FUJITSU
shaping tomorrow with you

PRIMEHPC FX100の特徴と概要

2015年 6月 19日

富士通株式会社
次世代テクニカルコンピューティング開発本部
千葉　修一

＊「京」は理化学研究所の登録商標です。
＊スーパーコンピュータ「京」は、理化学研究所と富士通の共同開発したシステムです。

Agenda

- キーテクノロジー
- ノード性能
- システム性能
- サポート機能

FUJITSU

キーテクノロジー

キーテクノロジー (1)

FUJITSU

FX10のフィードバック

評価点

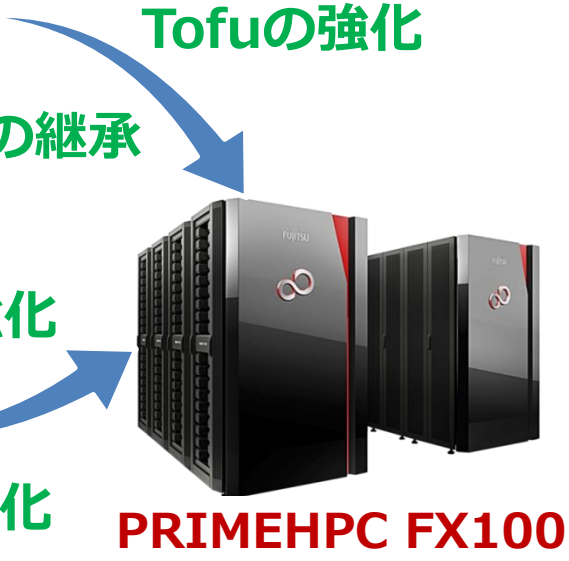
システム性能は評価高

- Tofuインターコネクトによる高いスケーラビリティ
- 超並列システムとして他に類を見ない信頼性
- 大規模演算を高速化する高いメモリスループット
- VISIMPACTによるハイブリッド並列

課題点

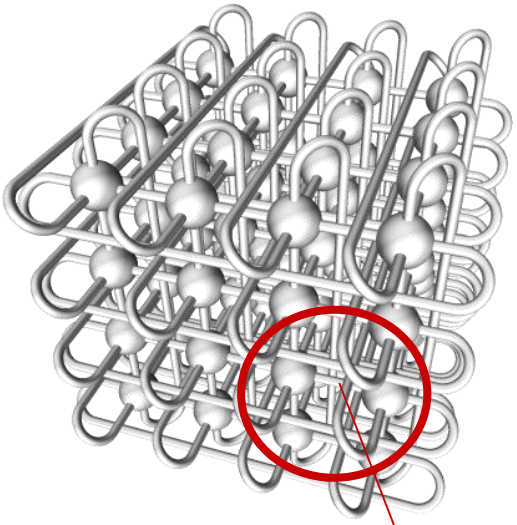
ノード性能が課題

- アウトオブオーダの資源不足
- L1キャッシュが貧弱
- 最適化の機能不足



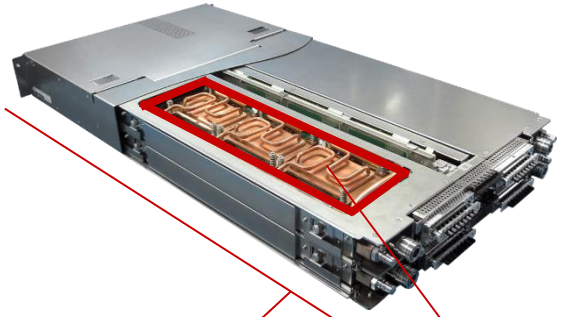
キーテクノロジー (2)

■ハードウェア構成

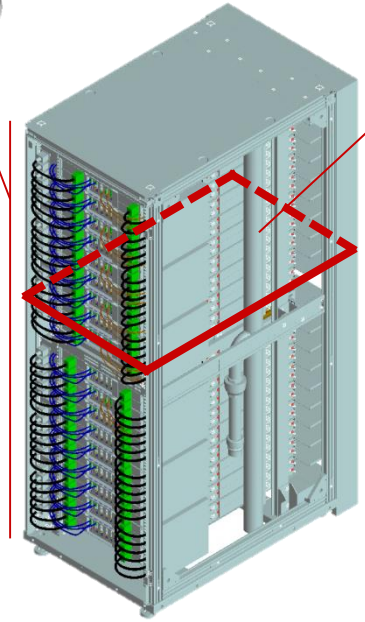


Tofu2

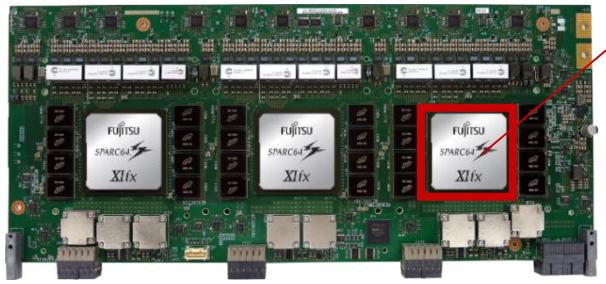
Chassis



SPARC64™ XIfx



Rack



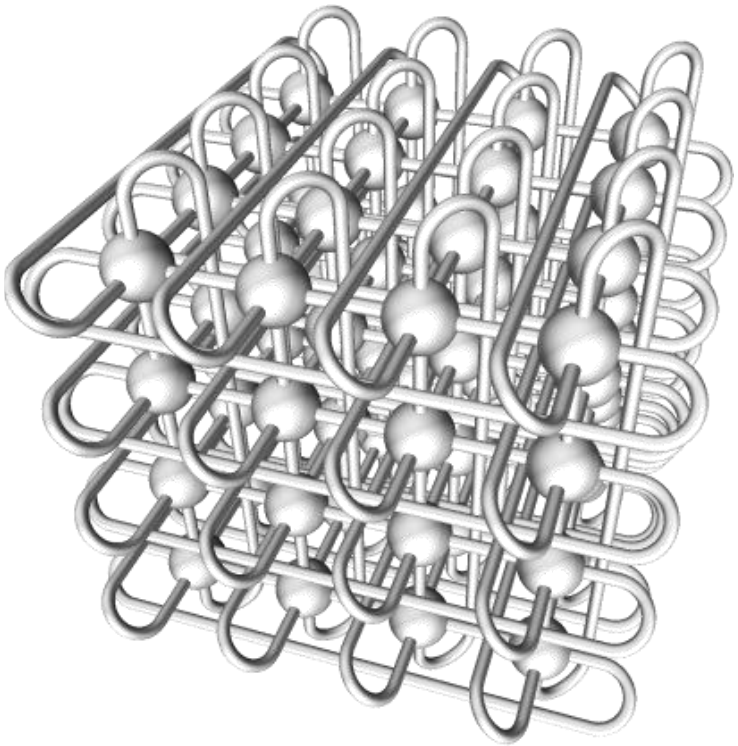
CPU Memory Board

キーテクノロジー (3)

FUJITSU

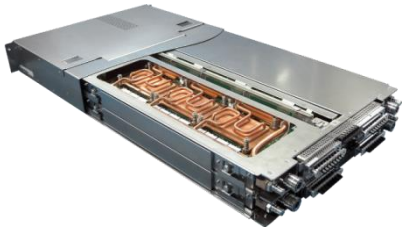
- Tofu2
  - 「京」互換のトポロジ、通信方式
  - 複数RDMAエンジンによる高速集団通信
  - ハードウェアバリアのサポート

	京/FX10	FX100
CPUとの関係	別LSI（ICC）	内蔵
トポロジ	6次元ﾒｯｼｭ/ﾄｰﾗｽ	←
ﾘﾝｸﾊﾞﾝﾄﾞ幅	5 GB/s (6.25 Gbps x 8 lanes x 10 dirs)	12.5 GB/s (25 Gbps x 4 lanes x 10 dirs)
ﾉｰﾄﾞﾊﾞﾝﾄﾞ幅	20 GB/s x in/out	50 GB/s x in/out
新機能	-	キャッシュインジェクション アトミック シャーシ間接続(全体の2/3)を光化



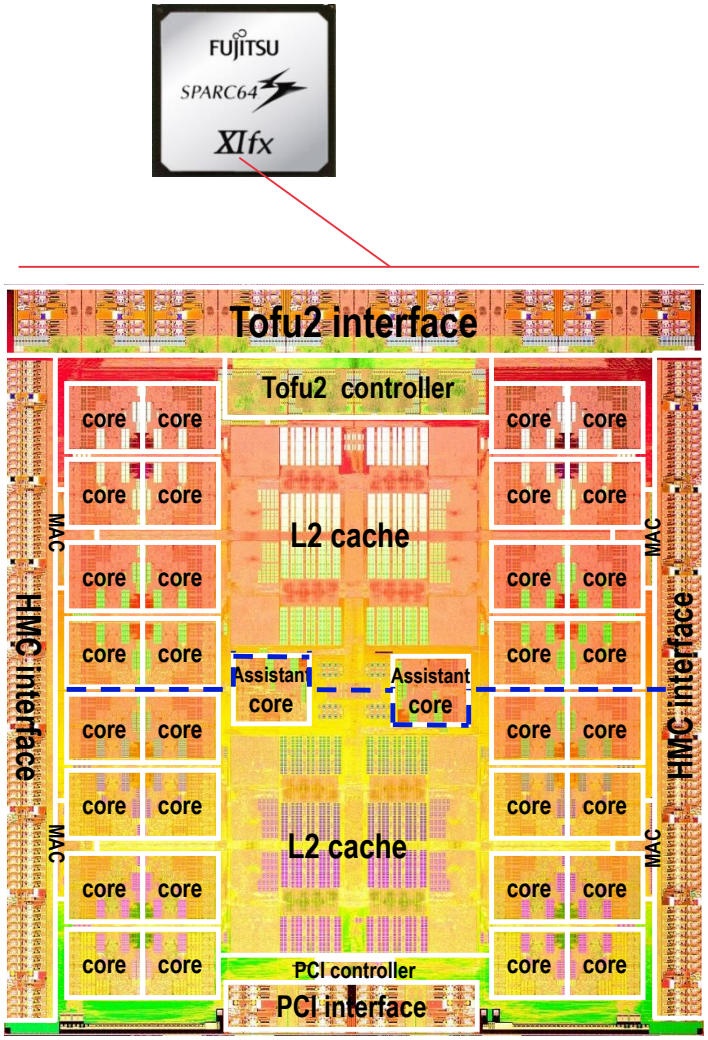
キーテクノロジー (4)

- 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



キーテクノロジー (5)

- SPARC64™ XIfx
  - HPC-ACE2
  - L1キャッシュ、Wayを2倍
  - スーパースカラーの強化
    - アウトオブオーダ資源の増加
    - 分岐予測の強化
  - 256 bit wide SIMD
    - 単精度倍幅モード
    - 8バイト整数命令
  - アシスタントコア
    - IO・OS・通信のデーモンを処理



キーテクノロジー (6)

FUJITSU

SPARC64™ XIfx

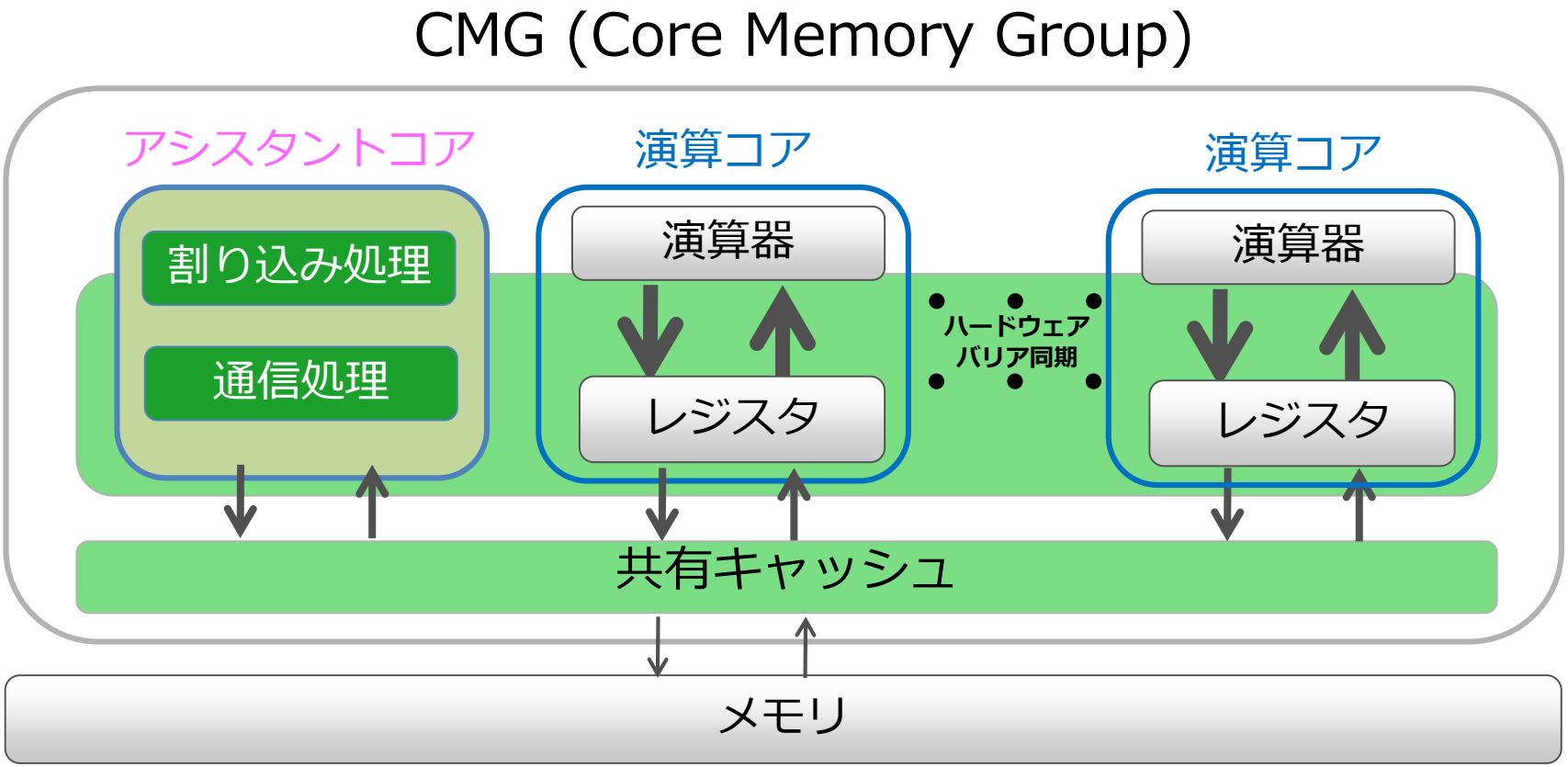
	京	FX10	FX100
アーキテクチャ	SPARC64 VIIIfx	SPARC64 IXfx	SPARC64 XIfx
CPU性能	128 GFlops	236.5 GFlops	1 TFlops Class
コア数/CPU	8	16	32+2※
SIMD データ幅	倍精度浮動小数点x2	倍精度浮動小数点x2	倍精度浮動小数点 x4 単精度浮動小数点 x8 64bit整数 x4
キャッシュ	L1I$: 32KB/core (2way) L1D$:32KB/core (2way) L2$: 6MB/CPU	L1I$: 32KB/core (2way) L1D$:32KB/core (2way) L2$: 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)

※アシスタントコア

キーテクノロジー (7)

FUJITSU

- アシスタントコア
  - OSやシステムソフトウェアによる割り込み処理
  - MPIノンブロッキング通信



FUJITSU

ノード性能

FX10の評価

FUJITSU

- ノード性能の課題
  - 命令レベルの並列化が弱い
    - 実アプリケーションへのSIMD命令適用率が低い
  - アプリケーションの高速化にチューニングが必須
    - L1キャッシュの32KB/2WAYが使いにくい
    - 実行性能にブレが発生する
    - チューニング時、キャッシュ効率or最適化の選択肢が難しい
    - コンパイラの最適化が不足
  - C／C＋＋アプリケーションの性能問題
    - 富士通コンパイラよりGNUコンパイラの翻訳コードの方が高速

ノード性能を支える技術

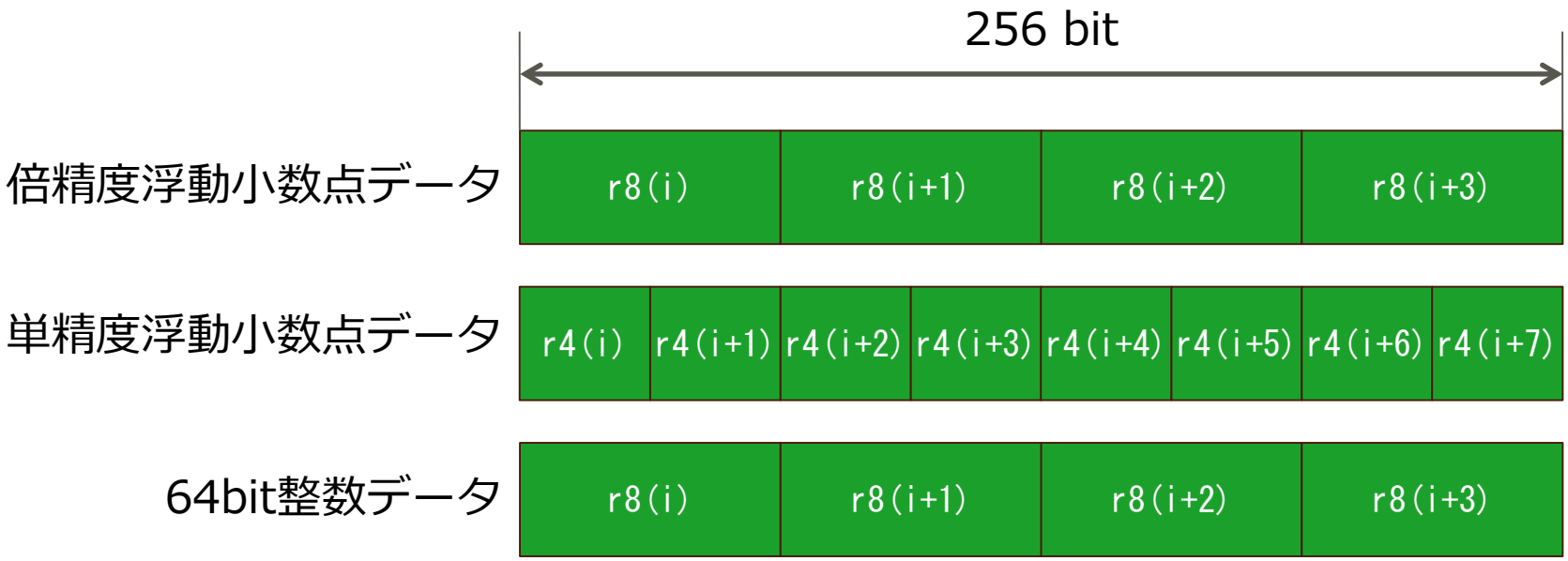
FUJITSU

- HPC-ACE2 (**H**igh **P**erformance **C**omputing - **A**rithmetic **C**omputational **E**xtensions **2**)
  - 256 bit wide SIMD
  - HPC向け拡張命令
- メモリ／キャッシュ
  - HMC採用によるスループット強化
  - Ｌ１キャッシュの強化
- コンパイラ
  - 最適化の強化
  - 並列化解析能力の強化

HPC-ACE2 (1)

FUJITSU

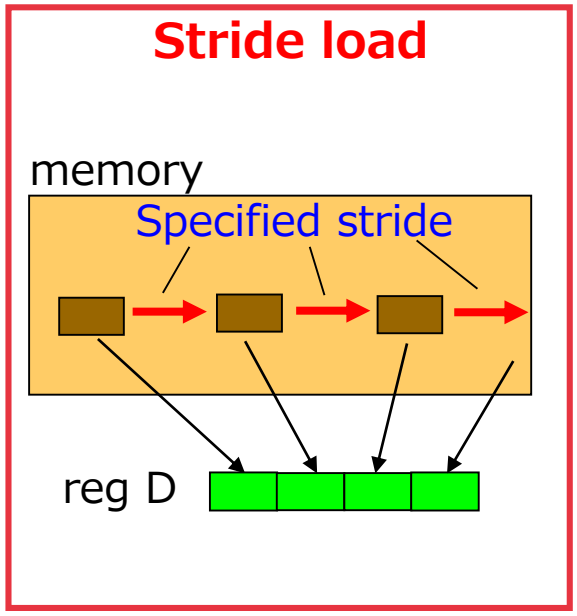
- 256 bit wide SIMD
  - 倍精度浮動小数点データ x 4
  - 単精度浮動小数点データ x 8
  - 64bit 整数データ x 4

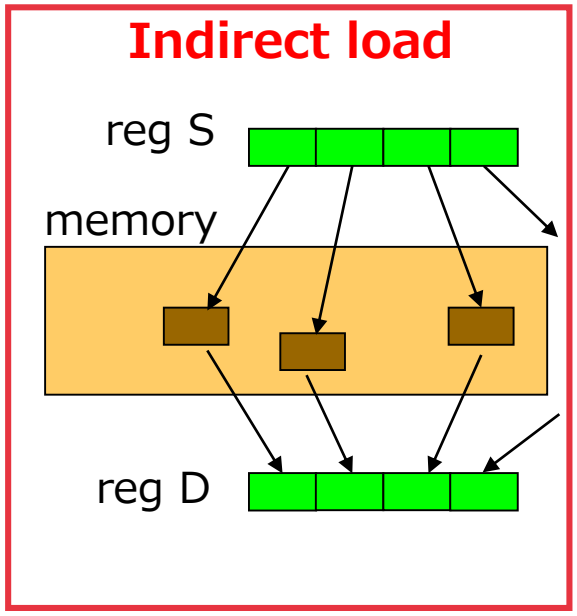


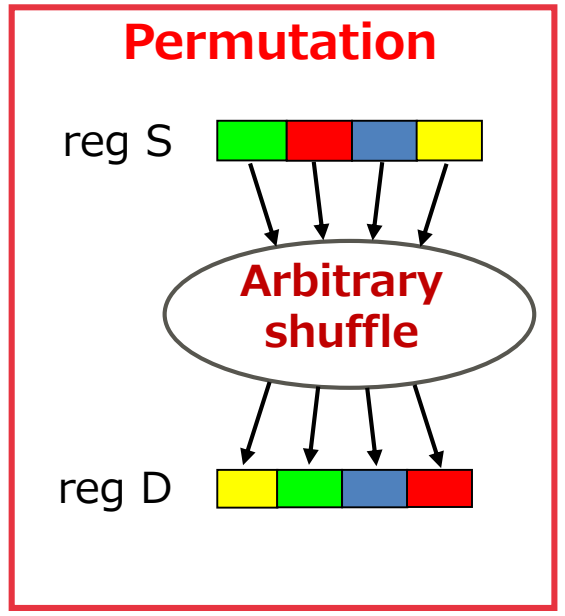
HPC-ACE2 (2)

FUJITSU

- 新規命令
  - Stride Load/Store
  - Indirect Load/Store
  - Permutation
  - Concatenate







メモリ／キャッシュ

FUJITSU

メモリ／キャッシュの強化

- HMCサポートによるスループット強化
- L１キャッシュの強化

	京	FX10	FX100
L１キャッシュ（命令）	32KB/core (2way)	32KB/core (2way)	64KB/core (4way)
L１キャッシュ（データ）	32KB/core (2way)	32KB/core (2way)	64KB/core (4way)
L２キャッシュ	L2$: 6MB/CPU	L2$: 12MB/CPU	L2$: 24MB/CPU
メモリ	32GB	32GB/64GB	32GB
スループット	64GB/s	85GB/s	240GB/s x2(R/W)

コンパイラ (1)

FUJITSU

- コンパイラの最適化を強化
  - クローニング

    ループを複写し条件ごとにループの動作を切り替え
  - ショートループ最適化

    回転数少のループへソフトウェアパイプライニングの適用
  - プロシージャ間最適化

    インライン展開、定数伝播、メモリレイアウト変更など
  - C/C++向け機能拡張

    ポインタ解析の強化、C++11規格サポート

コンパイラ (2)

FUJITSU

- コンパイラの並列化解析能力を強化
  - 命令レベルの並列化
  - コアレベルの並列化
  - etc.

並列化解析の対象要因の一例

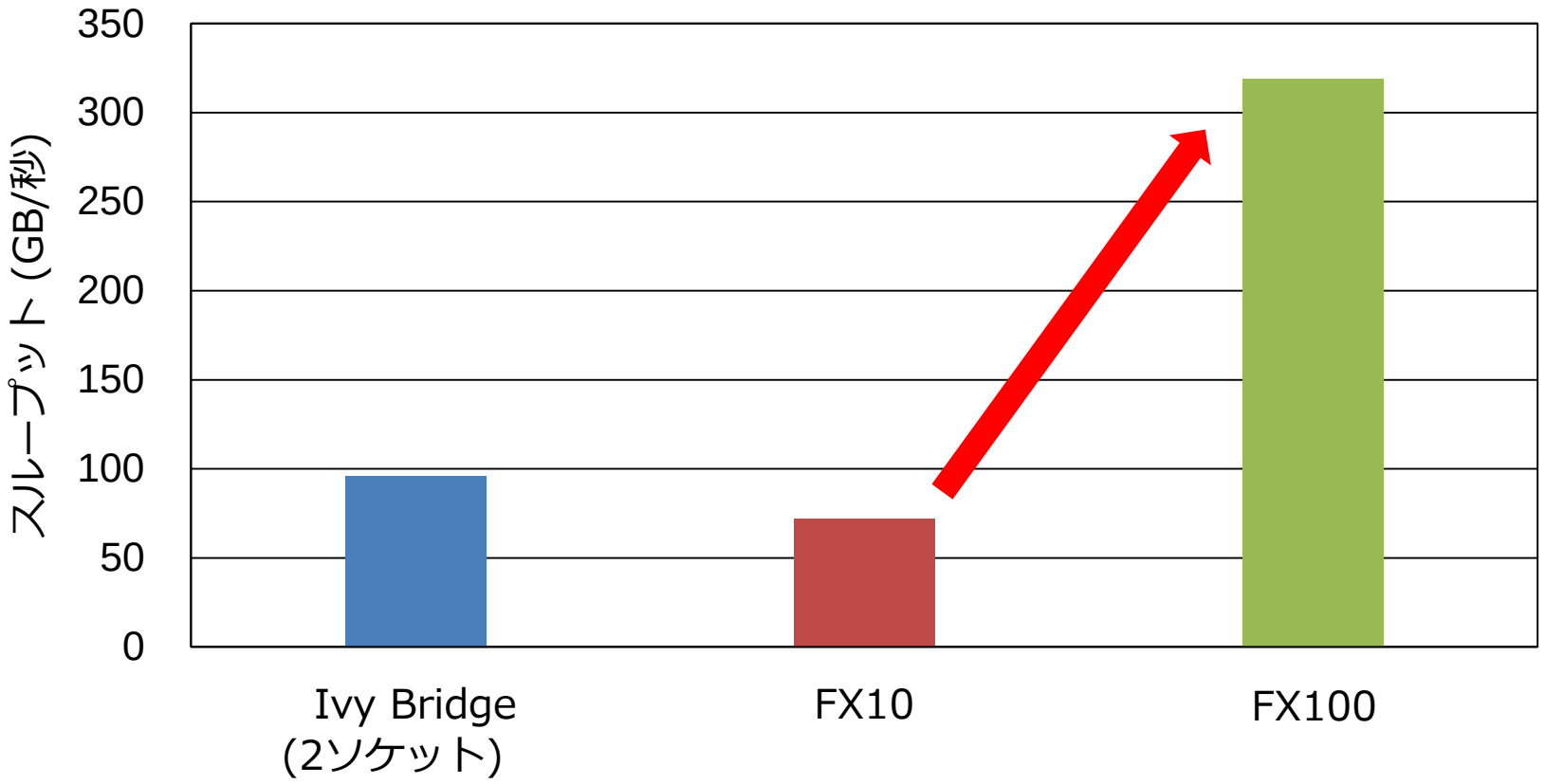
ループ内の演算	データ依存関係	その他
四則演算	依存なし	データ型
リダクション演算	順方向依存	対象ループ次元
収集・拡散	逆方向依存	粒度
ＤＯブランチ		分岐

ノード性能 (1)

FUJITSU

STREAM Triad

1ノード測定におけるスループット(GB/sec)の比較



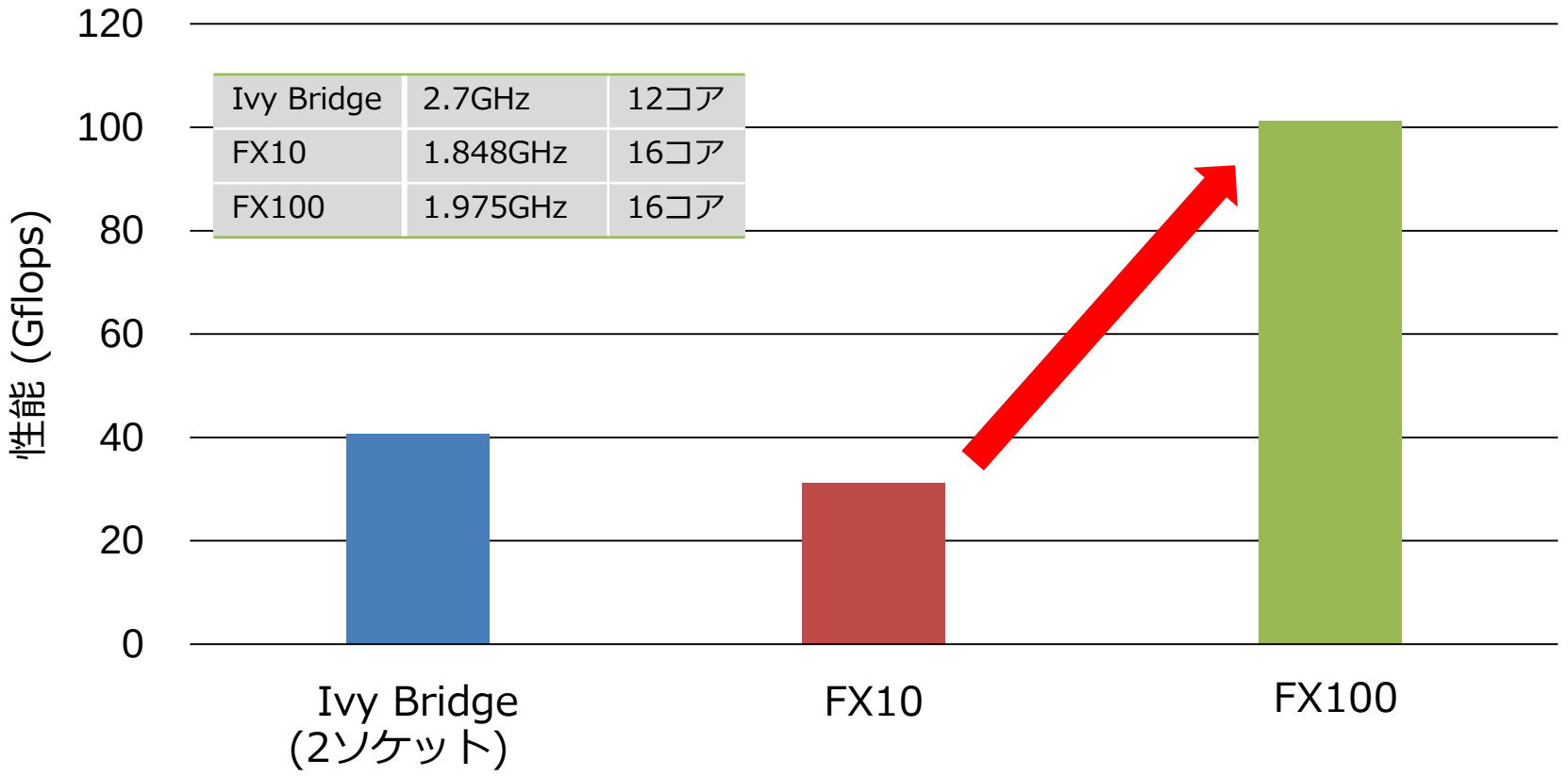
HMCの搭載によりFX10から約４倍の性能向上

ノード性能 (2)

FUJITSU

姫野ベンチマーク

1ノード測定における実行性能(GFlops)の比較



メモリネックな流体解析コードが性能向上

FUJITSU

SIMD (Single Instruction Multiple Data)

SIMD仕様 (1)

FUJITSU

- SIMD
  - １命令で複数の演算を実行
  - SIMDレジスタを利用し複数データを操作
  - コア内の命令の並列性を向上
- SIMD命令
  - SIMDレジスタを利用する命令
  - 演算命令およびメモリアクセス命令
- SIMD命令による高速化の適用方法
  - コンパイラによる最適化(SIMD化)
  - 組み込み関数によるプログラミング

```
fadd,s  f2  f4  f6
```

```
a(i)   = b(i)   + c(i)
a(i+1) = b(i+1) + c(i)
a(i+2) = b(i+2) + c(i)
a(i+3) = b(i+3) + c(i)
```

SIMD仕様 (2)

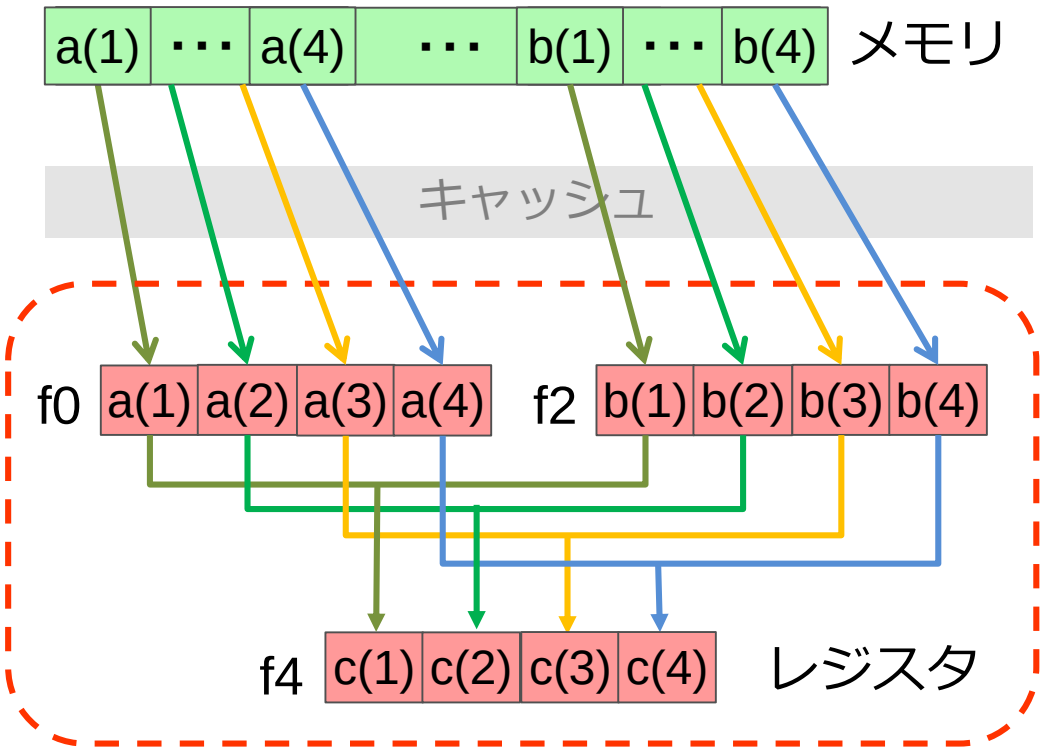
FUJITSU

- コンパイラによるSIMD化の原理
  - ループの回転方向や命令列の類似性を解析
  - 適用シーンに応じたSIMD命令を出力

```
do i=1,4
  c(i) = a(i) + b(i)
enddo
```

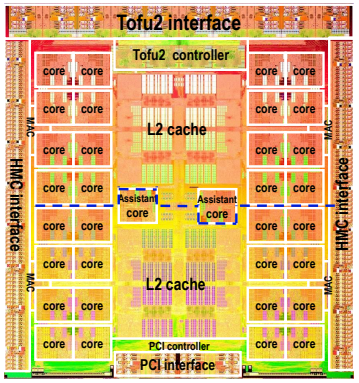
```
ldd,s   a(i:i+3) -> f0
ldd,s   b(i:i+3) -> f2
fadd,s  f0,f2 -> f4
std,s   f4 -> c(i:i+3)
```

=



SIMD仕様 (3)

- コンパイラによるSIMD化
  - VSIMD (Vectorize SIMD)
  - UXSIMD (Un-loop eXaminate SIMD)
  - PSIMD (Peephole SIMD)
- FX100におけるSIMD最適化
  - 256 bit wide SIMD
  - 積和演算をサポート
  - 最大８つの演算を同時処理
  - コア内で２つのSIMD演算命令を同時実行



SIMD仕様 (4)

FUJITSU

■機能比較

	京/FX10	FX100
倍精度実数 （演算／ロード／ストア）	2	2, 4
単精度実数 （演算／ロード／ストア）	2	2, 4, 8
整数 （演算／ロード／ストア）	–	2, 4
マスク付命令	実数	実数、 整数
データ境界（ロード）	型サイズ	型サイズ
データ境界（ストア）	型サイズ×2	型サイズ
非連続データアクセス命令 （ストライド）	–	ロード, ストア
非連続データアクセス命令 （インダイレクト）	–	ロード,ストア,プリフェッチ

SIMD技術 (1)

FUJITSU

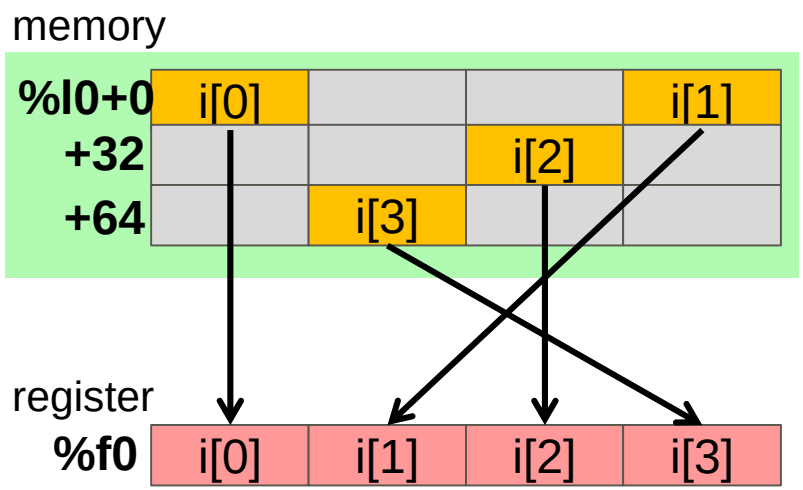
- 代表的な技術一覧
  - Stride Memory Access
  - Indirect Memory Access
  - Loop Division
  - Unaligned SIMD Store
  - Integer Condition Branch
  - New Complex-number Model
  - Concatenation Shift
  - Element Compress
  - Masking Loop SIMD

SIMD技術 (2)

Stride Memory Access

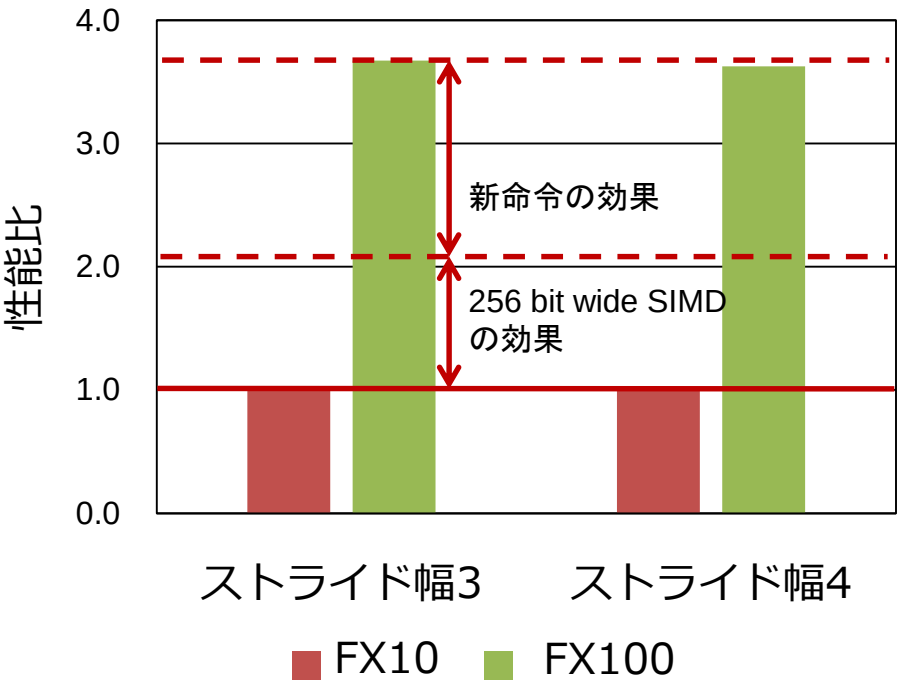
- ２から７のストライド幅に対して適用可能

ストライド幅3のロード命令



ldst,s [%l0]@stride 3, %f0

ストライドロード性能 (1コア)



連続体コードなどの間隔アクセスに対して性能効果

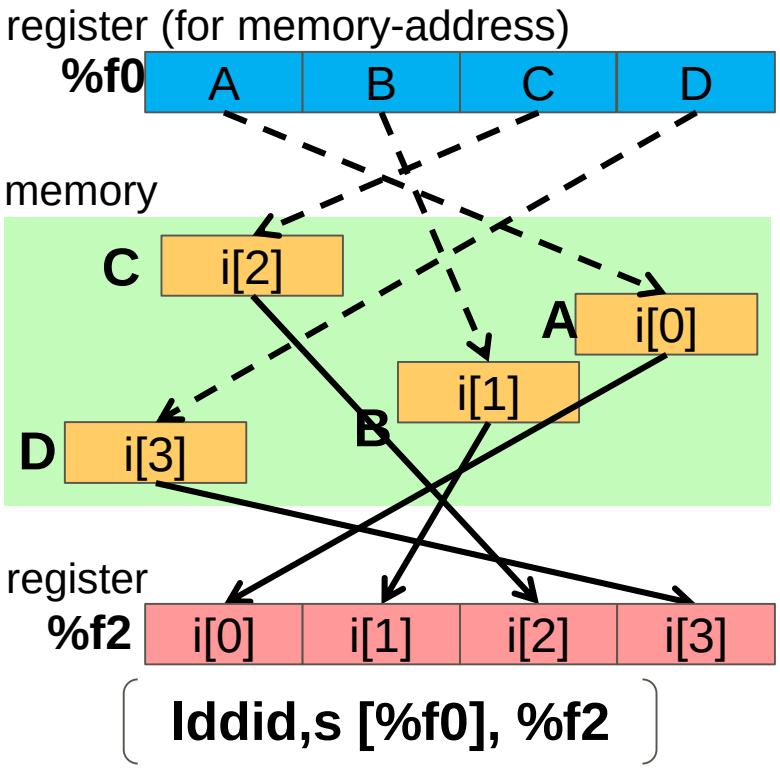
SIMD技術 (3)

FUJITSU

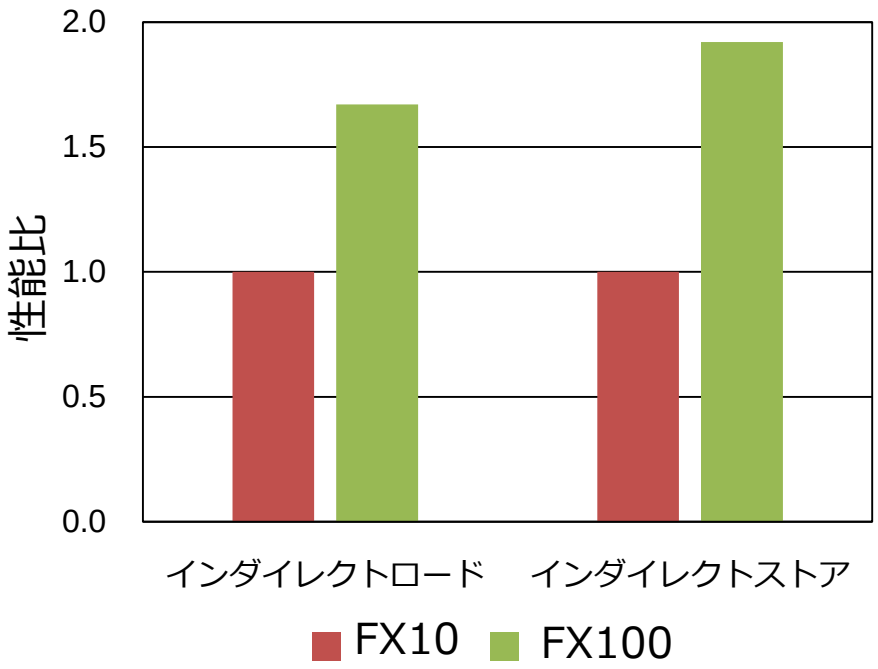
Indirect Memory Access

アドレス計算もSIMD命令で並列計算

インダイレクトロード命令



インダイレクトアクセス性能 (1コア)



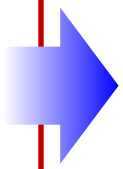
流体解析、FEMなどのリストアクセスに対して性能効果

SIMD技術 (4)

FUJITSU

- Loop Division
  - 単精度浮動小数点演算、倍精度浮動小数点演算の混在
  - 異なるSIMD幅となるループを分割

```
subroutine sub(a,b,c,d,e,f,n)
real(4),dimension(1:n) :: a,b,c
real(8),dimension(1:n) :: d,e,f
do i=1,n
  a(i) = b(i) + c(i)
  d(i) = e(i) + f(i)
enddo
end subroutine
```



8SIMD

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

4SIMD

```
do i=1,n
  d(i) = e(i) + f(i)
enddo
```

SIMD命令の適用率、スケジューリングの向上

SIMD技術 (5)

FUJITSU

- Unaligned SIMD Store
  - データの境界の違いに影響を受けないストア命令
  - 京/FX10で出力していた境界補正処理を削減

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

```
i=1
if (and(loc(a(1)),0xf) .ne. 0) goto 10
a(1) = b(1) + c(1)
i=2
10 conitnue
do i=i,n,2
  a(i:i+1) = b(i:i+1) + c(i:i+1)
enddo
if (i.ne.n) then
  a(n) = b(n) + c(n)
endif
```

境界補正処理を削減

コードサイズを削減し、命令キャッシュの効率化

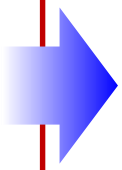
SIMD技術 (6)

FUJITSU

Integer Condition Branch

- 整数型の条件判定のIF構文を含むループをSIMD化
- 京/FX10では、浮動小数点型へ変換

```
real(8),dimension(1:n) :: a,b,c
integer(4),dimension(1:n) :: m
do i=1,n
  if (m(i) .gt. 0) then
    a(i) = b(i) + c(i)
  endif
enddo
```



FX100 整数型向け新命令

```
! 整数型SIMDロード
ldsw,s      m(i:i+3),%f32

! 整数型SIMD比較
fzero,s     %f34
fpcmpgtw,s  %f32,%f34,%f36
```

京/FX10での命令出力イメージ

```
if (real(m(i:i+1),kind=8) .gt. 0.0) then
    a(i:i+1)  = b(i:i+1) + c(i:i+1)
```

不要な変換命令の削減、スケジューリングの向上

SIMD技術 (7)

FUJITSU

New Complex-number Model

- Stride Memory Accessを利用した命令出力
- 京/FX10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  !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/FX10での命令数

全１１命令（ロード×３，ストア×２，演算×４，初期化×２）

命令数を削減し複素数型演算の高速化

SIMD技術 (8)

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(1:4)
do i=1, n-4, 4
  T2 = b(i+4:i+7)                          ※先行LOAD
  T3 = concatenate_shift(T1, T2, 1)  ※b(i+1:i+4)
  T4 = concatenate_shift(T1, T2, 2)  ※b(i+2:i+5)
  T5 = concatenate_shift(T1, T2, 3)  ※b(i+3:i+6)
  T6 = T1 + T3
  T7 = T4 + T5
  a(i:i+3) = T6 + T7
  T1 = T2
enddo
```

隣接アクセスのロード命令を削減

SIMD技術 (9)

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.D0)
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. 0D0
  VTD = B(I:I+3)
  CVTD=fecpd(VTD,MASK)
  A(J:J+3)=CVTD
  J=J+INT(fesummd(MASK),KIND=4)
ENDDO
```

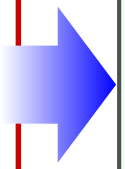
配列圧縮コードなどを高速化

SIMD技術 (10)

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
  faddd,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}
```

回転数が少ないループに対する高速化

FUJITSU

システム性能

システム性能を支える技術

FUJITSU

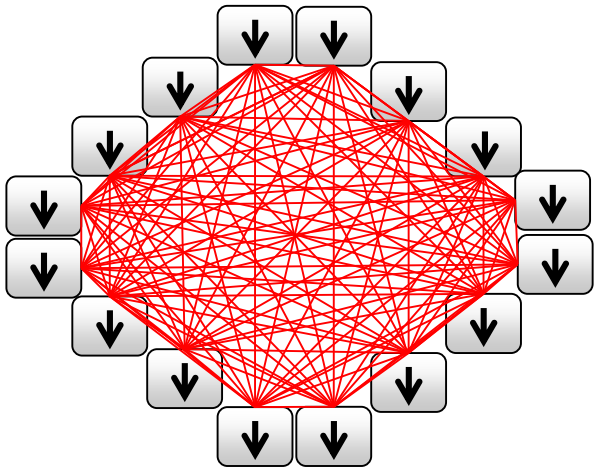
- VISIMPACT (Virtual Single Processor by Integrated Multi-core Parallel Architecture)
  - ハードウェアバリア
  - ハイブリッド並列
- Tofu2
  - リンクバンド幅の強化
  - キャッシュインジェクション機能
- MPIライブラリ
  - アトミック通信用の拡張インターフェース
  - アシスタントコア利用による通信のオーバラップ

FUJITSU

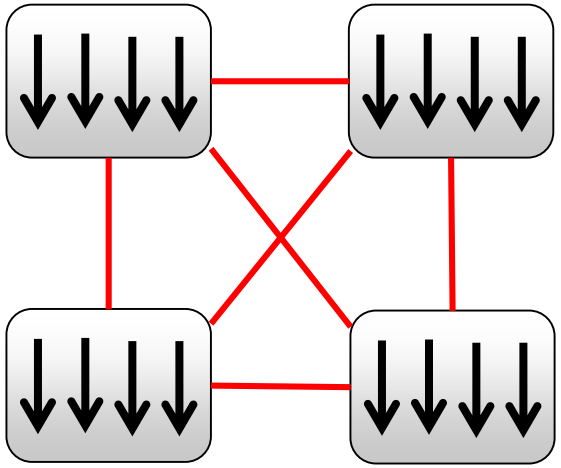
ハイブリッド並列

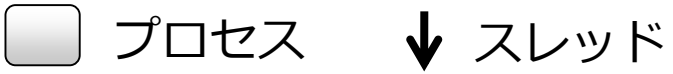
- マルチコアCPU内のハードウェアバリア機能
- コンパイラの高度な自動並列化機能

プロセス並列



ハイブリッド並列





高効率なハイブリッド並列を実現

Tofu2

FUJITSU

- リンクバンド幅の強化
  - 12.5 GB/s：25.78125 Gbps×４レーン
  - 伝送のロスをなくすため、本体間は光伝送を採用
- キャッシュインジェクション機能
  - 受信データをL2キャッシュメモリに直接書き込み
  - メモリ転送と比較し、0.16μs削減

通信種別	通信遅延
Put (to Memory)	0.87 μs
Put (to Cache)	0.71 μs

通信性能のさらなる向上

MPIライブラリ (1)

FUJITSU

- アトミック通信用の拡張インターフェース
  - Atomic Read Modify Write(ARMW)通信
  - 宛先ノードの４バイト／８バイトデータに対して演算
  - RDMA通信のレベルでメモリの原子性を保証

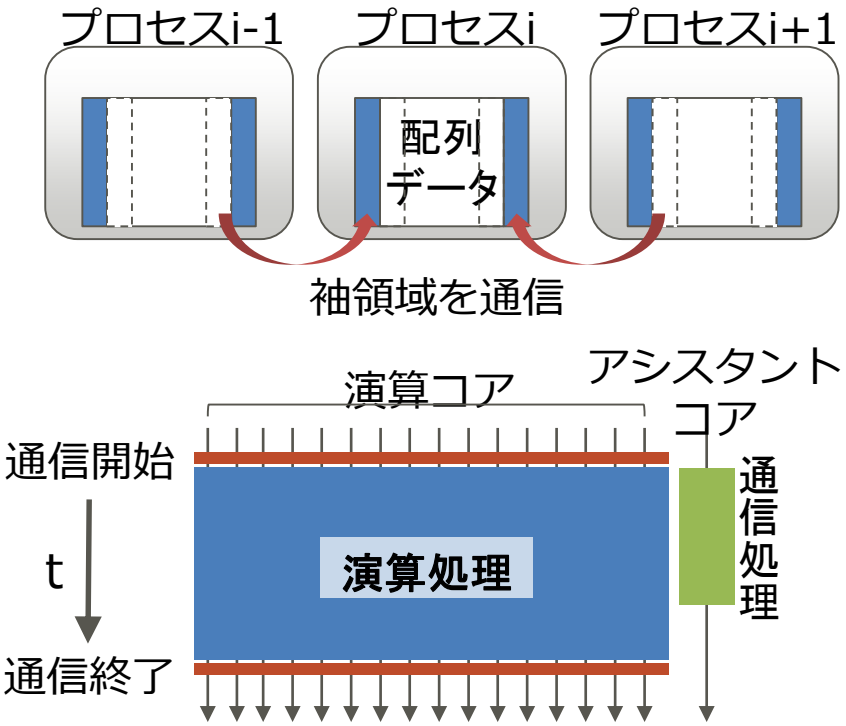
演算の種類	内容
Compare and Swap	データとオペランドを比較し置き換え
Swap	無条件に置き換え
Fetch and Add	データとオペランドの加算結果を置き換え
XOR	データとオペランドの排他的論理和を置き換え
AND	データとオペランドの論理積を置き換え
OR	データとオペランドの論理和を置き換え

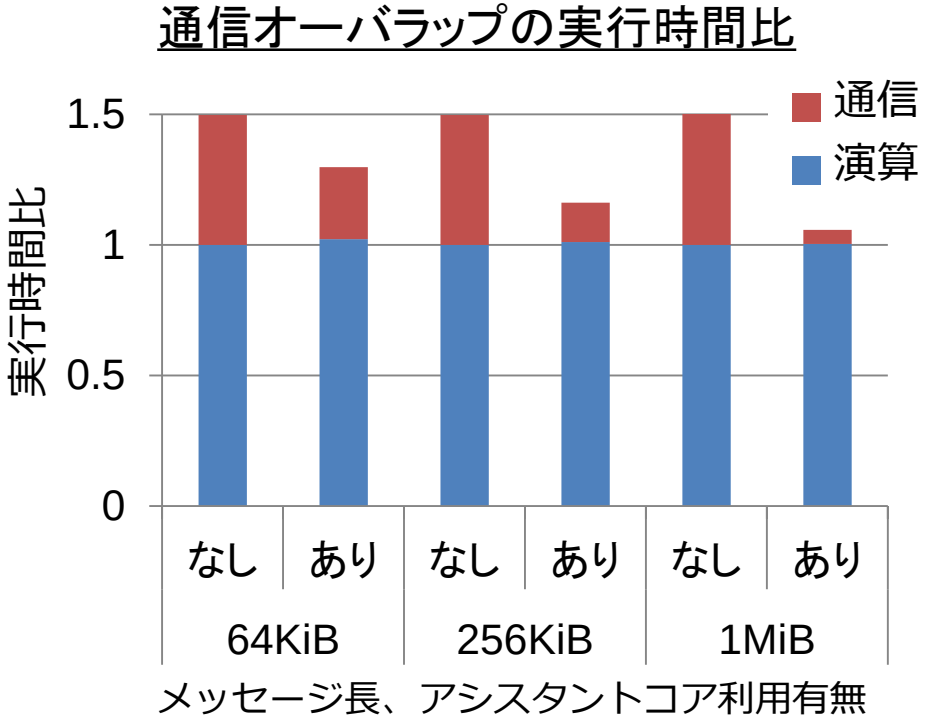
排他制御のオーバーヘッドを削減

MPIライブラリ (2)

FUJITSU

- アシスタントコア利用による通信のオーバラップ
  - １ノード内に２つあるアシスタントコアで通信の並列処理
  - ステンシル計算などの袖領域通信に有効





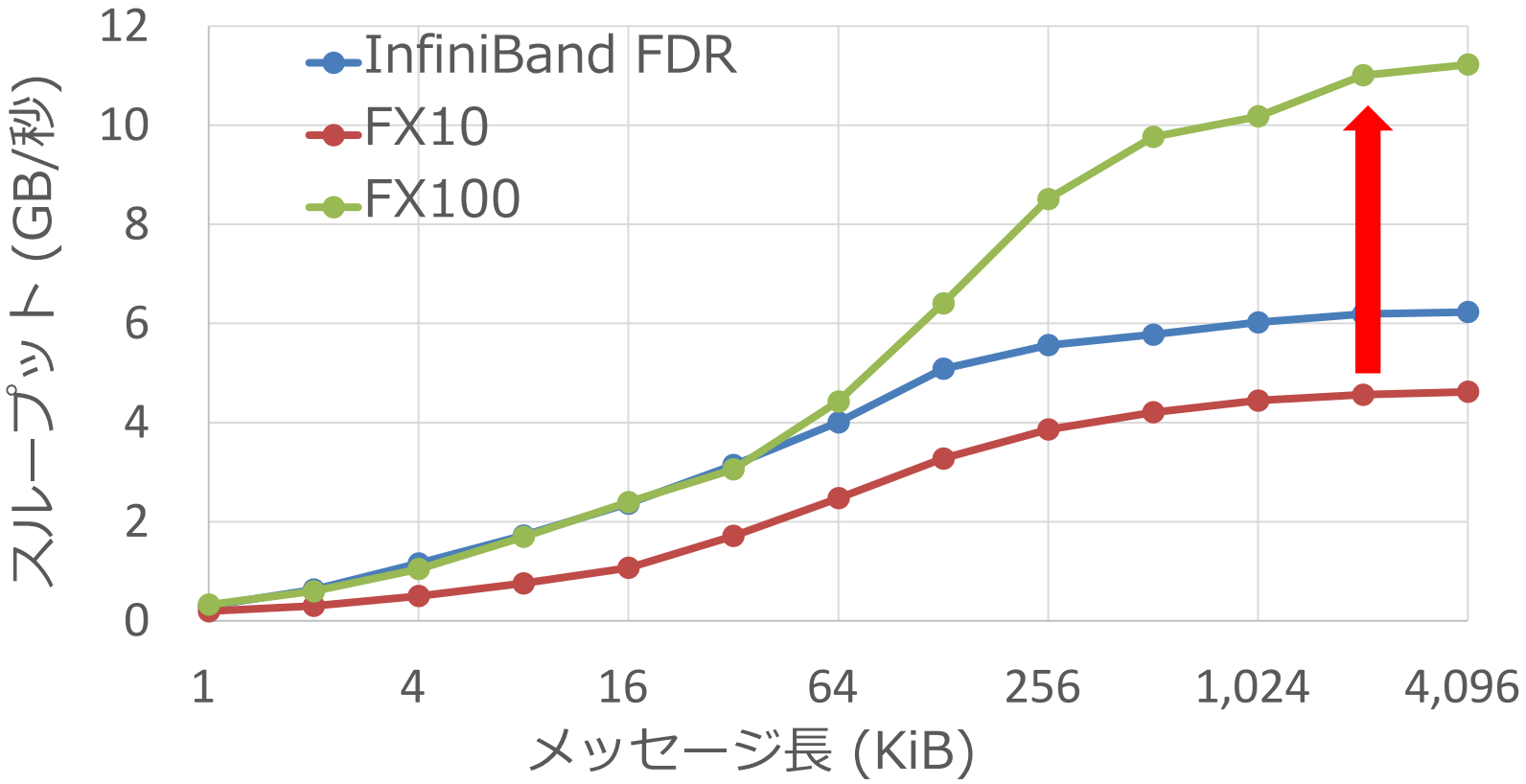
アシスタントコアを活用し通信時間を削減

システム性能 (1)

FUJITSU

■一対一通信性能

■２ノード間のメッセージ長／スループットの性能変化



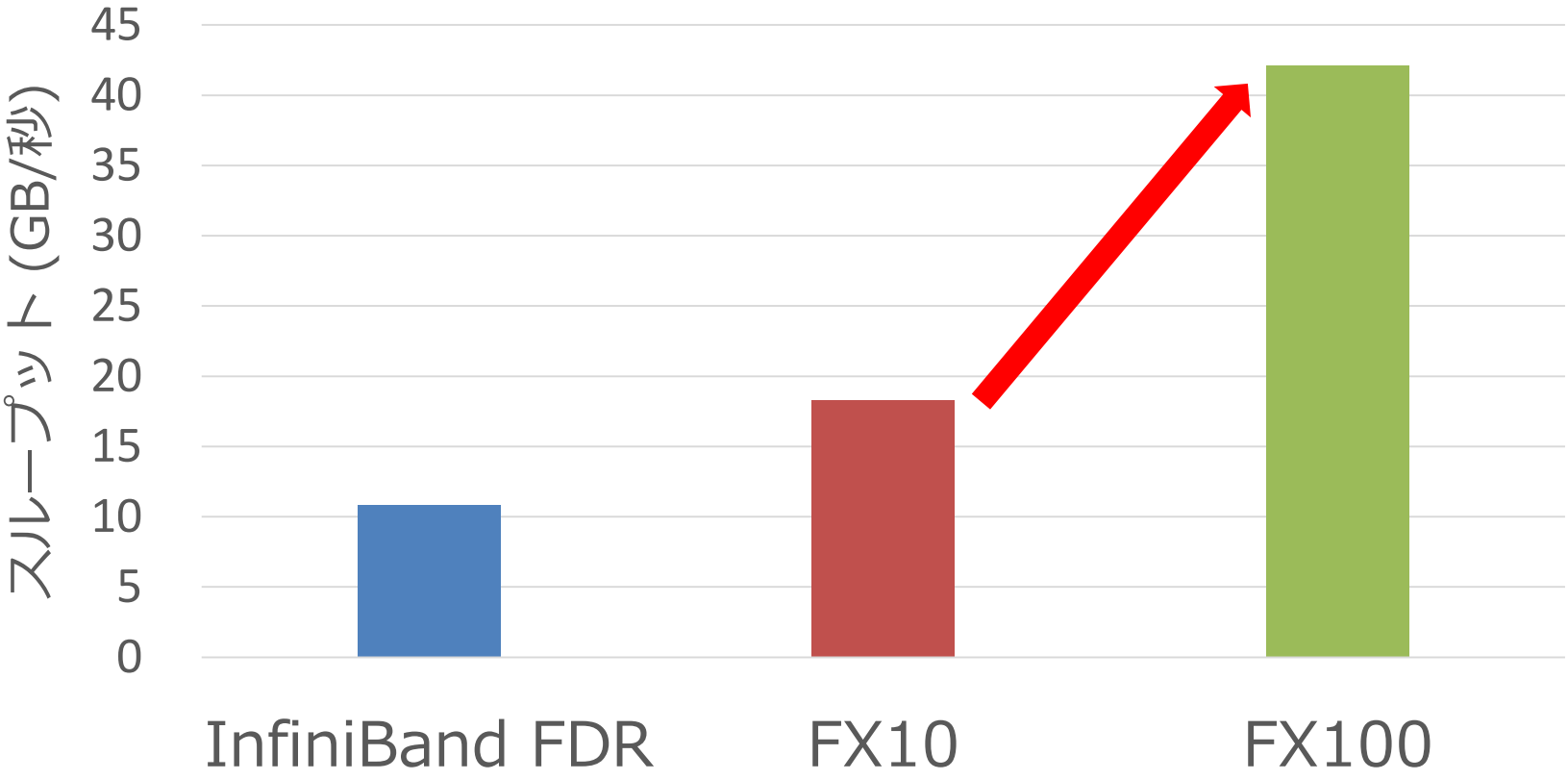
FX10から約2.４倍の性能向上

システム性能 (2)

FUJITSU

二方向同時通信性能

メッセージ長を4 MiBに固定した2方向同時通信の性能

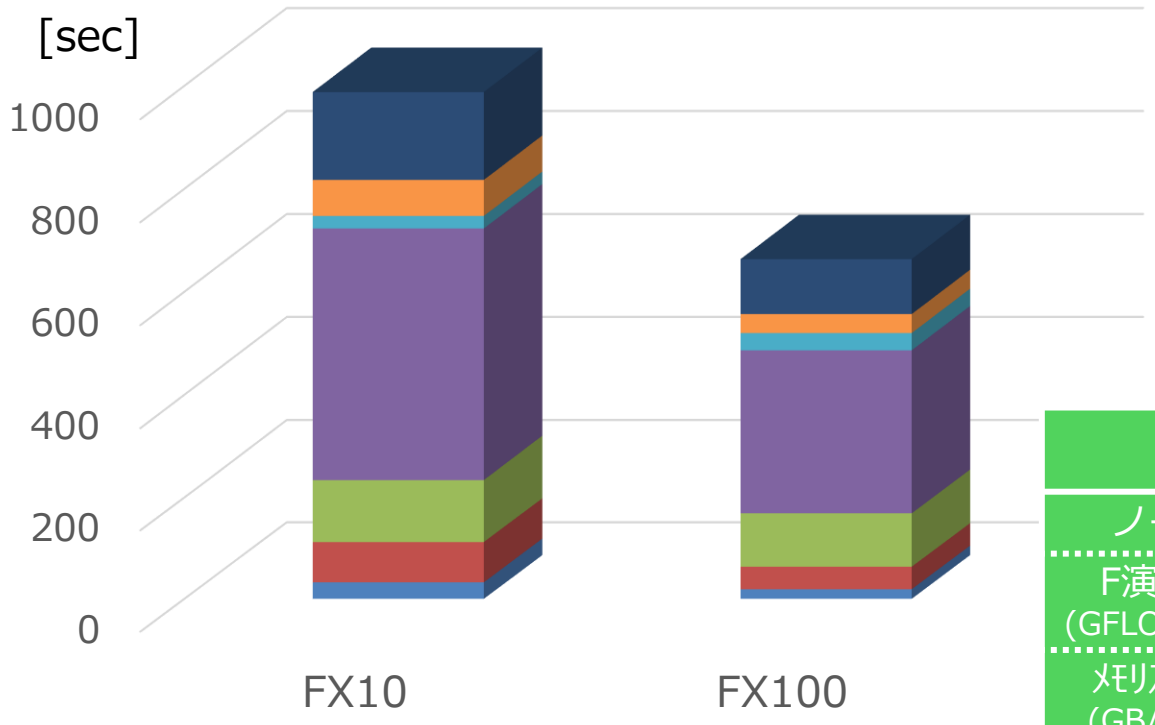


多方向同時通信に対しても性能向上を期待

システム性能 (3)

FUJITSU

- NICAM-DC-MINI
  - 高コストルーチン「vi_path2」
  - MPIプロセス数を同等にした場合のFX10性能比較

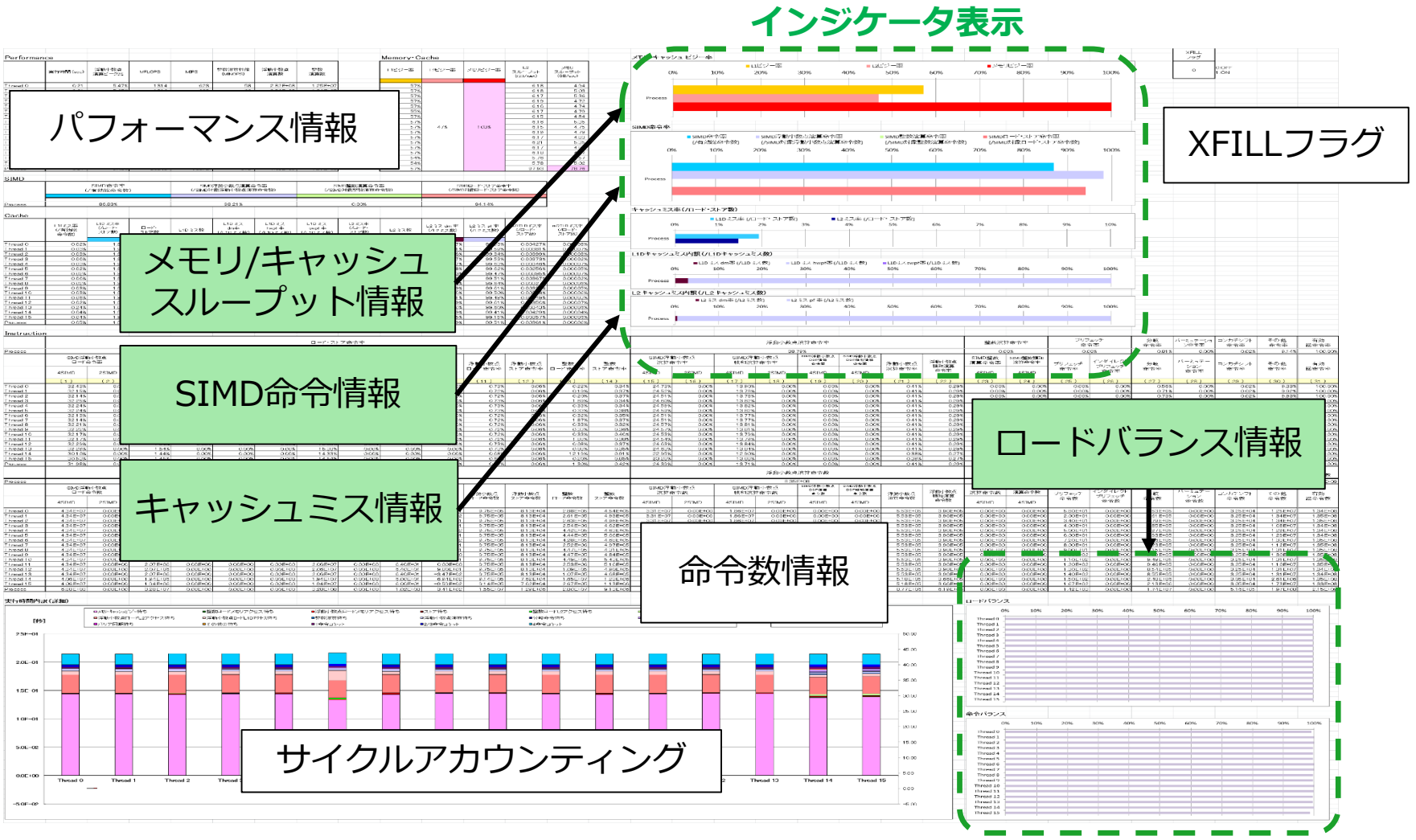


	FX10	FX100
ノード数	8	4
F演算性能 (GFLOPS/node)	14.60	43.81
ﾒﾓﾘｽﾙｰﾌﾟｯﾄ (GB/s/node)	74.2	232.1

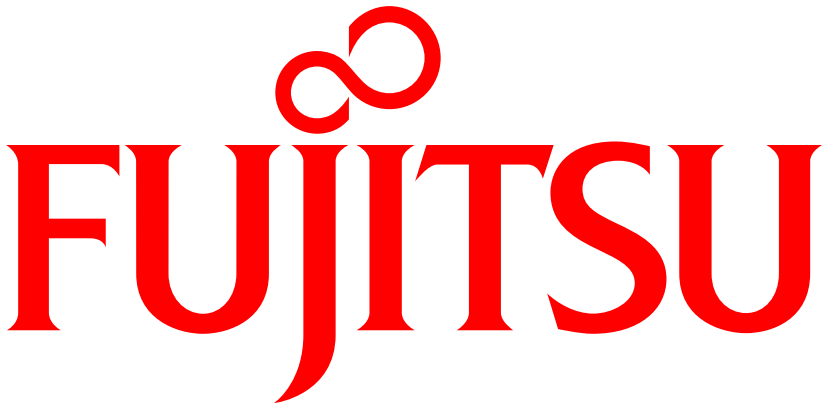
半分のノード数で 1.5倍性能を実現

サポート機能

PA(Performance Analysis)情報レポート



問題特定につながる情報量をアップ



shaping tomorrow with you