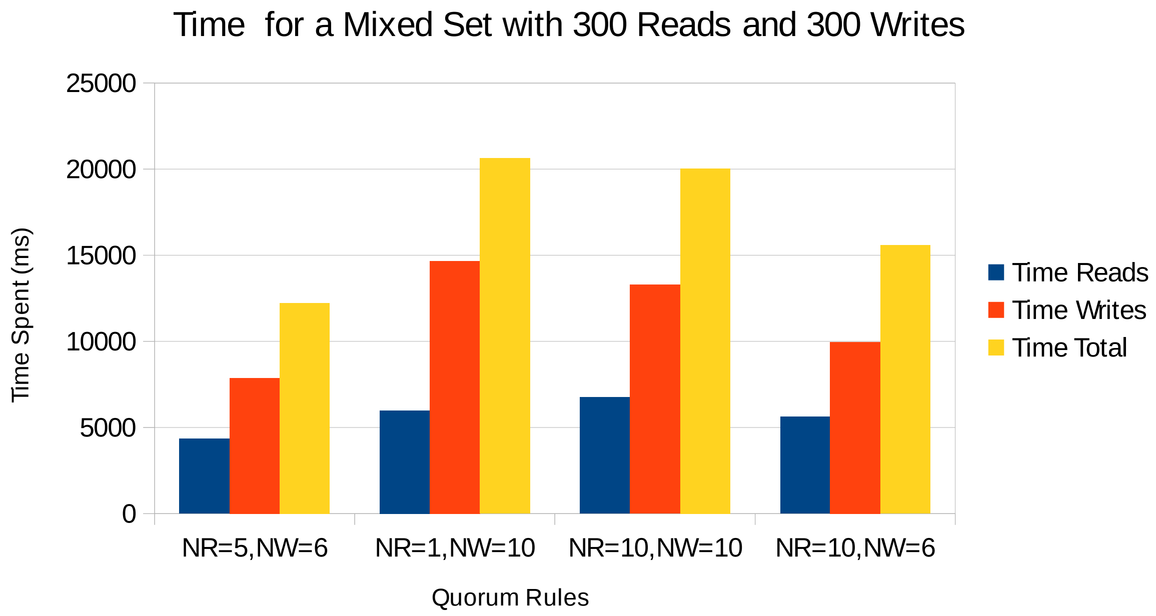
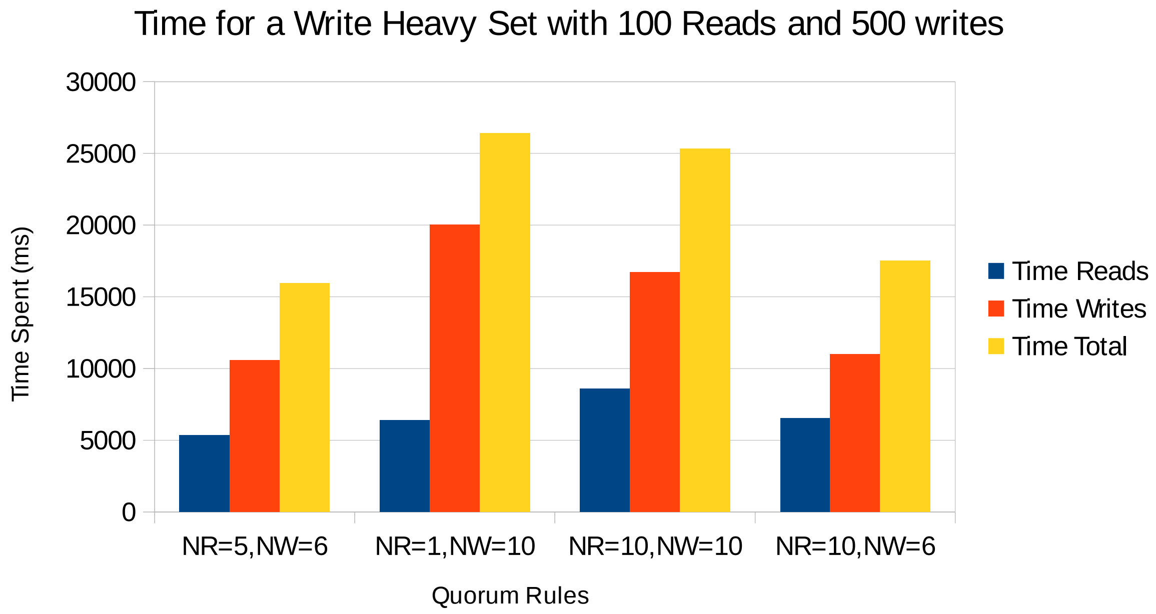
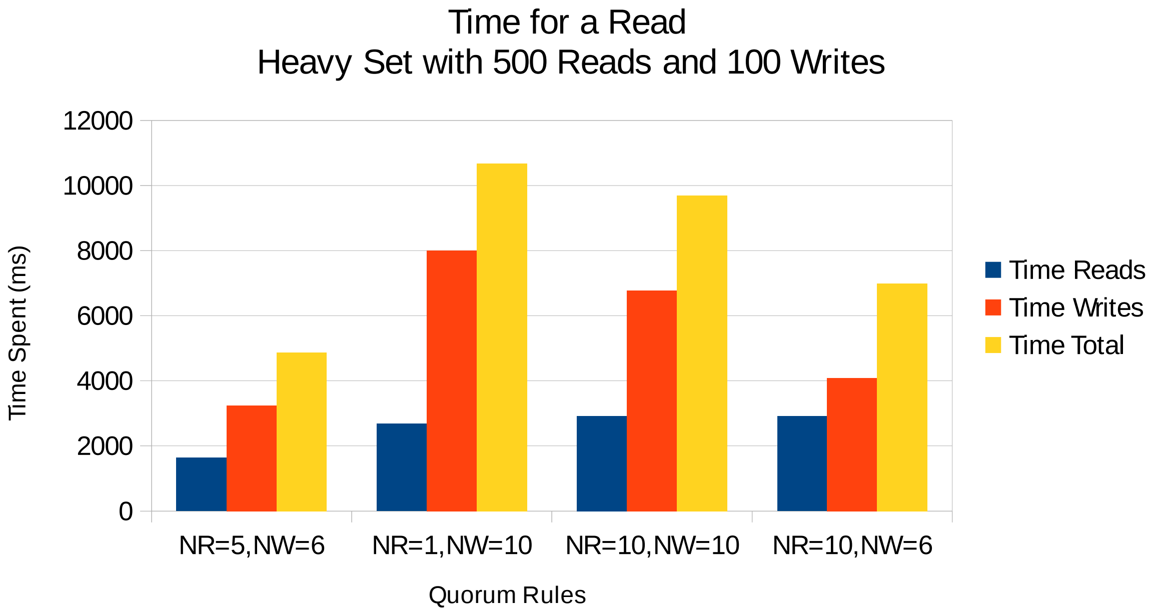
	Time Reads	Time Writes	Time Total
NR=5,NW=6	4349	7875.5	12224.5
NR=1,NW=10	5986	14652	20638
NR=10,NW=1(	6745.5	13282.5	20028
NR=10,NW=6	5612	9966	15578



	Time Reads	Time Writes	Time Total
NR=5,NW=6	5355.5	10588.5	15944
NR=1,NW=10	6396.5	20017.5	26414
NR=10,NW=1(	8606	16732	25338
NR=10,NW=6	6527	10984	17511


Time for a Write Heavy Set with 100 Reads and 500 writes

	Time Reads	Time Writes	Time Total
NR=5,NW=6	1635	3236	4871
NR=1,NW=10	2680	7999	10679
NR=10,NW=1(	2922	6778.5	9700.5
NR=10,NW=6	2910.5	4081.5	6992



In this instance the quorum rule with the least amount of servers will spend the least amount time waiting for their reads and writes to be verified and performed with the quorum protocol.

The reason for this is because to assemble a quorum the coordinator must make connections (across the network) as well as RPC calls to get the latest version of a file (file metadata), and then compute the latest version to read/write to. After which the coordinator then again has to provide all servers within the quorum the latest versions of a new file.

Thus we can infer that the larger the quorum, the longer the time taken to process and complete a request because of the operations needed across the network.

Quorums with more write servers spent more time writing in all data sets given all servers must perform writes to their filesystem. This also caused increased read times as while reads are concurrent, every write operation blocks the entire request queue from being processed as we cannot write to servers safely if there are reads going on. Thus, as **reads are subsequently blocked by writes**, the average time for read requests are also increased.

Datasets with more reads than writes spend less time overall, that is read heavy trials show that concurrent reads allow for a much shorter data access time, as each server request can be serviced by the coordinator threads concurrently in a multi threaded environment.