

Computer Systems and Networks

ECPE 170 – Jeff Shafer – University of the Pacific

Performance Measurement

Lab Schedule

Activities

- Today
 - Discussion on Performance Measurement (Lab 5)
 - **7** Finish Lab 4!
- Next Week
 - Zab 5 Performance Measurement

Assignments Due

- **7** Lab 4
 - Due by Feb 16th 5:00am
- **7** Lab 5
 - Due by Feb 23rd 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

Memory Leaks



Problem 1

```
int main() {
  int *array;
  array=(int *)malloc(sizeof(int)*1000);

for(i=0;i<1000;i++)
    array[i] = i;

return 0;
}</pre>
```

Where is the Memory Leak?

P1

Problem 2

```
int main() {
  int **array;
  array=(int **)malloc(sizeof(int *)*1000);

for(i=0;i<1000;i++)
  array[i] = (int *)malloc(sizeof(int)*500);

free(array);
  return 0;
}</pre>
```

(a) Where is the Memory Leak?

(b) How do you fix it?

P2

Performance Measurement



Lab 5 Goals

- 1. Measure program execution time
- 2. Break down program execution time by **specific subroutines / lines** of code
- Monitor program for memory leaks
 - Not really "performance", but uses same tool

Performance Measurement

Why is it <u>important</u> 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)
- **∇algrind produces a report that identifies these errors**

Valgrind Features

- **7** Callgrind module − Program profiler
- Callgraph shows you what function called what other functions
- How much CPU time does each function / code line consume?
- **∇algrind 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

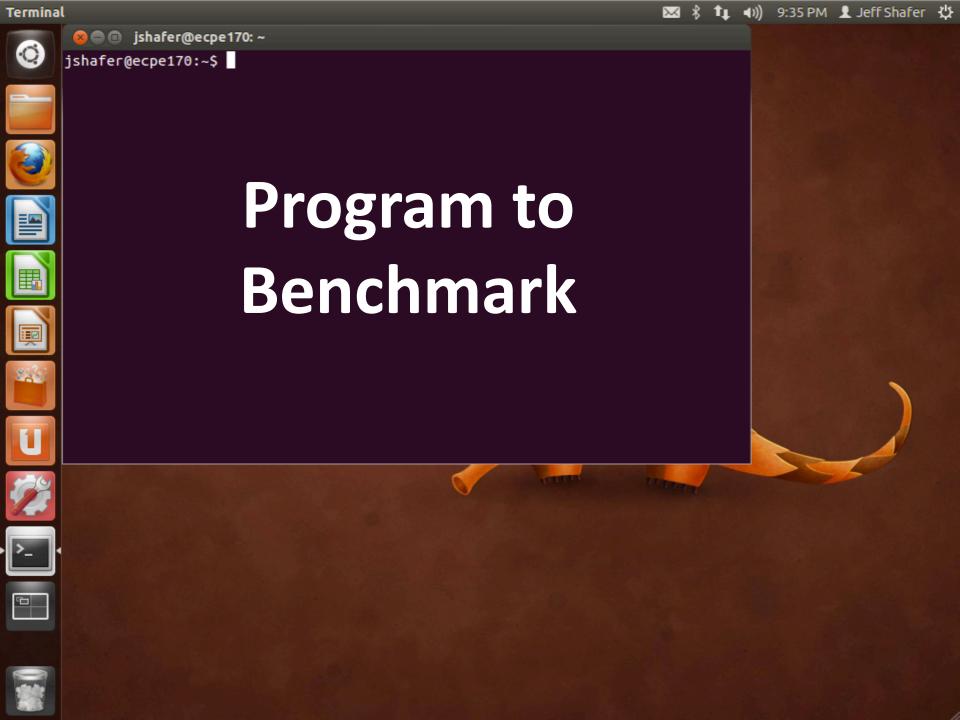
- Just-in time compiler
 - Your program is re-compiled onto a virtual (simulated) processor
 - Another example of a virtual machine!
- Benefit Valgrind can observe your program running at the machine instruction level
- Drawback Slow! (5x slower than normal)
 - **↗** But it's still better than fixing bugs without a tool...

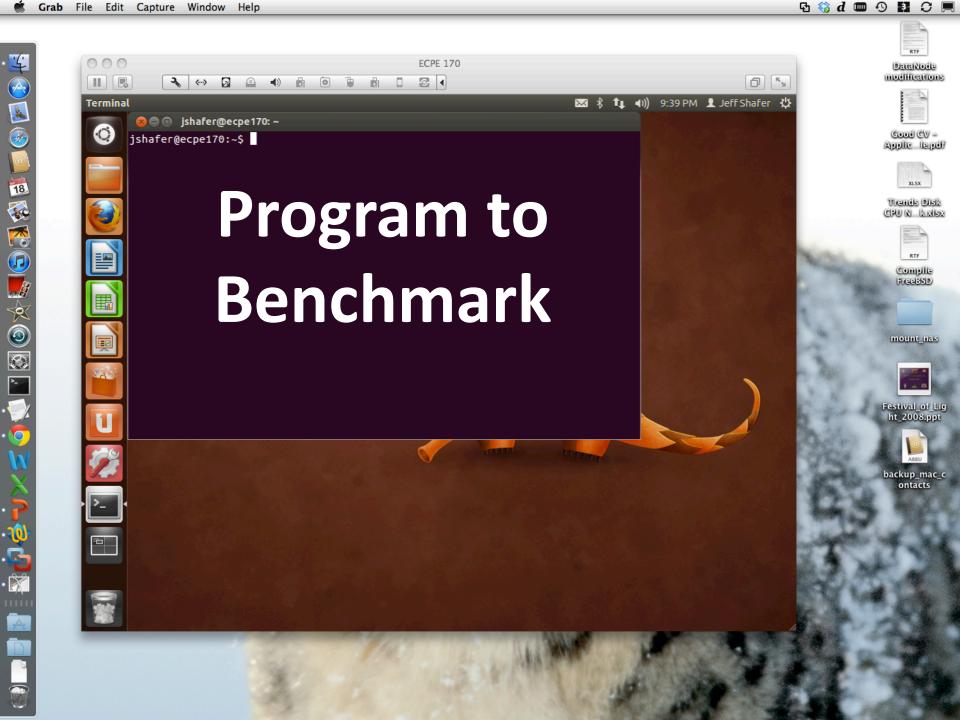
Profiling Basics

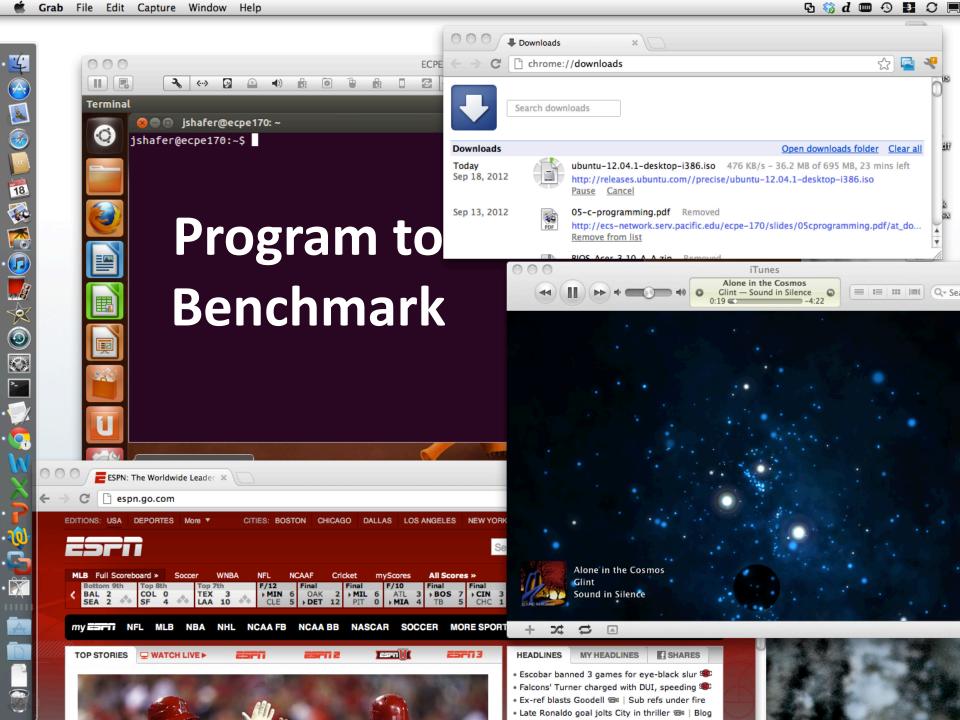


Profiling Basics

- 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 <u>best</u> in terms of producing the cleanest, most accurate, most reproducible results?







Profiling Basics

- 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