Memory Performance of AMD EPYC Rome and Intel Cascade Lake SP Server Processors

Markus Velten
Robert Schöne
Thomas Ilsche
Daniel Hackenberg
{givenname.lastname}@tu-dresden.de
Technische Universität Dresden, Center for Information Services and High Performance Computing (ZIH)
Dresden, Germany

ABSTRACT

Modern processors, in particular within the server segment, integrate more cores with each generation. This increases their complexity in general, and that of the memory hierarchy in particular. Software executed on such processors can suffer from performance degradation when data is distributed disadvantageously over the available resources. To optimize data placement and access patterns, an in-depth analysis of the processor design and its implications for performance is necessary. This paper describes and experimentally evaluates the memory hierarchy of AMD EPYC Rome and Intel Xeon Cascade Lake SP server processors in detail. Their distinct microarchitectures cause different performance patterns for memory latencies, in particular for remote cache accesses. Our findings illustrate the complex NUMA properties and how data placement and cache coherence states impact access latencies to local and remote locations. This paper also compares theoretical and effective bandwidths for accessing data at the different memory levels and main memory bandwidth saturation at reduced core counts. The presented insight is a foundation for modeling performance of the given microarchitectures, which enables practical performance engineering of complex applications. Moreover, security research on side-channel attacks can also leverage the presented findings.

CCS CONCEPTS

• **Computer systems organization** → **Multicore architectures**; *Single instruction, multiple data*; *Interconnection architectures.*

KEYWORDS

AMD Zen 2; AMD EPYC Rome; Intel Xeon Cascade Lake; Intel Xeon Skylake; cache coherence; memory hierarchy

ACM Reference Format:
Markus Velten, Robert Schöne, Thomas Ilsche, and Daniel Hackenberg. 2022. Memory Performance of AMD EPYC Rome and Intel Cascade Lake SP Server Processors. In *Proceedings of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22), April 9–13, 2022, Bejing, China.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3489525.3511689

ICPE 2022, April 9–13, Beijing, China

ACM ISBN 978-1-4503-9143-6/22/04…$15.00
https://doi.org/10.1145/3489525.3511689

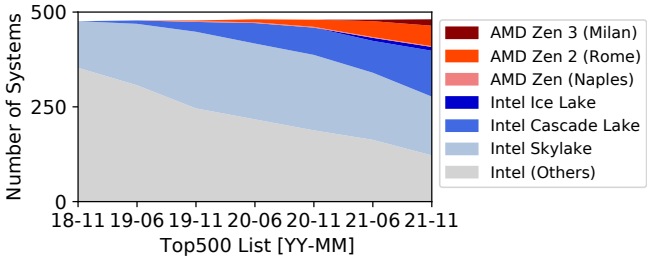


Figure 1: Stacked line chart for Intel and AMD microarchitectures found in Top500 systems [30]

1 INTRODUCTION

x86 processors dominate the HPC market, with 483 systems in the November 2021 Top500 list [30]. While most of these systems (408) are Intel-based, AMD continuously increases its market share in recent years (Figure 1). Processors of these two vendors mostly share the same instruction set architecture (ISA), but feature different internal architectures. The differences have strong implications on the performance of executed codes. Some criteria such as frequencies, SIMD widths, cache sizes, and number of cores are featured in high level specification descriptions. Other features are much less prominently documented. This includes the internal network, which connects cores, DRAM, and I/O, but also the implementation of the cache coherence protocol, which directly influences the memory-bandwidth and latencies under different conditions. This paper reveals data on these aspects for recent processors: AMD EPYC Rome (*Rome*, implementing the Zen 2 microarchitecture) and Intel Cascade Lake SP (*CLX*, implementing the Cascade Lake microarchitecture). This information is crucial for various fields, e.g., security [33] and optimization of parallel software [27].

The remainder of this paper is structured as follows: we introduce related research in Section 2 and discuss the Rome and CLX architectures in Section 3 and Section 4, respectively. Section 5 describes the measurement setup and Section 6 presents evaluations of local memory accesses. This is followed by a performance analysis for accesses to remote cache and memory locations within a socket in Section 7 and to a remote socket in Section 8. We summarize our findings and conclude this paper in Section 9.

2 RELATED WORK

Manuals from the processor vendors provide high level overviews, e.g., AMD's Processor Programming Reference [1] and the Intel Specification Updates [13, 14]. This information is complemented by the respective software optimization guidelines [2, 15] and additional articles, e.g., on Intel Skylake SP (SKX) [24]. Further details are presented by processor designers in peer-reviewed articles, which also describe physical implementations and control loops. Suggs et al. present the Zen 2 architecture in [7, 8]. Naffziger et al. describe how multiple dies form a chiplet in [25, 26]. Tam et al. [31] and Arafa et al. [4] detail the SKX and CLX architecture, respectively.

Researchers investigate finer details and independently validate specific details of x86 processors. The memory subsystem is the main focus of several publications. Molka et al. cover cache coherence and memory performances of older architectures in [21–23]. We continue and extend their work in this paper and compare the more recent architectures Rome and CLX. Alappat et al. investigated the CLX architecture in [6]. We extend the research with a more in-depth analysis of the memory latencies, validate the bandwidth results with different benchmarks and compare latency and bandwidth results with Rome. While cache and memory performance is influenced by power saving mechanisms [10, 28, 29], the focus of this paper is on the performance at constant processor frequencies.

3 THE AMD EPYC ROME ARCHITECTURE

3.1 General Concept

Rome processors use two different types of dies that are combined on one package [8, 26]. Up to eight Core Complex Dies (CCD) are connected to the I/O-die via AMD's Infinity Fabric (IF). The I/O-die (Figure 2) and its Infinity Fabric (IF) grid connect the CCDs among each other and to external components, including the 128 PCIe Gen 4 lanes [1, Figure 21], [25] and main memory. IF-switches are used to route data through the I/O-die, which induce a latency of at least 2 Fabric Clock (FCLK) cycles at the nominal IF frequency of 1467 MHz [25]. Additionally, IF-repeaters are used, which cause a 1 FCLK cycle latency [25].

Each CCD includes two Core Complexes (CCX), which consist of up to four Zen 2 cores each (see Figure 3), resulting in up to 64 cores per processor. While L1 and L2 caches are per core (see Figure 4), all cores within a CCX share a common 16 MiB L3 cache. The Infinity Fabric on Package (IFOP) interface on each CCD connects the two CCX to the I/O-die, but not to each other [1, 26].

3.2 Memory Architecture Details

Burd et al. published in [5] that Zen uses a MDOEFSI (*Modified, Dirty, Owned, Exclusive, Forward, Shared, Invalid*) protocol, without naming details. They also state that Zen's Infinity Fabric is based on an enhanced coherent HyperTransport protocol which has been used by AMD before (compare [20]). It can be assumed that these protocols are used in Zen 2 as well.

The layout of a Rome processors indicates the presence of four non-uniform memory access (NUMA) nodes. BIOS settings can be used to expose NUMA nodes to the operating system (OS), one, two

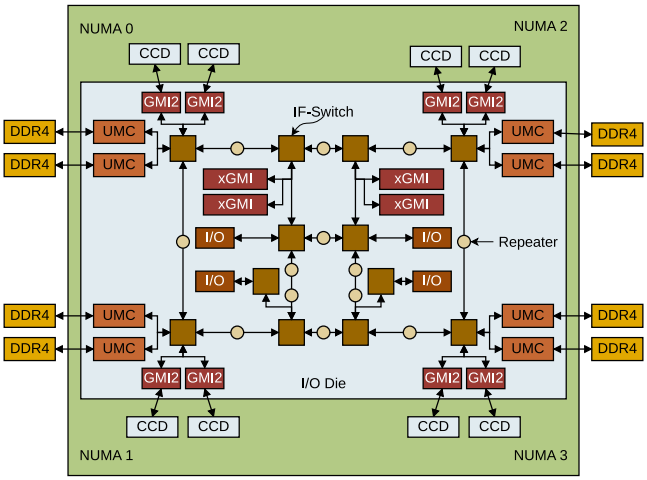


Figure 2: Layout of an AMD Rome processor with its central I/O-die, which connects CCDs, DRAM, and I/O via an Infinity Fabric (IF) network, Global Memory Interfaces (GMI), and Unified Memory Controllers (UMCs). [1, 25]

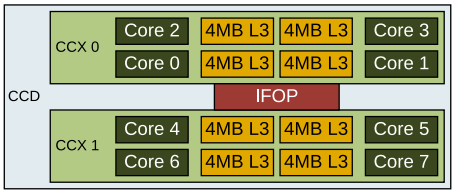


Figure 3: Layout of an AMD Rome Core Complex Die (CCD) hosting two Core Complexes (CCX) and the Infinity Fabric On-Package (IFOP). [8, 26]

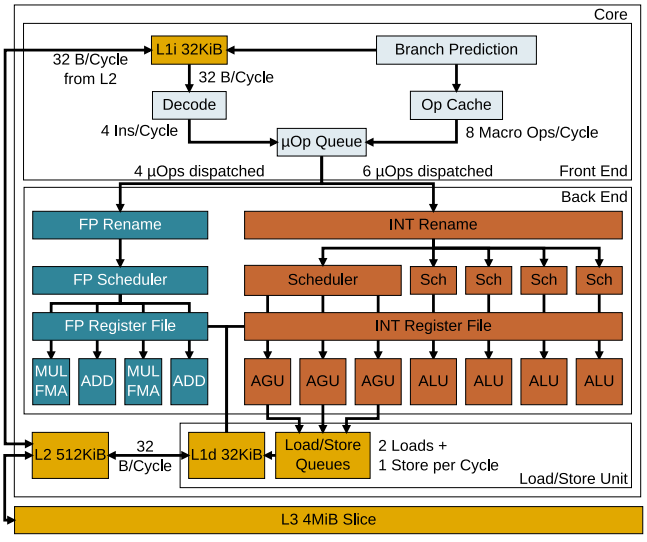


Figure 4: Layout of an AMD Zen 2 core (see [2, 8]).

and four nodes can be configured [19, Section 2.5] where *"[m]emory is interleaved across the [...] memory channels in each NUMA domain"*. Also, *"each server can be configured [...] with an additional option to configure L3 cache as NUMA nodes"*, enabling up to 16 NUMA nodes per processor. Due to the positions of the socket-to-socket Global Memory Interconnect (xGMI) interfaces, remote socket access latencies will depend on the relative position of the communication partners. The architecture supports up to two sockets.

Each of the cores of a CCX hold one slice of the victim L3 cache [2]. The L3 contains evicted cache lines from the L2 caches and valid copies of cache lines shared by multiple cores. Also, data transfers and cache coherency between L2 caches on the CCX are managed by the shadow tags in the L3. The L3 cache slices use a common L3 frequency, which is generally defined by the highest core frequency of all cores within the CCX [29].

AMD Zen 2 cores use three Address Generation Units (AGU) in the integer execution unit to perform up to two 256-bit loads and one 256-bit store per cycle [2, 8]. Compared to first generation Zen processors, these have been widened to match the increased width of SIMD execution. Zen 2 floating point units are able to process 256-bit Advanced Vector Extensions (AVX) instructions in one cycle [2, 8], even though this could limit core frequencies in some cases [29]. Load and store queues attached to the AGUs have 44 and 48 entries, respectively, and load data from a 32 KiB L1d cache [2, 8]. Instructions are loaded at 32 B/cycle from a 32 KiB L1i cache and buffered in a 4096 *entry* op cache [8]. The 512 KiB L2 cache is inclusive of both L1 caches [2, 8]. AMD uses hardware prefetchers for the L1d, L1i, and the L2 cache to avoid stalls due to cache misses [2, Section 2.1]. L1 and L2 prefetchers can be deactivated via the BIOS, or, according to our findings, by disabling bit 0 of MSR `0xc001102b` and enabling bit 16 of MSR `0xc0011022` for each core.

4 THE INTEL CASCADE LAKE SP ARCHITECTURE

4.1 General Concept

The CLX processor architecture succeeds the SKX architecture. Both are using a 14 nm process and a monolithic design [4] and are available with up to 28 cores per processor. As described in [4] and [6], changes between these two generations are minimal, including higher core frequencies, support for Optane DC, faster DRAM, and additional instructions. This is confirmed by Alappat et al., who note in [6, Section 2] that SKX and CLX behave identically in memory and floating point benchmarks.

A grid of horizontal and vertical fabric lanes connects all parts of the processor [4, 28, 31] (see Figure 5). Transfers within the 2D-mesh are always routed along the vertical axis, followed by the horizontal direction [31]. All uncore components operate on a common frequency which is managed by a hardware control loop from a range of frequencies [28, 31]. The number of PCIe ports, and thus lanes, depends on the number of cores, since entire columns or single cores of the grid may be deactivated/removed [11, 17]. This is a significant difference to AMD EPYC processors, where I/O interfaces are identical for all core counts.

4.2 Memory Architecture Details

Intel uses a unified execution unit design for CLX, unlike AMD (Section 3.2) [9, 15]. Four AGUs are available: two for 512-bit loads, one for 512-bit stores and one for address generation only [9, Table 11.1]. Port 6 is reserved for integer/logic-only instructions. The read buffers have 72 entries, 56 entries are available in the write buffers [9, Section 11.9]. Both, integer and SIMD floating point instructions, can be executed by ports 0, 1 and 5. Ports 0 and 1 can be fused for the execution of AVX-512 vector instructions. On some processors, port 5 has as a second AVX-512 unit, Intel lists the number of available AVX-512 units in [15, Page 10]. When executing AVX(-512) instructions, cores use dedicated frequencies in order to prevent thermal damages to the processor. These depend on the type of instruction and the number of cores that are running that instruction [14]. Schöne et al. describe side effects for transitions between normal and AVX-512 frequencies for SKX in [28].

CLX processors have a L1i and L1d cache of 32 KiB (Figure 6), equal to Zen 2. Both caches load data at 32 B/cycle. A 1536 entry op cache buffers instructions [9, Table 11.12]. The L2 cache is inclusive of the L1 caches and at 1 MiB twice as large as the Zen 2 L2 [31]. The L1 and L2 caches utilize hardware prefetchers to reduce the impact of cache misses [15, Section E.3].

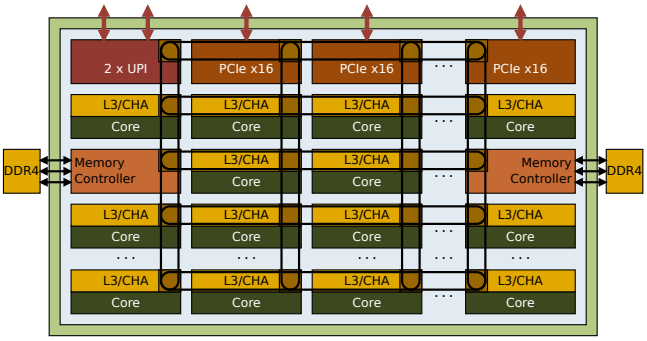


Figure 5: Layout of an Intel Cascade Lake SP processor, where a 2D-mesh connects core tiles, memory controllers and I/O [17, 28].

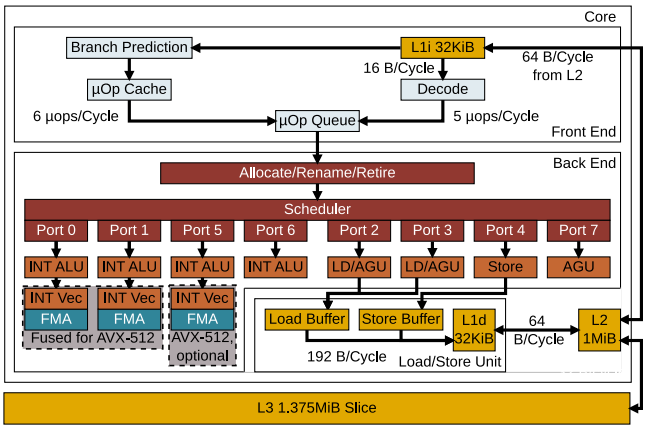


Figure 6: Layout of an Intel Cascade Lake SP core

As indicated in Figure 5, a core-tile holds not only a core, but also a slice of 1.375 MiB L3 cache and the Caching and Home Agent (CHA) [31]. The CHA maps accessed addresses outside its own caches and provides the necessary routing information through the mesh [24]. CLX has a non-inclusive L3 cache, which *"may appear as a victim cache"*, *"depending on the access pattern, size of the code and [accessed] data, and sharing behavior between cores"* [15, Section 2.2.1.2] [3]. Generations prior to SKX used smaller L2 and larger per-core L3 caches, the latter being inclusive. According to [24, Section "Cache Hierarchy Changes"], the new cache layout was chosen to reduce overall latencies, as more requests may be served from the larger L2. The non-inclusive L3 leads to a higher effective total cache size, since it does not necessarily hold data that is present in other cache levels. Although the per-core L3 cache size is higher for Zen 2 (4 MiB over 1.375 MiB), Intel's 2D mesh interconnect allows all cores to access all L3 slices with a worst-case overhead of 18 uncore cycles[1] [24]. However, Intel implements a suboptimal slice hash mechanism, which can lead to an imbalance in using the available L3-slices [16].

SKX and CLX processors have two memory controllers with three memory channels each [28]. They can be split into two NUMA nodes, referred to as Sub-NUMA Clusters (SNC), by assigning each half of the cores to one of the two memory controllers as well keeping the L3 cache slices local to the core's respective SNC [24].

SKX processors implement the MESIF coherency protocol with the states *Modified*, *Exclusive*, *Shared*, *Invalid*, *Forwarded* [12, Section 2.2.4]. The directory protocol names two additional states: *Any* and *L*[2] [12, Table 2-179], which are not documented and for this reason not investigated in this paper. We assume that CLX uses the same protocol.

Table 1: Overview on hardware and software of used test systems

Processor	2 × **AMD EPYC 7702**	2 × **Intel Xeon Gold 6248**
Cores	2 × 64	2 × 20
Avail. freq.s	1.2, 1.5, **2.0 GHz**, Turbo	1.2–2.5 GHz, Turbo
L1 cache	128 × (32 KiB + 32 KiB)	40 × (32 KiB + 32 KiB)
L2 cache	128 × 512 KiB = 64 MiB	40 × 1024 KiB = 40 MiB
L3 cache	32 × 16 MiB = 512 MiB	2 × 24.75 MiB = 49.5 MiB
Server Model	GIGABYTE	HPE
/ Mainboard	MZ62-HD0-00	ProLiant DL360 Gen10
Memory	Micron	HPE P03052-091
	18ASF4G72PDZ-3G2B2	
	16 × 32 GiB = 512 GiB	12 × 32 GiB = 384 GiB
	1600 MHz	1467 MHz
OS	CentOS 7.7.1908 (Core)	Ubuntu 18.04
Kernel	3.10.0 x86_64	4.15 x86_64
NUMA-Setup	2 × 4 = 8 (NPS4)	2 × 2 = 4 (SNC)
	— BenchIT —	
Compiler	GCC 10.2.0	GCC 7.5
	— STREAM —	
Compiler	Intel 19.0.1	Intel 19.0.5
Comp. flags	`-DSTATIC -DSTREAM_ARRAY_SIZE=800000000 -qopenmp -Ofast -mcmodel=large -shared-intel -qopt-streaming-stores always -march=cascadelake -xCORE-AVX512 -qopt-zmm-usage=high`	

5 TEST SYSTEM & BENCHMARKS

We performed our measurements on two dual socket servers with AMD EPYC 7702 and Intel Xeon Gold 6248 processors, respectively. Table 1 lists relevant details and configurations. Unless stated otherwise, we use nominal core frequencies. The uncore frequency of the Intel system was set to its nominal frequency of 2400 MHz with the `likwid-setFrequencies`[3] tool [32]. While the uncore clock can be lowered if the Thermal Design Power (TDP) is reached [28], we designed our experiments in a way that such a behavior is not triggered.

We use *BenchIT*[4] [18], with the *x86-membench* extension [20] for latency and bandwidth analyses. These benchmarks can be configured for a wide range of objectives by changing the configuration of the respective `PARAMETERS` file. To analyze the impact of the cache coherence protocol, we use the benchmark *memory_latency*.

As described in [20, Section 3.5.1], the *memory_latency* benchmark uses pointer-chasing to determine latencies. The size of the allocated buffer determines the memory level(s) to be analyzed. The accessed addresses are created by an additional thread using a random number generator to minimize prefetcher-effects. A coherence state control routine ensures that accessed cache lines have the requested coherence state on the requested core by running an additional thread on this core before the actual measurement. In [20, Section 3.3], Molka describes how the different coherence states are created. After measuring the start time (`rdtsc` serialized with `mfence` and `lfence` calls), the measurement thread accesses the memory addresses (loop with unrolled `mov (%rbx), %rbx`), and measuring the end time (serialized `rdtsc`). The duration is then adjusted by the measurement overhead, which is determined with the same routine without the memory accesses. Algorithm 1 provides a high-level overview of the steps to measure the memory accesses.

Listing 1: *memory_latency* overview. Thread M is only required for coherence-states Shared, Forward, and Owned. [20, Section 3]

```
1 pin threads 0,N(,M) via sched_setaffinity()
2 thread 0,N(,M): allocate memory via numa_set_membind()
3 thread 0: warm-up of TLB by touching memory
4 thread 0: signal thread 1 to prepare data
5 thread N(,M): touch data correctly [20, Sec. 3.3]
6 thread 0: wait for thread N(,M)
7 thread 0: access memory of thread N, measure latency
```

In order to achieve reproducible results, we flushed all cache levels before each run with the `BENCHIT_KERNEL_FLUSH_{L1|L2|L3}=1` flag in the PARAMETERS file of the benchmark. We use the default 512 B alignment to avoid the re-use of cache lines, set with `BENCHIT_KERNEL_ALIGNMENT=512`, unless otherwise noted.

[1] 9-hop distance for a 28 core configuration
[2] as defined in description, or *P* as defined in unit mask name
[3] https://github.com/RRZE-HPC/likwid/wiki/likwid-setFrequencies
[4] https://tu-dresden.de/zih/forschung/projekte/benchit/

Cache and main memory bandwidth values are determined with the *throughput* benchmark where we use different Streaming SIMD Extensions (SSE) and AVX instructions. To ensure that our bandwidth results are truly limited by the achievable bandwidth, we configure the benchmark with the `BENCHIT_KERNEL_BURST_LENGTH` parameter to use eight 64 bit xmm, 16 256 bit ymm or 32 512 bit zmm vector registers for SSE, AVX, and AVX-512 kernels, respectively. In order to use the 32 zmm vector registers, additional compiler flags (`-mavx512f` in our case) have to be used. We extended the benchmark with a range of AVX-512 kernels. Transparent huge pages are used for both benchmarks to clearly distinguish memory levels, as advised in [20]. Hardware prefetchers are enabled for all measurements on both systems, except for the latency benchmark on CLX. We flush all cache levels before each run in order to achieve reproducible results and to avoid the impact of prefetchers. We set `/sys/kernel/mm/ transparent_hugepage/enabled` to `always` to clearly distinguish memory levels.

We configure BenchIT to use four dataset sizes per memory level. For each of these values, BenchIT reports the minimum for latencies or maximum for bandwidths of three internal measurements to filter out outliers from external influences. We repeat each experiment ten times resulting in a total of 120 values (10×4×3) per reported data point, combined using minimum and maximum. As latencies for remote L1 accesses were noisy, we report the medians for these measurements, which coincide with the modes of the samples.

Additional measurements are performed with *STREAM*, as it provides more realistic memory access patterns with loads and stores. We use non-temporal stores in order to achieve a higher bandwidth closer to theoretical limits. *STREAM* repeats measurements ten times internally and reports the highest bandwidth. In addition, we run it ten times and use the maximum of the reported values.

6 LOCAL MEMORY ACCESS

6.1 Latencies

In the simplest case, processor cores access data in their own memory hierarchy without influences from additional cores and NUMA domains. We compare the local latencies of the two architectures, using the BenchIT *memory_latency* benchmark, in Figure 7. To focus on the efficiency of the architecture rather than the applied core frequency, we mainly present results in cycles and not in ns. The L1d caches can be accessed with 4 cycles on both architectures (2 and 1.6 ns for the Rome and CLX architecture, respectively) . L2 cache read latencies for Rome and CLX are 12 cycles (6 ns) and 14 cycles (5.6 ns), respectively. However, the Intel CLX platform provides twice as much L2 cache per core. This reduces the probability for L2 cache misses and even less efficient L3 cache accesses. For the locally accessible L3 cache[5], Intel provides more capacity available for a single core, but at higher access latencies (54 cycles (21.6 ns) vs 39 cycles (19.5 ns)). While single-threaded applications will benefit from larger local L2 and L3 caches on Intel platforms, the *total* size of L2 and L3 caches is higher for AMD processors, which is beneficial for parallel applications. Moreover, for equally clocked processors, the AMD system provides lower latencies to caches, which further reduces memory stall cycles.

[5]We set `BENCHIT_KERNEL_ALIGNMENT=64` for L3 latency tests on the Rome system to use all L3 slices equally.

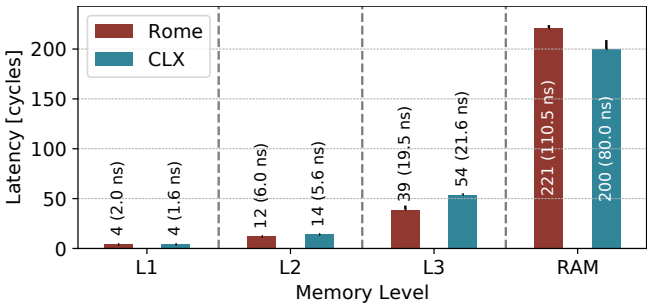


Figure 7: Local memory access latencies for AMD EPYC 7702 (Rome) and Intel Xeon Gold 6248 (CLX), error bars indicate maxima.

Main memory latencies cannot be easily compared between processors, since the specifications of the installed memory DIMMs also contribute to their access time. Moreover, adapting I/O p-states [29] and uncore frequencies [28] lead to more uncertainty. We measure a latency of about 220 cycles (110 ns) for the Rome system, and 200 cycles (80 ns) for CLX. The higher RAM latencies for Rome may be attributed to the data path routing via the I/O-die, whereas the memory controllers of the CLX processor are integrated into the fabric mesh.

6.2 Bandwidth

HPC applications often access main memory with a predictable pattern, which allows the compiler (via instruction re-ordering), the out-of-order engine, and hardware prefetchers to hide occurring latencies and thereby improve the effective memory bandwidth. However, bandwidth is also limited by the width of datapaths and concurrent access to shared resources, i.e., L3 cache and DRAM.

We use the BenchIT *throughput* kernel for our analysis, which only performs loads. This kernel relies on a streaming access to memory using SIMD extensions to benefit from wider load instructions. We extended the kernel so that 512-bit wide accesses can be measured. Intel SKX and CLX processors have dedicated AVX(-512) frequency ranges with dedicated nominal frequencies, which directly influence these measurements. To avoid an unwanted and unpredictable change of frequencies, we pinned core frequencies to 1.6 GHz – the nominal core frequency for AVX-512 for the Intel Xeon Gold 6248 processor [14, Figure 3]. Since AMD did not openly publish dedicated AVX frequencies, we use the nominal frequency (2.0 GHz) for the Rome system, even though highly demanding workloads could lead to throttling [29, Section V.E].

Figure 8 shows bandwidth results for memory reads on the AMD Rome system. The AMD cores reach the theoretical L1 bandwidth of 128 GB/s (64 B/cycle) if two 256-bit floating point pipes are utilized. The bandwidth for the L2 cache falls slightly short of their theoretical maximum, with measured 63.7 GB/s (31.4 B/cycle) instead of 64 GB/s. A single core can read from the L3 with up to 46 GB/s (23 B/cycle) when using AVX. According to our measurements, bandwidth for the L3 does not scale linearly for an entire CCX. With four cores the achieved bandwidth is only 151 GB/s (18.9 B/cycle/core). Also, RAM bandwidth for one CCD is saturated by using 3 cores of a single CCX in these measurements. While

adding accesses from the second CCX on one CCD (cores 0-7) does not help in gaining more performance, using the second CCD of a NUMA node (cores 0-15) increases the measured RAM bandwidth slightly. The results for SSE (add_pd instruction) and AVX do not differ significantly, except for L1.

Since an increase of used cores beyond three per CCX does not increase bandwidth, we evaluate further bandwidth saturation points: Figure 9 shows bandwidth measurements on a single NUMA node with two CCDs hosting two CCX each. We use *STREAM* for this analysis to achieve higher bandwidths for low core counts (due to non-temporal stores). We scaled the number of cores per CCX and thus per CCD of a single NUMA node. The highest bandwidth of a single NUMA node (42.9 GB/s) was measured in configurations with two participating CCDs and two cores per CCD. Since each NUMA node has its own set of memory channels, this bandwidth scales to ~171 GB/s for the entire socket, as dedicated measurements show (not depicted). Even with only one core per CCD, the measured bandwidth is only slightly lower at 42.3 GB/s. These numbers indicate that a small number of cores per CCX should be used for bandwidth-limited workloads. Our measurements imply that users with a demand for a high bandwidth may benefit from processors with two CCDs per NUMA node, even if the higher core counts might not be necessary with respect to compute performance.

Figure 10 shows bandwidth results for reading memory accesses on the Intel Cascade Lake system. The results do not reach the theoretical bandwidth of the L1d caches of 128 B/cycle when two 512-bit floating point units are used. Instead, we measure 116.25 B/cycle for AVX-512 instructions. This is lower than the sustained bandwidths listed in [15, Table 2-6], but in line with measurements in [6, Figure 3]. Notably, the L2 cache bandwidth varies for different instructions despite being consistently lower than the respective L1 bandwidth. The L3 bandwidth results are similar for all SIMD widths, achieving 11.3 B/cycle. Again, this value is lower than the sustained bandwidth of 15 B/cycle described in [15, Table 2-6]. The RAM bandwidth is barely affected by SIMD width. On a SNC, the RAM bandwidth can be saturated with eight cores.

The prerequisites for our measurements on the Rome and CLX system are significantly different. This needs to be taken into account when comparing the observed cache bandwidth of the two architectures. Running the AMD processor at its base clock speed may be a best case scenario, whereas the AVX-512 frequency we chose for CLX is more of a worst case, in particular for SSE and AVX.

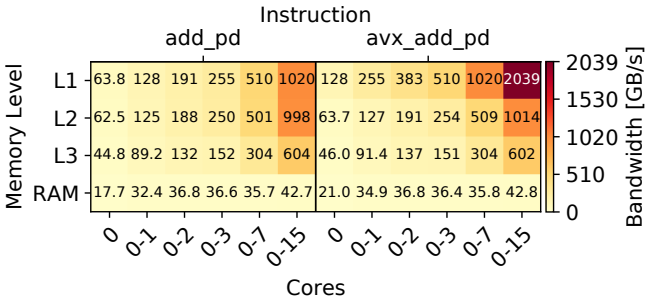


Figure 8: AMD EPYC 7702 – bandwidth at reference frequency (2 GHz) for one NUMA node.

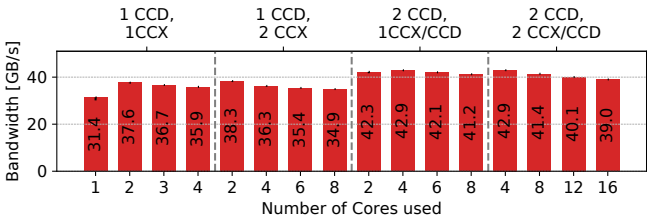


Figure 9: AMD EPYC 7702 – *STREAM* Triad: bandwidth measured on a single NUMA node with two CCDs and two CCX per CCD, error bars indicate minima.

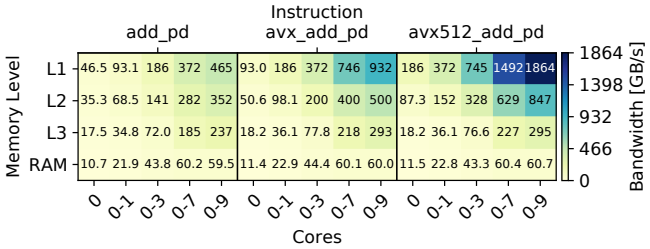


Figure 10: Intel Xeon Gold 6248 – bandwidth at nominal AVX-512 frequency (1.6 GHz) for one SNC

In real world workloads, both processors might be able to use turbo frequencies. As demonstrated in [29], the Rome processor might throttle to a lower, not publicly disclosed frequency. Ultimately, the used frequencies, and thus achieved bandwidths, depend on the workload and its distribution across cores and the thermal conditions the systems operate in. Nevertheless, a few conclusions may be drawn. Notably, the measured values for the CLX processor are lower than the maximum bandwidth and in many cases even the sustained bandwidth that are defined in [15, Table 2-6, Section 2.2.1.3]. Moreover, the L2 bandwidths depend on the used instruction. In an ideal situation for this processor, the SSE and AVX bandwidths would be close to the maximum throughput of the execution units for both the L1 and L2 caches, as the bandwidth requirement for these instructions is well within the maximum cache bandwidths.

This ideal case can be observed for Rome processors. Here, the L2 bandwidth for SSE instructions is very close to the L1 bandwidth as well as to the L2 bandwidth for AVX instructions and the theoretical maximum.

The main memory bandwidth of a single NUMA node of a CLX processor is higher than on Rome, as CLX has three memory channels per NUMA node, Rome two. However, the bandwidth of a NUMA node can be saturated with fewer cores on Rome than on CLX. Additionally, the overall bandwidth for an entire socket for Rome is higher than for CLX, which is caused by Rome's eight memory channels compared to CLX' six.

7 INTRA-SOCKET MEMORY LATENCIES

While local memory accesses as described in the previous section are most common, thread migration, wrong data placements, and communication via shared memory lead to situations where a core

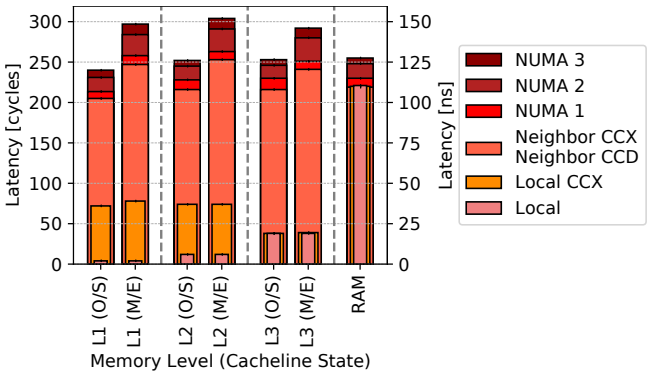


Figure 11: Memory read latencies for an AMD EPYC 7702 on one socket, measured from NUMA node 0 – NUMA characteristics can be clearly observed. *Owned* and *Shared* as well as *Modified* and *Exclusive* states result in identical latencies. Refer Table 2 for precise values.

accesses memory, which is not placed in its local memory hierarchy. To measure latencies for such accesses, we again use the BenchIT *memory_latency* benchmark. Threads are placed using the environment variables provided by x86-membench. We distinguish accesses to cache lines placed in the coherency states Modified, Owned (on Rome) and Exclusive, Shared, Invalid, Forward (on CLX). We omit invalid cache lines from further analyses. Since they are fetched from RAM at all times, the latencies are equal to those reported in Section 6.

7.1 AMD EPYC Rome

For Rome, we present Owned and Shared as well as Modified and Exclusive cache lines in combination as the observed performance pattern for each of these pairs is equal. We performed measurements at nominal frequencies of 2 GHz.

Figure 11 summarizes the measurements for the Rome system: When a core loads data from a different L1 cache within the same CCX, *Owned* and *Shared* cache lines can be served from the L2 cache within 72–74 cycles (36–37ns). This reflects the nature of L2 being inclusive of the L1 cache. Modified and Exclusive data carry a slight penalty when accessed on a different L1 cache in the same CCX increasing latency to 78 cycles (39 ns). Data stored in the shared L3 cache of a CCX can be read within 39 cycles (19.5 ns), which is about half the latency when accessing data in other L1 or L2 of other cores of a CXX. According to [2], the load-to-use latency of the L3 cache is 39 cycles.

Accesses to another CCX within the same NUMA node traverse through the I/O-die, since CCXs within a CCD are not directly interconnected. At least a 200 cycles (100 ns) latency is incurred when performing accesses to non-local L3 caches.

As shown in Section 6, reading from local RAM takes 220 cycles (110 ns). When reading memory from another NUMA node, the properties of the I/O-die can be observed. Accessing the RAM of the closest NUMA node requires additional ~10 cycles (5 ns). This translates to two IF-switch and IF-repeater hops. Assuming the IF operated at its nominal frequency of 1467 MHz, this translates to ~3–4 FCLK cycles per switch and repeater combination. A ~28 cycle (14 ns) penalty is incurred for accesses to the second NUMA node, and 35 cycles (17.5 ns) to the third NUMA node. This implies that each IF-switch (Figure 2) in a path to another NUMA node incurs a ~2–2.5 ns latency per direction in our workload.

7.2 AMD EPYC Rome - Intra-CCX

Figure 12 shows latencies for all possible communication patterns within a CCX when *Modified* cache lines are accessed. L3 latencies seem uniform. Accesses to other cores show a clear correlation between latencies and the relative position of the communicating cores within the CCX. It is possible that these latencies are an effect of the 512 B alignment, which we used for the L1 and L2 cache measurements. With this alignment, the slice hash mechanism, which selects the L3 slice for a given cache line based on its address, will only use the L3-slice of core 0. This gives us insight into the internal layout of a CCX: If only the L3 slice of core 0 is accessed, the latency penalties for accesses will be caused by the distance between the used cores. This can be observed when core 0 requests memory from other cores. Cache lines held by cores 1 or 2 can be accessed within the same time. Requests to core 3 take additional time, i.e., ~5–7 cycles for the L1 and L2 caches. A possible explanation would be that a ring interconnect is used between the cores/caches, where core 3 has a greater distance to core 0 than core 1 and 2.

7.3 AMD EPYC Rome - Complex Request Flow

Figure 13 shows more complex cache request flows when accessing *Modified* cache lines. For example, a program is started on a core attached to NUMA node 2 and allocates its data there (*home node*). Later, this data is used and changed by a core at NUMA node 3 (*forwarding node*) and holds the most recent copy of the memory in its cache. Finally, the data is accessed by a core located at NUMA node 0 (*requesting node*). The data request would go from the requesting core to the memory controller of the home node and being forwarded to the core on the forwarding node, which replies to the request. Please note that while three cores are involved in this scenario, the access pattern can be realized by a single thread, which is migrated by the operating system. A core on another CCX is chosen for cases in which NUMA node 0 is also the forwarding node, as cache request would be fulfilled within the CCX otherwise.

The highest latency we measured in this scenario, 330 cycles (165 ns), is higher than any latency for a native access in which

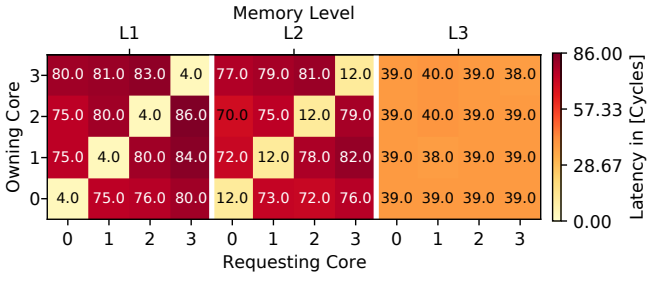


Figure 12: AMD EPYC 7702 – CCX cache read latencies (*Modified*): latencies for all communication patterns within a CCX.

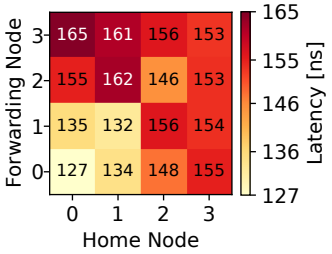


Figure 13: AMD EPYC 7702 – L2 read latencies for complex cache accesses for *Modified* cache lines: NUMA node 0 requests memory which is allocated at the Home Node, but present in a *Modified* state in the L2 cache of a core on the Forwarding Node. All potential scenarios for such accesses with node 0 as requesting node are evaluated. The diagonal from (0,0) to (3,3) indicates standard L2 latencies for direct access from node 0 with forwarding node being the home node.

home node and forwarding node are identical. If the home as well as the forwarding node are different from each other and from the requesting node 0, the resulting latency is determined by the home node. For instance, if NUMA node 1 is the home node, a latency of 322–324 cycles (161–162 ns) is measured if either NUMA node 2 or 3 are forwarding nodes. The same is true for other cases. If, however, NUMA node 0 is both the requesting as well as the home node, the latency is determined by the choice of forwarding node. Similarly, if NUMA node 0 is requesting and forwarding, the home node determines the latency.

Interestingly, the latencies are not symmetrical along the identity axis. Instead, if NUMA node 0 is requesting and home node, the penalty of using a different forwarding node is higher than in cases where NUMA node 0 is requesting and forwarding node. This is notable since in the first case the memory is already present at NUMA node 0 and must not necessarily be transferred to this node. In cases where the forwarding and home node are not identical and not at NUMA node 0, the missing symmetry can be observed as well. If NUMA node 1 is home node and NUMA nodes 2 or 3 are forwarding nodes, the latencies are almost identical. However, if the roles of forwarding and home node are reversed, the latencies decrease by 12–14 cycles (6–7 ns) when switching from NUMA node 2 to 3 as home node.

These results highlight the importance of a careful consideration of the NUMA properties of Rome processors. If the processor is set up to report only as a single NUMA domain to the OS, thread pinning should be used to avoid these latency penalties.

7.4 Intel Xeon Cascade Lake SP

We performed latency measurements on CLX at its nominal core frequency of 2.5 GHz and 2.4 GHz uncore frequency. Latencies for cache lines in state Forward and Shared are reported as one value as they are identical on CLX. As described by McCalpin in [17], processors with a reduced core count can have different deactivated core tiles. We used the `*_RING_BL_IN_USE` performance counters [12, Section 2.2.3] to find the mapping of CPUs (as reported by the OS) to core tiles in our system, which we present in Figure 14. While analyzing the layout, we observed that the third UPI link is not present and/or not used on our system. This behavior was also observed by McCalpin.

Figure 15 shows that latencies of the CLX system depend on the relative positions of the cores in the mesh, as seen in the case of latencies from core 0 to its closest and distant neighbor within the SNC. The differences between the lowest and highest L1 and L2 latencies within an SNC can be roughly assumed to be 2 ns or 5 cycles, highlighting the effectiveness of Intel's mesh design. Increased latencies can also be observed for accesses to memory on the second SNC. While L2 and L3 cache latencies for the states *Modified* and *Exclusive* are identical and have therefore been consolidated, L1 cache latencies from *Modified* cache lines are higher than those for *Exclusive* cache lines: Reading *Exclusive* cache lines from L1 seems to be identical to L2 accesses, indicating that they can be fetched from the inclusive L2 cache. *Shared* and *Forwarded* cache lines are fetched from the L3 cache, if they are not present in local caches, as described in [3].

When accessing the second SNC on the same socket, an additional latency of ~15 cycles (~6 ns) is incurred for all cache levels. In contrast, accesses to other CCX on Rome add an additional 130–200 cycles. These results indicate that CLX processors may be better suited for shared memory workloads where more than four cores work on the same data. RAM latencies for local and remote SNC are much lower than those on Rome, as it was to be expected due to the less complex design of CLX processors.

Our results highlight the benefit of having the memory controllers integrated into the fabric mesh as it is the case with the Intel processor. Whilst L3 latencies on CLX are ~15 cycles higher than on Rome, all CLX cores can access the L3 at similar latencies. This is a potential benefit compared to Rome, where only four cores within a CCX share a L3 cache, in particular for workloads that rely on many cores that share cache lines. If, however, a large per-core L3 is required, Rome processors may be better suited due to their larger per-core L3 slices and lower local L3 latencies.

8 INTER-SOCKET MAIN MEMORY LATENCIES

While previous measurements only showed memory characteristics for accesses within a single socket, adding another processor will further increase complexity. Again, we use BenchIT's latency benchmark to measure main memory latencies at nominal frequencies. We schedule threads to the first core within each NUMA domain and analyze RAM latencies for all inter-socket node-pairings.

We present results for the Rome system in Figure 16(a). Here, a clear split along one axis of the processor can be observed. The latency patterns indicate that the xGMI interfaces are not cross-connected but connected as shown in Figure 16(b). Threads scheduled on NUMA nodes that are closer to the xGMI interfaces communicate with lower latencies with the remote socket. Communication between nodes 0 and 6 or 2 and 4, respectively, takes the least time, 406–408 cycles (203–204 ns). Apparently, in those cases only one IF-switch hop is required per socket in addition to the mandatory hop when accessing the I/O-die. The highest latencies occur for "diagonal" accesses, e.g., from node 1 on socket 0 to node 5 on socket

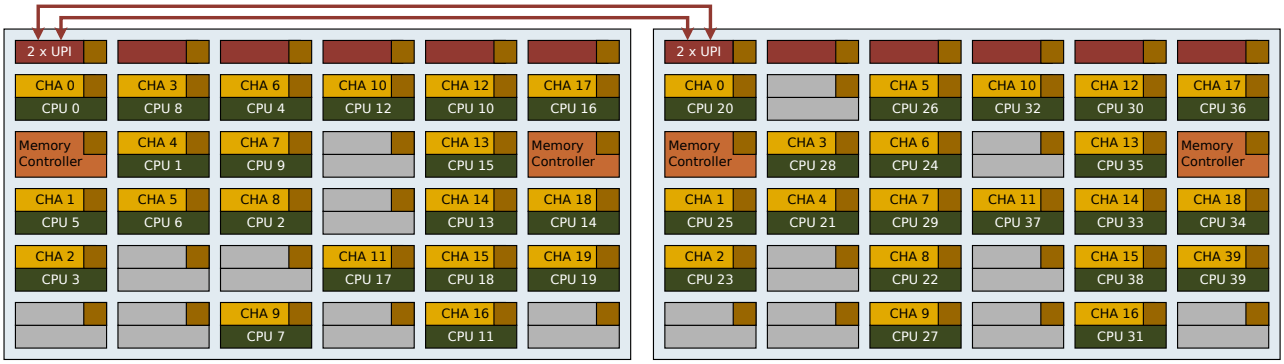


Figure 14: Mapping of CPUs as listed by the OS to core tiles on CLX test system, deactivated core tiles are depicted in gray.

1. For those cases, ~436 cycles (218 ns) are required, 30 cycles (15 ns) more than the fastest case.

Main memory accesses to the second socket on the CLX system traverse through the UPI interfaces. In our case, only the interface tile with two UPI links is being used (see Figure 14), as performance counter events proved[6]. As we show in Figure 17, the lowest latencies can be measured when core 0 requests memory from the first SNC on the second socket (SNC 2), independently of the core that allocated memory in this NUMA node. At 138 ns, this value is ~60 ns higher than a local main memory access. Accessing SNC 3 takes an additional 10 ns. Based on the location of the accessing core on the first socket, additional latencies can be observed.

Our measurements show that transferring data to/from a remote socket comes at a high additional latency penalty. If communication between to sockets is necessary, it is beneficial to schedule communicating threads carefully in order to avoid additional overhead.

[6]We measured {R|T}xL_FLITS_ALL_DATA events for all UPI ports for various inter-socket accesses.

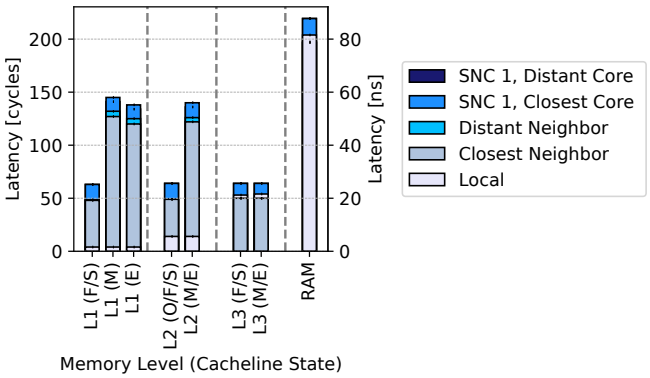


Figure 15: Memory read latencies for an Intel Xeon Gold 6248 on one socket, measured from SNC 0 – *Forwarded* and *Shared* states result in identical latencies, *Modified* and *Exclusive* latencies only for L2 and L3. Refer Table 3 for precise values.

9 SUMMARY, CONCLUSION, AND FUTURE WORK

In this paper, we provided in-depth descriptions of the architectures of current AMD Rome and Intel CLX server processors. The modular chiplet design of the Rome processor allows AMD to assemble up to 64 cores in one package and ensures that all SKUs can profit from the same number of I/O interfaces and memory controllers. This design results in accentuated NUMA properties and an L3 cache that is only shared among four cores at a time. In contrast, Intel uses a monolithic design without inherent NUMA properties. It scales up to 28 cores and shares a common L3 cache among all cores.

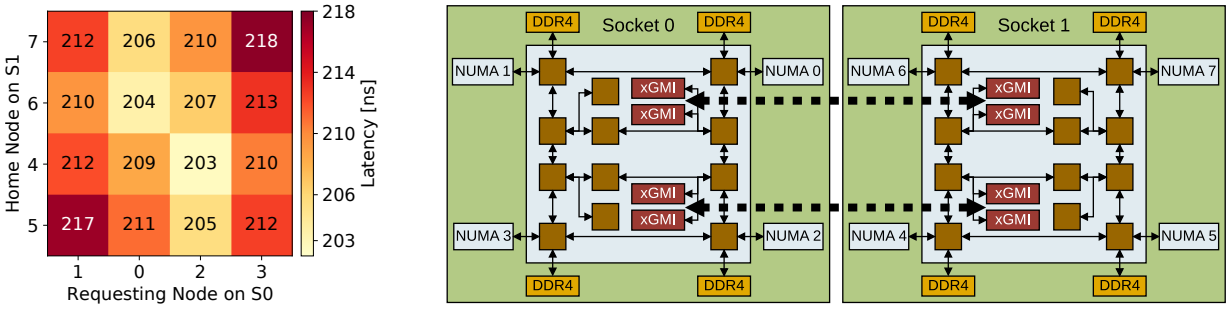
We updated the existing x86-membench code of the BenchIT suite and improved support for the cache design of these processors. The cache flush routines are now able to reflect the usage of inclusive L2 and exclusive L3 caches by both AMD and Intel in their current designs. The throughput benchmark can now use AVX-512 operations with up to 32 512-bit zmm vector registers.

We also evaluated the cache and main memory latencies of the two processors. Our measurements revealed similar latencies for local cache accesses to the L1 and L2 for both processors. The L3 access latencies to the CCX-local L3 slices take fewer cycles on AMD processors. However, as only four cores share the CCX-local L3, there is no commonly shared cache with low latencies for all cores on AMD Rome processors. Therefore, shared memory workloads which rely heavily on data sharing among cores may profit from Intel's monolithic design, as all cores share a common L3 with similar latencies.

We demonstrated the complex NUMA properties of AMD's Infinity Fabric grid, both for intra-socket latencies, and inter-socket latencies. The CLX architecture only exhibits small variances for remote cache access latencies, making these processors potentially better suited for shared memory workloads.

More in-depth analyses of latencies for the Rome processor stress the importance of well thought out data placement when using these processors. This is caused by the I/O-die and the latencies it adds to data transfers that pass through it.

The L1 and L2 cache bandwidth of the AMD processor is close to its theoretical maximum for each vector length we evaluated. This is not the case for the CLX processor. Instead, we observed



(a) Socket-socket RAM-latencies for several NUMA node pairings.

(b) Two AMD EPYC Rome processors with interconnect paths according to measurements.

Figure 16: AMD EPYC Rome 7702: socket-socket memory latencies. The xGMI interfaces to a second processor on the same mainboard cannot be accessed by all cores in the same time due to their asymmetric placement in the I/O-die.

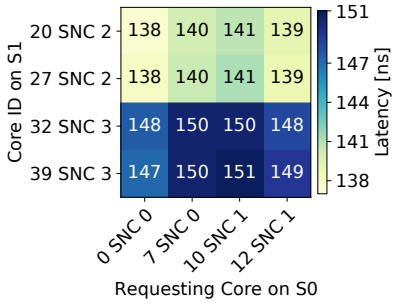


Figure 17: Socket-Socket main memory read latencies for an Intel Xeon Gold 6248. Only the UPI-tile with two UPI links is being used.

L1 bandwidths that were below the theoretical maximum for each vector length and below the sustainable bandwidths Intel defined in [15, Table 2-6, Section 2.2.1.3]. A similar picture emerges for the L2 cache bandwidth. Here, we expected bandwidths similar to the L1 bandwidth for AVX and SSE instructions, as the L1 bandwidth was well within the specified L2 bandwidth. Instead, we observed L2 bandwidths well below the results for the L1. Our cache bandwidth results are in line with other research on these processors [6].

We observed a 41 % lower RAM bandwidth for an entire socket for the CLX processors compared to Rome in our throughput benchmark. This is a result of the lower number of memory channels of CLX processors. Although the Intel design allows one core to access three memory channels when using the SNC configuration, a lower single core bandwidth was observed compared to AMD. We therefore conclude that Rome processors may be better suited for memory bandwidth bound workloads than their CLX counterparts, as long as data sharing among cores is avoided as much as possible.

Future research could focus on the new architectures by Intel and AMD, Ice Lake SP and EPYC Milan. Furthermore, the impact of our findings on real world applications should be considered in future work.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the compute resources and support provided by NHR@FAU for the Caskade Lake SP system. We used the Romeo partition by NEC of the Bull Cluster TAURUS at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for measurements on AMD Rome.

REFERENCES

[1] Advanced Micro Devices, Inc. 2020. *Preliminary Processor Programming Reference (PPR) for AMD Family 17hModel 31h, Revision B0Processors.* https://developer.amd.com/wp-content/resources/55803_B0_PUB_0_91.pdf

[2] Advanced Micro Devices, Inc. 2020. *Software Optimization Guide for AMD Family 17h Models 30h and Greater Processors.* https://developer.amd.com/wp-content/resources/56305.zip

[3] Akhilesh Kumar, Don Soltis, Irma Esmer, Adi Yoaz, and Sailesh Kottapalli. 2017. The New Intel® Xeon® Processor Scalable Family (Formerly Skylake-SP). https://old.hotchips.org/wp-content/uploads/hc_archives/hc29/HC29.22-Tuesday-Pub/HC29.22.90-Server-Pub/HC29.22.930-Xeon-Skylake-sp-Kumar-Intel.pdf Hot Chips 29.

[4] M. Arafa, B. Fahim, S. Kottapalli, A. Kumar, L. P. Looi, S. Mandava, A. Rudoff, I. M. Steiner, B. Valentine, G. Vedaraman, and S. Vora. 2019. Cascade Lake: Next Generation Intel Xeon Scalable Processor. *IEEE Micro* (2019). https://doi.org/10.1109/MM.2019.2899330

[5] T. Burd, N. Beck, S. White, M. Paraschou, N. Kalyanasundharam, G. Donley, A. Smith, L. Hewitt, and S. Naffziger. 2019. "Zeppelin": An SoC for Multichip Architectures. *IEEE Journal of Solid-State Circuits* (2019). https://doi.org/10.1109/JSSC.2018.2873584

[6] C. L. Alappat, J. Hofmann, G. Hager, H. Fehske, A. R. Bishop, and G. Wellein. 2020. Understanding HPC Benchmark Performance on Intel Broadwell and Cascade Lake Processors. In *International Conference on High Performance Computing.* https://doi.org/10.1007/978-3-030-50743-5_21

[7] D. Suggs, D. Bouvier, M. Clark, K. Lepak, and M. Subramony. 2019. AMD "ZEN 2". In *IEEE Hot Chips 31 Symposium (HCS).* https://doi.org/10.1109/HOTCHIPS.2019.8875673

[8] D. Suggs, M. Subramony, and D. Bouvier. 2020. The AMD "Zen 2" Processor. *IEEE Micro* (2020). https://doi.org/10.1109/MM.2020.2974217

[9] A. Fog. 2020. 3. The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers. https://www.agner.org/optimize/microarchitecture.pdf Technical University of Denmark.

[10] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *International Parallel and Distributed Processing Symposium Workshop.* https://doi.org/10.1109/ipdpsw.2015.70

[11] Intel Corporation. [n. d.]. Product brief - Intel Xeon Scalable Platform – Second Generation - Intel Xeon Scalable Processors. https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/2nd-gen-xeon-scalable-processors-brief-Feb-2020-2.pdf

[12] Intel Corporation. 2017. *Intel Xeon Processor Scalable Memory Family Uncore Performance Monitoring.* https://www.intel.com/content/www/us/en/processors/

xeon/scalable/xeon-scalable-uncore-performance-monitoring-manual.html
[13] Intel Corporation. 2020. *Intel® Xeon® Processor Scalable Family – Specification Update.* https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-scalable-spec-update.pdf
[14] Intel Corporation. 2020. *Second Generation Intel® Xeon® Scalable Processors – Specification Update.* https://www.intel.com/content/www/au/en/products/docs/processors/xeon/2nd-gen-xeon-scalable-spec-update.html
[15] Intel Corporation. 2021. *Intel® 64 and IA-32 Architectures Optimization Reference Manual.* https://software.intel.com/content/dam/develop/external/us/en/documents-tps/64-ia-32-architectures-optimization-manual.pdf
[16] John McCalpin. 2018. Address Hashing in Intel Processors). https://doi.org/10.26153/tsw/13161
[17] John McCalpin. 2021. Mapping Core and L3 Slice Numbering to Die Location in Intel Xeon Scalable Processors). https://doi.org/10.26153/tsw/13119
[18] G. Juckeland, S. Börner, M. Kluge, S. Kölling, W.E. Nagel, S. Pflüger, H. Röding, S. Seidl, T. William, and R. Wloch. 2004. BenchIT — Performance measurement and comparison for scientific applications. In *Parallel Computing*. https://doi.org/10.1016/S0927-5452(04)80064-9
[19] A. Kashyap. 2020. *High Performance Computing: Tuning Guide for AMD EPYC™ 7002 Series Processors.* Advanced Micro Devices, Inc. https://developer.amd.com/wp-content/resources/56827-1-0.pdf
[20] Daniel Molka. 2017. *Performance Analysis of Complex Shared Memory Systems.* Ph. D. Dissertation. Technische Universität Dresden, Dresden. https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa-221729
[21] Daniel Molka, Daniel Hackenberg, and Robert Schöne. 2014. Main Memory and Cache Performance of Intel Sandy Bridge and AMD Bulldozer. In *Workshop on Memory Systems Performance and Correctness (MSPC).* https://doi.org/10.1145/2618128.2618129
[22] D. Molka, D. Hackenberg, R. Schöne, and M. S. Müller. 2009. Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System. In *18th International Conference on Parallel Architectures and Compilation Techniques (PACT).* https://doi.org/10.1109/PACT.2009.22
[23] Daniel Molka, Daniel Hackenberg, Robert Schöne, and Wolfgang E. Nagel. 2015. Cache Coherence Protocol and Memory Performance of the Intel Haswell-EP Architecture. In *44th International Conference on Parallel Processing (ICPP).* https://doi.org/10.1109/ICPP.2015.83
[24] D. Mulnix. 2017. Intel® Xeon® Processor Scalable Family Technical Overview. https://software.intel.com/en-us/articles/intel-xeon-processor-scalable-family-technical-overview
[25] Samuel Naffziger, Noah Beck, Thomas Burd, Kevin Lepak, Gabriel H. Loh, Mahesh Subramony, and Sean White. 2021. Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA).* 57–70. https://doi.org/10.1109/ISCA52012.2021.00014
[26] S. Naffziger, K. Lepak, M. Paraschou, and M. Subramony. 2020. 2.2 AMD Chiplet Architecture for High-Performance Server and Desktop Products. In *International Solid- State Circuits Conference (ISSCC).* https://doi.org/10.1109/ISSCC19947.2020.9063103
[27] Sabela Ramos and Torsten Hoefler. 2013. Modeling Communication in Cache-Coherent SMP Systems: A Case-Study with Xeon Phi. In *22nd International Symposium on High-Performance Parallel and Distributed Computing (HPDC).* https://doi.org/10.1145/2462902.2462916
[28] Robert Schöne, Thomas Ilsche, Mario Bielert, Andreas Gocht, and Daniel Hackenberg. 2019. Energy Efficiency Features of the Intel Skylake-SP Processor and Their Impact on Performance. In *International Conference on High Performance Computing & Simulation (HPCS).* https://doi.org/10.1109/HPCS48598.2019.9188239
[29] Robert Schöne, Thomas Ilsche, Mario Bielert, Markus Velten, Markus Schmidl, and Daniel Hackenberg. 2021. Energy Efficiency Aspects of the AMD Zen 2 Architecture. In *2021 IEEE International Conference on Cluster Computing (CLUSTER).* 562–571. https://doi.org/10.1109/Cluster48925.2021.00087
[30] Erich Strohmaier, Jack Dongarra, Horst Simon, Martin Meuer, and Hans Meuer. 2021. TOP500. https://top500.org (accessed 2022-01-07).
[31] S. M. Tam, H. Muljono, M. Huang, S. Iyer, K. Royneogi, N. Satti, R. Qureshi, W. Chen, T. Wang, H. Hsieh, S. Vora, and E. Wang. 2018. SkyLake-SP: A 14nm 28-Core Xeon® processor. In *International Solid - State Circuits Conference (ISSCC).* https://doi.org/10.1109/ISSCC.2018.8310170
[32] J. Treibig, G. Hager, and G. Wellein. 2010. LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments. In *First International Workshop on Parallel Software Tools and Tool Infrastructures (PSTI).* https://doi.org/10.1109/ICPPW.2010.38
[33] Yuval Yarom and Katrina Falkner. 2014. FLUSH+ RELOAD: A high resolution, low noise, L3 cache side-channel attack. In *23rd USENIX Security Symposium.* https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-yarom.pdf

A CACHE AND MAIN MEMORY LATENCIES - SUMMARY

Table 2: AMD EPYC 7702 – memory read latencies: memory accesses on one socket. *Shared* cachelines fetched from another CCX are read from RAM, therefore they are listed for local CCX reads only. *Invalid* cache lines are fetched from RAM at all times.

		Latency in [ns] ([cycles])			
Source	**State**	**L1**	**L2**	**L3**	**RAM**
Local	M/O/E/S	2 (4)	6 (12)		
Same	M/E	39 (78)	37 (74)	19.5 (39)	
CCX	O/S	36 (72)	37 (74)		110 (220)
Same	M/E	123.5 (247)	126.5 (253)	120.5 (241)	
CCD	O/S	102.5 (205)	108 (216)	108 (216)	
NUMA 1	M/E	129 (258)	131.5 (263)	125.5 (251)	115 (230)
	O/S	107 (214)	114 (228)	115 (230)	
NUMA 2	M/E	142 (284)	145.5 (291)	140 (280)	144 (248)
	O/S	115.5 (231)	122.5 (245)	143 (246)	
NUMA 3	M/E	148.5 (297)	152 (304)	146 (292)	127.5 (255)
	O/S	120 (240)	126 (252)	151.5 (253)	

Table 3: Intel Xeon Gold 6248 – memory read latencies: memory accesses on one socket. *Modified* and *Exclusive* cache lines have the same latencies, except for remote L1 accesses. *Shared* and *Forwarded* cache lines can be read with identical latencies. *Invalid* cache lines are fetched from RAM at all times.

		Latency in [ns] ([cycles])			
Source	**State**	**L1**	**L2**	**L3**	**RAM**
Local	M/E/S/F	1.6 (4)	5.6 (14)		
Same SNC	M	50.8–52.8 (127–132)	48.8–50.4 (122–126)	21.6 (54)	80 (200)
	E	48–50 (120–125)			
	S/F	20 (50)			
Other SNC	M	56.4–58 (141–145)	54.4–56 (136–140)	25.6 (64)	86.4 (216)
	E	53.6–55.2 (134–138)	54–56 (135–140)		
	S/F	25.6 (64)			