Performance Measurement
Lab Schedule

Activities

- Today / Thursday
  - Background discussion
  - Lab 5 – Performance Measurement

- Next Week
  - Lab 6 – Performance Optimization

Assignments Due

- Lab 4
  - Due by Feb 20\textsuperscript{th} 5:00am

- Lab 5
  - Due by Feb 27\textsuperscript{th} 5:00am
Person of the Day: Bill Joy

- Co-founder of Sun Microsystems
- Author of vi text editor
- Key contributor to original BSD Unix while a Berkeley grad student
- First open-source OS with TCP/IP
Lab 5 Goals

1. Measure program execution time

2. Break down program execution time by specific subroutines / lines of code

3. Monitor program for memory leaks
   ➣ Not really “performance”, but uses same tool
Why is it important to measure application performance in detail?
Valgrind

http://valgrind.org/
Valgrind Features

- Memcheck module – Memory error detector
- Access 1 beyond the beginning / end of your array?
- Access un-initialized memory?
- Reading/writing memory after calling free()?
- Memory leak? (Lost pointer to a malloc() block)

Valgrind produces a report that identifies these errors
Valgrind Features

- **Callgrind** module – Program profiler
- Callgraph shows you what function called what other functions
- How much CPU time does each function / code line consume?
- Valgrind produces a report that summarizes CPU usage of your program
Valgrind Features

- **Massif** module – Heap profiler
  - Optimize your program to use less memory (by identifying where memory is being used)

- **Helgrind** module – Threading profiler
  - Bugs in multi-threaded programs are especially difficult to find!

- ... and more modules ...
Valgrind Common Uses

- **Your program runs and suddenly segfaults**
  - Recall a segfault means a memory address was accessed that doesn’t exist for your program

- **How do I find where this error is?**
  - Valgrind can monitor your program and detect accesses outside of static variables and dynamic memory regions
Valgrind Common Uses

- Your program gets slower and slower the longer it runs
  - Memory leak? (Slowing running out of heap memory because you `malloc()` without ever calling `free()`)

- How do I find where this error is?
  - Valgrind can monitor your program. It can’t tell you where you should free it, but it will tell you where you originally called `malloc()`, or where the pointer was lost
Valgrind Behind-the-Scenes

- **Just-in time compiler**
  - Your program is re-compiled onto a virtual (simulated) processor
  - Another example of a virtual machine!

- Benefit – Valgrind can observe your program running at the machine instruction level

- Drawback – Slow! (5x slower than normal)
  - *But it’s still better than fixing bugs without a tool...*
The next labs (5-7) ask you to measure application performance by conducting experiments:

- Execution time
- Processor usage
- Memory usage

Which of these system configuration do you think would be best in terms of producing the cleanest, most accurate, most reproducible results?
Program to Benchmark
Program to Benchmark
Program to Benchmark
The best approach (directly booting Linux) may not be convenient to achieve for this class.

But you can *definitely* avoid the worst configuration!

Keep your system simple when benchmarking.