ORACLE®

Migrating On-Premises Applications to the Cloud: Examining the Connection Strategy

Are connections still important?

ORACLE OPEN WORLD

October 1–5, 2017
SAN FRANCISCO, CA

Graham Wood
Andrew Holdsworth
Real-World Performance
Server Technologies

October, 2017

ORACLE

ORACLE
REAL-WORLD PERFORMANCE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

How Does Your Organization Do Performance ?

- Conventional
  - Focus is on “Good Enough” or “What the Business Needs”
  - Process Orientated/Part of QA
  - Spends most the time on Platform Tuning Issues
  - Only changes things within limited scope
  - Bottom up tuning approach
  - Looking for incremental gains

- Real-World
  - Focus on excellence and what the hardware and software can do
  - Innovate excellent performance and add intellectual property to your code
  - Everything is within scope
  - Holistic top down approach
  - Focus on orders of magnitude gains

RWP @Demoground

- Moscone West SOA-161
- Discuss your performance challenges with RWP staff
- Bring your AWR/ADDM/ASH/SQL* Monitor for analysis by RWP

We’ve Been Here Before

- RWP has been talking to customers about connection strategies for years
- Why are we still doing it?
- 85% of OLTP escalations are still caused by too many processes
- So we will keep talking about it and how you can avoid your system becoming one of our escalation

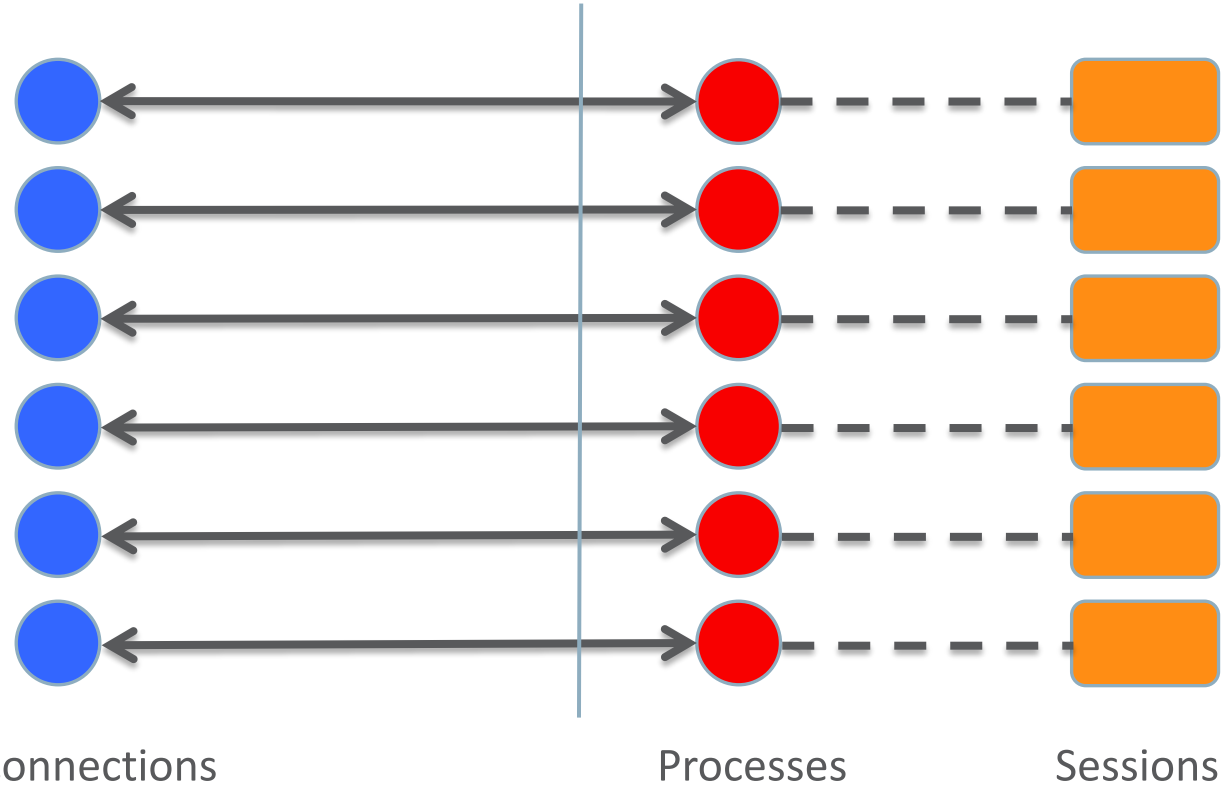
We’ve Been Here Before

- Moving to the Cloud makes architecture even more important
- Cloud provides large amounts of readily available resources
- Pressure to achieve more with less
- Legacy systems
  - often heavily handcrafted and fragile
  - often vastly over provisioned
  - How well can those systems use the power of the Cloud
- Consolidation increases system density
- Connections often the limiting factor

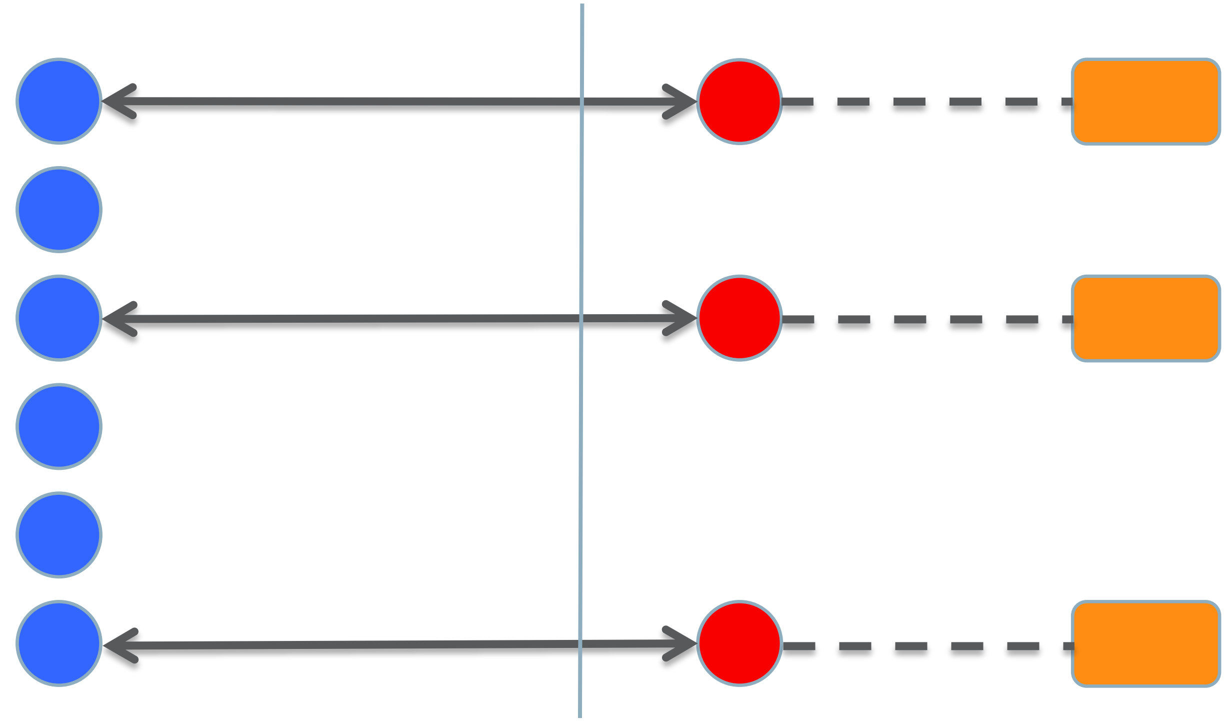
What Connection Architectures are Available?

- Client side
  - Dedicated connection
  - Connection Pool (e.g. UCP)
- Database side
  - Direct connection
  - Shared server
  - Database Resident Connection Pool (DRCP)

Dedicated Server (Persistent Model)



Dedicated Server (Acquire/Release Model)

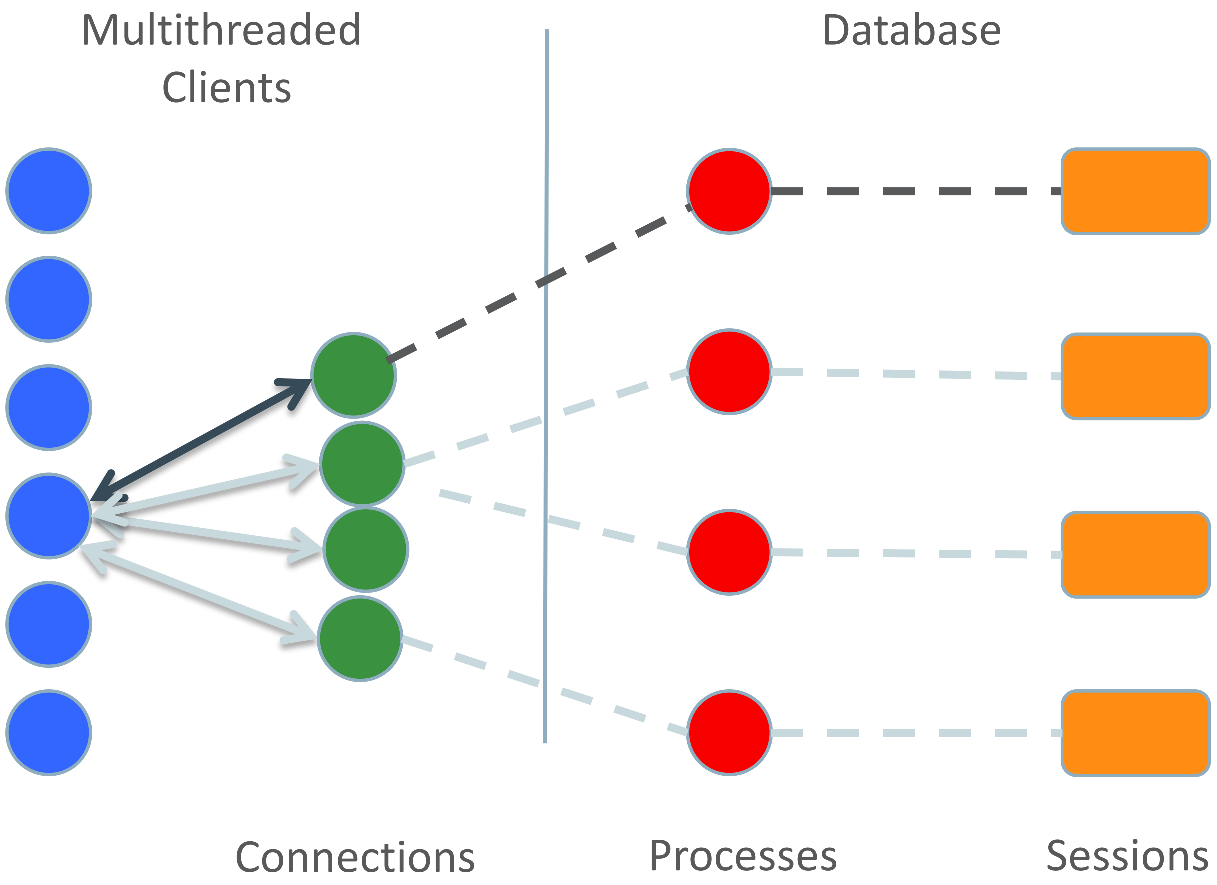


Connections

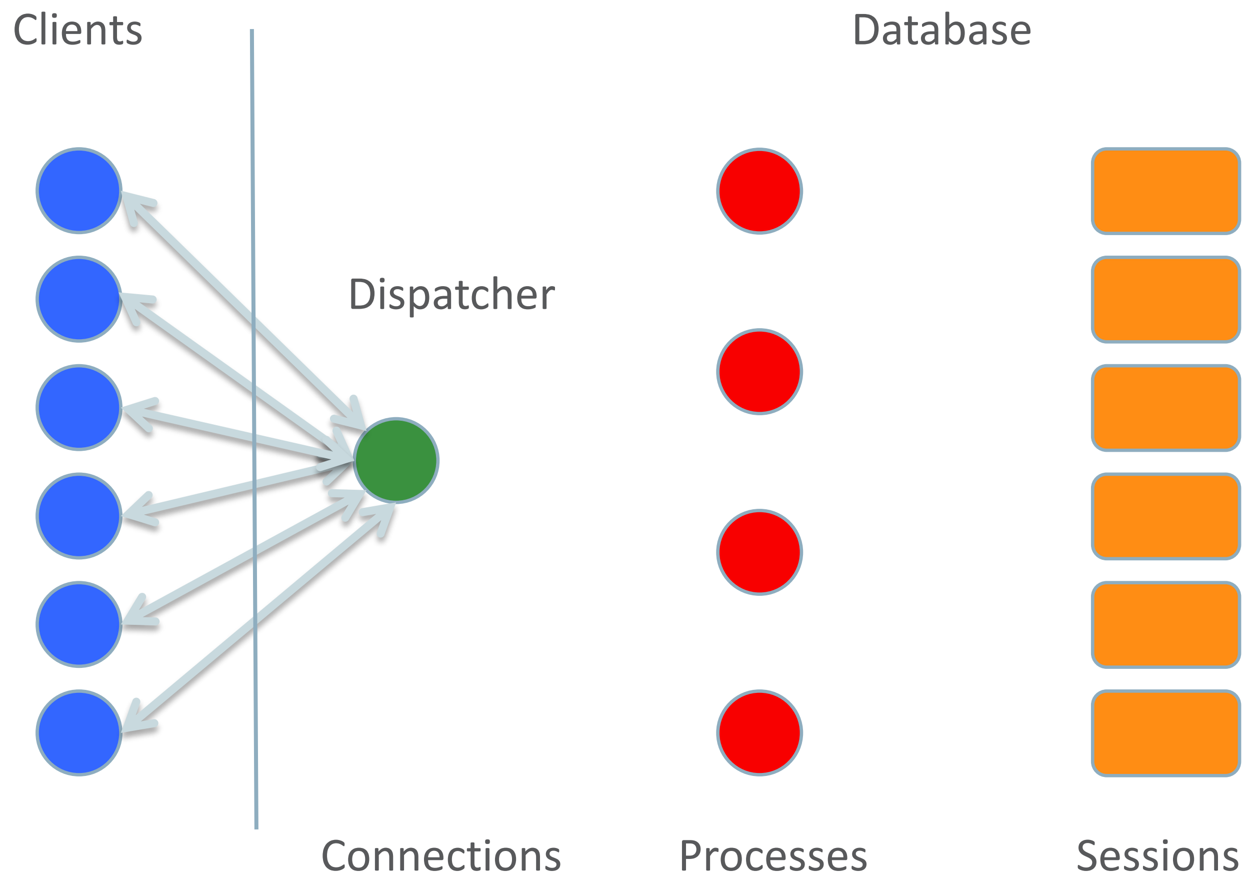
Processes

Sessions

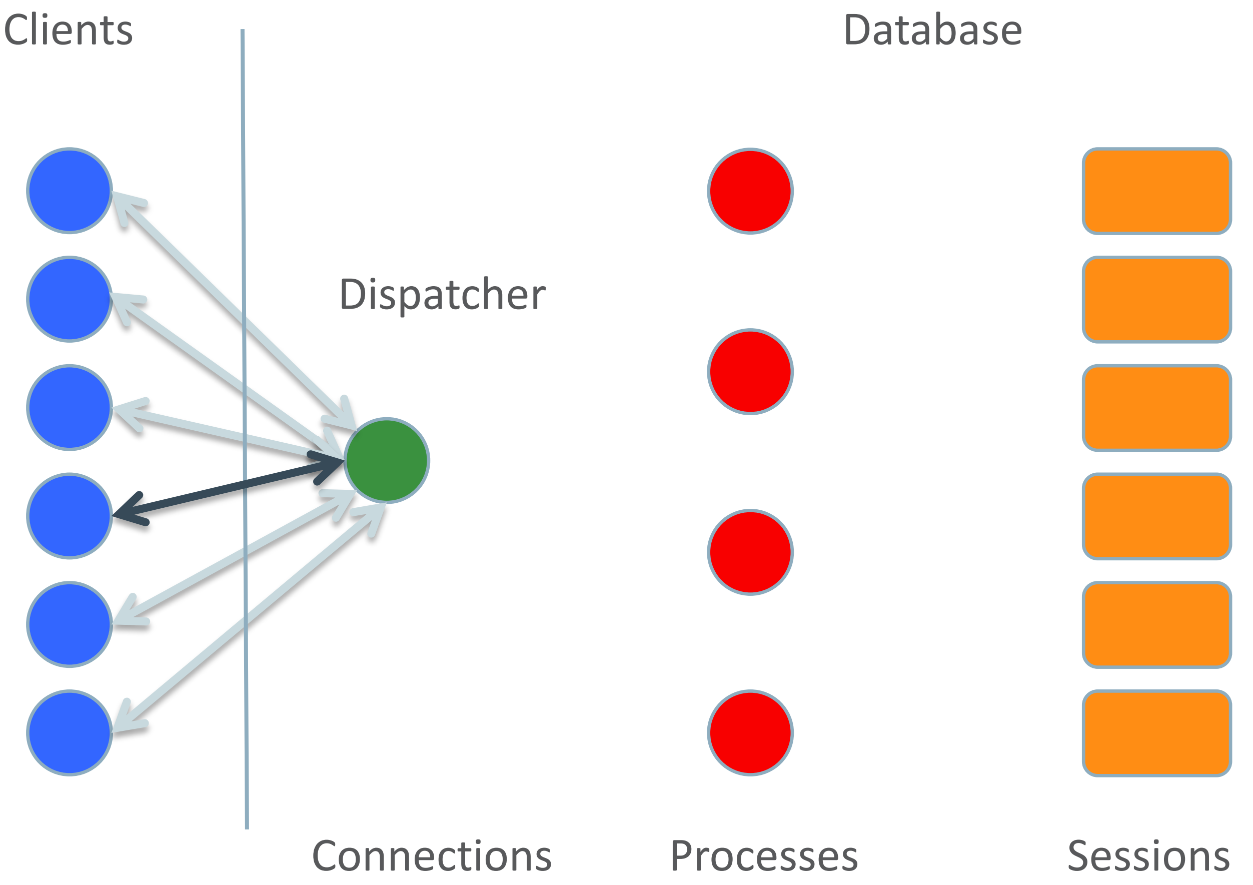
Client-side Connection Pool



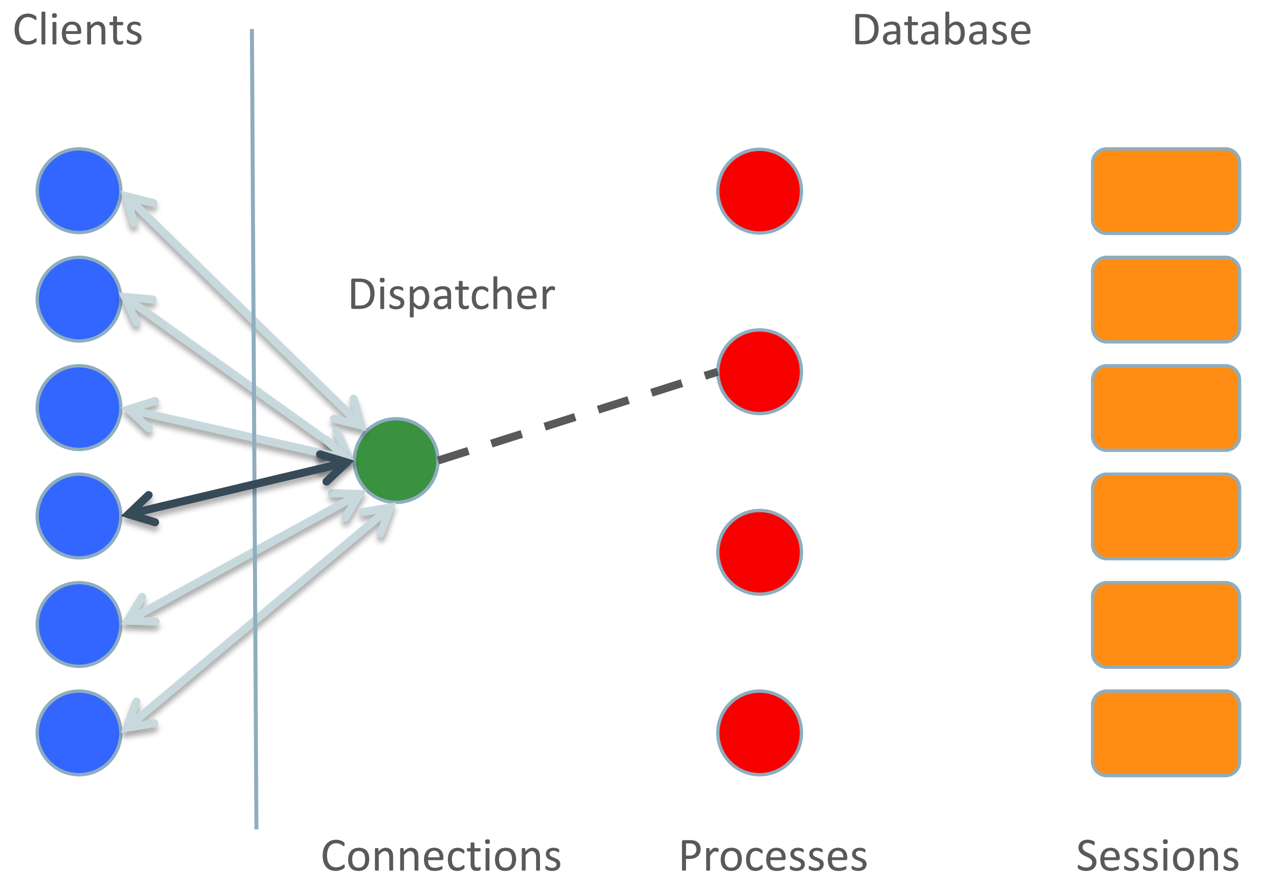
Shared Server



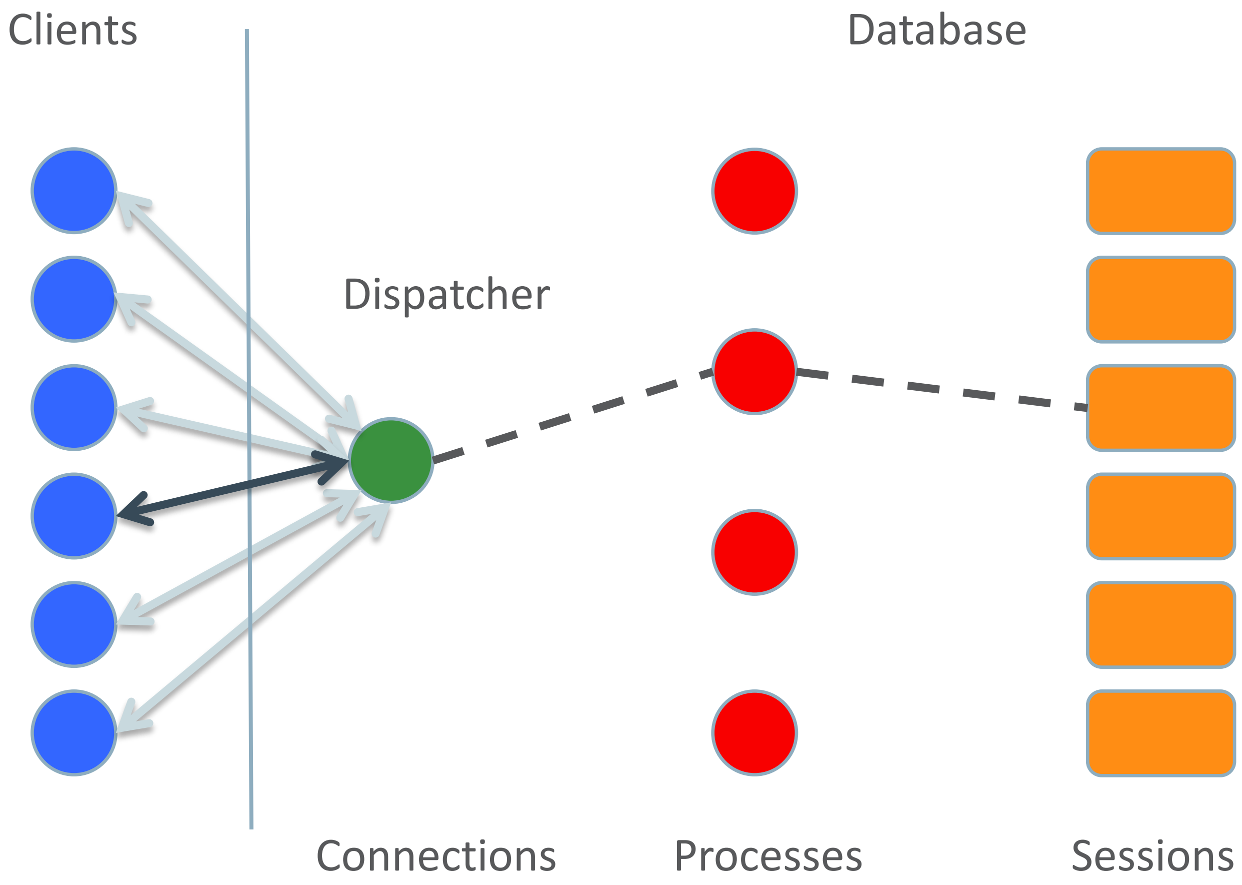
Shared Server



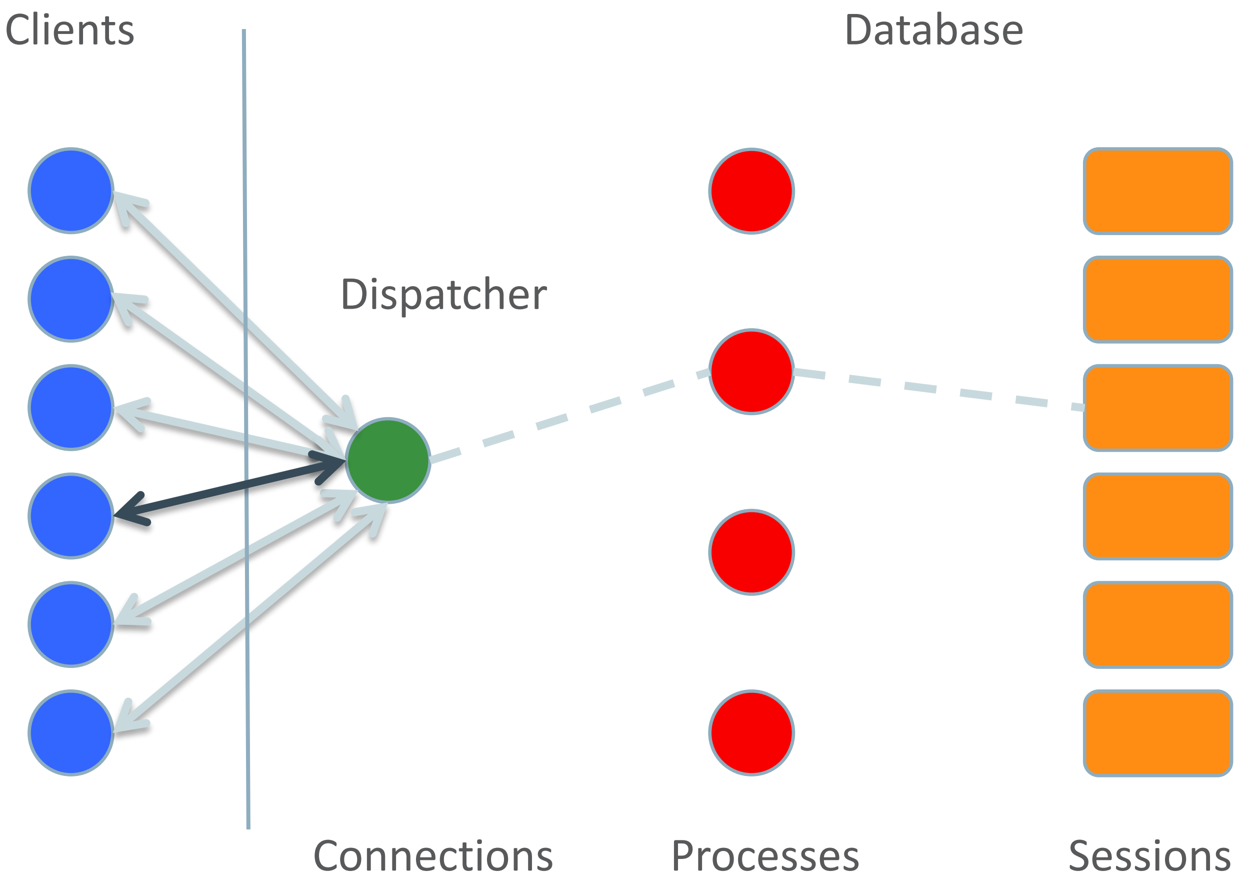
Shared Server



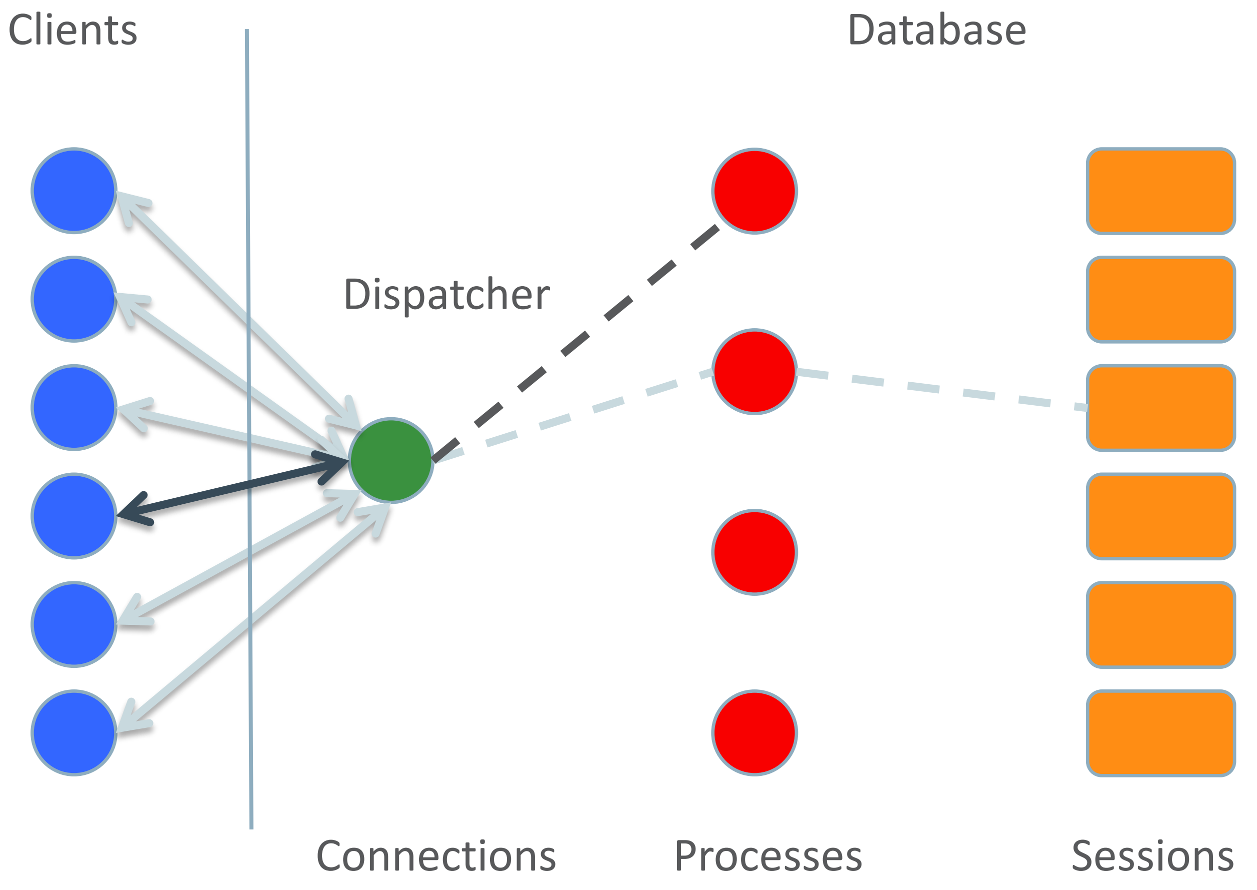
Shared Server



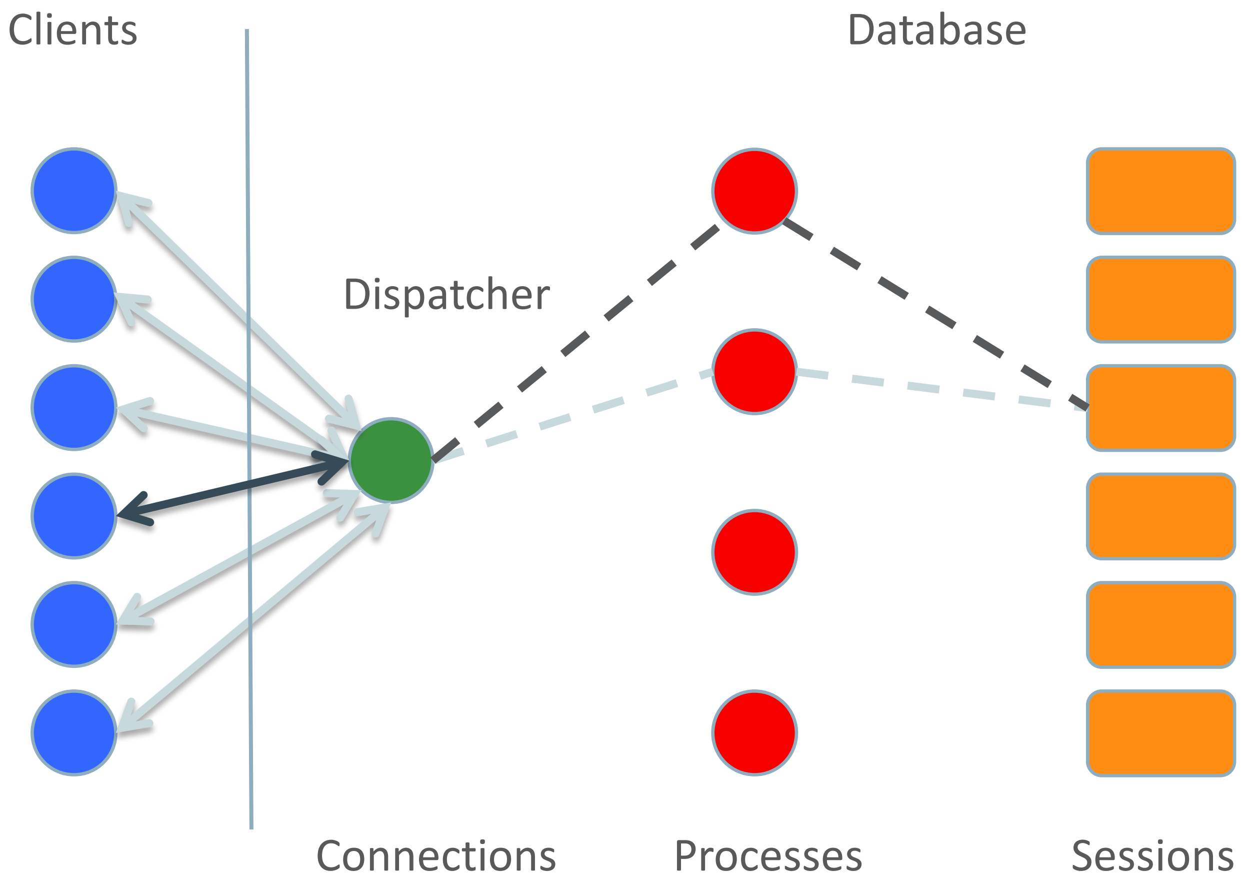
Shared Server



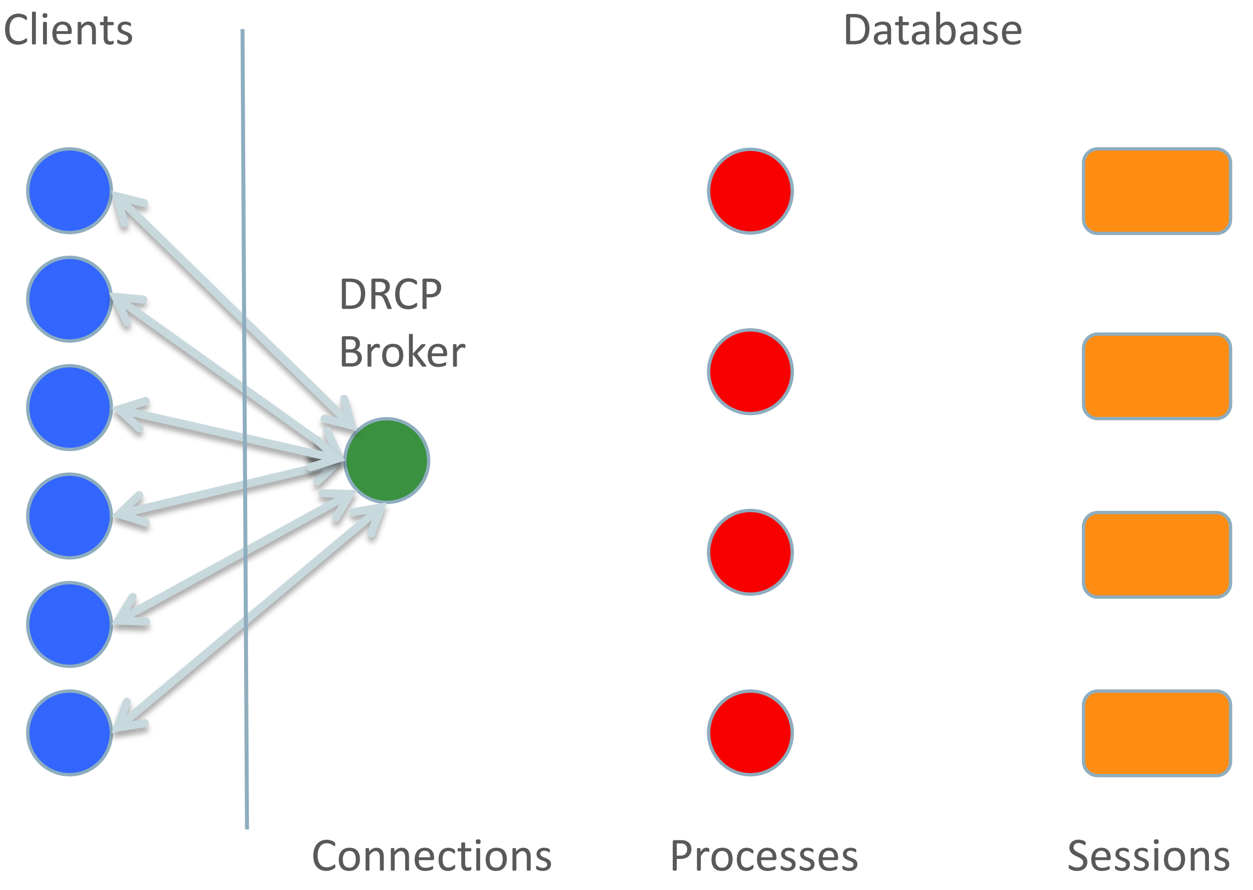
Shared Server



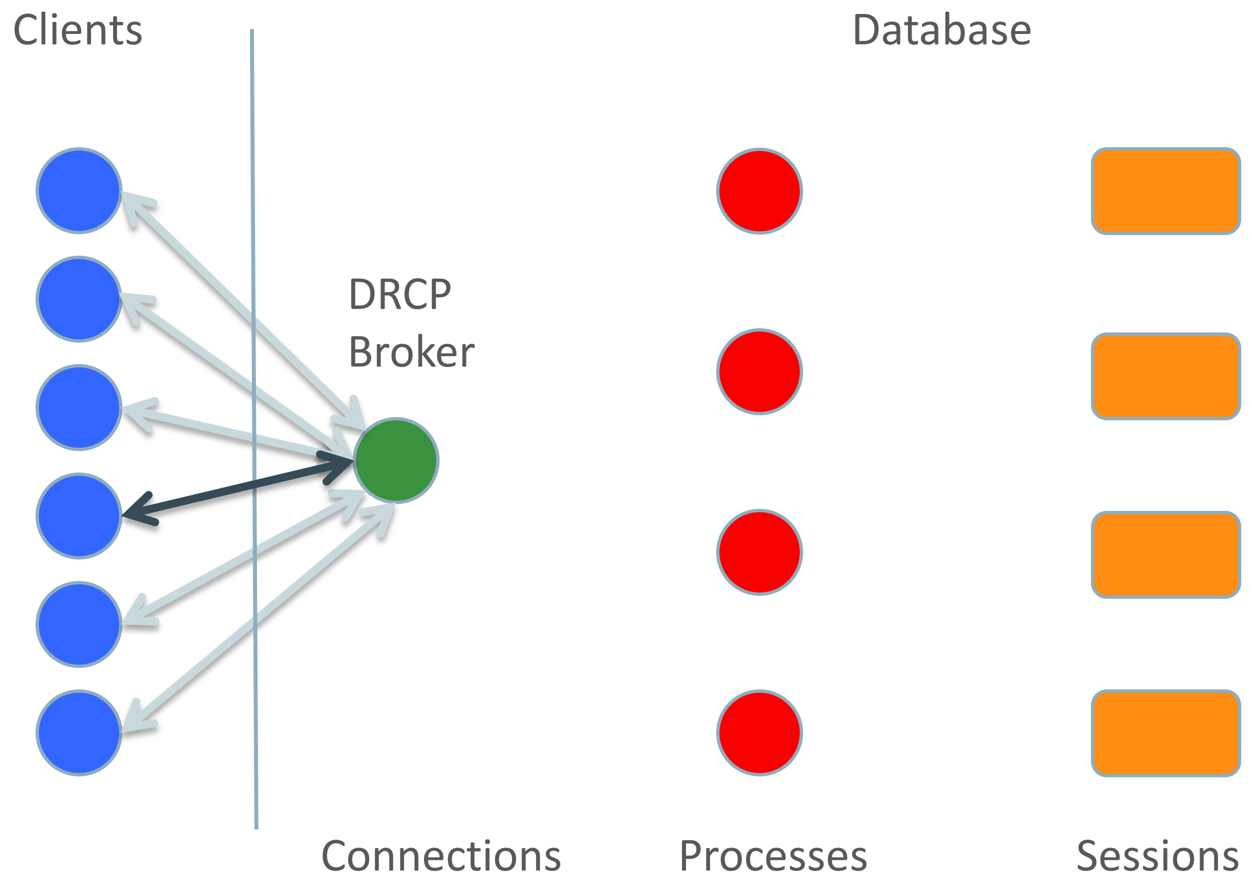
Shared Server



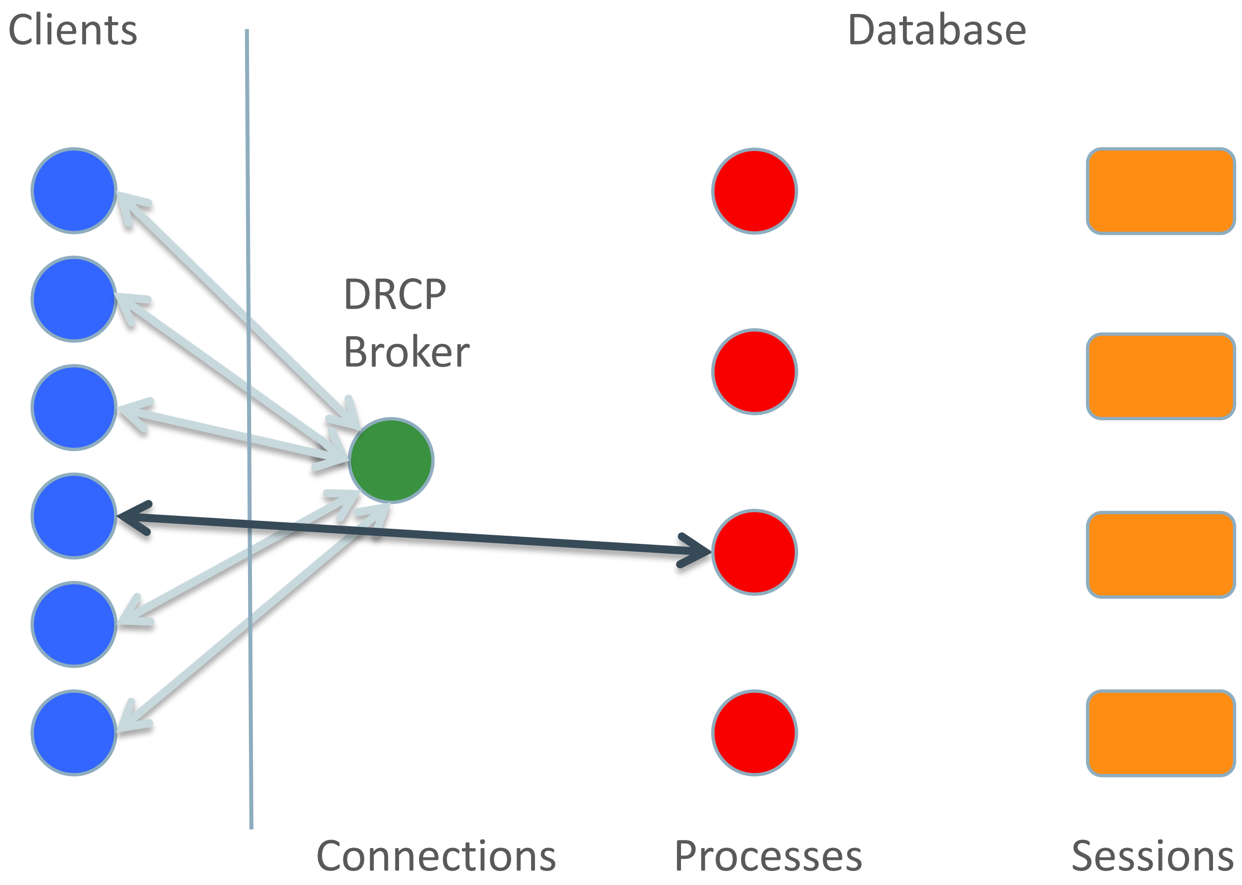
DRCP


Clients
Database
DRCP
Broker
Connections
Processes
Sessions

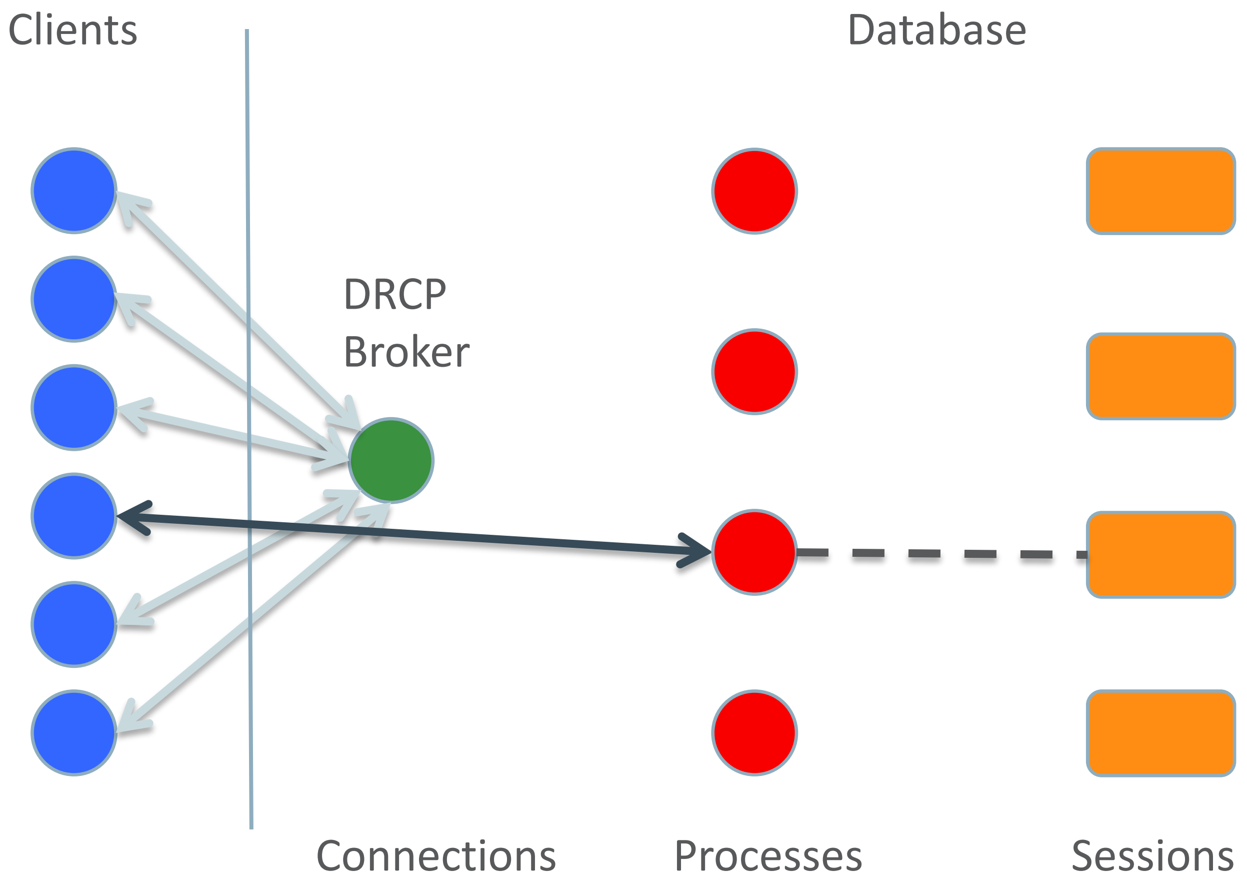
DRCP


Clients
Database
DRCP
Broker
Connections
Processes
Sessions

DRCP


Clients
Database
DRCP
Broker
Connections
Processes
Sessions

DRCP



Clients

Database

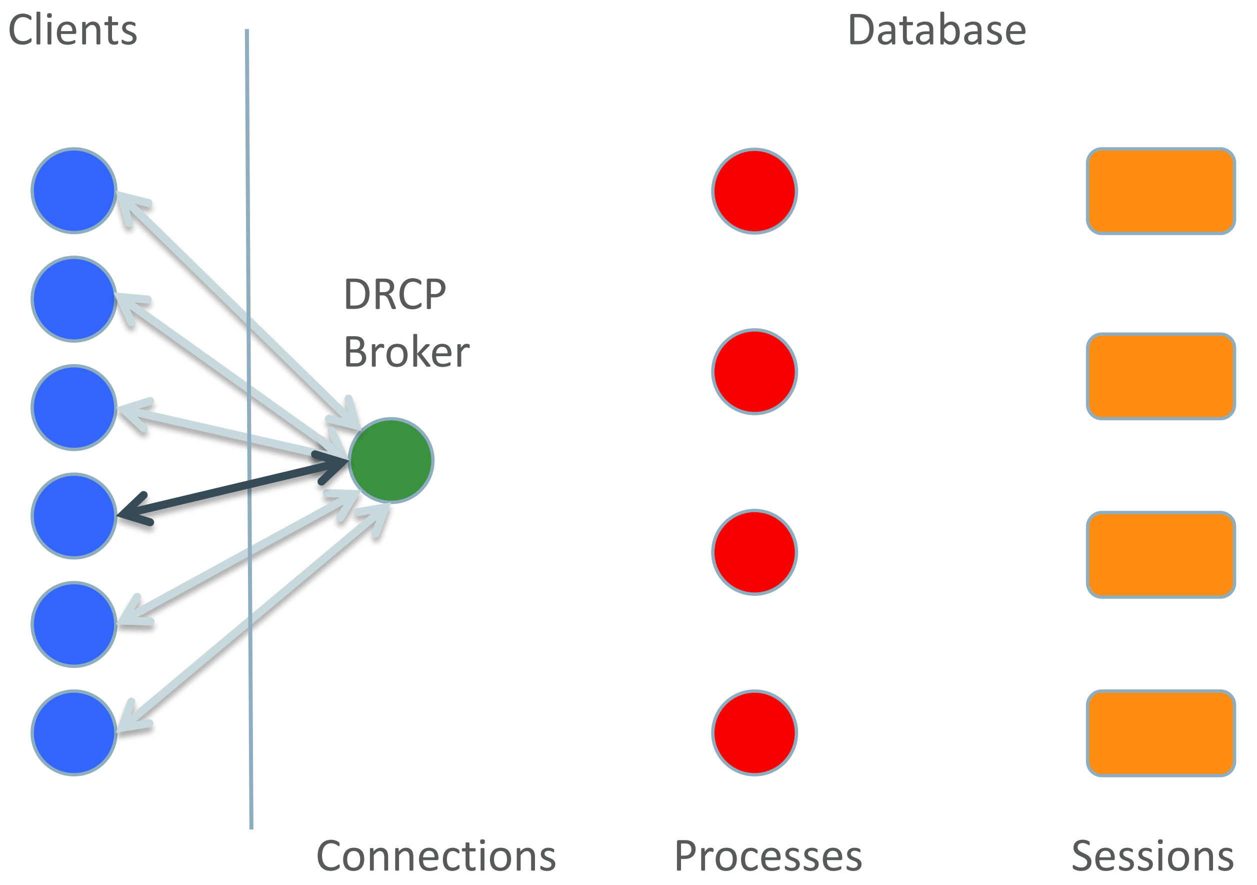
DRCP Broker

Connections

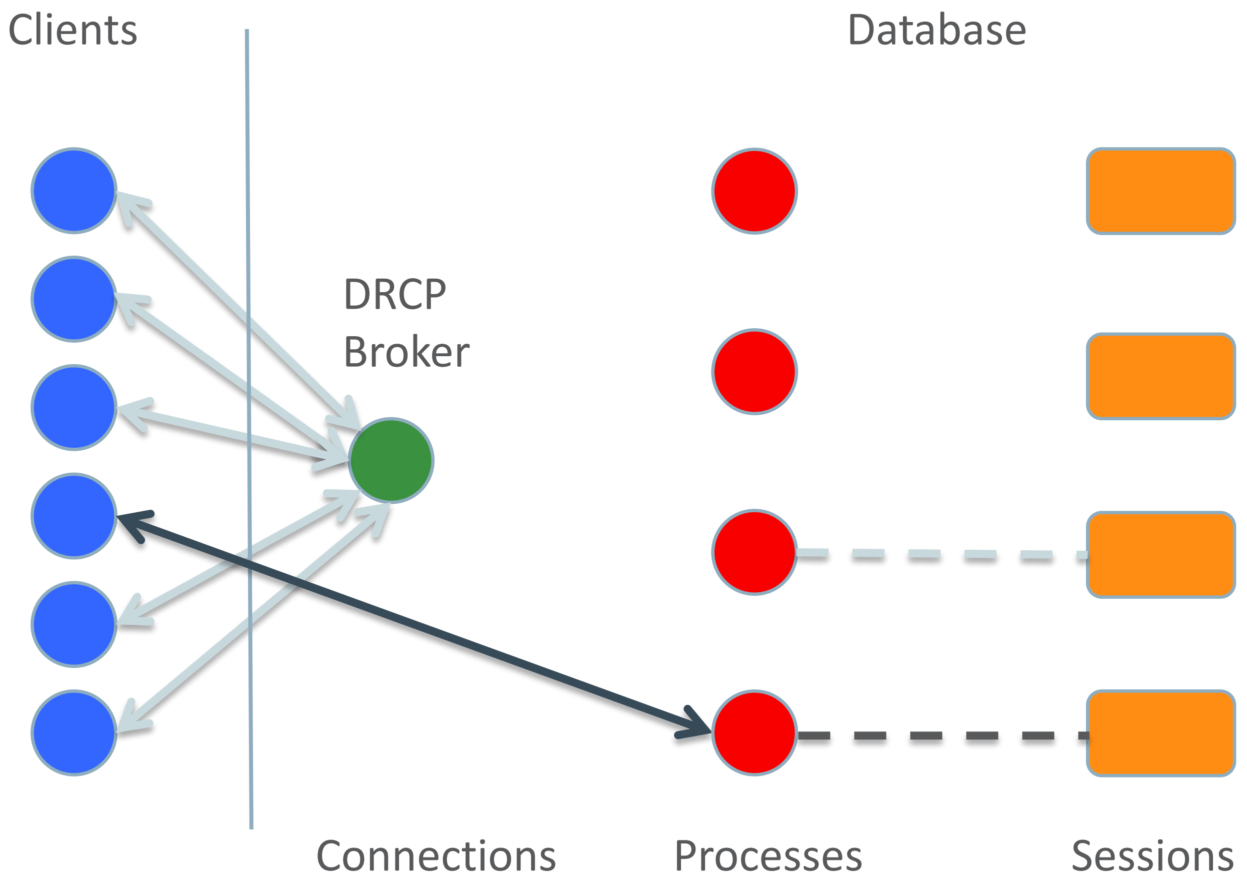
Processes

Sessions

DRCP


Clients
Database
DRCP
Broker
Connections
Processes
Sessions

DRCP


Clients
Database
DRCP
Broker
Connections
Processes
Sessions

What We Recommend

In the Real World

- Number of connections directly related to available CPU resources
- Use fixed size connection pools
- Typically in the range 1-10 connections per CPU core (not thread)
- Results in a sensible number of processes running on the server

What We See

In the Real World

- Large number of dynamic connection pools
  - X app servers each running Y JVMs, each with connection pools size capped at Z
    - X*Y*Z is a large number often 10,000+
    - Middle tiers cause increased number of connections rather than reduction
  - Connections grow over time because of session, cursor, lock leaks from application
- Large servers running at ~10% utilization, grossly overprovisioned

What We Hear

In Escalations

- It can’t be wrong because it works most of the time
- Most of our sessions are idle most of the time, so surely it’s not a problem
- We use dynamic connection pools so the application tier never has to wait to access the database
- We scale by adding connections/application servers

What We Hear

In Escalations

- Connections grow over time as load increases
  - Also because of application session, cursor, lock leaks
  - Also because of any issue in any level of stack below the app tier
- It works fine with large numbers of connections, we’ve tested it!
  - In the lab
  - Steady state
  - No errors

Let’s Run Some Tests!

Test Workload

- Simulates Real-World scenarios
  - Web application
  - Multi-tier architecture
  - Application think time can be increased or decreased to change the level of the system load

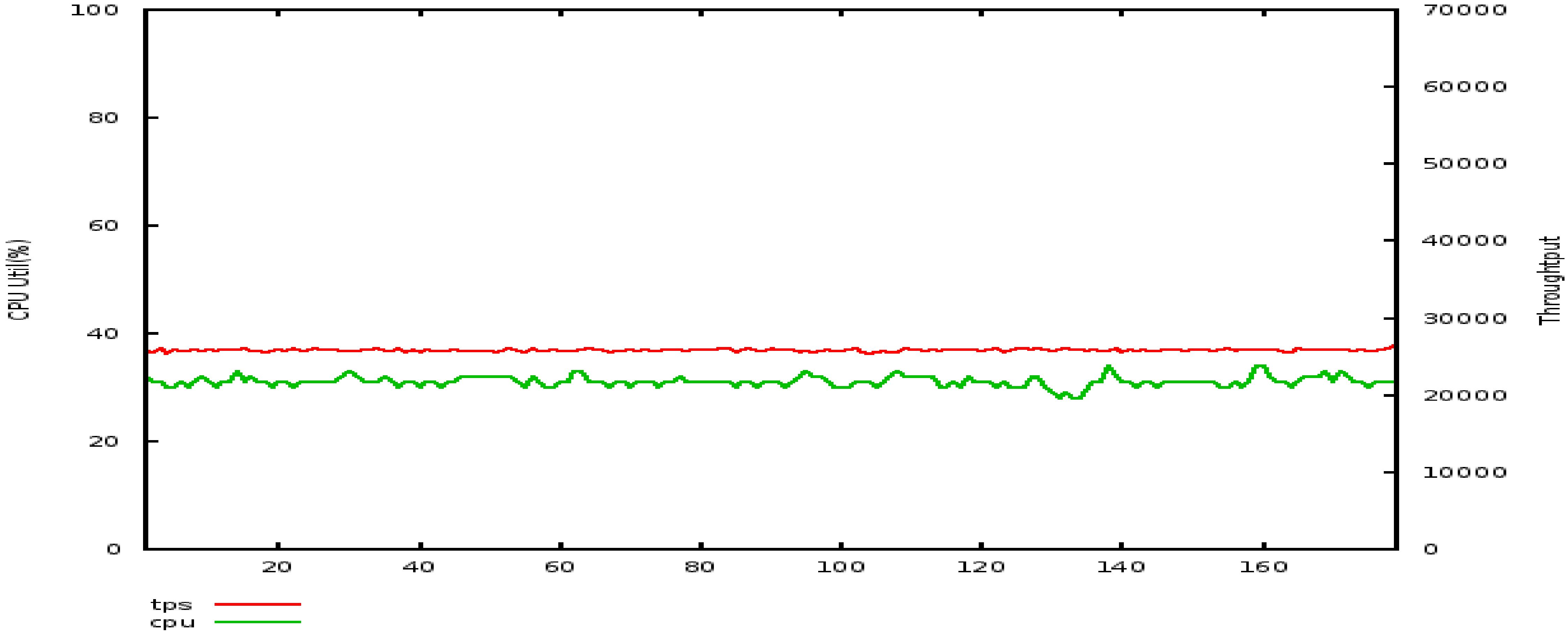
- 1DB server: 36 cores
- 2 APP servers : 36 cores for each
  - 36 JVMs in total

- 10,800 total application server threads

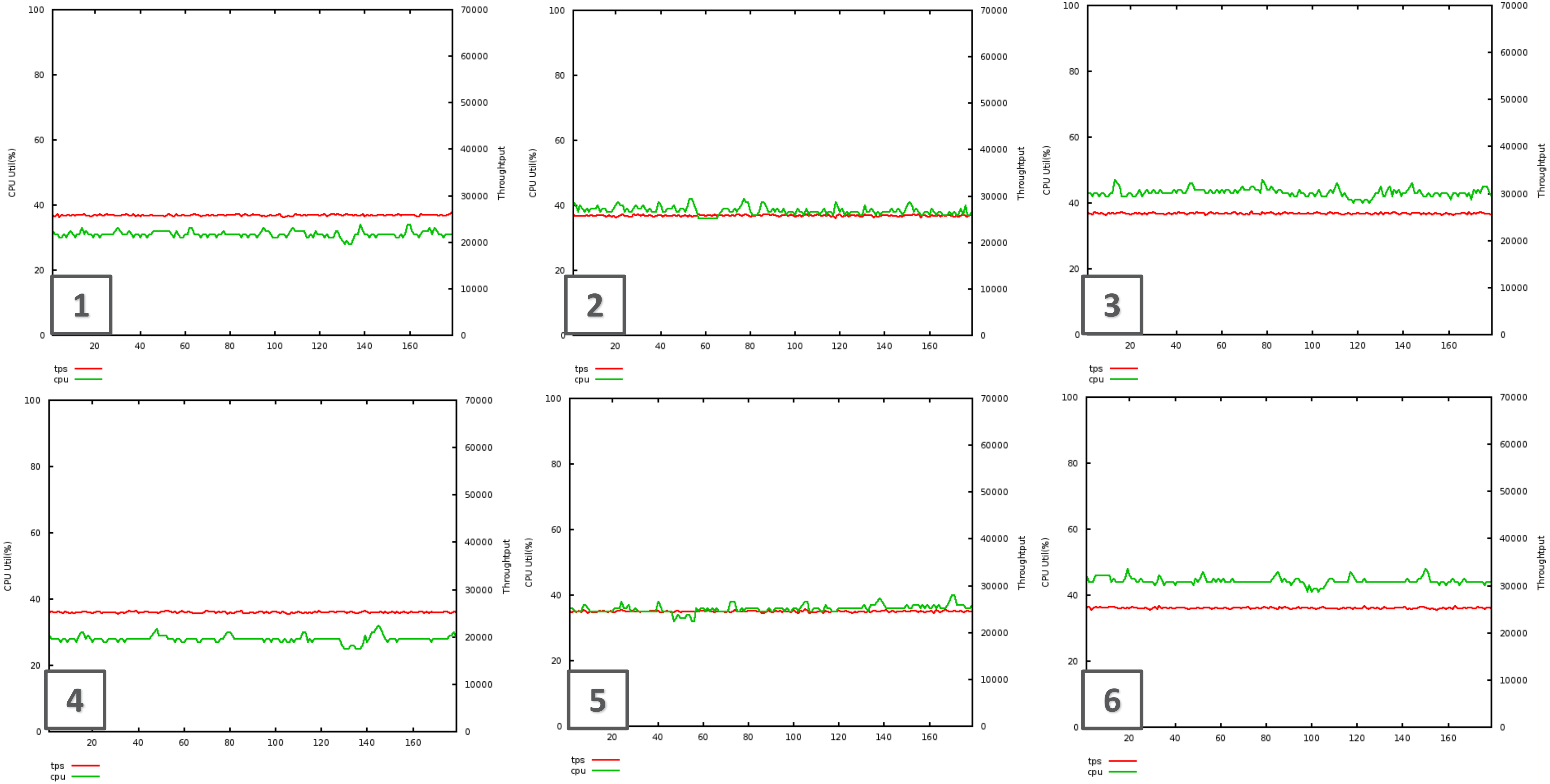
Test Configuration (36 JVM)

No.	Client	DB	Network connections	DB Sessions	Brokers/ Dispatchers	Pooled/ Shared servers
1	Dedicated	Dedicated	10800	10800		
2	Dedicated	Shared Server	10800	10800	16	288
3	Acquire/release	DRCP	10800	288	32	288
4	UCP	Dedicated	288	288		
5	UCP	Shared Server	1152	1152	16	288
6	UCP	DRCP	1152	288	32	288

10,800 Dedicated Connections 500ms Think Time



Graph with steady state workload (36 JVM)



Results with steady state workload (36 JVM)

No.	Client	DB	TPS	DB Server CPU%	App Server CPU%	Response Time (ms)
1	Dedicated	Dedicated	25826	31%	7%	1.9
2	Dedicated	Shared Server	25798	38%	6%	2.3
3	Acquire/release	DRCP	25790	43%	7%	2.2
4	UCP	Dedicated	25228	28%	7%	1.8
5	UCP	Shared Server	24522	36%	7%	2.2
6	UCP	DRCP	25288	44%	9%	2.3

Analysis

- What are these guys talking about!
- Best Performance is 10,800 Direct Connections
- All models work and give highly stable results
- Shared server and DRCP have significant CPU overhead
- Users wouldn’t notice the response time differences
- So we are done then!

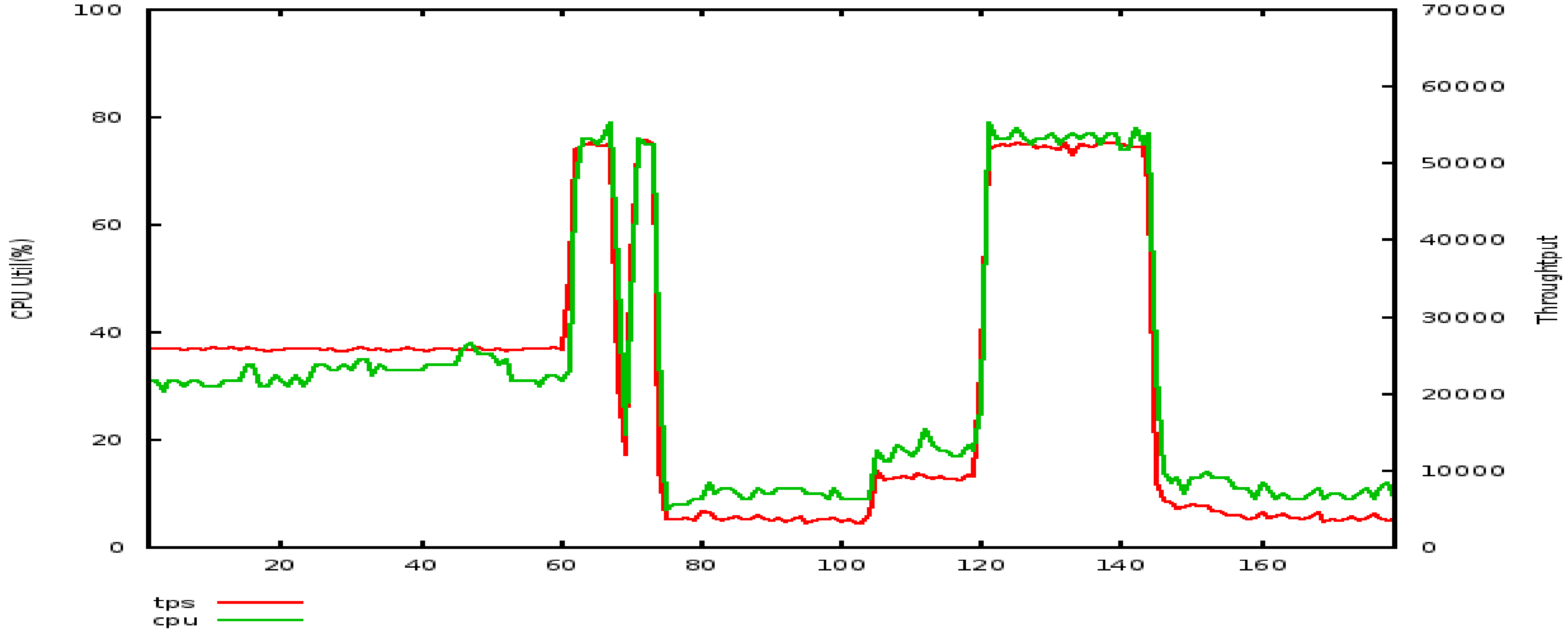
- We just re-ran your testing

Let’s Try That Again

But this time we will add a load surge

- Same configurations
- After one minute of running drop the application think time to double load

10,800 Dedicated Connections 500ms => 240ms after 1 min

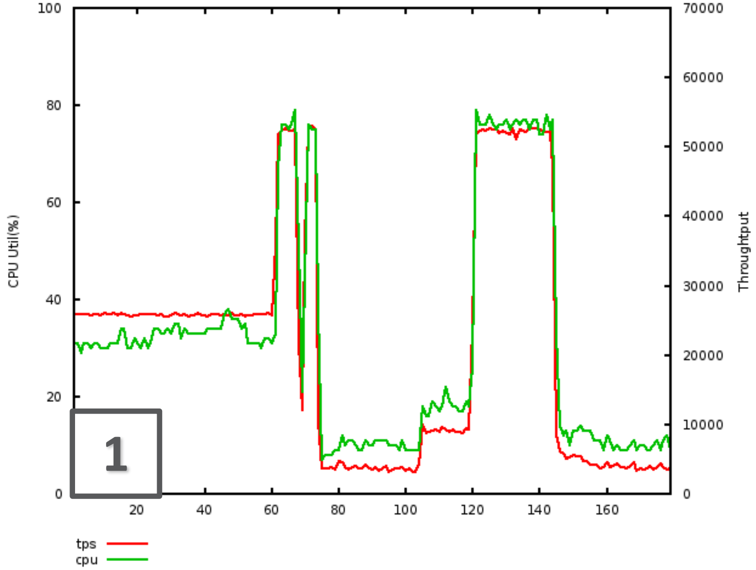


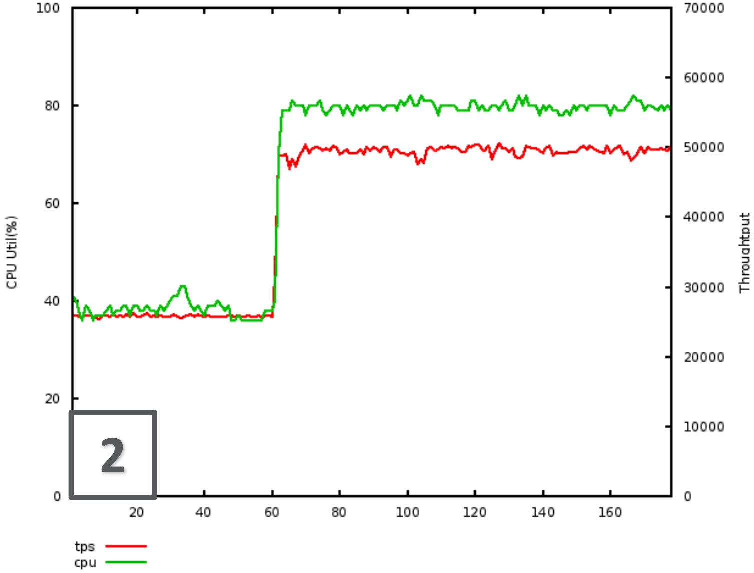
ORACLE® REAL-WORLD PERFORMANCE

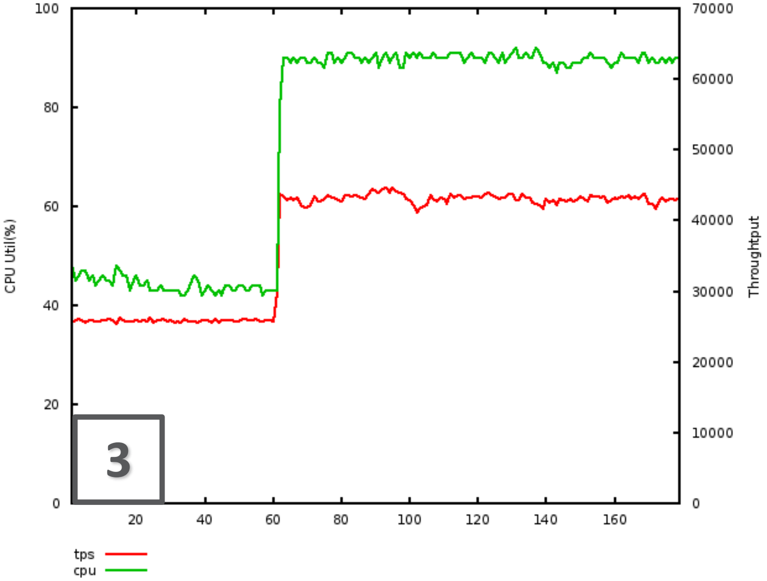
Analysis

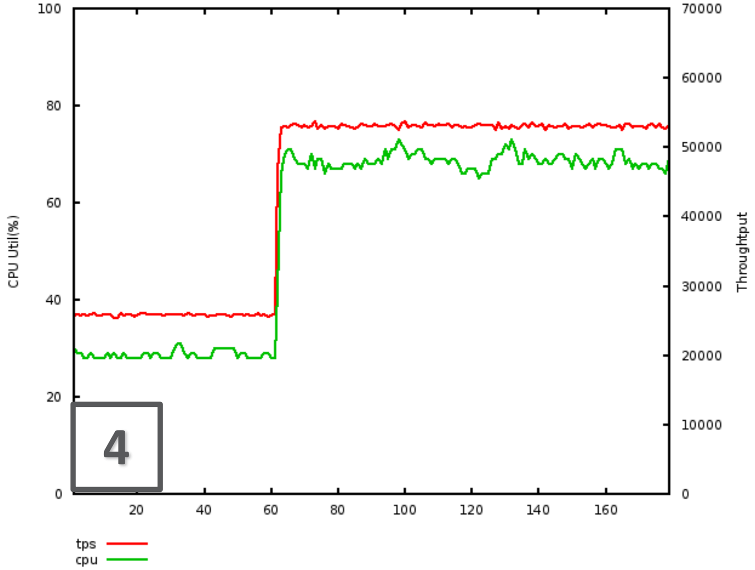
- All those dedicated connections don’t look so good now
- System goes unstable
- Throughput drops
- Let’s try the other runs

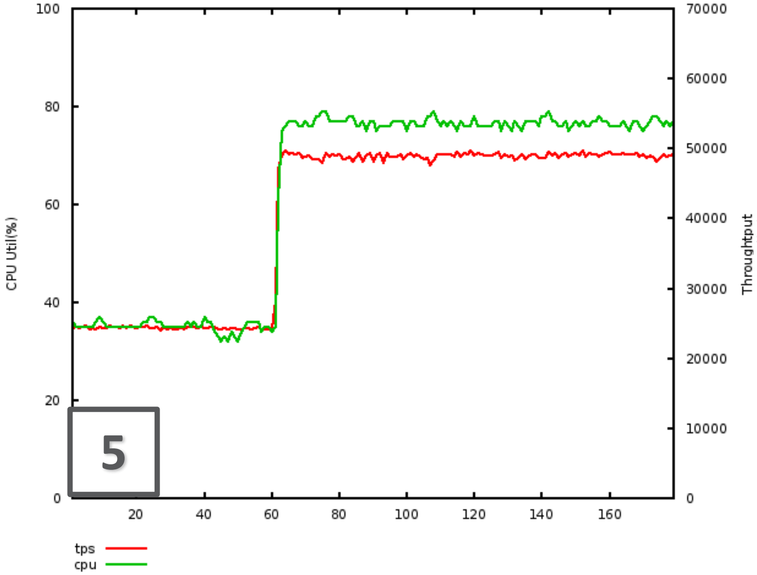
Graph with workload surge (36 JVM)

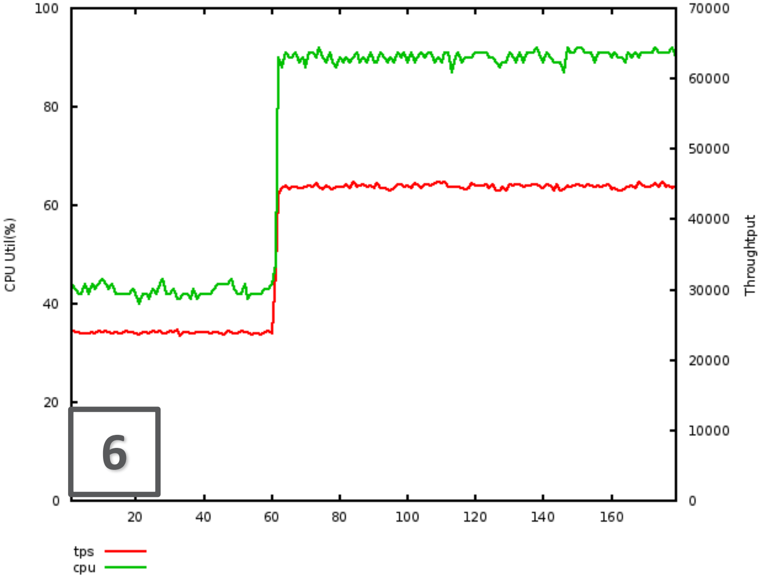












ORACLE

ORACLE
REAL-WORLD PERFORMANCE

Results with workload surge (36 JVM)

No.	Client	DB	TPS	DB Server CPU%	App Server CPU%	Response Time (ms)
1	Dedicated	Dedicated	21220	30%	6%	422
2	Dedicated	Shared Server	41460	66%	12%	13
3	Acquire/release	DRCP	37280	74%	8%	34
4	UCP	Dedicated	43803	55%	11%	3
5	UCP	Shared Server	40642	62%	12%	6
6	UCP	DRCP	37703	74%	14%	17

Analysis

- Apart from dedicated server test all other tests stay stable
- Response time, resource usage and throughput vary dramatically

Conclusion

- Some form of connection pooling either on the client side, the database side or both, is essential to ensure a reliable, stable system
- Most efficient place to do it is in the client

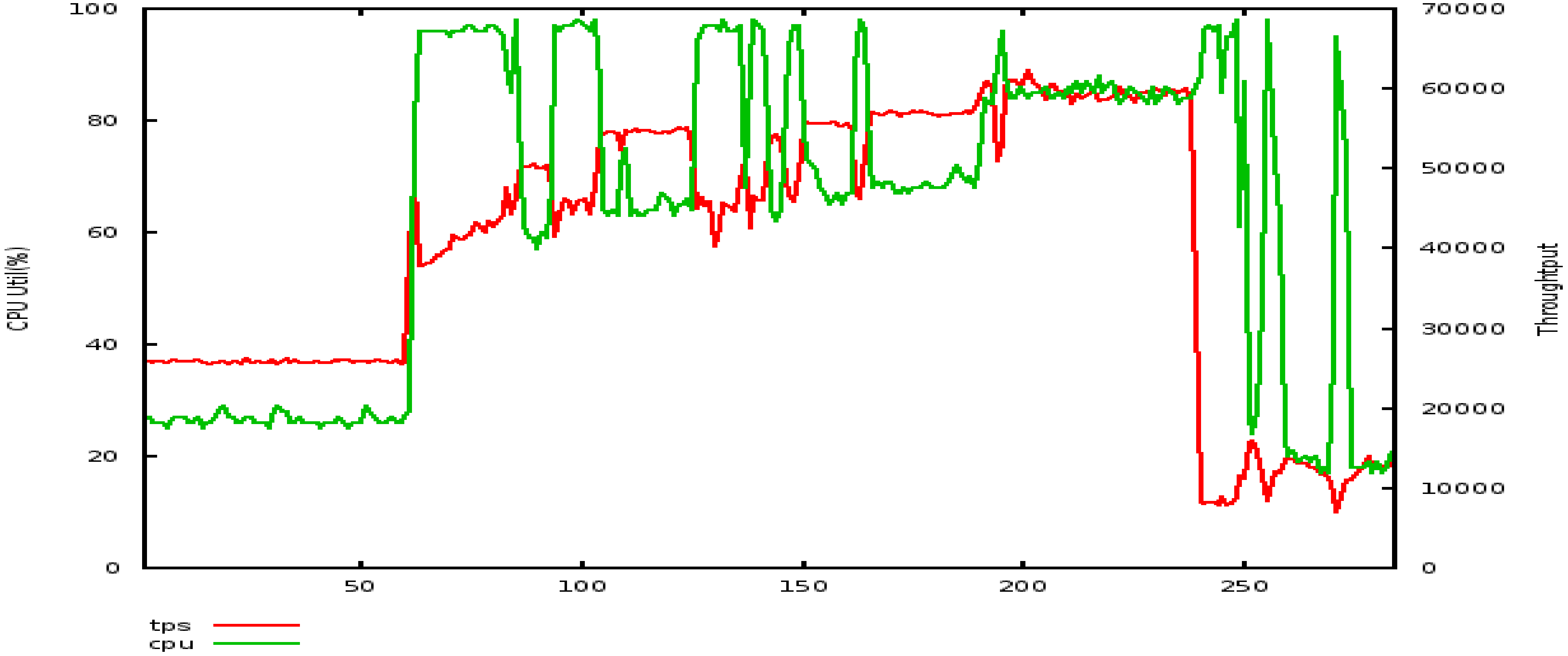
- But wait, there's more…
  - We were using fixed UCP connection pools
  - Most customers use dynamic connection pools, so they can grow with load

Let’s Try That Again

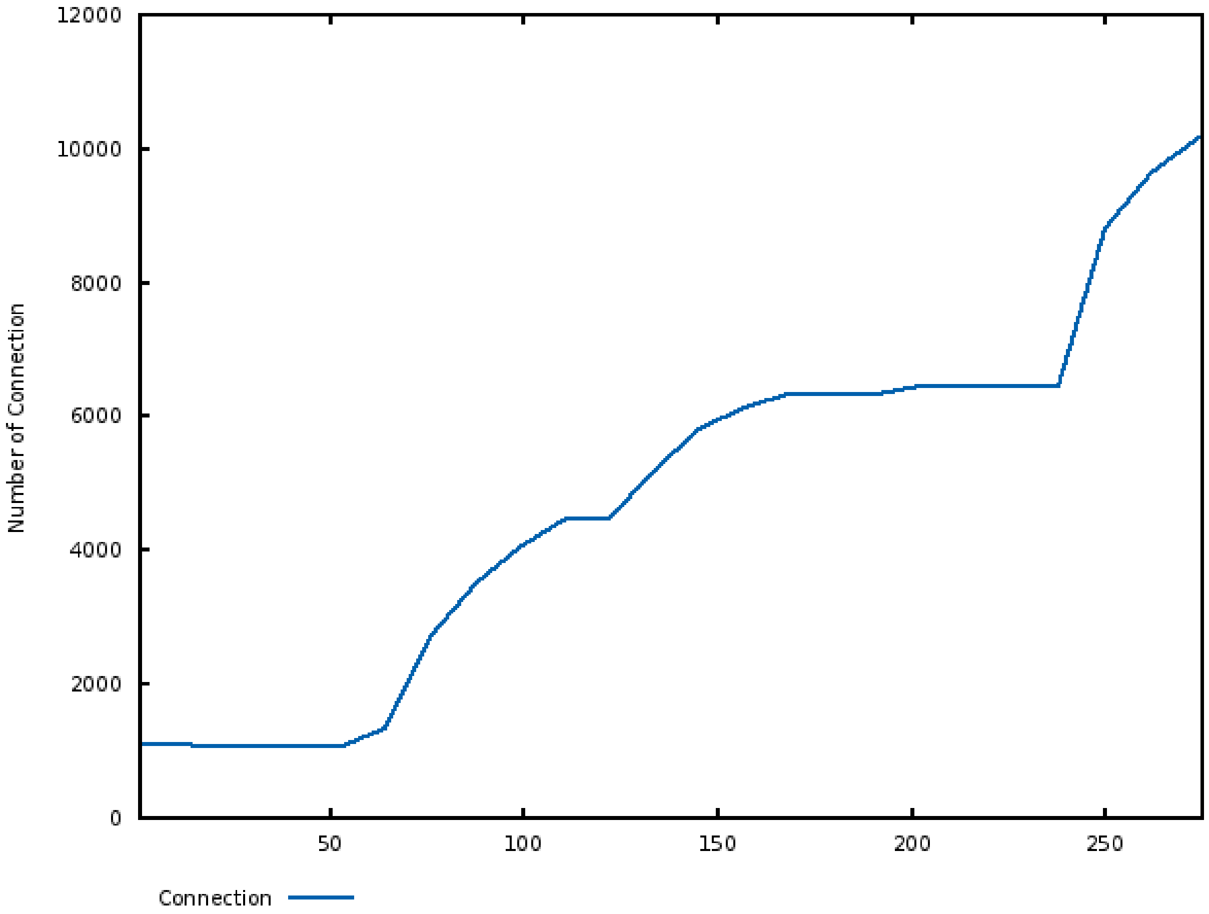
But this time with Dynamic Connection Pools

- UCP and Dedicated Connections
- Connections Pool: initial 8 connections, maximum 512
- After one minute of running drop the application think time to double load

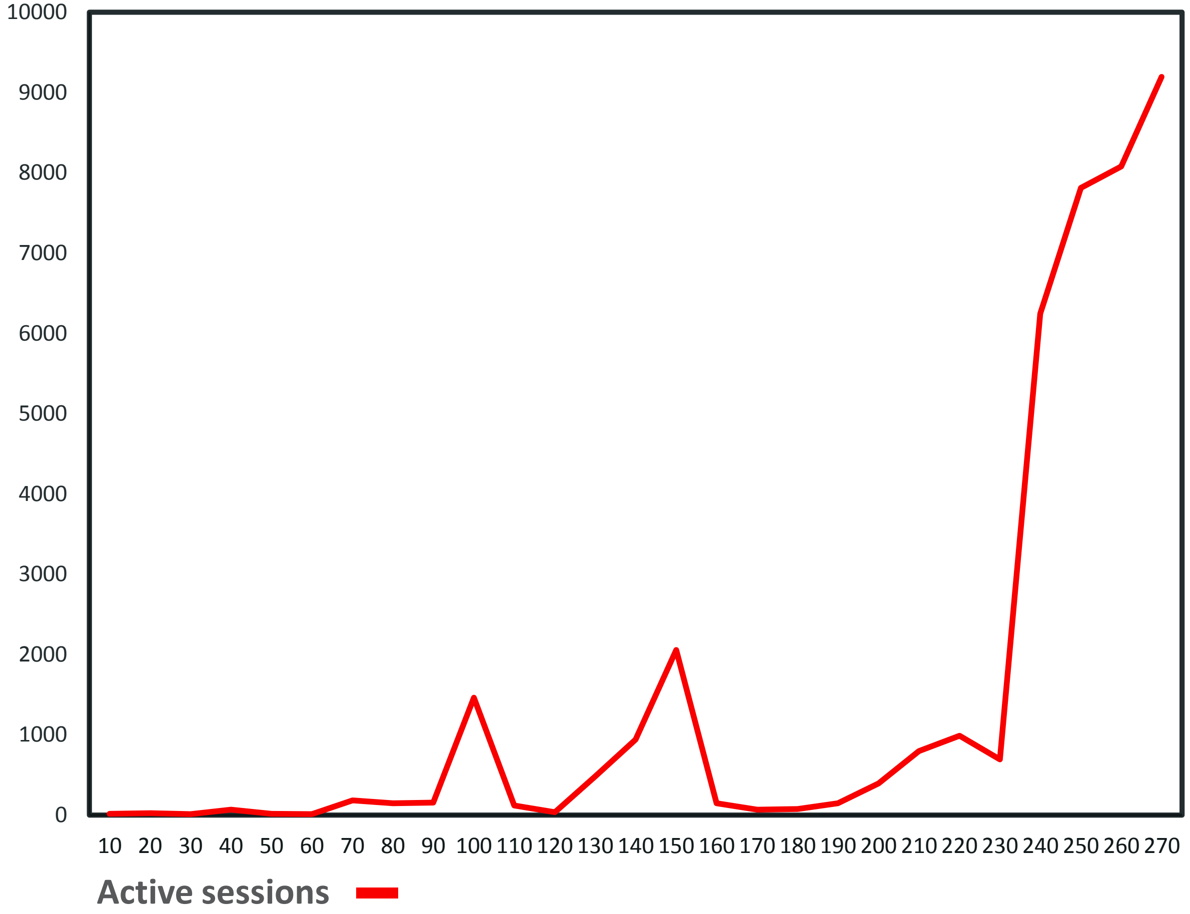
Workload Surge on Dynamic Connection Pool (min 8, max 512)



Workload Surge on Dynamic Connection Pool (min 8, max 512)



Database Connections



Active Database Connections

ORACLE® ORACLE® REAL-WORLD PERFORMANCE

Analysis

- System goes highly unstable when workload surge is applied
- Several logon storms
- For a while it seems like it may recover
- But then the large number of connections leads to the (now expected) big drop in throughput

Conclusions

- Connection strategy are still important!
- For stability use some form of connection pool
  - Preferably client side
- Dynamic connection pools will cause logon storms
  - Depending upon the severity this can lead to a performance glitch or an outage
- DON’T USE DYNAMIC CONNECTION POOLS
- Remember:

$$X*Y*Z < 10*\text{CPU cores}$$

RWP Sessions @ OOW17 Oct 4th Rm 3012

When	ID	Topic
11am	CON6560	Optimizing Table Scans in Today's Cloud Platforms
12pm	CON6561	Migrating On-Premises Applications to the Cloud: Examining the Connection Strategy
1pm	CON6629	Real-World Challenges with Cloud Migrations and Proof-of-Concept Projects
2pm	CON6660	Applying Oracle Database 12*c* and Real-World Performance Techniques to SAP

RWP @Demoground

- Moscone West SOA-161
- Discuss your performance challenges with RWP staff
- Bring your AWR/ADDM/ASH/SQL* Monitor for analysis by RWP

ORACLE®