Enabling CXL Memory Expansion for In-Memory Database Management Systems

Minseon Ahn
Donghun Lee
Jungmin Kim
minseon.ahn@sap.com
domg.hun.lee@sap.com
jungmin.kim@sap.com
SAP Labs Korea
Seoul, South Korea

Oliver Rebholz
oliver.rebholz@sap.com
SAP SE
Walldorf, Baden-Württemberg
Germany

Andrew Chang
Jongmin Gim
Jaemin Jung
Vincent Pham
Krishna T. Malladi
Yang Seok Ki
andrew.c1@samsung.com
gim.jongmin@samsung.com
j.jaemin@samsung.com
tung1.pham@samsung.com
k.tej@samsung.com
yangseok.ki@samsung.com
Samsung Semiconductor Inc.
San Jose, California, USA

ABSTRACT

Limited memory volume is always a performance bottleneck in an in-memory database management system (IMDBMS) as the data size keeps increasing. To overcome the physical memory limitation, heterogeneous and disaggregated computing platforms are proposed, such as Gen-Z, CCIX, OpenCAPI, and CXL. In this work, we introduce flexible CXL memory expansion using a CXL type 3 prototype and evaluate its performance in an IMDBMS. Our evaluation shows that CXL memory devices interfaced with PCIe Gen5 are appropriate for memory expansion with nearly no throughput degradation in OLTP workloads and less than 8% throughput degradation in OLAP workloads. Thus, CXL memory is a good candidate for memory expansion with lower TCO in IMDBMSs.

CCS CONCEPTS

• **Hardware** → **Emerging interfaces**; • **Information systems** → **Database management system engines**.

KEYWORDS

CXL, Compute Express Link, In-Memory Database, DBMS, Database Management Systems

ACM Reference Format:
Minseon Ahn, Donghun Lee, Jungmin Kim, Oliver Rebholz, Andrew Chang, Jongmin Gim, Jaemin Jung, Vincent Pham, Krishna T. Malladi, and Yang Seok Ki. 2022. Enabling CXL Memory Expansion for In-Memory Database Management Systems. In *Data Management on New Hardware (DaMoN'22), June 13, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3533737.3535090

DaMoN'22, June 13, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9378-2/22/06…$15.00
https://doi.org/10.1145/3533737.3535090

1 INTRODUCTION

Modern applications' growing data needs large DRAM capacity, particularly for in-memory database management systems (IMDBMS). Furthermore, considering data growth after initial deployment, the DRAM capacity of an on-premise server is usually overprovisioned. This results in an increased total cost of ownership (TCO) for IMDBMS servers. Emerging new technologies like NVMe [11], and RDMA [12, 13, 17] provide a flexible and integrated memory view larger than the physical memory. However, this needs application changes as well as resulting in longer latency. To overcome such limitations on physical memory expansion and to provide more flexible solutions, heterogeneous and disaggregated computing platforms, such as Gen-Z [2], CCIX [4], OpenCAPI [1], and most recently CXL (Compute Express Link) [5], are proposed. With wide adoption across the industry, CXL is the most promising candidate to mitigate memory overprovisioning issues.

CXL is a new class of interconnect for device connectivity, an open industry standard led by Intel®, and cache coherent interface using PCIe, enabling memory expansion and heterogeneous memory for disaggregated computing platforms. CXL has an alternate protocol that runs across the standard PCIe 5.0 physical layer, consisting of three protocols; (1) CXL.io for discovery, configuration, register access, and interrupt, (2) CXL.cache for device access to processor memory, and (3) CXL.memory for processor access to device attached memory. There are three types of CXL devices. Type 1 is a CXL device without host-managed device memory like NIC using CXL.io and CXL.cache. Type 2 is a CXL device with host-managed device memory like GPU or external computing units using all 3 CXL protocols. Type 3 is a CXL device only with host-managed device memory using CXL.memory. A typical application of type 3 is memory expansion.

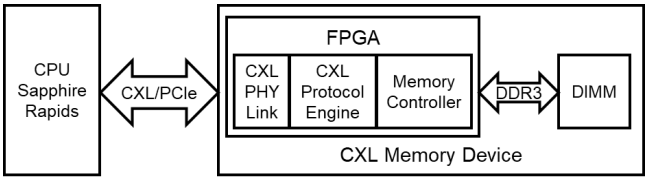


Figure 1: CXL prototype diagram

In this work, we introduce CXL type 3 memory expansion for IMDBMS [10]. First, we propose a CXL-based solution for flexible memory expansion in our IMDBMS to address the overprovisioning issue. Second, we introduce a prototype of CXL type 3 memory devices and a complete working system. Third, we evaluate the performance with common database benchmark tests. Our evaluation shows that CXL memory expansion has nearly no throughput degradation with TPC-C, one of OLTP workloads, and less than 8% throughput degradation with TPC-DS, one of OLAP workloads. This is a promising solution given the limited capability of the prototype. With PCIe Gen5 performance at the final product stage, CXL memory expansion is expected to provide competitive solutions to optimize TCO in IMDBMSs without large performance penalty.

The remainder of this paper is organized as follows: Section 2 discusses memory expansion in IMDBMSs. Section 3 introduces CXL memory expansion devices. The performance evaluation is addressed in Section 4. Section 5 represents the related work and Section 6 concludes the paper.

2 MEMORY EXPANSION IN IMDBMS

IMDBMSs widely support hybrid transactional and analytical processing (HTAP) [16]. In this work, SAP HANA in-memory database platform [10] is used as a base platform. It adopts the columnar storage [15], storing the data of each column in the read-optimized main storage and maintaining the separate delta storage for optimized writes. Additionally, a portion of memory is allocated for the operational data to keep the intermediate results while processing a query.

There are two options to enable the memory expansion using CXL memory devices in our IMDBMS. In the first, the additional memory space of the CXL device could be uniformly integrated with the host memory space. This allows both operational memory and main storage to be allocated in CXL memory. However, the random accesses to the operational data can degrade overall performance owing to longer access latency of the CXL device compared to the host DRAMs. In the second option, the CXL memory device is used only for the main storage. The delta storage and the operational data are stored in the host DRAMs. Like the approach used in the persistent memory [9], a prefetching scheme can effectively hide the longer latency of the CXL device when the main storage is sequentially accessed. In this work, we use the second option in our IMDBMS to take advantage of the prefetch.

3 CXL MEMORY EXPANSION

This section introduces a prototype of CXL type 3 memory expansion devices in E3.S form factor. It implements CXL.mem and CXL.io commands defined in CXL1.1 specification, carrying 128GB DRAM as media and supporting a theoretical bandwidth of 16GB/s with PCIe Gen4x8 as the bottleneck. As shown in Fig. 1, the prototype consists of a custom FPGA board and a single-channel-based DDR3 DIMM module. The FPGA is composed of CXL PHY link supporting the connection to CPU, CXL protocol engine managing CXL.mem and CXL.io, and memory controller for the DIMM module. DDR3 DIMM can be replaced with DDR4 or DDR5 DIMMs supporting multiple channels within a single CXL memory device in the future. On the FPGA, the SerDes technology in our prototype runs at 16 Gbps, same as PCIe Gen4 speed. It can be upgraded to 32 Gbps, PCIe Gen5 speed, with ASIC implementation. Because CXL and PCIe share the same physical layer, this memory device conveniently plugs into existing PCIe slots.

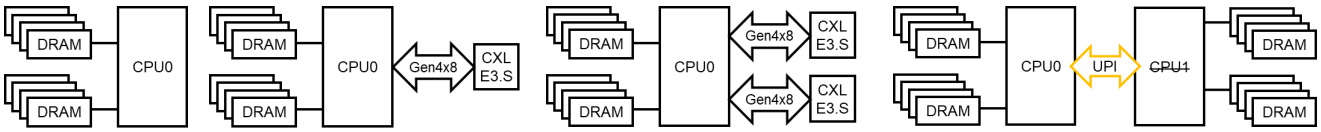
Our CXL device is recognized as a memory-only NUMA node or a DAX device. When CXL memory appears on the system memory map along with the host DRAMs, CPUs can directly load/store from and to the device memory through the host CXL interface without ever touching the host memory. To highlight CXL.mem protocol benefit of low latency, the translation logic between CXL protocol to DRAM media is kept to a minimum. CXL physical and link layers perform configuration and link negotiation with the PCIe root complex. CXL protocol layer unpacks CXL flits into command, address, and data fields for the internal data path. In this prototype, host physical addresses are directly mapped onto the device memory space, removing the need for translation. CXL read and write commands are handled by CXL protocol engine and the memory controller performs 64B read and write transactions to DRAM. Because the target throughput is 16GB/s, a single DDR channel is sufficient to match the performance. However, to get the best throughput, multiple outstanding transactions are required to mitigate the latency to and from the DDR interface. To maximize device memory bandwidth, CXL.mem read and write are arbitrated fairly in the first-come first-server order. Writes are completed when data is written into DDR memory. Responses for read and write are returned to the host in the same order of their completion.

4 PERFORMANCE EVALUATION

4.1 System Configuration

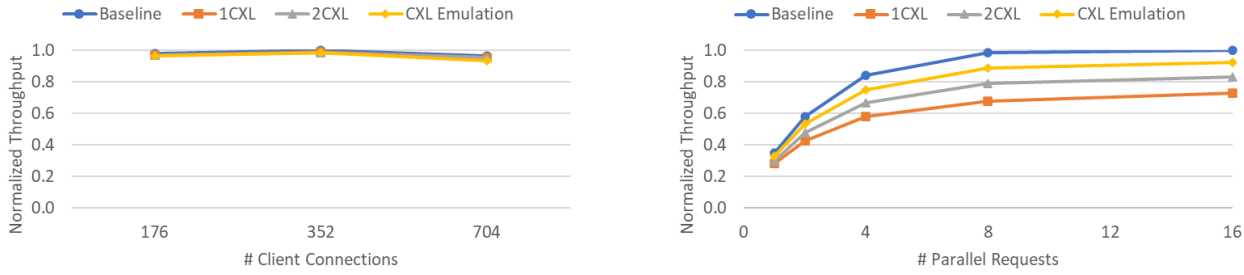
The evaluation was done on Intel's Sapphire Rapids (SPR) customer reference board with C0 stepping CPUs. For all configurations except CXL emulation in the single-node test and the scale-out configuration, we enable one socket per node. For CXL emulation and the scale-up configurations, we enable two sockets. Each socket has 512 GB DDR5 4800 MHz memory, one 64 GB DIMM per channel. CXL memory extensions are set up as DAX devices because the current release of our IMDBMS does not support the memory-only NUMA node yet. Thus, the memory space is recognized with persistent memory features [9] and the main storage is moved to CXL memory. The persistent memory features do not add any overhead when reading the main storage because there is no additional instruction required.

As the IMDBMS used for our experiments accepts only one DAX path per socket due to the limitation of the test version, we stripe multiple CXL memory devices using the device mapper [6] to see the impact of increased bandwidth beyond the current 16GB/s limit. We use TPC-C with 100 warehouses for OLTP workloads, increasing



(a) Baseline (b) 1CXL (c) 2CXL (d) CXL Emulation

Figure 2: System configuration for single-node test



(a) TPC-C (b) TPC-DS

Figure 3: Benchmark performance in single-node test

the number of client connections to maximize the performance. The number of client connections is set to 176, 352, and 704, which are respectively 4, 8, and 16 times of 44 physical cores of a single CPU in the system. We also use TPC-DS with SF=100 for OLAP workloads, increasing the number of parallel requests up to 32 in the client to see the performance scalability.

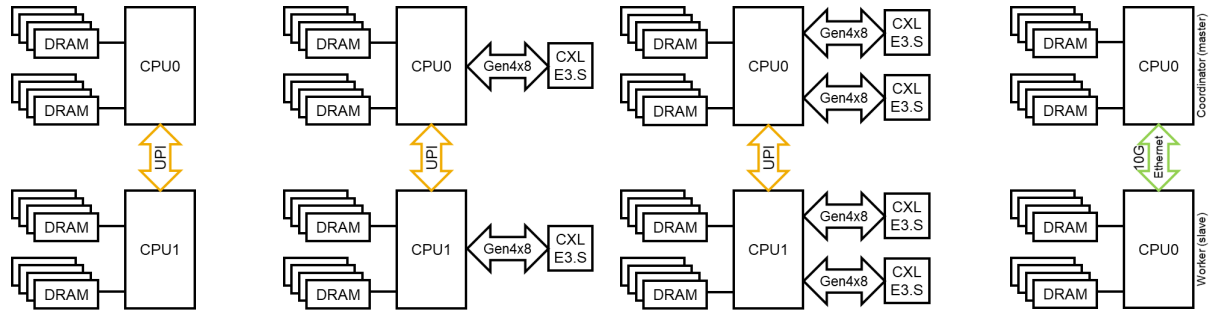
4.2 Evaluation of CXL Memory Devices

Single-node Test. We test a single-socket machine to evaluate the performance of CXL memory expansion. In this experiment, we have 4 configurations as shown in Fig. 2: (1) the baseline without CXL memory expansion, (2) with 1 CXL device, (3) with 2 CXL devices striped, and (4) CXL emulation in SPR by setting up the main storage in the memory of the remote socket as a DAX device after CPU affinity is set to CPU0 only, assuming that the access latency to the remote memory through UPI is similar to the future CXL memory expansion. The baseline allocates the main storage in the host DRAM, while the other configurations with CXL memory devices have it in the CXL memory expansion area. CXL emulation allocates the main storage in the remote memory.

Fig. 3 shows the normalized throughput to the maximum among all configurations. TPC-C has nearly no performance difference between the baseline and the other CXL configurations. As mentioned in Section 2, the latency of sequential accesses to the main storage is completely hidden by the prefetching scheme. Profile results using Intel® VTune™ Profiler [7] show low memory bandwidth bound in TPC-C. Thus, CXL memory expansion has no performance degradation in OLTP workloads. However, the average throughput degradation in TPC-DS is 27% in 1CXL, 18% in 2CXL, and 8% in CXL emulation. Larger performance degradation is mainly caused by the limited bandwidth of the current CXL prototype with PCIe Gen4x8. However, we expect that the degradation in TPC-DS would be less than 8% in the future CXL product as CXL memory bandwidth is increased with PCIe Gen5x16 (64 GB/s), which is more than UPI connections in CXL emulation.

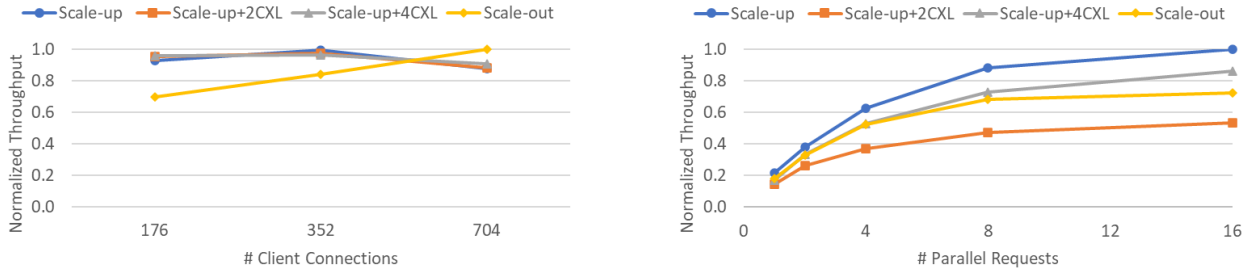
Comparison between scale-up and scale-out. To study further benefits of CXL memory expansion, we compare the performance of a scale-up with 2 CPUs and a 2-node scale-out system as shown in Fig. 4. First, the scale-up baseline has no CXL memory expansion. Second, scale-up+2CXL has two CXL memory devices, one per socket. Third, scale-up+4CXL has four CXL memory devices, two per socket with striping to increase the bandwidth. The scale-up baseline has the main storage in the host DRAM, while the scale-up with the CXL memory has it in the CXL memory. We use the default NUMA-aware location to achieve balanced memory usage across NUMA nodes in the scale-up configurations. In the scale-out, we prepare two nodes enabled with 1 CPU in each node to make a fair comparison. Then, we connect them with 10G Ethernet. We use hash partitioning on the first columns of the primary keys in all tables, which are used in all the join conditions.

As shown in Fig. 5, TPC-C has no significant performance difference between the scale-up baseline and the scale-up CXL configurations because of low memory bandwidth bound. Comparing the performance between the scale-up and the scale-out, the scale-up configurations outperform the scale-out before 704 connections (16 * 44 physical cores). We observe that the performance of the scale-up decreases for 704 connections due to the overhead caused by too many client connections in a single machine. The average throughput degradation for TPC-DS compared to the scale-up baseline is 39% in scale-up+2CXL and 16% in scale-up+4CXL due to the bandwidth limitation of the prototype. However, scale-up+4CXL shows slightly better throughput than the scale-out. Once CXL memory bandwidth is increased with PCIe Gen5, the scale-up with CXL memory is expected to have much better performance than



(a) Scale-up (b) Scale-up+2CXL (c) Scale-up+4CXL (d) Scale-out

Figure 4: System configuration for scale-up and scale-out



(a) TPC-C (b) TPC-DS

Figure 5: Performance comparison between scale-up and scale-out

the scale-out. Therefore, CXL memory expansion is a good solution to increasing the memory capacity with lower TCO, if the system provides sufficient computing resources.

5 RELATED WORK

Providing a flexible and integrated memory view larger than the physical memory is one of the important topics in IMDBMSs to overcome the capacity limitation. Guz et. al. [11] proposes NVMe-SSD disaggregation using NVMf (NVMe-over-fabrics) [3]. Koh et. al. [12] introduces a disaggregated memory system integrated with the hypervisor for cloud computing. Adding the disaggregated memory support to the memory management in the KVM hypervisor minimizes the overhead of remote direct memory accesses (RDMA). Korolija et. al. [13] proposes Farview, a disaggregated and network-attached memory using an FPGA-based smart NIC. Taranov et. al. [17] proposes CoRM, an RDMA-accelerated shared memory system.

As more flexible solutions for the heterogeneous and disaggregated computing platform, several technical standards for cache coherent interconnects have been devised. CCIX (cache coherent interconnect for accelerators) [4] is a protocol to enable coherent interconnects widely used in ARM-based System-on-Chips, while OpenCAPI [1] is an open standard for Cache Accelerator Processor Interface developed for IBM Power CPUs. Gen-Z [2] was an open system interconnect to provide cache coherent memory accesses, and now it is merged to CXL. NVLink [8] is also a cache coherent interconnect mainly for NVidia GPUs. It is also supported in IBM Power CPUs. Lutz et. al. [14] shows that fast interconnects like NVLink 2.0 can overcome the limits of the current GPUs, such as on-board memory capacity and interconnect bandwidth, thus resulting in better performance in CPU-GPU hash joins with a larger data size than the amount of GPU memory.

6 CONCLUSION

This work proposes a flexible CXL-based memory expansion with potentially lower TCO in an IMDBMS as one of the significant use cases of CXL memory. The evaluation results using common database benchmark tests proved the feasibility of CXL memory expansion in an IMDBMS. OLTP workloads have nearly no throughput degradation with CXL memory devices. Even though OLAP workloads have a certain amount of throughput degradation, its performance on the real CXL product can be dramatically improved once CXL devices are operating at PCIe Gen5 speed. Furthermore, considering that the current experiments were done using the pre-production SPR (C0 stepping) CPU and slow DDR3 DIMMs in CXL memory expansion, more performance improvement is anticipated with the mass-production SPR and CXL memory.

REFERENCES

[1] 2014. *OpenCAPI Consortium.* https://opencapi.org/
[2] 2016. *Gen-Z Consortium.* https://genzconsortium.org/
[3] 2016. *NVM Express over Fabric 1.0.* https://nvmexpress.org/
[4] 2017. *CCIX Consortium.* https://www.ccixconsortium.com/

[5] 2019. *Compute Express Link*. https://www.computeexpresslink.org/
[6] 2021. *Device Mapper*. https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/index.html
[7] 2021. *Intel® VTune™ Profiler*. https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html
[8] 2022. *NVLink*. https://www.nvidia.com/en-us/data-center/nvlink/
[9] Mihnea Andrei, Christian Lemke, Günter Radestock, Robert Schulze, Carsten Thiel, Rolando Blanco, Akanksha Meghlan, Muhammad Sharique, Sebastian Seifert, Surendra Vishnoi, et al. 2017. SAP HANA adoption of non-volatile memory. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1754–1765.
[10] Franz Färber, Norman May, Wolfgang Lehner, Philipp Große, Ingo Müller, Hannes Rauhe, and Jonathan Dees. 2012. The SAP HANA Database–An Architecture Overview. *IEEE Data Eng. Bull.* 35, 1 (2012), 28–33.
[11] Zvika Guz, Harry Li, Anahita Shayesteh, and Vijay Balakrishnan. 2017. NVMe-over-fabrics performance characterization and the path to low-overhead flash disaggregation. In *Proceedings of the 10th ACM International Systems and Storage Conference*. 1–9.
[12] Kwangwon Koh, Kangho Kim, Seunghyub Jeon, and Jaehyuk Huh. 2018. Disaggregated cloud memory with elastic block management. *IEEE Trans. Comput.* 68, 1 (2018), 39–52.
[13] Dario Korolija, Dimitrios Koutsoukos, Kimberly Keeton, Konstantin Taranov, Dejan Milojičić, and Gustavo Alonso. 2021. Farview: Disaggregated memory with operator off-loading for database engines. *arXiv preprint arXiv:2106.07102* (2021).
[14] Clemens Lutz, Sebastian Breß, Steffen Zeuch, Tilmann Rabl, and Volker Markl. 2020. Pump up the volume: Processing large data on GPUs with fast interconnects. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1633–1649.
[15] Hasso Plattner. 2014. The impact of columnar in-memory databases on enterprise systems: implications of eliminating transaction-maintained aggregates. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1722–1729.
[16] Iraklis Psaroudakis, Florian Wolf, Norman May, Thomas Neumann, Alexander Böhm, Anastasia Ailamaki, and Kai-Uwe Sattler. 2014. Scaling up mixed workloads: a battle of data freshness, flexibility, and scheduling. In *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 97–112.
[17] Konstantin Taranov, Salvatore Di Girolamo, and Torsten Hoefler. 2021. CoRM: Compactable Remote Memory over RDMA. In *Proceedings of the 2021 International Conference on Management of Data*. 1811–1824.