Lecture 6: Measuring Performance

William Gropp
www.cs.illinois.edu/~wgropp

How Should You Measure Performance?

- What is wrong with this code to time a loop:

```
Call cpu_time(tstart)
do i=1,n
    x(i) = a*y(i)
enddo
call cpu_time(tend)
tloop = tend – tstart
print *, tloop
```

Problem: Clock Ticks

- Timers are not infinitely accurate
  - ♦ All clocks have a granularity – the minimum time that they can measure
  - ♦ The error in a time measurement, even if everything is perfect, may be the size of this granularity (sometimes called a clock tick)
- Always know what your clock granularity is
- Ensure that your measurement is for a long enough duration (say 100 x the "tick")

Problem: Cold Start

- What happens when the code is executed? The assumption is that the code is ready to execute. But
  - Code may still be on disk, and not even read into memory.
  - Data may be in slow memory rather than fast (which may be wrong or right for what you are measuring)
- Multiple tests often necessary to ensure that cold start effects are not present
- Special effort often required to ensure data in the intended part of the memory hierarchy.

Problem: Smart Compiler

- If the result of the computation is not used, the compiler may eliminate the code
  - Performance will look impossibly high
  - Even worse, eliminate some of the code so the performance looks plausible
- Ensure that the results are (or may be) used.

Problem: Interference

- Other activities are sharing your processor
  - Operating system, system demons, other users
  - Some parts of the hardware do not always perform with *exactly* the same performance
- Make multiple tests and report
- Easy choices include
  - Average tests – represents what users might observe over time
  - Minimum value – Because must interference *slows* system, may be the most reproducible
    - Note that if multiple iterations are used to avoid clock tick problems, the best you can do is the minimum of an average
  - Box plot – show all data, giving mean, median, and first and third quartile
- Harder is to ensure reported result is statistically relevant

Problem: Reporting

- What is wrong with with reporting this time: 2.34784e-6?

Problem: Reporting

- What is wrong with with reporting this time: 2.34784e-6?

1. What are the units (seconds, hours)?
2. How accurate was your measurement:
   1. In absolute terms, this claims to 10^{-11} seconds (assuming units are seconds)
   2. In relative terms, this claims to one part in 10^5

- You can use simple formats to print data from your program, but don't simply copy every digit into your report/paper/ presentation

There's More

- Accurate, reproducible performance measurement is *hard*
- Think carefully about your experiment:
  - What is it, precisely, that you want to measure
  - How representative is your test to the situation that you are trying to measure?

Question For Review

- Fix the example on slide 2 to
  - ♦ Avoid cold start issues by running the loop once before timing
  - ♦ Avoid clock granularity by timing multiple iterations of the same loop, then dividing by the number of outer iterations
  - ♦ Avoid a smart compiler by computing something with the result
  - ♦ Avoid Interference by running the tests multiple time and report the minimum and average times.