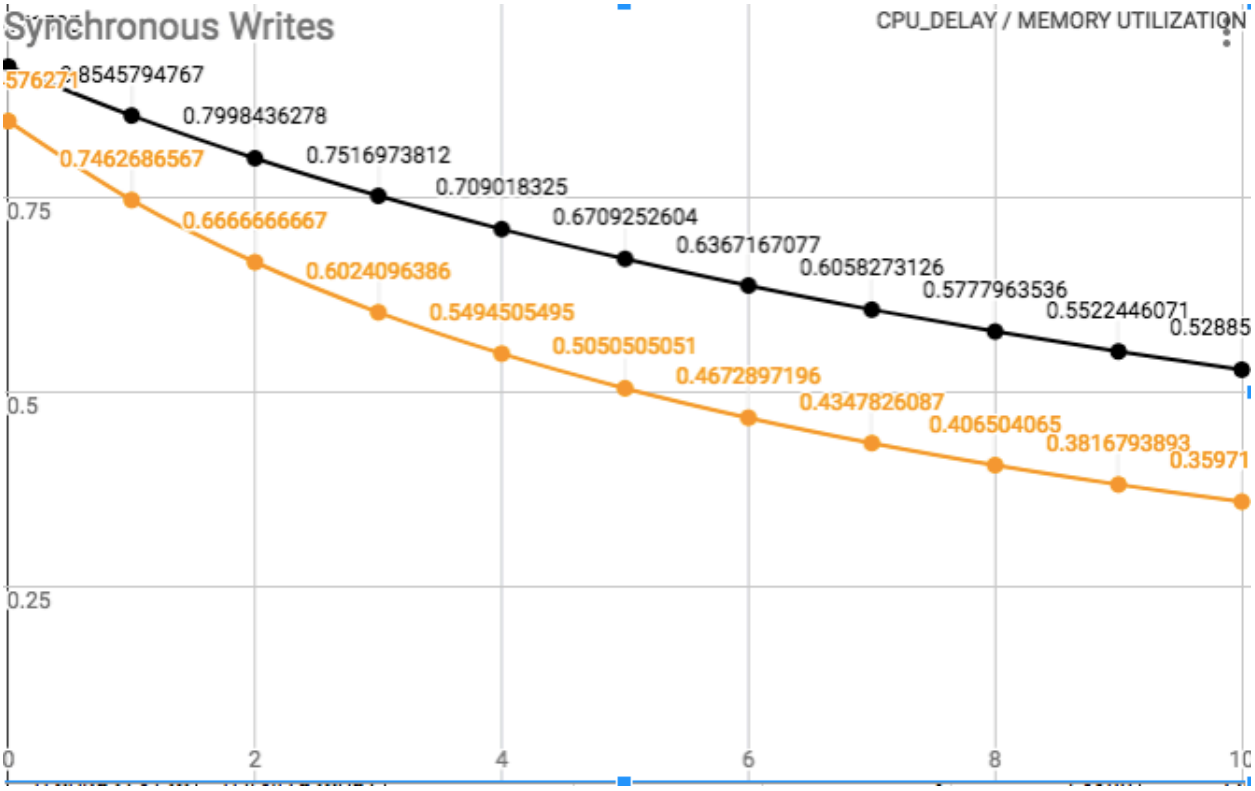
COMP 540

Mo Tang S01276578

1.Plot 1

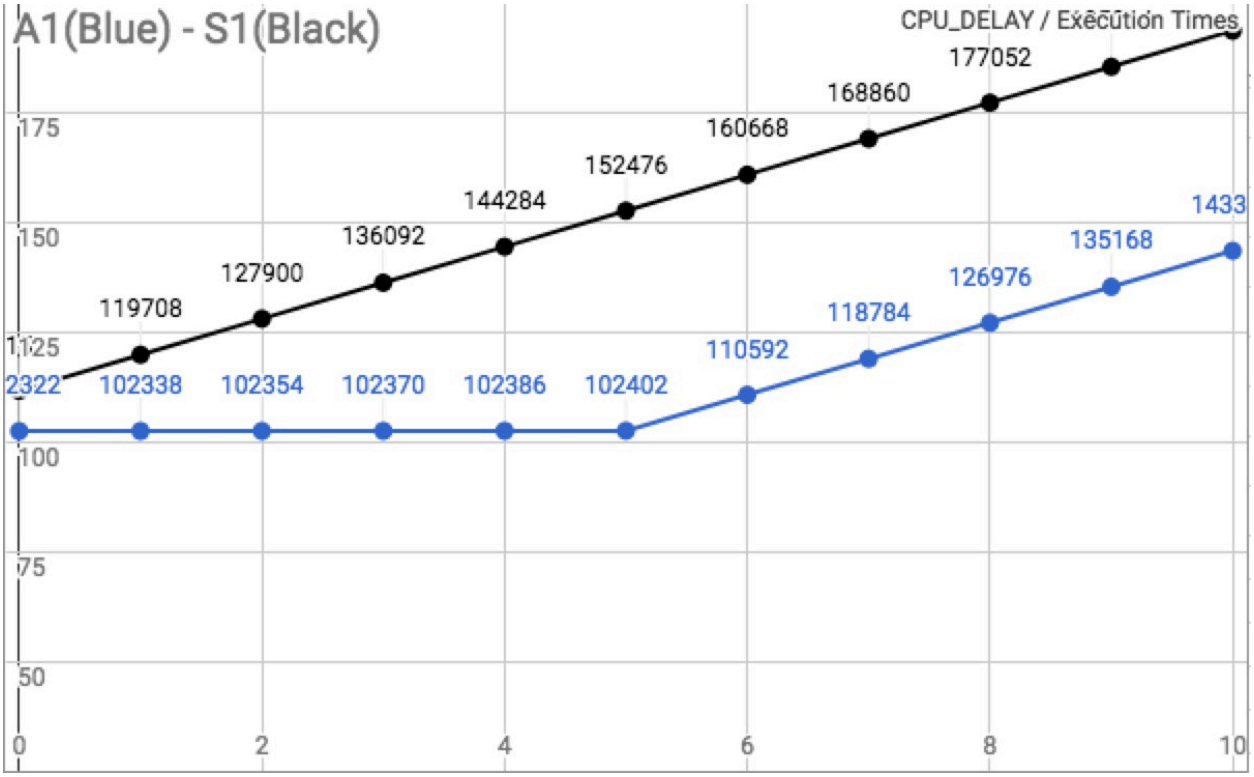


(i) We first analyze two curves separately. In the black curve, we can see that the memory utilization decreases with the increasement of CPU_DELAY. The time of write/read operation won't be affected by the CPU_DELAY, but the process time of CPU increases. So it can only wait for CPU, which increases the memory idle time. The same goes for orange curve.
As the CACHSIZE increases (BLKSIZE remains the same) to 2048x32 bytes, which is large enough to store all the data, then the system only need process read operation for once (the time of read operation then becomes 0). The idle time won't be affected, so the utilization will decrease.

(ii) 0

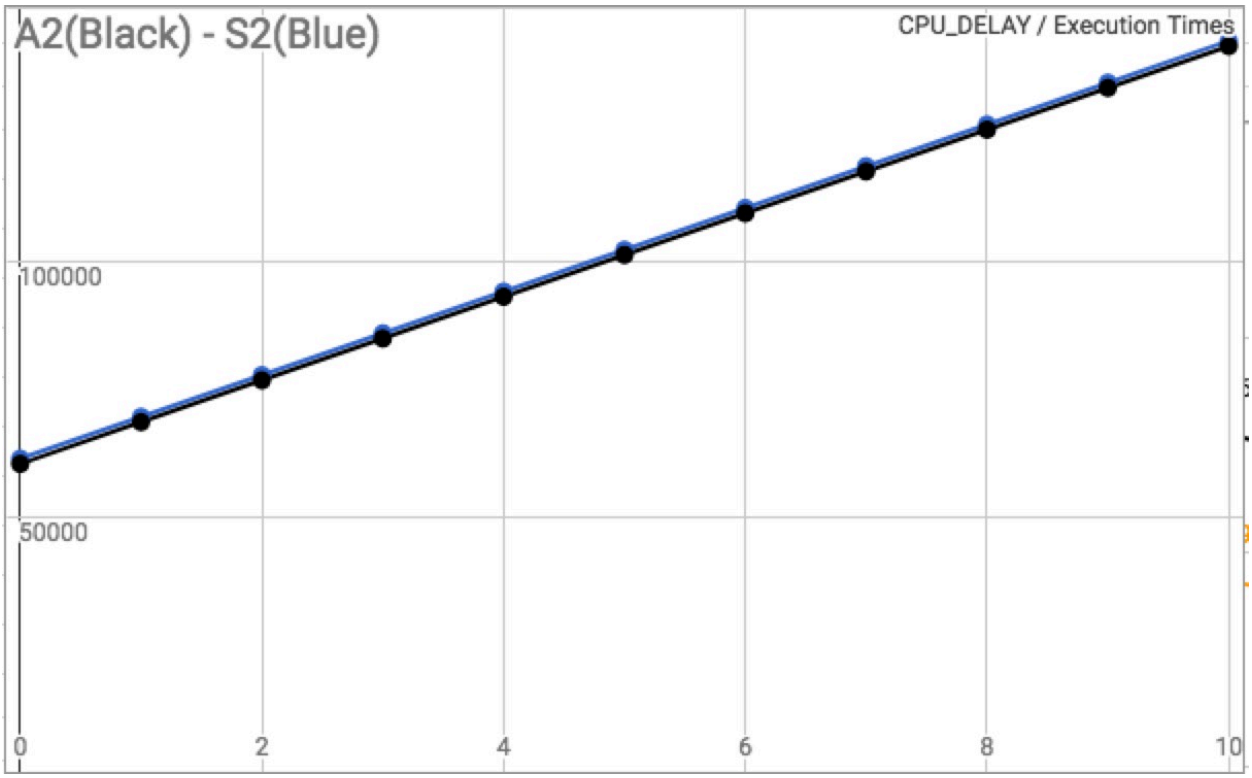
(iii) These two curves would be the same, because no matter how large the CACHESIZE is, write/read operation need to be processed in the read requests.

2.Plot 2



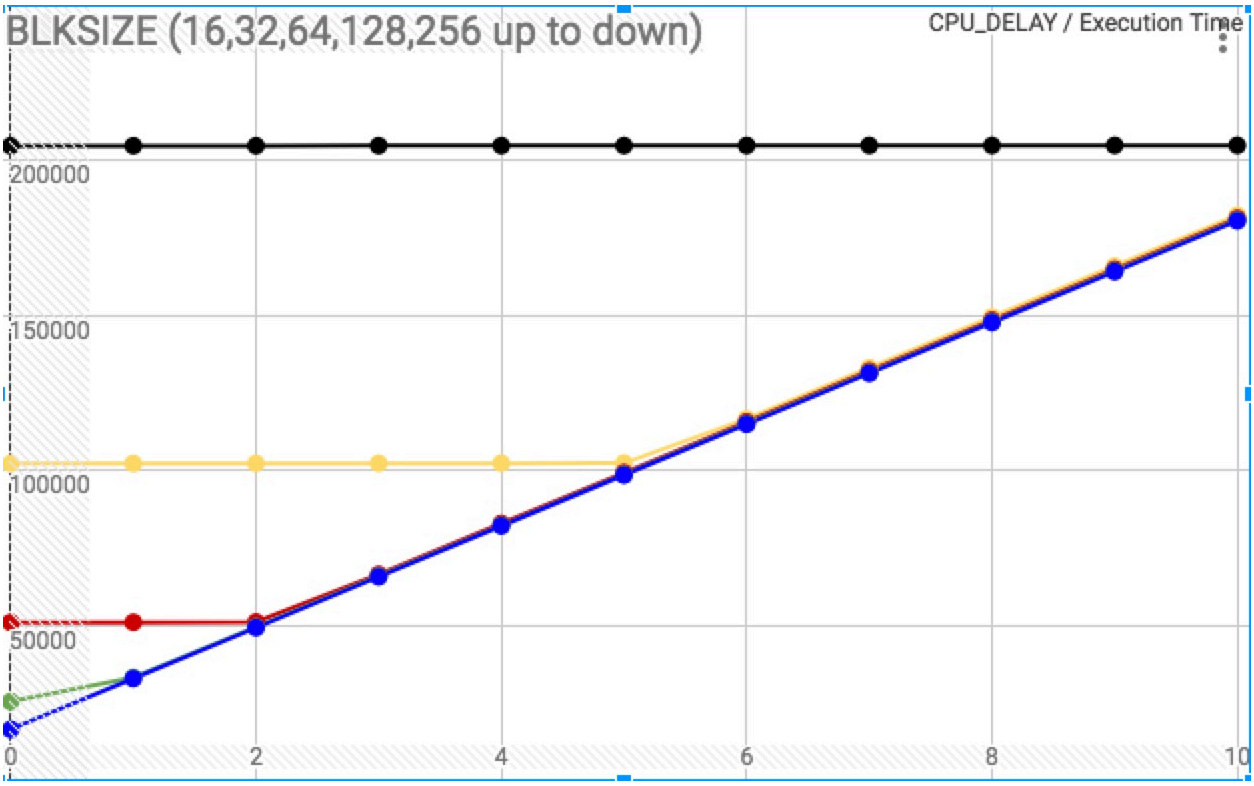
For the write request, the asynchronous process could process write/read operation ‘separately’ (for example, written back of dirty block can be done during write/read progress), so the execution time for synchronous is larger than asynchronous.
For the asynchronous curve, when the CPU_DELAY is less than 5 the execution time remains the same. Because read operation time is 10 more than write, in the synchronous progress, the write/read operation should wait 10 instead of CPU_DELAY (which represents less than 2x5=10). When the CPU_DELAY is larger than 5, the execution time will increase with it.

3.Plot 3



In this situation, the large amount of CACHESIZE would make synchronous progress faster than asynchronous. Because the CACHESIZE is enough to store all the data, in this case the frequent processing of write operation (it won't care whether the CACHESIZE can store all the data) will make the whole progress slow (the synchronous progress will omit the write operation).

4.Plot 4



With the increasement of BLKSIZE, the CACHESIZE (blocks x BLKSIZE) increases, which reduce the execution time. That is because the BLKSIZE is very large, we don't need to read every time before write operation, instead we read once for the first write operation and read the second time when the whole size of this block is used. The read time is saved, so the execution time decreases.

For the 16 BLKSIZE, the size is too small to affect the execution time. For 32,64, the CPU_DELAY is less than the gap between write/read operation, so the execution time remains the same at the beginning (same as p3).