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

-- Not Your Client, Inc.

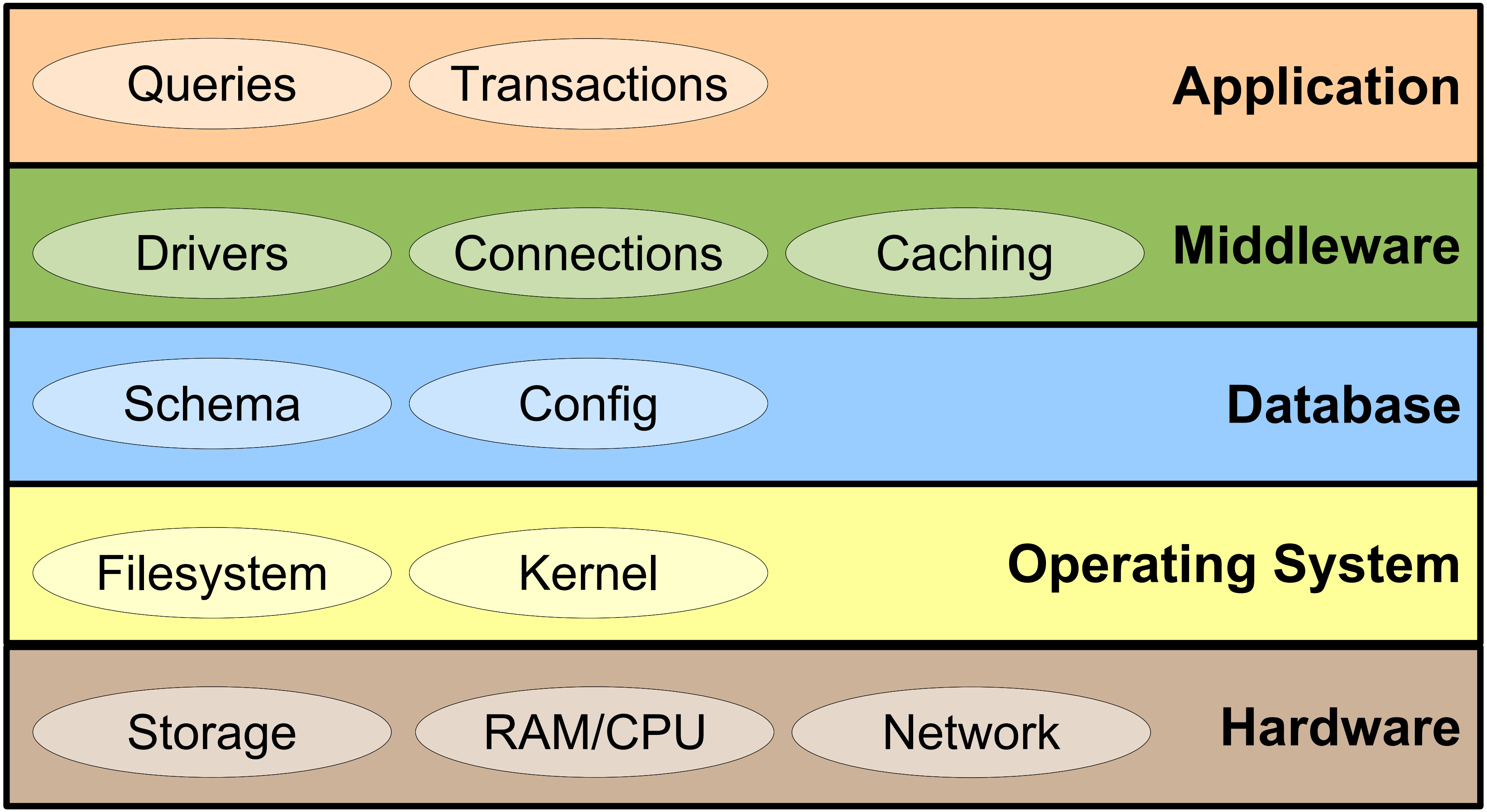
“My database is slow.”

-- Every Single Support Client LLC

Performance Whack-a-Mole

Part 1:
The Rules

The Stack



Queries Transactions **Application**

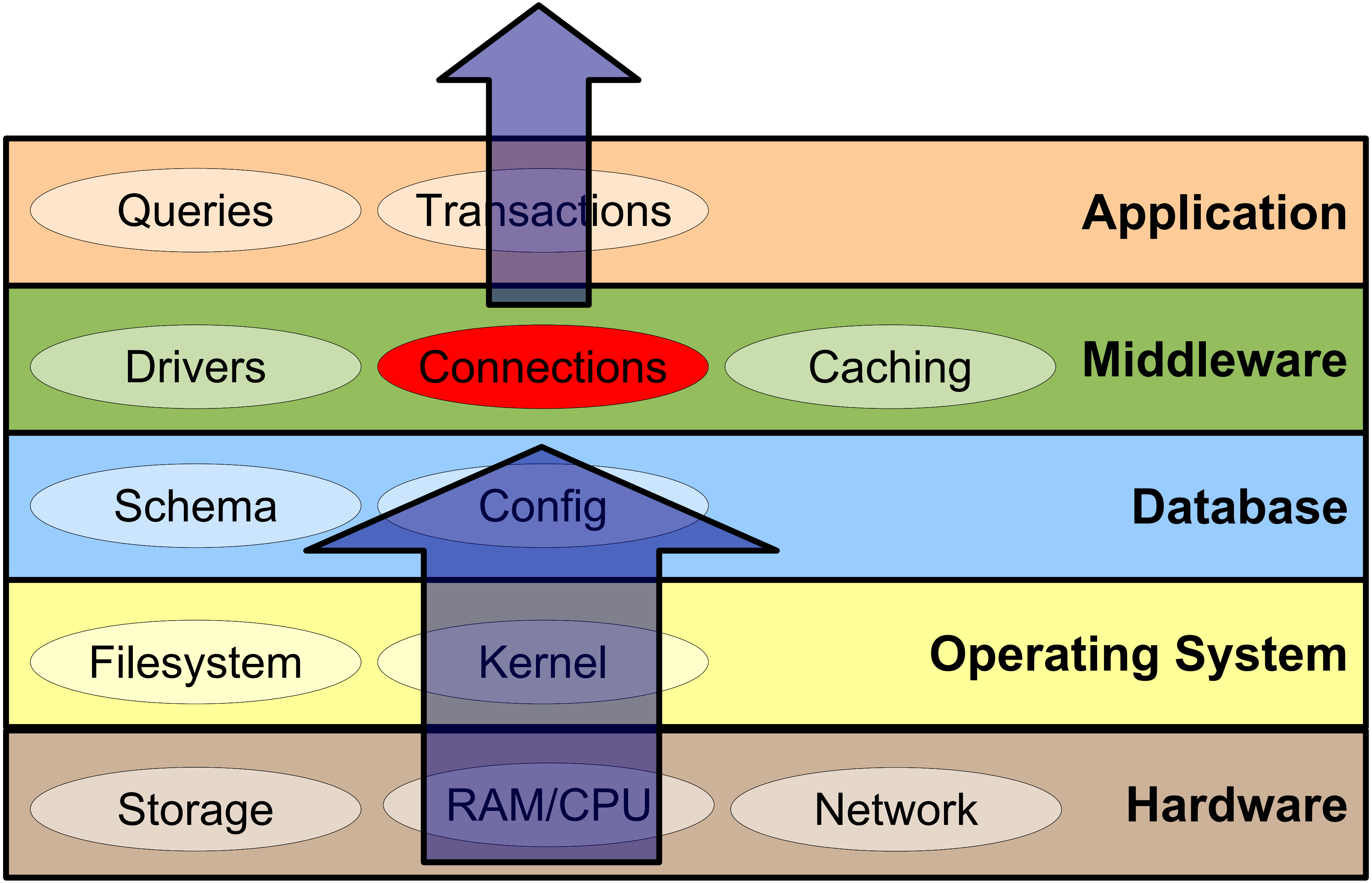
Drivers Connections Caching **Middleware**

Schema Config **Database**

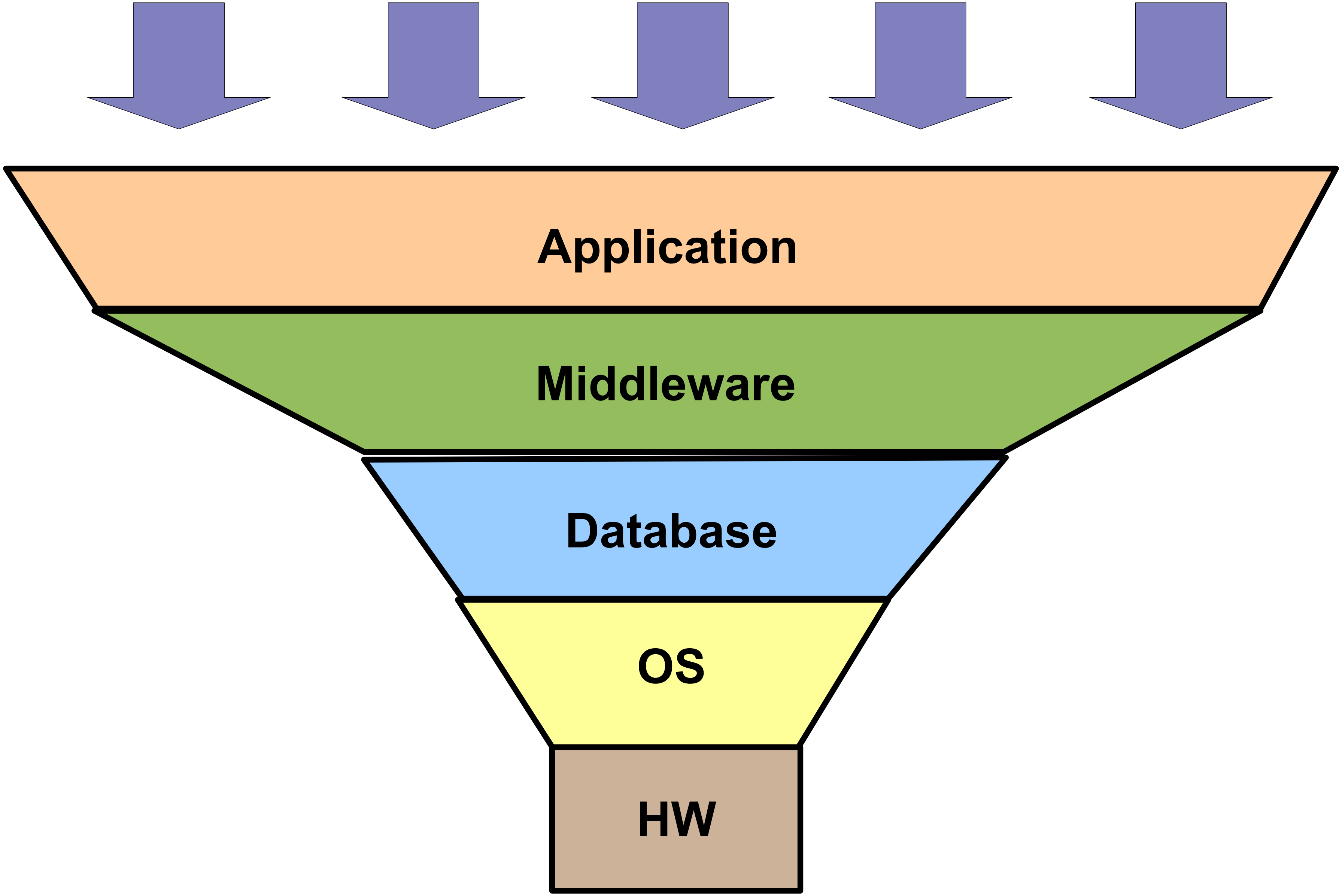
Filesystem Kernel **Operating System**

Storage RAM/CPU Network **Hardware**

The Stack


Queries
Transactions
Application
Drivers
Connections
Caching
Middleware
Schema
Config
Database
Filesystem
Kernel
Operating System
Storage
RAM/CPU
Network
Hardware

The Stack



Application

Middleware

Database

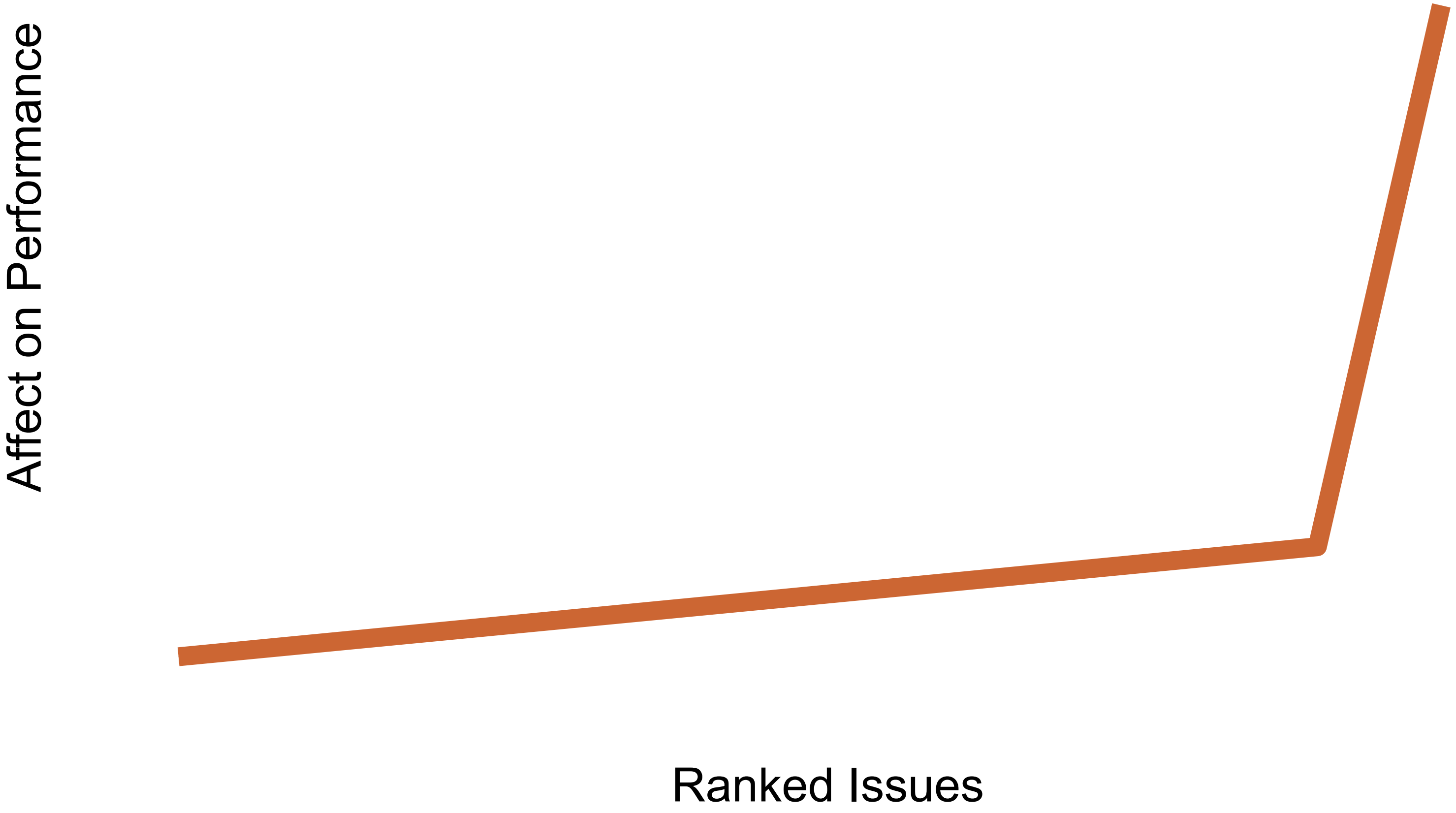
OS

HW

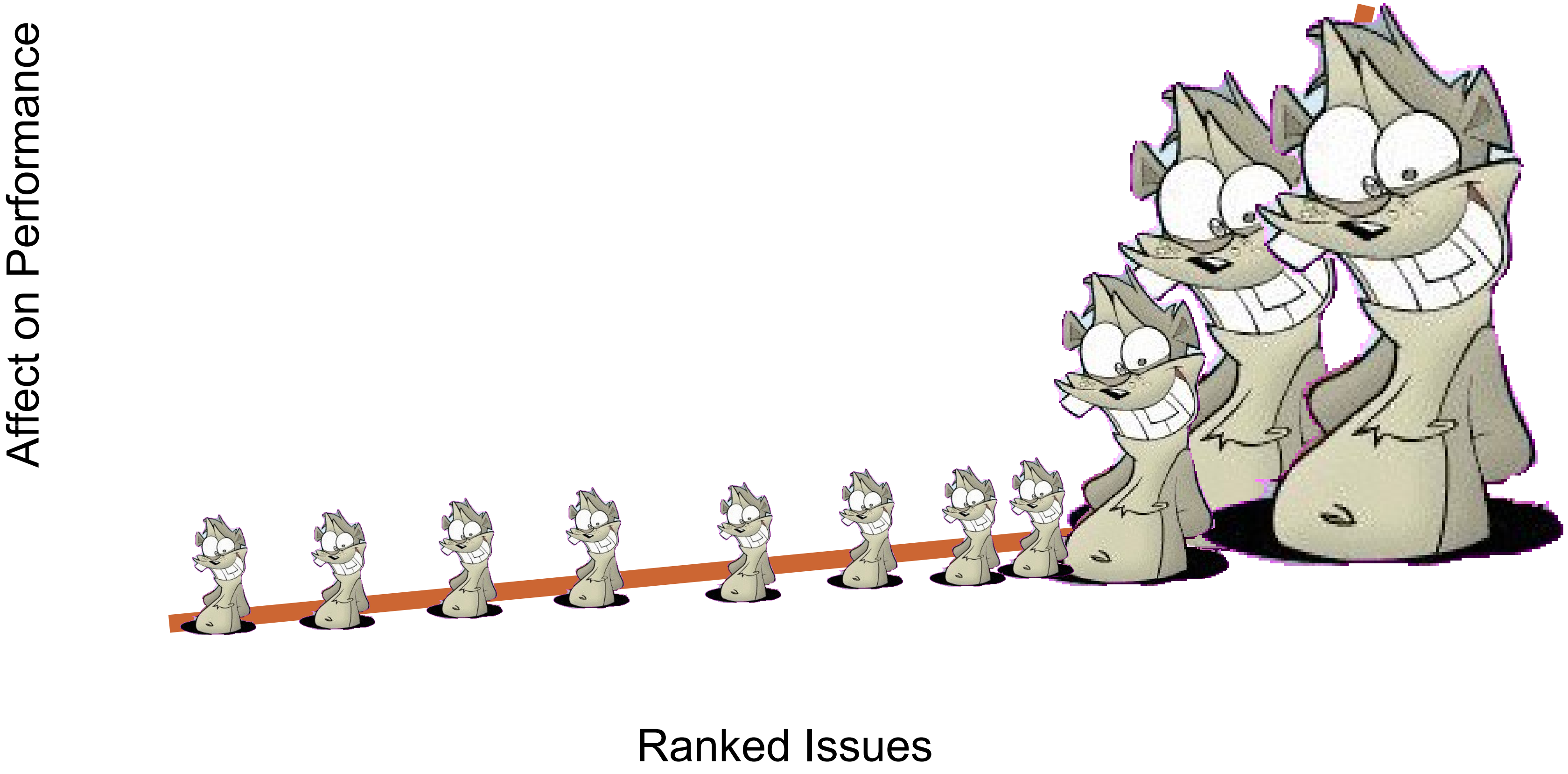
Rules of Whack-a-Mole

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

The Hockey Stick



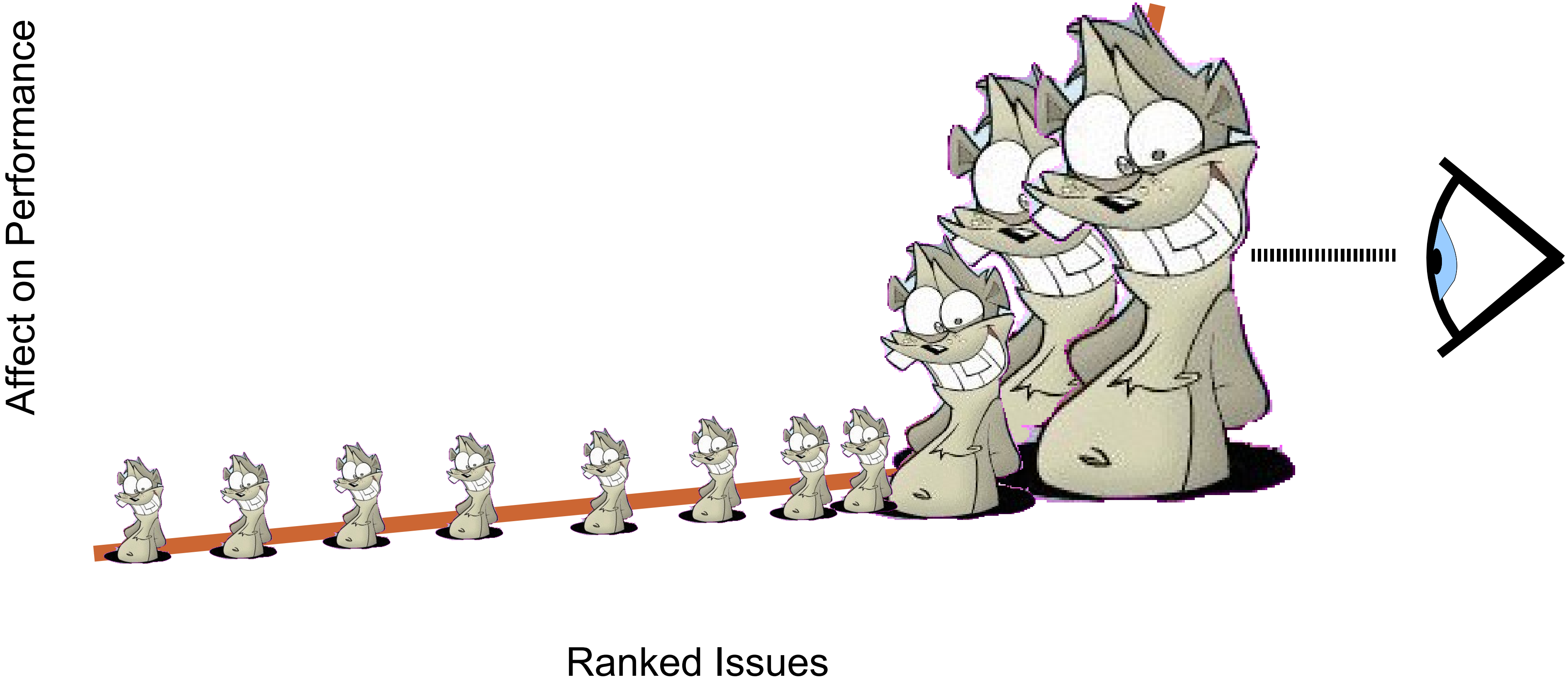
The Hockey Stick



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

The Hockey Stick



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 the “largest” Mole.

What Color Is My Application?

W ▶ Web Application (Web)

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

O ▶ Online Transaction Processing (OLTP)

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

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"

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

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.

Part 2: Baseline

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

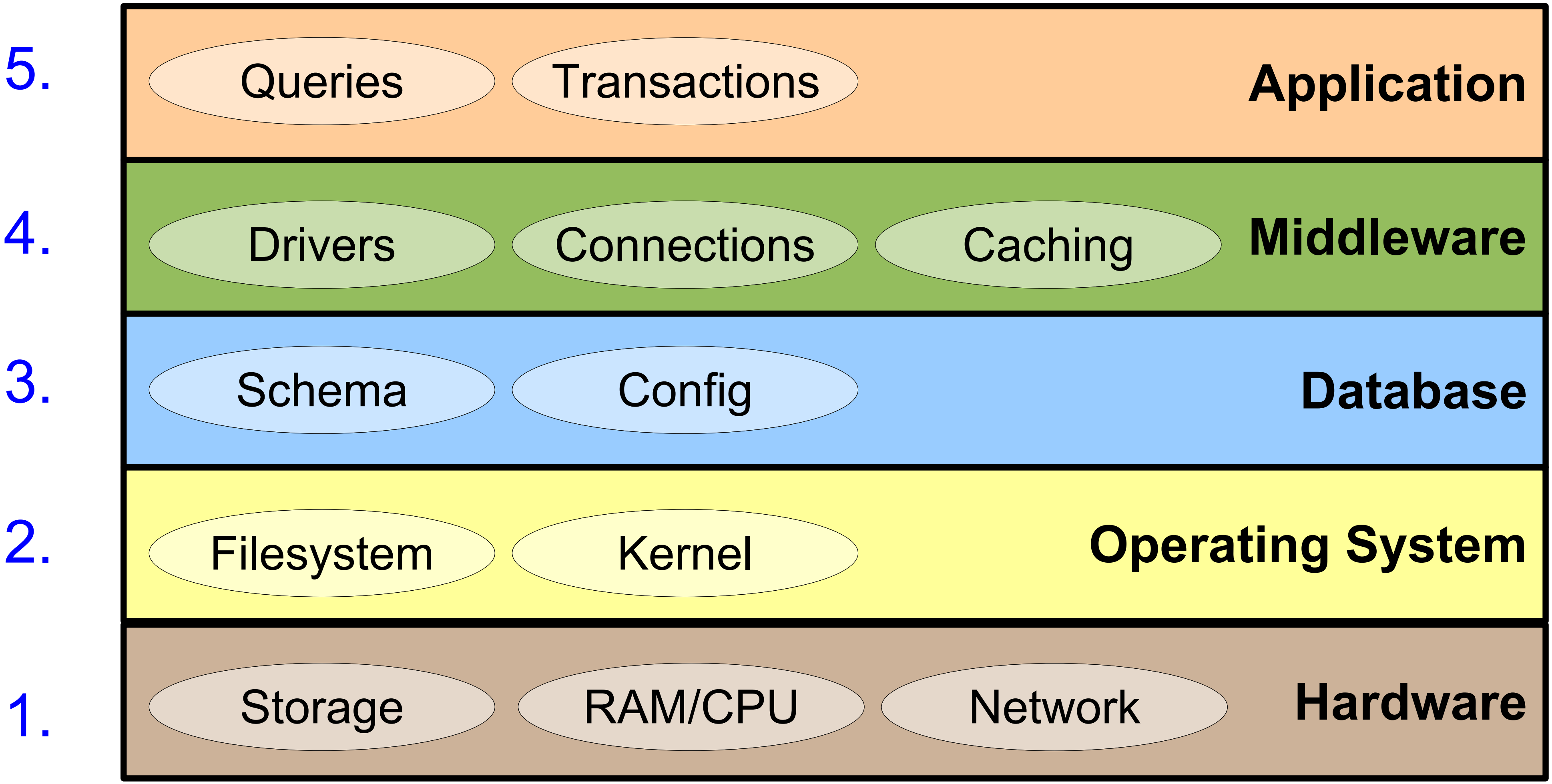
Why Baseline?

- 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. Database Configuration
4. Drivers, Pooling & Caching
5. Application Setup Information

Steps for Baseline



5. Queries Transactions **Application**

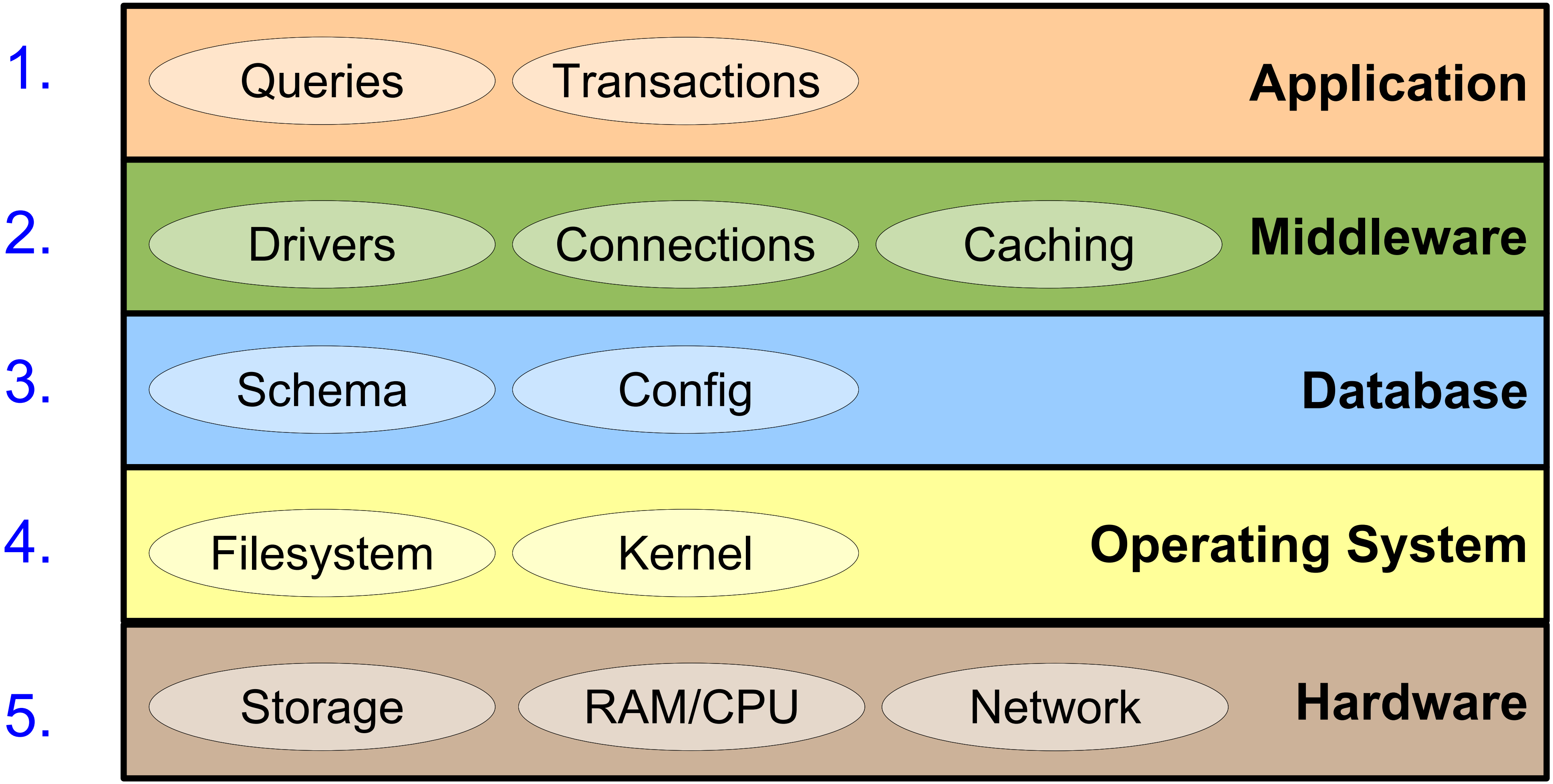
4. Drivers Connections Caching **Middleware**

3. Schema Config **Database**

2. Filesystem Kernel **Operating System**

1. Storage RAM/CPU Network **Hardware**

Steps for Baseline



1. Queries Transactions **Application**

2. Drivers Connections Caching **Middleware**

3. Schema Config **Database**

4. Filesystem Kernel **Operating System**

5. Storage RAM/CPU Network **Hardware**

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

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

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

Operating System Baseline

- Filesystem
  - gather data
    - filesystem type, partitions
    - locations of files for OS, Database, other apps
    - filesystem settings
  - baseline
    - move transaction log 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

Operating System Baseline

- OLTP Server running on Solaris 10
  - Updated to Update3
    - 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

Database Baseline

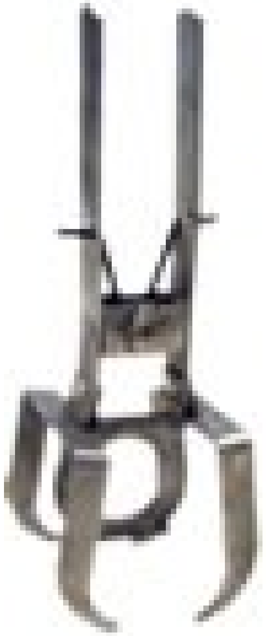
- Gather Data
  - schema
    - tables: design, data size, partitioning, tablespaces
    - indexes
    - stored procedures
  - configuration settings
    - ask about any non-defaults
  - maintenance
    - have maintenance routines been run?
    - when and with what settings?

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
    - 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

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: Database

- database admin views, DTrace
  - minimally invasive, fast
  - give you more internal data about what's going on in the DB realtime
  - let you spot schema, query, lock problems
- Database query log
  - somewhat invasive, slow
  - allows introspection on specific types of db activity
  - compute overall statistics on query, DB load
- Query Analysis
  - troubleshoot “bad queries”
  - for fixing specific queries only

Types of Tools: Application

- 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, DTrace
  - sometimes your performance issue is a genuine software bug

Part 3a: Operating System Tools

ps

- lets you see running processes
  - gives you an idea of concurrent activity & memory/cpu usage
  - lets you spot hung and long-running statements/connections

mpstat

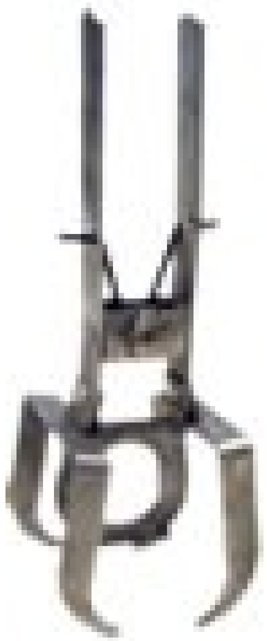
- see CPU activity for each CPU
  - find out if you're CPU-bound
  - see if all CPUs are being utilized
  - detect context-switch issues

vmstat, free

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

iostat

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



Part 3b: Benchmarks

Benchmarks vs. Microbenchmarks

- Benchmarks
  - work out multiple areas of performance
  - require time, effort, hardware to run
  - create reproduceable results
  - create comparable results
- Microbenchmarks
  - work out one area of performance
  - quick & easy to run
  - results may not be reproduceable or comparable

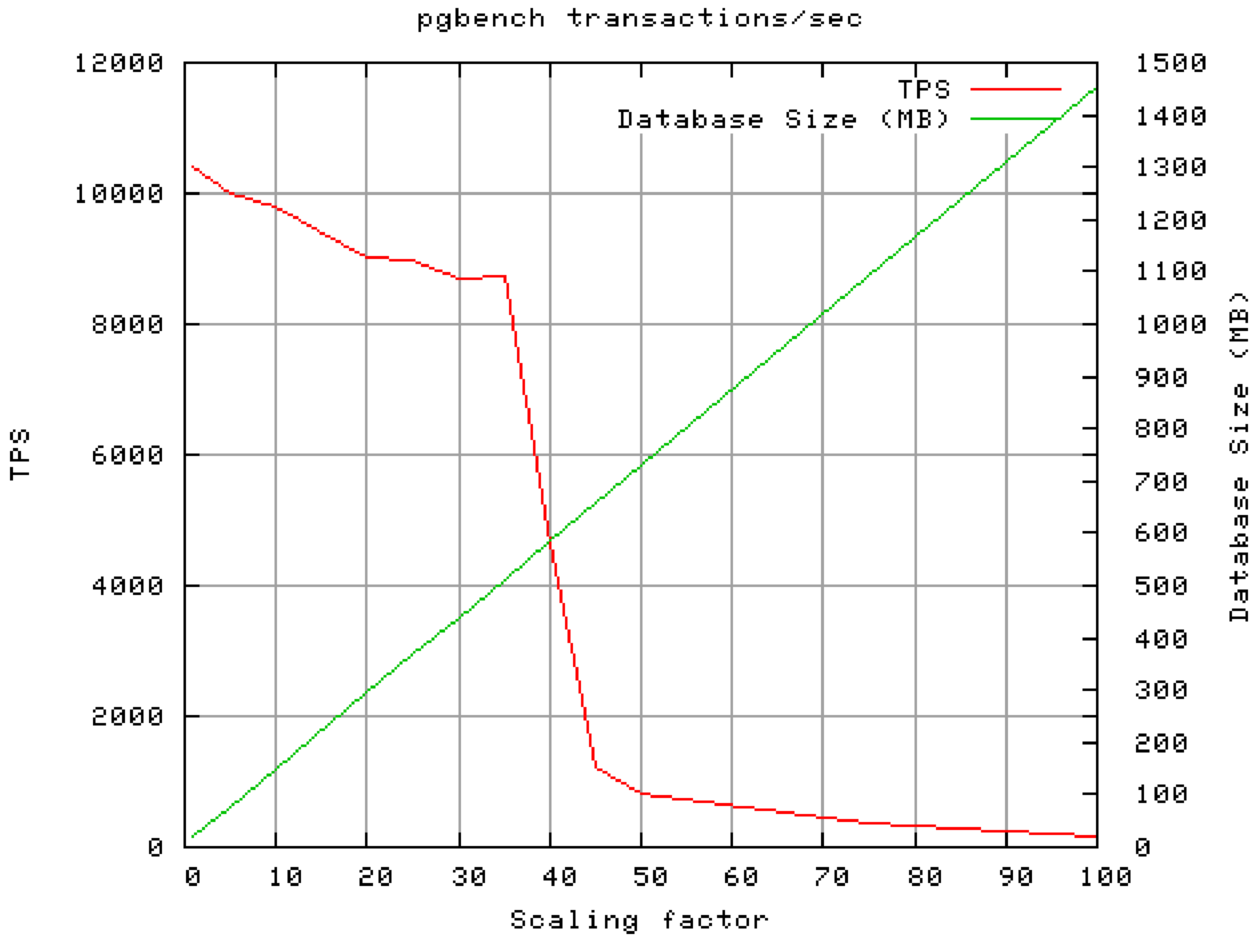
Microbenchmarks: bonnie++

- Filesystem performance test
  - see I/O throughput & issues
  - check seek, random write speeds

Database Microbenchmarks

- pgbench/Wisconsin/TPCB
  - tests mostly I/O and connection processing speed
  - doesn't test locking, computation, or query planning
  - results sometimes not reproducable
  - mostly useful to prove large OS+HW issues
    - *not* useful for fine performance tuning
- OSDB/PolePosition
  - tests specific database operations
  - useful to find specific queries/operations to avoid
  - not useful for general performance tests

Benchmarks: pgbench



Thanks to Greg Smith for this graph!

Benchmarks: Serious

- Use serious benchmarks only when you have a spare systems, or 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
- Each real benchmark tests a different workload
  - so pick the one closest to yours

Benchmarks: Serious

- DBT Benchmarks
  - Serious OLTP benchmark
    - based on TPCC
    - reproduceable results, works out a lot more of the system
    - complex & time-consuming to set up, run
- DBT3, DBT5
  - new OLTP and DW benchmarks
- Others being developed
  - web2.0
  - EAstress

Part 4:
Hunting Moles

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?

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

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
    - SMP scalability limits

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: Hunting Moles Examples

Slow DW

- Setup
  - Data warehousing application
  - Both bulk loads and large reporting queries were very slow
  - CPU and RAM were OK, and I/O seemed underused
    - except it never got above a very low ceiling
- The Hunt
  - used dd, bonnie++, iostat to check I/O behavior
  - throughput of JBOD was *much* slower than internal disk
  - compared with similar system by another vendor
- The Whack
  - the RAID card used in that model was defective, replaced

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

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
  - 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
  - Suggested improvements
    - upgrade to a J2EE architecture with better pooling

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()

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

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?

Further Questions

- Josh Berkus
  - josh@postgresql.org
  - josh@pgexperts.com
  - www.pgexperts.com
  - it.toolbox.com/blogs/database-soup
- More Advice
  - www.postgresql.org/docs
  - www.planetpostgresql.org
  - irc.freenode.net
    - #postgresql

Special thanks for borrowed content to:
www.MolePro.com for the WhackaMole Game
Greg Smith for pgbench and bonnie++ results