Exploring Coordination of Threads in Multi-Core Libraries

Michajlo Matijkiw
Faculty Adviser: Milo M. K. Martin

Background and Motivation

- Multiple cores on one chip
- High performance, energy efficient computing
- The future of computing
- Multi-core libraries make multi-core programming easy
- More software threads than cores means poor performance

8 Core Performance

- Even high performance mutexes suffer!
- Limit threads at runtime to increase performance!

8 Core Reaction time

- 8 threads are added to the system and then exit
- Use multiplicative decrease and additive increase for stable thread adjustment

How It Works...

- Worker threads run tasks as they become available
- A monitor thread watches system performance

system performance

Parallel work is divided into tasks and distributed to threads

Tasks are processed one at a time by each thread

Over-subscription is detected and a thread is blocked

System overloaded! Stop worker thread!

Extra tasks are picked up by the remaining threads

Finally all tasks are complete and the workers wait again

The main thread proceeds with serial execution

- Threads are unblocked when the system is no longer over-subscribed
- Thread count is never less than one or greater than the number of cores
- System load is based on the number of non-blocked processes
- Worker threads only contribute to system load when executing tasks

Design

- Modify Intel Threading Building Blocks: popular multi-core library
- Add support for dynamic thread adjustment
- Minimal overhead necessary
- Monitor thread tracks system utilization and communicates with scheduler

- Use processes in operating system run queue as a base
- Waiting and blocked worker threads not counted towards load