PostgreSQL

Looking under the hood with Solaris

OmniTI / Presentation / Theo Schlossnagle

PostgreSQL is Awesome

- Fast.
- Extensible.
- Tablespaces.
- Robust data types.
- Partitioning (albeit fake).
- Partial and functional indexes.
- Extremely supportive community.
- Extremely compliant with database standards.

PostgreSQL is Lacking

- No upgrades (AYFKM).
- pg_dump is too intrusive.
- Poor system-level instrumentation.
- Poor methods to determine specific contention.
- It relies on the operating system's filesystem cache.
  (which make PostgreSQL inconsistent across it's supported OS base)

Enter Solaris

- Solaris is a UNIX from Sun Microsystems.
- How is it different than other UNIX and UNIX-like systems?
  - Mostly it isn’t different (hence the term UNIX)
  - It does have extremely strong ABI backward compatibility.
  - It’s stable and works well on ***large*** machines.
- Solaris 10 shakes things up a bit:
  - DTrace
  - ZFS
  - Zones

Solaris / ZFS

- ZFS: Zettaback Filesystem.
  - 2^{64} snapshots, 2^{48} files/directory, 2^{64} bytes/filesystem, 2^{78} (256 ZiB) bytes in a pool, 2^{64} devices/pool, 2^{64} pools/system
- Extremely cheap differential backups.
  - I have a 5 TB database, I need a backup!
- No rollback in your database? What is this? MySQL?
- No rollback in your filesystem?
  - ZFS has snapshots, rollback, clone and promote.
  - OMG! Life altering features.
- Caveat: ZFS is slower than alternatives, by about 10% with tuning.

Solaris / Zones

- Zones: Virtual Environments.
- Shared kernel.
- Can share filesystems.
- Segregated processes and privileges.
- No big deal for databases, right?

But Wait!

Solaris / ZFS + Zones = Magic Juju

https://labs.omniti.com/trac/pgsoltools/browser/trunk/pitr_clone/clonedb_startclone.sh

- ZFS snapshot, clone, delegate to zone, boot and run.
- When done, halt zone, destroy clone.
- We get a point-in-time copy of our entire PostgreSQL database:
    - read-write,
    - low disk-space requirements,
    - NO LOCKS! Welcome back pg_dump, you don't suck anymore.
    - Fast snapshot to usable copy time:
        - On our 20 GB database: 1 minute.
        - On our 1.2 TB database: 2 minutes.

ZFS: how I saved my soul.

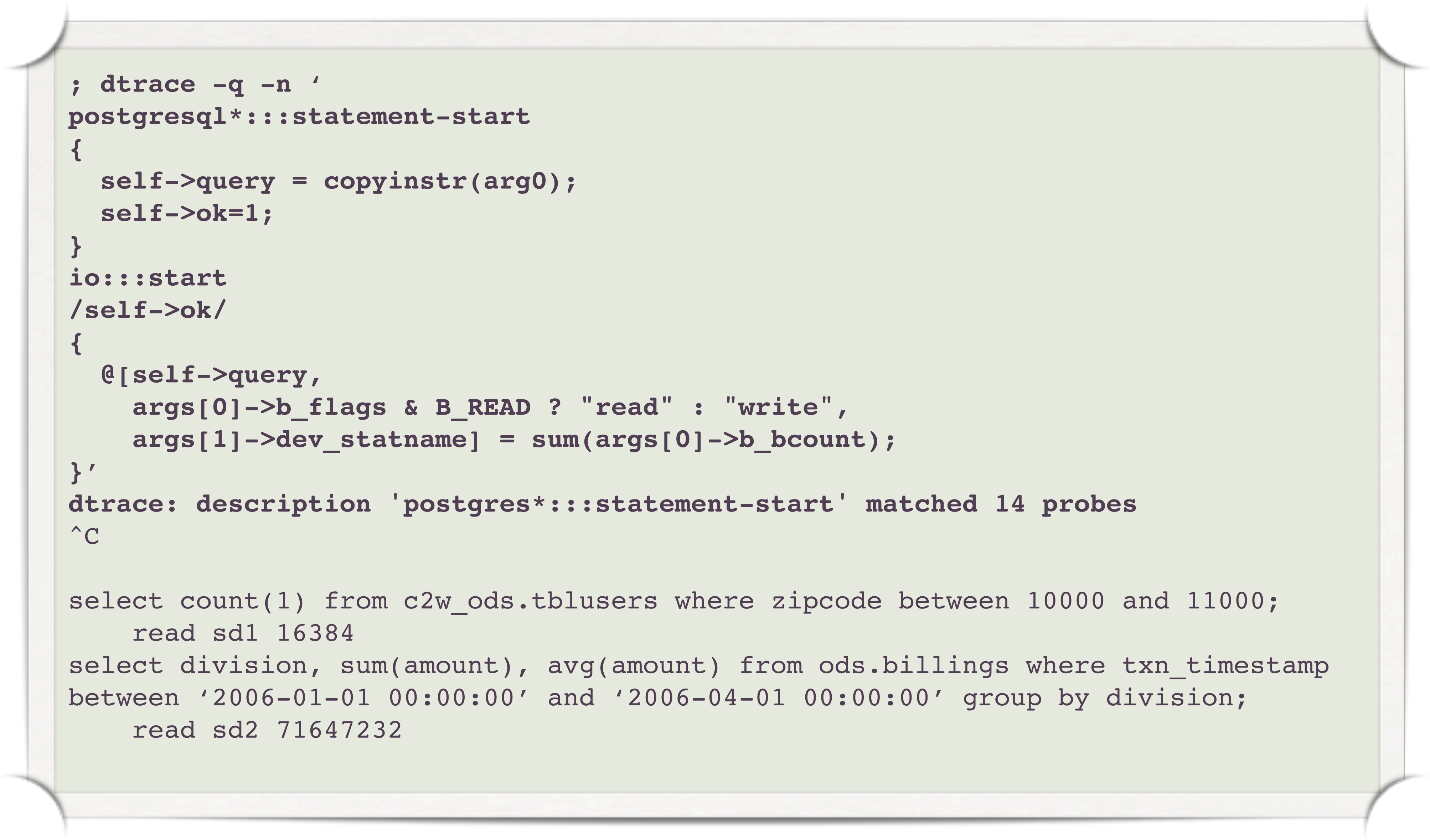
- Database crash. Bad. 1.2 TB of data... busted.
  The reason Robert Treat looks a bit older than he should.
- xlogs corrupted. catalog indexes corrupted.
- Fault? PostgreSQL bug? Bad memory? Who knows?
- Trial & error on a 1.2 TB data set can be a cruel experience.
  - In real-life, most recovery actions are destructive actions.
  - PostgreSQL is no different.
- Rollback to last checkpoint (ZFS), hack postgres code, try, fail, repeat.

Let DTrace open your eyes

- DTrace: Dynamic Tracing
- Allow you to dynamically instrument “stuff” in the system:
  - system calls (like strace/truss/ktrace).
  - process/scheduler activity (on/off cpu, semaphores, conditions).
  - see signals sent and received.
  - trace kernel functions, networking.
  - watch I/O down to the disk.
  - user-space processes, each function… *each machine instruction!*
  - Add probes into apps where it makes sense to you.

Can you see what I see?

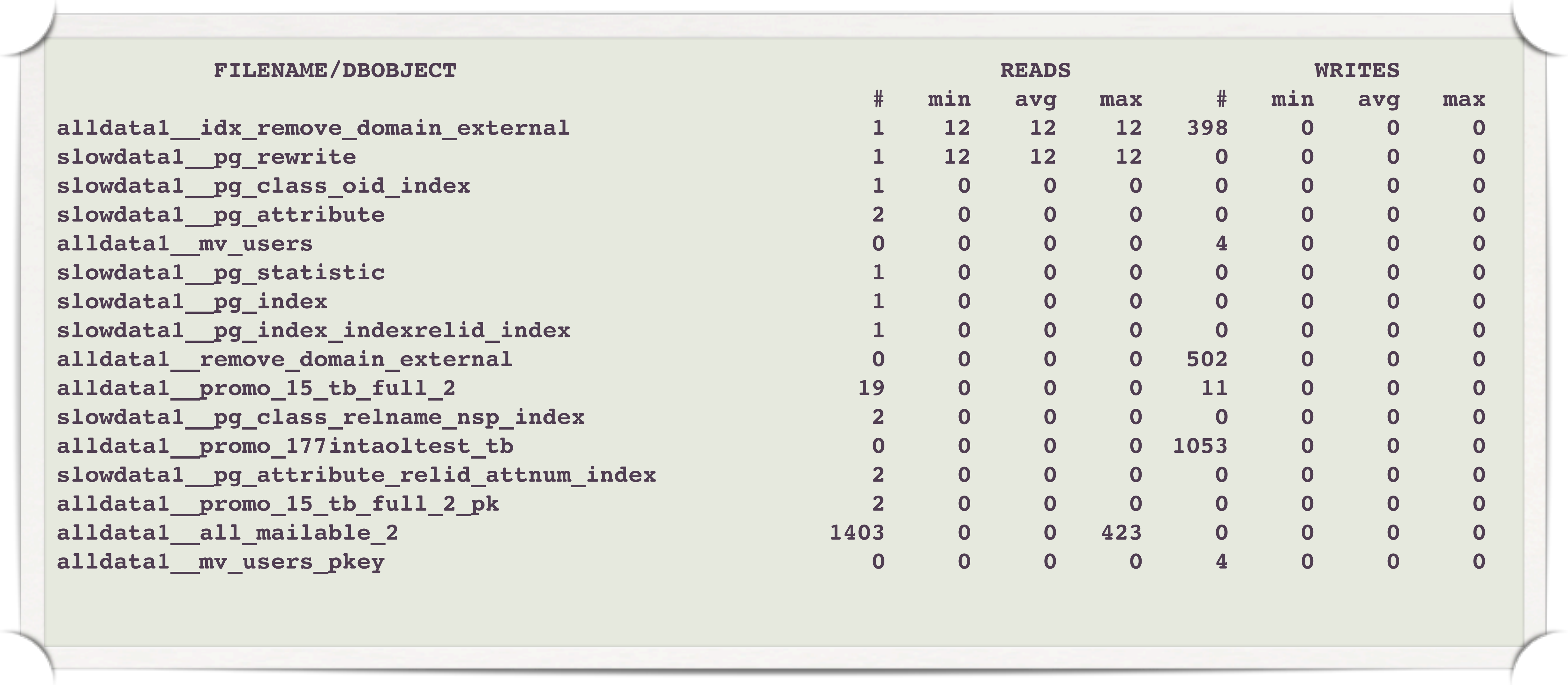
- There is EXPLAIN… when that isn’t enough…
- There is EXPLAIN ANALYZE… when that isn’t enough.
- There is DTrace.



```
; dtrace -q -n ‘
postgresql*:::statement-start
{
  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*:::statement-start' 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
```

- https://labs.omniti.com/trac/pgsoltools
  - Where we stick out PostgreSQL on Solaris goodies...
  - like pg_file_stress



```
          FILENAME/DBOBJECT                                        READS                      WRITES
                                                      #    min    avg    max      #    min    avg    max
alldata1__idx_remove_domain_external                  1     12     12     12    398      0      0      0
slowdata1__pg_rewrite                                 1     12     12     12      0      0      0      0
slowdata1__pg_class_oid_index                         1      0      0      0      0      0      0      0
slowdata1__pg_attribute                               2      0      0      0      0      0      0      0
alldata1__mv_users                                    0      0      0      0      4      0      0      0
slowdata1__pg_statistic                               1      0      0      0      0      0      0      0
slowdata1__pg_index                                   1      0      0      0      0      0      0      0
slowdata1__pg_index_indexrelid_index                  1      0      0      0      0      0      0      0
alldata1__remove_domain_external                      0      0      0      0    502      0      0      0
alldata1__promo_15_tb_full_2                         19      0      0      0     11      0      0      0
slowdata1__pg_class_relname_nsp_index                 2      0      0      0      0      0      0      0
alldata1__promo_177intaoltest_tb                      0      0      0      0   1053      0      0      0
slowdata1__pg_attribute_relid_attnum_index            2      0      0      0      0      0      0      0
alldata1__promo_15_tb_full_2_pk                       2      0      0      0      0      0      0      0
alldata1__all_mailable_2                           1403      0      0    423      0      0      0      0
alldata1__mv_users_pkey                               0      0      0      0      4      0      0      0
```

Thank you for listening.

Looking under PostgreSQL’s hood with Solaris.

OmniTI / Presentation