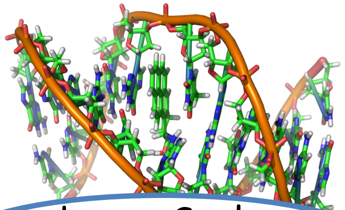
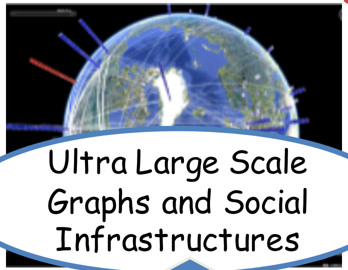
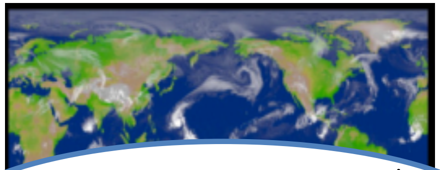
JST-CREST "Extreme Big Data" Project (2013-2018)

Future Non-Silo Extreme Big Data Scientific Apps

Large Scale Metagenomics

Ultra Large Scale Graphs and Social Infrastructures

Massive Sensors and Data Assimilation in Weather Prediction

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

Co-Design

Co-Design

Co-Design

EBD Bag

Graph Store

EBD System Software incl. EBD Object System

Cartesian Plane

EBD KVS

2Tbps HBM

4~6HBM Channels

1.5TB/s DRAM &

Exascale Big Data HPC

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

Issues regading Architectural, algorithmic, and system software evolution?

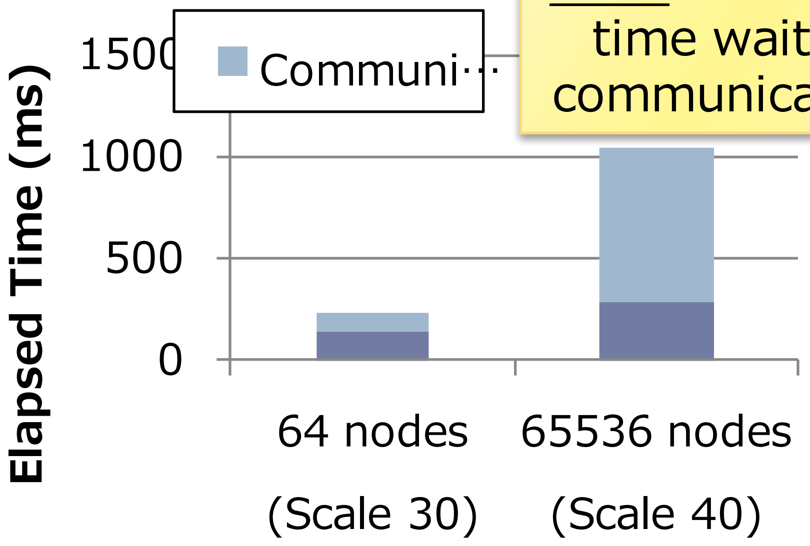
<u>Use of GPUs?</u>

Cloud IDC
Very low BW & Efficiency
Highly available, resilient

Supercomputers
Compute&Batch-Oriented
More fragile

The Graph500 – June 2014 and June 2015

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

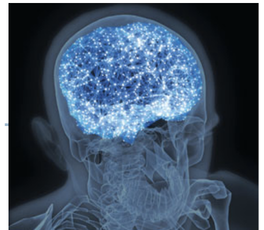


73% total exec time wait in communication

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

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 2015	**1**	**38621.4**	**Hybrid + Node Compression**

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



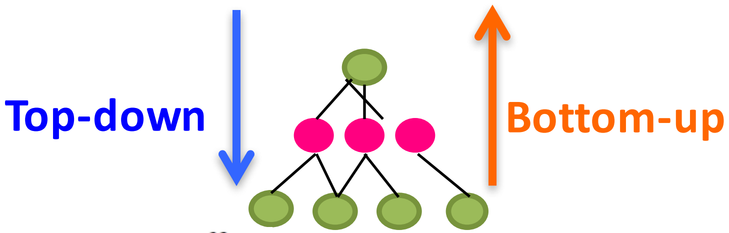
LLNL-IBM Sequoia
1.6 million CPUs
1.6 Petabyte mem

Large Scale Graph Processing Using NVM [Iwabuchi, IEEE BigData2014]

1. Hybrid-BFS (Beamer'11)

EBD Algorithm Kernels

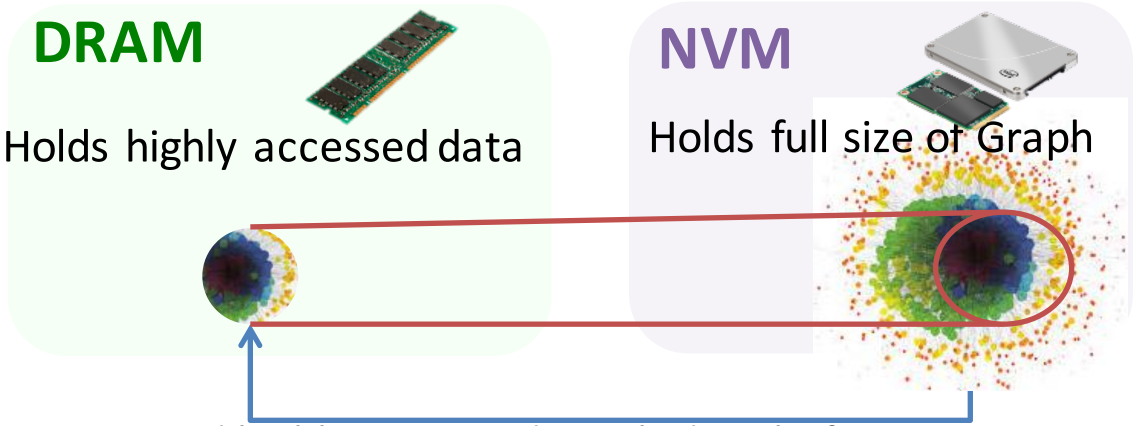
Switching two approaches



$n_{frontier} < \frac{n_{all}}{\beta}$ $\quad$ $n_{frontier} > \frac{n_{all}}{\alpha}$

# of frontiers: $n_{frontier}$, # of all vertices: n_{all}, Parameter : α, β

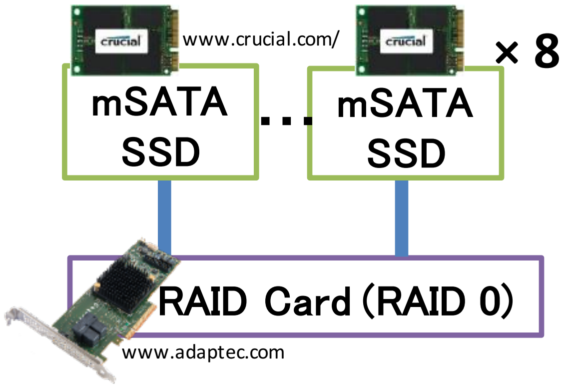
2. Proposal

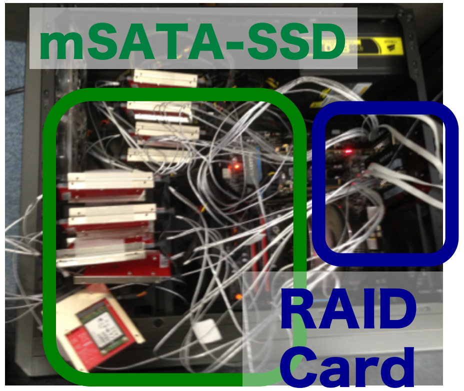


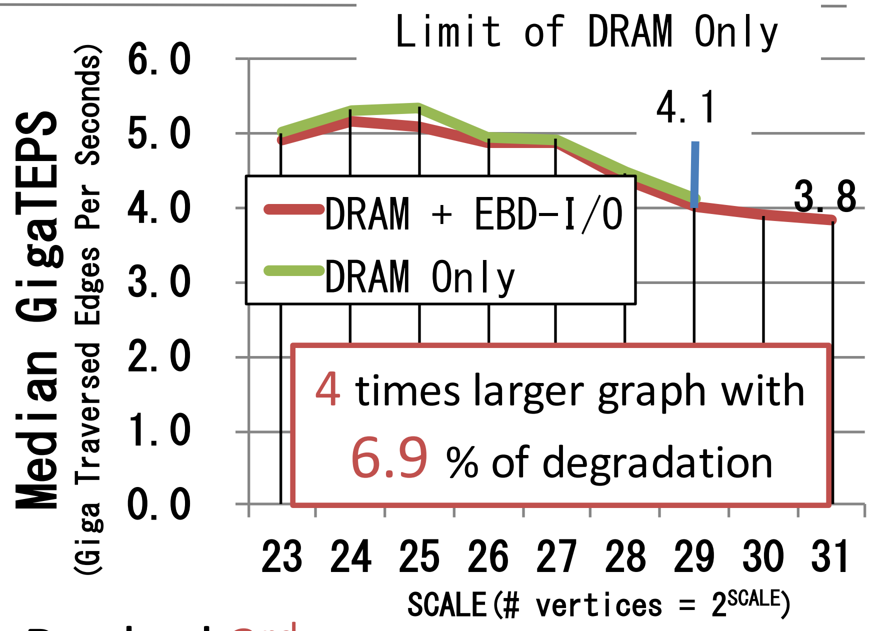
Load highly accessed graph data before BFS

3. Experiment

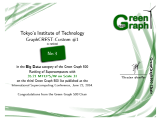
CPU	Intel Xeon E5-2690 × 2
DRAM	256 GB
NVM	EBD-I/O 2TB × 2







Ranked 3rd in Green Graph500 (June 2014)



GPU-based Distributed Sorting
[Shamoto, IEEE BigData 2014, IEEE Trans. Big Data 2015]

EBD Algorithm Kernels

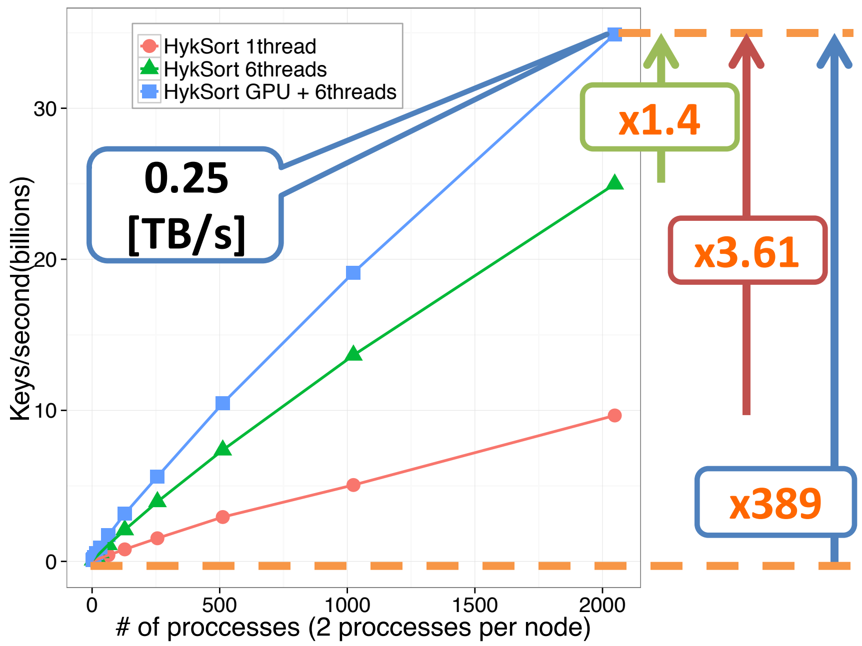
- Sorting: Kernel algorithm for various EBD processing
- Fast sorting methods
  - Distributed Sorting: Sorting for distributed system
    - Splitter-based parallel sort
    - Radix sort
    - Merge sort
  - Sorting on heterogeneous architectures
    - Many sorting algorithms are accelerated by many cores and high memory bandwidth.

- Sorting for large-scale heterogeneous systems remains unclear

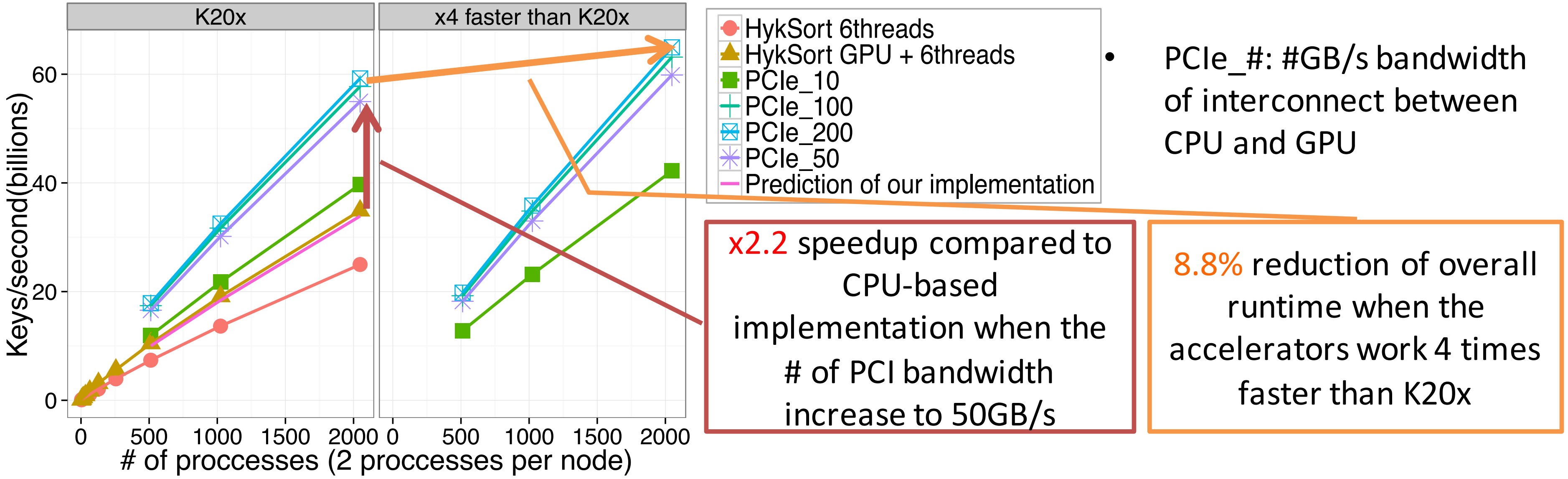
- We develop and evaluate bandwidth and latency reducing GPU-based HykSort on TSUBAME2.5 via latency hiding
  - Now preparing to release the sorting library

GPU implementation of splitter-based sorting (HykSort)

- Weak scaling performance (Grand Challenge on TSUBAME2.5)
  - 1 ~ 1024 nodes (2 ~ 2048 GPUs)
  - 2 processes per node
  - Each node has 2GB 64bit integer
- C.f. Yahoo/Hadoop Terasort: 0.02[TB/s]
  - Including I/O

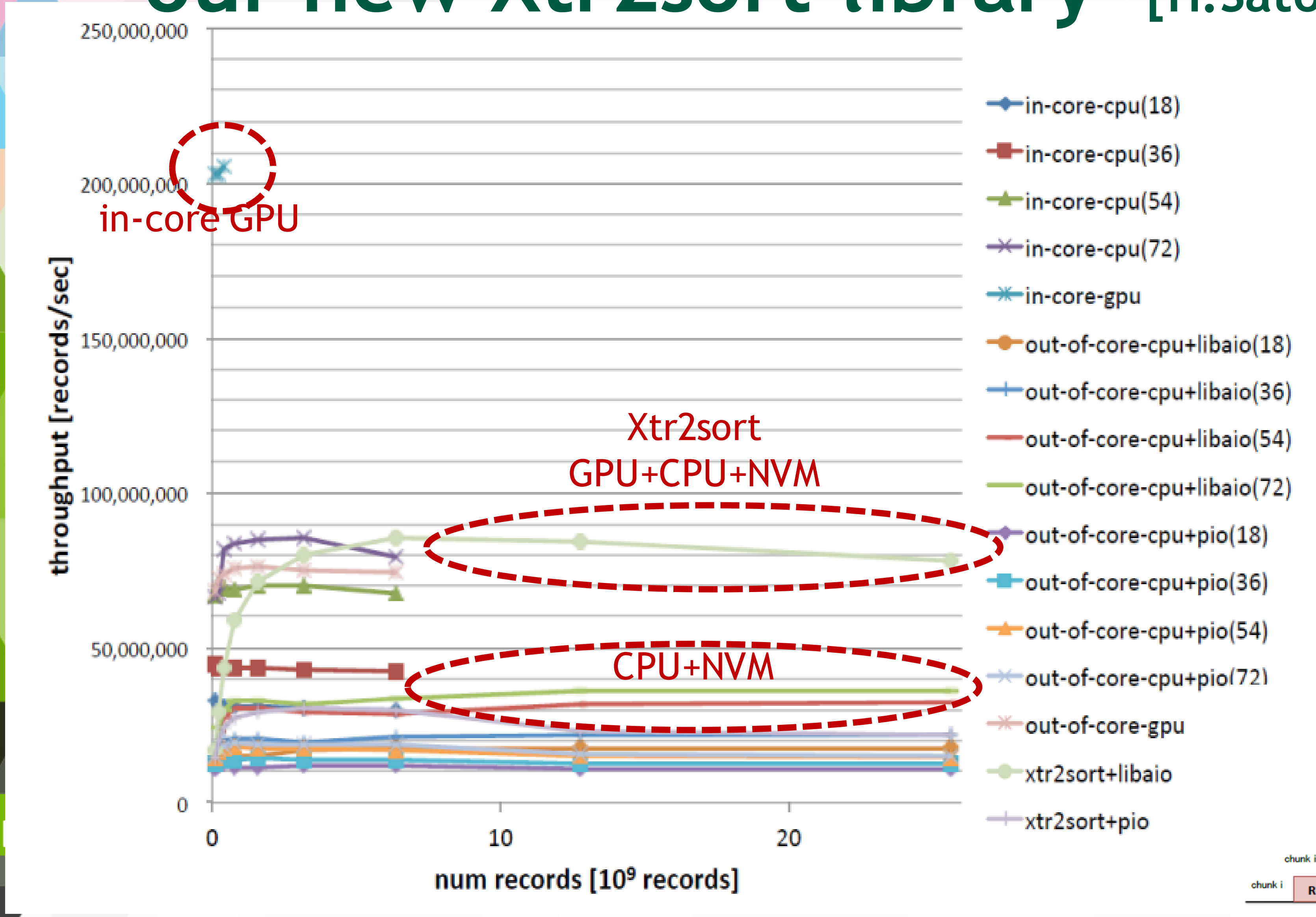


Performance prediction



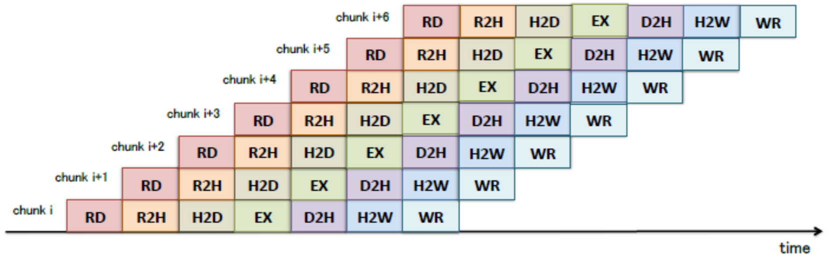
- PCIe_#: #GB/s bandwidth of interconnect between CPU and GPU

GPU + NVM + PCIe SSD Sorting our new Xtr2sort library [H.Sato et.al. SC15 Poster]



Single Node Xeon

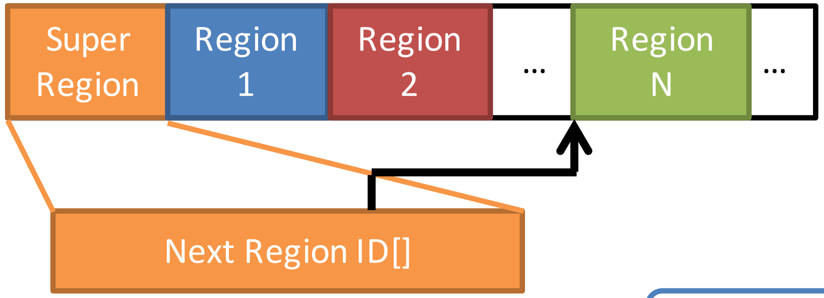
- 2 socket 36 cores
- 128GB DDR4
- K40 GPU (12GB)
- SSD PCIe card (2.4TB)

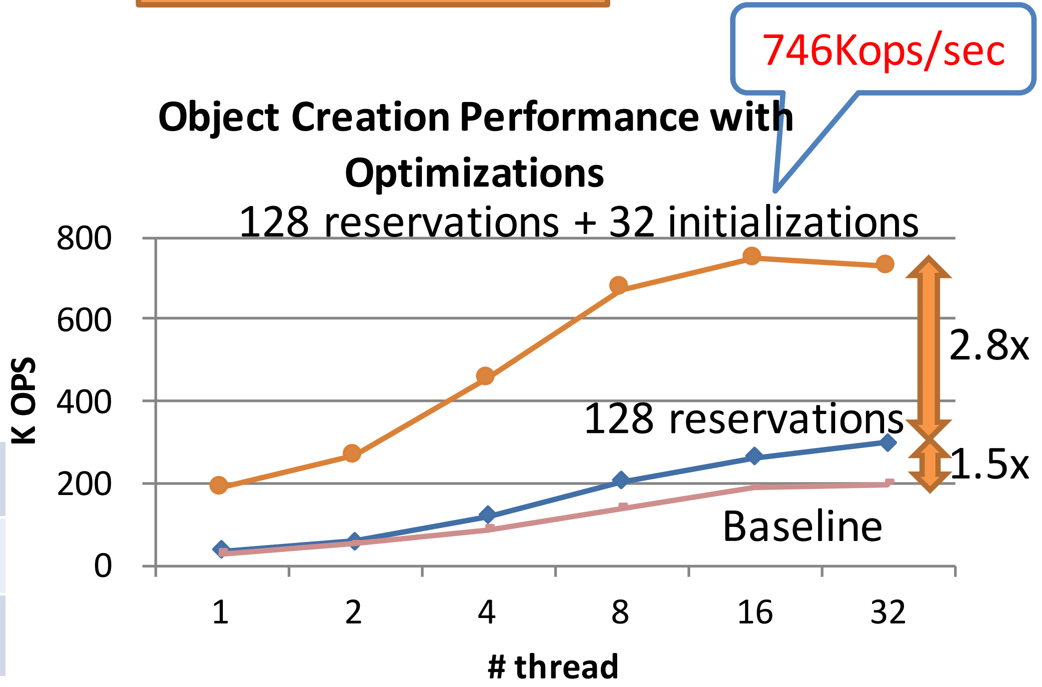


Object Storage Design in OpenNVM [Takatsu et al GPC 2015]

- New interface - Sparse address space, atomic batch operations and persistent trim
- Simple design by fixed-size Region enabled by sparse address space and persistent trim
  - Free'ed by persistent trim and no reuse
  - Enough region size to store one object
- Optimization techniques for object creation
  - Bulk reservation and bulk initialization

XFS	15.6 Kops/s
DirectFS	61.3 Kops/s
Proposal	746 Kops/s

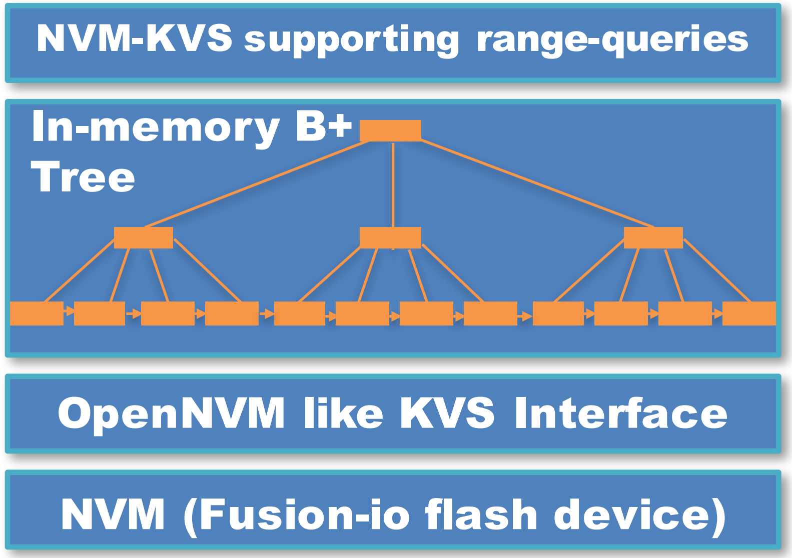


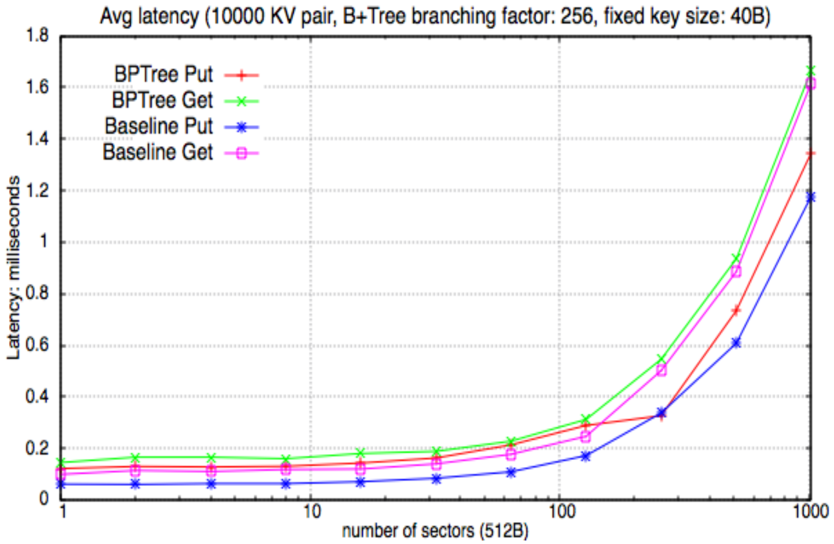


Fuyumasa Takatsu, Kohei Hiraga, and Osamu Tatebe, "Design of object storage using OpenNVM for high-performance distributed file system", the 10th International Conference on Green, Pervasive and Cloud Computing (GPC 2015), May 4, 2015

Concurrent B+Tree Index for Native NVM-KVS [Jabri]

- Enable range-queries support for KVS running natively on NVM like fusionio ioDrive
- Design of Lock-free concurrent B+Tree
  - Lock-free operations – search, insert and delete
  - Dynamic rebalancing of the Tree
  - Nodes to be split or merged are frozen until replaced by new nodes
- Asynchronous interface using future/promise in C++11/14





Performance Modeling of a Large Scale Asynchronous Deep Learning System under Realistic SGD Settings

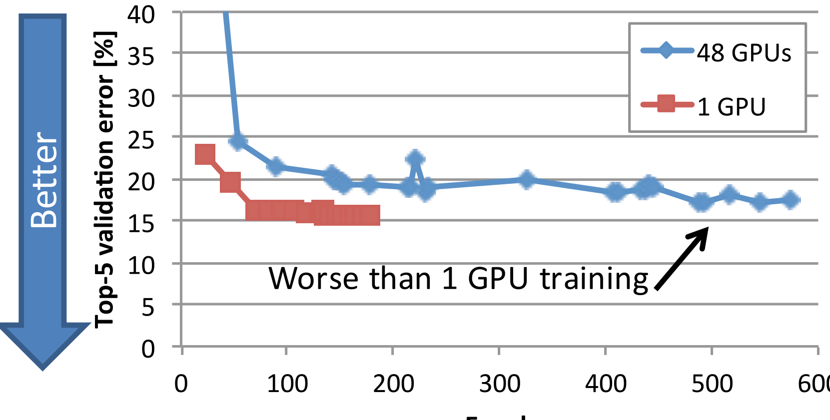
Yosuke Oyama[1], Akihiro Nomura[1], Ikuro Sato[2], Hiroki Nishimura[3], Yukimasa Tamatsu[3], and Satoshi Matsuoka[1]
[1]Tokyo Institute of Technology [2]DENSO IT LABORATORY, INC. [3]DENSO CORPORATION

Background

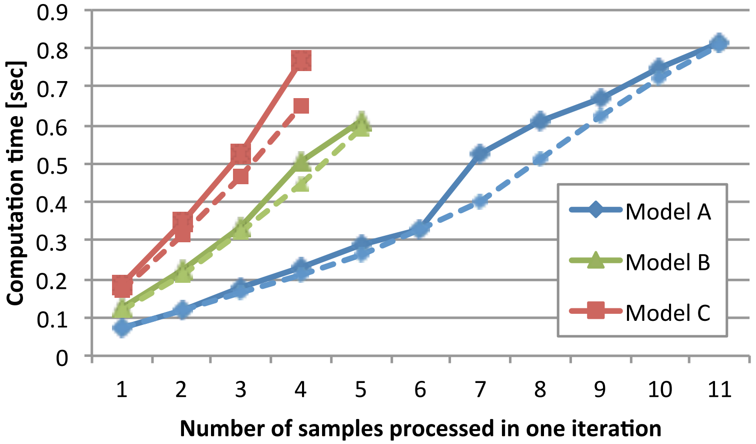
- Deep Convolutional Neural Networks (DCNNs) have achieved stage-of-the-art performance in various machine learning tasks such as image recognition
- Asynchronous Stochastic Gradient Descent (SGD) method has been proposed to accelerate DNN training
  - It may cause unrealistic training settings and degrade recognition accuracy on large scale systems, due to large non-trivial mini-batch size

Proposal and Evaluation

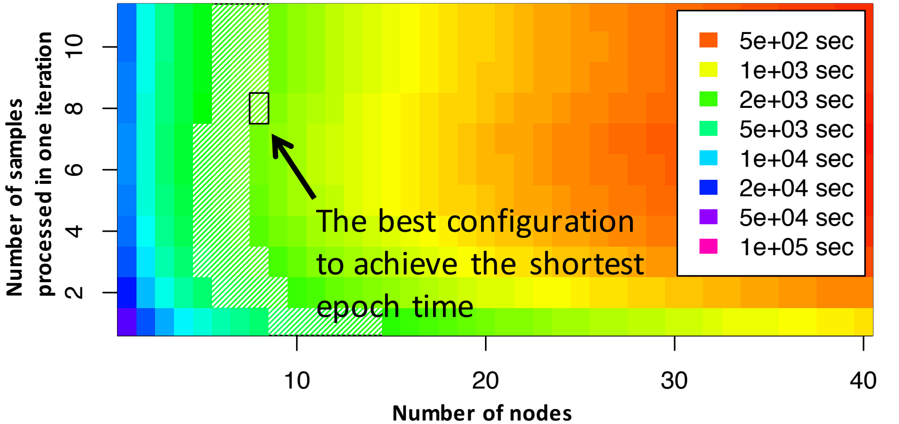
- We propose a empirical performance model for an ASGD training system on GPU supercomputers, which predicts CNN computation time and time to sweep entire dataset
  - Considering "effective mini-batch size", time-averaged mini-batch size as a criterion for training quality
- Our model achieves 8% prediction error for these metrics in average on a given platform, and steadily choose the fastest configuration on two different supercomputers which nearly meets a target effective mini-batch size



Validation Error of ILSVRC 2012 Classification Task on Two Platforms:
Trained 11 layer CNN with ASGD method



Measured Time (Solid) and Predicted Time (Dashed) of CNN Computation of Three 15-17 Layer Models



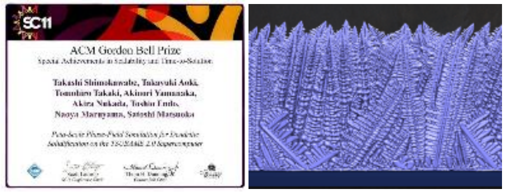
Predicted Epoch Time of ILSVRC 2012 Classification Task:
Shaded area indicate the effective mini-batch size is in 138±25%

2017 Q2 TSUBAME3.0 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"

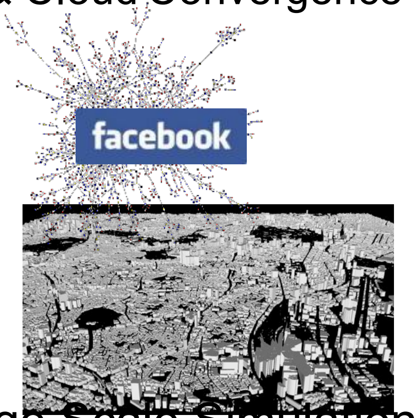


2011 ACM Gordon Bell Prize

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

2013 TSUBAME-KFC
#1 Green 500
/DL upgrade -> 1.5PF/rack

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

Comparison of Machine Learning / AI Capabilities

東京工業大学
Tokyo Institute of Technology

GSIC
Global Scientific Information and Computing Center

TSUBAME2.5(2013)
+TSUBAME3.0(2017) 8000GPUs

Deep Learning / AI Capabilities
FP16+FP32 <u>up to ~100 Petaflops</u>
<u>+ up to 100PB online storage</u>

X~10
>>
(effectively more due to optimized DL SW Stack on GPUs)

独立行政法人理化学研究所
計算科学研究機構
RIKEN
RIKEN Advanced Institute for Computational Science

K Computer (2011)

Deep Learning
<u>FP32 11.4 Petaflops</u>

BG/Q Sequoia (2011)
22 Petaflops SFP/DFP

2015 Proposal to MEXT – Big Data and HPC Convergent Infrastructure
=> "Nationoal Big Data Science Center" (Tokyo Tech GSIC)

- "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 + <u>data scientists</u> + CS/Infrastructure => Big data science center
- Data services including large data handling, big data structures e.g. graphs, ML/DNN/AI services…

<u>Present old style data science</u>

Domain labs segregated data facilities
No mutual collaborations
Inefficient, not scalable with
Not enough data scientists

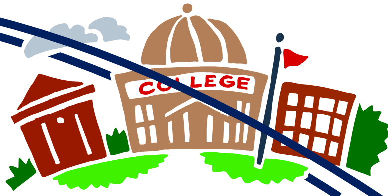
Main reason: We have shared resource HPC centers but no "Data Center" per se

Convergence of top-tier HPC and Big Data Infrastructure

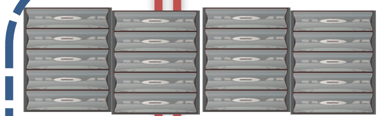
2013 TSUBAME2.5
Upgrade
5.7Petaflops 17PF DNN

2017Q1 TSUBAME3.0+2.5
Green&Big Data 60~80PF DNN
HPCI Leading Machine
Ultra-fast memory network, I/O

Data Management
Big Data Storage
Deep Learning
SW Infrastructure



National Labs With Data



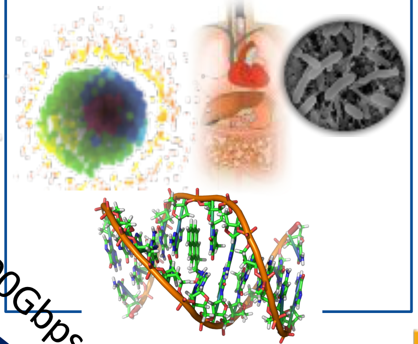
Mid-tier Parallel FS Storage

Archival Long-Term Object Store

Goal 100 Petabytes

SINET 5 400Gbps

Big Data Science Applications



100Gbps L2 Connection to commercial clouds

amazon webservices™

Virtual Multi-Institutional Data Science => People Convergence

TSUBAME4 beyond 2021~2022 K-in-a-Box (Golden Box) BD/EC Convergent Architecture

1/500 Size, 1/150 Power, 1/500 Cost, x5 DRAM+ NVM Memory

10 Petaflops, 10 Petabyte Hiearchical Memory (K: 1.5PB), 10K nodes

50GB/s Interconnect (200-300Tbps Bisection BW)

(Conceptually similar to HP “The Machine”)

Datacenter in a Box

Large Datacenter will become “Jurassic”

Acceleration of EBD Processing (1)

- Large Capacity – Multi-Terabytes, Petabytes, Exabytes
- Kernel algorithms for discrete data – graph, sort, etc.
  - EBD Characteristics
    - Sparse and random data structure
    - Involve frequent and abundant data transfer

Implies low latency and high bandwidth access

  - EBD Solutions (research)
    - High capacity at low power: non-volatile memory, deep memory hierarchy
    - High bandwidth: fast on-package memory + memory hierarchy+ Supercomputer Network (>100Gbps injection, Petabits bisection) + **bandwidth reducing algorithms for EBD**
    - Low Latency
      - latency reduction => memory 3-D stacking, fast on-package memory + low latency network
      - Latency hiding => many core + many threading + **latency reducing algorithms for EBD**

Our research: define & invent EBD architecture + algorithm + system SW

Acceleration of EBD Processing (2)

- Classification algorithms – statistical modeling/optimization, Machine Learning
  - EBD Characteristics: iterative numerical optimization
    - Kernel may be sparse (e.g., SVM) or dense (e.g., Deep Learning)
    - Parallelism difficult due to massive sample size (10~100 billion images)
  - EBD Solutions (our research)
    - Approach: Employ traditional and new HPC/supercomputer parallelization and acceleration strategies
    - Sparse algorithms – high bandwidth processors (e.g., GPU) w/stacked memory and on-package memory + memory hierarchy + supercomputing network + **<u>bandwidth reducing algorithms</u> (sparse linear algebra)**
    - Dense algorithms – many-core high FLOPS processor (e.g., GPU) + **<u>algorithmic advances for strong scaling</u>**
    - High volume data – **<u>utilize "burst buffer" technology (incl. Clouds)</u>**

Limited showing today

Optimized Graph500 program (1) – Bandwidth Reducing Algorithm
Sparse Matrix Representation with Bitmap

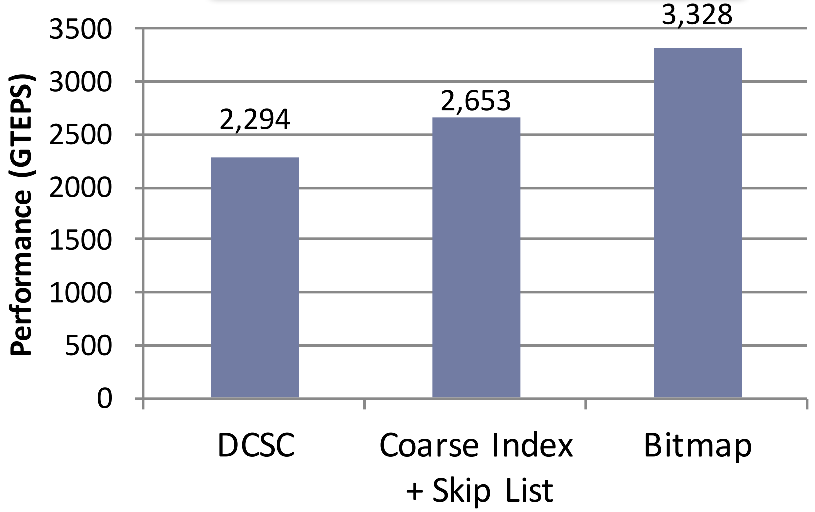
- Problem
  - Since the partitioned graph is a hyper sparse matrix, we need efficient hyper sparse matrix representation for large scale distributed graph processing.
- Our proposal: Sparse Matrix Representation with Bitmap
  - Enables **compression of row indexes** and **fast access to each row**.

Comparison with other methods

Data size of row index (MB/node)
(8064 partition, Scale 36)

CSR (Compressed Sparse Row)	1806
DCSC	861
Coarse Index + Skip List	309
Bitmap (Proposal)	**337**

Performance
(8064 nodes, Scale 36)



Optimized Graph500 program (2) – Bandwidth Reducing Algorithm Vertex Reordering for Bitmap Optimization

- Our idea
  - Creates reordered vertex number by sorting vertices by degree.
  - Use reordered number for bitmap access and original number for other processing.
- Result
  - 16% speedup by reduction of bitmap data, 28% speedup by localized memory access, and 49% speedup in total. (8064 nodes)