Our previous discussions about CPU scheduling have been limited by the assumption that hardware only has a single CPU. However, multi-core processors have become commonplace as a means of increasing performance. Multi-core processors have become mainstream in current processor design (Dhivya & SenthilKumar, March 2015; Mundada, & Chandak, March 2013) and even mobile devices are increasingly being powered by multicore processors (Fernandes & D’Souza, December 2015). Therefore, our discussion of scheduling needs to now extend also to multiprocessor systems. Generally, there are two approaches to scheduling jobs when a system has multiple CPUs: Single-Queue Multiprocessor Scheduling and Multi-Queue Multiprocessor Scheduling (Arpaci-Dusseau & Arpaci-Dusseau, 2012).

**Single-Queue Multiprocessor Scheduling (SQMS)**

In the Single-Queue Multiprocessor Scheduling approach, all processes that are to be scheduled are placed into a single queue, which is the same approach for scheduling that would be used if the hardware had only one CPU (Arpaci-Dusseau & Arpaci-Dusseau, 2012). The CPUs are then assigned processes in order of priority from that single queue. This is simple to implement, but it does have its drawbacks. First, SQMS has issues related to cache affinity. That is, when jobs are scheduled across different CPUs, there is a drop off in performance because of the need to load and reload the process states as they bounce among the various CPUs. This overhead can be mitigated by keeping processes running on the same CPU as often as possible. Second, SQMS does not scale well because of overhead related to synchronization. That is, as a program runs on one CPU and then another, there needs to be mechanisms which ensure that all data used across CPUs are up-to-date. This is often achieved by including lock and unlock code to routines. However, performance will decline as the number of CPUs increases and more time is spent in lock and not actually running processes.

**Multi-Queue Multiprocessor Scheduling (MQMS)**

In the Multi-Queue Multiprocessor Scheduling approach, there is a different scheduling queue for each CPU, with each queue following its own scheduling policy. “When a job enters the system, it is placed on exactly one scheduling queue, [...] Then it is scheduled essentially independently, thus avoiding the problems of information sharing and synchronization found in the single-queue approach” (Arpaci-Dusseau & Arpaci-Dusseau, 2012). Because the number of queues increases with the number of CPUs, MQMS is inherently scalable. Moreover, this mitigates the drawbacks of locks and cache affinity, with processes staying on the same CPU. There is, however, a downside to the MQMS approach, which is load imbalance. That is, processes on one CPU may end, leaving the CPU idle; whereas another CPU may be bogged down with many processes. This is not only and issue of efficiency, but it can also have down level ramifications like increasing the need for heat-sinks, air conditioning, fans, and even degradation of the overused CPU (Dhivya & SenthilKumar, March 2015). While migrating processes from one CPU to another can alleviate load imbalance, doing so can be complicated and even hinder performance as the number of CPUs rises.

**Summary**

The SQMS is simple and balances load across CPUs; but it has difficulty with cache affinity and scaling. The MQMS scales well and deals with cache affinity too; but it is complex and has issues with load imbalance, which can lead to hardware failure.

References

Arpaci-Dusseau, R. & Arpaci-Dusseau, A. (2012). Operating Systems: Three Easy Pieces. Madison, WI: University of Wisconsin-Madison. Retrieved from <http://pages.cs.wisc.edu/~remzi/OSTEP/>/

Dhivya, G. & SenthilKumar, A. (March 2015). “CPU Scheduling for Power/Energy Management Using RMS Algorithm”.International Journal of Advanced Research in Electronics and Communication Engineering, 4(3), 548-550. Retrieved from <https://www.techrepublic.com/resource-library/whitepapers/cpu-scheduling-for-power-energy-management-using-rms-algorithm/post/?regId=&asset=33090637&skipAutoLoad=1>

Fernandes, R. & D’Souza, R. (December 2015). “Survey on Web Service Discovery and Availability in Mobile Environments and Future Trends in Mobile Web Services”. International Journal of Computer Applications, 132(12), 16-19. Retrieved from <https://www.techrepublic.com/resource-library/whitepapers/survey-on-web-service-discovery-and-availability-in-mobile-environments-and-future-trends-in-mobile-web-services/post/?regId=&asset=33148293&skipAutoLoad=1>

Mundada, S. G. & Chandak, M.B. (March 2013). “An Approach for Multi-Core Real Time Parallel Processing”. International Journal of Computer Applications, 65(5), 39-43. Retrieved from <https://www.techrepublic.com/resource-library/whitepapers/an-approach-for-multi-core-real-time-parallel-processing/post/?regId=&asset=32881107&skipAutoLoad=1>