ECE 471 – Embedded Systems
Lecture 25

Vince Weaver

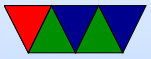
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

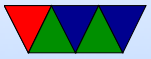
5 December 2013

Announcements

- Project Update
- HW#5 Update – due at beginning of class on Tuesday.

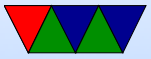


Optimizations – What does perf tell us?



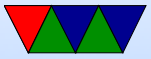
Low IPC

- IPC = Instructions Per Cycle
- On simple processors you max out at 1.0, on modern super-scalar can be higher
- Related metric: CPI (cycles per instruction) which is the inverse; you want this to be lower
- On in-order processors, the way to fix this is often code scheduling. Putting together instruction streams so that



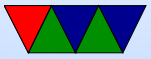
dependencies are avoided. Compilers can in theory do this for you.

- Out-of-order processors in theory do this for you
- Other things, such as cache misses, can also cause poor IPC. Look at cache metrics and stall metrics for more info.



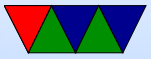
High Cache and TLB Miss Rate

- Cache misses slow down your program by stalling your processor waiting for memory.
- TLB misses are similar. The TLB caches the virtual-memory physical to virtual translations. TLBs are small, and when a TLB miss happens the operating system has to walk the page table which is slow.
- Causes of cache misses:
  - Cold misses – first access to a value and not in cache.



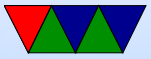
prefetching (both hardware and software) can help this case

- Capacity misses – Cache just isn't big enough to hold all of the values you are trying to access.
- Conflict misses – two values you are trying to access have addresses that put them into the same cache line. Set associative caches can help, but even so this can still happen. One way to avoid this is add padding to change the address of the two objects, though this can just push the problem somewhere else rather than really fixing it.

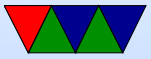


- Coherency misses – on multi-processor systems "cache coherency protocols" handle values shared across CPUs. If both CPUs are trying to write the same value, performance will suffer (also same writes to same cache line even if not the same variable– this is known as "false sharing")

- Various other things you can do to improve cache performance:
  - Use smaller data structures
  - Use cache blocking (break big data structures into

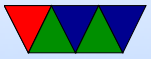


smaller pieces with better locality)



High Stall Rate

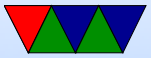
- A processors stalls when it cannot continue execution because it is waiting on some sort of resource
- Cache stalls: waiting on icache (for next instruction) or dcache (memory value)
- Cannot issue instruction because functional units not ready (not enough adders, floating point, etc)
- Various units can be full



High Branch Predictor Miss Rate

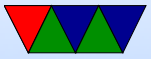
- If branch is predicted wrong, processor has to stop, kick out all wrong-executed instructions, and re-start

- What you can do:
  provide hints for direction
  re-arrange code to avoid aliasing in tables
  use Conditional moves/execution
  Loop unrolling (adds code size/complexity)

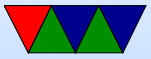


Other Optimizations

- Code Hoisting
- Dead code elimination
- Function Inlining
- Common Subexpression Elimination
- Use lower precision or unsafe floating point
- Change Algorithms! Use optimized libraries!



- Multi-thread your code



Validating Perf Counters

- Hard to do. Compilers and prefetchers too smart.
- Write code to get 100% cache miss rate? Prefetchers are smart
- Write poor matrix-matrix code as an example? Compiler fixes it