ISC2023- AHUG WORKSHOP


# MEMORY PREFETCHING EVALUATION USING GEM5 SIMULATIONS

NAM HO, CARLOS FALQUEZ, ANTONIO PORTERO, ESTELA SUAREZ,

NOVEL SYSTEM ARCHITECTURE DESIGN, JÜLICH SUPERCOMPUTING CENTRE, FORSCHUNGSZENTRUM JÜLICH

DIRK PLEITER

PDC CENTER FOR HIGH PERFORMANCE COMPUTING, KTH ROYAL INSTITUTE OF TECHNOLOGY

# OUTLINE

- Motivation
- Hardware memory prefetching
- Simulation methodology
- Application selection and memory access pattern analysis
- Prefetch evaluation
- Conclusion & future study

# MOTIVATION

- Emerging HPC applications are demanding advanced computing systems
  - Need both high **processing capability** as well as high **memory bandwidth**
- Recent innovations in the HPC sector
  - Arm-based high-end processors with SVE technology have entered the HPC sector (e.g. Fugaku supercomputer)
  - New memory technologies (e.g. HBMx) make available high bandwidths
- The memory wall
  - Scaling memory bandwidth with processing capability remains a challenge
  - Memory prefetching is well-known technique to hide long-latency memory accesses by predicting the data accesses and preloading data ahead in the cache that is expected to be requested in the near future

- Work objectives
  - Investigate **the effect of hardware prefeching techniques** on scientific applications in recent Arm-based high-end processors
  - Leverage gem5 capabilities and **developed a gem5-model of the latest Arm Neoverse V1 design with HBM2** based on the recently released Graviton3

# HARDWARE MEMORY PREFETCHING

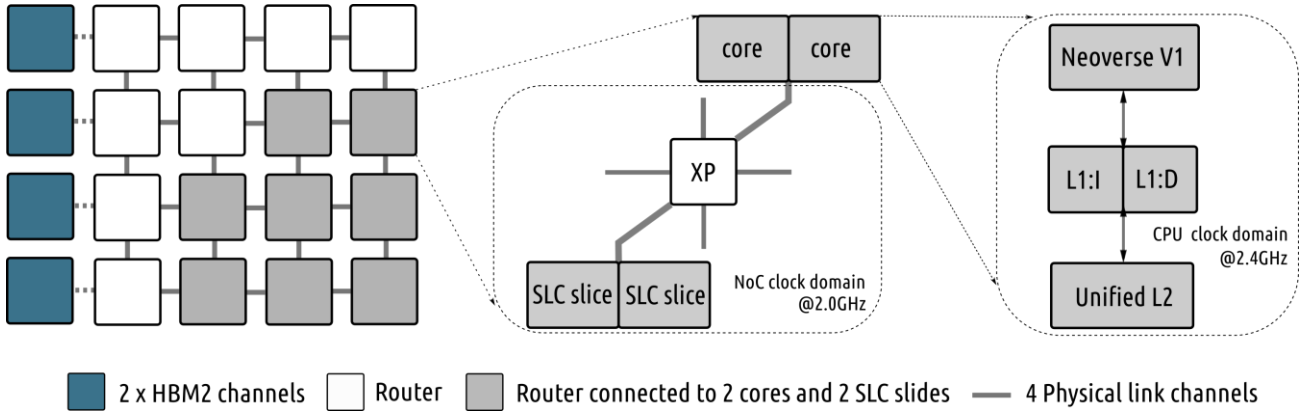
- A well-known technique to eliminate processor stalls due to long-latency memory accesses
  - Reduce cache misses by predicting the data accesses and fetching ahead data into cache that is expected to be requested in the near future
- Memory access pattern prediction
  - Stride (or delta) access pattern
    - Constant stride sequence: [A,A+k,A+2k,…]
    - Multi-delta sequence: [A,A+k,A+m,…B,B+k,B+m]
  - Address repetition pattern
    - [A,B,C,…,A,B,C]
- Prefetch effectiveness
  - Depend on prediction accuracy, prefetch coverage
  - Aggressive prefetching

** Sparsh Mittal. 2016. A Survey of Recent Prefetching Techniques for Processor Caches. ACM Comput. Surv. 49, 2, Article 35 (Nov. 2016).*

# SIMULATION METHODOLOGY: THE GEM5-MODEL OF NEOVERSE V1 ARCHITECTURE

- The 16-core Neoverse V1 architecture (**ExpArch**)
  - NoC:
    - Ruby Garnet + AMBA–CHI
    - 4x4 Mesh
    - 4 physical link channels for 4 VNETs
  - 16 cores + 16x1MB-SLCs
    - 2xSVE engines per core
    - Core's clock: 2.4GHz
    - System/NoC clock: 2.0GHz
  - Memory
    - HBM2 (8 x 38.4GB/s/channels)




North West quadrant of the modeled architecture: The left-most figure shows 16 cores, 16 SLC slices, and 8 HBM2 channels placed in a 2D mesh topology; The middle of the figure outlines a router architecture connecting to 2 cores and 2 SLC slides; The right most figure outlines core's cache hierarchy

Lilia Zaourar et.al. "*Multilevel simulation-based co-design of next generation HPC microprocessors.*" PMBS 2021: 18-29

# SIMULATION METHODOLOGY: QUANTITATIVE COMPARISON WITH REAL PLATFORMS

- To get better confidence in the gem5-model, we quantitatively compared the simulation results with N1SDP, and Graviton3
  - Two additional gem5-models
    - Gem5-model (N1) for N1SDP,
    - Gem5-model (G3) for Graviton3

- Microbenchmarking
  - FMLA-bench: A simple benchmark developed to test the maximum number of FP operations on an SVE machine

**Overview of three system architectures**

| | Core-Arch | NoC | Memory |
|---|---|---|---|
| Graviton3 | 64xNeoverse-V1 | AMBA-CHI | DDR5 (8x38.4GB/s/channel) |
| N1SDP | 4xNeoverse-N1 | AMBA-CHI | DDR4 (2x25.6GB/s/channel) |
| ExpArch | 16xNeoverse-V1 | AMBA-CHI | HBM2-Model (8x38.4GB/s/channel) |

**FMLA-bench results**

| Platform | Work size | Loop | CPU clock | Glop/s | Flop/cycle (% of peak) | Cycles | Retired instrs | SIMD instrs |
|---|---|---|---|---|---|---|---|---|
| gem5-model (G3) | 1000000 | 5 | 2.6 | 41.5 | 15.96 (99.7) | 7.51E+07 | 1.8E+08 | 1.5E+08 |
| Graviton3 | 1000000 | 5 | 2.6 | 41.5 | 15.96 (99.8) | 7.76E+07 | 1.8E+08 | 1.5E+08 |
| gem5-model (N1) | 1000000 | 5 | 2.6 | 20.8 | 8 (100) | 1.50E+07 | 3.30E+07 | 3.00E+07 |
| N1SDP | 1000000 | 5 | 2.6 | 20.8 | 8 (100) | 1.50E+07 | 3.30E+07 | 3.00E+07 |
| gem5-model (ExpArch) | 1000000 | 5 | 2.4 | 38.3 | 15.96 (99.7) | 7.51E+07 | 1.8E+08 | 1.5E+08 |

Simulation results match with that on the real platforms

# SIMULATION METHODOLOGY: QUANTITATIVE COMPARISON WITH REAL PLATFORMS (CONT.)

- To get better confidence in the gem5-model, we quantitatively compared the simulation results with N1SDP, and Graviton3

- Microbenchmarking
  - Triad-kernel (from STREAM benchmark)

```
a[i] = S*b[i] + c[i]
```

  - Read-only kernel

```
sum += a[i]
```

## N1SDP comparison with prefetchers disabled

| Platform | L1D cache miss | L2 cache miss | SLC cache miss | Retired instrs | L1D cache access | Bandwidth (GB/s) |
|---|---|---|---|---|---|---|
| **Read-only kernel - With prefetchers disabled** | | | | | | |
| gem5-model (N1) | 4.00E+07 | 5.03E+06 | 5.03E+06 | 1.60E+08 | 4.04E+07 | 1.83 |
| N1SDP | 5.00E+06(*) | 5.00E+06 | 5.00E+06 | 1.60E+08 | 4.00E+07 | 1.77 |
| Triad kernel - With prefetchers disabled | | | | | | |
| gem5-model (N1) | 8.61E+07 | 1.51E+07 | 1.51E+07 | 2.80E+08 | 1.21E+08 | 2.95 |
| N1SDP | 1.00E+07(*) | 1.00E+07(**) | 1.00E+07(**) | 2.80E+08 | 1.20E+08 | 2.57 |
| gem5-model (N1) (Stride=4) | 2.50E+07 | 1.51E+07 | 1.50E+07 | 7.01E+07 | 3.10E+07 | 2.15 |
| N1SDP (Stride=4) | 1.50E+07 | 1.50E+07 | 1.50E+07 | 7.00E+07 | 3.00E+07 | 1.68 |

(*) Diff for a factor 8x due to cache-line refill counting in N1SDP
(**) Diff. for a factor of 1.5x due to the effect of write-streaming mode in N1SDP

## Three platforms comparison with prefetchers enabled

| Platform | 1-thread (GB/s) | 4-threads (GB/s) | 16-threads (GB/s) |
|---|---|---|---|
| gem5-model (N1) | 13.1 | 30.83 | - |
| N1SDP | 19.72 | 29.29 | - |
| | | | |
| gem5-model (G3) | 39.11 | 115.75 | 173.66 |
| Graviton3 | 44.86 | 162.04 | 214.04 |
| ExpArch | 38.01 | 117.16 | 169.04 |

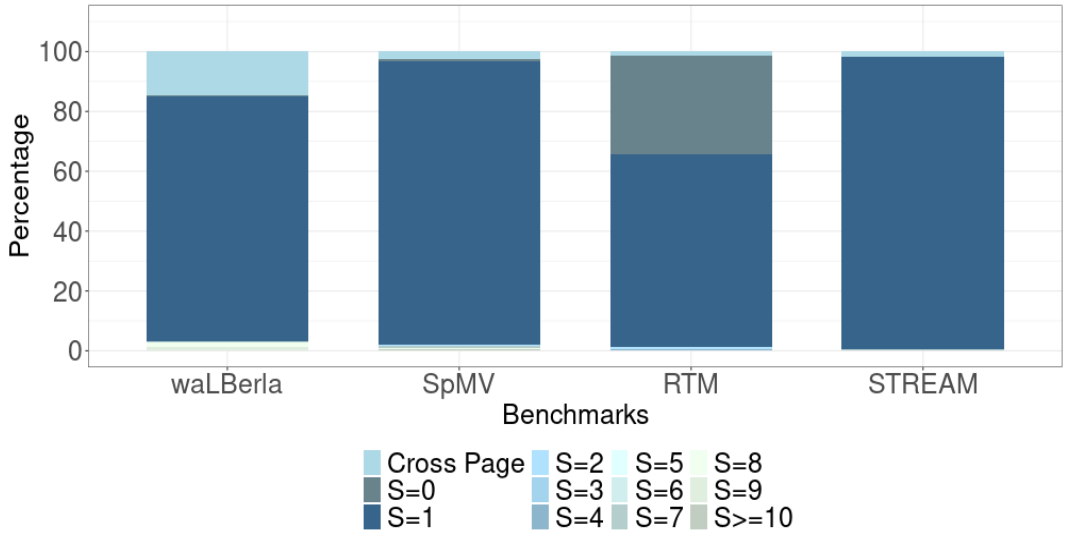
# SELECTED HPC APPLICATION KERNELS & THE CHARACTERIZATION

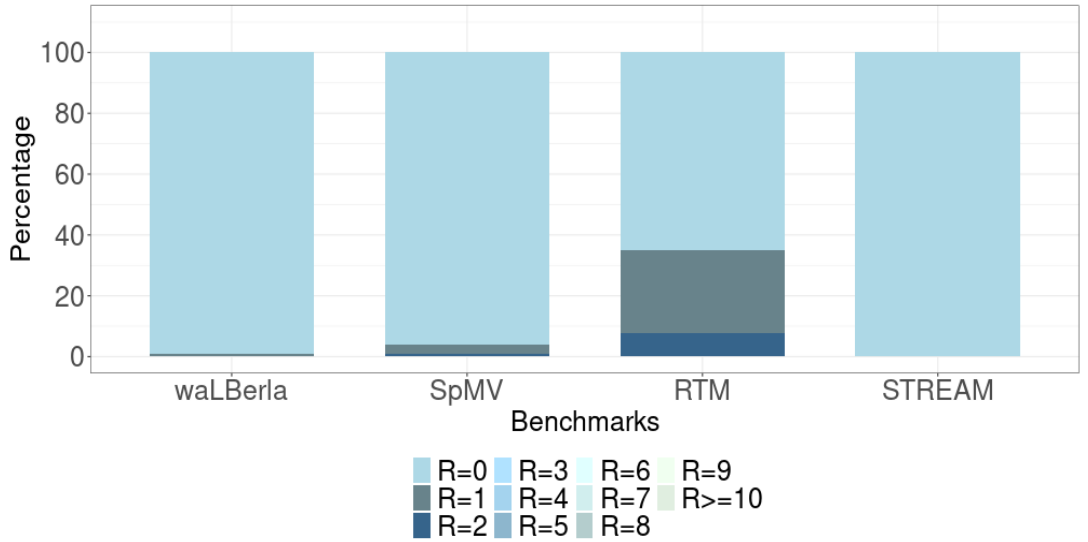
| | Description | (Pseudocode) Kernel code (via SVE optimization) | Access pattern |
|---|---|---|---|
| TRIAD | The triad kernel from STREAM benchmark | `// Triad`<br>`for i = 0; i < N; i++)`<br>`  a[i] <- S* b[i] + c[i]` | Constant stride |
| SPMV (MiniFE) | Matrix-vector product $y = y + Ax$, where $x$ and $y$ are dense vectors<br>We evaluated the hand-optimized version using SVE intrinsics (*)<br><br>*(*) Bine Brank et.al. "Porting Applications to Arm-based Processors. In 2020 CLUSTER. 559–566.* | `// Sliced ELLPACK format`<br>`for (s = 0; s < N; s += S)`<br>`  for (r = s; r < s + S; r++)`<br>`  for (i = slice_start[s/S] + r - s;`<br>`  i < slice_start[s/S + 1]; i += S)`<br>`  y[r] += x[column[i]] * value[i];` | Constant stride + random access with indirect indexing |
| waLBerla | A framework for computational fluid dynamics simulations based on the LBM.<br>We used the waLBerla use-case *UniformGridBenchmark* | `// Stencil code`<br>`D3Q19 model implementation` | Multi-stride |
| RTM | Reverse-Time Migration (RTM) is a well known method used in the reconstruction of geological subsurface structures.<br>We evaluated Isotropic wave equation (ISO) kernel | `// Stencil code`<br>`8th order Laplacian Stencil,`<br>`a 3D 27-point stencil` | Multi-stride |

| Benchmark | $I_{mem}$ (Byte) | $I_{fp}$ (Flop) | AI= (flops/byte) | Size | Memory footprint (MiB) | Miss-rate(%) L2 | Miss-rate(%) SLC | MPKI L2 | MPKI SLC | IPC | BW Util. (%peak) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| waLBerla | N.76.8 | N.189 | 0.31 | N=96.96.96 | 256 | 97.53 | 87.57 | 46.88 | 41.05 | 10.93 | 31.71 |
| SpMV | $N_{nnz}$.(8+4) | $N_{nnz}$.2 | 0.17 | N=63.63.63 | 80 | 91.3 | 92.82 | 58.1 | 53.92 | 8.1 | 23.6 |
| RTM | 16.2 | 34 | 2.1 | N=128.256.256 | 128 | 93.39 | 44.12 | 41.17 | 18.16 | 12.41 | 13.57 |

# HPC APPLICATION KERNELS & CHARACTERIZATION (CONT.)




* STRIDE pattern (Spatial Locality [1]):
- Calculation by searching back for a window of **`W`** memory accesses and calculate stride **`s`** as the minimum distance between the current access address and its nearest neighbor among those in the window.
  - **`W`** = 64 (units in 64-byte cache-line).

* REUSE pattern (Temporal Locality[1]):
- Calculation by searching back for a window of **`L`** memory accesses and calculate the number of times access address repeated
  - **`L`** = 64 (units in 64-byte cache-line).

[1] J. *Weinberg, M. O. McCracken, E. Strohmaier and A. Snavely, "Quantifying Locality In The Memory Access Patterns of HPC Applications," SC '05, pp. 50-50, doi: 10.1109/SC.2005.59.*

# HARDWARE PREFETCHERS USED FOR EVALUATION

| Hardware prefetcher configuration at L2 for evaluation | | | | | | |
|---|---|---|---|---|---|---|
| **HW Prefetcher (*)** | Description | Trigger on access | Cache snoop | Queue size | Specific parameters | Aggressive prefetching parameters |
| **Tagged** | Next-line prefetching | True | True | 64 | - | Varying prefetch degree [1..32] |
| **Stride** | Constant PC-stride detection | | | | table entries = 128 | |
| **AMPM**[1] | Delta-pattern detection | | | | hot zone size=2KiB, map table entries = 256 | |
| **SPP**[2] | Delta-pattern detection | | | | prefetch threshold = 0.25, lookahead threshold=0.01, signature table entries=256, pattern table entries=4096 | |

*(*) Prefetcher names are taken from the gem5*

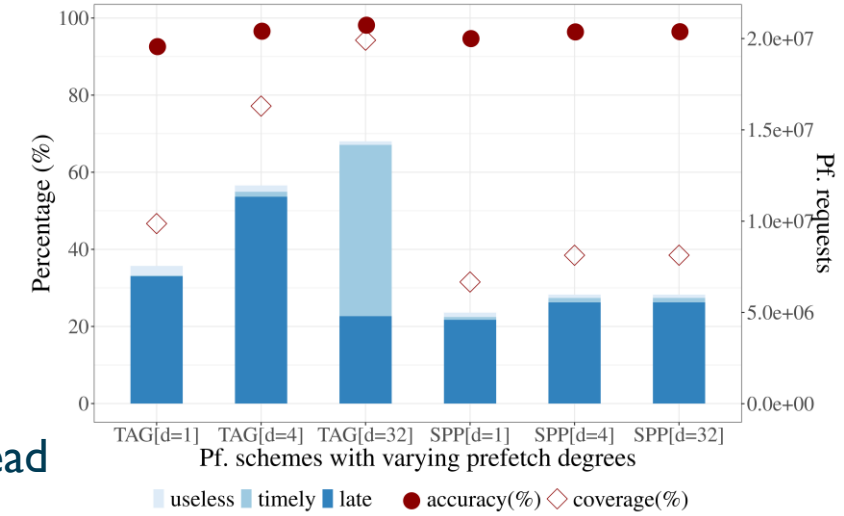
*[1] Ishii, Y. et. al. "Access map pattern matching for high performance data cache prefetch" Journal of Instruction-Level Parallelism, 13, 1-24 (2011).*
*[2] Kim J. et. al. "Path confidence based lookahead prefetching" MICRO-49, Article 60, 12 (2016).*

# PREFETCH COUNTERS & METRICS

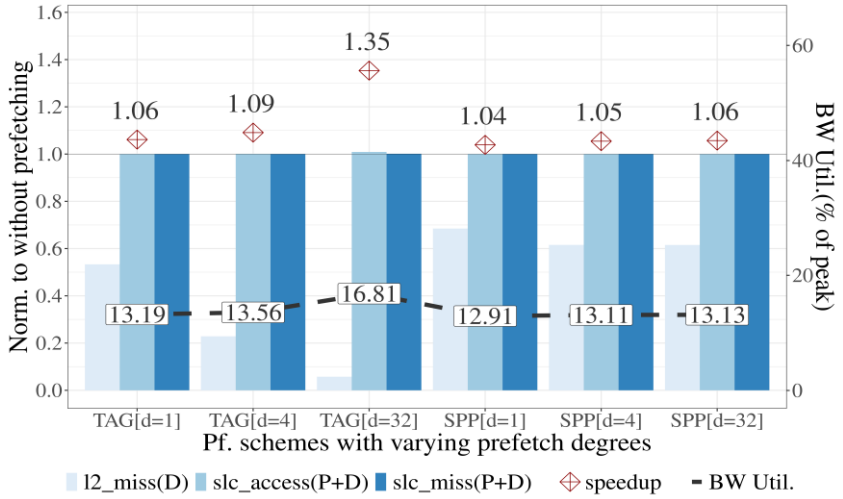
- *Number of prefetches*: A counter tracks number of prefetches actually sent to the next memory level.
- *Late prefetches*: A counter tracks number of accurate prefetch requests are being in the in-flight memory request buffer (TBE) waiting for the returned data by the time needed by the demand miss.
- *Timely prefetches*: A counter tracks number of accurate prefetch requests fetching the needed data by a demand miss in time.
- *Useless prefetches*: A counter tracks number of inaccurate prefetches, for which the fetched data in cache are not used.
- Accuracy: $\frac{timely\ prefetches + late\ prefetches}{number\ of\ prefetches}$
- Coverage: $\frac{timely\ prefetches + late\ prefetches}{timely\ prefetches + late\ prefetches + demand\ misses}$

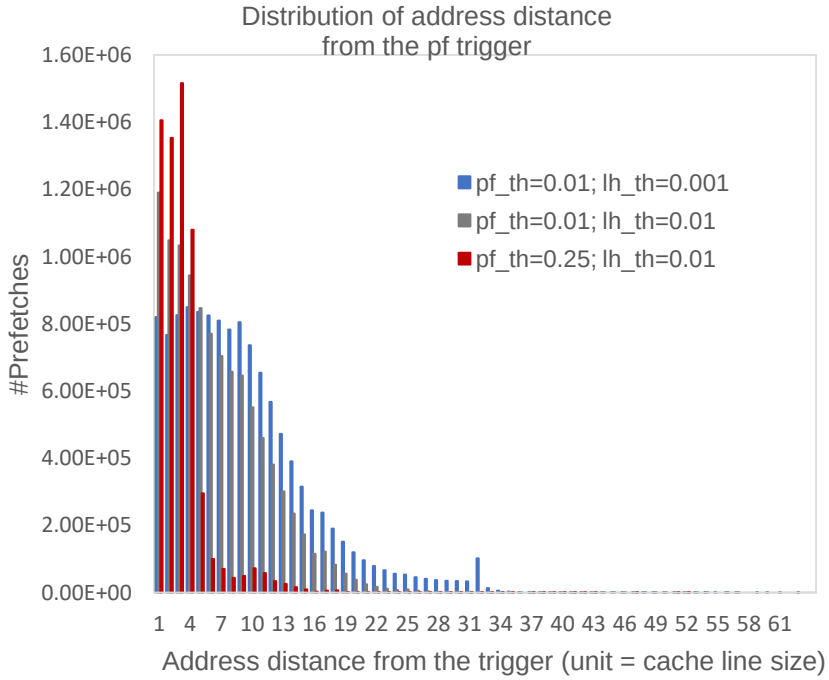
# EFFECTS OF AGGRESSIVE PREFETCHING

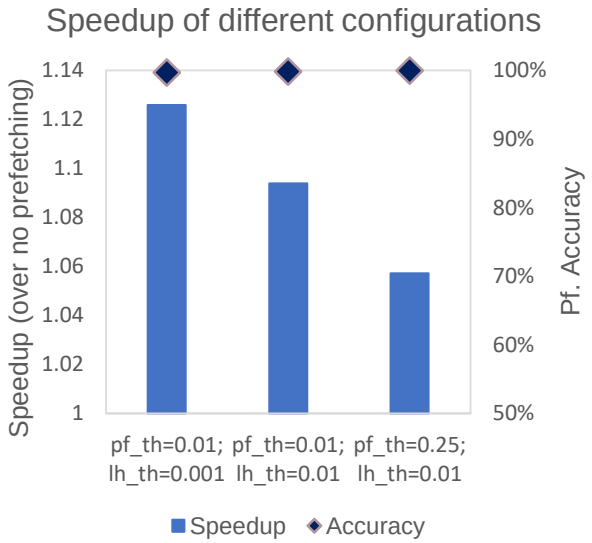



1-thread




- Triad kernel (STREAM benchmark)
  - **BW utilization saturated without prefetching** (74% of peak; **16-thread** evaluation)
  - High aggressiveness shows performance improvement for **single-thread**

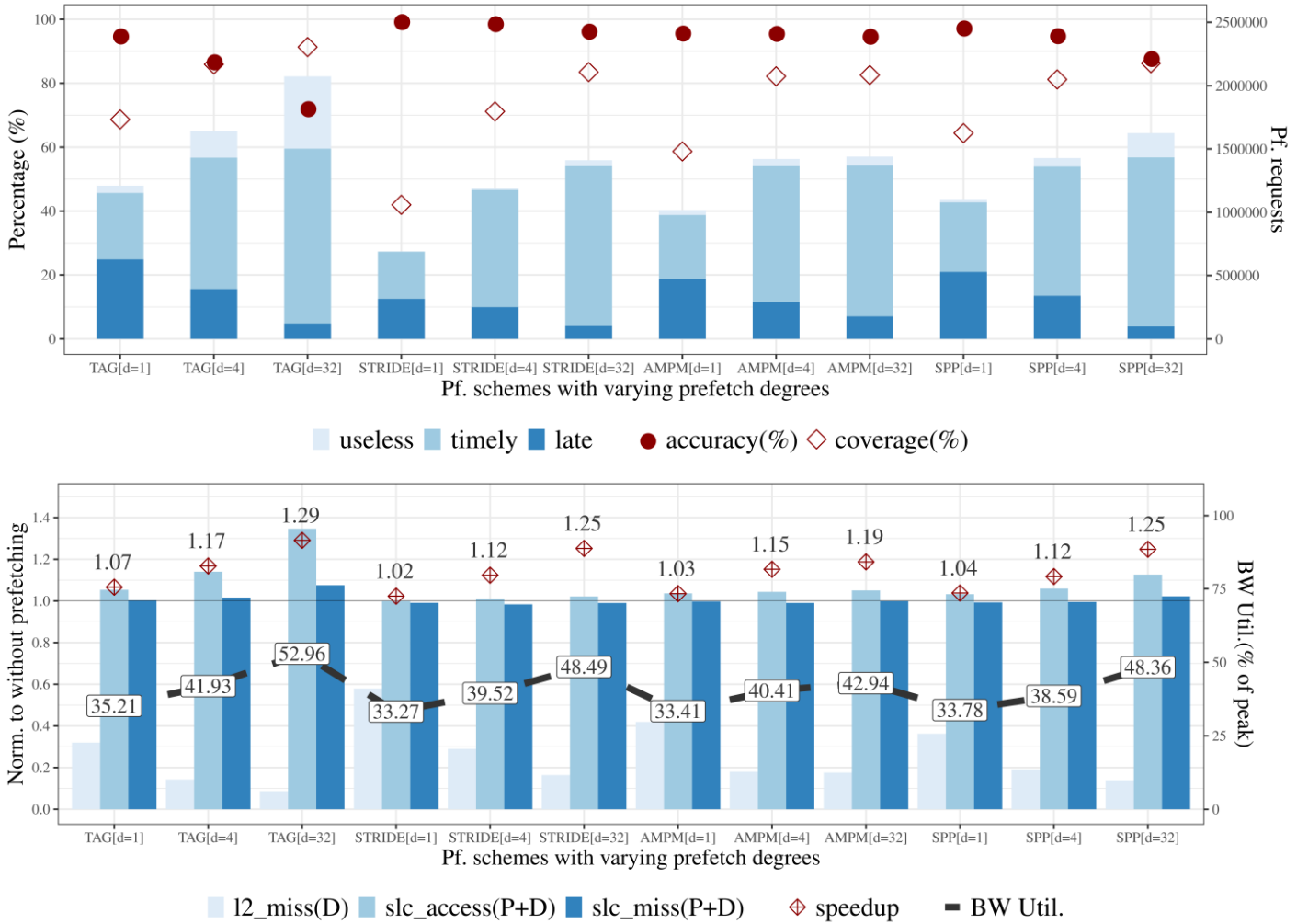



SPP exploration with diff. threshold configurations

# EFFECTS OF AGGRESSIVE PREFETCHING (CONT.)

- WaLBerla
  - Strongly benefits from aggressive prefetching
    - Low conservative prefetching (e.g. degree = 1) does not work well (~50% late prefetches)
    - Prefetching with degree = 32, all prefetchers show coverage of over 80%, and the Next-line (TAG) delivers speedup up to 1.29
  - More useless prefetches are observed with the Next-line (TAG)

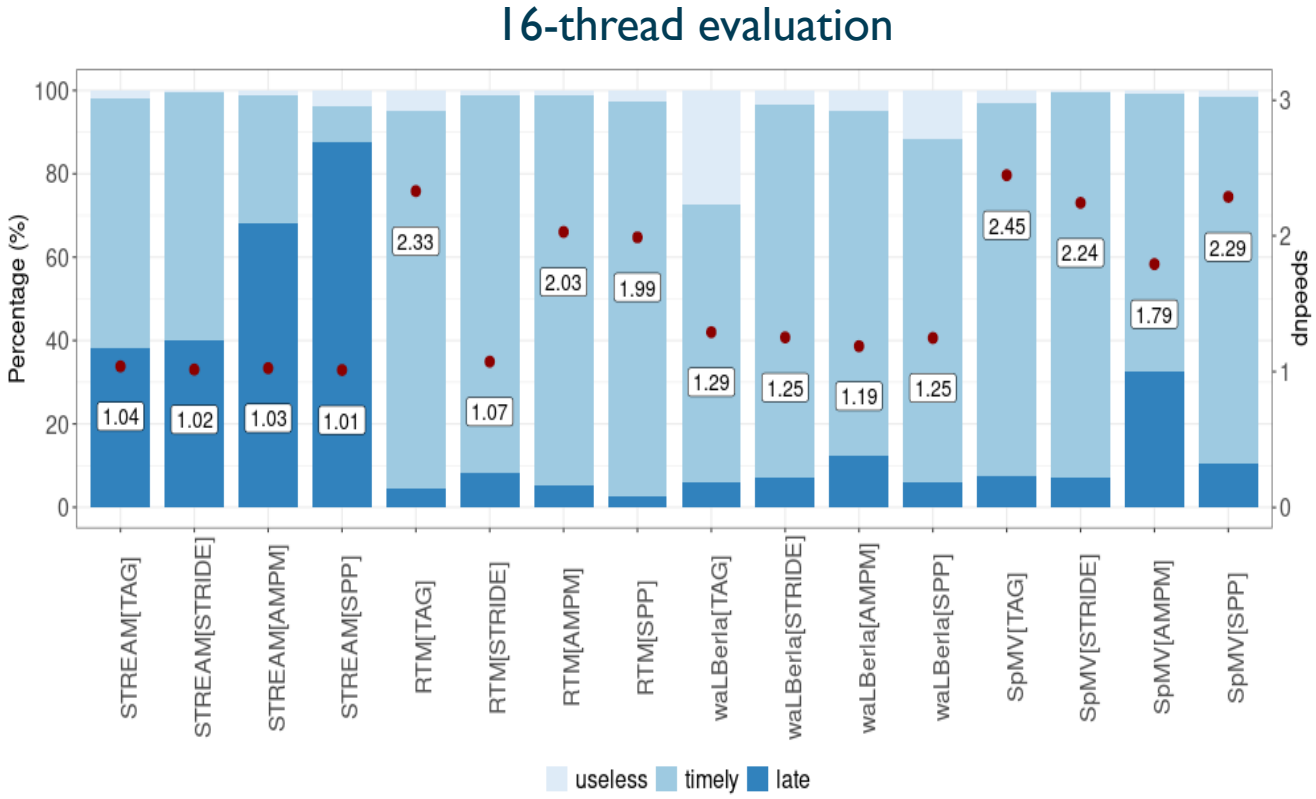



16-threads

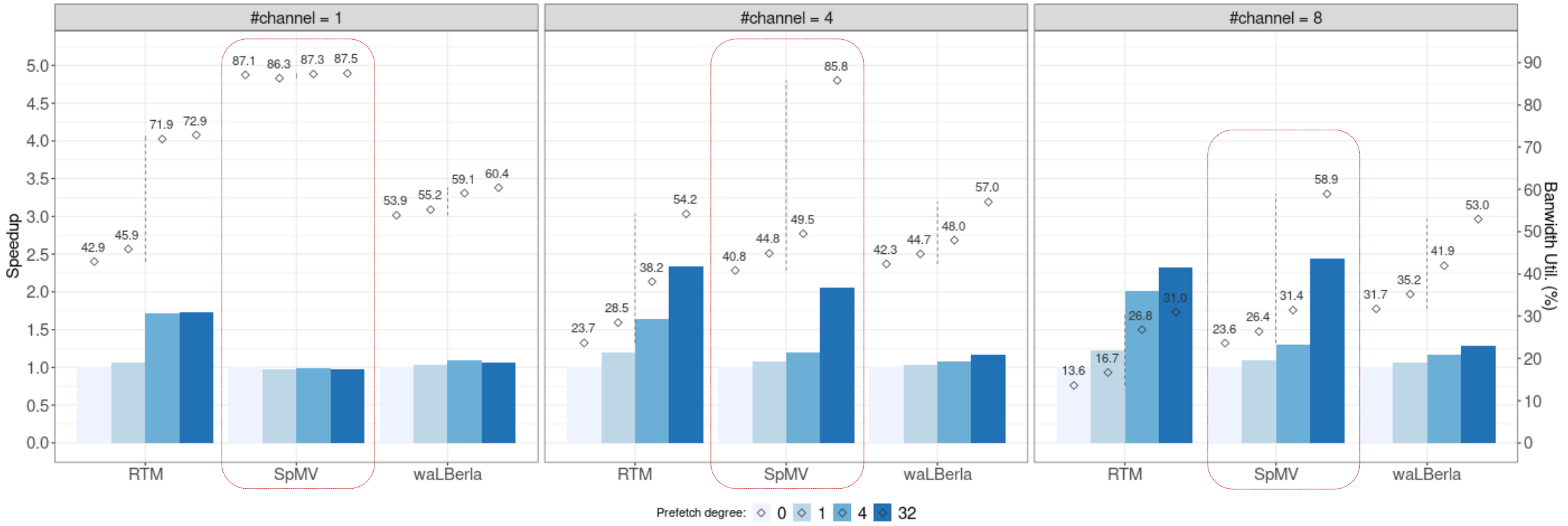
# PERFORMANCE IMPROVEMENT

- All selected prefetchers are effective showing speedups that highly correlate with the classified prefetches
- Among prefetchers, Next-line provides the highest effectiveness, although, this prefetcher shows the highest useless prefetches
- SpMV gains the most up to a speedup of 2.45×



# SENSITIVITY ANALYSIS-EFFECT OF MEMORY BANDWIDTH




Higher speedup gains when increasing #channels (Benefit of aggressive prefetching)

# CONCLUSIONS

- We have evaluated the effects of memory prefetching of scientific application on high-end Arm processors with attached HBM memory using a simulation methodology based on a gem5
- Lessons learnt
  - Memory access patterns from the stencil-style and SpMV codes commonly used in today's scientific applications benefit from spatial address-correlation prefetching schemes
  - Aggressive prefetching evaluated for the kernel codes (e.g. SpMV) is sensitive to memory bandwidth: more available bandwidths lead to higher speedups
- Future studies
  - In-deep analyze memory access patterns (delta, address repetition)
  - Extend the evaluation for other HPC application kernels

- Thanks for your attention!

- Acknowledgements
  - Tiago Muck (ARM Ltd.)
  - Stepan Nassyr, Bine Brank (JSC)

# EPI PARTNERS

# EPI FUNDING

This project has received funding from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 and Specific Grant Agreement No 101036168 EPI-SGA2. The JU receives support from the European Union's Horizon 2020 research and innovation programme and from Croatia, France, Germany, Greece, Italy, Netherlands, Portugal, Spain, Sweden, and Switzerland.