

PRIMEHPC FX100 の技術について

千葉 修一 (富士通株式会社)

Key Technology of PRIMEHPC FX100

by
Shuichi Chiba (FUJITSU LIMITED)

ABSTRACT

Combining high performance, scalability, and reliability with superior energy efficiency, FUJITSU Supercomputer PRIMEHPC FX100 further improves Fujitsu's supercomputer technology employed in the K computer and PRIMEHPC FX10. The system, from hardware to software, has been fully developed by Fujitsu, thereby delivering high levels of reliability and operability. The system can be flexibly configured to meet customer needs, capable of scaling to over 100 petaflops.

1. はじめに

FUJITSU Supercomputer PRIMEHPC FX100 は、2011 年に 2 期連続で世界一の性能^{*1} を達成したスーパーコンピュータ「京」(以下「京」)^{*2}、及び PRIMEHPC FX10 に適用した技術をさらに向上させたスーパーコンピュータである。高性能、高拡張性、高信頼性をあわせもち、省電力性に優れている。ハードウェアからソフトウェアまでを富士通で開発しており、利用者の要件にあわせて柔軟にシステム構築が可能で、最大構成 100PFLOPS を超える高いスケラビリティを実現する。本稿では、PRIMEHPC FX100 の技術について述べ、その評価結果を報告する。



Fig.1 Inheritance and Enhancement

2. PRIMEHPC FX100 の概要

2. 1. 超高速・超大規模の計算環境

PRIMEHPC FX100 のプロセッサである SPARC64TM XIfx は、最先端の 20nm プロセスで製造され、32 演算コア+2 アシスタントコアを実装し、1TFLOPS 以上の性能を発揮する。SPARC-V9 命令セットを HPC 向けに拡張した HPC-ACE2 (High Performance Computing - Arithmetic Computational Extensions 2) を導入することで利便性を大きく向上した。また、コアあたり 2 個の 256 ビット幅 SIMD ユニットの備えて演算スループットの高速化を図っている。1 ノードは 1 プロセッサから構成され、メモリに HMC (Hybrid Memory Cube) を採用したことにより、ノードあたり 480 GB/sec という圧倒的なメモリバンド幅を実現した。

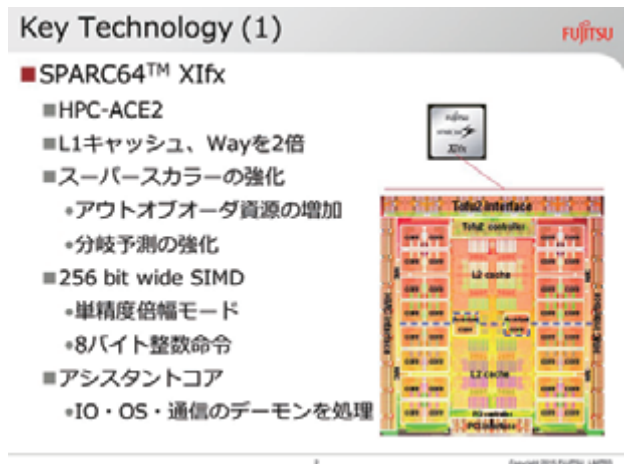


Fig.2 SPARC64TM XIfx

アシスタントコアは、OS やシステムソフトウェアの割り込み処理を行う。これにより、演算コアが演算処理に専念できる環境を提供する。また MPI のノンブロッキング通信に関しても、通信処理をアシスタントコアが処理することで、非同期処理の効率を高めている。

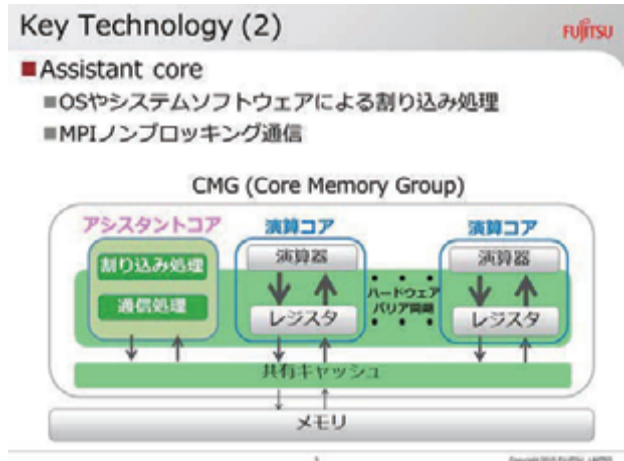


Fig.3 Assistant core

Tofu インターコネクト 2 (以下 Tofu2) は SPARC64™ XIfx プロセッサに統合されている。これにより、低遅延で実現されるノード間通信バンド幅はリンクあたり 12.5 GB/sec と高速化している。スケーラブルな Tofu2 により、10 万ノードを超える規模のシステム構築が可能である。

Key Technology (3) FUJITSU

■ Tofu2

- 「京」互換のトポロジ、通信方式
- 複数RDMAエンジンによる高速集団通信
- ハードウェアバリアのサポート

	京/FX10	FX100
CPUとの関係	SHST (ICC)	内蔵
トポロジ	6次元メッシュ-32	—
リンク バンド幅	5 GB/s (6.25 Gbps x 8 lanes x 10 dirs)	12.5 GB/s (25 Gbps x 4 lanes x 10 dirs)
ノード バンド幅	20 GB/s x in/out	50 GB/s x in/out
特徴	—	キャッシュ インジエクション アトミック シャード接続 (全体の2/3)を光化




Fig.4 Tofu2

2. 2. 高密度実装と水冷方式

2U の本体装置に 12 ノード、19 インチの専用ラックに本体装置を最大 18 台 (216 ノード) 搭載可能である。ノードを含め部品の 90%を直接水冷することで高密度と高信頼性を両立している。水冷により部品温度を下げることで、リーク電流の減少による消費電力の低減と同時に、部品の故障発生率の低減を実現する。

Key Technology (4) FUJITSU

■ Rack

- 216ノード / キャビネット
- CPU、メモリ、光モジュールを直接水冷(水冷率90%)

■ Chassis

- 19インチラックマウント型シャーシ
- 12ノード / 2U
- 本体装置間 Tofu2は光接続

■ CPU Memory Board

- CPU x 3
- 3 x 8 Micron's HMCs




Fig.5 Rack/Chassis/CPU Memory Board

2. 3. プログラミング開発環境

MPI 並列にスレッド並列を組み合わせたハイブリッド並列により、プロセス間通信やメモリ容量を効率的に活用することが可能になる。反面、ハイブリッド並列のプログラムは複雑になり開発コストが課題となる。

この課題に対して、VISIMPACT (Virtual Single Processor by Integrated Multi-core Parallel Architecture) はハイブリッド

並列を容易に実現することを可能とする。FUJITSU Software Technical Computing Suite のコンパイラにより MPI プログラムはハイブリッド並列に自動変換される。

また、ジョブ実行時にはプロセッサ内のハードウェアバリア、共有 L2 キャッシュにより高い実行性能を発揮する。FX100 では、「京」で課題であったキャッシュが大きく拡張されている。

Key Technology (5) FUJITSU

■ System comparison chart

	京	FX10	FX100
アーキテクチャ	SPARC64 VIIIIfx	SPARC64 IXIfx	SPARC64 XIfx
CPU性能	128 GFlops	236.5 GFlops	1 TFlops Class
コア数/CPU	8	16	32+2 ^①
SIMD データ幅	倍精度浮動小数点x2	倍精度浮動小数点x2	倍精度浮動小数点x4 単精度浮動小数点x8 64bit整数x4
キャッシュ	L1I\$: 32KB/core (2way) L1D\$: 32KB/core (2way) L2\$: 6MB/CPU	L1I\$: 32KB/core (2way) L1D\$: 32KB/core (2way) L2\$: 12MB/CPU	L1I\$: 64KB/core (4way) L1D\$: 64KB/core (4way) L2\$: 24MB/CPU
メモリ	16GB	32GB/64GB	32GB
スループット	64GB/s	85GB/s	240GB/s x2(R/W)

①アシスタントコア

Fig.6 System comparison chart

3. HPC-ACE2

PRIMEHPC FX100では、256 bit wide SIMDが搭載されている。倍精度浮動小数点演算に対する拡張に加え、整数型の SIMD 化を実現するための新命令、単精度浮動小数点演算を8演算同時実行するための新命令を追加している。また、HPC-ACE で搭載した三角関数補助命令、逆数近似命令 (除算、SQRT) に加え、新たに拡張命令を追加することで、命令レベルの並列性を高めることができる。以下に代表的な新命令を示す。

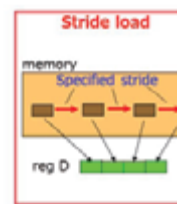
- Exponential Auxiliary 命令
- Stride Load/Store 命令
- Indirect Load/Store 命令
- Broadcast Load 命令

HPC-ACE2 (1) FUJITSU

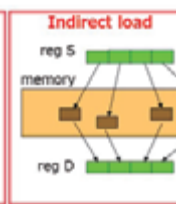
■ Instruction enhancement

- Stride Load/Store
- Indirect Load/Store
- Permutation

Stride load



Indirect load



Permutation

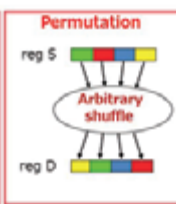


Fig.7 Instruction enhancement

3. 1. Stride Load/Store

等間隔のストライド幅でメモリをアクセスする場合、Stride Load/Store 命令を利用しメモリアクセス処理を同時実行することができる。この命令を利用することで、FX10 に比べ約 4 倍の性能向上が確認できている。これは SIMD 幅が 2 倍になったことに加え、命令レベルの高速化の効果である。特に連続体コードなどの間隔アクセスに対して性能効果が期待できる。

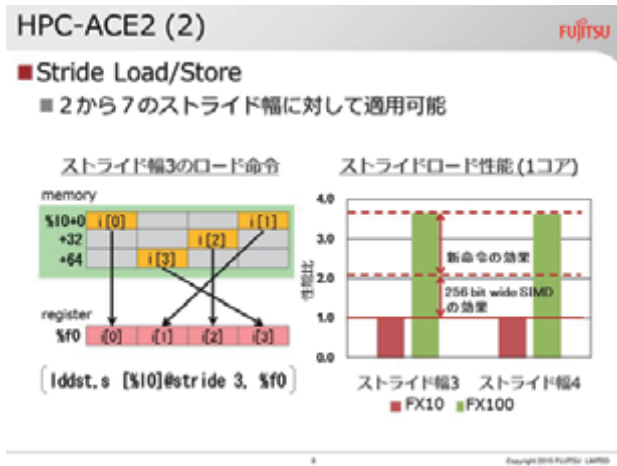


Fig.8 Stride Load/Store

3. 2. Indirect Load/Store

ストライド幅が異なるメモリをアクセスする場合、Indirect Load/Store 命令を利用しメモリアクセス処理を同時実行することができる。メモリアドレスの計算処理が複雑になるため、Stride Load/Store 命令に比べて効果は弱いが、コンパイラなどが適切な場所に適用することで大きな効果を得られる。流体解析、FEM などのリストアクセスに対して性能効果が期待できる。

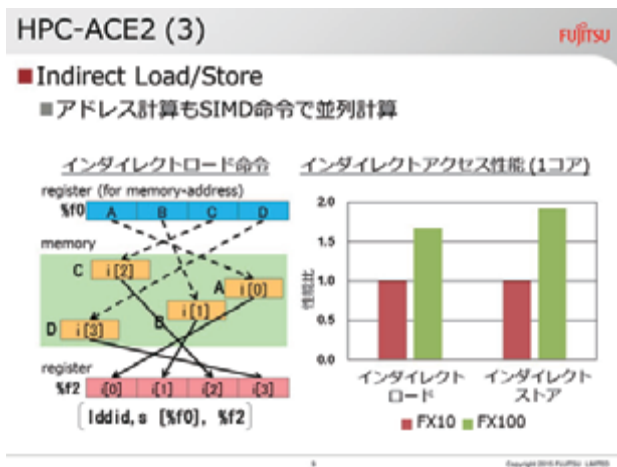


Fig.9 Indirect Load/Store

3. 3. Unaligned SIMD Store

「京」、および FX10 では SIMD Store 命令にデータ境界の制約があった。そのため、コンパイラが境界補正処理を挿入することで SIMD 化を適用可能としていた。FX100 では SIMD Store 命令がデータ境界に影響を受けることなく実行できるため、境界補正処理を削減することができる。これにより分岐命令などが削減され、命令レベルの最適化が促進する。また、コードサイズが削減されることで、命令キャッシュの効率的な利用にもつながる。

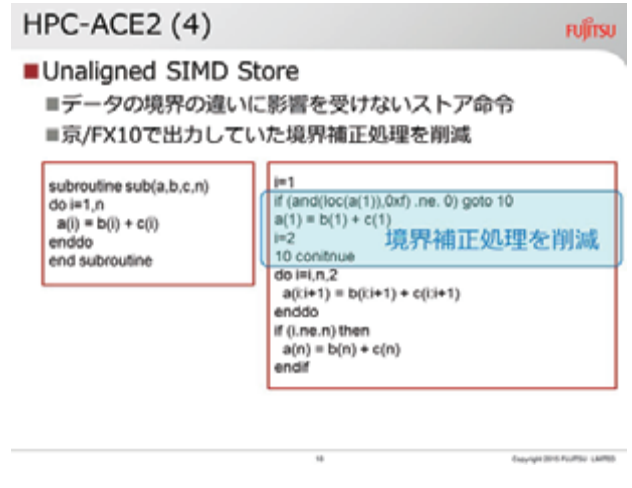


Fig.10 Unaligned SIMD Store

3. 4. Integer Condition Branch

「京」、および FX10 では整数型の条件判定の IF 構文を含むループを SIMD 化することができなかった。そのため、条件判定を実数型に変形することで SIMD 化を適用していたが、型変換を行うコストが必要となっていた。FX100 では、整数型の新命令を用意することで型変換を削減できる。これにより、命令レベルの並列性が向上し高速化が期待できる。

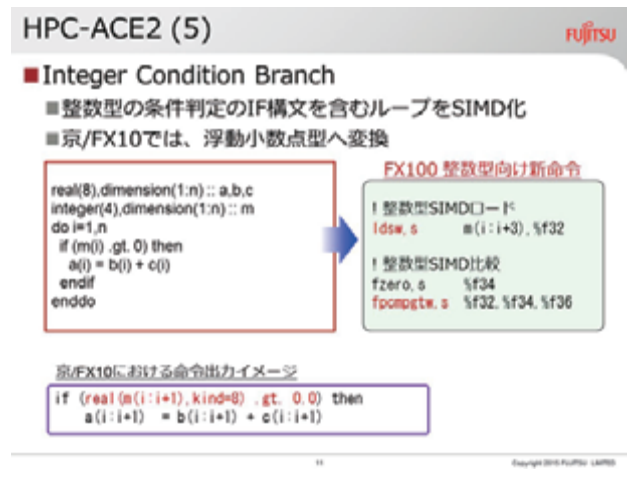


Fig.11 Integer Condition Branch

3. 5. New Complex-number Model

「京」、および FX10 では複素数の乗算処理に対して、隣接するメモリアクセスを SIMD 化し 11 命令で処理してい

た。FX100では先述した Stride Load/Store 命令を利用し7命令で処理することが可能となる。

HPC-ACE2 (6) fujitsu

■ **New Complex-number Model**

- Stride Memory Accessを利用した命令出力
- 京/FX100では隣接メモリをSIMD化

```

subroutine sub(a,b,r,n)
  complex(8),dimension(1:n) :: a,b
  real(4) :: r
  do i=1,n
    a(i) = b(i) * r
  enddo
end subroutine
                    
```

FX100 命令出力イメージ

```

ldd [%o2], %f2 1引数r
lddst.s b(i:i+3),r,%f4
lddst.s b(i:i+3),i,%f6
fmuld.s %f4,%f2,%f4
fmuld.s %f6,%f2,%f6
stdst.s %f4,a(i:i+3),r
stdst.s %f6,a(i:i+3),i
                    
```

⇒ 全7命令

※ 京/FX100での命令数
全11命令 (ロード×3,ストア×2,演算×4,初期化×2)

Fig.12 New Complex-number Model

3. 6. Concatenation Shift

Concatenation Shift 命令は SIMD レジスタの内容をシフトし結合する命令である。この命令を利用することで、ステンスル計算のような隣接したメモリアクセスを複数回行うコードのロード命令を削減することが可能となる。メモリ要求が多いコードに対して大きな効果が期待できる。

HPC-ACE2 (7) fujitsu

■ **Concatenation Shift**

- ループの回転方向のメモリ参照を結合

```

do i=1, n
  a(i) = b(i) + b(i+1) + b(i+2) + b(i+3)
enddo
                    
```

FX100 命令出力イメージ

```

T1 = b(i:4)
do i=1, n-4, 4
  T2 = b(i+4:i+7)
  T3 = concatenate_shift(T1, T2, 1)
  T4 = concatenate_shift(T1, T2, 2)
  T5 = concatenate_shift(T1, T2, 3)
  T6 = T1 + T3
  T7 = T4 + T5
  a(i:i+3) = T6 + T7
  T1 = T2
enddo
                    
```

※先行LOAD
 ※b(i+1:i+4)のLOAD命令を交換
 ※b(i+2:i+5)のLOAD命令を交換
 ※b(i+3:i+6)のLOAD命令を交換

ループ内は1つのSIMDロード命令のみ出力
b(i+4:i+7)

Fig.13 Concatenation Shift

3. 7. Element Compress

Element Compress 命令は、マスクを利用して SIMD レジスタの内容を圧縮することができる。この命令を利用することで、IF 構文を利用した配列データの収集処理から分岐命令を削除でき、ループに対して SIMD 化を適用することが可能となる。配列圧縮コードなどに対して高速化が期待できる。

HPC-ACE2 (8) fujitsu

■ **Element Compress**

- マスクを利用した圧縮、水平加算命令の出力
- ループ内の分岐を削減

```

J=1
DO I=1,10000
  IF (X(I) .GT. 0.00)
    THEN
      A(J)=B(I)
      J=J+1
    ENDIF
  ENDDO
                    
```

FX100 命令出力イメージ

```

J=1
DO I=1,10000,4
  MASK = X(I:I+3) .GT. 000
  VTD = B(I:I+3)
  CVTD=fecpd(VTD,MASK)
  A(J:I+3)=CVTD
  J=J+INT(fesumd(MASK),KIND=4)
ENDDO
                    
```

Fig.14 Element Compress

3. 8. Masking Loop SIMD

ループ処理に対して SIMD 化を適用した時、SIMD 幅に応じて余りの回転分の処理が必要となる。そこで、マスク演算を利用して余りの回転分の処理も SIMD 化し、コストを削減する。この処理は、演算が冗長実行となるため、最適化オプションを利用した場合のみ適用される。

今後、SIMD 幅の増加が考えられる HPC 分野では、余りの回転分の処理もコストに大きく影響する。Masking Loop SIMD により一定のコストで処理することが可能となる。

HPC-ACE2 (9) fujitsu

■ **Masking Loop SIMD**

- マスク演算を利用した小回転ループのSIMD化
- 演算が冗長実行となるため最適化オプションにより適用

```

do i=1,n
  a(i) = b(i) + c(i)
enddo
end subroutine
                    
```

FX100 命令出力イメージ

```

Loop:
  nn = 残り回転数/4
  idx = (nn=1) ?4:nn
  ldd.s mtbl[idx],%f8
  ldd.s b(i:i+3),%f2
  ldd.s c(i:i+3),%f4
  fadd.s %f2,%f4,%f6
  stdfr.s %f6,%f8,a(i:i+3)
  branch Loop,cond
                    
```

```

mtbl[0]=[F,F,F,F]
mtbl[1]=[T,F,F,F]
mtbl[2]=[T,T,T,F]
mtbl[3]=[T,T,T,T]
                    
```

Fig.15 Masking Loop SIMD

4. 性能評価

4. 1. ノード内性能

NAS Parallel Benchmarks FT コードを用いて、FX10 と FX100 における実行時間を比較検証する。比較検証する実行データは、以下の2つとする。

- シングルスレッド実行
- マルチスレッド実行(CPU内の最大コア数)

実行時間はシングルスレッド実行で2倍、マルチスレッド実行で3.3倍の性能向上となっている。シングルスレッド実行では、キャッシュ待ち、および演算待ちのコストが大きく削減されている。また、マルチスレッド実行では、メ

モリアクセス待ち、およびキャッシュアクセス待ちのコストが大きく削減されている。

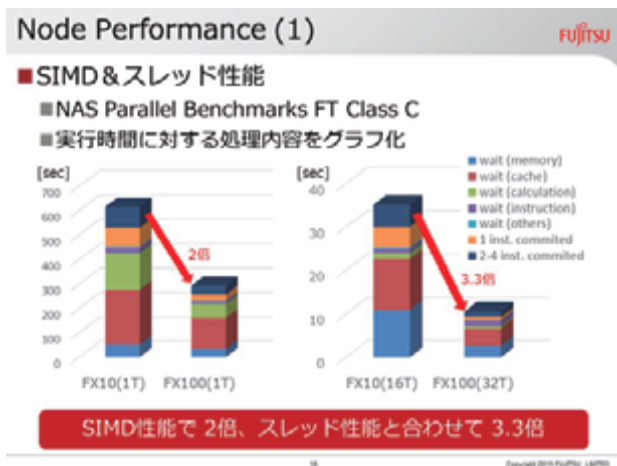


Fig.16 Performance Graph

FX10からFX100でSIMD化向上比率が1.0となっていることから、適用されたSIMD命令数に変化しないことがわかる。レーダーチャートで出力命令の分布を確認すると、FX10で適用していたSIMD Load/Store命令がFX100では新命令のStride Load/Store命令に変化していることがわかる。新命令の効果が大きく表れているコードといえる。

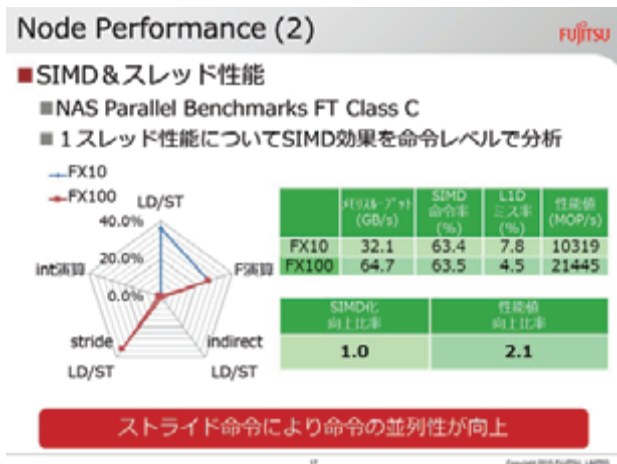


Fig.17 Performance Chart

4. 2. アプリケーション性能

宇宙航空研究開発機構 JAXA が所有する 6 種のシミュレーションコードを用いて、FX1、FX10、および FX100 における実行性能を比較する。FX1 の実行性能を 1 とした場合の性能向上率を比較した時、FX100 の実行性能は FX1 に対して 20 倍、FX10 に対して 4 倍となっている。

また、グラフが示すようにすべてのアプリケーションで同等の性能向上率を確認した。このことから、多くのアプリケーションで性能向上が期待できる。

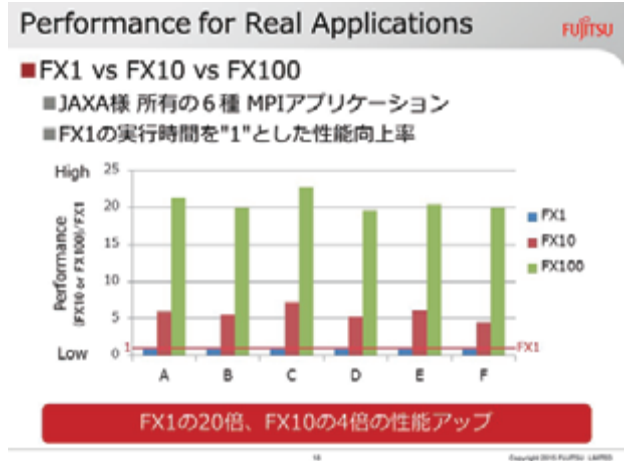


Fig.18 Performance for Real Applications

5. まとめ

PRIMEHPC FX100 は、スーパーコンピュータ「京」などのアーキテクチャーを継承した、最新鋭のスーパーコンピュータである。新技術を搭載により多岐にわたる分野でアプリケーション性能を向上が期待できる。

※1 世界一の性能: 2011年に発表された第37回および第38回TOP500リストにおいて達成

※2 理化学研究所と富士通が共同開発したスーパーコンピュータ「京」は理化学研究所の登録商標です。

参考文献

- 1) Shuichi Chiba, “PRIMEHPC FX10 後継機における性能と評価”, サイエンティフィック・システム研究会, 科学技術計算分科会 2014 年度会合, http://www.sskn.gr.jp/MAINSITE/event/2014/20141029-sci/lecture-04/SSKEN_sci2014_chiba_presentation.pdf
- 2) FUJITSU LIMITED, “FUJITSU Supercomputer PRIMEHPC FX100”, <http://www.fujitsu.com/global/products/computing/servers/supercomputer/primehpc-fx100/index.html>
- 3) NASA Advanced Supercomputing Division, <https://www.nas.nasa.gov/publications/npb.html>