Implementing shared memory on distributed systems

Shreyan Goswami

1

Introduction

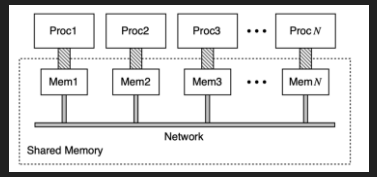
2

The promise of cloud

- Combine relatively cheap hardware to match performance of large computing clusters
- Organizations already have installed workstations, so reuse them
- Various systems such as distributed shared memory and message passing have been proposed

3

Distributed shared memory(DSM)



Source: https://www.cs.rochester.edu/u/sandhya/papers/computer96.pdf

- TreadMarks is a DSM, allowing processes/processors on different machines to share memory

4

Problems

5

Existing implementations

- Many run on in-house research platforms
- Requires kernel modifications
- Performance issues because they mimicked consistency protocols used by hardware shared memory multiprocessors.
- False sharing
- Example: IVY

6

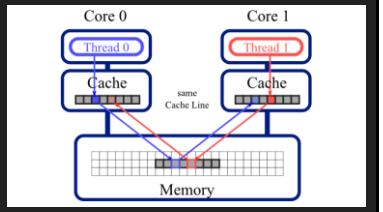
Messaging is expensive

- Communication across a network is expensive
- Involves interrupts, context switches and execution of several layers of network software
- For a performant DSM, consider minimizing the number of messages and volume of data sent across the network

7

False sharing

- Invalidates entire cache line in a multiprocessor system
- In DSM, entire page get invalidated
- False sharing is more undesirable in DSM systems.
- Why?



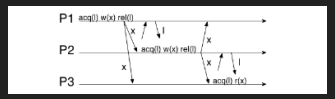
Source: https://haryachyy.wordpress.com/2018/06/19/learning-dpdk-avoid-false-sharing/

8

Solutions

9

Reducing the number of messages

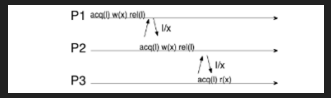


Source - class slides

- Traditional DSM would invalidate pages
- Ivy sends a message for each update and invalidate
- Further optimizations possible
- This is called eager release consistency

10

Reducing the number of messages



Source - class slides

- Observation: Only one processor can execute in the critical section
- Send the communication when the next processor requests for a lock.
- Changes to the shared data can be communicated during such a synchronization event
- This is called lazy release consistency

11

Reduce false sharing

- Traditional coherence protocols require exclusive access to data before write can happen
- Want to avoid invalidating every other processor's data
- Enable multiple writers to change a single page without invalidations.
- Communicate changes to a page during a synchronization event.
- Enabled through write notices and diff generation

12

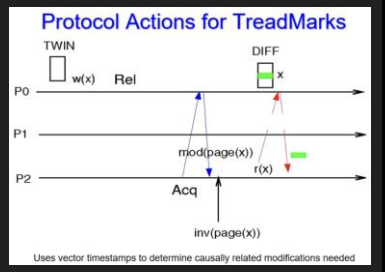
Write notices

- During synchronization wait for the information that a page has changed to be transmitted
- Issue the information that a change has happened during a synchronization but not the actual change - write notice
- Processor on receiving a write notice can behave in different ways.
- TreadMarks uses invalidation protocol
- The processor invalidates the page for which it has received a write notice
- When requesting for the page again, a page fault is triggered and the processor requests the diff from the processor who issued the write notice earlier.

13

Diff generation

- Before a processor writes to a shared page, it always creates a twin of the page
- Create a run length encoding of the changes by comparing with the twin
- No need to send the entire page for communicating change reducing the size of message
- Twin is discarded after diff creation



Source - class slides

14

Diff generation

- Diffs are not generated at the point of synchronization
- Processor can continue making changes until a diff is requested
- Leads to a lazy diff generation and improved performance

15

Data structures

16

PageArray

- Main data structure
- Contains the state of the shared page: no access, read-only or read/write
- An approximate copyset specifying which processors currently cache the page
- Most important: an array indexed by the processor, each index pointing to head/tail of a doubly linked list of write notices sorted in descending order of interval
- If the diff is generated for the write notice, the node points to the diff in the diff pool

17

Other supporting data structures

- ProcArray: an array which has one entry for each processor to the head and the tail of a doubly linked list of interval records.
- Interval records
- Set of write notices
- Diff pool

18

Kernel and compiler level changes

- No changes at the OS level
- No changes required at the compiler level

19

Implementation

20

Locks

- Locks have statically assigned managers assigned in a round robin manner
- Acquirer sends their current vector timestamp to the manager
- Manager forwards the request to the previous lock holder
- When lock is released, the releaser informs the acquirer of all intervals between the vector timestamp in the acquirer's lock request message and the releaser's current vector timestamp.

21

Barriers (client perspective)

- Barriers have a centralized manager
- At barrier arrival, each client informs the manager of the manager's last vector timestamp that the client is aware of.
- The client informs the manager of all it's intervals between the last vector timestamp of the manager that the client is aware of and the client's current vector timestamp

22

Barriers(manager perspective)

- When manager arrives at the barrier it adds these intervals into its data structures
- After all messages are received, the manager informs all clients of all the intervals between their vector timestamp and the manager's current vector timestamp.

23

Lazy release consistency

- Release consistency requires a processor p, can continue past its acquire if all updates that happened at intervals less than p's vector timestamp is visible to p
- At an acquire, p will send its timestamp to the previous acquirer q
- Processor q compares it's timestamp with that of p's
- Issues write notices to all intervals in q's current vector timestamp but not in p's vector timestamp
- Processor p invalidates all pages for which write notices have been issued

26

Diff/Interval creation

- Lazy RC allows diffs to be created when requested
- Normally, vector timestamp is updated every time an event occurs in the system
- Delay this to the point when there is a need to communicate with another processor
- At this point generate write notices for all pages that was modified since the last sync operation
- Allows the current processor to keep making changes without invalidations

27

Final questions, thoughts and comments

28