Petascaling Commodity onto Exascale with GPUs on TSUBAME1.2 onto TSUBAME2.0

Satoshi Matsuoka, Professor/Dr.Sci.

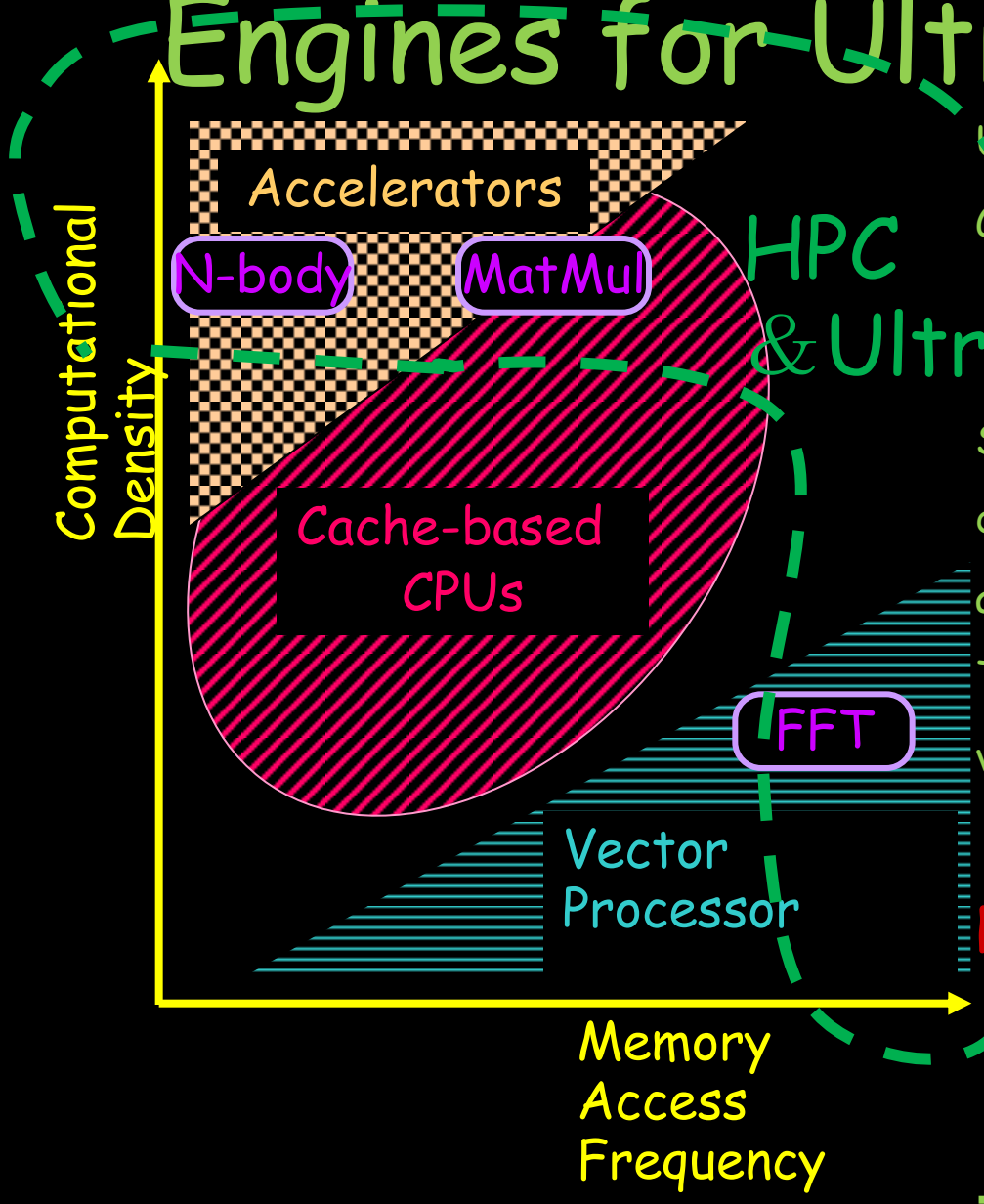
Global Scientific Information and Computing Center (GSIC)
Tokyo Inst. Technology

GPUs as Commodity Massively Parallel Vector Processors for Ultra Low Power

- E.g., NVIDIA Tesla, AMD Firestream
  - High Peak Performance > 1TFlops
    - Good for tightly coupled code e.g. Nbody
  - **High Memory bandwidth (>100GB/s)**
    - **Good for sparse codes e.g. CFD**
  - **Low latency over shared memory**
    - **Thousands threads hide latency w/zero overhead**
  - **Slow and Parallel and Efficient vector engines for HPC**
  - Restrictions: Limited non-stream memory access, PCI-express overhead, programming model etc.

How do we exploit them given vector computing experiences?

GPUs as Commodity Vector Engines for Ultra Low Power



Unlike the conventional accelerators, GPUs have high memory bandwidth and dense

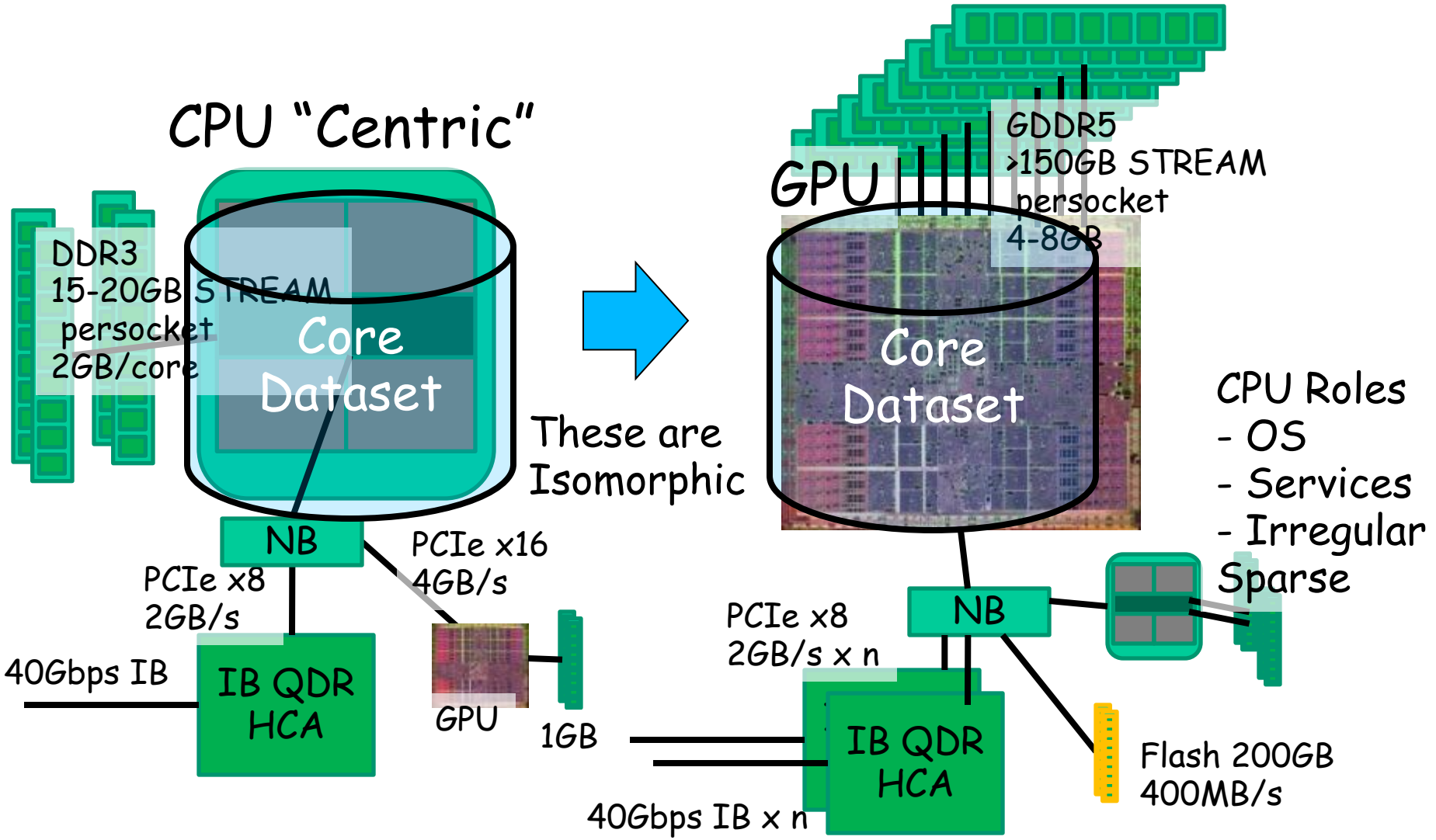
Since latest high-end GPUs support double precision, GPUs also work as commodity vector processors.

The target application area for GPUs is very wide.

Restrictions: Limited non-stream memory access, PCI-express overhead, etc.

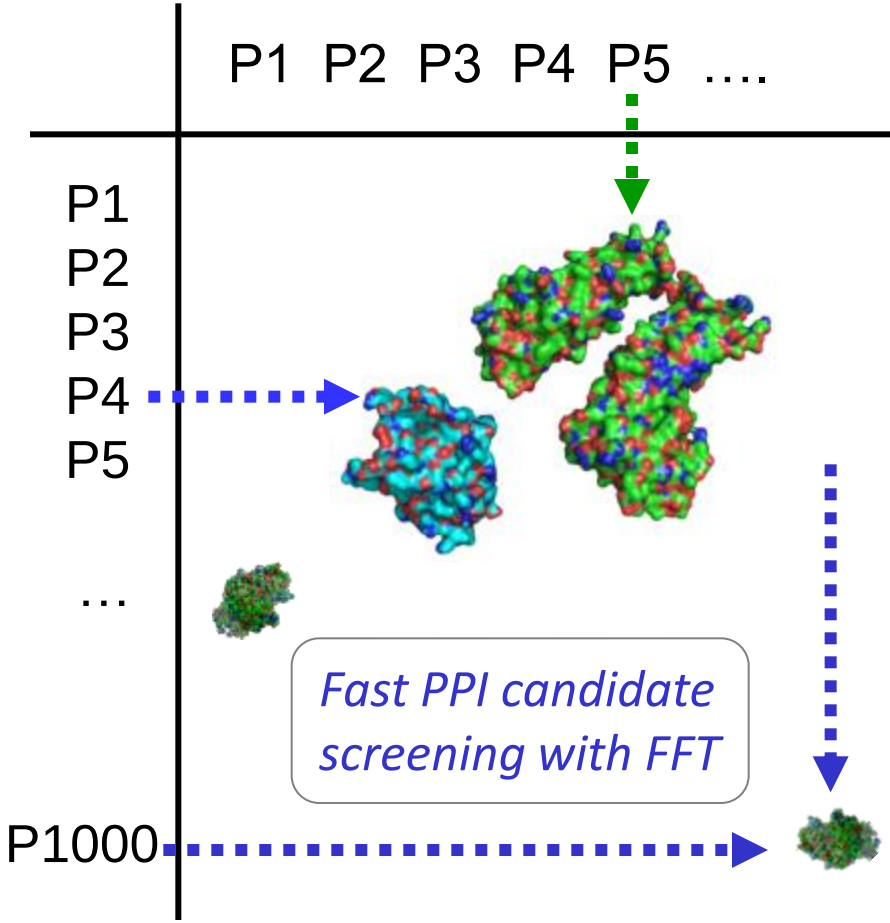
➔ How do we utilize them easily?

From CPU Centric to GPU Centric Nodes for Scaling



All-to-all 3-D Protein Docking Challenge

(Collaboration with Yutaka Akiyama, Tokyo Tech.)



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).

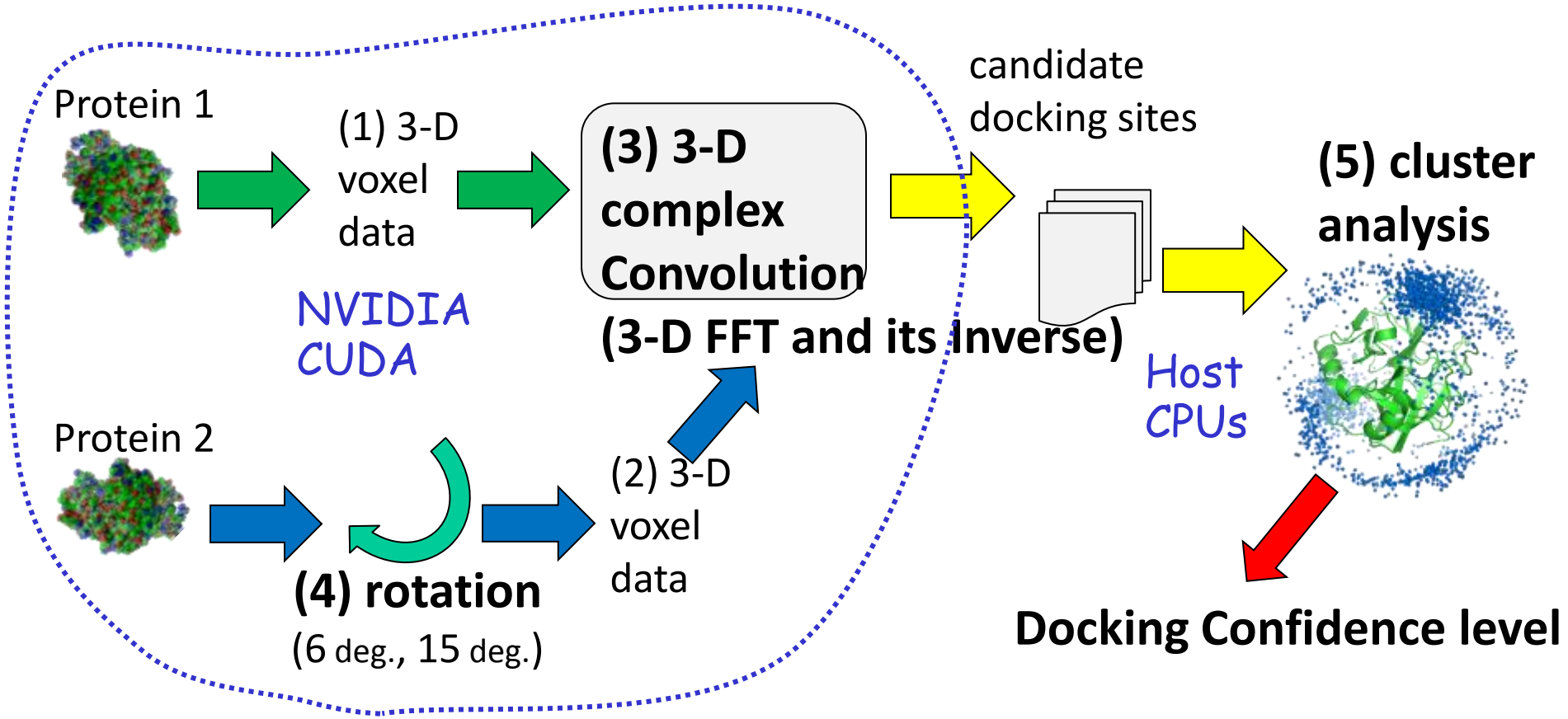
cf.

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

~ 2 years with 1-rack BlueGene/L

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

Algorithm for 3-D All-to-All Protein Docking



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

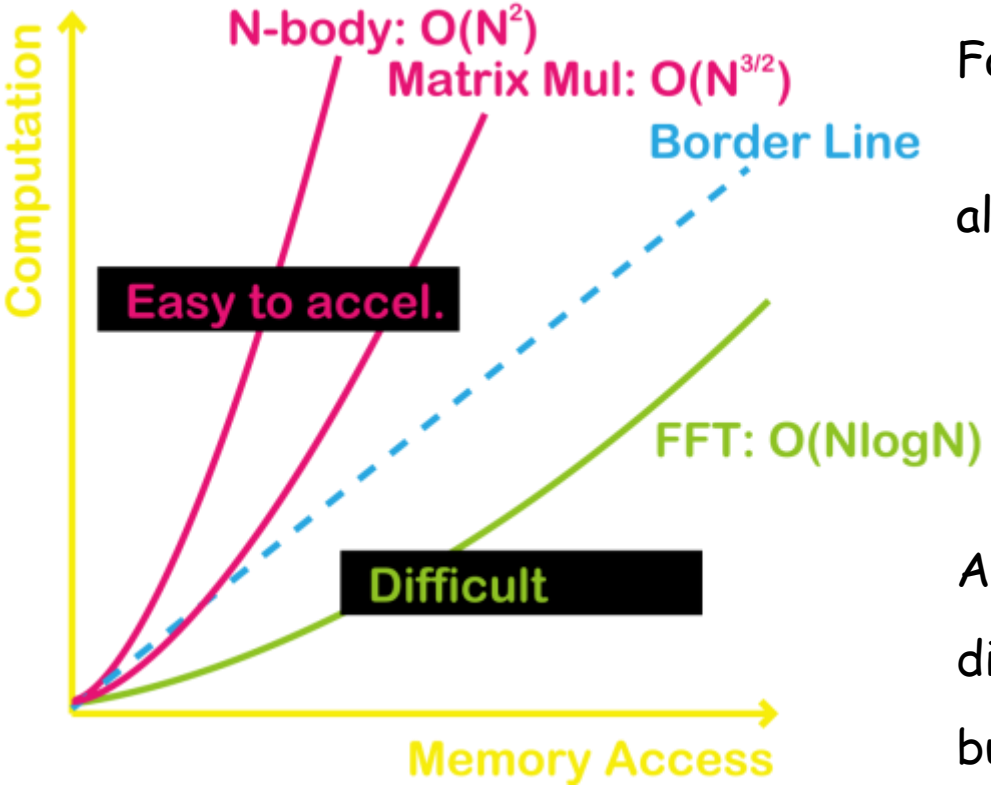
3-D complex convolution $O(N^3 \log N)$, *typically* $N = 256$

x

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

200 Exa Ops for 1000 x 1000

High Performance 3-D FFT on NVIDIA CUDA GPUs [Nukada et. al. SC08]

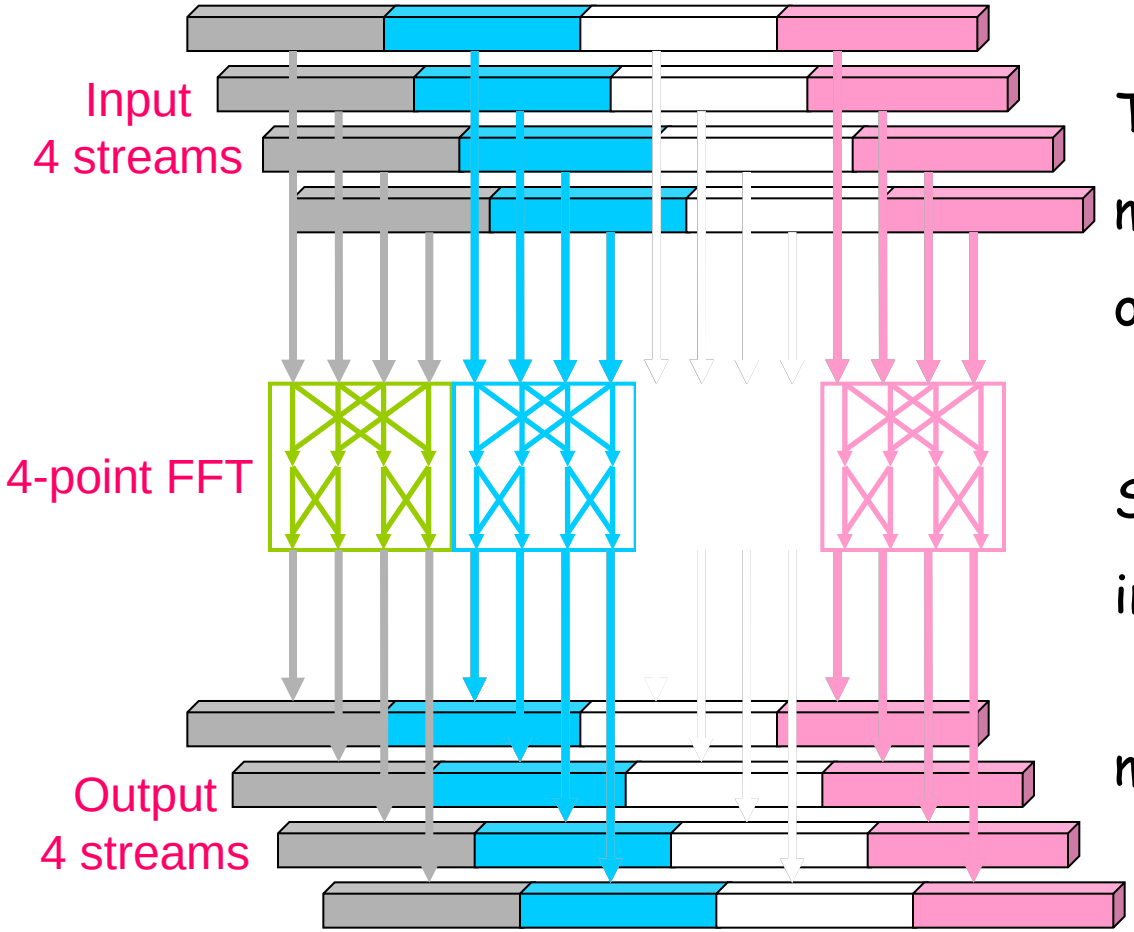


Fast Fourier Transform is an $O(N \log N)$ algorithm with $O(N)$ memory access.

Acceleration of FFT is much more difficult than N-body, MatMul, etc., but possible!

At least, GPUs' theoretical memory bandwidth is much higher than CPUs.

multi-row FFT algorithm for GPU [SC08]

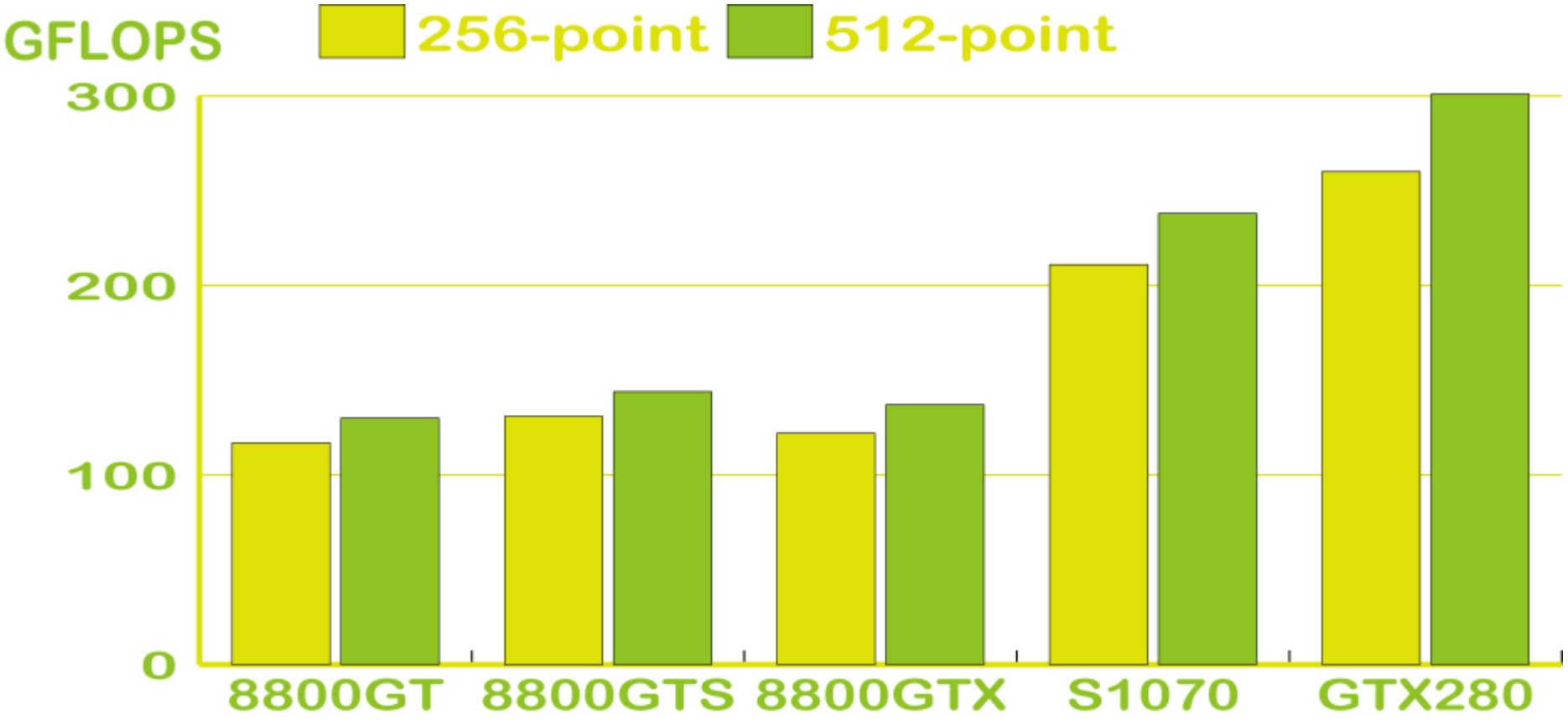


This algorithm accesses multiple streams, but each of them is successive.

Since each thread compute independent set of small FFT,
many registers are required.

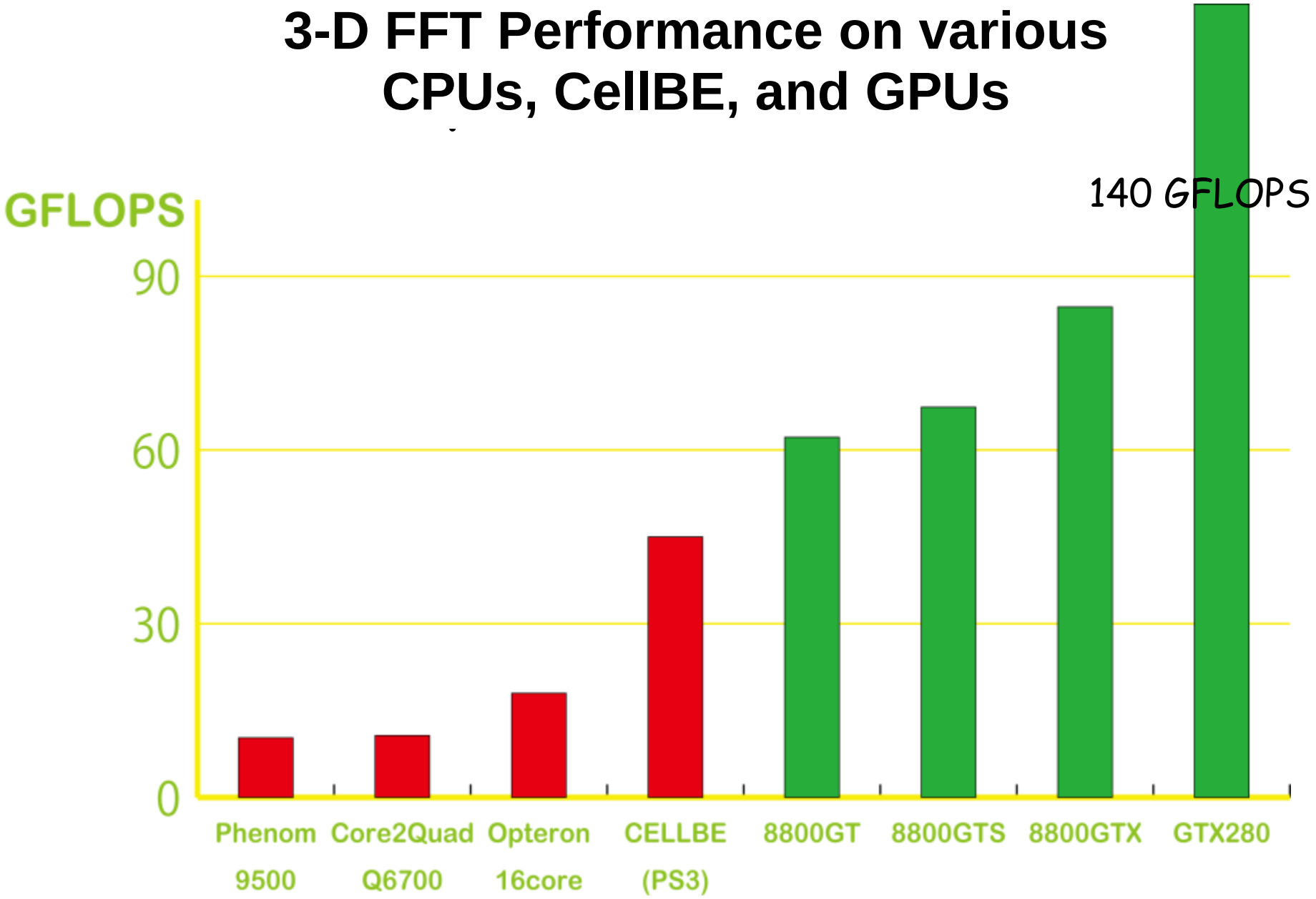
For 256-point FFT, use two-pass 16-point FFT kernels.

Performance of 1-D FFT

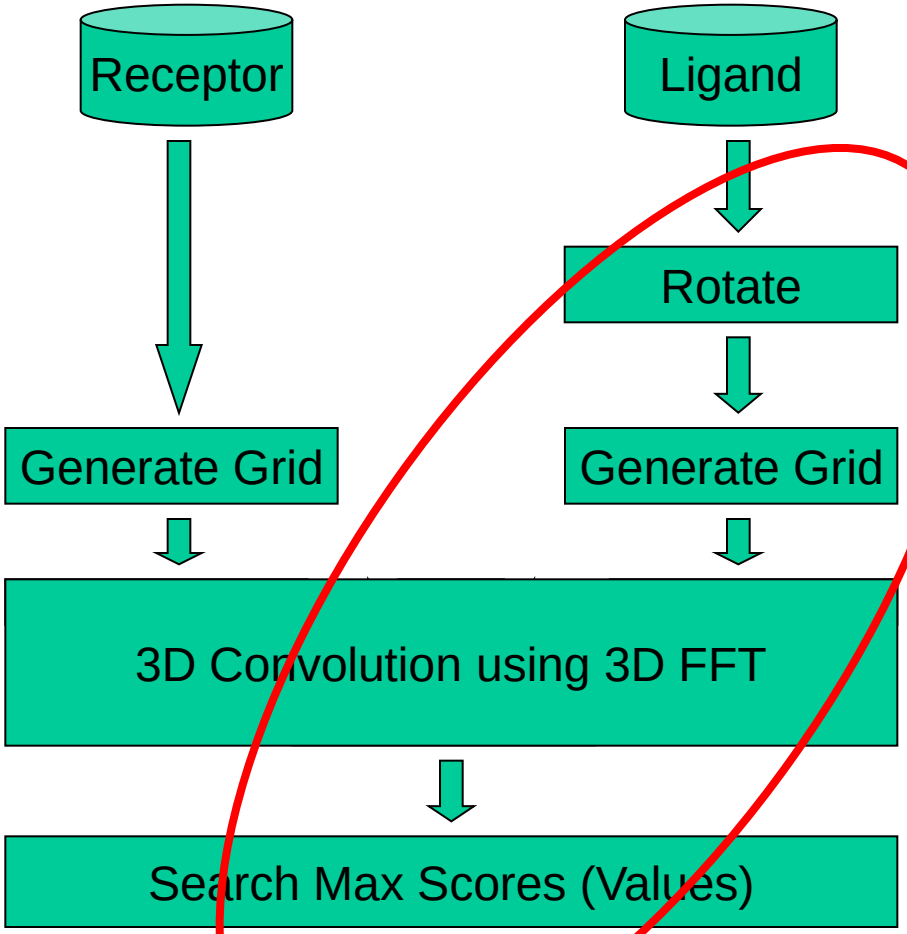


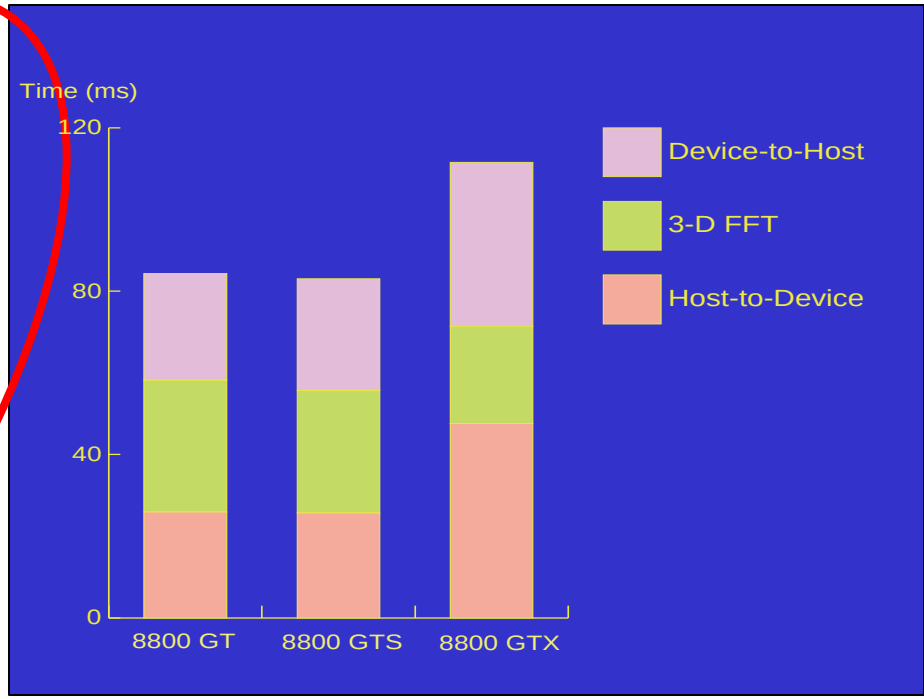
Note: An earlier sample with 1.3GHz is used for Tesla S1070.

3-D FFT Performance on various CPUs, CellBE, and GPUs



Overcoming the PCI-e BW Limitations---Keeping the Protein Data on the GPU





MAIN LOOP on GPGPU Card

=> pack as much card in node as possible

Heavily GPU Acclerated Windows HPC Prototype Cluster

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

Performance Estimation of 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	8~13

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,Japan)	4096 (4racks)	80	22.9	7.0	87.5
TSUBAME. (Opteron Only)	655 (~70 racks)	~700	50.3 (DP)	17.5 (DP)	25
GPU Accel .WinHPC	32 (4racks)	18	53.2	6.5	361

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

On full TSUBAME 1.2, ~100TFlops (1MW)-> ~52 rack BG/L

TSUBAME 1.2 Experimental 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
> 20 Million Threads
~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

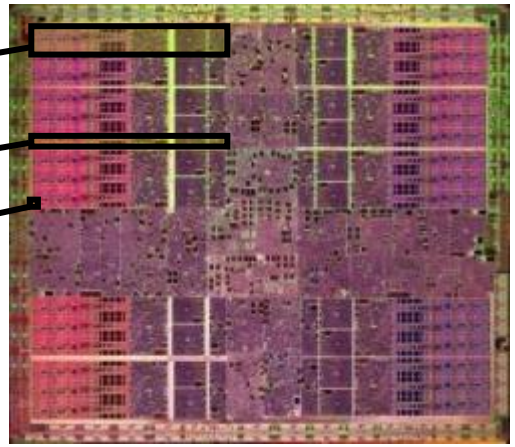
PCI-e

ClearSpeed CSX600
SIMD accelerator
648 boards,
52.2TeraFlops
SFP/DFP

170 Nvidia Tesla 1070, ~680 Tesla cards
High Performance in Many BW-Intensive Apps
10% power increase over TSUBAME 1.0


680 Unit Tesla Installation…
While TSUBAME in Production Service (!)

TSUBAME 1.2 Node Configuration

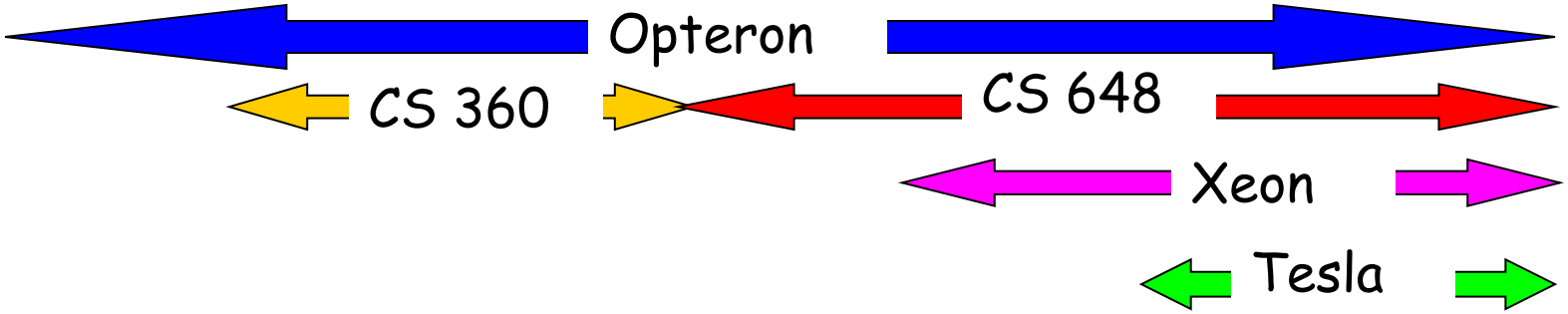


nVidia Tesla T10: 55nm, 470m2,
1.4billion transistors
>1TF SFP
90GF DFP

"Powerful Scalar"
x86
16 cores
2.4Ghz
80GFlops
20GBytes/s
32GB
PCI-e x8 gen1
2GB/s
240 SP Cores
30 SMs
1.4Ghz
1.08TFlops SFP
90GFlops DFP
GDDR3 4GB
102 GBytes/s
Tesla Accelerator
240 SP Cores
30 SMs
1.4Ghz
1.08TFlops SFP
90GFlops DFP
GDDR3 4GB
102 GBytes/s
Tesla Accelerator
Dual Rail x4 IB SDR
2 x 1GB/s

TSUBAME in Top500 Ranking

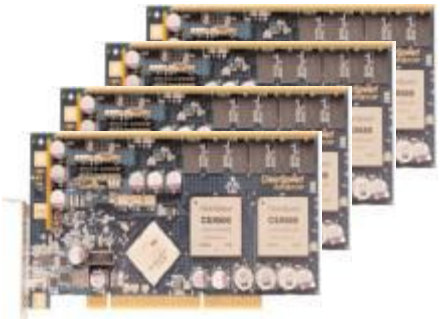
	Jun 06	Nov 06	Jun 07	Nov 07	Jun 08	Nov 08	Jun09
Rmax (Tflops)	38.18	47.38	48.88 [HPDC 2008]	56.43	67.70	77.48	87.01
Rank	7	9	14	16	24	29	41



- Continuous improvement for 6 times
- The 2nd fastest heterogeneous supercomputer in the world (No.1 is RoadRunner)

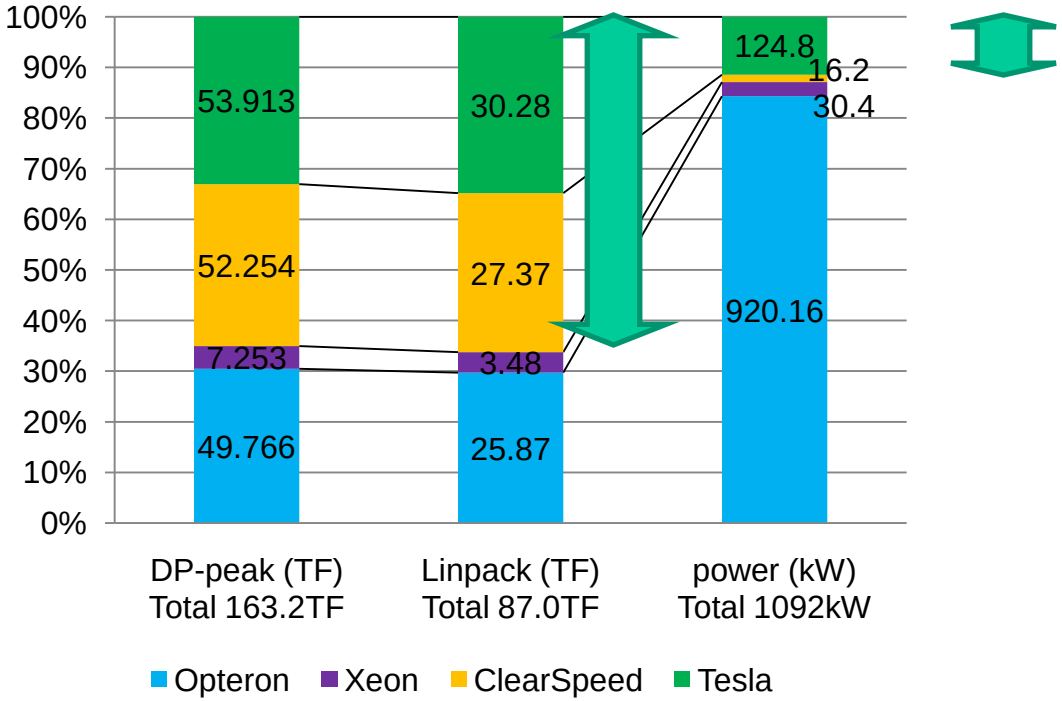
Linpack on TSUBAME---Heterogeneity at Work [Endo et.al. IPDPS08, etc.]

- Linpack implementation that uses cooperatively:
  - 10,368 Opteron cores
  - 612 Tesla GPUs
  - 648 ClearSpeed accelerators
  - 640 Xeon cores (another cluster named "tsubasa")
- Different strategy than on Roadrunner is required
- 87.01TFlops: The world's highest Linpack performance on GPU-accelerated clusters



Electric Power Consumption

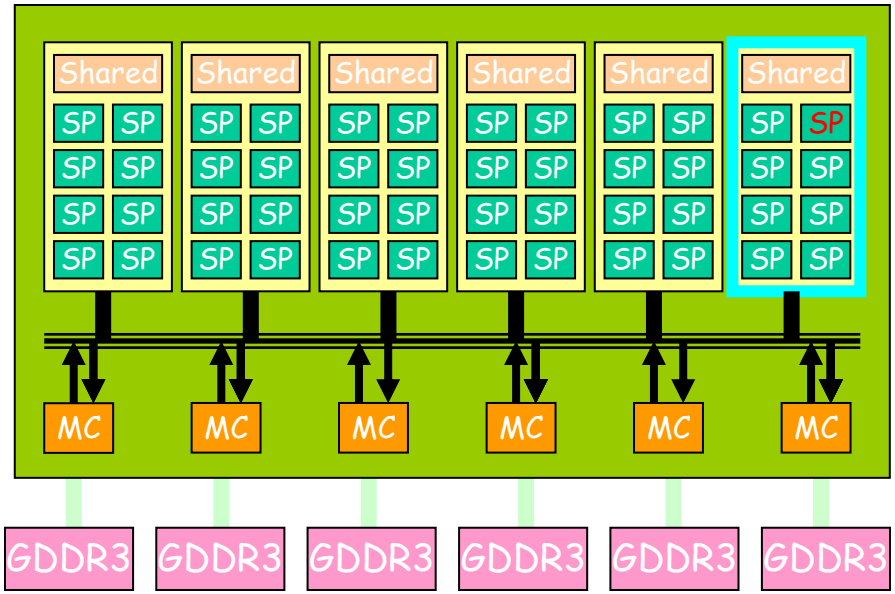
- About 1090kW during Linpack
  - An estimated value from sampling
  - Cooling, network are excluded



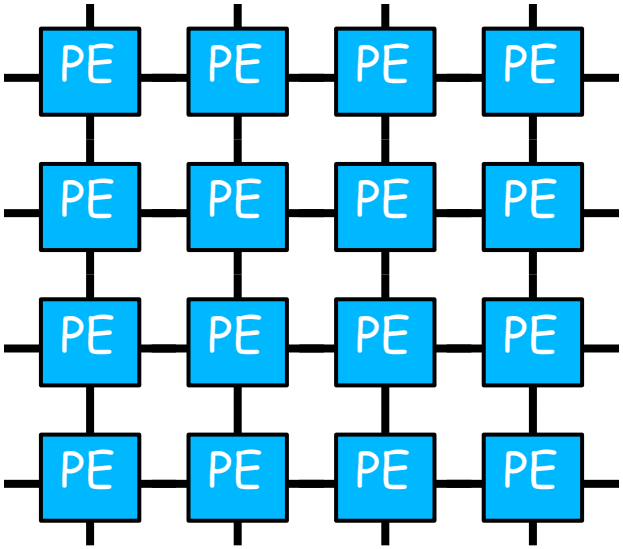
Even for dense problems GPUs are effective low power vector-like engines

Higher performance with much lower memory footprint

GPU vs. Standard Many Cores?



vs.



- Consider strong scaling simple stencil computation e.g., SOR

Multi-GPU in CFD: Riken Himeno Benchmark

(Joint Work w/NEC)

RIKEN Himeno CFD Benchmark

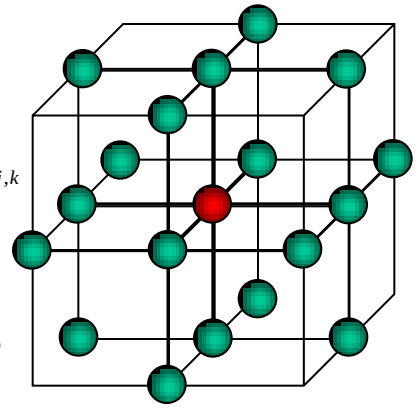
Poisson Equation:
(Generalized coordinate)

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

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

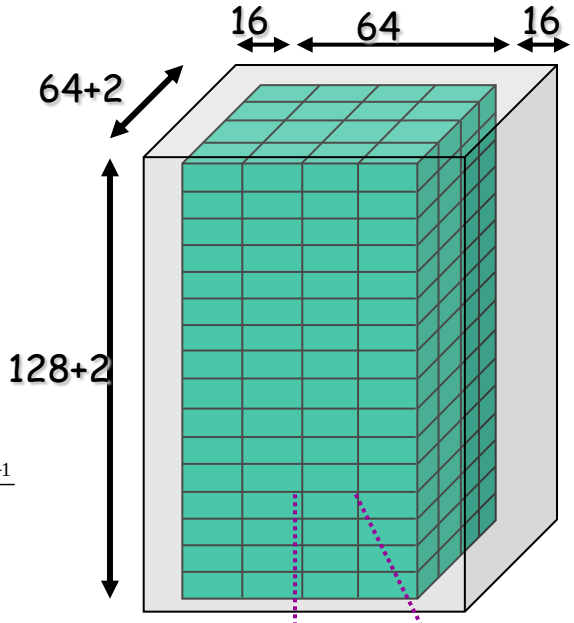
Discretized Form:

$$\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{\Delta x^2} + \frac{p_{i,j+1,k} - 2p_{i,j,k} + p_{i,j-1,k}}{\Delta y^2} + \frac{p_{i,j,k+1} - 2p_{i,j,k} + p_{i,j,k-1}}{\Delta y^2}$$
$$+ \alpha \frac{p_{i+1,j+1,k} - p_{i-1,j+1,k} - p_{i+1,j-1,k} + p_{i-1,j-1,k}}{4\Delta x \Delta y}$$
$$+ \beta \frac{p_{i+1,j,k+1} - p_{i-1,j,k+1} - p_{i+1,j-1,k} + p_{i-1,j,k-1}}{4\Delta x \Delta z}$$
$$+ \gamma \frac{p_{i,j+1,k+1} - p_{i,j+1,k-1} - p_{i,j-1,k-1} + p_{i,j-1,k-1}}{4\Delta y \Delta z} = \rho_{i,j,k}$$



18 neighbor point access

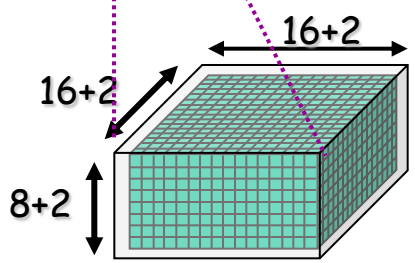
Himeno for CUDA



1 block = 16x16x8 compute region

Block has 256 thread

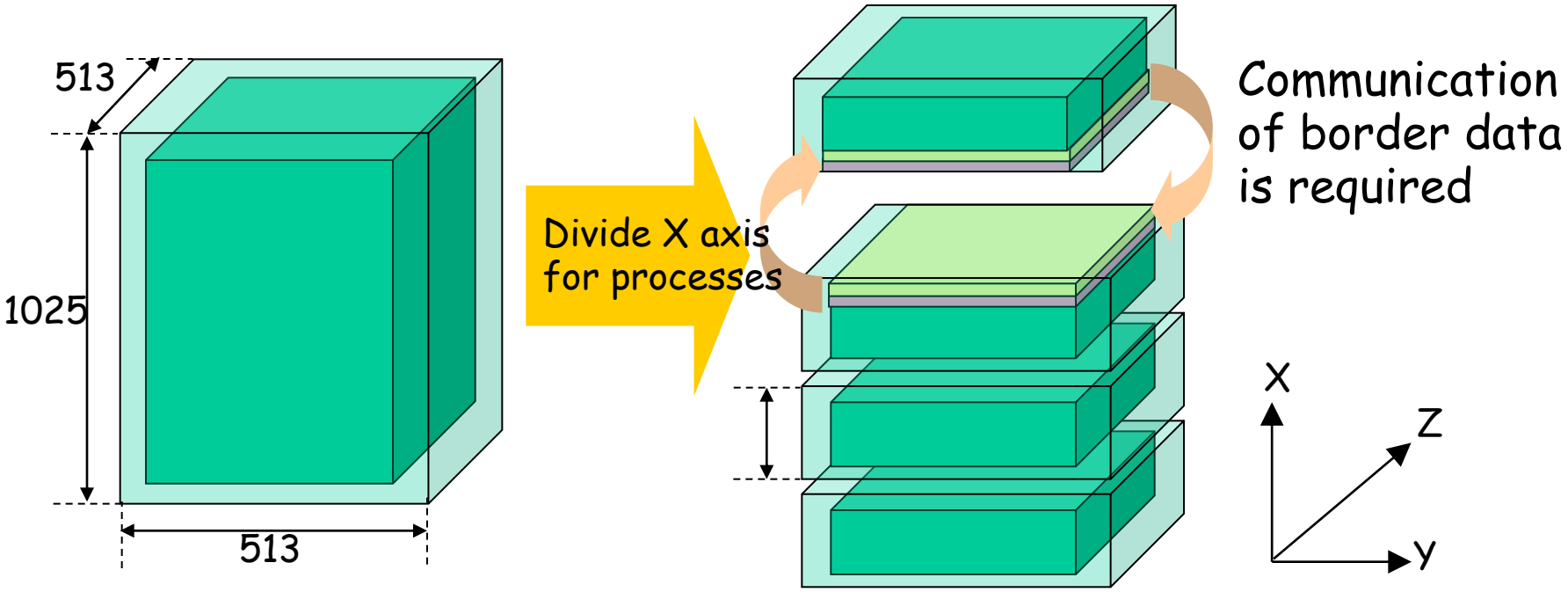
Total 256 blocks = 65536 threads



Block shared mem =16kB

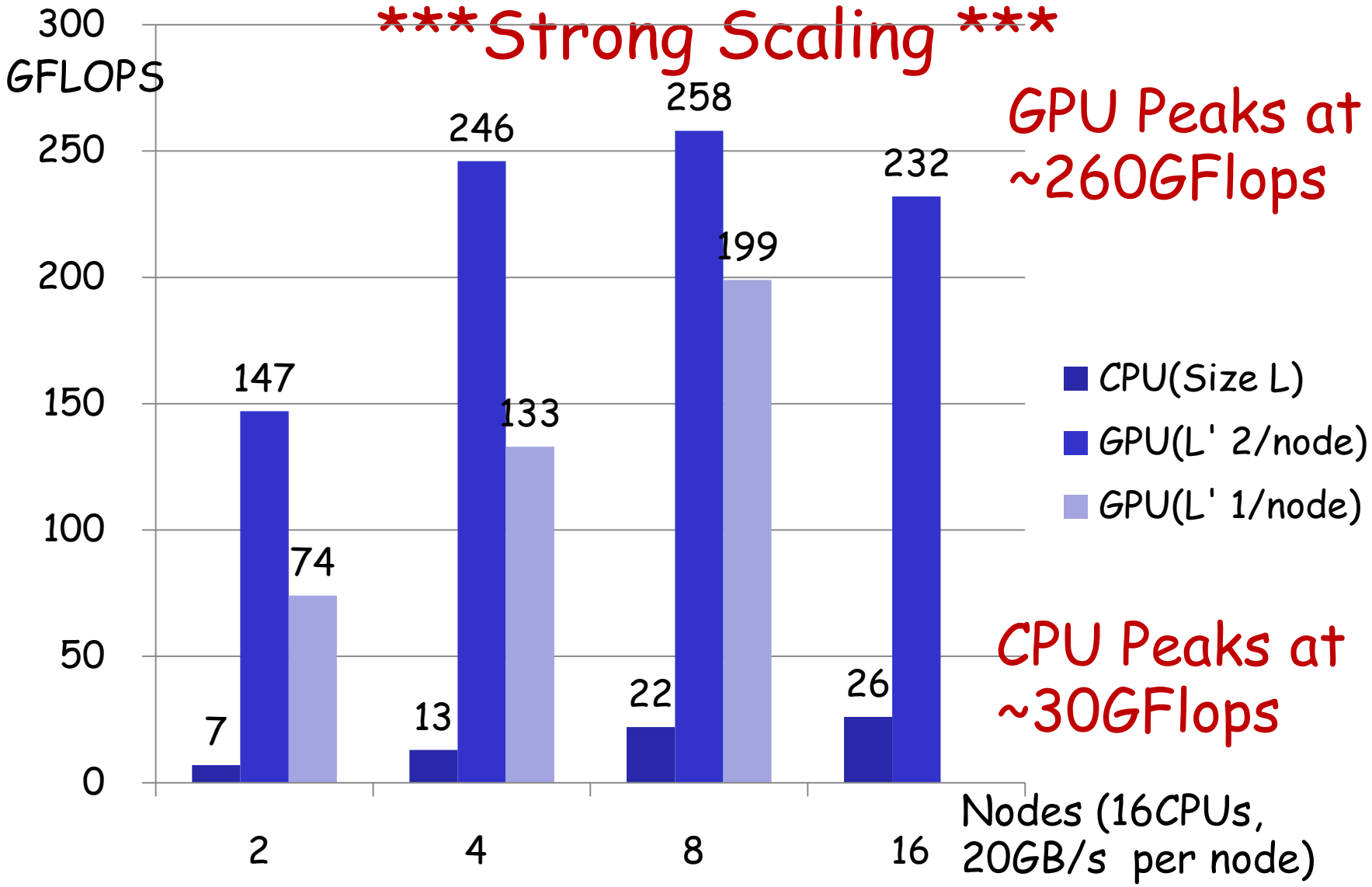
Boundary region used for transfer

Parallelization of Himeno Benchmark



- Although original Himeno supports 3D division, our GPU version currently supports only 1D division

Himeno Size L/L' (257x257x513) CPU vs. GPU Scaling on TSUBAME 1.2



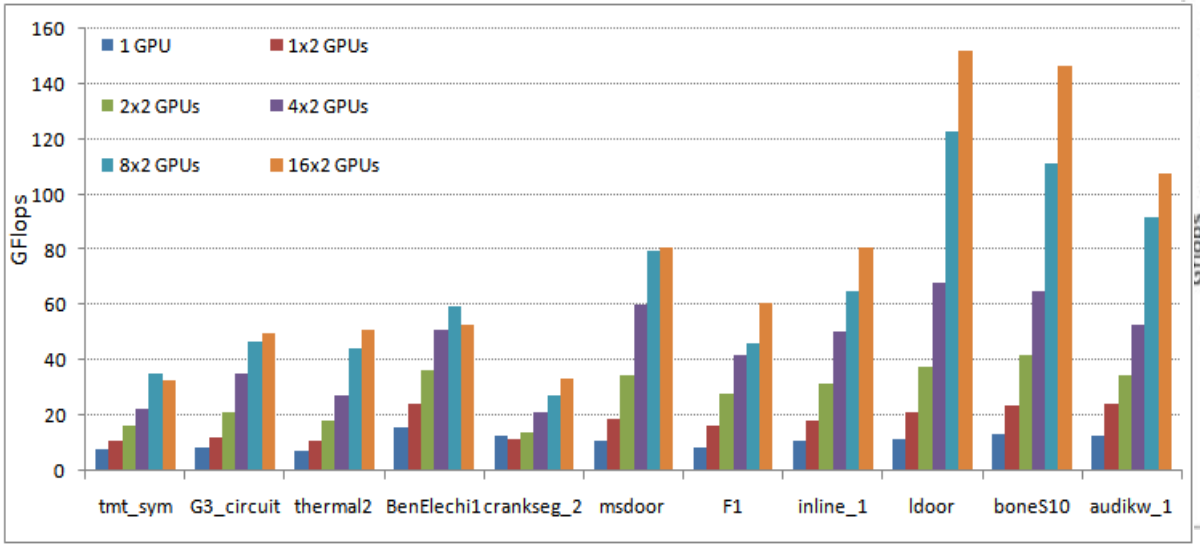
Conjugate Gradient Solver on a Multi-GPU Cluster

(A. Cevahir, A. Nukada, S. Matsuoka) [ICCS09 and follow on]

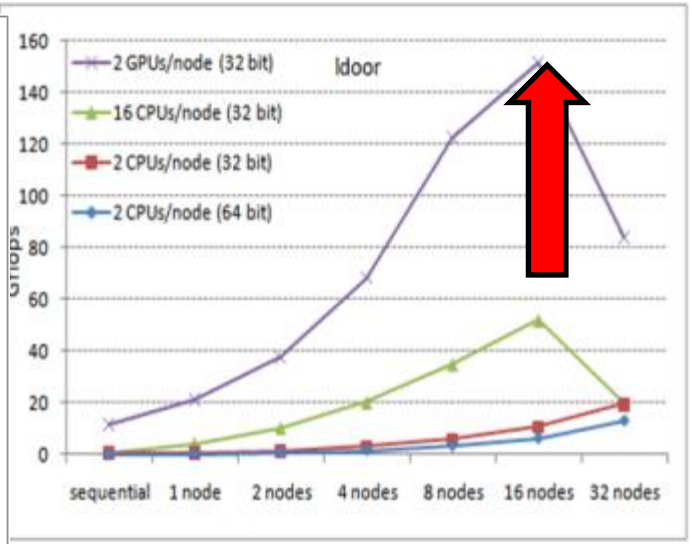
- Hypergraph partitioning for reduction of communication
  - GPU computing is fast, communication bottleneck is more severe
- All matrix and vector operations are implemented on GPUs
- Auto-selection of MxV kernel. GPUs may run different kernel

152GFlops on 32 NVIDIA GPUs on TSUBAME
(c.f. NPB CG ~3 GF on 32 node TSUBAME)

Performance comparisons over well-known matrices



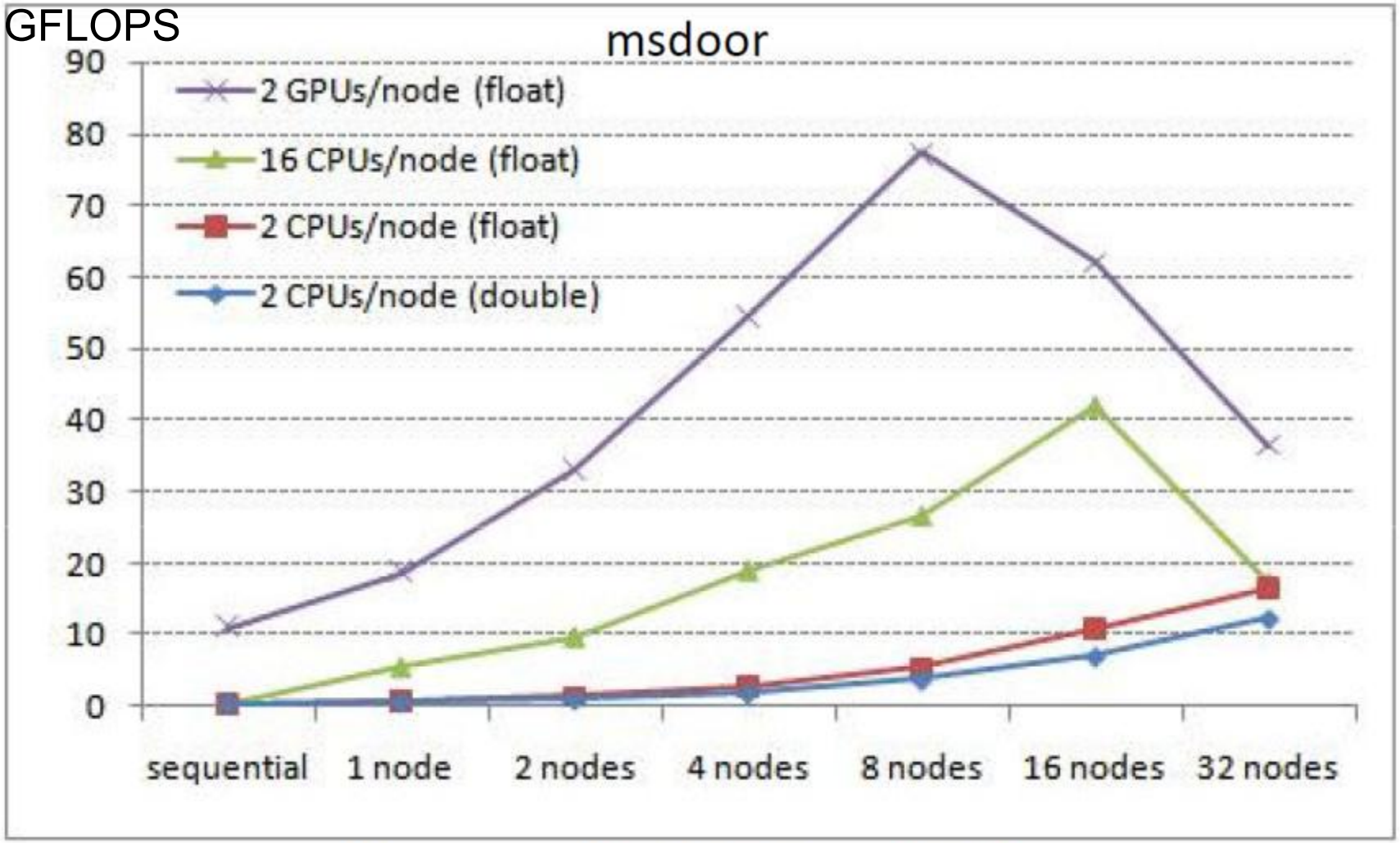
GPUs vs CPUs on TSUBAME
(Strong scaling)



Approximately x50-100 power efficient due to GPU + algorithmic improvement

Fast CG Result (1)

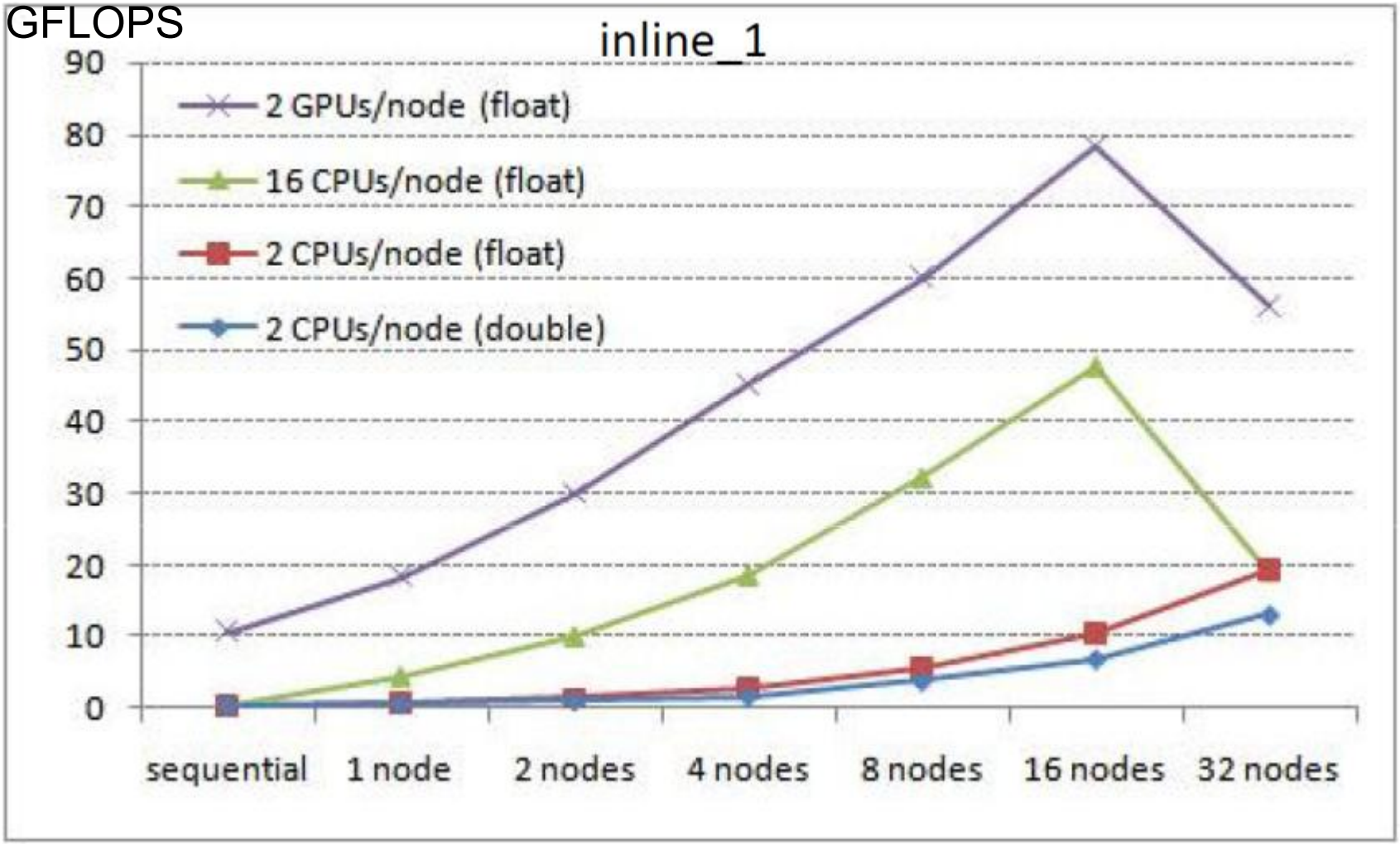
***Strong Scaling ***



Nonzeros: 20,240,935 n: 415,863

Fast CG Result (2)

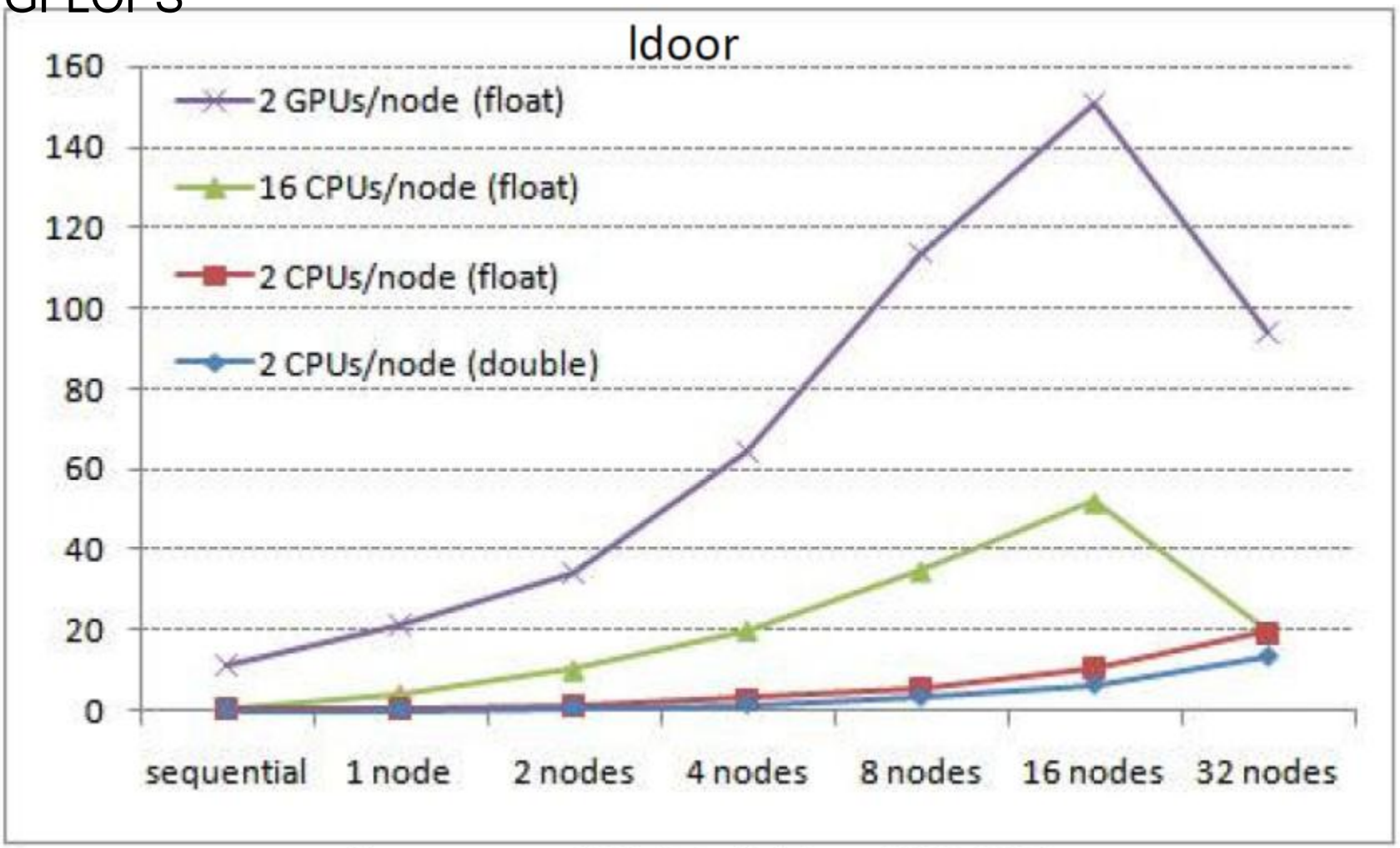
***Strong Scaling ***



Nonzeros: 36,816,342 n: 503,712

Fast CG Result (3)

***Strong Scaling ***



Nonzeros: 46,522,475 n: 952,253

(Faster Than) Real-time Tsunami Simulation

(Prof. Takayuki Aoki, Tokyo Tech.)

ADPC : Asian Disaster Preparedness Center

Early Warning System:

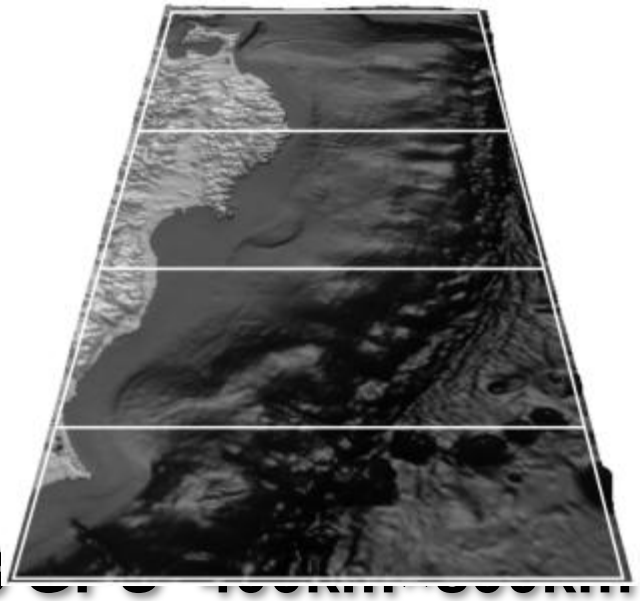
Data Based Extrapolation → high accuracy → (Faster than) Real-time CFD

Shallow-Water Equation

Conservative Form:

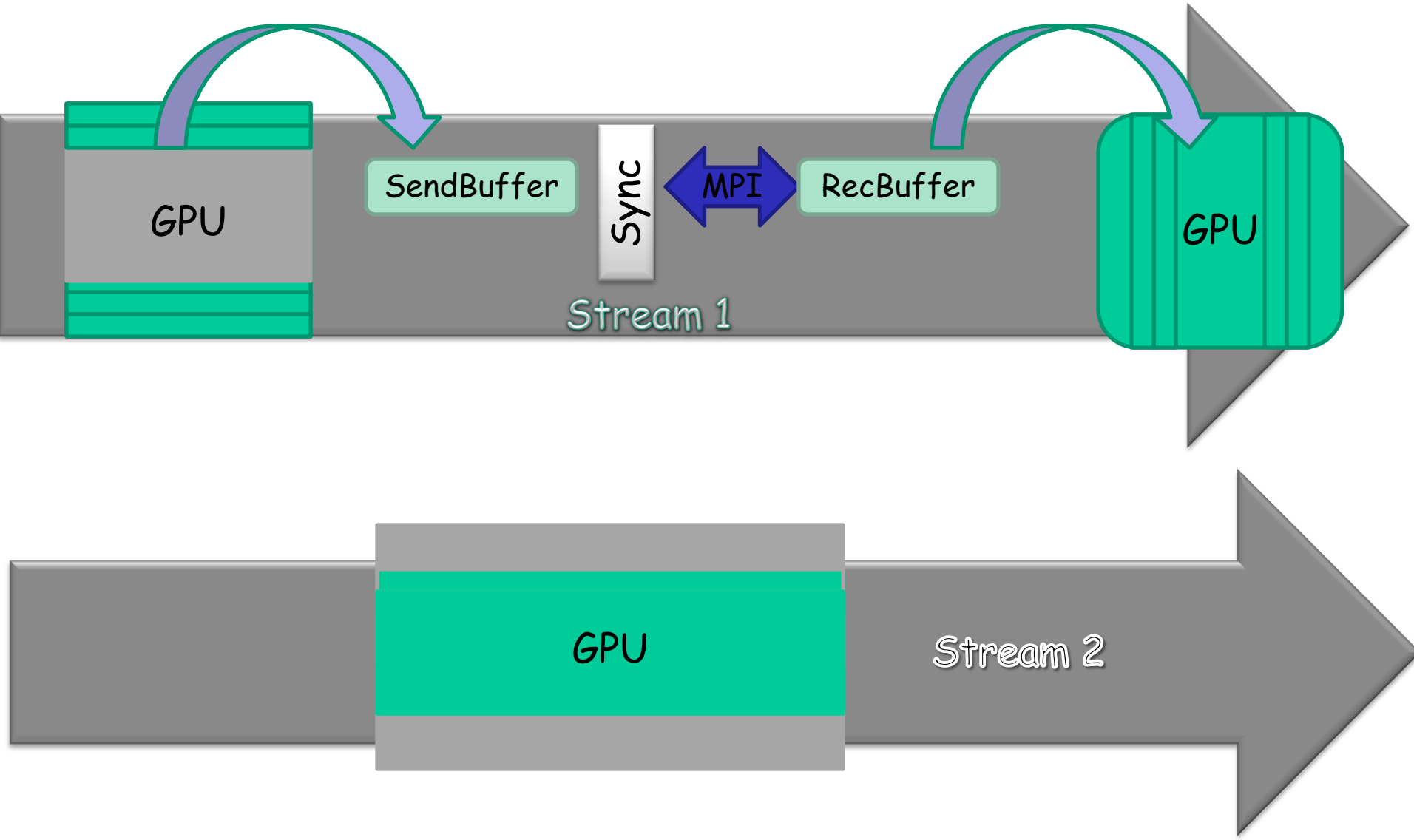
Assuming hydrostatic balance in the vertical direction,

3D → 2D equation



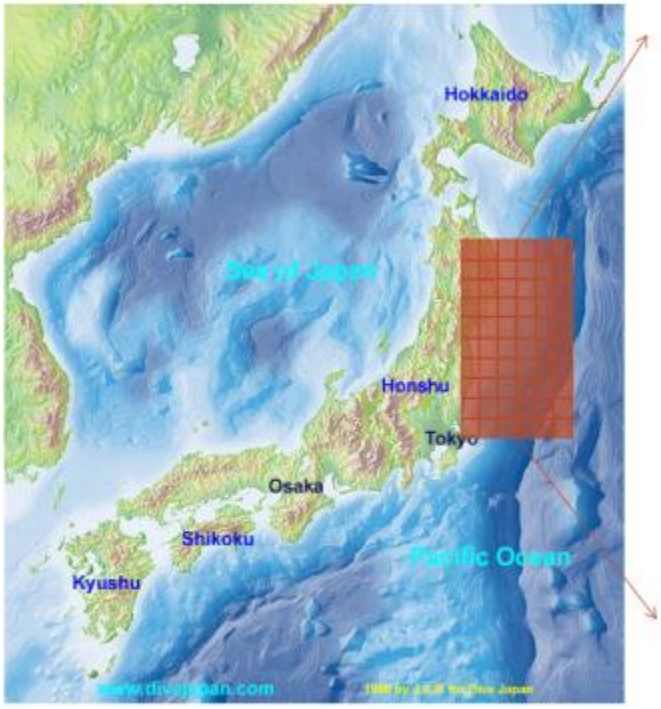
8 (100m mesh)

Asynchronous Streams

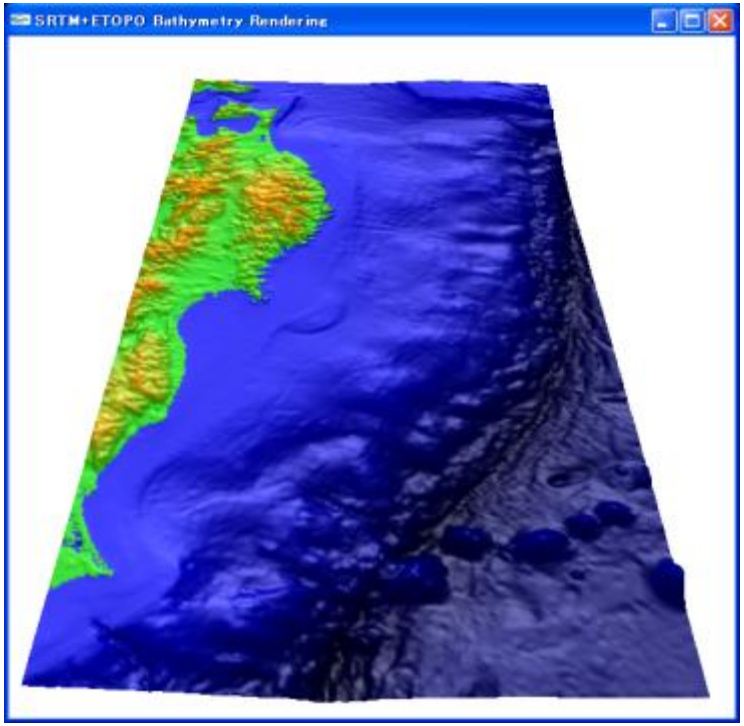


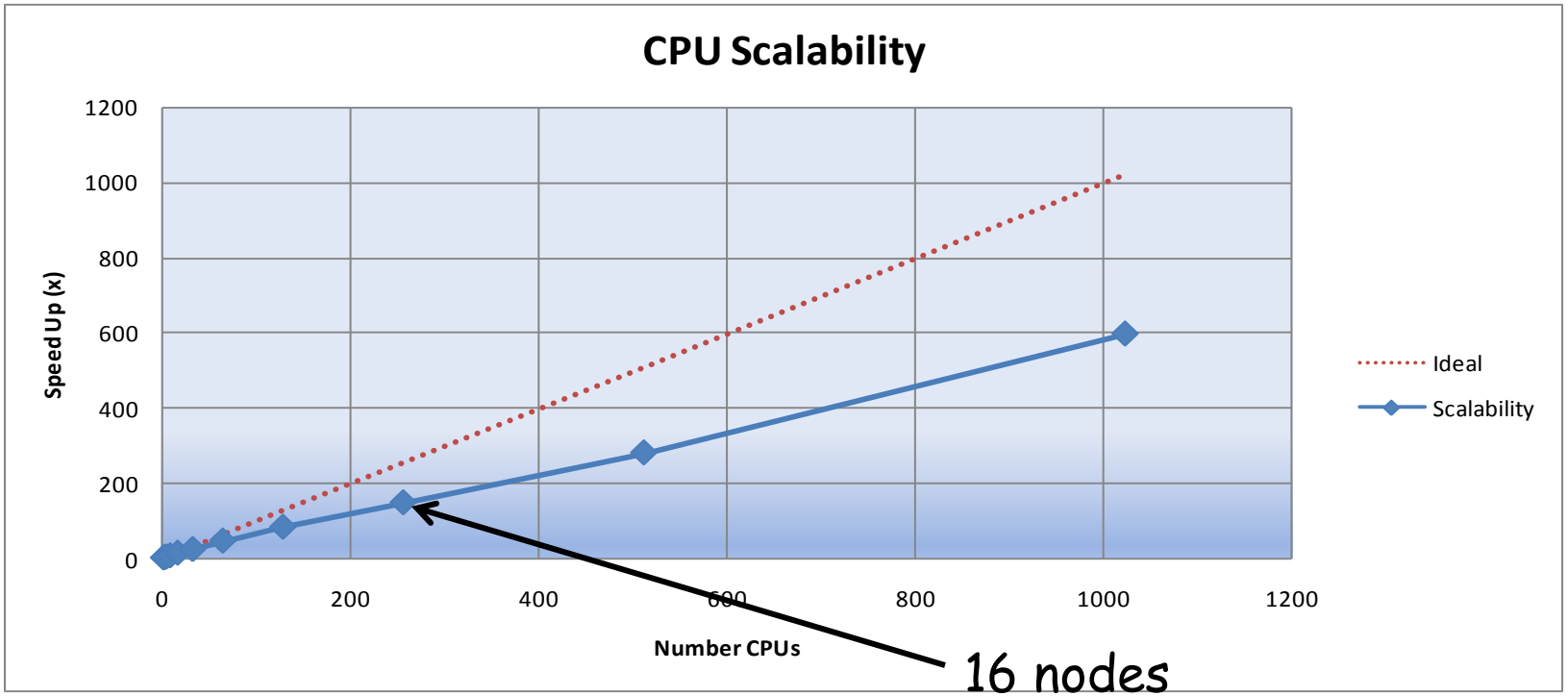
Tsunami Prediction of Northern Japan Pacific Coast

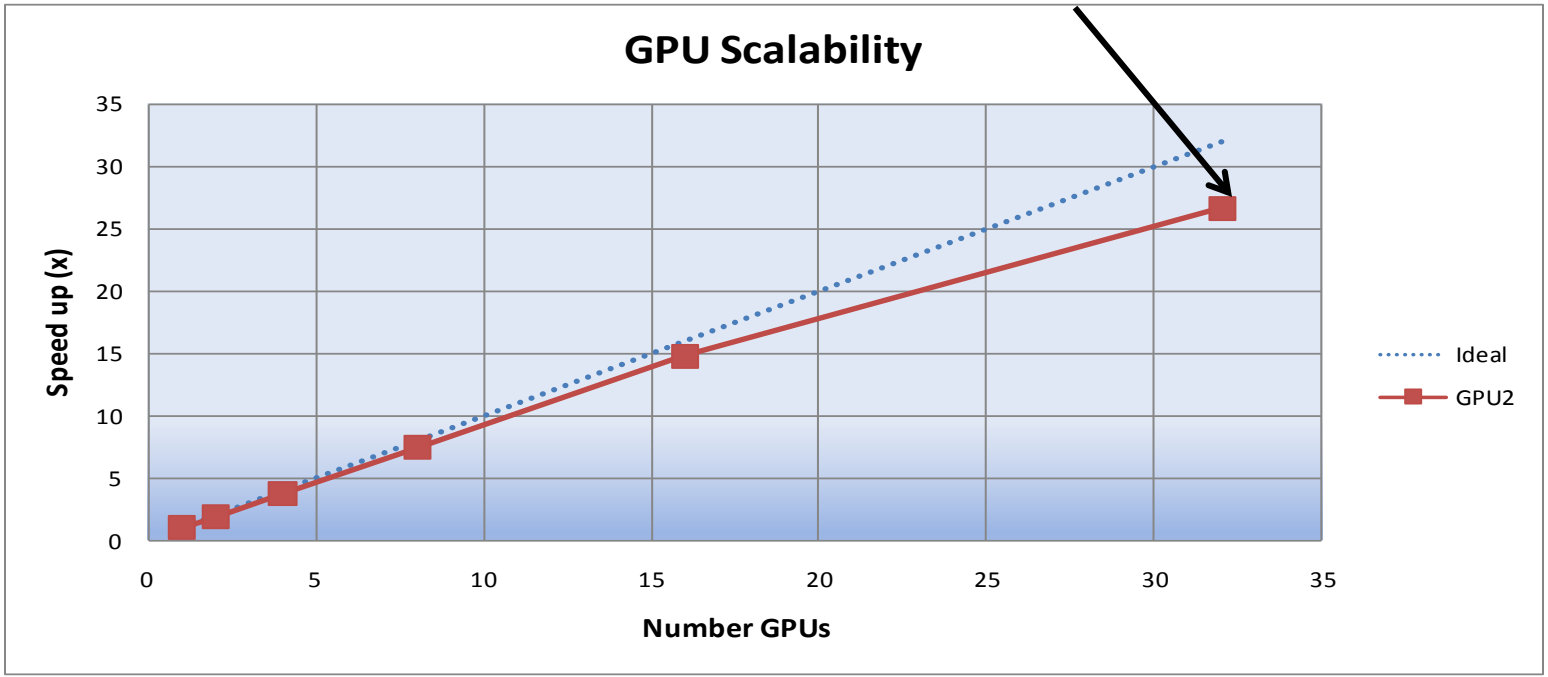
- Bathymetry



- Grid Size
  4096x8192
- Latitude
  $N35° - N42°$
- Longitude
  $E140°30' - E144°18'$
- Length
  370km x 740km



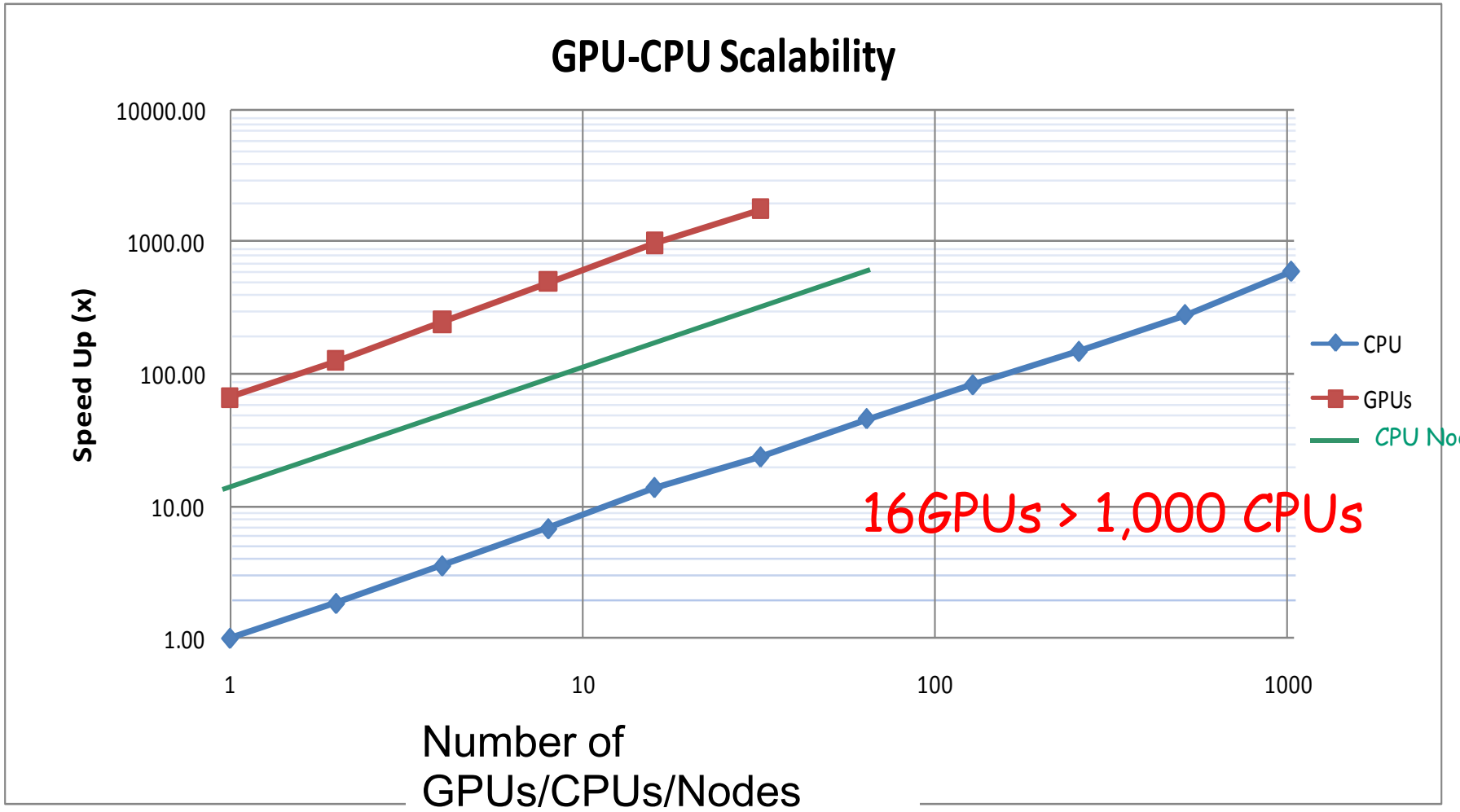

CPU Scalability
Speed Up (x)
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
Number CPUs
Ideal
Scalability
16 nodes


GPU Scalability
Speed up (x)
35
30
25
20
15
10
5
0
0
5
10
15
20
25
30
35
Number GPUs
Ideal
GPU2

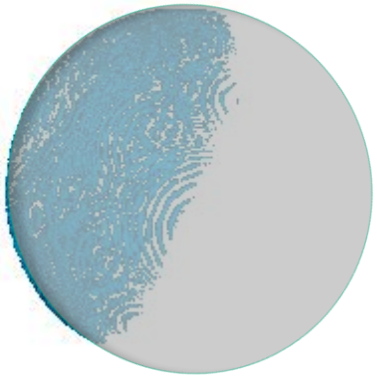
Multi-node CPU/GPU Comparison

*** Strong Scaling ***

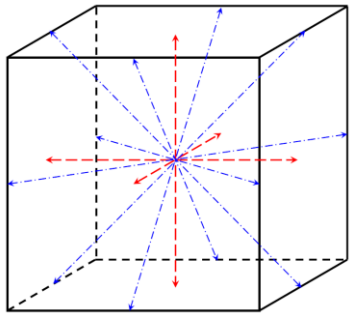
- Results on TSUBAME1.2



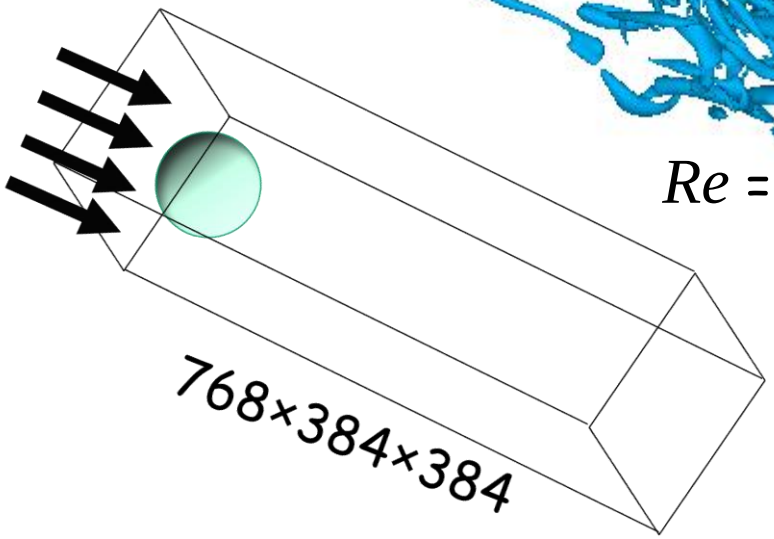
Lattice Boltzman Method on Multi-GPUs [Aoki etl al.]



$$\frac{\partial f_i}{\partial t} + \mathbf{e}_i \cdot \nabla f_i = -\frac{1}{\lambda}\left(f_i - f_i^{eq}\right)$$

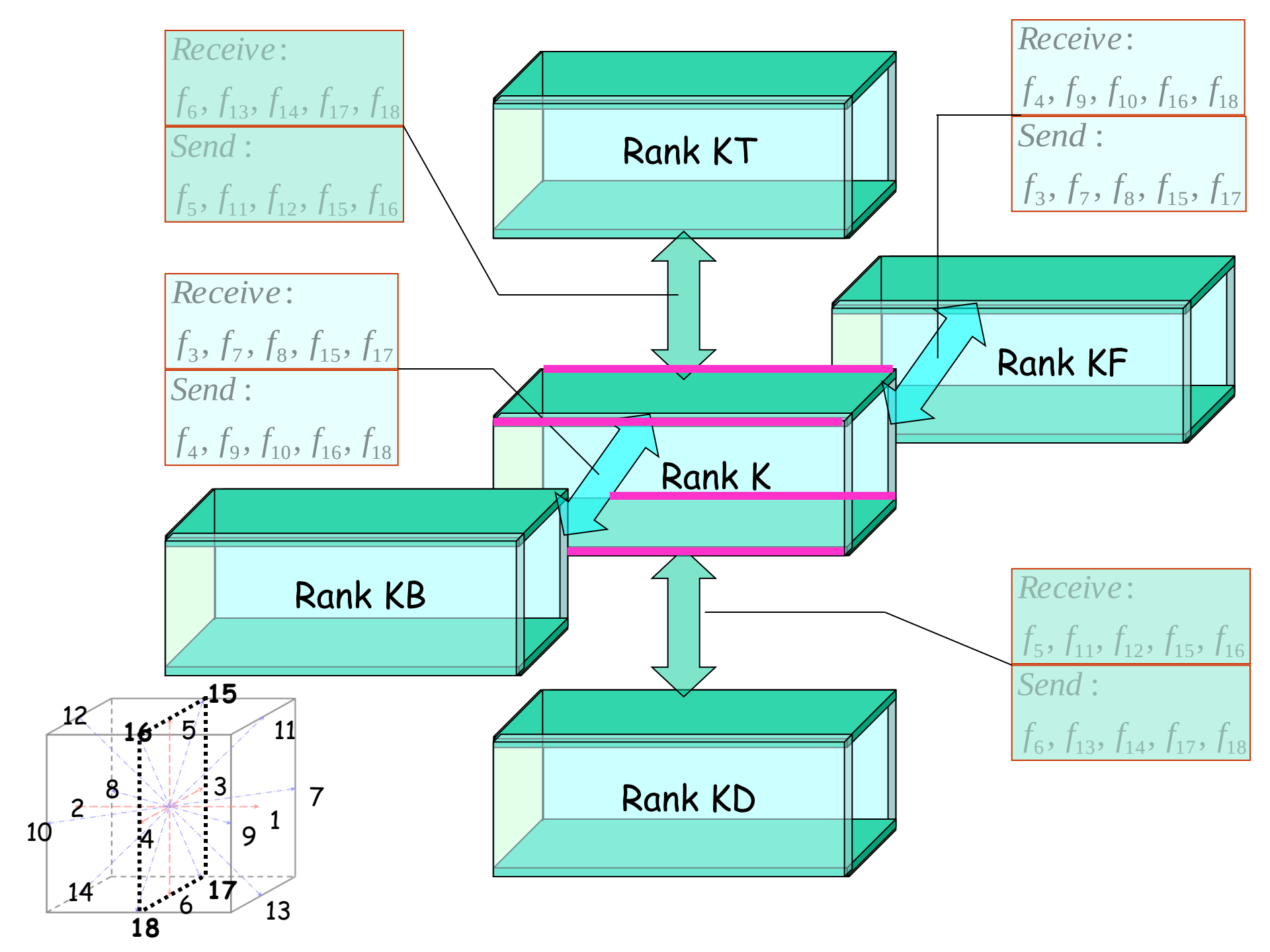


Uo

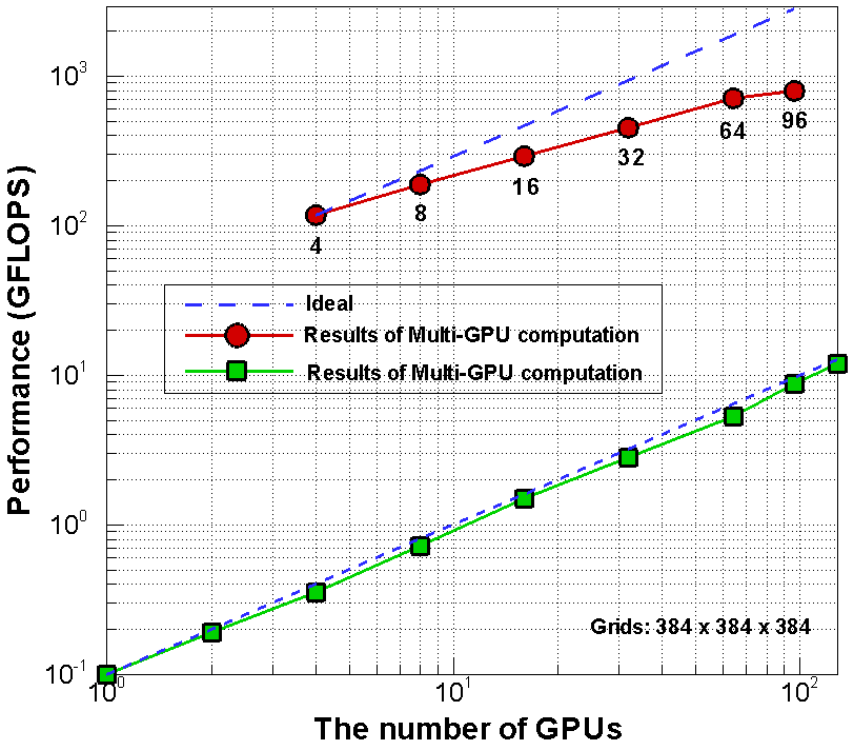


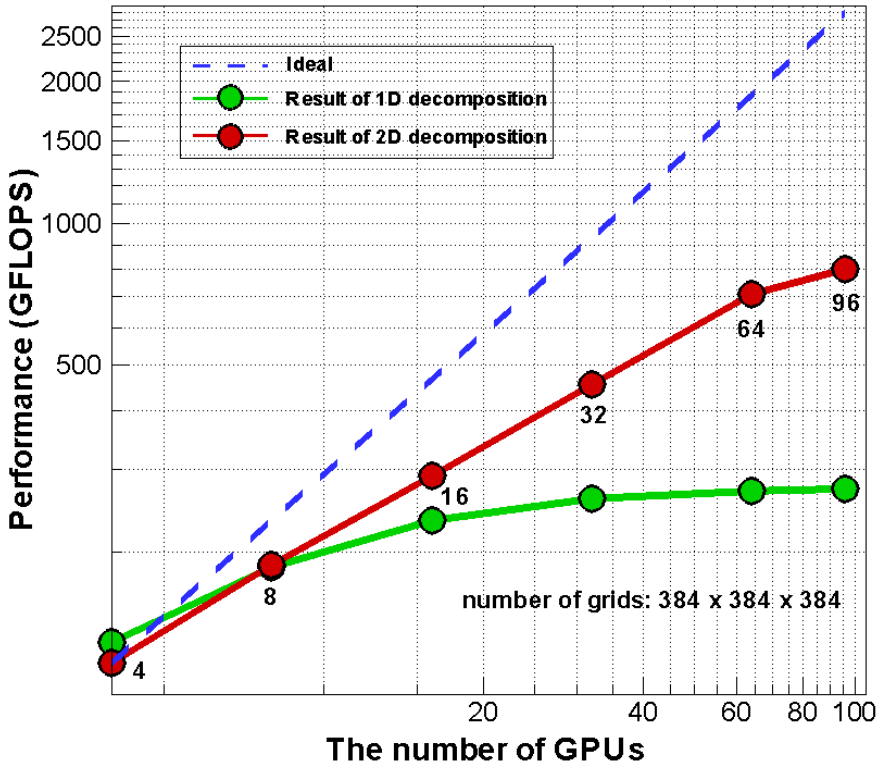
Re = 3000

Tesla S1070 64 GPU


Receive :
$f_6, f_{13}, f_{14}, f_{17}, f_{18}$
Send :
$f_5, f_{11}, f_{12}, f_{15}, f_{16}$
Rank KT
Receive :
$f_4, f_9, f_{10}, f_{16}, f_{18}$
Send :
$f_3, f_7, f_8, f_{15}, f_{17}$
Rank KF
Receive :
$f_3, f_7, f_8, f_{15}, f_{17}$
Send :
$f_4, f_9, f_{10}, f_{16}, f_{18}$
Rank K
Rank KB
Receive :
$f_5, f_{11}, f_{12}, f_{15}, f_{16}$
Send :
$f_6, f_{13}, f_{14}, f_{17}, f_{18}$
Rank KD
12
15
16
5
11
8
3
7
2
1
10
4
9
14
17
6
13
18

384 x 384 x 384 Lattice Boltzmann Scaling on Multi-GPUs on TSUBAME1.2





Comparison between the performances of multi-CPU and multi-GPU computations

Phase Field モデルによる　純金属の樹枝状凝固

Allen-Cahn方程式に基づく、異方性を考慮した3次元Phase Fieldモデル

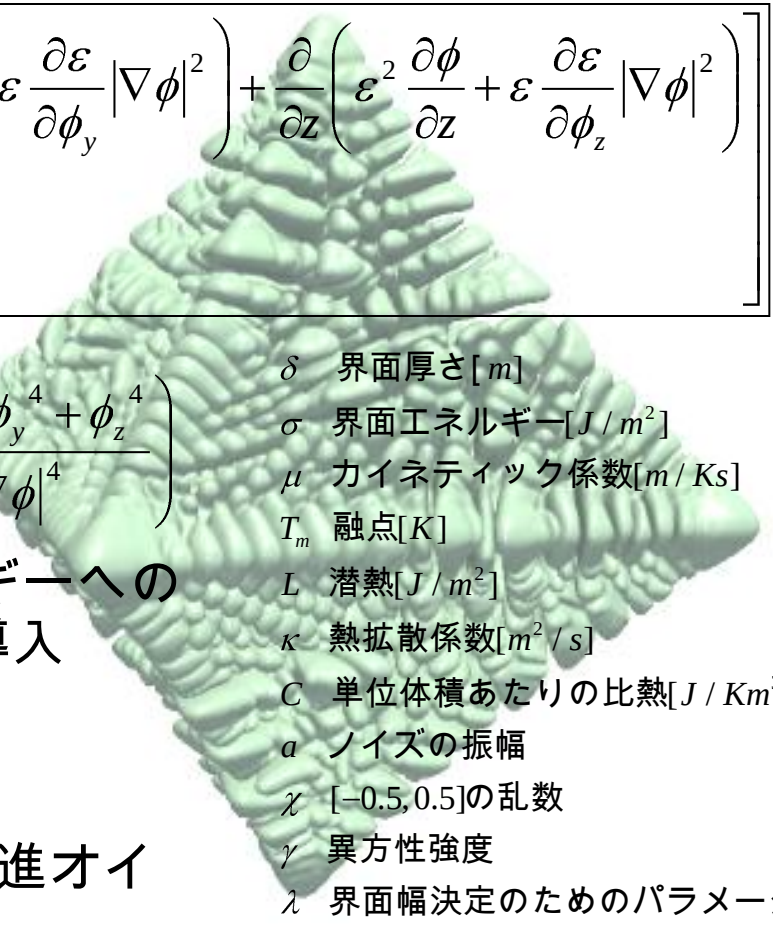
$$\frac{\partial \phi}{\partial t} = M\left[\frac{\partial}{\partial x}\left(\varepsilon^2\frac{\partial \phi}{\partial x}+\varepsilon\frac{\partial \varepsilon}{\partial \phi_x}|\nabla\phi|^2\right)+\frac{\partial}{\partial y}\left(\varepsilon^2\frac{\partial \phi}{\partial y}+\varepsilon\frac{\partial \varepsilon}{\partial \phi_y}|\nabla\phi|^2\right)+\frac{\partial}{\partial z}\left(\varepsilon^2\frac{\partial \phi}{\partial z}+\varepsilon\frac{\partial \varepsilon}{\partial \phi_z}|\nabla\phi|^2\right)\right.$$
$$\left.+4W\phi(1-\phi)\left\{\phi-\frac{1}{2}+\beta+a\chi\right\}\right]$$

$$\beta = -\frac{15L}{2W}\frac{T-T_m}{T_m}\phi(1-\phi) \qquad \varepsilon = \bar{\varepsilon}\left(1-3\gamma+4\gamma\frac{\phi_x^4+\phi_y^4+\phi_z^4}{|\nabla\phi|^4}\right)$$

熱伝導方程式

$$\frac{\partial T}{\partial t} = k\nabla^2 T + \frac{L}{C}30\phi^2(1-\phi)\frac{\partial \phi}{\partial t}$$

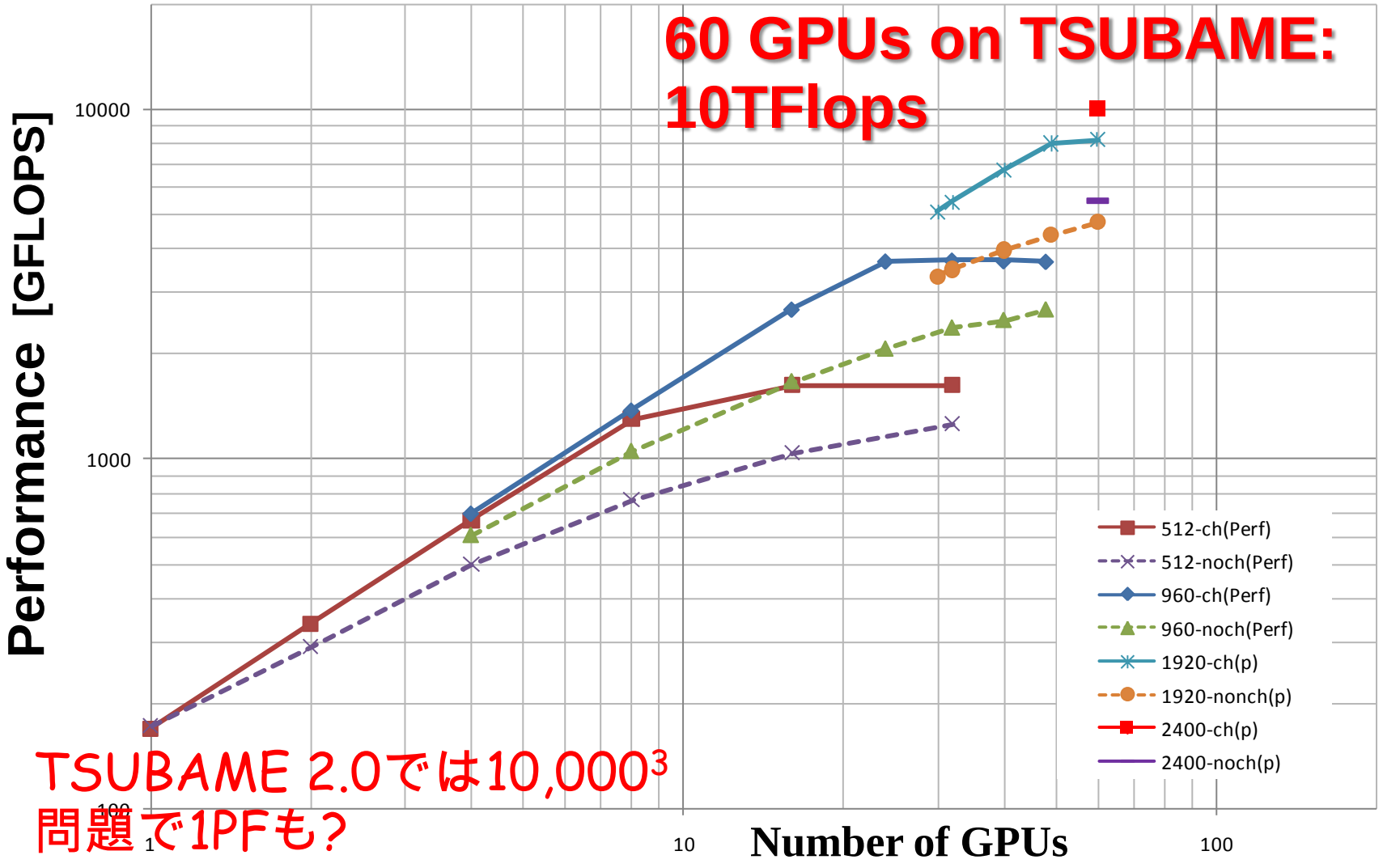
↑　界面エネルギーへの異方性の導入



- δ　界面厚さ$[m]$
- σ　界面エネルギー$[J/m^2]$
- μ　カイネティック係数$[m/Ks]$
- T_m　融点$[K]$
- L　潜熱$[J/m^2]$
- κ　熱拡散係数$[m^2/s]$
- C　単位体積あたりの比熱$[J/Km^3]$
- a　ノイズの振幅
- χ　$[-0.5, 0.5]$の乱数
- γ　異方性強度
- λ　界面幅決定のためのパラメータ

離散化：　2次中心差分法・時間積分 1次前進オイラー法

Scalability - 1 GPU/node- in Multi-GPU, Phase Field (New Ver. Code)

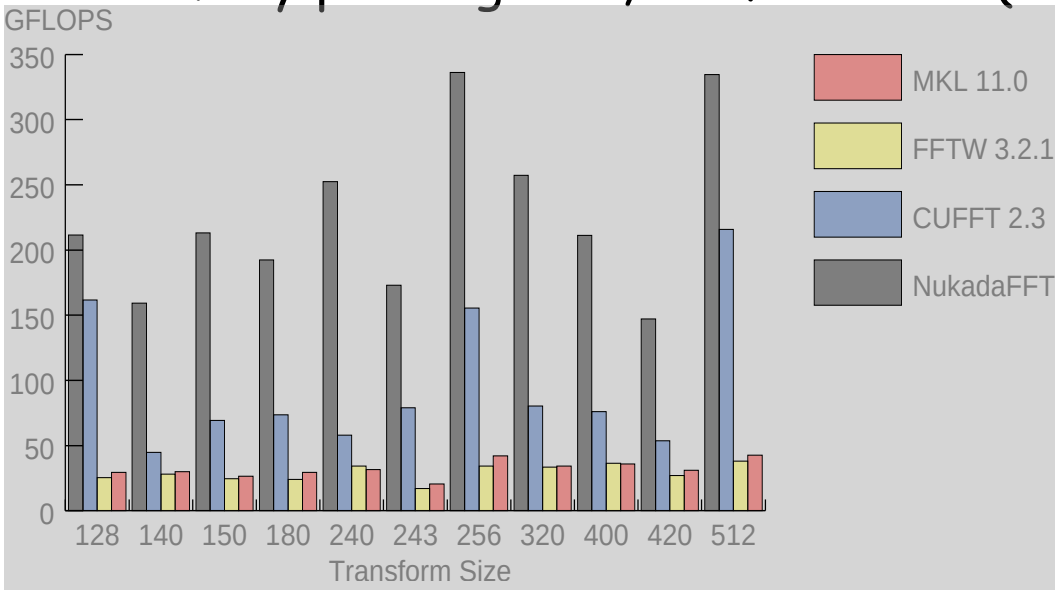


TSUBAME 2.0では$10{,}000^3$問題で1PFも?

Auto-Tuning FFT for CUDA GPUs [SC09]

HPC libraries should support GPU variations:

- # of SPs/SMs, Core/memory clock speed,
- Device/shared memory size, Property of memory bank, etc…
- Auto-tuning of various parameters!
- Selection of kernel radices (FFT-specific)
- Memory padding size, # of threads (GPU-specific)

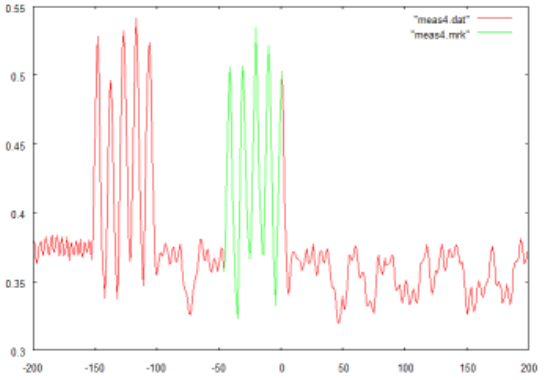


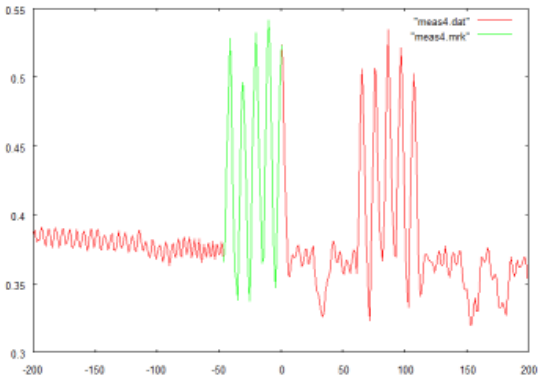
Our auto-tuned library is
- x4 power efficient than libraries on CPUs
- x1.2 -- 4 faster than vendor's library
- RELEASE NOV. 2009(!)

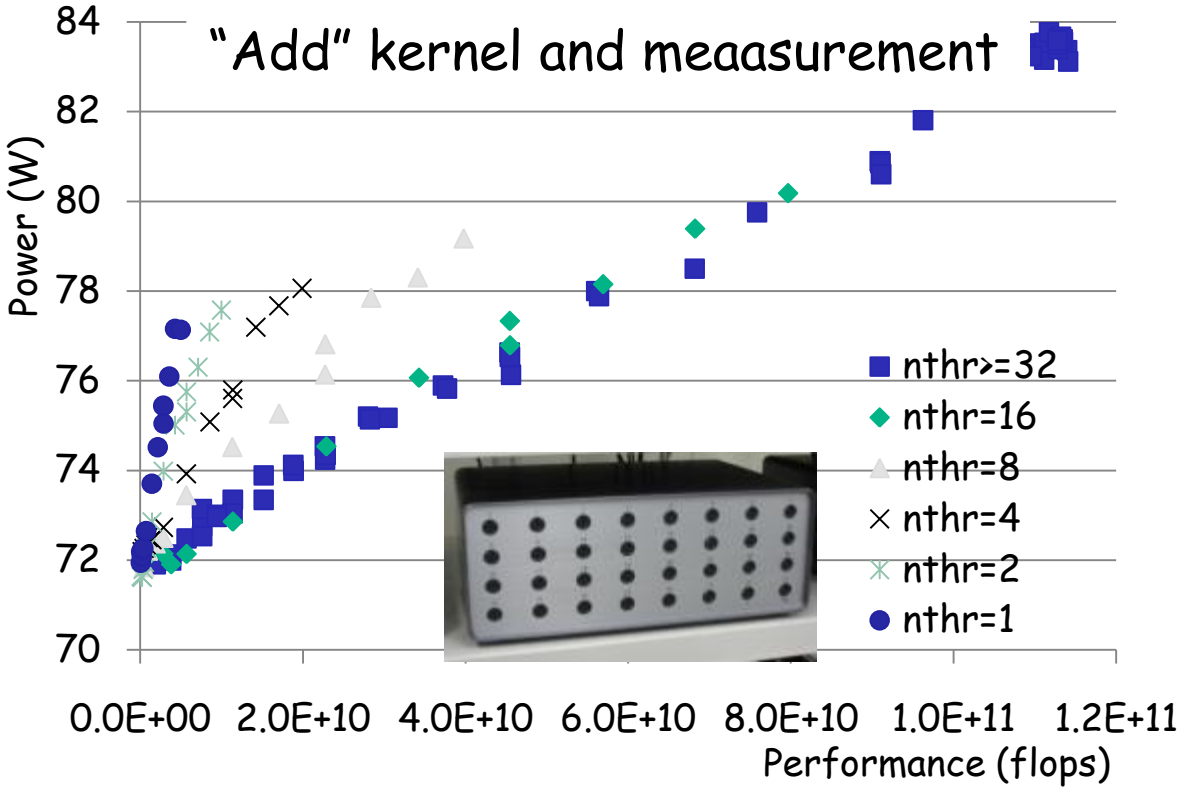
High Precision GPU Power Measurement (Reiji Suda, U-Tokyo, ULPHPC)

- "Marker" = Compute + Data transfer
- Automatically detect Markers
- Sample 1000s execution in 10s seconds

マーカーの自動検出例







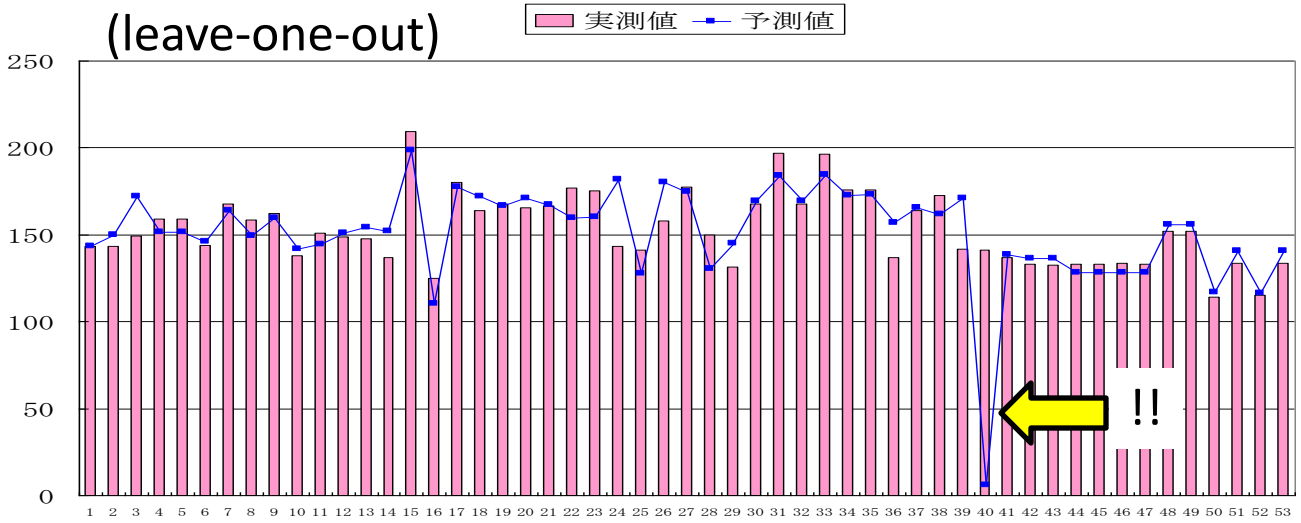
GPU Power Consumption Modeling

Employ learning theory to predict GPU power from performance counters

$$\text{GPU power} = \Sigma\, c_i \times p_i$$

p_i: GPU perf. Counter (12 types)、 c_i: learning constant

54 GPU kernels: prediction vs. measurement (leave-one-out)



Average error 7%

- Do need to compensate for extraneously erroneous kernel (!!)

TODO: Fewer couters for real-time prediction higher-precition, non-linear modeling

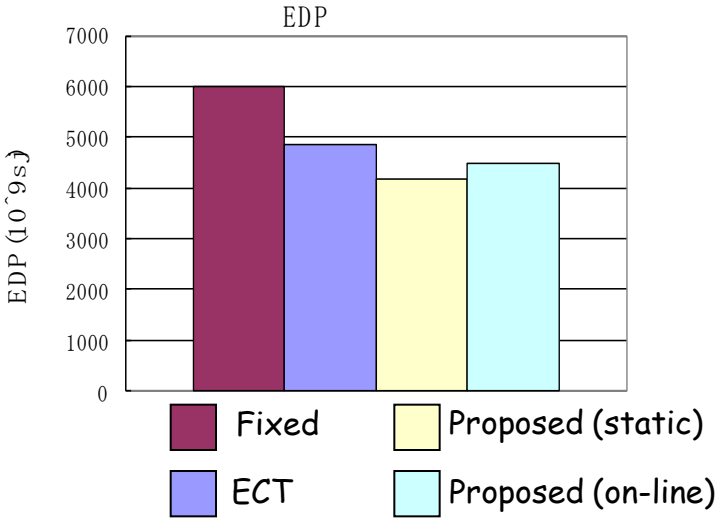
Low Power Scheduling in GPU Clusters

[IEEE IPDPS-HPPAC 09]

Objective : Optimize CPU/GPU Heterogeneous

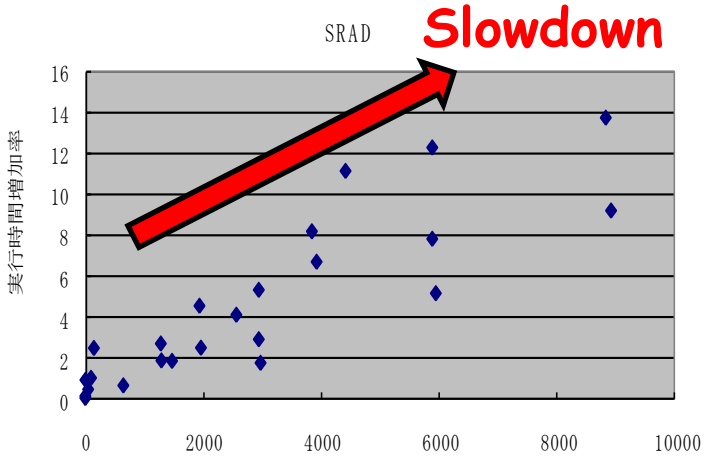
- Optimally schedule mixed sets of jobs executable on either CPU or GPU but w/different performance & power
- Assume GPU accel. factor (%) known

  30% Improvement Energy-Delay Product



TODO: More realistic environment

- Different app. power profile
- PCI bus vs. memory conflict
  - GPU applications slow down by 10 % or more when co-scheduled with memory-intensive CPU app.



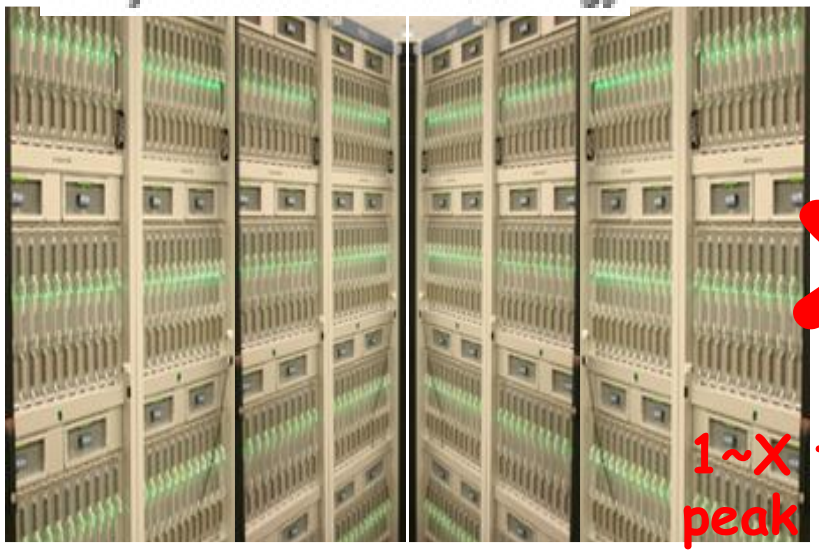
Game Changing Cluster Design for 2010 and Beyond

- Multi-petascale clusters should be built with:
  - Fat, teraflop-class, compute dense, high memory BW *vector processors* for single node scalability
  - Multithreaded shared memory processors to hide latency and limit memory capacity per node GPU
  - High but modest bandwidth, low latency, nearly-full bisection network for intra-node scalability
  - High-bandwidth node memory and I/O channels to accommodate all of above
  - Node-wise non-volatile silicone storage for scalable storage I/O
  - Software layers for attaining bandwidth, fault tolerance, programmability, and low power
  - Such an architecture is the basis towards Exascale

Tokyo Tech. TSUBAME2.0 Oct 2010

東京工業大学
Tokyo Institute of Technology

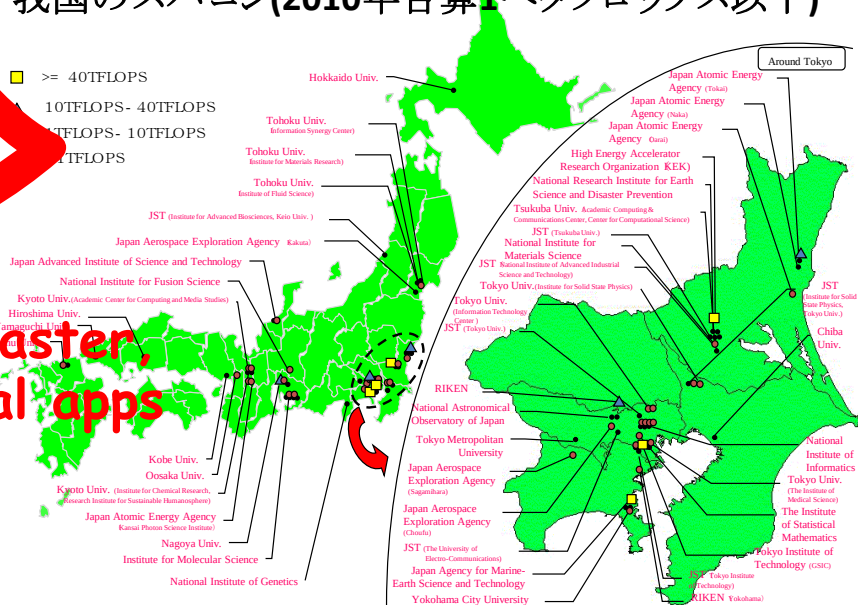
GSIC
Global Scientific Information and Computing Center



1~X times faster, peak and real apps

Entire Japan

我国のスパコン(2010年合算1ペタフロップス以下)



~3 Pflop CPU+GPU (DFP)
~PB/s Memory BW
~PB SSD, 0.5~1TB/s I/O BW
Several hundred Tb/s Bisection
~65 Racks, incl. Storage (200m2)
1MW Operational Power
Dynamic provisioning of Windows HPC + Linux

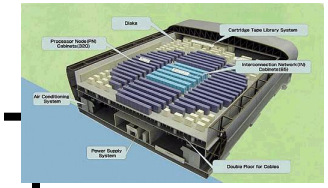
2010 Total < 1PF
60 sites
> $300 million

Highlights of TSUBAME 2.0 Design (Oct. 2010)

- Next gen multi-core x86 + next gen GPU
  - 1 CPU + 1 GPU ratio (or more), several thousands nodes
  - Massively Parallel Vector multithreading for scalability
- Near petabyte/s aggregate mem BW,
  - restrained memory capacity
- Modest IB-QDR BW, full bisection BW
- Flash memory per node, ~Petabyte, ~Terabyte/s I/O
- Low power & efficient cooling, comparable to TSUBAME 1.0 (despite ~x40 speedup)
- Virtualization and Dynamic Provisioning of Windows HPC + Linux, job migration, etc.

TSUBAME 2.0 Performance

Earth Simulator ⇒ TSUBAME 4years x 40 Downsizing



10PF

1PF

100TF

10TF

1TF

2002

2004

2006

2008

2010

2012

TSUBAME ⇒ TSUBAME 2.0 x40 downsizing again?

TSUBAME2.0 >3 PF (2010-2H)

Japanese NLP >10PF(2012)

US >10P (2011~12)

US/EU Petascales (Peak) (2008)

US NSF/DoE (2009 2H)

U-Tokyo, Kyoto-U, Tsukuba T2K 61-140TF (2008)

Others

BlueGene/L 360TF(2005)

TSUBAME 1.2 Upgrade (3Q2008) 170TF

TSUBAME 1.1 Upgrade 111TF, 1.6 PB, 128GB nodes (1Q2008)

TSUBAME 1.0 80TF (1Q2006)

Earth Simulator 40TF (2002)

Titech Campus Grid

1.3TF

KEK 59TF BG/L+SR11100