UNIX System and Network Performance Tuning

A tutorial introducing the gentle art of system performance tuning for UNIX system managers and like minded individuals wherein we propose to explain how, while systems are constantly growing faster with each passing year, they never are fast enough.

1

Topics

- **Introductory concepts**
- **CPU systems**
- **Memory systems**
- **Disk systems**
- **Windows, graphics and databases**
- **Networks**
- **Benchmarks**
- **Tuning up applications**
- **FINAL EXAM**

2

Introductory concepts

- What are the biasses of this course?
- The fundamental laws of system performance
- The fundamental laws of performance tuning
- Overview of hardware and O/S interaction

3

Course Biasses

- Much of system tuning is system and O/S dependent
- Examples based loosely on BSD UNIX
  - Example text reformatted to fit on viewgraphs
- When you try to do this stuff yourselves you will need to read the manual for your particular UNIX flavor
  - Pay close attention to the “see also” sections on performance related manuals
  - Some vendors provide better tools than others

4

Fundamental laws of system performance

- You can't have too much RAM
- The computer is never fast enough
- There is always one more bottleneck
- Every 2 years your computer is obsolete
- Every new software upgrade will cost performance
- Every new O/S upgrade will cost performance
- Performance is money
- Money can't buy happiness but it can buy performance

5

Fundamental laws of performance tuning

- Be a scientist
  - FIRST: Measure
  - SECOND: Hypothesize
  - THIRD: Test/Verify/Fiddle with things
  - GOTO FIRST
- Change only one thing at a time
- Note what you changed
- Save measurement data
- TANSTAAFL: There Ain't No Such Thing As A Free Lunch
- Don't look for a panacea

6

The uncertainty principle

- **Quantum mechanics discover that it is impossible to simultaneously measure both a particle's *location* and its *speed***
- **Heisenberg's uncertainty principle indicates that the act of measuring a system affects it - thereby throwing the measurement off**
- **The same applies for performance tuning!**
  - **Most system monitoring programs are resource pigs!**
  - **Some fancy X-windows based monitoring tools can completely crunch a system!**
  - **You must be able to factor (roughly) the effect of measuring into your measurements**

7

Hardware and O/S interaction

- **The key to understanding system performance is to understand how your software uses the underlying hardware**
- **Most performance problems are a result of one of:**
  - **Unbalanced systems**
  - **Incorrect algorithms**
  - **System bugs**
  - **Resource starvation**

8

Hardware and O/S interaction: unbalanced systems

- System capabilities must be on par to take advantage of eachother:
  - If you have a super hot CPU connected via a 14.4K PPP link your network will be ***slow*** even if you upgrade your CPU
  - If you have a super hot on a system with slow disk drives it will spend most of its cycles waiting for the disk to spin
  - If you have an incredibly fast disk connected to a slow controller it cannot transfer data faster than the controller
- Balancing system depends largely on what you want to do with it

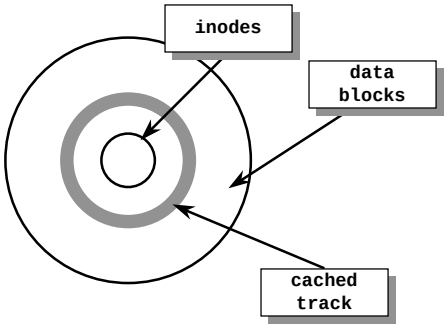
9

Hardware and O/S interaction: incorrect algorithms

- Sometimes system capabilities are ignored or even defeated by using the wrong algorithms either in the kernel or in applications:
  - One example is System V filesystem which (unwittingly) acted to defeat disk track caches
  - UFS (aka: "fast file system") disk geometry logic makes no sense on SCSI disks but is incredibly important on CDROMs
- What was a smart way to do things years ago may be a bad idea today because of changing technology
- If you are on or near the cutting edge you ***will*** get cut

10

Hardware and O/S interaction: incorrect algorithms - example



- To write a file block means updating inode and data block
- Requires a seek from inside to outside of disk for each write
- Seeks are slow
- Seeks also discard modern SCSI disk track caches

11

Hardware and O/S interaction: system bugs

- Bugs can manifest as performance problems
  - Network problems may cause NFS errors and retransmits which will ***appear*** to be a disk or server performance problem
  - Multiprocessor lock contention or lost locks may ***appear*** to be slow disks or unpredictable hangs
  - Memory leaks may cause slow system degradation [A favorite example is an X-Window server that slowly leaks memory and grows to 20-30MB before crashing the system]
- Tracking vendor bug reports and keeping the O/S upgraded regularly can be a cheap and easy form of performance tuning!

12

Hardware and O/S interaction: resource starvation

- **Configuring the system is a zero sum game:**
  - **In order to give in one place you usually have to take from another**
  - **The usual currency of system performance tradeoffs is RAM**
- **Example:**
  - **Marcus sets up a 16MB BSD-based system with MAXUSERS=128**
  - **MAXUSERS is used to calculate kernel process table sizes giving 2048 process table slots on Marcus' machine**
  - **2048 process table slots, plus associated stack pointers and other allocated memory uses *8MB of RAM* from the system!**
  - **By trying to make the machine faster Marcus actually slows it down and causes periodic crashes!!**

13

Hardware and O/S interaction: resource starvation

- **Resource starvation situations are the hardest to debug:**
  - **Often the system tries desperately to reallocate resources to get work done**
  - **This means that a shortage of RAM may *appear* to the user as slow disk I/O**
  - **Shortage of network buffers caused by shortage of RAM may *appear* as a slow network!**
- **Most system performance problems are the result of resource starvation ☹**

14

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

15

CPU systems

- The role of the CPU
- Measuring CPU performance
- Solving CPU performance problems

16

What does the CPU do?

- **CPU's role depends on your system**
  - **Some systems have hardware assists for various operations**
- **Generally assume the CPU has at least some involvement in everything that happens: it is the central coordinator of all activity**
- **General rule of thumb: things that happen *really* often will have a CPU impact**
  - **Network interrupts**
  - **Context switches**
  - **System calls**
  - **Bus chatter**
  - **Other mysterious things caused by space aliens**

17

What does the CPU do? The multiprocessor question

- **The idea of multiple processors is to let more than one thing happen at a time so more work gets done in the same amount of time**
- **Some things can't or shouldn't happen at once or the system will die a violent but creative death**

Therefore: something needs to control what can happen at once and what cannot

...and it's one hell of a bottleneck!

18

CPU performance measures: key concepts

- **Interactive response times**
  - **Also known as "*the natives are getting restless*"**
- **Load average**
- **CPU idle/system/user times**

19

CPU performance measures: key concepts: interactive response

- **When your mind-bogglingly fast RISC system *feels* slow to you, it probably *is*!**
- **Use the "time" command to get a rough idea how long some basic things normally take**
  - **Usually the same tasks should take approximately the same amount of time each time, all things being equal**
- **Scientific studies* show that computer user frustration rises sharply with delays in response**
  - **More than 1/2 second delay produces visible stress**
  - **Users begin repeatedly pressing <return> [3 times, on average]**
  - **Don't worry: if you're the systems admin they *will* let you know**

*** Yes, they really don't seem to have anything better to do**

20

CPU performance measures: key concepts: load average

- **Load average prints the average number of *runnable* processes during:**
  - **The last minute**
  - **The last 5 minutes**
  - **The last 15 minutes**
- **A runnable process is one that is ready to do work but cannot because the system is busy**
- **Load average is a very coarse approximation but usually when it is high you have a problem**
- **Some systems compute load average wrong and get very strange results**

21

CPU performance measures: key concepts: CPU idle/system/user

- **Many system diagnosis tools report CPU usage in terms of percent of time spent in:**
  - **Idle** - **The CPU is twiddling its little silicon thumbs**
  - **User** - **The CPU is doing computation on behalf of an application**
  - **System** - **The CPU is busy doing some kind of internal bookkeeping**
- **High values of system time usually indicate a performance problem:**
  - **I/O waits**
  - **Paging**
  - **System calls**
  - **Network service routines**

22

Measuring CPU: Uptime

```
. uptime
 11:48pm  up 1 day, 23:29,  49 users,  load average: 0.34, 0.11, 0.01
. while true; do
> sleep 1
> uptime
> done
 11:48pm  up 1 day, 23:30,  49 users,  load average: 0.21, 0.09, 0.00
 11:48pm  up 1 day, 23:30,  49 users,  load average: 0.21, 0.09, 0.00
 11:48pm  up 1 day, 23:30,  49 users,  load average: 0.21, 0.09, 0.00
 11:48pm  up 1 day, 23:30,  49 users,  load average: 0.28, 0.10, 0.01
 11:48pm  up 1 day, 23:30,  49 users,  load average: 0.28, 0.10, 0.01
 11:48pm  up 1 day, 23:30,  49 users,  load average: 0.28, 0.10, 0.01
 11:48pm  up 1 day, 23:30,  49 users,  load average: 0.28, 0.10, 0.01
^C.
```

you can tell nobody is doing much

23

Measuring CPU: vmstat

```
. vmstat 5
 procs     memory              page                    disk          faults
cpu
 r b w avm fre re at  pi  po  fr  de  sr s0 s1 s2 s3  in sy  cs us sy id
 0 0 0 0 2496  0  6  16   0   1   0   0  0  0  0  1 168 336  31  2  7 91
 0 0 0 0 2504  0  4   0   0   0   0   0  0  0  0  0  86 231  22  1  2 97
 0 0 0 0 2352  0  2   0   0   0   0   0  0  0  0  0  79 211  22  0  1 98
 0 0 0 0 2296  0  0   0   0   0   0   0  0  0  0  0  61 205  20  1  2 97
^C.
```

system time

user time

idle time

24

Measuring CPU: mpstat

```
. mpstat 5
 average      cpu 0        cpu 1        cpu 2
 us ni sy id  us ni sy id  us ni sy id  us ni sy id
  2  0  7 91   2  0  7 91   2  0  7 91   2  0  7 91
  0  0  1 98   1  0  1 98   0  0  2 97   0  0  1 99
  0  0  7 93   1  0  0 99   0  0  0100   0  0 21 79
  1  0  3 96   2  0  3 95   0  0  3 97   0  0  3 97
  1  0  4 95   2  0  5 94   1  0  3 95   0  0  3 96
  0  0  2 97   1  0  4 95   0  0  3 97   0  0  1 99
^C.
```

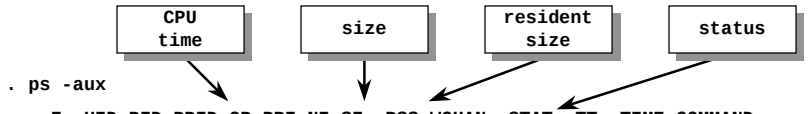
25

Measuring CPU: Accounting

- **Very system/version dependent**
- **Somewhat annoying to manage**
  - **Uses disk space**
  - **Some performance impact**
  - **Badly documented**
  - **Accounting reduction tools are awful. But that is what PERL is for**
- **Advantages:**
  - **Measure exactly which processes on your machine are biggest CPU or disk I/O hogs**
  - **Solve for once and for all vital issues such as what web browser is a bigger memory pig than the others**

26

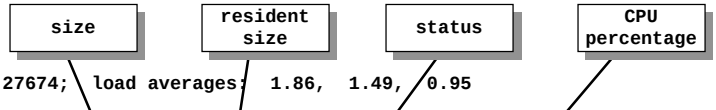
Measuring CPU: ps




```
. ps -aux
    F  UID PID PPID CP PRI NI SZ  RSS WCHAN  STAT  TT  TIME COMMAND
80003   0   0    0  0 -25  0   0    0 runout   D    ?   0:00 swapper
88001   0 187    1  0   1  0 104  360 select   I    ?   0:00 rpc.lockd
28001   0 195    1  2   5  0 400  816 child    I    ?   0:00 /usr/local/b
88001   0 198    1  0   1  0  96  392 select   I    ?   0:13 /usr/local/e
2001    0 213    1  0   1  0  40  112 socket   I    ?   0:00 sh -c while
26001   0 215 213  5   1  5 104  536 select   S N  ?   0:00 /usr/local/w
20001   0 218 215  5   1  5 112  560 select   S N  ?   0:12 wplmd60 -T s
20001   0 220    1  0   5  0  32  112 child    I    ?   0:00 /bin/sh /usr
800001  13239    1  0   1  0 184  944 select   I    ?   0:26 /usr/local/e
88001   0 241    1  5   1  0  72    0 socket   I    ?   0:00 /bin/httpd_3
```

27

Measuring CPU: top




```
last pid: 27674;  load averages:  1.86,  1.49,  0.95
          27681                   1.72,  1.47,  0.95     00:13:43
2264processes: 2533sleeping, 23running, 8 stopped
Cpu states:   0.8% user,  0.0% nice, 32.9% system, 66.2% idle
Memory: 170M available, 1245 in use, 465 free, 18M locked

 PID USERNAME PRI NI  SIZE RES STATE   TIME   WCPU    CPU COMMAND
27600 root   788 0   72K  328K rrun/2  1:33  79.42% 79.30 du
27670 mjr    538 0 14148K 1856 run/1   0:012 13.58%  7.03 top.Series5
27681 root    29 0  120K  232K run/0   0:00   0.00%  0.00% elpd
  257 root    1  0  120K  280K sleep   2:295  0.00%  0.00% elpd
  163 root    1  0   48K    0K sleep   2:04   0.00%  0.00% vnfsd
```

28

Solving CPU problems

- If your system spends most of its time with its time pegged in user time you may have overreached your CPU
- Graphics or numeric processing loaded systems are likely candidates for CPU problems
- Systems with highly loaded networks can spend so much time shuffling packets that they slow down
- CPU overload is the easiest problem to diagnose

29

Solving CPU problems: Bigger iron

- Buy a bigger system
- Add cache memory or a faster CPU
- Upgrading the whole system is usually better than just upgrading the CPU
  - Complete system upgrades may include bus speed increases which will help overall performance
- If at all possible get a loaner system and compare it with your existing system for the same load
- Take into account whether upgrading your hardware will require upgrading your software to possibly slower revisions

30

Solving CPU problems: More processors

- For compute-intensive applications multiprocessors are a win
- Understand the "parallelism" of your applications base
  - If your system is CPU-bottlenecked running a your single-process database server it will be worse on a multiprocessor system unless the database server is able to use more than one processor efficiently [Beware vendor claims here]
- For general multi-user workload multiple computers may outperform a multiprocessor box
  - Multiple machines increases aggregate bus bandwidth
  - More administrative hassles

31

Solving CPU problems: Less work

- Identify applications that can be offloaded to dedicated server systems
  - Example: buying a Pentium w/64MB of RAM as a dedicated news server is much cheaper than upgrading a departmental server
- Run processes at off-peak hours using cron or queueing systems
- Identify applications that are problems and tune them individually*

* This is what UNIX accounting was originally for

32

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

33

Memory systems

- The role of memory: real and virtual
- Measuring memory performance
- Solving memory performance problems

34

Real and virtual memory

- System has a limited amount of real RAM
- Pretends to have more by playing a shell game shuffling memory to and from disk
- System has a limited amount of virtual memory based on swap space
- For a fast modern processor having to page memory to and from disk has an impact on performance equivalent to a freight train slamming into the side of a cliff at 120MPH
- BUT some paging is normal and even good

35

Real and virtual memory: allocation strategies

- Some systems provide different views of virtual memory:
  - Total memory = sizeof(RAM) + sizeof(swap)
  - Total memory = sizeof(swap)
- Swap should always be larger than physical memory
  - How much depends on how much swap you usually use
  - Rule of thumb is 2-3 times sizeof(RAM)

36

Real and virtual memory: allocation strategies

- Some recommend multiple swap partitions on different controllers/disks
  - Really mostly useful for keeping paging (which is disk activity) off of disks that are usually active for filesystems
  - Remember: going to disk is so incredibly slow what's another millisecond between friends anyhow?
- Having a separate swap disk is a nice luxury
  - I like putting swap and /tmp on a small high speed disk
- Remember that paging contends for ***disk bandwidth*** with filesystem traffic

37

Real and virtual memory: allocation strategies

- Most versions of UNIX perform all kinds of clever tricks to try to page stuff out *before* it is *forced* to so that new requests for allocated memory can be fulfilled immediately
  - 2 hand paging algorithm
  - Scan rate
  - Minfree, Maxfree, and Desfree [the three evil brothers]
- Fiddling with these values is serious hard-core black magic juju
  - In fact it's so black magic that some vendors forget to adjust the values to take into account changes in technology and average memory sizes

38

Swapping

- Processes get swapped for one of two reasons:
  - They were asleep for a long time and its efficient for the system to kick the whole process out of memory and into long-term storage
  - The system is absolutely desperate for memory and it can't get by just stealing a few pages here and there so it starts to swap entire processes
- Some swapping activity is normal and good
  - Helps the system keep a free pool of memory
  - Nice place to store zombie processes
- Other types of swapping activity means your system has died and gone to hell

39

Paging

- 80% of the time a process is only executing 20% of its code
  - Hugely bloated X-Windows applications probably spend 95% of their time in 5% of their code
- Paging is a nice way of keeping just the important parts of a process in memory so it can run without hogging the system
  - Otherwise you'd need 1GB of RAM to support 30 users running Xmosaic or Netscape
- System uses version-dependent paging algorithms to shuffle less-often used pages of a program's address space to disk

40

Paging and the birth of a process

- Many versions of UNIX start a process by loading the beginning of the program and enough to get it started
- The rest of the program's text image is loaded into memory by page faulting it in on demand when a code segment gets activated
  - This is why the first time you pull down your "Hotlist" menu in Xmosaic it sits and "thinks" for a while - it is paging the code in
- Paging is most often visible when portions of user interface code get paged out and activating a seldom-used menu suddenly causes a lag

41

Paging and the birth of a process

- Page-ins occur as a normal part of the life of a process
- Page-outs occur as a normal part of the life of a process
- You only need to get concerned when the system is slow and is paging a lot

42

Thrashing

- Thrashing is the rare state where a program is trying to keep more things in memory than there is physical memory to fit and then it tries to access them all frequently
- This results in constant heavy paging activity
- Large old TinyMUD servers used to thrash systems
- Mis-configured database engines can thrash systems in exciting ways!
  - What if you have a box with 64MB of RAM and tell your database engine that it should keep a 64MB cache?
  - Remember there's system overhead: it doesn't ***really*** have 64MB to work with!

43

How memory is used in the system

- Other system resources consume memory:
  - Network buffers
  - TTY and PTY character typeahead
  - Process table slots
  - Inode cache
  - File system in-memory superblocks
  - Shared memory pages and semaphores
  - Mount table slots
  - File descriptor tables
  - Page table entries
- Increasing kernel values almost always consumes memory

44

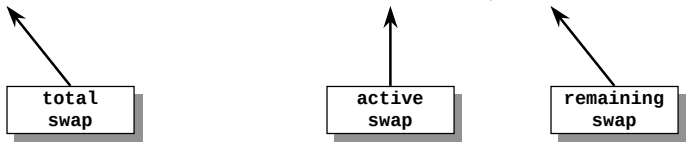
What happens when memory runs low

- The system always tries to keep a small “slush fund” of free memory to give out quickly when asked for it (desfree)
- When system needs free pages it starts to scan the page tables (2 hand scan)
- The more desperate it is the faster and more often it will scan
- When it cannot find enough fast enough it starts to swap
- When it starts to swap everything grinds to a halt

45

Measuring memory: pstat

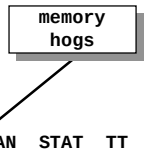
```
. pstat -s
55448k allocated + 20944k reserved = 76392k used, 538600k available
.
```



When out of swap entirely, processes start to die!

46

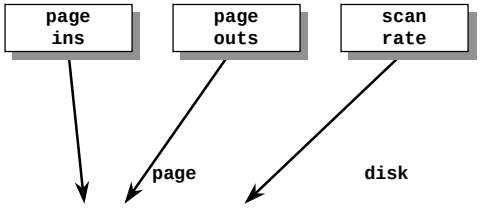
Measuring memory: ps




```
. ps -alx | sort -r -n +8
UID PID    PPID  CP PRI NI SZ  RSS WCHAN  STAT  TT  TIME COMMAND
175 15893 15884  0   1  0 920 1848 select   I    r2  0:57 pine
 59  9370 18173 18   1  0 584 1728 select   I    r1  0:05 gs -dQUIET -
174 23698 23191  3   1  0 624 1664 select   I    t1  0:52 xmh -geometr
 59 18173 16842  0   1  0 328 1528 select   I    r1  0:23 ghostview 5m
261  8197     1  0   1  0 328 1440 select   I    p6  0:04 xterm -title
261 25712     1  0   1  0 320 1368 select   I    ?   0:04 xterm -title
164 15882 15876  2   1  0 432 1368 select   I    r0  0:33 xmh -geometr
261 25721     1  0   1  0 320 1336 select   I    ?   0:03 xterm -title
^C
```

47

Measuring memory: vmstat




```
. vmstat 5
 procs     memory            page                disk       faults
cpu
 r b w avm fre  re at  pi po fr  de  sr s0 s1 s2 s3  in  sy  cs us sy id
 2 2 0   0 47600  0 6  16  0  1   0   0  0  0  0  1 168 342  31  2  7 91
 4 0 0   0 47456  0 8 104  0  0   0   0  1  1  0  3 407 559  53  2 59 40
 2 0 0   0 47312  0 1 104  0  0   0   0  0  0  0  0 3381632  47  3 64 33
 1 1 0   0 47280  0 0 112  0  0   0   0  0  0  0  0 279 979  40  3 56 41
^C.
```

48

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

49

Disk systems

- The role of disks
- The role of filesystems
- Measuring disk/filesystem performance
- Solving filesystem performance problems

50

Disk systems: the role of disks

- A disk is a place to store data you want to recover reasonably quickly when you want it again
- Physically disks have many properties
  - Many have on-disk caches of various sizes
  - Some perform read/write re-ordering and optimization
  - Some reveal their geometry
  - Some have flexible or meaningless geometry
- Anything this page says about disks will be obsolete by the time you read it
  - Technology moving very fast
  - Always getting smaller/faster/cheaper

51

Disk systems and filesystems

- Filesystem organizes data on disks (or similar devices) so that the user and system can keep track of it
- Filesystem design and database design are similar:
  - Tradeoffs of speed versus space
  - More efficient use of space seems to imply slower write throughput
  - Less efficient use of space usually implies faster write throughput [tossing socks on the floor is faster than putting them in your sock drawer but sooner or later it doesn't scale well and domestic harmony degrades]
  - Tradeoffs of speed versus reliability
  - Balance for read throughput or for write - never both

52

Disk caches and filesystem caches

- Most systems cache disk access
- Reads (and sometimes writes) are actually from/done to memory instead of to disk
- For multiuser systems a cache ***inverts*** the throughput equation:
  - 85% of I/O on average is reads
  - 95% of reads are fulfilled by the cache
  - ***Therefore the I/O mix becomes 5% reads and 95% writes!!!***
- Cache write policy is very version dependent and very arcane and extremely important

53

File system performance

- Filesystem performance largely determined by layout of blocks and meta-data
  - More robust filesystems write more meta-data which makes them slower
  - Faster filesystems store more information about layout in memory which makes them faster at reading but makes cache write policy more complex [BSD, for example, caches pathname components]
  - Writing big chunks at a time is faster than writing little chunks
  - Writing little chunks means you can use space more efficiently
- Filesystems and databases share many common design issues

54

File system performance: reading

- The more information the filesystem keeps in memory about where data is on disk the faster it can read
  - But the more memory it takes from the rest of the system
- UNIX filesystems use predictable sized block values to allow quick location of any single block in a file
- File blocks are read into cache and passed to application in smaller chunks
- Most versions of UNIX read-ahead by one block to try to already have data in memory
  - This is a gamble that usually pays off

55

File system performance: writing

- Writing data entails multiple I/Os to different parts of the disk:
  - Write the data block itself
  - Write the file's inode with new file size
  - Write the disk free block map when blocks are allocated
- Writing data so the filesystem will survive a crash entails writing data in synchronous mode:
  - System waits until the disk announces the data is written*
  - Inode and free block/superblock writes are synchronous
- This is why restoring a filesystem is so much slower than dumping it

* And we all know disks never lie

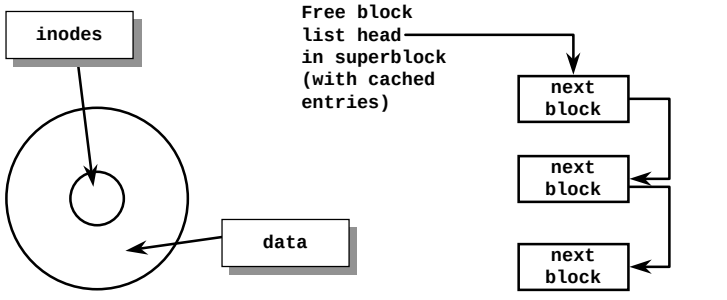
56

File system performance: allocating

- Allocation strategy very system dependent
- Typically blocks are allocated to be as close to other blocks of same file as possible
  - Minimize head seeks when reading/writing same file
  - Take advantage of disk internal caches if present
- UNIX file system maintains a free block bitmap and allocates quickly by searching in-memory structure
  - Updating filesystem meta-data still very expensive!
- Old UNIX file system allocated blocks from a linked list

57

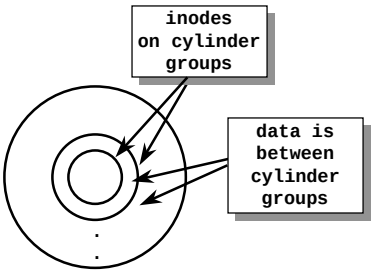
File system performance: System V filesystem

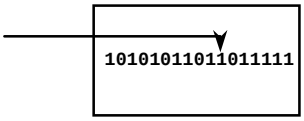


When free list gets scrambled the filesystem is corrupted

58

File system performance: UFS filesystem

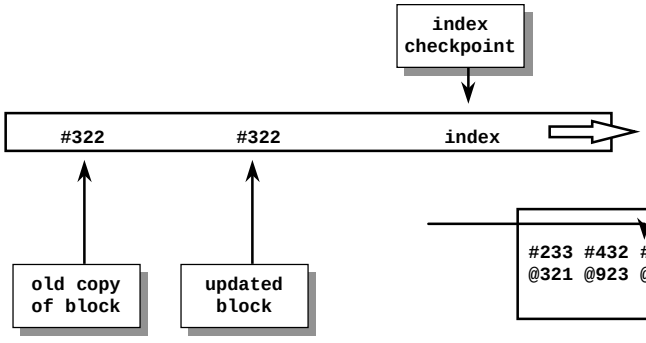




When allocating a block system searches in-memory bitmap based on cylinder group summaries

59

File system performance: Log / journaling systems



When allocating a block system searches in-memory index

60

File system performance: silicon disks

- Silicon disks have the advantage of very high speed for writes
- Usually fairly small
- Usually fairly expensive
- Widely used in databases for storing transaction logs and meta-data
  - Some silicon disks used to boost NFS and filesystem performance by absorbing filesystem meta-data writes to battery-backed memory and later committing them to real disk
  - NVRAM can boost NFS as much as 200% and regular UNIX filesystems as much as 15%
- Can turn I/O bottleneck into a CPU bottleneck

61

File system performance: RAIDs

- RAID devices aggregate multiple disks into a single virtual disk
  - Different RAID “levels” represent different update policy choices and different meta-data redundancy choices
  - Just because the RAID “level” is high does not mean it is better
- RAIDs give performance boost by either striping all I/O across the entire set of disks or by including RAM caching and NVRAM caching
- Some in-kernel RAID software can turn a disk bottleneck into a CPU bottleneck
- For all intents and purposes a RAID looks like a single disk to the system

62

Network File Systems: NFS

- In an attempt to insure integrity NFS protocol spec requires that all writes be synchronous
  - Therefore all writes are generally painfully slow
  - Some vendors support “asynchronous NFS” which is 200-700% faster
  - Asynch NFS should only be used for disposable data
- Generally NFS should be used to store shared non-disposable data only
- Local disk bandwidth can give better application performance for system executables, etc

63

Network File Systems: AFS

- Like NFS but with disk block caching to local hard disk
- Gives significant performance improvements over NFS for read-intensive applications

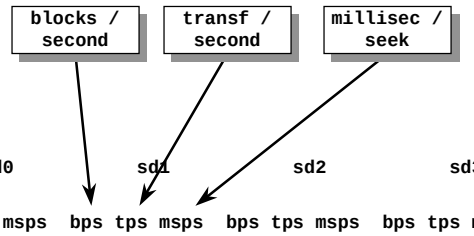
64

Measuring filesystems

- The main useful measurement of filesystem activity is how many blocks of data are moving back and forth
- Cross-index filesystem physical devices with mount points to determine what directories are “hot spots” and consider relocating them
- If one disk has significantly more I/O on it it is possible that 95% of the system may be waiting for that one disk to move data
- If one filesystem is a “hot spot” and it can be replicated consider doing so a cheap form of RAID

65

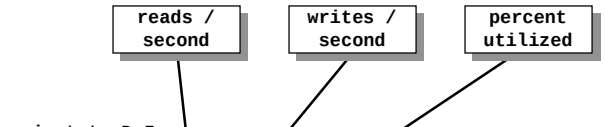
Measuring filesystems: iostat




```
. iostat 5
      tty      sd0          sd1          sd2          sd3
cpu
tin tout bps tps msps  bps tps msps  bps tps msps  bps tps msps
 4  274  4   0   15.6   1   0 13.6    5   1 12.0   24   4 13.2
 0   21  0   0   9.0    8   1 18.2    2   1 18.8    0   0 19.3
 0   15  0   0   0.0    0   0  0.0    0   0  0.0    3   0 12.3
 0   16  0   0   0.0    0   0  0.0    0   0  0.0    2   0 27.3
^C.
```

66

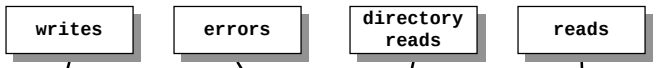
Measuring filesystems: iostat




```
. iostat -D 5
          sd0           sd1           sd3           sd7
 rps wps util  rps wps util  rps wps util  rps wps util
   0   0  2.0    1   1  3.6    3   3 13.6    7   6 25.9
   1   1  8.5    1   0  2.2    6   6 20.8   23  18 82.0
   0   0  1.2    1   0  2.4   10   7 37.2   17  16 71.3
   1   1  8.8    1   1  6.2    7   7 32.1   20  14 74.9
   1   0  4.2    1   0  1.4    8   7 38.2   15  13 58.3
^C.
```

67

Measuring filesystems: nfsstat




```
. nfsstat -c
Client rpc:
calls    badcalls retrans  badxid   timeout  wait     newcred  timers
1113240  59       12656    12145    12701    0        0        72441
Client nfs:
calls   badcalls   nclget     nclsleep
1113240 59         1113240    0
null    getattr    setattr    root       lookup     readlink   read
0  0%   136235 12% 2487  0%   0  0%      138468 12% 10152  0% 790623 71%
wrcache write      create     remove     rename     link       symlink
0  0%   15615  1%  8208  0%   4094  0%   3550  0%   1  0%      0  0%
mkdir   rmdir      readdir    fsstat
0  0%   0  0%      3717  0%   90  0%
.
```

68

Improving performance: buffer cache

- Some versions of UNIX use a fixed size buffer cache for filesystem I/O
- Increasing the buffer cache size will improve I/O but may trigger paging or resource starvation elsewhere in the system
- Before increasing buffer cache be careful to measure the existing system's virtual memory usage

69

Improving performance: more RAM

- Most current versions of UNIX use the virtual memory system as a disk cache as well
- I/O and memory bandwidth now entwined
- Adding more RAM will help both virtual memory performance and disk I/O
- I/O intensive applications reduce the available memory bandwidth for other applications
- Memory pig applications reduce the disk I/O bandwidth

70

Improving performance: load spreading

- Keep Statistics to determine which disk devices are most heavily loaded
  - Run iostat every 1/2 hour on each device for a week
  - Summarize reads and writes - some sites make pretty plots with excel or jgraph
  - Map devices to mount points
  - Examine statistics to see which devices are most busy and see if you have more than one important filesystem on the same disk
- When installing more disk space spread heavily loaded filesystems rather than simply mounting the new disk as a set of empty filesystems

71

Improving performance: NFS tricks

- Track NFS services on your fileserver using nfsstat to examine server statistics
- NFS server statistics don't tell what filesystem the accesses are against
  - Guess :)
  - Try to cross-index with iostat output
- If a filesystem is very busy and contains "disposable" data consider moving copies of the data to client local disk
  - Remember: kicking netscape off over an NFS-mounted filesystem triggers about 1MB of NFS I/O in no time at all

72

Improving performance: NFS tricks

- Check for clients that are causing especially heavy loads and examine them
  - Look out for cron jobs on clients that do a “find” [Some systems ship this way!]
- Check the state of nfsds on the server
  - Adding nfsds will allow the server to simultaneously service more NFS client requests
  - More nfsds means the server can work HARDER than before
  - More nfsds does not make the server faster but may speed things up for the client

73

Improving performance: NFS tricks

- Asynchronous NFS
  - NFS writes are always supposed to be synchronous
  - Makes NFS writes incredibly painfully slow
  - Some vendors support asynchronous NFS in which the server tells the client it has completed the write before it actually has
  - If the server crashes the client will think the file was successfully written but it wasn’t
  - Asynchronous NFS is best for things like /tmp - *but usually you don’t want a dynamic filesystem like /tmp NFS-mounted in the first place!*
  - Depending on size of server memory can provide up to 700% write performance boost

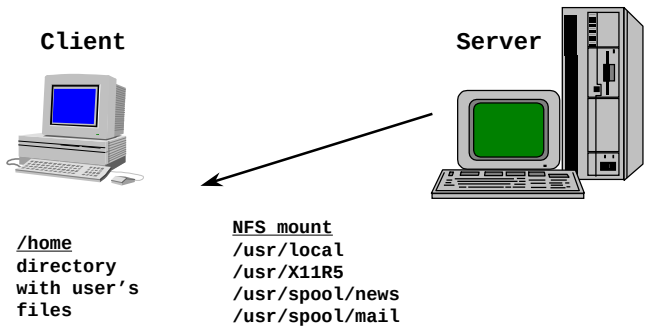
74

Improving performance: NFS tricks

- **NVRAM cache**
  - **System modified to write NFS write requests to NVRAM instead of disk**
  - **Client receives NFS write ack instantly**
  - **Asynchronous process flushes NVRAM cache through normal filesystem on server**
  - **Cache preserved across boots**
  - **Depending on cache size typically provides a performance boost of up to 300%**
  - **If cache is on motherboard make sure that the cache is flushed when field service replaces mother board. Also make sure that new motherboard caches are cleared before installing. :)**

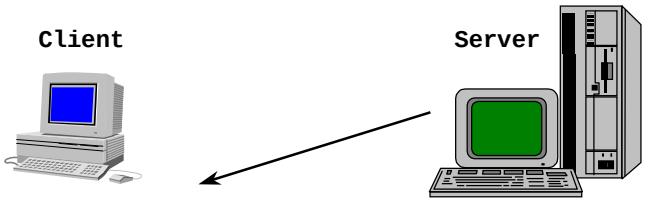
75

Improving performance: NFS set up wrong



76

Improving performance: NFS set up wrong



local system files	NFS mount
/	/home
/usr	/usr/spool/mail
/usr/local/bin	
/usr/X11R5	

77

Improving performance: smarter applications

- Some applications do not use disk bandwidth wisely
  - Shell scripts on NFS clients that generate large temporary files instead of using pipes are very slow
  - Using standard I/O library matches buffered I/O to system block size
  - Applications that rely on NFS locking are very slow*
  - Applications that use read/write may do much more I/O than necessary if not designed carefully
- If you narrow your I/O problems down to a few applications use application tuning techniques on them

* Also, they are crazy to rely on it. It's usually crippled by bugs

78

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

79

Windows, terminals, and graphics

- Effects of windowing systems on performance
- Improving X-Window performance
- Effects of graphic accelerators
- Effects of hardwired terminals

80

Effects of windowing systems on performance

- **Windowing systems result in lots of context switches**
  - **Mouse focus changes wake up and put applications to sleep**
  - **For fun, on a workstation move your mouse around while running vmstat and watch its effect**
  - **For fun, use MOTIF on a workstation with 16MB of memory and several Xterms open, compile something, and run vmstat, then click on a menu. :)**
- **The fancier the window the more memory it takes**
- **Putting a nifty background on your screen eats up (screenheight x screenwidth) bytes = usually 1MB**
  - **Put a nifty background up, run vmstat, and move windows around on your screen**

81

Improving X-Window performance

- **Buy more memory**
- **Buy a faster machine**
- *****

```
* I'd love to say "don't use X"
but that isn't an option these days
```

82

Effects of graphic accelerators

- **Custom graphic accelerators often require custom X servers or drivers**
  - **Limits your options for building local versions of X**

  [Sad But True story: one site buys a bunch of fancy 3D graphics boards and then builds and runs a local version of X11R5 instead of the X11R4-based server from the vendor. It takes a while for them to figure out why it is slow]
  - **Sometimes accelerators require more system memory**
- **If you are doing serious graphics expect to buy serious memory**

83

Hardwired terminals

- **When using hardwired terminals the CPU gets an interrupt for every keystroke**
- **Many modern versions of UNIX not optimized to lots of character I/O**
- **Offload keystroke processing to terminal servers if possible**
- **If system has hardwired terminals connected and has performance problems use iostat to see if it is doing a lot of terminal I/O**
  - **Sometimes a terminal wire goes bad and loops the operating system with a load of garbage**
  - **Getty will usually complain in syslog**

84

Databases

- A simple view of databases
- Resource effects of databases
- Measuring database impact on performance
- Solving database performance problems

85

Introduction to databases

- Most modern DBMS servers act like a miniature operating system
  - Handle their own cache
  - Do their own internal scheduling
  - Manage locking ***sometimes*** using O/S locking
  - Manage network connections
  - Use raw disk partition to avoid having UNIX do I/O buffering
- Many operating system tuning concepts apply to DBMS'
- Using UNIX tools can help you tell what the DBMS server is doing [sometimes]

86

Resource profile of databases

- Use ***lots*** of memory
- Large writes to raw disk partition
- Small synchronous writes to database journal
  - Some DBMS' write journal files to UNIX filesystem others to raw partition
- May use large amounts of CPU during query optimization and index joins
- Usually I/O bound for writes
- Usually CPU bound for reads

87

Measuring database impact on system

- Use iostat to see disk read/write mix on database devices
- Use vmstat to see virtual memory use patterns
- Use netstat to estimate network traffic caused by server
- Whenever possible when tuning a database make sure it is on a system by itself or otherwise characterize and "subtract" out user load

88

Solving database performance problems

- Usually the vendor can help
- Sometimes the vendor doesn't have a clue
  - Get vendor technical support in contact with the O/S vendor technical support
- Treat the DBMS as a normal application and use normal system performance tuning techniques
- If you imagine it is an operating system within an operating system you will approach the problem from the right point of view

89

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

90

Networks

- The role of networks
- Measuring network performance
- Solving network performance problems

91

Overview of networks

- TCP
  - Reliable end-to-end virtual circuit with sequenced delivery
  - High throughput
  - Congestion control moderates data rate
- UDP
  - Unreliable “message” delivery without sequencing
  - High throughput
  - Can spew packets as fast as the machine can send them*
- Those who don’t understand TCP are doomed to re-invent it
  - The world is full of UDP-based code with loads of logic that does what TCP does better [like NFS]

* Faster, actually. When it can’t send them it throws them away.

92

Measuring network performance

- **Ping**
  - **Measures packet round trip time**
  - **Useful for measuring packet lossage between hosts or networks**
- **Traceroute**
  - **Tries to determine route that packets are taking between hosts or networks**
  - **Useful for detecting weird routing situations [Sad But True story: one organization was routing traffic between 2 machines over an international link because the admin didn't understand the concept of subnets. Needless to say, this was slow.]**
- **Netstat**
  - **Tons of useful information about how much data system has sent and received**

93

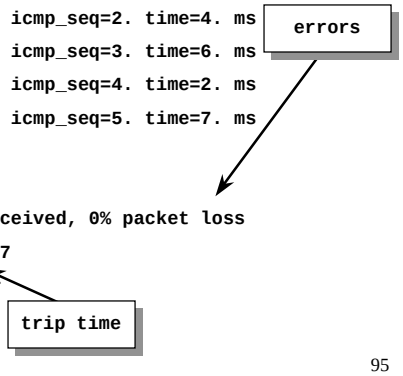
Measuring network performance

- **Ttcp**
  - **Measures TCP throughput between hosts**
- **NNstat**
  - **Very configurable network statistics sniffing and gathering tool**
  - **Cryptic and awkward to use**
  - **Generates really nice reports**
  - **Free [Do an Archie search]**
- **Etherman / Interman / Packetman**
  - **Graphical load monitor**
  - **X-based**
  - **Running it on your system will create other performance problems!**
  - **Free [Do an Archie search]**

94

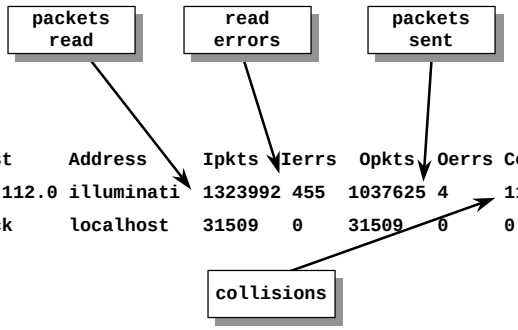
Measuring networks: ping

```
. ping sol
PING sol: 56 data bytes
64 bytes from sol (192.33.112.100): icmp_seq=0. time=2. ms
64 bytes from sol (192.33.112.100): icmp_seq=1. time=5. ms
64 bytes from sol (192.33.112.100): icmp_seq=2. time=4. ms
64 bytes from sol (192.33.112.100): icmp_seq=3. time=6. ms
64 bytes from sol (192.33.112.100): icmp_seq=4. time=2. ms
64 bytes from sol (192.33.112.100): icmp_seq=5. time=7. ms
^C
----sol PING Statistics----
6 packets transmitted, 6 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 2/4/7
.
```



95

Measuring networks: netstat




```
. netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs  Opkts  Oerrs Collis
le0   1500 192.33.112.0 illuminati  1323992 455  1037625 4      11715
lo0   1536 loopback     localhost    31509   0    31509   0      0
.
```

A few collisions is OK. This is less than 2%!!

96

Measuring networks: netstat

```
. netstat -m
192/320 mbufs in use:
        1 mbufs allocated to data
        1 mbufs allocated to packet headers
        75 mbufs allocated to socket structures
        99 mbufs allocated to protocol control blocks
        3 mbufs allocated to routing table entries
        11 mbufs allocated to socket names and addresses
        2 mbufs allocated to interface addresses
0/28 cluster buffers in use
68 Kbytes allocated to network (35% in use)
0 requests for memory denied
0 requests for memory delayed
0 calls to protocol drain routines
```

how much of the system's network buffers are left

can indicate a system without enough network buffers or RAM

97

Measuring networks: ttcp

TCP thruput

load / thruput

```
. ttcp -t -s illuminati
ttcp-t: nbuf=1024, buflen=1024, port=2000
ttcp-t: connect
ttcp-t: 0.0user 0.4sys 0:05real 9% 0i+43d 21maxrss 0+0pf 378+5csw
ttcp-t: 1048576 bytes processed
ttcp-r: 0.0user 0.2sys 0:05real 4% 0i+29d 19maxrss 0+1pf 773+774csw
ttcp-r: 1048576 bytes processed
ttcp-r: 0.26 CPU sec  = 3938.46 KB/cpu sec,  31507.7 Kbits/cpu sec
ttcp-r: 5.98 real sec = 171.074 KB/real sec, 1368.59 Kbits/sec
.
```

98

Measuring networks: NNStat

- Daemon listener
  - Accepts flexible and very powerful directions about what types of things to look for and what kinds of statistics to maintain about them
  - Efficient listener doesn't use much CPU
- Client stats-poller
  - Periodically downloads from daemon to disk
  - Can merge reports from multiple listening posts
- Interactive access
  - Administrator can interactively query server for statistics
- This is a good tool if you want to summarize traffic between hosts for a given service

99

Measuring networks: etherman, packetman, interman

- Graphical interface based on X
  - Represents a LAN with nodes
  - Plots traffic between nodes using bars of various thickness to indicate level of traffic
- On busy networks can completely swamp the workstation it is running on
- Very useful for a "one shot" check of network traffic
- Pretty pictures are very useful for printing out
- "Poor man's network management station"

100

Improving network performance

- 95% of network problems are either configuration errors or applications overloading the network
  - Track configuration errors using system tools, etherman, NNstat, etc
  - Track applications using packetman, NNstat, etc
  - Check for high levels of collisions or errors using netstat
- If networking on a given system is slow
  - Check netstat -m to see if low on memory buffers
  - Increase MAXUSERS on system to increase memory for networking
  - Don't make MAXUSERS too big or you'll actually waste memory

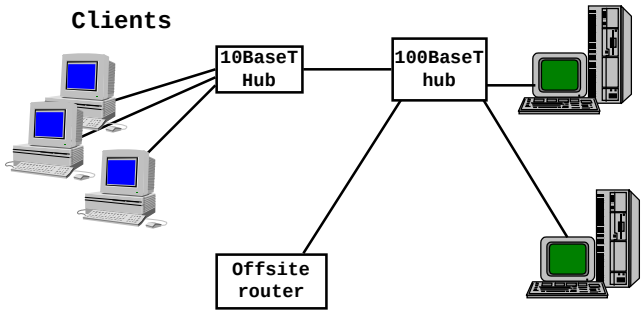
101

Improving network performance

- The Best way to improve network performance is to design your network carefully
- Put high point-to-point load servers on private networks with separate interfaces if local networks are swamping
- Use "smart" bridges or route traffic
- Move noisy workstations to segments behind "smart" bridges

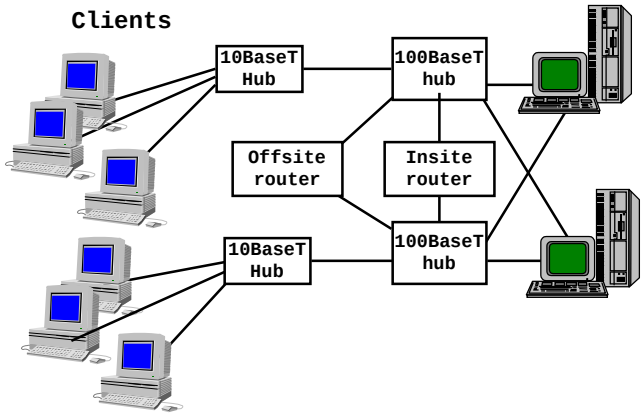
102

A simple network layout



103

A simple network layout grows



104

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

105

Benchmarks

- How vendors cheat on benchmarks
- How to interpret a benchmark
- Benchmarks you can trust(?)
- How to do a benchmark

106

How vendors cheat on benchmarks

- **The case of the growing SPECMARKs**
  - **Within one operating system release a vendor improves performance in floating point by a factor of 2**
  - **Older systems by same vendor benchmarks no longer published but their performance increases by 2 *also***
  - **Conclude: Someone cracked a benchmark**
  - **One year later all vendors reporting inflated floating point**
  - **SPEC Cooperative improves benchmark in next version**
- **Moral: When a benchmark is published that is an *average* of a number of test results check the entire test range and look for tests that match your needs**
  - **Also look for numbers that are seriously out of range**

107

How vendors cheat on benchmarks

- **The case of the "cache thrasher" benchmark**
  - **Vendor 'A' discovers a side effect of how Vendor 'B's UNIX implementation manages buffer cache**
  - **Vendor 'A' designs and publishes a benchmark that their system handles gracefully, which completely blows Vendor B's machine out of the water**
  - **Next release of Vendor B's system performs same as Vendor 'A' on the benchmark**
- **Moral: There is lots of kruft in vendor UNIX versions just for benchmark cooking**
  - **When you run it on your machine you'll discover that it causes resource starvation that chokes the system elsewhere**

108

How vendors cheat on benchmarks

- The case of the "custom library"
  - Customer provided benchmark uses fgets() to read a big file
  - Vendor runs the benchmark after first loading the file into the buffer cache by doing a "cat file > /dev/null"
  - Vendor compounds the performance increase by linking the benchmark against the Cnews fast fgets() implementation that is about 2X faster
  - Vendor gains approximate 400% performance increase
- Moral: Applications tuning is almost always applied to benchmarks

109

How vendors cheat on benchmarks

- The case of "I know the answer"
  - A large computer magazine uses seive of eratosthenes for the first 1000 primes as a benchmark
  - A compiler writer hand-codes a recognizer into the compiler to identify the benchmark at compile time
  - Compiler simply generates the correct answers on standard output
  - Benchmark runs in under a millisecond
- Moral: If the answer is too good to be true it probably is
- In this case the compiler writer did it as a joke and told the perplexed victim what he had done

110

How to interpret a benchmark

- Before you even look at a benchmark characterize your system and have a good idea what it needs to do well
- Look for benchmark figures related to those attributes ***only***
  - If your system will be running a DBMS ignore values that are "cumulative" test values that include things like floating point into an average value
  - Most "real" benchmarks include per-test-function measures
- Look for information about testing methodology
  - Some benchmarks specify that the benchmark be run on a machine using "as shipped" software and "as shipped" configuration

111

Benchmarks you can trust(?)

- Generally computer trade magazines that do benchmarks do not cook them for specific vendors
- Some computer trade magazine benchmarks are really really stupid [seive of eratosthenes?]
- Don't trust a benchmark that was run by the vendor - ***ever***
- Synthetic benchmarks (e.g.: SPEC) more useful than simulated workload (e.g.: AIM) only if all you do is number crunching
- The best benchmark is your application running on the system after *you* set it up and run it *yourself*

112

How to write a benchmark

- Don’t
- Use your real application that you install the way you would run it

113

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

114

Tuning applications

- The importance of application tuning
- Measuring application performance
- Solving application performance problems

115

Tuning Applications

- All the time your system spends in user time is a place where applications tuning may be applied
- Tuning applications is much cheaper (sometimes) than buying a new machine
- The case of the Real Dumb Database:
  - Site upgrades server every year for 3 years
  - One day a programmer discovers that a critical part of the application performs a linear search on a file
  - Application is rewritten to use a dbm database
  - New version of application running on desktop machine is now much faster than old version running on a big server

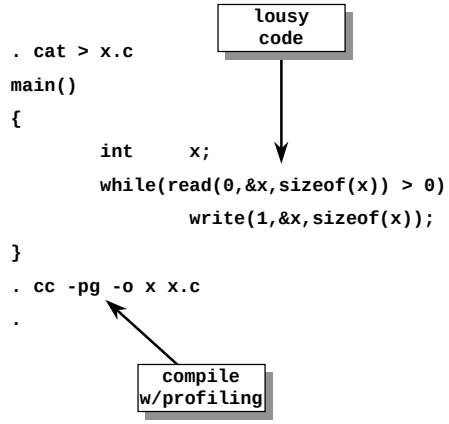
116

Measuring applications: accounting

- Accounting is a good way of seeing what applications your machine spends most of its time servicing
- Determine which are most CPU intensive and which are most disk intensive
- Worry only about the 2 or 3 at the top of the list
- If 5 applications use 50% of your system's compute cycles and you can speed them up by a factor of 2 each you have just gained the equivalent of a processor upgrade to next year's technology

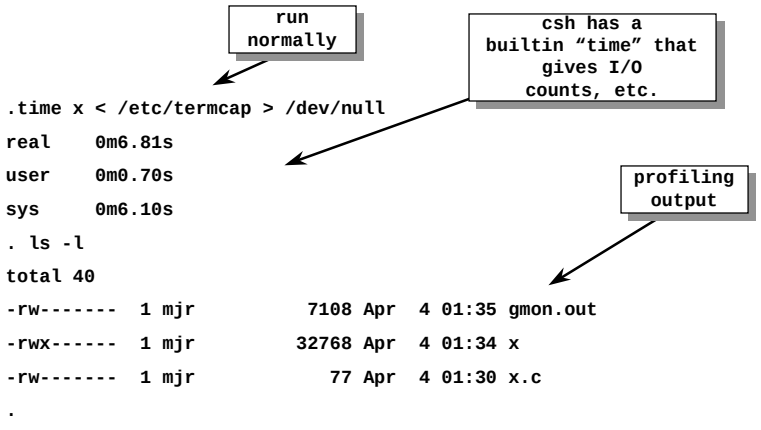
117

Measuring applications: gprof



118

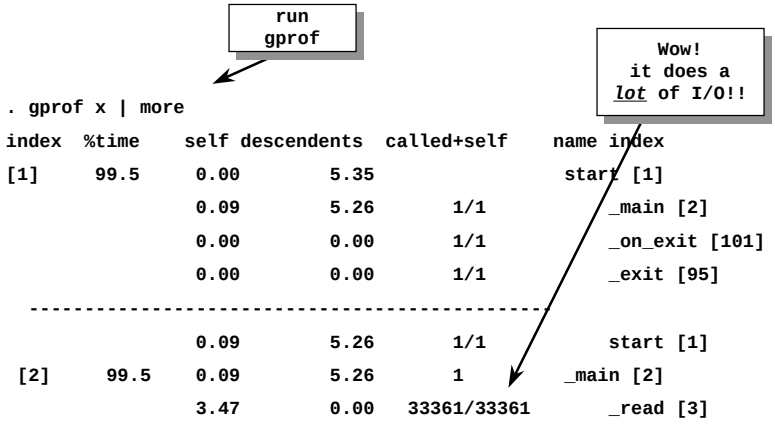
Measuring applications: gprof




```
.time x < /etc/termcap > /dev/null
real    0m6.81s
user    0m0.70s
sys     0m6.10s
. ls -l
total 40
-rw-------  1 mjr          7108 Apr  4 01:35 gmon.out
-rwx------  1 mjr         32768 Apr  4 01:34 x
-rw-------  1 mjr            77 Apr  4 01:30 x.c
.
```

119

Measuring applications: gprof

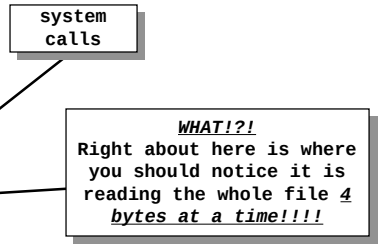



```
. gprof x | more
index  %time    self descendents  called+self    name index
[1]     99.5     0.00        5.35                 start [1]
                 0.09        5.26       1/1           _main [2]
                 0.00        0.00       1/1           _on_exit [101]
                 0.00        0.00       1/1           _exit [95]
  -----------------------------------------------
                 0.09        5.26       1/1           start [1]
 [2]     99.5    0.09        5.26       1         _main [2]
                 3.47        0.00   33361/33361       _read [3]
                 1.79        0.00   33360/33360       _write [4]
```

120

Measuring applications: trace

```
. trace x < /etc/termcap > /dev/null
open ("/dev/zero", 0, 02) = 3
mmap (0, 17036, 0x3, 0x80000002, 3, 0) = 0xf77fa000
close (3) = 0
getpagesize () = 4096
brk (0x9678) = 0
brk (0xa678) = 0
read (0, "# --", 4) = 4
write (1, "# --", 4) = 4
read (0, "----", 4) = 4
write (1, "----", 4) = 4
read (0, "----", 4) = 4
write (1, "----", 4) = 4
read (0, "----", 4) = 4
```



121

Topics

- Introductory concepts
- CPU systems
- Memory systems
- Disk systems
- Windows, graphics and databases
- Networks
- Benchmarks
- Tuning up applications
- FINAL EXAM

122

FINAL EXAM

123

FINAL EXAM: Question #1

124

FINAL EXAM: Question #2

125

FINAL EXAM: Question #3

126

FINAL EXAM: Question #4

127

FINAL EXAM: Question #5

128