COMPESCE: A Co-design Approach for memory subsystem Performance Analysis in HPC many-cores

Antoni Portero[1][0000−0003−1319−6404], Carlos Falquez[1][0000−0003−0382−7743], Nam Ho[1][0000−0002−6973−4120], Polydoros Petrakis[2][0000−0002−0224−5808], Stepan Nassyr[1][0000−0002−0035−244X], Manolis Marazakis[2][0000−0002−4768−3289], Romain Dolbeau[3][0000−0002−4466−8948], Jorge Alejandro Nocua Cifuentes[4][0000−0003−1148−7697], Luis Bertran Alvarez[4][0009−0006−0380−0953], Dirk Pleiter[5][0000−0001−7296−7817], and Estela Suarez[1][0000−0003−0748−7264]

[1] Jülich Supercomputing Centre, Novel System Architectures Design, Forschungszentrum Jülich GmbH, Jülich, Germany {a.portero, c.falquez, n.ho, s.nassyr, e.suarez}@fz-juelich.de
[2] Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH), Heraklion, Greece {ppetrak,maraz}@ics.forth.gr
[3] *SiPearl*, Rennes, France romain.dolbeau@sipearl.com
[4] *ATOS*, Les Clayes-sous-Bois, France {luis.bertranalvarez@atos.net,alejandro.nocua}@atos.net
[5] *KTH*, Royal Institute of Technology Stockholm, Sweden pleiter@kth.se

Abstract. This paper explores the memory subsystem design through gem5 simulations of a non-uniform memory access (NUMA) architecture with ARM cores equipped with vector engines. And connected to a Network-on-Chip (NoC) following the Coherent Hub Interface (CHI) protocol. The study quantifies the benefits of vectorization, prefetching, and multichannel NoC configurations using a benchmark for generating memory patterns and indexed accesses. The outcomes provide insights into improving bus utilization and bandwidth and reducing stalls in the system. The paper proposes hardware/software (HW/SW) advancements to reach and use the HBM device with a higher percentage than 80% at the memory controllers in the simulated manycore system.

Keywords: Co-design · HPC · Network on Chip · gem5

1 Introduction

ARM-based high-performance processors have lately joined the High-performance computing (HPC) sector, appearing on the Top500 list and proving that ARM-based systems can deliver significantly high computing performance [1]. An example of a relatively recent successful deployment of ARM-based systems is the Fugaku supercomputer at Riken. Being ranked #1 on the Top500 list in 2020 [2], the Fugaku supercomputer outperformed all competitors by using the Fujitsu

A64FX processor, one of the most potent ARM-based processors available today [3] [4]. Key innovations that led to the success of the Fugaku design are the Scalable Vector Extension (SVE) – a SIMD extension introduced by ARM [5,6].

This paper explores a path for accurately simulating an HPC chiplet-based processor [7]. To do this, we simulate the system before building the silicon and perform co-design exploration to determine the optimal hardware parameters for executing applications and kernels of interest. Such kernels, which are abstractions of the most characteristic HPC codes, represent the most computationally and memory intensive parts of the software that runs on a supercomputer.

The primary focus or this paper is the memory system, which offers more potential for improvement than the Central Processing Unit (CPU). Processors in the market implementing the AArch64 architecture already include the SVE.

Besides, the memory system design must be large enough to handle applications, libraries and codes with high bandwidth demands, rather than linear algebra programs such as GEneral Matrix to matrix MultiplicationS (i.e. HPL [8], GEMMS) architectures [9].

The HW/SW co-design approach relies on a reliable and accurate hardware simulation via the gem5 simulator [34]. The gem5 simulator is a modular, open-source, computer-system architecture research platform containing system-level architecture and processor microarchitecture features. The simulation setup description follows the AMBA-CHI protocol implemented in gem5 Ruby for the network. Furthermore, it provides a cycle-accurate model of the CPU [10].

The benchmark selected for our study is Spatter [11], which provides a tunable and configurable framework to test a variety of indexed access patterns, including variations of the Gather/Scatter patterns that are observed in HPC applications.

Our setup depicts a part of the chiplet with all its components. It is a set of RISC CPUs supporting the Scalable Vector Extension (SVE). These CPU cores are connected with a two-dimensional mesh Network-on-Chip (2D MESH NoC). The simulated setup has similar hardware components as encountered in current state-of-the-art hardware designs as A64FX[12], Graviton2 [13] or Graviton3 [14]. The paper explores the most critical knobs for optimising the memory subsystem. The results show a configuration that can benefit from external memory modules of High Bandwidth Memory (HBM2 [30])). A rational explanation about the specific architecture decisions are described further in the paper (see section 4 Fig. 1a presents a diagram of the architecture, and the specific values for the simulations are in Table 1, first column).

The contributions of the paper can be divided into two categories:

- The paper presents a HW/SW co-design methodology that can help to create a manycore processor for an HPC system in which the memory system is composed of a NoC AMBA-CHI configuration where the memory controller utilisation is higher than 80%. In terms of bandwidth, the System Cache Group (SCG) (which corresponds to a quadrant of the chiplet) achieves more than 250 GB/s; if the chiplet is composed of four quadrants and each

quadrant has one HBM2 [18] module, the chiplet would reach a bandwidth higher than 1 TB/s.
- The paper describes the main components involved and how to employ them to get such bandwidth performance. Our records show that few manycore processor chips can deal deftly with HBM [16], and in manycore systems, the NoC or memory wall could be the bottleneck of all system [15].

The rest of the paper can be divided into seven sections. Section 2 presents the background and motivation. Section 3 explains the related work. Section 4 depicts the proposed HPC architecture; section 5 describes the design space exploration methodology. Section 6 presents the case studies. Finally, the last section has the conclusions and future work.

2 Background and Motivation

The research objective is to reproduce a methodology via simulations of the proper architecture that ultimately benefits from the external HBM memories where the sustainable bandwidth (BW) at the CPUs is 80% or higher.

Existing, many-core designs benefit from HBM modules [16], but without the correct co-design for new designs, the NoC can become a bottleneck [15].

The Fujitsu A64FX [19] CPU has a sustainable bandwidth of 62% when running the Stream benchmark (triad) [20]; unless *zfill* compile flag is specified, which eliminates unnecessary memory accesses, then the bandwidth achieved is 80%—allowing a high usage of the external HBM devices.

Although the research question can be broader, the technical challenge is if achieving one terabyte per second in a manycore chiplet with 64 ARM cores armed with SVE 2x256bits, and which network-on-chip design would allow such performance?

The significance of the research is related to uncovering architectures that attenuate the memory wall, which implies solving the processor/memory performance gap. The memory wall limits many current HPC computation applications [21]. Therefore, memory-bound codes profit from the highest sustainable bandwidth at the core level. This research manuscript's significance and relevance are finding strategies to achieve the correct usage of the HBMs and which are the leading software and hardware features that bring an optimal design.

The paper innovations are about the Design Space Exploration (DSE) methodology to encounter optimal design, where the complex problem of the optimal memory system splits into more tractable subproblems with uni-direction constraints propagation. The quantitative assessment for the optimal architecture features, how the benchmarks are utilised to stress the memory system and observe stalls, how to detect them, and offer solutions. Many experiments use the Spatter [11] benchmark. The microbenchmark Spatter is used to assess the impact of indexed access patterns of Gather and Scatter (G/S) operations, which are widely used in many modern HPC applications. The design of Spatter is

composed of Gather and Scatter kernels that enable users to benchmark different access patterns to understand the implications of memory prefetching and compiler development.

3 Related work

Our effort goes toward co-designing the memory subsystem for an HPC architecture based on RISC CPUs technology. Similar previous works are Qureshi, Yasir Mahmood at al. [23] with gem5-X an infrastructure to simulate an Out-of-Order cluster with 3D high-performance memory (HBM2). But our effort, rather than embedded systems, emphasises HPC architectures, adding Vector Engines to the CPUs and networks with high bandwidth and low latencies. Our simulation choice was gem5[34] because it is a full-system (FS) architectural simulator widely used in academia and industry, as it supports multiple Instruction Set Architectures (ISAs), such as x86, ARMv8, RISCV, and others. In addition to various ISAs, it supports different CPU models for these ISAs, such as atomic, in-order, and out-of-order (OoO) CPU models, as well as multiple caching protocols and coherences. On the memory side, it supports many traditional and emerging memories. Further, gem5 supports FS simulations via several Linux-based operating systems, enabling applications to execute as they would on a real platform. Although gem5 is cycle-accurate and detailed in statistics created during execution, the turnaround is a long simulation time.

Other simulators for co-design of HPC systems are based on SystemC-TLM [35] plus QEMU; however, such a description can miss details on the out-of-order paths or trace base [36], where traces must come from a very similar architecture and hence not effective for detailed and heterogenous design exploration. Other simulators have their niche [37,38,39] for specific ISAs, but gem5 is more publicly proven.

4 HPC architecture

The architecture selected for the simulation is in several parameters similar to the Graviton3 (G3). The G3 comprises 64 Neoverse-V1 [40] cores, and each tile of the NoC or CPU has two cores with two SVE of 256 bits. Here, we describe the main differences between our setup defined as *Open Processor for Inception Systems* OPIS (see Table 1 parameters details), the G3 architecture, and A64FX. The simulated designs [10] setup is not outside the parameters ARM offers for their architectures. Moreover, we employ it to explore the memory system. For this study, when there is slack, we always take the larger size of the memories, but we design the decision to keep the two cores per tile as G3. In the case of G3, there is only one NUMA domain for the complete chiplet, while our setup has 4 NUMA domains of 16 cores each, also called System Cache Groups (SCGs). The design is for the SCG2 quadrant: the NUMA domain on the top-left of the chiplet (see Fig. 1a). Another difference is that G3 is connected to four external Double Data Rate (DDR5) memories with eight channels for 64 cores, while we are

simulating a configuration with four HBM2 modules with 32 memory channels in total. We only simulate one SCG, with one HBM2 module and eight channels, with 35.82 GiB/s per channel. G3 can exhaust the bandwidth with a few cores executing a memory-bound application like *STREAM*. The ARM CMN-700 [41] for G3 setup seems to follow an LVS1 strategy, meaning one channel per Virtual Network (VNET).

At the same time, we are exploring a higher number of VNETs (i.e. LVS2) to benefit from the higher bandwidth available from the HBM2.

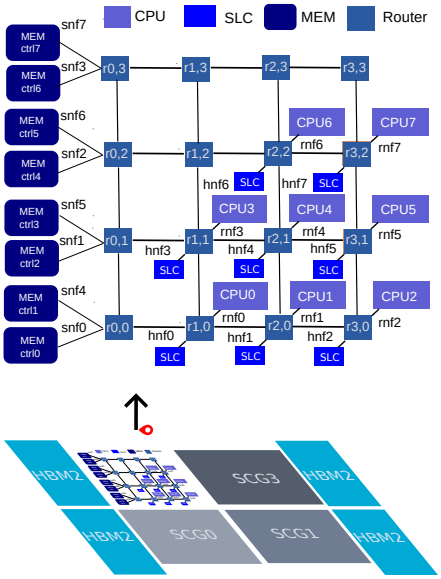
OPIS SCG (gem5)	Architectural Parameters
Clocks	System: 1.6 GHz; CPU: 2.4 GHz; NoC: 2.0 Ghz
CPU	#Cores: 16; Adjusted A76; Branch Pred.: BiMode; **Vector Unit: 2xSVE; None, SVE length:{256}**
L1	Line size: 64B; Size: 64∼KiB; Associativity: 4-way; Inclusion policy: strict inclusive ; TBEs: 256; Hit latency: 2-cycles (L1-D), 1-cycles (L1-I);
L2	Unified cache; Line size: 64B; Size: 1 MiB; Associativity: 8-way; Hit latency: 4-cycles; Inclusion policy: strict inclusive ; TBEs: 256 ;
SLC	Shared SLC cache; #Slices: 16; Line size: 64B; Associativity: 16-way; Hit latency: 20-cycles; Inclusion policy: Exclusive; TBEs: 256 per slice; Size per slice: 4 MiB;
NoC	Interconnect: CMN-650, Model: Garnet 3.0; Protocol: AMBA-CHI; Flit width: 64B; Router latency: 1-cycle; Link latency: 1-cycle; #VNETs: {lvs1:4, lvs2:7} ; Routing XY Topology: Mesh: 4x4; **Link configuration: {lvs1, lvs2}**
Memory Model	HBM2; #Channel: 8; Size: 2x8 GiB Bandwidth per channel: 35.82 GiB/s
Prefetcher	**off, on**

Table 1: Details of fixed and **explored** parameters setup for one OPIS SCG architecture

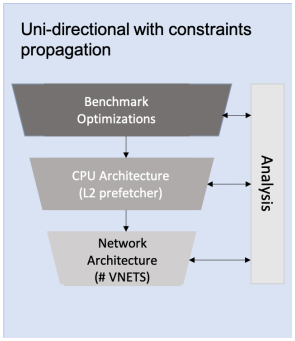
The main differences between G3 and A64FX[25] are that the A64FX can deal with external HBM memories (see Fugaku arch [26,12]). The A64FX chip includes four NUMA domains named Core Memory Group (CMG). Each CMG contains 12 cores for application execution. Each core has an SVE of 2×512 bits instead of the 2×256 like in G3. In addition, the A64FX Tofu network interconnection is a double-ring buffer[24] instead of the 2D MESH from G3 and OPIS.

Prefetcher: Our simulation gem5-based setup supports different prefetching schemes. In this paper, we focus on the next-line scheme [27], configured at the

L2 cache. For every memory access to the L2 cache, the prefetcher immediately triggers sequential cache accesses (up to 32 cache lines) and stores prefetch candidates in the prefetch queue. Before sending prefetch requests to the memory, the prefetcher needs to search (snoop) in the cache and drops the prefetch request if there is duplicate data. The prefetcher and the gather scatter hardware mechanisms are black boxes in G3, while for A64FX, documentation of the mechanisms is available [25].



(a) System Cache Group (SCG) Diagram



(b) Co-design Methodology: Unidirectional with constraints propagation for memory subsystem study.

Fig. 1: a) Chiplet with SCG Diagram, b) Unidirectional Methodology

The Fig 1a) presents a chiplet with 4 NUMA regions in the down part. Each NUMA region is named *SCG0-3* (System Cache Group). Each *SCG* has an external HBM2 device connected. Our simulations focus on the *SCG2* that is up-left. In the upper part of Fig 1a, a 2D MESH network is depicted, and routers are connected in the crosspoints of the SLC/L3 memories and the CPU. The CPU is composed of 1, 2 or 4 cores with 2x256 SVE vector engines. Each CPU has a private per core L1 and shared L2 per cluster. The Subordinate Nodes SN belongs to CHI protocol fundamentals [17]; SNFs nodes connect to memory devices that back the coherent memory space. Eight SNFs nodes connect the number of channels of the HBM2 device [18]. The CPUs work as Coherent Home Nodes (HN-Fs) to compose all requests to coherent memory and issue snoops to Request Nodes (RN-Fs).

5 Design Space Exploration Methodology

Architecture Virtualisation is fundamental for achieving rapid Design Space Exploration (DSE) of HPC microprocessors. The HPC systems design space is vast, with many dimensions to explore. It is the task of the design architects and co-design developers to evaluate and prioritise the most relevant knobs to get the most optimal design.

For this paper, our methodology approach follows the near-optimal design space exploration [28]. In the mentioned DSE, in contrast to existent DSE methods, the partition between the steps is selected so that they can be connected through unidirectional constraint propagation instead of bidirectional constraints. This route achieves a near-optimal result because constraints are not overlooked, which happens when the steps are considered partially independent. Moreover, it divides intractable problems into manageable subproblems.

The DSE methodology is applied to evaluate the memory sub-system. The projected DSE framework pursues step-wise with the division of all the available design space options into cases that correspond to these sub-problems, where the top-down division principle rigorously applies to top-down splits, which are connected through unidirectional constraint propagation.

The architecture exploration considers the SCG, which encloses a set of cores connected in a mesh NoC. The NoC protocol is CHI [29] and the memory model that represents a High Bandwidth Memory (HBM2 [30]) (see Fig. 1b).

5.1 Co-design Exploration: Memory sub-system

The first split divides the entire design space exploration DSE into two subspaces: On one side, the dimensions that belong to the SW optimisations; on the other, the dimensions that belong to the HW ones. From the SW side, we are not experimenting with the different compilers or manual code optimisations. Instead, vectorisation is a crucial dimension affecting the code's performance; for this study, we use auto-vectorisation. Since the developers must adapt the code to the platform, e.g. via loop reordering and specific flags for the compiler, typically, the first step is to vectorise the code to enhance the vector engine utilisation and hence, its influence on the performance.

Neoverse V1	*Possible knobs value*	*Chosen Value*
Num cores per tile	1, 2, 4	**2**
SVE size	2x256	**2x256**
L2 size	512 KiB OR 1 MiB (4 banks)	**1 MiB**
SLC size	2 MiB to 4 MiB, 16-way set associative	**4 MiB**

Table 2: Neoverse V1 knobs fixed for the exploration (in bold)

Regarding the HW size, this sub-space can be divided again between CPU knobs and NoC knobs. Regarding the CPU knobs, we set them to a specific

value: Although the CPU can have 1, 2 or 4 ARM V1 cores, we selected the two cores' organisation per tile (or in each router) with a total of 16 cores per SCG (see Fig 1a). One characteristic parameter is the vector size. We specified 2×-width vector units similar to the ARM-V1 architecture [31], with a width of 256 bits. Although there is also flexibility in the L2 and SLC/L3 sizes, we always opted for the larger cache sizes. L2 size can be 512 KiB or 1 MiB (4 banks). For the System-level cache (SLC), we have 1 Bank per core duplex and a size of 2 MiB to 4 MiB, 16-way set associative. Finally, for the memory system and the CPU, the hardware prefetcher in the L2 is a knob that enhances the performance.

The other subspace or subproblem related to the network knobs for this article is the network topology and the routing algorithm. We selected the 2D Mesh, XY algorithm, again similar to architectures like Graviton3 [32]. The external memory device is a High Bandwidth Memory (HBM2) [18,23].

Another knob we changed is the number of NoC physical links per VNET in the routers, intending to find the setup that most benefits from the external memory devices. Link-VNET-Support-1 (LVS1) is defined as one link per VNET, where the 4 VNETS defined in AMBA-CHI Cache Coherence (CC) protocol are: 0-request, 1-snoop, 2-response and 3-data (REQ, SNP, RSP and DATA). The second possibility (LVS2) uses two physical links per VNET. In the case of the snoop VNET (SNP), we still use one link, as there is little traffic in the examples under study. Hence, the LVS2 NoC configuration uses seven links in total.

5.2 Model Validation

To increase the confidence of the presented results, we have executed kernels in real machines prototype and compared them with a corresponding gem5 model. For instance, if the machine is the N1SDP [33] prototype board with two sockets, each containing 2 ARM-N1 cores. We developed the gem5 version of the platform using the board's datasheet. In addition, we considered the Performance Monitoring Unit (PMU) counters and compared the results. Finally, we apply a multi-level consistency validation [10] for reference applications and kernels.

6 Case studies

The section describes the design exploration of the memory sub-system through the unidirectional with constraints propagation. The subsection defines the Spatter Uniform Stride (US) behaviour in the CPU data path. It simplifies the compiler generation of Gather/Scatter instructions. Each thread performs some portion of the iteration. In addition, each thread's block gathers into a local destination buffer to ensure high performance. The effect avoids false sharing.

The Uniform Stride index buffer in Spatter is specified with `UNIFORM:N:STRIDE`. It generates an index buffer of size N with STRIDE. For example, the index buffer generated by `UNIFORM:8:2` is [0,2,4,8,10,12,14,16] or `UNIFORM:8:8` is [0,8,16,24,32,40,48,56]. This manuscript's Spatter Uniform Stride is similar to

the Stream benchmark [20]. The added value offers extra information about the pattern's index of common HPC applications too large for cycle-accurate simulation.

6.1 Unidirectional approach

The Unidirectional approach with constraints propagation method begins in the dimension with a higher impact on the outcomes. Then, when the space exploration dimension is optimised, we continue with the following one with the constraints from the previous one. Hence, no loops or iterations are needed. For example, starting from the previous methodology, we observed and analysed the impact of vectorisation; we then took the vectorised code and used it for the prefetching effect afterwards for the increase of links, the LVS1 vs LVS2 study. The expectation is to find an architecture where the external memory controllers' usage is 80% or higher. After continuing with the following dimensions, for the reasons analysed further below.

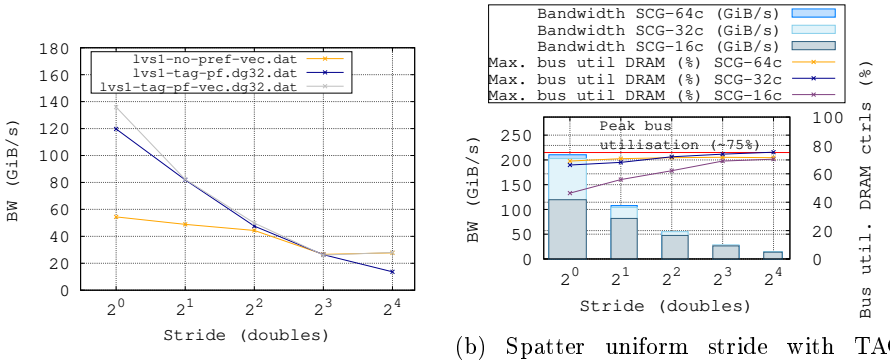
The dimension under study are a) vectorisation, b) enhancements due to prefetcher, and c) enhancing the number of links from LVS1 to LVS2. Table 3 presents the improvements due to code vectorisation, which ranges from 23% to 97%. Then we observed the impact of the prefetcher on the bandwidth (see Sub. Fig. 2a), showing improvements of $x2$. The Fig. 2b) shows a saturation in the NoC due LVS1 configuration.

	Spatter Uniform Stride: BW increase due to vectorisation				
Configuration	8:1	8:2	8:4	8:8	8:16
Performance Improvement(%)	60.3	93.9	96.8	29.3	23

Table 3: Spatter Uniform Stride: Bandwidth increase due to vectorisation

Bandwidth in the SCG with LVS1: We experimented with the OPIS SCG with LVS1, where we increased the number of cores per CPU. Instead of using the default SCG with 16 cores, we increased it to 4 and 8 CPUs per router, having an SCG of 32 cores and 64 cores, respectively, all the cores with the prefetcher enabled. The objective is to maximise the bus utilisation in the memory controllers. In addition, we want to observe the optimal use of the HBM2 memory devices. As the number of cores increases, the number of data requests to the memories increases. We want to observe the Miss Status Holding Register (MSHR) since it is the buffer to track outstanding requests.

The results are depicted in Figure 2b, showing the differences between the bandwidth observable in the memory controllers as a percentage of the bus utilisation versus the bandwidth reported by the Spatter application. While Spatter reports a lower bandwidth when the stride increases (as it measures effective bandwidth only), the bus utilisation increases to 75%, which means that from

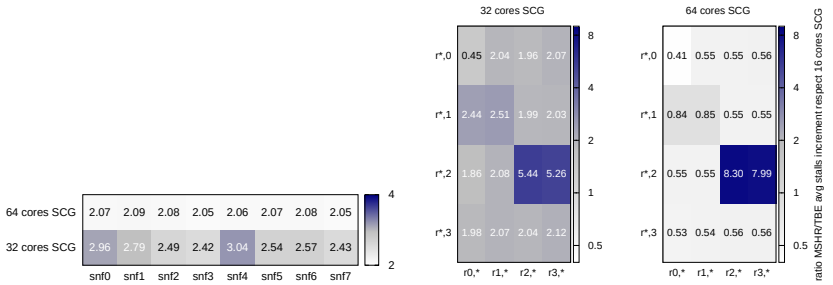


(a) Impact of the prefetcher on the bandwidth

(b) Spatter uniform stride with TAG prefetcher degree 32 executed in OPIS-SCG (LVS1)

Fig. 2: a) Bandwidth utilisation improvements due to data prefetching, b) HMB2 Bandwidth saturation due to LVS1-NoC.

the 286.56 GiB that can provide the HBM2 module, only 75% (215 GiB/s) is achieved. Bus bandwidth utilisations start to saturate at *stride-8* and *16*, even when using many more cores and increasing the outstanding requests by 4×.



(a) Memory controllers (snf) stalls increment from baseline (16 cores SCG) for 32 and 64 cores SCG

(b) SLC (hnf) stalls increment from baseline (16 cores SCG) for 32 and 64 cores SCG

Fig. 3: a) Stalls in memory controllers, b) stalls in the SLCs memories

Therefore, even when the number of MSHR requests increases since more cores (i.e. SCG 32 and 64 cores) request data from the memory controllers, the bandwidth at the memory controllers saturates when it achieves 75% of the peak HBM2 bandwidth. The LVS1 configuration can only partially benefit from the HBM2 bandwidth available. To understand the reason, we observe the situation of the TBE_avg (similar to MSHR but named differently in the gem5 simulator) in the routers for the point 2^4, which is our point of interest. This means there is

saturation in LVS1, and the HBM device provides 3/4 of its bandwidth capacity. The architects decide if this limitation is economically viable or if using cheaper devices with lower bandwidth (i.e. DDR5) but larger capacity is a better option.

Increasing stride will downsize the requirements for resource allocation in the micro-architecture (e.g. register file allocation) and thus could reduce stalls. Therefore, with a higher stride, the CPU can stress the memory system by sending more in-flight requests. To observe the saturated point of bus bandwidth utilisation at the memory controller, we ran experiments via a variant stride from 1 to 16. A *stride-16* will advise us of the maximum bus bandwidth utilisation.

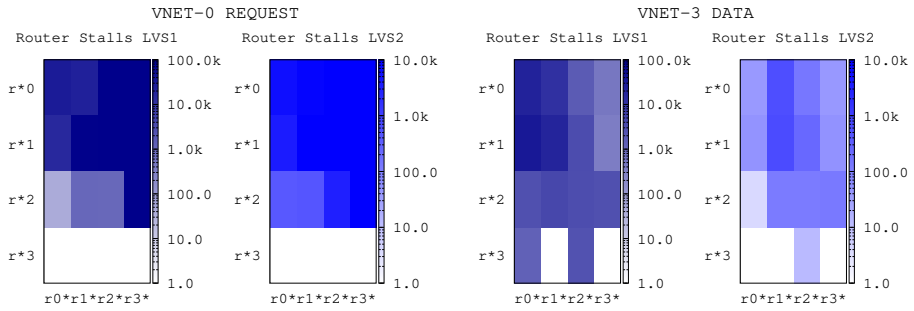


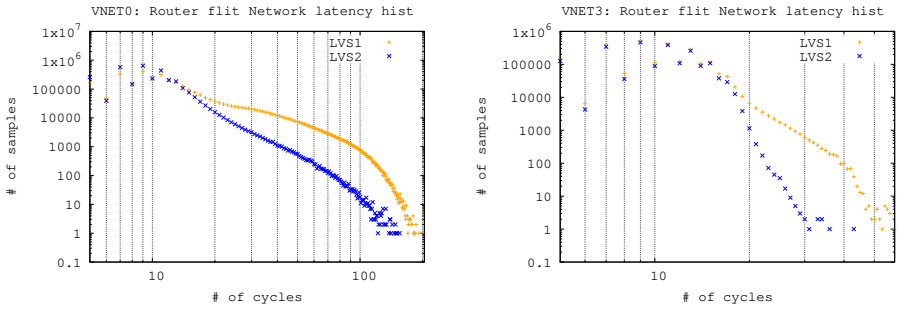
Fig. 4: Heatmap with router stalls (LVS1 vs LVS2). (a) REQUEST (b) DATA VNETs.

Heatmaps of the NoC and memory controllers for LVS1 configuration Figure 3a, presents the increment of stalls from the baseline architecture (i.e. 16-core SCG) to the architectures with 32- and 64-core SCG in the memory controllers. The increment in cores produces more outstanding requests to the memory controllers. Nevertheless, there are a pair of bottlenecks, a factor of $\times 3$ in stalls in *snf0* and *snf4* (see Figure 3a), and also a bottleneck in the tiles see Figure 3b in the routers far away from the memory controllers (i.e. *r22*, *r32*), with increments higher than 5 and 8 times for 32 and 64 cores configuration, respectively.

NoC VNET stalls LVS1 vs LVS2 Previous experiments provide enough insights to suggest that there are better configurations than LVS1 to optimise the usage of the HBM2 devices. Hence, we experimented with the baseline SCG but with the LVS2 configuration to observe the stalls in the routers.

Figure 4 presents another heatmap comparing the number of stalls for all the VNETS; there is no figure for the SNOOP VNET because the packet traffic is meagre in this study, and the RESPOND channel is similar the to REQUEST.

Again, we can observe that the decrease in stalls in LVS2 is a factor of $\times 10$ with respect LVS1.



(a) (b)

Fig. 5: Latencies histogram (LVS1 vs LVS2). (a) REQUEST; (b) DATA VNETS.

NoC VNET latencies LVS1 vs LVS2 In the same experiment, we checked the router flit network latency and created the histogram for the main VNETs. Figure 5 presents how the network latency reduces from the LVS1 to the LVS2 configuration.

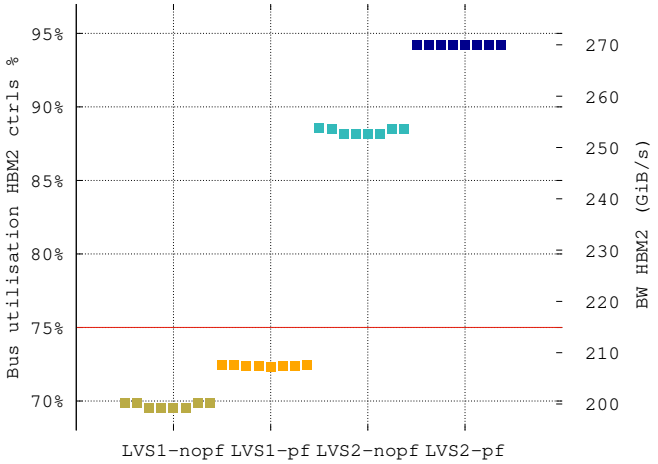


Fig. 6: Bus utilisation for spatter uniform stride, *stride-16* in several OPIS configurations: LVS1 without and with prefetcher(nopf, pf), LVS2 without and with prefetcher (nopf, pf)

Bus utilisation in MEM controllers Figure 6 exhibits the simulation outcomes for uniform *stride-16*. The memory controllers have a utilisation higher than 88% (achieving 252,17 GiB) for LVS2 with the prefetcher disabled and 93% when the prefetcher is enabled. Hence, compared with the LVS1 configuration, LVS2 decreases the stalls in the system and increases the bus utilisation. Therefore, it is recommended to use LVS2 if the external memory is an HBM module and we are interested in its optimal use. Figure 6 presents the bus utilisation of each of the eight memory controller configurations with and without enabled prefetcher. The best configuration is achieved for LVS2 with prefetcher active (*LVS2-pf*).

Insight: Conversely, to the intuition adding more links (i.e. LVS1 to LVS2) without other improvements in the design does not produce better performance. It is when the combination of several design features like vectorisation and aggressiveness of the prefetcher and extra links in the NoC produces a high usage of the memory device.

7 Conclusions and future work

The paper presents a methodology to explore the memory subsystem in a many-core system for HPC connected to HBM devices using the Spatter benchmark. The methodology permits finding bottlenecks in the system. For example, it was able to detect that a NoC configuration with only one link per VNET (defined as LVS1) can only use 75% of the bandwidth of the HBM2 device. Nevertheless, the additional analysis allowed us to observe that a configuration with two links per VNET (i.e. LVS2), with vectorised code and enough aggressiveness of the prefetcher, increases resource utilisation above 90% (observation at the memory controller).

In the future, using novel 3D stacked memory chiplets (i.e. HBM3), we would like to model new network topologies and routing algorithms other than 2D MESH to analyse if there are designs that bring benefit (i.e. reducing stalls and latencies). Another dimension not described in the paper is the power envelope; it would be relevant to estimate the overheads due to, for example, the aggressiveness of the prefetching or due to more oversized packet routing buses. Finally, it is still an open question whether our designs can keep the pace of future characteristics' external memories and use them efficiently.

Acknowledgment

This work has been performed in the context of the European Processor Initiative (EPI) project, which has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement №101036168 (EPI-SGA2).

References

1. M. Sato, et al."Co-design and system for the supercomputer "fugaku"," *IEEE Micro*, vol. 42, no. 2, pp. 26–34, 2022.
2. D. Monroe, "Fugaku takes the lead," *Commun. ACM*, vol. 64, no. 1, pp. 16–18, 2021.
3. S. Yamamura, et al. , "A64FX: 52-core processor designed for the 442petaflops supercomputer fugaku," in *ISSCC, San Francisco, CA, USA, February 20-26, 2022*, pp. 352–354, IEEE, 2022.
4. M. Sato, "The supercomputer "fugaku" and ARM-sve enabled A64FX processor for energy-efficiency and sustained application performance," in *ISPDC 2020*, pp. 1–5.
5. N. Stephens, et al. , "The ARM scalable vector extension," *CoRR*, vol. abs/1803.06185, 2018.
6. J. Lee, et al. , "Extending openmp SIMD support for target specific code and application to ARM SVE," in *Scaling OpenMP for Exascale Performance and Portability - 13th IWOMP 2017*
7. D. Reed, et al., "Reinventing high performance computing: Challenges and opportunities," 2022.
8. A. Petitet, et al., "Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers," December 2018.
9. Di Wu, J. Li, R. Yin, H. Hsiao, Y. Kim, and J. S. Miguel, "Ugemm: Unary computing architecture for gemm applications," in *ISCA*, pp. 377–390, 2020.
10. L. Zaourar et al., "Multilevel simulation-based co-design of next generation HPC microprocessors," (PMBS), St. Louis, MO, USA, 2021, pp. 18-29
11. P. Lavin, E. J. Riedy, R. Vuduc, and J. S. Young, "Spatter: A benchmark suite for evaluating sparse access patterns," *CoRR*, vol. abs/1811.03743, 2018.
12. Sato, M., et al., Co-Design for A64FX Manycore Processor and "Fugaku". *SC20: International Conference For High Performance Computing, Networking, Storage And Analysis*. pp. 1-15 (2020)
13. R. Mathá, D. Kimovski, A. Zabrovskiy, C. Timmerer, and R. Prodan, "Where to encode: A performance analysis of x86 and ARM-based amazon ec2 instances," in *eScience*, pp. 118–127, 2021.
14. ARM, "ARM® Neoverse™ V1- amazon's graviton3 server chip." `https://www.nextplatform.com/2022/05/24/the-value-proposition-for-amazons-graviton3-server-chip/`.
15. ECP, "Milestone M1 Report: HBM2/3 Evaluation on Many-core CPU WBS 2.4, Milestone ECP-MT-1000," *Exascale Computing Project*, June 2018.
16. A. Biswas, "Sapphire Rapids," 2021 IEEE Hot Chips 33 Symposium (HCS), Palo Alto, CA, USA, 2021, pp. 1-22, doi: 10.1109/HCS52781.2021.9566865.
17. ARM, "Learn the architecture - Introducing AMBA CHI," Non-Confidential. Issue 01, 102407_0100_01_e
18. "High bandwidth memory (hbm) dram.", JEDEC, 2020
19. B. Brank, S. Nassyr, F. Pouyan, and D. Pleiter, "Porting applications to ARM-based processors," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 559–566, 2020.
20. McCalpin, J. Memory Bandwidth and Machine Balance in Current High Performance Computers. *(TCCA) Newsletter*. pp. 19-25 (1995,12)
21. S. A. McKee, "Reflections on the memory wall," in *Proceedings of the First Conference on Computing Frontiers, 2004, Ischia, Italy, April 14-16, 2004*

22. Germann, T. Co-design in the Exascale Computing Project. *The International Journal Of High Performance Computing Applications.* **35**, 503-507 (2021)
23. Qureshi, Y., et al. Gem5-X: A Many-Core Heterogeneous Simulation Platform for Architectural Exploration and Optimization. *ACM Trans. Archit. Code Optim.*. **18** (2021,7)
24. Okazaki, et al.,. Supercomputer Fugaku CPU A64FX Realizing High Performance, High-Density Packaging, and Low Power Consumption. *Fujitsu Technical ReviewNo.32020.* (2020,11,11)
25. M. Hondou, "A64fx microarchitecture manual v1.8 released." `https://github.com/fujitsu/A64FX`, 2019.
26. Y. Nakamura,et al., "Fugaku codesign report," tech. rep., FLAGSHIP 2020 Project, RIKEN Center for Computational Science (R-CCS), RIKEN, 03/2022.
27. A. J. Smith, "Sequential program prefetching in memory hierarchies," *Computer*, vol. 11, p. 7–21, dec 1978.
28. A. Kritikakou, F. Catthoor, and C. Goutis, *Scalable and Near-Optimal Design Space Exploration for Embedded Systems.* Springer, 2014.
29. ARM, "AMBA® 5 CHI architecture specification." `https://developer.arm.com/documentation/ihi0050/ea/`, 2020.
30. Jedec, "High bandwidth memory (hbm) dram," Standards JESD235D, Joint Electron Device Engineering Council, Mar 2021.
31. ARM. Developer, "ARM® neoverse™ v1 core, rev:r1p1. technical reference manual," tech. rep., ARM- Advanced RISC Machines, 2021.
32. '/'Inside amazon's graviton3 ARM server processor." `https://www.nextplatform.com/2022/01/04/inside-amazons-graviton3-arm-server-processor` Accessed:2022-10-17.
33. ARM, "ARM® Neoverse™ N1 core - technical reference manual." `https://developer.arm.com/documentation/100616/0401/?lang=en`, 2020.
34. Binkert, N., et al., The gem5 simulator. *ACM SIGARCH Computer Architecture News.* **39**, 1-7 (2011)
35. Ventroux, N., et al. SESAM: An MPSoC simulation environment for dynamic application processing. *2010 10th IEEE CIT.* pp. 1880-1886 (2010)
36. Gómez, C., et al. Design Space Exploration of Next-Generation HPC Machines. *IPDPS, 2019.* pp. 54-65 (2019)
37. Hardavellas, N., et al. SimFlex: A Fast, Accurate, Flexible Full-System Simulation Framework for Performance Evaluation of Server Architecture. *SIGMETRICS Perform. Eval. Rev.*. **31**, 31-34 (2004,3)
38. Magnusson, et al. Simics: A full system simulation platform. *Computer.* **35**, 50-58 (2002)
39. Carlson, et al. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. *SC '11.* pp. 1-12 (2011)
40. "Microarchitecture description ARM v1.",ARM report, 2022
41. ARM, ARM® Neoverse™ CMN-700 Coherent Mesh Network, *Technical Reference Manual*, 102308_0300_05_en, 2022