Preliminary Performance Evaluation of the Fujitsu A64FX Using HPC Applications

RIKEN Center for Computational Science

Tetsuya Odajima, Yuetsu Kodama, Miwako Tsuji,
Motohiko Matsuda, Yutaka Maruyama, and Mitsuhisa Sato

Background and Purpose

- **Supercomputer “Fugaku”**
  - The system has been installing at RIKEN R-CCS as the successor system of the supercomputer K
  - The A64FX processor developed through a co-design between Fujitsu and RIKEN
  - HPL; 415.53 PFLOPS with 396 racks
  - HPCG; 13366.40 TFLOPS with 360 racks

- **Preliminary performance evaluation with various HPC applications and benchmarks on Fugaku**
  - Just using compiler optimizations for this evaluation
  - Sharing the performance characteristics and usability

A64FX Processor

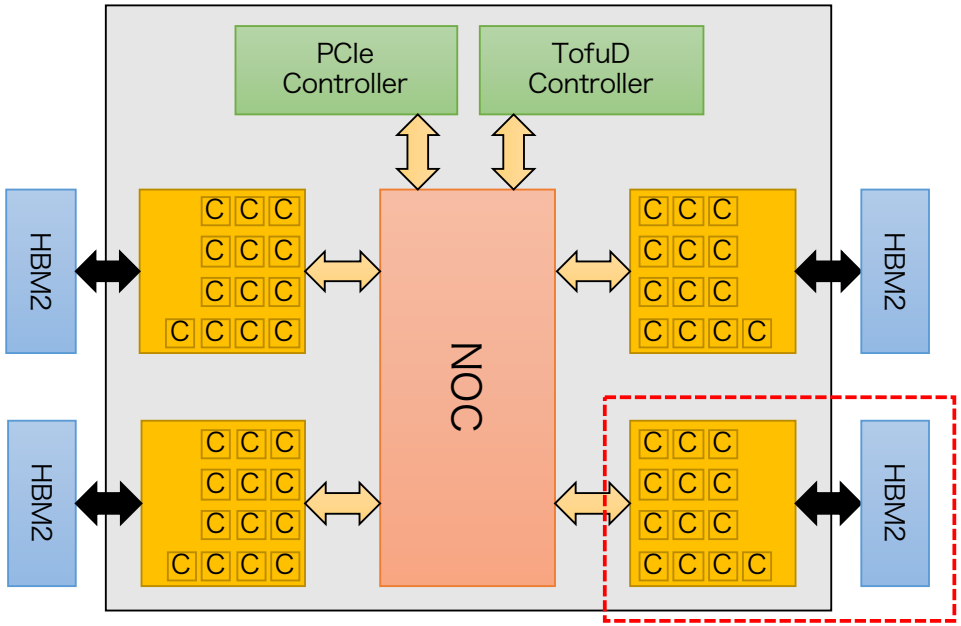
- **Specification**

ISA	Armv8.2-A + SVE (Scalable Vector Extension)
SIMD width	512-bit
# cores	48 (+ 2/4 for OS)
Memory	HBM2 32GB, 1024GB/s (8GB x 4, 256GB/s x 4)
Clock	2.0GHz (Normal), 2.2GHz (Boost)
Interconnect	Tofu interconnect D

- **Stream Triad : 830GB/s (> 80%)**
- **DGEMM: 2.7TFLOPS (> 90%)**

A64FX Document
https://github.com/fujitsu/A64FX

Image：https://pr.fujitsu.com/jp/news/2018/08/22-1.html



CMG (Core Memory Group)

Evaluation Environment

- **Comparing with Arm ThunderX2 and Intel Xeon Skylake**

	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 4.2.0 -Kfast,openmp	Arm HPC compiler 20.1 -Ofast -fopenmp -march=armv8.1-a	Intel compiler 19.0.5 -O3 -qopenmp -march=native

Applications and Benchmarks

Name of benchmarks	Developer	Language	Parallelization
RAJA Performance Suite	BSC	C++	OpenMP
BUDE	Univ. of Bristol	C	OpenMP
MiniFMM	Univ. of Bristol	C	OpenMP
LULESH	LLNL	C++	MPI+OpenMP
mega-sweep	UK Mini-app	Fortran	MPI+OpenMP
CloverLeaf	UK Mini-app	Fortran	MPI+OpenMP
TeaLeaf	UK Mini-app	Fortran	MPI+OpenMP

Performance Evaluation

- **Comparing the performance of A64FX, TX2, and SKL**
  - A64FX: 1 node (1 CPU)
  - TX2, SKL: 1 or 2 nodes (2 or 4 CPUs)
- **Strong scaling**

- **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.

TeaLeaf

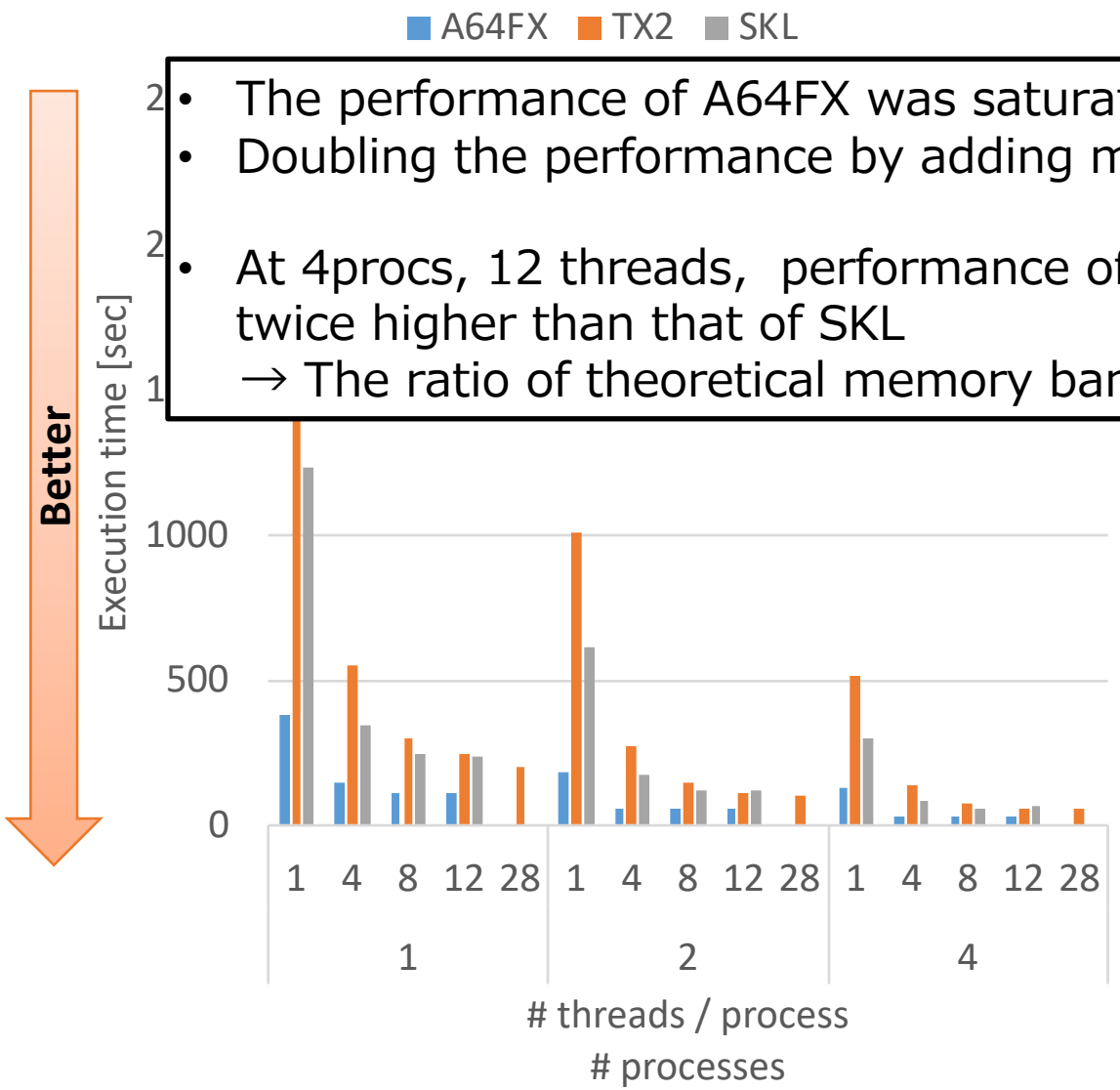
- **Mini-application for solving the linear heat conduction equation on a spatially decomposed grid using a 5-point stencil with implicit solvers**
- **Implicit solvers require a lot of memory accesses and have high pressure on memory**
- **Memory bandwidth-intensive application**

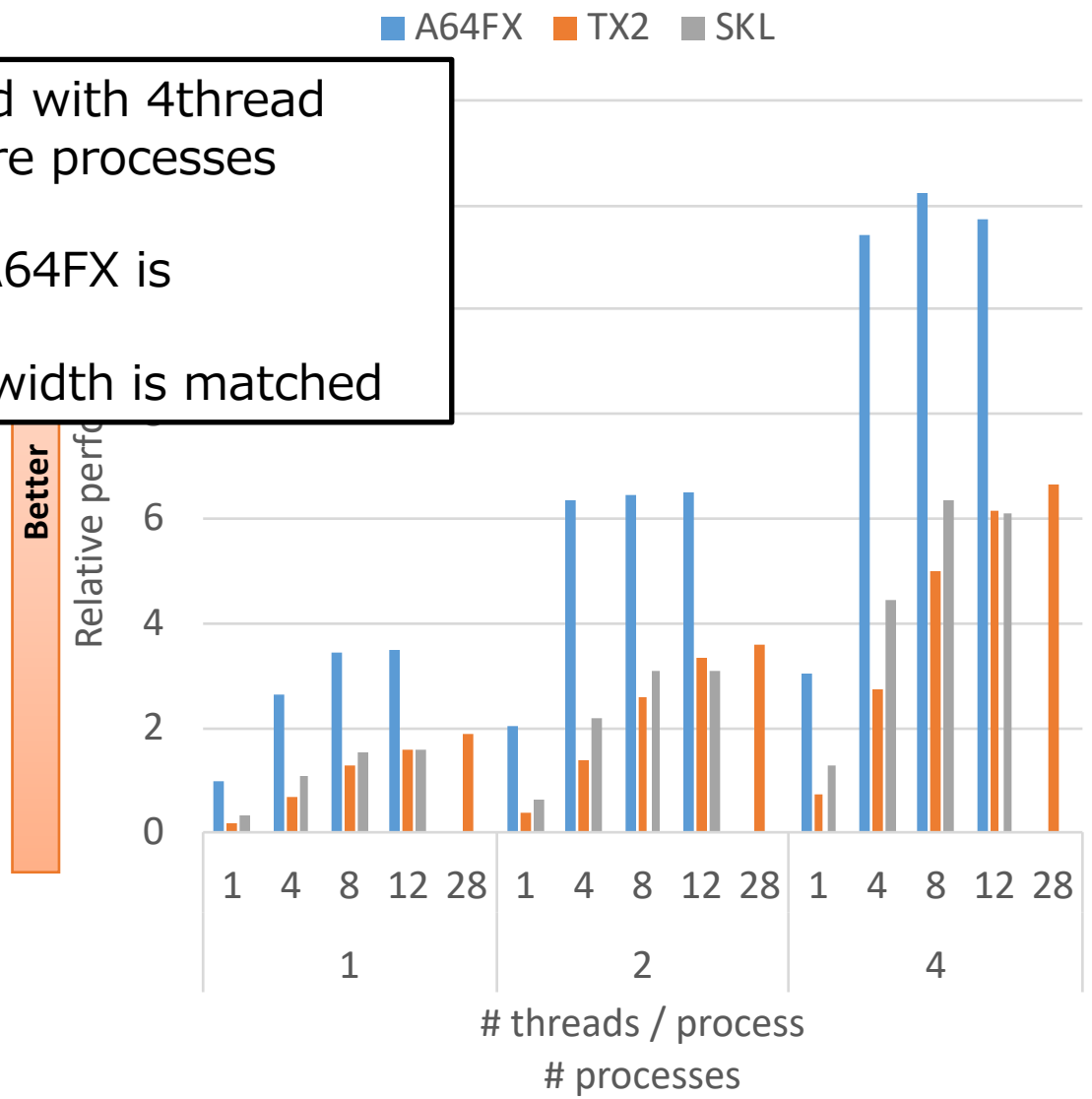
- **Problem size**
  - Parameter set; Benchmarks/tea_bm_5.in
  - Iterations; 3

TeaLeaf

- The performance of A64FX was saturated with 4thread
- Doubling the performance by adding more processes

- At 4procs, 12 threads, performance of A64FX is twice higher than that of SKL
  → The ratio of theoretical memory bandwidth is matched





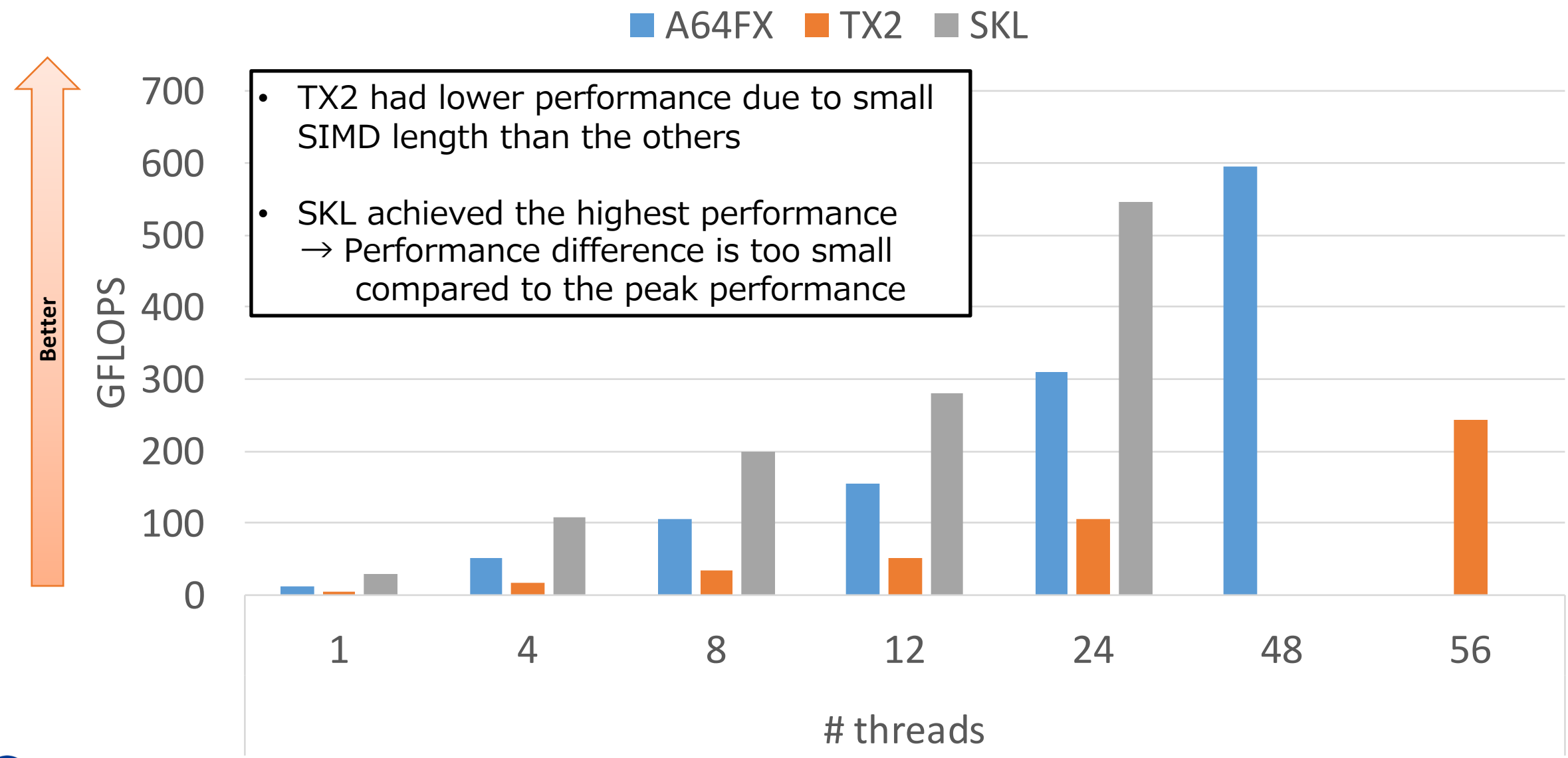
Normalized by A64FX 1-thread performance

BUDE

- **Bristol University Docking Engine**
- **Mini-application for general purpose molecular docking**
- **Calculation of potential energy using matrix information**
- **Compute-intensive application**

- **Problem size**
  - # molecules; 32,768
  - Iteration; 8 (default)

BUDE



- TX2 had lower performance due to small SIMD length than the others
- SKL achieved the highest performance
  → Performance difference is too small compared to the peak performance

Performance Features of the A64FX

- **For memory bandwidth-intensive applications**
  - A64FX outperform other processor
  - RAJA Performance Suite, CloverLeaf, TeaLeaf
- **For computing-intensive applications**
  - Actual performance of A64FX was not high compared to its peak performance
  - BUDE, MiniFMM

- **Insufficient out-of-order resources causes pipeline stall**
- **Application performance is degraded**

Amount of Hardware Resources

- Generally, the chip area per core is smaller than in Intel Xeon processors in manycore processors
- The number of buffers and registers required for out-of-order execution is small
- When instructions are chained in a loop, the frequency of arithmetic unit stalls increases due to insufficient resources

  → Overall computing performance will be degraded

	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)

Performance Improvement by Loop Split

- **Performance degradation of arithmetic unit due to due to lack of hardware resources**
  - When there are many variables in a loop, hardware resources be insufficient

    → Eliminating resource shortage by reducing variables in a loop
- **Efficient use of Out-of-Order resources**
  - Reducing variable usage per loop by dividing large loop
  - Pipelines operate efficiently and improve the performance of arithmetic units
  - “Loop fission” function of Fujitsu compiler makes it easy to implement loop split
    - Inserting pragma statements
    - Compiler splits a loop into multiple loops automatically

```
#pragma loop loop_fission_target
for (i = 0; i < N; i++) {
  …
}
```

Performance Improvement of Loop Fission

- **Apply loop fission function of Fujitsu compiler to BUDE**
- **The three loops in BUDE's kernel were split into few loops**

- **Performance [GFLOPS]**

	# threads					
	1	4	8	12	24	48
Original	13.06	52.25	104.49	154.78	309.28	594.26
Loop fission	18.00	71.99	143.97	213.23	425.88	817.26
Speedup	1.38	1.38	1.38	1.38	1.38	1.38

  - Increased operating rate of arithmetic units due to reduced pipeline stalls
  - Achieved 38% performance improvement by loop splitting

  - Loop fission does not always improve performance

Conclusion

- **We evaluated some HPC applications and benchmarks and compared the performance of A64FX against ThuderX2 and Skylake**

- **In memory bandwidth-intensive applications**

  → achieved high performance due to its high peak memory bandwidth

- **In compute-intensive applications**

  → the performance were very high due to the long instruction latency and the relatively small amount of hardware resources

- **Loop fission by Fujitsu compiler improved the performance**
  - 38% speedup in BUDE
  - However, loop fission does not always improve performance
  - Important to combine with other optimizations