

ポスト「京」の開発状況

理化学研究所
計算科学研究機構
石川 裕

第8回「学際計算科学による新たな知の発見・統合・創出」シンポジウム,
筑波大学、2016年10月17日

16:50 – 17:20

Outline of Talk

- **An Overview of FLAGSHIP 2020**
- **An Overview of post K system**
- **System Software**
- **Concluding Remarks**

An Overview of Flagship 2020 project

- Developing the next Japanese flagship computer, so-called “post K”



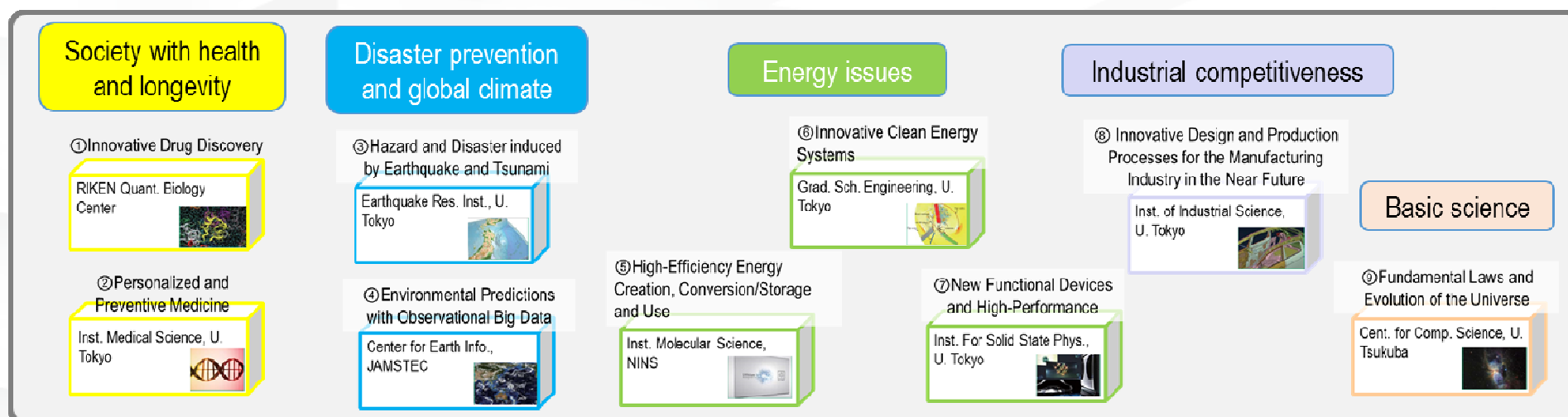
Vendor partner



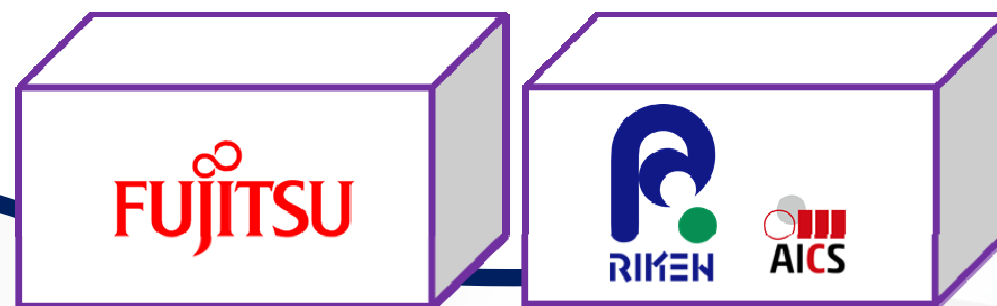
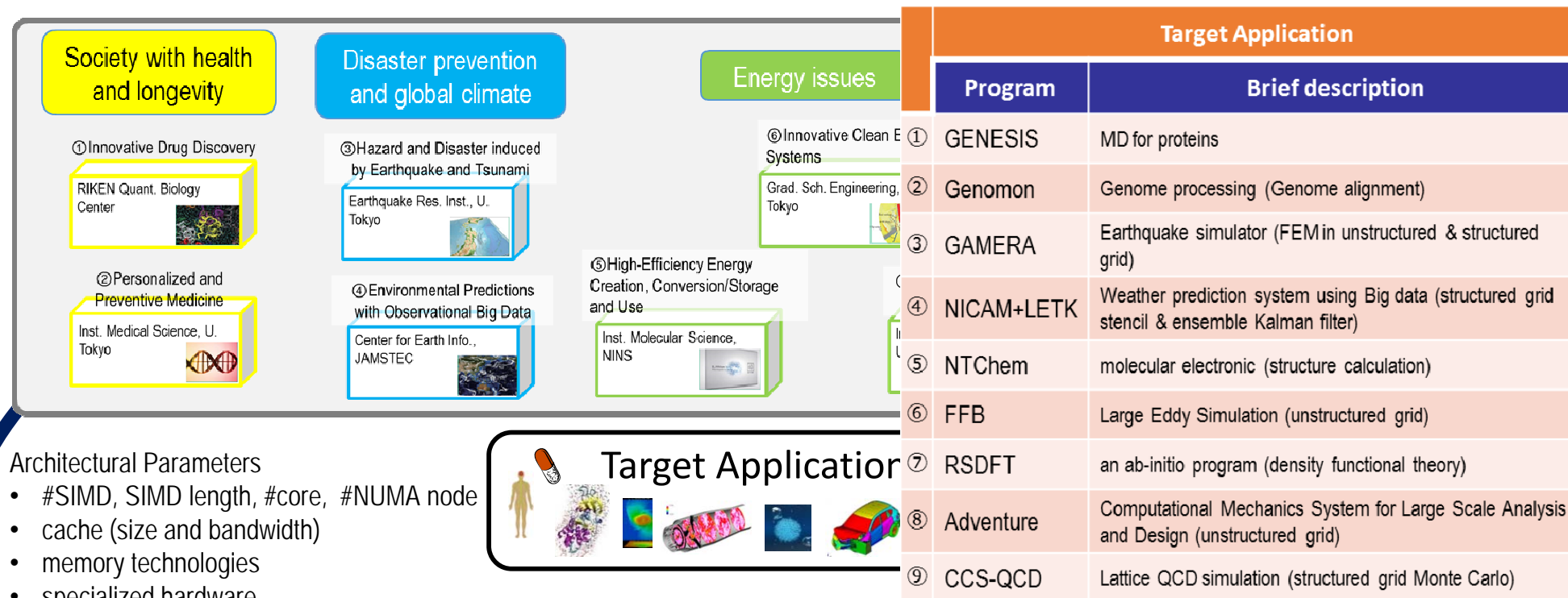
- Developing a wide range of application codes, to run on the “post K”, to solve major social and science issues



The Japanese government selected 9 social & scientific priority issues and their R&D organizations.

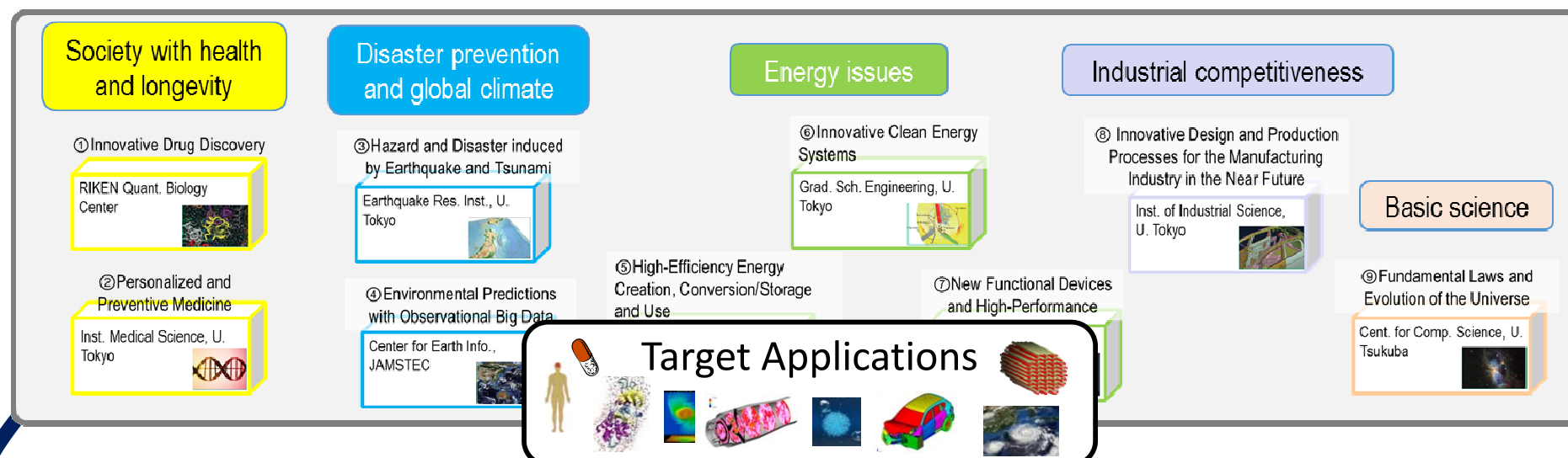


Co-design



Target Applications' Characteristics

Target Application			
	Program	Brief description	Co-design
①	GENESIS	MD for proteins	Collective comm. (all-to-all), Floating point perf (FPP)
②	Genomon	Genome processing (Genome alignment)	File I/O, Integer Perf.
③	GAMERA	Earthquake simulator (FEM in unstructured & structured grid)	Comm., Memory bandwidth
④	NICAM+LETK	Weather prediction system using Big data (structured grid stencil & ensemble Kalman filter)	Comm., Memory bandwidth, File I/O, SIMD
⑤	NTChem	molecular electronic (structure calculation)	Collective comm. (all-to-all, allreduce), FPP, SIMD,
⑥	FFB	Large Eddy Simulation (unstructured grid)	Comm., Memory bandwidth,
⑦	RSDFT	an ab-initio program (density functional theory)	Collective comm. (bcast), FFP
⑧	Adventure	Computational Mechanics System for Large Scale Analysis and Design (unstructured grid)	Comm., Memory bandwidth, SIMD
⑨	CCS-QCD	Lattice QCD simulation (structured grid Monte Carlo)	Comm., Memory bandwidth, Collective comm. (allreduce)



Architectural Parameters

- #SIMD, SIMD length, #core,
- cache (size and bandwidth)
- memory technologies
- specialized hardware
- Interconnect
- I/O network

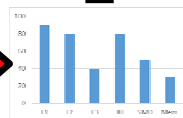
- ❑ Mutual understanding both computer architecture/system software and applications
- ❑ Looking at performance predictions
- ❑ Finding out the best solution with constraints, e.g., power consumption, budget, and space

Prediction of node-level performance

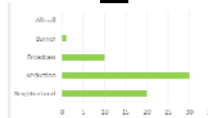


Profiling applications, e.g., cache misses and execution unit usages

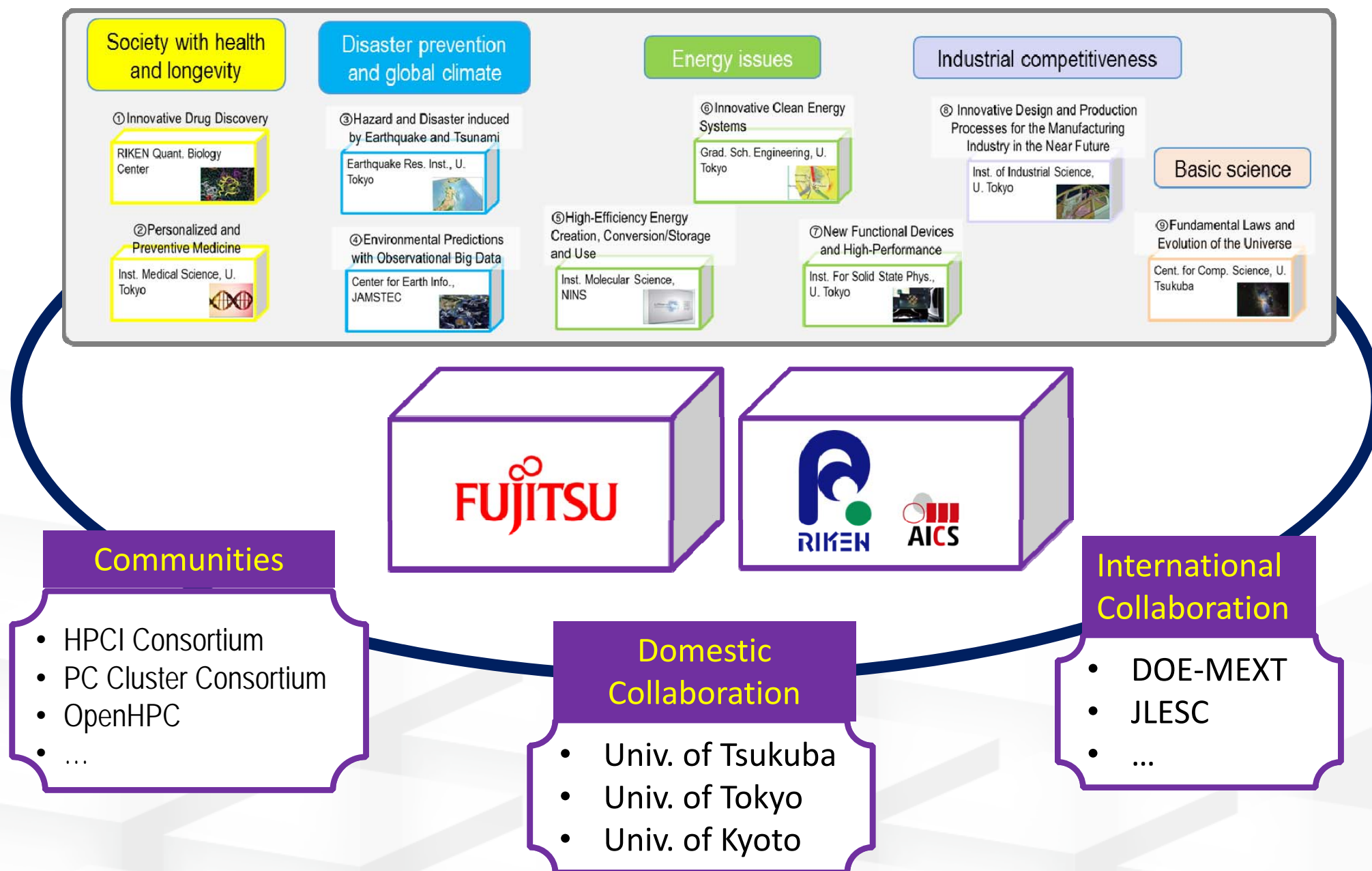
Prediction Tool



Prediction of scalability (Communication cost)



R&D Organization



Outline of Talk

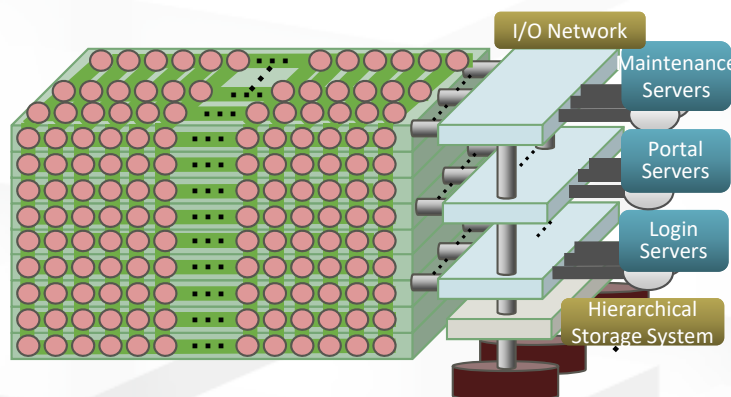


- **An Overview of FLAGSHIP 2020**
- **An Overview of post K system**
- **System Software**
- **Concluding Remarks**

An Overview of post K

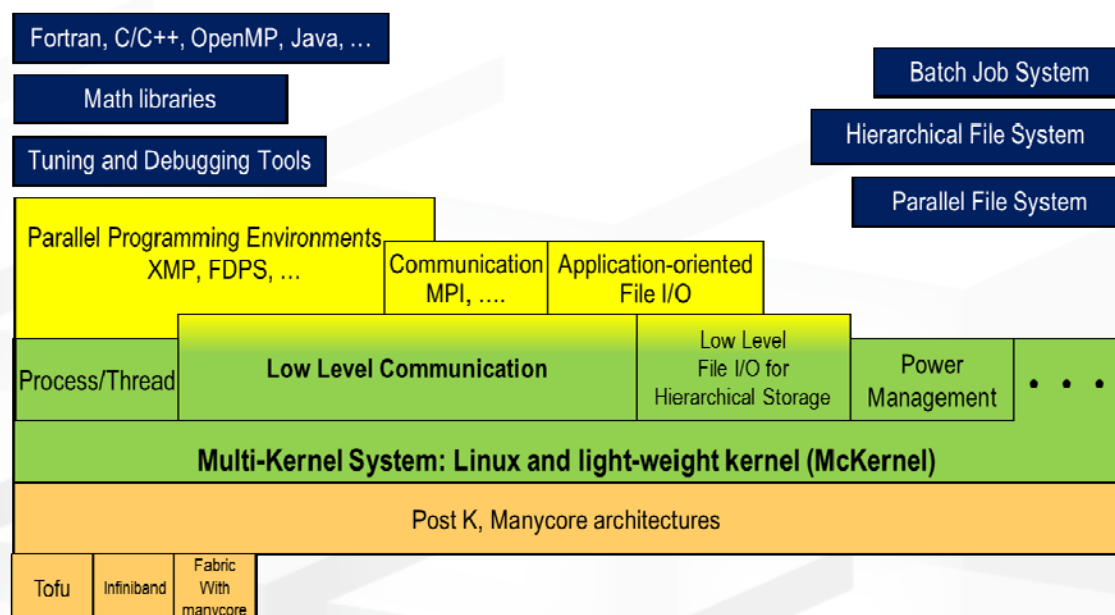
• Hardware

- Manycore architecture
- 6D mesh/torus Interconnect
- 3-level hierarchical storage system
 - Silicon Disk
 - Magnetic Disk
 - Storage for archive



• System Software

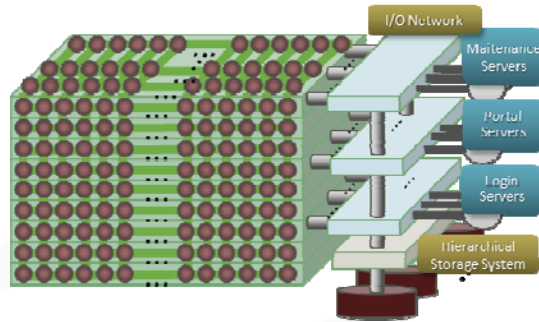
- Multi-Kernel: Linux with Light-weight Kernel
- File I/O middleware for 3-level hierarchical storage system and application
- Application-oriented file I/O middleware
- MPI+OpenMP programming environment
- Highly productive programming language and libraries



What we have done

- **Hardware**

- Instruction set architecture

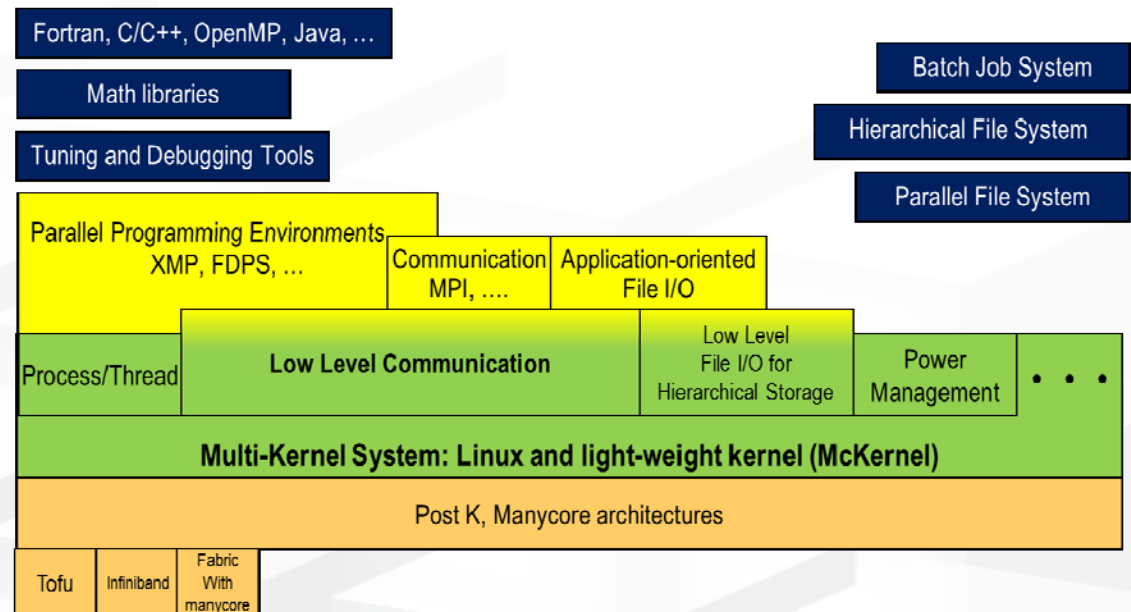


Continue to design

- Node architecture
- System configuration
- Storage system

- **Software**

- OS functional design
- Communication functional design
- File I/O functional design
- Programming languages
- Mathematical libraries





- **ARM V8 HPC Extension**

- Fujitsu is a lead partner of ARM HPC extension development
- Detailed features were announced at Hot Chips 28 - 2016
<http://www.hotchips.org/program/> SVE (Scalable Vector Extension)
Mon 8/22 Day1 9:45AM GPUs & HPCs
ARMv8-A Next Generation Vector Architecture for HPC

- **Fujitsu's inheritances**

- FMA
- Math acceleration primitives
- Inter core barrier
- Sector cache
- Hardware prefetch assist

Post-K: Fujitsu HPC CPU to Support ARM v8  

Post-K fully utilizes Fujitsu's proven supercomputer microarchitecture

Fujitsu, as a "lead partner" of ARM HPC extension development, is working to realize an ARM Powered® supercomputer w/ high application performance

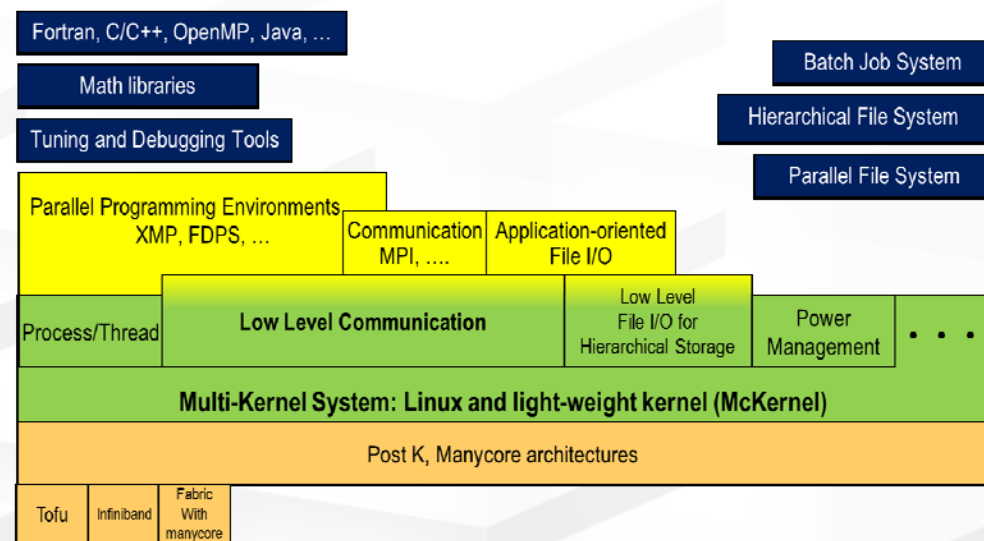
ARM v8 brings out the real strength of Fujitsu's microarchitecture

HPC apps acceleration feature	Post-K	FX100	FX10	K computer
FMA: Floating Multiply and Add	✓	✓	✓	✓
Math. acceleration primitives*	✓Enhanced	✓Enhanced	✓	✓
Inter core barrier	✓	✓	✓	✓
Sector cache	✓Enhanced	✓Enhanced	✓	✓
Hardware prefetch assist	✓Enhanced	✓Enhanced	✓	✓
Tofu interconnect	✓Integrated	✓Integrated	✓	✓

* Mathematical acceleration primitives include trigonometric functions, sine & cosines, and exponential function

Outline of Talk

- An Overview of FLAGSHIP 2020
- An Overview of post K system
- **System Software**
 - Multi-Kernel: Linux with Light-weight Kernel
 - File I/O middleware for 3-level hierarchical storage system and application
 - Application-oriented file I/O middleware
 - MPI+OpenMP programming environment
 - Highly productive programming language and libraries
- Concluding Remarks



OS Kernel

• Requirements of OS Kernel targeting high-end HPC

- Noiseless execution environment for bulk-synchronous applications
- Ability to easily adapt to new/future system architectures
 - E.g.: manycore CPUs, heterogenous core architectures, deep memory hierarchy, etc.
 - New process/thread management, memory management, ...
- Ability to adapt to new workloads
 - Big-Data, in-situ processing
 - Support data movement
 - Optimize data movement

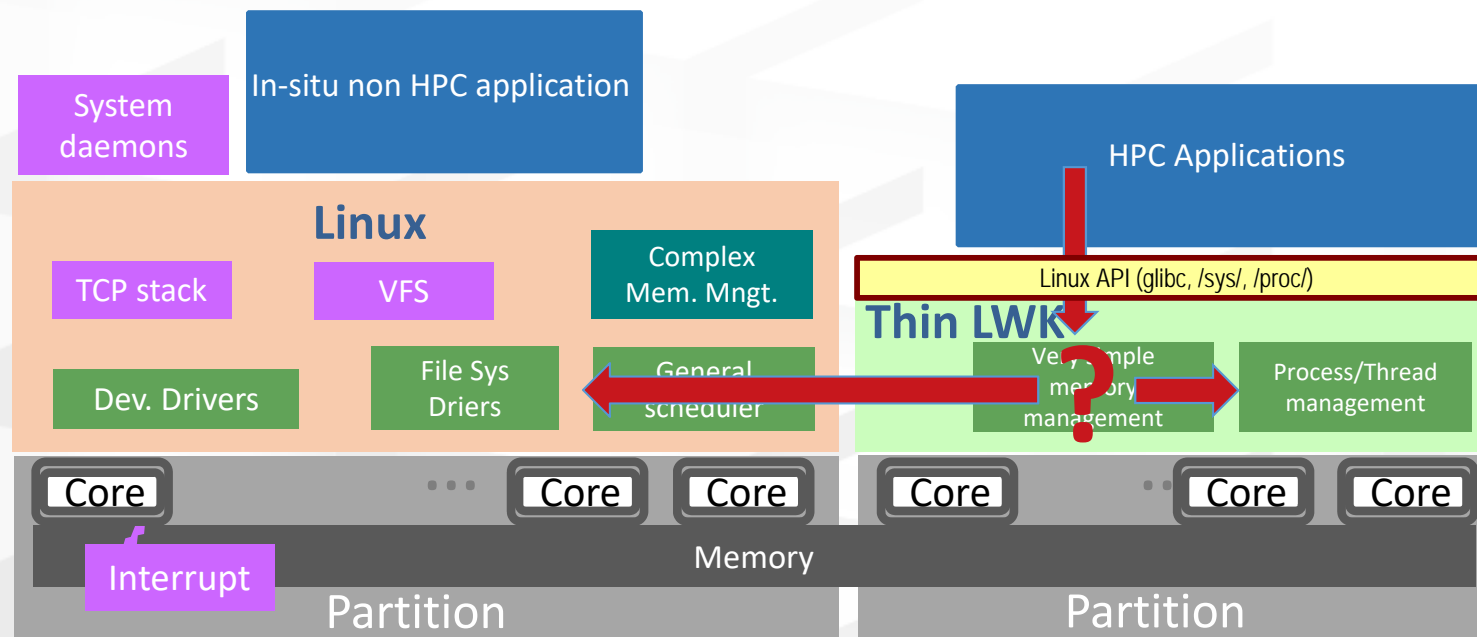
Our Approach:
Linux with Light-Weight Kernel

	Approach	Pros.	Cons.
Full-Weight Kernel (FWK) e.g. Linux	Disabling, removing, tuning, reimplementation, and adding new features	Large community support results in rapid new hardware adaptation	<ul style="list-style-type: none"> • Hard to implement a new feature if the original mechanism is conflicted with the new feature • Hard to follow the latest kernel distribution due to local large modifications
Light-Weight Kernel (LWK)	Implementation from scratch and adding new features	Easy to extend it because of small in terms of logic and code size	<ul style="list-style-type: none"> • Applications, running on FWK, cannot run always in LWK • Small community maintenance limits rapid growth • Lack of device drivers

McKernel developed at RIKEN

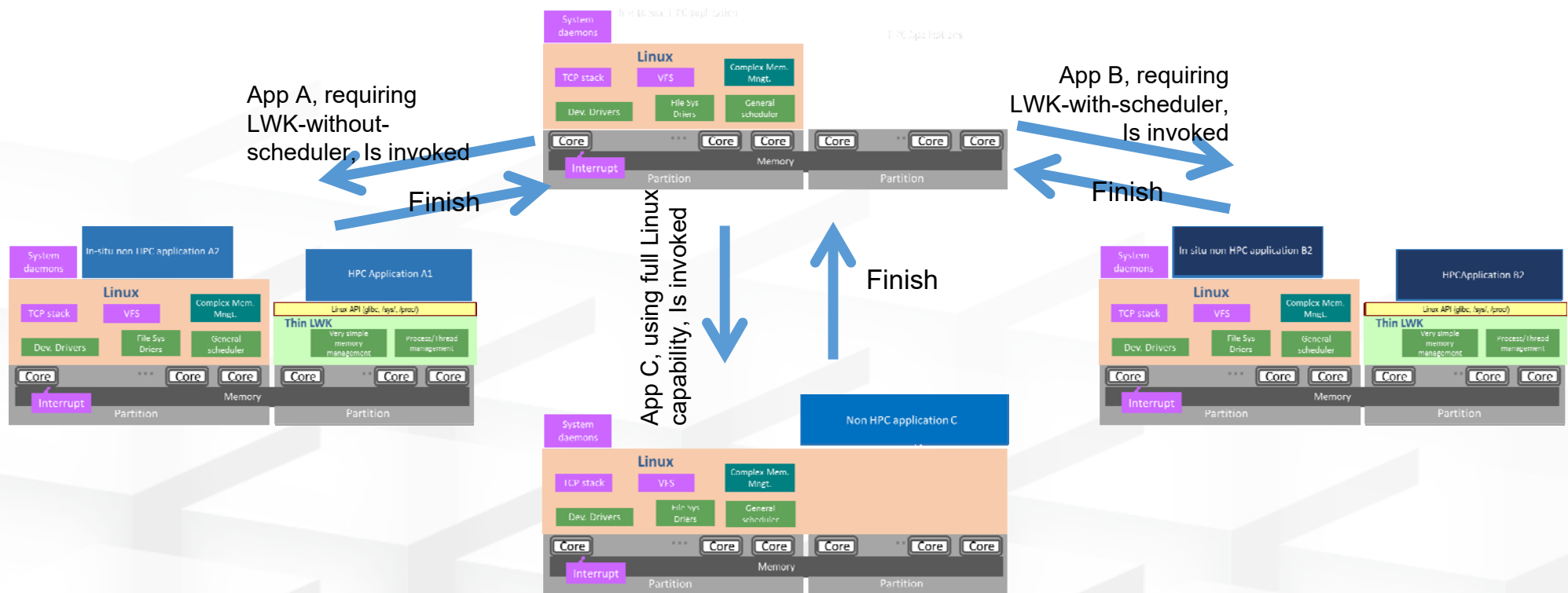
- Partition resources (CPU cores, memory)
- Full Linux kernel on some cores
 - System daemons and in-situ non HPC applications
 - Device drivers
- Light-weight kernel(LWK), McKernel on other cores
 - HPC applications
- McKernel is loadable module of Linux
- McKernel supports Linux API
- McKernel runs on
 - Intel Xeon and Xeon phi
 - Fujitsu FX10

McKernel is deployed to the Oakforest-PACS supercomputer, 25 PF in peak, at JCAHPC organized by U. of Tsukuba and U. of Tokyo



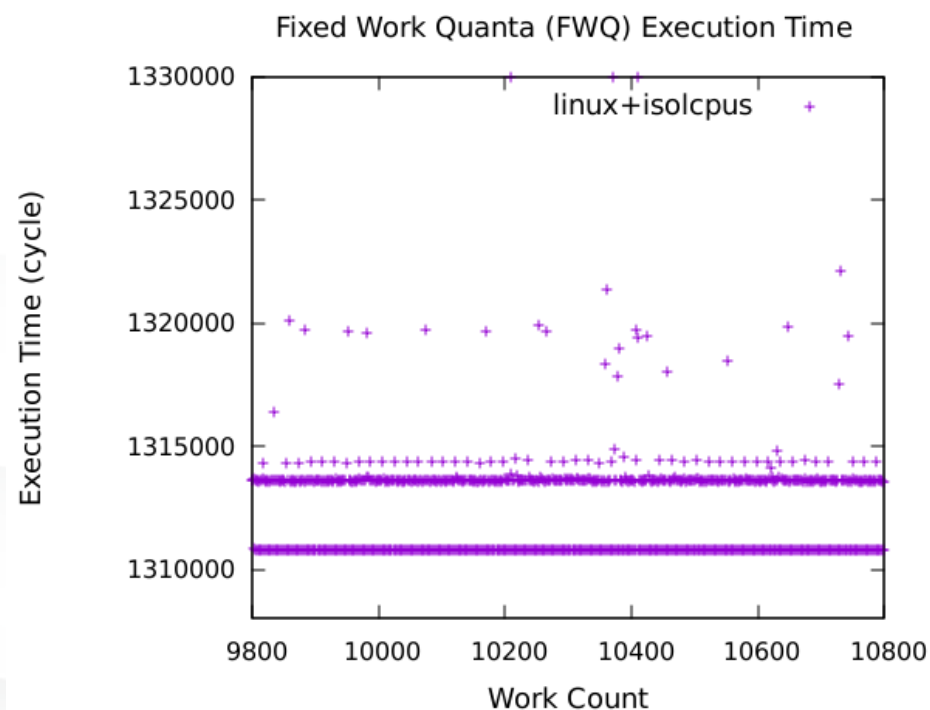
How to deploy McKernel

- Linux Kernel + Loadable LWK, McKernel
 - Linux Kernel is resident, and daemons for job scheduler and etc. run on Linux
 - McKernel is dynamically reloaded (rebooted) for each application

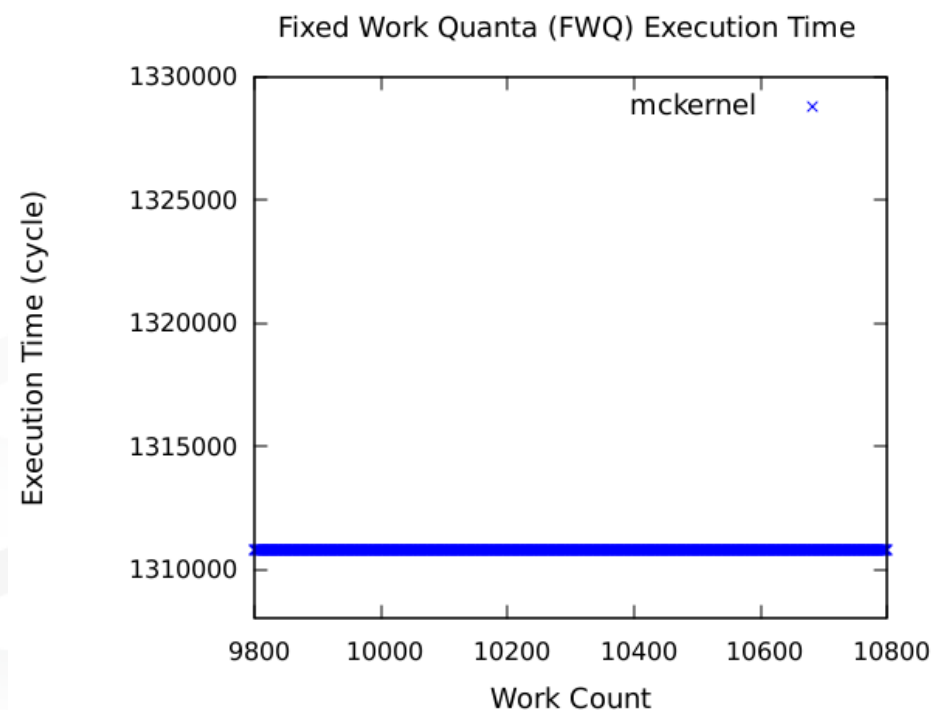


Results of FWQ (Fixed Work Quanta)

Linux with isolcpus



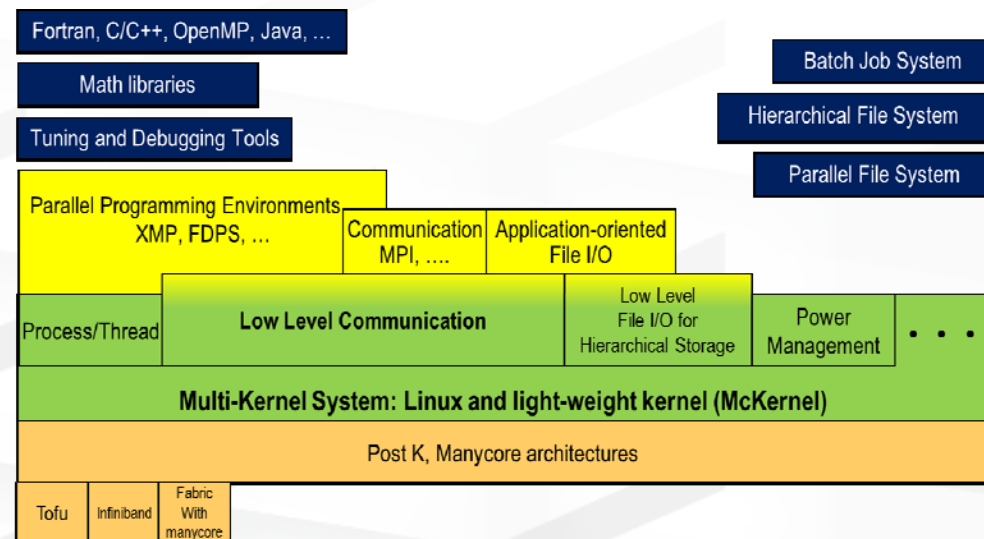
McKernel



<https://asc.llnl.gov/sequoia/benchmarks>

Outline of Talk

- An Overview of FLAGSHIP 2020
- An Overview of post K system
- **System Software**
 - Multi-Kernel: Linux with Light-weight Kernel
 - File I/O middleware for 3-level hierarchical storage system and application
 - Application-oriented file I/O middleware
 - MPI+OpenMP programming environment
 - Highly productive programming language and libraries
- Concluding Remarks

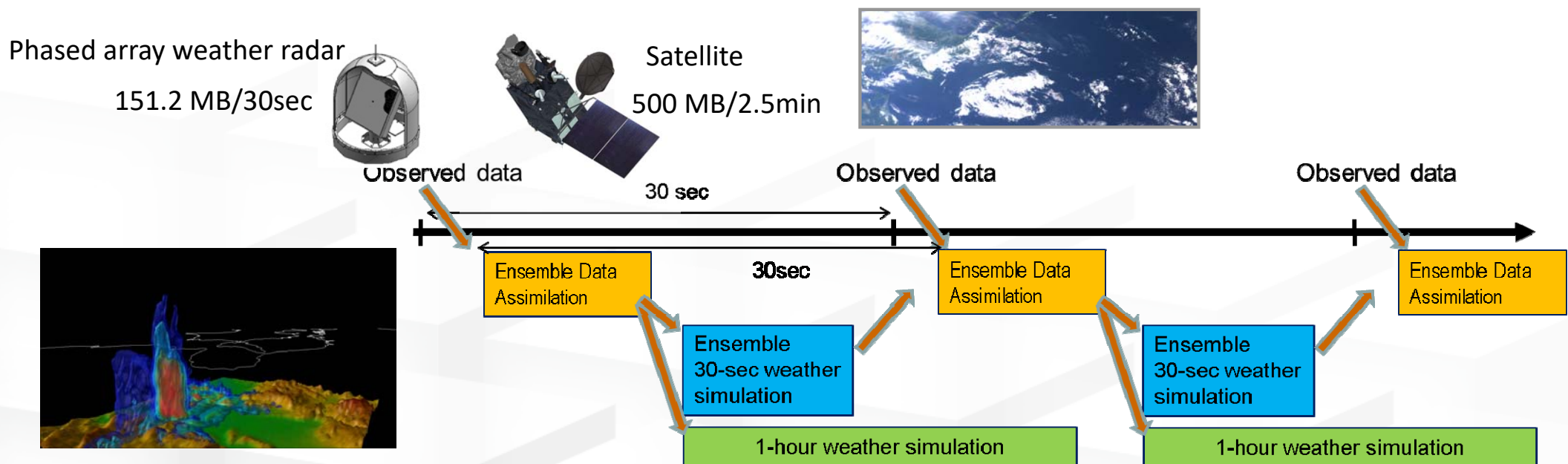


File I/O for Big data

PI: Takemasa Miyoshi, RIKEN AICS

“Innovating Big Data Assimilation technology for revolutionizing very-short-range severe weather prediction”

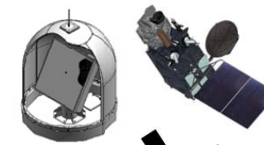
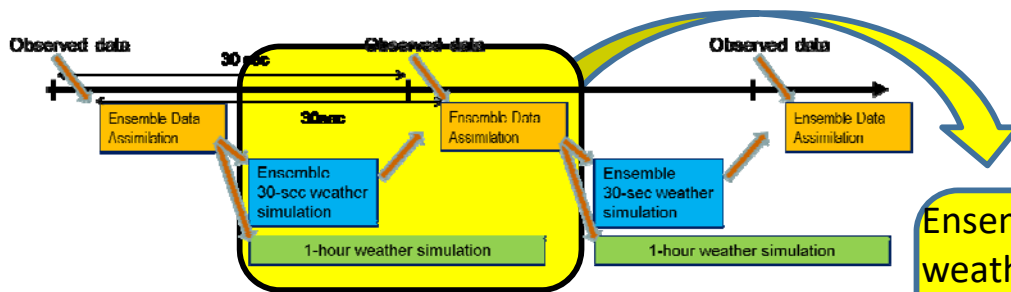
An innovative 30-second super-rapid update numerical weather prediction system for 30-minute/1-hour severe weather forecasting will be developed, aiding disaster prevention and mitigation, as well as bringing a scientific breakthrough in meteorology.



To meet real-timeness, 30 second responsibility, data exchange between data assimilation and weather simulations must be fast as much as possible

Rain particle

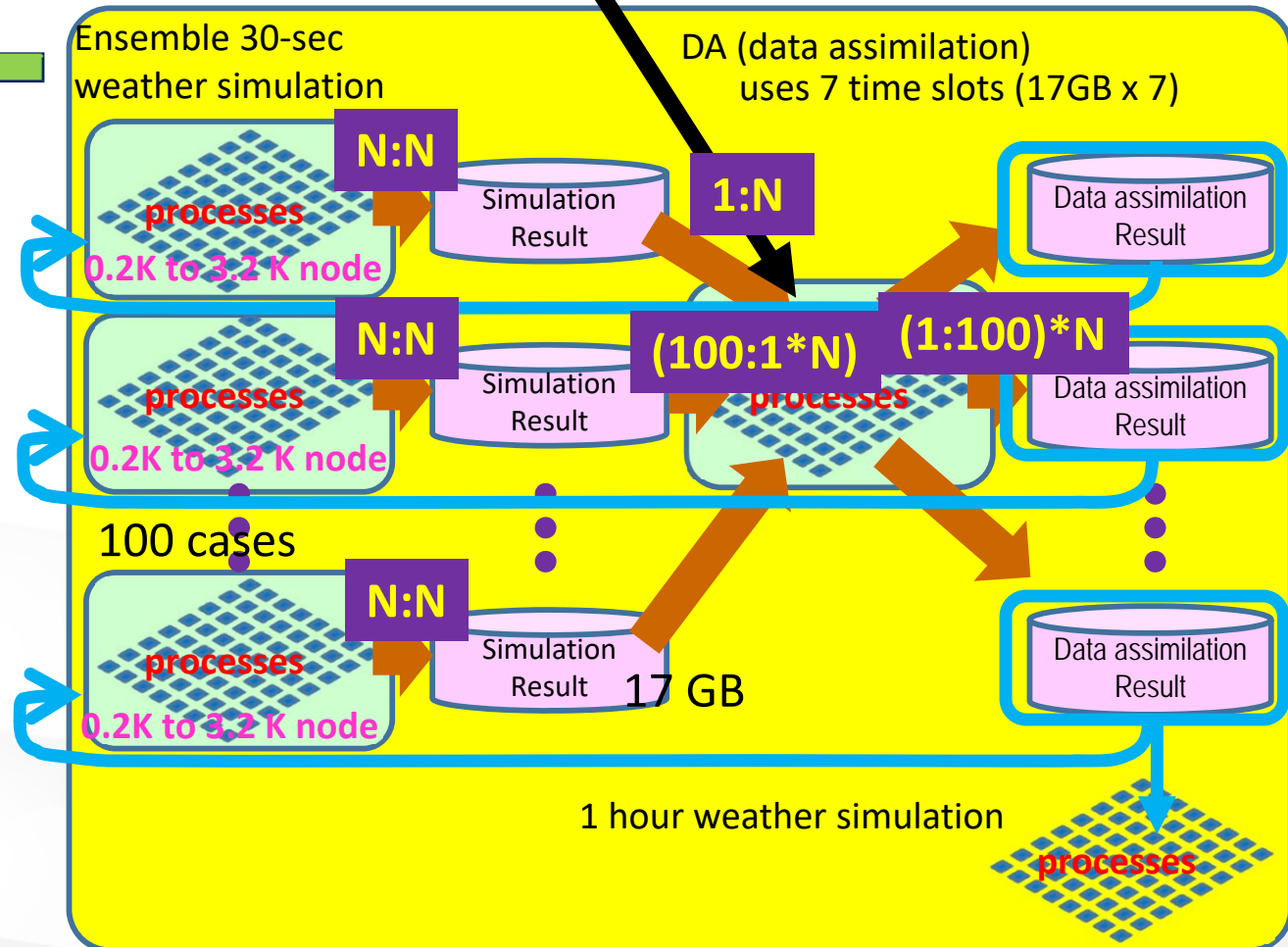
File I/O patterns in Ensemble simulation and data assimilation



Observed data via Internet

Every 30 second

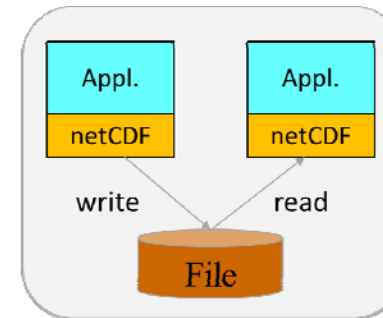
1. Each weather simulation generates 17 GB data. 1.7 TB in total for 100 cases
2. Data assimilation code reads all results (17GB) and observed data (-- 1GB), and outputs assimilated data for 100 cases. That is, 1.7 TB in total for 100cases



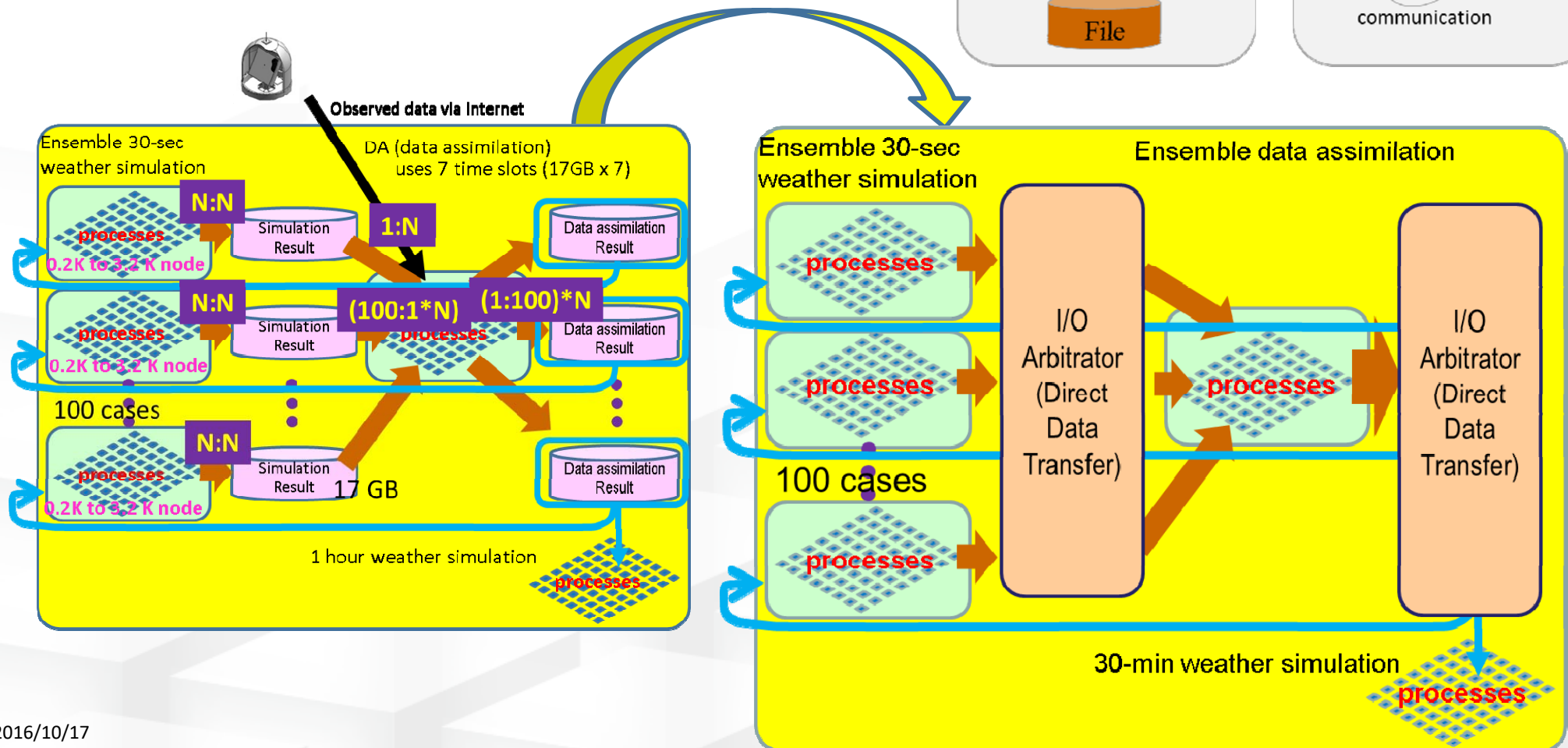
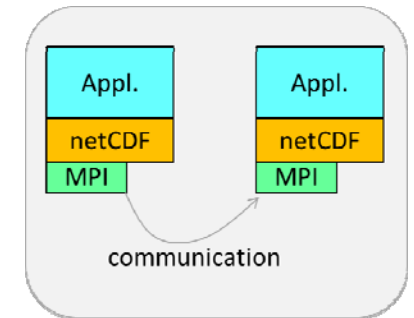
Approach: I/O Arbitrator

- Keeping the netCDF file I/O API
- Introducing additional API in order to realize direct data transfer without storing data into storage
 - E.g., asynchronous I/O

Original



Proposal



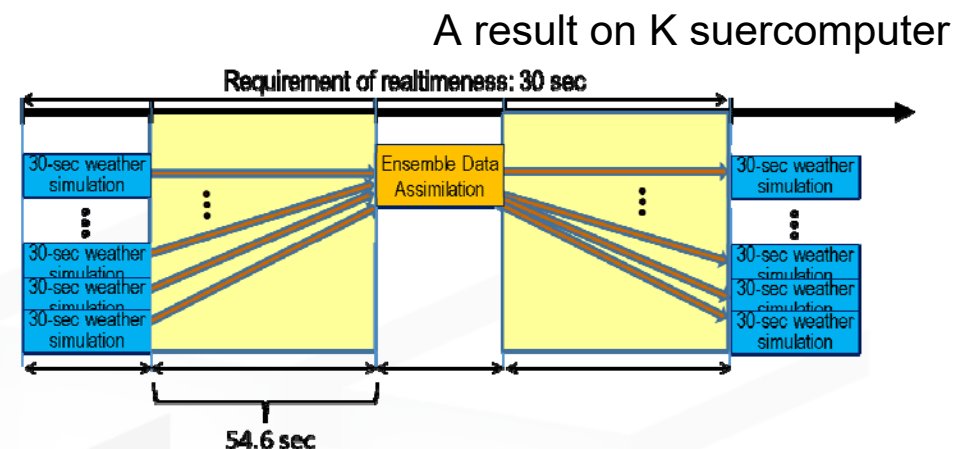
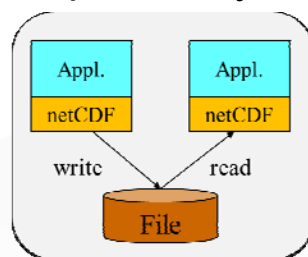
Prototype System Evaluation at RIKEN AICS

Case Study

- There are totally 11 variables, and each variable has $384 * 288 * 36$ grid data (double precision). The size of transfer data between 100-case simulations and data assimilation process is about 533GB. 4,800 nodes are used.

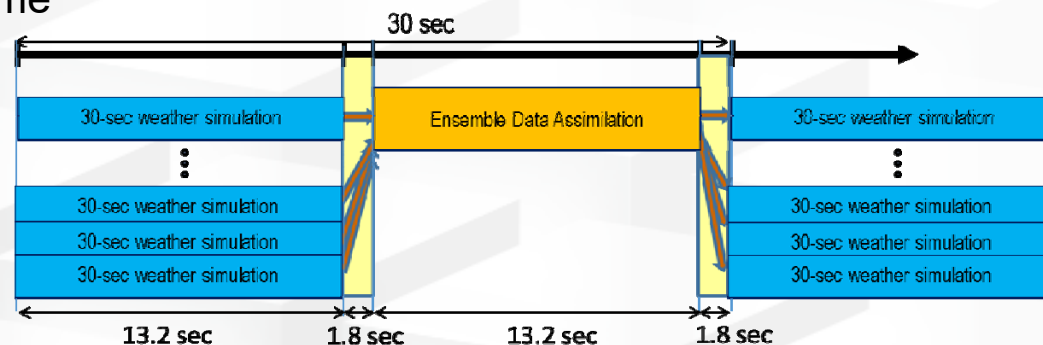
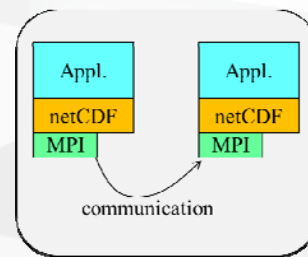
netCDF/File I/O: 54.6 sec

Cannot realize 30 second responsibility !



netCDF/MPI: 1.8 sec

Simulator and DA have 26.4 sec execution time



XcalableMP(XMP) <http://www.xcalablemp.org>

- **What's XcalableMP (XMP for short)?**

- A PGAS programming model and language for distributed memory , proposed by **XMP Spec WG**
- XMP Spec WG is a special interest group to design and draft the specification of XcalableMP language. It is now organized under **PC Cluster Consortium**, Japan. Mainly active in Japan, but open for everybody.

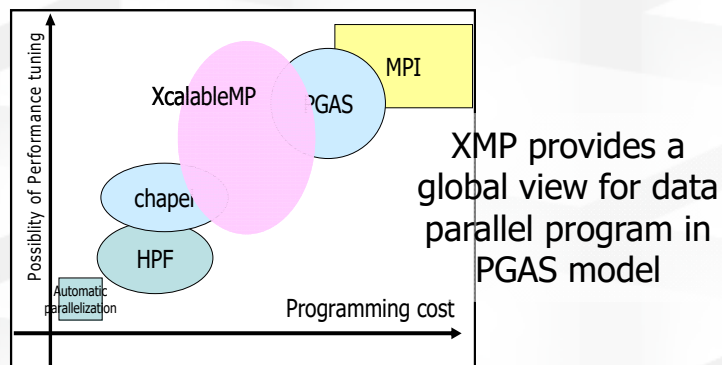
- **Project status (as of June 2016)**

- XMP Spec **Version 1.2.1** is available at XMP site. new features: mixed OpenMP and OpenACC , libraries for collective communications.
- Reference implementation by U. Tsukuba and Riken AICS: **Version 1.0 (C and Fortran90)** is available for PC clusters, Cray XT and K computer. Source-to- Source compiler to code with the runtime on top of MPI and GasNet.

- **HPCC class 2 Winner 2013. 2014**

- **Language Features**

- **Directive-based language extensions** for Fortran and C for PGAS model
- **Global view programming** with global-view distributed data structures for data parallelism
 - SPMD execution model as MPI
 - pragmas for data distribution of global array.
 - Work mapping constructs to map works and iteration with affinity to data explicitly.
 - Rich communication and sync directives such as "gmove" and "shadow".
 - Many concepts are inherited from HPF
- **Co-array feature** of CAF is adopted as a part of the language spec for **local view programming** (also defined in C).



Code example

```
int array[YMAX][XMAX];
```

```
#pragma xmp nodes p(4)
#pragma xmp template t(YMAX)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][*] to t(i)
```

data distribution

```
main(){
  int i, j, res;
  res = 0;
```

add to the serial code : incremental parallelization

```
#pragma xmp loop on t(i) reduction(+:res)
for(i = 0; i < 10; i++)
  for(j = 0; j < 10; j++){
    array[i][j] = func(i, j);
    res += array[i][j];
  }
}
```

work sharing and data synchronization

Concluding Remarks

- **The system software stack for Post K is being designed and implemented with the leverage of international collaborations**
 - The software stack developed at RIKEN is Open source
 - It also runs on Intel Xeon and Xeon phi
 - RIKEN would like to contribute to OpenHPC

