Arm Research Summit 2020

FUJITSU

shaping tomorrow with you

Performance Optimization of SVE Enabled Arm Processor A64FX using Fujitsu Compiler

Shinji Sumimoto

Fujitsu Limited

Outline of This Talk

FUJITSU

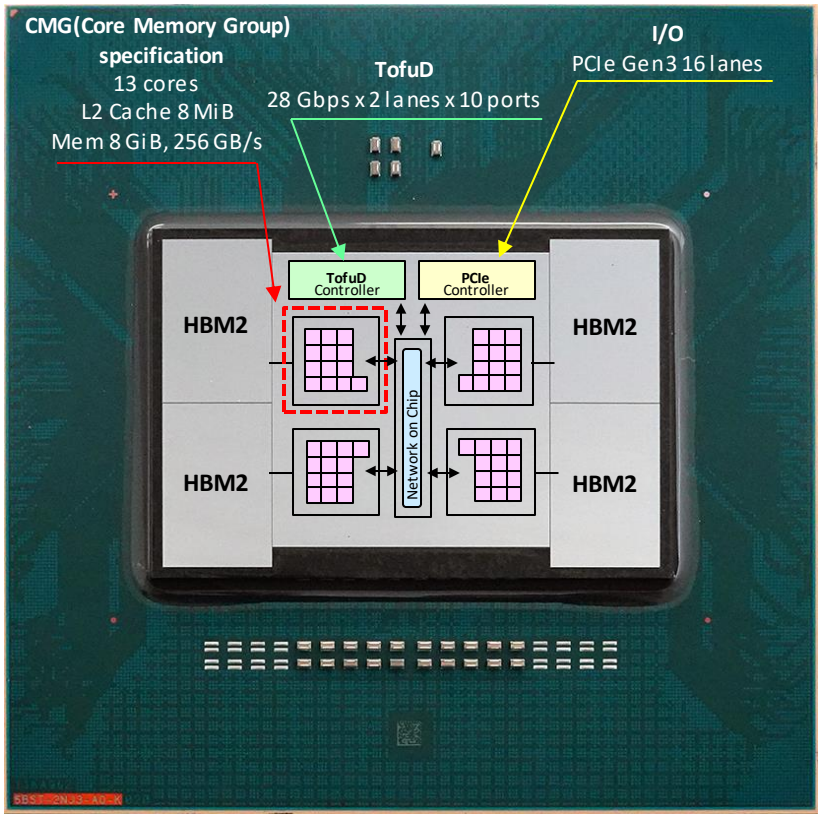
- Supercomputer Fugaku
  - Fugaku: Rank #1 Top500, HPCG, HPL-AI, Graph 500 @ISC 20, Jun. 2020

Benchmarks	Rank #1	Records	#2	Records	#1/#2 Ratio
TOP500	Fugaku	416 PFLOPS	Summit	148.6	2.80
HPCG	Fugaku	13.4 PFLOPS	Summit	2.93	4.56
HPL-AI	Fugaku	1.42 EFLOPS	Summit	0.55	3.19
Graph500	Fugaku	70,980 GTEPS	Sunway	23,756	2.98

- Overview of A64FX, PRIMEHPC FX1000 and FX700
- Brief Overview of Fujitsu Compiler
- Performance Optimization using Fujitsu Compiler: STREAM

High Performance Arm CPU “A64FX”

FUJITSU



Architecture features

ISA	Armv8.2-A (AArch64 only) SVE (Scalable Vector Extension)
SIMD width	512-bit
Precision	FP64/32/16, INT64/32/16/8
# of cores	48 computing cores + 4 assistant cores (4 CMGs)
Memory	HBM2: Peak B/W 1024 GB/s
Interconnect	TofuD: 28 Gbps x 2 lanes x 10 ports

New PRIMEHPC Lineup

FUJITSU

PRIMEHPC FX1000

Supercomputer optimized for large scale computing

High Performance | High Scalability | High Density

A64FX processor
384 nodes/Rack
Tofu Interconnect D
Water Cooling
Fujitsu Software Stack
for Supercomputing

PRIMEHPC FX700

Supercomputer based on standard technologies

Fugaku Technology

Ease to use | Installation

A64FX Processor
8 nodes/2U Rackmount
InfiniBand
Air Cooling
Utilize ISV and Open Source Software Stack

Comparison between the FX1000 and FX700

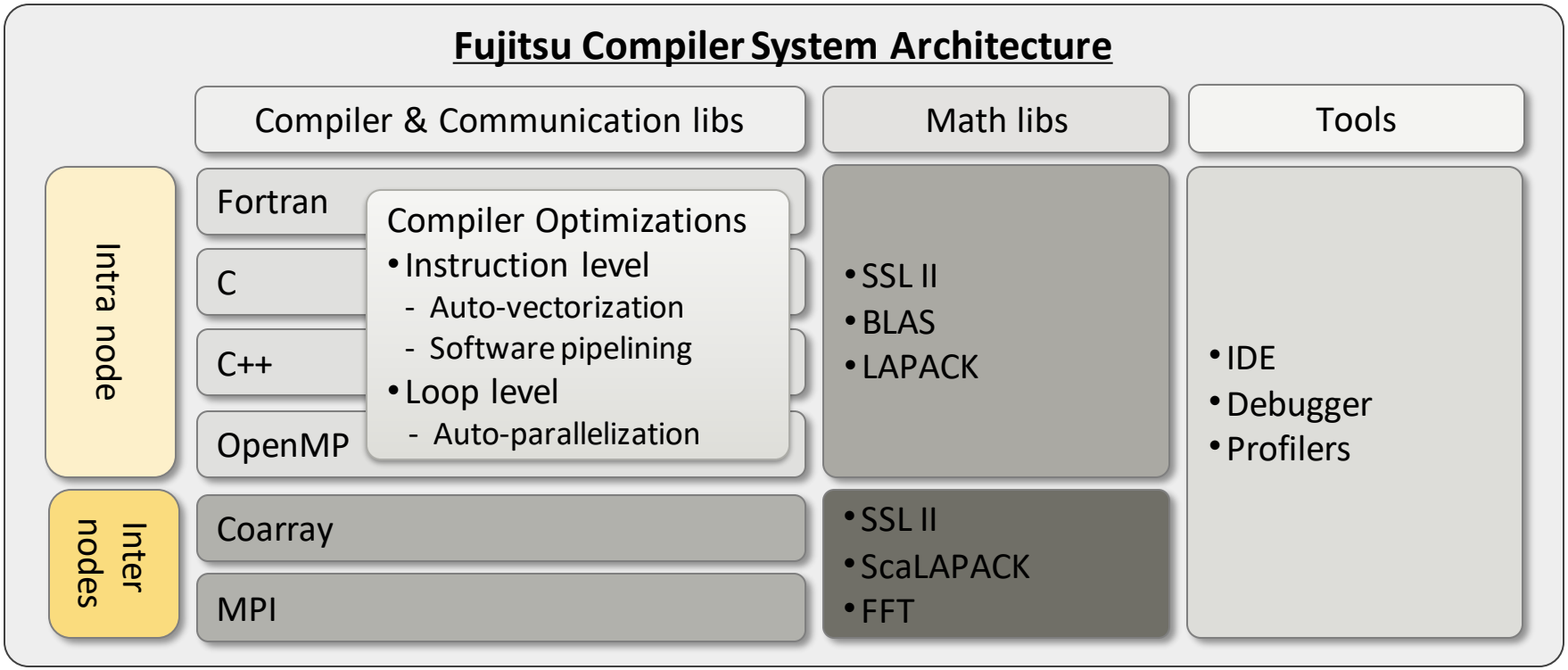
FUJITSU

		FX1000	FX700
CPU	Name	A64FX	A64FX
	ISA	Armv8.2-A SVE	Armv8.2-A SVE
	Cores	48 computation cores + 2-4 assistant cores	48 computation cores
	Frequency	2.2GHz	2.0GHz / 1.8GHz
Node	CPUs	1 CPU	1 CPU
	Memory capacity	32GiB (HBM2)	32GiB (HBM2)
	Memory bandwidth	1,024GB/s	1,024GB/s
	Interconnect	Tofu Interconnect D	InfiniBand
Enclosure	Form factor	Dedicated rack	2U rack-mountable chassis
	Node per rack or chassis	384 node/rack	8 node/chassis
	Heat-removal method	Water cooling + Air cooling	Air cooling

FUJITSU

Brief Overview of Fujitsu Compiler

Fujitsu Compiler System Architecture



- Develops a variety of programming tools for various programming models
- Designs and develops Software exploiting Hardware performance

A64FX Features and Compiler Approaches

FUJITSU

- A64FX CPU Inherits features of K computer and PRIMEHPC FX100
  - Usability including options and programming models are inherited
- Compiler targeting 512-bit wide vectorization to promotes optimization, such as constant folding, by fixing vector length
  - Vectorization as VLA(vector-length-agnostic) and NEON (Advanced SIMD)
  - Inter-core hardware barrier and Sector Cache, hardware and software prefetch

	Functions & Architecture	Post-K	FX100	K computer
Processor	Base ISA + SIMD Extensions	ARMv8-A+SVE	SPARC V9 +HPC-ACE2	SPARC V9 +HPC-ACE
	SIMD width [bits]	512	256	128
	Float Packed SIMD	✓ Enhanced	✓	-
	FMA	✓	✓	✓
	Reciprocal approx. inst. Math. acceleration inst.	✓	✓	✓
	Inter-core hardware barrier	✓	✓	✓
	Sector cache	✓ Enhanced	✓	✓
	Hardware “prefetch” assist	✓ Enhanced	✓	✓

Fujitsu Compiler: Language Standard Support

FUJITSU

Languages	Specification	Support Level
C	C11 (ISO/IEC 9899:2011)	fully supported
C++	C++14 (ISO/IEC 14882:2014) C++17 (ISO/IEC 14882:2017)	fully supported partially supported
Fortran	Fortran 2008 (ISO/IEC 1539-1:2010) Fortran 2018 (ISO/IEC 1539-1:2018)	fully supported partially supported
OpenMP	OpenMP 4.0 (released in July 2013) OpenMP 4.5 (released in Nov. 2015) OpenMP 5.0 (released in Nov. 2018)	fully supported partially supported partially supported

Promotes object-oriented programming and accelerates high performance by supporting latest language standards

FUJITSU

Optimization of A64FX by Fujitsu Compiler: STREAM

- Evaluation of STREAM Triad on PRIMEHPC FX1000

STREAM Benchmark Code triad in C Language

FUJITSU

- Basic Combination of Computation and Memory Access Benchmark
  - Four Benchmarks: Copy, Scale, Add and Triad
- STREAM Triad is the most memory intensive one

- Basic C source code of STREAM Triad Benchmark in OpenMP
  - Simple OpenMP Loop

https://github.com/jeffhammond/STREAM

s-sumi@sumi-server:~

File Edit Options Buffers Tools C Help

```
        times[3][k] = mysecond();
#ifdef TUNED
        tuned_STREAM_Triad(scalar);
#else
#pragma omp parallel for
        for (j=0; j<STREAM_ARRAY_SIZE; j++)
            a[j] = b[j]+scalar*c[j];
#endif
        times[3][k] = mysecond() - times[3][k];
        }
```

-UU-:----F1 stream.c 70% L339 (C/*l Abbrev)

s-sumi@sumi-server:~

File Edit Options Buffers Tools C Help

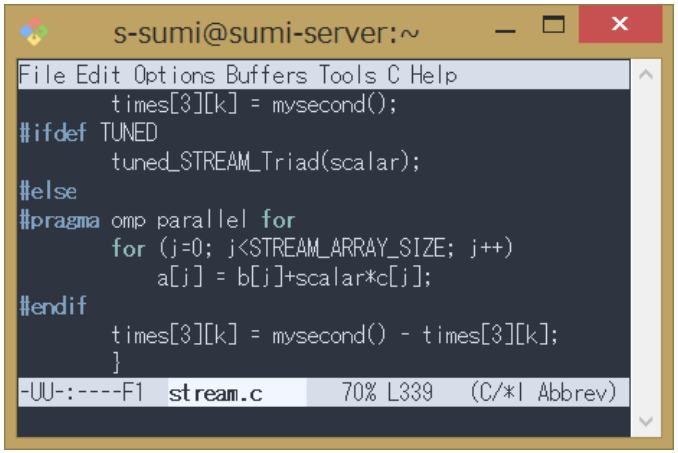
```
#ifndef STREAM_TYPE
#define STREAM_TYPE double
#endif

static STREAM_TYPE      a[STREAM_ARRAY_SIZE+OFFSET],
                        b[STREAM_ARRAY_SIZE+OFFSET],
                        c[STREAM_ARRAY_SIZE+OFFSET];
```

-UU-:----F1 stream.c 46% L175 (C/*l Abbrev)

STREAM Triad Performance Test Options

FUJITSU

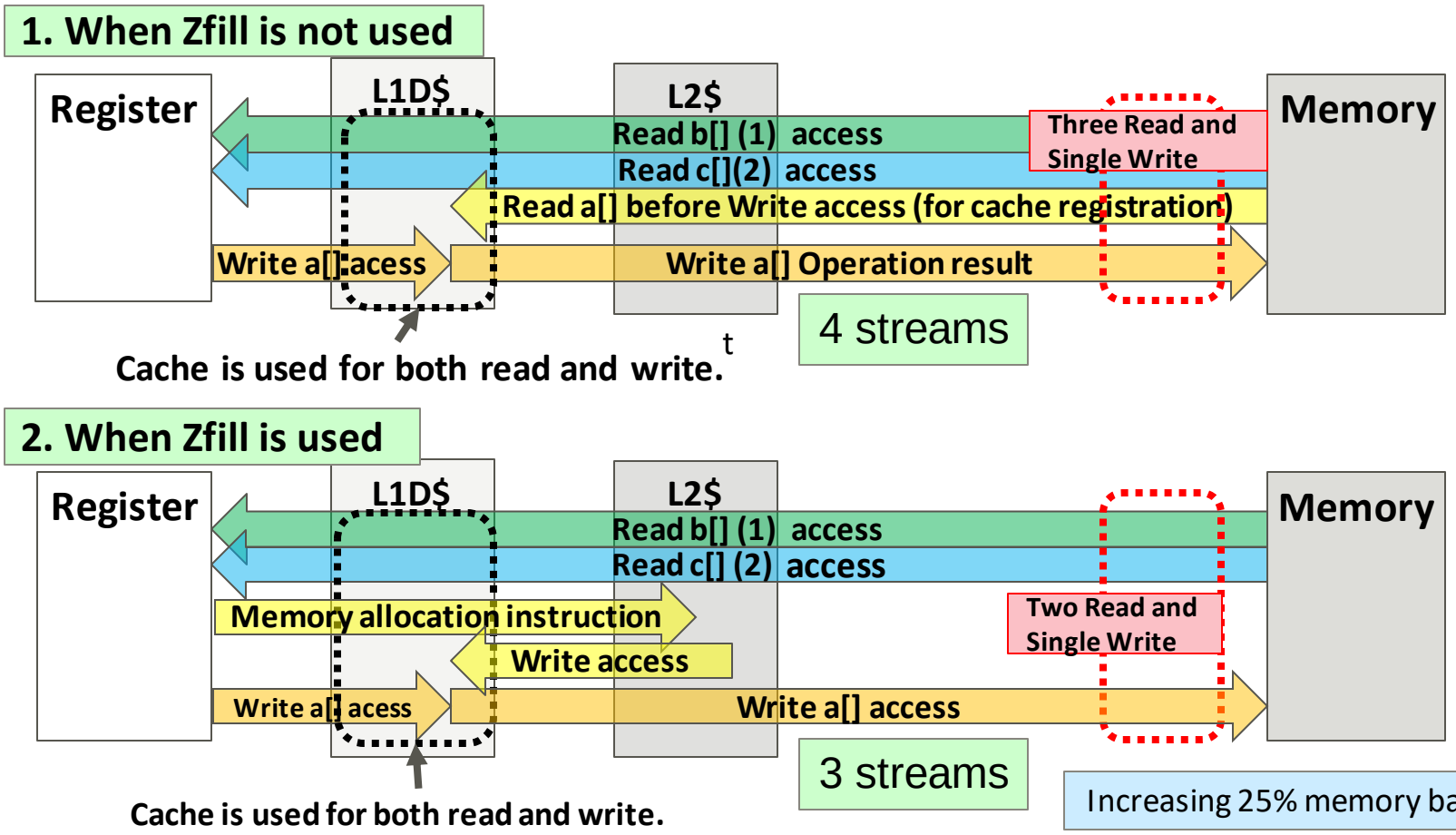
- To utilize maximum memory bandwidth and computation
  - Boosting memory prefetch access
  - Boosting SIMD operation
  - Removing extra memory access:
    - Extra memory prefetch operation
    - Needless memory read access



FJ Compiler Options	
(1) -O3 -KA64FX -KSVE -KARMV8_3_A –Kopenmp	Basic Optimization(Opt.)
(2) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kassume=memory_bandwidth	(1) + Memory Bandwidth Opt.
(3) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kzfill=100 -Kprefetch_sequential=soft -Kprefetch_line=8 -Kprefetch_line_L2=16	(1) + Zfill and Software Prefetch Opt.
(4) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kzfill=100 -Kprefetch_sequential=soft -Kprefetch_line=8 -Kprefetch_line_L2=16 -Knounroll	(3) – Loop unrolling

Zfill: Eliminating Needless Memory Access

- Usually, a memory write access needs read the memory data before writing
- Zfill eliminates extra memory read which is completely replaced in cache line
- The A64FX Zfill uses the DC ZVA instruction to eliminate needless read access.



Original STREAM Triad Performance Comparison

FUJITSU

FJ Compiler Options/ Number of threads	1	4	12	48 threads
(1) -O3 -KA64FX -KSVE -KARMV8_3_A –Kopenmp	72.3	152.7	155.4	629.2 GB/s
(2) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kassume=memory_bandwidth	38.1	151.5	207.2	822.1 GB/s
(3) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kzfill=100 -Kprefetch_sequential=soft -Kprefetch_line=8 -Kprefetch_line_L2=16	39.5	142.4	213.1	848.4 GB/s
(4) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kzfill=100 -Kprefetch_sequential=soft -Kprefetch_line=8 -Kprefetch_line_L2=16 -Knounroll	39.6	142.7	213.2	849.6 GB/s

XOS_MMM_L_HPAGE_TYPE=demand, XOS_MMM_L_HPAGE_TYPE=hugetlbfs
Memory per array = 512.0 MiB (= 0.5 GiB).

- By using zfill option and prefetch parameter optimization, over 800GB/s stream triad performance will be able to achieve.

Optimization Results: (1) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp

FUJITSU

Analyzed using -Nlst=t -Koptmsg=2 options

```
339   i          times[3][k] = mysecond();
340              #ifdef TUNED
341                    tuned_STREAM_Triad(scalar);
342              #else
343              #pragma omp parallel for
                 <<< Loop-information Start >>>
                 <<<  [OPTIMIZATION]
                 <<<    SIMD(VL: 8)
                 <<<    SOFTWARE PIPELINING(IPC: 3.00, ITR: 144, MVE: 3)
                 <<<    PREFETCH(HARD) Expected by compiler :
                 <<<      c, b, a
                 <<< Loop-information  End >>>
344   p     2v    for (j=0; j<STREAM_ARRAY_SIZE; j++)
345   p     2v       a[j] = b[j]+scalar*c[j];
346              #endif
347   i          times[3][k] = mysecond() - times[3][k];
348              }
349
```

- SIMD, SWP, Prefetch Optimization applied

Optimization Results: (2) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kassume=memory_bandwidth

FUJITSU

Analyzed using -Nlst=t -Koptmsg=2 options

```
339   i          times[3][k] = mysecond();
340              #ifdef TUNED
341                    tuned_STREAM_Triad(scalar);
342              #else
343              #pragma omp parallel for
                 <<< Loop-information Start >>>
                 <<<  [OPTIMIZATION]
                 <<<    SIMD(VL: 8)
                 <<<    SOFTWARE PIPELINING(IPC: 2.70, ITR: 160, MVE: 2)
                 <<<    PREFETCH(HARD) Expected by compiler :
                 <<<      c, b
                 <<<    PREFETCH(SOFT) : 2
                 <<<     SEQUENTIAL : 2
                 <<<      a: 2
                 <<<    ZFILL          :
                 <<<      a
                 <<< Loop-information  End >>>
344   p      v     for (j=0; j<STREAM_ARRAY_SIZE; j++)
345   p      v         a[j] = b[j]+scalar*c[j];
346              #endif
347   i          times[3][k] = mysecond() - times[3][k];
348              }
349
```

- SIMID, SWP(2), Prefetch, Zfill Optimization applied

Optimization Results: (3) -O3 -KA64FX -KSVE -KARMV8_3_A -Kopenmp -Kzfill=100 -Kprefetch_sequential=soft -Kprefetch_line=8 -Kprefetch_line_L2=16

FUJITSU

Analyzed using -Nlst=t -Koptmsg=2 options

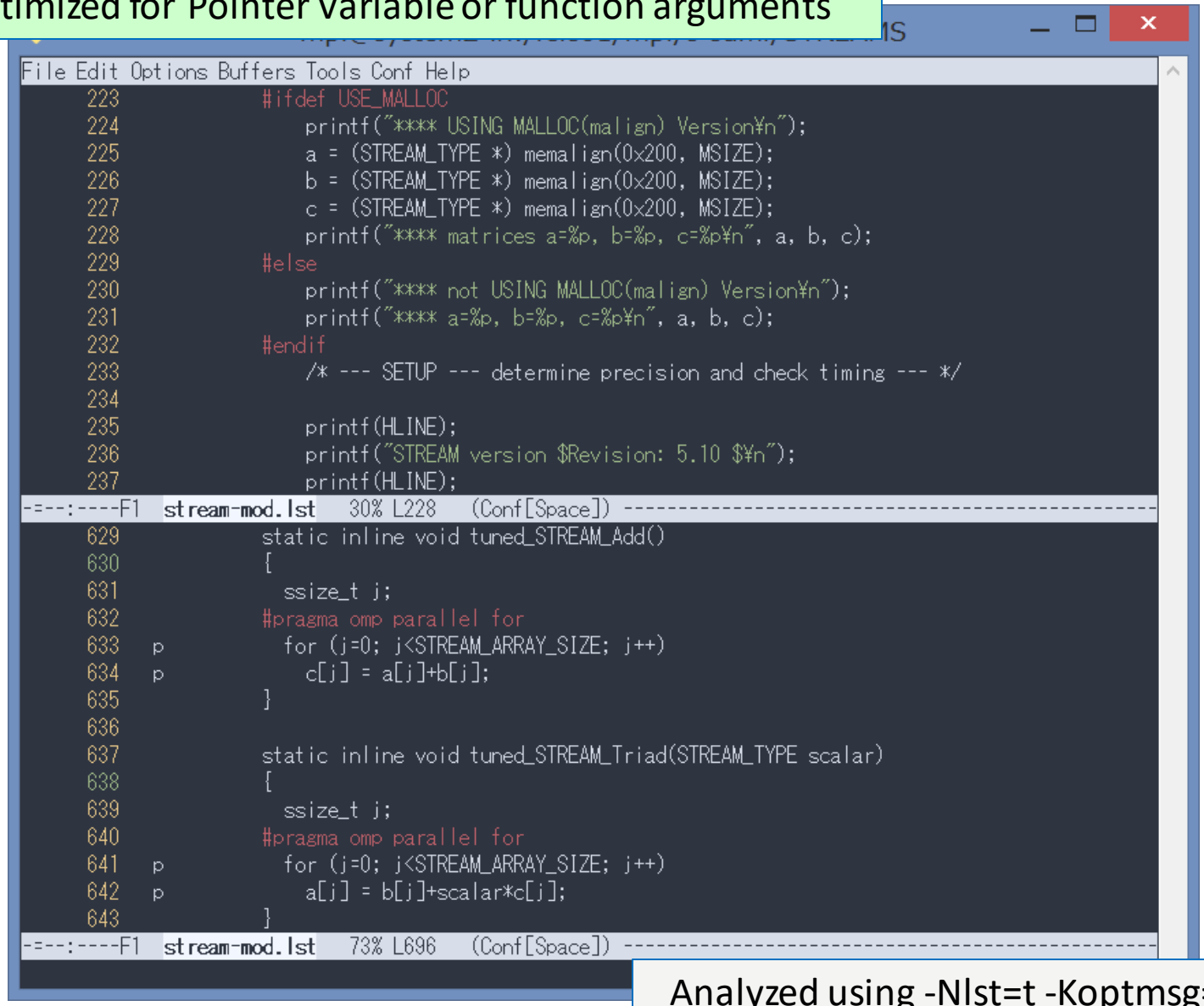
```
339   i          times[3][k] = mysecond();
340              #ifdef TUNED
341                    tuned_STREAM_Triad(scalar);
342              #else
343              #pragma omp parallel for
                 <<< Loop-information Start >>>
                 <<<  [OPTIMIZATION]
                 <<<    SIMD(VL: 8)
                 <<<    SOFTWARE PIPELINING(IPC: 2.81, ITR: 224, MVE: 3)
                 <<<    PREFETCH(SOFT) : 15
                 <<<     SEQUENTIAL : 15
                 <<<      c: 6, b: 6, a: 3
                 <<<    ZFILL          :
                 <<<      a
                 <<< Loop-information  End >>>
344   p      v    for (j=0;j<STREAM_ARRAY_SIZE;j++)
345   p      v        a[j] = b[j]+scalar*c[j];
346              #endif
347   i          times[3][k] = mysecond() - times[3][k];
348              }
349
```

- SIMID, SWP(3), Prefetch, Zfill Optimization applied

Some cases: Not optimized program sample

FUJITSU

- Not optimized for Pointer Variable or function arguments



```
File Edit Options Buffers Tools Conf Help
      223             #ifdef USE_MALLOC
      224                 printf("**** USING MALLOC(malign) Version¥n");
      225                 a = (STREAM_TYPE *) memalign(0x200, MSIZE);
      226                 b = (STREAM_TYPE *) memalign(0x200, MSIZE);
      227                 c = (STREAM_TYPE *) memalign(0x200, MSIZE);
      228                 printf("**** matrices a=%p, b=%p, c=%p¥n", a, b, c);
      229             #else
      230                 printf("**** not USING MALLOC(malign) Version¥n");
      231                 printf("**** a=%p, b=%p, c=%p¥n", a, b, c);
      232             #endif
      233                 /* --- SETUP --- determine precision and check timing --- */
      234
      235                 printf(HLINE);
      236                 printf("STREAM version $Revision: 5.10 $¥n");
      237                 printf(HLINE);
-=--:----F1  stream-mod.lst    30% L228    (Conf[Space]) ----------------------------------------------
      629             static inline void tuned_STREAM_Add()
      630             {
      631               ssize_t j;
      632             #pragma omp parallel for
      633   p           for (j=0; j<STREAM_ARRAY_SIZE; j++)
      634   p             c[j] = a[j]+b[j];
      635             }
      636
      637             static inline void tuned_STREAM_Triad(STREAM_TYPE scalar)
      638             {
      639               ssize_t j;
      640             #pragma omp parallel for
      641   p           for (j=0; j<STREAM_ARRAY_SIZE; j++)
      642   p             a[j] = b[j]+scalar*c[j];
      643             }
-=--:----F1  stream-mod.lst    73% L696    (Conf[Space]) ----------------------------------------------
```

Analyzed using -Nlst=t -Koptmsg=2 options

Improving by using const and __restrict

FUJITSU

mpi@system2-ln:/fefs01/mpi/s-sumi/STREAMS

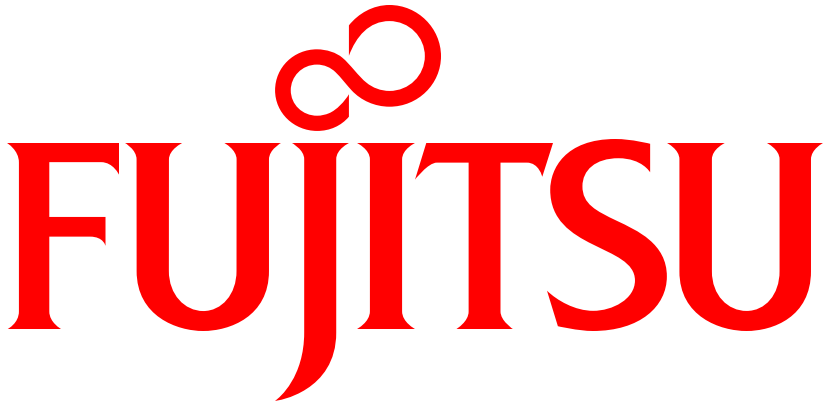
```
File Edit Options Buffers Tools Conf Help
      676            static inline void tuned_STREAM_Triad(STREAM_TYPE scalar)
      677            {
      678              const STREAM_TYPE * __restrict cc = c;
      679              const STREAM_TYPE * __restrict bb = b;
      680              STREAM_TYPE * restrict aa = a;
      681              ssize_t j;
      682            #pragma omp parallel for
                      <<< Loop-information Start >>>
                      <<<  [OPTIMIZATION]
                      <<<    SIMD(VL: 8)
                      <<<    SOFTWARE PIPELINING(IPC: 2.76, ITR: 192, MVE: 3)
                      <<<    PREFETCH(SOFT) : 15
                      <<<     SEQUENTIAL : 15
                      <<<       ZFILL    : 15
                      <<< Loop-information  End >>>
      683   p      v    for (j=0; j<STREAM_ARRAY_SIZE; j++)
      684   p      v      aa[j] = bb[j]+scalar*cc[j];
      685            }
      686            #endif
      687
      688            /* end of stubs for the "tuned" versions of the kernels */
      689            #endif
Total prefetch num: 174
Parallelization messages
  jwd6101s-i  "stream-mod.c", line 287: SIMD conversion is not applied because a statement that prevent¥
s SIMD conversion exists.
  jwd8663o-i  "stream-mod.c", line 287: This loop is not software pipelined because the software pipeli¥
ning does not improve the performance.
  jwd6101s-i  "stream-mod.c", line 306: SIMD conversion is not applied because a statement that prevent¥
s SIMD conversion exists.
  jwd8663o-i  "stream-mod.c", line 306: This loop is not software pipelined because the software pipeli¥
-:---:----F1  stream-mod.lst    77% L760    (Conf[Space]) ----------------------------------------------
```

Analyzed using -Nlst=t -Koptmsg=2 options

Summery

FUJITSU

- Supercomputer Fugaku, A64FX, PRIMEHPC FX1000 and FX700
- Brief Overview of Fujitsu Compiler
- Performance Optimization using Fujitsu Compiler: STREAM Triad
  - To utilize maximum memory bandwidth and computation
    - Boosting memory prefetch access
    - Boosting SIMD operation
    - Removing extra memory access: Extra memory prefetch operation and Needless read-modify-write
  - Achieved around 850GB/s STREAM Triad Performance



shaping tomorrow with you