3.5世代PCクラスタを中核とする理研RICC：その狙いと現状、今後

理化学研究所
情報基盤センター

内容

- PCクラスターの歴史
- Top500の新たな潮流
- GPGPUの特徴
- 第二世代PCクラスタ：RSCCの狙いとその結果
- 第３．５世代PCクラスタ：RICCの狙いと現状、今後

PCクラスターの歴史と
最近の潮流

PCクラスタヒストリー

- 第一世代：Beowulf型　個人
- 第二世代：SCoreIII、産総研、RSCC(MD-GRAPE)
  - 高性能、共用、センター運用
  - Grid Computing
  - RSCC:MD-GRAPEによる加速
  - ２GBメモリーの壁
- 第三世代：Fat Node、Multi-Core: Tsubame(2006年3月稼働、ClearSpeed）、T2K(2008年）
- 第3.5世代：　＋　汎用加速ボード
  - Tsubame1.2（2008年12月GPGPU）、理研RICC（2009年8月MD-GRAPE3+GPGPU）
  - GPGPUアクセラレータの一般化

第一世代：Beowulf型　個人

2000年　デスクトップPC
CPU: Pentium III 450MHz × 1CPU × 8ノード
Interconnect : Fast Ethernet x 1 (100Mbps)

2001年　デスクトップPC
CPU: Pentium4 1.5GHz × 1CPU × 8ノード
Interconnect : Fast Ethernet x 1 (100Mbps)

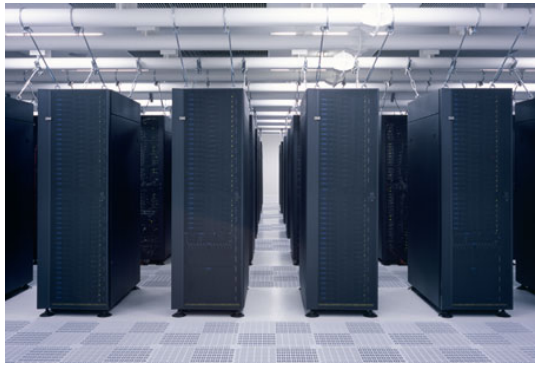
2001年 ラックマウント4U
CPU: Pentium 4 1.7GHz × 1CPU × 64ノード
Interconnect : Myrinet 2000
Peak :217.6 GFLOPS

第二世代：SCoreIII、産総研、RSCC

2001年 Score III
CPU: Pentium III 933MHz × 2CPU × 512ノード
Interconnect : Myrinet 2000 (2.0Gbps)
Peak：955.4 GFLOPS

2004年 RSCC
CPU: Xeon 3.06GHz × 2CPU × 1024ノード
Interconnect : InfiniBand
Peak：12.4 TFLOPS

2004年 AISTスーパークラスタ
CPU: Opteron 2.0GHz × 2CPU, Itanium2 1.3GHz × 4CPU、Xeon 3.06GHz × 2CPU
Interconnect : Myrinet、GbE
Peak：14.6TFLOPS



第三世代：Tsubame, T2K

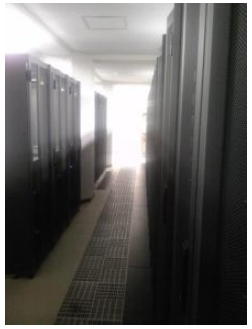
2006年 TSUBAME
CPU: **Dual Core Opteron (2.4GHz) × 8CPU /ノード、ClearSpeed CSX600**
Interconnect: Infiniband
Peak: 47.38TeraFlops

+GPGPU

3.5世代

2008年 Roadrunner (LANL)
CPU: **PowerXCell 8i 3.2 Ghz** /
Dual Core Opteron 1.8 GHz
Interconnect: Infiniband
Peak: 1.4567PFLOPS

2008年 T2K
CPU: **Quad core Opteron × 4CPU/ノード**
Interconnect: Myrinet, Infiniband
T2K東大: 140TFLOPS（952ノード）
T2Kつくば: 95TFLOPS（648ノード）
T2K京都: 61.2TFLOPS（416ノード）

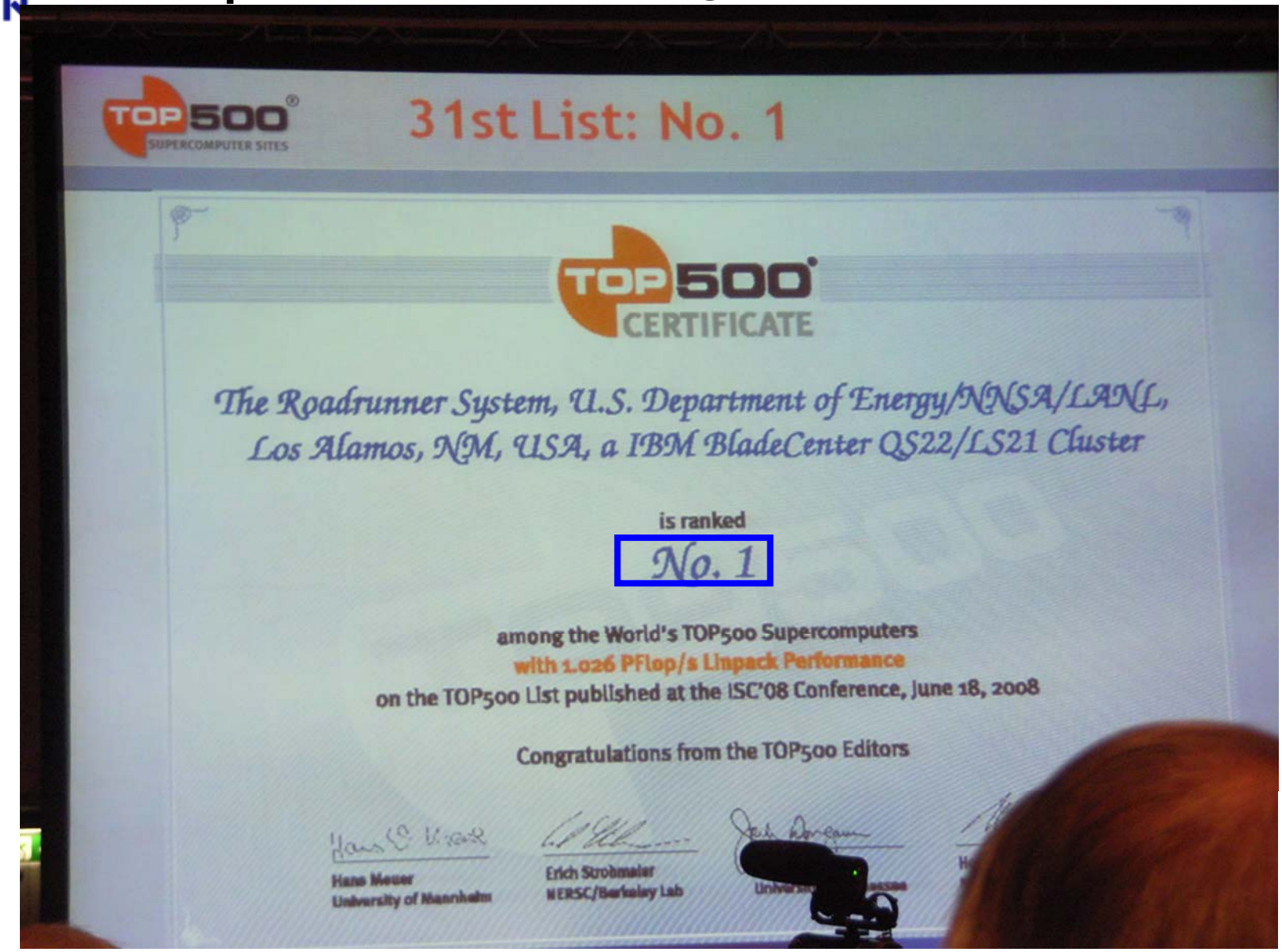


日本でのアクセラレータ

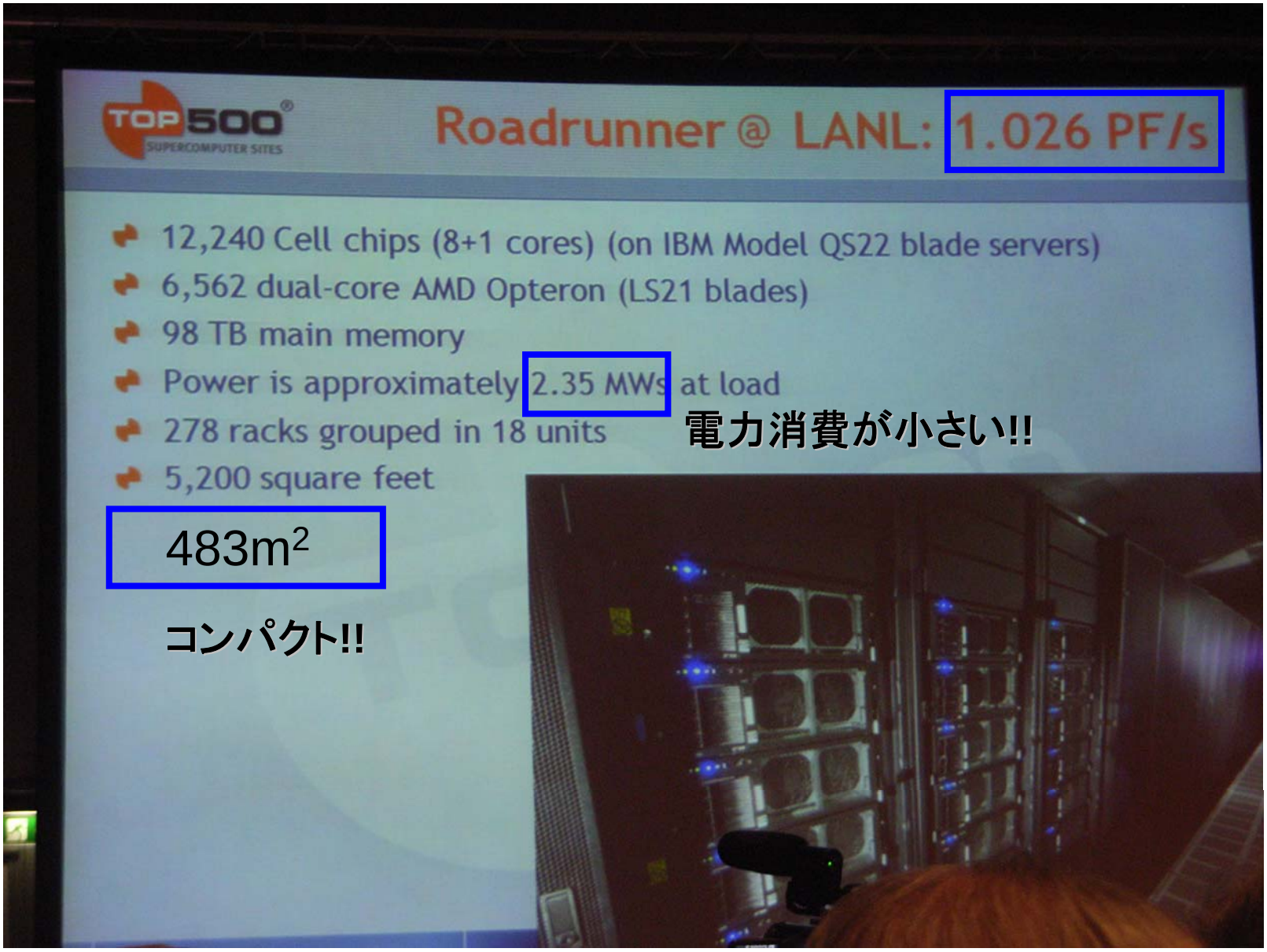
アクセラレータの性能はホスト計算機の10-100倍（コスト性能比・電力性能比も）

- 東大：GRAPEシリーズ
  - 現在はGRAPE-DR
  - 初代から数えて８代目
- 理研：
  - MD-GRAPE2、WINE、MD-GRAPE3
- 研究的なボード：FPGA
- 市販品
  - ClearSpeed
  - Tesla

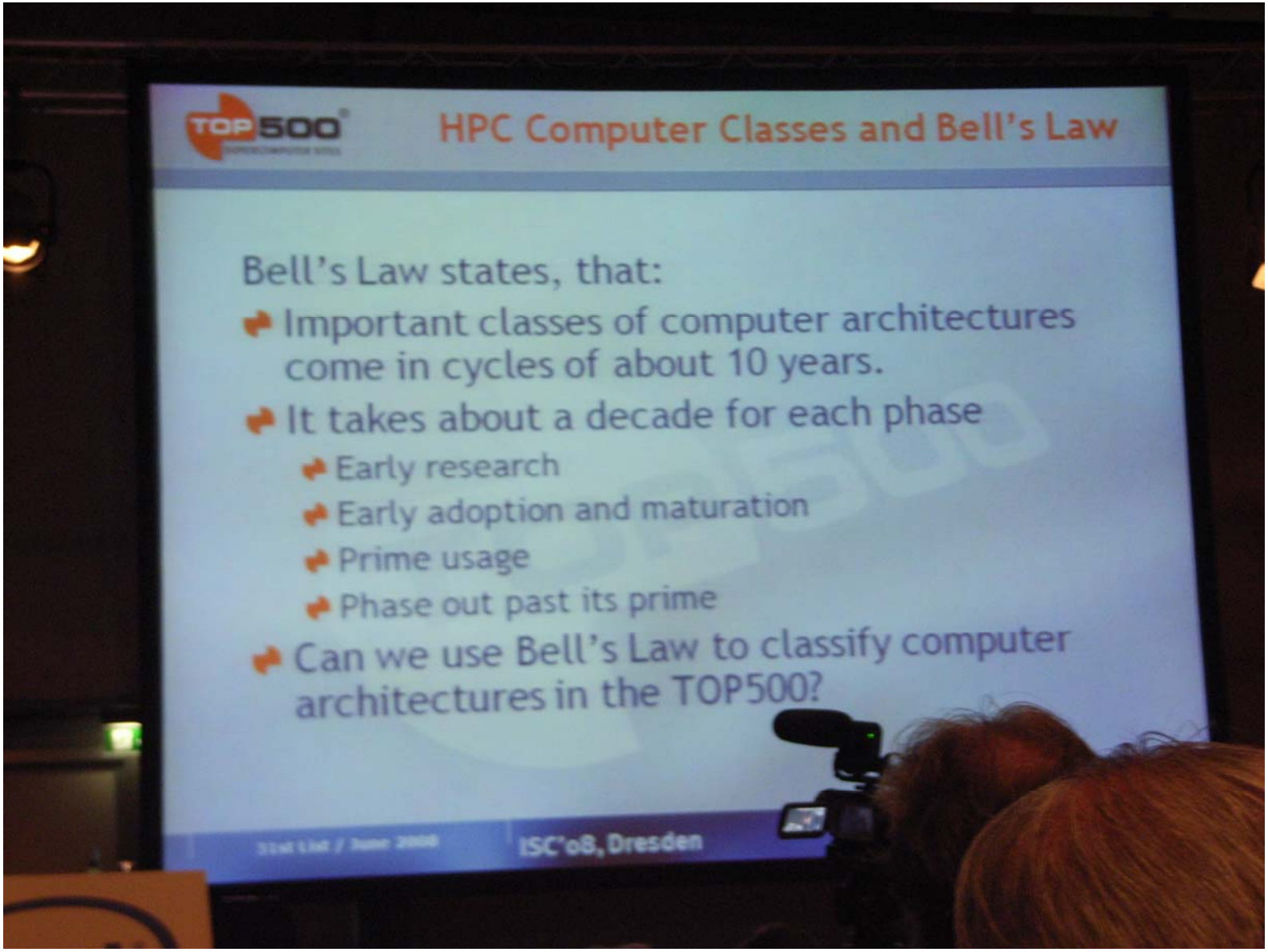
世界では？
08年のISC：No.1はRoadrunner



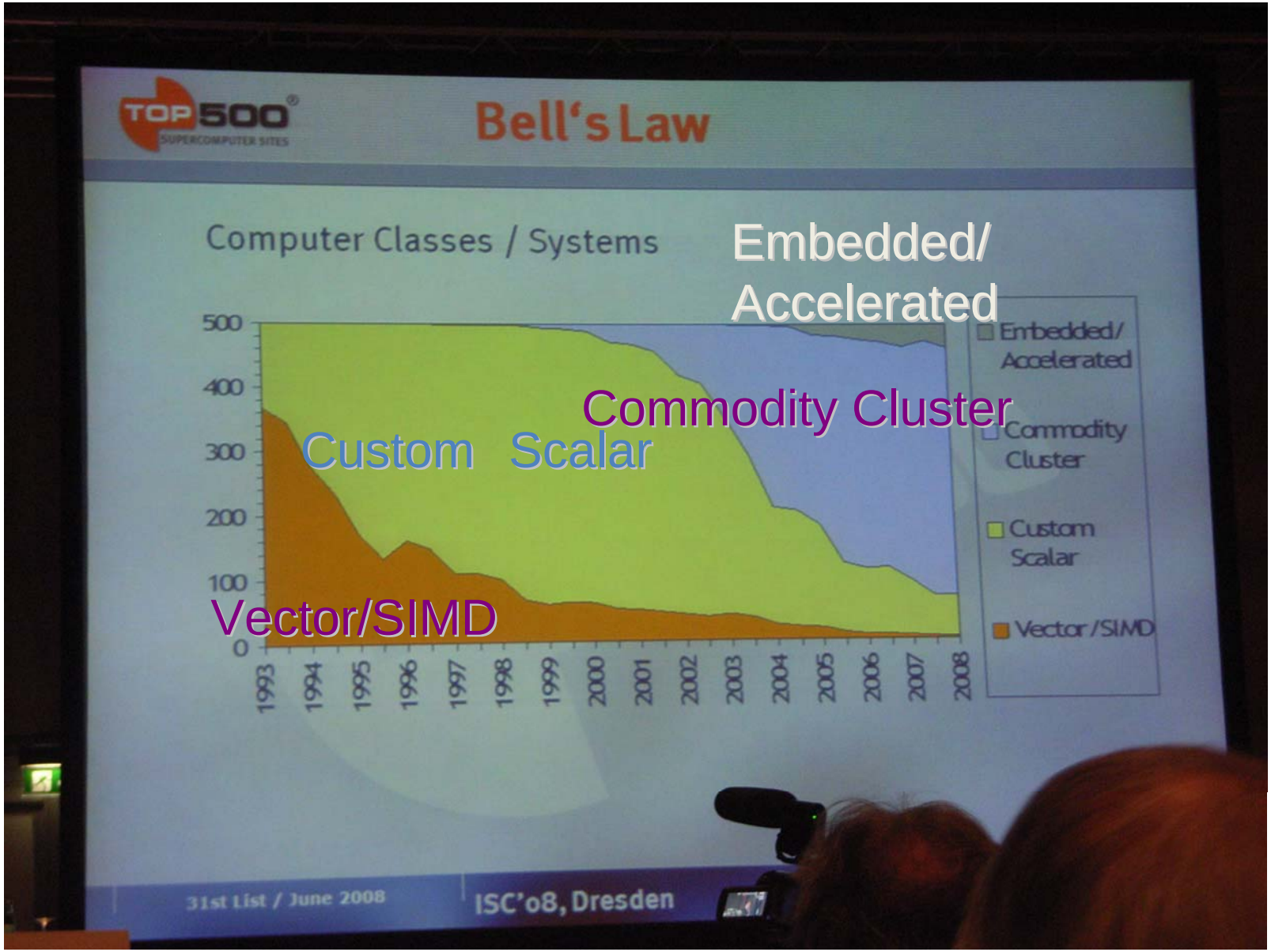
Roadrunner


TOP500 SUPERCOMPUTER SITES
Roadrunner @ LANL: 1.026 PF/s
12,240 Cell chips (8+1 cores) (on IBM Model QS22 blade servers)
6,562 dual-core AMD Opteron (LS21 blades)
98 TB main memory
Power is approximately 2.35 MWs at load
278 racks grouped in 18 units
5,200 square feet
483m2
電力消費が小さい!!
コンパクト!!

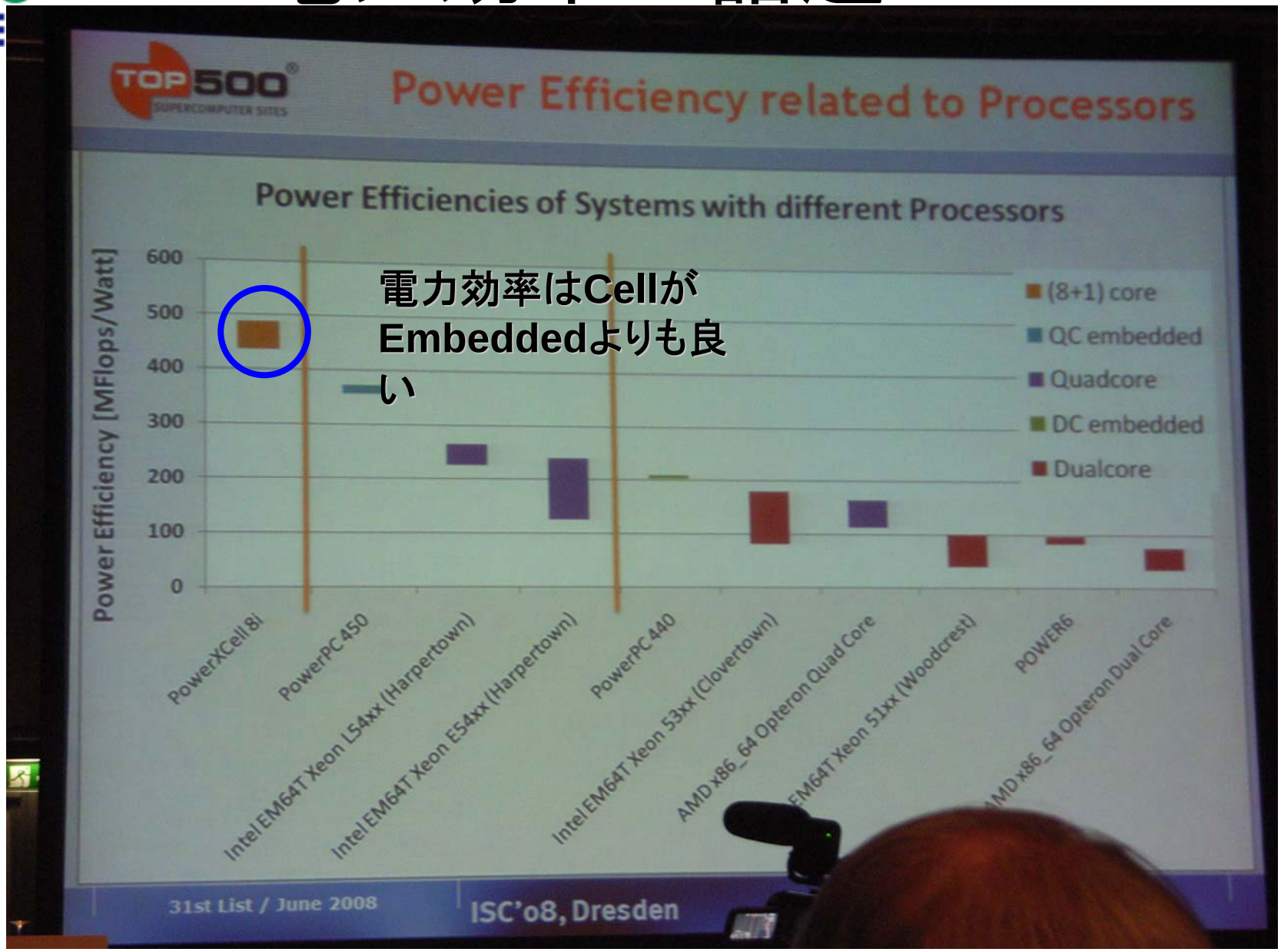
Bellの法則？


TOP500
HPC Computer Classes and Bell's Law
Bell's Law states, that:
Important classes of computer architectures come in cycles of about 10 years.
It takes about a decade for each phase
Early research
Early adoption and maturation
Prime usage
Phase out past its prime
Can we use Bell's Law to classify computer architectures in the TOP500?
ISC'08, Dresden

新たな潮流の登場


TOP500
SUPERCOMPUTER SITES
Bell's Law
Computer Classes / Systems
Embedded/
Accelerated
Commodity Cluster
Custom Scalar
Vector/SIMD
500
400
300
200
100
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
Embedded/ Accelerated
Commodity Cluster
Custom Scalar
Vector/SIMD
31st List / June 2008
ISC'08, Dresden

電力効率が話題に


TOP500
SUPERCOMPUTER SITES
Power Efficiency related to Processors
Power Efficiencies of Systems with different Processors
Power Efficiency [MFlops/Watt]
600
500
400
300
200
100
0
電力効率はCellがEmbeddedよりも良い
(8+1) core
QC embedded
Quadcore
DC embedded
Dualcore
PowerXCell 8i
PowerPC 450
Intel EM64T Xeon L54xx (Harpertown)
Intel EM64T Xeon E54xx (Harpertown)
PowerPC 440
Intel EM64T Xeon 53xx (Clovertown)
AMD x86_64 Opteron Quad Core
EM64T Xeon 51xx (Woodcrest)
POWER6
MD x86_64 Opteron Dual Core
31st List / June 2008
ISC'o8, Dresden

新しいトレンド？

- Accelerator
  - Cell
  - GPU
  - GRAPE
  - FPGA
  - ClearSpeed

- Enbedded
  - BlueGene/L,
  - BlueGene/P
  - BlueGene/Q

第3. 5世代PCクラスター

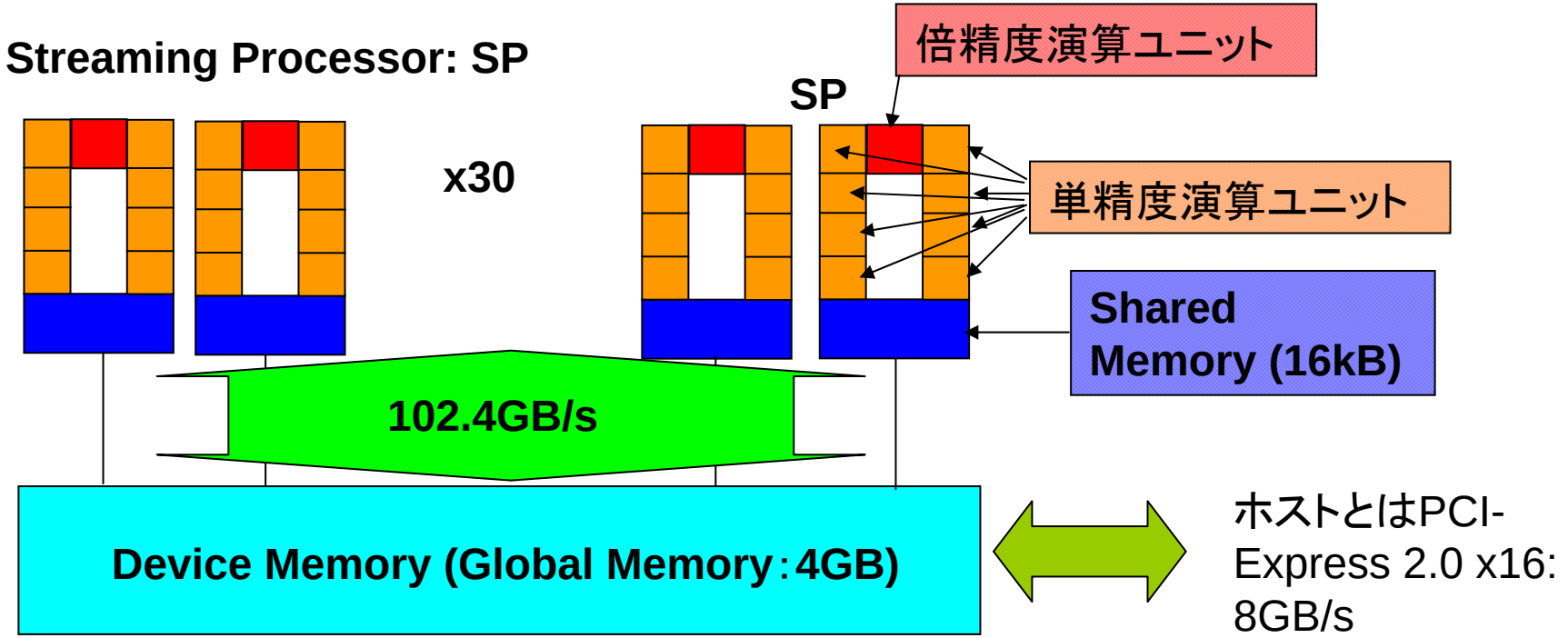
・マルチコアーPC　CPU

・マルチソケット/ノード

→　メモリー空間大

・アクセラレータ

GPUのハードウェア構造



※理研GPU講習会NEC資料を元にした

NVIDIA Tesla

構成	機種名	GPU数	クロック MHz	シェーダ		メモリ					性能 (GFLOPS)	形状など
				スレッドプロセッサ数	クロック (MHz)	最大帯域幅 (GB/s)	バス規格	バス幅 (bit)	総容量 (MiB)	クロック (MHz)		
GPUコンピューティングプロセッサ[1]	C870	1	600	128	1350	77	GDDR3	384	1536	1600	519	フルハイトビデオカード
デスクサイト・スーパーコンピュータ[1]	D870	2	600	256	1350	154	GDDR3	384	3072	1600	1037	デスクサイドシステム/ラックマウント装置
GPUコンピューティングサーバ[1]	S870	4	600	512	1350	307	GDDR3	384	6144	1600	2074	1Uラック
C1060 コンピューティングプロセッサ [2]	C1060	1	602	240	1300	102	GDDR3	512	4096	1600	936	フルハイトビデオカード IEEE 754r 機能
S1070 1U GPUコンピューティングサーバ[2]	S1070	4	602	960	1500	410	GDDR3	512	16384	1600	3744	1Uラック IEEE 754r 機能

注釈

- [1] NVIDIA が明らかにしていない仕様については、GeForce 8800GTX からの推測。
- [2] NVIDIA が明らかにしていない仕様については、GeForce GTX 280 からの推測。

http://ja.wikipedia.org/wiki/NVIDIA_Tesla
出典:wikipedia

PCとServer、Teslaの比較(1)

	PC	⇒	PC Server	⇒	nvidia Tesla C1060
CPU	Intel Core2 Duo E7500 2.93GHz single socket		Intel Xeon X5570 2.93GHz x 2Sockets		
core数	2	x 4	8 (4/Socket x 2 Sockects)	x 30	240
周波数	2.93 GHz	x 1	2.93 GHz	x 0.44	1.3 GHz
Peak Performance	**23.44 GFLOPS**	**x 4**	**93.76 GFLOPS**	**x 10** **x 0.83**	**933 GFLOPS(SP)** **78 GFLOPS(DP)**
消費電力	250 W	x 3	700 W	x 0.27	187.8 W
メモリ転送性能	DDR2-1066 8.533 GB/s	x 6	DDR3-1066 (3channel/CPU) 51.18GB/s	x 2	GDDR3 102GB/s
価格(姫野が見積もったもの)	約6万円		約100万円		約8万円

PCとServer、Teslaの比較(2)

	PC	SERVER	GPGPU		PC比	Server比
	Intel Core2 Duo	Intel Xeon X5570	nvidia Tesla C1060			
価格性能比 (GFLOPS/¥k)	0.391	0.094	SP	11.663	29.9	124.4
			DP	0.975	2.5	10.4
電力性能比 (GFLOPS/W)	0.094	0.134	SP	4.968	53.0	37.1
			DP	0.415	4.4	3.1

単精度の計算では非常に性能が高い

しかし、倍精度ではそれほど効果が高くない

PCクラスタに関する
理研での取り組み

2009年6月まで運用していたRSCC

RIKEN Super Combined Cluster

【システム構成】
スカラ＋ベクトル＋専用機
の複合システム

【スカラ部】
12.4テラフロップス

【ベクトル部】
0.28テラフロップス

入出力機器

【システム間接続】
1ギガビット毎秒のネットワークを使いグリッド接続

MD-GRAPE3：64テラフロップス追加(07年)

産業技術大賞
文部科学大臣賞・受賞
2005年4月

【特徴】

- 世界初のスカラ+ベクトル+専用機複合システム
  - 計算機センターでPCクラスタを採用
- 日本で初めてグリッド技術を全面的に採用した計算機センター
  - 利用者に利用計算機を意識させない
- 世界最大規模、日本で最速のPCクラスタ
  - Top500リスト（2004年6月）第7位
  - 高性能で低コスト

次世代スーパーコンピュータ開発のテストベッドとして使用

第2世代PCクラスタ：RSCCの狙い

1. スーパーコンピュータの既成概念に縛られず、コスト性能比の良い計算機をメインの計算機に
2. 新たな利用者の獲得
   - 実験データ処理、バイオインフォマティックスに適したシステムとし、理研内の新たな利用者を取り込む
   - Grid Computing、Web技術を用いて、それまで計算機センターのシステムに不慣れな研究者でも簡単に使えるシステムを構築
   - フリーソフトが多数利用できる計算機システムとすること
3. 従来の利用者も利用できる環境は維持

RSCCシステム開発と主な成果

LINPACK性能で
世界7位

産業技術大賞
文部科学大臣賞・受賞

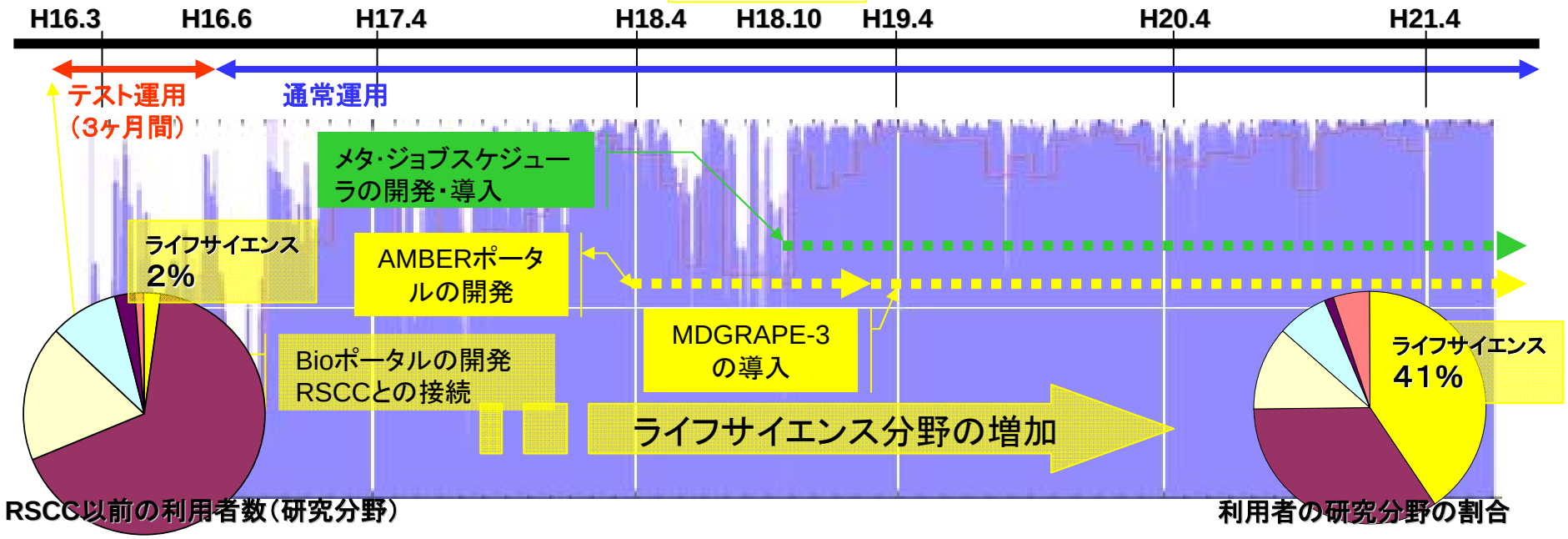
主なプレス発表

★電子の磁石の強さを1兆分の1の精度まで計算

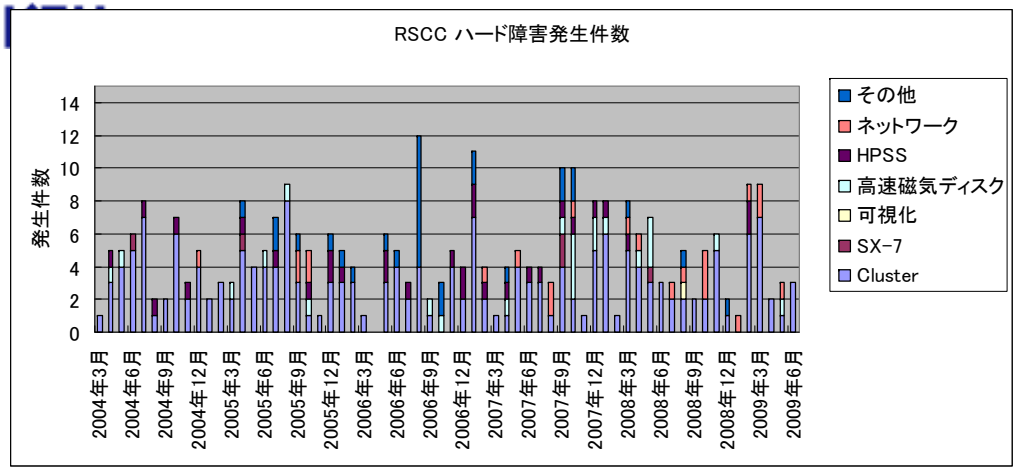
★汚い"物質中の電子が持つ美しい対称性『共形不変性』を世界で初めて実証 - 不規則系の臨界現象における理論手法の構築の第一歩 -

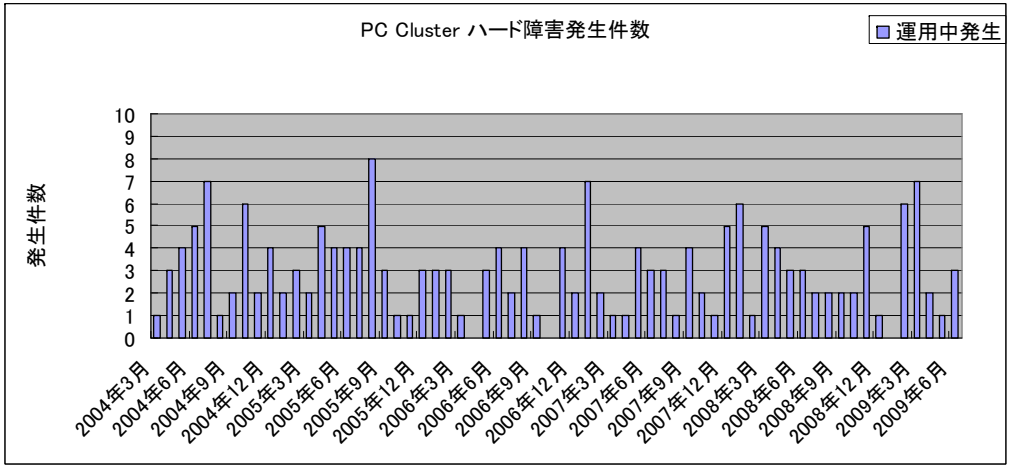
★素粒子の世界の真空エネルギーをコンピュータで計算

★金属表面で起こる分子の選択的分解反応と表面拡散運動の可視化に成功



RSCCの５年間の故障





- 2009年6月末までの統計データ
- LINPACK測定中のハード障害：
  2004年3月1/1、4月11/14、5月 6/11（トータルで、19台/33台）
- 予防交換は導入当初（2004年）の
  - Blade server nodes:128/128
  - InfiniBand ケーブル:512/1024
- それ以降の予防保守は大半がメモリの1ビットエラー検出による交換
- 初期不良と予防交換を除いたクラスタの平均故障率は約3台/月

第3.5世代PCクラスタ：RICC

RICCの狙い

1. 次世代スーパーコンピュータに向けたアプリケーション開発環境の整備
   - 大規模並列に対応するために、8000コア超の大規模並列ジョブ実行を推進
   - システム・ソフトウェア（ジョブ・スケジューラ）の機能強化
     - 一般利用の範囲で8000並列の大規模並列ジョブ実行を可能に
2. 新しい方向性：GPGPUアクセラレータへの挑戦
   - アクセラレータは時代の要請であり、トレンド
   - GPGPUアクセラレータを導入、利用を推進

RICCの概要

【システム構成】
超並列PCクラスタ＋GPUクラスタ＋専用機クラスタ＋大容量メモリ計算機
を単一の高速ネットワークで接続したクラスタ・オブ・クラスタ

実験データ

研究者

【超並列PCクラスタ】96.0テラフロップス

＊最新のCPUを採用した日本初の大規模PCクラスタ（8192 コア）

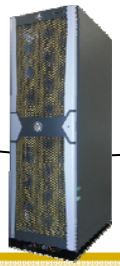
【多目的PCクラスタ】9.3テラフロップス
＋【GPGPUアクセラレータ】93.3TFLOPS

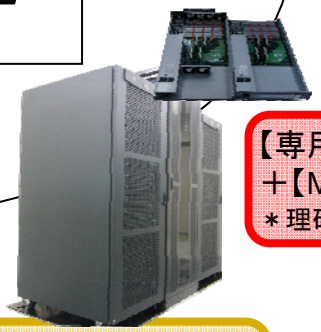
GPGPUの利用を容易にするためのビジュアル・プログラミング環境を日本IBMと共同で開発

2009年導入時点のTOP500リストで世界40位、日本で3位、PCクラスタ・システムでは日本最速（世界では11位）

テープアーカイブ装置
【HPSS】(4PB)

磁気ディスク装置
(550TB)





【専用機クラスタ】3TFLOPS
＋【MDGRAPE-3】64テラフロップス
＊理研で開発した分子動力学専用計算機を接続

【大容量メモリ計算機】
0.24テラフロップス、512GBメモリ
＊1プロセスで500GB以上のメモリを利用可能

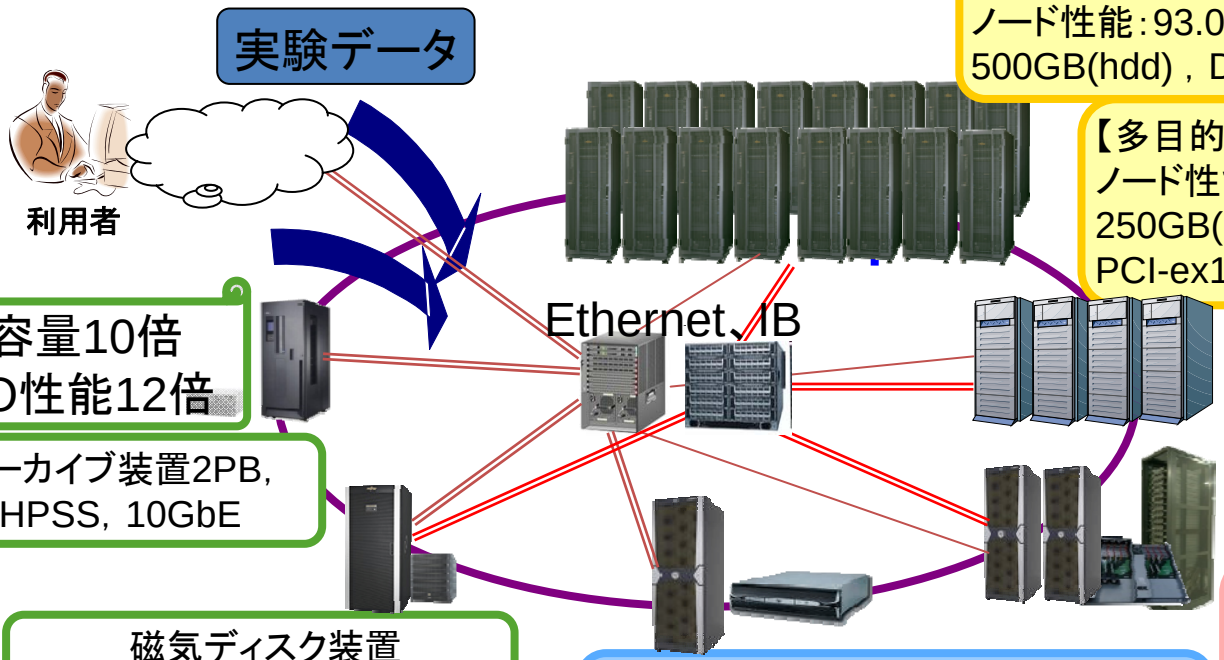
RICCとRSCCの比較

【システム構成】
PCクラスタ＋大容量メモリ計算機＋アクセラレータ

演算性能：8.5倍
メモリI/O性能：2.5倍

【超並列PCクラスタ】1024Nodes（8192core)
ノード性能：93.0GFLOPS, 12GB(mem),
500GB(hdd)，DDR IB×1

【多目的PCクラスタ】100Nodes(800core)
ノード性能：93GFLOPS, 24GB(mem),
250GB(hdd)，DDR IB×1,
PCI-ex16レーン×1



容量10倍
I/O性能12倍

アーカイブ装置2PB,
HPSS，10GbE

磁気ディスク装置
550TB，SRFS，DDR IB

容量27倍
I/O性能10倍

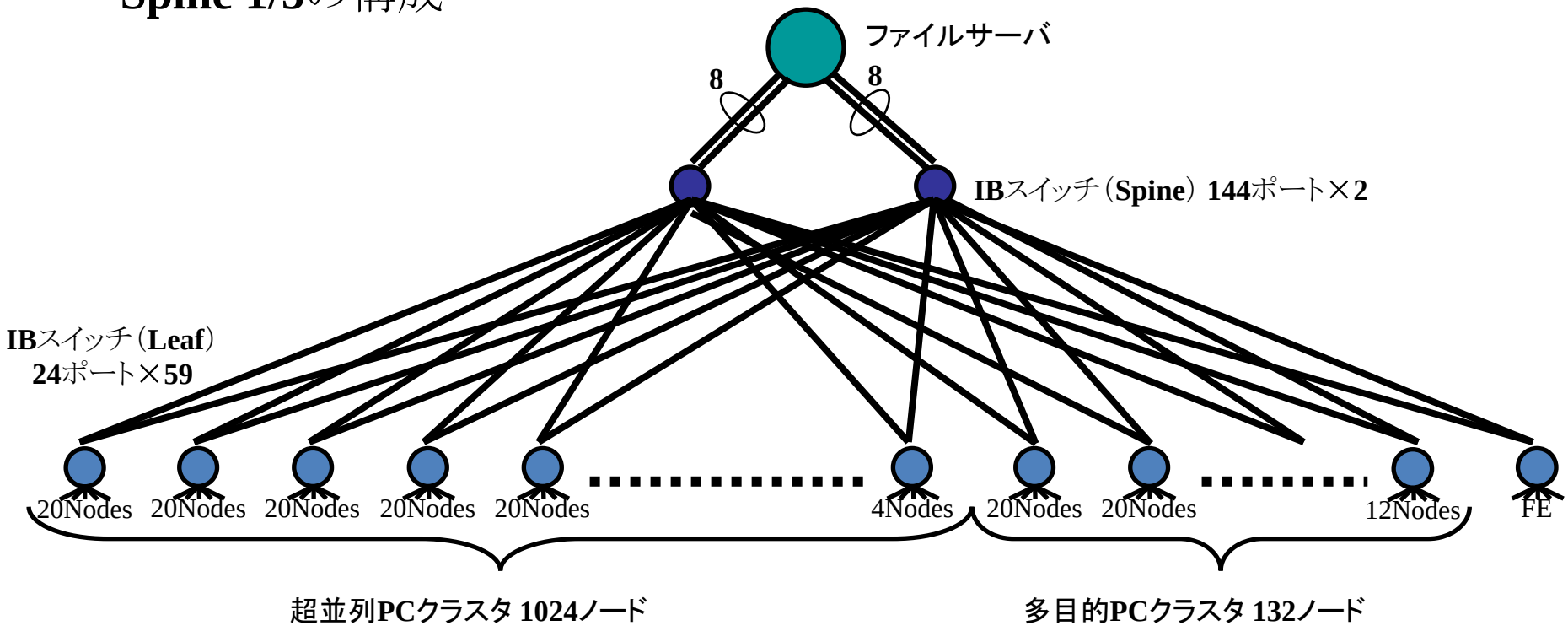
【大容量メモリ計算機】1Node(36core)
0.24TFLOPS、512GB(mem)
PCI-X，10GbE

メモリ容量を2倍

【分子動力学専用計算機】
64TFLOPS
ホストノード:32Nodes,
32GB/Node，DDR IB×1/Node

最大で8996 coreを同時に使えるインターコネクト（InfiniBand）構成

FBB構成よりも**Leaf 2/3**,
Spine 1/5の構成



All Rights Reserved, Copyright (c) RIKEN 2009-

メタ・ジョブ・スケジューラの機能

- 複数のクラスタ上で動作しているジョブ・スケジューラの上位で動作
- 異なるノード構成、アプリケーションなどを管理
- サブクラスタ間のロードバランス等の解消
- 様々なスケジューリングポリシー（優先順位など）での運用可能
- フェアシェア機能、バックフィル機能

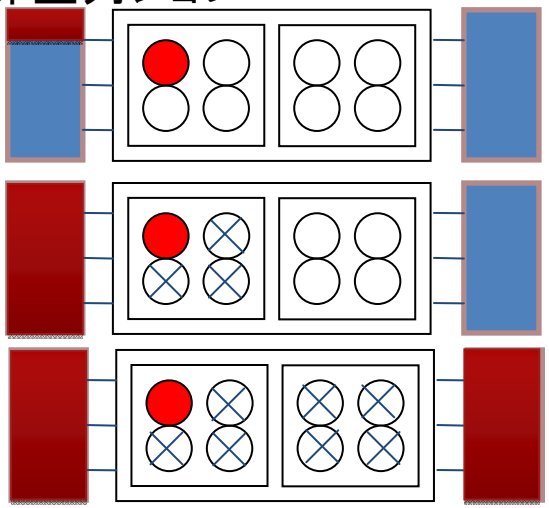
- マルチ・コアシステムでの効率的なジョブ管理・リソース管理(NEW!)
- ネットワーク・トポロジと利用方針を踏まえたノード・アロケーション管理(NEW!)
- 数万の単一CPU利用ジョブのスケジューリングに対応(NEW!)
- サーバ障害時のフェイルオーバーに対応(NEW!)

メタ・ジョブ・スケジューラ新機能（1）

マルチ・コアシステムの効率的なジョブ・リソース管理

- RICC（PCクラスタ）はquad coreのCPUを1ノードに2つ搭載
- 非並列、並列ジョブ（スレッド、プロセス）が混在

非並列ジョブ

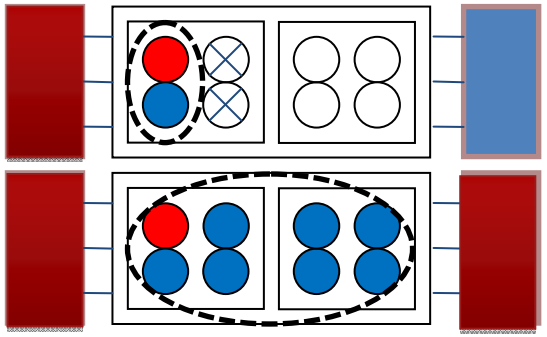


空きcoreは他ジョブが使用可能

1CPUを占有
空きcoreは他ジョブが使用可能

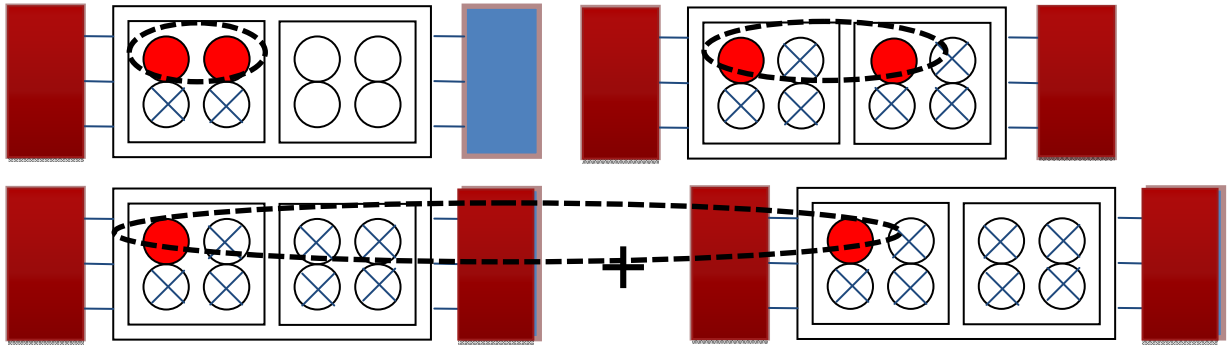
1ノードを占有

スレッド並列ジョブ(1プロセス)



利用者のニーズに合わせて、柔軟なリソースの利用を可能に

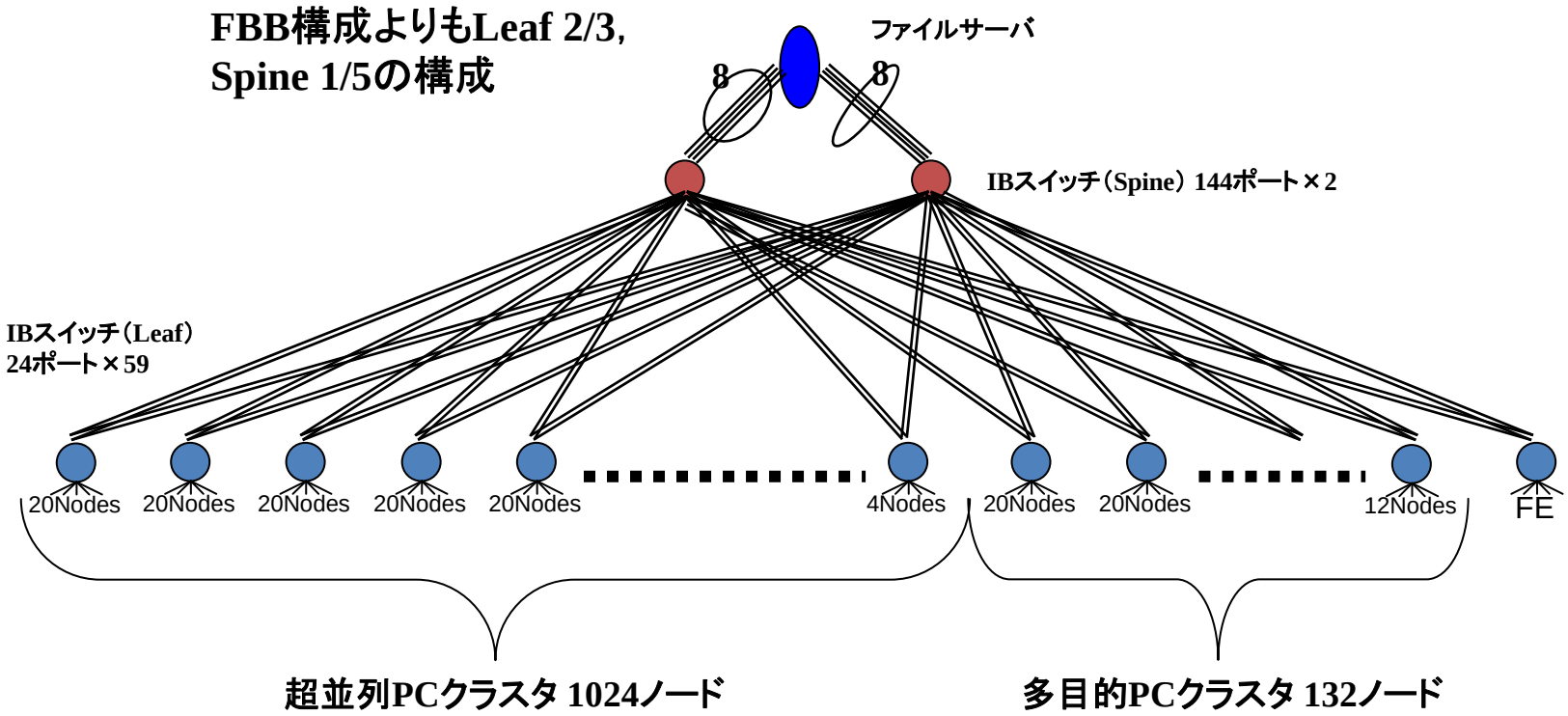
2プロセス並列ジョブ



メタ・ジョブ・スケジューラ新機能（2）

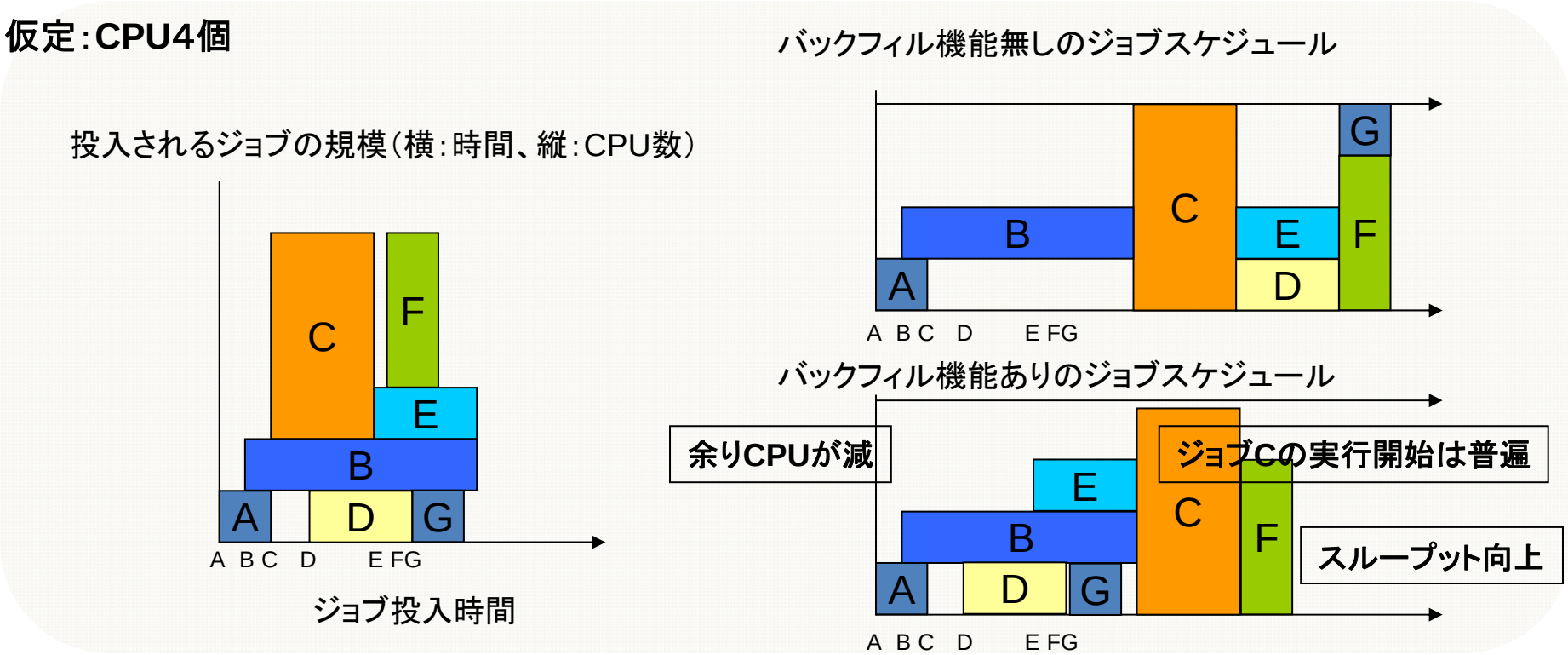
- ネットワーク・トポロジを踏まえたジョブのノード・アロケーション管理が必須

InfiniBandネットワーク構成



メタ・ジョブスケジューラの新機能(3)
大規模並列ジョブを日常的に実行可能に

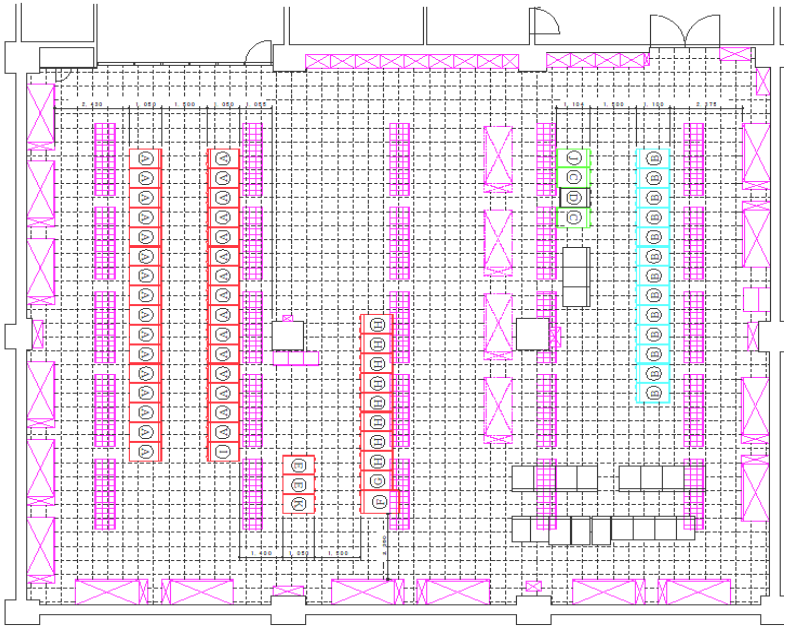
- ジョブ・スケジューラの機能強化（バックフィル）
  - ジョブを投入する際に実行時間を指定すると、効率的にジョブを実行
  - ジョブの平均待ち時間を削減



メタ・ジョブ・スケジューラ新機能（4）

- 実験データ解析やパラメータ・サーチなどでは、非並列ジョブを大量に実行
  - たとえば、高エネルギー実験データ解析では、1人で5千～1万本のジョブを投入
  - ジョブをまとめて1つのジョブとして投入可能にジョブ・スケジューラを改良（バルク・ジョブ）
- システム全体が大きくなり、ジョブ・スケジューラの重要性は増大
  - ジョブ・スケジューラの障害による影響大！
  - サーバを2重化し、障害時のフェイルオーバーに対応可能なようにソフト改修

システム緒元



	RSCC	RICC
理論性能	12.6TFLOPS	198.8TFLOPS + 93.3TFLOPS(SP)
設置面積（テープアーカイブ装置＆保守スペース除く）	約40㎡	約47㎡
重量	約24t	約40t
消費電力（ピーク）	約660kVA	約850kVA
発熱量	約550Mcal/h	約710Mcal/h

マシン室写真

超並列PCクラスタ

増設した空調機

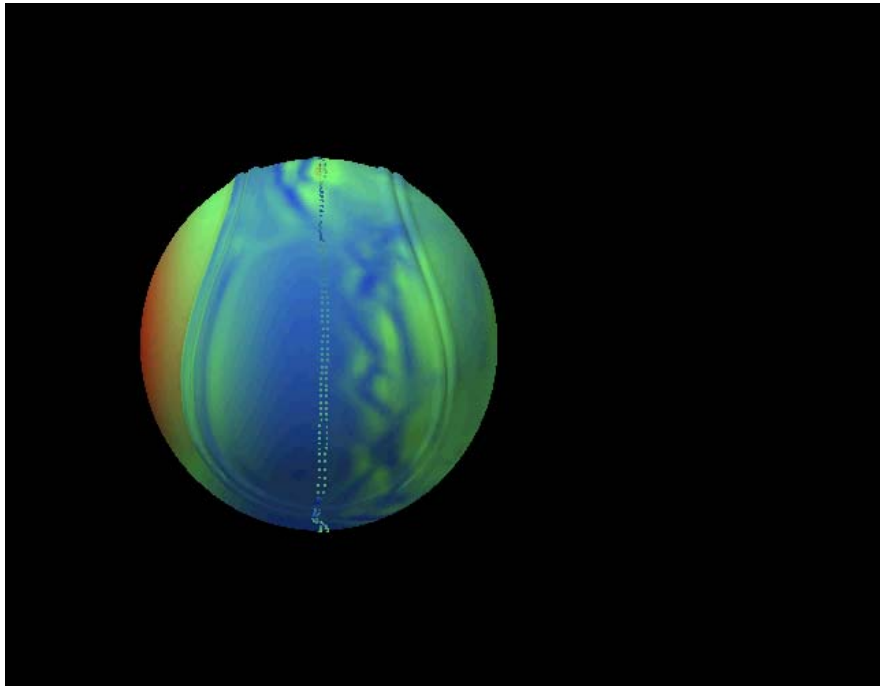
多目的PCクラスタ（GPGPU搭載）

MDGRAPE-3クラスタ＆
大容量メモリ計算機

RICCの性能

himenoBMTを使った性能測定

- himenoBMTとは
  - 非圧縮性のNavier-Stokes方程式のソルバーのカーネル（流体シミュレーション）
  - 物体適合格子を使った差分法
  - カーネルは圧力のポアソン方程式のソルバー(元はSOR法）



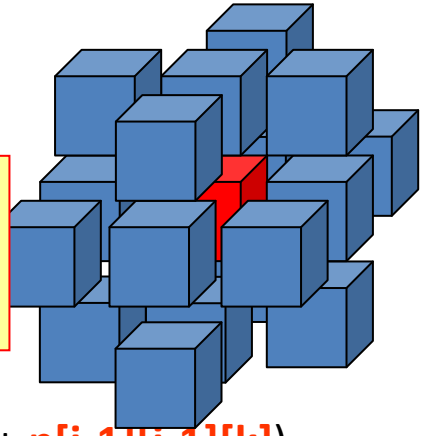
himenoBMTの特徴

- メモリーアクセスの特徴
  - 14個の 3次元配列
  - １つだけ再利用
  - 13個の配列は一度だけしか参照しない
  - キャッシュが効かない
- 性能のボトルネックはメモリーバンド幅
  - 14ストリームのデータ供給：高バンド幅

himenoBMTのカーネル・コード

```
for (i=1; i<imax-1; i++)
for (j=1; j<jmax-1; j++)
for (k=1; k<kmax-1; k++) {
s0 = a0[i][j][k] * p[i+1][j][k]
+ a1[i][j][k] * p[i][j+1][k]
+ a2[i][j][k] * p[i][j][k+1]
+ b0[i][j][k] * (p[i+1][j+1][k] – p[i+1][j-1][k] – p[i-1][j+1][k] + p[i-1][j-1][k])
+ b1[i][j][k] * (p[i][j+1][k+1] – p[i][j+1][k-1] – p[i][j-1][k+1] + p[i][j-1][k-1])
+ b2[i][j][k] * (p[i+1][j][k+1] – p[i+1][j][k-1] – p[i-1][j][k+1] + p[i-1][j][k-1])
+ c0[i][j][k] * p[i-1][j][k]
+ c1[i][j][k] * p[i][j-1][k]
+ c2[i][j][k] * p[i][j][k-1]
+ wrk1[i][j][k];
ss = (s0 * a3[i][j][k] – p[i][j][k]) * bnd[i][j][k];
wrk2[i][j][k] = p[i][j][k] + omega * ss;
}
```

配列P
差分ステンシルアクセス
再利用

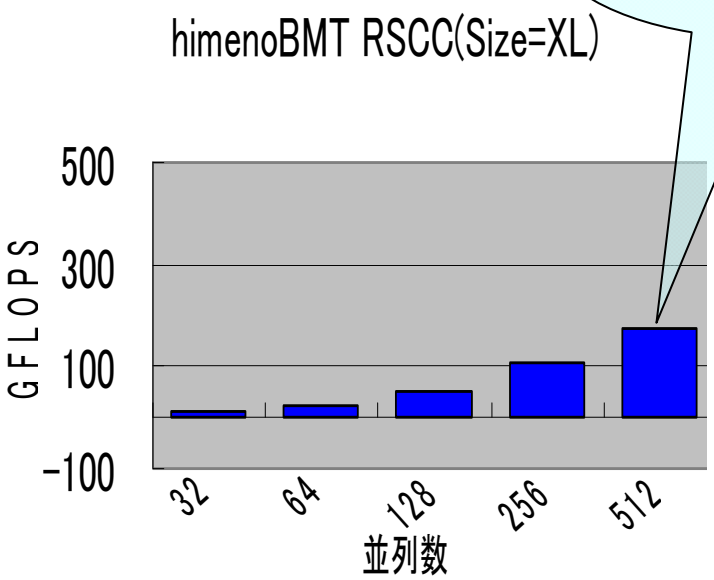


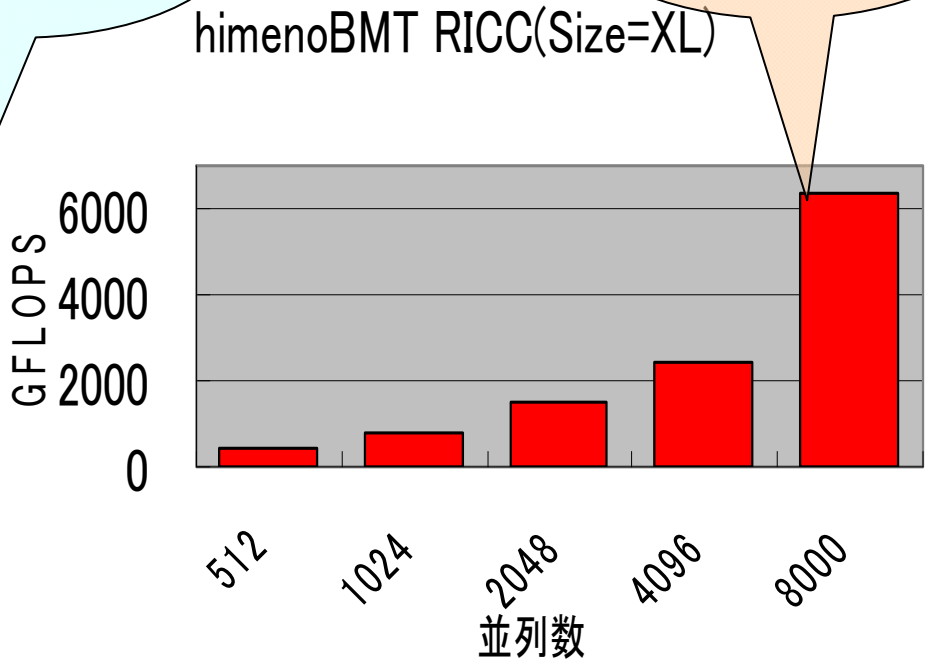
他の１３の配列
点アクセス
再利用無し

クラスター上でのhimenoBMT

512CPUで
174GFLPS

8000coreで
6.4TFLPS





GPGPUアクセラレータの理論性能

100ノード

＋

100枚

- 理論性能

9.3 TFLOPS

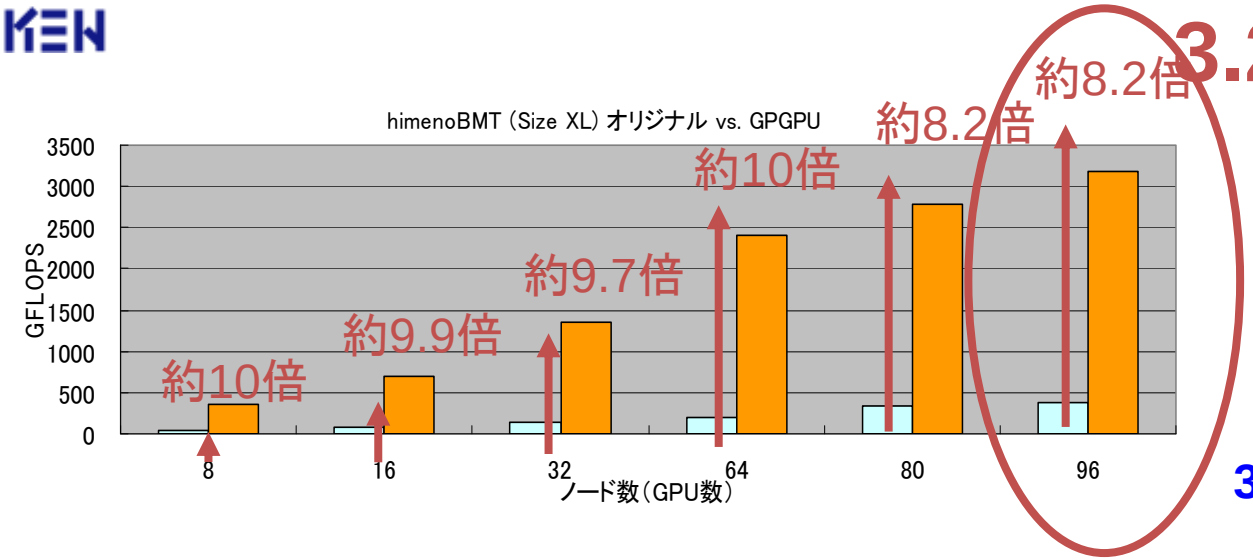
＋

- 理論性能（単精度）

93.3 TFLOPS

10倍

RICCでのGPGPUの性能測定結果



himenoBMT GPGPU版は(株)富士通研究所提供

理論値は

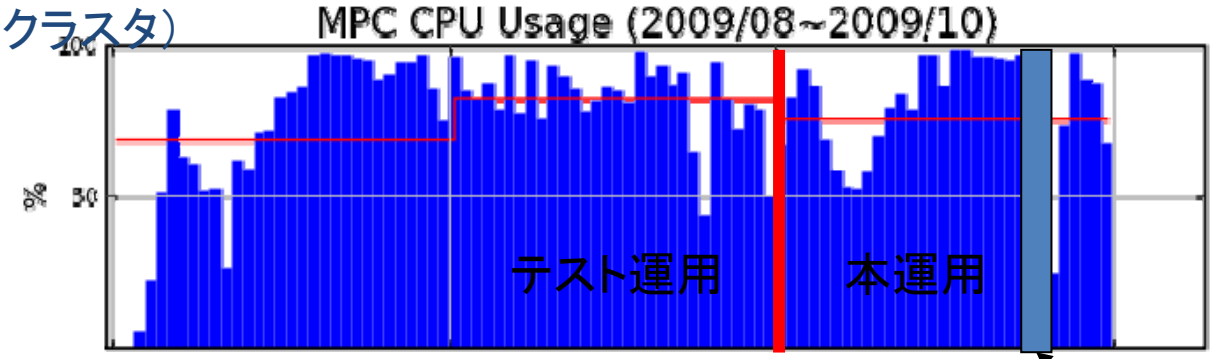
11倍

9.3 TFLOPS → 102.6 TFLOPS

RICC運用状況

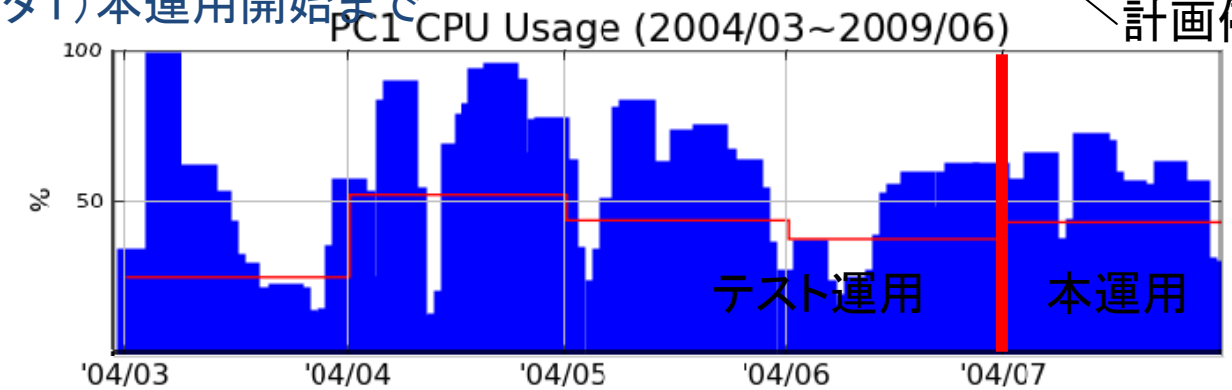
超並列PCクラスタ（mpc）の利用率

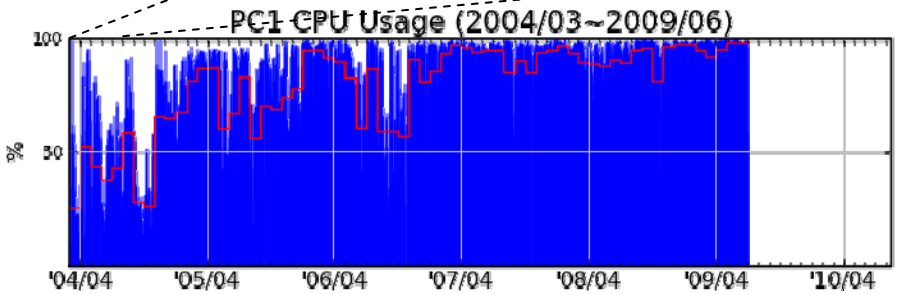
RICC（超並列PCクラスタ）



計画停電

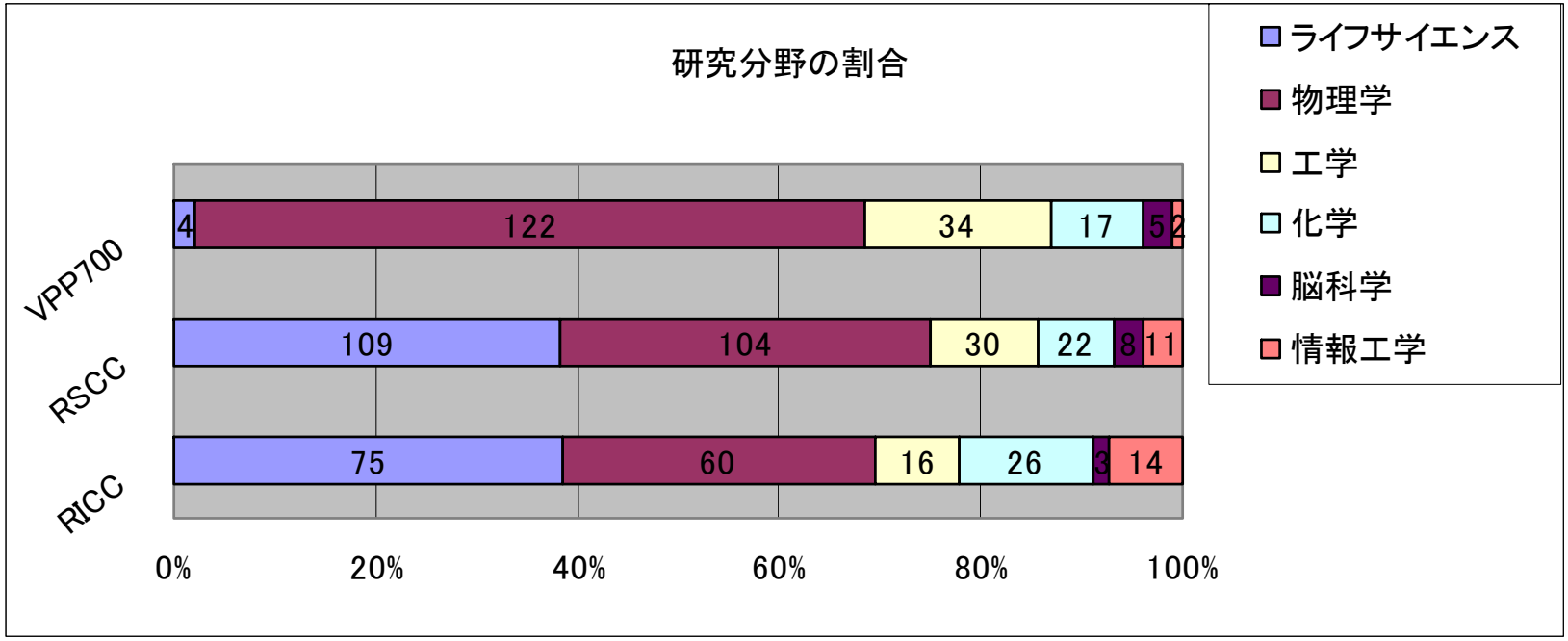
RSCC（PCクラスタ1）本運用開始まで





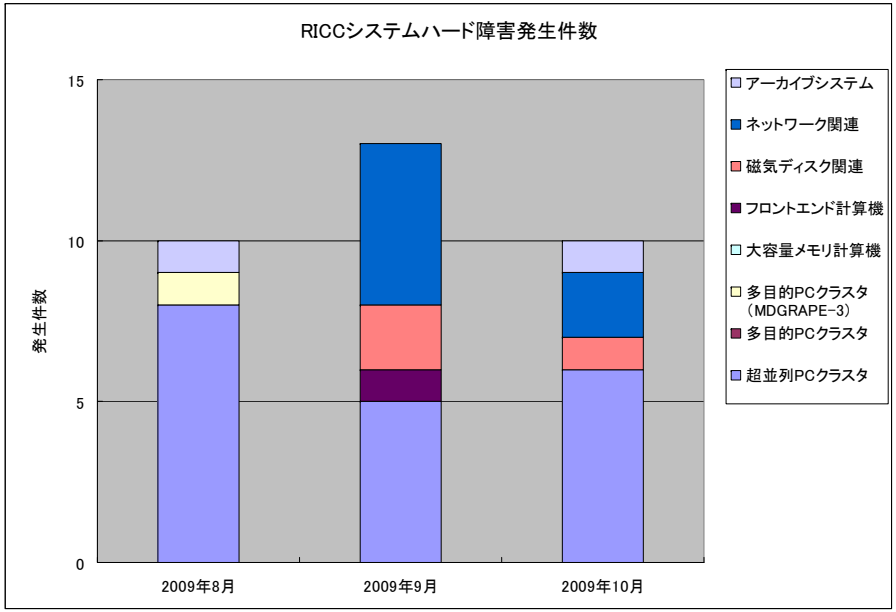
5年間のRSCC（PCクラスタ1）利用率

VPP700/RSCCとの比較



- VPP700（2004年2月）、RSCC(2009年6月末現在）とRICC（2009年10月末）の登録ユーザーの研究分野、主務による分類比較
- 登録ユーザー数は、VPP700で184名、RSCCで276名、RICCは144名
- VPP700と比べると、研究分野では、ライフサイエンスが大幅に増大している。

RICCハードウェア故障



- 2009年10月末までの統計データ
- 予防交換を除いたクラスタの平均故障率は約7台/月
- 9月、10月の磁気ディスク装置の障害では、home領域へのアクセス不可により、システム利用が停止

GPUの利用促進に向けて

GPUプログラムの問題

LU分解のオリジナル・プログラム（一部）

```
void kernelD(
        const Matrix<T, Z, C>& blockB,
        const Matrix<T, R, Z>& blockC,
        const Matrix<T, R, C>& blockD,
        Matrix<T, R, C>& result)
{
        struct timeval tvs, tve;
        std::stringstream ss;
        int i, j, k;
        gettimeofday(&tvs,NULL);
        ss << tvs.tv_sec << "." << tvs.tv_usec << " kernelD" << R << " start."  << std::endl;
        std::cerr << ss.str();
        ss.str("");

        /* To make the code simpler, input matrix is copied to the output one first */
        for(i = 0; i < R; i++) // row
            for(j = 0; j < C; j++) // column
                result.elementAt(i, j) = blockD.elementAt(i, j);

        /* Main loop of submatrix calculation */
        for (i = 0; i < R; i++) // row
            for (k = 0; k < Z; k++) // column or row
                for (j = 0; j < C; j++) // column
                    result.elementAt(i, j) += blockB.elementAt(k, j) * blockC.elementAt(i, k);

        gettimeofday(&tve,NULL);
        ss << tve.tv_sec << "." << tve.tv_usec << " kernelD" << R << " finish."  << std::endl;
        tve.tv_usec -= tvs.tv_usec;
        tve.tv_sec -= tvs.tv_sec;
        if( tve.tv_usec < 0 ){
            tve.tv_usec += 1000000;
            tve.tv_sec--;
        }
        ss << tve.tv_sec << "." << tve.tv_usec << " kernelD" << R << " used."  << std::endl;
        std::cerr << ss.str();
        ss.str("");
}
```

GPGPU版プログラム（一部）

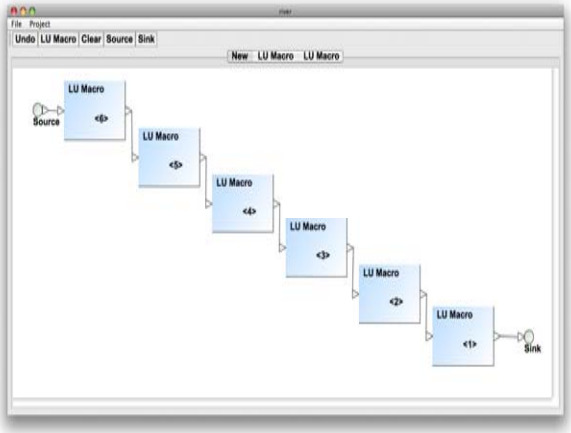
```
void kernelD(
  Matrix<float,MATRIX_SIZE,MATRIX_SIZE>* blockD,
  Matrix<float,MATRIX_SIZE,MATRIX_SIZE>* blockB,
  Matrix<float,MATRIX_SIZE,MATRIX_SIZE>* blockC,
  Matrix<float,MATRIX_SIZE,MATRIX_SIZE>* result);

extern "C"
void* udopLU_D(void* parm)
{
  uspade_udop_parm_t* uparm = (uspade_udop_parm_t*)parm;
  std::string blockDParm = "127.0.0.1:10003";
  std::string blockBParm = "127.0.0.1:10001";
  std::string blockCParm = "127.0.0.1:10002";
  std::string resultParm = "127.0.0.1:10004";

  for (std::map<std::string, std::string>::const_iterator it = uparm->parms.begin(); it != uparm->parms.end(); it++)
  {
    size_t pos;
    while ( (pos = blockDParm.find(it->first)) != std::string::npos ) blockDParm.replace(pos, it->first.length(), it->second);
    while ( (pos = blockBParm.find(it->first)) != std::string::npos ) blockBParm.replace(pos, it->first.length(), it->second);
    while ( (pos = blockCParm.find(it->first)) != std::string::npos ) blockCParm.replace(pos, it->first.length(), it->second);
    while ( (pos = resultParm.find(it->first)) != std::string::npos ) resultParm.replace(pos, it->first.length(), it->second);
  }

  InSocketPort<Matrix<float,MATRIX_SIZE,MATRIX_SIZE> > blockDPort(blockDParm);
  InSocketPort<Matrix<float,MATRIX_SIZE,MATRIX_SIZE> > blockBPort(blockBParm);
  InSocketPort<Matrix<float,MATRIX_SIZE,MATRIX_SIZE> > blockCPort(blockCParm);
  OutSocketPort<Matrix<float,MATRIX_SIZE,MATRIX_SIZE> > resultPort(resultParm);

  Matrix<float,MATRIX_SIZE,MATRIX_SIZE> blockD;
  Matrix<float,MATRIX_SIZE,MATRIX_SIZE> blockB;
  Matrix<float,MATRIX_SIZE,MATRIX_SIZE> blockC;
  Matrix<float,MATRIX_SIZE,MATRIX_SIZE> result;

  while ( uparm->active )
  {
    if ( uparm->active ) blockDPort.receive(blockD);
    if ( uparm->active ) blockBPort.receive(blockB);
    if ( uparm->active ) blockCPort.receive(blockC);

    struct timeval tv_st, tv_ed;
    gettimeofday(&tv_st, NULL);

    if ( uparm->active ) kernelD(
        &blockD,
        &blockB,
        &blockC,
        &result);

    gettimeofday(&tv_ed, NULL);
    printf("kernel fired!! (at %f in msec, %f [msec] to process kernel)¥n",
        (double)tv_ed.tv_sec * 1000 + (double)tv_ed.tv_usec / 1000,
        (double)(tv_ed.tv_sec - tv_st.tv_sec - 1) * 1000 + (double)(1000000 + tv_ed.tv_usec - tv_st.tv_usec) / 1000);

    if ( uparm->active ) resultPort.send(result);
  }

  return NULL;
}
```

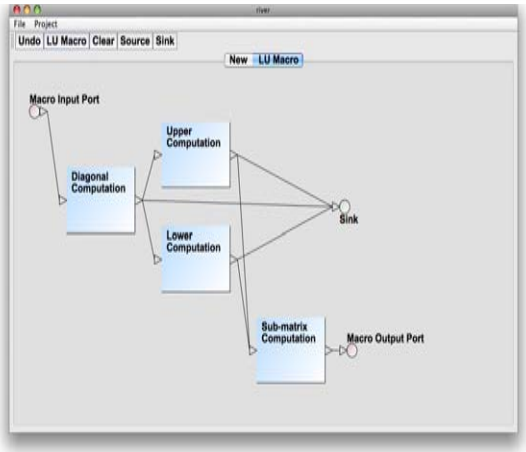
GPGPUアプリケーション開発環境

RIVER(Riken IBM Visual Programming EnviRonment)

- GPGPUは高速だが、その性能を引き出すには高度なプログラムのスキルが必要
- だれでも使えるように初心者向けの、ビジュアル・プログラミング環境を日本IBMと共同で開発中
- 部品ライブラリの中の部品を組み合わせるだけで、プログラミングが可能
- ノード並列もサポート

GPUが使えるコンパイラも近々利用可能になる予定
PGIコンパイラー（現在ベータ版）





連立一次方程式の前処理プロセス：LU分解の例

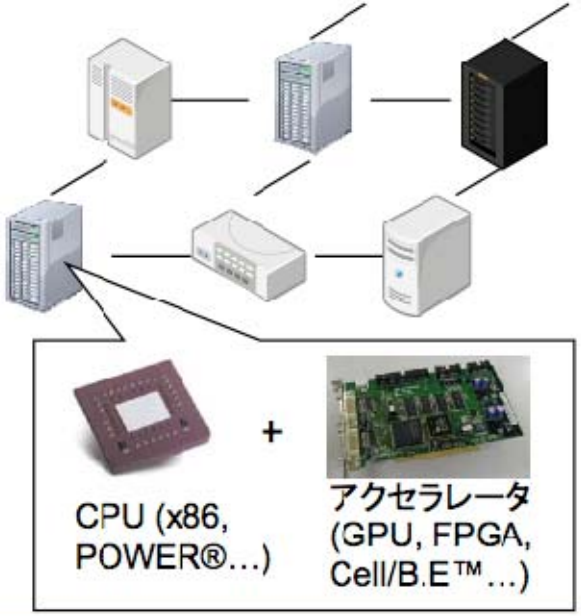
背景

課題

- アーキテクチャの異なるプロセッサーやアクセラレータが多種混在したハイブリッドシステムでは、性能を引き出すプログラムを自動生成もしくは、人手でコーディングすることは困難である

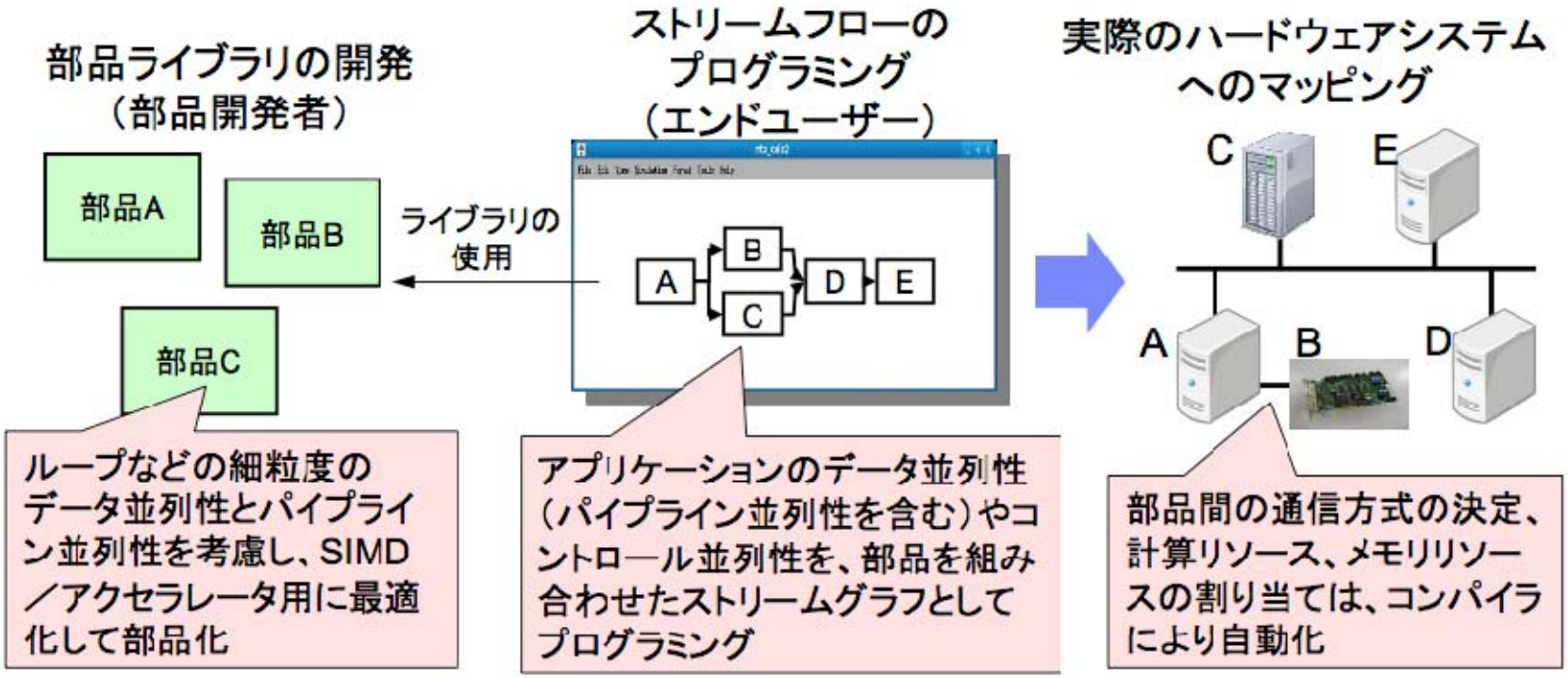
ハイブリッドシステムの例

- 例：BlueGene, Roadrunner, x86とGPGPUで構成されるPCクラスタ



© 2009 IBM Corporation

われわれのアプローチと利点



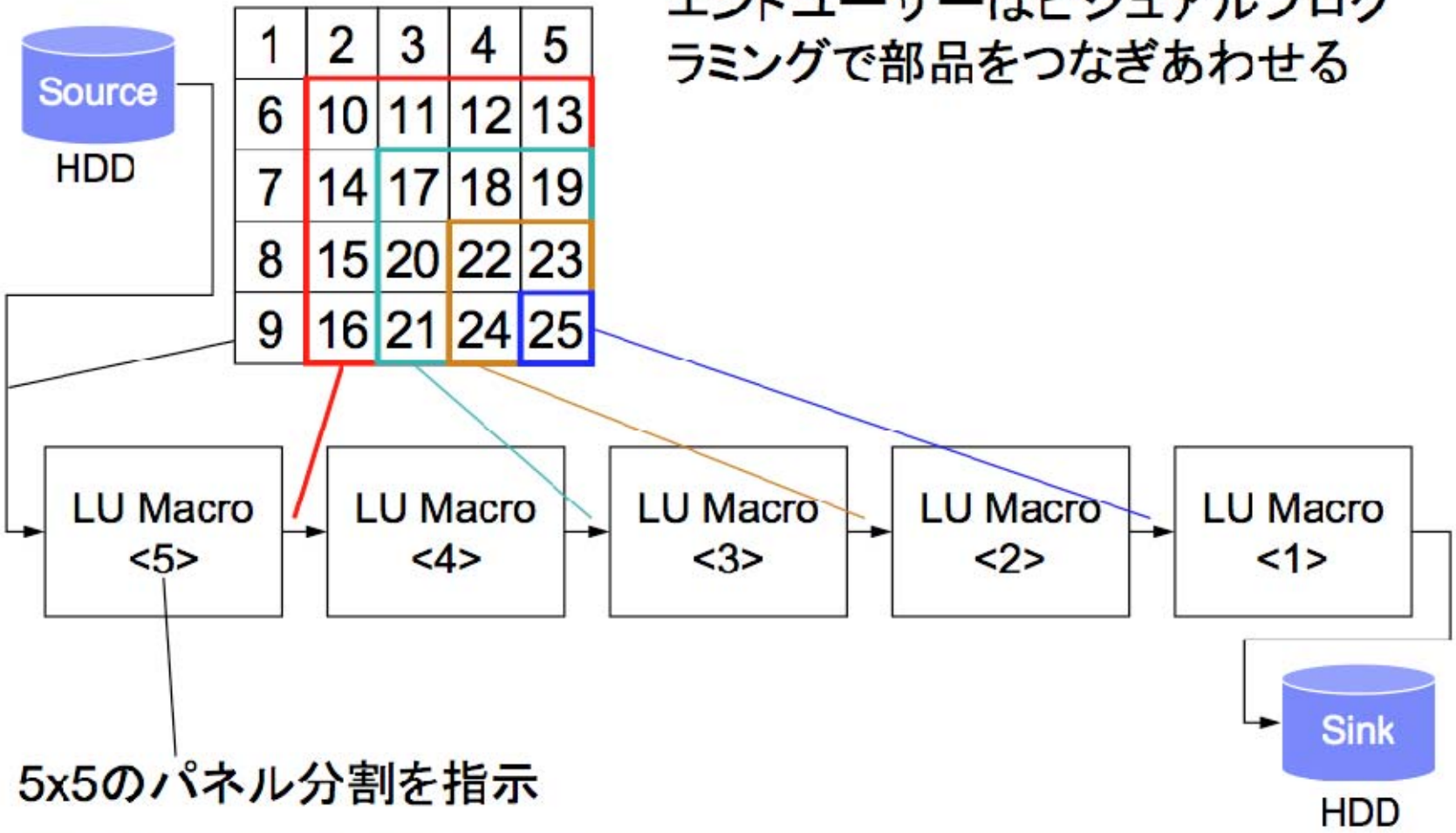
利点

- 多くの人は細粒度の最適化を気にすることなく、トップレベルのアルゴリズムの記述だけでスーパーコンピューティングを利用できる
- アルゴリズム記述とHW特化の最適化を分離することで、アプリケーションのポータビリティーが向上する

© 2009 IBM Corporation

ユーザープログラミング

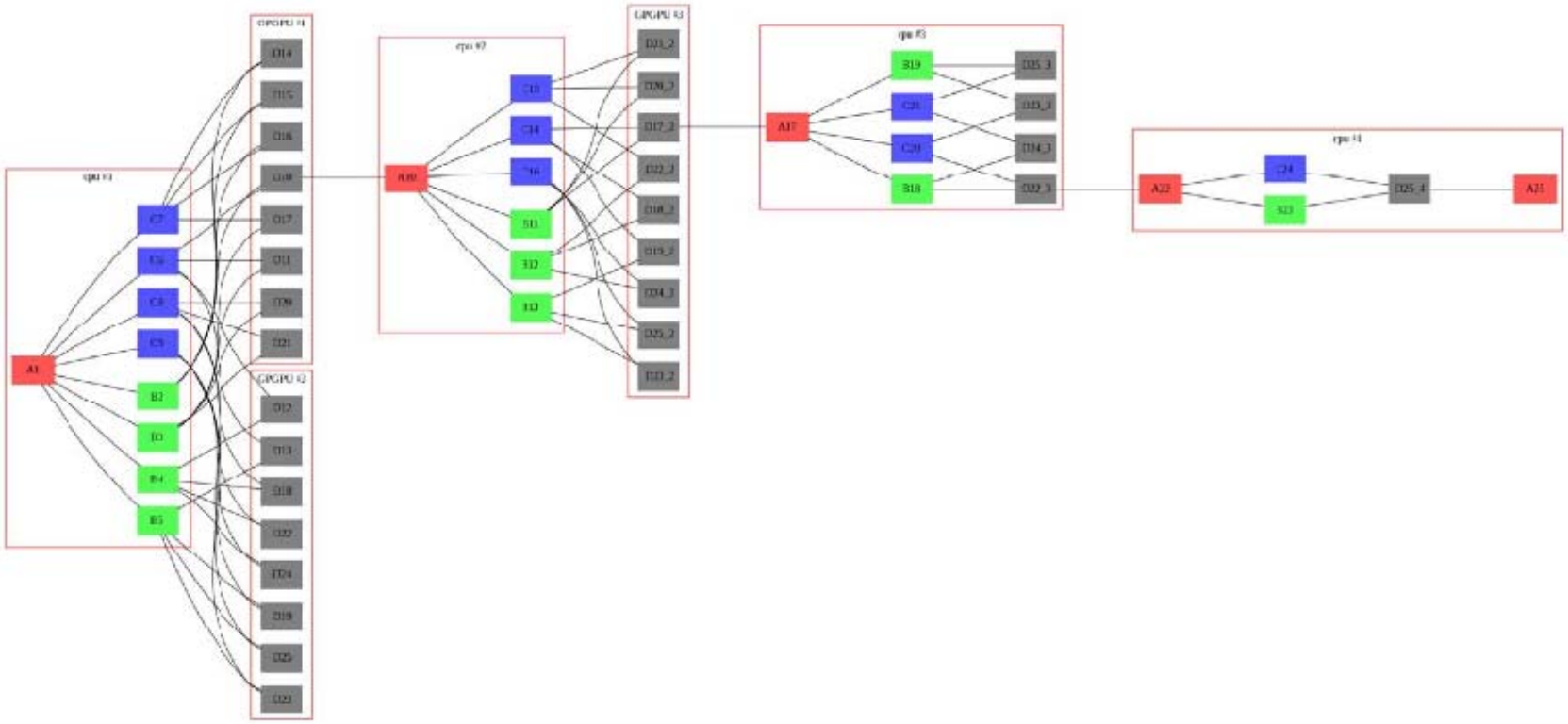
エンドユーザーはビジュアルプログラミングで部品をつなぎあわせる



5x5のパネル分割を指示

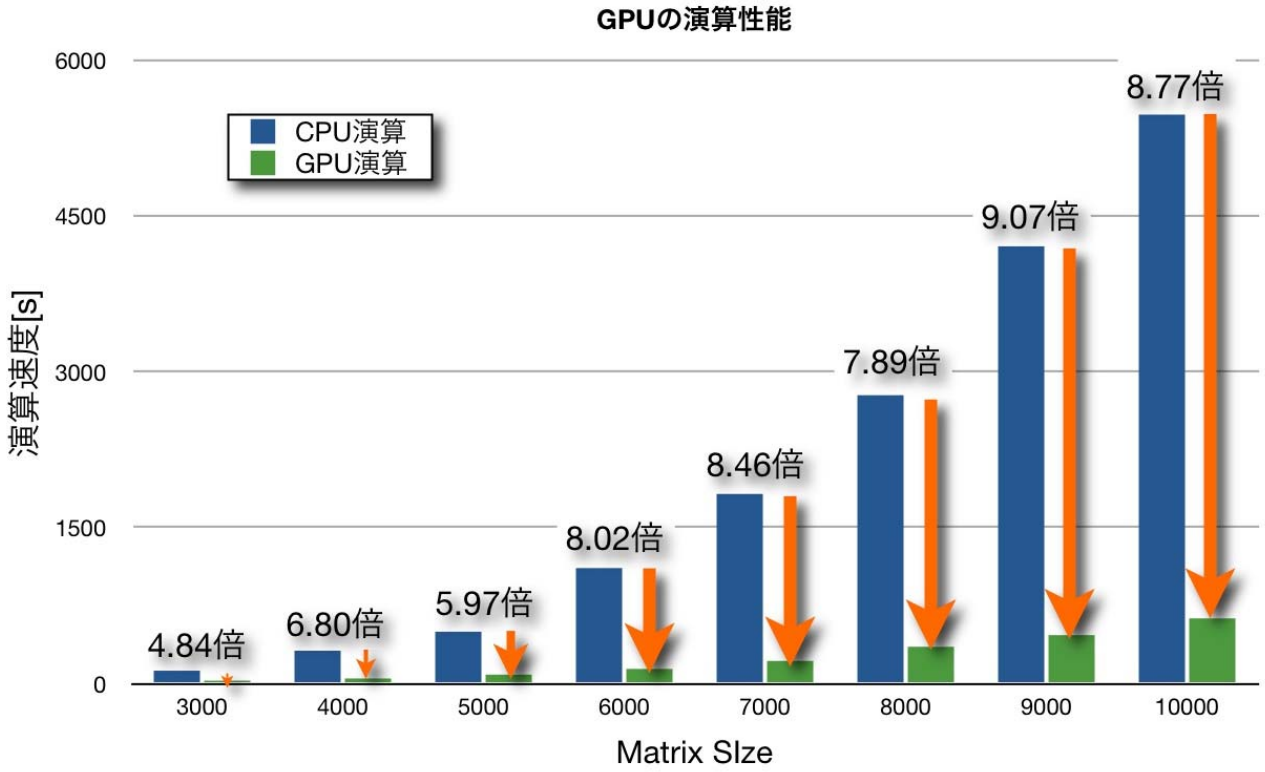
© 2009 IBM Corporation

コンパイル（ハードウェアマッピング）



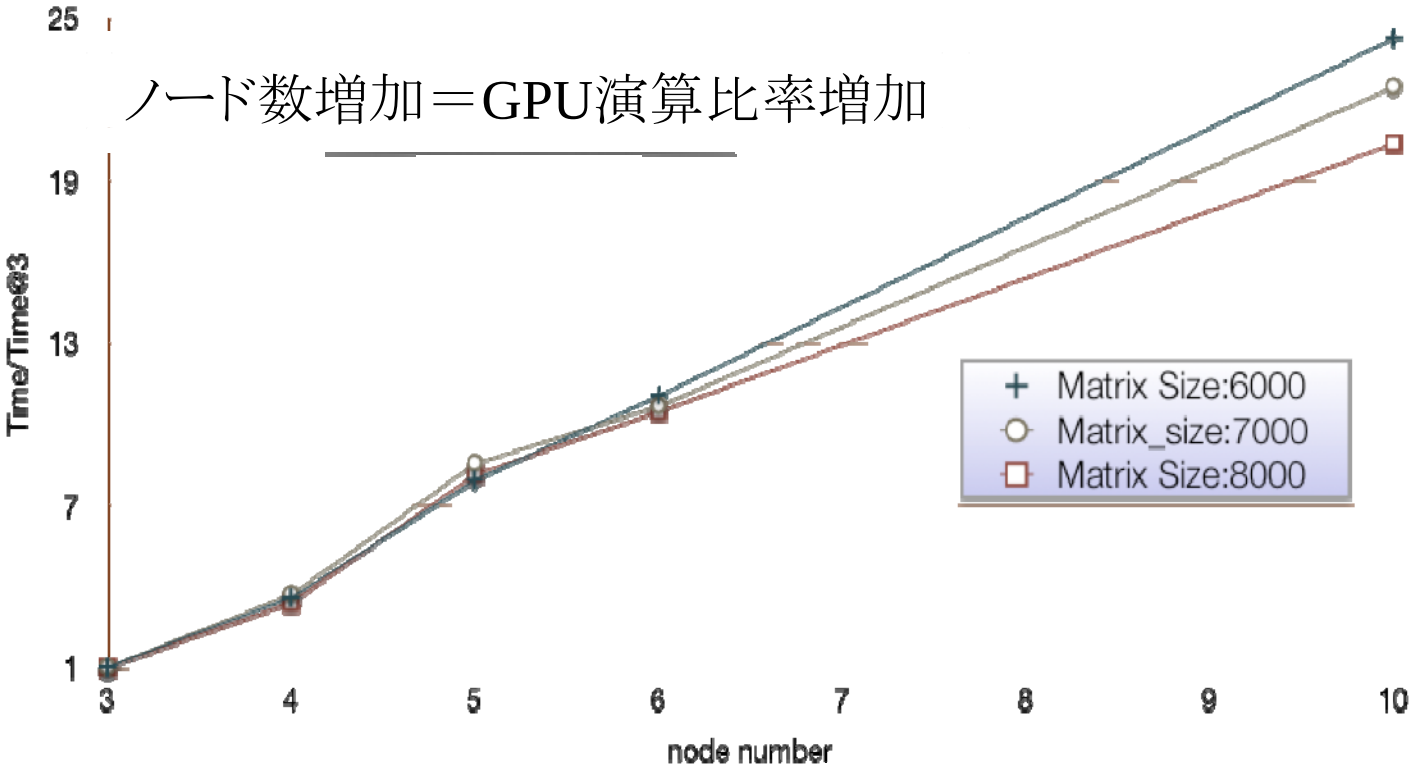
© 2009 IBM Corporation

測定結果（10並列）



All Rights Reserved, Copyright (c) RIKEN 2009-

並列性能



All Rights Reserved, Copyright (c) RIKEN 2009-

RIVERの現状と今後

- 部品が優秀なら、そこそこの性能
- 今後は理研内のアプリに応用してテスト
  - 部品ライブラリーを整備
  - RIVERと部品ライブラリーをフリーフェアとして配布

PCクラスターの歴史は

- 背景と動機
  - 生産台数：PC > WS > Server > Supercomputer
  - 価格あたりの性能：
    PC > WS > Server > Supercomputer
  - 安くて高性能なPCでスパコンを：PCクラスタ

- PCクラスターにアクセラレータを装着して、更にコスト性能を上げるのは自然

→汎用品を使って、用途にあった専用品を作る

RICCでの挑戦

- 千から万に至る並列性能のテスト
- アクセラレータによる性能向上の可能性と応用分野の開拓
  - 利用者の拡大：RIVER開発後はフリーに
  - コスト性能比、電力性能比の追求