Contribution ID: **64** Type: **not specified**

Linux Gen-Z Sub-system

Wednesday 11 September 2019 10:45 (45 minutes)

Gen-Z Linux Sub-system

Discuss design choices for a Gen-Z kernel sub-system and the challenges of supporting the Gen-Z interconnect in Linux.

Gen-Z is a fabric interconnect that connects a broad range of devices from CPUs, memory, I/O, and switches to other computers and all of their devices. It scales from two components in an enclosure to an exascale mesh. The Gen-Z consortium has over 70 member companies and the first version of the specification was published in 2018. Past history for new interconnects suggests we will see actual hardware products two years after the first specification - in 2020. We propose to add support for a Gen-Z kernel sub-system, a Gen-Z component device driver environment, and user space management applications.

A Gen-Z sub-system needs support for these Gen-Z features:

- Registration and enumeration services that are similar to existing sub-systems like PCI.
- Gen-Z Memory Management Unit (ZMMU) provides memory mapping and access to fabric addresses. The Gen-Z sub-system can provide services to track PTE entries for the two types of ZMMU's in the specification: page grid and page table based.
- Region Keys (R-Keys) - Each ZMMU page can have R-Keys used to validate page access authorization. The Gen-Z sub-system needs to provide APIs for tracking, freeing, and validating R-Keys.
- Process Address Space Identifier (PASID) - ZMMU requester and responder Page Table Entries (PTEs) contain a PASID. The Gen-Z sub-system needs to provide APIs for tracking PASIDs.
- Data mover - Transmit and receive data movers are optional elements in bridges and other Gen-Z components. The Gen-Z sub-system can provide a user space interface to a RDMA driver that uses a Gen-Z data mover. For example, a libfabric Gen-Z provider implementation can use a RDMA driver to access data mover queues.
- UUIDs - Components are identified by UUIDs. The Gen-Z sub-system provides interfaces for tracking UUIDs of local and remote components. A Gen-Z driver binds to a UUID similarly to how a PCI driver binds to a vendor/device id.
- Interrupt handling - Interrupt request packets in Gen-Z trigger local interrupts. Local components such as bridges and data movers can also be sources of interrupts.

We will discuss our proposed design for the Gen-Z sub-system illustrated in the following block diagram:

Indico rendering error

```
Could not include image: Cannot read image data. Maybe not an image file?
```

Gen-Z fabric management is global to the fabric. The operating system may not know what components on the fabric are assigned to it; the fabric manager decides which components belong to the operating system. Although user space discovery/management is unusual for Linux, it will allow the Gen-Z sub-system to focus on the mechanism of component management rather than the policy choices a fabric manager must make.

To support user space discovery/management, the Gen-Z sub-system needs interfaces for management services:

- Fabric managers need read/write access to component control space in order to do fabric discovery and configuration. We propose using /sys files for each control structure and table.
- User space Gen-Z managers need notification of management events/interrupts from the Gen-Z fabric. We propose using poll on the bridges' device files to communicate events.
- Local management services pass fabric discovery events from user space to the kernel. Our proposed design uses generic Netlink messages for communication of these component add/remove/modify events.

We are leveraging our experience with writing Linux bridge drivers for three different Gen-Z hardware bridges in the design of the Gen-Z Linux sub-system. Most recently, we wrote the DOE Exa-scale PathForward project's bridge driver with data movers (https://github.com/HewlettPackard/zhpe-driver). We wrote drivers for the Gen-Z Consortium's demonstration card that supports a block device and a NIC as well as a driver for the bridge in HPE's "The Machine" that is a precursor to Gen-Z.

From our work so far, here are questions we would like feedback on:

- We intend to expose control space in /sys so that user space fabric managers can work. We ask for feedback on the proposed hierarchy and mechanisms.
- Gen-Z uses PASIDs and the sub-system could use generic PASID interfaces. Any interest in this elsewhere in the kernel?
- We have need of generic IOMMU interfaces since Gen-Z ZMMU needs to interface with the IOMMU in a platform independent way. Any interest in this elsewhere in the kernel? We saw some patch sets along these lines.
- We intend to use generic NetLink for communication between user space and the kernel. Any thoughts on that decision?
- Gen-Z maps huge address spaces from remote components, and to get good performance those mappings need huge pages. Currently, the kernel does not support this use case. We would like to discuss how best to handle these huge mappings.
- We wrote a parser for the Gen-Z specification's control structure that generates C structures with bitfields. In general, we know the Linux kernel frowns on bitfields. Are bitfields ok in this context?

I agree to abide by the anti-harassment policy

Yes

Primary authors: HULL, Jim (Hewlett Packard Enterprise); DALL, Betty (HPE); PACKARD, Keith (Hewlett Packard Enterprise)

Presenters: HULL, Jim (Hewlett Packard Enterprise); DALL, Betty (HPE); PACKARD, Keith (Hewlett Packard Enterprise)

Session Classification: LPC Refereed Track