Benchmarking and Tuning
ACI-REF Workshop

Brett Zimmerman
University of Oklahoma – IT/OSCER

```
int power(int x, int y)
{
        int result;

        if (y < 0) {
                result = 0;
        } else {
                for (result = 1; y; y--)
                        result *= x;

        }
        return result;
}
```

A

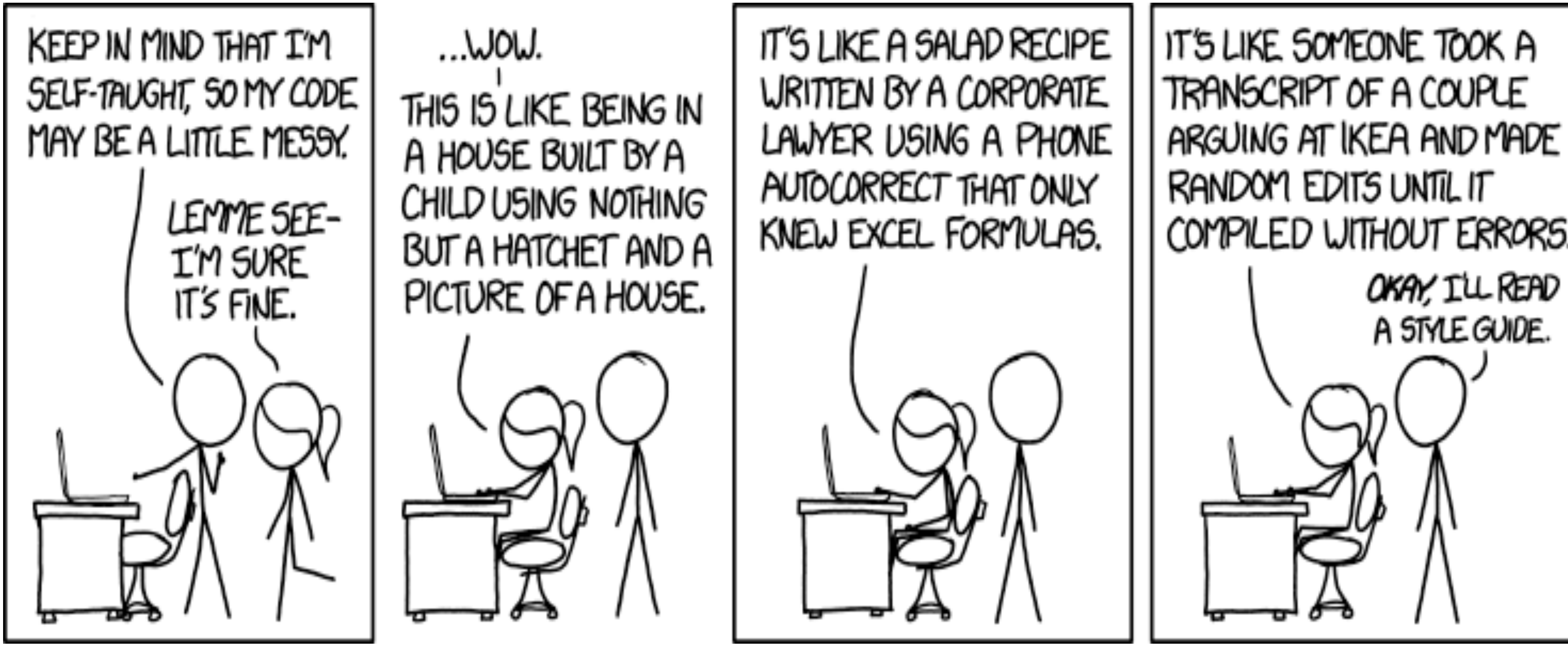
```
int power (int x, int y) {
   int result;
   if (y<0) {
      result = 0;
   } else {
      for (result = 1; y; y--)
          result *= x;
   }
   return result;
}
```

B

```
int power (int x, int y)
{
    int result;
    if (y<0)
    {
        result = 0;
    }
    else
    {
        for (result=1; y; y--)
        {
             result *=x;
        }
    }
    return result;
}
```

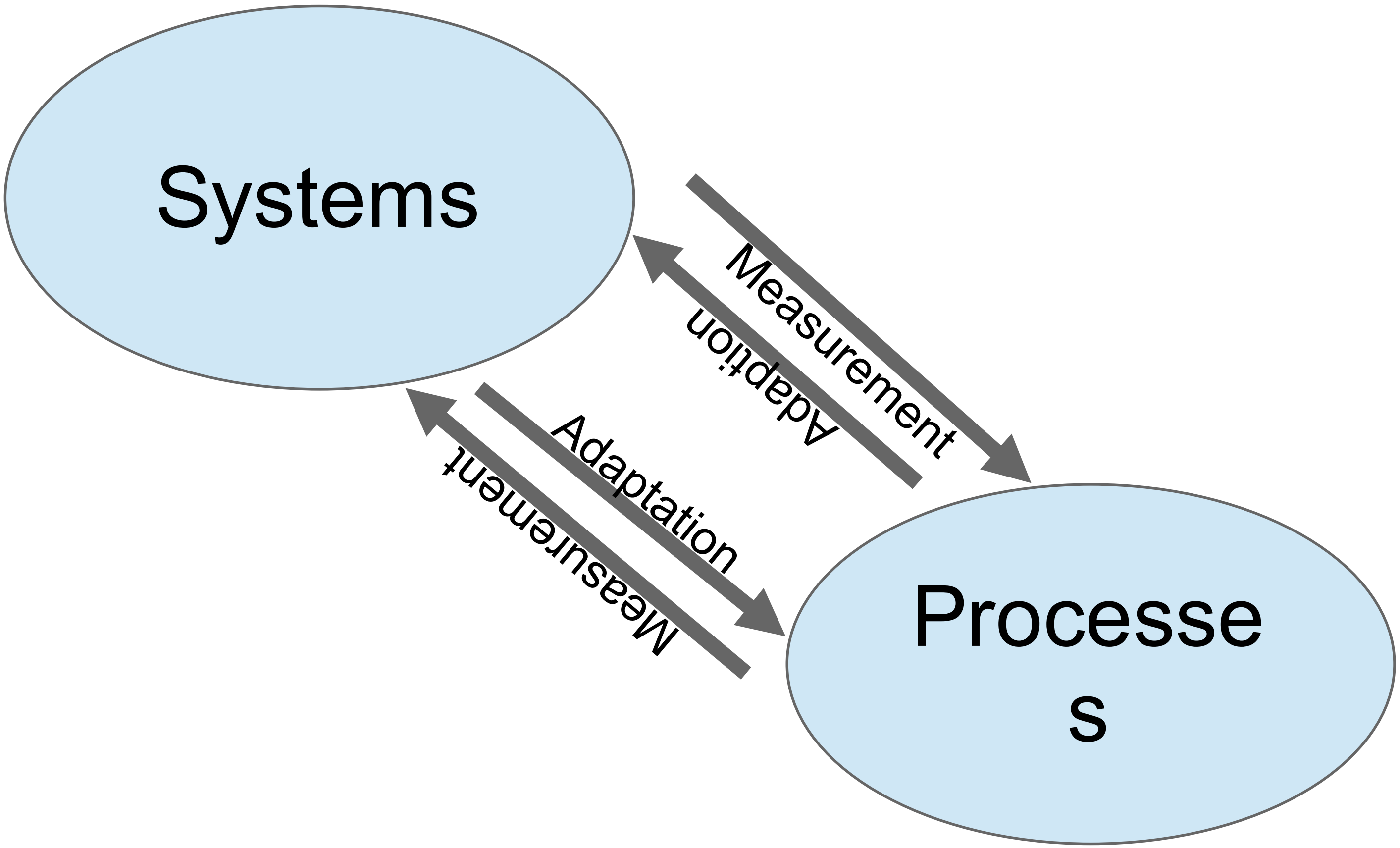
C

Don't



- Randall Munroe – http://xkcd.com/1513/

Complementary Processes



Resource Utilization

- CPU
- Memory (cache, RAM; space, bandwidth)
- Disk (intentional as well as swap and paging)
- Network (bandwidth, latency)

A bottleneck <u>always</u> exists.

Benchmarking and Tuning

- Systems
  - Measure the performance characteristics of the system
  - Adjust the system to accommodate a given use
    - Hardware adaptation
    - System tunables
- Processes
  - Measure the resource usage of a given piece of code
  - Adjust the code to make efficient use of the system

Units

- SI vs. IEC units
- KiB 2^{10} (1024) KB 10^3 (1000)
- MiB 2^{20} (1048576) MB 10^6 (1000000)
- GiB 2^{30} (1073741824) GB 10^9 (1000000000)
- TiB 2^{40} (1099511627776) TB 10^{12} (1000000000000)

Some Groundwork – Queueing

- $L = \lambda W$
  - Units of work within the system is equal to the product of the arrival rate of the units of work and the time the unit spends in the system

Broad Optimization Approaches

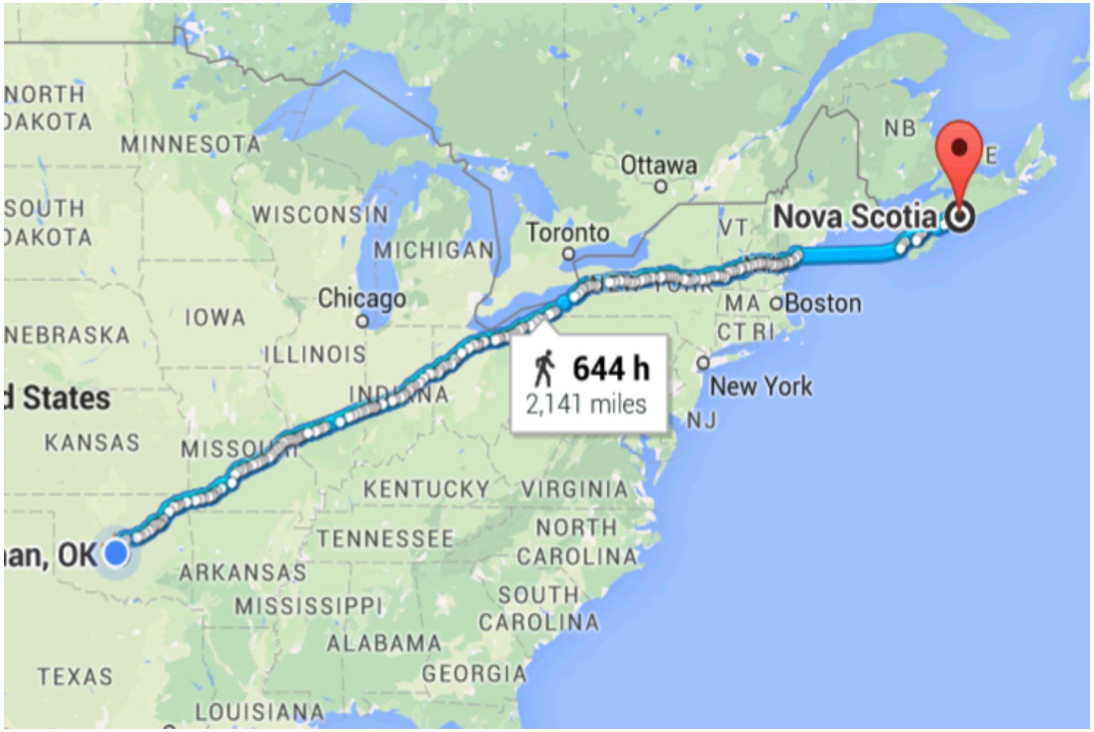
- Reducing Visit Counts
  - Amortize fixed overhead by aggregating operations
  - Not doing stuff
- Reducing Wait time
  - Reducing Overhead
  - Using multiple queues
  - Space tradeoffs/Caching
  - Efficiency
- Overriding concerns
  - The first priority for tuning is stability

Optimization Overview

- Outside measurement of resources
- Code review
  - Algorithm
  - Implementation
- Compiler
- 80/20 Rule
- Time/space tradeoffs
- Effect of the Storage Hierarchy

The Storage Hierarcy

- ~2 ns - L1
- ~5 ns - L2
- ~14 ns - L3
- ~60 ns - RAM
- ~3 ms - Disk



Determining System Performance

- Published specifications
  - Processor
  - Bus
  - Memory
- Known Benchmarks
  - Network (ping, netperf, qperf, OSU Micro)
  - IO (Iozone, Fio)
  - Global (HPL)

Coarse Timing Data

- /usr/bin/time rather than shell builtin

```
TOTAL RUN TIME: 0 days 0 hours 2 minutes 15 seconds
805 msec
Command being timed: "Orca/3.0.1/bin/orca
/home/zim/orca6.inp"
User time (seconds): 107.59
System time (seconds): 1.88
Percent of CPU this job got: 80%
Elapsed (wall clock) time (h:mm:ss or m:ss): 2:15.82
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 599168
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 219
Minor (reclaiming a frame) page faults: 598418
Voluntary context switches: 22746
Involuntary context switches: 1629
Swaps: 0
File system inputs: 642448
File system outputs: 787584
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

Measuring System Usage

- Crontab
  - */10 * * * *   /usr/lib64/sa/sa1 1 1 -S XALL
- Sysstat package
- Reports paging, IO usage, per block device usage, interrupt counts, power management, network utilization, cpu utilization, run-queue length, memory utilization, swap space, inode and dentry cache, switching
- e.g. sar -n EDEV will display per-device network errors for each network device on the system

System Tunables

- /proc
  - net
  - vm
  - kernel
  - fs
- proc.sys.net.ipv4.neigh.default.gc_thresh2
- Documentation
  - Kernel-doc package
  - /usr/share/doc/kerne-version/Documentation/sysctl

- ltrace -Sfc example

```
% time     seconds  usecs/call     calls      function
------ ----------- ----------- --------- --------------------
 50.03    2.159365     2159365         1 __libc_start_main
 46.43    2.004238       35162        57 strcpy
  0.63    0.027280          69       393 strlen
  0.43    0.018509       18509         1 getaddrinfo
  0.26    0.011290          68       166 malloc
  0.25    0.010619          68       156 free
  0.20    0.008802          68       128 realloc
  0.20    0.008774         165        53 SYS_open
  0.20    0.008771          67       129 ferror
  0.20    0.008723          69       126 fgets
  0.19    0.008099        8099         1 open
  0.09    0.003855          68        56 memcpy
  0.09    0.003750        1250         3 fclose
  0.09    0.003712         100        37 SYS_close
  0.06    0.002697        2697         1 connect
  0.06    0.002650         662         4 SYS_connect
```

gprof

- Compile with profiling
- Run code
  - Normal output
  - Side effect: profiling data written to gmon.out
- Report profiling data
  - gprof –line –flat-profile area-serial gmon.out

```
%    cumulative    self               self     total
time    seconds    seconds    calls  Ts/call  Ts/call  name
 0.00       0.00      0.00   200000     0.00     0.00  function_to_integrate
  (area_under_curve.c:139  @ 400b30)
 0.00       0.00      0.00        1     0.00     0.00  input_arguments
  (area_under_curve.c:55  @ 4008a8)
 0.00       0.00      0.00        1     0.00     0.00  sum_intervals
  (area_under_curve.c:108  @ 400a48)
```

Perf stat for Array walk

```
perf stat -Bd array_inner

 Performance counter stats for 'array_inner':

       4175.666892 task-clock                #    1.000 CPUs utilized
                 7 context-switches          #    0.002 K/sec
                 7 cpu-migrations            #    0.002 K/sec
             1,872 page-faults               #    0.448 K/sec
    11,577,144,187 cycles                    #    2.773 GHz                     [89.99%]
     3,943,947,255 stalled-cycles-frontend   #   34.07% frontend cycles idle    [89.99%]
     1,445,711,602 stalled-cycles-backend    #   12.49% backend  cycles idle    [79.98%]
    19,770,949,724 instructions              #    1.71  insns per cycle
                                             #    0.20  stalled cycles per insn [89.99%]
     4,361,497,083 branches                  # 1044.503 M/sec                   [89.99%]
         8,346,798 branch-misses             #    0.19% of all branches         [90.01%]
     3,844,013,042 L1-dcache-loads           #  920.575 M/sec                   [90.01%]
        81,717,861 L1-dcache-load-misses     #    2.13% of all L1-dcache hits   [90.01%]
         3,871,347 LLC-loads                 #    0.927 M/sec                   [90.01%]
         2,255,846 LLC-load-misses           #   58.27% of all LL-cache hits    [90.00%]

       4.176491482 seconds time elapsed
```

```
perf stat -Bd array_outer

 Performance counter stats for 'array_outer':

      13440.776165 task-clock                #    1.000 CPUs utilized
                18 context-switches          #    0.001 K/sec
                 5 cpu-migrations            #    0.000 K/sec
             1,872 page-faults               #    0.139 K/sec
    37,238,405,421 cycles                    #    2.771 GHz                      [89.99%]
    29,860,712,853 stalled-cycles-frontend   #   80.19% frontend cycles idle     [90.00%]
    23,981,968,848 stalled-cycles-backend    #   64.40% backend  cycles idle     [80.00%]
    19,791,761,293 instructions              #    0.53  insns per cycle
                                             #    1.51  stalled cycles per insn [90.00%]
     4,369,050,335 branches                  #  325.059 M/sec                    [90.00%]
         8,363,924 branch-misses             #    0.19% of all branches          [90.00%]
     3,851,175,888 L1-dcache-loads           #  286.529 M/sec                    [90.00%]
       563,864,367 L1-dcache-load-misses     #   14.64% of all L1-dcache hits    [90.00%]
        18,290,787 LLC-loads                 #    1.361 M/sec                    [90.00%]
        16,310,806 LLC-load-misses           #   89.17% of all LL-cache hits     [90.00%]

      13.442679796 seconds time elapsed
```

Perf Counters

- perf package
- perf list – List available counters and tracepoints
- perf top – System level profiling
  - /proc/sys/kernel/perf_event_paranoid – Allow unpriviledged users to collect performance counter data

  -1 not paranoid
- perf stat – Run a command and gather stats

SystemTap

- Allows fine-detail monitoring of the kernel
- systemtap, systemtap-runtime, kernel-debuginfo, kernel-debuginfo-common-arch, kernel-devel
- stap -v -e 'probe vfs.read {printf(“read\n”);exit()}'
- Systemtap automates adding instrumentation modules to the running kernel
- stapdev – privileged stap users. Effective root
- Stapusr – can use staprun to run modules in /lib/modules/*version*/system-tap/

SystemTap Example

```
global wevent,revent

probe vfs.write.return {
 wevent[execname()] += $return
}
probe vfs.read.return {
  revent[execname()] += $return
}
probe timer.s(10) {
   printf("___READS___\n")
   foreach(exe in revent- limit 10)
   printf("%15s: %d
bytes\n",exe,revent[exe])
   printf("\n")
   printf("___WRITES___\n")
   foreach(exe in wevent- limit 10)
   printf("%15s: %d
bytes\n",exe,wevent[exe])
}
```

```
___READS___
             dd: 335304334 bytes
     irqbalance: 288890 bytes
         crond: 57767 bytes
          sadc: 52016 bytes
          date: 13990 bytes
          sshd: 11952 bytes
    unix_chkpwd: 11934 bytes
           sa1: 11495 bytes
        screen: 5963 bytes
systemd-journal: 5112 bytes

___WRITES___
             dd: 335300191 bytes
           sshd: 12205 bytes
           sadc: 11228 bytes
         screen: 9077 bytes
         stapio: 3558 bytes
 systemd-logind: 2102 bytes
           ping: 1831 bytes
         auditd: 1783 bytes
        systemd: 472 bytes
          gdbus: 392 bytes
```