Empirical Evaluation of Cache Coherence Protocols

Group 5: Jatin Dev, Swapnil Raykar

1. INTRODUCTION

In a shared memory multiprocessor system with a separate cache memory for each processor, it is possible to have many copies of shared data: one copy in the main memory and one in the local cache of each processor that requested it. When one of the copies of data is changed, the other copies must reflect that change. Cache coherence protocols ensures that the changes in the values of shared operands(data) are propagated throughout the system.
Different Coherence Protocols impact different parameters of the system performance. The goal of the Project is to evaluate the most commonly known protocols **MSI,MESI and MOESI**. The Parameters used to evaluate performance are **Bus Traffic generated,Cache Reads,Memory writes, Coherence invalidations and avarege time to complete bus request and reply transactions.**

2. ASSUMPTIONS

We have assumed a Snoopy bus based model with cache to cache transfer.We have used three level memory model where first level is Private cache of the each processor,second level is LLC(Last level cache) and third level is DRAM and hierarchy is inclusive. We have implemented coherence protocols between private caches and LLC.

3. IMPLEMENTATION DETAILS

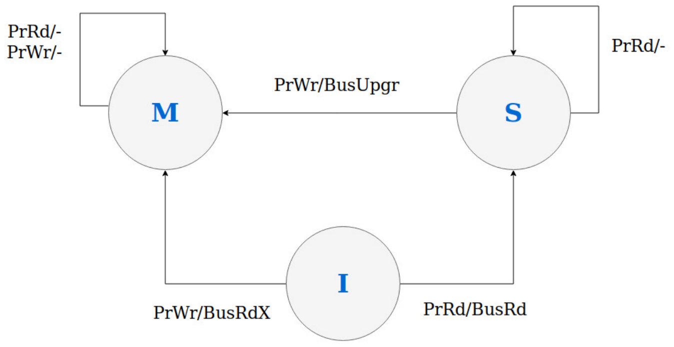
- We have used two Parsec applications Fluidanimate and strem cluster to generate multithreaded traces.

- we have calculated following parameters for every protocol
  Bus transactions: Counts all the BusRdx/Bus Rd and Bus upgrade transactions.
  Cache Reads: Counts number of times a value is read from the other processor cache
  Memory write: Number of times a value is written to the LLC or DRAM
  Average Request and Response waiting time: Compute average time to request and reply a bus transaction respectively.
  Coherence invalidations: Number of invalidations generatedin other processor's private cache.

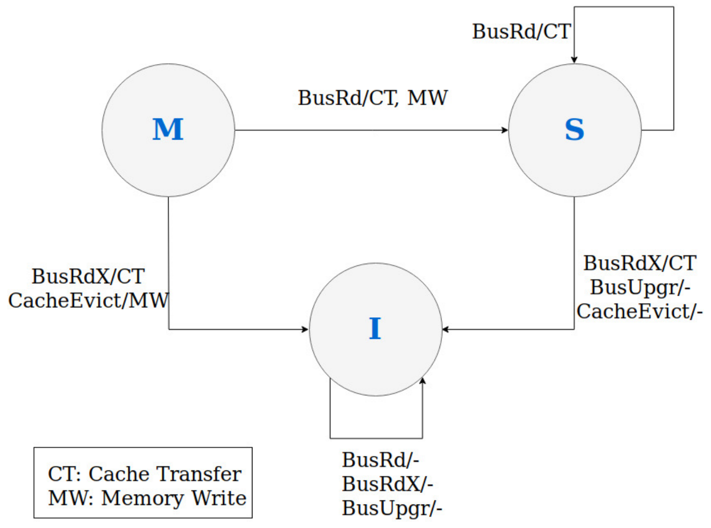
- For measuring average waiting time for **bus reply** transactions we maintained a private clock for each processor which gets incremented every time address of that processor is read.

- And to measure average waiting time for **bus request** transactions we maintained a single clock which gets incremented whenever a new address is read.Service time to lookup cache when transaction arrives on bus is assumed 5 and bus request queue and bus reply queue are assumed 60 and 20 respectively.

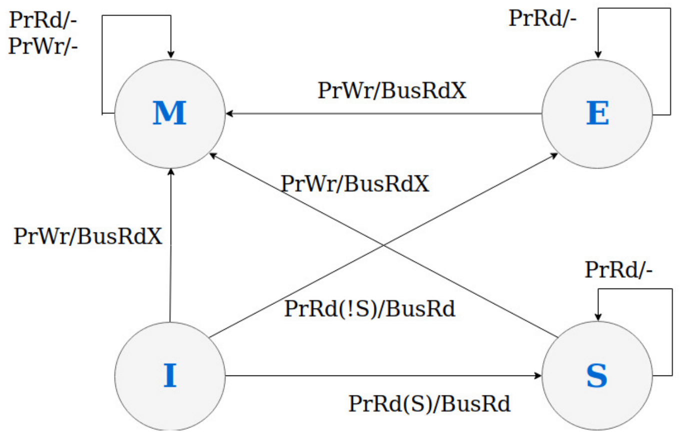
We have implemented the MSI,MESI,MOESI Cache coherence protocols as shown in the state diagrams. For every protocol ,we have shown two different state diagrams one for processor request and one for Bus transactions.



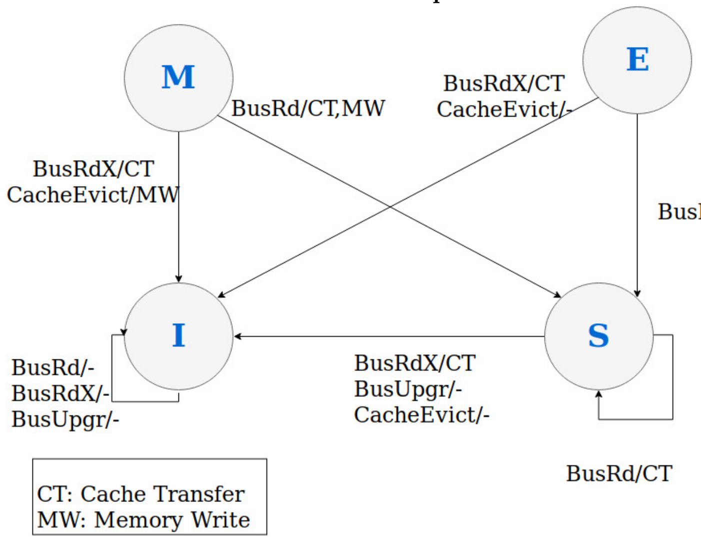
1.MSI Processor Request



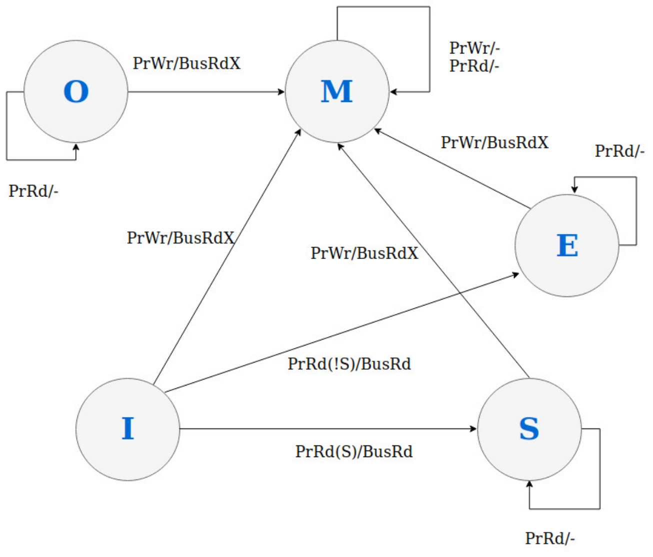
2.MSI Bus Transactions



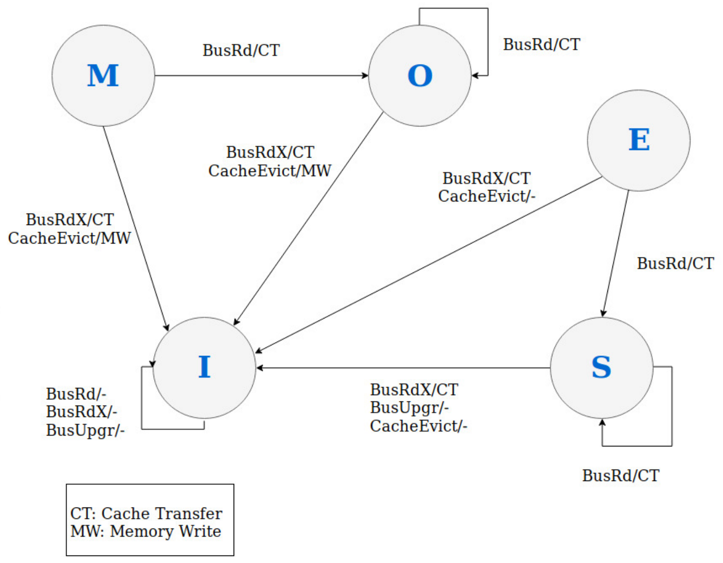
1.MESI Processor Request



2.MESI Bus Transactions



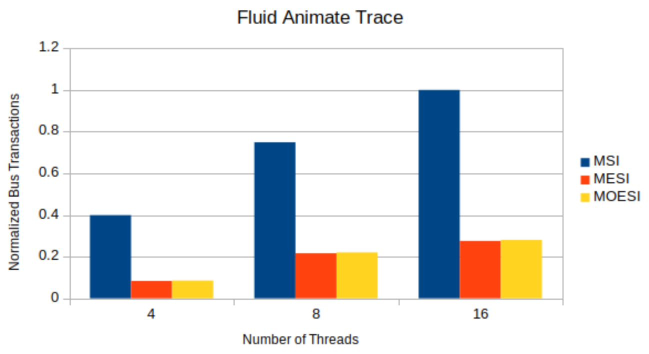
1.MOESI Processor Request



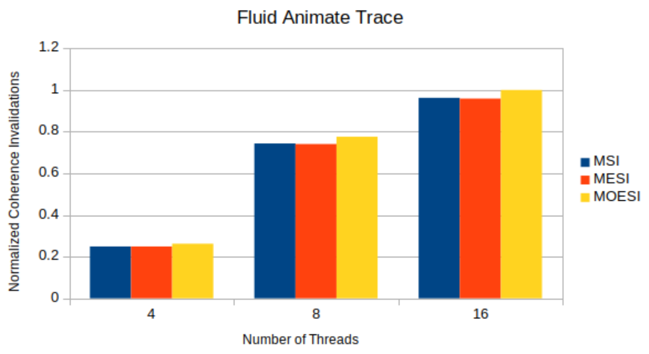
2.MOESI Bus Transactions

4. RESULTS

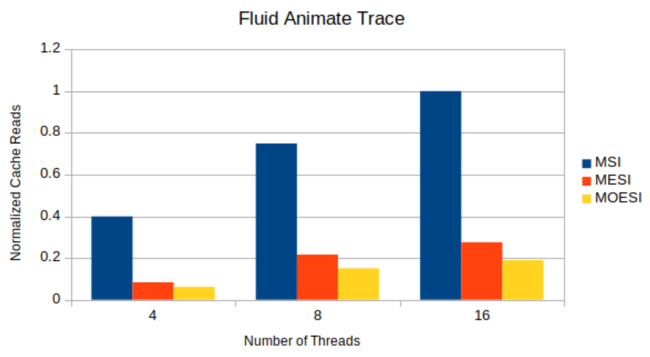
In below graphs we have shown comparison of different coherence protocols i.e.MSI,MESI and MOESI for both Trace We have used L2 sets 1024 and 8 ways for every processors and for LLC 2048 sets and 16 ways.



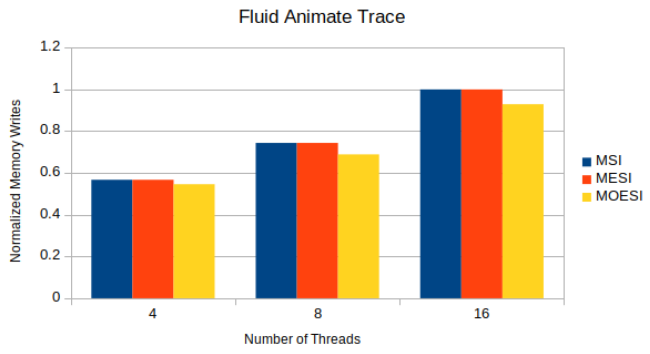
1.Bus transactions for Fluid Animate Trace



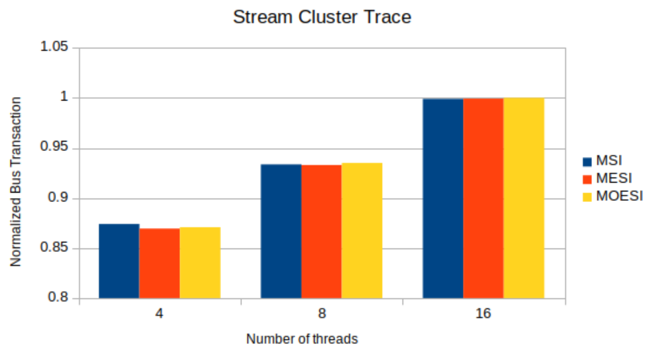
2.Coherence invalidations for Fluid Animate Trace



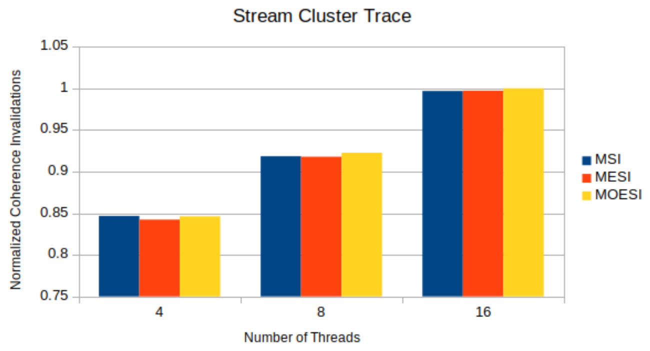
3.Cache Reads for Fluid Animate Trace



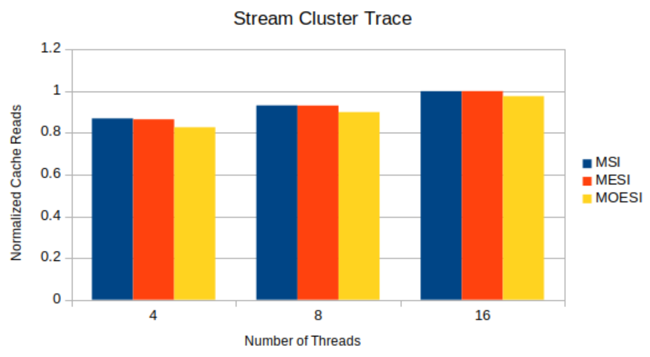
4.Memory Read for Fluid Animate Trace



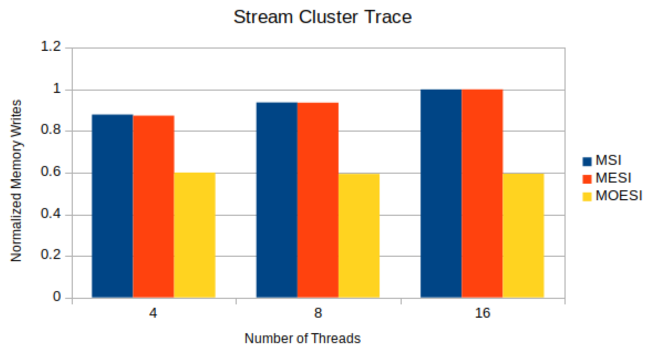
5.Bus transactions for Stream Cluster Trace



6.Coherence invalidations for Stream Cluster Trace



7.Cache Read for Stream Cluster Trace



8.Memory write for Stream Cluster Trace

Following are the observations from above graphs:

1. **Bus Transactions:** Number of bus transactions is more in MSI than MESI and MOESI. E state has reduced the number of bus transactions operation where a processor reads and then writes the same block.As the number of threads increases the bus transactions also increases.

2. **Memory Writes:** Memory writes is less in MOESI as compared to others. In case continuous reads and writes operations are performed by various caches on a particular block, then the data has to be flushed on to the bus every time and written to memory. But MOESI reduces this with the help of O state, now the block can go from M to O when read is performed after writes and hence saves the memory writes.

3.**Cache Reads:** Number of Cache Reads is more in MSI than MESI and MOESI because it has more number of bus transactions.

4.**Average waiting time for bus Request and Reply:** We have found that Average waiting time for MSI is much larger than MESI and MOESI because it has more number of transactions. While the average waiting time for MESI and MOESI is almost similar.