

Performance Debugging Shared Memory Multiprocessor Programs with MTOOL

A. Goldberg, J. Hennessy

Presented by Sam Angiuoli

What is MTOOL?

- Performance profiler
 - Shared memory bottlenecks, synchronization overhead, parallelization overhead
 - At least 2 profiled executions required
- Supported platforms
 - MIPS based architectures (+ others?)
 - SGI 380 (8x33 MHz processors and 256M shared mem)
 - C + ANL macros
 - Fortran with loop level parallelism

Overview of paper

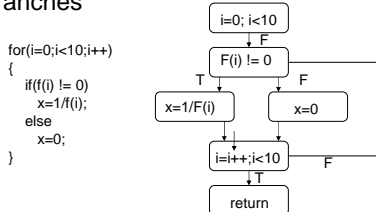
- Instrumentation
 - Timers
 - Basic block counters
- Efforts to minimize instrumentation overhead
- Description of memory/synchronization bottlenecks
- 2 case studies

Timers

- start_timer/stop_timer added to begin/end of procedures
- Bloat is minimized by scanning initial execution profile to exclude fast/frequently executed regions
 - Minimum of 5x the overhead of start/stop timer
- Alternative to timers is pc-sampling

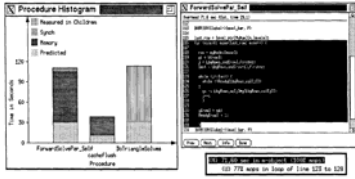
Basic block

- A sequence of one or more consecutive, executable statements containing no branches



Minimum Cost Basic Block Counting

- Minimize overhead while collecting block counts during program execution
- Only place counters on independent control paths
 - Derive dependent counts during post processing
 - Eg: Don't count both blocks of if/then/else
- Use loop counters to avoid counting each iteration



- MTOOL displays code block responsible for the bottleneck
- UI allows for reclassification of user spin-wait as synchronization overhead
- Code indicates that numerous global memory references may be saturating the shared bus and causing the bottleneck

Summary

- MTOOL profiling can identify memory and synchronization bottlenecks on a shared memory architecture with as few as 2 program executions
- MTOOL timer and basic block count instrumentations minimize overhead and program perturbation