Tech Preview

Emulated CXL Development Environments

Accelerate CXL Learning, Development, and Testing with MemVerge CXL Virtual Machines

Emulated CXL Made Easy

- Use emulated CXL devices in a QEMU Virtual Machine environment for learning, development, and testing
- Container images are available for rapid deployment
- Configure the desired number of CXL devices per VM
- Run as many Virtual Machines as you need

Emulating Memory Expanders

- Kernel Driver requires special kernel flag to work correctly with QEMU
- MemVerge providing pre-built images and docker containers to avoid this complexity

Emulating Memory Expanders

```
// Step #1: Pull the Container image
$ podman pull mvpool/qemu_cxl_memexp

// Step #2: Start the Container
$ podman run -it qemu_cxl_memexp bash

// Step #3: SSH to the new VM
$ ssh -p 2222 fedora@localhost
```

Emulated Memory Expanders

```
// Step #4: Create a Region
[fedora@localhost ~]$ ./create_region.sh
{
  "region":"region0",
  "resource":"0x390000000",
  "size":"4.00 GiB (4.29 GB)",
  "type":"ram",
  "interleave_ways":1,
  "interleave_granularity":4096,
  "decode_state":"commit",
  "mappings":[
    {
      "position":0,
      "memdev":"mem0",
      "decoder":"decoder2.0"
    }
  ]
}
cxl region: cmd_create_region: created 1 region
onlined memory for 1 device
```

Emulated CXL Made Easy

```
// Step #5: View & use the CXL-backed NUMA Node (Node 1)
[fedora@localhost ~]$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3
node 0 size: 3901 MB
node 0 free: 3506 MB
node 1 cpus:
node 1 size: 4096 MB
node 1 free: 4096 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10    << CXL
```

NUMA vs DAX mode

```
// Step #6: Switch modes with pre-made scripts
[fedora@localhost ~]$ ./dax_mode.sh
Offlined memory for 1 device
Reconfigured 1 device

[fedora@localhost ~]$ ./numa_mode.sh
Reconfigured 1 device
Onlined memory for 1 device
```

Emulating Shared Memory

- Multi-headed device emulation is complex
- We are hosting pre-built Containers/VMs to enable quick prototyping of software

Shared Memory Example

```
// Step #1: Pull the Container image
$ podman pull mvpool/qemu_cxl_shared

// Step #2: Start the Container
$ podman run -p 2222:2222 -p 2223:2223
mvpool/qemu_cxl_shared

// Step #3: SSH to the new VMs
$ ssh -p 2222 fedora@localhost
$ ssh -p 2223 fedora@localhost
```

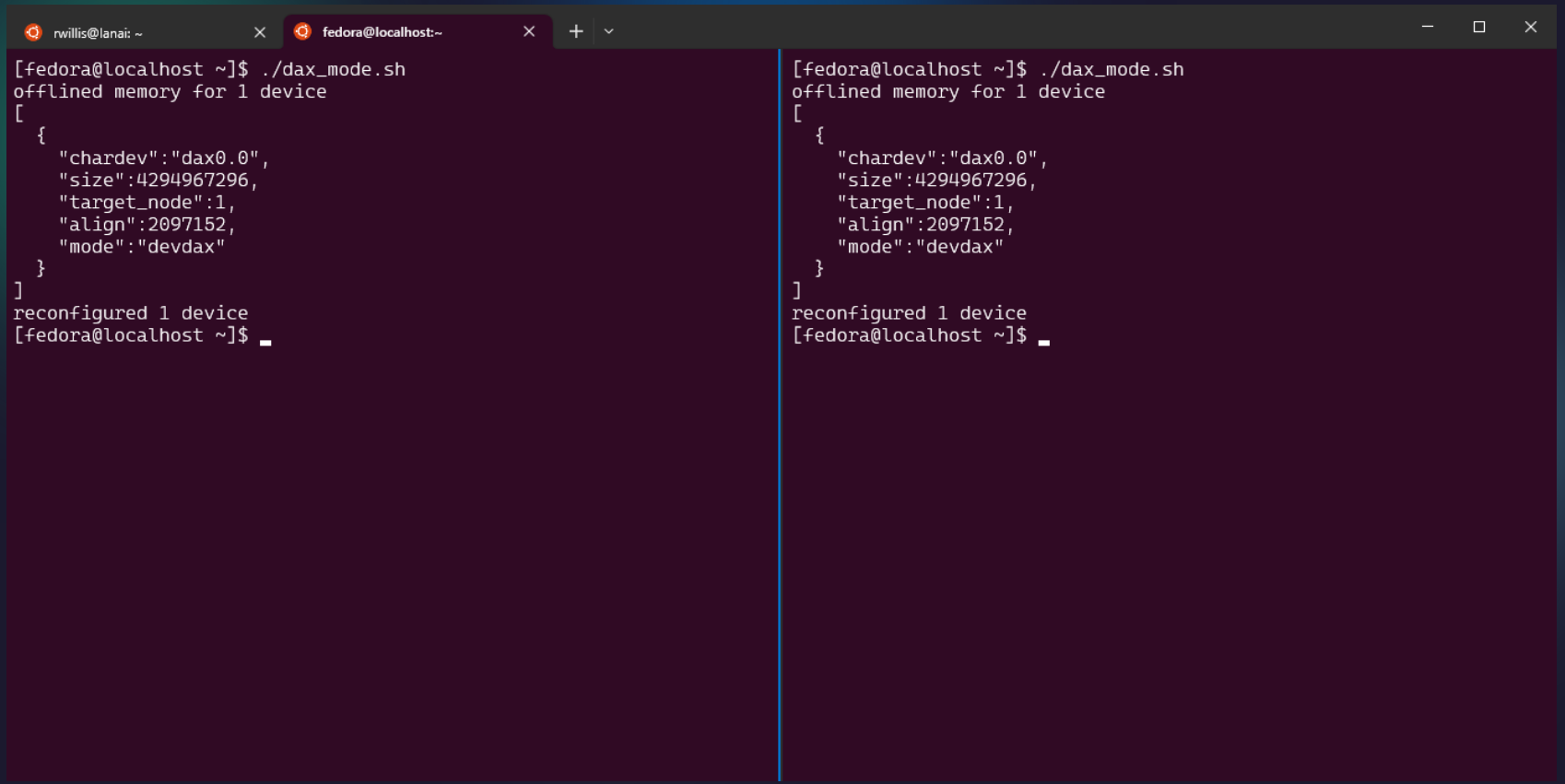
Create regions on VMs

rwillis@lanai: ~ | fedora@localhost:~

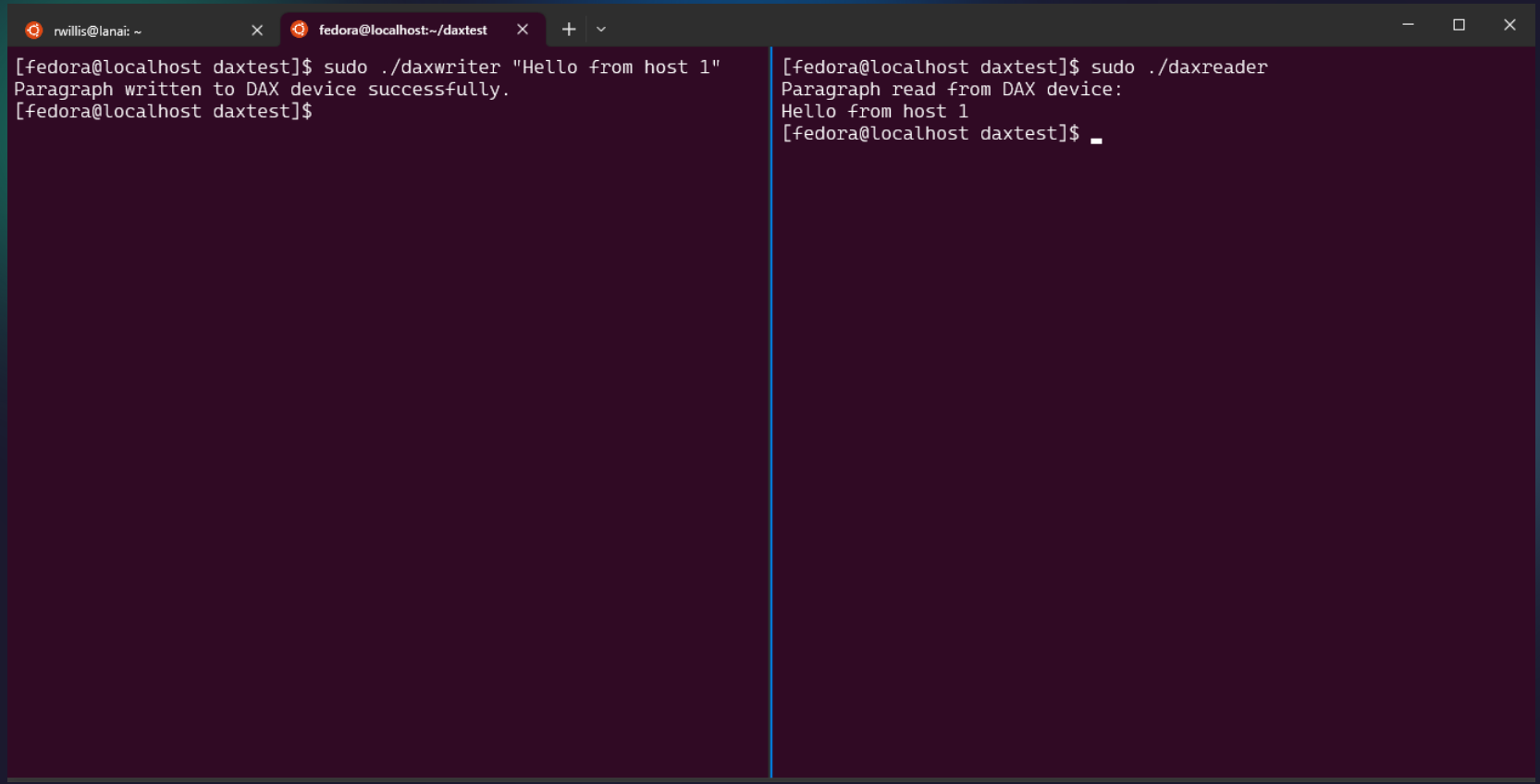
```
[fedora@localhost ~]$ ./create_region.sh
{
  "region":"region0",
  "resource":"0x390000000",
  "size":"4.00 GiB (4.29 GB)",
  "type":"ram",
  "interleave_ways":1,
  "interleave_granularity":4096,
  "decode_state":"commit",
  "mappings":[
    {
      "position":0,
      "memdev":"mem0",
      "decoder":"decoder2.0"
    }
  ]
}
cxl region: cmd_create_region: created 1 region
onlined memory for 1 device
[fedora@localhost ~]$
```

```
[fedora@localhost ~]$ ./create_region.sh
{
  "region":"region0",
  "resource":"0x390000000",
  "size":"4.00 GiB (4.29 GB)",
  "type":"ram",
  "interleave_ways":1,
  "interleave_granularity":4096,
  "decode_state":"commit",
  "mappings":[
    {
      "position":0,
      "memdev":"mem0",
      "decoder":"decoder2.0"
    }
  ]
}
cxl region: cmd_create_region: created 1 region
onlined memory for 1 device
[fedora@localhost ~]$
```

Set Both to Dax Mode



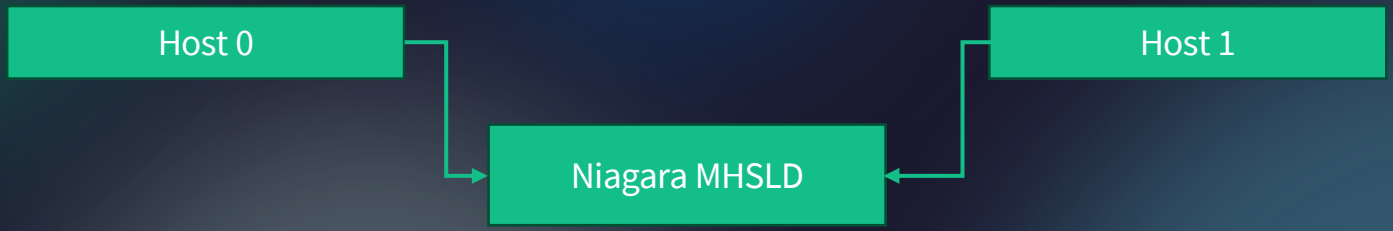
Run Shared Memory Test



Emulating Mulit-Headed SLD

- MemVerge and SK hynix have worked together to create a model of the SK hynix Niagara platform
- Patches submitted upstream to QEMU, driver to be submitted
- Multi-headed SLD with memory-pooling capabilities
- Will be releasing a prebuilt image and container @ mvpool:cxl_qemu_niagara:latest

Emulating Multi-Headed SLD



- Allocate memory from pool via mailbox commands
- Simple user software to online/offline allocated memory in numa node

Get Started Today!

https://memverge.com/emulating-cxl-shared-memory-devices-in-qemu/