

PRIMEHPC FX10後継機における 性能と評価

2014年10月29日

富士通株式会社

千葉 修一

- PRIMEHPC FX10後継機 (Post-FX10)
- ノード性能の評価
 - 1コア性能
 - SIMD性能
 - スレッド並列性能
 - アプリケーション性能
- まとめ



Post-FX10

■ 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

システムアーキの継承

Tofuの強化

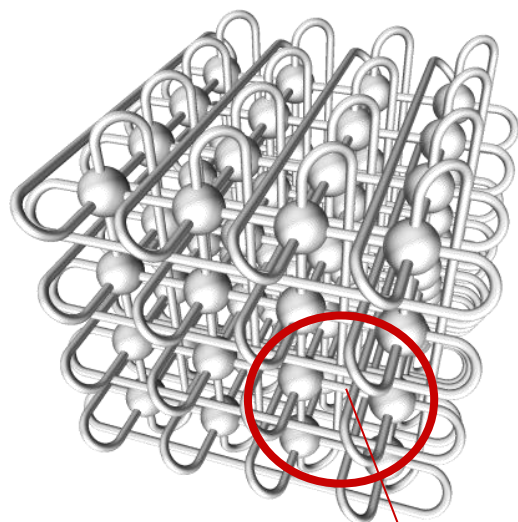
CPUコアの強化

コンパイラの強化

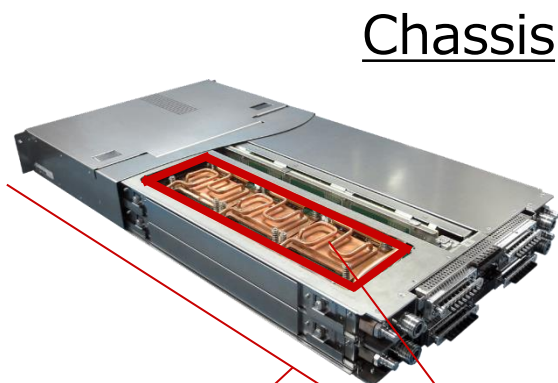


Post-FX10

■ハードウェア構成

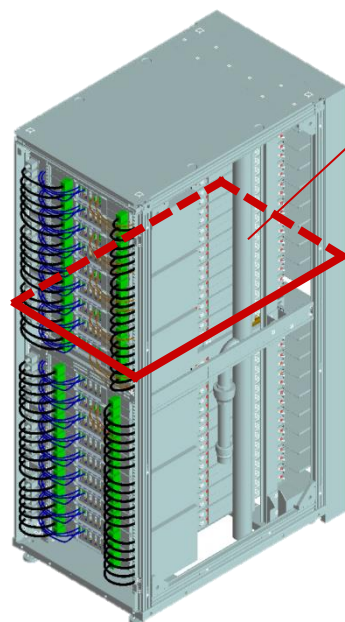


Tofu2

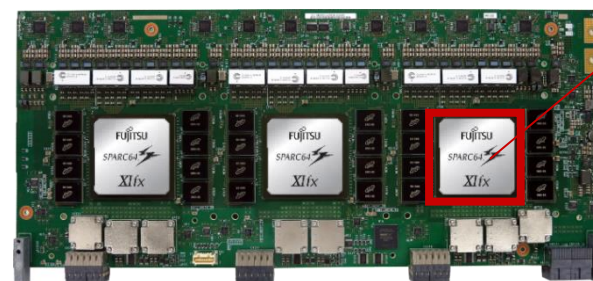


Chassis

SPARC64™ XIfx



Rack

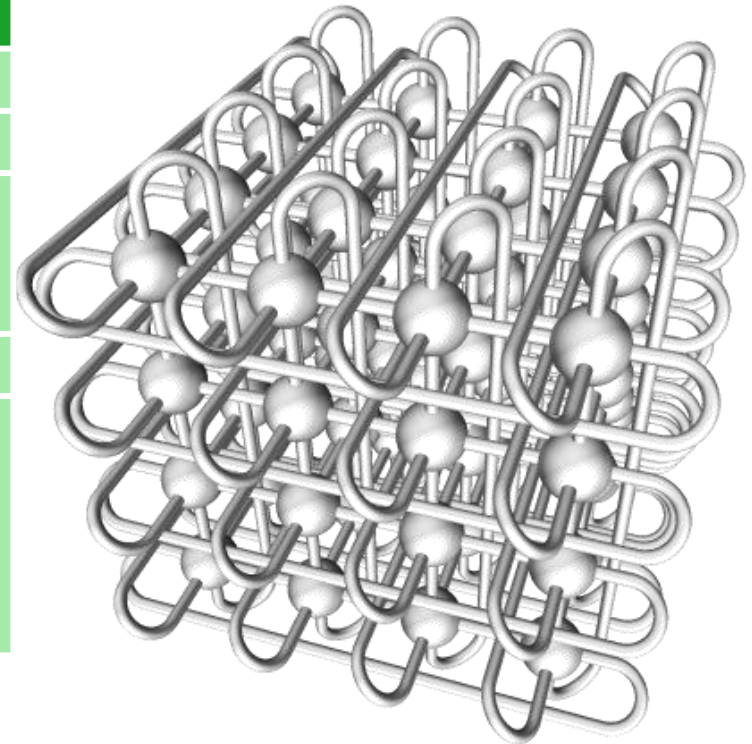


CPU Memory Board

■ 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	Post-FX10
CPUとの関係	別LSI (ICC)	内蔵
トポロジ	6次元メッシュ/トラス	←
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
新機能	-	キャッシュ インジェクション アトミック シャード間接続 (全体の2/3)を光化



■ Rack

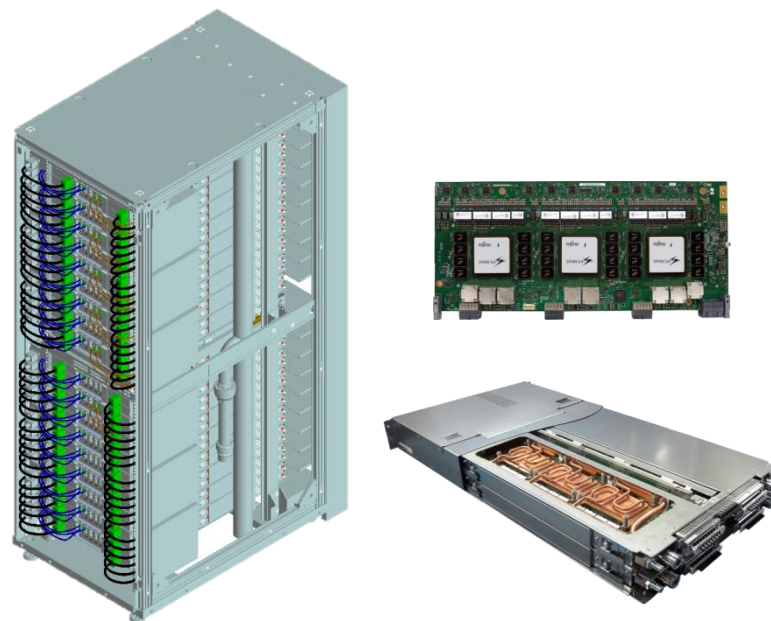
- 216ノード / キャビネット
- CPU、メモリ、光モジュールを直接水冷(水冷率90%)

■ 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



■ 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

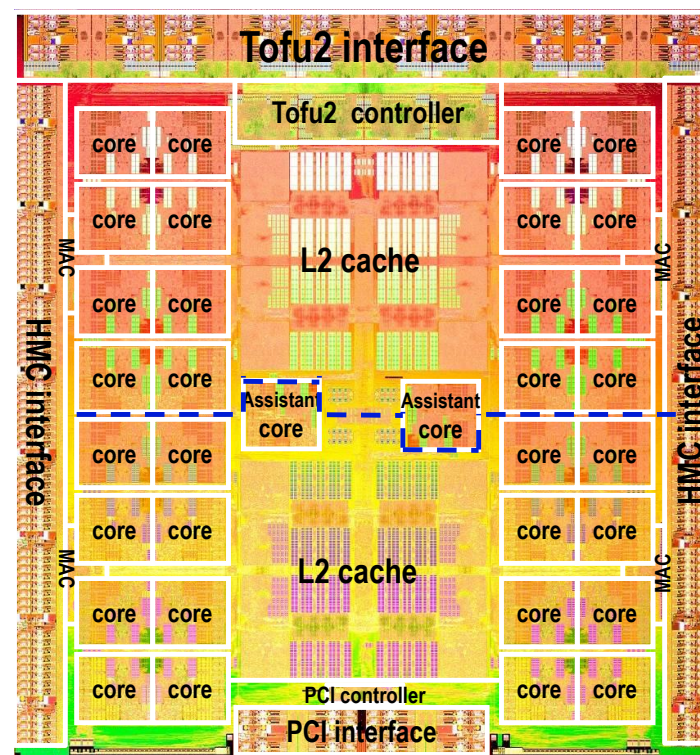
- アウトオブオーダー資源の増加
- 分岐予測の強化

■ 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
 - OSノイズの低減
 - 演算と通信のオーバラップ化



■ SPARC64™ XIfx

	京	FX10	Post-FX10
アーキテクチャ	SPARC64 VIIIIfx	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

ノード性能を支える技術

■ 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

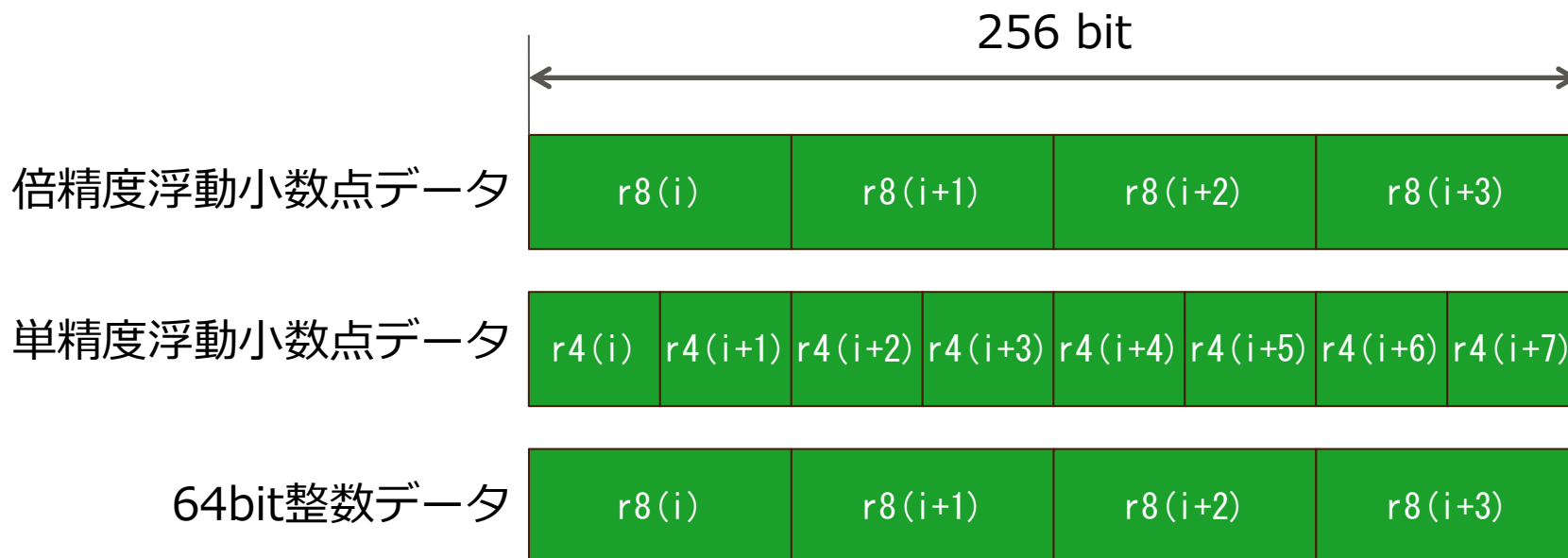
これらの課題を改善する機能を開発

- **HPC-ACE2** (High Performance Computing - Arithmetic Computational Extensions 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

各改善が連動することで最大限の性能を引き出す

■ 256 bit wide SIMD

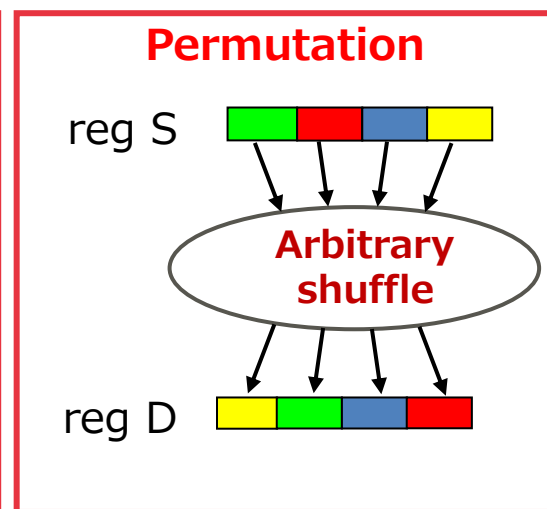
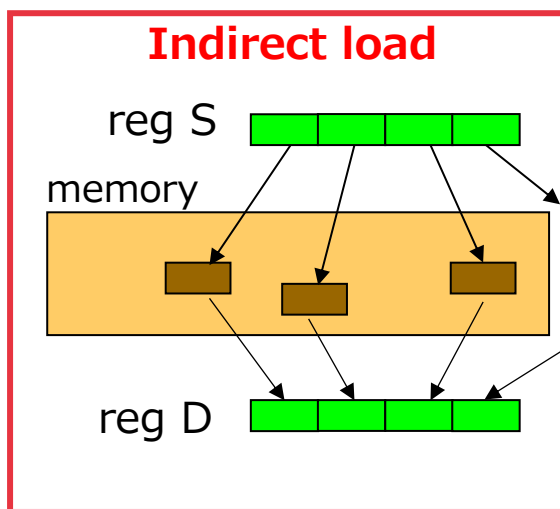
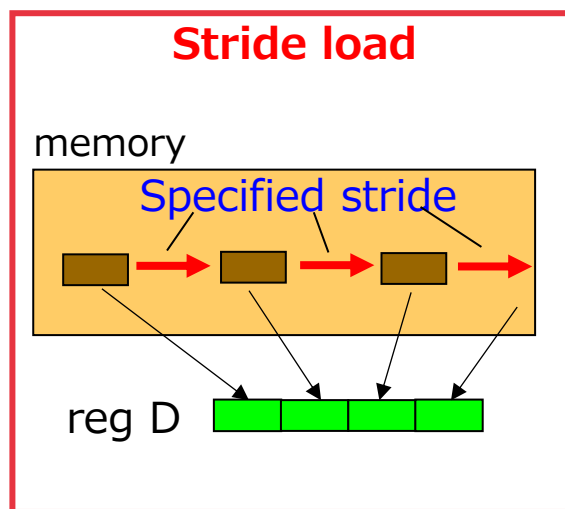
- 倍精度浮動小数点データ x 4
- 単精度浮動小数点データ x 8
- 64bit 整数データ x 4



FX10に比べ、倍精度 2 倍・単精度 4 倍のSIMD幅を実現

■ HPC-ACE(FX10)からの拡張

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



多種のカーネルに対してコンパイラがSIMD化を適用可能

■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	Post-FX10
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)

キャッシュサイズ & WAY数は2倍、スループットは大幅に増加

■ コンパイラの最適化を強化

■ クローニング

ループを複製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

回転数の少ないループへソフトウェアパイプラインの適用

■ 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

■ etc.

各種アプリケーションに対する最適化の適用率をアップ

■ 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

VPPの技術をベースに解析能力を強化

ノード性能の評価

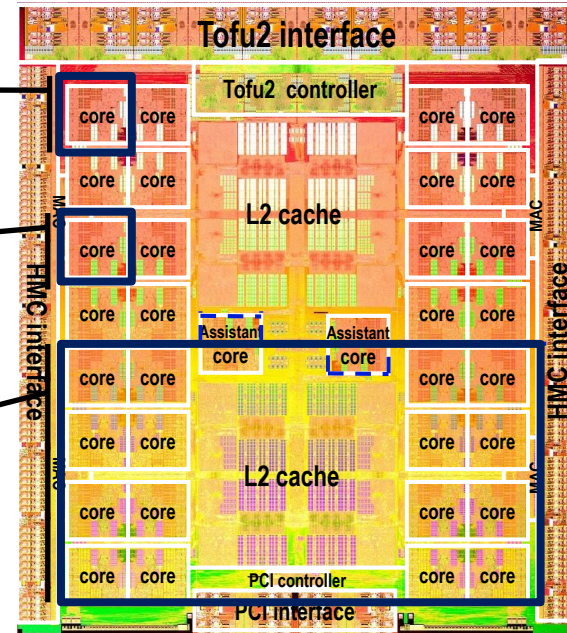
■ 4つの観点で評価

1コア性能

SIMD性能

スレッド並列化性能

アプリケーション性能

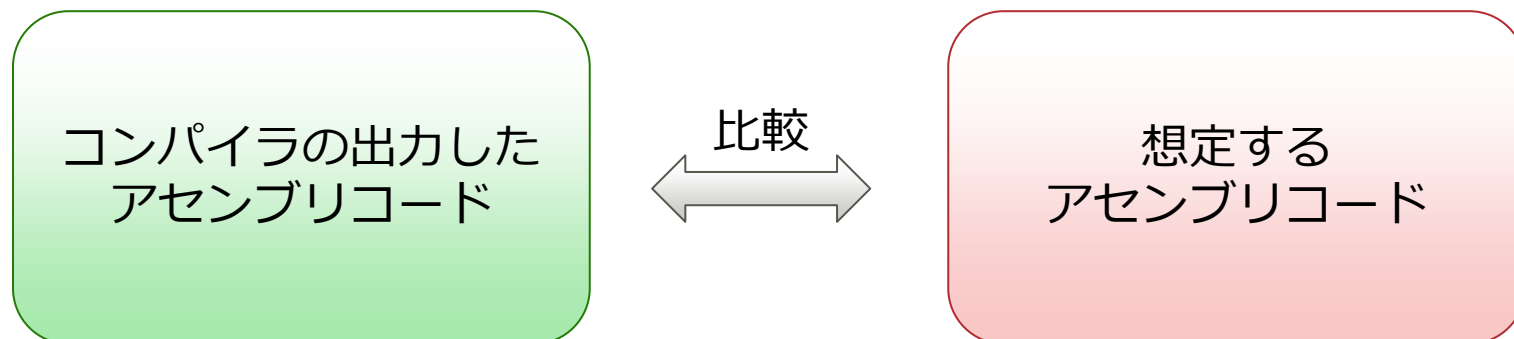


1 コア性能

■ ABCMarks

素性能を評価する目的で自社開発したカーネル群
(EuroBen Benchmarkライクなコード)

- 連続アクセス系カーネル
- 不連続アクセス系カーネル
- マスク系カーネル



コンパイラがHPC-ACE2を使い切れているかを評価

■ 連続アクセス系カーネル

ストリーム、DAXPYを含む基本カーネル

```
do i = 1,n  
  y(i) = c  
enddo
```

0 Load 1 Store

```
do i = 1,n  
  y(i) = x1(i)  
enddo
```

1 Load 1 Store

```
do i = 1,n  
  y(i) = x1(i) + x2(i)  
enddo
```

2 Load 1 Store

```
do i = 1,n  
  s = s + x1(i) * x2(i)  
enddo
```

Dot product

```
do i = 1,n  
  y(i) = y(i) + c * x1(i)  
enddo
```

DAXPY

```
do i = 1,n  
  cy(i) = cx1(i) * cx2(i)  
enddo
```

Complex Multiplication

1 コア性能 (3)

■ 不連続アクセス系カーネル

ストライド、インダイレクトアクセスを含む基本カーネル

```
do i = 1,3*n,3  
  y(i) = x1(i) * x2(i)  
enddo
```

Stride 3

```
do i = 1,4*n,4  
  y(i) = x1(i) * x2(i)  
enddo
```

Stride 4

```
do i = 1,n  
  z(1,i) = c1 * z(13,i)  
          + c2 * z(7,i)  
          + c3 * z(5,i)  
enddo
```

Long Stride

```
do i = 1,n  
  y(ind(i)) = x1(i)  
enddo
```

Scatter

```
do i = 1,n  
  y(i) = x1(ind(i))  
enddo
```

Gather

```
do i = 2,n  
  y(i) = x1(i) + y(i-1)  
enddo
```

Recursion

■ マスク系カーネル

整数型の判定、浮動小数点型の判定を含む基本カーネル

```
do i = 1,n
  if(real(i))
    y(i) = y(i) + c0*x1(i)
  enddo
```

rMask DAXPY 100※

```
do i = 1,n
  if(real(i))
    y(i) = y(i) + c0*x1(i)
  enddo
```

rMask DAXPY 50※

```
do i = 1,n
  if(int(i))
    y(i) = y(i) + c0*x1(i)
  enddo
```

iMask DAXPY 100※

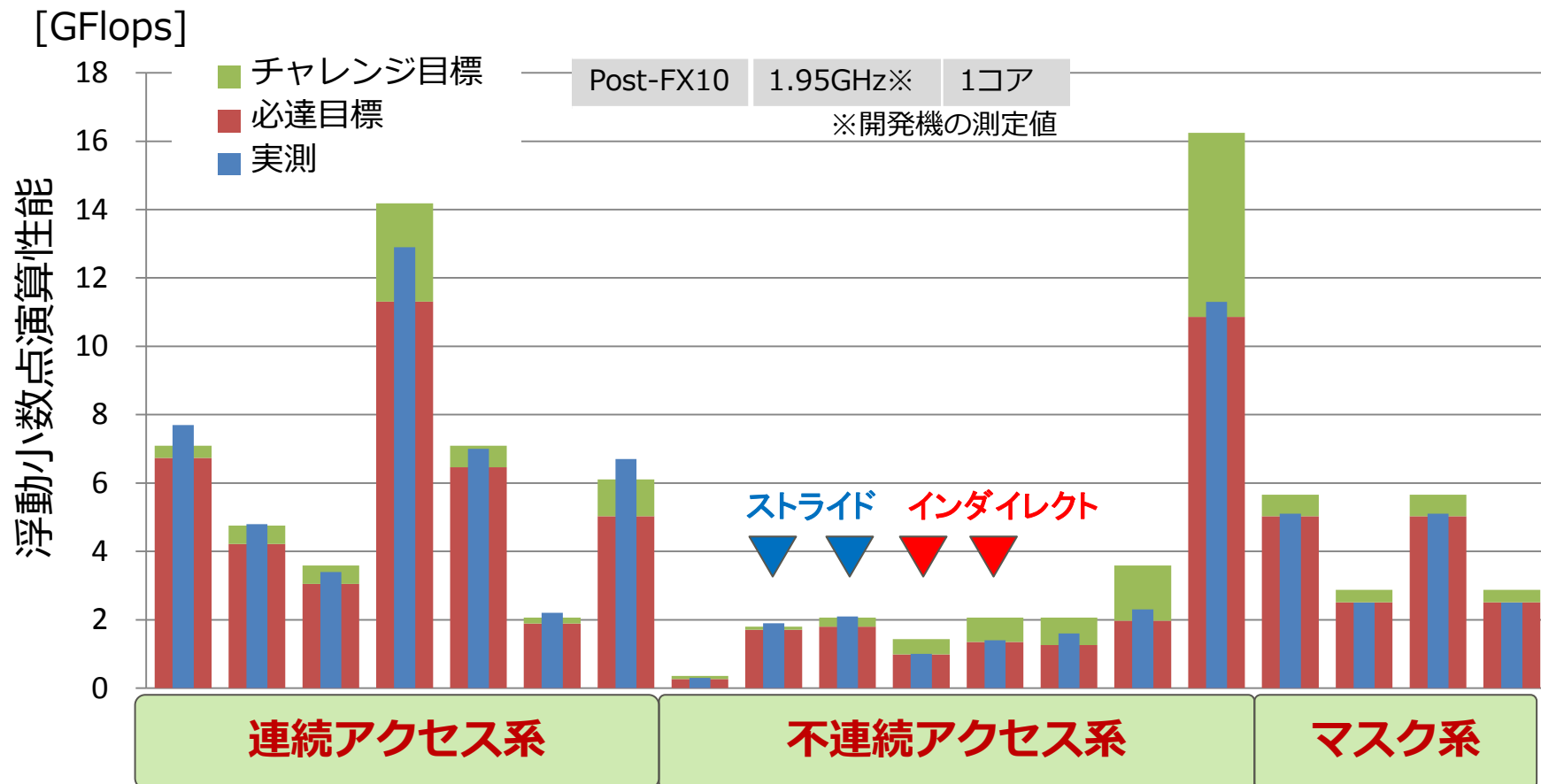
```
do i = 1,n
  if(int(i))
    y(i) = y(i) + c0*x1(i)
  enddo
```

iMask DAXPY 50※

※ 数値は、if構文の真率

■ ABCMarks

連続アクセス系はほぼ想定どおり、不連続アクセス系、マスク系を改善中



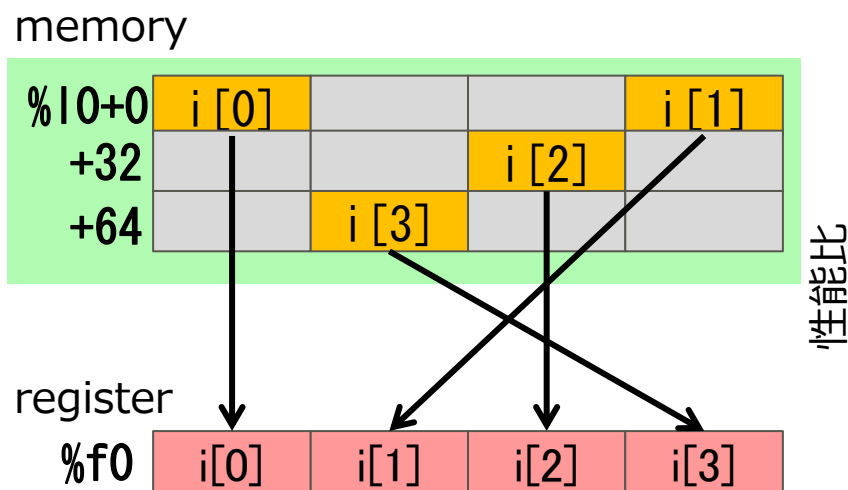
チャレンジ目標に向けてコンパイラを改善中

1 コア性能 (6)

■ ストライドアクセス

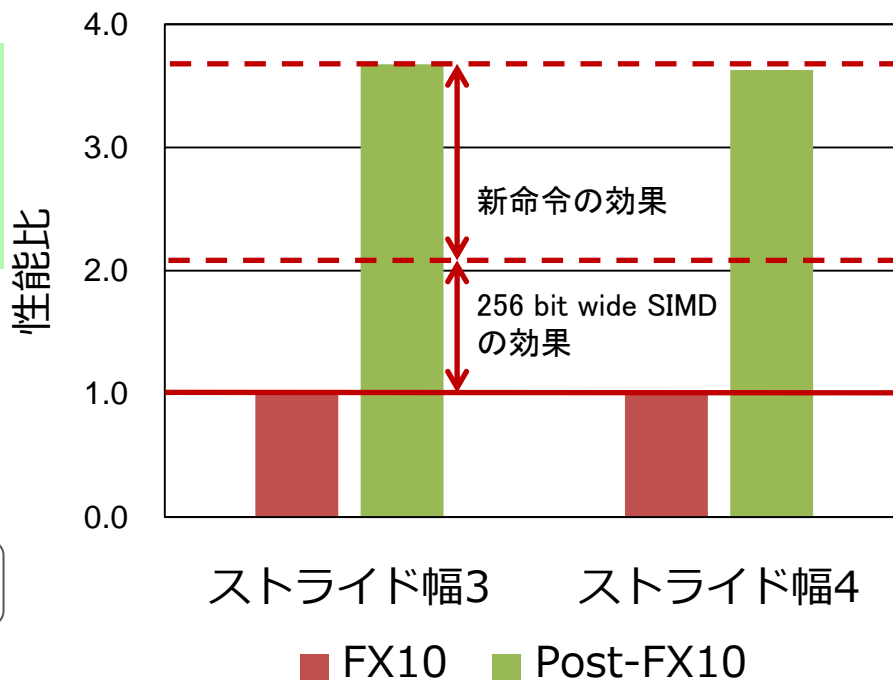
- 2 から 7 のストライド幅に対して適用可能

ストライド幅3のロード命令



```
[ lddst, s [%10]@stride 3, %f0 ]
```

ストライドロード性能 (1コア)

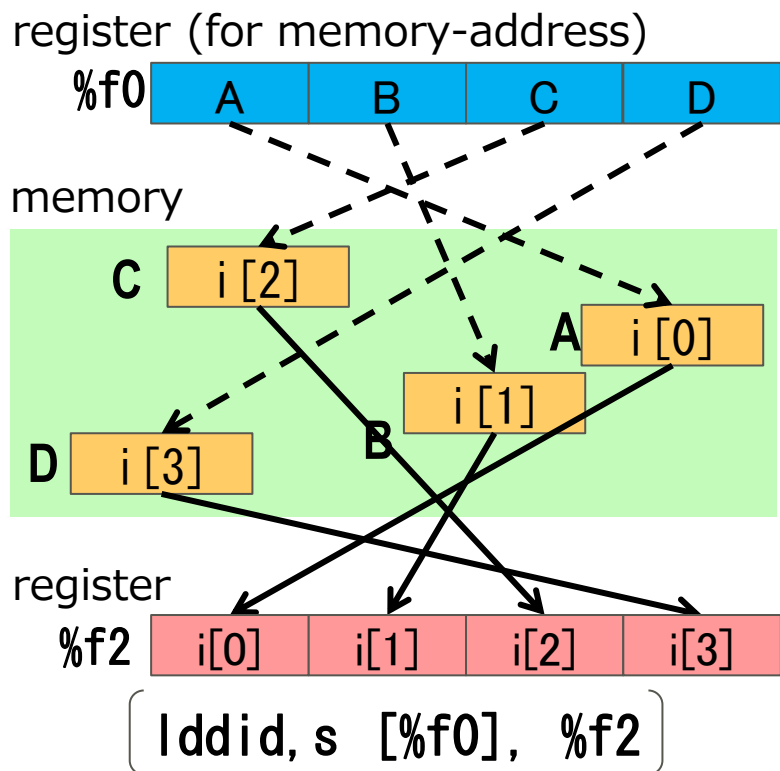


連続体コードなどの間隔アクセスに対して性能効果

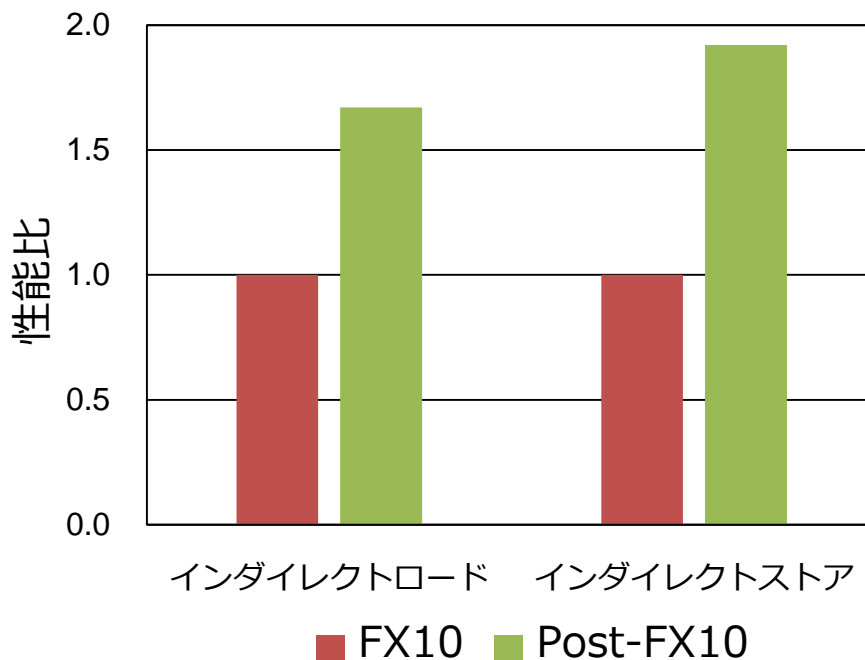
■ インダイレクトアクセス

■ 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)

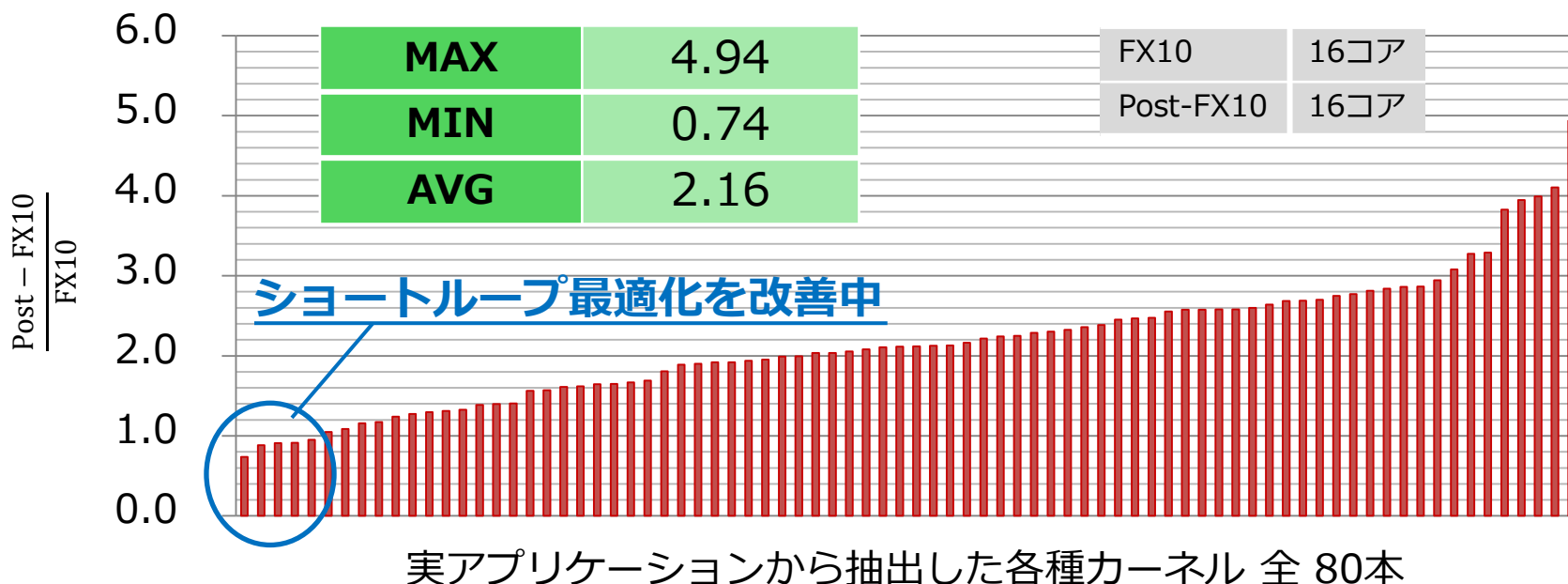


流体解析、FEMなどのリストアクセスに対して性能効果

SIMD性能

■ 評価カーネル (実アプリケーションから抽出)

- FX10 vs Post-FX10の実行性能を比較
- 周波数換算を行い、256 bit SIMD効果を可視化
- FX10性能を"1.0"とした場合の比率を昇順にソート



平均2倍の性能向上し、256 bit SIMDは効果大

■ ステンシルコードの高速化

■ 通常のSIMD化コード

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



通常の4 wide SIMD変換

```
do i=1, n, 4
  a(i:i+3) = b(i:i+3) + b(i+1:i+4) + b(i+2:i+5) + b(i+3:i+6) ! SIMD
enddo
```

- ループ内に4つのSIMDロード命令を出力
b(i:i+3)、 b(i+1:i+4)、 b(i+2:i+5)、 b(i+3:i+6)
- この時、それぞれのロード命令に同じ要素が含まれる
- 冗長のロードがL1キャッシュを圧迫

■ ステンシルコードの高速化

■ コンカチネーションシフト命令の利用コード

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



コンカチネーションシフト命令を利用し
ロード命令を削減

```
T1 = b(1:4)
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命令を変換
※b(i+2:i+5)のLOAD命令を変換
※b(i+3:i+6)のLOAD命令を変換

- ループ内は1つのSIMDロード命令のみ出力
b(i+4:i+7)

■ ステンシルコードに対するロードアクセスの削減

■ コンカチネーションシフト命令の適用イメージ

T1 = **LOAD**

b(i)	b(i+1)	b(i+2)	b(i+3)
------	--------	--------	--------

do i = 1, n, 4

T2 = **LOAD**

b(i+4)	b(i+5)	b(i+6)	b(i+7)
--------	--------	--------	--------

T3 = **CSIFT**

b(i+1)	b(i+2)	b(i+3)	b(i+4)
--------	--------	--------	--------

T4 = **CSIFT**

b(i+2)	b(i+3)	b(i+4)	b(i+5)
--------	--------	--------	--------

T5 = **CSIFT**

b(i+3)	b(i+4)	b(i+5)	b(i+7)
--------	--------	--------	--------

~

T1 = T2

enddo

コンカチネーションシフト命令によりロード命令が削減

■ ハンドチューニング

■ 気象系コードへの適用事例

```
real (8), allocatable, save :: b (:, :, :, :)  
  
do l=1, lmax  
!$omp do  
  do k=1, kmax  
    do n=nstart, nend  
      ij=n  
      q(n, k, l)=(  
        +b(0, ij, l, 1)*vx(ij, k, l) &  
        +b(1, ij, l, 1)*vx(ij, k, l) &  
        :  
        +b(6, ij, l, 3)*vz(ij3, k, l) &  
      )*fact
```

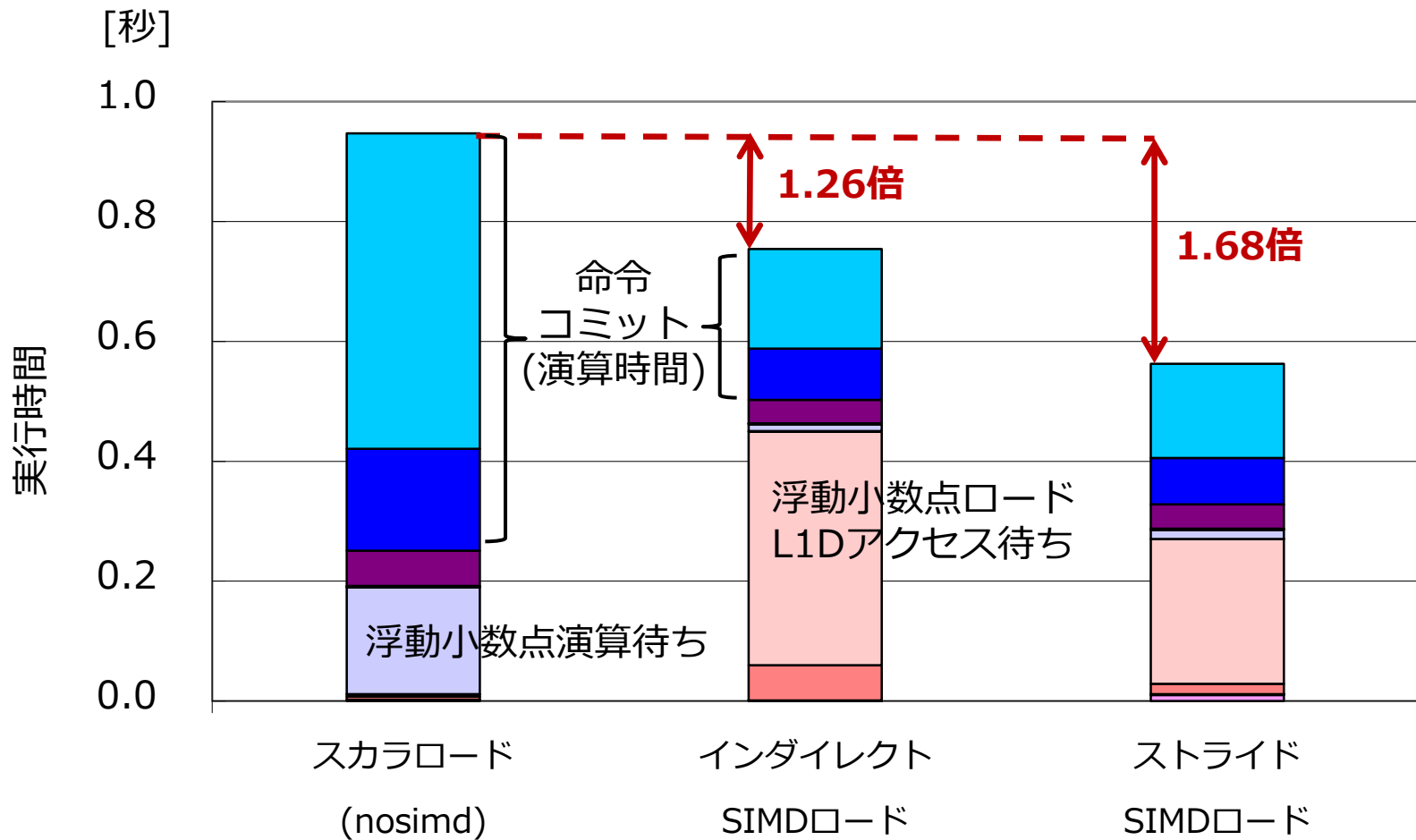
インダイレクトロードアクセス

```
real (8) :: b (0:6, lmax, kmax, 1:3)  
          (中略)  
  
do l=1, lmax  
!$omp do  
  do k=1, kmax  
    do n=nstart, nend  
      ij=n  
      q(n, k, l)=(  
        +b(0, ij, l, 1)*vx(ij, k, l) &  
        +b(1, ij, l, 1)*vx(ij, k, l) &  
        :  
        +b(6, ij, l, 3)*vz(ij3, k, l) &  
      )*fact
```

ストライドロードアクセス

判断材料を与えることでストライドロード命令を出力

■ ハンドチューニングによる効果

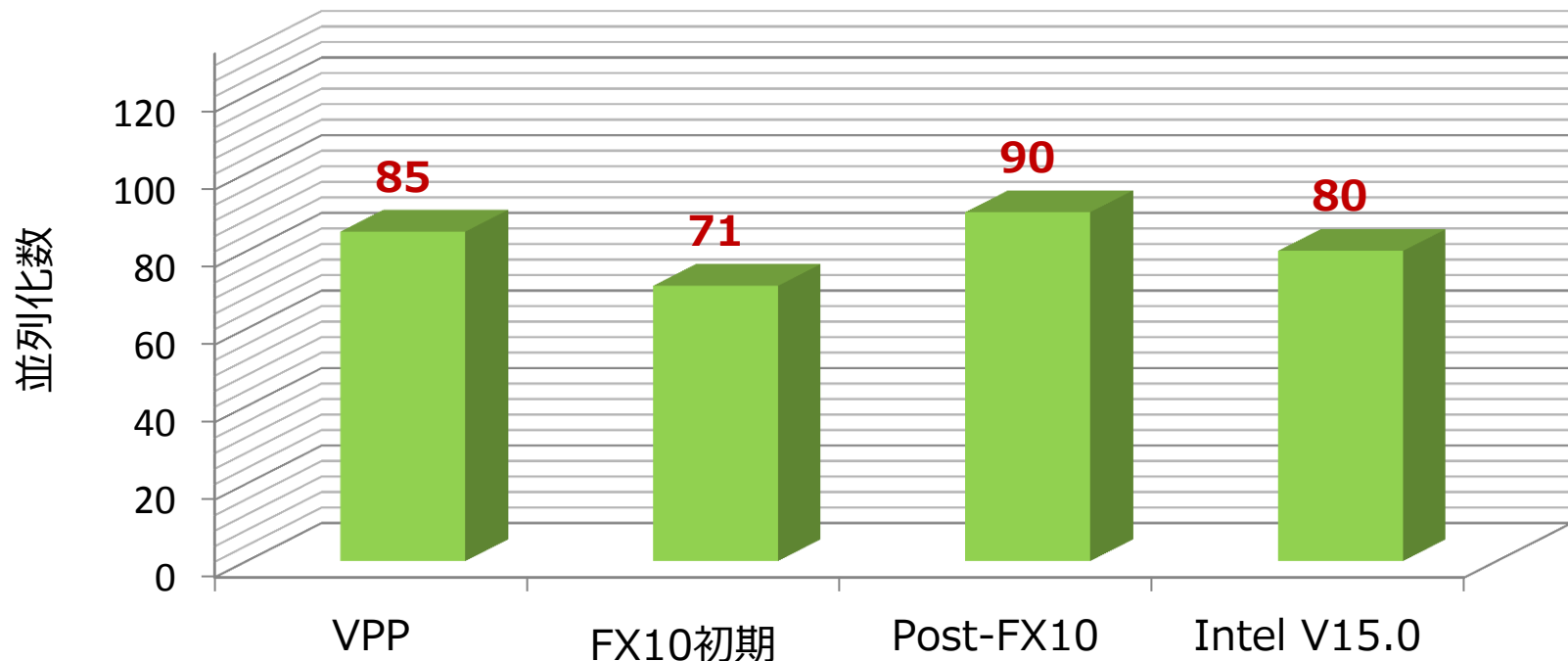


ストライド命令を利用し性能が向上

スレッド並列性能

■ ANL ベクトル化コンテスト

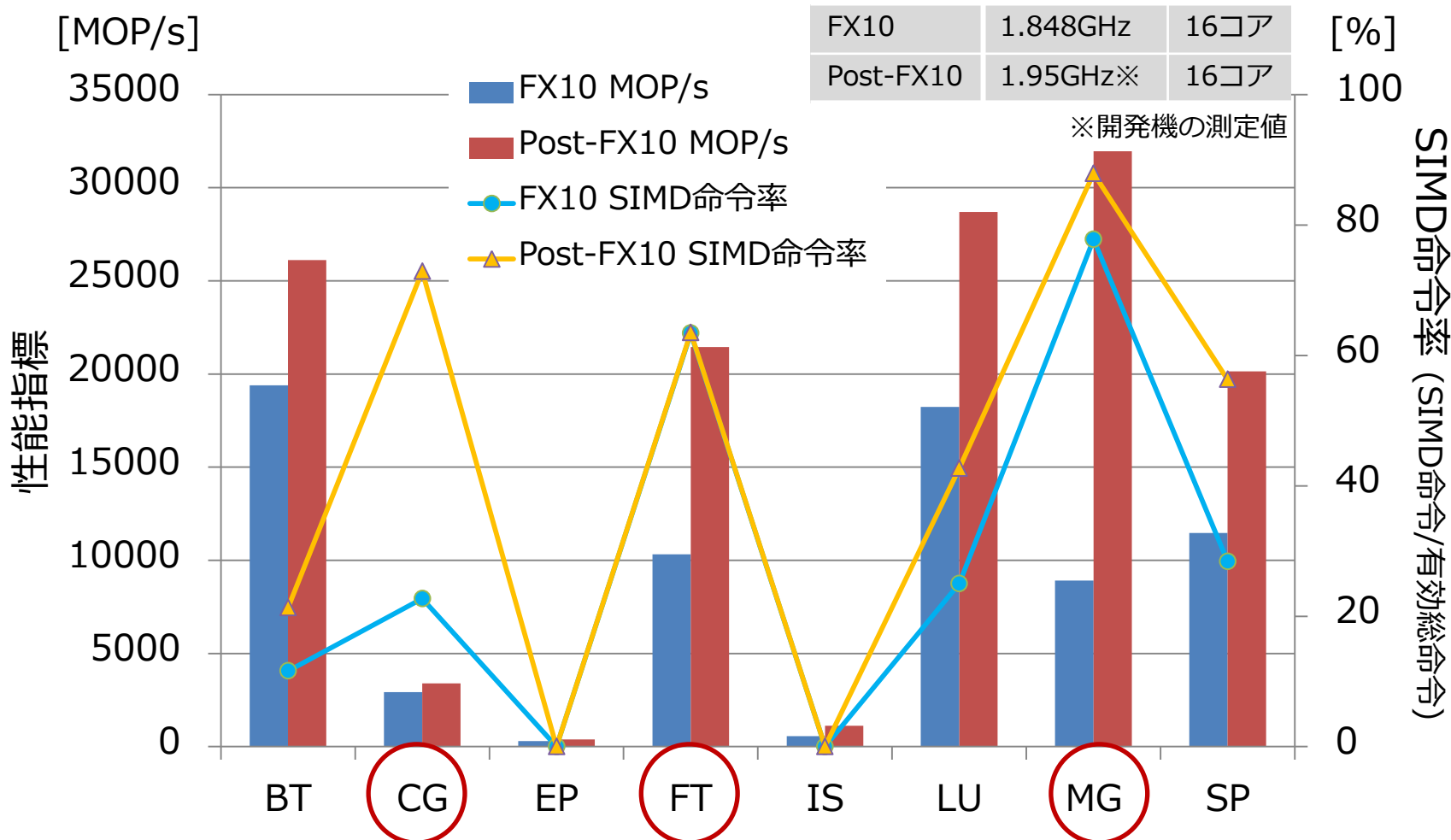
- コンパイラの解析能力を判定するプログラムの全135ループを比較
- VPPの技術を完全に取り込み解析能力を強化
- 「京」、FX10へフィードバック



コンパイラの解析能力の強化により並列化数が向上

アプリケーション性能

■ NAS Parallel Benchmarks

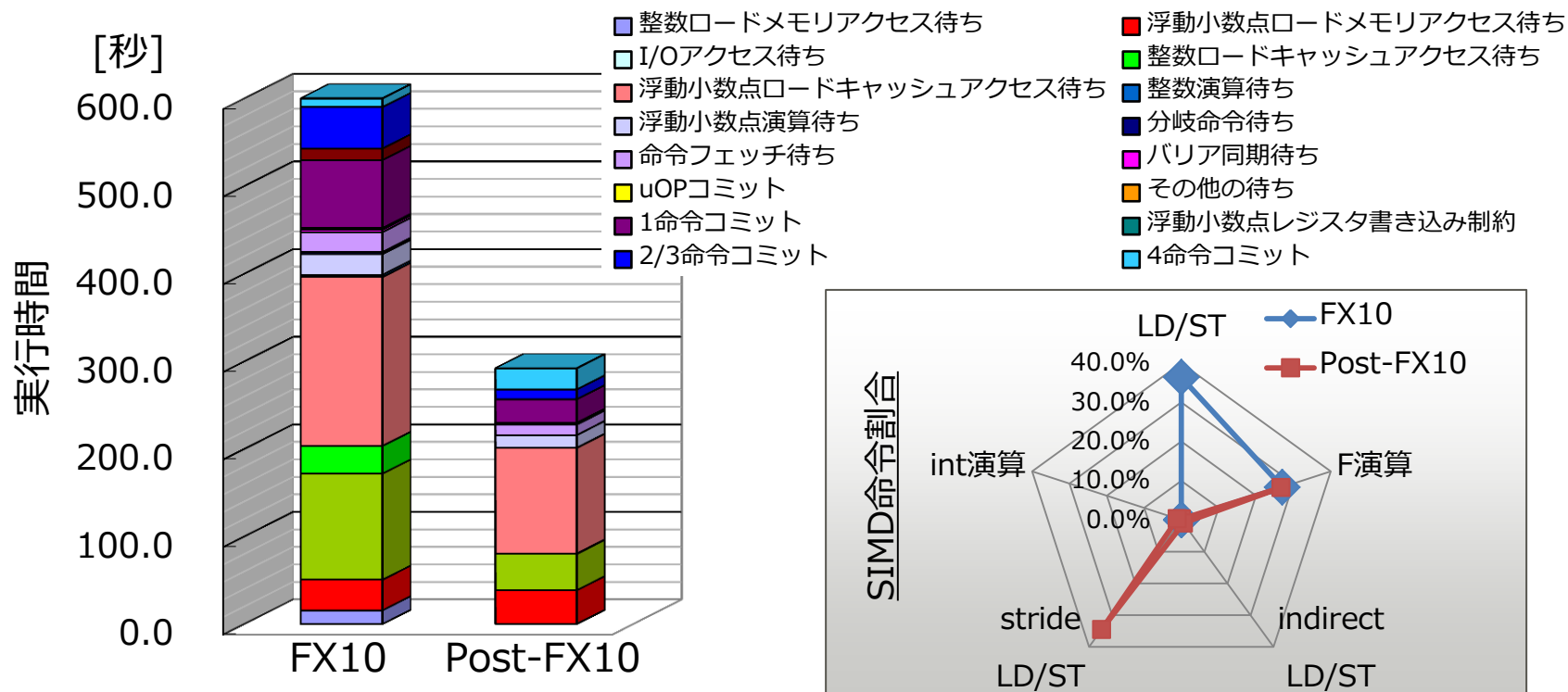


性能は向上しているがSIMD命令の出力状況に特徴あり

■ NAS Parallel Benchmarks

		メモリーバンド (GB/s)	SIMD命令率 (%)	L1Dミス率 (%)	性能値 (MOP/s)	SIMD化 向上比率	性能値 向上比率
BT	FX10	11.9	11.6	3.4	19396	1.8	1.3
	Post-FX10	16.2	21.3	1.4	26114		
CG	FX10	17.7	22.7	35.5	2929	3.2	1.2
	Post-FX10	20.5	72.9	32.9	3403		
EP	FX10	5.8	0	1.5	305	1.0	1.3
	Post-FX10	7.2	0	0.8	391		
FT	FX10	32.1	63.4	7.8	10319	1.0	2.1
	Post-FX10	64.7	63.5	4.5	21445		
IS	FX10	11.8	0	6.5	567	1.0	2.0
	Post-FX10	23.7	0	5.0	1118		
LU	FX10	56.7	25.0	6.6	18241	1.7	1.6
	Post-FX10	91.3	42.7	2.4	28706		
MG	FX10	29.8	77.8	11.0	8923	1.1	3.9
	Post-FX10	108.3	87.9	1.3	31957		
SP	FX10	67.3	28.4	5.6	11471	2.0	1.8
	Post-FX10	118.3	56.3	3.1	20149		

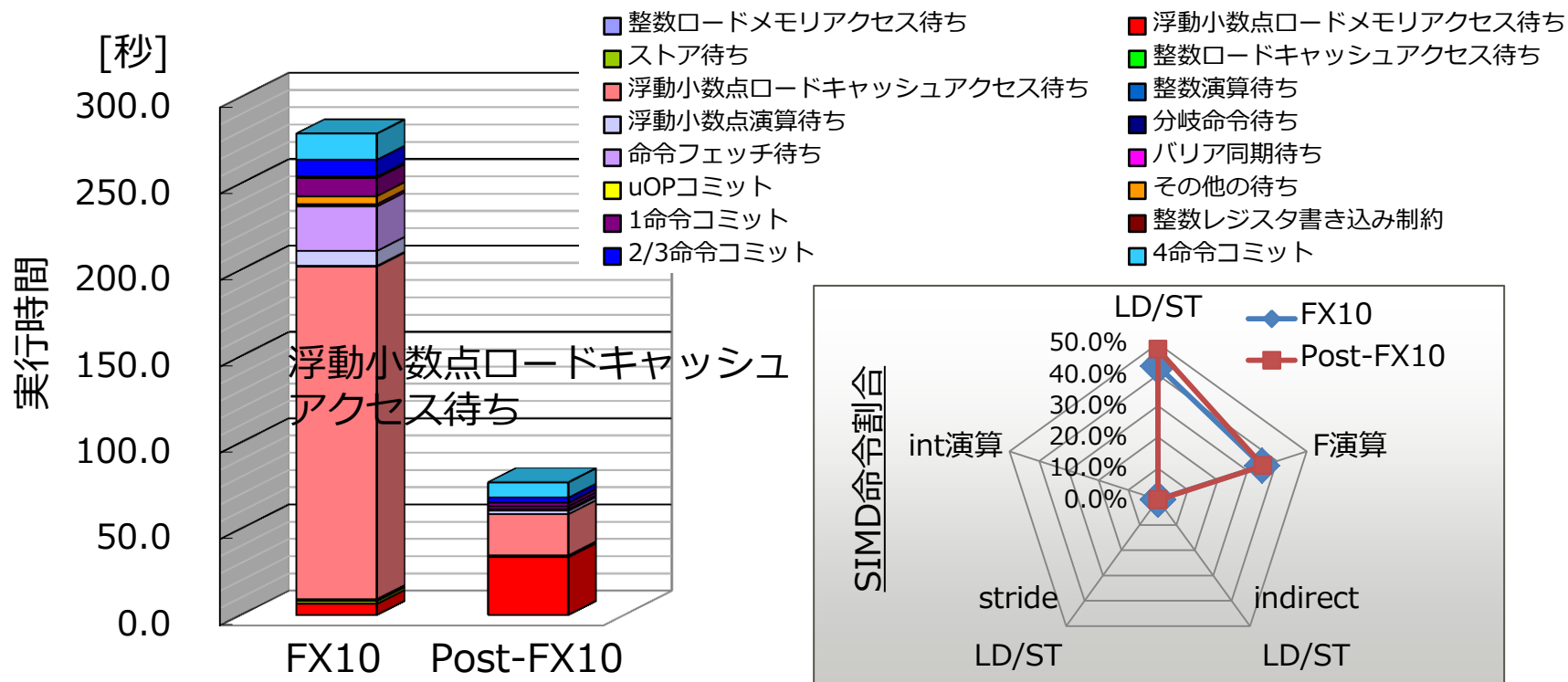
■ NAS Parallel Benchmarks : FT



		メモリーアップ (GB/s)	SIMD命令率 (%)	L1Dミス率 (%)	性能値 (MOP/s)	SIMD化 向上比率	性能値 向上比率
FT	FX10	32.1	63.4	7.8	10319	1.0	2.1
	Post-FX10	64.7	63.5	4.5	21445		

ストライド命令により命令の並列性が向上

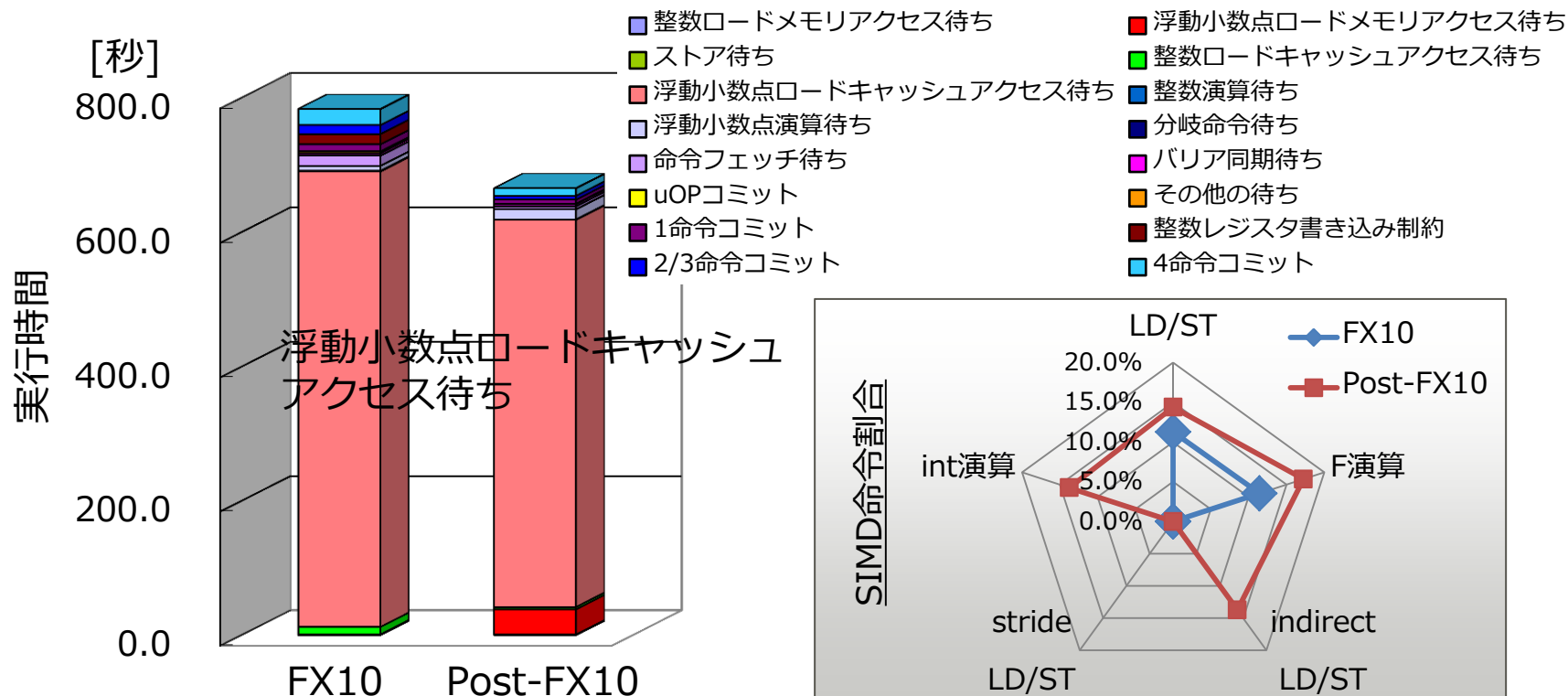
■ NAS Parallel Benchmarks : MG



		メモリーアップ (GB/s)	SIMD命令率 (%)	L1Dミス率 (%)	性能値 (MOP/s)	SIMD化 向上比率	性能値 向上比率
MG	FX10	29.8	77.8	11.0	8923	1.1	3.9
	Post-FX10	108.3	87.9	1.3	31957		

L1Dキャッシュの強化により性能向上

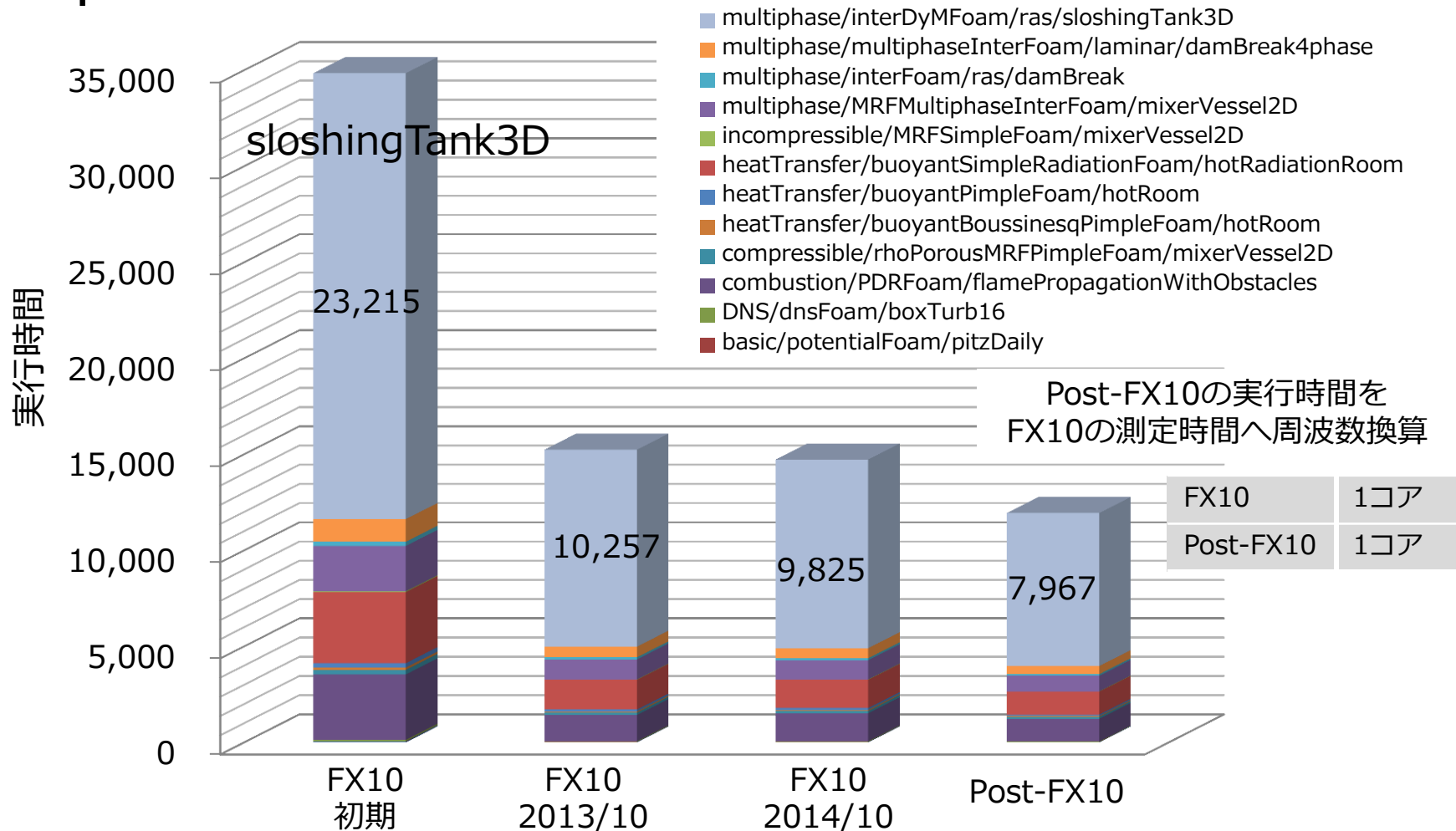
■ NAS Parallel Benchmarks : CG



		メモリーアップ (GB/s)	SIMD命令率 (%)	L1Dミス率 (%)	性能値 (MOP/s)	SIMD化 向上比率	性能値 向上比率
CG	FX10	17.7	22.7	35.5	2929	3.2	1.2
	Post-FX10	20.5	72.9	32.9	3403		

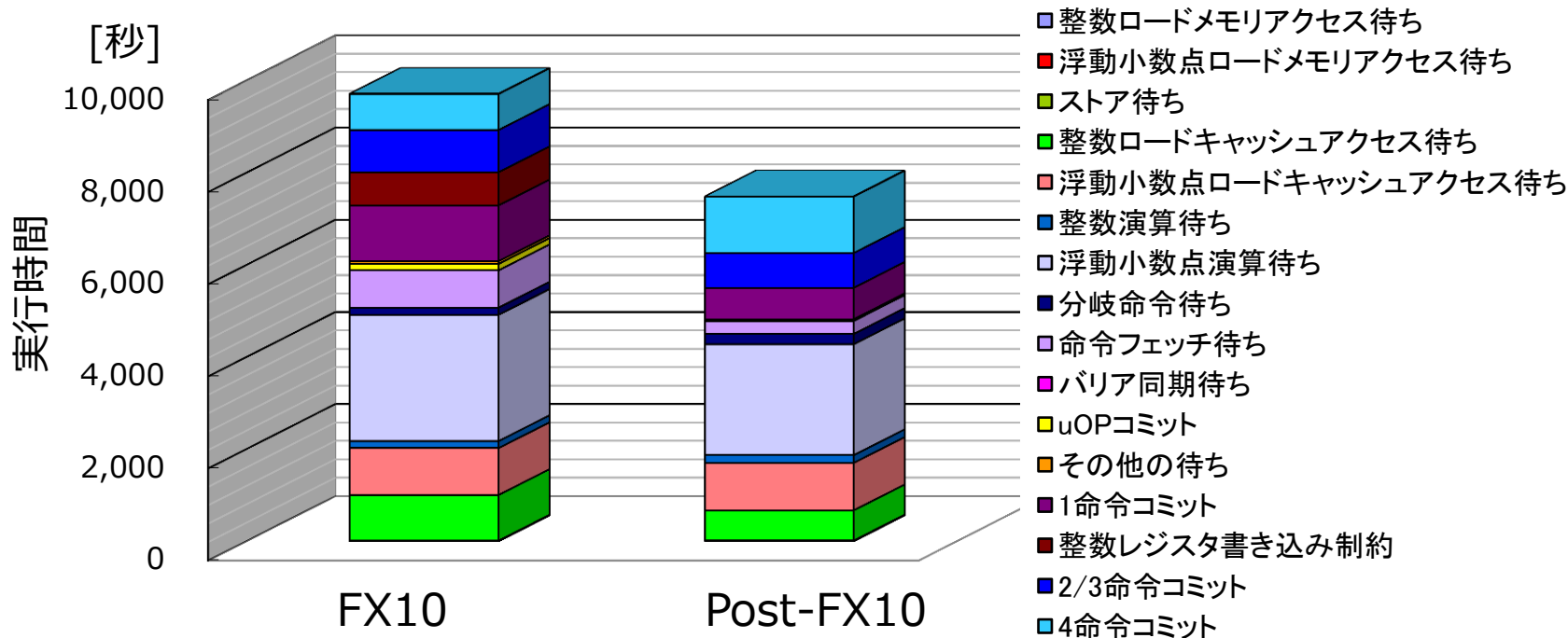
リストアクセスに対するコンパイラの命令出力を改善中

■ OpenFOAM 2.1.0



整数系を中心としたOSSコードでも性能向上あり

■ OpenFOAM: sloshingTank3D/interDyMFoam



浮動小数点演算ピーク比	MFLOPS	MIPS	浮動小数点演算数
1.18%	372.10	1959.69	2.79E+12
SIMD命令率(/有効総命令数)	SIMD命令率(/対象命令数)	有効総命令数	整数演算数
0.66%	1.51%	1.47E+13	5.16E+12

アウトオブオーダーの強化により整数演算の並列性が向上

まとめ

■ FX10で課題となったノード性能を大幅に改善

■ 命令レベルの並列化を強化

HPC向け拡張命令、コンパイラの強化

■ チューニングレスでアプリケーションの高速化


256 bit wide SIMD、HMCサポート、L1キャッシュの強化

■ C/C++アプリケーションの性能向上

コンパイラの強化



HPCをトップレベルで牽引するPost-FX10に乞うご期待



FUJITSU

shaping tomorrow with you