Distributed Shared Memory

Jonathan Walpole

Department of Computer Science & Engineering
OGI/OHSU

The Basic Idea

- multiple processes share a single virtual memory space
- processes do loads/stores from/to memory locations
- pages may be resident (local) or non-resident (& remote)
- accesses to non-resident pages generate page faults
- page faults are handled by the OS and serviced by the DSM middleware
  - perhaps by retrieving the page from another machine
- protection faults can also be used by the DSM system to intercept “interesting” references to the shared memory
  - perhaps by invalidating pages on another machine

Characteristics

- Inter-process communication is via modification and subsequent reading of shared memory locations
  - semantics defined by memory consistency model and use of synchronisation primitives
- Local and remote communication looks the same
  - remote communication is hidden behind MMU faults that are handled transparently to the application program
  - some memory accesses take (much) longer than others
  - analogy with cache misses on SMPs
- Its like programming a shared memory multiprocessor
  - UMA vs NUMA vs NORMA architectures

Key Issues and Challenges

- Performance
- Memory consistency model
- Implementation of synchronisation primitives

Performance

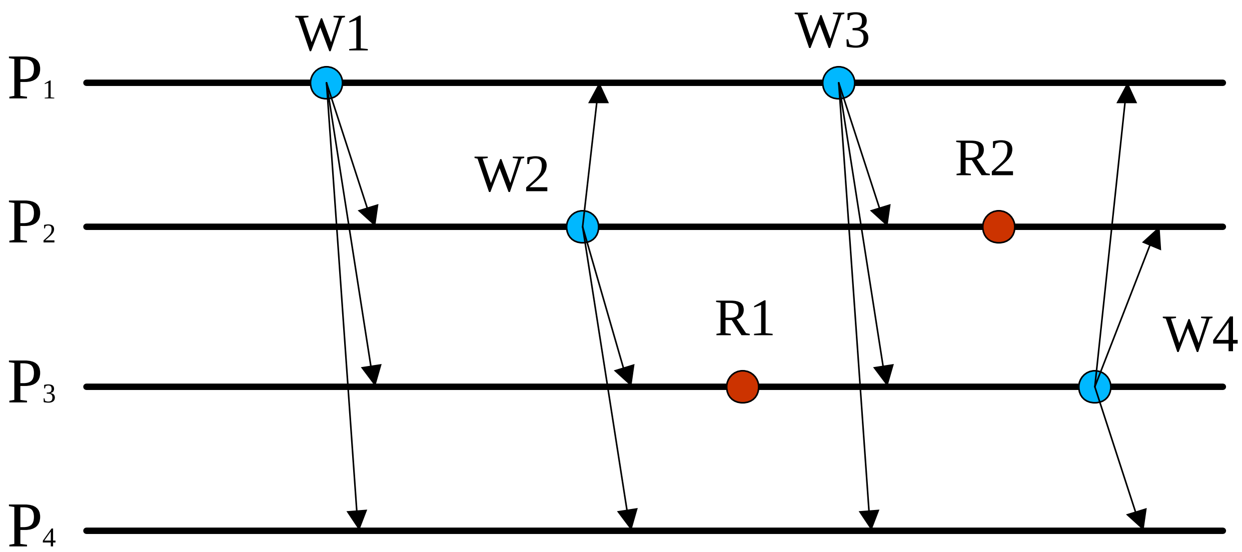
- Minimum unit of communication is a page
- Concurrent access to the same page by remote processes causes thrashing
  - repeated page faults and transfer of the page
  - may not be accessing the same location on the page
  - thrashing can be reduced by temporarily pinning the page
  - but this increases access latency for the other process
- Small pages reduce thrashing due to false sharing
  - but they increase management overhead and network overhead

Performance

- Performance is affected by the number of message exchanges required to service a page fault
  - how to locate a page?
  - how to invalidate copies of the page?
  - how to propagate updates to copies of pages?

- The number of protection faults and message exchanges is strongly influenced by
  - the memory consistency model
  - caching strategies

Strong Memory Consistency



Total order enforces sequential consistency (or linearizability if real-time order is also preserved):

- intuitively simple for programmers, but very costly to implement
- not even implemented in non-distributed machines!

Cost of Implementing Strong Consistency

- Centralized DSM systems
  - propagate all memory references to a well-known central server
  - central server serializes all requests to enforce strong consistency
  - all accesses are remote so performance sucks!
  - the central server is a bottleneck

Cost of Implementing Strong Consistency

- Migrating DSM systems
  - allow pages to migrate among processors
  - pages are faulted in on first access
  - with the right locality of reference most accesses will be local and few page migrations will be required
  - a single page is at a single location at any time so strong consistency is ensured
  - no parallel read access for any given page
  - many read faults so performance sucks!
- And anyway, how do you know where to look for the page you want?

Cost of Implementing Strong Consistency

- DSM systems with read-only page replication and central server for writes
  - first read access faults and causes creation of local copy of page
  - subsequent read accesses are serviced from the local copy
  - write accesses fault and are coordinated by the central server
  - either write accesses are propagated via atomic broadcast to all read-only copies, in which case write accesses are very expensive so performance sucks!
  - or, write accesses cause invalidation of all read-only copies via broadcast, in which case many more read faults occur and performance sucks!
  - Doesn't take advantage of locality for writes!

Cost of Implementing Strong Consistency

- DSM systems with read-write replication and migration
  - read or write accesses create local “cached” copy of a page
  - cached pages can be read or written locally
  - strong consistency requires reads and writes to be serialized
  - maintain a single writable copy at the page “owner” and read-only pages elswhere
  - migrate ownership and propagate updates or invalidations as before
- So how do you find the owner of the page you want, or the replicas to update or invalidate?
  - centralized vs decentralized location service?
  - use of broadcast protocols

What Do We Really Want?

- Parallel reads
- Parallel writes to different locations in a page
  - unless the programmer says they are related
- Parallel reads and writes to different locations in a page
  - unless the programmer says they are related
- Synchronization primitives to allow the programmer to specify when requests are related
- Local writes
  - writable copy located locally when writes occur
- Prefetching / eager update via multicast on write completion

Release Consistency

- Directives define boundaries of access to shared data
  - acquire - obtain updated copy
  - release - finished updating shared memory locations
- Weak consistency semantics based on timing of the propagation of updates
  - release consistency - updates propagated on release
  - lazy release consistency - updates propagated on subsequent acquire
- Release consistency makes the shared memory sequentially consistent for programmers that use synchronization primitives correctly
  - an optimized implementation of a strong consistency DSM?

Synchronization Primitives

- Acquire and release are examples of synchronization primitives for DSMs based on release consistency
  - they are explicit calls to the DSM system allowing it to manipulate remote pages appropriately before completing the calls
- What about locking primitives in strong consistency DSMs?
  - Test and set lock?
  - When exactly is a write fault triggered during TSL?
  - How is TSL implemented on an SMP?
  - Synchronization primitives need to implement cache invalidation and use memory barriers on some architectures (depending on the memory consistency model)
  - They need to be known to the DSM system