FUJITSU

shaping tomorrow with you

Toward Building up ARM HPC Ecosystem

Shinji Sumimoto, Ph.D.
Next Generation Technical Computing Unit
FUJITSU LIMITED

Sept. 12th, 2017

Outline

FUJITSU

- Fujitsu's Super computer development history and Post-K Processor Overview
- Compiler Development for ARMv8 with SVE
- Towards building up ARM HPC Ecosystem

FUJITSU

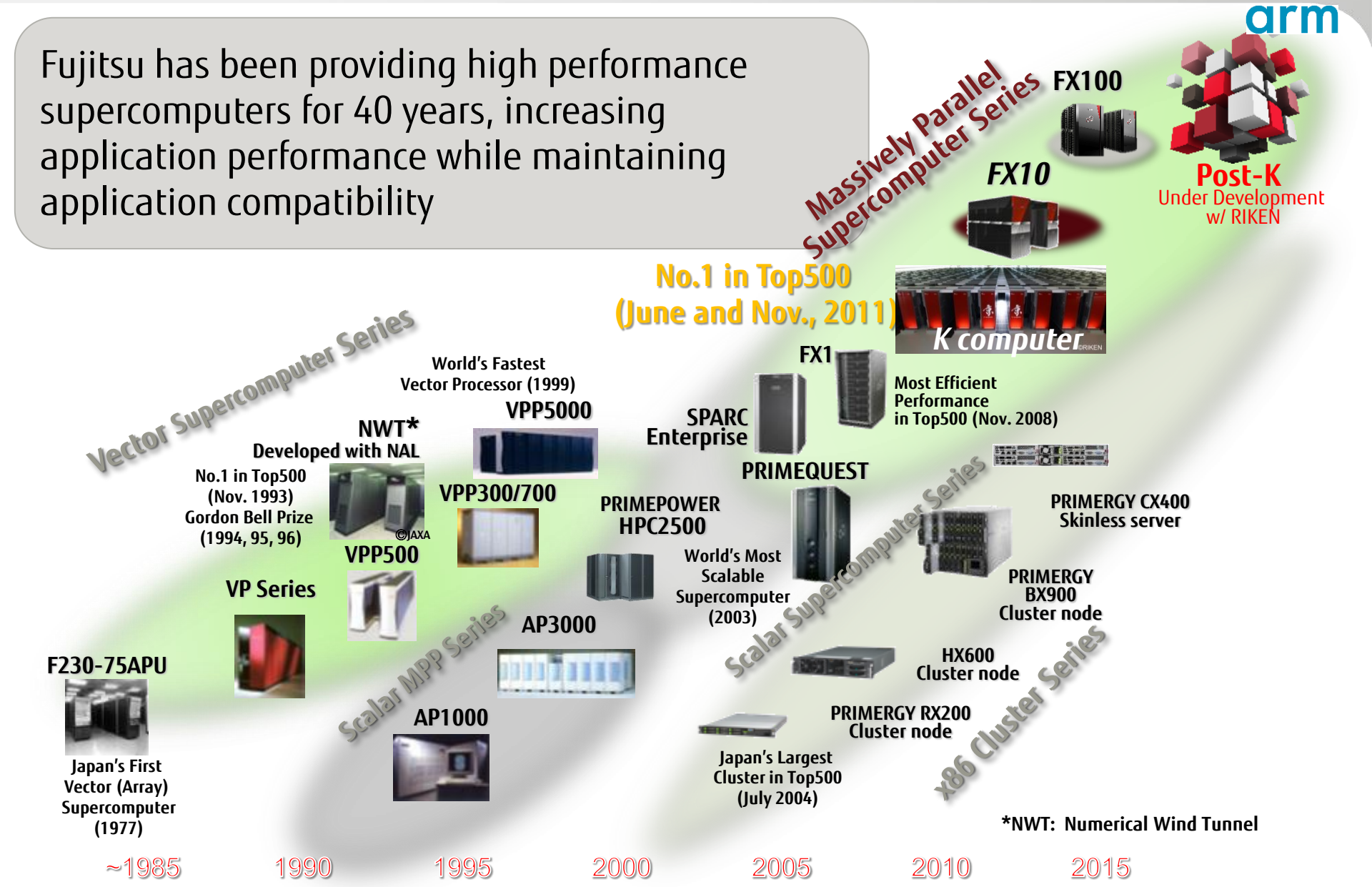
Fujitsu's Super computer development history and Post-K Processor Overview

Fujitsu Supercomputers

FUJITSU

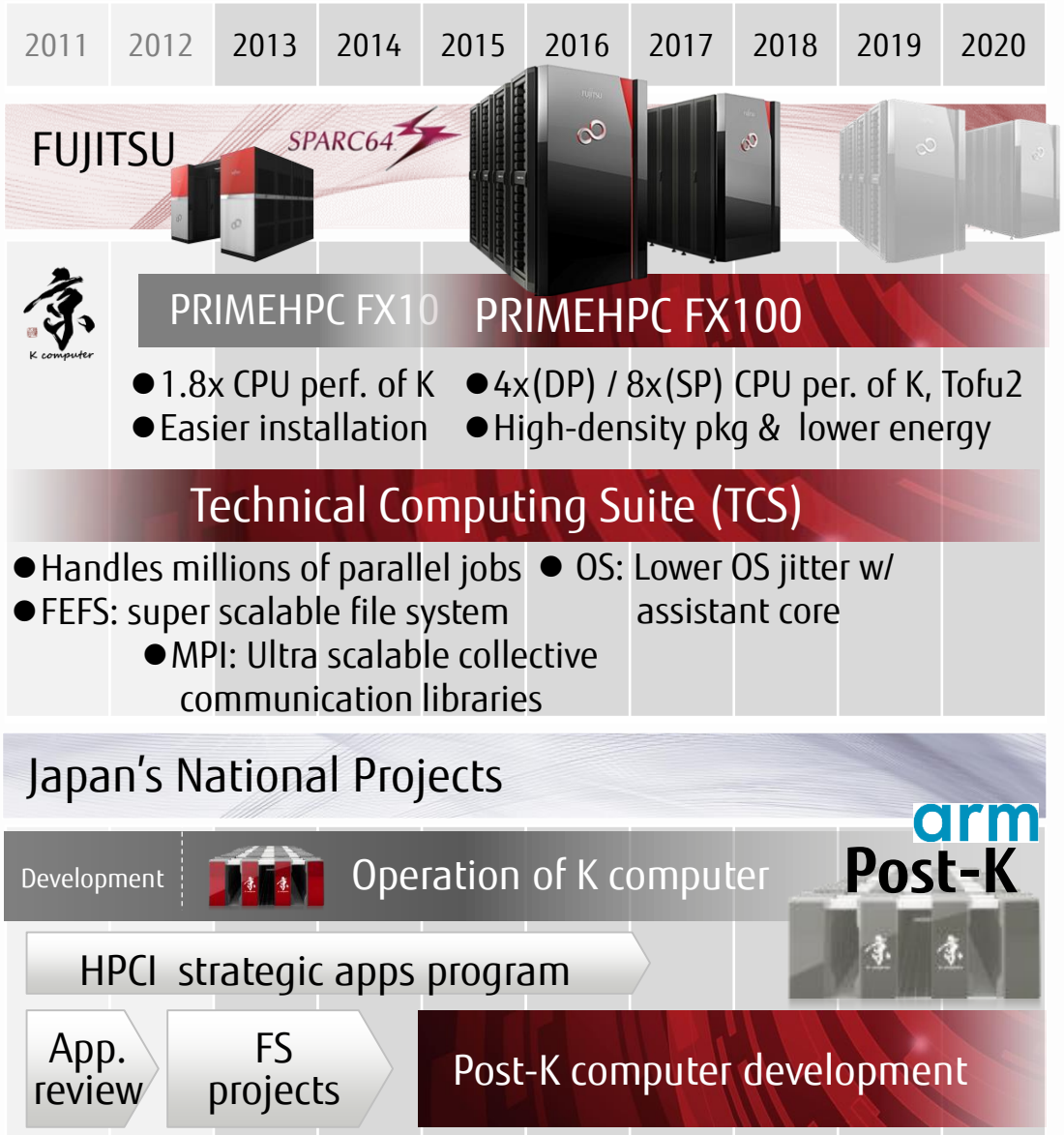
arm

Fujitsu has been providing high performance supercomputers for 40 years, increasing application performance while maintaining application compatibility



*NWT: Numerical Wind Tunnel

FUJITSU high-end supercomputers development



K computer and PRIMEHPC FX10/FX100 in operation

The CPU and interconnect of FX10/FX100 inherit the K computer architectural concept, featuring state-of-the-art technologies

System software TCS supports FUJITSU supercomputer with originally introduced technologies

Many applications are currently running and being developed for science and various industries

Post-K supercomputer

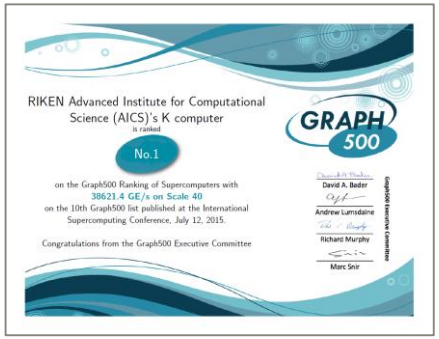
RIKEN and FUJITSU are working together to provide a successor of K computer with application R&D teams using co-design approach

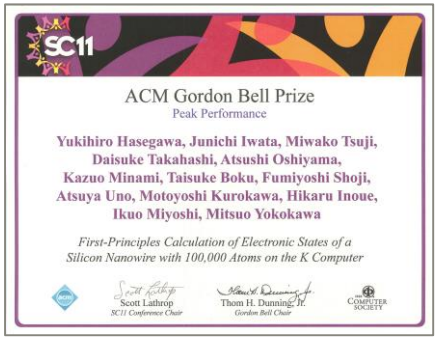
Achievements with the K computer

FUJITSU

- Prestigious Benchmark Awards
  - TOP500: 10.5Pflops, 93% efficiency — No. 8
  - HPCG: 602Tflops, 5.3% efficiency — No. 1
  - Graph500: 38.6TTEPS — No. 1
  - HPC Challenge Class 1: No.1 in all categories — No. 1
    (1) Global HPL, (2) Global Random Access, (3) EP STREAM, (4) Global FFT
- Gordon Bell Prize Awards
  - "First-Principles Calculation of Electron States of a Silicon Nanowire with 100,000 Atoms on the K computer" (2011)
  - "4.45 Pflops Astrophysical N-Body Simulation on K Computer – The Gravitational Trillion-Body Problem" (2012)
  - "Simulations of Below-Ground Dynamics of Fungi: 1.184 Pflops Attained by Automated Generation and Autotuning of Temporal Blocking Codes" (2016 finalist)

at ISC17
6 years from the initial delivery





Post-K Hardware Features

FUJITSU

- Fujitsu CPU cores support the ARM SVE instruction set architecture
- Fujitsu CPU & Tofu maintain the programming models and provide high application performance
- FP16 (“giant vector throughput”) for supercomputers

	Functions & architecture	 Post-K	 FX100	 FX10	 K
CPU Core	Instruction set architecture	ARMv8-A	SPARC V9		
	SIMD width	512bit	256bit	128bit	128bit
	Double precision (64bit)	✔	✔	✔	✔
	Single precision (32bit)	✔	✔	✔	✔
	Half precision (16bit)	✔	-	-	-
Interconnect	Tofu interconnect	Enhanced	Tofu2	Tofu	Tofu

Post-K Software Stack

FUJITSU

- Valuable feedbacks through “co-design” from application R&D teams

Post-K Applications

FUJITSU Technical Computing Suite / RIKEN Advanced System Software

Management Software	Hierarchical File I/O Software	Programming Environment
System management for highly available & power saving operation	Application-oriented file I/O middleware	XcalableMP
Job management for higher system utilization & power efficiency	Lustre-based distributed file system FEFS	MPI (Open MPI, MPICH)
		OpenMP, COARRAY, Math Libs.
		Compilers (C, C++, Fortran)
		Debugging and tuning tools

arm Linux OS / McKernel (Lightweight Kernel)

Post-K System Hardware

Post-K
Under Development
w/ RIKEN

FUJITSU

Compiler Development for ARMv8 with SVE

FUJITSU ARM Compiler for Post-K

FUJITSU

- Maximizes HPC application performance
  - Covers a wide range of applications where integer calculations are dominant
- Targets 512bit-wide vectorization as well as Vector-length-agnostic
  - Fixed-vector-length facilitates optimizations such as constant folding
- Inherits options/features of K computer, PRIMEHPC FX10 and FX100
- Language Standard Support
  - Fully supported : Fortran 2008, C11, C++14, OpenMP 4.5
  - Partially supported : Fortran 2015, C++1z, OpenMP 5.0
- Supports ARM C Language Extensions (ACLE) for SVE
  - ACLE allow programmers to use SVE instructions as C intrinsic functions

// C intrinsics in ACLE for SVE

```
svfloat64_t z0 = svld1_f64(p0, &x[i]);
svfloat64_t z1 = svld1_f64(p0, &y[i]);
svfloat64_t z2 = svadd_f64_x(p0, z0, z1);
svst1_f64(p0, &z[i], z2);
```

// SVE assembler

```
ld1d    z1.d, p0/z, [x19, x3, lsl #3]
ld1d    z0.d, p0/z, [x20, x3, lsl #3]
fadd    z1.d, p0/m, z1.d, z0.d
st1d    z1.d, p0, [x21, x3, lsl #3]
```

Smart Optimization Examples by SVE

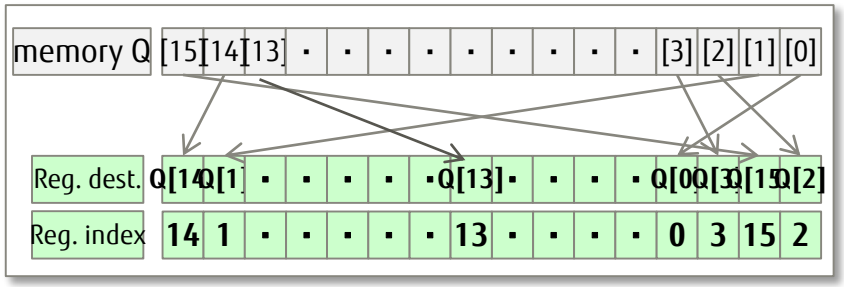
FUJITSU

4Byte x 16SIMD List Memory Access by utilizing 512bit Register

```
int index[n]
float P[n], Q[n];

for (i=0; i<n; ++i) {
  P[i] = Q[index[i]];
}
```

SVE



Various Types of SIMD Optimization by Utilizing Predicate Register

Loop including IF clause

```
for (int i=0; i<n; ++i) {
   if (mask[i] !=0) {  a[i] = b[i];   }
}
```

While Loop with Data Dependency

```
do {
   b[i] = a[i];
} while(a[i++] != 0);
```

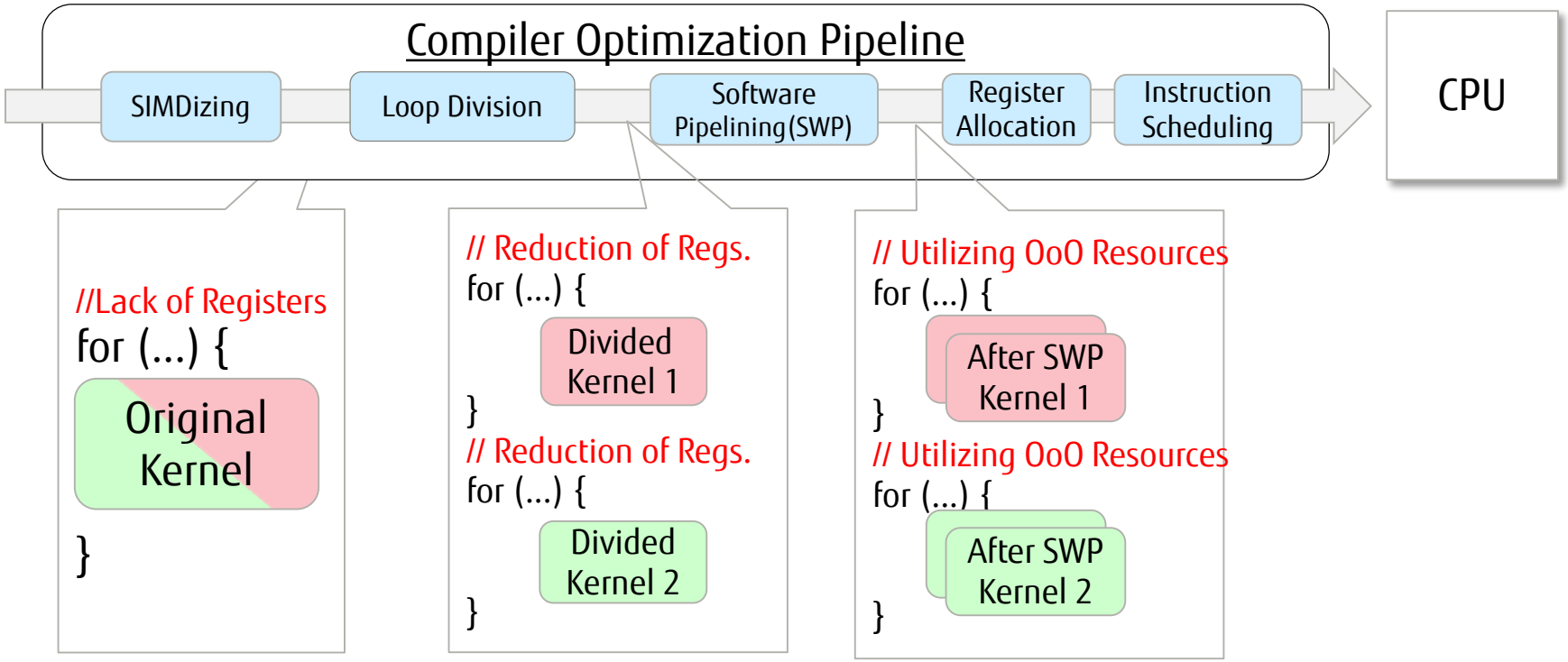
Small Loop less than SIMD Length

```
for (int i=0; i<VL/2; ++i) {
   a[i] = b[i] * c[i];
}
```

Post-K Compiler Improvement for ARM+SVE

FUJITSU

- Approach : Increasing Instruction Level Parallelism
  - Increasing Out-of-Order (OoO) Hardware Resources
  - Reducing Number of Required Registers by Loop Division
  - Increasing SIMD Execution Efficiency by Efficient Software Pipelining(SWP) and Register Allocation Techniques
  - Maximizing OoO resource Utilization by Efficient Instruction Scheduling



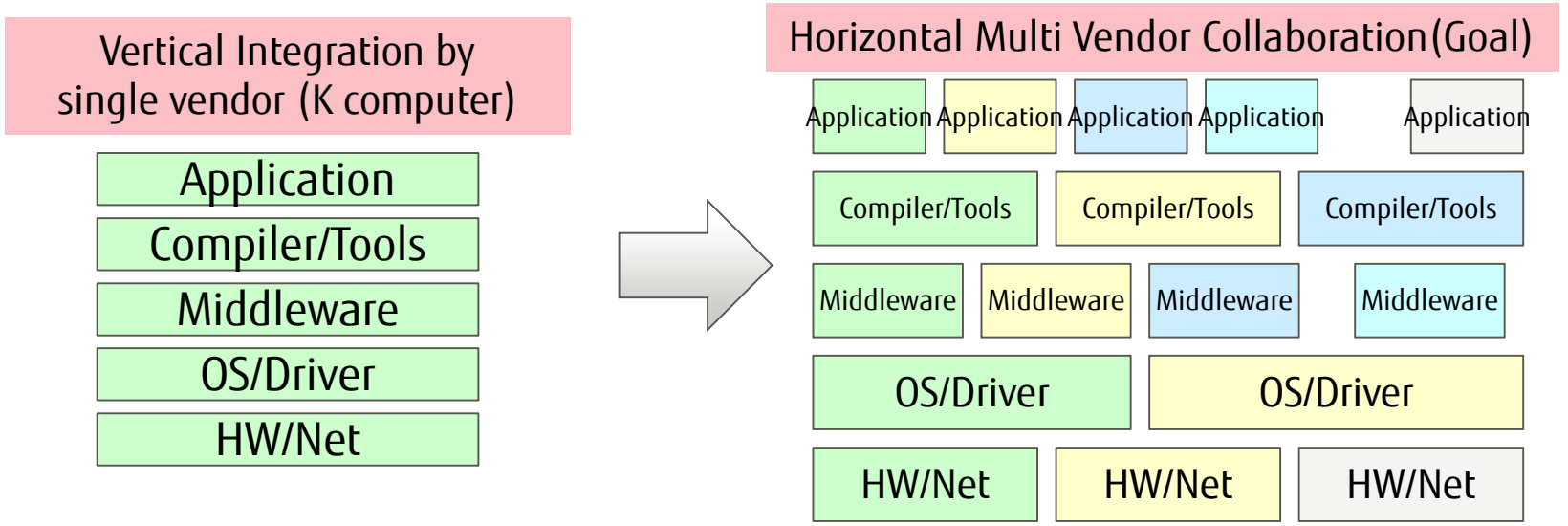
FUJITSU

Towards building up ARM HPC Ecosystem

Towards building up ARM HPC Ecosystem

FUJITSU

- Goal:
  - Horizontal multi vendor collaboration to build up commodity platform
- Ecosystem Building Steps:
  1. Preparing standard ARM HPC system software stack
  2. Building ARM HPC Market
  3. Expanding ARM HPC Market as a HPC Commodity Platform
- The First Step: Preparing ARM HPC system software stack
  - It is important to have a relationship with ARM HPC community

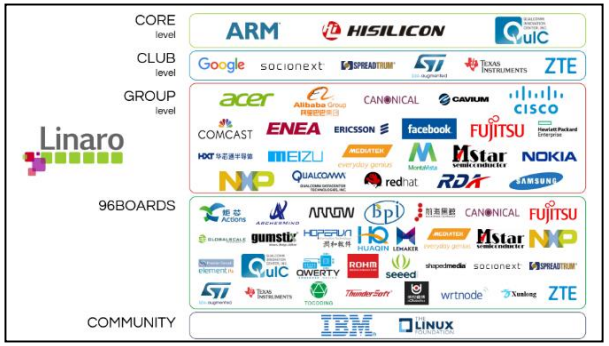


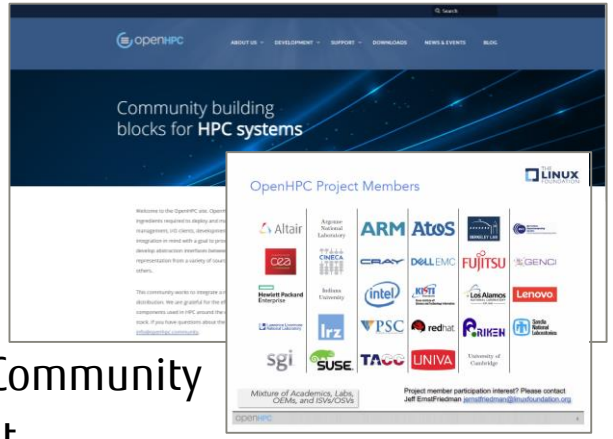
Relationship Strategy with ARM HPC community

FUJITSU

- ARM
  - Enablement of SVE to Linux, GCC SVE etc, and ARM OpenHPC Great Establishment and Contribution to ARM HPC base
    https://developer.arm.com/hpc
- Linaro
  - Standardization of ARM Basic System Software (Linux Kernel, glibc, GCC etc.) and Upstreaming to OSS community
  - Building execution binary portability on ARM HPC
  - Developing and upstreaming SVE software to OSS community
    https://www.linaro.org/sig/hpc/
- OpenHPC
  - Developing Standard IA and ARM HPC software portability
  - Distribution Schedule
    - 2016/11: v1.2　for ARM Tech. Preview Distributed
    - 2017/11: v1.3.3 for ARM Normal version will be distributed

    http://www.openhpc.community/
- Fujitsu
  - Contributing HPC experiences and technologies to ARM HPC Community
    →Supporting and developing ARM HPC software environment
  - Developing and providing SVE optimized Compiler and the other software environment to early development environment

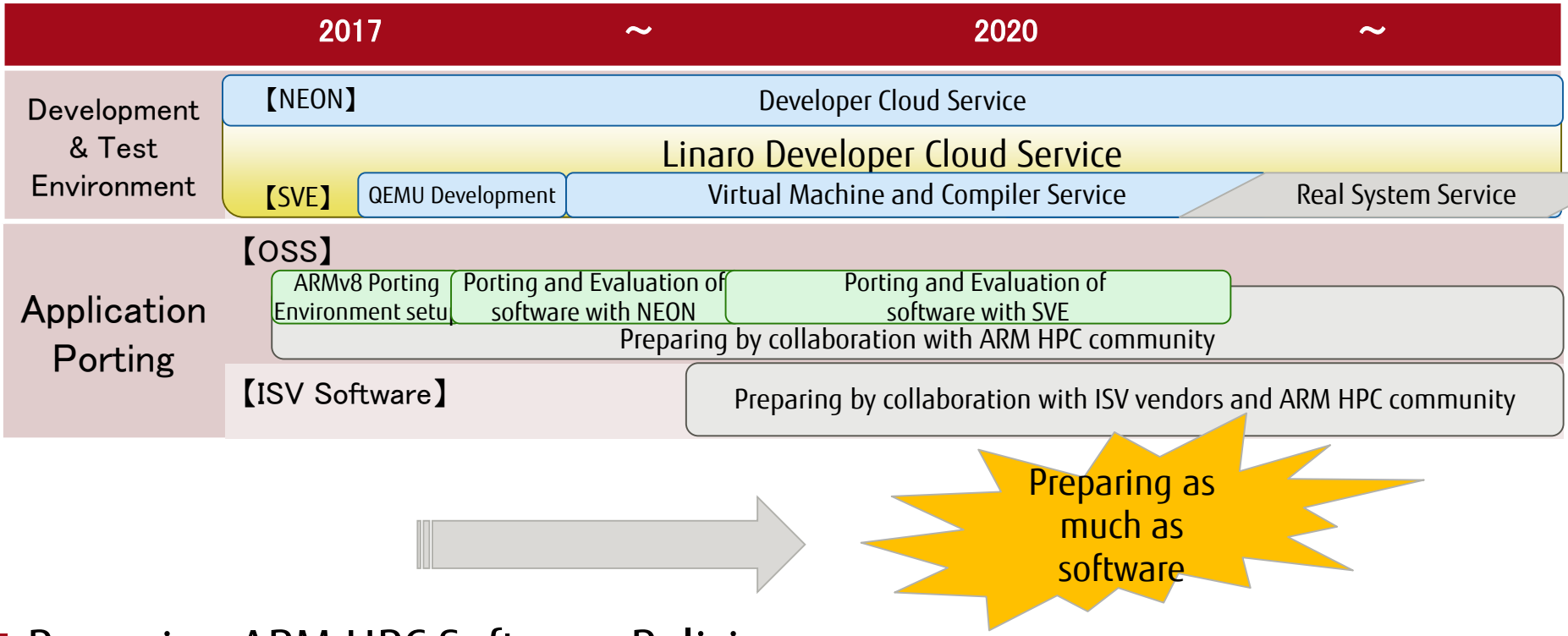




Fujitsu's ARM HPC Development Plan

FUJITSU

- ARM HPC software development plan with ARM HPC community



- Preparing ARM HPC Software Policies:
  - PC Cluster software can be executed only re-compilation of the software
    - Re-building of software package such as SRPM is preferable.
  - Expanding ARM based HPC software use and Keeping binary level portability
    - Binary Level Portability including Operating System Distribution and SVE

Fujitsu's Activities with ARM HPC Community

FUJITSU

- Linaro
  - QEMU with SVE Support
  - Providing Compiler Technology to LLVM
- OpenHPC
  - Porting Some OSS Package to ARM Environment
    - PLASMA /SCOTCH (PT-SCOTCH)/SLEPc
  - Providing RAS Technologies for Large System Orchestration
    - Orchestration : System Installation, Setting, Managing
  - Planning to nominate TSC(Technical Steering Committee)
- Open MPI
  - ARM Support Maintainer in the Community, Bug Fixes,
    - MPI 3.x, 4.x... function development
- Linux Distributers(RedHat/SUSE/Canonical etc.)
  - Requesting to provide SVE enabling distribution (Kernels, Compilers, Libraries, etc)

Linaro

https://www.linaro.org/

openHPC

http://openhpc.community

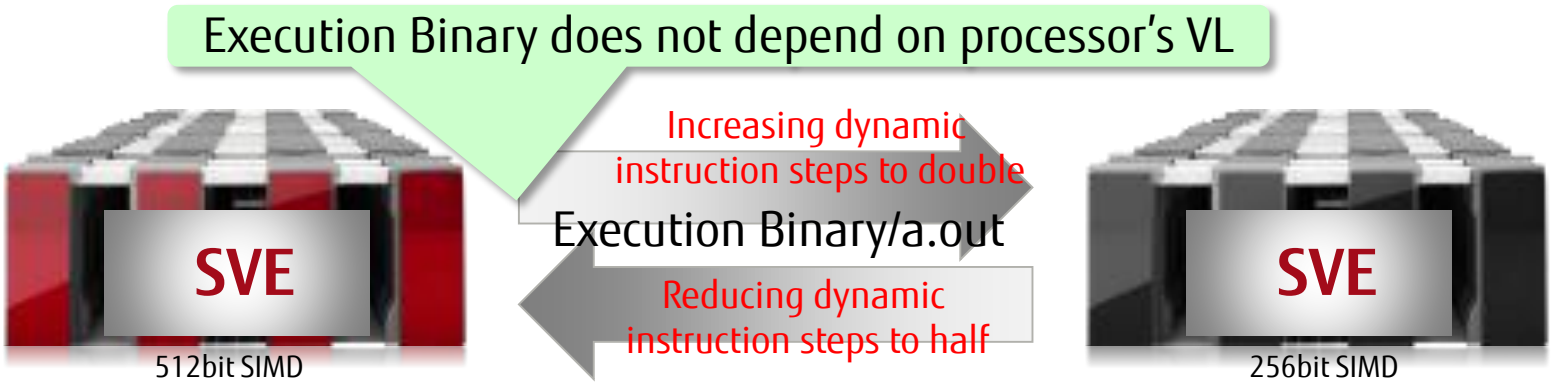
https://www.open-mpi.org/

Providing Execution Binary Portability(ARM, Linaro)

FUJITSU

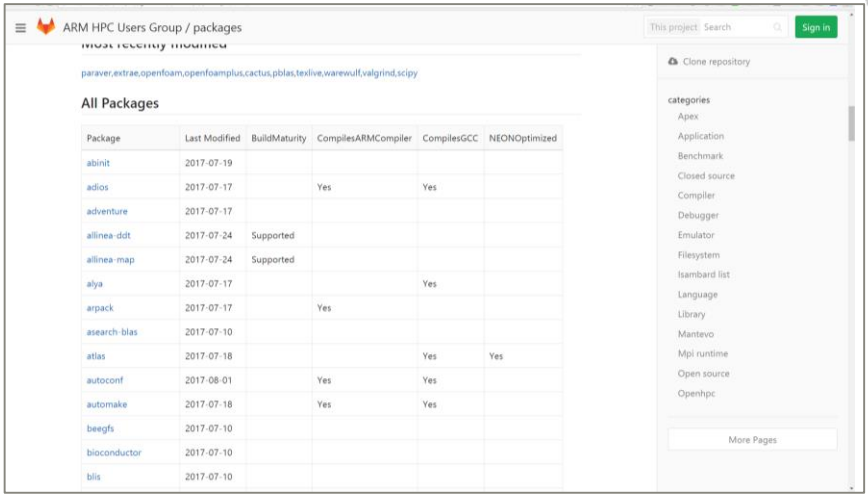
- Operating System Level Binary Portability
  - Two Specification defined by ARM, Linaro etc.
    - SBSA(Server Base System Architecture)
    - SBBR(Server Base Boot Requirements)
  - ARMv8 distribution, such as RedHat, SUSE, can be used without modification
- System Software Level Binary Portability
  - Linaro is building system software stack for ARM HPC.
- Application binary portability with different SIMD width
  - Provided by Scalable Vector Extension(SVE) Specification

Execution Binary Portability

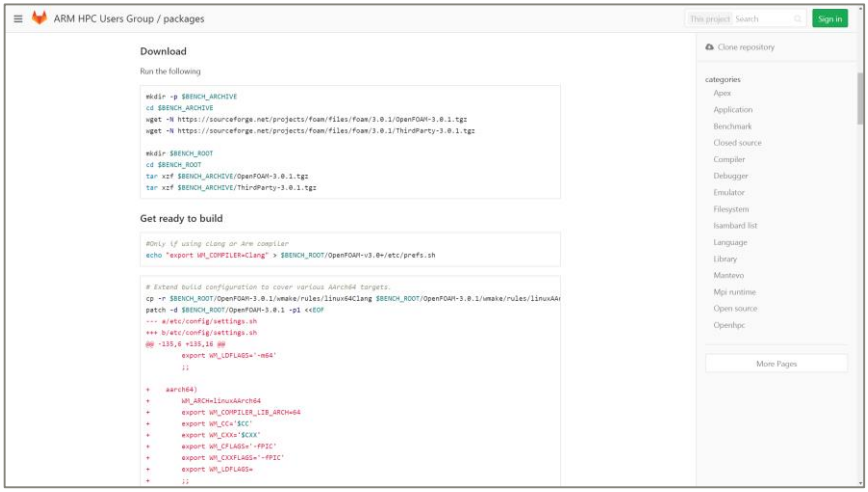


Activities of ARM HPC User Group
https://gitlab.com/arm-hpc

- Fujitsu plans to contribute our experiences to ARM HPC User Group
  - Now comparing our results to the ARM HPC User Group site results.
- How should we go forward?
  1. Porting packages
  2. Analyzing common/individual issues
  3. Fixing packages related common issues
  4. Fixing individual package issues
  5. Upstreaming OSS community
- Example of Common issue : OpenFOAM
  - Now: Requiring patches for each package because gcc/gfortran do not support –m64 option.
  - Future: Gcc/Gfortran should support –m64 option for IA/ARM portability.



https://gitlab.com/arm-hpc/packages/wikis/home



https://gitlab.com/arm-hpc/packages/wikis/packages/openfoam

gcc/gfortran vs LLVM Comparison(m64 option)

FUJITSU

- gcc/gfortran in ARM environment do not support m64 option.

Compiler	IA environment	ARM environment
gcc	○	×
gfortran	○	×
clang	○	○
flang	○	○

○: Support m64 option
×: Not support m64 option

```
[eco@ARM1-QEMU work]$ gfortran -m64 sample.f
gfortran: error: unrecognized command line option '-m64'
```

C Language Definition difference between IA and ARM

FUJITSU

- Specification of char is different between IA and ARM.
  - IA environment => signed char, ARM environment => unsigned char
    => If you use as signed char, you must specify fsigned-char option on gcc.

Sample program

```
#include<stdio.h>
int
main()
{
   char a=127, b=128;
   printf("%d,%d¥n",a,b);
}
```

IA environment

```
[eco@cn-r05-01 work]$ gcc sample.c
[eco@cn-r05-01 work]$ ./a.out
127,-128
[eco@cn-r05-01 work]$ clang sample.c
sample.c:5:19: warning: implicit conversion from 'int'
to 'char' changes value from 128 to -128 [-
Wconstant-conversion]
   char a=127, b=128;
1 warning generated.
[eco@cn-r05-01 work]$ ./a.out
127,-128
```

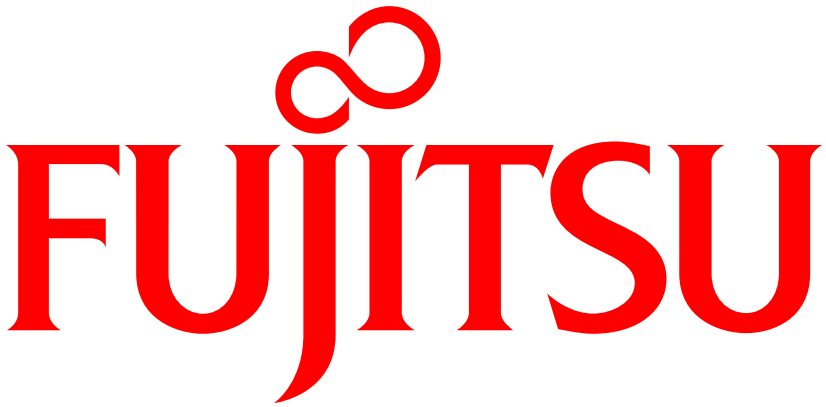
ARM environment

```
[eco@ARM1-QEMU work]$ gcc sample.c
[eco@ARM1-QEMU work]$ ./a.out
127,128
[eco@ARM1-QEMU work]$ clang sample.c
[eco@ARM1-QEMU work]$ ./a.out
127,128
[eco@ARM1-QEMU work]$
```

Summary

FUJITSU

- Fujitsu's Super computer development history and Post-K Processor and System software stack overview
- Fujitsu's Compiler Development for ARMv8 with SVE
  - Approach : Increasing Instruction Level Parallelism
    - Increasing Out-of-Order (OoO) Hardware Resources
    - Reducing Number of Required Registers by Loop Division
- Towards building up ARM HPC Ecosystem
  - Goal: Horizontal Multi vendor collaboration to build up commodity platform
  - The first step: Preparing ARM HPC system software stack
    - It is important to have a relationship with ARM HPC community
  - Current Status: We are now working with ARM HPC Community
    - ARM HPC Portability: Linaro, IA-ARM Portability: OpenHPC



shaping tomorrow with you