富岳向けQCDソルバー

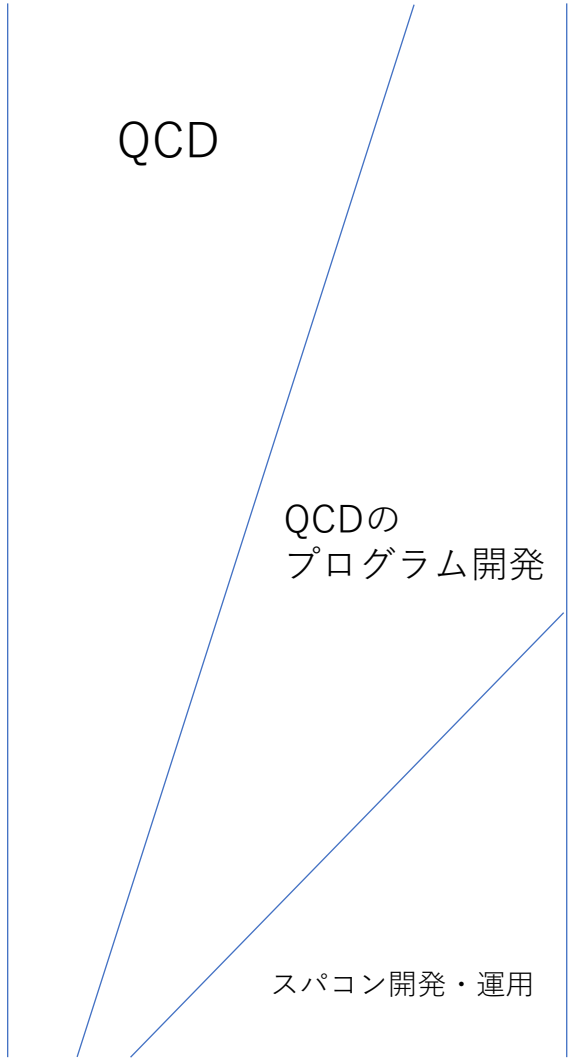
中村宜文
理化学研究所
第15回 HPC-Phys 勉強会

内容

- FS2020のLQCDのコデザイン（特にQCDのソルバー）について
  - 富岳コデザイン・レポート（2022年3月公開）
    - https://www.r-ccs.riken.jp/wp/wp-content/uploads/2022/03/fs2020-report-j.pdf
    - https://www.r-ccs.riken.jp/wp/wp-content/uploads/2022/03/fs2020-report-e.pdf
  - 102 PFLOPS Lattice QCD quark solver on Fugaku
    - https://arxiv.org/abs/2109.10687
  - 開発されたコード
    - https://github.com/RIKEN-LQCD/qws

自己紹介

- ドイツ電子シンクロトロン研究所（2005/11–2008/10）
- ドイツ・レーゲンスブルク大学（2008/11–2010/09）
  - IBMと共同で格子QCD専用計算機「QPACE」を開発。
    - 私にとっての初めてのコデザイン（協調設計）
- 筑波大学計算科学研究センター（2010/10-2011/02）
- 理化学研究所（2011/03 -

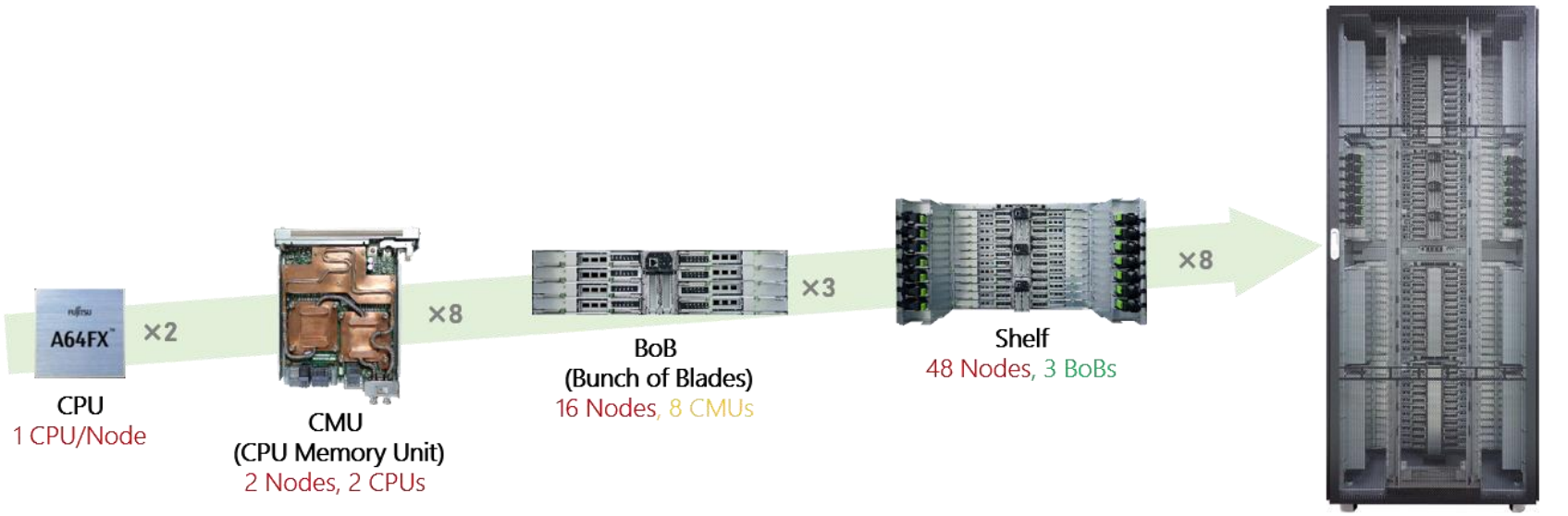


第12回 (HPC-Phys) 勉強会とほぼ同じ
https://hpc-phys.kek.jp/workshop/workshop210826/nakamura_210826.pdf

富岳

対「京」比
倍精度演算：約50倍
主記憶容量：約5倍
主記憶帯域：約30倍
ネット帯域：約4倍
（全インジェクション帯域）

- 理化学研究所と富士通株式会社の共同開発
- 158,976 ノード（A64FX CPU）
- 432 ラック
  - 396（ノード全搭載:384ノード）ラック
  - 36（ノード半搭載:192ノード）ラック
- 演算性能
  - ノーマルモード（2.0 GHz）
    - 倍精度 488 PFLOPS
    - 単精度 977 PFLOPS
    - 半精度 1.95 EFLOPS
    - 整数 3.90 EOPS
  - ブーストモード（2.2 GHz）
    - 倍精度 537 PFLOPS
    - 単精度 1.07 EFLOPS
    - 半精度 2.15 EFLOPS
    - 整数 4.30 EOPS
- 主記憶 : 4.85 PiB, 163 PB/s
- ストレージ
  - 第一階層：1.6TB高速SSD/16ノード
  - 第二階層：富士通FEFS(Lustreベースファイルシステム, 約150PB)
  - 第三階層：商用クラウド等



Courtesy of FUJITSU LIMITED

第12回 (HPC-Phys) 勉強会とほぼ同じ
https://hpc-phys.kek.jp/workshop/workshop210826/nakamura_210826.pdf

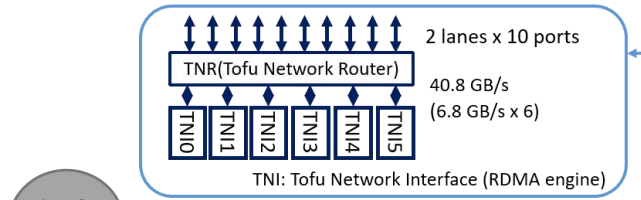
CPU

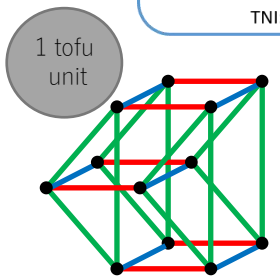
Architecture	**Armv8.2-A SVE (512-bit SIMD)** **Fujitsu extension : hardware barrier, sector cache, prefetch**
Core	48 (+ 2 or 4 assistant cores) 4 core memory group (4 CMG)
TFLOPS	2.0 GHz) DP: 3.072, SP: 6.144, HP: 12.288 2.2 GHz) DP: 3.3792, SP: 6.7584, HP: 13.5168
L1 cache (at 2GHz)	L1D/core: 64 KiB, 4way 256 GB/s (load), 128 GB/s (store) B/F=4
L2 cache (at 2GHz)	L2/CMG: 8 MiB, 16way L2/node: 4 TB/s (load), 2 TB/s (store) L2/core: 128 GB/s (load), 64 GB/s (store) B/F=2
Memory	HBM2 32 GiB, 1024 GB/s B/F=0.3
Interconnect	Tofu Interconnect D (28 Gbps x 2 lane x 10 port) 6 Tofu network interface (RDMA engine) 40.8 GB/s (6.8 GB/s x 6)
I/O	PCIe Gen3 x16
Technology	7nm FinFET

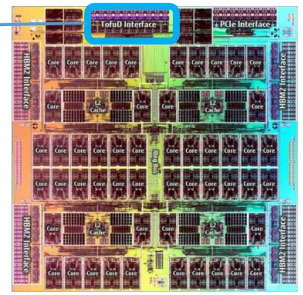
ネットワーク

- Tofu インターコネクト D
- トポロジー (X,Y,Z,a,b,c) : (24,23,24,2,3,2)
- Tofuハードウェアバリア機能での高速縮約
  - 3 浮動小数点, 8 整数　まで
- リンク帯域：6.8GB/s
- インジェクション帯域：40.8 GB/s
- 6基のRDMAエンジンで同時通信可能
- キャッシュインジェクション機能（FX100,Tofu2から）
  - 受信データを直接L2にも書き込む機能
  - L2ミスを減らせる可能性がある

LQCDとの
コデザインの成果







Courtesy of Fujitsu

開発経緯概略

- **ロードマップ策定**
  - 2020年代に実現されるべきアプリケーションとそれを可能とするスパコンの将来像をまとめた以下の白書を策定。当初、草の根的に始まった
    - 計算科学研究ロードマップ白書
    - HPCI技術ロードマップ白書
- **調査研究**
  - 東北大、筑波大、東大の3代表機関が企業と共にそれぞれスパコン基本方式とその性能見積もり等の調査研究を行った
  - 理研はアプリケーション開発者と共に将来の社会的・科学的課題とその解決のためのスパコンで動かすアプリケーションをまとめた
- **基本設計**
  - コンピュータハードウェアおよびソフトウェア、設置条件などの仕様（開発すべき項目等）を決めた
- **詳細設計**
  - 基本設計で確定した仕様を元に、具体的なハードウェア・ソフトウェア設計を行い試作機を制作し評価した
- **製造・設置・調整**
  - 富岳ハードウェアを製造・設置し、ハードウェアおよびソフトウェアの安定化および正式運用に向けた調整作業を行った

「スーパーコンピュータ「富岳」の開発経緯」（石川 裕）から抜粋
https://www.ssken.gr.jp/MAINSITE/event/2020/20210121-sci/lecture-01/20210121_sci_ishikawa.pdf

次世代計算基盤に係る調査研究事業：ポスト「富岳」調査研究
契約締結・事業開始　令和4年7月中旬頃～8月（予定）
https://www.mext.go.jp/b_menu/boshu/detail/mext_00217.html

ポスト「京」調査研究は
今からちょうど10年前

FS2020につながる活動

- アプリFS
  - 文部科学省「将来のHPCIシステムのあり方の調査研究」の「アプリケーション分野からみた将来のHPCI システムのあり方の調査研究」
  - 期間：2012年7月から2014年3月
  - 代表機関：理化学研究所　計算科学研究機構（当時、AICS）
  - 代表：富田浩文
  - 5年から10年後に想定されるアプリケーションのシステム要求値の集計、要求値の妥当性検証、「計算科学ロードマップ」査読、QCDアプリ性能調査を含め理研側で物理分野全般を担当
- 計算科学研究機構提案フラグシップマシンとFSアプリのマッチングWG(AICS内)
  - 2013年度
  - 各分野研究から提出されたアプリのシステム要求値の集計・精査・分析、システム要求と「計算科学ロードマップ」整合性チェックを担当
  - AICS提案フラグシップマシン上で格子QCDを実行する際のハードウェア諸元と汎用部・加速部の比率に関する検討を担当

フラッグシップ2020プロジェクト（FS2020）初期とQCD

- 当初エクサスケールコンピューティング開発プロジェクトという名称で開始（2014年4月）
  - 事務書類では、2015年10月27日まではエクサスケールコンピューティング開発、2016年1月8日以降はフラッグシップ2020となっていた
- 2014年夏秋ごろ総合科学技術・イノベーション会議(CSTI)で想定アプリCCS-QCDが、192^4の問題サイズでポスト京において京の50倍の性能を目指と設定
    - （総合科学技術・イノベーション会議評価専門調査会第２回評価検討会（平成26年10月28日）の資料）
  - 中村はこの目標性能の推定＆設定にはかかわっておらず後に知らされる・・・実に残念
- 2014年10月から富士通と基本設計を開始
- 2014年11月：他の重点アプリに先駆けて格子QCDが諸般先行することになった
    - QCDは机上性能見積もりが比較的容易、他の重点アプリは準備が整っていない
      - ポスト京重点課題が決定されたのは12月末ごろ
    - 11月下旬ごろに富士通と初めて打ち合わせ
  - この時点での困難
    - CCS-QCDは京での実績がなく、性能比較には京用、ポスト京用両方の最適化が必要だった
    - 京で実績のあるLDDHMCは京用に、組み込み関数でのSIMD化、8threads化されており、京/FX10でしかまともに動かない（8threadsだけ、2SIMDだけ使うとかでは動く）。LDDHMCに対してポスト京想定のSIMD幅8/16用に再最適化を短期間で行うのは困難だった
    - ターゲットアプリの性能予測を通してCPU等の仕様を決めないけない（コデザイン）
    - 基本設計終了まであまり時間がない
    - 伝えられる仕様情報があまりなく机上性能見積が手探り（主記憶、L2$、ネットワークバンド幅ネックのルーフラインモデルでの見積ぐらいしかできない）

ターゲットアプリ：LQCD

- 格子量子色力学（Lattice quantum chromodynamics：LQCD）計算シミュレーション用プログラムは公開プログラムがたくさんある
  - MILC（アメリカ）
  - CPS++（アメリカ）
  - Chroma（アメリカ、たぶんユーザーが一番多い）
  - BQCD（ドイツ、中村が2005年ぐらいから開発メンテしている）
  - tmLQCD（ドイツ）
  - CCS-QCD（筑波大、カーネル部分がminiappとして公開）
  - Bridge++（新学術「素核宇」->戦略分野５で開発。その後重点課題を経て有志でメンテ中）
  - Iroiro++（KEKで開発）
  - LDDHMC（京・FX10用コード。カーネル部分は理研筑波の共同研究で共用開始前に限界まで最適化、**非公開**）
  - カーネル部分と逆行列解法はいくつかあるが基本的にどれも同じ問題（Dirac方程式）を解いている
- LQCD用カーネルライブラリ
  - Bagel（もとはBluegene（qcdoc）用アセンブリ生成器）
  - Quda（GPU用のコード生成。 NVIDAが開発メンテ）
- ポスト「京」重点課題（９）「宇宙の基本法則と進化の解明」のターゲットアプリ：LQCD
  - 特定のプログラムにこだわらず、ポスト「京」で高速にDirac方程式の求解ができるものをつくる
  - （どれでもないほうがいろいろ良かったり・・・）ということで　LQCD　と呼ぶことにした。

基本設計の報告書執筆時点ではCCS-QCDではなくLQCDとしていたが、一部未修正の資料からの派生のため、FS初期の文科省や理研の資料ではターゲットアプリ名称がCCS-QCDとなっているものもある。

CCS-QCD

- fortran90
- Wilson 型
  - 離散化されたフェルミオン作用の一つ
  - 他にもstaggered型、domain wall型（5次元型）等がある。
- Array of Structure (AoS) -> Structure of array (SoA)
  （FX100用最適化の過程で。2014年12月頃）
- double/float
- SIMD演算：単精度8SIMD （FX100用最適化の過程で）
- Thread数: 16
- Solver：BiCGStab (even-odd前処理されたDirac行列の逆行列計算)
- 最適化：単精度mult_deo_tzyxの性能がFX100で24%（単精度ピーク比）

LDDHMC

- 主にFortran90（カーネル部分はSIMD組み込み関数をcで記述）
- Wilson 型
- Array of Structure (AoS)
- double/float (カーネル部分は単精度)
- SIMD演算：倍精度2SIMD（主記憶では単精度でデータを格納、load時に倍精度に変換）
- Thread数: 8
- Solver：BiCGStab (fullのDirac行列の逆行列計算)
  - even-oddされたDirac行列の逆行列計算にくらべメモリが必要
  - ただし領域分割法前処理が使える
- 前処理:領域分割法
  - 演算の割合増加、通信の割合減少。より高い実行効率を得られる
  - 反復回数も減少
- 最適化：京・FX10用に限界まで。BiCGStabの実行効率は約30%
  - https://arxiv.org/abs/1210.7398

FS2020でのLQCDの（厳しい）問題設定

- 目標：富岳全系使って192^4の問題サイズで対京比50倍
- 富岳の演算性能は、京と比較して（約50倍：倍精度演算）向上しているが、通信性能はそれほど向上していない（約4倍）。
  - 一般的に演算性能の向上と比較してバンド幅性能の向上は遅い。
  - データの移動はできるだけ少ないアルゴリズムがよい
- 192^4の問題サイズ
  - 京ベンチマーク：192^4の問題サイズは、京で実行可能、むしろ効率が出しやすい最適といってよいサイズ。ベンチマークコードのLDDHMCは、京用に非常に最適化されていた。
  - 富岳ベンチマーク：192^4の問題サイズは、富岳全系を使うにしては、問題サイズが小さすぎて、通信の待ち時間が支配的になり非効率になる。
    - プロジェクト初期からそうなるというのはわかっていた。実際そうなった。
    - 最終結果の大規模ベンチマークでは通信関係の時間が半分を占める。

QCDソルバー

- クリロフ部分空間法（反復法）で$Ax = b$を解く
- 前処理
  - 前処理行列M使って解きやすくする
    - $M^{-1}Ax = M^{-1}b$
    - $AM^{-1}(Mx) = b$
  - red-black (even-odd)
  - Schwarz Alternating Procedure (SAP)
  - 精度混合
    - 単精度ソルバー（BiCGStabなど）を倍精度ソルバーの前処理として使う
    - 単精度ソルバーの前処理で半精度の前処理を混ぜる

M^{-1} は、$M^{-1} \approx A^{-1}$で簡単に計算できるものがよい

反復法

- 固定反復法：$A^{-1} = \sum_{i=0}^{\infty}(1-A)^i$
  - ノイマン級数展開　（$\|A\| < 1$のとき収束）
- $A = D + L + U$と分解しておく(2x2ブロック行列)
  - $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, D = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}, L = \begin{pmatrix} 0 & 0 \\ A_{21} & 0 \end{pmatrix}, U = \begin{pmatrix} 0 & A_{12} \\ 0 & 0 \end{pmatrix}$
- $k+1$反復目の解ベクトルx^{k+1}
  - ブロック・ヤコビ法：$Dx^{k+1} = b - (L+U)x^k$
    - $A_{11}x_1^{k+1} = b_1 - A_{12}x_2^k$
    - $A_{22}x_2^{k+1} = b_2 - A_{21}x_1^k$
  - ブロック・ガウス・ザイデル法：$(D+L)x^{k+1} = b - Ux^k$
    - $A_{11}x_1^{k+1} = b_1 - A_{12}x_2^k$
    - $A_{21}x_1^{k+1} + A_{22}x_2^{k+1} = b_2$

Even-oddとSAP係数行列の比較（どちらもBiCGStab）

対「京」比
倍精度演算：約50倍
主記憶容量：約5倍
主記憶帯域：約30倍
ネット帯域：約4倍
（全インジェクション帯域）

- $(I - D_{eo}D_{oe})x_e = b_e$
  - 行列ベクトル積
    - ステンシル計算
    - 隣接通信
  - ベクトルーベクトル演算
  - リダクション（2要素）
  - ベクトルーベクトル演算
  - リダクション（1要素）
  - 行列ベクトル積
    - ステンシル計算
    - 隣接通信
  - ベクトルーベクトル演算
  - リダクション（3要素）
  - ベクトルーベクトル演算
  - リダクション（3要素）
- 一反復当たり演算数：少
- 通信量/演算量：多
- 求解までの反復回数：多
- 実行効率：低
- 求解までの時間：長

- $DM^{-1}y = b \quad \rightarrow \quad x = M^{-1}y$
  - 行列ベクトル積
  - 複数回
    - ステンシル計算 x 複数回
    - 隣接通信
  - ベクトルーベクトル演算
  - リダクション（2要素）
  - ベクトルーベクトル演算
  - リダクション（1要素）
  - 行列ベクトル積
  - 複数回
    - ステンシル計算 x 複数回
    - 隣接通信
  - ベクトルーベクトル演算
  - リダクション（3要素）
  - ベクトルーベクトル演算
  - リダクション（3要素）
- 一反復当たり演算数：多
- 通信量/演算量：少
- 求解までの反復回数：少
- 実行効率：高
- 求解までの時間：短（短くならない場合もある）

高効率演算の増加

通信時間の割合を減少
- 隠蔽用演算が増加するわけではない
- 行列ベクトル積の時間が増加
- 相対的にリダクションの時間が減る

Schwarz alternating procedure (SAP)

- additive Schwarz
  - オーバーラップがない場合は block Jacobi
- multiplicative Schwarz
  - オーバーラップがない場合は block Gauss-Seidel
  - LüscherのSAP
    - JHEP 0305, 052 (2003)
  - FS2020のQCDソルバーもこれ

$\|DK\| < 1$のとき収束

$$K^{-1}D^{-1} = (DK)^{-1} = \sum_{i=0}^{\infty}(1-DK)^i$$

$$D^{-1} = K\sum_{i=0}^{\infty}(1-DK)^i$$

ガウスザイデルの $(D+L)^{-1}$

$$K = \begin{pmatrix} D_{EE}^{-1} & 0 \\ -D_{OO}^{-1}D_{OE}D_{EE}^{-1} & D_{OO}^{-1} \end{pmatrix}$$

固定反復のパラメータ２つ

$$M^{-1} = K\sum_{i=0}^{N_{SAP}}(1-DK)^i$$

$D_{EE}^{-1} \sim \sum_{i=0}^{N_m}(1-D_{EE})^i$, $D_{OO}^{-1} \sim \sum_{i=0}^{N_m}(1-D_{OO})^i$,
これらはプロセス間通信なし

LQCDコード開発

- QCD Wide SIMD Library (QWS)
  - 「格子量子色力学シミュレーションのための富岳及び広いシングルインストラクション・マルチプルデータ幅の計算機システム向け計算ライブラリ」
  - C, C++で記述
  - BSD License (publicly available at 31/Mar/2020)
  - https://github.com/RIKEN-LQCD/qws
  - 富岳において高い性能
  - 開発
    - 2014年に中村が、ポスト「京」で高い実行効率を得るため新しいデータレイアウトを使い開発開始、最適化、評価
    - 2015年に向井氏（富士通、ポスト「京」共同開発実施者）参加、最適化、評価
    - 2018年に石川氏（広島大、ポスト「京」重点課題実施機関）参加、半精度加速最適化
    - 2019年に金森氏（理研、ポスト「京」重点課題実施機関）参加、通信区間最適化
  - 富岳において高い性能を得るため、複数の格子QCDアプリケーションにて、直接的、間接的、もしくは、最適化のお手本的に利用されることを期待
    - LDDHMC (developed by Ishikawa (Hiroshima), used on K)
    - Grid (developed mainly by P. Boyle (Brookhaven))
    - Bridge++ (developed mainly by Matsufuru (KEK))
    - BQCD (maintained mainly by Nakamura(RIKEN))
    - …
    - …
    - …

問題規模が小さく半精度32SIMDが利用できなかったので大規模ベンチマークテストでの利用はなし

- 大規模ベンチマークテストの性能に直結
- プロセス当たり問題サイズが小さすぎて通信比率が高い
- utofu利用サンプルプログラム（by 金森さん、石川さん）
  - https://github.com/RIKEN-LQCD/jacobi2d

富岳向け最適化

- 律速箇所の同定（fipp, fapp等）
  - わかっている場合は必要ないが、最適化効果確認のためfipp, fappは使う
- SIMD化（京の128bitから4倍の512bitになった）
  - 実部虚部分離
- 関数のインライン展開
- ループ分割/融合
- 不必要なメモリコピーの除去
- データのプリフェッチ
  - LQCDでは手動でソフトウェアプリフェッチを用いてプリフェッチ指示
- OpenMPのパラレルリージョンを拡大しオーバーヘッド削減
  - “omp parallel for/do”のように並列リージョン指示文と処理分散指示文を結合したものを小さいループで繰り返さない。並列リージョンはできるだけ広くして、処理分散指示文と適宜同期指示文を入れる
- データ局所化による狭いメモリ帯域の回避
- プロセスマッピング
  - プロセス間通信の輻輳を減らし、受信待ち時間を短縮
- uTofu（低レベル通信API）による通信の高速化
  - TofuリンクとTofu Network Interfaceへの通信データの負荷分散

演算レイテンシ
実数系FMA：9
複素数系FMA：15(fcmadd)/16(fcmla)

BQCDプログラムのローカルな関数（共役運動量でゲージ場を更新する関数）が実虚分離SIMD化、インライン展開、ループ分割で約6倍加速

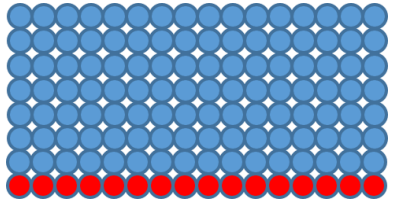
https://www.hpci-office.jp/pages/e_meeting_A64FX_201223#topic-1

QCDの富岳向けデータレイアウト

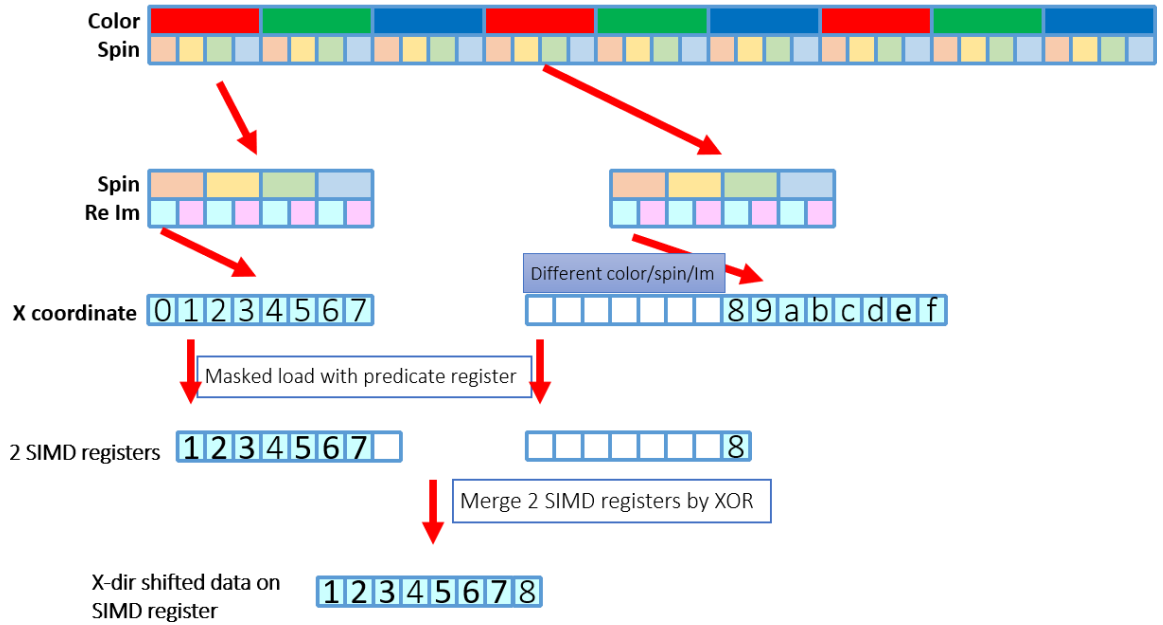
- 富岳の性能を引き出すには広いSIMD幅（512 bits,京の場合128 bits）に適したデータレイアウトが必要
  - 富岳：一つの命令で倍精度８要素の積和演算ができる。短精度なら16要素。半精度なら32要素
  - 京では一つの命令で倍精度２要素の積和演算ができる。複素数1要素の積も効率よくできる
  - QCDではこのような多要素をならべて同時に計算できるのは時空間の成分
    - FS2020プロジェクト当初、複素数演算や、データのシャッフルがどれほどサポートされるか不明で、９つのターゲットアプリで複素数演算主体アプリはLQCDの1つだけで、ターゲットアプリからは複素数演算強化に強い要請がかからないため、「京」の時のように複素数をそのまま並べる配列を採用するのはリスクがあった。
    - 複素数の実部と虚部は分離して、x方向の実部16要素を最初に並べる、手戻りがない単純なレイアウトを採用した。結果としてこれで良かった。

富岳演算レイテンシ
実数系FMA：9
複素数系FMA：15(fcmadd)/16(fcmla)

SIMD化



- LQCD：複素数の4次元ステンシル計算（時空間で一つ隣の点のデータが必要）
  - X方向差分計算以外、連続アクセスになりSIMD化率100%になるデータレイアウト
    - 富岳（倍精度）：[nt][nz][ny][nx/8][3][4][2][8]
    - 富岳（単精度）：[nt][nz][ny][nx/16][3][4][2][16]
    - 富岳（半精度）：[nt][nz][ny][nx/32][3][4][2][32]
    - 比較「京」：[nt][nz][ny][nx][3][4][2]
  - X方向はArm C Language Extensions (ACLE) で最適化



演算レイテンシ
実数系FMA：9
複素数系FMA：15(fcmadd)/16(fcmla)

FS2020でのLQCD達成倍率

LQCD参考性能
Sierra@LLNL < ~20 PFLOP (2018GB finalist)

- LQCDの富岳での性能：102 PFLOPS、38倍(対京比)
  - 京ベースライン、30.65 ms、実行効率は31.8%（対倍精度ピーク比）
    - データを単精度で保持し、倍精度演算を行う。データ転送量を半減し、バンド幅、ロードストアパイプライン負荷を減じる最適化を行っている。
    - 経過時間内訳
      - 演算区間：27.99 ms
      - 通信区間：2.66 ms
        - 隣接通信：0.95 ms
        - Allreduce：1.70 ms
  - 富岳実測、0.800 ms、実行効率は10.25%（対単精度ピーク比）
    - 経過時間内訳（本計測とは別。時間計測オーバーヘッドが若干ある。通信律速である。）
      - 演算区間：0.400 ms
      - 通信区間：0.407 ms
        - 隣接通信：0.345 ms（送信呼び出し:0.029 ms、受信待ち:0.254 ms、送信完了処理:0.062 ms）
        - Allreduce：0.062 ms (1要素x1:0.015ms, 2要素x1:0.016ms, 3要素x2:0.031ms)

ベースラインコードは既に単精度化されていた
アルゴリズム変更でオリジナルコードより実行効率UP（26% -> 31.8%）

単体性能

LQCD参考性能
Sierra@LLNL < ~20 PFLOP (2018GB finalist)

- 反復解法ソルバー500反復（性能倍率評価区間、区間名：all）
- 反復解法ソルバー500反復内部の領域分割内行列ベクトル積（区間名：in）

プロセスあたり格子サイズ	区間：all				区間：in			
	経過時間［ミリ秒］	浮動小数点演算			経過時間［ミリ秒］	浮動小数点演算		
		TFLOPS	ピーク比	演算数		TFLOPS	ピーク比	演算数
32x6x4x3	0.334867	0.8272	12.24%	69254421000	0.068043	1.1208	16.58%	19065600000
32x6x4x6	0.515010	1.0839	16.04%	139560981000	0.119455	1.3170	19.49%	39329280000
32x6x8x6	1.145754	0.9786	14.48%	280304661000	0.219403	1.4693	21.74%	80593920000
32x6x8x12	2.606202	0.8616	12.75%	561369621000	0.559297	1.1699	17.31%	163584000000
32x12x8x12	5.778703	0.7773	11.50%	1122981141000	1.192644	1.1146	16.49%	332328960000

- TFLOPSは1ノードの性能、演算数はCMGあたり、ピーク比は単精度浮動小数点演算ピーク比
- 2.2GHz、エコ無効、1ノード2プロセスで計測
- 演算数は理論的に数え上げて算出

富岳でのターゲット問題の性能：102 PFLOPS

（利用ノード数ピーク比:10.25%、全系ピーク比9.50%）

- 102.119799 PFLOPS （Allreduce演算数除外）
  - 1反復当たり演算数： 69254421000/500　（1プロセス当たり）
  - 1反復当たり経過時間：0.0008秒
  - プロセス数：589824
- 102.119806 PFLOPS （Allreduce演算数込み）
  - 1反復当たりAllreduce演算数：(1+2+3+3)*589824

まとめと反省

- 富岳向けQCDソルバーの紹介をしました。主に開発について

- 反省
  - FS2020の反省会をしていない
  - 半精度がまだちゃんとつかえていない

- ポスト「富岳」(10年後ぐらい)にむけて
  - QCDを大規模単一アプリにすると通信ネックになる
    - QCDの問題サイズは16倍ぐらい、マシンは50倍ぐらい？
    - ストロングスケーリングがよくなることはないと思われる
  - 実アプリ（ソルバーだけでなく）をターゲットアプリにしてすぐに開発後すぐにプロダクションできるように