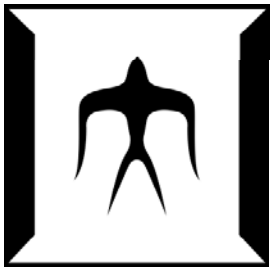
Hundred Million Cores in Commodity--- Why not? (or, will `custom'*finally* prevail?)
or
"The first Exa Machine shall be a Cloud"

Satoshi Matsuoka, Prof., Dr. Sci.

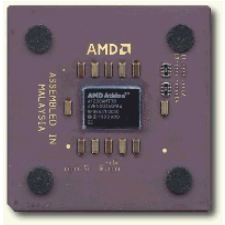
GSIC Center, Tokyo Institute of Technology / The NAREGI Project, National Institute of Informatics



CCGSC 2008 @ NC, USA Sep. 14-17 2008

Rise of the Commodity Clusters: "The Scenario"

High Performance Commodity Computing
-High Performance x86 CPUs
-Fast Commodity Interconnect
-Cluster Software

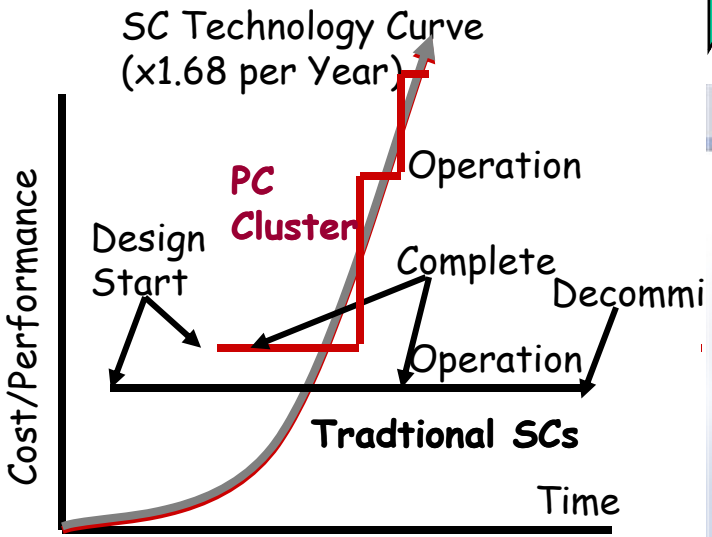


High-Performance x86 CPUs

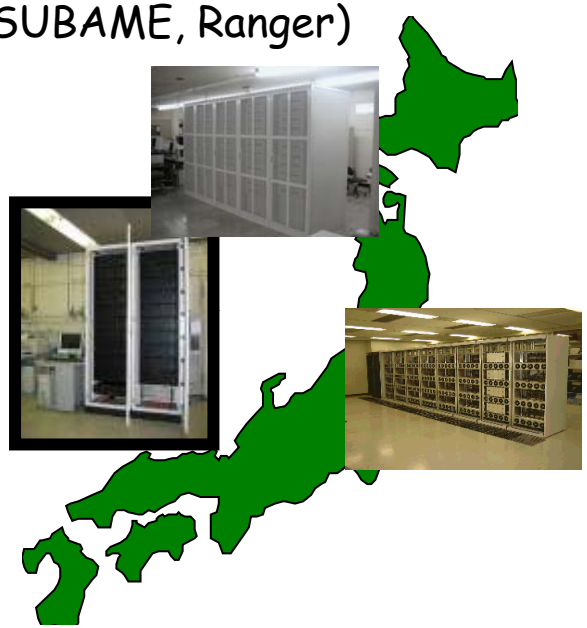
Myrinet, Infiniband, etc.

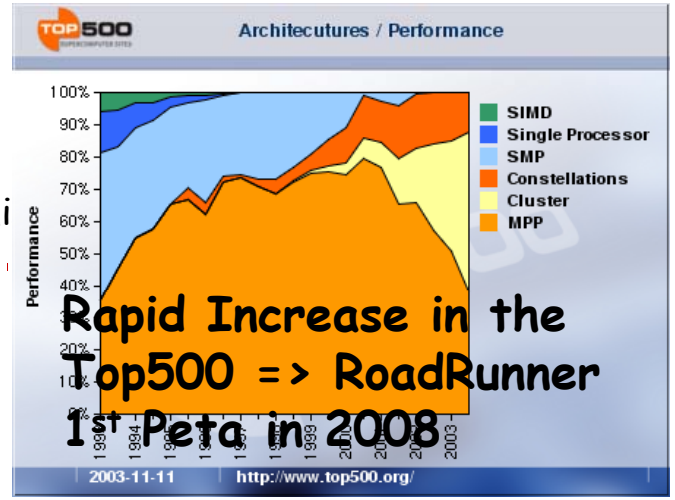
Rise and spread of Commodity Clusters and increase in their size

Real-time tracking of technology curve



Widespread Use of Clusters: Small to very large (e.g. TSUBAME, Ranger)

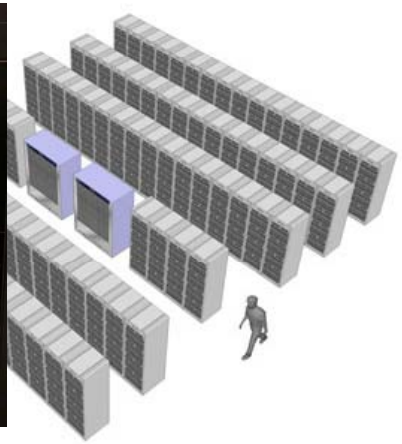




Rapid Increase in the Top500 => RoadRunner 1st Peta in 2008

And this went to Petascale, Despite all the Skepticism

- TACC Ranger
  - The largest x86 Linux Cluster ~50,000 x86 cores
  - Half a Petaflop



- RR: the first #1 "commodity" cluster on Top500
  - The first #1 machine to use IB
  - The first #1 Linux machine
- The first #1 "heterogeneous" SC (Cell and Opteron)

FUJITSU THE POSSIBILITIES ARE INFINITE

“Why HPC Architecture Must be Custom Built in the Exascal Era”

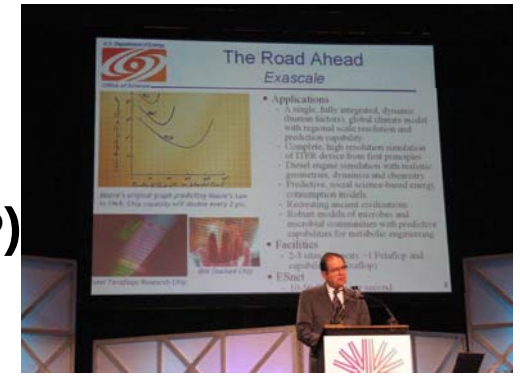
2007-11-28

Slides Coutesy of Hisa Ando
(Former) Senieor Architect
Fujitu Ltd.

(Abridged and Translated by Satoshi Matsuoka)

Exaflop HPC Energy Consumption

- In SC07, Ray Orbach “Exascale by 2016”
- Energy Consumption at 90nm
  - FPU: ～５００ｐJ/DPFOP
    - (GRAPE-DR: 65W/256GFlops=250pJ/DP FOP)
  - General Purpose CPU： ２０ｎJ/Cycle
    - 4FOP/Cycle => x10 power over FPU

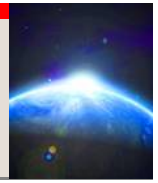


- 1 Exa Flops power requirement
  - Circa 2006-7: 90nm technology: 500pJ x 10^{18} = 500x10^{6} W
  - Circa 2016- (conservatively) suppose 22nm technology
    - Gate capacitance 22/90 = x 0.24, Vcc 0.8V/1V = x 0.8
    - Power $\propto CV^2$ = 0.24 x 0.8^2 = x 0.15
    - 1 ExaFlop FPU array： 0.15 x 500pJ x 10^{18} = 75MW(!)

FUJITSU

THE POSSIBILITIES ARE INFINITE

Why Special Architecture for Exascale?



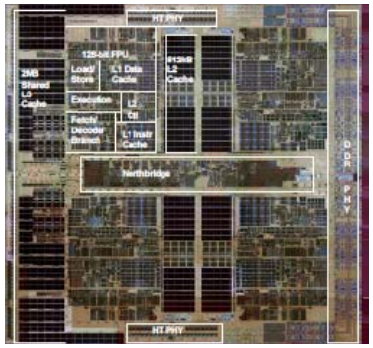
- **Total System Power ~= 1.5GW～2GW (!?)**
  - **Extrapolate to gen. purpose CPU: 75MWx10 = 750MW**
  - **Memory, power delivery loss, cooling, I/O and storage… incur additional x2~x3 overhead**
  - **> $100 million in Utility Bill(!)**
- **Save Power, save power, and save power:**
  - **Objective: 1/30 power reduction**
    - **Energy reduction of FPUs---low power design**
    - **SIMD-parallel control of massive FMA FPUs + Powerful scalar processor---beat Amdahl's law**

Tokyo Electric Co.
Sodegaura PP
3.6GW

- **Claim (by Ando) such a processor cannot be general-purpose （= for Commercial Apps）**
- **I.e., Exascale machine must be (made of) special-purpose HPC architecture**

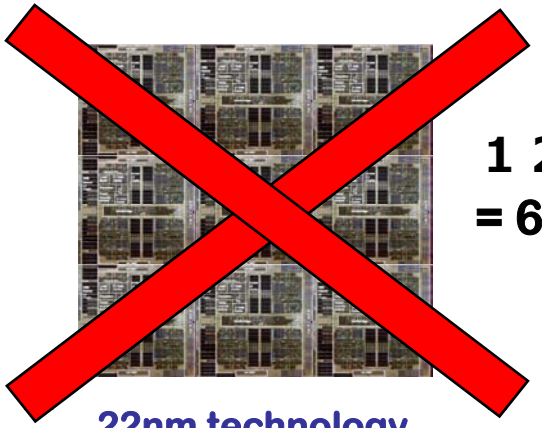
FUJITSU
THE POSSIBILITIES ARE INFINITE

Special Purpose Processor for Exascale circa 2016



65nm technology
AMD 4 Core Opteron
Chip 283mm²
Core 26mm²

Server Processor



1 2 8 Fopx5GHz
= 640GFlops

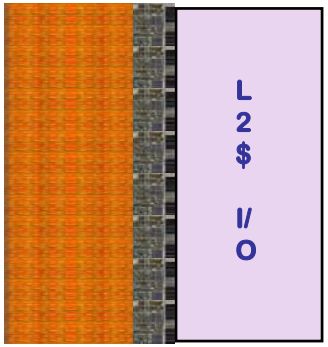
22nm technology
32 Cores
Chip 283mm²
Core 3.5mm²

Grape-DR
90nm technology
512FM+FA

Suppose :
16FMA/mm²

22nm technology
25(FM+FA)s/mm²

Custom HPC processor



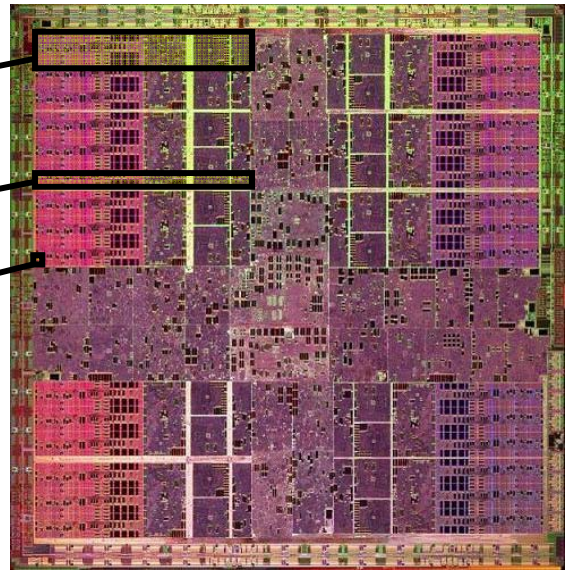
8 Core (28mm²)
+2048FMA (128mm²)
+16MB L2$/LM (35mm²)
+I/O (60mm²)

4096Fopx5GHz
= 20TFlops

22nm technology
Chip ~250mm²

FUJITSU
THE POSSIBILITIES ARE INFINITE

But wait, we now have this in commodity…the GPUs (Tesla, FireStream, Larrabee, ClearSpeed)



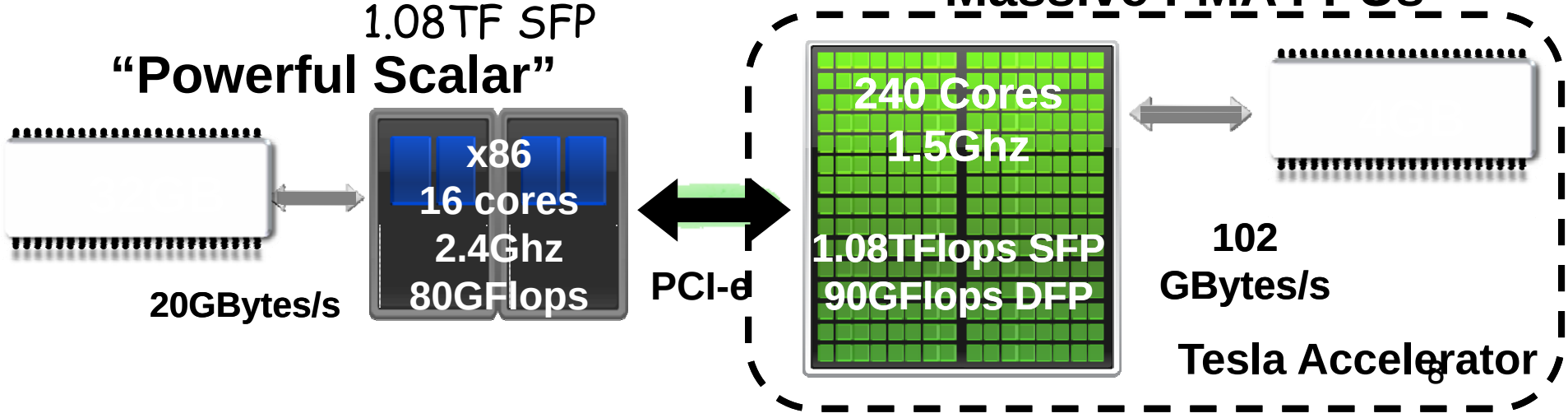
65~55nm(2008)
=> 15 nm (2016)
x20 transitors (30 bil)
20TF FMA SFP
10TF FMA DFP

nVidia Tesla T10: 65nm, 600m2, 1.4 bil Tr

1.08TF SFP

"Powerful Scalar"

"Massive FMA FPUs"



TSUBAME 1.2 Evolution (Oct. 2008)

The first "Petascale" SC in Japan

Voltaire ISR9288 Infiniband x8
10Gbps x2 ~1310+50 Ports
~13.5Terabits/s
(3Tbits bisection)

NEC SX-8i

500GB
48disks

Storage
1.5 Petabyte (Sun x4500 x 60)
0.1Petabyte (NEC iStore)
Lustre FS, NFS, CIF, WebDAV (over IP)
60GB/s aggregate I/O BW

10Gbps+External NW

Unified Infiniband network

10,000 CPU Cores
300,000 SIMD Cores
~900TFlops-SFP,
~170TFlops-DFP
80TB/s Mem BW (1/2 ES)

Sun x4600 (16 Opteron Cores)
32~128 GBytes/Node
10480core/655Nodes
21.4TeraBytes
50.4TeraFlops
OS Linux (SuSE 9, 10)
NAREGI Grid MW

NEW Deploy:
GCOE TSUBASA
Harpertown-Xeon
90Node 720CPU
8.2TeraFlops

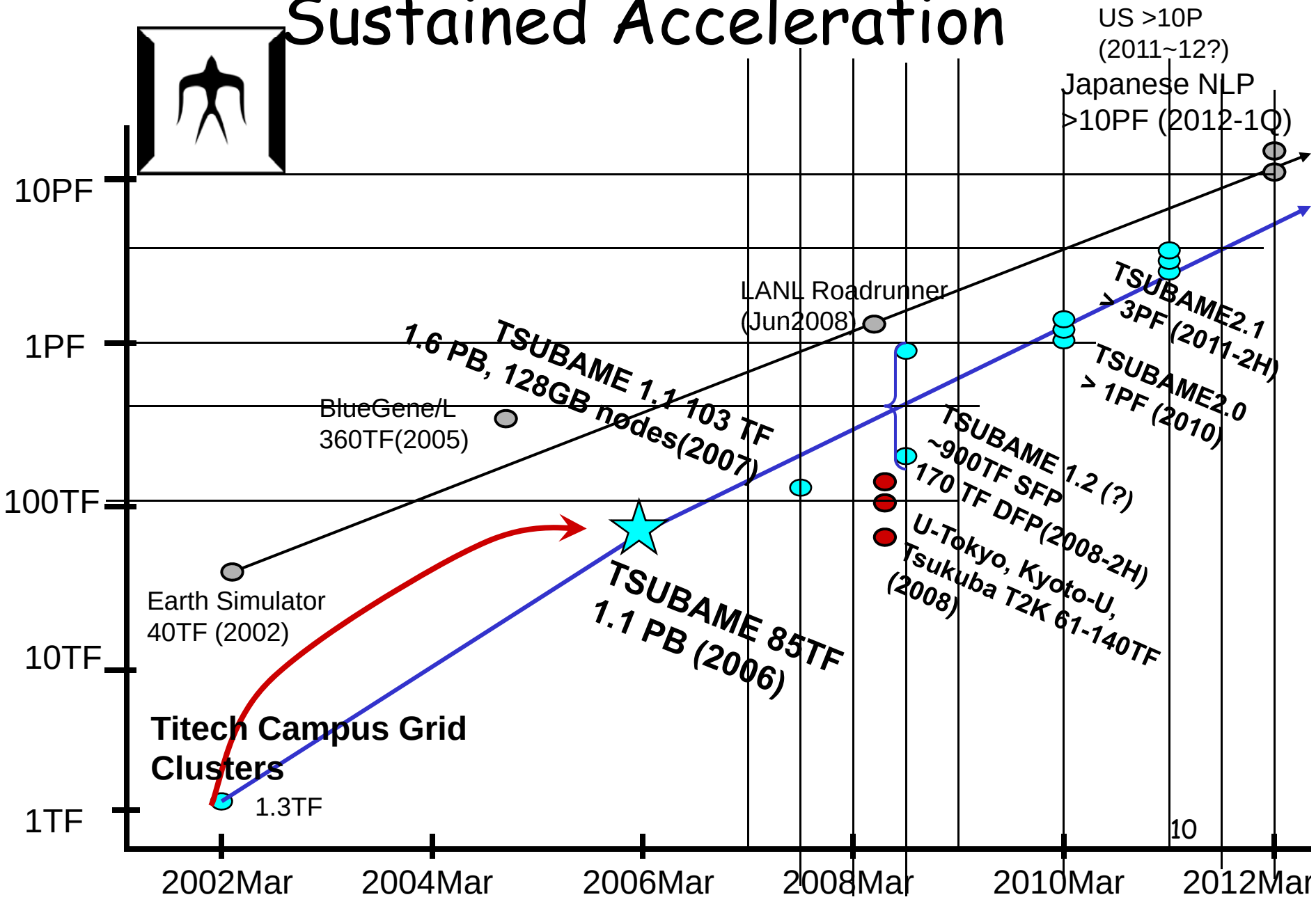
NEW: co-TSUBAME
90Node 720CPU (Low Power)
~7.2TeraFlops

PCI-e

ClearSpeed CSX600
SIMD accelerator
360 648 boards,
35 52.2TeraFlops

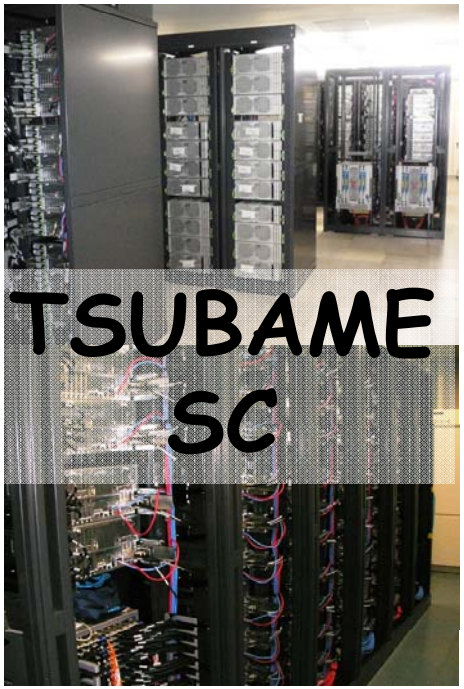
Nvidia Tesla T10P-one card per node, ~680 cards
High Performance in Many BW-Intensive Apps
10% power increase over TSUBAME 1.0 (130TF SFP / 80TF DFP)

TSUBAME Upgrades Towards Petaflops Sustained Acceleration

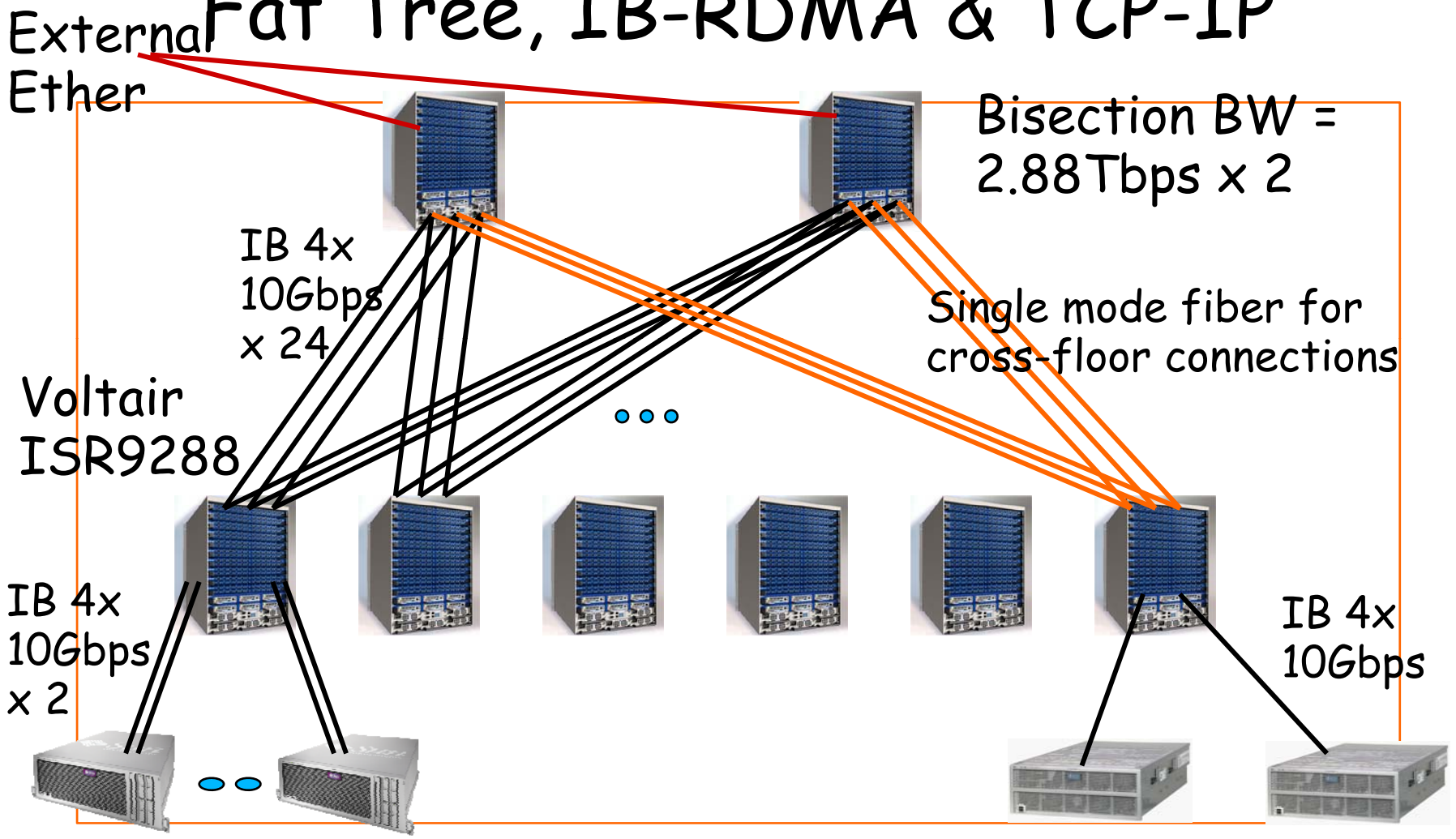


IDC Servers and Cluster SC--- the differences (or are there?)

- The Same---the nodes
  - Processors (x86)
  - Memory (DDR DRAM)
  - I/O (PCI-e)
  - OS (Linux/Windows), MW
- Differences
  - Network (IB vs GbE) (< 10% of machine cost)
  - Parallel Storage
  - Power Density
  - Parallel SW Stack: (MPI, OpenMP, BQ, …)
  - Operations as a SC
- Accelerators?



TSUBAME Network: ~1400 port Fat Tree, IB-RDMA & TCP-IP



X4600 x 120nodes (240 ports) per switch
=> 600 + 55 nodes, 1310 ports, 13.5Tbps

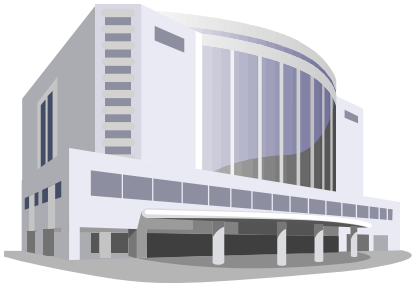
X4500 x 60nodes (60 ports)
=>60ports 600Gbps

A Cloud Exaflops Machine

- Suppose 100MW power capacity, 1TFlop / 100W
- Hmm, this is easy, as my Nvidia GTX280 or AMD Radeon 4870 is ~200-250W incl. system overhead in SFP circa 2008…
- Need 10x => Expect 1TF DFP @ 100W system possible in 2012~13… (22nm)
- Network? Approx 10%-25% of system power and cost, so this is not a problem
- Rest are software and operational issues
- If Cloud centers can charge x2 standard fee, then it is a good business case

Mar 2012… the Japanese "> 10 PF" SC to be Online

25~30MW Power
~1mil ft2 floorspace
$1 bil construction



Kobe, Japan
"the site"

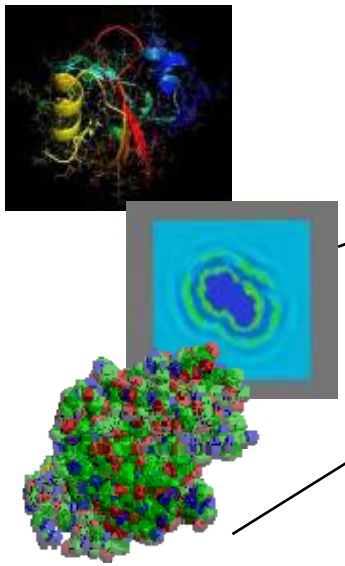
In fact we can build a Exaflop Cloud SC in 2012(!)

- @Tokyo---One of the Largest IDC in the World (in Toyosu, Tokyo... Built in 2003)
- Can fit a 10PF easy, 1 Exaflop in 2012
- On top of a 55KV/6GW Substation
- 150m diameter, 1,400,000 ft2 IDC floorspace
- 70+70 MW = 140MW power
- Can fit both Google/MS IDC or Any DOE center
- Remember interconnect cost 20% at most
- And can run Linux, Cloud/Grid interfaces, and HPC languages for accelerated & hybrid programming
- **Merger of "SC Centers" & "Cloud"**

Overview of HPC-GPGPU Project

(work w/MS Research TCI)

Advanced Bioinformatics/ Proteomics



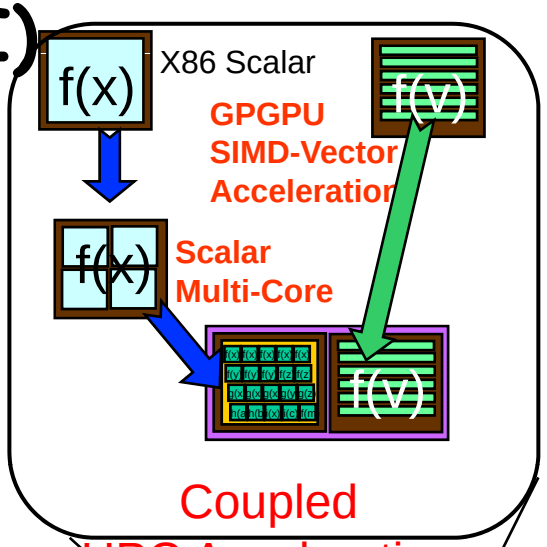
Need x1000 acceleration over standard PCs

Research Focus

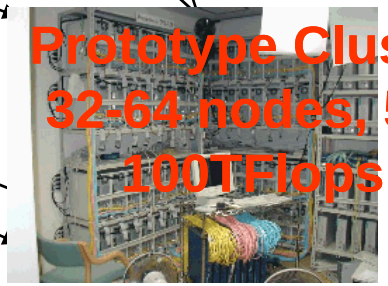
Bioinformatics Acceleration
e.g., 3-D All-to-All Protein Docking

GPGPU-CPU Hybrid Massively Parallel "Adaptive" Solvers + GPGPU FFT and other Acceleration Kernels

- Improving GPGPU Programmability w/ Library/Languages e.g. MS Accelerator
- High Dependability w/ large-scale GPGPU Cluster
- Model-based GPGPU-CPU Load Balancing



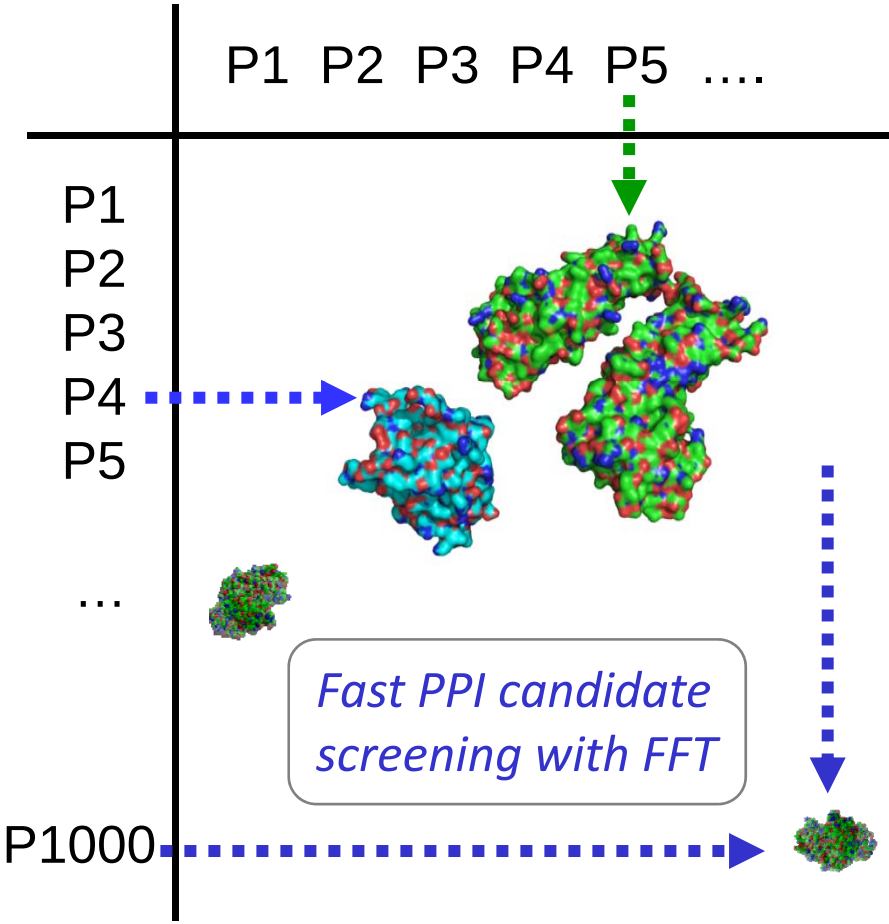
Coupled HPC Acceleration



Prototype Cluster 32-64 nodes, 50-100TFlops

Towards Next Gen Petascale Personal Clusters and Desksides

All-to-all 3-D Protein Docking Challenge



1,000 x1,000 all-to-all docking fitness evaluation will take only

1-2 months (15 deg. pitch)

with a 32-node HPC-GPGPU cluster (128 GPGPUs).

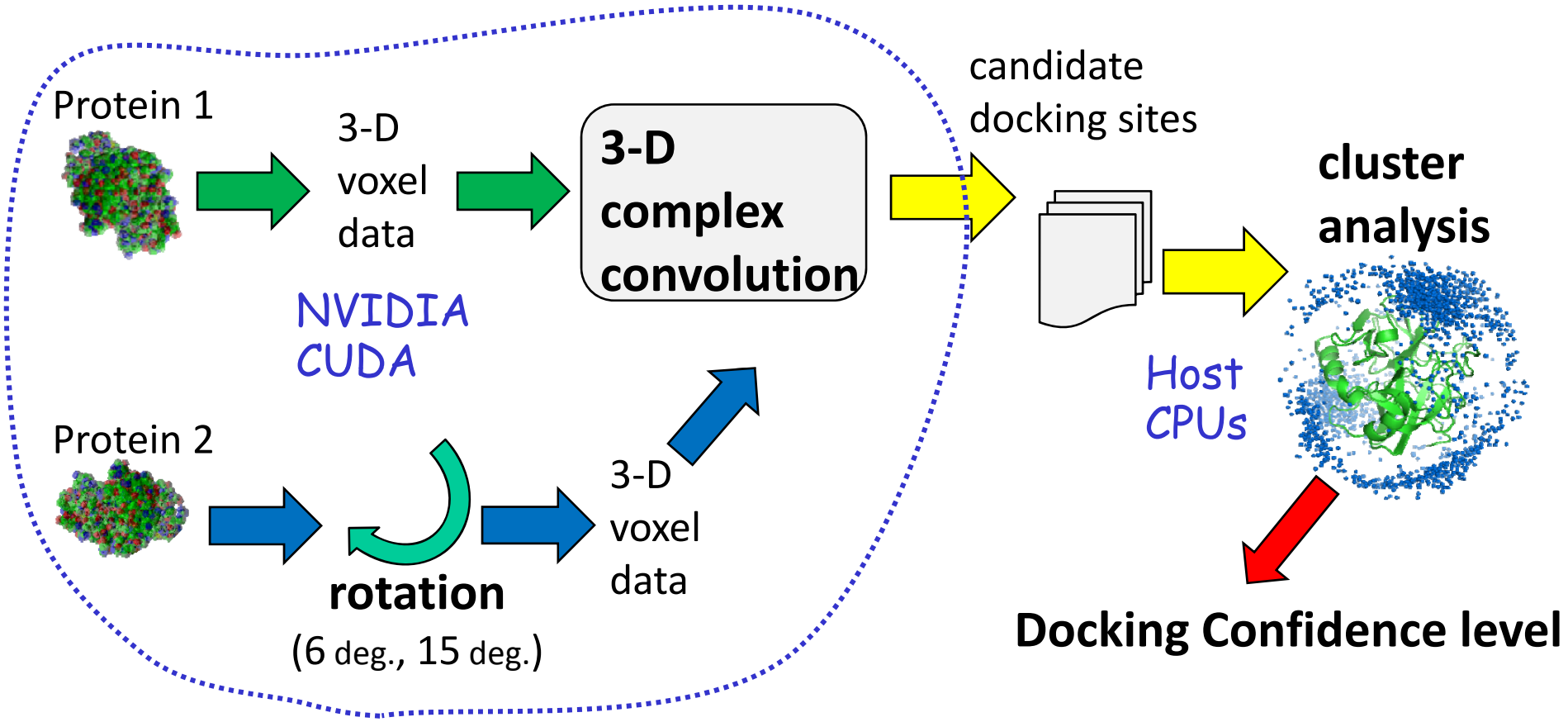
cf.

~ 500 years with single CPU (sus. 1+GF)

~ 2 years with 1-rack BlueGene/L

Blue Protein system
CBRC, AIST
(4 rack, 8192 nodes)

Calculation Flow for 3-D AA docking



Calculation for **a single protein-protein pair**: **~= 200 Tera ops.**

3-D complex convolution $O(N^3 \log N)$, *typically* $N = 256$

x

Possible rotations $R = 54{,}000$ *(6 deg. pitch)*

200 Exa Ops for 1000 x 1000[18]

High Performance 3-D FFT on NVIDIA CUDA GPUs [SC08 paper preview]

Akira Nukada

Tokyo Institute of Technology, GSIC.

Bandwidth Intensive Approach

Our 3-D FFT algorithm consists of the following two algorithms

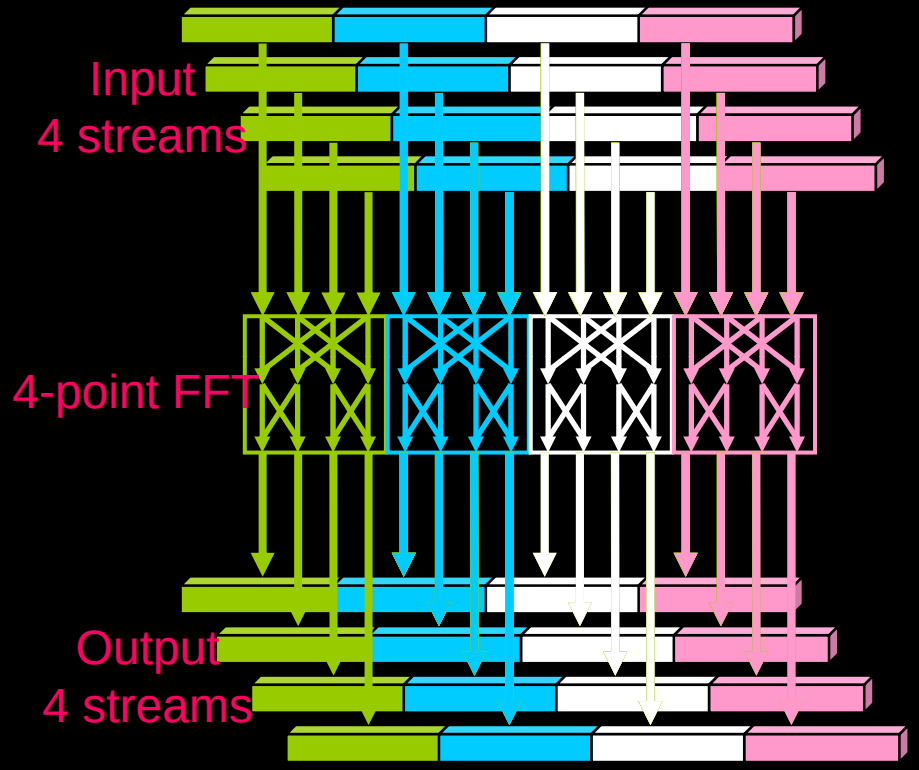
to maximize the memory bandwidth.

(1) optimized 1-D FFTs for dimension X,

(2) *multi-row FFT* for dimension Y & Z.

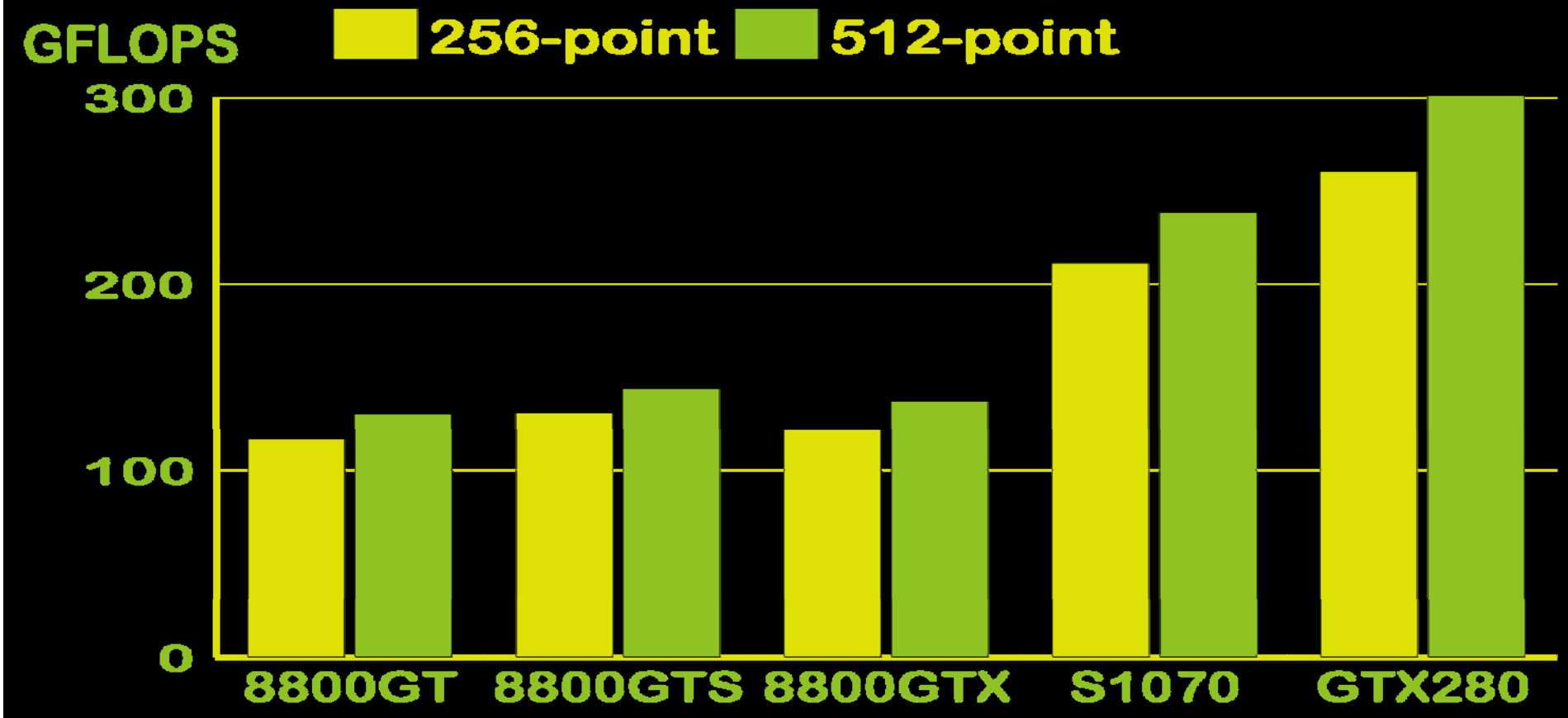
The multi-row FFT computes multiple 1-D FFTs simultaneously.

Adapted from vector algorithms, assuming high memory bandwidth.



This algorithm accesses multiple streams, but each of them is successive.
Since each thread compute independent set of small FFT, thousands of registers are required
Solution: for 256-point FFT, use two- pass 16-point FFT kernels.

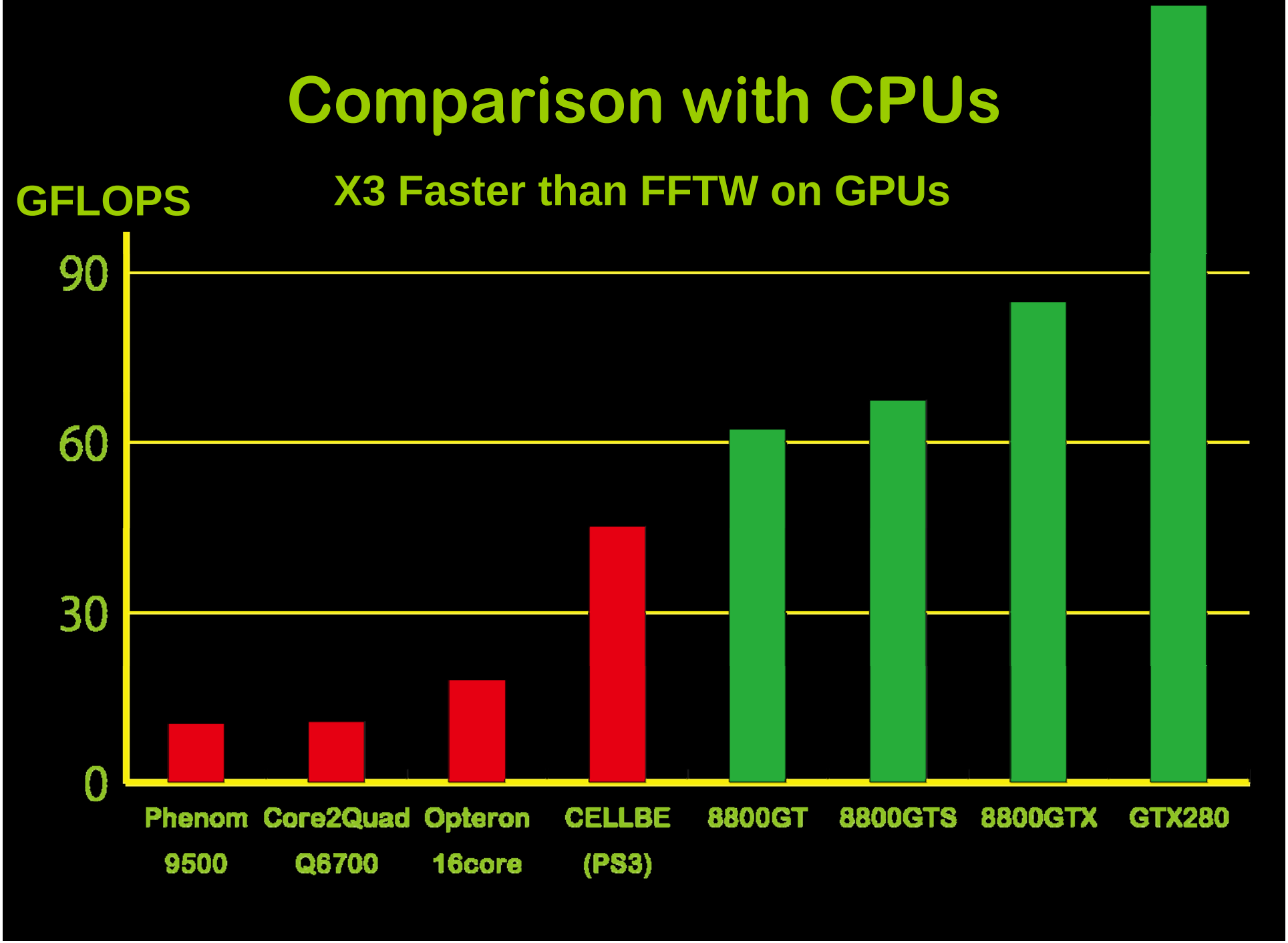
Performance of 1-D FFT



Note: An earlier sample with 1.3GHz is used for Tesla S1070.

Comparison with CPUs

X3 Faster than FFTW on GPUs



Heavily Acclerated Prototype Cluster System Configuration

- 32 compute nodes
- 128 8800GTS GPGPUs
- one head node.
- Gigabit Ethernet network
- Three 40U rack cabinets.
- Windows Compute Cluster Server 2003 SP1, planned 2008 migration
- Visual Studio 2005 SP1
- nVidia CUDA 2.x

Performance Estimation for 3D PPD

Single Node

	Power (W)	Peak (GFLOPS)	3D-FFT (GFLOPS)	Docking (GFLOPS)	Nodes per 40 U rack
Blue Gene/L	20	5.6	-	1.8	1024
TSUBAME	1000 (est.)	76.8 (DP)	18.8 (DP)	26.7 (DP)	10
8800 GTS *4	570	1664	256	207	10~13

System Total ! Only CPUs for TSUBAME. DP=double precision.

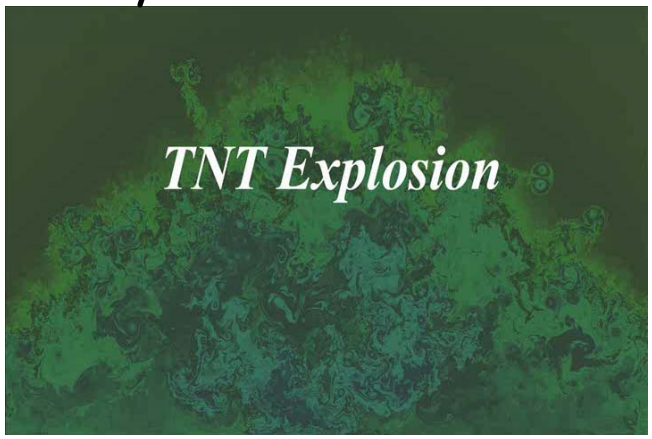
	# of nodes	Power (kW)	Peak (TFLOPS)	Docking (TFLOPS)	MFLOPS/W
Blue Gene/L (Blue Protein @ AIST)	4096 (4racks)	80	22.9	7.0	87.5
TSUBAME	655 (~70 racks)	~700	50.3 (DP)	17.5 (DP)	25
8800 GTS	32 (3racks)	18	53.2	6.5	361

Can compute 1000x1000 in 1 month (15 deg.) or 1 year (6 deg.)

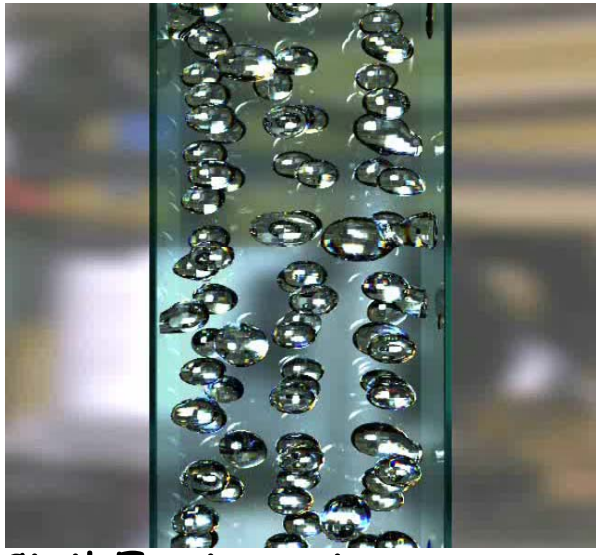
CFD on GPUs

(Material from Prof. Takayuki Aoki, Tokyo Tech.)

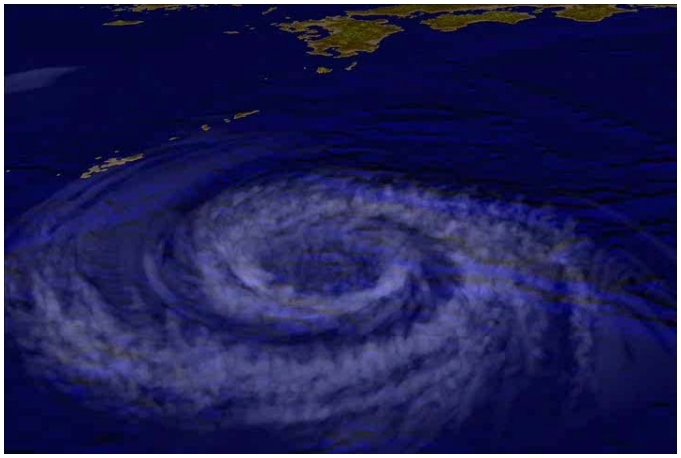
Safety



Nuclear (Cooling)



Weather/Environmental



Civil Engineering

Animations Courtesy Prof. Takayuki Aoki @ Tokyo Tech.

Riken Himeno Benchmark

(Prof. Takayuki Aoki, Tokyo Tech)

RIKEN Himeno CFD Benchmark

Poisson Equation:
(Generalized coordinate)

$$\nabla \cdot (\nabla p) = \rho$$

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} + \alpha \frac{\partial^2 p}{\partial xy} + \beta \frac{\partial^2 p}{\partial xz} + \gamma \frac{\partial^2 p}{\partial yz} = \rho$$

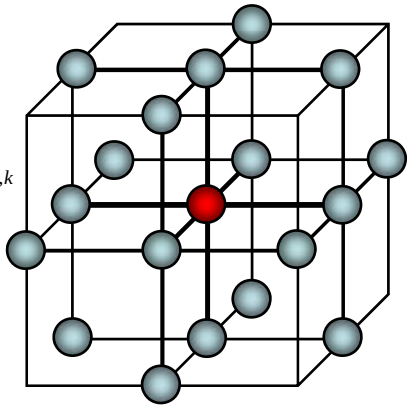
Discretized Form:

$$\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{\Delta x^2} + \frac{p_{i,j+1,k} - 2p_{i,j,k} + p_{i,j-1,k}}{\Delta y^2} + \frac{p_{i,j,k+1} - 2p_{i,j,k} + p_{i,j,k-1}}{\Delta y^2}$$

$$+ \alpha \frac{p_{i+1,j+1,k} - p_{i-1,j+1,k} - p_{i+1,j-1,k} + p_{i-1,j-1,k}}{4\Delta x \Delta y}$$

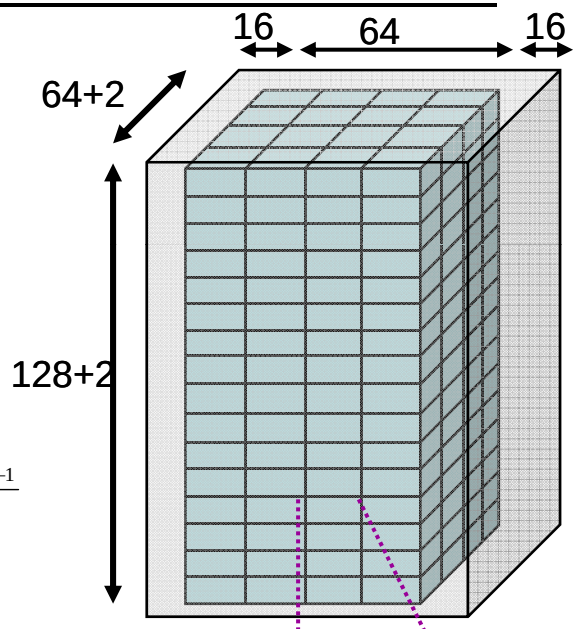
$$+ \beta \frac{p_{i+1,j,k+1} - p_{i-1,j,k+1} - p_{i+1,j-1,k} + p_{i-1,j,k-1}}{4\Delta x \Delta z}$$

$$+ \gamma \frac{p_{i,j+1,k+1} - p_{i,j+1,k-1} - p_{i,j-1,k-1} + p_{i,j-1,k-1}}{4\Delta y \Delta z} = \rho_{i,j,k}$$



18 neighbor point access

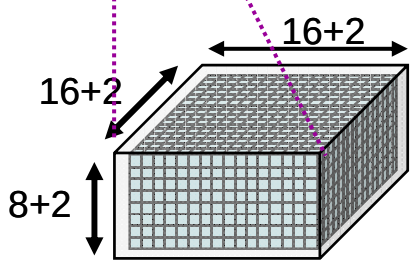
Himeno for CUDA



1 block = 16x16x8 compute region

Block has 256 thread

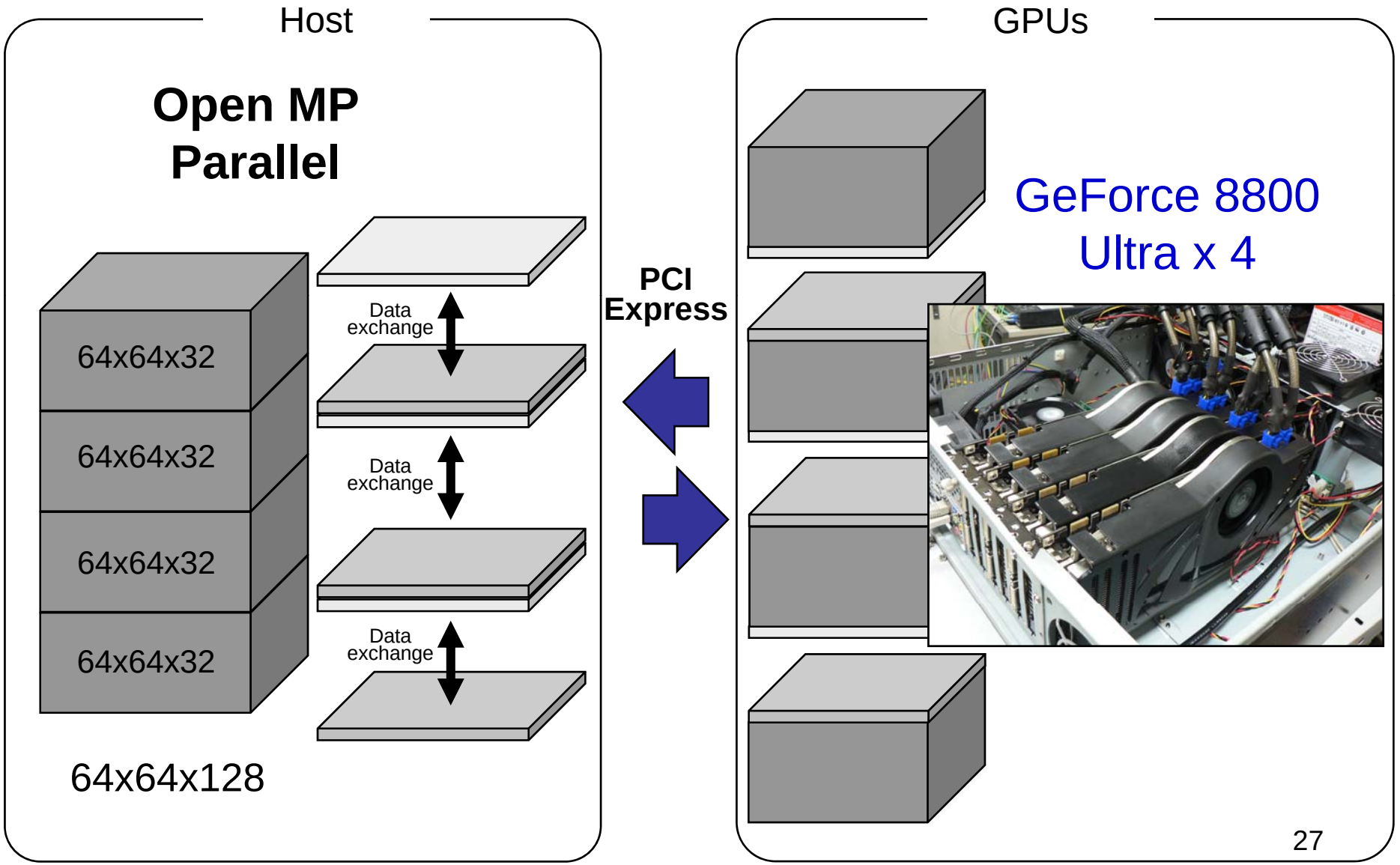
Total 256 blocks = 65536 threads



Block shared mem =16kB

Boundary region used for transfer

4 GPU node parallelization



Parallel Performance

S Model [65x65x129]

1 GPU (no data transfer) 30.6 GFLOPS (0.269sec)

2 GPU (16kB transfer) 42.5 GFLOPS (0.193sec)

4 GPU (32kB transfer) 51.9 GFLOPS (0.158sec)

x53.1 acceleration

0.976 GFLOPS (8.431sec)

Reference

M Model [129x129x257]

1 GPU (no data transfer)	29.4 GFLOPS	(2.328sec)
2 GPU (66kB transfer)	53.7 GFLOPS	(1.275sec)
4 GPU (131kB transfer)	83.6 GFLOPS	(0.819sec)

L Model [257x257x512]

1 GPU (no data transfer)	…….	
2 GPU (262kB transfer)	…….	
4 GPU (524kB transfer)	**93.6 GFLOPS**	(5.974sec)

C.f. NEC SX-8 6 CPU (96GF Peak) 38.3GFLOPS Size XL


RIKEN BMT 賞

1PC 部門

青木 尊之 様
東京工業大学 学術国際情報センター

小川 慧 様
東京工業大学大学院

Original Code: 8.431 秒
Tuning Code : 0.158 秒
スピードアップ : 53.1 倍

システム名称 : Βακυσοκυ-SPE
CPU: AMD Phenom 9600 (2.3GHz), 4GB DDR2-800
GPU: MSI NX8800ULTRA-T2D768E-HD-OC 2枚
ELSA GD988-768ERU 2枚
768MB GDDR3-2300(MSI), 768MB GDDR3-2160(ELSA)
OS: CentOS 5.1
Compiler: CUDA SDK1.1, gcc4.1.2

第五回 RIKEN BMT コンテスト
2008年3月13日

独立行政法人
理化学研究所 情報基盤センター
姫野 龍太郎

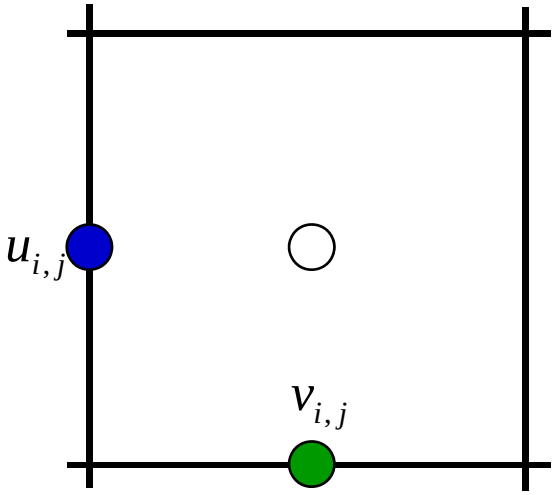
Two-dimensional Burgers Equation

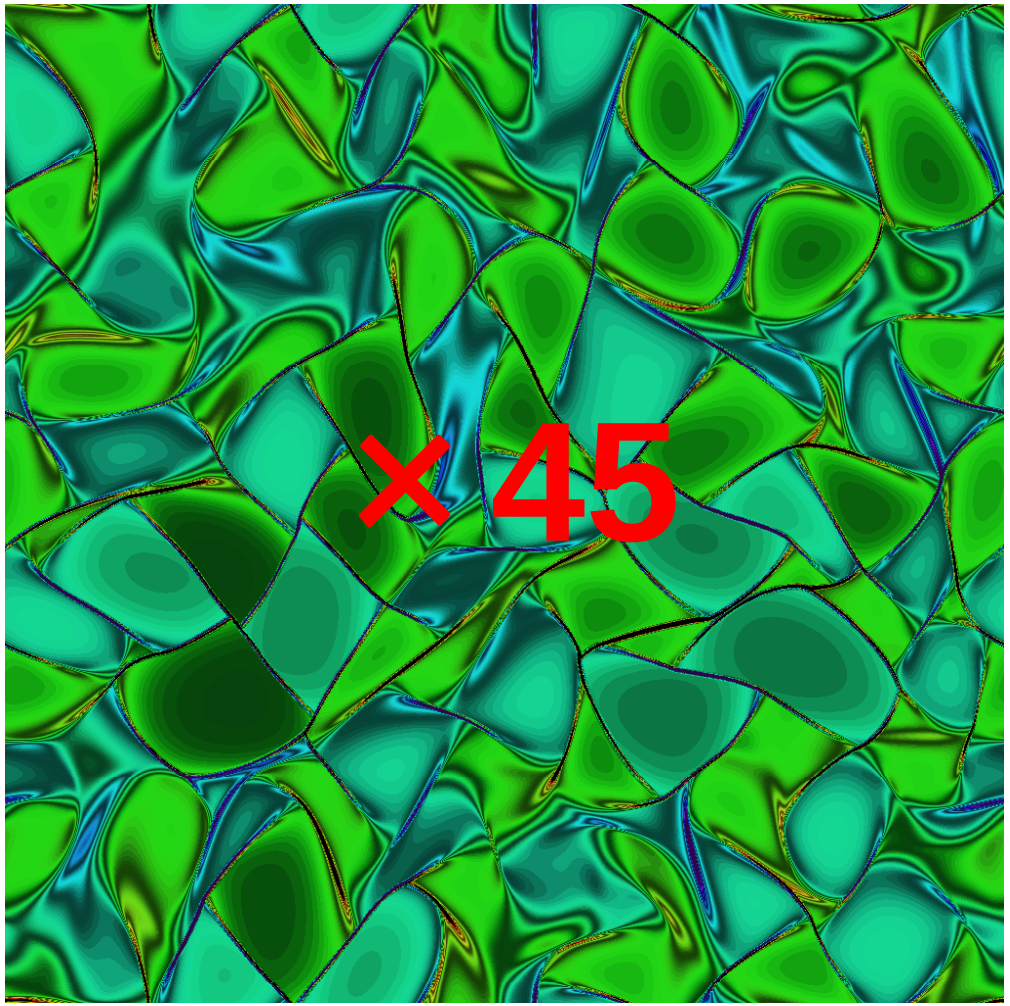
$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\kappa\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=\kappa\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$$

GeForce 8800 GTS

40 GFLOPS





1024×1024

velocity u at the v-*point* $u_s=\dfrac{u_{i,j}+u_{i+1,j}+u_{i,j-1}+u_{i+1,j-1}}{4}$

Homogeneous Isotopic Turbulence

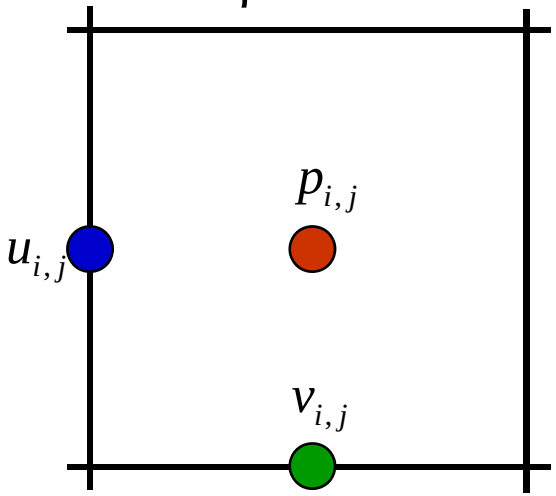
Burgers equation

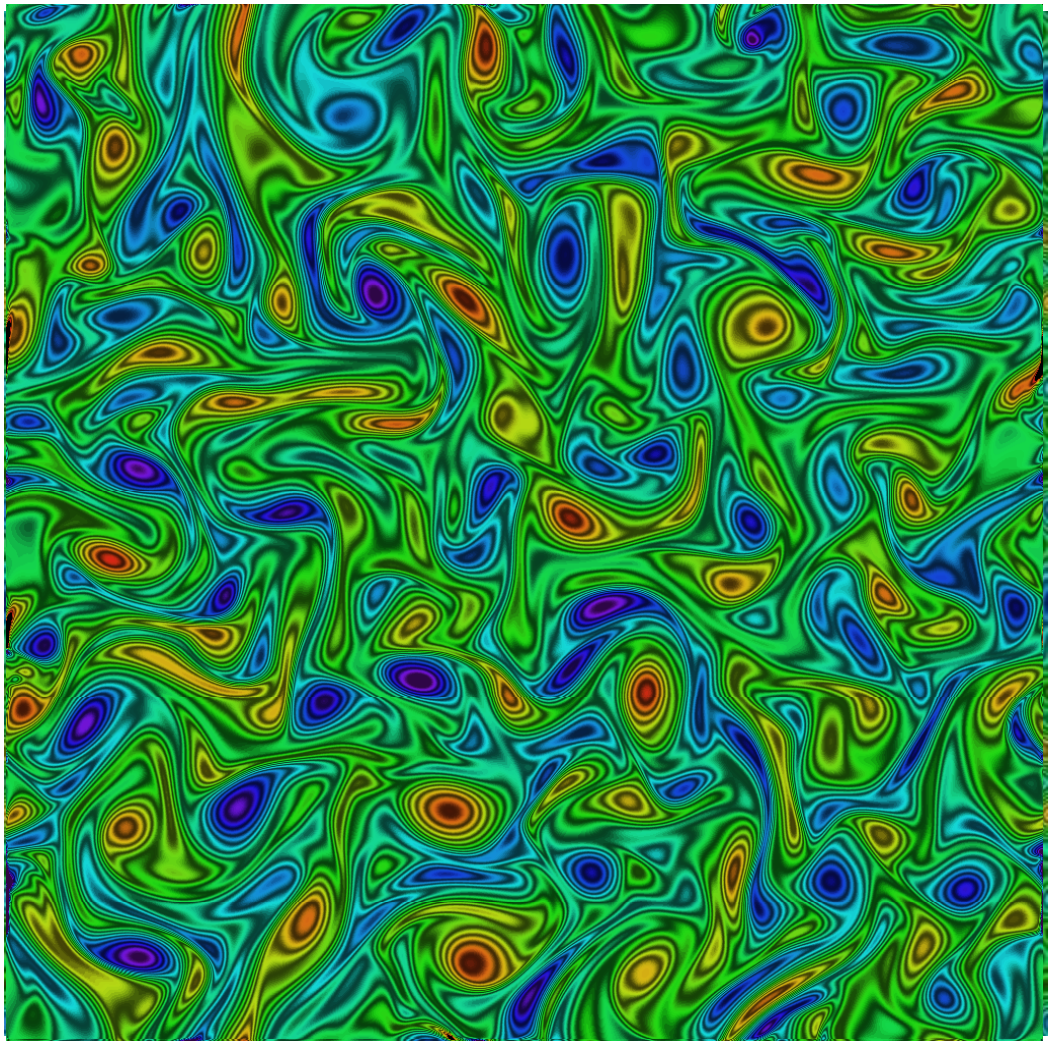
Poisson equation

$$\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}=\frac{1}{\Delta t}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)$$

Correction

$$\frac{\partial u}{\partial t}=-\frac{1}{\rho}\frac{\partial p}{\partial x} \qquad \frac{\partial v}{\partial t}=-\frac{1}{\rho}\frac{\partial p}{\partial y}$$





1024 × 1024

Two-Stream Instability in Plasma Physics

Vlasov-Poisson Equation:

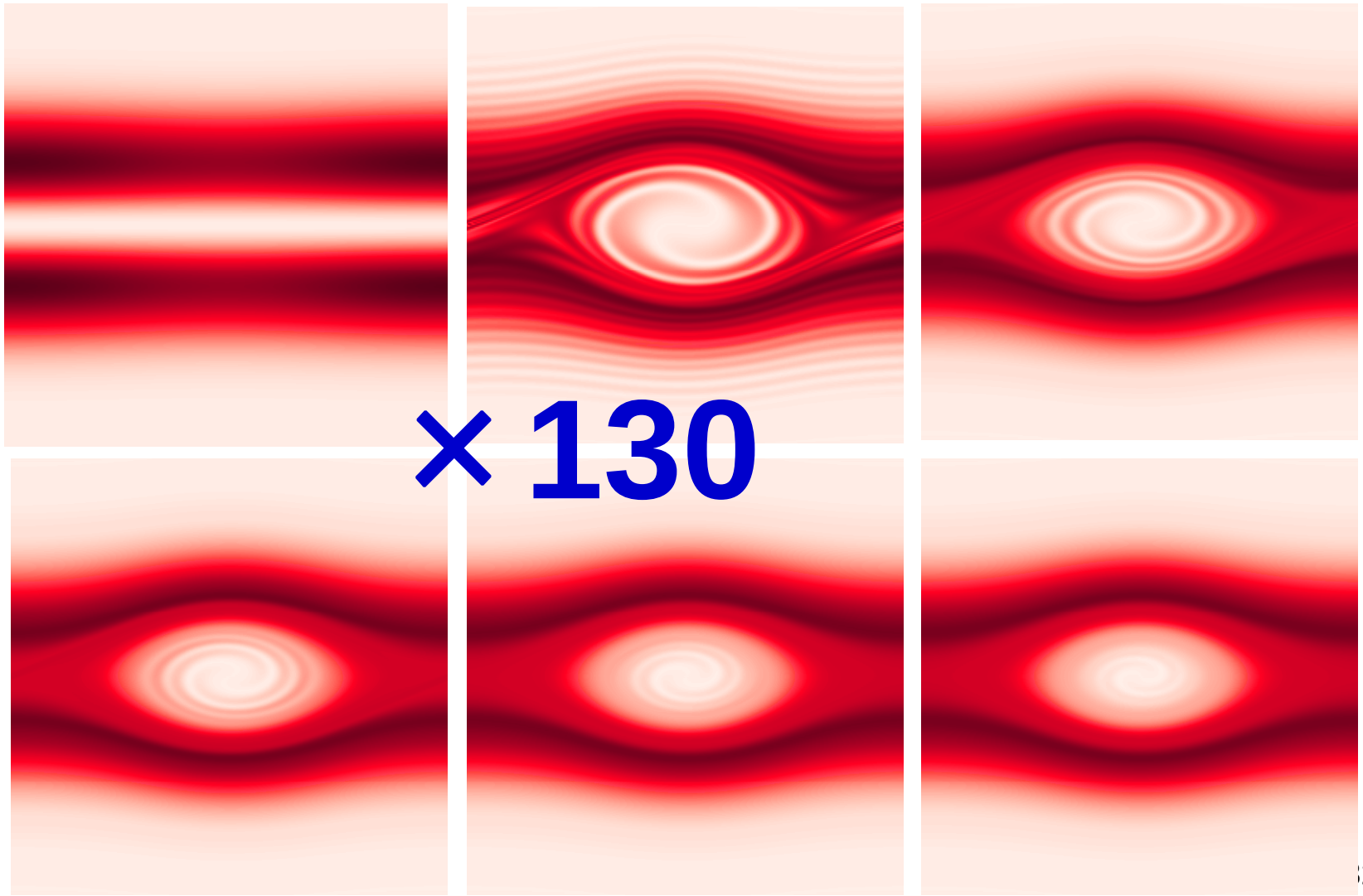
$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \frac{eE}{m_e}\frac{\partial f}{\partial v} = 0 \qquad \frac{\partial^2 \phi}{\partial x^2} = \frac{e(n_e - n_i)}{\varepsilon_0}$$

$$\left(E = -\frac{\partial \phi}{\partial x}, \quad n_e = \int f dv \right)$$

f **: electron distribution function**

n **: electron number density**

120 GFLOPS using 8800GTS



Rayleigh-Taylor Instability

Heavy fluid lays on light fluid and unstable.

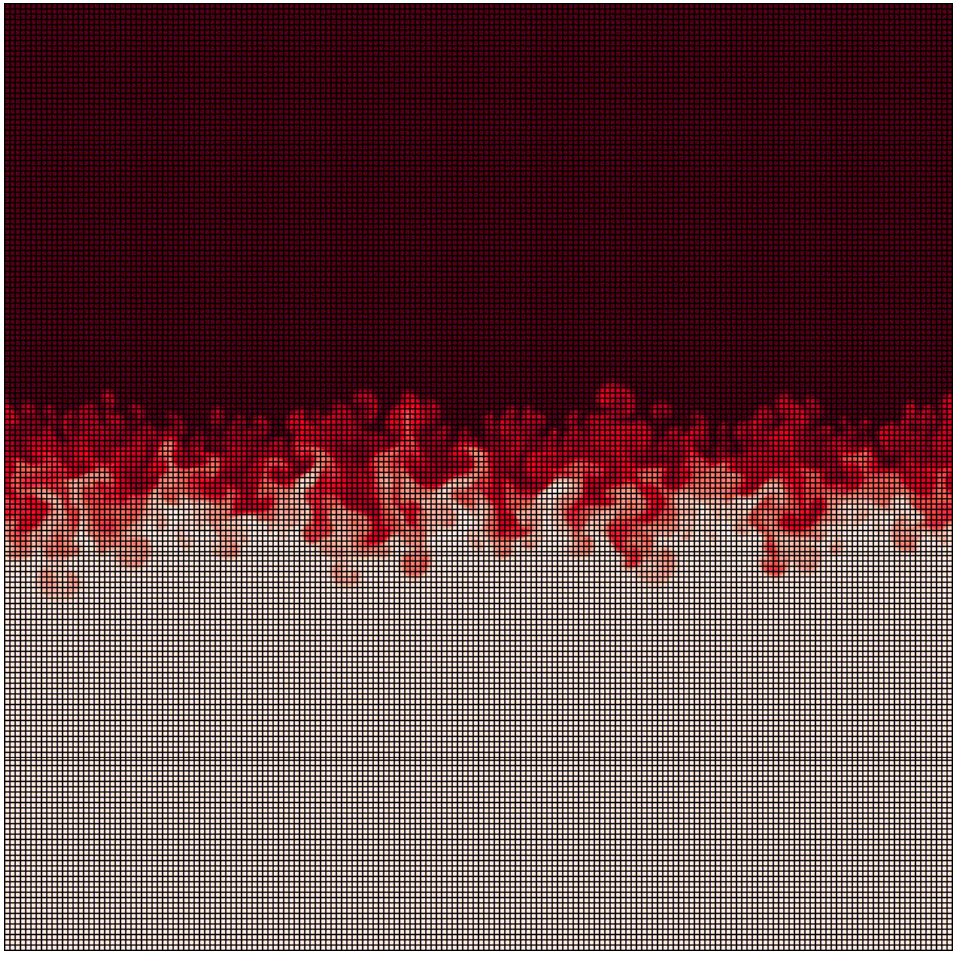
×90

512 x 512

Euler equation:

$$\frac{\partial \boldsymbol{Q}}{\partial t}+\frac{\partial \boldsymbol{E}}{\partial x}+\frac{\partial \boldsymbol{F}}{\partial y}=0$$

$$\boldsymbol{Q}=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ e\end{bmatrix}\quad \boldsymbol{E}=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ eu+pu\end{bmatrix}\quad \boldsymbol{F}=\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ ev+pv\end{bmatrix}$$



88 GFLOPS using GTX280

Phase Separation

Phase transition dynamics is described by the Phase Field Model.

Cahn-Hilliard equation:

$$\frac{\partial \psi}{\partial t} = L\nabla^2\left(\frac{\partial H}{\partial \psi} - C\nabla^2\psi\right)$$

H : free energy

$$\frac{\partial H}{\partial \psi} = \tau\psi - u\psi^3$$

Discretization:

$$\frac{\partial^4 \psi}{\partial x^4} = \frac{\psi_{i+2,j} - 4\psi_{i+1,j} + 6\psi_{i,j} - 4\psi_{i-1,j} + \psi_{i-2,j}}{\Delta x^4}$$

$$\frac{\partial^4 \psi}{\partial x^2 \partial y^2} = \left(\psi_{i+1,j+1} - 2\psi_{i,j+1} + \psi_{i-1,j+1} \right.$$

$$- 2\psi_{i+1,j} + 4\psi_{i,j} - 2\psi_{i-1,j}$$

$$\left. + \psi_{i+1,j-1} - 2\psi_{i,j-1} + \psi_{i-1,j-1} \right)$$

3-D Computation of Phase Separation

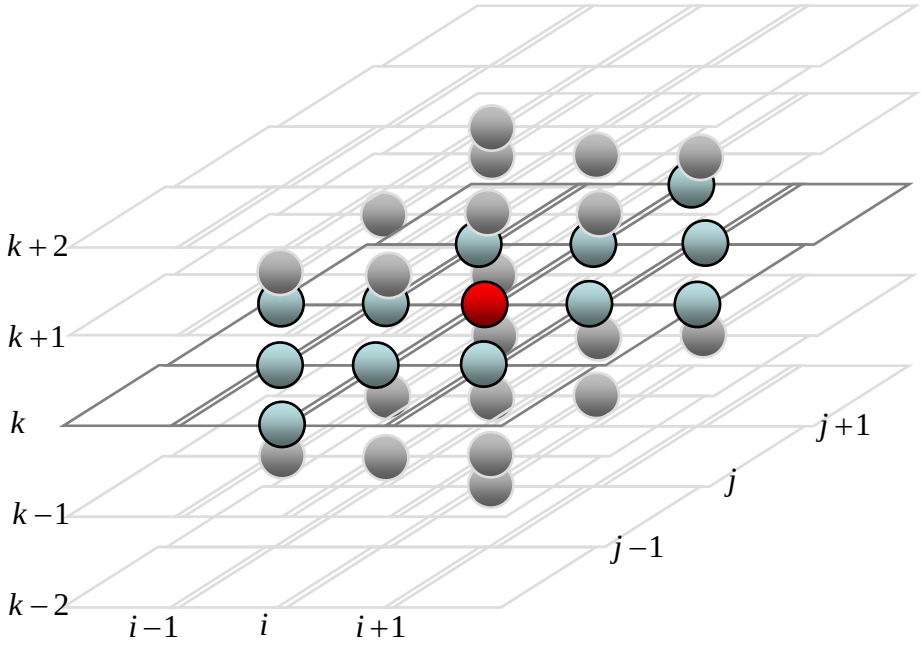
Mixture of Oil and Water:

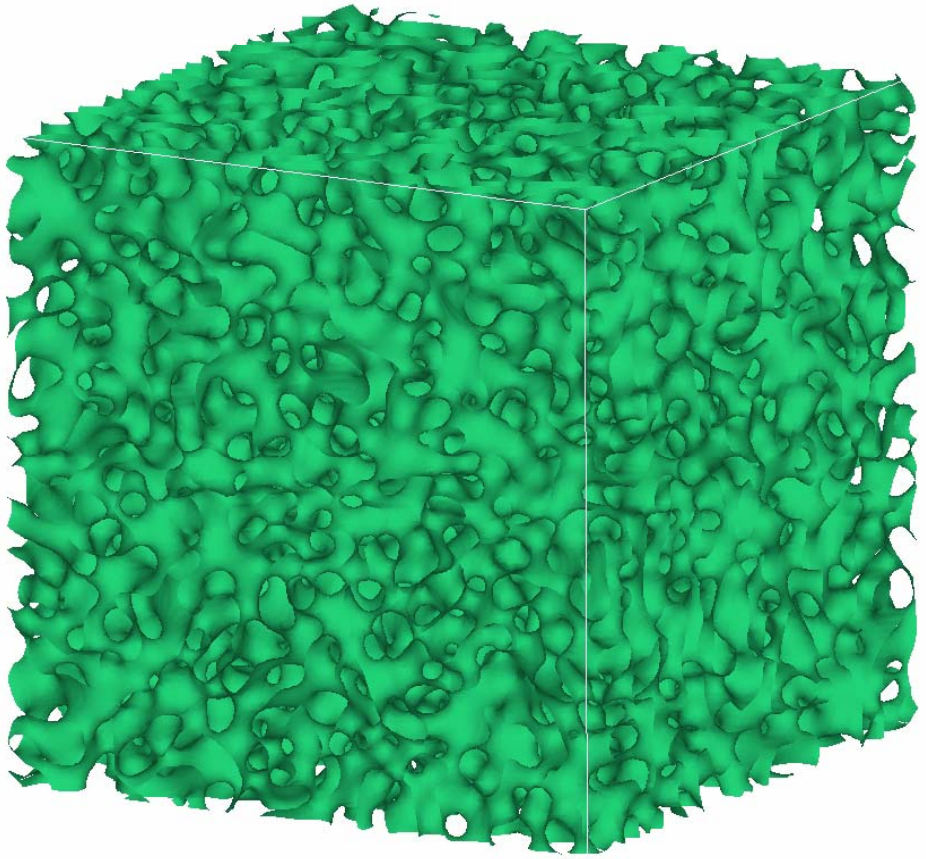
158 GFLOPS using GTX280

×160 256 x 256 x 256

Used register number = 46

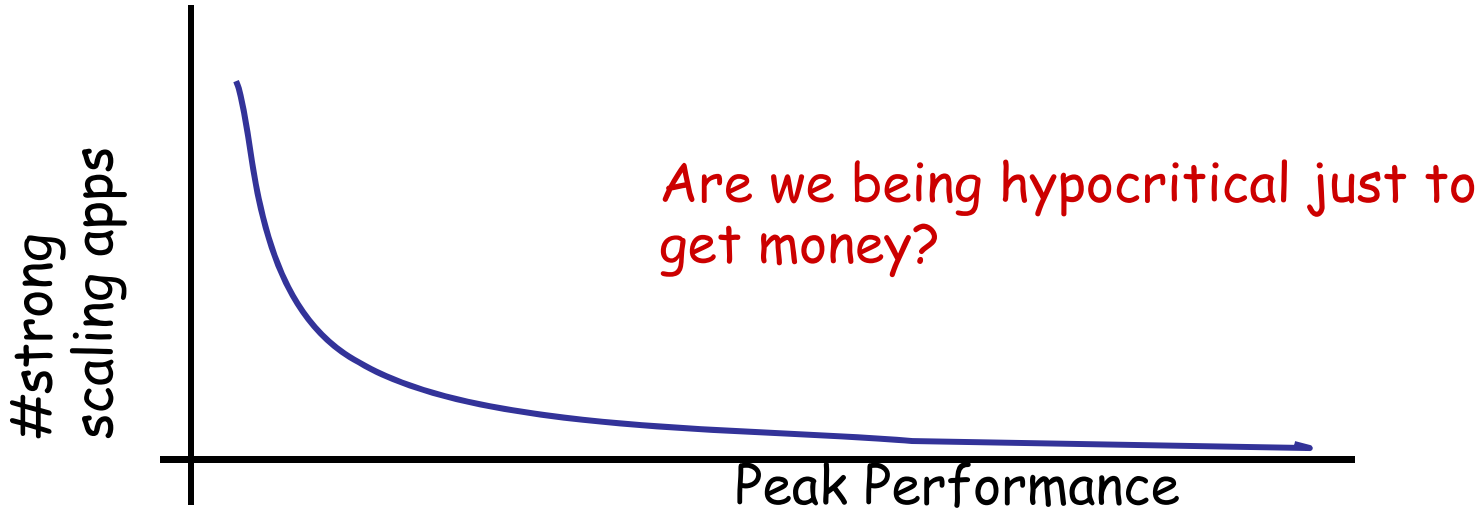
※ nvcc option –maxrregcount 32
for G80, 92





Can we make 100 million cores scale in "non-capacity capability app"?

- Capability --- latency matters, strong scaling
- ➔ requiring 1~10KB 1us messages to be efficient ➔ computation loop less than 1 us.
  => Can only tolerate 1/1000 fluctuation i.e. both loop and communicaiton will be 1ns, c.f. strong scaling code on a petaflops machine
  => Even with 3-D stencils expect 1/30~1/100 i.e. 10-30ns



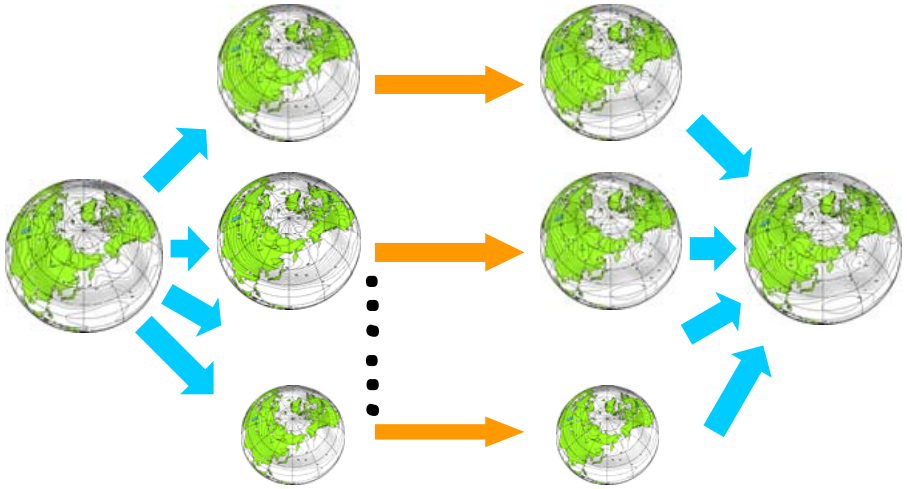
A Typical "Weak Scaling Capability App"
- Capacity App in Disguise -

```
Initialize;
Loop until computation gets done {
  MPI_AllScatter();
  Do work within node for seconds,
  minutes, hours…;
  MPI_AllGather();
}
Finalize;
```

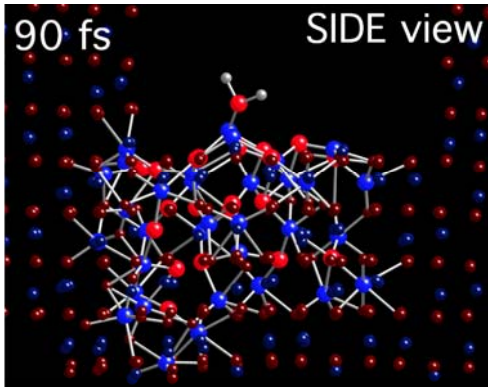
--- And is grossly inefficient compared to say simple workstealing parameter-sweep esp. if load is unbalanced

So the world will mostly go ensemble

--- capability at core, capacity at large ---



Barotropic S-Model
Ensemble climate
simulation



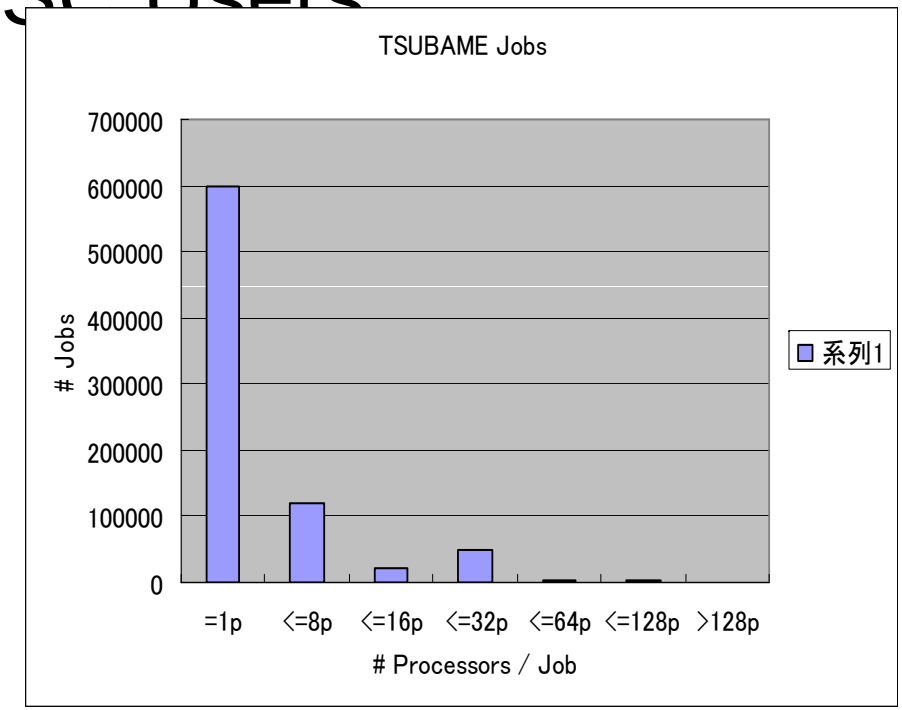
QM/MM Molecular
Simulation

"How are we to judge sciences, in that using 100,000 cores in a single MPI app has more scientific significance than 100,000 single-threaded app, as they both require system scalability in the design?"

BTW, MW may need to scale better for "capacity" e.g. BQ systems

TSUBAME Job Statistics
Dec. 2006-Aug.2007 (#Jobs)
1400 SC Users

- 1400 SC Users, 797,886 Jobs (~3270 daily)
- 597,438 serial jobs (74.8%)
- 121,108 <=8p jobs (15.2%) 90%
- 129,398 ISV Application Jobs (16.2%)
- *However, >32p jobs account for 2/3 of cumulative CPU usage*



Coexistence of ease-of-use in both
- *short duration* parameter survey
- large scale MPI
(Both are hard for *physically* large-scale distributed grid)

Issues, Issues, Issues

- Large Scale Fault Tolerance
- Programming Models
- Storage (terabytes of bandwidth)

- Being Green: Power consumption and cooling
  - Not just the money but CO2 emission
- Sustainability

- $$$ - Subsidy Necessary

Resources	Yens/CPU hour
Amazon E2C Single Instance	10.5
Tokyo Tech. GSIC SLA Svc.	0.543
Tokyo Tech. GSIC Best Effort Svc.	0.171
Univ. Tokyo Dedicated Q	0.89

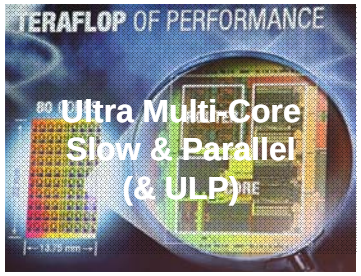
- Education
  - Great hopes on hybrid programming

Biggest Problem is Power…

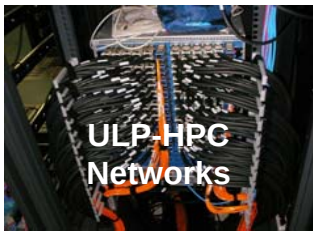
Machine	CPU Cores	Watts	Peak GFLOPS	Peak MFLOPS/ Watt	Watts/ CPU Core	Ratio c.f. TSUBAME
TSUBAME(Opteron)	10480	800,000	50,400	63.00	76.34	
TSUBAME2006 (w/360CSs)	11,200	810,000	79,430	98.06	72.32	
TSUBAME2007 (w/648CSs)	11,776	820,000	102,200	124.63	69.63	1.00
Earth Simulator	5120	6,000,000	40,000	6.67	1171.88	0.05
ASCI Purple (LLNL)	12240	6,000,000	77,824	12.97	490.20	0.10
AIST Supercluster (Opteron)	3188	522,240	14400	27.57	163.81	0.22
LLNL BG/L (rack)	2048	25,000	5734.4	229.38	12.21	1.84
Next Gen BG/P (rack)	4096	30,000	16384	546.13	7.32	4.38
TSUBAME 2.0 (2010Q3/4)	160,000	810,000	1,024,000	1264.20	5.06	10.14

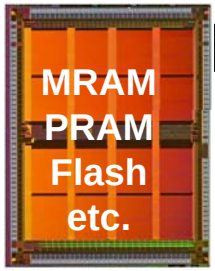
TSUBAME 2.0 x24 improvement in 4.5 years…? ➔ ~ x1000 over 10 years

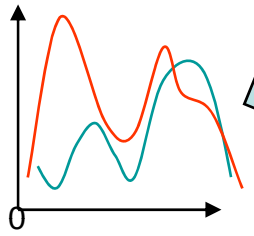
The new JST-CREST "Ultra Low Power (ULP)-HPC" Project 2007-2012



ULP-HPC SIMD-Vector (GPGPU, etc.)







Modeled ULP-HPC HW, Middleware, etc.

Generalized Autotuning Scheme

Bayesian Merging of Model and Measurement

- Bayes model and distribution

$$y_i \sim N(\mu_i, \sigma_i^2)$$

$$\mu_i \mid \beta, \sigma_i^2 \sim N(x_i^T \beta, \sigma_i^2 / \kappa_0)$$

$$\sigma_i^2 \sim \text{Inv-}\chi^2(\nu_0, \sigma_0^2)$$

Model-based predect of exe. time

- Measured distribution after n trials

$$y_i \mid (y_{i1}, y_{i2}, \cdots, y_{in}) \sim t_{\nu_n}(\mu_{in}, \sigma_{in}^2 \kappa_{n+1} / \kappa_n)$$

$$\nu_n = \nu_0 + n, \quad \kappa_n = \kappa_0 + n, \quad \mu_n = (\kappa_0 x_i^T \beta + n\bar{y}_i)/\kappa_n$$

$$\nu_n \sigma_n^2 = \nu_0 \sigma_0^2 + \sum (y_m - \bar{y}_i)^2 + \kappa_0 n (\bar{y}_i - x_i^T \beta)^2 / \kappa_n$$

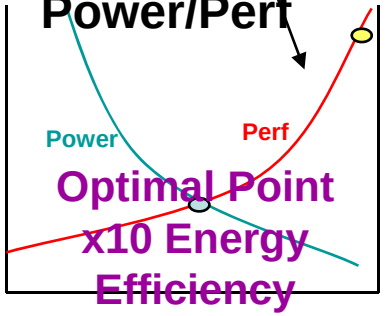
Measured exec. time

ABCLibScript: Algorithm Selection

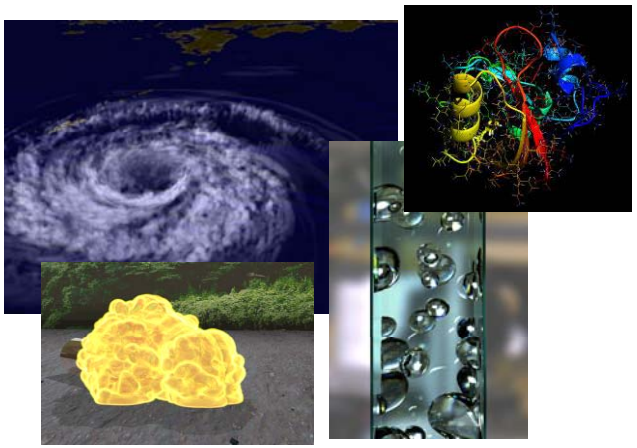
```
!ABCLib$ static select region start
!ABCLib$ parameter (in CacheS, in NB, in NPrc)
!ABCLib$    select sub region start
!ABCLib$    according estimated
!ABCLib$    (2.0d0*CacheS*NB)/(3.0d0*NPrc)
            Target 1 (Algorithm 1)
!ABCLib$    select sub region end
!ABCLib$    select sub region start
!ABCLib$    according estimated
!ABCLib$    (4.0d0*ChcheS*dlog(NB))/(2.0d0*NPrc)
            Target 2 (Algorithm 2)
!ABCLib$    select sub region end
!ABCLib$ static select region end
```

Specify start of Tuning region
Input Parameters into the Cost Function
Cost Functions
Tuning Targets

Optimize Power/Perf



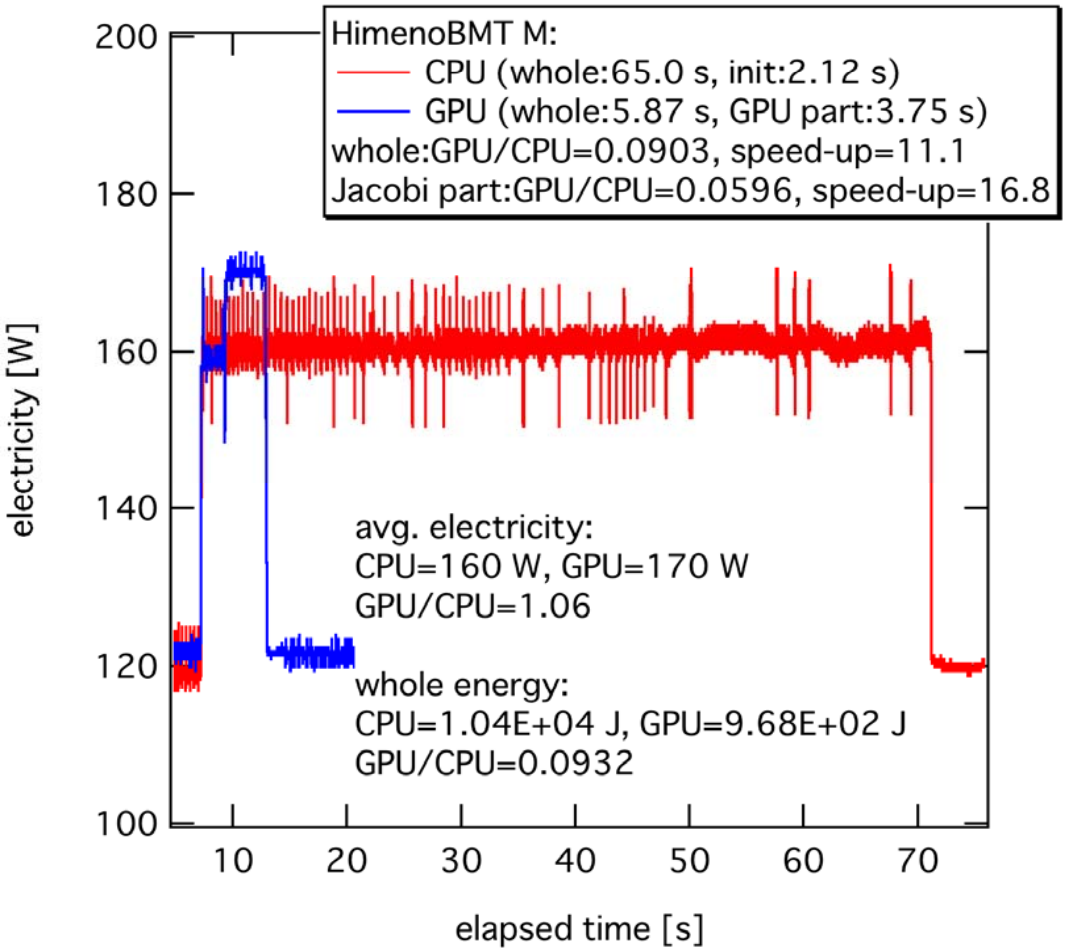
Optimal Point x10 Energy Efficiency



Modeled ULP-HPC Applications

2016 TSUBAME becomes 1/1000

Himeno Size M Power Measurement



Jacobi
* CPU
Av. Power 160W
Exec. Time：62.88s
* GPU
Av. Power：170W
Exec. Time：3.75s

GPU/CPU
Power：106%
Energy:1/15.8
＝6.33%

For extreme memory intensive CFD app GPU uses only 6% of CPU energy

Power Efficiency in 3-D FFD

GPU	Computation	Idle	Power	GFLOPS	GFLOPS/W
RIVA128		126 W	140 W	10.3	0.074
8800 GT	On GPU	180 W	215 W	62.2	
8800 GTS	On GPU	196 W	238 W	67.2	
8800 GTX	On GPU	224 W	290 W	84.4	

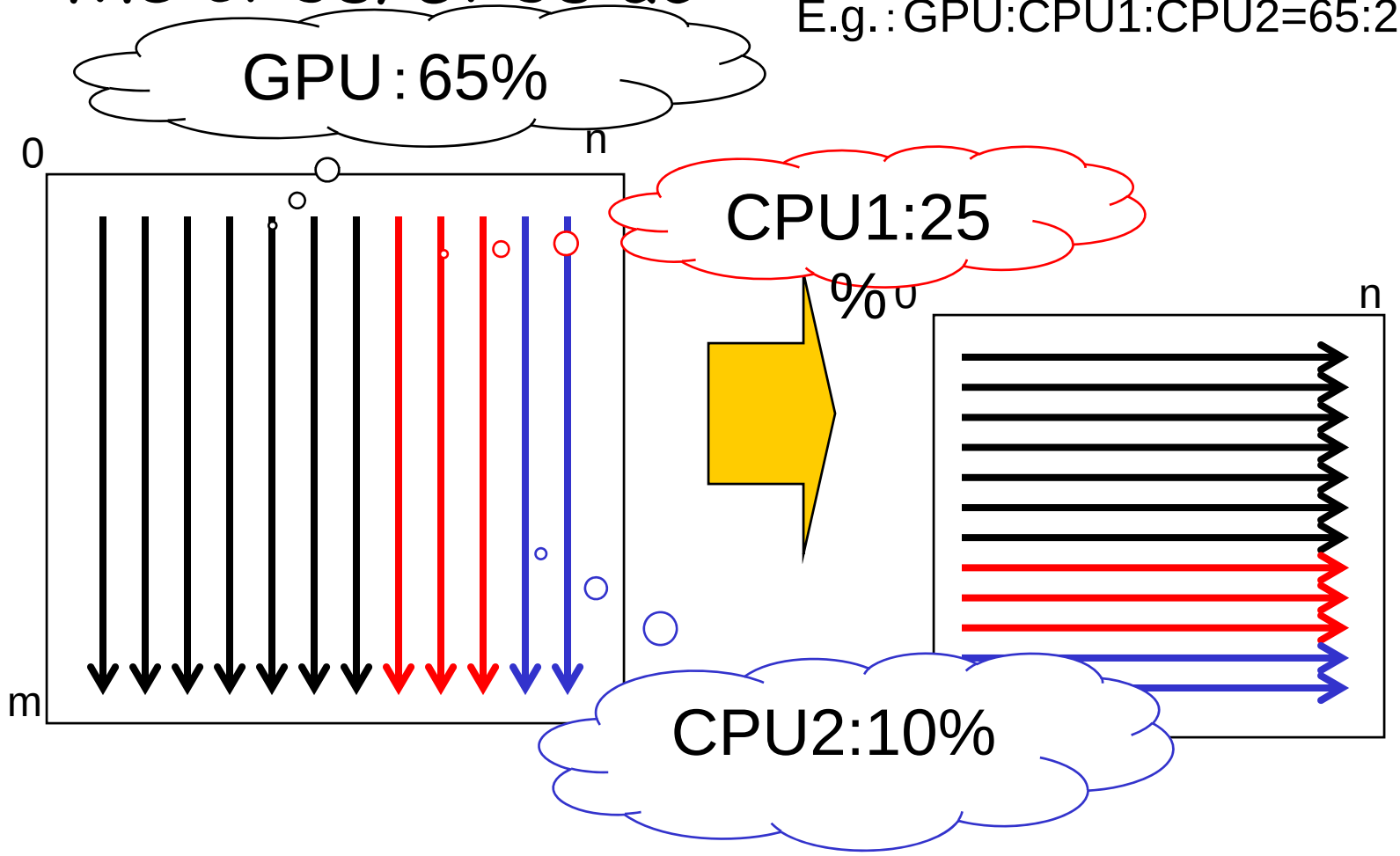
CUDA GPUs have four times
higher power efficiency than CPU.

RIVA128 is an old, low-power GPU,
to measure pure power consumption
of host system (CPU, chipset, memory).
The interface is legacy PCI.

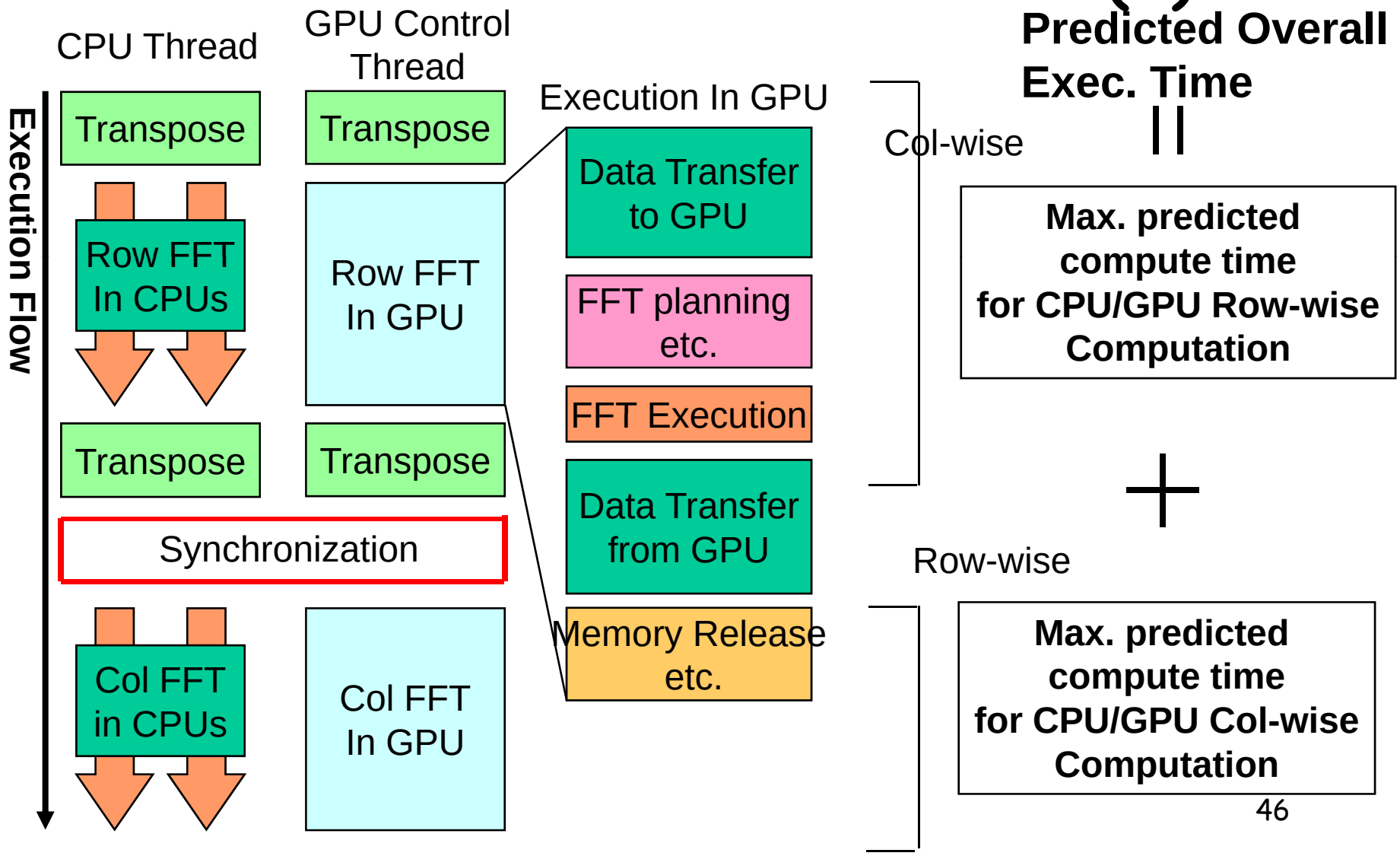
Dividing the Labor in 2D FFT ([IEEE IPDPS-HCW08])

- Divide the labor for each 1D FFT among the CPUs/GPUs ac

E.g.: GPU:CPU1:CPU2=65:25:10

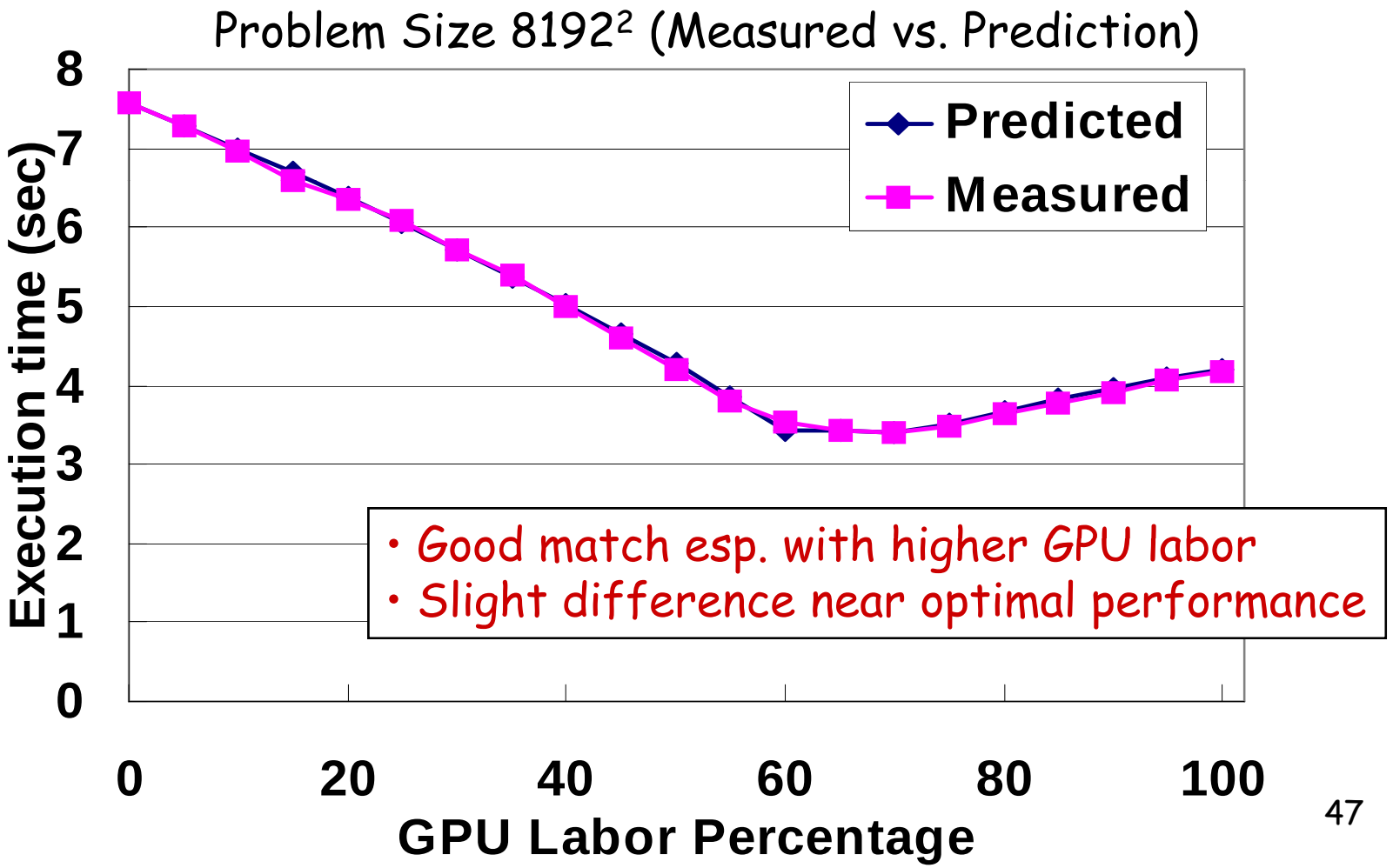


Performance Model of Combined CPU/GPU FFT Execution (1)



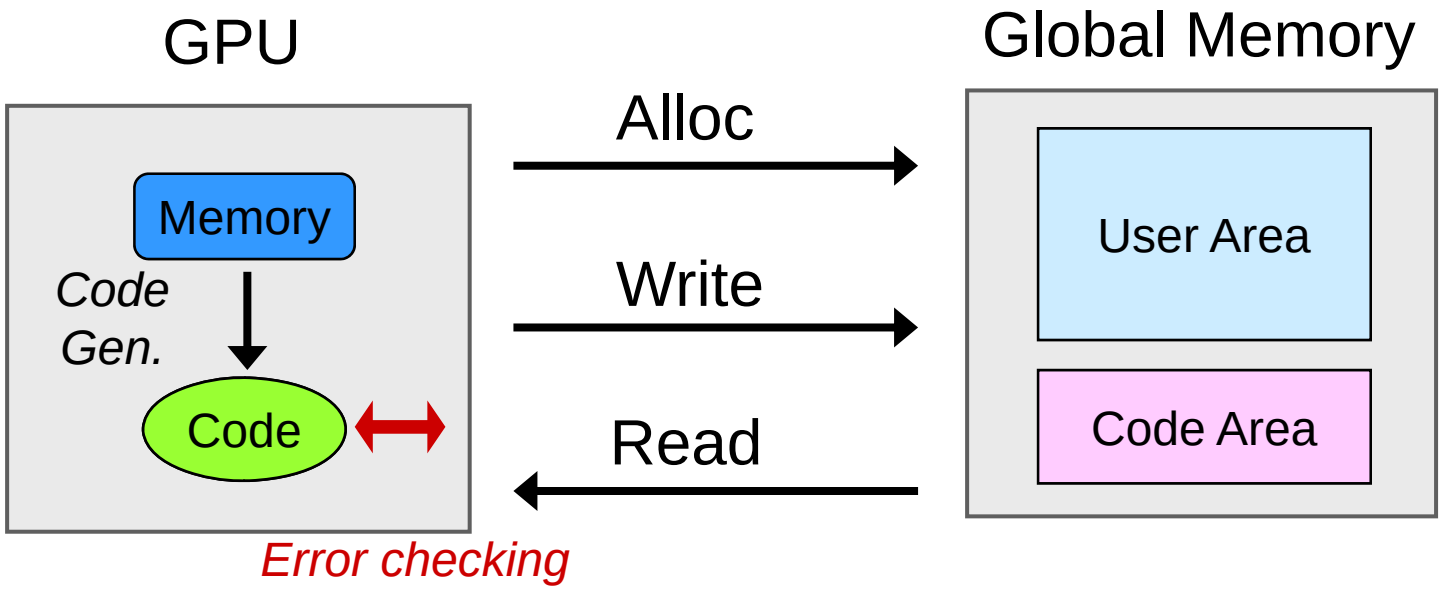
Performance Model Accuracy(1)

- Problem Size 8192^2, 1 CPU + GPU
- Predicted perf. of 8192^2 derived from our model



Software-Based Error Checking on GPUSs

- Protects applications from bit-flip errors in global memory using error-checking code (e.g., parity and ECC)
  - memory allocation → *allocates additional global memory for storing error check code*
  - write accesses → *calculates code for data written*
  - read accesses → *calculates and compares code for data read*



N-body Problem with Error Checking

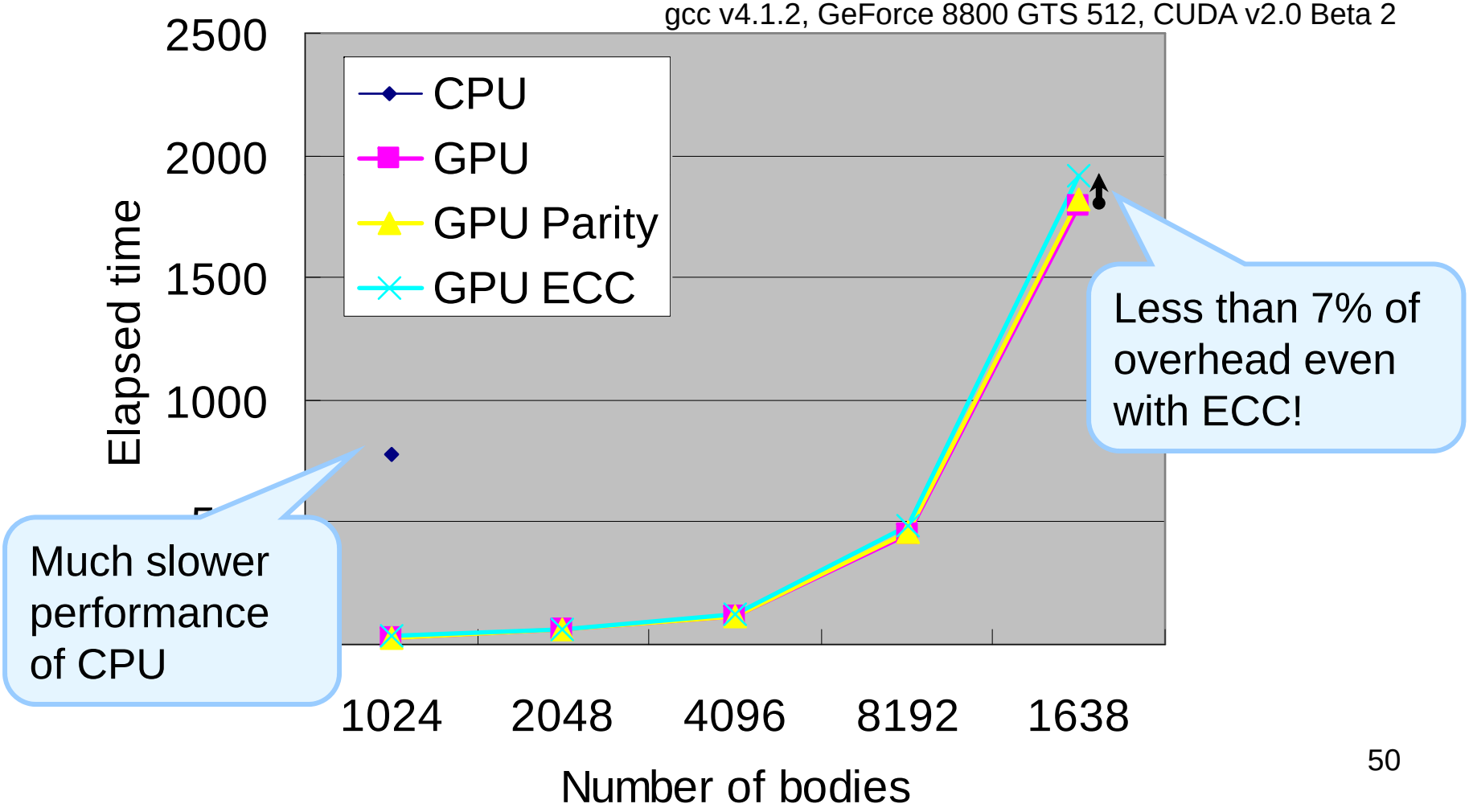
- The sample code shipped with the CUDA SDK
- Error checking
  - read accesses to position array
  - write accesses to position and velocity arrays

```
nbody (float *pos, unsigned *pos_code,
       float *vel, unsigned *vel_code) {
  float4 *mypos = pos[i];
  check_float4(mypos, i);
  for (j in BLOCK) {
    shared float4 block[];
    block[k] = pos[k];
    check_float4(block[k], k);
    for (pt in block_size) {
      acc+=compute_acc(mypos,block[pt]);
    }
  }
  update_pos(pos, vel);
  write_check_code_float4(pos);
  update_vel(vel, acc);
  write_check_code_float4(vel);
}
```

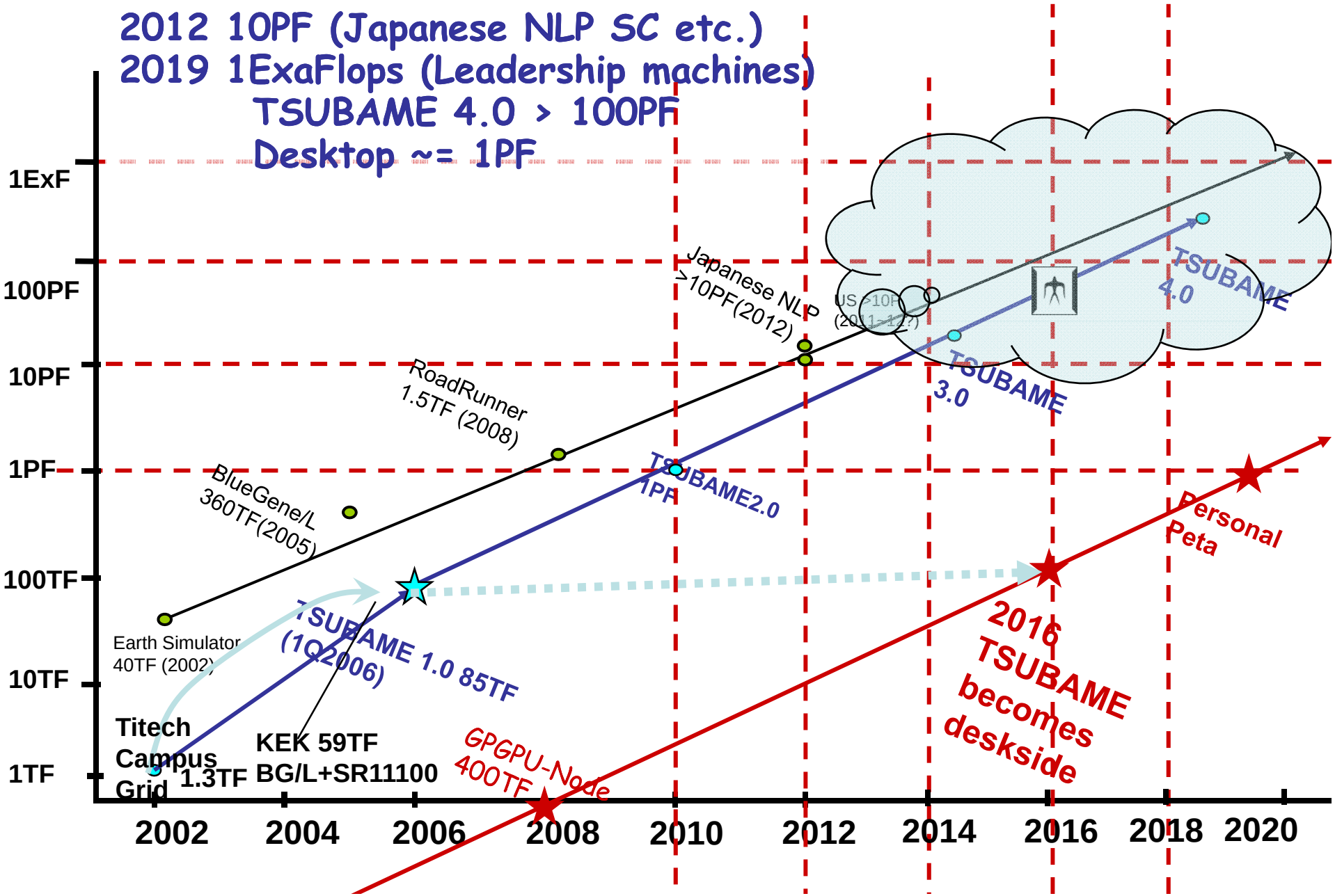
Results of N-body Problem

CPU (4 cores with OpenMP) vs GPU vs GPU+Parity vs GPU+ECC

Phenom 9850 2.5 GHz, 4 GB of DRAM, Linux v2.6.23, gcc v4.1.2, GeForce 8800 GTS 512, CUDA v2.0 Beta 2



Road to Exascale



So the first exascale machine will be a Cloud with massive user base (?)

~100TF
Capacity
#Users
~100
~1GF

> x10,000

~1ExaF
Capacity
x10^6
x10^7
#Users
~1,000,000
~100GF