Vol.2020-HPC-173 No.6
2020/3/16

HPCベンチマークプログラムによるA64FXプロセッサ試作機の性能評価

小田嶋 哲哉[1,a)]　児玉 祐悦[1]　辻 美和子[1]　松田 元彦[1]　丸山 豊[1]　佐藤 三久[1]

概要：理化学研究所計算科学研究センターではスーパーコンピュータ「富岳」の導入を進めている．そのCPUとして，Armv8-A+SVEアーキテクチャをベースとしたFujitsu A64FXプロセッサが採用されている．本稿では，A64FXテストチップを搭載した試作機において8つのHPCベンチマーク/アプリケーションの評価を行った．Arm ThunderX2プロセッサおよびIntel Xeon Skylakeプロセッサとの性能比較では，A64FXのもつ高いメモリバンド幅によりStreamのようなメモリバンド幅が重要なアプリケーションでは高い性能が得られた．一方，命令が連鎖する複雑なカーネルでは，ハードウェアリソースが不足することで演算性能の低下が発生したベンチマークも確認された．

1. はじめに

理化学研究所計算科学研究センターでは，「京」の後継機としてスーパーコンピュータ「富岳」の導入を進めている．そのCPUとして，ARM，富士通，理研のコデザインで開発が進められてきたA64FXプロセッサ [1] が採用されている．A64FXはArmv8.2-Aアーキテクチャをベースとし，SVE（Scalable Vector Extension）拡張やHBM2（High Bandwidth Memory ver.2）を搭載することで，高い演算性能とメモリバンド幅を有した汎用プロセッサである．2019年11月のGreen500 at SC19では，A64FXを搭載した富岳プロトタイプシステムが16.876GFLOPS/Wを達成し，アクセラレータを搭載したシステムより高い電力効率を達成したこともあり，非常に高い注目を集めている．

同センターのアーキテクチャ開発チームでは，これまでにA64FXのプロセッサシミュレータとして「理研シミュレータ」 [2] の開発を継続的に行っており，実際にA64FXを搭載したシステムの利用に先行して，アプリケーションの性能評価やチューニングに活用されている．2019年の中頃より，富士通の沼津工場に設置されている「富岳共用前評価環境」としてA64FXテストチップを搭載したA64FXプロセッサ試作機を数ノード利用することが可能になった．本チームでは，理研シミュレータと実際のプロセッサの性能の合わせ込みだけではなく，様々なHPCアプリケーションやベンチマークの評価も同時に行ってきた．

本稿では，海外の研究機関との共同研究として，A64FXプロセッサ試作機を用いて様々なHPCアプリケーションやベンチマークの性能評価を行う．評価アプリケーションに対して，アーキテクチャ固有のチューニングは行わず，As Isのプログラムをコンパイルすることで並列化や最適化を適用する．

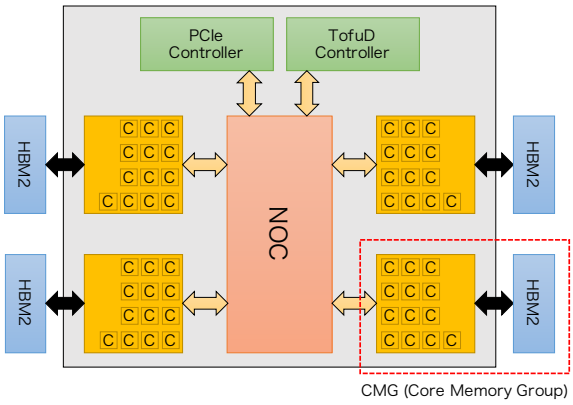


図 1　A64FX プロセッサの構成

2. A64FX プロセッサ試作機

本稿で使用するA64FXプロセッサ試作機には，実際のチップを作成する過程で作成されたテスト用のCPUが搭載されている．また，使用する富士通製のコンパイラは開発が進められている途中であるため，本稿の評価は実際の富岳やFX1000/FX700などのA64FX搭載のシステムの性能を保証しているものではないことに留意されたい．

図 1 にA64FXプロセッサの構成を示す．本評価で使用するCPUはテストチップであるが，基本的な性能は製品

[1] 理化学研究所 計算科学研究センター
[a)] tetsuya.odajima@riken.jp

表 1　評価アプリケーション

ベンチマーク	言語	並列化
RAJA Performance Suite	C++	OpenMP
BUDE	C	OpenMP
MiniFMM	C	OpenMP
LULESH	C++	MPI+OpenMP
mega-sweep	Fortran	MPI+OpenMP
CloverLeaf	C/Fortran	MPI+OpenMP
TeaLeaf	Fortran	MPI+OpenMP
HPCG	C++	MPI+OpenMP

版と同等である．A64FX には，12 個の計算コアと HBM2 が組となった CMG（Core Memory Group）と呼ばれる NUMA ノードが 4 個搭載されている．各 CMG には OS などのシステム用のコアとして 1 ないし 0 個のアシスタントコアが搭載されており，これらは演算に参加することはない．さらに，PCIe Controller と TofuD Controller がチップ内のネットワークで接続されている．詳しくは文献 [1] を参照されたい．

A64FX の計算コアは 64KB ずつの L1 データ/L1 命令キャッシュを持ち，CMG ごとに 8MB の L2 キャッシュを共有するとともに，CMG 間のキャッシュコヒーレンスもハードウェアでサポートしている．演算パイプとして 512-bit の浮動小数点演算ユニットと固定小数点演算ユニットを 2 つずつ有しており，動作周波数 2.0GHz 時のプロセッサ全体の理論ピーク演算性能は倍精度/単精度でそれぞれ 3.072TFLOPS/6.144TFLOPS である．DGEMM による実行効率は 90%以上と非常に高い演算性能を達成している．各 HBM2 は 8GB の容量と 256GB/s のメモリバンド幅であり，プロセッサ全体では 32GB のメモリ容量と 1024GB/s のメモリバンド幅を有している．Stream Triad による実行効率は 80%以上と，実行メモリバンド幅も非常に高い．

3. 評価アプリケーション/ベンチマーク

本稿で評価する HPC アプリケーション/ベンチマークは，**表 1** に示す 8 つである．評価に用いるデータサイズは，オンメモリになるように調整している．表には各ベンチマークの記述言語と並列化方法が示されている．本章では，各ベンチマークの概要とその特徴について述べる．

3.1 RAJA Performance Suite

RAJA Performance Suite [3] は，Lawrence Livermore National Laboratory で開発が行われているベンチマークスイートである．HPC アプリケーションで頻繁に使用されるループベースのカーネルをまとめており，カーネルごとの性能評価を行うことができる．「Apps」に属するカーネルは実際の HPC アプリケーションで使用されているものから構成され，「Basic」や「Stream」に属するカーネルは小規模な典型的な計算ループから構成される．本稿では「Basic_MULADDSUB」「Stream_DOT」「Apps_PRESSURE」「Apps_LTIMES」の評価を行う．

3.2 BUDE

BUDE（Bristol University Docking Engine）[4] は University of Bristol で開発が行われているベンチマークである．本プログラムは github などでオープンに公開されていないが，文献 [4] からコンタクトを取ることで取得することが可能である．BUDE は，タンパク質のドッキングに関するプログラムであり，分子間の相互作用を計算する多体問題のようなカーネルから構成される．

3.3 MiniFMM

MiniFMM [5] は University of Bristol で開発が行われている Fast Multipole Method（FMM）のミニアプリである．MiniFMM は，FMM を用いて 3 次元極座標平面におけるラプラス方程式の解を求める．FMM は多体問題の高速化手法として用いられており，遠くの粒子をグループ化して計算するツリー法を発展させたものである．直接法の演算量 $O(N^2)$ に対して，$O(N)$ で計算を行うことが可能になる．FMM の詳細については文献 [6] を参照されたい．

3.4 LULESH

LULESH（Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics）[7] は，Lawrence Livermore National Laboratory で開発が行われている流体力学のミニアプリである．LULESH は，非構造格子における 3 次元の Lagrangian hydrodynamics をシミュレートし，そのメモリアクセスは間接参照となる．アプリケーションの詳細については文献 [8] を参照されたい．

3.5 mega-sweep

mega-sweep [9] は，UK Mini-App Consortium で開発が行われている Stream 系のミニアプリである．mega-sweep は，同 Consortium が開発している mega-stream を主なカーネルとし，サイズが異なる配列を Stream Triad のように実行することでメモリバンド幅の上限に達しないというテストを目的として開発されている．これより，メモリバンド幅よりもメモリアクセスのレイテンシがパフォーマンスへの影響を大きくする．mega-stream の詳細については文献 [10] を参照されたい．

3.6 CloverLeaf

CloverLeaf [11] は，UK Mini-App Consortium で開発が行われている圧縮性流体のミニアプリである．２次元の圧縮性オイラー方程式を求める．データ構造は，セルの中心だけではなく角にもデータが存在するスタッガード格子と

表 2 評価環境一覧

	A64FX	TX2 (ThunderX2)	SKL (Skylake)
# cores	48C	56C (28Cx2S)	24C (12Cx2S)
Frequency	2.0GHz	2.0GHz	2.6GHz
SIMD	512-bit SVE	128-bit NEON	512-bit AVX512
Memory Peak BW	HBM2 1,024GB/s	DDR4-8ch 341.2GB/s	DDR4-6ch 255.9GB/s
Network	TofuD	IB FDRx1	IB HDRx4

なっている．ステンシル計算が主となり，メモリバンド幅が性能に大きく影響するアプリケーションである．プログラムの詳細は文献 [12] を参照されたい．

3.7 TeaLeaf

TeaLeaf [13] は，UK Mini-App Consortium で開発が行われている熱伝導方程式のミニアプリである．5 点差分による陰解法であるが，データが疎行列であるためカーネルでは CG 法などのソルバが使用されている．つまり，メモリバンド幅インテンシブなアプリケーションと言える．

3.8 HPCG

HPCG（High Performance Conjugate Gradients） [14] は，Jack Dongarra らが提案した主に共役勾配法の性能に関する HPC ベンチマークである．HPCG はリファレンスコードが提供されているが，本評価では ARM が開発を行っている SVE 向け拡張のコードを使用する．本ベンチマークのレギュレーションでは，十分に大きいデータサイズを 30 分以上実行することが必要であるが，本稿における評価では比較的小さいデータサイズを短い時間で実行する．

4. 性能評価

本章では，3 章で紹介した 8 つのアプリケーション/ベンチマークに対して，Fujitsu A64FX テストチップ，ARM ThunderX2 プロセッサおよび Intel Xeon Skylake プロセッサによる性能評価を行う．

4.1 評価環境

評価には**表 2** に示す 3 種類のプロセッサ環境を使用する．ThunderX2 (TX2) は，Marvel (旧 Cavium) の Armv8.1-A アーキテクチャのプロセッサである．サーバグレードの ARM CPU として，DOE のスーパーコンピュータ Astra にも採用されている．Skylake（SKL）は，Intel のサーバグレード CPU では最新の Skylake アーキテクチャのプロセッサである．本評価では筑波大学計算科学研究センターに設置されているスーパーコンピュータ Cygnus [15] の Deneb ノードを使用する．

表 2 より，各プロセッサの概要について説明する．A64FX には 1CPU で 48 個の計算コアを搭載しているが，TX2 および SKL はノードごとに 2CPU を搭載し，総コア数は表に示すとおりである．各プロセッサは，コアごとに浮動小数点演算ユニットを 2 つ搭載している．一方，周波数とベクトル幅がアーキテクチャごとに異なるため，コア当たりのピーク演算性能は SKL が最も高い．しかしながら，ノード単位では搭載するコア数が SKL の 2 倍ある A64FX が最も高い演算性能を達成している．TX2 および SKL はともに DDR4 2,666MHz のメモリを搭載しているが，チャネル数の違いによりピークメモリバンド幅の値が異なる．一方，A64FX は GPU などのアクセラレータに搭載されている HBM2 を採用しているため，1,024GB/s と汎用 CPU としては非常に高いピークメモリバンド幅を有している．

本稿では，データサイズを固定した Strong Scaling による評価を行う．OpenMP 実行の評価では SKL が 24 コアであるため，1 スレッドから 24 スレッドまでの評価とする．すべてのプロセッサにおいて，12 スレッドまでは 1CPU（1CMG），24 スレッドでは 2CPU（2CMG）を用いてプログラムを実行する．MPI+OpenMP ハイブリッド実行では，SKL は最大 2 ノード 4CPU の合計 48 スレッドまで評価を行う．一方，TX2 はコア数としては 1 ノード 2CPU で実行することが可能であるが，複数のノードを使用することで総メモリバンド幅が向上するため，SKL と同様に最大 2 ノード 4CPU とし，1CPU では最大 12 スレッドで実行する．A64FX は 48 コアを搭載するため，1 ノード 1CPU で評価を行う．

コンパイラはそれぞれ，Fujitsu Compiler 4.1.0 (開発中)，Arm HPC Compiler 19.1, Intel Compiler 19.0.3.199 を使用する．コンパイラオプションはそれぞれ “-Kfast,openmp”，“-Ofast -fopenmp -march=armv8.1-a”，“-O3 -qopenmp -march=native” を使用する．富士通コンパイラは現在開発が進められている途中であるため，実際の富岳のオペレーション開始時とは性能が異なることに留意されたい．また，SVE はベクトル幅に依存しない Vector Length Agnostic プログラミングモデルに対応しているが，本評価で使用するオプションの指定では富士通コンパイラは A64FX 向けの最適化として 512-bit 固定長のバイナリを生成する．

4.2 RAJA Performance Suite

問題サイズとして，キャッシュに収まらず，メモリにアクセスが発生するようにサイズを調整した．**図 2** と**図 3** にそれぞれ，RAJA Performance Suite の実行時間とベンチマークごとの A64FX 1 スレッドに対する相対性能を示す．各グラフの縦軸は実行時間および相対性能，横軸はスレッド数を示す．図 3 より，A64FX はすべてのカーネルで TX2 および SKL よりも高い性能を示している．

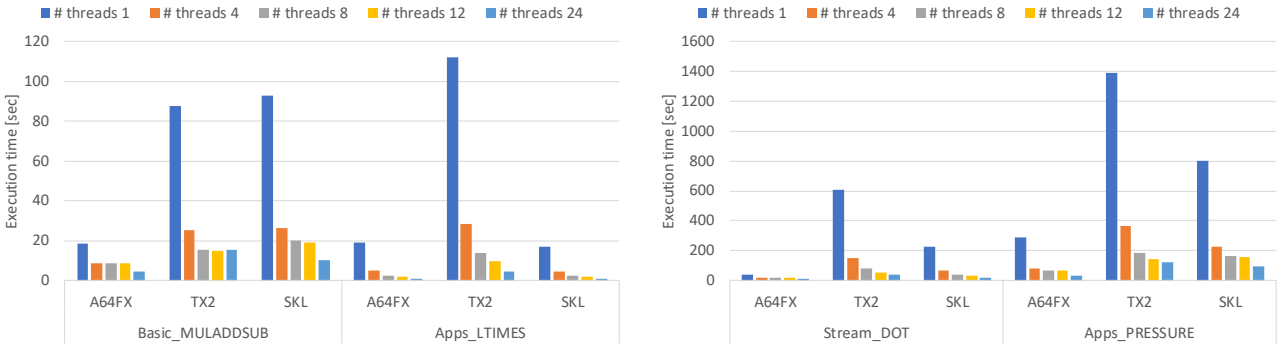


図 2 RAJA Performance Suite: 実行時間

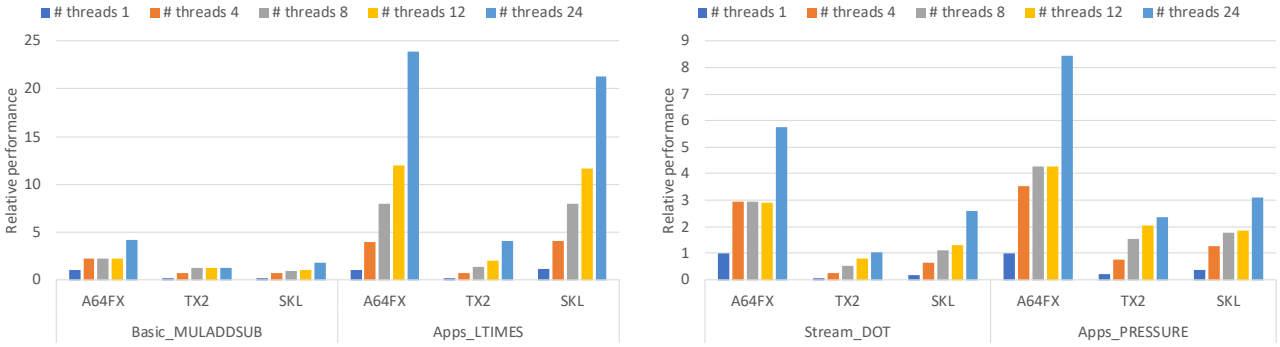


図 3 RAJA Performance Suite: A64FX 1 スレッドに対する相対性能

```
// Basic_MULADDSUB
for (Index_type i = ibegin; i < iend; ++i ) {
  out1[i] = in1[i] * in2[i] ;
  out2[i] = in1[i] + in2[i] ;
  out3[i] = in1[i] - in2[i] ;
}
```

Basic_MULADDSUM のカーネルは上記の通りである．2Load/3Store 命令に対して 3 つの演算命令しか無いため，メモリバンド幅律速なカーネルである．1 スレッドの実行時間で比較すると，A64FX は TX2 および SKL に対して約 5 倍高速である．各プロセッサの 1 チャネル当たりの理論ピークバンド幅性能比をみると $HBM2 : DDR4 = 6 : 1$ となり，実行時間とおおよそ比率は一致する．メモリアクセスのインターリーブにより 1 チャネルの性能以上のバンド幅が出ている可能性もあるが，1 スレッドでは演算量が少ないため結果としてメモリ 1 チャネルの性能に律速すると考えている．また，A64FX では 4 スレッドから，TX2 および SKL では 8 スレッド当たりで実行性能が変わらなくなっているが，このスレッド数でメモリバンド幅が飽和していると考えられる．24 スレッドで実行する際，2 つの NUMA ノードを使用するため総メモリバンド幅が 2 倍になり，実行時間も比例して 2 倍に高速化している．

```
// App_LTIMES
for (Index_type z = 0; z < num_z; ++z ) {
  for (Index_type g = 0; g < num_g; ++g ) {
    for (Index_type m = 0; m < num_m; ++m ) {
      for (Index_type d = 0; d < num_d; ++d ) {
        phi[m+ (g * num_g) + (z * num_z * num_g)] +=
        ell[d+ (m * num_m)] *
        psi[d+ (g * num_g) + (z * num_z * num_g];
      } } } }
```

App_LTIMES のカーネルは上記の通りである．演算性能律速なカーネルである．そのため，ベクトル幅が大きい A64FX および SKL の実行時間が TX2 よりも高速であることがわかる．1 スレッドの結果において，TX2 と他のプロセッサではベクトル幅は 4 倍の違いであるが，実行時間は 6 倍の差が出ている．これは，ベクトル幅だけではなくコンパイラの最適化の違いによるものと考えられる．A64FX は SKL よりも周波数が低いが，実行時間はおおよそ同じである．これは，SKL の AVX512 は通常の命令よりも実行周波数が低下することが知られており，その影響が実行時間に反映されたのではないかと考えている．

```
// Stream_DOT
for (Index_type i = ibegin; i < iend; ++i ) {
  dot += a[i] * b[i];
}
```

Stream_DOT のカーネルは上記の通りである．内積計算

であるため，メモリバンド幅律速なカーネルである．そのため，Basic_MULADDSUB で示したように 1 チャネル当たりのメモリバンド幅について，HBM2 は DD4 よりも 6 倍高速であるため，1 スレッドにおいて A64FX の実行時間は SKL に対して約 5 倍高速であることにおおよそ比例する．一方，TX2 がメモリバンド幅の比率以上に実行時間が大きい．これは，TX2 はベクトル幅が他のプロセッサより短いため，演算に必要なデータ転送がメモリバンド幅の上限に達していないことが理由であると考えられる．

```
// App_PRESSURE
for (Index_type i = ibegin; i < iend; ++i ) {
  bvc[i] = cls * (compression[i] + 1.0);
}
for (Index_type i = ibegin; i < iend; ++i ) {
  p_new[i] = bvc[i] * e_old[i] ;
  if ( fabs(p_new[i]) < p_cut ) p_new[i] = 0.0 ;
  if ( vnewc[i] >= eosvmax ) p_new[i] = 0.0 ;
  if ( p_new[i] < pmin ) p_new[i] = pmin ;
}
```

App_PRESSURE のカーネルは上記の通りである．カーネルは 2 つのループ文から構成されており，1 つ目のループは AXPY に近い演算であり，2 つ目のループは前のループの結果を用いて要素ごとにデータを比較して配列を更新するためメモリのプレッシャーが大きい．そのため，図 2 ではメモリバンド幅が大きい A64FX の実行性能が他のプロセッサよりも高いことがわかる．また，A64FX は 4 スレッドからメモリバンド幅律速に達し，24 スレッドでは総実行メモリバンド幅が 2 倍になるため実行時間が半分になっている．他のプロセッサはおおよそ 8 または 12 スレッドでメモリバンド幅律速に達している．12 スレッドの結果を見ると，理論ピークメモリバンド幅が大きい TX2 のほうが SKL よりも高速であることがわかる．しかしながら，TX2 において 2CPU 24 スレッドで実行すると SKL よりも性能が低下している．これは，OpenMP ライブラリにおけるスレッドマネジメントの差であると考えられるが，詳細は現在調査中である．

4.3 BUDE

BUDE は多体問題のようなカーネルを持つため，演算律速になりやすい．問題サイズとして，粒子数 32768，イテレーション数 1 を用いた．**図 4** に BUDE の実行性能を示す．縦軸は実効性能値である GFLOPS，横軸はスレッド数を示す．これより，どのスレッド数においても SKL が最も高い性能であることがわかる．演算性能インテンシブなカーネルでは，周波数の差はあるがベクトル幅が同じ A64FX は SKL に近い性能になると考えられるが，A64FX

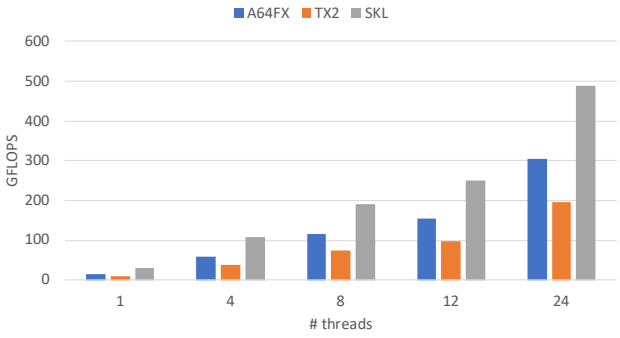
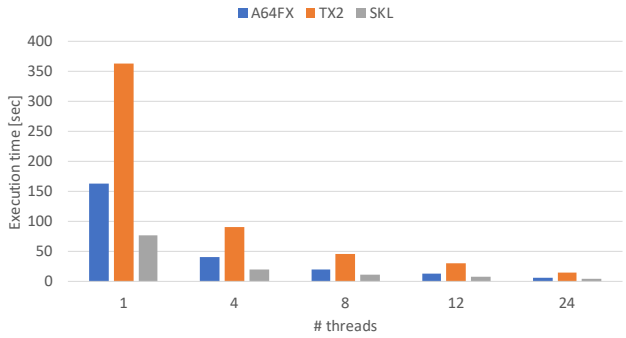


図 4 BUDE: GFLOPS

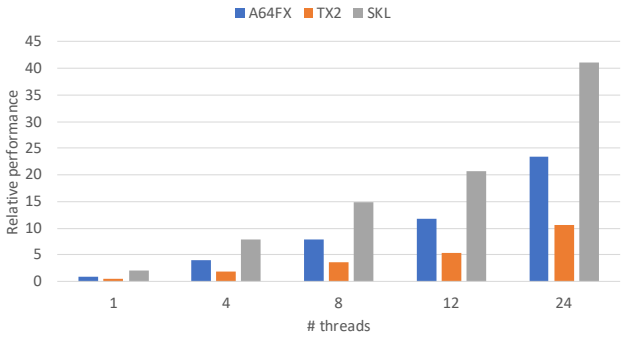
は SKL に対して 50〜60%の FLOPS にとどまる．当然ながら，コンパイラの最適化の違いもあることが想定されるが，それ以上にプロセッサが持つハードウェアリソースの違いが性能に大きく影響していると考えられる．SKL は 1 チップに 12 コアが搭載されているが，A64FX は 1 チップで 48（+4）コアを搭載する．そのため，ダイサイズは異なるが，1 コアあたりのチップ面積は SKL が大きくなるため，多くのハードウェアリソースを搭載することができる．BUDE のカーネルは，多くの変数をレジスタにロードし，それを使いまわして演算を行うため，Out-of-Order 実行を進めていくと途中でハードウェアリソースが足りなくなり，演算パイプラインがストールしてしまう．A64FX のようなメニーコアプロセッサは，Intel Xeon よりも相対的にハードウェアリソースが少ないため，ストールの頻度が多くなり，演算性能が低下してしまうため，A64FX の性能が向上しなかったと考えられる．一方，TX2 はベクトル幅が他のプロセッサより小さいため，スレッド数が同じ場合において演算性能が低くなったと考えられる．しかしながら，TX2 は比較のために搭載するコアの半分弱しか使用していないため，全コアを動員すればより高い性能を達成することが想定される．

4.4 MiniFMM

MiniFMM は，dual tree traversal によりツリーを走査しながら隣接するセルとのエネルギーを計算していく．ツリー走査では OpenMP のタスク構文を使用しており，同時に実行できるタスク，つまり，スレッド数が増えるほど演算性能が高くなる演算性能インテンシブなミニアプリである．問題サイズとして，ミニアプリが提供している “large” サイズを用いた．**図 5(a)** と**図 5(b)** にそれぞれ MiniFMM の実行時間と A64FX 1 スレッドに対する相対性能を示す．縦軸は実行時間および相対性能，横軸はスレッド数を示す．これより，BUDE と同様に SKL が最も高い性能であり，A64FX は SKL に対して 40〜50%低い性能であり，TX2 はさらに A64FX の半分程度の性能である事がわかる．演算性能インテンシブなアプリであるため，TX2 の性能が振るわないことは容易に想像が付くが，A64FX はコアあたり



(a) 実行時間



(b) A64FX 1 スレッドに対する相対性能

図 5 MiniFMM

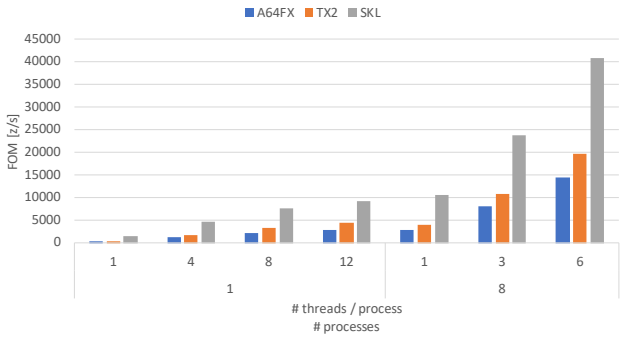


図 6 LULESH: 実行性能

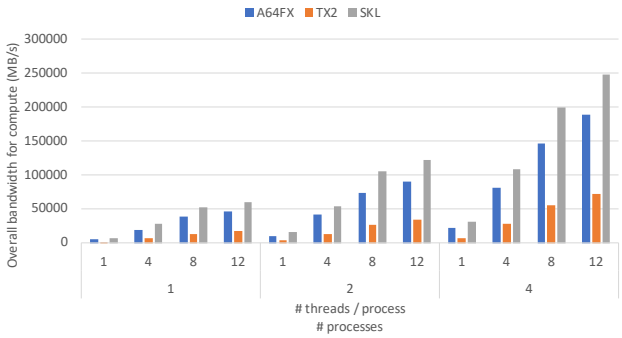


図 7 mega-sweep: Overall bandwidth

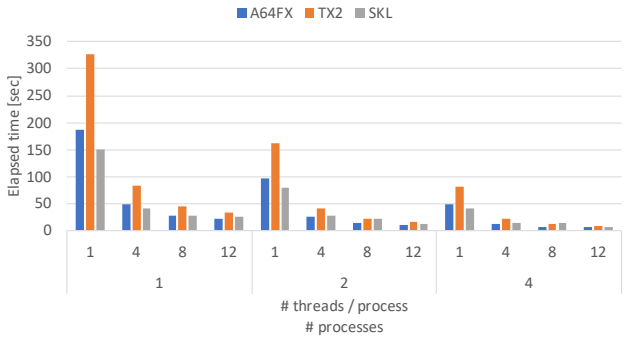
の演算性能として SKL とおおよそ近い性能が期待される．図 5(b) のような結果になった原因として，BUDE と同じくハードウェアリソースの差が性能に影響を与えたと考えられる．FMM では，全粒子に対して直接法で計算をする手法ではないものの，ツリー走査のあとに少なからず直接法で計算をする部分がある．多体問題の直接法は，演算命令の連鎖が多く，要求されるハードウェアリソースが多くなる．そのため，BUDE と同じようにハードウェアリソースの枯渇が頻繁に発生する．そのため，常に演算器を回すことが困難となり，アプリケーションの実行性能を低くしてしまったと考えられる．

4.5 LULESH

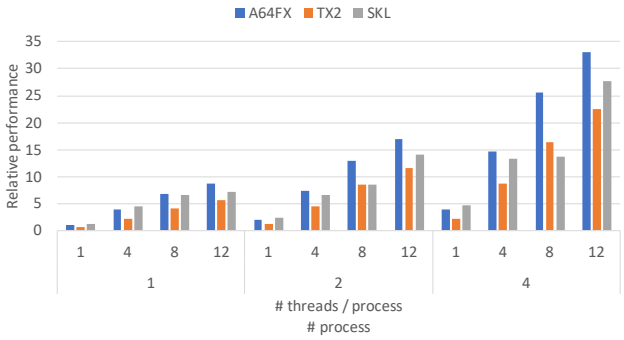
LULESH は，非構造格子の要素に対して間接参照を用いてアクセスするため，メモリに相当のプレッシャーがかかるミニアプリである．問題サイズはデフォルト値を使用する．**図 6** に LULESH の実行性能を示す．縦軸は時間あたりの処理性能である FOM，横軸は外側の値がプロセス数，内側の値がプロセスあたりのスレッド数を示す．これより，A64FX はどのプロセス数・スレッド数の組み合わせにおいても，TX2 および SKL よりも性能が低いということがわかる．この原因として，現在の富士通コンパイラは LULESH で使用されている間接参照部について，コンパイラの最適化としてインライン展開しか適用していない．A64FX のような Wide SIMD をもつプロセッサにおいて，一部でもベクトル化がされない部分があると，演算器が 1 要素ずつ処理を行うため演算器の性能を発揮することができない．つまり，配列アクセスが SIMD 化されていないため演算やメモリアクセスの効率が低下し，アプリケーションの実行性能が低下してしまったと考えられる．TX2 も同様に SIMD 化ができていないが，実行性能は A64FX よりも高い．これは TX2 のほうが A64FX と比べてベクトル幅が小さいため，メモリや演算器のレイテンシが小さいことが原因であると考えている．SKL はアセンブラを読む限り，128-bit での SIMD 化を行っている部分が存在する．これによって，他のプロセッサより倍以上の性能を達成したと考えられるが，調査を継続している．

4.6 mega-sweep

mega-sweep は，Stream 系のミニアプリである．単純な Stream 系アプリと異なる点は，イテレーションごとにデータにアクセスするパターンを変えていることである．mega-sweep ではイテレーションごとに 4 種類のアクセスパターンで配列を走査している．二次元配列の x, y 方向に対して各軸方向に $x++$ や $x--$ のように昇順，降順でアクセスしているためプリフェッチによるデータの先読みが困難である．最内ループは連続方向のアクセスであるため，ストライドアクセスは発生していない．通常，プリ

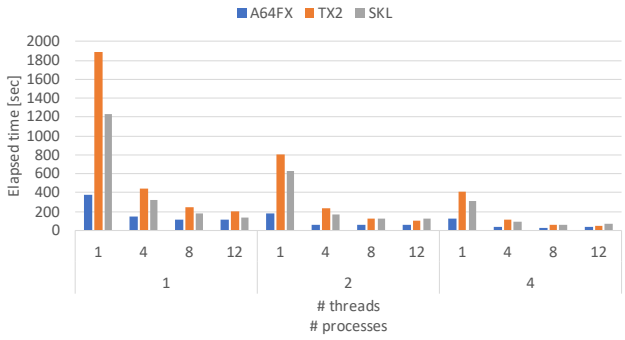


(a) 実行時間

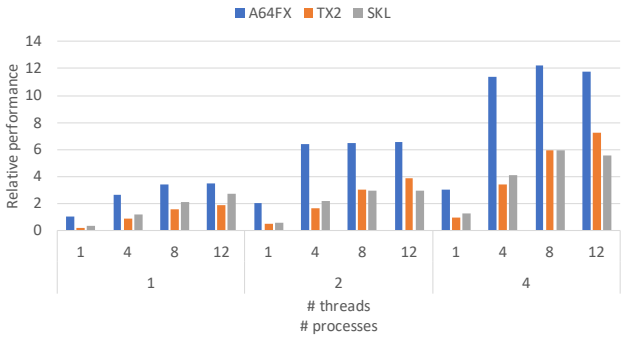


(b) A64FX 1 スレッドに対する相対性能

図 8 CloverLeaf



(a) 実行時間



(b) A64FX 1 スレッドに対する相対性能

図 9 TeaLeaf

フェッチでデータがすでにロードされている場合，演算はすぐに命令パイプに投入されるため，メモリや演算器のスループットが高くなる．一方，このアプリケーションではプリフェッチがうまく機能しないアクセスパターンであるため，演算をする場合毎回メモリにアクセスし，データをロードしてくる必要があるため，スループットが高くなりにくく，結果としてメモリアクセスレイテンシがアプリケーションの性能に影響する．**図 7** に通信時間を除いた “Overall bandwidth for compute” の値を示す．縦軸は実行バンド幅，横軸は外側の値がプロセス数，内側の値がプロセスあたりのスレッド数を示す．これより，SKL が最も高い性能であり，続いて A64FX である．TX2 はバンド幅性能が他のプロセッサよりも低いが，これも他のベンチマーク同様にベクトル幅が小さいため演算量が少なく，メモリへのプレッシャーが小さいことが原因である．搭載するすべてのコアを使用することで，よりバンド幅性能が向上することが考えられる．SKL のベクトル幅は A64FX と同じ 512-bit であるが，動作周波数と搭載するメモリの種類が異なる．SKL は定格で 2.6GHz であるが，AVX512 命令の実行では 1 割程度周波数の低下が発生する．加えて，Intel Xeon Phi（Knights Landing）のように，MCDRAM と DDR4 を比較すると DDR4 のアクセスレイテンシが小さい [16]．A64FX に搭載する HBM2 もどちらかといえば MCDRAM のようなスループットメモリであると想定されるため，A64FX と SKL の性能差はリーズナブルと言える．

4.7 CloverLeaf

CloverLeaf は，ステンシル計算が主となる圧縮性流体力学のミニアプリである．問題サイズは，アプリケーションが提供している “InputDecks/clover_bm16_short.in” を用いる．**図 8(a)** と**図 8(b)** にそれぞれ CloverLeaf の実行時間と A64FX 1 スレッドに対する相対性能を示す．縦軸は実行時間および相対性能，横軸は外側の値がプロセス数，内側の値がプロセスあたりのスレッド数を示す．1 プロセス 1 スレッドの性能は SKL が最も高く，1 プロセス 8 スレッドでおおよそ A64FX と同じになる．TX2 は SKL と同等の性能になるプロセス・スレッドの組み合わせもあるが，おおよそ SKL より低い性能である．アプリケーションのホットスポットがステンシル計算であるため，メモリバンド幅性能にアプリケーションの性能が比例することが多いが，図 8(b) ではピークメモリバンド幅の比率ほどの差は見られない．プロファイラの結果より，ホットスポットのカーネルでは，メモリアクセスの待ち時間と浮動小数点演算の待ち時間が同じ程度になっている．そのため，単純にメモリバンド幅に比例した性能になっていないと考えられる．プロセスあたりのスレッド数が増えたときに性能が向上していることからも，演算やメモリアクセス性能に律速していないことが言える．

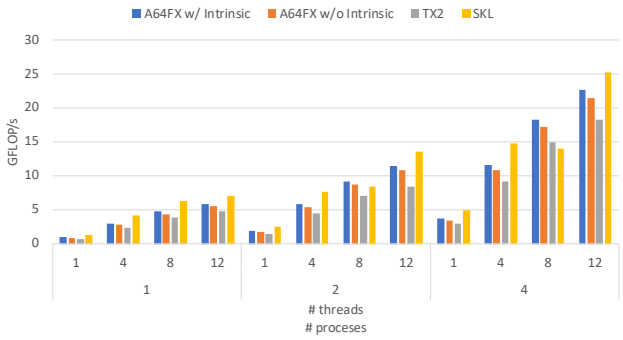


図 10 HPCG: 実行性能

4.8 TeaLeaf

TeaLeaf は，5 点差分のステンシル計算を主なカーネルとする熱伝導方程式のミニアプリである．内部で SPMV や CG などのソルバを持つが，陰解法である差分法は演算量が多い反面，データの再利用性が低いため，メモリに対するプレッシャーが大きい．問題サイズは，アプリケーションが提供している “Benchmarks/tea_bm_5.in” の繰り返し回数を 5 から 3 に減らしたパラメータセットを用いる．**図 9(a)** と**図 9(b)** にそれぞれ TeaLeaf の実行時間と A64FX 1 スレッドに対する相対性能を示す．縦軸は実行時間および相対性能，横軸は外側の値がプロセス数，内側の値がプロセスあたりのスレッド数を示す．これらのグラフより，A64FX が他のプロセッサよりも高い性能であることがわかる．特に，A64FX はプロセスあたりのスレッド数が 8 辺りから性能が飽和しており，プロセス数が増えるとそれに比例して性能が向上している．メインカーネルのホットスポットは CG を差分法を用いて解く部分であり，メモリへのプレッシャーが大きいためピークメモリバンド幅が高い A64FX が最も高い性能を達成している．SKL も同様におおよそ 8 スレッドで性能が飽和している一方で，TX2 は 12 スレッドまで性能が向上している．これは，TX2 のほうが搭載するメモリがソケット当たり 2ch 分多く，ピークメモリバンド幅が高いことが要因である．さらに，TX2 はベクトル幅が SKL よりも小さいため，スレッド数を増やさなければメモリバンド幅律速になるほどの演算量が確保できないため，このような結果になったと考えられる．

4.9 HPCG

HPCG は，CG 法について陽解法を用いて解くベンチマークである．本評価には，Arm から提供を受けたソースコードを用いる．これには複数の実装があるが，今回は ACLE（Arm C Language Extension）という Arm SVE 向けの組み込み関数（intrinsic）を用いた実装 “w/ Intrinsics” と ACLE を使わずに並列化を行った実装 “w/o Intrinsics” を用いる．TX2 および SKL では ACLE を使っていない実装に対して，各コンパイラの最適化を適用する．問題サイズは “–nx=96 –ny=96 –nz=96 –rt=0”[*1] を用いる．**図 10** に HPCG の実行性能を示す．Arm の実装は SVE を使用しているが，A64FX に最適化をしておらず，問題サイズも比較的小さいため，4 プロセス 12 スレッドでおおよそ 23GFLOPS となっている．A64FX での比較をすると，Intrinsic を使った実装のほうがコンパイラの最適化のみによる並列化よりも高い性能であることがわかる．現在の富士通コンパイラは開発途中であるため，今後更に SIMD 化やソフトウェアパイプライニングによる最適化が進むことが期待される．TX2 はピークメモリバンド幅が SKL よりも高いが，実行性能は SKL に劣っている．これは，TX2 はプロセッサに 28 コアを搭載しているが，今回の評価ではその半分以下しか使っていないため，有効にメモリバンド幅を使えていないことが理由としてあげられる．SKL はこれらのプロセッサの中で最も高い性能である．しかし，2 プロセスおよび 4 プロセスの 4，8 スレッド実行で性能がほぼ変わらず，12 スレッドになると急激に性能が向上している部分については現在調査中である．スレッド分割数が増え，キャッシュの利用性が高まったことが原因と推測している．

5. おわりに

本稿では，A64FX テストチップを搭載した試作機を用いて 表 1 に示す 8 つの HPC ベンチマーク/アプリケーションの評価を行ってきた．さらに，アプリケーションごとに ThunderX2 プロセッサおよび Intel Xeon Skylake プロセッサとの性能を比較した．これらより，A64FX は RAJA Performance Suite の Stream_DOT や CloverLeaf，TeaLeaf のようにメモリバンド幅が性能に影響するベンチマークで非常に高い性能を達成している．これは，512-bit という Wide SIMD を用いて十分な演算量を確保することで，HBM2 の高いメモリバンド幅を十分に使えていることが要因と考えられる．このように，高い演算性能とメモリバンド幅をもつ A64FX であるが，メニーコアプロセッサということもあり，コアあたりのハードウェアリソースが相対的に少ない．そのため，BUDE や MiniFMM では浮動小数点演算器を常に動かし続けることができず，SKL よりも低い実行性能となってしまった．このようなループ内の命令の依存関係が長いカーネルでは，ループ分割を行うことでリソース不足を緩和させる手法がある．富士通コンパイラでは，このようなループ分割を自動で行うための最適化が導入されているため，今後，このような機能を使って性能改善が得られるかを評価したい．

今後の課題として，今回はプログラムを As Is でコンパイルした結果を各プロセッサで評価したが，本来であればアーキテクチャごとに固有のチューニングやパラメータの

[*1] –rt=0 は Quick Path を使用するという意味

セットを行う必要がある．また今回の評価では，A64FX は 1 ノードに閉じていたが，複数ノードでの評価により通信性能を含めた検証を行いたいと考えている．

謝辞 本稿の一部は，文部科学省「特定先端大型研究施設運営費等補助金（次世代超高速電子計算機システムの開発・整備等 ）」で実施された内容に基づくものである．

参考文献

[1] Toshio Yoshida: Fujitsu High Performance CPU for the Post-K Computer, *Hot Chips: A Symposium on High Performance Chips (HC30)* (2018).
[2] Yuetsu Kodama, Tetsuya Odajima, Akira Asato and Mitsuhisa Sato: Accuracy Improvement of Memory System Simulation for Modern Shared Memory Processor, *HPCAsia2020: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, pp. 142–149 (2020).
[3] Lawrence Livermore National Laboratory: RAJA Performance Suite. `https://github.com/LLNL/RAJAPerf`.
[4] University of Bristol: BUDE. `http://www.bris.ac.uk/biochemistry/research/bude`.
[5] University of Bristol: MiniFMM. `https://github.com/UoB-HPC/minifmm`.
[6] 牧野淳一郎：高速多重極展開法とツリー法多体シミュレーションのための高速算法，応用数理， Vol. 8, No. 4, pp. 277–287 (1998).
[7] Lawrence Livermore National Laboratory: LULESH. `https://computing.llnl.gov/projects/co-design/lulesh`.
[8] Karlin, I., Bhatele, A., Keasler, J., Chamberlain, B. L., Cohen, J., Devito, Z., Haque, R., Laney, D., Luke, E., Wang, F., Richards, D., Schulz, M. and Still, C. H.: Exploring Traditional and Emerging Parallel Programming Models Using a Proxy Application, *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pp. 919–932 (2013).
[9] UK Mini-App Consortium: mega-sweep. `https://github.com/UK-MAC/mega-stream`.
[10] Tom Deakin and Wayne Gaudin and Simon McIntosh-Smith: On the Mitigation of Cache Hostile Memory Access Patterns on Many-Core CPU Architectures, *High Performance Computing*, Springer International Publishing, pp. 348–362 (2017).
[11] UK Mini-App Consortium: CloverLeaf. `https://github.com/UK-MAC/mega-stream`.
[12] Mallinson, A., Beckingsale, D., Gaudin, W., Herdman, A., Levesque, J. and Jarvis, S.: CloverLeaf: Preparing Hydrodynamics Codes for Exascale (2013).
[13] UK Mini-App Consortium: TeaLeaf. `https://uk-mac.github.io/TeaLeaf`.
[14] : HPCG Benchmark. `https://www.hpcg-benchmark.org/`.
[15] 筑波大学 計算科学研究センター：スーパーコンピュータ Cygnus. `http://www.tsukuba.ac.jp/wp-content/uploads/201903261300.pdf`.
[16] Avinash Sodani: Intel® Xeon Phi™ Processor “KnightsLanding” Architectural Overvie, *ISC2015 Workshop* (2015).