IEICE-ICT-Webinar, 1月19日 2021年

スーパーコンピュータ「富岳」の開発とコデザイン

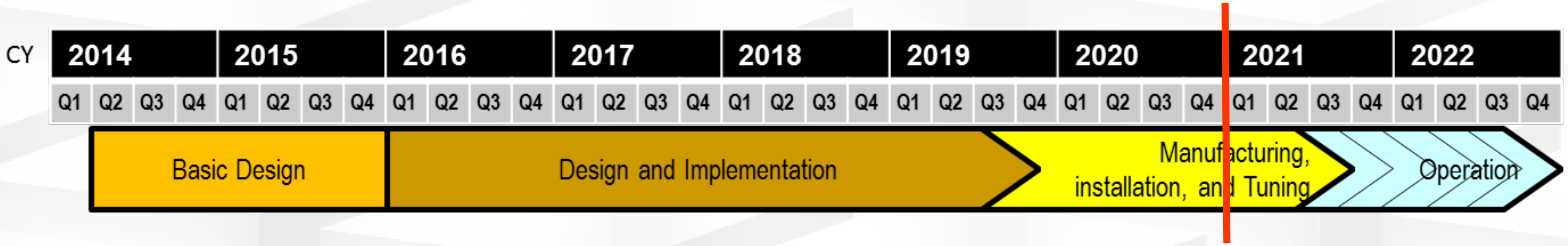
佐藤三久

フラグシップ2020プロジェクト・副プロジェクトリーダー
アーキテクチャ開発チーム・チームリーダ
理化学研究所計算科学研究センター・副センター長

RIKEN R-CCS

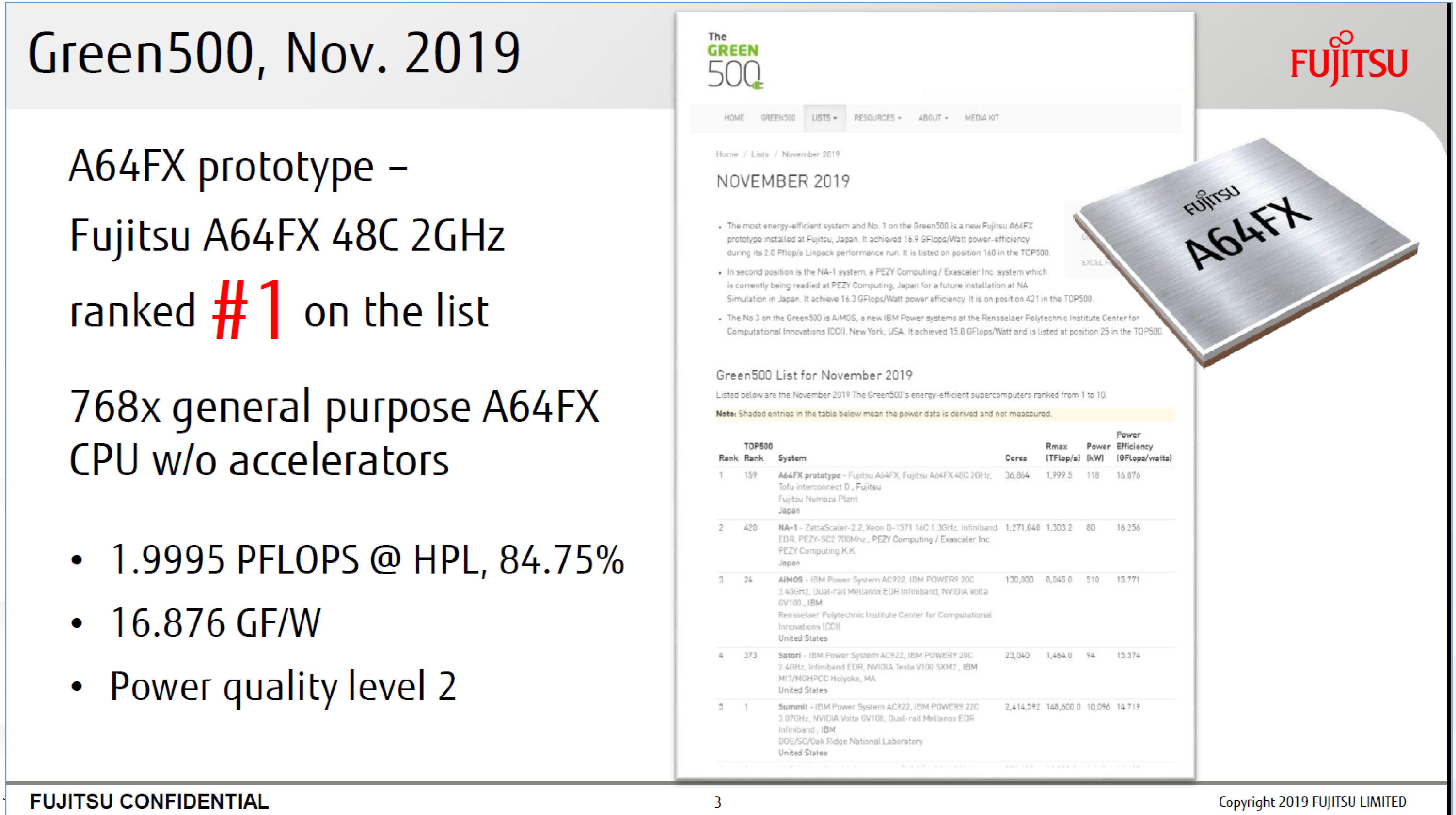
FLAGSHIP2020 Project “Fugaku”: Status and Update

- March 2019: The Name of the system was decided as “Fugaku”
- Aug. 2019: The K computer decommissioned, stopped the services and shutdown (removed from the computer room)
- Oct 2019: access to the test chips was started.
- Nov. 2019: Fujitsu announce FX1000 and FX700, and business with Cray.
- Nov 2019: Fugaku clock frequency will be 2.0GHz and boost to 2.2 GHz.
- Nov 2019: Green 500 1st position!
- Oct-Nov 2019: MEXT announced the Fugaku “early access program” to begin around Q2/CY2020
- Dec 2019: Delivery and Installation of “Fugaku” was started.
- May 2020: Delivery completed
- June 2020: 1st in Top500, HPCG, Graph 500, HPL-AI at ISC2020
- 2020: 成果創出プログラム、コロナ対策課題プロジェクト、などで一部、利用されている
- Nov 2020: 再度、4ベンチマークで、世界１位！
- 2021年に、共用サービス開始の予定



No.1 in Green500 at SC19!

Announce from Fujitsu at SC19



Green500, Nov. 2019

A64FX prototype –
Fujitsu A64FX 48C 2GHz
ranked #1 on the list

768x general purpose A64FX CPU w/o accelerators

- 1.9995 PFLOPS @ HPL, 84.75%
- 16.876 GF/W
- Power quality level 2

NOVEMBER 2019

- The most energy-efficient system and No. 1 on the Green500 is a new Fujitsu A64FX prototype installed at Fujitsu, Japan. It achieved 16.9 GFlops/Watt power-efficiency during its 2.0 Pflop/s Linpack performance run. It is listed on position 160 in the TOP500.
- In second position is the NA-1 system, a PEZY Computing / Exascaler Inc. system which is currently being readied at PEZY Computing, Japan for a future installation at NA Simulation in Japan. It achieve 16.3 GFlops/Watt power efficiency. It is on position 421 in the TOP500.
- The No 3 on the Green500 is AiMOS, a new IBM Power systems at the Rensselaer Polytechnic Institute Center for Computational Innovations (CCI), New York, USA. It achieved 15.8 GFlops/Watt and is listed at position 25 in the TOP500.

Green500 List for November 2019

Listed below are the November 2019 The Green500's energy-efficient supercomputers ranked from 1 to 10.

Note: Shaded entries in the table below mean the power data is derived and not measured.

Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	Power (kW)	Power Efficiency (GFlops/watts)
1	159	**A64FX prototype** - Fujitsu A64FX, Fujitsu A64FX 48C 2GHz, Tofu interconnect D , Fujitsu Fujitsu Numazu Plant Japan	36,864	1,999.5	118	16.876
2	420	**NA-1** - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz , PEZY Computing / Exascaler Inc. PEZY Computing K.K. Japan	1,271,040	1,303.2	80	16.256
3	24	**AiMOS** - IBM Power System AC922, IBM POWER9 20C 3.45GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Volta GV100 , IBM Rensselaer Polytechnic Institute Center for Computational Innovations (CCI) United States	130,000	8,045.0	510	15.771
4	373	**Satori** - IBM Power System AC922, IBM POWER9 20C 2.4GHz, Infiniband EDR, NVIDIA Tesla V100 SXM2 , IBM MIT/MGHPCC Holyoke, MA United States	23,040	1,464.0	94	15.574
5	1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	10,096	14.719

Fugaku won 1st position in 4 benchmarks ! (ISC2020, June 2020)

Bencmark	1st	Score	Unit	2nd	Score	1st/ 2nd
TOP500 (LINPACK)	Fugaku	415.5	PFLOPS	Summit（US）	148.6	2.80
HPCG	Fugaku	13.4	PFLOPS	Summit（US）	2.93	4.57
HPL-AI	Fugaku	1.42	EFLOPS	Summit（US）	0.55	2.58
Graph500	Fugaku	70,980	GTEPS	太湖之光 TaihuLight (China)	23,756	2.99

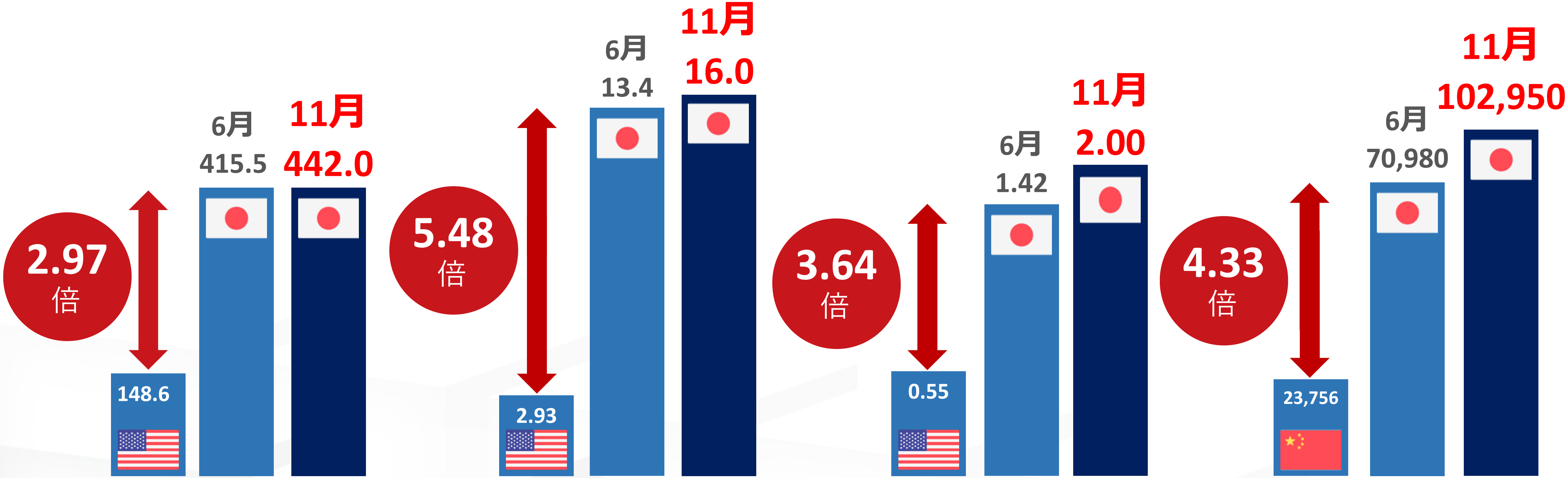
2 to 4 times faster in every benchmark!

Fugaku Keeps Crown !
(SC20, Nov 2020)

- June: partial system → Nov.: full system (#nodes, frequency 2.2GHz)
- Algorithmic efficiency improvements

	Measured	Peek Perf	Efficiency	(June 2020)	Perf of 2nd System	1st/ 2nd
LINPACK	442.01 PF	537.21 PF	82.3%	(415.5 PF)	148.60 PF	3.0
HPCG	16.00 PF	537.21 PF	3.0%	(13.4PF)	2.92 PF	5.5
HPL-AI	2.00 EF	2.14 EF	93.2%	(1.42EF)	0.55 EF	3.6
Graph500	102.95 Teps			(70.98)	23.75 Teps	4.3

「富岳」全系によるベンチマークテスト結果（6月の結果との比較）



TOP500
（単位：PFLOPS）
浮動小数点の演算
での性能評価

HPCG
（単位：PFLOPS）
実際にアプリケーションを
稼働させた性能評価

HPL-AI
（単位：EFLOPS）
AI処理での
性能評価

Graph500
（単位：GTEPS）
ビッグデータ処理
での性能評価

2位に対して、3倍から5.5倍近い性能差を実現

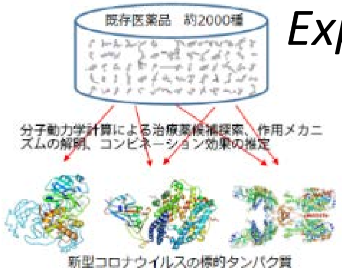
2020年6月22日発表
International Supercomputing Conference (ISC 2020)　スーパーコンピュータ世界ランキング

MEXT Fugaku Program: Fight Against COVID19

Fugaku resources made available a year ahead of general production (more research topics under international solicitation)

A partner of international COVID-19 HPC Consortium

Medical-Pharma


既存医薬品　約2000種
分子動力学計算による治療薬候補探索、作用メカニズムの解明、コンビネーション効果の推定
新型コロナウイルスの標的タンパク質

Exploring new drug candidates for COVID-19

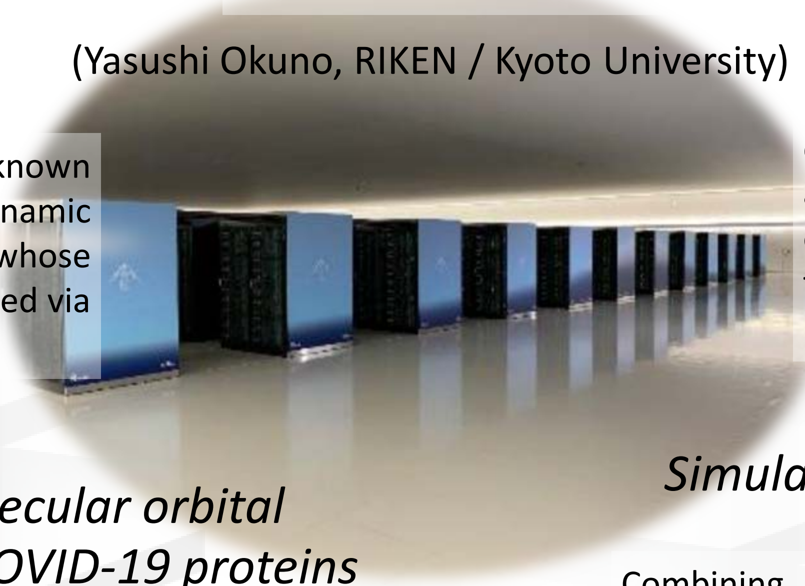
Large-scale MD to search & identify therapeutic drug candidates showing high affinity for COVID-19 target proteins from 2000 existing drugs

(Yasushi Okuno, RIKEN / Kyoto University)

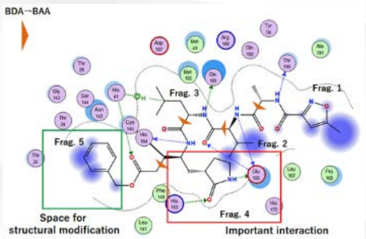
Prediction of conformational dynamics of proteins on the surface of SARS-Cov-2

GENESIS MD to interpolate unknown experimentally undetectable dynamic behavior of spike proteins, whose static behavior has been identified via Cryo-EM

(Yuji Sugita, RIKEN)



Fragment molecular orbital calculations for COVID-19 proteins



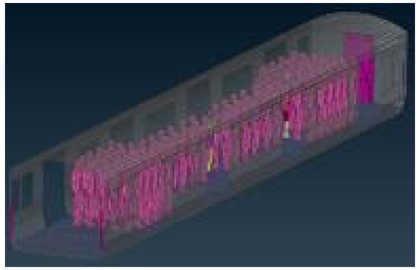
Large-scale, detailed interaction analysis of COVID-19 using Fragment Molecular Orbital (FMO) calculations using ABINIT-MP

(Yuji Mochizuki, Rikkyo University)

Societal-Epidemiology

Prediction and Countermeasure for Virus Droplet Infection under the Indoor Environment

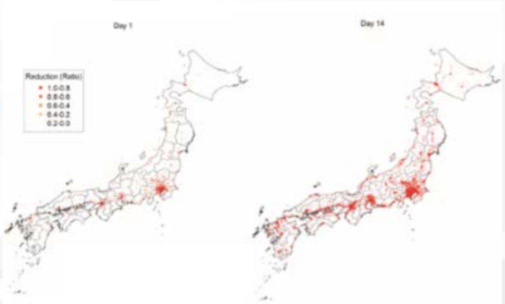
Massive parallel simulation of droplet scattering with airflow and hat transfer under indoor environment such as commuter trains, offices, classrooms, and hospital rooms



(Makoto Tsubokura, RIKEN / Kobe University)

Simulation analysis of pandemic phenomena

Combining simulations & analytics of disease propagation w/contact tracing apps, economic effects of lockdown, and reflections social media, for effective mitigation policies



(Nobuyasu Ito, RIKEN)

Outline of my talk

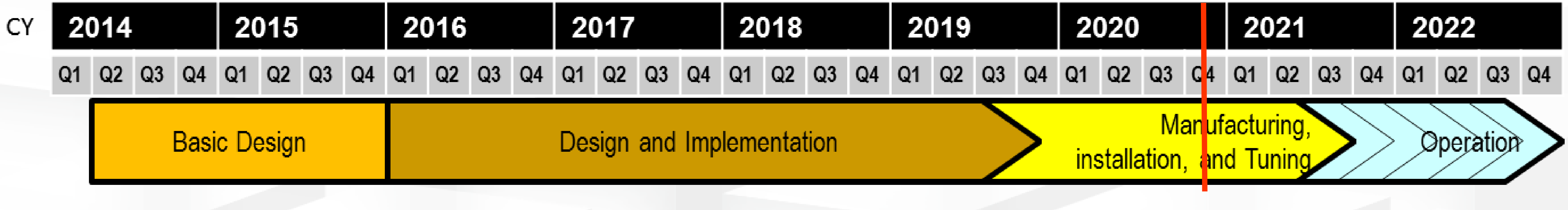
- **Overview of FLAGSHIP 2020 project**
- **Co-Design Methodology**
- **Co-Design of Post-K**
- **The supercomputer "Fugaku"**
  - Performance of "Fugaku"
- **Co-Design of Compiler and Applications**
- **System software**
- **Conclusion with some retrospective comments**

SC20 technical paper. "Co-Design for A64FX Manycore Processor and "Fugaku""

M. Sato, Y. Ishikawa, H. Tomita, Y. Kodama, T. Odajima, M. Tsuji, H. Yashiro, M. Aoki, N. Shida, I. Miyoshi,K. Hirai, A. Furuya, A. Asato, K. Morita, T. Shimizu

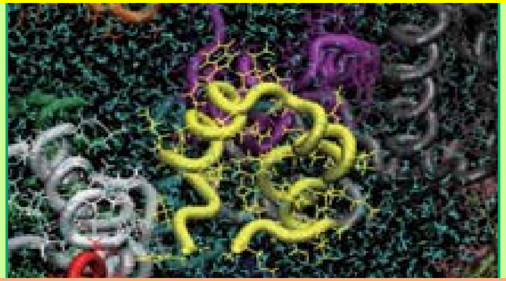
FLAGSHIP2020 Project: Mission and Timeline

- **Missions**
  - Building the Japanese national flagship supercomputer “Fugaku “(a.k.a post K), and
  - Developing wide range of HPC applications, running on Fugaku, in order to solve social and science issues in our country and all over the world
- **Organization**
  - The RIKEN Center for Computational Science in charge of the research and development of the Post-K.
  - Fujitsu is a vendor partner
- **Schedule : Started from 2014**



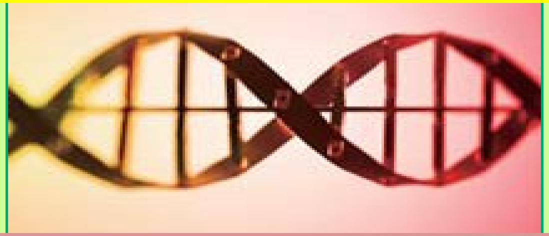
Target science: 9 Priority Issues

①Innovative Drug Discovery



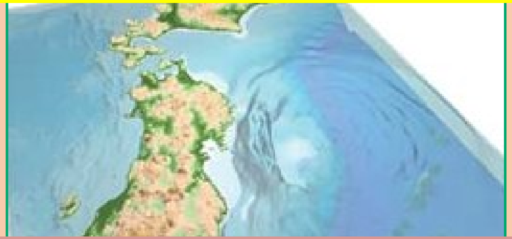
RIKEN Quant. Biology Center

②Personalized and Preventive Medicine



Inst. Medical Science, U. Tokyo

③Hazard and Disaster induced by Earthquake and Tsunami



Earthquake Res. Inst., U. Tokyo

⑧ Innovative Design and Production Processes for the Manufacturing Industry in the Near Future

Cent. for Earth Info., JAMSTEC

⑨Fundamental Laws and Evolution of the Universe

Cent. for Comp. Science, U. Tsukuba

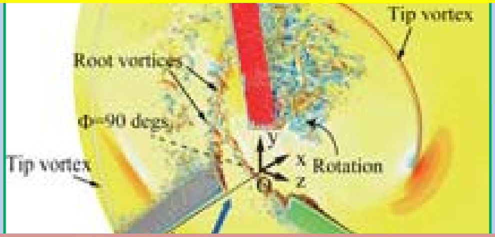
④Environmental Predictions with Observational Big Data

Center for Earth Info., JAMSTEC

⑦New Functional Devices and High-Performance

Inst. For Solid State Phys., U. Tokyo

⑥Innovative Clean Energy Systems



Grad. Sch. Engineering, U. Tokyo

⑤High-Efficiency Energy Creation, Conversion/Storage and Use

Inst. Molecular Science, NINS

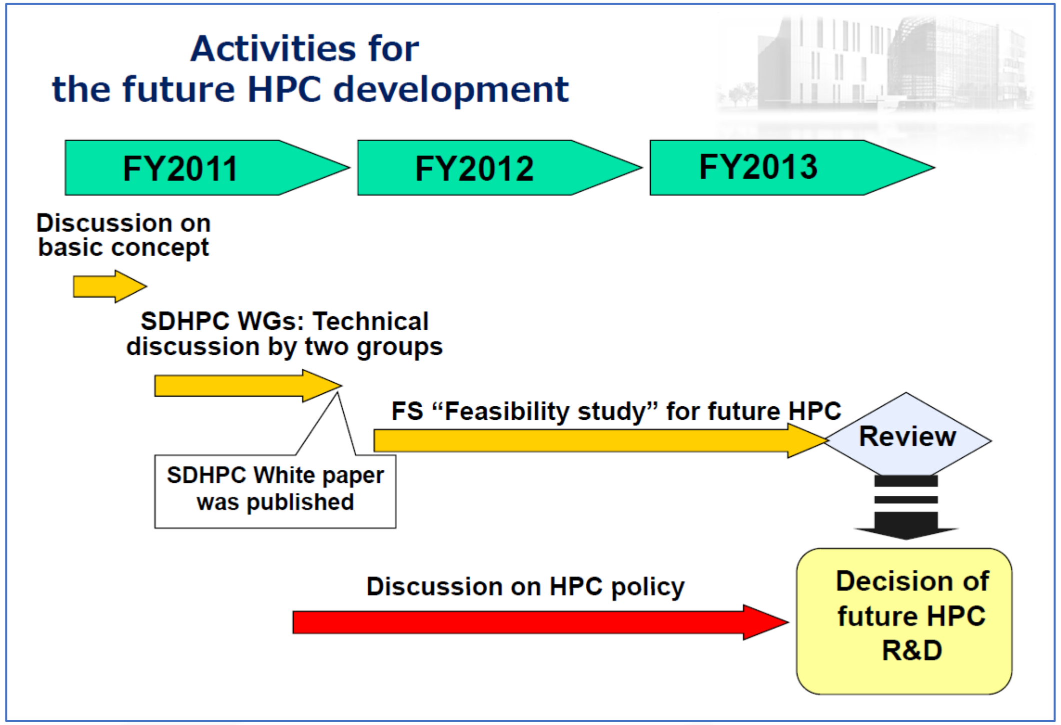
KPIs on Fugaku development in FLAGSHIP 2020 project

3 KPIs (key performance indicator) were defined for Fugaku development

- **1. Extreme Power-Efficient System**
  - Maximum performance under Power consumption of 30 - 40MW (for system)
- **2. Effective performance of target applications**
  - It is expected to exceed 100 times higher than the K computer’s performance in some applications
- **3. Ease-of-use system for wide-range of users**

Before starting FLAGSHIP2020 Project

- **The K computer project (2006-2012)**
  - The public service of the K computer started from 2012.
- **Feasibility study project (2012-2013)**
  - Application study team leaded by RIKEN AICS (Tomita)
  - System study team leaded by U Tokyo (Ishikawa)
    - Next-generation “General-Purpose” Supercomputer
  - System study team leaded by U Tsukuba (Sato)
    - Exascale heterogeneous systems with accelerators
  - System study team leaded by Tohoku U (Kobayashi)
- **Initial Plan at the beginning (2014) was a combined system with “General-Purpose” Supercomputer and “accelerators”**
  - But, the development of “accelerators” was canceled due to budget problem.



Activities for the future HPC development

FY2011 → FY2012 → FY2013

Discussion on basic concept

SDHPC WGs: Technical discussion by two groups

SDHPC White paper was published

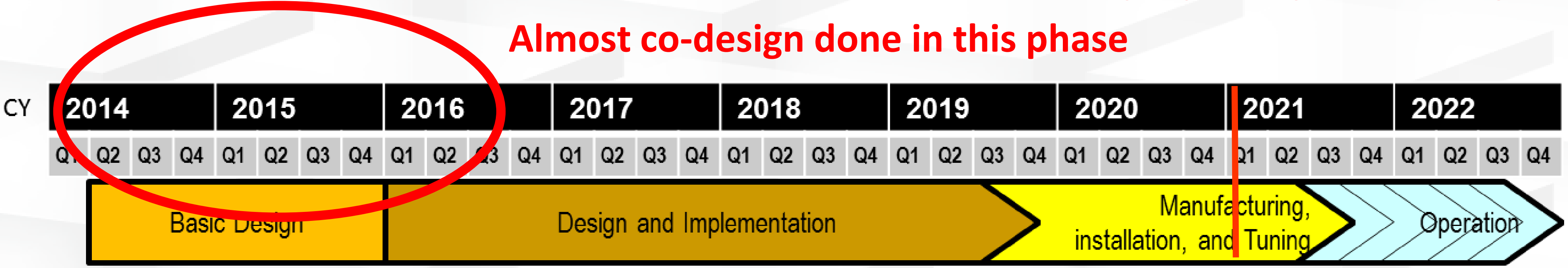
FS “Feasibility study” for future HPC

Review

Discussion on HPC policy

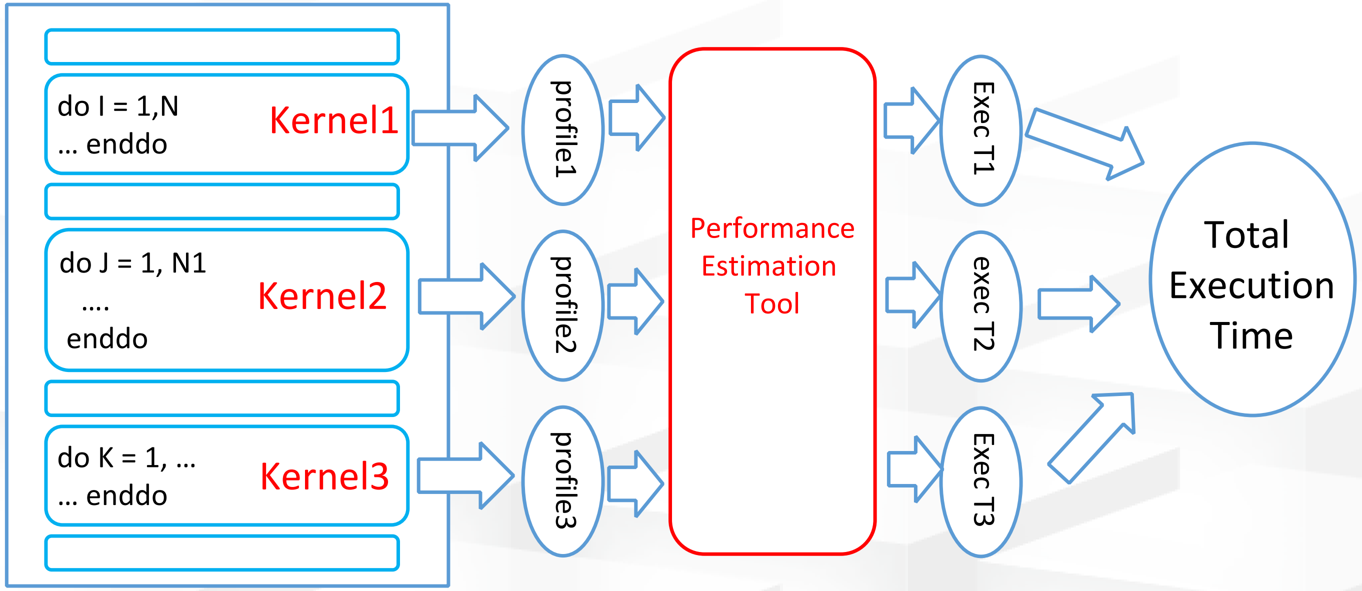
Decision of future HPC R&D

The Basic design follows “general purpose” processor by FS of U. Tokyo



Almost co-design done in this phase

Co-Design Methodology



do I = 1,N
… enddo
Kernel1

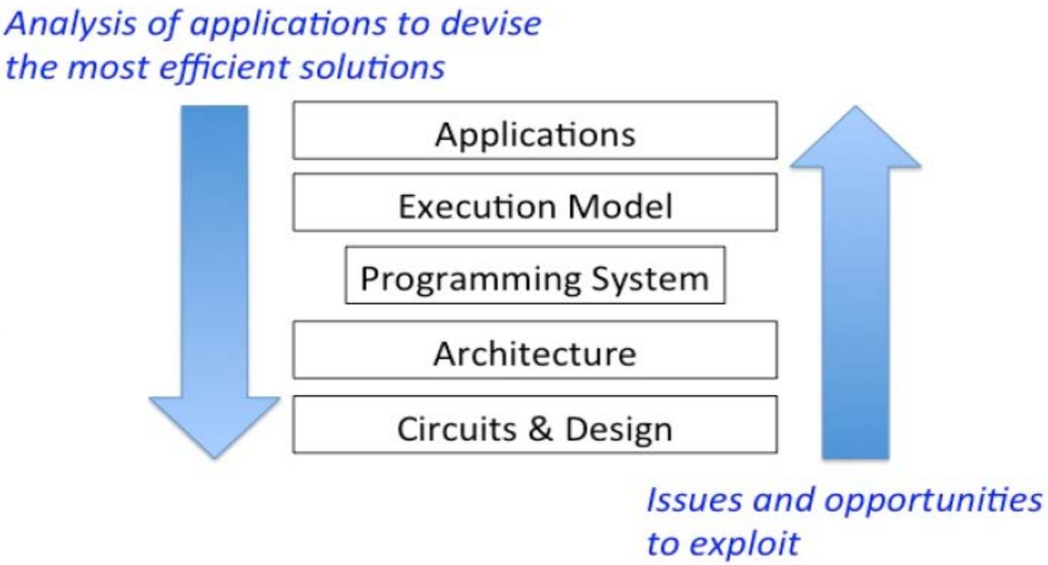
do J = 1, N1
….
enddo
Kernel2

do K = 1, …
… enddo
Kernel3

Co-Design in HPC for exascale computing

- **As a key strategy for exascale computing, co-design has received considerable attention in the HPC community for several years**
  - In modern very high-end parallel system, more performance can be delivered (even upto “exascale”) by increasing the number of nodes
  - need to design the system by trade-off between “energy/power” and “cost” and performance
  - need to design the system by taking characteristics of applications in account

- **The co-design of HPC must optimize and maximize the benefits to cover many applications as possible.**
  - different from "co-design" in embedded systems. For example, in embedded field, co-design sometimes includes "specialization" for a particular application.



Analysis of applications to devise the most efficient solutions

Applications

Execution Model

Programming System

Architecture

Circuits & Design

Issues and opportunities to exploit

- **Richard F. BARRETT, et.al. “On the Role of Co-design in High Performance Computing”, *Transition of HPC Towards Exascale Computing***

Target Applications for co-design

- **Target applications provided by each application project (“9 priority issues”)**

Applications	Brief Description	Computational Method	Target Perf. Relative to K	Codesign points										
				Structured Mesh	Unstructured Mesh	Particle Comp	Integer comp.	Dense Matrix (single node)	Dense Matrix (multi-nodes)	Comm (low latency)	Comm (high bandwidth)	Neighbor Comm.	Global Comm.	Large Data I/O
GENESIS[5]	Analysis for proteins	Molecular Dynamics	125			✓						✓	✓	
Genomon[6]	Genome processing	Large data processing for Genome alignment	8				✓							✓
GAMERA[7]	Earthquake simulation	FEM on unstructured & structured grid	45		✓					✓	✓		✓	
NICAM[8] +LETKF[9]	Weather prediction system with Data assimilation	FVM, stencil on using structured grid & ensemble Kalman filter	120	✓				✓			✓	✓		✓
NTChem[10]	Molecular electronic structure calculation	High-precision MO method, sparse and dense matrix computation.	40					✓	✓		✓	✓	✓	
FFB [12]	Large Eddy Simulation	FEM on unstructured grid	35		✓							✓	✓	
RSDFT [11]	An ab-initio simulation with DFT on real space	density functional theory, dense matrix computation	30	✓				✓			✓	✓	✓	
ADVENTURE[13]	Computational mechanics simulation	FEM on unstructured grid	25		✓			✓	✓	✓		✓	✓	
LQCD [14]	Lattice QCD simulation	structured grid Monte Carlo	25	✓						✓	✓	✓	✓	

Target Applications for co-design

Performance Targets

- Perf. Targets are defined relative to the K
- 30 to 40 MW power consumption
- Some apps: 100 times faster than K for some applications (tuning included)

Co-design points

- Type of apps: structured mesh, unstructured mesh, particle …
- Major algorithms: dense Matrix, sparse ..
- Comm. Pattern, I/O

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

Applications	Brief Description	Computational Method	Target Perf. Relative to K	Codesign points										
				Structured Mesh	Unstructured Mesh	Particle Comp	Integer comp.	Dense Matrix (single node)	Dense Matrix (multi-nodes)	Comm (low latency)	Comm (high bandwidth)	Neighbor Comm.	Global Comm.	Large Data I/O
GENESIS[5]	Analysis for proteins	Molecular Dynamics	125			✓						✓	✓	
Genomon[6]	Genome processing	Large data processing for Genome alignment	8				✓							✓
GAMERA[7]	Earthquake simulation	FEM on unstructured & structured grid	45		✓					✓	✓		✓	
NICAM[8] +LETKF[9]	Weather prediction system with Data assimilation	FVM, stencil on using structured grid & ensemble Kalman filter	120	✓				✓			✓	✓		✓
NTChem[10]	Molecular electronic structure calculation	High-precision MO method, sparse and dense matrix computation.	40					✓	✓		✓	✓	✓	
FFB [12]	Large Eddy Simulation	FEM on unstructured grid	35		✓							✓	✓	
RSDFT [11]	An ab-initio simulation with DFT on real space	density functional theory, dense matrix computation	30	✓				✓			✓	✓	✓	
ADVENTURE[13]	Computational mechanics simulation	FEM on unstructured grid	25		✓			✓	✓	✓		✓	✓	
LQCD [14]	Lattice QCD simulation	structured grid Monte Carlo	25	✓						✓	✓	✓	✓	

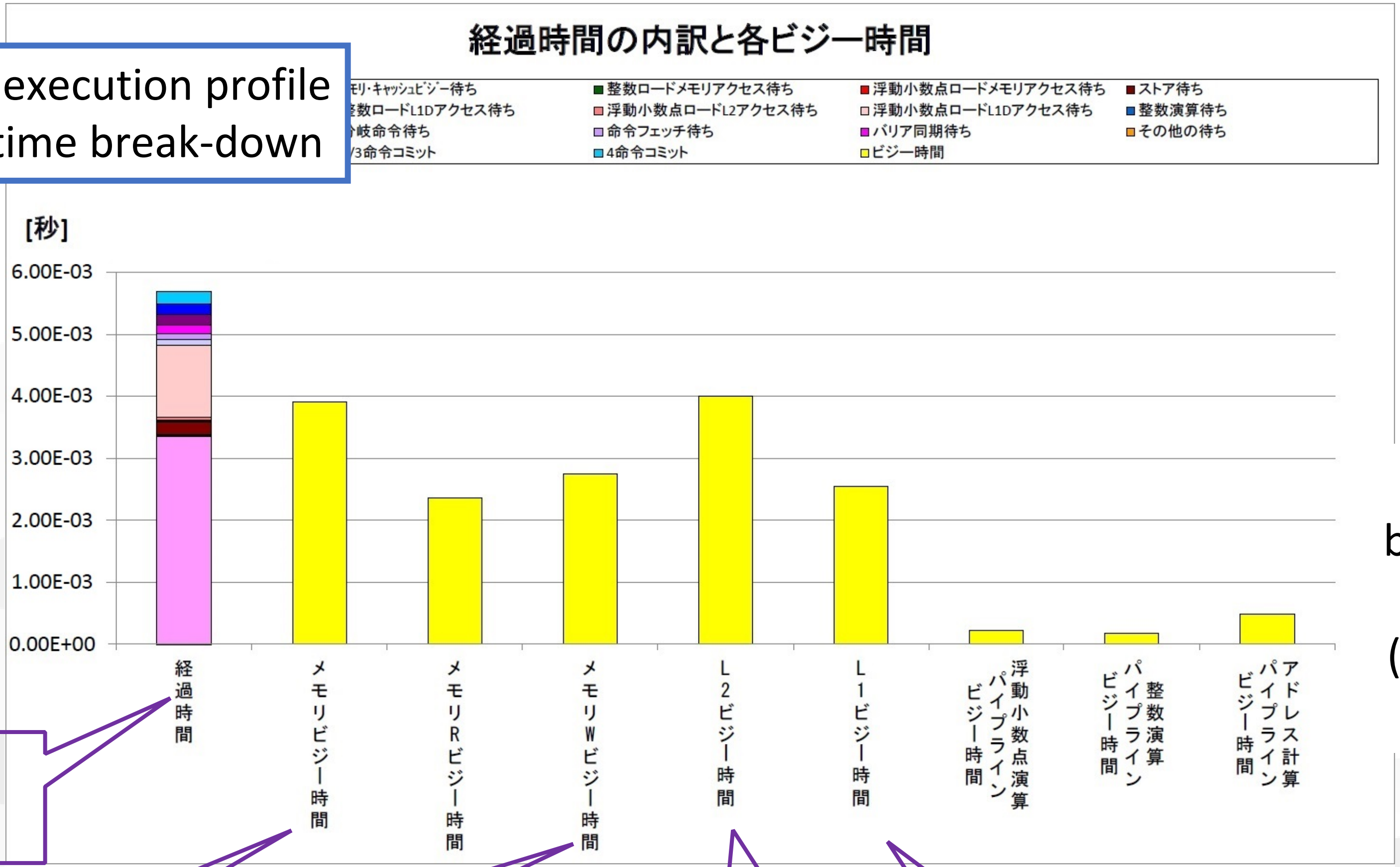
Tools for co-design

- **Performance estimation tool**
  - Enables performance projection using Fujitsu FX100 execution profile to a set of arch. parameters.
  - Modeled by Fujitsu micro-architecture
- **Fujitsu in-house simulators**
  - Extended FX100 (SPARC) simulator and compiler for preliminary studies
  - Armv8+SVE simulator and compiler
- **Hardware Emulator**
  - Used for logic design verification accurate evaluation of performance and power consumption.
- **Gem5 simulator for the Post-K**
  - It has been developed after co-design for architecture verification and performance tuning by RIKEN

Performance estimation tool

- **Estimate the performance of multithreaded programs on “new architecture” (post-K) from the profile data taken on Fujitsu FX100**
- **Estimate a maximum performance when busy-time of each component is hidden by others, based on Fujitsu’s micro-architecture.**

Fujitsu FX100 execution profile And execute-time break-down



Break-down exec time

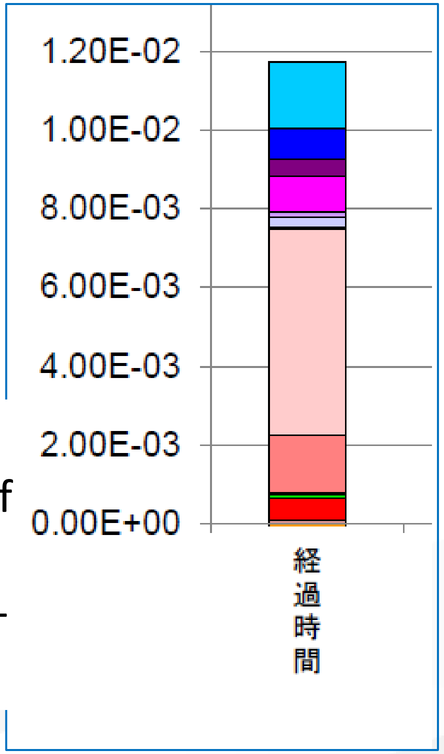
Memory acc busy time

Memory W acc busy time

L2 busy time

L1 busy time

Projection by throughput of each element (similar to “roof-line mode”)



Estimated execute-time break-down on new architecture by projection

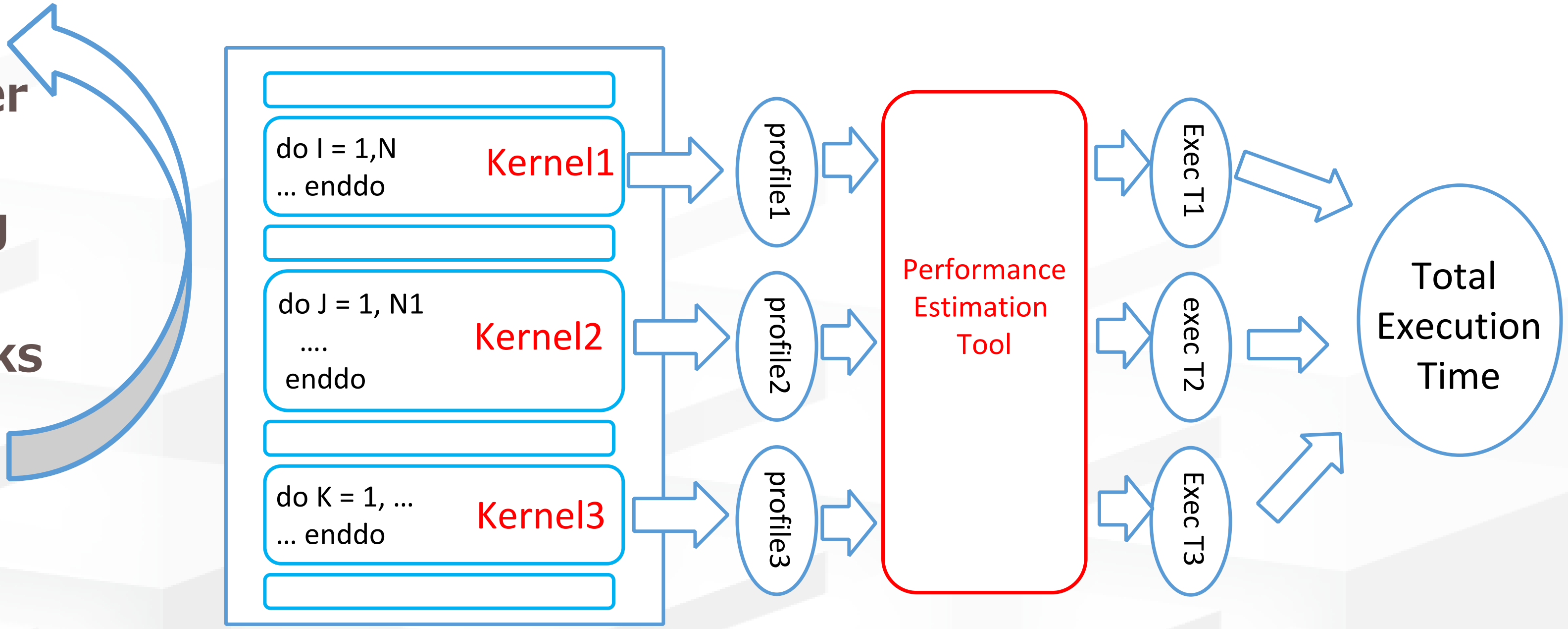
Tools for co-design

- **Performance estimation tool**
  - Enables performance projection using Fujitsu FX100 execution profile to a set of arch. parameters.
  - Modeled by Fujitsu micro-architecture
- **Fujitsu in-house simulators**
  - Extended FX100 (SPARC) simulator and compiler for preliminary studies
  - Armv8+SVE simulator and compiler
- **Hardware Emulator**
  - Used for logic design verification accurate evaluation of performance and power consumption.
- **Gem5 simulator for the Post-K**
  - It has been developed after co-design for architecture verification and performance tuning by RIKEN

Co-design Methodology

- Our performance estimation tool estimates performance according to "throughput" of each block in core, according to Fujitsu micro-architecture.
- Identify kernels from each application benchmark, and breakdown to a set of kernels to estimate execution time of each kernel.
- Co-design process for each kernels

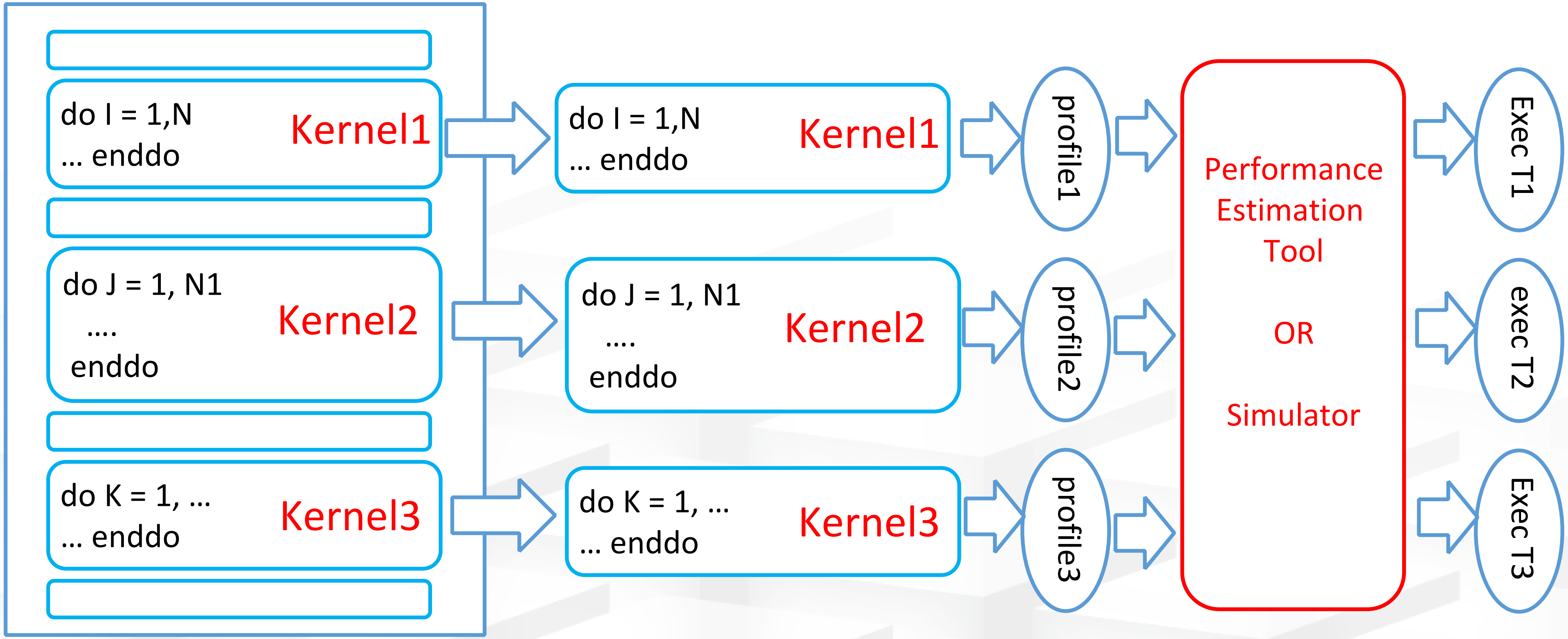
1. Setting a set of system parameters
2. Tuning target applications under the system parameters
3. Evaluating execution time using "estimation tool"
4. Identifying hardware bottlenecks and changing the set of system parameters



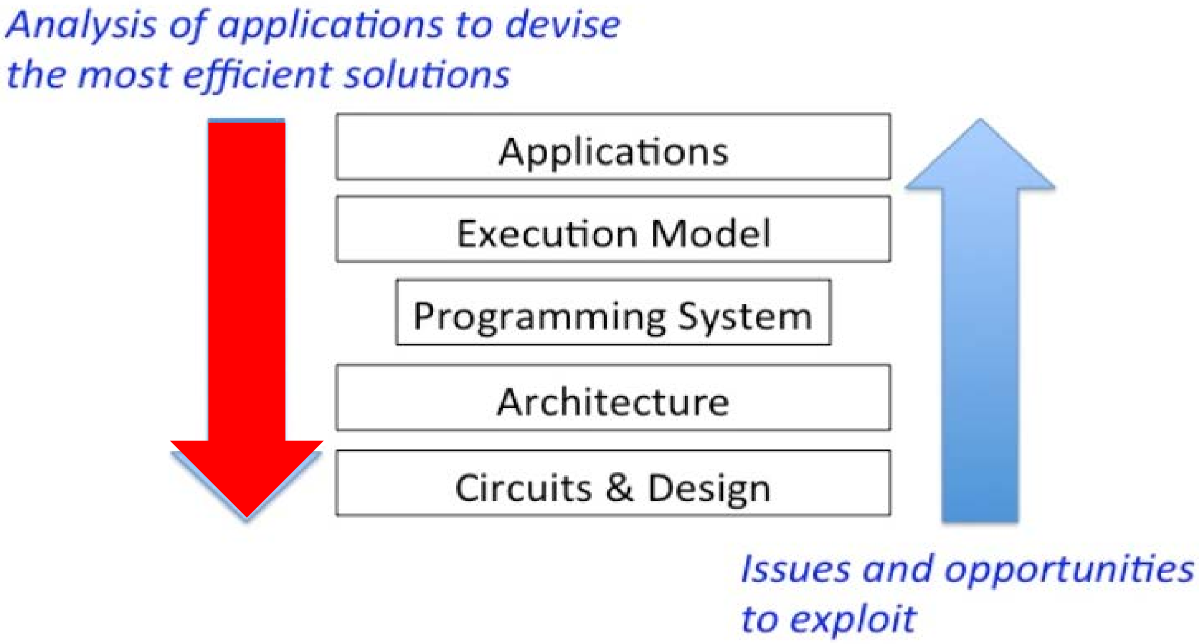
- For communication, we use analytical models for communication pattern.

Extract Kernels

- Extract kernels from each application benchmark for more detail analysis.
- This set of kernels is useful to evaluate performance on the simulator, which takes long time to execute.



Co-Design of Post-K



Analysis of applications to devise the most efficient solutions
Applications
Execution Model
Programming System
Architecture
Circuits & Design
Issues and opportunities to exploit

- **Richard F. BARRETT, et.al. "On the Role of Co-design in High Performance Computing", *Transition of HPC Towards Exascale Computing***

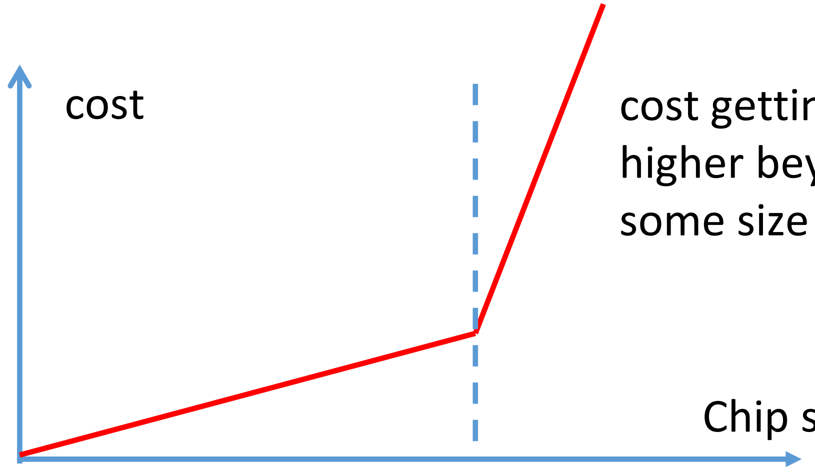
Technologies and Architectural Parameters to be determined

- **Basic Architecture Design (by Feasibility Studies)**
  - Manycore approach, O3 cores, some parameters on chip configuration and SIMD
- **Instruction Set Architecture and SIMD Instructions**
  - Fujitsu collaborated with Arm, contributing to the design of the SVE as a lead partner
- **Chip configuration**
- **Memory technology**
  - DDR, HBM, HMC …
- **Cache structure**
- **Out of order (O3) resources**
- **Enhancement for Target Applications**
- **Interconnect between Nodes**
  - SerDes, topologies “Tofu” or other network?

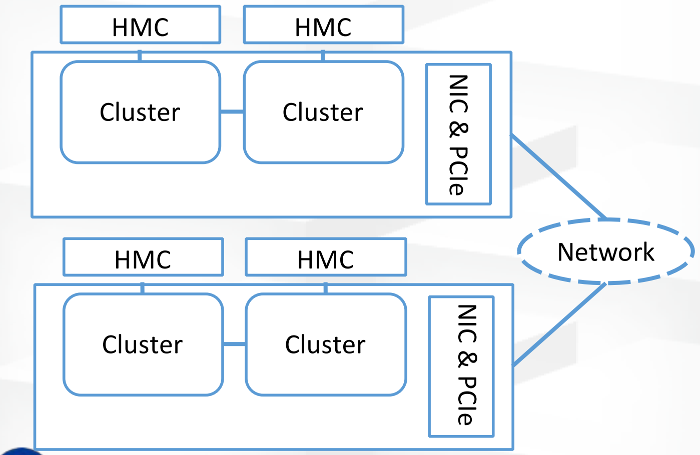
- ✓ The number of cores in a CMG
- ✓ The number of CMGs in a chip
- ✓ How to connect cores to shared L2 in a CMG
- ✓ The number of ways, the size, and throughputs of the L1
- ✓ and L2 caches
- ✓ The topology of network-on-chip to connect CMGs
- ✓ The die size of the chip
- ✓ The number of chips in a node

Chip configuration (1/2)

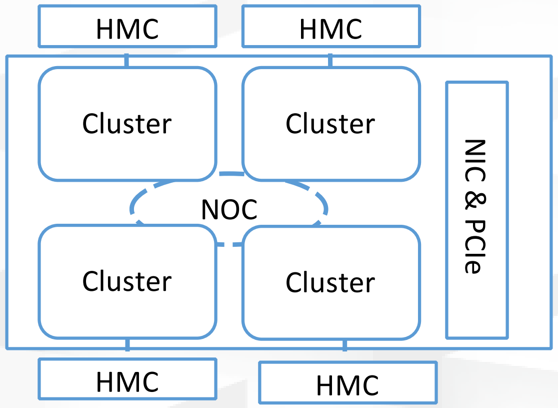
- **Basic configuration was given by Feasibility Study**
- **Parameters to be determined**
  - #CMG(cluster), #core/CMG, Network on Chip (NOC)
  - MCM & interposer (organic, silicon, glass …)
- **Chip size is a major factor of cost.**
- **Performance was estimated by "analytical models" for simple kernel such as HPL**



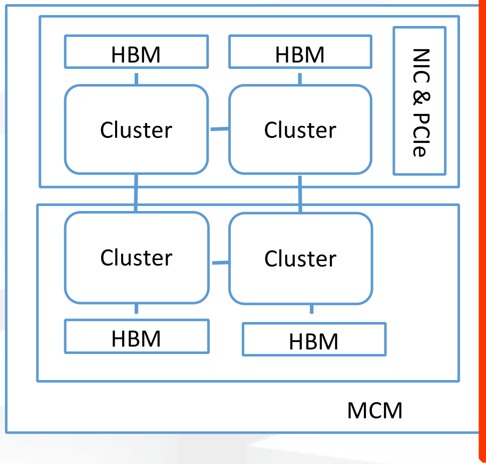
Small chip, HMC



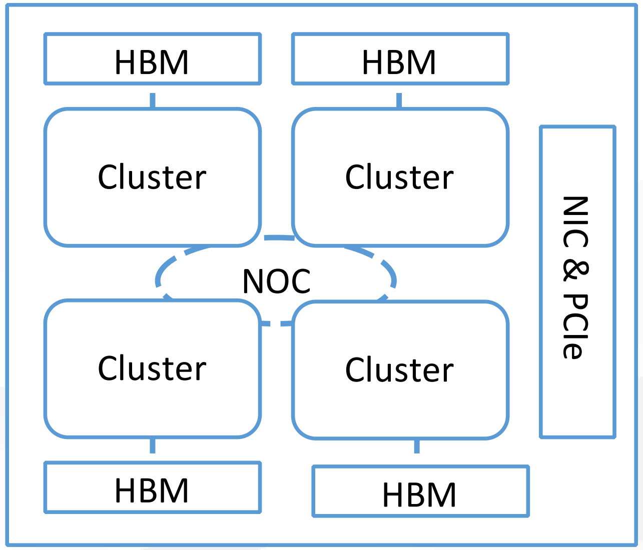
Large chip, HMC



Small chip + HMC with MCM



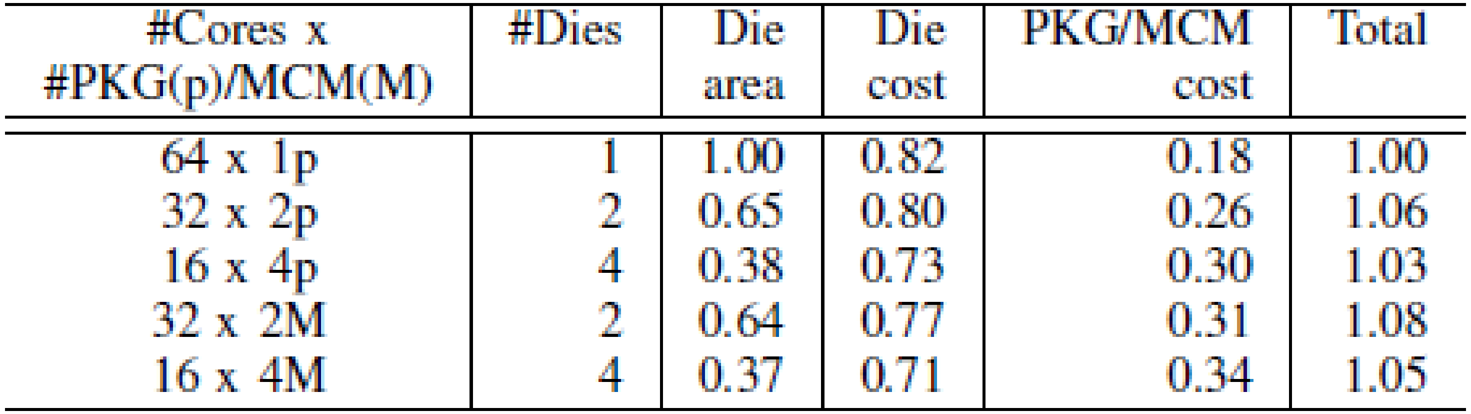
Large chip + HBM
(selected as A64FX)



Chip configuration (2/2)

- **Our decision was to use a single large die containing some CMGs and the network interface for interconnect and PCIe for I/O connected by a network-on-chip.**
  - The size of the die was about 400mm^2, reasonable in terms of cost for 7-nm FinFET technology.
- **Why not MCM ("chiplets" approach as in recent AMD chips)**
  - A small chip can be relatively cheap with a good yield.
  - High cost of packaging
  - May need different kinds of chips for CPU and IO, network, … resulting in high cost.
  - The connection between chips on the MCM would also increase the power consumption

Configuration and Cost, comparing to MCM, multi-PKG



#Cores x #PKG(p)/MCM(M)	#Dies	Die area	Die cost	PKG/MCM cost	Total
64 x 1p	1	1.00	0.82	0.18	1.00
32 x 2p	2	0.65	0.80	0.26	1.06
16 x 4p	4	0.38	0.73	0.30	1.03
32 x 2M	2	0.64	0.77	0.31	1.08
16 x 4M	4	0.37	0.71	0.34	1.05

Note: this estimation was done for 64 cores

Memory Technologies

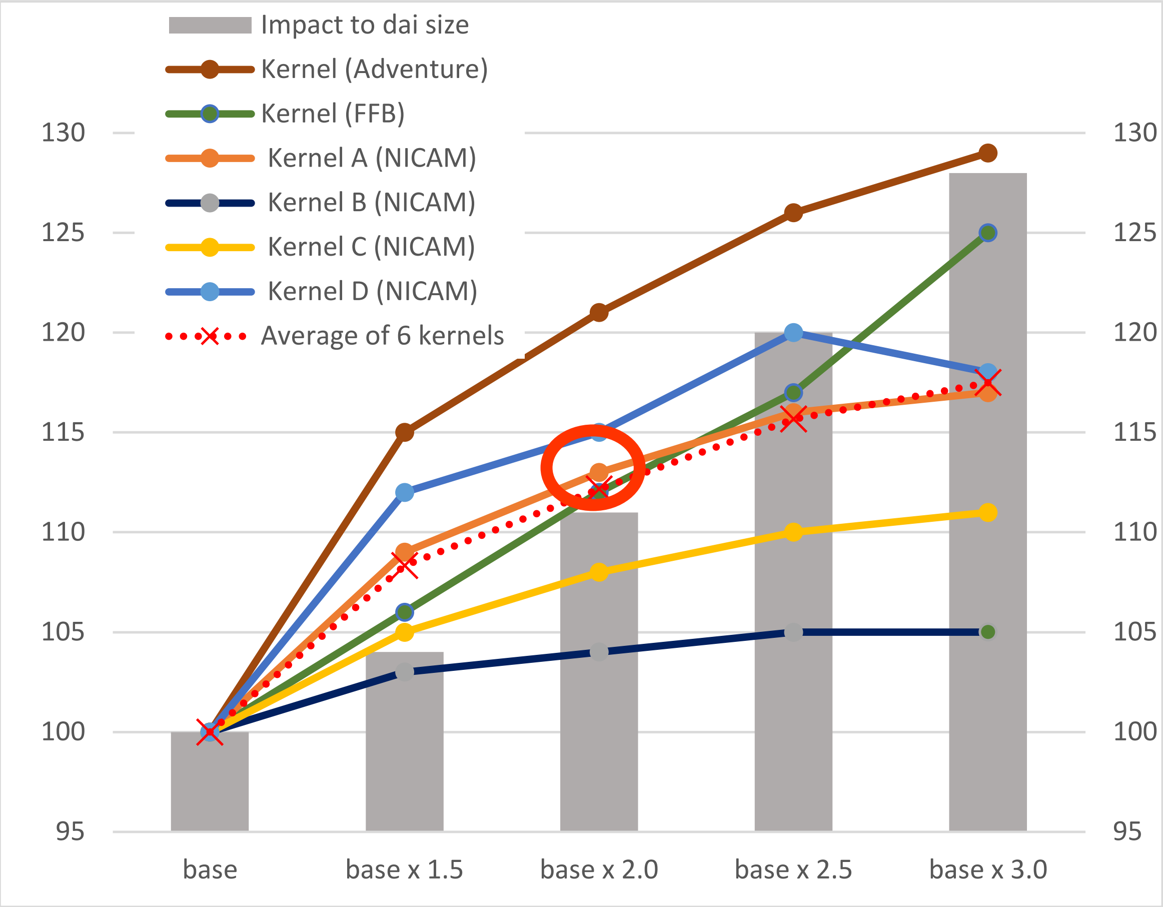
- **As a memory technology available around 2019, HBM2 was chosen for its power efficiency and high memory bandwidth for both read and write.**
  - We decided not to use any additional DDR memory to reduce the cost.
  - The size of memory is small having only HBM2, but the most target apps are scalable.

Memory Technology	decision	Pros & Cons
DDR4	✕	× Memory bandwidth is too low for modern HPC core. ○ Low cost and possibly large capacity
HMC (Hybrid Memory Qube)	△	○ High bandwidth by multiple serial links △ Asymmetric read/write. × Power consumption to drive the serial links is large
HBM (High Bandwidth memory)	○	◎ High bandwidth read/write per module. (256GB/s for HBM2) × Capacity is small (up to 8 GiB for HBM2) × Cost is high because the silicon interposer is required.

Out-of-Order (O3) Resources

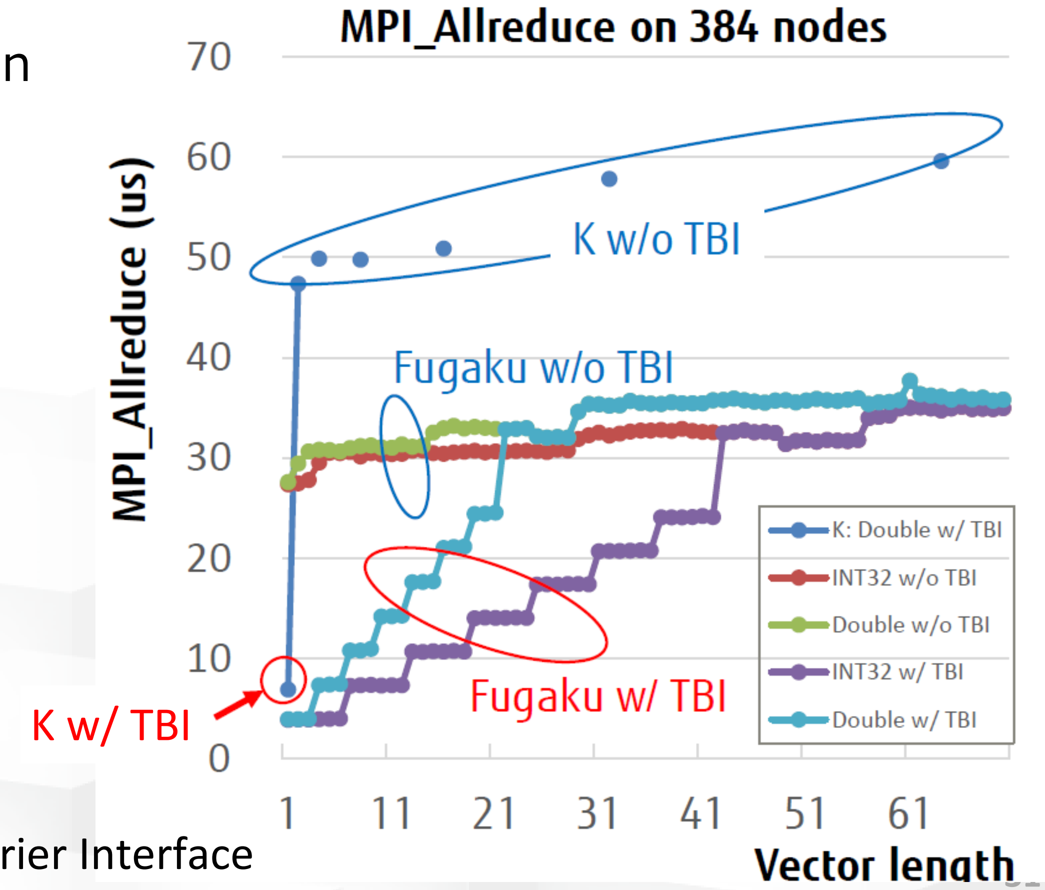
- **How to design:**
  - First, decide the ratio of the O3 resources by examining several combinations of these parameters
  - Keeping the decided ratio, the amount of the O3 resources was decided by the trade-off between the performance and the impact to the die size.
  - **We choose "Base x 2.0"**
- **The most difficult problems because:**
  - At the basic design phase, the compiler optimization was immature
  - Inherit the micro architecture of previous processor (SPARC)

Base Set of O3 paramters:
#RS(entries in reservation station)=40,
#ROB(Re-Order Buffers)=64,
Renaming regs #FP=48, #GP=32,
#Fetch Port=20, #Store Port=12, #Write Buffer=4



Co-design of network

- **We have selected "Tofu" network for performance compatibility in large-scale applications.**
- **Select Link-speed 28Gbps x 4 due to technology availability around 2019**
- **Communication patterns ware extracted, and the communication performance was estimated by "analytical model".**
  - Many target applications have neighbor communication pattern, or communication to near nodes. ⇒ "Tofu" and 28Gbps link were sufficient.
  - Some apps have all-to-all communication.
    - We studied the benefits or feasibility of additional "dedicated" all-to-all network, but it ware not selected due to cost
    - Support 3 DP reduction by Tofu TBI for QCD apps
- **6 TNIs(Tofu Network Interface: RDMA engine)**
  - Increase injection bandwidth (the K had 4 TNI)
  - Integrated in one chip.

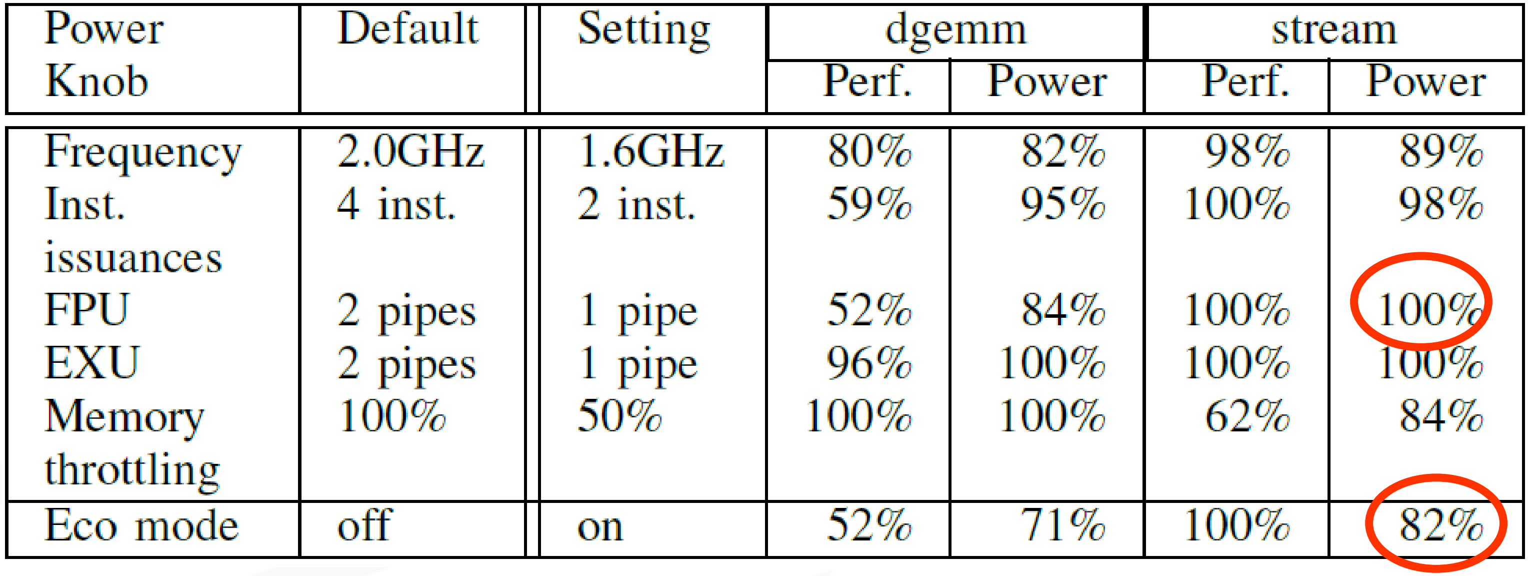


TBI: Tofu Barrier Interface

Co-Design for Low Power

- **One of the most important challenges for an exascale system is to reduce the power consumption.**
- **A64FX provides power management function called “Power Knob”**

Evaluation of power knob for dgemm and stream



Power Knob	Default	Setting	dgemm		stream	
			Perf.	Power	Perf.	Power
Frequency	2.0GHz	1.6GHz	80%	82%	98%	89%
Inst. issuances	4 inst.	2 inst.	59%	95%	100%	98%
FPU	2 pipes	1 pipe	52%	84%	100%	100%
EXU	2 pipes	1 pipe	96%	100%	100%	100%
Memory throttling	100%	50%	100%	100%	62%	84%
Eco mode	off	on	52%	71%	100%	82%

  - FL pipeline usage: FLA only, EX pipeline usage : EXA only, Frequency reduction …
  - User program can change “Power Knob” by Sandia Power-API
  - “Energy monitor” facility enables chip-level power monitoring and detailed power analysis of applications
- **“Eco-mode” : FLA only with lower “stand-by” power for ALUs**
  - “Stand-by” power for stable operation for arithmetic unit.
  - Reduce the power-consumption for memory intensive apps.

Eco mode and Boost mode for target apps

- **Boost mode: running at 2.2 GHz (normal: 2.0GHz)**
  - The eco mode can also be active in the boost mode
- **4 modes of ‘boost’ (B), ‘boost+eco’ (B+E), ‘normal’ (N), and ‘normal+eco’ (N+E) can be selected.**

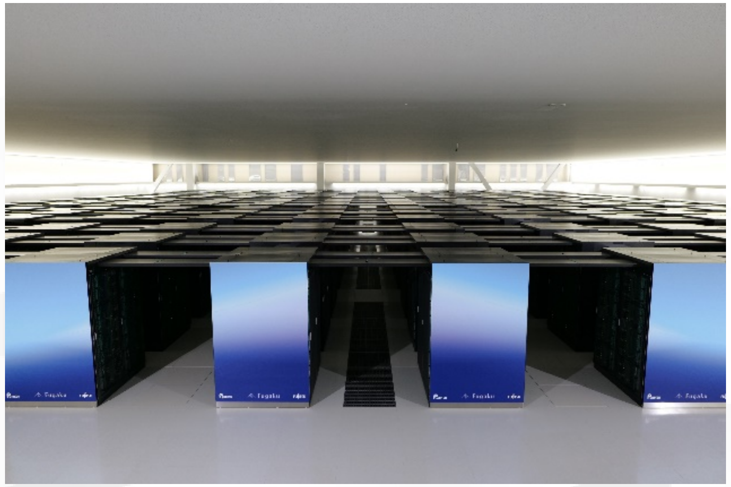
Power mode and perf. and power For Target Apps under 30MW-40MW,

Performance and Power is relative to Normal mode(N)

Application	Pow. mode	B		N+E		B+E	
		perf.	pow.	perf.	pow.	perf.	pow.
GENESIS	B	1.09	1.20	1.00	0.80	1.09	0.96
Genomon	B	1.10	1.17	-	-	-	-
GAMERA	B+E	1.06	(1.14)	1.00	0.81	1.06	0.89
NICAM+LETKF	B+E	1.07	(1.18)	0.97	0.79	1.04	0.91
NTChem	B	1.08	1.21	0.57	0.69	0.62	0.83
ADVENTURE	N	1.07	(1.21)	0.90	0.85	0.98	1.00
RSDFT	B	1.06	1.20	0.71	0.80	0.77	0.90
FFB	B+E	1.10	(1.17)	1.00	0.80	1.10	0.94
LQCD	B+E	1.05	1.17	1.00	0.74	1.05	0.83

- **4 apps out of 9 target apps choose the ‘boost eco mode’ for maximum performance under the limitation of power-budget since the one FLU SIMD pipe in “eco” mode is sufficient. (at the time of power estimation)**

The supercomputer “Fugaku”

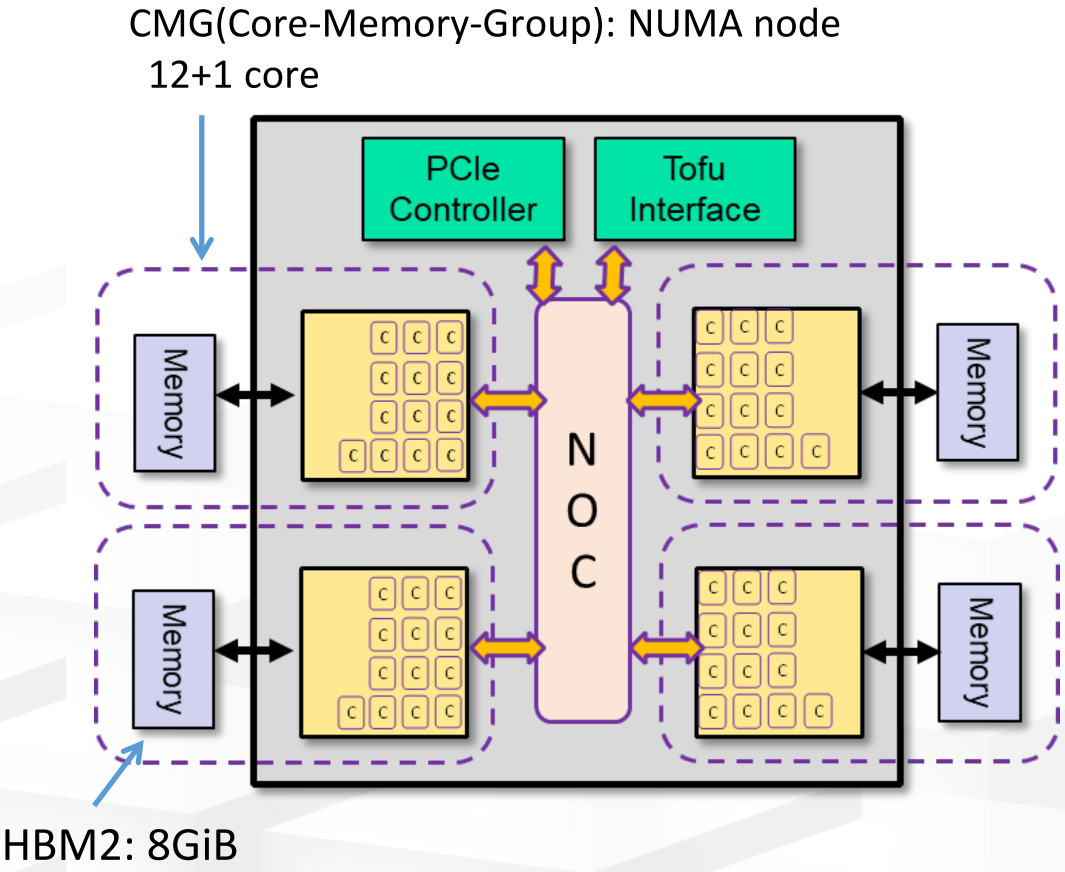


CPU Architecture: A64FX

- **Armv8.2-A (AArch64 only) + SVE (Scalable Vector Extension)**
  - FP64/FP32/FP16 (https://developer.arm.com/products/architecture/a-profile/docs)
- **SVE 512-bit wide SIMD**
- **# of Cores: 48 + (2/4 for OS)**
- Co-design with application developers and high memory bandwidth utilizing on-package stacked memory: **HBM2(32GiB)**
- Leading-edge Si-technology (7nm FinFET), low power logic design **(approx. 15 GF/W (dgemm))**, and power-controlling knobs
- Clock frequency:
  - 2.0 GHz(normal), 2.2 GHz (boost)
- Peak performance
  - 3.0 TFLOPS@2GHz (>90% @ dgemm)
  - Memory B/W 1024GB/s (>80% stream)
  - Byte per Flops: 0.33

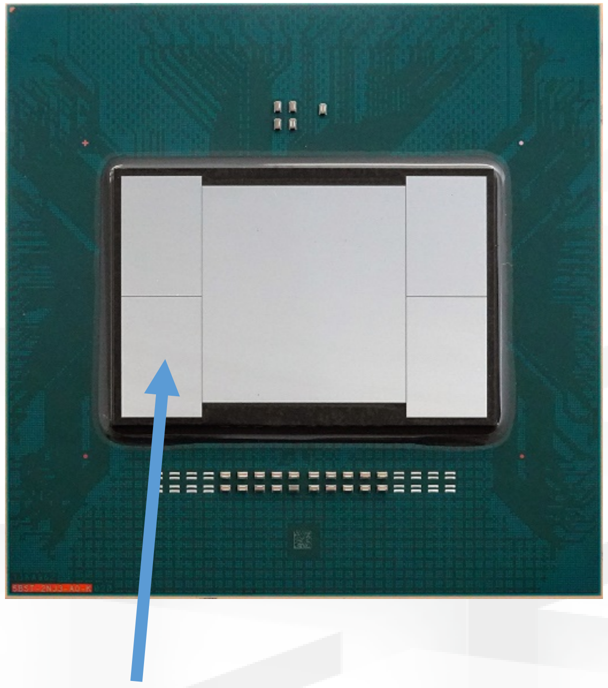
◆ "Common" programing model will be to run each MPI process on a NUMA node (CMG) with OpenMP-MPI hybrid programming.

◆ 48 threads OpenMP is also supported.

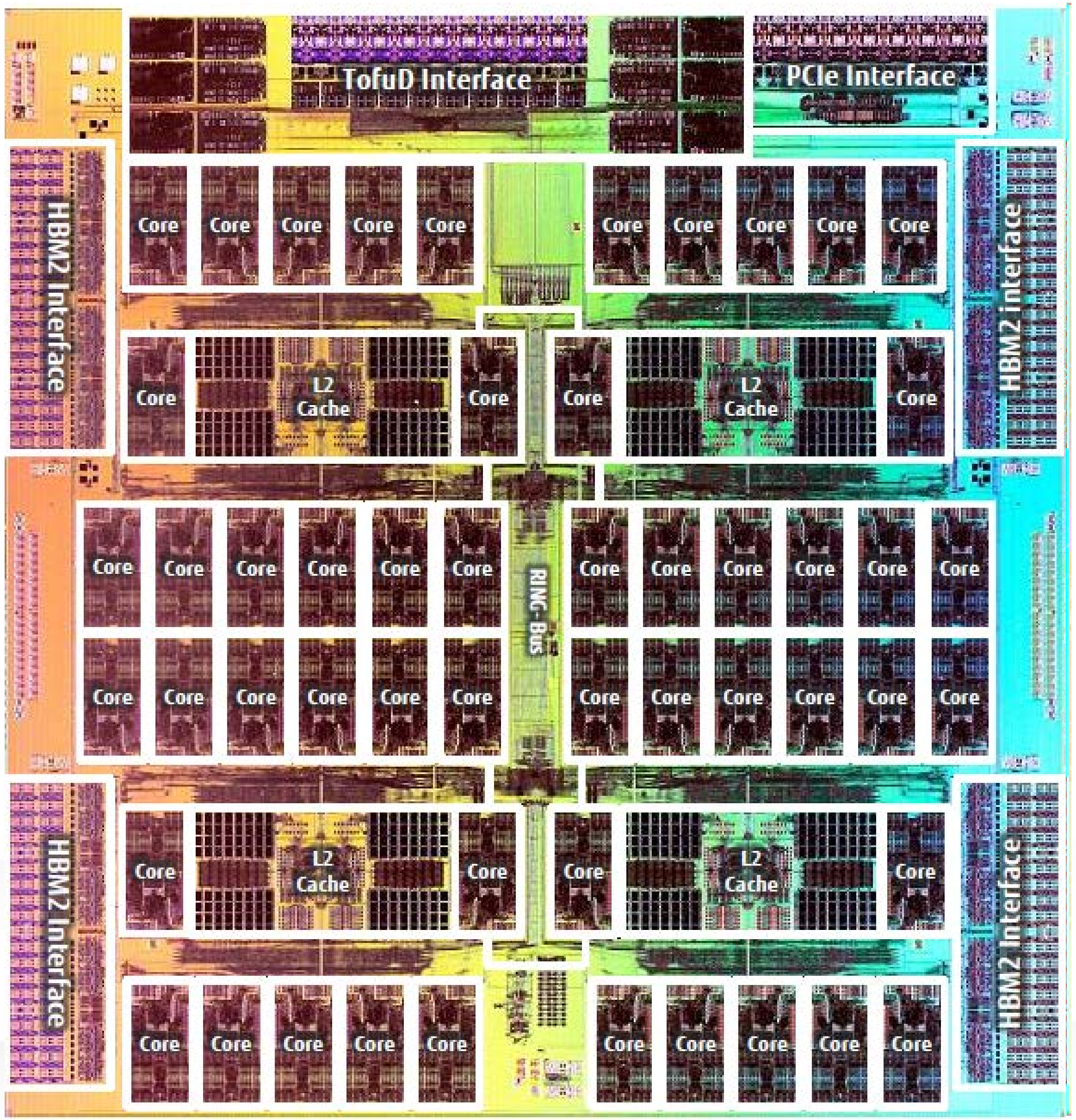

CMG(Core-Memory-Group): NUMA node
12+1 core

HBM2: 8GiB

- **TSMC 7nm FinFET**
- **400 mm^2**
- **HBM2 chips are mounted on Si-interposer connected by TSMC CoWoS technology**



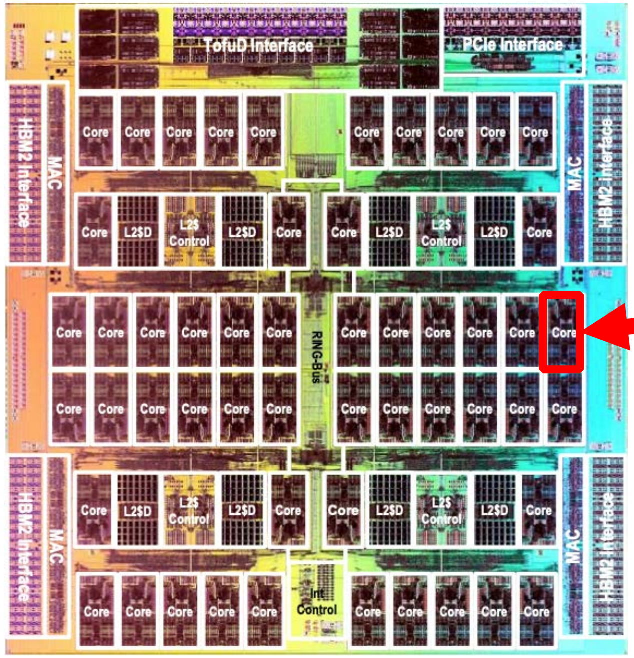
HBM2



Comparison of Die-size

- **A64FX: 52 cores (48 cores), 400 mm² die size (8.3 mm²/core), 7nm FinFET process (TSMC)**
- **Xeon Skylake: 20 tiles (5x4), 18 cores, ~485 mm² die size (estimated) (26.9 mm²/core), 14 nm process (Intel)**
- **A64FX core is more than 3 times smaller per core.**

A64FX:
400 mm²
(20 x 20)



https://www.fujitsu.com/jp/solutions/business-technology/tc/catalog/ff2019-post-k-computer-development.pdf

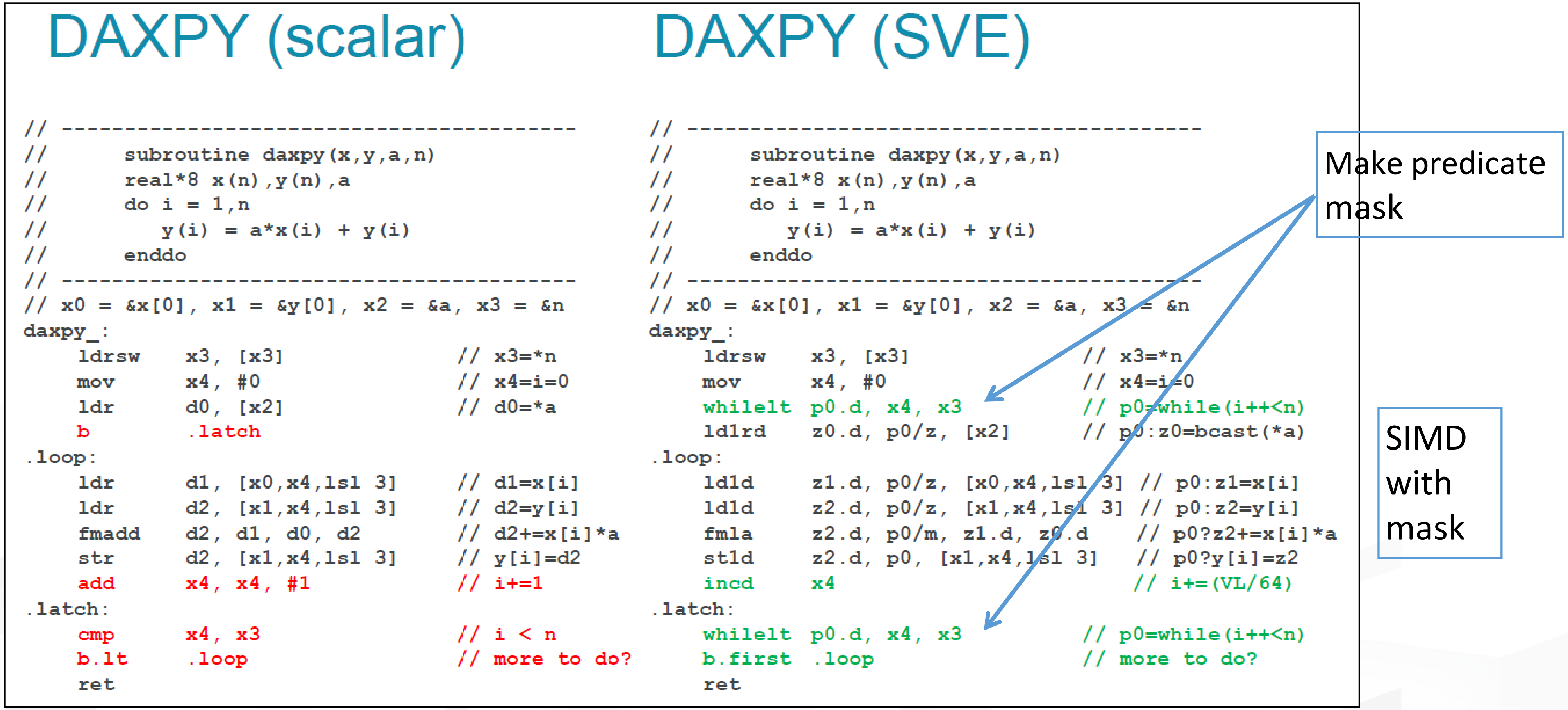
Xeon Skylake, High Core Count:
4 x 5 tiles, 18 cores, 2 tiles used for memory interface
485 mm² (22 x 22)

https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)

ARM v8 Scalable Vector Extension (SVE)

- **SVE is a complementary extension that does not replace NEON, and was developed specifically for vectorization of HPC scientific workloads.**
- **The new features and the benefits of SVE comparing to NEON**
  - **Scalable vector length (VL)** : Increased parallelism while allowing implementation choice of VL
  - **VL agnostic (VLA) programming**: Supports a programming paradigm of write-once, run-anywhere scalable vector code
  - **Gather-load & Scatter-store**: Enables vectorization of complex data structures with non-linear access patterns
  - **Per-lane predication**: Enables vectorization of complex, nested control code containing side effects and avoidance of loop heads and tails (particularly for VLA)
  - Predicate-driven loop control and management: Reduces vectorization overhead relative to scalar code
  - Vector partitioning and SW managed speculation: Permits vectorization of uncounted loops with data-dependent exits
  - Extended integer and floating-point horizontal reductions: Allows vectorization of more types of reducible loop-carried dependencies
  - Scalarized intra-vector sub-loops: Supports vectorization of loops containing complex loop-carried dependencies

SVE example



DAXPY (scalar)

```
// ---------------------------------------
//      subroutine daxpy(x,y,a,n)
//      real*8 x(n),y(n),a
//      do i = 1,n
//         y(i) = a*x(i) + y(i)
//      enddo
// ---------------------------------------
// x0 = &x[0], x1 = &y[0], x2 = &a, x3 = &n
daxpy_:
    ldrsw    x3, [x3]               // x3=*n
    mov      x4, #0                 // x4=i=0
    ldr      d0, [x2]               // d0=*a
    b        .latch
.loop:
    ldr      d1, [x0,x4,lsl 3]      // d1=x[i]
    ldr      d2, [x1,x4,lsl 3]      // d2=y[i]
    fmadd    d2, d1, d0, d2         // d2+=x[i]*a
    str      d2, [x1,x4,lsl 3]      // y[i]=d2
    add      x4, x4, #1             // i+=1
.latch:
    cmp      x4, x3                 // i < n
    b.lt     .loop                  // more to do?
    ret
```

DAXPY (SVE)

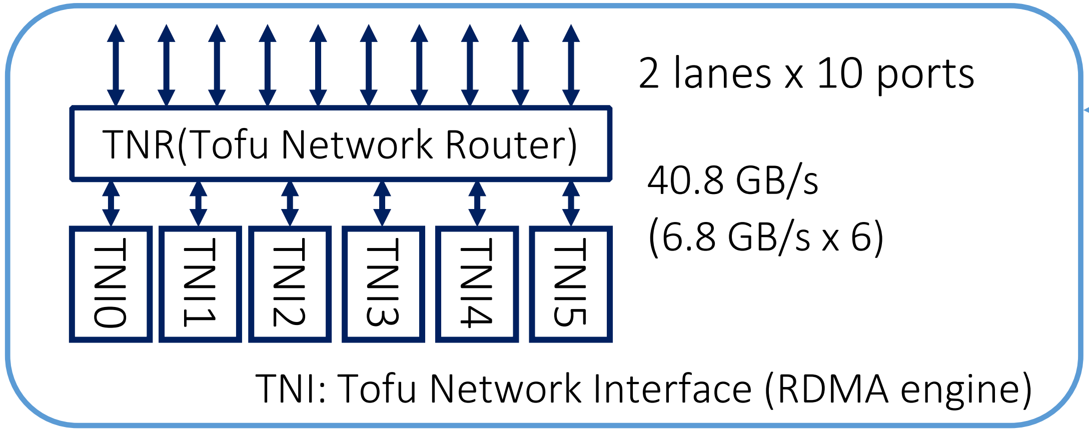
```
// ---------------------------------------
//      subroutine daxpy(x,y,a,n)
//      real*8 x(n),y(n),a
//      do i = 1,n
//         y(i) = a*x(i) + y(i)
//      enddo
// ---------------------------------------
// x0 = &x[0], x1 = &y[0], x2 = &a, x3 = &n
daxpy_:
    ldrsw    x3, [x3]                   // x3=*n
    mov      x4, #0                     // x4=i=0
    whilelt  p0.d, x4, x3               // p0=while(i++<n)
    ld1rd    z0.d, p0/z, [x2]           // p0:z0=bcast(*a)
.loop:
    ld1d     z1.d, p0/z, [x0,x4,lsl 3]  // p0:z1=x[i]
    ld1d     z2.d, p0/z, [x1,x4,lsl 3]  // p0:z2=y[i]
    fmla     z2.d, p0/m, z1.d, z0.d     // p0?z2+=x[i]*a
    st1d     z2.d, p0, [x1,x4,lsl 3]    // p0?y[i]=z2
    incd     x4                         // i+=(VL/64)
.latch:
    whilelt  p0.d, x4, x3               // p0=while(i++<n)
    b.first  .loop                      // more to do?
    ret
```

Make predicate mask

SIMD with mask

- **Compact code for SVE as scalar loop**
- **OpenMP SIMD directive is expected to help the SVE programming**

TofuD Interconnect



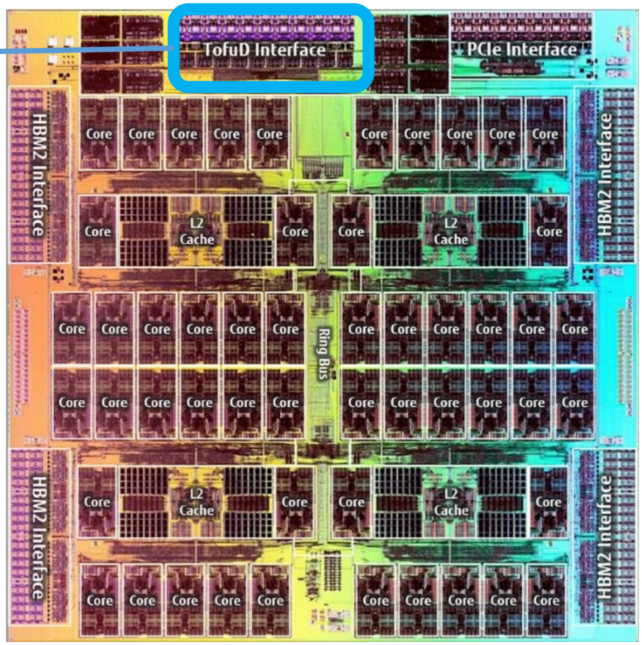
2 lanes x 10 ports

TNR(Tofu Network Router)

40.8 GB/s
(6.8 GB/s x 6)

TNI0
TNI1
TNI2
TNI3
TNI4
TNI5

TNI: Tofu Network Interface (RDMA engine)



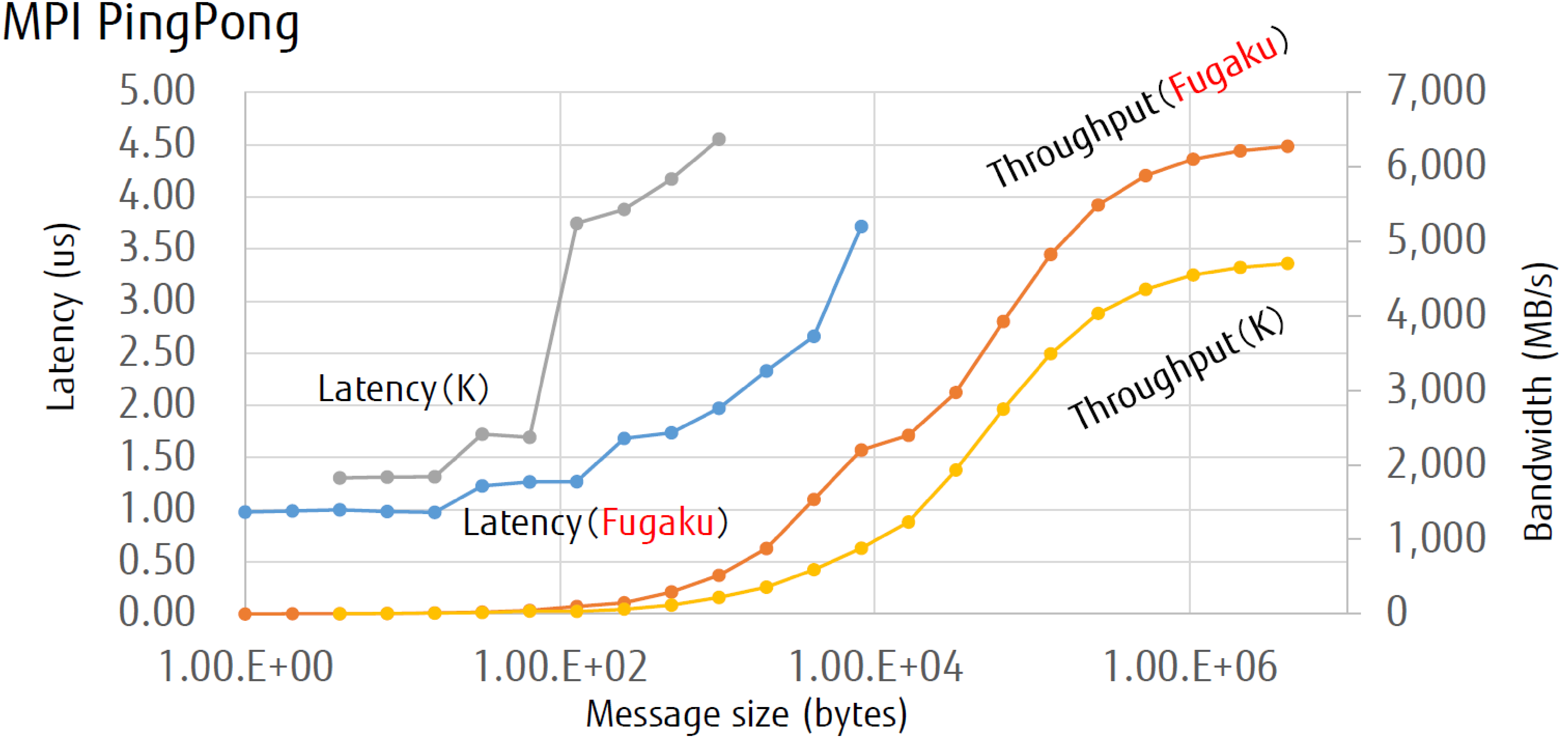
- 6 RDMA Engines
- Hardware barrier support
- Network operation offloading capability

8B Put latency	0.49 – 0.54 usec
1MiB Put throughput	6.35 GB/s

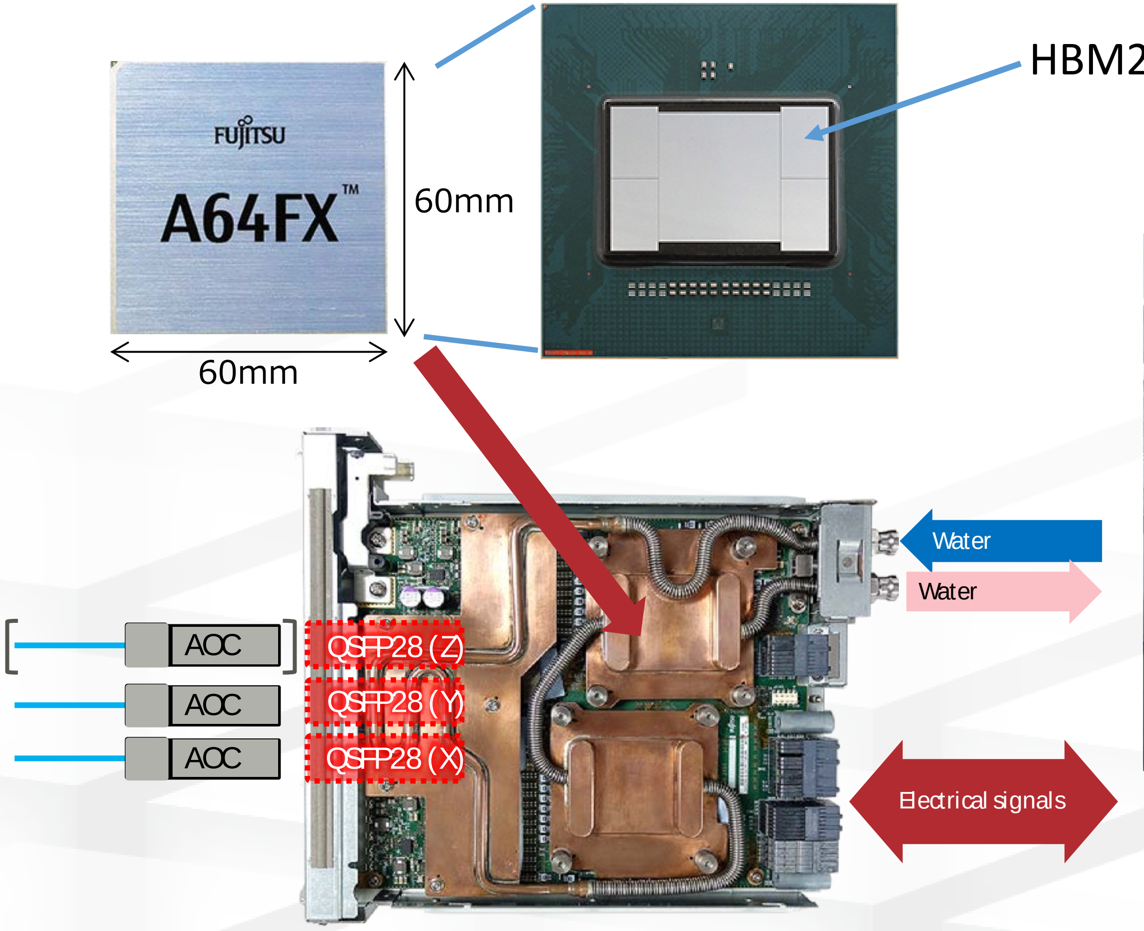
rf. Yuichiro Ajima, et al. , “The Tofu Interconnect D,” IEEE Cluster 2018, 2018.

TofuD: MPI_Send/Receive Latency and BW

- MPI PingPong



Fugaku prototype board and rack



2 CPU / CMU

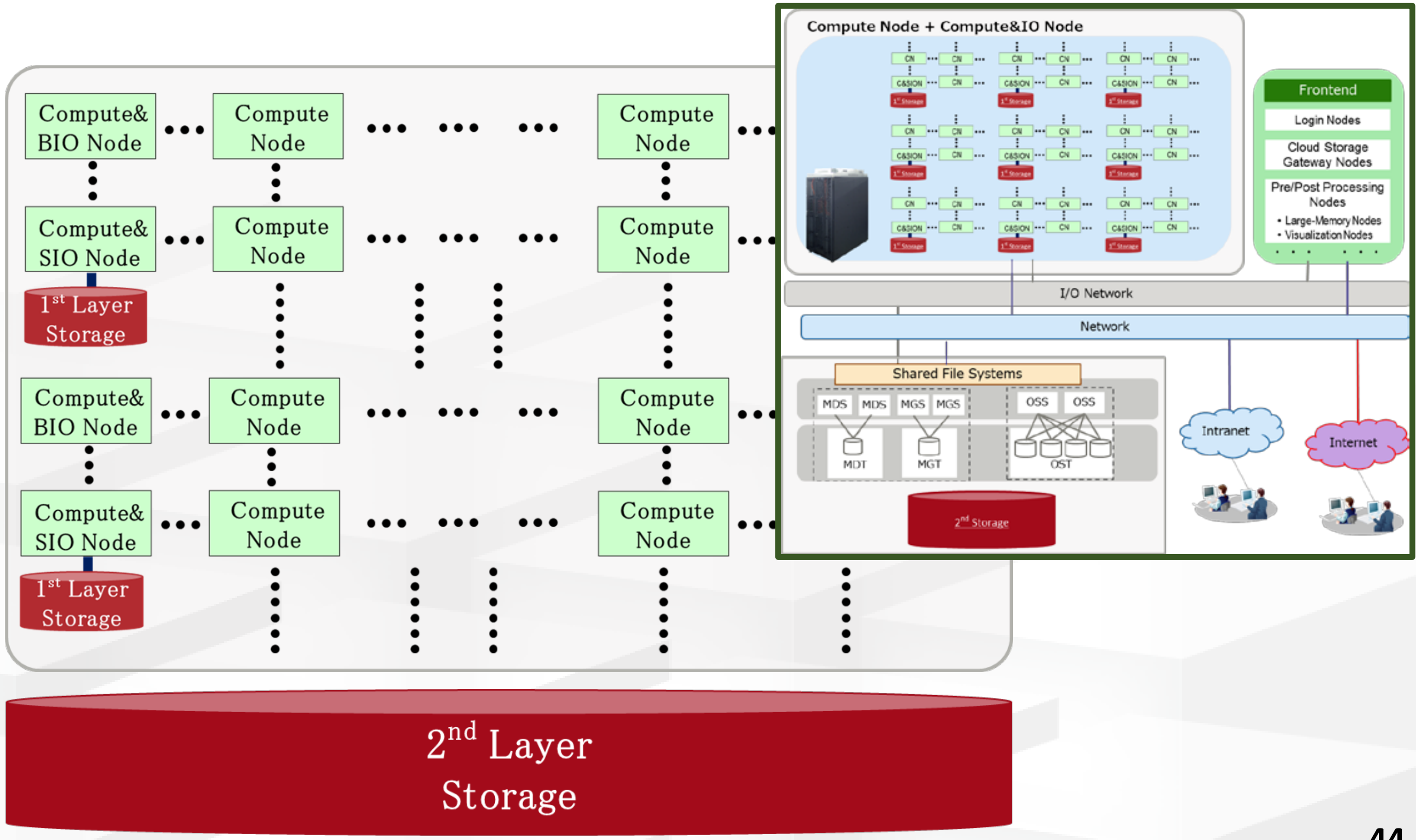
Shelf: 48 CPUs (24 CMU)
Rack: 8 shelves = 384 CPUs (8x48)

Fugaku System Configuration

- **158,976 node**
- **Two types of nodes**
  - Compute Node and Compute & I/O Node connected by Fujitsu Tofu-D, 6D mesh/torus Interconnect
- **3-level hierarchical storage system**
  - 1st Layer
    - One of 16 compute nodes, called Compute & Storage I/O Node, has SSD about 1.6 TB
    - Services
      - Cache for global file system
      - Temporary file systems
        - Local file system for compute node
        - Shared file system for a job
  - 2nd Layer
    - Fujitsu FEFS: Lustre-based global file system, about 150 PB
  - 3rd Layer
    - Cloud storage services

Storage System

		Minimum Throughput	Measured Throughput
1st Storage	Write	49 MB/s /node	125 MB/s /node
	Read	113 MB/s /node	293 MB/s /node
2nd Storage	Write	200 GB/s /volume	220 GB/s /volume
	Read		211 GB/s /volume



Co-Design for Storage and File I/O (1/2)

- *Application I/O Characteristics*

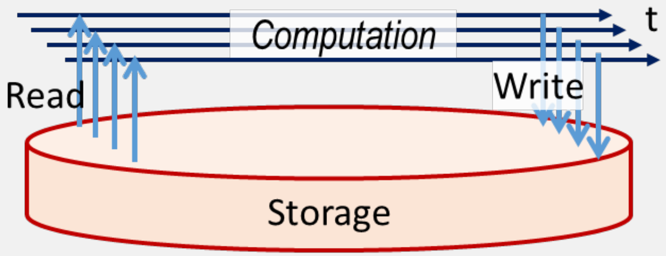
I/O pattern	Target Application	Read (GB)	Total Cyclic Write (GB)	Write (GB)	Execution Time (sec)
P2	GAMERA	67,019	109,486 (3,650/cycle)	3,650	84,390
P3	NICAM + LETKF	412,317	13,194,140 (3,298,535/cycle)	3,221	77,840

In GAMERA, the total file size is about 180 TB. although we assume 844 s (only 1% overhead of total execution time (84,390 s)) for file I/O processing is allowed, the required effective I/O throughput is:

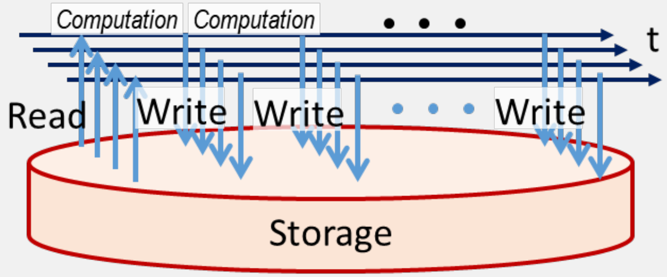
213*GB/s* ➡ *NOT HIGH DEMAND*
180 *TB* / 844 *sec*

		Minimum Throughput	Measured Throughput
1st Storage	Write	49 MB/s /node	125 MB/s /node
	Read	113 MB/s /node	293 MB/s /node
2nd Storage	Write	200 GB/s /volume	220 GB/s /volume
	Read		211 GB/s /volume

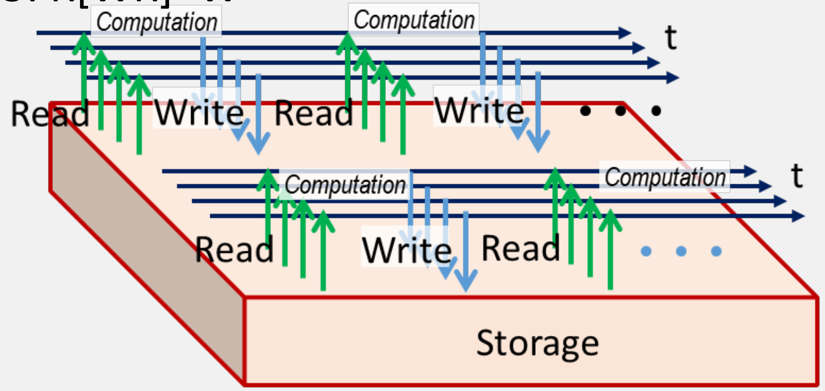
P1: RW



P2: R[W]*W



P3: R[WR]*W



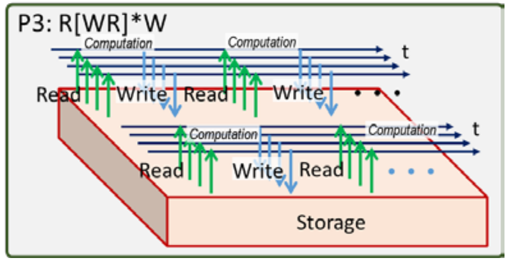
Co-Design for Storage and File I/O (1/2)

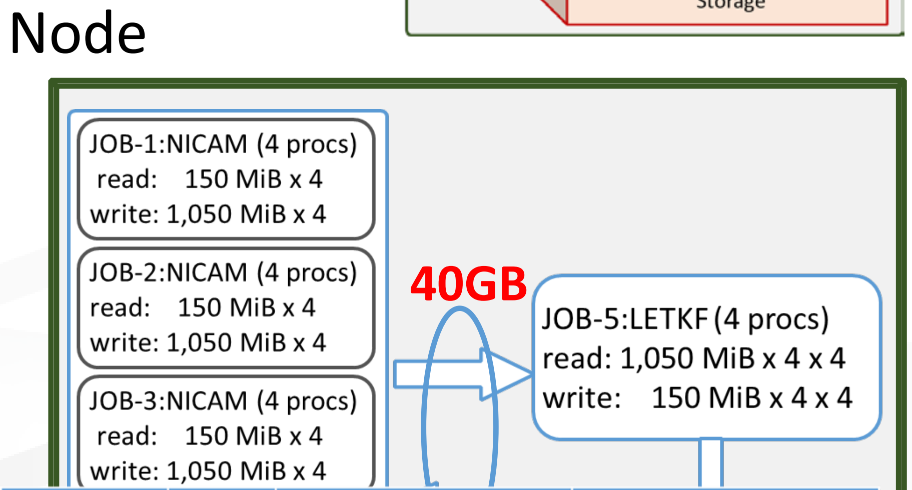
- **NICAM+ LETKF**
  - 4 NICAM (Ensemble Simulation) jobs followed by 1 LETKF (Data Assimilation) job

  Using 163,840 nodes (at the co-design phase)
- **Characteristics of File I/O pattern in one Cyclic Execution Phase (19,460 sec)**
  - Data exchanges between 4 NICAM jobs and 1 LETKF job
    - 40 GB / cycle
  - No needs to save the data to a persistent storage
    - Keeping data on the 1st layer storage

The I/O time for 40 GB on the 1st layer storage is about
816 sec (40GB/49MB/s /node)
=> 4 % I/O time of 1 cycle execution time

I/O pattern	Target Application	Read (GB)	Total Cyclic Write (GB)	Write (GB)	Execution Time (sec)
P3	NICAM + LETKF	412,317	13,194,140 (3,298,535/cycle)	3,221	77,840





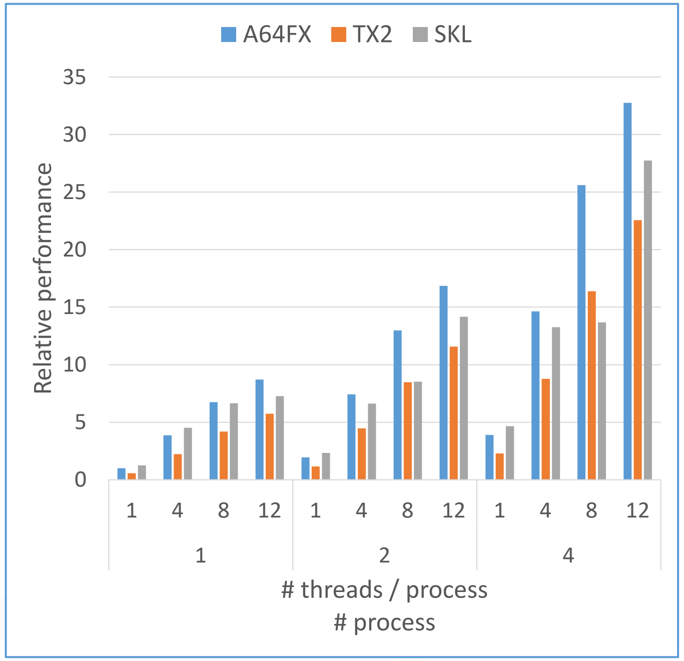
JOB-1:NICAM (4 procs)
read: 150 MiB x 4
write: 1,050 MiB x 4

JOB-2:NICAM (4 procs)
read: 150 MiB x 4
write: 1,050 MiB x 4

JOB-3:NICAM (4 procs)
read: 150 MiB x 4
write: 1,050 MiB x 4

JOB-5:LETKF (4 procs)
read: 1,050 MiB x 4 x 4
write: 150 MiB x 4 x 4

		Minimum Throughput	Measured Throughput
1st Storage	Write	49 MB/s /node	125 MB/s /node
	Read	113 MB/s /node	293 MB/s /node
2nd Storage	Write	200 GB/s /volume	220 GB/s /volume
	Read		211 GB/s /volume



Performance of “Fugaku”

CloverLeaf from UK benchmark, and Open-Source HPC Software

Benchmark Results on test chip A64FX

- **CloverLeaf (UK Mini-App Consortium), Fortran/C**
  - A hydrodynamics mini-app to solve the compressible Euler equations in 2D, using an explicit, second-order method
  - Stencil calculation
- **TeaLeaf (UK Mini-App Consortium), Fortran**
  - A mini-application to enable design-space explorations for iterative sparse linear solvers
  - https://github.com/UK-MAC/TeaLeaf_ref.git
  - Problem size: Benchmarks/tea_bm_5.in, end_step=10 -> 3
- **LULESH (LLNL), C**
  - Mini-app representative of simplified 3D Lagrangian hydrodynamics on an unstructured mesh, indirect memory access

Processors for comparison

	A64FX	TX2 (ThunderX2)	SKL (Skylake)
# cores	48 (1 CPU)	56 (28 x 2 sockets)	24 (12 x 2 sockets)
Clock	2.0 GHz (Normal)	2.0 GHz	2.6 GHz (※)
SIMD	SVE 512-bit	NEON 128-bit	AVX512 512-bit
Memory Peak bandwidth	HBM2 1,024 GB/s	DDR4-8ch 341 GB/s	DDR4-6ch 256 GB/s
Network	TofuD	InfiniBand FDR x 1	InfiniBand HDR x 1
Compiler Options	Fujitsu compiler 4.1.0 -Kfast,openmp	Arm HPC compiler 19.1 -Ofast -fopenmp -march=armv8.1-a	Intel compiler 19.1 -O3 -qopenmp -march=native

(※) AVX512 instruction is executed at 90% peak Feq.

Threads and sockets and nodes

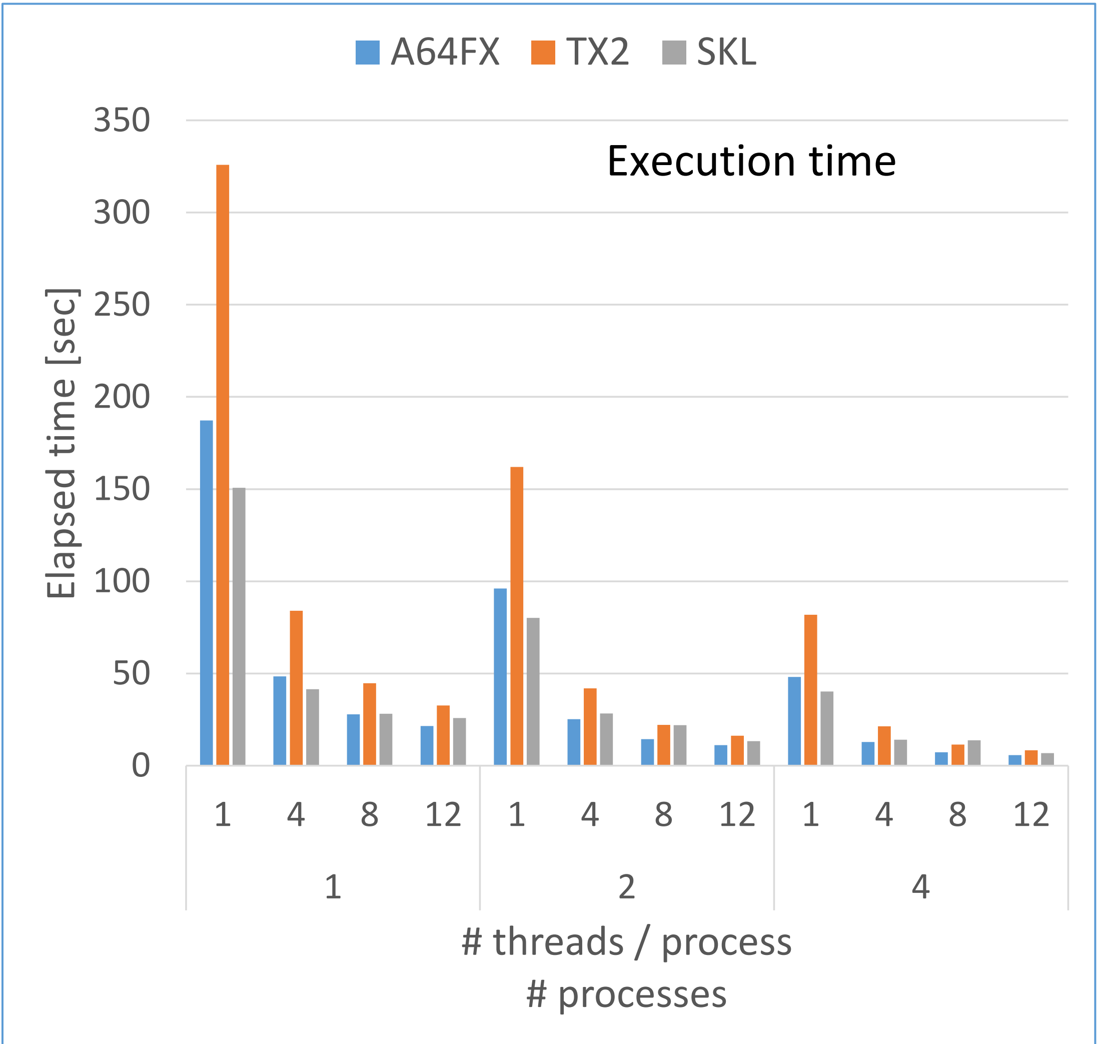
- **#threads ≦ 12**
  - A64FX: execute on only CMG0
  - TX2, SKL: execute on only Socket0
- **12 < #threads ≦ 24**
  - A64FX: execute on CMG0 and CMG1
  - TX2, SKL: execute on one node (max #threads: 12 on a socket)
- **24 < #threads ≦ 48**
  - A64FX: execute no one node
  - TX2, SKL: execute on two node (max #threads: 12 on a socket)

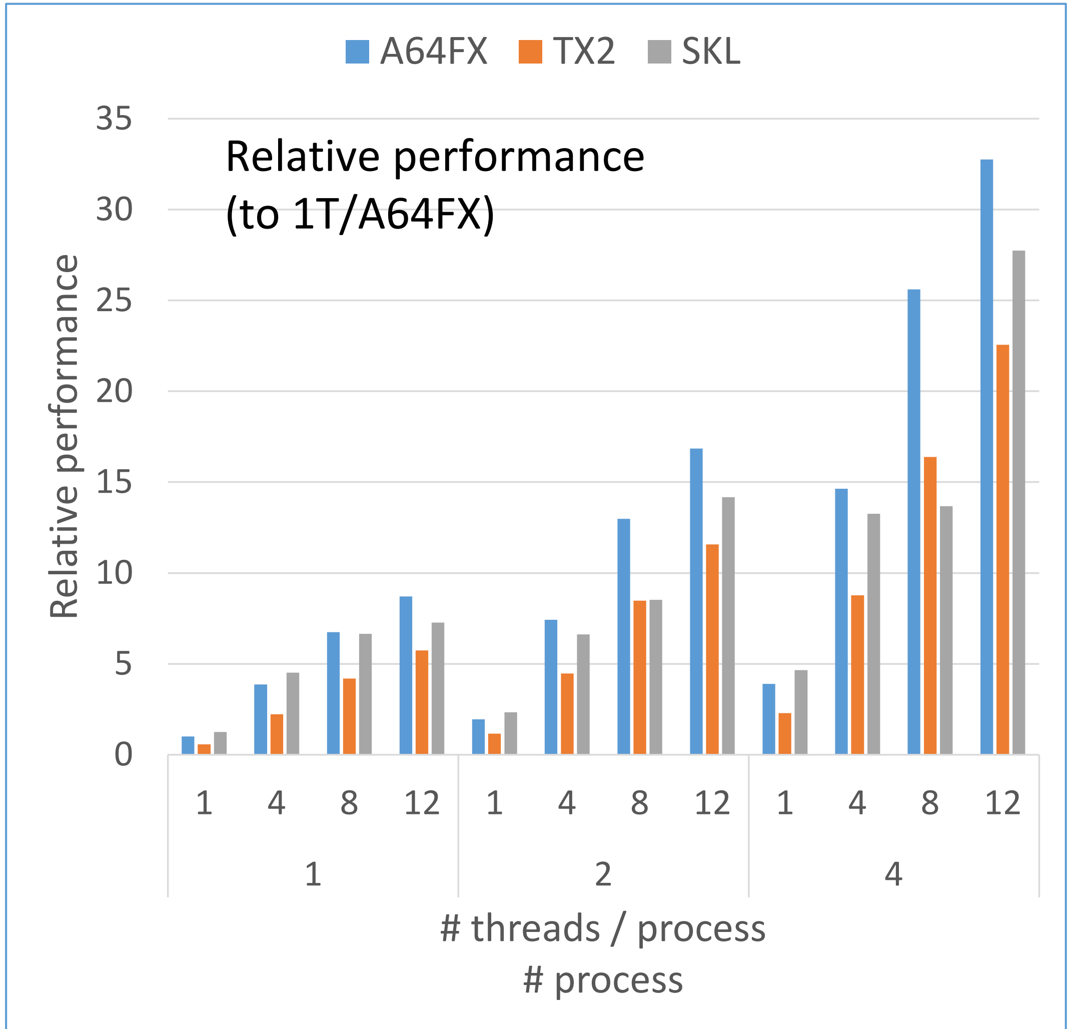
Disclaimer:
The software used for the evaluation, such as the compiler, is still under development and its performance may be different when the supercomputer Fugaku starts its operation.

CloverLeaf

Taken form UK benchmarks:
A hydrodynamics mini-app to solve the compressible Euler equations in 2D, using an explicit, second-order method

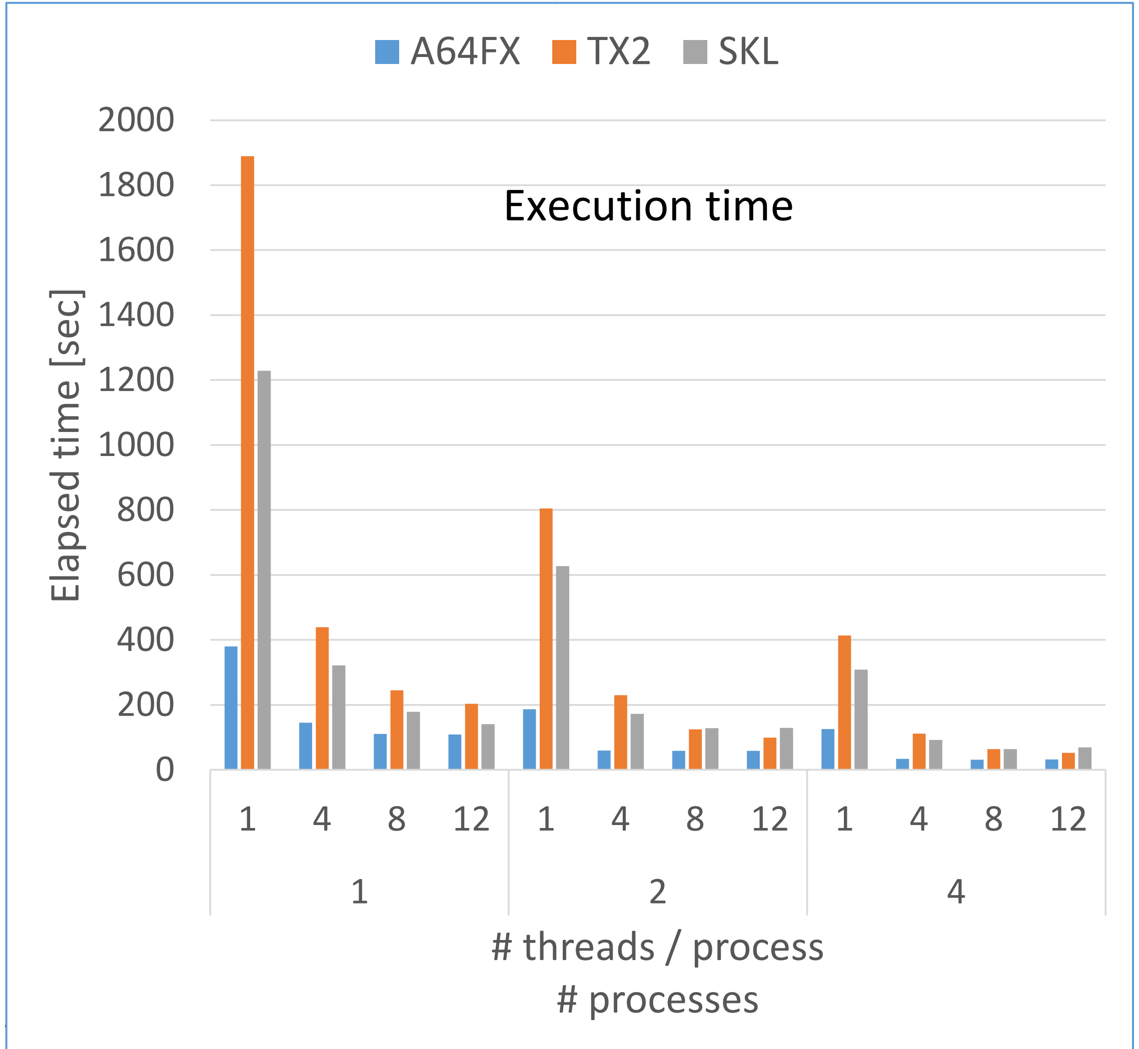
- Comparison with two nodes of TX2 (dual) and Skylake (dual)
- Good scalability by increasing the number of threads within CMG.
- The performance of one A64FX is comparable (better) to that of two nodes (4 chips) of Skylake

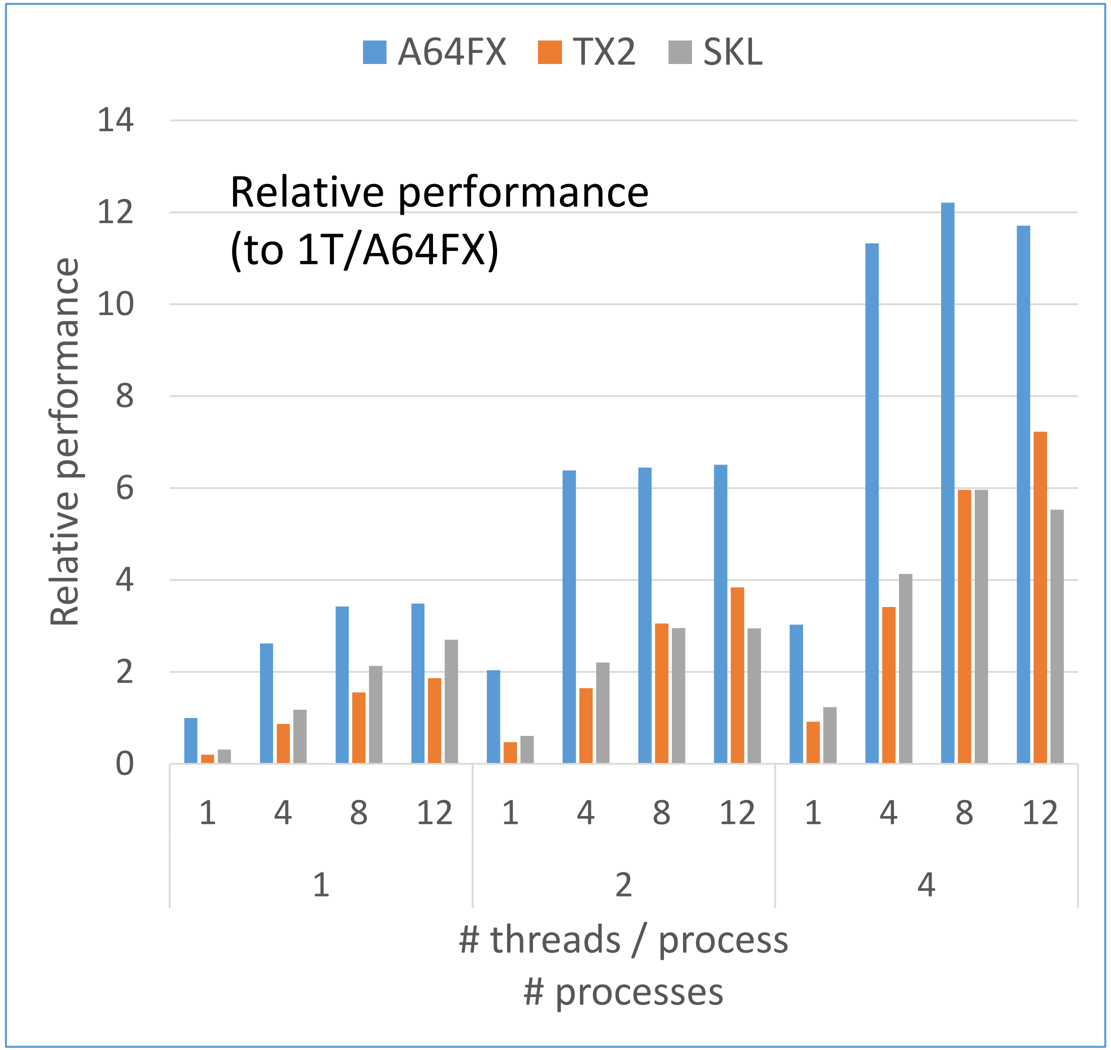




TeaLeaf

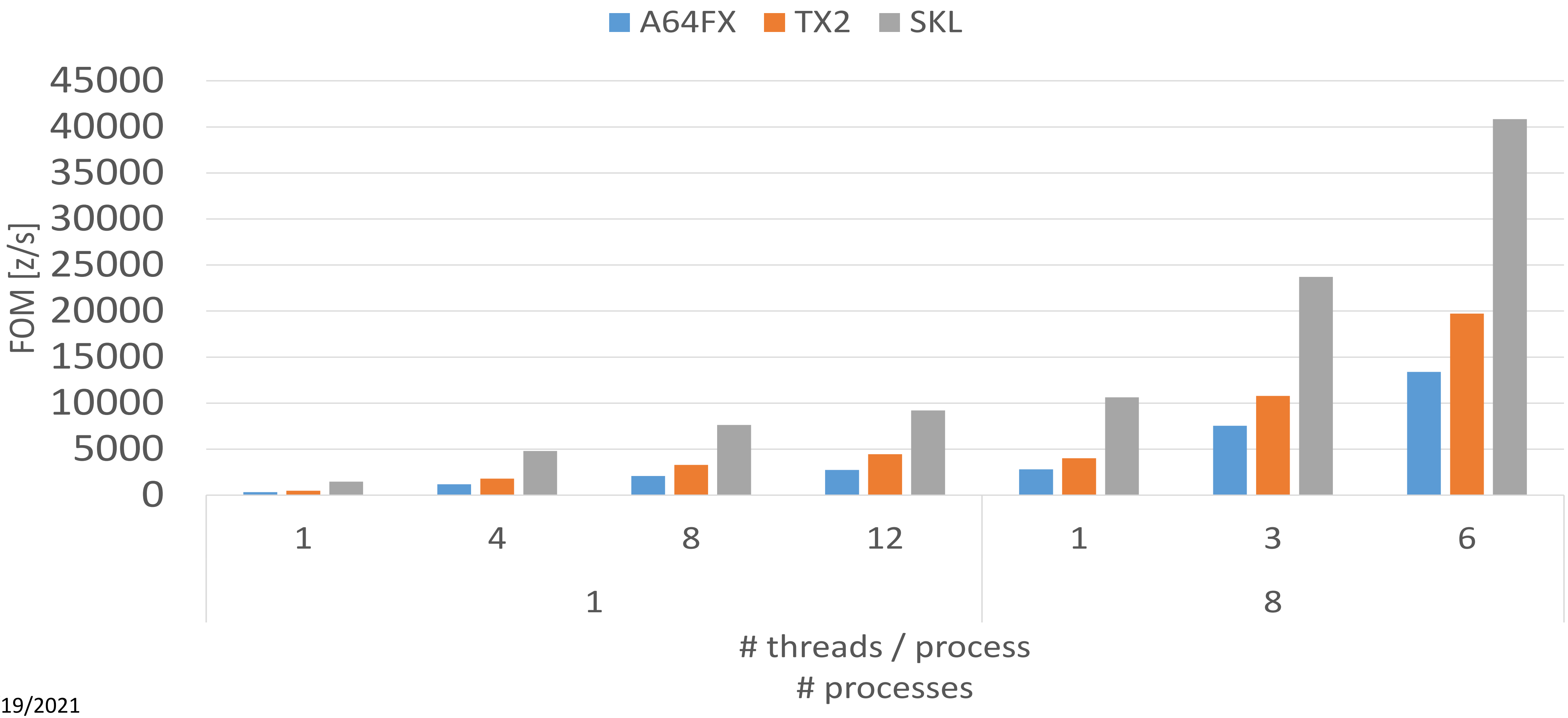
- Memory bandwidth intensive application. The speedup is limited for more than 4 threads due to the memory bandwidth.
- The performance of one A64FX is twice better than that of two nodes (4 chips) of Skylake. It reflects the difference of total memory bandwidth.





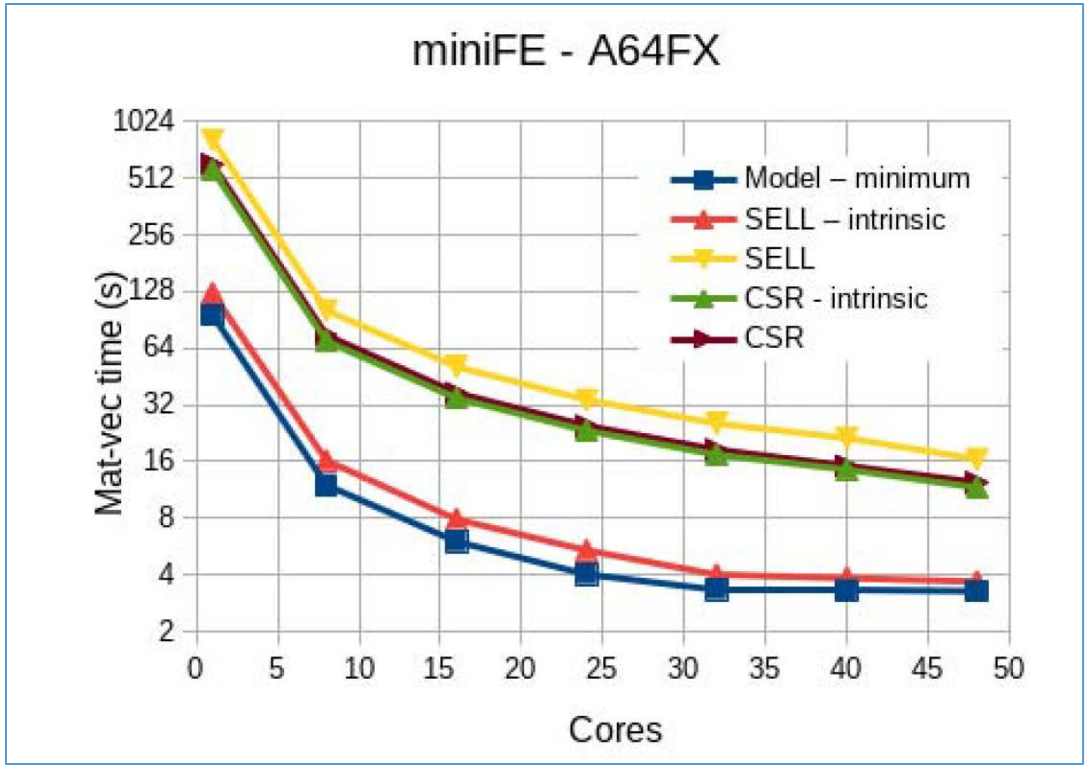
LULESH

- A64FX performance is less than Thx2 and Intel one
- We found low vectorization (SIMD (SVE) instructions ratio is a few %)
- We need more code tuning for more vectorization using SIMD



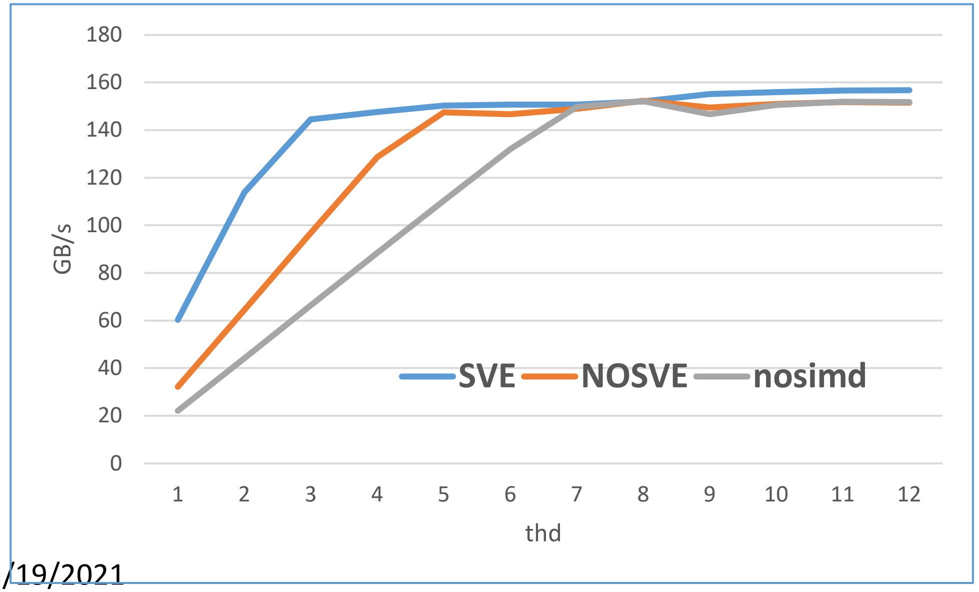
How to improve the performance of sparse-matrix code

- **Storage format is important:**
  - Sliced ELLPACK format shows significantly better performance than CSR, but only when it is vectorized manually using intrinsics.”
  - CSR is not good even with manual vectorizing.
- **Vectorizing with SVE is important to get memory bandwidth.**



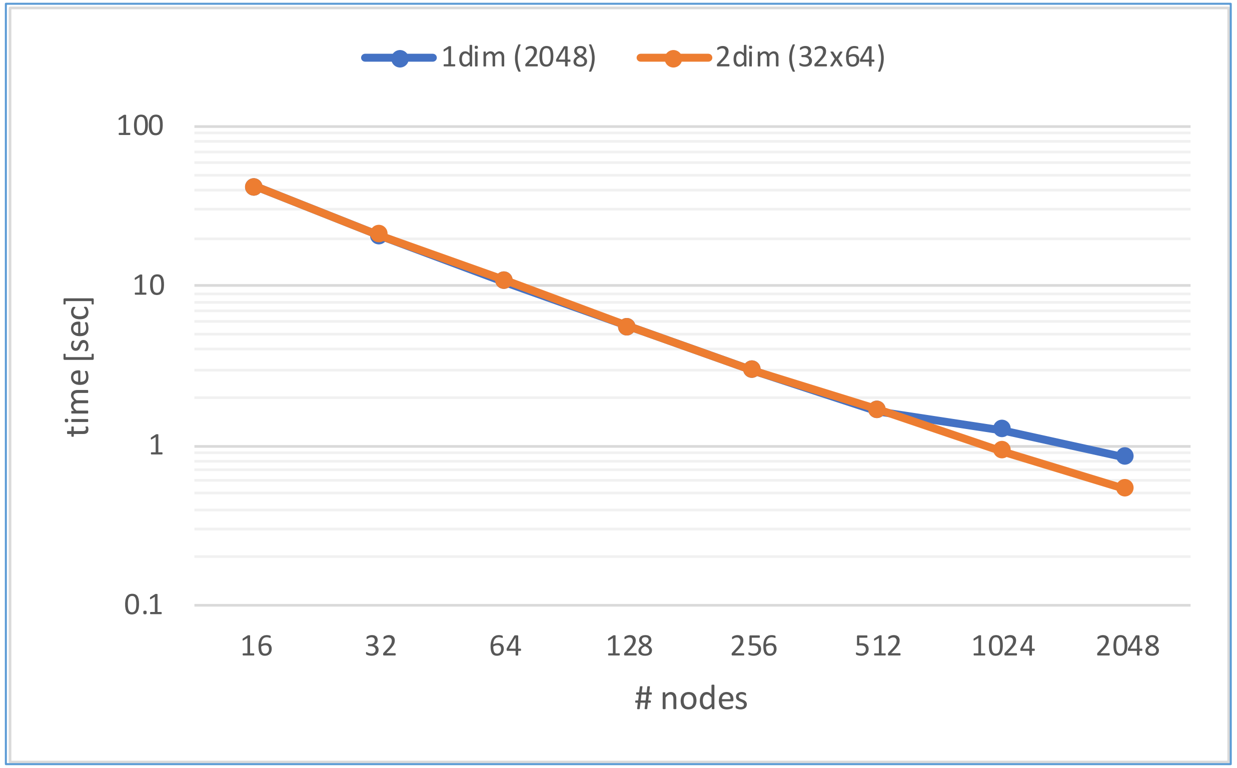
B. Brank, S. Nassyr, F. Pouyan and D. Pleiter, "Porting Applications to Arm-based Processors," EAHPC Workshop, *IEEE CLUSTER* 2020 , Kobe, Japan, 2020, pp. 559-566, doi: 10.1109/CLUSTER49012.2020.00079.

Memory bandwidth with hardware prefetch



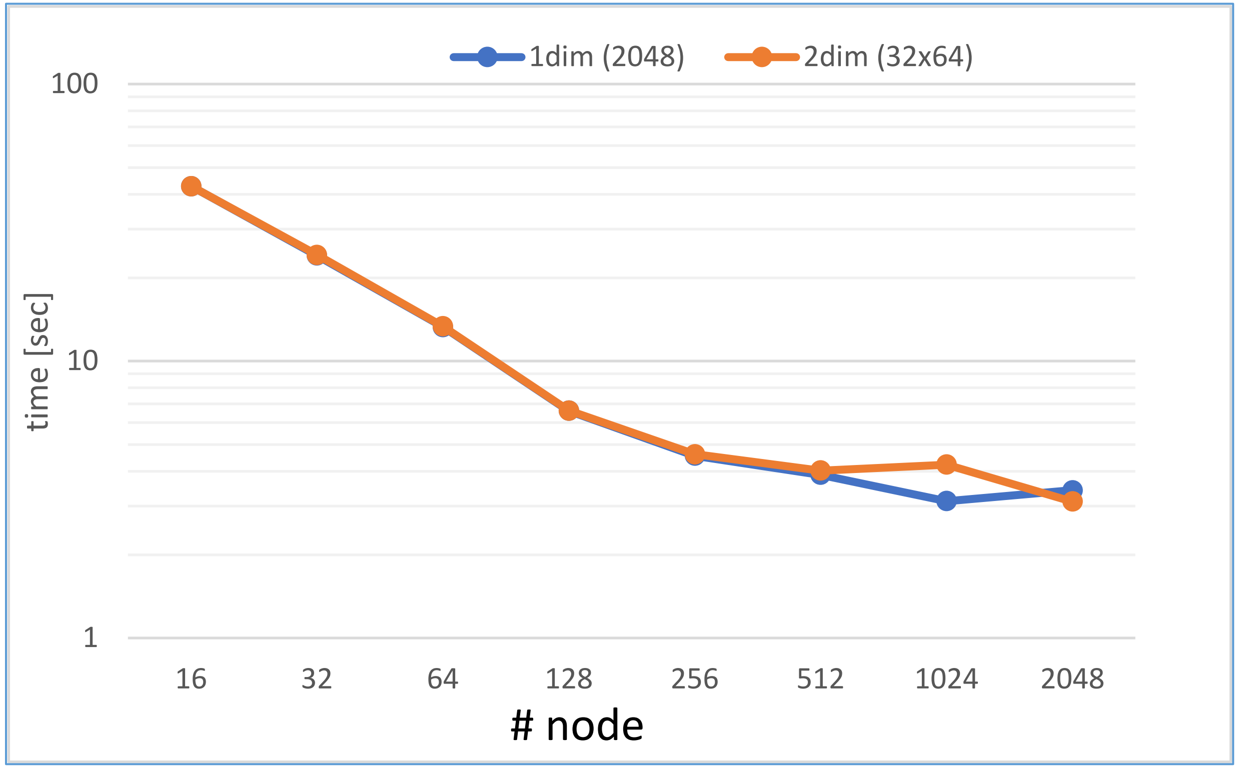
Scalability for Multi-nodes

- **Strong scaling in CloverLeaf and TeaLeaf (FlatMPI) up to 2048 nodes**
- **CloverLeaf : Good scalability for 2D**
- **TeaLeaf: Limited by communication (helo and dot)**



CloverLeaf

Problem Size: InputDecks/clover_bm2048_short.in

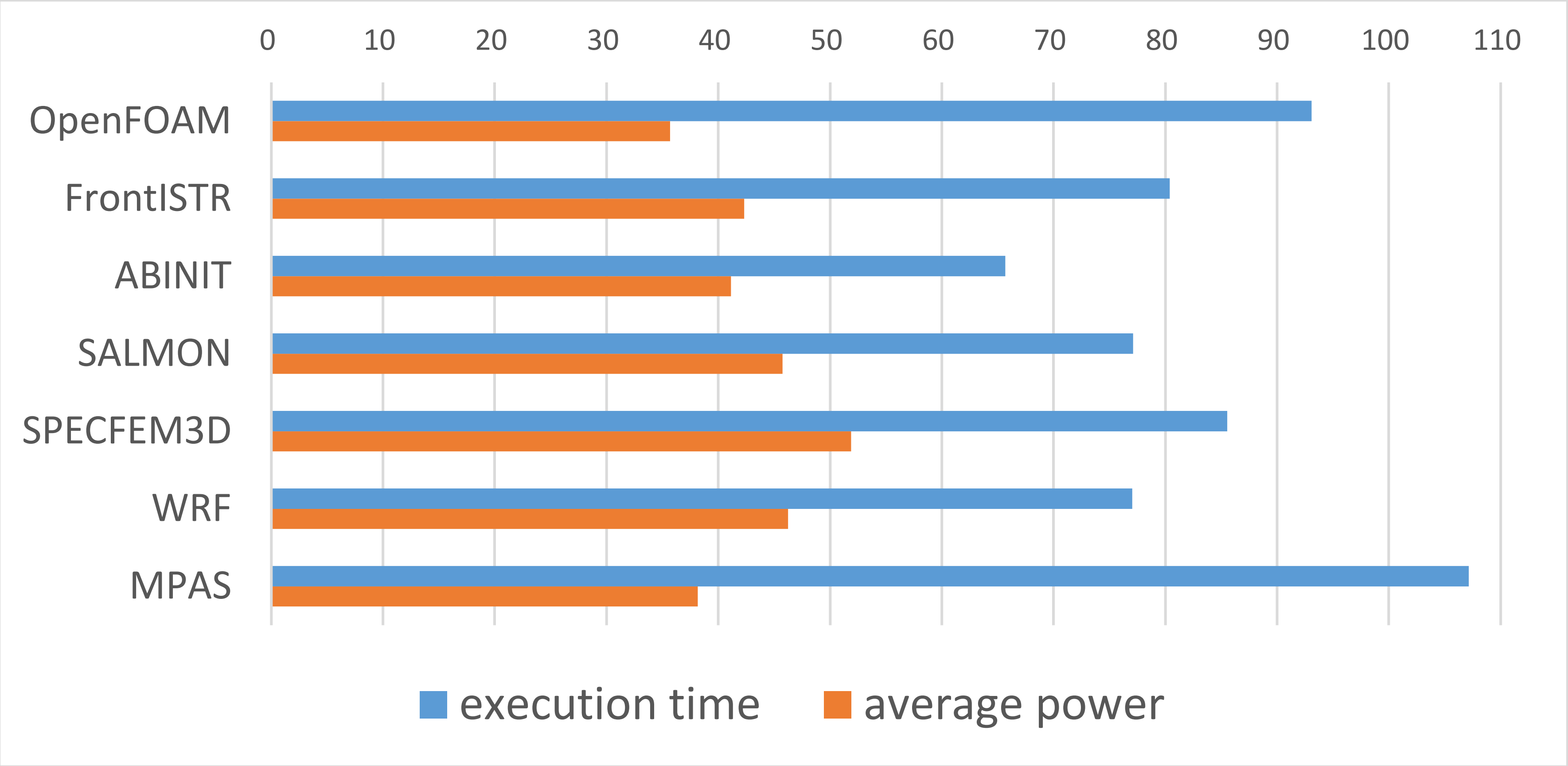


TeaLeaf

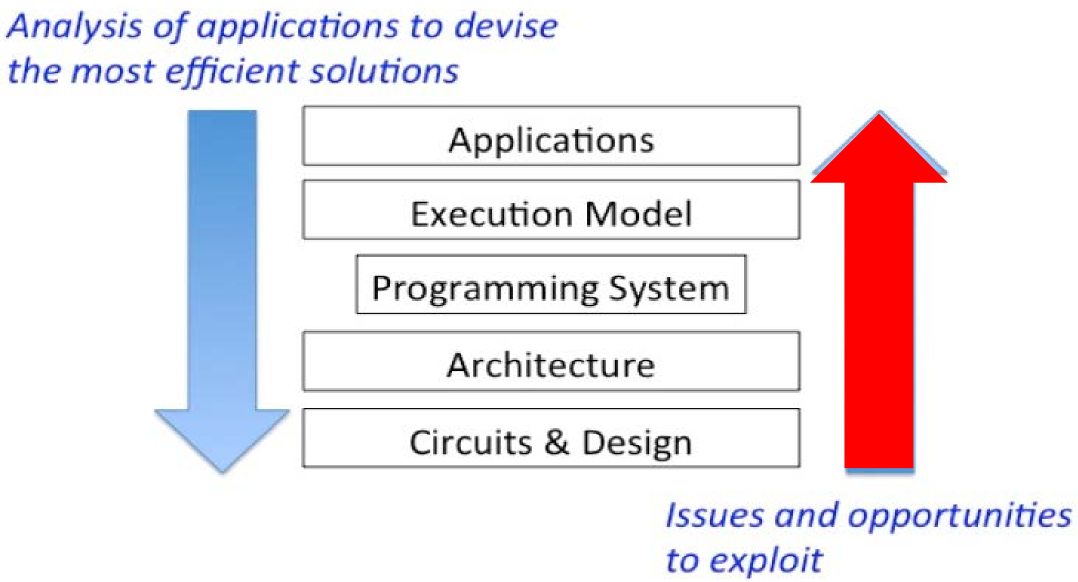
Performance and Power-efficiency of HPC OSS

- **Several Open-source software were already ported and evaluated.**
- **Evaluation using one chip A64FX and dual chips of Xeon.**
- **The almost same performance to dual sockets of Xeon with half of power consumption.**

Performance and power efficiency of open-source applications (results are shown in %, relative to Intel Xeon Platinum 8268 (Cascadelake, 2.90 GHz, 24 cores/socket) (dual sockets))



Co-Design of Compiler and Applications



Analysis of applications to devise the most efficient solutions

Applications

Execution Model

Programming System

Architecture

Circuits & Design

Issues and opportunities to exploit

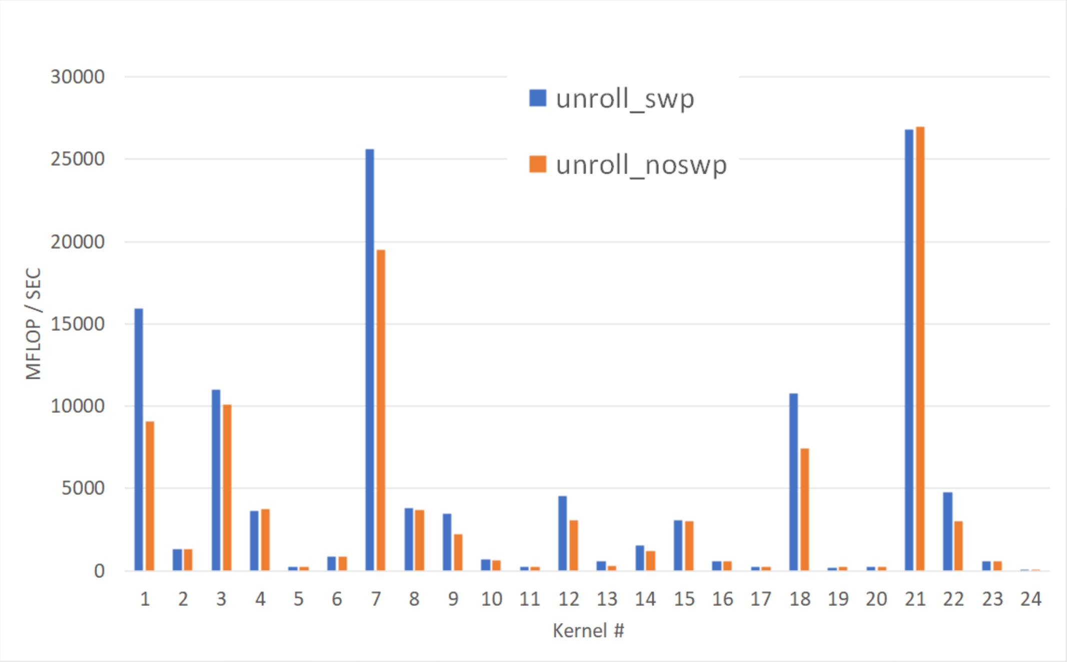
Compiler optimization for A64FX processor

- **HPC-oriented design**
  - Small core ⇒ Less O3 resources
  - (Relatively) Long pipeline
    - 9 cycles for floating point operations
    - Core has only L1 cache
  - High-throughput, but long-latency
  - Pipeline often stalls
    for loops having complex body.

	A64FX	Skylake
ReOrder Buffer	128 entries	224 entries
Reservation Station	60 (=10x2+20x2) entries	97 entries
Physical Vector Register	128 (=32 + 96) entries	168 entries
Load Buffer	40 entries	72 entries
Store Buffer	24 entries	56 entries

A64FX : https://github.com/fujitsu/A64FX
Skylake : https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)

- **Compiler optimization (Fujitsu compiler)**
  - SWP: software pipelining
    - ～ 20% speedup in Livermore Kernels
  - Automatic and Manual loop fissions



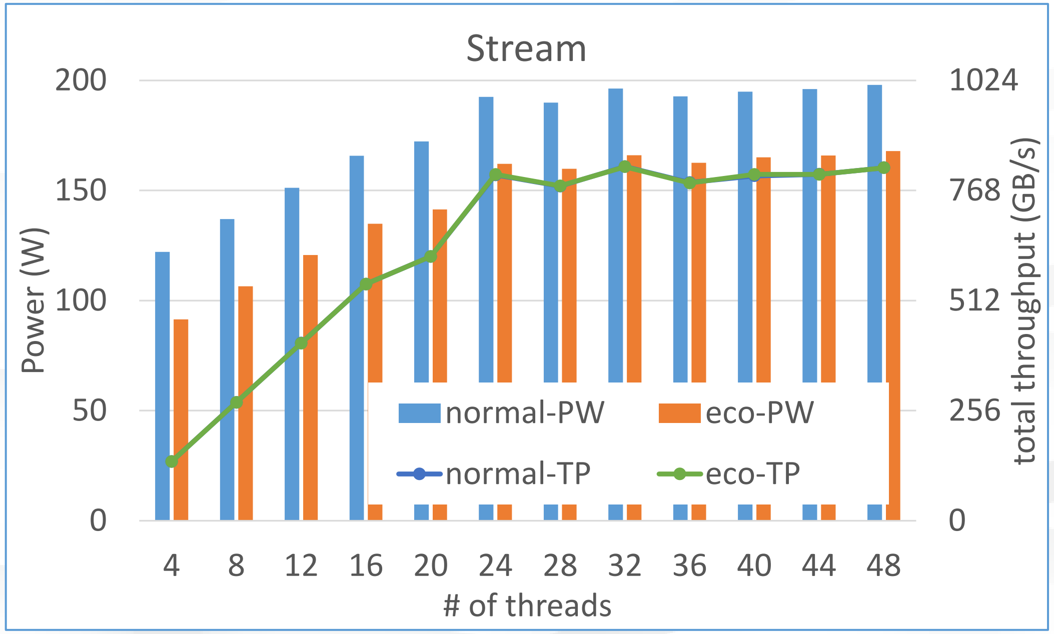
Performance improvement by SWP in Livermore Kernels by Fujitsu compiler

Co-Design of Applications: NICAM case

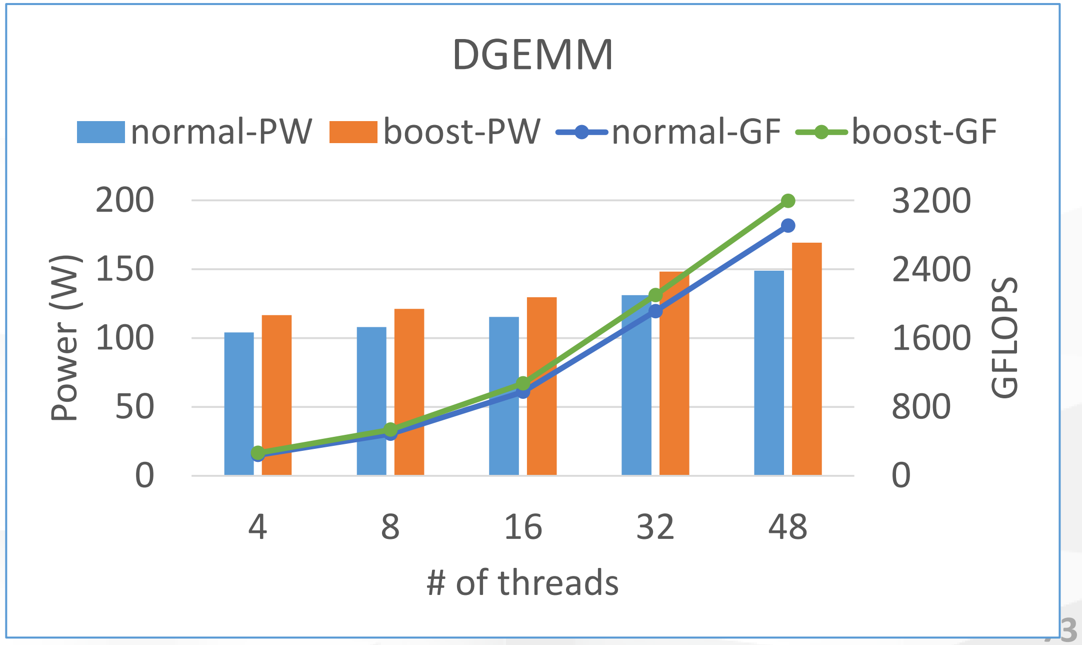
- **NICAM : global cloud-resolving weather application**
- **Coding to use structure load/store instructions**
  - If the distance to the next element is known at compile time by passing the dimension size as an argument to the subroutine
  - the structure load/store instruction, which is more efficient than an indirect load/store
- **Loop fission for large loop body**
  - The part computing the physical processes has loops containing a large loop body. 30% performance improved.
- **Mixed-precision computation:**
  - For the solver of the fluid dynamics and some parts of the physical process.
  - promising optimization for memory-intensive computation because the demand for memory bandwidth is decreased. 60% performance improved.

Evaluation power mode: Boost mode & Eco mode

- **Power & Performance of STREAM using Eco mode**
  - The performance is almost the same as that in normal mode (24 threads hits 80% of peak memory bandwidth
  - The power increases upto 24 threads.
  - 15%-25% reduction comparing to that in normal mode.



- **Power & Performance of DGEMM (in Fujitsu Lib) using Boost mode**
  - Reach to 95% out of peak performance
  - The performance is 10% better than that in normal mode.
  - The power increases by 13.7%
  - The power-efficiency decreases by 3.3 %



Advances from the K computer

	K computer	Fugaku	ratio
# core	8	48	
Si tech. (nm)	45	7	
Core perf. (GFLOPS)	16	64(70)	4(4.4)
Chip(node) perf. (TFLOPS)	0.128	3.072 (3.379)	24 (26.4)
Memory BW (GB/s)	64	1024	
B/F (Bytes/FLOP)	0.5	0.33	
#node / rack	96	384	4
#node/system	82,944	158,976	

System perf.(DP PFLOPS)
(SP PFLOPS)
	10.6	488 (537)
		977(1070)
			42.3(52.2)
			84.6(104.4)

Si Tech

SVE

CMG&Si Tech

HBM

More than 7.6 M General-purpose cores!

- SVE increases core performance
- Silicon tech. and scalable architecture (CMG) to increase node performance
- HBM enables high bandwidth

Value in blankets
Indicate the number
At boost mode (2.2GHz

Conclusion

Concluding remarks

- **We are now sure to achieve 3 KPIs**
  - Power-efficiency ⇒ Eco-mode and 1st in Green500 at SC19!
  - Effective Performance of applications. ⇒ 2 target apps (GENESIS and NICAM+LETF) achieved 100 times faster than that of K computer.
  - Ease-of-use ⇒ easy for porting OpenMP+MPI programs without any accelerator programming.
- **Retrospective comments**
  - Support for AI/ML workload: Although AI/ML topics were not included as a target application, the SVE has a SIMD instruction for half-precision floating-point operations.
  - Small HBM memory
  - Decision of O3 resources: It was difficult to decide the O3 resources because the compiler was immature for a new Arm instruction set.
  - Future "disruptive" architecture: For future systems beyond exascale, a more disruptive architecture, such as accelerators and specialized hardware, would be required