PRIMEPOWER HPC2500の ハードウェアとソフトウェア

2004年3月12日
富士通株式会社
ソフトウェア事業本部
青木　正樹
e-mail : m-aoki@jp.fujitsu.com

HPC2500のハードウェア概要

PRIMEPOWERシリーズの位置づけ

ベクトル並列
スーパー
コンピュータ

スカラ並列
スーパーコンピュータ

VX/VPP300
VPP700

VPP5000

PRIMEPOWER
HPC2500

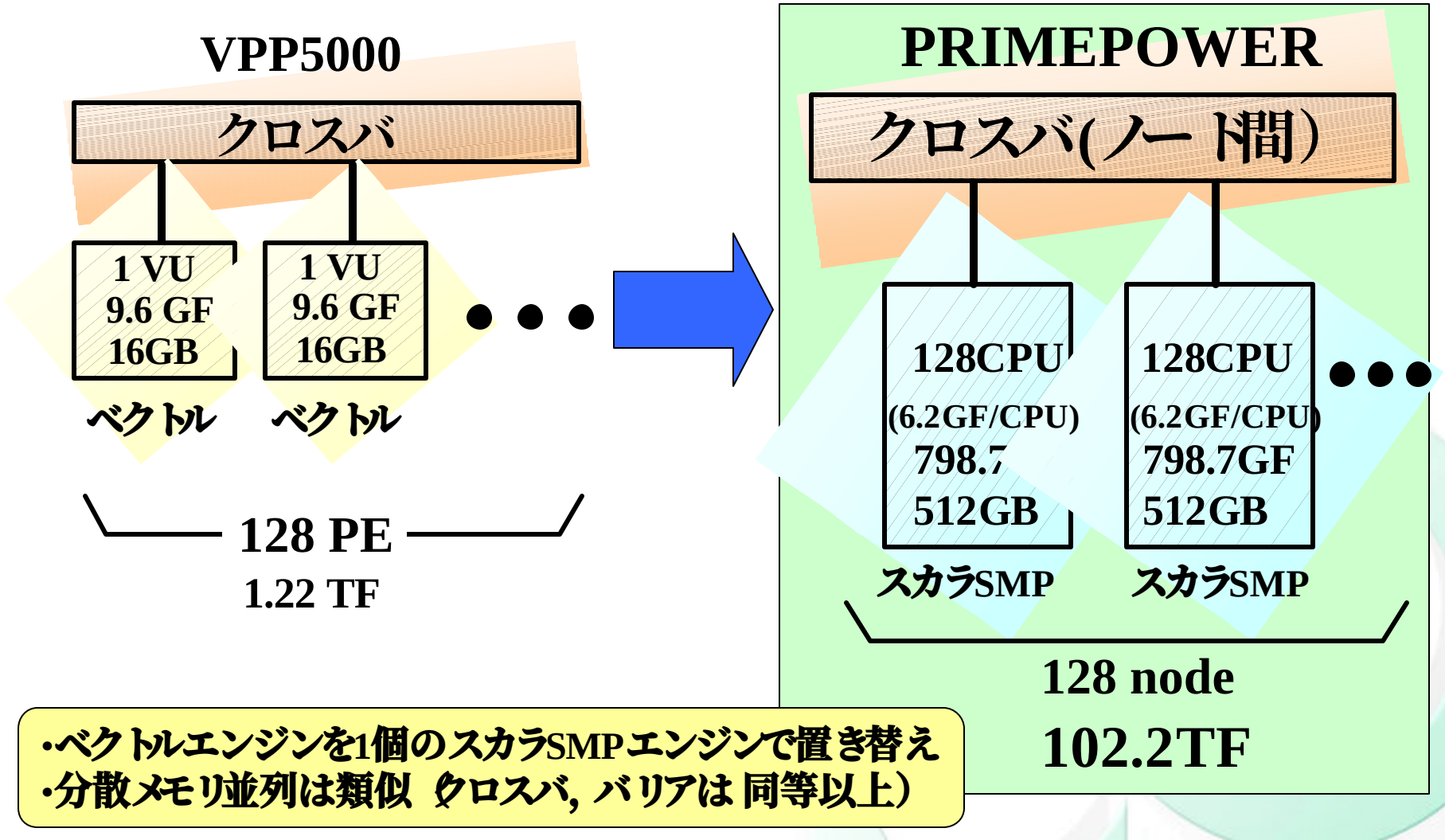
GP7000
AP3000

PRIMEPOWER
2000

ハイエンドサーバ

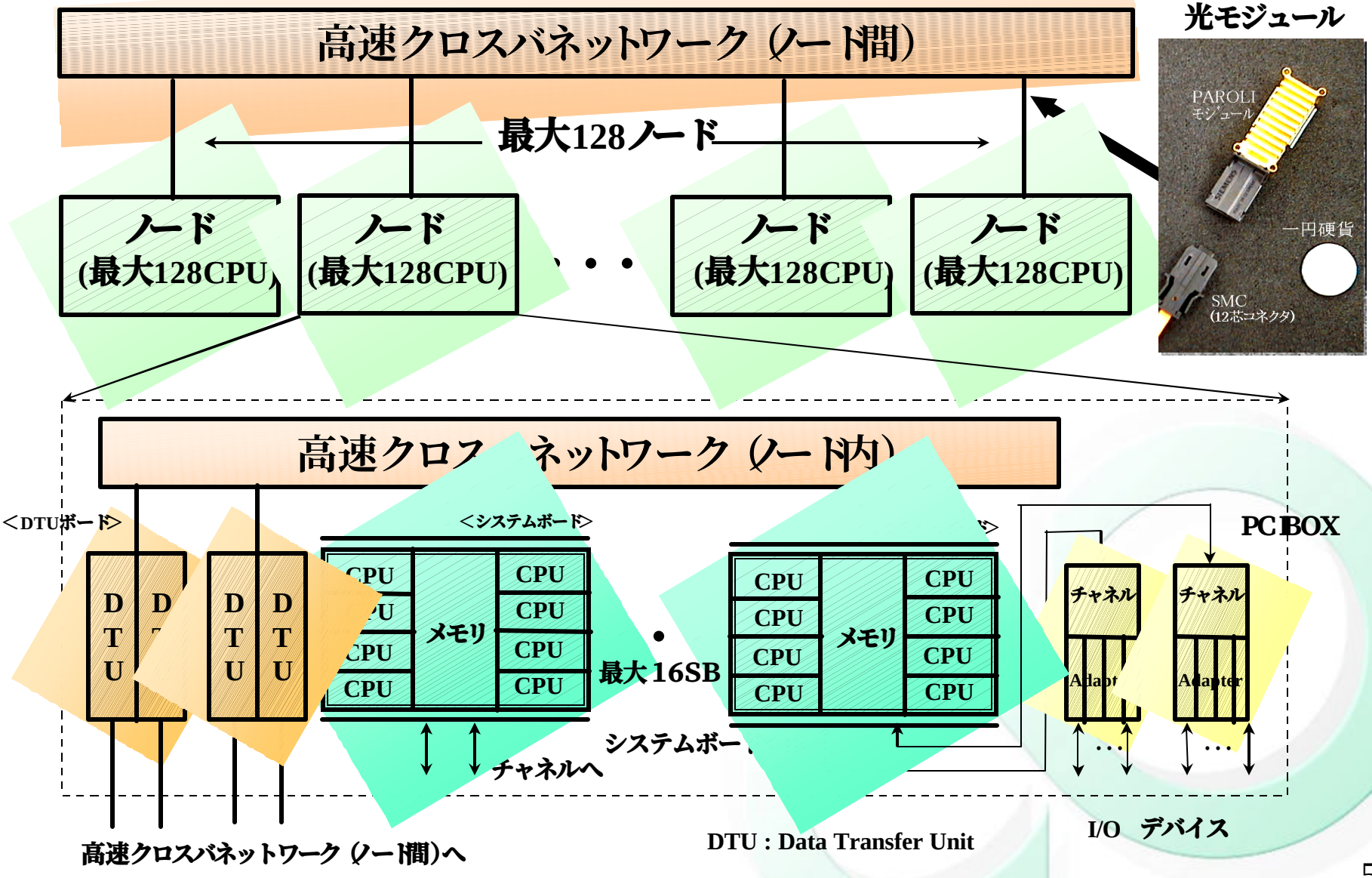
1998 1999 2000 2001 2002 2003

ベクトルからスカラSMPへ



・ベクトルエンジンを1個のスカラSMPエンジンで置き替え
・分散メモリ並列は類似（クロスバ, バリアは 同等以上）

HPC2500のシステム構成



カタログ性能（VPP5000との比較）

	HPC2500	VPP5000
最大ピーク性能		
計算ノード	798.7GFLOPS	9.6GFLOPS
システム	102.2TFLOPS	4.9TFLOPS
CPU数	8～16,384	1～512
ノード間通信	16 GB/s x 2	1.6 GB/s x 2
設置面積	2m²	40m²
消費電力	40KVA	180KVA

ピーク 82倍

ピーク 21倍

(注) 設置面積、消費電力は0.5TFLOPS構成時

HPC2500の高速化技術

改善のポイント		高速化技術
命令の同時動作不足	⇨	- 4命令同時発行 - 6演算器同時実行 - 2ロード/ストア - (M&A/M/A/DIV/√) x 2
キャッシュミスによる データ枯渇	⇨	- コンパイラ/ハードウェアによる 高度なプリフェッチ - 16-outstandingメモリアクセス - メモリバス高速化
並列実行オーバヘッド大	⇨	- ハードウェア同期機構

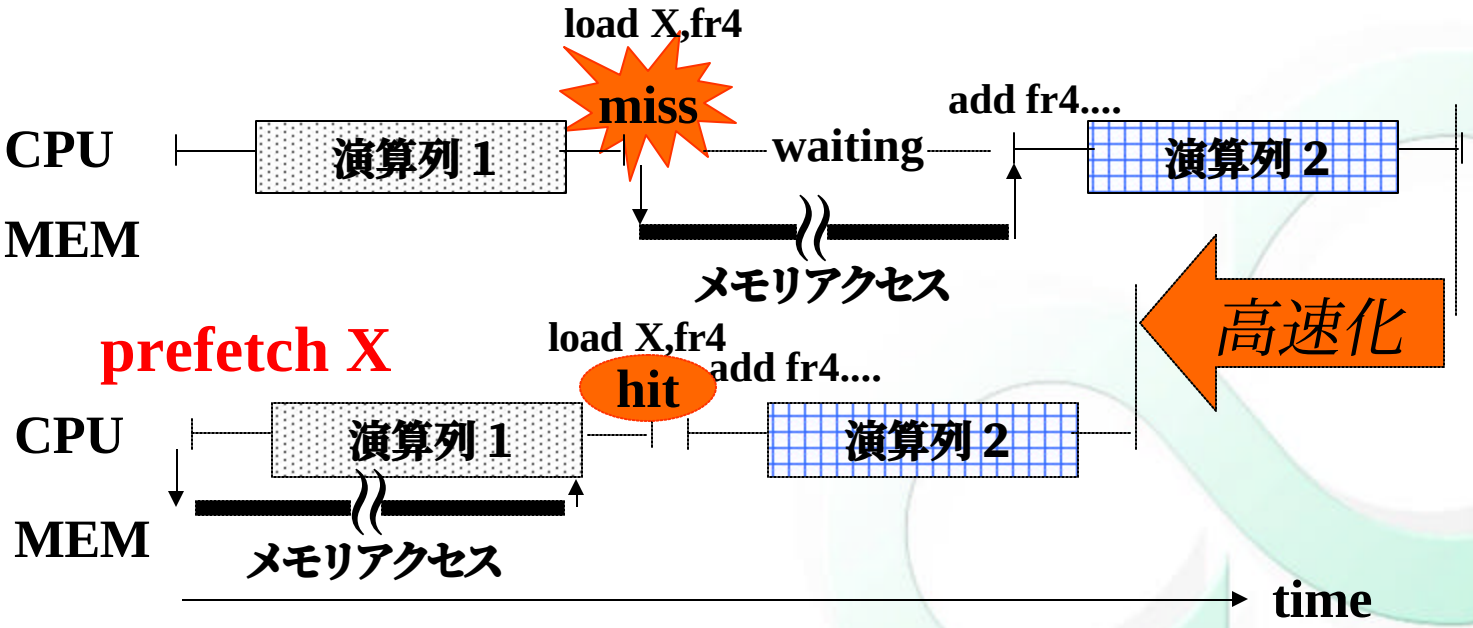
プリフェッチ機能

プリフェッチ :演算実行の充分前に、メモリからキャッシュにデータを先読み
データアクセス待ち時間を隠蔽し、高速化を実現

アドレス予測（コンパイラ技術）

突き放し制御（ハードウェア技術）

高速化実現



JAXA

Central Numerical Simulation System (CeNSS)

- **PRIMEPOWER HPC2500 system was installed to the Japan Aerospace Exploration Agency (JAXA) on Oct. 2002 as a main compute engine.**

- **Configuration of CeNSS**

PRIMEPOWER HPC2500
~ 14 Compute Cabinets ~
-Peak Performance: 9.3TFlops
-Memory (Total): 3.6TB

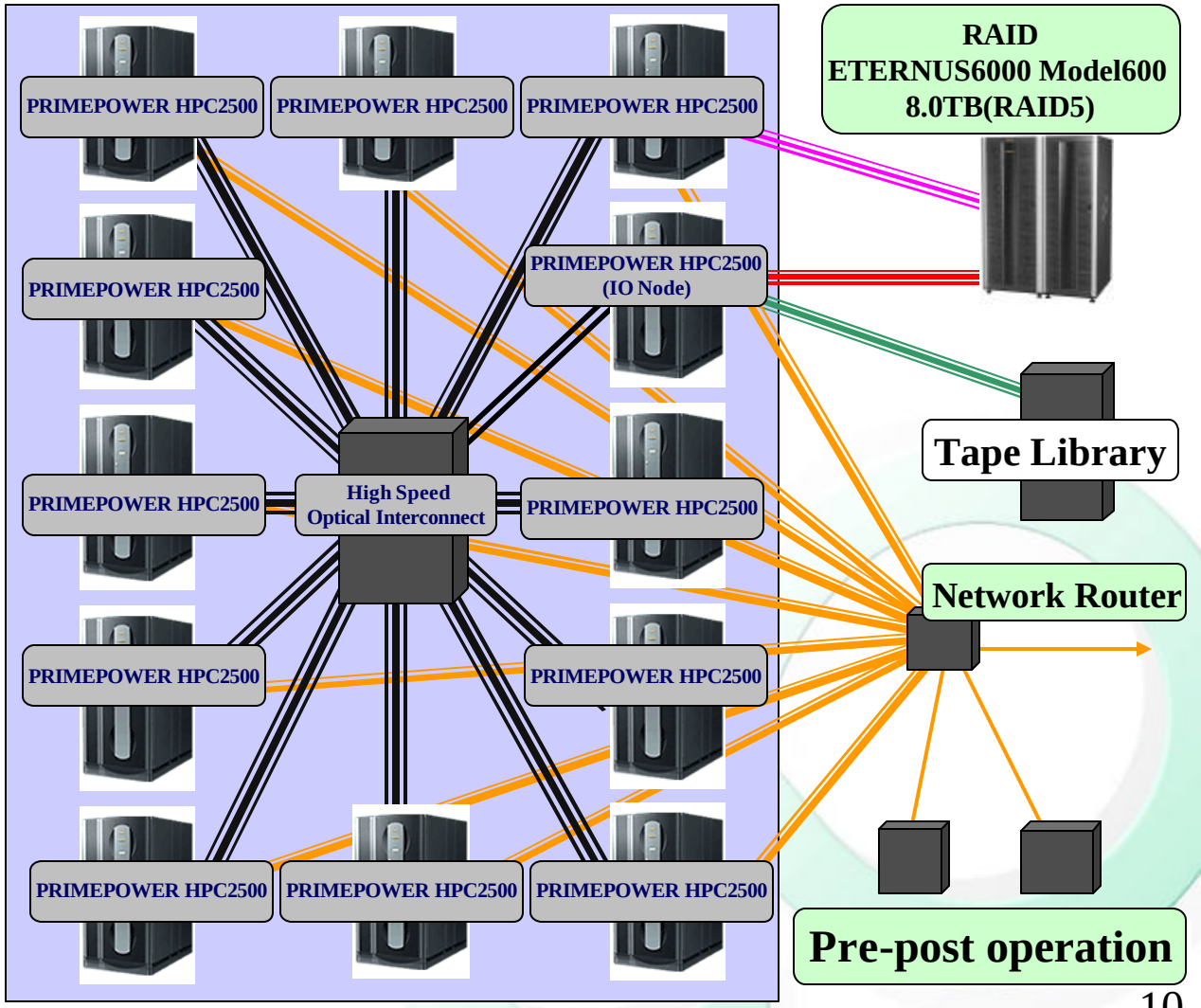
- **HPC2500(1Cabinet):**
  - **CPU: SPRAC64 V(1.3GHz) x 128CPU**
  - **Memory: 256GB**

- **Interconnect :**
  - **Crossbar Switch: 4GB/s(Bi-directional)**
  (Node to Node communication)

Kyoto University

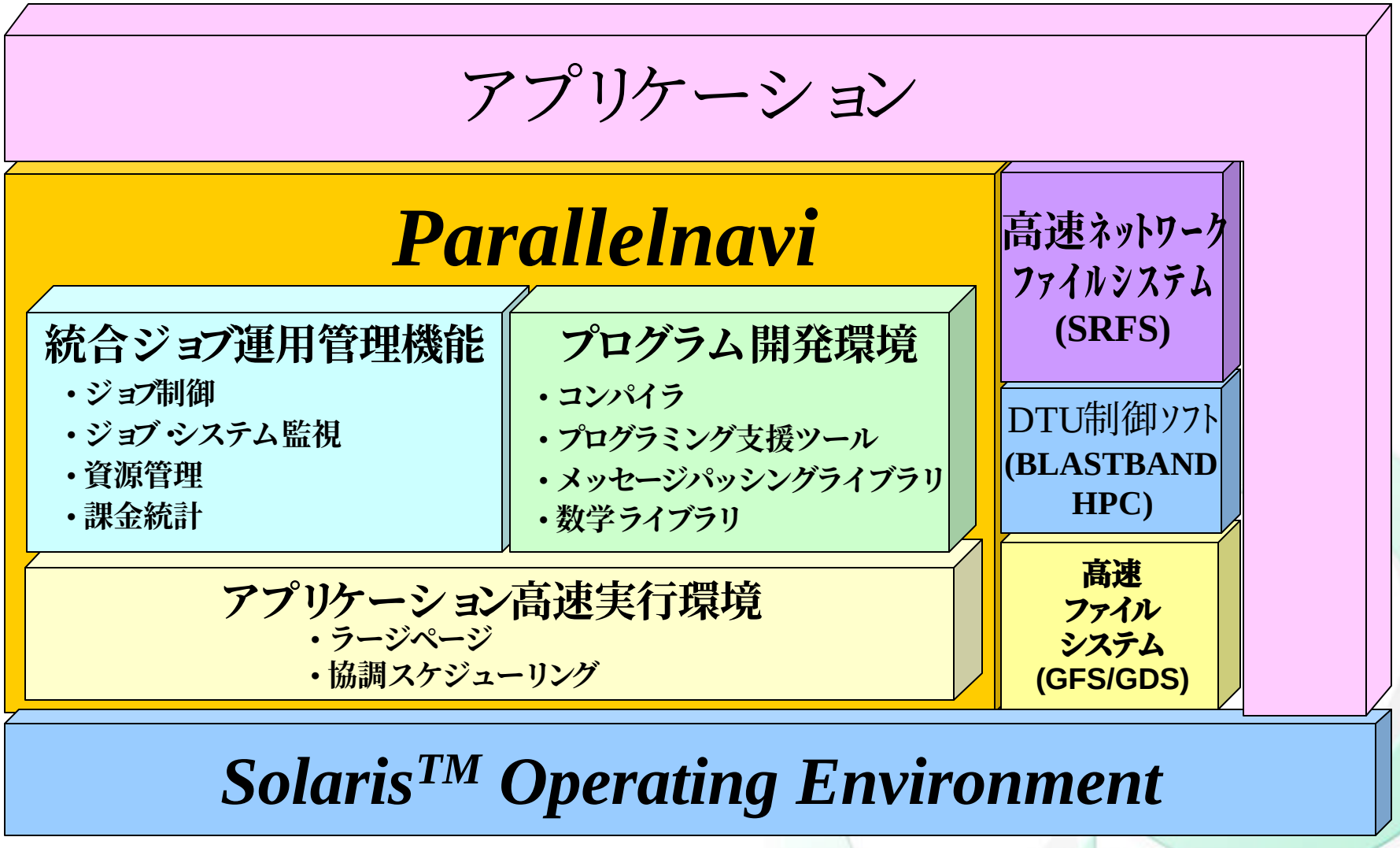
Supercomputer
PRIMEPOWER HPC2500
9.185TFLOPS, Memory:5.75TB

- **The largest class of supercomputer system in the world.**
- **The largest supercomputer system among Japanese university centers.**
- **Configuration**
  [PRIMEPOWER HPC2500]
  - **128CPU/Node 11Cabinets (Compute Nodes)**
  - **64CPU/Node 1Cabinet (I/O Node)**



HPC2500のソフトウェア概要

ソフトウェアの構成



プログラム開発環境

言語コンパイラ・MPL	プログラミング支援	数学ライブラリ
• Fortran • C • C++ 逐次処理 OpenMP 自動並列	• Parallelnavi Workbench*2	• SSL II • C-SSL II • BLAS • LAPACK
• MPI		• ScaLAPACK
• XPFortran *1		• SSL II/XPF *3

*1:eXtended Parallel Fortran(VPP Fortran仕様を包含). Parallelnaviとは別製品
*2:統合プログラミング環境
*3:SSL II/VPPと同じ機能を提供。XPFortranにバンドル。

プログラム開発環境の特長

高速性の実現

PRIMEPOWERのシステム性能を最大限に引き出し、プログラムの高速化およびスケーラビリティに優れた高い並列性能を実現します。

最新言語仕様/業界標準仕様のサポート

国際規格および業界標準言語仕様を一早くサポートし、可汎性と先進性を兼ね備えたアプリケーションの開発と実行を可能にします。

より使いやすく

一連のプログラム開発をスピードアップする、洗練されたGUIによるネットワーク透過的プログラミング環境を提供します。

VPP資産の継承

VPPシステムのお客様の資産を継承し、動作互換を保証します。

標準性の追求

最新の国際規格及び業界標準規格の採用

Fortran	ISO/IEC 1539-1:1997、JIS X3001-1:1998(Fortran95) (FORTRAN77/Fortran90を包含)
C	ISO/IEC 9899:1999(C99規格)、 X3.159-1989(ANSI C規格)、K&R仕様
C++	ISO/IEC 14882:1998 汎用クラスライブラリ(Rogue Wave Tools.h++ V8)
OpenMP	OpenMP Fortran Application Program Interface Version 2.0 OpenMP C and C++ Application Program Interface Version 2.0
MPI	MPI-2: Extension to the Message-Passing Interface (July 18,1997)

並列プログラミングモデル

自動並列化	ループを自動並列化	スレッド並列
OpenMP	共有メモリ向け業界標準の並列拡張言語 指示行挿入による高度で柔軟な並列プログラミング	
XPFortran	分散メモリ向けのデータ並列型拡張言語 (VPP Fortran仕様を包含)	プロセス並列
MPI	プロセス間通信の業界標準言語仕様 (クラスタ型並列・分散処理)	

スレッド並列:同一のプロセス空間で複数スレッドが並列に動作

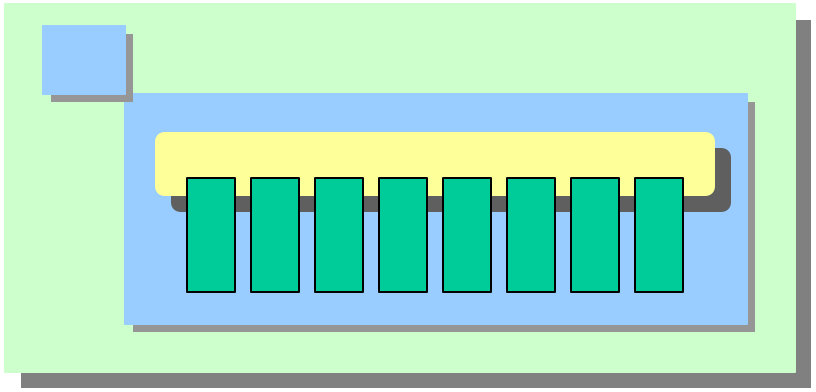
プロセス並列:複数のプロセスが通信を行ないながら並列に動作

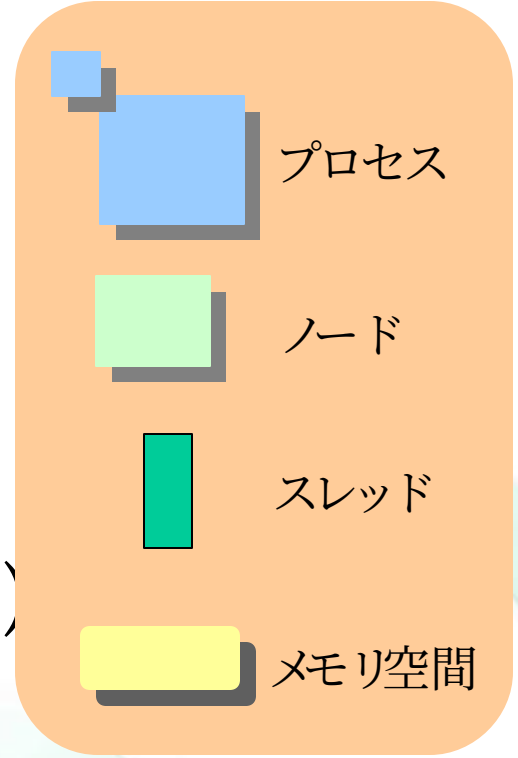
並列プログラミングモデル(組み合わせ)

ノード内（SMP）	ノード間
自動並列・OpenMP	XPFortran
自動並列・OpenMP	MPI
XPFortran	
MPI	

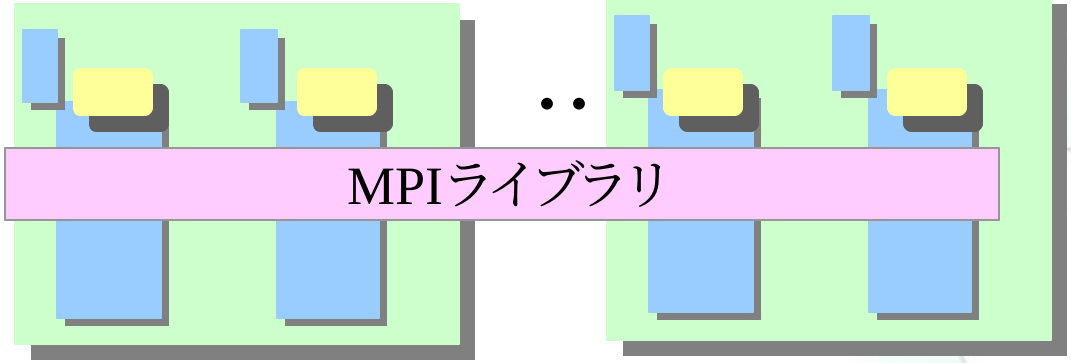
並列処理イメージ

自動並列／OpenMP（ノード内並列処理）





MPI（ノード内・ノード間並列処理）



プロセスとスレッド

```
      program main
      dimension dif(1000),u(1000)
        :
      c = 2.0
!$OMP PARALLEL DO
      do i = 2, 999
        dif(i) = u(i+1) - c*u(i) + u(i-1)
      end do
        :
      end program main
```

OpenMP

```
      program main
!XOCL PROCESSOR P(4)
      dimension u(1000),dif(1000)
!XOCL INDEX PARTITION Q=(P,INDEX=1:1000)
!XOCL GLOBAL u(/Q(overlap=(1,1))),dif(/Q)
!
!XOCL PARALLEL REGION
        :
      c = 2.0
!XOCL OVERLAPFIX(u)(id)
!XOCL MOVE WAIT(id)
!XOCL SPREAD DO REGIDENT(u,dif) /Q
      do i = 2, 999
        dif(i) = u(i+1) - c*u(i) + u(i-1)
      end do
!XOCL END SPREAD
        :
!XOCL END PARALLEL
      end program
```

XPFortran

```
program main
  include "mpif.h"
  real(kind=4),dimension(:),allocatable :: dif,u
  integer STATUS(MPI_STATUS_SIZE)
    :
  call MPI_INIT(ierr)
  call MPI_COMM_SIZE(MPI_COMM_WORLD,   npe,ierr)
  call
MPI_COMM_RANK(MPI_COMM_WORLD,myrank,ierr)
  im   =  1000
  ilen =  (im + npe - 1 )/npe
  ist  =  myrank*250 + 1
  iend =  ist + ilen - 1
  allocate( u(ist-1:iend+1), dif(ist:iend) )
  nright = myrank + 1
  nleft  = myrank - 1
  if(myrank==    0) then
    nleft  = MPI_PROC_NULL
  else if(myrank==npe-1) then
    nright = MPI_PROC_NULL
  end if
  call MPI_SENDRECV( u(iend  ),1,MPI_REAL,nright,0, &
                     u(ist-1 ),1,MPI_REAL, nleft,0, &
                     MPI_COMM_WORLD,STATUS,IERR )
  call MPI_SENDRECV( u(ist   ),1,MPI_REAL, nleft,1, &
                     u(iend+1),1,MPI_REAL,nright,1, &
                     MPI_COMM_WORLD,STATUS,IERR )
  c       =  2.0
  ist_do  =  max(  2,ist )
  iend_do =  min(999,iend)
  do i = ist_do, iend_do
    dif(i) = u(i+1) - c*u(i) + u(i-1)
  end do
    :
  call MPI_FINALIZE(ierr)
  end program main
```

MPI

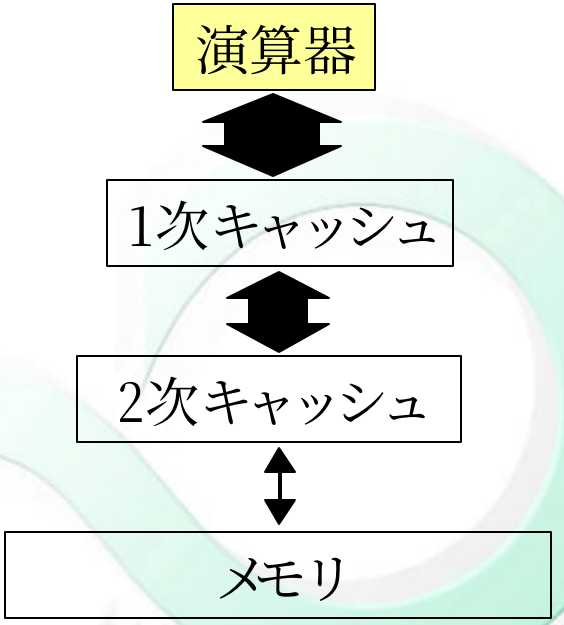
キャッシュメモリ最適化

- 1次キャッシュ／2次キャッシュへのデータプリフェッチ機能
- インダイレクトアクセスのプリフェッチ
- ディスタンスアクセスはハードウェアプリフェッチ* と連携
  *：参照アドレスを予測し動的にプリフェッチを行うハードウェア機能
- 翻訳指示行（OCL）で，きめ細かいプリフェッチ指定も可能

例） 1次・2次キャッシュへのデータプリフェッチ

```
DO I=1,N                    DO I=1,N
  SUM=SUM+A(I)        ⇨       SUM = SUM + A(I)
END DO                        Prefetch2 A(I+17) :2次キャッシュへ
                              END DO
```

Prefetch1 A(I+1):1次キャッシュへ



SSLII 数値計算ライブラリ)

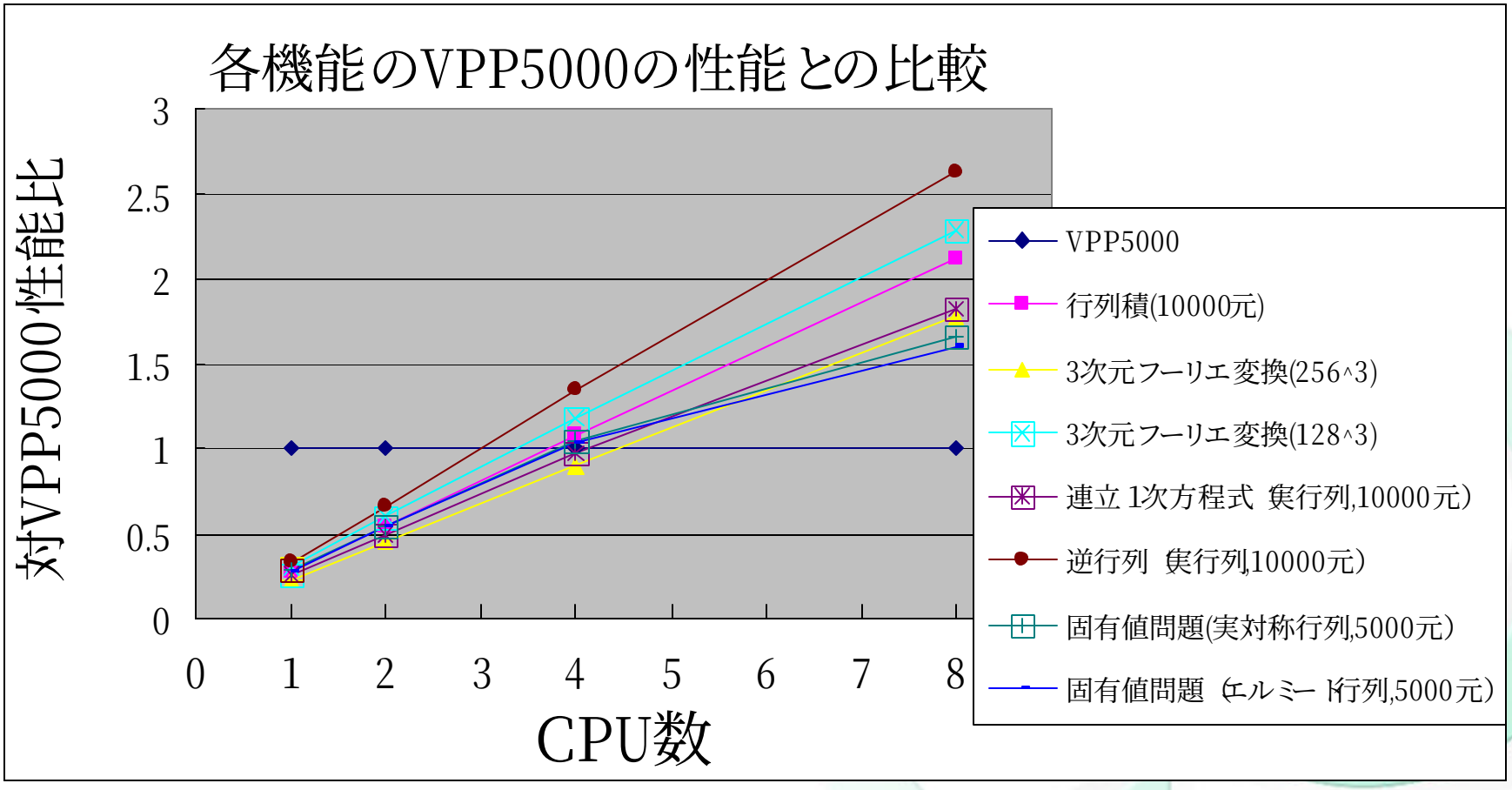
- 10分野の数値計算アルゴリズムを精選（270機能）
  大学・研究機関の最新の解法を高度にチューニングして提供
  SSLII :Fortranライブラリ, C-SSLII :SSLIIのCインターフェース
- スレッドセーフ対応(複数スレッドから同時呼び出し可能)
- スレッド並列ライブラリ(OpenMP Fortranで記述）を提供

機能概要一覧

線形計算	:連立一次方程式、逆行列、最小二乗解、特異値分解
固有値問題	:固有値・固有ベクトル
非線形方程式	:代数方程式、超越方程式、連立非線形方程式
極値問題	:関数の極小化、線形計画問題、非線形計画問題
補間・近似	:補間式／補間値、近似式、平滑化式／平滑値、級数展開
変換	:フーリエ変換、ラプラス変換、ウェーブレット変換
数値微分・積分	:離散点／関数入力、有限区間／無限領域、一次元／二次元
微分方程式	:連立一階常微分方程式、連立一階スティフ常微分方程式
特殊関数	:ベッセル関数、楕円積分、指数積分、正弦・余弦積分
擬似乱数	:乱数生成（一様／正規／指数／ポアソン／二項)、乱数検定

数学ライブラリ（ベクトルとの比較）

VPP5000／1PEと3～4CPU程度で同等性能



OpenMP

共有メモリ向け業界標準の並列言語

- 最新規格 2.0準拠で、配列代入文やFORALL文などの並列処理をサポート

豊富な機能

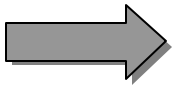
- DOループ並列、配列並列、タスク並列、マルチスレッド範囲の指定、
- 同期/排他制御/逐次化、リダクション演算、データスコープ指定

柔軟な運用性

- 段階的な並列化/高速化、プロセッサ数の動的指定
- 大規模システムでもスレッド並列が適用可能
  （並列化を陽に記述するので、 オーバーヘッドに対する相対的粒度を大きくできる）

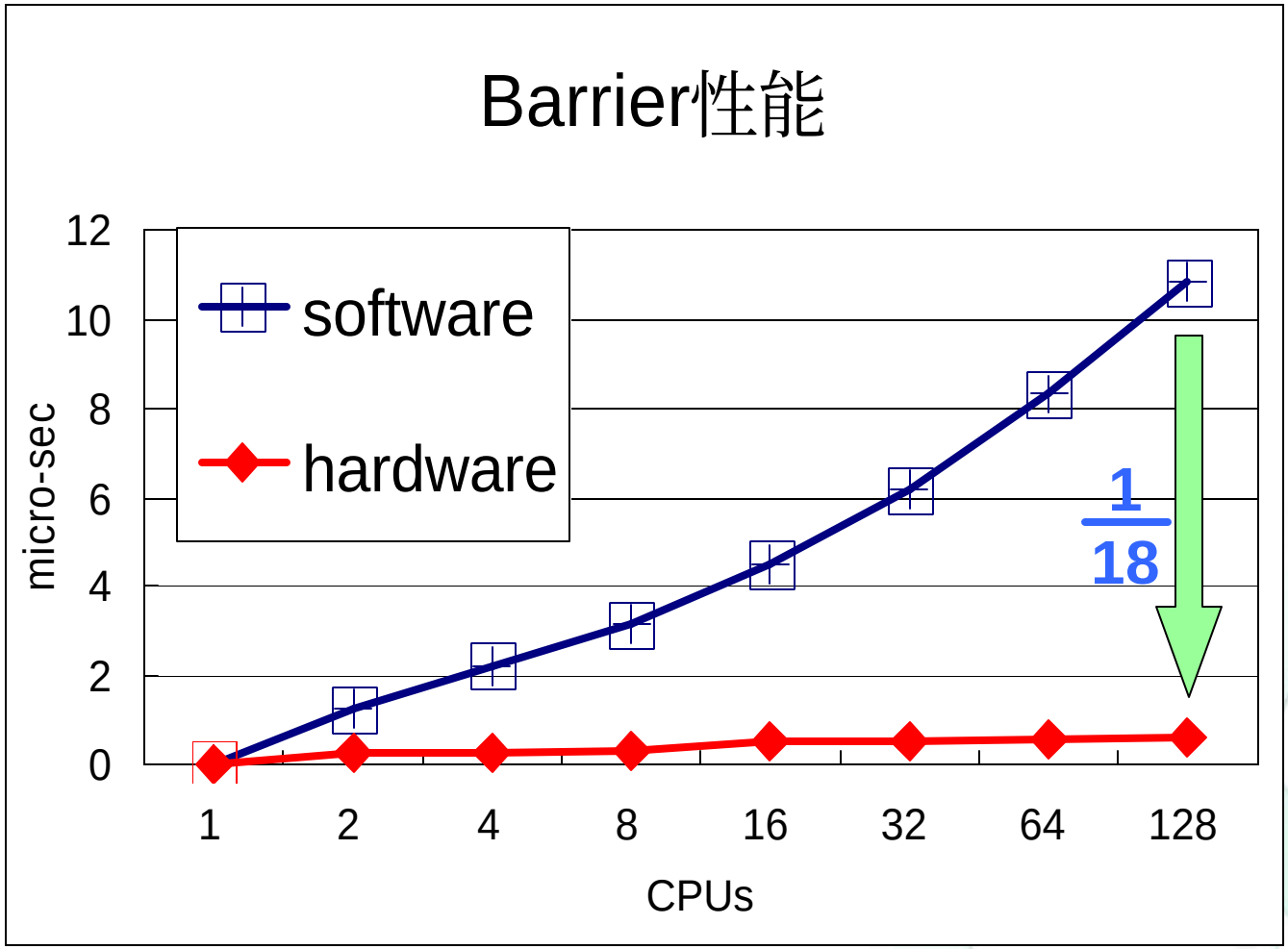
例:

```
 :
DO I=1,1000
  B(I)=(A(I)+A(I+1)/2.0)
END DO
 :
```



```
                :
!$OMP PARALLEL DO
      DO I=1,1000
        B(I)=(A(I)+A(I+1))/2.0
      END DO
!$OMP END PARALLEL DO
                :
```

ノード内 Barrier 性能(実測)



以下で利用。

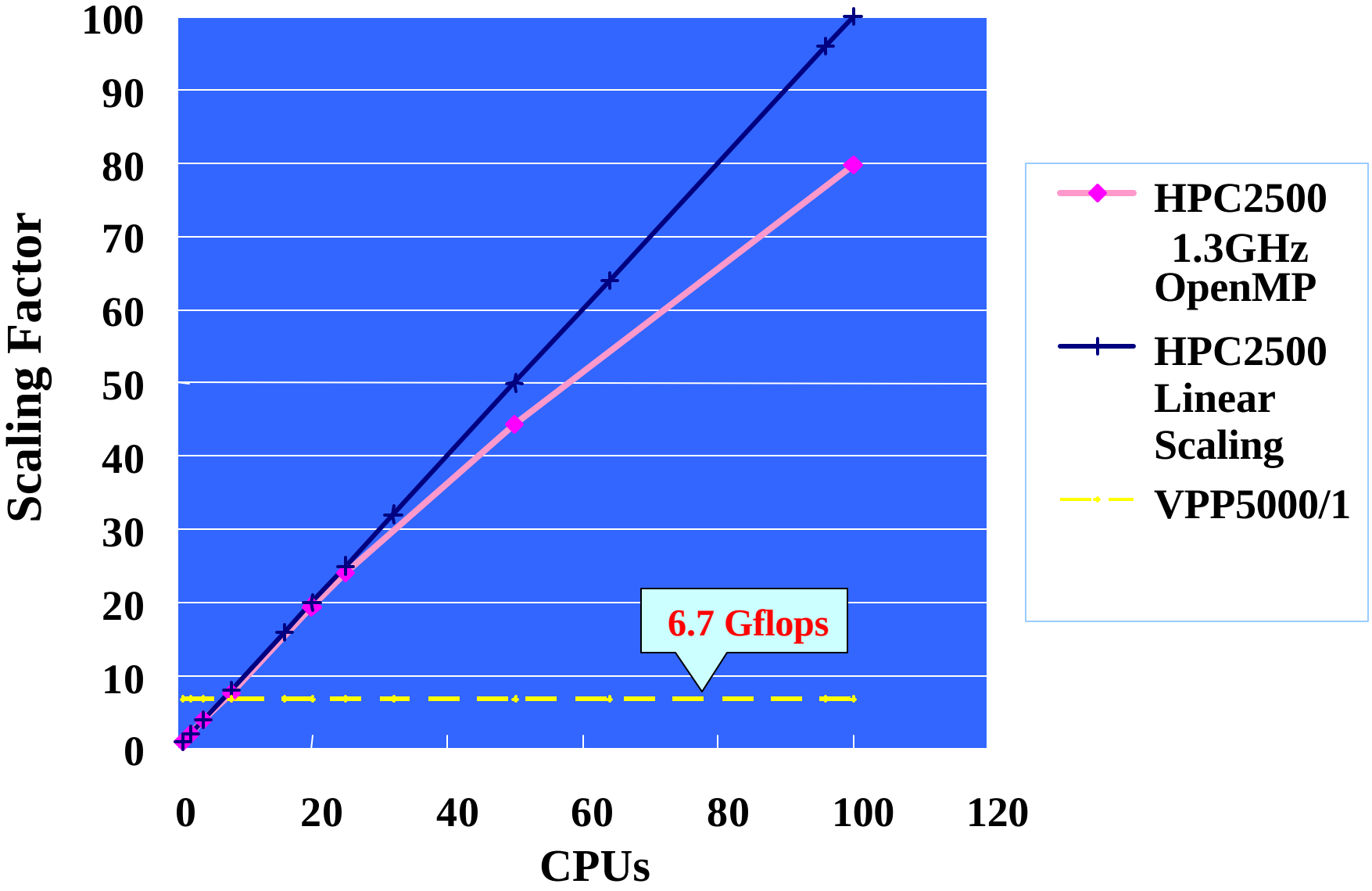
スレッド並列

- 自動並列
- OpenMP

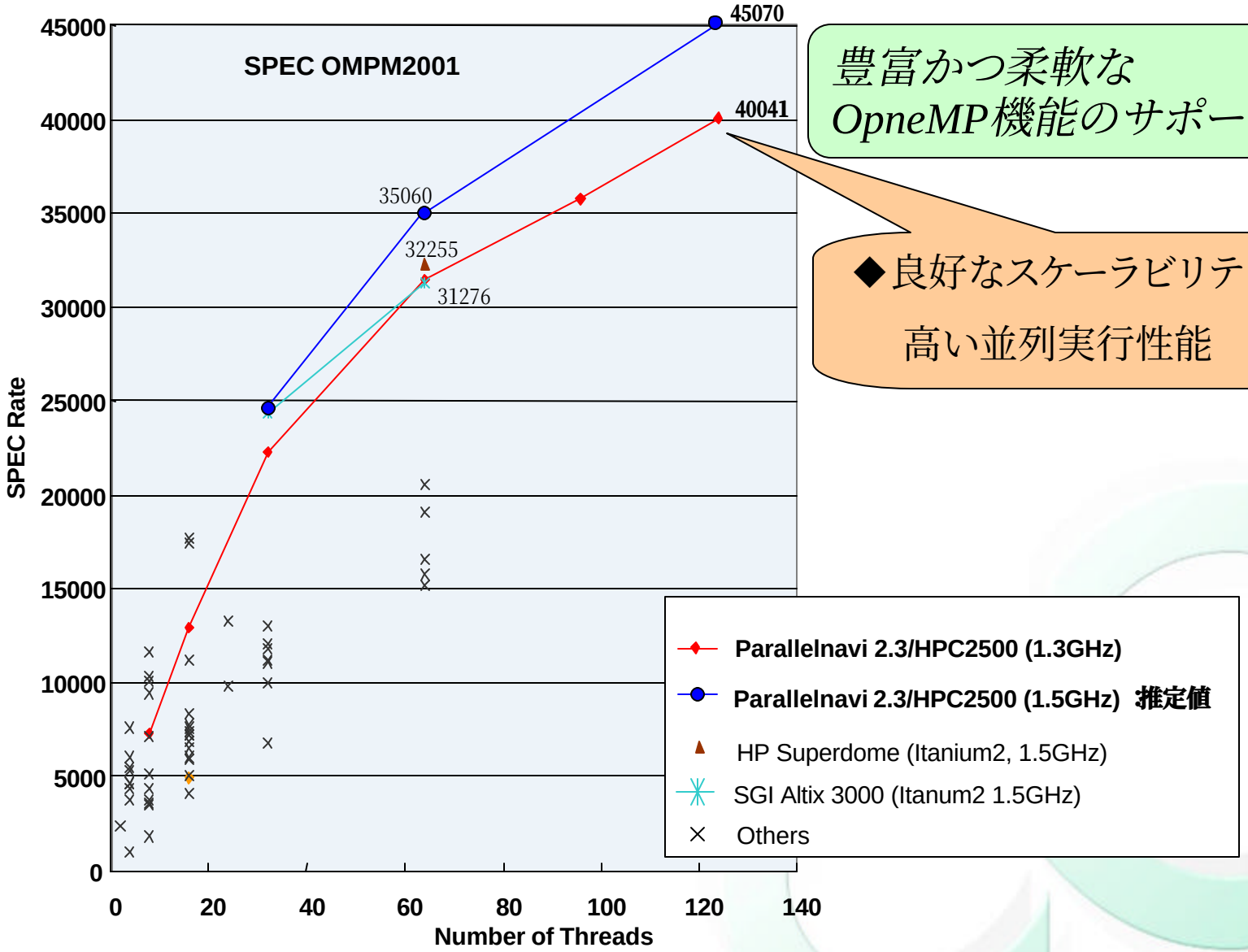
ノード内に閉じたプロセス並列

- MPI
- XPFortran

NAS Parallel BT Class B



OpenMP性能（SPEC OMPM2001）



豊富かつ柔軟な
OpneMP機能のサポート

◆良好なスケーラビリティと
高い並列実行性能

MPI

MPI2規格に準拠

- MPI1規格：1対1通信、集団通信、派生データ型、コミュニケータ、プロセス・トポロジー
- MPI2規格：動的プロセス生成、片側通信、並列入出力

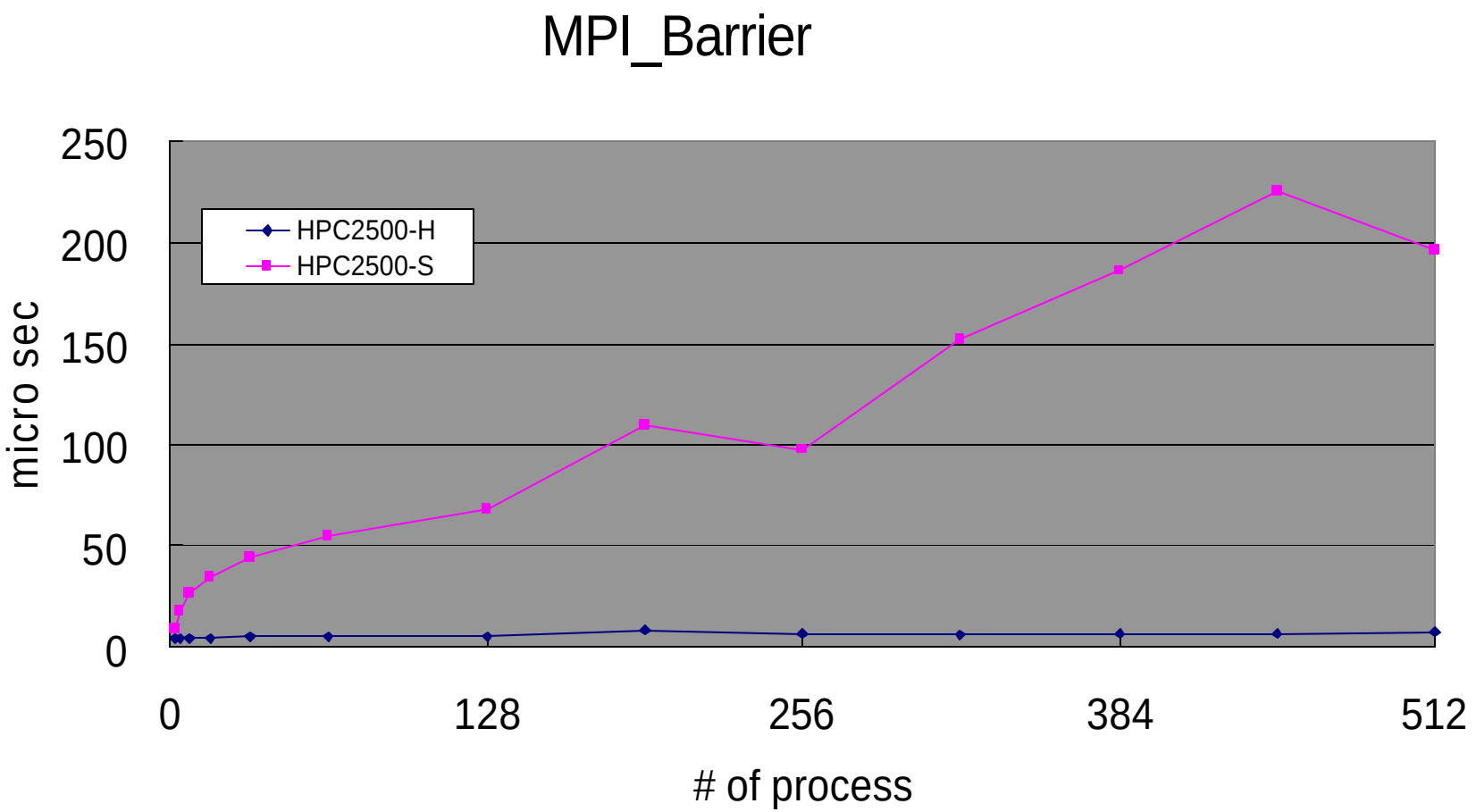
高速プロセス間通信

- 共有メモリ通信、高速光インターコネクトによるノード間通信
- Send On Request方式(受信要求時のメッセージ転送）による高速データ通信

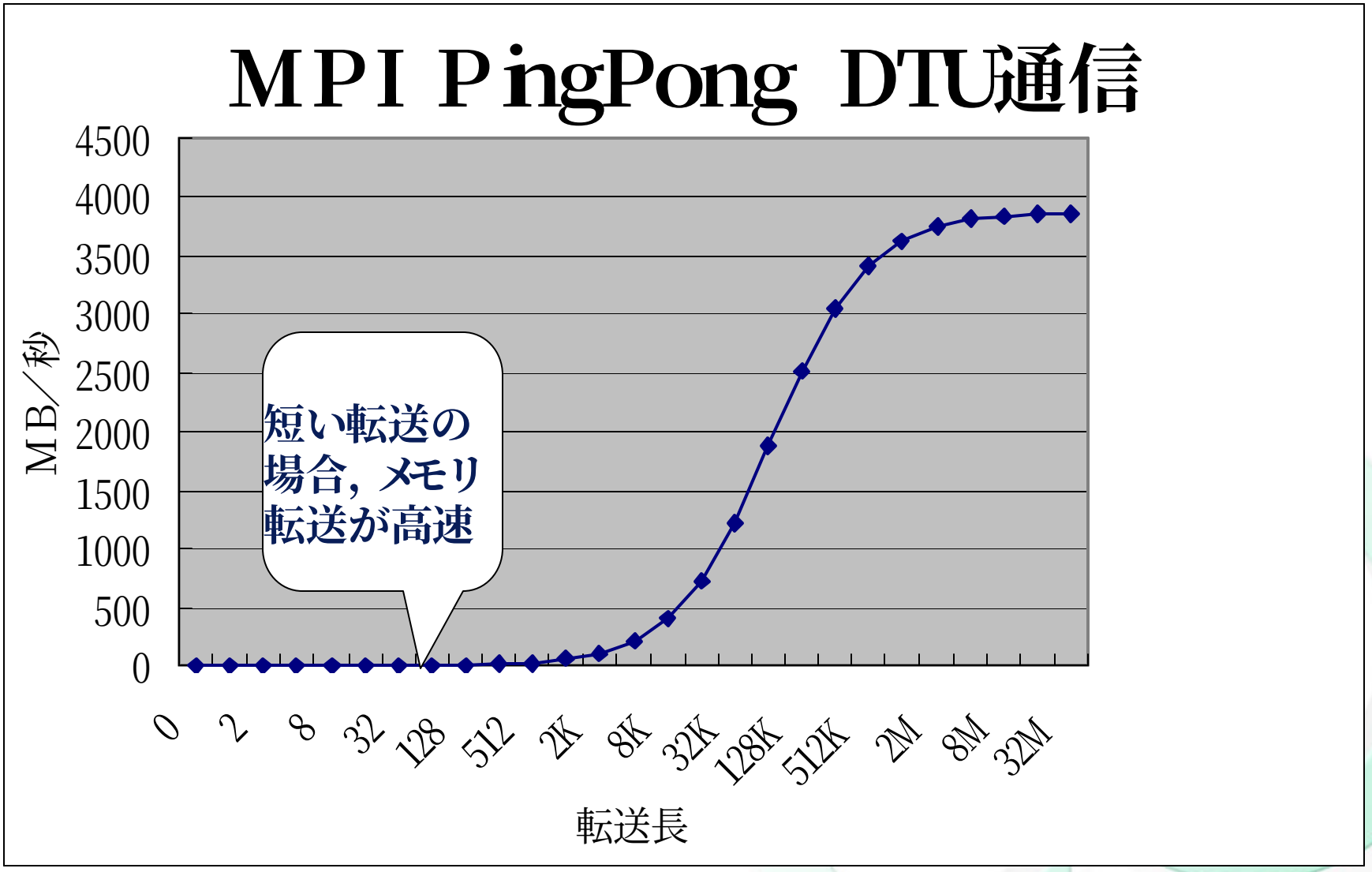
柔軟な操作性

- 自動並列/OpenMPとの共用
- プログラム支援環境によるMPIプログラミング開発支援

ノード間 Barrier 性能



MPIの転送性能



スカラ並列チューニングのポイント ⇒チューニングポイント検出ツール重要

スカラーチューニング

◆キャッシュミス率を下げる ◆TLBミス率を下げる ◆演算器の使用効率向上

並列チューニング

スレッド並列チューニング

◆並列化率を上げる

◆並列化粒度を上げる

◆ロードバランスを均等化

◆スレッド間のキャッシュ競合回避

プロセス並列チューニング

◆並列化率を上げる

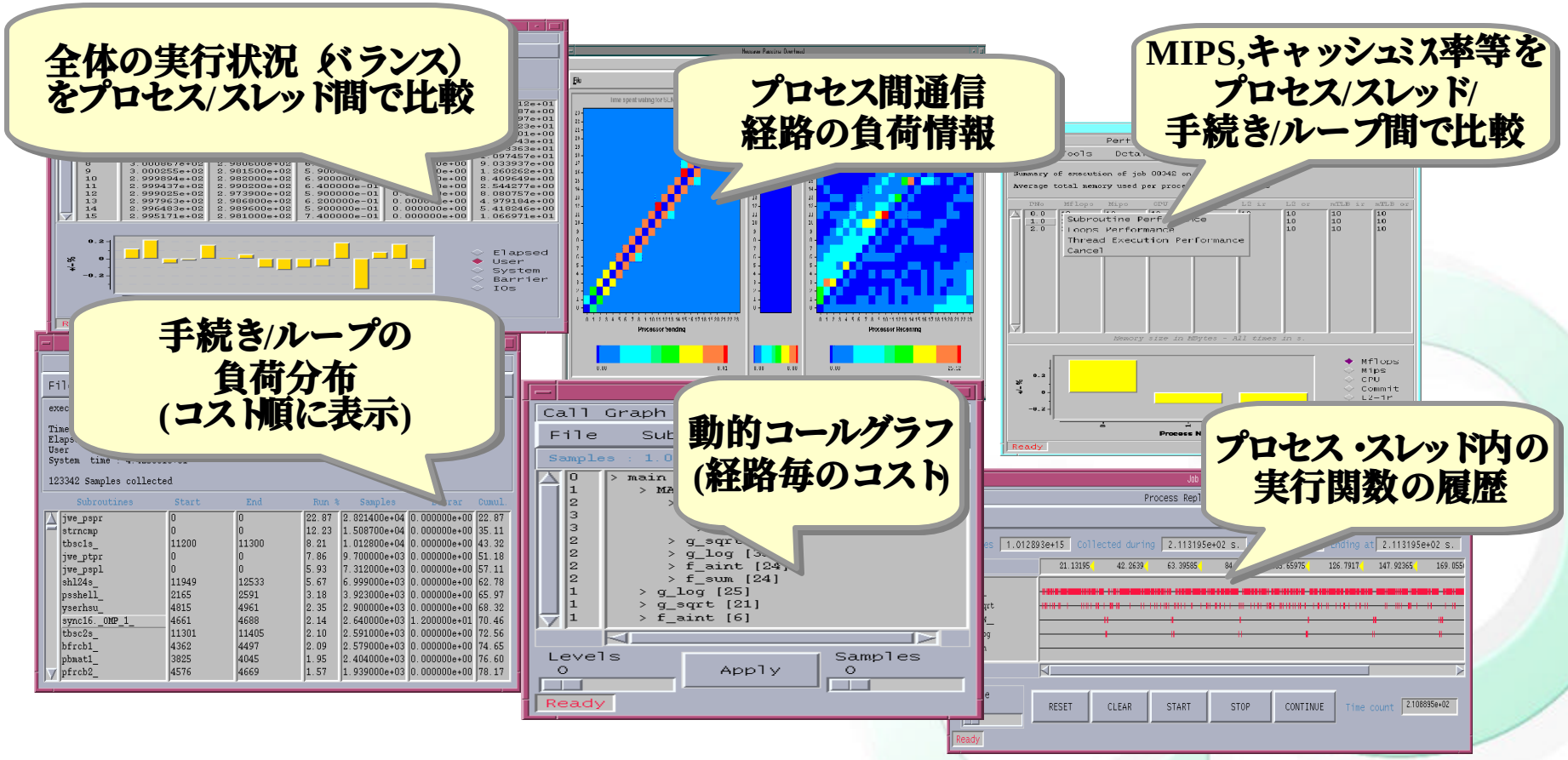
◆並列化粒度を上げる

◆ロードバランスを均等化

◆プロセス間の通信コスト削減

チューニングツール 動的プロファイラ

性能改善候補(高負荷個所）を簡単に見つけられます。
OpenMP/自動並列、MPI(混合)プログラムに対応
(タイマ/サンプリングにより実行情報を収集し、事後解析)
プロセス/スレッドの並列バランス、負荷の高い手続き/ループの分析
ハードウェアアクセス性能(キャッシュ、TLBミス)、DTU性能の分析



プロファイラの活用例（CPU負荷）

ソースプログラム

```
      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

プロファイラ情報（ロードバランス）

Performance Analysis

Elapsed	User	System	
1.563679e+01	4.050000e+00	3.630000e+00	Process 0

各スレッドのバランス悪い！

```
                                - 0 +
+-----------------------------+-----------------------------+
|                             |********************         | + 77%  1.420000e+02  Thread  0
|                             |**********                   | + 38%  1.110000e+02  Thread  1
|                             |                             | -  0%  8.000000e+01  Thread  2
|                **********   |                             | - 39%  4.900000e+01  Thread  3
|     ********************    |                             | - 76%  1.900000e+01  Thread  4
+-----------------------------+-----------------------------+
```

Balance against average time per Thread

プロファイラの活用例（CPU負荷改善）

改善後ソースプログラム

```
      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
```

プロファイラ情報（ロードインバランス）

```
Performance Analysis

      Elapsed            User          System
-------------------------------------------------------------------------
9.884062e+00   4.180000e+00   4.470000e+00      Process   0
```

PARALLEL BRANCEを均一化したことにより、1．6倍向上

```
- 0 +
+------------------------------+------------------------------+
|                              |                              | - 1%  8.200000e+01  Thread   0
|                              |                              |   0%  8.300000e+01  Thread   1
|                              |                              | + 1%  8.400000e+01  Thread   2
|                              |                              |   0%  8.300000e+01  Thread   3
|                              |                              |   0%  8.300000e+01  Thread   4
+------------------------------+------------------------------+
Balance against average time per Thread
```

OMPでｽｹｼﾞｭｰﾘﾝｸﾞ

各ｽﾚｯﾄﾞのバランス改善！！

最後に

◇スカラサーバの高性能化

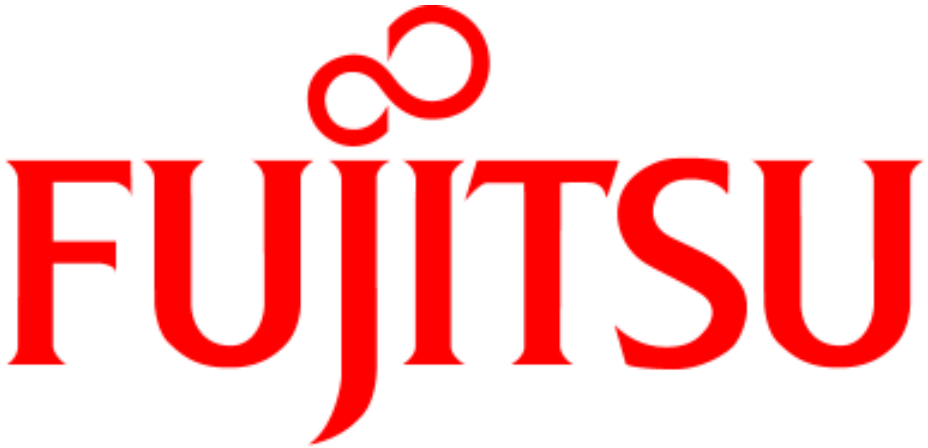
・ノードあたりのピーク性能の飛躍的向上

◇顧客の価格性能要求に応じた最適なシステムの提供

・大規模SMP、超高速インターコネクト技術による
ハイスケーラビリティの実現

◇VPPからのスムーズな移行が可能

・自動並列化, XPFortran



THE POSSIBILITIES ARE INFINITE