学際大規模情報基盤共同利用・共同研究拠点　萌芽型共同研究 採択課題
EX17306（東京大学情報基盤センター推薦課題）
9th Symposium
Joint Usage / Research Center for Interdisciplinary Large-scale Information Infrastructures

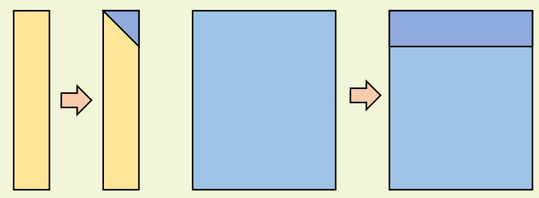
高柳　雅俊（山梨大学　総合研究部）

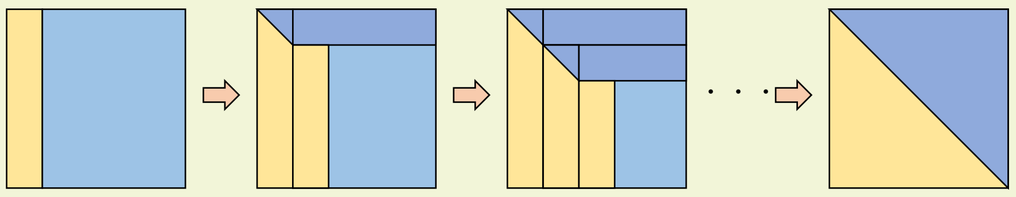
ハイブリッドクラスタシステムにおける通信削減QR分解実装

目的：高速QR分解ルーチンのGPUクラスタ実装

CPU-GPU混在環境向け実装

- **ブロックアルゴリズム** → パネル分解+後続行列更新
- パネル分解：逐次性が強く、memory-bound → CPU
- 後続行列更新：並列性高く、compute intensive → GPU
- 行列データをGPU (device) メモリ上に保存
- 非常に高速な実装が得られている (MAGMA)





GPUクラスタ

- GPUを搭載したクラスタシステムの登場 → 今後の主流
  - GPUは電力性能比が高い
- しかし、GPUクラスタ用の数値線形代数ライブラリは得られていない
  - ScaLAPACK ← マルチコアCPUクラスタ向け
- GPUメモリはホストメモリと比べ小さい→大規模な行列に対応できない
- GPU間は高速な通信路が用意されているが、ノード間の通信が遅い

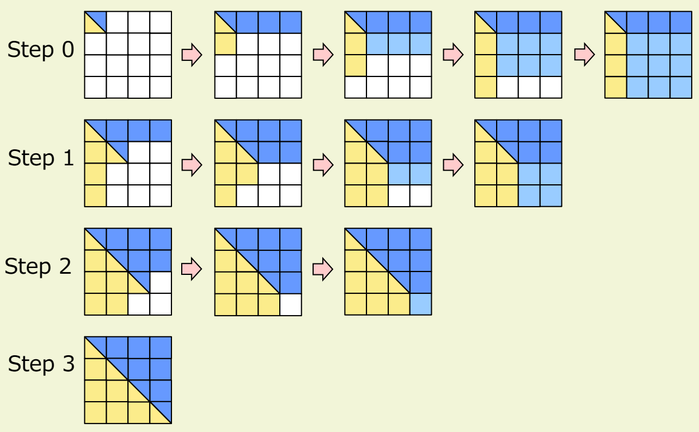
November 2016 The Green500

Green 500 Rank	MFLOPS	Total Power	Computer	Site
1	9462.1	349.5	NVIDIA DGX-1, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, NVIDIA Tesla P100	NVIDIA Corporation
2	7453.5	1312	Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100	Swiss National Supercomputing Centre (CSCS)
3	6673.8	150.0	ZettaScaler-1.6, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband FDR, PEZY-SCnp	Advanced Center for Computing and Communication, RIKEN
4	6051.3	15371	Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway	National Supercomputing Center in Wuxi
5	5806.3	77	PRIMERGY CX1640 M1, Intel Xeon Phi 7210 64C 1.3GHz, Intel Omni-Path	Fujitsu Technology Solutions GmbH

これまでの研究成果

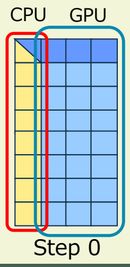
タイルアルゴリズムの導入

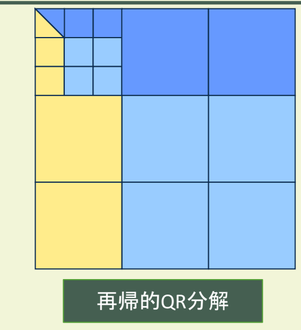
- 行列を小行列 (タイル)に分割、タイルごとにタスクを実行
- ホストメモリ上に行列データを保持
  → 大規模行列に対応　GPUメモリ使用量削減
- OpenMP 4.0 task構文depend節による動的スケジューリング実装



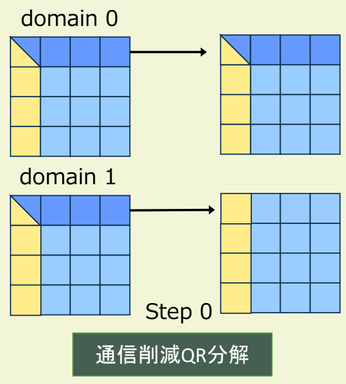
GPUクラスタシステム実装

- 再帰的QR分解
  - CPU・GPUで最適なタイルサイズが異なる
    CPU：タイルサイズ小 ⇔ GPU：タイルサイズ大
  - タイルをさらに小行列に分割し、それぞれに最適化
- 通信削減QR分解
  - 縦方向のタスクは逐次処理のためボトルネック
  - タイルをドメインに分割しQR分解を1Step実行
  - その後、上三角部分を組み合わせ一番に集約させる
- GPU向け実装
  - 分解タスク→CPU、更新タスク→GPU





再帰的QR分解



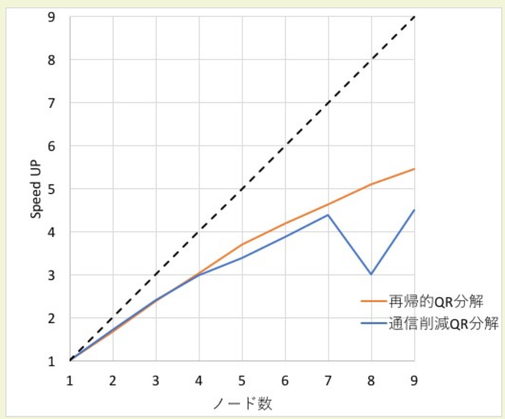
通信削減QR分解

これまでの研究成果つづき

速度測定結果

- TSUBAME 2.5で性能測定
  - 1ノードあたりの行列サイズ40960x40960
  - Weak Scalingを測定

- 測定結果
  - 5ノードまでは良くスケールしている
  - ノード数が増加すると効率が落ちる
    →9ノードでは理論性能の約60%



TSUBAME 2.5での速度測定

今後の研究計画

GPUクラスタ向けの実装

- 性能低下の原因調査
  - プログラムのトレース取得
  - CPUタスクが少ないことが原因の１つ
  - 行列データ分散
    - 現在：1-Dサイクリックデータ分散（縦方向，横方向）
    - 今後：2-Dサイクリックデータ分散（ScaLAPACK方式）
- Look-aheadによるCPUタスクの割り当て増加