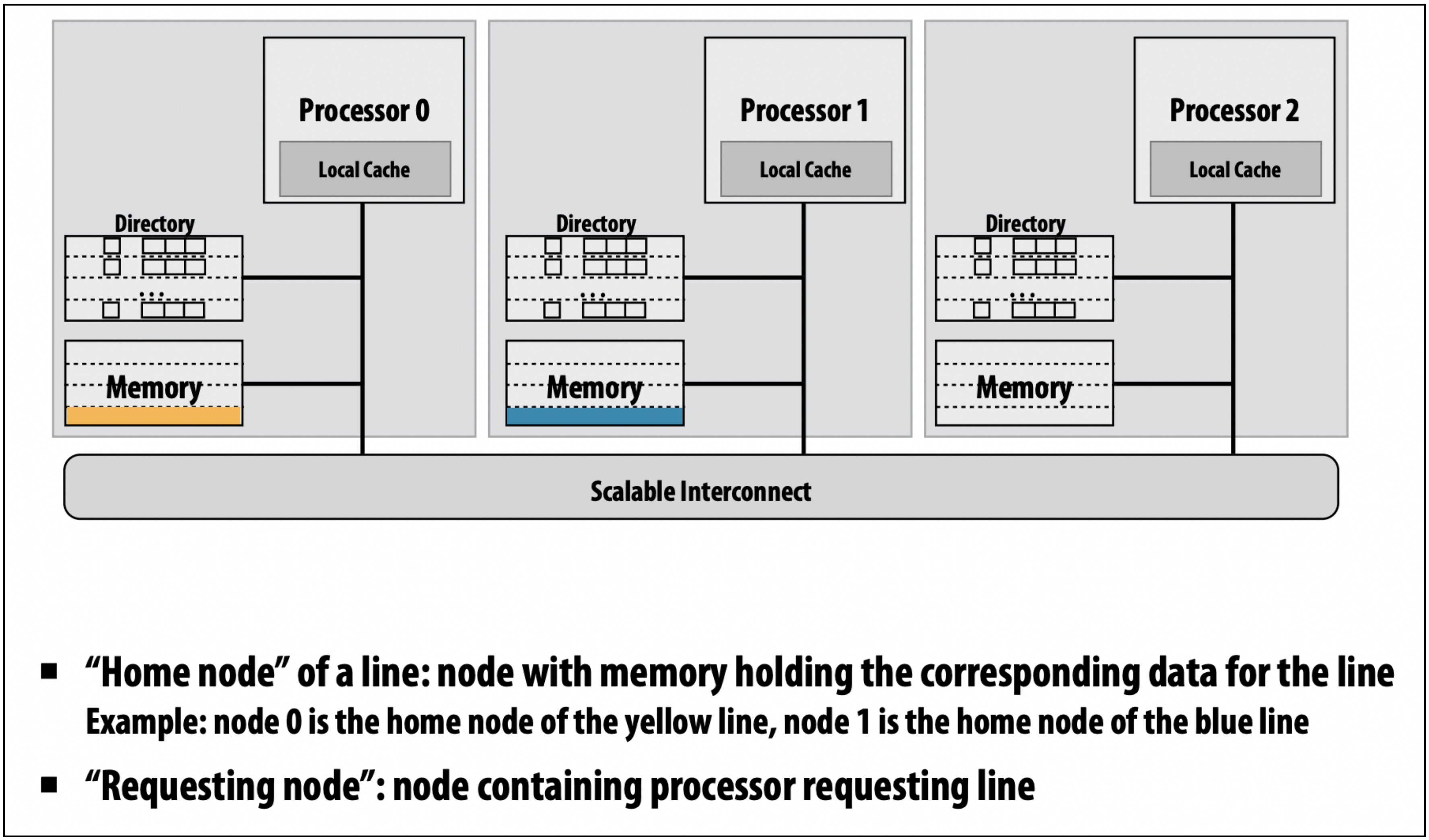
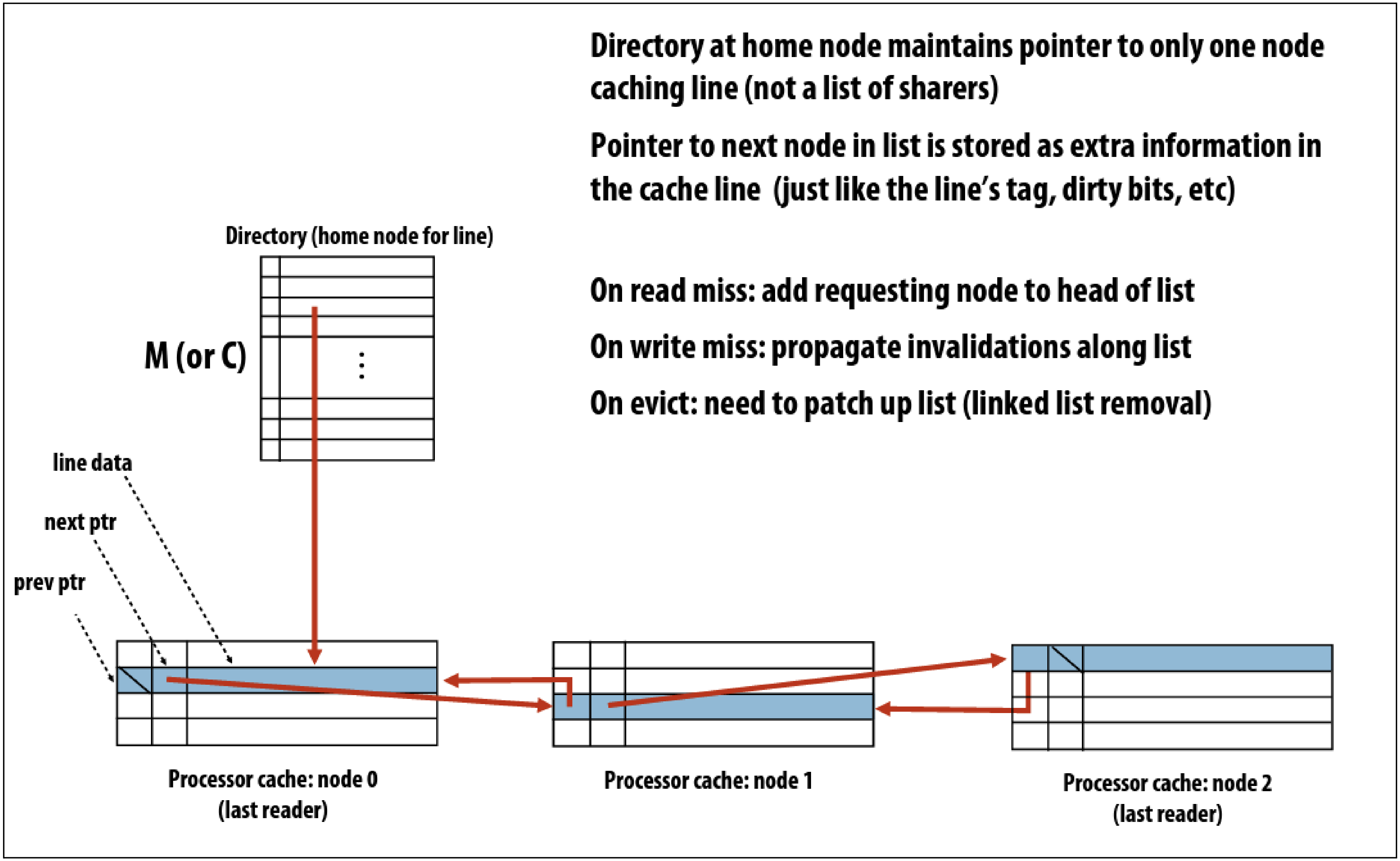
Implementing Cache Coherence Protocols in Multicore Systems

15-418/15-618: Final Project Report - TEAM: Bharathi Sridhar (bsridha2), Tanvi Daga (tdaga)

Analyzing advanced cache coherence protocols in multicore systems, focusing on implementing directory-based approaches like limited pointer and sparse directory schemes. Test and benchmark performance against various cache access patterns and analyzing key metrics such as cache hits, misses, and interconnect traffic.



Implemented protocols include a distributed directory-based approach for NUMA systems, a limited pointer scheme for reduced memory overhead, and a sparse directory method with efficient cache line management. Emphasis on balancing system performance and memory requirements, ensuring coherent cache operations across multicore processors.



Summary of Directory Changes Observed Across different Cache Access Scenarios

Case	Distributed Directory Scheme	Limited Pointer Scheme	Sparse Directory Scheme
Read to clean line from home-node processor to itself	Direct access with no state change (local update is made).	Direct access, limited pointers unaffected (local update is made).	Sparse directory structure unchanged (local update is made).
Read to clean line from non home-node processor	Read request sent, state change to SHARED.	Read request sent, limited pointers updated to track requesting processor.	Read request sent, add processor to sparse directory entry linked list.
Write to clean/dirty line from home-node processor to itself	Direct write, state change to MODIFIED.	Direct write, limited pointers list updated.	Direct write, directory entry linked list modified.
Write to clean/dirty line from non home-node processor	Invalidate other caches, state change to MODIFIED.	Invalidate caches in limited pointer list, state change.	Invalidate along sparse directory list, state change.
Read to dirty line from home-node processor to itself	Direct access, state remains EXCLUSIVE_MODIFIED.	Direct access, limited pointers unchanged.	Direct access, sparse directory structure maintained.
Read to dirty line from non home-node processor	Read request sent, state change to SHARED, invalidates original.	Read request and invalidation limited to pointer list.	Read request sent, original node invalidated, added to list.

Directory, Cache and Interconnect Benchmarking Metrics

Interconnect traffic measurement metrics:

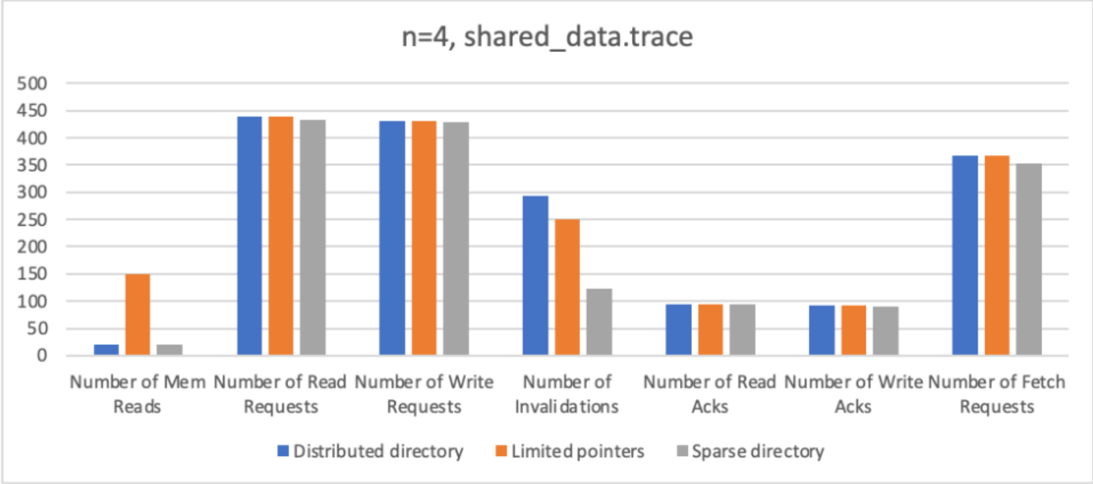
- Number of Memory Request
- Number of Read requests
- Number of Write requests
- Number of invalidations
- Number of Read Acknowledgments: In specific forwarding based read requests, we use an ack to ensure a transaction has been completed
- Number of Write Acknowledgments: In specific forwarding based write requests, we use an ack to ensure a transaction has been completed
- Number of Fetch Requests

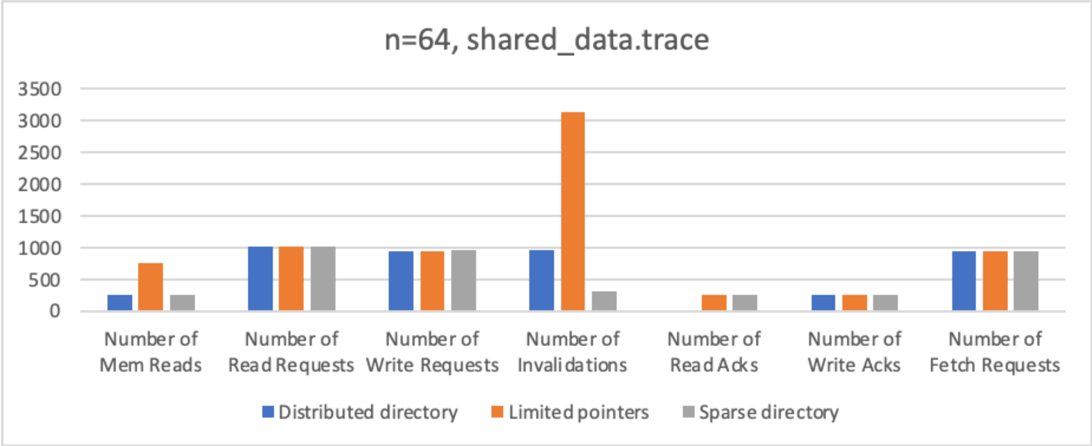
Traces:

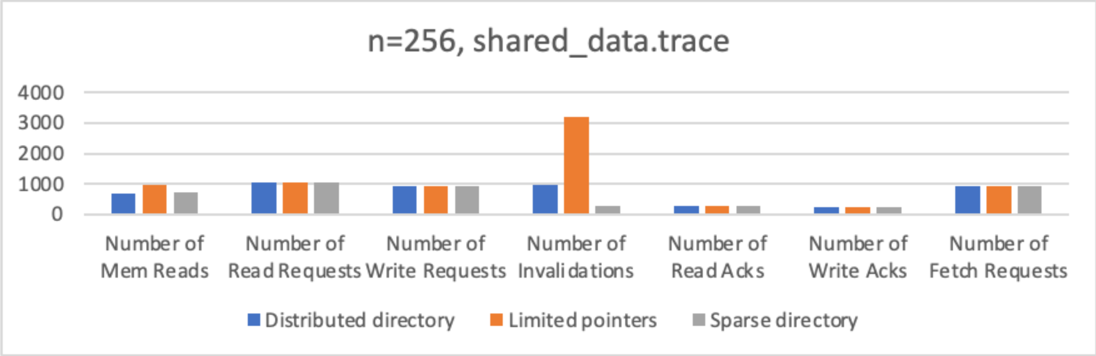
- Random: random access pattern.
- Shared Data: load and store the same cache block across processors.
- False sharing: processors access different addresses that belong to the same block(s).
- Producer-Consumer Problem.
- Matrix Transpose: access pattern of a parallelized matrix transpose operation.

Machine: GHC Machines

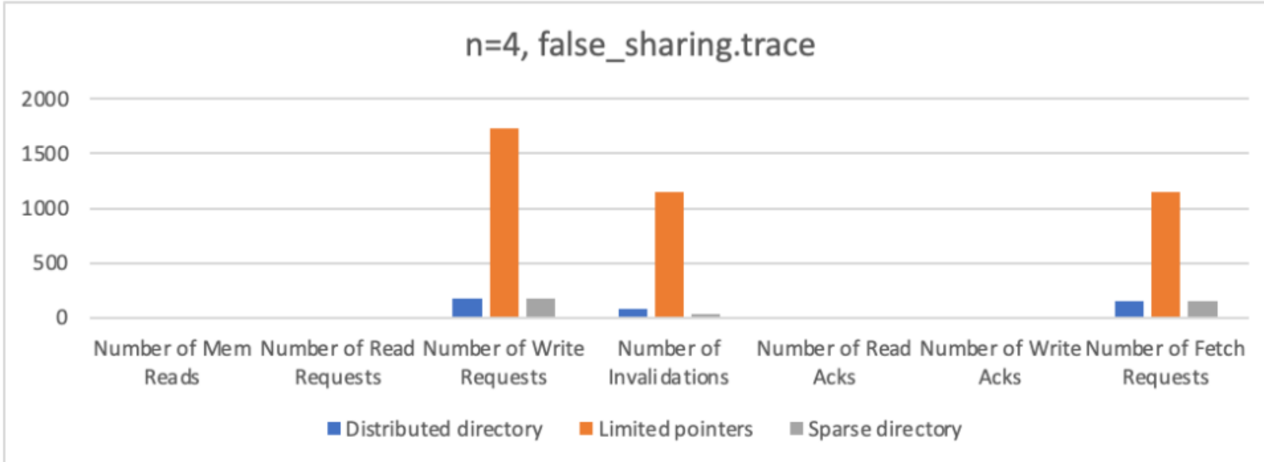
Shared Data Trace

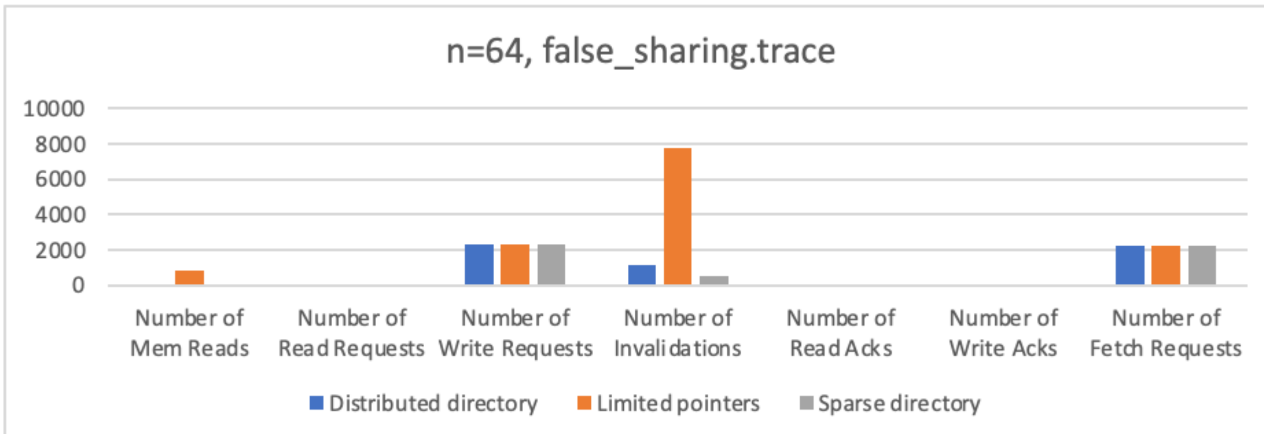


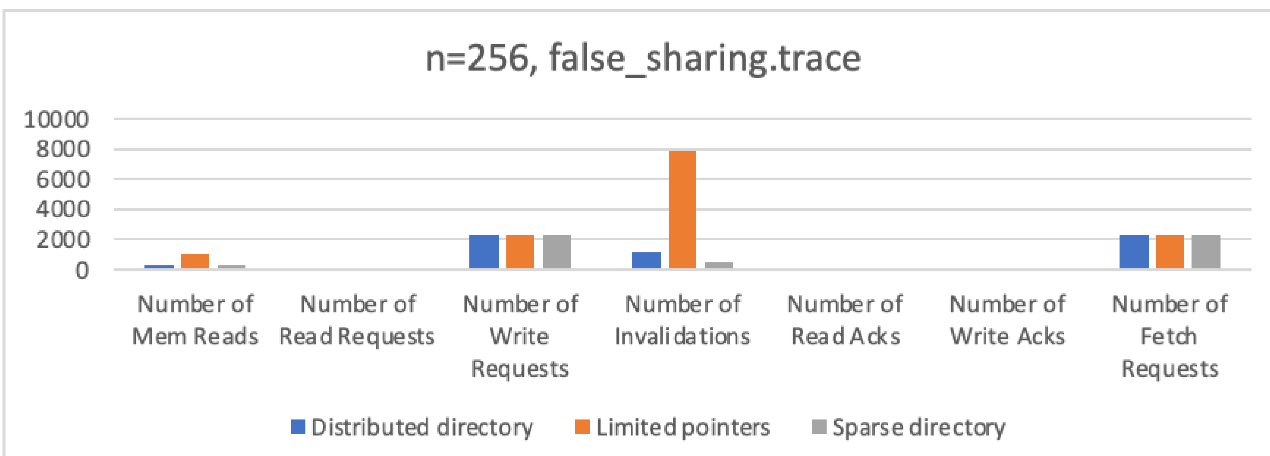




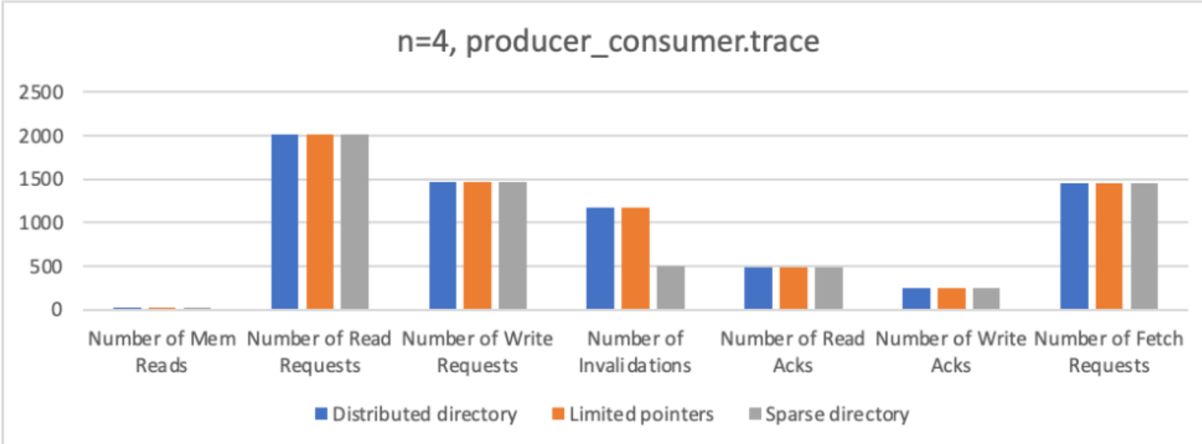
False Sharing Trace

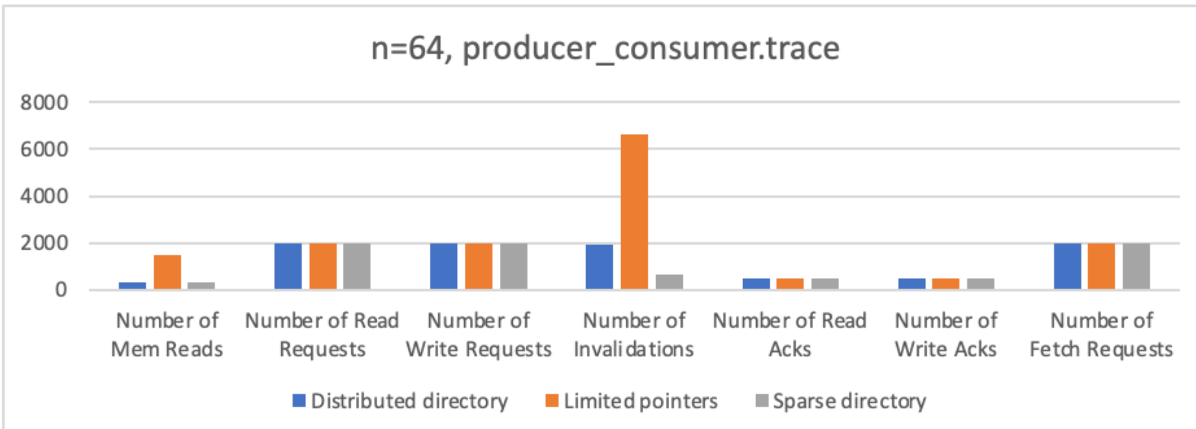


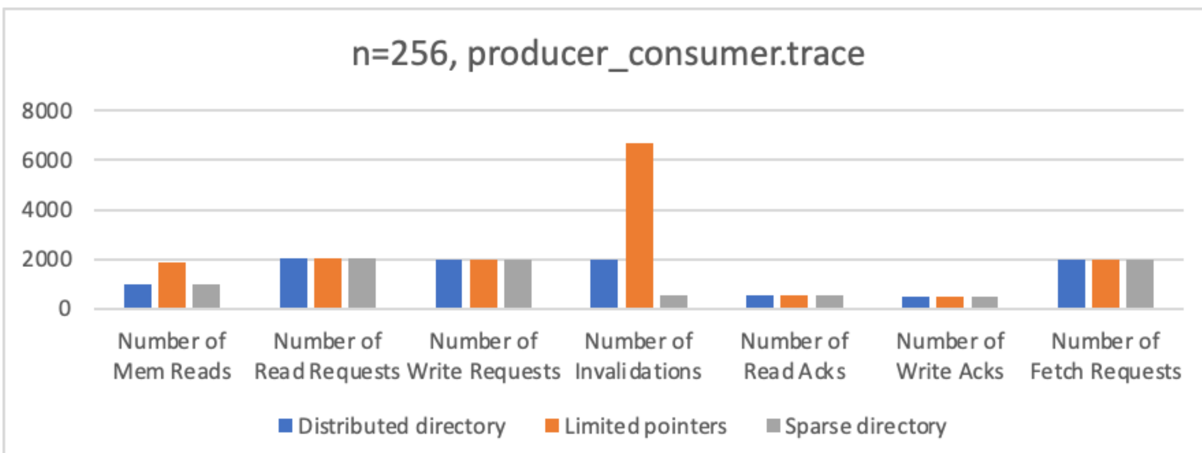




Producer Consumer Trace







Conclusions

Distributed Directory Scheme: Enhances NUMA performance by tracking cached lines, reducing coherence traffic, and minimizing data movement with distributed, localized directories.

Limited Pointer Scheme: Balances directory size and eviction complexity, ideal when few processors share cache lines, reducing storage overhead but less efficient under high cache sharing.

Sparse Directory Scheme: Memory-efficient for low sharing scenarios, this approach only tracks actively shared cache lines, optimizing directory space in large multicore systems with limited line sharing.

Suggestions for future work or improvements

- Multi-thread the sequential cache simulator implementations.
- Optimizing directory based coherence using intervention and request forwarding.

Limitations affecting Optimal Simulation / Performance

- Concurrency handling
- Design of the interconnect might not be optimal for simulating the data latency constraints that might be expected in real world