Prof. Mitsuhisa Sato received the M.S. degree and the Ph.D. degree in information science from the University of Tokyo in 1984 and 1990. He was a senior researcher at Electrotechnical Laboratory from 1991 to 1996. From 2001, he was a professor of Graduate School of Systems and Information Engineering, University of Tsukuba. He has been working as a director of Center for computational sciences, University of Tsukuba from 2007 to 2013. Since October 2010, he is appointed to the research team leader of programming environment research team in Advanced Institute of Computational Science (AICS), RIKEN. He is a Professor (Cooperative Graduate School Program) and Professor Emeritus of University of Tsukuba.

FLAGSHIP 2020 Project: Challenges for Post-petascale and Exascale Computing
FLAGSHIP 2020 project: Challenges for post-petascale and exascale computing

Mitsuhisa Sato  Team Leader of Architecture Development Team

FLAGSHIP 2020 project
RIKEN Advance Institute of Computational Science (AICS)

SC16 HPC Connection WS, 16th Nov, 2016
Outline of Talk

- An Overview of FLAGSHIP 2020 project
- An Overview of post K system
- System Software
- Concluding Remarks
Towards the Next Flagship Machine

<table>
<thead>
<tr>
<th>PostT2K</th>
<th>PostK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch: Upscale Commodity Cluster Machine</td>
<td>Flagship Machine Arch: co-design by RIKEN and Vendor</td>
</tr>
<tr>
<td>Soft: Technology Path-Forward Machine</td>
<td></td>
</tr>
<tr>
<td>Manycore architecture</td>
<td>Manycore architecture</td>
</tr>
<tr>
<td>O(10K) nodes</td>
<td>O(100K-1M) nodes</td>
</tr>
</tbody>
</table>

- Oakforest-PACS (PostT2K) is a production system operated by both Tsukuba and Tokyo
- The post K project is to design the next flagship system (pre-exascale) and deploy/install the system for services, around 2020
- the project was launched at 2014

RIKEN
9 Universities and National Laboratories

U. of Tsukuba
U. of Tokyo
Kyoto U.

T2K
9 Universities and National Laboratories

K Computer
FLAGSHIP Machine
RIKEN
An Overview of Flagship 2020 project

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

- 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:

- District with health and longevity
  - Innovative Drug Discovery
    - RIKEN Quant. Biology Center
  - Personalized and Preventive Medicine
    - Inst. Medical Science, U. Tokyo

- Disaster prevention and global climate
  - Hazard and Disaster induced by Earthquake and Tsunami
    - Earthquake Res. Inst., U. Tokyo
  - Environmental Predictions with Observational Big Data
    - Center for Earth Info., JAMSTEC

- Energy issues
  - High-Efficiency Energy Creation, Conversion/Storage and Use
    - Inst. Molecular Science, NINS
  - Innovative Clean Energy Systems
    - Grad. Sch. Engineering, U. Tokyo

- Industrial competitiveness
  - Innovative Design and Production Processes for the Manufacturing Industry in the Near Future
    - Inst. of Industrial Science, U. Tokyo
  - Fundamental Laws and Evolution of the Universe
    - Cent. for Comp. Science, U. Tsukuba
Co-design

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

Target Applications
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)
## Target Applications’ Characteristics

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

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

**Target Applications**
- 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

Communities
- HPCI Consortium
- PC Cluster Consortium
- OpenHPC
- ...

Domestic Collaboration
- Univ. of Tsukuba
- Univ. of Tokyo
- Univ. of Kyoto

International Collaboration
- DOE-MEXT
- JLESC
- ...

Society with health and longevity
- Innovative Drug Discovery
- RIKEN Quant. Biology Center
- Personalized and Preventive Medicine
- Inst. Medical Sciences, U. Tokyo

Disaster prevention and global climate
- Hazard and Disaster induced by Earthquake and Tsunami
- Earthquake Res. Inst., U Tokyo
- Environmental Predictions with Observational Big Data
- Center for Earth Info., JAMSTEC

Energy issues
- Innovative Clean Energy Systems
- Grad. Sch. Engineering, U. Tokyo
- High-Efficiency Energy Creation, Conversion/Storage and Use
- Inst. Molecular Science, NINS

Industrial competitiveness
- Innovative Design and Production Processes for the Manufacturing Industry in the Near Future
- Inst. of Industrial Science, U. Tokyo
- Fundamental Laws and Evolution of the Universe
- Cent. for Comp. Science, U. Tsukuba

Basic science
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

**MC-kernel: a lightweight Kernel for manycore**

**XcalableMP PGAS language**

**FPDS DSL**

---

For more information, visit [RIKEN AICS](https://aics.riken.jp/).
What we have done

- **Hardware**
  - Instruction set architecture
  - Continue to design:
    - Node architecture
    - System configuration
    - Storage system

- **Software**
  - OS functional design (done)
  - Communication functional design (done)
  - File I/O functional design (done)
  - Programming languages (under development)
  - Mathematical libraries (under development)
Instruction Set Architecture

- **ARM V8 with HPC Extension SVE**
  - Fujitsu is a lead partner of ARM HPC extension development
  - Detailed features were announced at Hot Chips 28 - 2016

http://www.hotchips.org/program/
Mon 8/22 Day1 9:45AM GPUs & HPCs
“ARMv8-A Next Generation Vector Architecture for HPC” SVE (Scalable Vector Extension)

- **Fujitsu’s additional support**
  - FMA
  - Math acceleration primitives
  - Inter-core hardware-supported 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
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 architectural state

- Scalable vector registers
  - Z0-Z31 extending NEON’s V0-V31
  - DP & SP floating-point
  - 64, 32, 16 & 8-bit integer

- Scalable predicate registers
  - P0-P7 lane masks for ld/st/arith
  - P8-P15 for predicate manipulation
  - FFR first fault register

- Scalable vector control registers
  - ZCR_ELx vector length (LEN=1..16)
  - Exception / privilege level EL1 to EL3
SVE example

**DAXPY (scalar)**

```c
// 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]
    fma   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)**

```c
// 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)
    ldldr  z0.d, p0/z, [x2] // p0:z0=bcast(*a)
    .loop:
    ldld  z1.d, p0/z, [x0,x4,lsl 3]  // p0:z1=x[i]
    ldld  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
    stdl  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
```

- Compact code for SVE as scalar loop
- OpenMP SIMD directive is expected to help the SVE programming
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 programing 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, heterogeneous core architectures, deep memory hierarchy, etc.
    - New process/thread management, memory management, ...
  - Ability to adapt to new/future application demand
    - Big-Data, in-situ applications
    - Support data flow from Internet devices to compute nodes
    - Optimize data movement

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pros.</th>
<th>Cons.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full-Weight Kernel (FWK)</strong></td>
<td>Disabling, removing, tuning, reimplementation, and adding new features</td>
<td>Large community support results in rapid new hardware adaptation</td>
</tr>
<tr>
<td>e.g. Linux</td>
<td></td>
<td>• Hard to implement a new feature if the original mechanism is conflicted with the new feature</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Hard to follow the latest kernel distribution due to local large modifications</td>
</tr>
<tr>
<td><strong>Light-Weight Kernel (LWK)</strong></td>
<td>Implementation from scratch and adding new features</td>
<td>Easy to extend it because of small in terms of logic and code size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Applications, running on FWK, cannot run always in LWK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Small community maintenance limits rapid growth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Lack of device drivers</td>
</tr>
</tbody>
</table>

Our Approach: Linux with Light-Weight Kernel
McKernel developed at RIKEN

- Enable 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
OS: McKernel

Results of FWQ (Fixed Work Quanta)

**Linux with isolcpus**

isolcpus — Isolate CPUs from the kernel scheduler.

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

```c
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)
main()
{
    int i, j, res;
    res = 0;
    #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];
} add to the serial code: incremental parallelization
```

Data distribution

Work sharing and data synchronization
Programming model: “MPI+X” for exascale?

- X is OpenMP!

- “MPI+Open” is now a standard programming for high-end systems.
  - I’d like to celebrate that OpenMP became “standard” in HPC programming

- Questions:
  - “MPI+OpenMP” is still a main programming model for exa-scale?
Question

- What happens when executing code using all cores in manycore processors like this?

```c
MPI_recv ...
#pragma omp parallel for
for ( ... ; ... ; ... ) {
    ... computations ...
}
MPI_send ...
```

- What are solutions?
  - MPI+OpenMP runs on divided small “NUMA domains” rather than all cores?

- Data comes into “main shared memory”
- Cost for “fork” become large
- Data must be taken from Main memory
- Cost for “ barrier” become large
- MPI must collect data from each core to send
XcalableMP 2.0

**Specification v 1.2:**
- Support for Multicore: hybrid XMP and OpenMP is defined.
- Dynamic allocation of distributed array

**A set of spec in version 1 is now “converged”. New functions should be discussed for version 2.**

**Main topics for XcalableMP 2.0: Support for manycore**
- Multitasking with integrations of PGAS model
- Synchronization models for dataflow/multitasking executions
- Proposal: tasklet directive
  - Similar to OpenMP task directive
  - Including inter-node communication on PGAS

```c
int A[100], B[25];
#pragma xmp nodes P()
#pragma xmp template T(0:99)
#pragma xmp distribute T(block) onto P
#pragma xmp align A[i] with T(i)
/ ... /
#pragma xmp tasklet out(A[0:25], T(75:99))
   taskA();
#pragma xmp tasklet in(B, T(0:24)) out(A[75:25])
   taskB();
#pragma xmp taskletwait
```
Concluding Remarks

- We are very excited and believe that ARM SVE will deliver high-performance and flexible SIMD-vectorization to our “post-K” manycore processor.
- We think establishment of “eco-system” for ARM SVE in high-end HPC area very important, and are willing to do collaborations with partners who are interested in ARM SVE, as well as ARM.
- Extend “mobile” eco-system of ARM to HPC!!!
- The system software for Post K is being designed and implemented with the leverage of international collaborations
- The system software developed at RIKEN is Open source
- RIKEN will contribute to OpenHPC project