The first "exascale" supercomputer Fugaku & beyond

- *Satoshi Matsuoka*
- *Director, RIKEN Center for Computational Science*

- **20190815 Modsim Presentation**

RIKEN R-CCS

Arm64fx & Fugaku 富岳 /Post-K are:

- **Fujitsu-Riken design A64fx ARM v8.2 (SVE), 48/52 core CPU**
  - ***HPC Optimized:*** Extremely high package high memory BW (1TByte/s), on-die Tofu-D network BW (~400Gbps), high SVE FLOPS (~3Teraflops), various AI support (FP16, INT8, etc.)
  - Gen purpose CPU – Linux, Windows (Word), other SCs/Clouds
  - Extremely power efficient – > **<u>10x power/perf efficiency for CFD benchmark</u>** over current mainstream x86 CPU
- **Largest and fastest supercomputer to be ever built circa 2020**
  - > 150,000 nodes, superseding LLNL Sequoia
  - > 150 PetaByte/s memory BW
  - Tofu-D 6D Torus NW, 60 Petabps injection BW (10x global IDC traffic)
  - 25~30PB NVMe L1 storage
  - many endpoint 100Gbps I/O network into Lustre
  - The first 'exascale' machine (not exa64bitflops but in apps perf.)

Brief History of R-CCS towards Fugaku

2006 — 2010 — 2011 — 2012 — 2013 — 2014 — 2018 — 2019

January 2006
Next Generation Supercomputer Project (K Computer) start

July 2010
RIKEN AICS established
August 2010
HPCI Project start
September 2010
K computer installation starts
First meeting of SDHPC (Post-K)

June 2011
#1 on Top 500
November 2011
#1 on Top 500 > 10 Petaflops
ACM Gordon Bell Award
End of FY 2011 (March 2012)
SDHPC Whitepaper

April 2012
Post-K Feasibility Study start
3 Arch Teams and 1 Apps Team
June 2012
K computer construction complete
September 2012
K computer production start
November 2012
ACM Gordon bell Award

End of FY2013 (Mar 2014)
Post-K Feasibility Study Reports

April 2014
Post-K project start
June 2014
#1 on Graph 500

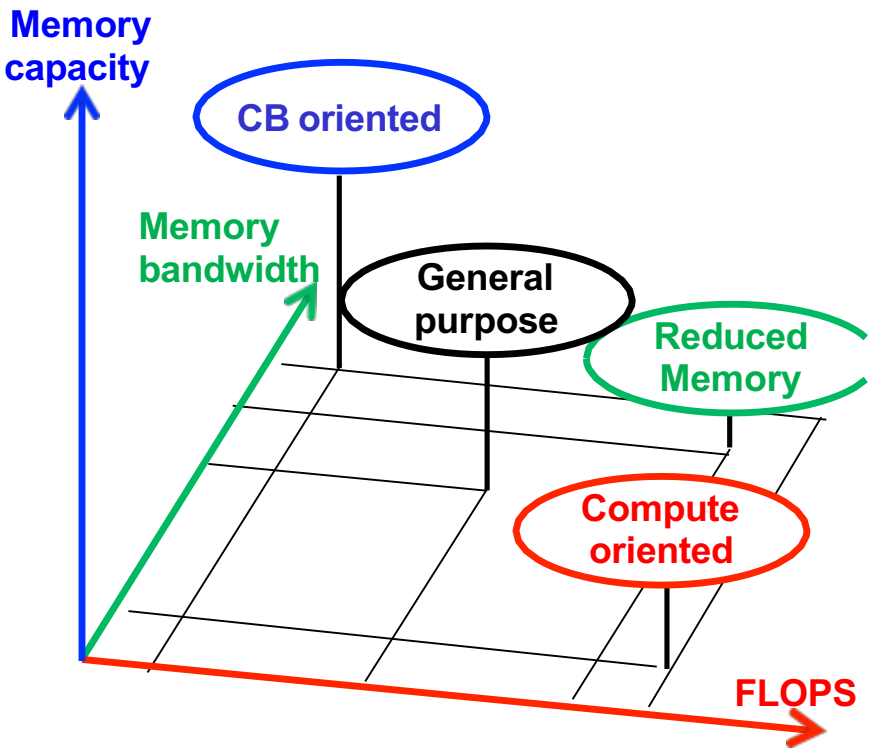
April 2018
AICS renamed to RIKEN R-CCS.
Satoshi Matsuoka becomes new Director
Aug 2018
Arm A64fx announce at Hotchips
Oct 2018
NEDO 100x processor project start
Nov 2018
Post-K Manufacturing approval by Prime Minister's CSTI Committee

March 2019
Post-K Manufacturing start
May 2019
Post-K named "Supercomputer Fugaku"
July 2019
Post-Moore Whitepaper start
Aug 2019
K Computer shutdown
Dec 2019
Fugaku installation start (planned)

SDHPC (2011-2012) Candidate of ExaScale Architecture

https://www.exascale.org/mediawiki/images/a/aa/Talk-3-kondo.pdf

- Four types of architectures are considered
  - General Purpose (GP)
    - Ordinary CPU-based MPPs
    - e.g.) K-Computer, GPU, Blue Gene, x86-based PC-clusters
  - Capacity-Bandwidth oriented (CB)
    - With expensive memory-I/F rather than computing capability
    - e.g.) Vector machines
  - Reduced Memory (RM)
    - With embedded (main) memory
    - e.g.) SoC, MD-GRAPE4, Anton
  - Compute Oriented (CO)
    - Many processing units
    - e.g.) ClearSpeed, GRAPE-DR



SDHPC (2011-2012) Performance Projection

- Performance projection for an HPC system in 2018
  - Achieved through continuous technology development
  - Constraints: 20 – 30MW electricity & 2000sqm space

Node Performance	Total CPU Performance (PetaFLOPS)	Total Memory Bandwidth (PetaByte/s)	Total Memory Capacity (PetaByte)	Byte / Flop
General Purpose	200~400	20~40	20~40	0.1
Capacity-BW Oriented	50~100	50~100	50~100	1.0
Reduced Memory	500~1000	250~500	0.1~0.2	0.5
Compute Oriented	1000~2000	5~10	5~10	0.005

Network

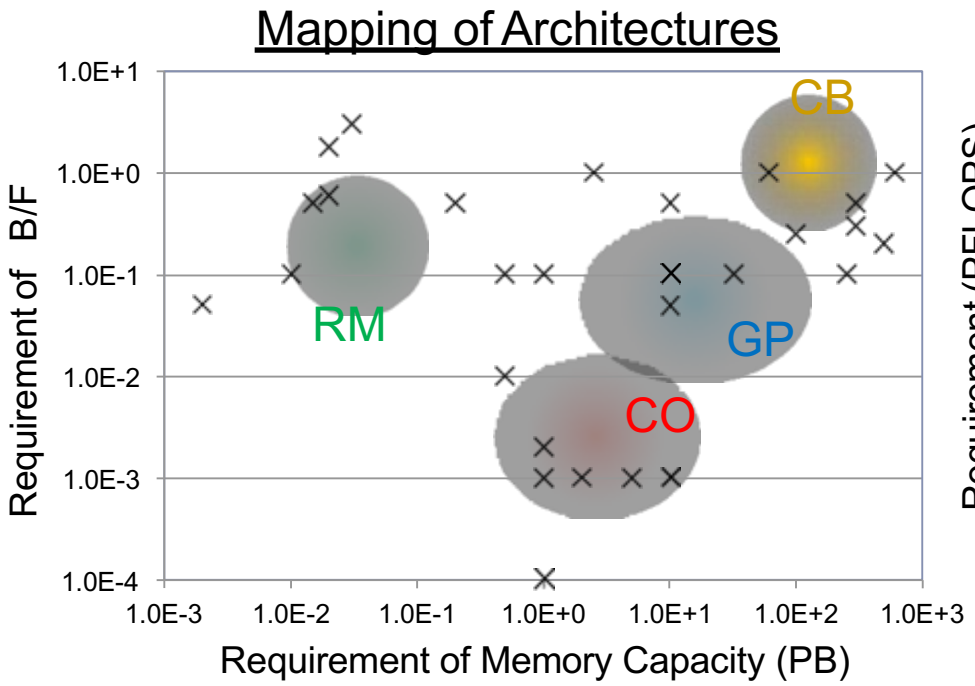
	Injection	P-to-P	Bisection	Min Latency	Max Latency
High-radix (Dragonfly)	32 GB/s	32 GB/s	2.0 PB/s	200 ns	1000 ns
Low-radix (4D Torus)	128 GB/s	16 GB/s	0.13 PB/s	100 ns	5000 ns

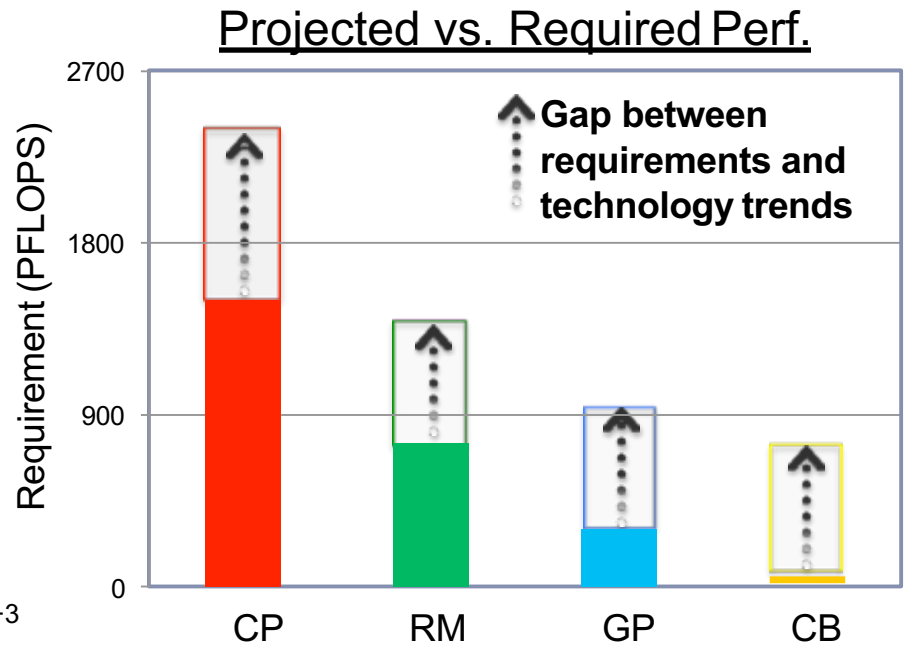
Storage

Total Capacity	Total Bandwidth
1 EB	10TB/s
100 times larger than main memory	For saving all data in memory to disks within 1000-sec.

SDHPC (2011-2012) Gap Between Requirement and Technology Trends

- Mapping four architectures onto science requirement
- Projected performance vs. science requirement
  - Big gap between projected and required performance





Needs national research project for science-driven HPC systems

Post-K Feasibility Study (2012-2013)

- **3 Architecture Teams, from identified architectural types in the SDHPC report**
  - General Purpose --- balanced
  - Compute Intensive --- high flops and/or low memory capacity & high memory BW
  - Large Memory Capacity --- also w/high memory BW
- **The A64fx processor satisfied multiple roles - basically balanced but also compute intensive**

- **Application Team (Tomita, Matsuoka)**
  - Put all the K-Computer applications stakeholders into one room
  - Templated reporting of science impact possible on exascale machines and their computational algorithms / requirements
  - 600 page report (English summary available)

Post-K Application Feasibility Study 2012-2013

https://hpci-aplfs.r-ccs.riken.jp/document/roadmap/roadmap_e_1405.pdf

Computational Science Roadmap -Overview-

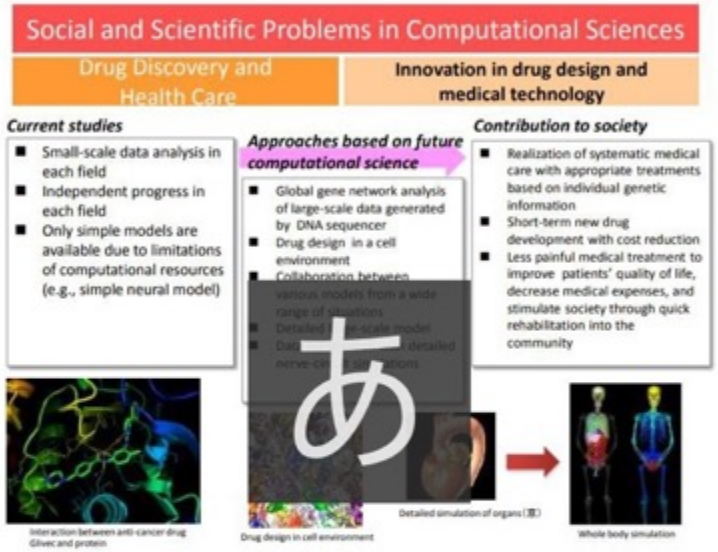
Social Contributions and Scientific Outcomes Aimed for by Innovations through Large-Scale Parallel Computing

May, 2014

Feasibility Study on Future HPC Infrastructures

(Application Working Group)

mechanisms, such as blood clot formation in the heart or brain infarctions, and will be effective in improving patients' Quality of Life (QOL) through the development of minimally invasive treatments, which only pose a slight burden to the patient, and of the medical devices required for these treatments. It will further be effective in revitalizing society through patients' early re-entry into the community and in reducing costs of medical treatment.



The supercomputer's vast computational power will undoubtedly greatly contribute to the development of various aspects in the field of life science, such as detailed neural and cellular simulations, simulations over extended periods of time and space, and almost real-time assimilation[4] of those data. Eventually it could form an important scientific basis for innovative drug design and medical technologies.

The table below lists the computational performance required in the future for the respective areas of drug discovery and healthcare.

[4] One of the methods to merge different observational and experimental data into a numerical model at a high degree.

Subject	Performance (PFLOPS)	Memory bandwidth (PB/s)	Memory size per case (PB)	Storage size per case (PB)	Elapse Time /Case (hour)	Number of Cases	Total operation count (EFLOP)	Summary and numerical method	Problem size	Notes
Personal Genome Analysis	0.0054	0.0001	1.6	0.1	0.7	200000	2700	Sequence matching	Cancer Genome Analysis: Short read mapping and mutation identification of 200,000 people's genome	1 case = 1 person Integer operations are dominant. "Total operation count" total instruction count (Total FLOP = 46 EFLO
Gene Network Analysis	25	89	0.08	0.016	0.34	26000	780000	Baysian network estimation and L1-regularization	40,000 transcripts x 26,000 data sets consisting of 2,800,000 arrays	
MD and Free-energy calculation for drug design and so on	1000	400	0.0001		0.0012	1000000	4300000	Molecular dynamics simulation with all-atom model	Number of Cases: 100,000 ligands X 10 target proteins	B/F=0.4. Supposed to run 100-1000 cases simultaneously. Memory size per case is estimated for a 100 node run.
MD simulations under cellular environments or MD simulations of Virus	490	49	0.2	1.2	48	10	[illegible]	Molecular dynamics simulations with all-atom or coarse-grained model	100,000,000 particles	B/F=0.1
Simulations of cellular signaling pathways	42	100	10	10	[illegible]	[illegible]	[illegible]	[illegible]	1,000 to 10,000 cells	integer operations
Precise Structure-Based Drug Design	0.83	0.14	1	0.001	1	100	[illegible]	[illegible] chemical calculations on the interactions between proteins and drugs	proteins (500 residues) + ligands in solution	1TB/s IO speed required to dump 1TB dataset per second
Design of Biological Devices	1.1	0.19	1	0.001	1	100	400	Spectroscopic analyses of proteins (200-500 residues)	more than 100,000 orbitals	1TB/s IO speed required to dump 1TB dataset per second
Multi-scale simulation of a blood clot	400	64	1	1	170	10	2500000	Semi-implicit FDM simulation of fluid-structure interaction with chemical factors	Length:100mm, D:100um, Calculation Time:10s, Grid size:0.1um, Velocity:10^{-2}m/s, Delta T:1us	
High Intensity Focused Ultrasound	380	460	54	64	240	10	3300000	Explicit FDM simulation of sound wave and heat transfer	Calculation Area:400mm^3, Grid: 225×10^{12}, Steps: 1459200, FLOP/grid/step: 1000	
Simulations of Brain and Neural Systems	* 6.9	* 7.6	* 56	* 3600	0.28	100	700	Single compartment model	100 billion neuons, 10000 synapses/neuron, 10^5steps	
Data assimulation of whole insect brain via communication between a phisological experiment and a simulation. Parameter estimator in insect brain simulation	* 71	* 60	* 0.2	* 20	28	20	140000	Multi-compartment HH model with local Crank-Nicolson method, evolutionary algorithm	1000 neurons, 10^6 genes, 100 generations	Supposing 100 MB/s communication to external environment will be required

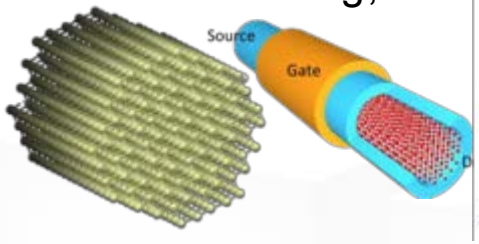
*Figures marked with a * are still under examination. The website will show more accurate figures as they become available.*

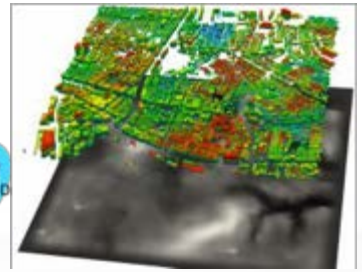
Co-Design Activities in Fugaku

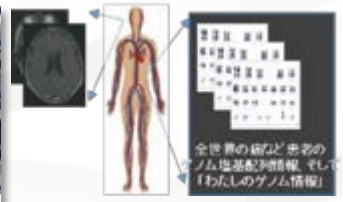
Multiple Activities since 2011

Science by Computing

・9 Priority App Areas: High Concern to General Public: Medical/Pharma, Environment/Disaster, Energy, Manufacturing, …



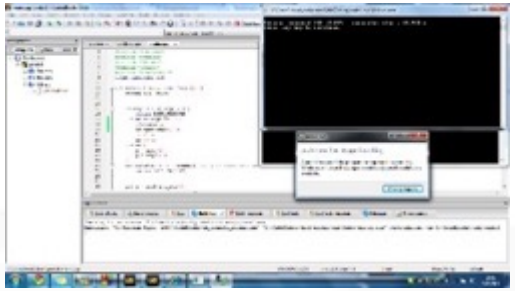




Select representatives from 100s of applications signifying various computational characteristics

Design systems with parameters that consider various application characteristics

Science of Computing



- Extremely tight collabrations between the Co-Design apps centers, Riken, and Fujitsu, etc.
- Chose 9 representative apps as “target application” scenario
- Achieve up to x100 speedup c.f. K-Computer
- Also ease-of-programming, broad SW ecosystem, very low power, …

Research Subjects of the Post-K Computer

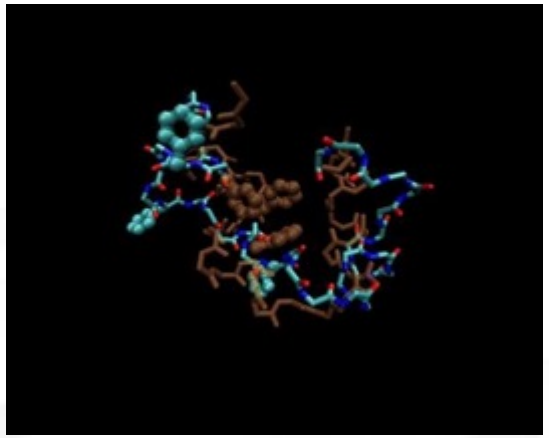
The post K computer will expand the fields pioneered by the K computer, and also challenge new areas.

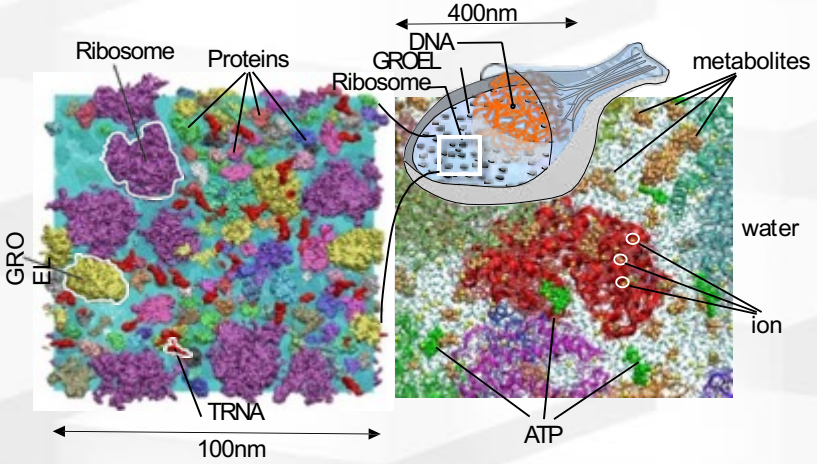
Genesis MD: proteins in a cell environment

Protein simulation before K

- Simulation of a protein in isolation

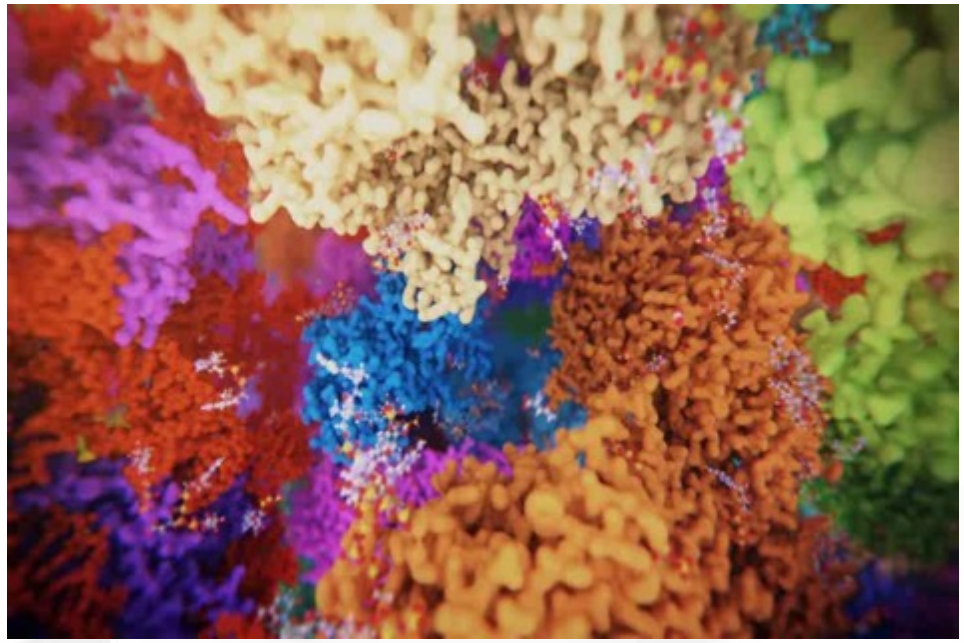
Folding simulation of Villin, a small protein with 36 amino acids





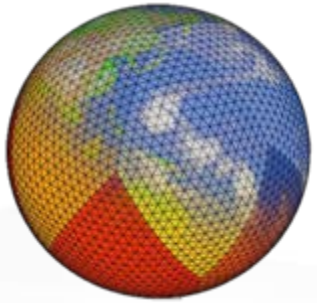
Protein simulation with K

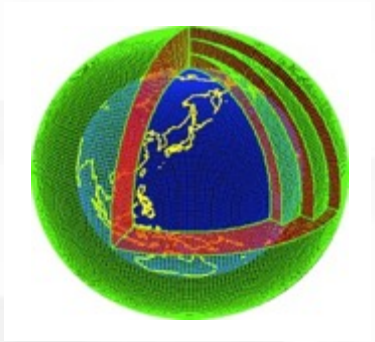
- all atom simulation of a cell interior
- cytoplasm of Mycoplasma genitalium



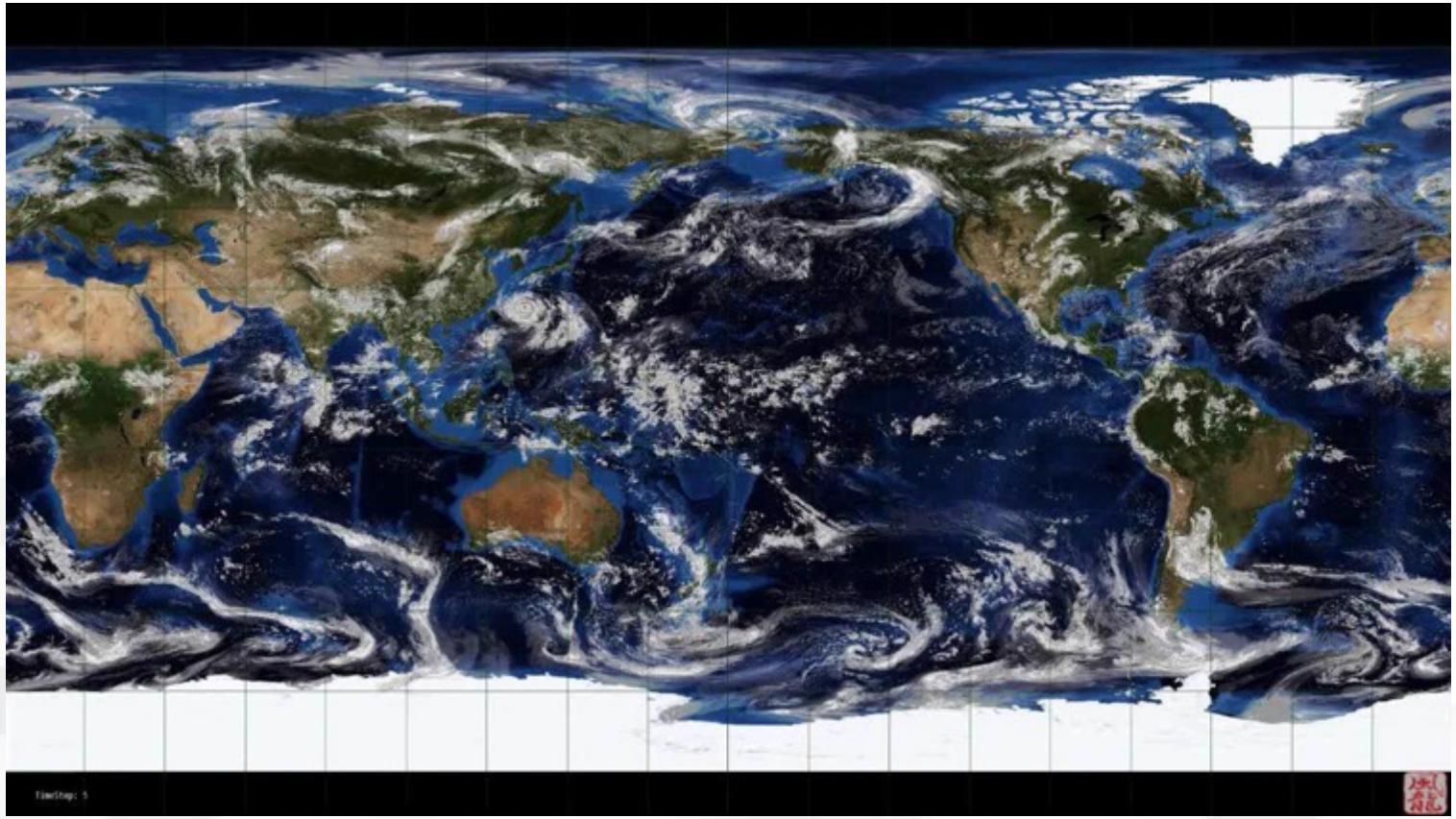
NICAM: Global Climate Simulation

- Global cloud resolving model with 0.87 km-mesh which allows resolution of cumulus clouds
- Month-long forecasts of Madden-Julian oscillations in the tropics is realized.





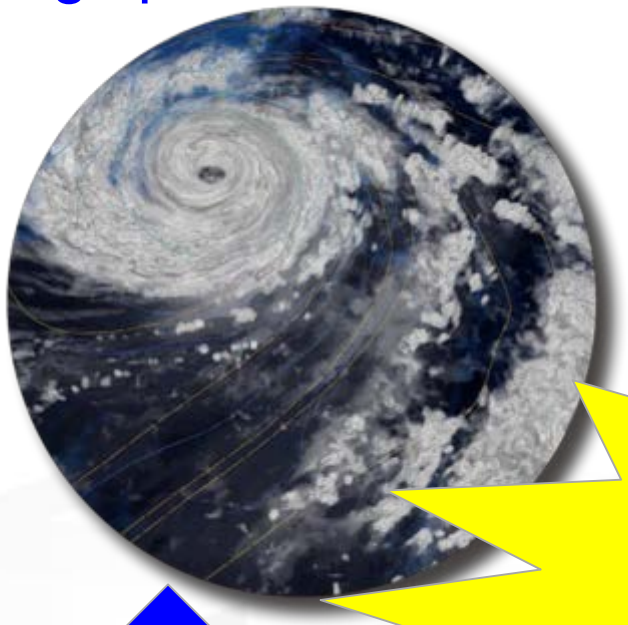
Global cloud resolving model



Miyamoto et al (2013) , Geophys. Res. Lett., 40, 4922–4926, doi:10.1002/grl.50944.

“Big Data Assimilation” NICAM+LETKF

High-precision Simulations

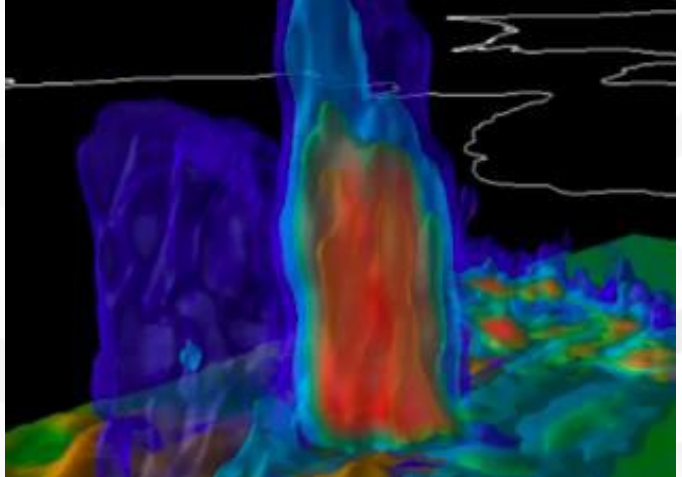


国立研究開発法人 科学技術振興機構 CREST
Japan Science and Technology Agency

Future-generation technologies available 10 years in advance

BDA
BIG DATA ASSIMILATION

High-precision observations



Mutual feedback

Co-design from Apps to Architecture

- **Architectural Parameters to be determined**
  - #SIMD, SIMD length, #core, #NUMA node, O3 resources, specialized hardware
  - cache (size and bandwidth), memory technologies
  - Chip die-size, power consumption
  - Interconnect
- **We have selected a set of target applications**
- **Performance estimation tool**
  - Performance projection using Fujitsu FX100 execution profile to a set of arch. parameters.
- **Co-design Methodology (at early design phase)**

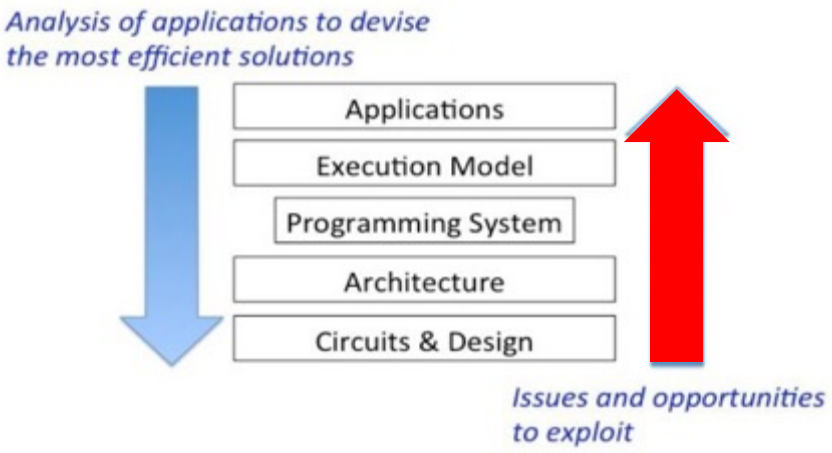
1. Setting set of system parameters
2. Tuning target applications under the system parameters
3. Evaluating execution time using prediction tools
4. Identifying hardware bottlenecks and changing the set of system parameters

Target applications representatives of almost all our applications in terms of computational methods and communication patterns in order to design architectural features.

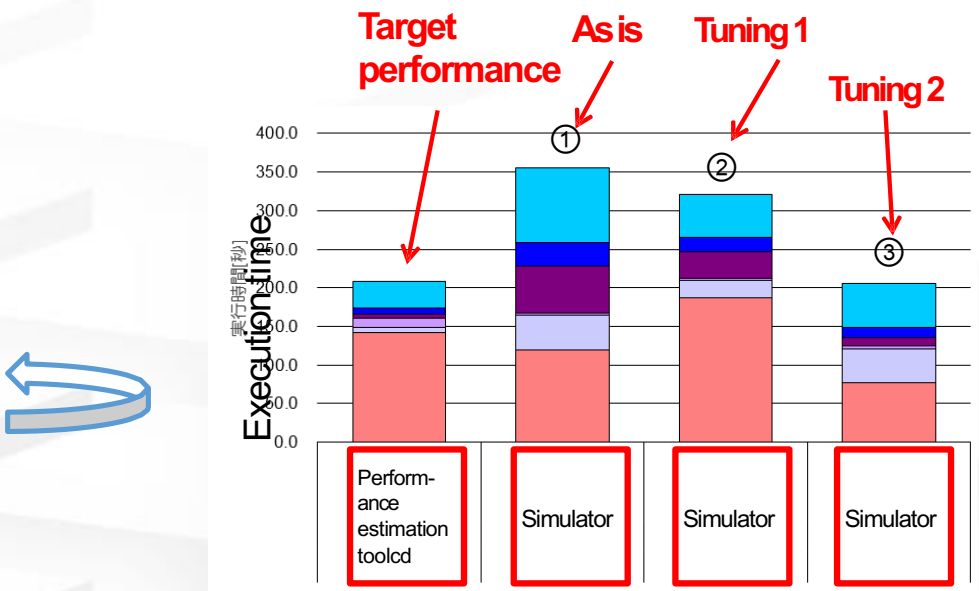
	Target Application	
	Program	**Brief description**
①	GENESIS	MD for proteins
②	Genomon	Genome processing (Genome alignment)
③	GAMERA	Earthquake simulator (FEM in unstructured & structured grid)
④	NICAM+LETK	Weather prediction system using Big data (structured grid stencil & ensemble Kalman filter)
⑤	NTChem	molecular electronic (structure calculation)
⑥	FFB	Large Eddy Simulation (unstructured grid)
⑦	RSDFT	an ab-initio program (density functional theory)
⑧	Adventure	Computational Mechanics System for Large Scale Analysis and Design (unstructured grid)
⑨	CCS-QCD	Lattice QCD simulation (structured grid Monte Carlo)

Co-design of Apps for Architecture

- **Tools for performance tuning**
  - Performance estimation tool
    - Performance projection using Fujitsu FX100 execution profile
    - Gives "target" performance
  - **Post-K processor simulator**
    - **Based on gem5, O3, cycle-level simulation**
    - **Very slow, so limited to kernel-level evaluation**



- **Co-design of apps**
  - 1. Estimate "target" performance using performance estimation tool
  - 2. Extract kernel code for simulator
  - 3. Measure exec time using simulator
  - 4. Feed-back to code optimization
  - 5. Feed-back to compiler

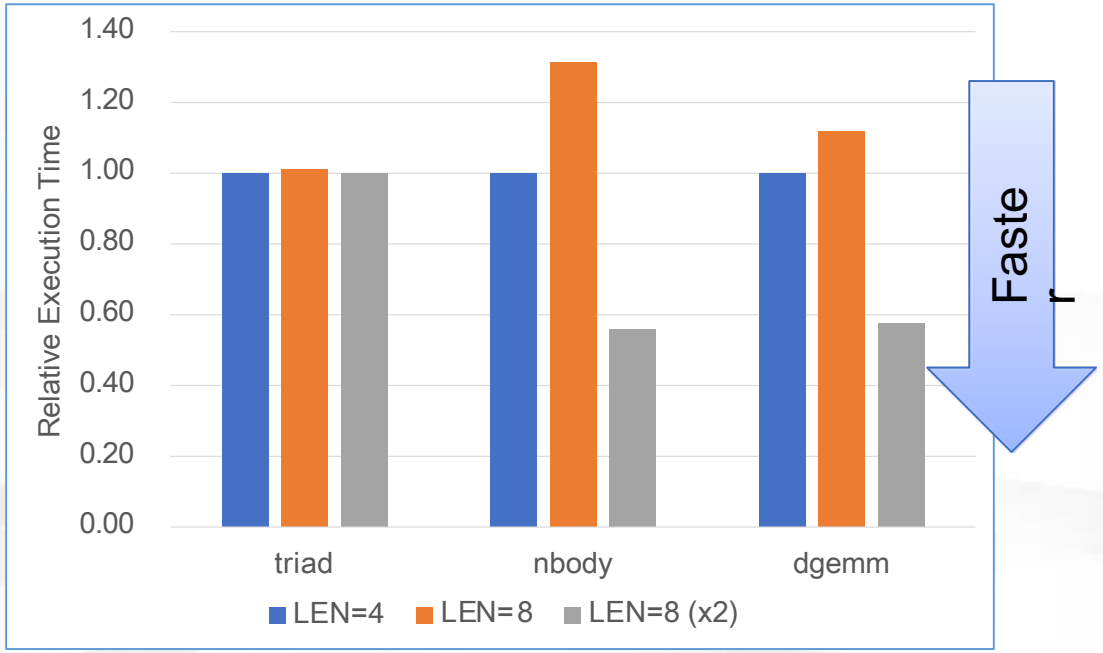


ARM for HPC - Co-design Opportunities

- ARM SVE Vector Length Agnostic feature is very interesting, since we can examine vector performance using the same binary.
- We have investigated how to improve the performance of SVE keeping hardware-resource the same. (in "Rev-A" paper)
  - ex. "512 bits SVE x 2 pipes" vs. "1024 bits SVE x 1 pipe"
  - Evaluation of Performance and Power (in "coolchips" paper) by using our gem-5 simulator (with "white" parameter) and ARM compiler.
  - Conclusion: Wide vector size over FPU element size will improve performance if there are enough rename registers and the utilization of FPU has room for improvement.

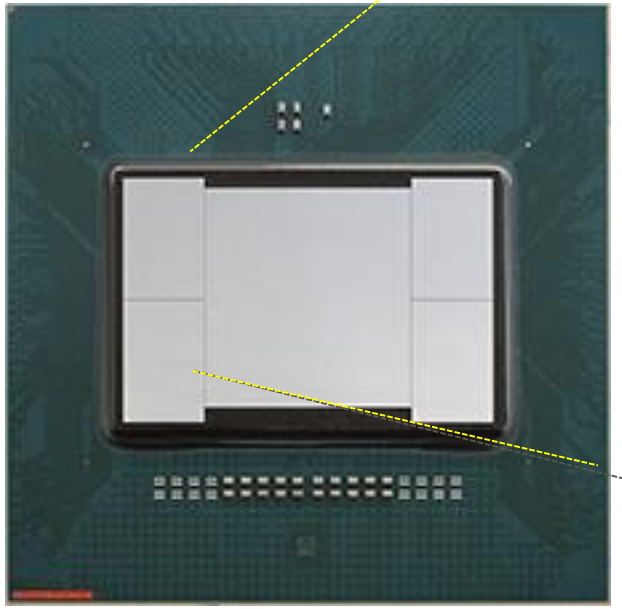
Note that these researches are not relevant to "post-K" architecture.

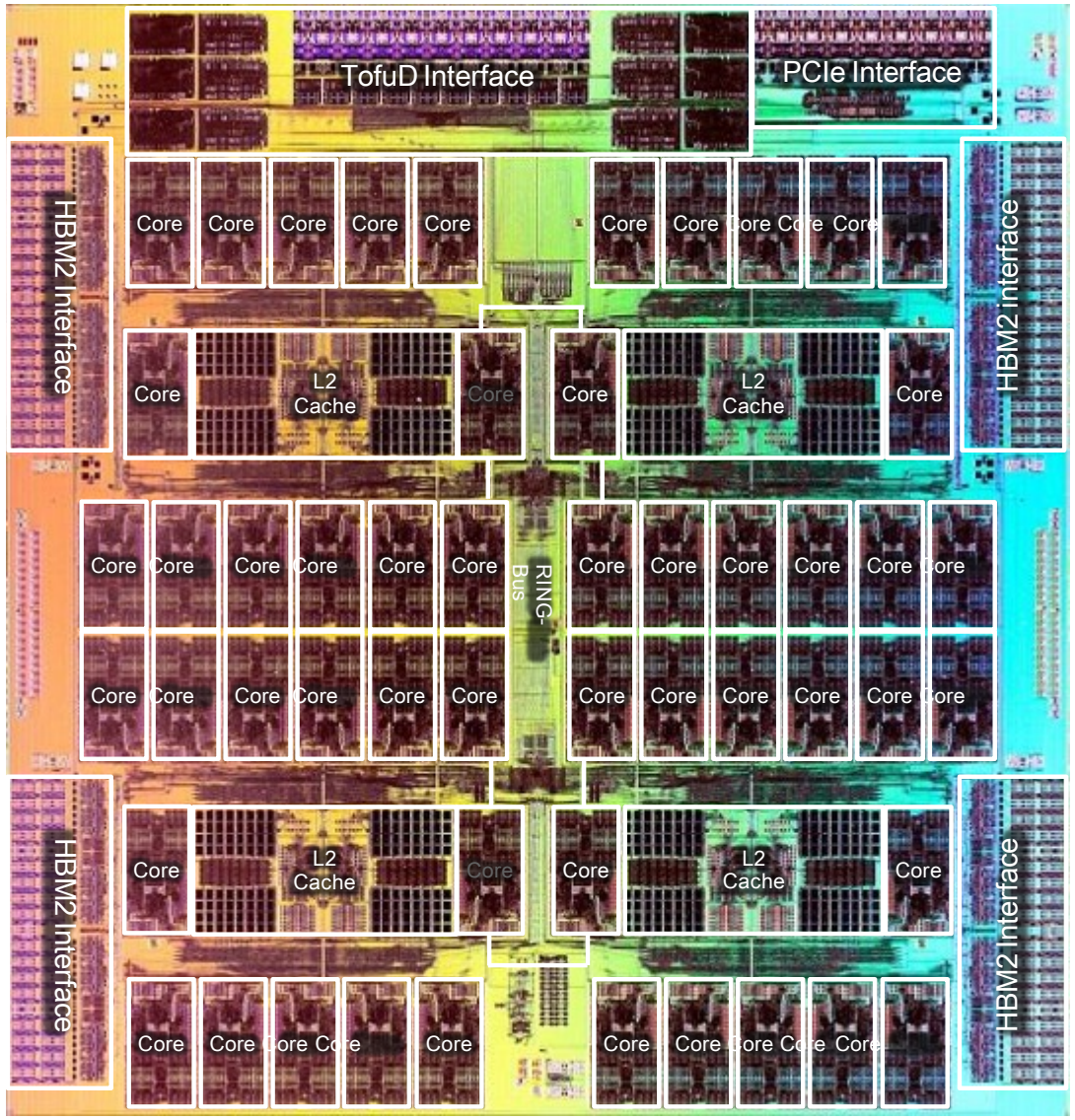
- Y. Kodama, T. Oajima and M. Sato. "Preliminary Performance Evaluation of Application Kernels Using ARM SVE with Multiple Vector Lengths", In Re-Emergence of Vector Architectures Workshop (Rev-A) in 2017 IEEE International Conference on Cluster Computing, pp. 677-684, Sep. 2017.
- T. Odajima, Y. Kodama and M. Sato, "Power Performance Analysis of ARM Scalable Vector Extension", In IEEE Symposium on Low-Power and High-Speed Chips and Systems (COOL Chips 21), Apr. 2018



A64FX Leading-edge Si-technology

- TSMC 7nm FinFET & CoWoS
  - Broadcom SerDes, HBM I/O, and SRAMs
  - 8.786 billion transistors
  - 594 signal pins





Copyright 2019 FUJITSU LIMITED

Fugaku: The Game Changer

1. **Heritage of the K-Computer, HP in simulation via extensive Co-Design**

- High performance: up to x100 performance of Kin real applications
- Multitudes of Scientific Breakthroughs via Fugaku application programs
- Simultaneous high performance and ease-of-programming

2. New Technology Innovations of Fugaku

- **High Performance, esp. via high memory BW**

  Performance boost by "factors" c.f. mainstream CPUs in many HPC & Society5.0 apps via BW & Vector acceleration

- **Very Green e.g. extreme power efficiency**

  Ultra Power efficient design & various power control knobs

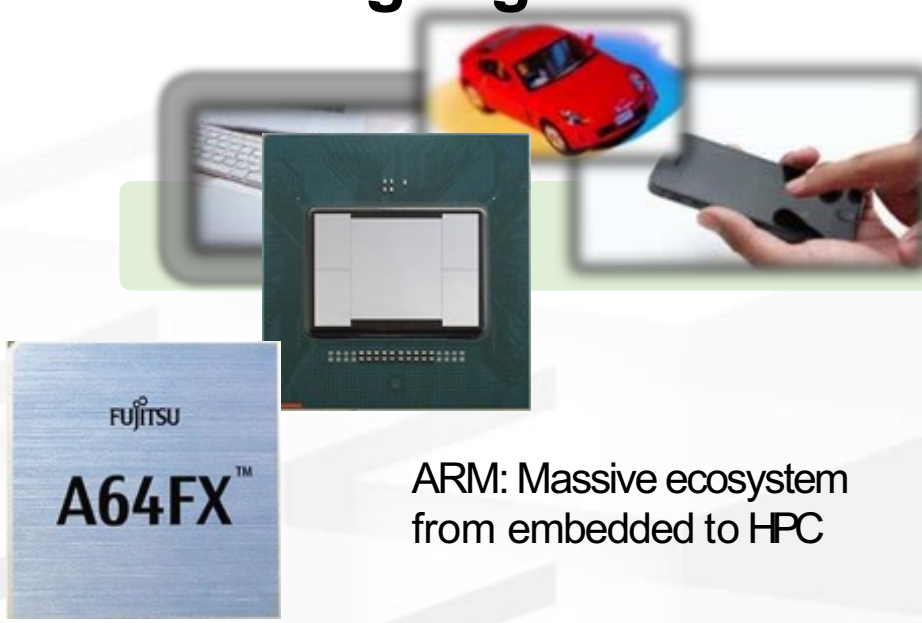
- **Arm Global Ecosystem & SVE contribution**

  Top CPU in ARM Ecosystem of 21 billion chips/year, SVE co-design and world's first implementation by Fujitsu

- **High Perf. on Society5.0 apps incl. AI**

  Architectural features for high perf on Society 5.0 apps based on Big Data, AI/ML, CAE/EDA, Blockchain security, etc.

Global leadership not just in the machine & apps, but as cutting edge IT

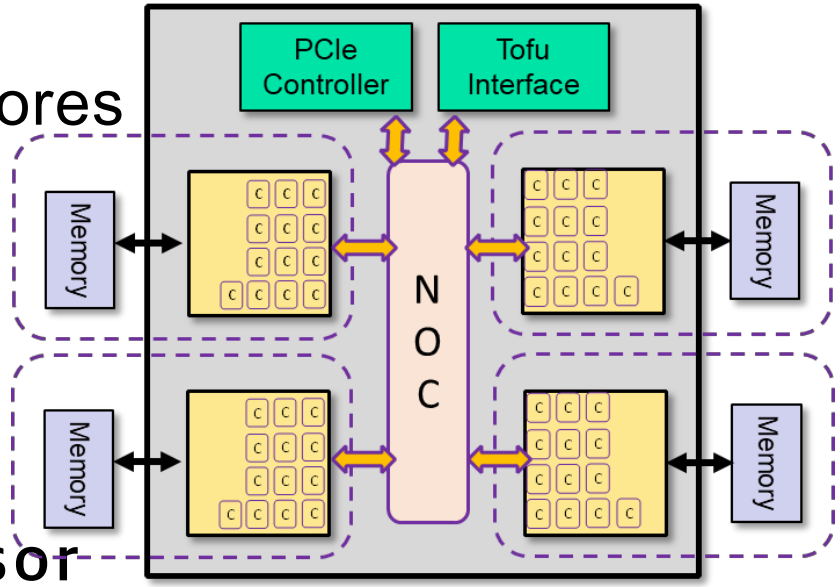


ARM: Massive ecosystem from embedded to HPC

Technology not just limited to Fugaku, but into societal IT infrastructures e.g. Clouds

Fugaku's FUjitsu A64fx Processor is…

- **an Many-Core ARM CPU…**
  - 48 compute cores + 2 or 4 assistant (OS) cores
  - Brand new core design
  - Near Xeon-Class Integer performance core
  - ARM V8 --- 64bit ARM ecosystem
  - Tofu-D + PCIe 3 external connection



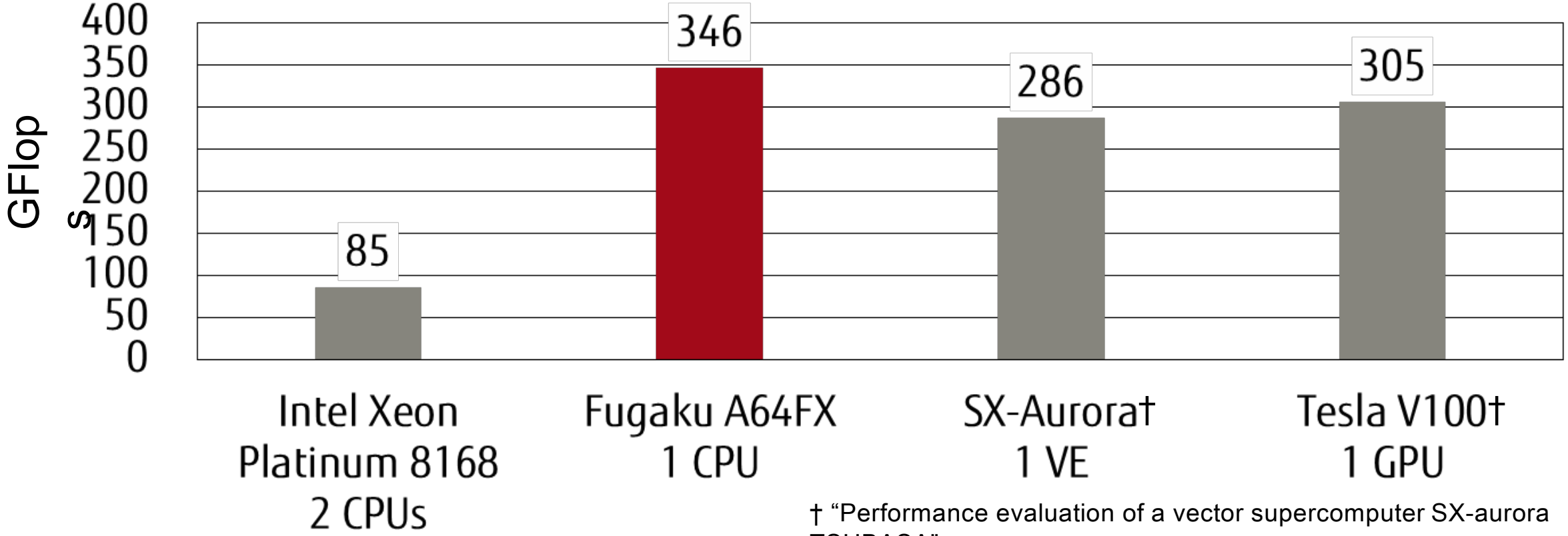
- **…but also an accelerated GPU-like processor**
  - SVE 512 bit x 2 vector extensions (ARM & Fujitsu)
    - Integer (1, 2, 4, 8 bytes) + Float (16, 32, 64 bytes)
  - Cache + scratchpad-like local memory (sector cache)
  - HBM2 on package memory – Massive Mem BW (Bytes/DPF ~0.4)
    - Streaming memory access, strided access, scatter/gather etc.
  - Intra-chip barrier synch. and other memory enhancing features

- **GPU-like High performance in HPC, AI/Big Data, Auto Driving…**

“Fugaku” CPU Performance Evaluation (2/3)

FUJITSU

- Himeno Benchmark (Fortran90)
  - Stencil calculation to solve Poisson’s equation by Jacobi method



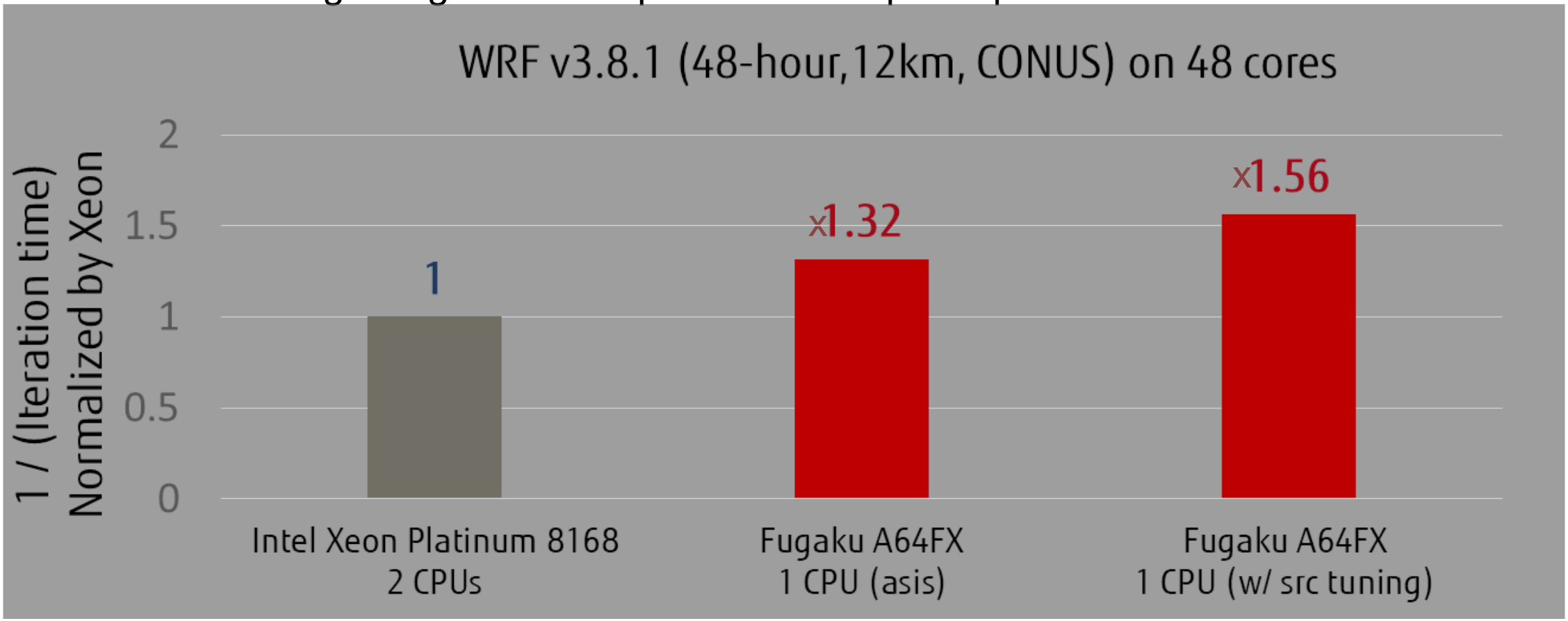
† “Performance evaluation of a vector supercomputer SX-aurora TSUBASA”, SC18, https://dl.acm.org/citation.cfm?id=3291728

Copyright 2019 FUJITSU LIMITED

“Fugaku” CPU Performance Evaluation (3/3)

FUJITSU

- WRF: Weather Research and Forecasting model
  - Vectorizing loops including IF-constructs is key optimization
  - Source code tuning using directives promotes compiler optimizations

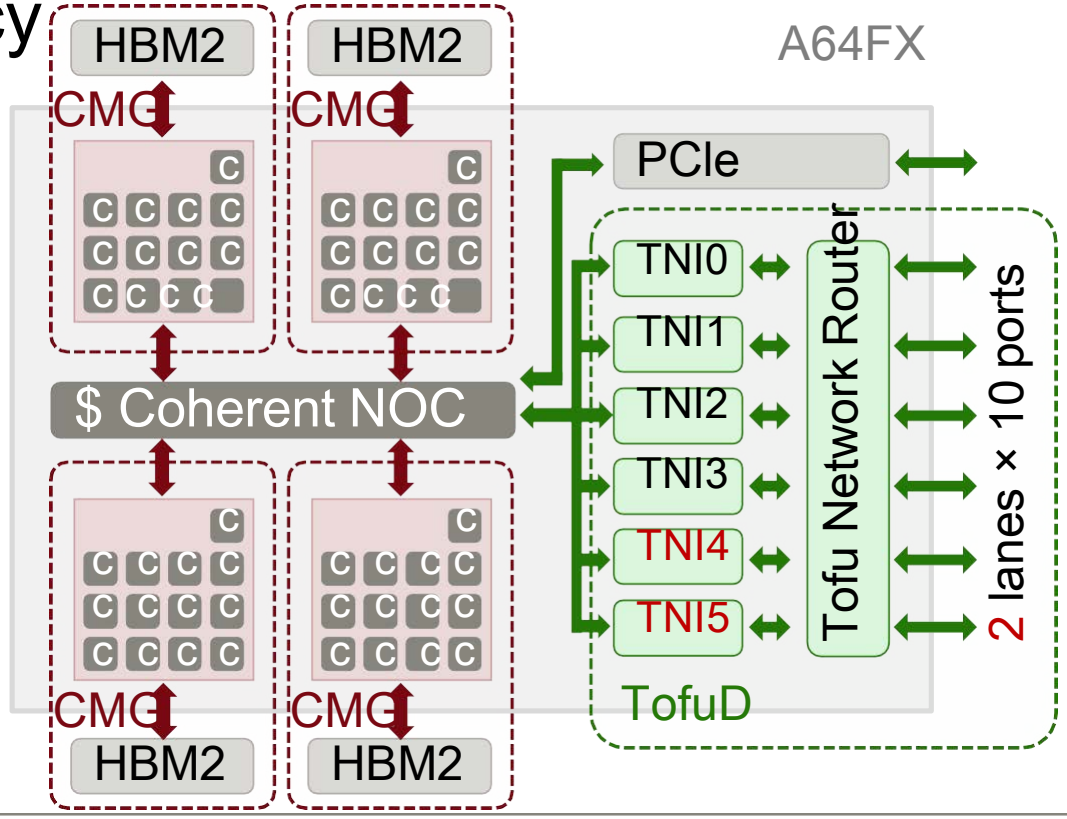


Copyright 2019 FUJITSU LIMITED

A64FX: Tofu interconnect D

- Integrated w/ rich resources
  - Increased TNIs achieves higher injection BW & flexible comm. patterns
  - Increased barrier resources allow flexible collective comm. algorithms
- Memory bypassing achieves low latency
  - Direct descriptor & cache injection

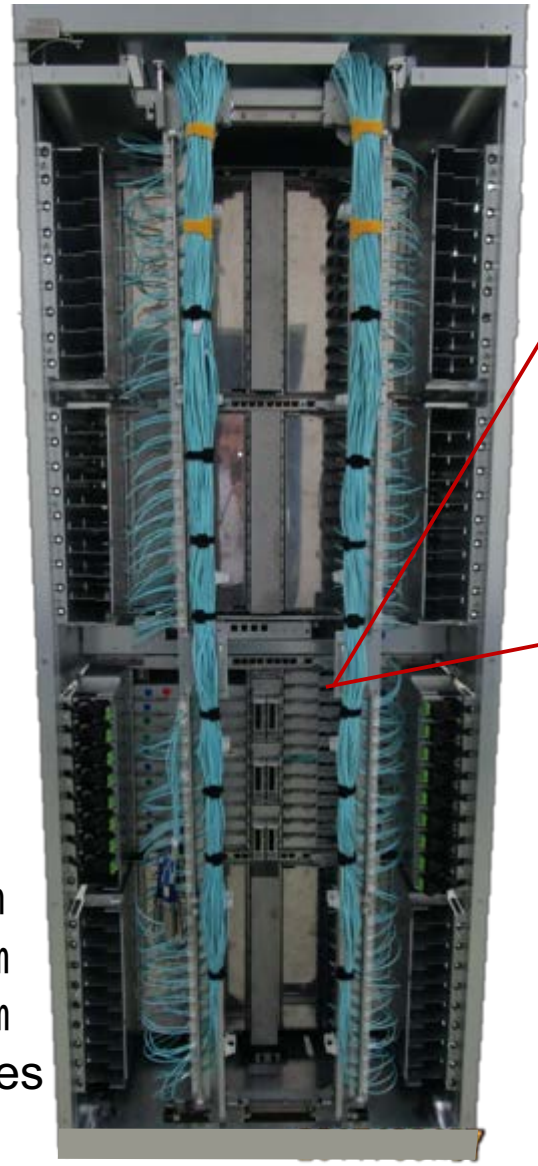
	TofuD spec
Port bandwidth	6.8 GB/s
Injection bandwidth	40.8 GB/s
	Measured
Put throughput	6.35 GB/s
Ping-pong latency	0.49~0.54 μs



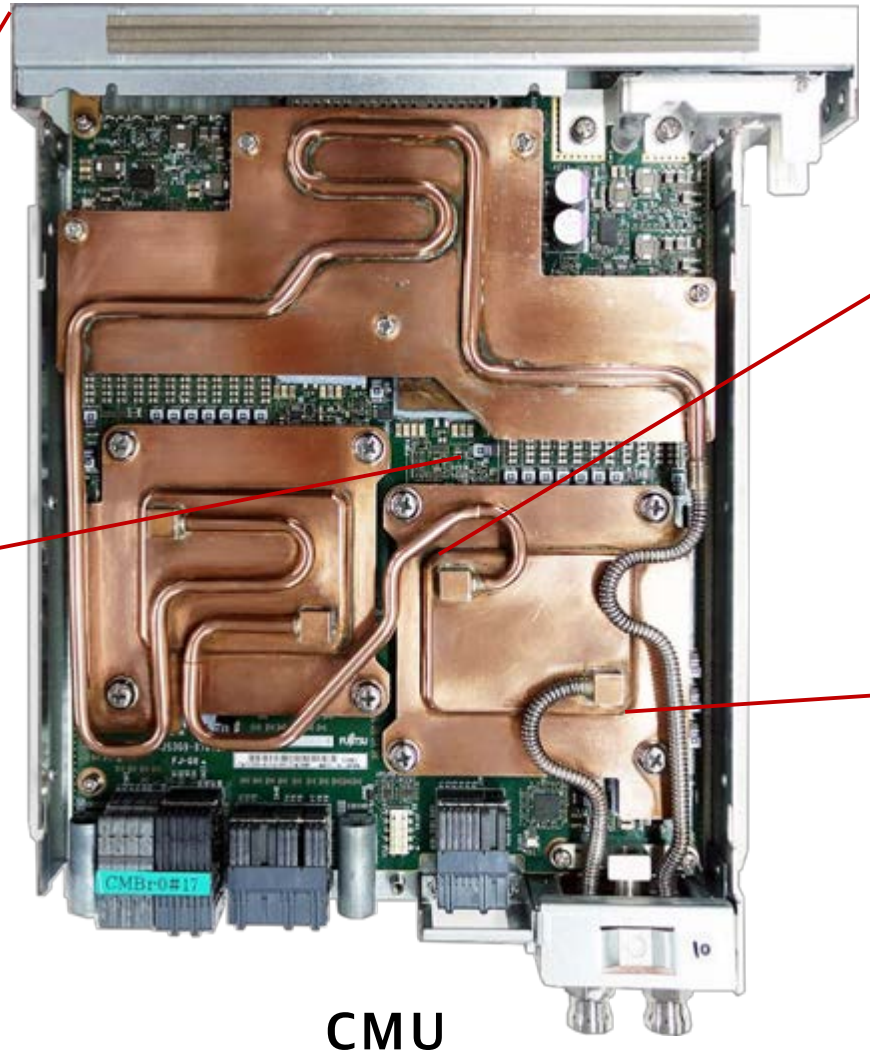
© 2019 FUJITSU

Fugaku Chassis, PCB (w/DLC), and CPU Package

FUJITSU



W 800mm
D1400mm
H2000mm
384 nodes



CMU

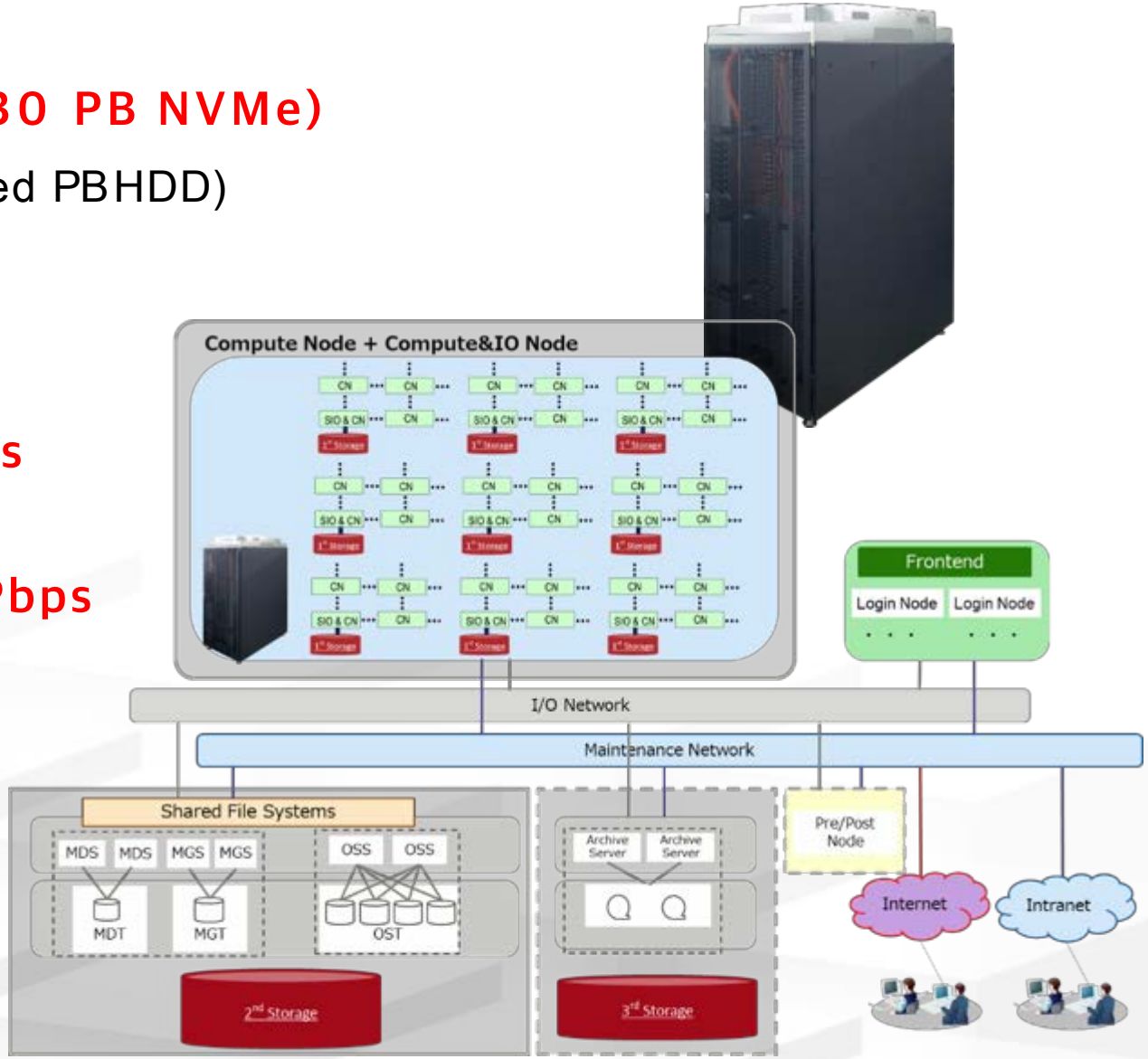
CPU Package

A0 Chip Booted in June
Undergoing Tests

Copyright 2018 FUJITSU LIMITED

Overview of Fugaku System & Storage

- 3-level hierarchical storage
  - 1st Layer: GFS Cache + Temp FS (25~30 PB NVMe)
  - 2nd Layer: Lustre-based GFS (a few hundred PB HDD)
  - 3rd Layer: Off-site Cloud Storage
- Full Machine Spec
  - >150,000 nodes
    ~8 million High Perf. Arm v8.2 Cores
  - > 150PB/s memory BW
  - Tofu-D 10x Global IDC traffic @ 60Pbps
  - > 400 racks
  - ~40 MegaWatts Machine+IDC
    PUE ~ 1.1 High Pressure DLC
  - NRE pays off: ~= 15~30 million
    state-of-the art competing CPU
    Cores for HPC workloads
    (both dense and sparse problems)



Fugaku Performance Estimate on 9 Co-Design Target Apps

Performance target goal

- ✓ 100 times faster than Kfor some applications (tuning included)
- ✓ 30 to 40 MW power consumption

Peak performance to be achieved

	PostK	K
Peak DP (double precision)	>400+ Pflops (34x +)	11.3 Pflops
Peak SP (single precision)	>800+ Pflops (70x +)	11.3 Pflops
Peak HP (half precision)	>1600+ Pflops (141x +)	--
Total memory bandwidth	>150+ PB/sec (29x +)	5,184TB/sec

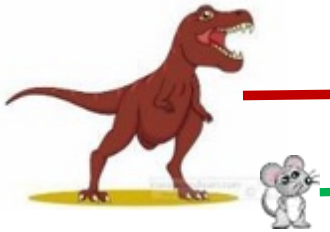
Geometric Mean of Performance Speedup of the 9 Target Applications over the K-Computer

> 37x+

As of 2019/05/14

Category	Priority Issue Area	Performance Speedup over K	Application	Brief description
Health and longevity	1. Innovative computing infrastructure for drug discovery	**125x +**	GENESIS	MD for proteins
	2. Personalized and preventive medicine using big data	8x +	Genomon	Genome processing (Genome alignment)
Disaster prevention and Environment	3. Integrated simulation systems induced by earthquake and tsunami	45x +	GAMERA	Earthquake simulator (FEM in unstructured & structured grid)
	4. Meteorological and global environmental prediction using big data	**120x +**	NICAM+ LETKF	Weather prediction system using Big data (structured grid stencil & ensemble Kalman filter)
Energy issue	5. New technologies for energy creation, conversion / storage, and use	40x +	NTChem	Molecular electronic simulation (structure calculation)
	6. Accelerated development of innovative clean energy systems	35x +	Adventure	Computational Mechanics System for Large Scale Analysis and Design (unstructured grid)
Industrial competitiveness enhancement	7. Creation of new functional devices and high-performance materials	30x +	RSDFT	Ab-initio simulation (density functional theory)
	8. Development of innovative design and production processes	25x +	FFB	Large Eddy Simulation (unstructured grid)
Basic science	9. Elucidation of the fundamental laws and evolution of the universe	25x +	LQCD	Lattice QCD simulation (structured grid Monte Carlo)

Many Core Era



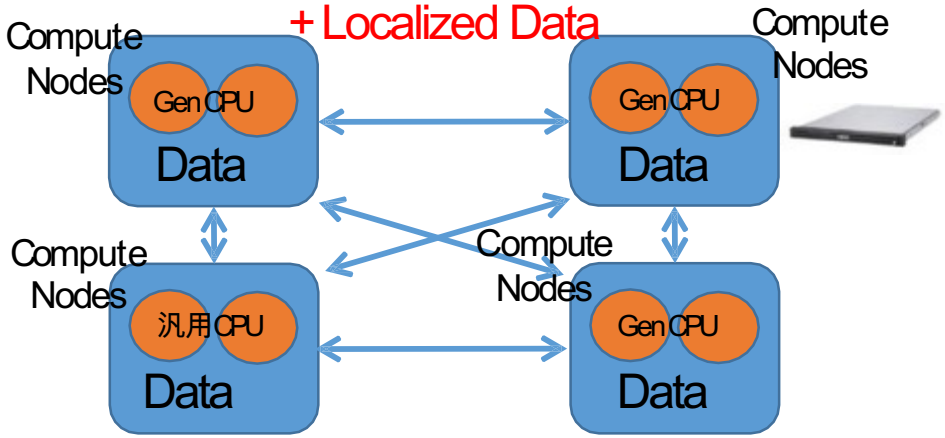
Post Moore
Cambrian Era

Flops-Centric Monolithic Algorithms and Apps

Flops-Centric Monolithic System Software

Hardware/Software System APIs
Flops-Centric Massively Parallel Architecture

Homogeneous General Purpose Nodes
+ Localized Data



Loosely Coupled with Electronic Interconnect

Transistor Lithography Scaling
(CMOS Logic Circuits, DRAM/SRAM)

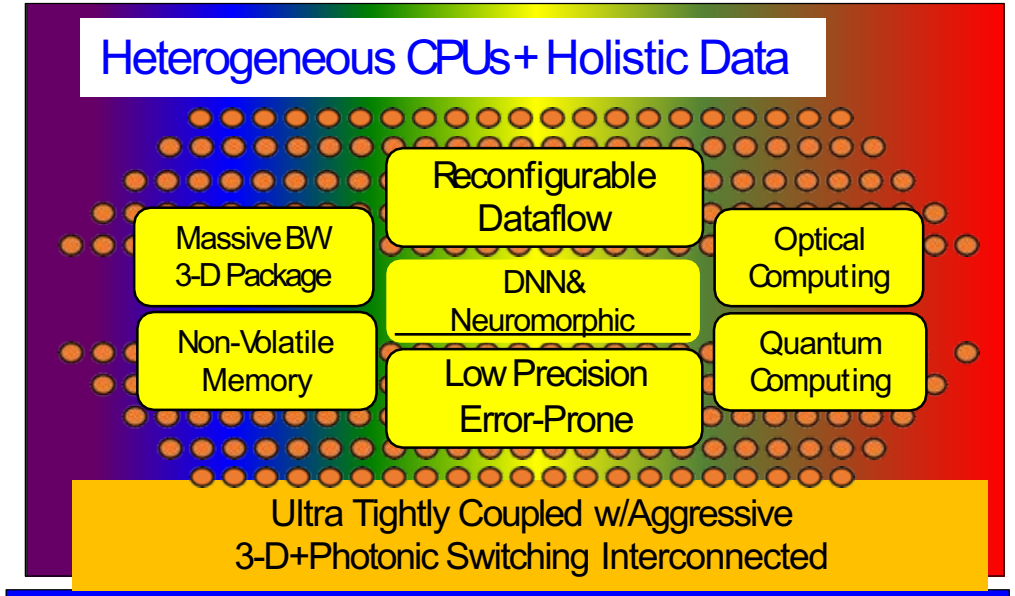
~2025
M-P Extinction
Event

Cambrian Heterogeneous Algorithms and Apps

Cambrian Heterogeneous System Software

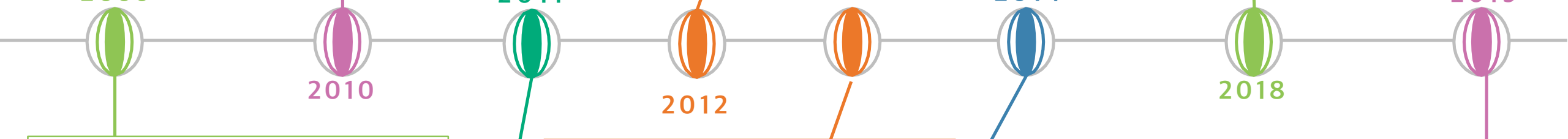
Hardware/Software System APIs
"Cambrian" Heterogeneous Architecture

Heterogeneous CPUs + Holistic Data



Novel Devices + CMOS (Dark Silicon)
(Nanophotonics, Non-Volatile Devices etc.)

Brief History of R-CCS towards Fugaku



2006

January 2006
Next Generation Supercomputer Project (K Computer) start

2010

July 2010
RIKEN AICS established
August 2010
HPCI Project start
September 2010
K computer installation starts
First meeting of SDHPC (Post-K)

2011

June 2011
#1 on Top 500
November 2011
#1 on Top 500 > 10 Petaflops
ACM Gordon Bell Award
End of FY 2011 (March 2012)
SDHPC Whitepaper

2012

April 2012
Post-K Feasibility Study start
3 Arch Teams and 1 Apps Team
June 2012
K computer construction complete
September 2012
K computer production start
November 2012
ACM Gordon bell Award

2013

End of FY2013 (Mar 2014)
Post-K Feasibility Study Reports

2014

April 2014
Post-K project start
June 2014
#1 on Graph 500

2018

April 2018
AICS renamed to RIKEN R-CCS.
Satoshi Matsuoka becomes new Director
Aug 2018
Arm A64fx announce at Hotchips
Oct 2018
NEDO 100x processor project start
Nov 2018
Post-K Manufacturing approval by Prime Minister's CSTI Committee

2019

March 2019
Post-K Manufacturing start
May 2019
Post-K named "Supercomputer Fugaku"
July 2019
Post-Moore Whitepaper start
Aug 2019
K Computer shutdown
Dec 2019
Fugaku installation start (planned)

Retrospect - have we done the right modsim?

- **New AI methodologies and architectures - how do we deal with them?**
- **Post-Moore speedup methodologies**
  - FLOPS no longer free – towards BW-centric?
  - Extreme heterogeneity - neuromorphic, (pseudo-) quantum
  - Severe power constraints and high failure rates
- **Methodological questions**
  - How do we modsim new computing models?
  - Are we picking the right benchmarks for modsim – (not contrived? C.f. Berkely "Motifs")
  - Are we using the right modsim technologies – are we stuck on first principle simuations?
  - How do we modsim inexact systems – perf variations, frequent failures, inexact calculations, etc.

Double-precision FPUs in High-Performance Computing: An Embarrassment of Riches?

33rd IEEE IPDPS, 21. May 2019, Rio de Janeiro, Brazil

Satoshi MATSUOKA Laboratory
Dept. of Math. and Compute Sci.
Tokyo Institute of Technology

Jens Domke, Dr.

Motivation and Initial Question (To float … or not to float …?)

- Thanks to the (curse of) the TOP500 list, the HPC community (and vendors) are chasing higher FP64 performance, thru frequency, SIMD, more FP units, …

Motivation:

- Less FP64 units →
  - **Saves power**
  - **Free chip area** (for e.g.: FP16)
  - **Less divergence** of "HPC-capable" CPUs from mainstream processors

Resulting Research Questions:

- Q1: How much do HPC workloads actually **depend on FP64** instructions?
- Q2: How well do our HPC workloads **utilize the FP64** units?
- Q3: Are our **architectures well- or ill-balanced**: more FP64, or FP32, Integer, memory?

… and …

- Q4: How can we actually **verify our hypothesis**, that we need less FP64 and should invest $ and chip area in more/faster FP32 units and/or memory)?

Approach and Assumptions

Idea/Methodology

- Compare two similar chips; different balance in FPUs ➔ Which?
- Use 'real' applications running on current/next-gen. machines ➔ Which?

Assumptions

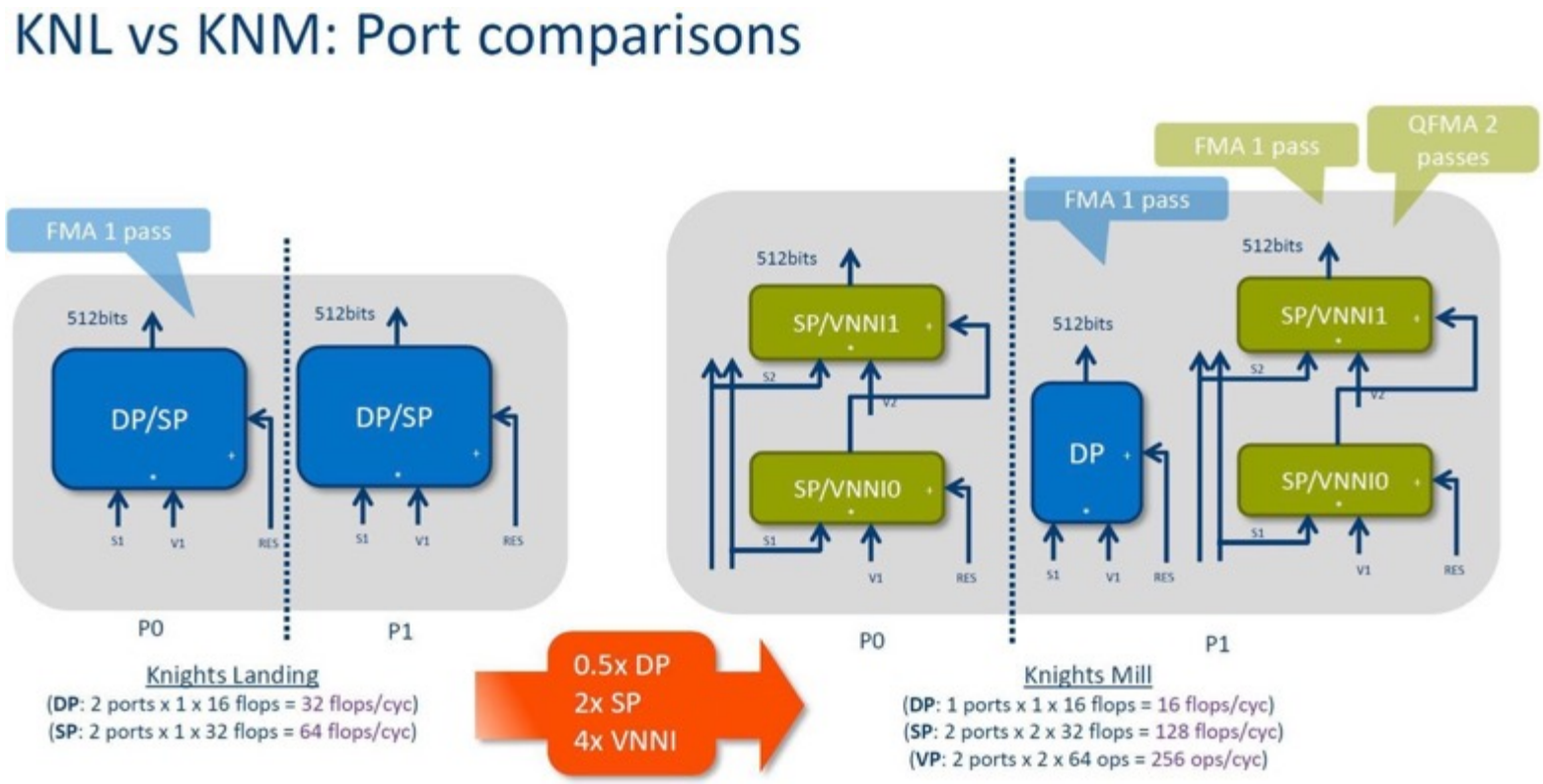
- Our HPC (mini-)apps are well-optimized
  - Appropriate compiler settings
  - Used in procurement of next gen. machines (e.g. Summit, Post-K, …)
  - Mini-apps: Legit representative of the priority applications [1]
- We can find two chips which are similar
  - No major differences (besides FP64 units)
  - Aside from minor differences we know of (…more on next slide)
- The measurement tools/methods are reliable
  - Make sanity checks (e.g.: use HPL and HPCG as reference)

[1] Aaziz et at, "A Methodology for Characterizing the Correspondence Between Real and Proxy Applications", in IEEE Cluster 2018

Methodology – CPU Architectures

- Two very **similar CPUs with large difference in FP64** units
- Intel dropped 1 DP unit for 2x SP and 4x VNNI (similar to Nvidia's TensorCore)
- Vector Neural Network Instruction (VNNI) supports SP floating point and mixed precision integers (16-bit input/32-bit output) ops

➔ **KNM: 2.6x higher SP** peak performance and **35% lower DP** peak perf.



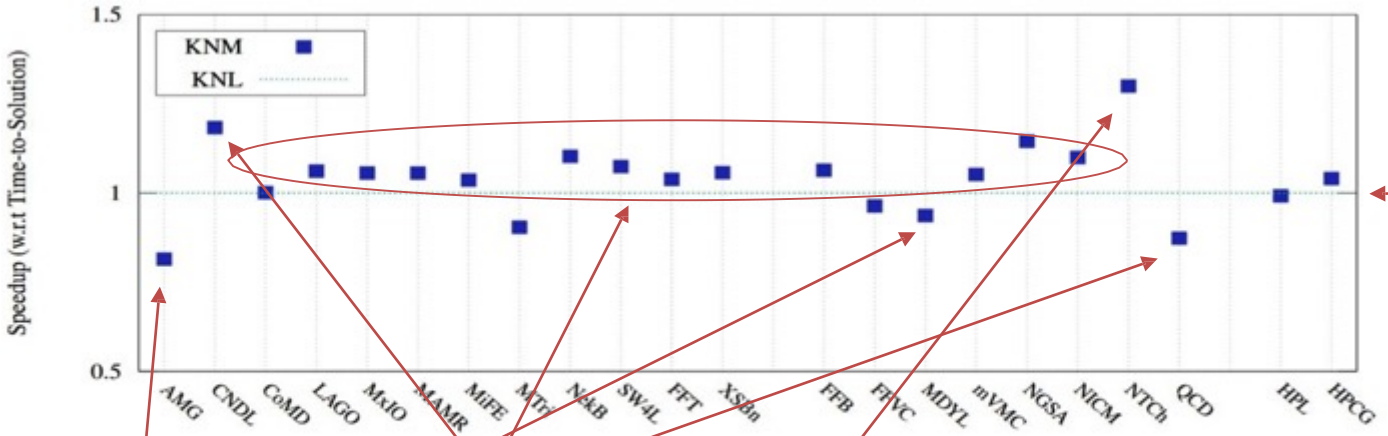
(Figure source: https://www.servethehome.com/intel-knights-mill-for-machine-learning/)

Methodology – Benchmarks and Execution Environment

- 23 mini-apps used in procurement process of next-gen machines

ECP	Workload	Post-K	Workload
AMG	Algebraic **multigrid solver** for unstructured grids	**CCS QCD**	Linear equation solver (sparse matrix) for **lattice chromodynamics** (QCD) problem
CANDLE	**DL predict drug response** based on molecular of tumor cells	**FFVC**	Solves the 3D unsteady **thermal flow** of the incompressible fluid
CoMD	Generate atomic transition pathways between any two structures of a **protein**	**NICAM**	Benchmark of **atmospheric** general circulation **model** reproducing the unsteady baroclinic oscillation
Laghos	Solves the Euler equation of compressible **gas**	**mVMC**	Variational **Monte Carlo** method applicable for a wide range of Hamiltonians for interacting **fermion systems**
MACSio	Scalable **I/O** Proxy Application	**NGSA**	Parses data generated by a next-generation genome sequencer and **identifies genetic differences**
miniAMR	Proxy app for structured adaptive mesh refinement (**3D stencil**) kernels used by many scientific codes	**MODYLAS**	**Molecular dynamics** framework adopting the fast multipole method (**FMM**) for electrostatic interactions
miniFE	Proxy for unstructured implicit **finite element** or finite volume applications	**NTChem**	Kernel for molecular electronic structure calculation of standard **quantum chemistry** approaches
miniTRI	Proxy for dense subgraph detection, characterizing graphs, and improving **community detection**	**FFB**	Unsteady incompressible **Navier-Stokes** solver by element method for thermal flow simulations
Nekbone	High order, incompressible **Navier-Stokes** solver spectral element method	**Bench**	**Workload**
SW4lite	Kernels for **3D seismic modeling** in 4th order	**HPL**	Solves **dense system** of linear equations Ax = b
SWFFT	Fast Fourier transforms (**FFT**) used in by Hardware Accelerated **Cosmology** Code (HACC)	**HPCG**	Conjugate gradient method on **sparse matrix**
XSBench	Kernel of the **Monte Carlo neutronics** app: OpenMC	**Stream**	Throughput measurements of **memory** subsystem

Results – Compare Time-to-Solution in Solver



KNL baseline

Fig. 4. Speedup of KNM over KNL as baseline. MiniAMR included since the input is the same for both Phi; Proxy-app abbreviations acc. to Section II-B

- Only 3 apps seem to suffer from missing DP (MiniTri: no FP; FFVC: only int+FP32)
- VNNI may help with CANDLE perf. on KNM; NTChem improvement unclear
- KNL overall better (due to 100MHz freq. incr.?)
- Memory throughput on Phi (in *cache mode*) doesn't reach peak of *flat mode* (only ~86% on KNL; ~75% on KNL)

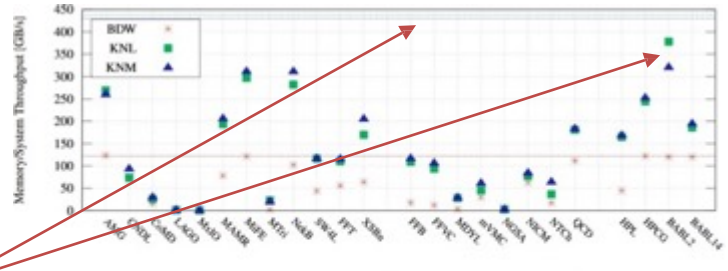
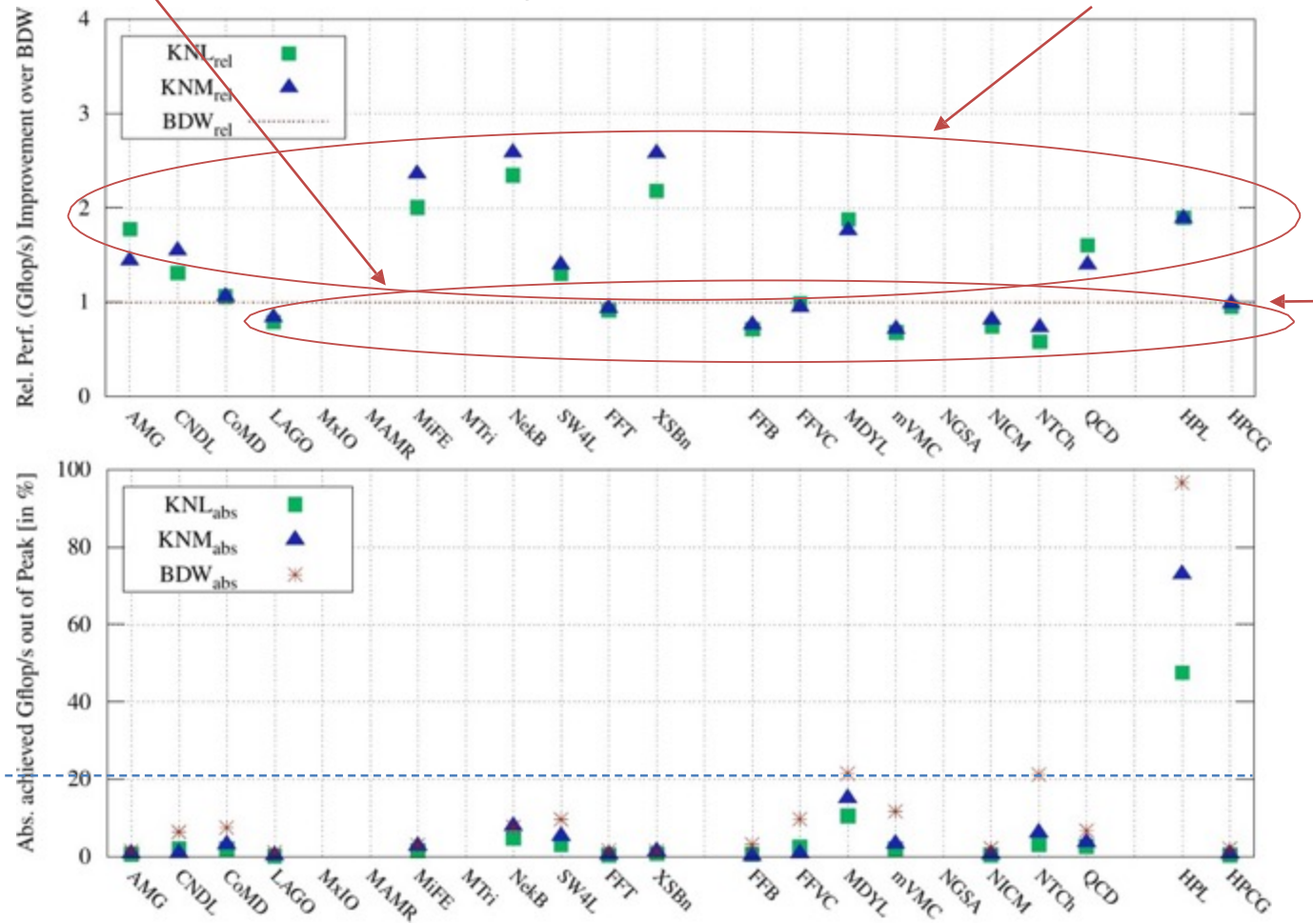


Fig. 5. Memory throughput (only DRAM for BDW, DRAM+MCDRAM for Phi) per proxy-app; Dotted lines indicate Triad stream bandwidth (flat mode, cf. Tab. I); BabelStream for 2 GiB (BABL2) and 14 GiB (BABL14) vector

Note: MiniAMR not strong-scaling ➔ limited comparability

Results – Compare Gflop/s in Comp. Kernel/Solver

- 8 apps out of 18: less Gflop/s on Phi than on BDW (ignoring I/O & Int-based apps)
- All apps (ignoring HPL) with **low FP efficiency**:
  ≤ 21.5% on BDW, ≤ 10.5% on KNL, ≤ 15.1% on KNM (Why? ➔ next slides)
- Phi performance comes from higher peak flop/s, Iop/s and/or faster MCDRAM?



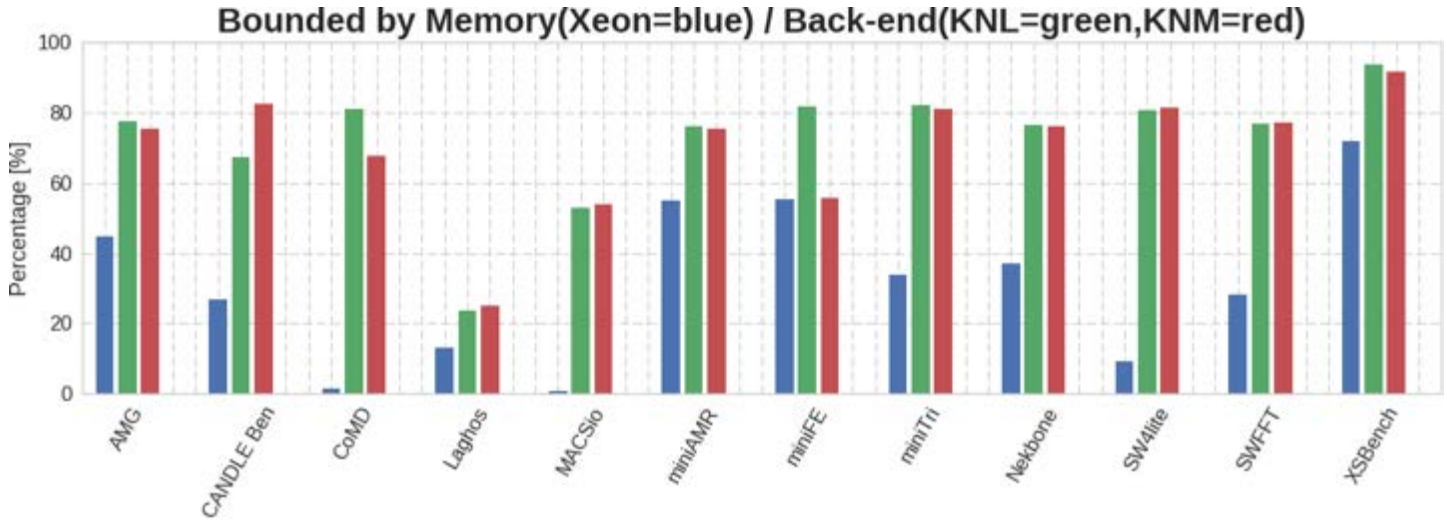
Relative perf. over BDW baseline

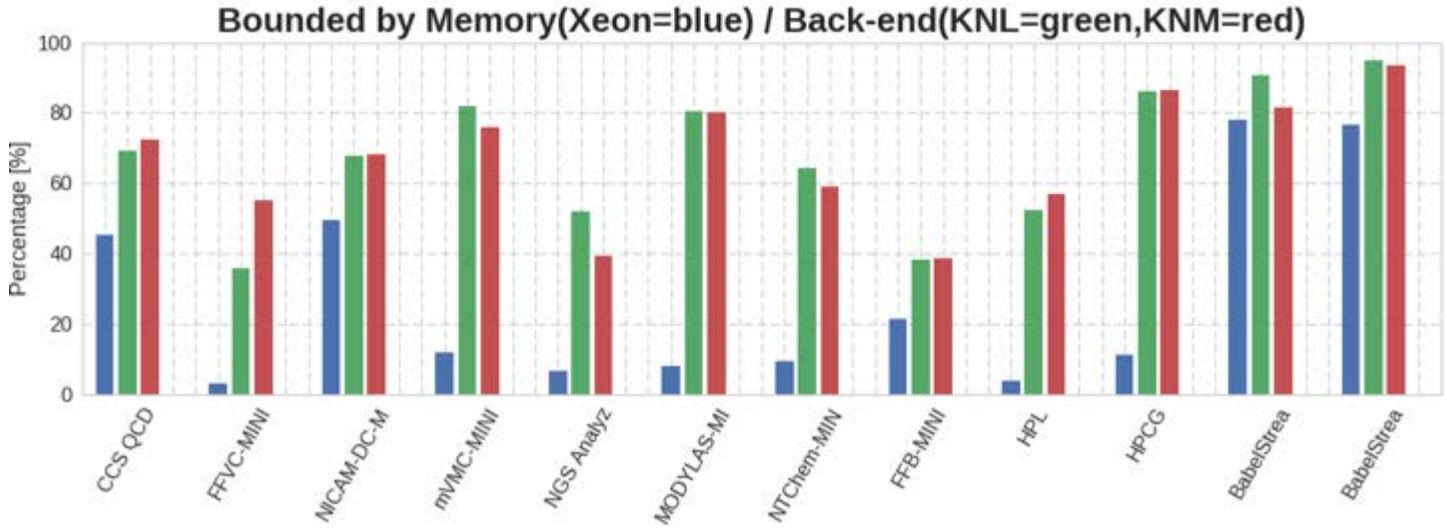
Absolute Gflop/s perf. compared to theor. peak

20% of theor. peak

Results – Memory-/Backend-bound (VTune)

- Surprisingly high (~80% for Phi) ➔ “unclear” how VTune calculates these %
  (Memory-bound != backend-bound ➔ no direct comparison BDW vs Phi)





Results – Roofline Analysis for Verification

- **Supports** our previous **hypothesis** that most of the proxy-/mini-apps are **memory-bound**
- Outlier: only Laghos seems (intentionally?) poorly optimized
- Verifies our assumption about **optimization status of the apps** (➔ similar to other HPC roofline plots)
- KNL/KNM roofline plots show nearly same results (omitted to avoid visual clutter)

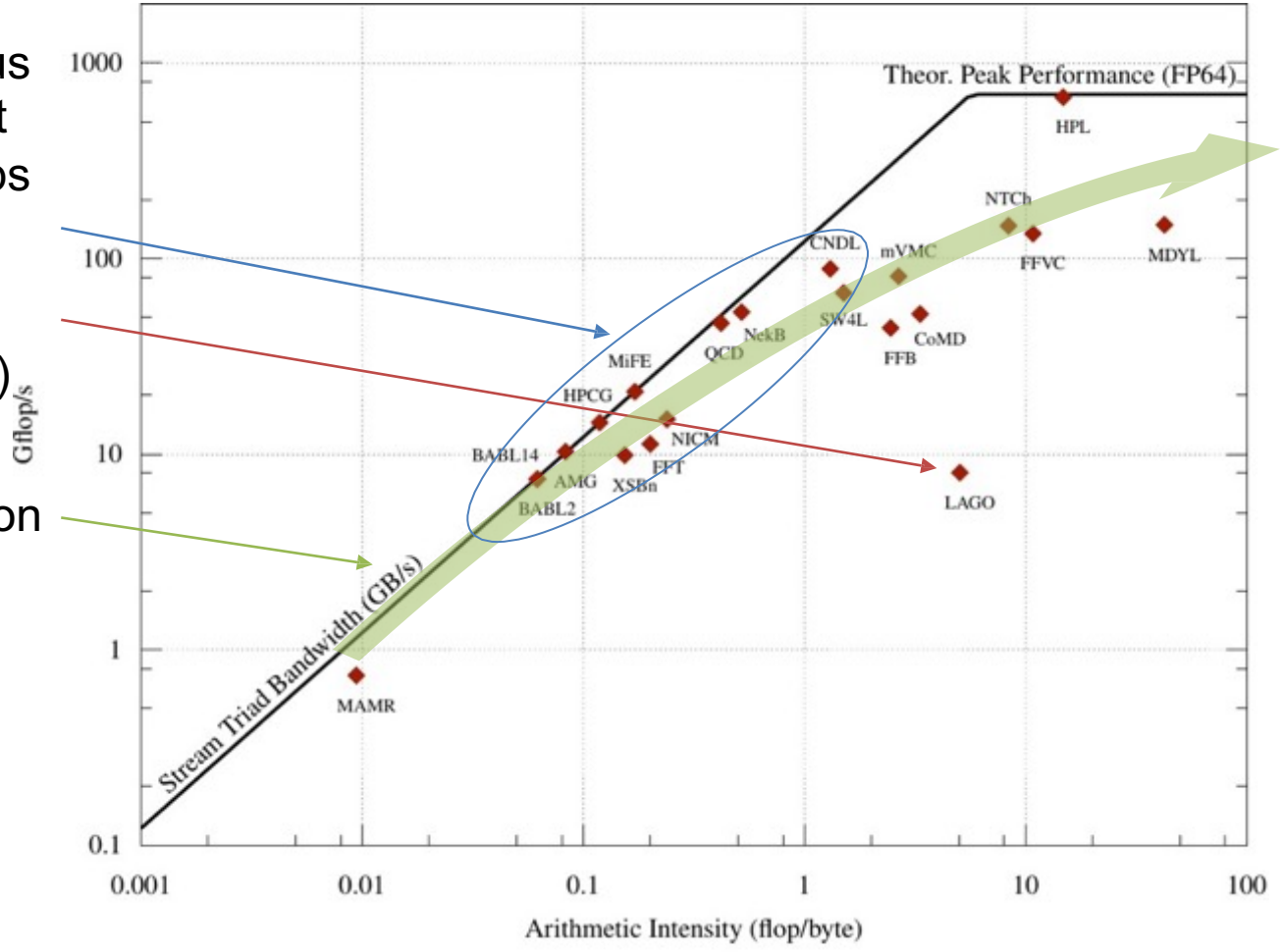


Fig. 5. Roofline plot (w.r.t dominant FP operations and DRAM bandwidth) for Broadwell-EP reference system; Filtered proxy-apps with negligible FP operations: MxIO, MTri, and NGSA; Proxy-app labels acc. to Section II-B

Results – Requirement for a "Weighted Look" at Results

- Studied HPC utilization reports of 8 centers across 5 countries
- **Not every app equally important** (most HPC cycles dominated by Eng. (Mech./CFD), Physics, Material Sci., QCD)

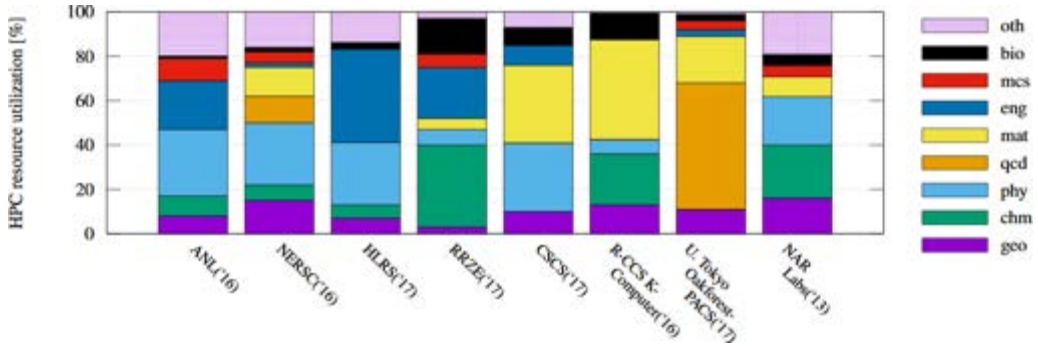


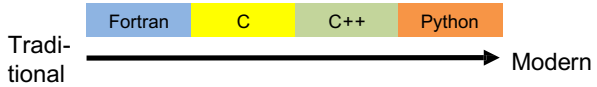
TABLE II

APPLICATION CATEGORIZATION, COMPUTE PATTERNS, AND MAIN PROGRAMMING LANGUAGES USED; MACSIO, HPL, HPCG, AND BABELSTREAM BENCHMARKS OMITTED

ECP	Scientific/Engineering Domain	Compute Pattern	Language
AMG	Physics and Bioscience	Stencil	C
CANDLE	Bioscience	Dense matrix	Python
CoMD	Material Science/Engineering	N-body	C
Laghos	Physics	Irregular	C++
miniAMR	Geoscience/Earthscience	Stencil	C
miniFE	Physics	Irregular	C++
miniTRI	Math/Computer Science	Irregular	C++
Nekbone	Math/Computer Science	Sparse matrix	Fortan
SW4lite	Geoscience/Earthscience	Stencil	C
SWFFT	Physics	FFT	C/Fortra
XSBench	Physics	Irregular	C

RIKEN	Scientific/Engineering Domain	Compute Pattern	Language
FFB	Engineering (Mechanics, CFD)	Stencil	Fortran
FFVC	Engineering (Mechanics, CFD)	Stencil	C++/For
mVMC	Physics	Dense matrix	C
NICAM	Geoscience/Earthscience	Stencil	Fortran
NGSA	Bioscience	Irregular	C
MODYLAS	Physics and Chemistry	N-body	Fortran
NTChem	Chemistry	Dense matrix	Fortran
QCD	Lattice QCD	Stencil	Fortran/C

- Some supercomputers are "specialized"
  - **Dedicated HPC** (e.g.: weather forecast)
- For system *X* running **memory-bound** apps
  - **Why pay premium for FLOPS?**
  - NASA applies this pragmatic approach [2]



[2] S. Saini et al., "Performance Evaluation of an Intel Haswell and Ivy Bridge-Based Supercomputer Using Scientific and Engineering Applications," in HPCC/SmartCity/DSS, 2016

What is meant by Convergence of HPC & AI?

- **Acceleration of Simulation (first principles methods) with AI (empirical method)** ***AI for HPC Systems***
  - Interpolation & Extrapolation of long trajectory MD
  - Reducing parameter space on Paretho optimization of results
  - Adjusting convergence parameters for iterative methods etc.
- **AI *replacing* simulation**
  - When exact physical models are unclear, or excessively costly to compute
- **Acceleration of AI with HPC** ***HPC for AI (Summit, Fugaku etc.)***
  - HPC Processing of training data -data cleansing
  - Acceleration of (Parallel) Training: Deeper networks, bigger training sets, complicated networks, high dimensional data…
  - Acceleration of Inference: above + real time streaming data
  - Various modern training algorithms: Reinforcement learning, GAN, Dilated Convolution, etc.

Convergence of HPC & AI in Modsim

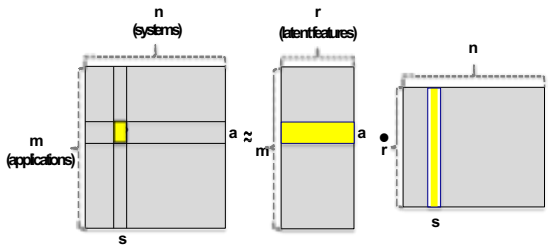
- **Performance modeling and prediction with AI (empirical method)** ***AI for modsim of HPC systems***
  - *C.f. GEM5 simulation – first principle perf. modeling*
  - AI Interpolation & Extrapolation of system performance
  - Objective categorization of benchmarks
  - Optimizing system performance using machine learning

- **Performance Modeling of AI esp. Machine Learning** ***HPC modsim techniques for AI***
  - Perf. modeling of Deep Neural Networks on HPC machines
  - Large scaling of Deep Learning on large scale machines
  - Optimization of AI algorithms using perf modeling
  - Architectural survey and modeling of future AI systems

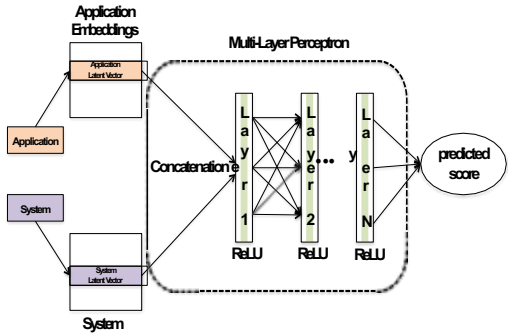
Using AI Techniques for Modsim of HPC

Learning Neural Representations for Predicting GPU Performance

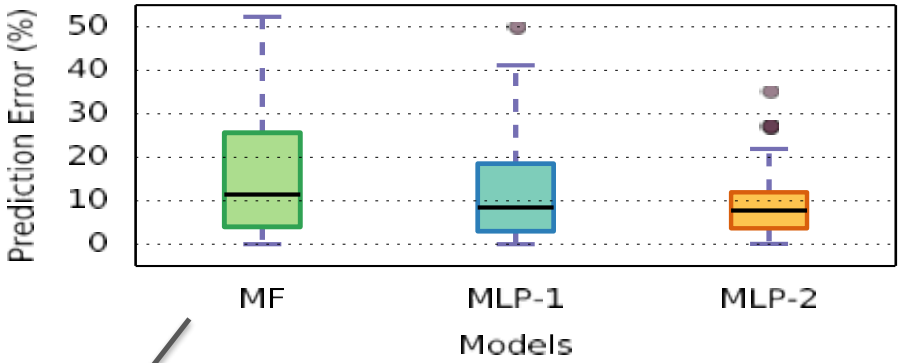
- Motivation
  - New specialized chips are being introduced e.g. Fujistu's A64FX
  - A wide range of choices to run scientific workloads
- Problem
  - Modeling performance across systems with different GPU microarchitectures
- Proposal
  - A collaborative filtering based matrix factorization (MF) approach to automatically learn latent features describing performance of applications on systems
  - A multi-layer perceptron (MLP) to model complex non-linear interactions between applications and systems
- Evaluation
  - 30 workloads from 9 different domains
  - 7 GPUs ranging from Nvidia's Kepler to Volta microarchitecture
  - Metric to predict: IPS



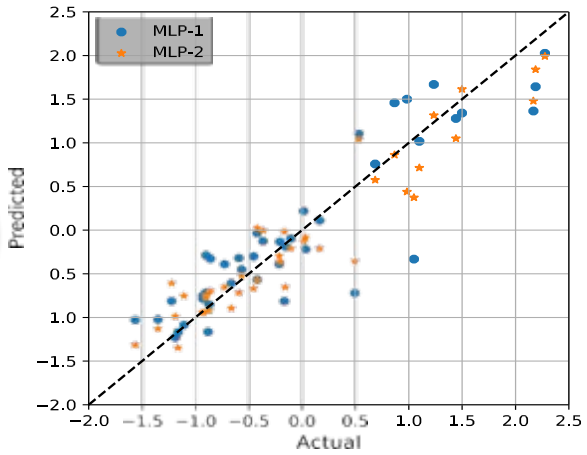
Mapping of applications and systems into a shared latent spacing using MF



Multi-layer perceptron model using latent features



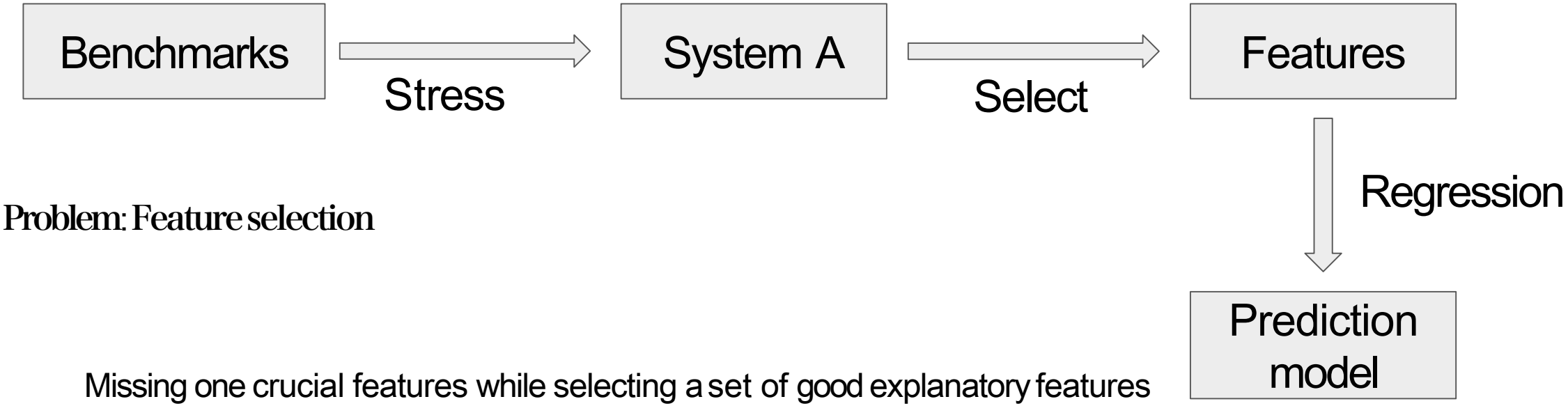
90.6% prediction accuracy achieved using MLP-2



Shweta Salaria, Aleksandr Drozd, Artur Podobas, Satoshi Matsuoka. Learning Neural Representations for Predicting GPU Performance. ISC High Performance 2019 (ISC), Frankurt, Germany, June 2019

Problem Statement

- Cherry-picking a set of features may not always be good enough



Problem: Feature selection

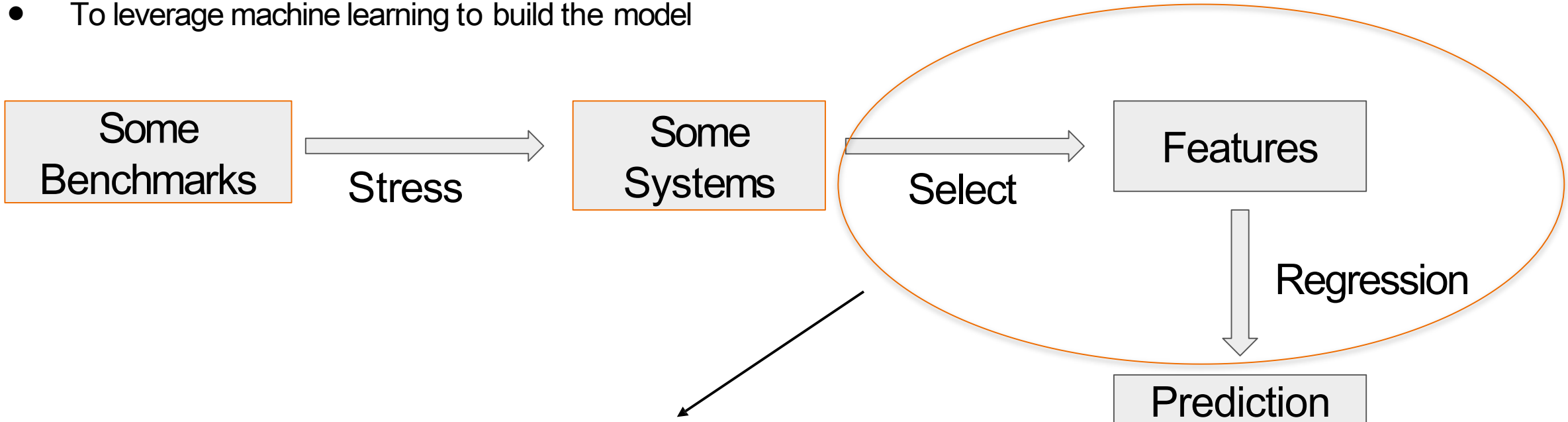
Missing one crucial features while selecting a set of good explanatory features

→ Difficult to repeat feature selection process for each new application and system

→

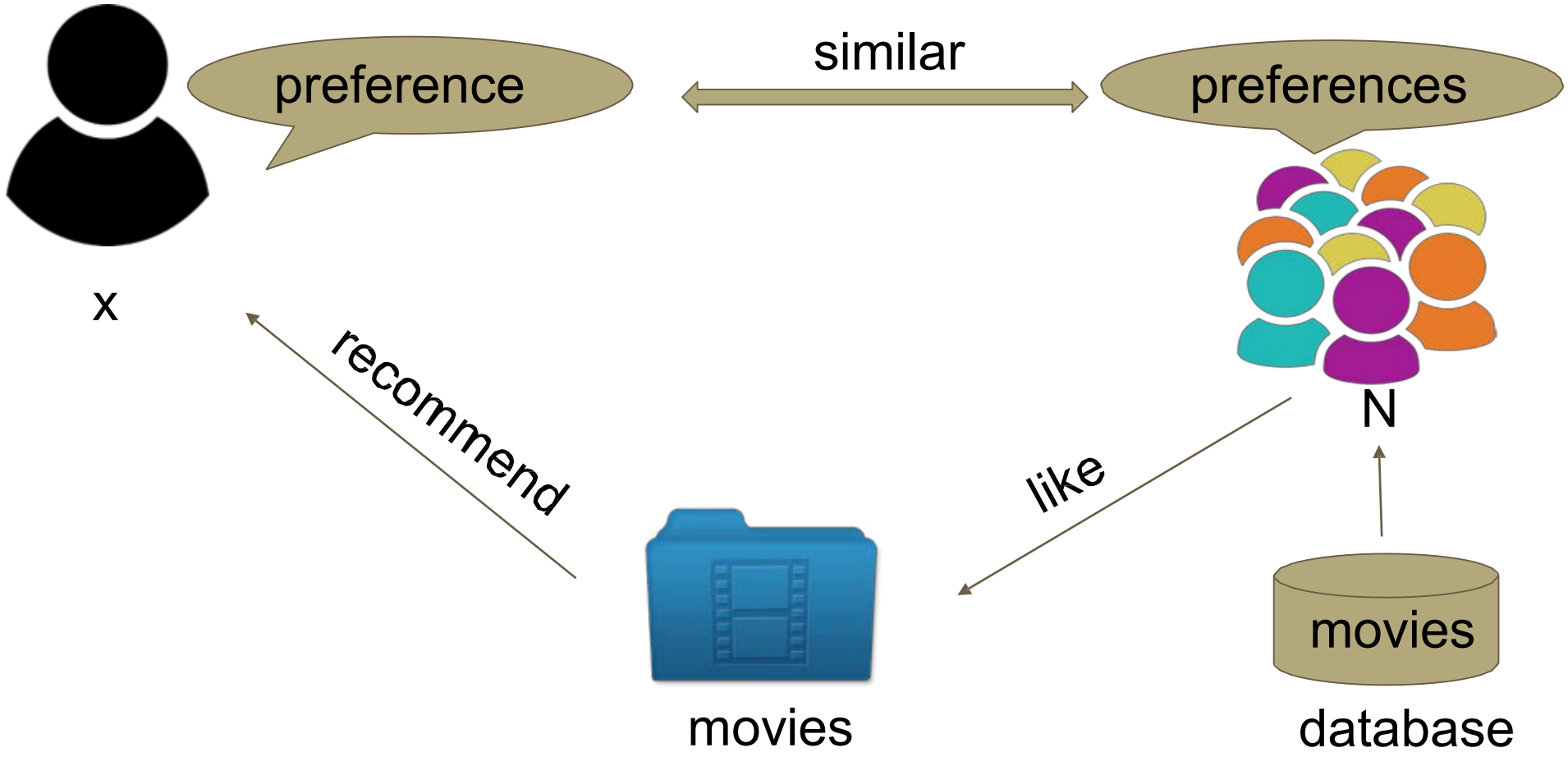
Insight

- To leverage machine learning to build the model



Collaborative filtering (CF) based algorithms handle this by identifying inter-dependencies linking benchmarks with systems

Collaborative Filtering (CF): Automatic Feature Learning



Use Case: Movie Recommendation System

Problem Formulation

- We construct an M applications x N systems matrix such as:

	System 1	System 2	..	System N
App 1				
App 2				
..				
App M				

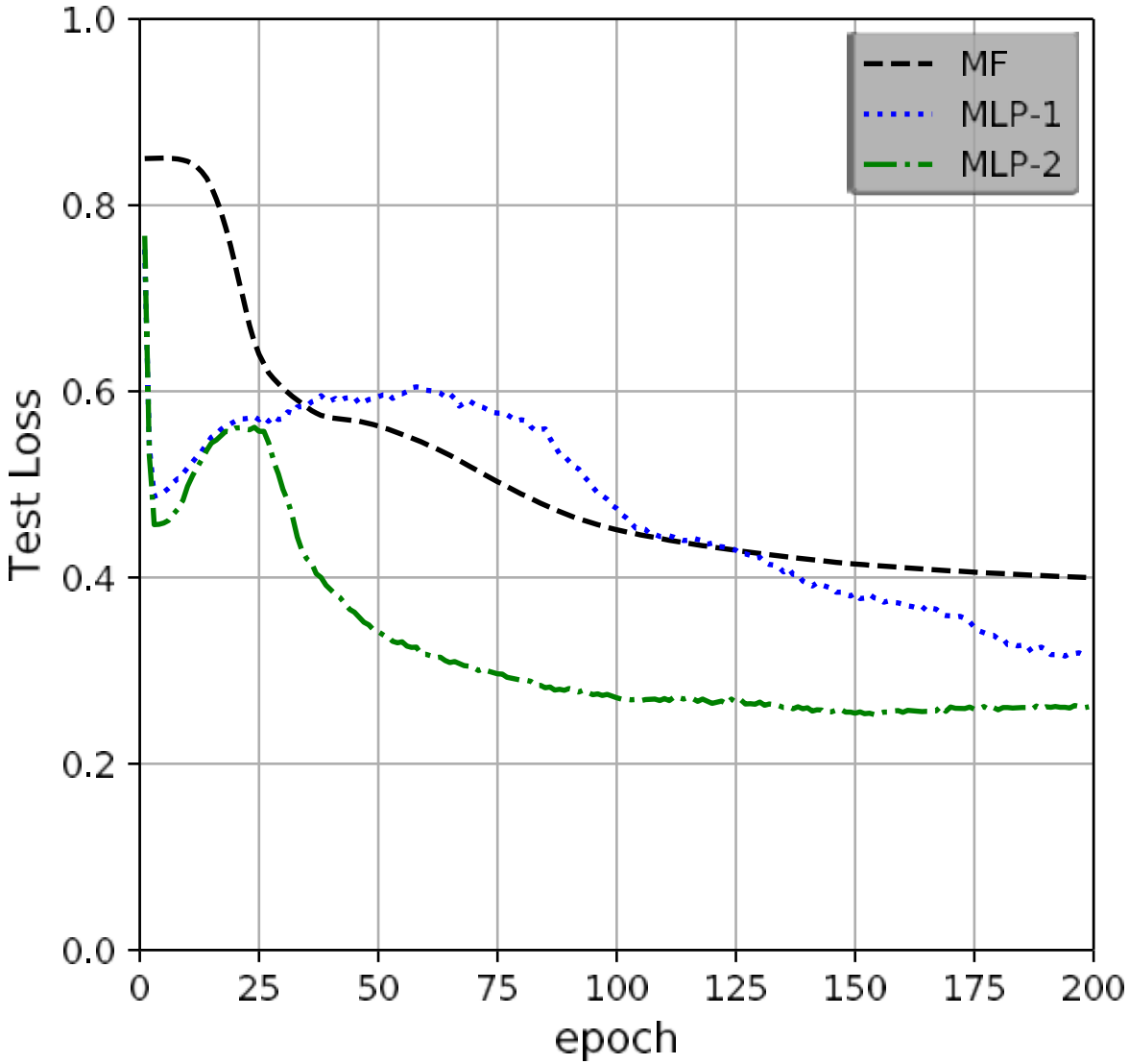
Known performance score

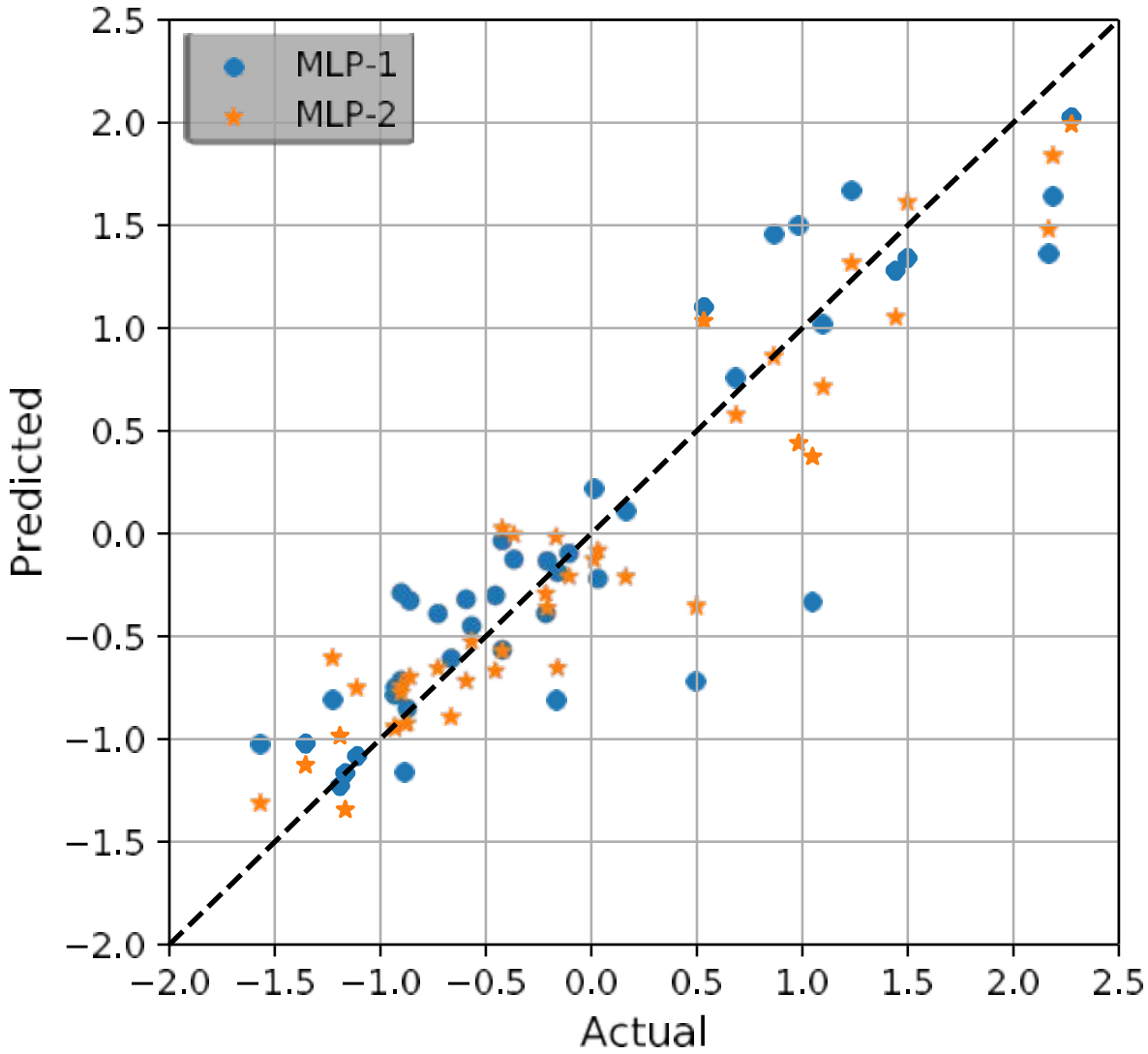
Unknown scores

- Known performance scores are normalized Instructions Per Second (IPS) values
- Our goal is to predict all the zero entries of the matrix

Experimental Setup

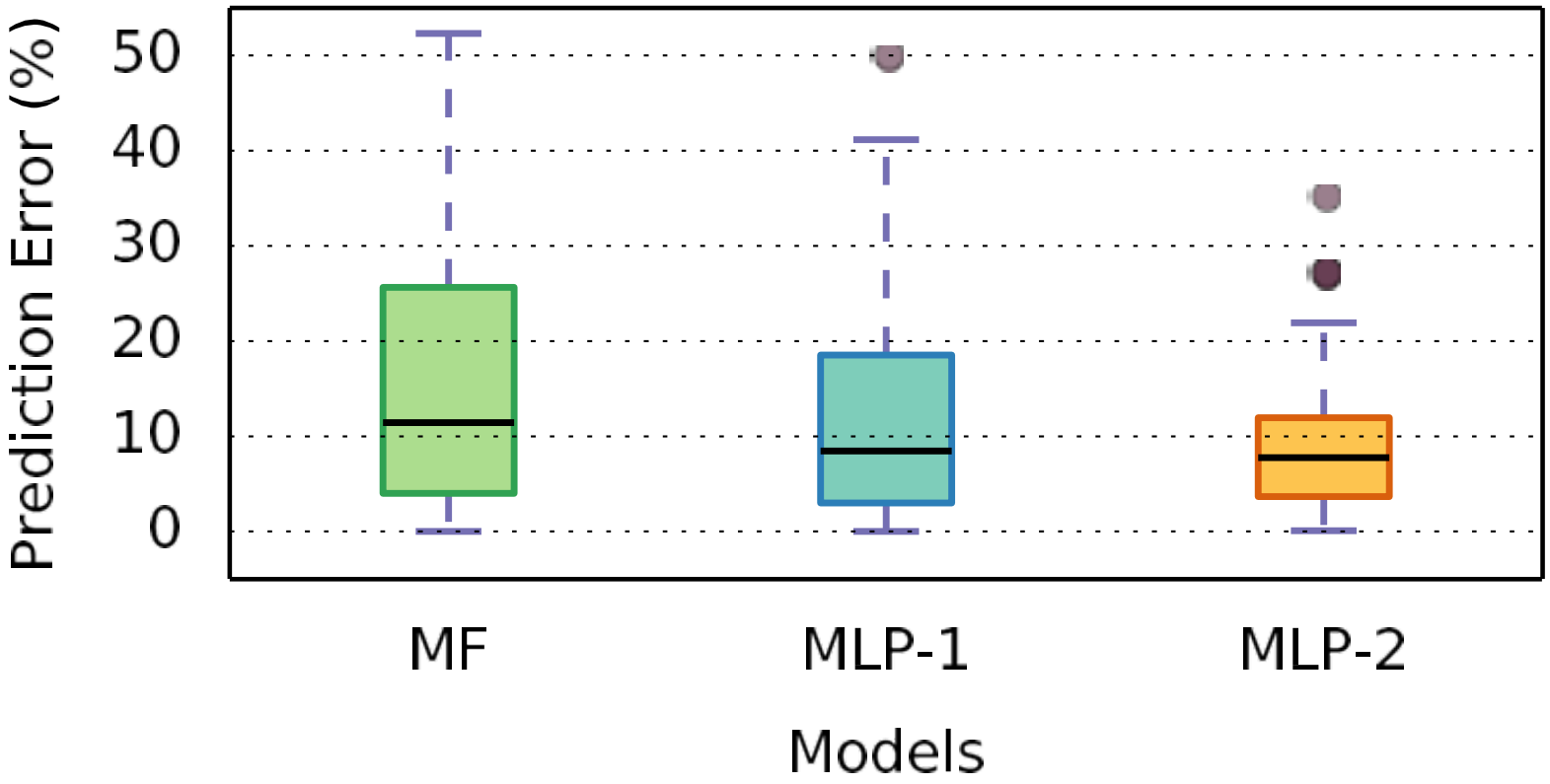
- 30 workloads from Rodinia benchmark suite and Polybench GPU
- Workloads from 9 different domains
  - Linear Algebra (11 workloads)
  - Data Mining and Pattern Recognition (4)
  - Stencils (3)
  - Signal Processing (1)
  - Image Processing (3)
  - Simulation (3)
  - Graph Traversal (3)
  - Fluid and Molecular Dynamics (1)
  - Bioinformatics (1)


MF
MLP-1
MLP-2
Test Loss
epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200


MLP-1
MLP-2
Predicted
Actual
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

Results: Multi-Layer Perceptron

Performance: MLP-2 > MLP-1 > MF



Model	Avg. Error	Geometric Mean
MF	15.8%	7.4%
MLP-1	11.9%	6.3%
MLP-2	**9.4%**	**6.0%**

Accuracy of MF, MLP-1 and MLP-2 using IPS dataset

Classification of benchmarks by machine learning using memory access trace

Toshiki Tsuchikawa, Toshio Endo, Yosuke Oyama, Akihiro Nomura, Masaaki Kondo, Satoshi Matsuoka

Motivation: Select benchmarks which help to design supercomputers

- **Existing benchmark set**
  - Most of them are collections of benchmarks used in each field, such as drug discovery and fluid dynamics.
  - Benchmarks with the same properties may be in the same benchmark set.
- **Seven Dwarfs**
  - Benchmarks classification based on HPC's typical algorithm
  - This is not a scientific classification method because the type of classification is determined in a top-down design.

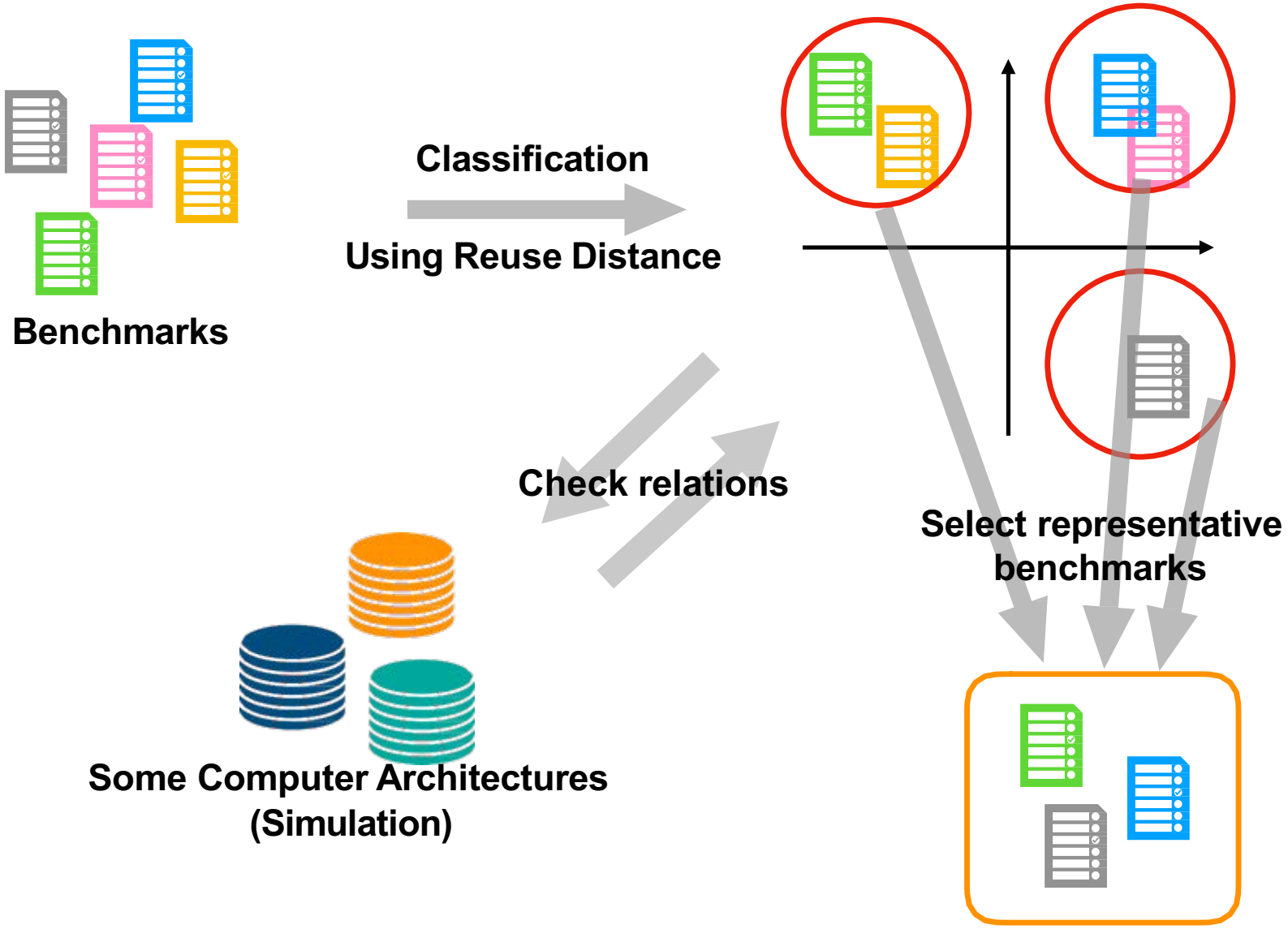
Select benchmarks with the bottom up design by evaluating benchmark performance.

By using

- Feature of memory access(Reuse Distance)
- Machine Learning(Classification method)

Classification of benchmarks by machine learning using memory access trace

Toshiki Tsuchikawa, Toshio Endo, Yosuke Oyama, Akihiro Nomura, Masaaki Kondo, Satoshi Matsuoka



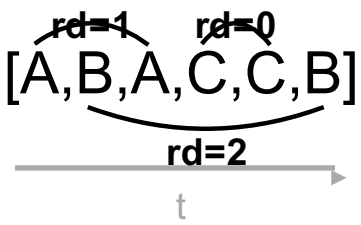
Classification of benchmarks by machine learning using memory access trace

Toshiki Tsuchikawa, Toshio Endo, Yosuke Oyama, Akihiro Nomura, Masaaki Kondo, Satoshi Matsuoka

Background: Reuse Distance

- The number of distinctive addresses accessed between two consecutive uses of the same address
- Reuse Distance is an application-specific feature that does not depend on cache or memory structure

But It takes a lot of memory and complexity



rd=Reuse Distance

Efficient calculation of Reuse Distance

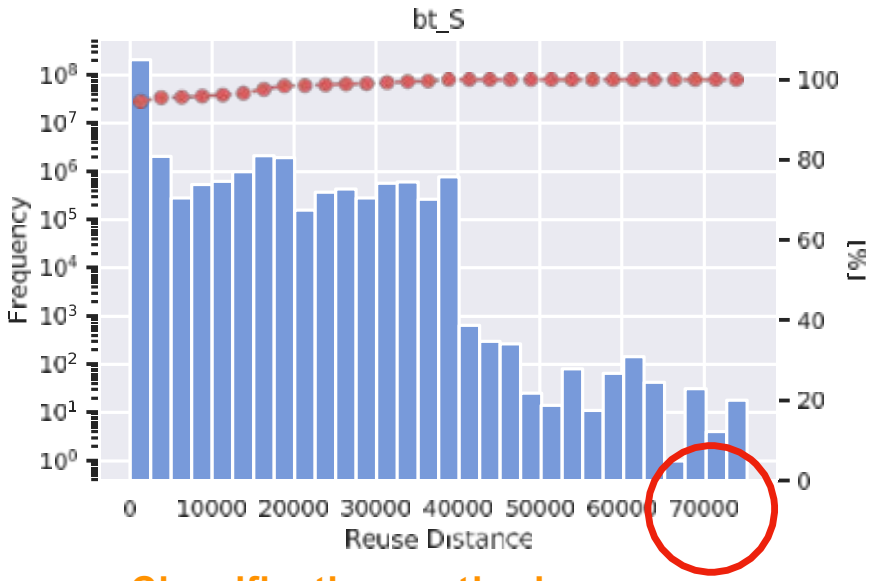
- Reduce computational complexity using three existing research methods
- Reduce memory usage using the SSD of computational node and put address information and calculate reuse distance
- Reduce memory usage saving Reuse Distance in the form of a histogram

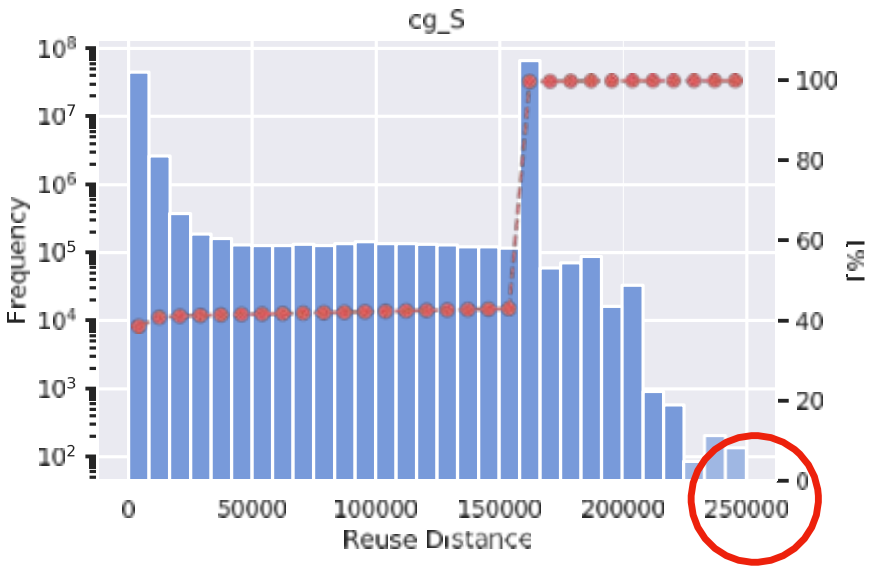
Achieved calculation efficiency more than 1000 times

Classification of benchmarks by machine learning using memory access trace

Toshiki Tsuchikawa, Toshio Endo, Yosuke Oyama, Akihiro Nomura, Masaaki Kondo, Satoshi Matsuoka

- **Methodology: The length of traces of Reuse Distance is disjointed for each benchmark**
  - Divide Reuse Distance histograms at equal intervals to make the number of traces uniform
  - The vector of each benchmarks is the logarithm of each frequency





Classification method

- Use Unsupervised learning
  - K-Means and VBGMM(Variational Bayesian Gaussian Mixture)
- Use 44 benchmarks
  - NAS, Bots, Rodinia and so on
  - Treat different input sizes as different benchmarks

Classification of benchmarks by machine learning using memory access trace

Toshiki Tsuchikawa, Toshio Endo, Yosuke Oyama, Akihiro Nomura, Masaaki Kondo, Satoshi Matsuoka

- **Evaluation1: Classify by K-Means(8Classes)**
  - This figure shows same clusters in columns
  - The kind of shape and color of points mean different clusters
  - Use PCA method to map 100 dimension vector to 2 dimension

Figure: classifications result of K-Means

- **Evaluation2: Investigate relations between clustering results and some architecture specifications**
  - Experimented with three architectures(BDW, KNL, ABCI)
  - By change the preprocessing method we want to find the relations

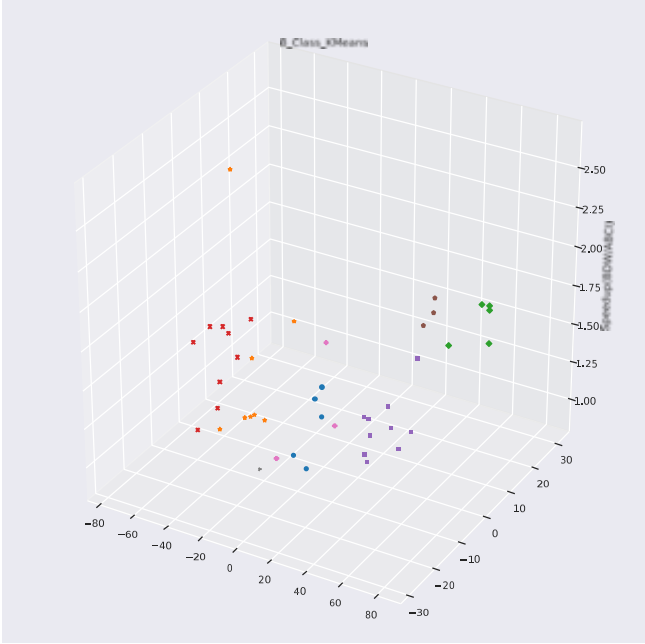


Figure: relations between classification result and speedup

Modsim of AI-HPC systems

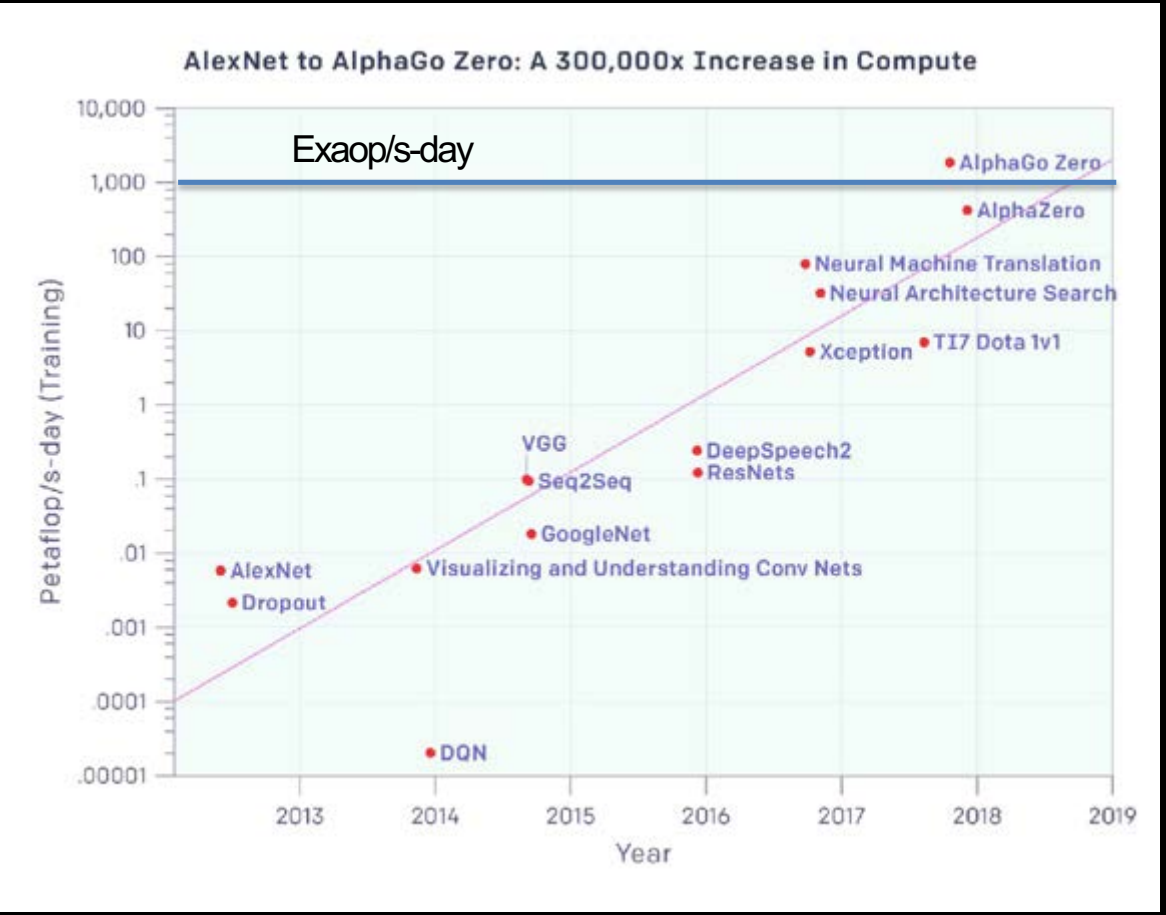
Deep Learning Meets HPC

6 orders of magnitude compute increase in 5 years

[Slide Courtesy Rick Stevens @ANL]

Exascale Needs for Deep Learning

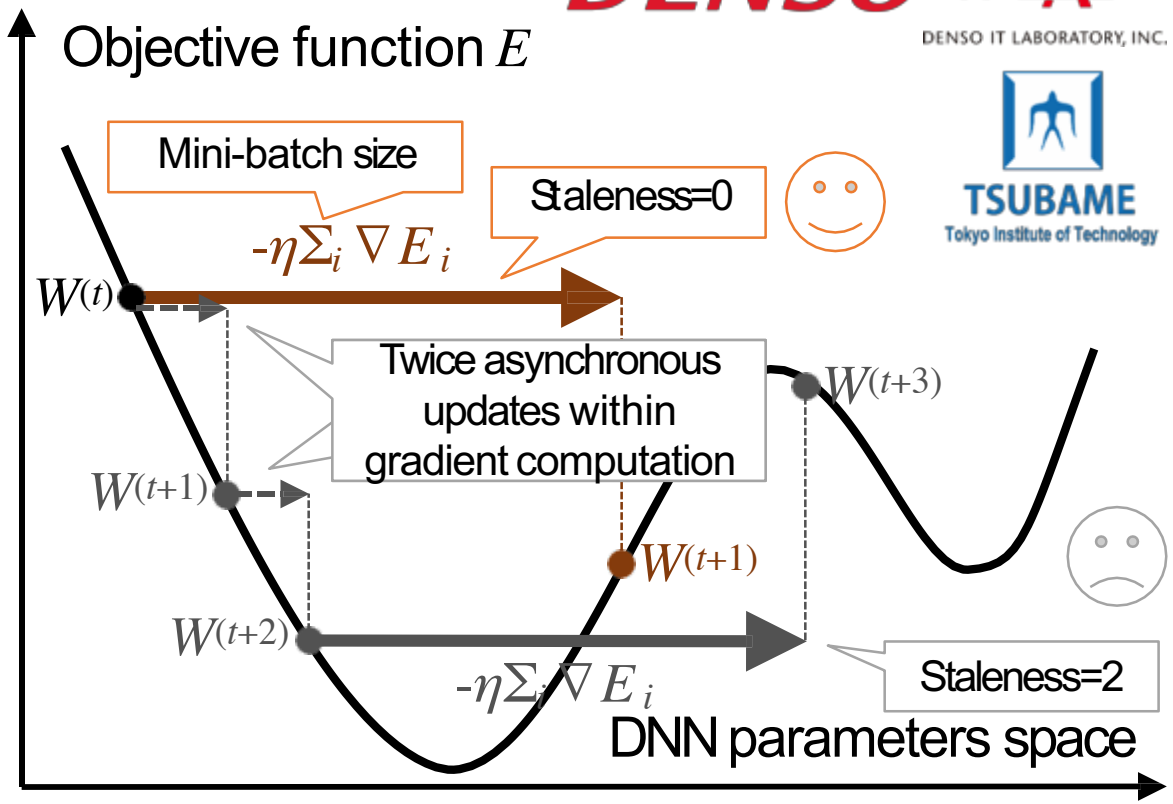
- Automated Model Discovery
- Hyper Parameter Optimization
- Uncertainty Quantification
- Flexible Ensembles
- Cross-Study Model Transfer
- Data Augmentation
- Synthetic Data Generation
- Reinforcement Learning



Predicting Statistics of Asynchronous SGD Parameters for a Large-Scale Distributed Deep Learning System on GPU Supercomputers

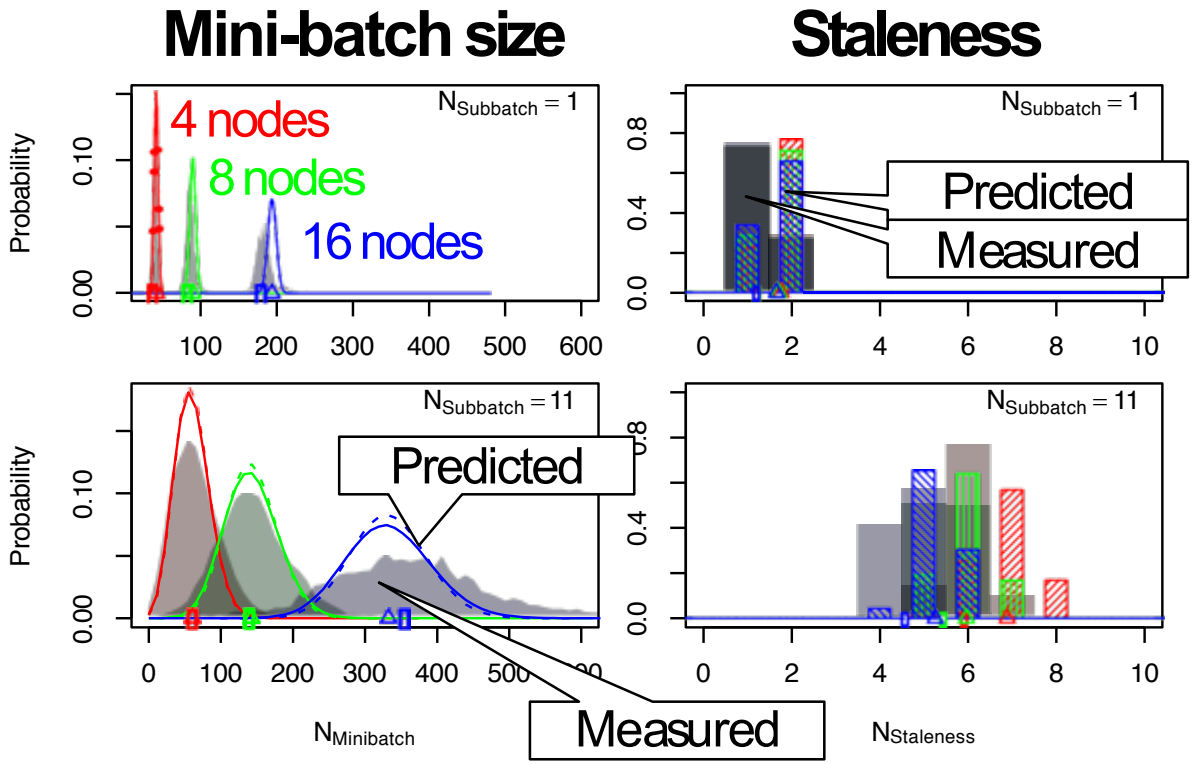
Background

- In large-scale Asynchronous Stochastic Gradient Descent (ASGD), mini-batch size and gradient staleness tend to be large and unpredictable, which increase the error of trained DNN



Proposal

- We propose a empirical performance model for an ASGD deep learning system SPRINT which considers probability distribution of mini-batch size and staleness



($N_{Subbatch}$: # of samples per one GPU iteration)

- Yosuke Oyama, Akihiro Nomura, Ikuro Sato, Hiroki Nishimura, Yukimasa Tamatsu, and Satoshi Matsuoka, "**Predicting Statistics of Asynchronous SGD Parameters for a Large-Scale Distributed Deep Learning System on GPU Supercomputers**", in proceedings of 2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington D.C., Dec. 5-8, 2016

Massive Scale Deep Learning on Fugaku

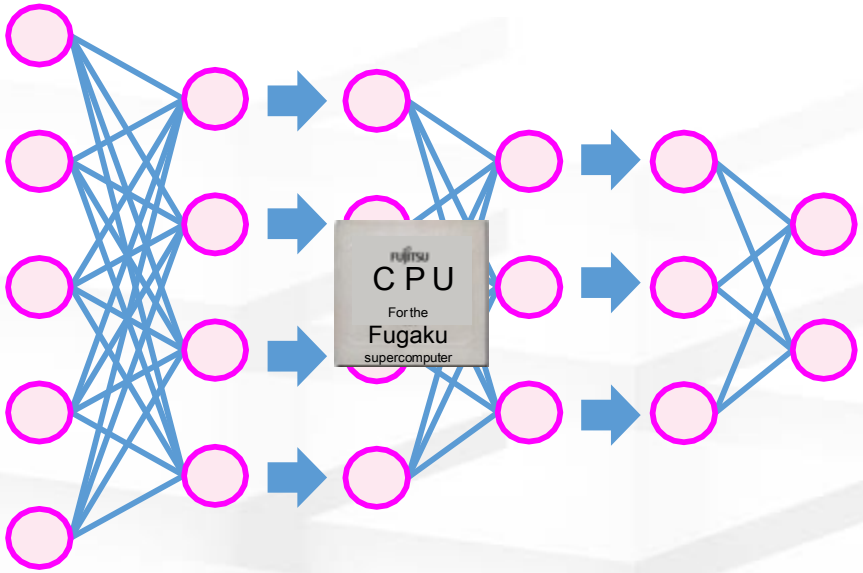
Fugaku Processor

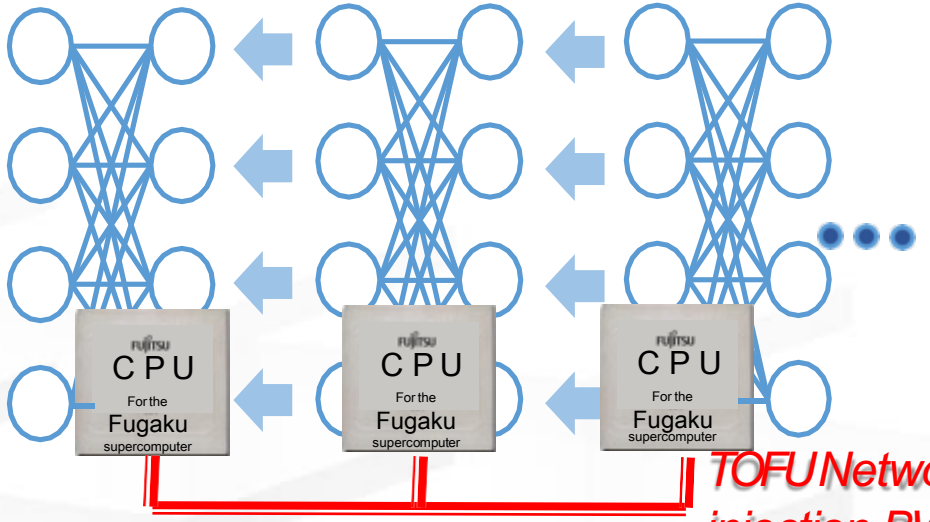
- High perf FP16&Int8
- **High mem BW for convolution**
- **Built-in scalable Tofu network**

Unprecedened DL scalability

High Performance and Ultra-Scalable Network for massive scaling model & data parallelism

High Performance DNN Convolution





TOFU Network w/high injection BW for fast reduction

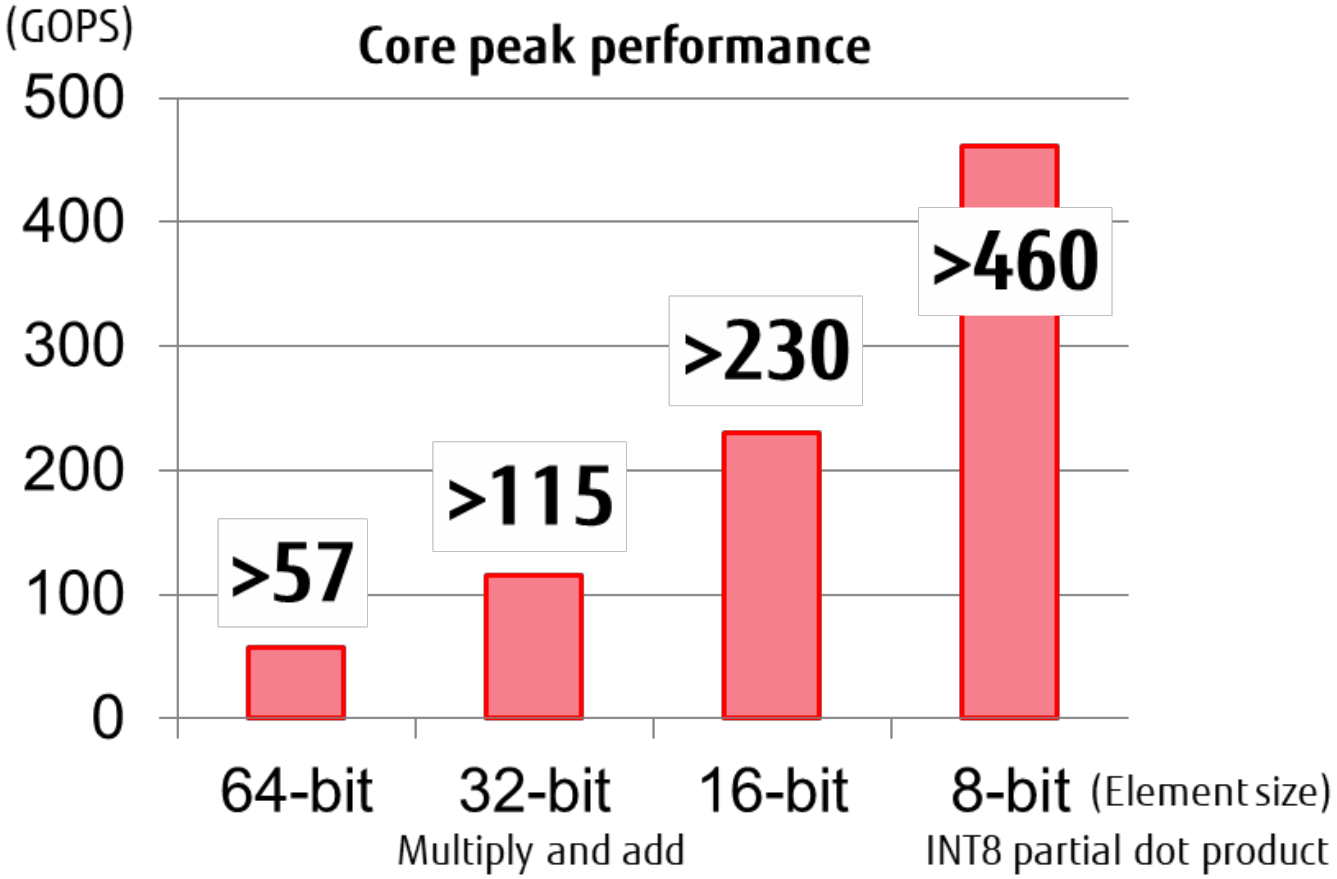
Low Precision ALU + High Memory Bandwidth + Advanced Combining of Convolution Algorithms (FFT+Winograd+GEMM)

Unprecedented Scalability of Data/

A64FX technologies: Core performance

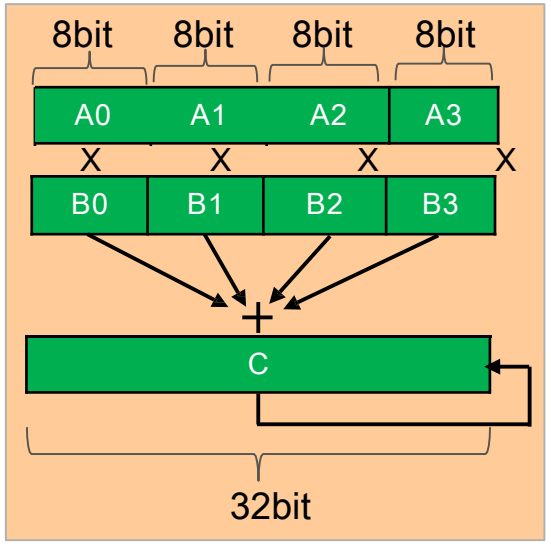
FUJITSU

- High calc. throughput of Fujitsu's original CPU core w/ SVE
  - 512-bit wide SIMD x 2 pipelines and new integer functions



INT8 partial dot product

$C = \Sigma\ (Ai \times Bi) + C$



© 2019 FUJITSU

2021/4/22 A64fx 試作CPU A版測定結果(node performance)

Machine	GEMM パラメータ			Intel(R) Skylake				NVIDIA Volta				PRIMEHPC FX100		post-K A CPU			
Precision				single (JIT)		single(gemm)		single		half		single		single		half	
core				20core		20core		CUDA core		TensorCore		16core x 2CMG		12core x 4CMG		12core x 4CMG	
	M	N	K	effficency	TFlops	effficency	TFlops	effficency	TFlops	effficency	TFlops	effficency	TFlops	effficency	TFlops	effficency	TFlops
①	512	392	4608	82.9%	2.545	30.9%	0.950	32.7%	4.577	9.7%	10.818	74.0%	1.497	79.8%	4.903	79.3%	9.744
②	32	12544	4800	84.3%	2.590	27.6%	0.847	66.5%	9.311	26.4%	29.515	12.6%	0.255	50.2%	3.084	26.2%	3.219
③	512	784	4800									84.2%	1.703	88.7%	5.450	87.1%	10.703
④	256	25088	64	73.8%	2.267	38.4%	1.180	77.3%	10.815	46.9%	52.538	47.8%	0.967	64.7%	3.975	65.0%	7.987
⑤	2048	3136	64									61.8%	1.250	24.7%	1.518	48.8%	5.997
⑥	1024	3136	512	82.9%	2.546	46.6%	1.432	73.7%	10.313	23.4%	26.210	92.3%	1.867	82.6%	5.075	90.2%	11.084
⑦	2048	1568	512									86.4%	1.747	68.3%	4.196	83.3%	10.236

- 測定条件
  - Skylake: Intel(R) Xeon(R) Gold CPU 6148, Volta: NVIDIA Tesla V100-PCIE-16GB, FX100, Fugaku A版CPUのnode当たりのgemmの効率と性能を比較した．
  - FX100，Fugaku A版CPUの測定は1CMGで行っており，node性能は換算している．
  - Skylakeの①,②,④,⑥並びにVolta CUDA coreの測定パターン①では，img2colなどの前後処理を含む．
  - ピーク性能は，Skylakeが3.072TFlops, Volta CUDA coreは単精度14TFlops, Volta TensorCoreは半精度112TFlops, FX100は1node単精度2.0224TFlops，Fugakuは，1node単精度6.144TFlops, 半精度12.288TFlopsである．
- 結果
  - Volta CUDA core単精度での効率が高い．逆にTensorCore半精度ではピーク性能が8倍であるため，性能は上がるが，効率は相対的に低い．
  - Skylakeでは，JITのノード性能が概ね高い．gemmの効率は低い．
  - Fugaku試作A版CPUでの効率は，SkylakeでのJIT効率やVolta のCUDA coreでの効率と同程度である．
  - Fugaku試作A版CPUでの半精度効率は，概ね単精度と同程度の効率である．
  - gemmのM次元を大きくするMN次元のバランシングを行ったものが，②→③，④→⑤，⑥→⑦である．性能が低い②は，バランシングの効果が高いことが分かる．Fugakuの性能値としては，バランシング前後で良い方の性能が出ると考えて良い．

  - Isopower (電力一定)では、A64fx 2個 ~= Volta 1個程度であり、その補正を行うと、SingleでもTensorCore比較においても、①では2倍の性能、⑥ではほぼ同等の性能となる。一方、②④では負ける。
  - 今後、MLのフルスタックを実装し、更に最適化を行ってMLPerfなどの実環境ベンチを行う予定

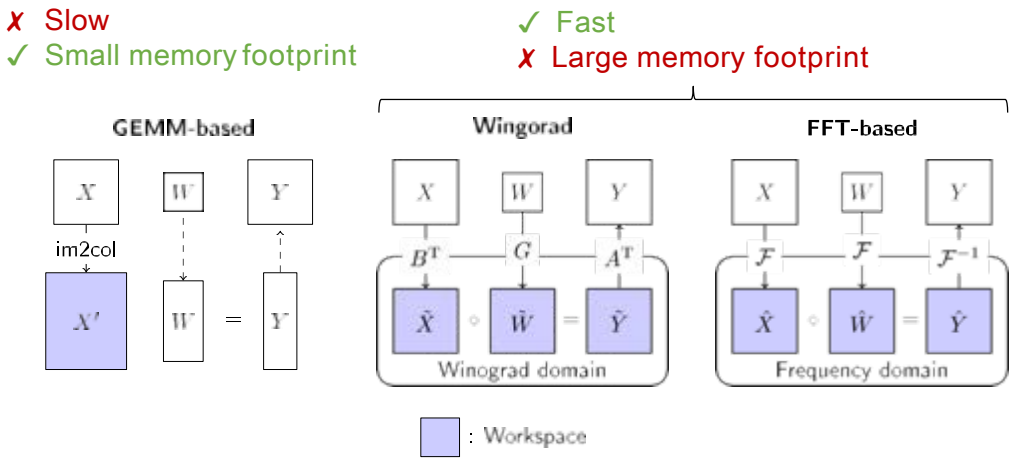
*** A64fxは NVIDIA Volta v100 GPUに対し機械学習において十分比較しうる競争力があると予想される**

Copyright 2018 RIKEN

K computer

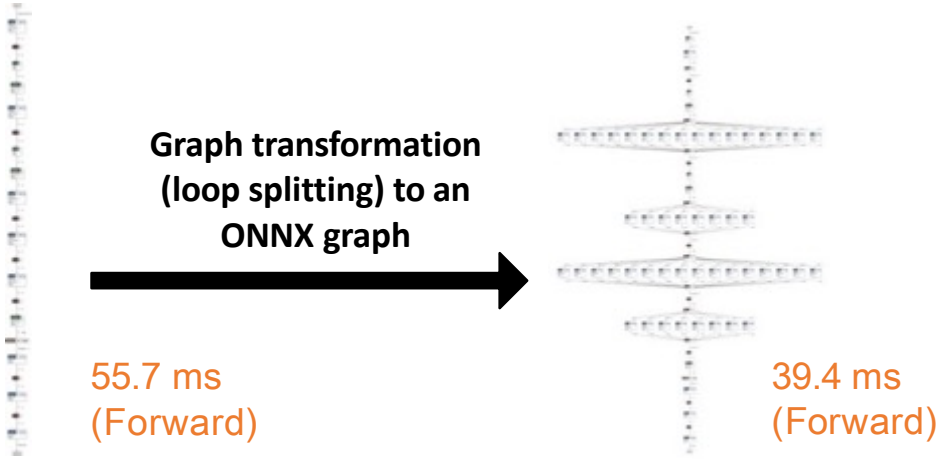
Applying Loop Transformations/Algorithm Optimizations to Deep Learning Kernels on cuDNN [1] and ONNX [2]

- **Motivation**: How can we use faster convolution algorithms (FFT and Winograd) with a small workspace memory for CNNs?
- **Proposal**: μ-cuDNN, a wrapper library for cuDNN, which applies loop splitting to convolution kernels based on DP and integer LP techniques
- **Results**: μ-cuDNN achieves significant speedups in multiple levels of deep learning workloads, achieving 1.73x of average speedups for DeepBench's 3×3 kernels and 1.45x of speedup for AlexNet on Tesla V100



Convolution algorithms supported by cuDNN

- **Motivation**: How can we extend μ-cuDNN to support arbitrary types of layers, frameworks and loop dimensions?
- **Proposal**: Apply graph transformations on the top of the ONNX (Open Neural Network eXchange) format
- **Results**: 1.41x of speedup for AlexNet on Chainer only with graph transformation and Squeezing 1.2x of average speedup for DeepBench's 3x3 kernels by multi-level splitting



AlexNet before/after the transformation

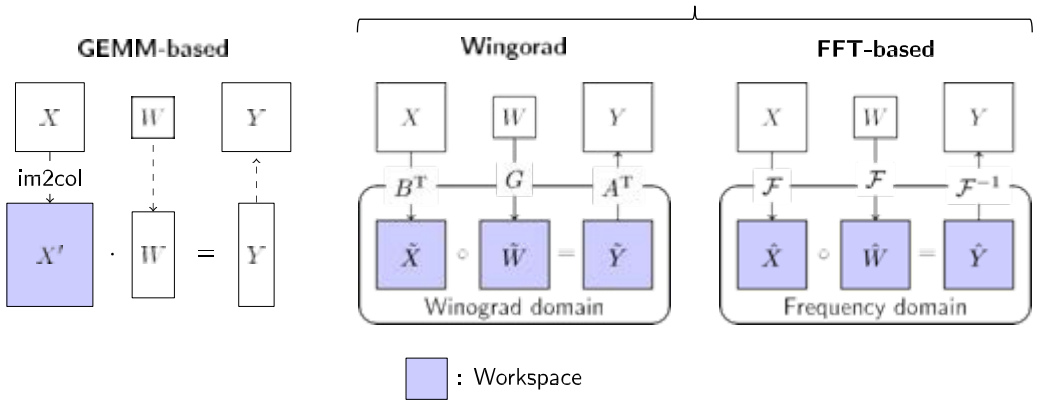
1 Yosuke Oyama, Tal Ben-Nun, Torsten Hoefler, Satoshi Matsuoka, Accelerating Deep Learning Frameworks with Micro-batches, In proceedings of IEEE Cluster 2018, Belfast UK, Sep. 10-13, 2018.
2(To appear) Yosuke Oyama, Tal Ben-Nun, Torsten Hoefler, Satoshi Matsuoka, Applying Loop Transformations to Deep Neural Networks on ONNX, 情報処理学会研究報告, 2019-HPC-170. In 並列/分散/協調処理に関するサマーワークショップ(SWoPP2019), Jul. 24-26, 2019.

μ-cuDNN: Accelerating Deep Learning Frameworks with Micro-batches [1]

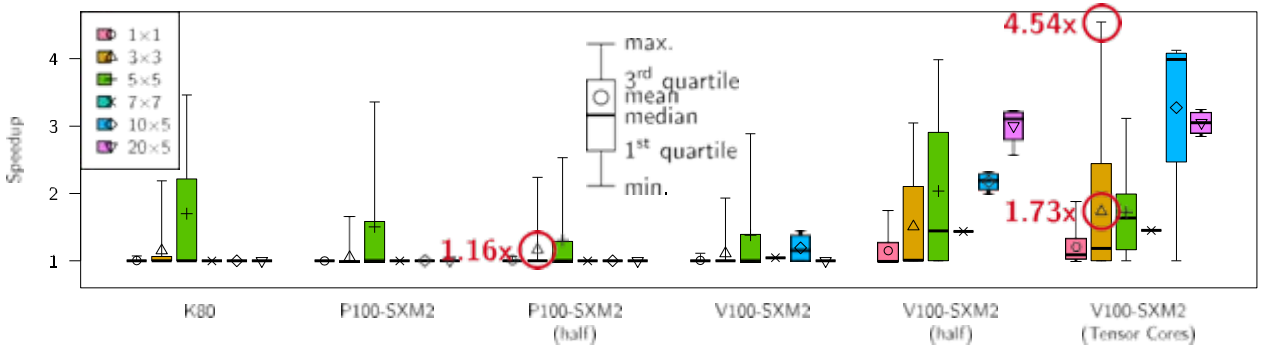
- **Motivation**: How can we use faster convolution algorithms (ex. FFT and Winograd) with a small workspace memory for Convolutional Neural Networks (CNNs)?
- **Proposal**: μ-cuDNN, a wrapper library for the math kernel library cuDNN which is applicable for most deep learning frameworks
  - μ-cuDNN applies loop splitting by using dynamic programming and integer linear programming techniques
- **Results**: μ-cuDNN achieves significant speedups in multiple levels of deep learning workloads
  - 1.16x, 1.73x of average speedups for DeepBench's 3×3 kernels on Tesla P100 and V100 respectively
  - achieves 1.45x of speedup (1.60x w.r.t. convolutions alone) for AlexNet on V100

✗ Slow
✓ Small memory footprint

✓ Fast
✗ Large memory footprint



Convolution algorithms supported by cuDNN



Relative speedups of DeepBench's forward convolution layers against cuDNN

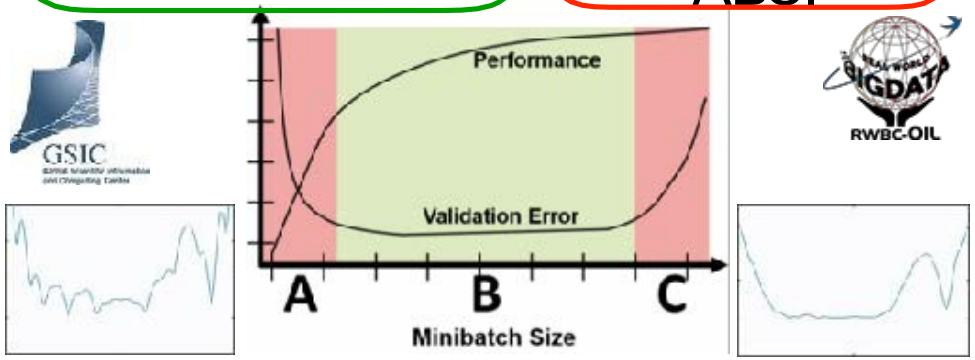
[1] Yosuke Oyama, Tal Ben-Nun, Torsten Hoefler, Satoshi Matsuoka, Accelerating Deep Learning Frameworks with Micro-batches, In proceedings of IEEE Cluster 2018, Belfast UK, Sep. 10-13, 2018.

Training ImageNet in Minutes

RioYokota,Kazuki Osawa,YoheiTsuji,Yuichiro Ueno,Hiroki Naganuma,Shun Iwase,Kaku Linsho,
Satoshi Matsuoka Tokyo Institute ofTechnology/Riken
+Akira Naruse (NVIDIA)

TSUBAME3.0

ABCI

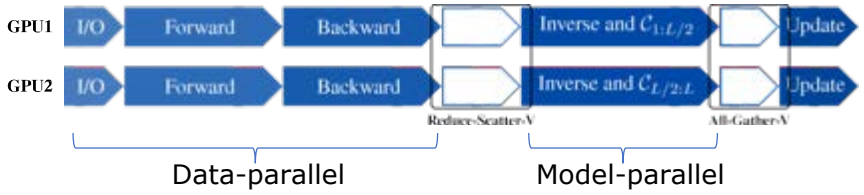


Source Ben-nun & Hoefler https://arxiv.org/pdf/1802.09941.pdf

	#GPU	time
Facebook	512	30 min
Preferred Networks	1024	15 min
UC Berkeley	2048	14 min
Tencent	2048	6.6 min
Sony (ABCI)	~3000	3.7 min
Google (TPU/GCC)	1024	2.2 min
TokyoTech/NVIDIA/Riken (ABCI)	4096	? min

Accelerating DL with 2nd Order Optimization and Distributed Training [Tsuji et al.] => *Towards 100,000 nodes scalability*

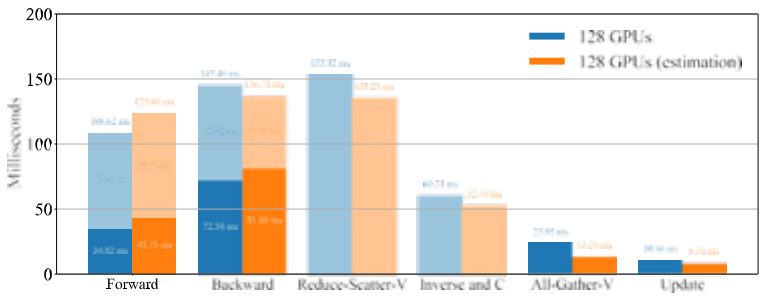
- Background
  - Large complexity of DL training.
  - Limits of data-parallel distributed training.
  - *> How to accelerate the training further?*
- Method
  - Integration of two techniques: 1) data- and model-parallel distributed training, and 2) K-FAC, an approx 2nd order optimization.
- Evaluation and Analysis
  - Experiments on ABCI supercomputer.
  - Up to 128K batch size w/o accuracy degradation.
  - Finish training in 35 epochs/10 min/1024 GPUs in 32K batch size.
  - A performance tuning / modeling.



Design our hybrid parallel distributed K-FAC

	Batch size	# Iterations	Accuracy
Goyal et al.	8K	14076	76.3%
Akiba et al.	32K	3519	75.4%
Ying et al.	64K	1760	75.2%
Ours	**128K**	**978**	**75.0%**

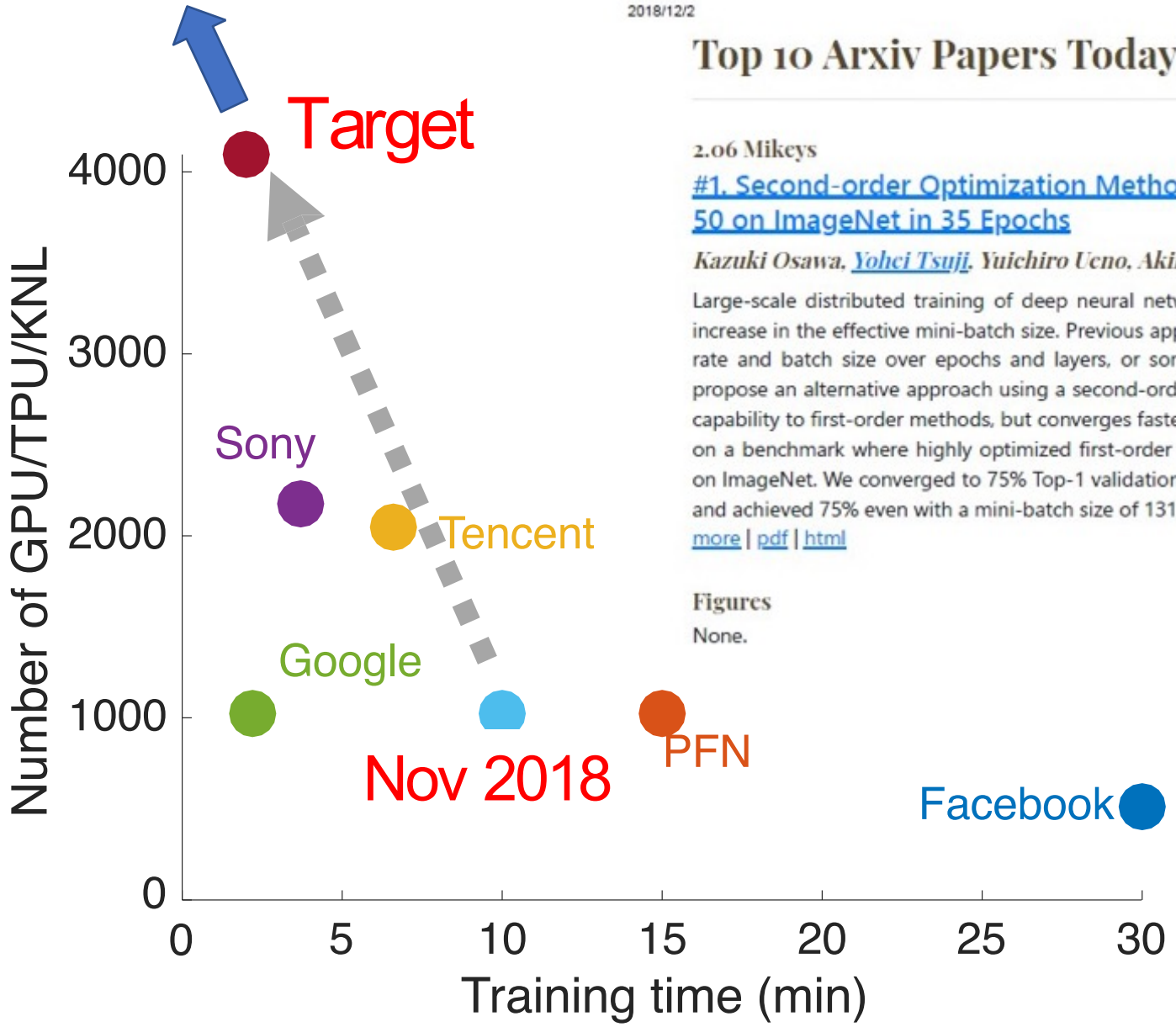
Comparison with related work (ImageNet/ResNet-50)



Time prediction with the performance model

Osawa et al., Large-Scale Distributed Second-Order Optimization Using Kronecker-Factored Approximate Curvature for Deep Convolutional Neural Networks, CVPR 2019

Fast ImageNet Training



2018/12/2 Assert

Top 10 Arxiv Papers Today in Computer Science

2.06 Mikeys

#1. Second-order Optimization Method for Large Mini-batch: Training ResNet-50 on ImageNet in 35 Epochs

Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, Satoshi Matsuoka

Large-scale distributed training of deep neural networks suffer from the generalization gap caused by the increase in the effective mini-batch size. Previous approaches try to solve this problem by varying the learning rate and batch size over epochs and layers, or some ad hoc modification of the batch normalization. We propose an alternative approach using a second-order optimization method that shows similar generalization capability to first-order methods, but converges faster and can handle larger mini-batches. To test our method on a benchmark where highly optimized first-order methods are available as references, we train ResNet-50 on ImageNet. We converged to 75% Top-1 validation accuracy in 35 epochs for mini-batch sizes under 16,384, and achieved 75% even with a mini-batch size of 131,072, which took 100 epochs.

more | pdf | html

Figures

None.

ImageNet/ResN … world record)

Yoshiki Tanaka, Hisahiro Su … yama Sony Corporation

Succes… … rk

Librarie…

Neural Network Libraries

Issues … ster

Issue 1
Commun…
synchron…

Approac…
2D-Toru…
rings in h…

RI…
RI…
RING…

arXiv:1811.12019v2 [cs.LG] 5 Dec 2018

Second-order Op…
Training R…

Kazuki Osawa[1] Yohei Tsuji[1] Yu…
[1]School o…
[2]Global Scientific Informe…
[4]RIKI…
ana…

Tokyo …
IEEE CV…
Batch size

Abstract

Large-scale distributed training of deep … suffer from the generalization gap caused … in the effective mini-batch size. Previous … solve this problem by varying the learning … size over epochs and layers, or some … tion of the batch normalization. We prop… tive approach using a second-order optim… that shows similar generalization capabili… methods, but converges faster and can han… batches. To test our method on a benchma… optimized first-order methods are availab… we train ResNet-50 on ImageNet. We co… Top-1 validation accuracy in 35 epochs for … under 16,384, and achieved 75% even wi… size of 131,072, which took 100 epochs.

1. Introduction

As the size of deep neural network mod… which they are trained on continues to incr… demand for distributed parallel computing … common approach for achieving distribute… deep learning is to use the data-parallel … the data is distributed across different proc… model is replicated across them. When the … per process is kept constant to increase th… putation over communication, the effective … over the entire system grows proportional … processes.

When the mini-batch size is increased … point, the validation accuracy starts to degr…

arXiv:1903.12650v1 [cs.LG] 29 Mar 2019

Yet Another Accelerated SGD: ResNet-50 Training on ImageNet in 74.7 seconds

Masafumi Yamazaki, Akihiko Kasagi, Akihiro Tabuchi, Takumi Honda, Masahiro Miwa,
Naoto Fukumoto, Tsuguchika Tabaru, Atsushi Ike, Kohta Nakashima
Fujitsu Laboratories Ltd.
{m.yamazaki, kasagi.akihiko, tabuchi.akihiro, honda.takumi, masahiro.miwa,
fukumoto.naoto, tabaru.tsuguchika, ike.atsushi, nakashima.kohta}@fujitsu.com

Fujitsu Lab
2048 GPUs on ABCI
74.7 seconds
Batch size 81,920

***Abstract*—There has been a stro… can execute machine learning as fa… of deep learning has accelerated by 30 times only in the past two years. Distributed deep learning using the large mini-batch is a key technology to address the demand and is a great challenge. … it is difficult to achieve high scalability on large clusters with … compromising accuracy. In this paper, we introduce optimization methods which we applied to this challenge. We achieved the training time of 74.7 seconds using 2,… applying these methods. The training throughput is over 2.52 million images/sec and the top-1 validation accuracy is 75.08%.**

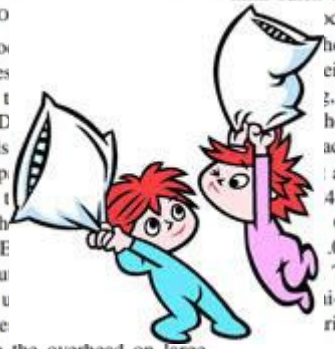
I. Introductio…

Deep neural network (DNN) mo… datasets are delivering impressive res… such as object detection, language t… However, the computation cost of D… larger since the sizes of DNN models … Distributed deep learning with data p… be an effective approach to accelerate t… In this approach, all processes launch… the same DNN model and weights. E… model with different mini-batches bu… from all processes are combined to u… This communication overhead become… for large clusters. In order to reduce the overhead on large clusters, we increase mini-batch size of DNN and compute DNN trainings in parallel. However, the training with large mini-batch generally results in the worse validation accuracy of DNN models. Thus, we used several techniques to increase mini-batch size, which denotes the number of input images computed in an iteration, without compromising validation accuracy.

We performed our experimental result, using 2,048 GPUs of AI Bridging Cloud Infrastructure (ABCI) cluster and self-optimized MXNet deep learning framework. We achieved 75.08% validation accuracy of ResNet-50 on ImageNet using 81,920 mini-batch size in 74.7 seconds.

II. Related works

This section introduces the related works about the large mini-batch challenges. Alex et al. [8] achieved high accuracy for the image recognition in ILSVRC. This paper shows that convolutional layers are effective for 2D image deep neural



use convolu… for 2D and 3D image data. Ioffe et … h normalization technique, in which the feature values in hidden layers are normalized to avoid vanishing gradients. In addition, this technique enables training … large number of layers, such as ResNet.

Generally, the mini-batch size should be large for distributed … clusters. Goyal et al. [2] proposed the … deep the validation accuracy with 8,192 mini-batch size. Google [3] and Sony [7] used the variable mini-batch size which becomes larger and achieved highly … cessing.

…he difference between the weight gradient norm … eight norm of each layer causes the unstable of … LARS of [10] normalizes the difference of each … he DNN can train with 32,768 without the loss of accuracy.

… al. [4] achieved ResNet-50 training in 15 minutes … 4 GPUs. Jia et al. [5] also achieved ResNet-50 … 6.6 minutes using 2,048 GPUs. Ying et al. [6] … .05 million images/sec by using 1,024 TPU v3 … The training times of ResNet-50 with 32,768 and … i-batch sizes are 2.2 and 1.8 minutes. These results … rized in the table I.

III. Our approach

In this section, we introduce our techniques applied to improve both accuracy and training throughput.

A. Accuracy Improvement

We used Stochastic Gradient Descent (SGD) that is commonly used for deep learning optimizer. When training on large mini-batch, the number of SGD updates decreases as mini-batch size increases, so improving final validation accuracy on large mini-batch is a big challenge, and we adopted the following techniques.

1) Learning Rate Control: We need to use high learning rate to accelerate training due to the small number of updates. However, high learning rate makes training of models unstable in early stages. Thus, we stabilize SGD by using the warm-up [2] which raises leaning rate gradually. Moreover, the same learning rate of all layer is too high for some layers.

…puter Scale

…oulong Cheng

…oogle 2.2min

…dware vendors have … ining deep learning … thmic and systems … lated optimizations: … ch sizes, (2) input … torus all-reduce to … to train ResNet-50 … TPU v3 Pod with a … accuracy drop.

…as been driven by dramatic … al networks use some variant … ls typically require multiple … bstantial number of floating … ple, on ImageNet [14], the … of 3.2 × 10^{18} floating point … epochs to converge.

… iteration times, large neural … accelerator. One commonly … devices via distributed SGD … y, asynchronous distributed … nt work [4] has shown that … erms of time to convergence … distributed SGD, large-scale … l to maintain model quality. … e real-world end-to-end wall

…e for …scape

epoch

…19K

…720GPU 3808GPU

Toward Training a Large 3D Cosmological CNN with Hybrid Parallelization

[1] Tokyo Institute of Technology, [2] Lawrence Livermore National Laboratory, [3] University of Illinois at Urbana-Champaign, [4] Lawrence Berkeley National Laboratory, [5] RIKEN Center for Computational Science, * oyama.y.aa@m.titech.ac.jp

August 5, 2019

The 1st Workshop on Parallel and Distributed Machine Learning 2019 (PDML'19) – Kyoto, Japan

Yosuke Oyama [1,2,*], Naoya Maruyama [2], Nikoli Dryden [3,2], Peter Harrington [4], Jan Balewski [4], Satoshi Matsuoka [5,1], Marc Snir [3], Peter Nugent [4], and Brian Van Essen [2]

LLNL-PRES-XXXXXX

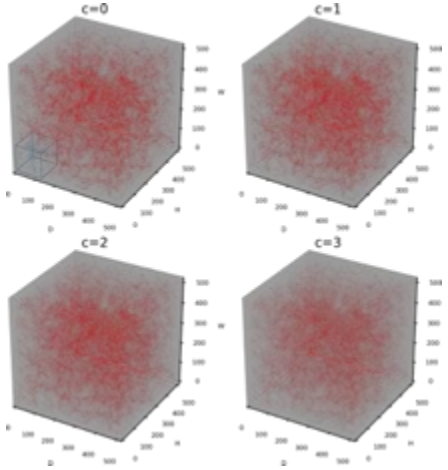
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Background

- **CosmoFlow** [1] is a project to estimate cosmological parameters from 3-dimensional universe data by using a 3D CNN

Input
($4 \times 512 \times 512 \times 512$ voxels)



Input → **CNN** **53 GiB** → Predict →

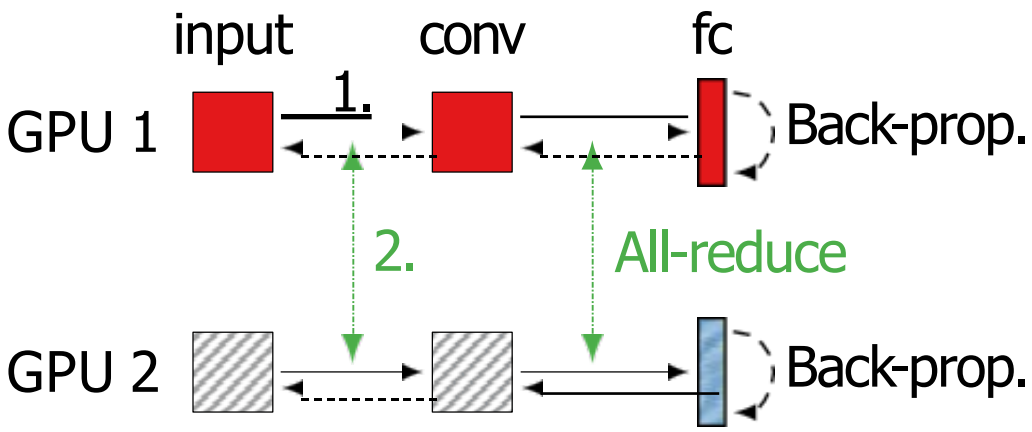
$$\begin{bmatrix} \Omega_m \\ \sigma_8 \\ n_s \end{bmatrix} = \begin{bmatrix} 0.242 \\ 0.145 \\ -0.489 \end{bmatrix}$$

Output
(A vector of length 4)

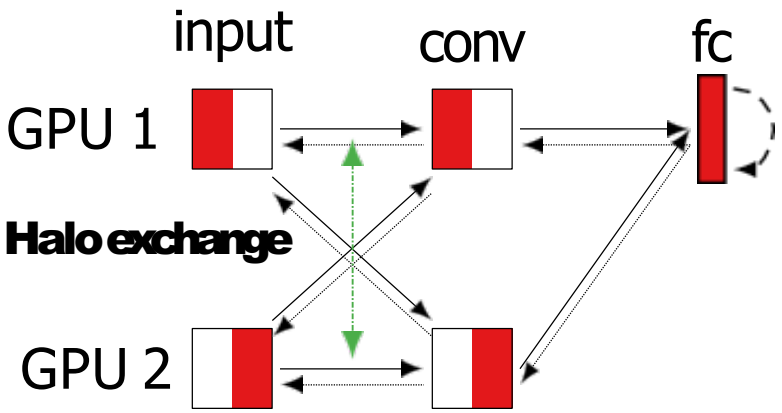
- **Problem**: GPU memory is too small to process high-resolution universe data
  → Another way to parallelize the model efficiently?

Background

- **Data-parallel** training distributes data samples among GPUs
  - ✓ Good weak scalability (O(1000) GPUs)



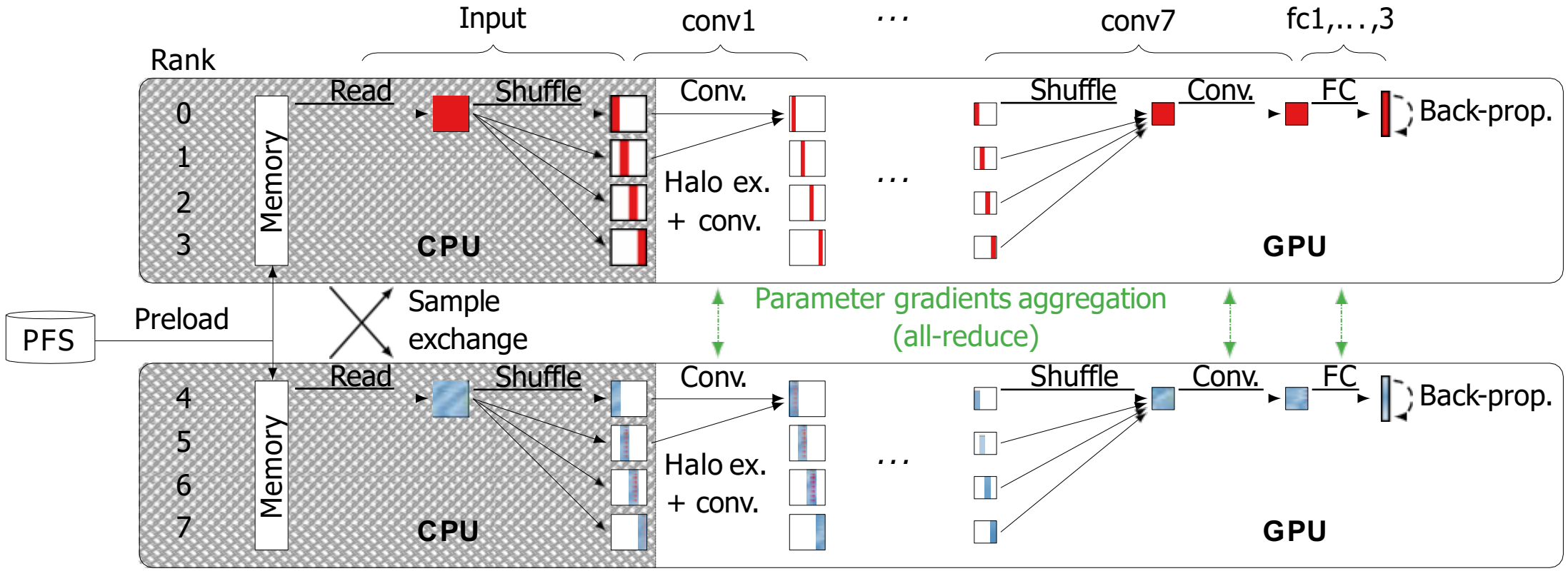
- **Model-parallel** training distributes the computation of a single sample (model) among GPUs
  - ✓ Can use more GPUs per sample
  - ✓ Can train larger models



- Data-parallelism + model-parallelism = **Hybrid-parallelism**

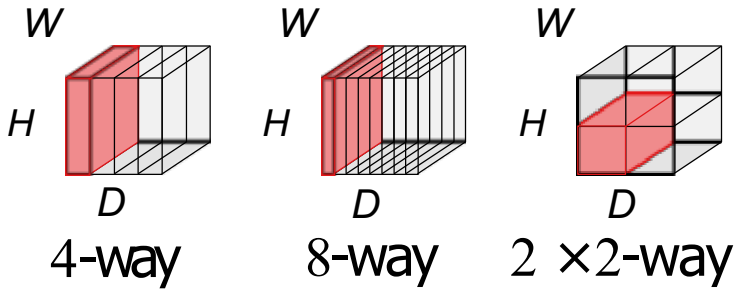
Proposal: Extending Distconv for 3D CNNs

- **LBANN** + **Distconv** [2]: A parallelized stencil computation-like hybrid-parallel CNN kernel library



Evaluation: Weak scaling

- Achieved **111x** of speedup over 1 node by exploiting hybrid-parallelism, even if layer-wise communication is introduced
- The 8-way partitioning is **1.19x of 4-way partitioning** with a mini-batch size of 64



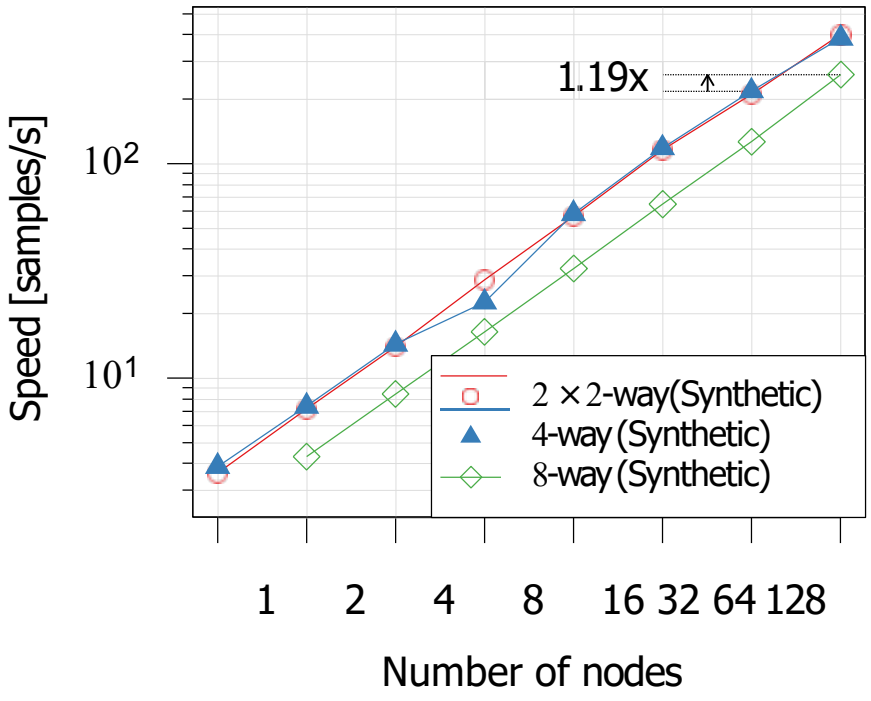


Figure: Weak scaling of the CosmoFlow network.

Evaluation: Strong scaling

- Achieved **2.28x** of speedup on 4 nodes (16 GPUs) compared to one node when $N = 1$
- The scalability limit here is 8 GPUs, and the main bottleneck is input data loading

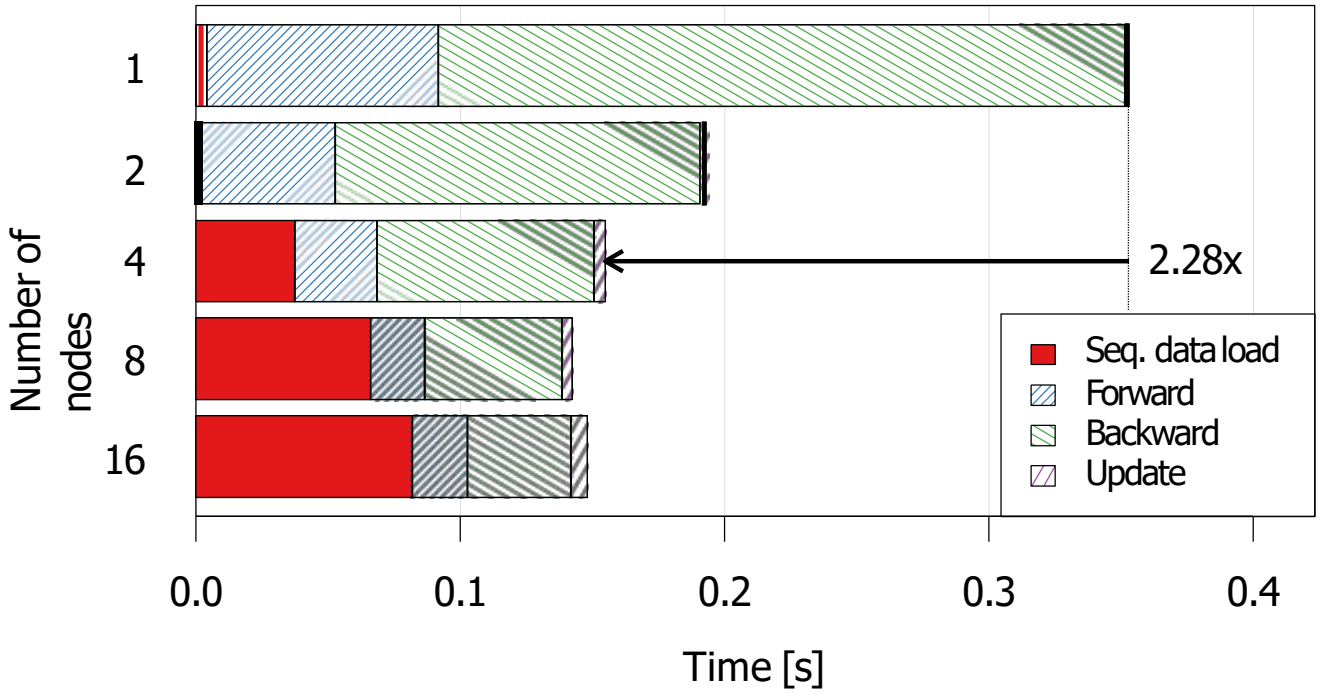


Figure: Breakdown of the strong scaling experiment when $N = 1$.

Breaking the limitation of GPU memory for Deep Learning

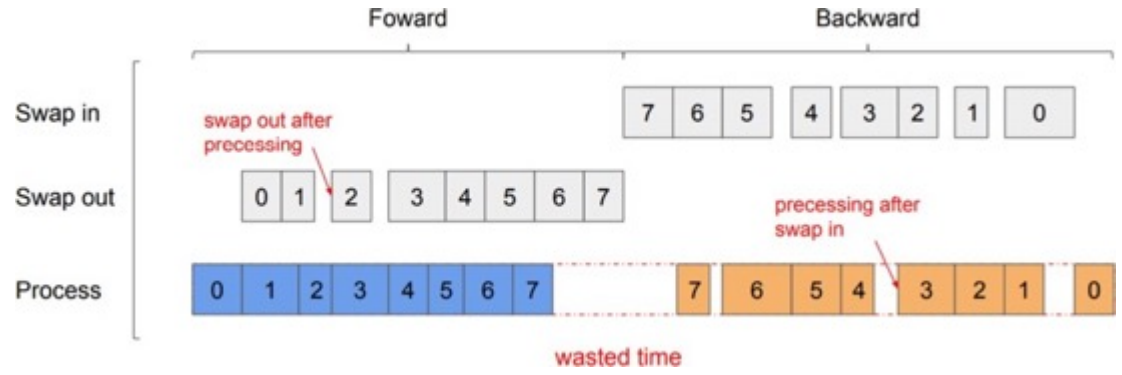
Haoyu Zhang,Wahib Mohamed, Lingqi Zhang,Yohei Tsuji,Satoshi Matsuoka

Motivation: GPU memory is relatively small in comparison to recent DL work load

Analysis:

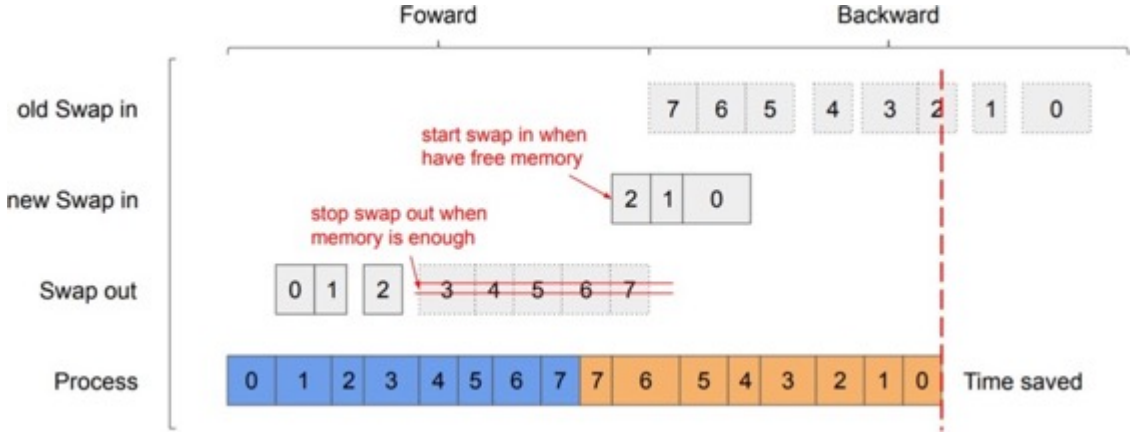
- vDNN-like strategy

$$Util = T_{Proc}/T$$
$$= \frac{Num\_buffer(l)/Proc_{Th}(l)}{max(Num\_buffer(l)/Proc_{Th}(l), Num\_buffer(l-1)/Swap\_in_{Th})}$$



- Capacity based strategy

$$Util = \begin{cases} \frac{Num\_buffer(l)/Proc_{Th}(l)}{max(Num\_buffer(l)/Proc_{Th}(l), Num\_buffer(l-1)/Swap\_in_{Th})}, \\ 1, \quad T < \frac{\sum_{i=1}^{Catch}(Proc_{th}(i) \times T_{proc}(i))}{Swap\_in_{Th}} \end{cases}$$



Breaking the limitation of GPU memory for Deep Learning

Haoyu Zhang, Wahib Mohamed, Lingqi Zhang, Yohei Tsuji, Satoshi Matsuoka

Proposal：

OOC-Paleo

Paleo's computation time estimation
- GPU
- Algrithm
- ...

+

Data transfer time estimation
- Connection
- Far Memory
- Near Memory
- ...

+

Data transfer strategy
- vDNN-like
- Capacity-Based
- ...

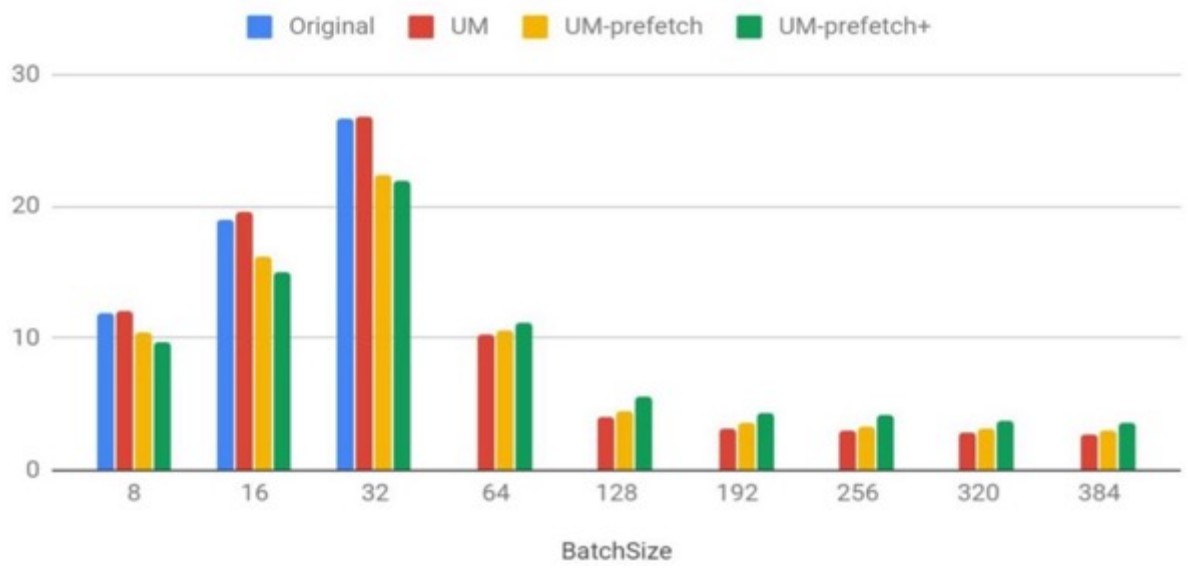
↓

OOC-training time estimation

UM-Chainer

prefetch()->explicit swap-in

no explicit swap-out



Case Study & Discussion:

Memory Capacity:
- Not so important as latency and throughput

Latency:
- Higher Bandwidth make no sense when buffer is too small
- Latency is decided by physical law

Bandwidth:
- Higher connection bandwidth
- Lower Memory bandwidth

Processor:
- Slower processor is acceptable

Breaking the limitation of GPU memory for Deep Learning

Haoyu Zhang,Wahib Mohamed, Lingqi Zhang,Yohei Tsuji,Satoshi Matsuoka

Assuming we have higher Bandwidth...

Resnet50,Batch-size=128

16GB/s->64GB/s:
Training time can be half

64GB/s->128GB/s:
Only a little time reduced

>128GB/s:
Most of the layers can not make full use of the bandwidth

>512GB/s:
Time almost do not decrease

Bandwidth	Time	percentage (bandwidth not full)	percentage (computation can not overlap)
16	967.8595572	0.409	0.733
32	569.9550342	0.466	0.642
64	407.2978908	0.574	0.472
128	371.9318064	0.688	0.438
256	362.5661138	0.835	0.398
512	359.7637498	0.915	0.398
1024	359.3012901	0.983	0.386
∞	359.3012901	1.000	0.386
Original version	306.9286403	N/A	N/A

Optimizing Collective Communication in DL Training (1 of 3)

- Reducing **training time** of large-scale AI/DL on GPUs-system.
  - Time for inference = O(seconds)
  - **Time for training = O(hours or days)**
- Computation is one of the bottleneck factors
  - Increasing the batch size and learning in **parallel**
    - Training ImageNet in 1 hour [1]
    - Training ImageNet in ~20 minutes [2]
- Communication also can become a bottleneck
  - Due to **large message sizes**

1 P.Goyal, P.Doll´ar, R.Girshick, P.Noordhuis, L.Wesolowski, A.Kyrola, A.Tulloch, Y.Jia, and K.He, "Accurate, large minibatch SGD: training imagenet in 1 hour," arXiv preprint arXiv:1706.02677, 2017.

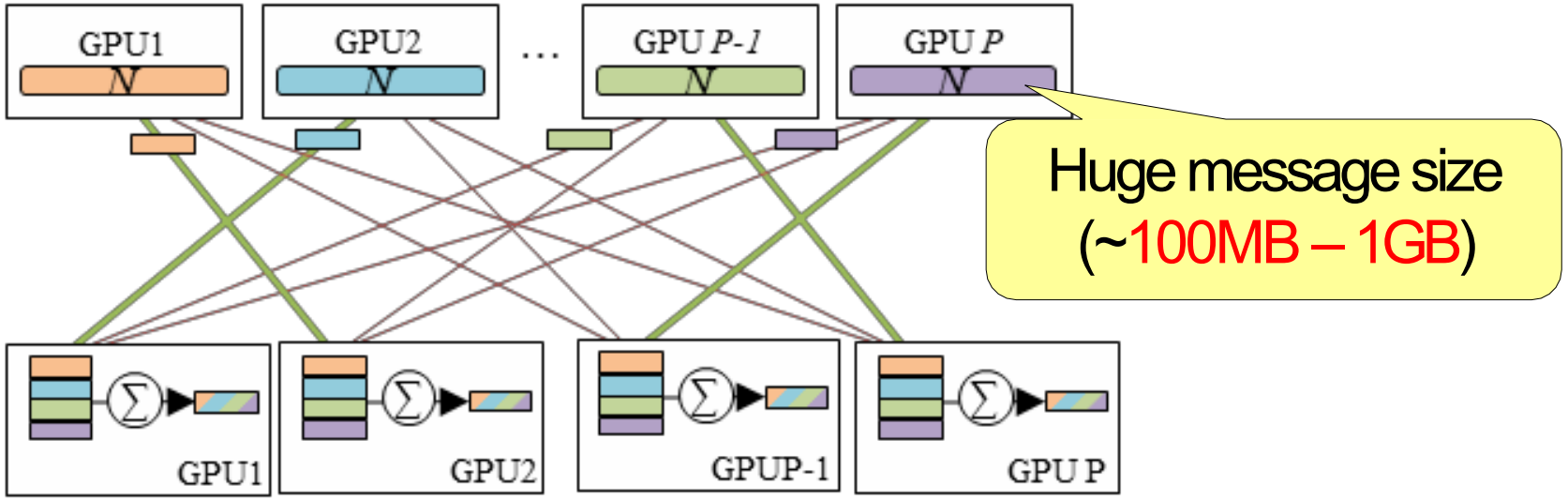
2 Y.You, Z.Zhang, C.Hsieh, J.Demmel, and K.Keutzer, "Imagenet training in minutes," CoRR, abs/1709.05011, 2017.

Optimizing Collective Communication in DL Training (2 of 3)
(Challenges of Large Message Size)

Compute the gradient G_i of the weight on each GPUi

GPUs communication to compute the mean of the gradients (Allreduce operation)

$$G = \sum_{i=1}^{P} Gi$$



Example of Image Classification, ImageNet data set

Model	AlexNet (2012)	GoogleNet (2015)	ResNet (2016)	DenseNet (2017)
# of gradients [1]	61M	5.5M	1.7 – 60.2M	15.3 – 30M
Message size	244 MB	22MB	240 MB	120 MB

[1] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," arXiv preprint arXiv:1802.09941, 2018.

Optimizing Collective Communication in DL Training (3 of 3)

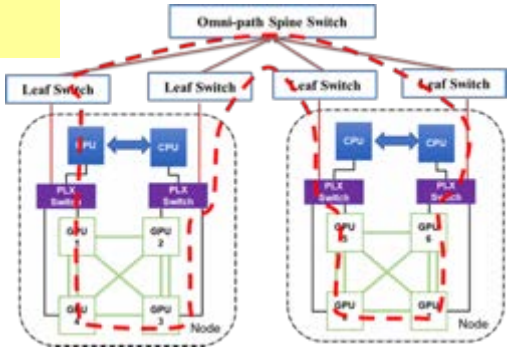
Proposal: Separate intra-node and inter-node comm. ➔ **multileader hierarchical algorithm**

- Phase 1: Intra-node reduce to the node leader
- Phase 2: Inter-node all-reduce between leaders
- Phase 3: Intra-node broadcast from the leaders

Key Results:

- Cut down the communication time up to **51%**
- Reduce the power consumption up to **32%**

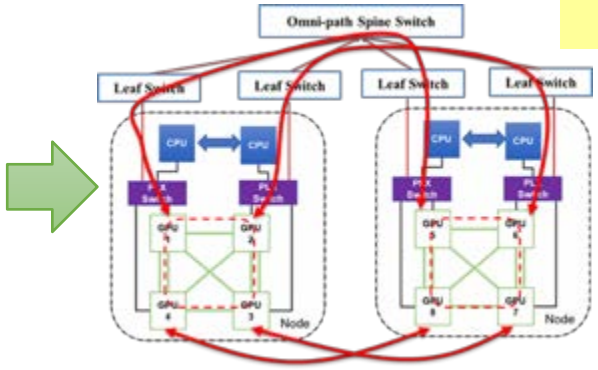
$2(P\text{-}1)$ steps, send $\frac{N}{P}$ per step



Ring-based algorithm

- Good for large message size
- Worse with inter-node comm.

$2(\frac{P}{k}\text{-}1)$ steps, $\frac{N(p-k)}{Pk}$ per step

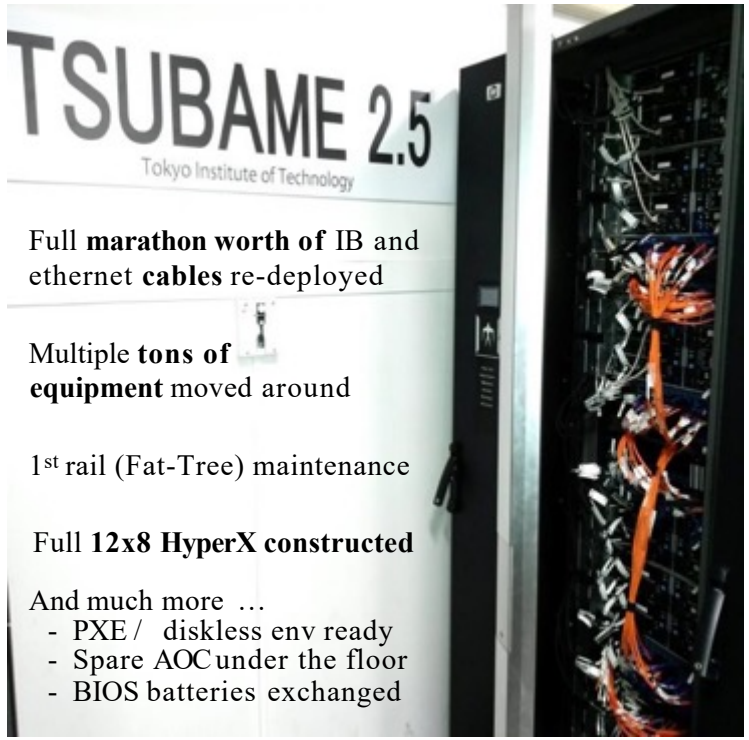


Multileader hierarchical algorithm

- Optimized for inter-node comm.

"Efficient MPI-Allreduce for Large-Scale Deep Learning on GPU-Clusters", Truong Thao Nguyen, Mohamed Wahib, Ryousei Takano, Journal of Concurrency and Computation: Practice and Experience (CCPE) , Accepted: to appear in 2019.10

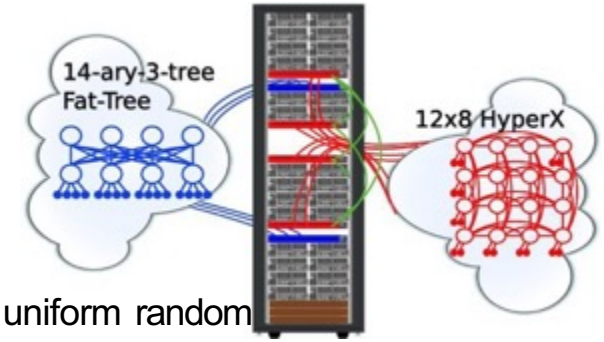
Evaluating the HyperX Topology: A Compelling Alternative to Fat-Trees?[SC19]



➔ **First large-scale 2.7 Pflop/s (DP)**

1:1 comparison (as fair as possible) of 672-node 3-level Fat-Tree and 12x8 2D HyperX

- NICs of 1st and 2nd rail even on same CPU socket
- Given our HW limitations (few "bad" links disabled)



Advantages (over FT) assuming **adaptive routing (AR)**

- **Reduced HW cost** (AOC/switches) → similar perf.
- **Lower latency** when scaling up (less hops)
- **Fits** rack-based **packaging** model for HPC/racks
- **Only needs 50% bisection BW** to provide 100% throughput for uniform random

Q1: Will reduced bisection BW (57% for HX vs. ≥100%) impede Allreduce performance?

Q2: Mitigation strategies against lack of AR? (→ eg. placement or smart routing)

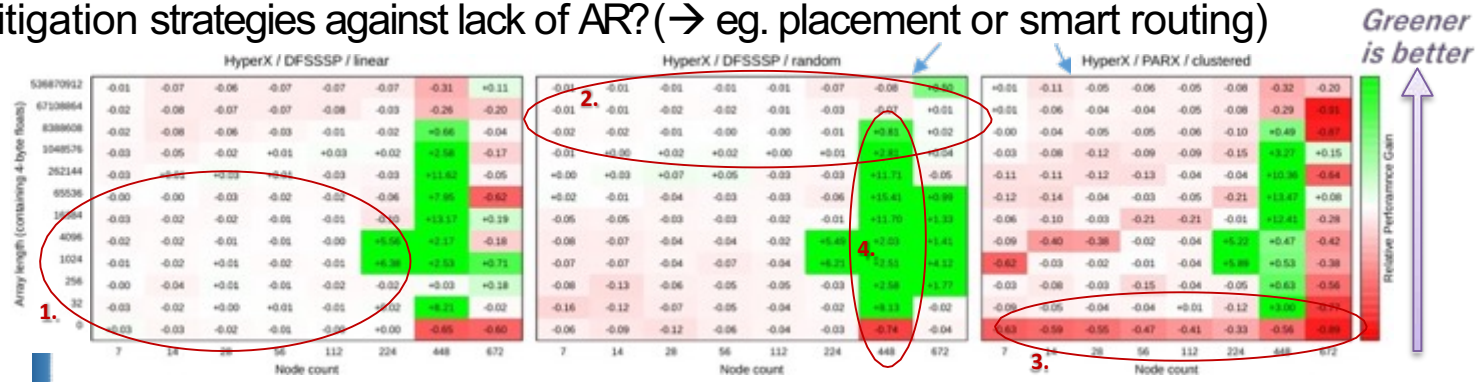


Fig.2: Baidu's (DeepBench) Allreduce (4-byte float) scaled 7→672 cn (vs. "Fat-tree / ftree / linear" baseline)

1. *Linear* good for small node counts/msg. size
2. *Random* good for DL-relevant msg. size (+- 1%)
3. Smart routing suffered SW stack issues
4. FT+ftree had bad 448-node corner case

HyperX topology is promising and cheaper alternative to state-of-the-art Fat-Tree networks!

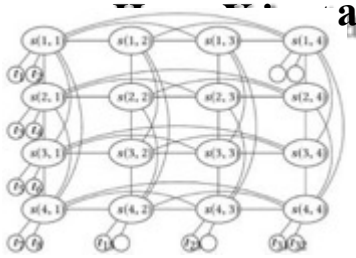


Fig.1: HyperX with n-dim. integer lattice (d_1,…,d_n) base structure fully connected in each dim.

Our 2D HyperX:

- **24 racks** (of 42 T2 racks)
- 96 QDR switches (+ 1st rail)
- 1536 IB cables (720 AOC)
- **672 compute nodes**
- **57% bisection bandwidth**

[1] Domke et al. "HyperX Topology: First at-scale Implementation and Comparison to the Fat-Tree" to be presented at SC'19 and HOTI'19

Funded by and in collaboration with Hewlett Packard Enterprise, and supported by Fujitsu, JSPS KAKENHI, and JSPS CREST

Machine Learning Models for Predicting Job Run Time-Underestimation in HPC system [SCAsia 19]

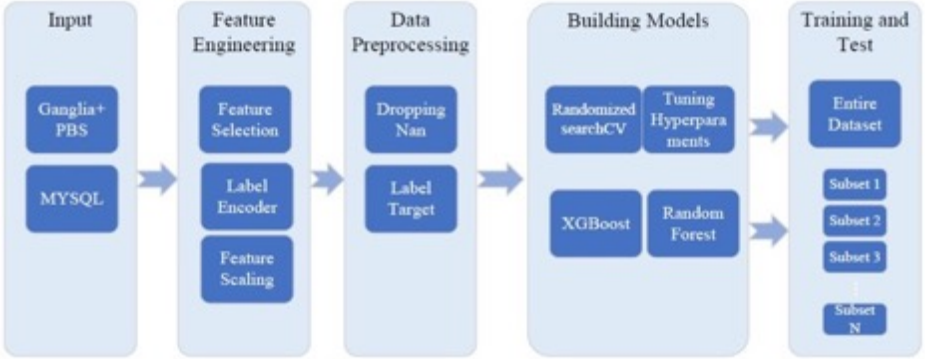
▪ Motivation & Negative effects

1. When submitting a job, users need to estimate their job runtime
2. If job runtime is underestimated by the users
3. Job will be terminated by HPC system upon reaching its time limit

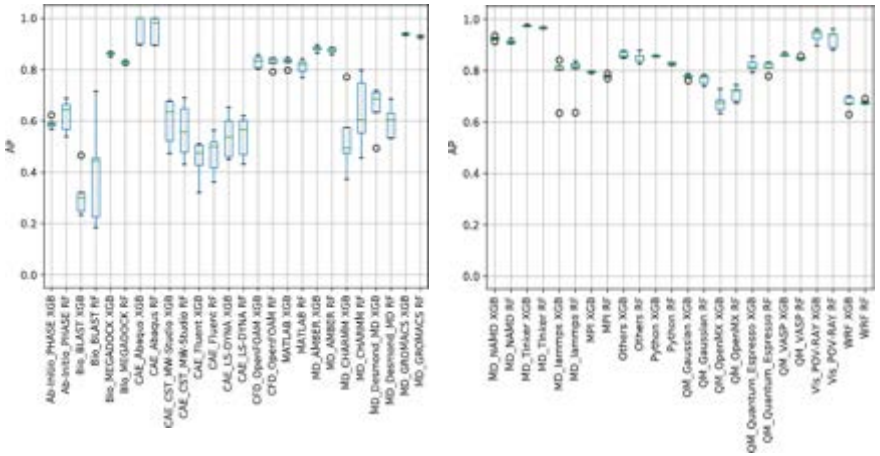
- Increasing time and financial cost for HPC users
- Wasting time and system resources.
- Hindering the productivity of HPC users and machines

▪ Method

- Apply machine learning to train models for predicting whether the user has underestimated the job run-time
- Using data produced by TSUBAME 2.5



▪ Evaluating by Average Precision(AP)



▪ Evaluating by Simulation with Saved-Lost Rate (SLR)

$$SLR_c = \frac{Saved}{Saved + Lost} = \frac{\sum_{tp=1}^{TP} (job_{tp} - C_p)}{\sum_{tp=1}^{TP} job_{tp} + \sum_{fp=1}^{FP} C_p}$$

- Runtime-underestimated jobs can be predicted with different accuracy and SLR at different checkpoint times
- Summing up the “Saved” time of all the applications at best SLRs checkpoints, 24962 hours can be saved in total with existing TSUBAME 2.5 data
- Helping HPC users to reduce time and financial loss
- Helping HPC system administrators free up computing resources

Guo, Jian, et al. "Machine Learning Predictions for Underestimation of Job Runtime on HPC System." Asian Conference on Supercomputing Frontiers. Springer, 2018