Preliminary Performance Evaluation of Application Kernels using ARM SVE with Multiple Vector Lengths

Y. Kodama, T. Odajima, M. Matsuda,
M. Tsuji, J. Lee and M. Sato

RIKEN AICS
(Advanced Institute for Computational Science)

Outline

- Background and our research agenda
  - Overview of SVE
- Evaluation Environment
  - Gem5 simulator
  - Architectural parameters
  - Evaluated programs
- Evaluation Results
- Discussion
- Conclusion

Background (1)

Processor trends:

- Many core:
  - ✓Intel Knights Landing (KNL): 60~72core
- Wide SIMD
  - ✓AVX-2 (256-bit); Intel Xeon E5 v4 (Broadwell)
  - ✓AVX512 (512-bit); Intel KNL, Xeon E5 v5 (Skylake)

No program compatibility between different SIMD length

- Re-compile is required between AVX-2 and AVX512

Background (2)

ARM Scalable Vector Extension (SVE)

- **Vector Length Agnostic programming**
  independent of vector length
  - ✓Same binaries run on different vector length environment.
- Support 128bit～2048bit SIMD
  - ✓Each processor may support different vector length
  - ✓SVE instructions don’t have vector length information, but refer the value of LEN implicitly.
  - ✓LEN is in system register, that specifies current vector length

  LEN=1:128bit, 2:256bit, 4:512bit, 8:1024bit, 16:2048bit
  - ✓LEN can be changed by kernel call.

Vector length agnostic programming

ex) for (int i = 0; i < N; i++)

y[i] = 3.0 * x[i] + y[i];

This code runs with any vector length

Scalar

```
        fmov    d2, 3.0e+0
        mov     x0, 0 // int i
.L2:
        ldr     d0, [x2, x0]
        ldr     d1, [x1, x0]
        fmadd   d0, d0, d2, d1
        str     d0, [x1, x0]
        add     x0, x0, 8 // i++
        cmp     x0, 1024 // i < N?
        bne     .L2
```

SVE

```
        fmov    z0.d, #3.00000000
        whilelo p0.d, xzr, x9 // 0 < N?
.LBB0_1:
        ld1d    z1.d, p0/z, [x10, x8, lsl #3]
        ld1d    z2.d, p0/z, [x11, x8, lsl #3]
        fmad    z1.d, p0/m, z0.d, z2.d
        st1d    z1.d, p1, [x11, x8, lsl #3]
        incd    x8 // i+=(# of elements)
        whilelo p1.d, x8, x9 // i < N?
        b.first .LBB0_1 // p0[0] is true ?
```

This SVE code correctly runs for any N iterations, even if N is not the multiple of vector elements.

Our research agenda

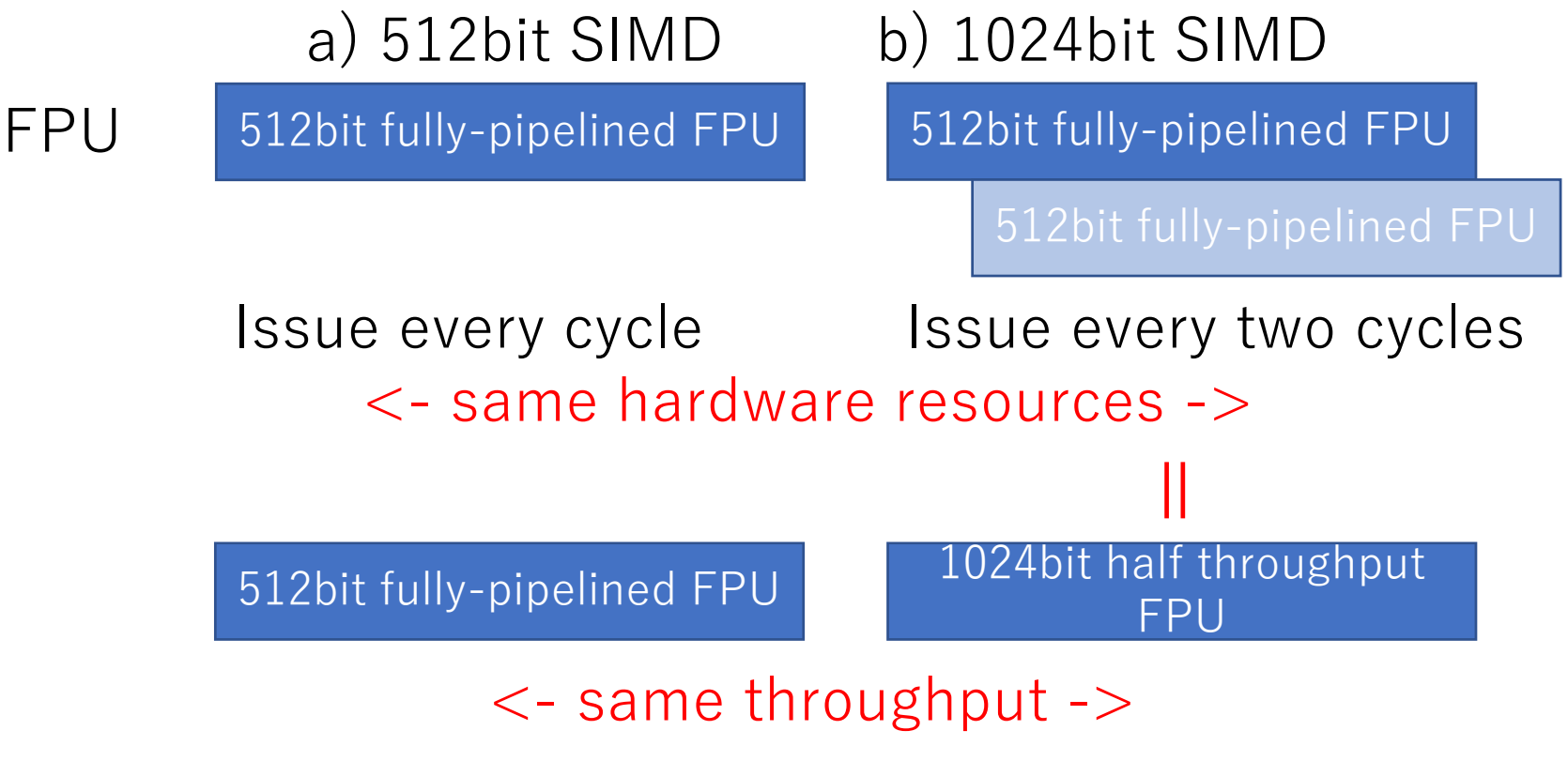
How different is the performance depending on the vector length ?

- SVE is very useful because vector length agnostic programming enables to run same binaries on different vector length.
- Wide SIMD using wide FPU improves peak performance but it is trivial.

We evaluate effects of vector length under almost same amount of hardware resources.

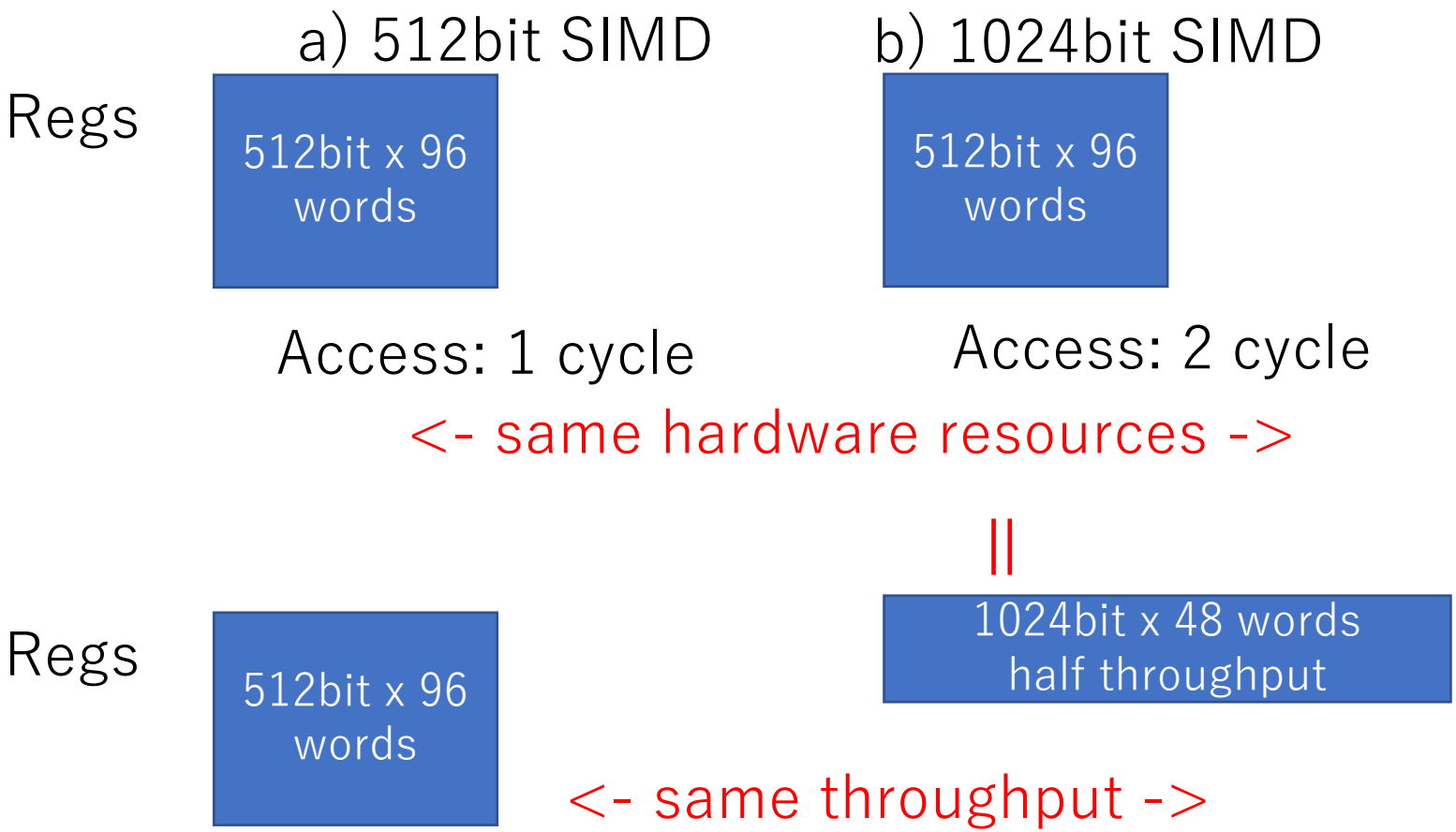
- Compare different vector length
- Fixed following resources, those are the major hardware resources for wide SIMD.
  - FPU resources
  - Register resources

How to keep resource size the same?



We can control the amount of hardware resources by the throughput.

How to keep resource size the same?



Outline

- Background and our research agenda
  - Overview of SVE
- Evaluation Environment
  - Gem5 simulator
  - Architectural parameters
  - Evaluated programs
- Evaluation Results
- Discussion
- Conclusion

Gem5 simulator

- Processor simulator
  - supports multiple ISA: Alpha, SPARC, x86, ARM
- CPU model
  - Atomic: instruction level simulation
  - O3: Out of Order pipeline simulation
    - Can estimate execution cycles
- Development “gem5-sve”
  - Atomic mode for SVE is provided by ARM Ltd.
  - We originally developed o3 mode for SVE based on it.

Gem5 | O3 pipeline

- Based on Alpha21264
  - ✓7 stages pipeline: Fetch, Decode, Rename, Issue, Execute, Write Back, Commit
- Parameter file
  - ✓can specify latency and operation width for each pipeline stage, number of arithmetic units, latency of each instruction, etc.
- Control of Throughput of each execution unit
  - ✓Original gem5 only supports fully-pipelined or not-pipelined.
  - ✓We extend the control of execution unit to be issued every n cycle, which called 1/n throughput.
  - ✓ex) 1024bit unit with TP=1/2

  - ✓realize same throughput and same hardware resources with 512bit fully pipelined unit in gem5

Gem5 | architecture parameters

- Based on O3_ARM_v7a.py that is preset parameter in gem5.
- Add instruction latency for SVE referred to NEON

Hardware parameters			
Clock Frequency	2.0GHz	# of core	1
L1 Dcache, Icache size	32kB	L2 cache size	2MB
Integer pipeline	2	Load/Store unit	1/1
Floating pipeline	2	Fetch width	3

OoO resource parameters	
IQ (Reservation Station)	64 (←32)
ROB (Re-order Buffer)	64 (←48)
LQ (Load Queue)	16
SQ (Store Queue)	16
Physical Vector Register	96 (new)

Evaluation Environment (1)

- In gem5, bit width of execution unit and register file set to the vector length, so we control by throughput.

vector length	LEN=4 (512bit)	LEN=8 (1024bit)	LEN=8(x2) (1024bit)
FPU throughput	512bit / cycle x 2 pipe	1024bit / 2cycle x 2 pipe	→
L1 throughput	512bit / cycle	1024bit / 2cycle	→
L2 throughput	256bit / cycle	→	→
Number of registers	512bit x 96	1024bit x 48	1024bit x 96
(architecture registers)	512bit x 32	1024bit x 32	→
(rename registers)	512bit x 64	1024bit x 16	1024bit x 64

Evaluation Environment (2)

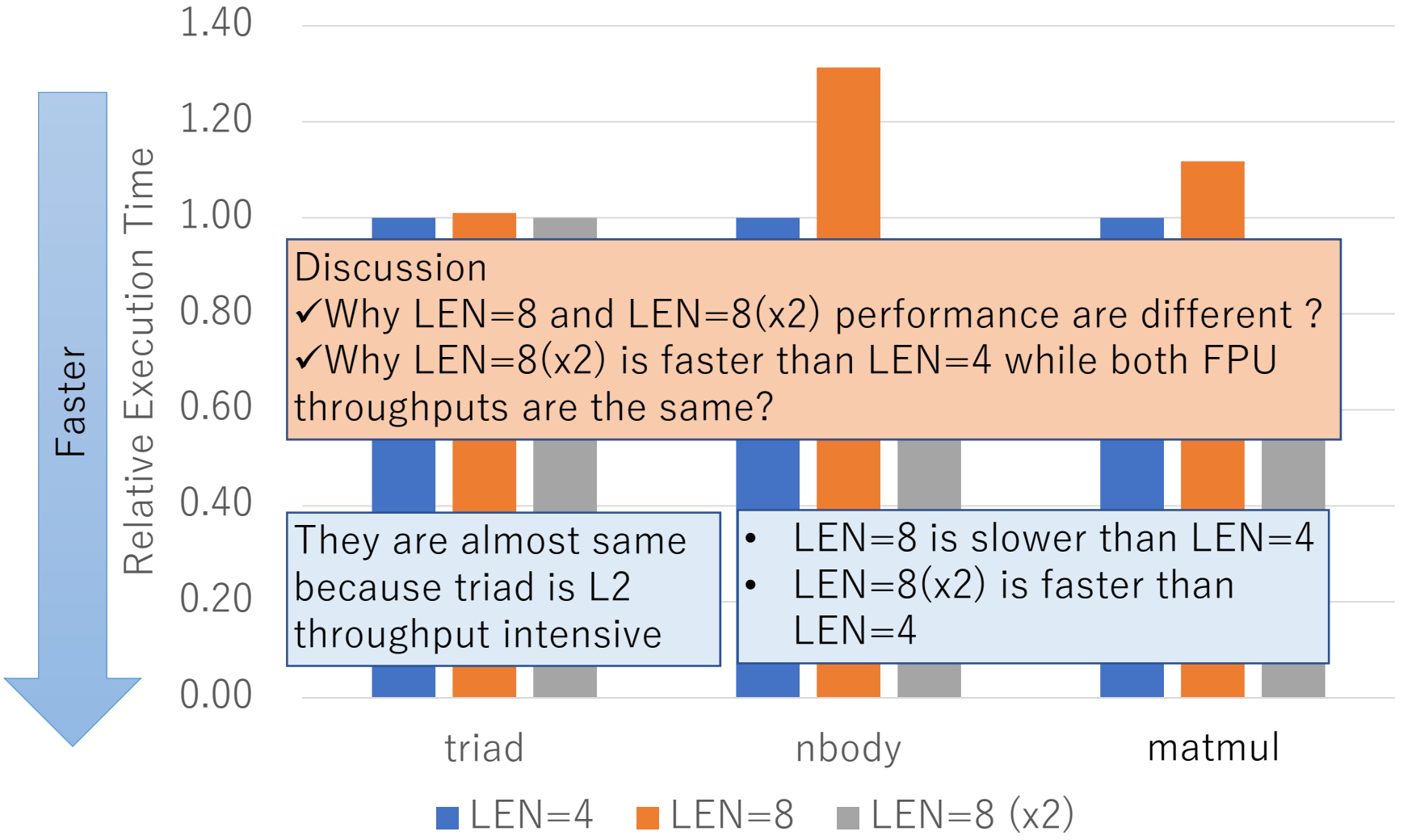
- Compiler
  - ARM clang version 1.1 (build number 15), -Ofast
  - Prototype compiler for SVE
- Evaluation kernels
  - Stream Triad: N=25600 on L2
    - ✓ L2 throughput intensive
  - N-body: N=512 on L1
    - ✓ Computation intensive , long instruction dependency
  - Matrix multiply: N=256 on L2
    - ✓ Theoretically computation intensive but current optimization is yet L1 throughput intensive

We use same binaries for different vector length

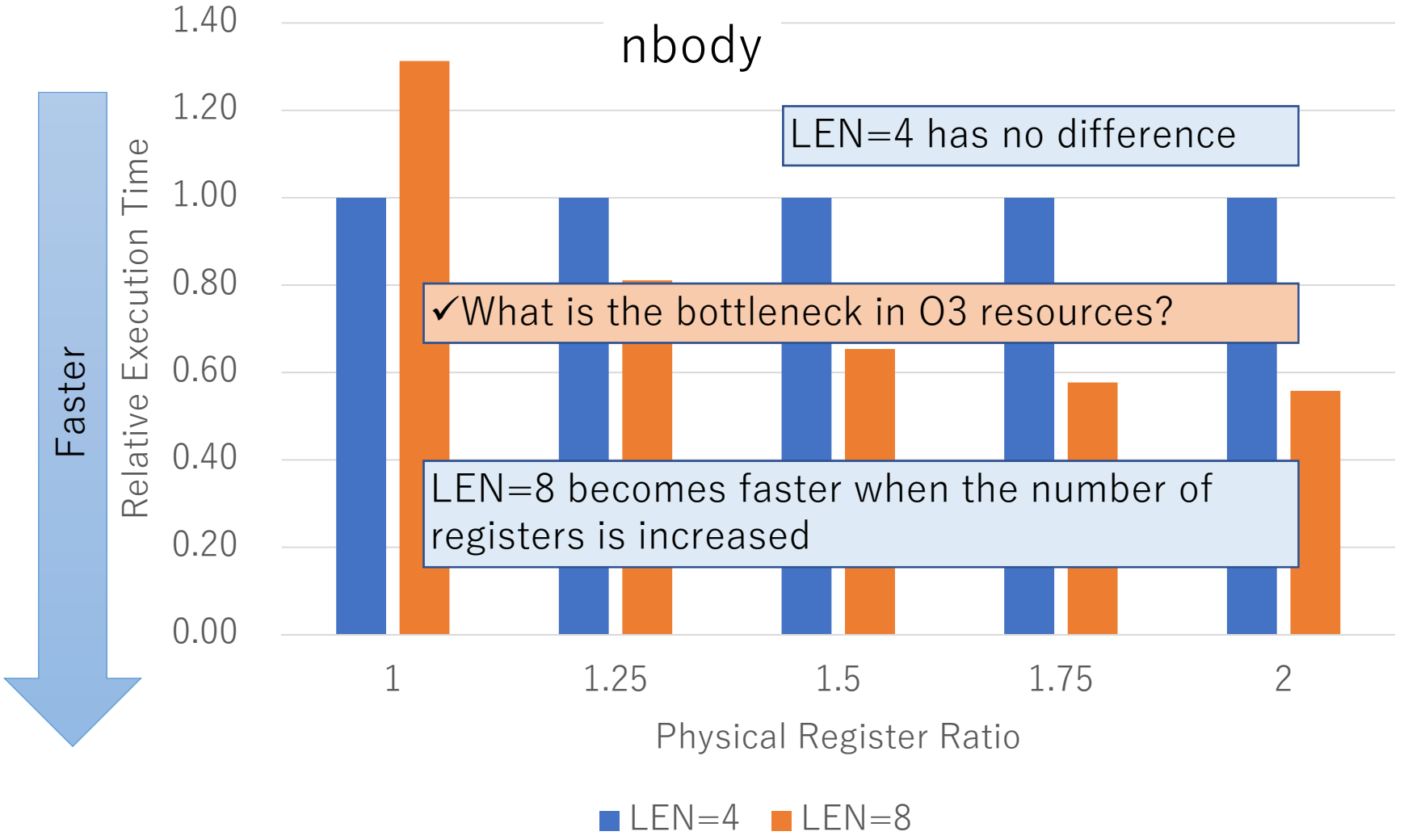
Outline

- Background and our research agenda
- Overview of SVE
  - Vector length agnostic programming
- Evaluation Environment
  - Gem5 simulator
  - Architectural parameters
  - Evaluated programs
- **Evaluation Results**
- Discussion
- Conclusion

Evaluation Results



Effects of # of registers



What is the bottleneck in O3 resources?

We checked resource-full cycles in execution.

- LEN=4 has no RegFull cycles, so changing Register size has no effect.
- LEN=8 with small registers has many RegFull cycles, so performance is degraded, but LEN=8 with enough registers has no RegFull cycles, and performance is improved.

nbody

LEN	Reg Ratio	IQFull	ROBFull	LQFull	SQFull	RegFull
LEN=4	1.00	0	118242	30	0	0
	1.25	0	118242	30	0	0
	1.50	0	118242	30	0	0
	1.75	0	118242	30	0	0
	1.25	0	0	0	0	49671
	1.50	0	3	56	0	57290
	1.75	0	3148	60	0	64852
	2.00	0	61151	86	0	0

✓LEN=8 resource balance was bad
✓to get good performance, LEN=8 will have enough registers.

FPU utilization

efficiency	LEN=4	LEN=8(x2)
Matmul	10%	17%
Nbody	52%	93%

- In matmul, the utilization with LEN=4 is only 10%, so there is many room for improvement even if LEN=8 uses twice cycles on FPU.
- In nbody, the utilization with LEN=4 is about 50%, there is also room for improvement.
- However both kernels have not been fully optimized yet, so we need re-evaluate using fully optimized version.

Conclusion

- Wide vector size over FPU element size will improve performance if there are enough rename registers and the utilization of FPU has room for improvement.
- But our evaluation is preliminary one, and many future works remains.
  - ✓ We should evaluate other O3 resources, such as reorder buffer and reservation station, effects on performance.
  - ✓ We should evaluate fully optimized programs.
  - ✓ We should use architecture parameters for HPC, such as fetch width, number of load units, etc.
  - ✓ We should evaluate more and larger programs.