creases more rapidly as the simulation progresses, while the distributed synchronization waiting time is linear.

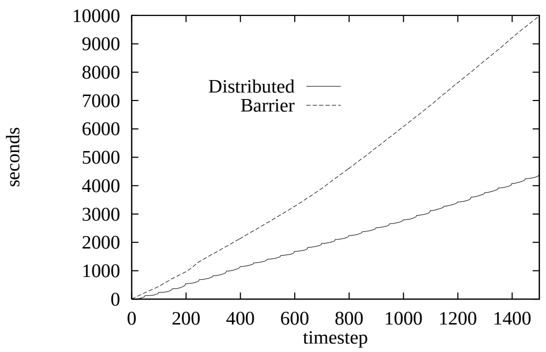

Figure 2: Cumulative synchronization time for all processors.

In Figure 3 we present measurements that show how the benefit from distributed synchronization increases as the amount of parallelism is varied from 1 processor to 8 processors. Observe that the cost of distributed synchronization is growing more slowly than the cost of barrier synchronization, at least on this small-scale multiprocessor. Measurements show that lower sensitivity to the fluctuating load balance is responsible for most of the improvement. In the case of barrier synchronization, all the processors have to wait for the slowest processor on each timestep. In the distributed scheme, processors only wait for neighbors, and therefore short-term load fluctuations are much less of a problem, as illustrated in Figure 4. This is a simplified one-dimensional model, so that each precinct has at most 2 neighbors (precincts 0 and 6 have only one neighbor each because they are at the ends). Assume that all precincts have finished processing timestep t, that precincts 1 through 5 have finished processing timestep $t+1$, but that precincts 0 and 6 are still busy processing timestep $t+1$. With barrier synchronization we have to wait for all the precincts to finish processing timestep $t+1$ before starting work on any of the precincts for timestep $t+2$. The distributed scheme, on the other hand, can continue scheduling precincts until timestep $t+3$: precincts 2, 3, and 4 can do work for timestep $t+2$, and precinct 3 for timestep $t+3$. The work on timestep $t+3$ for precinct 3 can only begin after precincts 2, 3, and 4 have finished working on timestep $t+2$. A short-term load fluctuation that holds up precincts 0 and 6 does not immediately restrict us, and we also have time to recognize the imbalance and correct it before it impacts performance.

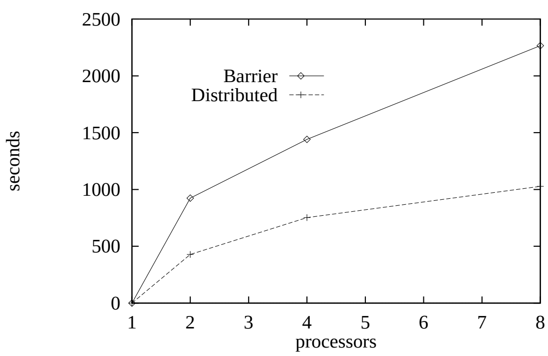

Figure 3: Total synchronization time vs increased parallelism.

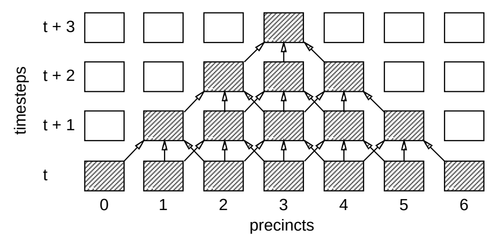

Figure 4: Tolerating load fluctuations through distributed synchronization.

There is additional overhead associated with distributed synchronization because we have to read the timestep counters of all neighboring precincts before proceeding. This overhead cancels out much of the benefit of distributed synchronization on small-scale parallel machines, such as the one we performed our measurements on. The total execution times of the two schemes are thus within 2% of each other.

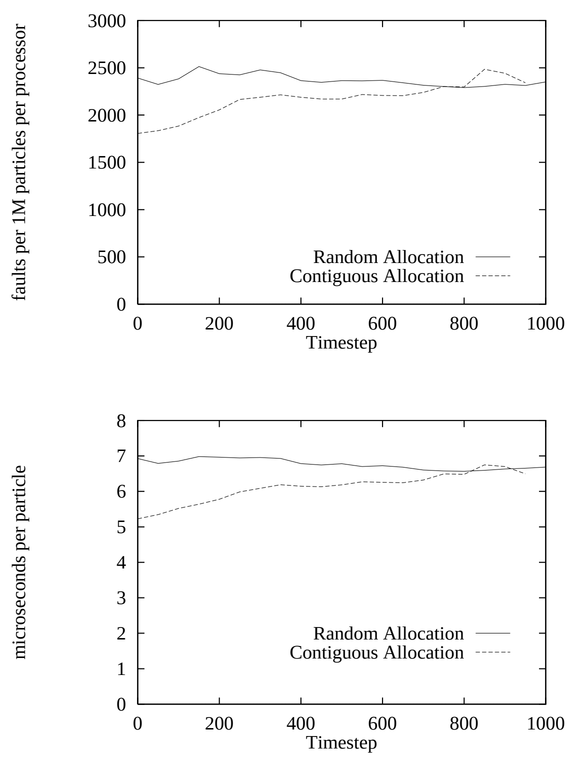

Figure 5: (a) L2 TLB misses per processor per timestep (b) Execution time per timestep, showing effect of increasing TLB misses.

An unexpected complication that our detailed measurements revealed is the substantial performance impact of