Connection Pool Sizing Concepts

Toon Koppelaars
Real-World Performance
Oracle Server Technologies

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.

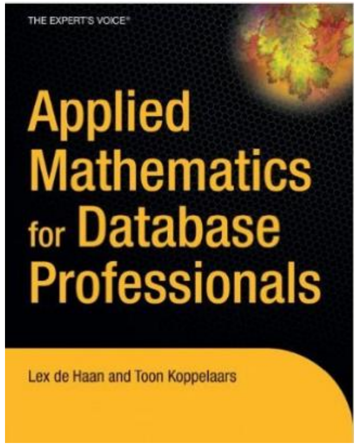
ORACLE

ORACLE REAL-WORLD PERFORMANCE

About Me

- Part of Oracle eco-system since 1987
  - Have done and seen quite a lot of application development
  - Database design, SQL and PL/SQL
- Big fan of “Using Database As a Processing Engine”
  - Not just as a persistence layer
- Member of Oracle’s Real-World Performance Group

@ToonKoppelaars



ORACLE REAL-WORLD PERFORMANCE

Topics

- Web Application Architecture
  - Application Threads, Connection Pool, Connection Queueing
- From CPU Oversubscription to Database Oversubscription
- Sizing Your Connection Pool
  - %Idle-Time in Foreground Processes
- Recommendations

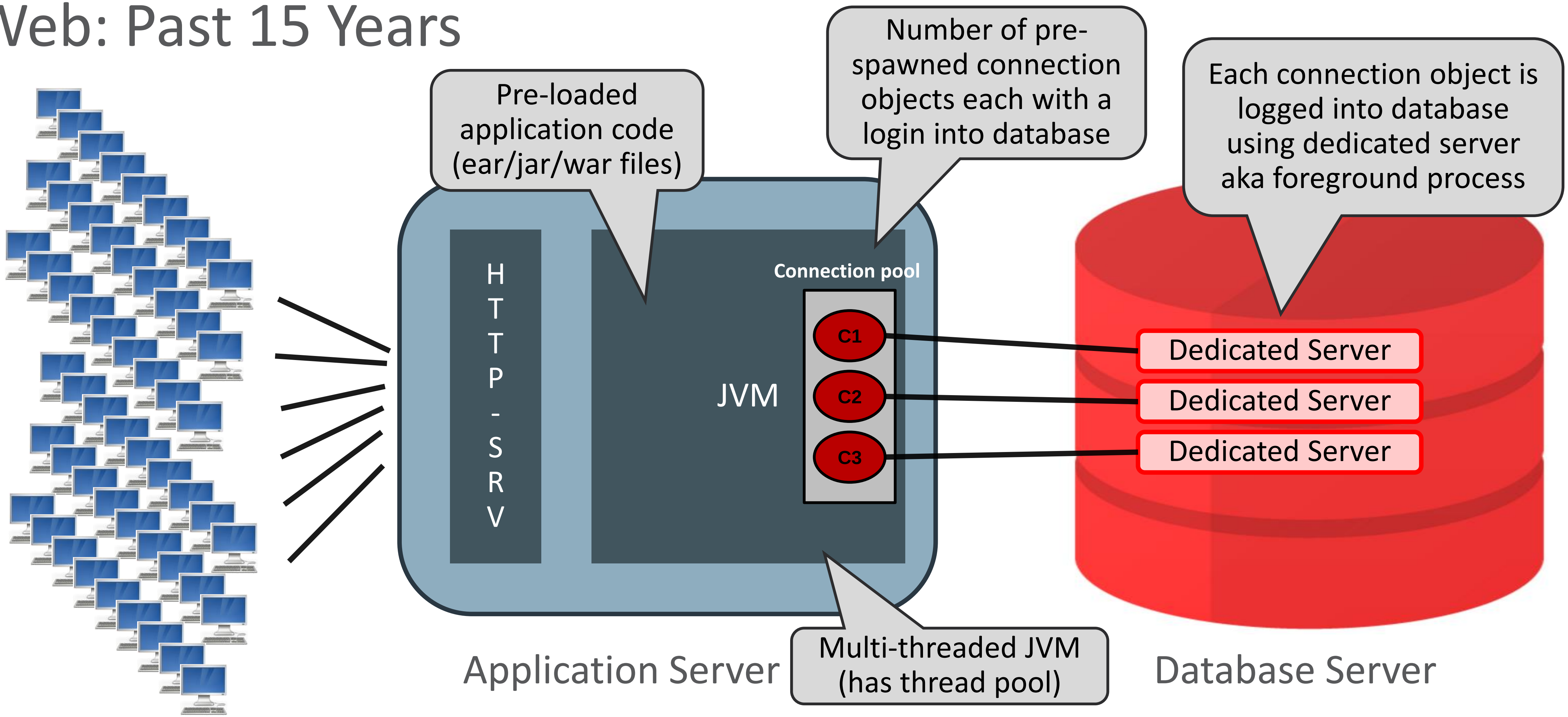
Application Architecture

- N-tier architecture has been most common architecture past 15 years
- Widely used by architects, designers and developers
- Standard for most Java EE applications
- Architecture involves:
  - Browsers with html (and JavaScript)
  - Web server that takes care of http(s) traffic
  - Application server that runs application code
  - Database server that provides data persistency services

ORACLE

ORACLE REAL-WORLD PERFORMANCE

Web: Past 15 Years



ORACLE
ORACLE REAL-WORLD PERFORMANCE

Connection Pool Configuration (WLS)

Console->Services->Data Sources

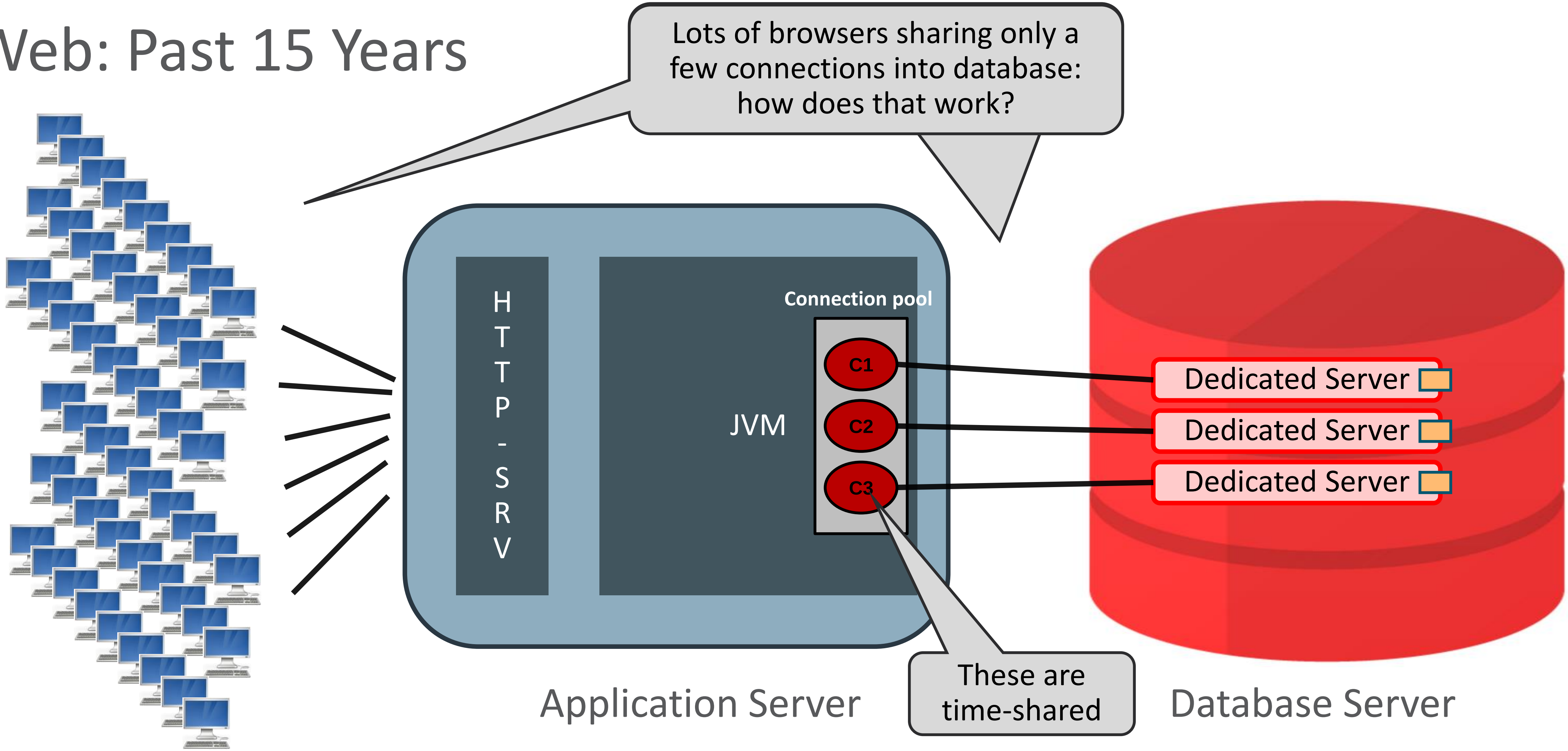
Initial Capacity:	10	The number of physical connections to create when creating the connection pool in the data source. If unable to create this number of connections, creation of the data source will fail. More Info...
Maximum Capacity:	150	The maximum number of physical connections that this connection pool can contain. More Info...
Minimum Capacity:	10	The minimum number of physical connections that this connection pool can contain after it is initialized. More Info...

- When you start application server, WLS will initialize connection pool in JVM

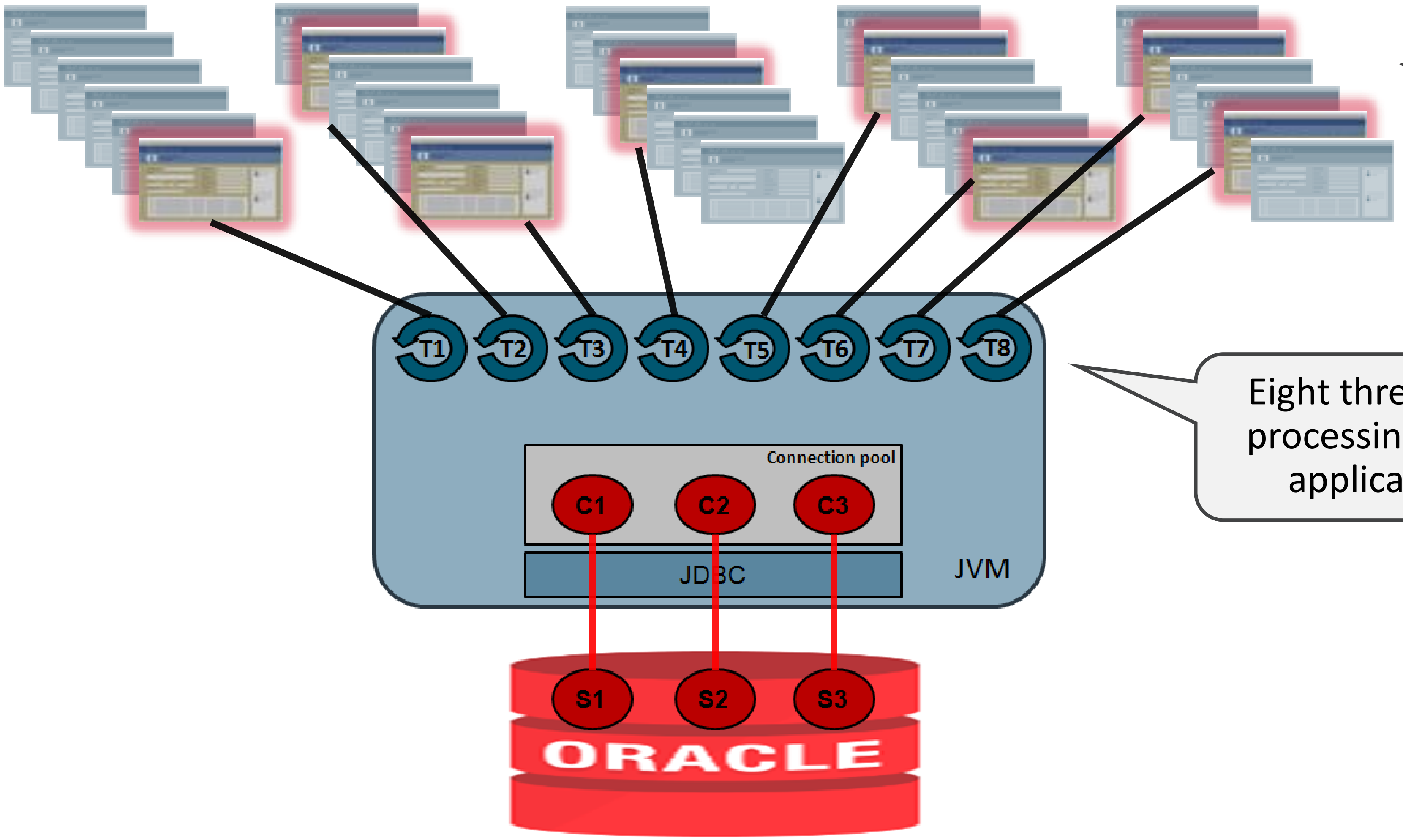
ORACLE

ORACLE REAL-WORLD PERFORMANCE

Web: Past 15 Years



Browser Requests Cause Working Application Threads



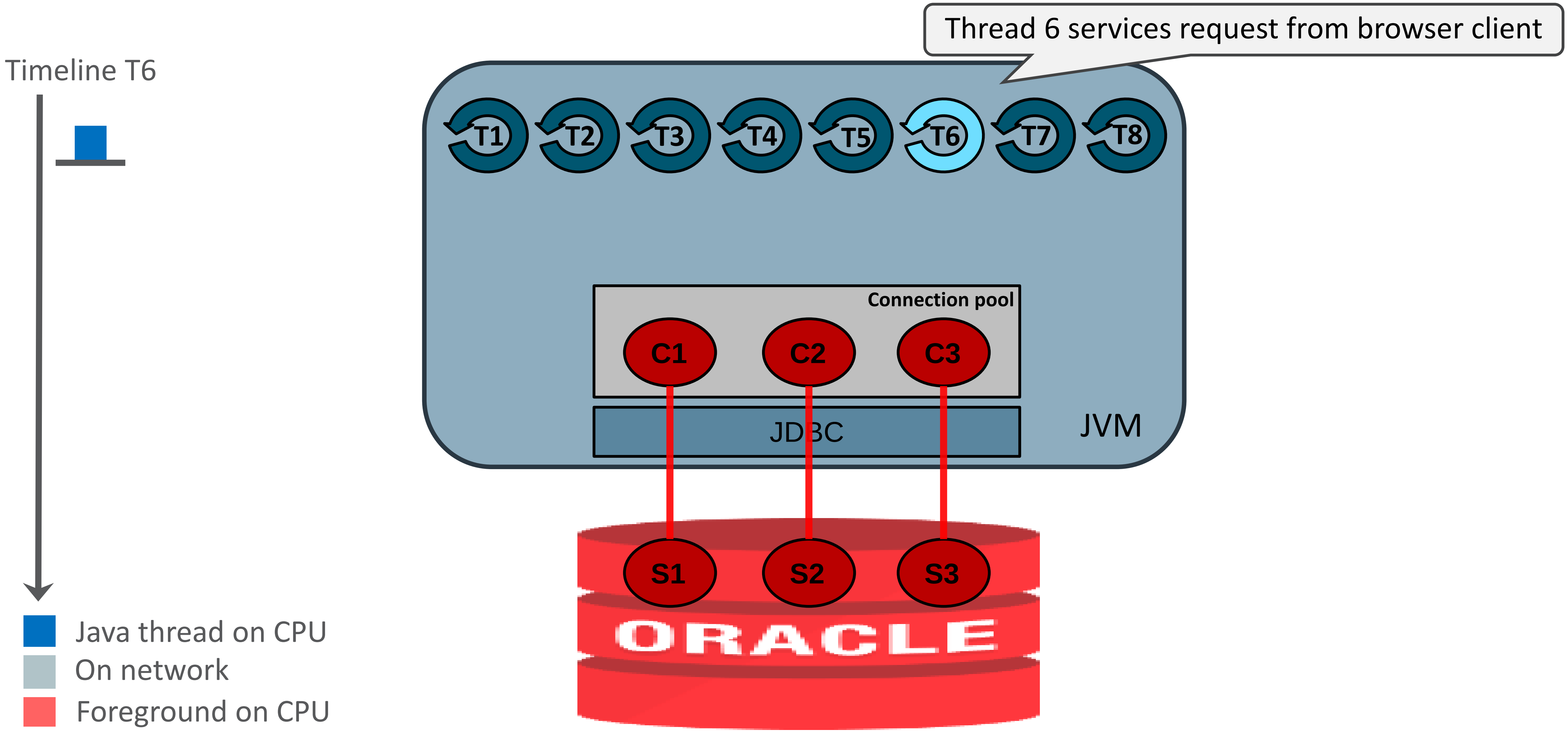
Eight browsers currently waiting for request to complete

Eight threads currently processing requests on application server

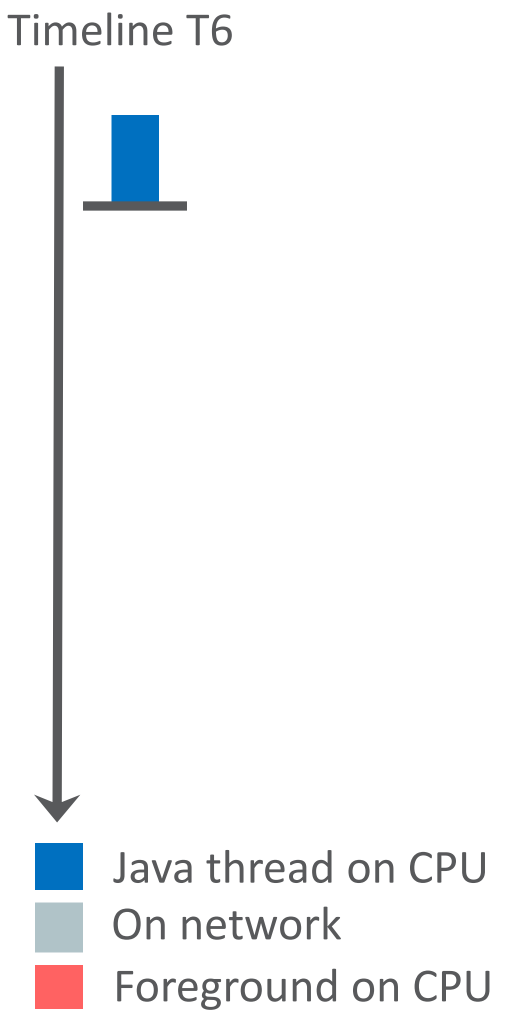
Introducing Term: Connection Reservation

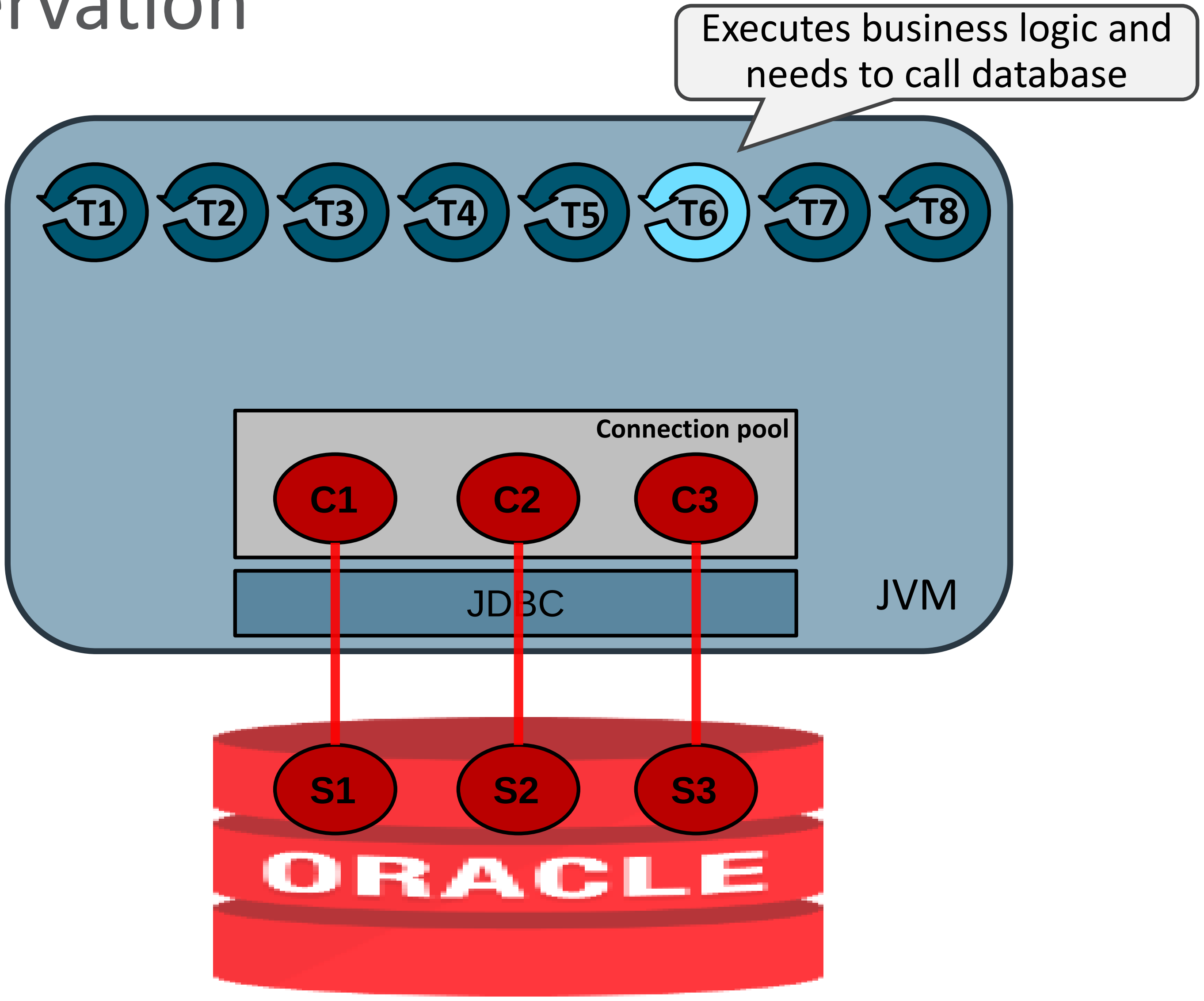
- Time during which thread has claimed one of the connections from pool to do database work

Connection Reservation

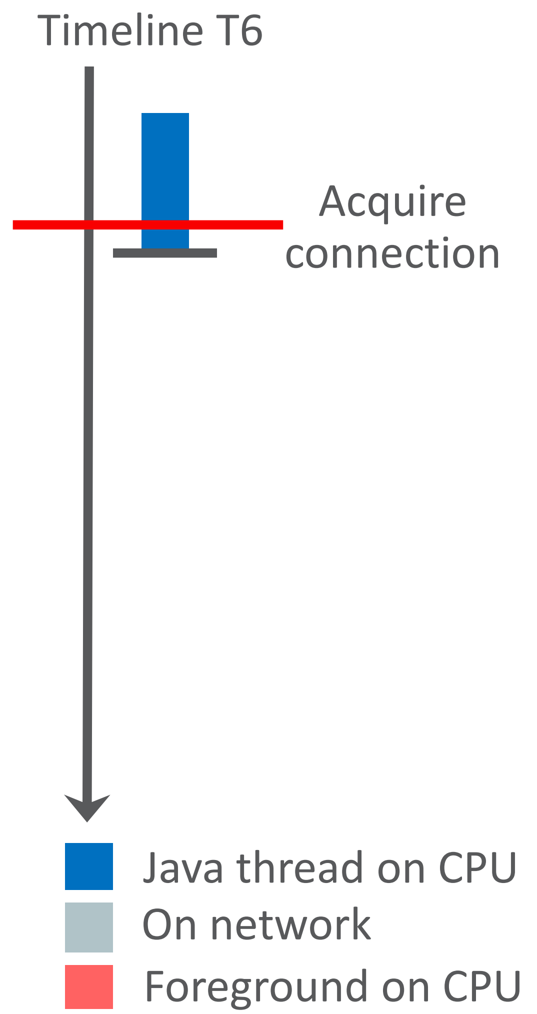


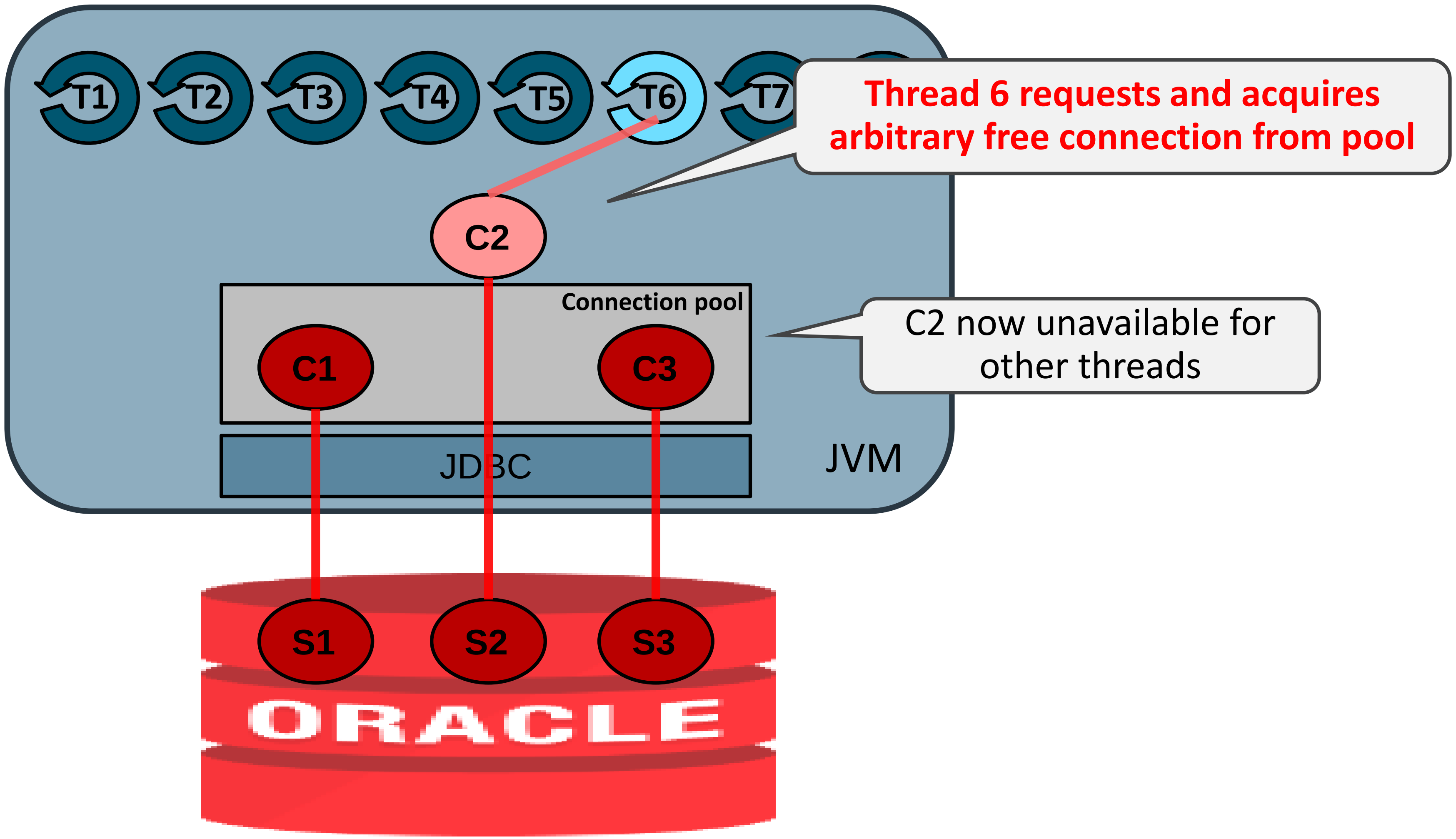
Connection Reservation



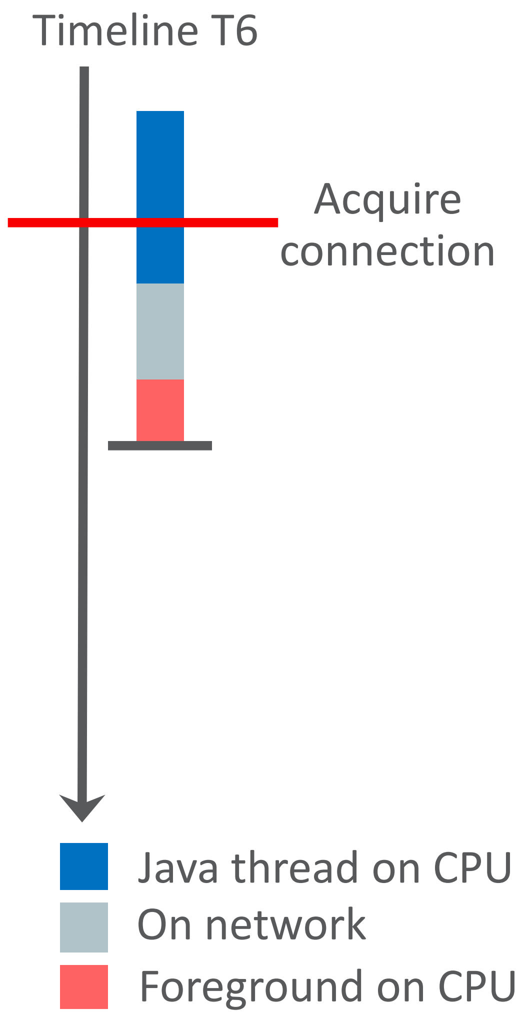


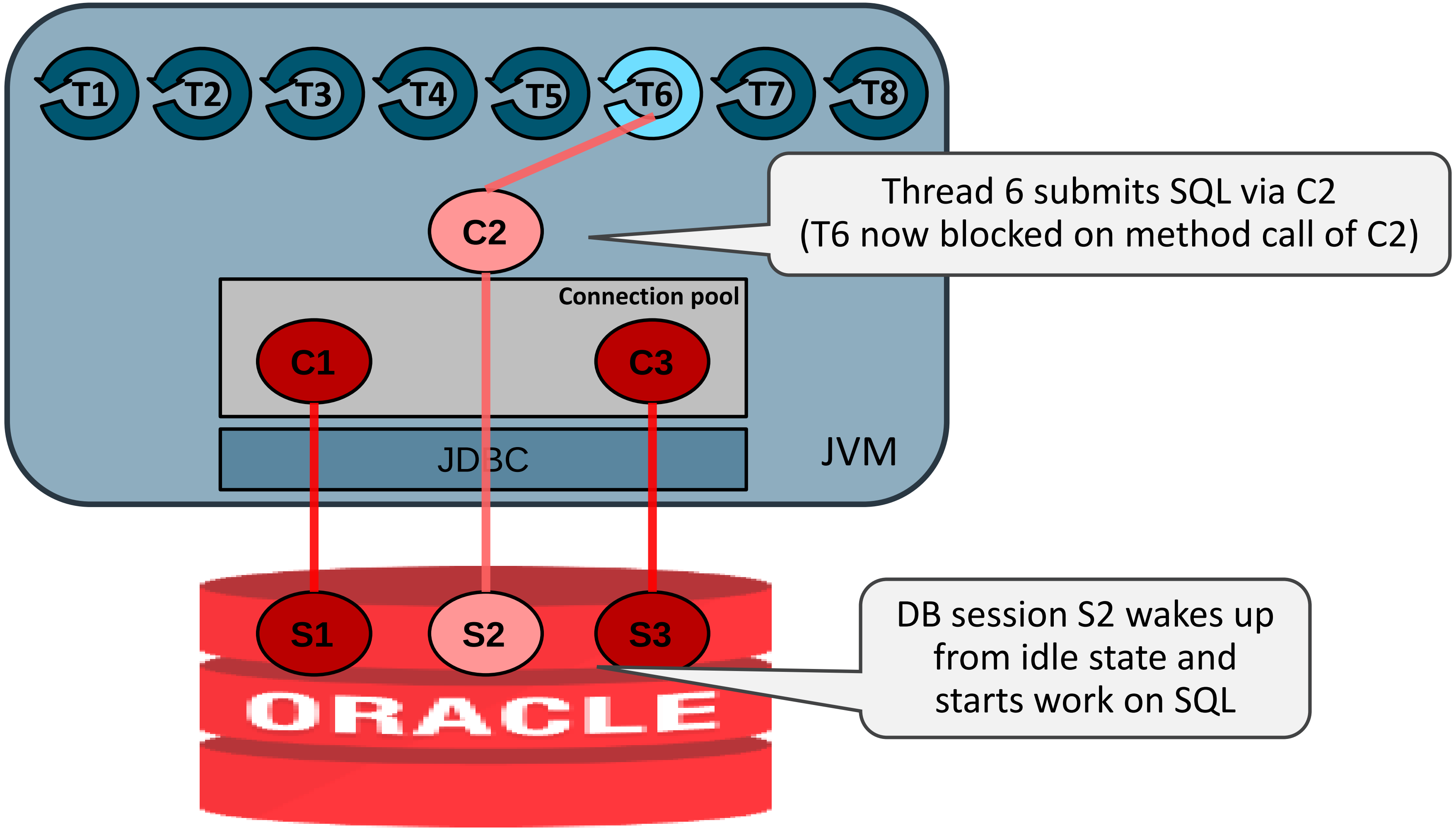
Connection Reservation



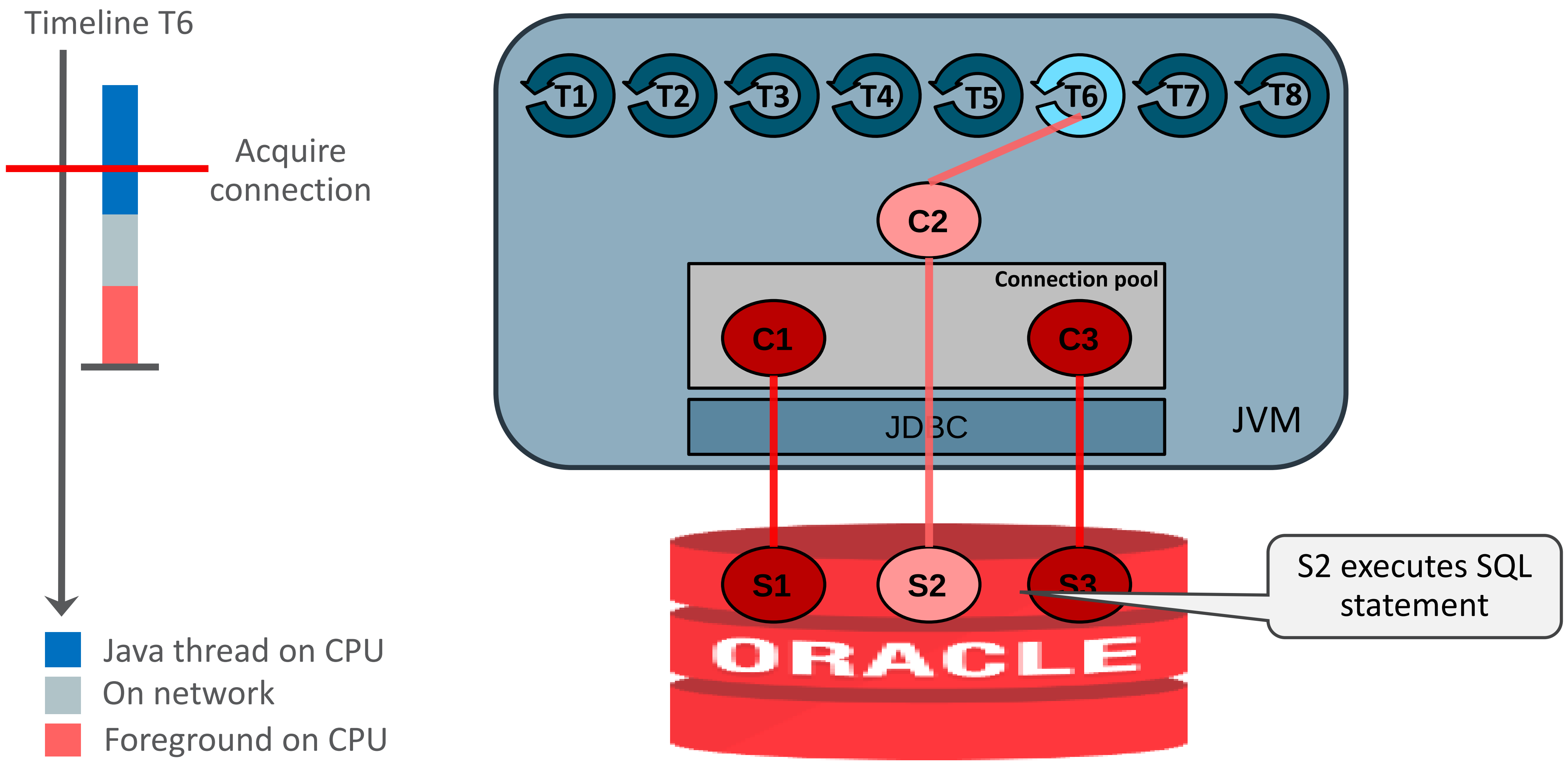


Connection Reservation

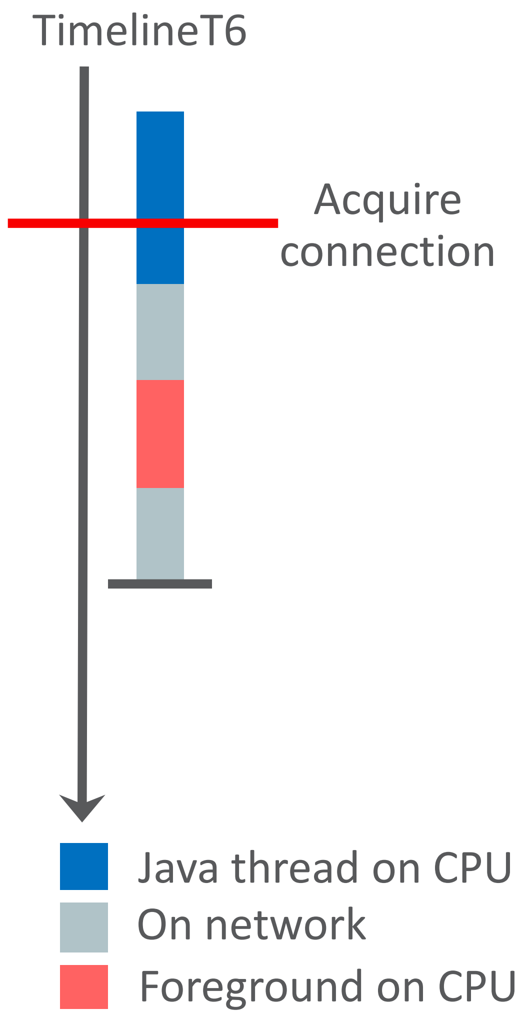


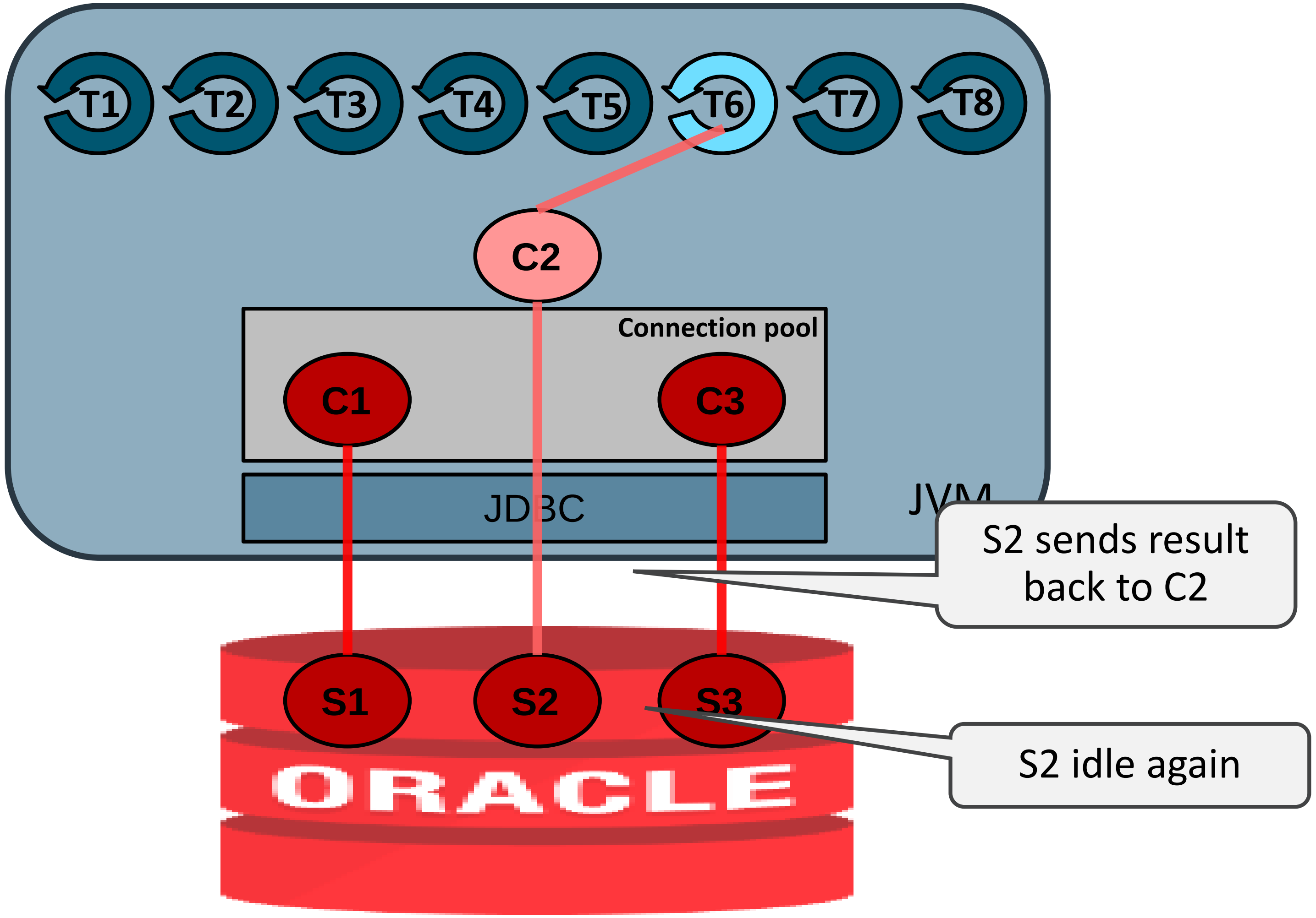


Connection Reservation

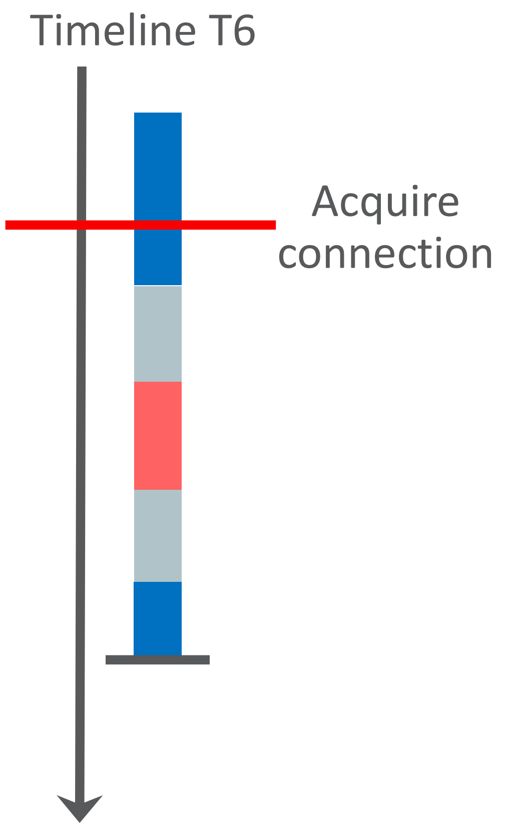


Connection Reservation

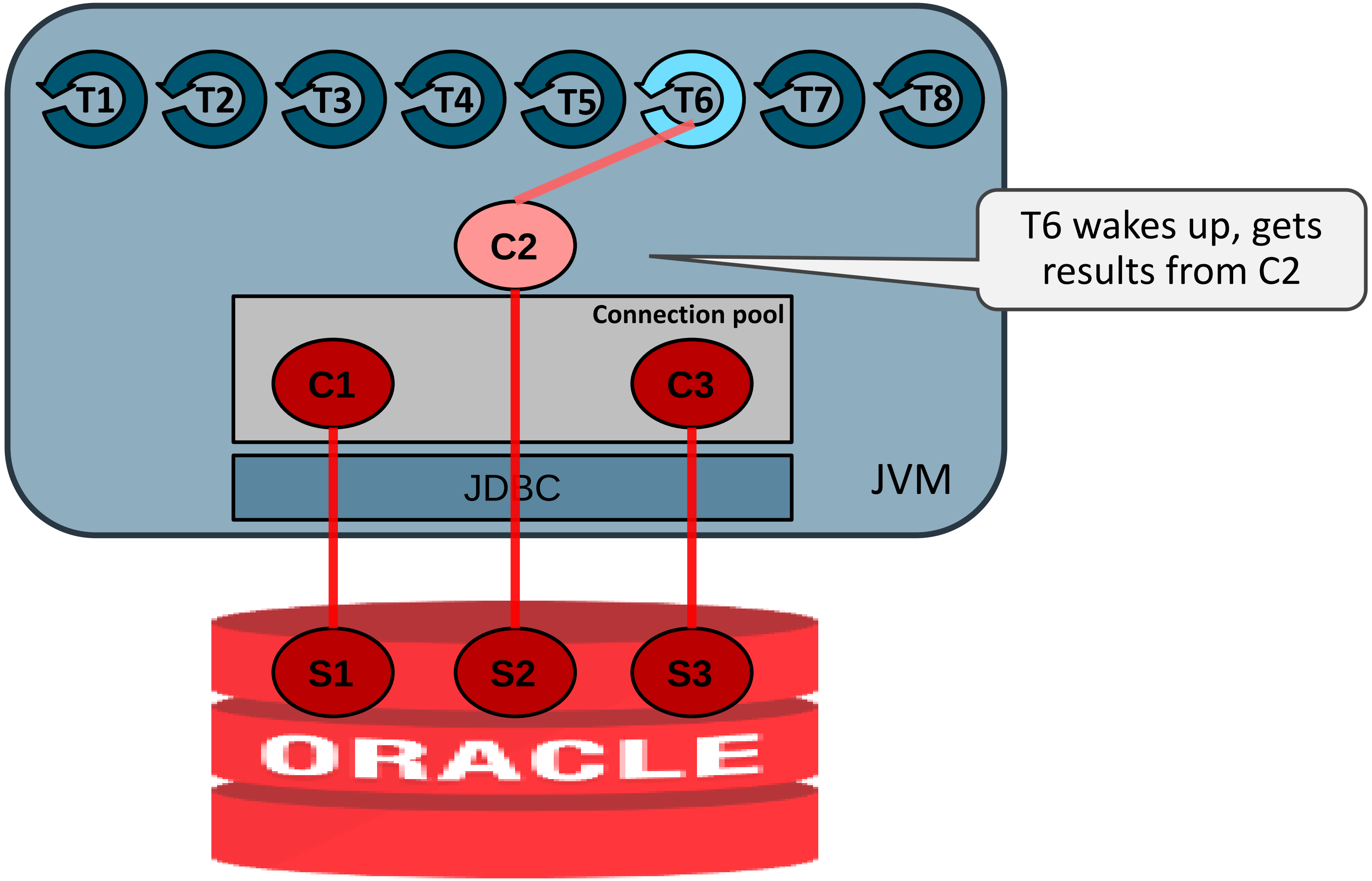




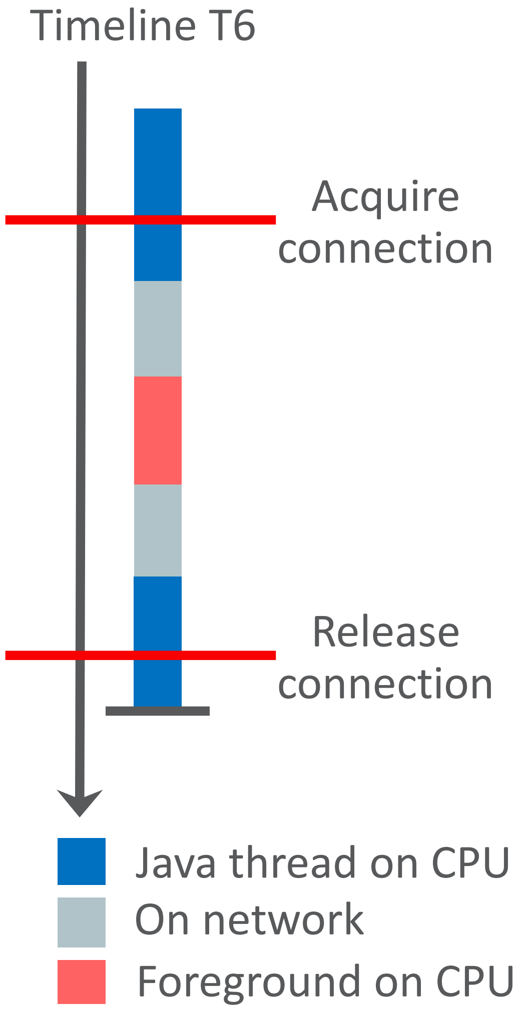
Connection Reservation

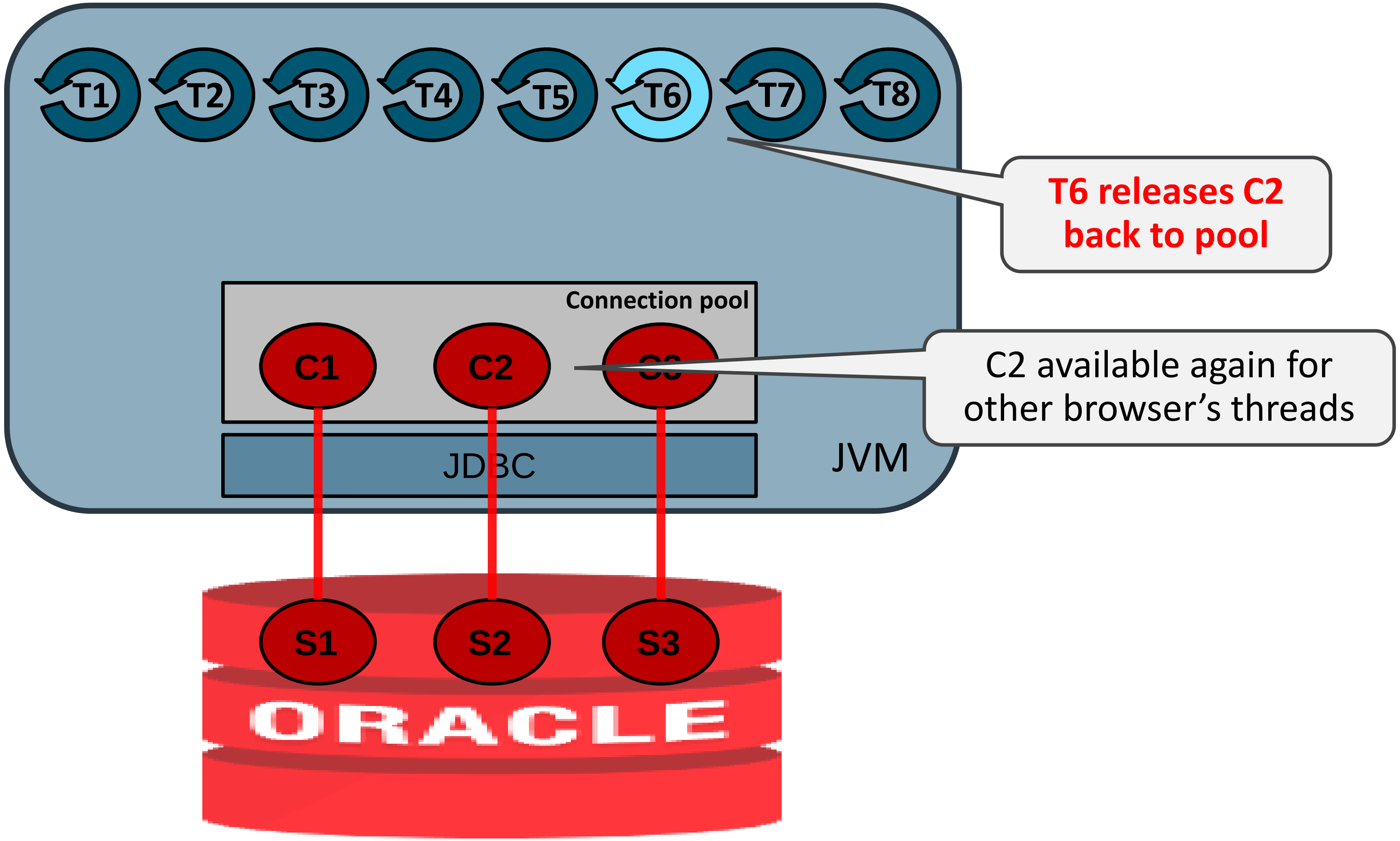


Java thread on CPU
On network
Foreground on CPU

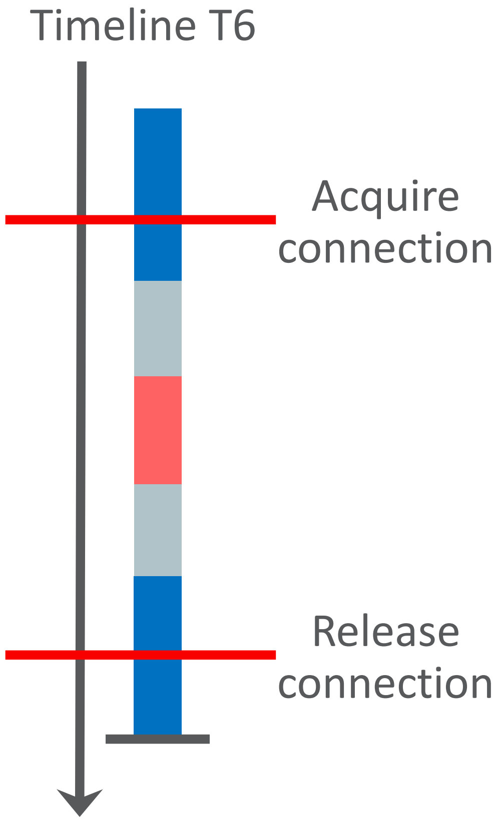


Connection Reservation

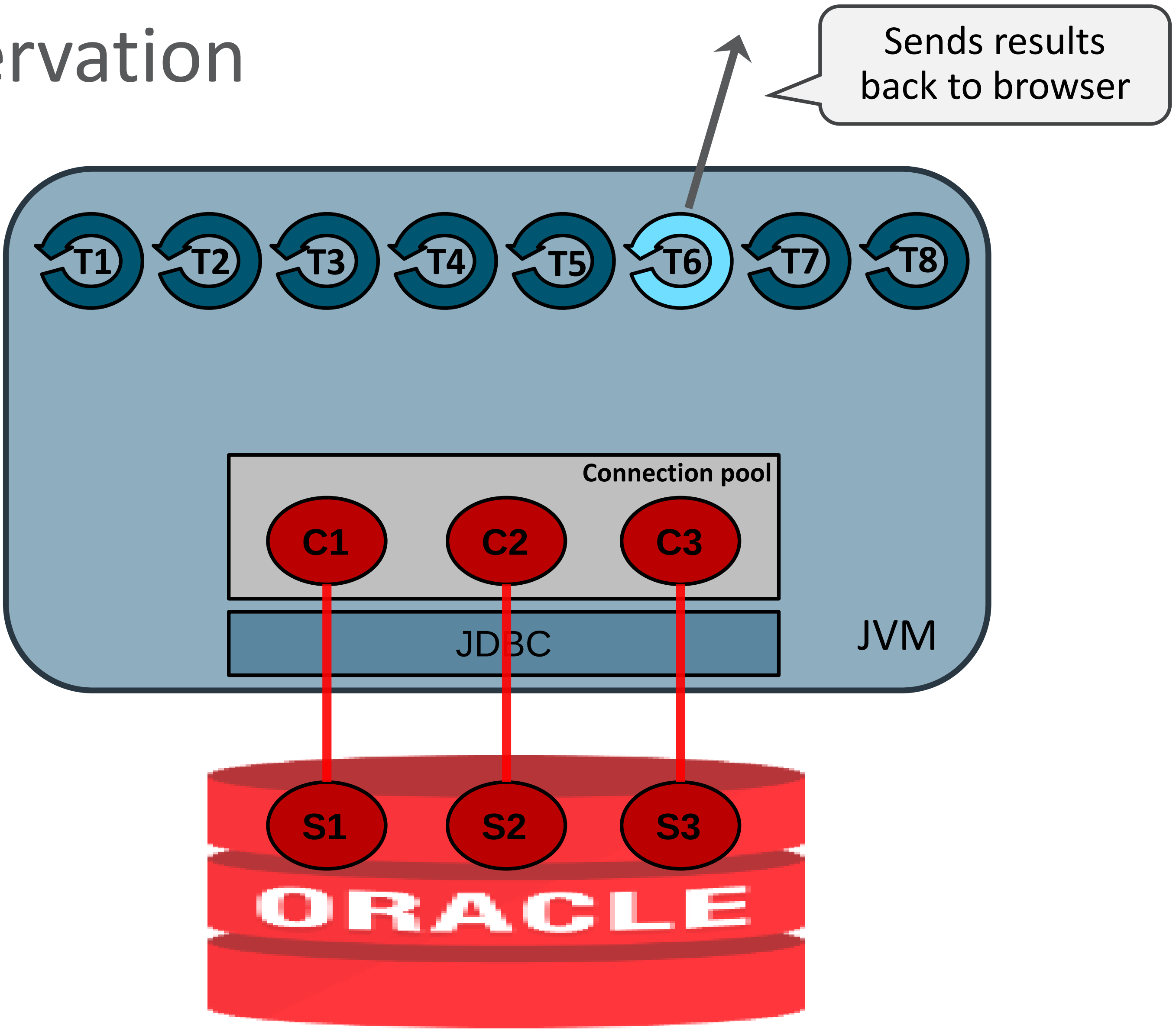




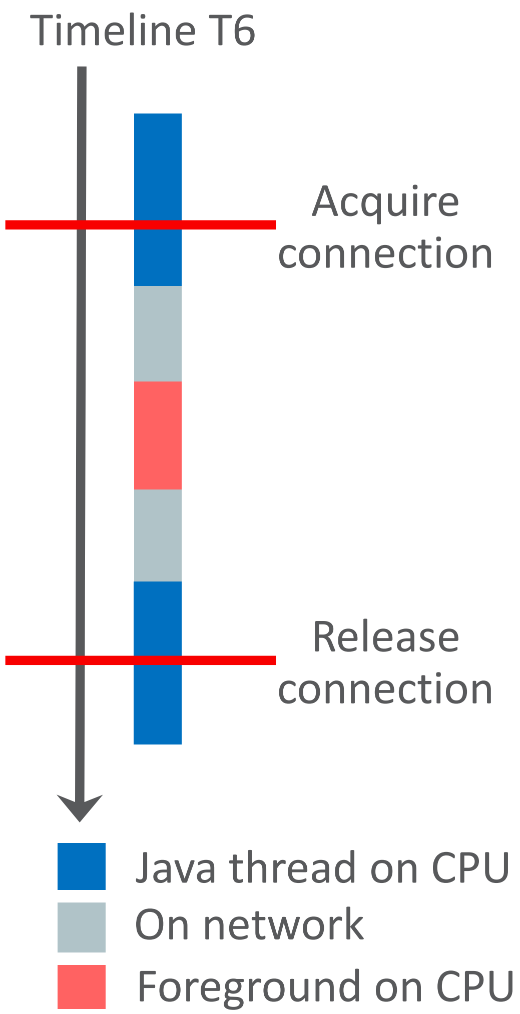
Connection Reservation

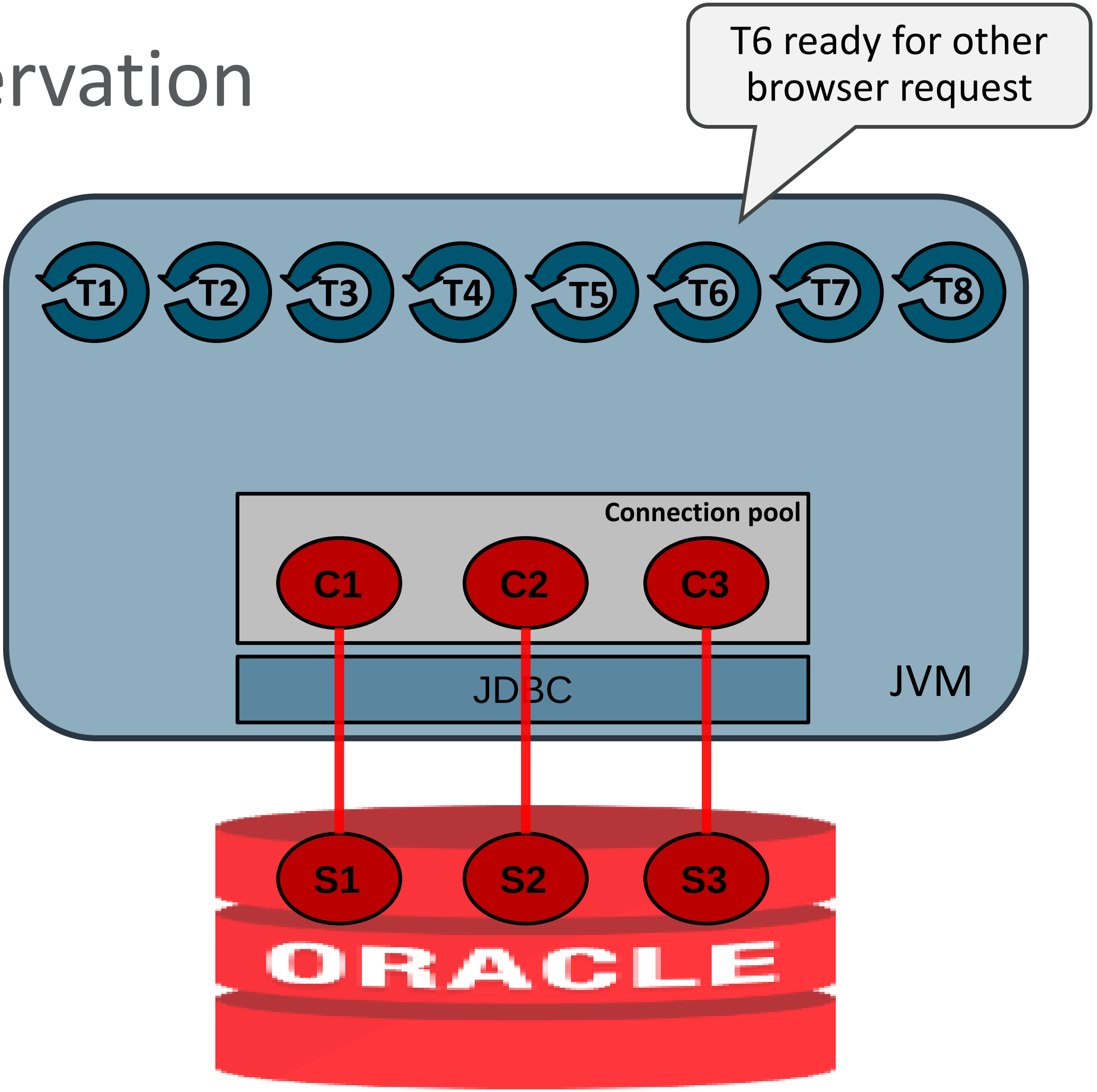


Java thread on CPU
On network
Foreground on CPU

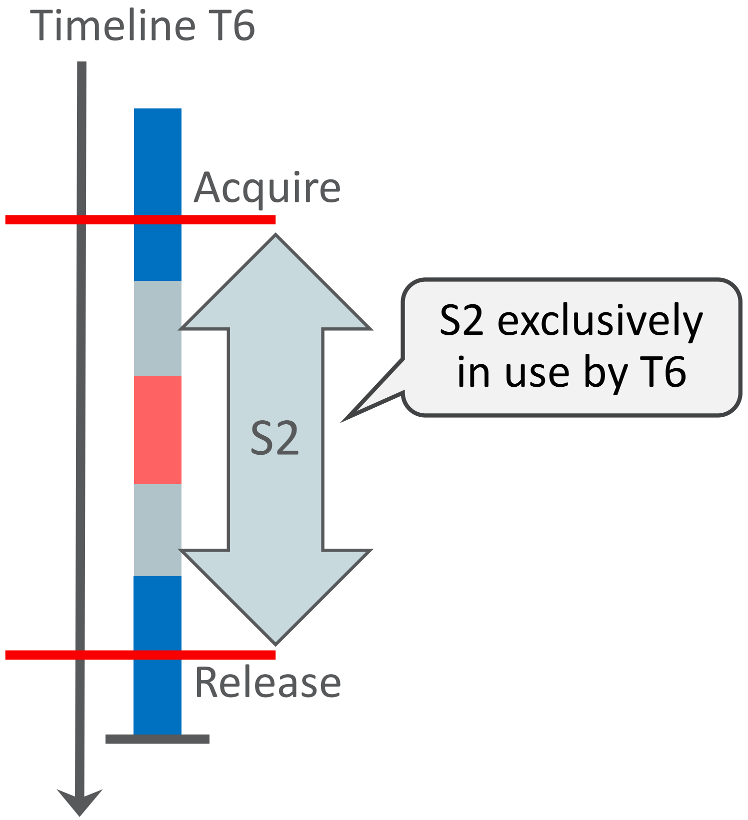


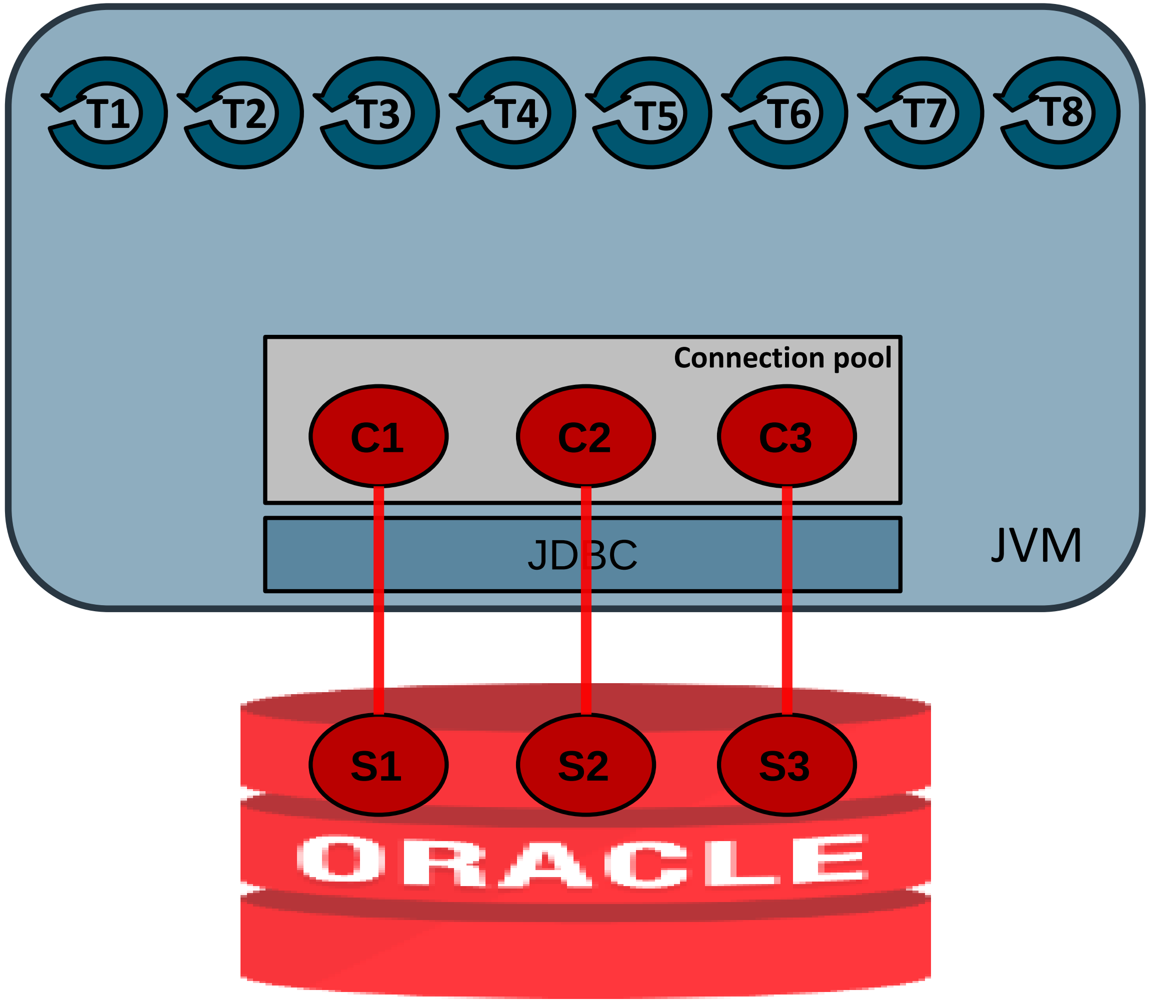
Connection Reservation





Timesharing Connections / Foregrounds

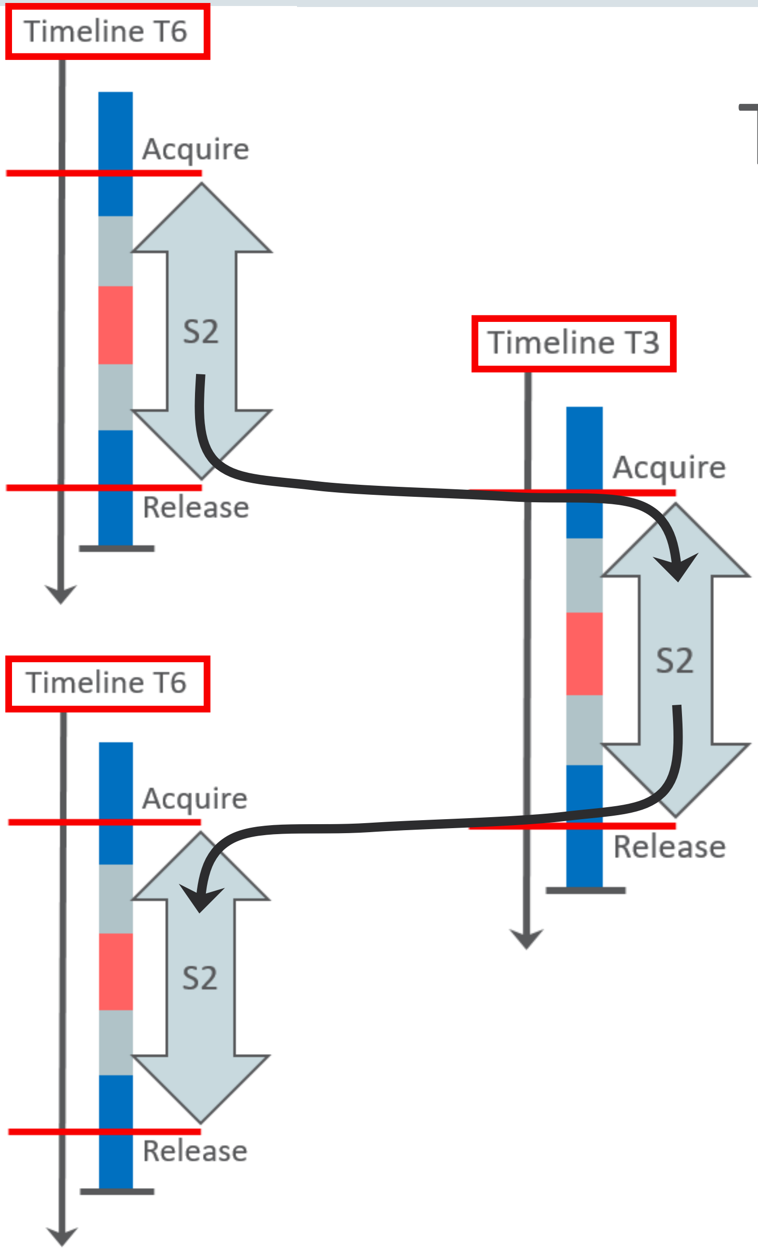


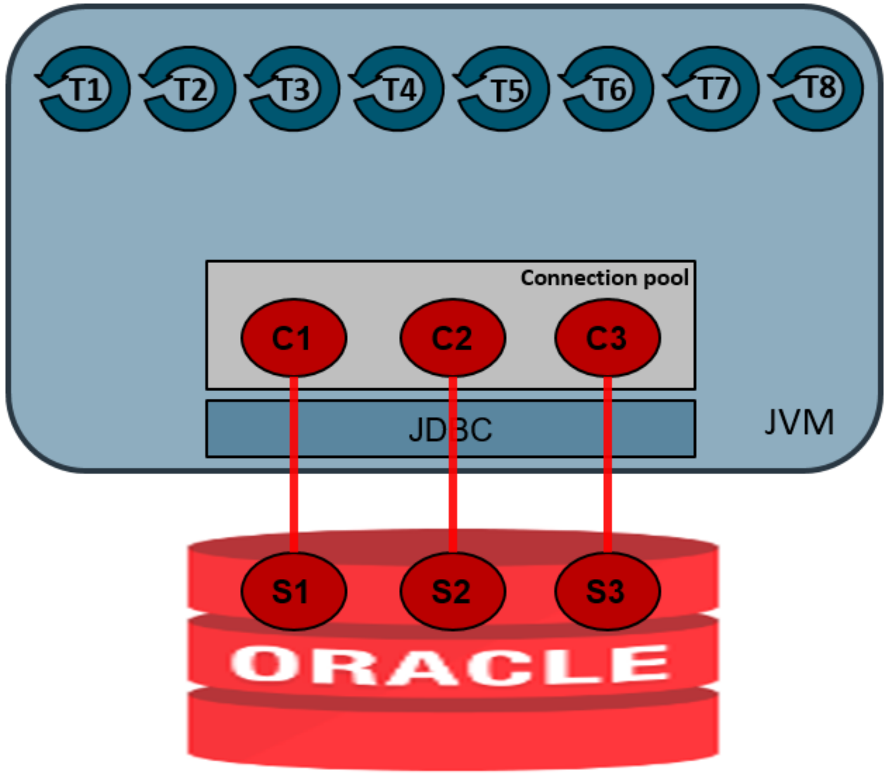


ORACLE

ORACLE REAL-WORLD PERFORMANCE

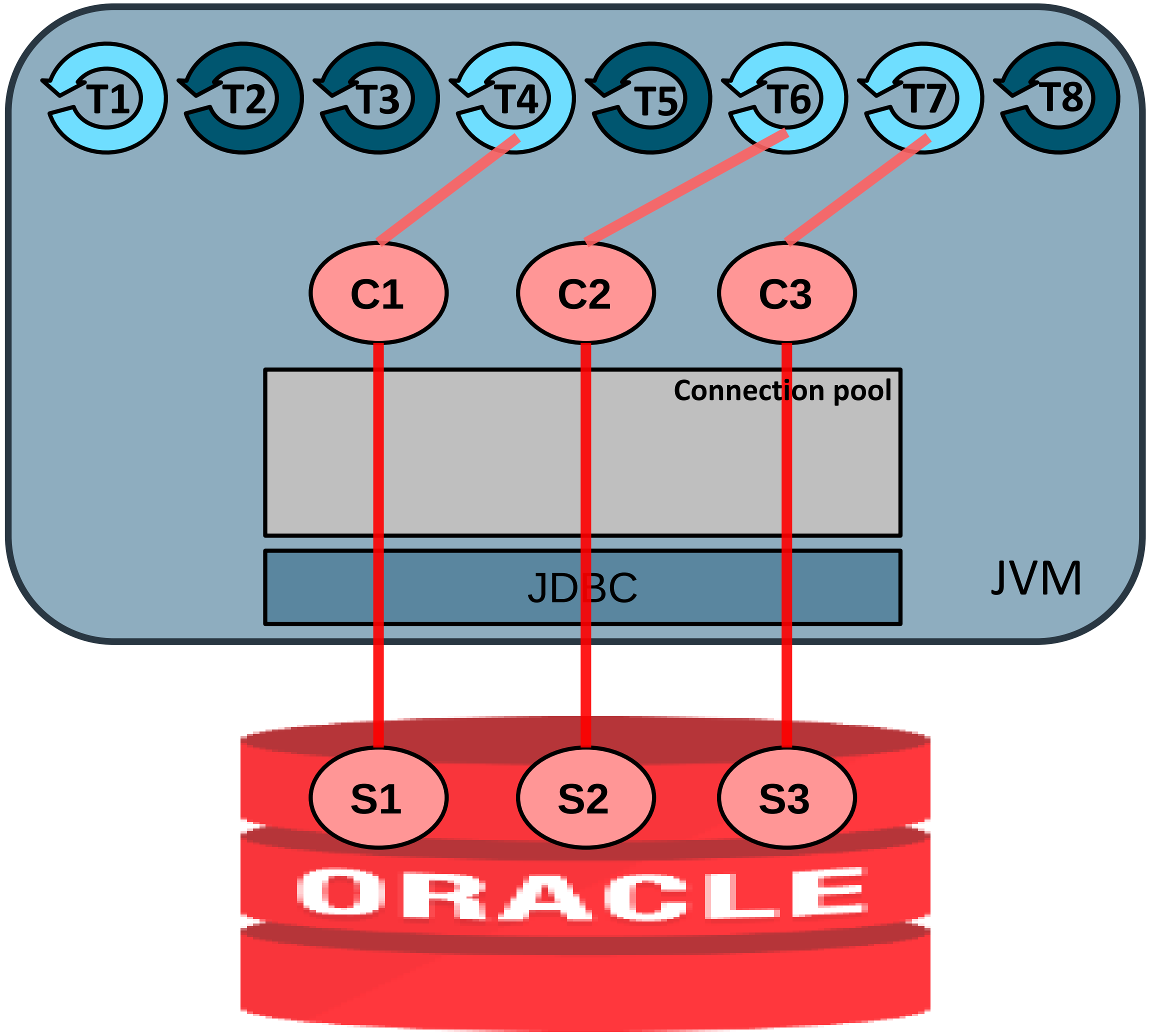
Timesharing Connections / Foregrounds





Connection Pool At Full Capacity

What if thread T1 wants to do DB work?



What If 4th Thread Wants to Do Database Work?

- Depends on how you have configured your connection pool

A. Your pool is configured to dynamically grow (max # of connections is not yet reached)

   1. New 4th connection created and handed out to thread T1

B. Your pool has reached max # of connections configured
   Two options:

   2. Your thread will get Java exception
   3. Your thread will be put to sleep, until connection becomes available

ORACLE

ORACLE REAL-WORLD PERFORMANCE

Connection Pool Configuration (WLS)

Console->Services->Data Sources

Initial Capacity:	10	The number of physical connections to create when creating the connection pool in the data source. If unable to create this number of connections, creation of the data source will fail. More Info...
Maximum Capacity:	150	The maximum number of physical connections that this connection pool can contain. More Info...
Minimum Capacity:	10	The minimum number of physical connections that this connection pool can contain after it is initialized. More Info...

Maximum Waiting for Connection:	2147483647	The maximum number of connection requests that can concurrently block threads while waiting to reserve a connection from the data source's connection pool. More Info...
Connection Reserve Timeout:	10	The number of seconds after which a call to reserve a connection from the connection pool will timeout. More Info...

ORACLE

ORACLE REAL-WORLD PERFORMANCE

What If 4th Thread Wants to Do Database Work?

- Developers do not want:
  - Their threads to receive an exception from connection pool manager
  - Their thread to be put “on-hold” by connection pool manager

- So we nearly always see connection pools that can grow to very large # of connections

- We call these: dynamic connection pools
  - These can cause database to become CPU-oversubscribed

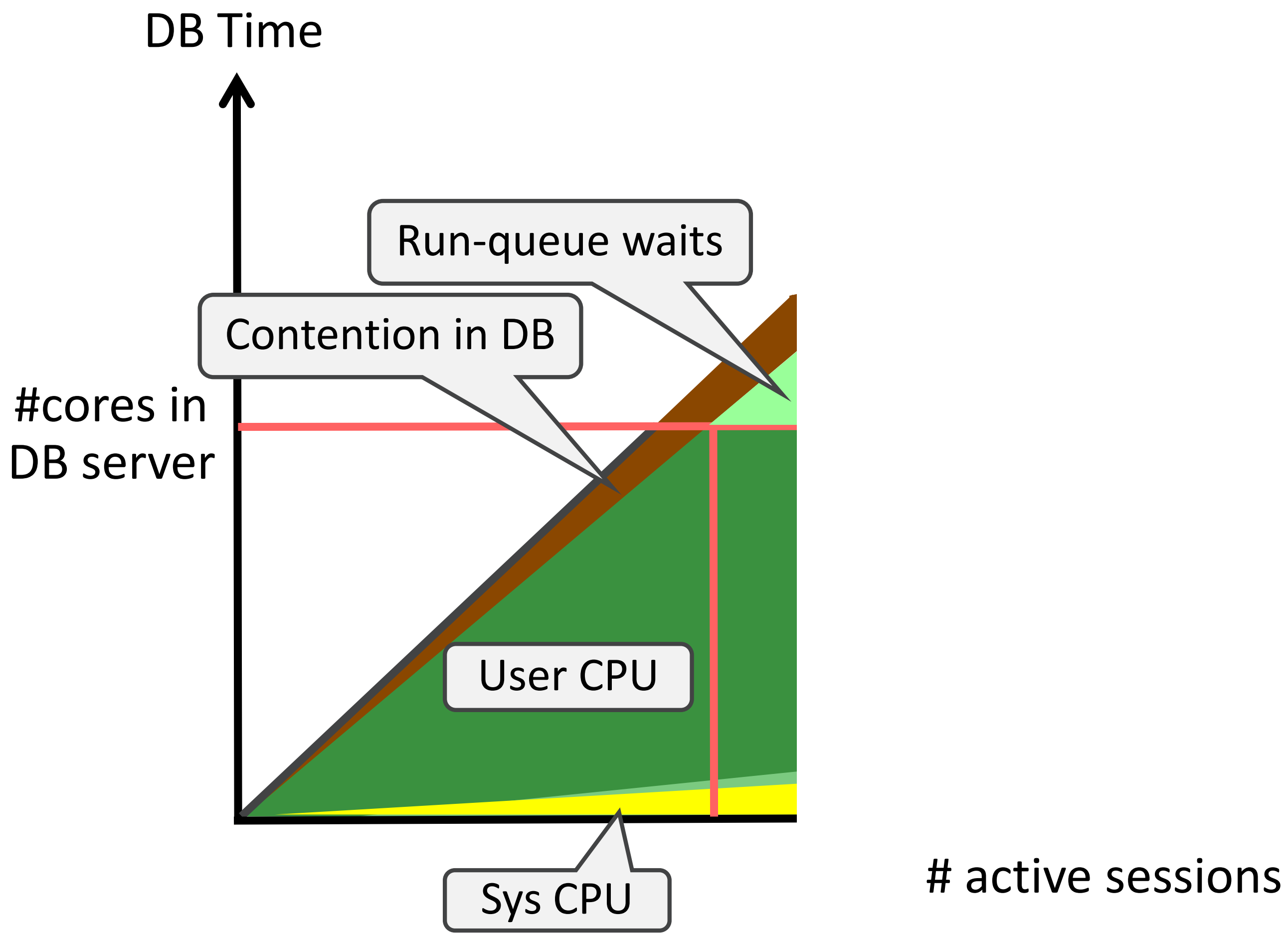
Topics

- Web Application Architecture
  - Application Threads, Connection Pool, Connection Queueing
- From CPU Oversubscription to Database Oversubscription
- Sizing Your Connection Pool
  - %Idle-Time in Foreground Processes
- Recommendations

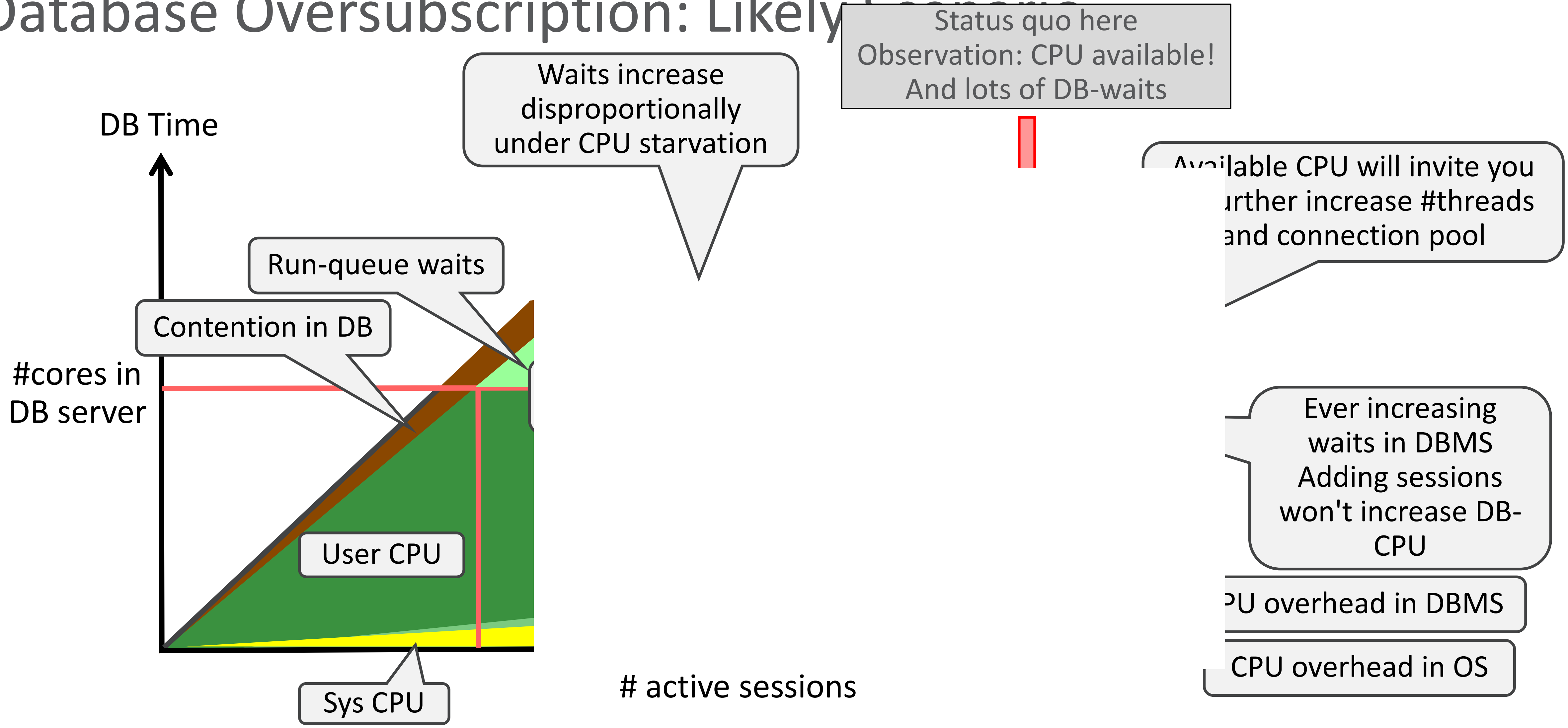
#1 Issue in Real-World: Database Oversubscription

- You might be having this problem without knowing it
- Majority of customers that come to us with escalations, experience this
- It's not obvious that symptoms point to this problem
- Symptoms might lead you down wrong path
- And spend a lot of time with support, without getting anywhere

CPU Oversubscription

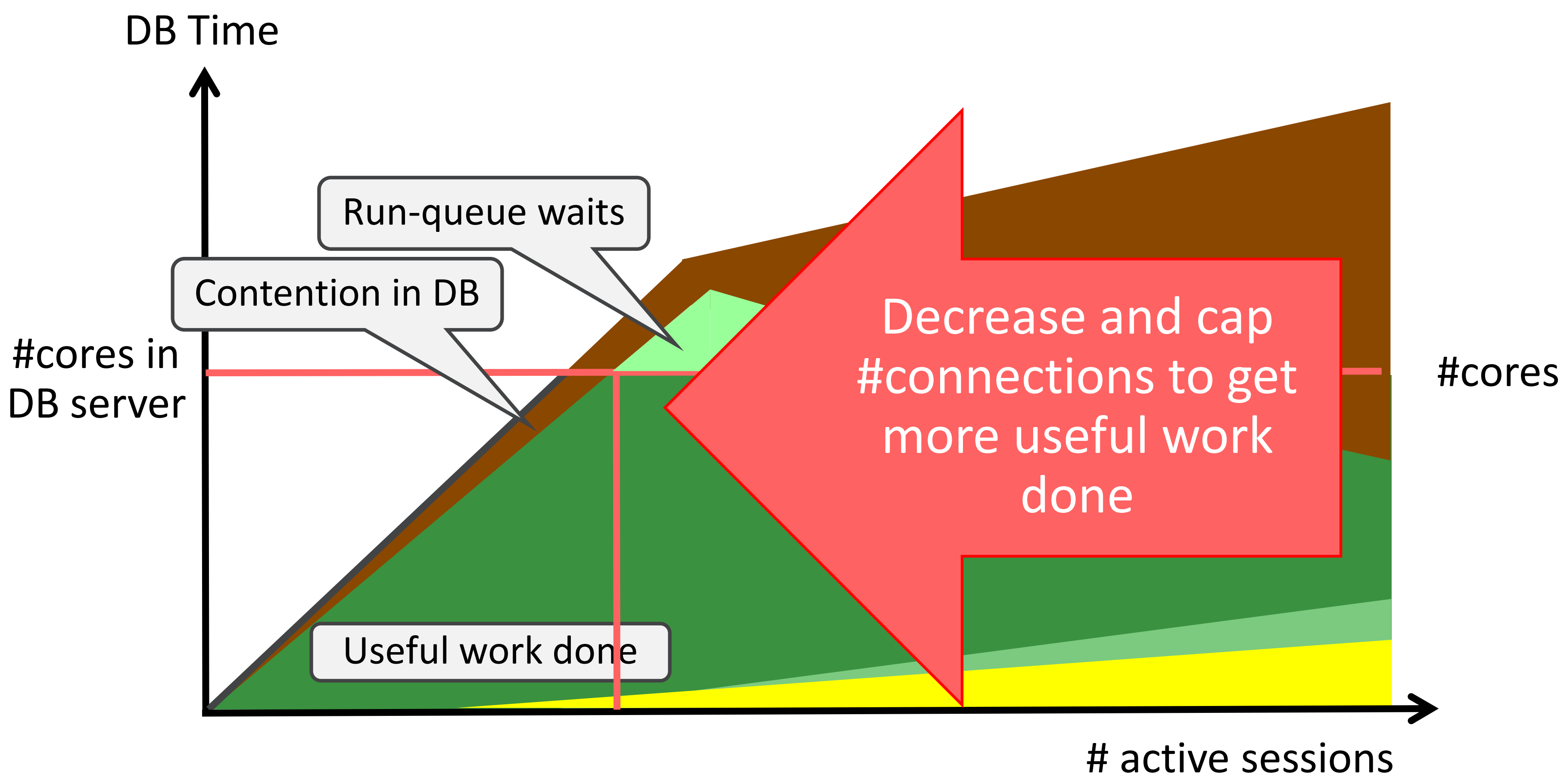


Database Oversubscription: Likely



 |

Only One Thing You Should Do



Example Database Oversubscription

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
my.company.com	Solaris[tm] OE (64-bit)	240	30	4	1012.00

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	7510	16-Aug-18 14:00:27	3667	6.5
End Snap:	7511	16-Aug-18 15:00:07	7978	2.5

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
buffer busy waits	6,617,394	2544K	384	47.9	Concurrency
enq: TX - contention	7,137,323	2198.7K	308	41.4	Other
DB CPU		251K		4.7	
latch: enqueue hash chains	2,015,409	176.1K	87	3.3	Other
db file sequential read	37,173,715	61K	2	1.1	User I/O
[illegible]	[illegible]391	48.1K	19	.9	Concurrency
[illegible]	[illegible]685	10.3K	2	.2	User I/O

Operating System Statistics - Detail

Snap Time	Load	%busy	%user	%sys	%idle	%iowait
16-Aug 14:00:27	104.09					
16-Aug 15:00:07	167.55	49.27	32.42	16.85	50.73	0.00

ORACLE

ORACLE REAL-WORLD PERFORMANCE

Moral…

- It is far better to have threads queue for pooled connection
  Than,
  It is for database to be oversubscribed

The Big Question

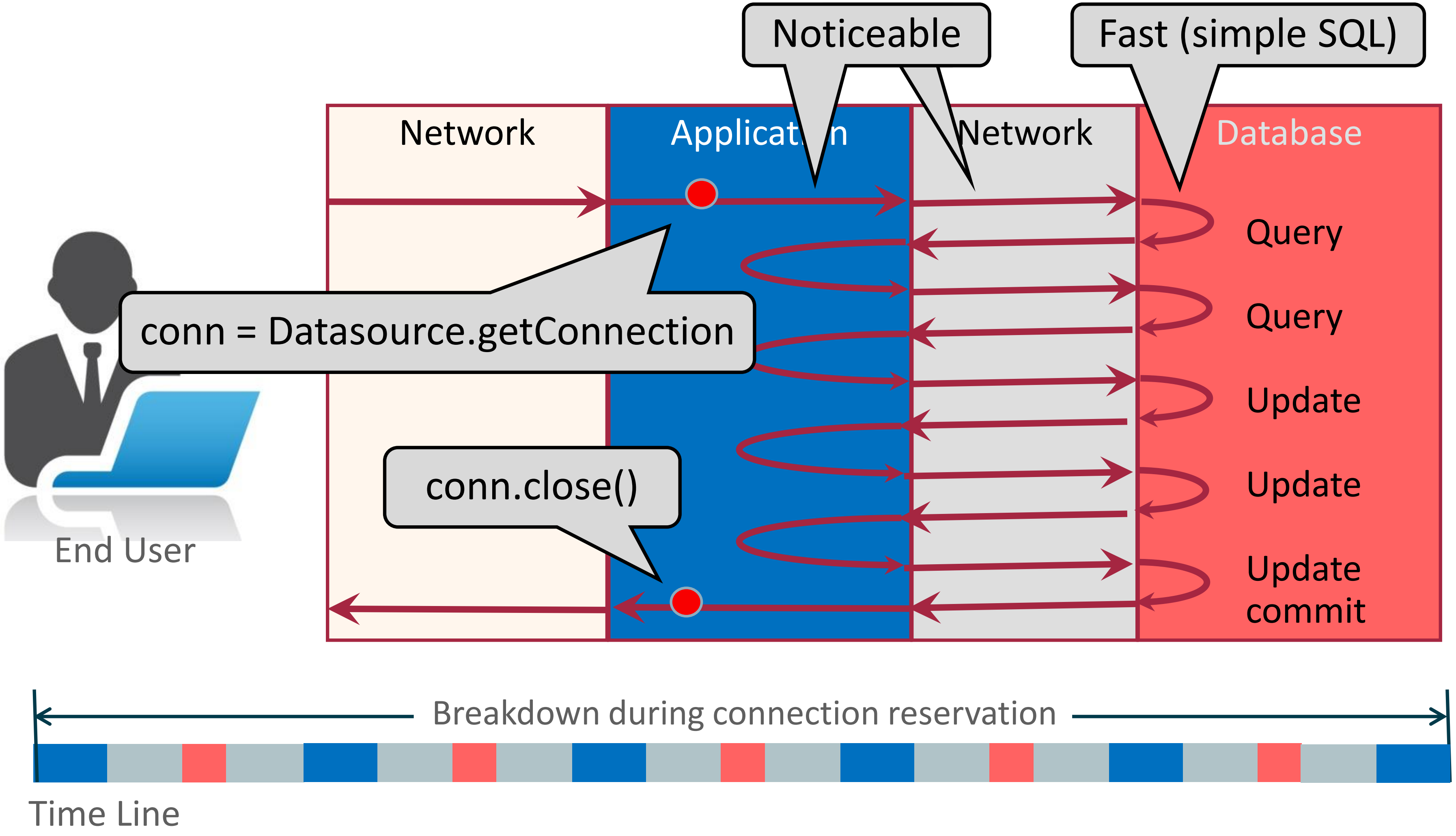
- What is the appropriate size for my connection pool?
- So that:
  - Database is near CPU oversubscription
  - And application threads are willing to queue

- Too small connection pool size will cause database to have unused capacity
- Too large connection pool size will get us in database oversubscription land

ORACLE

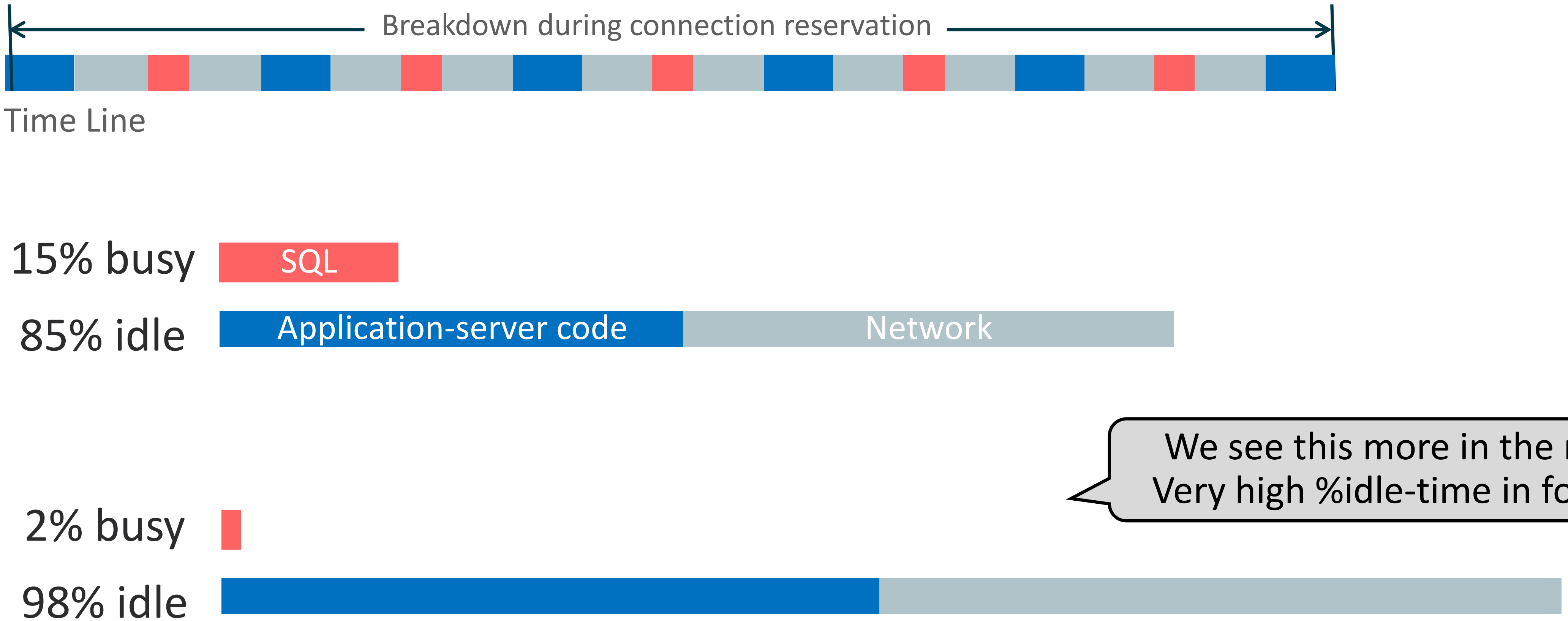
ORACLE
REAL-WORLD PERFORMANCE

What Is Your Foreground Doing?



ORACLE

Breakdown of Your Foreground Session Time



We see this more in the real world
Very high %idle-time in foregrounds

Implication of High %Idle Time in Foreground Session

- So your SW-architecture keeps database session busy only 2% of time

- What does that imply?
  - You would need 50 sessions to get one DB-core busy (= 50 connections in conn.pool)
  - If you have 32 cores in your database server:
    → You would need 1600 sessions to get all cores busy

- This is assuming that all your DB Time is DB CPU!
  - You likely need even more connections/sessions

ORACLE

Connection Pool Sizing

- 0% FG idle time inside reservation → 1 connection per core required
- 80% FG idle time inside reservation → 5 connections per core required
- 90% FG idle time inside reservation → 10 connections per core required
- 95% FG idle time inside reservation → 20 connections per core required
- 98% FG idle time inside reservation → 50 connections per core required

- Appropriate connection pool size := $\frac{100}{100 - X} * \#cores$

X = % FG Idle Time inside reservation

This drives connection pool size!

Waiting in Mid-Tier Versus Waiting in DB-Tier

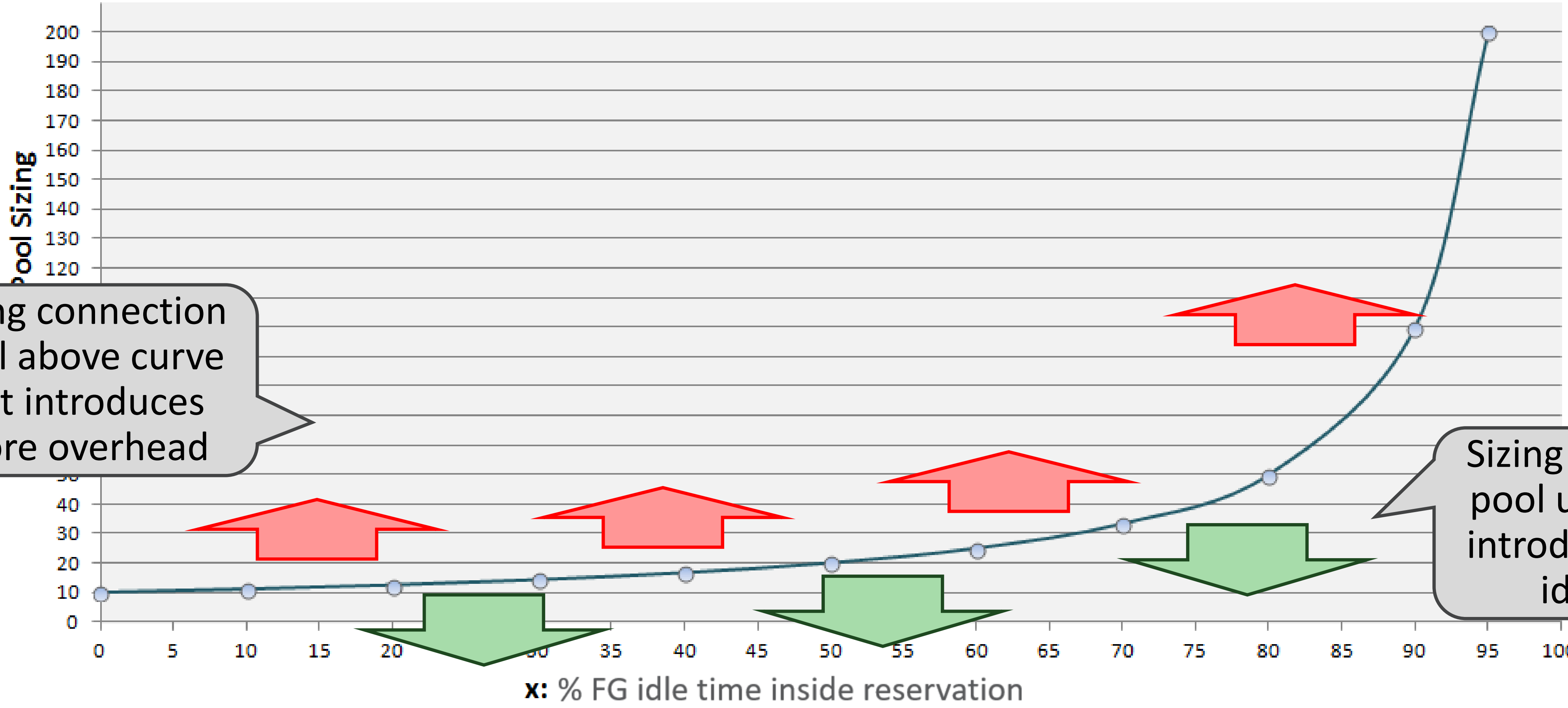
- Again:
  you need to configure threads are willing to queue for connection

- The formula is good starting point for appropriate connection pool size

- Let's plot that curve $$\frac{100}{100 - X} * \#cores$$

Basic Formula Upper Bound Connection Pool Size
10 Core Database Server

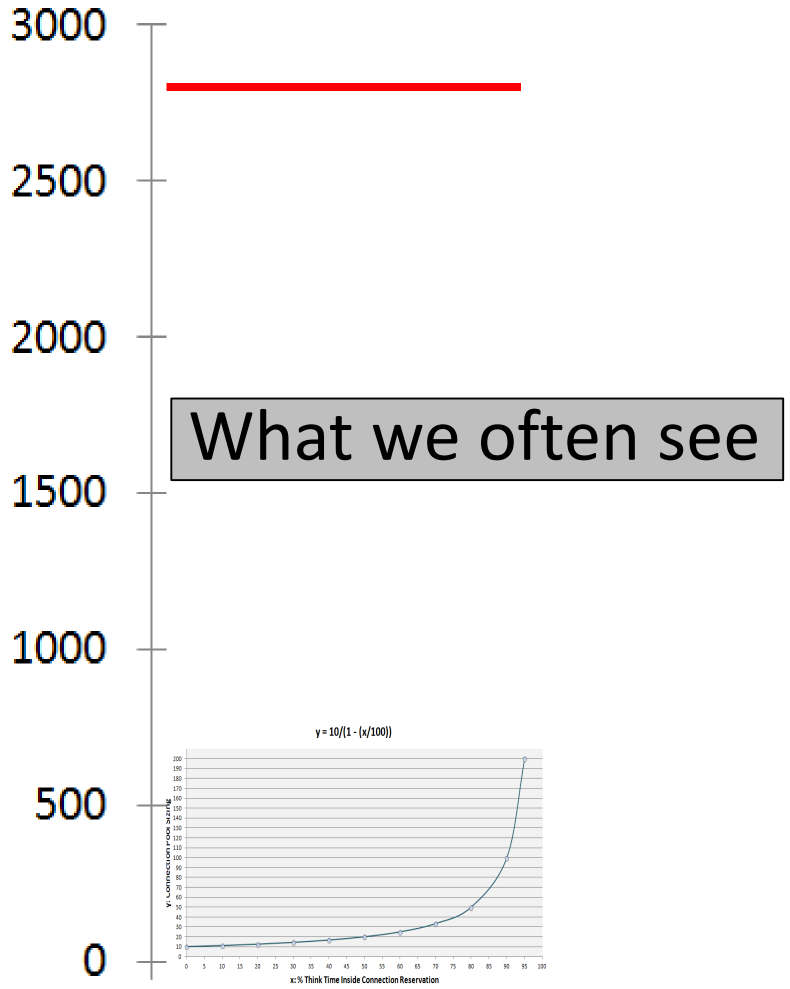
$$y = \frac{100}{100 - X} * \#cores$$



Sizing connection pool above curve just introduces more overhead

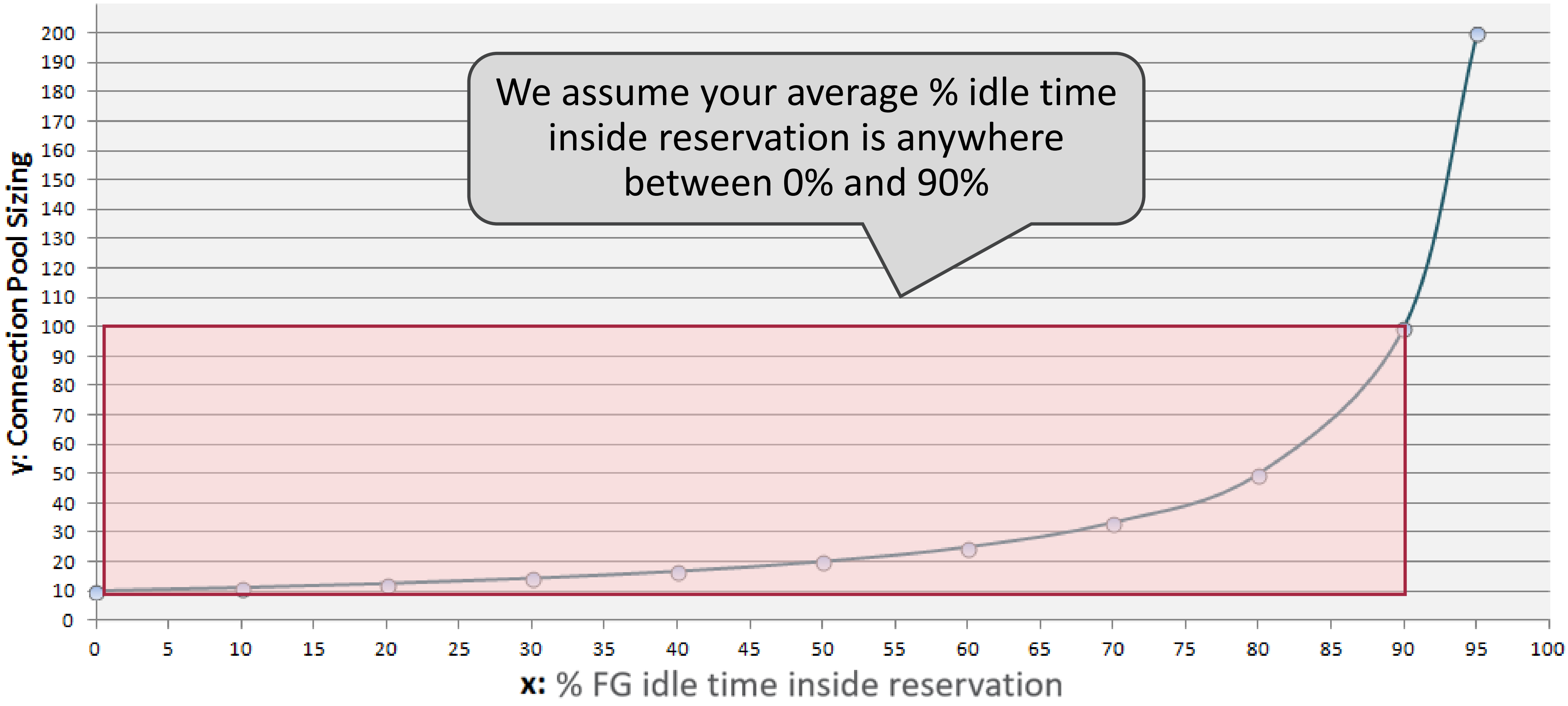
Sizing connection pool under curve introduces server idle time

Connection Pool Sizes in Real-World



Our Rule-of-Thumb: <10 Times Number of Cores

$$y = \frac{100}{100 - X} * \#cores$$



ORACLE

ORACLE REAL-WORLD PERFORMANCE

Why Shrinking Connection Pool Won't Always Work

- If you have substantial %idle time inside connection reservation
  And you are not aware of that

- Shrinking might disable full use of available CPU power on DB-server
  - And so, won't give expected result
  - Your only option then is to change the application and decrease %idle time during reservation

Toon, That's All Very Interesting and All, But …

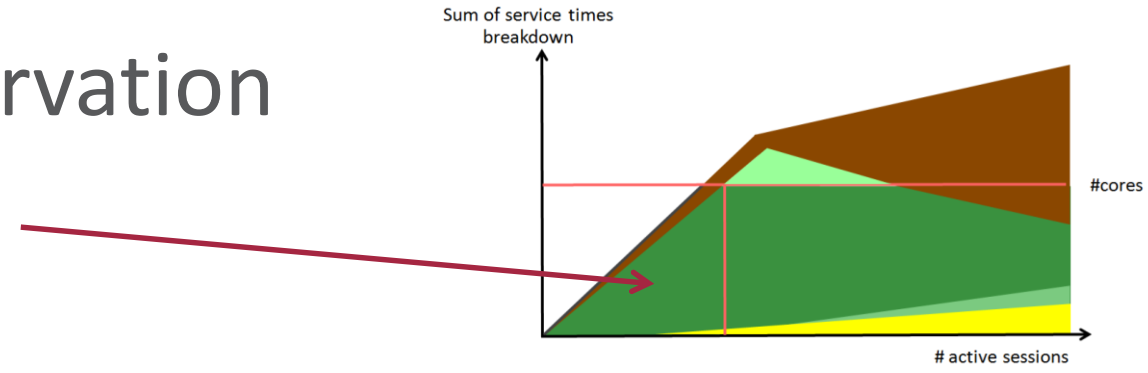
- What the heck is the "%Idle Time Inside Reservation" for my application?

A Challenge For You

- Ideally you'd want this to be instrumented by Java developers in their code
- To create awareness with them too

- However, can you as DBA get a clue?

Finding %Idle Time Inside Reservation



- Assuming you're still in the safe zone
  - Use test-system, or "quiet" prod-system

- Just do a SQL-trace of one of connection-pool sessions for a minute during representative workload and investigate trace-file

- Likely you'll be able to spot "acquire" and "release" events via some repetitive pattern. For example:
  - "Release" typically would be XCTEND (commit or rollback)
  - "Acquire" typically starts with "timestamp" during quiet period

    Or immediately after XCTEND

    Or you can spot it via some initialization statement

ORACLE REAL-WORLD PERFORMANCE

Example Trace File

Acquire connection object

```
*** 2017-02-16 11:22:51.009
PARSE #140293387629512:c=0,e=17,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1534564159,tim=18431411073127
EXEC #140293387629512:c=999,e=824,p=1,cr=4,cu=2,mis=0,r=0,dep=0,og=1,plh=1534564159,tim=18431411074018
FETCH #140293387629512:c=0,e=11,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=1,plh=1534564159,tim=18431411074107
FETCH #140293387629512:c=0,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=1534564159,tim=18431411074389
CLOSE #140293387629512:c=0,e=6,dep=0,type=3,tim=18431411074697
PARSE #140293387610760:c=0,e=13,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=18431411074733
EXEC #140293387610760:c=0,e=97,p=0,cr=1,cu=6,mis=0,r=1,dep=0,og=1,plh=0,tim=18431411074847
CLOSE #140293387610760:c=0,e=3,dep=0,type=3,tim=18431411075068
PARSE #140293387630952:c=0,e=12,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1421812382,tim=18431411075102
EXEC #140293387630952:c=0,e=66,p=0,cr=4,cu=1,mis=0,r=1,dep=0,og=1,plh=1421812382,tim=18431411075185
CLOSE #1…0952:c=0,e=3,dep=0,type=3,tim=18431411075339
XCTEND … only=0, tim=18431411075364
```

Release

Acquire

```
*** 2017-02-16 11:22:59.999
PARSE #140293387629512:c=0,e=33,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1534564159,tim=18431420062989
EXEC #140293387629512:c=0,e=888,p=1,cr=4,cu=3,mis=0,r=0,dep=0,og=1,plh=1534564159,tim=18431420063979
FETCH #140293387629512:c=0,e=13,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=1,plh=1534564159,tim=18431420064082
FETCH #140293387629512:c=0,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=1534564159,tim=18431420064445
CLOSE #140293387629512:c=0,e=10,dep=0,type=3,tim=18431420064780
PARSE #140293387610760:c=0,e=14,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=18431420064818
EXEC #140293387610760:c=0,e=188,p=0,cr=1,cu=6,mis=0,r=1,dep=0,og=1,plh=0,tim=18431420065023
CLOSE #140293387610760:c=0,e=4,dep=0,type=3,tim=18431420065306
PARSE #140293387630952:c=0,e=13,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1421812382,tim=18431420065340
EXEC #140293387630952:c=0,e=97,p=0,cr=4,cu=1,mis=0,r=1,dep=0,og=1,plh=1421812382,tim=18431420065453
CLOSE #140293387630952:c=0,e=3,dep=0,type=3,tim=18431420065636
XCTEND … only=0, tim=18431420065668
```

Release

ORACLE REAL-WORLD PERFORMANCE

Approximating %Non-DB-Time Inside Reservation

- Investigate one acquire-release block in trace-file
  - Determine DB-time by summing all ”e=“ values (dep=0 only)
  - Determine Elapsed time from difference between first and last “tim=“ values
    - Add e-value from first tim value, as tim values represent “time when completed”

- %Non-DB-time-inside-reservation is: ((Elapsed – DB-Time)/Elapsed) * 100

ORACLE

ORACLE
REAL-WORLD PERFORMANCE

Approximating Think-Time-Inside-Reservation

```
*** 2017-02-16 11:22:51.009
PARSE #140293387629512:c=0,e=17,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1534564159,tim=18431411073127
EXEC #140293387629512:c=999,e=824,p=1,cr=4,cu=2,mis=0,r=0,dep=0,og=1,plh=1534564159,ti...=18431411074018
FETCH #140293387629512:c=0,e=11,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=1,plh=1534564159,tim=...8431411074107
FETCH #140293387629512:c=0,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=1534564159,tim=...431411074389
CLOSE #140293387629512:c=0,e=6,dep=0,type=3,tim=18431411074697
PARSE #140293387610760:c=0,e=13,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=1843141...4733
EXEC #140293387610760:c=0,e=97,p=0,cr=1,cu=6,mis=0,r=1,dep=0,og=1,plh=0,tim=1843141...347
CLOSE ...:c=0,e=3,dep=0,type=3,tim=18431411075068
PARSE ...c=0,e=12,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1...411075102
EXEC ...=0,e=66,p=0,cr=4,cu=1,mis=0,r=1,dep=0,og=1,pl...11075185
CLOSE ...c=0,e=3,dep=0,type=3,tim=18431411075339
XCTEND rlbk=0, rd_only=0, tim=18431411075364

*** 2017-02-16 11:22:59.999
PARSE #140293387629512:c=0,e=33,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1534564159,tim=18431420062989
EXEC #140293387629512:c=0,e=888,p=1,cr=4,cu=3,mis=0,r=0,dep=0,og=1,plh=1534564159,tim=18431420063979
FETCH #140293387629512:c=0,e=13,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=1,plh=1534564159,tim=18431420064082
FETCH #140293387629512:c=0,e=1,p=0,c...1420064445
CLOSE #140293387629512:c=0,e=10,dep=...
PARSE #140293387610760:c=0,e=14,p=0,...18
EXEC #140293387610760:c=0,e=188,p=0,...23
CLOSE #140293387610760:c=0,e=4,dep=0,...
PARSE #140293387630952:c=0,e=13,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1421812382,tim=18431420065340
EXEC #140293387630952:c=0,e=97,p=0,cr=4,cu=1,mis=0,r=1,dep=0,og=1,plh=1421812382,tim=18431420065453
CLOSE #140293387630952:c=0,e=3,dep=0,type=3,tim=18431420065636
XCTEND rlbk=0, rd_only=0, tim=18431420065668
```

Sum is:
1,053us

Difference is:
2,237us + 17us
= 2,254us

((2,254 – 1,053)/2,254)*100% =
53% approximate non-DB-time inside reservation
→ Connection pool size about 2X number of cores

A Word on Batch Programs

- Context so far has been: web applications with many browser users

- Batch programs:
  - Developers usually create some kind of do-it-yourself parallelism
  - Configurable number of threads to get work done
  - We see comparable behavior of these threads wrt. connection pool usage
    - They loop and do a transaction per iteration
    - For each transaction they acquire/release a connection
  - Same math applies

Recommendations Connection Pool Sizing

- It all starts by knowing your (average) %Idle Time of foregrounds

ORACLE

ORACLE
REAL-WORLD PERFORMANCE

Recommendations Connection Pool Sizing

- It all starts by knowing your (average) %Idle Time of foregrounds
- Assuming CPU-bound, formula is a good starting point
- Set minimum/maximum/initial # of connections, all to same value
  - And configure threads to wait for connection to become available

ORACLE

ORACLE
REAL-WORLD PERFORMANCE

Recommendations Connection Pool Sizing

- It all starts by knowing your (average) %Idle Time of foregrounds
- Assuming CPU-bound, formula is a good starting point
- Set minimum/maximum/initial # of connections, all to same value
  - And configure threads to wait for connection to become available
- As DBA you can maybe decrease network-time component to get lower %idle time

ORACLE

ORACLE REAL-WORLD PERFORMANCE

Recommendations Connection Pool Sizing

- It all starts by knowing your (average) %Idle Time of foregrounds
- Assuming CPU-bound, formula is a good starting point
- Set minimum/maximum/initial # of connections, all to same value
  - And configure threads to wait for connection to become available
- As DBA you can maybe decrease network-time component to get lower %idle time
- On your next application development effort try to be aware, or better in control, of Acquire/Release cycles, and (Java) code execution during these cycles
- Your solution should minimize %Idle Time of foregrounds

Questions?

Twitter: @ToonKoppelaars

Hardware and Software
Engineered to Work Together

ORACLE