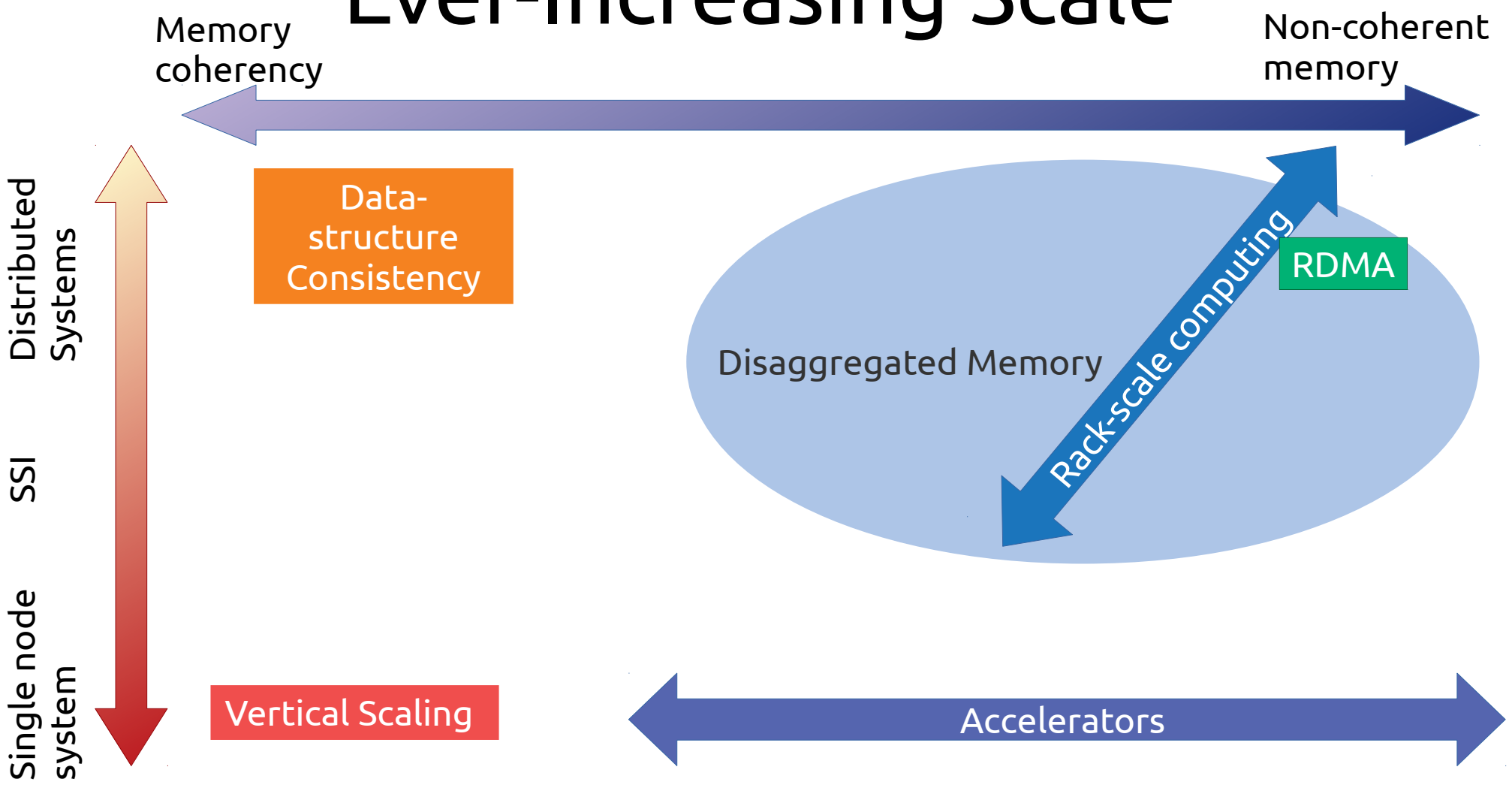
Ch'i: *Scaling Microkernel Capabilities* in *Cache-Incoherent Systems*

Yuxin Ren[*], *Gabriel Parmer*[*], Dejan Milojicic[†]

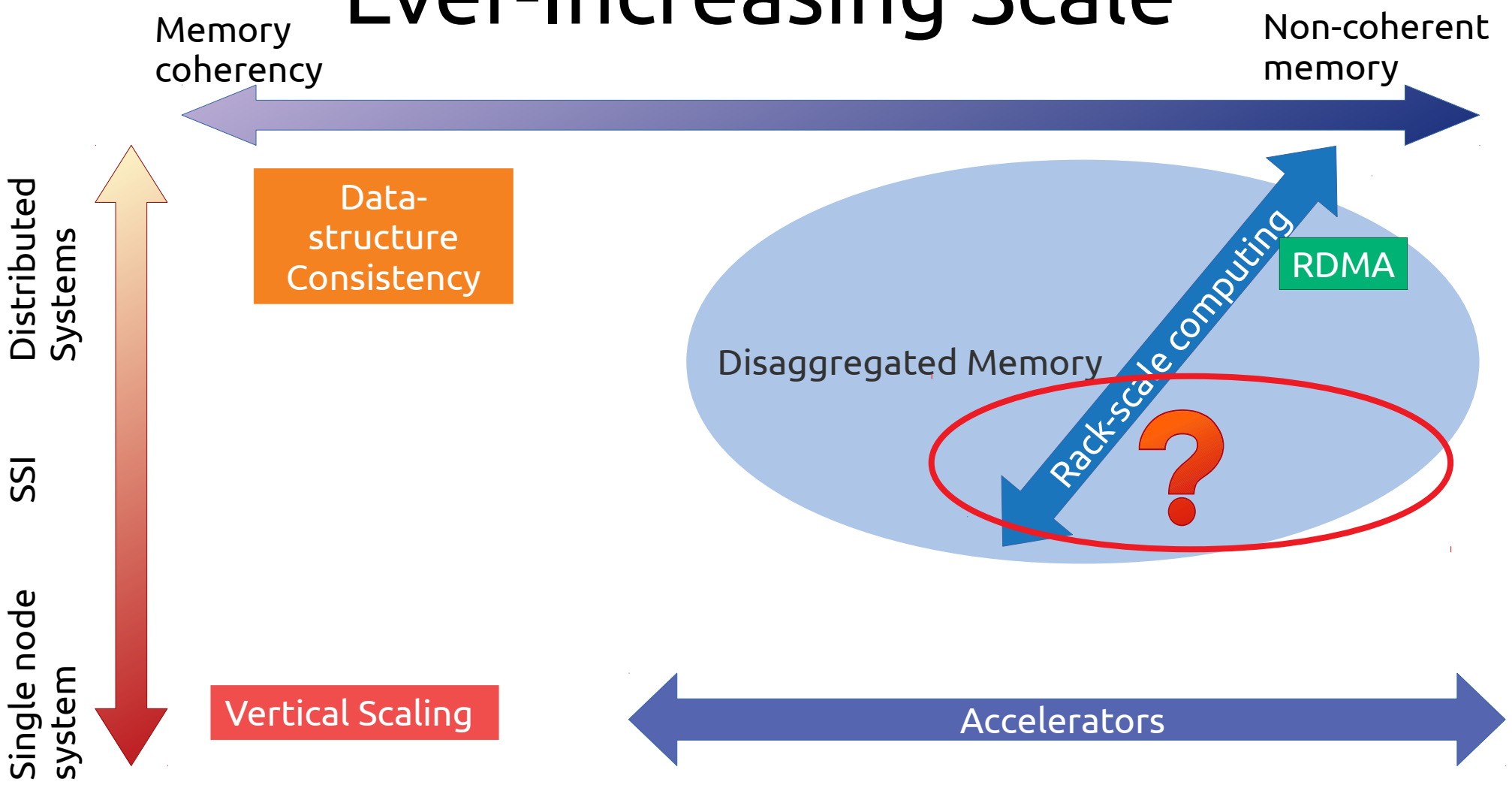
[*] GW

[†] Hewlett Packard Enterprise

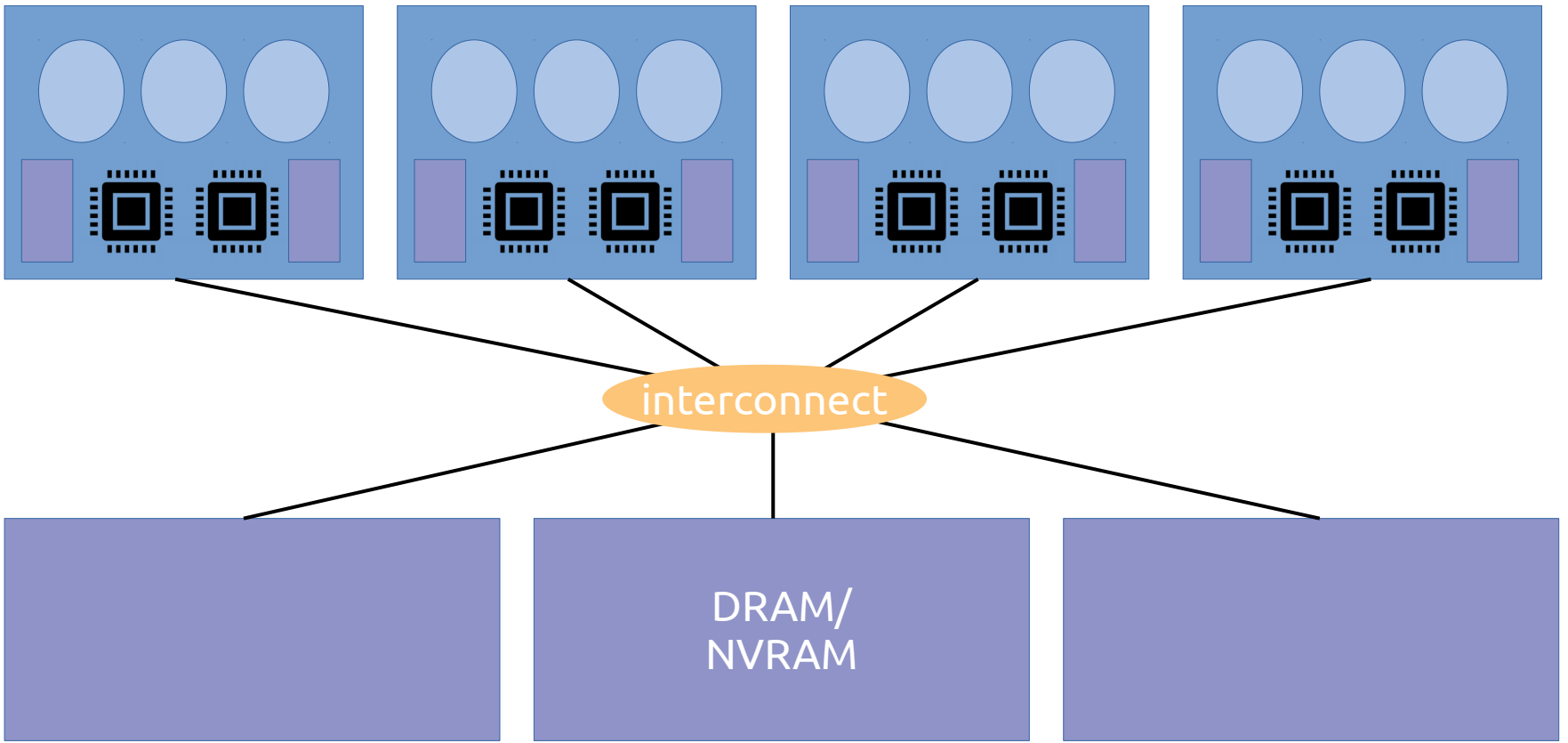
Ever-Increasing Scale



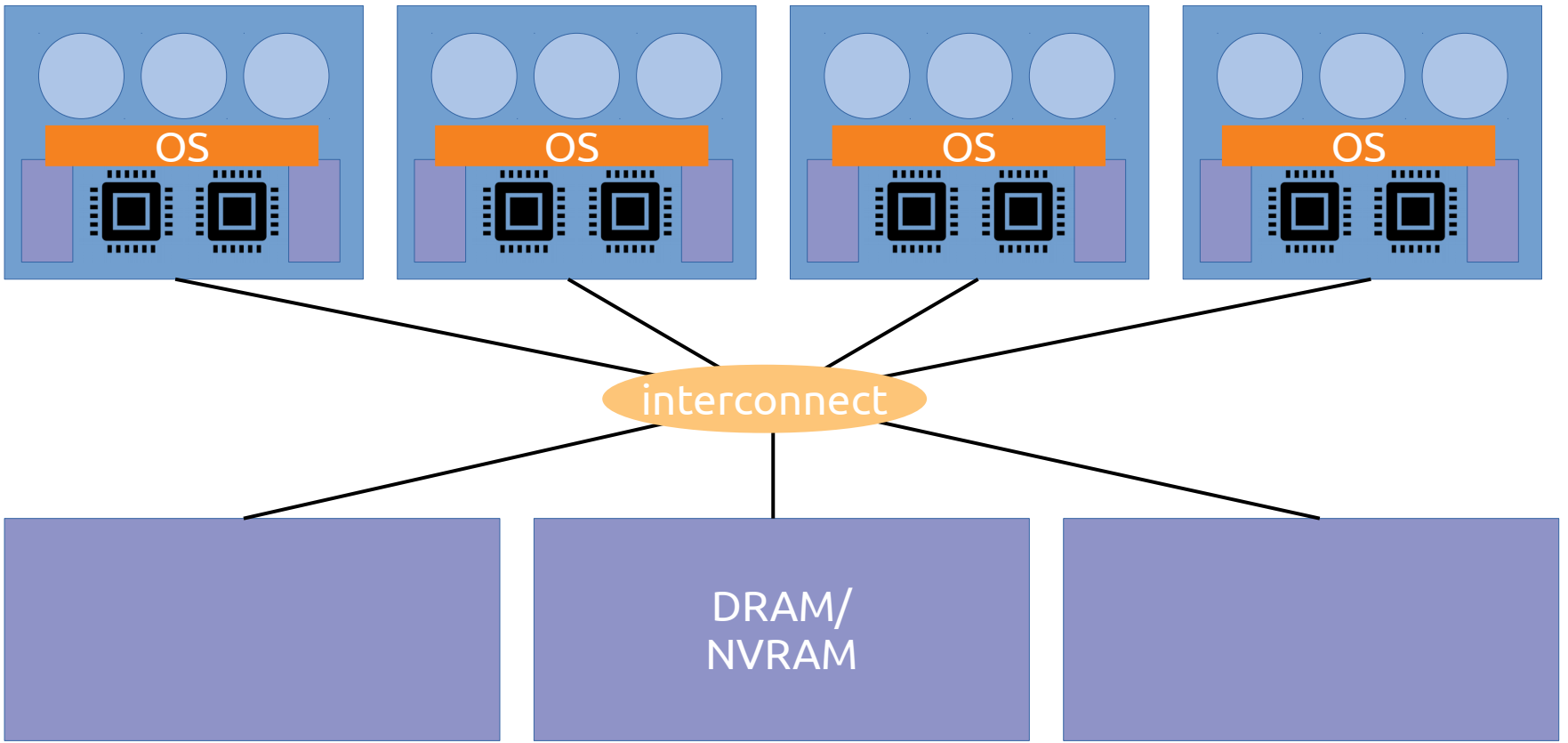
Ever-Increasing Scale



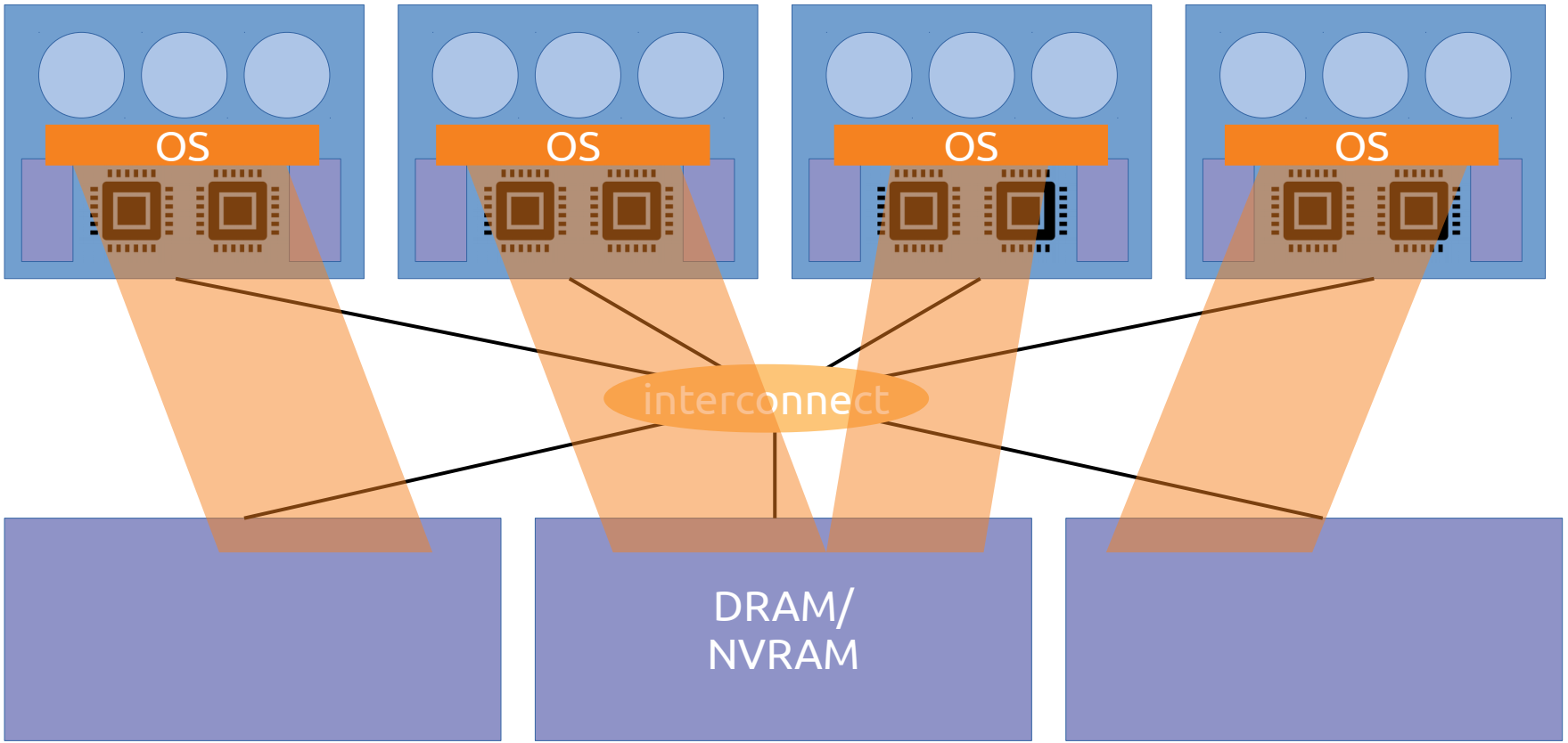
HW w/ Disaggregated Mem.



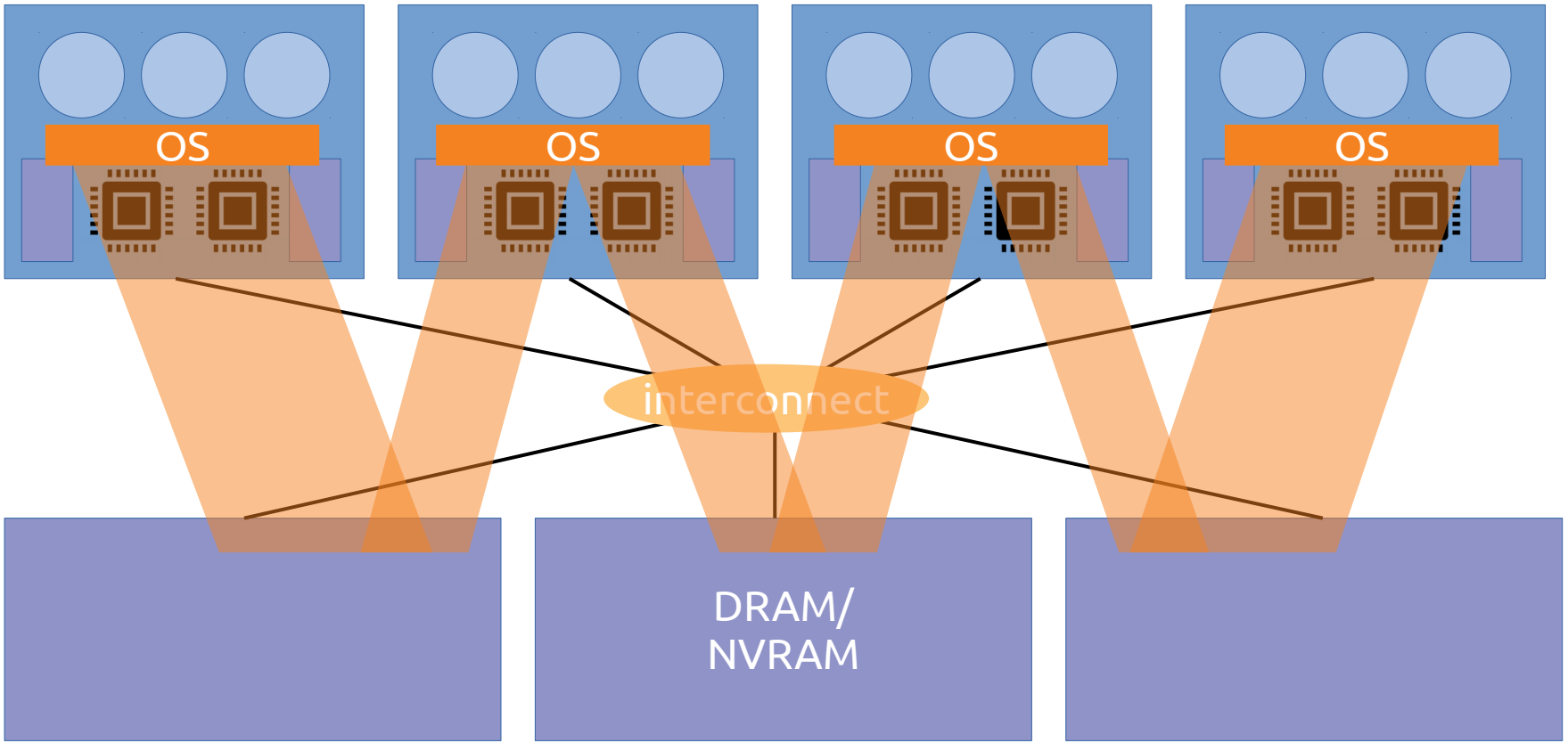
OS Support for Disaggregated Mem.



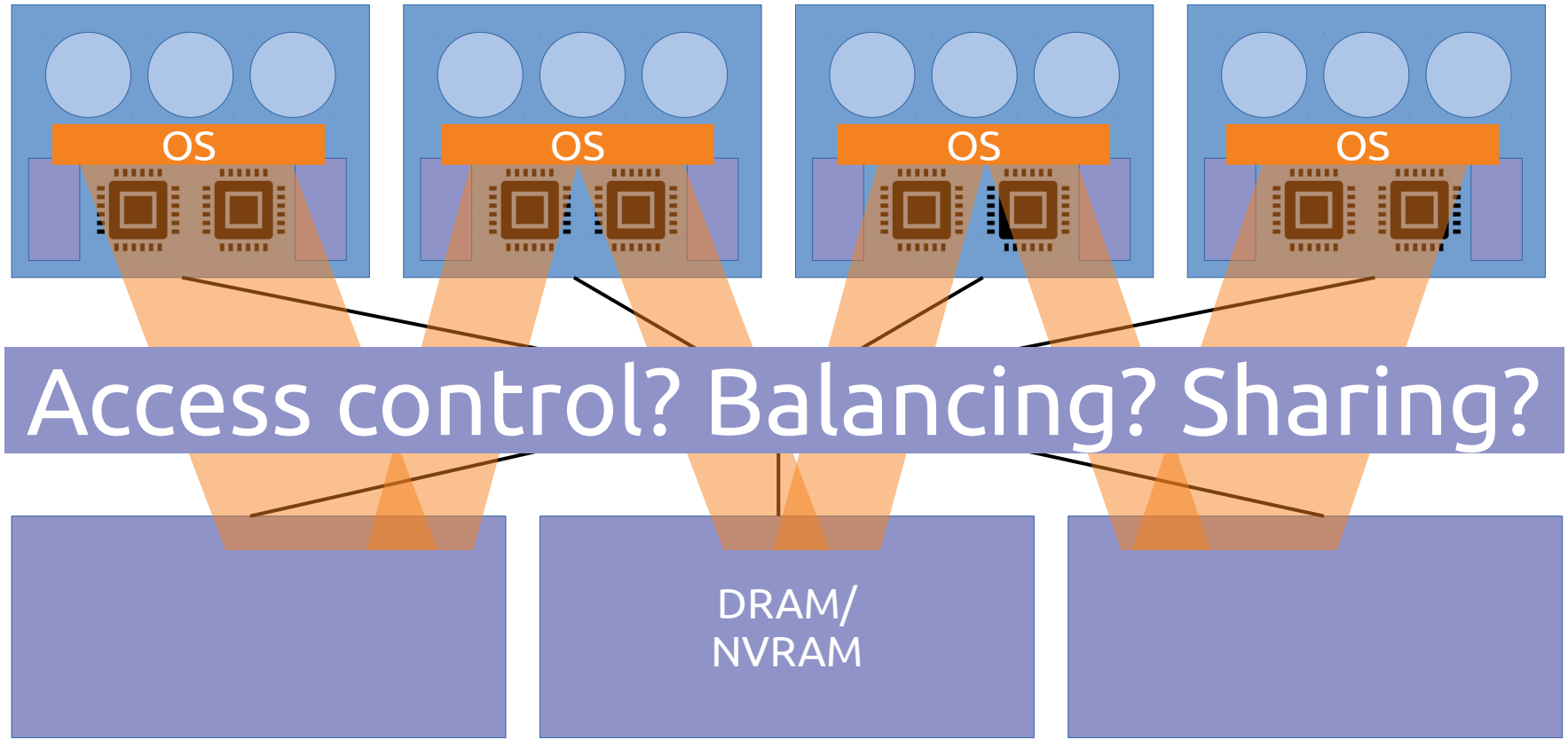
OS Support: Hot-swappable Mem.



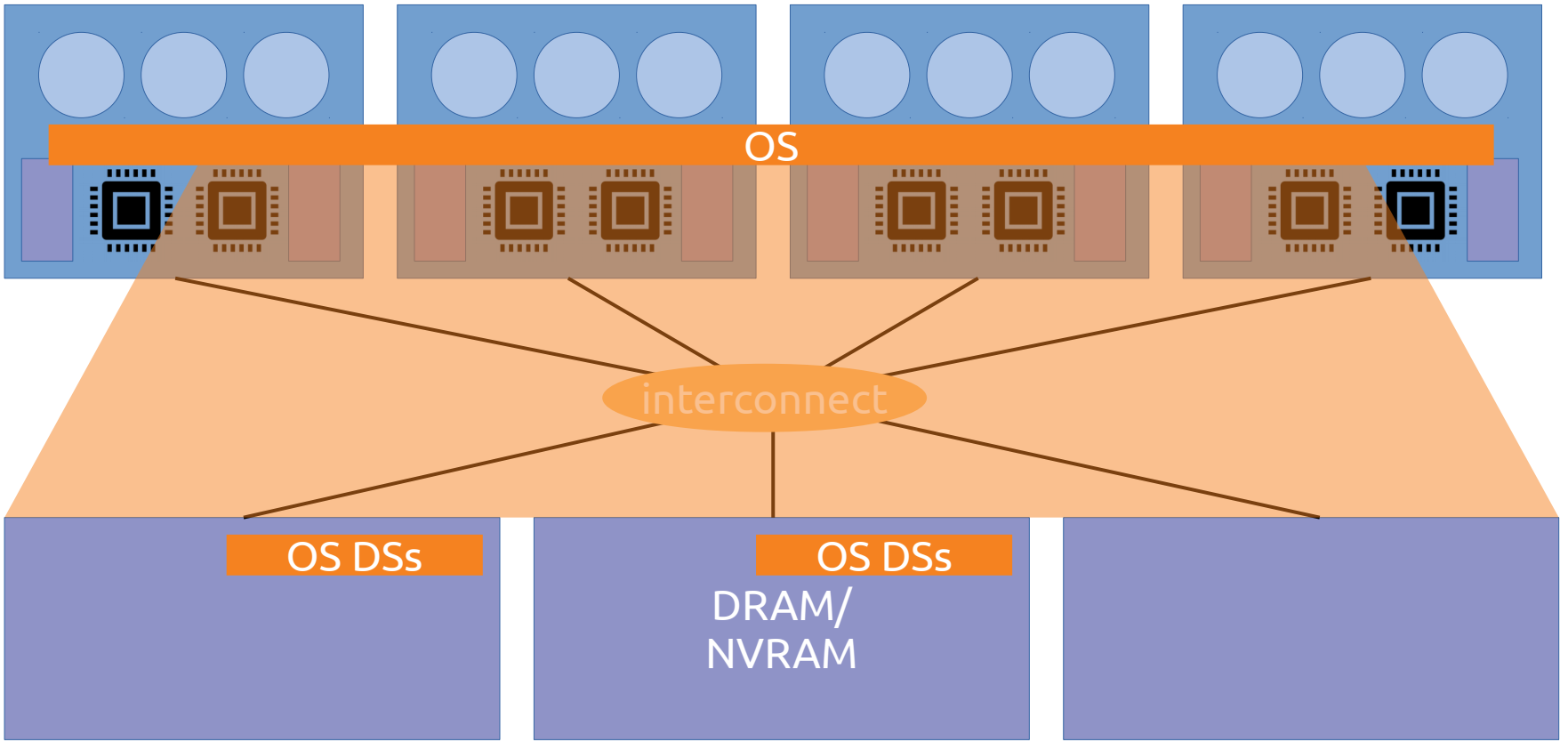
OS: Shared Hot-swappable Mem.



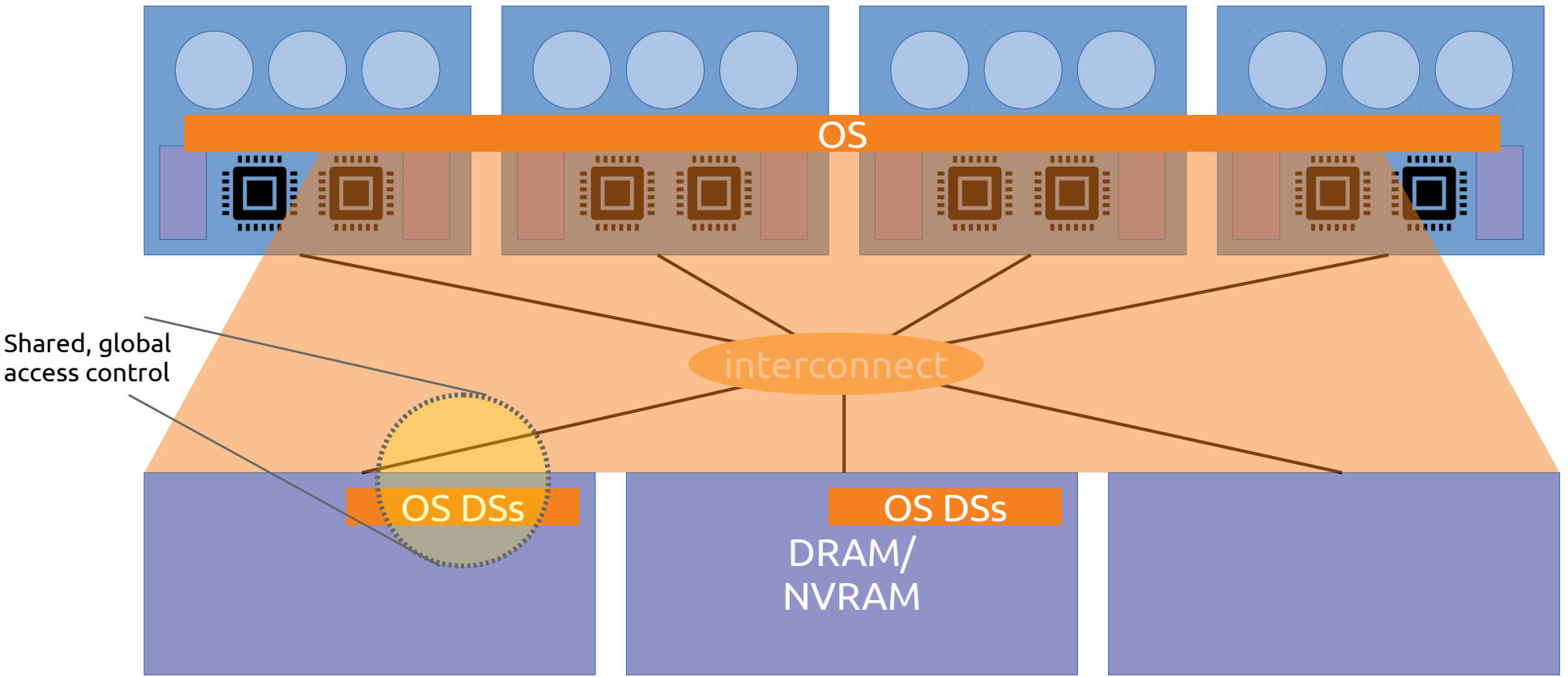
OS: Shared Hot-swappable Mem.



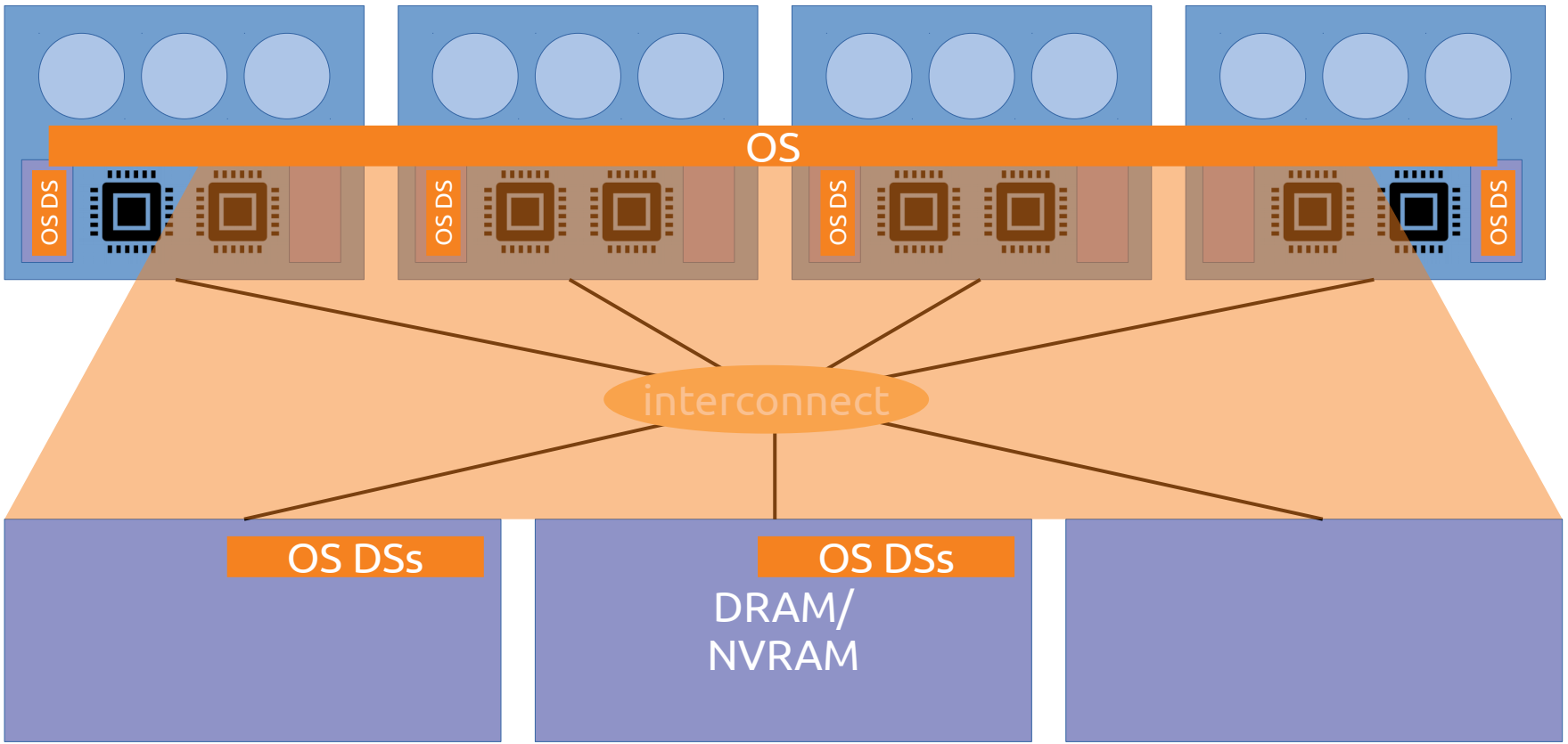
Single OS in Disaggregated Memory



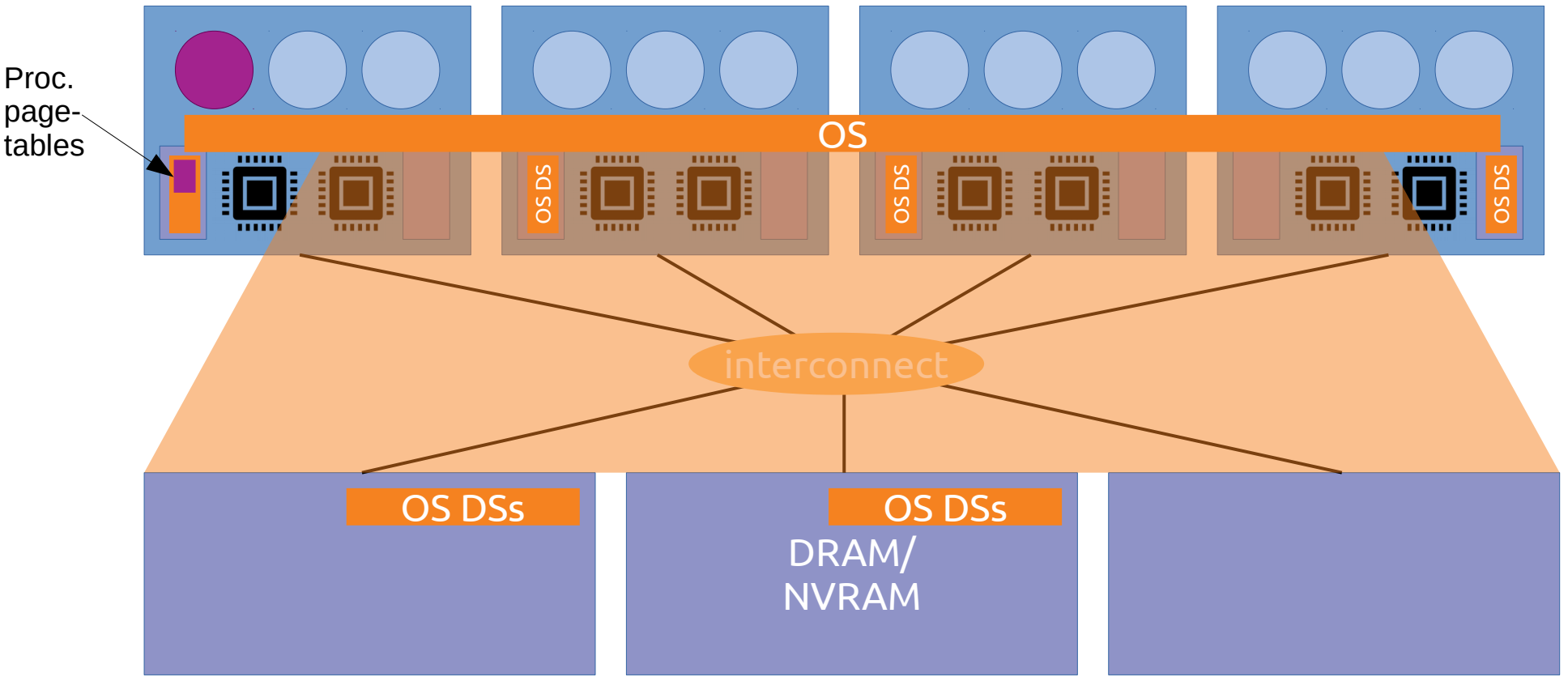
Global Access Control



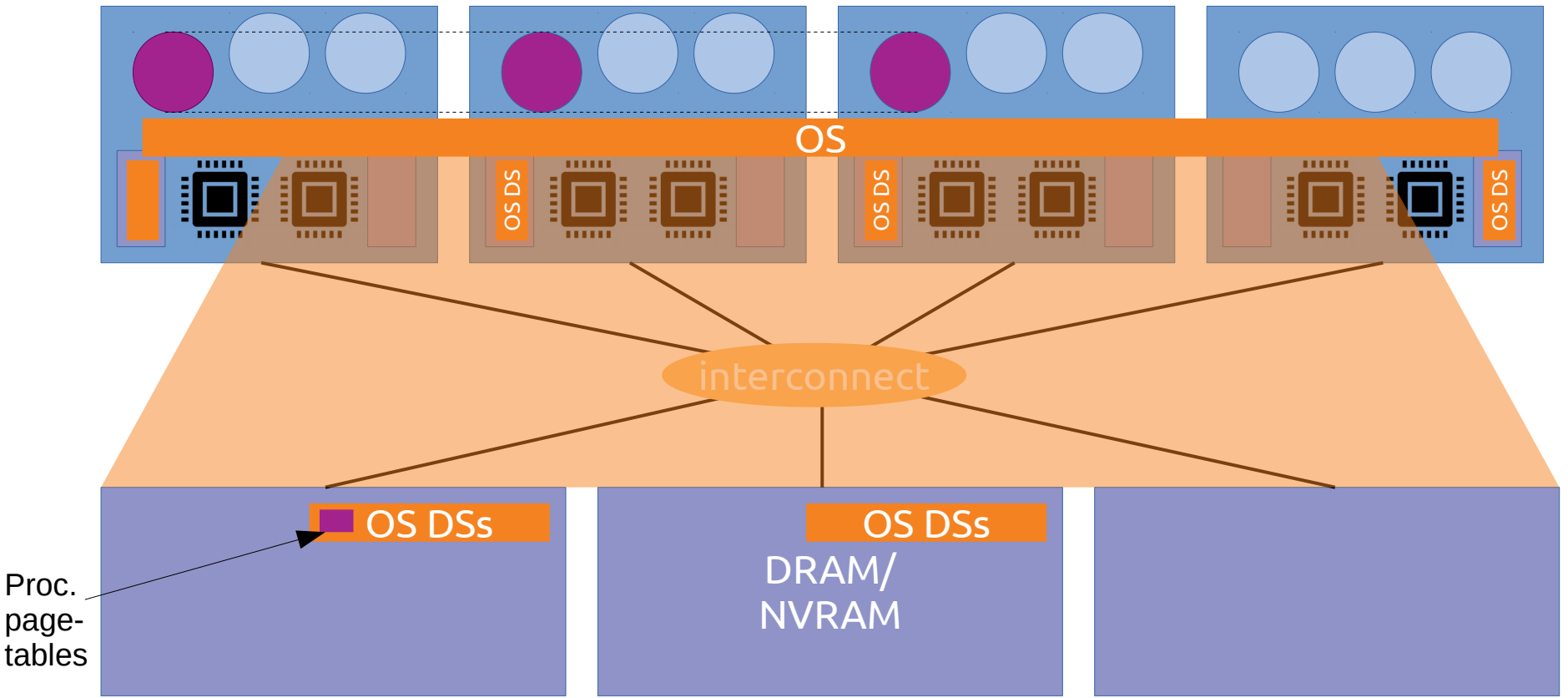
Local Management of Memory



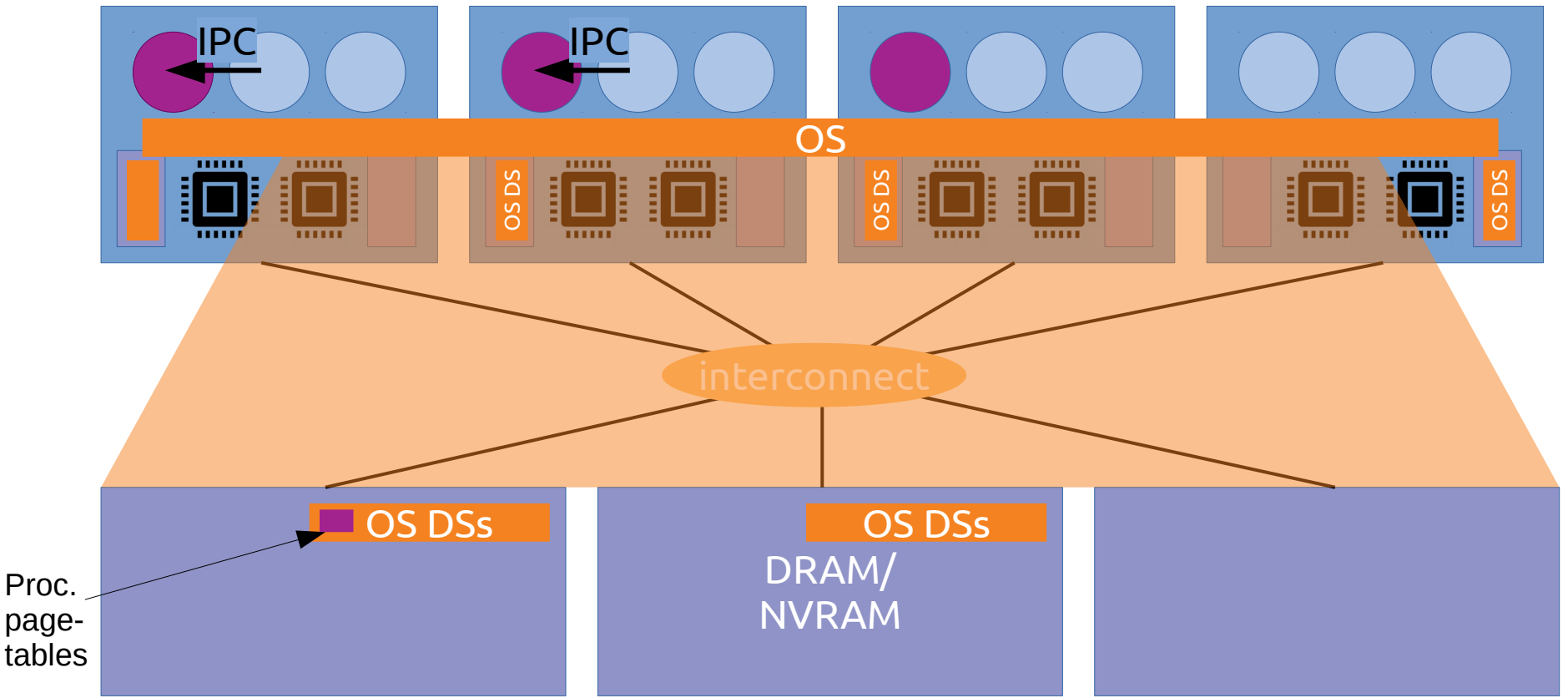
Local Computations



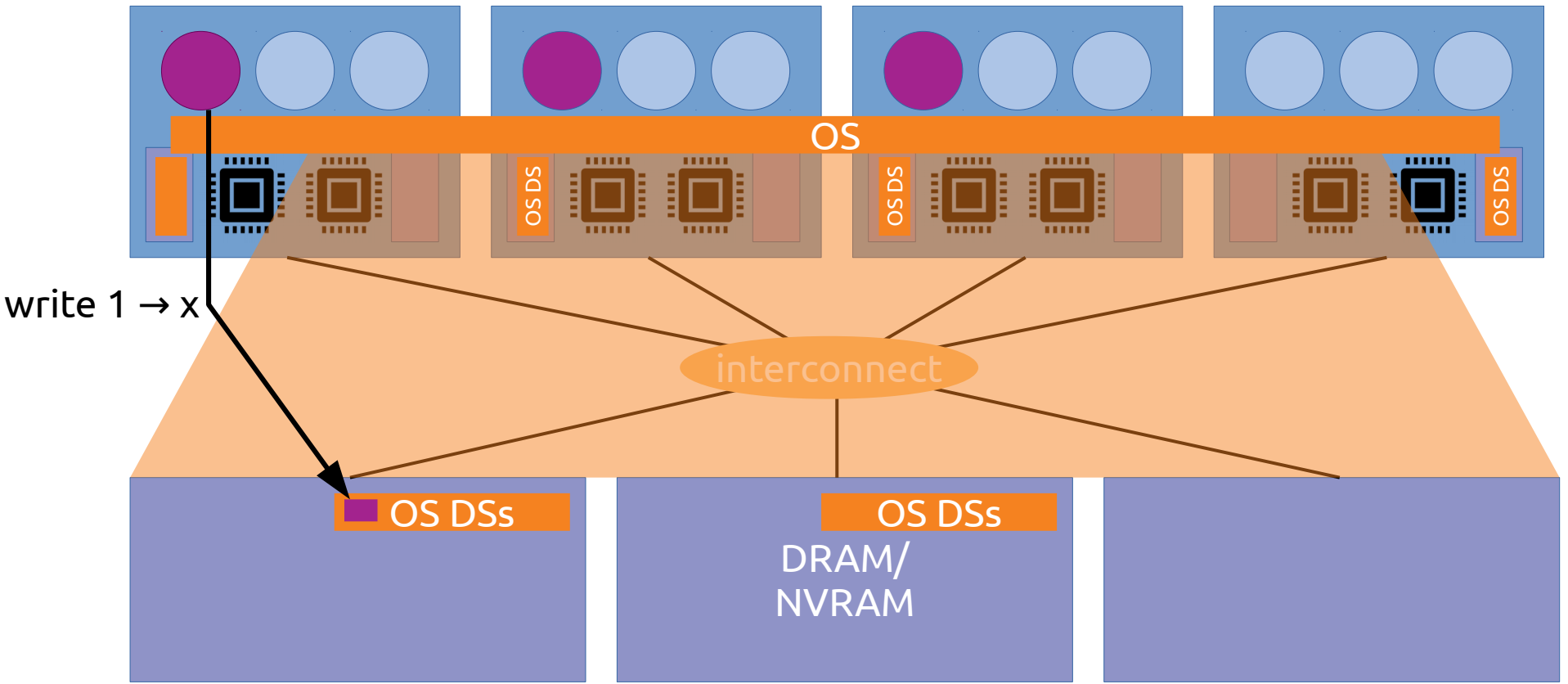
Shared Computations



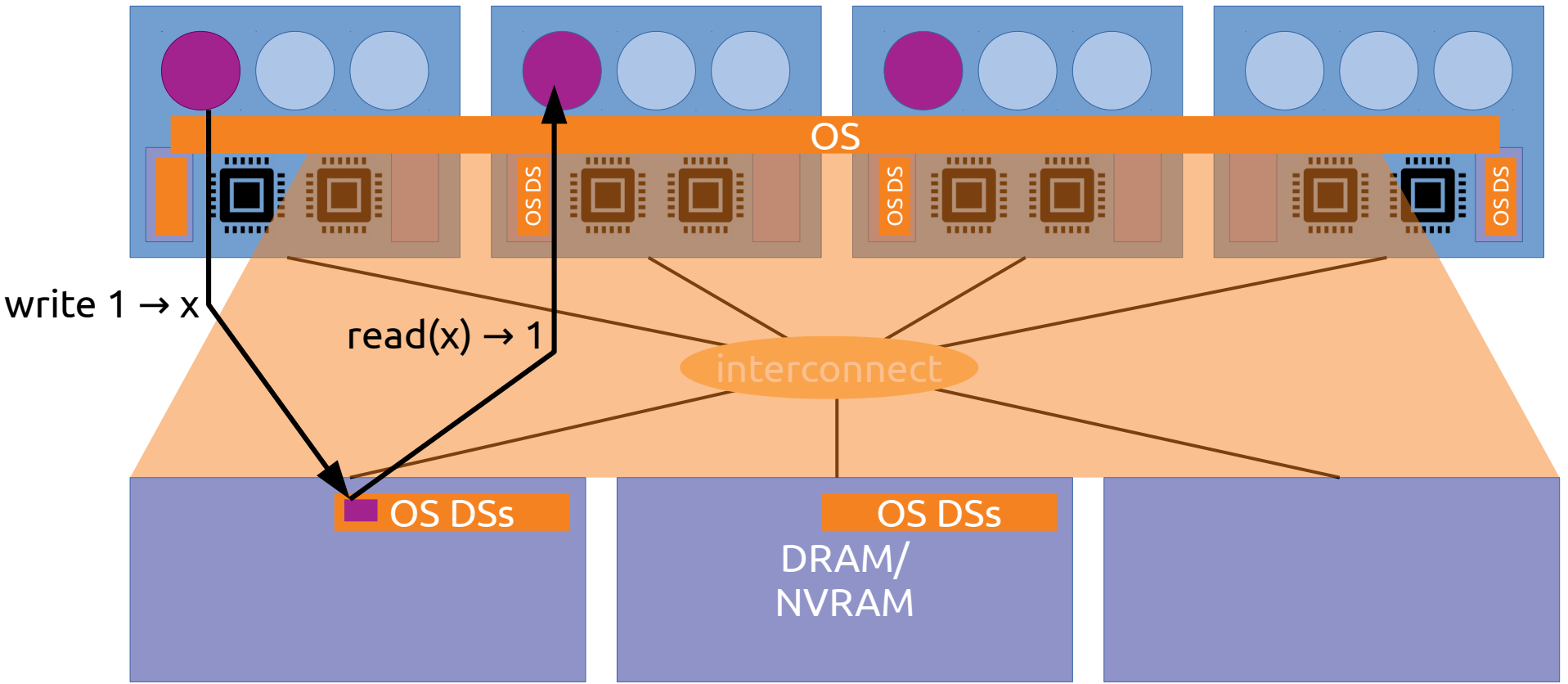
Shared Computations



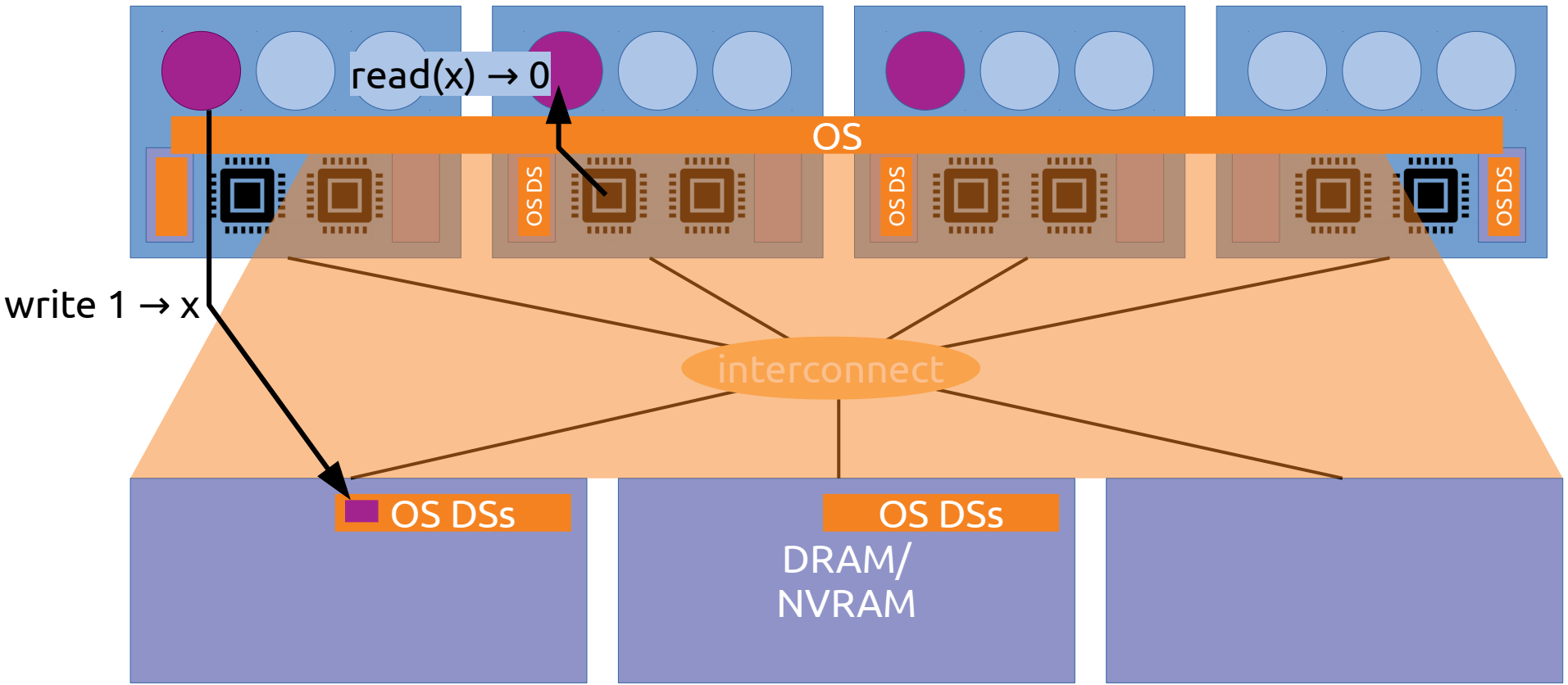
Memory Coherency between Nodes?



Memory Coherency between Nodes?



Memory Coherency between Nodes?



Programming for Non-Coherence

- Each range of memory has one owner
  - partitioned data-structures
  - Access and modification requires communication
- Consistency + replication (e.g. 2PC)
  - Message passing coordination on modification

Ch'i

An OS for *disaggregated* memory

- Non-coherent data-structures
- Global, capability-based access control

Enables local access to shared,
protected components

Sharing Non-coherent Memory

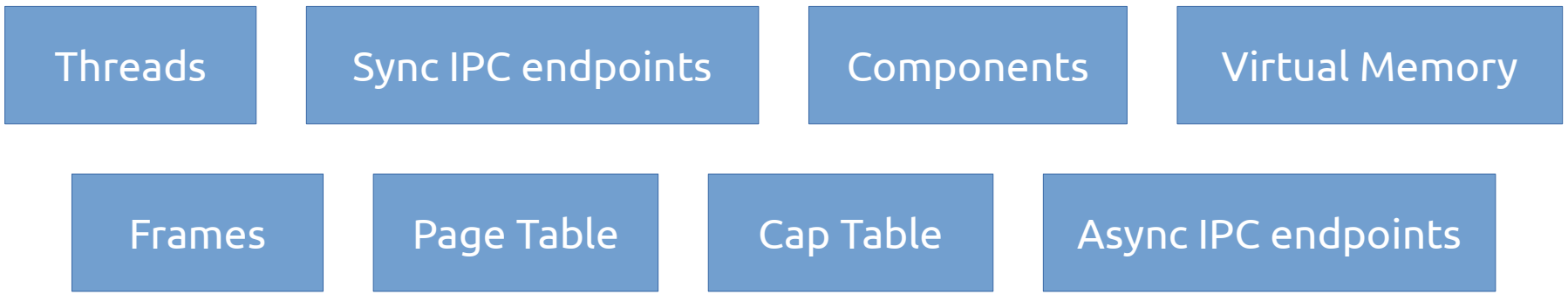
Bounded Incoherence[1]
Load/store coherence ensured within a *bounded time window*. In-memory CAS.

- Update synchronization limited to a single CAS
  - API similar to RCU, prefers lock-free data-structures
  - Modify vs. read-copy-update
- Coherence ensured with *periodic* flushes
  - Cannot *reuse* freed node until period updates storage

[1] Ren et al., *Bounded Incoherence: A Programming Model for Non-Cache-Coherent Shared Memory Architectures*, PMAM, 2020

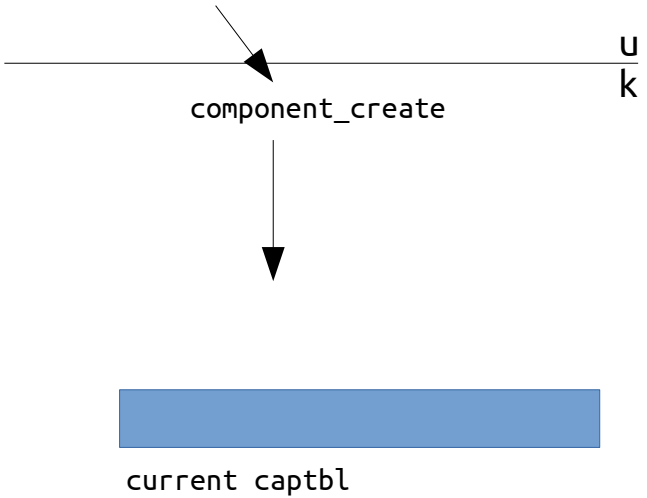
Ch'i: An OS for disaggregated, non-coherent memory

- Composite-based Microkernel
  - Capability-based access control
- Entirely wait-free inter-core coordination



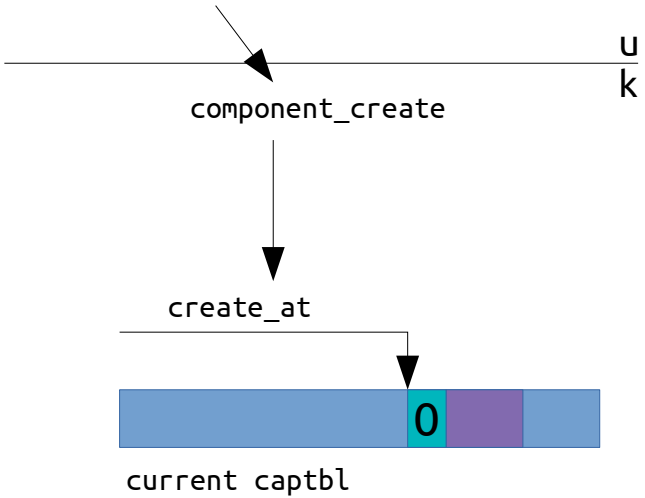
Example Kernel Resource Creation

```
component_create(create_at, pgtbl, captbl):
    /* find a slot in the current component's captbl */
    entry = captbl_lookup(create_at);
    /* reserve the slot for our modifications */
    if (cas(entry→header, 0, RESERVED)) return ECOLLISION;

    entry→data = {pgtbl, captbl};

    /* Activate, and writeback cache-lines */
    entry→header = ACTIVE;
    clwb(entry);
```



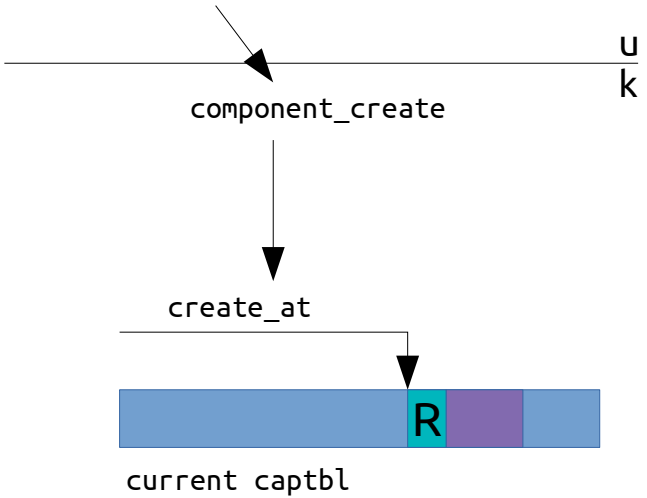
Example Kernel Resource Creation

```
component_create(create_at, pgtbl, captbl):
    /* find a slot in the current component's captbl */
    entry = captbl_lookup(create_at);
    /* reserve the slot for our modifications */
    if (cas(entry→header, 0, RESERVED)) return ECOLLISION;

    entry→data = {pgtbl, captbl};

    /* Activate, and writeback cache-lines */
    entry→header = ACTIVE;
    clwb(entry);
```



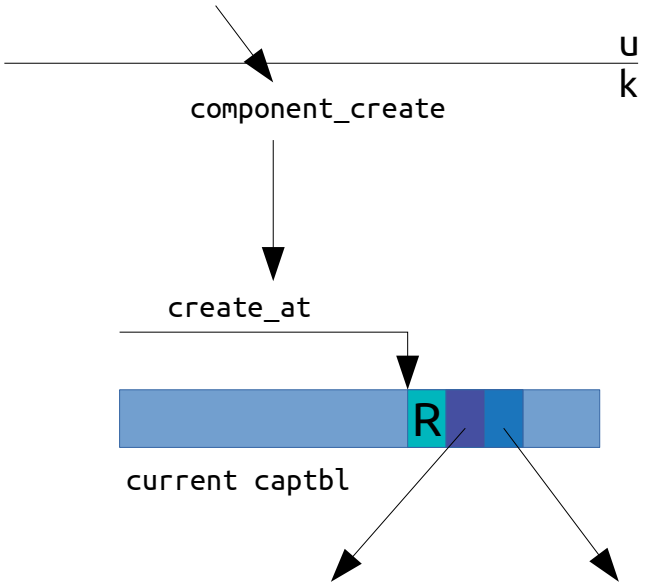
Example Kernel Resource Creation

```
component_create(create_at, pgtbl, captbl):
    /* find a slot in the current component's captbl */
    entry = captbl_lookup(create_at);
    /* reserve the slot for our modifications */
    if (cas(entry→header, 0, RESERVED)) return ECOLLISION;

    entry→data = {pgtbl, captbl};

    /* Activate, and writeback cache-lines */
    entry→header = ACTIVE;
    clwb(entry);
```



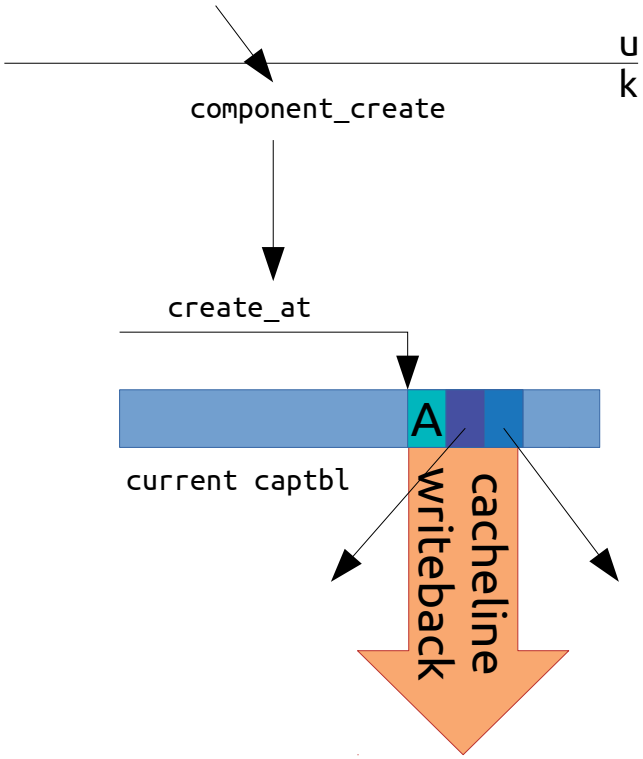
Example Kernel Resource Creation

```
component_create(create_at, pgtbl, captbl):
    /* find a slot in the current component's captbl */
    entry = captbl_lookup(create_at);
    /* reserve the slot for our modifications */
    if (cas(entry→header, 0, RESERVED)) return ECOLLISION;

    entry→data = {pgtbl, captbl};

    /* Activate, and writeback cache-lines */
    entry→header = ACTIVE;
    clwb(entry);
```

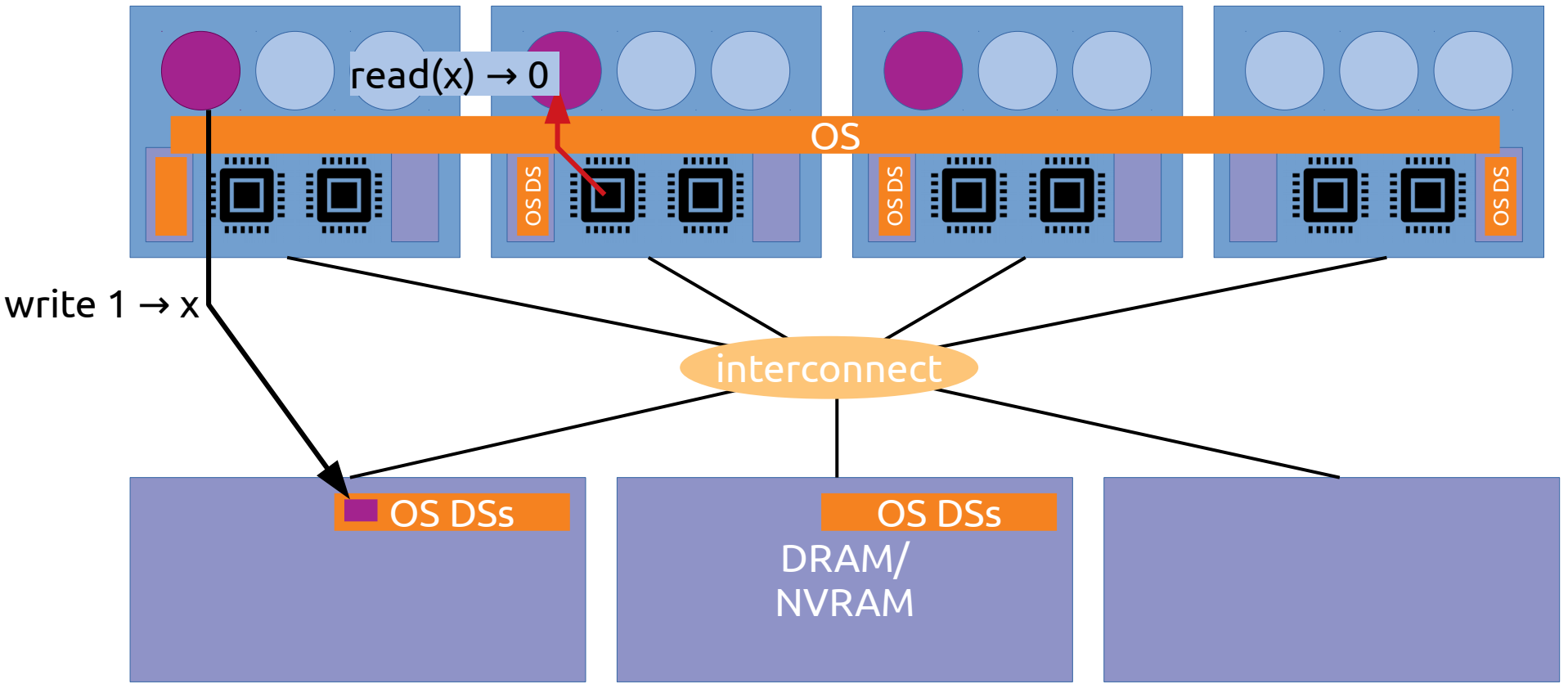


Example Kernel Resource Creation

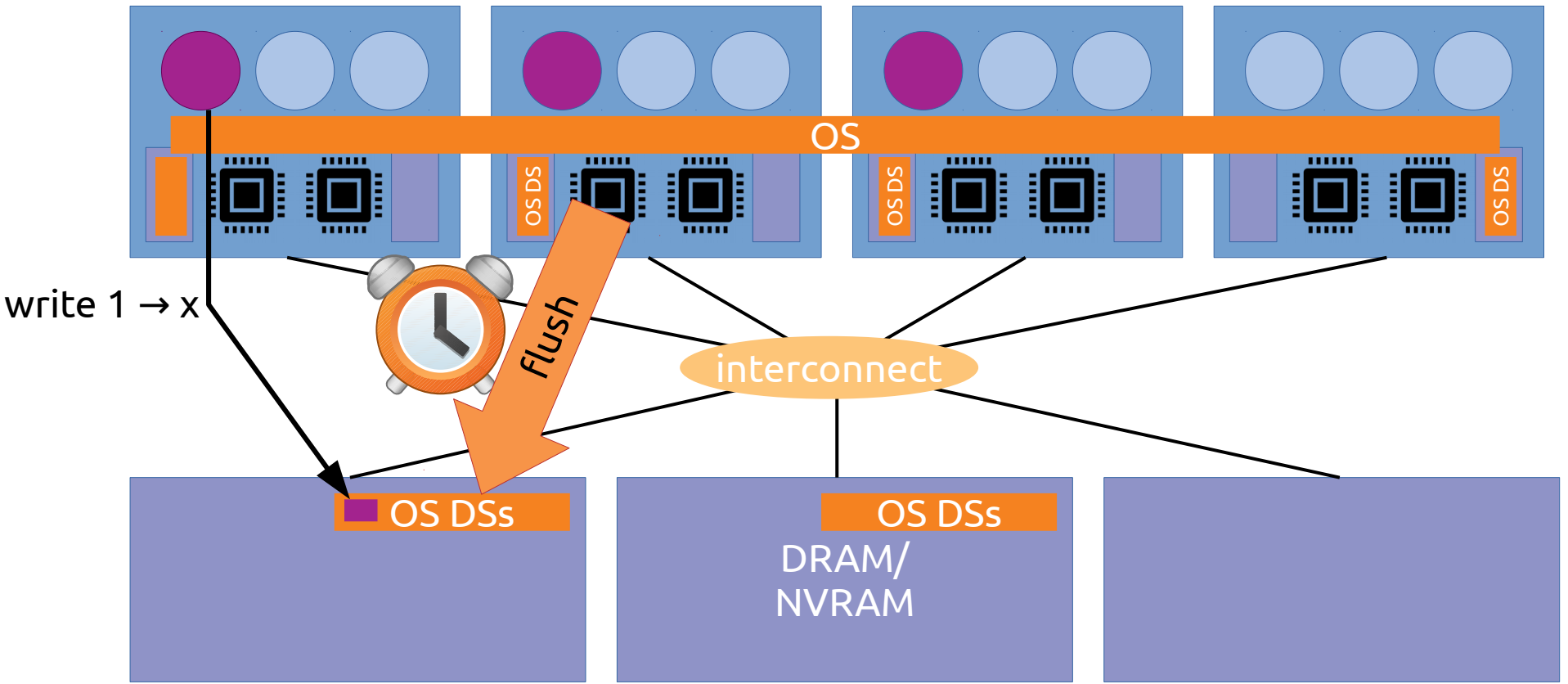
```
component_create(create_at, pgtbl, captbl):
    /* find a slot in the current component's captbl */
    entry = captbl_lookup(create_at);
    /* reserve the slot for our modifications */
    if (cas(entry→header, 0, RESERVED)) return ECOLLISION;

    entry→data = {pgtbl, captbl};

    /* Activate, and writeback cache-lines */
    entry→header = ACTIVE;
    clwb(entry);
```



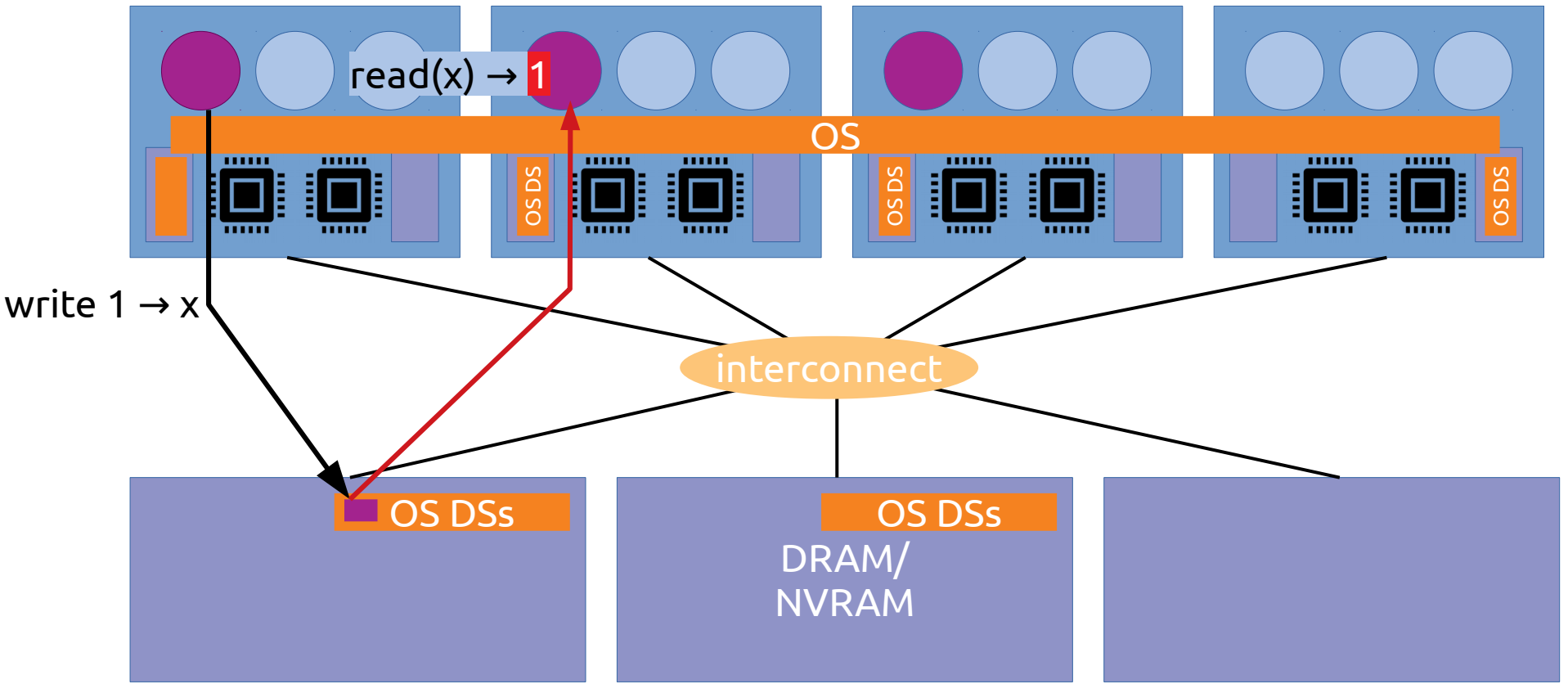
Periodic Cache Coherency



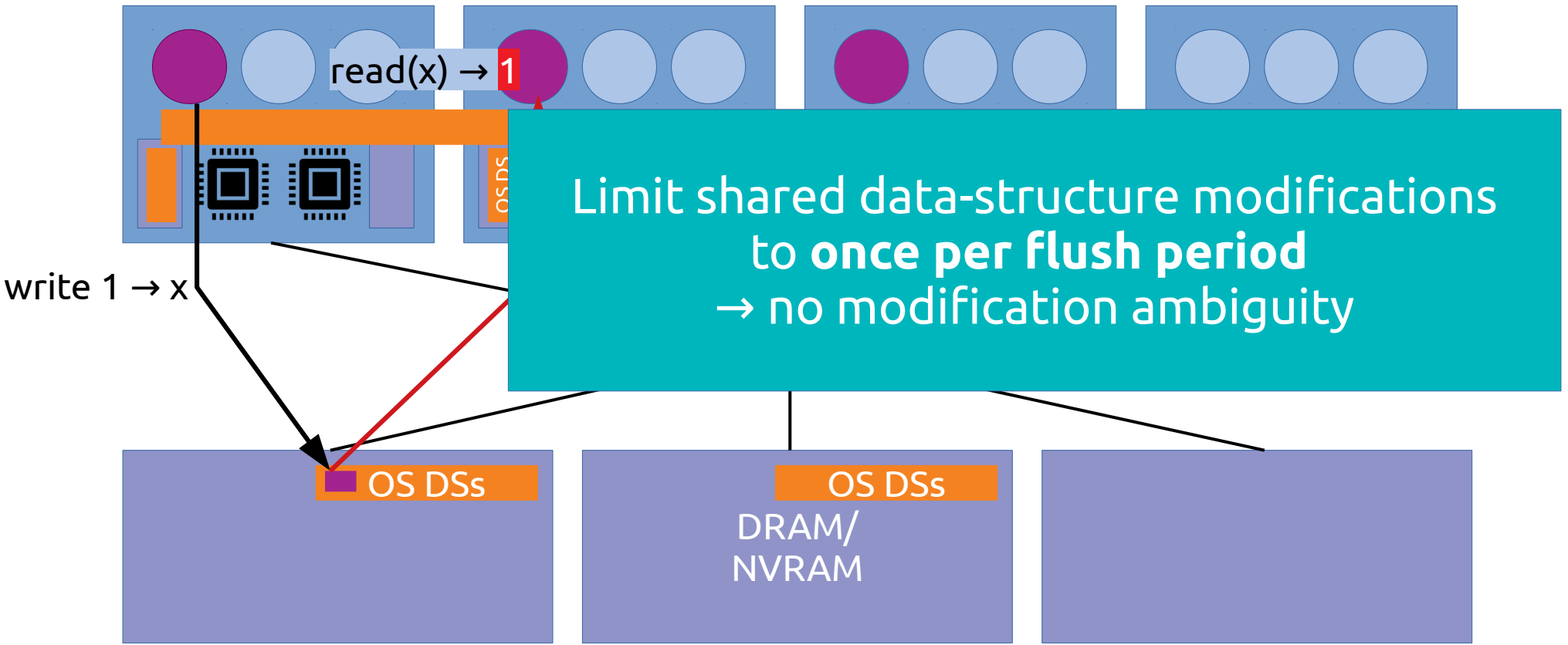
Periodic Cache Coherency



Periodic Cache Coherency



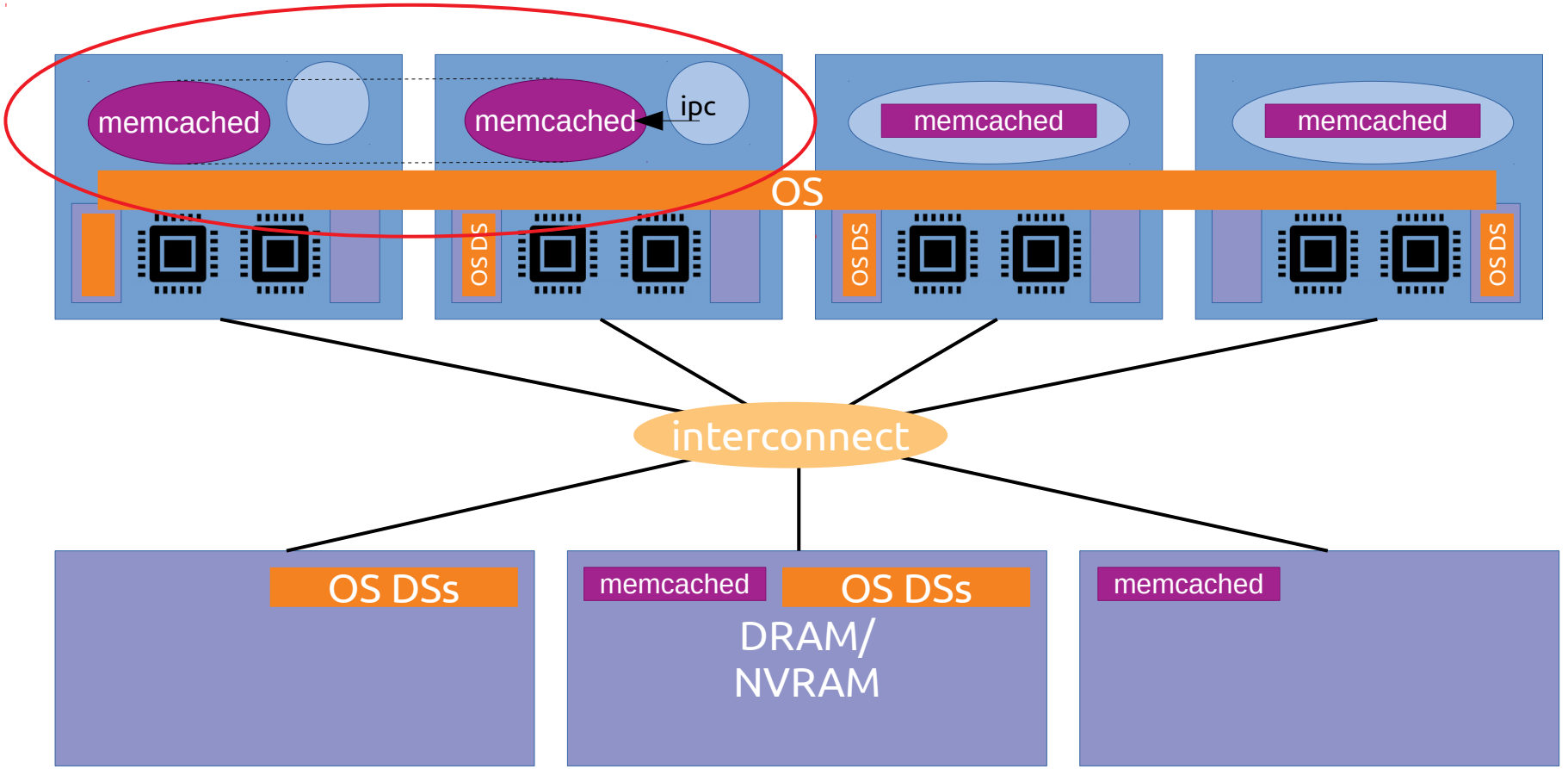
Periodic Cache Coherency



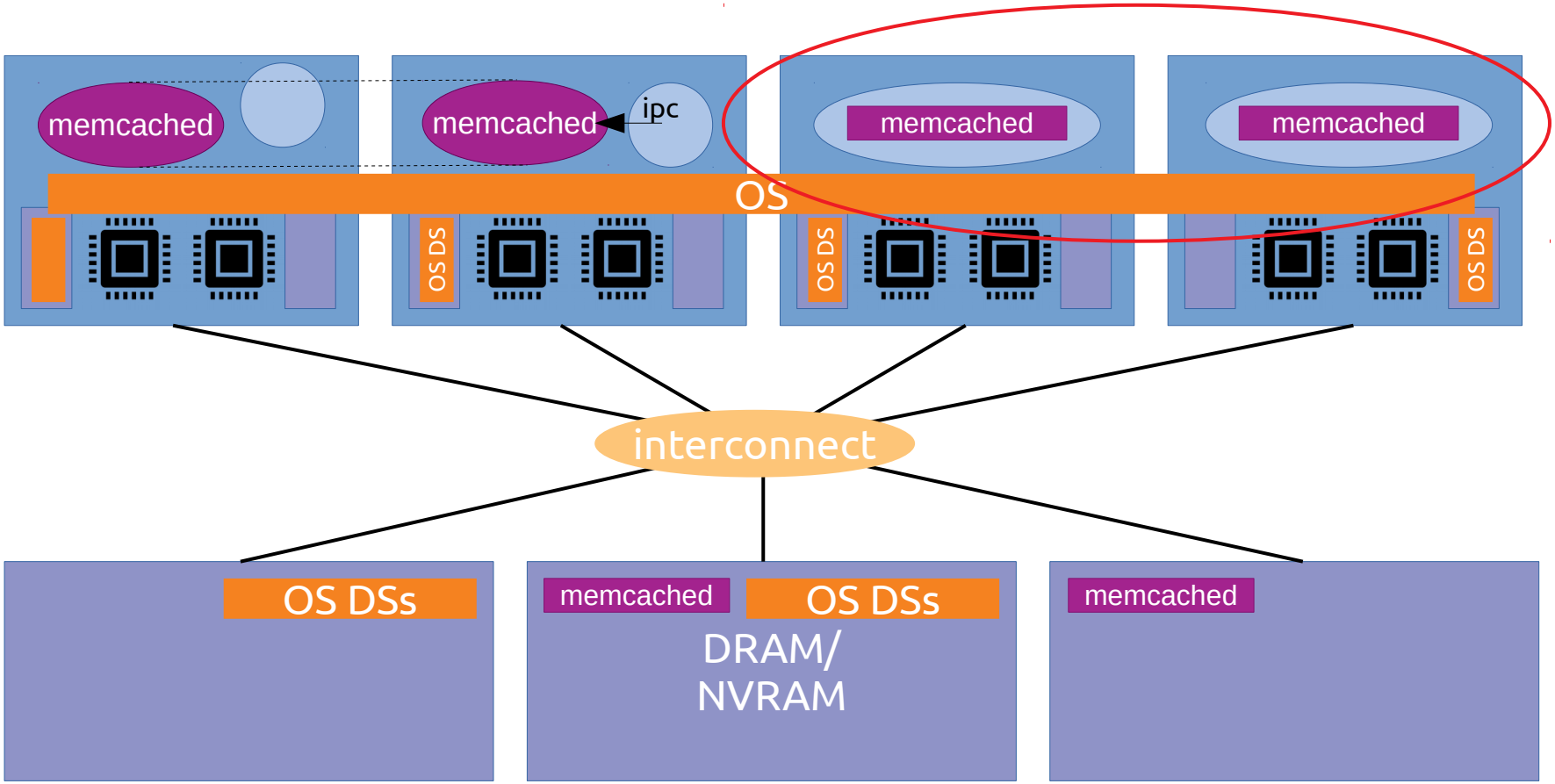
Periodic Cache Quiescence (Flush)

- Each node flushes its cache periodically
  - Whole cache: very expensive
- Flushes only *accessed* cachelines
  - Use *accessed bits* in kernel page-table
  - Only flush those pages
- Other considerations for pgtbls + TLBs
- Not addressed: user-level coherence

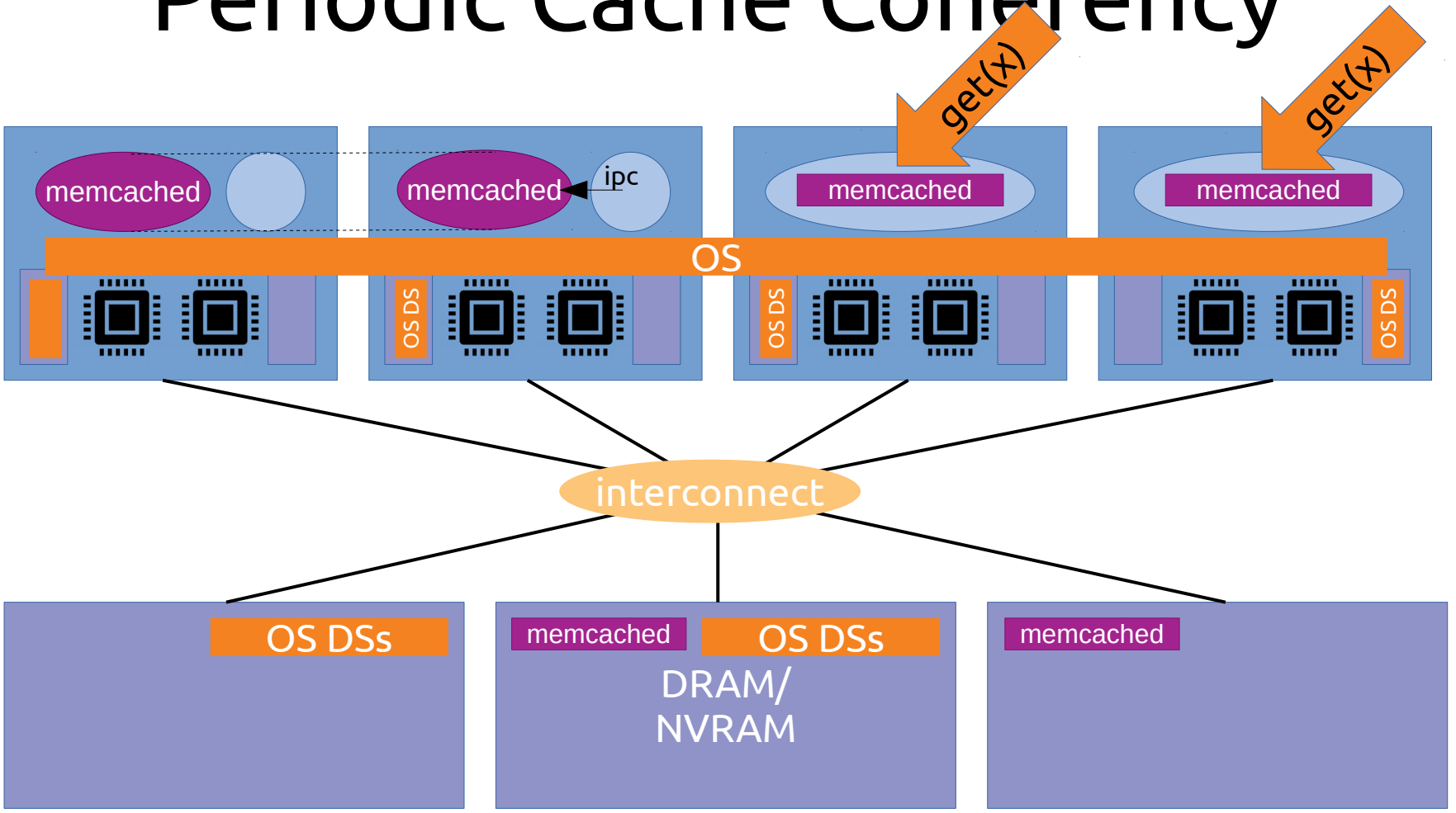
Periodic Cache Coherency



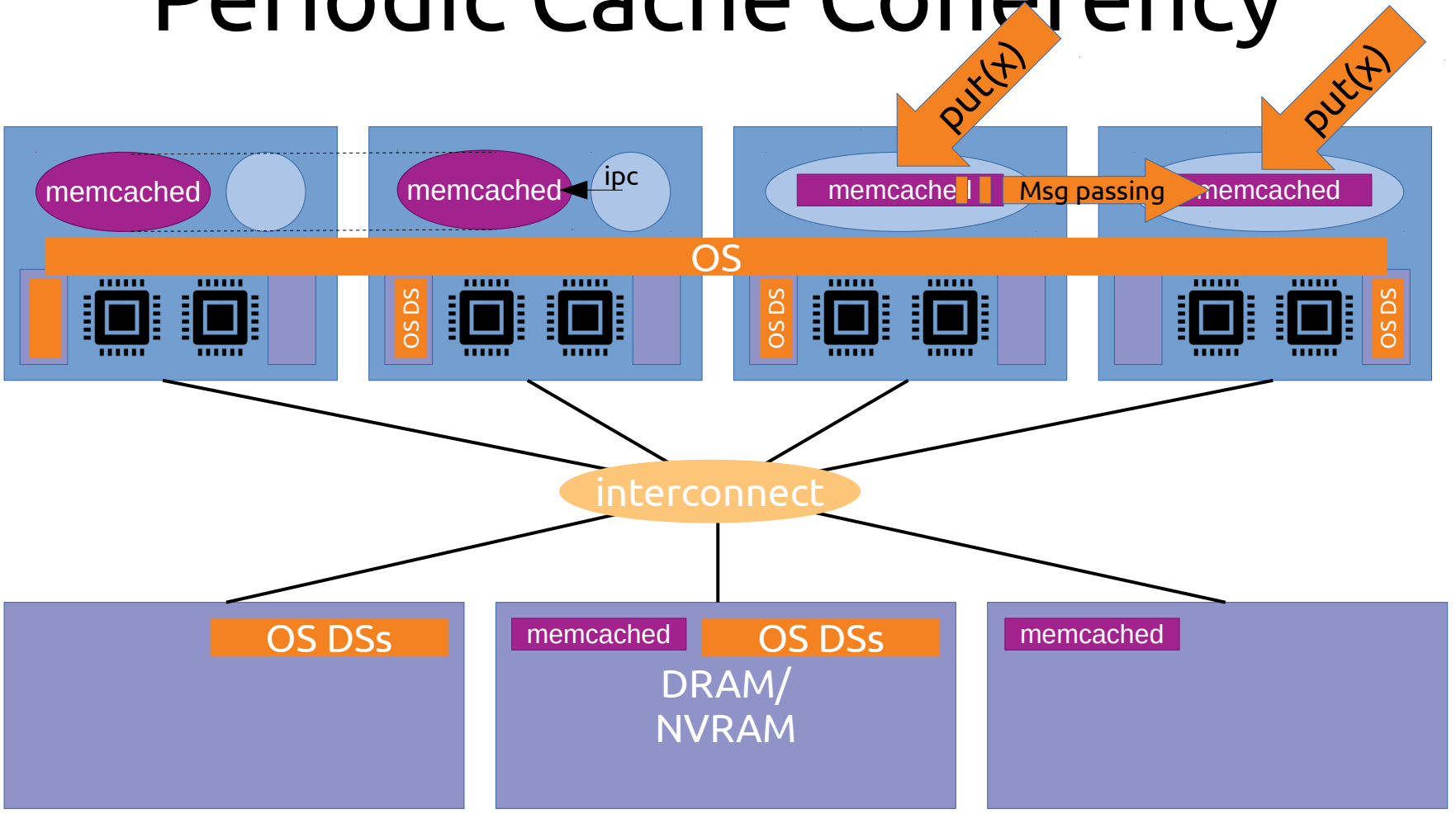
Periodic Cache Coherency



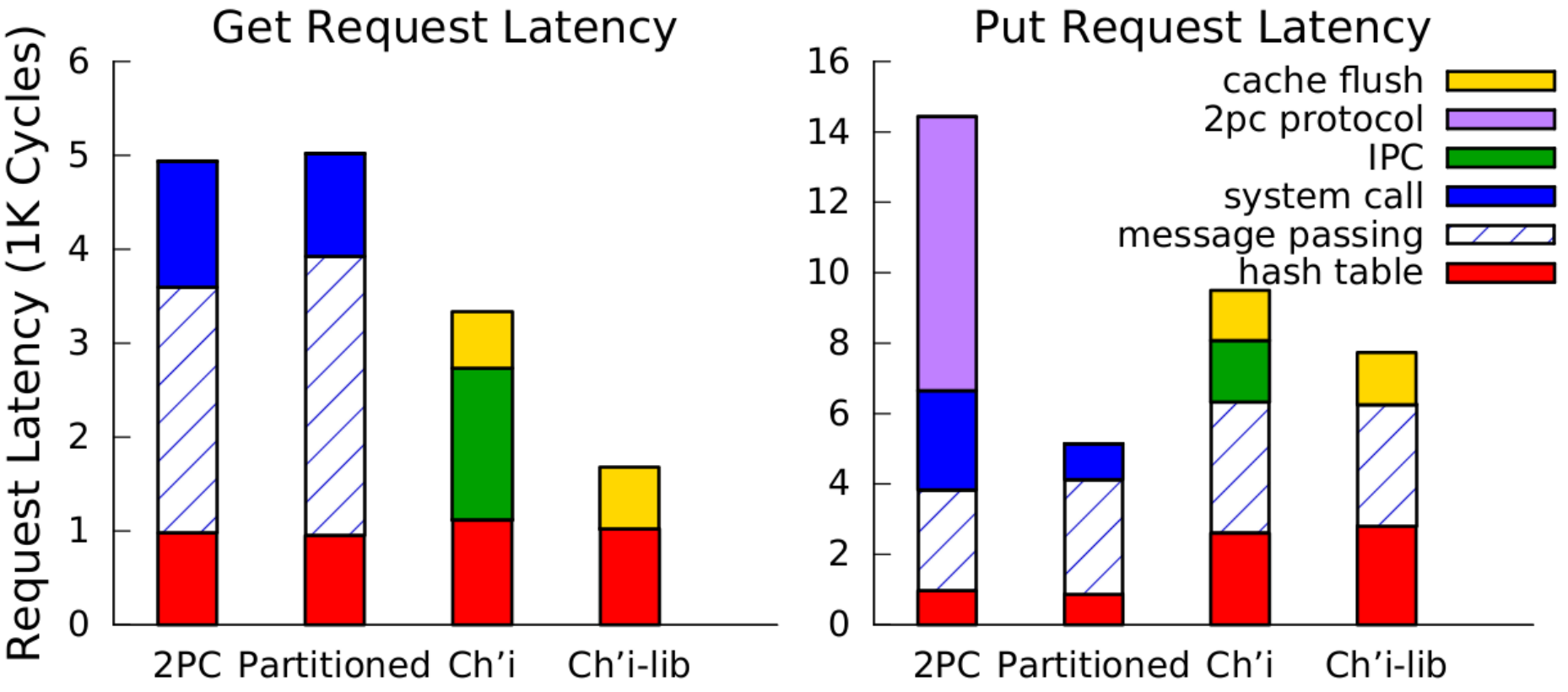
Periodic Cache Coherency



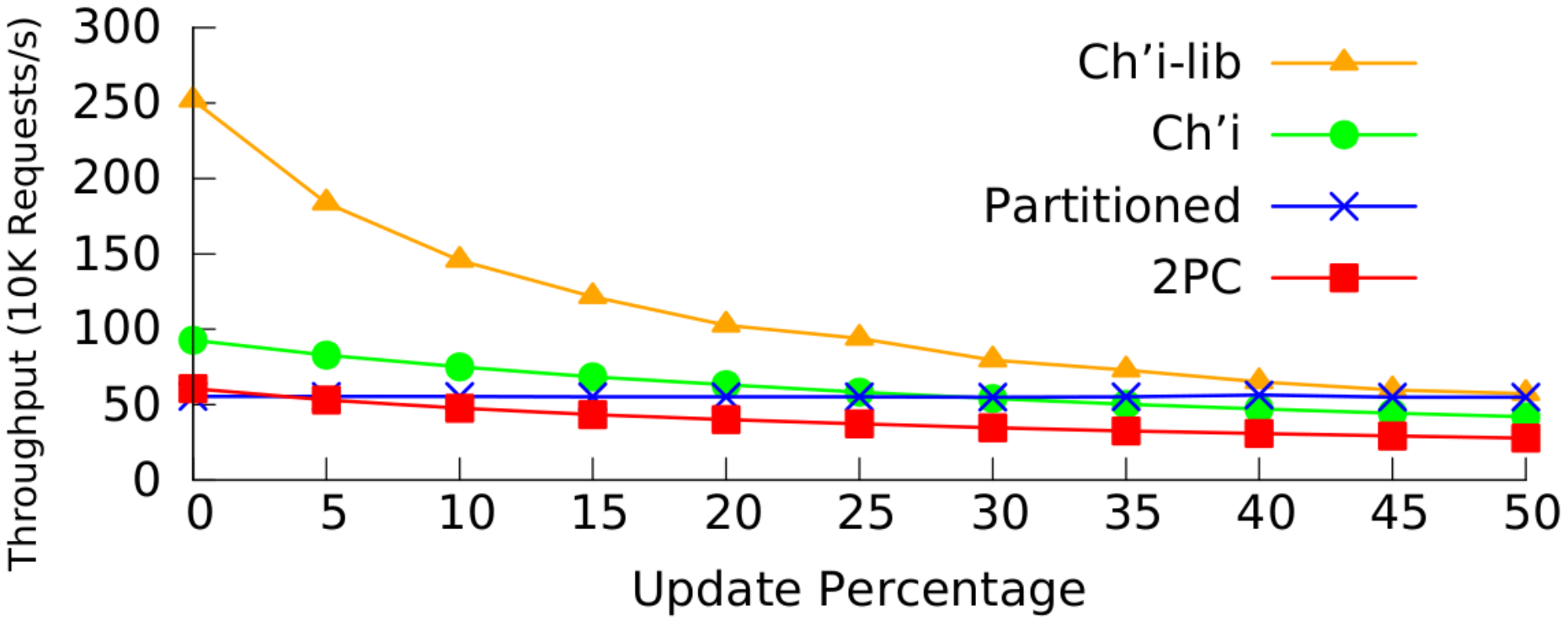
Periodic Cache Coherency



Memcache overhead breakdown



Memcached throughput



Ch'i Summary

- OS for **non-coherent**, shared memory
  - Full-system access control across incoherent mem
- Leverages simple, wait-free kernel
- Uses Bounded Incoherence for coherency

composite.seas.gwu.edu