Asynchronous Systems
Cache Coherence - Project Plan

Karthik Reddy - 109721778

Project Description:

Comprehensive comparison of cache coherence protocols in DistAlgo[1].
Measure the performance metrics of various cache coherence protocols on benchmarks and compare with other protocols.

Problem Description

Why we need cache coherence?
Cache memory is fast access memory which is tightly integrated with the processor. It is random access memory and has significantly lower access latencies than regular main memory. Since, cache memory is expensive monetarily, size of caches is only order of KB or under 3MB. Since each processor has it's own cache, there may be inconsistencies in the shared data stored across some of the caches.

Cache coherence is the consistency of shared resource data that ends up stored in multiple local caches. Cache coherence algorithms guarantee consistency of data stored across all caches[2]. Few examples are Token Coherence[3], Tardis[4] and MESI[5] protocols. The requirements include: safety, starvation avoidance.

Applications where cache coherence algorithms are used are: Computer architectures(SPARC, X86), distributed shared memory systems like IVY[6]

In terms of computer architecture
Input: An architecture where each processor has its own local cache of memory, besides main memory.
Output: Consistency of shared resource data that ends up stored in multiple local caches

My Contribution

- Implement MESI [5] cache coherence algorithm in DistAlgo.
- Verify correctness. MESI protocol guarantees sequential consistency[7]. This will be verified in the implementation of MESI.
- Measure performance metrics(benchmarking).

- Compare and visualize performance of MESI protocol against other cache coherence protocols(to be done by team).

Benchmarks

It will contain per process sequence of loads and stores to shared memory. For now we have found a benchmark trace on the web at [8], but plan to include others in the future. We run these benchmarks on various cache coherence protocols to obtain the below mentioned performance metrics.

Performance Parameters

1. Time taken for the benchmark to run
2. Number of messages sent and received
3. Number of cache misses
4. Number of cache hits
5. Memory overhead for tracking states/tokens

Implementation Reference

The following two sources have MESI implementations.

http://www.m5sim.org/Ruby - This implementation is used in Token Coherence research paper [3]. Considered to be state of the art. Written in SLICC, it is a domain specific language that is used for specifying cache coherence protocols. Using many of the performance metrics mentioned in this implementation.

https://github.com/samuelbritt/CS6290-prj3 - another implementation written in C++.It is easy to understand implementation accompanied with test cases. Validate script runs the traces to check for correctness. It is also accompanied with python script to generate plots which can be used as visualization tool.

Evaluation Procedure

Comparing with other teammates on the above mentioned performance parameters for varying benchmarks. Visualize performance metrics against number of processors, as plots.

Individual Project Plan

Week 1:
Project planning with other team members
Brainstorm on the cache coherence protocols.
Decide common performance evaluation matrices
Reach a consensus about the scope of the project.
Understand the protocol MESI theoretically
Better understand the existing protocol implementation mentioned in state-of-art section.

Week 2:
Implementing MESI protocol in DistAlgo.
Correctness testing of the implementation.
Evaluation: Compare the implementation with other team members protocols on basis of the performance metrics mentioned above.

Week 3:
Organize the knowledge gained from the comparison of the algorithms.
Make Visualizations: Graphs/Charts varying cache sizes across algorithms to facilitate the presentation.
Make team presentation.

Team Member's role

Parag Gupta : MSI
Karthik Reddy : MESI
Paul Mathew : MI, feasibility Token Coherence
Amit Khandelwal : MOESI
Garima Gehlot : MOSI

References

[1] Yanhong A. Liu, Bo Lin, and Scott Stoller. "DistAlgo Language Description". https://github.com/DistAlgo/distalgo
[2] https://en.wikipedia.org/wiki/Cache_coherence
[3] Martin, Milo MK, Mark D. Hill, and David Wood. "Token coherence: Decoupling performance and correctness." Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on. IEEE, 2003.
[4] Yu, Xiangyao, and Srinivas Devadas. "TARDIS: Timestamp based Coherence Algorithm for Distributed Shared Memory." arXiv preprint arXiv:1501.04504 (2015).
[5] Mark S. Papamarcos and Janak H. Patel. "A low overhead coherence solution For multiprocessors with private cache memory"
[6] Li, Kai. "IVY: A Shared Virtual Memory System for Parallel Computing." ICPP (2) 88 (1988): 94.
[7] http://www.cs.jhu.edu/~gyn/publications/memorymodels/node6.html
[8] https://github.com/samuelbritt/CS6290-prj3/tree/master/traces