STUDYING DIFFERENT BFS ALGORITHM IMPLEMENTATIONS WITH GEM5

Carlos Falquez[1], Nam Ho[1], Antonio Portero[1], Estela Suarez[1], Dirk Pleiter[2]

[1]Novel System Architectures Design, Jülich Supercomputing Centre, Forschungszentrum Jülich

[2]PDC Center for High Performance Computing, KTH Royal Institute of Technology

AHUG Workshop @ ISC23
May 25, 2023

STUDYING DIFFERENT BFS ALGORITHM IMPLEMENTATIONS WITH GEM5

Outline

- Motivate use of gem5 simulator as tool for developers assesing performance on different architectures
- Study effects of different architectural parameters:
  - Instruction Width, Number of LS Units
  - System Level Cache (SLC) Sizes
  - Memory Controller Latency
- BFS algorithm as illustrative case:
  - Algebraic (vectorized) and Vertex-Based implementations with different memory access patterns
  - Sensitive to different architectural parameters
- Understand differences in performance by comparing results on base and modified architectures.

STUDYING GRAPH ALGORITHMS WITH GEM5

Graph Algorithms

- Graph algorithms fundamental to a wide variety of applications
  - Machine learning
  - Network analysis
  - Bioinformatics
  - ...
- Data-dependent non-sequential access patterns
- Opportunities for optimization

Exploration with gem5

- gem5 is a modular cycle accurate computer architecture simulator.
- Use gem5 to study impact of microarchitectural parameters, cache sizes and memory latency on different implementations.

BREADTH-FIRST SEARCH (BFS)

Edge/Vertex-based approach

- Traverse graph advancing vertex frontier over connecting edges
- Ligra [4] is a state-of-the-art shared memory implementation of graph edge/vertex-traversal, optimized for both sparse and dense frontiers.
- The graph is represented by a list of edges sorted by adjacency. Each vertex stores a pointer to the start of its edges. (Adjacency array format).
- Graph structure determines memory access locality

BREADTH-FIRST SEARCH (BFS)

Algebraic approach

- Graph traversal can be represented as an iterative matrix-vector multiplication over various algebraic semirings [1].
- Graph characteristics determine matrix sparsity.
- Ongoing effort of implementing and optimizing SpMV based graph operations (see e.g. [2]).

Prototype vecBFS implementation

- Vectorized SpMV implementation based on SELL-C-sigma [3] sparse matrix format.
- Hand coded using SVE intrinsics.
- Strided memory access pattern.

VECBFS PERFORMANCE COMPARED TO LIGRA

Single thread speedup for varying problem size on AWS Graviton3 (average)

Scale	vecBFS [cycles]	Ligra BFS [cycles]	Speedup	A size	x size
12	1.88E+05	2.67E+05	1.42	512kB	16kB
13	3.55E+05	5.43E+05	1.53	1MB	25kB
14	7.63E+05	1.08E+06	1.41	2MB	50kB
15	1.63E+06	2.18E+06	1.34	4MB	100kB
16	3.60E+06	4.11E+06	1.14	8MB	180kB
17	7.92E+06	8.34E+06	1.05	16MB	350kB
18	1.82E+07	1.69E+07	0.93	32MB	680kB
19	4.55E+07	3.53E+07	0.78	64MB	1.3MB

- Measured average time-to-solution (TTS) for single-thread benchmark on c7g.metal
- Speedup relative to Ligra drops when matrix size goes beyond available L3 size (32MB).

GEM5 O3CPU MODEL

O3 Pipeline Model and Functional Units

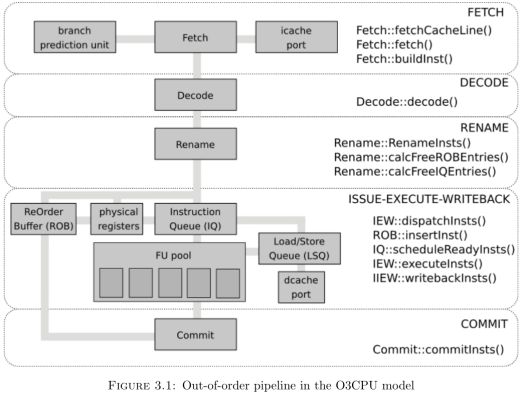


FIGURE 3.1: Out-of-order pipeline in the O3CPU model

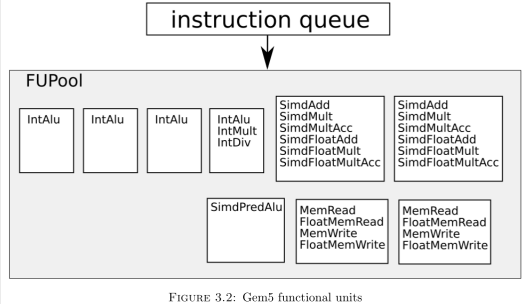
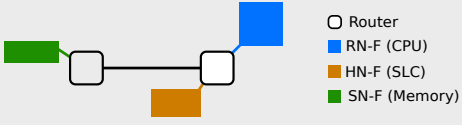


FIGURE 3.2: Gem5 functional units

Figures taken from: Bine Brank, PhD Thesis (2023)

GEM5 MODEL FOR SINGLE THREAD BENCHMARKS

Base configuration



CPU	1x 64-bit ARM (AArch64)
SVE	2x256
Private L1	64kB 4-Way
Private L2	1MB 8-Way
SLC	1x2MB 16-Way
Memory	30GB/s (Simple Memory)
NOC	Arm CHI Ruby + Garnet

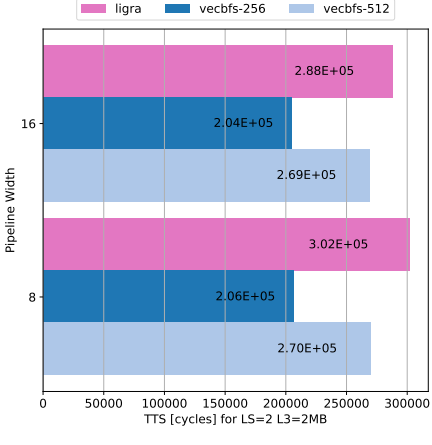
Simulation Parameters

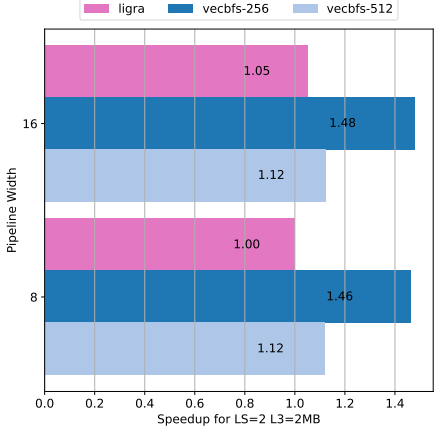
SVE Vector length	256,512
Pipeline Width	8,16
Load/Store Units	2,4
SLC Sizes	2,4,8 MB
Memory Latency	5..300ns

- Workloads: Scale 12,15 Kronecker Graphs (around 512kb and 4MB respectively)

PERFORMANCE COMPARISON: PIPELINE WIDTH

Small Problem Size S=12 (512kB)

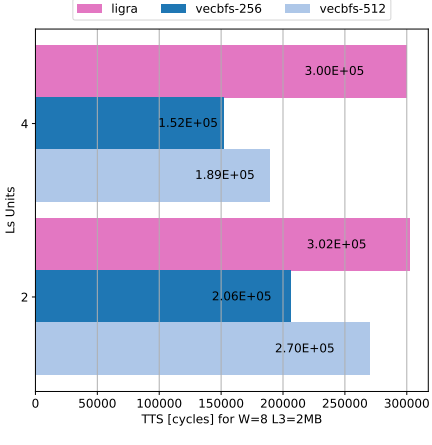


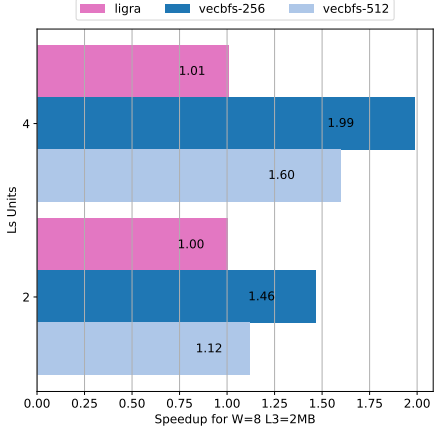


JÜLICH Forschungszentrum

PERFORMANCE COMPARISON: LS UNITS

Small Problem Size S=12 (512kB)

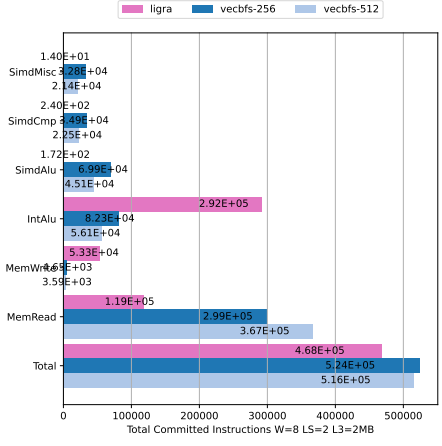


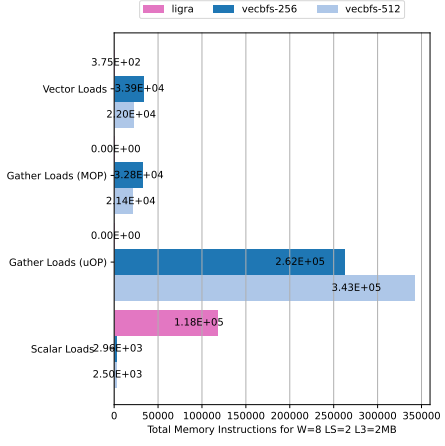


JÜLICH Forschungszentrum

COMMITTED INSTRUCTIONS

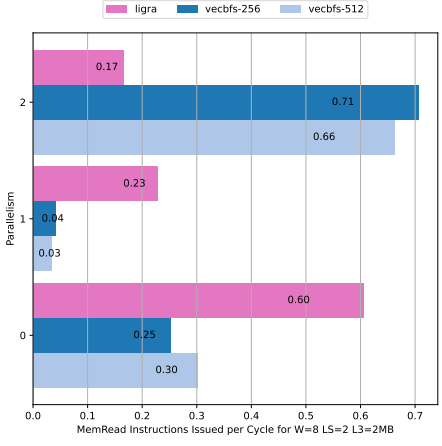
Small Problem Size S=12

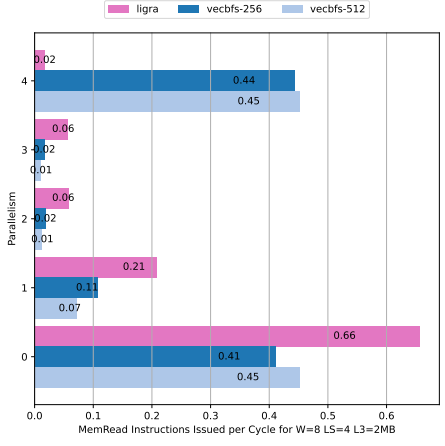




PARALLEL ISSUED MEMREAD INSTRUCTIONS

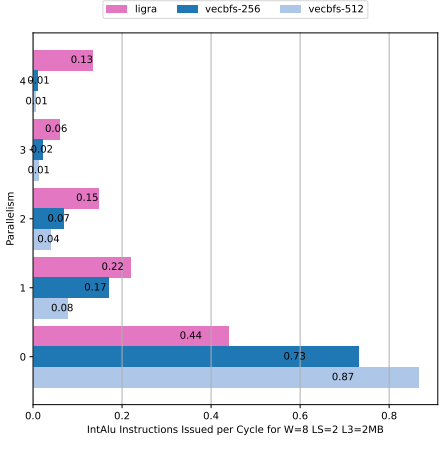
Effect of LS Units on ILP for S=12

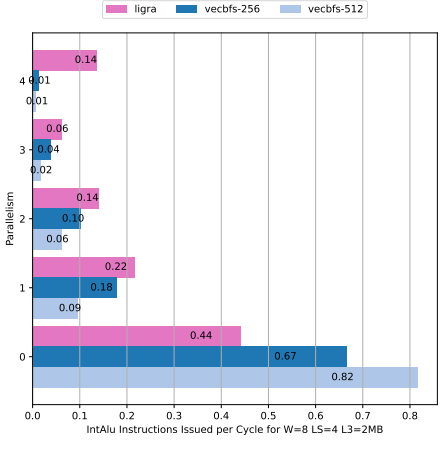




PARALLEL ISSUED INTALU INSTRUCTIONS

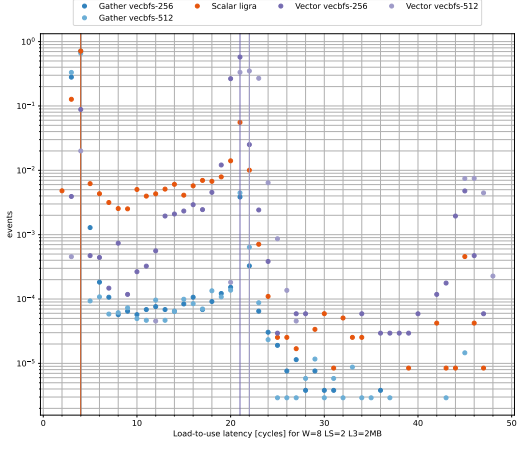
Effect of LS Units on ILP for S=12





LOAD-TO-USE LATENCIES

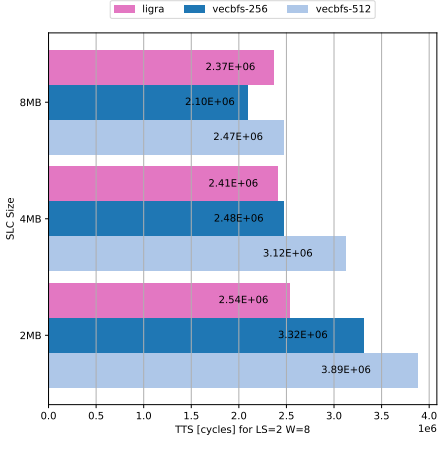
Small Problem Size S=12 (512kB) and SLC=2MB

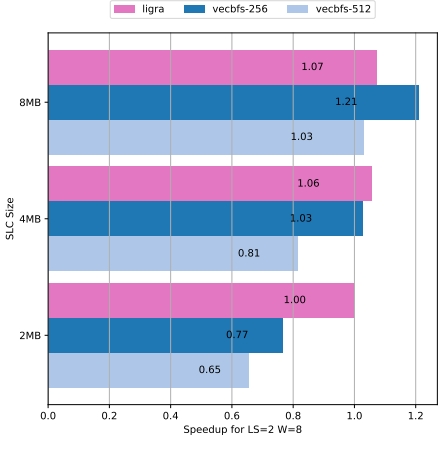


JÜLICH Forschungszentrum

PERFORMANCE COMPARISON: SLC SIZE

Large Problem Size S=15 (4MB)

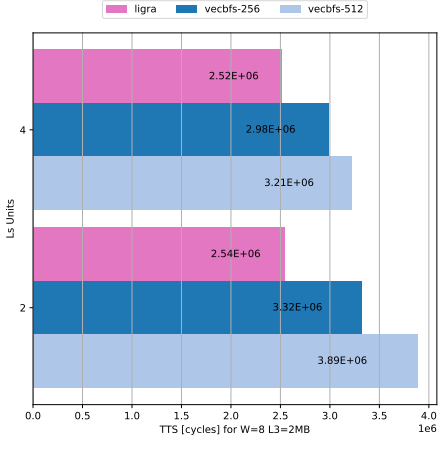


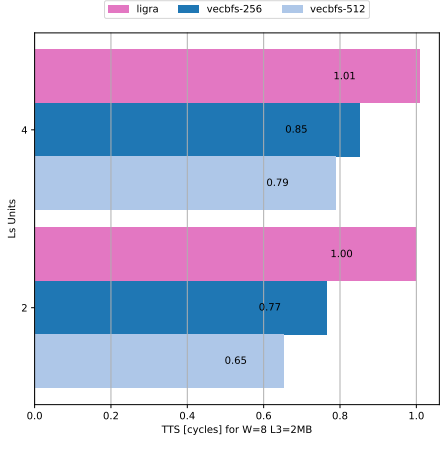


JÜLICH Forschungszentrum

PERFORMANCE COMPARISON: LS UNITS

Large Problem Size S=15 (4MB)

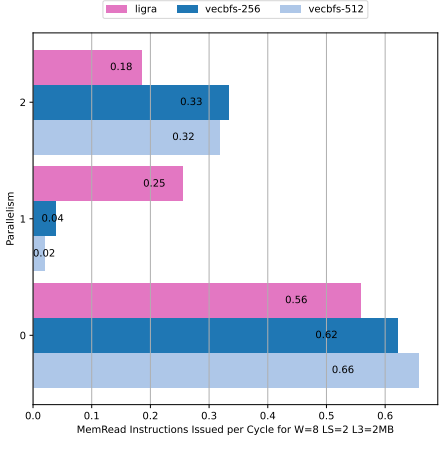


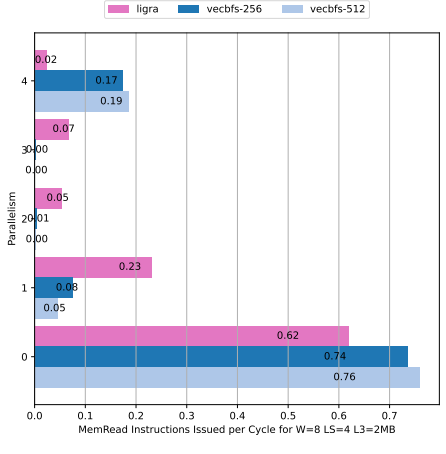


JÜLICH Forschungszentrum

PARALLEL ISSUED MEMREAD INSTRUCTIONS

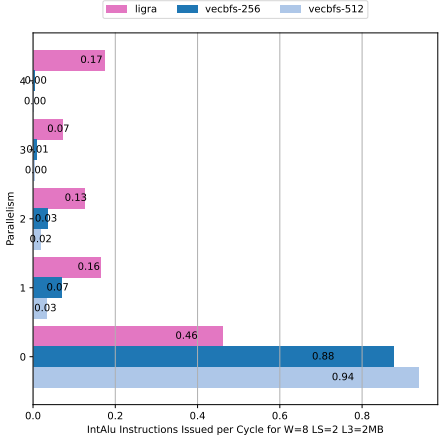
Effect of LS Units on ILP for S=15

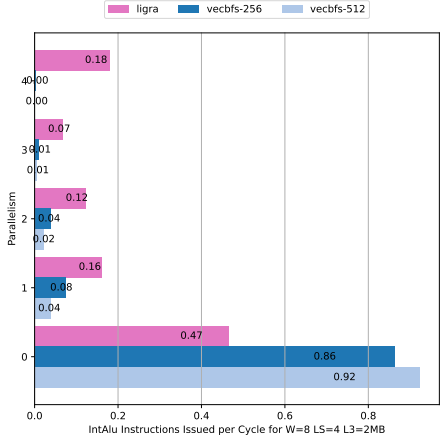




PARALLEL ISSUED INTALU INSTRUCTIONS

Effect of LS Units on ILP for S=15

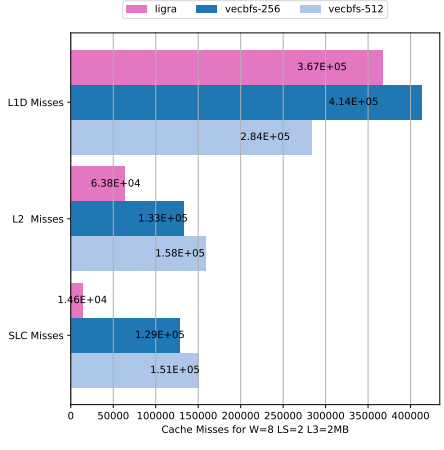


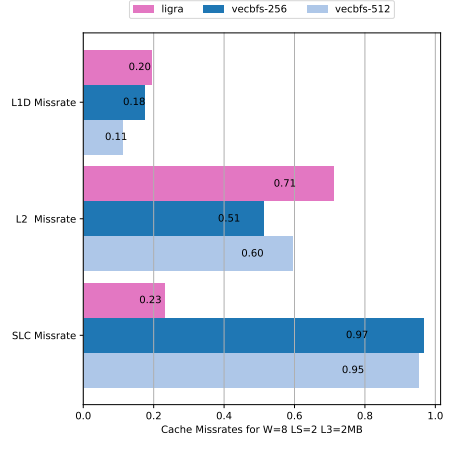


JÜLICH Forschungszentrum

CACHE UTILIZATION

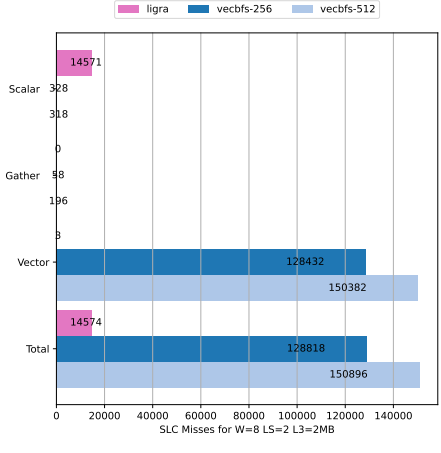
Large Problem Size S=15 (4MB)

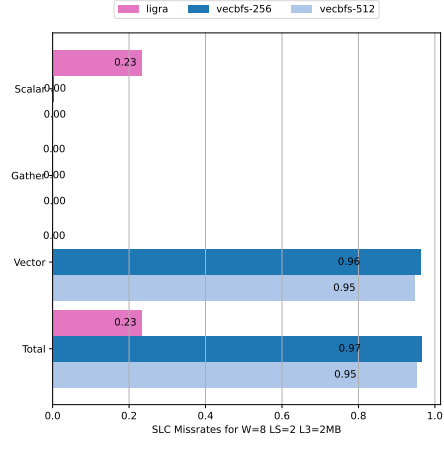




SLC UTILIZATION

Large Problem Size S=15 (4MB)

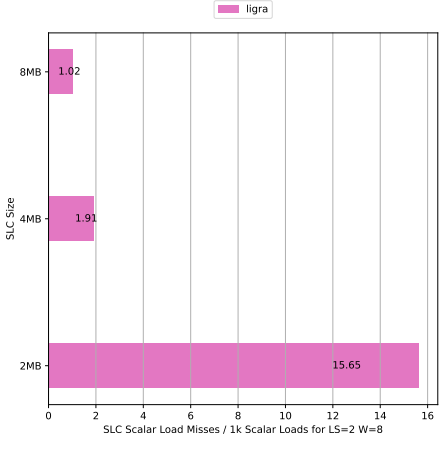


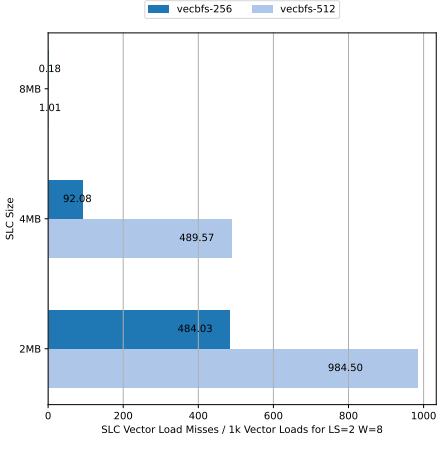


JÜLICH Forschungszentrum

SLC UTILIZATION

Large Problem Size S=15 (4MB)

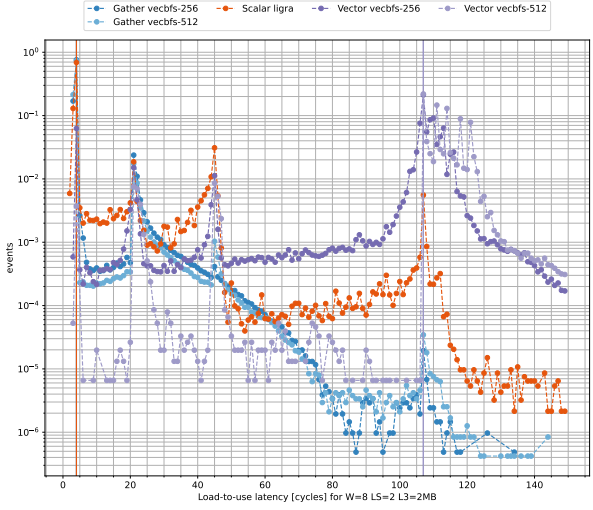




JÜLICH Forschungszentrum

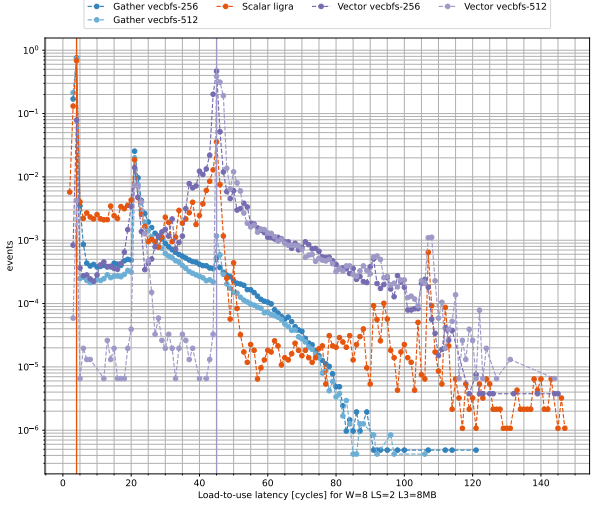
LOAD-TO-USE LATENCIES

Large Problem Size S=15 (4MB) and SLC=2MB



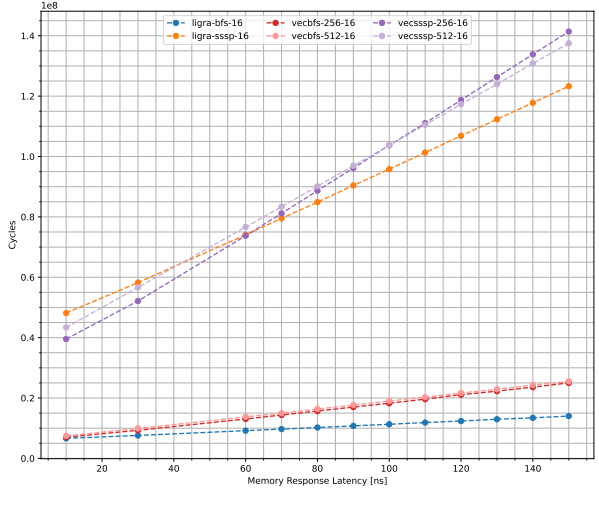
LOAD-TO-USE LATENCIES

Large Problem Size S=15 (4MB) and SLC=8MB



MEMORY LATENCY

Memory Latency sensitivity for S=16



JÜLICH Forschungszentrum

SUMMARY AND CONCLUSIONS

Summary

- Vectorized implementation benefits from high parallelism of gather load micro instructions, as long as the vector loads are utilizing the caches.
- But issue rate is still limited by available LS units, so larger vector lengths will need greater number of cycles to retire vector instructions which depend on the gather micro ops.
- Once the problem size is large enough, SLC misses from vector loads impose large latency penalties.
- Ligra's edge traversal shows low level of instruction parallelism, but much better cache utilization.

SUMMARY AND CONCLUSIONS

Conclusions

- gem5 can be a useful tool not only for architecture research but for software development.
- Need a reference architecture and benchmark to compare performance impact.
- By varying the reference architecture, test how performance changes between benchmarks.
- Don’t optimize for gem5! But use it to gain insight on performance differences between reference and target workloads.

Thank You!

REFERENCES

[1] Jeremy Kepner and John Gilbert, eds. Graph Algorithms in the Language of Linear Algebra. Society for Industrial and Applied Mathematics, 2011. DOI: 10.1137/1.9780898719918.

[2] Jeremy Kepner et al. “Mathematical foundations of the GraphBLAS”. In: 2016 IEEE High Performance Extreme Computing Conference (HPEC). 2016, pp. 1–9. DOI: 10.1109/HPEC.2016.7761646.

[3] Moritz Kreutzer et al. “A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units”. In: SIAM Journal on Scientific Computing 36.5 (2014), pp. C401–C423. DOI: 10.1137/130930352.

[4] Julian Shun and Guy E. Blelloch. “Ligra: A Lightweight Graph Processing Framework for Shared Memory”. In: Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2013, pp. 135–146. DOI: 10.1145/2442516.2442530.