Performance Whack-a-Mole 2009

PGX
POSTGRESQL
EXPERTS, INC.

“The database is so fast. I don't know if we'll ever max it out.”

-- Not Your Client, Inc.

PGX
POSTGRESQL
EXPERTS, INC.

“My database is slow.”

-- Every Single Support Client LLC

Part 1:
The Rules

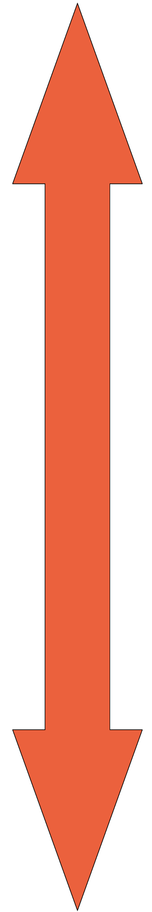
PGX
POSTGRESQL
EXPERTS, INC.

The Layer Cake

PGX
POSTGRESQL
EXPERTS, INC.

The Layer Cake

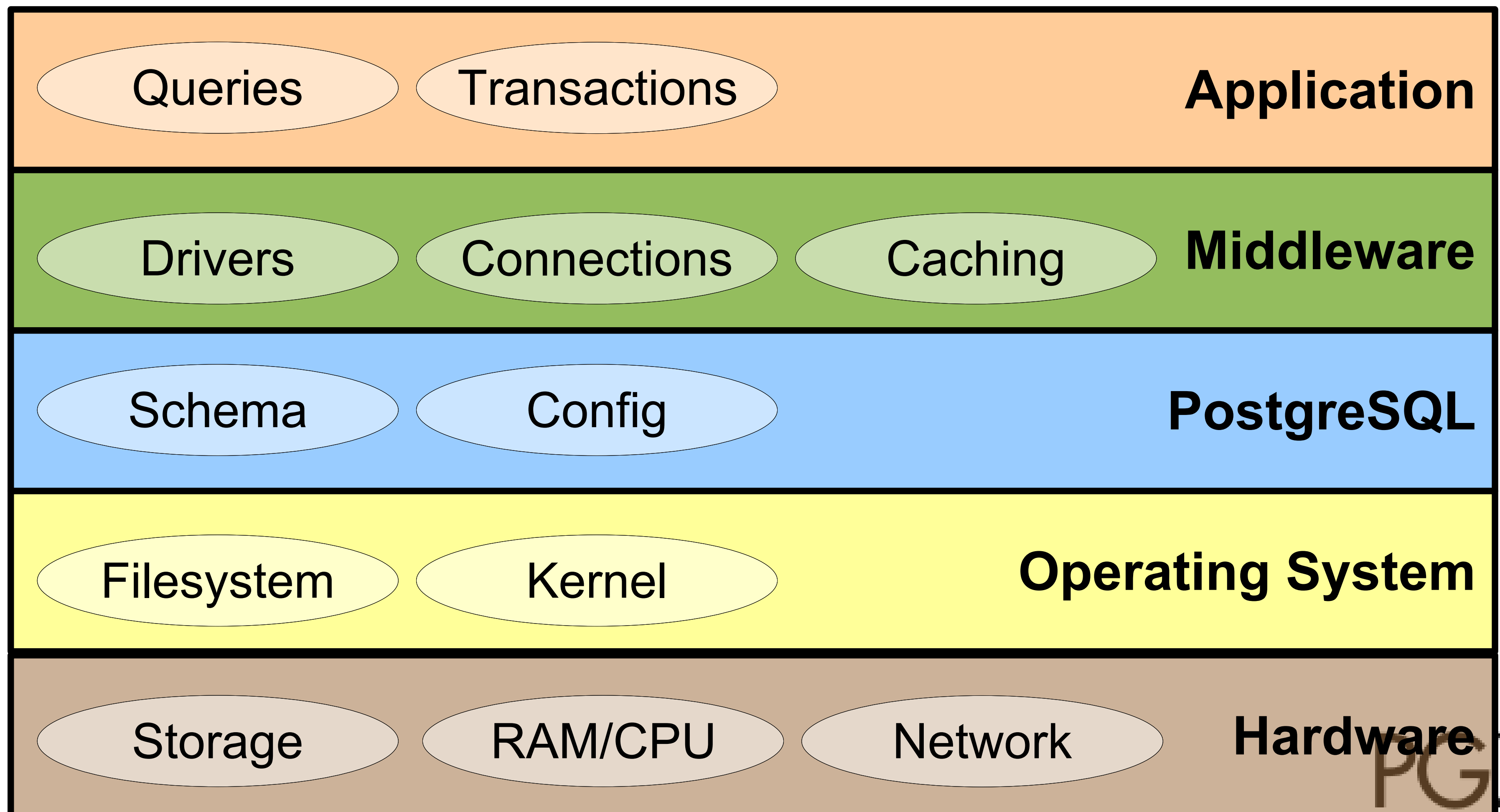
User



Silicon

PGX
POSTGRESQL
EXPERTS, INC.

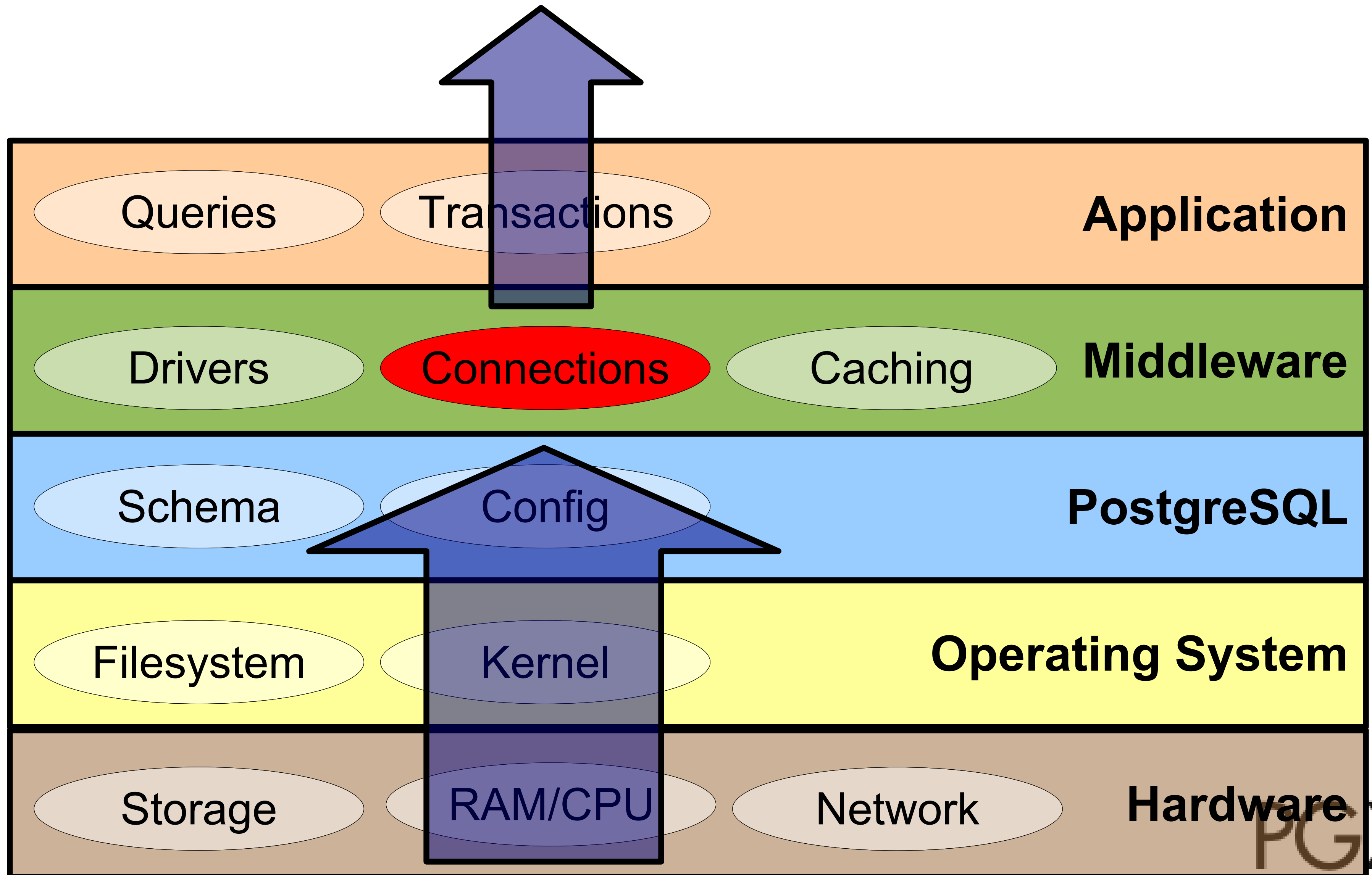
The Layer Cake



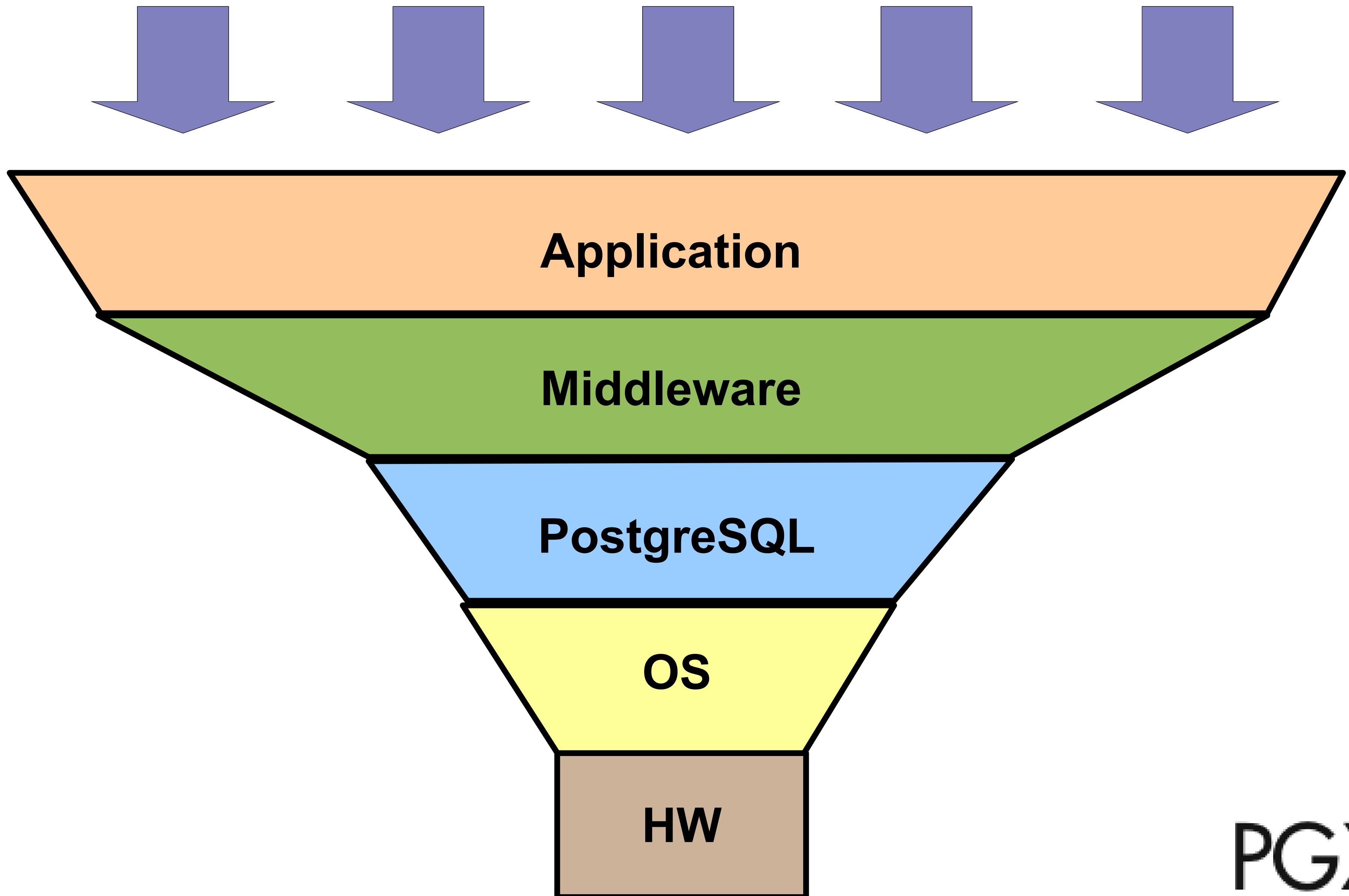
The Layer Cake

PGX
POSTGRESQL
EXPERTS, INC.

The Layer Cake



The Funnel



PGX
POSTGRESQL EXPERTS, INC.

Rules of Whack-a-Mole

1. Most “database performance problems”, or *Moles*, are not actually *database* performance problems.

PGX
POSTGRESQL
EXPERTS, INC.

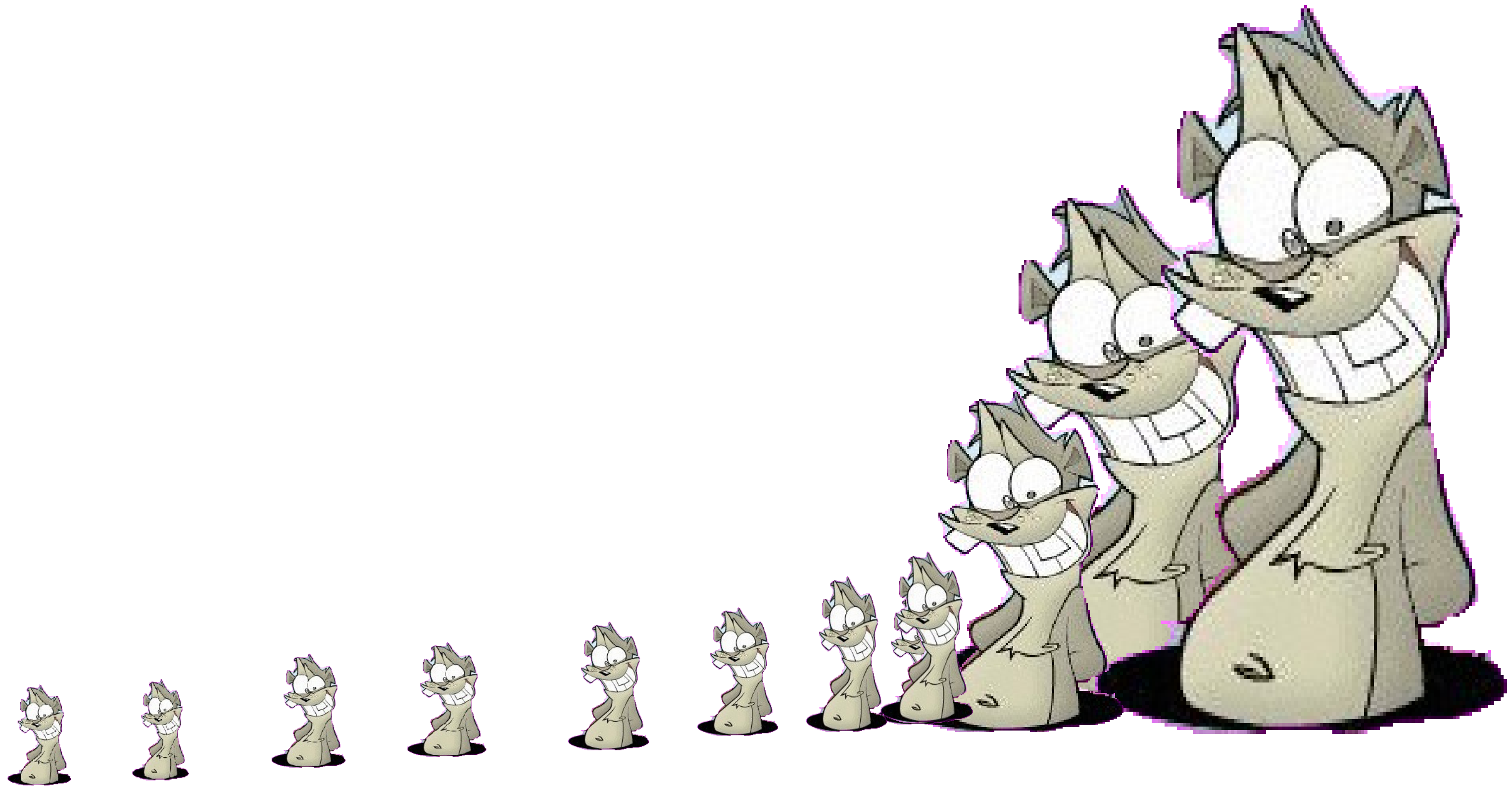
The Hockey Stick

PGX POSTGRESQL EXPERTS, INC

*the Ottawa Senators trademark and logo are property of the Ottawa Senators

The Hockey Stick

Effect on Performance



Ranked Issues

PGX
POSTGRESQL
EXPERTS, INC.

The Hockey Stick

PGX
POSTGRESQL
EXPERTS, INC.

Rules of Whack-a-Mole

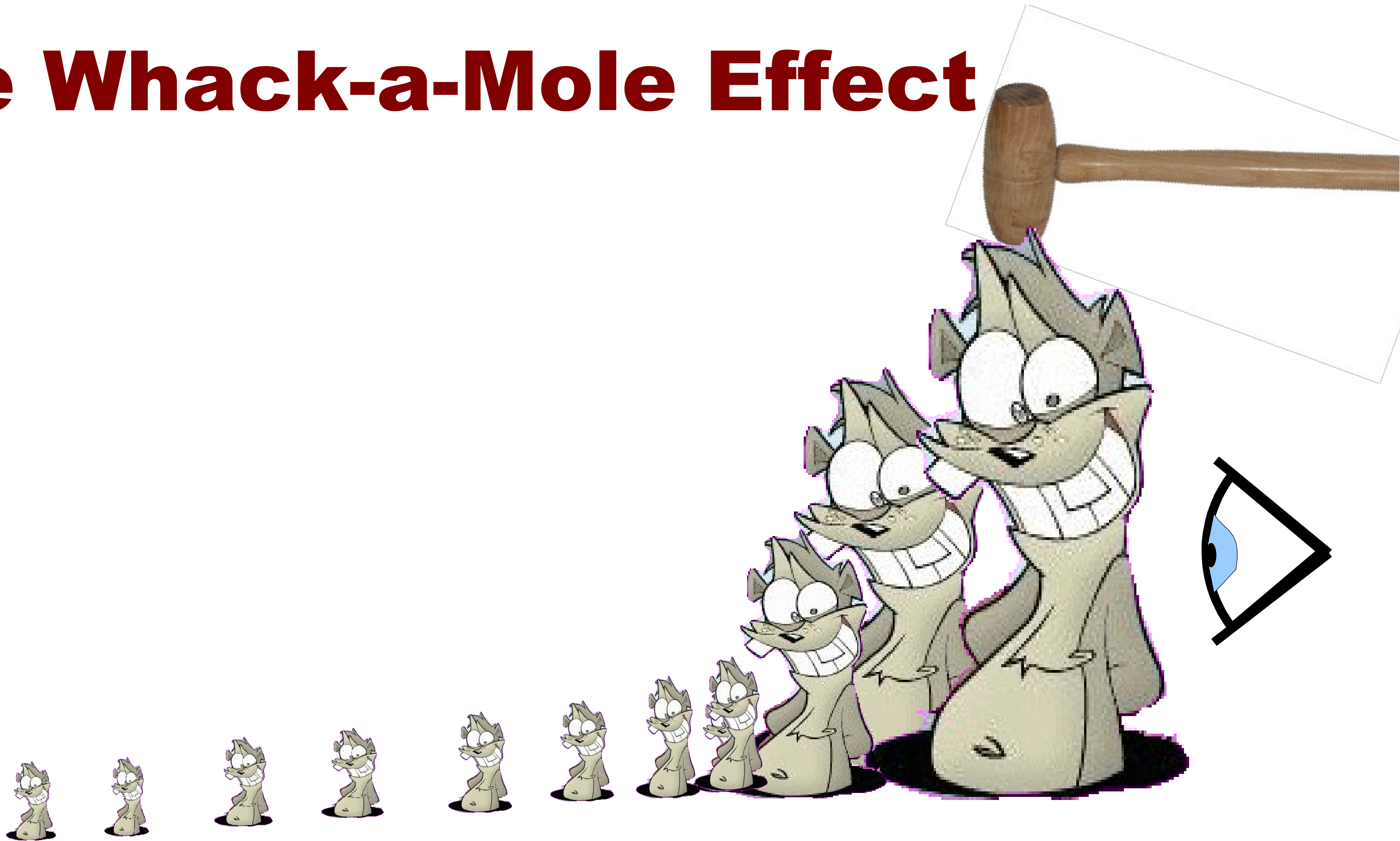
1. Most "database performance problems", or Moles, are not actually *database* performance problems.
2. Less than 10% of Moles cause 90% of performance degradation.

Rules of Whack-a-Mole

1. Most “database performance problems”, or Moles, are not actually *database* performance problems.
2. Less than 10% of Moles cause 90% of performance degradation.
   - corollary: we don't care about the other 90% of Moles

PGX
POSTGRESQL
EXPERTS, INC.

The Whack-a-Mole Effect

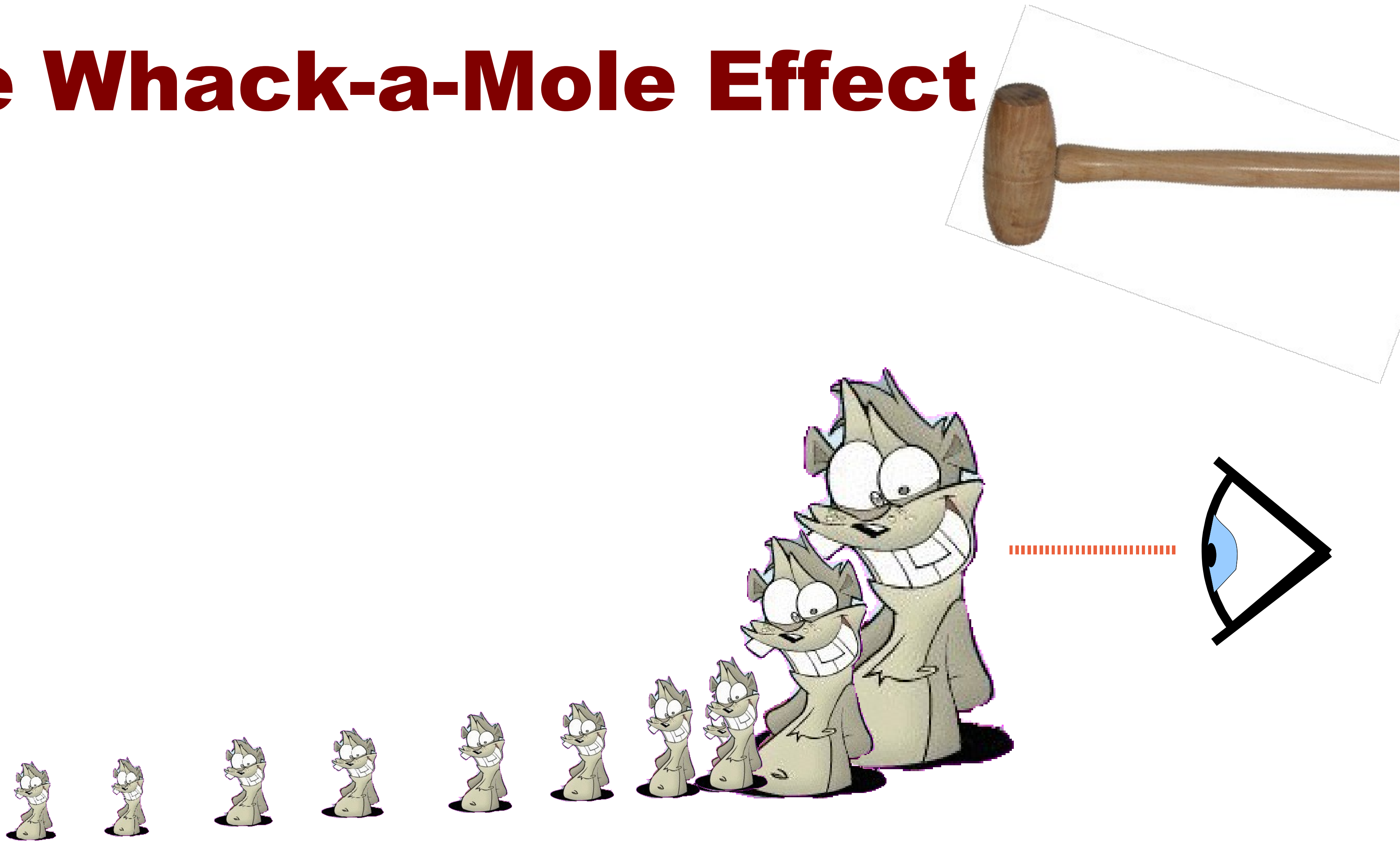


Effect on Performance

Ranked Issues

PGX
POSTGRESQL
EXPERTS, INC.

The Whack-a-Mole Effect



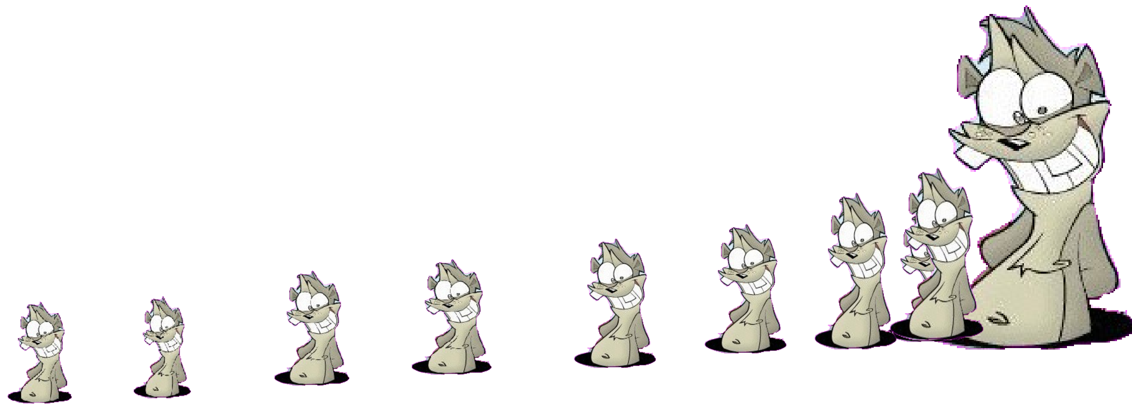
Effect on Performance

Ranked Issues

PGX
POSTGRESQL
EXPERTS, INC.

The Whack-a-Mole Effect

Effect on Performance



Ranked Issues

PGX
POSTGRESQL
EXPERTS, INC.

Rules of Whack-a-Mole

1. Most “database performance problems”, or Moles, are not actually *database* performance problems.
2. Less than 10% of Moles cause 90% of performance degradation.
   - corollary: we don't care about the other 90% of Moles
3. At any time, it is usually only possible to observe and troubleshoot, or *Whack*, the “largest” Mole.

PGX
POSTGRESQL
EXPERTS, INC.

What Color Is My Application?

W ▶ Web Application (Web)

O ▶ Online Transaction Processing (OLTP)

D ▶ Data Warehousing (DW)

What Color Is My Application?

- W ▶ Web Application (Web)
  - DB smaller than RAM
  - 90% or more simple queries
- O ▶ Online Transaction Processing (OLTP)
- D ▶ Data Warehousing (DW)

PGX
POSTGRESQL
EXPERTS, INC.

What Color Is My Application?

- W ▶ Web Application (Web)
  - DB smaller than RAM
  - 90% or more simple queries
- O ▶ Online Transaction Processing (OLTP)
  - DB slightly larger than RAM to 1TB
  - 20-40% small data write queries
  - Some long transactions and complex read queries
- D ▶ Data Warehousing (DW)

What Color Is My Application?

W ▶ Web Application (Web)

- DB smaller than RAM
- 90% or more simple queries

O ▶ Online Transaction Processing (OLTP)

- DB slightly larger than RAM to 1TB
- 20-40% small data write queries
- Some long transactions and complex read queries

D ▶ Data Warehousing (DW)

- Large to huge databases (100GB to 100TB)
- Large complex reporting queries
- Large bulk loads of data
- Also called "Decision Support" or "Business Intelligence"

PGX
POSTGRESQL
EXPERTS, INC.

What Color Is My Application?

W ▶ Web Application (Web)
- CPU-bound
- Moles: caching, pooling, connection time

O ▶ Online Transaction Processing (OLTP)
- CPU or I/O bound
- Moles: locks, cache, transactions, write speed, log

D ▶ Data Warehousing (DW)
- I/O or RAM bound
- Moles: seq scans, resources, bad queries, bulk loads

Rules of Whack-a-Mole

1. Most “database performance problems”, or Moles, are not actually *database* performance problems.
2. Less than 10% of Moles cause 90% of performance degradation.
   - corollary: we don't care about the other 90% of Moles
3. At any time, it is usually only possible to observe and troubleshoot, or *Whack*, the “largest” Mole.
4. Different application types usually have different Moles and need different troubleshooting.

PGX
POSTGRESQL
EXPERTS, INC.

Whack-a-Mole Strategy

1. setup
   - identify the application type
   - gather problem reports
2. baseline
3. the hunt
   - use tools to seek mole in most likely locations
   - keep trying locations until mole is found
4. the whack
5. repeat hunt and whack
   - until enough moles are gone

Part 2: Baseline

PGX
POSTGRESQL
EXPERTS, INC.

What's a Baseline?

- Gather information about the system
  - you need to know what's happening at every level of the stack
  - identify potential trouble areas to come back to later
- Basic Setup
  - check the hardware/OS setup for sanity
  - apply the conventional postgresql.conf calculations
  - do conventional wisdom middleware and application setup
  - should be fast run-though, like an hour

PGX
POSTGRESQL
EXPERTS, INC.

Why Baseline?

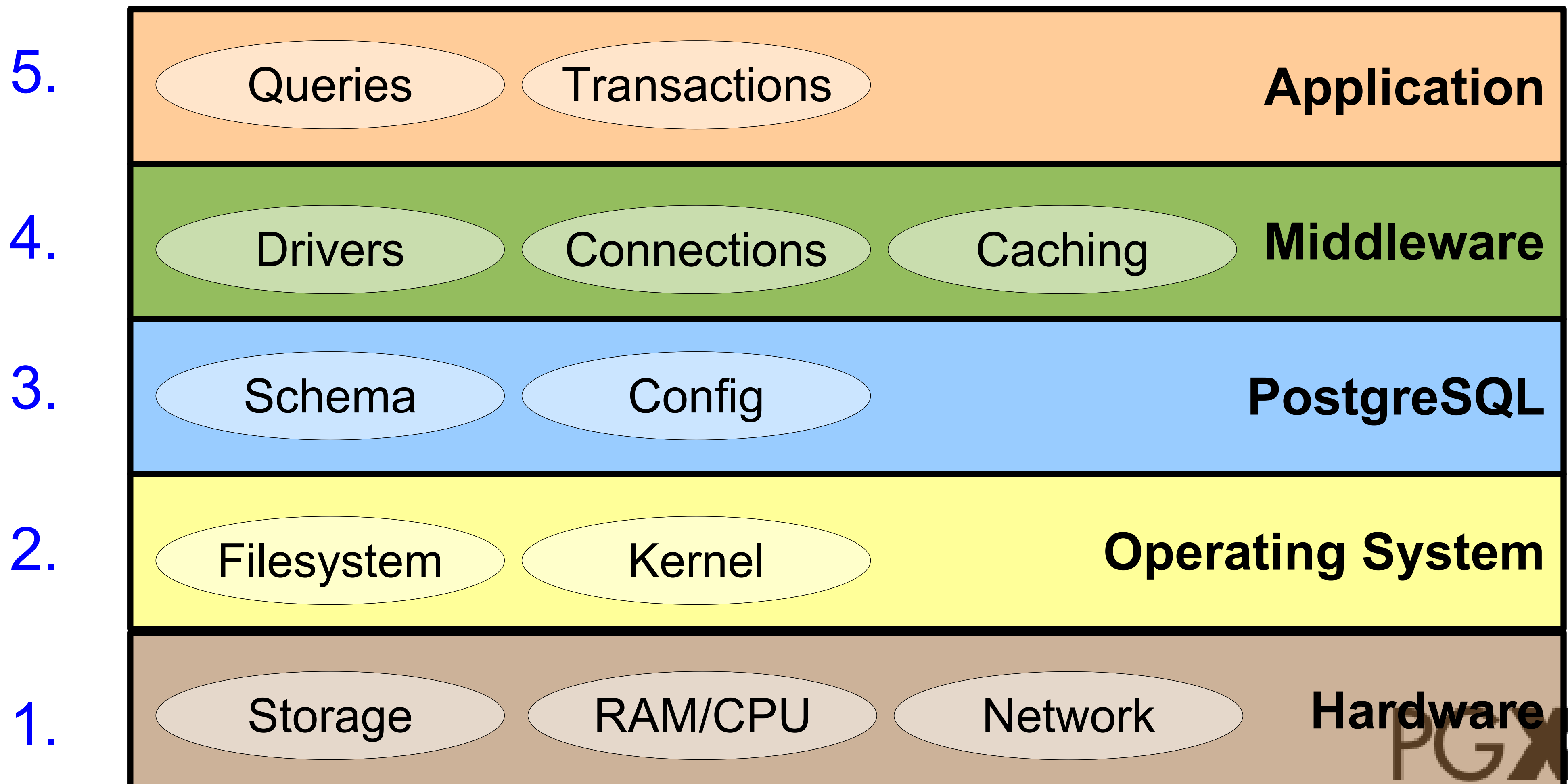
- Why not just go straight to Whacking?
  - extremely poor basic setup may mask more serious issues
  - baseline setup may turn out to be all that's needed
  - deviations from baseline can be clues to finding Moles
  - baseline will make your setup comparable to other installations so you can check tests
  - clients/sysadmins/developers are seldom a reliable source of bottleneck information

Steps for Baseline

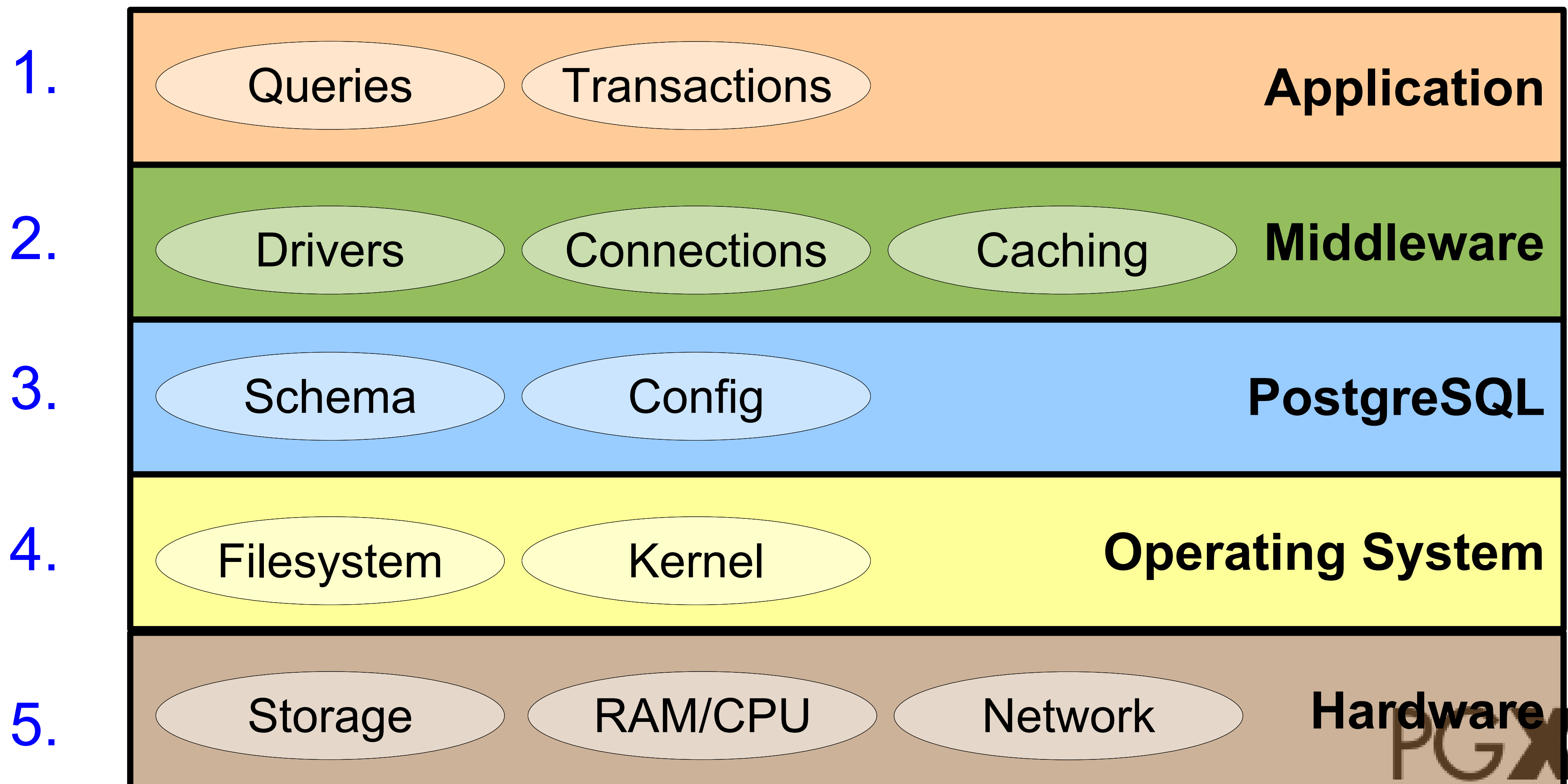
1. Hardware setup
2. Filesystem & OS Setup
3. PostgreSQL.conf
4. Drivers, Pooling & Caching
5. Application Setup Information

PGX
POSTGRESQL
EXPERTS, INC.

Steps for Baseline



Steps for Baseline



PGX
POSTGRESQL
EXPERTS, INC.

Hardware Baseline

- Gather Data
  - Server
    - CPU model, speed, number, arch
    - RAM quantity, speed, configuration
  - Storage
    - Interface (cards, RAID)
    - Disk type, size, speed
    - Array/SAN configuration
  - Network
    - network type and bandwith
    - devices and models
    - switch/routing configuration

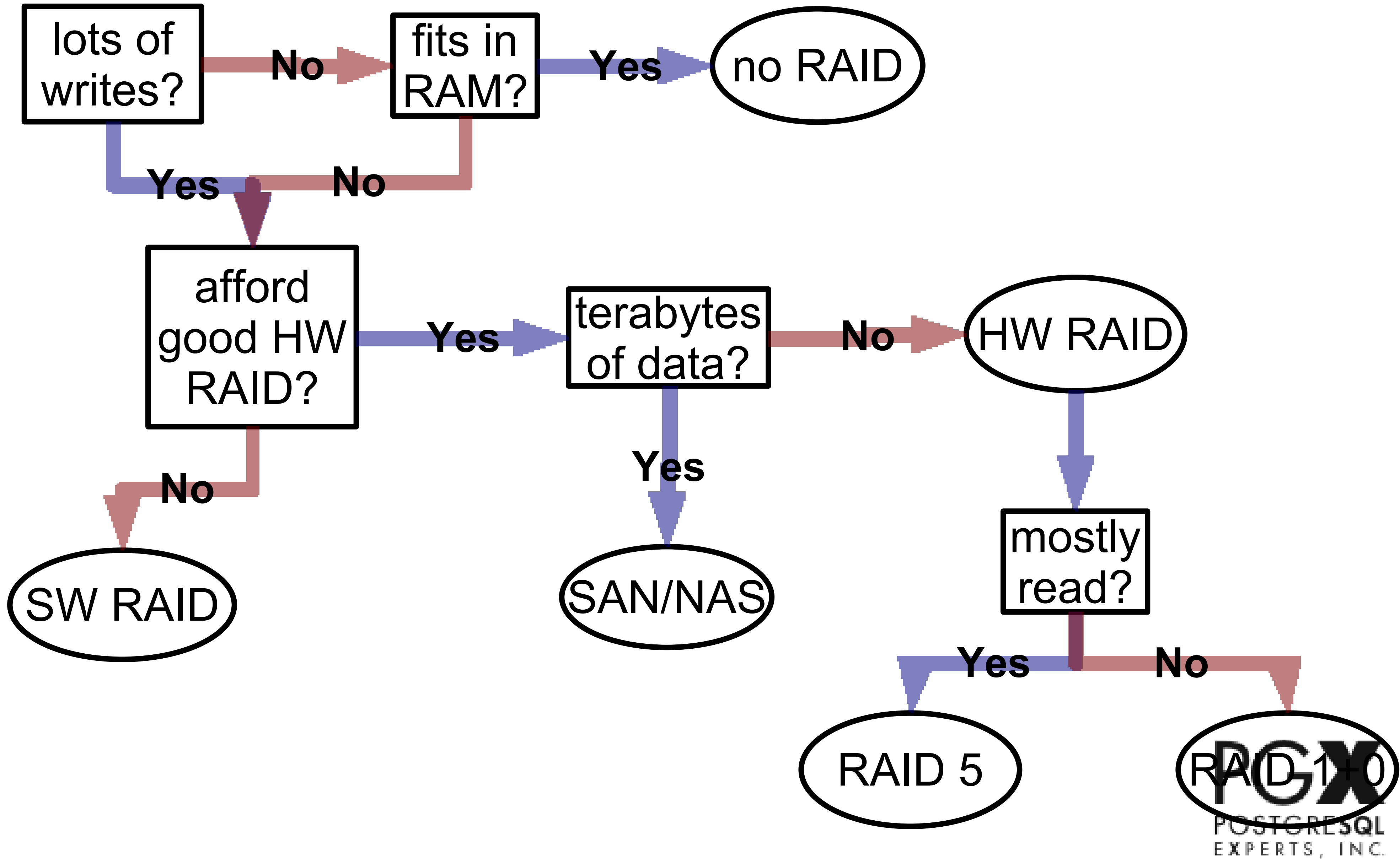
PGX
POSTGRESQL
EXPERTS, INC.

Hardware Baseline

- Baseline
  - Storage
    - Use appropriate RAID configuration
    - Turn on write caching if safe
    - Make sure you're using all channels/devices
  - Network
    - application servers & DB server should be on dedicated network
    - use redundant connections & load balancing if available

PGX
POSTGRESQL
EXPERTS, INC.

Storage Decision Tree



Hardware Baseline

- Medium-Volume OLTP Application
  - 2 Appservers, 1 DB server
    - on private gig-E network
  - DB server is HP DL380
    - 2x Quad Xeon
    - 16 GB RAM
  - Attached to shared SCSI storage box
    - 7 drives available
      - 2 in RAID 1 for xlog
      - 4 in RAID 1+0 for DB
      - OS on internal drives

PGX
POSTGRESQL
EXPERTS, INC.

Operating System Baseline

- OS
  - gather data
    - OS, version, patch level, any modifications made
    - hardware driver information
    - system usage by other applications (& resource usage)
  - baseline
    - update to latest patch level (probably)
    - update hardware drivers (probably)
    - migrate conflicting applications
      - other DBMSes
      - other applications with heavy HW usage

PGX
POSTGRESQL
EXPERTS, INC.

Operating System Baseline

- Filesystem
  - gather data
    - filesystem type, partitions
    - locations of files for OS, PostgreSQL, other apps
    - filesystem settings
  - baseline
    - move xlog to separate disk/array/partition
    - set filesystem for general recommendations
      - lower journaling levels
      - directio for xlog (if possible)
      - aggressive caching for DB
      - other settings specific to FS

PGX
POSTGRESQL
EXPERTS, INC.

Operating System Baseline

- OLTP Server running on Solaris 10
  - Updated to Update5
    - Fibercard driver patched
  - Dedicated Server
    - MySQL removed to less critical machine
  - Solaris settings configured:
    - set segmapsize=10737418240
    - set ufs:freebehind=0
    - set segmap_percent=50
  - Filesystem configured:
    - mount -o forcedirectio /dev/rdsk/cntndnsn /mypath/pg_xlog
    - tunefs -a 128 /mypath/pg_xlog

PGX
POSTGRESQL
EXPERTS, INC.

PostgreSQL Baseline

- Gather Data
  - schema
    - tables: design, data size, partitioning, tablespaces
    - indexes
    - stored procedures
  - .conf settings
    - ask about any non-defaults
  - maintenance
    - have vacuum & analyze been run?
    - when and with what settings?

PGX
POSTGRESQL
EXPERTS, INC.

PostgreSQL Baseline

- .conf Baseline for modern servers
  - shared_buffers = 25% RAM
  - work_mem = [W] 512K [O] 2MB [D] 128MB
    - but not more than RAM / no_connections
  - maintenance_work_mem = 1/16 RAM
  - checkpoint_segments = [W] 8, [O],[D] 16-64
  - wal_buffers = 1MB [W], 8MB [O],[D]
  - effective_cache_size = 2/3 * RAM

PGX
POSTGRESQL
EXPERTS, INC.

PostgreSQL Baseline

- maintenance baseline
  - [W][O] set up autovaccuum
    - autovacuum = on
    - vacuum_cost_delay = 20ms
    - lower *_threshold for small databases
  - [D] set up vacuum/analyze batches with data batch import/update

PGX POSTGRESQL EXPERTS, INC.

Middleware Baseline

- Gather data
  - DB drivers: driver, version
  - Connections: method, pooling (if any), pooling configuration
  - Caching: methods, tools used, versions, cache configuration
  - ORM: software, version
- Baseline
  - Update to latest middleware software: drivers, cache, etc.
  - Utilize all pooling and caching methods available
    - use prepared queries
    - plan, parse, data caching (if available)
    - pool should be sized to the maximum connections needed
    - 5-15 app connections per DB connection
    - persistent connections if no pool

Application Baseline

- Gather data
  - application type
  - transaction model and volume
  - query types and relative quantities
    - get some typical queries, or better, logs
  - stored procedure execution, if any
  - understand how the application generally works
    - get a use perspective
    - find out purpose and sequence of usage
    - usage patterns: constant or peak traffic?

Part 3: Tools for Mole-Hunting

PGX POSTGRESQL EXPERTS, INC.

Types of Tools: HW & OS

- Operating system tools
  - simple & easy to use, non-invasive
  - let you monitor hardware usage, gross system characteristics
  - often the first option to tell what kind of Mole you have
- Benchmarks & microbenchmarks
  - very invasive: need to take over host system
  - allow comparable testing of HW & OS

Types of Tools: PostgreSQL

- pg_stat* views, DTrace
  - minimally invasive, fast
  - give you more internal data about what's going on in the DB realtime
  - let you spot schema, query, procedure, lock problems
- PostgreSQL log & pg_fouine & csvlog
  - somewhat invasive, slow
  - allows introspection on specific types of db activity
  - compute overall statistics on query, DB load
- Explain Analyze
  - troubleshoot “bad queries”
  - for fixing specific queries only

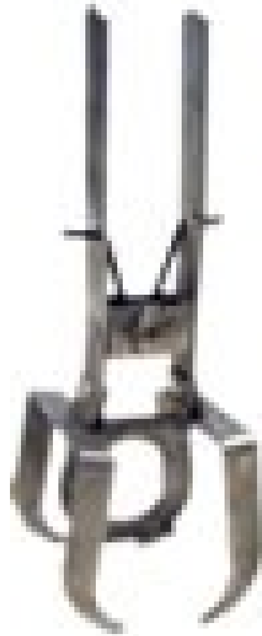
PGX
POSTGRESQL
EXPERTS, INC.

Types of Tools Not Covered

... but you should use about anyway

- Application server tools
  - response time analysis tools
  - database activity monitoring tools
  - cache usage monitoring
- Workload simulation & screen scraping
  - the best benchmark is a simulation of your own application
  - tools like lwp and log replay tools
- Bug detection tools
  - valgrind, MDB, GDB
  - sometimes your performance issue is a genuine software bug

PGX
POSTGRESQL
EXPERTS, INC.



Part 3a: Operating System Tools

PGX
POSTGRESQL
EXPERTS, INC.

OS Tools: ps (dbstat)

- lets you see running PostgreSQL processes
  - gives you an idea of concurrent activity & memory/cpu usage
  - lets you spot hung and long-running statements
- pg_top is better (Linux)
  - gives you ps content
  - plus information about what queries are running

OS Tools: mpstat

- see CPU activity for each CPU
  - find out if you're CPU-bound
  - see if all CPUs are being utilized
  - detect context-switch issues

OS Tools: vmstat, free

- Watch memory usage
  - see if RAM is saturated
    - are you not able to cache enough?
    - are you swapping?

PGX
POSTGRESQL
EXPERTS, INC.

OS Tools: iostat

- monitor usage of storage
  - see if I/O is saturated
  - see if one storage resource is bottlenecking everything else
  - watch for checkpoint spikes

PGX
POSTGRESQL
EXPERTS, INC.

OS Tools: sar (Linux)

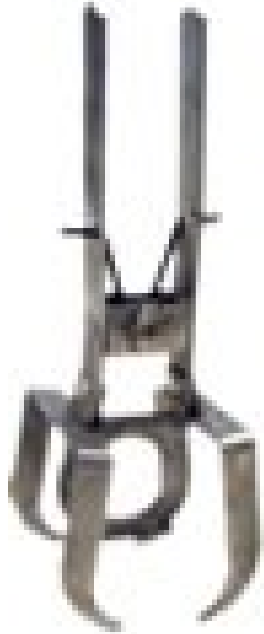
- retrieve iostat, mpstat, vmstat etc. information retroactively
  - Linux stores a snapshot of this data every 10 minutes
    - may not be detailed enough
  - check system load for when the crash/bottleneck happened even if you weren't monitoring

PGX
POSTGRESQL
EXPERTS, INC.

OS Tools: DTrace (Solaris, BSD)

- scriptable tracing tool
  - trace the full application stack
  - compute resource uses by single query or type of operation
  - look for "deep" performance bottlenecks in the PostgreSQL code

PGX
POSTGRESQL
EXPERTS, INC.



Part 3b: Benchmarks

PGX
POSTGRESQL
EXPERTS, INC.

Benchmarks: filesystem

- dd
  - simple sequential writes / reads only
- bonnie++ 1.94
  - see I/O throughput & issues
  - check seek, random write speeds
    - concurrency limited
  - use version 1.94 to check concurrency & lag time
- IOZone
  - check speeds on specific operations
    - do not run in "auto mode"
    - concurrency broken

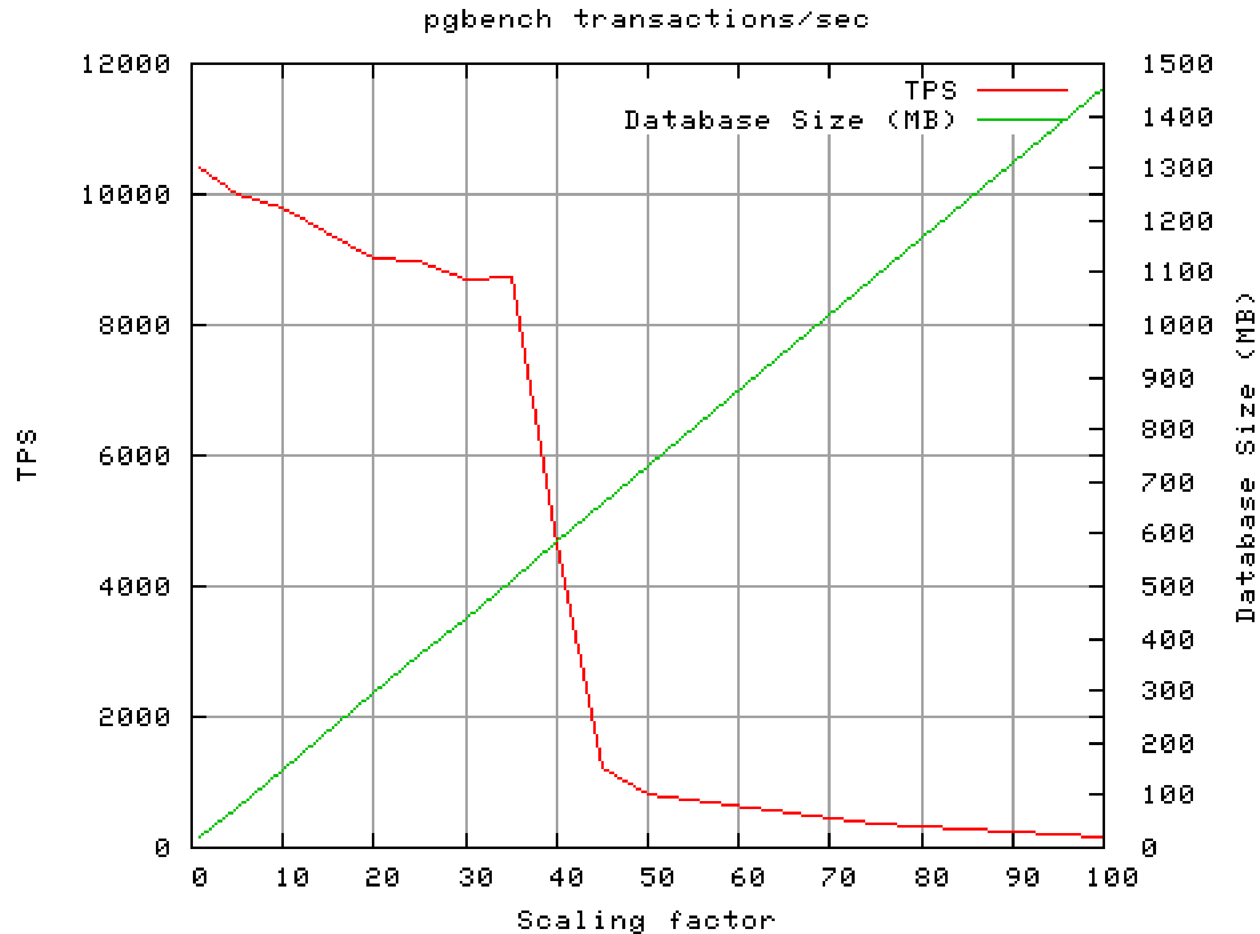
PGX POSTGRESQL EXPERTS, INC.

Benchmarks: pgbench

- Very simple DB microbenchmark
  - tests mostly I/O and connection processing speed
  - doesn't test locking, computation, or query planning
  - results sometimes not reproduceable
  - mostly useful to prove large OS+HW issues
    - *not* useful for fine performance tuning
- Run test appropriate to your workload
  - cached in shared_buffers size
  - cached in RAM size
  - on disk size
    - a little
    - a lot

PGX
POSTGRESQL
EXPERTS, INC.

Benchmarks: pgbench



Thanks to Greg Smith for this graph!

PGX POSTGRESQL EXPERTS, INC.

Benchmarks: Serious

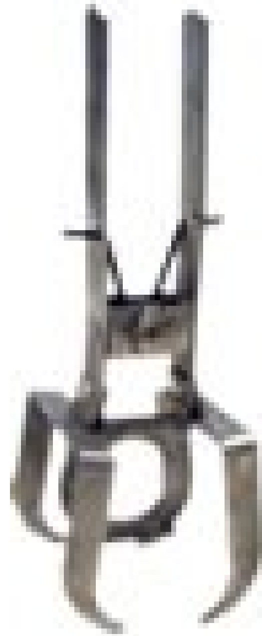
- Use serious benchmarks only when you have a problem which makes the system unusable
  - you'll have to take the system offline
  - it gives you reproduceable results to send to vendors & mailing lists
  - best way to go after proven bugs you can't work around

PGX
POSTGRESQL
EXPERTS, INC.

Benchmarks: Serious

- DBT2
  - Serious OLTP benchmark
    - based on TPCC
    - reproducable results, works out a lot more of the system
    - complex & time-consuming to set up, run
- DBT3, DBT5 in process
  - new OLTP plus DW benchmarks
- Others being developed
  - pgUnitTest
  - EAstress
  - BenchmarkSQL

PGX
POSTGRESQL
EXPERTS, INC.



Part 3c: System Views

PGX POSTGRESQL EXPERTS, INC.

pg_stat_database, pg_database_size

- Get general traffic statistics
  - number of connections
  - transaction commits througput
- See rollback and hit ratios
  - are you dealing with a lot of rollbacks due to aborted transactions?
  - is little or none of the database fitting in the cache?
- see how large your database is
  - scope RAM & I/O scaling
  - check RAID config

PGX
POSTGRESQL
EXPERTS, INC.

pg_tables, pg_relation_size

- scope out the tables
  - how many are there?
  - do they have triggers?
    - may cost you on updates
- check size of each table & index
  - monitor for bloating
  - see if tablespaces or partitions are recommended

PGX POSTGRESQL EXPERTS, INC.

pg_stat_activity

- check concurrent query activity
  - get an idea of the proportion of idle connections
  - spot check types of activity
  - much better than ps for catching runaway transactions
- use pg_top instead
  - for above plus CPU/RAM usage

PGX POSTGRESQL EXPERTS, INC.

pg_locks

- Spot-check for lock conflicts
  - a few are normal in high-data-integrity applications, but a lot is bad
    - locks held for a long time are *really* bad
  - often a sign that you should change your data locking strategy
    - or simply lower deadlock_timeout
  - if you have ungranted locks, check them against pg_stat_activity

PGX
POSTGRESQL
EXPERTS, INC.

pg_stat[io]_user_tables, pg_stat[io]_user_indexes

- check relative table activity
  - how much select vs. update traffic?
- look for seq scans
  - do we need more/different indexes?
- check index activity
  - should some indexes be dropped?
  - are some very large indexes dominating I/O?

PGX POSTGRESQL EXPERTS, INC.

pg_stat_bgwriter

- see if the bgwriter is clearing the caches
  - are we suffering checkpoint spikes?

PGX
POSTGRESQL
EXPERTS, INC.

pg_stat_user_functions (new 8.4)

- check execution time for each function
  - including difference between code execution and callouts
- find your slowest functions
  - then instrument them with auto_explain (see later)

PGX POSTGRESQL EXPERTS, INC.

pg_stat_statements (new in 8.4)

- realtime "top query" information
  - how many queries executing
  - slowest/most frequent queries

Part 3d: Activity Log

PGX
POSTGRESQL
EXPERTS, INC.

How to use the pg_log

1. Figure out what behavior you're trying to observe
2. Turn only those options on
3. Run a short, reproducible test case (if possible)
   - if not, just deliberately trigger the problem behavior
4. Digest the log results

How to use the pg_log

- If you have to log a production server, you'll need to filter out the noise. Try:
  - rotating the log every hour,
  - turning on query logging for minutes to an hour,
  - or logging only one connection.

PGX
POSTGRESQL
EXPERTS, INC.

Basic query monitoring

```
log_destination = 'csvlog'
redirect_stderr = on
log_min_duration_statement = 0
```

PGX
POSTGRESQL
EXPERTS, INC.

pg_fouine

- Calculates overall query statistics
  - find the slowest queries
  - find the ones running the most frequently
  - probably your best way to find the “biggest query moles”
- Other similar tools
  - PGSI
  - PQA

PGX
POSTGRESQL
EXPERTS, INC.

Harvesting slow queries

```
log_min_duration_statement = 30
log_locks = on
deadlock_timeout = 5s
log_temp_files = 32kB
```

Monitoring connections

```
log_connections = on
log_disconnections = on
```

Auto-Explain (new for 8.4)

- log explain plans to the pg_log
  - turn on and off dynamically
  - add logging of explain plans for specific queries in your code
    - especially functions
- helps solve
  "I can't reproduce the slow query in development"

PGX POSTGRESQL EXPERTS, INC.

Part 3e: EXPLAIN ANALYZE

PGX POSTGRESQL EXPERTS, INC.

The “Bad Query” tool

- After you've found your most costly queries
- Use EXPLAIN ANALYZE to find out why they're so costly
  - sometimes you can fix them immediately
  - other times they indicate problems in other areas
    - HW issues
    - schema issues
    - lack of db maintenance

PGX
POSTGRESQL
EXPERTS, INC.

Reading EXPLAIN ANALYZE

- It's an inverted tree
  - don't start at the top
  - execution starts at the innermost node and works up and out
  - look for the *lowest* node with a problem
- Read it holistically
  - some nodes execute in parallel and influence each other
  - “gaps” between nodes can be significant
  - subtrees which are slow don't matter if other subtrees are slower

PGX
POSTGRESQL
EXPERTS, INC.

Things to Look For: Examples

- Bad rowcount estimates
  - cause the query to choose bad query plans
    - worse than 3x or 0.3x will often cause wrong plan
  - generally can be fixed with increased planner statistics
    - or adjusting function row estimate
  - sometimes require query re-writing

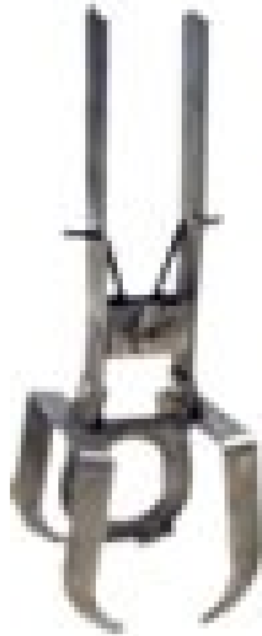
PGX
POSTGRESQL
EXPERTS, INC.

Things to Look For: Examples

- Slow Scans
  - index or seq scans which seem too slow by estimate
  - usually indicates either
    - table/index bloat due to poor maintenance
    - I/O saturation
    - I/O problems
    - not enough RAM

Things to Look For: Examples

- On-disk sorts
  - disk sorts are much slower than in memory
    - look at for queries using more sort RAM than is allocated
    - increase work_mem



Part 4: Hunting Moles

PGX POSTGRESQL EXPERTS, INC.

Hunting Moles

- What kind?
  - What are the symptoms?
    - response times
    - error messages
- When?
  - activity which causes the problem
    - general slowdown or specific operation, or periodic?
    - caused just by one activity, or by several?
  - concurrent system activity
    - system/DB load?
    - what other operations are going on on the system?

PGX
POSTGRESQL
EXPERTS, INC.

Common Types of Moles

- I/O Mole
  - behavior: cpu underutilized: ram available, I/O saturated for at least one device
  - habitats: [D], [O], any heavy write load or very large database
  - common causes:
    - bad I/O hardware/software
    - bad I/O config
    - not enough ram
    - too much data requested from application
    - bad schema: missing indexes or partitioning needed

Common Types of Moles

- CPU Mole
  - behavior: cpus at 90% or more: ram available, I/O not saturated
  - habitats: [W], [O], mostly-read loads or those involving complex calculation in queries
  - causes:
    - too many queries
    - insufficient caching/pooling
    - too much data requested by application
    - bad queries
    - bad schema: missing indexes
  - *can be benign*: most DB servers should be CPU-bound at maximum load

PGX
POSTGRESQL
EXPERTS, INC.

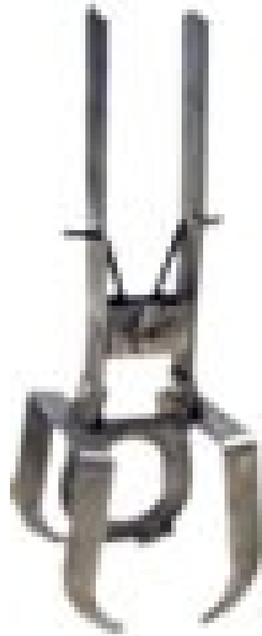
Common Types of Moles

- Locking Mole
  - behavior: nothing on DB or App server is at maximum, but many queries have long waits, often heavy context switching, pg_locks sometimes shows waits
  - habitats: [O], [D], or loads involving pessimistic locking and/or stored procedures
  - causes:
    - long-running transactions/procedures
    - cursors held too long
    - pessimistic instead of optimistic locking or userlocks
    - poor transaction management (failure to rollback)
    - various buffer settings in .conf too low
    - PostgreSQL SMP scalability limits

PGX
POSTGRESQL
EXPERTS, INC.

Common Types of Moles

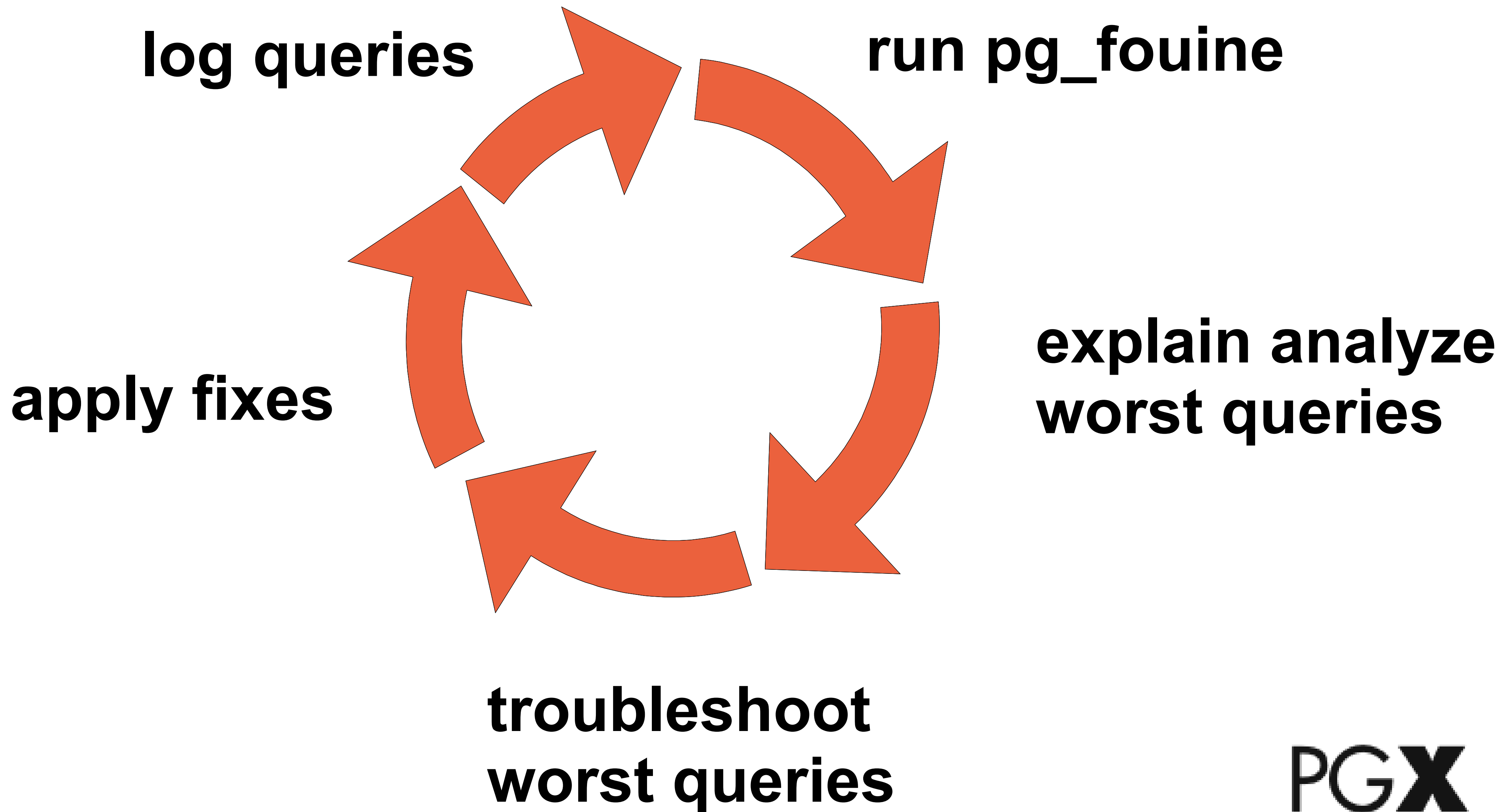
- Application Mole
  - behavior: nothing on DB server is at maximum, but RAM or CPU on the App servers is completely utilized
  - habitats: common in J2EE
  - causes:
    - not enough application servers
    - too much data / too many queries
    - bad caching/pooling config
    - driver issues
    - ORM



Part 4a:
The Optimization Cycle

PGX
POSTGRESQL
EXPERTS, INC.

Query Optimization Cycle

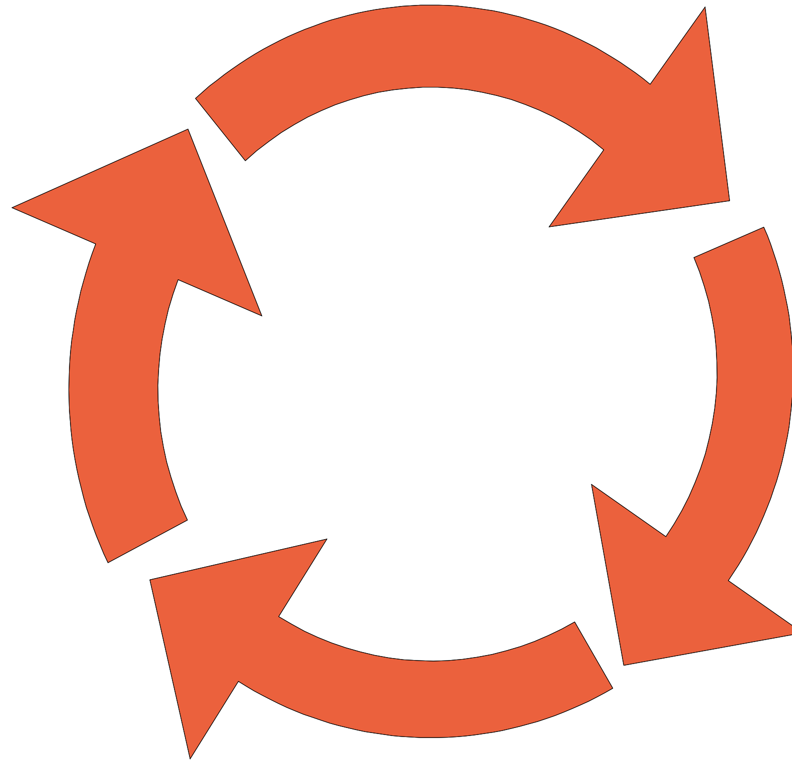


Query Optimization Cycle (8.4)

check pg_stat_statement

explain analyze
worst queries

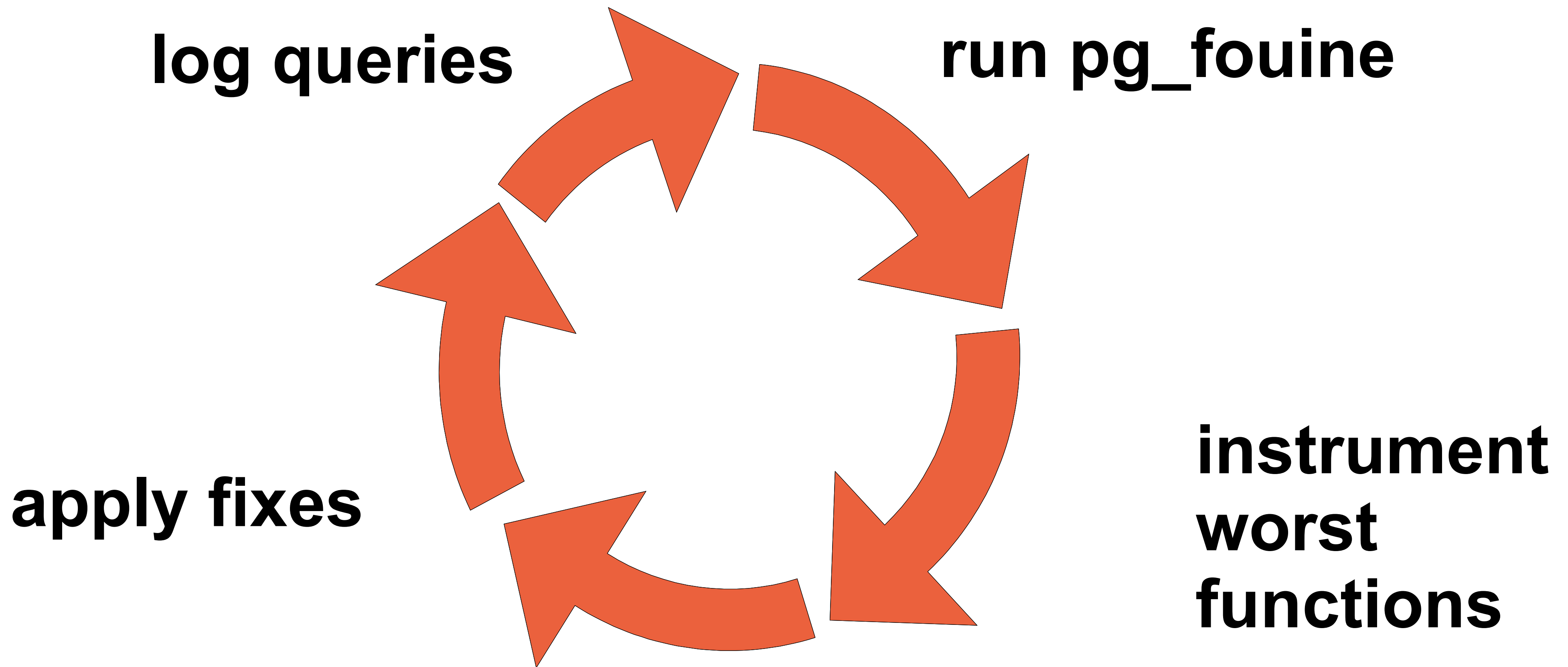
apply fixes



troubleshoot
worst queries

PGX
POSTGRESQL
EXPERTS, INC.

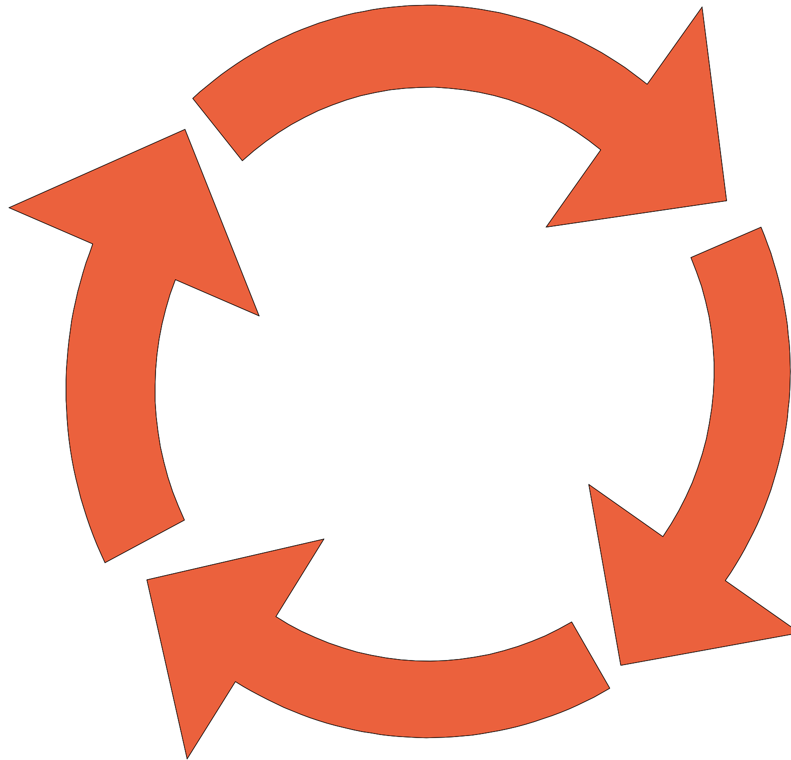
Procedure Optimization Cycle



PGX
POSTGRESQL
EXPERTS, INC.

Procedure Optimization (8.4)

check pg_stat_function

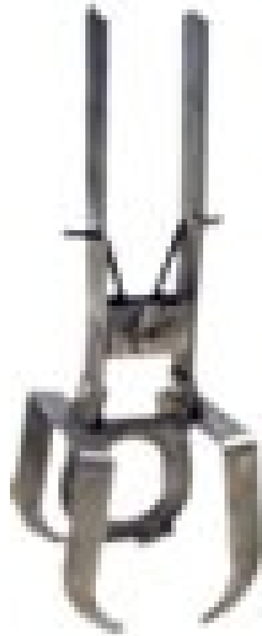


instrument
worst
functions

find slow
operations

apply fixes

PGX
POSTGRESQL
EXPERTS, INC.



Part 4b: Hunting Moles Examples

PGX
POSTGRESQL
EXPERTS, INC.

Too Many Queries

- The Setup
  - c++ client-server application took 3+ minutes to start up
- The Hunt
  - set pg_log to log queries
    - ran application startup
  - ran through pg_fouine
    - showed over 20,000 queries during startup
    - most of them identical when normalized
- The Whack
  - the application was walking several large trees, node-by-node
  - taught the programmers to do batch queries and use connect_by()

PGX
POSTGRESQL EXPERTS, INC.

Slow DW

- Setup
  - Data warehousing / monitoring application
    - DB was 300GB, server 16GB RAM
  - Some queries would time out
    - despite few users on the server
  - CPU was available, RAM was full of cached data
  - I/O seemed underused
    - except it never got above a very low ceiling

Slow DW

- The Hunt
  - checked some slow queries using EXPLAIN
    - older data partitions were slow
  - used dd, bonnie++, ioZone to check I/O behavior
    - iSCSI storage was *very* slow (60mb/s)
- The Whack
  - recommended fix/replace of iSCSI storage
    - wasn't feasible so:
  - upgraded server to 64GB RAM
  - exported large objects in DB to separate filesystem
    - shrank database by 75%

PGX
POSTGRESQL
EXPERTS, INC.

Connection Management

- The Setup
  - JSP web application good 23 hours per day, but bombing during the peak traffic hour
    - DB server would run out of RAM and stop responding
- The Hunt
  - watched pg_stat_activity and process list during peak periods, took snapshots
    - saw that connections went up to 2000+ during peak, yet many of them were idle
    - verified this by logging connections & disconnections
  - checked Tomcat configuration
    - connection pool: 200 connections
    - servers were set to reconnect after 10 seconds timeout

Connection Management

- The Whack
  - Tomcat was “bombing” the database with thousands of failed connections
    - faster than the database could fulfill them
  - Fixed configuration
    - min_connections for pool set to 700
    - connection_timeout and pool connection timeout synchronized at 20 seconds
  - Suggested improvements
    - upgrade to a J2EE architecture with better pooling

PGX
POSTGRESQL
EXPERTS, INC.

Locked Database

- Setup
  - monitoring application
    - constant data inflow
    - constant user queries against data
    - periodic materialized view creation via cron jobs
  - database "locked up"
    - all queries were timing out
- The Hunt
  - check pg_locks and pg_stat_activity
    - several CREATE TABLE statements were pending locks
    - several bulk updates and inserts were pending locks
    - all SELECTs were on hold behind these

PGX
POSTGRESQL
EXPERTS, INC.

Locked Database

- The Whack
  - application was creating new partitions at runtime
    - created a circular deadlock situation with UPDATEs
  - changed application to pre-allocate partitions nightly
    - locking situation went away

Checkpoint Spikes

- Setup
  - OLTP benchmark, but not as fast as MySQL
  - Nothing was maxxed
  - Query throughput cycled up and down
- The Hunt
  - checked iostat, saw 5-minute cycle
  - installed, checked pg_stat_bgwriter
    - showed high amount of buffers_checkpoint
- The Whack
  - increased bgwriter frequency, amounts
  - spikes decreased, overall throughput rose slightly

PGX
POSTGRESQL
EXPERTS, INC.

Undead Transactions

- The Setup
  - Perl OLTP application was fast when upgraded, but became slower & slower with time
- The Hunt
  - checked db maintenance schedule: vacuum was being run
    - yet pg_tables showed tables were growing faster than they should, indexes too
    - vacuum analyze verbose showed lots of “dead tuples could not be removed”
  - checked pg_stat_activity and process list
    - “idle in transaction”
    - some transactions were living for days

PGX POSTGRESQL EXPERTS, INC.

Undead Transactions

- The Whack
  - programmers fixed application bug to rollback failed transactions instead of skipping them
  - added “undead transaction” checker to their application monitoring

Is The Mole Dead?

Yes, which means it's time to move on to the *next* mole.

Isn't this fun?

PGX POSTGRESQL EXPERTS, INC.

Further Questions

- Josh Berkus
  - josh@postgresql.org
  - pgexperts
    - josh.berkus@pgexperts.com
    - www.pgexperts.com
  - it.toolbox.com/blogs/ database-soup
- More Advice
  - www.postgresql.org/docs
  - pgsql-performance list
  - www.planetpostgresql.org
  - irc.freenode.net
    - #postgresql
- Slides/files
  - www.pgexperts.com/document.html

Special thanks for borrowed content to:
www.MolePro.com for the WhackaMole Game
Greg Smith for pgbench and bonnie++ results
Robert Treat and Jignesh Shah for Dtrace samples
The Ottawa Senators name and the Senators Logo are property of the Ottawa Senators

This talk is copyright 2009 Josh Berkus, and is licensed under the creative commons attribution license