Active Monitor

Javier Cortes, Kyle Polansky

Introduction

- Active Monitor is an improvement over Java Monitor in multi threaded applications.
- Active Monitor runs on a background thread, and is an active system artifact.
- When tasks are submitted to Active Monitor, a Future is returned allowing async execution
- Worker threads are spawned for each task and combined before returning Future

Task Reordering

Tasks can be reordered following these rules:

(1) only one worker thread can execute each

(2) all tasks from a process must be submitted to worker threads in the order that they were submitted (serialization)

(3) Only one worker may be executing a task per unique lock object at a time (mutual exclusion)

Reordering will run the same critical sections on the same worker threads to take advantage of hardware caching.

Performance Increases

Biggest performance increases in multithreaded code with lots of critical sections.

We see bigger performance increases with more worker threads

In code with few critical sections, extra threads just creates overhead

Demo

Conclusion

In CS heavy applications, Active Monitor can potentially improve performance.

In programs with few threads or few CS, Active Monitor may create extra overhead.

Questions?