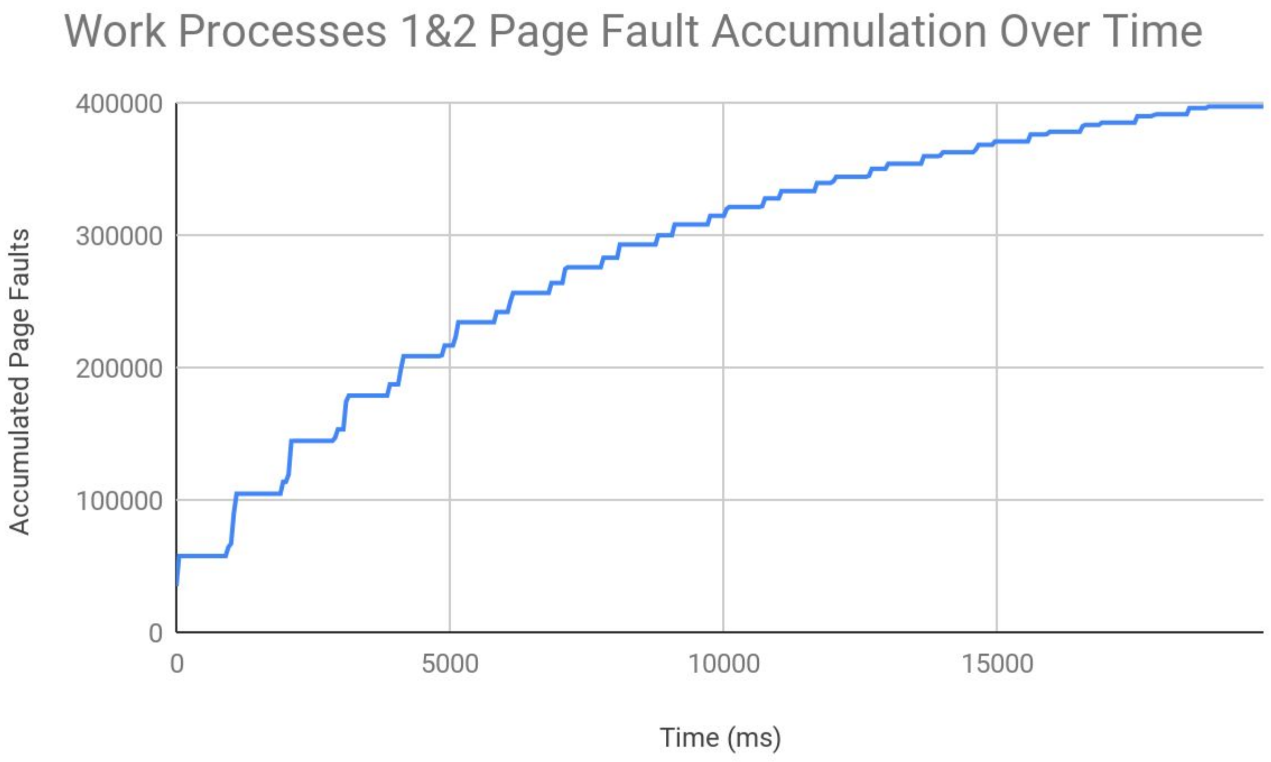
Case Study 1: Thrashing and Locality

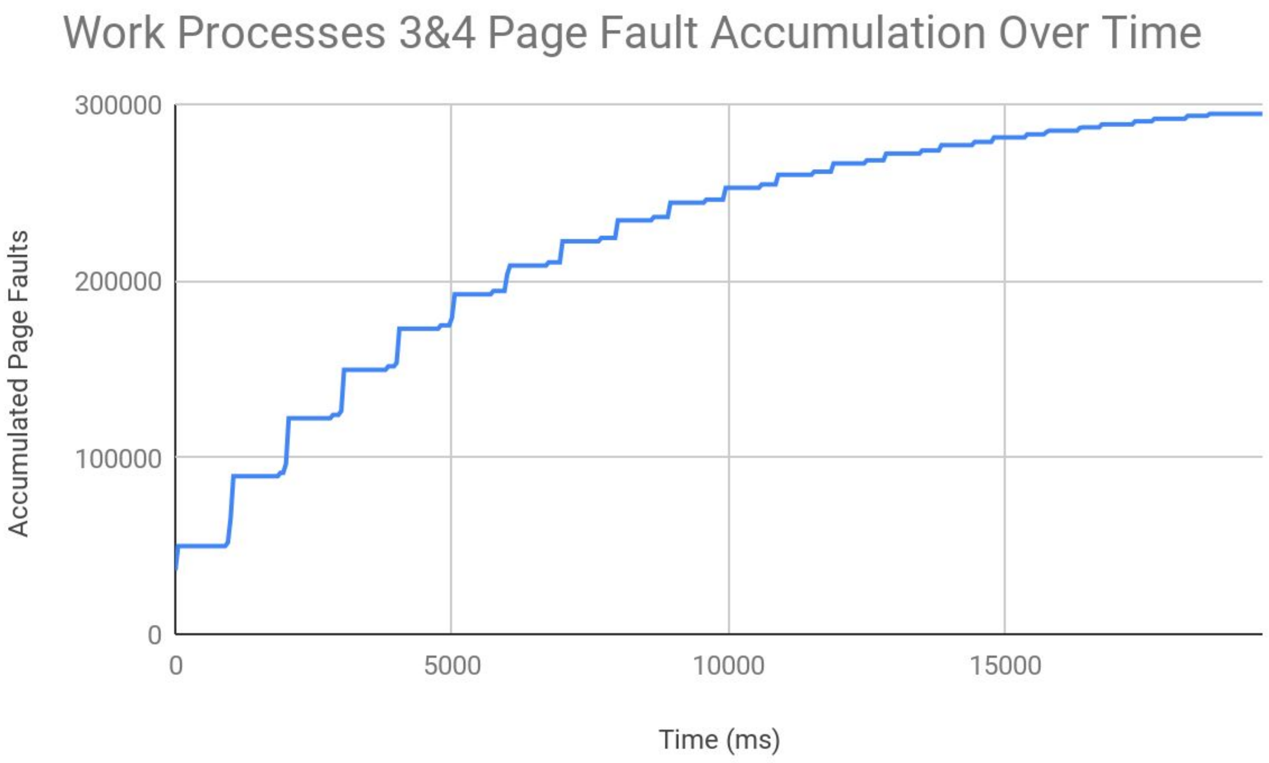


In the first scenario, two test programs were run that both performed random accesses, with a different number of accesses per iteration. The accumulated page faults of both programs over time is displayed here.

Each second, work process 1 makes 50,000 memory requests and work process 2 makes 10,000 memory requests. Because both processes were started at different times, we can observe a big rise and a smaller rise with a frequency of about second in the number of accumulated page faults. The large rise is due to work processes 1, which makes 5x more requests than work process 2.

As program execution continues, we can observe the number of page faults incurred by both programs decreases over time. This is because the TLB caches more and more of the requested pages, reducing the number of page faults in the initial iterations.

However, around 10,000ms, the page faults appear to begin increasing at a linear rate. The TLB isn't big enough to hold all requested pages, so around 10,000ms, all subsequent page faults are compulsory and the number of page faults incurred per iteration is constant. After 20 iterations, around 400,000 page faults were incurred.



In the second scenario, two test programs were run. One issued 50,000 random memory request each second similar to the first scenario, but unlike the first scenario, the second test program performed 10,000 memory requests with locality.

Similar to scenario 1, both processes make memory requests each second, and we can see rises in the total accumulated page faults with a frequency of 1 second. However, because the second test program makes accesses with locality, it causes very few page faults, and the smaller rise in page faults in the graph above is much smaller than the previous scenario.

Similar to scenario 1, the page fault begins to rise at a linear rate as time progresses. After 20 iterations, the number of page faults incurred is around 300,000, which is 100,000 less than scenario 1. The second test program makes a total of 10,000 * 20 = 200,000. It is interesting to note that by converting from random to local accesses, it caused half as many page faults.