SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

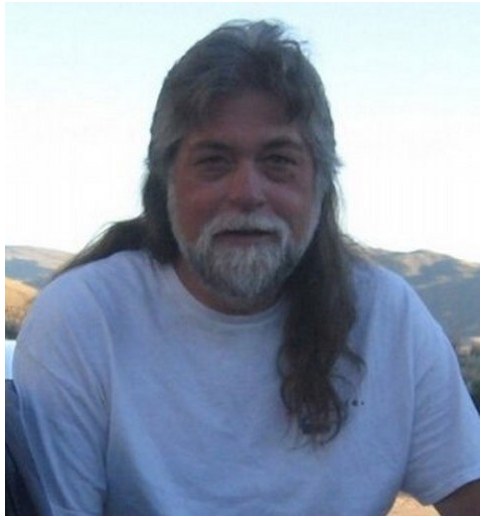
September 16-18, 2024
Santa Clara, CA

Designing and Optimizing Complex CXL Configurations

Craig Carlson – AMD
Grant Mackey – JackRabbit Labs
Andy Banta – Magnition IO

Andy Banta

Magnition.io (Consultant)
SolidFire (VMware development, acq. by NetApp)
DataGravity (Container exploitation lead)
VMware (iSCSI Tech Lead, IPO)
Sun Microsystems (Initial Fibre Channel development)
Patent, early distributed network projects, data acquisition
@andybanta



Magnition

SDC 24

JACKRABBIT LABS

Grant Mackey

Jackrabbit Labs (CTO)
Western Digital (Sr. Technologist)
LANL (Researcher)

https://www.linkedin.com/in/grant-mackey/

SDC 24

Craig Carlson

CXL Continuing Innovation

Section Subtitle

AMD CXL Innovation Content

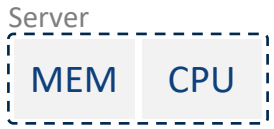
- Awaiting AMD …

Composable Memory Systems & Software Development

Emulation, Simulation, and otherwise

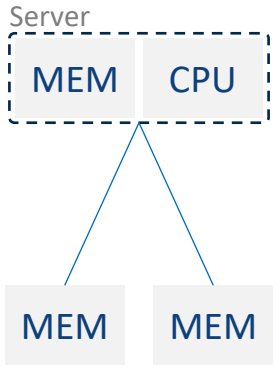
How CXL Addresses Industry Computing Challenges

Standard Server



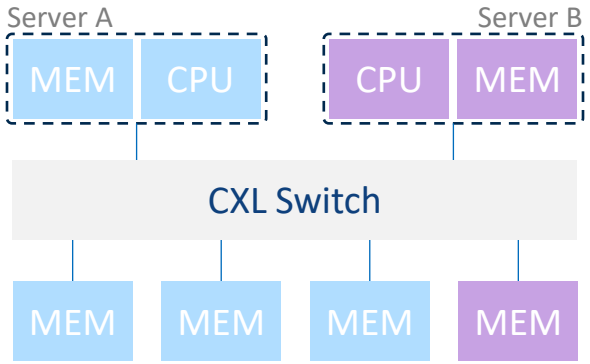
Single Node resource constraints

Memory Expansion



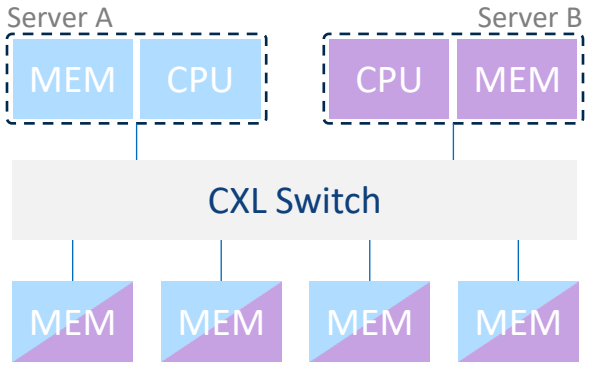
Locally attached ‘far’ memory alleviates ‘core stranding’ issues in single servers

Pooled Memory



Fabric Attached ‘far’ memory allows for refinement of memory resource allocation

Shared Memory



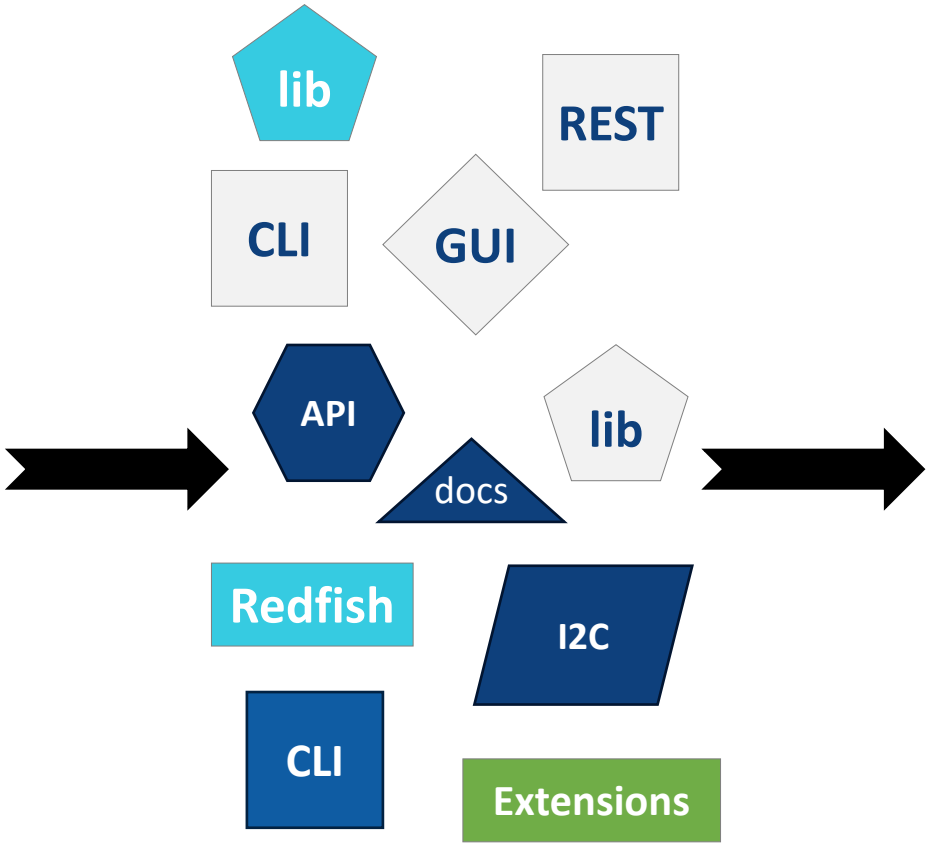
Shared memory and enhanced fabric composability alleviates network latencies needs at local cluster levels

Why is Open-Source Software Needed for CXL?

Emerging Hardware Ecosystem

- Specification
- Compute
- Switches
- Memory Controllers
- Memory Modules
- Appliances

Fragmented Software Ecosystem



Standardized Software Stack

- App Plugins
- Management API
- Management Libraries
- Host Agent(s)
- In-band Management Protocol
- Management Endpoints
- SwitchOS
- Hardware Abstraction Layer

Software Layers in Need of More Development Effort

1: SwitchOS

- Runs on the BMC of the switch appliance. Performs hardware discovery, configuration, and monitoring. Based on the SONiC operating system used today for Ethernet switch appliances

2: A Robust fabric orchestration protocol

- Existing fabric management APIs are designed to be spoken between SwitchOS and the CXL switch ASIC. A new protocol is needed that is spoken between the host and the SwitchOS

3: Orchestration framework & CLI utilities

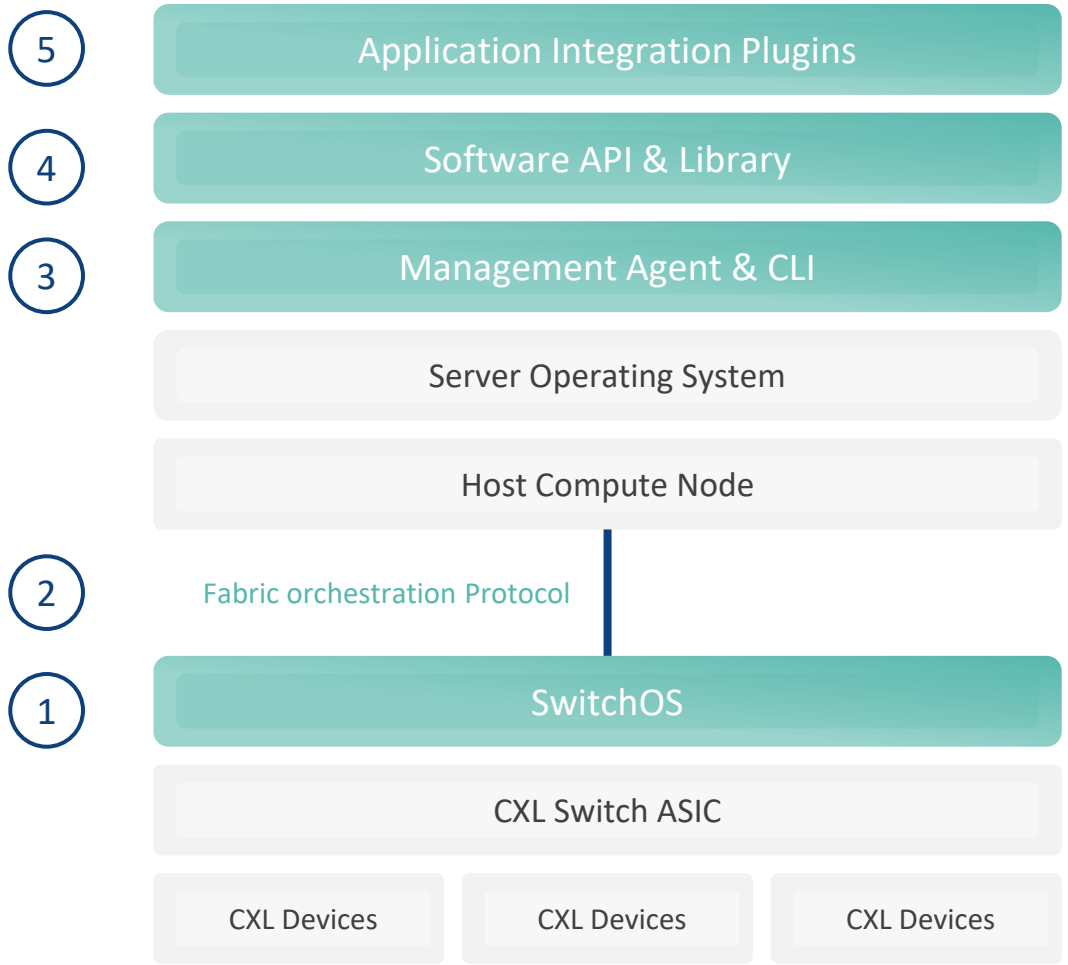
- Each host connected to the external CXL switch requires an agent to manage communication with the SwitchOS or to implement the operating system changes needed

4: Software API & library

- Provides a software library interface to orchestration applications that need to configure the underlying CXL hardware

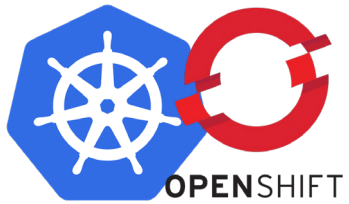
5: Orchestration application integration / plugins

- Comprises the software libraries / changesets needed to integrate CXL fabric management into existing orchestration applications (e.g. Kubernetes, OpenStack, Libvirt)



Consumers of Composable Memory Fabrics: "I'm paid to lead not to read"

- Resource schedulers don't want to know how memory fabrics like CXL work
  - They don't care about Ultra/Ethernet/Infiniband/NV or UALink either.
  - They want the OS or a module to handle it so they can schedule resources



Container 'x' interfaces
- Resource, CRI
- Storage, CSI
- Network, CNI

AND device plugin support

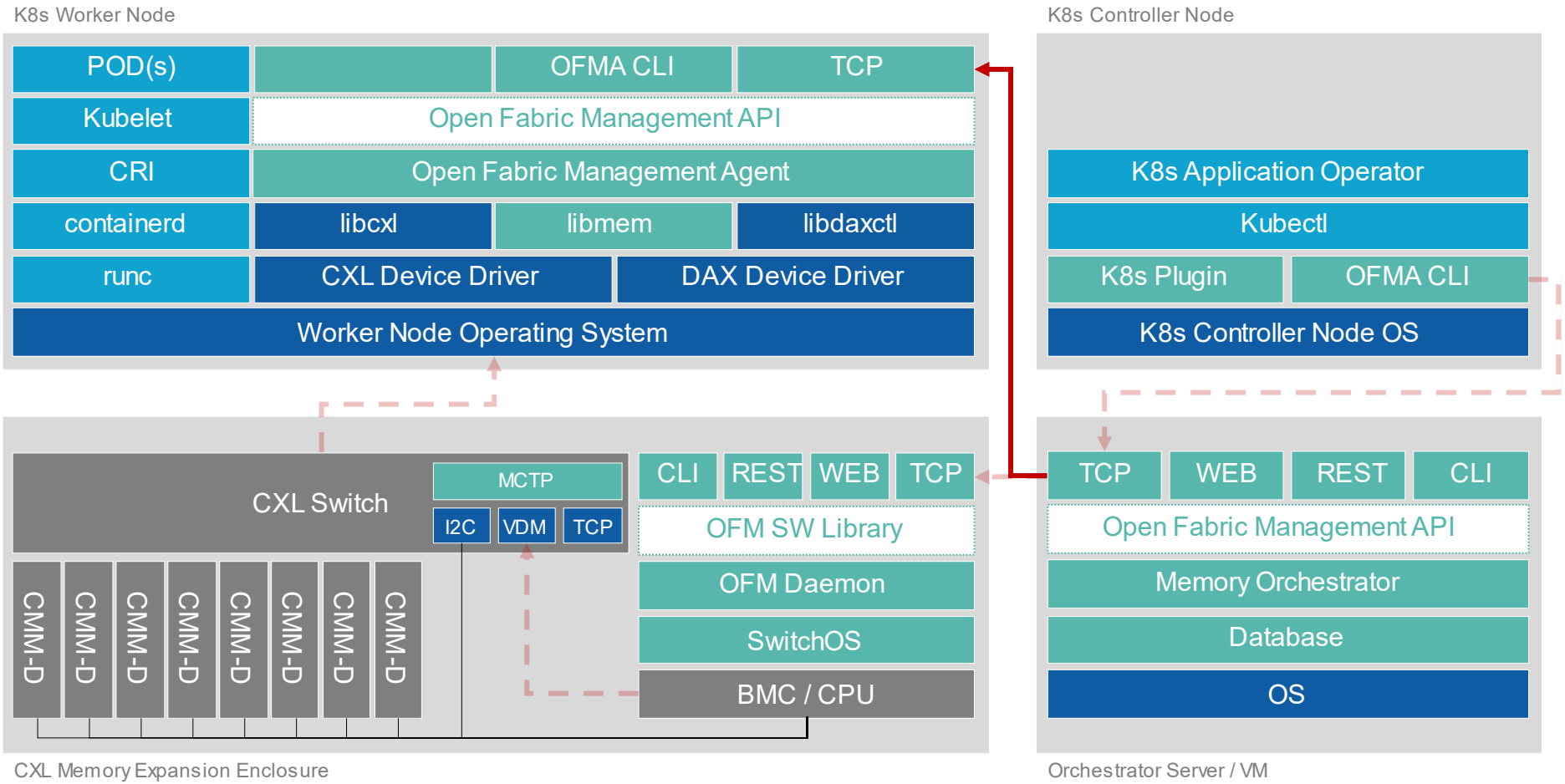
completely punt on caring about hardware

Ceph – storage
corosync – state sync

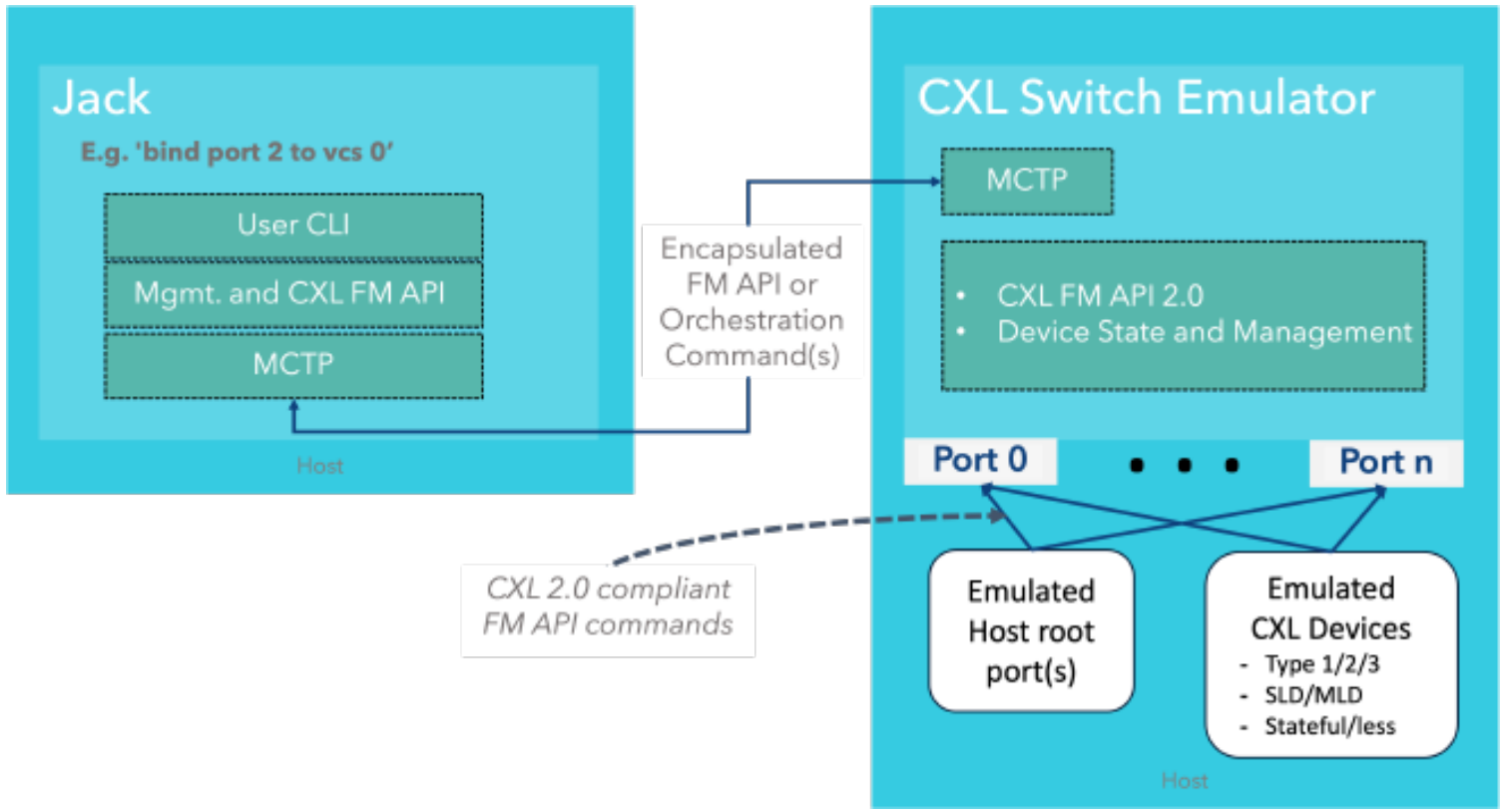
The chimera!

Has 41 types of resource services with varying levels of hardware abstraction

Composable Memory and K8s?

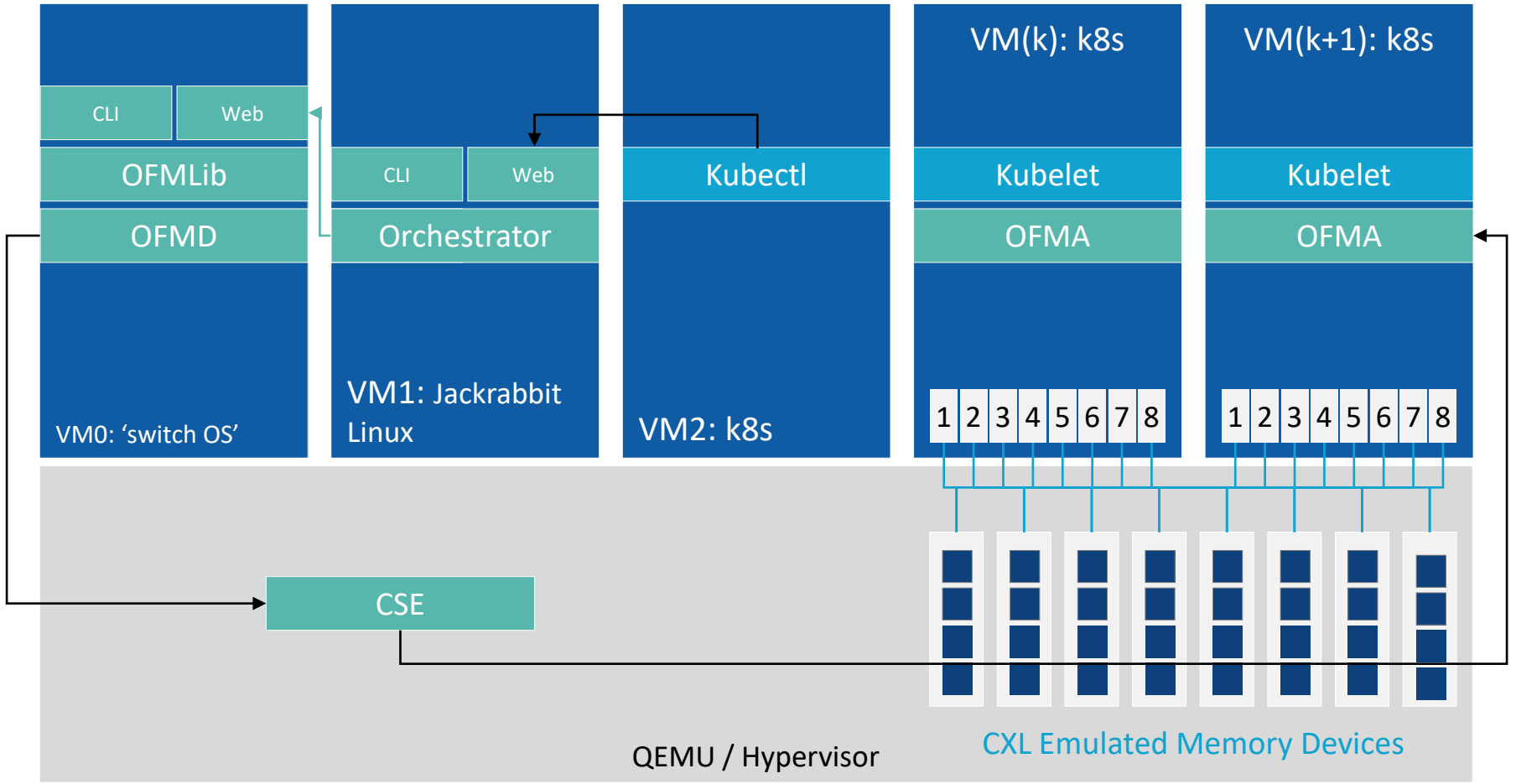


Fabric Orchestration and Management

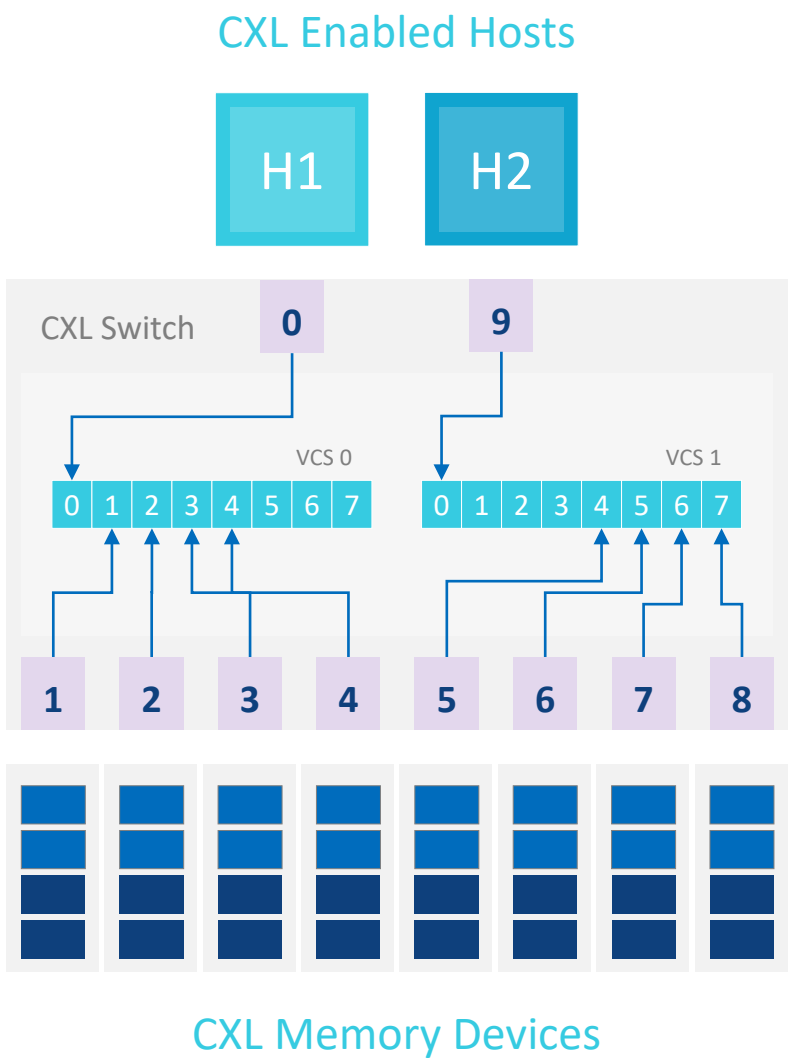


- Open-source CXL Fabric Mgmt. and orchestration
  - Apache2.0
  - JRL GitHub
- Jack (Just Another Cli orKestrator)
  - Command line tool to interface with mgmt. and CXL FMAPI libs
  - Commands sent to CSE via MCTP (presently)
- CXL Switch Emulator (CSE)
  - Switch and device emulation
  - Basic fabric orchestration
  - Fully CXL2.0 FMAPI and errata compliant
  - Qemu CXL device and switch support

What we're developing now



YT placeholder




```
# jack show port

#    @  Port State  Type    LD  Ver  CXL Ver   MLW  NLW  MLS  CLS  Speeds    LTSSM     LN  Flags
---  -  ----------  ------  --  ---  --------  ---  ---  ---  ---  --------  --------  --  ------------
0    +  Upstream    T1      -   2.0  AB        16   16   5.0  5.0       45   L0        0     P
1    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
2    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
3    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
4    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
5    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
6    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
7    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
8    +  Downstream  T3-MLD  16  2.0  AB        16   16   5.0  5.0       45   L0        0     P
9    +  Upstream    T1      -   2.0  AB        16   16   5.0  5.0       45   L0        0     P

# jack show vcs 0

Show VCS:
VCS ID  : 0
State   : Enabled
USP ID  : 0
vPPBs   : 8

vPPB  PPID LDID Status
----  ---- ---- -----------
   0:    0    - Bound Physical Port
   1:    1    0 Bound LD
   2:    2    0 Bound LD
   3:    3    0 Bound LD
   4:    4    0 Bound LD
   5:    -    - Unbound
   6:    -    - Unbound
   7:    -    - Unbound
```

https://github.com/JackrabbitLabs/jack

Composable Memory and CXL are Moving

- Challenges
  - Potential fragmentation of the shared memory software ecosystem will delay adoption
  - Lack of application development in the open
  - Lack of platforms emulated or real to do said development on
- Call to Action
  - Software application development doesn't have to wait until hardware is available
    - Evaluate where these tools can be used in your projects

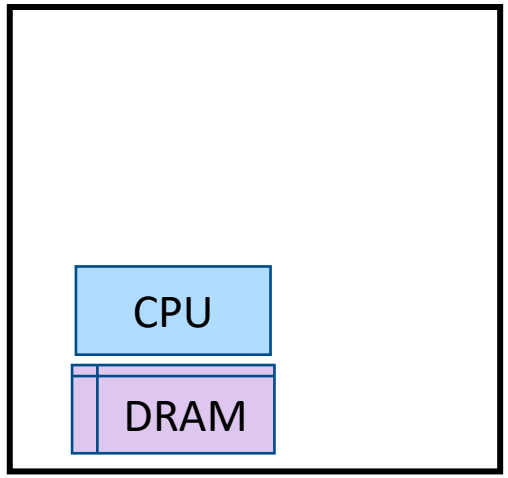
- Linux & QEMU CXL Meeting Notes:
  - https://pmem.io/ndctl/collab/
- Linux Kernel Mailing list:
  - https://lore.kernel.org/linux-cxl/
- OCP Composable Memory Systems
  - https://www.opencompute.org/wiki/Server/CMS

Intel	libcxl	https://github.com/pmem/ndctl
Jackrabbit Labs	libmem	https://github.com/JackrabbitLabs/libmem
Jackrabbit Labs	Jack - CXL FM API CLI Tool	https://github.com/JackrabbitLabs/jack
Jackrabbit Labs	CXL Switch Emulator	https://github.com/JackrabbitLabs/cse
Samsung	Scalable Memory Development Kit (SMDK)	https://github.com/OpenMPDK/SMDK
Micron	CXL Memory Resource Kit (CMRK)	https://github.com/cxl-micron-reskit/cxl-reskit
SK Hynix	Heterogeneous Memory Software Development Kit (HMSDK)	https://github.com/skhynix/hmsdk
Micron	CXL Library CLI	https://github.com/cxl-micron-reskit/mxcli
Micron	FAMFS	https://github.com/cxl-micron-reskit/famfs
Intel	Unified Memory Framework	https://github.com/oneapi-src/unified-memory-framework
QEMU	QEMU	https://github.com/qemu/qemu
Samsung	libcxlmi	https://github.com/computexpresslink/libcxlmi

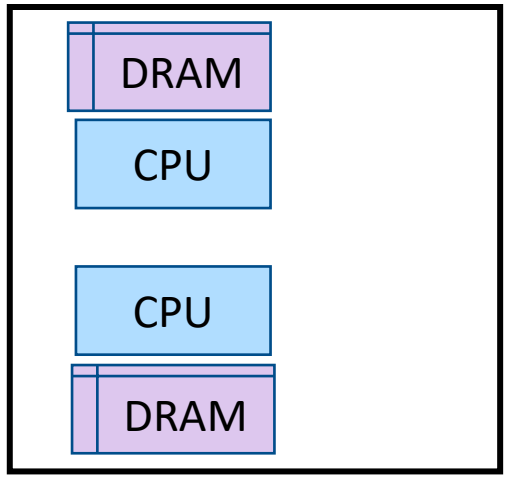
CXL Test Drives

Section Subtitle

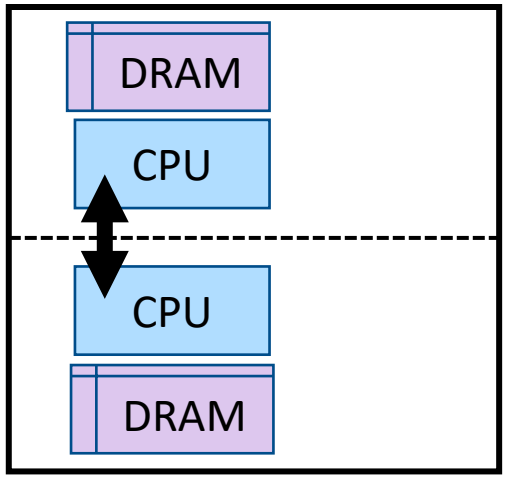
Multi-level Caching within CXL Hosts



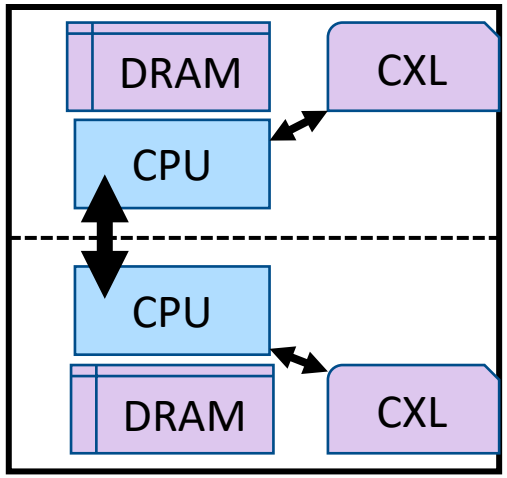
Multi-level Caching within CXL Hosts



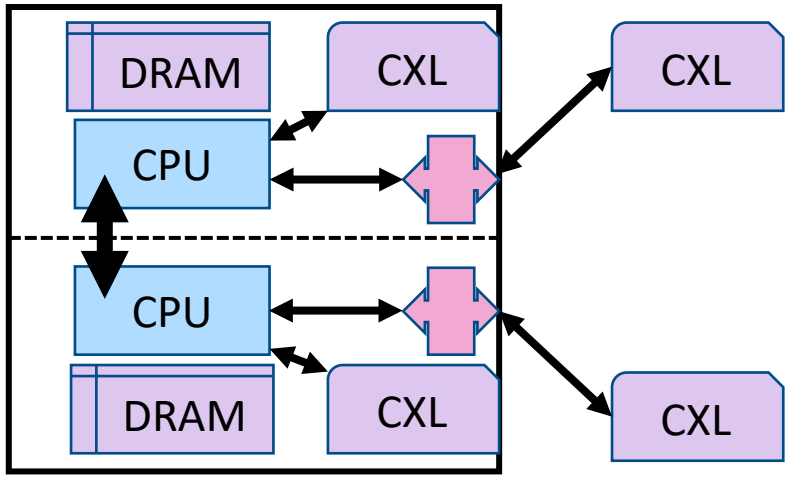
Multi-level Caching within CXL Hosts



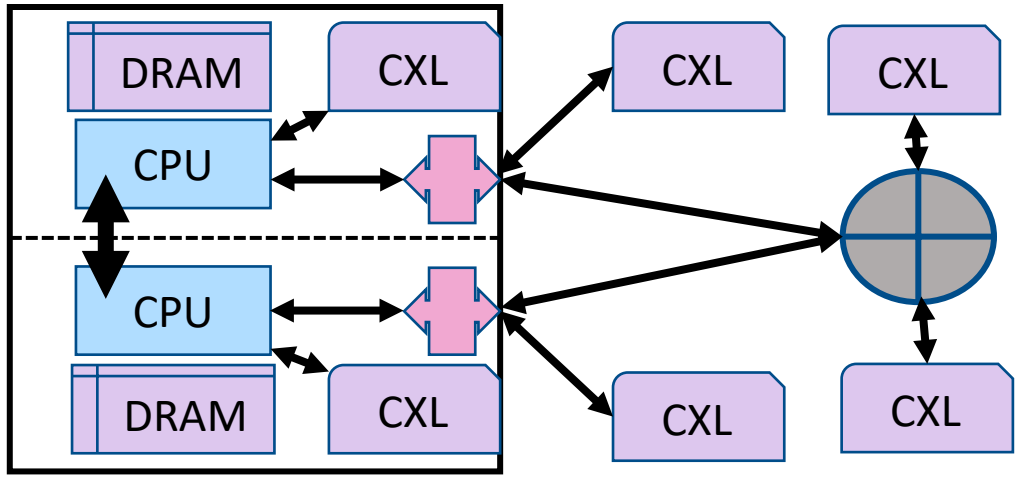
Multi-level Caching within CXL Hosts



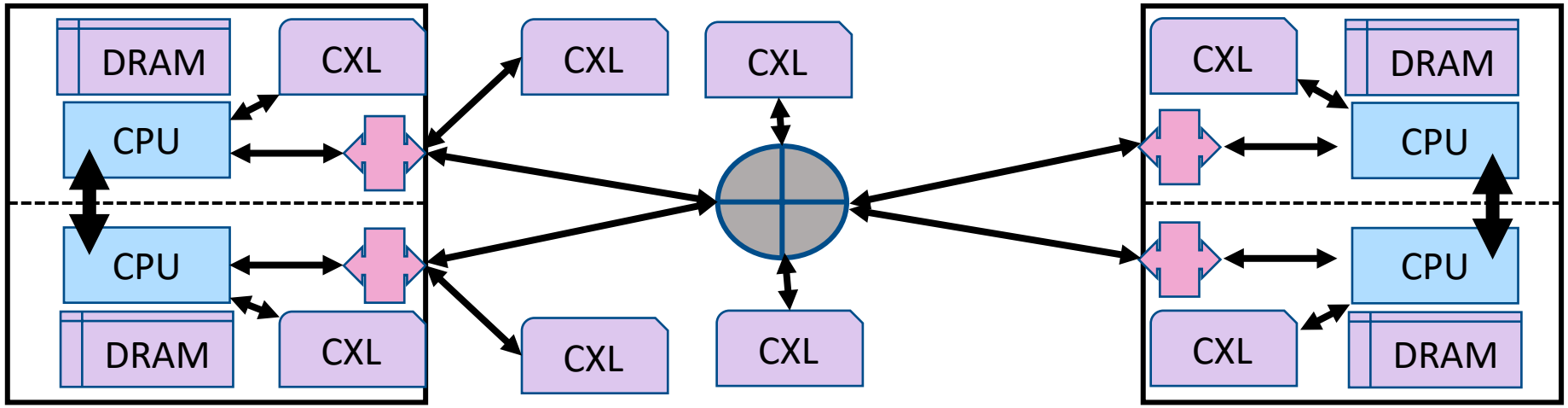
Multi-level Caching within CXL Hosts



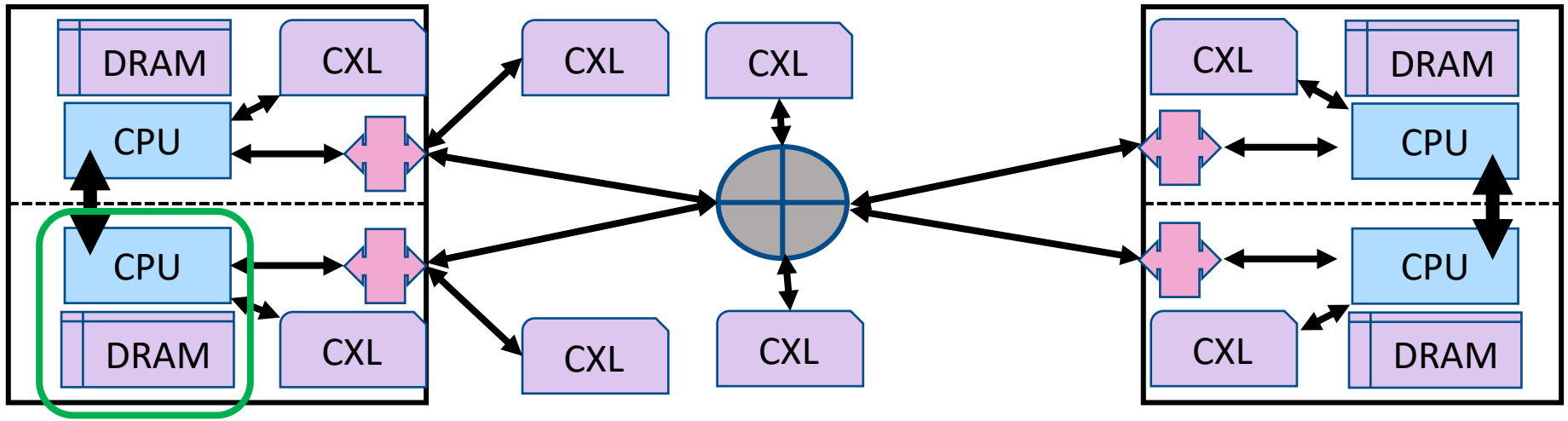
Multi-level Caching within CXL Hosts



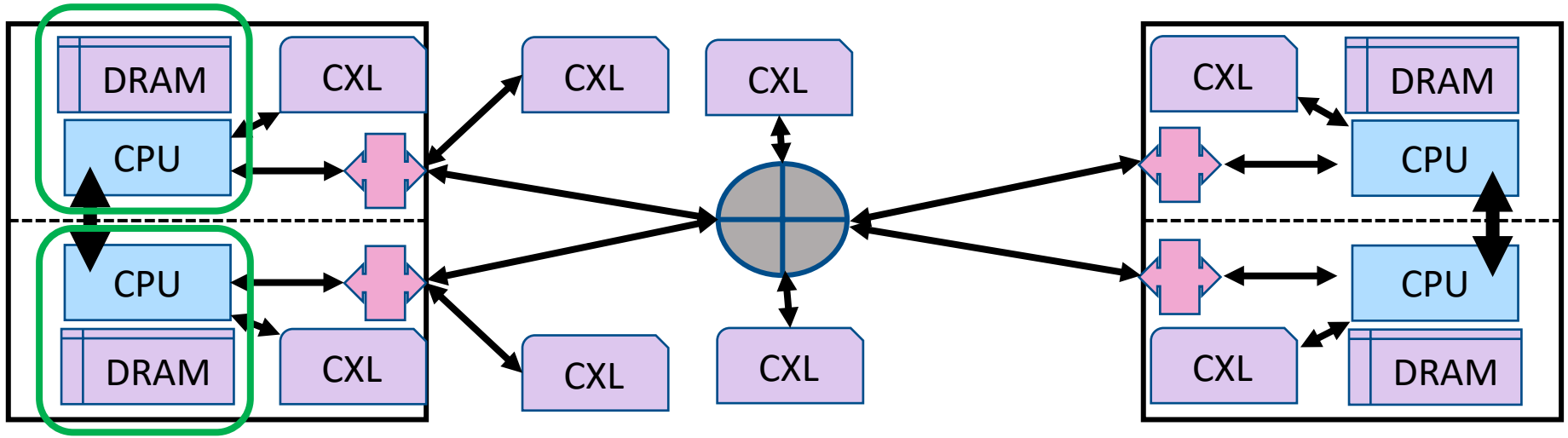
Multi-level Caching within CXL Hosts



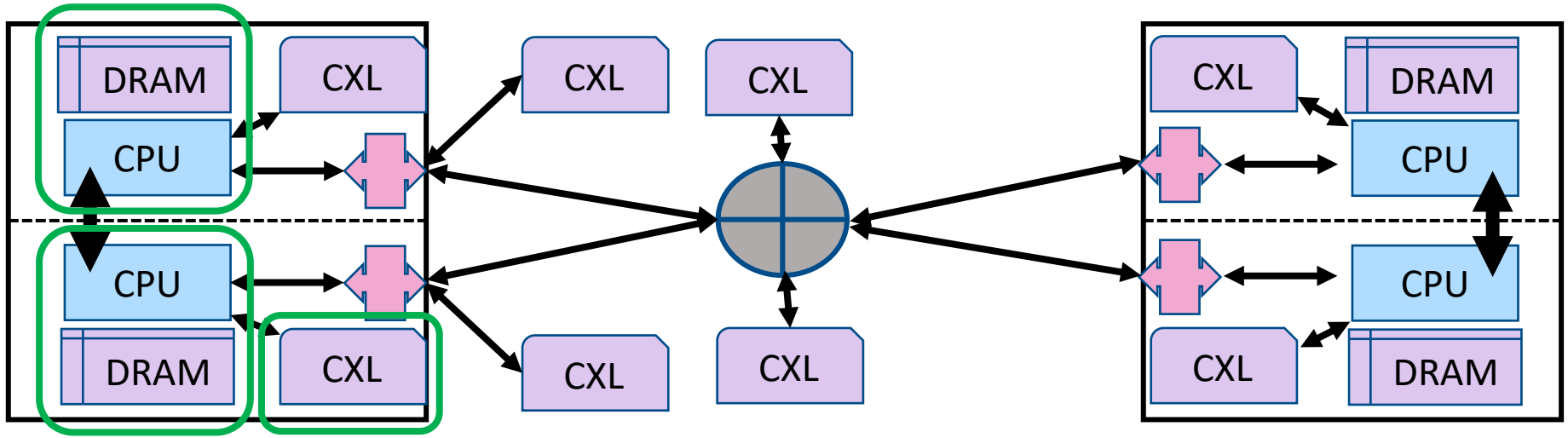
Multi-level Caching within CXL Hosts



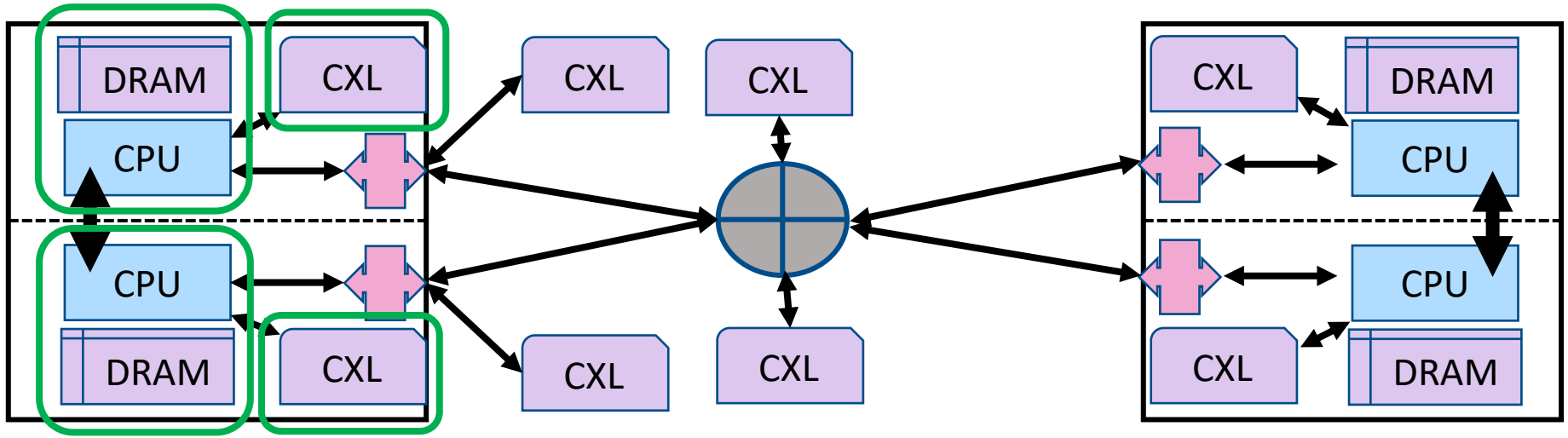
Multi-level Caching within CXL Hosts



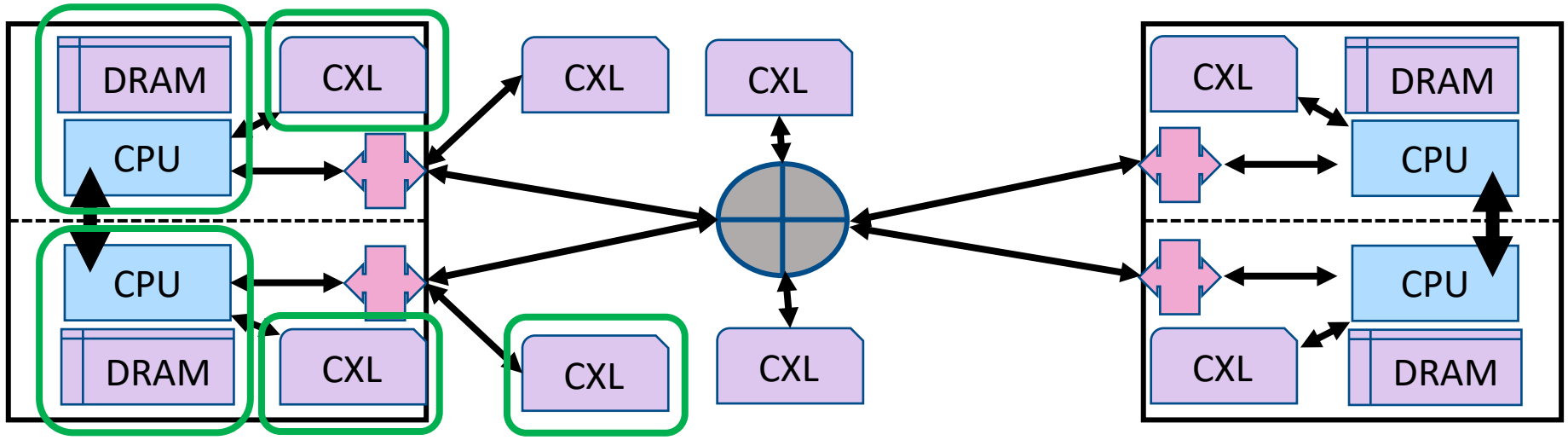
Multi-level Caching within CXL Hosts



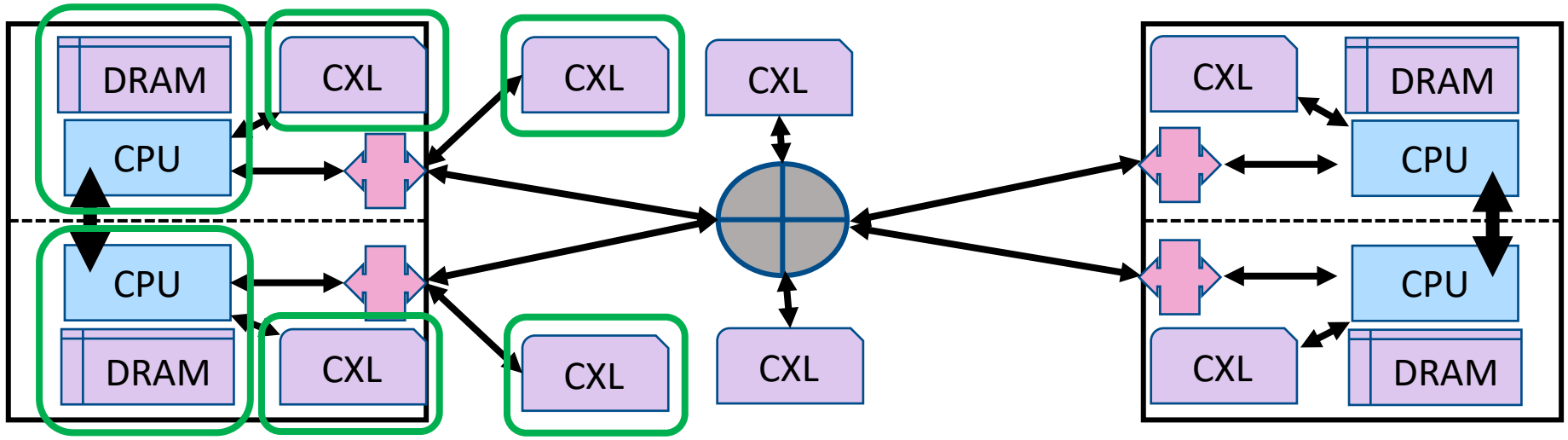
Multi-level Caching within CXL Hosts



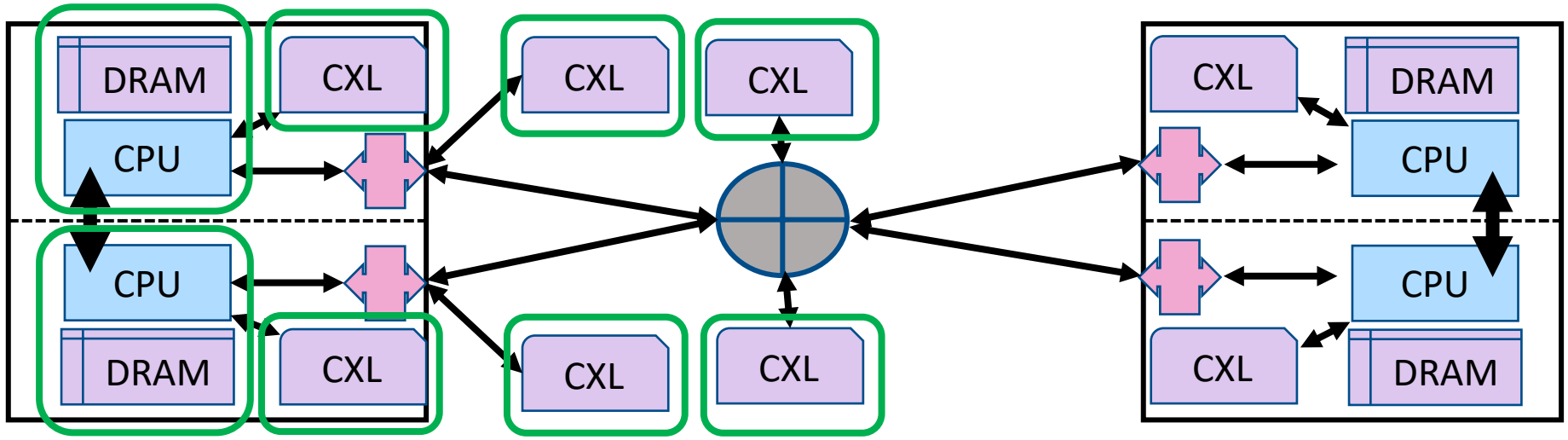
Multi-level Caching within CXL Hosts



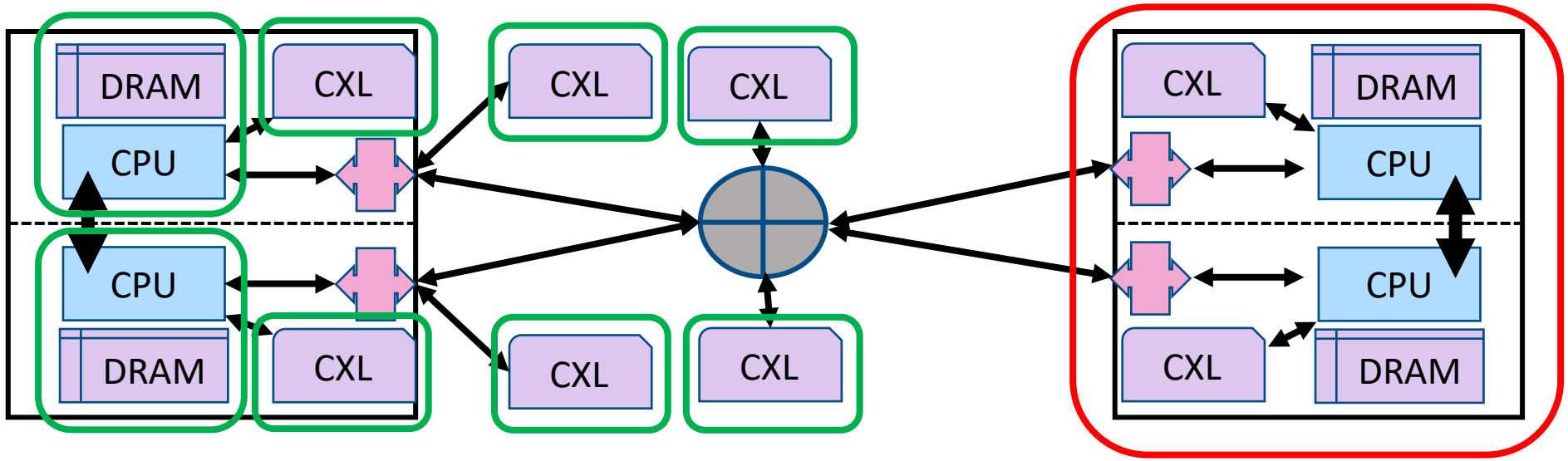
Multi-level Caching within CXL Hosts



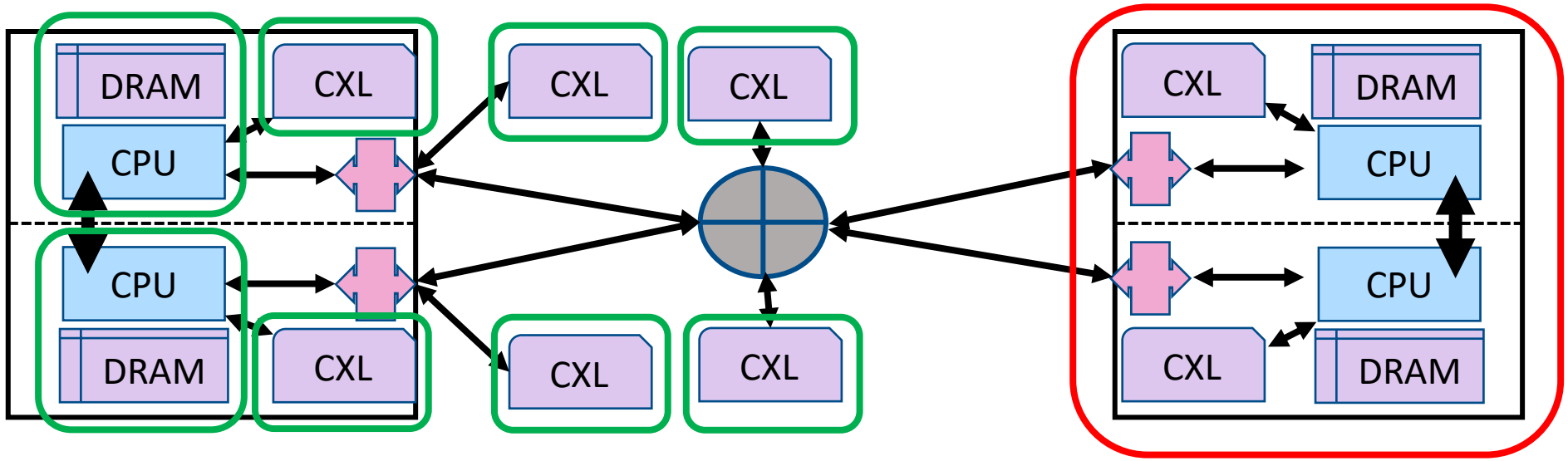
Multi-level Caching within CXL Hosts



Multi-level Caching within CXL Hosts

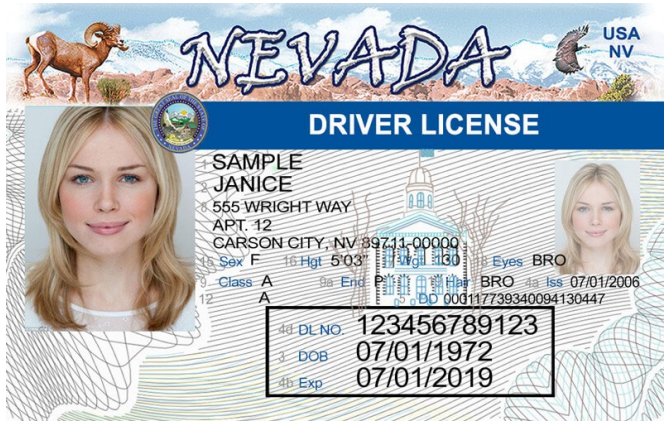


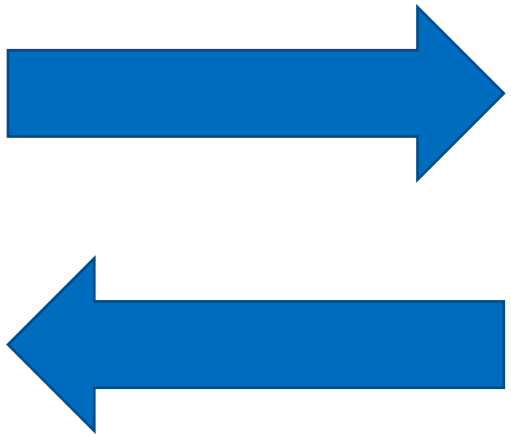
Multi-level Caching within CXL Hosts



7 different NUMA nodes plus contention (Abstract distance values)

Cool. How do I try it out?





Test Drive Limitations

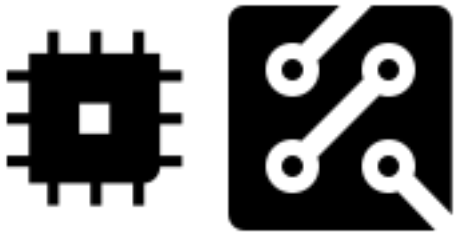
- Long road trip
- How much can it carry
- How is it on a fast country road

CXL Test Drive

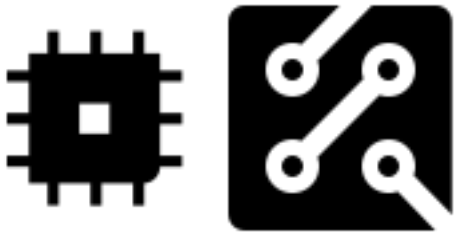
- Different workloads
  - Ability to run whatever you want
  - Multiple competing workloads
- Capacity planning
- Data placement
- Speed of individual tiers
- Adaptability over time
- **This is engineering!**

Engineering Means Simulation

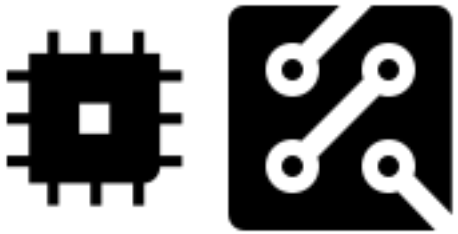
Engineering Means Simulation



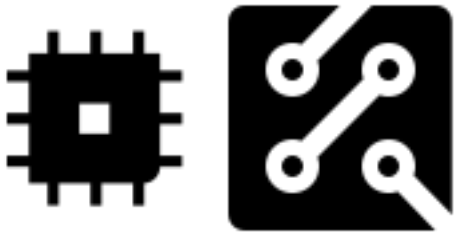
Engineering Means Simulation



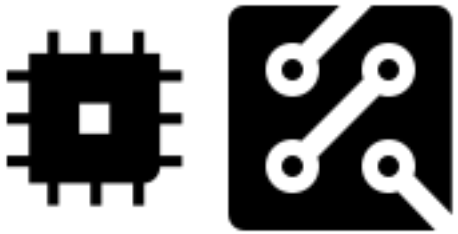
Engineering Means Simulation

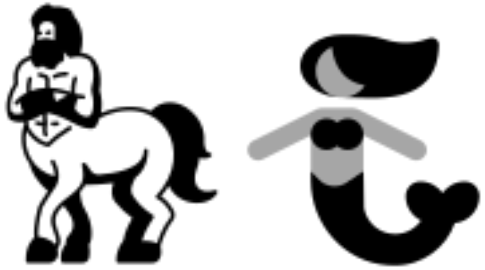


Engineering Means Simulation



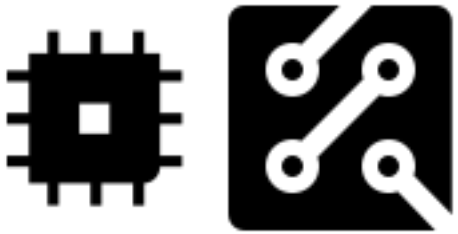
Engineering Means Simulation





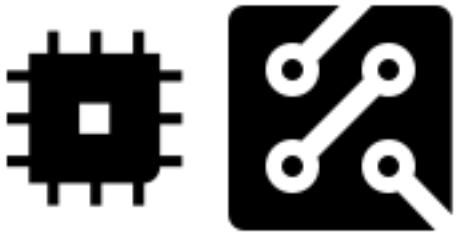
Engineering Means Simulation

- Cheaper, faster, more flexible than system building



Engineering Means Simulation

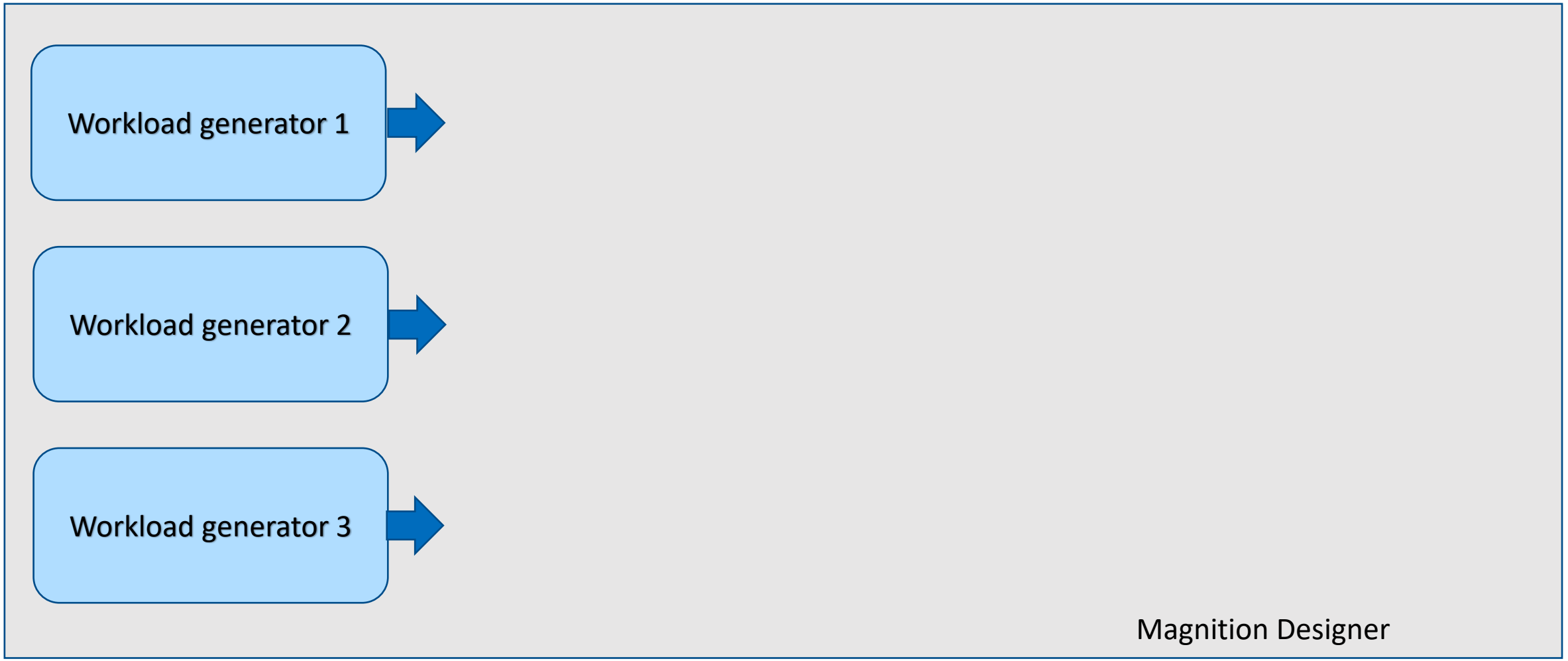
- Cheaper, faster, more flexible than system building
- Engineering design uses simulations, why not software?



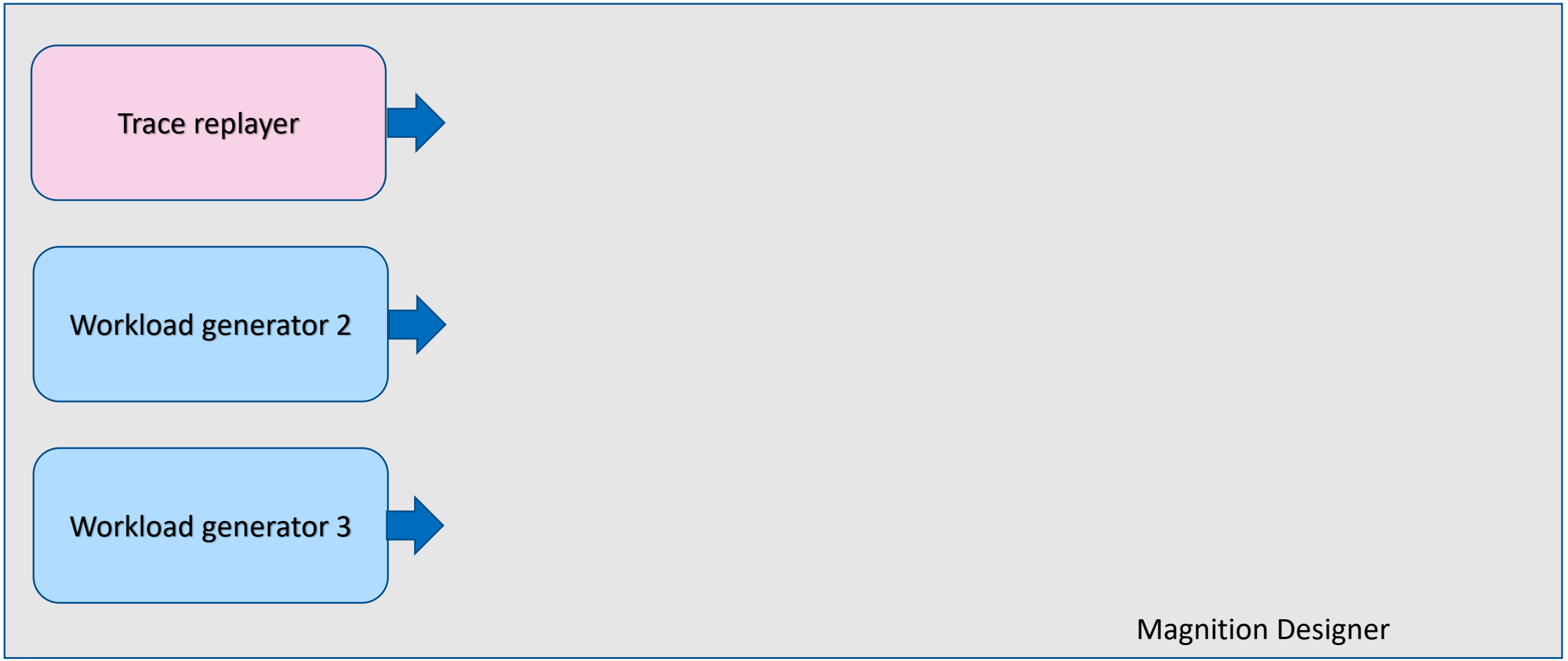
How to Build a Test Drive

Magnition Designer

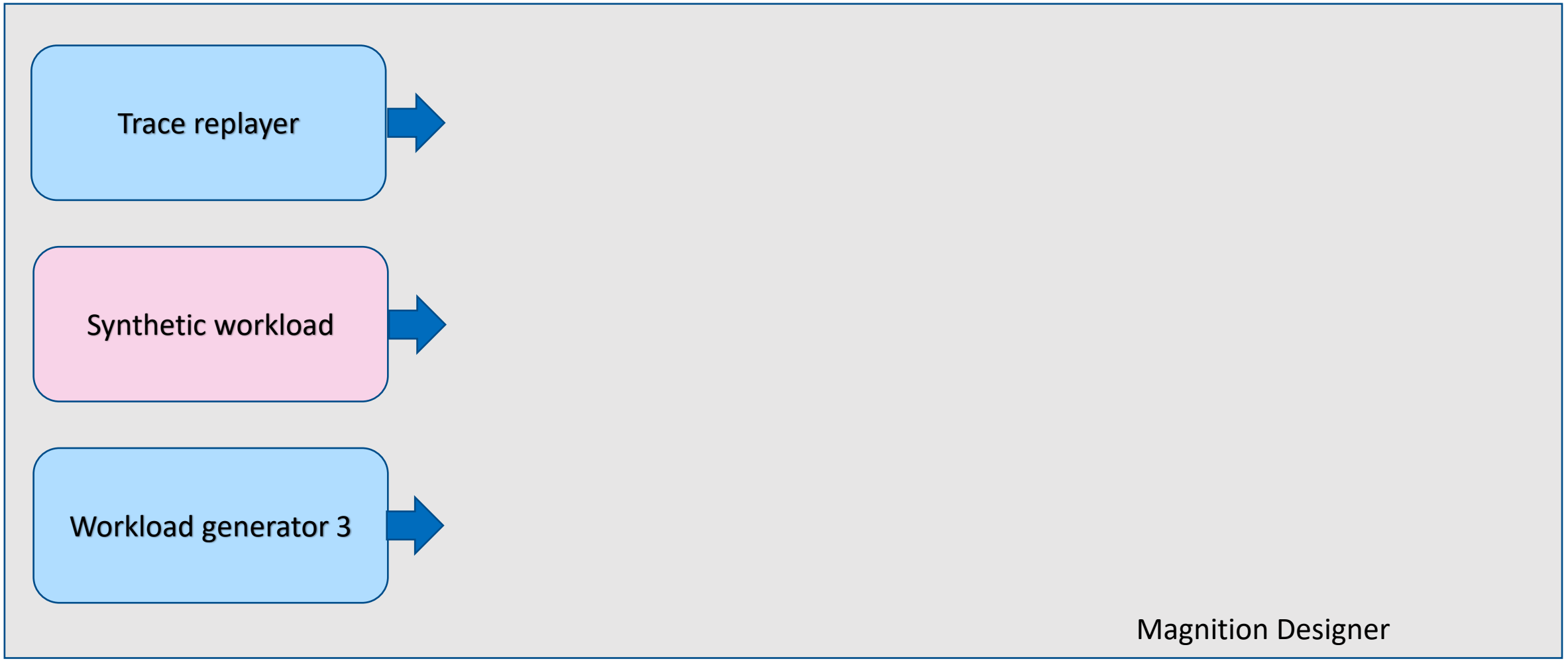
How to Build a Test Drive



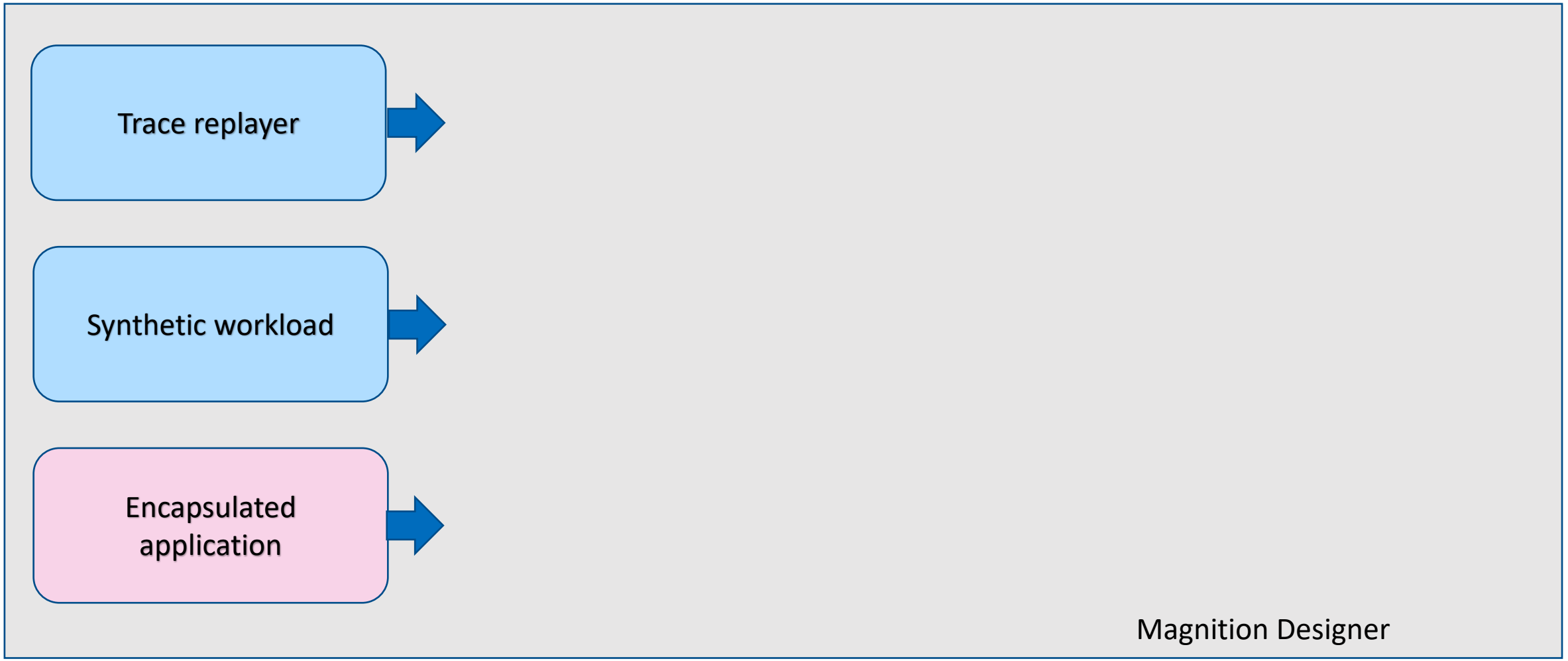
How to Build a Test Drive



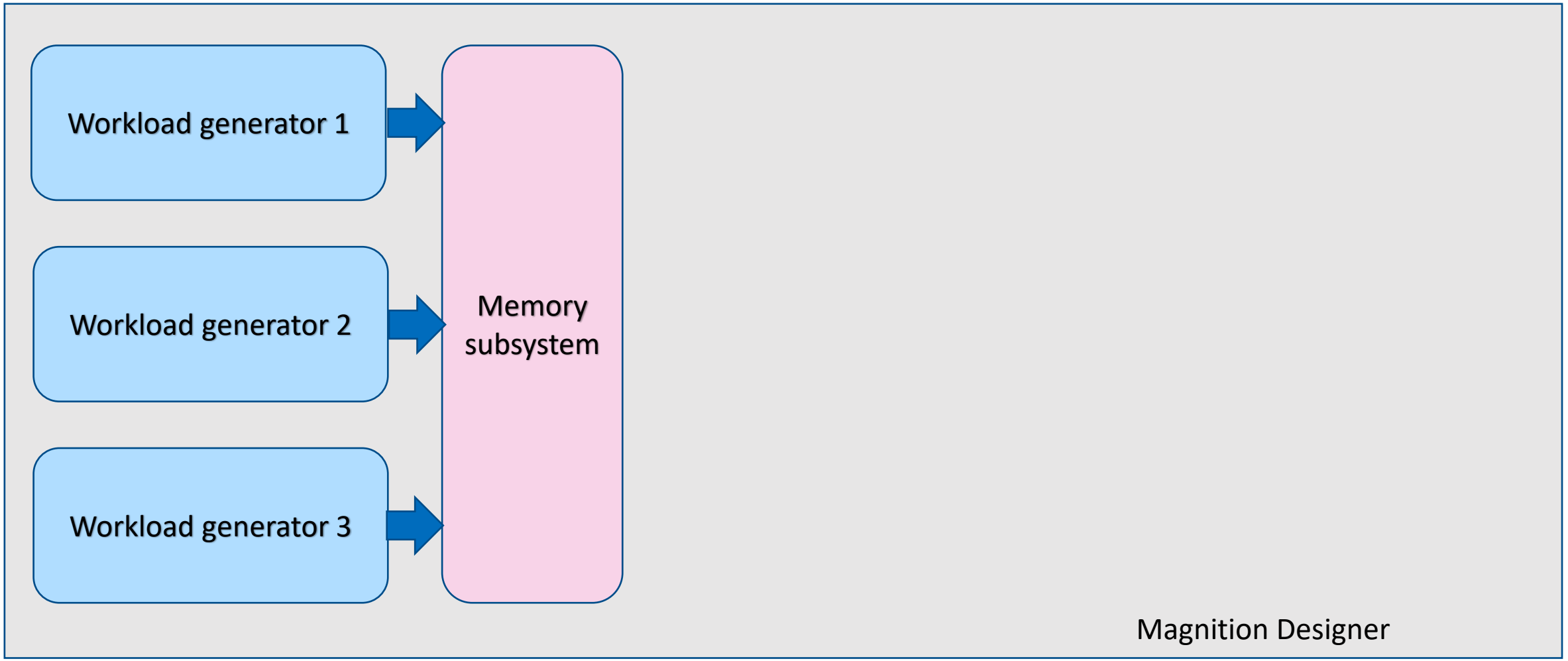
How to Build a Test Drive



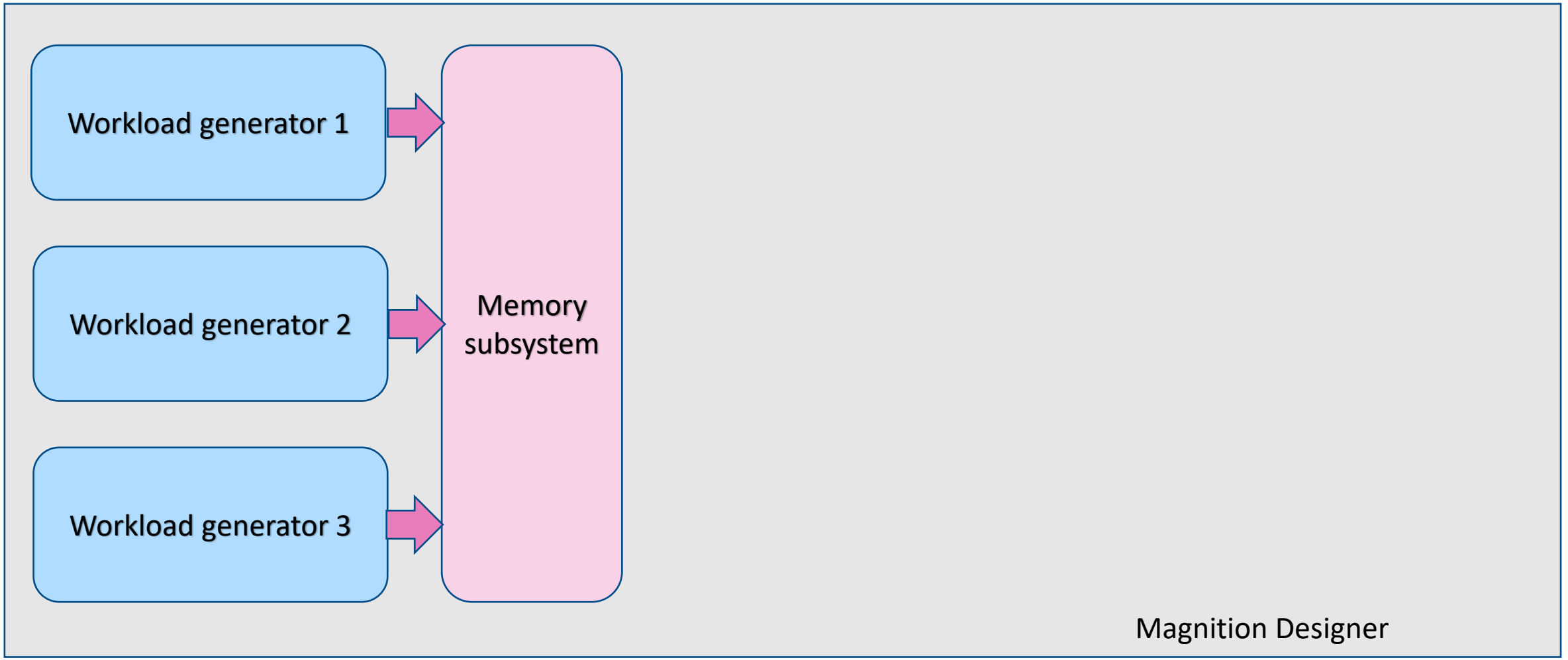
How to Build a Test Drive



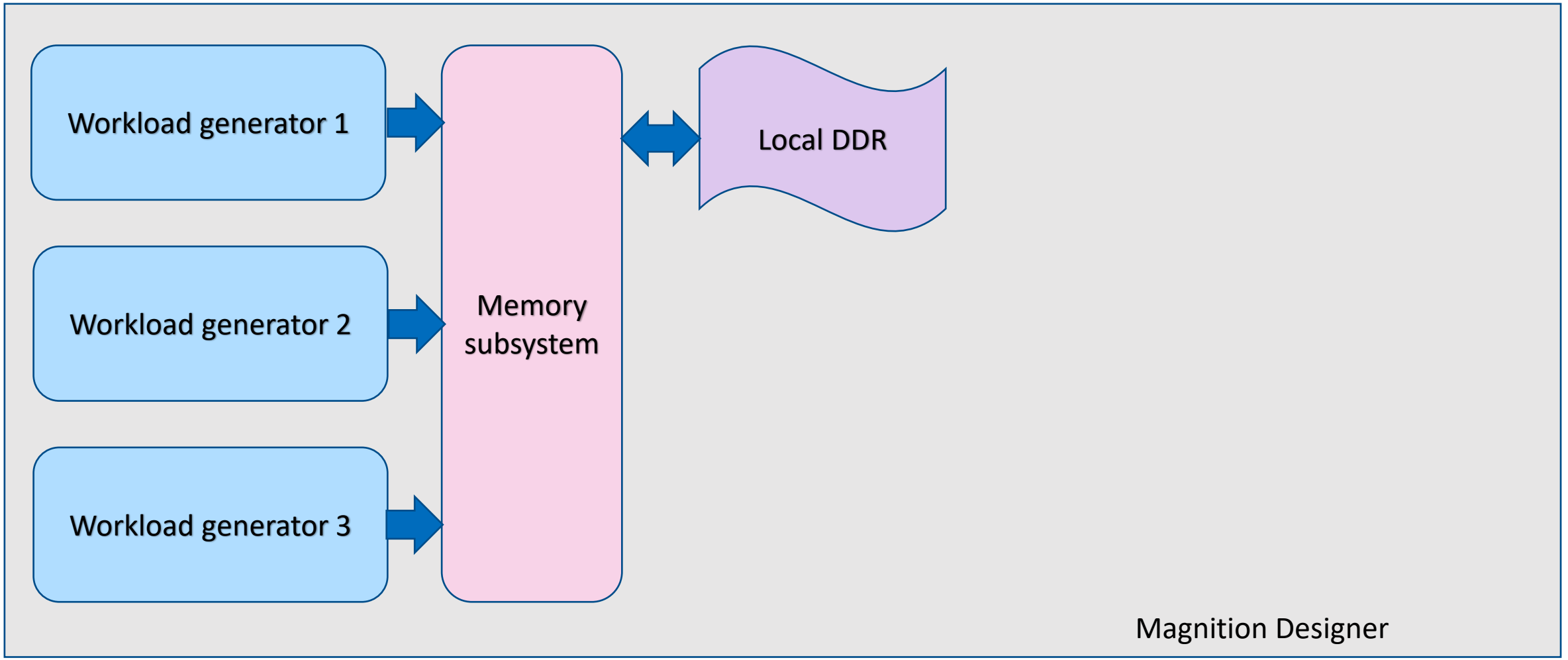
How to Build a Test Drive



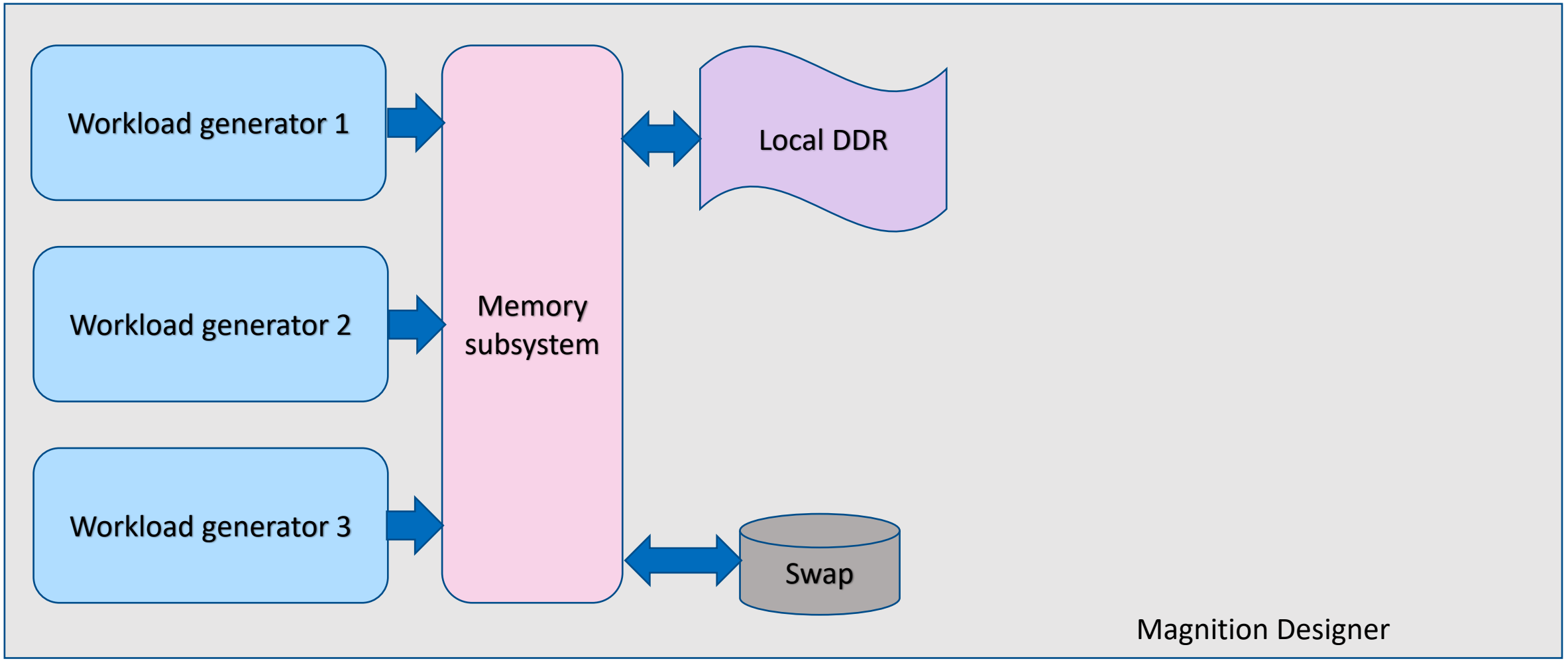
How to Build a Test Drive



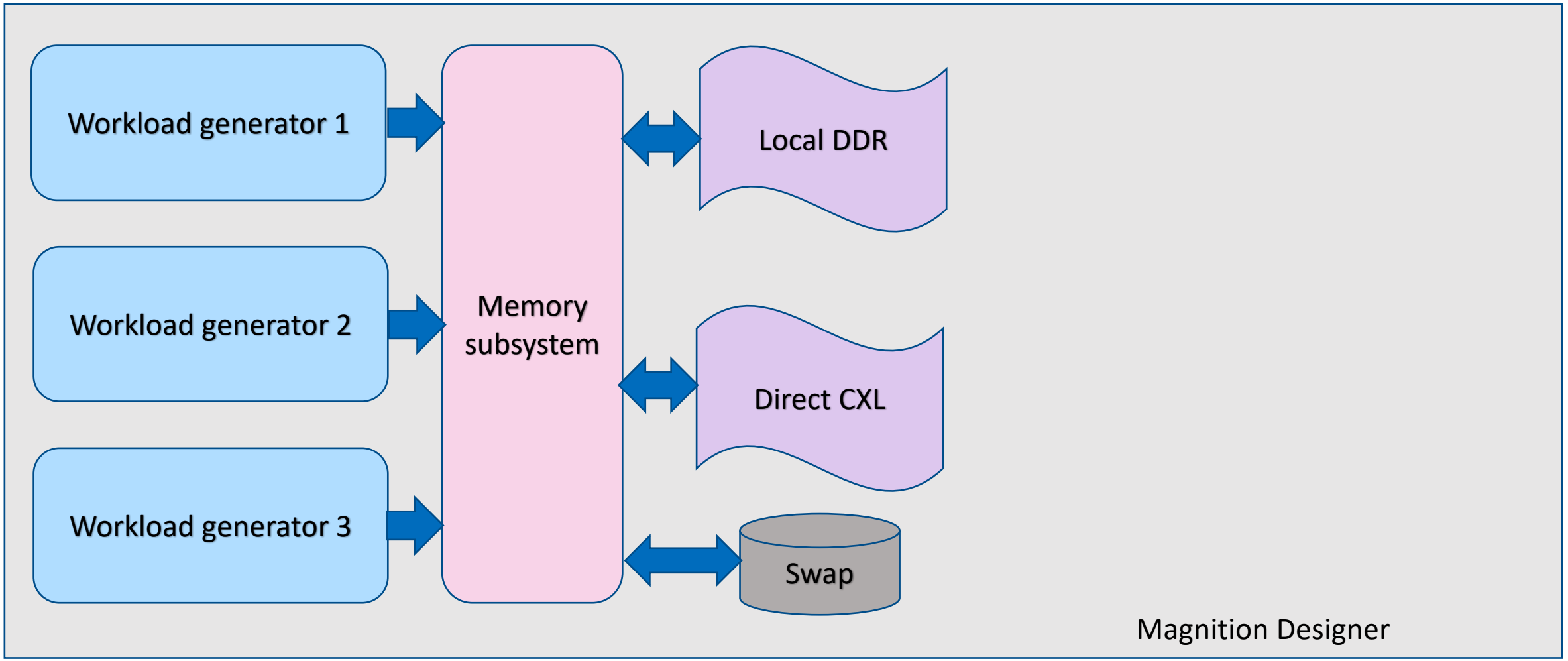
How to Build a Test Drive



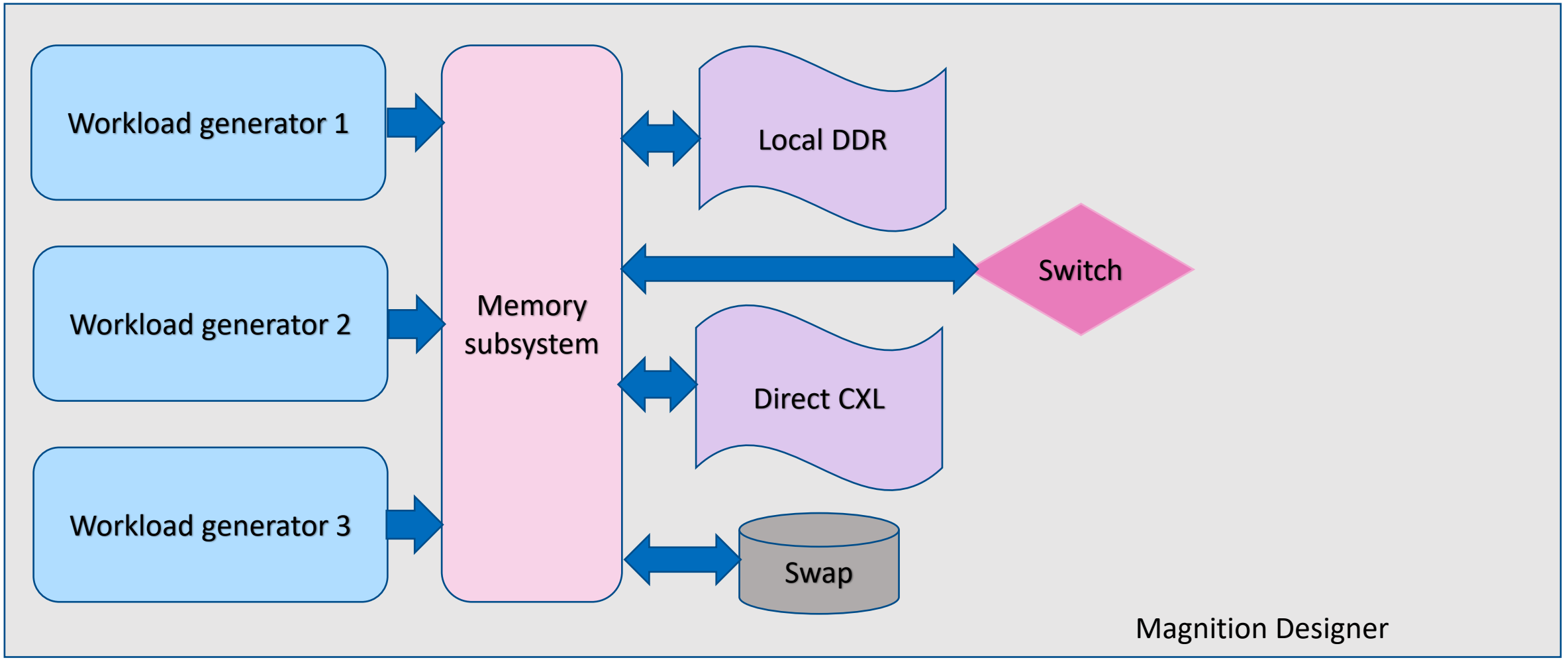
How to Build a Test Drive



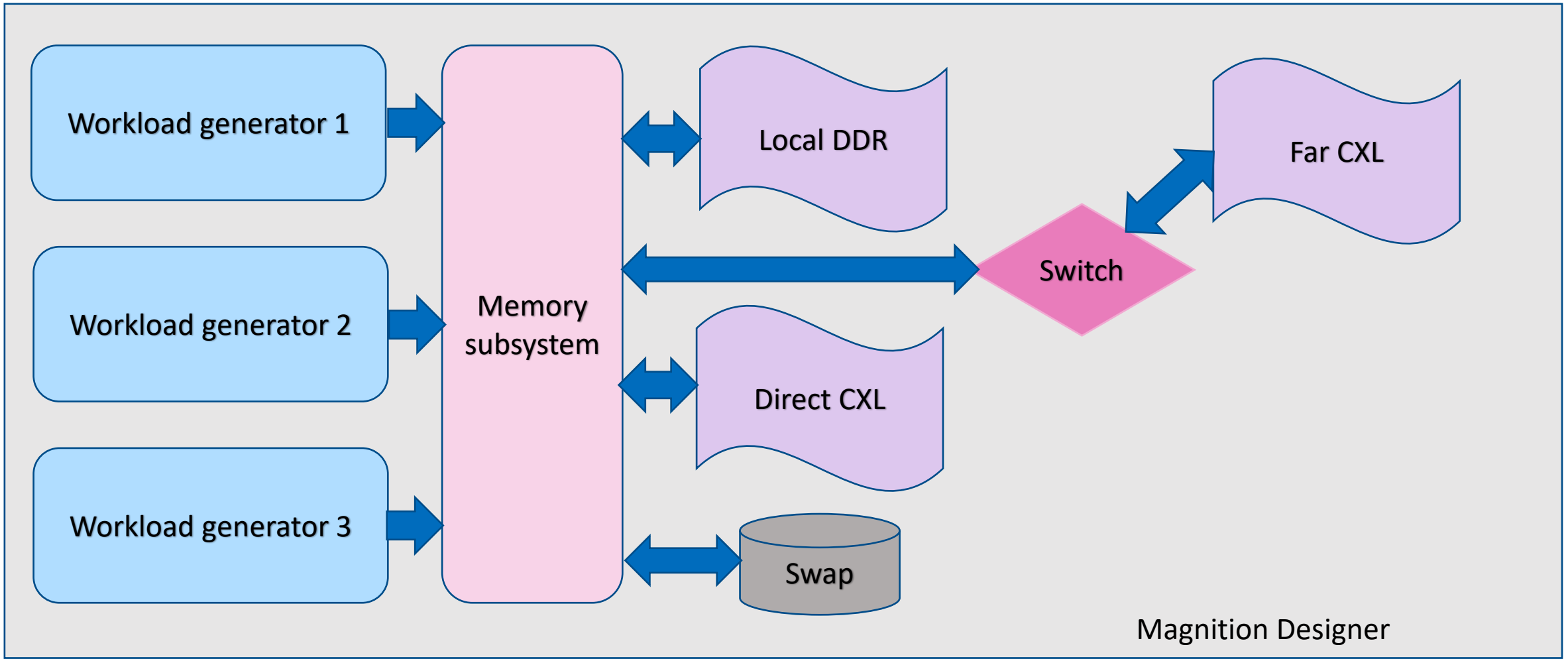
How to Build a Test Drive



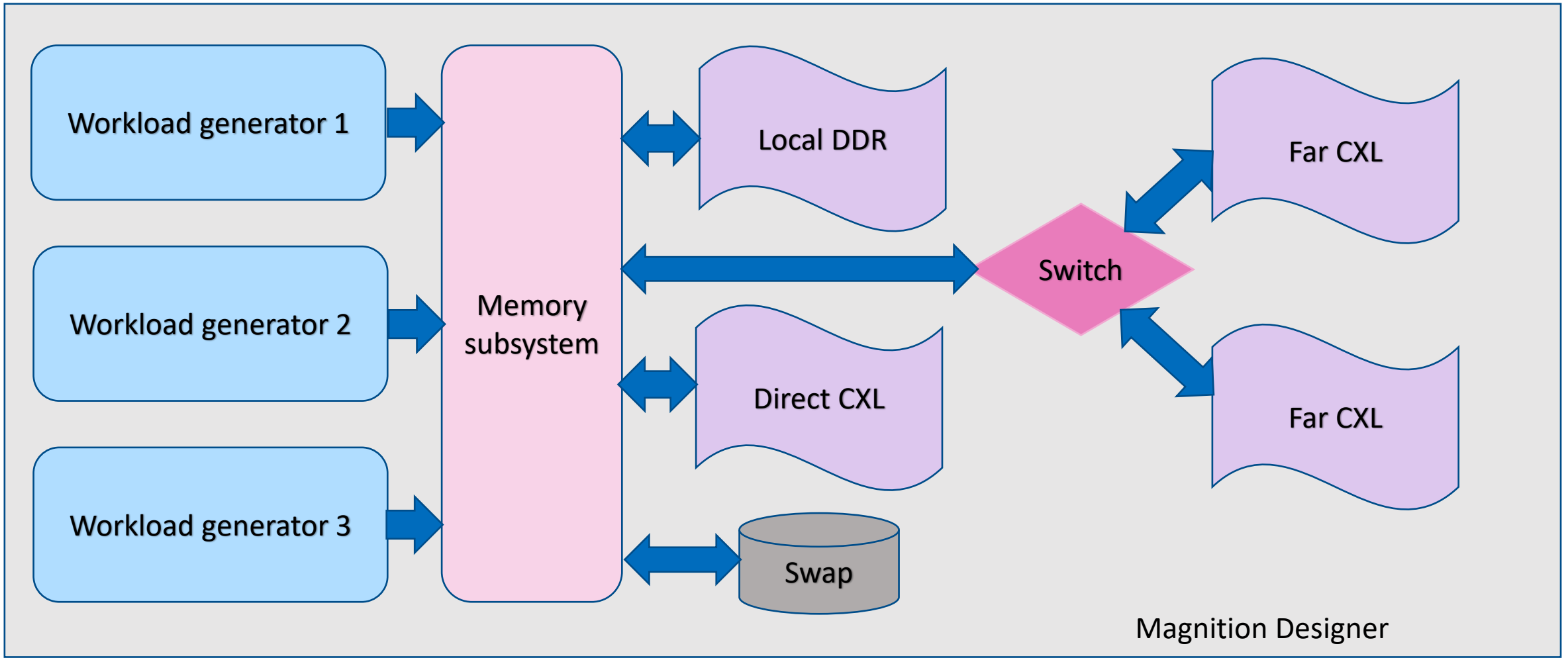
How to Build a Test Drive



How to Build a Test Drive



How to Build a Test Drive



Simulation Code Example

- Bullets 1
  - Bullets 2
    - Bullets 3
      - Bullets 4
        - Bullets 5

Graphical Representation of Code

- Bullets 1
  - Bullets 2
    - Bullets 3
      - Bullets 4
        - Bullets 5

Configuration File

- Bullets 1
  - Bullets 2
    - Bullets 3
      - Bullets 4
        - Bullets 5

Working at Enterprise Scale

- 100+ cores
- 100+ PCIe lanes
  - CXL capabilities
  - Network capabilities
  - Not limited to memory or IO bound loads
- Clusters of numerous nodes

picture

Experiment 1

- 3 applications with roughly the same memory footprint
- Each application runs independently on a system with local DDR and swap
- Combinations of 2 applications run simultaneously on a system with local DDR and swap
- All 3 applications run simultaneously on a system with local DDR and swap

Experiment 2

- 3 applications with roughly the same memory footprint
- All 3 applications run simultaneously on a system with local DDR, local CXL and far CXL (and unused swap), several times each each
- All memory treated as flat access
- Repeat experiment with different ordering and intervals between start times

Consistency is the Hobgoblin of Performance Engineers

- Applications can be assigned to particular NUMA nodes
- Node-picking in Linux with numactl(8)
  - Process level
  - --cpunodebind
  - --membind
  - --localalloc
  - --preferrednode
  - --interleave

Experiment 3

- 3 applications with roughly the same memory footprint
- All 3 applications run simultaneously on a system with local DDR, local CXL and far CXL (and unused swap), several times each
- Different abstract distance values applied to different applications
- Repeat experiment with different ordering and intervals between start times

Today I Learned Something

- CXL offers a lot of flexibility *and complexity*
- CXL innovation is outpacing hardware deliveries
- CXL switching is being developed and improved by emulating against the standard
- Varied applications can be optimized in simulated large-scale CXL environments

Please take a moment to rate this session.

Your feedback is important to us.

