FUJITSU THE POSSIBILITIES ARE INFINITE

スーパーコンピュータ向けCPU SPARC64™ VIIIfx について

2009年11月25日
富士通株式会社
次世代TC開発本部 PAプロジェクト
青木正樹
(m-aoki@jp.fujitsu.com)

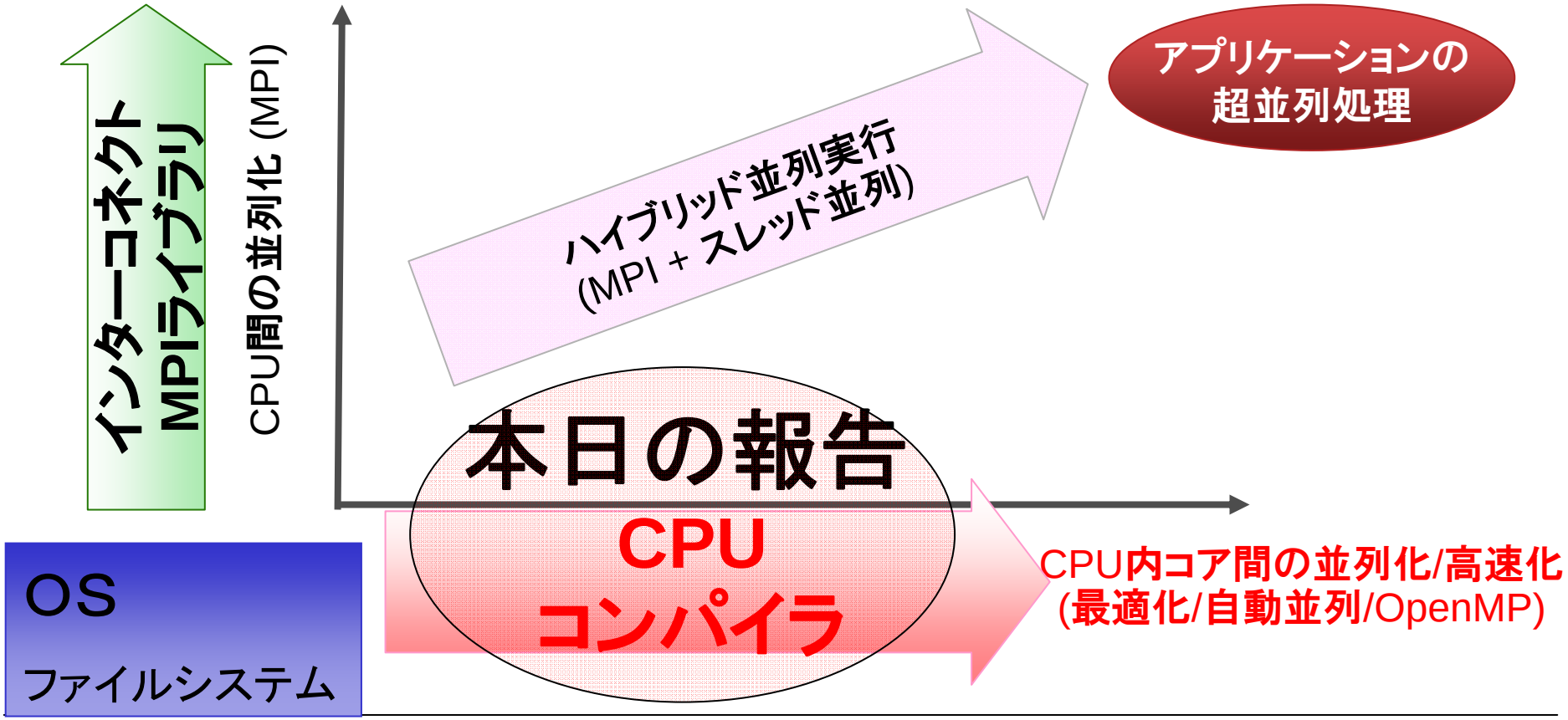
発表骨子

- SPARC64™ VIIIfxの概要
  - 開発の狙い
  - Chip概要
  - 諸元
- SPARC64™ VIIIfxの高速化技術
  - 評価環境／評価方法（サイクルアカウンティング（PA））
  - HPC－ACE
    - レジスタ拡張、セクタキャッシュ、SIMD命令、マスク演算
    - 逆数近似命令、三角関数補助、高機能prefetch
  - VISIMPACT
    - ハード機構
    - 自動並列化
- その他
  - サイクルアカウンティング（PA）を用いたチューニング
  - PCクラスタ、ベクトル機からの移行
- まとめ

【お断り】
講演で使用したスライドの一部（性能データ等）は、開発途中での測定結果であるため、当Web掲載版からは削除しています。

開発の狙い：超並列に向けた富士通のアプローチ

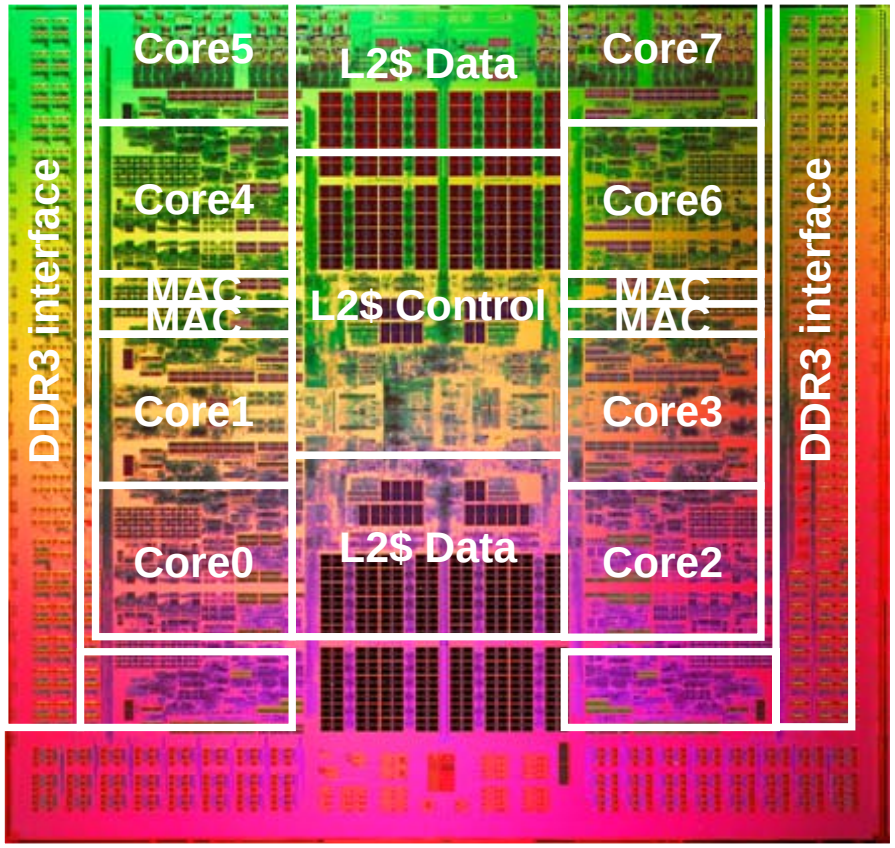
- **CPUのマルチコア化とともにHPC向けに強化**
- **CPU-メモリ間のバンド幅を重視した 1CPU-1ノード構成**
- **ハイブリッド並列実行（推奨）により高効率な超並列処理を実現**
  - CPU内コア間スレッド並列処理(自動並列/(OpenMP)) ＋ CPU間プロセス並列(MPI)



SPARC64™ VIIIfxの概要

SPARC64™ VIIIfx Chip 概要

設計目標：高性能、省電力かつ高信頼性



- アーキテクチャ
  - 8 コア
  - 5 MB の共有L2キャッシュ
  - メモリコントローラ内蔵
  - クロック 2 GHz
- FMLの 45nm CMOS
  - 22.7mm x 22.6mm
  - 760M トランジスタ
  - 信号ピン数 1271
- ピーク性能
  - 演算性能　128GFlops
  - メモリスループット　64GB/s
- 消費電力
  - 58W (TYP, 30℃)
  - 水冷
    - リーク電流削減、信頼性向上

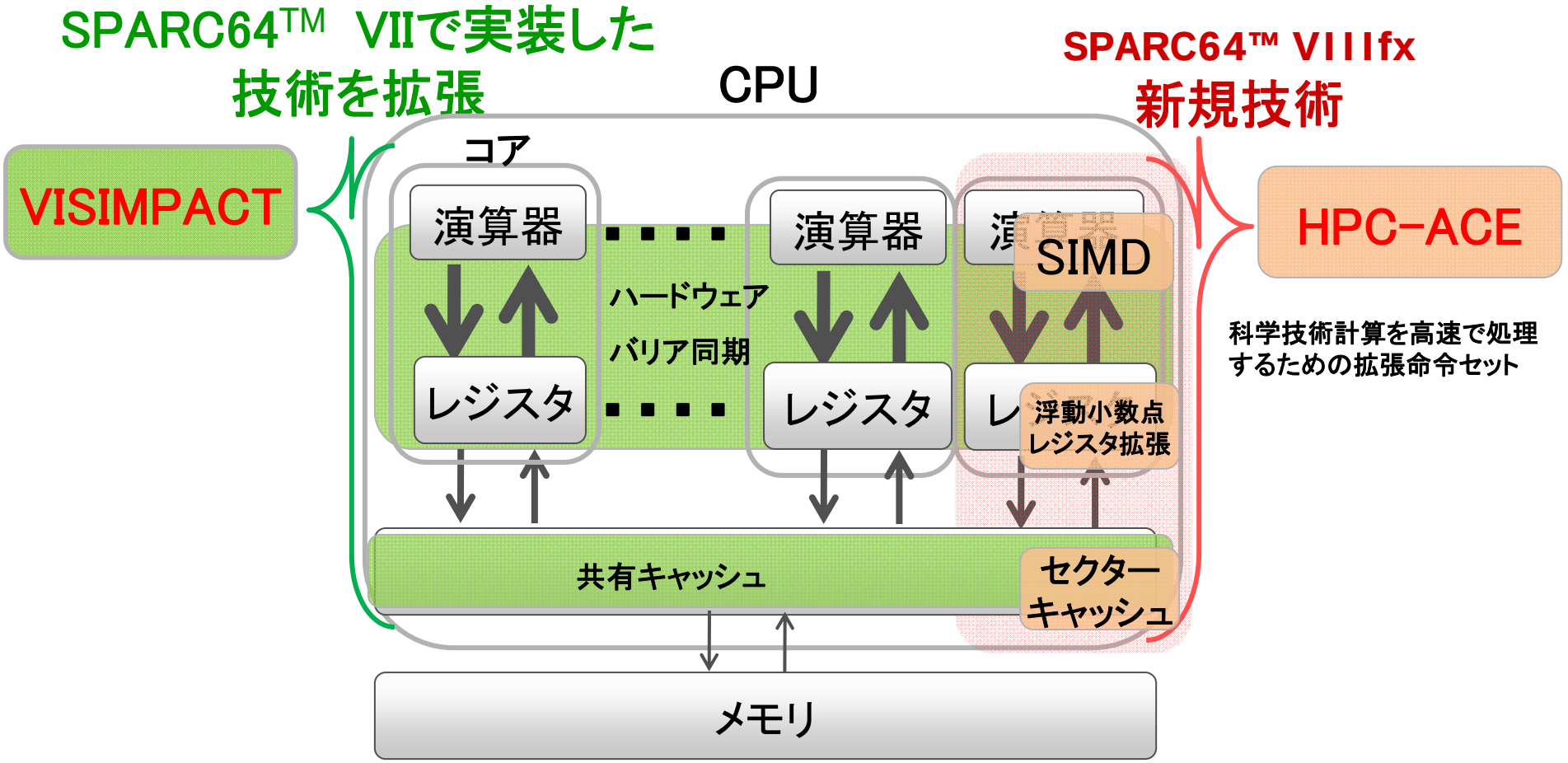
SPARC64™ VIIIfx 諸元

	SPARC64™ VIIIfx	参考 SPARC64™ VII	参考 **Nehalem-EP**
コア数	8	4	4
マシンクロック	2.0GHz	2.52GHz	2.93GHz
命令セット	SPARC-V9/JPS1 ＋HPC-ACE	SPARC-V9/JPS1	Intel® 64
FPレジスタ数	256	32	16
倍精度演算性能	チップあたり 128.00GFLOPS	チップあたり 40.08GFLOPS	チップあたり 46.88GFLOPS
クロック・コアあたり 倍精度演算実行数	8	4	4
倍精度演算器構成	2FMA×2SIMD	2FMA	(FM+FA)x2SIMD
L1キャッシュ	コアあたり 命令：32KB/2WAY データ：32KB/2WAY	コアあたり 命令：64KB/2WAY データ：64KB/2WAY	コアあたり 命令：32KB/4WAY データ：32KB/8WAY
L2キャッシュ	コア共用 5MB/10WAY	コア共用 6MB/12WAY	コアあたり 256KB/8WAY
L3キャッシュ	なし	なし	コア共用 8MB/16WAY

SPARC64™ VIIIfxの高速化技術

- HPC-ACE
- VISIMPACT

SPARC64™ VIIIfx 高速化技術



→ コンパイラ/アプリケーションで、いかに活用できるか/するかが高速化のキーポイント

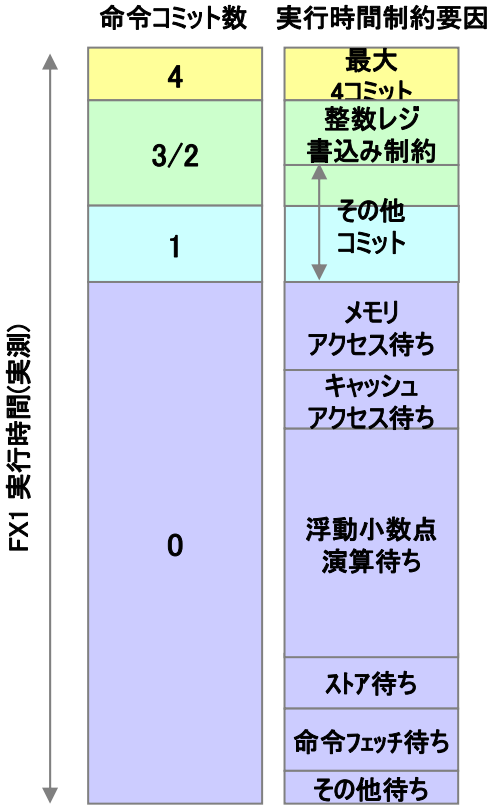
コンパイラ・アプリから見た高速化機能の特徴

特徴	高速化要因
8コア共用キャッシュ	細粒度スレッド並列時のFalse sharing削減
チップ内ハードバリア	並列時バリアオーバヘッド削減
SIMD命令	L1＄上で演算/アクセス×2倍高速化
256FPレジスタ	命令並列性向上 レジスタ不足によるspill/fill削減
セクタキャッシュ	ソフトによるキャッシュ制御
マスク演算	SIMD化促進、命令並列性向上

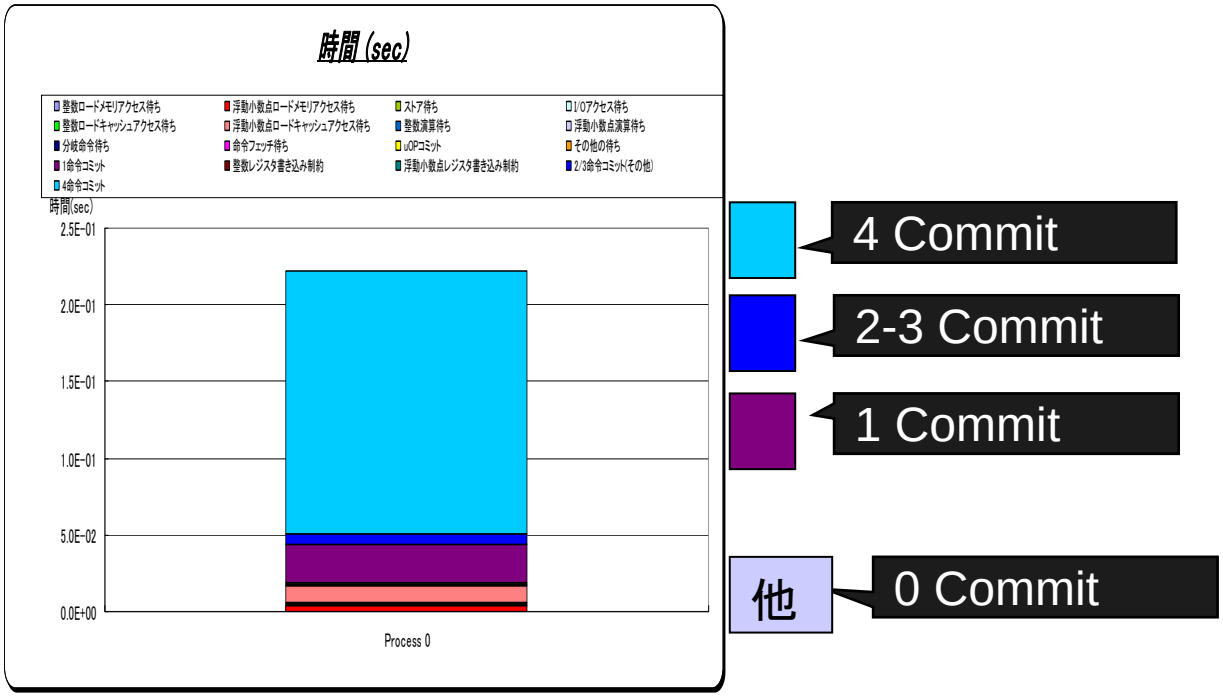
評価方法：サイクルアカウンティング(PA)

- サイクルアカウンティングとは、性能ボトルネック要因分析の手法である。
  （SPARC64™ VIIIから）豊富なPAイベント情報を備えており、アプリケーションプログラム実行時のCPU動作状態の情報が取得できる。あるアプリケーションプログラムを実行するためにかかった総時間（CPUサイクル数）をCPUの動作状態で分類し、CPU内のどの部分にボトルネックがあるかを把握することで、詳細な性能分析やチューニングを行うことができる。

概略図



例：行列積　A=MATMUL(B,C)



Commit:1マシンサイクルでn命令実行したことを示す。

OCommit部分は、何らかの要因で命令が動作していない時間。

SPARC64™ VIIIfxの高速化技術

・HPC-ACE

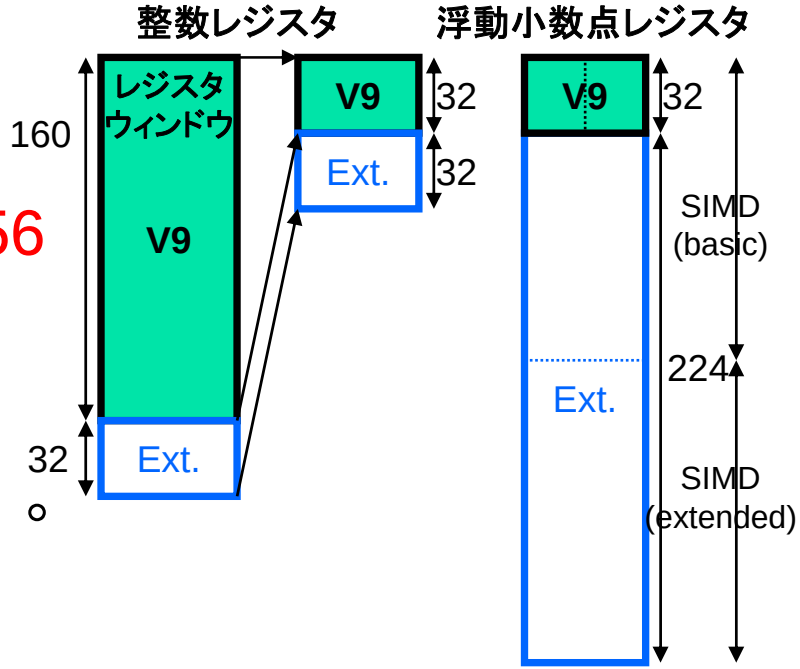
HPC-ACE

(High Performance Computing - Arithmetic Computational Extensions)

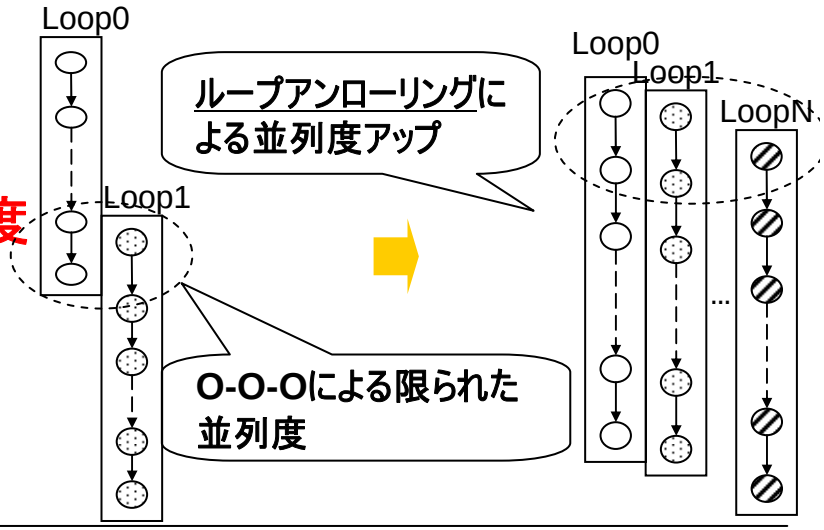
- SPARC64™ VIIIfxのISA (Instruction Set Architecture)
  - 準拠仕様
    - SPARC-V9 仕様
    - JPS (Joint Programmer's Specification): SPARC-V9拡張仕様
  - HPC-ACE: 富士通独自のHPC向け命令セット拡張
    - レジスタ拡張
    - セクターキャッシュ
    - SIMD (single instruction multiple data) 命令
    - マスク演算
    - 除算/平方根の逆数近似
    - 三角関数補助命令
    - 高機能prefetch/メモリアクセス制御機構
    - 浮動小数点数の大小比較
- 以下のSPARC64™ VIIIfx 関連文書は次のURLからダウンロードできます。
  http://jp.fujitsu.com/solutions/hpc/brochures/
  - The SPARC® Architecture Manual Version 9
  - SPARC® Joint Programming Specification (JPS1): Commonality
  - SPARC64™ VIIIfx Extensions

レジスタ拡張

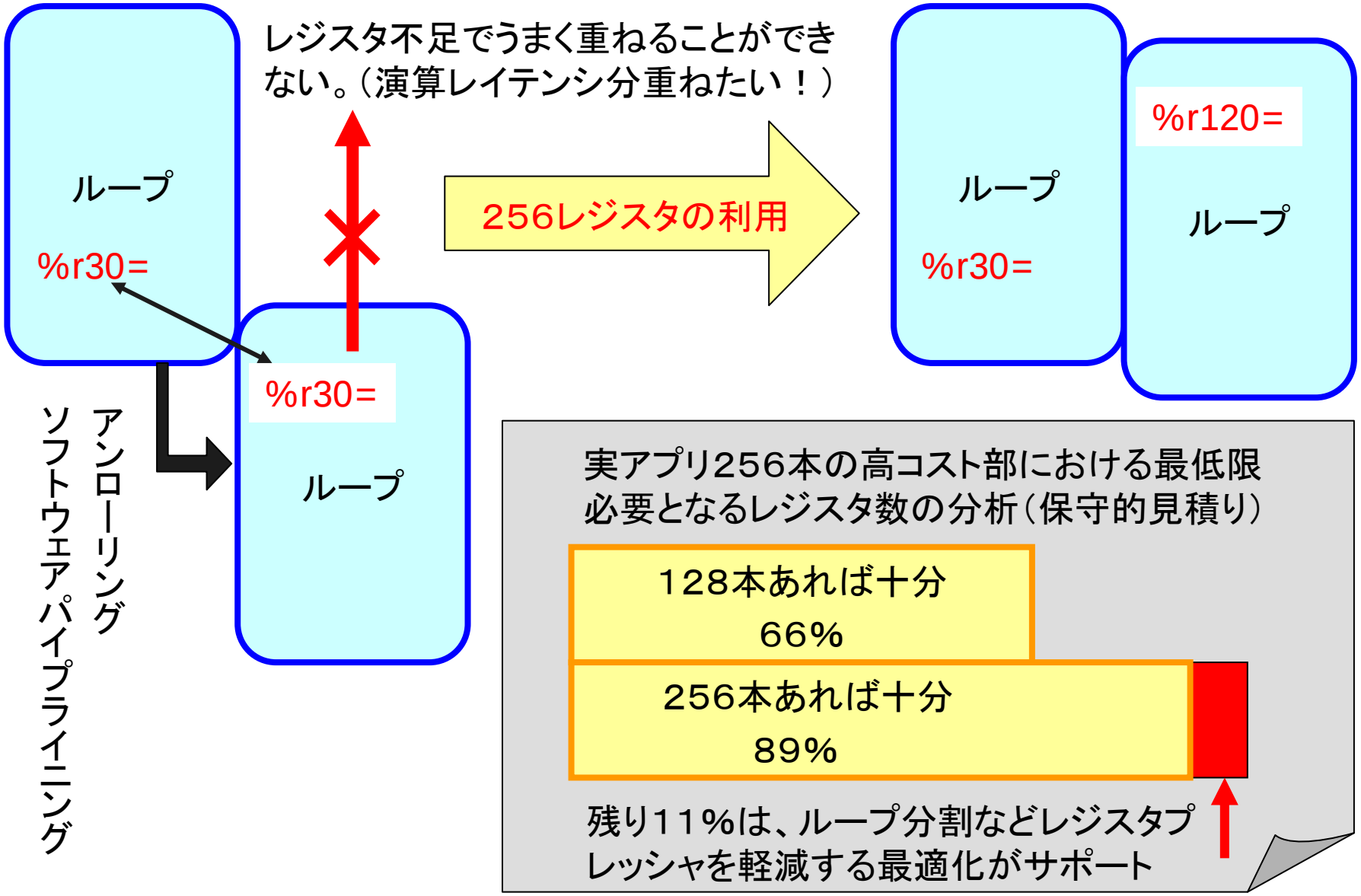
- V9からレジスタを拡張
  - 整数レジスタ　32→64
  - 倍精度浮動小数点レジスタ　32→256
    - 下位の32個のレジスタは SPARC-V9と共通
    - 浮動小数点レジスタはすべて同等。 拡張浮動小数点レジスタは 非SIMD命令からもアクセス可能。



- 拡張の理由
  - レジスタ数による制約で制限されている並列度を上げるため
  - レジスタのspill/fillによるオーバヘッドを削減



レジスタ拡張の利用例：命令並列性向上

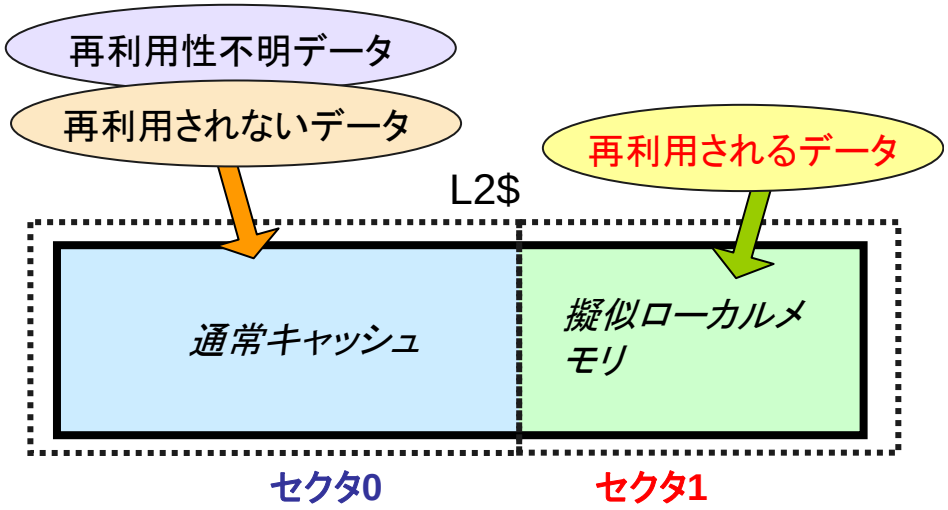


ソフトウェア制御キャッシュ：セクターキャッシュ

- **セクターキャッシュ: *擬似ローカルメモリ***

➔ソフトウエアが、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

- 再利用する配列 ➔ セクタ1を使用
- その他 ➔ 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ない
- ユーザは指示行でセクタ1に載せる配列を指定できる




```
!OCL L2_SECTOR_NWAYS_LOOP(8,2)
!OCL SECTOR1_DATA(a)
do j=1,m
   do i=1,n
      a(i) = a(i) + b(i,j) *c(i,j)
   enddo
enddo
```

〈意図〉
ループ中で配列 b と配列 c のアクセスによって配列 a がキャッシュから追い出されない

セクタキャッシュ指定のコンパイラ指示行の使用例

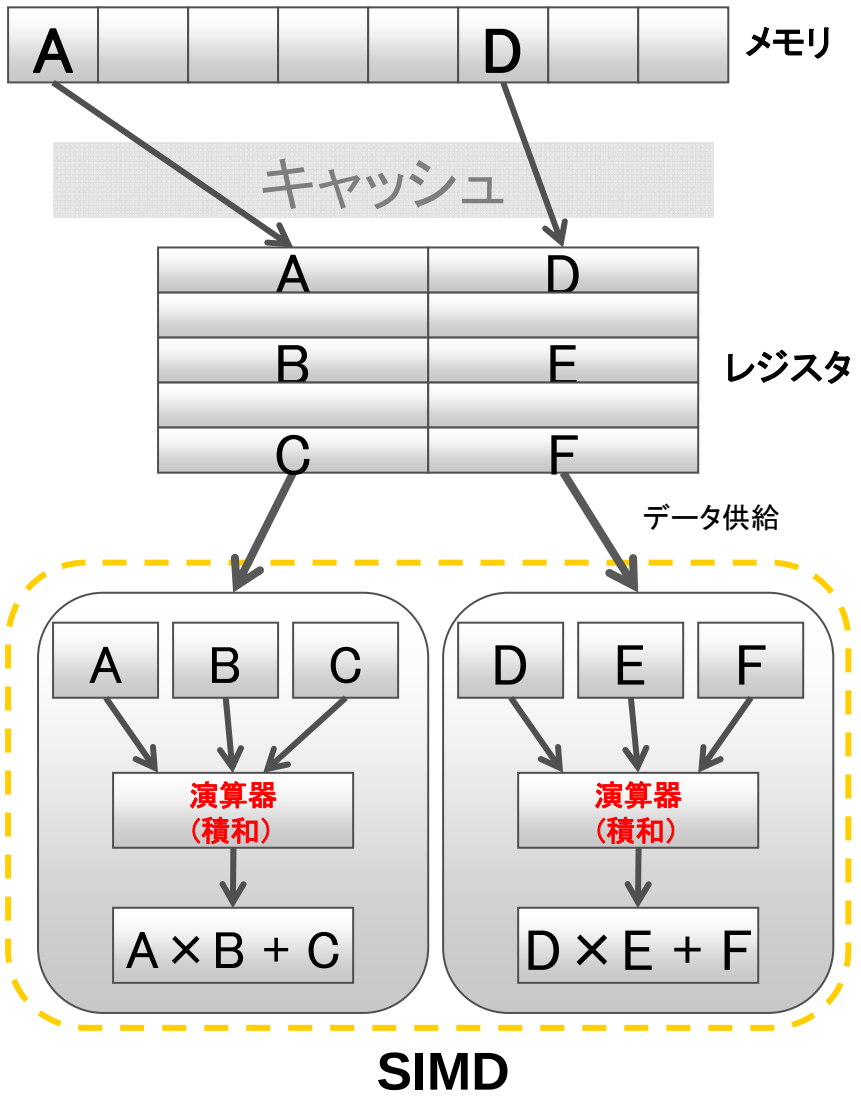
SIMD化（Single Instruction Multiple Data）

- SIMD
  - 1つの命令で，複数の演算を並列処理すること
- SPARC64™ VIIIfx のSIMDの特徴
  - 1命令で2つの演算を並列処理
  - 積和演算をサポート
  - 1つのコアで2つのSIMD命令を同時に実行可能（単精度および倍精度命令）

1つのコアで8個(1CPUで64個)の演算処理を同時に実行可能

  - 倍精度SIMDロードは、8バイト境界でもOK
  - 非連続なメモリ空間から1個づつデータを取得し，SIMDとして演算可能

「アプリが使いやすいSIMD」、
「計算処理の高速化」 を実現



SIMD化プログラミング

◆SIMD化　　≒　自動ベクトル化

≒　最内ループ自動並列化

◆ベクトル化プログラミングテクニックと類似

① 基本は、最内のループを対象

② 対象ループは、ループ繰り返しによるデータ依存性なし

➔自動並列化が適応された最内ループは、自動SIMD化可能

非連続データのSIMD化

```
do j = 1, 1000
  do i = 1, 1000
    a(i,j) = b(j,i) * 2.0
  enddo
enddo
```

b(j,i)の部分は同じ色の
擬似命令部分

SPARC64™ VIIIfxではSIMD化可能

```
load    b(1,1), %vr1.front
load    b(1,2), %vr1.back
vmult   %vr1, 2.0, %vr2
vstore %vr2, a(1,1:2,1)
```

load, load → スカラ命令
vmult → SIMD命令

他のアーキではSIMD化不可能

もしくは性能低下

```
vload  b(1,1), %vr1
vshift  %vr1, sizeof_type(b(1,1)), %vr1
vload  b(1,2), %vr2
vpack %vr1, %vr2, %vr3
vmult  %vr3, 2.0, %vr4
vstore %vr4, a(1,1:2,1)
```

全てSIMD命令

SPARC64™ VIIIfxでは副作用なしにSIMD化可能

SPARC64™ VIIIfxではスカラ命令とSIMD命令で使用するレジスタの区別がなく副作用がないため、SIMD化の促進が可能となる。

COMPLEX演算のSIMD化

- 複素数の乗算 y=x1*x2 (yrはyの実部、yiはyの虚部)
  - yr=x1r*x2r-x1i*x2i
  - yi=x1r*x2i+x1i*x2r
- SIMD命令を使った複素数の乗算
  - 複素数の実部・虚部をSIMDレジスタのペアに格納し、2つの積和命令でSIMD計算できる

```
yr=0.0
yi=0.0
```

SPARC64™ VIIIfxの積和命令では
特殊なSIMD計算が可能

```
yr= - x1i * x2i - yr
yi=   x1i * x2r - yi
```

実部と虚部で乗算の符号を変える
x1は両方とも虚部を使用
x2は実部と虚部を逆に使う

```
yr= x1r * x2r + yr
yi= x1r * x2i + yi
```

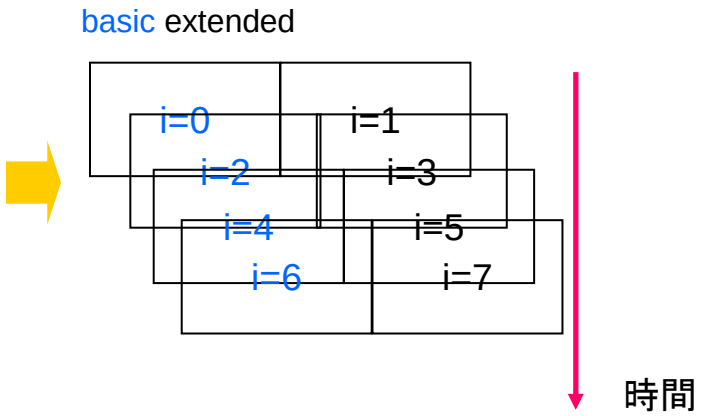
x1は両方とも実部を使用

マスク付きSIMD化

- 条件付き命令実行（マスク演算）
  - ‘if’ を含むループの実行を効率化
  - 以下の条件付き命令を使うことで、条件分岐命令を削除することが可能
    - FPレジスタを比較して結果をFPレジスタに書き込む
    - FPレジスタの値に基づいて、選択的にFPレジスタ間でデータ転送
    - FPレジスタの値に基づいて、選択的にFPレジスタの値をメモリストア

➔ ソフトウェアパイプラインによる最適化が可能

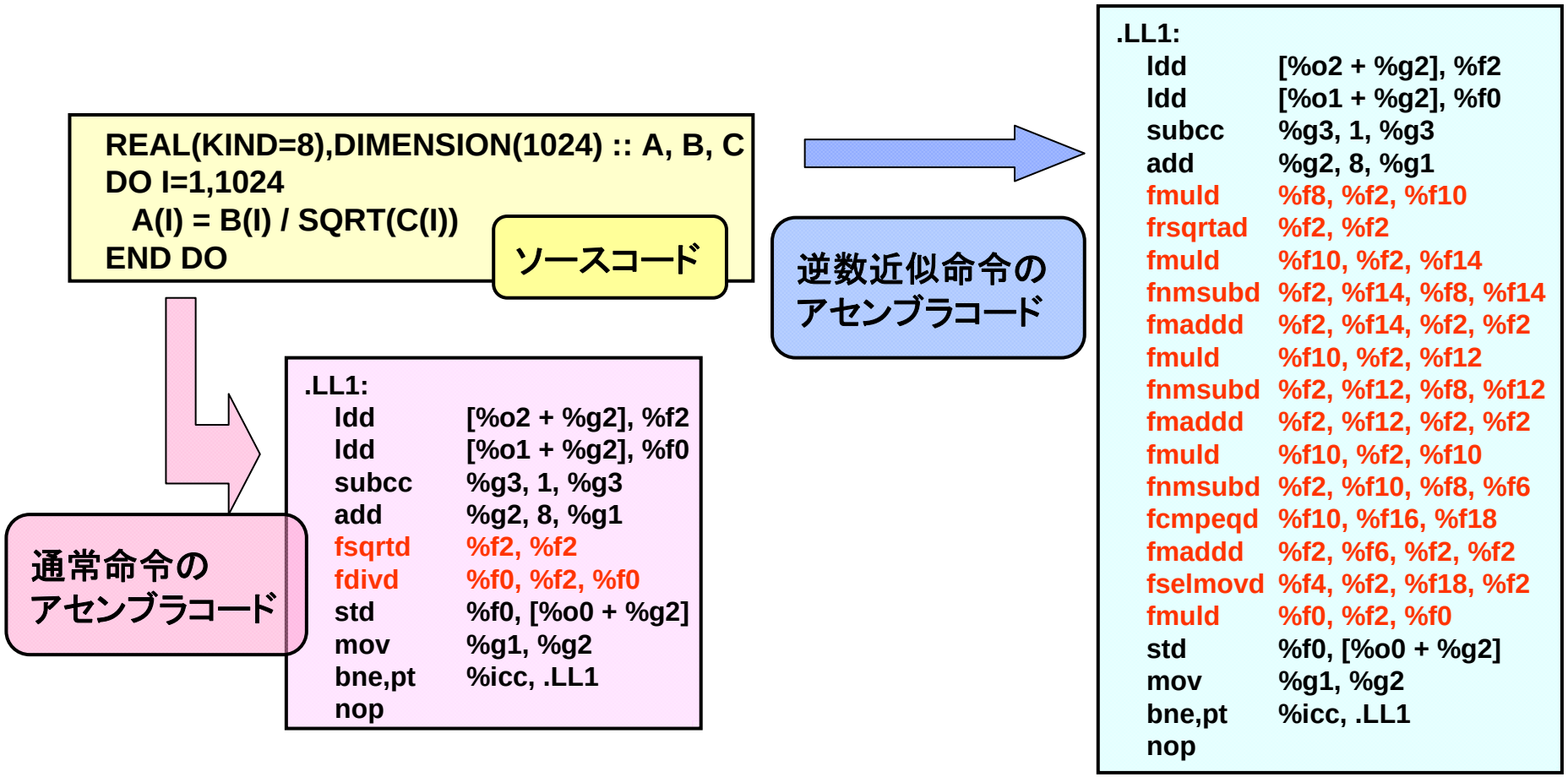
```
for (i=0; i<00; i++) {
  if (A(i)>0)
    FPop X(i)
  else
    FPop Y(i)
}
```



SIMD化でスループット向上

逆数近似命令

- 除算/平方根の逆数近似
  - 丸め誤差＜1/256で逆数近似を計算
  - パイプライン動作により除算/平方根を高速に計算



三角関数補助命令

- 三角関数の計算方法(従来)

  sin(x)は周期性を利用してx=2πP+π/2×Q+R (Pは整数、Q={0,1,2,3}、
  −π/4<R≦π/4) となるQ, Rを決め、Qの値により計算方法を選択する

  Q mod 4 = 0 のとき　　sin(R)を最良近似式で求める
  Q mod 4 = 1 のとき　　cos(R)を最良近似式で求める
  Q mod 4 = 2 のとき　　-sin(R)を最良近似式で求める
  Q mod 4 = 3 のとき　　-cos(R)を最良近似式で求める

- 三角関数補助命令を使った計算方法
  - sin、cosの近似式を以下のように共通化し、Qの値により係数を自動選択することで判定を無くした

$$\sin^*(R) = R\,(1+R^2(s_0+R^2(s_1+R^2(s_2+R^2(s_3+\ldots)))))$$
$$\cos^*(R) = 1\,(1+R^2(c_0+R^2(c_1+R^2(c_2+R^2(c_3+\ldots)))))$$

Q, Rからsinを計算する命令列

```
ftrismuld R,Q,M
ftrisseld R,Q,N
ftrimadd S,M,7,S
ftrimadd S,M,6,S
ftrimadd S,M,5,S
ftrimadd S,M,4,S
ftrimadd S,M,3,S
ftrimadd S,M,2,S
ftrimadd S,M,1,S
ftrimadd S,M,0,S
fmuld S,N,S
```

ftrisseld命令でR or 1 および結果の符号を選択

ftrismuld命令で係数テーブルを選択し、ftrimaddd命令で近似式を計算する

SPARC64™ VIIIfxの高速化技術

- VISIMPACT

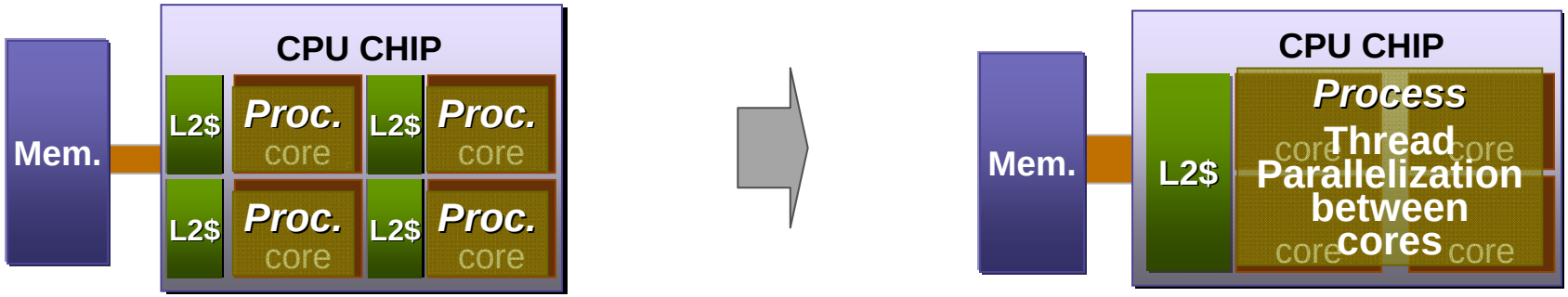
VISIMPACT – ハイブリッド並列を容易にするための仕組み –
(Virtual Single Processor by Integrated Multi-core Parallel Architecture)

●コンセプト

■マルチコアCPU内の高効率なスレッド並列を実現する技術

■高効率なハイブリッド並列実行モデルの実現を支援

MPI + スレッド並列処理(自動並列 /OpenMP)



●狙い

■マルチコアCPUを高速な一つのCPUを扱うことで……

◆MPI プロセス数を 1/nコアに減らす

◆並列処理効率を向上

◆メモリアクセスを軽減

●技術的挑戦

■コア間のスレッドレベル並列処理のオーバヘッドをどう減らすか?

VISIMPACTの仕組み

◆CPU 技術

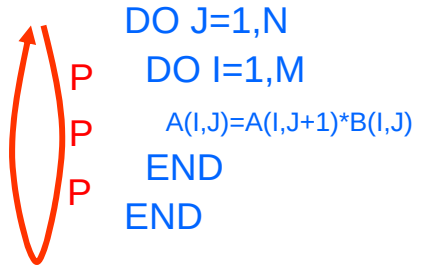
- ■コア間のハードバリア機能
  - ➔ソフトウェアバリアと比較して10倍の高速化
  - ➔コア数に依存せずに一定のオーバヘッドを実現
- ■共有L2キャッシュメモリ (5 MB)
  - ➔キャッシュ間のメモリ交換(false share)を軽減
  - ➔キャッシュメモリの利用を効率化
- ■HPC-ACE
  - ➔チップ内演算の高速化

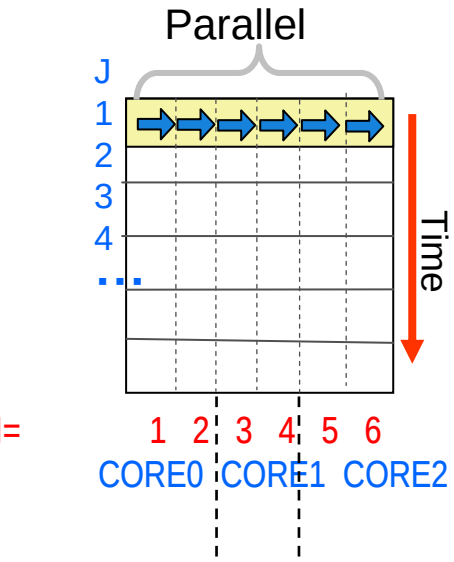
  （L2$セクターキャッシュは、スレッド並列前提）

◆コンパイラ技術

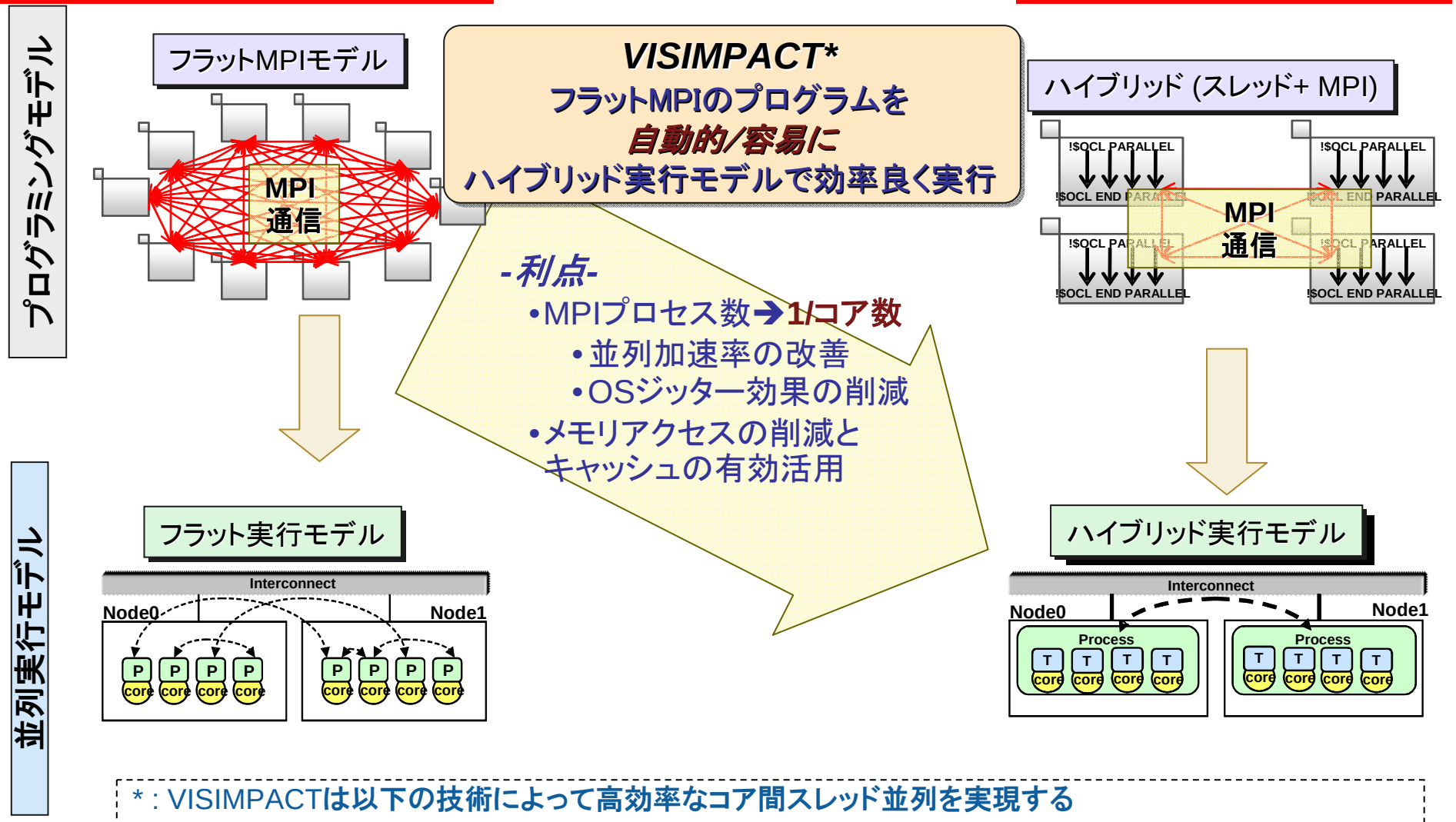
- ■ベクトル化の技術を適用した、高効率なスレッド並列(自動並列、OpenMP) を実現

●並列化のオーバヘッド削減で細粒度並列も可能





アプリ重視のアプローチ - VISIMPACT -



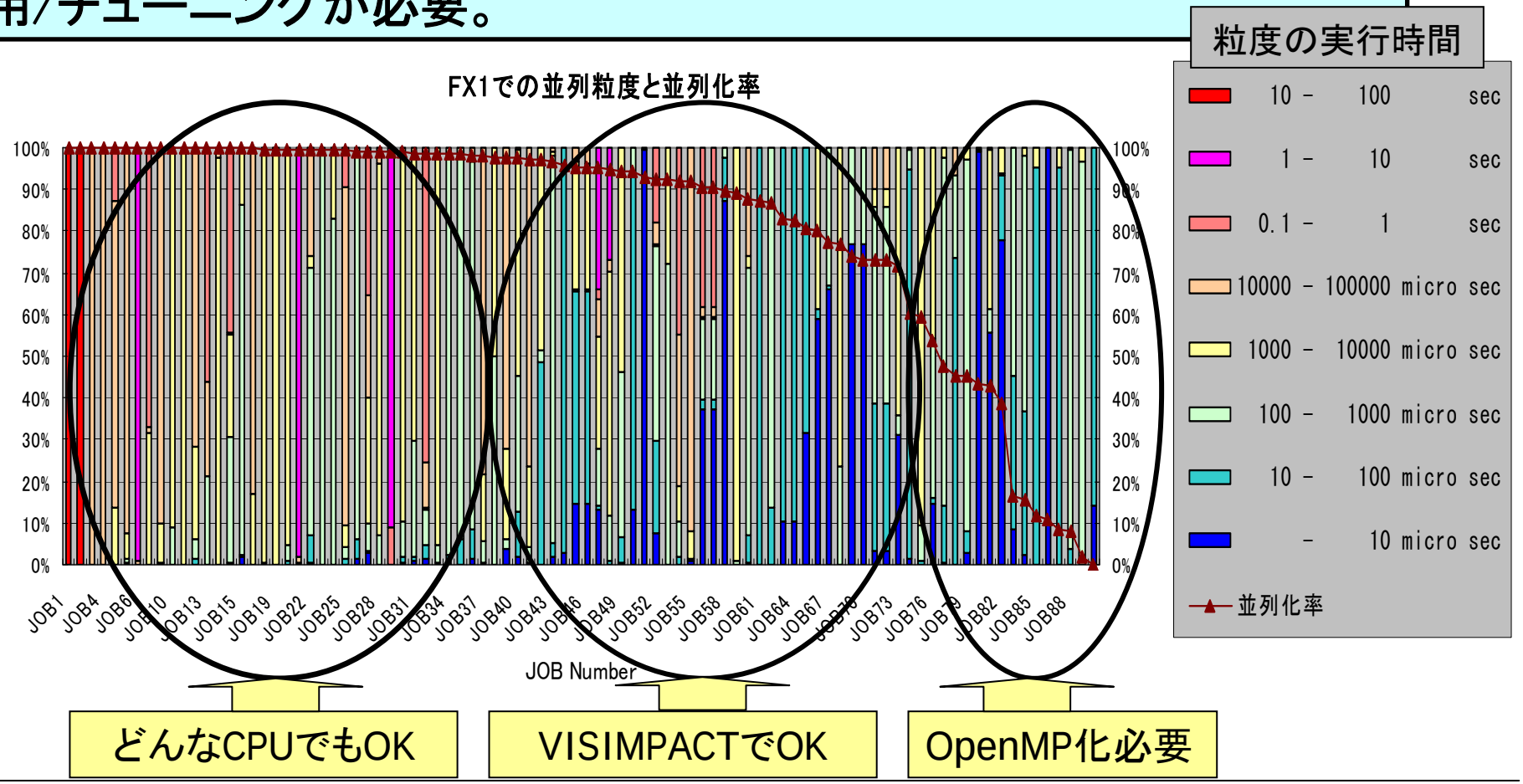
* : VISIMPACTは以下の技術によって高効率なコア間スレッド並列を実現する

CPU : コア間高速ハードバリア、共有L2キャッシュ、HPC-ACE
コンパイラ : 自動ベクトル化技術を継承した高効率なスレッド並列機能(自動並列化 or OpenMP)

自動並列化で行けるのか？　粒度と並列化率

- 検証：FX1＋現行コンパイラを用いて100本の実コードで粒度と並列化率を検証

結果：VISIMPACT機構を装備しているシステムならば3/4のコードが自動並列化のみである程度の並列効果。残り1/4はOpenMPとの併用/チューニングが必要。



自動並列化能力

◆ VISIMPACT機構により、ベクトル化より広い範囲に適用！

		ベクトル化	SPARC64™ VIIIスレッド並列(予定)	現行スレッド並列
範囲		○(最内ループ)	◎(最内ループ＋手続き)	△(外側ループ)
データ型		○(4/8バイト型)	◎(全て)	◎(全て)
データ依存関係	依存なし	○	○	○
	順方向依存	○	○	×
	逆方向依存	×	○	×
ループの演算内容	四則演算	○	○	○
	リダクション演算	○	○	○
	収集・拡散	○	○ (予定)	×
	DOブランチ	○	○ (予定)	×
粒度		○(ループ長数十)	○(ループ長数十)	×(ループ長数千)

ANLベクトル化コンテストプログラム(全135ループ)を用いて、各種ループの解析能力を比較

	ベクトル化	SPARC64™ VIII スレッド並列(予定)	現行スレッド並列
ベクトル化、並列化可能なループ数	86ループ	89ループ以上	67ループ

その他

- サイクルアカウンティング（PAイベント情報）を用いたチューニング
- PCクラスタ、ベクトル機からの移行

PAイベント情報を用いたチューニング事例

サイクルアカウンティング（PAイベント情報）は、
アプリ実行の挙動が容易に把握でき、
チューニングのツールとして非常に有効。

- 1次キャッシュ・スラッシング回避例
- スレッド並列のロードバランス不均等例

今回は、SPARC64™ VIIのPA情報。

SPARC64™ VIIIfxでも同様。

今後、アプリ開発者・利用者向けに
PAイベント情報を用いた
チューニングチュートリアルを
整備し公開（予定）

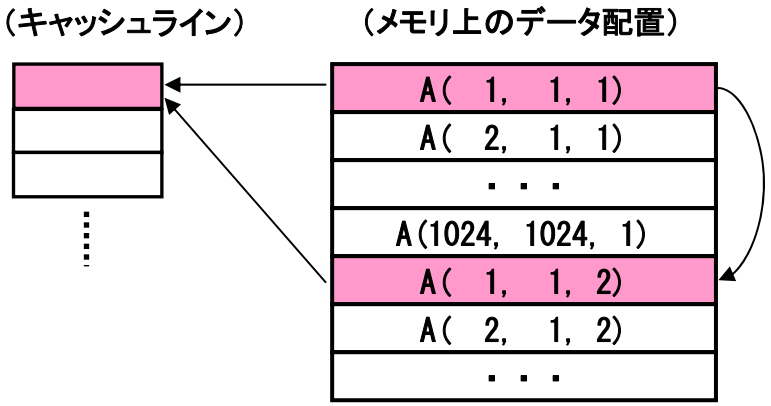
１次キャッシュ・スラッシング回避（前）

SPARC™ 64VIIのL1D$キャッシュは、64KB/2Wayです。

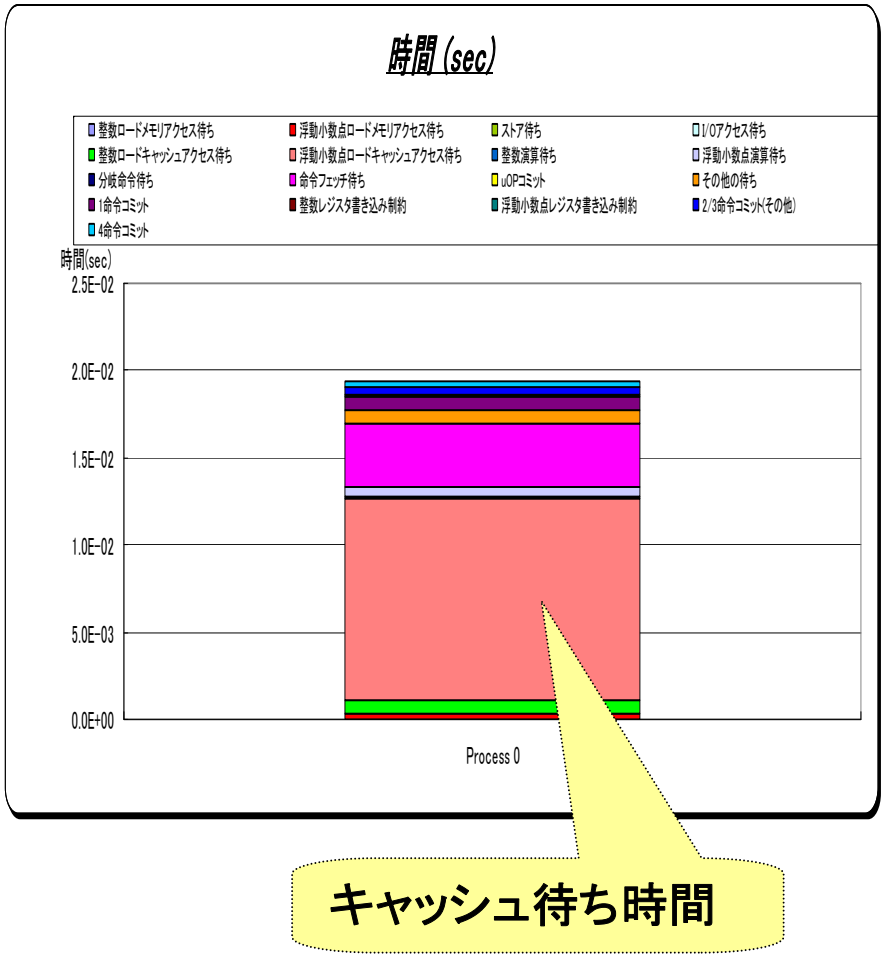
配列の形状が2のべき乗になっていると、キャッシュ競合で性能低下する場合があります。

オリジナルソース

```
REAL*8 A(1024, 1024, 4)
DO I=1,N
  A(I,1,1)=A(I,1,2)+A(I,1,3)+A(I,1,4)
ENDDO
```



同じキャッシュラインへの競合

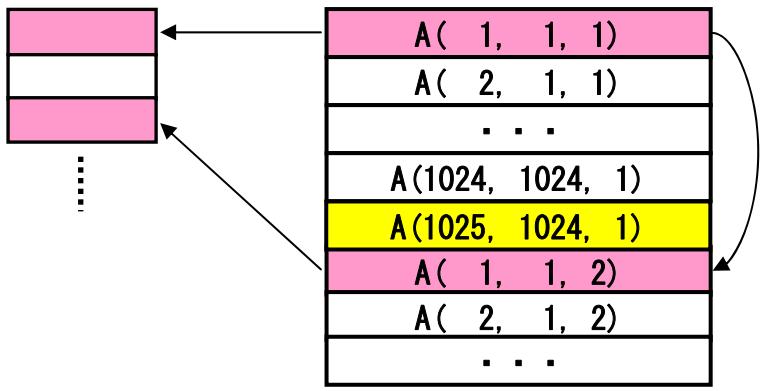


１次キャッシュ・スラッシング回避（後）

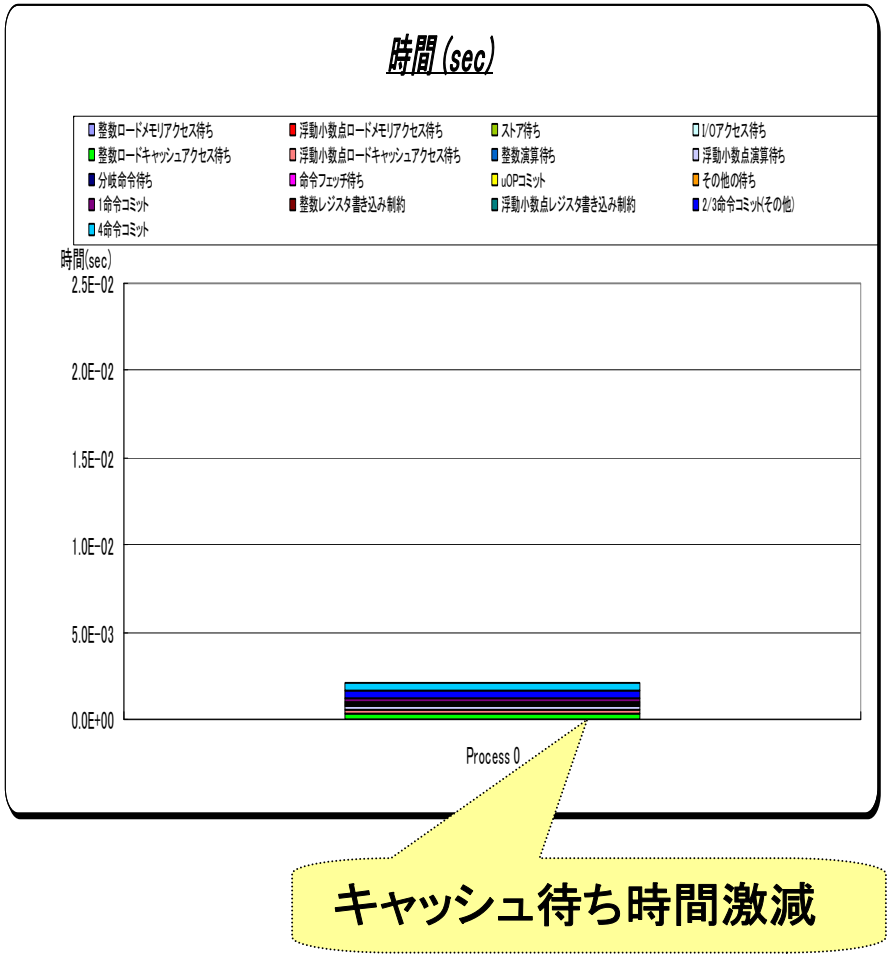
パディング（内側にすき間を空ける）を行うことにより、キャッシュ競合を削減します。

コンパイラ最適化のイメージ

```
REAL*8 A(1025, 1024, 4)
DO I=1,N
  A(I,1,1)=A(I,1,2)+A(I,1,3)+A(I,1,4)
ENDDO
```



異なるキャッシュラインで競合回避



キャッシュ待ち時間激減

スレッド並列のロードバランス不均等(前)

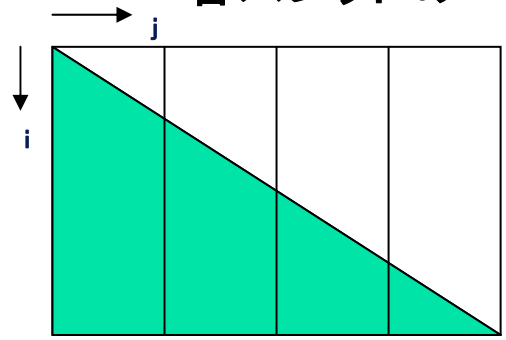
並列チューニングの観点から、解決しなければならない課題にロードバランスの均等化があります。スレッド並列の三角ループを外側ループで均等に割り付ける場合は、注意が必要です。

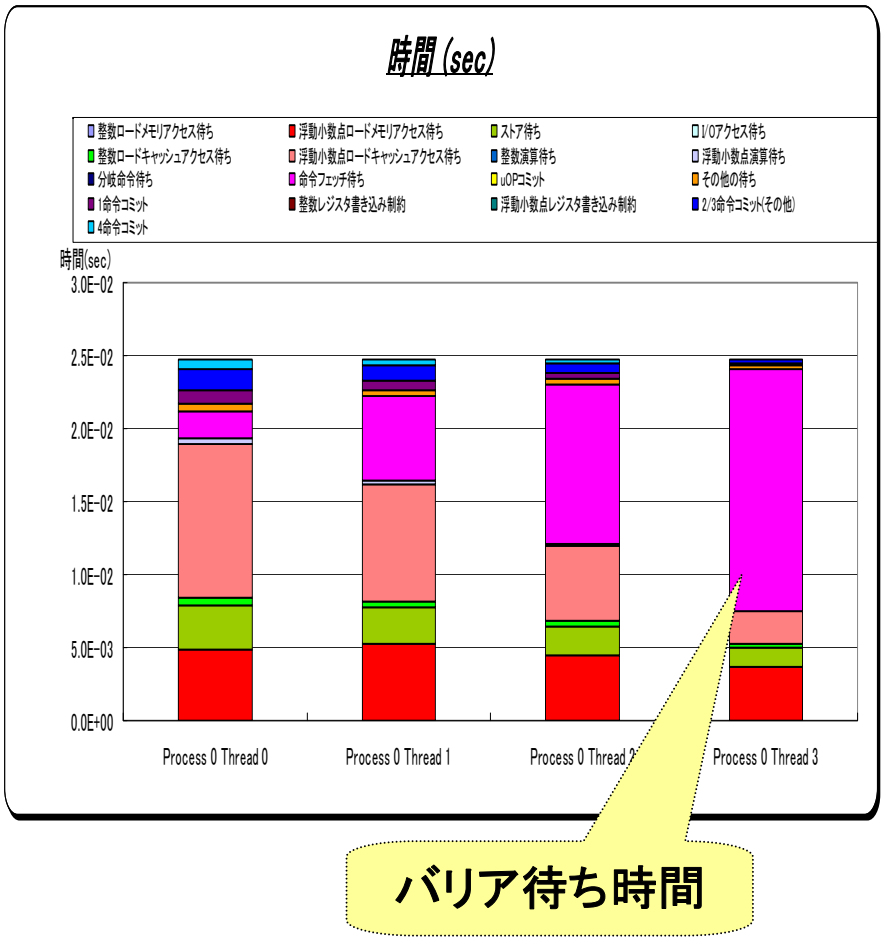
オリジナルソース

```
COMMON A,B,C,D
REAL*8 A(4097,4096),B(4097,4096),C(4097,4096)
!$OMP PARALLEL DO
DO J=1,4096
  DO I=J,4096
    A(I,J)=B(I,J)+C(I,J)
  ENDDO
ENDDO
```

三角ループ

各スレッドのバランス悪い！





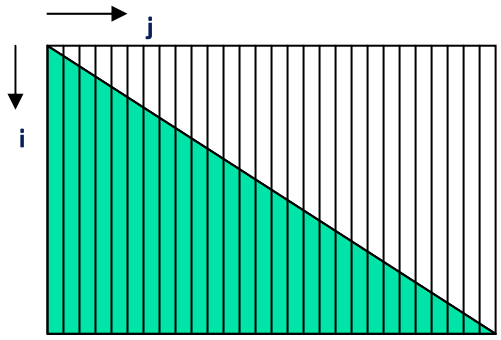
スレッド並列のロードバランス不均等(後)

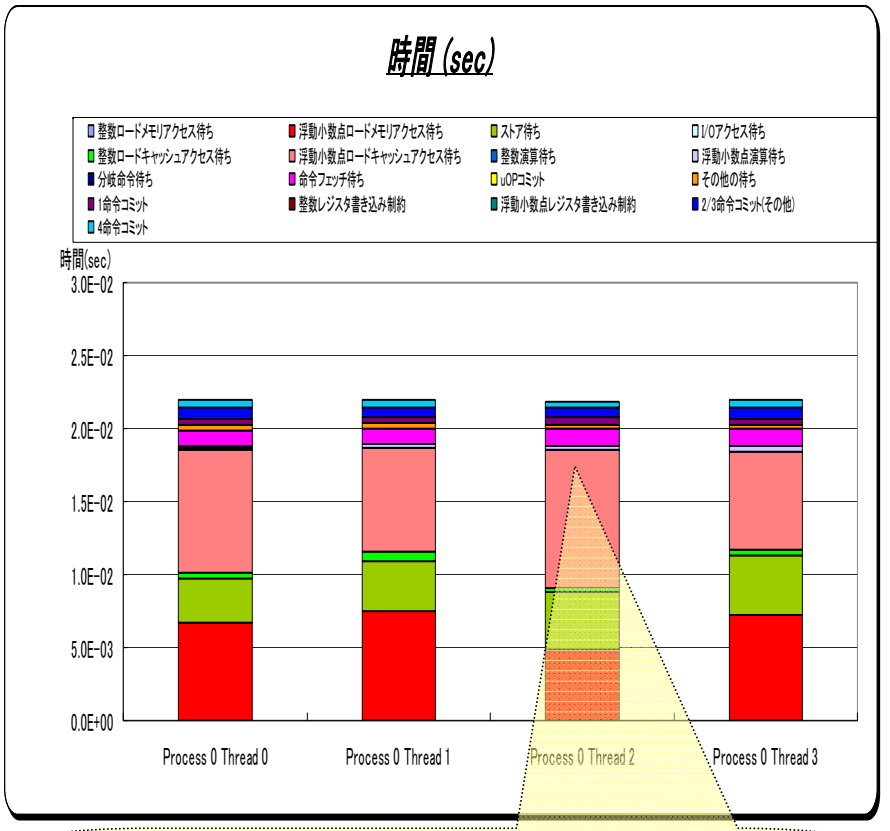
三角ループは、サイクリック割付を行うことにより、ロードバランスの均等化が可能となります。

チューニング後ソース

```
COMMON A,B,C,D
REAL*8 A(4097,4096),B(4097,4096),C(4097,4096)
!$OMP PARALLEL DO SCHEDULE(STATIC,1)
DO J=1,4096
  DO I=J,4096
    A(I,J)=B(I,J)+C(I,J)
  ENDDO
ENDDO
```

三角ループ





バリア待ち時間均一。ただし、1刻みだとL1$のFalse sharing影響見える

その他

- サイクルアカウンティング（PAイベント情報）を用いたチューニング
- PCクラスタ、ベクトル機からの移行

PCクラスタからの移行

- 弊社製プログラミング環境を標準搭載している場合、プログラムをスムーズに次期スパコンに移行できます。

		PCクラスタ	次期スパコン
プログラミングモデル	プロセス並列	MPI, XPFortran	MPI, XPFortran
	スレッド並列	自動並列, OpenMP (小規模スレッド並列)	自動並列, OpenMP (小規模スレッド並列)
	ハイブリッド並列	推奨モデル	推奨モデル
ソース互換	ソースプログラム	上位互換	
	ディレクティブ	上位互換	
コンパイラオプション		上位互換	
チューニング技法	メモリアクセス (キャッシュ)	効果的	チューニングが有効
	演算部	効果的	チューニングが有効

ベクトル機からの移行

- ベクトル機の特徴的な機能は、以下のように**SPARC64™ VIIIfx**の高速化機能とマッピングできます。

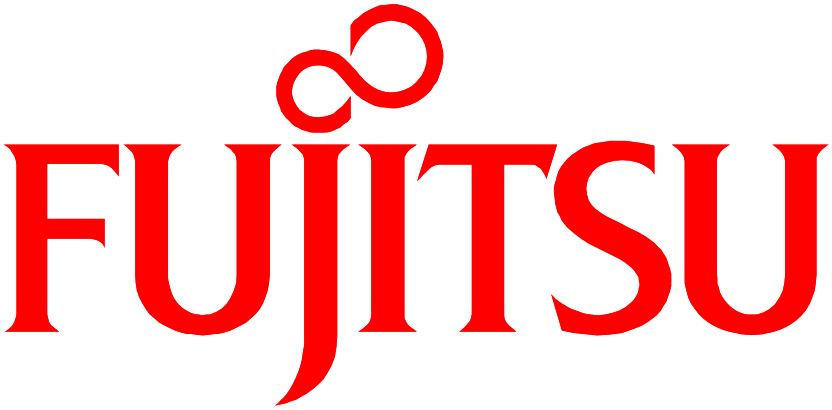
ベクトル機の特徴	次期スパコンでのマッピング
ベクトル化	VISIMPACT
ベクトル演算	8コアスレッド並列×SIMD 最大64演算/サイクル
高バンド幅メモリアクセス	キャッシュ＋プリフェッチ
ベクトルレジスタ	256レジスタ＋セクタキャッシュ
マスク演算	マスク演算

注）ベクトル機のアーキ固有にチューニング（外側ループアンローリング等）しているアプリは、スカラCPU向けの書き換えが必要です。

まとめ

SPARC64™ VIIIfx まとめ

- HPCアプリの観点からの高速化アプローチ HPC-ACE、VISIMACT
  「SPARC64™ VIIIfx のSIMD命令体系は、IntelのSSE命令よりも利用しやすい！」:コンパイラ開発者原口さん
- SPARC64™ VIIIfx チップは試作機動作中
- 開発中版コンパイラではあるが、HPC-ACE、VISIMACTの基本部分の有効性を確認できた
  - ➔当社、超並列アプローチ（推奨ハイブリッド並列実行モデル）のCPUコア/スレッド並列部分の高い実行性能
  - ➔今回の評価過程でSPARC64™ VIIIfx チップ向けの性能向上のポイントが明確になりつつある。今後、チューニングガイド等を整備しアプリ開発者/利用者と情報共有を行っていきたい
- コンパイラがまだ開発中であるため、実コード評価までには至らなかったが、今後の性能に期待が持てる
- 新時代の要求に応えるべく、富士通はSPARC64™ シリーズおよびコンパイラの開発を継続



THE POSSIBILITIES ARE INFINITE