Towards Lattice QCD on Fugaku: SVE Compiler Studies and Micro-Benchmarks in the RIKEN Fugaku Processor Simulator

Nils Meyer[†], *Tilo Wettig*[†], *Yuetsu Kodama*[#], *Mitsuhisa Sato*[#]

The Fugaku supercomputer, successor to the Japanese flagship K-Computer, will start operation in 2021. Fugaku incorporates the Fujitsu A64FX processor, which is the first hardware implementation supporting the Arm SVE instruction set, in this case a 512-bit version. Real hardware is not accessible today, but RIKEN has designed a simulator of the A64FX. We present micro-benchmarks relevant for Lattice QCD obtained in the RIKEN Fugaku processor simulator and compare three different SVE compilers.

[†] University of Regensburg (Regensburg, Germany), [#]RIKEN R-CCS (Kobe, Japan)

Motivation

- The Arm Scalable Vector Extension (SVE) features operands of length 128 to 2048 bits
- Fugaku, successor to the K-Computer, features the Fujitsu A64FX processor, which implements 512-bit SVE
- In the QPACE 4 project we pursue evaluation and enhancement of existing SVE software and upcoming SVE hardware for Lattice QCD applications [1, 2]

Fujitsu A64FX

- First CPU to support SVE [3, 4], available in 2020
- Clock frequency not disclosed yet
- At least 2.7 TFlops/s (DP), 32 GiB HBM2 memory
- 4 Core Memory Groups (CMG)
- 8 MiB L2 shared amongst 12 compute cores per CMG
- 64 kiB L1 private data cache per core
- Peak 32 Flops/cycle per core (dual-issue fma, DP)
- 2 · 512-bit load or 1 · 512-bit store from/to L1 per cycle

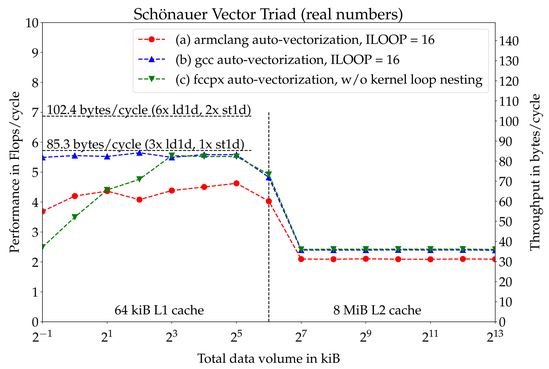
Benchmark

- Micro-benchmarks identify potential bottlenecks and guide code optimization strategies
- Schönauer vector triad

```
1 for(i=0; i<R; i++)           // R = number of repetitions
2     for(j=0; j<N; j+=ILOOP) // vector triad kernel
3         for(k=j; k<j+ILOOP; k++)
4             A[k] = B[k] + C[k] * D[k];
```

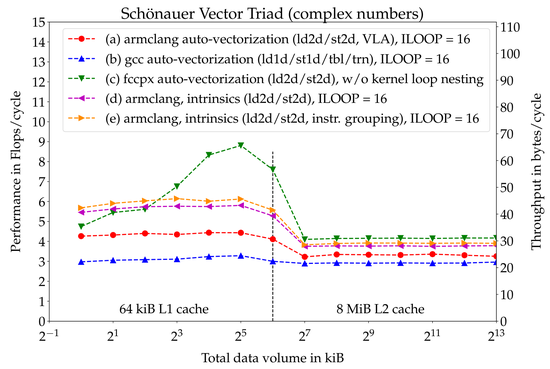
- We test 64-bit `double` and 128-bit `double _Complex` (two-element structure of real and imaginary part)
- Complex version is performance-relevant for Lattice QCD, i.e., the application of the Dirac operator on a spinor
- 1 addition and 1 multiplication (real), or 4 additions and 4 multiplications (complex) per triad
- 4 or 8 triads computed in parallel using 512-bit vectors
- 3 loads and 1 store per vector triad
- Benchmarks of a single core in the RIKEN Fugaku processor simulator [5], which is based on gem5 [6]
- Binaries generated by closed-source compilers armclang 19.2, Arm gcc 8.2.0 and Fujitsu fccpx 4.0.0

Real numbers



- 8 triads are processed in parallel
- armclang is not aware of the vector length in hardware and auto-vectorization yields a vector-length agnostic (VLA) loop, containing 3 ordinary load, 1 fma and 1 ordinary store instructions, resulting in suboptimal usage of the load ports (a)
- Arm gcc supports fixed-size 512-bit SVE and yields twofold loop unrolling, the loop body contains 6 load, 2 fma and 2 store instructions, resulting in optimal usage of the load ports (b)
- fccpx supports fixed-size 512-bit SVE and generates two- and eightfold loop unrolling if loop nesting is removed, with results comparable to gcc if the dimension of the streams is sufficiently large (c)
- No binary achieves the peak throughput, which is 85.3 bytes/cycle for partial use of load/store units (armclang) and 102.4 bytes/cycle for optimal use (gcc, fccpx)

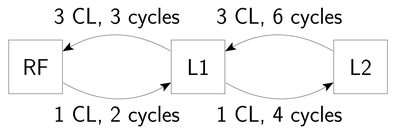
Complex numbers



- 4 or 8 triads are processed in parallel (using ordinary load/store `ld1d/st1d` or structure load/store `ld2d/st2d`, respectively)
- Structure load/store separates/reassembles real and imaginary parts into/from separate registers
- Auto-vectorization with armclang yields a VLA loop using structure load/store (a)
- Auto-vectorization with Arm gcc yields loop unrolling using ordinary load/store and permutations (b)
- Auto-vectorization with fccpx yields 6-fold loop unrolling, structure load/store and optimized instruction scheduling, beneficial if the data volume is sufficiently large (c)
- Intrinsics versions with twofold loop unrolling were also tested (d, e), in (e) we hide the instruction latencies
- Peak throughput of 102.4 bytes/cycle should be possible, and we attribute the lower performance to the (unknown) details of the A64FX simulator

L2 cache access model

- We model L1 as a single-port cache and assume L1 misses for all loads/stores
- 6 ordinary loads (`ld1d`) of 64 bytes each (3 cache lines) from L2 via L1 to register file (RF) take 6 + 3 cycles, 2 ordinary stores (`st1d`, 1 cache line) from RF to L1 take 2 cycles, and eviction to L2 takes 4 cycles
- Peak throughput is 8 · 64 bytes/15 cycles = 34.1 bytes/cycle
- Suboptimal usage of processor resources results in performance degradation



Cache line (CL) transfers between register file (RF), L1 and L2 caches proceed sequentially.

Summary and Outlook

We ran the Schönauer vector triad as single-core micro-benchmark in the RIKEN Fugaku processor simulator. We tested real and complex numbers, relying on the auto-vectorization capabilities of three compilers. Intrinsics implementations were also tested. We conclude that latency hiding is mandatory for performance on the A64FX, which can be achieved by loop unrolling and proper instruction scheduling. For processing complex numbers, structure load/store yields best performance.

In future work we will also investigate software prefetching and multi-core benchmarks accessing the HBM2.

Disclaimer

The result of the RIKEN Post-K processor simulator [5] is just an estimated value, and it does not guarantee the performance of the supercomputer Fugaku at the start of its operation. We use the processor simulator compiled on 11th of September 2019. The Fujitsu fccpx compiler is a pre-release version under development.

Acknowledgments

We acknowledge support from the HPC tools team at Arm.

References

[1] N. Meyer, D. Pleiter, S. Solbrig, and T. Wettig. "Lattice QCD on upcoming Arm architectures". In: *Proceedings of LATTICE 18* (2019), p. 316. arXiv: 1904.03927.

[2] N. Meyer, P. Georg, D. Pleiter, S. Solbrig, and T. Wettig. "SVE-enabling Lattice QCD Codes". In: *2018 IEEE International Conference on Cluster Computing (CLUSTER)* (2018), p. 623. DOI: 10.1109/CLUSTER.2018.00079. arXiv: 1901.07294. URL: https://ieeexplore.ieee.org/document/8514923.

[3] T. Yoshida. *Fujitsu High Performance CPU for the Post-K Computer*. Hot Chips 2018 [http://www.fujitsu.com/jp/Images/20180821hotchips30.pdf]. 2018.

[4] T. Shimizu. *Post-K Supercomputer with Fujitsu's Original CPU, A64FX Powered by Arm ISA*. [https://www.fujitsu.com/global/Images/post-k_supercomputer_with_fujitsu%27s_original_cpu_a64fx_powered_by_arm_isa.pdf]. 2018.

[5] Y. Kodama, T. Odajima, A. Asato, and M. Sato. "Evaluation of the RIKEN Post-K Processor Simulator" (2019), p. 6. arXiv: 1904.06451.

[6] gem5. [http://gem5.org/Main_Page].