Github References.

- Amughrabi sources
- jk 78346.

Project Report

Sweta Subhra Datta SS Datta

NC State University
Department of Electrical and Computer Engineering

Name -Sweta Subhra Datta Unity_id-ssdatta. 200374821
Project Report on Cache Coherence Protocol

Project Overview ::
This Project explores the concept of Cache Coherence using the MSI,MESI and Dragon Protocol.
In this project I have created 4 caches with change in states for senders and receivers for each cache for every protocol.

TestCases:
Current status -All given test cases passed

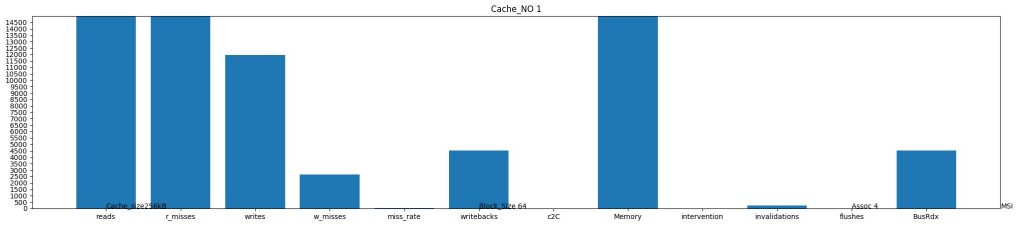
Performance Measures:
I made a separate python script called Gen_Graphs.py which reads the output file and creates graphs to keep track of configuration of every cache for each protocol.

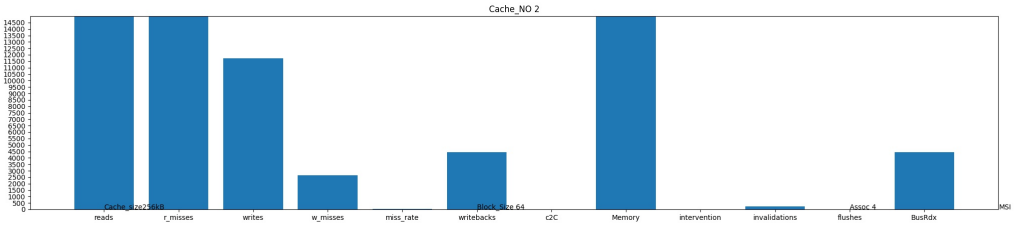
◉ *Performance Measure Type1) Each Cache State Varying Cache Size size and varying associativity with multiple protocol(from automated graph_greneration.py)*
◉*{Mix and Match Style}*

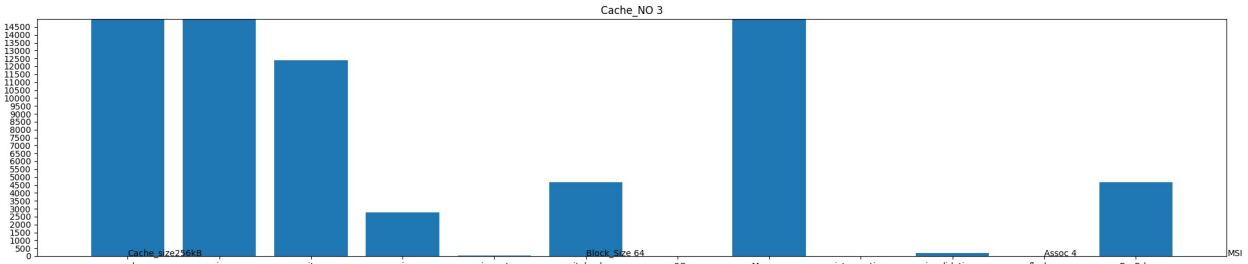
Getting the states of every cache is tedious for every parameter and also its very difficult to study and attach the plot to a report.
However I am still attaching the plots which Incas able to do for

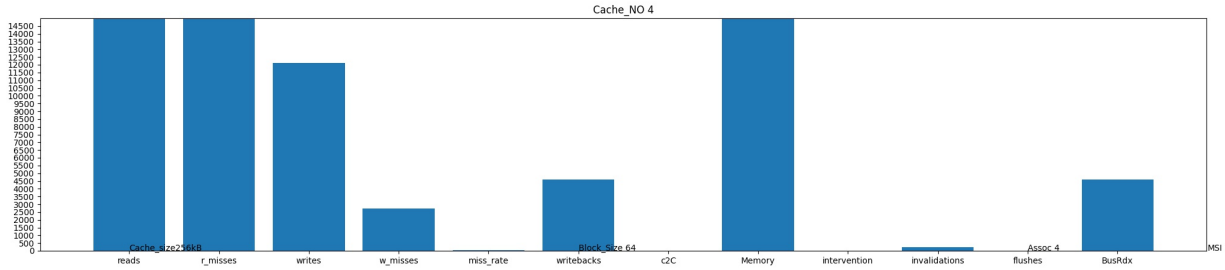
1)Size 256KB Caches with associativity 4:: MSI protocol
2) Size 8 MB Cache with associativity 8:: MESI protocol.
3.) Size 8 MB Cache with associativity 16:: DRAGON protocol.
Later I have only attached plots of varying parameters with protocols

Each Cache state for 256KB assoc 4 MSI

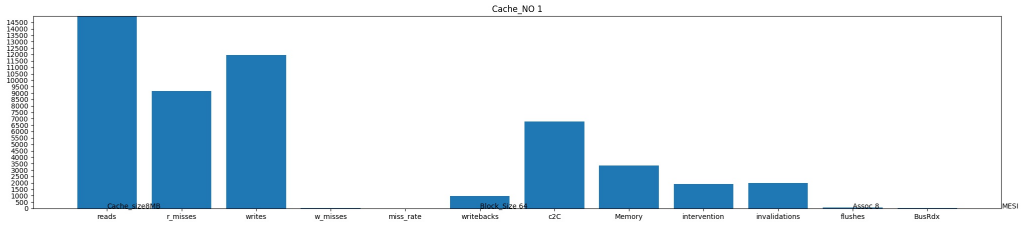


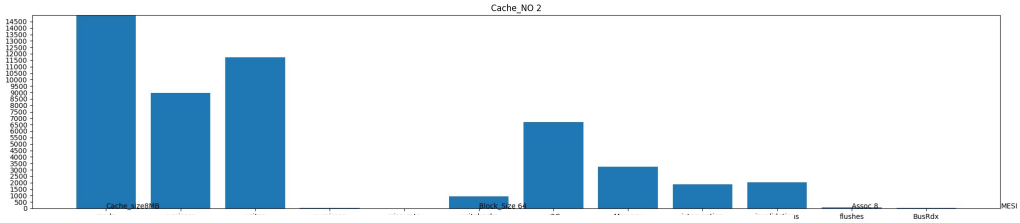


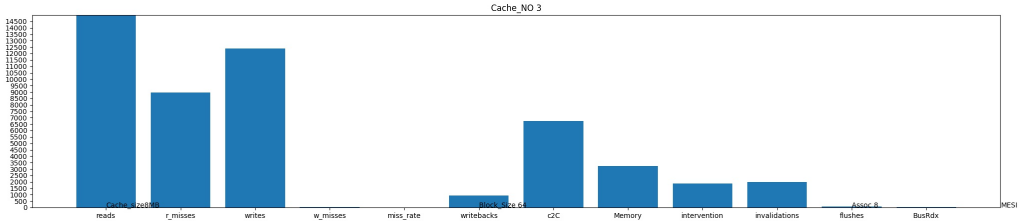


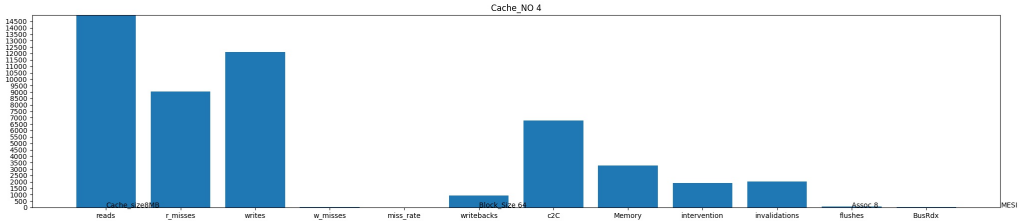


Each Cache state for 8MB assoc 8 MESI

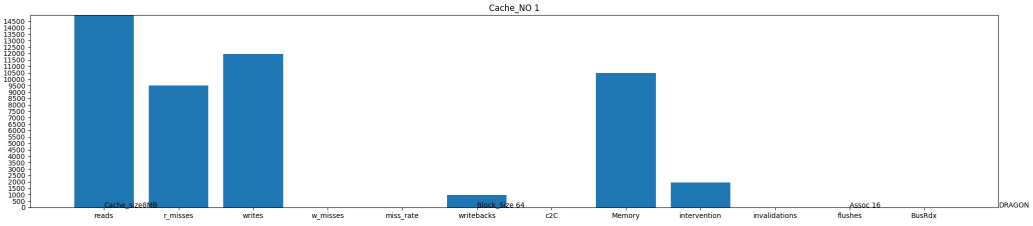


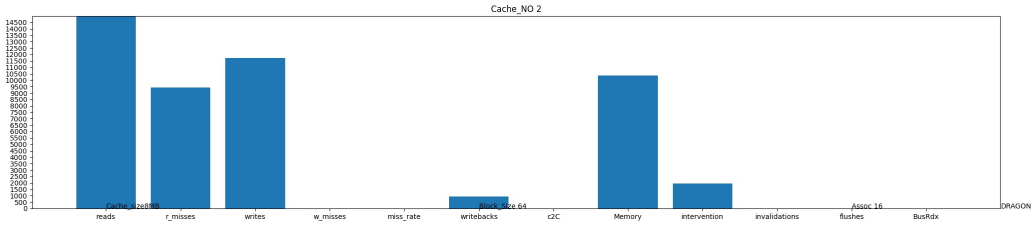


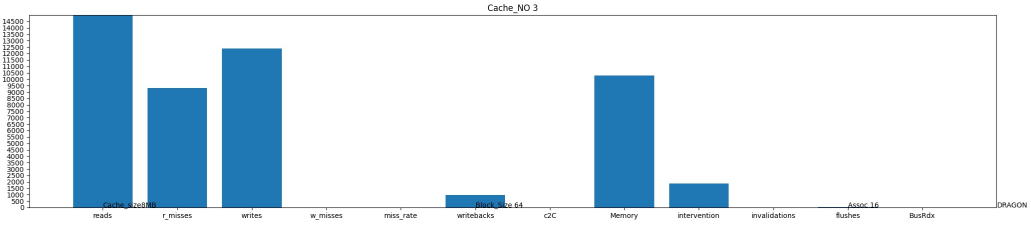


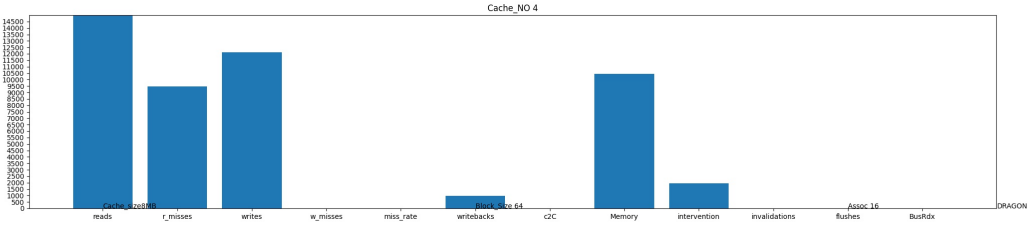


Each Cache state for 8MB assoc 16 ::Dragon





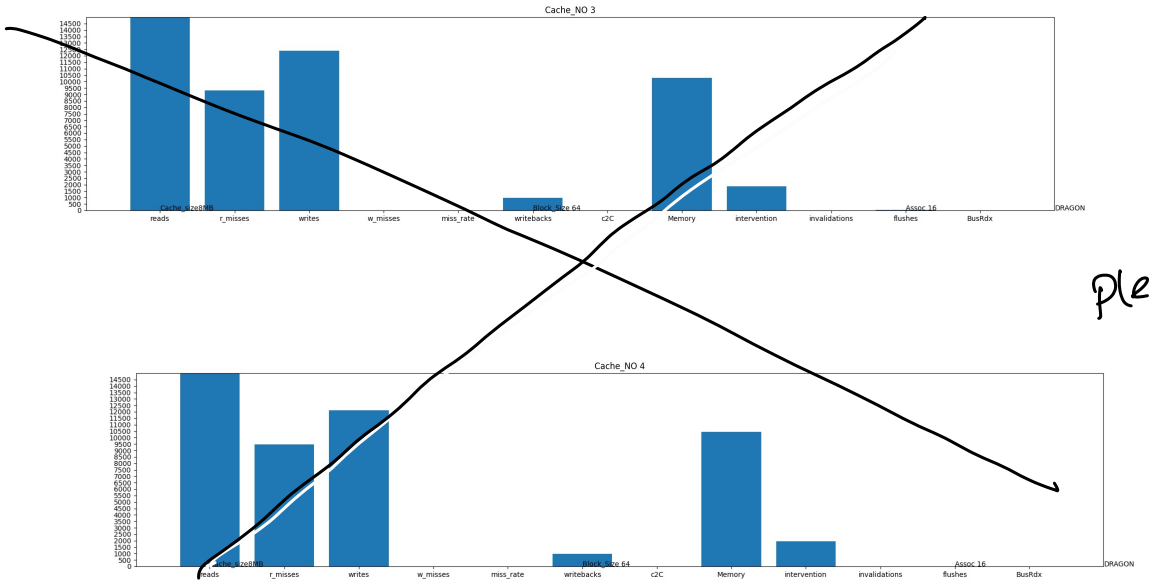




Inference from Performance measure 1:

In this performance measure I tried to present a higher level over view of each cache during each protocol when some parameter changes, a stark difference can be noted when seeing dragon protocol and MSI and MESI.Also notice the change in bus transaction for MESI and MSI.

Seeing the mix and match style of graphing the main advantages of using each protocol comes into view but we will go deeper with other performance measure in the next part.



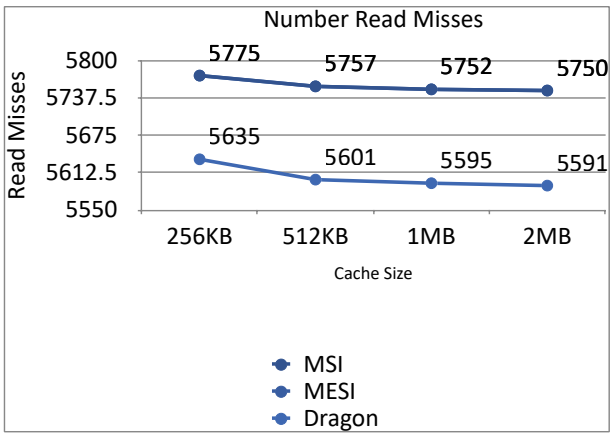
Please on 7 '

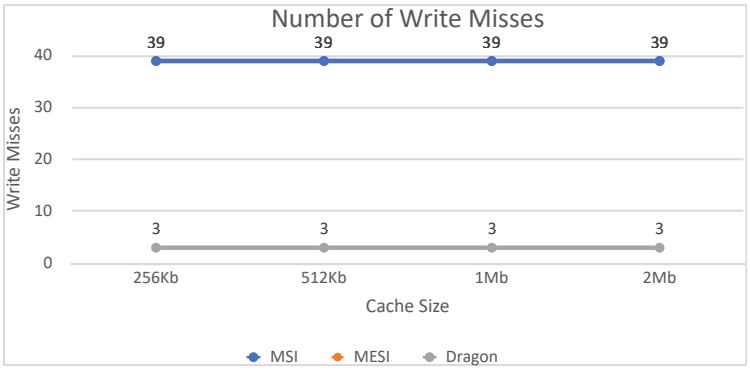
Inference from Performance measure 1:

The smaller the cache size miss rate increased a litle bit and so was the memory transaction but a more pronounced effect with size was seen with Bus transactions as the size increased from 256 KB to 8 MB we see a drastic increase in Bustransactions.

Increasing associativity has more pronounced effects on number of write backs and missrate(read and write misses)

Performance Measure Type2)<u>Varying Cache size for each parameter(these graphs were generated using Excel from the outputs)</u>

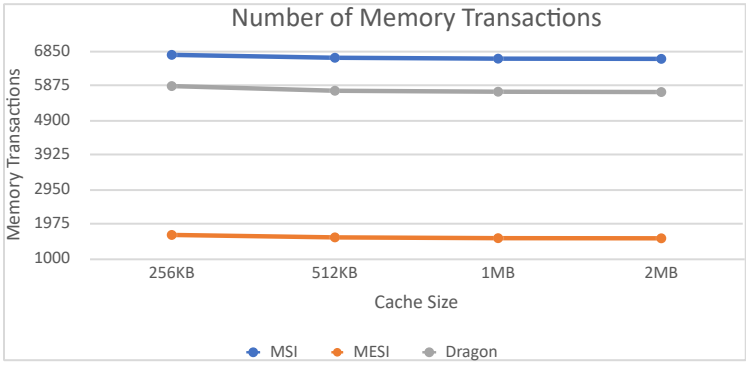


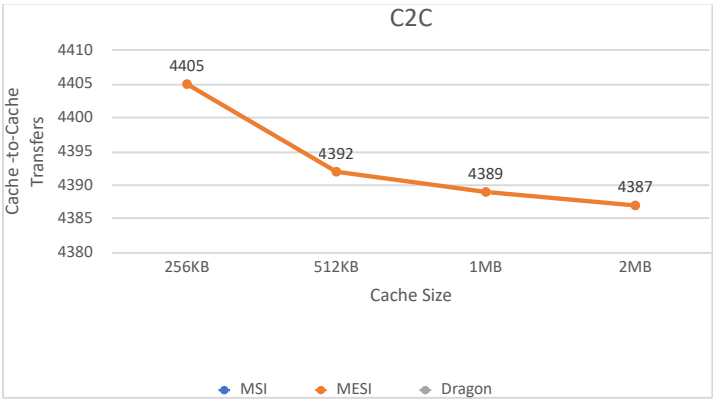


As we can see with increase in cache size the number of read and write misses will decrease but again not by huge margin.Here we even see that there is no effect on write misses but also the no of writes in the trace file was significantly less .

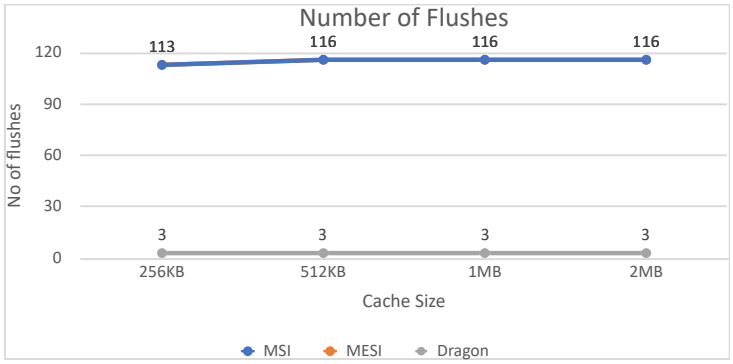
Also as Dragon is update based protocol with increase in cache size we will have less capacity misses which was a weak point in dragon.

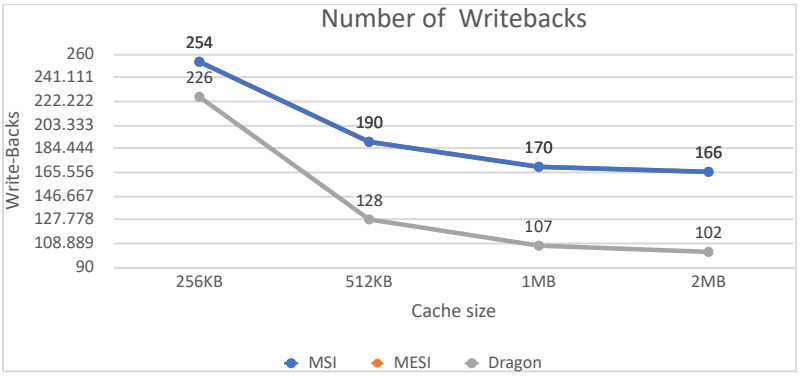
So as cache size increase RWmiss for Dragon< RW miss for MSI and MESI.



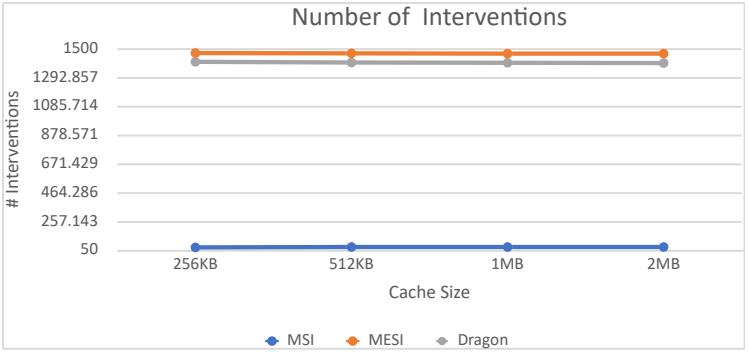


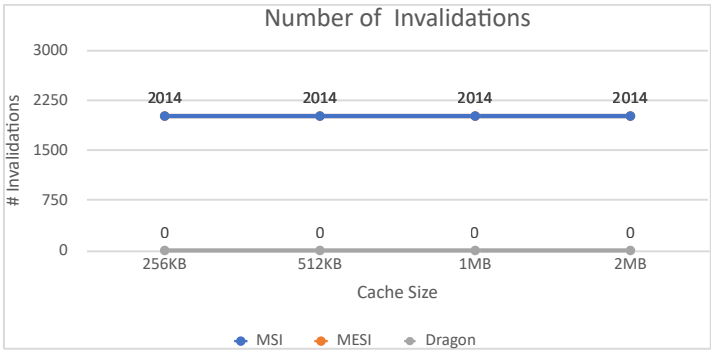
The number of memory transaction for MSI and Dragon is greater than MESI and C2C is only present for MESI and this serves as an inference for the former graph.
Due to Cache to cache transfer in MESI there is very less memory transactions for this protocol as compared to MSI and Dragon.



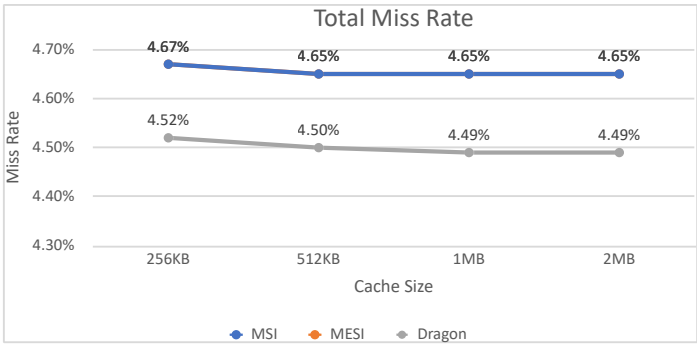


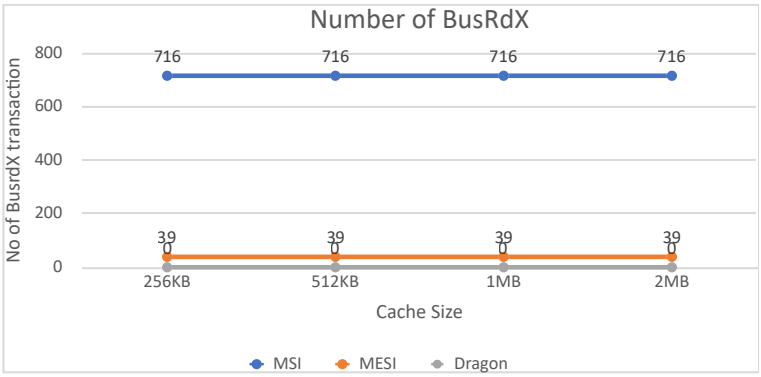
Dragon has the least number of flushes as compared to MSI and MESI ,in case of write backs we see that Dragon has the least write back mainly because it only writes from Sm state (when block is evicted).MSI and MESI shows almost same performance in the above graphs and Dragon comes out to be the winner.





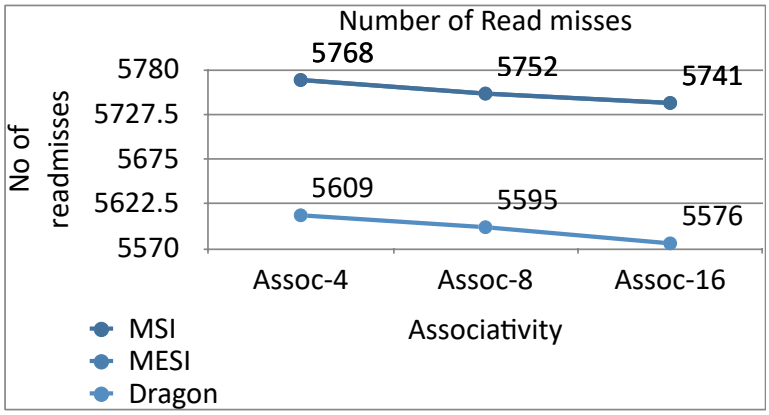
Dragon is update based and invalidations doesn't takes place here while MESI and MSI shows same no of invalidations which makes sense if we see the state machines of both the protocols. (i.e the invalidations doesn't change with change in size).
The number of interventions is larger for MESI and Dragon protocol as the intervention will increase when modified or exclusive state will go to shared state in MESI and shared clean and shared modified in Dragon protocol. But in the case of MSI, it will happen when the state is being changed from Modified to shared. Thus the number of interventions is less in MSI protocol

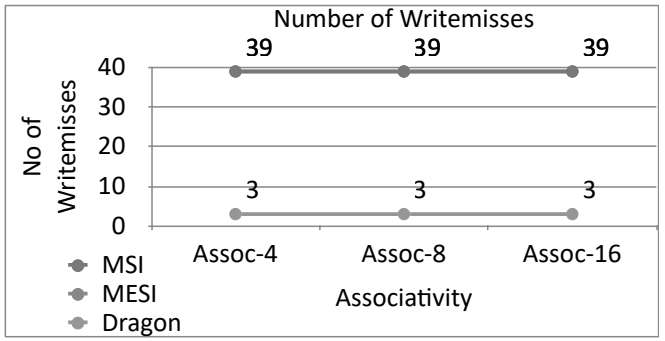




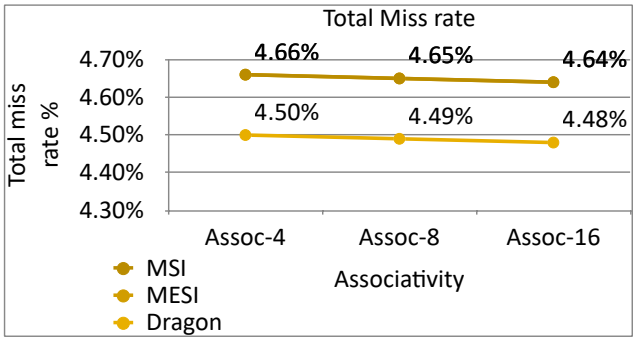
Dragon being update based protocol has no BusRdx but we can see here MESI BusTransactions>> MSI Bus-Transactions which was one of the main reason for choosing MESI over MSI.
Dragon will have lesser miss rate as its an update based protocol and we are increasing the cache size.MESI and MSI almost have the same miss rate.
one thing to note here is the miss rate decrease is surprisingly very less for Dragon with increase in size.So increasing the cache size with dragon is not such a great option and increasing cache sizes is a costly affair and just for an advantage <1% miss rate

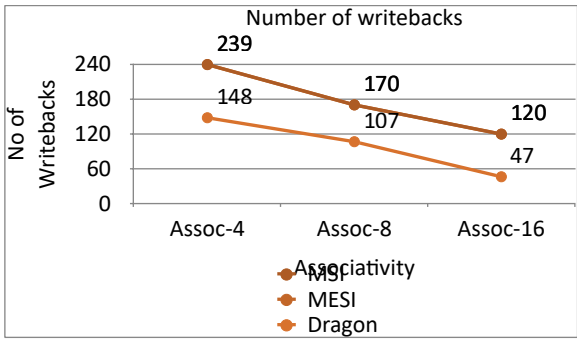
Performance Measure Type3)<u>Varying Associativity for each parameter(these graphs were generated using Excel from the outputs)</u>

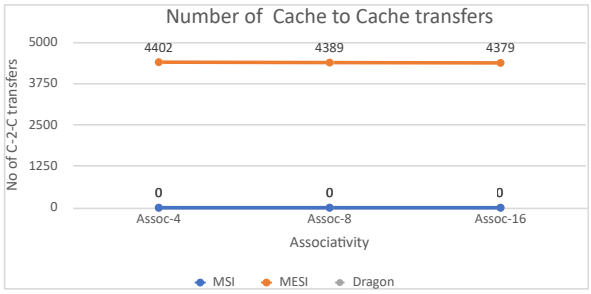


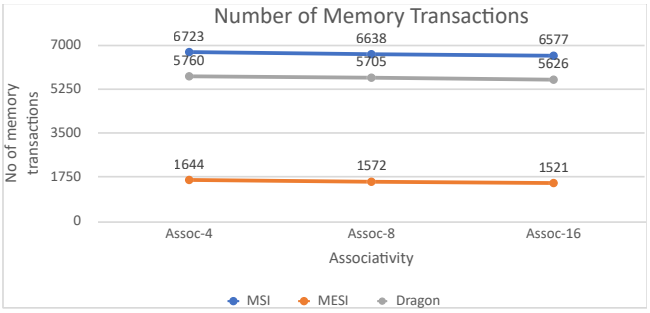


As we have seen from the performance measure type 1 with block size 8MB increasing associativity will help in reducing the read misses and write misses.Write misses are significantly less for Dragon mainly because there is no invalidations in Dragon hence blocks are not lost and in a way also maintains a good amount of temporal locality.
We can also see that decrease in read misses is more pronounced when associativity goes from 1 to 16 but not that much when going from 8 to 16







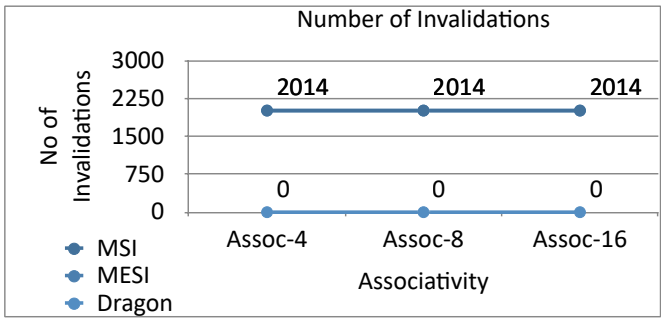


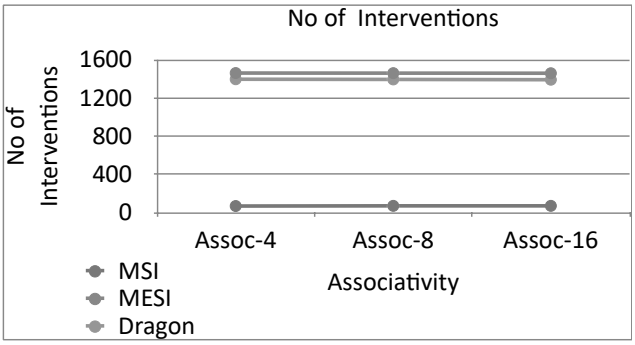
Total Miss rate-Increase in assoc will decrease the miss rate.Dragon being an update based protocol maintains a greater amount of temporal locality as its recent blocks are not invalidated and hence enjoys a decreased miss rate.

Total writebacks-Writabacks decreases as associativity will increase true for every protocol.But Dragon enjoys less write backs as associativity increases as write back is occurring only from one state (Sm) in such a large cache.

The cache-to-cache transfers are only present in MESI protocol and it is decreasing. Cache to cache transfer happens when there is read or write misses and thus as these misses are decreasing with the increase in associativity. The number of C2C also decreases.

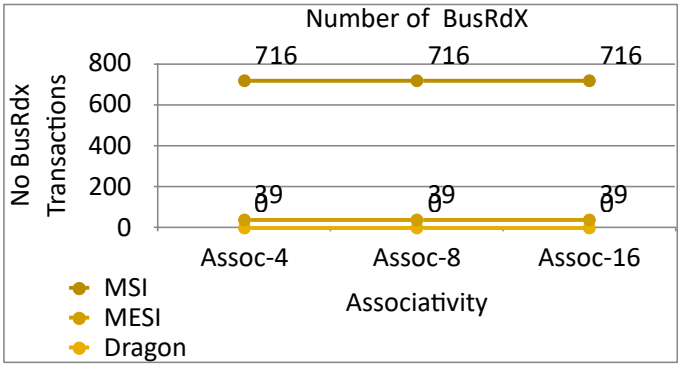
Mem transactions are greater for MSI>MESI>Dragon .MESI>MSI because the former has c2c transfers .Dragon is lesser of all because of less number of invalidations

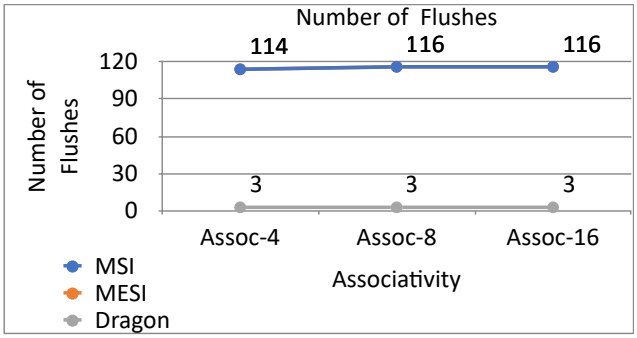




Invalidations does not change the assoc.

The number of interventions is larger for MESI and Dragon protocol as the intervention will increase when modified or exclusive state will go to shared state in MESI and shared clean and shared modified in Dragon protocol. But in the case of MSI it is only when M->S.

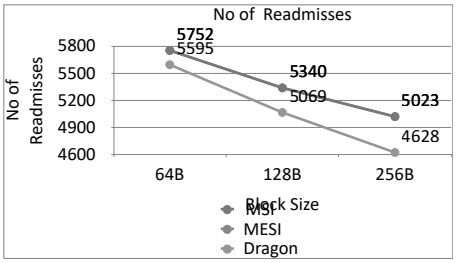


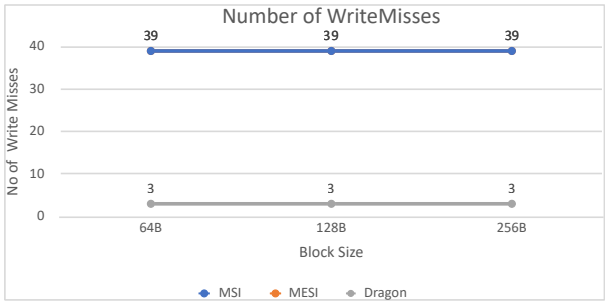


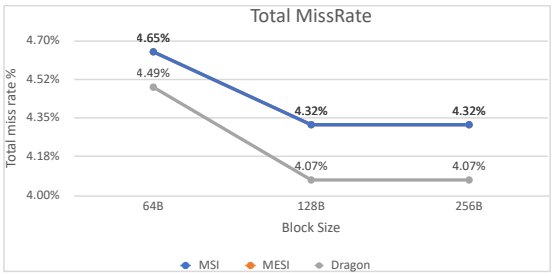
Busrdx are significantly lesser for MSI and MESI when Assoc is increased.MSI still have high Bus transactions because of no E state which is present in MESI.Dragon has no BusRdx

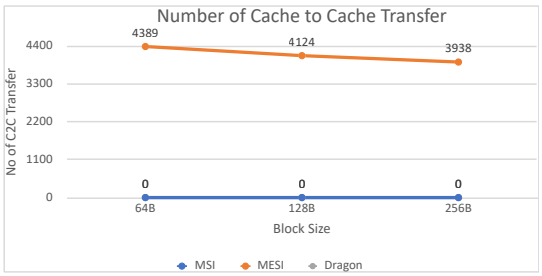
Flushes have increased for MSI and MESI with increase in cache size and it will also increase with increase in cache associativity. Dragon on the other hand enjoys quite a few flushes because of its optional flush protocols and Sm state

Performance Measure Type4)<u>Varying Block size for each parameter(these graphs were generated using Excel from the outputs)</u>

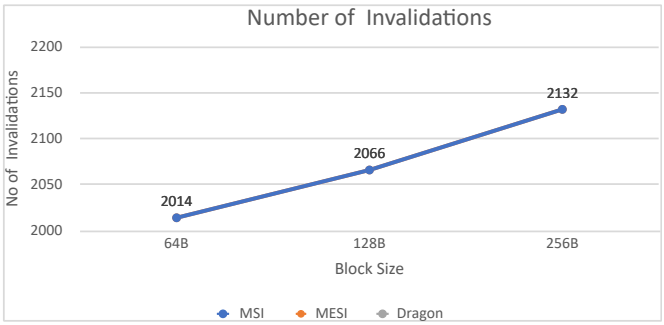


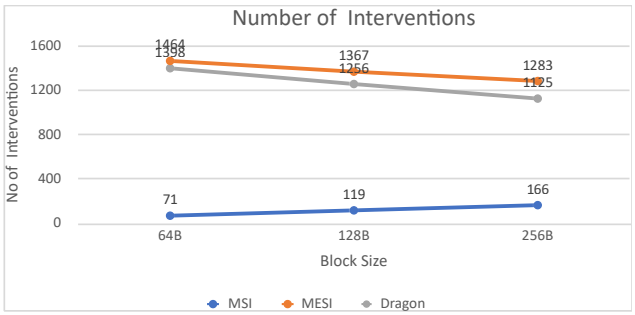






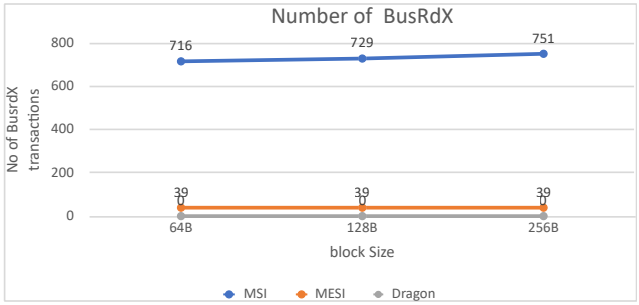
Increase in block size will decrease the no of read and write misses but to by a huge margin.Increase in block size serves good for dragon as it is update based and no invalidations these also decreases its capacity misses .Same reason can be drawn out for write misses and total miss rate.
C2C decreases as block size increase in MESI as more cache hits occurs.

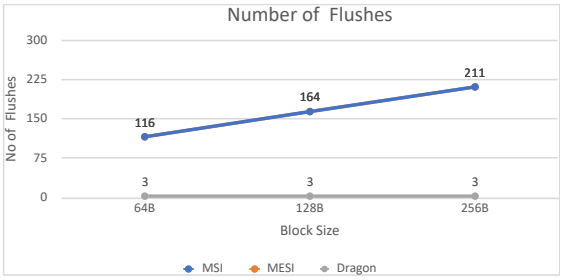


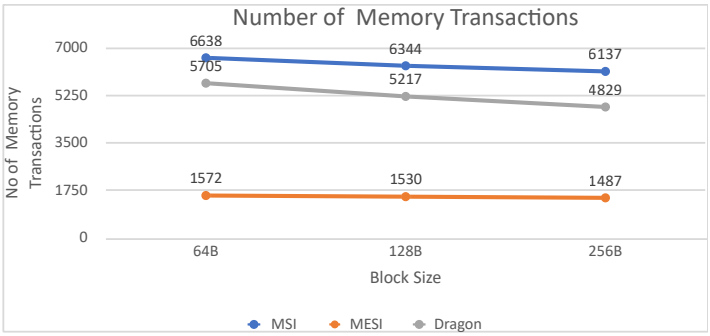


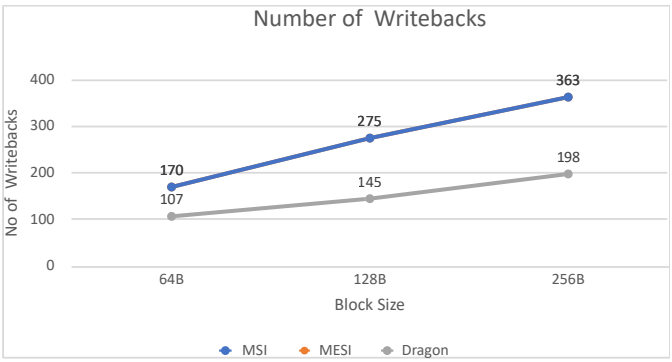
The number of interventions is decreasing for both Dragon and MESI protocols but there is an increase for MSI protocol.

Invalidations happen only in MSI and MESI protocols as it has invalid state. Invalidations are increasing with increasing the block size. This pattern is slightly different from what we have









Memory transactions will be reduced for any protocol with increase in block size.
No of flushes (MSI and MESI) and write-backs(Dragon,MESI,MSI) will increase with increase in block and cache size .See performance measure type1 (for detailed comparison)
BusRdX is increasing for MSI with an increase in block size but is almost constant for MESI.Dragon doesnt have Busrdx.

<u>Final Inference</u>-Dragon is looking good on pen and paper but is obsolete mainly because idragon works best when cache size is big enough otherwise dragon suffers from conflict and capacity misses.
In reality performance measure of MESI and MOESI are promising and its morphed versions like MESI-F ,MOESI-F currently used in many industries

Currently Dragon+ firefly protocol is obsolete