FUJITSU

shaping tomorrow with you

プログラミングガイド
プロセッサ編

V1.4

2021年8月

富士通株式会社

はじめに

FUJITSU

- 当資料は、A64FX プロセッサの特徴/マイクロアーキおよび基本的な性能情報をアプリケーション開発者やチューニング実施者の視点でまとめたものです。
- 当資料と併せて以下も参照してください。
  - Fortran 使用手引書
  - C言語 使用手引書
  - C++言語 使用手引書
  - プロファイラ 使用手引書
  - プログラミングガイド　プログラミング共通編
  - プログラミングガイド　Fortran編
  - プログラミングガイド　チューニング編
- 当資料の記載においては、以下の文章を参考にしています。
  - A64FX 論理仕様書
  - A64FX®Microarchitecture Manual
  - ARM® Architecture Reference Manual (ARMv8 , ARMv8.1 , ARMv8.2 , ARMv8.3)
  - ARM® Architecture Reference Manual Supplement The Scalable Vector Extension

- 商標について
  - Linux® は、Linus Torvalds 氏の米国およびそのほかの国における登録商標または商標です。
  - Red Hat は、Red Hat,Inc. の米国およびそのほかの国における登録商標または商標です。
  - ARMは、ARM Ltd.の英国およびその他の国における登録商標です。
  - そのほかの会社名、製品名等の固有名詞は、各社の登録商標または商標です。
  - 本資料に記載されているシステム名、製品名等には、必ずしも商標表示(®、™)を付記していません

目次

FUJITSU
shaping tomorrow with you

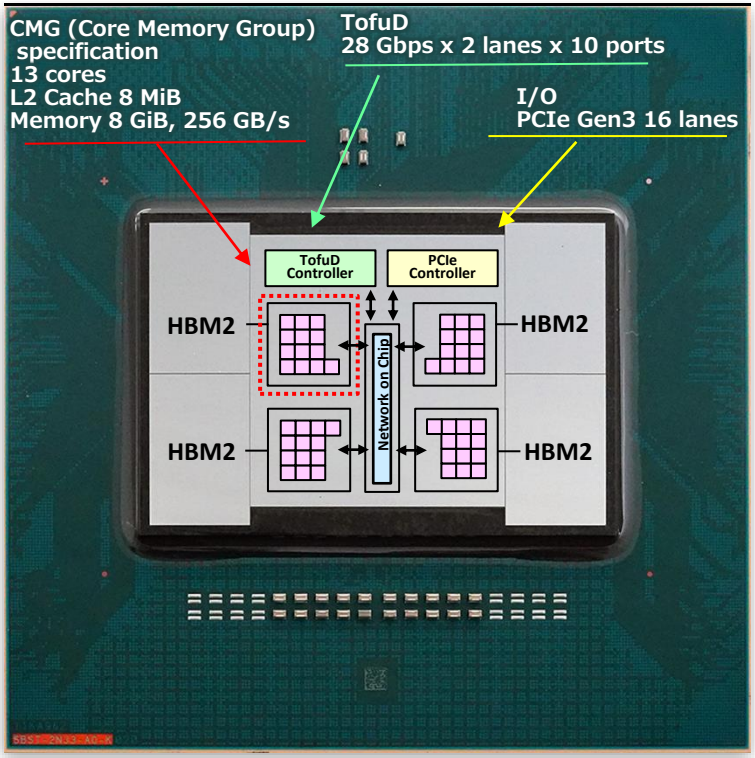
A64FXプロセッサ

- A64FXプロセッサ概要
- A64FXプロセッサ諸元
- インターコネクト “Tofu interconnect D“(TofuD) 概要

A64FXプロセッサ概要 (1/3)

FUJITSU

- Arm SVE を採用した高性能・高効率CPU
  - 倍精度演算性能 ： 3.072TFLOPS@2GHz , 90+%@DGEMM
  - メモリバンド幅 ： 1024 GB/s , 80+%@STREAM Triad



	A64FX
ISA (Base, extension)	Armv8.2-A, SVE
プロセステクノロジ	7 nm
倍精度ピーク性能	3.072TFLOPS@2GHz
SIMD幅	512-bit 256-bit/128-bit もサポート
コア数	48 + 4
メモリ容量	32 GiB (HBM2 x4)
メモリバンド幅	1024 GB/s
PCIe	Gen3 16 lanes
インターコネクト	TofuD(*) integrated

(*1) Tofu interconnect D

A64FXプロセッサ概要 (2/3)

FUJITSU

- A64FXプロセッサ (以下、A64FXと記述する) はHigh Performance Computing (HPC) 向けに設計され、ARMv8-A profile アーキテクチャ、および Scalable Vector Extension for ARMv8-A に準拠したアウト・オブ・オーダー実行型スーパースカラ・プロセッサである。
  A64FX はHPC向けにいくつかの特徴的なアーキテクチャを採用している。

  - Scalable Vector Extension

    A64FX は ARM命令セットアーキテクチャのベクトル拡張である Scalable Vector Extension (SVE) をサポートする。

  - Core Memory Group

    A64FXは13個のプロセッサ・コア、独立したL2キャッシュ、独立したメモリ・コントローラからなる Core Memory Group (CMG) と呼ばれるグループを持つ。
    プロセッサは4つのCMGを持ち、CMG間は Non-Uniform Memory Access (NUMA) 構成である。

  - セクタ・キャッシュ

    キャッシュを Way単位で仮想的に分割し、命令レベルで使用できる領域を指定できる機能である。プログラムはタグド・アドレスを使用することで領域を指定できる。
    L1キャッシュは4パーティション、L2キャッシュは2パーティションを2グループ持つ。

A64FXプロセッサ概要 (3/3)

FUJITSU

- ハードウェア・バリア

  ソフトウェアのプロセス、またはスレッド間の同期をハードウェアでサポートする機能である。
  この機能により、メモリ操作を行わずに高速に同期処理ができる。
  ハードウェア・バリア可能な範囲はCMG内であり、CMG間の同期処理は、ソフトウェア・バリアで行う。

- ハードウェア・プリフェッチ・アシスト

  ハードウェア・プリフェッチの振る舞いをプログラムから制御できる機能である。プログラムはシステムレジスタとタグド・アドレスを用いてハードウェアのプリフェッチ機構に情報を与えることができる。

- High Bandwidth Memory

  メインメモリにHigh Bandwidth Memory Gen2 (HBM2) を採用し、高いメモリ帯域を提供している。

A64FXプロセッサ諸元

FUJITSU

項目	諸元
プロセッサ・コア数	52 (13 cores / CMG)
CMG数	4
L1Iキャッシュ・サイズ	64KiB / 4way
L1Dキャッシュ・サイズ	64KiB / 4way
L2キャッシュ・サイズ	32MiB / 16way (8MiB / CMG)
キャッシュライン・サイズ	256B
メモリ・サイズ	32GiB(8GiB/CMG)
インターコネクト	Tofu Interconnect D
I/O	PCI-Express Gen3 16 Lanes
命令セット・アーキテクチャ	ARMv8-A, ARMv8.1, ARMv8.2, ARMv8.3 (*1), SVE

(*1) ARMv8.3 は Complex number support 命令のみサポートする。

A64FXプロセッサ諸元：バンド幅、レイテンシ

FUJITSU

		A64FX	備考
周波数【GHz】		2.0	
CPU数/ノード		1	
演算コア数/ノード		48	
CMG数/ノード		4	
メモリ容量/ノード【GiB】		32	
キャッシュ容量	L1【KiB/コア】	64(命令)+64(データ)	
	L2【MiB/CMG】	8	(注意) アシスタントコア付きノードの場合、アプリケーション利用は7MiB/14way と考える。
キャッシュレイテンシ【cycle】	L1	5 (EX, short) 8 (FL, short) 11 (FL, long)	
	L2	37～47	平均42
キャッシュバンド幅【B/cycle/コア】	L1	ヒット時：128	
	L2	42.7	
ノードあたり演算性能 (1コアあたり)【GFlops】	倍精度 単精度 半精度	3072 (64) 6144 (128) 12288 (256)	
基本演算レイテンシ【cycle】	整数add命令 整数mult命令 FMA命令	1 5 9	
メモリレイテンシ【ns】		150	
ノードあたり理論メモリバンド幅 (CMGあたり)【GB/秒】		1024(256)	
CMG間バンド幅		128GB/s×2(双方向)	

A64FXプロセッサ諸元：その他

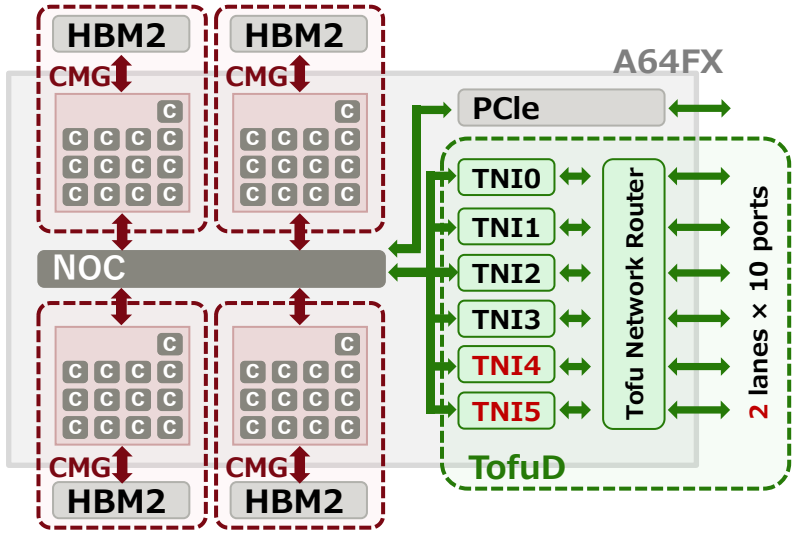
	A64FX	備考
最大デコード数　cycle当たり	4	
ハードウェアプリフェッチキュー	16	

インタコネクト “Tofu interconnect D“(TofuD) 概要

FUJITSU

- CPU にインタコネクトコントローラを実装
  - TNI を増やし、高いインジェクションBW と柔軟な通信パターンを実現
  - バリアリソースを増やし、柔軟な集団通信アルゴリズムの実装を可能に
- メモリバイパス技術により低レイテンシ通信を実現
  - ダイレクトディスクリプタとキャッシュインジェクション

	TofuD spec
Data rate	28.05 Gbps
Link bandwidth	6.8 GB/s
Injection bandwidth	40.8 GB/s
	Measured
Put throughput	6.35 GB/s
PingPong latency	0.49~0.54 μs



FUJITSU

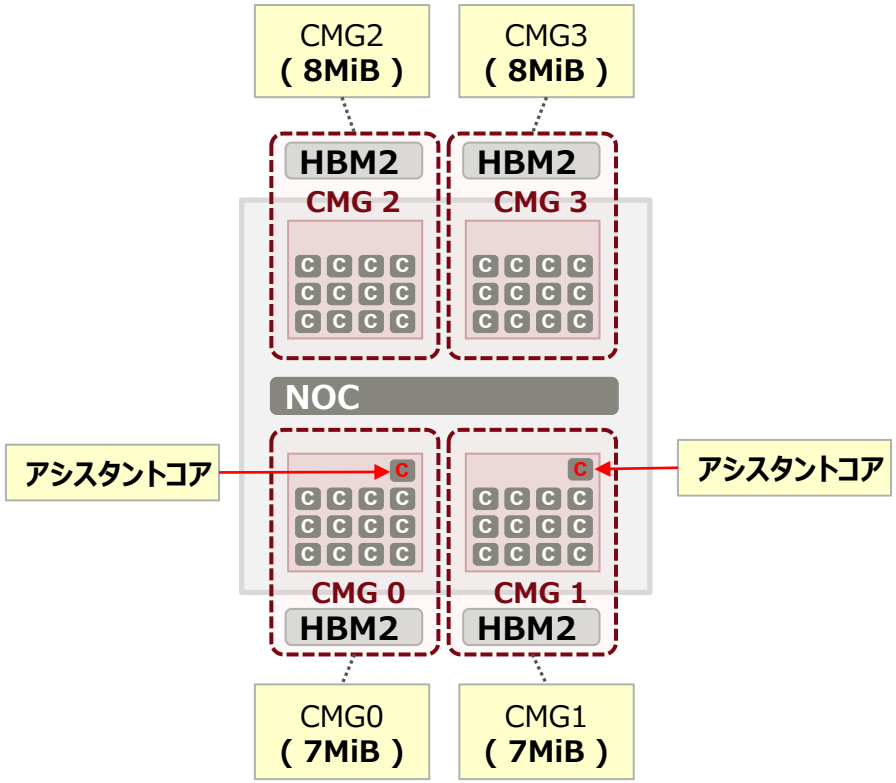
アプリで使用可能なL2キャッシュ容量について

- アプリで使用可能なL2キャッシュ容量について
- 検証：アシスタントコアの有無による性能

アプリで使用可能なL2キャッシュ容量について

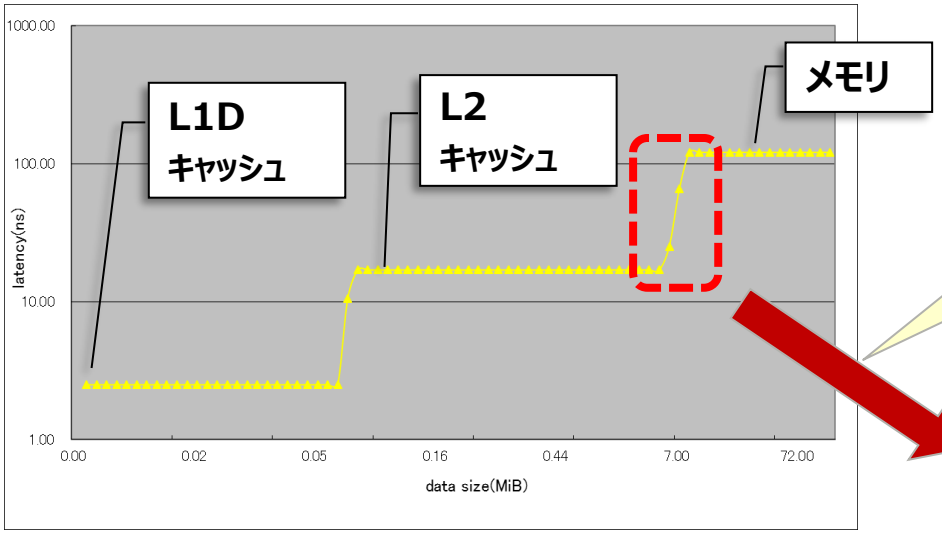
- アシスタントコアを含むCMGでは、L2キャッシュの一部(2way=1MiB分)をアシスタントコア用に使用する。
  そのため、アシスタントコアを含むCMGではユーザプログラム用として使用できるL2キャッシュは7MiBとなる。

計算ノードの場合

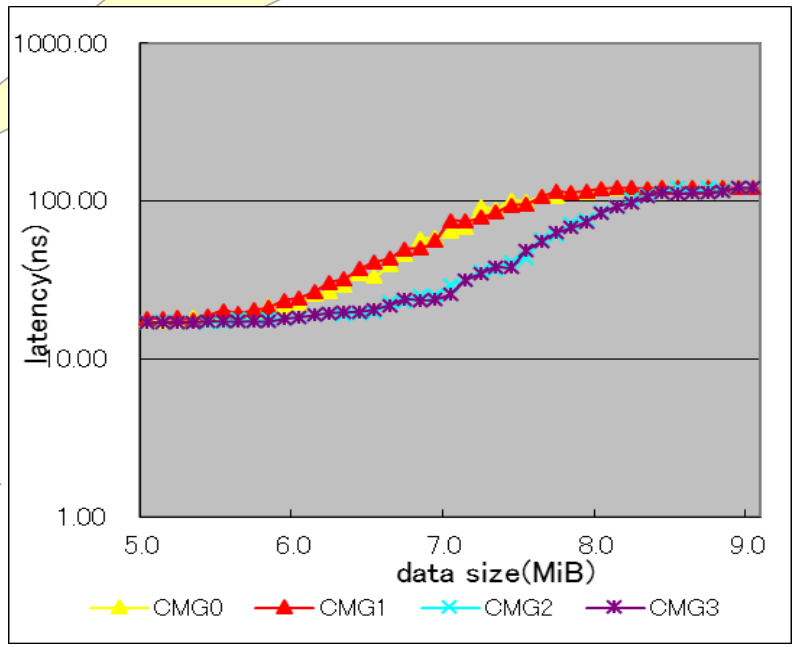


検証：アシスタントコアの有無による性能

- lmbench 性能 (2.0GHz)
  - CMG0 lmbench によるデータアクセスレイテンシの測定結果(整数アクセス時)



L2キャッシュアクセスからメモリアクセスへ変化する
データサイズ(5MiB～9MiB)をCMG毎に採取した



アシスタントコアを含むCMG(CMG0･CMG1)で
L2アクセスからメモリアクセスへの移行が早くなる

- CPU諸元としては、CMG当たりの
  L2キャッシュサイズは8MiB。
  だが、アプリ実行時は、7MiBと考えなければいけない。
- L2キャッシュサイズを意識したブロッキング等のチューニングを行うケースには注意が必要。
  (目安値はL2キャッシュサイズ7MiB弱)

FUJITSU
shaping tomorrow with you

マイクロアーキ

- プリフェッチ
- SFI(Store Fetch Interlock)について
- PMUイベント
- ラージページ
- セクタキャッシュ
- 高速ストア(zfill)
- データアクセスのアライメント制約
- アウトオブオーダー実行の検証
- SIMD幅
- 電力制御

プリフェッチ

- プリフェッチについて
- ハードウェアプリフェッチの動作
- プリフェッチ距離設定コマンド
- プリフェッチ距離検証
- 実アプリ(NICAM)を使用したハードウェアプリフェッチ評価
- A64FXで提供されるプリフェッチ命令
- ハードウェアとソフトウェアのプリフェッチ協調
- プリフェッチの協調 検証

プリフェッチについて (1/2)

FUJITSU

- A64FXでは、ハードウェアとソフトウェアのプリフェッチとして、以下の新機能をサポートしている。
  - ハードウェアプリフェッチ(HWPF)
    - ハードウェアプリフェッチ距離設定機能
    - ハードウェアストライドプリフェッチ機能
  - ソフトウェアプリフェッチ(SWPF)
    - プリフェッチ距離の自動調整
    - SVE Gatherプリフェッチ命令のサポート

- これらのプリフェッチを利用することにより、データアクセスのレイテンシを隠蔽しアプリの高速実行を可能としている。（レイテンシ隠蔽）

プリフェッチについて (2/2)

FUJITSU

- プリフェッチの距離について

  ハードウェアプリフェッチ・ソフトウェアプリフェッチでは、以下のライン先のデータに対しプリフェッチを行う。

	ハードウェアプリフェッチ		ソフトウェアプリフェッチ	
	L1プリフェッチ	L2プリフェッチ	L1プリフェッチ	L2プリフェッチ
FX100	2ライン	最大16ライン	3ライン	15ライン
A64FX	**最大6ライン**	**最大40ライン**	**自動**	**自動**

ハードウェアプリフェッチの距離はユーザの設定も可能になる

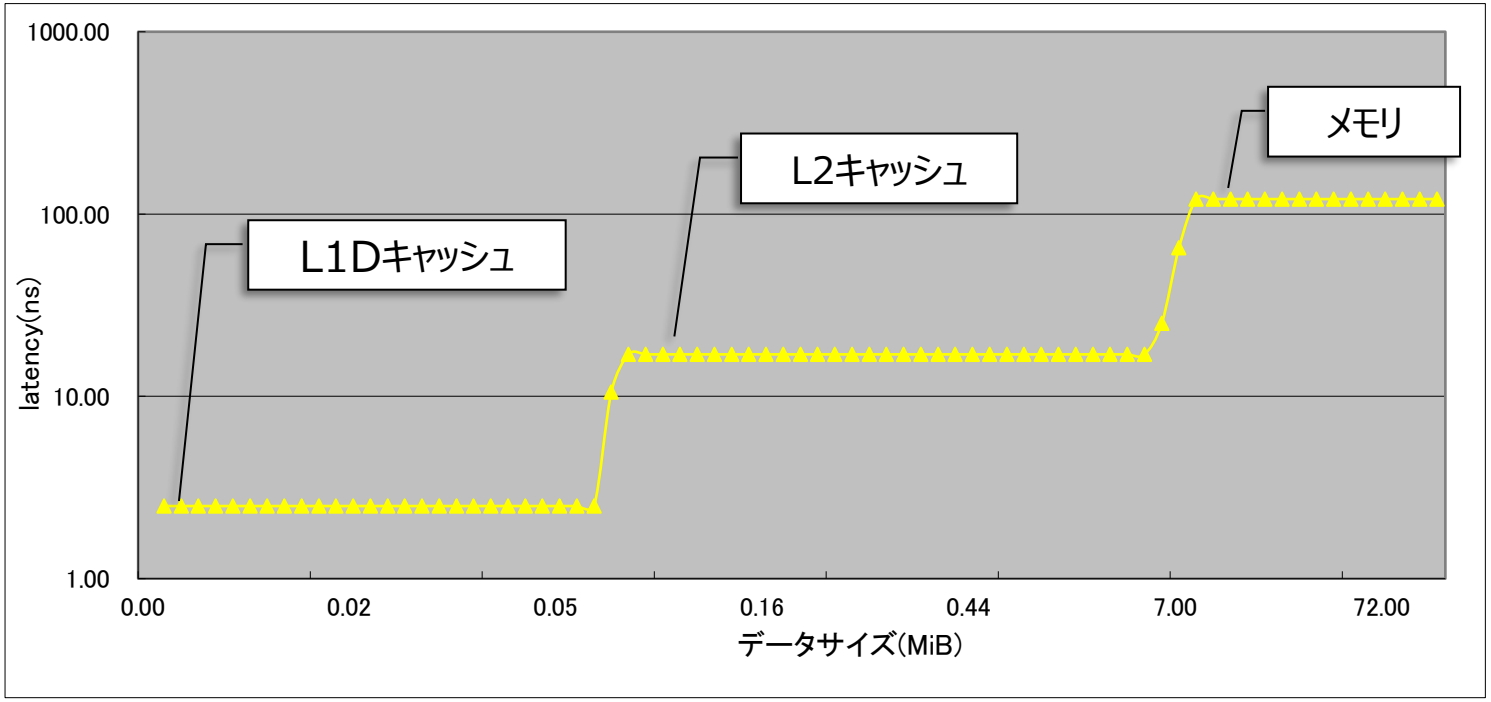
ソフトウェアプリフェッチは距離の自動調整がある

- ループブロッキングや外側ループアンローリングのチューニング実施時、内側ループの繰返し回数が少ないとハードウェアプリフェッチが動作しなくなる場合があるので注意が必要です。

【参考】レイテンシの隠蔽とは

- レイテンシの隠蔽とは、データアクセスのレイテンシ（データの転送要求をしてから戻ってくるまでの時間）をプリフェッチで隠蔽することを意味する。データアクセスは,L1Dキャッシュ、L2キャッシュ、メモリの 3 種類がある。そのうちL2キャッシュおよびメモリのレイテンシ隠蔽がプリフェッチの対象となる。

- LMbench に よるデータアクセスレイテンシの測定結果 （整数アクセス時）



ハードウェアプリフェッチの動作

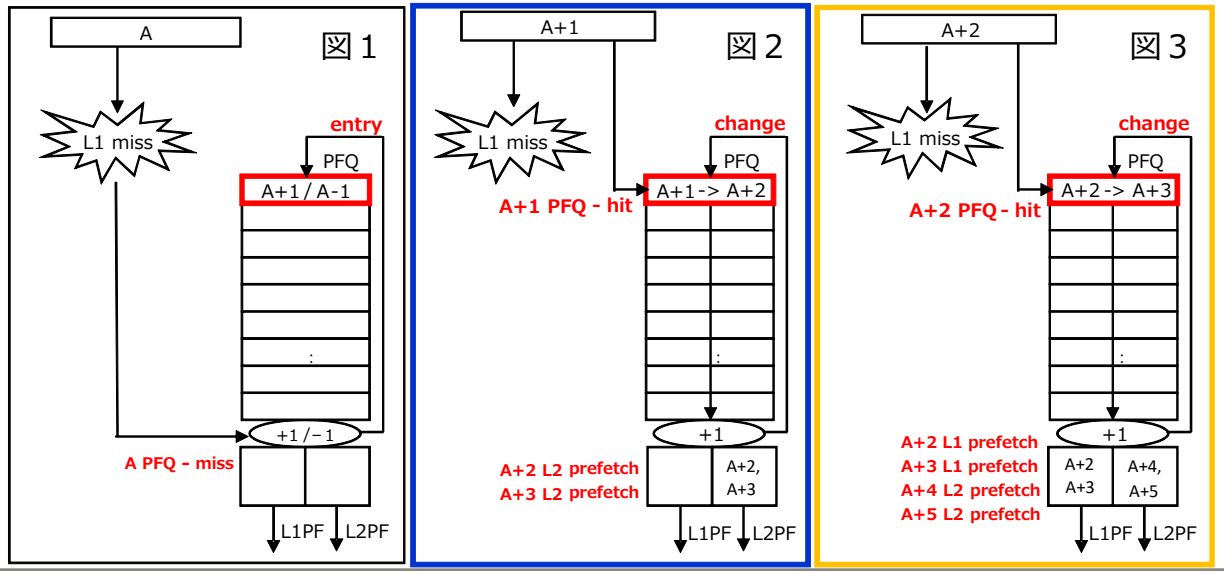
ハードウェアプリフェッチ動作条件

- L1ミスを起因とする
- L1ミスがキャッシュライン単位で連続に進む

ハードウェアプリフェッチの動作アルゴリズム

1. キャッシュラインA のミスが発生すると、ラインA+1/A-1が、PFQ(プリフェッチキュー)と呼ばれる16 エントリあるFIFO(キュー)に新規登録される。(図１)
2. 後続アクセスがラインA+1にアクセスすると、PFQに登録されているA+1にヒットした時に昇順ストリームアクセスと判断して、以降は昇順方向にHWPFを開始する。また、PFQのA+1はA+2に更新される。(図２)
3. 上記のストリームアクセス検出後は、L2プリフェッチは1度に2ライン分のデータをプリフェッチすることで40ライン先まで拡張する。A64FXではL1プリフェッチもL2プリフェッチ同様に1度に2ライン分のデータをプリフェッチすることで6ライン先まで拡張する。(図３)

プリフェッチの概略



注)表中の数字はラインで示している。バイト表記では １ ライン ＝ ２５６ バイトで読み替えること。

PFQ とプリフェッチアドレスの対応

PFQ-hit	L2HWPF-ADRS	L1PHWF-ADRS
A+ 1	A+ 2, A+ 3	
A+ 2	A+ 4, A+ 5	A+ 2, A+ 3
A+ 3	A+ 6, A+ 7	A+ 4, A+ 5
A+ 4	A+ 8, A+ 9	A+ 6, A+ 7
A+ 5	A+10, A+11	A+ 8, A+ 9
A+ 6	A+12, A+13	A+10
A+ 7	A+14, A+15	A+11
A+ 8	A+16, A+17	A+12
A+ 9	A+18, A+19	A+13
A+10	A+20	A+14
A+11	A+21	A+15
A+12	A+22	A+16
A+13	A+23	A+17
A+14	A+24	A+18
A+15	A+25	A+19

※L2PF=10ライン先、L1PF=4ライン先の場合

ハードウェアプリフェッチ距離設定コマンド

FUJITSU

ハードウェアプリフェッチ距離設定用コマンド(hwpfctl)の提供

項目	内容
書式	hwpfctl [--disableL1] [--disableL2] [--distL1 lines_l1] [--distL2 lines_l2] [--weakL1] [--weakL2] [--verbose] command {arguments ...} hwpfctl --default [--verbose] command {arguments ...} hwpfctl --reset [--verbose] hwpfctl –help
説明	hwpfctlコマンドは、A64FXに搭載されているハードウェアプリフェッチ(stream detect mode)の振る舞いを変更する。 変更対象となるCPUコアは、プロセスアフィニティに準ずる。
オプション	--disableL1 --disableL2 L1/L2キャッシュに対するハードウェアプリフェッチを無効とする。省略時はハードウェアプリフェッチが有効である。 --distL1=lines_l1 --distL2=lines_l2 L1/L2キャッシュに対してプリフェッチするキャッシュラインを、キャッシュミスが発生したキャッシュラインを起点とするキャッシュラインの数で指定する。lines_l1にはL1キャッシュに対するプリフェッチ対象ラインを1から15までの値を指定できる。また、lines_l2にはL2キャッシュに対するプリフェッチ対象ラインを1から60までの値を指定できる。ただし、lines_l2に指定した値は4の倍数に切り上げられてシステムレジスタに書き込まれる。0を指定した場合はCPUのデフォルト値で動作する。省略時ならびに無効な値が指定された場合は0が指定されたものとみなす。 --weakL1 --weakL2 L1/L2キャッシュに対するプリフェッチ要求の優先度をweakとすることを指示する。省略時はstrongである。 --default デフォルト設定でcommandを起動する。--verbose以外のオプションは無視する。 --reset システムレジスタの値を初期化する。--verbose以外のオプションは無視する。 --verbose システムレジスタの変更前後の値を出力する。 --help 使用方法を表示する。

ハードウェアプリフェッチ距離設定用コマンド(hwpfctl)の使用例

```
hwpfctl  –distL1=6  –distL2=40  a.out
```

プリフェッチ距離検証：条件

FUJITSU

プリフェッチ性能評価 測定条件

	パターン
検証コード	- Triad L2キャッシュアクセス（L1プリフェッチ距離評価） - Triad メモリアクセス（L2プリフェッチ距離評価）
評価プリフェッチ距離	ハードウェアプリフェッチ距離・ソフトウェアプリフェッチ距離(strong) - L1プリフェッチ：3~10line先 - L2プリフェッチ：10,15,20,25,30,35,40line先 (L1はデフォルト(HWPF=6、SWPF=自動(Triadでは4))
測定コア	12コア実行(CMG0)
翻訳オプション	-Kfast ソフトウェアプリフェッチ評価時： -Kprefetch_sequential=soft -Kprefetch_line=? -Kprefetch_line_L2=?
アクセス範囲	Triadコード評価は以下の条件とする - 演算配列には倍精度型を使用 - bss を使用 - 最内ループ繰返し数、配列サイズ(n) - L1プリフェッチ評価：174720 (アクセスする配列サイズの合計がL2キャッシュの1/2) - L2プリフェッチ評価：10485120 (アクセスする配列サイズの合計がL2キャッシュの30倍) ※各配列は256バイトアラインで確保している - 外側ループ繰返し数(iter) - L1プリフェッチ評価：10000 - L2プリフェッチ評価：3000

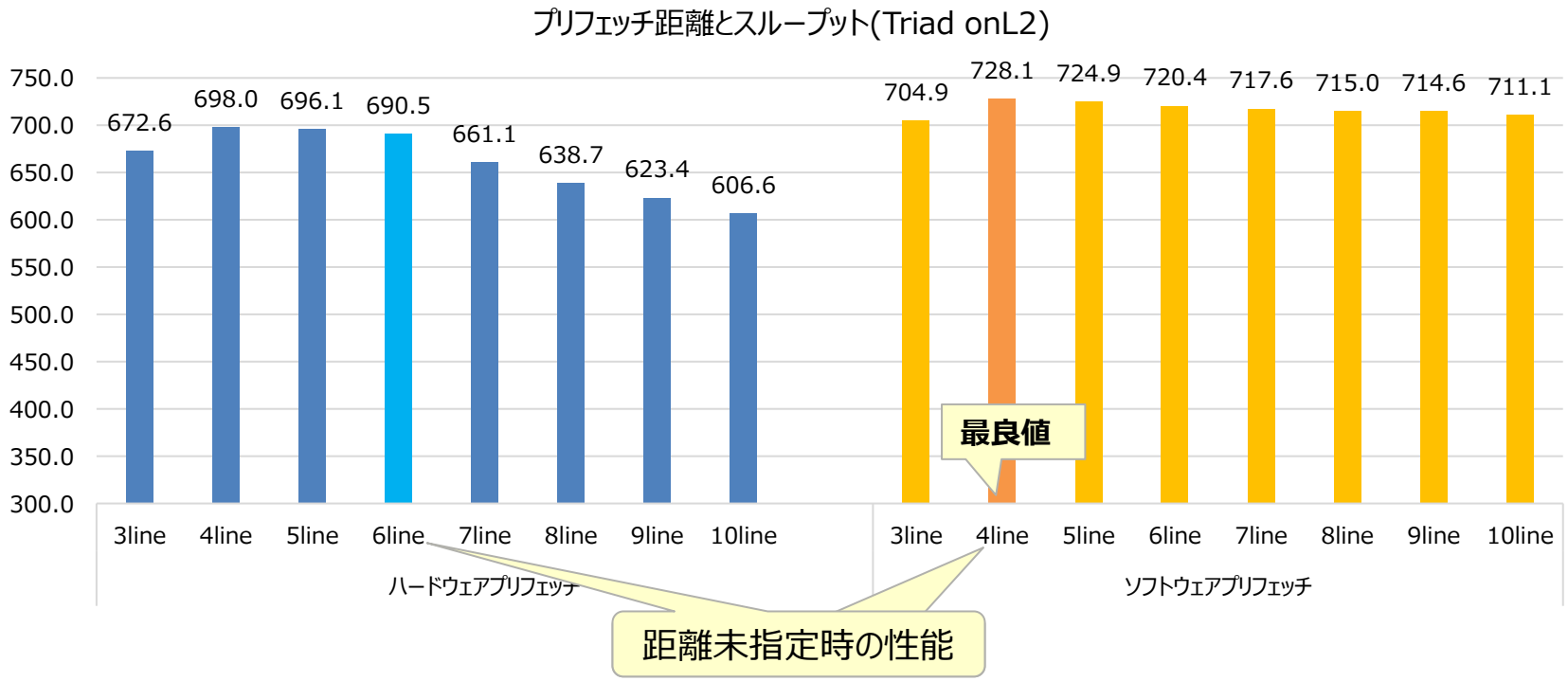
プリフェッチ距離検証：L1プリフェッチ

FUJITSU

- Triad(L2キャッシュアクセスケース)によるL1プリフェッチ距離評価

Triad

```
!$omp parallel
 Do j = 1, iter
!$omp do
  Do i = 1, n
y(i)=x1(i) + c0 * x2(i)
  End Do
!$omp end do nowait
 End Do
!$omp end parallel
```

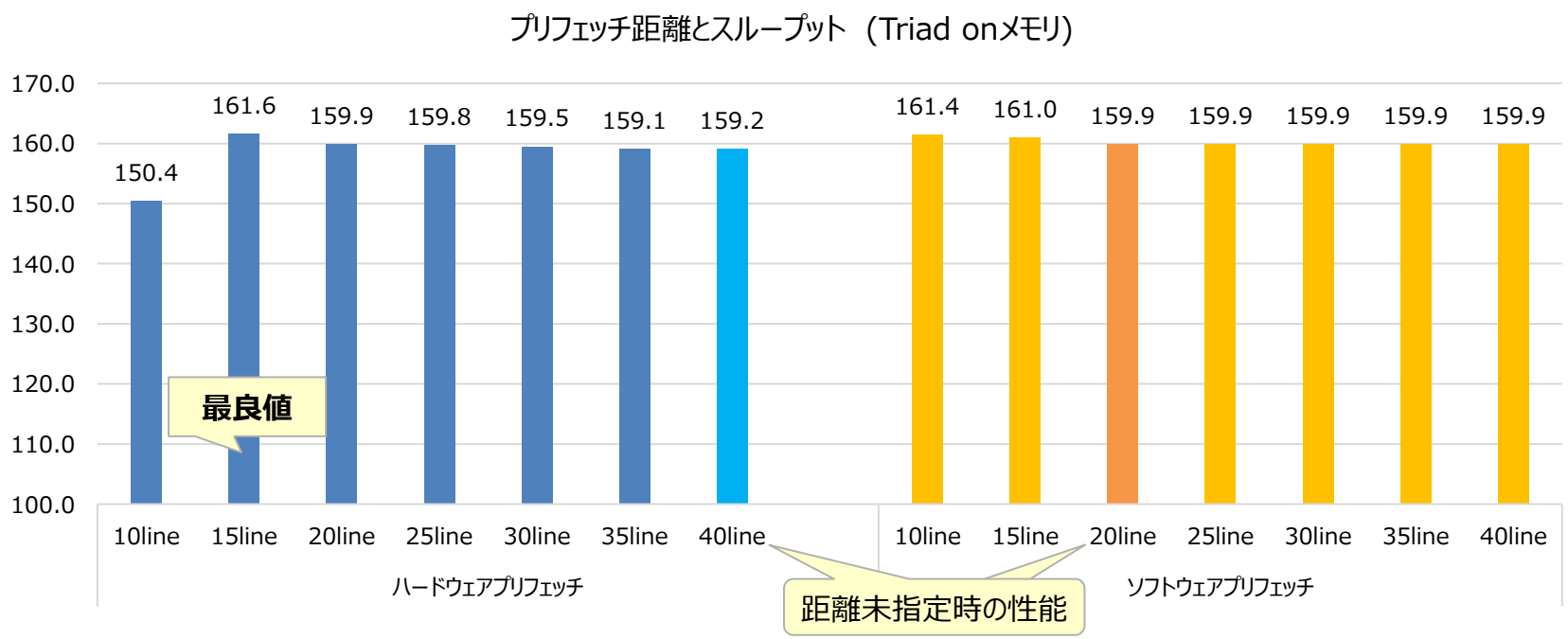


プリフェッチ距離検証：L2プリフェッチ

- Triad(メモリアクセスケース)による L2プリフェッチ距離評価

Triad

```
!$omp parallel
  Do j = 1, iter
!$omp do
     Do i = 1, n
        y(i)=x1(i) + c0 * x2(i)
     End Do
!$omp end do nowait
  End Do
!$omp end parallel
```



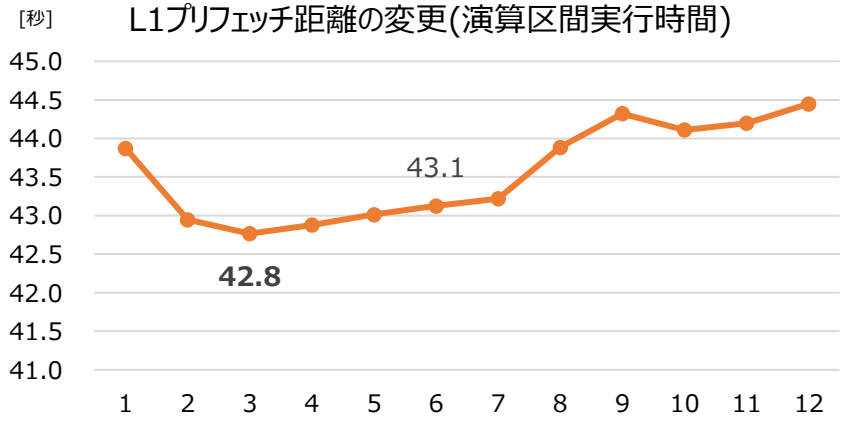
※スループットは、データ書き込み先キャッシュラインの読み込み分を含めない値になります

実アプリ(NICAM)を使用したハードウェアプリフェッチ評価

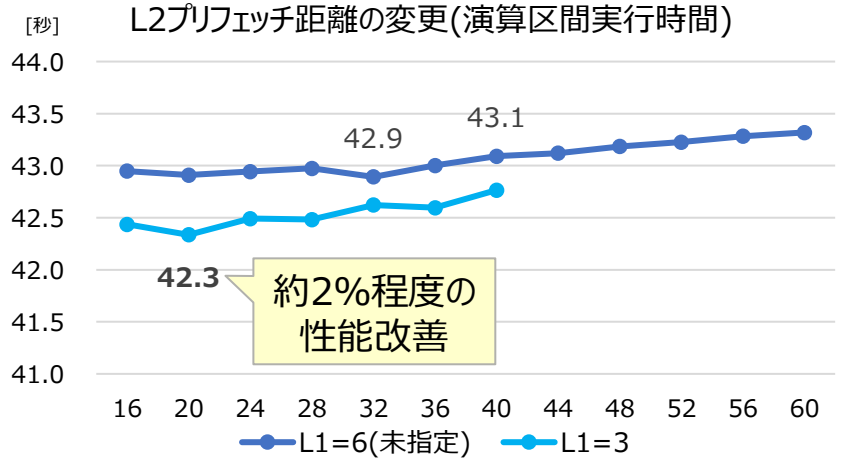
FUJITSU

- ハードウェアプリフェッチ距離調整評価
  ハードウェアプリフェッチ距離調整機能を使用し、実アプリ(NICAM)での性能評価を行った。

  - L1プリフェッチ距離の評価
    L2プリフェッチ距離は初期値(40)を使用、L1＝3で最速値となった。



  - L2プリフェッチ距離の評価
    まずは、L1プリフェッチ距離は初期値(6)を使用、近距離(16～32)で高速になることを確認。
    次に、L1プリフェッチ距離の評価からL1＝3を指定し、近距離の測定を行い、約2%程度の向上を確認。



A64FX(ISA)で提供されるプリフェッチ命令

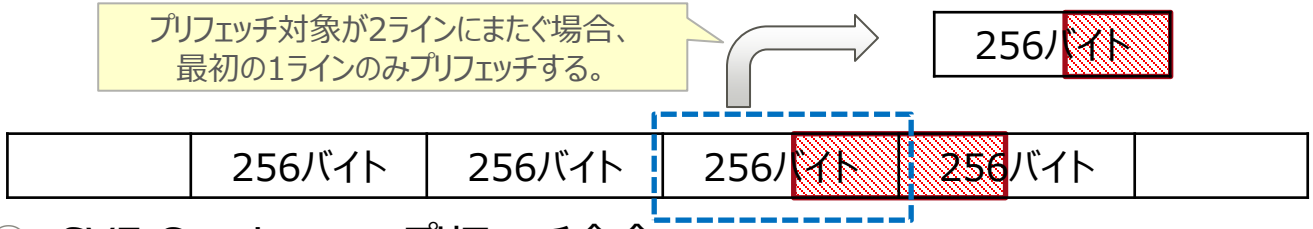
FUJITSU

- A64FXで提供されているプリフェッチ命令

①ARMv8プリフェッチ命令と、
②SVE Contiguousプリフェッチ命令は、
連続アクセスの際に使用される命令

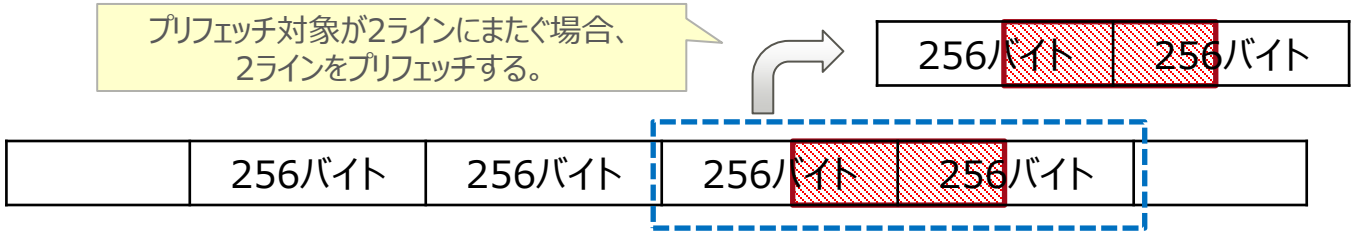
① ARMv8 プリフェッチ命令

連続ロード/ストア命令に対応するプリフェッチ命令(ラインをまたぐ際の考慮なし)



② SVE Contiguous プリフェッチ命令

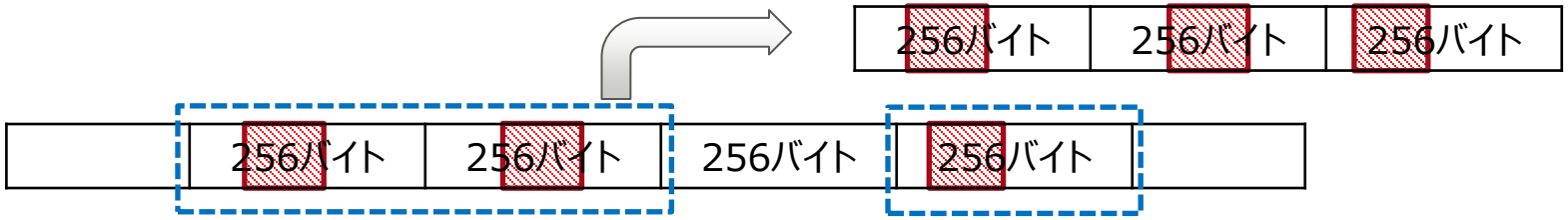
連続ロード/ストア命令に対応するプリフェッチ命令(ラインをまたぐ際の考慮あり)



③ SVE Gather プリフェッチ命令

離散アクセス命令(Gather/Scatter)に対応するプリフェッチ命令

③SVE Gatherプリフェッチ命令は、
インダイレクトアクセスの際に使用される命令



ハードウェアとソフトウェアのプリフェッチ協調

以下のケースではハードウェアプリフェッチが生成されないこともある。
ソフトウェアプリフェッチで補完することで理想的な性能を実現することができる。

- ストリーム数が16個を超える場合
- ブロックストライド
  - ループブロッキングしているケース（配列置換、行列演算等に有効）
  - 外側ループでアンローリングしているケース
- マスク付きSIMDによるアクセス
  if文内にあるストリームをアクセスする際、if文の真率により連続アクセスとはならず、
  ハードウェアプリフェッチが生成されないケースがある。

ハードウェアプリフェッチが生成されないケースをソフトウェアプリフェッチで補う
という意味で「ハードウェアとソフトウェアのプリフェッチ協調」と呼ぶ。

プリフェッチの協調 検証 (1/2)

FUJITSU

■ ストリーム数が16を超えるケースの評価

最内ループにて、ストリーム数が16を超えるケースでは、一部のプリフェッチをSWPFで補うこと(協調)が行われる。
SWPFの対象は、ストリーム数-16(20ストリームの場合、ソフトでは4ストリーム)となる。
これはコンパイラが自動で識別し、ユーザが意識することなく行われている。

● 今回の検証の評価ケース

①HWPFのみ

16ストリームを超えるケースでもSWPFで補うことをせず、HWPFのみでプリフェッチを行った実行を評価する。
PFQが不足し、プリフェッチの効果が十分に得られない可能性が出てくる。
ショートループのようなプログラムではHWPFの立ち上がりが顕著になることもある。

②SWPFのみ

すべてのプリフェッチをSWPFで行った実行を評価する。
命令の増加が副作用となり性能向上しない可能性もある。

③HWPF+SWPF(コンパイラのデフォルト)

16ストリームを超えたケースでSWPFが補う動作をする実行を評価する。

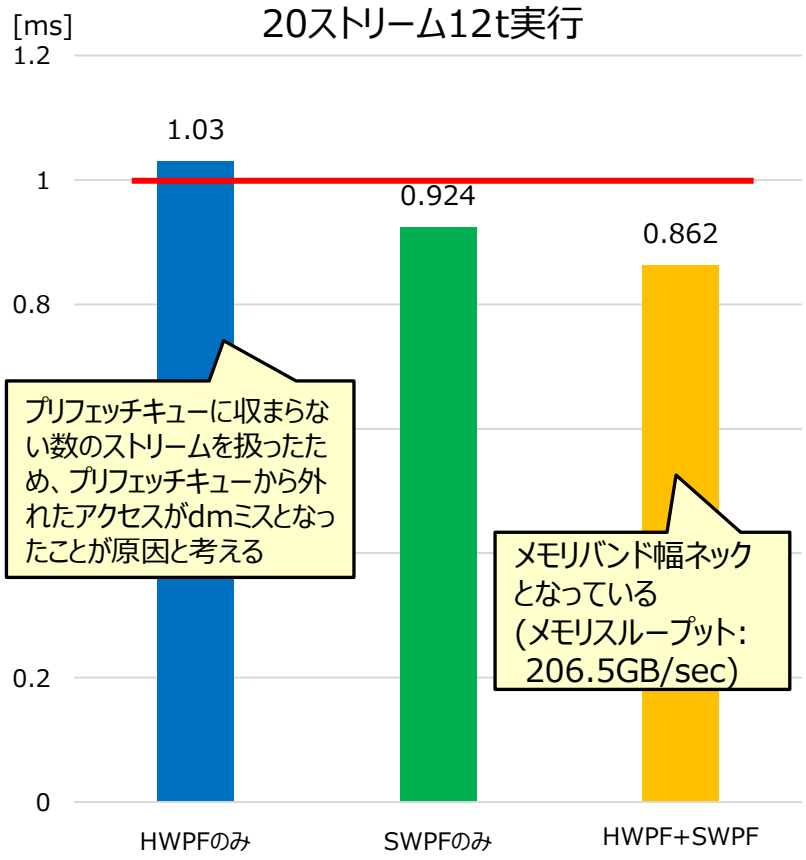
プリフェッチの協調 検証 (2/2)

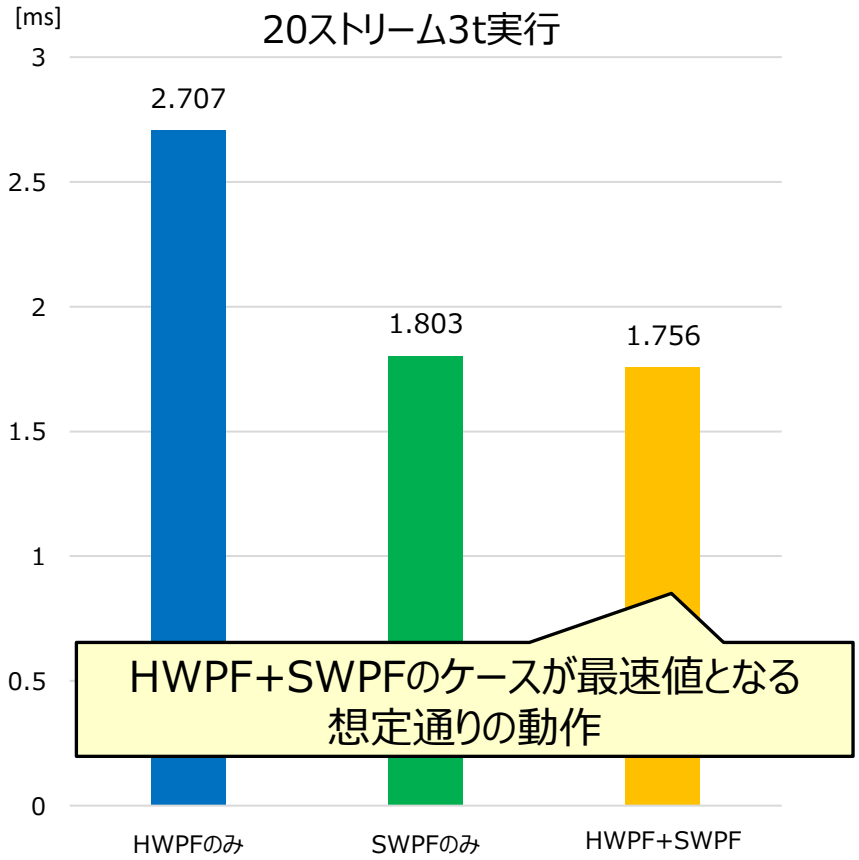
FUJITSU

- 実行結果

```
do k = 1, iter                                    iter=3 n=323323
!$omp parallel do
  do i = 1, n
    y(i)  = (((((((x1(i) &
          *c0 +x2(i) ) *c1 +x3(i) ) *c2 +x4(i) ) *c3 +x5(i) ) &
          *c4 +x6(i) ) *c5 +x7(i) ) *c6 +x8(i) ) *c7 +x9(i) )
    y2(i) = (((((((x11(i) &
          *c0 +x12(i) ) *c1 +x13(i) ) *c2 +x14(i) ) *c3 +x15(i) ) &
          *c4 +x16(i) ) *c5 +x17(i) ) *c6 +x18(i) ) *c7 +x19(i) )
  end do
end do
```

3スレッド実行することで
メモリバンド幅ネックとならない状態(レイテンシネック）を評価





FUJITSU

SFI(Store Fetch Interlock)について

- SFI とは
- 過剰 SFI になる要因と SFI プリチェックをする要因
- 測定ケースの説明
- コンパイラによる過剰のSFI回避

SFI とは

FUJITSU

- SFI (Store Fetch Interlock) とは
  - 先行するstore命令と後続のload命令のアドレスが重なっている場合、loadがstoreを追い越さないようにインターロックをかける制御機構
  - 基本的にはstoreが行われるアドレスのみにロックがかかる。

- 過剰 SFI について

  コンパイラにより回避することが可能(回避したケースを後述する)

  以下のケースで発生する (詳細は後述する)
  - マスク付SIMDのケースで、マスク判定が0(storeしない)のアドレスもロックされる。
  - GatherLoadの纏め上げ機能が動作した場合、GatherLoadでのSFI確認対象がキャッシュライン(に含まれる全要素)になる。実際にはloadを行わないアドレスのSFIを検出しロックされていると判断する場合がある。

- CPU解析レポートにより SFI の発生は確認可能

過剰SFIになる要因とSFIプリチェックをする要因

FUJITSU

過剰 SFI になる要因

要因	過剰SFIの内容	関連アプリ
プレディケートマスク=0	プレディケートマスクが0になっている要素のストアもSFI対象になる。	ADVENTURE
Gather Load命令の纏め上げ	Gather Load命令で纏め上げ機能が動作した場合、SFI確認対象がキャッシュラインになる。	GENESIS
ロード命令で4KB境界を跨ぐアクセス	ロード命令で4KB境界を跨ぐアクセスの場合、常に物理アドレスのbit11-0でSFIチェックが行われる。	
SIMDの全ての要素サイズで、アドレスが4B境界にアラインされていない場合	常にデータ長を4B境界に合わせてSFIチェックする。	
Multiple Structure命令	・SFI確認対象がキャッシュラインになる（要素サイズ4B/8B）。 ・VL=0,1でもストアのSFI確認対象がVL=3と同じ（要素サイズ1B/2B）。	

SFI プリチェックをする要因

要因	SFIプリチェックの内容	関連アプリ
先行するストア命令がパイプラインに存在する場合	先行するストア命令がパイプラインに存在する場合、物理アドレスのbit11-0でSFIプリチェックが行われる。	on L1$の配列アクセス
先行するストア命令がL1D$ミスしているケース	ストアのデータがL1D$に到着するまで、物理アドレスのbit11-0でSFIプリチェックが行われる。ロード命令の再投入はストアデータ到着後。	

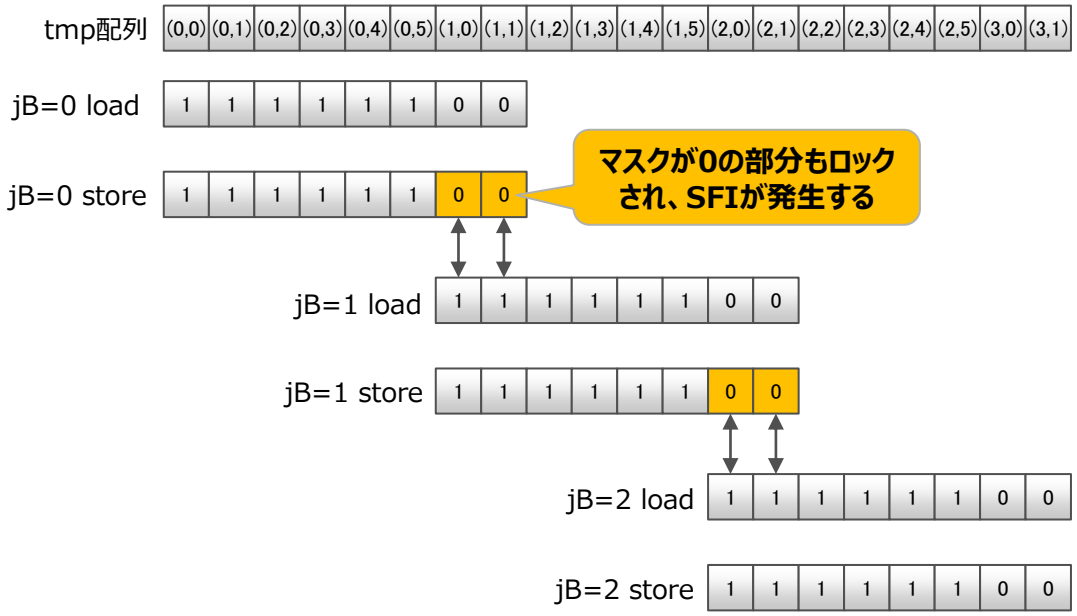
FUJITSU

測定ケースの説明

- アクセス命令のマスク値が0のケースの過剰SFI
- Gather load命令纏め上げ機能によりSFI対象がキャッシュラインになるケースの過剰SFI
- SIMDのアドレスが4B境界にないケースの過剰SFI
- ロード命令で4KB境界を跨ぐケースの過剰SFI
- Multiple Structure の過剰SFI
- SFIプリチェックをする要因
- マスク値が０のケース簡単例
- マスク値が０のケースの性能影響

アクセス命令のマスク値が0のケースの過剰SFI

```
for (jBlock = 0; jBlock < iBlock; jBlock++) {
  for ( iv = 0; iv < 6; iv++ ){
    double v_j = vector2[jBlock][iv];
    double m_i0_j = OffDiagComponents[instanceId][offset + jBlock][0][iv];
    double m_i1_j = OffDiagComponents[instanceId][offset + jBlock][1][iv];
    double m_i2_j = OffDiagComponents[instanceId][offset + jBlock][2][iv];
    double m_i3_j = OffDiagComponents[instanceId][offset + jBlock][3][iv];
    double m_i4_j = OffDiagComponents[instanceId][offset + jBlock][4][iv];
    double m_i5_j = OffDiagComponents[instanceId][offset + jBlock][5][iv];
    tmp[threadId][jBlock][iv]
              += m_i0_j * v_i0 + m_i1_j * v_i1 + m_i2_j * v_i2
               + m_i3_j * v_i3 + m_i4_j * v_i4 + m_i5_j * v_i5;
  }
}
```



Gather load命令纏め上げ機能によりSFI対象がキャッシュラインになるケースの過剰SFI

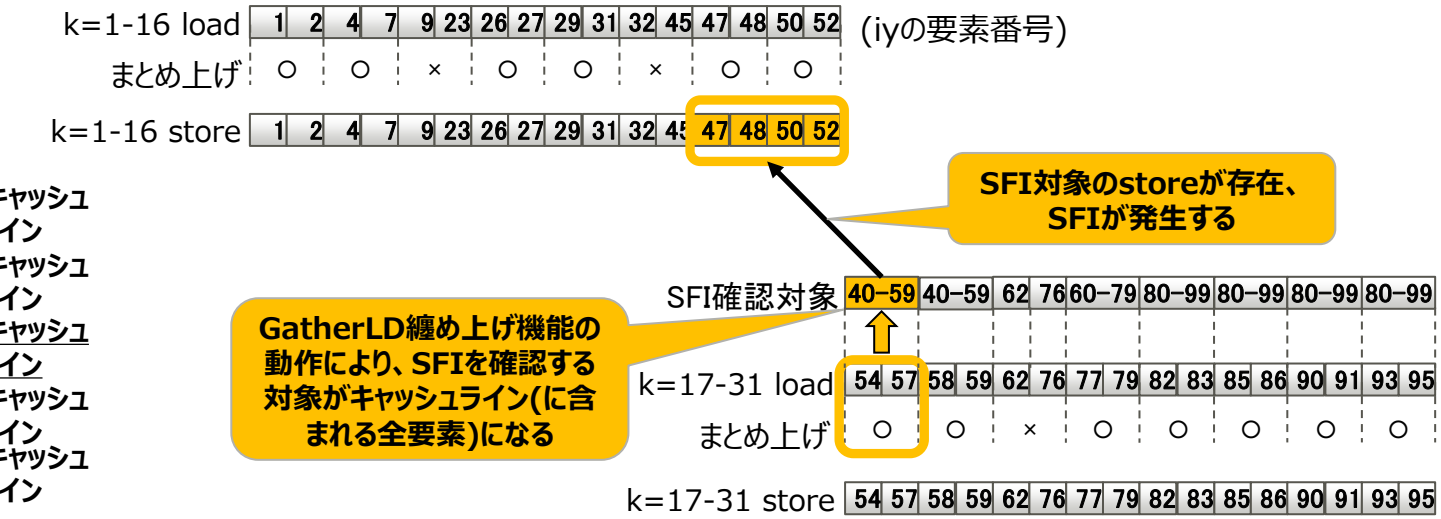
```
do k = 1, num_nb15_calc(ix,i,ik)

  ij = nb15_calc_list(k,ix,i,ik)
  j  = int(real(ij)*inv_MaxAtom)
  iy = ij - j*MaxAtom

  ...

  force(1,iy,j,id+1,ik) = force(1,iy,j,id+1,ik) + work(1)
  force(2,iy,j,id+1,ik) = force(2,iy,j,id+1,ik) + work(2)
  force(3,iy,j,id+1,ik) = force(3,iy,j,id+1,ik) + work(3)

end do
```

注意事項：
1. 配列のリスト値は、簡単のため、説明しやすい値に変更。
2. force(1,..), force(2,..), force(3,..)の間でも、過剰SFIが発生するが、説明は省略。

配列のiyの要素番号と128Bアラインの関係

128B

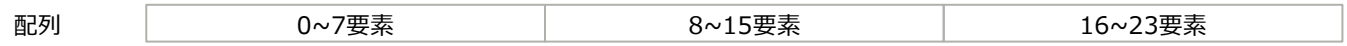
0	1	2	…	9	同一キャッシュライン
10	11	12	…	19	
20	21	22	…	29	同一キャッシュライン
30	31	32	…	39	
40	41	42	…	49	同一キャッシュライン
50	51	52	…	59	
60	61	62	…	69	同一キャッシュライン
70	71	72	…	79	
80	81	82	…	89	同一キャッシュライン
90	91	92	…	99	



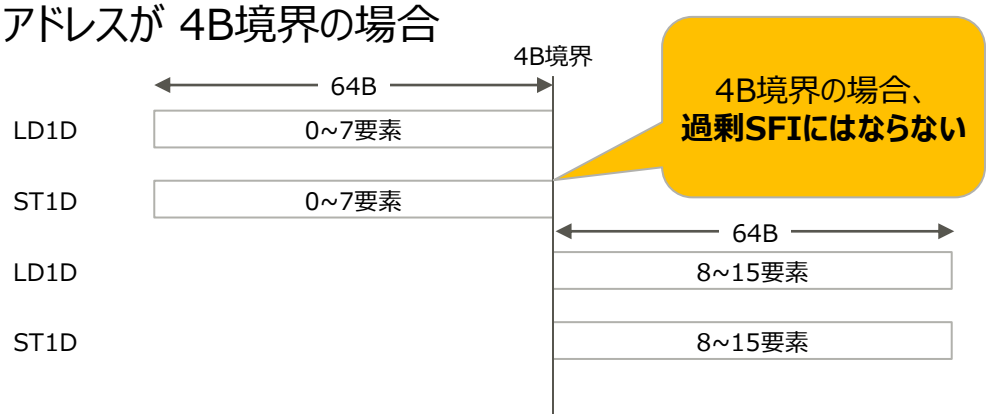
SIMDのアドレスが4B境界にないケースの過剰SFI

- SIMDの全ての要素サイズで、アドレスが4B境界にアラインされていない場合に、SFIの確認対象が常に4B境界となる。

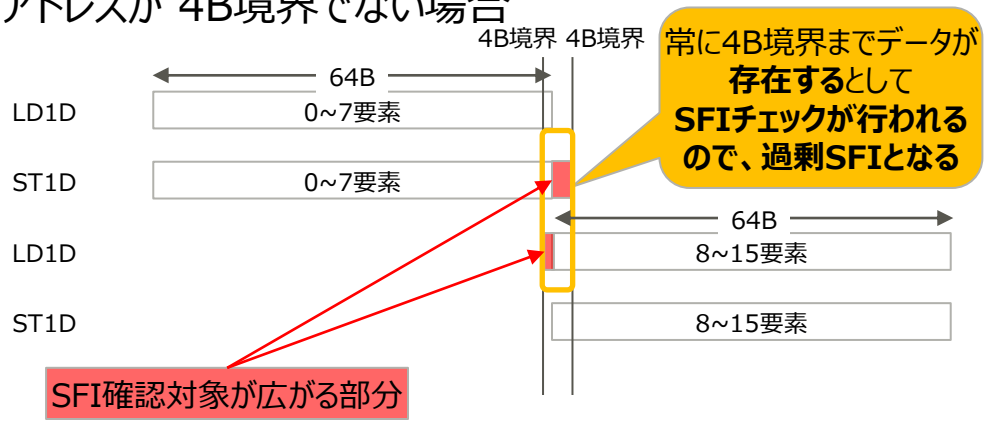
  例：倍精度で配列の要素を読み込んで更新するケース



- アドレスが 4B境界の場合



- アドレスが 4B境界でない場合



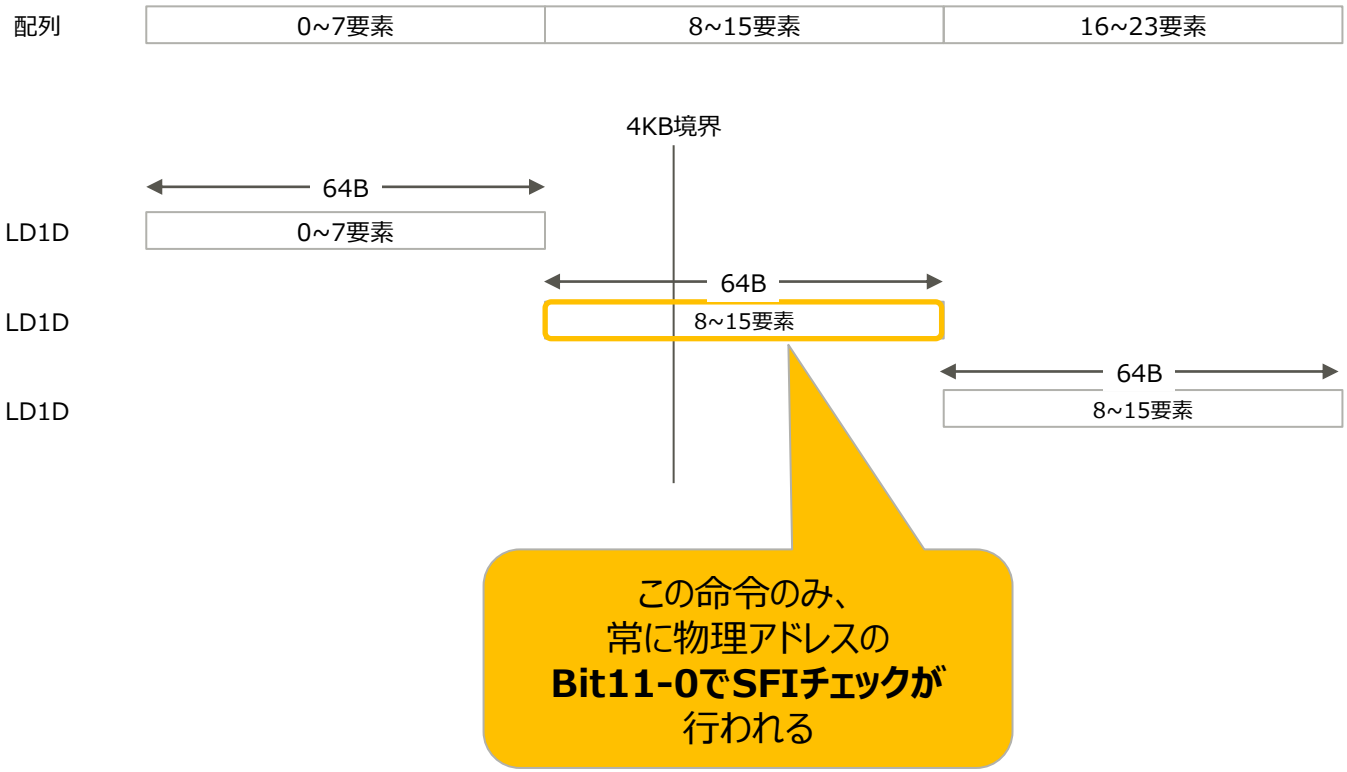
- ✓ 4B/8B要素の場合には、基本的に4B/8B境界になっているので、問題はない。
- ✓ FP16や整数の時には、注意が必要。但し、ソフトウェアパイプライニングで緩和できる可能性あり。

ロード命令で4KB境界を跨ぐケースの過剰SFI

FUJITSU

- 1つのロード命令で4KB境界を跨ぐアクセスをした場合、常に物理アドレスのbit11-0 でSFIチェックが行われる。

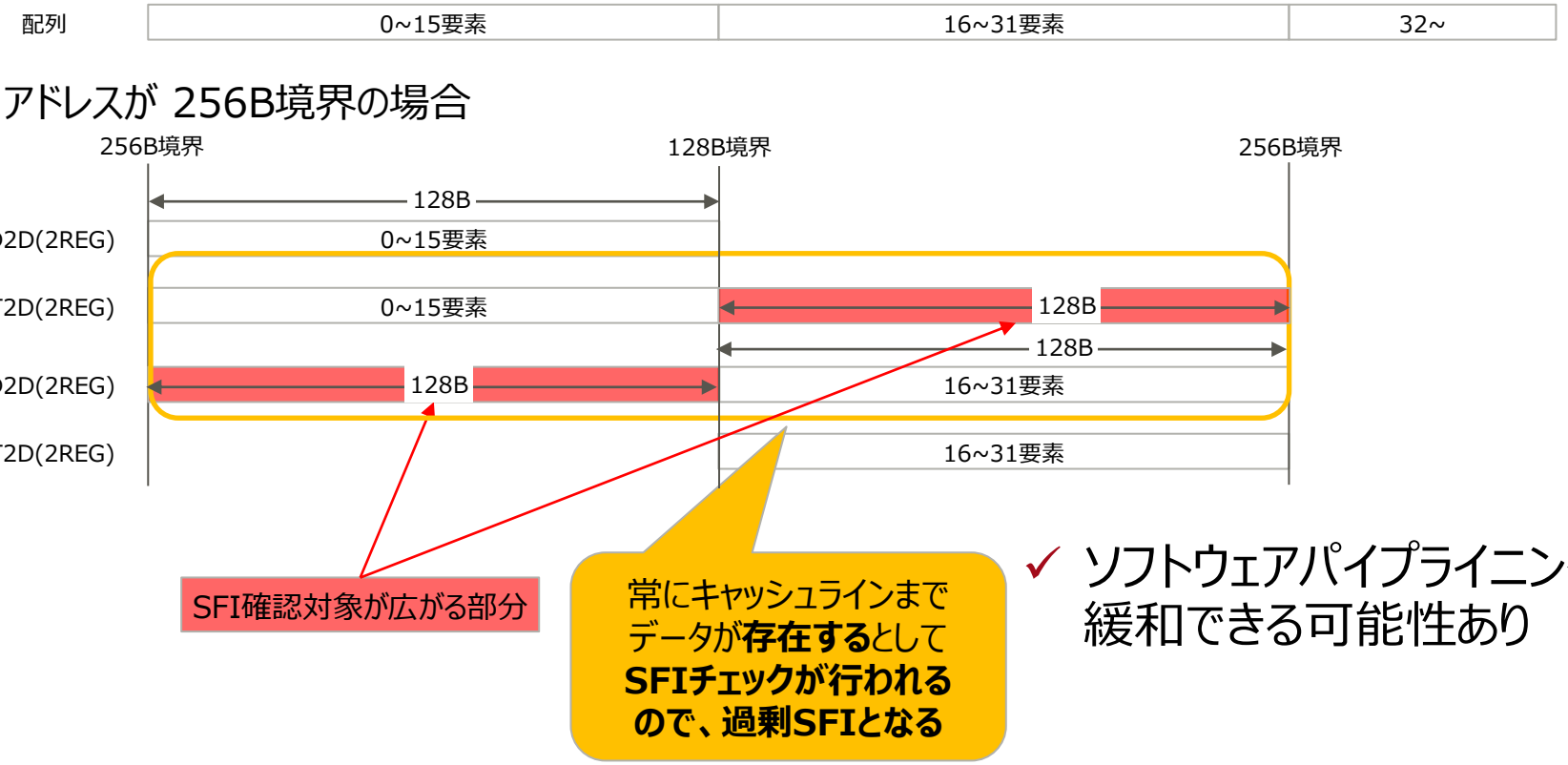
  例：倍精度で配列の要素を読み込むケース



Multiple Structure の過剰SFI (1/2)

- 要素サイズが 4B/8B の Multiple Structure Store/Load 命令において、SFIの対象確認がキャッシュラインになる。

  例：LD2D命令/ST2D命令で配列の要素を読み込んで更新するケース



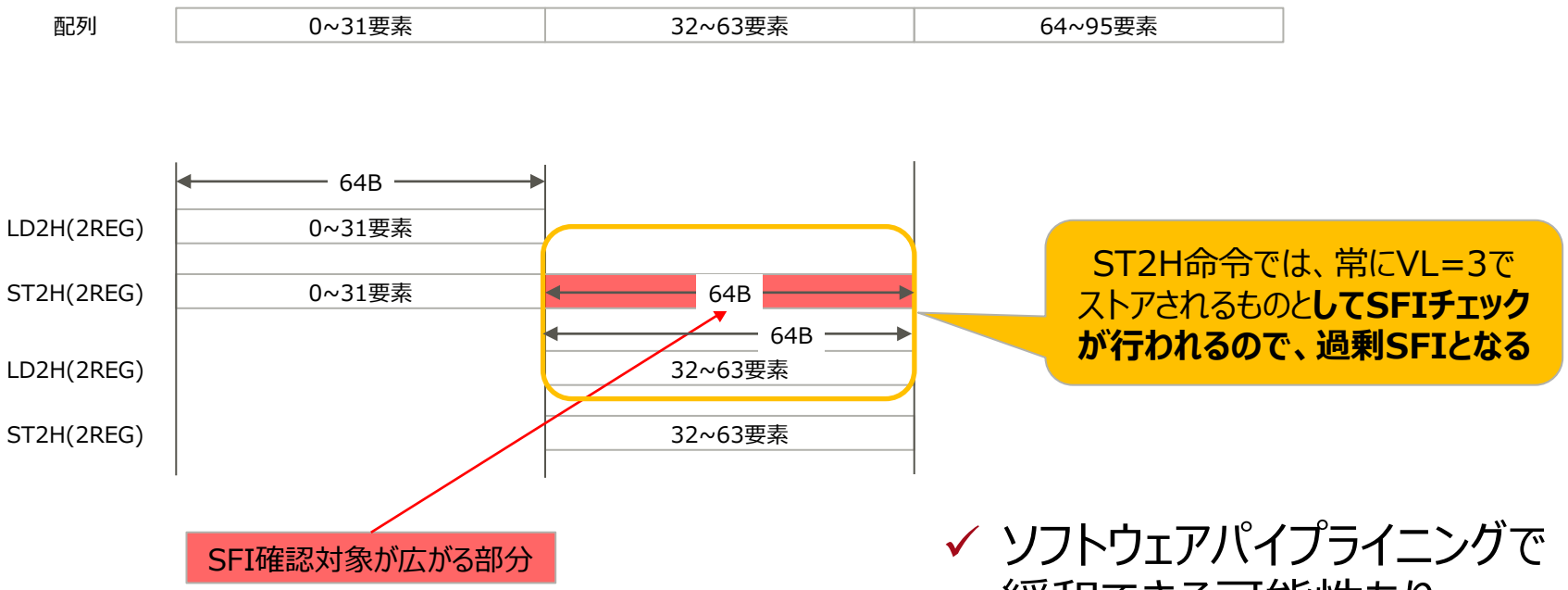
✓ ソフトウェアパイプライニングで緩和できる可能性あり

Multiple Structure の過剰SFI (2/2)

FUJITSU

- 要素サイズが 1B/2B の Multiple Structure Store 命令において、VL（ベクトルレングス）が 0 (128bitSIMD) , 1 (256bitSIMD) の時、SFIの対象確認がVL=3 (512bitSIMD) の時と同じになる。

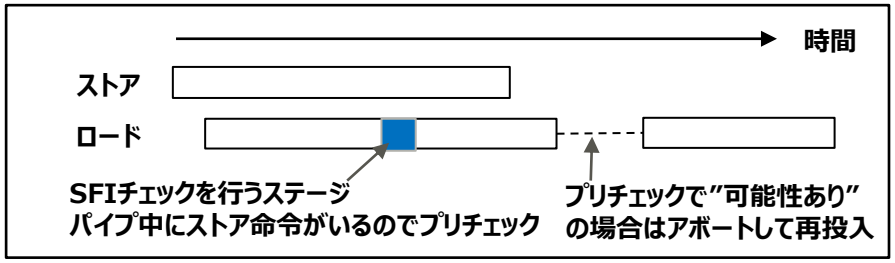
  例：VL=1 (256bitSIMD) の時、LD2H命令/ST2H命令で配列の要素を読み込んで更新するケース



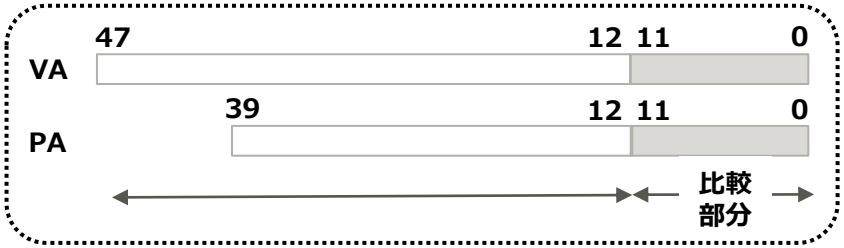
✓ ソフトウェアパイプライニングで緩和できる可能性あり

SFIプリチェックをする要因

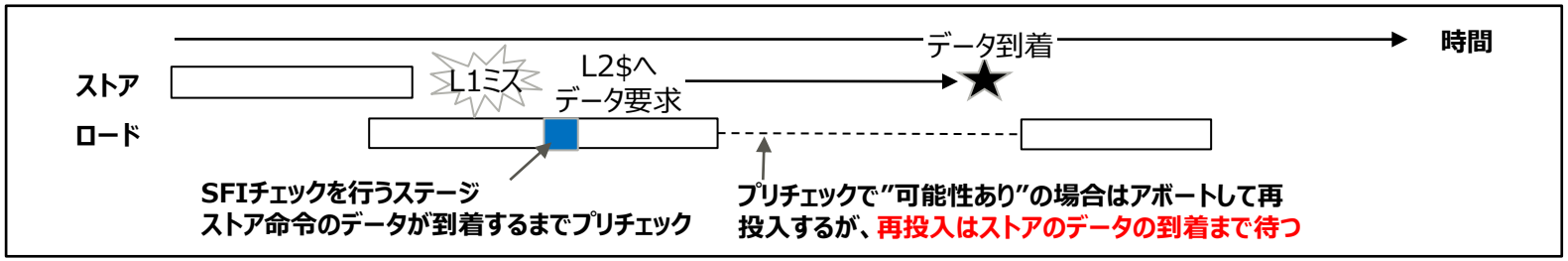
① ロード命令のパイプライン処理中に、先行するストア命令がパイプラインに存在する場合



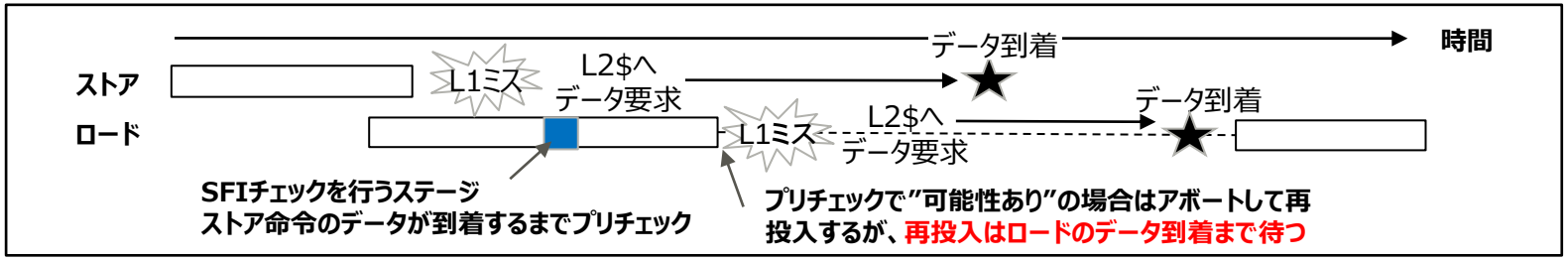
SFIプリチェックに用いるアドレス



② 先行するストア命令がL1D$キャッシュミスして、L2$からデータを持ってくるケース１
（ロード命令がL1D$ヒットするケース：ストアのデータ到着までプリチェック+待ち）



③ 先行するストア命令がL1D$キャッシュミスして、L2$からデータを持ってくるケース２
（ロード命令がL1D$ミスするケース：ストアのデータ到着までプリチェック+待ち）



✓ L2$へデータ要求は行われるので性能影響はないが、PAのSFIカウントが更新される

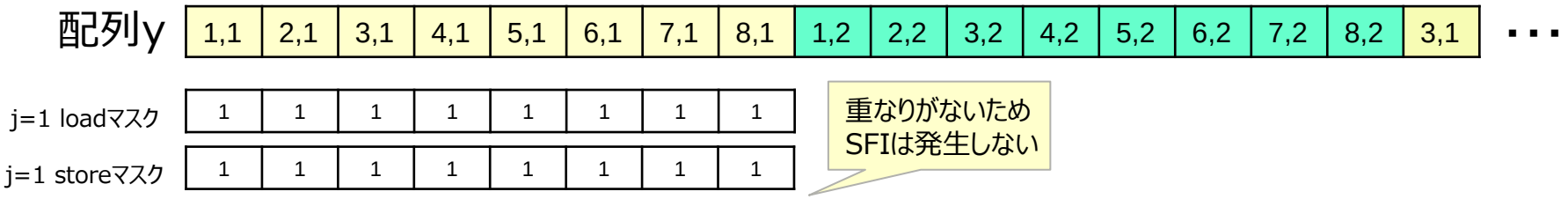
マスク値が０のケース簡単例

■アクセスイメージ

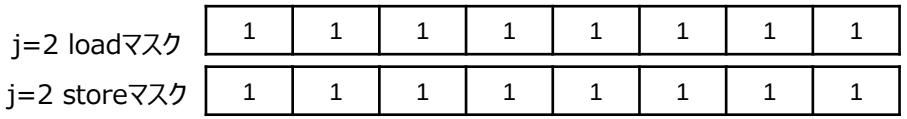
測定コード

```
real*8 y(X,n), x1(X,n)    <- X=8,6
  Do k = 1, iter
    Do j = 1, n
      Do i = 1, X         <- X=8,6
        y(i,j) = y(i,j) + x1(i,j)
      End Do
    End Do
  End Do
```

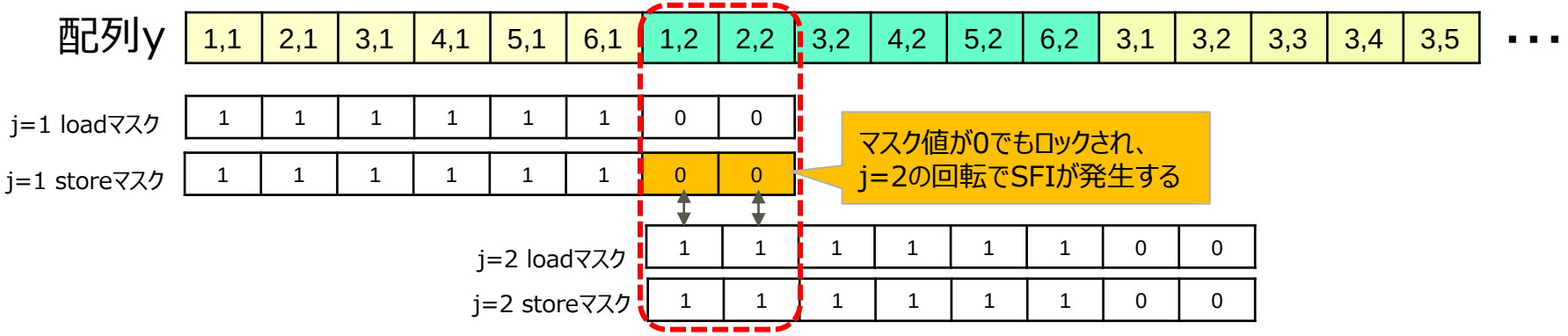
■X=8 (SFI発生しない)



重なりがないため
SFIは発生しない



■X=6 (SFI発生)



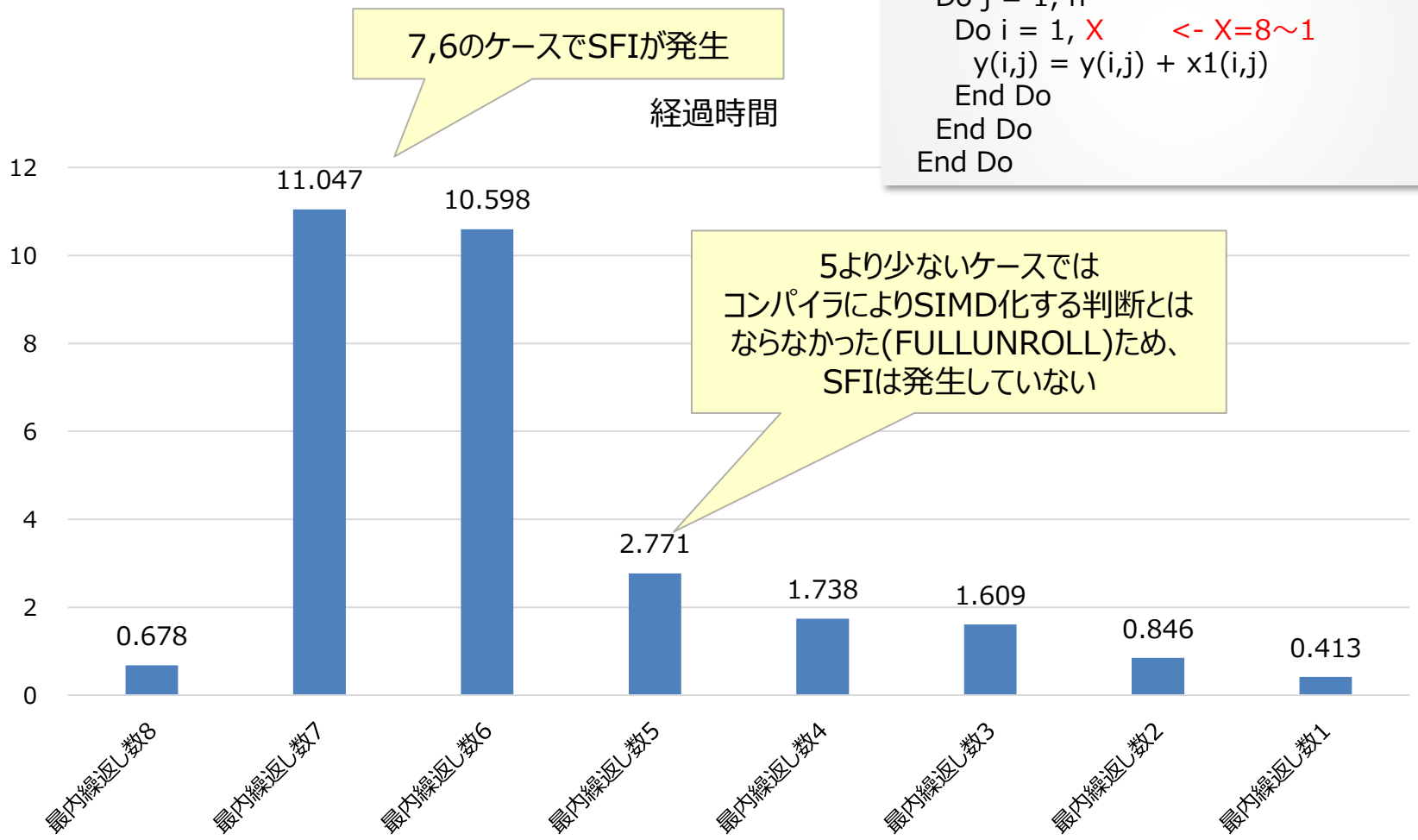
マスク値が0でもロックされ、
j=2の回転でSFIが発生する

マスク値が０のケースの性能影響

■ 測定結果

測定コード

```
real*8 y(X,n), x1(X,n)    <- X=8～1
  Do k = 1, iter
    Do j = 1, n
      Do i = 1, X         <- X=8～1
        y(i,j) = y(i,j) + x1(i,j)
      End Do
    End Do
  End Do
```



FUJITSU

コンパイラによる過剰のSFI回避

- マスク値が０のケース：コンパイラによる過剰のSFI回避
- マスク値が０のケース：コンパイラによる過剰のSFI回避結果

マスク値が０のケース：コンパイラによる過剰のSFI回避

FUJITSU

■ コンパイラのスケジューリングにより回避可能

最内ループでSIMD化した際に外側ループでのスケジューリング(SWPL)を促進し過剰SFIの回避を行う

改善前

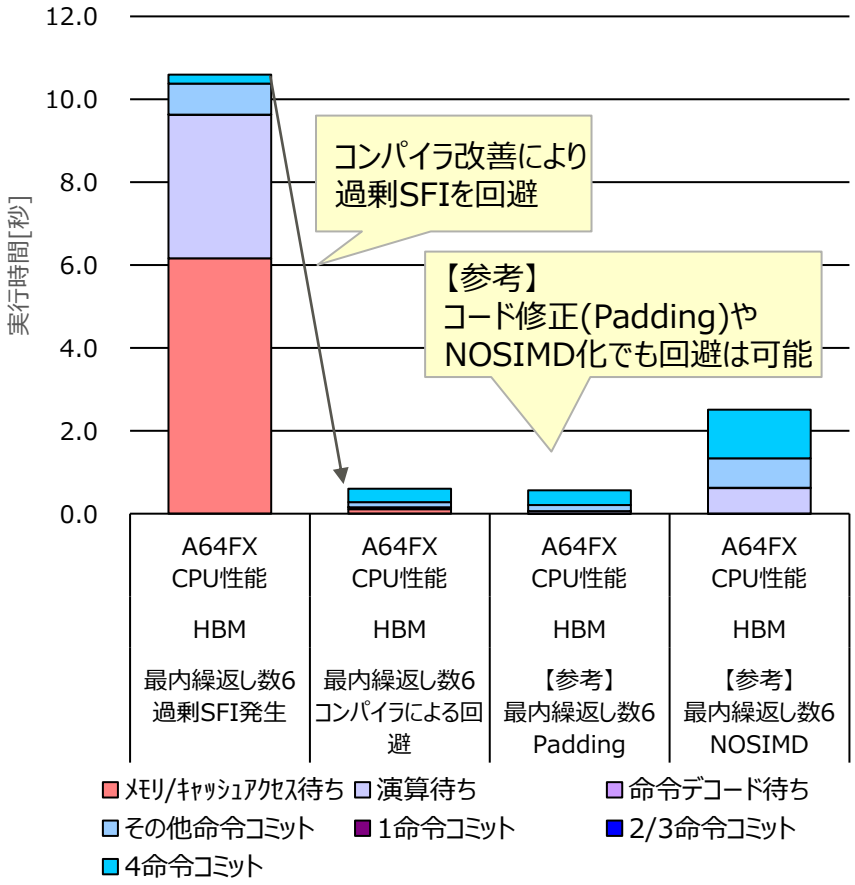
```
 5   1            Do k = 1, iter
 6   2      2       Do j = 1, n
                 <<< Loop-information Start >>>
                 <<<  [OPTIMIZATION]
                 <<<    SIMD(VL: 8)
                 <<< Loop-information  End >>>
 7   3      2v        Do i = 1, 6
 8   3      2v          y(i,j) = y(i,j) + x1(i,j)
 9   3      2v        End Do
10   2      2       End Do
11   1            End Do
```

改善(回避対応)後

```
 5   1            Do k = 1, iter
                 <<< Loop-information Start >>>
                 <<<  [OPTIMIZATION]
                 <<<    SOFTWARE PIPELINING
                 <<< Loop-information  End >>>
 6   2    2       Do j = 1, n
                 <<< Loop-information Start >>>
                 <<<  [OPTIMIZATION]
                 <<<    SIMD(VL: 8)
                 <<< Loop-information  End >>>
 7   3    2v        Do i = 1, 6
 8   3    2v          y(i,j) = y(i,j) + x1(i,j)
 9   3    2v        End Do
10   2    2       End Do
11   1            End Do
```

マスク値が０のケース：コンパイラによる過剰のSFI回避結果

■ コンパイラによる回避、測定結果



	A64FX CPU 性能実機	A64FX CPU 性能実機	A64FX CPU 性能実機	A64FX CPU 性能実機
ソースコード版数	最内繰返し数6 過剰SFI 発生	最内繰返し数6 コンパイラ による回避	【参考】 最内繰返し数6 Padding	【参考】 最内繰返し数6 NOSIMD
浮動小数点精度	倍精度	倍精度	倍精度	倍精度
SIMD幅	8	8	8	1
スレッド数	1	1	1	1
集計スレッド番号	0	0	0	0
実行時間[s]	1.06E+01	6.02E-01	5.65E-01	2.51E+00
有効総命令数	3.47E+09	2.81E+09	3.09E+09	1.08E+10
GFLOPS(プロセス)	0.29	5.10	5.44	0.92
メモリスループット [GB/s/プロセス]	0.00	0.00	0.00	0.00
L1ビジー率/スレッド	42.25%	80.41%	88.26%	99.49%
SFI(/cycle)/スレッド	0.68	0.03	0.00	0.00

マスク値が0のケースによる過剰SFIは回避することができる。

PMUイベント

- CPU性能解析レポートの特徴
- CPU解析レポートの京・FX100製品からの改良点
- CPU性能解析レポートの表示例
  - CPU性能解析レポート 表示イメージ：全体
  - CPU性能解析レポート 表示イメージ：グラフ
  - CPU性能解析レポート 表示イメージ：表
  - DGEMM ：表示例・分析例
  - STREAM：表示例・分析例
- FLOPS に関する注意事項

CPU性能解析レポートの特徴

- PMUイベントに基づきCPUの性能情報をエクセルで表示
- 主要データは5回の計測でも採取した性能情報を表示可能
- 推奨は11回の計測で性能情報を表示(FX100相当)
- 追加計測することで、性能情報を詳細化可能

計測回数に対する採取可能な性能情報

性能情報のレベル	単体レポート	簡易レポート	標準レポート	詳細レポート
計測回数	1	5	11	17
統計情報	✓	✓※	←	←
サイクルアカウンティング		✓	✓※	←
メモリ・キャッシュビジー状況		✓	✓※	✓※
キャッシュミス状況		✓	✓※	←
命令ミックス		✓	✓※	✓※
負荷不均衡		✓	←	←
消費電力量			✓	←
ハードウェアプリフェッチ情報				✓
CMG間データ転送状況				✓
その他の性能情報				✓

(※) レベルが高いほど情報量が増加

CPU解析レポートの京・FX100製品からの改良点

FUJITSU

改良項目	京、FX100製品	A64FXでの改良点
計測回数に応じた性能情報表示	京では7回計測、FX100では11回計測しなければ、精密PA可視化機能（CPU性能解析レポート）を使用不可	1回、5回の計測でもCPU性能解析レポートとして表示可能 11回、17回の計測で追加情報の表示が可能
		サイクルアカウンティングなど追加計測でCPU性能情報の詳細化可能
情報量の増加	---	様々な情報を追加 - 浮動小数点演算、整数演算パイプラインのビジー - L2キャッシュミスの内訳(dmミス率、swpfミス率、hwpfミス率) - Predicate マスク情報 - Gather 纏め上げ数 - Spill/Fill 数 - CMG間アクセス量 - 電力　　等
レイアウト	---	従来製品に比べて情報量が増えたが、主要情報をA3の１ページに表示

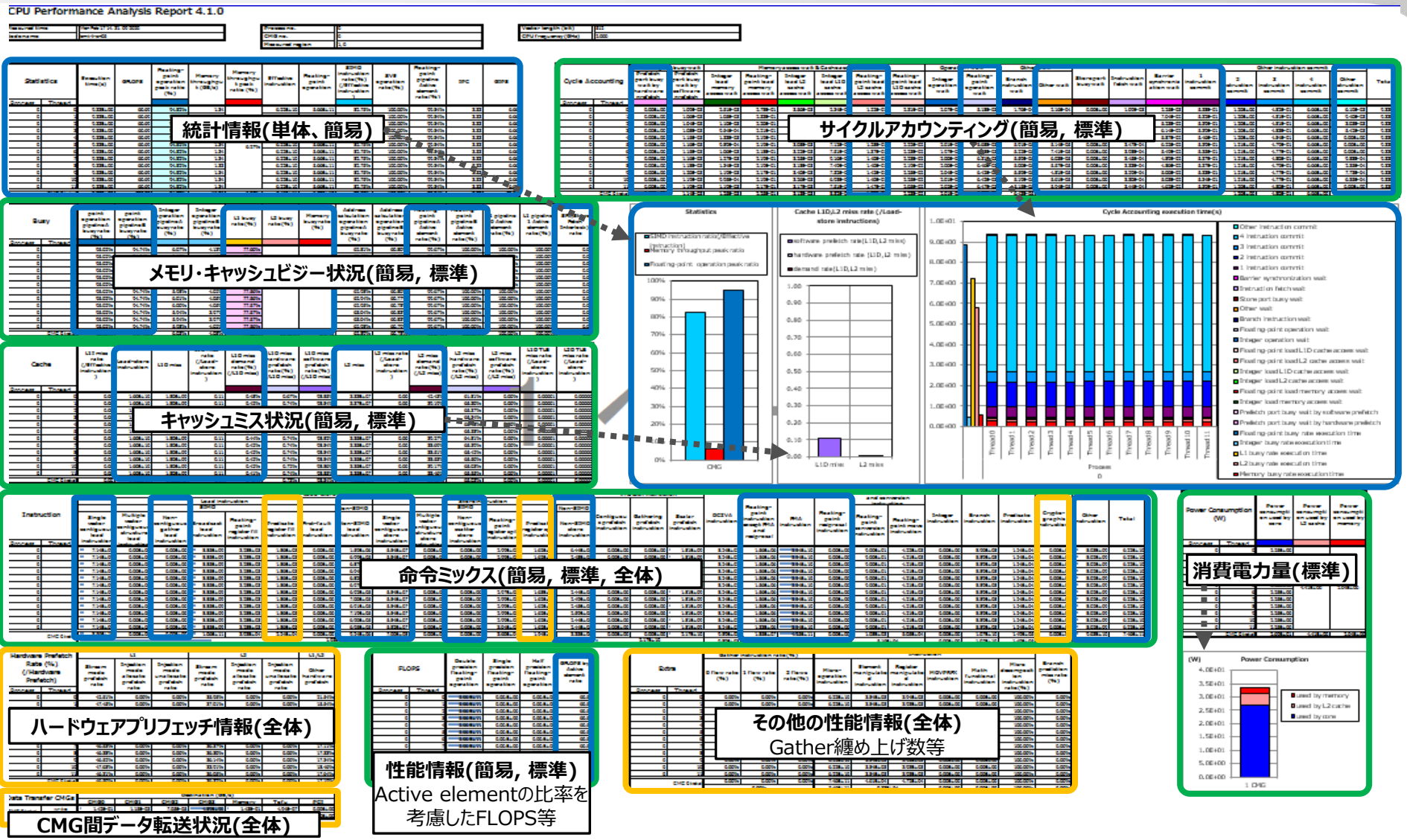
CPU性能解析レポートの表示例

FUJITSU

- 対象アプリケーション
  - DGEMM
  - STREAM

- 計測条件は以下の通り
  - 動作環境
    - A64FX
  - CPU性能解析情報採取のレベル・計測回数
    - 簡易情報(5回計測)

CPU性能解析レポート 表示イメージ (全体)

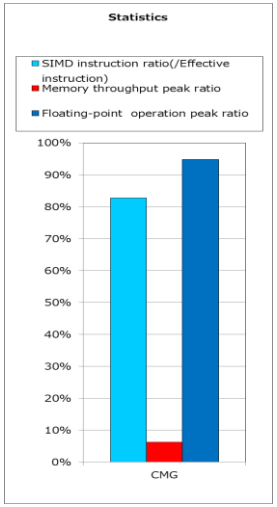


データ収集の回数によって、単体、簡易, 標準, 全体の順に表・グラフが詳細化します。
（下記のような各色の枠は実際のCPU性能解析レポートにはありません）

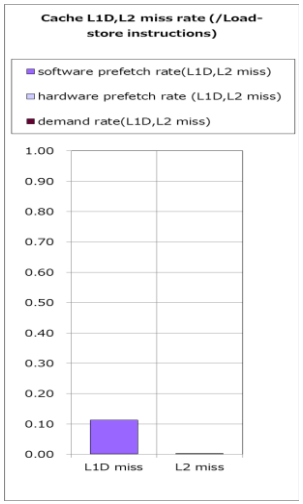
簡易　標準　詳細

CPU性能解析レポート 表示イメージ (グラフ)

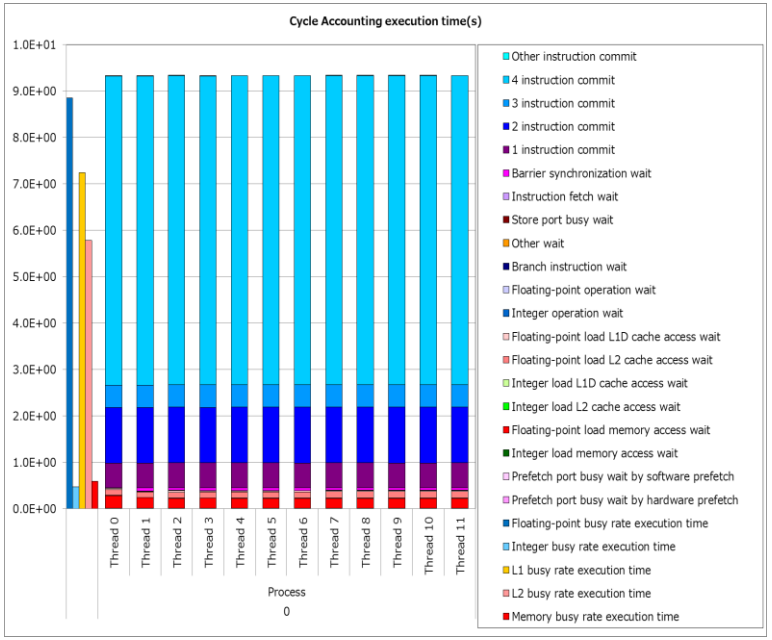
- 下記情報を見るだけで大まかな性能状態の把握が可能
  - 演算ピーク・スループットピーク・SIMD比



  - キャッシュ状況



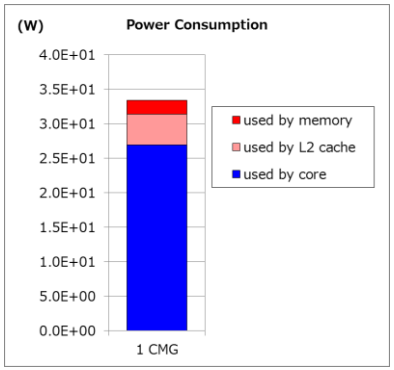
  - サイクルアカウンティング



  - CMG間データ転送状況

Data Transfer CMGs		Destination (GB/s)						
		CMG0	CMG1	CMG2	CMG3	Memory	Tofu	PCI
CMG 0 total	write	1.42E-01	1.15E-03	7.05E-05	4.89E+00	1.43E-01	4.94E-07	0.00E+00
	read					4.53E+00	5.49E-07	0.00E+00

  - 消費電力量



CPU性能解析レポート 表示イメージ (表1)

FUJITSU

統計情報

Statistics		Execution time (s)	GFLOPS	Floating-point operation peak ratio (%)	Memory throughput (GB/s)	Memory throughput peak ratio (%)	Effective instruction	Floating-point operation	SIMD instruction rate (%) (/Effective instruction)	SVE operation rate (%)	Floating-point pipeline Active element rate (%)	IPC	GIPS
Process	Thread												
0	0	9.33E+00	60.69	94.82%	1.34		6.2	5.66E+11	82.75%	100.00%	99.84%	3.33	6.66
0	1	9.33E+00	60.69	94.82%	1.36						99.84%	3.33	6.66
0	2	9.33E+00	60.69	94.82%	1.35						.84%	3.33	6.66
0	3	9.33E+00	60.69	94.82%	1.33						.84%	3.33	6.66
0	4	9.33E+00	60.69	94.82%	1.33						.84%	3.33	6.66
0	5	9.33E+00	60.69	94.82%	1.34						.84%	3.33	6.66
0	6	9.33E+00	60.69	94.82%	1.34	6					99.84%	3.33	6.66
0	7	9.33E+00	60.69	94.82%	1.34		6.22E+10	5.66E+11	82.75%	100.00%	99.84%	3.33	6.66
0	8	9.33E+00	60.69	94.82%	1.33		6.22E+10	5.66E+11	82.75%	100.00%	99.84%	3.33	6.66
0	9	9.33E+00	60.69	94.82%	1.34		6.22E+10	5.66E+11	82.75%	100.00%	99.84%	3.33	6.66
0	10	9.33E+00	60.69	94.82%	1.34		6.22E+10	5.66E+11	82.75%	100.00%	99.84%	3.33	6.66
0	11	9.33E+00	60.69	94.82%	1.34		6.22E+10	5.66E+11	82.75%	100.00%	99.84%	3.33	6.66
CMG 0 total		9.33E+00	728.23	94.82%	16.06	6.27%	7.46E+11	6.79E+12	82.75%	100.00%	99.84%	3.33	79.93

実行時間、浮動小数点演算数、
メモリスループット、SIMD命令率など
主要な統計情報が表示される

サイクルアカウンティング

Cycle Accounting		Prefetch port busy wait		Memory access wait & Cache access wait						Operation wait		Other wait		Store port busy wait	Instruction fetch wait	Barrier synchronization wait	1 instruction commit	Other instruction commit				Total
		Prefetch port busy wait by hardware prefetch	Prefetch port busy wait by software prefetch	Integer load memory access wait	Floating-point load memory access wait	Integer load L2 cache access wait	Integer load L1D cache access wait	Floating-point load L2 cache access wait	Floating-point load L1D cache access wait	Integer operation wait	Floating-point operation wait	Branch instruction wait	Other wait					2 instruction commit	3 instruction commit	4 instruction commit	Other instruction commit	
Process	Thread																					
0	0	0.00E+00	1.09E-03	2.51E-03	2.78E-01	3.56E-03	2.34E-03	1.22E-01	2.31E-02	2.07E-03	5.18E-03	1.76E-04	2.16E-04	0.00E+00	1.09E-03	2.25E-03	5.38E-01	1.20E+00	4.82E-01	6.66E+00	6.10E-03	9.33E+00
0	1	0.00E+00	1.06E-03	1.05E-03	2.33E-01	2.65E-03	6.65E-04	1.20E-01	8.50E-03	1.88E-03	5.08E-03	5.14E-0		0.00E+00	3.78E-04	7.04E-02	5.32E-01	1.20E+00	4.81E-01	6.66E+00	9.40E-03	9.33E+00
0	2	0.00E+00	1.04E-03	1.10E-03	2.20E-01	2.93E-03	7.80E-04	1.22E-01	3.67E-02	2.11E-03	5.32E-03	5.60F			3.47E-04	5.23E-02	5.39E-01	1.20E+00	4.81E-01	6.66E+00	3.03E-03	9.33E+00
0	3	0.00E+00	1.08E-03	9.54E-04	2.21E-01	2.89E-03	7.74E-04	1.26E-01	2.25E-02	2.01E-0											E-03	9.33E+00
0	4	0.00E+00	1.15E-03	1.33E-03	2.19E-01	2.96E-03	7.15E-04	1.31E-01	2.22E-02	2.02E-											+00	9.33E+00
0	5	0.00E+00	1.16E-03	9.89E-04	2.19E-01	3.08E-03	7.12E-04	1.28E-01	2.22E-02	2.01E-											+00	9.33E+00
0	6	0.00E+00	1.15E-03	1.06E-03	2.18E-01	3.22E-03	7.51E-04	1.37E-01	2.23E-02	1.97E-											+00	9.33E+00
0	7	0.00E+00	1.16E-03	1.27E-03	2.19E-01	3.25E-03	9.16E-04	1.40E-01	2.23E-02	2.00E-											-04	9.33E+00
0	8	0.00E+00	1.18E-03	1.34E-03	2.19E-01	3.18E-03	7.49E-04	1.40E-01	2.19E-02	2.00E-											-04	9.33E+00
0	9	0.00E+00	1.20E-03	1.19E-03	2.17E-01	3.40E-03	7.32E-04	1.42E-01	2.25E-02	2.04E-											-04	9.33E+00
0	10	0.00E+00	1.19E-03	9.95E-04	2.19E-01	3.26E-03	6.93E-04	1.40E-01	2.25E-02	2.01E-0											E-04	9.33E+00
0	11	0.00E+00	1.19E-03	1.19E-03	2.17E-01	3.17E-03	7.51E-04	1.47E-01	2.05E-02	2.02E-03	6.47E-03	5.15E-05	5.94E-05	0.00E+00	3.44E-04	4.62E-02	5.39E-01	1.20E+00	4.80E-01	6.66E+00	0.00E+00	9.33E+00
CMG 0 total		0.00E+00	1.14E-03	1.25E-03	2.25E-01	3.13E-03	8.82E-04	1.33E-01	2.23E-02	2.01E-03	5.93E-03	6.43E-05	7.01E-05	0.00E+00	4.14E-04	5.05E-02	5.37E-01	1.20E+00	4.80E-01	6.66E+00	2.22E-03	9.33E+00

メモリ・キャッシュビジー待ち時間、浮動小数点演算待ち時間、
バリア同期待ち時間、命令コミット時間など
積み上げグラフの数値(秒)が表示される

CPU性能解析レポート 表示イメージ (表2)

FUJITSU

メモリ・キャッシュビジー情報

Busy		Floating-point operation pipeline A busy rate (%)	Floating-point operation pipeline B busy rate (%)	Integer operation pipeline A busy rate (%)	Integer operation pipeline B busy rate (%)	L1 busy rate (%)	L2 busy rate (%)	Memory busy rate (%)	Address calculation operation pipeline A busy rate (%)	Address calculation operation pipeline B busy rate (%)	Floating-point pipeline A Active element rate (%)	Floating-point pipeline B Active element rate (%)	L1 pipeline 0 Active element rate (%)	L1 pipeline 1 Active element rate (%)	SFI(Store Fetch Interlock) rate
Process	Thread														
0	0	95.02%	94.74%	6.07%	4.17				62.81%	60.80%	99.67%	100.00%	100.00%	100.00%	0.00
0	1	95.02%	94.74%												
0	2	95.02%	94.74%												
0	3	95.02%	94.74%												
0	4	95.02%	94.74%												
0	5	95.02%	94.74%												
0	6	95.02%	94.74%	5.98%	4.02%	77.56%	61.93%	6.27%	62.98%	60.80%					
0	7	95.02%	94.74%	6.01%	4.05%	77.56%			62.94%	60.77%					
0	8	95.02%	94.74%	6.00%	4.05%	77.57%			62.95%	60.78%					
0	9	95.02%	94.74%	5.94%	3.97%	77.57%			63.04%	60.83%					
0	10	95.02%	94.74%	5.94%	3.97%	77.57%			63.04%	60.83%					
0	11	95.02%	94.74%	5.98%	4.02%	77.56%			62.98%	60.79%					
CMG 0 total		95.02%	94.74%	6.03%	4.08%	77.56%	61.93%	6.27%	62.89%	60.78%					

L1ビジー率、L2ビジー率、メモリビジー率、
浮動小数点演算・整数演算パイプラインビジー率など

- メモリビジー率について
  - 京・FX100のPAシートから、分母としている数値が変更(1CMGあたりのメモリバンド幅理論性能値(256GB/s))となった。
  - 80%程度でビジーと考える

キャッシュミス状況

Cache		L1I miss rate (/Effective instruction)	Load-store instruction	L1D miss	L1D miss rate (/Load-store instruction)	L1D miss demand rate (%) (/L1D miss)	L1D miss hardware prefetch rate (%) (/L1D miss)	L1D miss software prefetch rate (%) (/L1D miss)	L2 miss	L2 miss rate (/Load-store instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)	L1D TLB miss rate (/Load-store instruction)	L2D TLB miss rate (/Load-store instruction)
Process	Thread														
0	0	0.00	1.60		0.11	0.48%	0.67%	98.85%	3.33E+07	0.00	42.43%	61.81%	0.00%	0.00001	0.00000
0	1								07	0.00	39.19%	65.30%	0.00%	0.00001	0.00000
0	2								07	0.00	38.98%	65.37%	0.00%	0.00001	0.00000
0	3								07	0.00	39.04%	65.34%	0.00%	0.00001	0.00000
0	4								07	0.00	38.74%	65.48%	0.00%	0.00001	0.00000
0	5	0.00	1.60E+10	1.80E+09	0.11	0.42%	0.74%	98.84%	3.35E+07	0.00	38.94%	65.38%	0.00%	0.00001	0.00000
0	6	0.00	1.60E+10	1.80E+09	0.11	0.44%	0.74%	98.82%	3.35E+07	0.00	39.27%	64.81%	0.00%	0.00001	0.00000
0	7	0.00	1.60E+10	1.80E+09	0.11	0.42%	0.74%	98.84%	3.35E+07	0.00	38.69%	65.39%	0.00%	0.00001	0.00000
0	8	0.00	1.60E+10	1.80E+09	0.11	0.42%	0.74%	98.84%	3.35E+07	0.00	38.51%	65.42%	0.00%	0.00001	0.00000
0	9	0.00	1.60E+10	1.80E+09	0.11	0.42%	0.74%	98.84%	3.35E+07	0.00	38.53%	65.50%	0.00%	0.00001	0.00000
0	10	0.00	1.60E+10	1.80E+09	0.11	0.42%	0.72%	98.86%	3.35E+07	0.00	39.17%	65.03%	0.00%	0.00001	0.00000
0	11	0.00	1.60E+10	1.80E+09	0.11	0.41%	0.74%	98.85%	3.35E+07	0.00	38.46%	65.55%	0.00%	0.00001	0.00000
CMG 0 total		0.00	1.93E+11	2.16E+10	0.11	0.43%	0.73%	98.84%	4.02E+08	0.00	39.16%	65.03%	0.00%	0.00001	0.00000

ロード・ストア数、L1Dミス率、L2Dミス率、DTLBミス率
L2ミス内訳（dmミス率、hwpfミス率、swpfミス率）など

CPU性能解析レポート 表示イメージ (表3)

命令ミックス

Instruction		Load-store instruction																											
		Load instruction									Store instruction							Prefetch instruction				Floating-point instruction			Floating-point move and conversion instruction						
		SIMD							Non-SIMD		SIMD					Non-SIMD															
Process	Thread	Single vector contiguous load instruction	Multiple vector contiguous structure load instruction	Non-contiguous gather load instruction	Broadcast load instruction	Floating-point register fill instruction	Predicate register fill instruction	First-fault load instruction	Non-SIMD load instruction	Single vector contiguous store instruction	Multiple vector contiguous structure store instruction	Non-contiguous scatter store instruction	Floating-point register spill instruction	Predicate register spill instruction	Non-SIMD store instruction	Contiguous prefetch instruction	Gathering prefetch instruction	Scalar prefetch instruction	DCZVA instruction	Floating-point instruction except FMA and reciprocal	FMA instruction	Floating-point reciprocal instruction	Floating-point conversion instruction	Floating-point move instruction	Integer instruction	Branch instruction	Predicate instruction	Crypto-graphic instruction	Other instruction	Total
0	0	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	1.59E+06	5.84E+07	0.00E+00	0.00E+00	2.99E+03	1.62E+03		0.00E+00	0.00E+00	1.81E+09	8.24E+02	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.22E+03	0.00E+00	8.90E+08	1.24E+04	0.00E+00	8.03E+09	6.22E+10
0	1	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.99E+05	5.84E+07	0.00E+00	0.00E+00	2.99E+03	1.62E+03		0.00E+00	0.00E+00	1.81E+09	8.24E+02	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.21E+03	0.00E+00	8.89E+08	1.24E+04	0.00E+00	8.02E+09	6.22E+10
0	2	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.87E+05	5.84E+07	0.00E+00	0.00E+00																		
0	3	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.94E+05	5.84E+07	0.00E+00	0.00E+00																		
0	4	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.82E+05	5.84E+07	0.00E+00	0.00E+00																		
0	5	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.97E+05	5.84E+07	0.00E+00	0.00E+00																		
0	6	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.92E+05	5.84E+07	0.00E+00	0.00E+00																		
0	7	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	7.00E+05	5.84E+07	0.00E+00	0.00E+00																		
0	8	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.91E+05	5.84E+07	0.00E+00	0.00E+00																		
0	9	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	7.19E+05	5.84E+07	0.00E+00	0.00E+00	2.99E+03	1.62E+03	2.59E+05	0.00E+00	0.00E+00	1.81E+09	8.24E+02	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.21E+03	0.00E+00	8.89E+08	1.24E+04	0.00E+00	8.02E+09	6.22E+10
0	10	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.90E+05	5.84E+07	0.00E+00	0.00E+00	2.99E+03	1.62E+03	2.44E+05	0.00E+00	0.00E+00	1.81E+09	8.24E+02	1.56E+06	3.54E+10	0.00E+00	9.00E+01	4.21E+03	0.00E+00	8.89E+08	1.24E+04	0.00E+00	8.02E+09	6.22E+10
0	11	7.14E+09	0.00E+00	0.00E+00	8.85E+09	3.28E+03	1.86E+03	0.00E+00	6.95E+05	5.82E+07	0.00E+00	0.00E+00	3.04E+03	1.62E+03	2.44E+05	0.00E+00	0.00E+00	1.81E+09	8.24E+02	1.33E+06	3.54E+10	0.00E+00	9.00E+01	4.21E+03	0.00E+00	8.89E+08	1.24E+04	0.00E+00	8.02E+09	6.22E+10
CMG 0 total		8.56E+10	0.00E+00	0.00E+00	1.06E+11	3.93E+04	2.24E+04	0.00E+00	9.24E+06	7.00E+08	0.00E+00	0.00E+00	3.60E+04	1.94E+04	3.35E+06	0.00E+00	0.00E+00	2.17E+10	9.89E+03	1.85E+07	4.25E+11	0.00E+00	1.08E+03	5.05E+04	0.00E+00	1.07E+10	1.49E+05	0.00E+00	9.63E+10	7.46E+11

ロード・ストア命令、プリフェッチ命令、浮動小数点演算命令、整数演算命令、分岐命令 など

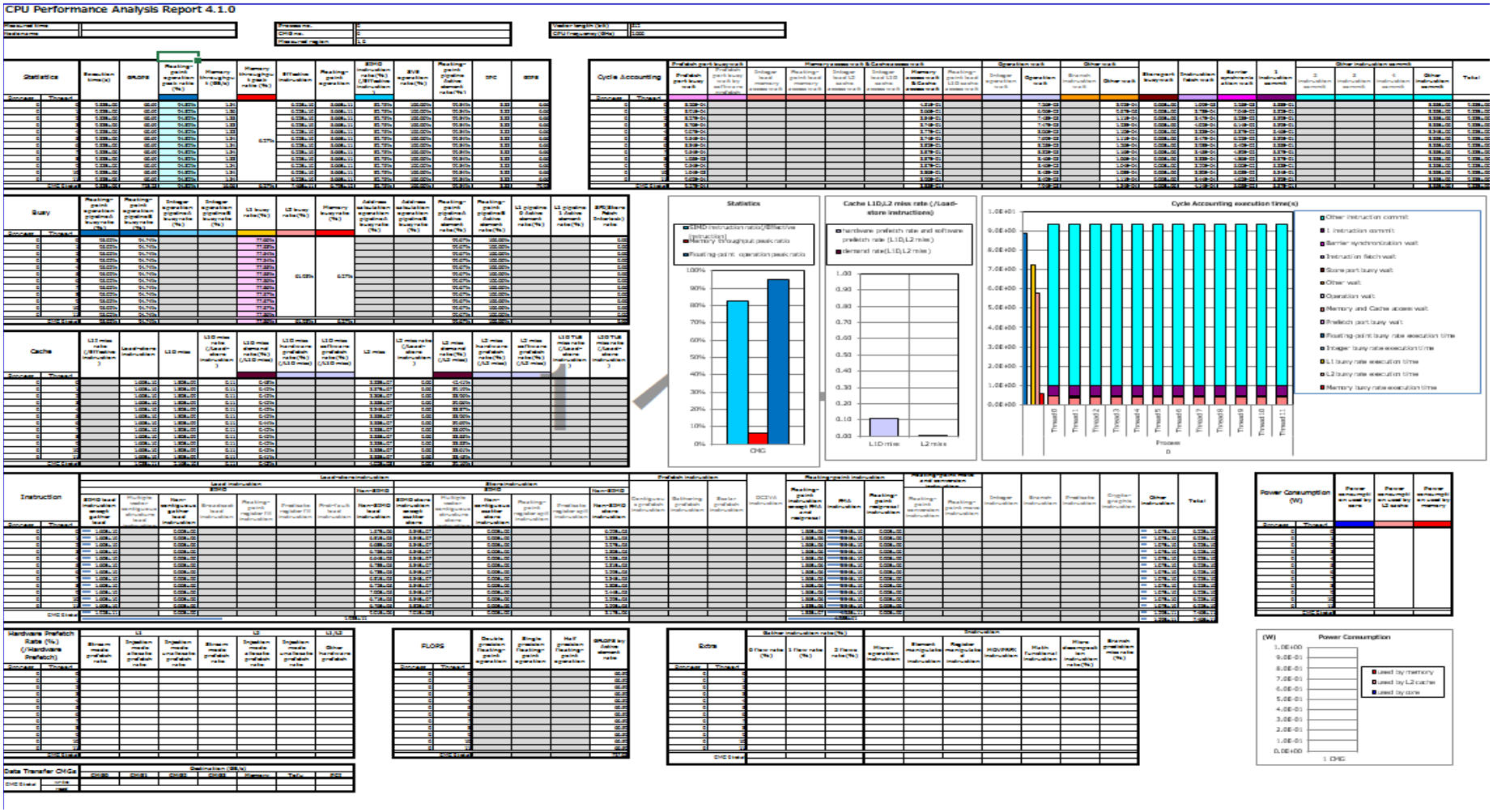
性能情報

FLOPS		Double precision floating-point operation	Single precision floating-point operation	Half precision floating-point operation	GFLOPS by Active element rate
Process	Thread				
0	0		0.00.E+00	0.00.E+00	60.59
0					
0					
0					
0	4	5.66.E+11	0.00.E+00	0.00.E+00	60.59
0	5	5.66.E+11	0.00.E+00	0.00.E+00	60.59
0	6	5.66.E+11	0.00.E+00	0.00.E+00	60.59
0	7	5.66.E+11	0.00.E+00	0.00.E+00	60.59
0	8	5.66.E+11	0.00.E+00	0.00.E+00	60.59
0	9	5.66.E+11	0.00.E+00	0.00.E+00	60.59
0	10	5.66.E+11	0.00.E+00	0.00.E+00	60.59
0	11	5.66.E+11	0.00.E+00	0.00.E+00	60.59
CMG 0 total		6.79.E+12	0.00.E+00	0.00.E+00	727.03

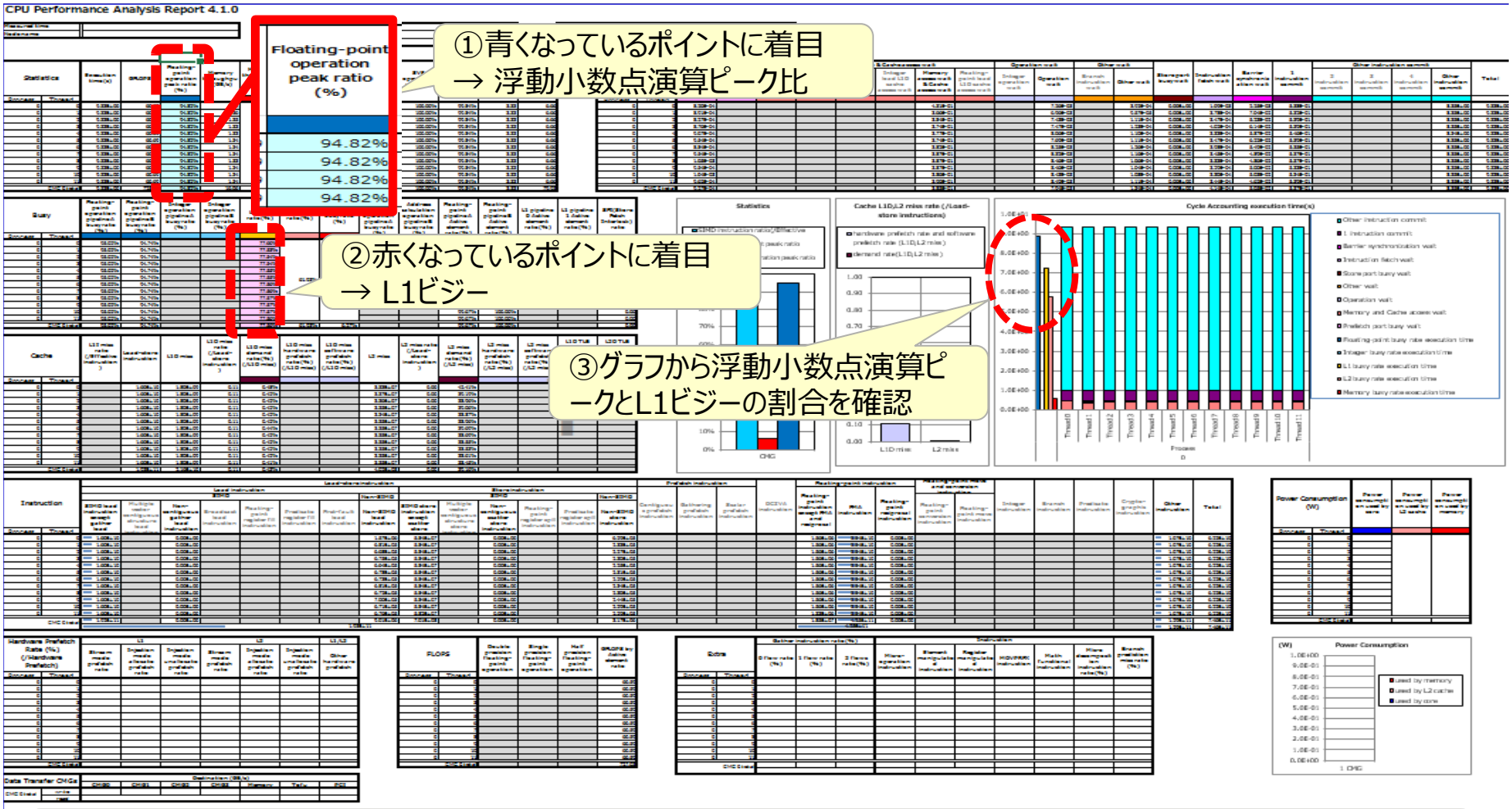
型別浮動小数点演算率(半精度・単精度・倍精度)、Predicate マスク情報を使ったGFLOPS値

- **型別浮動小数点演算率**
  測定区間で行われた浮動小数点演算のうち、各型(半精度、単精度、倍精度)を%で表示
- **Predicate マスク情報を使ったGFLOPS値**
  Predicateマスク情報を使用して算出したGFOPS値
  (Predicateマスク情報が演算のみの数値ではないため参考値)

表示例 (DGEMM：簡易情報(5回計測))

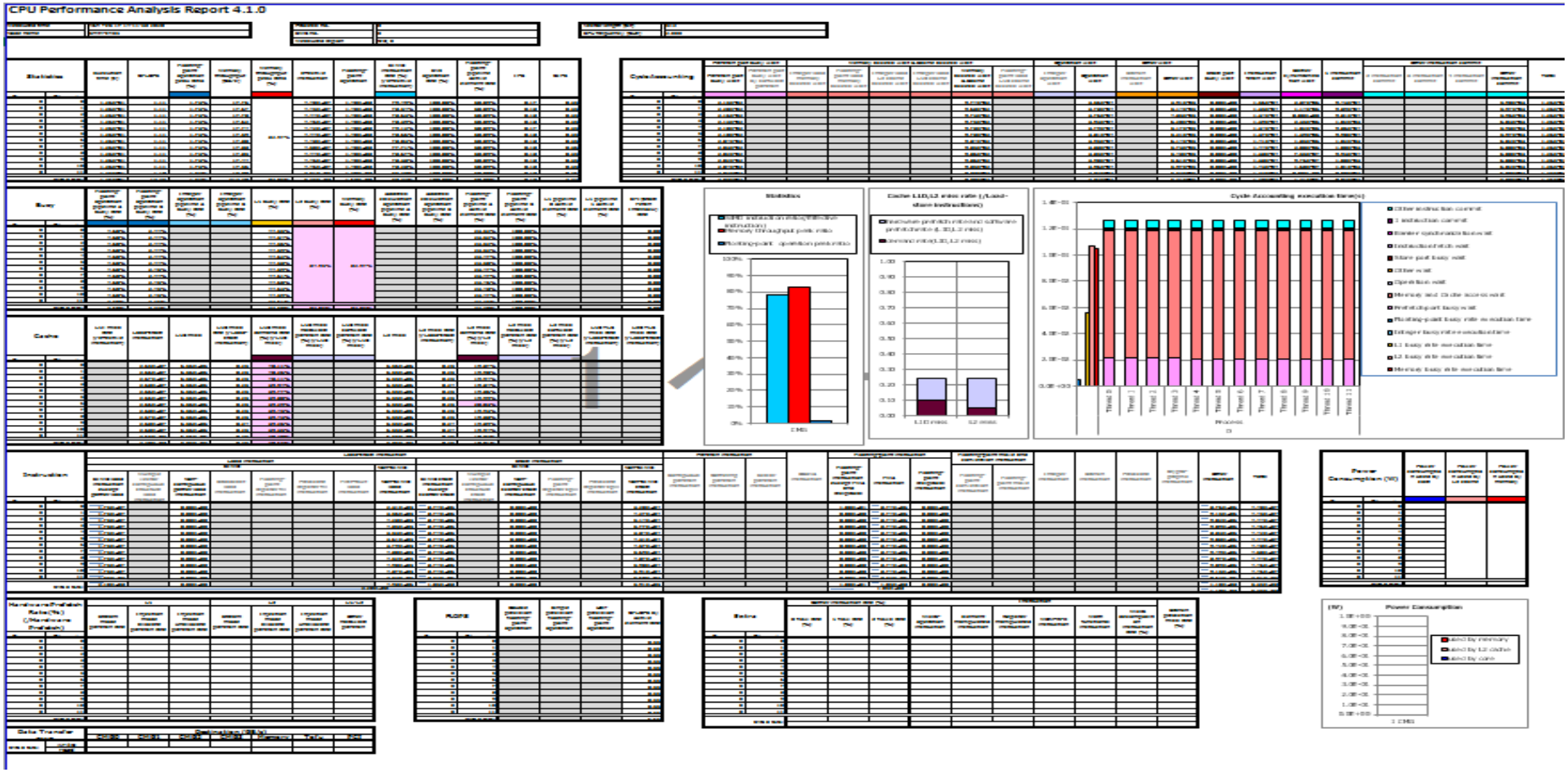


分析例（DGEMM）

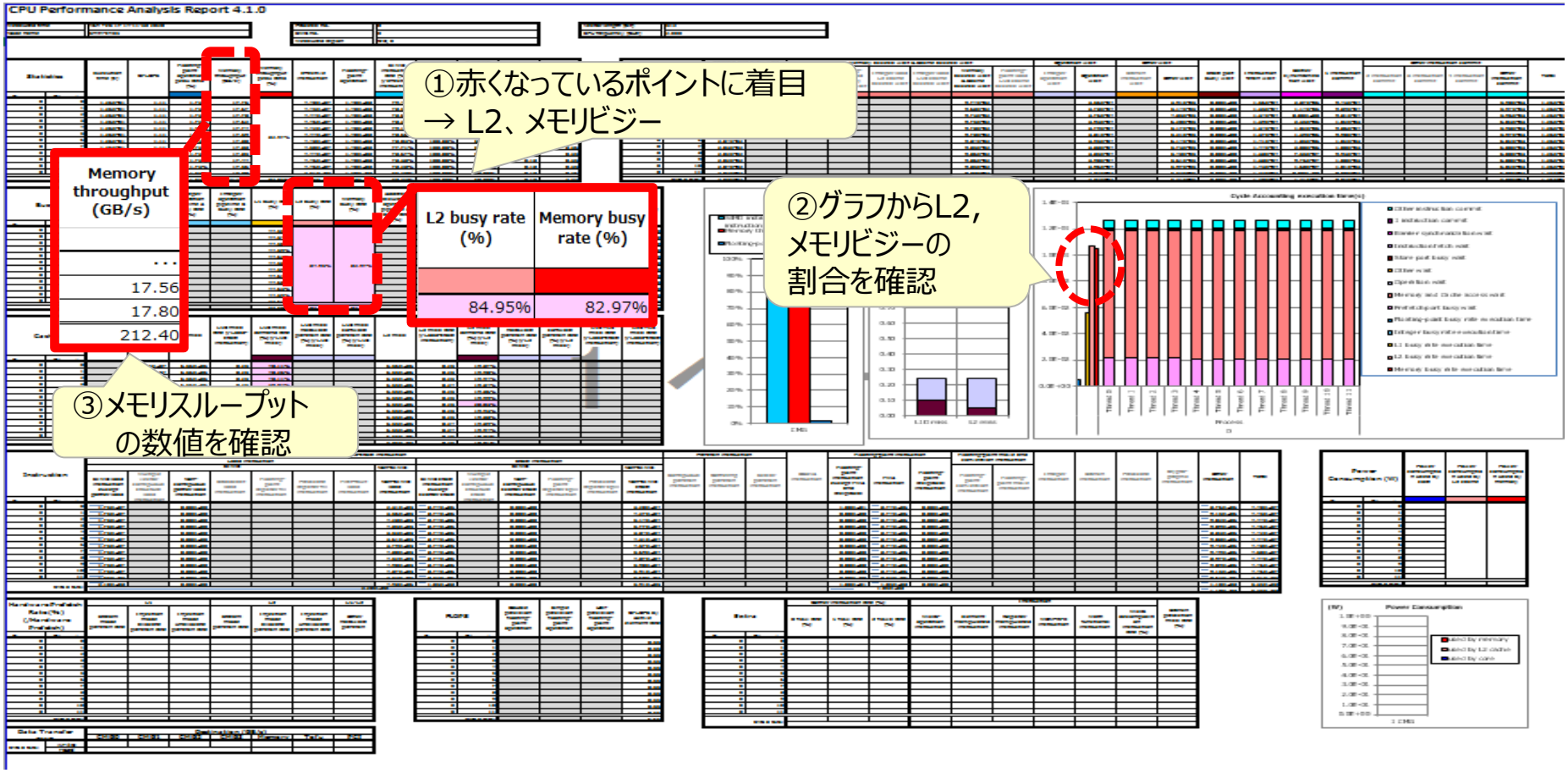


浮動小数点ピーク比(①)と、L1ビジー率(②)が高いが、グラフ(③)から浮動小数点ピーク比のほうが高いことが確認できる。数値も94.82%となっており、高い演算性能を出すことができていることがわかる。

表示例 (STREAM：簡易情報(5回計測))



分析例 (STREAM)



ビジー情報(①)やグラフ(②)からL2ビジー率、メモリビジー率が高いことが確認できる
数値(③)を確認すると207.69GB/sとなっており、これは1CMGあたりのメモリバンド幅(256GB/s)の80%以上
の性能が出ていることがわかる。

FLOPSについての注意事項

FUJITSU

■ 演算数のカウントについて

CPU性能解析レポートでは、演算数が多くカウントされるケースが存在し、
そのため本来のFLOPS値よりも高いFLOPS値になる可能性がある。
例えば、マスク付きSIMD命令が使用された場合、一部のマスク(Predicate)が偽でも、すべて真として演算数がカウントされ、多くなることがある。
上記を含め演算数が多くカウントされるケースについて、以下にまとめた。

項目	概要	京・FX100との違い	回避方法
浮動小数点除算・SQRT関数	コンパイラにより複数演算の命令列に置き換えられるため、演算数はコード上から見える数に比べ多くカウントされる	同じ	-Knofp_relaxed を指定
数学関数・数値関数			なし
リダクション	リダクション演算が含まれるループにて自動ループスライスが動作した場合、演算数はコード上から見える数に比べ多くカウントされる	同じ	-Knoreduction を指定
IF構文を含むループのSIMD化	IF構文を含むループをマスク付きSIMD命令を利用し最適化した場合、一部のマスク(Predicate)が偽でも、すべて真として演算数がカウントされる	-Ksimd=2と同等	-Ksimd=1 を指定
ループの冗長実行によるSIMD化	ループ繰返し数がSIMD長で割り切れないケースにてマスク付きSIMD命令を利用し最適化した場合、一部のマスク(Predicate)が偽でも、すべて真として演算数がカウントされる	新規対応	以下の OCL を指定 SIMD_NOREDUNDANT_VL

ラージページ

- ラージページについて
- ラージページ仕様
- ラージページ設定用環境変数
- メモリ使用量に関する注意事項(副作用)について
- ラージページ評価
  - Stream Triad (1スレッド実行)
- ARENA_FREE の性能
- PAGING_POLICY の性能
- ARENA_LOCK_TYPE の性能

ラージページについて

FUJITSU

ラージページとは

- 大規模なデータを扱うアプリケーションに対して、通常のページ(ノーマルページ)より大きなページサイズのメモリ(ラージページ)を割り当てることで、
  - CPUのアドレス変換処理によるオーバーヘッドを低減します。
  - メモリアクセス性能を向上させます。

- A64FXシステム環境での、ノーマルページのサイズは64KiBであり、ラージページとして利用可能なサイズは、2MiBです。
  - 環境変数設定により以下の動作設定が可能
    - ✓ ラージページ割り当て動作の有効化/無効化
    - ✓ スタック領域のラージページ割り当て動作の有効化/無効化
    - ✓ 各メモリ領域のページング方式(ページの割り当て契機)の選択
  - 32MiB/1GiB/16GiBなどの様々なページサイズはMcKernelで実現可能。

ラージページ仕様

■ ラージページ仕様

メモリ領域	MP10/FX10/FX100	A64FX			
	ページサイズ	ページサイズ			ページング (_付はdefault)
		ノーマルページ	ラージページ base	ラージページ base+stack (default)	
テキスト (.text)	8KiB	64KiB	64KiB	64KiB	–
静的データ (.data)	4MiB (default), 8KiB, 32MiB, 256MiB	64KiB	2MiB	2MiB	常にprepage
静的データ (.bss)		64KiB	2MiB	2MiB	demand | prepage
スタック (*1)		64KiB	64KiB	2MiB	demand | prepage
動的メモリ (*2)		64KiB	2MiB	2MiB	demand | prepage
共有メモリ		64KiB	64KiB	64KiB	–

＊１ ：プロセススタック/メインスレッド用スタック/スレッドスタック領域が対象
＊２ ： プロセスヒープ/メインスレッド用ヒープ/スレッドヒープ/mmap領域が対象

ラージページ設定用環境変数 (1/2)

FUJITSU

■ 基本設定/ページング方式の設定

環境変数名	指定値 (_付はdefault)	説明
XOS_MMM_L_HPAGE_TYPE	hugetlbfs | none	ラージページライブラリによるラージページ割り当て動作の有効化/無効化を選択する設定です。 「hugetlbfs」の場合、HugeTLBfsによるラージページ化をします。 「none」の場合、ラージページライブラリによるラージページ化は行いません。
XOS_MMM_L_LPG_MODE	base+stack | base	スタック領域およびスレッドスタック領域のラージページ割り当て動作の有効化/無効化を選択する設定です。 「base+stack」の場合、静的データおよび動的メモリ確保領域だけではなく、スタック領域およびスレッドスタック領域もラージページ化します。 「base」の場合、静的データおよび動的メモリ確保領域のみラージページ化します。スタック領域およびスレッドスタック領域はラージページ化しません。
XOS_MMM_L_PAGING_POLICY	[demand | prepage]: [demand | prepage]: [demand | prepage]	各メモリ領域のページング方式(ページの割り当て契機)を選択する設定です。 「demand」はデマンドページング方式、「prepage」はプリページング方式を意味します。本変数はコロン(:)区切りで3つのメモリ領域のページング方式を指定します。 第1指定は、静的データの.bss領域です。(静的データの.data領域はページング方式指定の対象外で常にprepageとなります。) 第2指定は、スタック領域およびスレッドスタック領域です。 第3指定は、動的メモリ確保領域です。 指定値以外の値を指定した場合は、「prepage:demand:prepage」を指定したものとみなします。

ラージページ設定用環境変数 (2/2)

FUJITSU

■ チューニング用の設定 (ラージページライブラリ独自の環境変数)

環境変数名	指定値 (_付はdefault)	説明
XOS_MMM_L_ARENA_FREE	1 | 2	free(3)で解放されるヒープ領域の扱いに関する設定です。 「1」を指定した場合は、解放可能なメモリを即時に解放します。「2」を指定した場合は、メモリを一切解放せず、全メモリをプールして再利用します。
XOS_MMM_L_ARENA_LOCK_TYPE	0 | 1	メモリ割り当てポリシーに関する設定です。 「0」はメモリ獲得性能優先、「1」はメモリ使用効率優先を意味します。
XOS_MMM_L_MAX_ARENA_NUM	1 以上INT_MAX 以下の整数値 [10進数]	生成可能なアリーナ(プロセスヒープとスレッドヒープ領域の総和)の数を設定します。 XOS_MMM_L_ARENA_LOCK_TYPE=0のときに有効になります。
XOS_MMM_L_HEAP_SIZE_MB	MALLOC_MMAP_THRESHOLD_の2倍 以上 ULONG_MAX以下の整数値<MiB単位> [10進数]	スレッドヒープ領域を使用する場合に、スレッドヒープ領域の生成時および拡張時に獲得するメモリサイズを設定します。
XOS_MMM_L_COLORING	0 | 1	キャッシュカラーリングの有無を設定します。プロセッサのL1キャッシュのコンフリクトを軽減します。「0」の場合、キャッシュカラーリングを行いません。 「1」の場合、MALLOC_MMAP_THRESHOLD_(デフォルトは128MiB)以上のサイズで行われるmmap(2)によるメモリ獲得時にはキャッシュカラーリングをします。
XOS_MMM_L_FORCE_MMAP_THRESHOLD	0 | 1	MALLOC_MMAP_THRESHOLD_(デフォルトは128MiB)以上のサイズのメモリ獲得時にmmap(2)を優先するかどうかの設定です。 「0」の場合、mmap(2) は優先しません。まずヒープ領域の空きを検索し、空きがあればヒープ領域の空きメモリを返します。ヒープ領域の空きが見つからないときにのみmmap(2) でメモリを獲得します。「1」の場合、mmap(2) を優先します。ヒープ領域の空きは検索せず、(例え空きがあっても)mmap(2) でメモリを獲得します。

■ glibcの環境変数(MALLOC_MMAP_THRESHOLD_等)についてはユーザ向けガイドを参照

メモリ使用量に関する注意事項(副作用)について

FUJITSU

- 静的データ(.data)領域

  常に①、②の副作用が発生します。

- 静的データ(.bss)領域

  以下の条件を満たす場合、①、②の副作用が発生します。

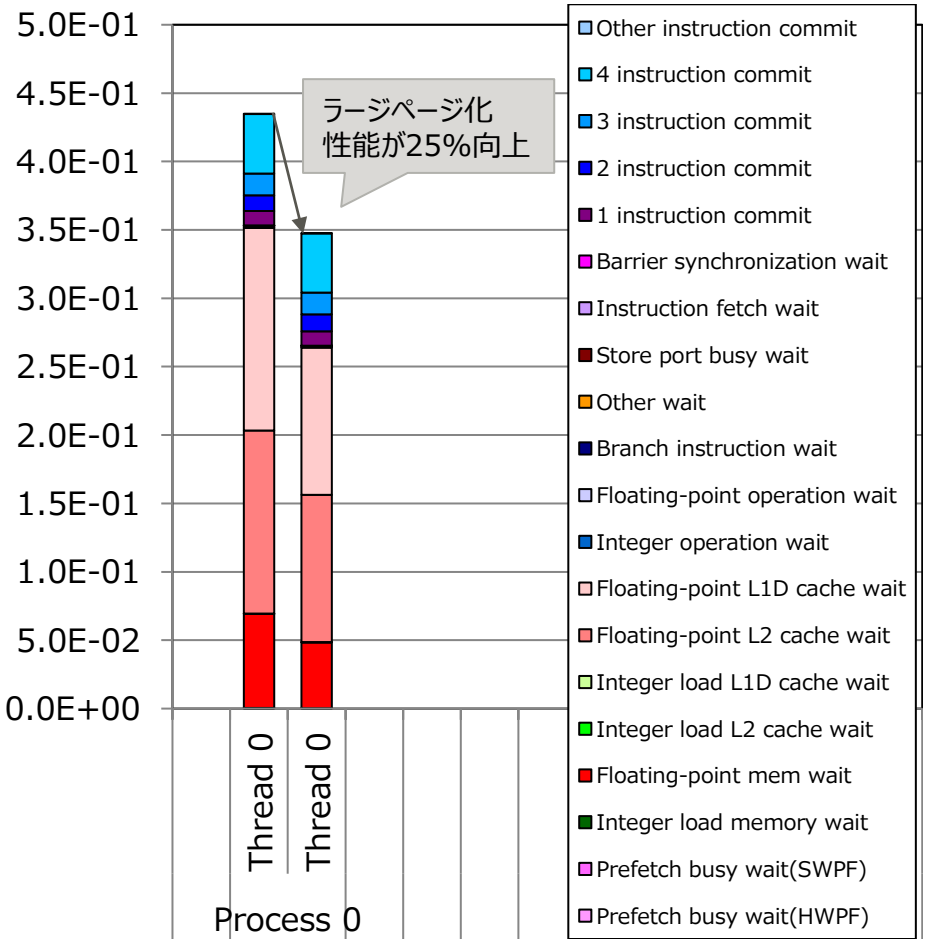
  a. .dynsym セクション(dynamic section)に存在するシンボルである
  b. シンボルが(メインプログラムの) bss 領域(bss_start, bss_end)の範囲内のアドレスにある
  c. シンボルがグローバル(STB_GLOBAL)またはウィーク(STB_WEAK)である
  d. シンボルのタイプが変数(STT_OBJECT)である
  e. シンボルのサイズが 0 より大きい

- 副作用

  ① 2倍または3倍のメモリ領域を使用することがあります。
  ② XOS_MMM_L_PAGING_POLICYで静的データ(.bss)領域のページング方式をデマンドページング方式("demand")に設定していても、プリーページング方式("prepage")で動作します。

ラージページ評価１：Stream Triad (1スレッド実行)

Stream Triad (1スレッド実行)



推定方法	A64FX	A64FX
カーネル化、縮小	n=83880960	
ソースコード版数	**ノーマルページ**	**ラージページ**
浮動小数点数精度	倍精度実数	
SIMD幅	8	
集計スレッド番号	0	
実行時間[s]	4.35E-01	3.47E-01
GFLOPS/プロセス	3.86	4.83
メモリスループット [GB/s/プロセス]	61.76	77.29
実行命令数/スレッド[10^9]	5.11E+08	5.11E+08
ロード・ストア命令数/スレッド[10^9]	3.15E+08	3.15E+08
分岐命令数/スレッド[10^9]	1.31E+07	1.31E+07
その他命令数/スレッド[10^9]	7.87E+07	7.87E+07
SIMD命令率/スレッド	82.03%	82.04%
L1Dミス率/スレッド	25.26%	25.00%
L2ミス率/スレッド	25.00%	25.00%
L1D TLBミス率/スレッド	0.09786%	0.00324%
L2D TLBミス率/スレッド	0.09776%	0.00033%
L1ビジー率/プロセス	73.95%	84.13%
L2ビジー率/プロセス	16.77%	20.75%
メモリビジー率/プロセス	24.13%	30.19%

ラージページ化により、L2D TLB miss率が低減し性能が25%向上

ラージページ評価２：ARENA_FREE の性能(1/2)

FUJITSU

■ 測定条件

条件	パターン
検証コード	右記コード (マニュアル抜粋)
測定スレッド数	1スレッド
コンパイラ	A64FX向けコンパイラ
翻訳オプション	-Kfast
アクセス範囲	N=1024 MALLOC_CNT=1024
評価条件	以下の2条件で実行結果を比較 - XOS_MMM_L_ARENA_FREE=1 (デフォルト・メモリを解放する) - XOS_MMM_L_ARENA_FREE=2 (メモリを解放せず再利用する)

測定コード

```
while(loop <2){
  printf("malloc start.¥n");
  clock_gettime(CLOCK_REALTIME, &time1);
  for(i=0;i<MALLOC_CNT;i++){
    c[i]=(double *)malloc(sizeof(double)*N*N);
    if (c[i] == NULL) {
      fprintf(stderr, "malloc error: cnt=%d, errno=%d¥n", i, errno);
      exit(1);
    }
  }
  clock_gettime(CLOCK_REALTIME, &time2);
  printf("malloc end.¥n");
  sec = (time2.tv_sec - time1.tv_sec);
  nsec= (time2.tv_nsec-time1.tv_nsec);
  if(nsec<0){
    sec--;
    nsec += 1000000000L;
  }
  printf("MALLOC TIME:%d:%010d¥n", sec, nsec);
  sleep(10);

  printf("free start.¥n");
  clock_gettime(CLOCK_REALTIME, &time1);
  for(i=0;i<MALLOC_CNT;i++){
    free(c[i]);
  }
  clock_gettime(CLOCK_REALTIME, &time2);
  printf("free end.¥n");
  sec = (time2.tv_sec - time1.tv_sec);
  nsec= (time2.tv_nsec-time1.tv_nsec);
  if(nsec<0){
    sec--;
    nsec += 1000000000L;
  }
  printf("FREE TIME:%d:%010d¥n", sec, nsec);
  loop++;
}
```

mallocを1024回繰り返す

mallocの経過時間を印字

freeを1024回繰り返す

freeの経過時間を印字

malloc,freeを行うループを2回

ラージページ評価２：ARENA_FREE の性能(2/2)

FUJITSU

- XOS_MMM_L_ARENA_FREE=1
  (デフォルト・メモリの開放を行う)

```
malloc start.
malloc end.
MALLOC TIME:0:0501782690
free start.
free end.
FREE TIME:0:0260215760
malloc start.
malloc end.
MALLOC TIME:0:0500856220
free start.
free end.
FREE TIME:0:0260336510
```

- XOS_MMM_L_ARENA_FREE=2
  (メモリを解放せず再利用する)

```
malloc start.
malloc end.
MALLOC TIME:0:0510782430
free start.
free end.
FREE TIME:0:0000496350
malloc start.
malloc end.
MALLOC TIME:0:0000308220
free start.
free end.
FREE TIME:0:0000252020
```

524倍の高速化

1625倍の高速化

1033倍の高速化

**XOS_MMM_L_ARENA_FREE=2
を指定することで、freeや2回目以降の
mallocが高速で実行される。**

同じサイズの malloc 、free を繰り返すプログラムでは
XOS_MMM_L_ARENA_FREE が有効。

ラージページ評価３：PAGING_POLICY の性能

- XOS_MMM_L_PAGING_POLICY=prepage:demand:prepage
  - スレッド並列で複数CMGにて走行する時は、以下のメモリアロケーションとなる。
    - プリページングでは、ロードモジュール起動時に、CMG0からデータが載る。
    - デマンドページングでは、初回アクセス時に実行CMGにデータが載る。
  - **複数CMGをまたぐときはデマンドページングを推奨します。**

```
14                    Subroutine sub(n,iter,x1,x2,y1)
15                      real(8) :: x1(n), x2(n), y1(n),c0
16                      integer n,i,k
17                      c0=2.0
 :                        ……
20    1                  do k=1,iter
21    1                 !$omp parallel do
            <<< Loop-information Start >>>
            <<<  [OPTIMIZATION]
            <<<    SIMD(VL: 8)
            <<<    SOFTWARE PIPELINING(IPC: 2.45, ITR: 128, MVE: 2, POL: S)
            <<<    PREFETCH(SOFT) : 10
            <<<     SEQUENTIAL : 10
            <<<      x2: 4, x1: 4, y1: 2
            <<<    ZFILL          :
            <<<      y1
            <<< Loop-information  End >>>
22    2  p   v            do i=1,n
23    2  p   v              y1(i) = x1(i) + c0 * x2(i)
24    2  p   v            end do
25    1                  enddo
 :                        ……
30                      parameter(N=45000000,ITER=100)
31                      real*8 x1(N),x2(N),y1(N)
32                      call init(N,ITER,x1,x2,y1)
33                      call sub(N,ITER,x1,x2,y1)
```

ストリーム(データサイズは約1GB)

プリページング(prepage指定)ではCMG0からデータが載るため、48スレッドのストリームの性能がでません。
デマンドページング(demand指定)に変更することで実行CMGにデータが載り、性能が大幅に向上します。

ページングポリシー	Memory throughput (GB/s)
prepage指定 (default)	93GB/s
demand指定	804GB/s

実行時のページングポリシー:
XOS_MMM_L_PAGING_POLICY=
demand:demand:demand

ラージページ評価４：ARENA_LOCK_TYPE の性能

- XOS_MMM_L_ARENA_LOCK_TYPE
  - メモリ割り当てポリシーに関する設定です。
  - 「0」はメモリ獲得性能優先。パラレルリージョン内でmallocしている場合に推奨します。(メモリの獲得/開放がスレッド毎に独立したメモリプールから行われるようになり、デフォルト設定よりも排他制御のコストが減って性能改善する場合があります)
  - 「1」はメモリ使用効率優先 (デフォルト)。メモリ使用量が多い場合に推奨します。

```
 1                subroutine sub(n,m,iter,x1,x2,y2)
 2                  integer(8) :: pZ1(iter)
 3                  real(8) :: x1(n), x2(n), y2(n,m),c0
 4                  c0=2.0
 5
 6                !$omp parallel do shared(n,m,iter,x1,x2,c0,y2)
                                    private(pZ1,i,j,k) default(none)
 :                    ......
 7    1   p           do k=1,m
 :                    ......
 8    2   p   s         do j=1,iter
 9    2   p   m           pZ1(j) = malloc(8 * n)
10    2   p   v         end do
11    1
 :                    ......
12    2   p   2v          do i=1,n
13    2   p   2v            y2(i,k) = x1(i) + c0 * x2(i)
14    2   p   2v          end do
15    1
16    2   p   s         do j=1,iter
17    2   p   s           call free( pZ1(j))
18    2   p   s         end do
19    1   p           end do
20                   end subroutine sub
21
22                   program main
23                     parameter(N=1048512,ITER=80)
24                     real*8 x1(N),x2(N),y2(N,12)
25                     call sub(N,12,ITER,x1,x2,y2)
```

ラージページのロックタイプ(環境変数XOS_MMM_L_ARENA_LOCK_TYPE)をメモリ使用率優先(1)からメモリ獲得優先(0)にすることで、mallocの性能が向上

ロックタイプ	実行時間(秒)
使用効率優先(=1) [default]	0.56
メモリ獲得性能優先(=0)	0.35

実行時のロックタイプ:
XOS_MMM_L_ARENA_LOCK_TYPE=0

FUJITSU

セクタキャッシュ

- セクタキャッシュとは
- セクタキャッシュの使用方法
- セクタキャッシュの事例
- DGEMMでのセクタキャッシュの効果

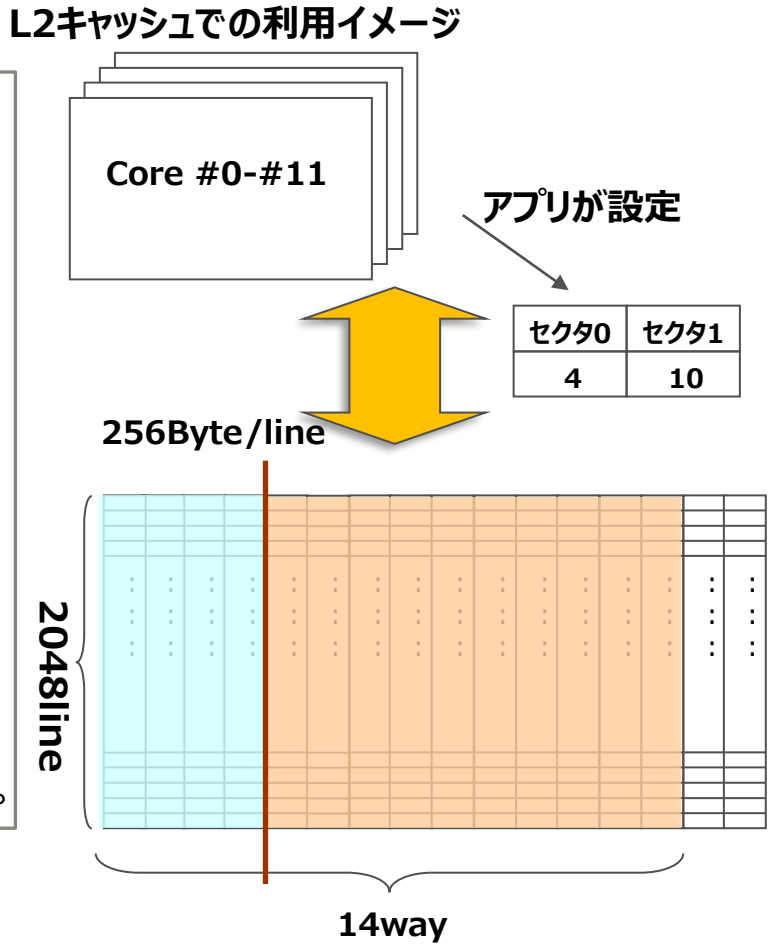
セクタキャッシュとは

セクタキャッシュ

セクタキャッシュとは、再利用性のあるデータが、再利用性の無いデータによってキャッシュから追い出されることを防ぐことができるキャッシュ機構です。アプリが再利用性のあるデータと再利用性の無いデータをセクタ毎に分けて配置することができます。（再利用する配列はセクタ1を使用し、その他はセクタ0を使用）

セクタキャッシュの詳細

- L1Dキャッシュ・L2キャッシュいずれにも複数個のセクタを設定できます。セクタの最大数はL1Dが4,L2は2です。
- 各セクタの容量はWAY数で指定します。
- 容量は目標値として働きます。
  ハードは、ラインリプレース時に各セクタが指定された容量に近づくように制御します。
  → 容量オーバーしていても強制的に無効化しない
- セクタ内の追い出しはLRU（最近最も使われていないデータを最初に捨てる）アルゴリズムで制御します。
- セクタ０、１の用途はアプリケーションで決めることが可能です。
  ただし、命令列はセクタ０に格納されます。
- 2次キャッシュはアシスタントコアが常時2wayを使用します。



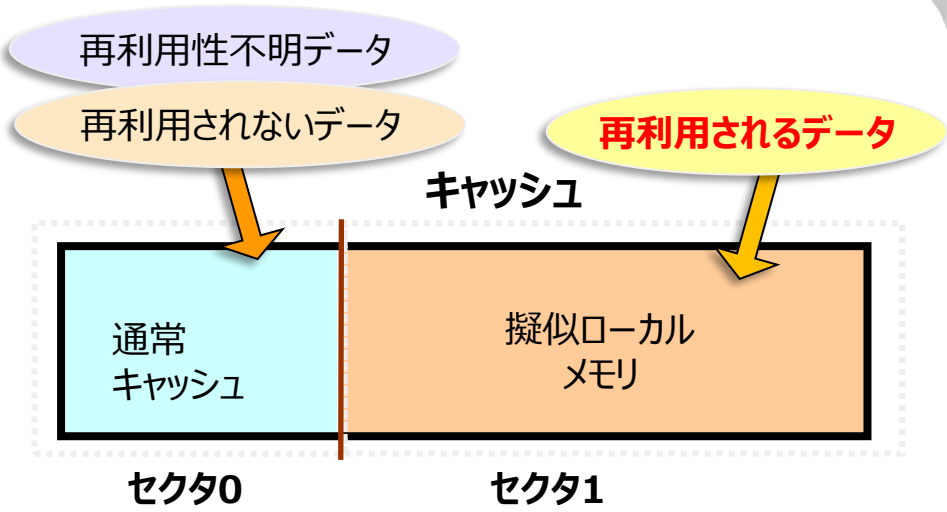
セクタキャッシュの使用方法（1/2）

FUJITSU

■ セクターキャッシュ：疑似ローカルメモリ

ソフトウェアが、データの再利用性に応じて
セクタを使い分けることが可能

- 再利用するデータ → セクタ1を使用
- その他のデータ → セクタ0を使用
- セクタ1上のデータは、他のデータによって
  追い出されることがない
- 指示行でセクタ1に載せる配列を指定できる



セクタキャッシュ指定のコンパイラ指示行の使用例

```
!OCL SCACHE_ISOLATE_WAY(L2=10)
!OCL SCACHE_ISOLATE_ASSIGN(a)
do j=1,m
   do i=1,n
      a(i) = a(i) + b(i,j)*c(i,j)
   enddo
enddo
!OCL END_SCACHE_ISOLATE_ASSIGN
!OCL END_SCACHE_ISOLATE_WAY
```

旧仕様の指定方法（京,FX100）

```
!OCL CACHE_SECTOR_SIZE(4,10)
!OCL CACHE_SUBSECTOR_ASSIGN(a)
do j=1,m
   do i=1,n
      a(i) = a(i) + b(i,j)*c(i,j)
   enddo
enddo
!OCL END_CACHE_SUBSECTOR
!OCL END_CACHE_SECTOR_SIZE
```

<狙い>
ループ中で配列 b と配列 c のアクセスによって
再利用性のある配列 a がキャッシュから追い出されないようにする

セクタキャッシュの使用方法（2/2）

FUJITSU

セクタキャッシュを使用する場合は、以下の最適化制御行を指定します。

最適化指示子	意味	指定可能な最適化制御行			
		プログラム単位	DOループ単位	文単位	配列代入文単位
SCACHE_ISOLATE_WAY(L2=n1[,L1=n2]) END_SCACHE_ISOLATE_WAY	1次キャッシュと2次キャッシュのセクタ1の最大way数を指示します。	○	×	○	×
SCACHE_ISOLATE_ASSIGN(array1[,array2]…) END_SCACHE_ISOLATE_ASSIGN	キャッシュのセクタ1に載せる配列を指示します。	○	×	○	×

■ 注意事項

- 2次キャッシュはアシスタントコアが常時2wayを使用します。そのため、n1およびn2に指定できる範囲は以下のようになります。
  0 ≦ *n1* ≦ “2次キャッシュの最大way数 - 2”
  0 ≦ *n2* ≦ “1次キャッシュの最大way数”

- アシスタントコアを含むCMGでは、L2キャッシュの一部(2way=1MiB分)をアシスタントコア用に使用します。そのため、アシスタントコアを含むCMGでは、2次キャッシュの最大way数は14, サイズは7MiBとなります。

A64FX諸元	
CMG数	4
L1Iキャッシュ・サイズ	64KiB / 4way
L1Dキャッシュ・サイズ	64KiB / 4way
L2キャッシュ・サイズ	32MiB / 16way (8MiB / CMG)

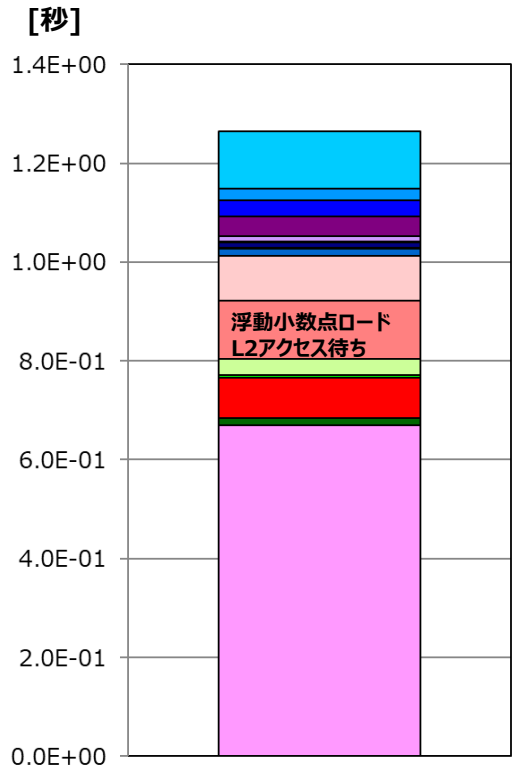
セクタキャッシュ：事例1（改善前）

配列b のデータはキャッシュから追い出されてしまうため、再利用できません。
そのため浮動小数点ロードL2アクセス待ちが多くなっています。

改善前ソース

```
66                      parameter(n=8*1024*1024, m=9*512*1024/8)
67                      real*8    a(n), b(m), s
68                      integer*8 c(n)
69                      real*8    dummy1(140),dummy2(140)
70                      common /data/a,dummy1,c,dummy2,b
71
                  <<< Loop-information Start >>>
                  <<<  [PARALLELIZATION]
                  <<<    Standard iteration count: 843
                  <<<  [OPTIMIZATION]
                  <<<    SIMD(VL: 8)
                  <<<    SOFTWARE PIPELINING(IPC: 2.66, ITR: 176, MVE: 4, POL: S)
                  <<<    PREFETCH(HARD) Expected by compiler :
                  <<<      c, a
                  <<< Loop-information  End >>>
72    1  pp   2v           do i=1,n
73    1   p   2v             a(i) = a(i) + s * b(c(i))
74    1   p   2v           enddo
```

配列サイズ
a: 64MiB
b: 4.5MiB
c: 64MiB



改善前

Cache	L1I miss rate (/Effective instruction)	Load-store instruction	L1D miss	L1D miss rate (/Load-store instruction)	L1D miss demand rate (%) (/L1D miss)	L1D miss hardware prefetch rate (%) (/L1D miss)	L1D miss software prefetch rate (%) (/L1D miss)	L2 miss	L2 miss rate (/Load-store instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)
改善前	0.00	4.76E+09	7.89E+08	0.17	0.89%	99.11%	0.00%	7.34E+08	0.15	0.77%	100.00%	0.00%

	Memory throughput (GB/s)
改善前	203.11

メモリスループットネックに陥っている

L2キャッシュミス率が高い

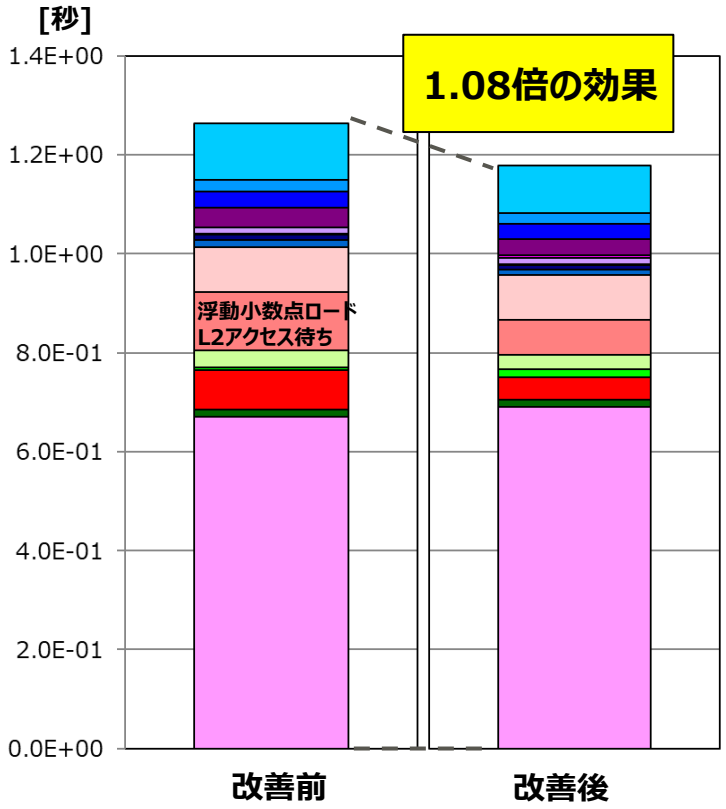
セクタキャッシュ：事例1（ソースチューニング）

FUJITSU

配列bをセクタ１に載せることでキャッシュ効率が向上し、浮動小数点ロードL2アクセス待ちが改善されました。

改善後ソース（最適化制御行チューニング）

```
58                    parameter(n=8*1024*1024, m=9*512*1024/8)
59                     real*8    a(n), b(m), s
60                     integer*8 c(n)
61                     real*8    dummy1(140),dummy2(140)
62                     common /data/a,dummy1,c,dummy2,b
63
64                !OCL SCACHE_ISOLATE_WAY(L2=10)
65                !OCL SCACHE_ISOLATE_ASSIGN(b)
                  <<< Loop-information Start >>>
                  <<<  [PARALLELIZATION]
                  <<<    Standard iteration count: 843
                  <<<  [OPTIMIZATION]
                  <<<    SIMD(VL: 8)
                  <<<    SOFTWARE PIPELINING(IPC: 2.66, ITR: 176, MVE: 4, POL: S)
                  <<<    PREFETCH(HARD) Expected by compiler :
                  <<<      c, a
                  <<< Loop-information  End >>>
66    1  pp   2v          do i=1,n
67    1   p   2v            a(i) = a(i) + s * b(c(i))
68    1   p   2v          enddo
69                !OCL END_SCACHE_ISOLATE_ASSIGN
70                !OCL END_SCACHE_ISOLATE_WAY
```



	L1I miss rate (/Effective instruction)	Load-store instruction	L1D miss	L1D miss rate (/Load-store instruction)	L1D miss demand rate (%) (/L1D miss)	L1D miss hardware prefetch rate (%) (/L1D miss)	L1D miss software prefetch rate (%) (/L1D miss)	L2 miss	L2 miss rate (/Load-store instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)
改善前	0.00	4.76E+09	7.89E+08	0.17	0.89%	99.11%	0.00%	7.34E+08	0.15	0.77%	100.00%	0.00%
改善後	0.00	5.19E+09	7.93E+08	0.15	1.19%	98.81%	0.01%	5.99E+08	0.12	1.93%	99.69%	0.00%

	Memory throughput (GB/s)
改善前	203.11
改善後	188.07

L2ミスが減少した

セクタキャッシュ：事例2（改善前）

FUJITSU

配列uのデータはキャッシュから追い出されてしまうため、再利用できません。
そのため、メモリ・キャッシュビジー待ちが多くなっています。

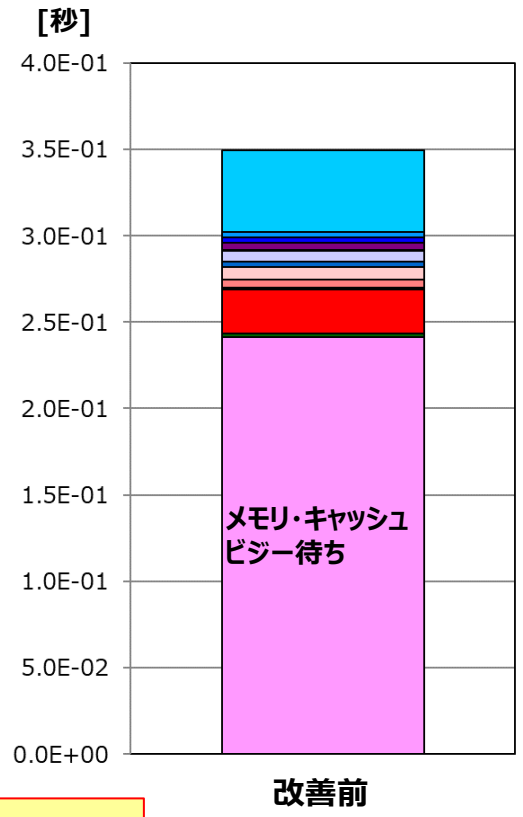
改善前ソース

```
167    1  s          do iter = 1, niter
                  <<< Loop-information Start >>>
                  <<<  [PARALLELIZATION]
                  <<<    Standard iteration count:
                  <<< Loop-information  End >>>
168    2  pp           do k=1,n3-2
                  <<< Loop-information Start >>>
                  <<<  [OPTIMIZATION]
                  <<<    PREFETCH(HARD) Expected by compiler :
                  <<<      u, rhs, unew
                  <<< Loop-information  End >>>
169    3  p             do j=1,n2-2
                  <<< Loop-information Start >>>
                  <<<  [OPTIMIZATION]
                  <<<    SIMD(VL: 8)
                  <<<    SOFTWARE PIPELINING(IPC: 3.71, ITR: 136, MVE: 9, POL: S)
                  <<<    PREFETCH(HARD) Expected by compiler :
                  <<<      u, rhs, unew
                  <<< Loop-information  End >>>
170    4  p   v          do i=1,n1-2
171    4  p   v            unew(i,j,k) = &
172    4                     ((u(i+1,j,k) + u(i-1,j,k)) * h1sqinv  &
173    4                     +(u(i,j+1,k) + u(i,j-1,k)) * h2sqinv &
174    4                     +(u(i,j,k+1) + u(i,j,k-1)) * h3sqinv &
175    4                     -rhs(i,j,k)) * hhhinv
176    4  p   v          end do
177    3  p             end do
178    2  p           end do
179    1            end do
```

n1=452
n2=52
n3=322

配列サイズ
unew: 60.5MB
u: 60.5MB
rhs: 60.5MB

配列uのi,j次元に再利用性があるため、配列uをオンキャッシュにしたい。



メモリスループットネックに陥っている

	Memory throughput (GB/s)
改善前	215.62

Cache	L2 miss	L2 miss rate (/Load-store instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)
改善前	2.46E+08	0.15	2.57%	98.19%	0.00%

セクタキャッシュ：事例2（ソースチューニング）

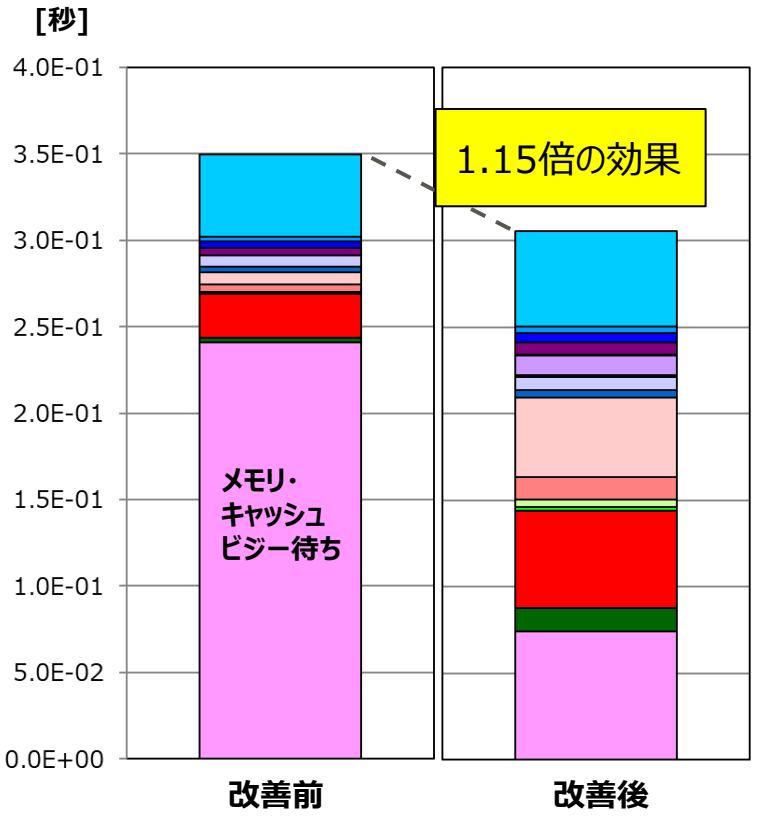
FUJITSU

配列*u*の*k*次元の一部をセクタ１に載せることで、キャッシュ効率が向上し、メモリ・キャッシュビジー待ちが改善されました。

改善後ソース

```
166                !OCL SCACHE_ISOLATE_WAY(L2=13)
167                !OCL SCACHE_ISOLATE_ASSIGN(u)
168    1   s         do iter = 1, niter
                  <<< Loop-information Start >>>
                  <<<  [PARALLELIZATION]
                  <<<    Standard iteration count: 2
                  <<< Loop-information  End >>>
169    2  pp          do k=1,n3-2
                  <<< Loop-information Start >>>
                  <<<  [OPTIMIZATION]
                  <<<    PREFETCH(HARD) Expected by compiler :
                  <<<      u, rhs, unew
                  <<< Loop-information  End >>>
170    3   p           do j=1,n2-2
                  <<< Loop-information Start >>>
                  <<<  [OPTIMIZATION]
                  <<<    SIMD(VL: 8)
                  <<<    SOFTWARE PIPELINING(IPC: 3.71, ITR: 136, MVE: 9, POL: S)
                  <<<    PREFETCH(HARD) Expected by compiler :
                  <<<      u, rhs, unew
                  <<< Loop-information  End >>>
171    4   p    v        do i=1,n1-2
172    4   p    v          unew(i,j,k) = &
173    4                     ((u(i+1,j,k) + u(i-1,j,k)) * h1sqinv  &
174    4                     +(u(i,j+1,k) + u(i,j-1,k)) * h2sqinv &
175    4                     +(u(i,j,k+1) + u(i,j,k-1)) * h3sqinv &
176    4                     -rhs(i,j,k)) * hhhinv
177    4   p    v        end do
178    3   p          end do
179    2   p        end do
180    1          end do
181                !OCL END_SCACHE_ISOLATE_ASSIGN
182                !OCL END_SCACHE_ISOLATE_WAY
```

配列uの再利用性が向上



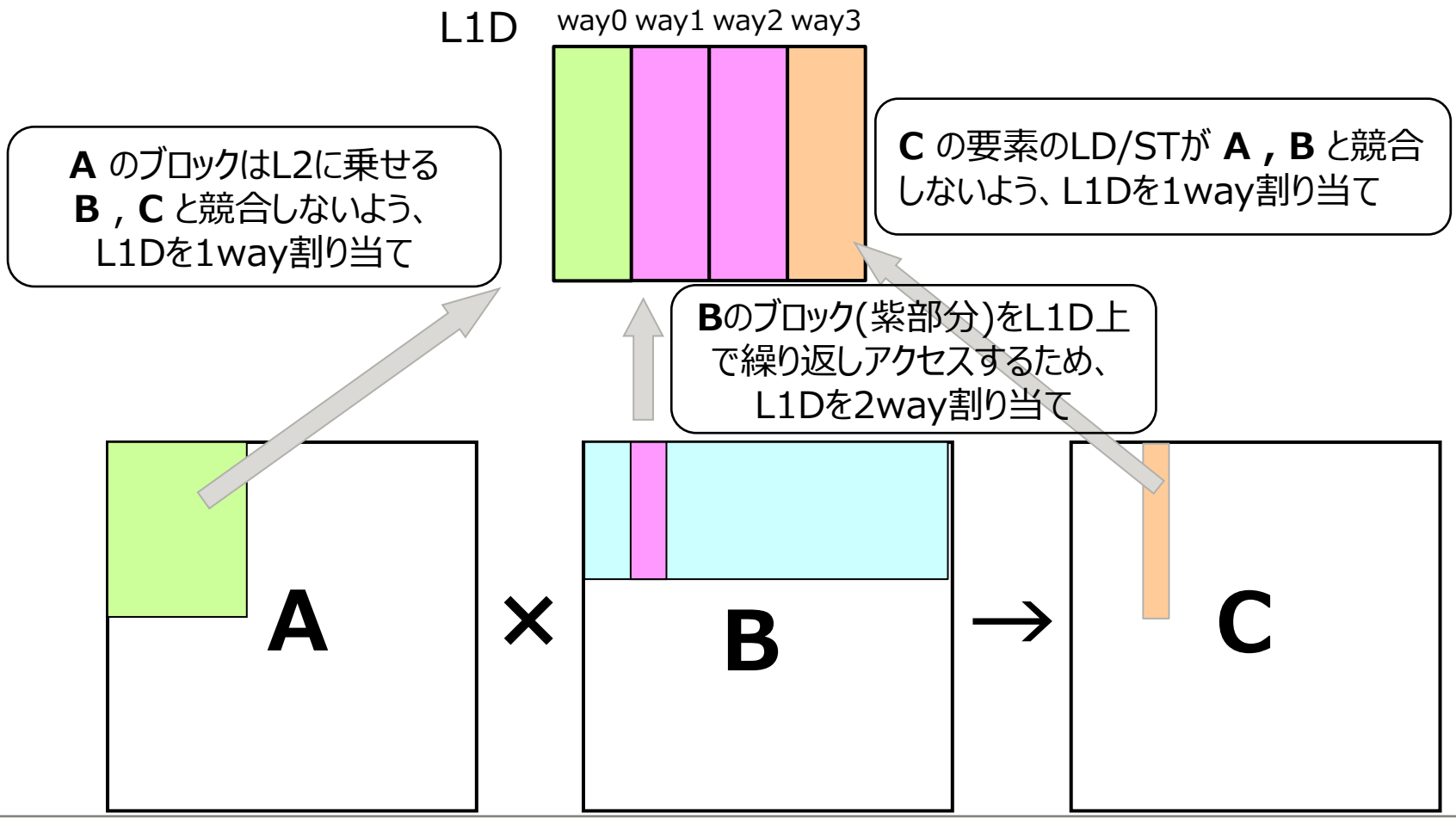
L2ミスが減少

	Memory throughput (GB/s)
改善前	215.62
改善後	205.39

	L2 miss	L2 miss rate (/Load-store instruction)	L2 miss demand rate (%) (/L2 miss)	L2 miss hardware prefetch rate (%) (/L2 miss)	L2 miss software prefetch rate (%) (/L2 miss)
改善前	2.46E+08	0.15	2.57%	98.19%	0.00%
改善後	1.95E+08	0.10	13.43%	87.39%	0.00%

DGEMMでのセクタキャッシュの効果 (1/2)

- 行列積(C=AB)は行列をブロックに分け，L1D，L2キャッシュに乗せて計算
- A,B のブロックはスラッシングを避けるため、作業域にコピー
- A,B,C はキャッシュに確実に乗せるために、L1のセクタキャッシュを使用
  特にL1D上で繰り返し使用する B のブロックが重要



DGEMMでのセクタキャッシュの効果 (2/2)

FUJITSU

- セクタキャッシュ利用あり/なしのDGEMMの性能
  - 1CMG

	DGEMM効率
セクタキャッシュ利用あり	94%
セクタキャッシュ利用なし	74%

- セクタキャッシュを利用することによって効率20%アップ

FUJITSU

高速ストア(zfill)

- 高速ストア(zfill)のイメージ
- コンパイラによる高速ストア(zfill)について
- 高速ストア命令
- 高速ストア(zfill)の動作条件
- 高速ストア(zfill)の評価
- 【参考】高速ストア(zfill)による性能向上
- 【注意】オンキャッシュデータの高速ストアについて

高速ストア(zfill)のイメージ

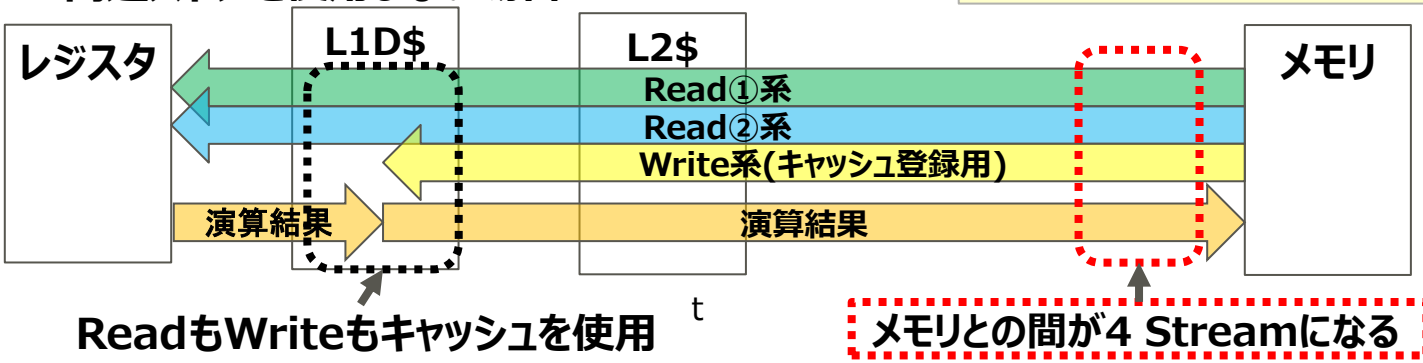
- 高速ストア（zfill）とは

  キャッシュ上に書き込み用のキャッシュラインを確保する機能です。これにより、メモリからのキャッシュラインの読み込みが削減できるので、メモリスループットがネックとなっているプログラムでは性能が改善します。

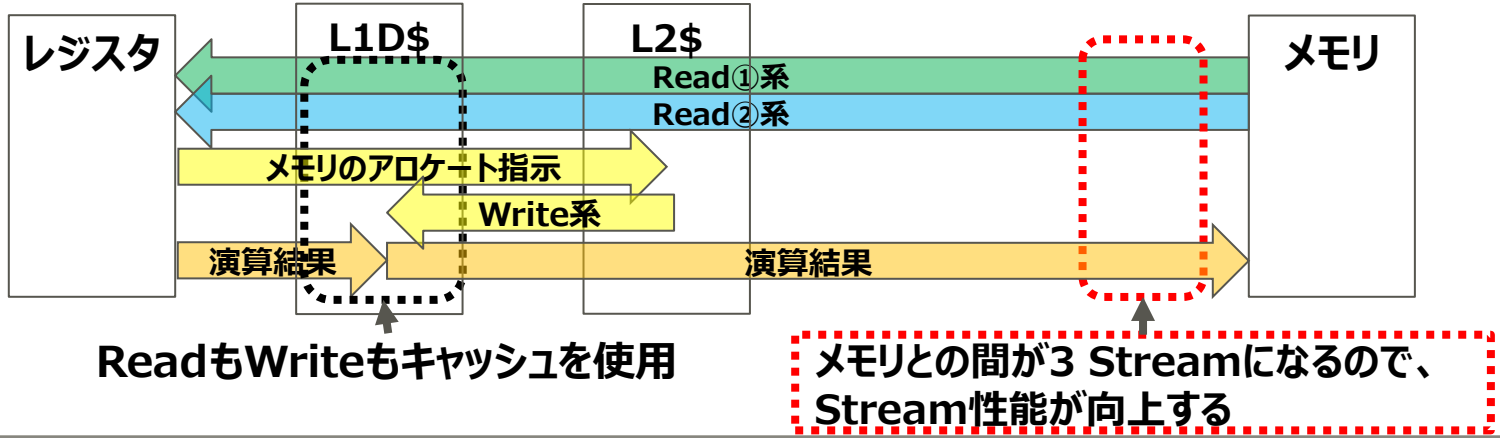
A64FXでは、DC ZVA命令を使用します。

- イメージ図 (Stream Triad ケースでのイメージ)

1．高速ストアを使用しない場合



2．高速ストアを使用した場合



コンパイラによる高速ストア(zfill)について

FUJITSU

■ オプション説明

-Kfast からの誘導はされない

オプション・OCL指示子の名称が変更になる
XFILL ⇒ zfill
※ただし古い指定(-KXFILLや!OCL XFILL)でも互換動作する

{ zfill[=*N*] | nozfill } 1 ≦ *N* ≦ 100
ループ内で書き込みのみ行う配列データについて、データをメモリからロードすることなく、キャッシュ上に書き込み用のキャッシュラインを確保する命令を使って、ストア命令を高速化します。
Nを指定することで、Nキャッシュライン先のデータを最適化の対象とします。
Nは1～100の範囲で指定できます。Nの指定を省略した場合、コンパイラが自動的に値を決定します。本オプションは -O2 オプション以上が有効な場合に意味があります。デフォルトは –Knozfill です。

■ 最適化概要

zfill は、データをメモリからロードすることなく、キャッシュ上に書き込み用のキャッシュラインを確保する命令を使い、データを高速にストアする最適化です。zfill は、ループ内でストアされる配列データに対して適用されます。
ただし、同一ループ内に参照がある配列、非連続アクセスされる配列またはIF構文配下でストアされる配列に対しては最適化されません。
また、zfill が適用された場合、二次キャッシュへのプリフェッチ命令は出力されません。

zfill の最適化が確保したキャッシュラインは必ずストアするようなループ変形を行うため、以下の最適化が適用できなくなります。これにより、実行性能が低下することがあります。
・ループアンローリング
・ループストライピング
ただし、zfill の最適化が動作した場合でも、ループ変形の最適化が動作すると、翻訳メッセージや最適化情報に上記の最適化が実施された情報を出力する場合があります。
また、以下の場合にも実行性能が低下することがあります。
・回転数が小さいループ
・-Kzfill=N でキャッシュライン数を指定した場合に、キャッシュラインに入る要素数より回転数が小さい場合
・メモリバンド幅ネックでないプログラム
-Kzfill の指定によって実行性能の低下が起きる場合は指定しないでください。zfill の制御は本来ループ単位で行うことが望ましいです。
したがって、プログラム全体に影響のあるオプション指定より、最適化指示子"ZFILL"を指定することを推奨します。

高速ストア命令

FUJITSU

FX100

```
ldd,s   [%xg0+-8],%f34
ldd,s   [%xg1+-8],%f32
subcc   %o1,1,%o1
ldd,s   [%xg0+24],%f40
ldd,s   [%xg1+24],%f38
ldd,s   [%xg0+56],%f44
ldd,s   [%xg1+56],%f42
ldd,s   [%xg0+88],%f48
ldd,s   [%xg1+88],%f46
ldd,s   [%xg0+120],%f52
ldd,s   [%xg1+120],%f50
ldd,s   [%xg0+152],%f56
ldd,s   [%xg1+152],%f54
fmaddd,s        %f36,%f34,%f32,%f34
ldd,s   [%xg0+184],%f60
ldd,s   [%xg1+184],%f58
fmaddd,s        %f36,%f40,%f38,%f40
ldd,s   [%xg0+216],%f64
ldd,s   [%xg1+216],%f62
add     %xg1,256,%xg1
fmaddd,s        %f36,%f44,%f42,%f44
add     %xg0,256,%xg0
fmaddd,s        %f36,%f48,%f46,%f48
fmaddd,s        %f36,%f52,%f50,%f52
fmaddd,s        %f36,%f56,%f54,%f56
fmaddd,s        %f36,%f60,%f58,%f60
fmaddd,s        %f36,%f64,%f62,%f64
stxa    %g0, [%xg2 + %xg6] 0xf4
std,sd  %f34,[%xg2+-8]
std,sd  %f40,[%xg2+24]
std,sd  %f44,[%xg2+56]
std,sd  %f48,[%xg2+88]
std,sd  %f52,[%xg2+120]
prefetch        [%xg2+760],2
std,sd  %f56,[%xg2+152]
std,sd  %f60,[%xg2+184]
std,sd  %f64,[%xg2+216]
add     %xg2,256,%xg2
bne,pt  %icc, .L91
```

※sxar命令は省略

FX100の高速ストア命令

A64FX

```
ld1d    {z1.d}, p0/z, [x4, -3, mul vl]
ld1d    {z0.d}, p0/z, [x9, -3, mul vl]
dc      ZVA, x7
add     x7, x7, 256
ld1d    {z4.d}, p0/z, [x4, -2, mul vl]
ld1d    {z3.d}, p0/z, [x9, -2, mul vl]
subs    w3, w3, 1
ld1d    {z6.d}, p0/z, [x4, -1, mul vl]
ld1d    {z5.d}, p0/z, [x9, -1, mul vl]
ld1d    {z16.d}, p0/z, [x4, 0, mul vl]
ld1d    {z7.d}, p0/z, [x9, 0, mul vl]
add     x9, x9, 256
add     x4, x4, 256
fmad    z1.d, p0/m, z2.d, z0.d
fmad    z4.d, p0/m, z2.d, z3.d
fmad    z6.d, p0/m, z2.d, z5.d
fmad    z16.d, p0/m, z2.d, z7.d
st1d    {z1.d}, p0, [x1, -3, mul vl]
st1d    {z4.d}, p0, [x1, -2, mul vl]
st1d    {z6.d}, p0, [x1, -1, mul vl]
st1d    {z16.d}, p0, [x1, 0, mul vl]
prfm    16, [x1, 824]
add     x1, x1, 256
bne     .L92
```

A64FXの高速ストア命令

高速ストアではハードウェアプリフェッチが抑止されるため、L1プリフェッチにはソフトウェアプリフェッチが使用される

高速ストア(zfill)を使用する場合、キャッシュライン分(256B)のストアを行う必要があるため、必要な回転数分をストライピングを使用し展開する。

※キャッシュライン(256B)分＝FX100では4SIMD×8回転分、A64FXでは8SIMD×4回転分

高速ストア(zfill)の動作条件 (1/2)

FUJITSU

- 動作条件
  - ストア対象の配列がイタレーション間で依存が無いこと
  - 定義のある配列の参照が無いこと
  - 連続的なメモリアクセスであること
- 動作するケース

連続的なメモリアクセスのため zfill が動作する。

【Triad】

```
 real*4 y(n), x1(n), x2(n), c0
 do j = 1, iter
!$omp parallel do
   Do i = 1, n
     y(i)=x1(i) + c0 * x2(i)
   End Do
 enddo
```

iter = 3
n = 3145728

【ストリーム的演算カーネル pattern1】

```
do l = 1, lall
   !$OMP PARALLEL DO
   do k = 1, kall
   do g = 1, gall
      rho(g,k,l) = PROG(g,k,l,1) / metrics(g,k,l)
      vx (g,k,l) = PROG(g,k,l,2) / PROG(g,k,l,1)
      vy (g,k,l) = PROG(g,k,l,3) / PROG(g,k,l,1)
      vz (g,k,l) = PROG(g,k,l,4) / PROG(g,k,l,1)
      ein(g,k,l) = PROG(g,k,l,5) / PROG(g,k,l,1)
   enddo
   enddo
enddo
```

```
do iq = 1, qall
   do l  = 1, lall
   !$OMP PARALLEL DO
   do k  = 1, kall
   do g  = 1, gall
      q(g,k,l,iq) = PROGq(g,k,l,iq) / PROG(g,k,l,1)
   enddo
   enddo
   enddo
enddo
```

gall=16900、kall=96
lall=1、qall=6

高速ストア(zfill)の動作条件 (2/2)

FUJITSU

動作するケース

【ストリーム的演算カーネル pattern2】

```
do l = 1, lall
  !$OMP PARALLEL DO
  do k = 1, kall
  do g = 1, gall
    tendency(g,k,l,1) = tendency_0(g,k,l,1) + tendency_11(g,k,l) + tendency_21(g,k,l)
    tendency(g,k,l,2) = tendency_0(g,k,l,2) + tendency_12(g,k,l) + tendency_22(g,k,l)
    tendency(g,k,l,3) = tendency_0(g,k,l,3) + tendency_13(g,k,l) + tendency_23(g,k,l)
    tendency(g,k,l,4) = tendency_0(g,k,l,4) + tendency_14(g,k,l) + tendency_24(g,k,l)
    tendency(g,k,l,5) = tendency_0(g,k,l,5) + tendency_15(g,k,l) + tendency_25(g,k,l)
    tendency(g,k,l,6) = tendency_0(g,k,l,6) + tendency_16(g,k,l) + tendency_26(g,k,l)
  enddo
  enddo
enddo
```

zfill が動作できるコードだが、現在のコンパイラではストア対象の配列の依存が無いことを見切れない。改善の余地がある。
(ループ分割をすることで現在のコンパイラでもzfillを動作させることはできるため今回はループ分割し評価する)

動作しないケース

【ストリーム的演算カーネル pattern3】

```
do l = 1, lall
  !$OMP PARALLEL DO
  do k = 1, kall
  do g = 1, gall
    value(g,k,l,1) = value(g,k,l,1) + tendency(g,k,l,1) * fraction
    value(g,k,l,2) = value(g,k,l,2) + tendency(g,k,l,2) * fraction
    value(g,k,l,3) = value(g,k,l,3) + tendency(g,k,l,3) * fraction
    value(g,k,l,4) = value(g,k,l,4) + tendency(g,k,l,4) * fraction
    value(g,k,l,5) = value(g,k,l,5) + tendency(g,k,l,5) * fraction
    value(g,k,l,6) = value(g,k,l,6) + tendency(g,k,l,6) * fraction
  enddo
  enddo
enddo
```

定義のある配列の参照があるためzfill が動作しない

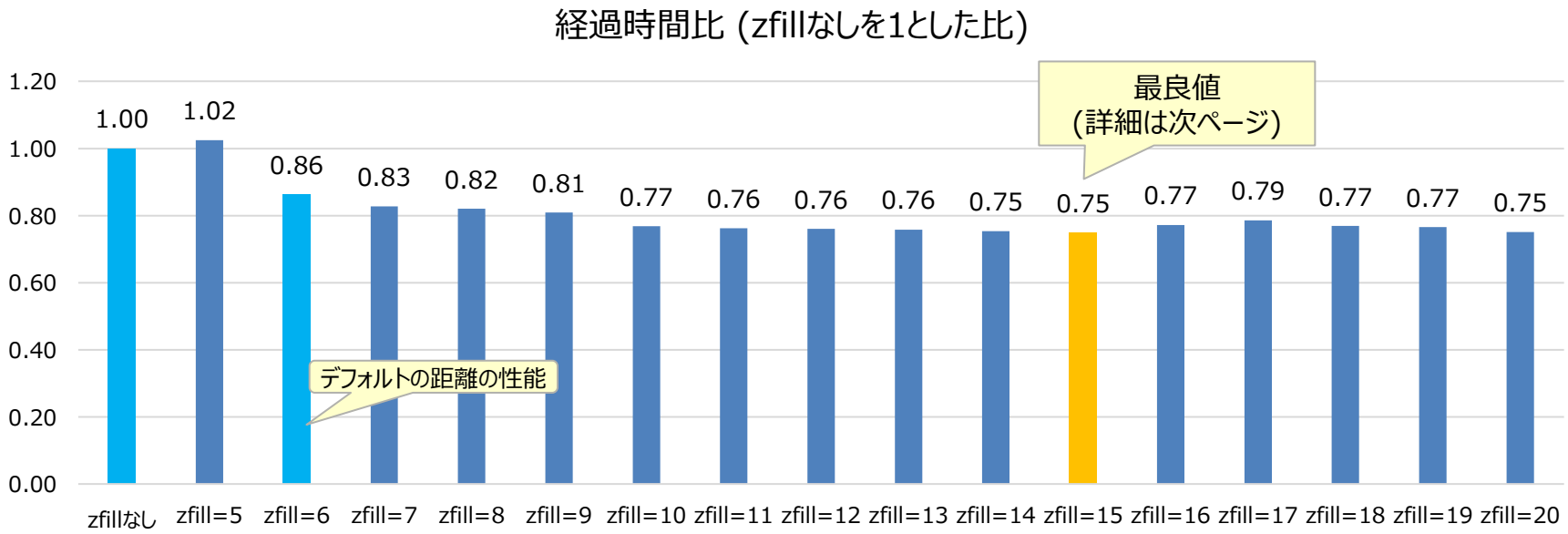
高速ストア(zfill)の評価：Triad （1/2)

- 高速ストア(zfill)は、プリフェッチと同様に、レイテンシを隠蔽する時間を見切り、命令を前もって実行する必要がある。
  Triadのケースにて高速ストア(zfill)の距離を検証した。

  ⇒ 15キャッシュラインで性能向上が見られた。
  (詳細を次ページに示す)

STRRAM Triad/1CMG実行

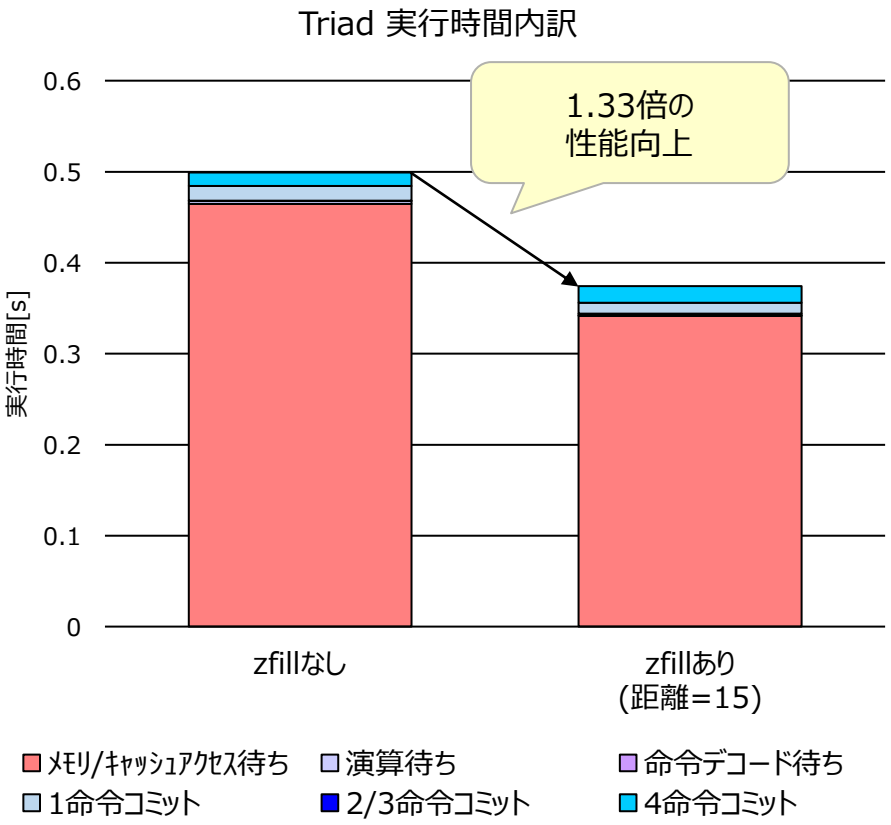
```
!$omp parallel
  Do j = 1, iter    iter=3000
!$omp do
     Do i = 1, n       n=1048512
       y(i)=x1(i) + c0 * x2(i)
     End Do
!$omp end do nowait
  End Do
!$omp end parallel
```



高速ストア(zfill)の評価：Triad （2/2)

FUJITSU

- zfill による性能向上



PAの値

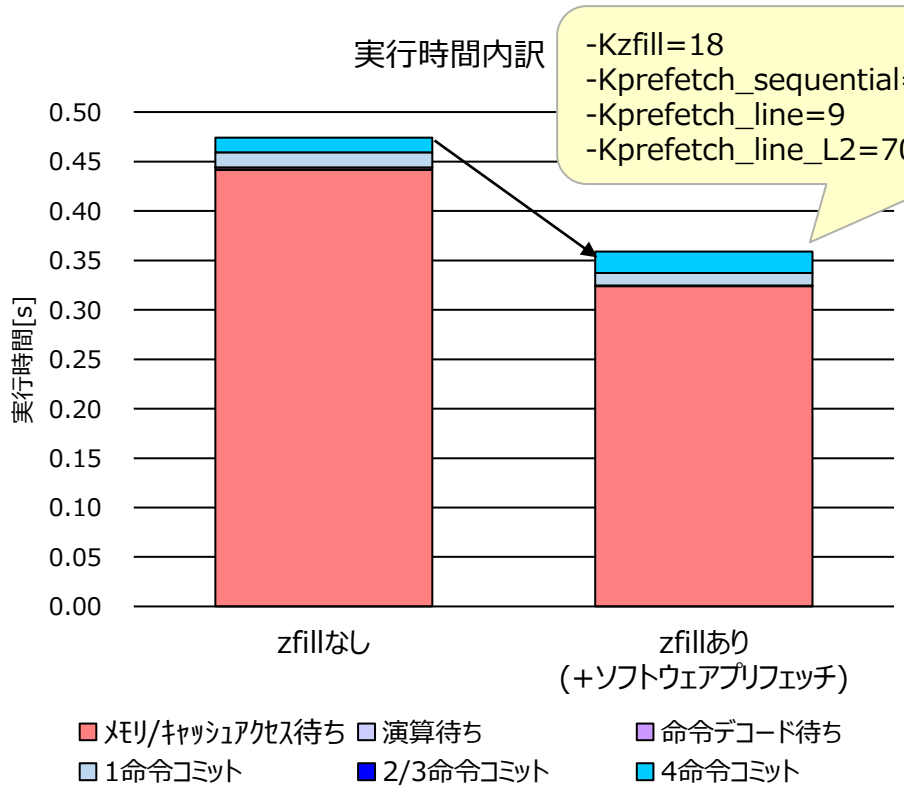
	zfill なし	zfill あり (距離=15)
集計スレッド番号	0	0
実行時間 [s]	0.499	0.374
有効総命令数	2.16E+09	2.18E+09
GFLOPS	12.60	16.83
メモリスループット [GB/s]	206.28	207.88
L1ビジー率/スレッド	59.97%	56.39%
L2ビジー率	87.65%	93.35%
メモリビジー率	80.58%	81.21%
浮動小数点パイプライン ビジー率/スレッド	FLA:4.61% FLB:2.70%	FLA:6.44% FLB:3.32%
L1ミス数/スレッド	2.47E+07	2.47E+07
L1ミス デマンド率/スレッド	3.65%	2.18%
L2ミス数/スレッド	2.55E+07	1.72E+07
L2ミス デマンド率/スレッド	5.83%	5.27%

高速ストア(zfill)の効果によりメモリアクセス数が削減し、性能が向上した

高速ストア(zfill)の評価：Triad
【参考:最内ループ繰返し数10倍】

FUJITSU

■ zfill とソフトウェアプリフェッチの性能



PAの値

	zfill なし	zfill あり (+ソフトウェアプリフェッチ)
集計スレッド番号	0	0
実行時間 [s]	0.474	0.359
有効総命令数	2.16E+09	2.52E+09
GFLOPS	13.27	17.54
メモリスループット [GB/s]	**222.5**	210.8
L1ビジー率/スレッド	52.9%	55.09%
L2ビジー率	85.1%	94.39%
メモリビジー率	86.8%	82.32%
浮動小数点パイプライン ビジー率/スレッド	FLA:4.90% FLB:2.80%	FLA:7.14% FLB:3.01%
L1ミス数/スレッド	2.46E+07	2.46E+07
L1ミス デマンド率/スレッド	9.40%	0.08%
L2ミス数/スレッド	2.62E+07	1.64E+07
L2ミス デマンド率/スレッド	12.21%	0.31%

※アクセスサイズ(最内ループ繰返し数、配列サイズ)を240MB(10倍)にして評価

ソフトウェアプリフェッチと zfill を的確に行えば、210.8GB/ s まで性能向上する

高速ストア(zfill)の評価：ストリーム的演算カーネル pattern1 (1/2)

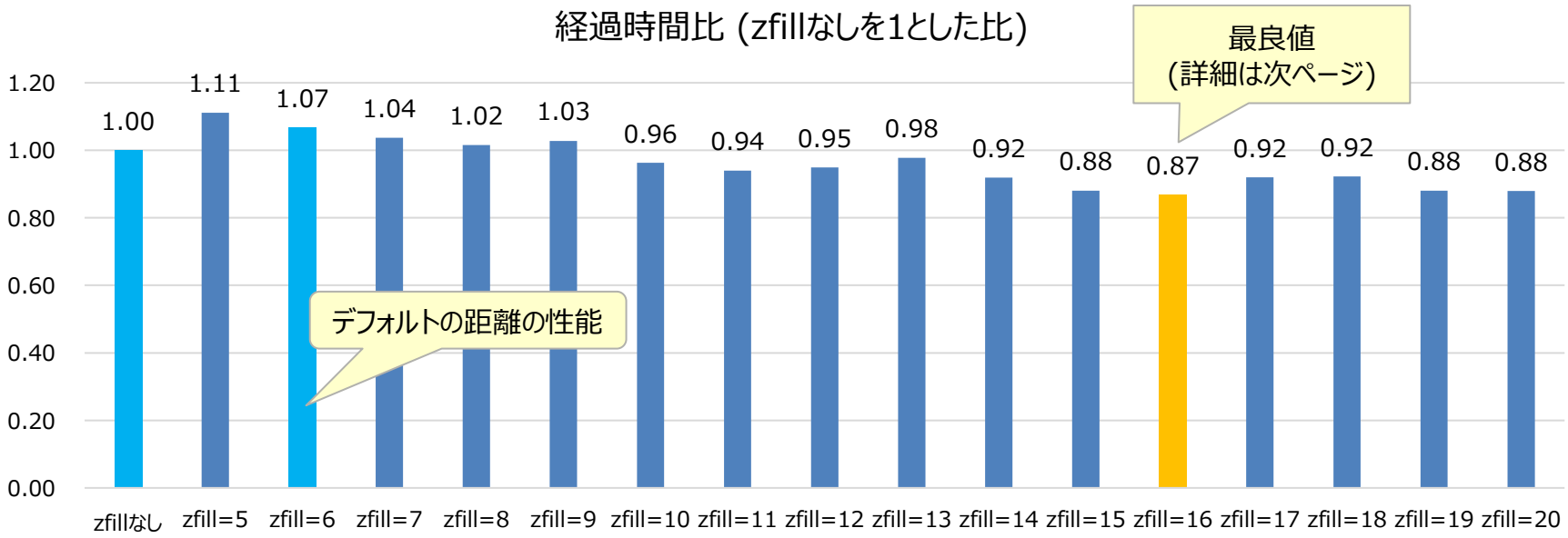
- ストリーム的演算カーネル(パターン1)のケースにて高速ストア(zfill)の距離を検証した。
  ⇒16キャッシュラインで性能向上が見られた。
  (次ページに詳細を示す)

ストリーム的演算カーネル pattern1

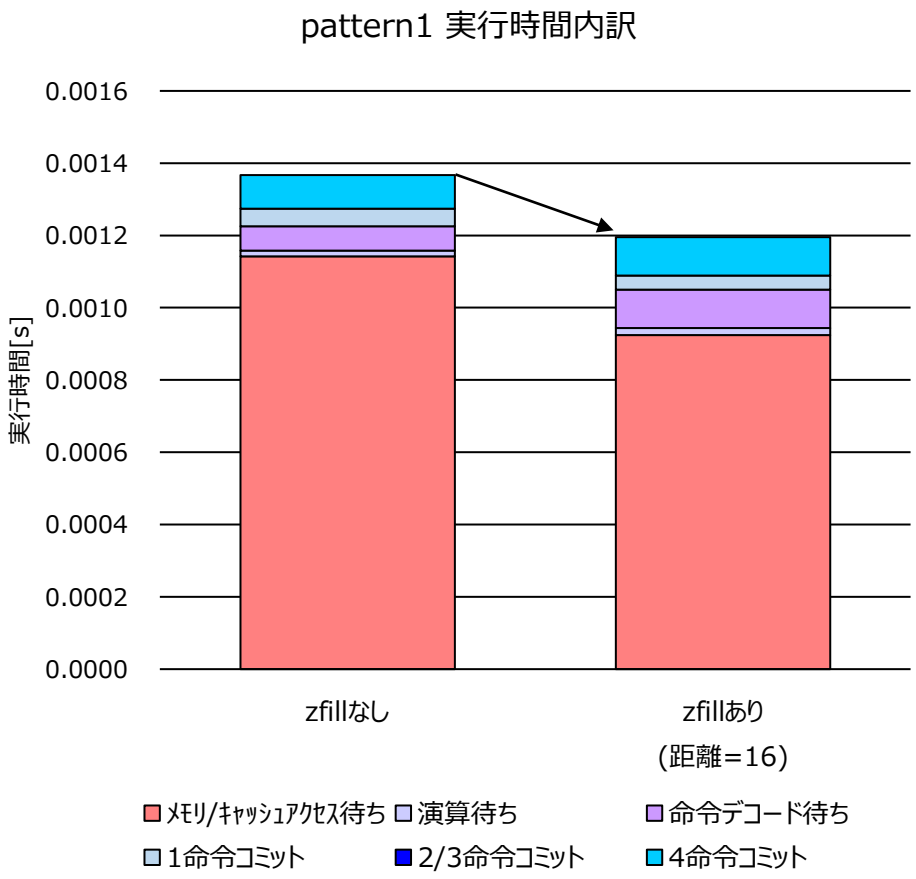
```
do l = 1, lall
  !$OMP PARALLEL DO
  do k = 1, kall
  do g = 1, gall
    rho(g,k,l) = PROG(g,k,l,1) / metrics(g,k,l)
    vx (g,k,l) = PROG(g,k,l,2) / PROG(g,k,l,1)
    vy (g,k,l) = PROG(g,k,l,3) / PROG(g,k,l,1)
    vz (g,k,l) = PROG(g,k,l,4) / PROG(g,k,l,1)
    ein(g,k,l) = PROG(g,k,l,5) / PROG(g,k,l,1)
  enddo
  enddo
enddo
```

```
do iq = 1, qall
  do l  = 1, lall
  !$OMP PARALLEL DO
  do k  = 1, kall
  do g  = 1, gall
    q(g,k,l,iq) = PROGq(g,k,l,iq) / PROG(g,k,l,1)
  enddo
  enddo
  enddo
enddo
```

単精度評価
gall=16900、kall=96
lall=1、qall=6



高速ストア(zfill)の評価：ストリーム的演算カーネル pattern1 (2/2)



PAの値

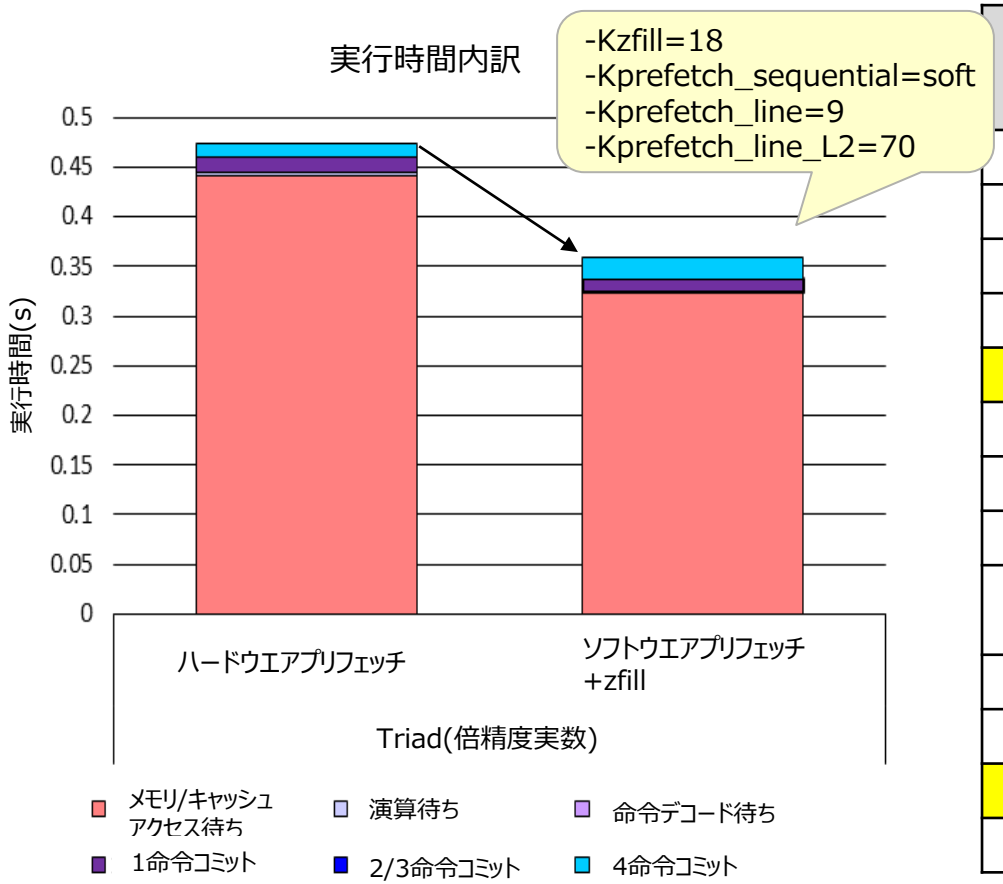
	zfill なし	zfill あり (距離=16)
集計スレッド番号	0	0
実行時間 [s]	0.001372	0.001206
有効総命令数	10775582	11158332
GFLOPS	91.10	103.7
メモリスループット [GB/s]	211.31	194.84
L1ビジー率	47.98%	46.77%
L2ビジー率	83.70%	83.85%
メモリビジー率	83.36%	76.91%
浮動小数点パイプライン ビジー率/スレッド	FLA:11.64% FLB:10.92%	FLA:12.92% FLB:12.59%
L1ミス数/スレッド	7.15E+04	7.21E+04
L1ミス デマンド率/スレッド	10.93%	5.37%
L2ミス数/スレッド	6.96E+04	5.49E+04
L2ミス デマンド率/スレッド	15.31%	13.25%

高速ストア(zfill)により、メモリへのアクセス数削減の効果が確認できた。
ただし、zfill 先のデフォルト値の変更が必要となる

【参考】高速ストア(zfill)による性能向上

FUJITSU

■ ソフトウェアプリフェッチと zfill の性能



PAの値

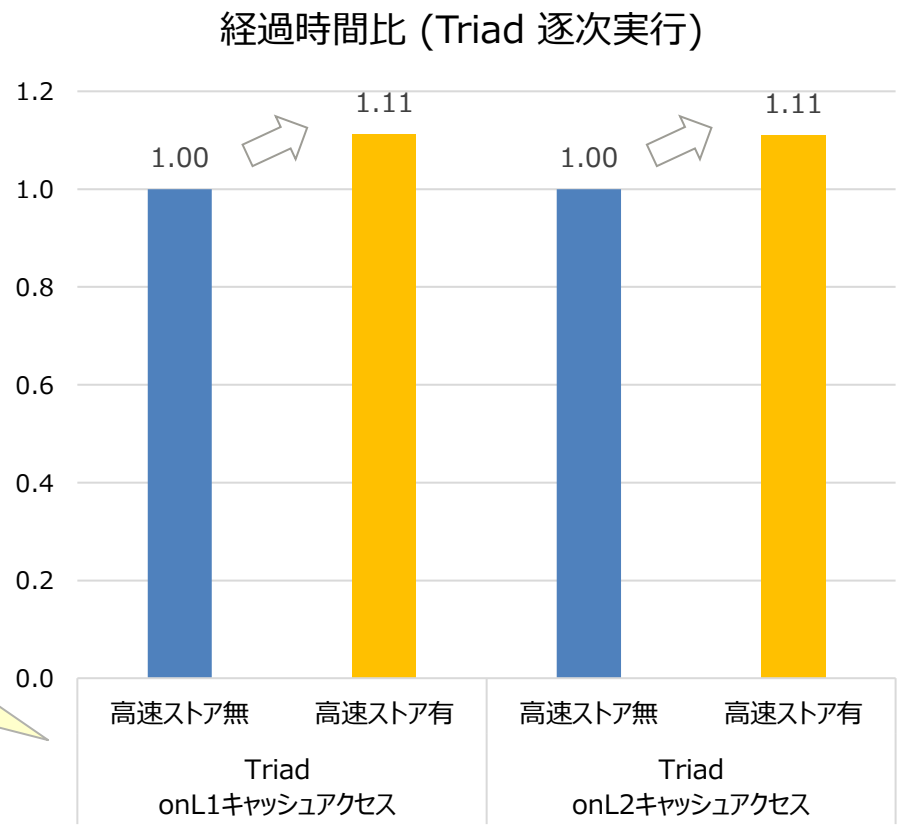
	ハードウェア プリフェッチ	ソフトウェア プリフェッチ + zfill
集計スレッド番号	0	0
実行時間 [s]	0.474	0.359
有効総命令数	2.16E+09	2.52E+09
GFLOPS	13.27	17.54
メモリスループット [GB/s]	**222.5**	210.8
L1ビジー率/スレッド	52.9%	55.09%
L2ビジー率	85.1%	94.39%
メモリビジー率	21.7%	20.58%
浮動小数点パイプライン ビジー率/スレッド	FLA:4.90% FLB:2.80%	FLA:7.14% FLB:3.01%
L1ミス数/スレッド	2.46E+07	2.46E+07
L1ミス デマンド率/スレッド	9.40%	0.08%
L2ミス数/スレッド	**2.62E+07**	1.64E+07
L2ミス デマンド率/スレッド	**12.21%**	0.31%

※アクセスサイズ(最内ループ繰返し数、配列サイズ)を240MB(10倍)にして評価

高速ストア(zfill)を的確に行うことで、210.8GB/ s まで性能向上する

【注意】オンキャッシュデータの高速ストアについて

- キャッシュアクセス時の高速ストア
  オンキャッシュのデータに対して高速ストアを行うと、逆に性能低下を起こす可能性がある。
  必要性を見極めて指定する必要がある。



L1キャッシュ上、L2キャッシュ上のデータアクセスに対する高速ストアで、約11%の性能低下が見られた

高速ストア(zfill)は対象の配列のアクセスを見極めた上で使用することで性能向上する

データアクセスのアライメント制約

- アライメントの影響を受けるマイクロアーキ
- Gather命令の纏め上げ機能
- Multiple Structures 命令の高速動作について
- WB(ライトバッファ)の動作について

以下のアライメント変化に対する性能については、後述の基礎カーネル性能の「アライメントの変化による性能影響」を参照のこと。

- 測定結果：連続SIMDロード
- 測定結果：連続SIMDストア
- 測定結果：Gatherロード
- 測定結果：Scatterストア
- 測定結果：構造体ロード(LD2命令)
- 測定結果：構造体ストア(ST2命令)

アライメントの影響を受けるマイクロアーキ

FUJITSU

- ロード/ストア命令の中にはアライメントの変化により性能差が生じるケースがある。
  アライメントの影響を受ける要因として以下が想定される。
  - Gather命令の纏め上げ機能
  - Multiple Structures 命令の高速動作
  - WB(ライトバッファ)の動作(WBスプリットとWBマージ)
  - (ストアフェッチバイパスの動作)

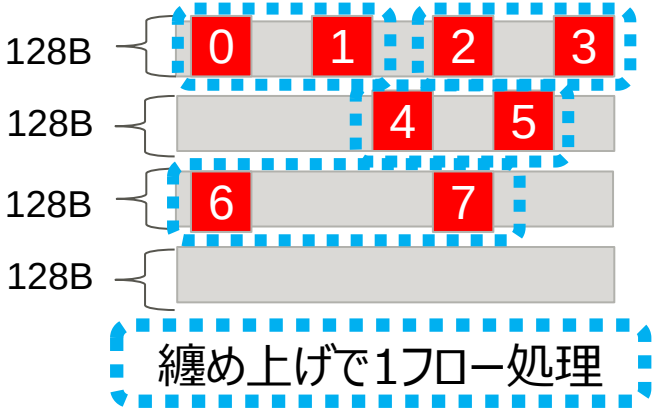
Gather命令の纏め上げ機能

FUJITSU

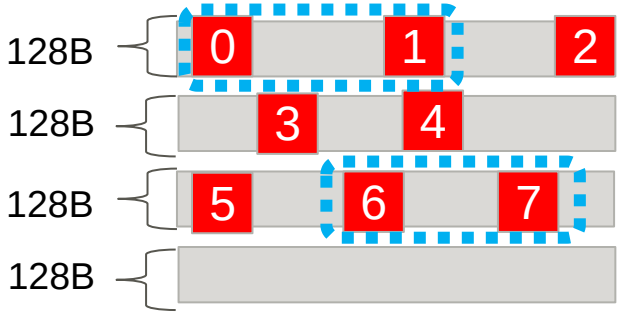
Gather命令の纏め上げ機能

Gather命令は、1つのFPから同時発行される2要素が隣接する場合に高速処理ができる。
隣接する2要素のアドレスが128バイト内で一致する場合には、まとめて1フローで処理する
ことで高速化できる。

- 隣接2要素が、同一の128バイトブロックに属する場合、それらを纏めて1フローで処理する。以下のアドレスパターン例で　の部分が纏め上げされて 、2要素をL1D$パイプライン1フローで処理する。
  - アドレスパターン例1



  - アドレスパターン例2

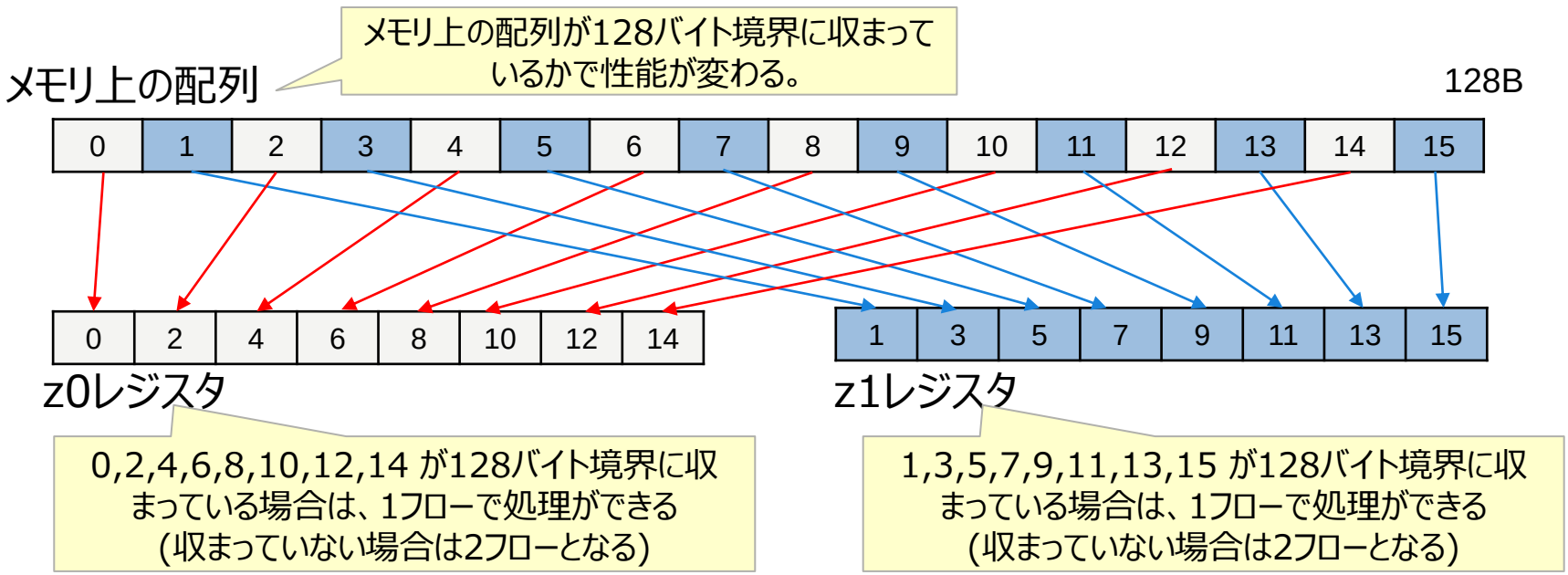


Gather命令の纏め上げ機能を活用するために配列の先頭アドレスには注意して実装して下さい。

Multiple Structures 命令の高速動作について

Multiple Structures 命令(LD2、ST2)は、扱うデータのアドレス範囲が128 バイト境界に収まる場合には1 レジスタあたり1 フローで処理することができ、高速化につながる。

例) LD2 命令　ld2d {z0.d, z1.d}, p0/z, [(配列アドレス)]

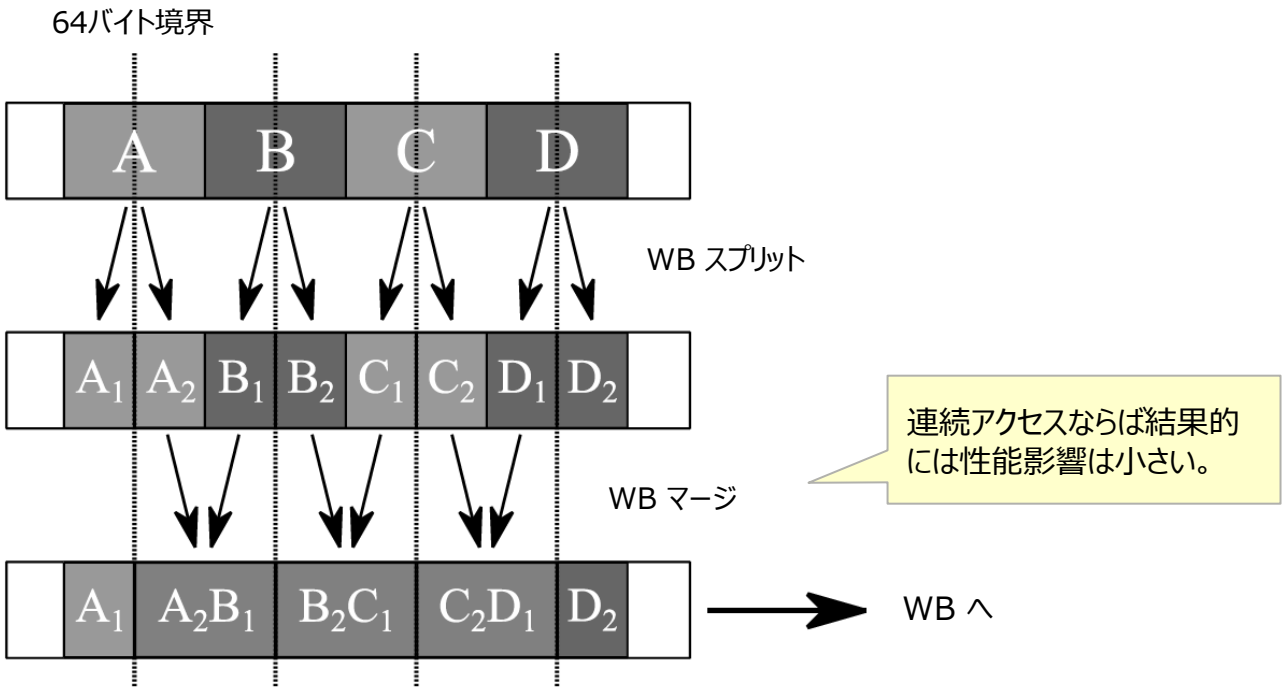


WB(ライトバッファ)の動作について

FUJITSU

WB(ライトバッファ)の動作について

SP(ストアポート)からWB(ライトバッファ) への書き込み操作は64バイト単位で行われるため、ストアデータが64バイト境界をまたいで存在していると、それらを全てストアするには必ずデータの個数よりも多くの書き込み操作を要する。　ストアデータが各々64 バイト境界をまたいで存在している場合、SPからWB に書き込む際に64バイト境界で分割(WBスプリット) しなければならない。この後、分割されたデータのうち同一の64バイト境界に収まっているものについては1つに合体(WBマージ) されてからWBに書き込まれる。



FUJITSU

アウトオブオーダー(OoO)実行の検証

- アウトオブオーダー関連リソース
- 検証の目的
- アウトオブオーダー効果検証
- アウトオブオーダーとソフトによるスケジューリングの協調
- まとめ

アウトオブオーダー関連リソース

■ アウトオブオーダー関連リソース

項目	単位	A64FX	京	FX100
演算レイテンシ FL(FMA)	(サイクル)	9	6	6
インフライト命令数(CSE)	(命令/コア)	128	48	64
インフライトロード命令数(FP)	(命令/コア)	40	20	32
インフライトストア命令数(SP)	(命令/コア)	24	8	20
RSA(アドレス計算+整数演算)	(エントリ/コア)	20	10	16
RSE(整数演算+実数演算)	(エントリ/コア)	20x2	10+16	16+20
RSBR(分岐)	(エントリ/コア)	19	8	10
リネーミングレジスタ数(GPR)	(エントリ/コア)	64	32	48
リネーミングレジスタ数(FPR)	(エントリ/コア)	96	48	64
リネーミングレジスタ数(PDR)	(エントリ/コア)	32	-	-

検証の目的

FUJITSU

- アウトオブオーダー効果検証
  - 実機にてアウトオブオーダーによる性能向上効果があることを確認する
  - 演算数(連鎖)の変化によりアウトオブオーダーの効果の変化を確認する
  - A64FXのアウトオブオーダー資源数と最適なスケジューリングに必要な資源数から定量的な性能変化を確認する
- アウトオブオーダーとソフトによるスケジューリングの協調
  - ソフト(コンパイラ)と、ハード(アウトオブオーダー)のスケジューリングの協調の効果を確認する

アウトオブオーダー効果検証 (1/4)

FUJITSU

評価概要

配列yに対し、定数c0～c9を使用しFMA演算を行う。
FMAの数を1～9の9ケース用意する

配列 y にアクセスする計算式だが、mが引数で渡されてくるため、ソフト(コンパイラ)ではmを見切ることができなく、iのループを順次に実行するようなコードとなっている。以下のケースを比較する

①OoO有効/ソフトのスケジューリング無効
m=0：依存関係がなくなり処理を追い越すことが可能 (-Kfast,noswp,unroll=2)

②OoO無効/ソフトのスケジューリング無効
m=1：依存関係が現れiのループは順次に実行しなければならない (-Kfast,noswp,unroll=2)

③SWPL(OoO有効/ソフトのスケジューリング有効)
m=0：NORECURRENCEを指定(-Kfast)

【1FMA】
```
 Do j = 1, n
  Do i = 1, 8
   y(i,j) = c0 + y(i,j-m) * c1
  End Do
 End Do
```

【2FMA】
```
 Do j = 1, n
  Do i = 1, 8
   y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m) * c2 )
  End Do
 End Do
```

：(省略)

【9FMA】
```
 Do j = 1, n
  Do i = 1, 8
   y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m)* ¥
            (c2 + y(i,j-m)*(c3 + y(i,j-m)* ¥
            (c4 + y(i,j-m)*(c5 + y(i,j-m)* ¥
            (c6 + y(i,j-m)*(c7 + y(i,j-m)* ¥
            (c8 + y(i,j-m)* c9)))))))))
  End Do
 End Do
```

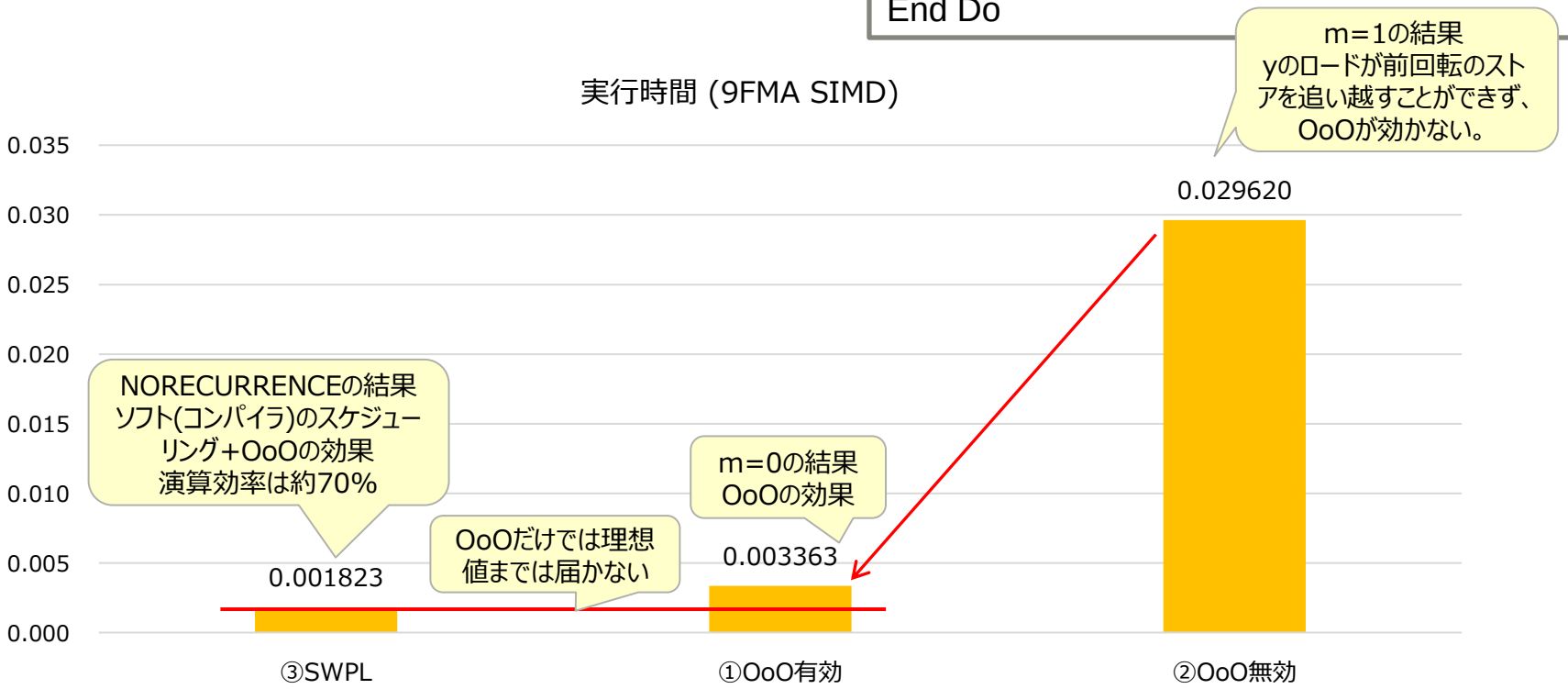
アウトオブオーダー効果検証 (2/4)

FUJITSU

■ 実行結果(9FMA)

OoO効果あるが、演算数・連鎖多いため、
性能はSWPL ＞ OoOとなる。

```
Do j = 1, n
  Do i = 1, 8
    y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m)* ¥
             (c2 + y(i,j-m)*(c3 + y(i,j-m)* ¥
             (c4 + y(i,j-m)*(c5 + y(i,j-m)* ¥
             (c6 + y(i,j-m)*(c7 + y(i,j-m)* ¥
             (c8 + y(i,j-m)* c9))))))))
  End Do
End Do
```

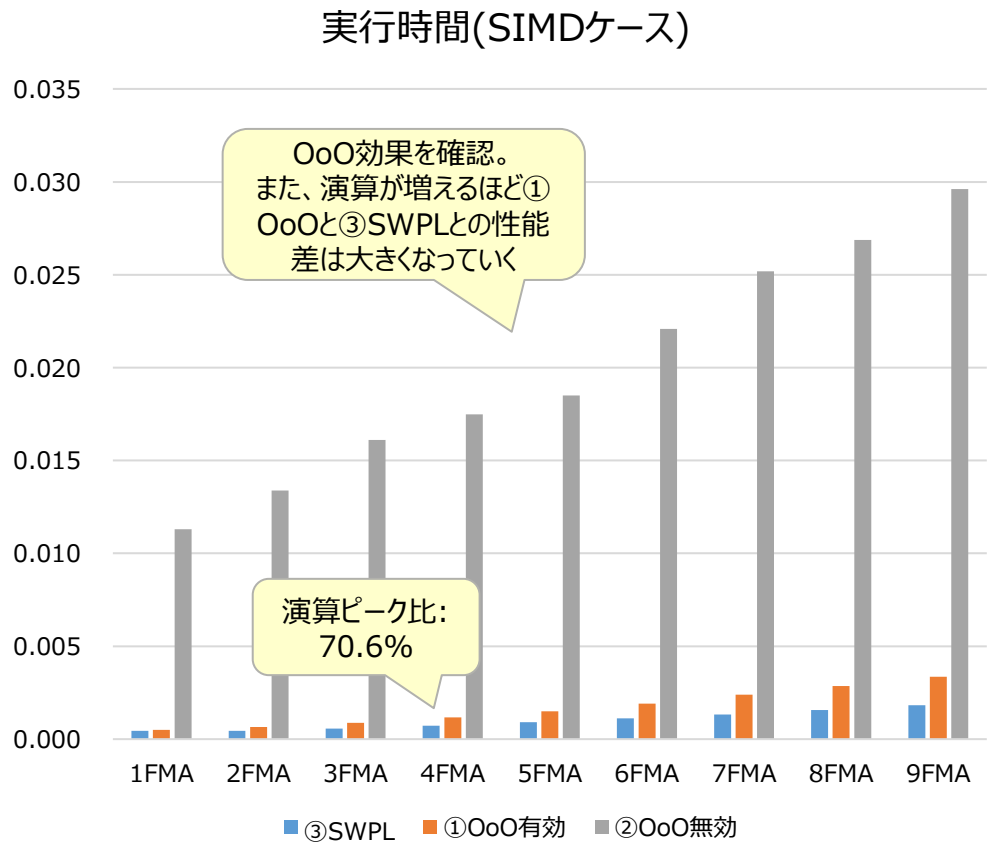


アウトオブオーダー効果検証 (3/4)

FUJITSU

■ 各ケースの実行結果(実行時間)

FMA2ケース程度のループでは、
性能が、SWPL≒OoO　となる。




```
【1FMA】
 Do j = 1, n
  Do i = 1, 8
   y(i,j) = c0 + y(i,j-m) * c1
  End Do
 End Do

【2FMA】
 Do j = 1, n
  Do i = 1, 8
   y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m) * c2 )
  End Do
 End Do

 : (省略)

【9FMA】
 Do j = 1, n
  Do i = 1, 8
   y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m)* ¥
            (c2 + y(i,j-m)*(c3 + y(i,j-m)* ¥
            (c4 + y(i,j-m)*(c5 + y(i,j-m)* ¥
            (c6 + y(i,j-m)*(c7 + y(i,j-m)* ¥
            (c8 + y(i,j-m)* c9))))))))
  End Do
 End Do
```

アウトオブオーダー効果検証 (4/4)

FUJITSU

- 理想的(演算レイテンシを隠蔽できる)な実行に必要なOoO資源数

	A64FX 資源数	2FMA	3FMA	9FMA
必要 レジスタ数	32+96	39	40	46
必要 RSE数	20x2	36	54	162
必要 RSBR数	16	9	9	9
必要 FP数	40	18	18	18
必要 SP数	24	18	18	18

資源数から乖離していくほど良いスケジューリングができなくなる

必要レジスタ数 = 定数+(2*レイテンシ*パイプライン数)
必要RSE数 = FMA数*レイテンシ*パイプライン数
必要RSBR数 = レイテンシ*パイプライン数/UNROLL数
必要FP数 = ロード数*レイテンシ*パイプライン数
必要SP数 = ストア数*レイテンシ*パイプライン数

【2FMA】
```
Do j = 1, n
  Do i = 1, 8
    y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m) * c2 )
  End Do
End Do
```

【3FMA】
```
Do j = 1, n
  Do i = 1, 8
    y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m) * ¥
             (c2 + y(i,j-m)* c3 ))
  End Do
End Do
```

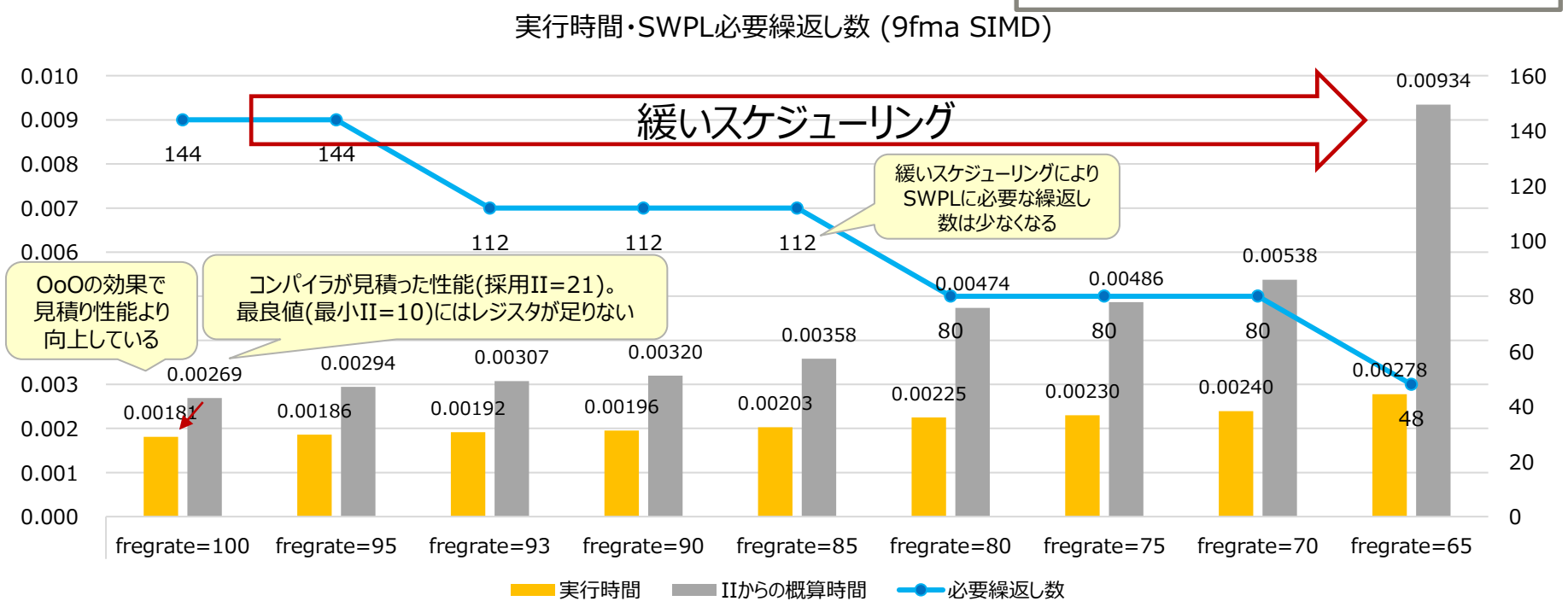
: (省略)

【9FMA】
```
Do j = 1, n
  Do i = 1, 8
    y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m)* ¥
             (c2 + y(i,j-m)*(c3 + y(i,j-m)* ¥
             (c4 + y(i,j-m)*(c5 + y(i,j-m)* ¥
             (c6 + y(i,j-m)*(c7 + y(i,j-m)* ¥
             (c8 + y(i,j-m)* c9))))))))
  End Do
End Do
```

アウトオブオーダーとソフトによるスケジューリングの協調

- アウトオブオーダーとソフトによるスケジューリング(SWPL)
  SWPLはOoOと協調することで性能の向上が実現されている
- 緩いスケジューリングとアウトオブオーダー
  SWPLによるスケジューリングを緩くすることで、命令列の並びは悪くなるがが、OoOにより劣化幅は大きくならない。
  緩くした分SWPLに必要なループ繰返し数が減るため、繰返し数が既知なループのチューニングに活用することができる。

```
Do j = 1, n
  Do i = 1, 8
    y(i,j) = c0 + y(i,j)*(c1 + y(i,j)* ¥
             (c2 + y(i,j)*(c3 + y(i,j)* ¥
             (c4 + y(i,j)*(c5 + y(i,j)* ¥
             (c6 + y(i,j)*(c7 + y(i,j)* ¥
             (c8 + y(i,j)* c9)))))))
  End Do
End Do
```



まとめ

FUJITSU

- アウトオブオーダー実行の効果を確認。

- 演算の連鎖(数)が増えていくほどソフトによるスケジューリング(SWPL)の重要性が高くなる。
  → SWPL不可ループに対するループ分割でのSWPL促進

- SWPLを意図的に緩くスケジューリングすることで明に繰返し数の決まっているループに対してチューニング余地となる可能性がある。

FUJITSU

SIMD幅

- SIMD幅について
- 固定長SIMD幅指定の性能
- SIMD幅を意識した最適化(チューニング)による性能影響
- 可変長SIMD指定の性能
- 【参考】SIMD幅による電力の変化

SIMD幅について

FUJITSU

- プロセッサでサポートしているSIMD幅

  ARM の SVE拡張は Vector Length(SIMD幅)が 128bit から 2048bit の範囲の128bit 単位で実装者が自由に決めることが可能である。
  A64FX では Vector Length を 512bit で実装を行っている。

  - A64FX でサポートする Vector Length は以下。
    - 512bit
    - 256bit
    - 128bit

- コンパイラオプション

  - -Ksimd_reg_size={ 128 | 256 | 512 | agnostic }
    デフォルトは、-Ksimd_reg_size=512 です。
    - simd_reg_size={ 128 | 256 | 512 }
      SVEのベクトルレジスタサイズを指定します。単位はビットです。翻訳時に、本オプションで指定した値がSVEのベクトルレジスタサイズであるとみなして、最適化を実施します。ただし、生成した実行可能プログラムは、本オプションで指定したサイズのSVEのベクトルレジスタを実装しているCPUアーキテクチャでのみ正常に動作します。
    - simd_reg_size=agnostic
      SVEのベクトルレジスタを特定のサイズとみなさず翻訳を行い、実行時にSVEのベクトルレジスタサイズを決定する実行可能プログラムを作成します。この実行可能プログラムは、CPUアーキテクチャに実装されたSVEのベクトルレジスタサイズによらず実行可能です。
      ただし、-Ksimd_reg_size={128|256|512}オプションを指定した場合と比べて、実行性能が低下する場合があります。

固定長SIMD幅指定の性能

FUJITSU

- SIMD幅による性能の変化

SIMD幅256bitで2倍、SIMD幅128bitで4倍までが適正。
それより大きい場合には別の問題を起こしている可能性がある

プログラム	浮動小数点演算ピーク比	メモリスループットピーク比	実行時間比		
			SIMD幅=512 (Default)	SIMD幅=256	SIMD幅=128
Adventure.region1	26.98%	62.38%	1.00	1.67	3.25
Adventure.region2	20.85%	0.02%	1.00	1.49	2.74
FFB.callap_kernel2	39.12%	76.33%	1.00	1.02	1.19
FFB.spmmv_vec8	28.38%	33.94%	1.00	0.99	5.11
GAMERA.TIMER_COMP_MATVEC_IF	20.29%	8.90%	1.00	1.56	2.81
GENESIS.Nonb15F(June)	7.73%	6.41%	1.00	1.78	3.38
GENESIS.Nonb15F(July)	8.04%	4.46%	1.00	1.45	2.07
GENESIS.PairList(June)	6.71%	7.96%	1.00	1.79	1.88
GENESIS.PairList(July)	4.76%	3.04%	1.00	1.08	1.14
NICAM.Horizontal_Adv_flux	2.59%	1.94%	1.00	1.20	1.96
NICAM.Horizontal_Adv_limiter	14.37%	41.61%	1.00	1.21	1.67
NICAM.Radiation_adding	14.48%	47.57%	1.00	1.27	1.84
NICAM.Radiation_dtrn31	9.64%	36.38%	1.00	1.24	1.81
NICAM.Radiation_ptfit2	11.84%	64.94%	1.00	1.05	1.35
NICAM.Vertical_Adv_limiter	6.10%	73.82%	1.00	0.97	1.01
NICAM.diffusion	15.62%	43.44%	1.00	1.78	3.35
NICAM.divdamp	33.16%	36.00%	1.00	1.56	2.81
NICAM.vi_rhow_solver	15.49%	56.23%	1.00	1.34	2.00
NICAM_nsw6.M3_mp_nsw6_OMP9	10.29%	68.25%	1.00	1.01	1.01
NICAM_nsw6.M3_vadv1d_getflux_new	3.06%	54.72%	1.00	1.19	1.54
streamlike_pattern1_check	9.38%	78.69%	1.00	1.00	1.01
streamlike_pattern2_check	2.38%	78.10%	1.00	1.54	2.94
streamlike_pattern3_check	4.15%	78.41%	1.00	0.99	1.00
MD	78.42%	0.02%	1.00	2.70	3.87
N体	90.36%	0.00%	1.00	1.95	3.32
(幾何平均)			1.00	1.34	2.01

次ページで分析する

基本的にはバンド幅ネックのカーネルでは影響はない

SIMD幅を意識した最適化(チューニング)による性能影響

FUJITSU

FFB.spmmv_vec8の場合

[SIMD幅256bit]

```
182                !$omp parallel default(none) private(IP,II,JJ,IP2)
183                !$omp&shared(NP,LIST,AX,A,X)
184                !$omp do
185                !ocl nounroll
186                !ocl swp
          <<< Loop-information Start >>>
          <<<  [OPTIMIZATION]
          <<<    SOFTWARE PIPELINING(IPC: 3.01, ITR: 48, MVE: 3, POL:S)
          <<<    PREFETCH(HARD) Expected by compiler :
          <<<      LIST, A, AX
          <<< Loop-information  End >>>
187    1   p              DO IP=1,NP
191    1                !ocl nounroll,loop_nofission
          <<< Loop-information Start >>>
          <<<  [OPTIMIZATION]
             <<<    SIMD(VL: 8)
                <<< Loop-information  End >>>
192    2   p   v            DO JJ=1,8
193    2   p   v              AX(JJ,IP  )=AX(JJ,IP  )+A( 1,IP  )*X(JJ,LIST( 1,IP  ))
194    2   p   v              AX(JJ,IP  )=AX(JJ,IP  )+A( 2,IP  )*X(JJ,LIST( 2,IP  ))
   :
226    2   p   v            ENDDO
   :
238    1   p              ENDDO
239                !$omp end do
240                !$omp end parallel
```

最内8回転をSIMD幅で展開し、外側ループでスケジューリングの最適化を行っている

[SIMD幅128bit]

```
182                !$omp parallel default(none) private(IP,II,JJ,IP2)
183                !$omp&shared(NP,LIST,AX,A,X)
184                !$omp do
185                !ocl nounroll
186                !ocl swp
187    1   p              DO IP=1,NP
191    1                !ocl nounroll,loop_nofis
          <<< Loop-information St
          <<<  [OPTIMIZATION]
          <<<    SIMD(VL: 4)
          <<<    PREFETCH(HARD) Expected by compiler :
          <<<      X
          <<<    PREFETCH(SOFT) : 30
          <<<     SEQUENTIAL : 30
          <<<      X: 28, AX: 2
          <<<    SPILLS :
          <<<      GENERAL   : SPILL 4  FILL 20
          <<<      SIMD&FP   : SPILL 0  FILL 4
          <<<      SCALABLE  : SPILL 0  FILL 0
          <<<      PREDICATE : SPILL 0  FILL 0
          <<< Loop-information  End >>>
192    2   p   v            DO JJ=1,8
193    2   p   v              AX(JJ,IP  )=AX(JJ,IP  )+A( 1,IP  )*X(JJ,LIST( 1,IP  ))
194    2   p   v              AX(JJ,IP  )=AX(JJ,IP  )+A( 2,IP  )*X(JJ,LIST( 2,IP  ))
   :
226    2   p   v            ENDDO
   :
238    1   p              ENDDO
239                !$omp end do
240                !$omp end parallel
```

SIMD幅が小さくなり、最内ループをSIMDで展開しても回転が残り、スケジューリングの最適化が最内ループに適用されるようになった

SIMD幅を意識した最適化(チューニング)が行われた場合、
想定以上に性能差異が発生する可能性がある。

可変長SIMD指定の性能

FUJITSU

■ 可変長SIMD(-Ksimd_reg_size=agnostic)の性能

(可変長SIMD指定＋システムSIMD幅512を1とした比)

プログラム	-Ksimd_reg_size=512 (Default)	-Ksimd_reg_size=agnostic		
	SIMD幅=512	SIMD幅=512	SIMD幅=256	SIMD幅=128
Adventure.region1	1.00	1.00	1.69	3.32
Adventure.region2	1.02	1.00	1.62	2.94
FFB.callap_kernel2	**0.18**	1.00	0.87	0.92
FFB.spmmv_vec8	**0.25**	1.00	1.09	1.55
GAMERA.TIMER_COMP_MATVEC_IF	0.85	1.00	1.64	2.84
GENESIS.Nonb15F(June)	0.73	1.00	1.70	3.18
GENESIS.Nonb15F(July)	0.82	1.00	1.31	1.94
GENESIS.PairList(June)	0.59	1.00	1.07	1.15
GENESIS.PairList(July)	1.02	1.00	1.14	1.24
NICAM.Horizontal_Adv_flux	0.68	1.00	1.46	1.63
NICAM.Horizontal_Adv_limiter	0.96	1.00	1.20	1.73
NICAM.Radiation_adding	0.77	1.00	1.39	2.26
NICAM.Radiation_dtrn31	0.74	1.00	1.23	1.64
NICAM.Radiation_ptfit2	1.04	1.00	1.15	1.53
NICAM.Vertical_Adv_limiter	1.04	1.00	1.02	1.04
NICAM.diffusion	**0.10**	1.00	1.07	1.22
NICAM.divdamp	0.81	1.00	1.42	2.15
NICAM.vi_rhow_solver	1.01	1.00	1.28	1.89
NICAM_nsw6.M3_mp_nsw6_OMP9	1.00	1.00	1.01	1.07
NICAM_nsw6.M3_vadv1d_getflux_new	0.81	1.00	1.42	2.19
streamlike_pattern1_check	0.99	1.00	1.00	1.00
streamlike_pattern2_check	0.81	1.00	1.39	2.28
streamlike_pattern3_check	1.01	1.00	1.02	1.01
MD	0.98	1.00	2.71	3.88
N体	0.96	1.00	1.81	3.46
(幾何平均)	0.72	1.00	1.31	1.79

実行SIMD幅

1.0以下は、可変長SIMDオプション指定で性能低下したコード。
SIMD幅を意識した最適化が可能な場合には性能が悪くなる場合がある。

【参考】SIMD幅による電力の変化

FUJITSU

■ SIMD幅512bit→128bitでの電力効率

プログラム	性能向上比 (SIMD幅512→128) ※大きいほうが性能向上	電力比 (SIMD幅512→128) ※大きいほうが電力が大きくなる	性能向上比/電力比 ※小さい方が効率的
Adventure.region1	0.80	0.71	0.88
Adventure.region2	0.43	0.88	2.04
FFB.callap_kernel2	0.83	0.93	1.12
FFB.spmmv_vec8	0.20	0.86	4.43
GAMERA.TIMER_COMP_MATVEC_IF	0.36	0.93	2.62
GENESIS.Nonb15F(June)	0.30	0.94	3.13
GENESIS.Nonb15F(July)	0.50	1.02	2.03
GENESIS.PairList(June)	0.91	1.01	1.12
GENESIS.PairList(July)	0.86	0.90	1.05
NICAM.Horizontal_Adv_flux	0.56	0.94	1.68
NICAM.Horizontal_Adv_limiter	0.60	0.93	1.53
NICAM.Radiation_adding	0.55	1.01	1.83
NICAM.Radiation_dtrn31	0.55	0.84	1.52
NICAM.Radiation_ptfit2	0.73	0.82	1.13
NICAM.Vertical_Adv_limiter	0.97	0.96	0.99
NICAM.diffusion	0.28	0.91	3.24
NICAM.divdamp	0.37	0.88	2.38
NICAM.vi_rhow_solver	0.50	1.03	2.05
NICAM_nsw6.M3_mp_nsw6_OMP9	0.98	0.94	0.96
NICAM_nsw6.M3_vadv1d_getflux_new	0.65	0.99	1.53
streamlike_pattern1_check	1.00	1.04	1.04
streamlike_pattern2_check	0.34	0.87	2.56
streamlike_pattern3_check	0.98	1.05	1.07
(幾何平均)	0.56	0.93	1.64

SIMD幅512より効率は悪い

SIMD幅128bitとすることで電力は多少抑えることはできるが、
性能の低下率から考えて効率的だとは言えない。

FUJITSU

電力制御

- ブーストモードとは
- エコモードとは
- モード毎、基礎カーネルの電力
- ブーストモードでの性能検証
- エコモードでの性能検証
- 【参考】実アプリ(NICAM)の時系列電力と性能

ブーストモードとは

- ブーストモードとは
  A64FXでは、性能と電力効率のバランスを考えて、ノーマルモードでは2.0GHzで動作させるが、アプリケーション（ジョブ）を少しでも速く動作させたい場合のことを考え、CPUの周波数を2.2GHzで動作させるブーストモードも用意した。ブーストモードでは、ノーマルモードより高い周波数で動作させるため、CPUの駆動電圧を昇圧する。
- ブーストモードでは、CPUの周波数は2.2GHzとなるが、HBMの周波数はノーマルモードと同じである。下表にブーストモードにおける性能向上が見込めるコード特性を示す。

コード特性	ブーストモードによる性能効果
L1バンド幅ネック	○
L2バンド幅ネック	○
メモリバンド幅ネック	×
演算ネック	○
アクセスレイテンシネック	○
Tofu通信バンド幅ネック	×
Tofu通信レイテンシネック	△　（CPU動作部分のみ）

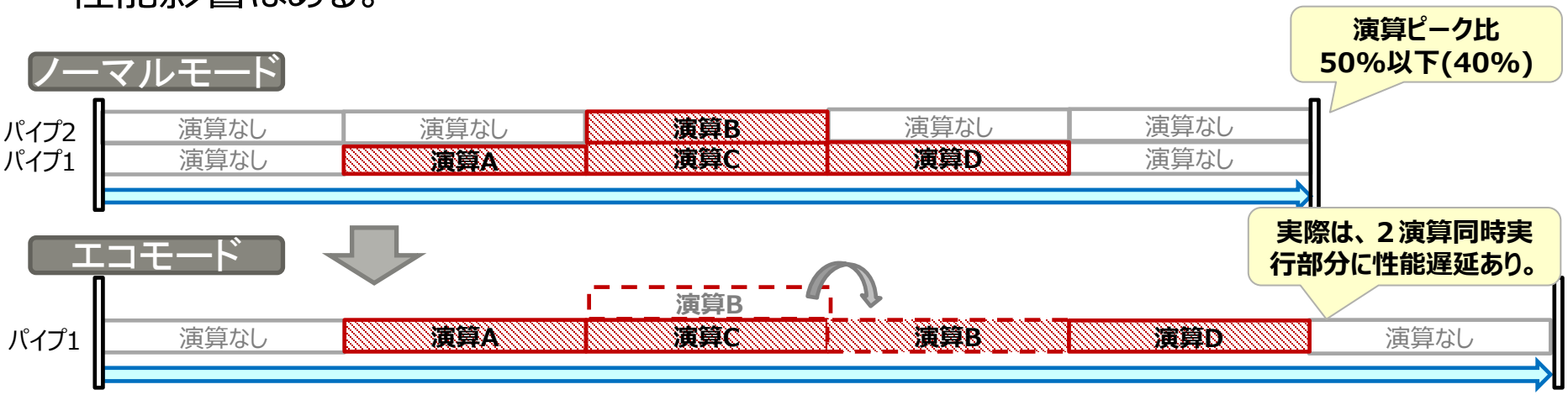
エコモードとは (1/2)

- エコモードとは
  エコモードとは、FPU のパイプラインを1つのみ使用するパワーノブをONにするとともに、底上げ電力を浮動小数点演算1パイプ分のみと半減させるモードである。エコモードではピークの浮動小数点演算性能は低下するが、Run 状態およびStanby状態の電力削減に効果がある。

- ノーマルモードとエコモードの演算パイプラインの動き

  エコモードでは、演算パイプラインを１パイプ分のみと半減させるため、浮動小数点演算ピーク比、50%を超えるコードに関して確実に性能低下となる。

  浮動小数点ピーク比は50%を超えないケースでも、演算命令の動作タイミングにより性能影響はある。



エコモードとは (2/2)

- 下表にエコモードにおける性能影響（低下）が見込まれるコード特性を示す。
- 電力に関しては、確実にノーマルモードよりもエコモードは削減される。

コード特性	エコモードによる性能影響
L1バンド幅ネック	なし
L2バンド幅ネック	なし
メモリバンド幅ネック	なし
演算ネック	あり
アクセスレイテンシネック	なし
ノード内バリアネック	なし
Tofu通信バンド幅ネック	なし
Tofu通信レイテンシネック	なし

モード毎、基礎カーネルの電力 (1/2)：検証条件

FUJITSU

■ 検証条件

項目	詳細
測定パターン	- L1キャッシュバンド幅ネックコード - L2キャッシュバンド幅ネックコード - メモリバンド幅ネックコード (配列の値が異なる3パターン) ①すべて0 ②先頭から通番(0,1,2,3,4,...) ③先頭から64バイト毎、0(全ビット0)と、最小値(全ビット1)を、交互に設定 - レイテンシネック(L2キャッシュ)コード
アクセス範囲 ※bss領域を使用	- L1キャッシュバンド幅ネックコード：48KB(L1サイズの3/4) ※コア毎 - L2キャッシュバンド幅ネックコード：4MB(L2サイズの1/2) ※CMG毎 - メモリバンド幅ネックコード：240MB(L2サイズの30倍) ※CMG毎 - レイテンシネックコード：1.8MB ※CMG毎
配列型	- キャッシュバンド幅ネックコード：倍精度実数型 - メモリバンド幅ネックコード、レイテンシネックコード：8バイト整数型
測定並列数 (プロセス、スレッド)	4プロセス、12スレッド (1CMGに1プロセス、1コアに1スレッド)
翻訳オプション	-Kfast,openmp

測定コード1
(キャッシュ・メモリバンド幅ネック)

```
!$omp parallel
  do j = 1, iter
!$omp do
      Do i = 1, n
         y(i)=x1(i) + c0 * x2(i)
      End Do
!$omp end do nowait
  enddo
!$omp end parallel
```

測定コード2 (レイテンシネック)

```
for (i = 0; i < rep; i++) {
   p = index2[0];
   for (j = 0; j < NL; j++) {
      p = (uint64_t **)*p;
   }
   ans = p;
 }
```

モード毎、基礎カーネルの電力 (2/2)：電力値

FUJITSU

■ 検証結果

プログラム	浮動小数点演算ピーク比	メモリスループットピーク比	ビジー率 (ノーマルモード、エコモードOFF時)					電力			
								ノーマルモード(2.0GHz)		ブーストモード(2.2GHz)	
			L1 ビジー率	L2 ビジー率	メモリ ビジー率 ※ハードの問題は残る	浮動小数点PL ビジー率	整数PL ビジー率	エコモード OFF	エコモード ON	エコモード OFF	エコモード ON
DGEMM(※参考)	(94%)	-	(79%)	-	-	(94%)	-	177	125	206	143
STREAM(※参考)	-	(81%)	-	(92%)	(81%)	-	-	196	164	218	178
L1バンド幅ネック	21.58%	0.00%	93.68%	0.00%	0.00%	54.87%	33.86%	159	132	188	152
L2バンド幅ネック	7.55%	0.00%	98.09%	97.42%	0.00%	19.62%	9.67%	151	122	176	138
メモリバンド幅ネック①	0.00%	82.50%	76.66%	91.97%	82.50%	6.42%	3.02%	169	143	194	155
メモリバンド幅ネック②	0.00%	81.80%	75.91%	90.74%	81.80%	6.37%	2.99%	189	157	209	171
メモリバンド幅ネック③	0.00%	82.50%	76.61%	91.90%	82.50%	6.42%	3.02%	199	171	219	186
レイテンシネック(L2キャッシュ)	0.00%	0.00%	10.59%	12.67%	0.00%	0.00%	0.54%	121	88	139	99

初期値の違いにより
約20~30W差の発生を確認

①:配列の値に0を使用
②:配列の値に通番(0,1,2,3,4,...)を使用
③:配列の値に0(全ビット0)、最小値(全ビット1)を、64バイト毎に交互
(括弧):参考値

注意）電力値についてはCPU毎の個体差あるため、数値については相対値として見てください。

【参考】モード毎、アプリカーネルのビジー率と電力値

FUJITSU

■ ビジー率と電力の関係性

ソート対象

プログラム	単・倍精度	浮動小数点演算ピーク比	メモリスループットピーク比	ビジー率（ノーマルモード、エコモードOFF時）					補正後電力			
									ノーマルモード(2.0GHz)		ブーストモード(2.2GHz)	
				L1 ビジー率	L2 ビジー率	メモリ ビジー率 ※ハードの問題は残る	浮動小数点 PLビジー率	整数PL ビジー率	エコモード OFF	エコモード ON	エコモード OFF	エコモード ON
Adventure.region1	倍	23.5%	54.7%	66.0%	89.5%	54.7%	54.3%	1.6%	203	158	226	179
NICAM.Radiation_dtrn31	単	6.3%	77.4%	53.9%	80.5%	77.4%	6.8%	8.5%	198	163	222	180
FFB.callap_kernel2	単	38.8%	74.8%	57.8%	57.2%	74.8%	80.0%	3.5%	194	140	227	157
streamlike_pattern3_check	単	4.0%	85.1%	43.3%	78.8%	85.1%	5.4%	4.2%	192	155	213	173
NICAM.Horizontal_Adv_flux	単	13.8%	46.4%	72.3%	66.6%	46.4%	17.8%	10.5%	191	160	221	178
streamlike_pattern1_check	単	11.9%	84.5%	46.2%	81.0%	84.5%	10.6%	1.5%	189	158	215	170
NICAM_nsw6.M3_mp_nsw6_OMP9	単	10.5%	67.2%	41.4%	64.6%	67.2%	9.8%	3.4%	187	153	209	170
NICAM.vi_rhow_solver	単	14.8%	54.3%	45.8%	53.5%	54.3%	14.6%	22.4%	185	147	213	172
NICAM.divdamp	単	13.2%	41.4%	53.3%	75.1%	41.4%	12.4%					168
Adventure.region0	倍	16.5%	93.7%	45.1%	81.0%	93.7%	19					161
NICAM.Radiation_adding	単	11.3%	88.2%	48.9%	84.8%	88.2%	9.1%					156
streamlike_pattern2_check	単	2.3%	86.0%	48.0%	80.9%	86.0%	4.1%					161
NICAM_nsw6.M3_vadv1d_getflux_new	単	3.0%	79.1%	37.1%	71.0%	79.1%	11.5%					150
NICAM.Vertical_Adv_limiter	単	34.5%	37.0%	26.9%	26.3%	37.0%	49.7%	4.3%	170	127	196	143
FFB.spmmv_vec8	単	27.0%	32.3%	52.8%	27.0%	32.3%	39.9%	20.1%	166	131	185	150
NICAM.Radiation_ptfit2	単	16.6%	49.0%	52.6%	46.5%	49.0%	41.3%	4.8%	166	127	189	141
NICAM.Horizontal_Adv_limiter	単	8.6%	34.7%	37.2%	53.1%	34.7%	14.6%	8.1%	162	127	189	144
Adventure.region2	倍	20.0%	0.0%	78.3%	47.1%	0.0%	50.7%	4.3%	156	119	182	138
NICAM.diffusion	単	17.4%	10.4%	50.6%	72.3%	10.4%	16.1%	13.1%	156	120	179	137
GAMERA.TIMER_COMP_MATVEC_IF	単	21.0%	10.4%	57.7%	9.8%	10.4%	36.9%	15.0%	149	113	172	128
QCD.ddd_in_s_	単	26.3%	0.1%	41.3%	18.5%	0.1%	20.9%	6.9%	149	113	172	129
QCD.jinv_ddd_in_s_	単	27.4%	0.1%	49.4%	25.8%	0.1%	24.5%	7.0%	148	113	172	129
GENESIS.Nonb15F(June)	単	7.4%	7.0%	51.3%	12.2%	7.0%	21.7%	1.6%	142	112	165	126
GENESIS.PairList(June)	単	7.0%	9.1%	27.8%	6.8%	9.1%	18.9%	29.9%	139	106	160	119
GENESIS.Nonb15F(July)	単	7.9%	5.7%	29.3%	7.2%	5.7%	13.9%	6.2%	133	101	151	114
GENESIS.PairList(July)	単	5.5%	4.2%	24.0%	6.3%	4.2%	14.9%	31.9%	133	99	152	113

メモリビジー率が高いほうが
上位に集まりやすい

FUJITSU

ブーストモードでの性能検証

- ブーストモード：コード特性毎の性能検証条件
- ブーストモード：基礎カーネル性能
- 【参考】アプリカーネル測定結果

ブーストモード：コード特性毎の性能検証条件

FUJITSU

測定条件

		パターン
検証コード	基礎演算カーネル	①Triad L1キャッシュアクセス (L1バンド幅性能) ②Triad L2キャッシュアクセス (L2バンド幅性能) ③Triad メモリアクセス (メモリバンド幅性能) ④DGEMM (演算性能) ⑤L1アクセスレイテンシコード (演算/L1アクセスレイテンシ性能)
	アプリカーネル	28カーネル
測定コア数	基礎演算カーネル	①⑤：1コア実行、 ②③④：12コア実行(CMG0)
	アプリカーネル	12コア実行(CMG0)
翻訳オプション	基礎演算カーネル	①⑤ -Kfast ② -Kfast,openmp -Kprefetch_sequential=soft ¥ -Kprefetch_cache_level=1 -Kprefetch_line=4 ③ -Kfast,openmp -Kzfill=18 ¥ -Kprefetch_sequential=soft -Kprefetch_line=9 -Kprefetch_line_L2=70 ④ -Kfast,openmp
	アプリカーネル	各カーネル毎で個別に決められたオプションをそのまま使用
アクセス範囲	基礎演算カーネル	①⑤ L1キャッシュサイズの半分(32KB) ② L2キャッシュサイズの半分(4MB) ③ L2キャッシュサイズの30倍(240MB) ④ TRANSA=N, TRANSB=N, M=23040, N=23040, K=640

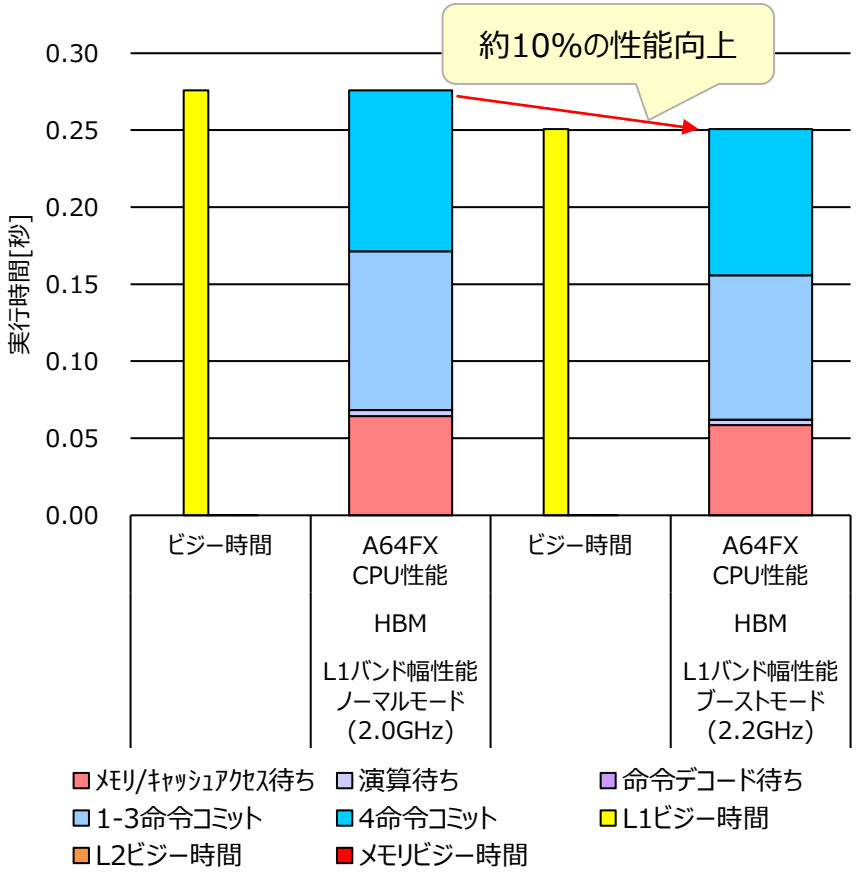
ブーストモード：基礎カーネル性能 (1/5)

FUJITSU

- L1バンド幅性能（１コア）

→ 約10%の性能向上となることを確認。
(想定通り)

```
Do j = 1, iter
      Do i = 1, n
            y(i)=x1(i) + c0 * x2(i)
      End Do
End Do
```



	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)	比率 (2.2GHz ÷2.0GHz)
ソースコード版数	L1バンド幅性能	L1バンド幅性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	1	1	
集計スレッド番号	0	0	
実行時間[s]	0.276	0.251	0.91
有効総命令数	1.29.E+09	1.29.E+09	
GFLOPS(プロセス)	14.85	16.33	1.10
メモリスループット [GB/s/プロセス]	0.00	0.00	
L1ビジー率/スレッド	99.95%	99.93%	
L2ビジー率/スレッド	0.00%	0.00%	
メモリビジー率/スレッド	0.00%	0.00%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	58.61% 34.38%	58.60% 34.37%	
L1スループット [GB/s/スレッド]	178.1	196.0	1.10

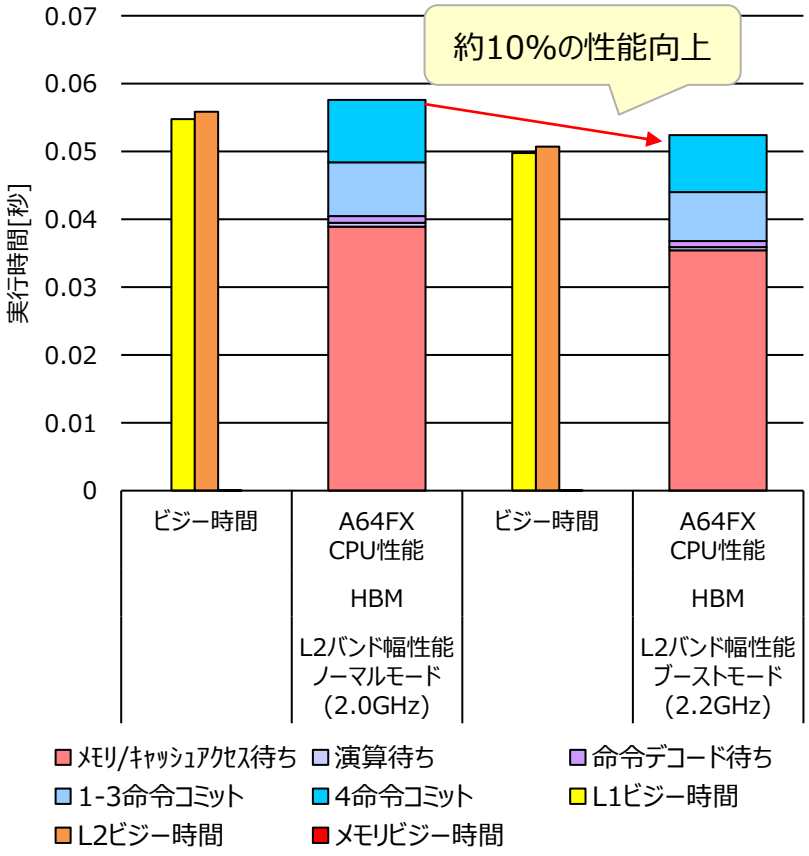
ブーストモード：基礎カーネル性能 (2/5)

FUJITSU

- L2バンド幅性能（１ＣＭＧ）

→ 約10%の性能向上となることを確認。
(想定通り)

```
!$omp parallel
   Do j = 1, iter
!$omp do
        Do i = 1, n
            y(i)=x1(i) + c0 * x2(i)
        End Do
!$omp end do nowait
   End Do
!$omp end parallel
```



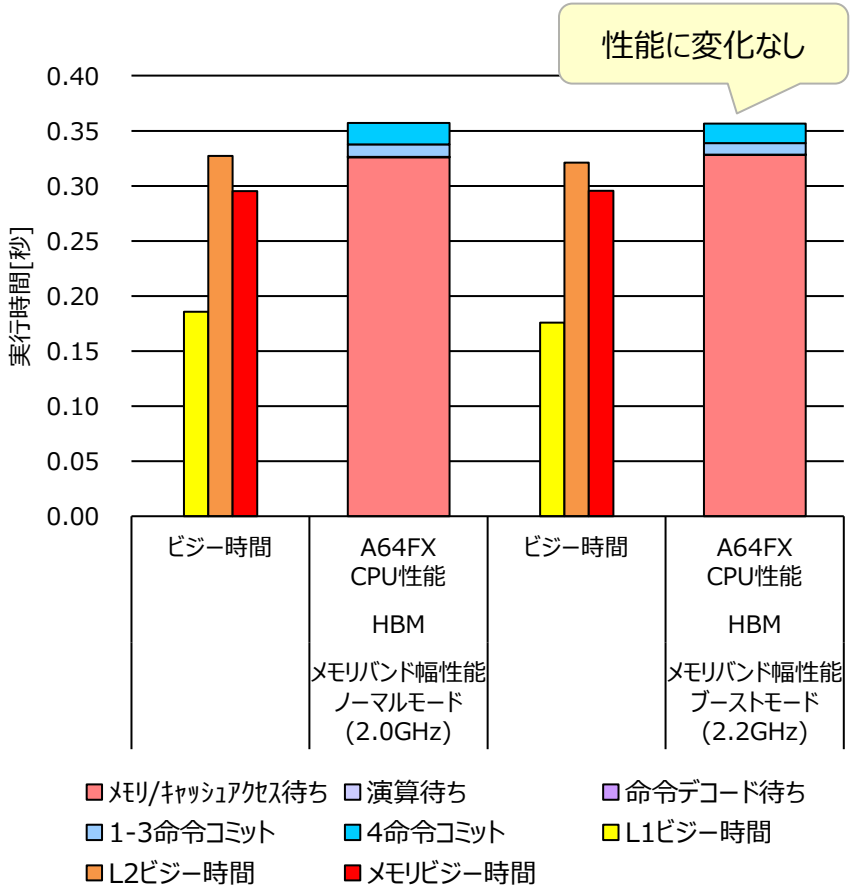
	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)	比率 (2.2GHz ÷2.0GHz)
ソースコード版数	L2バンド幅性能	L2バンド幅性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	12	12	
集計スレッド番号	0	0	
実行時間[s]	0.058	0.052	0.91
有効総命令数	1.35.E+09	1.35.E+09	
GFLOPS(プロセス)	60.63	66.67	1.10
メモリスループット [GB/s/プロセス]	0.00	0.01	
L1ビジー率/スレッド	95.08%	94.97%	
L2ビジー率/スレッド	96.93%	96.82%	
メモリビジー率/スレッド	0.00%	0.00%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	20.73% 10.88%	20.70% 10.86%	
L1Dミス数/スレッド	1.38E+07	1.38E+07	
L1Dミス デマンド率/スレッド	1.00%	1.00%	
L2スループット [GB/s/プロセス]	727.59	799.99	1.10

ブーストモード：基礎カーネル性能 (3/5)

FUJITSU

- メモリバンド幅性能 (zfillあり/1CMG)

→ 性能に変化がないことを確認。
(想定通り)




```
!$omp parallel
   Do j = 1, iter
!$omp do
      Do i = 1, n
         y(i)=x1(i) + c0 * x2(i)
      End Do
!$omp end do nowait
   End Do
!$omp end parallel
```

	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)	比率 (2.2GHz ÷2.0GHz)
ソースコード版数	メモリバンド幅性能	メモリバンド幅性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	12	12	
集計スレッド番号	0	0	
実行時間[s]	0.357	0.356	1.00
有効総命令数	2.52.E+09	2.52.E+09	
GFLOPS(プロセス)	17.60	17.66	1.00
メモリスループット [GB/s/プロセス]	211.50	212.18	1.00
L1ビジー率/スレッド	51.97%	49.32%	
L2ビジー率/スレッド	91.62%	90.04%	
メモリビジー率/スレッド	82.62%	82.88%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	6.46% 2.73%	5.89% 2.48%	
L1Dミス数/スレッド	2.46E+07	2.46E+07	
L1Dミス デマンド率/スレッド	0.07%	0.07%	
L2ミス数/スレッド	1.64E+07	1.64E+07	
L2ミス デマンド率/スレッド	0.29%	0.29%	

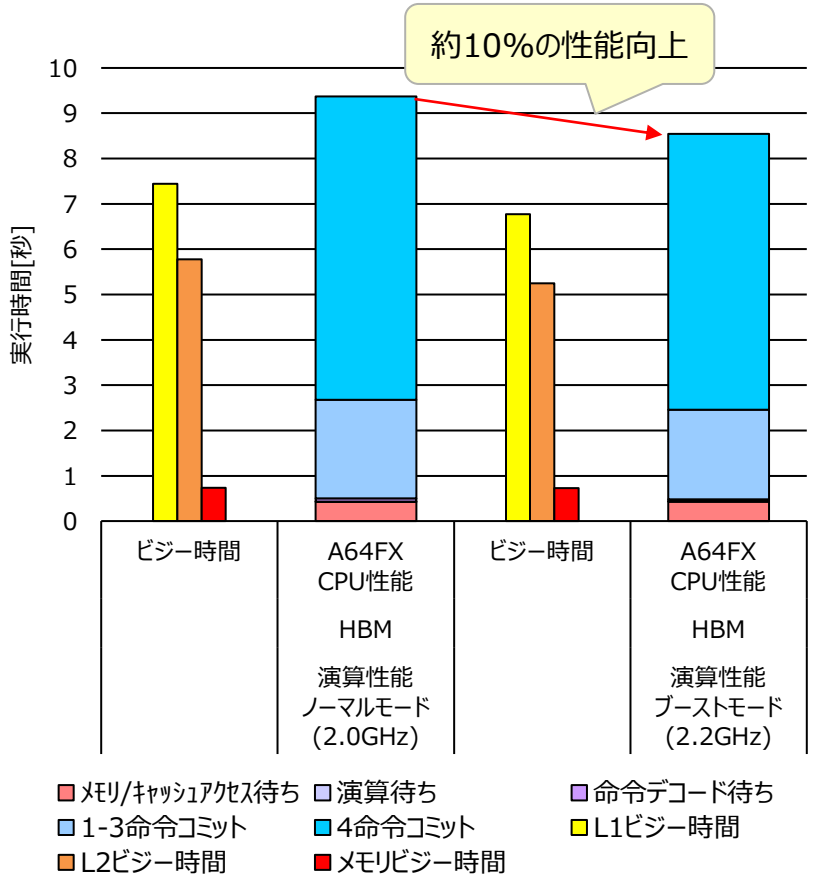
ブーストモード：基礎カーネル性能 (4/5)

FUJITSU

- 演算性能 (DGEMM/1CMG)

→ 約10%の性能向上となることを確認。
(想定通り)

DGEMMのパラメータ				呼び出し回数
TRANSA TRANSB	M	N	K	
NN	23040	23040	640	10



	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)	比率 (2.2GHz ÷2.0GHz)
ソースコード版数	演算性能	演算性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	12	12	
集計スレッド番号	0	0	
実行時間[s]	9.435	8.565	0.91
有効総命令数	7.46.E+11	7.46.E+11	
GFLOPS(プロセス)	720.20	793.31	1.10
メモリスループット [GB/s/プロセス]	20.02	21.91	
L1ビジー率/スレッド	79.40%	79.30%	
L2ビジー率/スレッド	61.62%	61.45%	
メモリビジー率/スレッド	7.82%	8.56%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	94.62% 94.38%	94.51% 94.28%	
L1Dミス数/スレッド	1.80E+09	1.80E+09	
L1Dミス デマンド率/スレッド	0.45%	0.46%	
L2ミス数/スレッド	4.60E+07	4.61E+07	
L2ミス デマンド率/スレッド	52.38%	53.49%	

ブーストモード：基礎カーネル性能 (5/5)

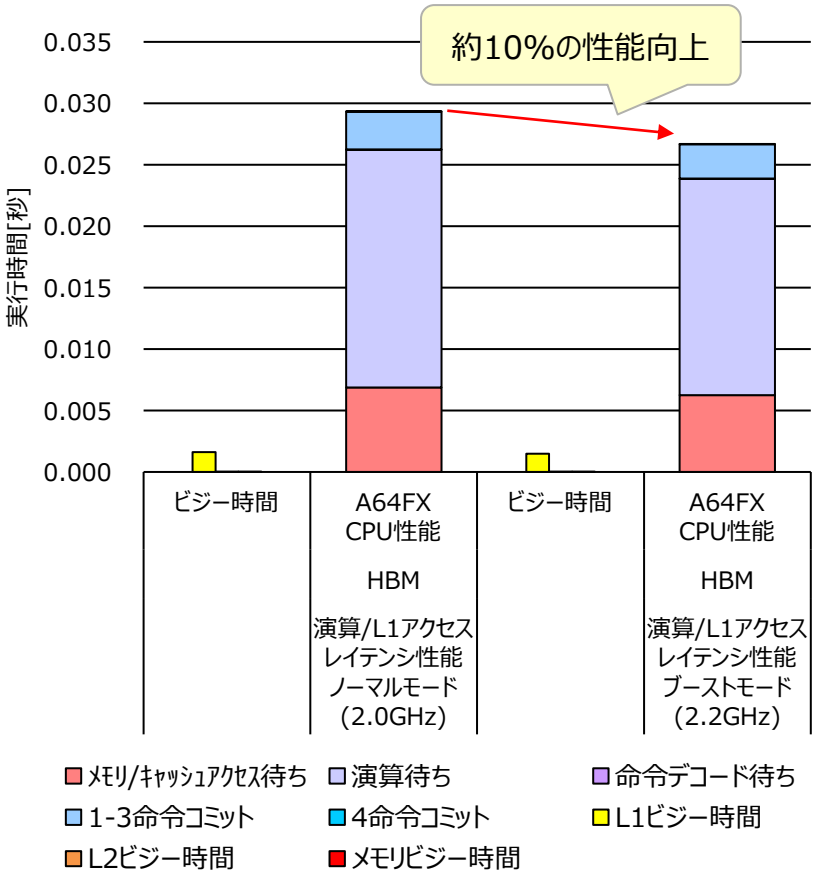
FUJITSU

- 演算/L1アクセスレイテンシ性能

→ 約10%の性能向上となることを確認。
(想定通り)

```
Do j = 1, n
  Do i = 1, 8
    y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m)* ¥
             (c2 + y(i,j-m)*(c3 + y(i,j-m)* ¥
             (c4 + y(i,j-m)*(c5 + y(i,j-m)* ¥
             (c6 + y(i,j-m)*(c7 + y(i,j-m)* ¥
             (c8 + y(i,j-m)* c9))))))))
  End Do
End Do
```

m=1

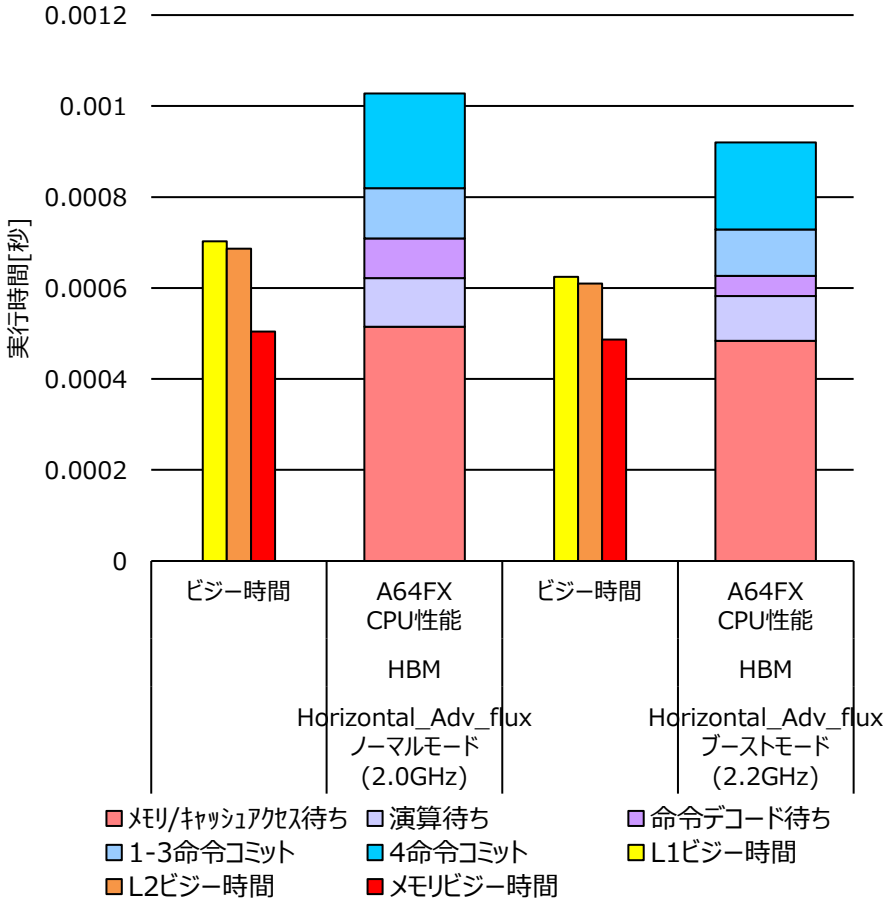


	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)	比率 (2.2GHz ÷2.0GHz)
ソースコード版数	演算/L1アクセスレイテンシ性能	演算/L1アクセスレイテンシ性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	1	1	
集計スレッド番号	0	0	
実行時間[s]	0.029	0.027	0.91
有効総命令数	7.72.E+06	7.74.E+06	
GFLOPS(プロセス)	2.52	2.76	1.10
メモリスループット [GB/s/プロセス]	0.00	0.00	
L1ビジー率/スレッド	5.55%	5.54%	
L2ビジー率/スレッド	0.01%	0.01%	
メモリビジー率/スレッド	0.00%	0.00%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	4.39% 4.36%	4.39% 4.36%	

【参考】アプリカーネル測定結果 (1/2)

FUJITSU

NICAM Horizontal_Adv_flux



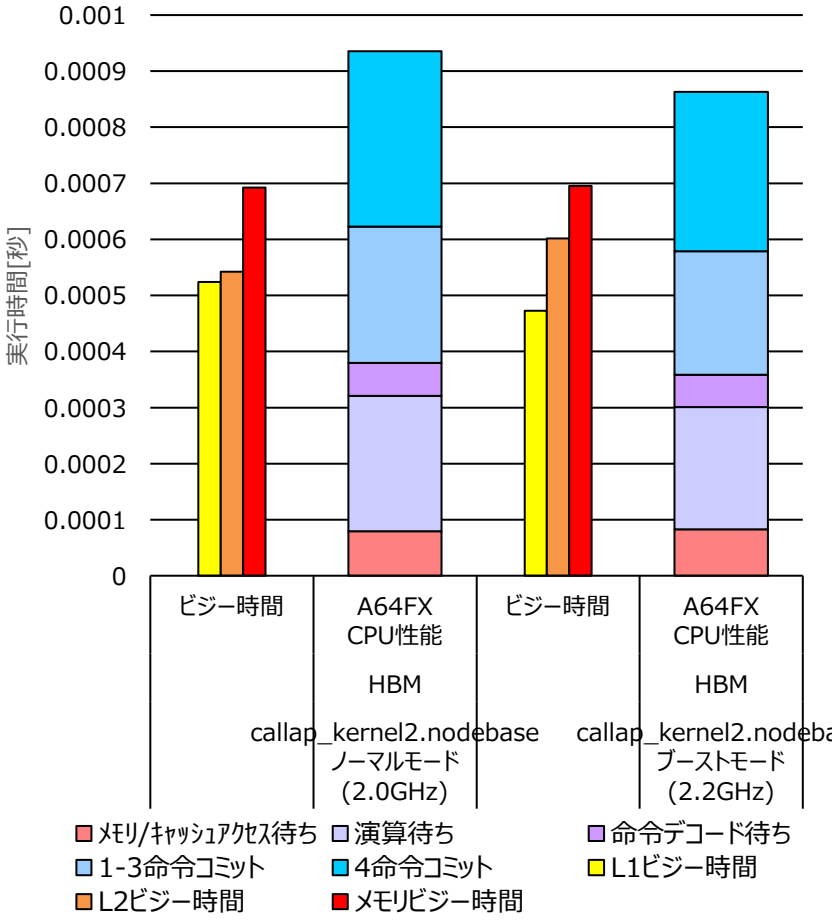
※ 50～80μ程度のPA計測オーバーヘッドあり

	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)
ソースコード版数	Horizontal_Adv_flux	Horizontal_Adv_flux
浮動小数点精度	単精度	単精度
SIMD幅	16	16
スレッド数	12	12
集計スレッド番号	0	0
実行時間[s]	0.001005	0.000929
有効総命令数	2.43.E+07	2.43.E+07
GFLOPS(プロセス)	108.75	117.62
メモリスループット [GB/s/プロセス]	123.04	133.74
L1ビジー率/スレッド	68.40%	67.87%
L2ビジー率/スレッド	66.77%	66.28%
メモリビジー率/スレッド	49.04%	52.90%
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	18.06% 14.00%	17.86% 13.85%
L1Dミス数/スレッド	7.99E+04	8.04E+04
L1Dミス デマンド率/スレッド	20.81%	20.78%
L2ミス数/スレッド	2.18E+04	2.61E+04
L2ミス デマンド率/スレッド	6.78%	7.92%

【参考】アプリカーネル測定結果 (2/2)

FUJITSU

FFB callap_kernel2.nodebase



	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)
ソースコード版数	callap_kernel2. nodebase	callap_kernel2. nodebase
浮動小数点精度	倍精度	倍精度
SIMD幅	8	8
スレッド数	12	12
集計スレッド番号	0	0
実行時間[s]	0.000939	0.000858
有効総命令数	4.13.E+07	4.13.E+07
GFLOPS(プロセス)	282.65	309.34
メモリスループット [GB/s/プロセス]	185.36	203.37
L1ビジー率/スレッド	56.05%	54.81%
L2ビジー率/スレッド	57.98%	69.72%
メモリビジー率/スレッド	74.01%	80.58%
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	76.90% 61.84%	74.75% 60.10%
L1Dミス数/スレッド	5.09E+04	5.09E+04
L1Dミス デマンド率/スレッド	2.15%	2.13%
L2ミス数/スレッド	5.05E+04	5.05E+04
L2ミス デマンド率/スレッド	0.89%	0.87%

※ 50～80μ程度のPA計測オーバーヘッドあり

FUJITSU

エコモードでの性能検証

- エコモード：コード特性毎の性能検証条件
- エコモード：基礎カーネル性能
- 【参考】アプリカーネルのエコモード性能影響
- 【参考】アプリカーネル測定結果
- 【参考】実アプリ(NICAM)の時系列電力と性能

エコモード：コード特性毎の性能検証条件

FUJITSU

■ 測定条件

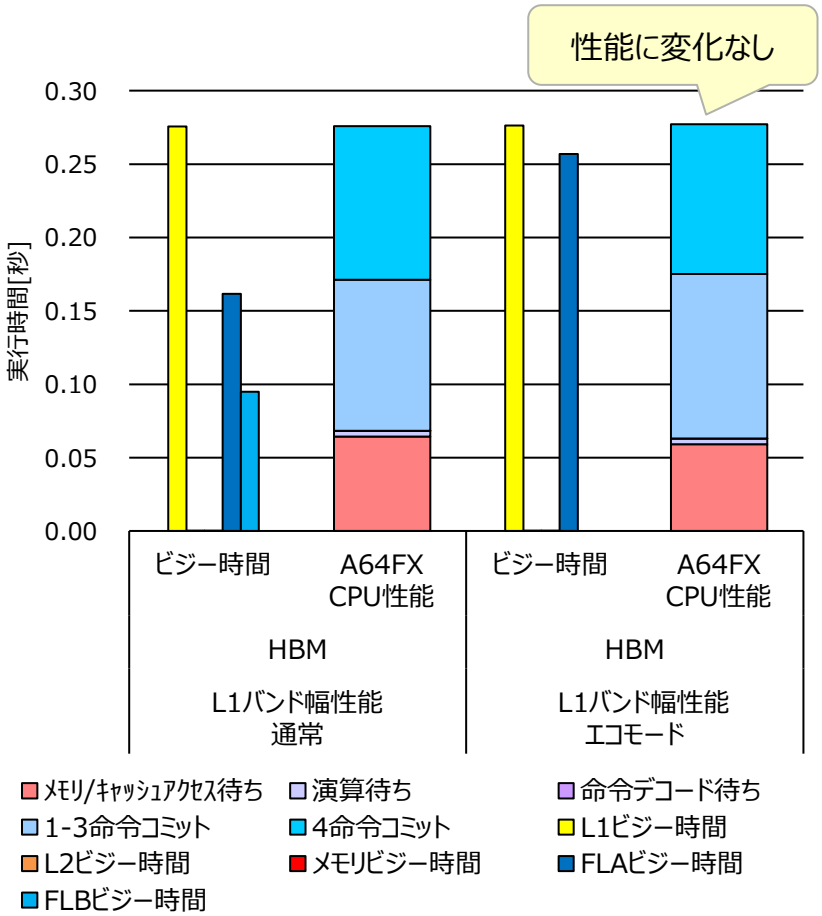
		パターン
検証コード	基礎演算カーネル	①Triad L1キャッシュアクセス (L1バンド幅性能) ②Triad L2キャッシュアクセス (L2バンド幅性能) ③Triad メモリアクセス (メモリバンド幅性能) ④DGEMM (演算性能) ⑤L1アクセスレイテンシコード (演算/L1アクセスレイテンシ性能)
	アプリカーネル	28カーネル
測定コア数	基礎演算カーネル	①⑤：1コア実行、 ②③④：12コア実行(CMG0)
	アプリカーネル	12コア実行(CMG0)
翻訳オプション	基礎演算カーネル	①⑤ -Kfast ② -Kfast,openmp -Kprefetch_sequential=soft ¥ -Kprefetch_cache_level=1 -Kprefetch_line=4 ③ -Kfast,openmp -Kzfill=18 ¥ -Kprefetch_sequential=soft -Kprefetch_line=9 -Kprefetch_line_L2=70 ④ -Kfast,openmp
	アプリカーネル	各カーネル毎で個別に決められたオプションをそのまま使用
アクセス範囲	基礎演算カーネル	①⑤ L1キャッシュサイズの半分(32KB) ② L2キャッシュサイズの半分(4MB) ③ L2キャッシュサイズの30倍(240MB) ④ TRANSA=N, TRANSB=N, M=23040, N=23040, K=640

エコモード：基礎カーネル性能(1/5)

FUJITSU

- L1バンド幅性能（１コア）

→ 性能に影響がないことを確認。
(想定通り)




```
Do j = 1, iter
      Do i = 1, n
            y(i)=x1(i) + c0 * x2(i)
      End Do
End Do
```

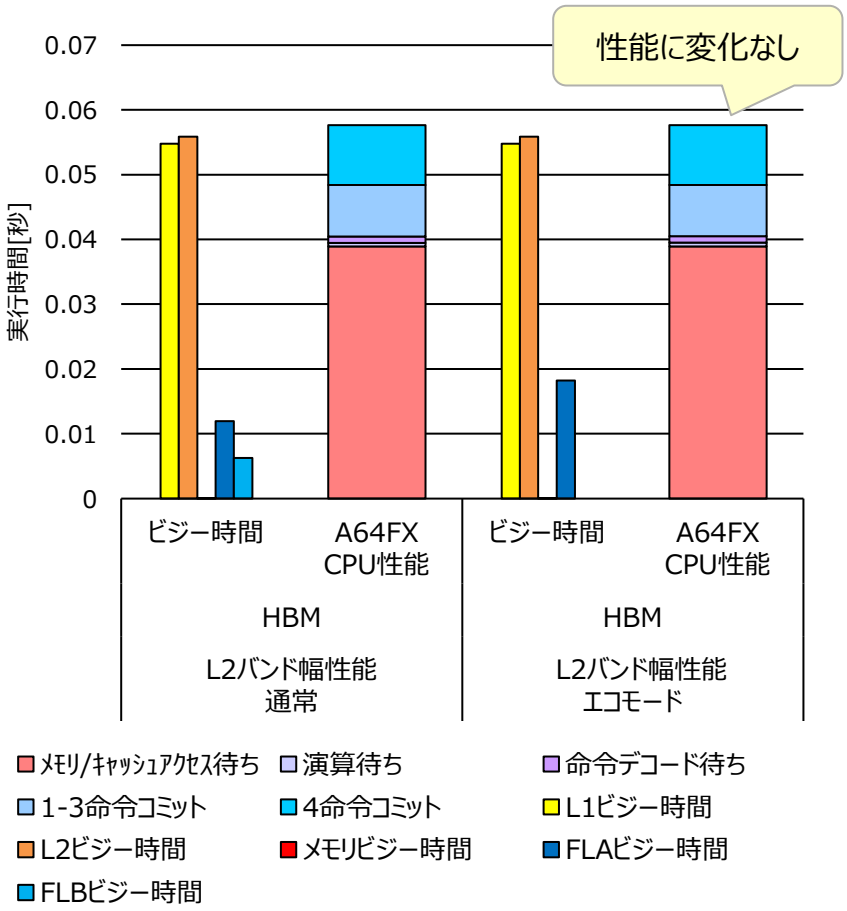
	A64FX CPU 性能 通常	A64FX CPU 性能 エコモード	比率 (エコモード ÷通常)
ソースコード版数	L1バンド幅性能	L1バンド幅性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	1	1	
集計スレッド番号	0	0	
実行時間[s]	0.276	0.277	1.00
有効総命令数	1.29.E+09	1.29.E+09	
GFLOPS(プロセス)	14.85	14.78	1.00
メモリスループット [GB/s/プロセス]	0.00	0.00	
L1ビジー率/スレッド	99.95%	99.75%	
L2ビジー率/スレッド	0.00%	0.00%	
メモリビジー率/スレッド	0.00%	0.00%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	58.61% 34.38%	92.74% 0.00%	
L1スループット [GB/s/スレッド]	178.1	177.4	1.00

エコモード：基礎カーネル性能(2/5)

FUJITSU

- L2バンド幅性能（１ＣＭＧ）

→ 性能に影響がないことを確認。
(想定通り)




```
!$omp parallel
   Do j = 1, iter
!$omp do
      Do i = 1, n
         y(i)=x1(i) + c0 * x2(i)
      End Do
!$omp end do nowait
   End Do
!$omp end parallel
```

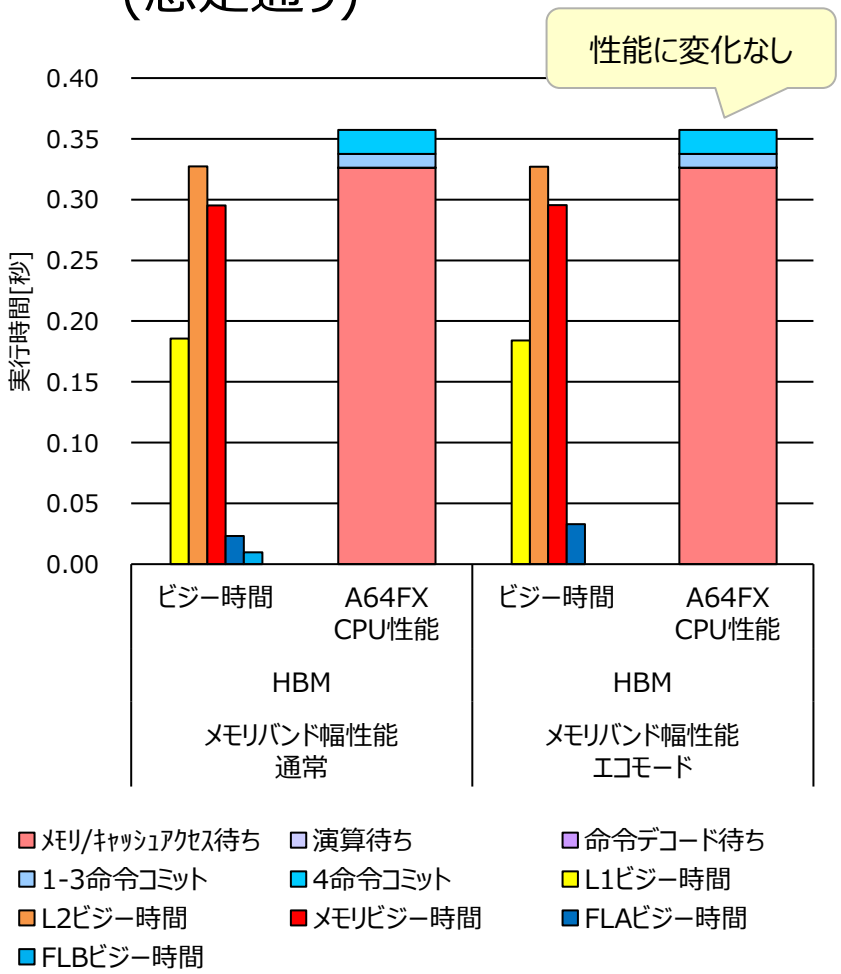
	A64FX CPU 性能 通常	A64FX CPU 性能 エコモード	比率 (エコモード÷通常)
ソースコード版数	L2バンド幅性能	L2バンド幅性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	12	12	
集計スレッド番号	0	0	
実行時間[s]	0.058	0.058	1.00
有効総命令数	1.35.E+09	1.35.E+09	
GFLOPS(プロセス)	60.63	60.68	1.00
メモリスループット [GB/s/プロセス]	0.00	0.00	
L1ビジー率/スレッド	95.08%	95.00%	
L2ビジー率/スレッド	96.93%	96.87%	
メモリビジー率/スレッド	0.00%	0.00%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	20.73% 10.88%	31.58% 0.00%	
L1Dミス数/スレッド	1.38E+07	1.38E+07	
L1Dミス デマンド率/スレッド	1.00%	1.01%	
L2スループット [GB/s/プロセス]	727.59	728.15	1.00

エコモード：基礎カーネル性能(3/5)

FUJITSU

- メモリバンド幅性能 (zfillあり/1CMG)

→ 性能に変化がないことを確認。
(想定通り)




```
!$omp parallel
   Do j = 1, iter
!$omp do
      Do i = 1, n
         y(i)=x1(i) + c0 * x2(i)
      End Do
!$omp end do nowait
   End Do
!$omp end parallel
```

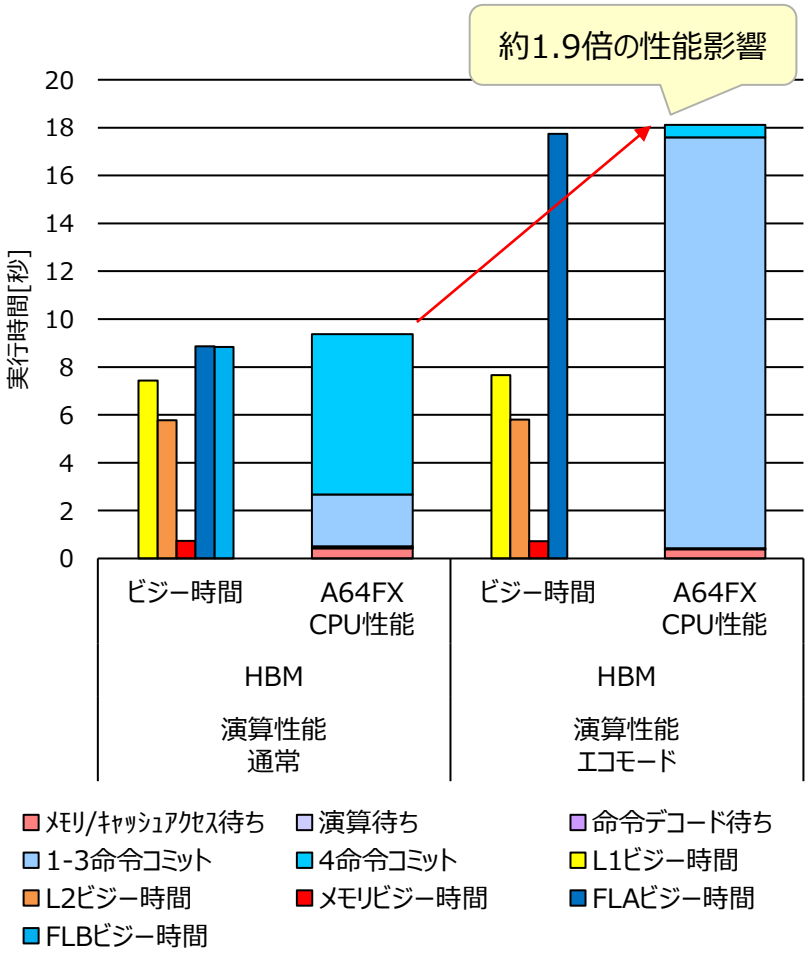
	A64FX CPU 性能 通常	A64FX CPU 性能 エコモード	比率 (エコモード÷通常)
ソースコード版数	メモリバンド幅性能	メモリバンド幅性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	12	12	
集計スレッド番号	0	0	
実行時間[s]	0.357	0.357	1.00
有効総命令数	2.52.E+09	2.52.E+09	
GFLOPS(プロセス)	17.60	17.62	1.00
メモリスループット [GB/s/プロセス]	211.50	211.78	1.00
L1ビジー率/スレッド	51.97%	51.50%	
L2ビジー率/スレッド	91.62%	91.55%	
メモリビジー率/スレッド	82.62%	82.73%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	6.46% 2.73%	9.18% 0.00%	
L1Dミス数/スレッド	2.46E+07	2.46E+07	
L1Dミス デマンド率/スレッド	0.07%	0.07%	
L2ミス数/スレッド	1.64E+07	1.64E+07	
L2ミス デマンド率/スレッド	0.29%	0.29%	

エコモード：基礎カーネル性能(4/5)

FUJITSU

- 演算性能(DGEMM/1CMG)

→ 約1.9倍の性能影響を確認。



DGEMMのパラメータ				呼び出し回数
TRANSA TRANSB	M	N	K	
NN	23040	23040	640	10

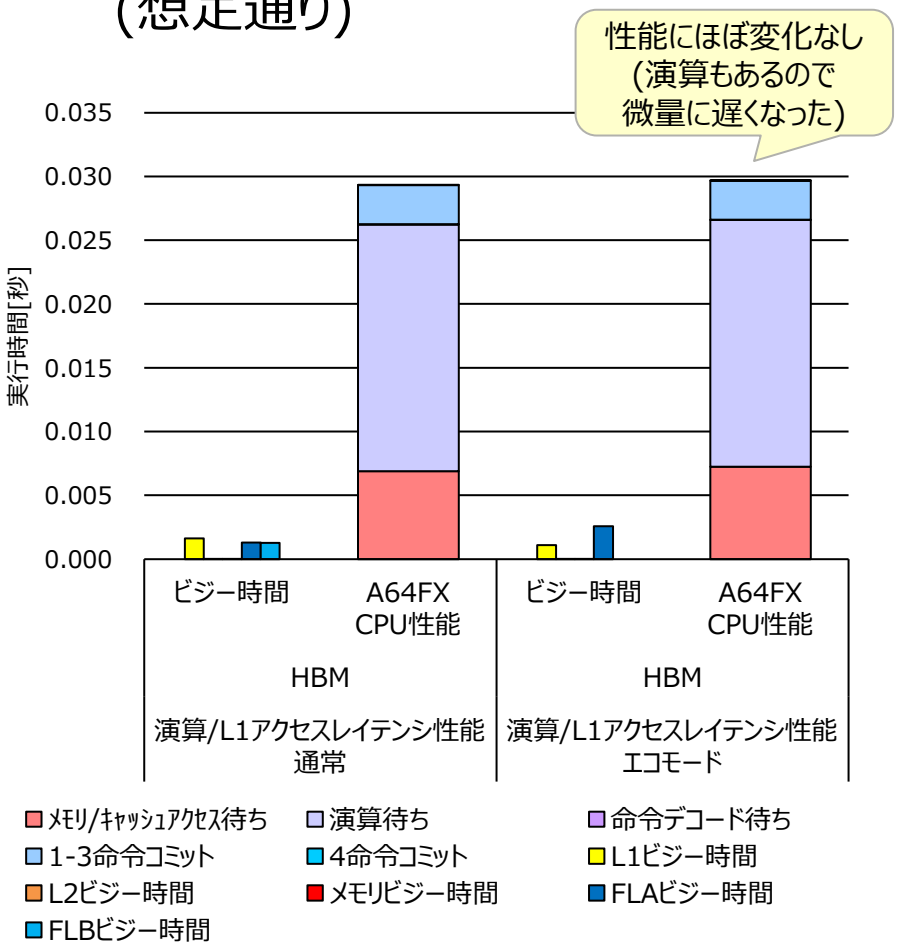
	A64FX CPU 性能 通常	A64FX CPU 性能 エコモード	比率 (エコモード÷通常)
ソースコード版数	演算性能	演算性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	12	12	
集計スレッド番号	0	0	
実行時間[s]	9.435	18.093	1.92
有効総命令数	7.46.E+11	7.46.E+11	
GFLOPS(プロセス)	720.20	375.56	0.52
メモリスループット [GB/s/プロセス]	20.02	10.25	0.51
L1ビジー率/スレッド	79.40%	42.33%	
L2ビジー率/スレッド	61.62%	32.06%	
メモリビジー率/スレッド	7.82%	4.00%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	94.62% 94.38%	97.99% 0.00%	
L1Dミス数/スレッド	1.80E+09	1.80E+09	
L1Dミス デマンド率/スレッド	0.45%	0.45%	
L2ミス数/スレッド	4.60E+07	4.57E+07	
L2ミス デマンド率/スレッド	52.38%	52.32%	

エコモード：基礎カーネル性能(5/5)

FUJITSU

- 演算/L1アクセスレイテンシ性能

→ 性能に影響がないことを確認。
(想定通り)

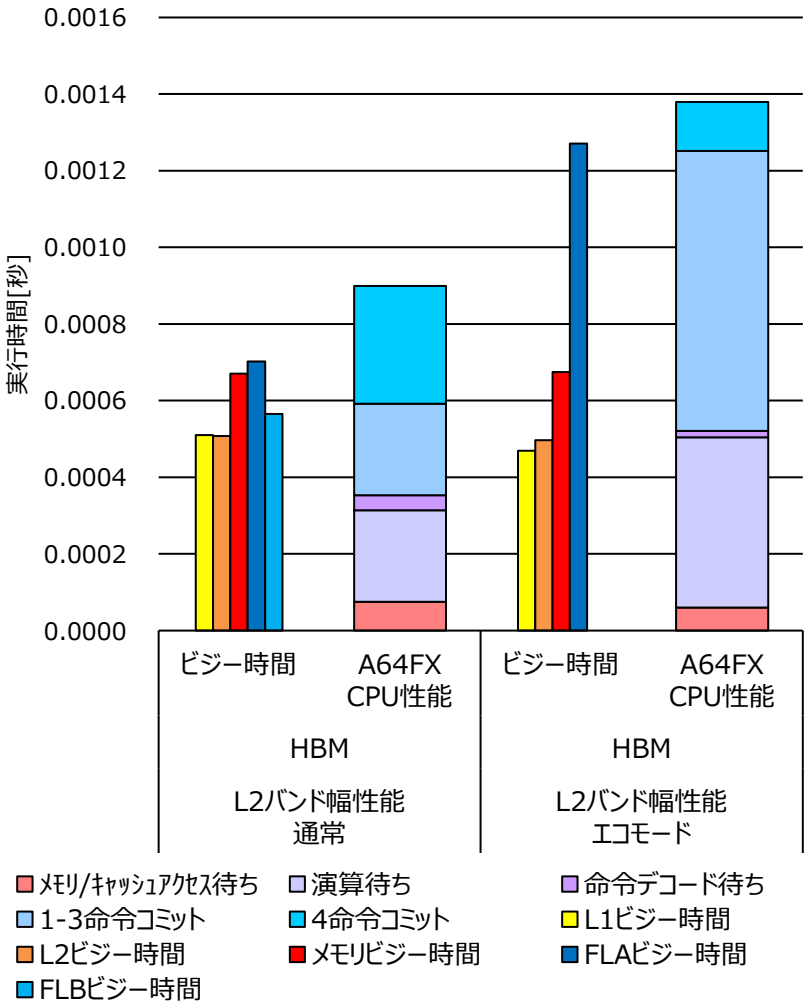



```
Do j = 1, n                                                    m=1
  Do i = 1, 8
    y(i,j) = c0 + y(i,j-m)*(c1 + y(i,j-m)* ¥
             (c2 + y(i,j-m)*(c3 + y(i,j-m)* ¥
             (c4 + y(i,j-m)*(c5 + y(i,j-m)* ¥
             (c6 + y(i,j-m)*(c7 + y(i,j-m)* ¥
             (c8 + y(i,j-m)* c9)))))))
  End Do
End Do
```

	A64FX CPU 性能 通常	A64FX CPU 性能 エコモード	比率 (エコモード ÷通常)
ソースコード版数	演算/L1アクセスレイテンシ性能	演算/L1アクセスレイテンシ性能	
浮動小数点精度	倍精度	倍精度	
SIMD幅	8	8	
スレッド数	1	1	
集計スレッド番号	0	0	
実行時間[s]	0.029	0.030	1.01
有効総命令数	7.72.E+06	7.74.E+06	
GFLOPS(プロセス)	2.52	2.48	0.99
メモリスループット [GB/s/プロセス]	0.00	0.00	
L1ビジー率/スレッド	5.55%	3.72%	
L2ビジー率/スレッド	0.01%	0.01%	
メモリビジー率/スレッド	0.00%	0.00%	
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	4.39% 4.36%	8.64% 0.00%	

【参考】アプリカーネルのエコモード性能影響

- FFB callap_kernel2.nodebase
  ノーマルモードとエコモード



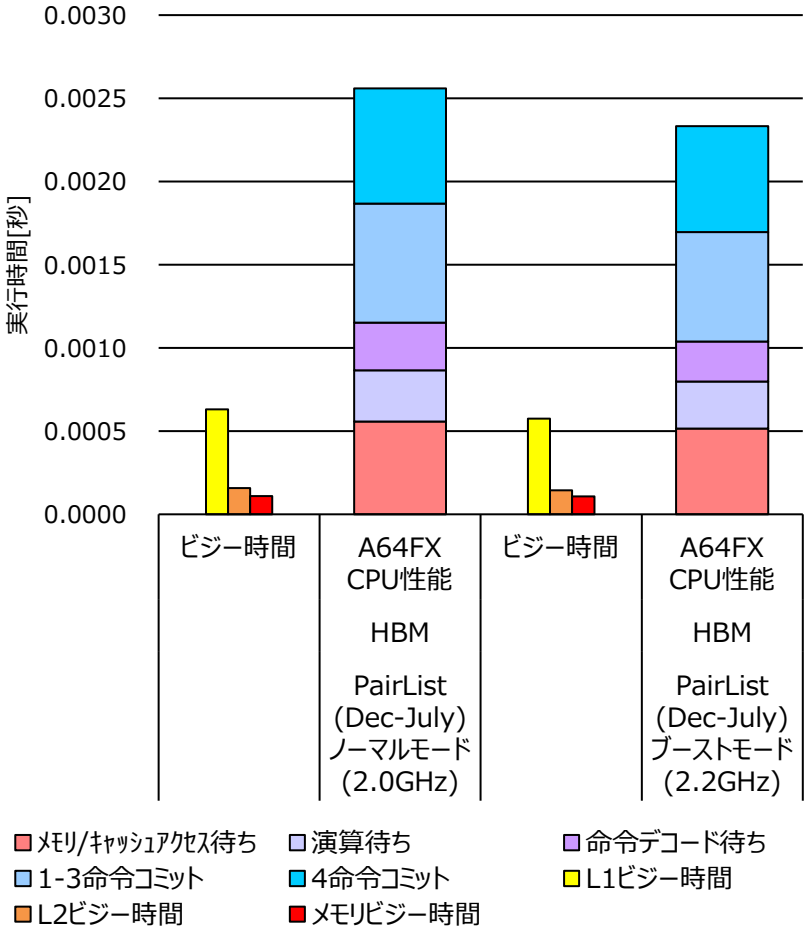
	A64FX CPU 性能 通常	A64FX CPU 性能 エコモード
ソースコード版数	callap_kernel2.nodebase	callap_kernel2.nodebase
浮動小数点精度	単精度	単精度
SIMD幅	8	8
スレッド数	12	12
集計スレッド番号	0	0
実行時間[s]	0.000912	0.001392
有効総命令数	3.45.E+06	3.45.E+06
GFLOPS(プロセス)	291.05	190.80
メモリスループット [GB/s/プロセス]	190.72	125.15
L1ビジー率/スレッド	56.76%	34.02%
L2ビジー率/スレッド	56.42%	35.99%
メモリビジー率/スレッド	74.54%	48.90%
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	78.13% 62.85%	92.14% 0.00%
L1Dミス数/スレッド	5.10E+04	5.10E+04
L1Dミス デマンド率/スレッド	2.23%	2.30%
L2ミス数/スレッド	5.07E+04	5.07E+04
L2ミス デマンド率/スレッド	1.05%	1.01%

※ -Hmethod=normalで測定したPAの値を使用しているため実行時間が多少大きくなっている

【参考】アプリカーネル測定結果

FUJITSU

GENESIS PairList (Dec-July)



	A64FX CPU 性能 ノーマルモード (2.0GHz)	A64FX CPU 性能 ブーストモード (2.2GHz)
ソースコード版数	PairList (Dec-July)	PairList (Dec-July)
浮動小数点精度	倍精度	倍精度
SIMD幅	8	8
スレッド数	12	12
集計スレッド番号	0	0
実行時間[s]	0.002542	0.002347
有効総命令数	1.00.E+08	1.00.E+08
GFLOPS(プロセス)	42.31	45.82
メモリスループット [GB/s/プロセス]	10.84	11.61
L1ビジー率/スレッド	24.60%	24.67%
L2ビジー率/スレッド	6.14%	6.16%
メモリビジー率/スレッド	4.26%	4.56%
浮動小数点パイプライン ビジー率/スレッド (FLA、FLB)	15.29% 13.22%	15.36% 13.29%
L1Dミス数/スレッド	1.37E+04	1.36E+04
L1Dミス デマンド率/スレッド	73.13%	72.85%
L2ミス数/スレッド	6.28E+03	6.28E+03
L2ミス デマンド率/スレッド	32.14%	30.22%

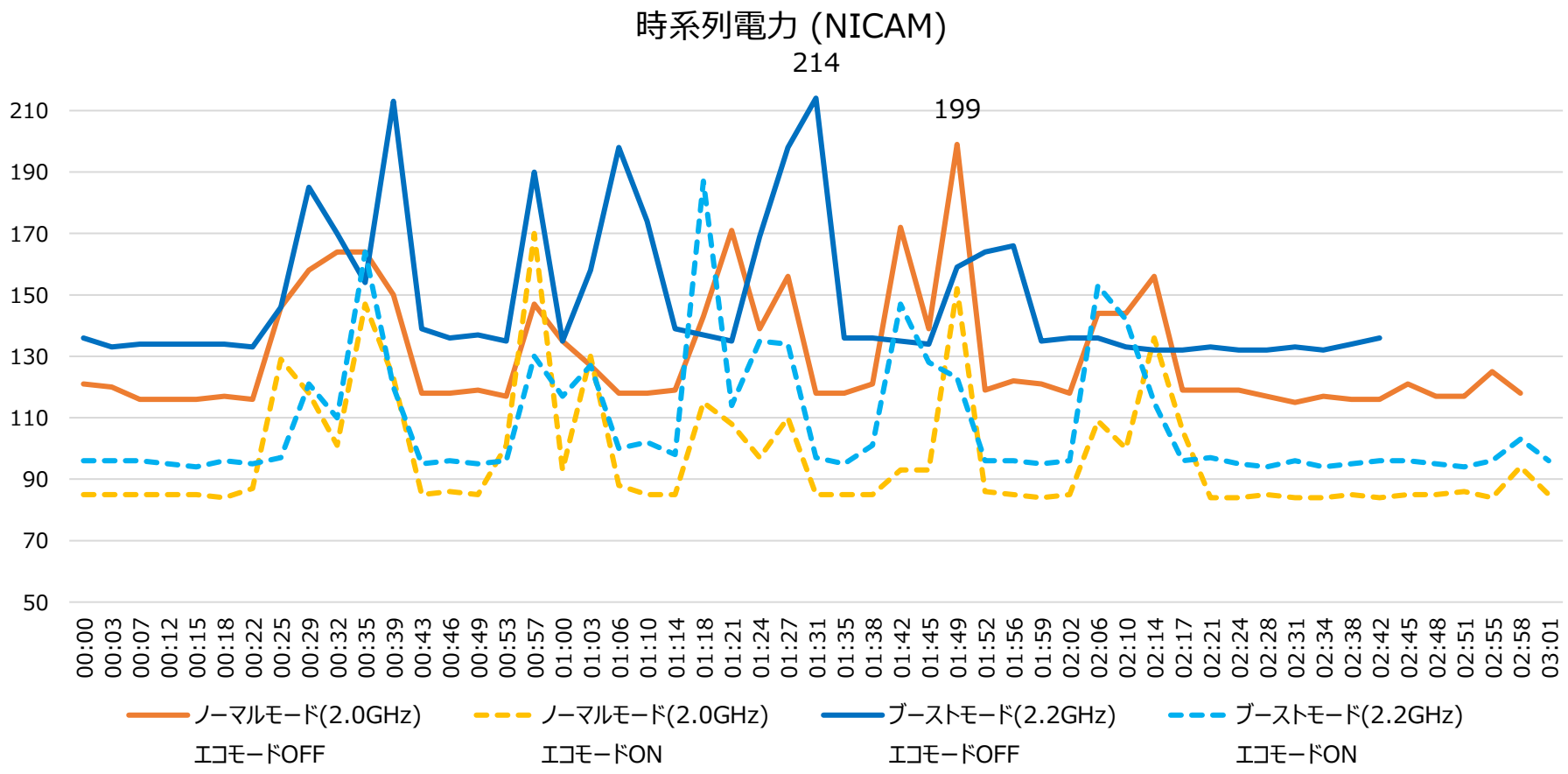
※ 50～80μ程度のPA計測オーバーヘッドあり

【参考】実アプリ(NICAM)の時系列電力と性能

FUJITSU

- 時系列電力集計

電力値測定は約3秒間隔（縦軸：電力値、横軸：経過時間）
NICAMは、2019年7月版を使用



FUJITSU
shaping tomorrow with you

基礎カーネル性能

注意）記載の性能値に関しては、測定ソフトウェア環境（コンパイラやライブラリ）によって多少の凸凹が発生する可能性があります。

評価条件と評価環境

- 評価環境：評価システム
- 評価条件：評価コード
- 評価条件：コンパイラオプション

評価環境：評価システム(1/2)

FUJITSU

		京	FX100	Haswell (Xeon E5-2698 v3)	Skylake (Xeon Platinum 8168)	A64FX 1ノード
周波数【GHz】		2.0	1.975	1.9 ※1	1.9 ※1	2.0
CPU数/ノード		1	1	2	2	1
コア数/ノード		8	32	32	48	48
CMG数/ノード			2			4
メモリ容量/ノード【GB】		16	32	256	384	32
キャッシュ容量	L1【KiB/コア】	32	64	32	32	64
	L2【KiB/コア】			256	1024	
	LL【MiB/CPU】	6	12	40	33	8
キャッシュレイテンシ	L1【cycle】	4	5	4	4	5 (EX, short) 8 (FL, short) 11 (FL, long)
	L2【cycle】			11	12	
	LL【cycle】	31	54	～34	45	37～47
キャッシュスループット	L1【B/cycle】	32	64	96	-	ヒット時：128
	L2【B/cycle】			64		
	LL【B/cycle】	16	32		-	42.7

※1 ： 1.9GHz 固定で動作するように設定

評価環境：評価システム(2/2)

FUJITSU

		京	FX100	Haswell (Xeon E5-2698 v3)	Skylake (Xeon Platinum 8168)	A64FX 1ノード
ノードあたり演算性能 (1コアあたり演算性能) 【GFlops】	倍精度	128 (16)	1011.2 (31.6)	972.8 (30.4)	2918.4 (60.8)	3,072.0 (64.0)
	単精度		2022.4 (63.2)	1945.6 (60.8)	5836.8 (121.6)	6144.0 (128.0)
主記憶レイテンシ 【ns】		86	160		80	150
ノードあたり理論メモリバンド幅 (CMGあたり理論メモリバンド幅) 【GB/秒】		64	480 (240)	136	255	1024 (256)

評価条件：評価コード

評価条件	パターン
評価コード	① on L1$ - 四則演算・平方根 - 数学関数 - 数値関数 - 型変換 - アクセス性能(連続アクセス) L1$アクセス - アクセス性能(ストライドアクセス) - アクセス性能(間接アクセス) ②アクセス性能(連続アクセス) L2$アクセス ③アクセス性能(連続アクセス) メモリアクセス
評価コア数	① 1コア実行 ②③ 12コア実行(1CMG)
アクセス範囲	① L1キャッシュサイズの3/4(48KB) ② L2キャッシュサイズの半分(4MB) ③ L2キャッシュサイズの3倍(24MB)

評価条件：コンパイラオプション

評価環境	評価オプション
京 (2.0GHz)	-Kfast -V -Nlst=t -Koptmsg=2 [Fortranのみ] -Cpp -Kautoobjstack,temparraystack [ilfunc評価のみ] -Kilfunc,nomfunc
FX100 (1.975GHz)	-Kfast -V -Nlst=t -Koptmsg=2 [Fortranのみ] -Cpp -Kautoobjstack,temparraystack [ilfunc評価のみ] -Kilfunc,nomfunc
PRIMERGY RX2530 M1 Haswell（FJコンパイラ）(1.9GHz)	-Kfast,CORE_AVX2 -Nlst=t -Koptmsg=2 [Fortranのみ] -Cpp -Kautoobjstack,temparraystack
PRIMERGY RX2540 M4 Skylake（Intelコンパイラ）(1.9GHz)	-O3 -no-prec-div -fp-model fast=2 -xCORE-AVX512 -qopt-zmm-usage=high
A64FX (2.0GHz)	-Kfast -V -Nlst=t -Koptmsg=2 [Fortranのみ] -Cpp -Kautoobjstack,temparraystack

FUJITSU

基礎演算カーネル性能

- 四則演算・平方根
- 数学関数

四則演算・平方根 (1/2)

FUJITSU

■ 四則演算・平方根(実数型)の測定結果

		京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX			
		浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	演算効率 (%)	SIMD化効果
倍精度	加算	1.38	1.00	3.37	2.48	4.61	3.52	8.57	6.54	7.42	5.38	11.59	6.55
	減算	1.38	1.00	3.40	2.49	4.62	3.52	8.46	6.46	7.42	5.38	11.59	6.55
	乗算	1.38	1.00	3.40	2.50	4.62	3.53	7.51	5.73	7.42	5.38	11.59	6.70
	積和演算	2.87	1.00	6.96	2.46	9.51	3.49	15.79	5.80	14.84	5.18	23.19	6.55
	除算	10.39	1.00	19.06	1.86	3.22	0.33	9.24	0.94	39.57	3.81	61.84	7.94
	逆数	10.79	1.00	19.02	1.79	5.60	0.55	8.38	0.82	39.85	3.69	62.26	8.33
	平方根	11.90	1.00	21.63	1.84	4.84	0.43	8.14	0.72	34.78	2.92	54.34	12.75
単精度	加算	1.69	1.00	6.57	3.93	9.11	5.66	13.59	8.45	13.84	8.18	10.81	12.16
	減算	1.68	1.00	6.62	3.99	9.11	5.71	13.78	8.64	13.84	8.24	10.81	12.16
	乗算	1.69	1.00	6.62	3.96	9.09	5.65	13.49	8.39	13.84	8.18	10.81	12.16
	積和演算	3.46	1.00	13.52	3.96	19.18	5.84	26.60	8.09	27.68	8.00	21.62	12.20
	除算	9.85	1.00	15.97	1.64	26.98	2.88	39.32	4.20	61.13	6.21	47.76	14.42
	逆数	9.99	1.00	16.84	1.71	28.13	2.97	44.43	4.68	72.00	7.21	56.25	18.64
	平方根	9.61	1.00	17.15	1.81	8.41	0.92	49.29	5.40	52.07	5.42	40.68	23.83

四則演算・平方根 (2/2)

FUJITSU

■ 四則演算(整数型)の測定結果

		京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX			
		整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	演算効率 (%)	SIMD化効果
8バイト整数	加算	0.77	1.00	3.43	4.51	4.74	6.47	8.86	12.09	7.42	9.62	11.59	7.91
	減算	0.77	1.00	3.43	4.51	4.74	6.48	8.72	11.90	7.42	9.62	11.59	7.91
	乗算	0.33	1.00	3.47	10.64	1.21	3.87	5.86	18.66	7.42	22.44	11.59	7.84
	積和演算	0.66	1.00	7.01	10.69	2.61	4.13	9.21	14.61	14.84	22.36	23.19	13.91
	除算	0.18	1.00	0.18	1.00	0.07	0.42	0.31	1.78	0.09	0.50	0.14	0.58
4バイト整数	加算	0.84	1.00	3.88	4.71	9.41	11.86	15.53	19.56	13.84	16.56	10.81	14.54
	減算	0.84	1.00	3.88	4.71	9.42	11.86	15.21	19.16	13.84	16.56	10.81	14.69
	乗算	0.33	1.00	3.80	11.60	7.28	23.07	11.90	37.72	13.84	41.54	10.77	14.62
	積和演算	0.66	1.00	7.64	11.65	14.27	22.61	25.98	41.17	27.68	41.67	21.62	26.08
	除算	0.18	1.00	0.18	1.00	0.23	1.35	1.57	9.11	0.28	1.54	0.22	1.82

数学関数 (1/2)

FUJITSU

■ 数学関数の測定結果

性能向上比は
京との比率
(クロックノーマライズ)

		京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX		
		浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	SIMD化効果
倍精度	atan	8.71	1.00	13.25	1.54	4.36	0.53	18.80	2.27	15.41	1.77	10.85
	atan2	8.87	1.00	9.69	1.11	2.32	0.28	16.19	1.92	6.88	0.78	15.46
	cos	11.55	1.00	19.43	1.70	4.49	0.41	21.10	1.92	24.44	2.11	8.04
	exp	7.93	1.00	15.22	1.94	4.05	0.54	22.55	2.99	18.04	2.27	10.70
	exp10	7.99	1.00	15.18	1.92	1.56	0.21	22.19	2.92	17.25	2.16	10.29
	log	6.97	1.00	7.87	1.14	3.14	0.47	16.01	2.42	11.80	1.69	8.09
	log10	7.62	1.00	9.45	1.26	2.36	0.33	20.70	2.86	11.84	1.55	8.82
	sin	11.54	1.00	19.47	1.71	5.42	0.49	22.21	2.03	24.17	2.10	7.98
	べき乗	7.48	1.00	8.88	1.20	2.66	0.37	13.32	1.88	8.47	1.13	14.97
単精度	atan	7.29	1.00	9.76	1.36	1.72	0.25	29.51	4.26	36.30	4.98	22.38
	atan2	8.00	1.00	6.76	0.86	1.28	0.17	28.70	3.78	25.67	3.21	22.10
	cos	10.98	1.00	16.37	1.51	1.30	0.13	39.92	3.83	48.93	4.46	15.88
	exp	6.74	1.00	11.44	1.72	1.55	0.24	47.42	7.41	35.75	5.30	21.34
	exp10	7.37	1.00	11.65	1.60	1.52	0.22	43.10	6.15	37.59	5.10	23.35
	log	5.91	1.00	5.49	0.94	0.99	0.18	41.46	7.39	28.96	4.90	17.47
	log10	6.29	1.00	6.67	1.07	1.11	0.19	52.90	8.86	30.70	4.88	17.73
	sin	10.98	1.00	16.36	1.52	1.36	0.13	45.36	4.35	48.98	4.46	17.77
	べき乗	7.30	1.00	6.55	0.91	1.02	0.15	25.56	3.69	17.30	2.37	17.95

数学関数 (2/2)

FUJITSU

■ 他CPU比較

		京 (GFLOPS)	FX100 (GFLOPS)	Skylake (GFLOPS)	A64FX (GFLOPS)	京 比	FX100 比	Skylake 比
倍精度	atan	8.71	13.25	18.80	15.41	1.77	1.16	0.78
	atan2	8.87	9.69	16.19	6.88	0.78	0.71	0.40
	cos	11.55	19.43	21.10	24.44	2.11	1.26	1.10
	exp	7.93	15.22	22.55	18.04	2.27	1.19	0.76
	exp10	7.99	15.18	22.19	17.25	2.16	1.14	0.74
	log	6.97	7.87	16.01	11.80	1.69	1.50	0.70
	log10	7.62	9.45	20.70	11.84	1.55	1.25	0.54
	sin	11.54	19.47	22.21	24.17	2.10	1.24	1.03
	べき乗	7.48	8.88	13.32	8.47	1.13	0.95	0.60
単精度	atan	7.29	9.76	29.51	36.30	4.98	3.72	1.17
	atan2	8.00	6.76	28.70	25.67	3.21	3.80	0.85
	cos	10.98	16.37	39.92	48.93	4.46	2.99	1.16
	exp	6.74	11.44	47.42	35.75	5.30	3.13	0.72
	exp10	7.37	11.65	43.10	37.59	5.10	3.23	0.83
	log	5.91	5.49	41.46	28.96	4.90	5.28	0.66
	log10	6.29	6.67	52.90	30.70	4.88	4.60	0.55
	sin	10.98	16.36	45.36	48.98	4.46	2.99	1.03
	べき乗	7.30	6.55	25.56	17.30	2.37	2.64	0.64
GEOMEAN						2.66	2.00	0.76

改善余地あり 2020/09時点

FUJITSU

その他基礎演算カーネル評価

- 数値関数
- 型変換

数値関数

■ 数値関数の測定結果

		京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX		
		浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	SIMD化効果
倍精度	abs	1.71	1.00	4.82	2.85	6.93	4.26	13.38	8.23	9.78	5.72	7.08
	max	1.42	1.00	3.43	2.45	4.54	3.38	8.08	6.00	7.42	5.24	6.55
	min	1.42	1.00	3.43	2.45	4.57	3.39	8.21	6.10	7.42	5.24	6.55
	mod	1.27	1.00	9.29	7.40	5.37	4.44	18.70	15.47	6.04	4.76	9.38
	sign	1.58	1.00	3.45	3.32	2.46	2.81	7.07	8.07	5.85	6.35	6.37
単精度	abs	2.02	1.00	5.70	2.86	14.01	7.30	20.59	10.72	17.16	8.49	10.73
	max	1.73	1.00	3.88	2.27	9.00	5.47	12.14	7.39	13.84	7.99	12.16
	min	1.74	1.00	3.83	2.23	9.06	5.48	12.09	7.31	13.84	7.95	12.16
	mod	2.06	1.00	9.63	4.73	13.79	7.05	27.86	14.24	11.11	5.39	18.67
	sign	1.58	1.00	3.90	2.49	4.99	3.32	13.44	8.94	11.64	7.36	12.81

型変換

■ 型変換の測定結果

	京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX		
	整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	整数演算性能 (GOPS)	性能向上比	SIMD化効果
dble(単精度実数)	1.64	1.00	4.79	2.96	3.79	2.44	9.19	5.91	9.06	5.54	5.78
dble(4バイト整数)	1.29	1.00	4.78	3.76	4.59	3.75	9.74	7.96	9.06	7.04	13.10
real(倍精度実数)	1.91	1.00	5.39	2.86	4.64	2.56	8.70	4.80	11.19	5.86	7.59
real(4バイト整数)	1.66	1.00	5.96	3.64	12.07	7.67	18.23	11.57	17.21	10.38	24.87
int(倍精度実数)	1.11	1.00	5.79	5.26	4.78	4.53	8.78	8.31	11.19	10.06	27.01
int(単精度実数)	1.13	1.00	5.98	5.34	11.99	11.13	17.09	15.86	17.21	15.18	40.48
aint(倍精度実数)	0.83	1.00	4.79	5.82	3.56	4.50	6.37	8.05	9.78	11.75	6.23
aint(単精度実数)	0.87	1.00	5.95	6.92	7.39	8.94	12.60	15.23	17.21	19.77	10.80
nint(倍精度実数)	0.80	1.00	4.07	5.15	0.83	1.10	3.92	5.15	9.78	12.19	26.65
nint(単精度実数)	0.82	1.00	4.25	5.22	2.49	3.17	9.10	11.61	17.26	20.93	47.52
anint(倍精度実数)	0.69	1.00	4.79	7.04	1.24	1.89	5.25	8.03	9.78	14.21	6.13
anint(単精度実数)	0.71	1.00	5.87	8.37	2.47	3.66	9.17	13.58	17.21	24.21	10.80

アクセス性能

- 基本アクセス性能

基本アクセス性能 (1/2)

FUJITSU

アクセス性能(連続アクセス)：STREAM Triadケース

L1アクセス

	京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX			
	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	浮動小数点演算性能 (Gflops)	性能向上比	演算効率 (%)	演算効率(理想値) (%)
連続SIMDロード×2・連続SIMDストア×1	2.87	1.00	6.94	2.45	9.02	3.31	11.69	4.29	14.84	5.17	23.19	25.00

L2アクセス　FX100,A64FXは1CMG 他は1CPU評価

	京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX	
	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比
連続SIMDロード×2・連続SIMDストア×1	133.77	1.00	418.62	3.13	301.74	2.26	312.42	2.34	698.88	5.22

メモリアクセス　FX100,A64FXは1CMG 他は1CPU評価

	京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX		
	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比	スループット (GB/s)	性能向上比	スループット効率 (%)
連続SIMDロード×2・連続SIMDストア×1	34.89	1.00	103.52	2.97	43.30	1.24	77.20	2.21	150.68	4.32	78.48

zfill 系なしの結果
zfill ありでは210GB/s程度

基本アクセス性能 (2/2)

FUJITSU

アクセス性能(ストライドアクセス) L1アクセス

		京		FX100		PRIMERGY RX2530 M1		A64FX			
		浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	演算効率(%)	演算効率(理想値)(%)
ロードなし・間接SIMDストア×1	飛び幅2	0.88	1.00	1.36	1.56	1.87	2.23	1.60	1.82	2.50	
	飛び幅16	0.82	1.00	1.00	1.23	1.75	2.24	1.41	1.71	2.20	
間接SIMDロード×2・間接SIMDストア×1	飛び幅2	1.22	1.00	1.58	1.31	2.23	1.93	1.96	1.61	3.07	
	飛び幅16	0.85	1.00	1.24	1.48	2.25	2.79	0.90	1.06	1.40	
間接SIMDロード×2・連続SIMDストア×1	飛び幅2	1.82	1.00	2.41	1.34	2.27	1.32	4.72	2.60	7.38	10.00
	飛び幅16	0.91	1.00	2.25	2.49	2.01	2.32	3.05	3.34	4.76	5.56

アクセス性能(間接アクセス) L1アクセス

		京		FX100		PRIMERGY RX2530 M1		PRIMERGY RX2540 M4		A64FX			
		浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	浮動小数点演算性能(Gflops)	性能向上比	演算効率(%)	演算効率(理想値)(%)
ロードなし・間接SIMDストア×1	固定値	0.98	1.00	1.82	1.88	1.82	1.95	0.15	0.16	1.56	1.59	2.44	
	連続	0.82	1.00	1.58	1.94	1.79	2.30	0.16	0.21	1.53	1.86	2.39	
	飛び幅16	0.75	1.00	0.91	1.23	1.80	2.53	0.39	0.54	1.22	1.63	1.90	
間接SIMDロード×2・間接SIMDストア×1	固定値	1.47	1.00	2.29	1.57	1.51	1.08	0.29	0.21	2.06	1.40	3.22	
	連続	1.05	1.00	1.84	1.78	1.48	1.49	0.34	0.34	2.01	1.92	3.14	
	飛び幅16	0.55	1.00	1.13	2.08	1.50	2.88	0.64	1.23	0.90	1.65	1.41	
間接SIMDロード×2・連続SIMDストア×1	固定値	1.45	1.00	2.19	1.52	1.74	1.26	1.99	1.44	4.12	2.84	6.44	10.00
	連続	1.12	1.00	2.18	1.97	1.75	1.64	1.95	1.83	4.12	3.67	6.44	10.00
	飛び幅16	0.98	1.00	1.72	1.77	1.38	1.47	1.59	1.70	2.73	2.77	4.26	5.56

FUJITSU

スループット性能

- 測定条件
- 他CPU比較 測定結果の算出方法
- L1キャッシュアクセス
- L2キャッシュアクセス
- メモリアクセス：zfill なし
  - 【参考】zfill による性能向上
- メモリアクセス：CMGを跨ぐアクセス

測定条件

FUJITSU

■ 測定条件

	パターン
測定パターン	3本 - L1キャッシュアクセススループット測定 - L2キャッシュアクセススループット測定 - メモリアクセススループット測定
測定コア	- L1キャッシュアクセススループット測定 ⇒ 1コア実行 (CMG0の計算コア) - L2キャッシュ/メモリアクセススループット測定 ⇒ 12コア実行(CMG0)
翻訳オプション	-Kfast ※プリフェッチ系,zfillのオプションは適宜資料内に記載
型	倍精度実数型
アクセス範囲	- bss を使用 - n(最内ループ繰返し数、配列サイズ)は以下 - L1=L1キャッシュサイズの3/4 (A64FX で 2048) - L2=L2キャッシュサイズの1/2 (A64FX で 174720) - メモリ=L2キャッシュサイズの3倍 (A64FX で 1048512) ※各配列は256バイトアラインで確保している
外ループ繰返し(iter)	- L1=1000000 - L2=10000 - メモリ=3000

検証コード

```
!$omp parallel
  Do j = 1, iter
!$omp do
      Do i = 1, n
         y(i)=x1(i) + c0 * x2(i)
      End Do
!$omp end do nowait
  End Do
!$omp end parallel
```

諸元(スループット):
・L1アクセス : read : 256 GB/s,
write : 128 GB/s
・L2アクセス : 1024 GB/s
・メモリアクセス : 256 GB/s (1CMG)

他CPU比較 測定結果の算出方法

FUJITSU

■ 他CPU比較 測定結果の算出方法

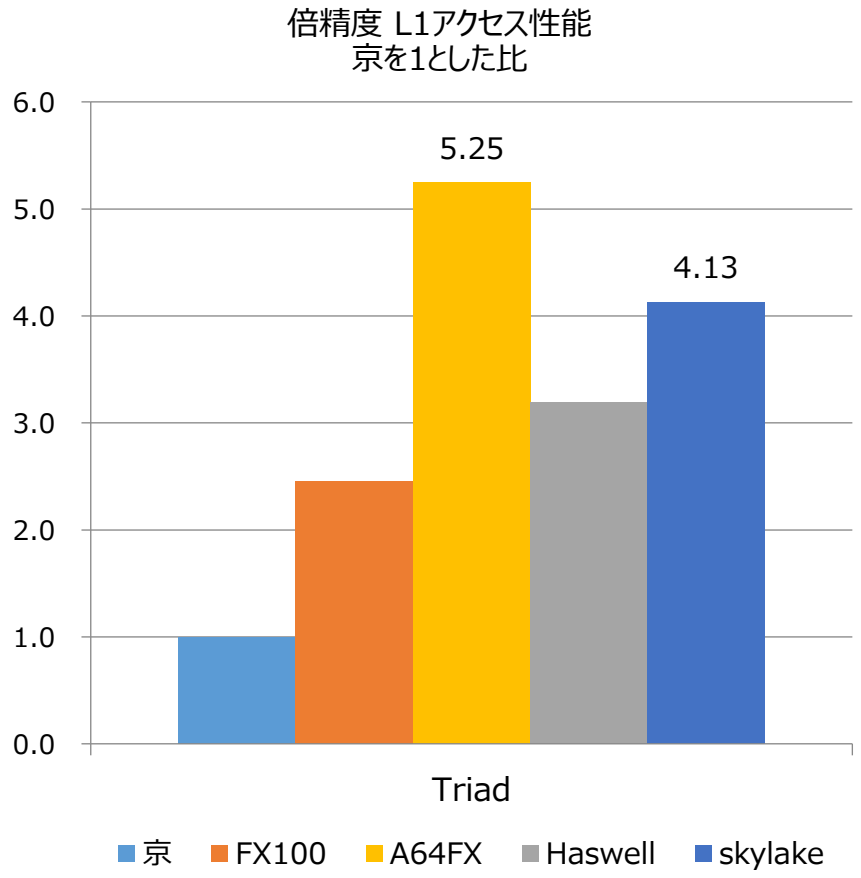
測定結果	
浮動小数点演算性能 (Gflops)	浮動小数点演算数÷経過時間÷10^9 ただし、PRIMERGY(Haswell, Skylake) における浮動小数点演算数は FX100 と同一とした。
整数演算性能 (GOPS)	整数演算数÷経過時間÷10^9 ただし、整数演算数は浮動小数点演算数と同一とした。
スループット (GB/s)	データ転送量÷経過時間÷10^9 ただしデータ転送量は、STREAM ベンチマークと同様に、データ書き込み先キャッシュラインの読み込み分を含めない値とした。
性能向上比 (京を1とした比)	演算性能の場合： 浮動小数点演算性能÷京における浮動小数点演算性能 整数演算性能÷京における整数演算性能 データアクセス性能(スループット)の場合： スループット÷京におけるスループット ただし演算性能については、京と同じCPU 動作周波数に換算し計算した。

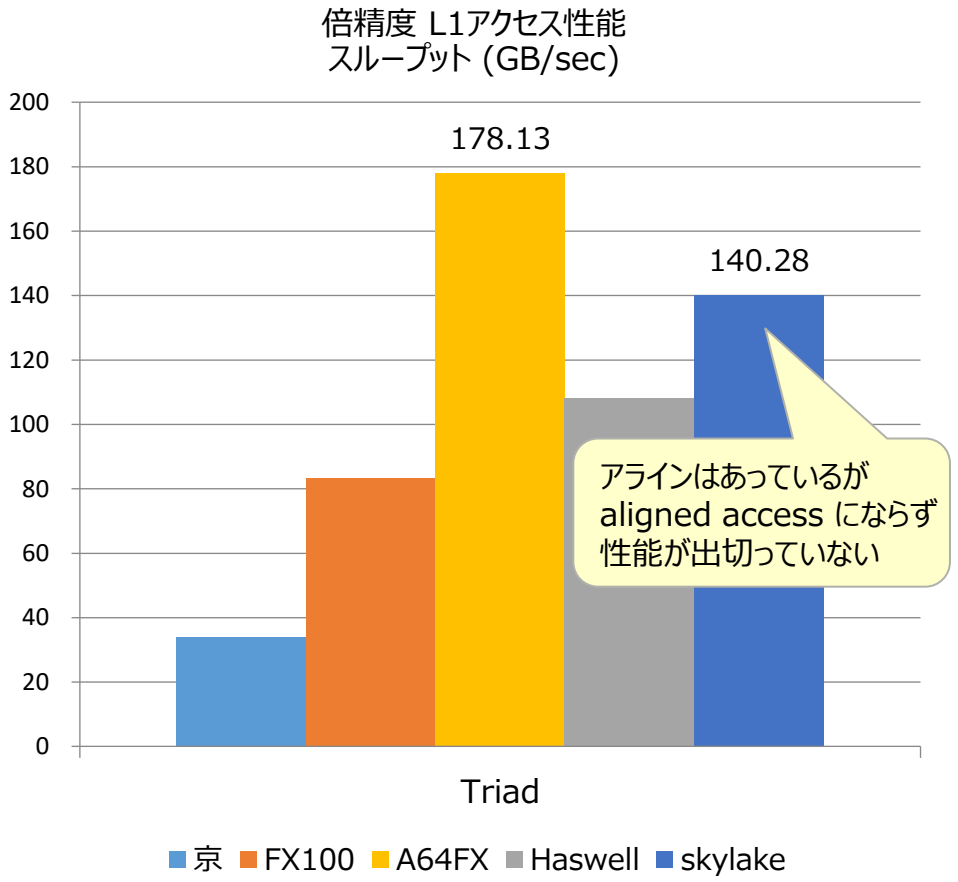
L1キャッシュアクセス

- 京を1とした比 、スループット
  - Triad 連続を評価

	Scatter ST	ST	Gather LD	LD	式
Triad 連続	0	1	0	2	y(i) = x1(i) + scalar * x2(i)

L1キャッシュスループットの
諸元数値は
Read : 256 GB/s,
Write : 128 GB/s



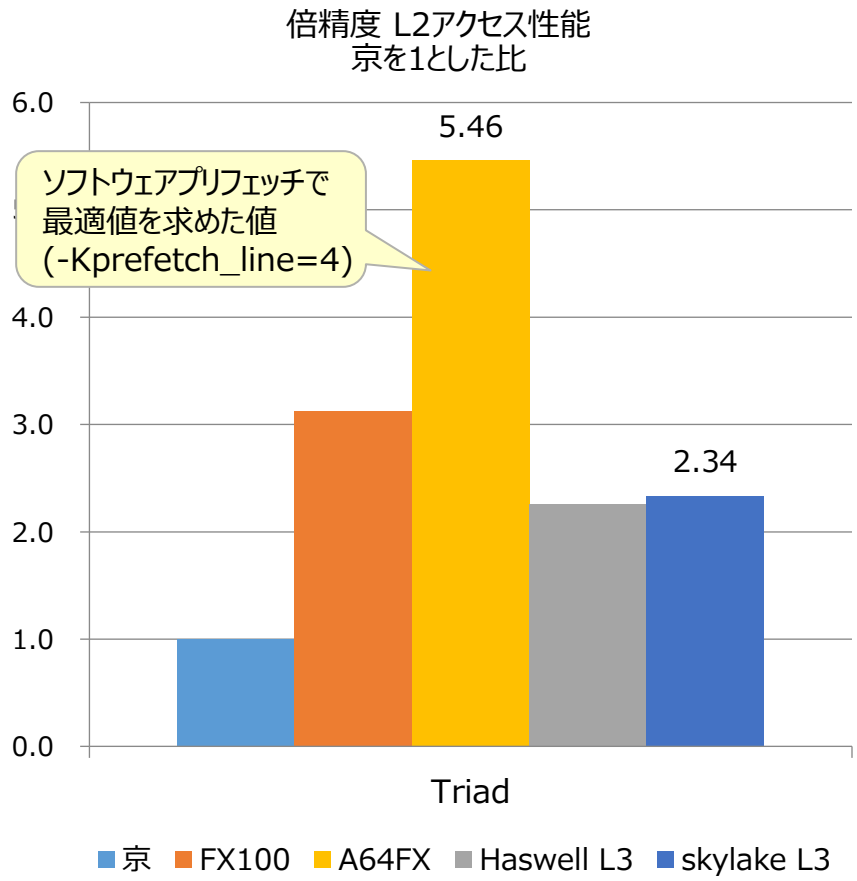


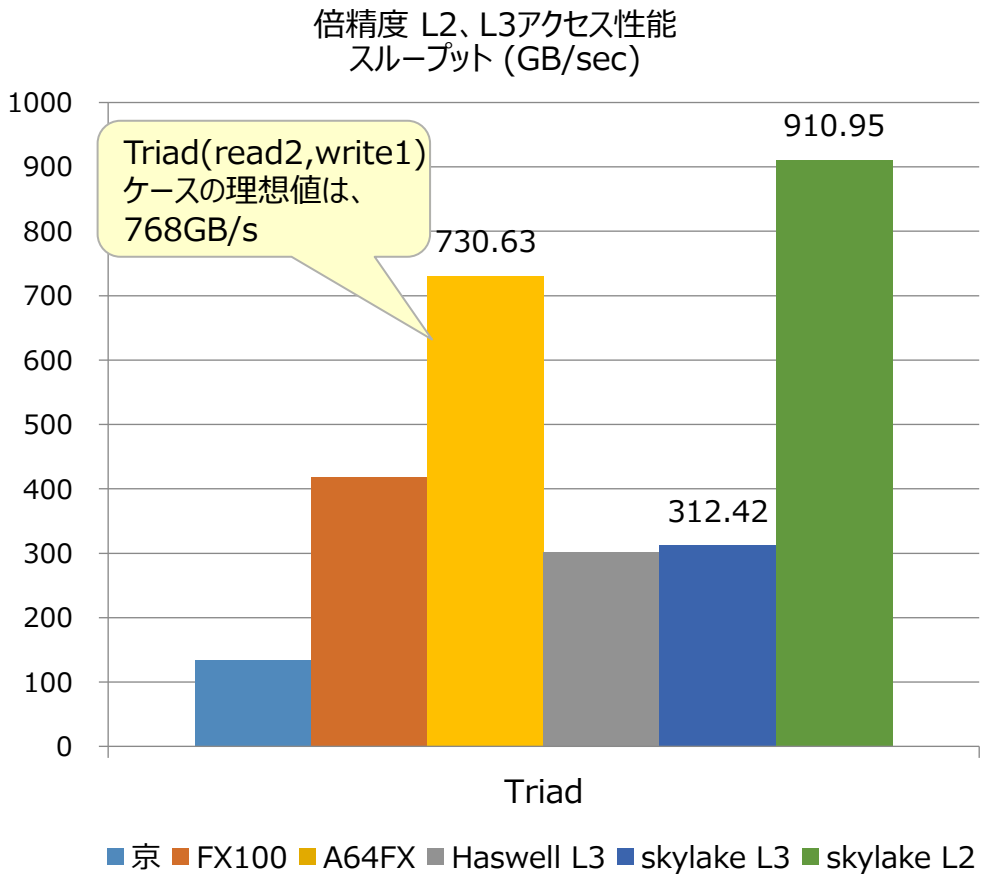
L2キャッシュアクセス (1/3)

- 京を1とした比 、L2スループット (CMGあたり)
  - Triad 連続を評価

	Scatter ST	ST	Gather LD	LD	式
Triad 連続	0	1	0	2	y(i) = x1(i) + scalar * x2(i)

L2キャッシュスループットの
諸元数値は
1024 GB/s





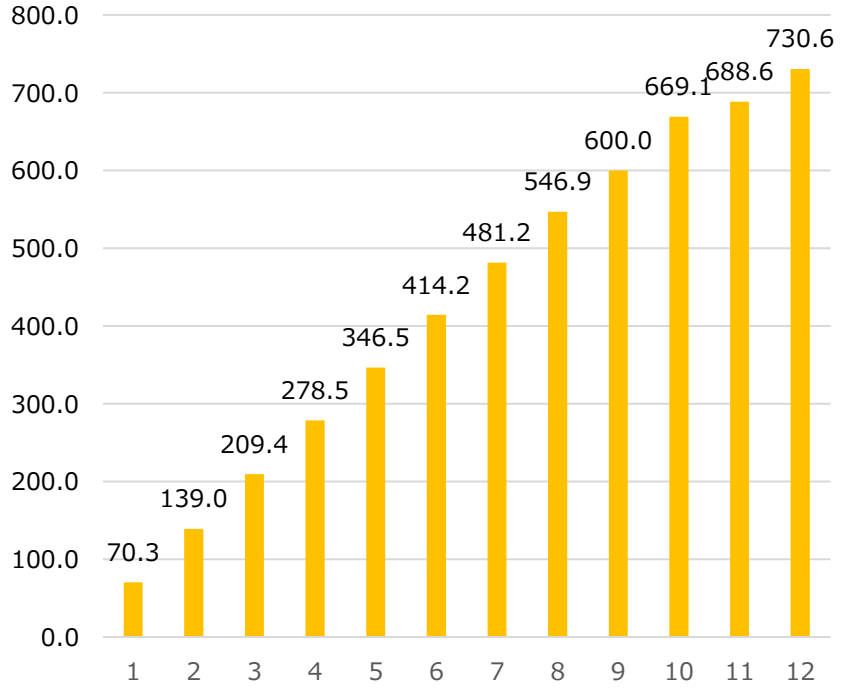
L2キャッシュアクセス (2/3)

FUJITSU

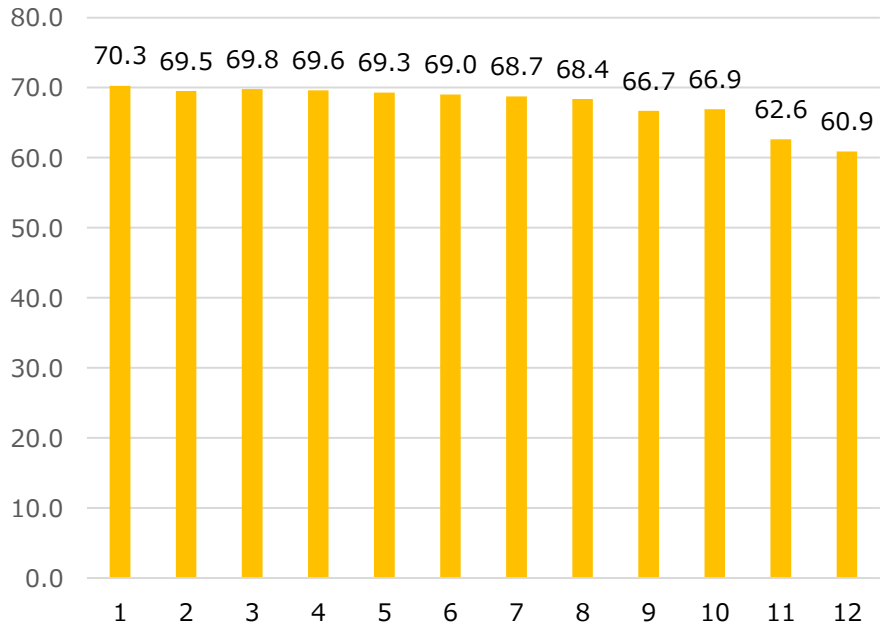
- 実行スレッド数とスループット

CMG0を使用し、
1スレッド～12スレッド実行

スループット (GB/sec)



スループット/コア数
(GB/sec)

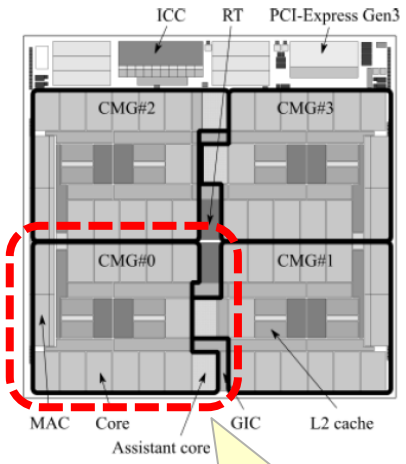


10並列(10コア)でCMGの性能をほぼ出すことができている。

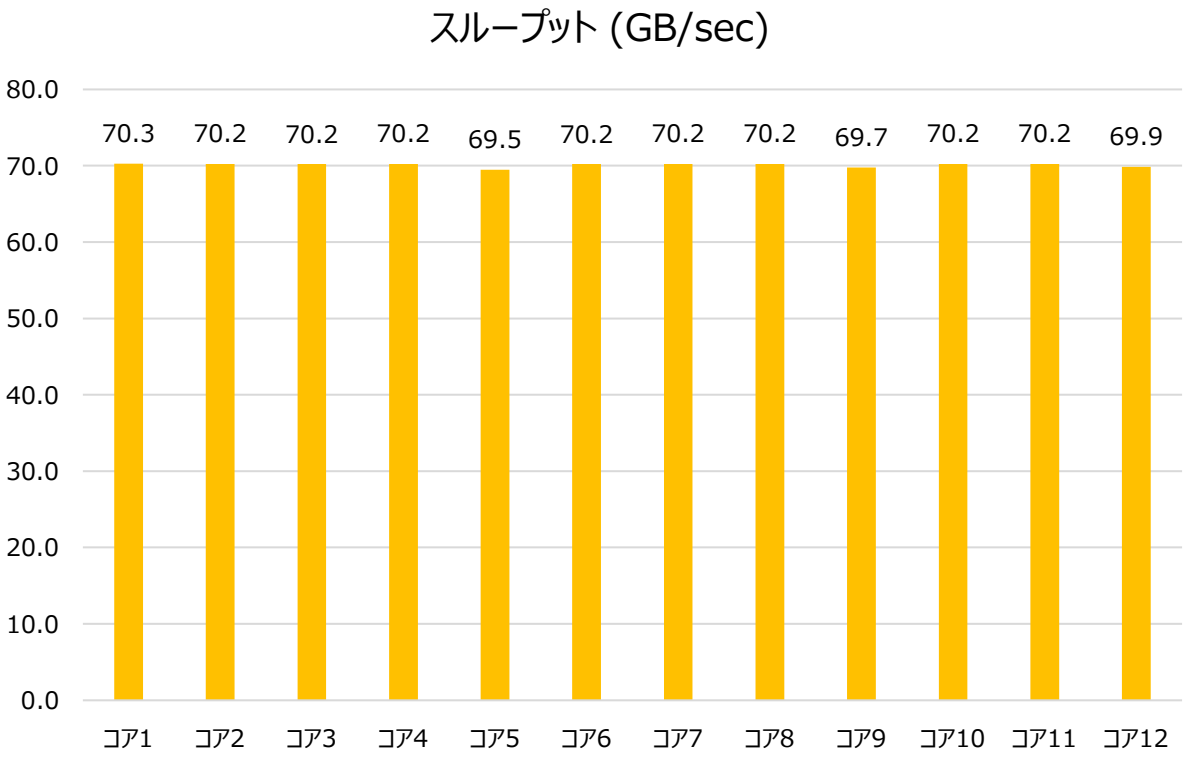
L2キャッシュアクセス (3/3)

FUJITSU

■ 実行コアによるスループットの違い



CMG0のコア1～12を使用し、コア毎に1スレッド実行



実行コアによるスループットに大きな違いは見られなかった。

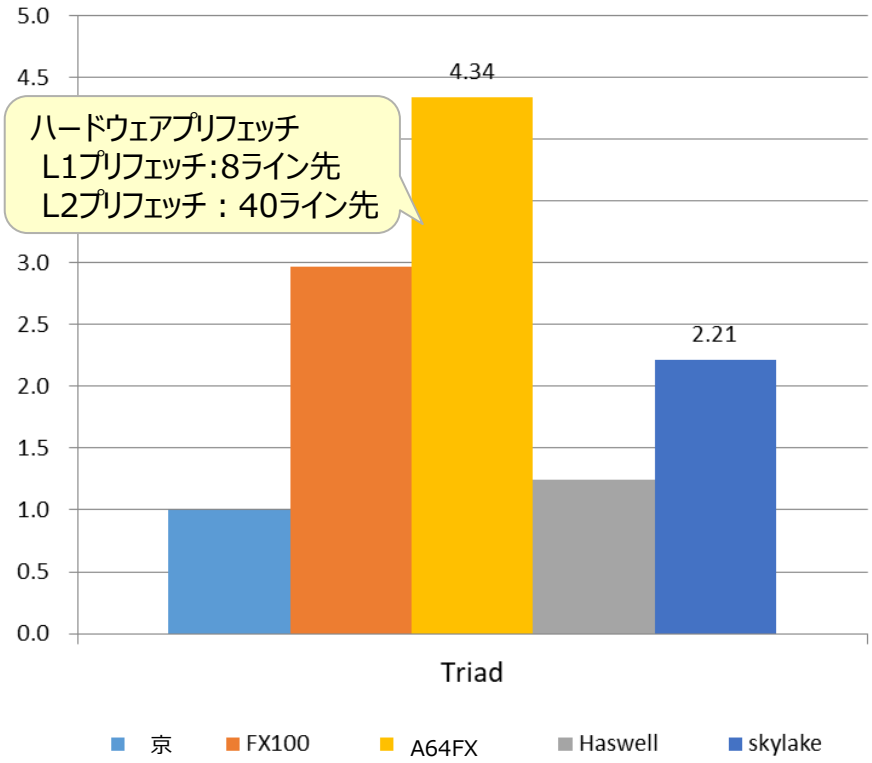
メモリアクセス：zfill なし (1/3)

- 京を1とした比 、メモリスループット (CMGあたり)
  - Triad 連続を評価

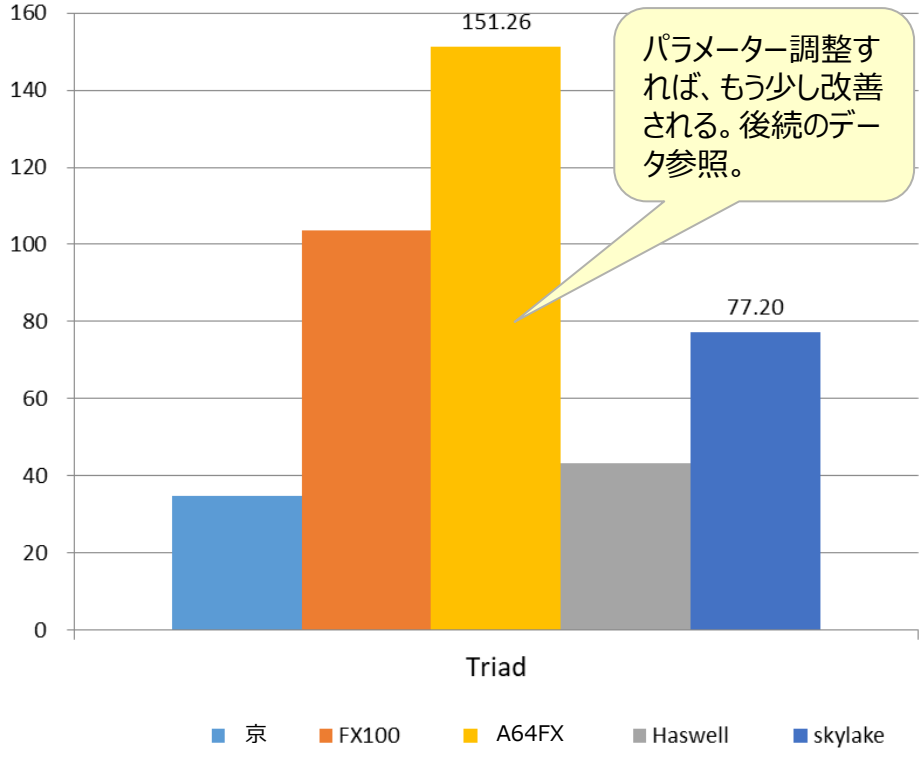
	Scatter ST	ST	Gather LD	LD	式
Triad 連続	0	1	0	2	y(i) = x1(i) + scalar * x2(i)

※ ここでの数値(メモリスループット)は、データ書き込み先キャッシュラインの読み込み分を含めない値になります。(STREAMベンチマークと同様)

倍精度 Memoryアクセス性能
京を1とした比



倍精度 Memoryアクセス性能
スループット (GB/sec)



メモリアクセス：zfill なし (2/3)

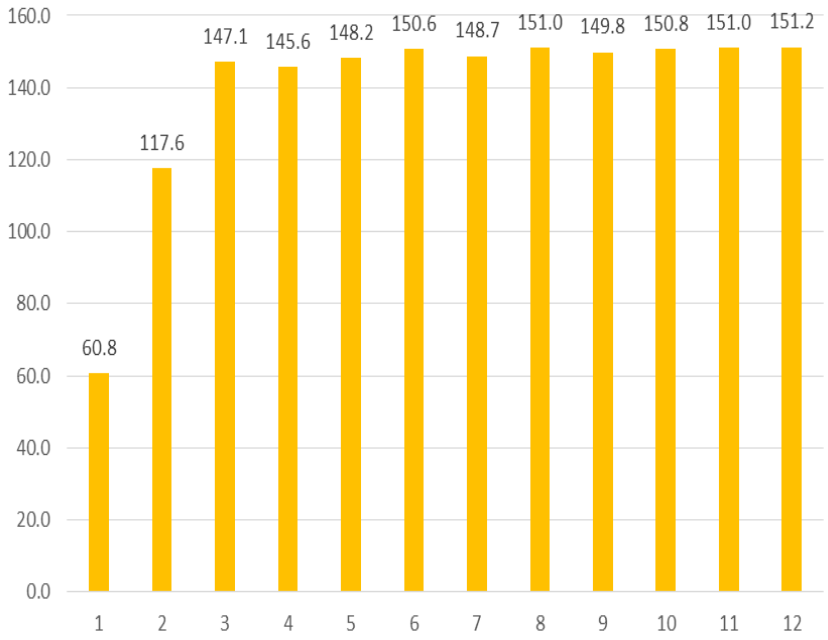
FUJITSU

- 実行コア数とスループット
  (ハードウェアプリフェッチ)

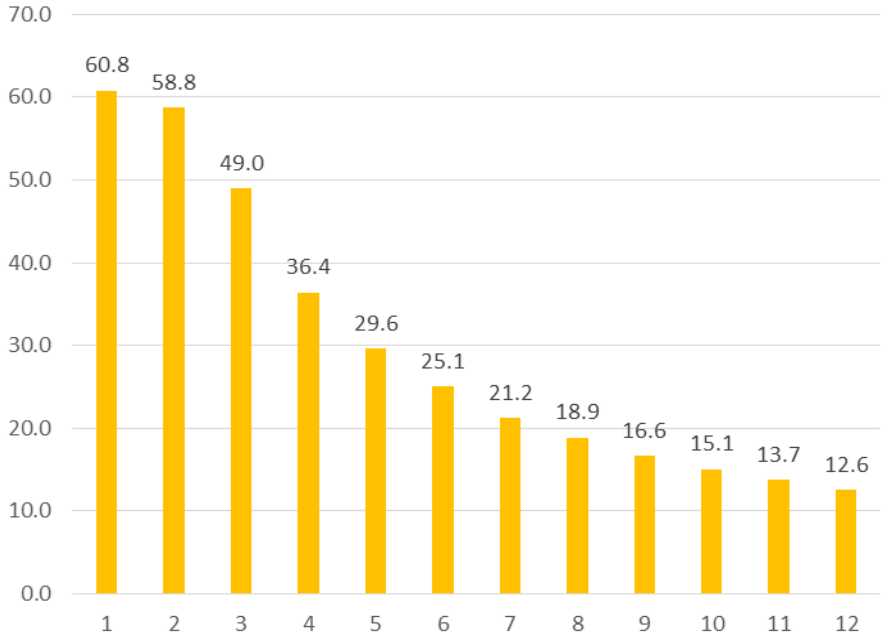
※ ここでの数値(メモリスループット)は、
データ書き込み先キャッシュラインの
読み込み分を含めない値になります。
(STREAMベンチマークと同様)

CMG0を使用し、
1スレッド～12スレッド実行

スループット (GB/sec)



スループット/コア数 (GB/sec)



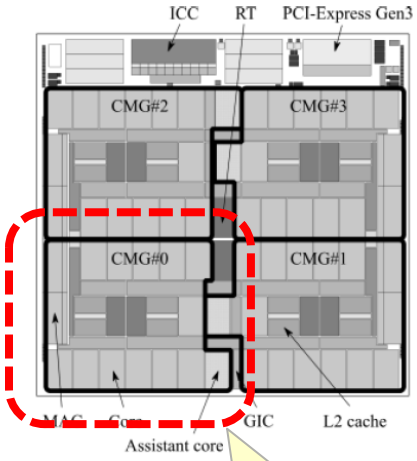
3並列(3コア)でCMGの性能をほぼ出すことができている。

メモリアクセス：zfill なし (3/3)

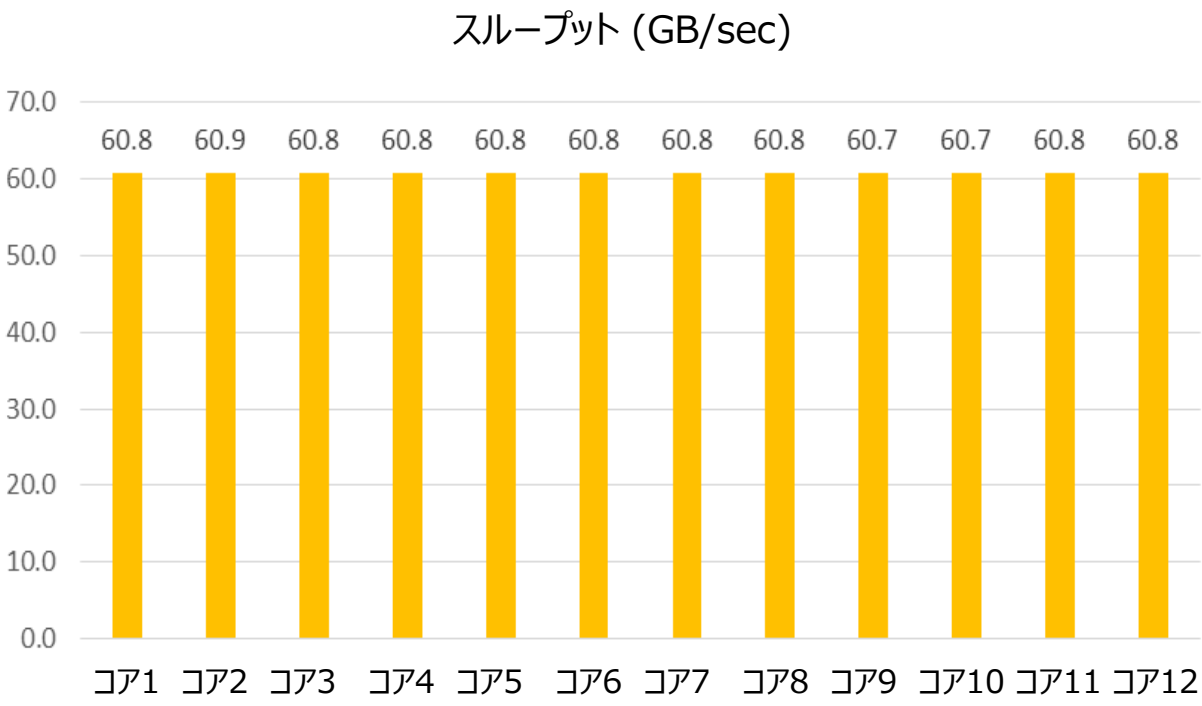
FUJITSU

- 実行コアによるスループットの違い
  (ハードウェアプリフェッチ)

※ ここでの数値(メモリスループット)は、
データ書き込み先キャッシュラインの
読み込み分を含めない値になります。
(STREAMベンチマークと同様)



CMG0のコア1～12を
使用し、コア毎に1スレ
ッド実行

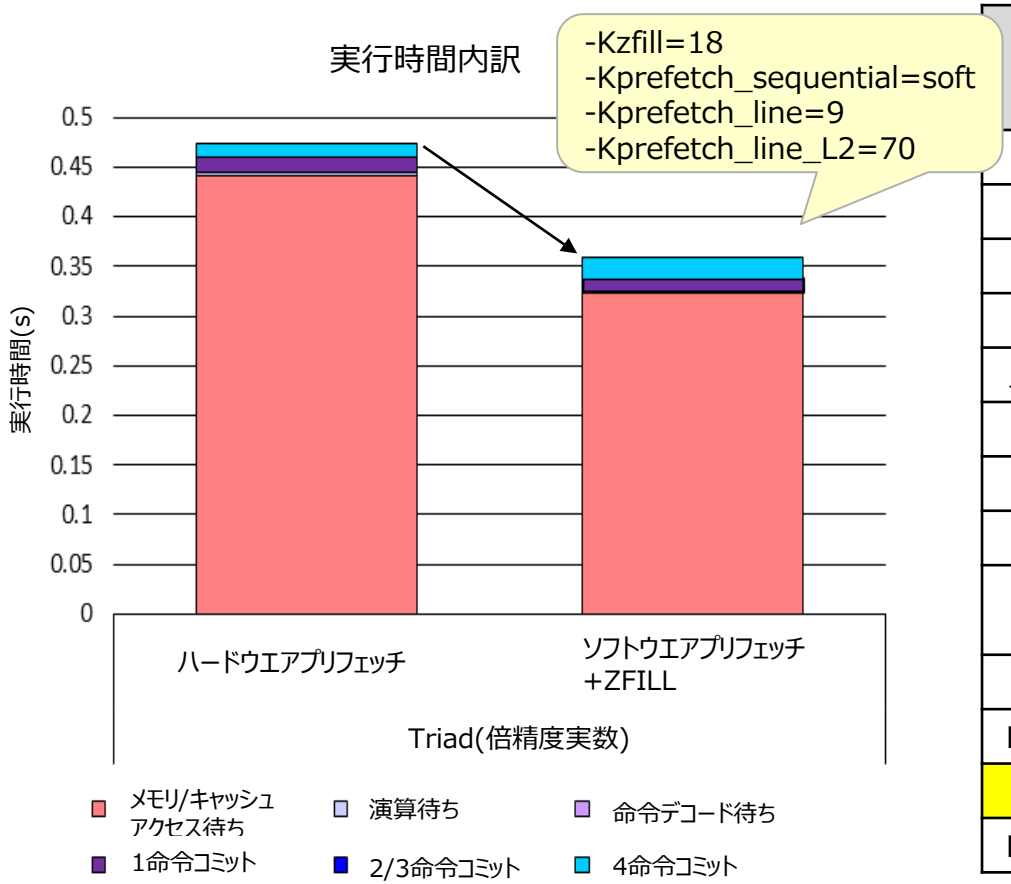


実行コアによるスループットに大きな違いは見られなかった

【参考】zfill による性能向上

FUJITSU

ソフトウェアプリフェッチと zfill の性能



PAの値

	ハードウェア プリフェッチ	ソフトウェア プリフェッチ + zfill
集計スレッド番号	0	0
実行時間 [s]	0.474	0.359
有効総命令数	2.16E+09	2.52E+09
GFLOPS	13.27	17.54
メモリスループット [GB/s]	**222.5**	210.8
L1ビジー率/スレッド	52.9%	55.09%
L2ビジー率	85.1%	94.39%
メモリビジー率	21.7%	20.58%
浮動小数点パイプライン ビジー率/スレッド	FLA:4.90% FLB:2.80%	FLA:7.14% FLB:3.01%
L1ミス数/スレッド	2.46E+07	2.46E+07
L1ミス デマンド率/スレッド	9.40%	0.08%
L2ミス数/スレッド	**2.62E+07**	1.64E+07
L2ミス デマンド率/スレッド	**12.21%**	0.31%

※アクセスサイズ(最内ループ繰返し数、配列サイズ)を240MB(10倍)にして評価

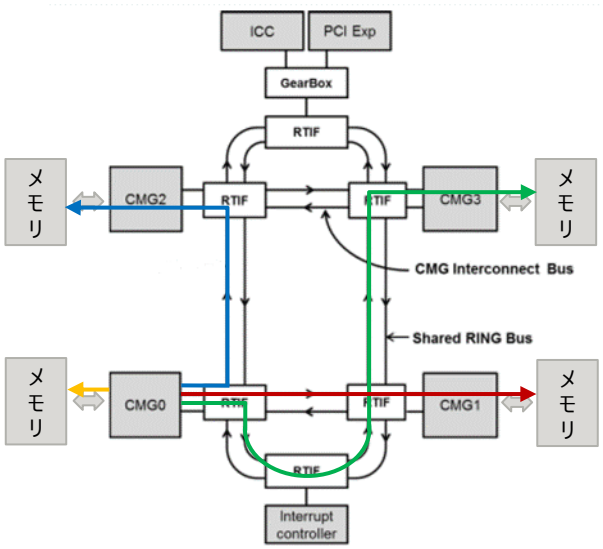
zfill を的確に行えば、210.8GB/ s まで性能向上する。

メモリアクセス：CMGを跨ぐアクセス

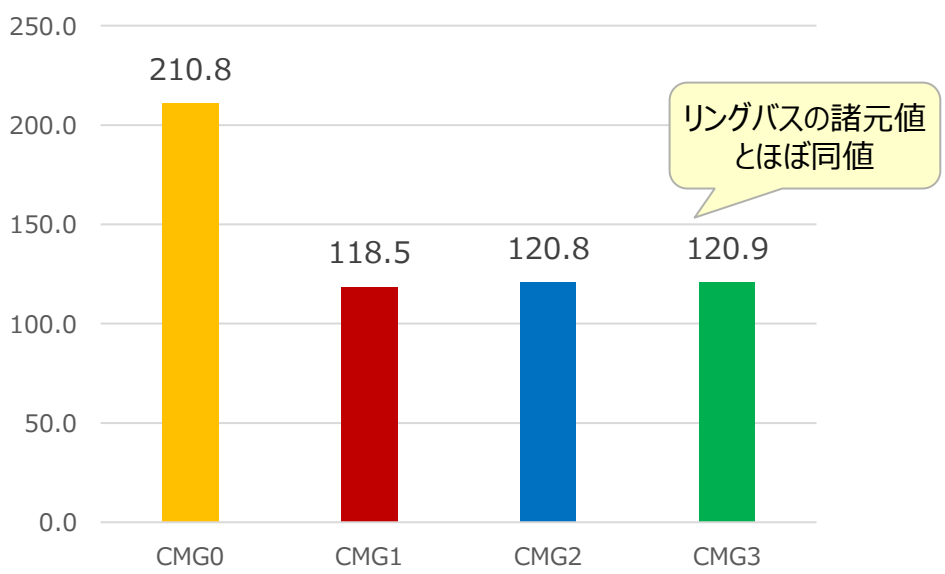
FUJITSU

- CMGを跨ぐメモリアクセス

12スレッド実行(CMG0)
numactl –m4～7
CMG0～3のメモリにアクセス



zfill でチューニングされたプログラムを使用



リングバスの諸元値とほぼ同値

ほぼ諸元値とおりの性能がでているが、
CMGを跨ぐメモリアクセスは約半分の転送速度となる
並列(OpenMPやMPI)をする際には注意が必要

FUJITSU

データ型の違いによる性能評価

- 連続アクセス
- Gatherロード・Scatterストアアクセス

連続アクセス

- 評価条件
- 評価結果： L1 キャッシュスループット
- 評価結果： L2 キャッシュスループット
- 評価結果： メモリスループット (zfill あり/なし)

連続アクセス評価条件

FUJITSU

- スループット評価条件
  - 評価アクセス領域
    L1キャッシュ、 L2キャッシュ、 メモリ(zfill有・無)
  - 型パターン
    - 実数型：倍精度実数、 単精度実数、 半精度実数(FP16)
    - 整数型：8バイト整数、 4バイト整数、 2バイト整数、 1バイト整数
  - データアクセスパターン
    - 連続SIMDロード2+連続SIMDストア1
  - 評価コードは右記。
  - その他詳細は以下。

半精度実数は2バイト整数パターンを基にアセンブラを手修正して評価

評価コード

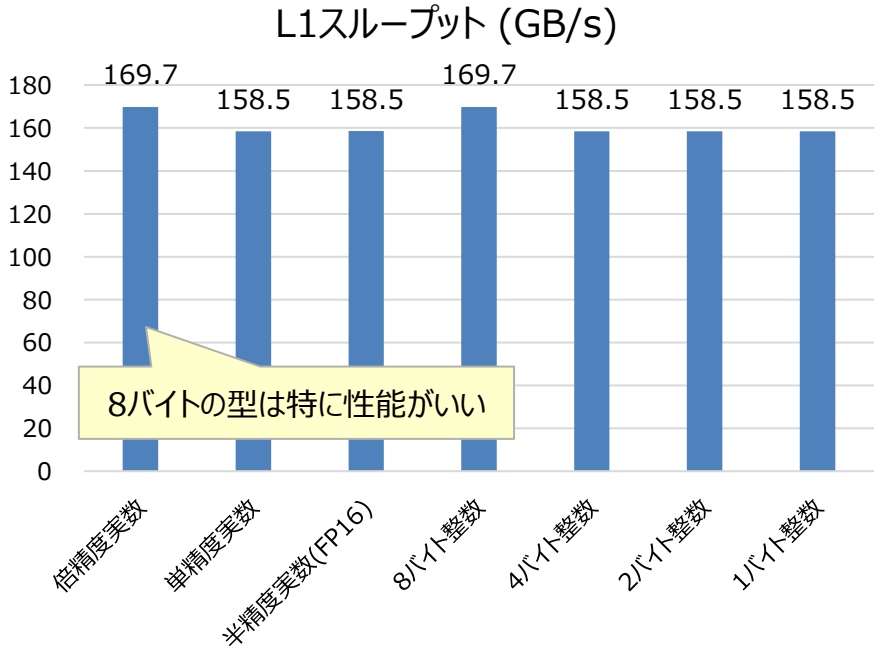
```
do k  = 1, iter
   do i = 1,n
      y(i) = x1(i) + c * x2(i)
   enddo
enddo
```

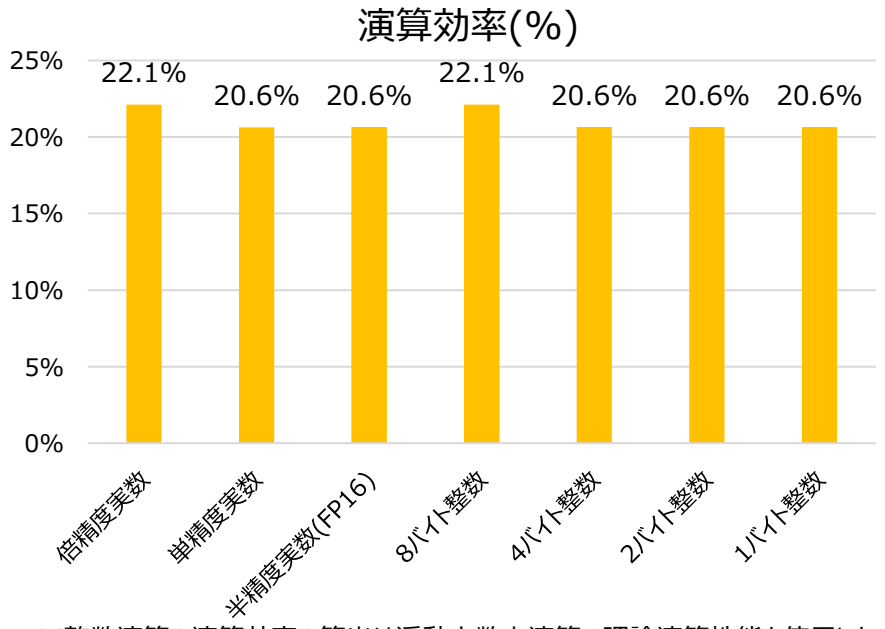
評価アクセス領域	実行コア数	n(配列サイズ・最内ループ)	iter(外側ループ)
L1キャッシュ	1	L1キャッシュサイズの1/2	1000000
L2キャッシュ	12	L2キャッシュサイズの1/2	10000
メモリ	12	L2キャッシュサイズの30倍	300

連続アクセス評価結果：L1キャッシュスループット

FUJITSU

L1キャッシュスループットの諸元数値は
Read 1 ：256 GB/s,
Write 1 ：128 GB/s,
Read2,Write1:192 GB/s





※整数演算の演算効率の算出は浮動小数点演算の理論演算性能を使用した

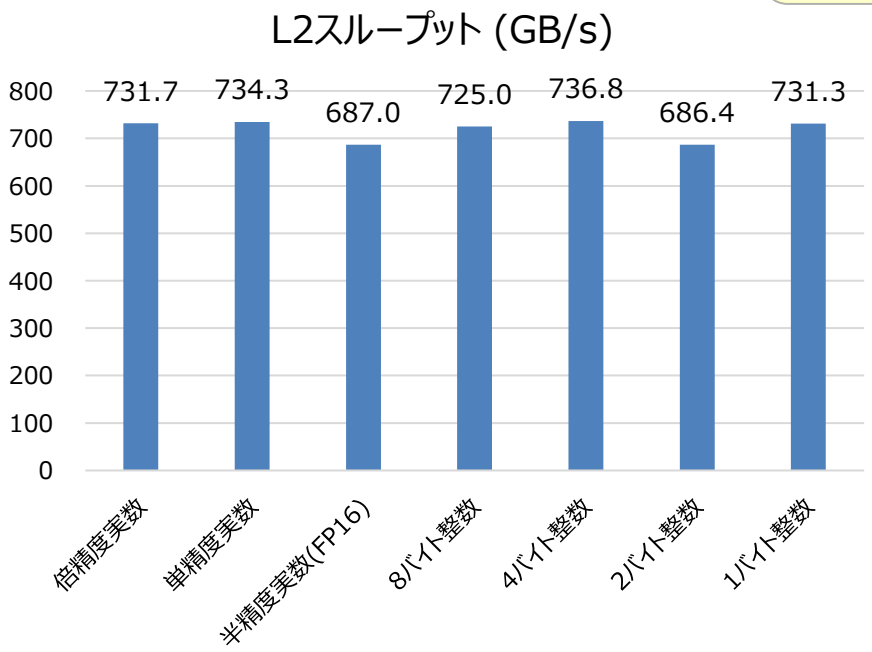
型	スループット (GB/s)	GFLOPS /GOPS	演算性能比	L1ビジー率
倍精度実数	169.7	14.1	0.54	96.7%
単精度実数	158.5	26.4	1.00	96.2%
半精度実数(FP16)	158.5	52.8	2.00	97.4%
8バイト整数	169.7	14.1	0.54	96.7%
4バイト整数	158.5	26.4	1.00	97.4%
2バイト整数	158.5	52.8	2.00	97.4%
1バイト整数	158.5	105.7	4.00	97.4%

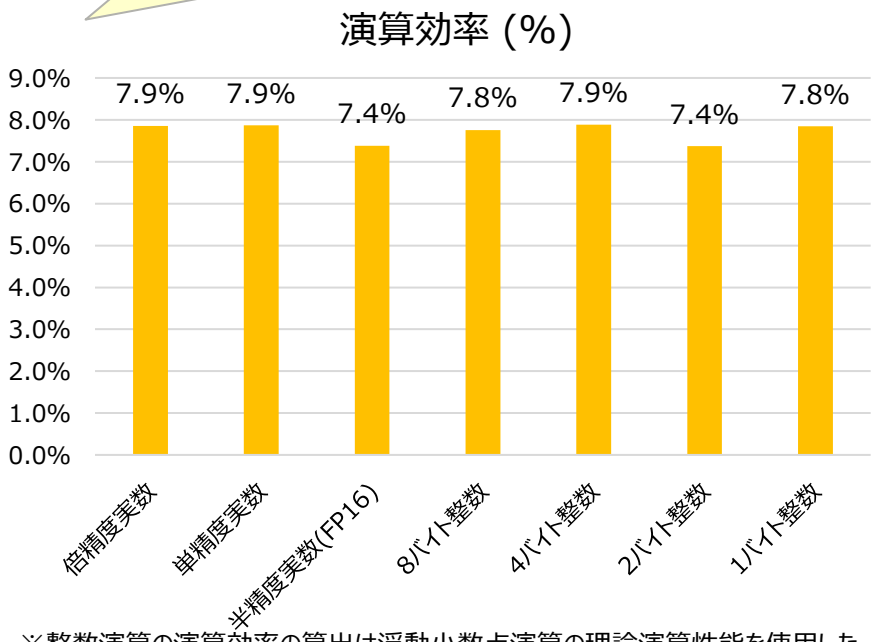
連続アクセス評価結果：L2キャッシュスループット

FUJITSU

-Kprefetch_sequential=soft
-Kprefetch_cache_level=1
-Kprefetch_line=4

L2キャッシュスループットの諸元数値は
Read 1 ：1024 GB/s,
Write 1 ：512 GB/s,
Read2,Write1: 768 GB/s



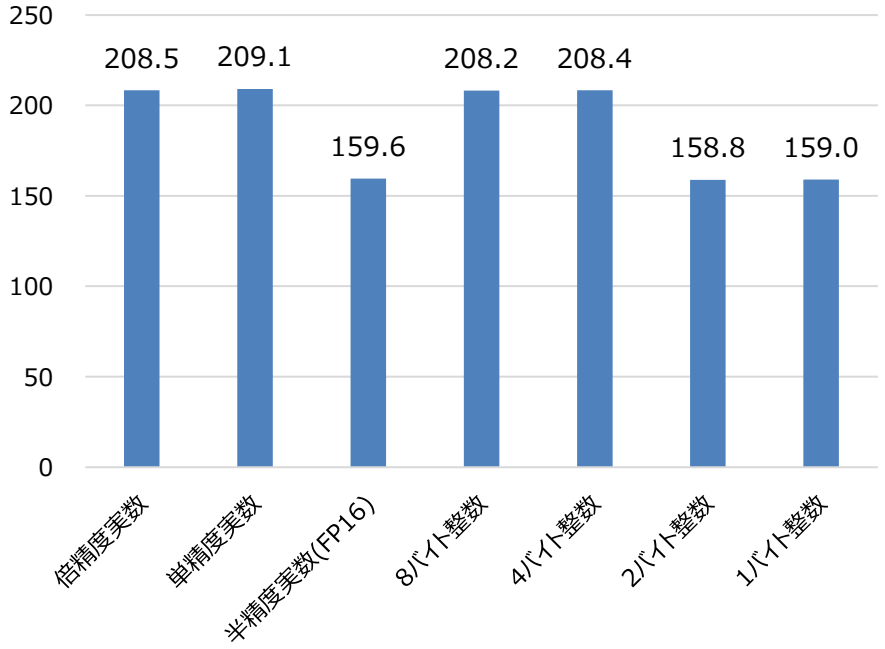


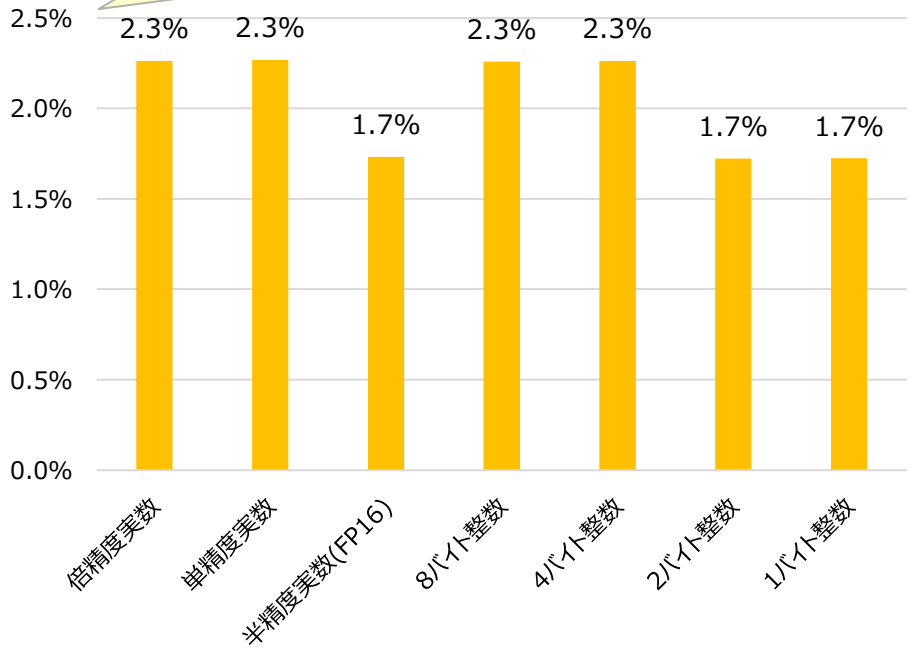
※整数演算の演算効率の算出は浮動小数点演算の理論演算性能を使用した

型	スループット (GB/s)	GFLOPS /GOPS	L2ビジー率	L1Dミス数
倍精度実数	731.7	60.3	96.5%	1.66E+08
単精度実数	734.3	120.9	97.0%	1.66E+08
半精度実数(FP16)	687.0	226.7	90.5%	1.65E+08
8バイト整数	725.0	59.6	96.5%	1.66E+08
4バイト整数	736.8	121.2	96.9%	1.66E+08
2バイト整数	686.4	226.5	90.5%	1.65E+08
1バイト整数	731.3	482.2	96.5%	1.66E+08

連続アクセス評価結果：メモリスループット(zfill あり)

-Kzfill=18
-Kprefetch_sequential=soft
-Kprefetch_line=9
-Kprefetch_line_L2=70





※整数演算の演算効率の算出は浮動小数点演算の理論演算性能を使用した

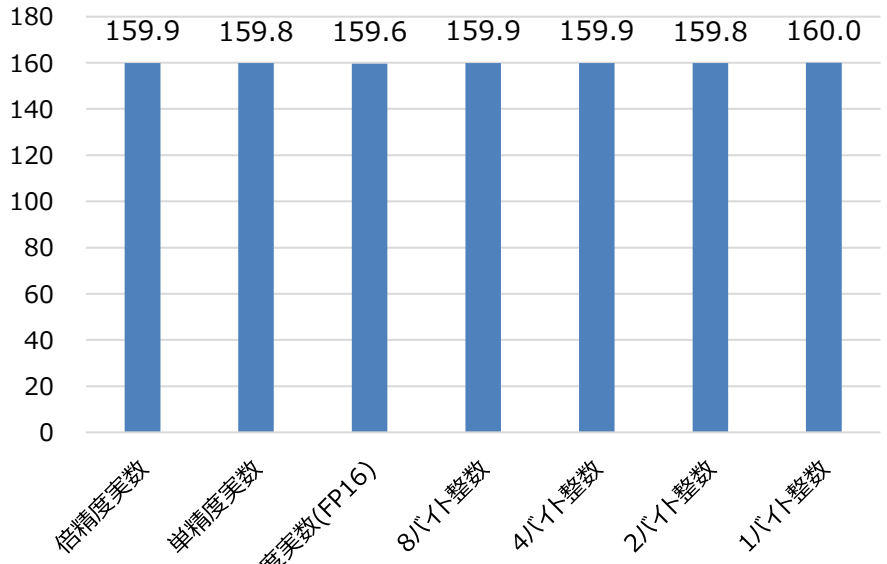
型	スループット (GB/s)	GFLOPS /GOPS	L2ビジー率	メモリビジー率	L1Dミス数	L2ミス数
倍精度実数	208.5	17.4	93.7%	81.7%	2.95E+08	1.97E+08
単精度実数	209.1	34.8	93.2%	81.9%	2.95E+08	1.98E+08
半精度実数(FP16)	159.6	53.2	84.8%	83.3%	2.95E+08	2.96E+08
8バイト整数	208.2	17.3	93.7%	81.6%	2.95E+08	1.98E+08
4バイト整数	208.4	34.7	93.2%	81.7%	2.95E+08	1.97E+08
2バイト整数	158.8	52.9	84.8%	83.0%	2.95E+08	2.96E+08
1バイト整数	159.0	106.0	84.8%	83.0%	2.95E+08	2.96E+08

連続アクセス評価結果：メモリスループット(zfill なし)

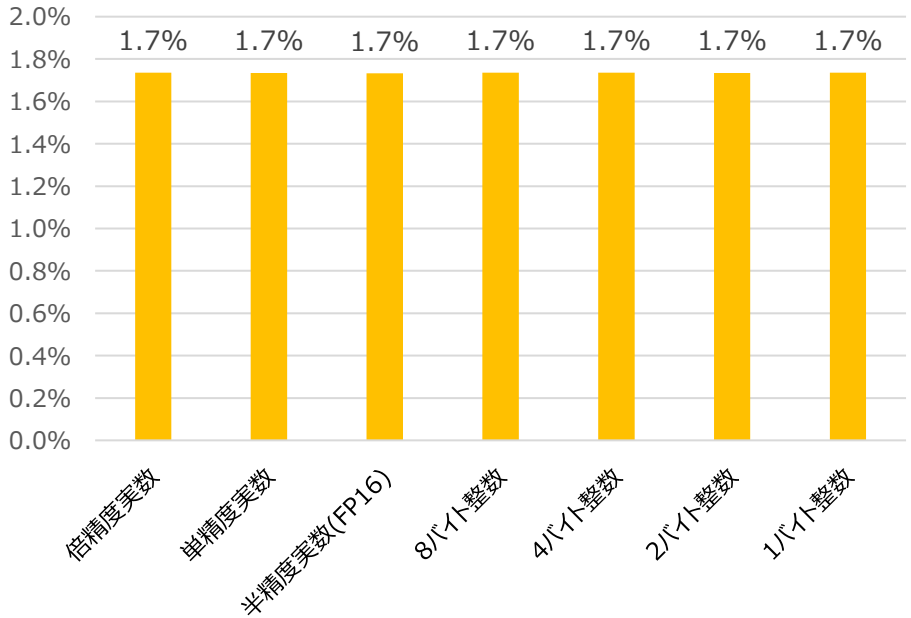
FUJITSU

-Kprefetch_sequential=soft
-Kprefetch_line=9
-Kprefetch_line_L2=70

メモリスループット(GB/s)



演算効率 (%)



storeを1アクセスとカウント。
2アクセスとカウントすると 約213GB/S

※整数演算の演算効率の算出は浮動小数点演算の理論演算性能を使用した

型	スループット (GB/s)	GFLOPS /GOPS	L2ビジー率	メモリビジー率	L1Dミス数	L2ミス数
倍精度実数	159.9	13.3	84.4%	83.5%	2.95E+08	2.96E+08
単精度実数	159.8	26.6	84.6%	83.5%	2.95E+08	2.96E+08
半精度実数(FP16)	159.6	53.2	84.7%	83.4%	2.95E+08	2.96E+08
8バイト整数	159.9	13.3	84.5%	83.5%	2.95E+08	2.96E+08
4バイト整数	159.9	26.7	84.5%	83.5%	2.95E+08	2.96E+08
2バイト整数	159.8	53.3	84.4%	83.5%	2.95E+08	2.96E+08
1バイト整数	160.0	106.6	84.6%	83.5%	2.95E+08	2.96E+08

FUJITSU

Gatherロード・Scatterストアアクセス

- GatherロードのISAについて
- 評価条件
- 評価結果： L1 キャッシュスループット
- 評価結果： L2 キャッシュスループット
- 評価結果： メモリスループット (zfill あり/なし)

Gatherロード・ScatterストアのISAについて

FUJITSU

- 4バイト整数型、8バイト整数型
  命令上、実数(単精度、倍精度)と同じ命令となるため、評価からは除外した。
  - 4バイト整数、単精度実数(例：Gatherロード命令)

```
ld1w    {z1.s}, p0/z, [x11, z0.s, sxtw]
```

  - 8バイト整数、倍精度実数(例：Gatherロード命令)

```
ld1d    {z1.d},  p1/z,  [x11,  z0.d]
```

- 1バイト整数、2バイト整数、半精度実数(FP16)
  1バイト整数、2バイト整数、半精度実数は命令仕様上、512バイト全体を使ったSIMD化(1バイト=64SIMD/2バイト=32SIMD)することができない。
  また、現在は8SIMD化までの対応となっている(将来的には16SIMDとなる)

```
ld1sh   {z1.d}, p1/z, [x11, z0.d]
```

インデックスが指定される
この指定の仕様により32SIMD,64SIMDの実装はできない

Gatherロード・Scatterストアアクセス評価条件

FUJITSU

- スループット評価条件
  - 評価アクセス領域
    L1キャッシュ、L2キャッシュ、メモリ(zfill有・無)
  - 型パターン
    - 実数型：倍精度実数、単精度実数
  - データアクセスパターン
    - 連続SIMDロード2+連続SIMDストア1
    - Gatherロード2+連続SIMDストア1
    - 連続SIMDロード2+Scatterストア1
    - Gatherロード2+Scatterストア1
  - 評価コードは右記。
    (評価コード２はGatherロード+Scatterストア)
  - その他詳細は以下。

評価コード例1

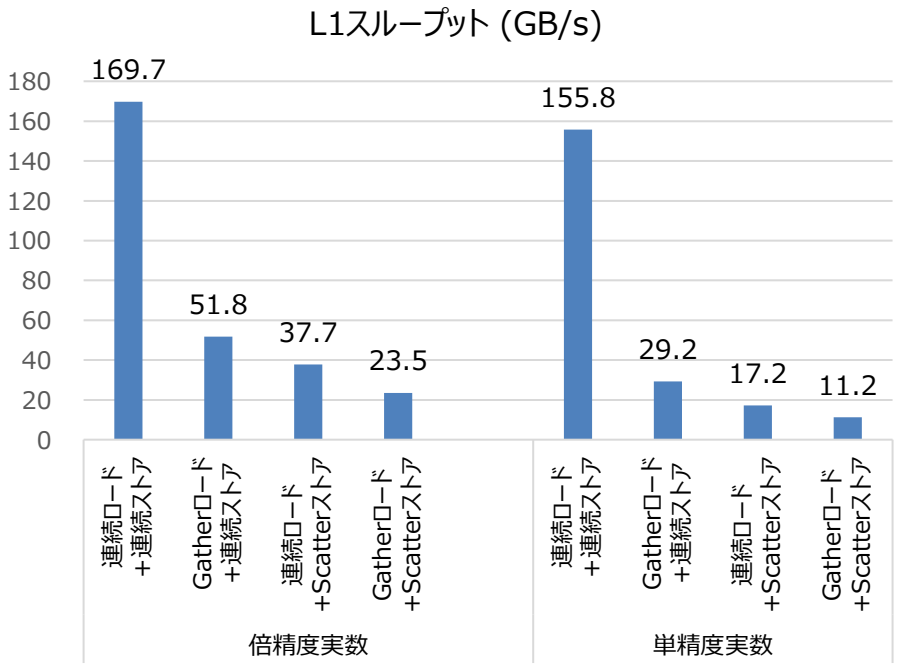
```
do k  = 1, iter
   do i = 1,n
      y(i) = x1(i) + c * x2(i)
   enddo
enddo
```

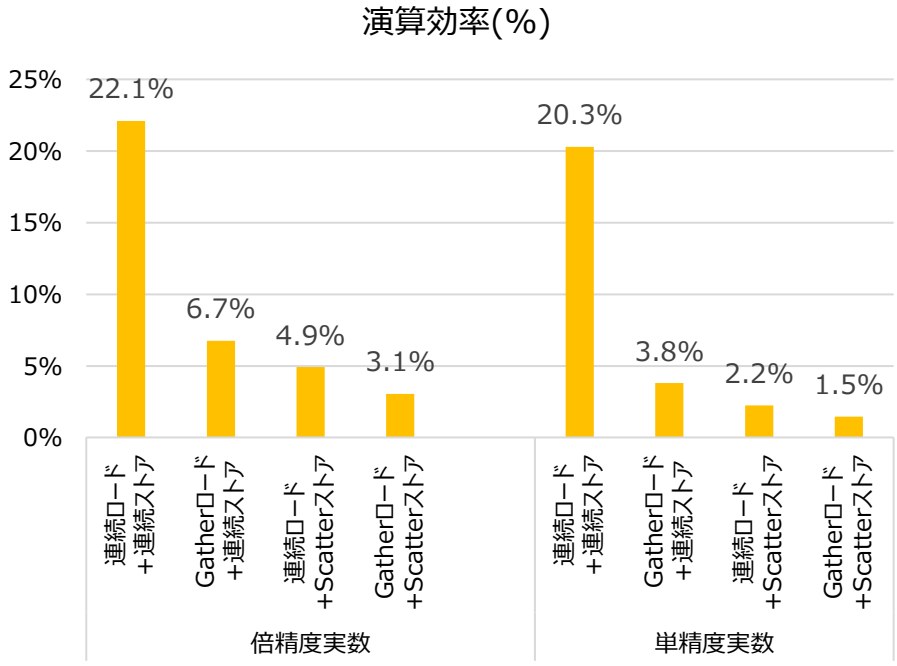
評価コード例2

```
do k  = 1, iter
   do i = 1,n
      y(1,i) = x1(1,i) + c * x2(1,i)
   enddo
enddo
```

評価アクセス領域	実行コア数	n(配列サイズ・最内ループ)	iter(外側ループ)
L1キャッシュ	1	L1キャッシュサイズの1/2	1000000
L2キャッシュ	12	L2キャッシュサイズの1/2	10000
メモリ	12	L2キャッシュサイズの30倍	300

評価結果：L1キャッシュスループット

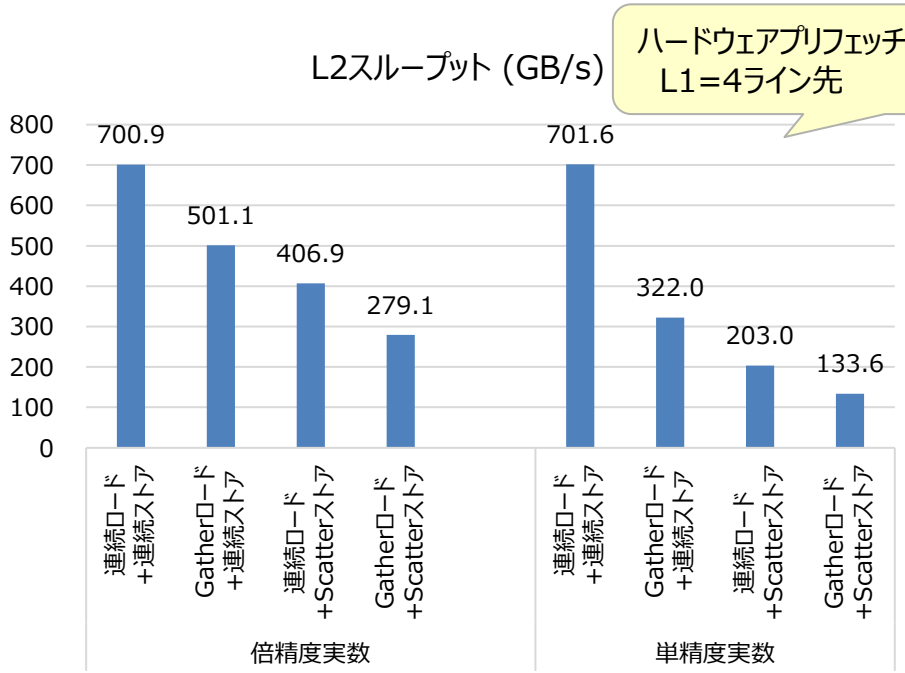


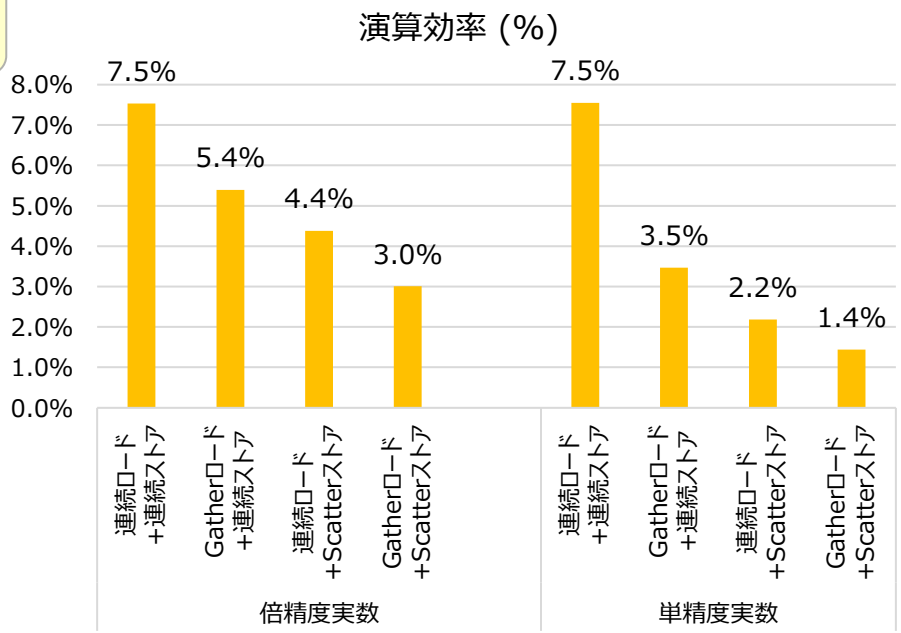


型	ロード・ストア	スループット (GB/s)	GFLOPS /GOPS	演算性能比	L1ビジー率
倍精度実数	連続ロードx2+連続ストアx1	169.7	14.1	1.00	96.7%
	Gatherロードx2+連続ストアx1	51.8	4.3	0.31	74.4%
	連続ロードx2+Scatterストアx1	37.7	3.1	0.22	59.3%
	Gatherロードx2+Scatterストアx1	23.5	2.0	0.14	55.0%
単精度実数	連続ロードx2+連続ストアx1	155.8	26.0	1.00	96.2%
	Gatherロードx2+連続ストアx1	29.2	4.9	0.19	73.9%
	連続ロードx2+Scatterストアx1	17.2	2.9	0.11	46.0%
	Gatherロードx2+Scatterストアx1	11.2	1.9	0.07	49.8%

Gather・Scatterを含むケースではL1スループットネックとはならない。

評価結果：L2キャッシュスループット

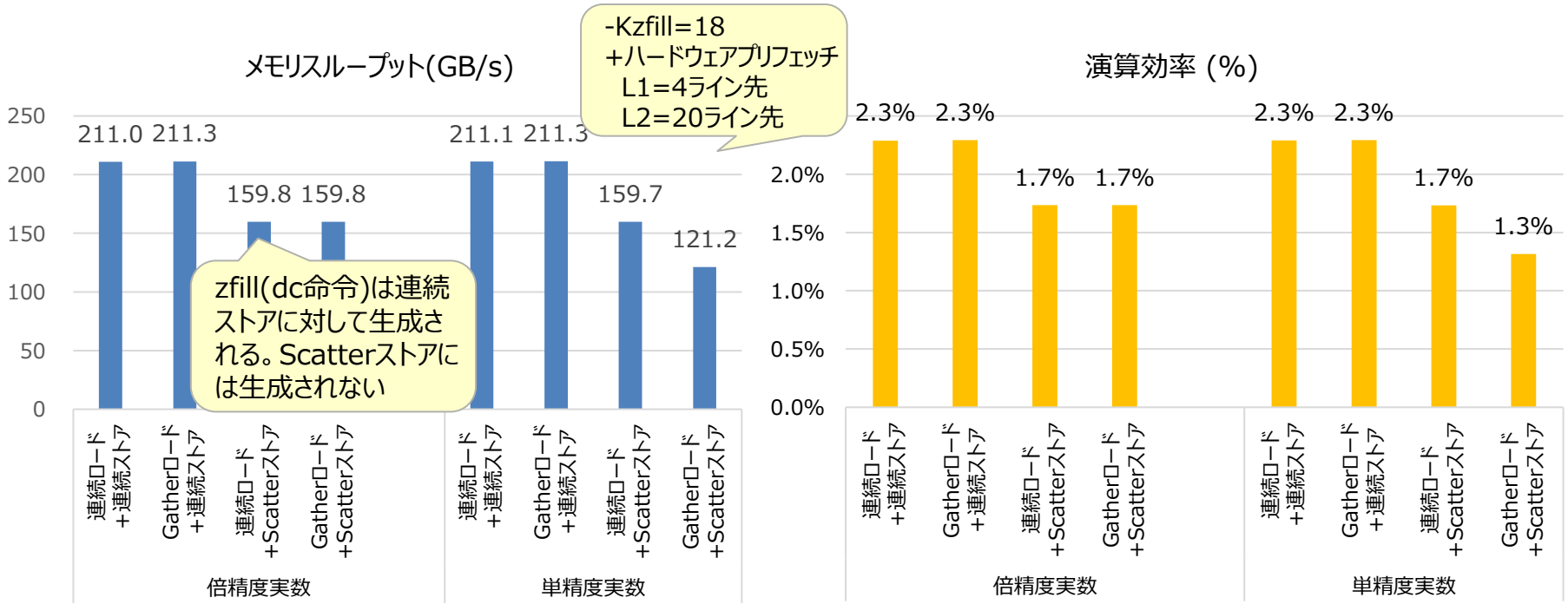




型	ロード・ストア	スループット (GB/s)	GFLOPS /GOPS	L2ビジー率	L1Dミス数
倍精度実数	連続ロードx2+連続ストアx1	700.9	57.9	97.6%	1.65E+08
	Gatherロードx2+連続ストアx1	501.1	41.5	71.6%	1.65E+08
	連続ロードx2+Scatterストアx1	406.9	33.7	59.0%	1.65E+08
	Gatherロードx2+Scatterストアx1	279.1	23.1	40.4%	1.65E+08
単精度実数	連続ロードx2+連続ストアx1	701.6	115.9	97.6%	1.65E+08
	Gatherロードx2+連続ストアx1	322.0	53.3	45.7%	1.65E+08
	連続ロードx2+Scatterストアx1	203.0	33.6	29.2%	1.65E+08
	Gatherロードx2+Scatterストアx1	133.6	22.1	20.4%	1.65E+08

Gather・Scatterを含むケースではL2スループットネックとはならない。

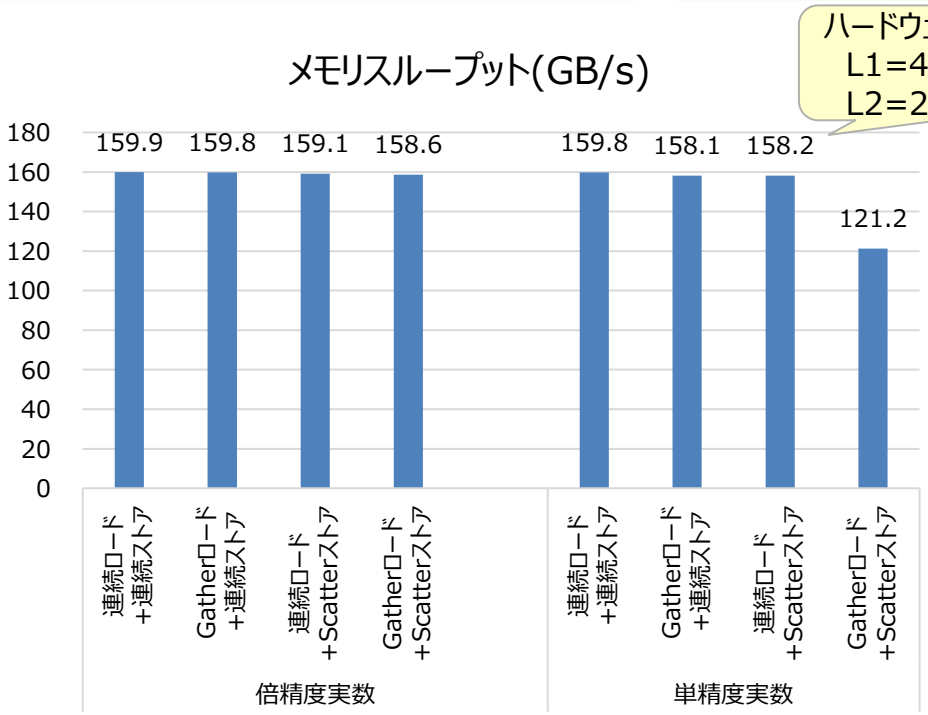
評価結果：メモリスループット(zfill あり)

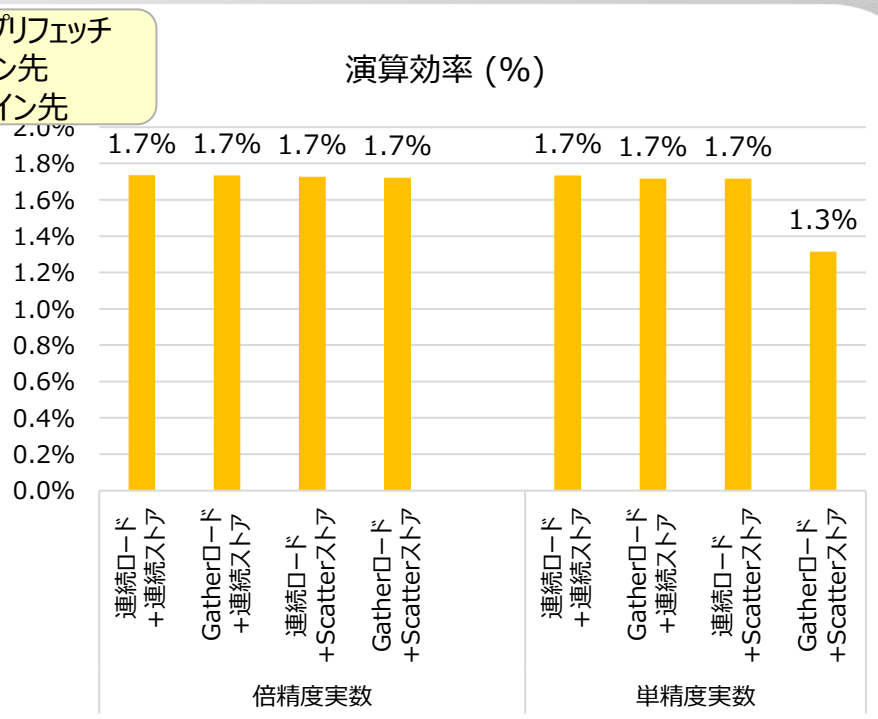


型	ロード・ストア	スループット (GB/s)	GFLOPS /GOPS	L2ビジー率	メモリビジー率	L1Dミス数	L2ミス数
倍精度実数	連続ロードx2+連続ストアx1	211.0	17.6	94.1%	99.9%	2.95E+08	2.60E+08
	Gatherロードx2+連続ストアx1	211.3	17.6	94.4%	95.6%	2.95E+08	2.43E+08
	連続ロードx2+Scatterストアx1	159.8	13.3	85.1%	87.0%	2.95E+08	3.12E+08
	Gatherロードx2+Scatterストアx1	159.8	13.3	84.5%	86.8%	2.95E+08	3.12E+08
単精度実数	連続ロードx2+連続ストアx1	211.1	35.2	93.9%	99.8%	2.95E+08	2.59E+08
	Gatherロードx2+連続ストアx1	211.3	35.2	94.0%	92.4%	2.95E+08	2.32E+08
	連続ロードx2+Scatterストアx1	159.7	26.6	84.8%	86.0%	2.95E+08	3.08E+08
	Gatherロードx2+Scatterストアx1	121.2	20.2	48.4%	63.3%	2.95E+08	2.96E+08

倍精度実数ならばメモリスループットネックになる。

評価結果：メモリスループット(zfill なし)





型	ロード・ストア	スループット (GB/s)	GFLOPS /GOPS	L2ビジー率	メモリビジー率	L1Dミス数	L2ミス数
倍精度実数	連続ロードx2+連続ストアx1	159.9	13.3	86.0%	95.8%	2.95E+08	3.54E+08
	Gatherロードx2+連続ストアx1	159.8	13.3	85.2%	87.4%	2.95E+08	3.15E+08
	連続ロードx2+Scatterストアx1	159.1	13.3	85.3%	86.5%	2.95E+08	3.12E+08
	Gatherロードx2+Scatterストアx1	158.6	13.2	84.5%	86.1%	2.95E+08	3.12E+08
単精度実数	連続ロードx2+連続ストアx1	159.8	26.6	85.9%	95.7%	2.95E+08	3.54E+08
	Gatherロードx2+連続ストアx1	158.1	26.4	84.6%	85.7%	2.95E+08	3.11E+08
	連続ロードx2+Scatterストアx1	158.2	26.4	84.5%	85.1%	2.95E+08	3.08E+08
	Gatherロードx2+Scatterストアx1	121.2	20.2	48.4%	63.2%	2.95E+08	2.96E+08

倍精度実数ならば全アクセスパターン、メモリスループットネックになる。

FUJITSU

アライメントの変化による性能影響

- 測定条件
- 測定結果：連続SIMDロード
- 測定結果：連続SIMDストア
- 測定結果：Gatherロード
- 測定結果：Scatterストア
- 測定結果：構造体ロード(LD2命令)
- 測定結果：構造体ストア(ST2命令)

測定条件

FUJITSU

- 測定概要

  以下のデータアクセスにて、アライメントの変化による影響を検証する。

  - 連続SIMDロード
  - 連続SIMDストア

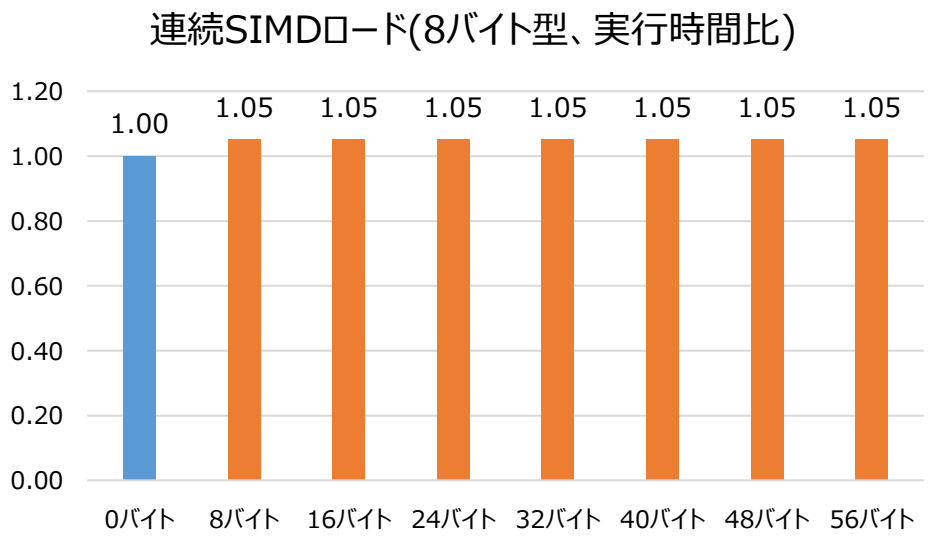
- 測定条件
  - 256バイトでアラインされた配列を0～63バイト目からアクセスし、性能を測定する。
  - 測定データ型は以下。

    8バイト型(倍精度実数・８バイト整数)、4バイト型(単精度実数・4バイト整数)、2バイト型(半精度実数(FP16)・2バイト整数)、1バイト型(1バイト整数)
  - 測定コードは、測定結果と併せて後述する。
  - 最内ループ(n)はL1キャッシュサイズの1/2をアクセスする配列サイズ・繰返し数、外側ループ(iter)は1000000で評価。
  - 1コア実行(逐次実行)で測定。
  - Multiple Structures命令の評価では、SWPL,ループアンローリングは抑止(アウトオブオーダーによるスケジューリングのみ)

測定結果：連続SIMDロード (1/2)

FUJITSU

- 8バイト型(倍精度実数・8バイト整数)

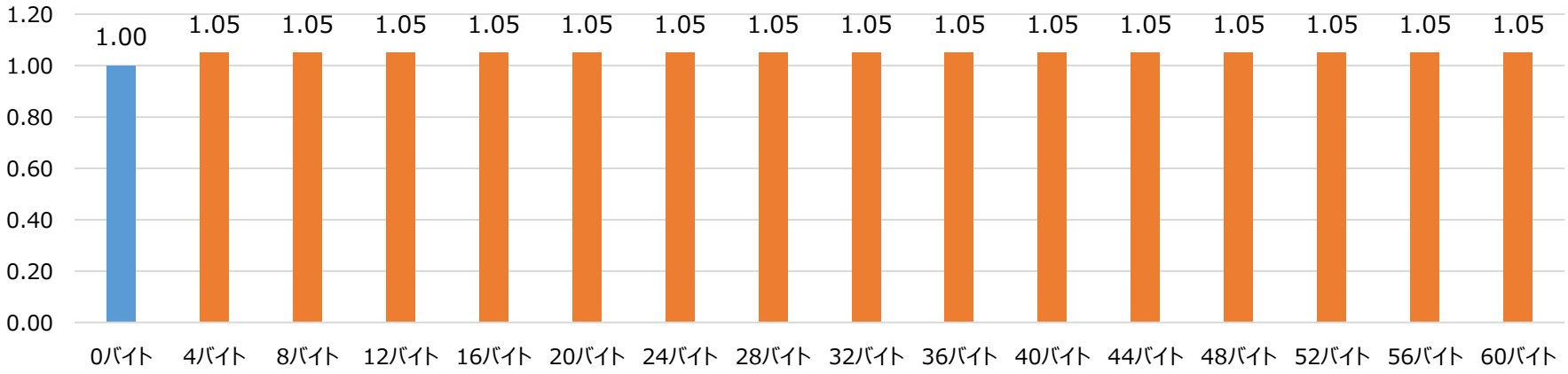


測定コード

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

アセンブリコードからストア命令を削除する

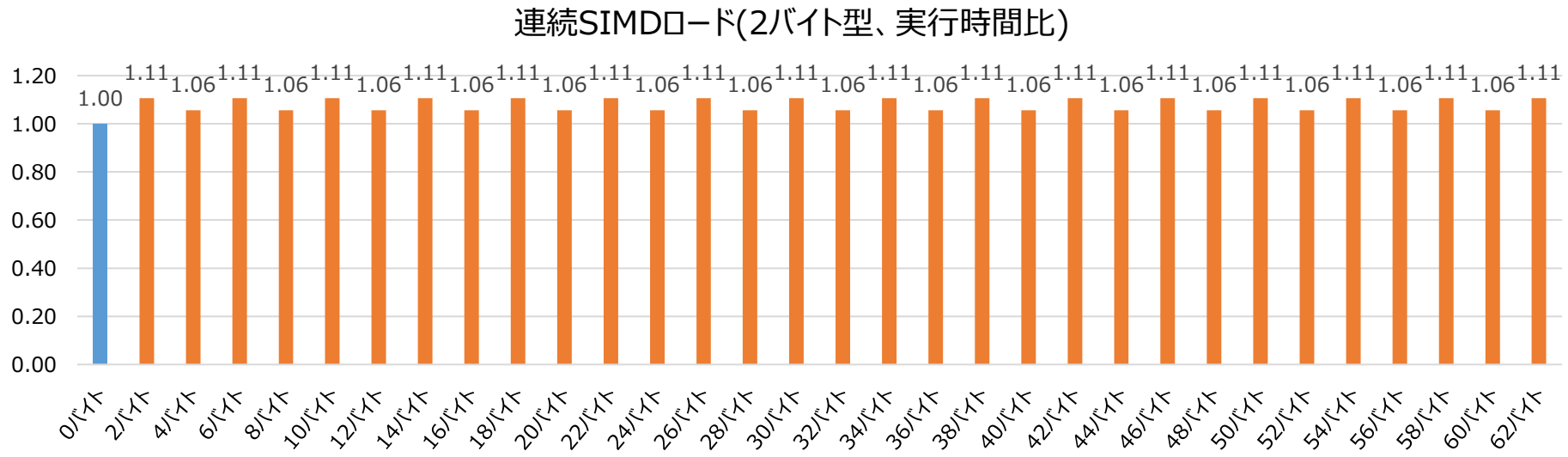
- 4バイト型(単精度実数・4バイト整数)



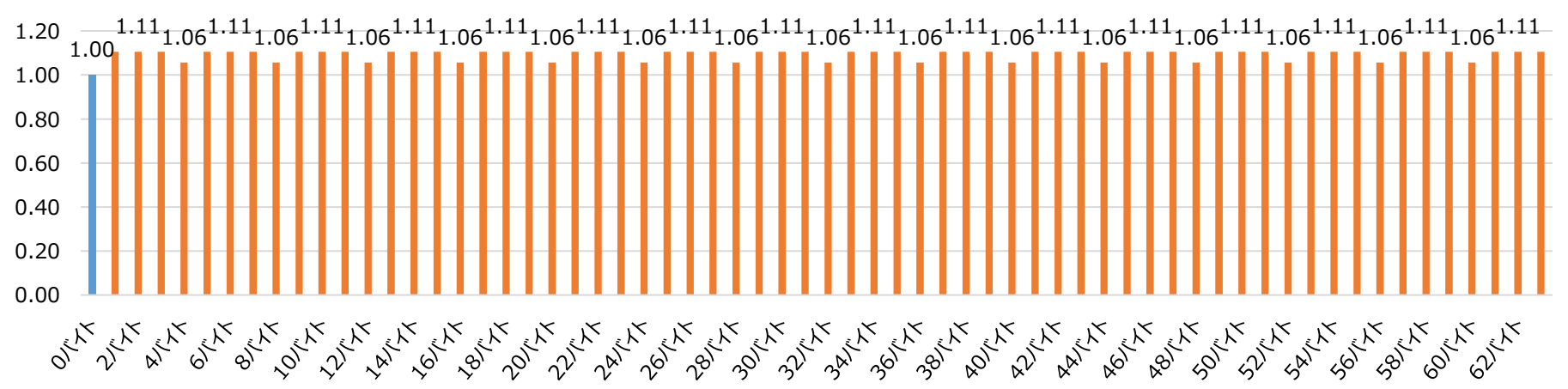
測定結果：連続SIMDロード (2/2)

FUJITSU

- 2バイト型(半精度実数(FP16)・2バイト整数)



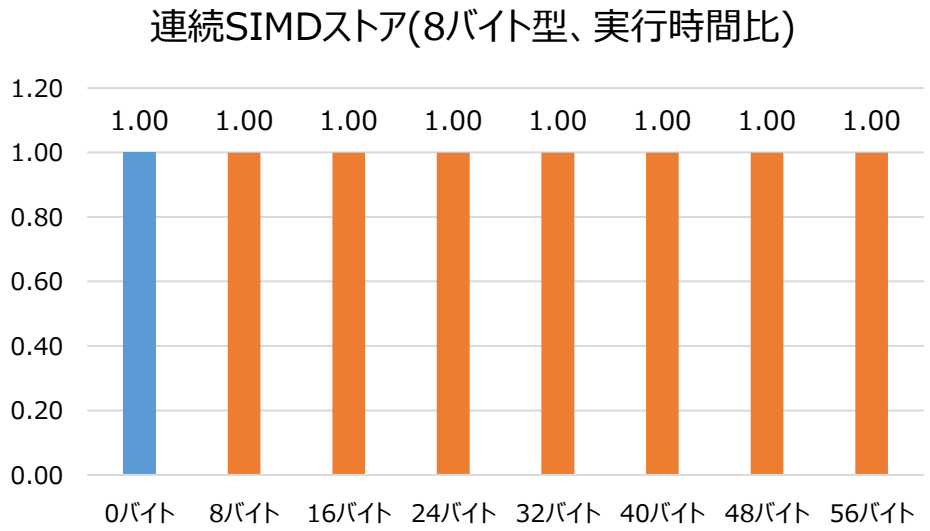
- 1バイト型(1バイト整数)



測定結果：連続SIMDストア (1/2)

FUJITSU

- 8バイト型(倍精度実数・8バイト整数)

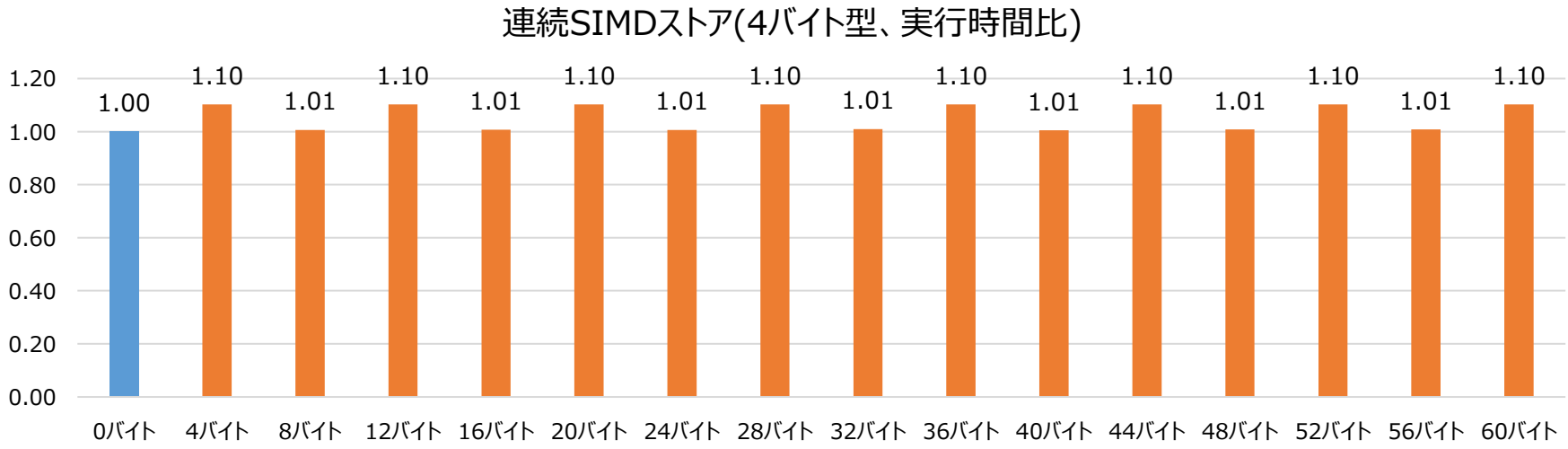


測定コード

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

アセンブリコードからロード命令を削除する

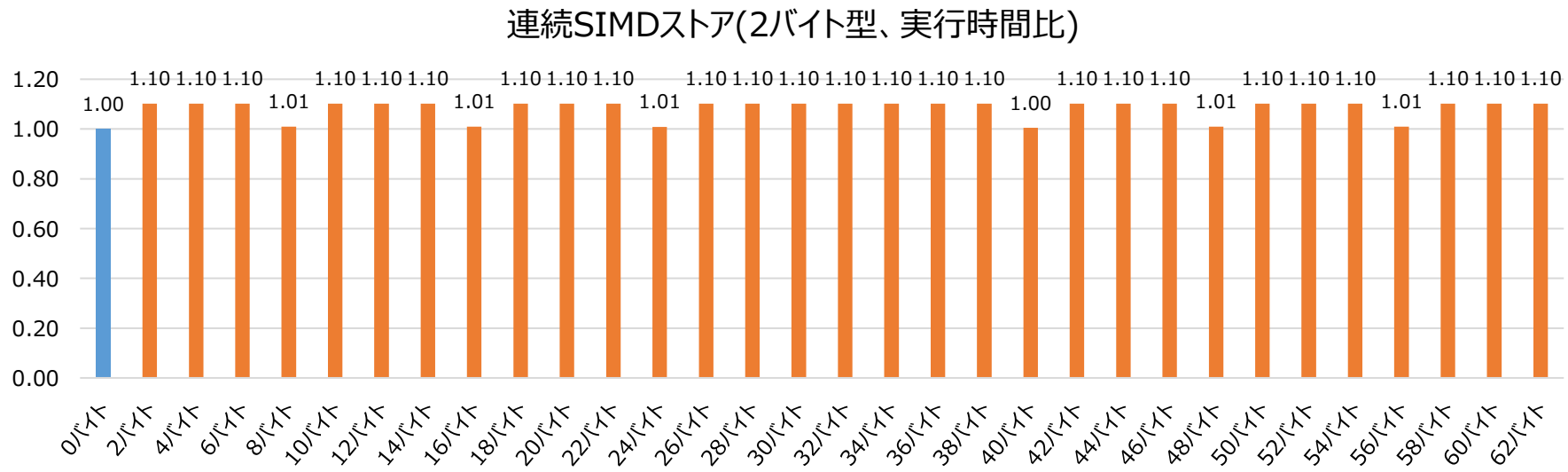
- 4バイト型(単精度実数・4バイト整数)



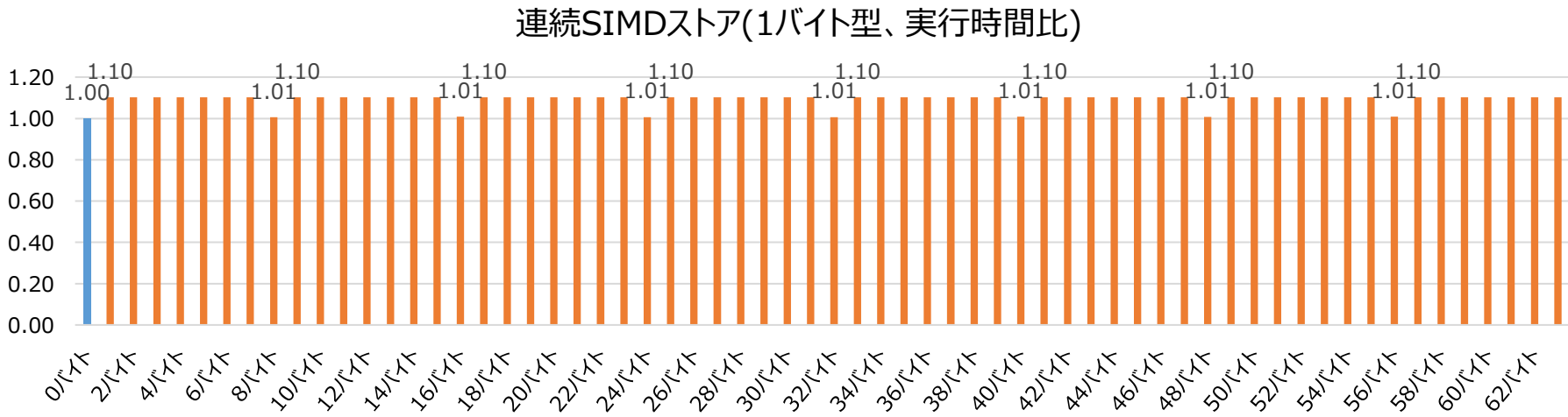
測定結果：連続SIMDストア (2/2)

FUJITSU

- 2バイト型(半精度実数(FP16)・2バイト整数)



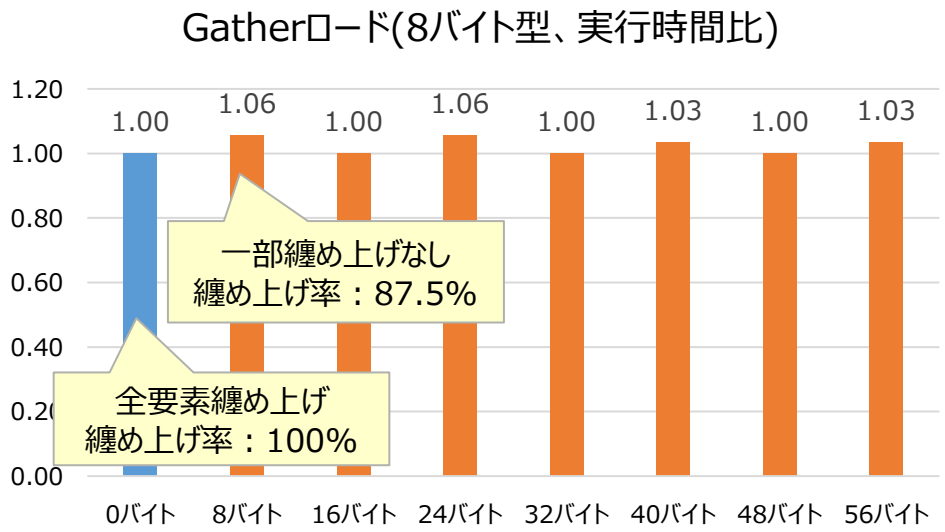
- 1バイト型(1バイト整数)



測定結果：Gatherロード

FUJITSU

8バイト型(倍精度実数・8バイト整数)

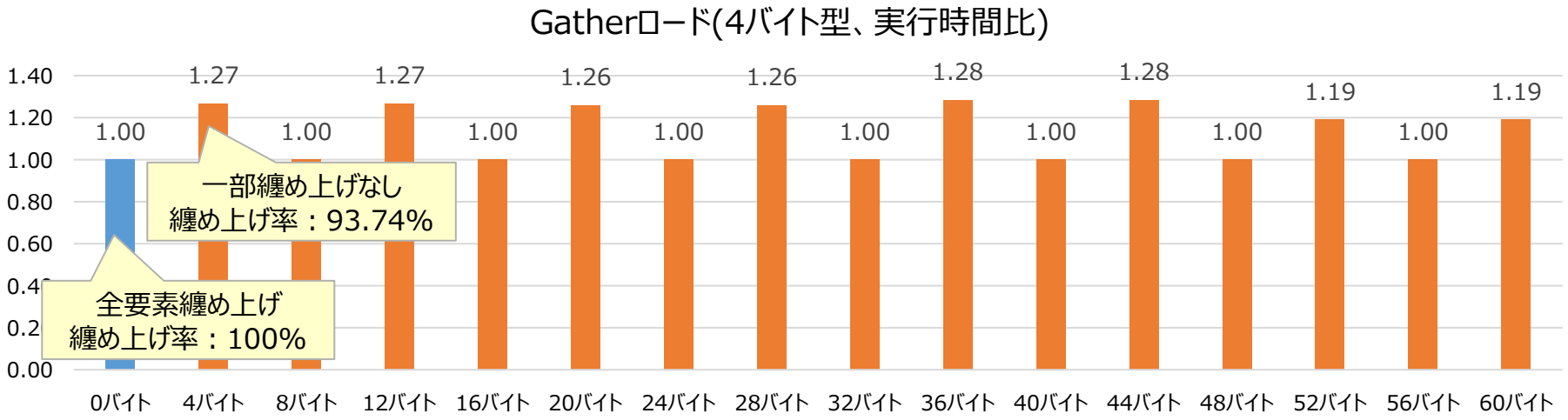


測定コード

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

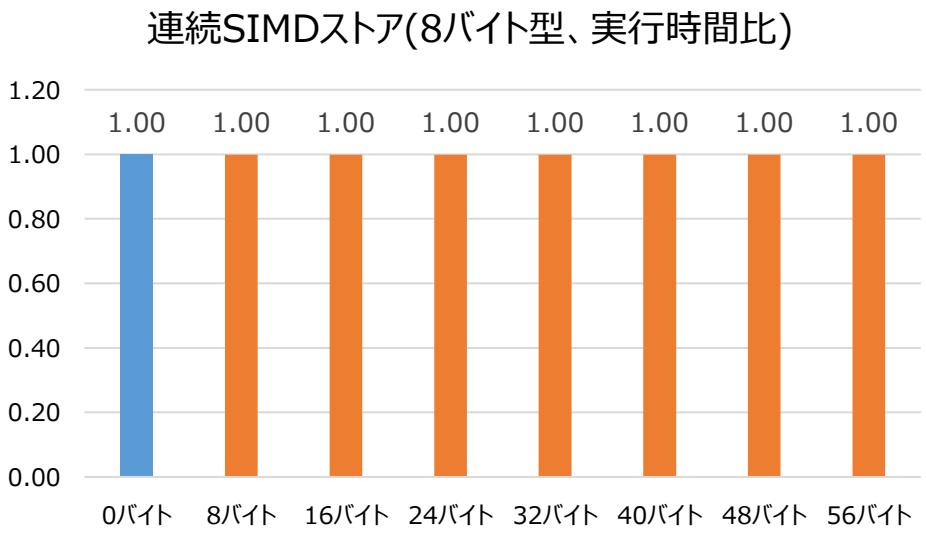
アセンブリコードからストア命令を削除する

4バイト型(単精度実数・4バイト整数)



測定結果：Scatterストア

- 8バイト型(倍精度実数・8バイト整数)

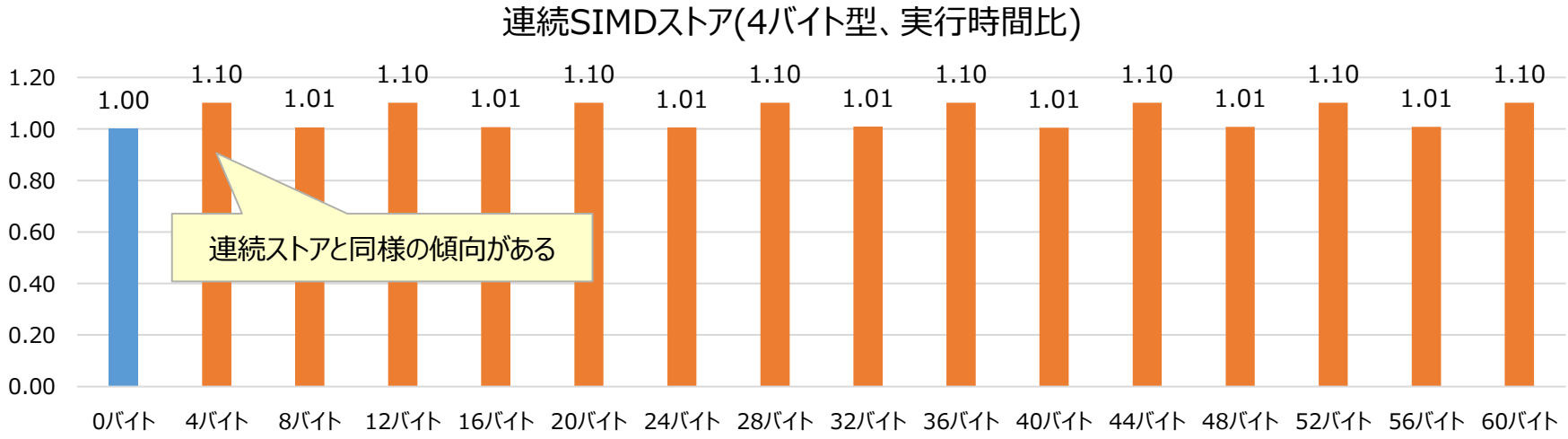


測定コード

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

アセンブリコードから
ロード命令を削除する

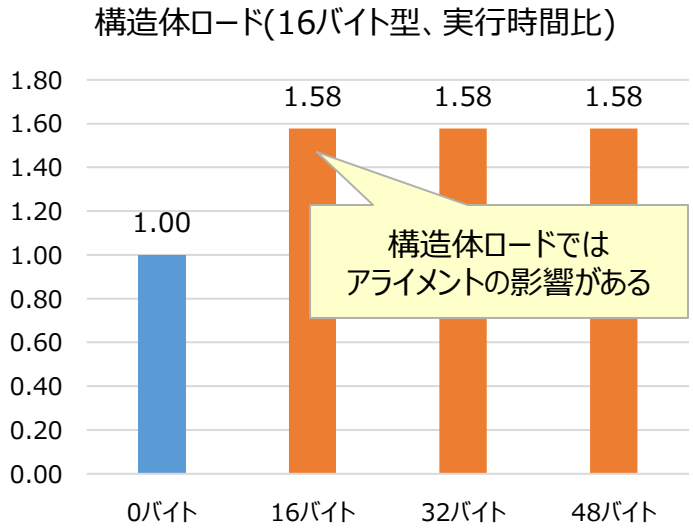
- 4バイト型(単精度実数・4バイト整数)



測定結果：構造体ロード(LD2命令)

FUJITSU

- 倍精度複素数型(16バイトComplex)

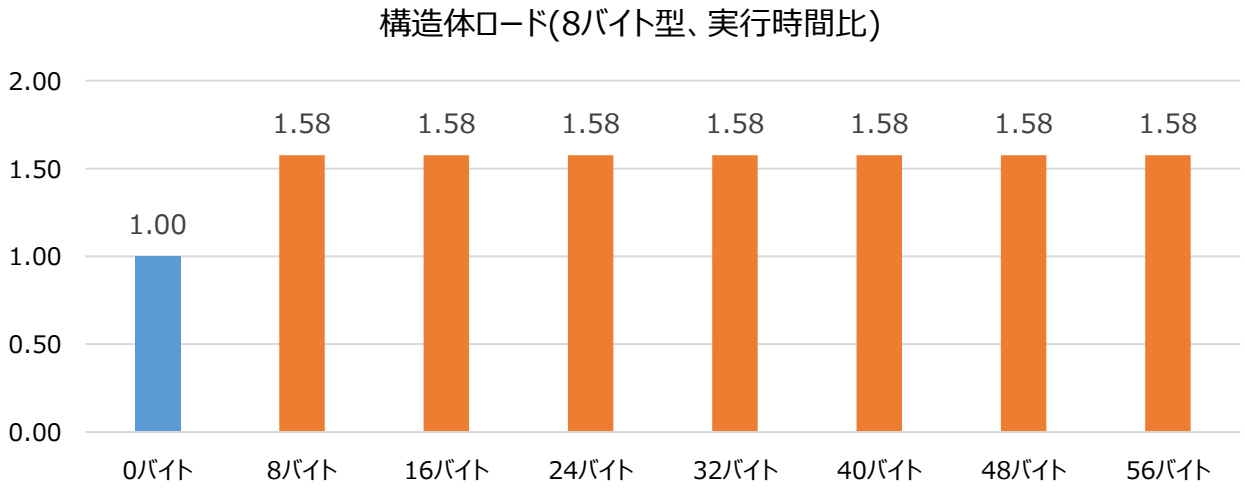


測定コード

```
do k  = 1, iter
   do i = 1,n
      y(i) =  c1 * x1(i)
   enddo
enddo
```

複素数型を使用し、
アセンブリコードからfadd命令
とストア命令を削除する

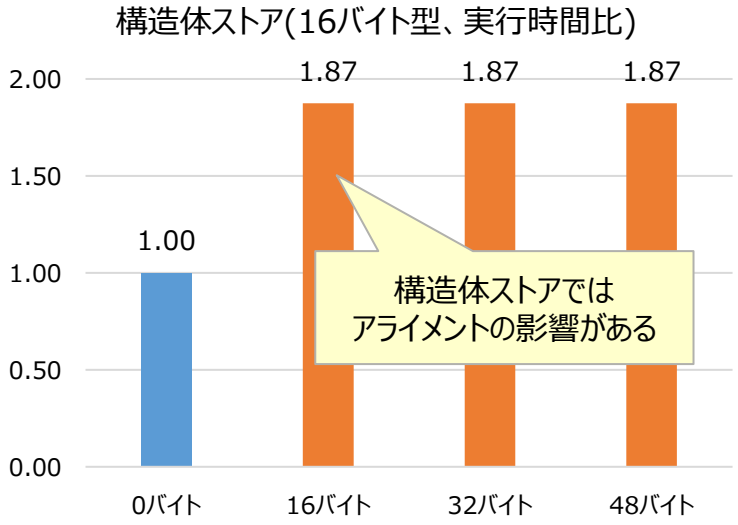
- 単精度複素数型(8バイトComplex)



測定結果：構造体ストア(ST2命令)

FUJITSU

- 倍精度複素数型(16バイトComplex)

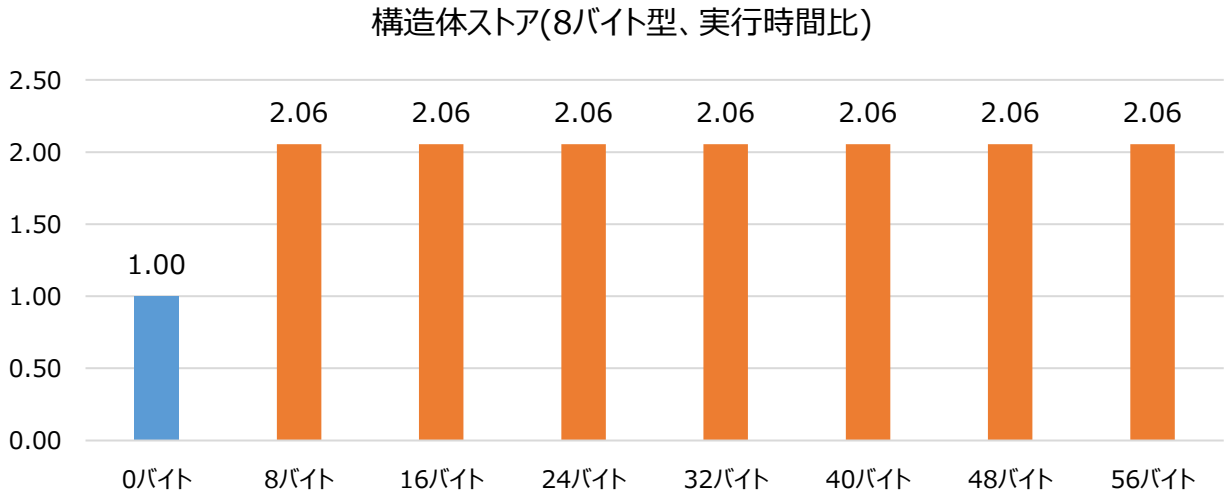


測定コード

```
do k  = 1, iter
   do i = 1,n
      y(i) = c1 * x1(i)
   enddo
enddo
```

複素数型を使用し、
アセンブリコードからfadd命
令とロード命令を削除する

- 単精度複素数型(8バイトComplex)



メモリコピー性能

- 目的
- 測定条件
- 【参考】メモリコピー性能 測定結果
- まとめ

目的

FUJITSU

- 目的
  - Fortranコード、memcpy(glibc、高速版)を使用したメモリコピー性能を評価し、利用シーンを明確にする。

- 高速版memcpyについて
  - オプション指定により使用可。(-Kfast,A64FX,SVE,optlib_string,nolib)
  - MPIでも高速版 memcpy が使用される。

測定条件

■ 測定条件

条件	パターン
検証コード	- Fortran コード　キャッシュクリーン無 - Fortran コード　キャッシュクリーン有 - memcpy(C言語) キャッシュクリーン無 - memcpy(C言語) キャッシュクリーン有
測定スレッド数	- 1スレッド、12スレッド
コンパイラ	- A64FX向けコンパイラ
翻訳オプション	- Fortranコード　　：-Kfast　　※ Fortranコードは、-Kzfill 有も評価 - memcpy default　　：-Kfast - memcpy高速版　　：-Kfast,A64FX,SVE,optlib_string,nolib
アクセス範囲	- 1スレッド　：コピーサイズ 256Byte ～ 1,073,741,824Byte - 12スレッド：コピーサイズ 288Byte ～ 1,073,741,856Byte ※12スレッド測定時の1スレッド辺りのコピーサイズは、上記コピーサイズを÷12 している

キャッシュクリーン 無
　1回のメモリコピー実行毎にキャッシュをクリアしない。
　コピーサイズによってonキャッシュになる。
キャッシュクリーン 有
　1回のメモリコピー実行の毎に、キャッシュをクリアする。
　どのコピーサイズでもメモリアクセスになる。

測定コード

```
double y[n], x1[n]   <- 1スレッド：32～134217728
for (j = 0; j < iter; j++) {
  タイマ計測開始
  memcpy(y, x1, (コピーサイズ)
  タイマ計測終了
  キャッシュクリーン
}
```

Fortranコードは、以下を実行

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

memcpy は、以下の評価を実施
　default (glibc)
　高速版

※default (glibc)はA64FX向け
　高速化前の版で測定

【参考】メモリコピー性能 測定結果 (1/4)

FUJITSU

■ 1スレッド、キャッシュクリーン無

コピーサイズ (Byte)	スループット (GB/sec)			
	Fortran コード	Fortran コード zfill 有	memcpy default	memcpy 高速版
256	5.69	5.39	3.74	4.53
512	11.25	10.14	6.92	8.98
1,024	19.69	17.66	10.34	16.79
2,048	36.57	29.47	15.06	31.75
4,096	63.02	46.55	18.24	54.25
8,192	88.56	38.64	21.33	79.15
16,384	88.80	40.76	22.90	96.95
32,768	79.63	39.27	20.99	76.03
65,536	54.64	44.04	19.51	55.54
131,072	56.17	45.95	19.57	59.82
262,144	57.28	47.08	19.77	62.10
524,288	57.49	47.72	19.85	63.34
1,048,576	57.78	48.02	19.90	64.12
2,097,152	57.87	47.91	19.90	64.44
4,194,304	44.65	41.41	18.52	35.40
8,388,608	49.66	43.83	18.03	35.79
16,777,216	49.71	43.89	18.03	36.01
33,554,432	49.44	43.70	18.00	35.86
67,108,864	45.24	40.59	17.50	33.71
134,217,728	45.29	40.57	17.50	33.76
268,435,456	45.15	40.56	17.48	33.67
536,870,912	44.45	40.49	17.46	33.56
1,073,741,824	44.42	40.50	17.46	33.59

【参考】メモリコピー性能 測定結果 (2/4)

FUJITSU

■ 1スレッド、キャッシュクリーン有

コピーサイズ (Byte)	スループット (GB/sec)			
	Fortran コード	Fortran コード zfill 有	memcpy default	memcpy 高速版
256	2.20	1.36	1.88	2.09
512	4.23	2.43	2.47	3.98
1,024	7.56	4.28	2.81	6.87
2,048	9.92	6.85	3.10	9.66
4,096	16.25	11.74	7.13	16.06
8,192	23.11	19.67	10.79	23.21
16,384	30.01	25.13	10.92	30.62
32,768	39.31	33.13	15.37	39.27
65,536	41.39	36.75	16.22	27.84
131,072	44.36	39.71	17.05	27.84
262,144	45.95	41.46	17.53	27.60
524,288	46.62	42.34	17.77	27.64
1,048,576	47.01	42.84	17.90	27.59
2,097,152	47.24	43.00	17.97	27.62
4,194,304	47.34	43.14	18.00	35.73
8,388,608	47.15	43.00	17.99	35.74
16,777,216	44.64	40.60	17.61	34.00
33,554,432	44.07	39.86	17.49	33.65
67,108,864	44.10	39.87	17.49	33.65
134,217,728	44.14	39.96	17.50	33.72
268,435,456	44.03	39.84	17.48	33.67
536,870,912	43.92	39.46	17.46	33.59
1,073,741,824	43.94	39.73	17.46	33.59

【参考】メモリコピー性能 測定結果 (3/4)

FUJITSU

- 12スレッド、キャッシュクリーン無

コピーサイズ (Byte)	スループット (GB/sec)			
	Fortran コード	Fortran コード zfill 有	memcpy default	memcpy 高速版
288	1.05	1.05	0.77	0.76
576	2.76	2.76	1.57	1.55
1,056	5.82	5.77	2.88	2.87
2,112	11.77	11.42	5.83	5.93
4,128	25.48	25.02	11.20	11.90
8,256	47.86	47.59	22.74	23.76
16,416	88.74	90.45	41.51	41.04
32,832	153.42	160.16	74.70	93.01
65,568	277.83	221.89	120.31	182.64
131,136	467.51	283.54	174.62	315.61
262,176	522.78	298.78	188.62	481.50
524,352	467.34	330.40	200.33	439.71
1,048,608	530.94	362.15	220.41	562.11
2,097,216	570.44	374.27	227.93	601.61
4,194,336	328.48	294.05	189.47	424.89
8,388,672	140.45	206.26	141.31	143.16
16,777,248	140.18	206.39	141.53	143.32
33,554,496	140.23	206.37	141.70	143.61
67,108,896	140.20	206.59	142.16	206.10
134,217,792	140.30	206.83	142.21	206.36
268,435,488	140.21	206.65	142.32	206.51
536,870,976	140.02	206.79	142.38	206.42
1,073,741,856	139.88	206.44	142.45	206.44

【参考】メモリコピー性能 測定結果 (4/4)

FUJITSU

- 12スレッド、キャッシュクリーン有

コピーサイズ (Byte)	スループット (GB/sec)			
	Fortran コード	Fortran コード zfill 有	memcpy default	memcpy 高速版
288	0.82	0.83	0.97	0.93
576	1.51	1.50	1.91	1.68
1,056	2.82	2.81	3.33	3.10
2,112	5.69	5.58	6.09	5.71
4,128	11.01	11.05	10.41	10.49
8,256	21.44	20.46	17.33	21.50
16,416	36.04	35.88	25.45	36.04
32,832	61.95	55.93	33.42	60.97
65,568	86.67	75.37	44.53	87.19
131,136	109.33	116.36	65.06	101.73
262,176	123.00	144.29	92.06	121.89
524,352	133.54	167.15	102.95	118.93
1,048,608	141.42	185.02	120.94	130.78
2,097,216	137.25	201.39	132.87	139.55
4,194,336	137.96	197.40	133.33	138.75
8,388,672	138.41	204.53	138.73	141.53
16,777,248	138.96	204.34	139.54	142.39
33,554,496	140.32	205.50	141.09	143.06
67,108,896	139.77	206.85	141.68	204.27
134,217,792	140.28	206.23	141.84	205.77
268,435,488	139.95	206.84	141.98	205.82
536,870,976	140.20	205.95	142.16	206.01
1,073,741,856	139.85	205.48	142.27	206.03

まとめ

FUJITSU

- memcpy 高速版
  - どのケースにおいても、memcpy default (glibc版)より速い。
    → memcpy は高速版の使用を推奨
  - (1スレッドあたりの)コピーサイズが 4MiB を超えると zfill が自動で有効になる。
    - 1スレッド実行時(非メモリビジー時)は、性能が低下するため注意が必要。
    - 12スレッド実行時(メモリビジー時)は、性能が向上する。
      → コピーサイズによっては、メモリビジーでもzfillが有効にならないケースあり。
- Fortranコード
  - memcpy 高速版と同等以上の性能だが、zfill は手動で設定する必要がある。
  - memcpy 高速版より速いケース
    - 非メモリビジー時は、コピーサイズが 4MiB～1GiBの 場合、Fortranコードが速い。
    - メモリビジー時は、Fortranコード zfill 有が速い。
      (memcpy 高速版の zfill 無効時と比較)

CMG間性能の評価

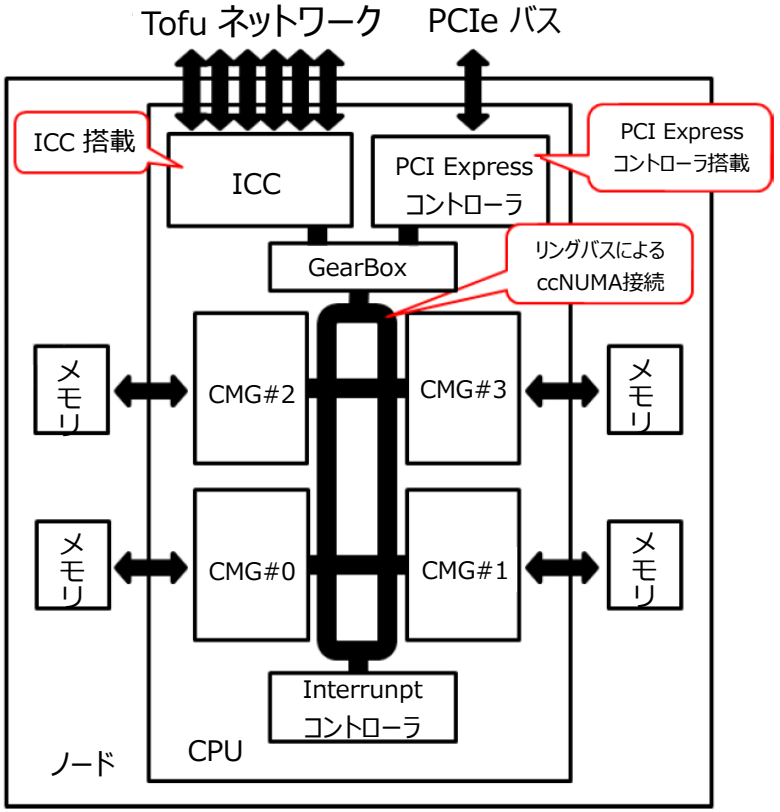
- ハードウェアの概要
- numactl の指定(コア・メモリ・インターリーブ)について
- スループット測定 測定条件
- スループット測定 測定結果 (12スレッド実行)
- 測定結果 (48スレッド実行)
- レイテンシ測定 測定条件
- レイテンシ測定 測定結果
- まとめ
- DGEMMのCMG跨ぎ A64FXの実装
- 複数CMGでのDGEMMの効率

ハードウェアの概要

- メモリ(HBM)
  メモリは、HBM(High Bandwidth Memory) 4個をCPU LSIに直接接続する。
  1個のHBMは最大転送速度2Gbpsのデータインターフェースを1024ビット持っている。メモリ帯域は1024ビット(双方)×2Gbps×4 (HBM)の1024GB/sを有している。

- CMG 間接続
  CMG 間を接続しているチップ内ネットワーク構成を示す。以下の6点を接続する双方向リングバス方式のネットワークである。
  - 4つのCMG
  - インターコネクトコントローラ(ICC)・PCI Expressコントローラ
  - 割り込みコントローラ

  リングバスは、64バイトバスが2本(双方向)あり、リングバスの転送能力は128GB/s×2(双方向)である。



numactl の指定(コア・メモリ・インターリーブ)について

FUJITSU

- numactl の指定について

  numactl コマンドでは以下を使用します。

  - コアの指定　：　-C
  - メモリの指定　：　-m
  - インターリーブの指定：　–-interleave

CMG	コア	メモリ
CMG0	12～23	4
CMG1	24～35	5
CMG2	36～47	6
CMG3	48～59	7

numactl の例)

48スレッド(48コア)、すべてのメモリを使用する例、指定の数値は右記を参照。

```
numactl –C12-59 –m4-7
```

スループット測定 測定条件

FUJITSU

測定条件

	パターン
測定コード	- Triad メモリアクセス
測定コア数、 メモリ設定	12コア実行(CMG0) - CMG内メモリ使用 - CMG間メモリ使用 48コア実行(CMG0～3) - デマンドページング - プリページング - インタリーブ
翻訳オプション	-Kfast,openmp,zfill
アクセス範囲	- アクセス配列の合計バイト数：240MiB - bss を使用 - 演算配列には倍精度型を使用 - 最内ループ繰返し数、配列サイズ(n) ：10485120 外側ループ繰返し数(iter) ： 3
測定数値	- メモリスループット(GB/s) ※ メモリアクセス量 / 計測時間 で算出。 - メモリスループットピーク比(%) ※ 分母は12スレッド実行で256GB/s、48スレッド実行で1024GB/s

測定コード

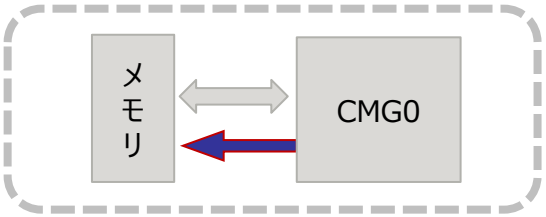
```
!$omp parallel
  do j = 1, iter
!$omp do
      Do i = 1, n
         y(i)=x1(i) + c0 * x2(i)
      End Do
!$omp end do nowait
  enddo
!$omp end parallel
```

スループット測定 測定結果 (12スレッド実行)

FUJITSU

CMG内メモリ使用

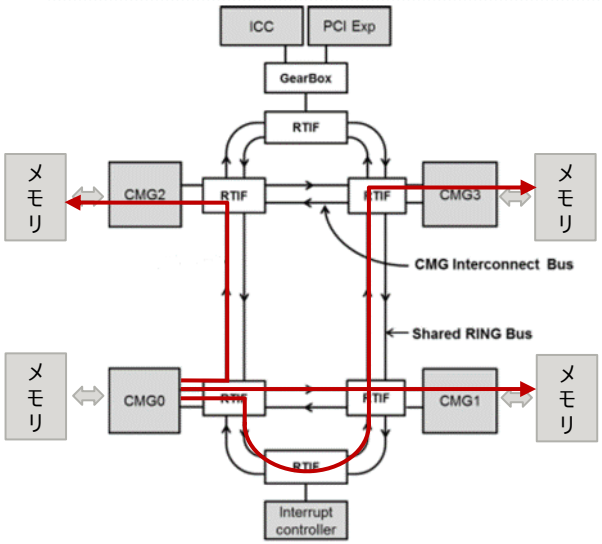
numactl –C12-23 –m4 (CMG0のメモリにアクセス)



実行スレッド数	実行コア	指定メモリ	スループット (GB/s)	メモリスループットピーク比 (%)
12	12～23 (CMG0)	4 (CMG0)	210.8	82.3%

CMG間メモリ使用

numactl –C12-23 –m5～7 (CMG1～3のメモリにアクセス)



実行スレッド数	実行コア	指定メモリ	スループット (GB/s)	メモリスループットピーク比 (%)
12	12～23 (CMG0)	5 (CMG1)	118.5	46.3%
12	12～23 (CMG0)	6 (CMG2)	120.8	47.2%
12	12～23 (CMG0)	7 (CMG3)	120.9	47.2%

測定結果 (48スレッド実行)

FUJITSU

- 4CMGメモリの使用
  - デマンドページング

export XOS_MMM_L_PAGING_POLICY=demand:demand:demand
numactl –C12-59 –m4-7

実行スレッド数	実行コア	指定メモリ	スループット (GB/s)	メモリスループットピーク比 (%)
48	12～59 (CMG0～3)	4～7 (CMG0～3)	819.7	80.0%

  - プリページング

export XOS_MMM_L_PAGING_POLICY=prepage:prepage:prepage
numactl –C12-59 –m4-7

実行スレッド数	実行コア	指定メモリ	スループット (GB/s)	メモリスループット ピーク比 (
48	12～59 (CMG0～3)	4～7 (CMG0～3)	99.3	9.7%

メモリ指定は4~7としているが、宣言がCMG0での実行となり、配列すべてがCMG0のメモリ(4)に割り当たってしまっているため性能がでない

  - インターリーブ

numactl –C12-59 --interleave=4-7

実行スレッド数	実行コア	指定メモリ	スループット (GB/s)	メモリスループット ピーク比 (
48	12～59 (CMG0～3)	インタリーブ (CMG0～3)	476.0	46.5%

アプリの特性上、CMG毎でデータ分割ができない場合、インターリーブを使う

- 1CMGメモリの使用(参考)

numactl –C12-59 –m4

実行スレッド数	実行コア	指定メモリ	スループット (GB/s)	メモリスループットピーク比 (%)
48	12～59 (CMG0～3)	4 (CMG0)	93.2	9.1%

レイテンシ測定 測定条件

FUJITSU

測定条件

	パターン
測定コード	メモリアクセスレイテンシ測定
測定コア	- 1コア実行×48コア分 (CMG0～3の全計算コア) ⇒ CMG内のメモリへのアクセス ⇒ CMG0のメモリへのアクセス(CMG1～3のコアはCMG外アクセス)
翻訳オプション	-Kfast
アクセス範囲	- bss を使用 - 内側ループ繰返し数(NL) : 1024 - 外側ループ繰返し数(rep) : 1
計測数値	アクセスレイテンシ(サイクル数)

測定コード

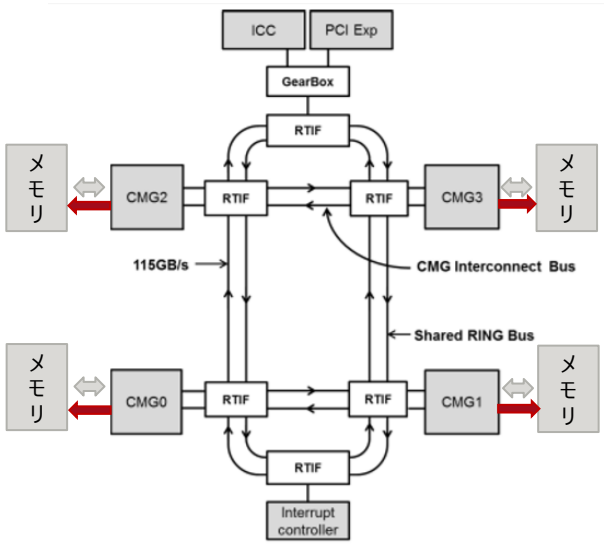
```
for (i = 0; i < rep; i++) {
    p = 0;
    for (j = 0; j < NL; j++) {
       p = array[p];
    }
    ans = p;
}
```

レイテンシ測定 測定結果 (1/2)

FUJITSU

■ 測定結果 (メモリアクセス①)

numactl –m(4~7)
各CMGのメモリにアクセス



CMG	コア	レイテンシ (サイクル)	CMG平均 (サイクル)
0	12	259.86	261.87
	13	257.17	
	14	259.87	
	15	259.06	
	16	262.60	
	17	259.06	
	18	262.42	
	19	262.22	
	20	263.14	
	21	265.98	
	22	262.56	
	23	268.71	
1	24	256.65	260.75
	25	256.70	
	26	259.86	
	27	256.03	
	28	262.19	
	29	259.02	
	30	262.22	
	31	262.86	
	32	263.14	
	33	262.73	
	34	262.48	
	35	265.35	

131ns

CMG	コア	レイテンシ (サイクル)	CMG平均 (サイクル)
2	36	262.19	264.54
	37	262.92	
	38	262.89	
	39	262.19	
	40	262.94	
	41	262.94	
	42	263.58	
	43	263.15	
	44	262.88	
	45	265.34	
	46	262.85	
	47	281.17	
3	48	262.19	265.66
	49	259.84	
	50	262.19	
	51	262.95	
	52	262.19	
	53	262.19	
	54	262.28	
	55	263.18	
	56	266.10	
	57	263.58	
	58	281.16	
	59	281.17	

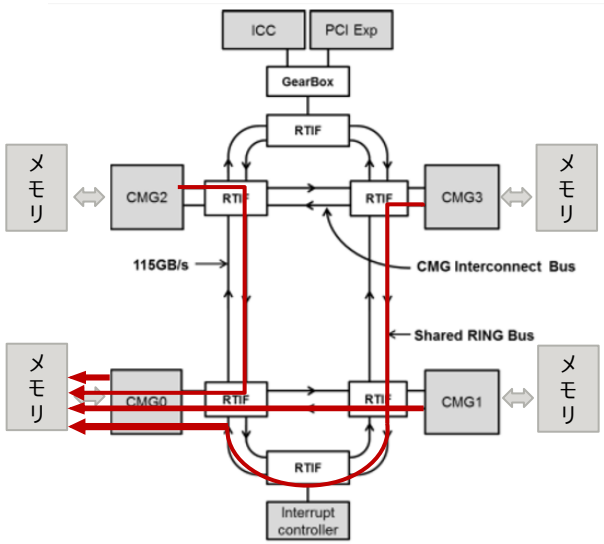
諸元150nsに対して、131nsの計測値
問題なし

レイテンシ測定 測定結果 (2/2)

FUJITSU

■ 測定結果 (メモリアクセス②)

numactl –m4
CMG0のメモリにアクセス



CMGを跨ぐスレッド並列や、
ノード内のMPI通信性能の参考値

CMG	コア	レイテンシ (サイクル)	CMG平均 (サイクル)
0	12	259.86	261.87
	13	257.17	
	14	259.87	
	15	259.06	
	16	262.60	
	17	259.06	
	18	262.42	
	19	262.22	
	20	263.14	
	21	265.98	
	22	262.56	
	23	268.71	
1	24	387.24	389.86
	25	387.88	
	26	388.93	
	27	387.22	
	28	388.93	
	29	387.24	
	30	388.98	
	31	388.96	
	32	393.26	
	33	393.26	
	34	393.26	
	35	393.28	

131ns

CMG	コア	レイテンシ (サイクル)	CMG平均 (サイクル)
2	36	423.55	429.96
	37	423.52	
	38	423.52	
	39	423.54	
	40	424.16	
	41	423.52	
	42	436.25	
	43	436.25	
	44	437.00	
	45	436.25	
	46	436.25	
	47	436.25	
3	48	423.57	429.99
	49	423.52	
	50	423.54	
	51	424.29	
	52	424.10	
	53	423.52	
	54	436.46	
	55	436.43	
	56	436.25	
	57	436.25	
	58	436.25	
	59	436.25	

215ns

まとめ

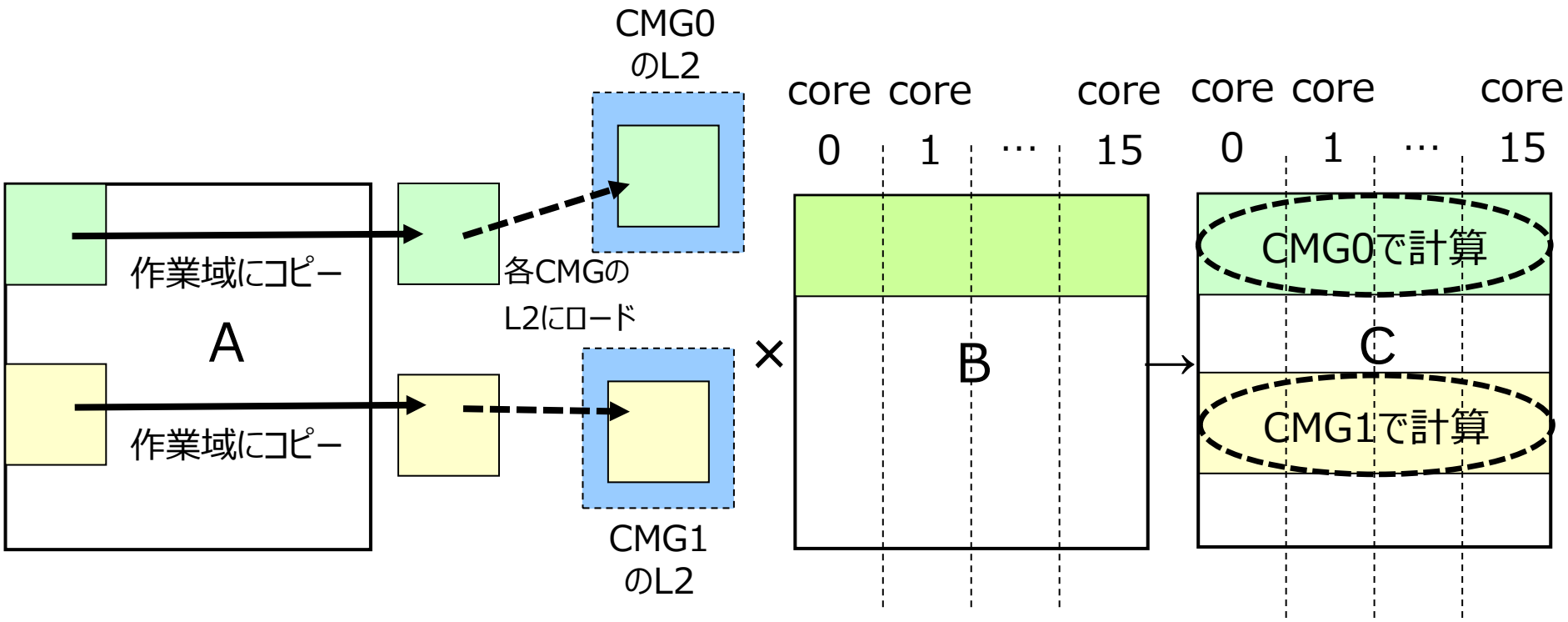
- ローカル（メモリアクセスが自CMGのみ）アクセスの場合
  - 12スレッド・48スレッド実行で効率80%以上の性能が出ており問題なし。
  - 推奨している4プロセス12スレッド実行では問題がない。
- グローバル（メモリアクセスが他CMGにある）アクセスの場合
  - CMG間バス性能（2.0GHzの場合ピーク128GB/S）に律速
  - ただし、１ＣＭＧにのみメモリ配置され、他3CMGからのアクセスの場合は～ 100ＧＢ/S程度となる。
- 48スレッド実行で性能を出せるかはアプリ次第

  → アプリ特性を把握し、インターリーブ実行も検討する必要がある。

DGEMMのCMG跨ぎ A64FXの実装

FUJITSU

- 多数コアのための分割案
  - 行列の分割次元を、CMG単位とコア単位で別にする。

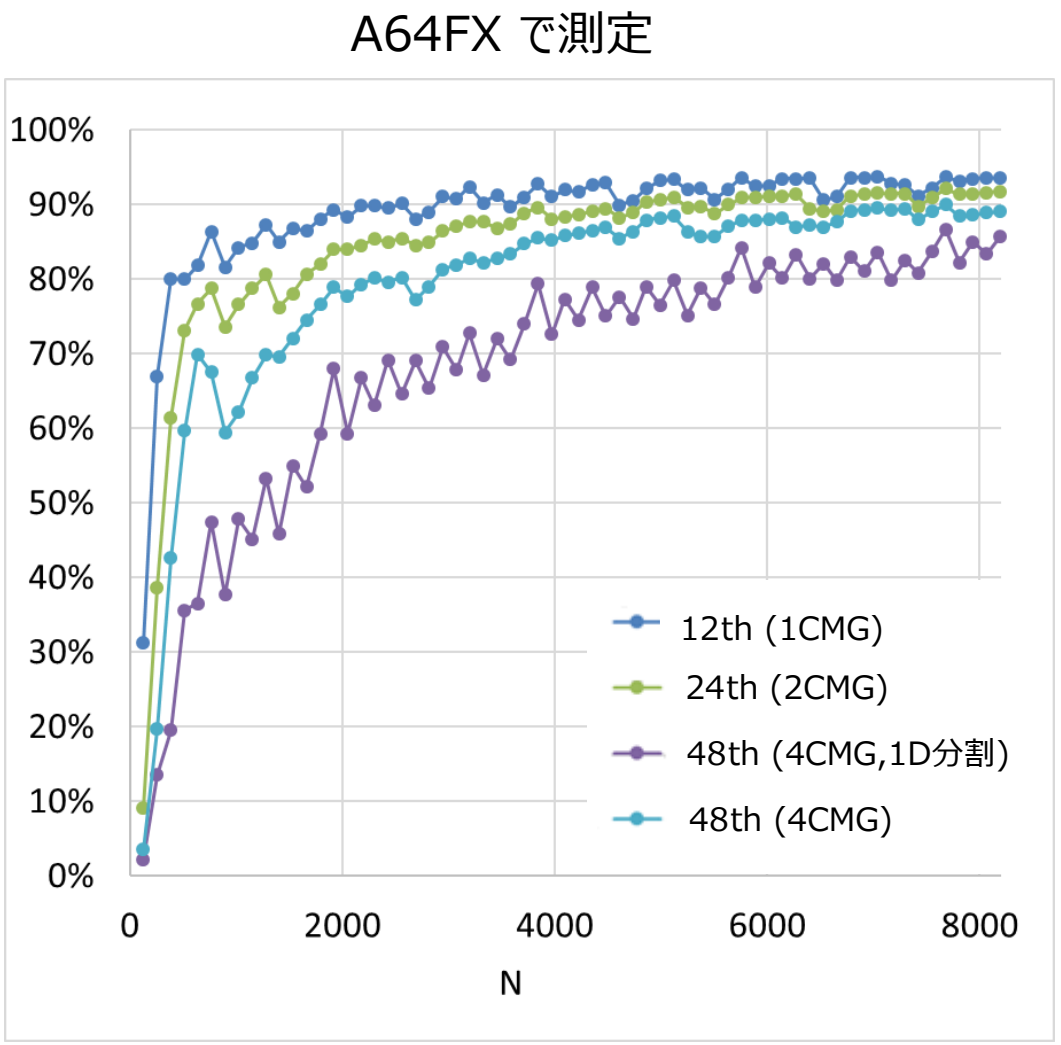


スレッドを1Dに割り当てていたものを、CMG単位とコア単位の2Dに割り当てるようにして、コア間の無駄な要素の移動を削減する。

複数CMGでのDGEMMの効率

FUJITSU

- 大きい部分(N=7～8000)での性能ダウンは、1→2CMGで1.5%、1→4CMGで5%程度
- 1D分割からの改善
  48th 1D分割ではNの辺だけを48分割して細くなる影響もあって、大きく性能が落ちていたが、改善された
  (1D分割でグラフがギザギザしているのは、レジスタブロッキング×スレッド数の余り部分の影響によるもの)
- 改善後も性能が落ちる要因
  - 1CMGあたりのサイズが小さくなることによる性能ダウン
  - 元の行列データがCMGを跨いだところにある場合の性能ダウン



FUJITSU

OpenMPオーバヘッド評価

- ２つのOpenMPライブラリ(LLVM版、富士通版)
- 富士通 OpenMPライブラリの性能
- LLVM OpenMPライブラリと富士通 OpenMPライブラリのOpenMPの基礎性能
- Streamベンチ性能：各コンパイラ×ライブラリ

2つのOpenMPライブラリ(LLVM版、富士通版) (1/2)

FUJITSU

- LLVM OpenMP ライブラリとは、LLVM OpenMP Runtime LibraryをベースにA64FX向けに拡張したOpenMP ライブラリです。

OpenMPライブラリ	オプション	サポート機能
LLVM OpenMPライブラリ	-Nlibomp （デフォルト）	OpenMP 4.5 と 5.0 の一部 ハードウェアバリア（デフォルトはソフトウェアバリア） セクタキャッシュ コアバインド（デフォルト）
富士通 OpenMPライブラリ	-Nfjomplib	OpenMP 3.1 ハードウェアバリア セクタキャッシュ コアバインド（ジョブ実行時はデフォルト）

- コンパイラとの組み合わせ
  - FortranとC/C++のtradモードでは.oは共通で、使用するライブラリを -Nfjomplip / libompオプションで指定可能です。(clangモードの.oが含まれる場合は、libompのみ)

オプション	Fortran	C/C++	
		tradモード	clangモード
-Nlibomp	○	○	○
-Nfjomplib	○	○	×

2つのOpenMPライブラリ(LLVM版、富士通版) (2/2)

FUJITSU

- 選択方法
  - 翻訳時オプション（リンク時）で選択
    - -Nlibomp (デフォルト)：LLVM OpenMPライブラリを使用
    - -Nfjomplib：富士通開発のOpenMPライブラリを使用

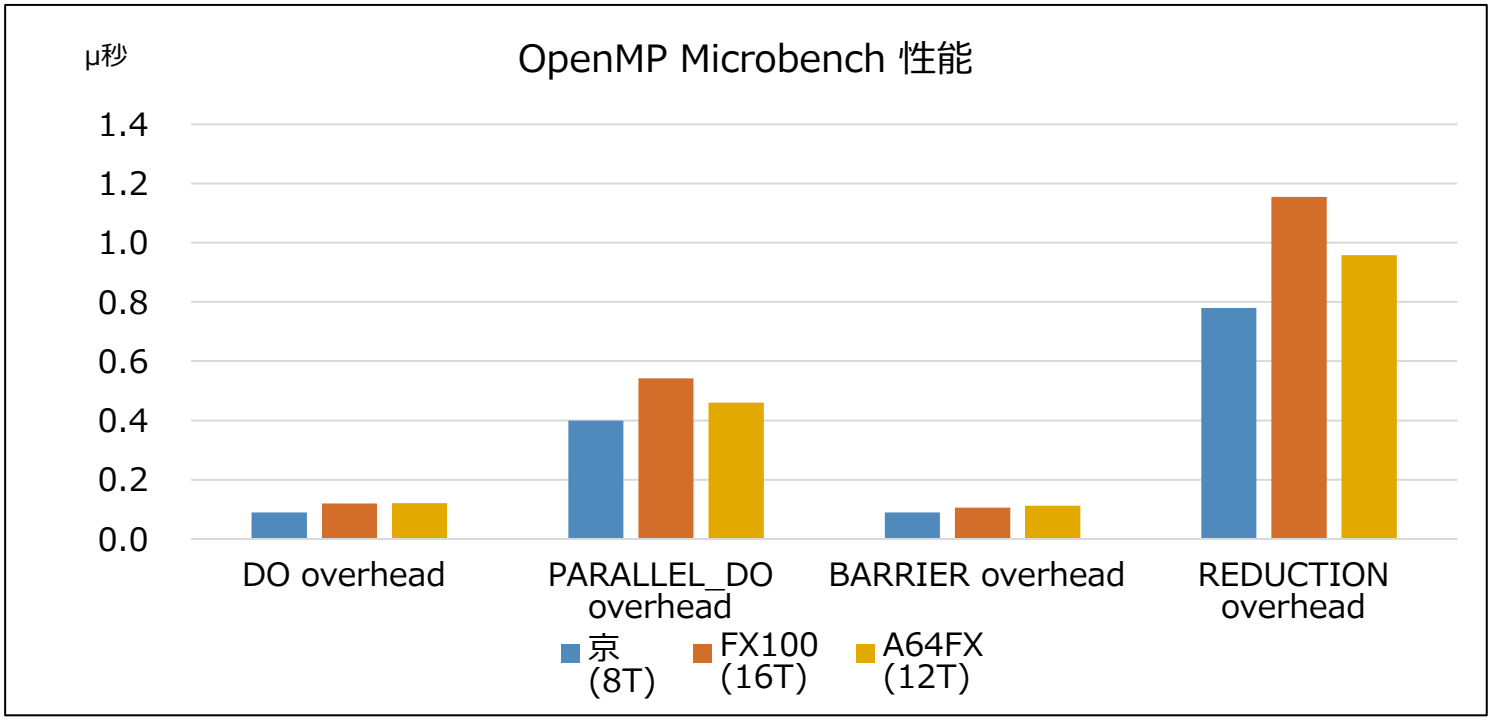
- 仕様の違い
  - スレッドスタックの大きさ

オプション	デフォルトの大きさ	大きさ変更用の環境変数
-Nlibomp	• 8MiB	OMP_STACKSIZE
-Nfjomplib	• プロセススタックのサイズを継承 • プロセススタックサイズがunlimited指定の場合 (メモリサイズ / スレッド数) / 5	OMP_STACKSIZE または THREAD_STACK_SIZE

富士通 OpenMPライブラリの性能 (1/3)

- 代表的なディレクティブ性能比較

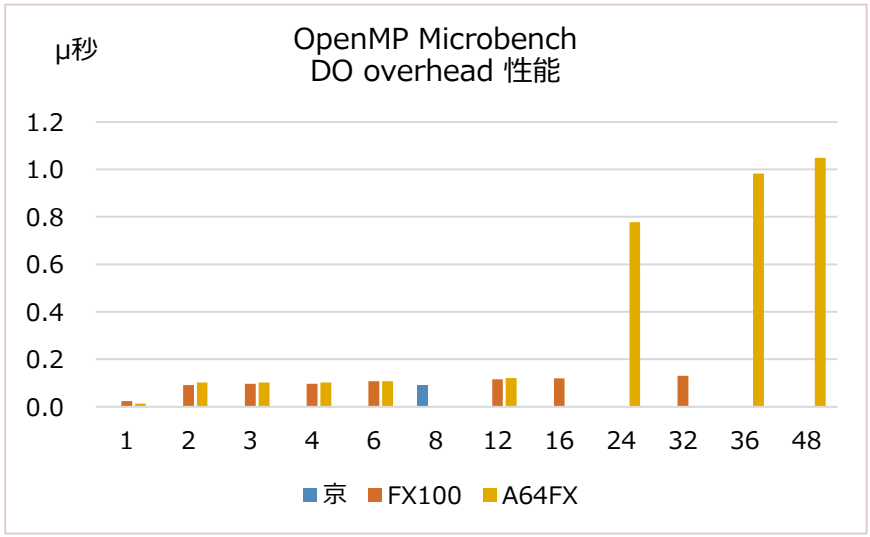
OpenMP Microbench 単位：μ秒	京 (8T)	FX100 (16T)	A64FX (12T)
DO overhead	0.090	0.120	0.121
PARALLEL_DO overhead	0.400	0.542	0.461
BARRIER overhead	0.090	0.106	0.113
REDUCTION overhead	0.780	1.154	0.957

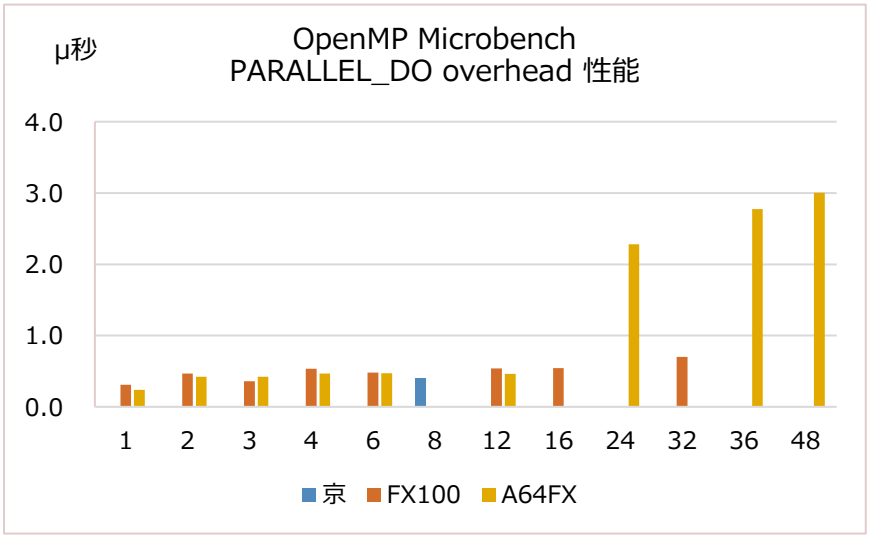


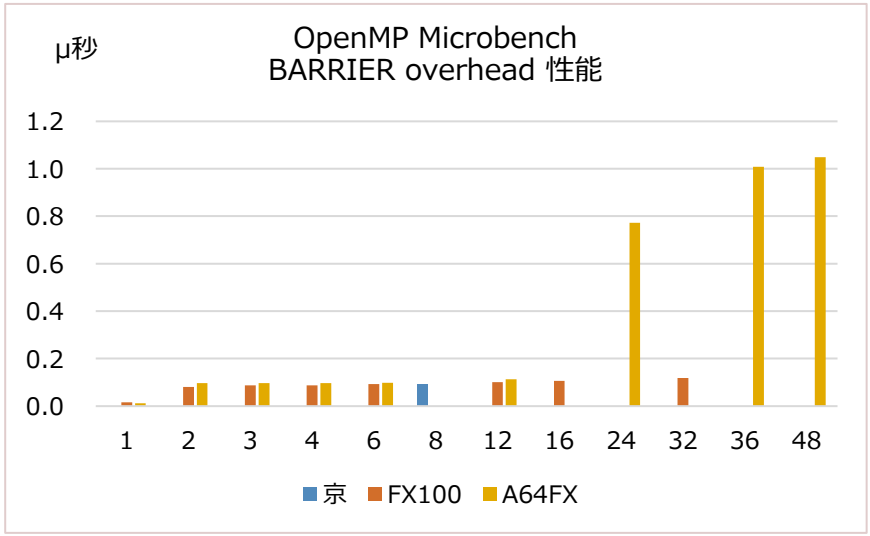
富士通 OpenMPライブラリの性能 (2/3)

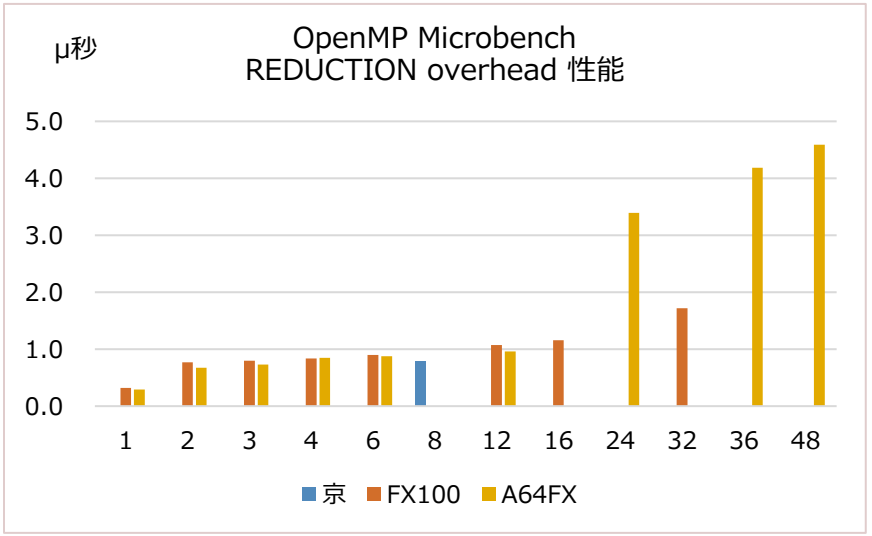
FUJITSU

性能比較 (スレッド数毎)









富士通 OpenMPライブラリの性能 (3/3)

FUJITSU

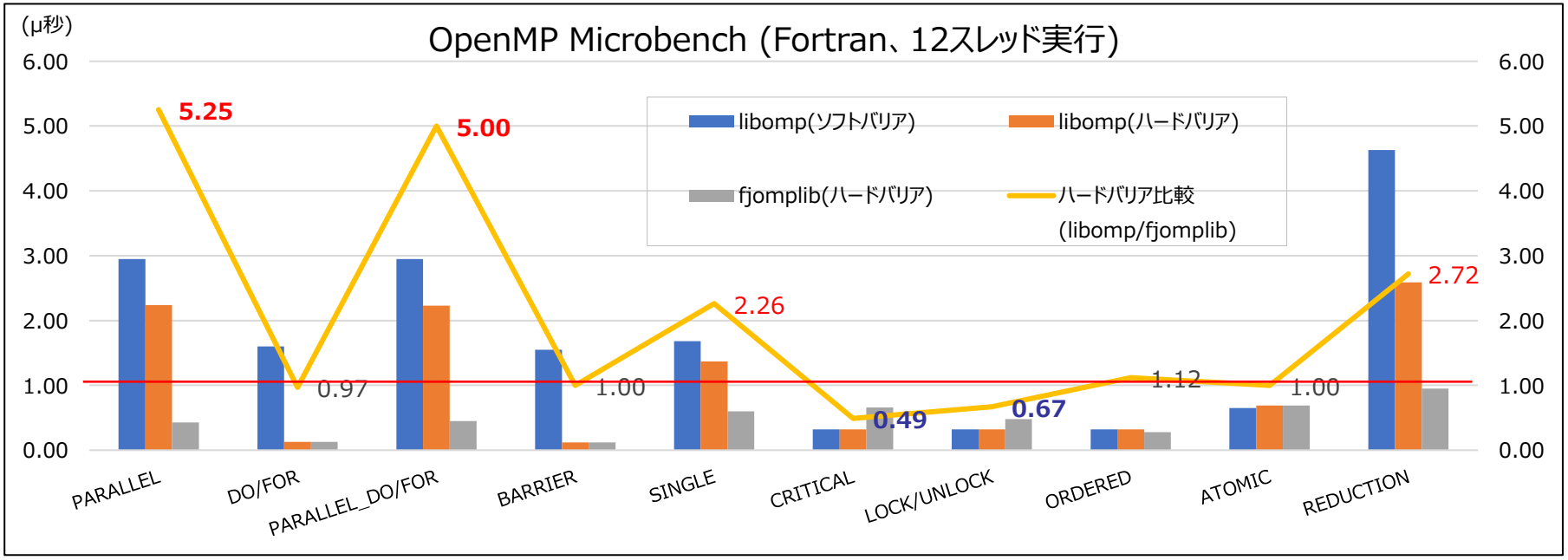
■ 性能比較 (スレッド数毎)

OpenMP Microbench 単位：μ秒		スレッド数											
		1	2	3	4	6	8	12	16	24	32	36	48
DO overhead	京						0.090						
	FX100	0.023	0.092	0.097	0.097	0.107		0.115	0.120		0.130		
	A64FX	0.014	0.103	0.103	0.103	0.108		0.121		0.778		0.983	1.049
PARALLEL_DO overhead	京						0.400						
	FX100	0.309	0.466	0.358	0.535	0.478		0.535	0.542		0.701		
	A64FX	0.236	0.422	0.422	0.464	0.471		0.461		2.283		2.774	3.004
BARRIER overhead	京						0.090						
	FX100	0.016	0.081	0.087	0.088	0.092		0.101	0.106		0.119		
	A64FX	0.012	0.097	0.097	0.097	0.098		0.113		0.773		1.008	1.049
REDUCTION overhead	京						0.780						
	FX100	0.317	0.769	0.794	0.835	0.899		1.073	1.154		1.717		
	A64FX	0.292	0.670	0.732	0.850	0.875		0.957		3.390		4.182	4.589

LLVM OpenMPライブラリと富士通 OpenMPライブラリのOpenMP基礎性能

処理	実行時間(μ秒)			ハードバリア比較(libomp/fjomplib)
	libomp		fjomplib	
	ソフトバリア	ハードバリア	ハードバリア	
PARALLEL	2.95	2.24	0.43	**5.25**
DO/FOR	1.60	0.13	0.13	0.97
PARALLEL_DO/FOR	2.95	2.23	0.45	**5.00**
BARRIER	1.55	0.12	0.12	1.00
SINGLE	1.68	1.37	0.60	2.26
CRITICAL	0.32	0.32	0.66	**0.49**
LOCK/UNLOCK	0.32	0.32	0.48	**0.67**
ORDERED	0.32	0.32	0.28	1.12
ATOMIC	0.65	0.69	0.69	1.00
REDUCTION	4.63	2.59	0.95	2.72

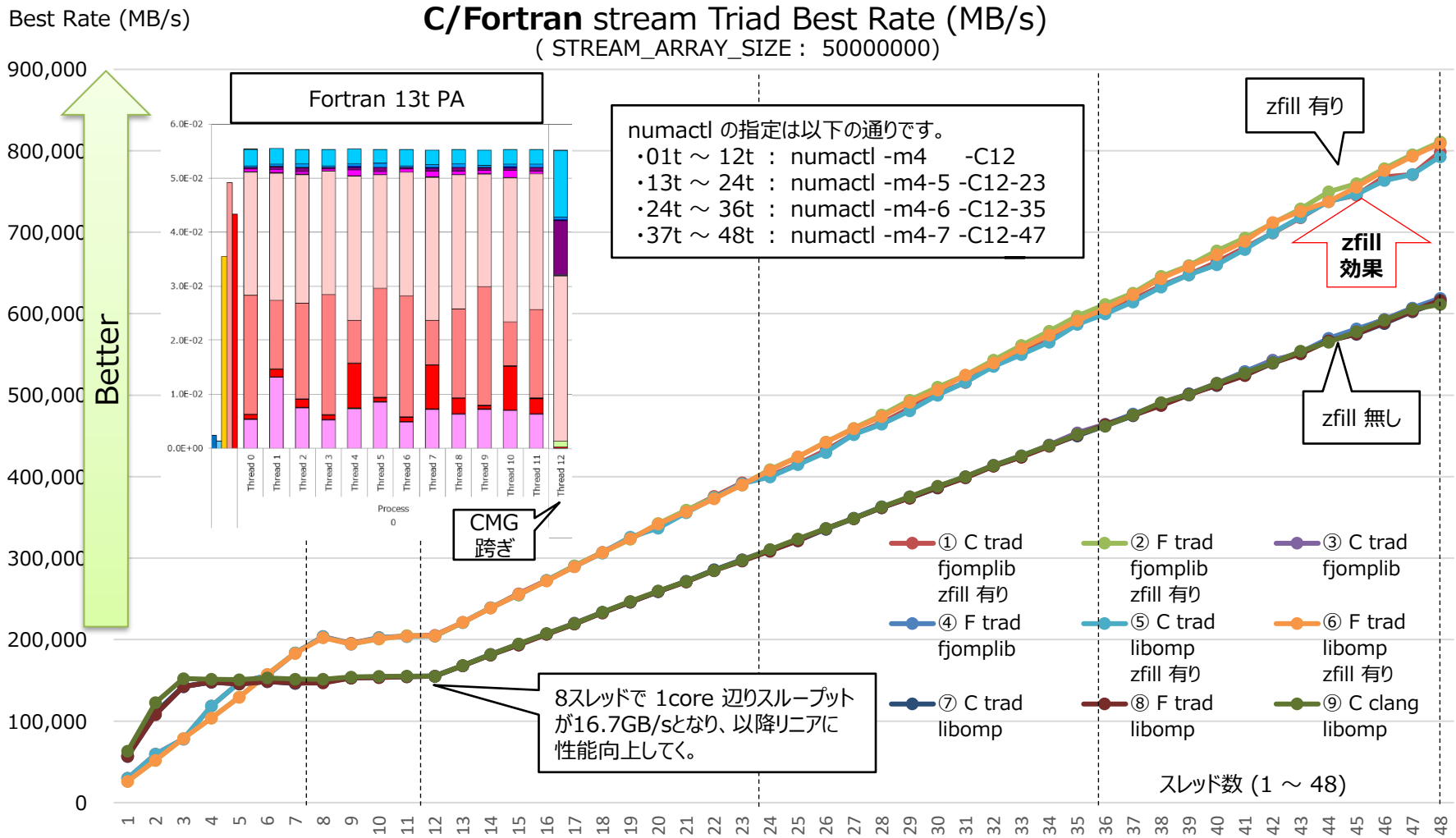
LLVM OpenMPライブラリ(default)使用時、PARALLEL構文は注意が必要



Streamベンチ性能：各コンパイラ×ライブラリ

FUJITSU

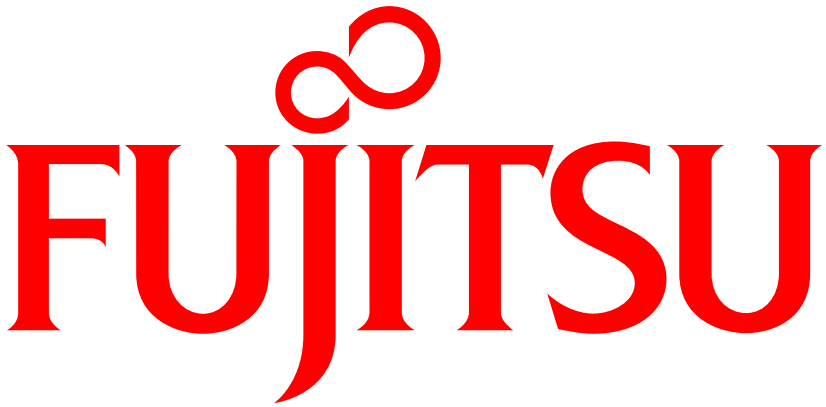
- C(trad/clang) と Fortran の性能比較
  - ➡性能は、同等



更新履歴

■ 更新履歴

版数	発行月	変更理由及び内容
初版(1.0版)	2020/2	・新規作成
1.1版	2020/3	・記事見直しによる誤字の修正
1.2版	2020/9	・記事見直しによる誤字や表現の修正
1.3版	2021/3	・ソフトウェア版数アップによる差分修正、および、記事見直しによる誤字や表現の修正
1.4版	2021/8	・「メモリコピー性能」のglibc記事の修正



shaping tomorrow with you