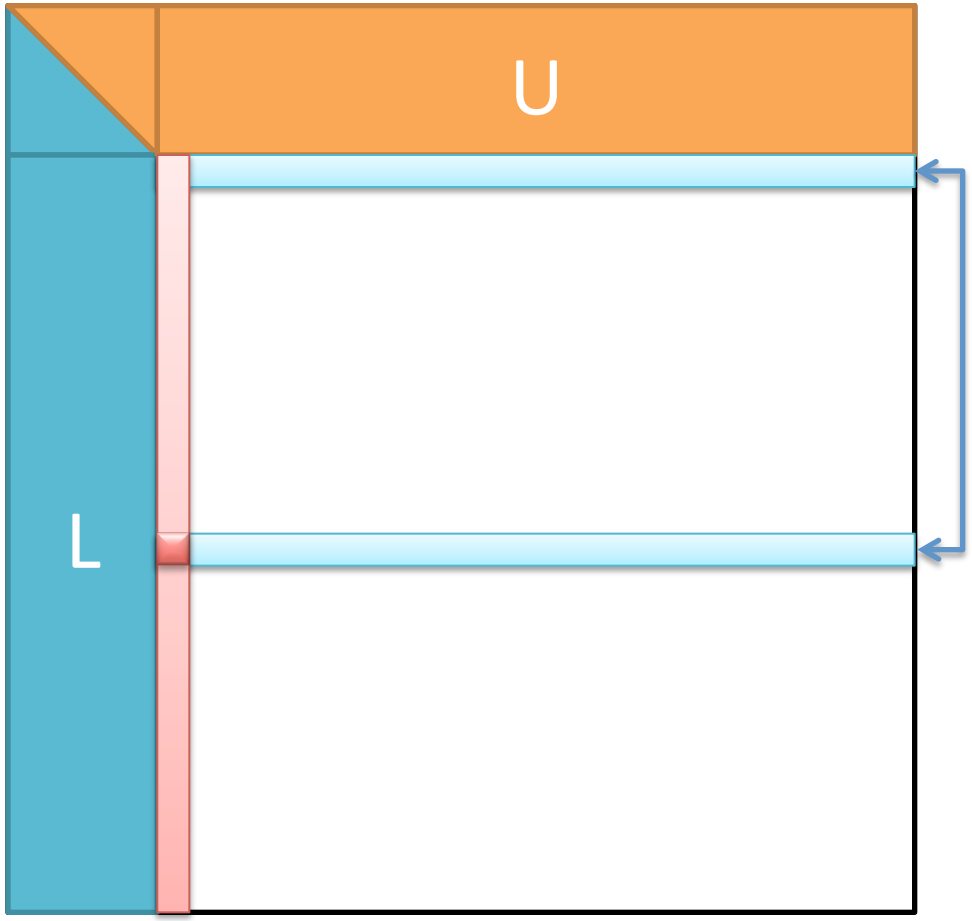
- 10. HPL 書き直し (2009/9/2 書きかけ)
- 11. HPL 書き直しその2 (2009/9/12 書きかけ)
- 12. HPL 書き直しその3 (2009/9/14 書きかけ)
- 13. eASIC と次世代 GRAPE(2009/9/14 書きかけ)
- 14. HPL 書き直しその4 (2009/9/16 書きかけ)
- 15. HPL 書き直しその5 (2009/9/16 書きかけ)
  - 15.1. 2009/12/8
  - 15.2. 2009/12/10
  - 15.3. 2009/12/30
  - 15.4. 2010/1/12
- 16. HPL 書き直しその5 (2009/10/3 書きかけ)
- 17. HPL 書き直しその6 (2010/1/13 書きかけ)
  - 17.1. 並列動作確認について
  - 17.2. 制御プロセッサの変更等
- 18. HPL 書き直しその7 (2011/7/21 書きかけ)
  - 18.1. 検討するべきこと
  - 18.2. SSD の速度
  - 18.3. 行列乗算の手順と速度

http://jun.artcompsci.org/articles/on_cs/face.html

lu2の特徴

- ブロックサイクリック分割、Gustavsonの再帰LU分解は踏襲
- 行列の格納方式をRow Majorに変更
  - 行交換が連続アクセスになり高速に
  - 再帰で幅が小さくなってきたら（縦長の行列になったら）一度転置
    - Pivot探索、柱LU分解のメモリアクセス効率向上
- 行交換と消去（DGEMM）をオーバーラップさせて隠蔽
- DTRSMも三角行列の逆行列とのDGEMMに

Pivot探索と行交換



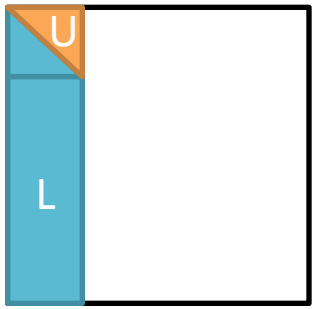
1. 列の中から最大値を探し

2. 行を交換

- 精度よく連立方程式を解くのに必要な操作
- 行列の次元数だけこの操作は発生する
  - 行交換はブロック化可能
- HPLが単に演算器を回すだけのベンチになっていないのはこれがあるから

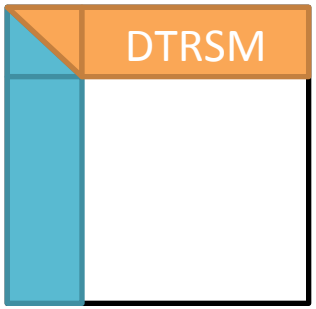
ブロック化LU分解

- 右側をpending出来るというのが基本的な考え方
  - 左側は右側に依存しないため
  - 左側の消去履歴にもとづき右側を処理

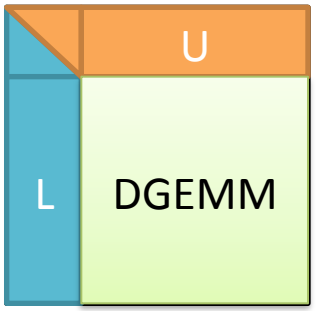


左の柱をLU分解

右側行交換



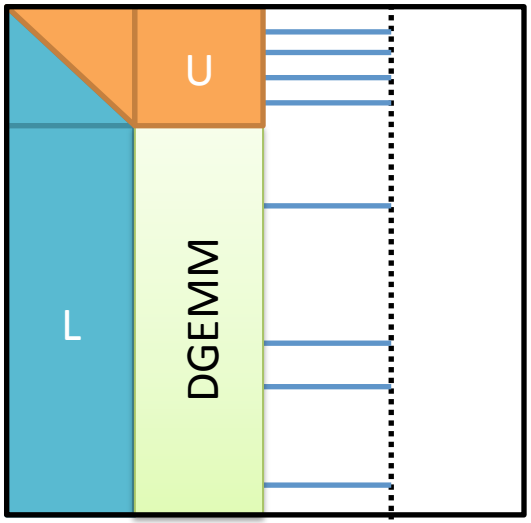
右上部処理

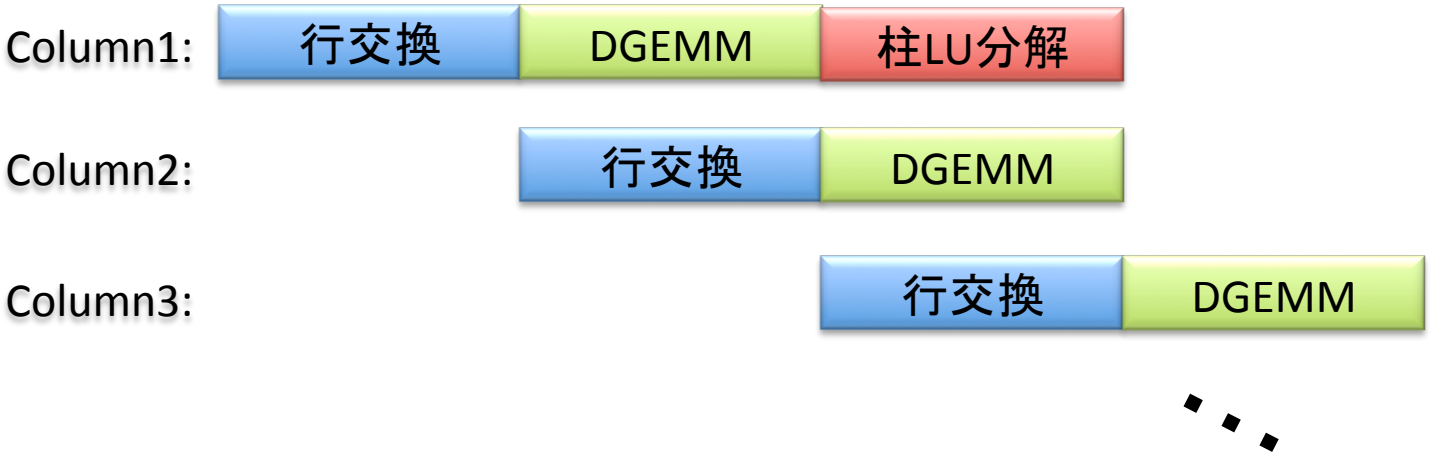


右下アップデート（一番美味しい）

lu2でのオーバーラップ

- 行交換と消去をパイプライン化
- 次の柱のLU分解と放送もなるだけ早く開始





雑談：Fujitsuアーキで同じことはできるか？

- アシスタントコアが行交換、16コアでひたすらDGEMM
  - 性能比1:16
  - メモリサイズ、ネットワークバンド幅には余裕がある
  - 通信はTofuのRDMAで

PEZY-SC向けの変更点
(私がやったわけじゃないけど)

- 2014/10 (for Suiren)
  - GRAPE-DRよりメモリが大きい(32GB/chip)ので、DGEMMの際L行列はメモリに置いたままにする
  - メインの行列そのものはホストに
- 2015/03 (Device memory version)
  - メインの行列もデバイスメモリに置いてしまう
  - Suirenで前回1位の電力効率を超える
- 2015/06 (Host-Device memory version)
  - 行列をホスト／デバイスで分割して保持
    - Shoubu/Suiren Blueでメモリが半減したのを補う
    - 16GB/chip (8GbitのDDR4が調達できなかったため)
    - 64GB/Xeon (LP-DIMM x4という制限のため)

Green 500のルール

（不明点も多いので話半分に）

- 同期のTop 500の500位に入らなければ足切り（Little list行き）
  - 2014/06から順に133.7T、153.4T、164.8T
    - 順調な鈍化傾向に助けられる
  - ただしTop500とは独立にsubmitできる
- 冷却の電力は含めなくてよい（らしい）
  - 立地条件よりも技術を競うため？
  - DCファンの不要な水冷、液浸システムが有利
- 20%ルール
  - 悪名高きルール、次項で説明

20%ルール

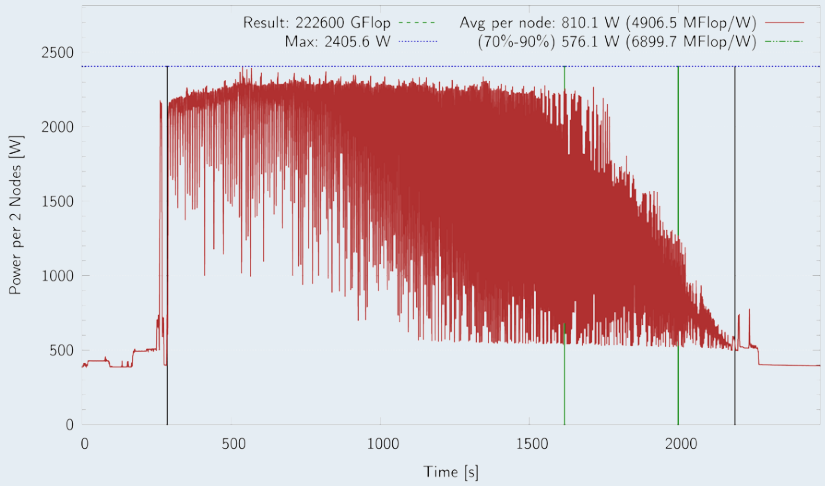
- 最初と最後の10%を除いて20%区間の電力平均を用いてもいい
  - 理由はよくわからない
  - 後半で性能と電力の落ち込むアクセラレータに有利

Playing with the rules

FIAS Frankfurt Institute for Advanced Studies

The Green500 rules state that the power measurement interval must at least cover 20% of the middle 80% of the core phase: For instance the period 70%-90%.

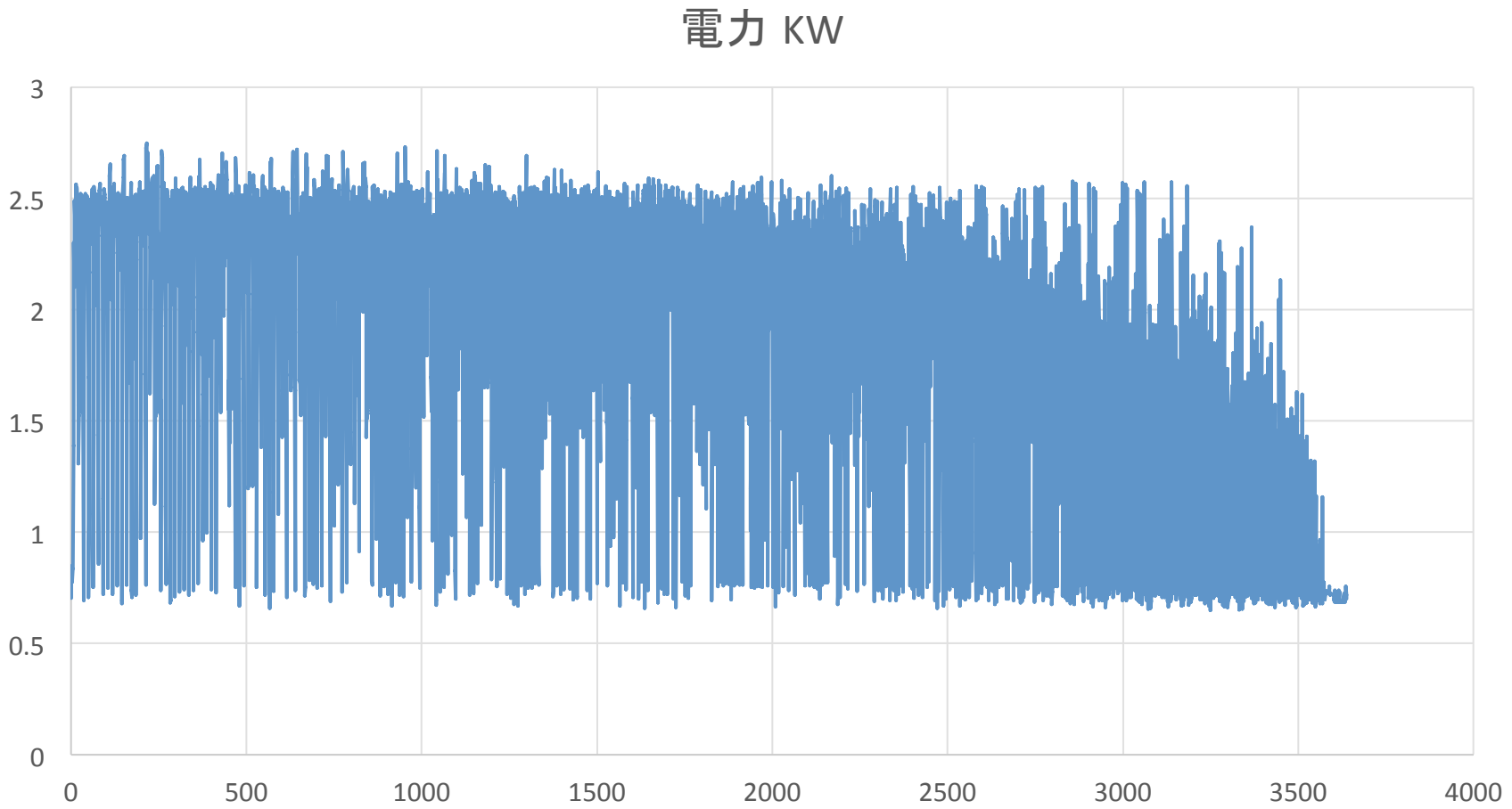
GFLOPS/W:	
Full measurement:	**5296**
70%-90%:	6010
Short Run:	4907
Short Run, 70%-90%:	**6900**



www.goethe-universitaet.de

2014/11で1位となった
L-GSCのBoF発表資料より

Shoubu電力測定値

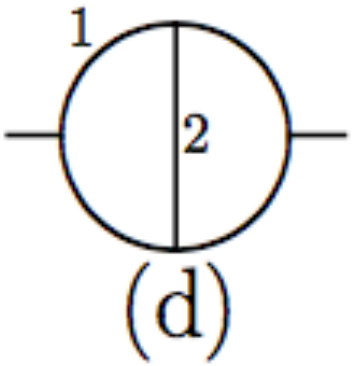


DGEMMの実装

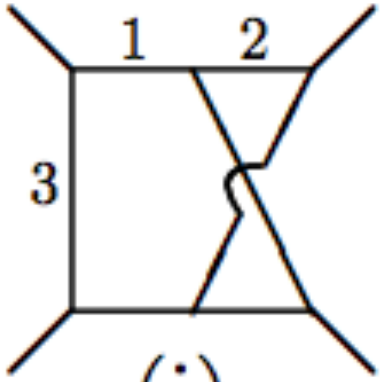
- ここでどこまでピークに近づくかが勝負
  - （私が書いたわけではないけど）
- 行列乗算：(N, K)×(K, M)→(N, M)　($C_{ij} = \Sigma_k A_{ik} B_{kj}$)
- PEあたり16KiBあるローカルメモリとキャッシュの階層性を利用
  - PE単位 (8 threads)
    - (4, 256)×(256, 128)→(4, 128)　赤字はローカルメモリ
  - City単位 (16 PEs)
    - (64, 256)×(256, 128)→(64, 128)　　[(16, 1)倍]
  - Prefeture単位 (16 Cities)
    - (256, 256)×(256, 512)→(256, 512)　[(4, 4)倍]
  - チップ全体 (4 Prefectures)
    - (256, 256)×(256, 2048)→(256, 2048)　　[(1, 4)倍]

ファインマン・ループ積分

素粒子物理学では、ファインマン図を使って、素粒子の崩壊や素粒子反応の散乱断面積を計算する。
高次の補正を計算する場合には、ループのあるファイマン図を計算する。
ループのあるファインマン図を計算するには、ループ積分を計算する必要がある。
PEZYにおいて、いくつかのファインマン・ループ積分を行った。



(d)



(i)

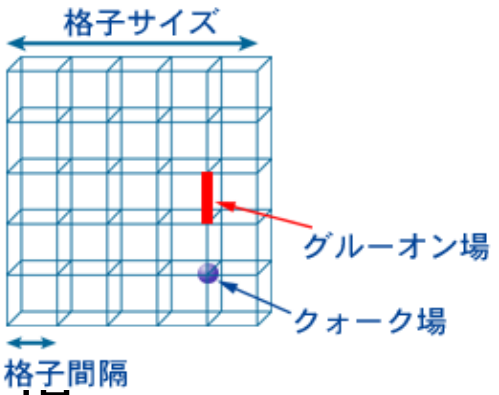
Lattice QCD on Pezy

- 開発メンバー
  - 石川健一 (広島大)
  - 青山龍美 (名古屋大KMI)
  - 松古栄夫 (KEK)
  - 鈴木朋浩 (Pezy)
- 2015.04 本格的な開発開始

Lattice QCD

- QCD: Quantum Chromodynamics
  - クォーク間に働く「強い相互作用」を記述
  - 「グルーオン場」が相互作用を媒介
- 格子QCD: 4次元格子上のQCD
  - クォーク場: 格子点上の場
  - グルーオン場: リンク上のSU(3)行列の場
  - 経路積分による量子化

  → Monte Carlo法によって数値計算が可能
  - 格子間隔ゼロの極限でQCDに帰着
  - QCDに基づいた一般的な計算が可能な唯一の方法



Lattice QCD

- Monte Carlo 法による高精度の評価のためには、疎行列の線形方程式を多数回解く必要
  - Monte Carlo 法でアンサンブルを作る際に必要
  - クォークの伝搬関数(グリーン関数=線形方程式の解)を用いて物理量を評価
  - 反復解法 (CG法など)＋前処理法
  - GPUで用いられた手法がPezyにも適用可

Implementation

- 1格子点を 1 thread に割り当てるのが標準的
- 疎行列: 隣接する格子点上のデータが必要
  - データ通信が律速: 階層的に処理
  - ノード間通信 (MPI)
  - CPU - device memory 間のデータ転送
  - Device global memory – core 間のデータ転送

Lattice QCD on accelerator

- Multi-device, multi-node： GPUと同じ手法を適用
  - 線型方程式の前処理にdeviceを使う
  - Mixed precision preconditioner

  倍精度ソルバーの前処理に単精度ソルバーを利用
  - Domain-decomposition preconditioner

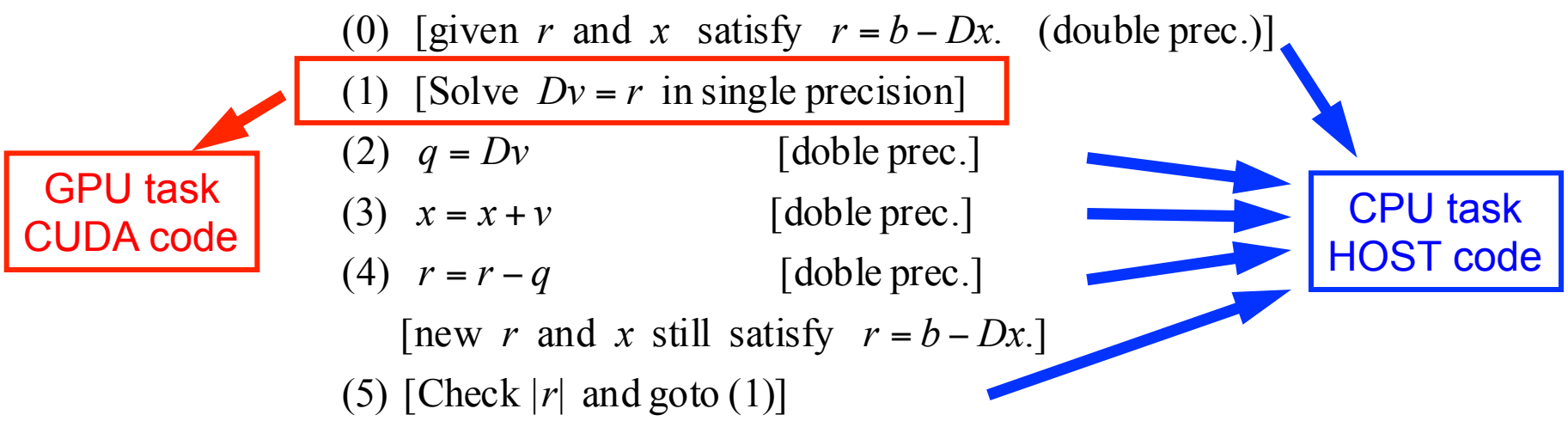
  領域分割によってデータ通信を局所化した前処理
  - Cf. 石川さんのスライド

Development status

(2015.07までの状況)

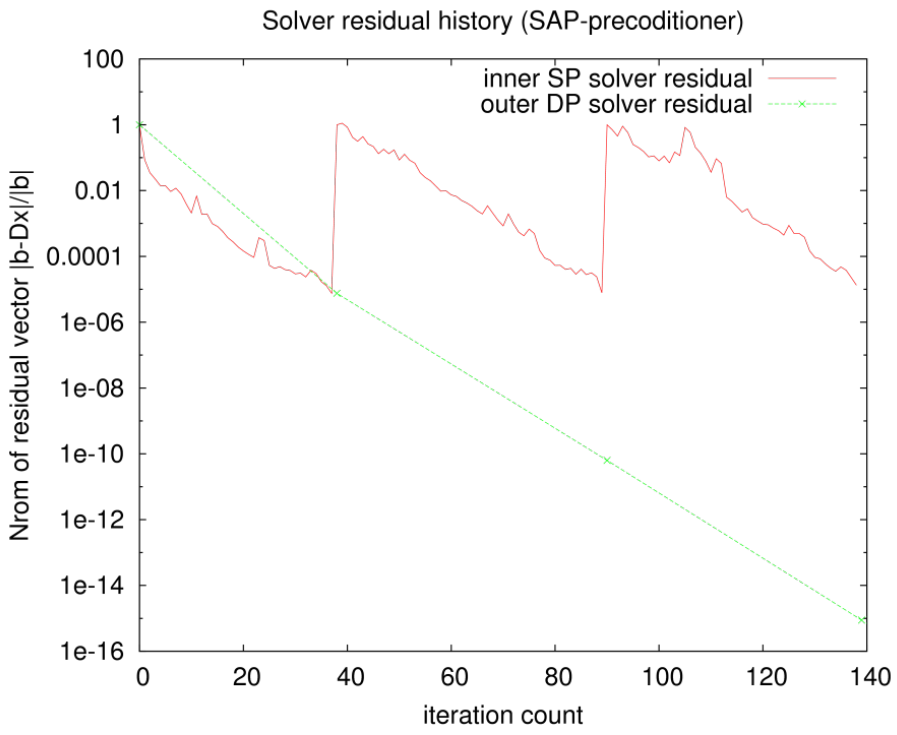
- 代表的な格子作用、アルゴリズムを移植中
- Mixed precision/domain-decomposition solver
  - 移植、動作確認
- Pezyのためのチューニング
  - 複数のコードを開発しながら経験をためている段階
  - 性能実測値、チューニングのノウハウをまとめ中
  - 実装上の要望等をPezy社にフィードバック

- GPU acceleration LQCD Kernel *D[U]*
  - Mixed precision solver (strategy) [early proposal by Buttari, Dongrra, Langou, Langou, Luszczek, Kurzak (2007)]
    - To solve *Dx =b*.



    - *The iterative refinement technique* with full single precision solver (10^-7) can solve full double precision (10^-14) solution within 3-5 refinement iterations.
    - Most computing time is spent in the S.P. solver.
    - GPU is employed for Full single precision solver.

Mixed precision solver residual history (example)



32^3x64 lattice

Outer solver : BiCGStab (Double precision)

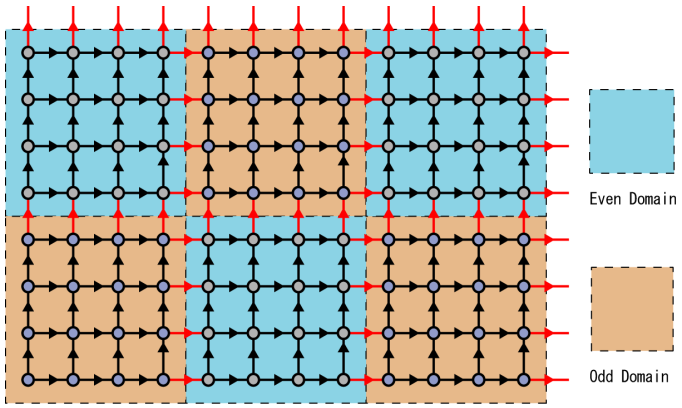
Inner solver : BiCGStab (Single precision)

Single Precision solver is called 3 times in the outer solver to obtain Double precision solution.

Most of the computational time is spent in the single precision solver => Accelerator(Pezy).

Domain decomposition Preconditioner =Schwartz Alternating Procedure (SAP)

$$x = M_{SAP} b \approx D^{-1} b$$



$$D_f x = b \Rightarrow \begin{pmatrix} D_{EE} & D_{EO} \\ D_{OE} & D_{OO} \end{pmatrix} \begin{pmatrix} x_E \\ x_O \end{pmatrix} = \begin{pmatrix} b_E \\ b_O \end{pmatrix}$$

$$\Rightarrow \begin{cases} D_{EE} x_E + D_{EO} x_O = b_E \\ D_{OO} x_O + D_{OE} x_E = b_O \end{cases} \Rightarrow \begin{cases} x_E = (D_{EE})^{-1}(b_E - D_{EO} x_O) \\ x_O = (D_{OO})^{-1}(b_O - D_{OE} x_E) \end{cases}$$

(0) $x_O^{(0)} = 0$

(1) for k = 0,1, $N_{SAP} - 1$

(2) $x_E^{(k+1)} = (D_{EE})^{-1}(b_E - D_{EO} x_O^{(k)})$

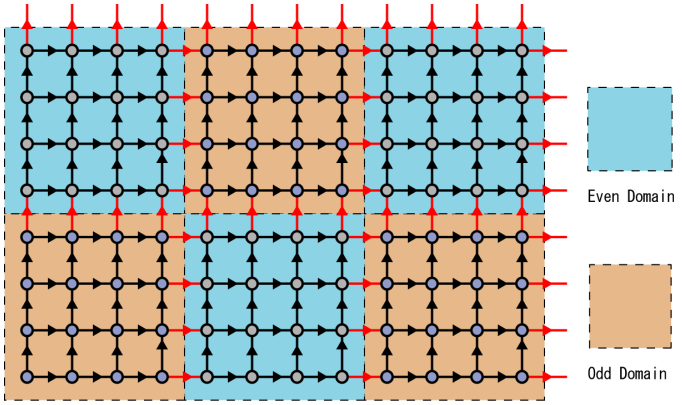
(3) $x_O^{(k+1)} = (D_{OO})^{-1}(b_O - D_{OE} x_E^{(k+1)})$

(4) endfor

Alternatingly solve the equation in each domain = Two-color Block Jacobi Iteration

Domain decomposition Preconditioner

Schwartz Alternating Procedure (SAP)



$$D_f x = b \Rightarrow \begin{pmatrix} D_{EE} & D_{EO} \\ D_{OE} & D_{OO} \end{pmatrix} \begin{pmatrix} x_E \\ x_O \end{pmatrix} = \begin{pmatrix} b_E \\ b_O \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} 1 & (D_{EE})^{-1} D_{EOo} \\ (D_{OO})^{-1} D_{OE} & 1 \end{pmatrix} \begin{pmatrix} x_E \\ x_O \end{pmatrix} = \begin{pmatrix} (D_{EE})^{-1} b_E \\ (D_{OO})^{-1} b_O \end{pmatrix}$$

$$\Rightarrow \left[1 - (D_{EE})^{-1} D_{EO} (D_{OO})^{-1} D_{OE}\right] x_E = \left[(D_{EE})^{-1} b_E - (D_{EE})^{-1} D_{EO} (D_{OO})^{-1} b_O\right]$$

$$x_O = -(D_{OO})^{-1} D_{OE} x_E + (D_{OO})^{-1} b_O$$

$\hat{D}_{EE} x_E = \hat{b}_E$: Equation after the Domain-Decomposed Even/Odd preconditioing

Current sustained performance (very preliminary)

- lattice size: $16^{\wedge 3} \times 32$
- H/W: Suiren at KEK, single node, single device
  - no data transfer between CPU and device included
- Wilson fermion matrix multiplication:
  - double precision: 54.8 GFlops
  - single precision: 107.6 GFlops
- ただしコードチューニング及びシステム構成の最適化により 性能向上の可能性有り

Suirenでの多倍長精度FP演算

- OpenCLでの多倍長浮動小数点演算実装を移植
  - 第148回HPC研究会で報告済み (2015-HPC-148(8))
- OpenCL版をPZCLに移植する際の手間
  - 大きな修正は必要なく動作する
- OpenCL版をPZCLに移植する際の手間
  - PZCLはC++インターフェイスをサポートしていない
  - カーネル生成はオフラインのみ
  - カーネル名に規約あり (“pzc_XXXX”とする必要あり)
  - カーネル用のAPIの名称が異なる
  - カーネル内で最後に”flush()”を呼ぶ必要がある

PEZY-SCでの性能評価

- MYFP(Int方式)の性能評価 (MFLOPS)
  - 8倍精度、STREAM的なc[i] = a[i] op b[i]

	加算	乗算	除算1	除算2	除算3	4 OPS
PZCL	144	130	49.3	68.4	60.1	432
R280X	729	1825	22	47	241	461
W9100	942	525	28.6	52.0	28.1	597
W8100	980	450	28.1	52.2	47.1	610
E5-2670	274	189	16.7	35.4	19.2	254

まとめ（雑感）

- MIMDメニーコアで同じ28nmのGPUを電力効率で上回ったのは本当に驚きでした
  - 2007/11のリストを見るとBlueGene/Pが首位、その後Cellなのでこれまで全て何らかのSIMD
- アプリについて
  - OpenCLからの移植はすんなりとできるという印象です
  - MIMDの活きるアプリがもっと出てくるのを期待します
    - 分岐やテーブル参照のある数学関数を多用するものなど