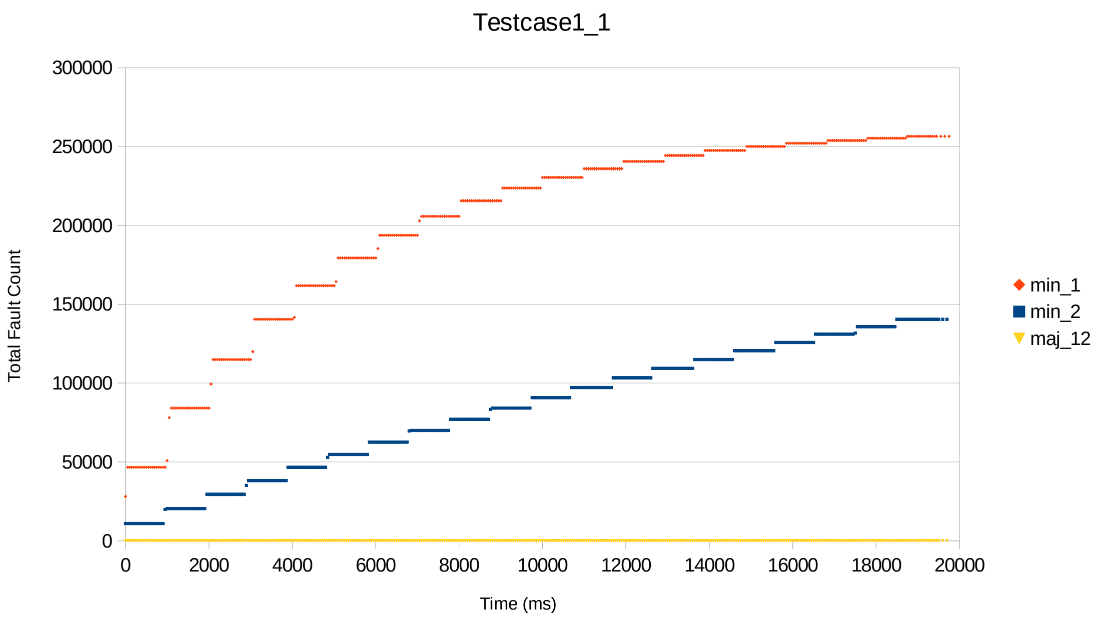
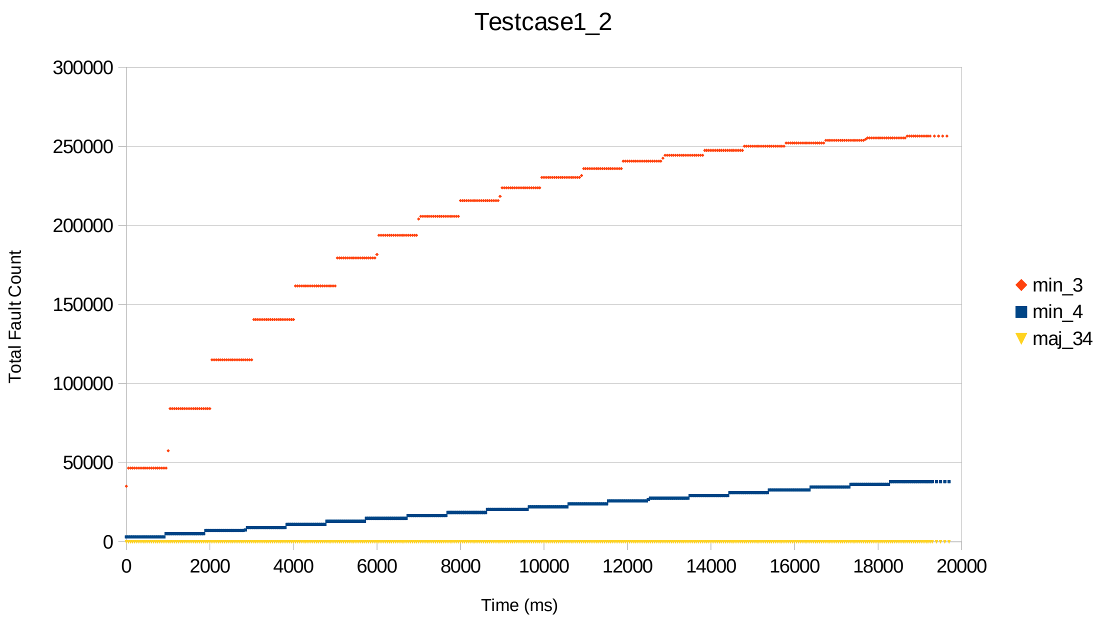
Case Study 1: Thrasing and Locality

Plotings:





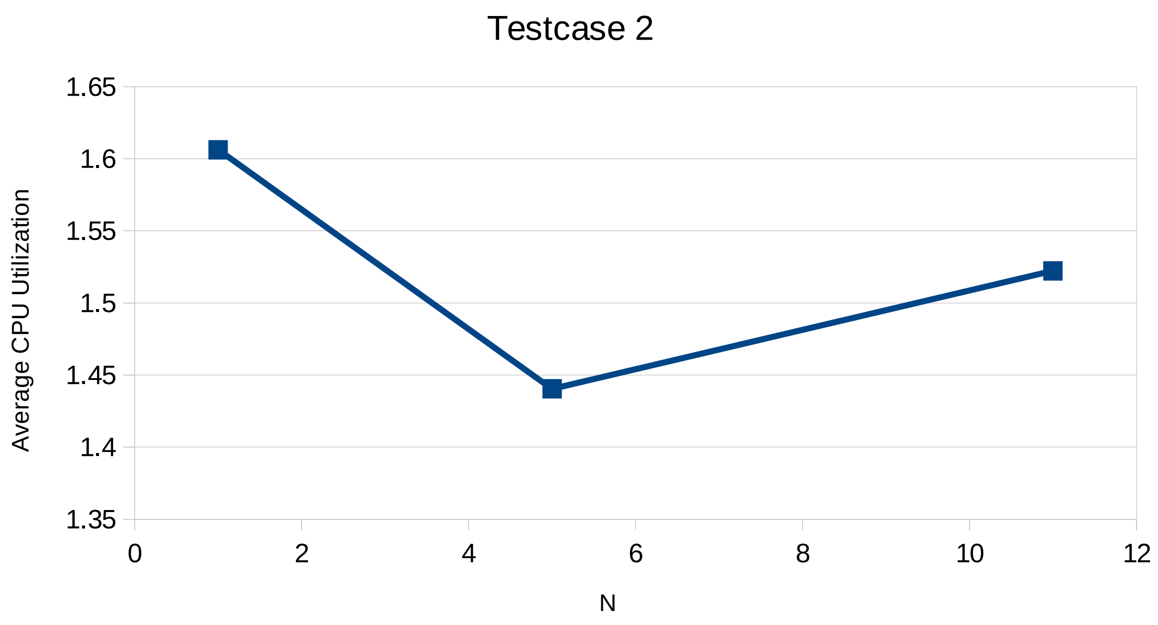
*Note: The accumulated minor fault count of Work Process 1 is represented as “min_1”, Work Process 2 as “min_2”, and so on.

Analysis:

As the graphs indicate, we can notice the obvious decrease of total minor fault count by comparing Work Process 2 and 4. It results from the different memory access modes: Work Process 2 access the memory randomly, while 4 access that locally. The local access pattern help the operating system to reuse virtual memory address which is probably accessed recently. However, the random access pattern does not follow the rule of recently-used-first, making the system to meet more page faults. The minor fault count of Work Process 1 and 3 are the same and they are both higher than 2 and 4 because they are executing the same amount of jobs both with random access pattern.

Case Study 2. Multiprogramming

N	1	5	11
Total Time (ms)	19300	19300	19350
CPU Time (ms)	31000	139000	324000
Total CPU Utilization	1.6062176166	7.2020725389	16.744186047
Average CPU Utilization	1.6062176166	1.4404145078	1.5221987315



Analysis:
As shown in the graph, when N increases, the average CPU utilization does not increase. Instead, CPU utilization tends to decrease. This is called thrashing, which indicates when the virtual memory is over accessed, the CPU utilization will decrease, and the performance of the operating system will be affected.