Expanding System Memory Boundaries Through CXL-Enabled Device - *A Case Study of CXL Memory Expander for In-memory Database Management Systems*

Andrew Chang, Jaemin Jung,
Vincent Pham, Shuyu Lyu,
Ramdas Kachare, Yang Ki

Memory Solution Lab,
Samsung Semiconductor Inc.,
San Jose, California, USA

Minseon Ahn, DongHun Lee,
Jungmin Kim,
SAP Labs, Korea

Oliver Rebholz, SAP SE, Walldorf,
Baden-Württemberg, Germany

1. Introduction

With the growth of data volumes in data analytics and machine learning applications, the importance of a balanced compute and memory system performance is becoming increasingly critical. With more powerful processors, the memory technology has evolved both in bandwidth and capacity through designing with higher IO speed as well as 3D stacked HBM cube with wider internal data path to relieve the access penalty across the memory-wall. However as Dennard scaling has reached its limit, other solutions exploit new memory media, such as 3D Xpoint and NVDIMM-P, to expand system memory on existing DIMM slots. Even though many of these efforts have failed to become mainstream, they highlighted the urgent need to increase host system memory and a new interface to address a new class of memory that is coherent with system memory but also has lower latency and better read/write performance over traditional SSD block devices.

To serve these requirements, several heterogeneous and disaggregated interfaces have been proposed, such as Gen-Z, CCIX, OpenCAPI, and the most recent addition, CXL (Compute Express Link). CXL (Compute Express Link) is a new interconnect standard for device connectivity that is open and endorsed by CPU and memory vendors across the industry. It is a cache coherent interface using the PCIe technology and enables memory expansion and heterogeneous memory for disaggregated computing platforms. CXL operates across the standard PCIe 5.0 physical layer and consists of three distinct protocols: CXL.io, CXL.cache, and CXL.memory.

CXL.io is responsible for discovery, configuration, register access, and interrupt handling. CXL.cache allows device access to processor memory and CXL.memory enables processor access to device attached memory. These three protocols are dynamically interleaved on a fixed flit format at data link layer to ensure the lowest latency access. There are three different types of CXL devices that vary based on the use of these protocols.

Type 1 CXL devices are those without host-managed device memory, such as NICs, which use CXL.io and CXL.cache. Type 2 devices are CXL devices with host-managed device memory, such as GPUs or external computing units, which utilize all three protocols. Finally, Type 3 CXL devices only have host-managed device memory and only use CXL.memory. This type of memory device is mapped to host physical address space and host accesses data using load/store semantics the same way as it accesses to host DRAM. This is a big advantage over PCIe 4KB page transfer when addressing 64B cache line granularity.

Within the Type 3 device, the memory media access protocol is decoupled from the CXL interface. This opens up total cost of ownership (TCO) optimization opportunities for device vendors. For example, this device can use DDR5 memory for best performance or solid-state driver (SSD) for capacity and power efficiency. As long as it supports CXL.io and CXL.mem, this Type 3 device can seamlessly work in the host system memory coherence domain as a memory expansion.

At system level, having a PCIe ecosystem means PCIe connectors, cable and cooling designs can be readily leveraged. This also includes the flexibility to choose from a wide range of form factors for different device and system configurations.

As a case study, we present an evaluation of the Samsung CXL Memory Expander device on real SAP HANA application. This is the result of a research collaboration between Samsung Memory Solution Lab and SAP lab Korea. It helps us understand the benefit of system memory expansion for IMDBMS. The remainder of this paper will be organized as the following: 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 for IMDBMS

One of the fast growing applications in today's data economy is databases. It is not only the fundamental infrastructure of our daily online transactions and queries, but also a platform to turn data into intelligence through data analytics and machine learning. In order to achieve lowest latency, the powerful in-memory database stores and operates from host DRAM. Therefore it is particularly sensitive to system memory capacity ceiling because the entire database will come to a halt when it runs out of memory. System vendors have to overprovision DRAM capacity for the "worst case scenario" to keep up with future demands for on premise deployment. This increases total cost of ownership (TCO) for IMDBMS servers and still cannot guarantee successful operation. Emerging technologies like NVMe and RDMA provide a flexible and integrated memory view larger than the physical memory. But this usually incurs longer latency and most importantly needs application level changes. To overcome these limitations on physical memory expansion, heterogeneous and disaggregated computing platforms, such as Gen-Z [2], CCIX [4], OpenCAPI [1], and most recently CXL (Compute Express Link) [5], have emerged. CXL, in particular, has garnered wide industry adoption and is the most promising solution for mitigating memory overprovisioning issues. These new interfaces provide a more flexible and cost effective way for physical memory expansion in IMDBMS without incurring significant performance penalties.

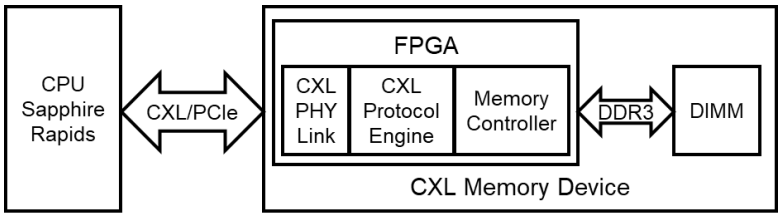


Figure 1: CXL Prototype Diagram

Hybrid transactional and analytical processing (HTAP) is widely supported by in-memory database management systems (IMDBMSs) [16]. These two types of workload have different memory access characteristics. Transactional processing tends to have small, random patterns while analytical processing often sequential and read dominant. In this study, SAP HANA in-memory database platform [10] is used as a base platform. It leverages the columnar storage [15], which stores each column in the read-optimized main storage while maintaining separate delta storage for optimized writes. Furthermore, a portion of memory is allocated for the operational data to hold intermediate results while processing a query.

To enable memory expansion in our IMDBMS using CXL memory devices, there are two options available. In the first option, the CXL device's additional memory space could be uniformly integrated with the host memory space, allowing for both operational memory and main storage to be allocated in CXL memory. However, this may degrade overall performance, especially transactional processing, due to the longer access latency compared to host DRAMs when accessing operational memory randomly. The second option involves using the CXL memory device only for the main storage while delta storage and the operational data are stored in the host DRAMs. Similar to this 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 study, we use the second option in our IMDBMS to take advantage of the prefetch.

3. CXL Memory Expander

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 128G DRAM as media and supporting a theoretical bandwidth of 16GB/s with PCIe Gen4x8 as the bottleneck. As shown in Fig. 1, the proto-type consists of a custom FPGA board and a single-channel-based DDR3 DIMM DIMM. 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 protocols to DRAM media is kept to a minimum. CXL physical and link layers

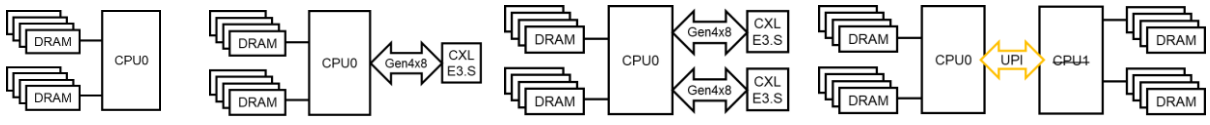
perform configuration and link negotiation with the PCIe root complex. CXL protocol layer unpacks CXL flits into command, address, and data fields for the internal data path. In this prototype, host physical addresses are directly mapped onto the device memory space, removing the need for translation. 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 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.



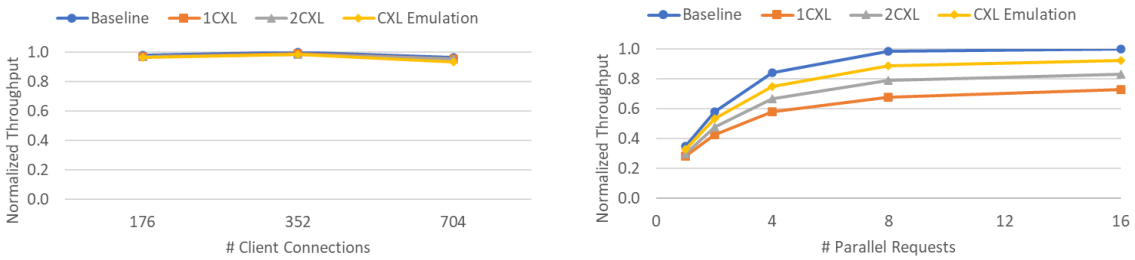
(a) Baseline (b) 1 CXL (c) 2 CXL (d) CXL Emulation

Figure 2: System Configuration for a Single-Node Test

4.2 Evaluation of CXL memory Device

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

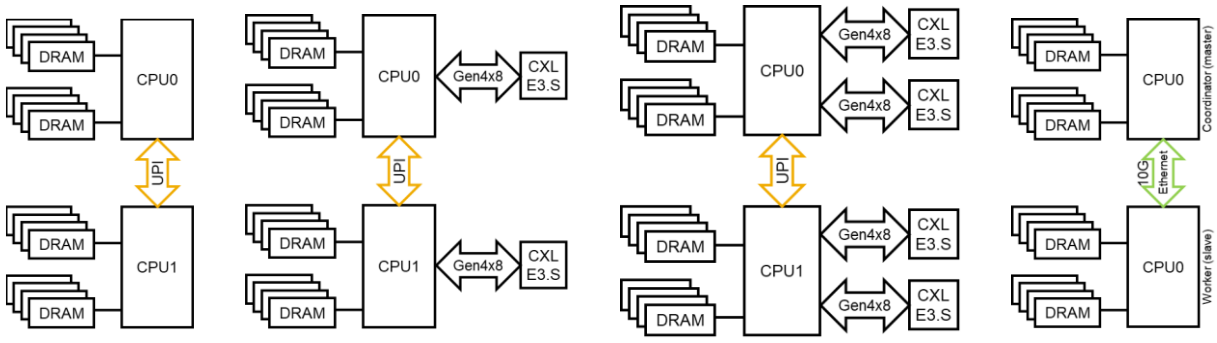
without CXL memory expansion, (2) with 1 CXL device, (3) with 2 CXL devices striped, and (4) CXL emulation in SPR by setting up the main storage in the memory of the remote socket as a DAX device after CPU affinity is set to CPU0 only, assuming that the access latency to the remote memory through UPI 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.



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

Figure 3: Benchmark Performance for a Single-Node Test

Fig. 3 shows the normalized throughput to the maximum among all configurations. TPC-C has nearly no performance difference be- tween 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® VTuneTM 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.

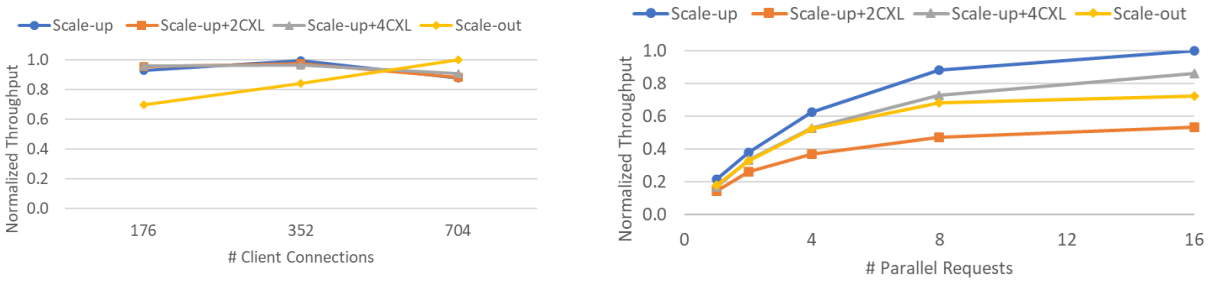


(a) Scale-up (b) Scale-up+2CXL (c) Scale-up+4CXL (d) Scale-out

Figure 4: System Configuration for Scale-up and Scale-out

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.



(a) TPC-C

(b) TPC-DS

Figure 5: Performance Comparison between Scale-up and Scale-out

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 base- line 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 that 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

Overcoming the capacity limitations by providing a flexible and integrated memory view larger than the physical memory is a crucial topic in IMDBMSs. To address this challenge, several solutions have been proposed in the literature. Guz et. al. [11] propose NVMe-SSD disaggregation using NVMf (NVMe-over-fabrics) [3]. Koh et. al. [12] introduce a disaggregated memory system integrated with the hypervisor for cloud computing, where the memory management in the KVM hypervisor is enhanced to reduce the overhead of remote direct memory accesses (RDMA). Korolija et. al. [13] propose Farview, a disaggregated and

network-attached memory using an FPGA-based smart NIC. Taranov et. al. [17] propose CoRM, an RDMA-accelerated shared memory system.

Several technical standards for cache coherent interconnects have been developed to provide more flexible solutions for the heterogeneous and disaggregated computing platform. 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

We studied a case of using Samsung memory expander to lower TCO in an IMDBMS without significant loss of overall performance. The current FPGA implementation carries longer latency because it only operates at PCIe Gen4 speed and uses DDR3 as back media. Nevertheless, the evaluation results using common database benchmark tests proved the feasibility of CXL memory expansion in an IMDBMS. OLTP workloads have nearly no through-put 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.

Being able to use the insights of data processing pipeline and to separate latency sensitive and throughput oriented workload is one important learning from this PoC. CXL memory is a new class of memory that is further away from the host complex and inherently will have longer latency. But if the application can take advantage of throughput at fine granularity and its flexible capacity, it can benefit overall performance and TCO optimization.

Decoupling the media management to CXL protocol, enables more opportunity in next generation memory device solutions. As the leader in both DRAM and NAND Flash SSD, Samsung is committed to develop the best solutions to help our customers tackling the memory expansion challenges.

7. 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
14] Minseon Ahn et al. “Enabling CXL Memory Expansion for In-Memory Database Management Systems”, DaMoN’22, June 13, 2022, Philadelphia, PA, USA