Advanced Production Troubleshooting

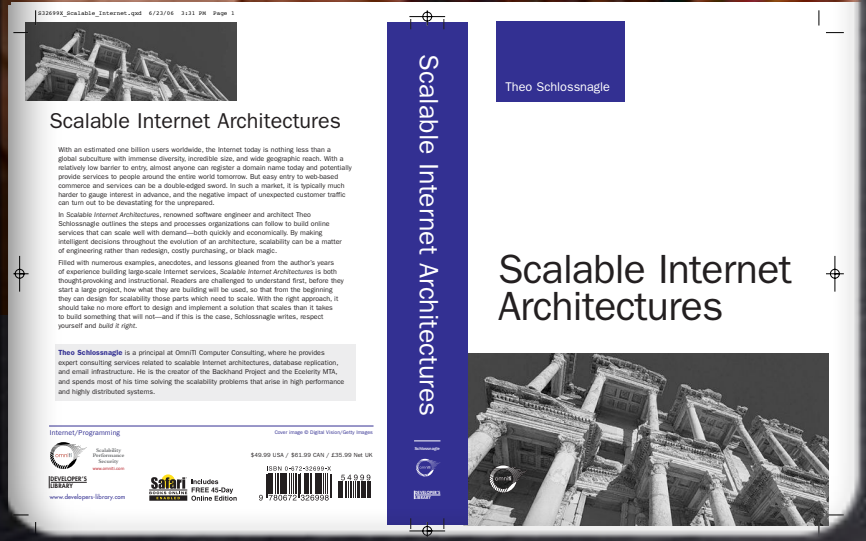
Theo Schlossnagle

Principal theo@omniti.com

OmniTI

omniti.com

Who is this guy?



- Principal @ OmniTI
- Open Source

  mod_backhand, spreadlogd, OpenSSH+SecurID, Daiquiri, Wackamole, libjlog, Spread, etc.
- Closed Source

  Ecelerity and EcCluster
- Author

  Scalable Internet Architectures

Production Troubleshooting

- What is “production?”
- What is troubleshooting?
- Why would we ever troubleshoot in production?
- Methods and techniques.

Production:
when it matters most

- your business depends on it
- your livelihood depends on it
- others depend on you fixing it
- it matters that you fix it and fix it now

Troubleshooting:
the @#$%* site is &*$%# down!

- It's 3am
- You're losing money
- You're in charge
- You didn't cause it
- You're responsible
- Everyone wants it fixed ***now***!

Why would I work in production?

Choose no life. Choose no career. Choose no family. Choose a fucking big computer, choose disk arrays the size of washing machines, modem racks, CD-ROM writers, and electrical coffee makers. Choose no sleep, high caffeine and mental insurance. Choose no friends. Choose black jeans and matching combat boots. Choose chairs for your office in a range of fucking fabrics. Choose SMTP and wondering why the fuck you are logged on on a Sunday morning. Choose sitting in that swivel chair looking at mind-numbing, spirit-crushing web sites, stuffing fucking junk food into your mouth. Choose rotting away at the end of it all, pishing your last in some miserable newsgroup, nothing more than an embarrassment to the selfish, fucked up lusers Gates spawned to replace the computer-literate.

Choose your future.

Choose to sysadmin.

Because

it's

broken

The Scope

- The scope doesn't get larger.
- Absolutely anything could be causing the problem.
- It doesn't matter if the cause is something for which you are directly responsible.

The rules of engagement

- There must be process.
  - Diagnosis has weak process.
    - Resolution has strong process.

Diagnosis

- If there was a strong process, there would be a plan for avoidance.
- Requires good puzzle-solving skills.
- Requires multi-dimensional attack strategies.
- You must be smart.
- You must think about **every** part of your architecture.
- 95% art, 5% science.

診断

Resolution

- Here's where science re-enters:
  - 5% art, 95% science
- The solution must be:
  - understood
  - accurate
  - stable
  - localized

解決

Glossary

- **Problem**: the specific, unambiguous issue.
- **Solution**: the exact process of fixing the problem.
- **Victim**: the entity experiencing the problem.
- **Witness**: the ability to see the problem.
- **Offender**: the entity causing the problem.

Methods & Techniques
some simple rules

Instrumenting applications
(after the fact)
is a good way to waste
precious time.

- System tools are good.
- Don’t lose sight of the problem.
- Let the problem drive your diagnosis.
- When it doubt, use brute force.

Things I can remember during a crisis

- I started as an SA.
  - I like SA tools.
  - Most problems can be identified on a systemic level.
- I don't want to muck with code to *find* a problem.
- I like passive analysis tools:
  - network packet dumpers: tcpdump/ethereal
  - system call tracers: strace/ktrace/truss
  - dynamic tracers: Dtrace

Before we embark. Protect.

- Troubleshooting sometimes involves hacking.
- Changes to production code or configuration are always dangerous.
- Not understanding the changes afterwards is simply irresponsible.

omniti

Do it yourself

- Keep track of changes you make
- Understand what is running in production:
  - now
  - yesterday
  - last week

Let something else track it.

- Don't leave room for human error.
- On a big central server:

```
; mkdir /data/backups
; svnadmin create --fs-type fsfs /data/projects/svn/systems
; cd /data/backups
; curl -O http://www.omniti.com/~jesus/projects/autorev.pl
; vi sysconflist
  [ add your hosts here ]
  [ add /data/backups/autorev.pl to cron ]
```

autorev.pl

```
# sample sysconflist


[crank-va-1 10.225.209.34]
rsync etc/ etc/ --exclude=cups --exclude=mail/statistics --exclude=ntp/drift
rsync opt/ opt/ --exclude=oracle --exclude=status.txt --exclude=openldap-data
rsync local/ usr/local/

[admin-va-1 10.225.209.68]
rsync etc/ etc/ --exclude=cups --exclude=mail/statistics --exclude=ntp/drift
rsync opt/apache/conf/ opt/apache/conf/

[www-va-1 10.225.209.71]
rsync etc/ etc/ --exclude=cups --exclude=mail/statistics --exclude=ntp/drift
rsync opt/apache/ opt/apache/
rsync var/apache/ var/apache/
```

What does autorev buy us?

- Everything we care about is in revision control.
  - It is automatically versioned; no human error.
    - If we have a host that TFTPs hardware configs:
      - We have change history on our routers, switches, firewalls, etc.

Yes. This is cool.

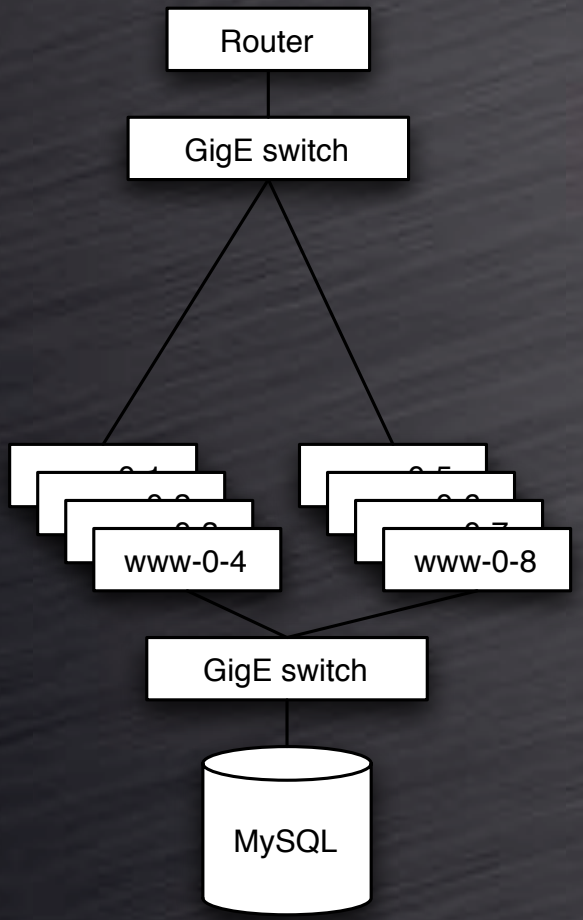
- We can watch our production architecture change.
  - vital binaries (web server, database, etc.)
  - configs (host, app, appliance)
  - custom application
- We can rollback to a known “live” state.
- Integrate with Trac and get an RSS feed of changes.

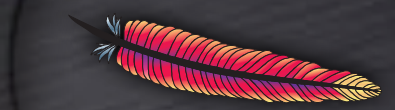
A simple first example.

Web application hangs
Speedy sometimes.
15-60 second pages loads other times

Architecture

Architecture

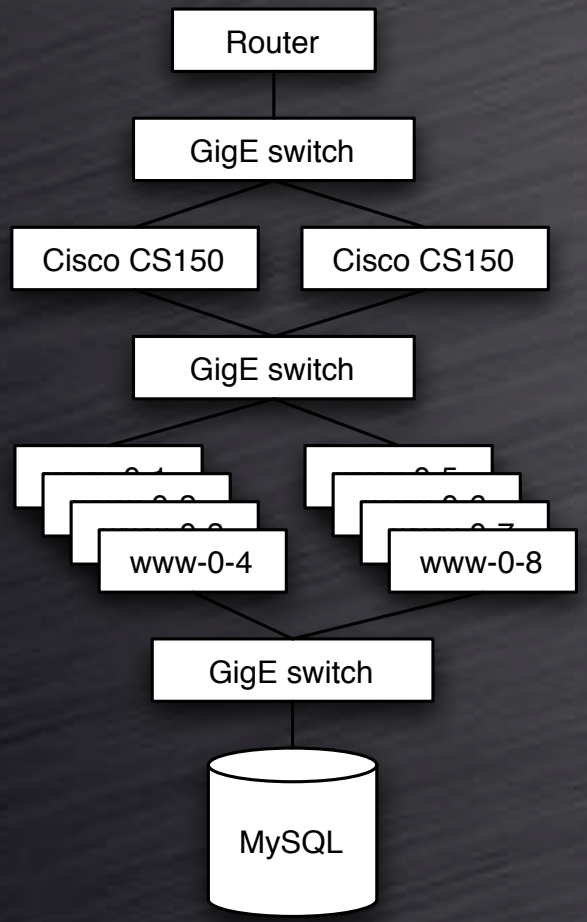




MySQL

Networking

Architecture



MySQL

Networking
load
balancing

omniti

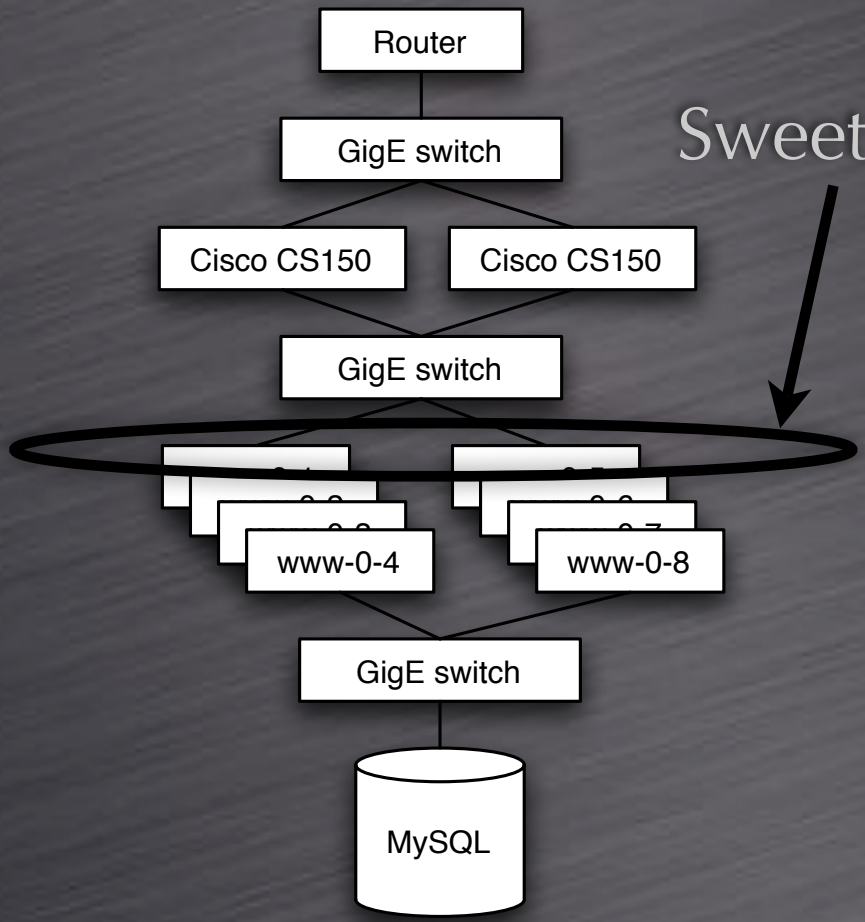
A Beginning

- Where to start looking?
  - This should be influenced by the tools you know best.
  - It's all about speed.
- I like to repeat the problem first.
  - Do what the victim does -- become the victim.
  - Tight, simple, repeatable tests are best.
- Lose your pride
  - Don't assume your stuff works.
  - Don't assume "Bob's" stuff is broken.
  - Don't assume anything.

Where to start...

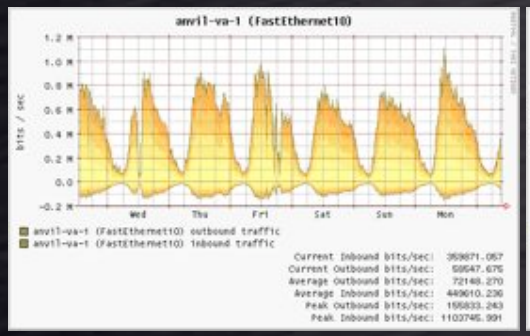
- Repeat the problem:
  - manual browser use
  - even better... a script
- The logical place to start:
  - where the victim touches the architecture
  - where you can most easily witness the problem.

Architecture

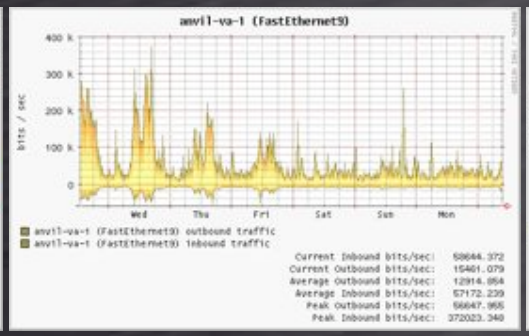


Which web server?

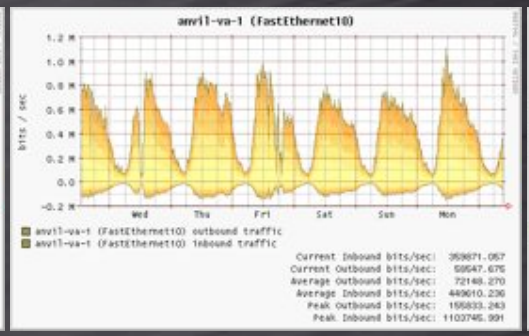
Don’t assume anything.



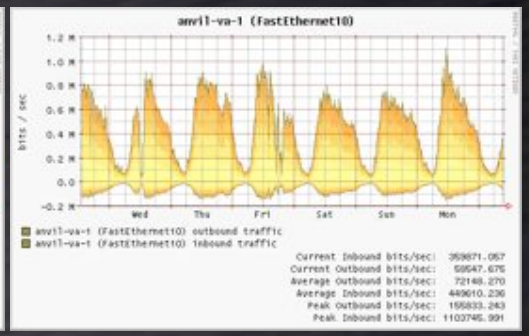
www-0-1



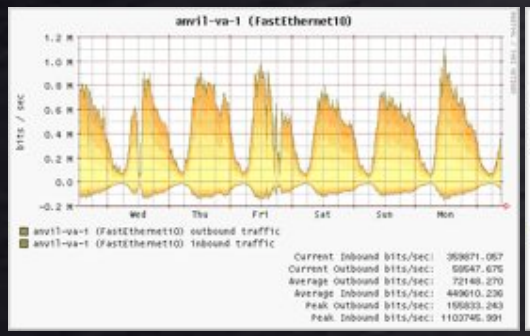
www-0-2



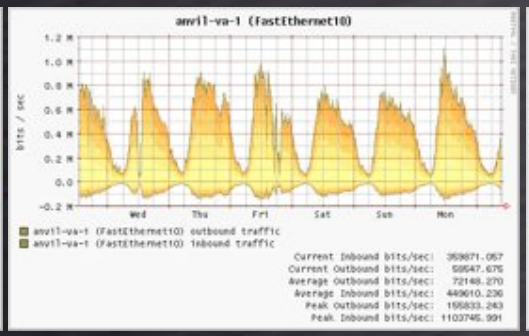
www-0-3



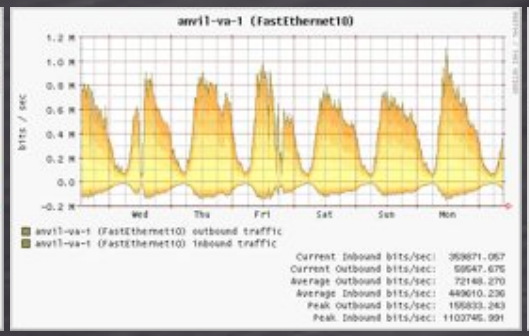
www-0-4



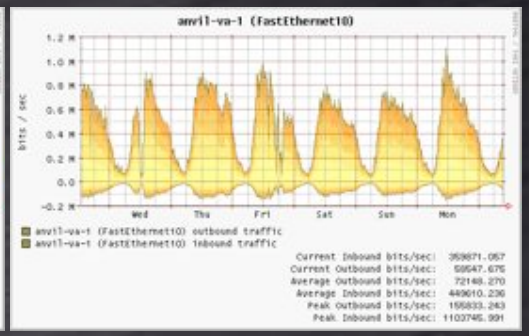
www-0-5



www-0-6



www-0-7



www-0-8

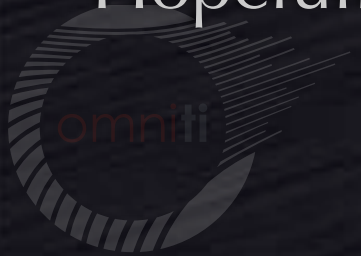
There is clearly something wrong with www-0-2

Approach 1: system tracing

```
; ps auxww | grep httpd
nobody     417  0.0  0.0 26904  344 ?   S     2005 149:34 /opt/apache_1.3.33/bin/httpd -DSSL
root       416  0.0  0.0 26952  120 ?   S     2005   5:53 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   19436  0.0  0.5 33212 6136 ?   S    16:40   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   20416  0.0  0.6 33292 6540 ?   S    17:30   0:01 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   20494  0.0  0.5 32572 5512 ?   S    17:34   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   20500  0.0  0.6 33312 6616 ?   S    17:35   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   20501  0.0  0.6 33224 6304 ?   S    17:35   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   23718  0.0  0.6 33068 6592 ?   S    20:23   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   23729  0.0  0.2 29396 2468 ?   S    20:24   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   24646  0.0  0.7 32832 7792 ?   S    21:13   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   25407  0.0  0.3 29368 3800 ?   S    21:54   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   25735  0.0  0.3 29260 3424 ?   S    22:11   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
nobody   26062  0.0  0.4 29356 4596 ?   S    22:29   0:00 /opt/apache_1.3.33/bin/httpd -DSSL
```

Which process?

Hopefully they all exhibit signs of the problem.



Approach 1: system tracing

```
; strace -p 20500
semop(4292610, 0x80fa734, 1)            = 0
select(18, [16 17], NULL, NULL, NULL)   = 1 (in [17])
accept(17, {sa_family=AF_INET, sin_port=htons(64868),
           sin_addr=inet_addr("66.249.65.15")}, [16]) = 4
semop(4292610, 0x80fa73a, 1)            = 0
rt_sigaction(SIGUSR1, {SIG_IGN}, {0x80b923d, [],
            SA_RESTORER|SA_INTERRUPT, 0x401c47c8}, 8) = 0
fcntl64(4, F_SETFD, FD_CLOEXEC)         = 0
getsockname(4, {sa_family=AF_INET, sin_port=htons(8012),
              sin_addr=inet_addr("192.168.209.71")}, [16]) = 0
setsockopt(4, SOL_TCP, TCP_NODELAY, [1], 4) = 0
read(4, "GET /IM/storedetail.html?store=B"..., 4096) = 258
rt_sigaction(SIGUSR1, {SIG_IGN}, {SIG_IGN}, 8) = 0
time(NULL)                              = 1149010483
gettimeofday({1149010483, 613252}, NULL) = 0
... lots and lots of output ...
read(4, "", 4096)                       = 0
time(NULL)                              = 1149011137
close(4)                                = 0
rt_sigaction(SIGUSR1, {0x80b923d, [], SA_RESTORER|SA_INTERRUPT, 0x401c47c8},
            {0x80b923d, [], SA_RESTORER|SA_INTERRUPT, 0x401c47c8}, 8) = 0
semop(4325378, 0x80fa734, 1
```

Brief lull

15 second lull

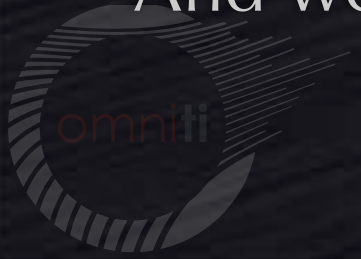
Approach 1: system tracing

```
; lsof -p 20500
...
httpd 22282 nobody  4u  IPv4  517989990  TCP 66.249.65.15:47451->www-va-1:http (ESTABLISHED)
...
httpd 22282 nobody 17u  IPv4  497473156  TCP *:80   (LISTEN)
...
```

File descriptor 4 is the connection to our client.

And we are stuck reading from it.

And we read nothing and then return to servicing others.



Guessing the problem

- Revisiting the “evil” lull:
  - It is exactly 15 seconds. Every time.
  - This can be confirmed with strace -ttt
  - What is so special about 15 seconds?
    - Our application?
    - Apache?
    - libc?
    - kernel?

My guess: Apache

Apache talks directly to the client and issued the “read” system call on which we are stuck.

```
; cd ~/src/apache_1.3.33/
; grep '#define.* 15$' `find . -name \*.h`
./src/include/httpd.h:#defi    DEFAULT_KEEPALIVE_TIMEOUT 15
./src/include/httpd.h:#define M_INVALID    15
./src/lib/expat-lite/xmltok.h:#define XML_TOK_PROLOG_S 15
./src/modules/standard/mod_rewrite.h:#define MAX_ENV_FLAGS 15
```

Recap

- Keep-alives were our problem:
  - Apache children were tied up waiting.
  - A limited number of children.
- When all children are used:
  - We have to wait until a child is free.
  - Up to 15 seconds.
  - Much much longer if a back queue exists.

This could have gone differently.

Approach 1a: system tracing

```
; strace -p 20500
semop(4292610, 0x80fa734, 1)            = 0
select(18, [16 17], NULL, NULL, NULL)   = 1 (in [17])
accept(17, {sa_family=AF_INET, sin_port=htons(64868),
           sin_addr=inet_addr("66.249.65.15")}, [16]) = 4
semop(4292610, 0x80fa73a, 1)            = 0
... lots and lots of output ...
time(NULL)                              = 1150834255
write(4, "HTTP/1.1 200 OK\r\nDate: Tue, 20 J"..., 195) = 195
select(8, [4], NULL, NULL, {0, 0})      = 0 (Timeout)
gettimeofday({1150834255, 873521}, NULL) = 0
writev(4, [{"554c\r\n", 6},
           {"\n@import \'/c/styles..., 212836},
           {"\r\n", 2}], 3) = 212844
select(5, [4], NULL, NULL, {0, 0})      = 0 (Timeout)
write(4, "0\r\n\r\n", 5)                = 5
time([1150834256])                      = 1150834256
gettimeofday({1150834256, 300626}, NULL) = 0
times({tms_utime=29, tms_stime=7, tms_cutime=0, tms_cstime=0}) = -448735555
shutdown(4, 1 /* send */)               = 0
select(5, [4], NULL, NULL, {0, 0})      = 1 (NULL, left {0, 0})
read(4, "", 512)                        = 0
close(4)                                = 0
...
```

Brief lull

3 second lull

What's up here?!

- Sending data to client gets "stuck"
  - writev() sticking is due to kernel buffers filling up.
- make the buffers bigger
  - Kernel buffer enlargement
  - SendBufferSize in Apache
- install a web accelerator.

Looking inward.

- We've looked outward (toward clients)
  - it has been quick and painless
- the same technique can work for inward problems
  - at least some of them

(same old) system tracing

```
; strace -p 20500
semop(4292610, 0x80fa734, 1)            = 0
select(18, [16 17], NULL, NULL, NULL)   = 1 (in [17])
accept(17, {sa_family=AF_INET, sin_port=htons(64868),
           sin_addr=inet_addr("66.249.65.15")}, [16]) = 4
semop(4292610, 0x80fa73a, 1)            = 0
... lots and lots of output ...
gettimeofday({1151148309, 837141}, NULL) = 0
time(NULL)                               = 1150834255
write(7, "\0\223\0\0\6\0"..., 147) = 147
read(7, "\0\374\0\0\6\0\0"..., 2064) = 252
gettimeofday({1151148309, 837882}, NULL) = 0
...
```

Brief lull

< 1 second lull

(same old) system tracing

```
; lsof -p 20500
...
httpd 22282 nobody  4u  IPv4  517989990  TCP 66.249.65.15:47451->www-va-1:http (ESTABLISHED)
...
httpd 22282 nobody  7u  IPv4  517989990  TCP www-va-1.int:47451->dbhost:5432 (ESTABLISHED)
...
httpd 22282 nobody 17u  IPv4  497473156  TCP *:80   (LISTEN)
...
```

File descriptor 7 is the connection to our database.

And we are stuck reading from it.

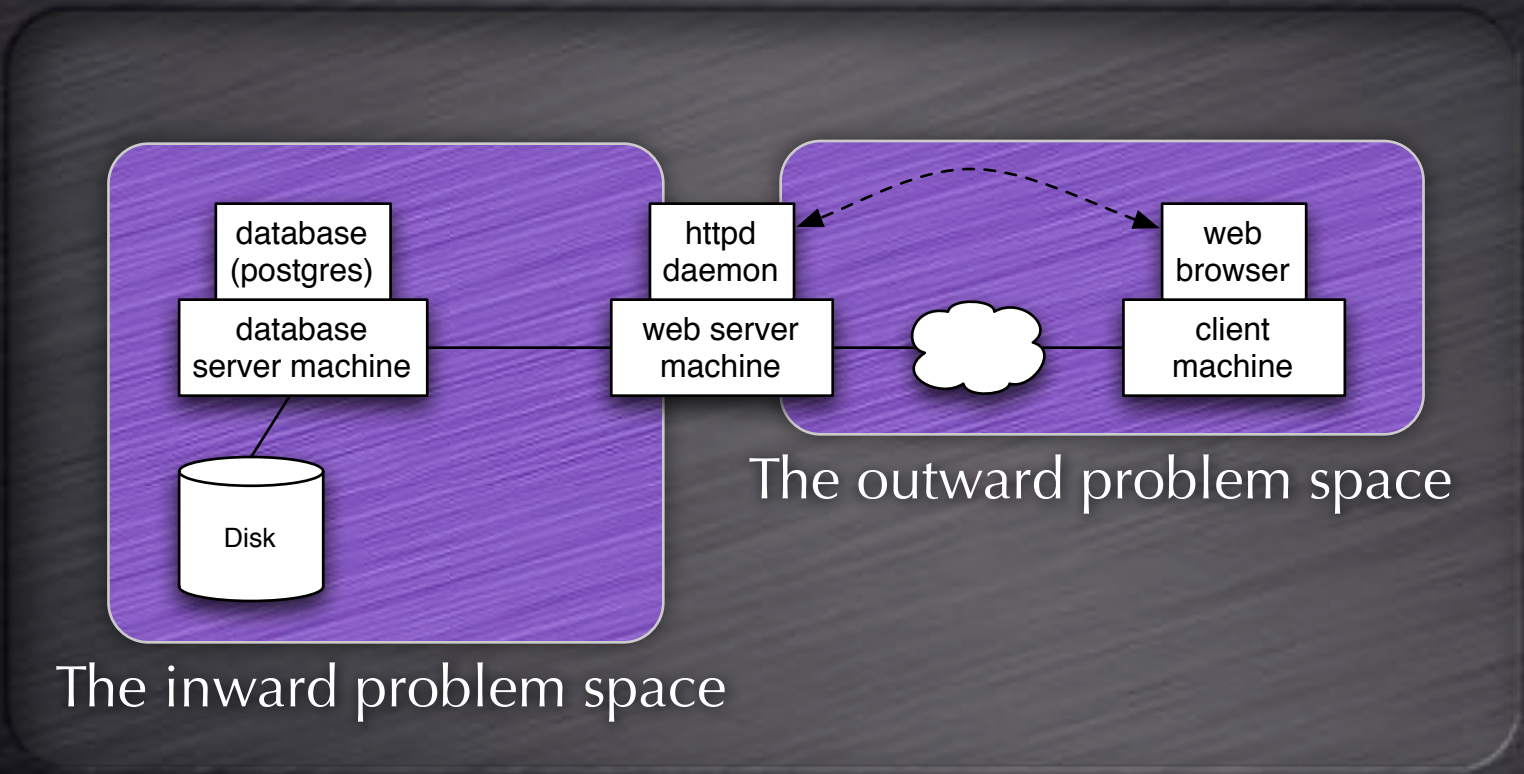
Hmm... that's a query, with a slow(ish) response.

Not where we want to be

- Did we successfully locate the problem?

  No...
- We know it behind the web application.
  - Not necessarily behind the web server.

An over-simplified web transaction



Perspective from analysis on a web server

Jumping to the next step

- In the outward case,
  jumping outside (to the client) is not an option.
  - Jumping to a load balancer, switch or router is.
  - The tools there are not so good.
  - Best to stay on the web server.
- In the inward case,
  jumping inside (to the database) is an option.
  - The web server is delayed reading from the DB
  - Confirm the DB is indeed working.

Looking at the system (database)

- What's wrong with the system?
  - Rephrased: what looks wrong?
  - Wrong and right are relative with respect to system returning correct results.
  - What looks "different" is a better question.
  - That requires a basis of comparison.

Looking at processes (database)

- So, what's running?
  - Clearly, a database.
  - The site works (albeit slow),
    so the database as functioning correctly.
  - It's not performing well.
  - Which queries are running slow and why?
  - ***Don't assuming anything.***

Database: on top

```
load averages: 24.89, 20.66, 20.66                                  15:01:14
116 processes:  109 sleeping, 3 zombie, 4 on cpu
CPU states: 0.9% idle, 26.0% user,  8.1% kernel, 64.0% iowait,  0.0% swap
Memory: 16G real, 13G free, 1984M swap in use, 42G swap free

   PID USERNAME LWP PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
 20768 jesus      1  49    0 1564K 1176K cpu/1    0:00 23.03% sirtoppemhat
 20508 postgres   1  21    0    0K    0K cpu/2    2:03  0.60% postgres
 18562 postgres   1  48    0    0K    0K sleep    1:16  0.75% postgres
  8240 postgres   1  59    0    0K    0K sleep   54:33  0.10% postgres
  7544 postgres   1  54    0    0K    0K sleep   12:12  0.10% postgres
  4532 postgres   1  33    0    0K    0K sleep   13:09  0.10% postgres
  5672 postgres   1  35    0    0K    0K sleep    9:56  0.10% postgres
 23467 postgres   1  33    0    0K    0K sleep    2:01  0.10% postgres
 11232 postgres   1  50    0    0K    0K sleep    1:27  0.10% postgres
 11332 postgres   1  40    0    0K    0K sleep   15:14  0.10% postgres
 12546 postgres   1  38    0    0K    0K sleep   16:37  0.10% postgres
  7656 postgres   1  35    0    0K    0K sleep    7:08  0.10% postgres
  2384 postgres   1  35    0    0K    0K sleep    8:41  0.10% postgres
  8240 postgres   1  41    0    0K    0K sleep    8:17  0.10% postgres
  3324 postgres   1  49    0    0K    0K sleep   12:20  0.10% postgres
  1912 postgres   1  47    0    0K    0K sleep    7:38  0.00% postgres
  8241 postgres   1  51    0    0K    0K sleep    2:25  0.00% postgres
  1312 root      26  59    0 3968K 3088K sleep    2:16  0.00% nscd
```

Database: on prstat

```
  PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/NLWP
21755 postgres  37 0.9 0.1 0.0 0.0 0.0  58 3.6  1K  1K  1K   0 postgres/1
 8241 postgres 0.5 1.3 0.0 0.0 0.0 0.0  97 1.5  2K 240 14K   0 postgres/1
 8242 postgres 0.7 1.1 0.0 0.0 0.0 0.0  97 1.3  2K  87 12K   0 postgres/1
21821 postgres 0.4 0.4 0.0 0.0 0.0 0.0  99 0.1 134  80 620   0 postgres/1
21825 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 125  80 573   0 postgres/1
21805 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 132  55 600   0 postgres/1
21857 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 124  68 552   0 postgres/1
21829 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 129  62 625   0 postgres/1
21801 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 126  93 574   0 postgres/1
21841 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 113  75 513   0 postgres/1
21797 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.2 114  53 503   0 postgres/1
21837 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.2 111  63 510   0 postgres/1
21809 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 126  21 538   0 postgres/1
21833 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.1 118  47 521   0 postgres/1
21853 postgres 0.3 0.3 0.0 0.0 0.0 0.0  99 0.2 110  71 464   0 postgres/1
21849 postgres 0.2 0.3 0.0 0.0 0.0 0.0  99 0.1 102  53 430   0 postgres/1
21845 postgres 0.2 0.3 0.0 0.0 0.0 0.0  99 0.1  99  64 421   0 postgres/1
 NPROC USERNAME  SIZE   RSS MEMORY      TIME  CPU
    29 postgres   18G   17G    97%   1:07:25  13%
     1 nobody   9720K 8400K   0.0%   0:33:53 0.0%
    37 root      132M   77M   0.4%   0:19:37 0.0%
     4 daemon     10M 6052K   0.0%   0:03:49 0.0%

Total: 116 processes, 489 lwps, load averages: 24.16, 20.16, 20.73
```

Database: on iostat

```
                    extended device statistics
   r/s    w/s   kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
  80.0  233.0    0.0 49627.6  0.0 46.8    0.2  201.0   5 100 c3t60003930000016A06d0
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d2
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d1
 410.0   68.0    0.0 56406.0  0.0 26.1    0.3  383.9   2 100 c3t60003930000016A06d0
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d2
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d1
 490.0   48.0    0.0 41290.5  0.0 30.0    0.4  624.1   2 100 c3t60003930000016A06d0
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d2
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d1
 521.0   42.0    0.0 38846.0  0.0 30.7    0.3  730.4   1 100 c3t60003930000016A06d0
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d2
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d1
 110.0  136.0    0.0 44998.6  0.0 33.0    0.2  242.7   3 100 c3t60003930000016A06d0
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d2
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d1
 400.0   64.0    8.0 38333.4  0.0 30.9    0.2  475.8   2 100 c3t60003930000016A06d0
   0.0    0.0    0.0     0.0  0.0  0.0    0.0    0.0   0   0 c3t60003930000016A06d2
   0.0    2.0    0.0     1.5  0.0  0.1    0.1   50.3   0  10 c3t60003930000016A06d1
```

@#$%! Those are some high service times!

So, we know the situation is bad

- Knowing the situation is bad is good.
  - We already knew that.
  - We have made progress, we know disk service latency on the database is a likely cause.
  - ~~Which queries are running slowly?~~

This is a bad question.
We know the database is slow due to slowed disk access.
But, we don't know that the database is the *cause*.

Which is the offending process?

- It is pretty hard to tell this on Linux.
  - my technique usually involvess peeking at top
  - taking a guess
  - stracing
  - repeat.

How can we be more efficient?

- My database server runs Solaris 10
  - I have DTrace
  - I do not suffer from inadequacies.
  - Next step... world domination.

Database: on DTrace

```
; dtrace -n 'io:::start { @[pid,execname] = sum(args[0]->b_bcount); }'
dtrace: description 'io:::start ' matched 6 probes
^C

     3896  postgres                                                352256
     3876  postgres                                                393216
     3884  postgres                                                393216
     3868  postgres                                                413696
     3880  postgres                                                413696
     3872  postgres                                                425984
     3864  postgres                                                438272
     3860  postgres                                                524288
     3848  postgres                                                544768
     3856  postgres                                                569344
     3852  postgres                                                573440
     3844  postgres                                                589824
     3836  postgres                                                716800
     3840  postgres                                                724992
     3832  postgres                                                851968
        3  fsflush                                                 861696
     3828  postgres                                                872448
     4589  tar                                                   75634228
```

This could have gone differently.

Database: on DTrace

```
; dtrace -n 'io:::start { @[pid,execname] = sum(args[0]->b_bcount); }'
dtrace: description 'io:::start ' matched 6 probes
^C

     3896  postgres                                                352256
     3876  postgres                                                393216
     3884  postgres                                                393216
     3868  postgres                                                413696
     3880  postgres                                                413696
     3872  postgres                                                425984
     3864  postgres                                                438272
     3860  postgres                                                524288
     3848  postgres                                                544768
     3856  postgres                                                569344
     3852  postgres                                                573440
     3844  postgres                                                589824
     3836  postgres                                                716800
     3840  postgres                                                724992
     3832  postgres                                                851968
        3  fsflush                                                 861696
     3828  postgres                                               4372448
     3823  postgres                                               6534428
     4589  postgres                                              12634228
```

Database: the Offender

```
; echo "SELECT now() - query_start as duration, current_query
          FROM pg_stat_activity
         WHERE procpid = 4589" | psql pgods

 duration        | current_query
-----------------+-----------------------------------------------------
 00:01:23.345221 | UPDATE users SET emailaddress = lower(emailaddress);
```

- It should be rather obvious now.
- Someone issued an enormous update.
- It is inducing enormous disk I/O
- Slowing everything else in the system.

Why DTrace is so cool.

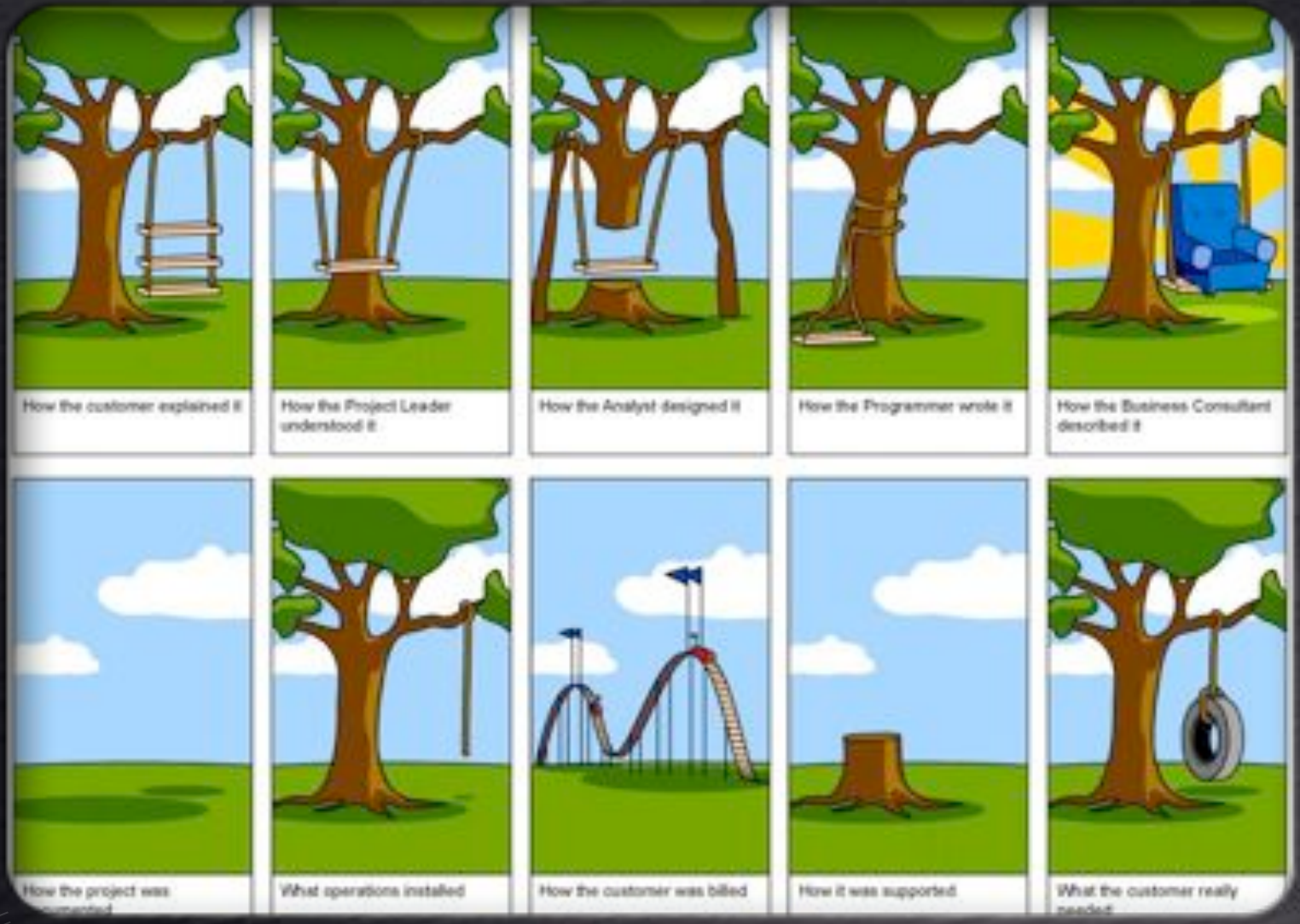
```
; dtrace -q -n '
postgres*:::report-activity
{
  self->query = copyinstr(arg0);
  self->ok=1;
}
io:::start
/self->ok/
{
  @[self->query,
    args[0]->b_flags & B_READ ? "read" : "write",
    args[1]->dev_statname] = sum(args[0]->b_bcount);
}'
dtrace: description 'postgres*:::report-activity' matched 14 probes
^C

select count(1) from c2w_ods.tblusers where zipcode between 10000 and 11000;
    read sd1 16384
select division, sum(amount), avg(amount) from ods.billings where txn_timestamp
between '2006-01-01 00:00:00' and '2006-04-01 00:00:00' group by division;
    read sd2 71647232
```

The Importance of Historical Data

- Trend Analysis.
- Provides a control for experiments.
- Provides a foundation for conjectures.
- At a *bare* minimum
  - bandwidth
  - load
  - disk I/O and service times
  - memory usage

Anecdotes



Questions?

Credits

- OmniTI, Inc. - best place to work ever.
- Sun and the DTrace team.
- My wife and children.