Progress Report: Page Cache Consistency Model

Zhengyi Chen

December 4, 2023

The System

- Remote node(s) abstracted as shared memory device "`/dev/rshm`"
- Heterogeneous Memory Management (HMM) ensures unified address space between local and device memory.
- Migration of pages between CPU and "device" is transparent to userspace – no need for copying/mapping.
- In reality, "`/dev/rshm`" a handler for RDMA access between nodes.
  - This involves remote read/write and moving page content between nodes.
  - Local node serves as *home node & address space host* at share time.
  - Remote nodes attached on `/dev/rshm` as accelerator.

The Problem: Consistency Protocol

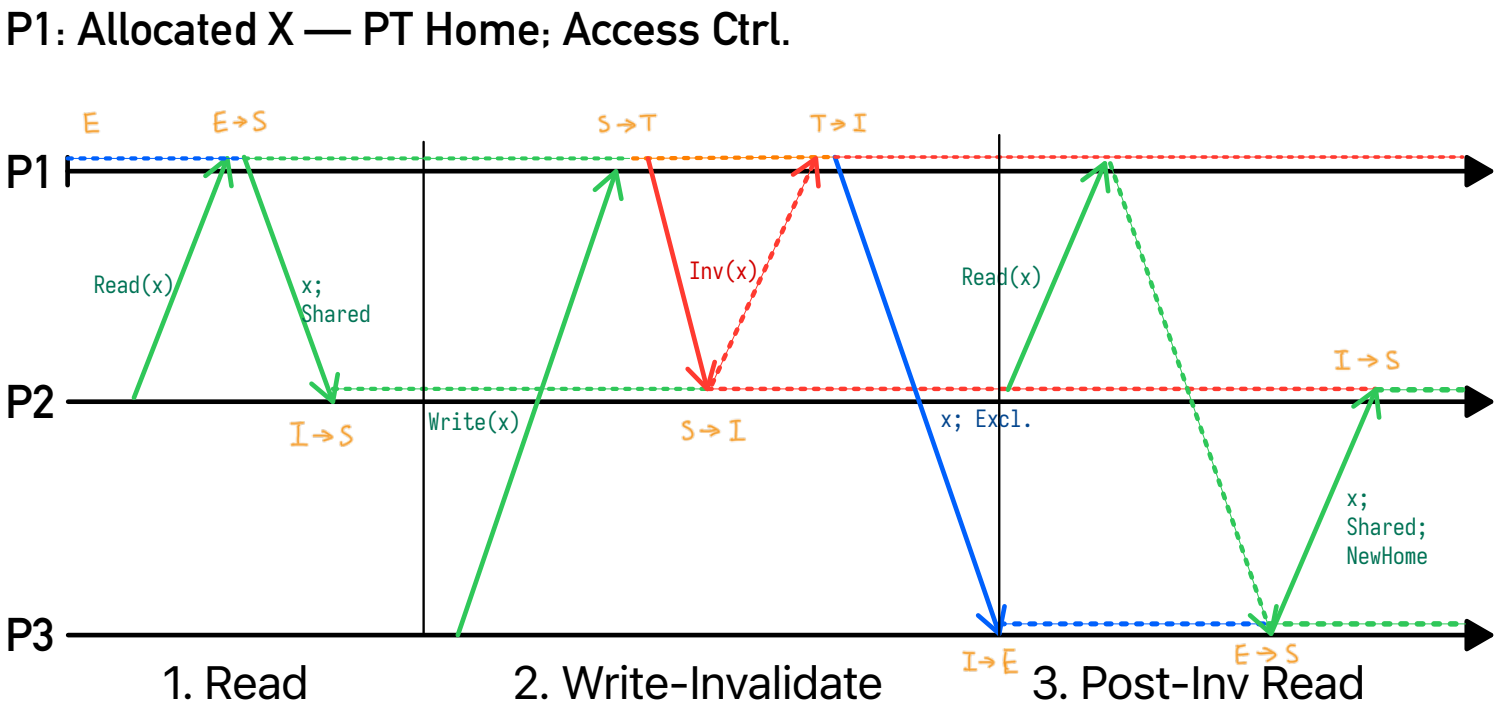
- Single-Writer, Multiple-Reader Protocol
- Need to be performant. . . with some ergonomics
- Two Hypothetical Protocols:
  - "RwLock" Consistency Protocol
  - Acq-Rel Consistency Protocol
- Former ensures *strong* single-writer consistency
  - – Also easier to program with!
- Latter allows concurrent in-memory *non-committal* computation

"RwLock" Consistency Protocol

Similar to a read-write lock where:

- Multiple readers can exist for a clean page – the page is **shared**.
- Only one write is allowed for a clean page – the page becomes **exclusive**.
- For one writer node to be allowed sole write access to some page, all other readers need to have their page cache invalidated.
- While the sole writer node has not yet committed, no other reader or writer nodes are allowed to be served this page.
- When the sole writer commits, it becomes the new home node which serves the updated page content.

"RwLock" Consistency Protocol



Note: The blue arrow should be acknowledged by P3 – forgot to put the ack. arrow in.

Acq-Rel Consistency Protocol

In RwLock's case, read requests result in installation of read-only pages at remote nodes.
Alternatively, this protocol allows read/write pages to be installed at remote nodes at read time. Such writes are *non-committal* and cannot be synced with the entire system.
To summarize:

- "Readers" can write to its locally installed page without any means to synchronize the change.
- "Writers" need to acquire global write access from the *PT node*, which invalidates all shared pages.
- i.e., Instead of write-invalidate, perform acquire-invalidate.

Consistency Protocol: Knobs and Mods

We can modify these two protocols further as follows:

- Multi-home Protocol: instead of having one home at a time, have multiple homes (e.g., when writer commits) to prevent network bottleneck.
- Auto-share: Mark pages shared via `/dev/rshm` as automatically shared to some remote nodes such that 1-way communications suffice to re-validate invalidated pages.
  - Potential for communication reduction – debatable.

What about Consistency **Model**?

- The weaker a consistency model is, the more difficult it is to program with.
  - Weak ordering architectures (e.g., ARMv8) more or less depends on compiler/interpreter to emit barriers as see fit Haynes, *Sequential consistency in armv8*.
  - Bad for usability/portability – programs may need to be compiled using a modified toolchain, else need to add these synchronization instructions/function calls everywhere.
- [1] uses Partial Store Order.
  - Preserves RAR, WAR – "synchronous read. . . asynchronous write"
  - Easier to use than relaxed ordering.
- [2] uses strong consistency, but warns about its scalability.

---

[1] Cai et al., "Efficient distributed memory management with RDMA and caching".

[2] Wang et al., "Concordia: Distributed shared memory with {In-Network} cache coherence".

Consistency Model: Cont.

- Similar to Concordia[3], the proposed protocols also assume strong consistency.
- Further work needed to see how to adapt these protocols for weaker consistency models.

[3] Wang et al., "Concordia: Distributed shared memory with {In-Network} cache coherence".