Dennis McWherter
7 October 2011
CS 241
dmcwhe2

In MP5 I had the task of implementing a simple multithreaded make utility. The syntax was similar to that of a GNU Makefile. In my MP I had to share three resources across my threads which needed to be protected: two queues and a counting variable. I had one queue for commands which needed to be processed, and another queue for commands which had already been processed. Before accessing either, I had to use a mutex to secure them. Likewise, I had a count for how many threads were sleeping/completed and in order to increment this variable, I had to also protect it with a mutex.

To guarantee threads were kept as busy as possible, I made certain to continually check the unprocessed queue for new jobs. If nothing was pulled from the queue, I determined whether or not a sufficient amount of jobs had been processed in order for the thread to sleep; if not, I restarted and tried to grab an element off the queue. This works properly since I simply put a command at the back of the queue if all of its dependencies had not yet been met.

Amdahl's law applies to this MP since there can be only a certain amount of parallelism in processing jobs limited by dependencies. For instance, if every other process depends on one process which takes the majority of the running time, the program is nearly sequential since everything blocks until that one dependency has finished all of its work. Therefore, Amdahl's law tells us that concurrency is limited by certain factors for which other parts of the program rely. In this case specifically, however, it happens to be our dependencies.