Rack Scalable OS for The Machine and the Case for Capabilities

Dejan Milojicic, Hewlett Packard Labs

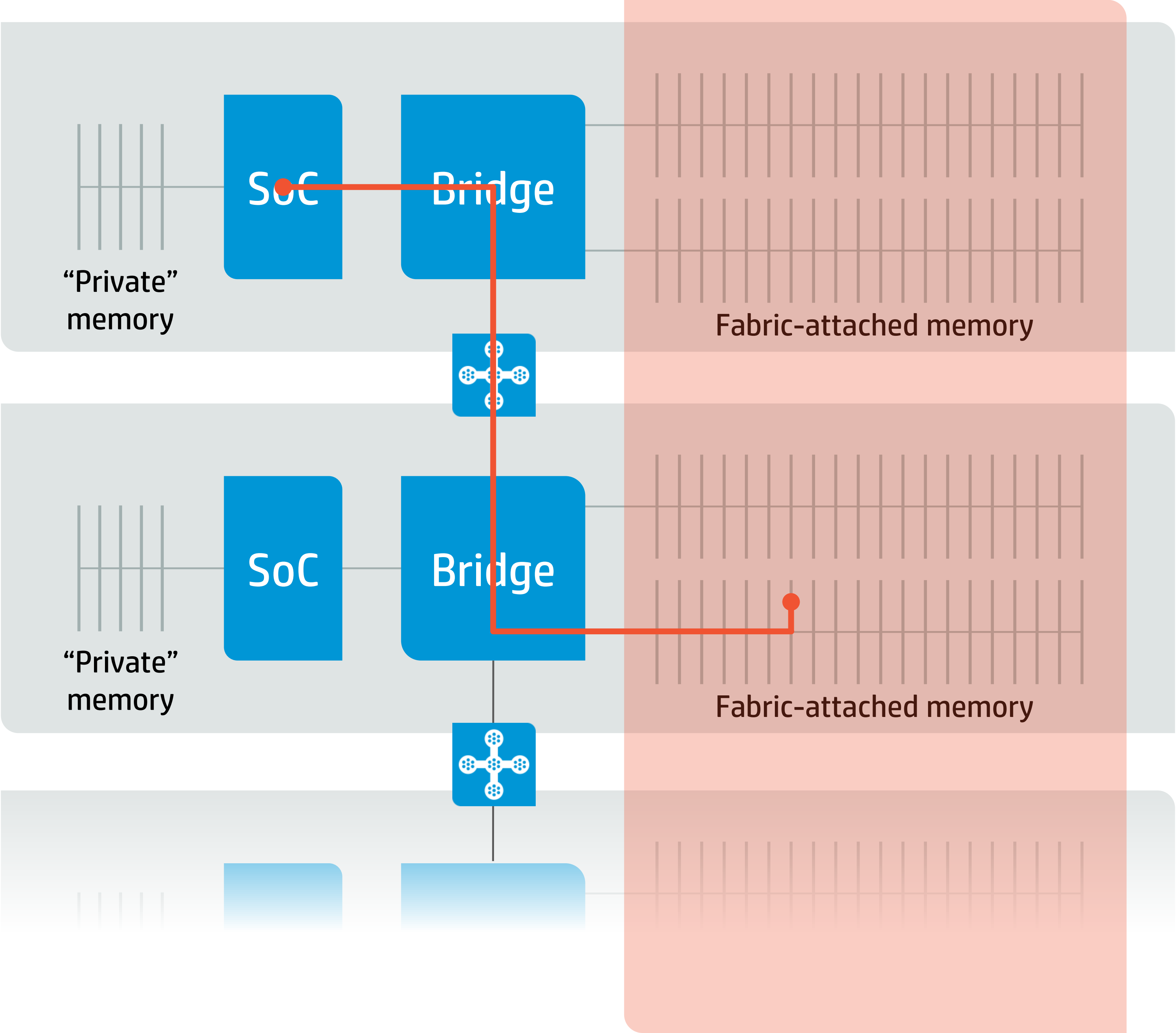
(The First?) CHERI Microkernel Workshop
Cambridge University, April 23rd, 2016

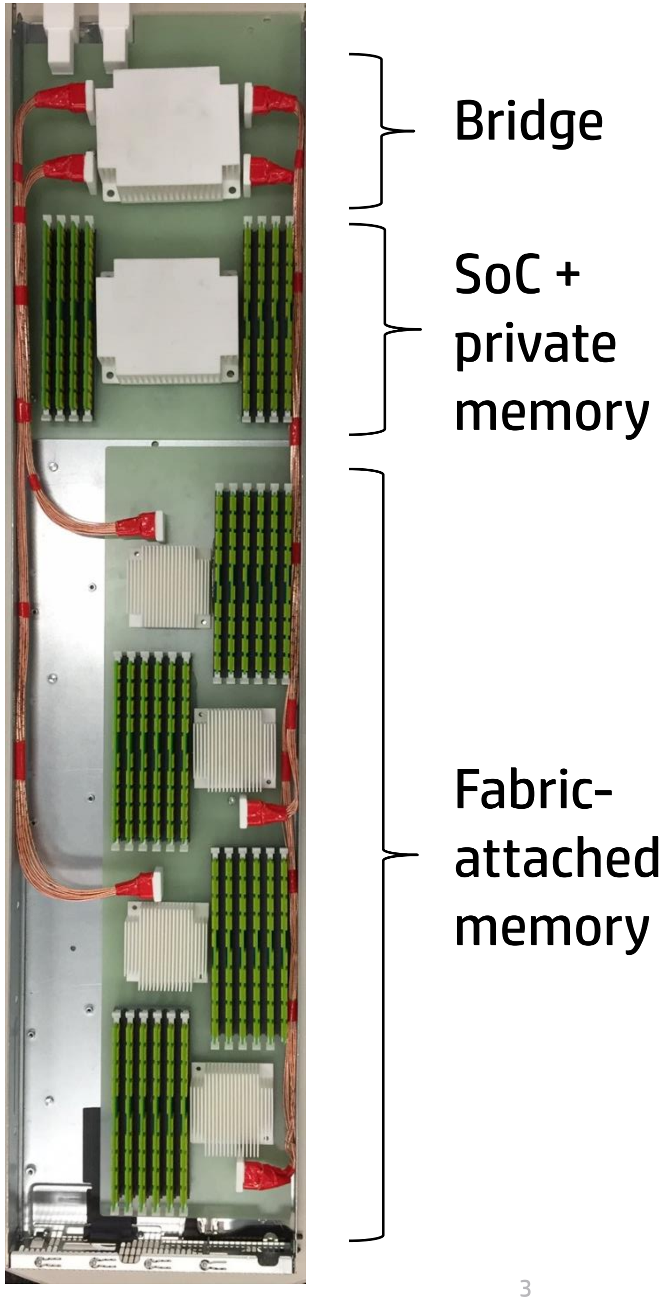
Memory-centric rack-scale architectures

Intel
Rack Scale
Architecture

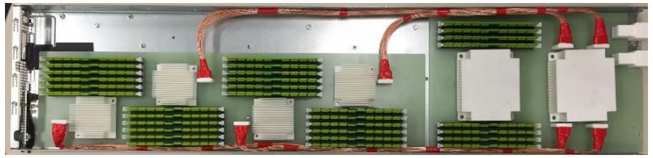
UC Berkeley
Firebox

HP
The Machine


SoC
Bridge
"Private" memory
Fabric-attached memory
SoC
Bridge
"Private" memory
Fabric-attached memory


Bridge
SoC + private memory
Fabric-attached memory

Prototype of The Machine



Node:

- SoC
- Local “private” memory
- Bridge to memory fabric
- Fabric-attached memory
- Ethernet

Enclosure:

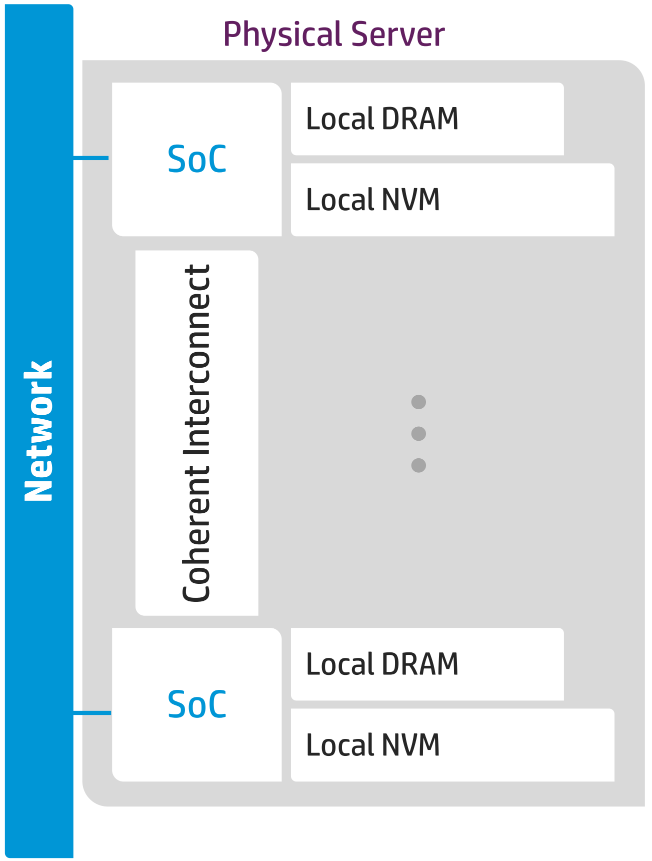
- 10 nodes in 5U enclosure

Rack:

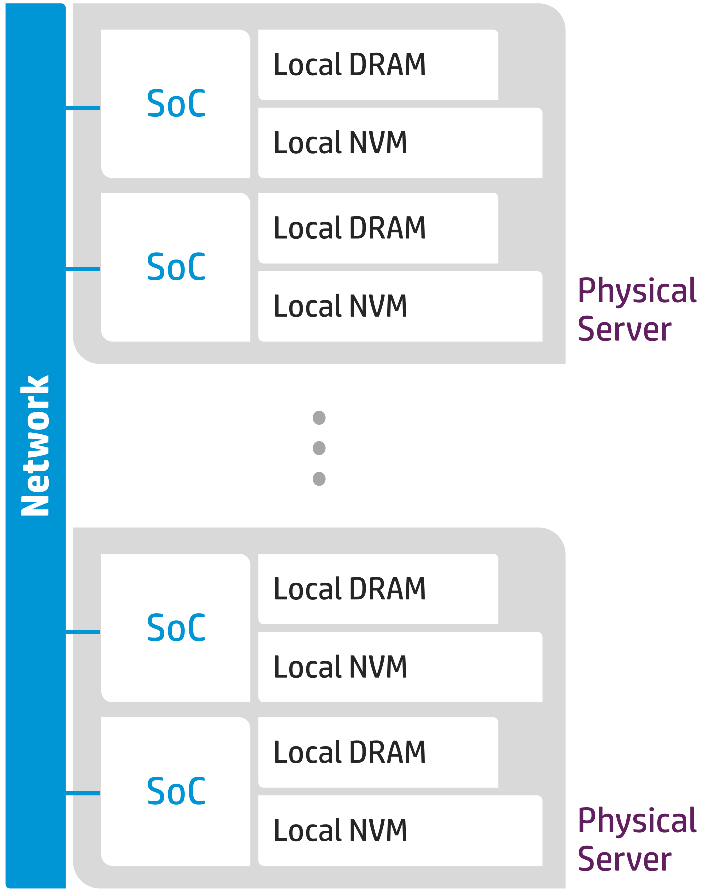
- 8 enclosures
- **320 TB fabric-attached memory**
- **80 SoCs**

- No cache coherence between SoCs
- Explicit software coherence model aided by
  - Synchronization features designed into fabric
  - Libraries

Traditional system architectures



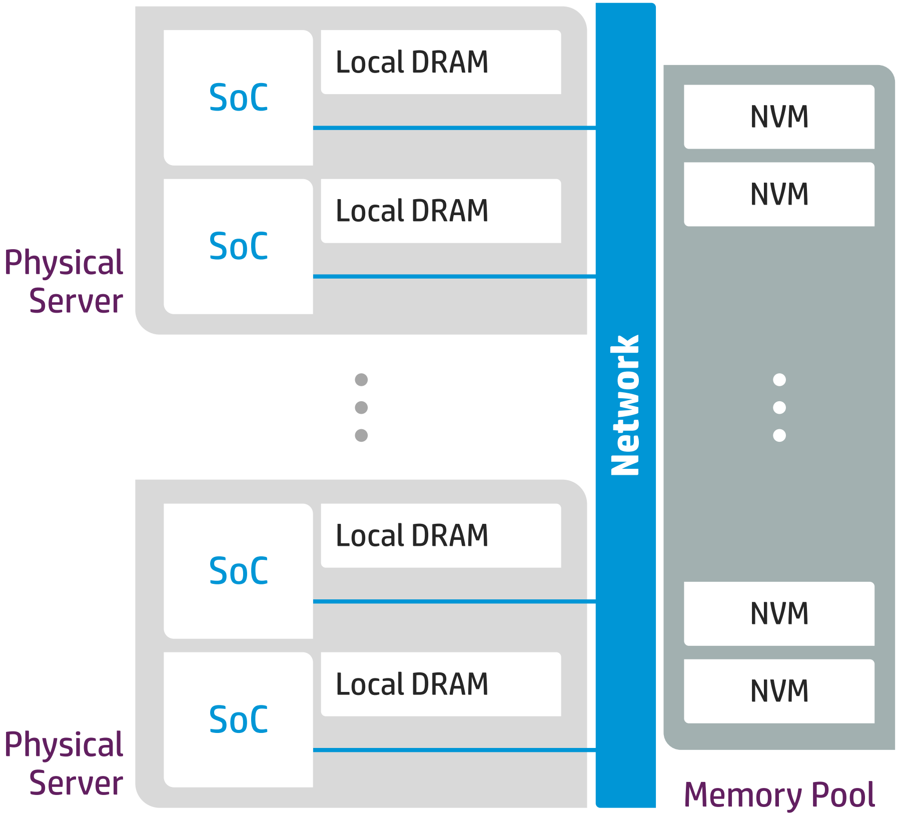
Shared everything



Shared nothing

Future memory-centric architecture

Shared something



Converging memory and storage

Byte-addressable non-volatile memory (NVM) replaces hard drives and SSDs

Shared memory pool

NVM pool is accessible by all compute resources

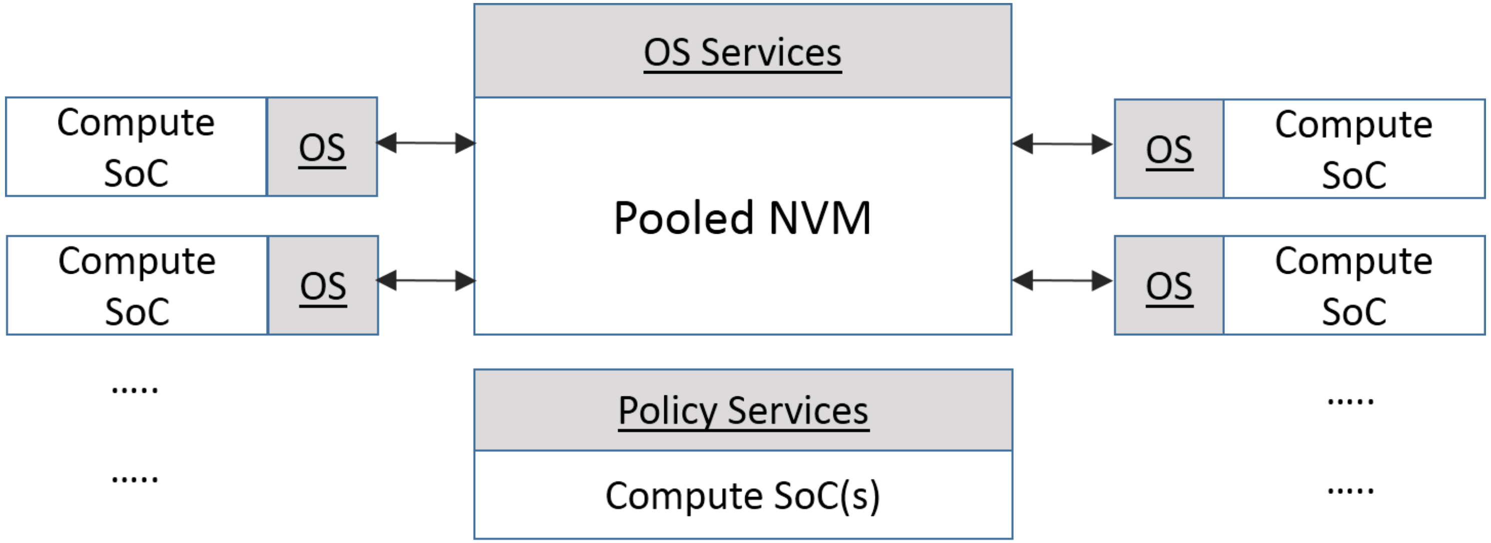
Optical networking advances provide near-uniform latency

“Private” memory provides lower-latency “performance tier”

Heterogeneous compute resources distributed closer to data

Hewlett Packard Enterprise

Distribution of memory management functionality



Memory management functions move from processor-centric OS to distributed services

Allocation, protection, synchronization, de/encryption, (de)compression, error handling

Policy services: quotas, QoS

Cluster: memory-side controllers, accelerators and more novel computational elements

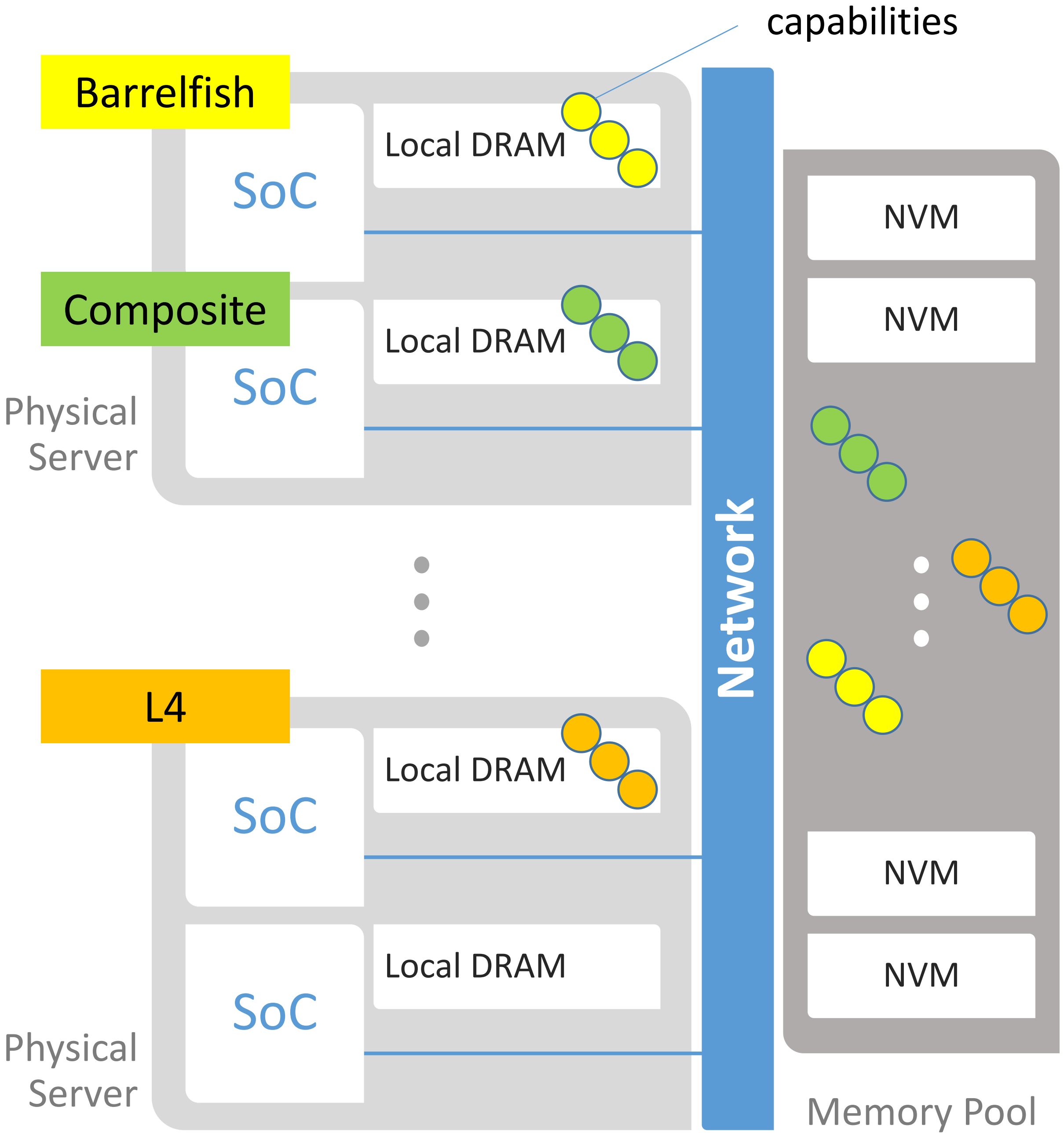
Motivation for CHERI Capabilities

CHERI Capabilities

- Separate translation from protection in OSes through hardware-software co-design
- Serve as non-forgeable handles to access memory
- Have tremendous potential for fine grain security, eliminating viruses/bugs

The Machine has vast amounts of memory

- Need to manage it (allocate, free, deal with failures, etc.)
- Programmatic access
- Need to share it
- Need to protect it


capabilities
Barrelfish
SoC
Local DRAM
Composite
SoC
Local DRAM
Physical Server
L4
SoC
Local DRAM
SoC
Local DRAM
Physical Server
Network
NVM
NVM
NVM
NVM
Memory Pool

What is needed

Our priorities

- Persistency
- Kernel compartmentalization
- Opportunities for the memory side management functions

A series of research efforts and development experiments needed

- Making L4 kernel capabilities persistent (in progress with Dresden)
- Making CHERI capabilities persistent (Alex's work)
- Exploring CHERI support for kernel capabilities (in progress here and elsewhere)
- Distributed capabilities (non-trivial task)
- Exploring global memory and interconnect support for CHERI (big task)

Thank you

Dejan Milojicic
dejan.milojicic@hpe.com

Q&A