TECHNICAL NOTE

The GigaIO™ FabreX™ Network

Technical Note

Memory Pooling Overview

CONTENTS

TABLE OF FIGURES

This document is for informational purposes only and may contain typographical errors and technical inaccuracies. The content is provided as is, without express or implied warranties of any kind.

February 2020| Rev 1.0

Introduction

Disaggregating and pooling the various resources found in servers today is a key benefit of GigaIO's FabreX network. We have discussed GPU, FPGA, CPU and storage in prior Technical Notes; this one will focus on Memory. Memory can be either volatile or non-volatile in nature and can be attached to the network as a separate entity (a memory pool) or can be some portion of the memories in the individual servers attached to the network (a virtual memory pool). Regardless of the type, these memory pools can then be accessed by any compute element attached to the network.

Memory pools may be used either as 'memory pools' or as 'shared memory'. 'Shared memory' is considered one pool of memory that multiple compute units use and can be an ideal environment for multiple agents to share data via this shared pool of memory. Coherency of this memory is not provided by FabreX and must be addressed through the server hosting the memory. 'Memory pools' on the other hand, are a large pool of memory where portions of the memory are allocated to individual compute elements or virtual machines (VMs) across the network and are dedicated to that particular compute element for the life of the workload. Compute elements or VMs can release memory back into the general unallocated pool at the completion of their job to be re-allocated again.

Alternately, FabreX's unique architecture and its technology allows for transforming a dedicated segment of system memory of the servers attached to the network to serve as a 'virtual' shared memory resource for memory pooling applications (either shared memory or memory pools). This is a natural outcome of the ability of every server attached to FabreX having visibility into every other server's system memory. We will not discuss this approach in detail in this Note, as it is expected the performance of these virtual memory pools will be slower than those implemented as a separate box as there are fewer layers to arbitrate.

Memory Resource Definition

Memory devices which can be used in memory pools within FabreX cover the full range of memory technology, including but not limited to, the following:

a) Static Memory

b) Dynamic Memory (DDR3, DDR4, etc.)

c) NVDIMM

d) Persistent Memory technologies including 3D-XPoint, 3DXP, MRAM, etc.

e) Non-volatile Flash Memory including NAND and related technologies

As the latency of the memory device decreases, it is obviously important for the transport latency of the network to be as low as possible in order to minimize the latency delays and thereby maximize the throughput of the entire system. This is because a compute agent accessing the memory pool will essentially enter wait states executing idle cycles for the duration of the latency time in order for the data to be valid.

Flash memory used by storage appliances are typically not used for memory pooling applications as these devices typically have a device latency in the 10s of microseconds and are too slow for memory applications. As a result, Block and File storage protocols have evolved to make this memory ideal for Storage applications. FabreX also handles storage devices extremely well and these applications are covered in the '*New Frontiers in NVMe-oF*' white paper. But as Flash technology gets faster and as the BAR size for addressing this memory increases, its usefulness as part of a memory pool increases.

Memory Pooling

Figure 1 is one example of compute resources accessing a memory pool, where the memory pool resides in either a 'memory server' (orange or pink boxes on the bottom side of Figure below) or in a Resource Box (magenta or green boxes). Memory accesses consist of Load and Store memory semantics applied to the memory resource. These memory semantics originate from the processor of a given server and flow through the processor's I/O bus. These I/O paths exclusively use PCI Express (PCIe) technology and are supported by all computing elements.

In the case of FabreX, PCIe also constitutes the data transport of the network. As a result, attached memory pools act like memory resources resident in the individual backplanes of all the respective servers attached to FabreX. Further, FabreX supports atomic memory operations that are invoked and originate with the compute elements. Memory pool resources supporting this feature can be used to perform 'Read Modify Write' operations by the compute elements which are commonly used to implement algorithms that provide some levels of memory coherency. For configurations that use the memory server, full coherency could be implemented within the server, albeit with a performance penalty.

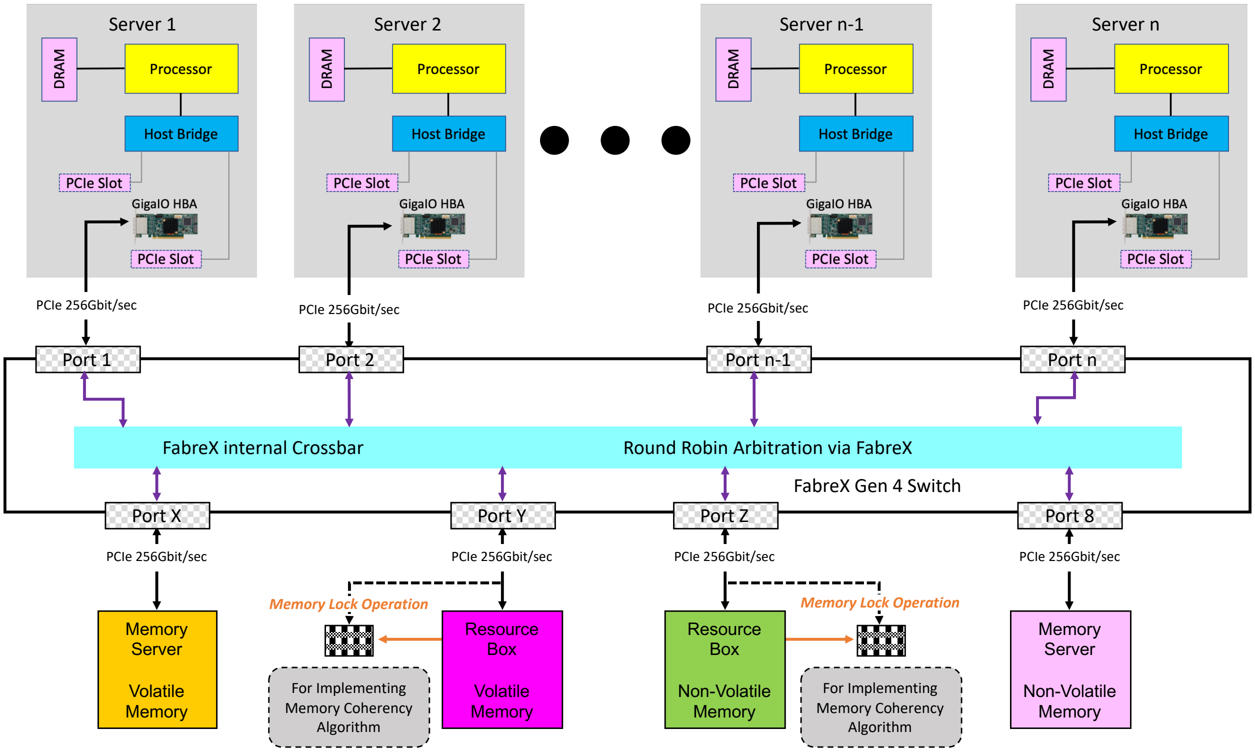


Figure 1 –Compute Resource Access to Memory Pools Example

Memory Pooling Between Disaggregated Elements

The function of memory pooling implies and assumes a level of autonomy of the memory resource and consequently, the pool is not necessarily confined to access by processors but can be accessed by compute elements of all types including GPUs, FPGAs and ASICs and even including NVMe storage.

Figure 2 depicts a scenario where the memory pool is used as a holding container for interim hot data waiting to be processed by the next processing element. This method allows the data to be accessed by numerous processing resources. This scheme can very effectively be used in lieu of traditional systolic processing by having the processed data repeatedly stored at the same location accessible by all processing elements in the chain.

The figure shows GPU data being stored in the memory pool that acts as in-flight data that needs to be further processed by other processing elements. As with accesses by the GPUs, storage data can be transferred from NVMe drives to the memory pool. This mechanism allows for ping-pong of hot and cold data as required and demanded by the applications.

One point to note in this example is all the data transfers between the GPUs in the JBOG (or JBOG Server) and the NVME drives in the JBOF (or JBOF Server) take place with DMA resources resident in these devices. Alternately, since FabreX has integrated DMA engines built-in at every port, it allows for the use of these DMA resources to transfer data between the memory pool and any other memory attached to it.

One additional advantage of transferring data with DMA between these resources is that it greatly reduces the load of the server processors and allows users to maximize their utility in demanding environments.

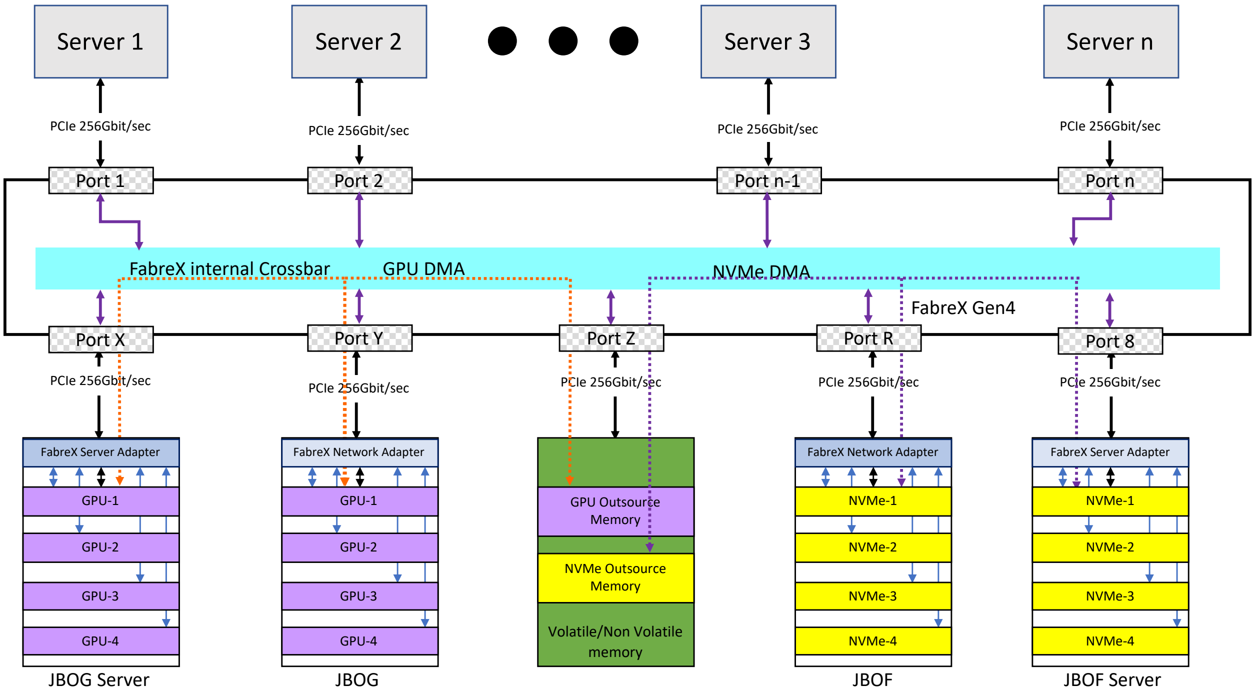
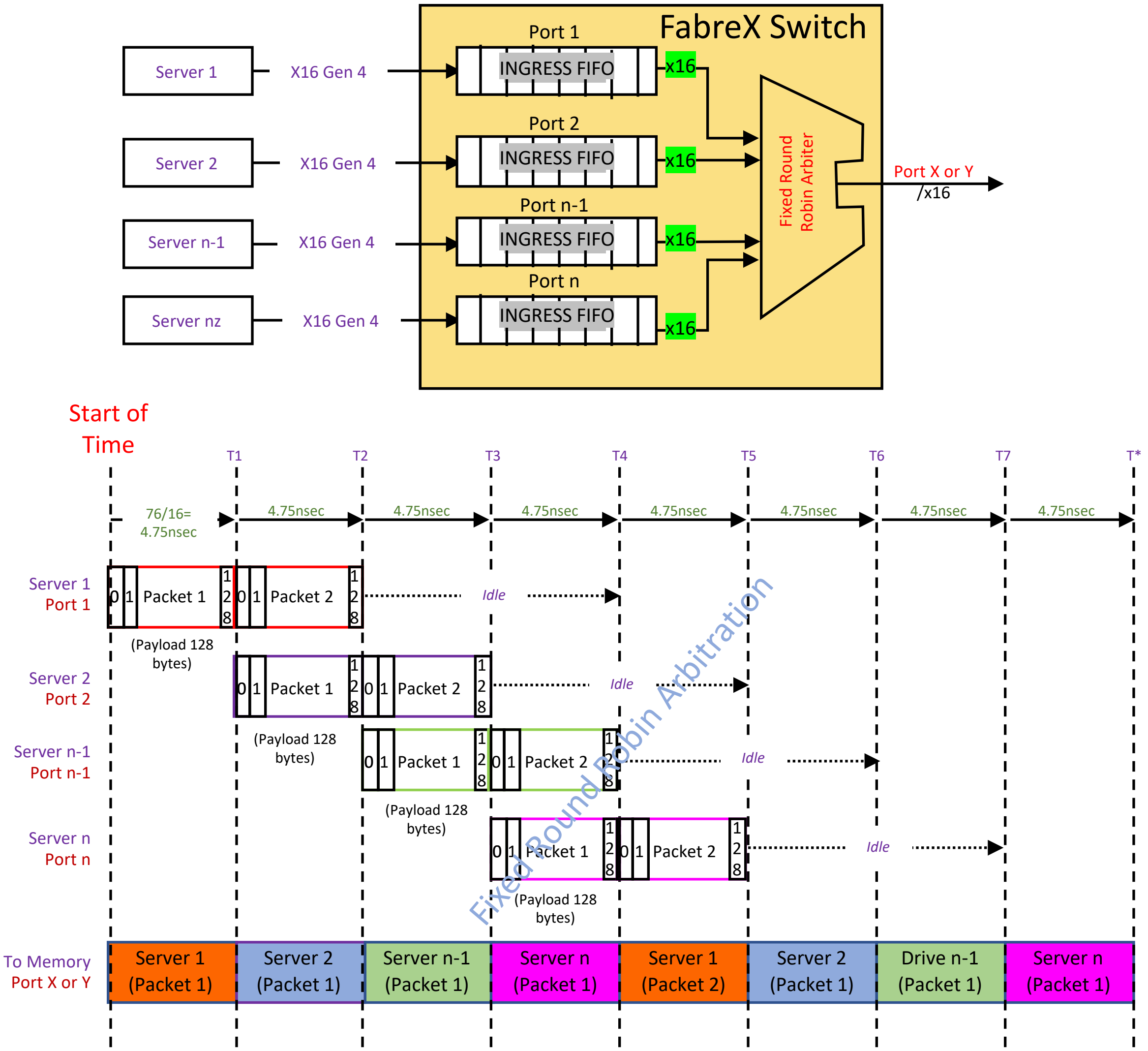


Figure 2 – Memory Pools as Containers

Performance of Memory Pooling using FabreX

Figure 3 shows the performance characteristics when using memory pooling with FabreX. The example shows the impact of 4 servers simultaneously accessing the memory pool at the same instant of time. This example assumes a fixed round robin arbitration scheme. However, FabreX's QoS capabilities allows for several arbitration schemes, including fixed round robin, weighted round robin, and 'time slice' where each port can be given a predefined unit of time dedicated to the transfer of data.

This figure also shows the effective bandwidth of the access paths of every server to the memory pool in the case of simultaneous accesses. This number will be lower in real world application since it needs to include the latency involved with the access time of the memory devices in this context.



Bandwidth for Server 1, 2, (n-1) and n for send and/or receive 128 Bytes every (4x4.75) 19 nsec which is equivalent to = 19 nsec/128 = 0.128 nsec = 6.74 GBytes/sec

Figure 3 – Performance Characteristics

Summary

FabreX is the only technology available that allows for communication between all resources attached to the fabric to take place at PCIe native speeds. The latency incurred in these communication paths is the lowest in the industry as there is no communication protocol transformation involved with the I/O signaling emanating from the RCP (Root Complex Processor) and the communication path to I/O resources including memory.

FabreX is the only communication technology in the industry today that allows for a true memory atomic operations of Load and Store between Servers and I/O devices, including memory pools, across the fabric.

Since data transfers happen at PCIe native speeds, the bandwidth of the communication path of FabreX is identical to the bandwidth of the I/O signaling path emanating out of the processor.

About GigaIO

GigaIO was established in 2016 by networking industry veterans with decades of domain expertise in communications, data centers, high-performance computing, open source, and infrastructure management. The company is headquartered in Carlsbad, CA, and home to more than 30 staff members, most of whom are engineers with advanced degrees and more than 15 years of industry experience. GigaIO develops innovative, high-performance interconnect network for computing clusters, with the objective of accelerating large-scale workloads on-demand, using industry-standard technology. GigaIO FabreX eliminates conversion layers, maximizes throughput, and enables data centers to run at full efficiency and obtain outstanding performance. The company's patented network technology facilitates development of broad and deep network architecture. GigaIO's extreme connectivity for high-end computing delivers optimized resource utilization and reduced total cost of ownership. For more information, contact the GigaIO team at info@gigaio.com or visit www.gigaio.com.