Collaborative Research between DoE Labs and Tokyo Tech GSIC o Extreme Scale Computing - Success Stories

Satoshi Matsuoka
Professor
Global Scientific Information and Computing (GSIC) Center
Tokyo Institute of Technology
Fellow, Association for Computing Machinery (ACM) & ISC

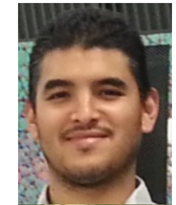
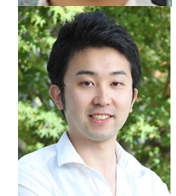
AICS Symposium
AICS-Riken, Kobe Japan
20160222

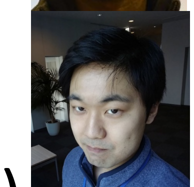
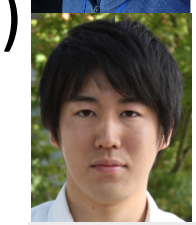
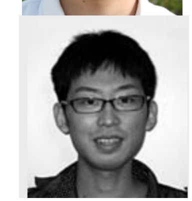
Successful Model of DoE Lab / Tokyo Tech Collaboration

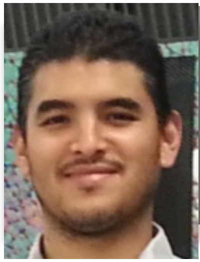
- 1. Initial agreement on collaboration area w/DoE group
  - Funding on both sides not mandated but desirable
- 2. Send a Ph.D. ~~guinea pig~~ student for short-term (2mo) exploratory ~~hard labor~~ internship
- 3. Usually Tokyo Tech student performs extremely well => tangible collaborative research advance
- 4. Student asked back for longer-term (6 mo or greater) ~~more hard labor~~ internship
- 5. Papers published, OSS deliverables, awards, …
- 6. Student obtains Ph.D. => hired as postdoc at DoE Lab (much higher salary than being hired in Japan!)

Tokyo Tech Collaboration Topics with DoE Labs in the recent years

- Exascale Resiliece (Leonardo Bautista-Gomez@ANL, Kento Sato@LLNL)
- Performance of OpenMP-MPI Hybrid Programming on Many-Core (Abdelhalim Amer@ANL)
- Performance Visualization (Kevin Brown@LLNL)
- Performance Modeling of Tee Code with ASPEN (Keisuke Fukuda@ORNL)
- Large-Scale Graph Store in NVM (Keita Iwabuchi@LLNL)
- OpenACC Data Layout Extensions (Tetsuya Hoshino@ORNL)
- More to come…



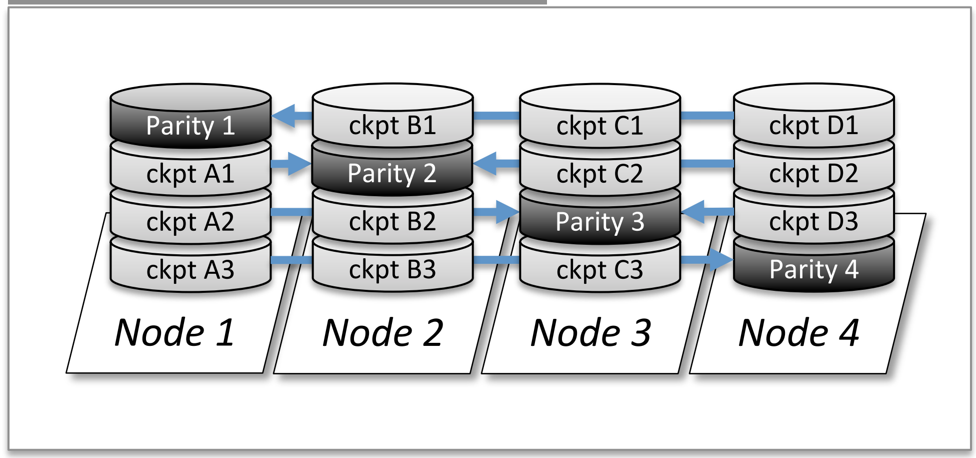
FTI: High Performance Fault Tolerance Interface

[SC11, EuroPar12 & Cluster12 (Leonardo Bautista-Gomez et al.)]
Internship at ANL => PostDoc at ANL

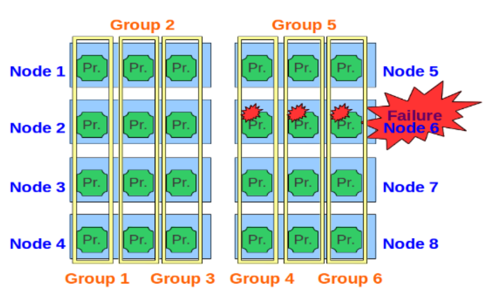
API

- Diskless checkpoint:
  - Create redundant data across local storages on compute nodes using a encoding technique such as Reed-solomon, XOR
    - Scalable by using distributed disks
  - Can restore lost checkpoints on a failure caused by small # of nodes like RAID-5

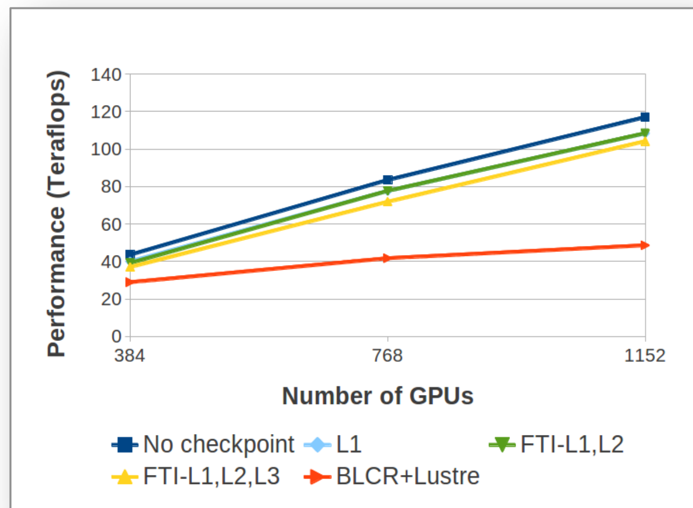
Diskless checkpointing



Diskless checkpoint runtime library using Reed-Solomon encoding



- FTI implements a scalable Reed-Solomon encoding algorithm by utilizing local storages such as SSD
- FTI analyzes the topology of the system and create encoding clusters that increase the resilience



FTI (Multilevel checkpointing)

- FTI is a multilevel checkpointing library with 4 levels of reliability. It has over 8000 lines of c/c++ (with Fortran bindings) under GPL2.1.
- Download at **http://www.github.com/leobago/fti** and you can access the documentation at **http://leobago.github.io/fti**
- FTI discovers the location of the processes in the hardware and creates topology-aware virtual rings to enhance reliability.
- FTI can protect dynamic datasets, where the size, pointers or structure of the dataset changes during the runtime.
- FTI offers the option to dedicate one process per node for fault tolerance to minimize the checkpoint overhead.
- While using dedicated processes for asynchronous tasks FTI allows the user to do a fine-grained selection about the tasks to offload.
- While using dedicated processes, FTI splits the global communicator and returns a new communicator to isolate the FT-dedicate ranks.
- FTI monitors the timestep length and can dynamically adapt the checkpointing interval during runtime, keeping a consistent state.
- Applications ported: HACC, CESM (ice module), LAMMPS, GYSELA5D, SPECFEM3D (CUDA version), HYDRO.

API and code example

Local Storage: SSD, PCM, NVM.
Fastest checkpoint level.
Low reliability, transient failures.

Partner Copy: Ckpt. Replication.
Fast copy to neighbor node.
It tolerates single node crashes.

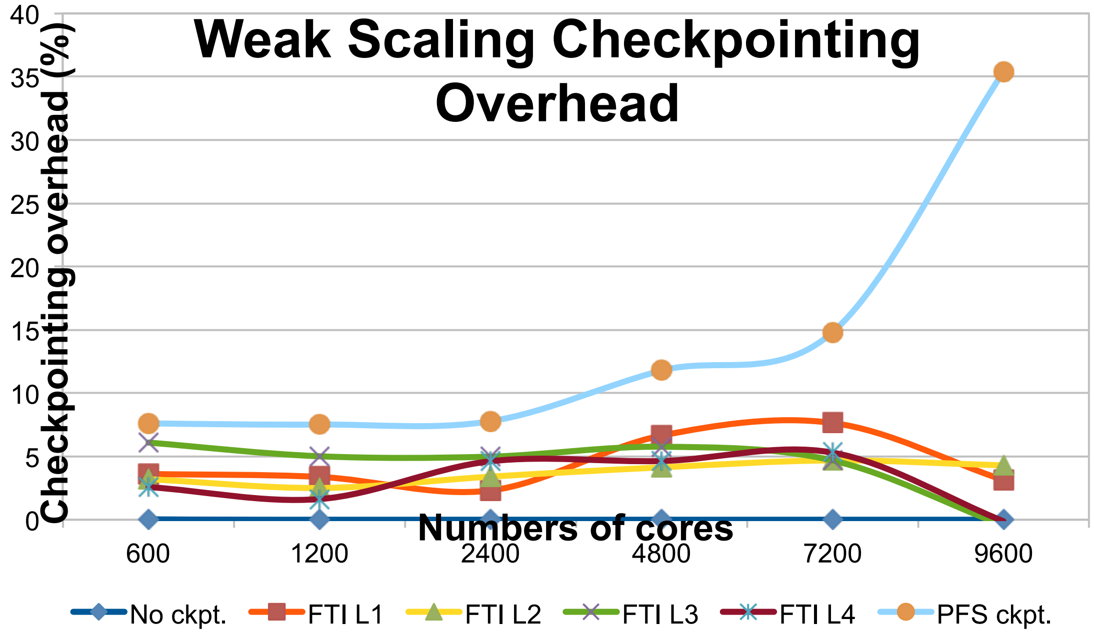
RS Encoding: Ckpt. Encoding.
Slow for large checkppoints.
Reliable, multiple node crashes.

File System: Classic Ckpt.
Slowest of all levels.
The most reliable. Power outage.

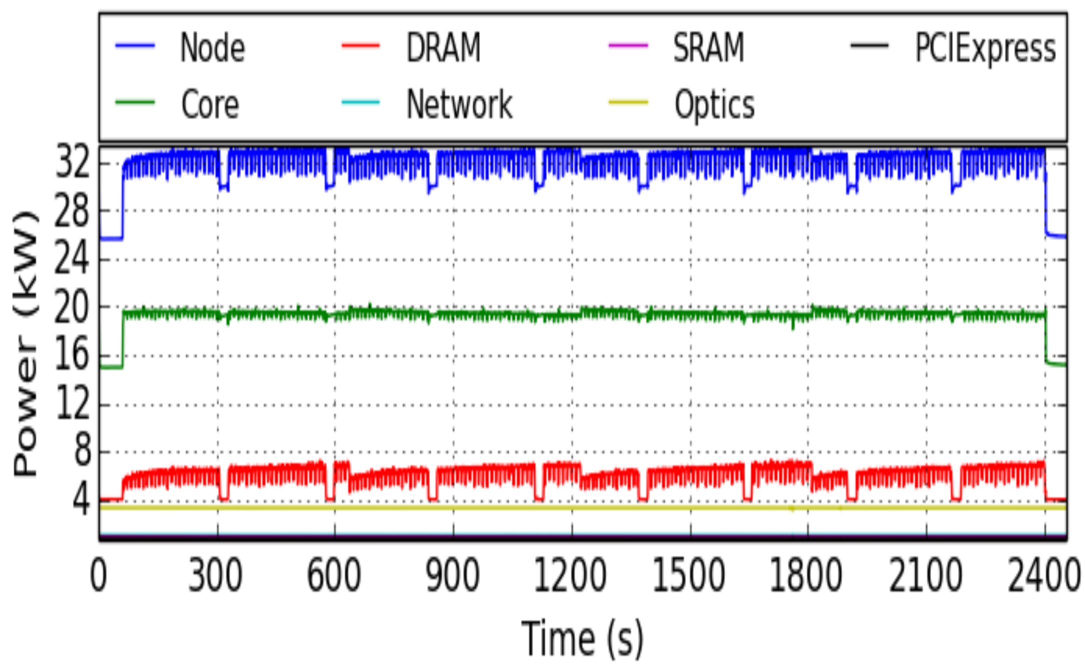
```
int main(int argc, char **argv) {

  MPI_Init(&argc, &argv);
  FTI_Init("conf.fti", MPI_COMM_WORLD);

  double *grid;
  int i, steps=500, size=10000;
  initialize(grid);
  FTI_Protect(0, &i,   1,   FTI_INTG);
  FTI_Protect(1, grid, size,FTI_DFLT);

  for (i=0; i<steps; i++) {
    FTI_Snapshot();
    kernel1(grid);
    kernel2(grid);
    comms(FTI_COMM_WORLD);
  }

  FTI_Finalize();
  MPI_Finalize();
  return 0;
}
```

FTI scaling

- **Weak scaling to ~10k proc.**
- CURIE supercomputer in France
- SSD on the compute nodes
- HYDRO scientific application
- Checkpointing every ~6 minutes

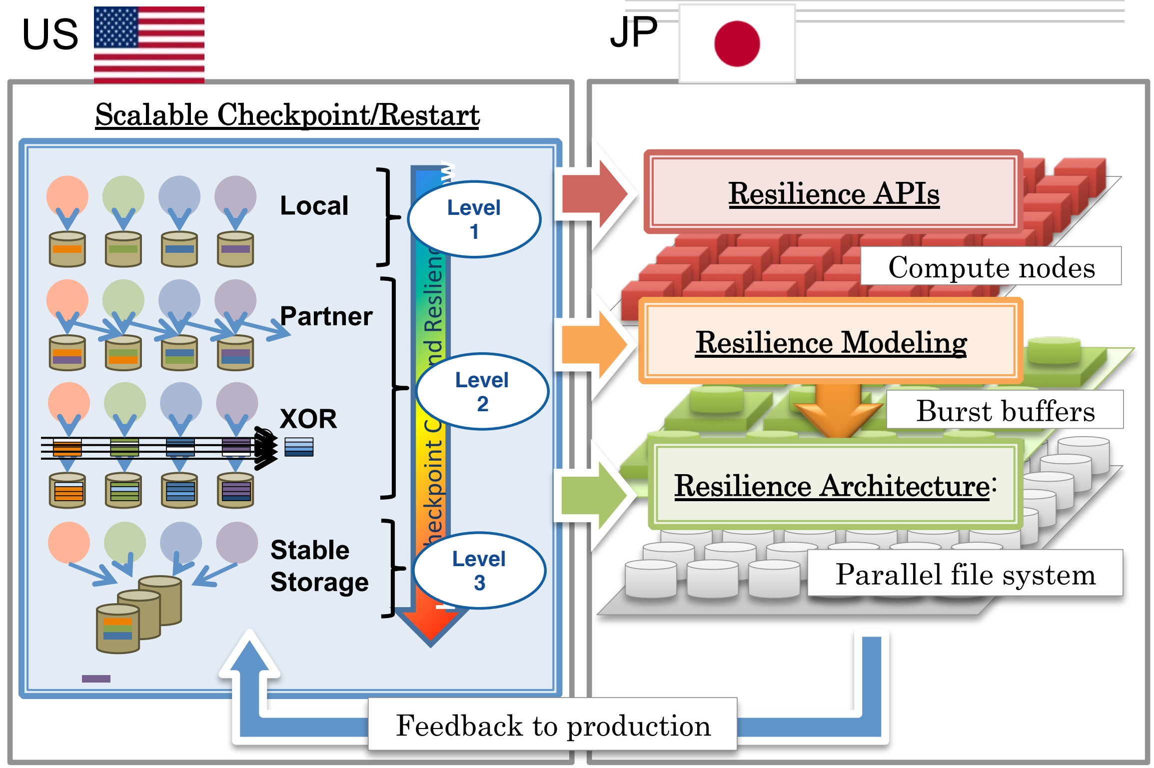


- **Weak scaling on MIRA (BG\Q)**
- LAMMPS, Lennard-Jones simulation of 1.3 billion atoms
- 512 nodes, 64 MPI processes per node (32,678 processes)
- Power monitoring and checkpoint every ~5 minutes
- Less than 5% overhead on time to completion

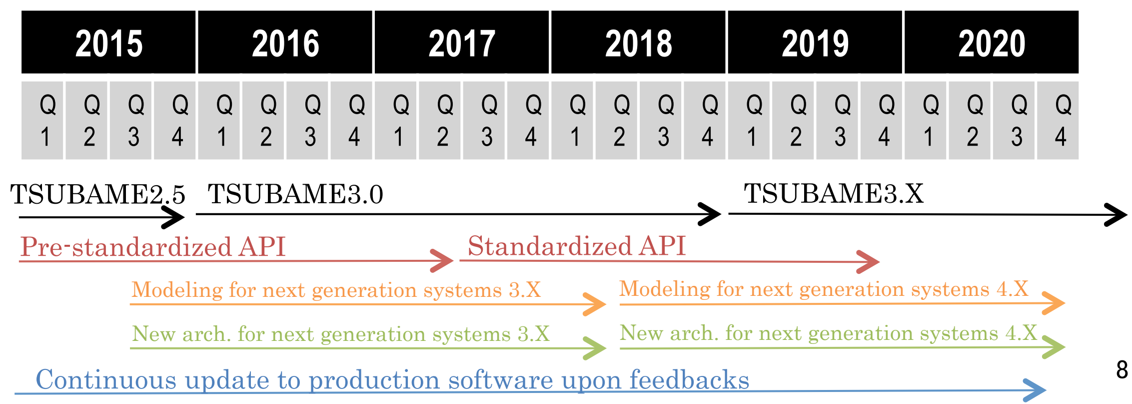


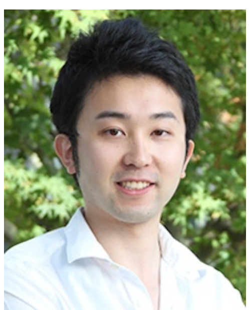
Extreme-Scale Resilience for Billion-Way Parallelism

- Coordinators
  - US: Kento Sato, Kathryn Mohror, Adam Moody, Todd Gamblin, Bronis R. de Sipinski (LLNL)
  - JP: Satoshi Matsuok (Tokyo Tech), Naoya Maruyama (RIKEN)
- Description
  - The Tokyo Tech group creates resilience APIs for transparent and fast recovery, resilience modeling for optimizing environment, and resilience architecture for scalable and reliable checkpoint/restart, then feeds back to SCR, the production resilience library developed at LLNL. The production library will be deployed in TSUBAME3.0
- How to collaborate
  - Biweekly meeting
  - Student / young researchers exchange
- Deliverables
  - Pre-standardization of Resilience API
  - Production resilience interface, SCR



- Schedule (DRAFT)





Kento Sato
LLNL Internship
Now LLNL PostDoc

FMI: Fault Tolerant Messaging Interface

[IPDPS2014, Kento Sato et al.]

Example code & Evaluation

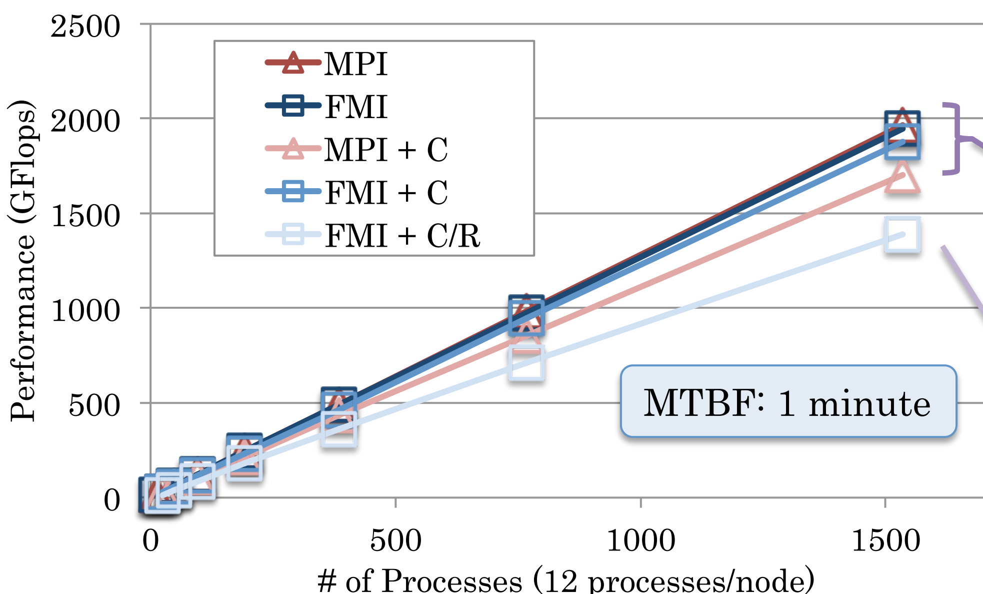
FMI example code

```
int main (int *argc, char *argv[]) {
  FMI_Init(&argc, &argv);
  FMI_Comm_rank(FMI_COMM_WORLD, &rank);
  /* Application's initialization */
  while ((n = FMI_Loop(…)) < numloop) {
    /* Application's program */
  }
  /* Application's finalization */
  FMI_Finalize();
}
```

- FMI_Loop enables transparent recovery and roll-back on a failure
  - Periodically write a checkpoint
  - Restore the last checkpoint on a failure

P2P communication performance

	1-byte Latency	Bandwidth (8MB)
MPI	3.555 usec	3.227 GB/s
FMI	3.573 usec	3.211 GB/s

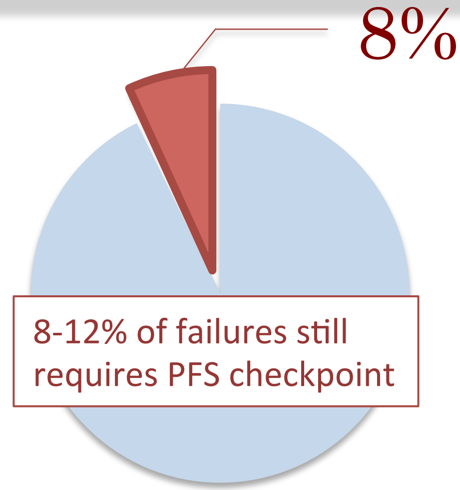


FMI directly writes checkpoints via memcpy, and can exploit the bandwidth

Even with the high failure rate, FMI incurs only a 28% overhead

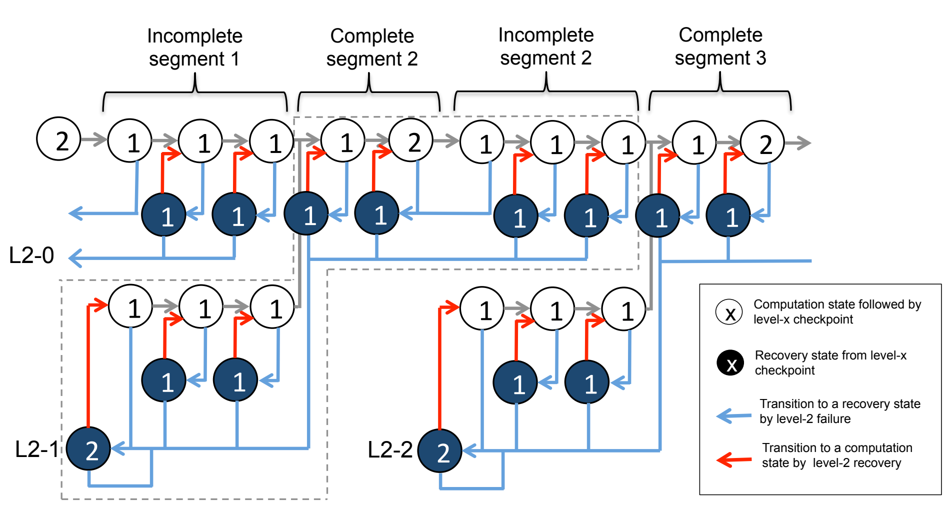
Design and Modeling of Async. Checkpointing
[SC12, Kento Sato et al.]

- **Objective**: Minimize checkpoint overhead to PFS
  - Minimize CPU usage, memory and network bandwidth
- **Proposed method**: Implementation and modeling Non-blocking checkpointing
  - Asynchronously write checkpoints to PFS through Staging nodes using RDMA
  - Determine the optimal checkpoint interval on the asynchronous checkpoint scheme

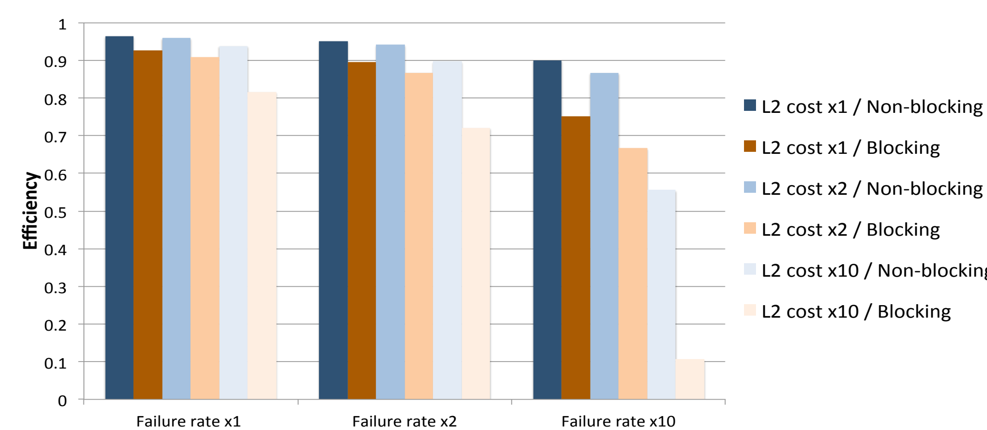


Failure analysis on TSUBAME2.0

Async. checkpointing model



90% of efficiency in most cases

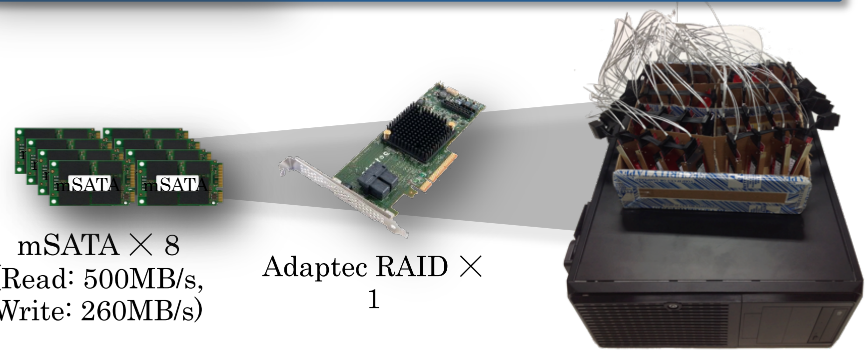


Burst Buffers for Resilient Checkpoint/Restart
[CCGrid2014 (Best Paper Award), Kento Sato et al.]

TSUBAME3.0 EBD Prototype mSATA High I/O BW, low power & cost

- Provide POSIX-like I/O interfaces
  - open, read, write and close
  - Client can open any files on any servers
- IBIO use ibverbs for communication between clients and servers
  - Exploit network bandwidth of infiniBand

CCGrid2014 Best Paper Award
(Kento Sato, Kathryn Mohror, Adam Moody, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama & Satoshi Matsuoka)
Lawrence Livermore National Laboratory



mSATA × 8
(Read: 500MB/s, Write: 260MB/s)

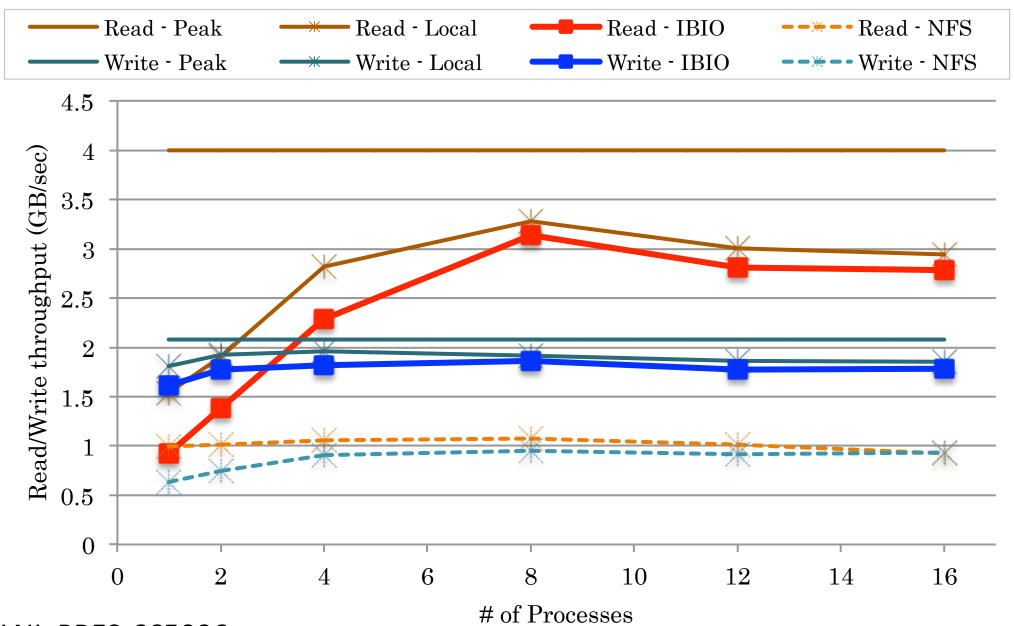
Adaptec RAID × 1

EBD I/O

Node specification

CPU	Intel Core i7-3770K CPU (3.50GHz x 4 cores)
Memory	Cetus DDR3-1600 (16GB)
M/B	GIGABYTE GA-Z77X-UD5H
SSD	Crucial m4 msata 256GB CT256M4SSD3 (Peak read: 500MB/s, Peak write: 260MB/s)
SATA converter	KOUTECH IO-ASS110 mSATA to 2.5' SATA Device Converter with Metal Fram
RAID Card	Adaptec RAID 7805Q ASR-7805Q Single

Interconnect :Mellanox FDR HCA (Model No.: MCX354A-FCBT)



Burst Buffers for Resilient Checkpoint/Restart

[CCGrid2014 (Best Paper Award), Kento Sato et al.]

Resilience modeling overview

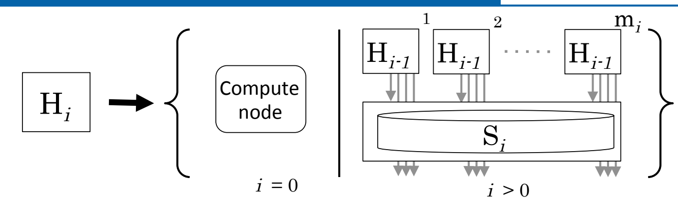
To find out the best checkpoint/restart strategy for systems with burst buffers, we model checkpointing strategies

C/R strategy model

$$O_i = \begin{cases} C_i + E_i & \text{(Sync.)} \\ I_i & \text{(Async.)} \end{cases} \qquad L_i = C_i + E_i$$

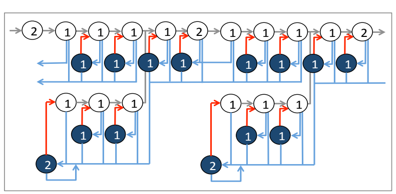
$$C_i \text{ or } R_i = \frac{\text{< C/R date size / node >} \times \text{<\# of C/R nodes per } S_i^* \text{>}}{\text{< write perf. (} w_i \text{) > or <read perf. (} r_i \text{) >}}$$

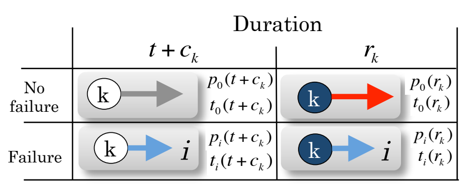
Recursive structured storage model



Storage Model: $H_N\{m_1, m_2, \ldots, m_N\}$

MLC model



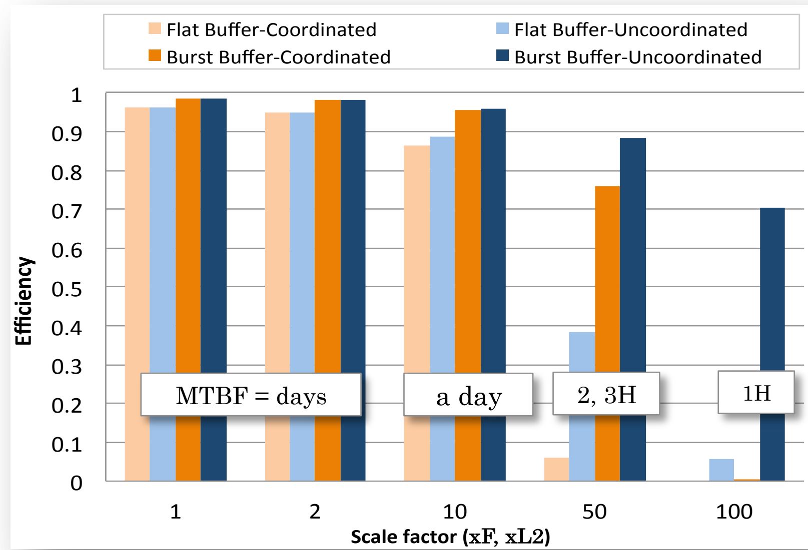


t : Interval
c_c : c-level checkpoint time
r_c : c-level recovery time
λ_i : i-level checkpoint time

$$p_0(T) = e^{-\lambda T}$$
$$t_0(T) = T$$
$$p_i(T) = \frac{\lambda_i}{\lambda}(1 - e^{-\lambda T})$$
$$t_i(T) = \frac{1 - (\lambda T + 1) \cdot e^{-\lambda T}}{\lambda \cdot (1 - e^{-\lambda T})}$$

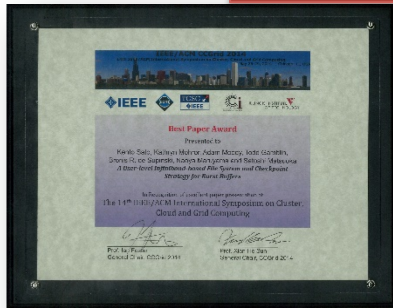
$p_0(T)$: No failure for T seconds
$t_0(T)$: Expected time when $p_0(T)$

$p_i(T)$: i - level failure for T seconds
$t_i(T)$: Expected time when $p_i(T)$

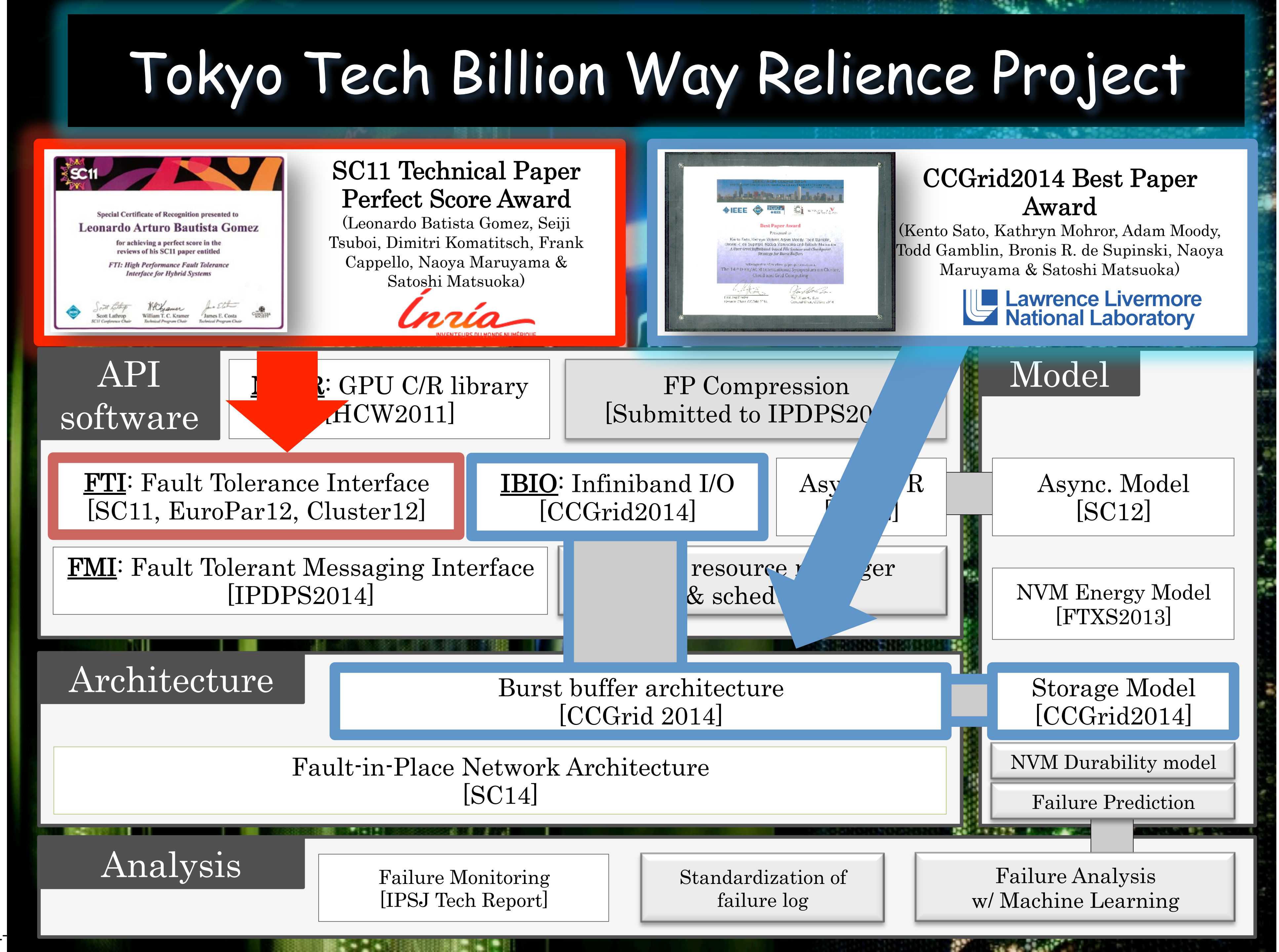


Publications

IEEE/ACM CCGrid2014
Best Paper Award



- Kento Sato, Kathryn Mohror, Adam Moody, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "A User-level InfiniBand-based File System and Checkpoint Strategy for Burst Buffers", In Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2014), Chicago, USA, May, 2014. **(Best Paper Award !!)**
- Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "FMI: Fault Tolerant Messaging Interface for Fast and Transparent Recovery", In Proceedings of the International Conference on Parallel and Distributed Processing Symposium 2014 (IPDPS2014), Phoenix, USA, May, 2014.
- Kento Sato, Satoshi Matsuoka, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski and Naoya Maruyama, "Burst SSD Buffer: Checkpoint Strategy at Extreme Scale", IPSJ SIG Technical Reports 2013-HPC-141, Okinawa, Sep, 2013
- Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Design and Modeling of a Non-blocking Checkpointing System", In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis 2012 (SC12), Salt Lake, USA, Nov, 2012.
- Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Towards a Light-weight Non-blocking Checkpointing System", In HPC in Asia Workshop in conjunction with the International Supercomputing Conference (ISC'12), Hamburg, Germany, June, 2012 (Poster)
- Kento Sato,Adam Moody,Kathryn Mohror,Todd Gamblin,Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Design and Modeling of a Non-Blocking Checkpoint System", In ATIP - A*CRC Workshop on Accelerator Technologies in High Performance Computing, Singapore, March, 2012. (Poster)
- Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Design and Modeling of an Asynchronous Checkpointing System", IPSJ SIG Technical Reports 2012-HPC-135 (SWoPP 2012), Tottori, Aug, 2012.
- Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Towards an Asynchronous Checkpointing System", IPSJ SIG Technical Reports 2011-ARC-197 2011-HPC-132 (HOKKE-19), Hokkaido, Nov, 2011.



Tokyo Tech Billion Way Relience Project
SC11 Technical Paper Perfect Score Award
(Leonardo Batista Gomez, Seiji Tsuboi, Dimitri Komatitsch, Frank Cappello, Naoya Maruyama & Satoshi Matsuoka)
Special Certificate of Recognition presented to
Leonardo Arturo Bautista Gomez
for achieving a perfect score in the reviews of his SC11 paper entitled
FTI: High Performance Fault Tolerance Interface for Hybrid Systems
Inria
CCGrid2014 Best Paper Award
(Kento Sato, Kathryn Mohror, Adam Moody, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama & Satoshi Matsuoka)
Lawrence Livermore National Laboratory
API software
: GPU C/R library
HCW2011]
FP Compression
[Submitted to IPDPS20
Model
FTI: Fault Tolerance Interface
[SC11, EuroPar12, Cluster12]
IBIO: Infiniband I/O
[CCGrid2014]
Async. Model
[SC12]
FMI: Fault Tolerant Messaging Interface
[IPDPS2014]
resource
& sched
NVM Energy Model
[FTXS2013]
Architecture
Burst buffer architecture
[CCGrid 2014]
Storage Model
[CCGrid2014]
NVM Durability model
Fault-in-Place Network Architecture
[SC14]
Failure Prediction
Analysis
Failure Monitoring
[IPSJ Tech Report]
Standardization of failure log
Failure Analysis w/ Machine Learning

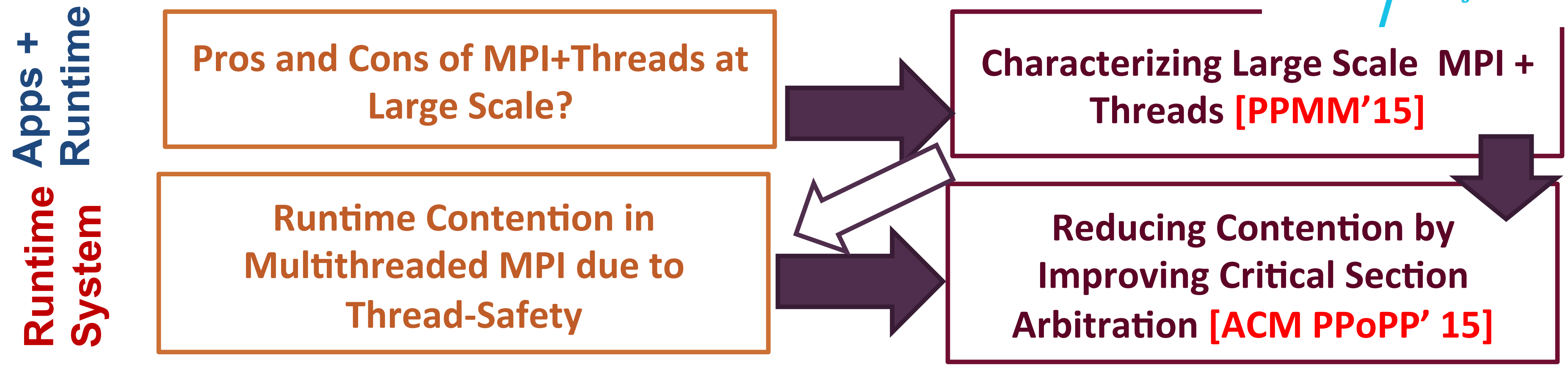
LLNL-

OpenMP-MPI Performance collaboration w/ANL - Abdelhalim Amer

Abdelhalim Amer (Halim)
Postdoctoral Researcher, ANL

- Visits
  - Abdelhalim Amer, PhD. Student at Tokyo Institute of Technology
  - Sept 2013 – Nov 2013 (Tokyo Tech → ANL)
    - Characterizing lock contention in multithreaded MPI applications
  - Nov 2013 – Apr 2013 (ANL → Tokyo Tech)
    - Develop hybrid MPI kernels relying on multithreaded communication
  - Apr 2014 - Sep2014 (Tokyo Tech → ANL)
    - Large scale analysis of hybrid MPI graph traversal kernels
    - Characterize and mitigate thread arbitration issues to enhance communication progress
  - Apr 2015 ~: Postdoc at ANL. Planning for future collaborations/visits
- Outcome
  - Two publications (PPoPP'15 and PPMM')
  - Software contribution to the MPICH library
  - Ongoing collaboration

Research and Achievements Summary

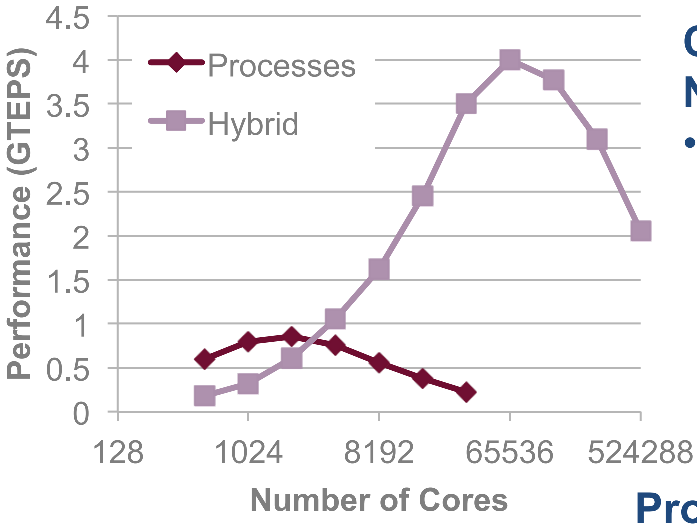


- Characterizing state-of-the-art MPI+Threads runtimes
  - Application and runtime perspectives
  - Large scale analysis (512K cores on Mira)
- Exposing thread-synchronization issues the MPI-runtime
- Develop MPI-aware thread-synchronization to improve runtime performance

[ACM PPOPP'15] Abdelhalim Amer, Huiwei Lu, Yanjie Wei, Pavan Balaji and Satoshi Matsuoka. MPI+Threads: Runtime Contention and Remedies. ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)

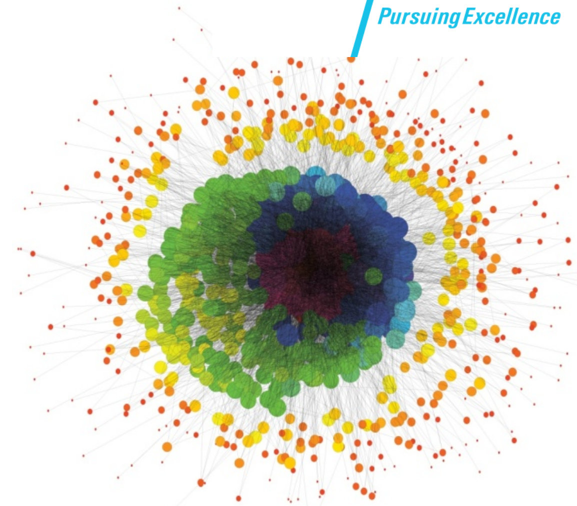
[PPMM'15] Abdelhalim Amer, Huiwei Lu, Pavan Balaji, and Satoshi Matsuoka. *Characterizing MPI and Hybrid MPI+Threads Applications at Scale: Case Study with BFS.* Workshop on Parallel Programming Model for the Masses (PPMM)

Large-Scale MPI+Threads Graph Analytics Characterization on BG/Q [PPMM’15]



Core-to-Core vs. Node-to-Node Data Movement:

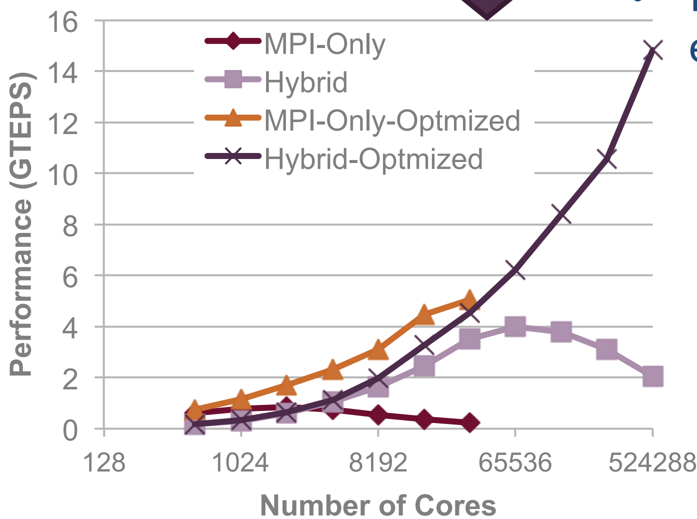
- MPI+Threads does better but cannot do miracles!



This small, synthetic graph was generated by a method called Kronecker multiplication. (Jeremiah Willcock, Indiana University)

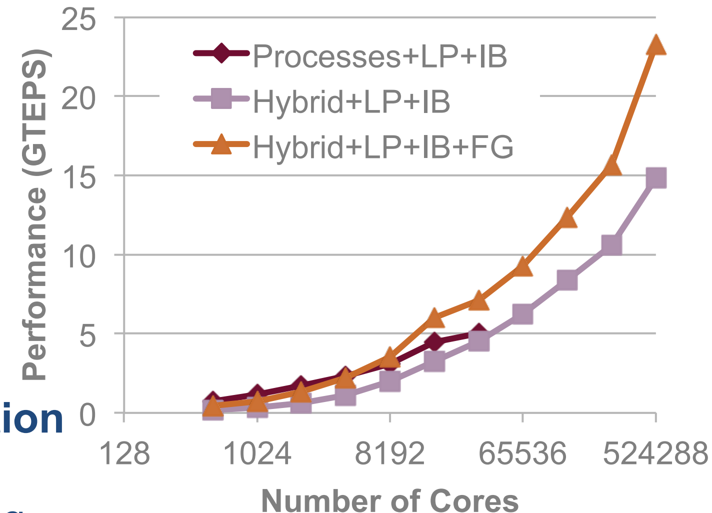
Process-level scalability optimizations:

- MPI+Threads experiences overheads



Thread-synchronization optimization:

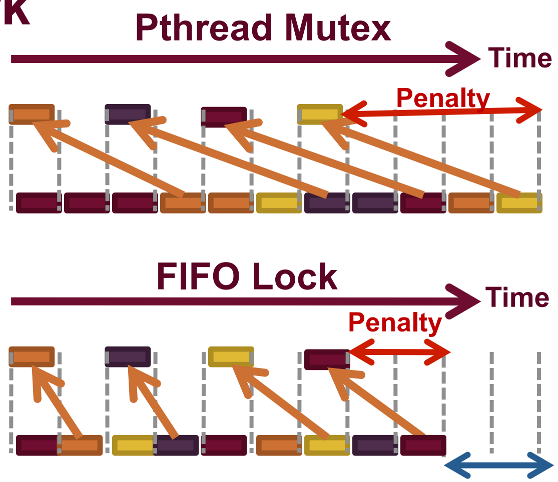
- Fine-grained locking



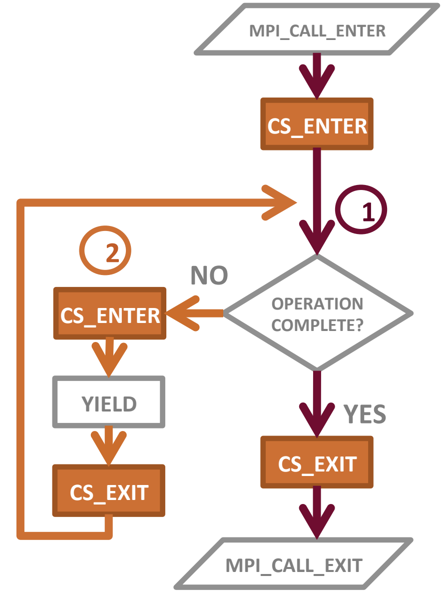
Communication Progress and Thread-Synchronization: Beware of Unbounded-Unfairness [PPoPP'15]

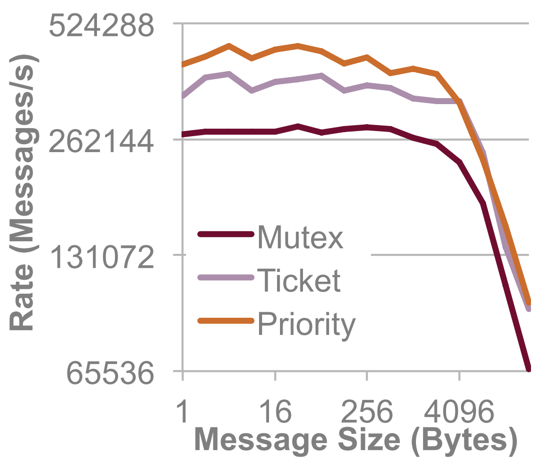
Adapt arbitration to maximize work

- FIFO locks overcome the shortcomings of mutexes
- Polling for progress can be wasteful (**waiting does not generate work**!)
- Prioritizing issuing operations
  - Feed the communication pipeline
  - Reduce chances of wasteful internal process (e.g. more requests on the fly ➔ higher chances of making progress)

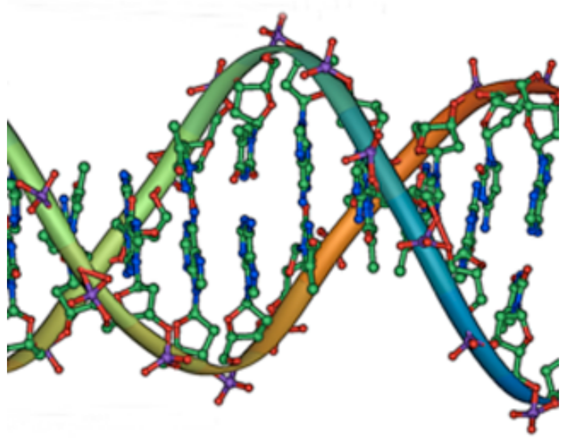


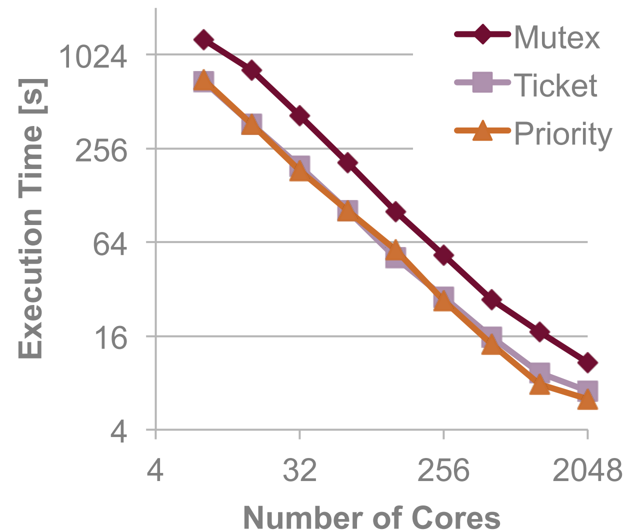
Fairness (FIFO) reduces wasted resource acquisitions





Message Rate between two 36 Haswell cores nodes

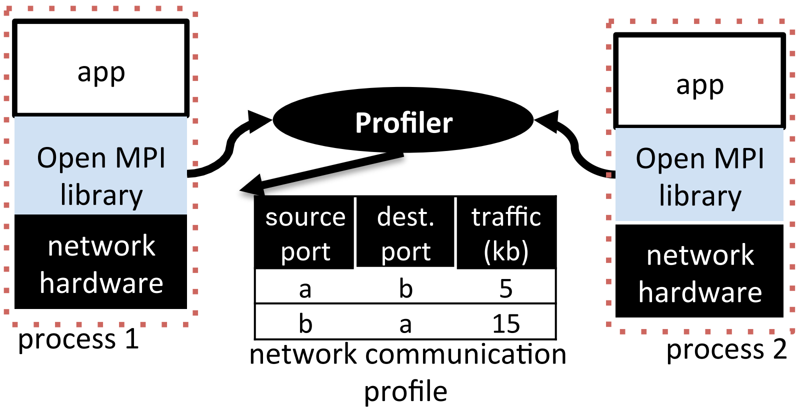




SWAP-Assembler Genome assembly application

Insightful Analysis of Performance Metrics on Fat-tree Networks[Kevin Brown, ICPADS15]

1

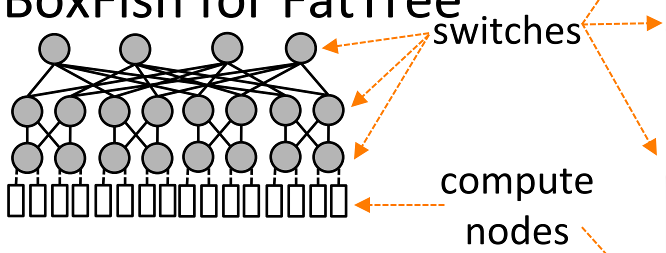


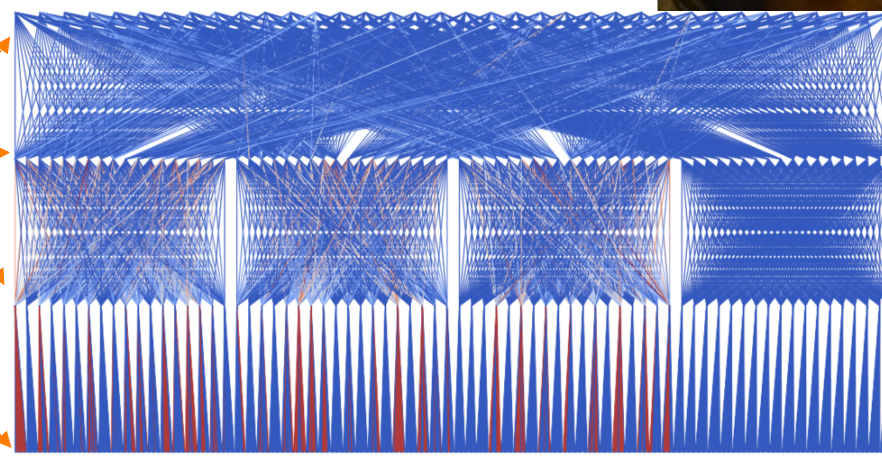
Non-intrusive collection of performance metrics w/ our **ibprof** profiler

- Low overhead
- Captures links traffic

2

Hardware-centric traffic visualization BoxFish for FatTree

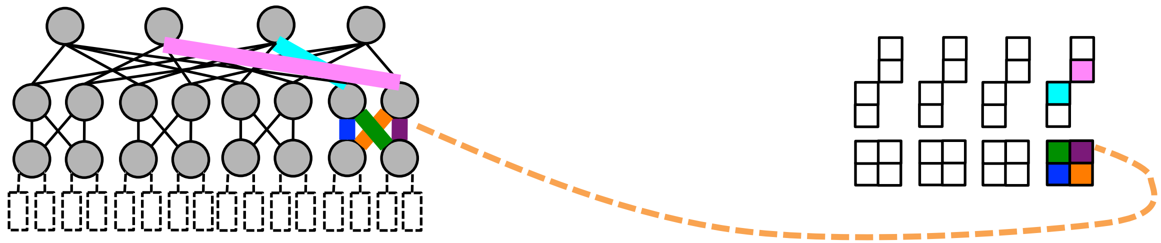




Tree-topology viz. design

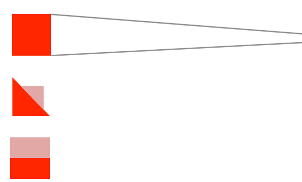
Insightful Analysis of Performance Metrics on Fat-tree Networks

3 Tree-topology viz. design → Adjacency matrix viz. design



Each element represents a link

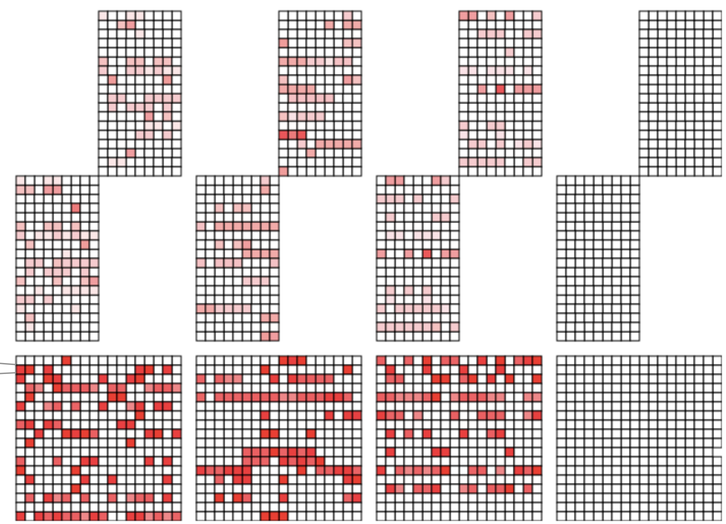
- ✓ No occlusion of data
- ✓ Space efficient design
- ✓ More link design options



Square

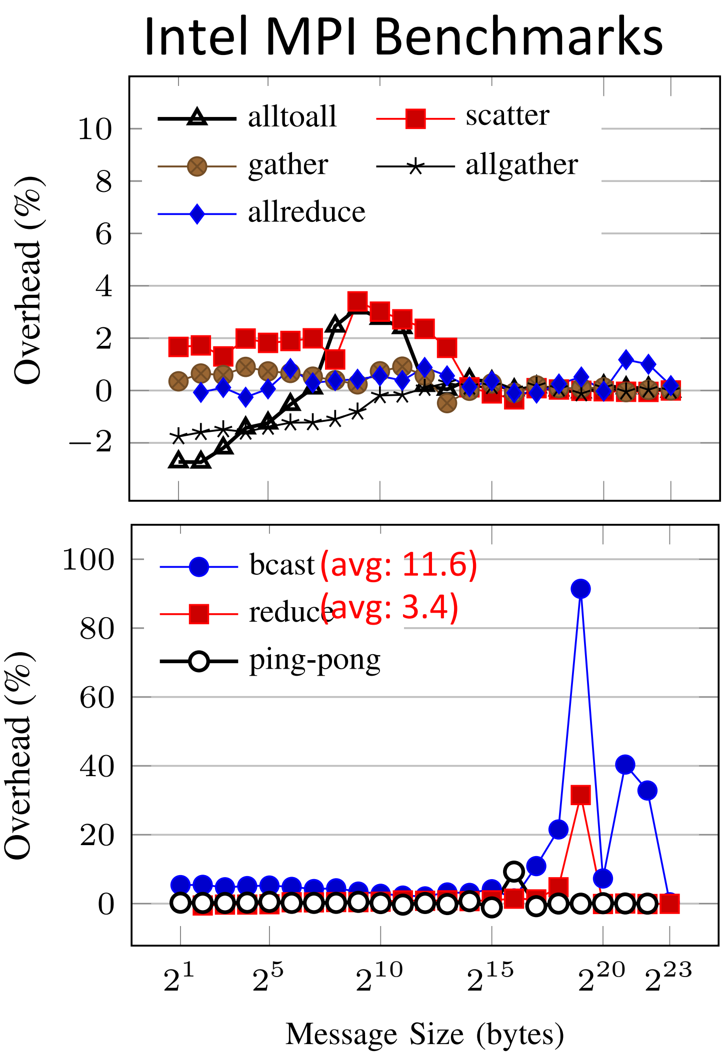
Triangle pair

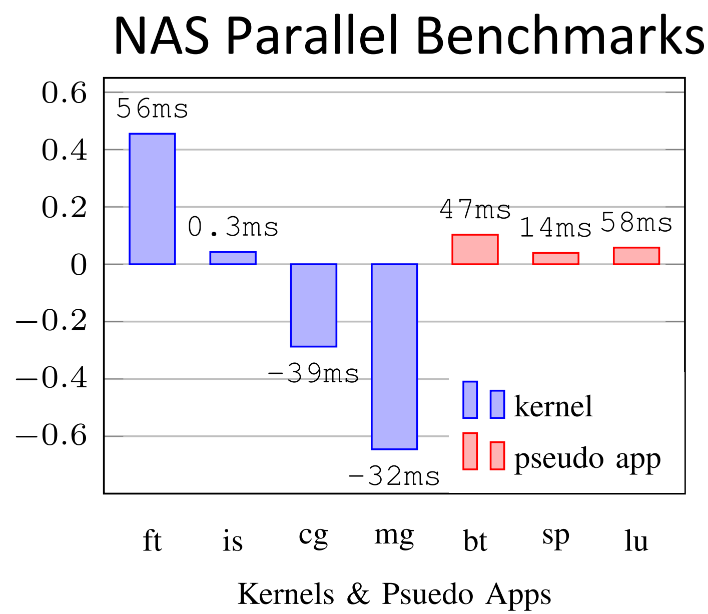
Bisected square



Data (traffic, load, etc.) is encoded in the size, shape, color, and/or hue of the links

ibprof's Profiling Overhead

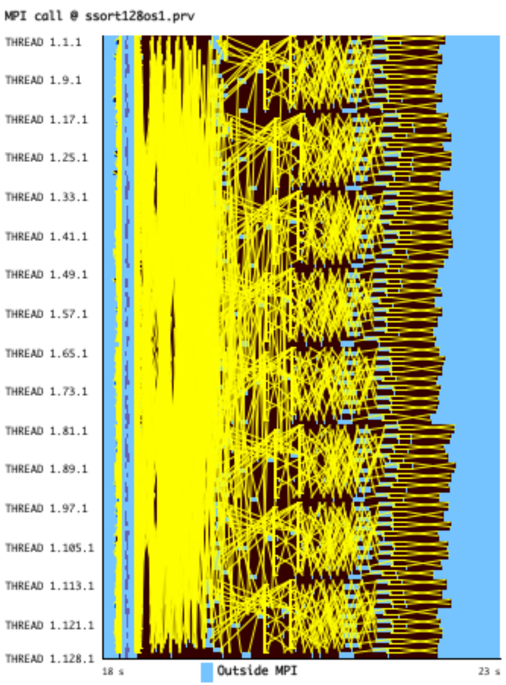




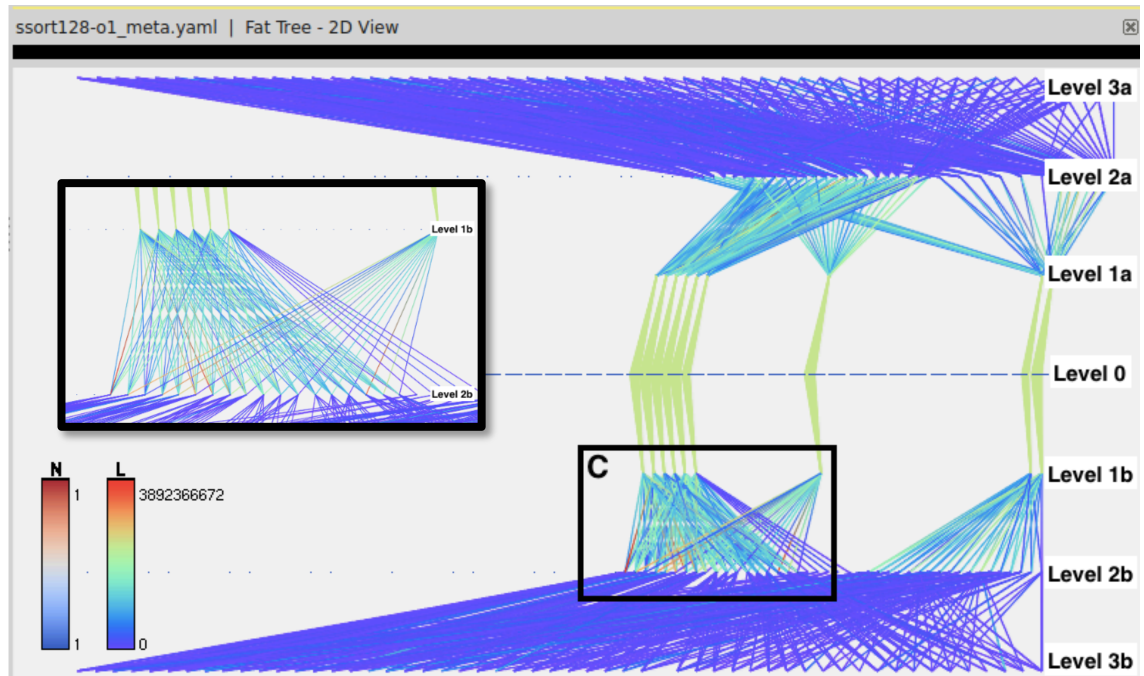
- All NPB apps averaged < 1%
- Peak overhead occurred with MPI_Bcast when Open MPI switched from send/recv to RDMA
- All other collectives averaged < 5%

Process-centric Visualizations vs. Boxfish Fat Tree Visualization

Samplesort on 128 nodes of TSUBAME2.5



vs.



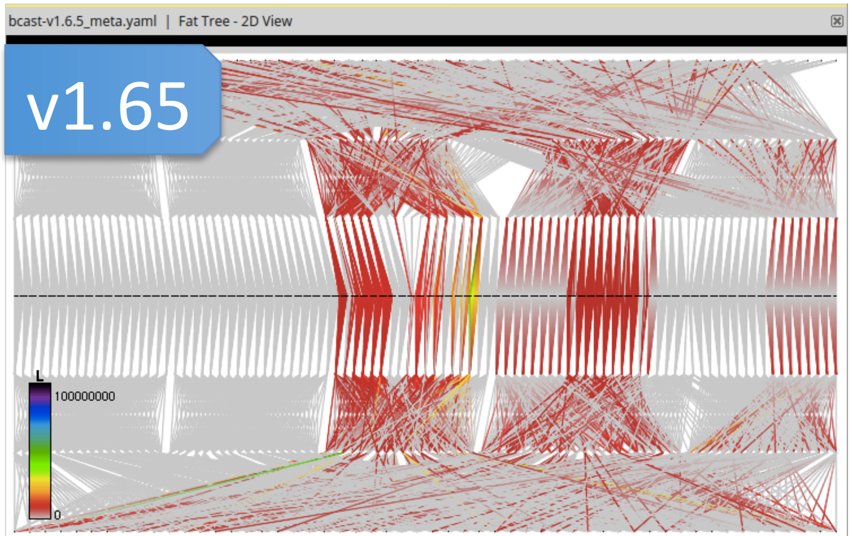
Paraver

Does not show network traffic hotspots

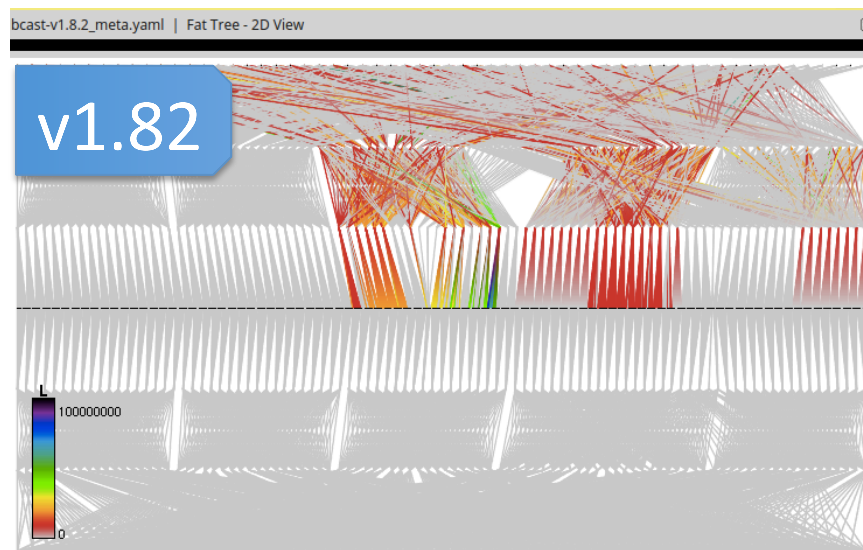
Boxfish

Capable of highlighting network hotspots and traffic patterns

Visualizing the Traffic Patterns of Different Open MPI Library version



Open MPI v1.65 balances traffic over both subnets ofTSUBAME2.5 with the default configuration



Open MPI v1.82 uses a single subnet per operation with the default configurations on TSUBAME2.5

Publications

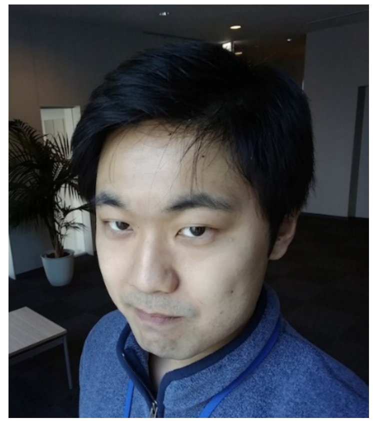
Poster (Prior to internship but using LLNL's work):

Kevin A. Brown, Jens Domke, and Satoshi Matsuoka. *"Tracing Data Movements within MPI Collectives"*. In Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/ASIA '14).

Paper:

Brown, K.A.; Domke, J.; Matsuoka, S., *"Hardware-Centric Analysis of Network Performance for MPI Applications".* In 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)

Challenges to model a tree-based irregular applications with Aspen



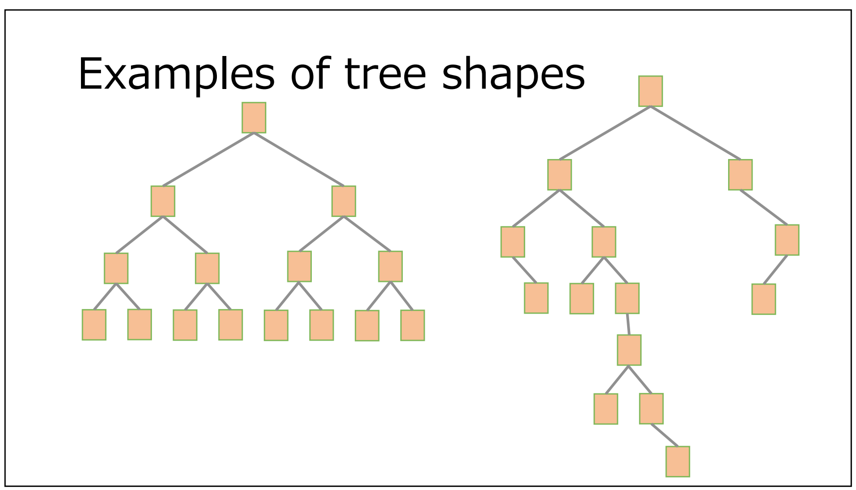
Keisuke Fukuda (Ph.D Student)
Research Internship @ORNL

- 2013 Sep-Nov
- 2014 Oct-Nov

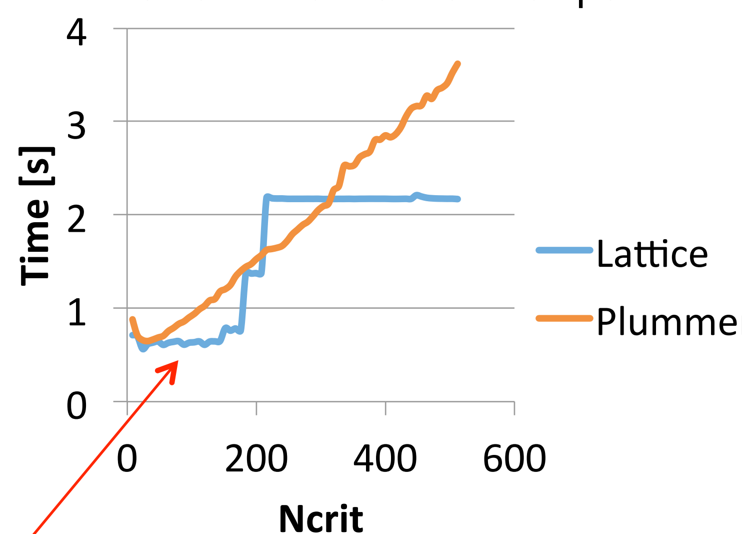
- Now long-term intern at AICS 2015 Oct-2016 Sep

Challenges in modeling irregular applications

- Performance modeling of application is used to:
    - Runtime (power, memory) estimation
    - Hardware/machine design
- Conventional, ad-hoc mathematical modeling is not suitable if irregular data structure (e.g. tree) and control flows affect the performance
- How to model such applications?
    - We focus on the Fast Multipole Method



Performance variation caused by "shape" of a tree for a fixed number of particles



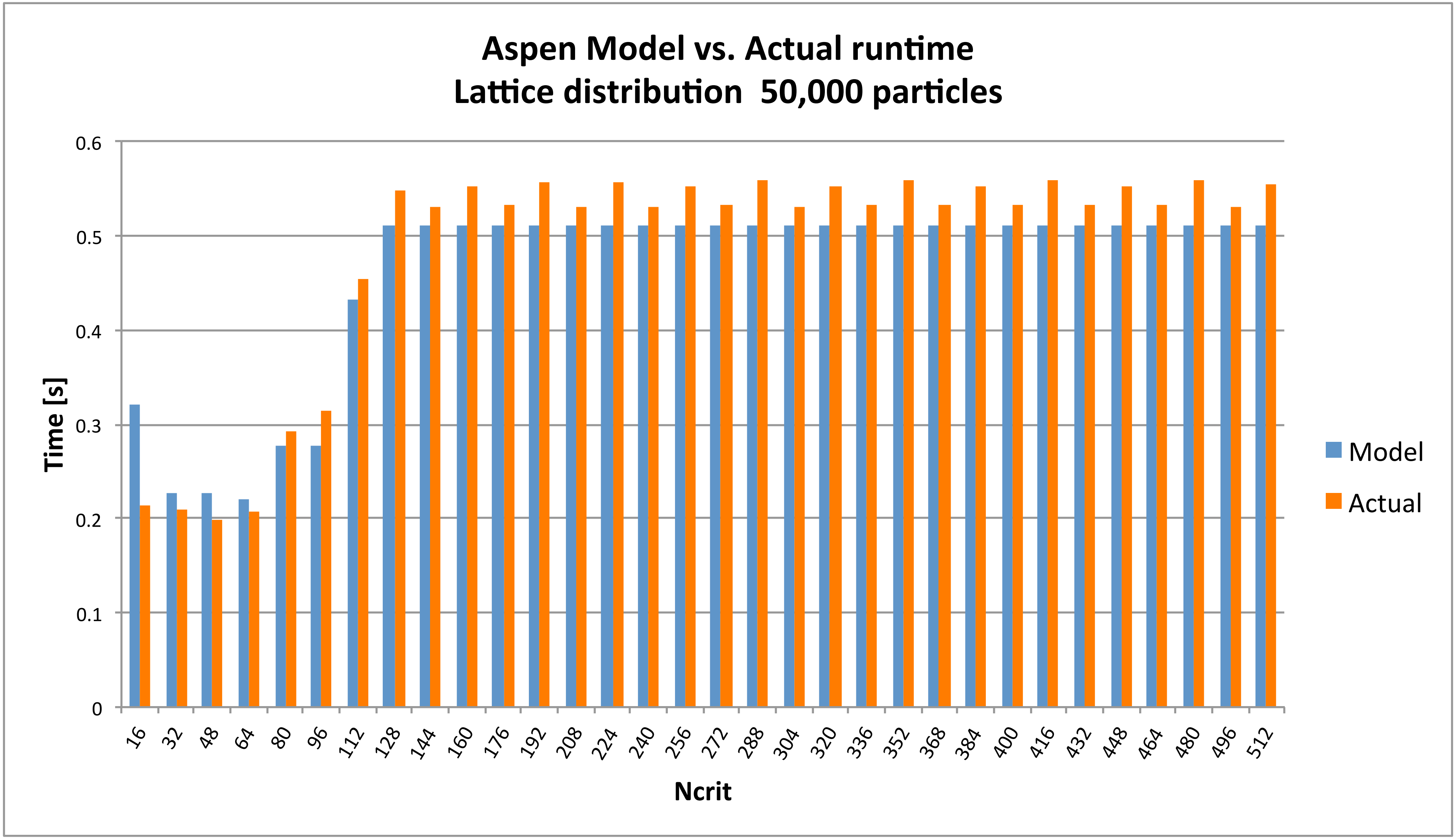
Each plot point represents a particular shape of tree

(this figure will be shown and described again)

- Applied Aspen modeling language to FMM
  - Runtime estimation for lattice, sphere, plummer distribution, Ncrit = 16〜512
- Estimation errror was 7-13% error in avg.

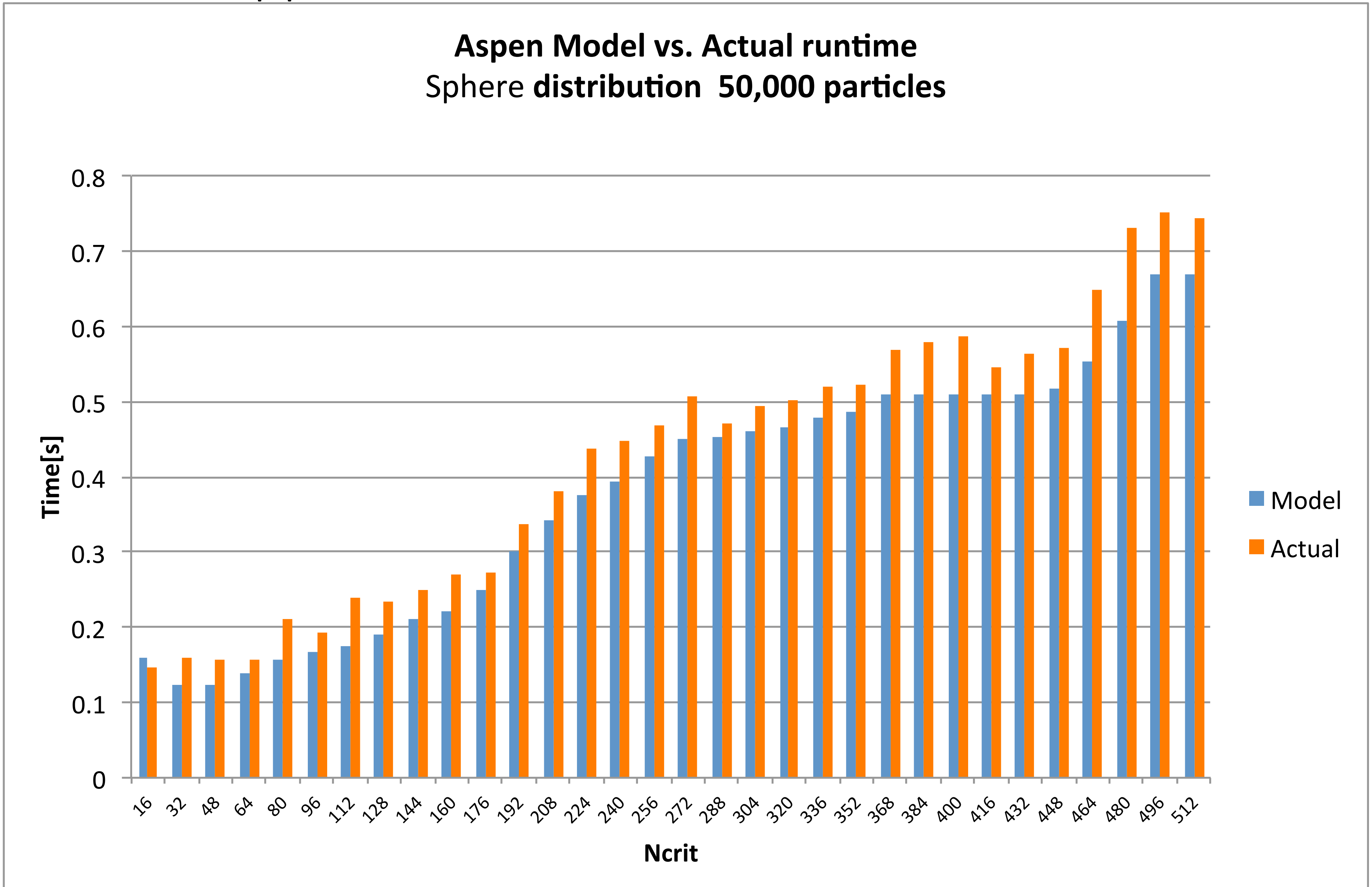
- Room for optimization for find-grained kernels and in deriving constants
- Aspen requires large time and memory to evaluate the models

Whole-app model of ExaFMM

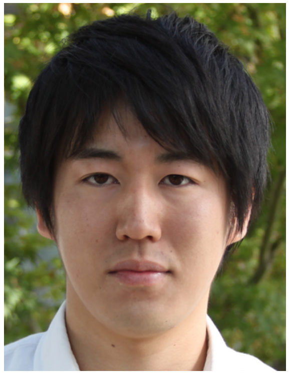


Error: avg 7.7%, max 33.2%, min 3.7%

Whole-app model of ExaFMM



Error: avg 12.8%, max 26.9%, min 4.0%



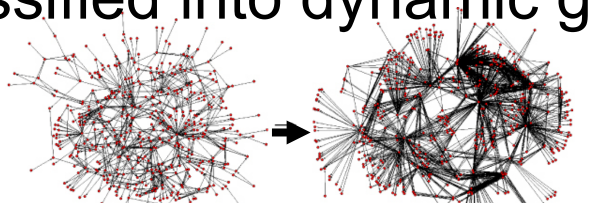
Distributed Large-Scale Dynamic Graph Data Store

Keita Iwabuchi[1, 2], Scott Sallinen[3], Roger Pearce[2],
Brian Van Essen[2], Maya Gokhale[2], Satoshi Matsuoka[1]

1. Tokyo Institute of Technology (Tokyo Tech)
2. Lawrence Livermore National Laboratory (LLNL)
3. University of British Columbia

Dynamic Graphs (temporal graph)

- the structure of a graph changes dynamically over time
- many real-world graphs are classified into dynamic graph



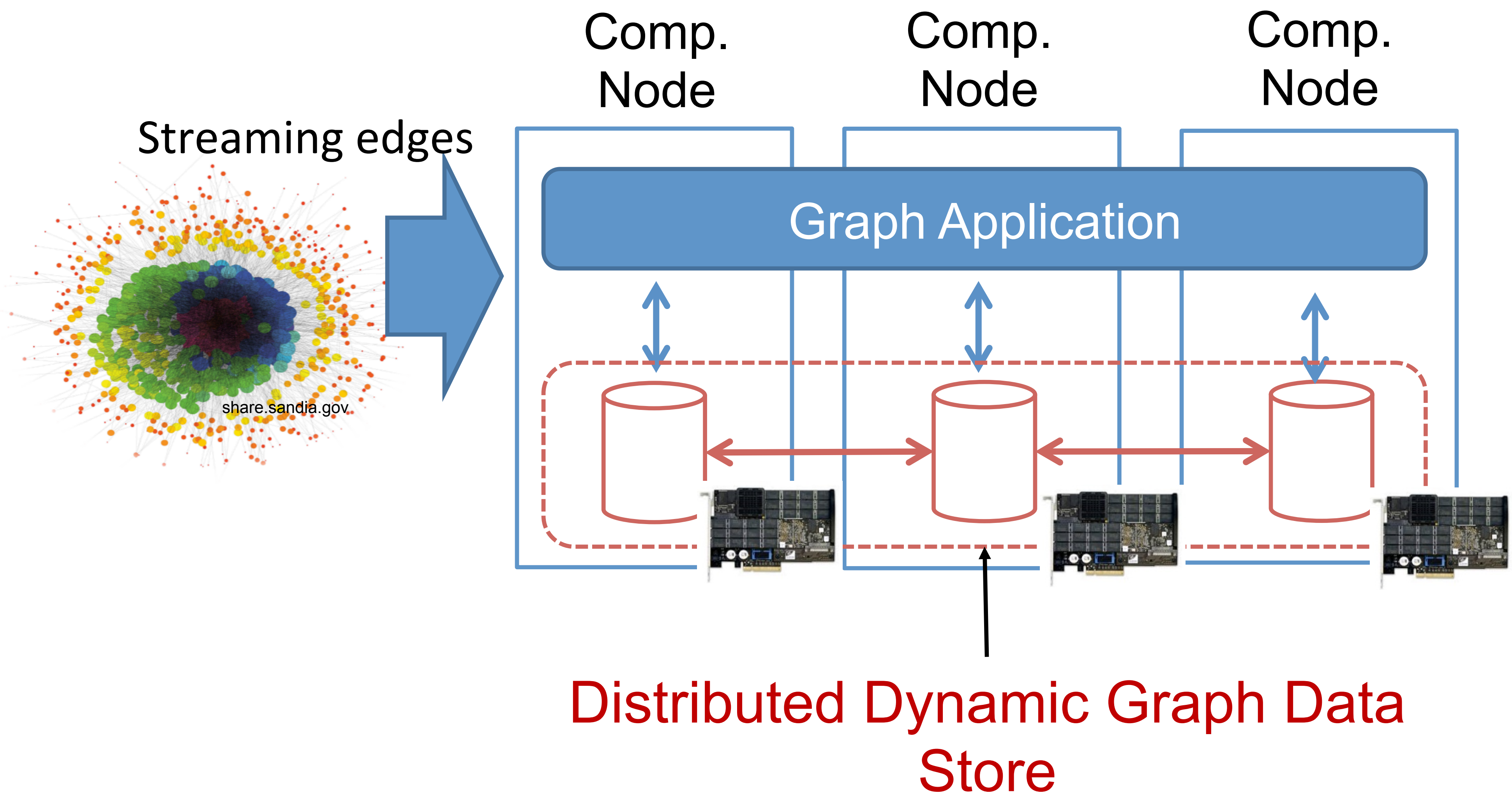
Source: Jakob Enemark and Kim Sneppen, "Gene duplication models for directed networks with limits on growth", Journal of Statistical Mechanics: Theory and Experiment 2007

Sparse Large Scale-free

- social network, genome analysis, WWW, etc.
  - e.g., Facebook manages 1.39 billion active users as of 2014, with more than 400 billion edges

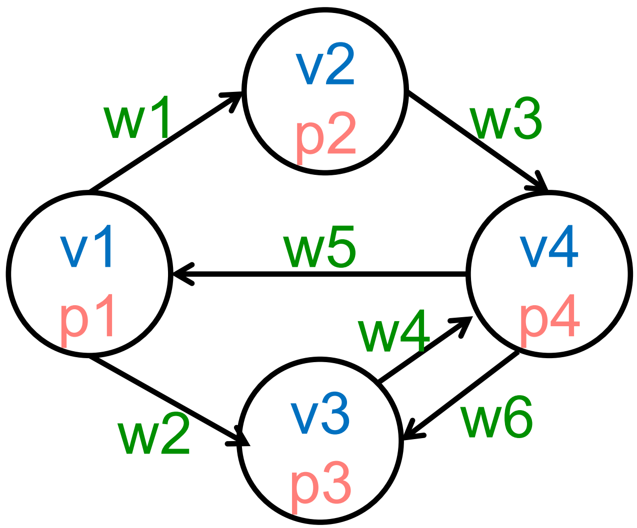
- Most studies for large graphs have not focused on a dynamic graph data structure, but rather a static one, such as Graph 500
- Even with the large memory capacities of HPC systems, many graph applications require additional out-of-core memory (this part is still at an early stage)

Developing a distributed dynamic graph store for data intensive supercomputers equipped with locally attached NVRAM

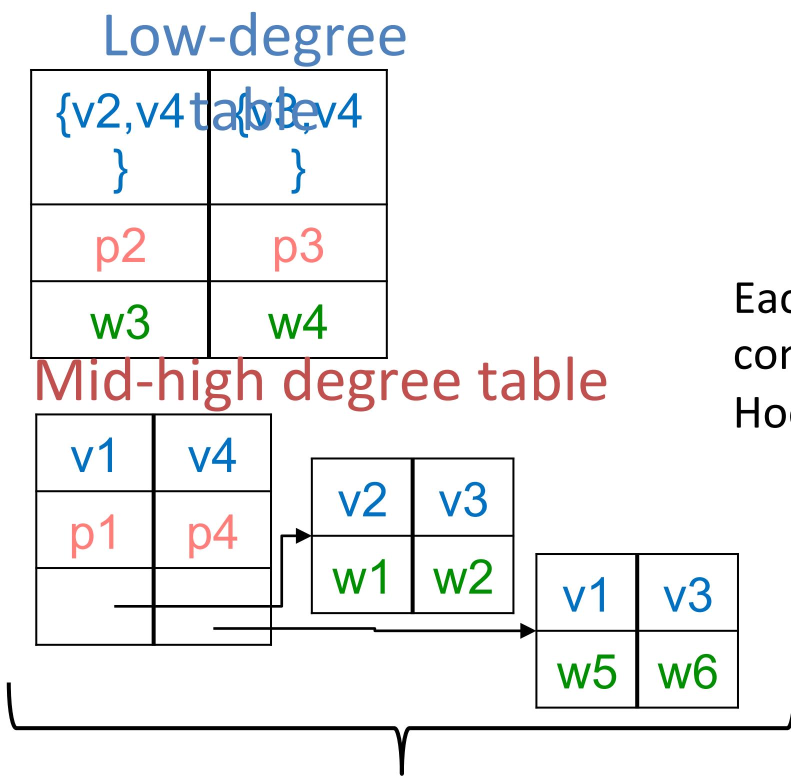


Degree Aware Dynamic Graph Data Store (DegAwareRHH)

- Degree aware data structures, where low-degree vertices are compactly represented
- Use Robin Hood Hashing[1] because of its locality properties to minimize the number of accesses to NVRAM, reducing page misses.



Vertex ID
Vertex property
Edge weight



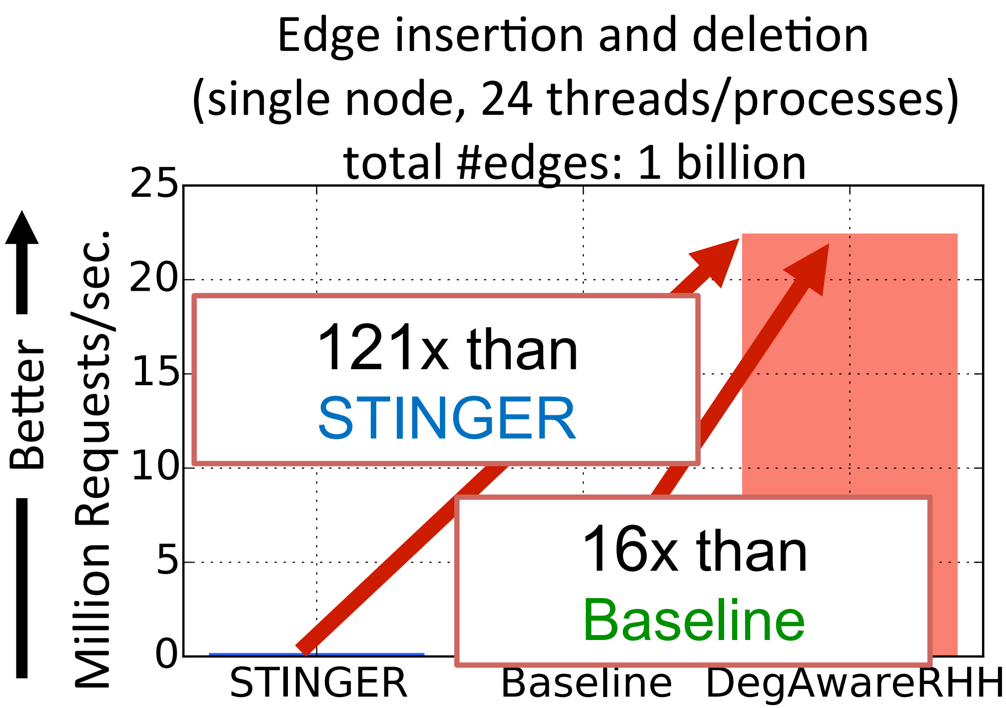
Each table is composed of Robin Hood Hashing

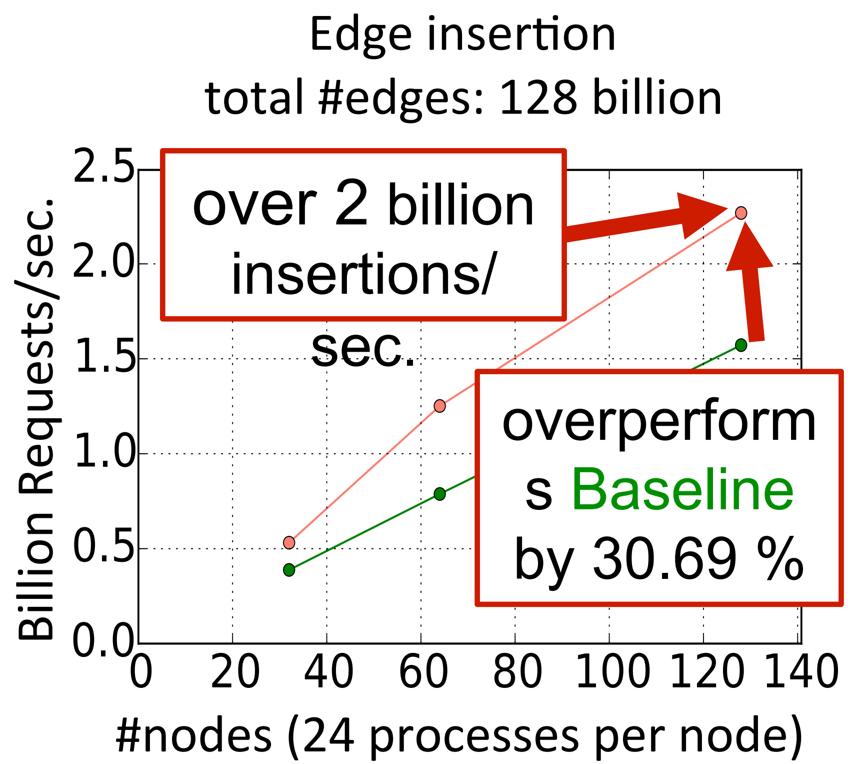
Extend DegAwareRHH for distributed-memory using a async. MPI communication framework[2][3]

[2] R. Pearce, et al, "Scaling techniques for massive scale-free graphs in distributed (external) memory," IPDPS' 13

Dynamic Large-Scale Graph Construction (on-memory)

- **STINGER**: a state-of-the-art shared-memory dynamic graph processing framework developing at Georgia Tech
- **Baseline**: a baseline model using *Boost.Interprocess*
- **DegAwareRHH**: our proposed dynamic graph store



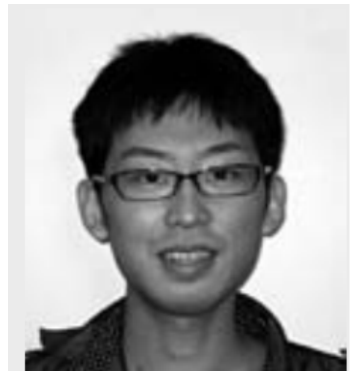


Due to a skewness of the data set (RMAT graph), DegAwareRHH overperforms the both implementations significantly

Publication list

- Keita Iwabuchi, Roger A. Pearce, Brian Van Essen, Maya Gokhale, Satoshi Matsuoka, "**Design of a NVRAM Specialized Degree Aware Dynamic Graph Data Structure**", SC 2015 Regular, Electronic, and Educational Poster, International Conference for High Performance Computing, Networking, Storage and Analysis 2015 (SC '15), Nov. 2015
- Keita Iwabuchi, Roger A. Pearce, Brian Van Essen, Maya Gokhale, Satoshi Matsuoka, "**Design of a NVRAM Specialized Degree Aware Dynamic Graph Data Structure**", 7th Annual Non-Volatile Memories Workshop 2016, Mar. 2016

An OpenACC Extension for Data Layout Transformation w/ORNL

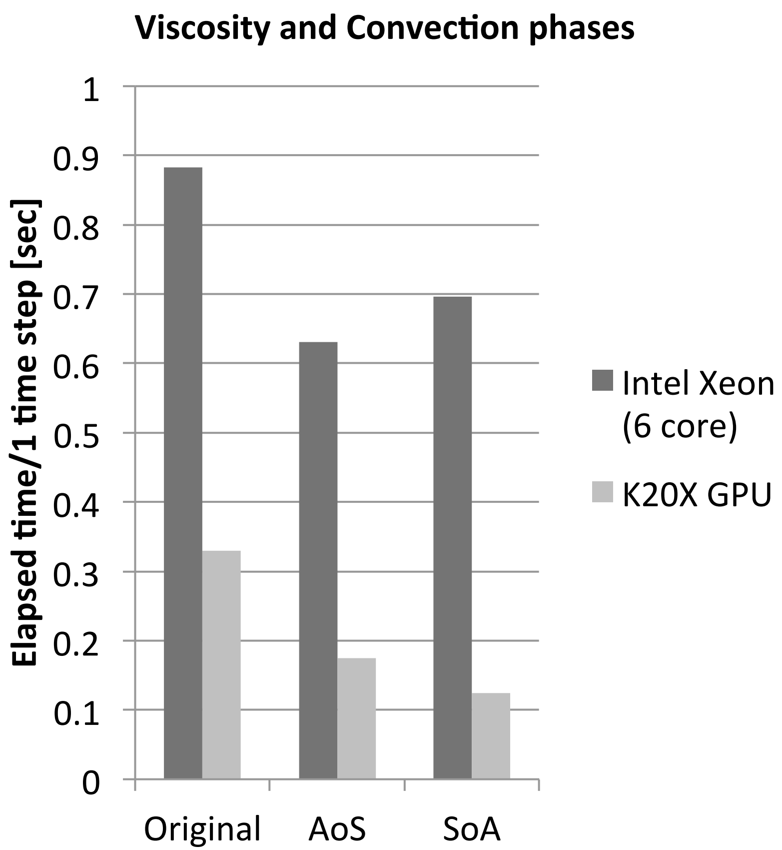


Tetsuya Hoshino(Ph.D Student)

Research Internship @ORNL 2014 Sep-Nov

Now: Assistant Professor @ Supercomputing Center, The University of Tokyo

Why the extension is needed?



The graph shows the result of manual data layout transformation for the viscosity and convection phases of a real-world CFD application UPACS
(Hoshino et al. "CUDA vs OpenACC: Performance Case Studies with Kernel Benchmarks and a Memory-Bound CFD Application", CCGrid13)

- An OpenACC program can be executed on any devices
  - multi-core CPU, Xeon Phi, GPUs
- OpenACC target devices have different performance characteristics especially about memory access
  - ex. SoA and AoS
- Data layout of real-world applications is complicated and is shared in the whole program
  - Auto-tuning is required

An OpenACC extension #pragma acc transform

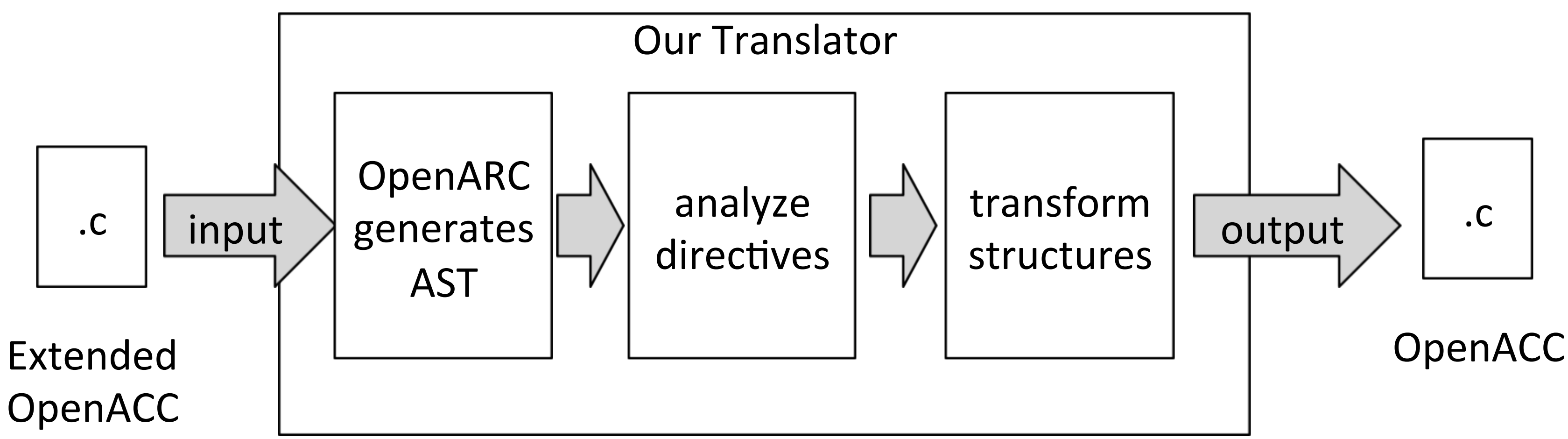
- Specification

```
#pragma acc transform [clause [[,] clause] …] new-line
        structured block
```

- Clause list
  - **transpose**(array_name::transpose_rule)
    - for multi-dimensional array
    - A[Z][Y][X][3] → A'[3][Z][Y][X] (transpose rule :: [4,1,2,3])
  - **redim**(array_name::redim_rule)
    - for 1 dimensional array
    - B[Z*Y*X*3] → B'[Z][Y][X][3] → B''[3][Z][Y][X] (by transpose clause)
  - **expand**(derived_type_array_name)
    - for array of structures
    - C[Z][Y][X].c[3] → C'[Z][Y][X][3] → C''[3][Z][Y][X] (by transpose clause)

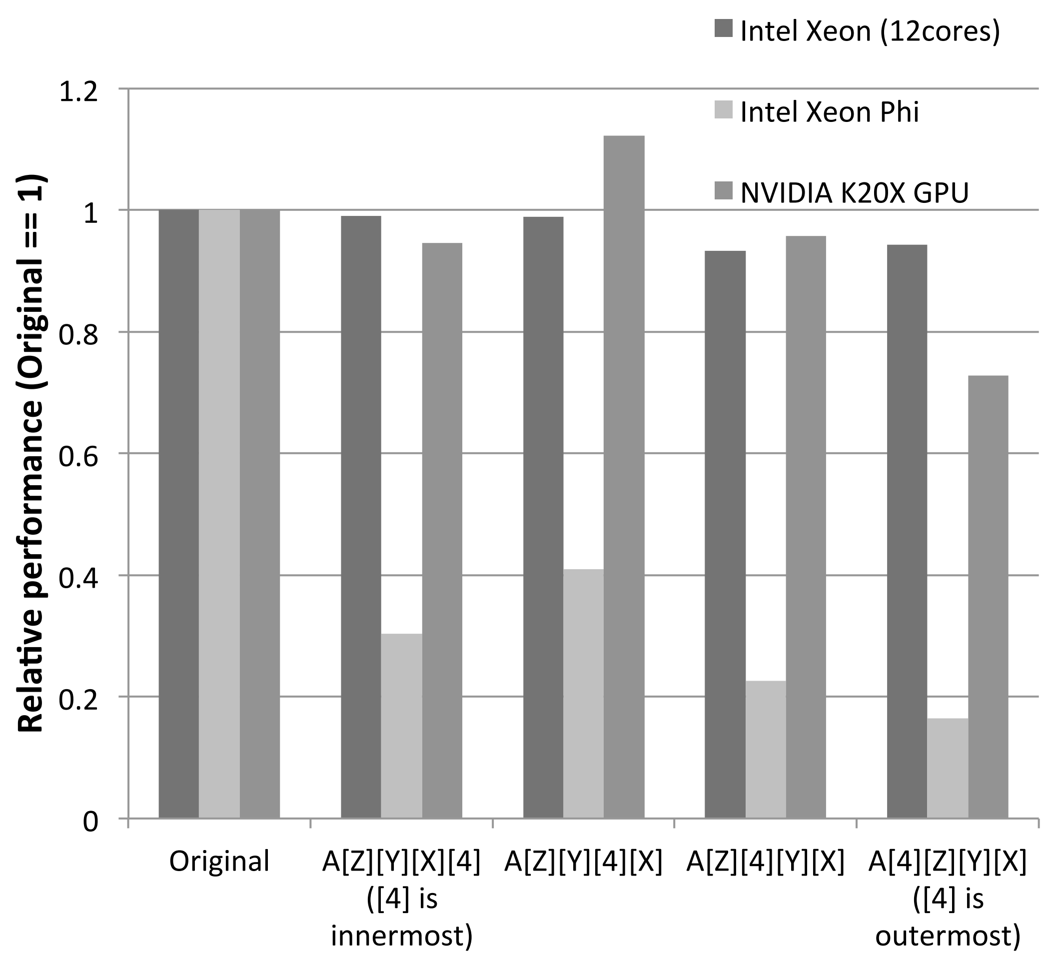
Collaborate with ORNL

- Implement the directive top on OpenARC that is an Open-source OpenACC compiler developed by ORNL
  - Source-to-Source translator
    - Input : Extended OpenACC program
    - Output : OpenACC program
  - It is on going work



Evaluate with Himeno benchmark (27-point stencil program)

- Apply *transpose* to coefficient arrays of Himeno benchmark
  - But the transformation is applied by hands
  - Transformed program is same as the output program that OpenARC should output
- Performance evaluation
  - CPU : Original is the best
  - GPU : 24% up
  - MIC : more than 60% down
    - Translator change the coefficient multidimensional array to 1-dimensional array, it disturbs prefetching



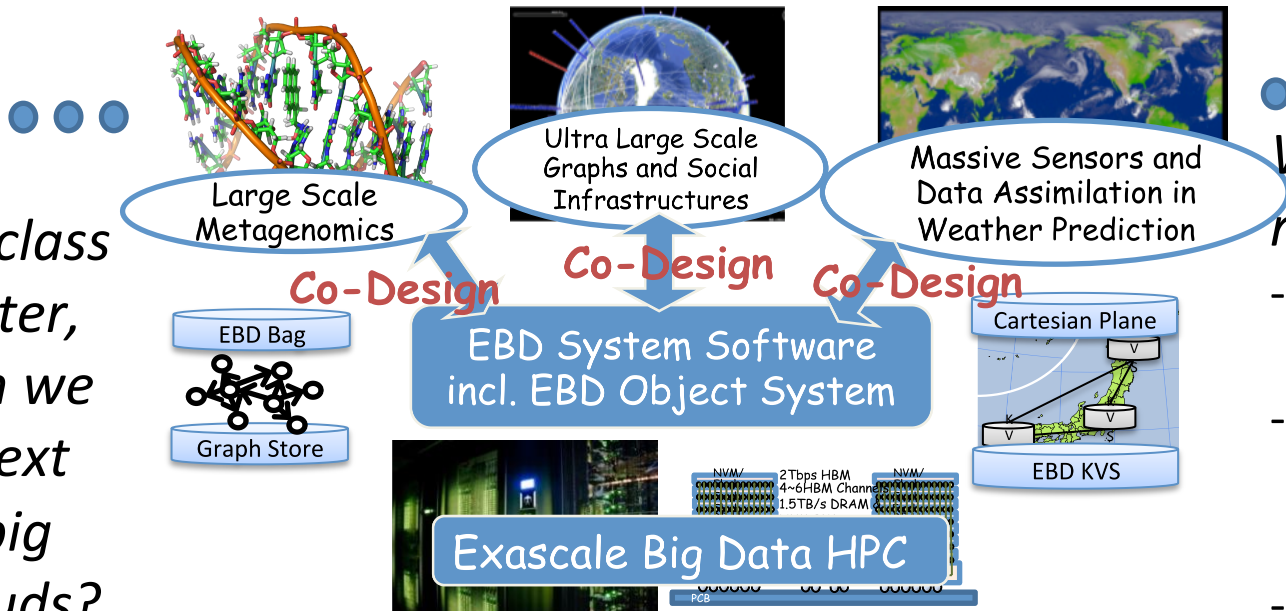
Lessons Learned

- Sending actual Ph.D. students to DoE labs extremely productive for both sides for tangible collabration
- Tokyo Tech Ph.D. students are extremely good and well trained by global standards – they usually survive the filtering of summer interns and produce tangible results
- Many students end up being hired by DoE labs. Others go to Japanese univ. & labs, etc. => great talent pool
- Some administrative obstacles, esp. travel and funding from both ends – need more flexibility in purpose, airlines, gaps in travel itinerary, etc.

Tokyo Tech Research on Big Data Convergence JST-CREST "Extreme Big Data" Project (2013-2018)

Future Extreme Big Data Scientific Apps

Given a top-class supercomputer, how fast can we accelerate next generation big data c.f. Clouds?



World-leading results:

- ***#1 Graph 500 2014, 2015***
- ***#1 Green Graph 500 (TsubameKFC)***
- ***GPU sort scalable to ~30Petabyte/s on future SCs***
- ***OSSs in dev.***

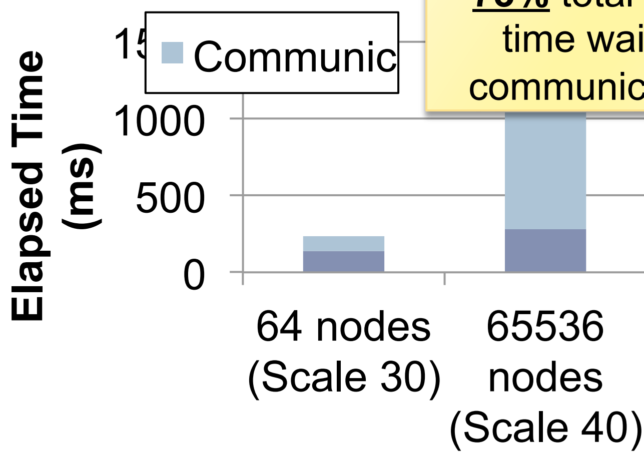
Convergent Architecture (Phases 1~4)
Large Capacity NVM, High-Bisection NW

Cloud IDC
Very low BW & Efficiency
Highly available, resilient

Supercomputers
Compute&Batch-Oriented
More fragile

The Graph500 – June 2014 and June/Nov 2015

K Computer #1 Tokyo Tech[EBD CREST] Univ. Kyushu [Fujisawa Graph CREST], Riken AICS, Fujitsu



73% total exec time wait in communication

88,000 nodes, 700,000 CPU Cores
1.6 Petabyte mem
20GB/s Tofu NW

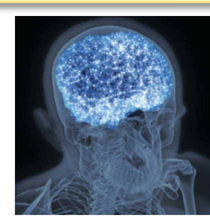
K computer

≫

LLNL-IBM Sequoia
1.6 million CPUs
1.6 Petabyte mem

List	Rank	GTEPS	Implementation
November 2013	4	5524.12	Top-down only
June 2014	1	17977.05	**Efficient hybrid**
November 2014	2		**Efficient hybrid**
June/Nov 2015	**1**	**38621.4**	**Hybrid + Node Compression**

*Problem size is weak scaling "Brain-class" graph

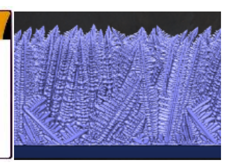


2017 Q1 TSUBAME3.0+2.5 Towards Exa & Big Data

1. "Everybody's Supercomputer" – High Performance (15~20 Petaflops, ~4PB/s Mem, ~1Pbit/s NW), innovative high cost/performance packaging & design, in mere 100m²…
2. "Extreme Green" – 9~10GFlops/W power-efficient architecture, system-wide power control, advanced cooling, future energy reservoir load leveling & energy recovery
3. "Big Data Convergence" – Extreme high BW &capacity, deep memory hierarchy, extreme I/O acceleration, Big Data SW Stack for machine learning /DNN, graph processing, …
4. "Cloud SC" – dynamic deployment, container-based node co-location & dynamic configuration, resource elasticity, assimilation of public clouds…
5. "Transparency" - full monitoring & user visibility of machine & job state, accountability via reproducibility

2006 TSUBAME1.0
80 Teraflops, #1 Asia #7 World
"Everybody's Supercomputer"

2010 TSUBAME2.0
2.4 Petaflops #4 World
"Greenest Production SC"

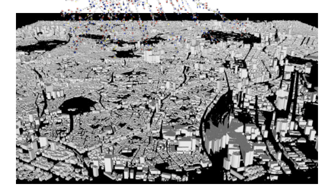


2011 ACM Gordon Bell Prize

2013
TSUBAME2.5
upgrade
5.7PF DFP /
17.1PF SFP
20% power
reduction

2013 TSUBAME-KFC
#1 Green 500

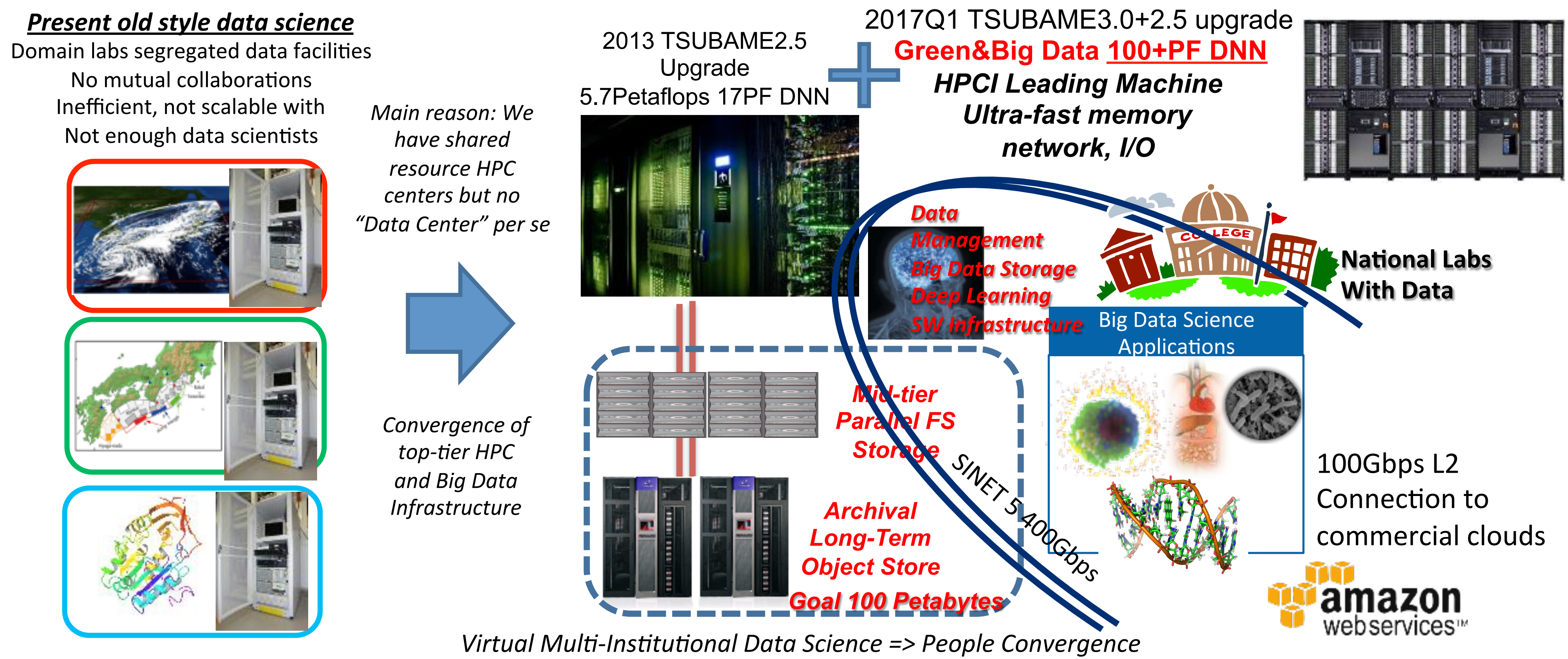
2017 TSUBAME3.0
15~20PF(DFP) ~4PB/s Mem BW
9~10GFlops/W power efficiency
Big Data & Cloud Convergence



Large Scale Simulation
Big Data Analytics
Industrial Apps

Big Data and HPC Convergent Infrastructure => "Big Data & Supercomputing Convergent Center"

- "Big Data" currently processed managed by domain laboratories => No longer scalable
- HPCI HPC Center => Converged HPC and Big Data Science Center
- People convergence: domain scientists + data scientists + CS/Infrastructure => Big data science center
- Data services including large data handling, big data structures e.g. graphs, ML/DNN/AI services…



New collaborations under consideration

- Fault tolerance towards exascale
  - Modeling & analyzing soft errors with "realistic" machine fault models (Kobayashi)
  - General system-level GPU checkpointing (Suzuki)

To be presented @ DoE/MEXT workshop)

- Big Data / IoT / Machine Learning-AI & HPC Convergence
  - Modeling deep learning algorithms performance (Ooyama)
  - Counterpart to Tokyo Tech Extreme Big Data (EBD) Project w/DENSO
- Post-Moore computing
  - Programming / Performance modeling future FPGAs (also w/Riken AICS Naoya Maruyama (Hamid)
  - FLOPS to BYTES – from compute intensive to bandwidth/capacity intensive computing (w/Kengo Nakajima, Toshio Endo et. al.)
- ADAC (Accelerated Data Analytics and Computing) Institute – ORNL – ETH/CSCS – Tokyo Tech GSIC