SS研HPCフォーラム

FUJITSU
shaping tomorrow with you

富士通の
エクサスケールに向けた取り組み

追永　勇次

2012年8月20日
富士通株式会社

目次

FUJITSU

- 超並列システムへの取り組み
- 超並列システムの新たな成果
  - ベクトル機並みの実行効率の達成
  - VISIMPACTの新たな成果
- エクサスケールに向けた取り組み
- まとめ

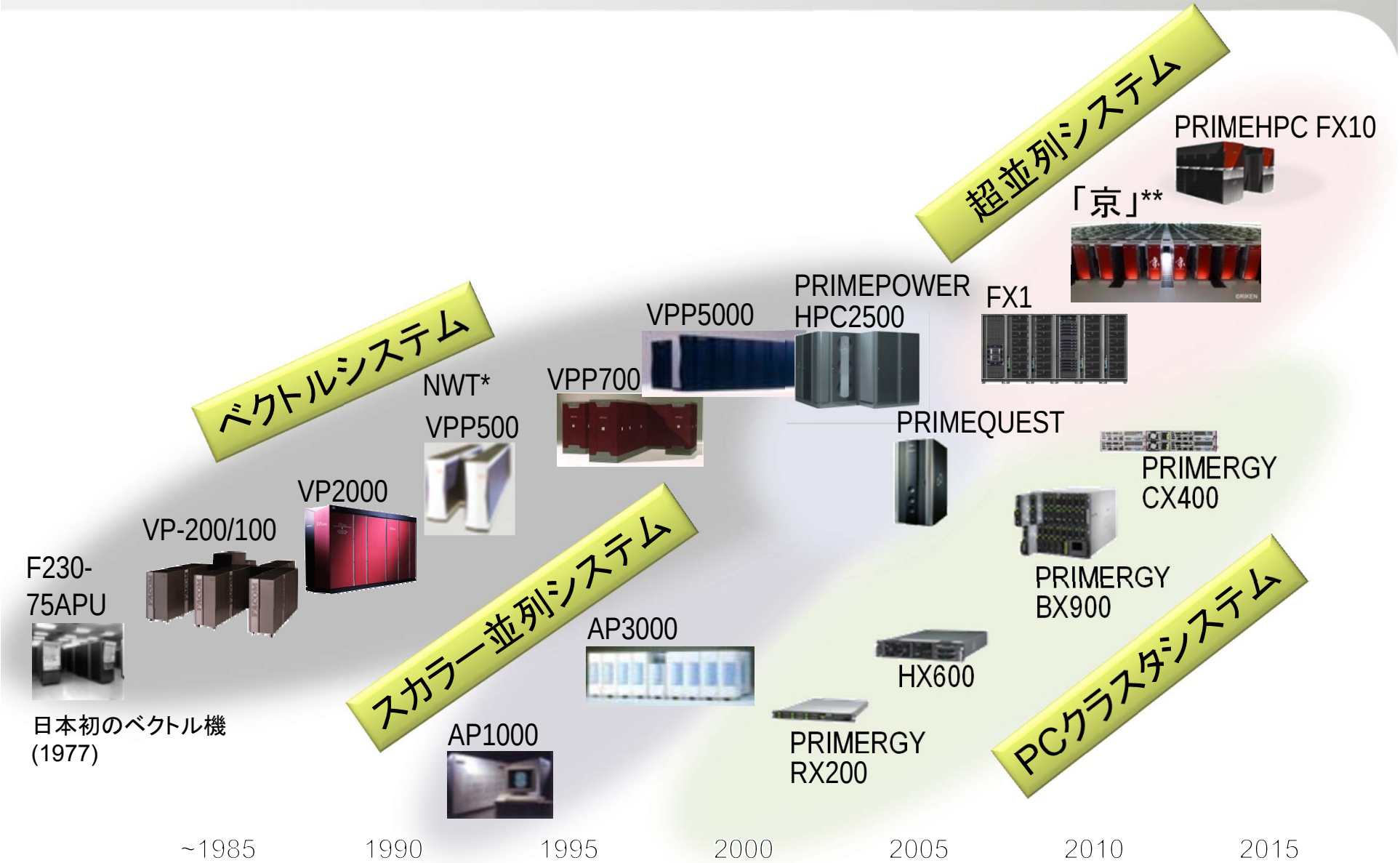
Copyright 2012 FUJITSU LIMITED

超並列システムへの取り組み

Copyright 2012 FUJITSU LIMITED

富士通のスパコンの歴史

FUJITSU



* NWT: 航技研様と共同開発

**「京」: 理研様と共同開発

Copyright 2012 FUJITSU LIMITED

超並列システムへの取り組み

FUJITSU

- マルチコアを想定したシングルCPU/ノード構成
  - FX1(4コア) → 京(8コア) → FX10(16コア)
  - 高いBF比(8 DIMM/ノード)
- 連続性に配慮しながら、超並列で必須となるアーキを段階的に導入
  - 高性能、マルチ・メニーコアへの対応：
    - SIMD拡張、自動ハイブリッド並列(VISIMPACT*)

      * ***Vi***rtual ***S***ingle Processor by ***I***ntegrated ***M***ulti-core ***P***arallel ***A***r***c***hi***t***ecture
  - 高いスケーラビリティ：
    - 集団通信のHWサポート、直接網(Tofu)

自動ハイブリッド並列
集団通信のHWサポート

直接網（Tofu）
SIMD拡張ほか（HPC-ACE）

FX1

CY2008～
40GF, 4コアCPU

K computer

CY2010～
128GF, 8コアCPU

PRIMEHPC FX10

CY2012～
236.5GF, 16コアCPU

Copyright 2012 FUJITSU LIMITED

HPC-ACEアーキテクチャ

FUJITSU

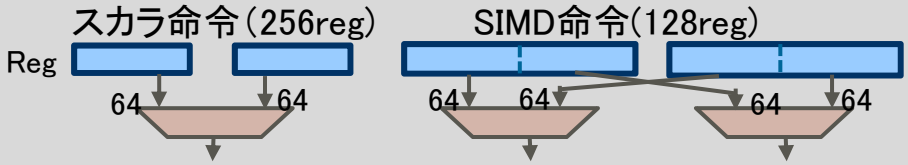
SIMD / レジスタ拡張

レジスタ

・64bit浮動小数点レジスタ数を 32→ 256に拡張

・128bit x 128個のSIMDレジスタとしても利用

演算命令



SIMDレジスタ仕様比較

HPC-ACE	Intel (AVX)	Intel (KNC)	IBM (P7)
128bit x 128	256bit x 16	512bit x 32	128bit x 64

メモリアクセス命令

スカラ命令(256reg): 64bitロード/ストア

SIMD命令(128reg): 128bitロード/ストア

リストアクセスサポート比較

HPC-ACE	Intel (AVX)	Intel(KNC)	P7
スカラ命令で代替可	AVX2から導入	あり	なし

マスク付きSIMD命令

・マスク生成、マスク付きmove、マスク付きストア

セクターキャッシュ

機能

キャッシュを2セクタに分け、ソフトで制御

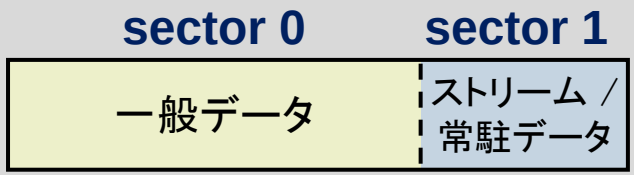
使用例

①ストリームデータを特定セクタに割り当て

→ 一般データのキャッシュを汚さない

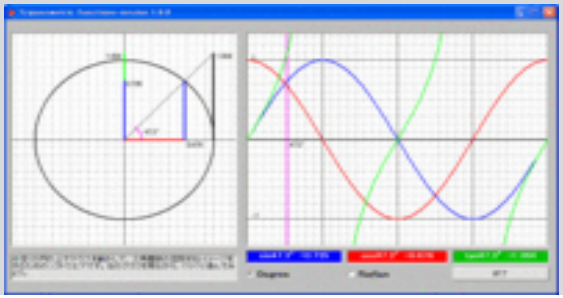
②常駐データを特定セクタに割り当て

→ 一般データに追い出されない領域確保



科学技術計算加速命令

逆数近似、三角関数加速命令のサポートで、従来と比べて数倍高速化



Copyright 2012 FUJITSU LIMITED

VISIMPACT （Virtual Single Processor by Integrated Multi-core Parallel Architecture）

FUJITSU

■ ハイブリッド並列のメリット/デメリット

○プロセス並列のスケーラビリティの向上 (プロセス数削減による通信量の削減)

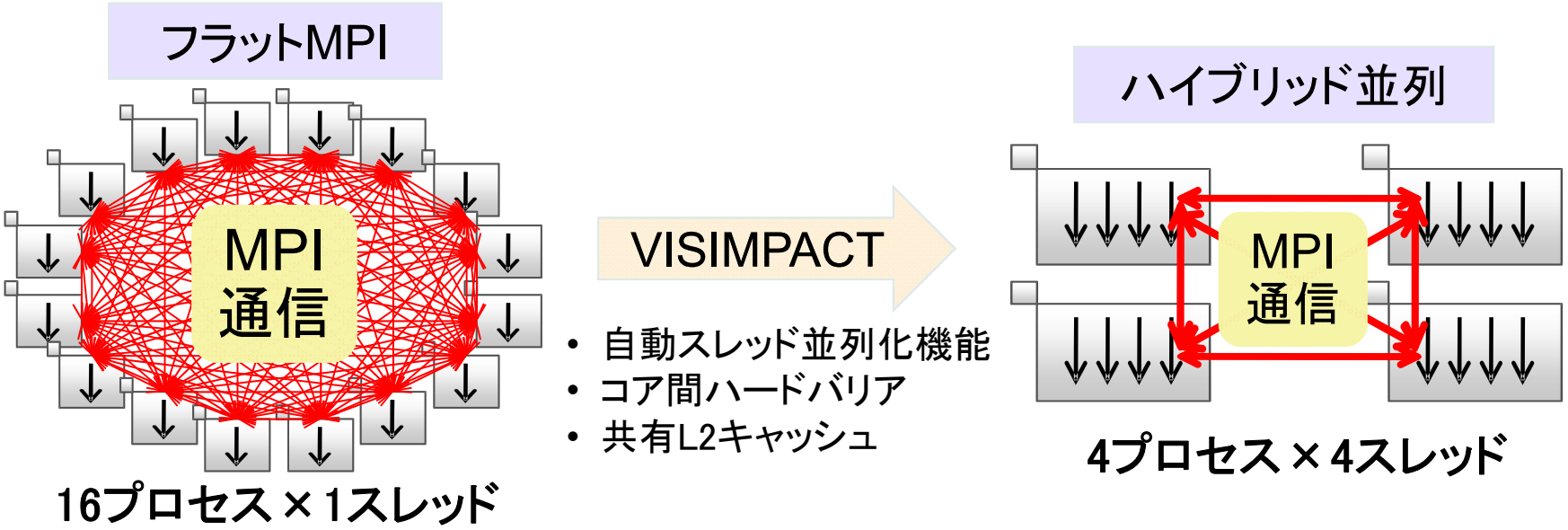
○利用可能なプロセスあたりメモリを大きく確保できる

○メモリの効率的な使用 (作業領域の削減)

×2階層の並列化が必要でプログラミング工数が大

■ VISIMPACT：MPIプログラムを自動的にハイブリッド並列化

- ■ 自動ベクトル化技術を発展させ高度な自動スレッド並列化機能
- ■ スレッド並列時の低オーバーヘッドを実現するSPARC64のハードアシスト



Copyright 2012 FUJITSU LIMITED

Tofuインターコネクトのネットワークトポロジー

FUJITSU

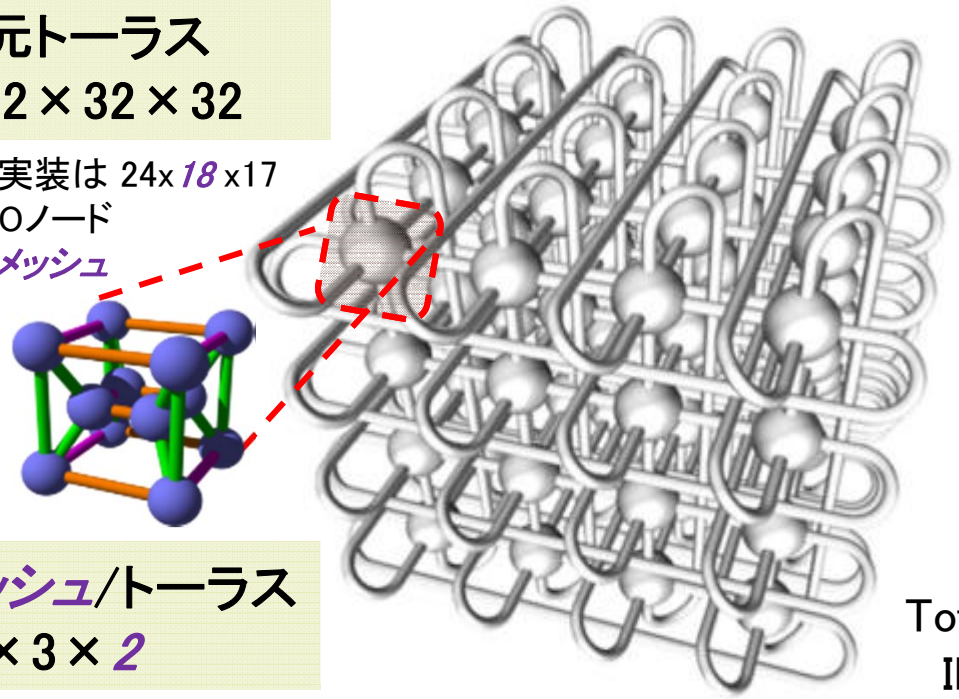
直接網 6次元*メッシュ*/トーラス（最大32x32x32x*2* x3x*2* ）

イタリックの軸がメッシュ

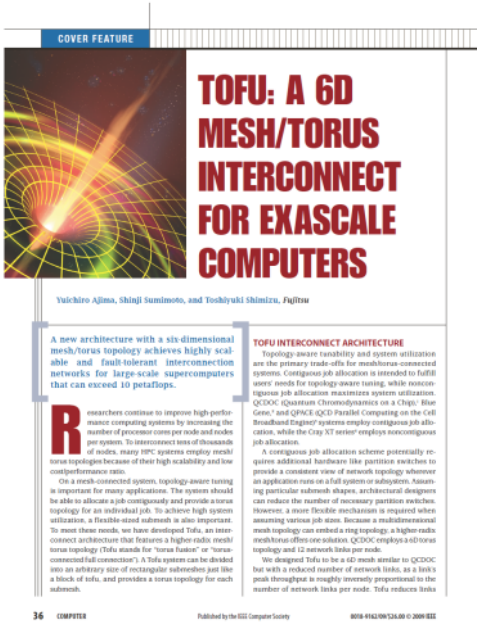
- 3次元トーラスを超える高いスケーラビリティを実現
- 高い信頼性と運用性を実現
- 高次元化で平均ホップ数、バイセクション帯域も改善

3次元トーラス
最大32×32×32

「京」での実装は 24x*18* x17
Z=0：IOノード
Y軸：*メッシュ*



3次元*メッシュ*/トーラス
2*×3×*2



COVER FEATURE

TOFU: A 6D MESH/TORUS INTERCONNECT FOR EXASCALE COMPUTERS

Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu, Fujitsu

A new architecture with a six-dimensional mesh/torus topology achieves highly scalable and fault-tolerant interconnection networks for large-scale supercomputers that can exceed 10 petaflops.

Researchers continue to improve high-performance computing systems by increasing the number of processor cores per node and nodes per system. To interconnect tens of thousands of nodes, many HPC systems employ mesh/torus topologies because of their high scalability and low cost/performance ratio.

On a mesh-connected system, topology-aware tuning is important for many applications. The system should be able to allocate a job contiguously and provide a torus topology for an individual job. To achieve high system utilization, a flexible-sized submesh is also important. To meet these needs, we have developed Tofu, an interconnect architecture that features a higher-radix mesh/torus topology (Tofu stands for "torus fusion" or "torus-connected full connection"). A Tofu system can be divided into an arbitrary size of rectangular submeshes just like a block of tofu, and provides a torus topology for each submesh.

TOFU INTERCONNECT ARCHITECTURE

Topology-aware tunability and system utilization are the primary trade-offs for mesh/torus-connected systems. Contiguous job allocation is intended to fulfill users' needs for topology-aware tuning, while noncontiguous job allocation maximizes system utilization. QCDOC (Quantum Chromodynamics on a Chip),[1] Blue Gene,[2] and QPACE (QCD Parallel Computing on the Cell Broadband Engine)[3] systems employ contiguous job allocation, while the Cray XT series[4] employs noncontiguous job allocation.

A contiguous job allocation scheme potentially requires additional hardware like partition switches to provide a consistent view of network topology wherever an application runs on a full system or subsystem. Assuming particular submesh shapes, architectural designers can reduce the number of necessary partition switches. However, a more flexible mechanism is required when assuming various job sizes. Because a multidimensional mesh topology can embed a ring topology, a higher-radix mesh/torus offers one solution. QCDOC employs a 6D torus topology and 12 network links per node.

We designed Tofu to be a 6D mesh similar to QCDOC but with a reduced number of network links, as a link's peak throughput is roughly inversely proportional to the number of network links per node. Tofu reduces links

36 COMPUTER

Published by the IEEE Computer Society

0018-9162/09/$26.00 © 2009 IEEE

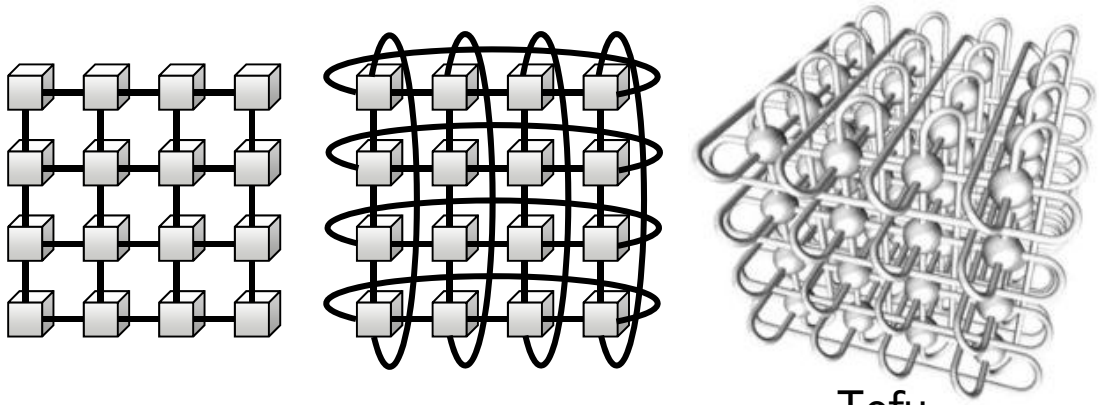
Tofuインターコネクト技術の論文が
IEEE Computer（2009年）に掲載

Copyright 2012 FUJITSU LIMITED

インターコネクトのネットワークトポロジー

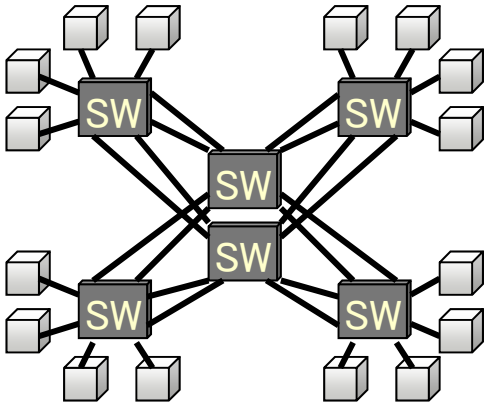
FUJITSU

直接網
（ノード同士が相互に接続）



2次元メッシュ　2次元トーラス　Tofu
（6次元メッシュ/トーラス）

間接網
（ノードをスイッチで接続）



2段Fat-Tree

	直接網			間接網
	3次元メッシュ	3次元トーラス	Tofu	3段Fat-tree
スケーラビリティ	○	○	○	×
信頼性・運用性	×	×	○	○
平均ホップ数	×	△	○**	◎
バイセクション帯域 *	×	△	○(トーラス)	◎

* ネットワークのボトルネック部分の帯域

**Tofuの6次元メッシュ/トーラスはショートカットパスを有する

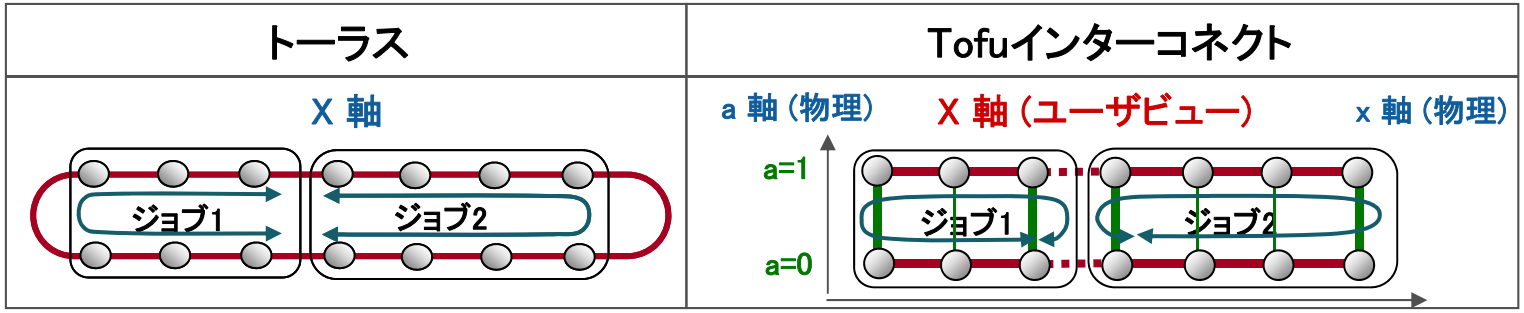
Copyright 2012 FUJITSU LIMITED

Tofuインターコネクトの運用性、信頼性

FUJITSU

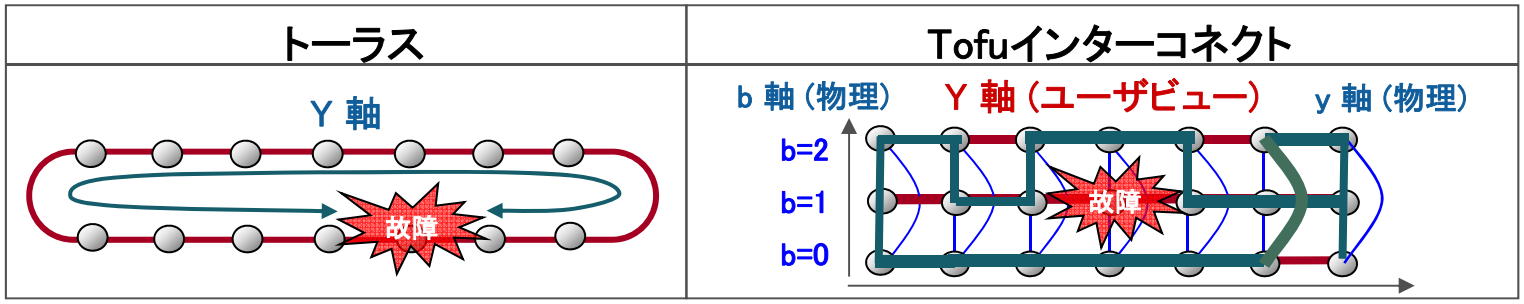
■複数ジョブ実行時

- ■トーラス: 分割するとメッシュになる
- ■Tofu: 分割してもトーラス維持



■ノード故障時

- ■トーラス: メッシュになる
- ■Tofu: 故障を回避してトーラス維持

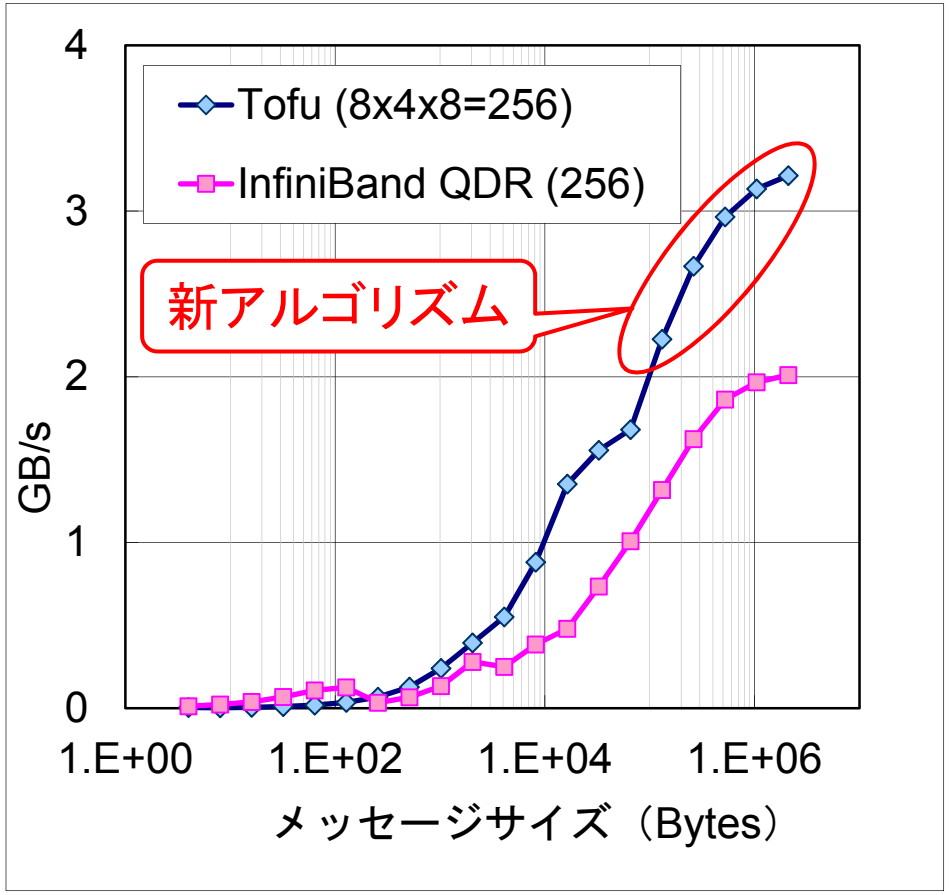


Copyright 2012 FUJITSU LIMITED

Alltoall通信の性能

FUJITSU

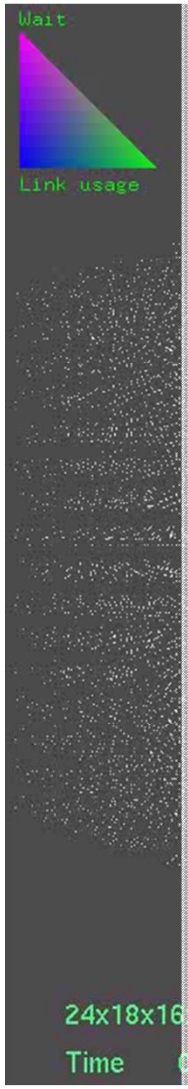
- Alltoall通信では、全リンクの利用率を高く維持することが重要
- Tofuを活用する新アルゴリズムを開発
  - 5GB/s × 双方向 × 10リンクの高バンド幅を万遍なく利用
  - 4送信＋4受信を同時に実行
- 256ノードの実測でFat-Tree（InfiniBand QDR）を凌駕
  - 既存アプリの移植が容易
- HPCチャレンジ賞獲得に貢献
  - 並列FFTでAlltoallの高速化が有効



Copyright 2012 FUJITSU LIMITED

Tofu上でのAlltoall通信の様子（動画）-2

FUJITSU



実行条件:

24×18×16×2×3×2 ＝ 82,944 ノード

MPI_Alltoall 32KB/ノードペア

左: 新アルゴリズム

実行時間:2.77秒（効率88.3%）

右: OpenMPI標準（Pair-wise exchange）

実行時間:24.08秒（効率10.2%）

リンク使用状況を色で表示

緑:　リンク使用率高

青:　リンク使用率低

紫:　待ち時間によりリンク使用率低

Copyright 2012 FUJITSU LIMITED

システムソフトウェアスタック

FUJITSU

- すべて自前で開発サポート（OSSベース開発製品も含む）
- PCクラスタおよびハイブリッドシステムも同一環境を提供

アプリケーション（高並列アプリ・アルゴリズムの研究開発）

HPC Portal / System Management Portal

Technical Computing Suite

システム管理ソフト
- システムの管理、制御、監視、運用サポート
- 故障の自動リカバリを実現（365日24時間運用）

ジョブ管理ソフト
- 高速並列ジョブ起動
- 高効率スケジューラ
- 充実したセンタ運用機能

エクサバイト対応 高性能ファイルシステム（FEFS）
- Lustreをベースに拡張
- 高いスケーラビリティ（IOサーバ数千台に対応）
- 1TB/sクラスのIO転送性能実績

Whamcloudとコラボ

自動並列化コンパイラ
- Fortran, C, C++を提供
- 高レベルのSMID並列、マルチコア並列をサポート

ツール・数学ライブラリ
- 多くの海外ツールをサポート
- 高効率の数学ライブラリ（SSL II/BLAS etc.）

並列言語・通信ライブラリ
- OpenMP, MPI(Open MPIベース), XPFortranを提供
- マルチコア、超並列に対応

Linux based OS (Enhanced for FX10)：OSノイズ対応

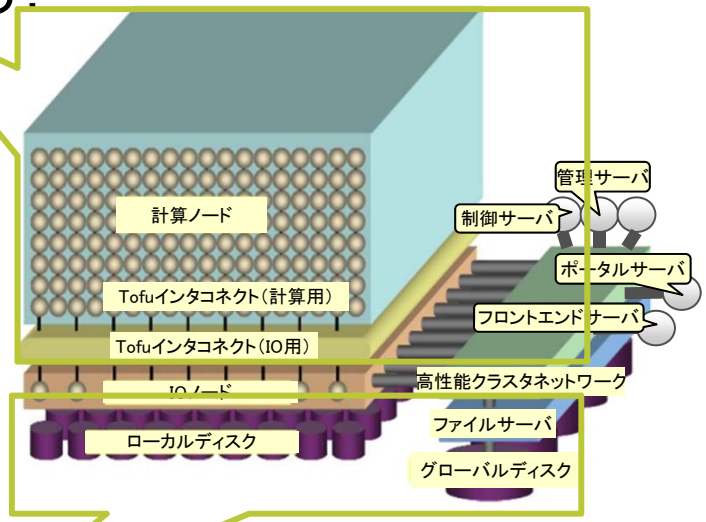
「京」/ PRIMEHPC FX10 / PCクラスタシステム

Copyright 2012 FUJITSU LIMITED

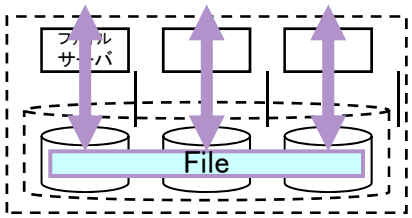
超並列対応のOS/運用管理/FEFS

FUJITSU

- 9万ノードでシングルジョブ実行可能にする：OS/運用管理機能
  - 性能ブレを防ぐためOSノイズを極限まで削減
  - 故障に対して自動復旧し、24時間365日間システムを止めないシステム運用管理（計算ノード、IOサーバ群、ストレージ群、ネットワーク機器群）
  - ワンタッチで全システムを起動・停止・動的構成変更するシステム制御



- 10PFLOPSの計算パワーに相応しい超大規模クラスタファイルシステム：FEFS
  - 次の世代を見越しエクサバイトまで拡張可能
  - IOノード群(5000台以上)、ストレージ群(ETERNUS 4000台以上, Disk 90000個以上)を超大規模ファイルシステムとして安定運用
  - 世界初1TB/s超の実効ストレージバンド幅達成



クラスタファイルシステム（FEFS）

OSS数	80%実測	100%予測
Read	1.31 TB/s	1.68 TB/s
Write	0.83 TB/s	1.06 TB/s

IORベンチマーク性能

Copyright 2012 FUJITSU LIMITED

言語

FUJITSU

■ コンパイラ

■ 言語仕様のエンハンス

- F2003、C99、C++ 03に加えてGCC互換性を改善、更にF2008、C++ 11を開発中

■ 逐次性能の改善

- 256個の浮動小数点レジスタを活用するソフトウェアパイプライニングの強化
- SIMD化を強化
  - 自動ベクトル化を発展させたVSIMD機能
  - 同一回転内の演算をSIMD化するUXSIMD機能

■ スレッド並列性能の改善

- 自動ベクトル化を発展させた自動並列化で、世界最高レベルの並列化機能を実現

ANLベクトル化コンテスト 並列化ループ（135個中）	並列化	IN社並列化	IB社並列化	ベクトル化	エンハンス目標
	81個※1	78個	71個	85個	102個

※1 2012年12月出荷予定

■ MPIライブラリ

■ Open MPIをベースとしてTofuの性能を最大限に引き出せるように拡張

- バイパスの追加によるレイテンシィの改善（4.12μs → 1.26μs）
- 集団通信（Bcast、Allreduce、Reduce、Allgather、Allgatherv、Alltoall、Alltoallv）でTofuのトポロジーを活かすアルゴリズムを実装

Copyright 2012 FUJITSU LIMITED

超並列システムの新たな成果

Copyright 2012 FUJITSU LIMITED

京・FX10で得られた新たな成果

FUJITSU

1. 複数の実アプリでピーク比効率で30%を超える性能（ベクトル機並みの効率）を達成
   - HPC-ACEを利用してソフトウェアパイプライニング化を追求
   - PAカウンタ情報を利用したチューニング技術の確立

2. 複数の実アプリでVISIMPACTによるハイブリッド並列がFlat MPIを上回る性能を達成

Copyright 2012 FUJITSU LIMITED

1. 大規模アプリケーションの実行効率

FUJITSU

■ 京/FX10で測定した実アプリの実行性能状況

■ 11本中6本が30%以上のピーク比性能を達成

アプリ名	使用コア数	演算効率		参考）FX1のピーク性能比
		ピーク性能比	IPC	
アプリA*	98,304	30%	1.7	8%
アプリB*	98,304	41%	1.5	18%
アプリC*	98,304	32%	1.3	5%
アプリD*	98,304	27%	1.0	
アプリE*	98,304	12%	0.6	
アプリF*	98,304	52%	1.5	
アプリG*	98,304	9%	0.7	
アプリH*	20,480	3%	0.6	
アプリI*	32,768	23%	0.9	
アプリJ*	98,304	40%	1.5	
アプリK	6,144	30%	1.6	15%

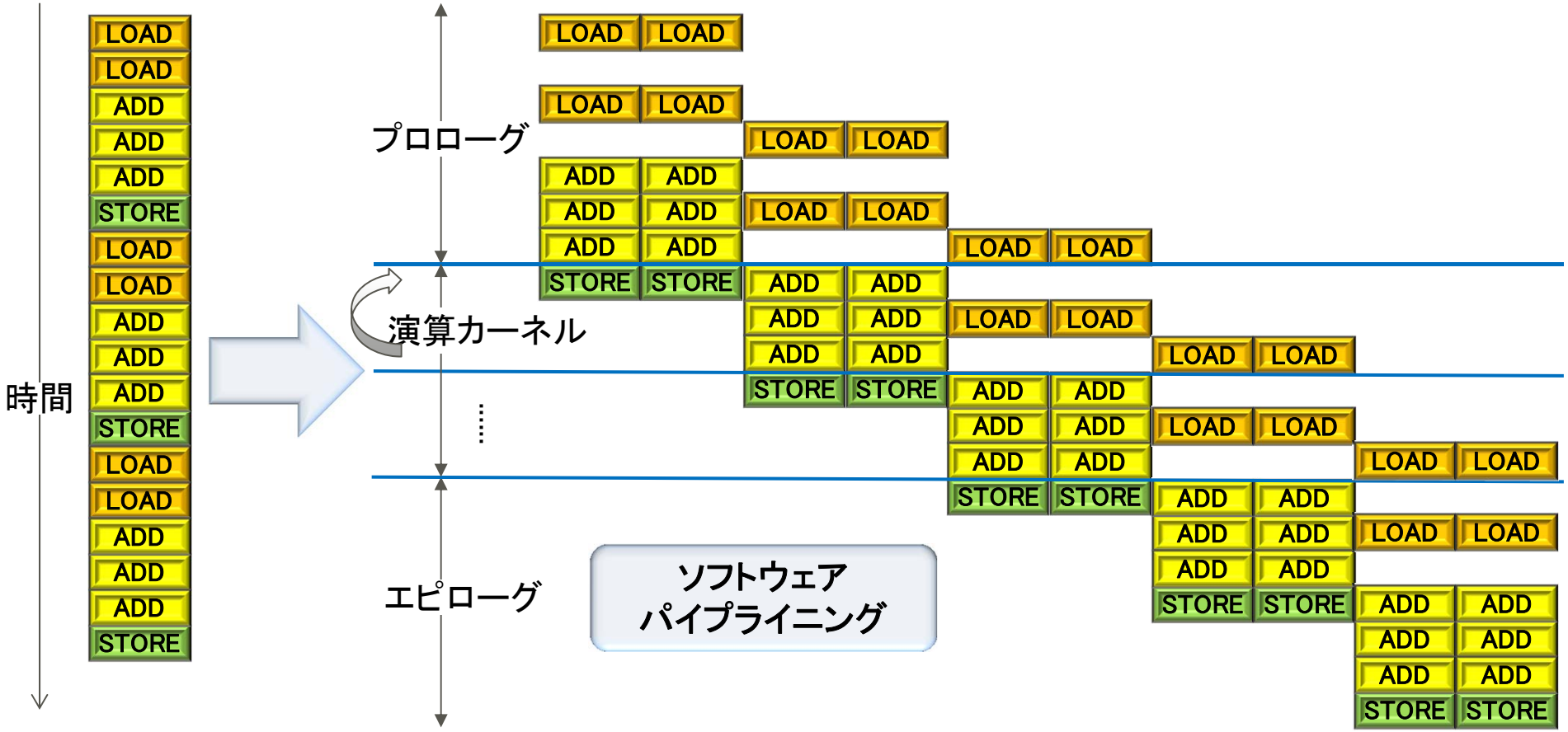
*京での実行：試行期間中のものであり最終状況ではありません。

Copyright 2012 FUJITSU LIMITED

ソフトウェアパイプラニング化の追求

FUJITSU

- HPC-ACEを利用してコンパイラによるソフトウェアパイプライニングを促進することで、演算器の利用効率が向上
  - 演算数多いループでアウトオブオーダ資源増強より優位
  - HPC-ACE：マスク付SIMD命令、レジスタ拡張、逆数近似命令

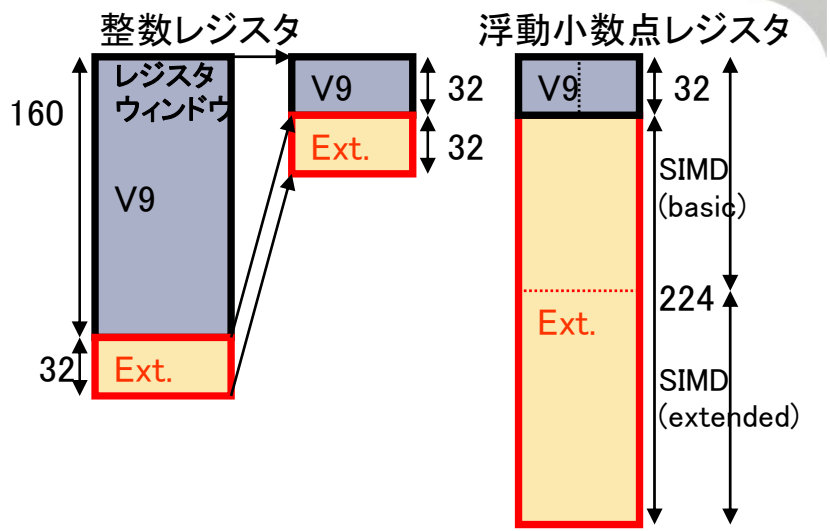


Copyright 2012 FUJITSU LIMITED

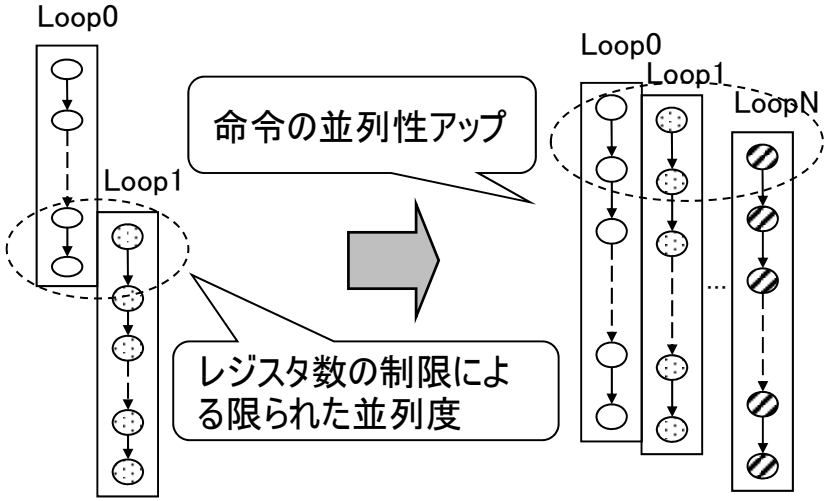
レジスタ拡張

FUJITSU

- SPARC V9仕様を拡張
  - 整数レジスタ　32個→64個
  - 倍精度浮動小数点レジスタ 32個→256個
    - 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
      - 拡張レジスタも非SIMD命令からアクセス可能



- レジスタ拡張の狙い
  - レジスタ数による制約で制限されている並列度の向上
  - レジスタのspill/fillによるオーバヘッドを削減



レジスタ数を増やすことで命令の並列性を増やし、大規模ループのソフトウェアパイプライニングを促進

Copyright 2012 FUJITSU LIMITED

マスク付きSIMD命令

FUJITSU

- 条件分岐（IF文）の問題
  - コンパイラの命令スケジューリングが阻害される
  - 命令パイプライン中の命令がキャンセルされる

➔マスク付きSIMD化を行うことで、条件分岐を除去

```
subroutine sub(a, b, c, x, n)
 real*8  a(n), b(n), c(n), x(n)
 do i = 1 , n
   if ( x(i) .gt. 0.0 ) then
      a(i) = b(i) * c(i)
    else
      a(i) = b(i) - c(i)
   endif
 enddo
 end
```

マスク付き
SIMD化

```
vgt        x(i:i+1) , 0.0, %vr1   } マスク生成
vnot1      %vr1, %vr2             }
  : 演算
vmaskstore %vr3, %vr1, a(i:i+1)   } マスク付き
vmaskstore %vr4, %vr2, a(i:i+1)   } ストア
```

マスク付き命令を使用して条件分岐を除去して命令列を
ストレート化することで、ソフトウェアパイプライニングを促進

Copyright 2012 FUJITSU LIMITED

逆数近似命令

FUJITSU

■ 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

```
REAL(KIND=8),DIMENSION(1024) :: A, B, C
DO I=1,1024
  A(I) = B(I) / SQRT(C(I))
END DO
```

通常命令のみ

```
.LL1:
    ldd       [%o2 + %g2], %f2
    ldd       [%o1 + %g2], %f0
    subcc     %g3, 1, %g3
    add       %g2, 8, %g1
    fsqrtd    %f2, %f2
    fdivd     %f0, %f2, %f0
    std       %f0, [%o0 + %g2]
    mov       %g1, %g2
    bne,pt    %icc, .LL1
    nop
```

逆数近似命令適用

```
.LL1:
    ldd       [%o2 + %g2], %f2
    ldd       [%o1 + %g2], %f0
    subcc     %g3, 1, %g3
    add       %g2, 8, %g1
    fmuld     %f8, %f2, %f10
    frsqrtad  %f2, %f2
    fmuld     %f10, %f2, %f14
    fnmsubd   %f2, %f14, %f8, %f14
    fmaddd    %f2, %f14, %f2, %f2
    fmuld     %f10, %f2, %f12
    fnmsubd   %f2, %f12, %f8, %f12
    fmaddd    %f2, %f12, %f2, %f2
    fmuld     %f10, %f2, %f10
    fnmsubd   %f2, %f10, %f8, %f6
    fcmpeqd   %f10, %f16, %f18
    fmaddd    %f2, %f6, %f2, %f2
    fselmovd  %f4, %f2, %f18, %f2
    fmuld     %f0, %f2, %f0
    std       %f0, [%o0 + %g2]
    mov       %g1, %g2
    bne,pt    %icc, .LL1
    nop
```

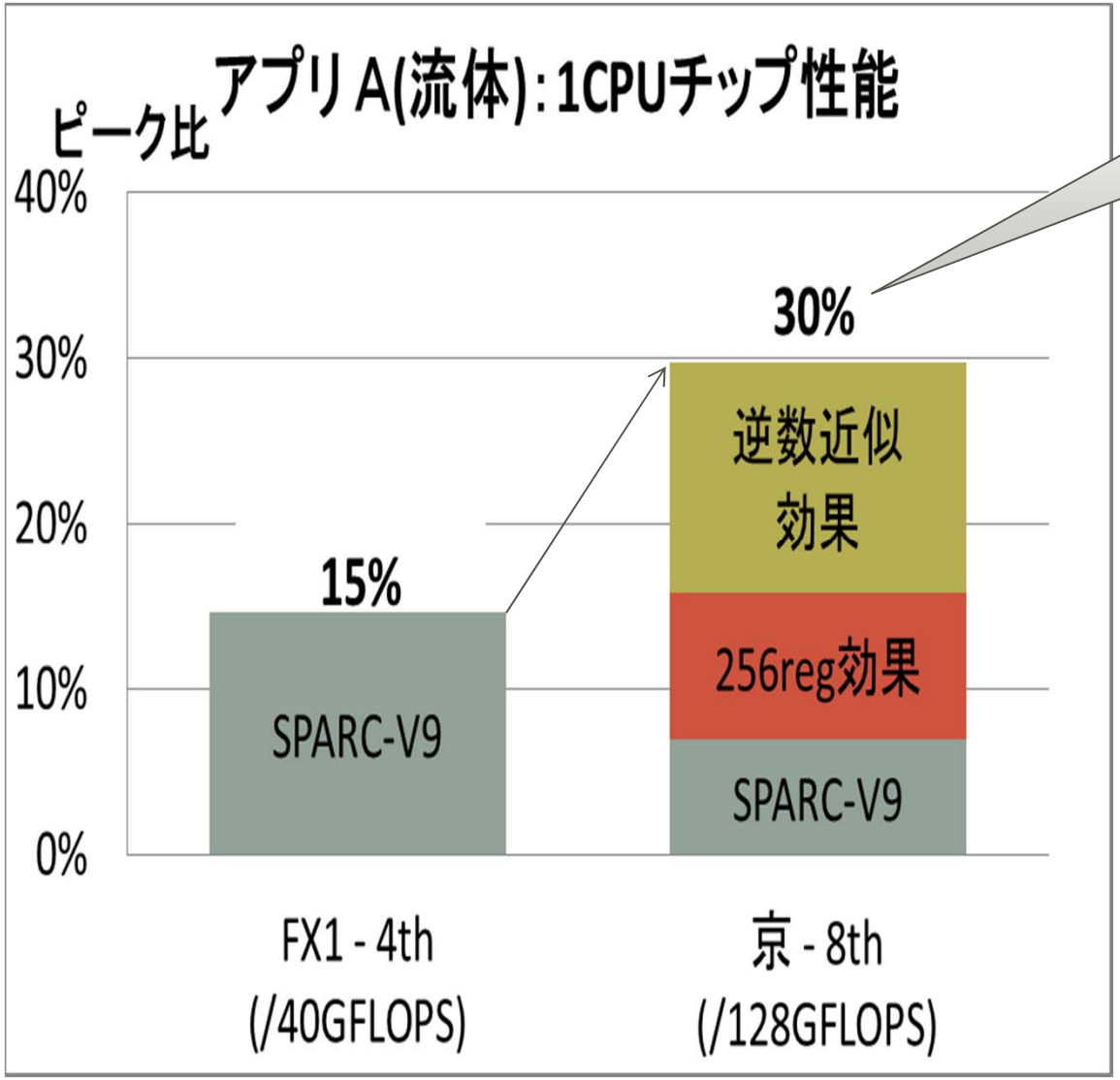
パイプライン動作できる逆数近似命令を使用することで、
除算・平方根を含むループのソフトウェアパイプライニングを促進

Copyright 2012 FUJITSU LIMITED

アプリの実効効率　FX1と京の比較（1）

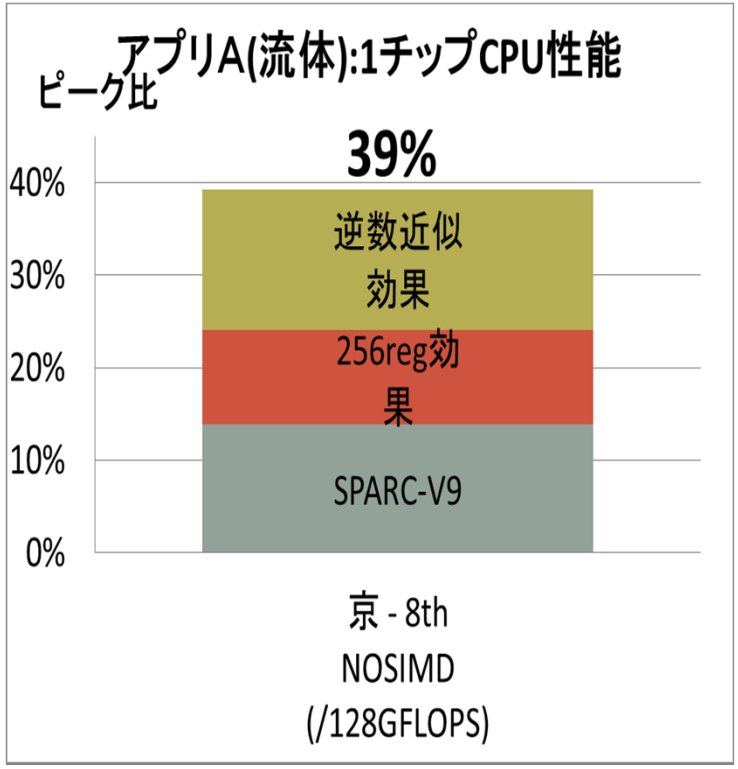
FUJITSU

HPC-ACEの利用で実行効率大幅UP!



SIMD/マルチコア並列
でも実行効率30%超え

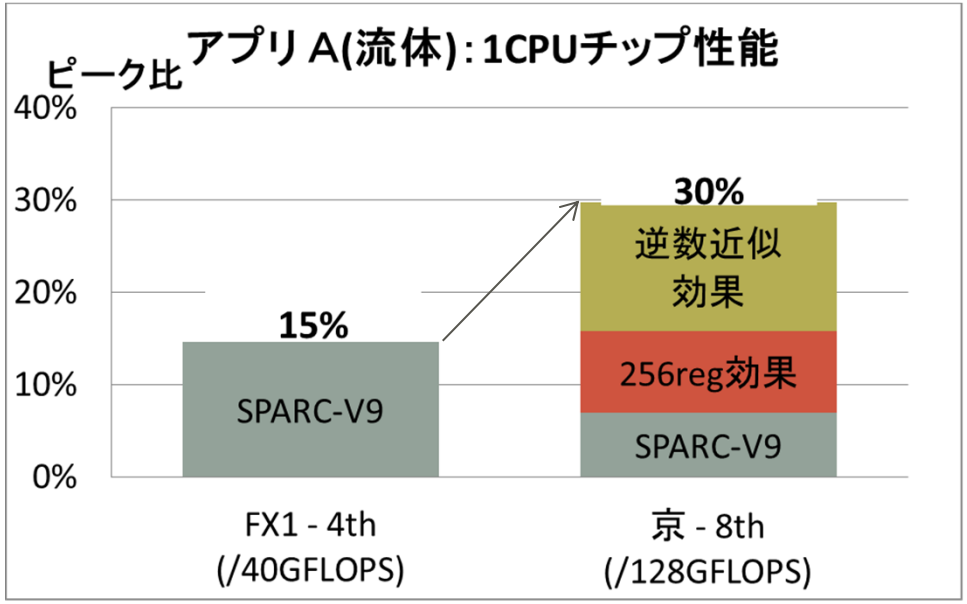
参考）SIMDなしの場合

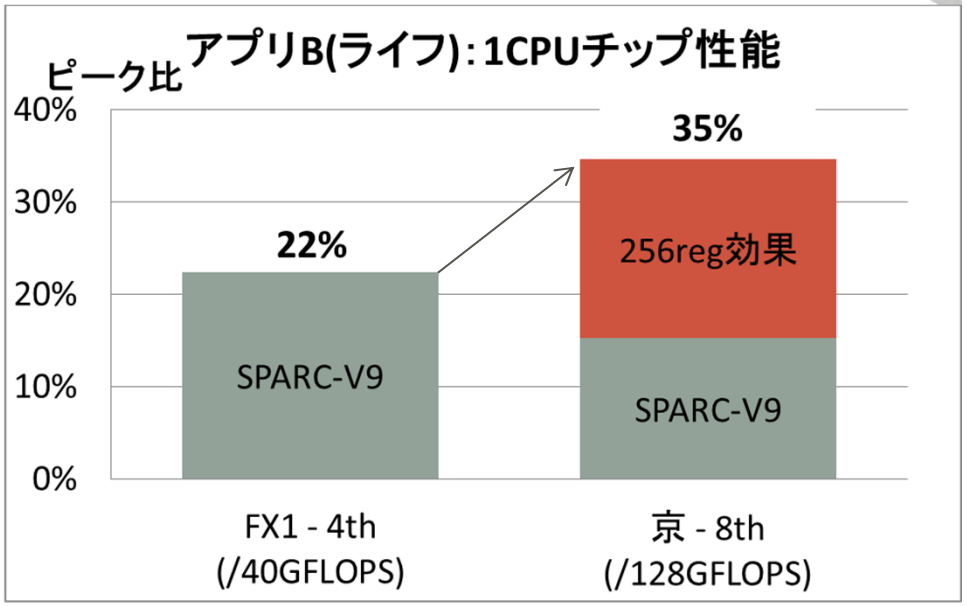


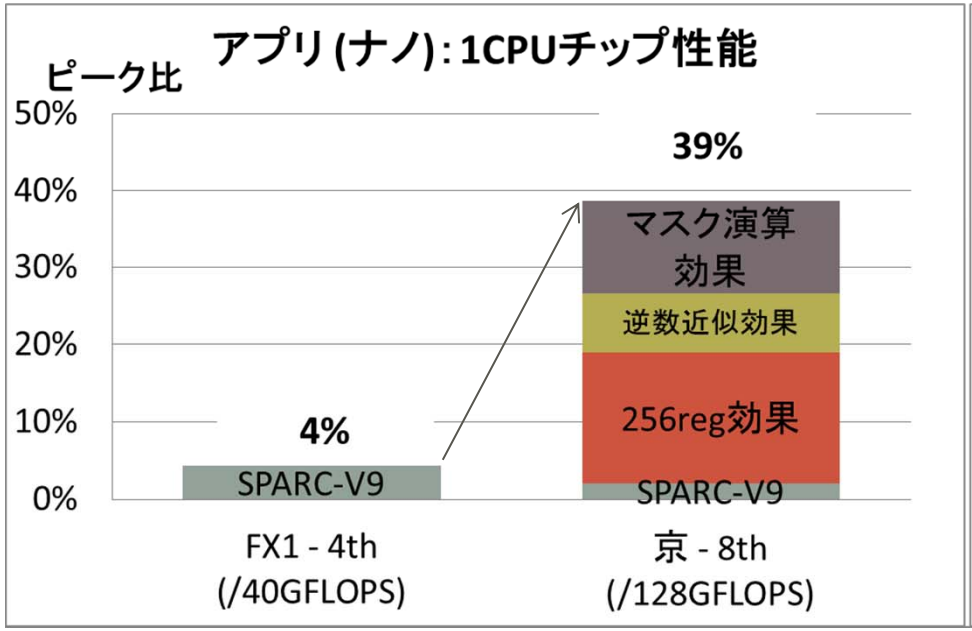
Copyright 2012 FUJITSU LIMITED

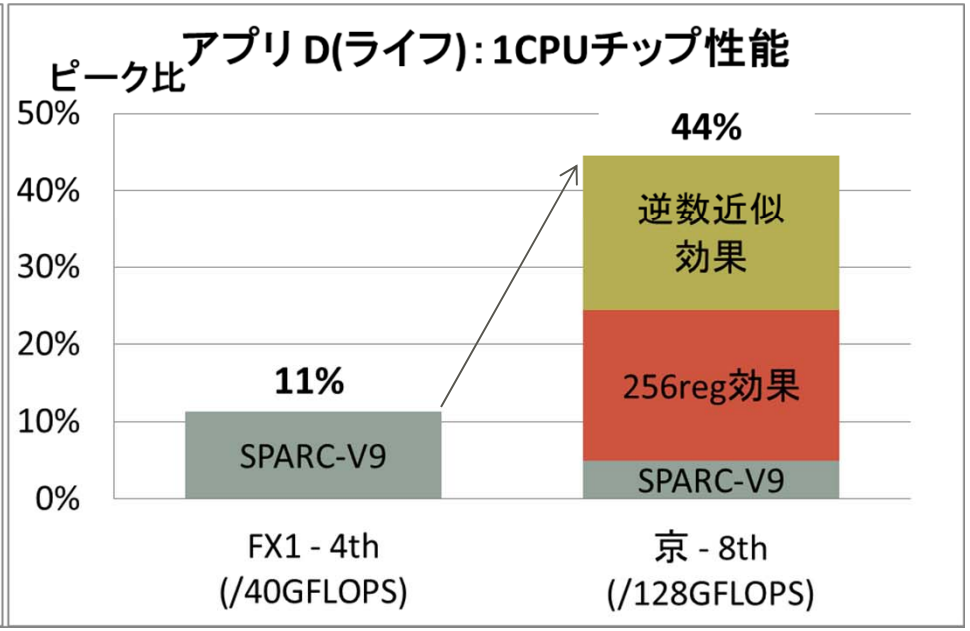
アプリの実効効率　FX1と京の比較（2）

FUJITSU









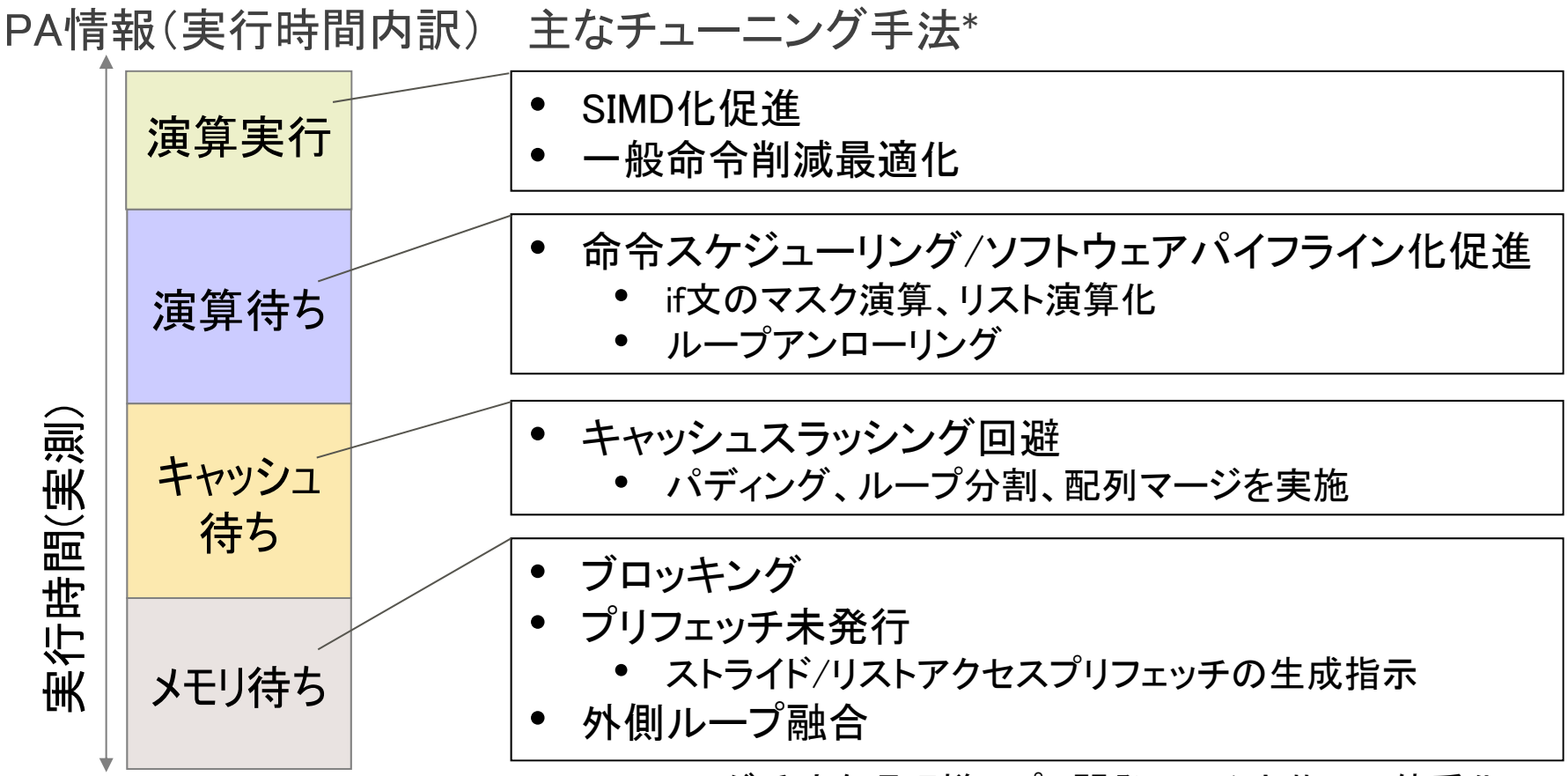
Copyright 2012 FUJITSU LIMITED

チューニング技術確立（1）

FUJITSU

■ CPU-PAを活用した性能分析ツールの進化

- 性能問題の原因特定の容易化⇒規定のチューニング手法適応
- 特にアクセス系は、コンパイラでの見極めが難しいため、PA分析が有用



* チューニング手法を理研様アプリ開発チームと共同で体系化

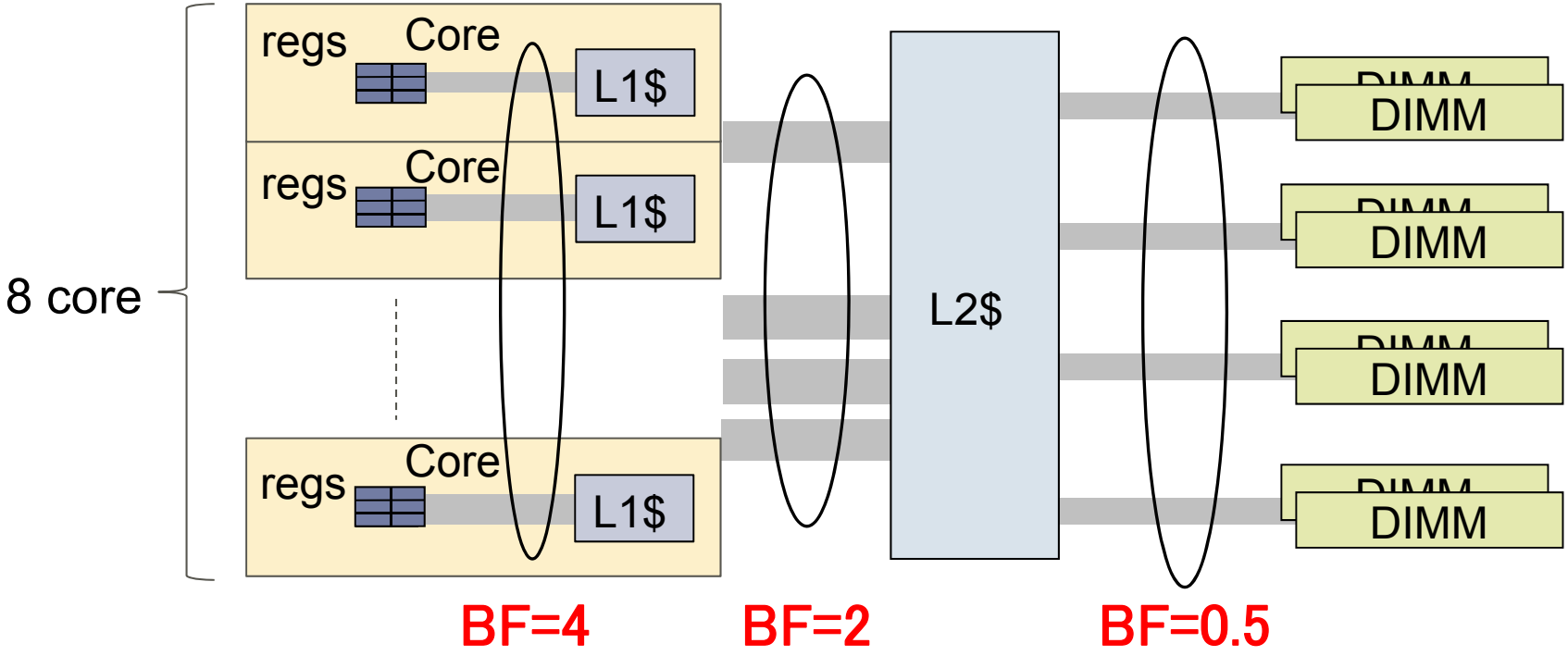
Copyright 2012 FUJITSU LIMITED

チューニング技術確立（2）

FUJITSU

■ 各種バンド幅のバランスを考慮したハード依存のチューニング適応

	「京」バンド幅	バンド幅計算式
L1D$	512 GB/sec	2 SIMD load / cycle → 2 * 16B * 8 core * 2GHz
L2$	256 GB/sec	2 core 共有, 32B幅bus → 32B * 4 Bus * 2GHz
Memory	64 GB/sec	8B * 8 Bus * 1GHz

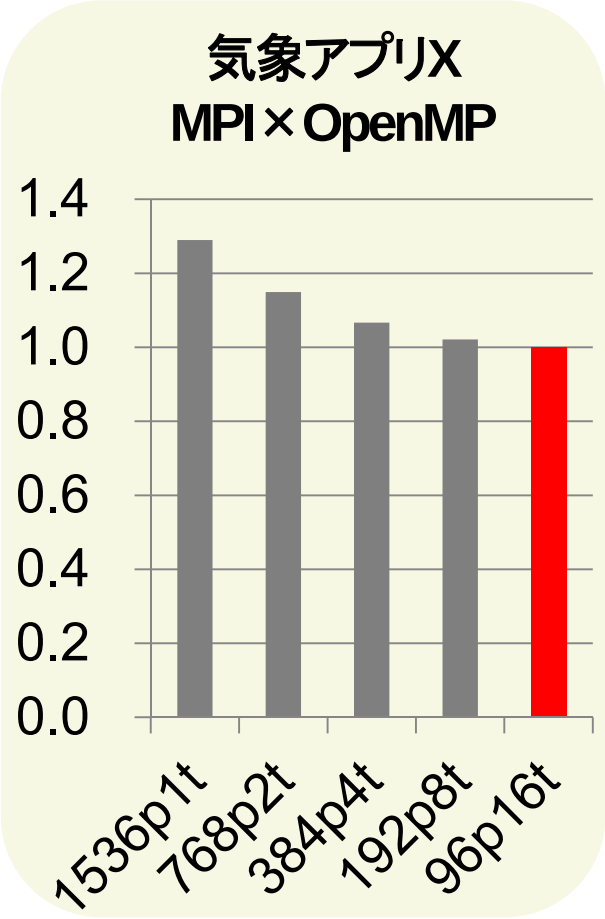


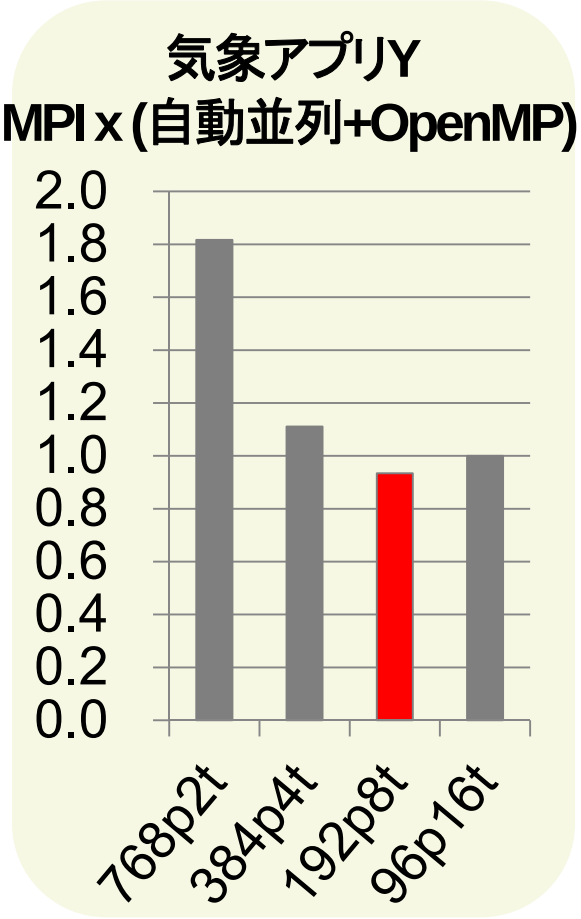
Copyright 2012 FUJITSU LIMITED

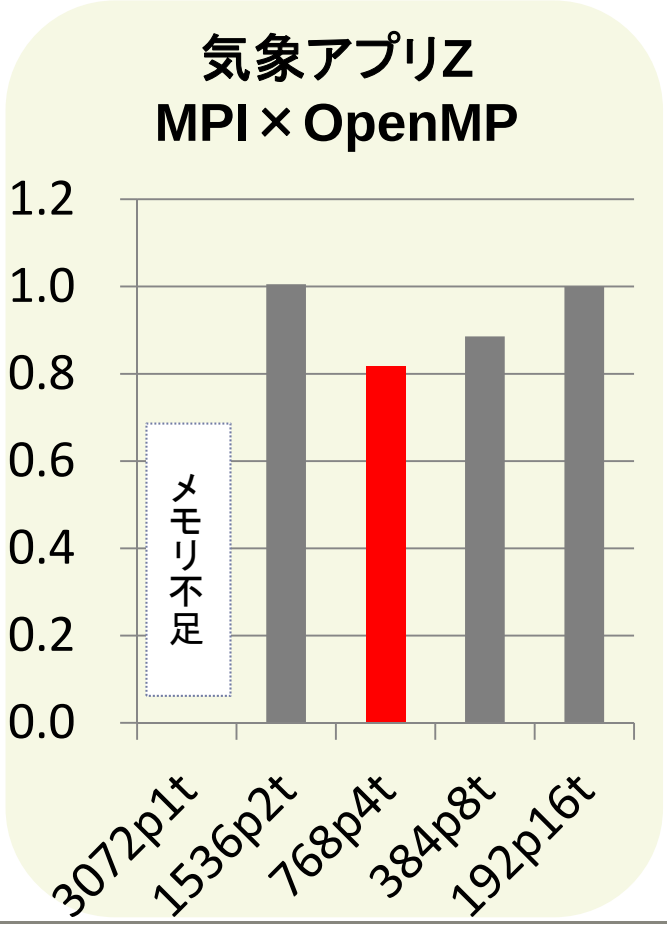
2. ハイブリッド並列の優位性（対FlatMPI）

FUJITSU

- FX10上で気象アプリをCore数を変えずに、プロセス数（P）とスレッド数（T）の組み合わせによる実行時間比
- アプリにより、最適なプロセス・スレッドの組み合わせ







Copyright 2012 FUJITSU LIMITED

ハイブリッド並列　FX10 アプリ実行例

FUJITSU

■ ハイブリッド並列化で、アプリの特性に合わせて適切なプロセス数・スレッド数を選択することで最速な実行条件を選択

アプリ名	プログラミング	ノード内プロセス×スレッド構成					備考（主要因）
		1P16T	2P8T	4P4T	8P2T	16P1T	
アプリ1	MPI *		○				ロードインバランス
アプリ2	MPI+OpenMP	○					ロードインバランス
アプリ3	MPI					○	スレッド粒度小
NPB.CG	MPI	○					通信時間、共用キャッシュ効果
NPB.LU	MPI					○	スレッド粒度小、キャッシュスラッシング
アプリ6	MPI+OpenMP		○				通信時間
アプリ7	MPI+OpenMP				○		ロードインバランス、通信時間
アプリ8	MPI *			○			ロードインバランス、通信時間
アプリ9	MPI+OpenMP			○			ロードインバランス、スレッド化率
アプリ10	MPI *	○					通信時間
アプリ11	MPI+OpenMP				○		スレッド化率、通信時間

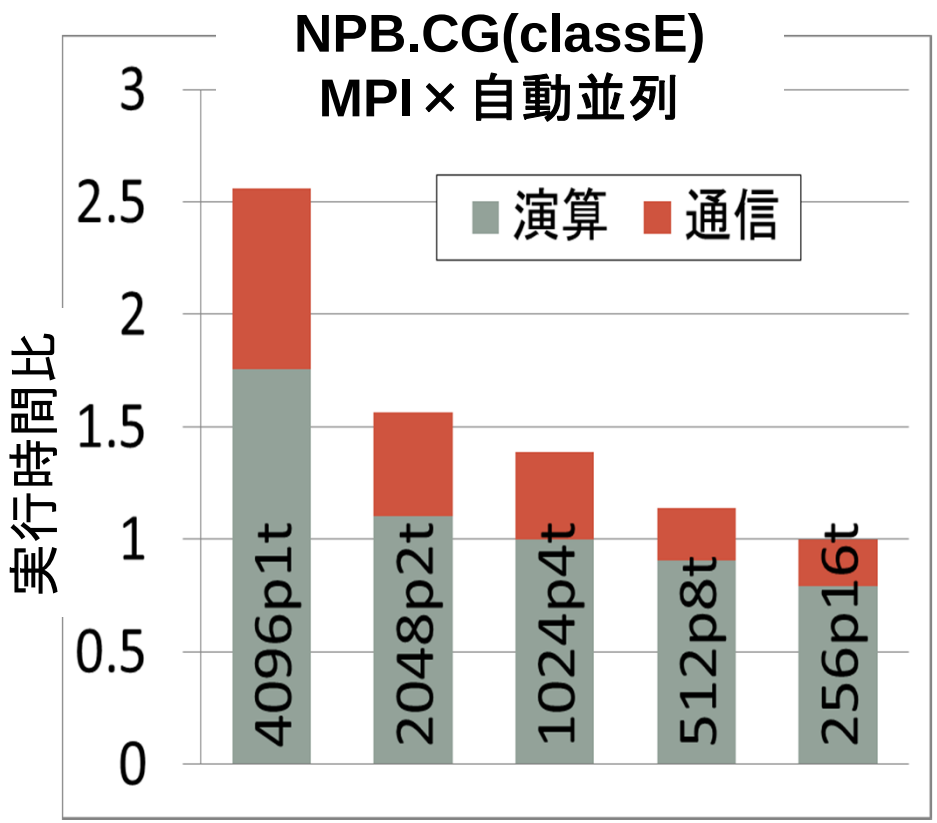
MPI * ：指示行（OpenMP）追加　　○：最速なノード内のプロセス×スレッド構成

Copyright 2012 FUJITSU LIMITED

ハイブリッド並列による演算性能改善(NPB)

FUJITSU

16T実行が良い例
理由:L2D$の再利用性(ヒット率)向上

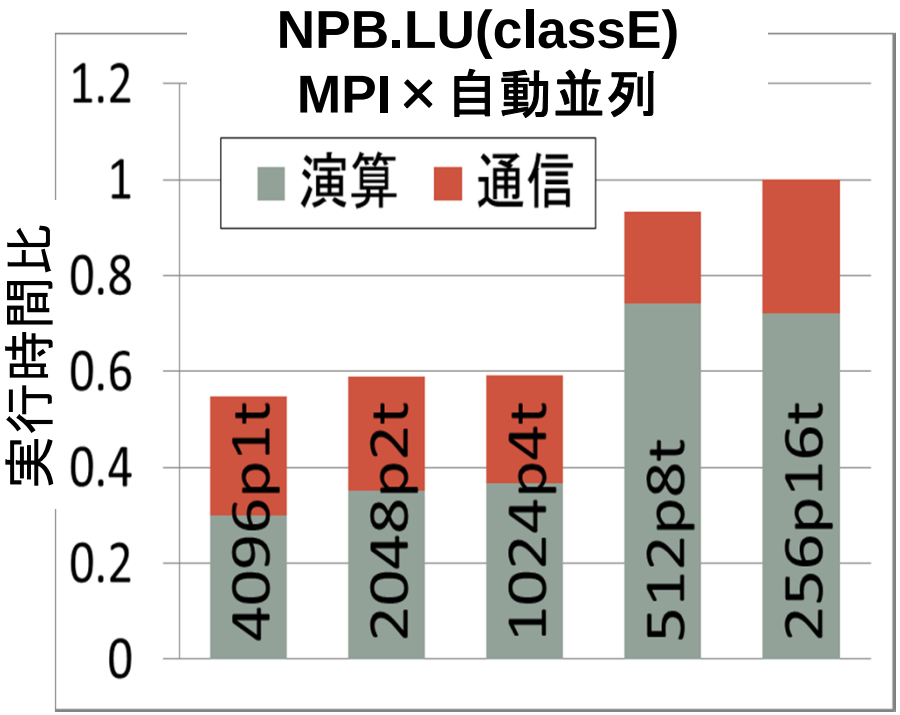


L2D$ミス率 4.4%(1t) → 2.7%(16t)

スレッド間でp(colidx(k))のキャッシュラインが再利用されるため

参考

スレッド並列が悪い例
理由:L1D$のキャッシュライン競合発生
(False sharing)



L1D$ミス率 4.3%(1t) → 11.2%(16t)

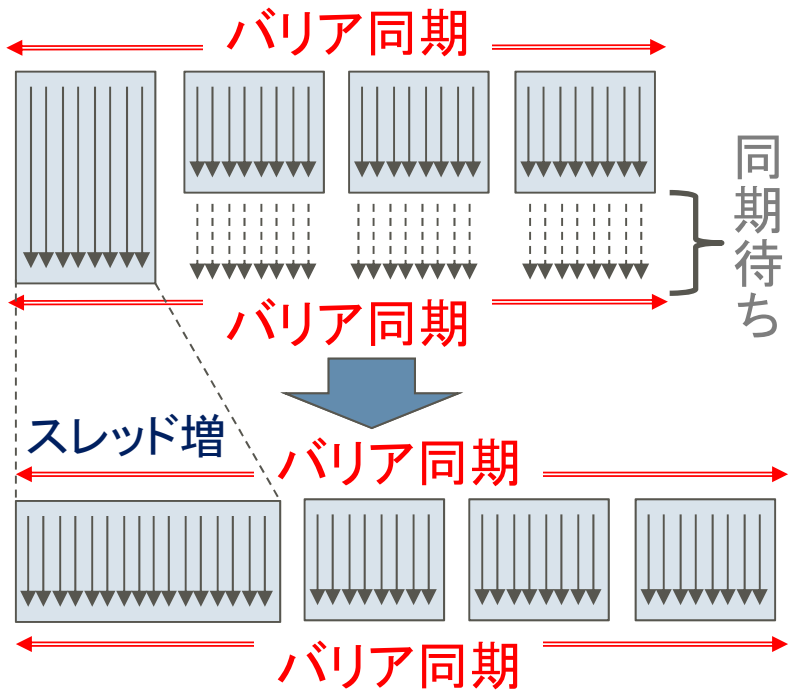
キャッシュライン書き込み競合が原因
→ Paddingで回避可能

Copyright 2012 FUJITSU LIMITED

VISIMPACTのまとめ

FUJITSU

- Flat MPIより高い実行性能であることを多数のアプリで実証
- プロセス当たりのメモリ量を大きくアサイン可能
  - メニーコア時代においては大きなアドバンテージ
- 将来に向けた新たな可能性
  - JOB内のプロセス数とスレッド数の組み合わせをヘテロ化
    - 処理量が多いプロセスのスレッド数を増やすことで、ロードインバランスを改善
    - 連成計算の各解法に適切なスレッド数を選択することで、性能を改善
  - スレッド毎に異なる処理の実行
    - スレッド並列へのタスク並列（ヘテロ）への適用
    - 演算と並行して特定スレッドに通信を割り当てることで、演算と通信のオーバーラップ化



ロードインバランスの改善

Copyright 2012 FUJITSU LIMITED

エクサスケールに向けた取り組み

Copyright 2012 FUJITSU LIMITED

エクサスケールに向けた取組み

FUJITSU

- エクサスケールシステムの実現は現在の技術の延長では困難、少なくとも二段階の開発が必要
- 最大の課題は低消費電力化と高密度実装

- 第一段階：既に100ペタフロップス級システムを開発中

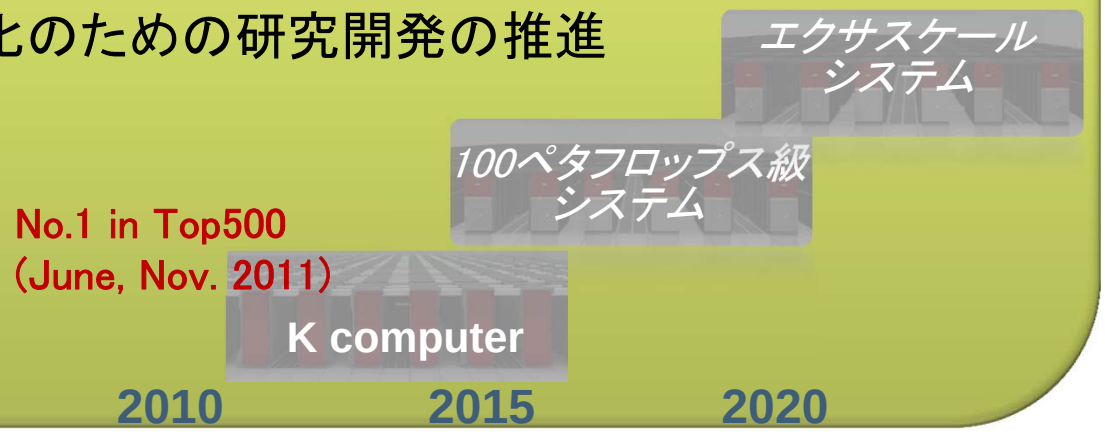
技術開発

  - さらなるCPUのマルチコア化やSIMD拡張
  - さらなるインターコネクトの高性能、低消費電力化
  - 高性能・高密度のメモリ技術

- 第二段階： エクサスケールシステムの実現

研究開発

  - さらなる高性能、低消費電力化のための研究開発の推進
  - 高信頼化のための研究開発の推進



Copyright 2012 FUJITSU LIMITED

第一段階（100ペタフロップス級システム）での重点技術開発

FUJITSU

目標

消費電力当たり性能の大幅向上
筐体当たり性能の大幅向上

取り組み

- 半導体
  ⇒最新テクノロジの追及
- メモリの革新
  ⇒最先端メモリの採用
- システム実装
  ⇒集積度、実装密度の更なる向上
- 高速伝送
  ⇒最先端光テクノロジの検討

期待される効果

- 消費電力当たり性能の大幅向上
- 筐体当たり性能の大幅向上（高密度実装）
- エクサにつながるキーテクノロジ

Copyright 2012 FUJITSU LIMITED

まとめ

FUJITSU

- 超並列システムの開発に継続的に取り組み、大規模システムを提供
  - 高性能CPUと直接網（Tofuインタコネクト）の独自開発
  - 超並列性能を向上させる機能（HPC-ACE，VISIMPACT)の導入
- 「京」で実アプリの高い実行効率とVISIMPACTの新しい成果
- 超並列アーキテクチャを更に発展させ、段階的にエクサスケールに取り組む

Copyright 2012 FUJITSU LIMITED

FUJITSU

shaping tomorrow with you