CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

It is argued that the multicore processors pose unique scheduling problems that require a multiagent based software approach that utilizes the large number processors very effectively. Actually, the work of dispatcher was eliminated with the help of processor agents itself. Each processor scheduling will be similar to the self scheduling employed in the traditional multiprocessor system. This is possible only with the help of processor agents assigned for every processor. In the affinity based scheduling, the CPU utilization is actually maximum for the critical tasks and ideal processors are utilized well in the case of non critical tasks. Even though there is a cost of migrating the non critical tasks to some other processor efficient and maximum utilization of the CPU is the primary concern.

Actually, multicore processors pose unique free space management problems that require an agent based software approach that utilizes the large number processors very effectively. It is also proved that lot of drastic enhancements in the traditional unused space collector part of operating system that optimizes for CPU cycle utilization. It is discovered that the CPU performance increases slowly with the increase of free space. As a conclusion the new novel approach eliminates the complexity of collecting the unused space files in the many core systems and improved the CPU utilization to the maximum level and ultimately the load balancing is done.
The load balancing algorithm must be designed in such a way that it has to reduce this waiting time. The proposed results shows that the load balancing delay is 0.01% lesser compared to other algorithms. The agent based load balancing produced less response time compared to other algorithms. Overall an improvement of 12% has been shown by the simulation results. Context switching will not be taking place under affinity based scheduling combined with BST load balancing agent. So load balancing is achieved without any process migration. The overhead complexity is negligible (0.0001%) in this scenario. But with the normal agent based scheduling it cannot distinguish between critical tasks and noncritical tasks. So the BST load balancing agent encounters context switch overhead. Thus the BST load balancing agent performance degraded to 0.43% under normal workload because of process migration delays.

Performance analysis of several scheduling algorithms and the proposed agent based scheduling algorithm is evaluated where in the number of processes are varied from 50 to 500 and the observation is made for multicore cpu utilization. As and when there is an increase in the number of processes, the utilization of cpu is increased from 10% to 98%, keeping the number of cores as const=50. Workload performance for Various Cores with respect to Average energy per task is explained in chapter5. Different make spans and their performance in terms of speed are compared. The schedule length is very large, for example 50% higher over other scheduling algorithms and it is shown that none of the processor will be kept in the idle state. Initially, the simulation set up for minimum number of processes (50) and increased upto maximum number of processes(500), then the comparative analysis with respect to the cpu utilization and evaluation accuracy for affinity vs round robin, affinity vs SJF, affinity vs EDF and the summary for all the scheduling algorithm is presented.
Although the results from the Linux kernel version 2.6.11 analysis are encouraging, there are many open questions. Even though the improvement (average waiting time reduction) possible with number of cores, for some workloads there is a limitation by the following properties of the hardware: the high off-chip memory bandwidth, the high cost to migrate a process, the small aggregate size of on-chip memory, and the limited ability of the software (agents) to control hardware caches. It is expected that the future multicore will be used to adjust some of these properties in favor of the multiagents based scheduling. More intelligent algorithm can be proposed in view of the memory limitation with respect to operating system. As the power consumption model was presented in abstract model, indepth analysis is required for the complete system.