Gen-Z: emerging technology for memory intensive applications

Pekon Gupta
Arthur Sainio
Agenda

- Need for new protocols
- Comparison between Protocols
  - Serial attached: OpenCAPI v/s CXL v/s CCIX
  - Fabric attached: NVMe-oF v/s RoCEv2 v/s Gen-Z
- Gen-Z topology
- Exploring Gen-Z
- Summary
- References
**Why do we need new interconnect standards?**

**Data is growing exponentially**
Modern data applications require large volumes of data to be processed in real-time. Each individual uses multiple IOT devices, which continuously generate data. Everyone is using real-time streaming and real-time analytics.

**Demand for heterogeneous computing**
New forms of compute (GPU, FPGA, SoC, TPU) and accelerators require continuous feeding of data. Reduce data copy, and process the data near the location where it is generated.

**But, amount of memory available per core is flat or decreasing**
PCB real-estate is saturated, we cannot have more DIMM slots. Memory bandwidth per core continues to decrease. Cost of PCB manufacturing is continuously decreasing.

**To solve above challenges, industry needs architecture to support**
Persistent memory and storage class memory tiers need to be integrated into servers. Rack-scale composability, which requires a **high bandwidth, low latency fabric**. Seamlessly plug into existing ecosystems **without requiring OS changes**.

---

**Explosive Growth of Data**
- More than 37% of total data generated in 2020 (40 ZB) will have Big Data value

**Need Answers ... FAST!**
- Business demanding real-time insight
- Increasing amounts of data to be analyzed
# Memory Tiers: How far is the memory from CPU?

<table>
<thead>
<tr>
<th>Latencies</th>
<th>Location from CPU</th>
<th>Capacities as of today</th>
<th>Examples</th>
</tr>
</thead>
</table>
| < 10 ns   | L1/L2 cache: on chip  
* “dedicated” to a given CPU Core | 100s of KB to 10s of MB  |
| < 50 ns   | L3 cache: on-chip  
* “shared” among various CPU and hardware engines. | 100s MB to few Giga Bytes |
| < 100 ns  | Direct attached (Parallel Bus)  
* outside chip, shared among CPU sockets  
* DDR-4, NVDIMM-N, NVDIMM-P | 100s of Giga Bytes |
| < 500 ns  | Serial attached  
* inside chassis, may or may not be directly connected to board  
* CCIX, CXL, OpenCAPI (OMI) | 100s of Giga Bytes to Tera Bytes |
| < 1000 ns (1 us) | Network attached  
* Outside chassis, distributed across racks  
* Gen-Z, RDMA | Tera Bytes to few Peta Bytes |
## Memory Tiers: How far is the memory from CPU?

<table>
<thead>
<tr>
<th>Latencies</th>
<th>Location from CPU</th>
<th>Capacities as of today</th>
<th>Examples</th>
</tr>
</thead>
</table>
| < 10 ns   | L1/L2 cache: on chip  
* “dedicated” to a given CPU Core | 100s of KB to 10s of MB |          |
| < 50 ns   | L3 cache: on-chip  
* “shared” among various CPU and hardware engines. | 100s MB to few Giga Bytes |          |
| < 100 ns  | Direct attached (Parallel Bus)  
* outside chip, shared among CPU sockets  
* DDR-4, NVDIMM-N, NVDIMM-P | 100s of Giga Bytes |          |
| < 500 ns  | Serial attached  
* inside chassis, may or may not be directly connected to board  
* CCIX, CXL, OpenCAPI (OMI) | 100s of Giga Bytes to Tera Bytes |          |
| < 1000 ns (1 us) | Network attached  
* Outside chassis, distributed across racks  
* Gen-Z, RDMA | Tera Bytes to few Peta Bytes |          |
Difference between interconnects
## Direct attached memory: DDR4, DDR5

<table>
<thead>
<tr>
<th>Latencies</th>
<th>Location from CPU</th>
<th>Capacities as of today</th>
<th>Examples</th>
</tr>
</thead>
</table>
| < 10 ns    | L1/L2 cache: on chip  
* "dedicated" to a given CPU Core | 100s of KB to 10s of MB |          |
| < 50 ns    | L3 cache: on-chip  
* "shared" among various CPU and hardware engines. | 100s MB to few Giga Bytes |          |
| < 100 ns   | Direct attached (Parallel Bus)  
* outside chip, shared among CPU sockets  
* DDR-4, NVDIMM-N, NVDIMM-P | 100s of Giga Bytes |          |
| < 500 ns   | Serial attached  
* inside chassis, may or may not be directly connected to board  
* CCIX, CXL, OpenCAPI (OMI) | 100s of Giga Bytes to Tera Bytes |          |
| < 1000 ns (1 us) | Network attached  
* Outside chassis, distributed across racks  
* Gen-Z, RDMA | Tera Bytes to few Peta Bytes |          |
Challenges with parallel memory interfaces

“I can’t squeeze any more DDR slots”
- Board Designer

“These high speed buses are using more PCB layers and require special techniques”
- Manufacturer
# Serial attached memory: OpenCAPI, CCIX, and CXL

<table>
<thead>
<tr>
<th>Latencies</th>
<th>Location from CPU</th>
<th>Capacities as of today</th>
<th>Examples</th>
</tr>
</thead>
</table>
| < 10 ns   | L1/L2 cache: on chip  
* “dedicated” to a given CPU Core | 100s of KB to 10s of MB |          |
| < 50 ns   | L3 cache: on-chip  
* “shared” among various CPU and hardware engines. | 100s MB to few Giga Bytes |          |
| < 100 ns  | Direct attached (Parallel Bus)  
* outside chip, shared among CPU sockets  
* DDR-4, NVDIMM-N, NVDIMM-P | 100s of Giga Bytes |          |
| < 500 ns  | Serial attached  
* inside chassis, may or may not be directly connected to board  
* CCIX, CXL, OpenCAPI (OMI) | 100s of Giga Bytes to Tera Bytes |          |
| < 1000 ns (1 us) | Network attached  
* Outside chassis, distributed across racks  
* Gen-Z, RDMA | Tera Bytes to few Peta Bytes |          |
### Serial attached Interconnects: OpenCAPI, CCIX, and CXL

<table>
<thead>
<tr>
<th></th>
<th>OpenCAPI (Coherent Accelerator Processor Interface)</th>
<th>CCIX (Cache Coherent Interconnect for Accelerators)</th>
<th>CXL (Compute Express Link)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use-cases</strong></td>
<td>Memory expansion, Acceleration</td>
<td>Memory Expansion, Acceleration</td>
<td>Memory expansion, Acceleration</td>
</tr>
<tr>
<td><strong>Promoters</strong></td>
<td>IBM, AMD, Google, Mellanox (nVidia), Micron, Western Digital, Microchip</td>
<td>AMD, ARM, Huawei, Mellanox (nVidia), Qualcomm, Xilinx</td>
<td>Intel, AMD, ARM, IBM, Alibaba, Cisco, Dell, HPE, Google, Facebook, Western Digital, Microchip, Xilinx</td>
</tr>
<tr>
<td><strong>Compatibility</strong></td>
<td>New open standard serial interface called OMI.</td>
<td>Reuses PCIe-Gen4 PHY and Data link layers. Supports additional speeds up to 25GT/s.</td>
<td>Reuse PCIe-Gen5 PHY and add 528 bit packet boundaries called Flits.</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>In production</td>
<td>Available by end of 2020</td>
<td>2021-2022</td>
</tr>
<tr>
<td><strong>Host (CPU) Support</strong></td>
<td>IBM Power-8, Power-9 systems</td>
<td>ARM based servers</td>
<td>Intel</td>
</tr>
<tr>
<td></td>
<td>Ampere Computing, ARM Neoverse</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hardware Ecosystem</strong></td>
<td>ASIC based controller Controller IP available for FPGA based implementations</td>
<td>Primarily using FPGA with in-build acceleration engines. IP available for ASIC implementations</td>
<td>ASIC based Memory expansion devices. FPGA based acceleration devices</td>
</tr>
<tr>
<td>Latencies</td>
<td>Location from CPU</td>
<td>Capacities as of today</td>
<td>Examples</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------</td>
<td>------------------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| < 10 ns   | L1/L2 cache: on chip  
* “dedicated” to a given CPU Core | 100s of KB to 10s of MB | |
| < 50 ns   | L3 cache: on-chip  
* “shared” among various CPU and hardware engines. | 100s MB to few Giga Bytes | |
| < 100 ns  | Direct attached (Parallel Bus)  
* outside chip, shared among CPU sockets  
* DDR-4, NVDIMM-N, NVDIMM-P | 100s of Giga Bytes | |
| < 500 ns  | Serial attached  
* inside chassis, may or may not be directly connected to board  
* CCIX, CXL, OpenCAPI (OMI) | 100s of Giga Bytes to Tera Bytes | |
| < 1000 ns (1 us) | Network attached  
* Outside chassis, distributed across racks  
* Gen-Z, RDMA | Tera Bytes to few Peta Bytes | |
## Network attached: NVMe-oF, RDMA and Gen-Z

<table>
<thead>
<tr>
<th>NVMe-oF (Storage)</th>
<th>RDMA (Memory)</th>
<th>Gen-Z (Memory)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use-cases</strong></td>
<td>Store data on remote location. Storage (Cold Data)</td>
<td>Improve performance, and increase utilization by resource sharing</td>
</tr>
<tr>
<td><strong>How it works</strong></td>
<td>Extension of NVMe, blocked based data transfers.</td>
<td>Reuses PCIe and Ethernet PHY, but defines its own MAC and Transport layer packet structures.</td>
</tr>
<tr>
<td><strong>Hardware Requirements</strong></td>
<td><strong>None</strong></td>
<td><strong>Specialized hardware</strong></td>
</tr>
<tr>
<td></td>
<td>Reuse underlying network protocols like infiniband, TCP/IP, UDP, to share storage pool across the data-center.</td>
<td>Requires special NIC to support lossless transmission of packets over Layer-2 and Layer-3.</td>
</tr>
<tr>
<td><strong>Software Requirements</strong></td>
<td><strong>None</strong></td>
<td><strong>Specialized hardware</strong></td>
</tr>
<tr>
<td></td>
<td>Device drivers available in standard kernels.</td>
<td>Provide low latency access to disaggregated resource pools made of memory, storage or compute (GPU, FPGA)</td>
</tr>
</tbody>
</table>

**RoCE v2** *(RDMA over Converged Ethernet)*

- Applications running on different nodes, allocate memory buffers (address regions in local DRAM) which are directly accessible by remote agent.
- Reuses existing network protocols TCP/IP (iWrap), UDP (RoCEv2) for sending messages and commands.

**Hardware Requirements**

- None
- Reuse underlying network protocols like infiniband, TCP/IP, UDP, to share storage pool across the data-center.

**Software Requirements**

- None
- Device drivers available in standard kernels.

- Special middle ware required to expose RDMA Verbs to standard applications.

- Special middle ware required to expose RDMA Verbs to standard applications.
Network attached: NVMe-oF, RDMA and Gen-Z… (contd 2)

<table>
<thead>
<tr>
<th>NVMe-oF (Storage)</th>
<th>RDMA (Memory)</th>
<th>Gen-Z (Memory)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Availability</strong></td>
<td>In production</td>
<td>Prototyping</td>
</tr>
<tr>
<td><strong>Current</strong></td>
<td>ASIC based</td>
<td>ASIC and FPGA based</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>ASIC based</td>
<td>ASIC and FPGA based</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>High Latency &gt; 10us</td>
<td><strong>Lower latency for large sized data transfers, through RDMA.read(), RDMA.write() verb semantics</strong>&lt;br&gt;<strong>Latencies are substantial through RDMA.send() and RDMA.receive() verb semantics.</strong></td>
</tr>
<tr>
<td><strong>Use-cases</strong></td>
<td><strong>Cold Storage</strong>&lt;br&gt;• Low cost,&lt;br&gt;• More writes, less reads (Backup)</td>
<td><strong>Data sharing among applications</strong>&lt;br&gt;• Requires specialized hardware&lt;br&gt;• Requires change in application to incorporate RDMA Verbs</td>
</tr>
<tr>
<td><strong>Industry Support</strong></td>
<td>Multiple vendors (Industry proven)</td>
<td>Mellanox (now nVidia), Intel (Limited vendors)</td>
</tr>
</tbody>
</table>
# Interconnect standards at a glance

<table>
<thead>
<tr>
<th>Hardware View</th>
<th>Direct Attached</th>
<th>Serial (PCIe) Attached</th>
<th>Fabric Attached</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>&lt; 100ns</td>
<td>&lt; 500ns (efforts to bring latency comparable to DDR4)</td>
<td>&lt; 1000ns (efforts to bring latency up to 500ns)</td>
</tr>
<tr>
<td>Capacity</td>
<td>10s of GB – 100s of GB</td>
<td>100s of GB – few TB</td>
<td>100s of TB to few PB</td>
</tr>
<tr>
<td></td>
<td>• Too many pins &amp; traces on board. • Dependent on memory technology and configuration.</td>
<td>• Serial interfaces utilize less CPU pins • Take less space on board • Can be routed to larger distance on PCB • Decoupling interface protocol from memory technology enables reuse of existing board infrastructure like PHY, re-timers and connectors.</td>
<td>• Disaggregated architecture, where large pool of memory is outside the chassis on a different rack. So enough space and power • Efficient utilization and lower TCO. • Protocol redesigned to enable hardware assisted memory semantics (load/store) over fabric.</td>
</tr>
<tr>
<td>Software View</td>
<td>• Directly addressable. • No OS driver required</td>
<td>• Directly addressable. • No OS driver required. But requires BIOS update.</td>
<td>• Gen-Z memory server looks like large pool of “shared” memory to the Host, with in-built functions like vector atomic, scatter-gather, ...</td>
</tr>
</tbody>
</table>

**Take away**
- Performance and Performance
- Scalability and Density
- Scalability, Density and Cost ($$ per TB)
Gen-Z Topology
Gen-Z Fabric Topology

- Compute (Hosts)
  - CPU (Intel, AMD x86, ARM)
  - SoC
  - GPU
  - FPGA

- Devices
  - Persistent Memory, SCM
  - DDR DIMM
  - Storage SSD
  - Network Cards
  - Accelerators

Source: Modified from Gen-Z Consortium 2017
Gen-Z Fabric Topology

Gen-Z supports **disaggregation of compute, memory & storage components.**

Gen-Z fabric has two components:

**Compute Memory Controller**
- Initiates request as commands e.g. load-store, read-write, get-put
- Enforces ordering, reliability & path selection

**Media Controller**
- Executes commands & return responses
- **Abstracts type of memory technology** like DRAM, Persistent Memory, Flash
- **Offloads media specific functions** like wear-leveling, bad block management, memory refresh
- Provides acceleration & caching capabilities
- **Eliminates need for tight timed budget**

CXL and Gen-Z Working Together

SMART builds all the storage and compute modules for data center and HPC systems shown below.

The CPU supplies the CXL interface and the DDR channels.

Gen-Z provides a bridging device to attach the external or internal pools of resources.

DDR memory with CXL (PCIe) attachment and Gen-Z attachment.

Exploring Gen-Z
Gen-Z: Where to start? Development kit and products

**Gen-Z HHHL PCIe**
- 45GB/s data transfer rate
- 350 ~ 400ns latency
- 768GB of Flash
- 16GB of DRAM

**Gen-Z ZMM (3” Short EDSFF)**
- Supports 4 x Gen-Z requestor, responder configuration.

**User Programmable ZMM (3” Short EDSFF)**
- Dual Rank DDR-4 User Programmable memory module.

---

**Features & Benefits**
- Stable Gen-Z Linux kernel framework with device drivers and user-space applications
- Disaggregated “shared” pool of memory in Peta Bytes, sitting on different rack and with in-band sharing and management.
- User Programmable FPGA, enables customization of features for differentiation

**Applications**
- HPC
- Computational Memory: In-line compression and encryption functions while write/reading to memory.
Gen-Z µDK: Everything you need to explore Gen-Z

© Gen-Z Consortium 2020
Gen-Z µDK (Micro Development Kit)

The Gen-Z µDK is standalone development kit which allows companies to explore Gen-Z protocol without investing in buying new server or developing software.


© Gen-Z Consortium 2020

Target Use Cases

<table>
<thead>
<tr>
<th>Use cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen-Z OS Software development</td>
</tr>
<tr>
<td>• Develop Gen-Z Linux support and drivers.</td>
</tr>
<tr>
<td>Gen-Z Management Software</td>
</tr>
<tr>
<td>• Gen-Z In-band management software</td>
</tr>
<tr>
<td>• Fabric manager software*</td>
</tr>
<tr>
<td>Gen-Z Software Applications</td>
</tr>
<tr>
<td>• Load/Store Gen-Z memory access applications</td>
</tr>
<tr>
<td>• Memory-driven computing (MDC) applications</td>
</tr>
<tr>
<td>• Storage Pool OLAP and collective operation/map-reduce applications</td>
</tr>
</tbody>
</table>
Summary
Gen-Z – Composable Memory

Gen-Z Enabled Rack-scale

- DRAM/PM can be placed anywhere in a rack!
- Utilize compute, storage, or new form factors

Features

- Enables load/store memory semantics across the fabric
- Direct addressable memory. Does not require address translations or re-mapping.
- Implicit link error detection and recover mechanisms which provide reliable and guaranteed access on existing data-center infrastructure like Ethernet or PCIe bus.

Benefits

- Lowers TCO by increased utilization of resources (whether its memory, storage or compute) across racks and nodes.
- Lower power and cooling requirements as data is processed near the source.
- Support of advance functions like collectives, vector atomics enabling distributed compute operation or scatter-gather semantics to memory devices

Applications

Memory intensive application like scientific algorithm (HPC) or in-memory databases

Applications

Single-wide Compute (CPU, GPU, FPGA, etc)
Double-high Compute (CPU, GPU, FPGA, etc)
Single-wide DRAM Fabric Attached Memory
Double-wide DRAM Fabric Attached Memory
Single-wide SCM Fabric Attached Memory
Double-wide SCM Fabric Attached Memory
Benefits of composable memory architecture

“I don’t need to overprovision my system, I can add and re-distribute the resources as needed.”

“I can add modules without shutting down the server (high availability)”

“Memory can be shared across the nodes, and meets throughput and latency requirement”
References

- To learn more about Gen-Z protocol
  - Gen-Z Webinars
  - https://genzconsortium.org/educational-materials/
  - Gen-Z Specification
- To order or explore Gen-Z Micro Development kit and memory modules
  - Gen-Z Micro Dev Kit
  - Gen-Z Storage & Memory Product Innovation
  - Platform Software Development Kit for Gen-Z Enablement
- And visit www.smartm.com/genz
Please take a moment to rate this session.

Your feedback matters to us.
Backup
# Gen-Z µDK: Orthus ARM Host card (FPGA)

![Orthus ARM Host Card](image.jpg)

<table>
<thead>
<tr>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orthus ARM Host Card</td>
<td>PCIe full-height, 3/4 length Gen3 x16 AIC using Xilinx Zynq MPSoC FPGA with Quad Core ARM CPU - includes 4GB DDR4-2400 memory</td>
</tr>
<tr>
<td>Orthus Dual QSFP28</td>
<td>Each QSFP28 can be attached to 1 Gen-Z x4 lane connector at 100Gbps.</td>
</tr>
<tr>
<td>Orthus Debug Card</td>
<td>Ethernet board for remote management - JTAG debug card for ARM FPGA development</td>
</tr>
<tr>
<td>Orthus Software</td>
<td>Preloaded with Boot-loader, Linux Kernel &amp; Gen-Z applications utilities</td>
</tr>
</tbody>
</table>
# Gen-Z 64GB/256GB DRAM Memory Module (ZMM)

## Attributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Memory</th>
<th>Dual DDR4@2400 channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td></td>
<td>64GB and 256GB</td>
</tr>
<tr>
<td>Form Factor</td>
<td></td>
<td>EDSFF E3.S (3 inch, short) (SFF-TA-1008 compliant)</td>
</tr>
<tr>
<td>Host I/F</td>
<td></td>
<td>4C (x16) Gen-Z responder</td>
</tr>
<tr>
<td>Throughput</td>
<td></td>
<td>35 GB/s</td>
</tr>
<tr>
<td>Latency</td>
<td></td>
<td>366ns 64B read (FPGA bound)</td>
</tr>
<tr>
<td>Power</td>
<td></td>
<td>45W</td>
</tr>
</tbody>
</table>

## PHY

- GTY 802.3 5m+ copper cable
- Lane reversal and reduction
- Polarity Inversion

## Features

- Integrated Gen-Z switch
- Debug shell
- Packet history
- Performance monitoring
- Thermal management

---

© Gen-Z Consortium 2020
What type of media? DRAM, NAND, Storage Class Memory

<table>
<thead>
<tr>
<th></th>
<th>Volatile Memory (DDR4, DDR5 DRAM)</th>
<th>Storage Class Memory (3DXP, Intel Optane)</th>
<th>Storage (NAND Flash)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use</strong></td>
<td>Hot Data: Temporal data like</td>
<td>Warm Data</td>
<td>Cold Data</td>
</tr>
<tr>
<td></td>
<td>• packet or frame buffers,</td>
<td>• Storage cache</td>
<td>• Permanent data</td>
</tr>
<tr>
<td></td>
<td>• data structures, and</td>
<td>• Even a single uncorrectable error can</td>
<td>which are final</td>
</tr>
<tr>
<td></td>
<td>• control information</td>
<td>cause system crash or compromise the</td>
<td>outputs of a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>security</td>
<td>computational stage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>like Object blobs,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>logs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Raw Input from</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>external world which</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is to be processed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>like Image or video</td>
</tr>
<tr>
<td><strong>Granularity</strong></td>
<td>• Bytes (Cache-line granularity)</td>
<td>• Bytes (Cache-line granularity)</td>
<td>• Block/page-size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>granularity (Kilo-Bytes)</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td><strong>Theoretically infinite endurance.</strong></td>
<td><strong>Limited Endurance</strong></td>
<td><strong>Limited Endurance</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Requires firmware and middleware</td>
<td>Reliability is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>changes to improve reliability.</td>
<td>increased by:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ECC is necessary</td>
<td>• Adding redundancy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>and duplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Erasure coding</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>Low (&lt;100ns)</td>
<td>Moderate (200-1000ns)</td>
<td>High (10000-25000ns)</td>
</tr>
<tr>
<td><strong>Ordering</strong></td>
<td>Strongly Ordered</td>
<td>As required by application</td>
<td>Loosely ordered</td>
</tr>
</tbody>
</table>