Using Performance Insights to Optimize Database Performance

Kyle Hailey
Principal Product Manager
Amazon RDS

Who am I?

- Passionate about performance monitoring
- Worked at Quest on **Spotlight**
- Designed **OEM** perf pages Oracle 10.2
- Designed **DB Optimizer** for Embarcadero/Idera
- Advised on **Lab128**
- Joined Amazon RDS 2 years ago for **Performance Insights**

- Talk is for Amazon RDS but can be achieved with other tools
  - If you use Oracle, much should be familiar. If Postgres or MySQL, then new ideas
  - ASH package for PostgreSQL by Bernard Drouvot
  - Solarwinds DPA for PostgreSQL ? (not sure , maybe PG not good enough till 10)

Agenda

What is Performance Insights?

Sampling

Average active sessions (AAS)

Bottleneck analysis

Exploring Performance Insights

What is Performance Insights?

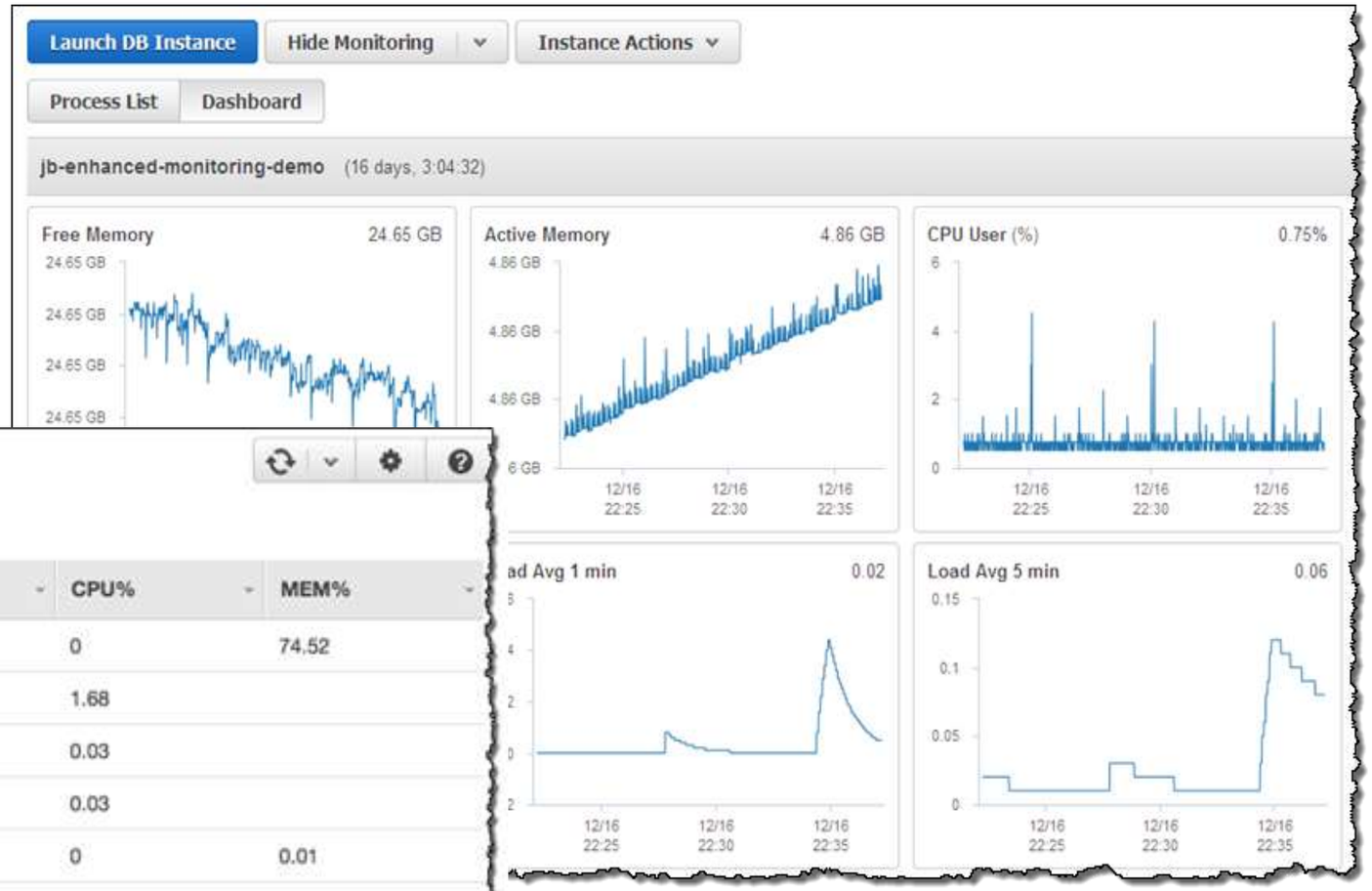
What is Amazon RDS Performance Insights?

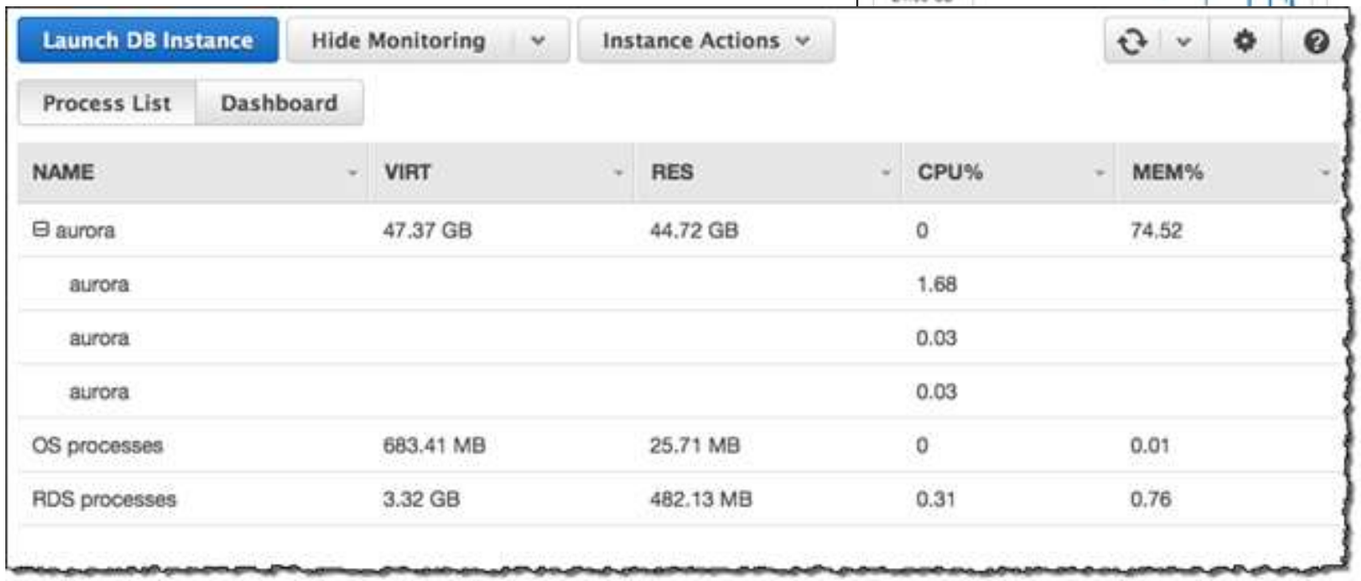
Customers asked for

- Visibility into performance of Amazon Relational Database Service (Amazon RDS) databases
  - Want to optimize cloud database workloads
- Easy tool
  - Often only part-time DBA or no DBA
- Single pane of glass

First step: Amazon RDS Enhanced Monitoring

- Released 2016
  - OS metrics
  - Process/Thread list
  - Up to 1 second granularity





(Lots of graphs)

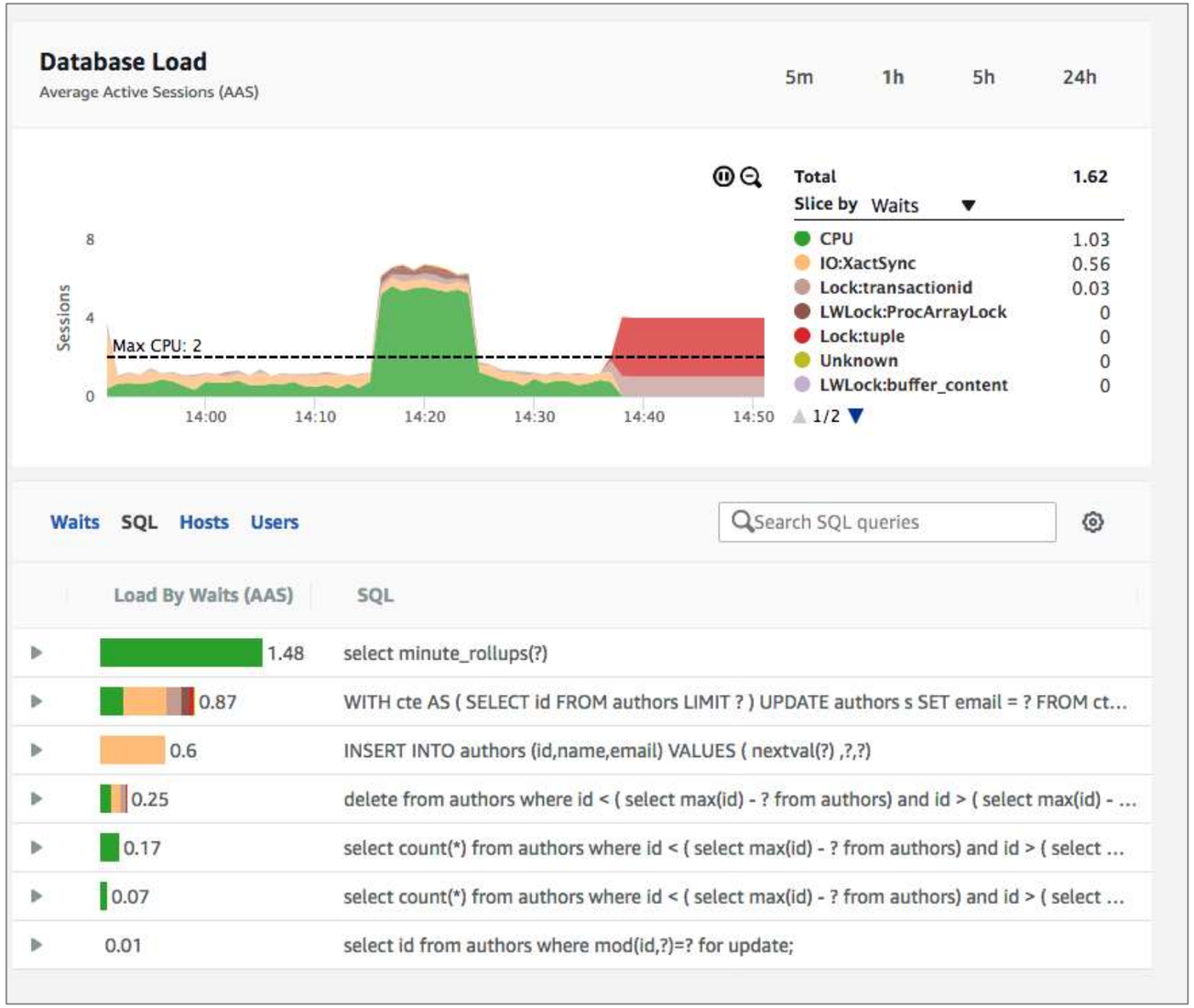
Introducing: Performance Insights

- Dashboard
  - DB load
  - Adjustable timeframe
  - Filterable by attribute (SQL, User, Host, Wait)
  - SQL causing load
- Phased Amazon RDS delivery
  - Aurora, MySQL, PostgreSQL, Oracle, SQL Server, MariaDB
- Guided discovery of performance problems
  - For both beginners & experts
  - Core metric “**database load**”

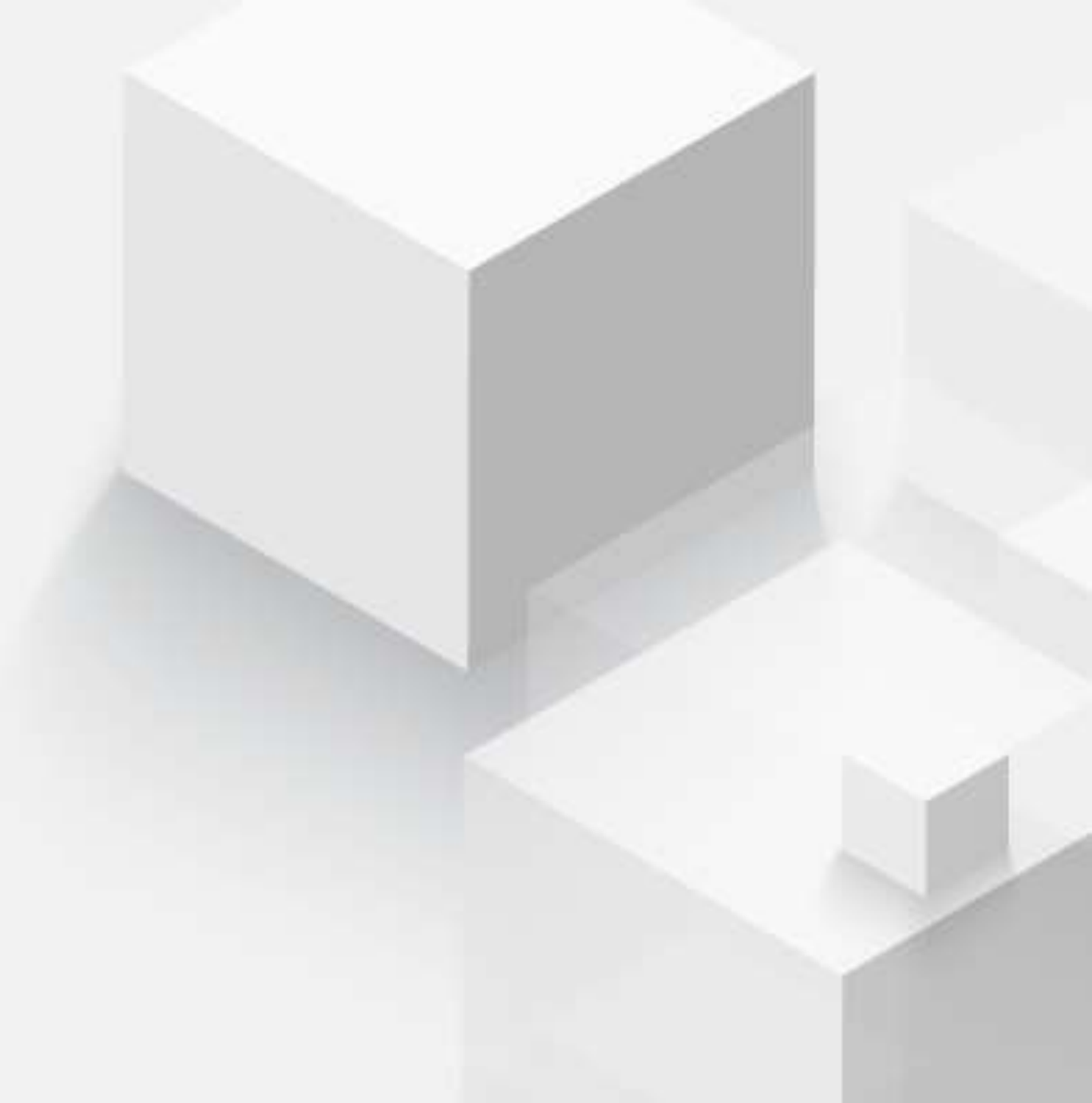
What is “database load”?

- All engines have a connections list showing
  - Active
  - Idle
- We sample every second
  - For each active session, collect
    - SQL
    - State: CPU, I/O, lock, commit log wait, and more ***
      - Key data called “wait event”. PostgreSQL waits became robust in PG 10
    - Host
    - User
- Expose as “average active sessions” (AAS)

Performance Insights dashboard



Sampling



Sampling

Sampling - a light weight powerful method to collect data we can correlate

Sampling Answers:

Is there a bottleneck?

Where does the bottleneck come from?

Why is the bottleneck happening

"It’s this simple: if you don’t sample, you don’t scale.”

– Charity Majors, Honeycomb.io

Sampling on Databases

Connection structure

Status	Who	SQL	Host	State ***
Active	Scott	Select …	10.1.1.25	CPU
Idle	Sue		10.1.1.4	
idle	Joe		10.1.1.251	
idle	Sally		10.1.1.225	
Active	Tony	Update	10.1.1.15	Log Write
Idle	Adam		10.1.1.98	
Idle	Richard		10.1.1.27	

Databases – Active Sessions

PostgreSQL : pg_stat_activity (waits usable in PG 10)

MySQL : performance_schema

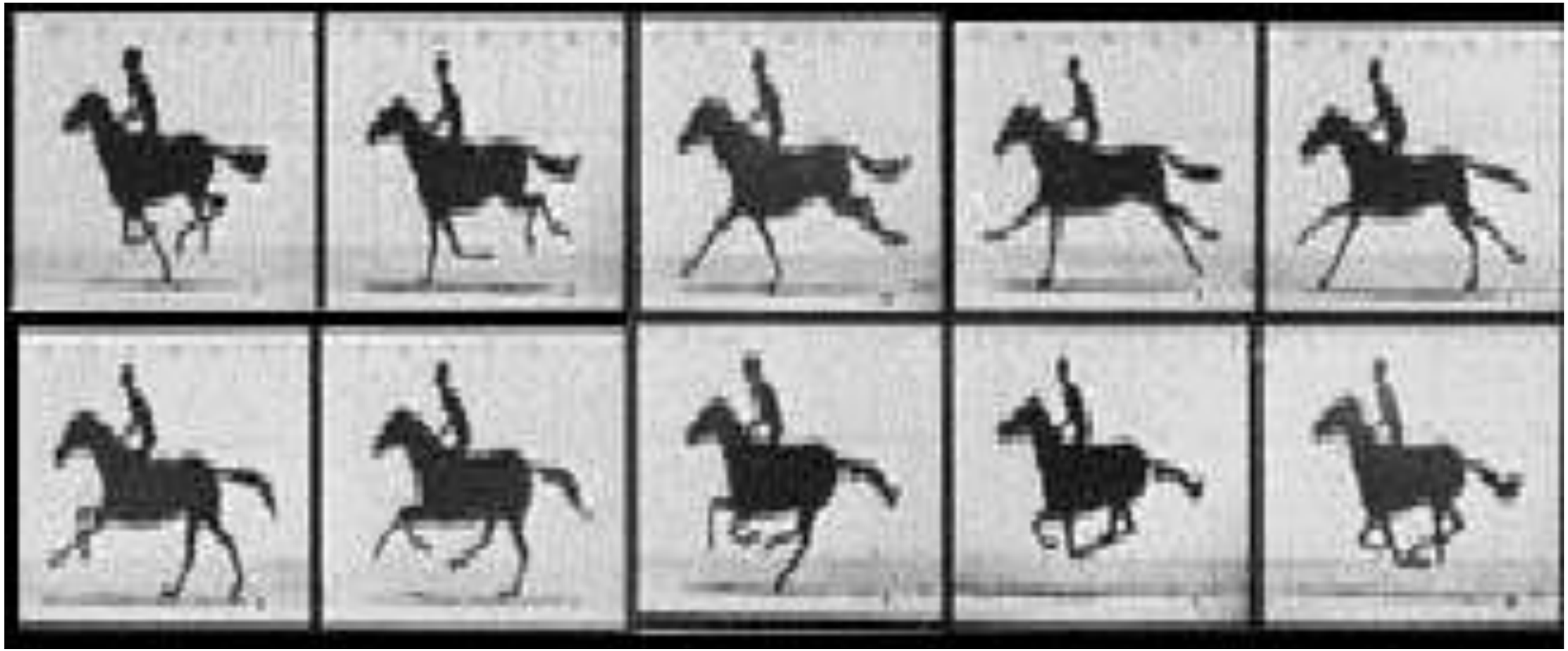
- **threads**
- **waits_current**

SQL Server : master..sysprocesses

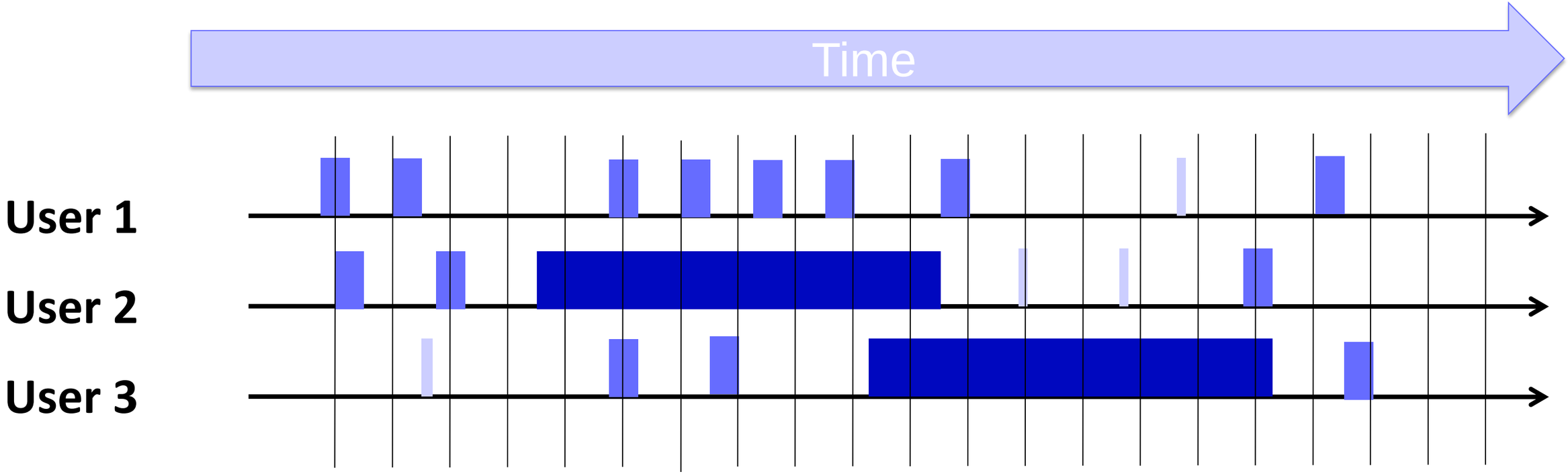
- or
- **sys.dm_exec_sessions**
- **sys.dm_exec_requests**

Oracle : v$session (x$ksuse)

Sampling is like film

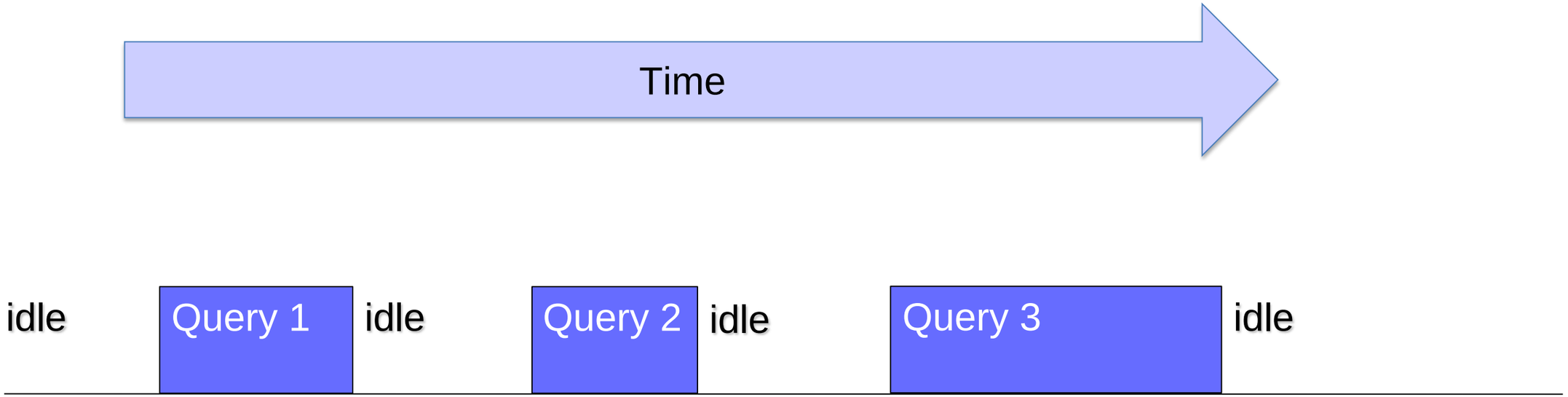


Sampling every second

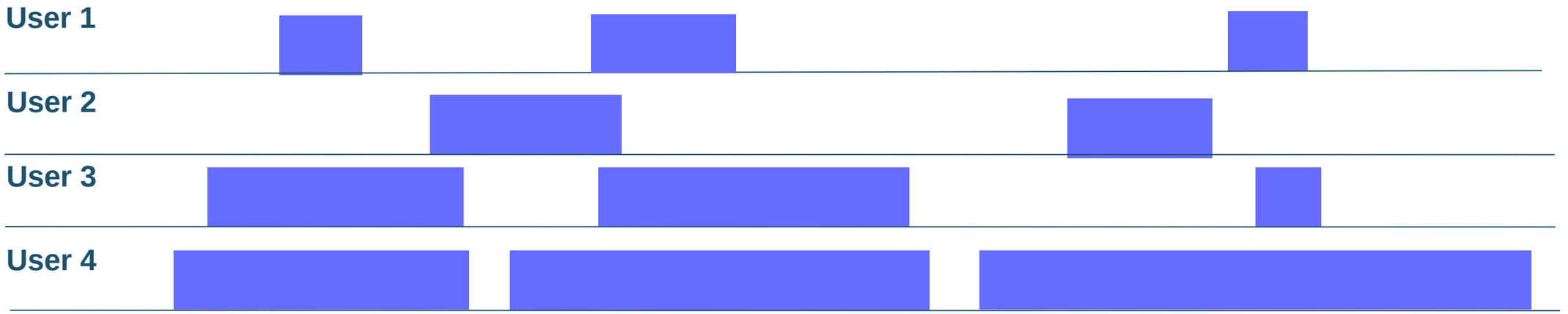


Fast query run rarely
Query run often
Slow query

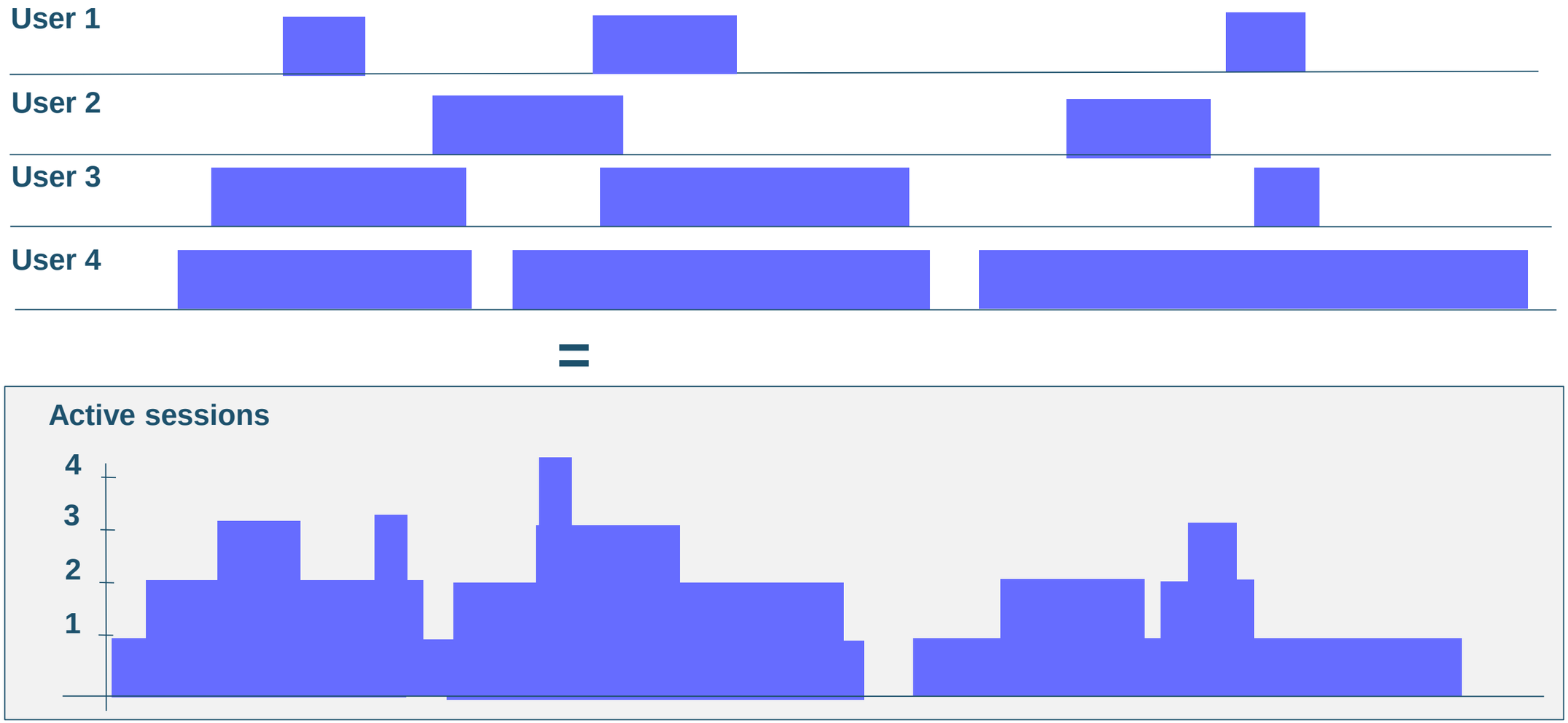
Active session state



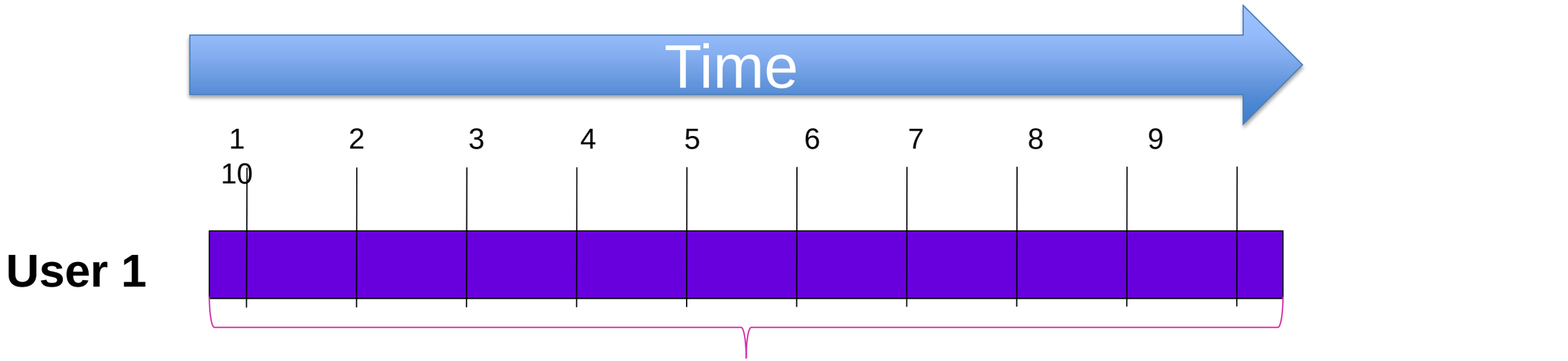
AAS load graph

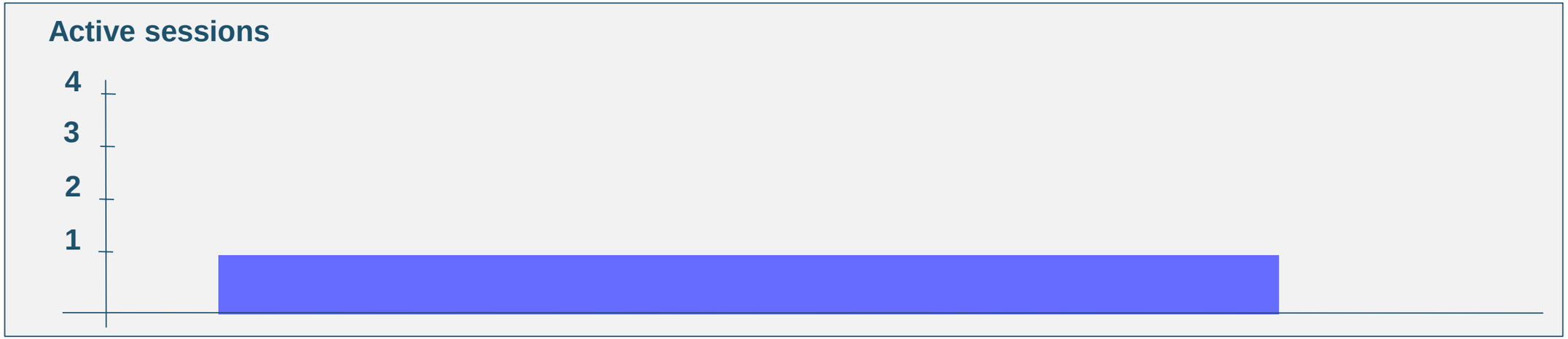


AAS load graph

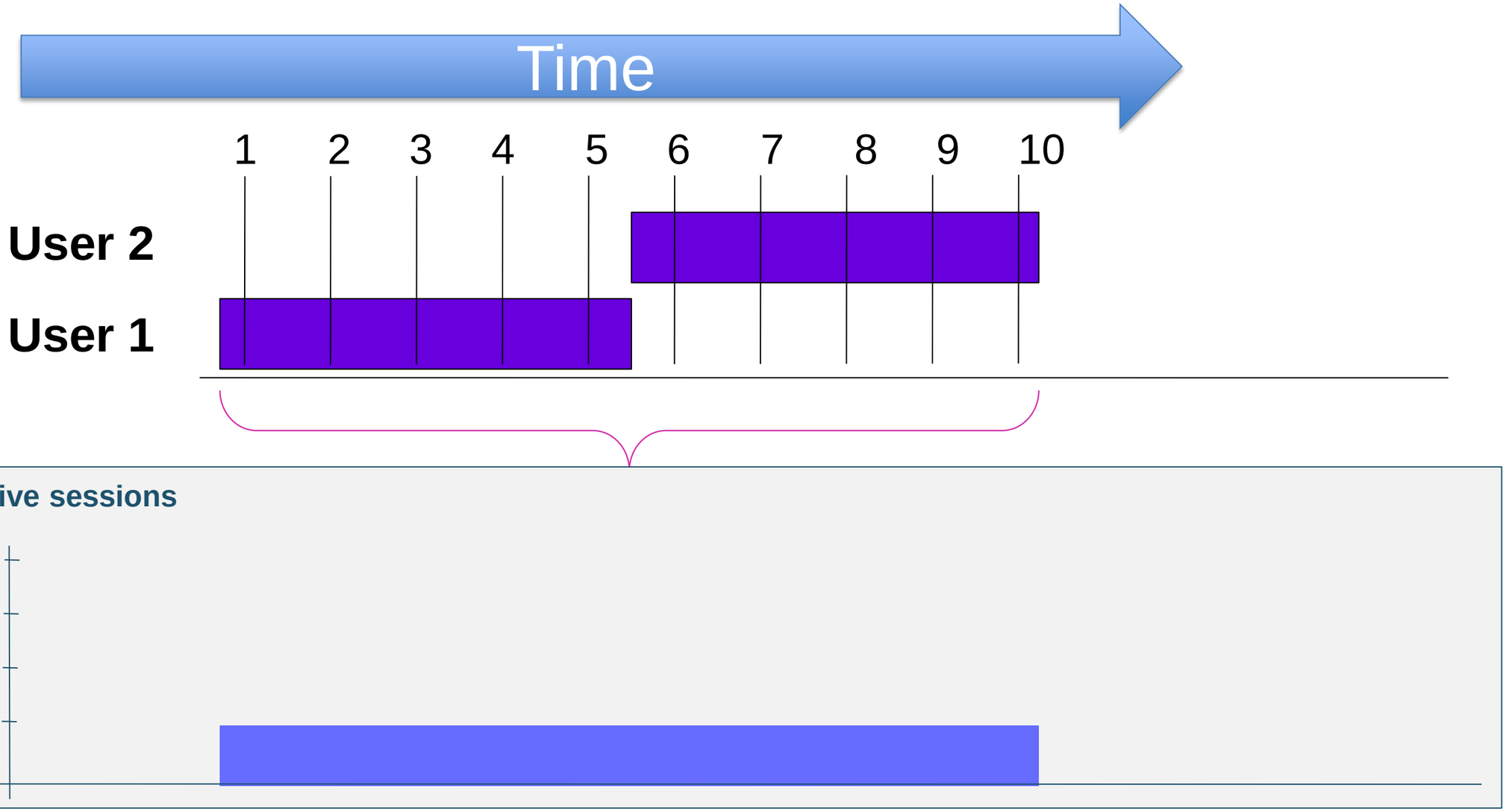


Active session

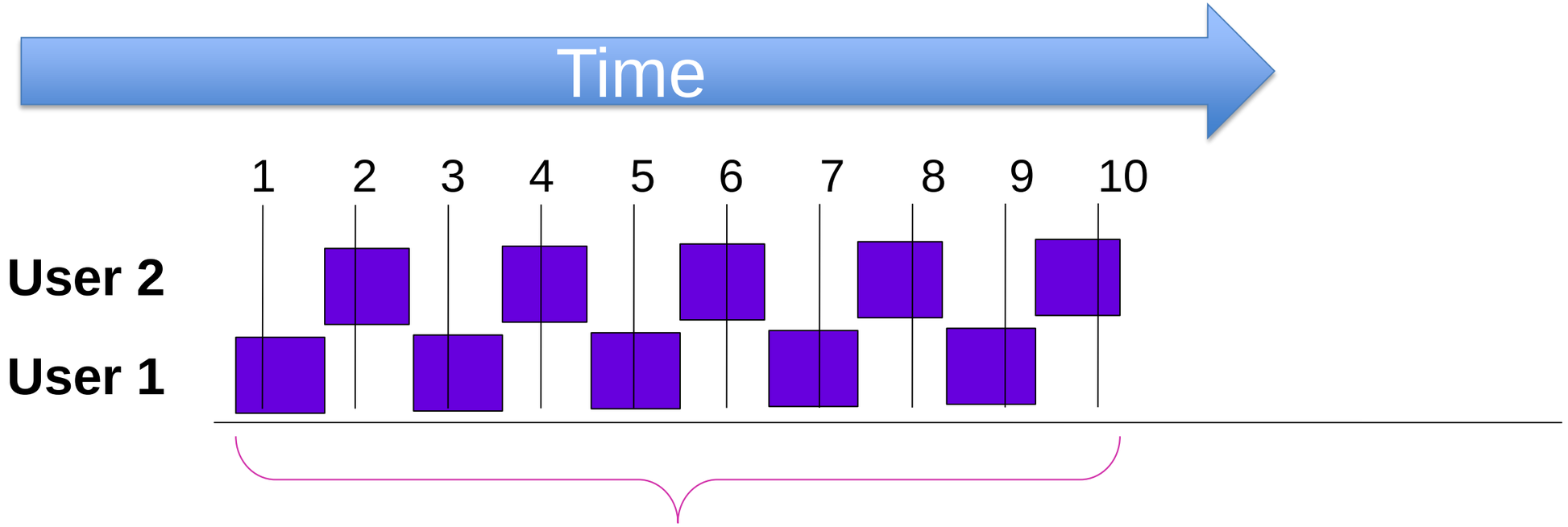


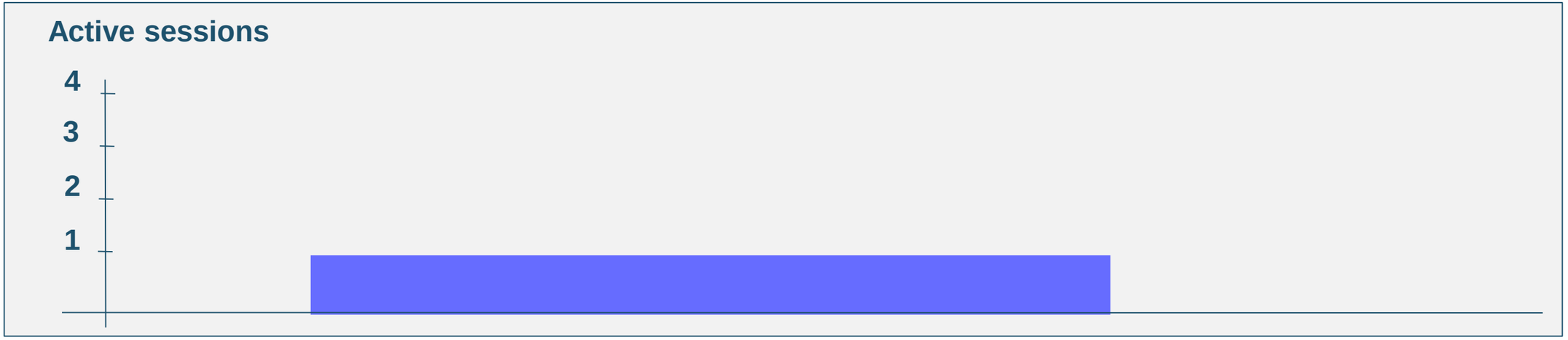


Active session

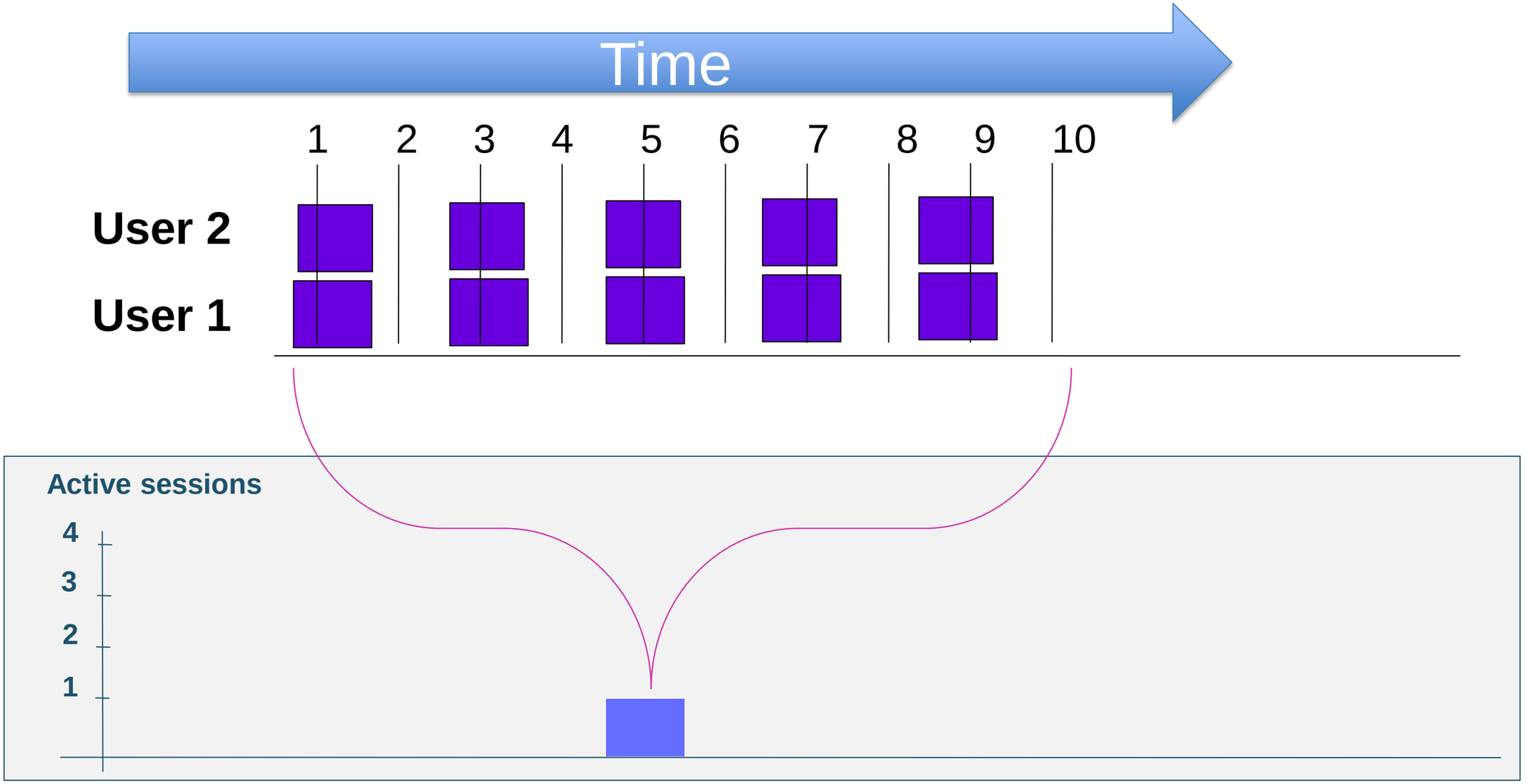


Active session

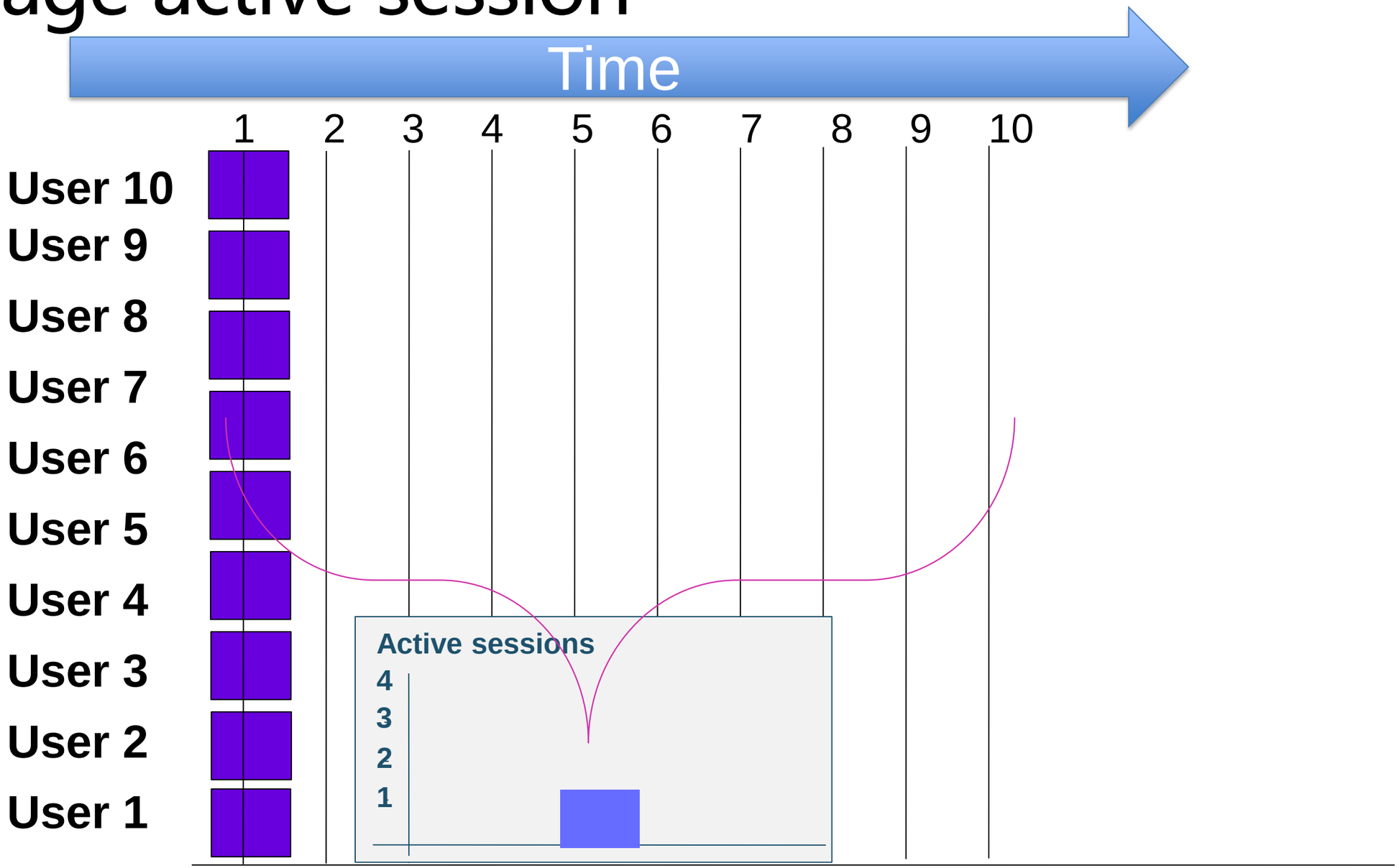




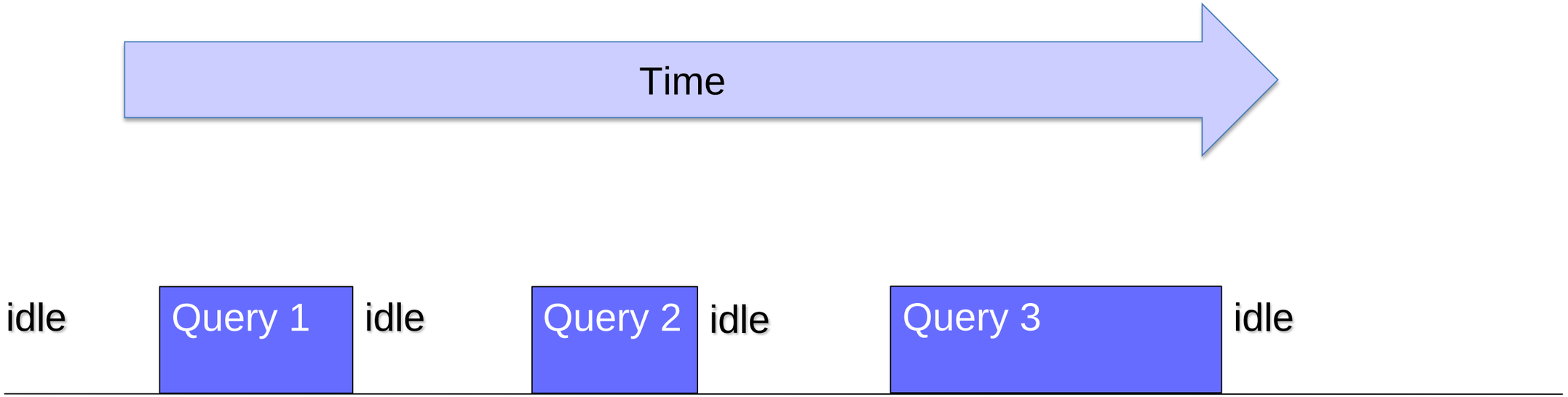
Average active session



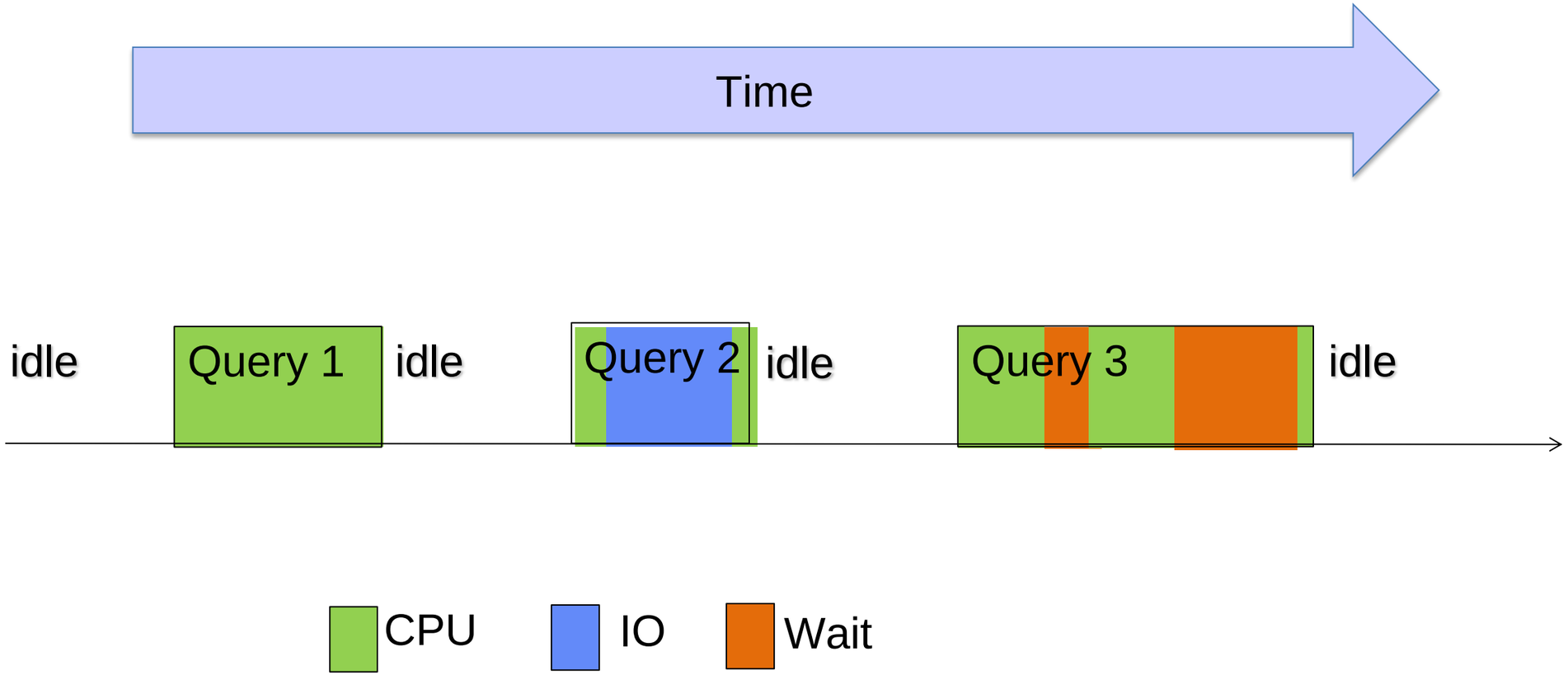
Average active session



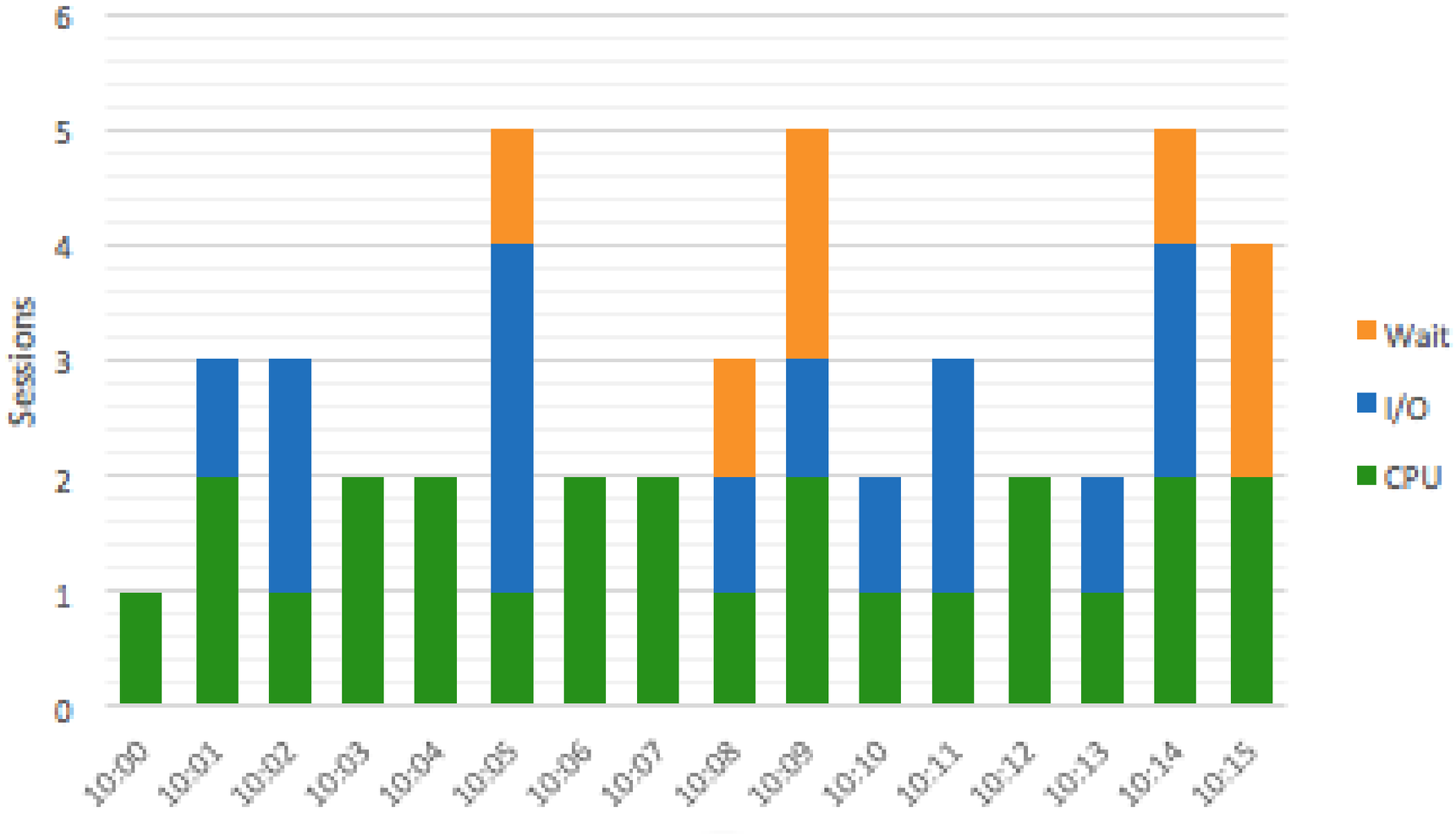
Active session state



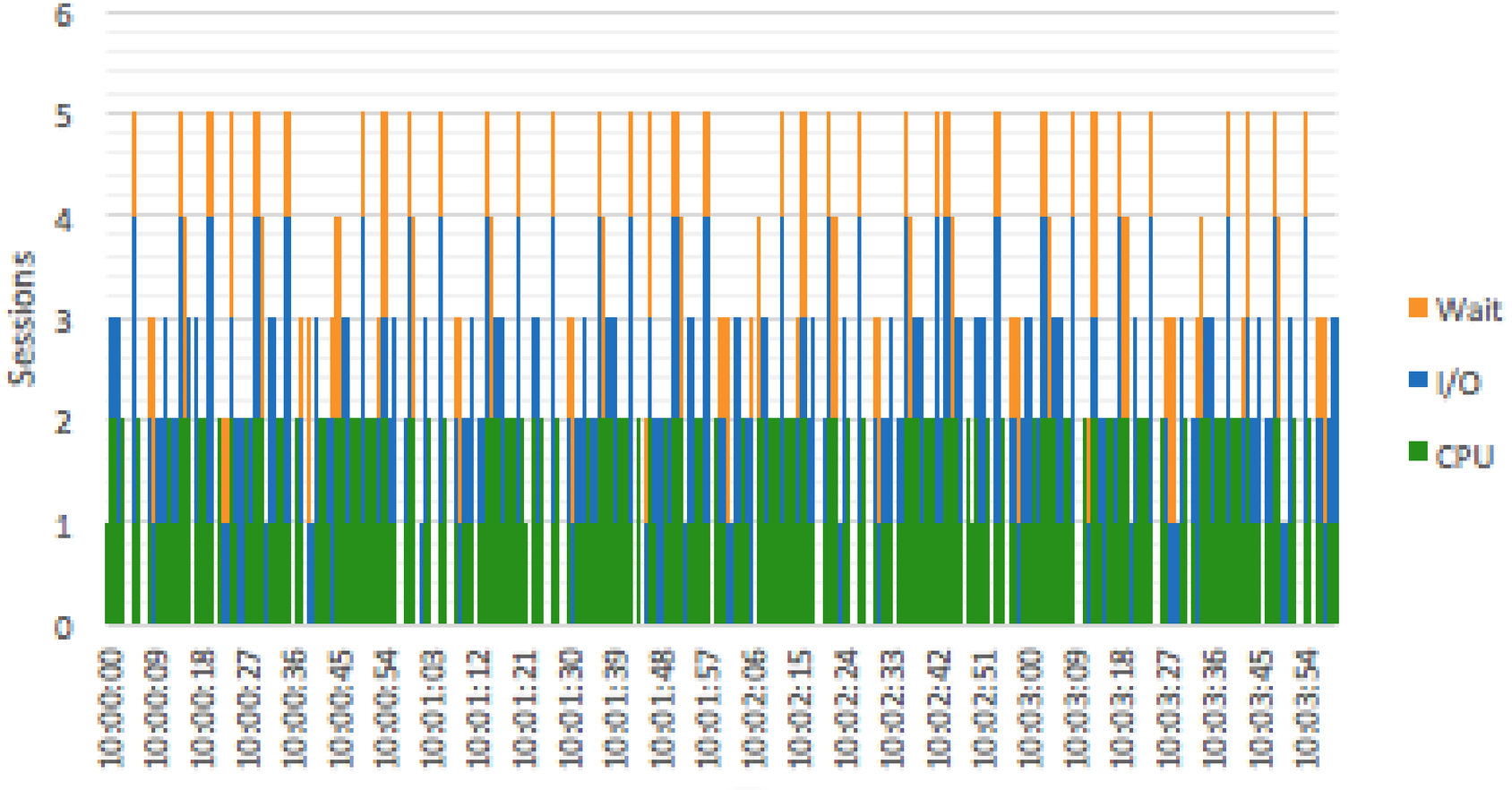
Active session state



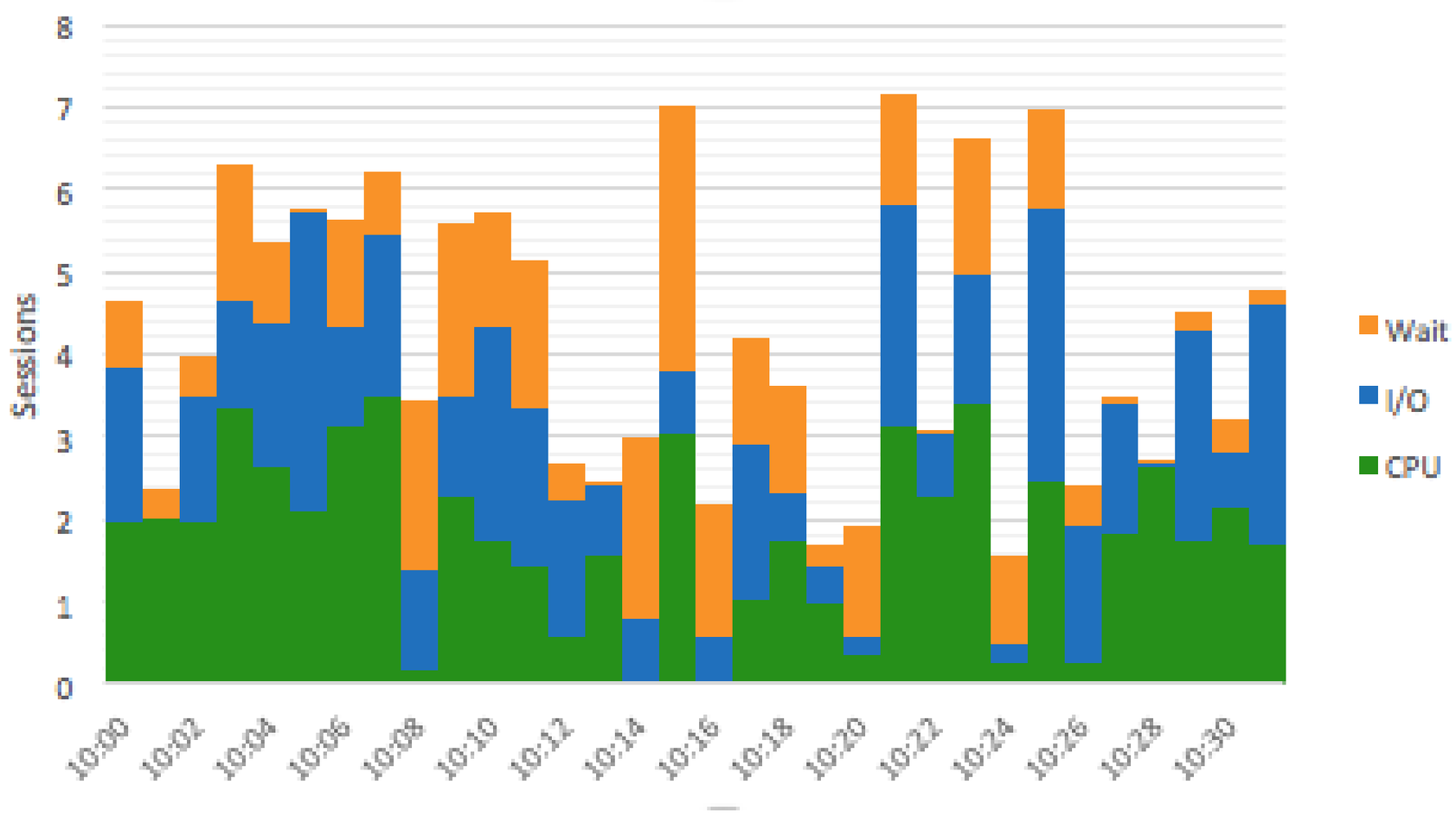
AAS by session state



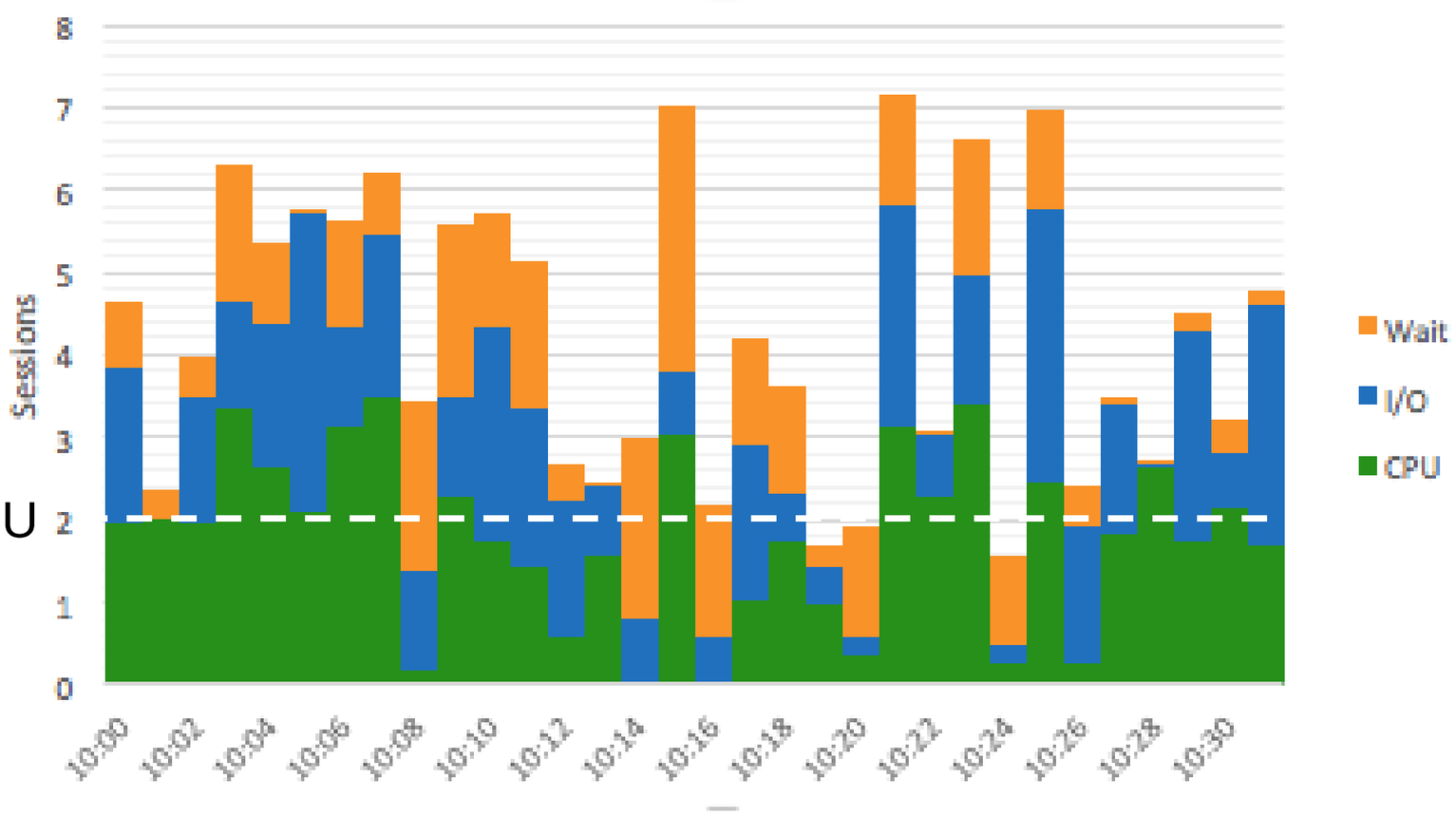
Showing per second samples



AAS over one minute averages



AAS compared to max CPU



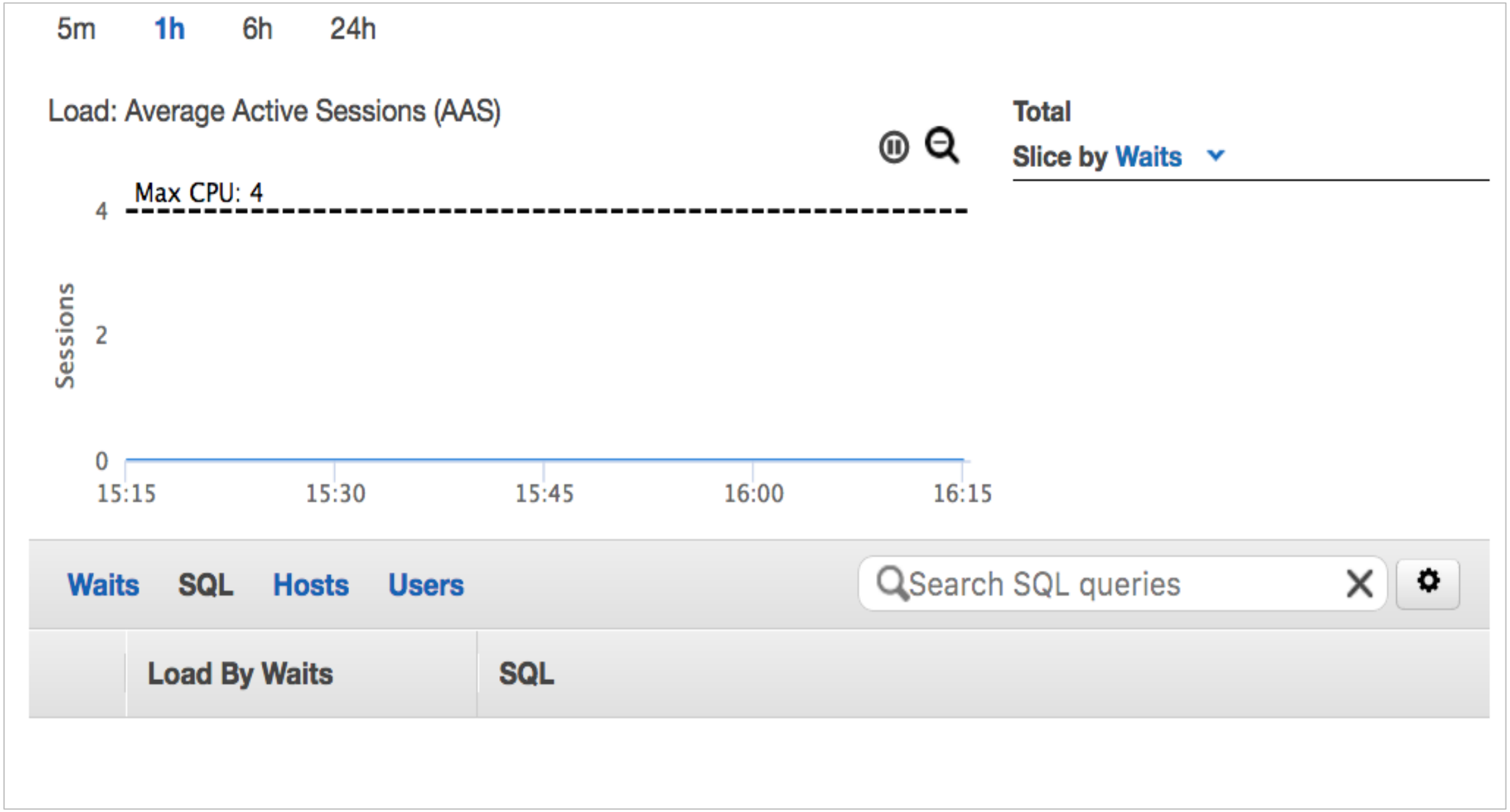
Average active sessions

AAS rules: Using CPU count as yardstick

- ✓ AAS < 1
  - Database is not blocked
- ✓ AAS ~= 0
  - Database basically idle
  - Problems are in the APP not DB
- ✓ AAS < # of CPUs
  - CPU available
  - Are any single sessions 100% active?
- • AAS > # of CPUs
  - Could have performance problems
- ❖ AAS >> # of CPUS
  - There is a bottleneck

When Users say

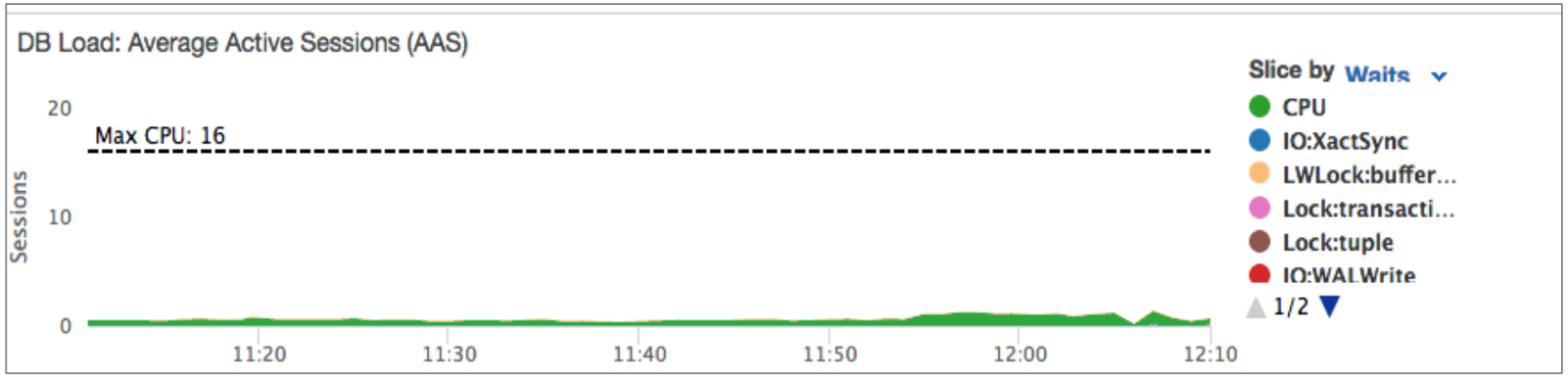
The Database is slow



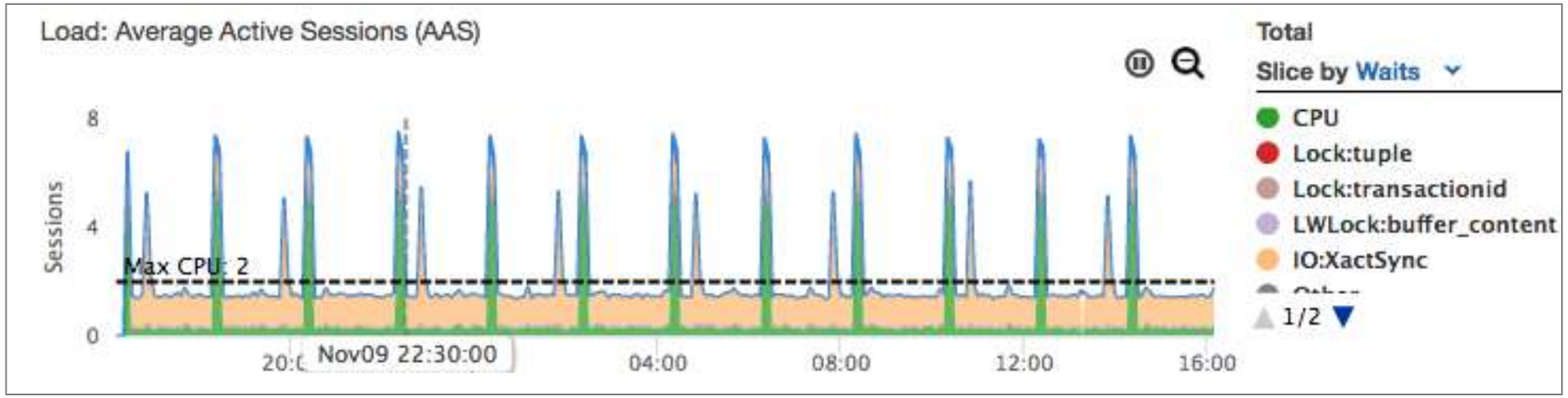
AAS ~= 0

Also useful for sizing

- If load significantly less than #vCPU then oversized

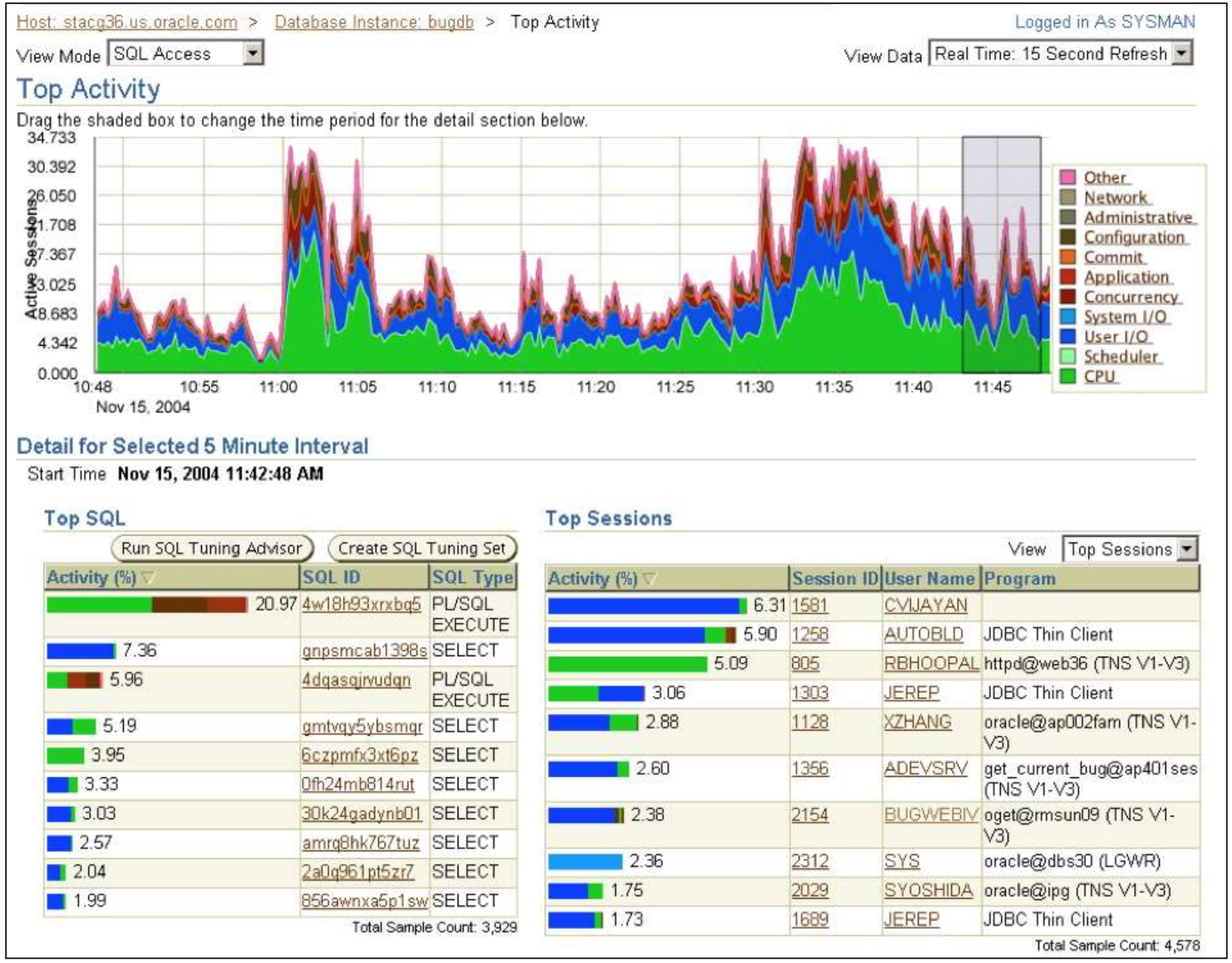


- If load Is > #vCPU undersized

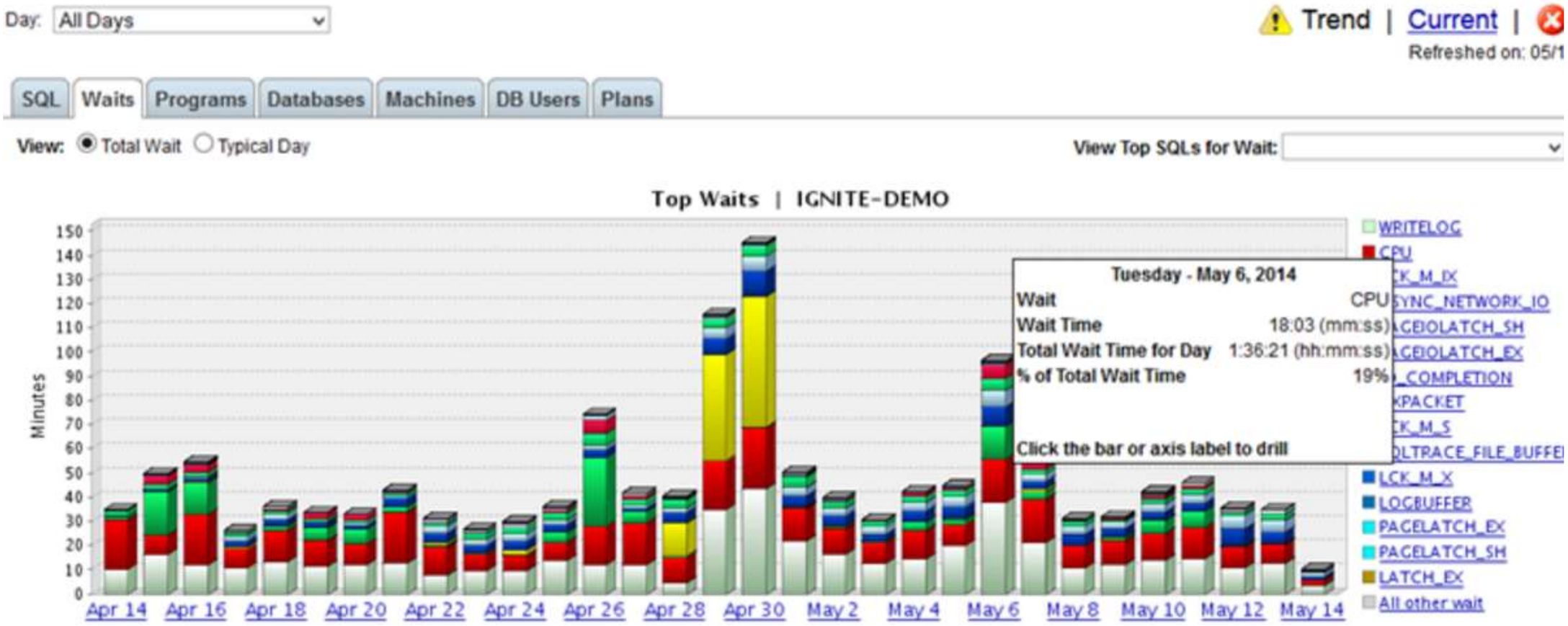


UI in Products

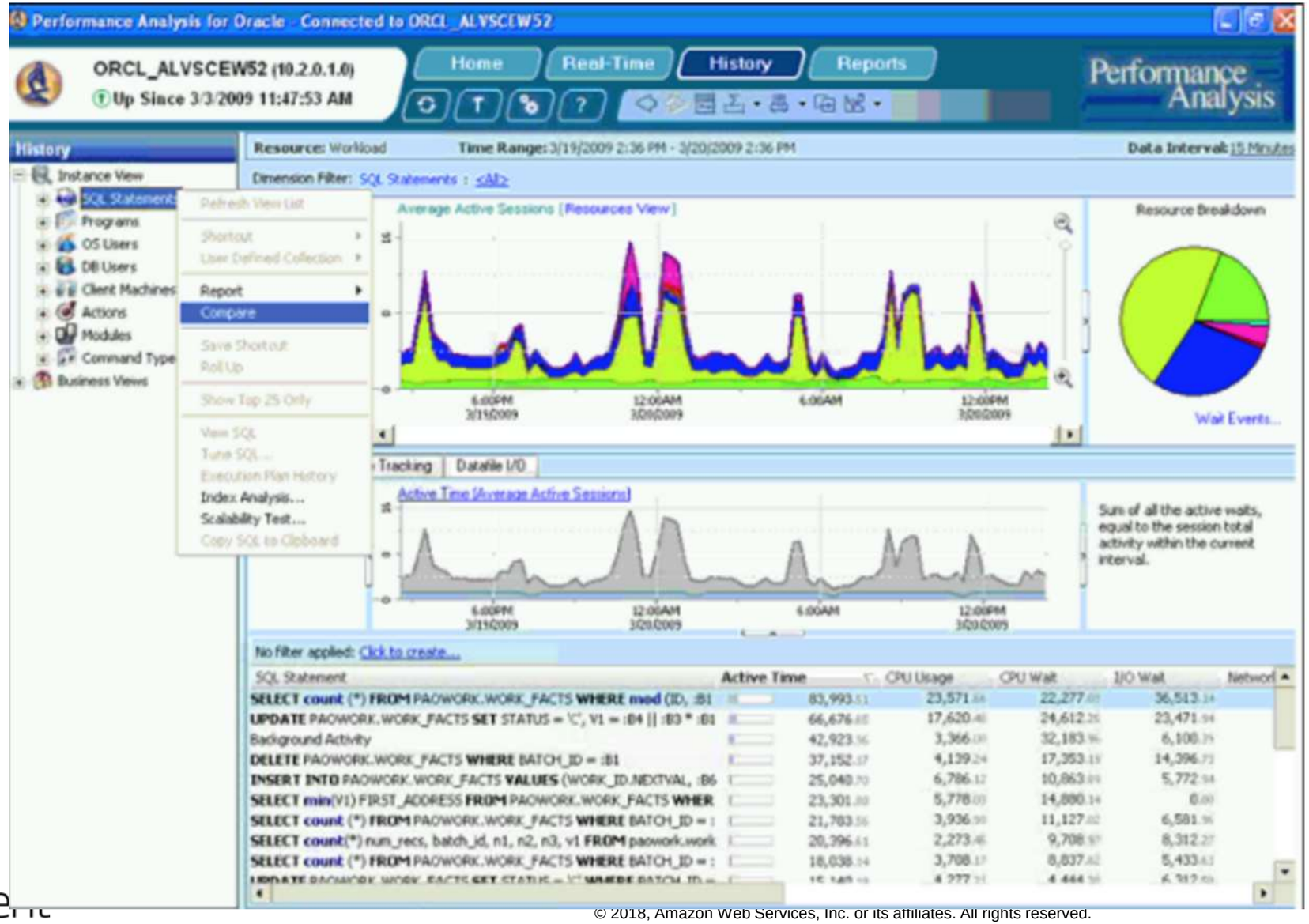
Load - Oracle OEM



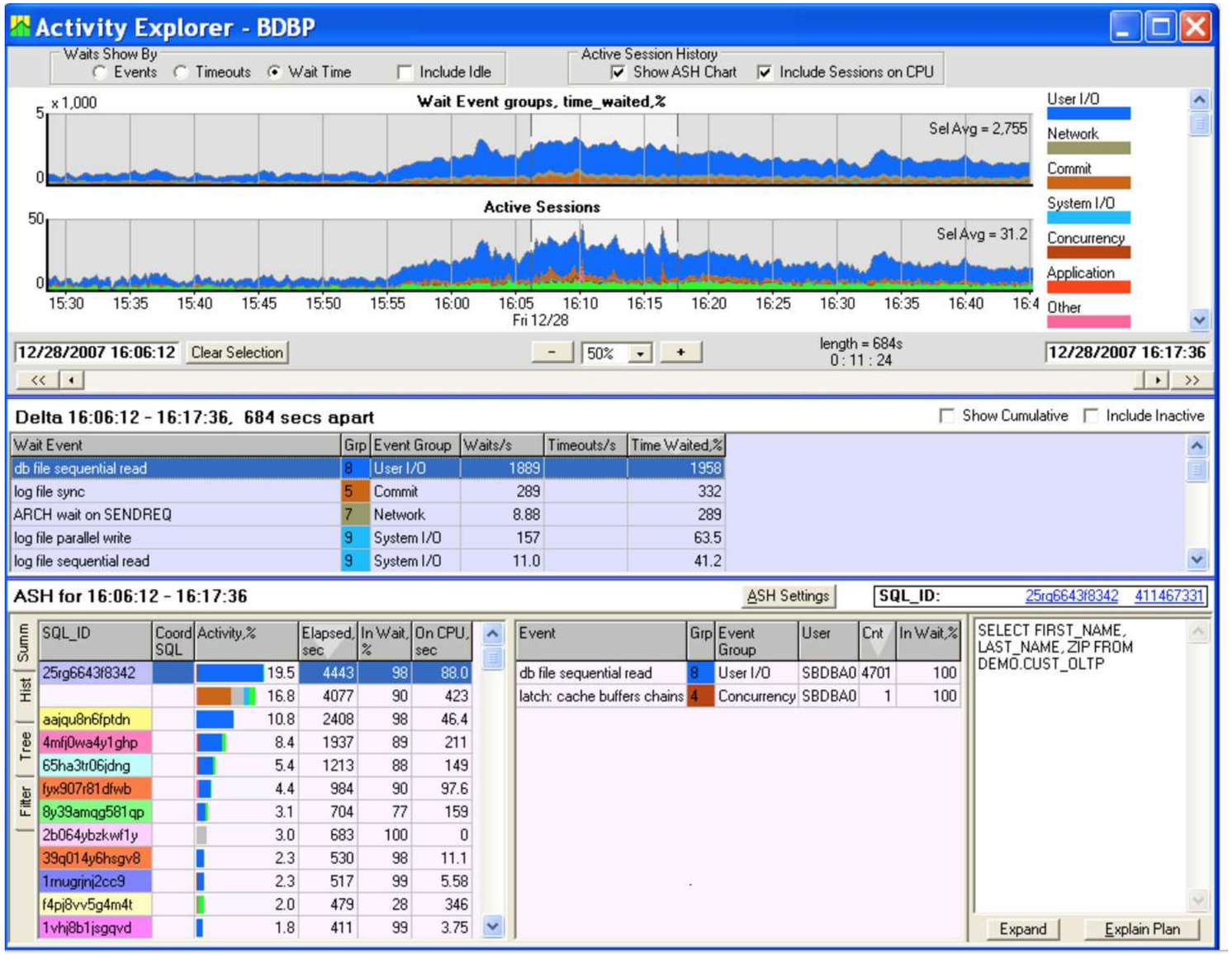
Load - Solarwinds (Confio Ignight) Database Performance Analyzer



Load – Quest Foglight

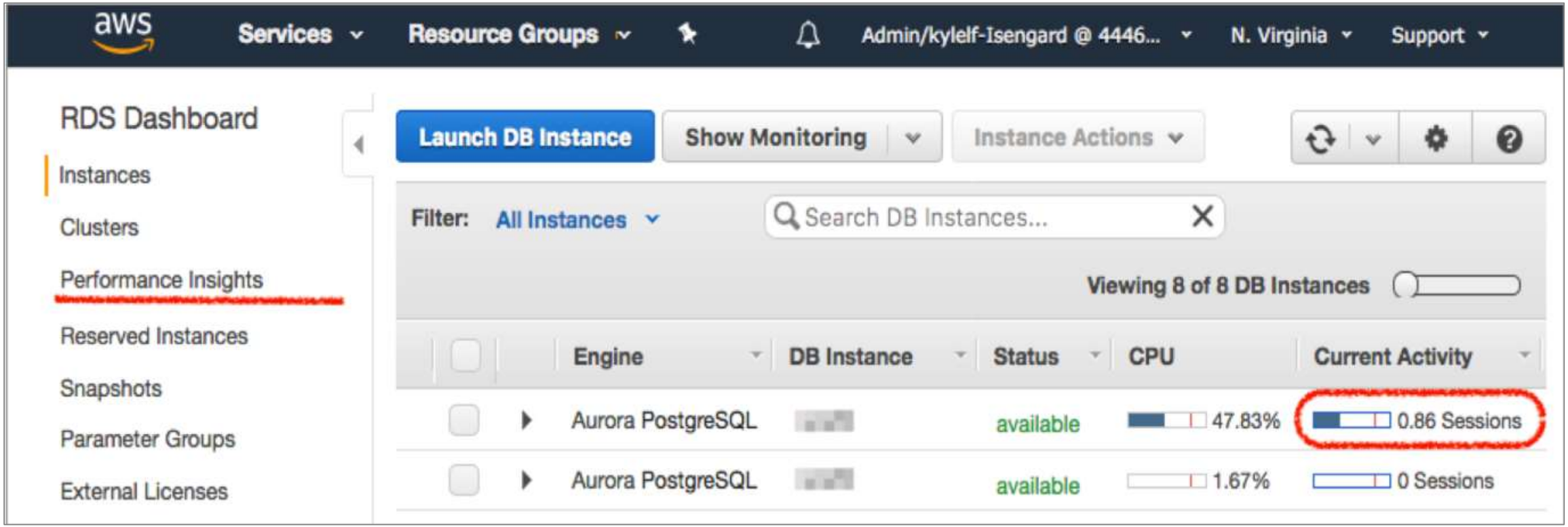


Lab128

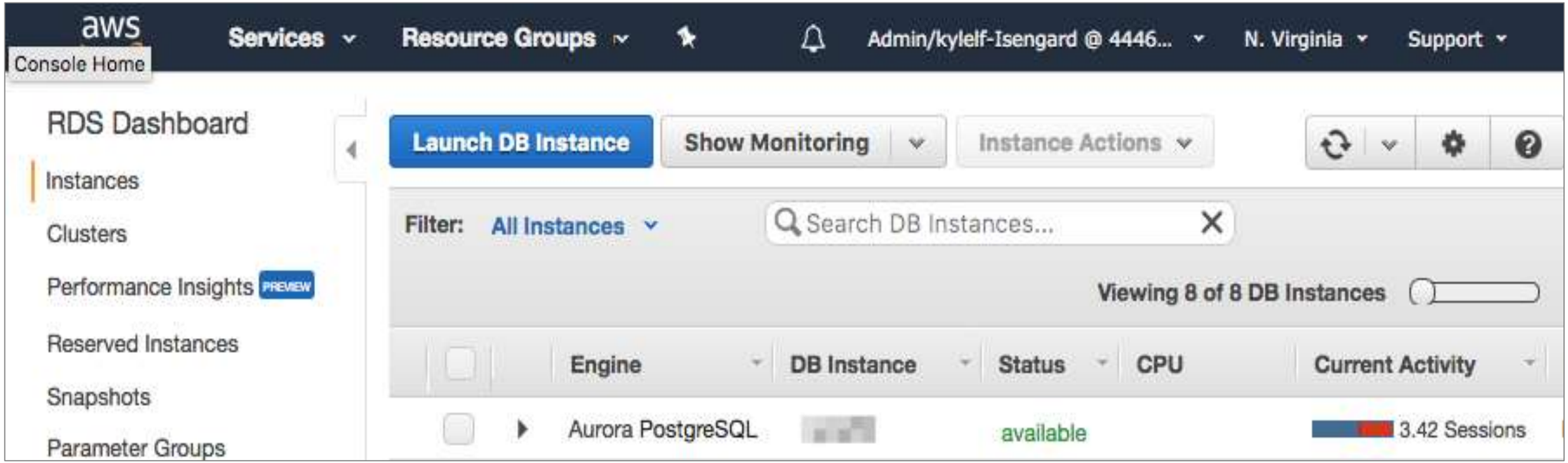

Activity Explorer - BDBP
Waits Show By
Events
Timeouts
Wait Time
Include Idle
Active Session History
Show ASH Chart
Include Sessions on CPU
Wait Event groups, time_waited,%
x 1,000
Sel Avg = 2,755
Active Sessions
Sel Avg = 31.2
User I/O
Network
Commit
System I/O
Concurrency
Application
Other
15:30 15:35 15:40 15:45 15:50 15:55 16:00 16:05 16:10 16:15 16:20 16:25 16:30 16:35 16:40
Fri 12/28
12/28/2007 16:06:12
Clear Selection
50%
length = 684s
0 : 11 : 24
12/28/2007 16:17:36
Delta 16:06:12 - 16:17:36, 684 secs apart
Show Cumulative
Include Inactive
Wait Event | Grp | Event Group | Waits/s | Timeouts/s | Time Waited,%
db file sequential read | 8 | User I/O | 1889 | | 1958
log file sync | 5 | Commit | 289 | | 332
ARCH wait on SENDREQ | 7 | Network | 8.88 | | 289
log file parallel write | 9 | System I/O | 157 | | 63.5
log file sequential read | 9 | System I/O | 11.0 | | 41.2
ASH for 16:06:12 - 16:17:36
ASH Settings
SQL_ID:
25rg6643f8342
411467331
Summ
Hist
Tree
Filter
SQL_ID | Coord SQL | Activity,% | Elapsed, sec | In Wait, % | On CPU, sec
25rg6643f8342 | 19.5 | 4443 | 98 | 88.0
16.8 | 4077 | 90 | 423
aajqu8n6fptdn | 10.8 | 2408 | 98 | 46.4
4mfj0wa4y1ghp | 8.4 | 1937 | 89 | 211
65ha3tr06jdng | 5.4 | 1213 | 88 | 149
fyx907r81dfwb | 4.4 | 984 | 90 | 97.6
8y39amqg581qp | 3.1 | 704 | 77 | 159
2b064ybzkwf1y | 3.0 | 683 | 100 | 0
39q014y6hsgv8 | 2.3 | 530 | 98 | 11.1
1rnugrjnj2cc9 | 2.3 | 517 | 99 | 5.58
f4pj8vv5g4m4t | 2.0 | 479 | 28 | 346
1vhj8b1jsgqvd | 1.8 | 411 | 99 | 3.75
Event | Grp | Event Group | User | Cnt | In Wait,%
db file sequential read | 8 | User I/O | SBDBA0 | 4701 | 100
latch: cache buffers chains | 4 | Concurrency | SBDBA0 | 1 | 100
SELECT FIRST_NAME, LAST_NAME, ZIP FROM DEMO.CUST_OLTP
Expand
Explain Plan

Accessing Performance Insights

Accessing Performance Insights

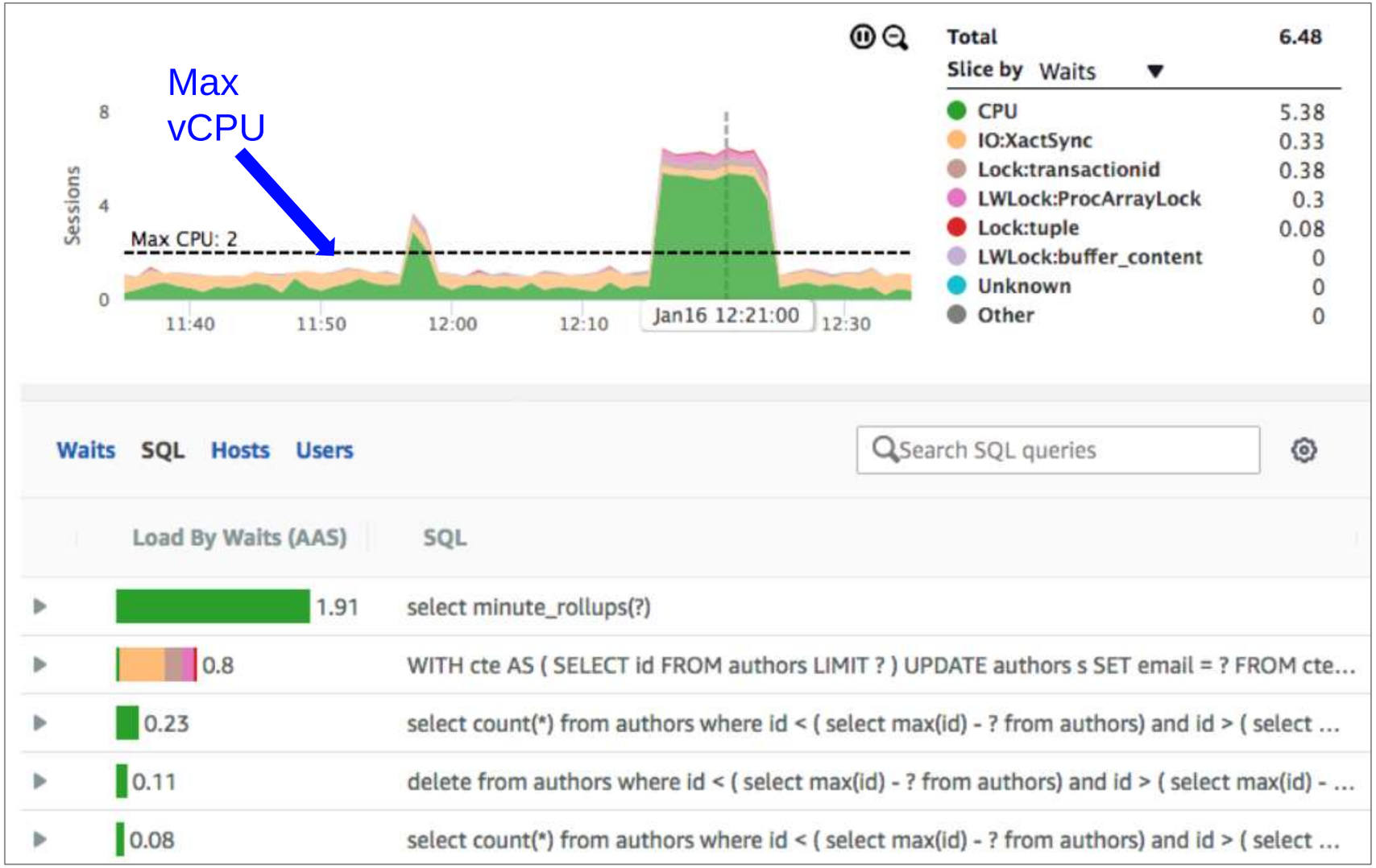


Access to Performance Insights

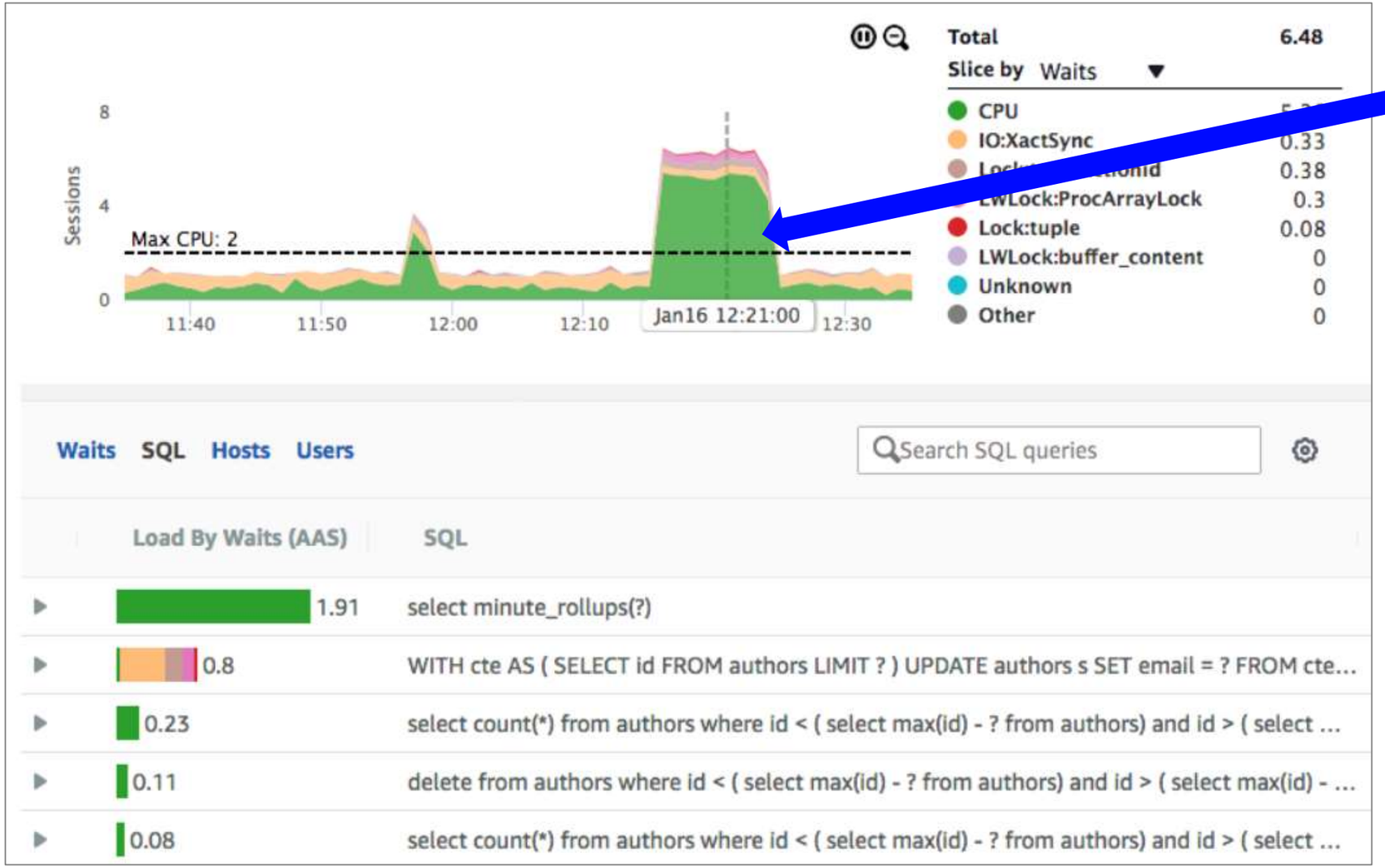


Accessing Performance Insights

Max vCPU

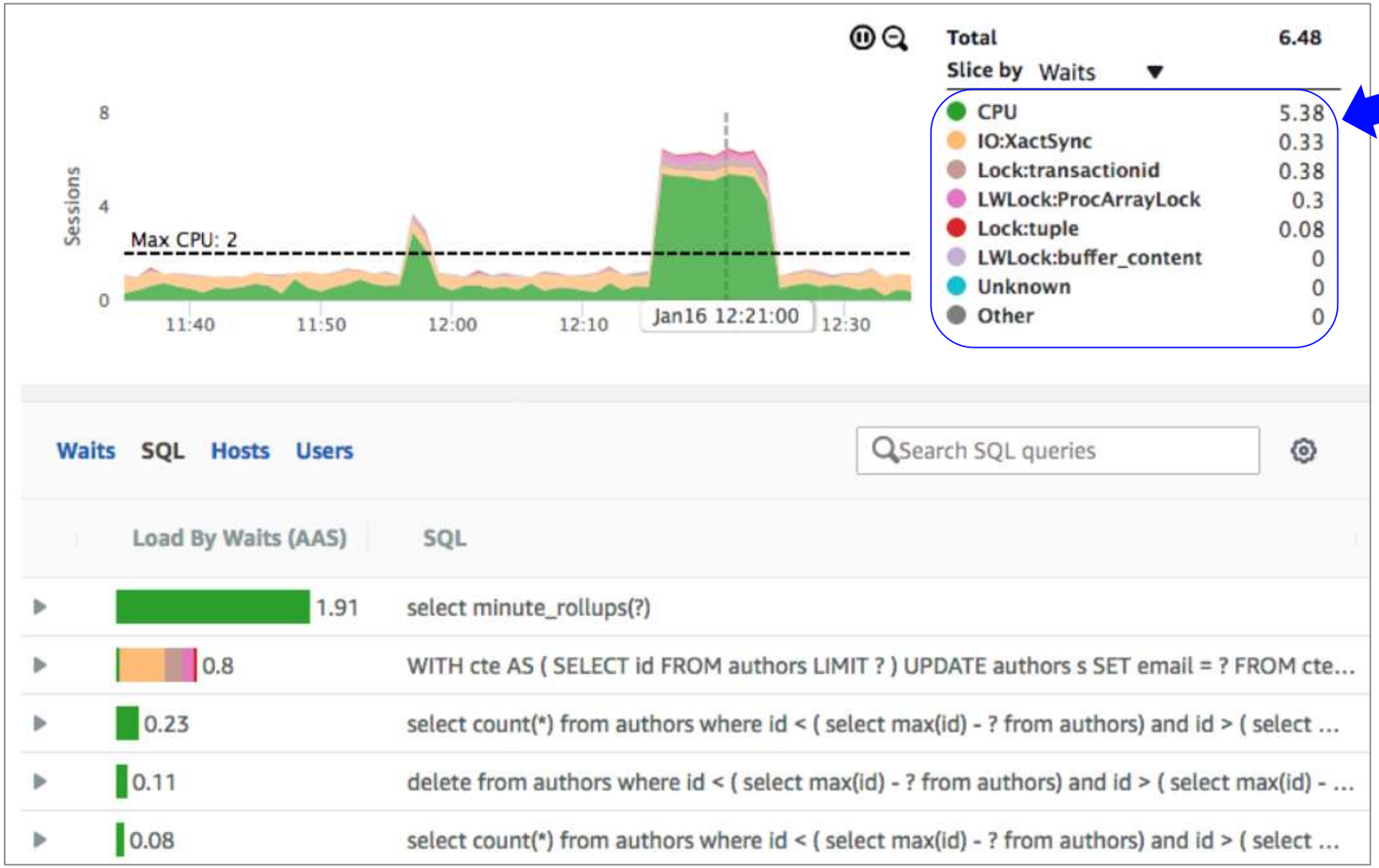


CPU bottleneck



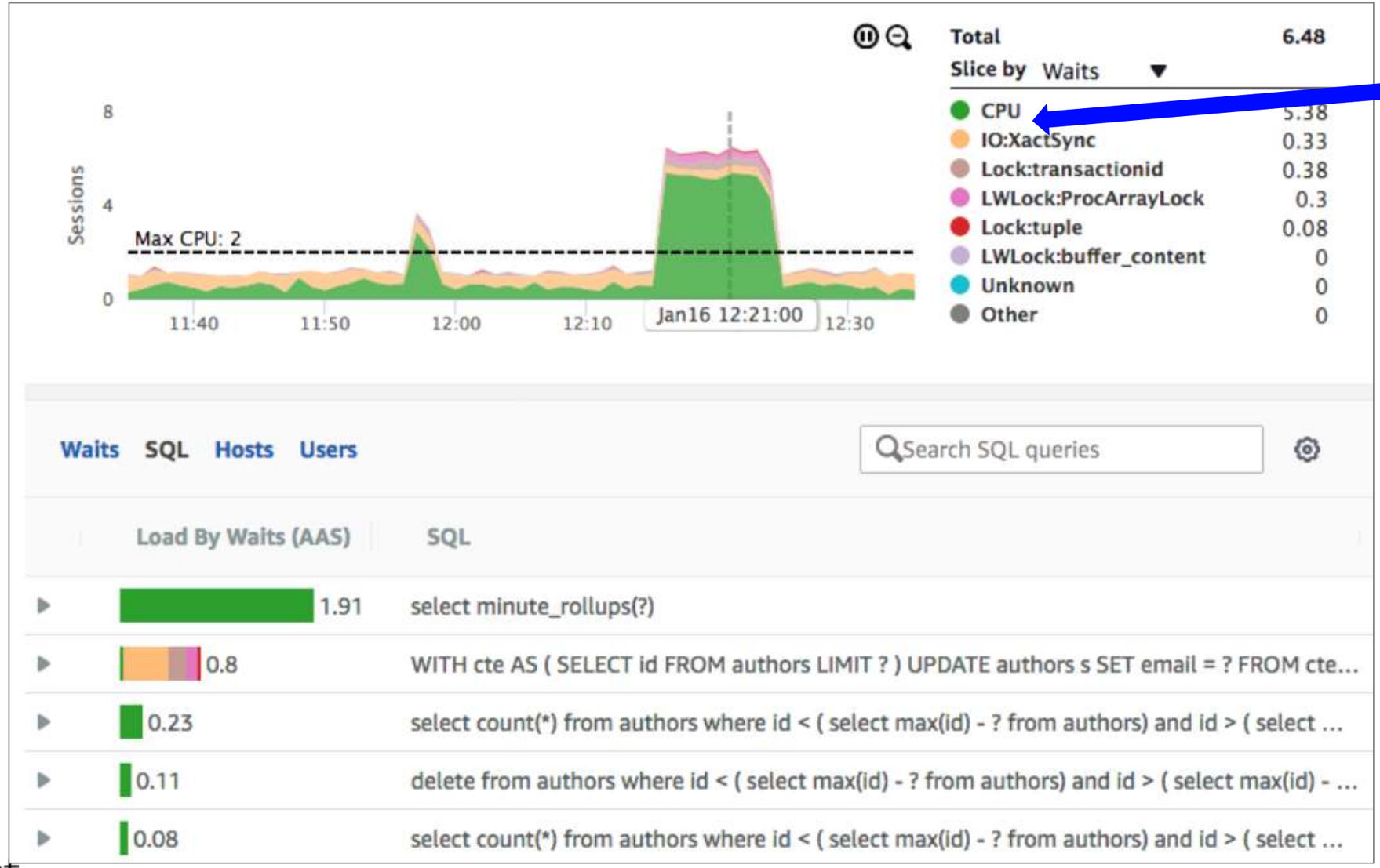
Bottleneck

Customer use case: CPU bottleneck



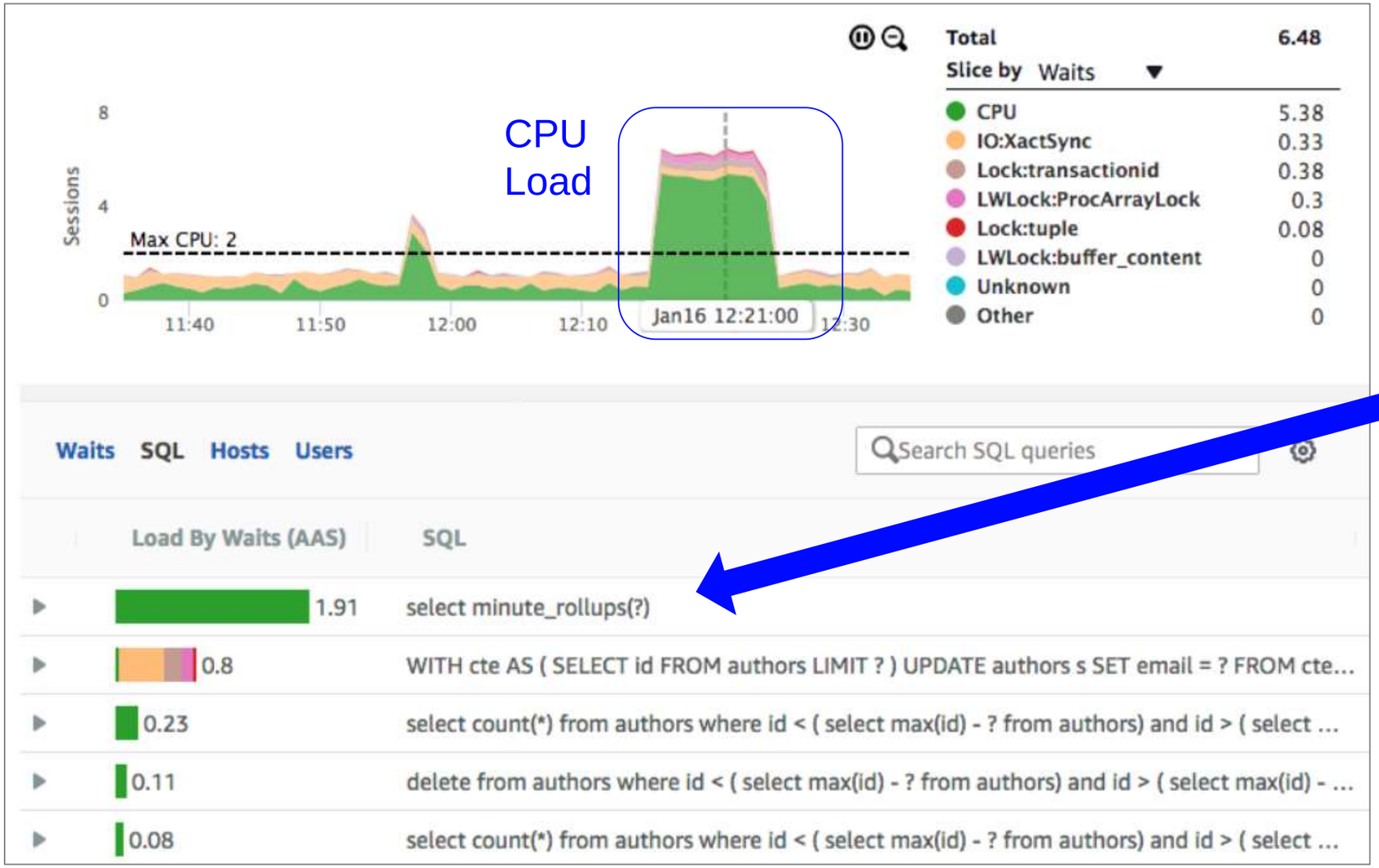
Wait States

CPU bottleneck



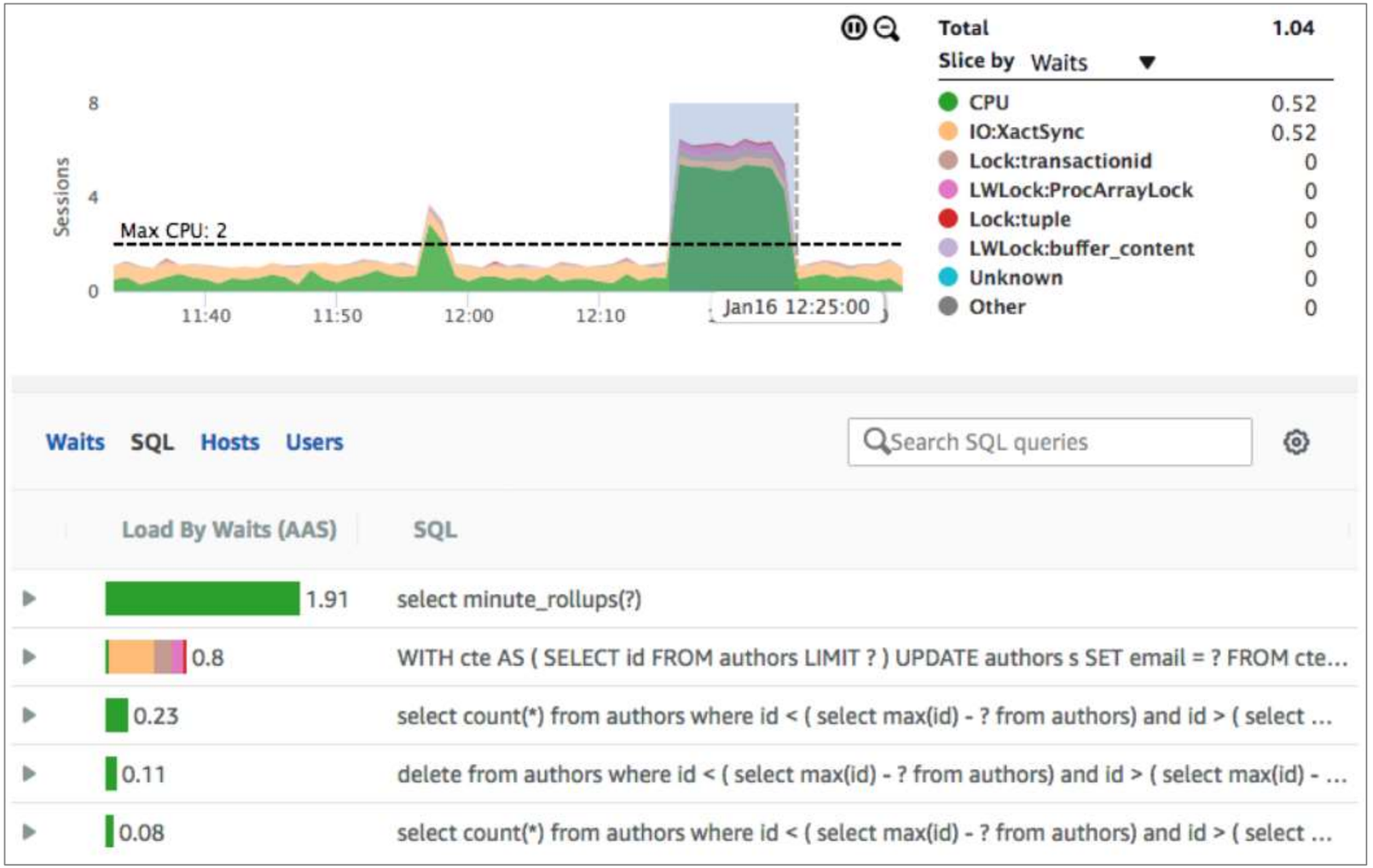
CPU

CPU bottleneck

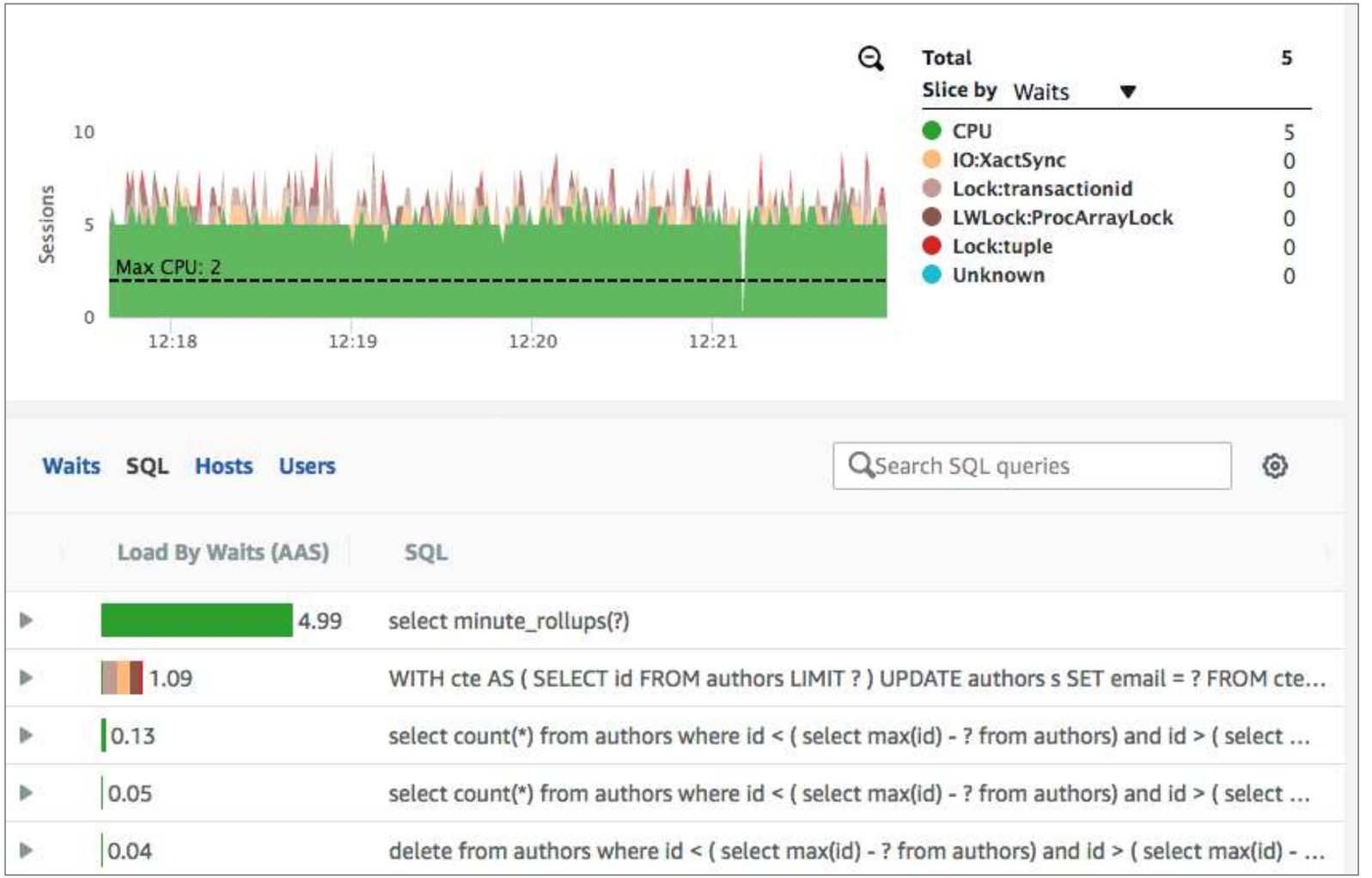


SQL
with high CPU

Click and drag

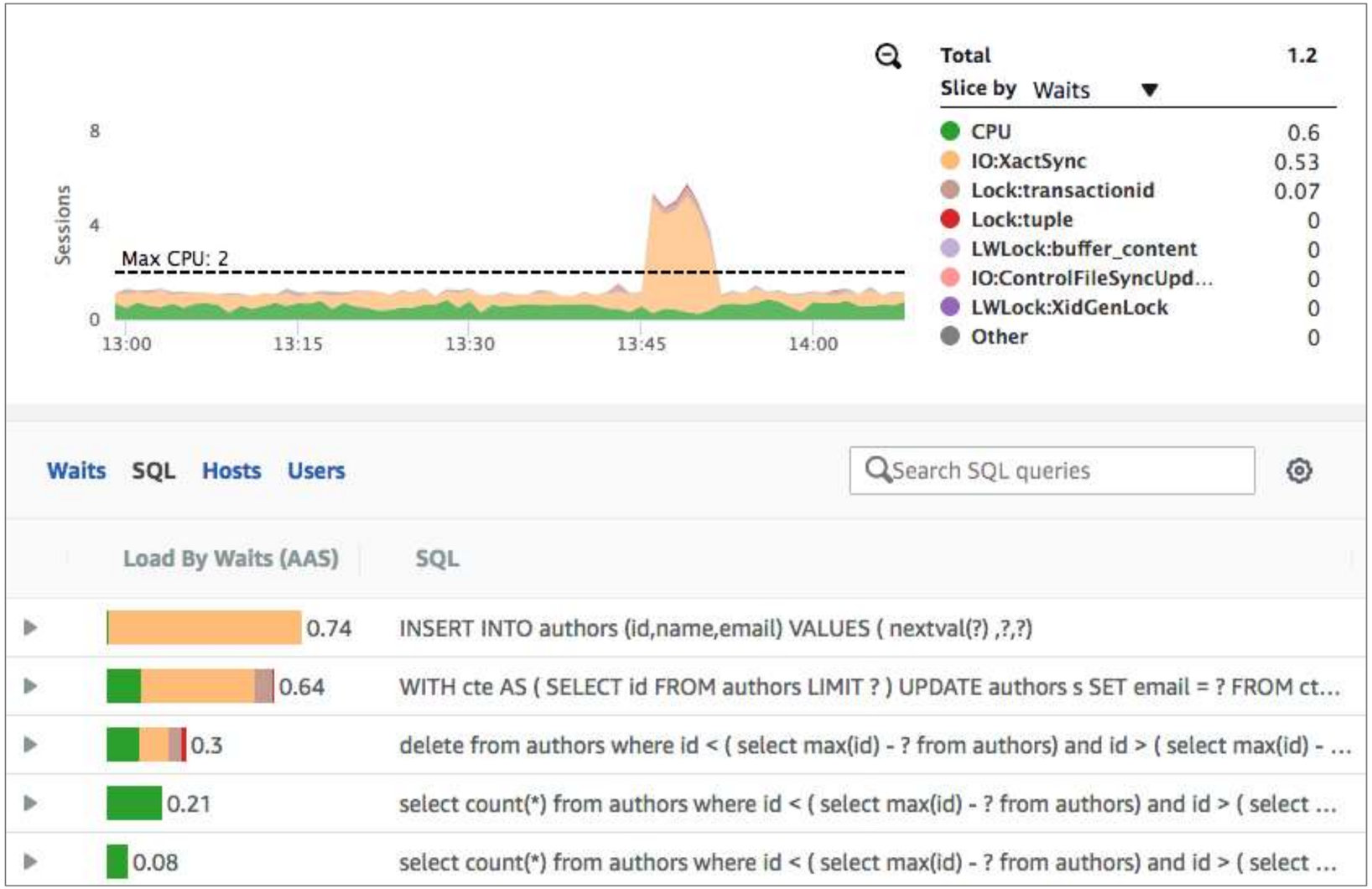


Zoom in

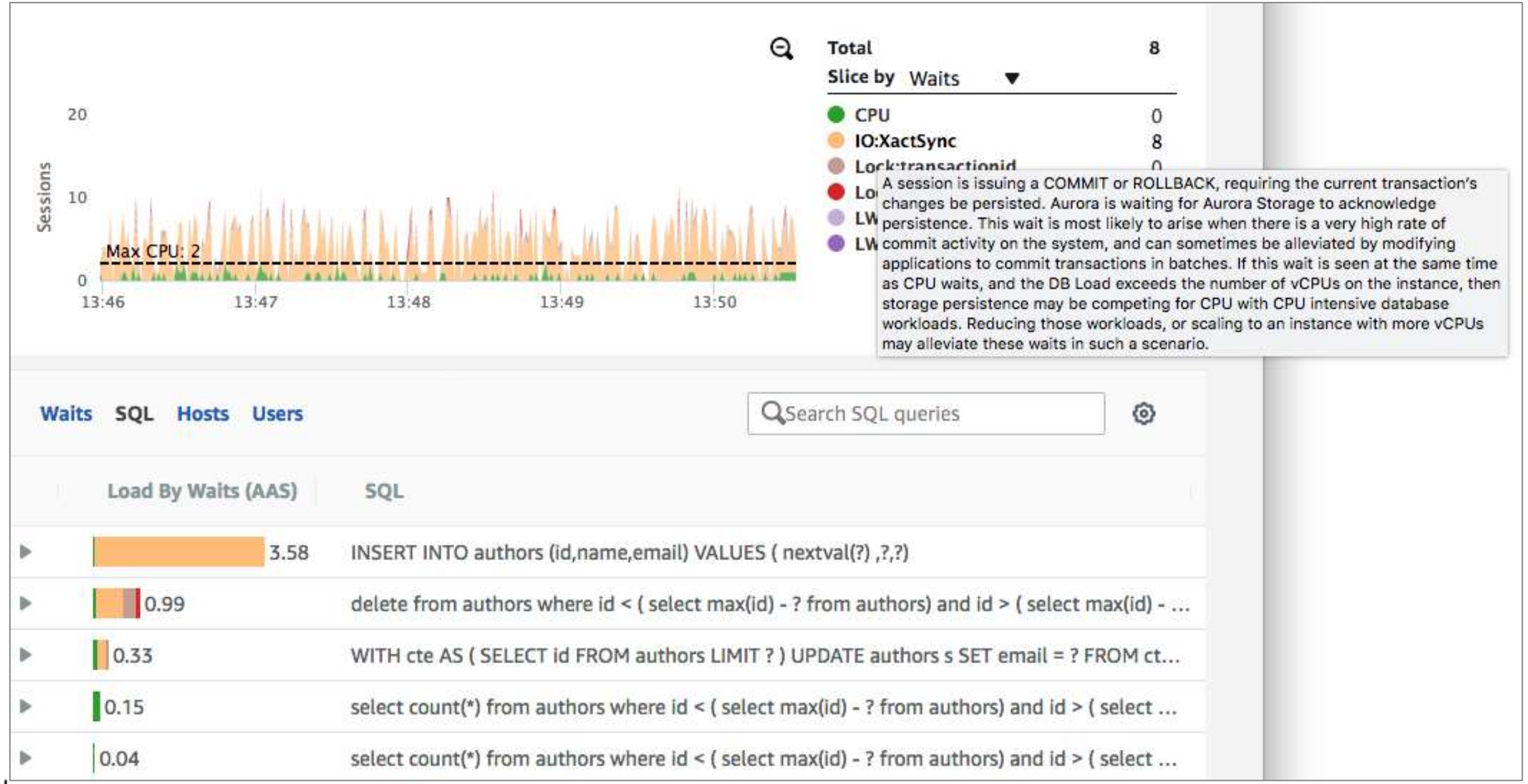


Customer use case: Wait bottleneck

Wait bottleneck

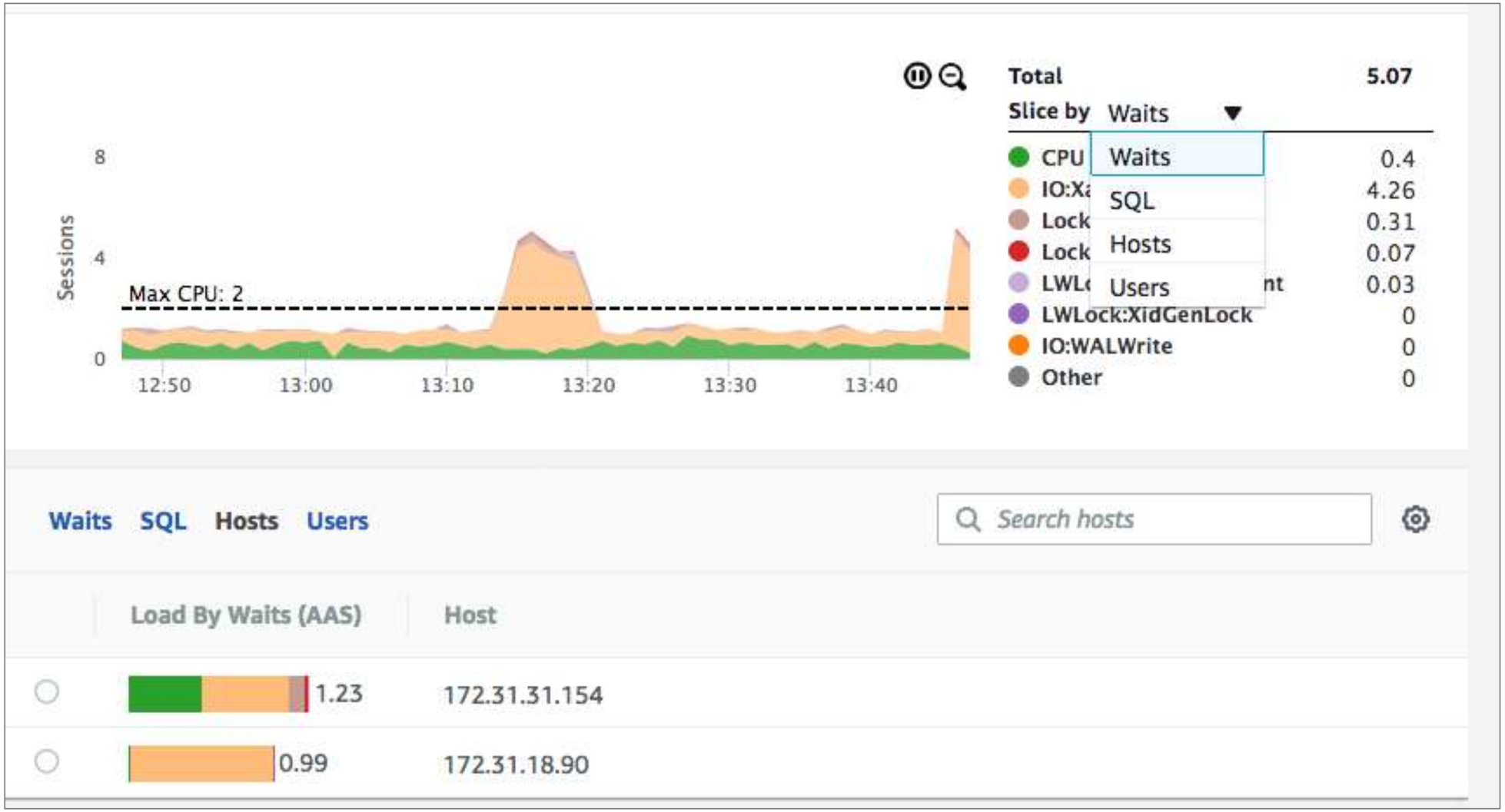


Wait bottleneck

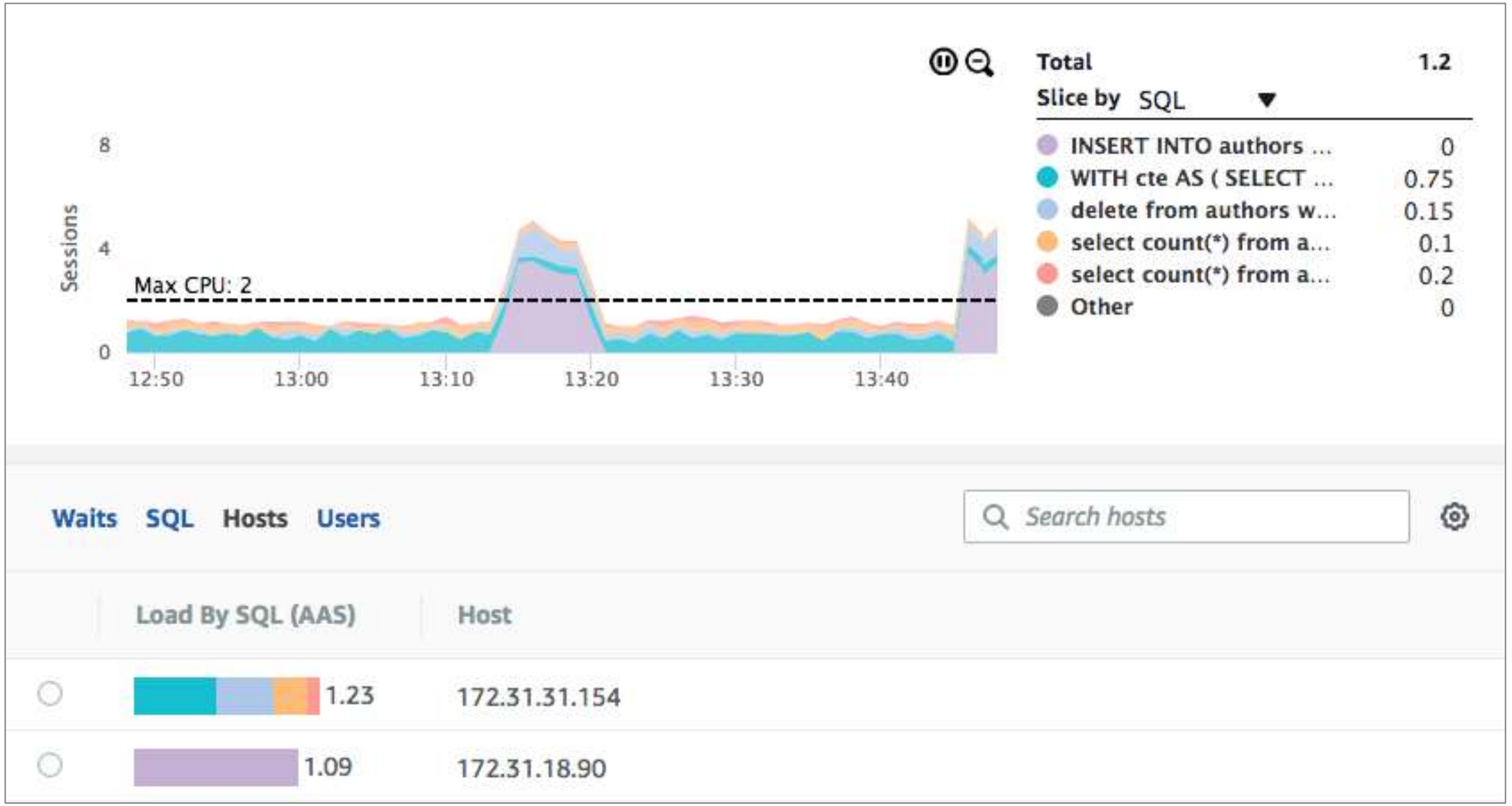


Dashboard: Other grouping dimensions

Other dimensions



Top host by SQL statement

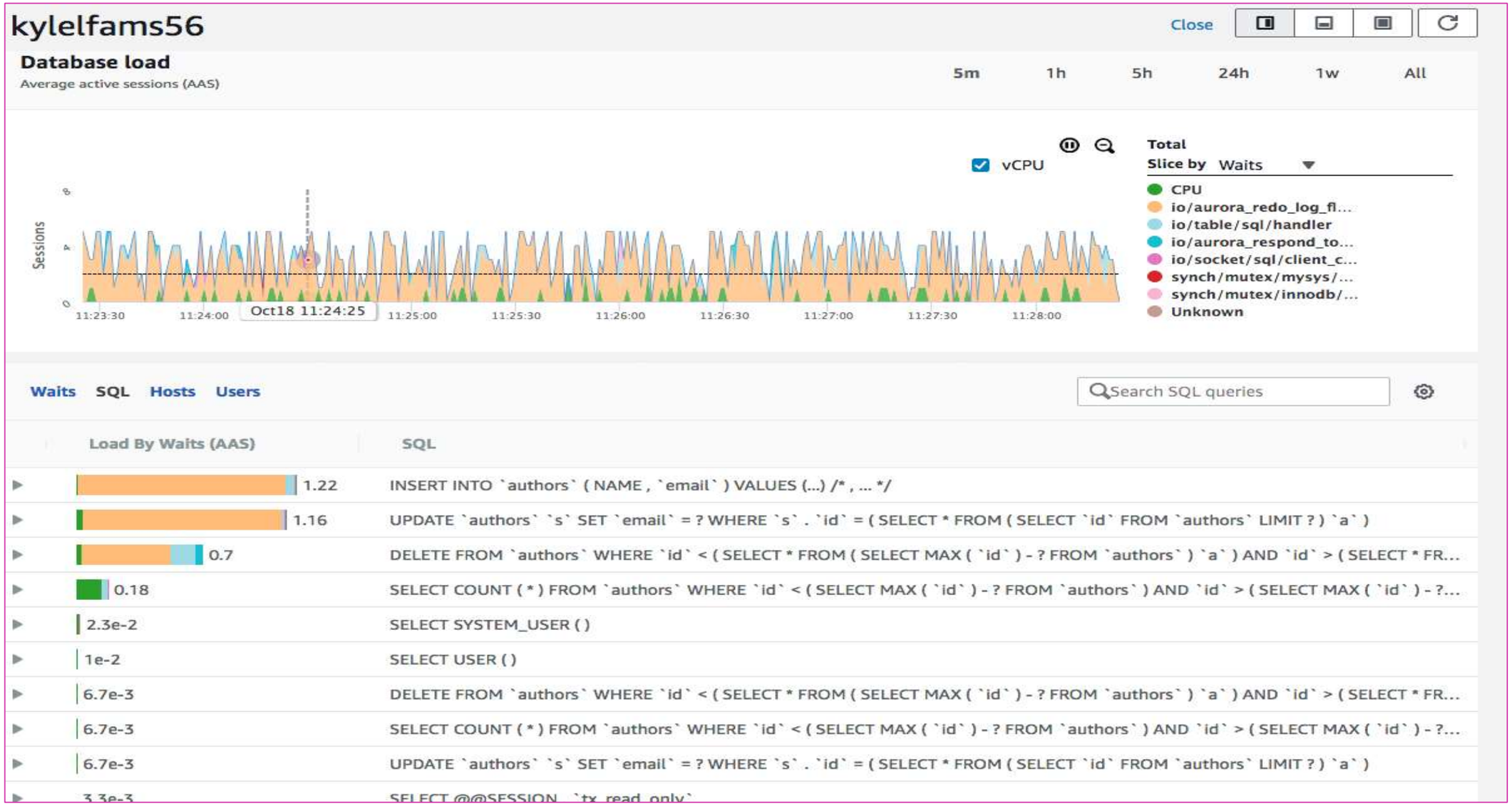


Performance Insights across engines

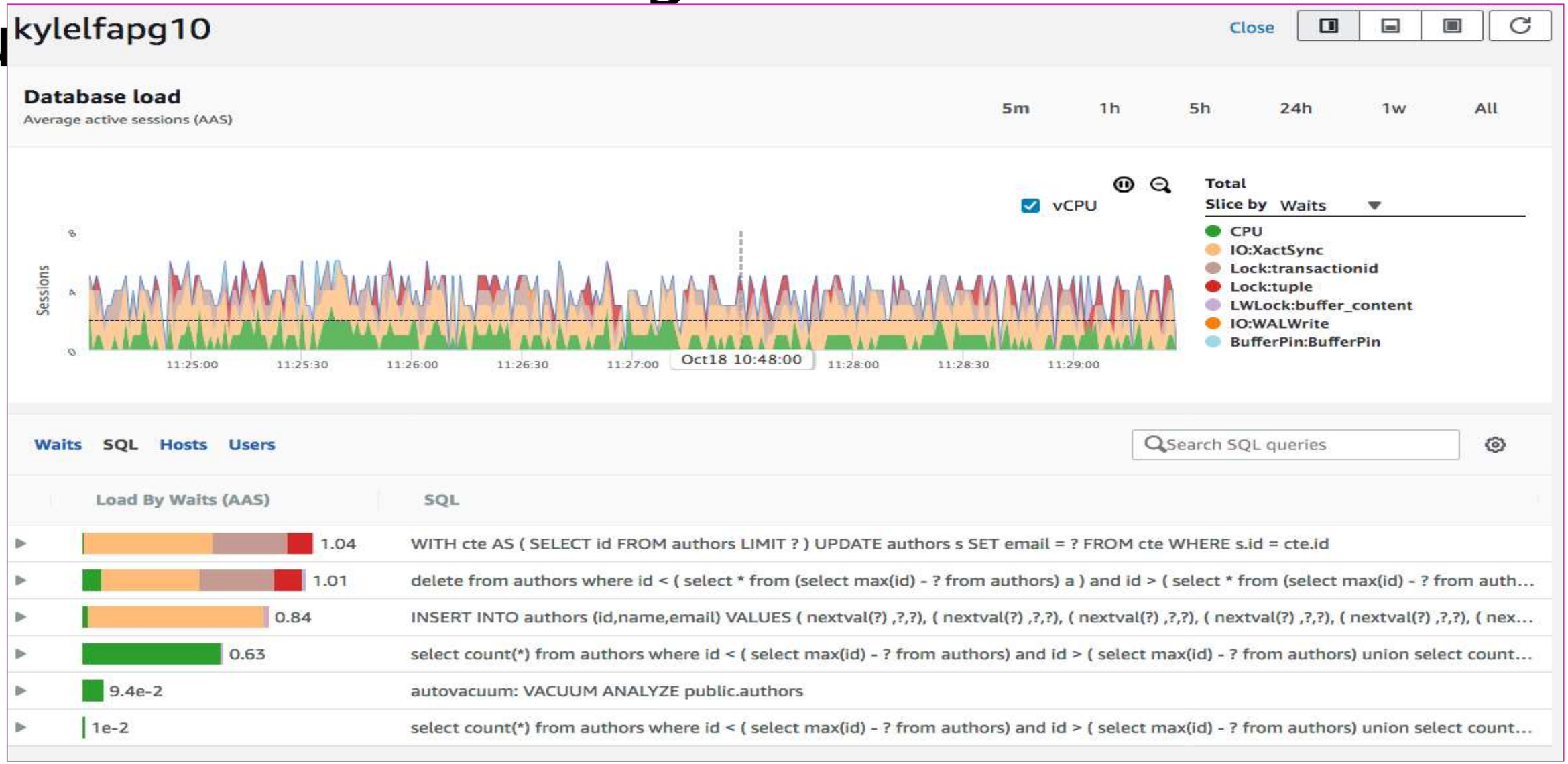
Performance Insights across DB engines

- Performance Insights supports
  - Amazon Aurora
    - MySQL
    - Postgres
  - Amazon RDS
    - MySQL
    - Postgres
    - Oracle
  - RDS MariaDB and Oracle forthcoming
- Interface is the same across different engines
  - Allows DBA to do performance work across different engines easily
  - Dashboard content same
    - Only difference is the wait event names, which are engine dependent

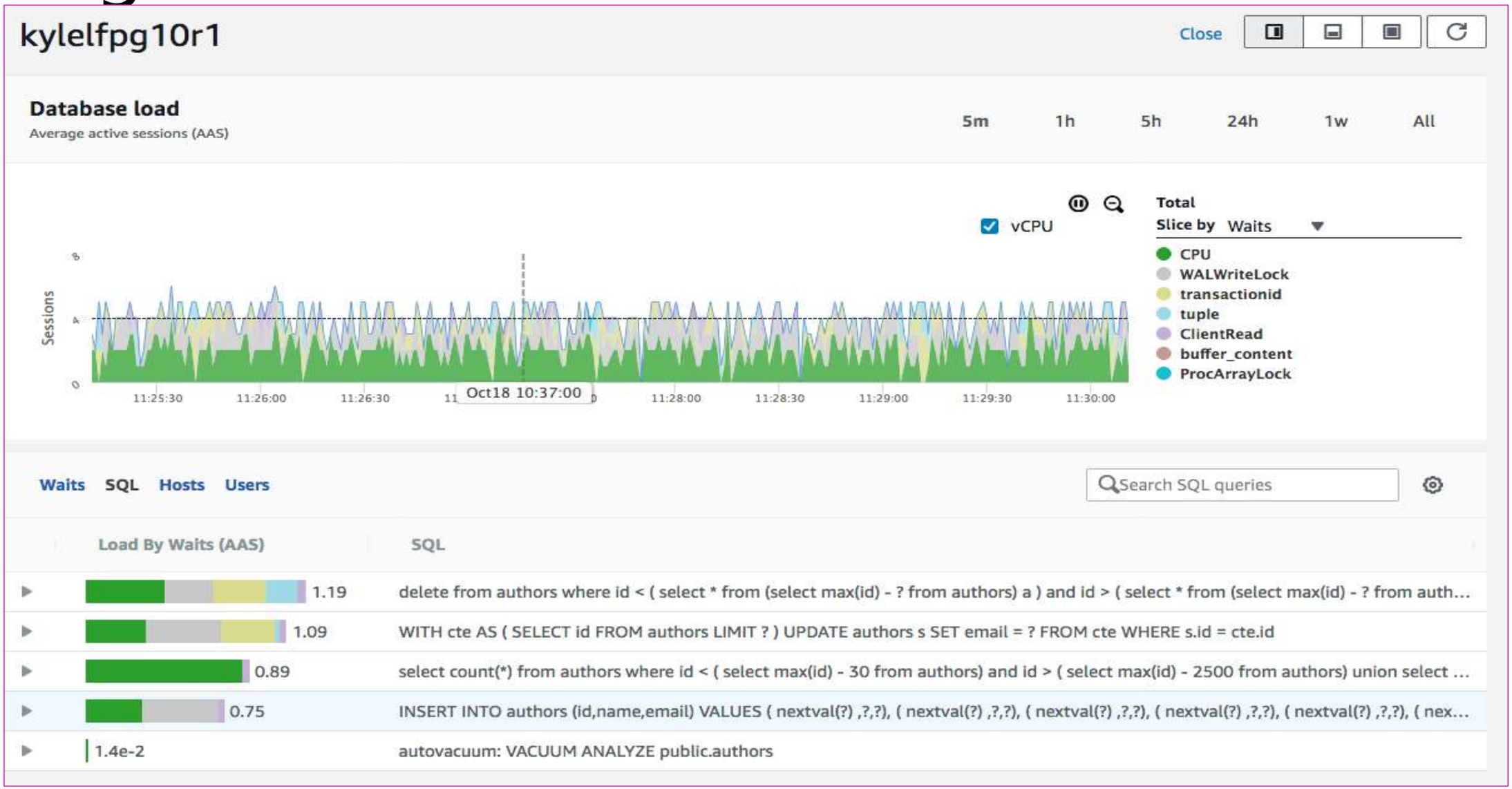
Amazon Aurora MySQL—Five users



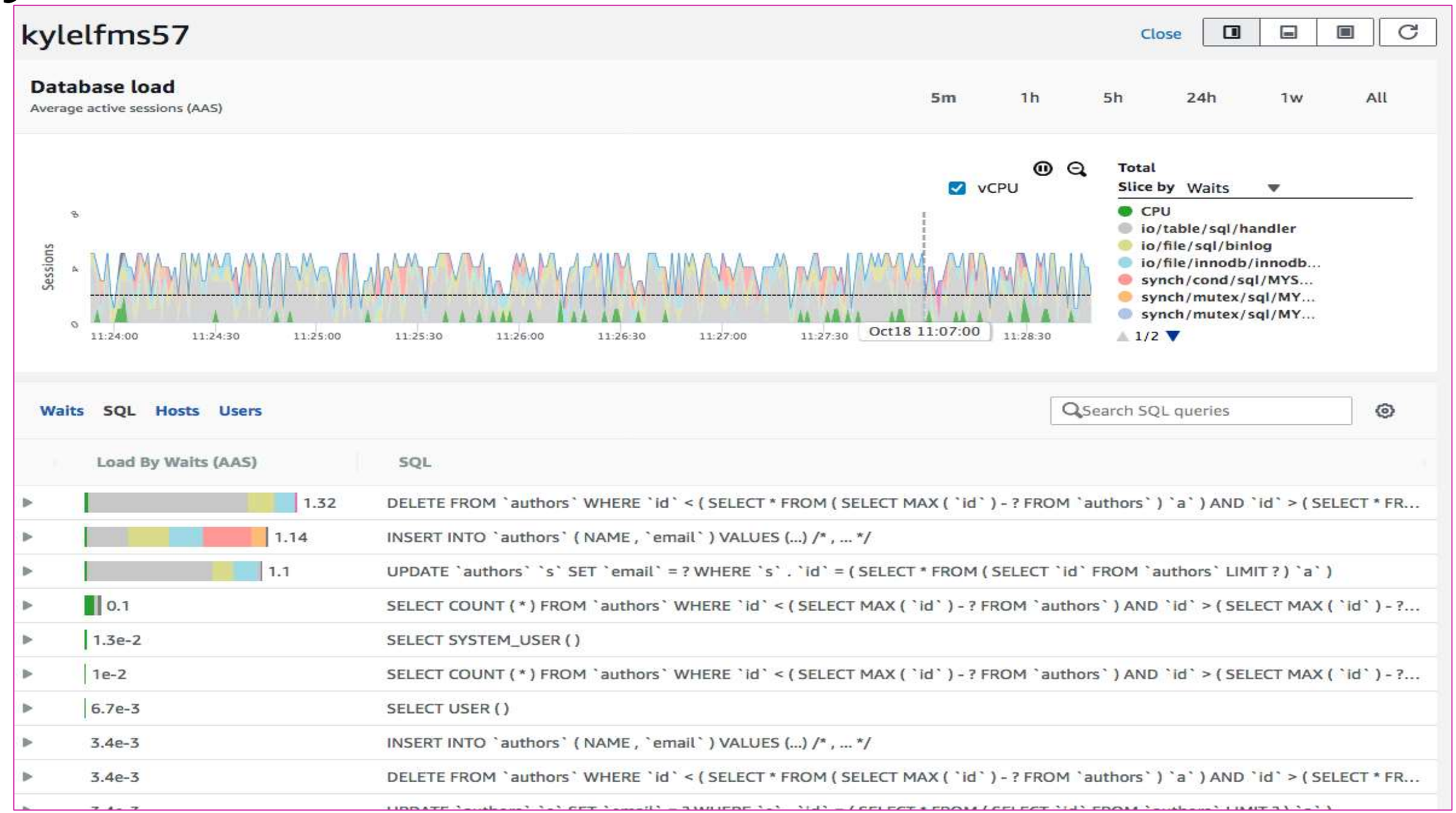
Amazon Aurora PostgreSQL—Five u



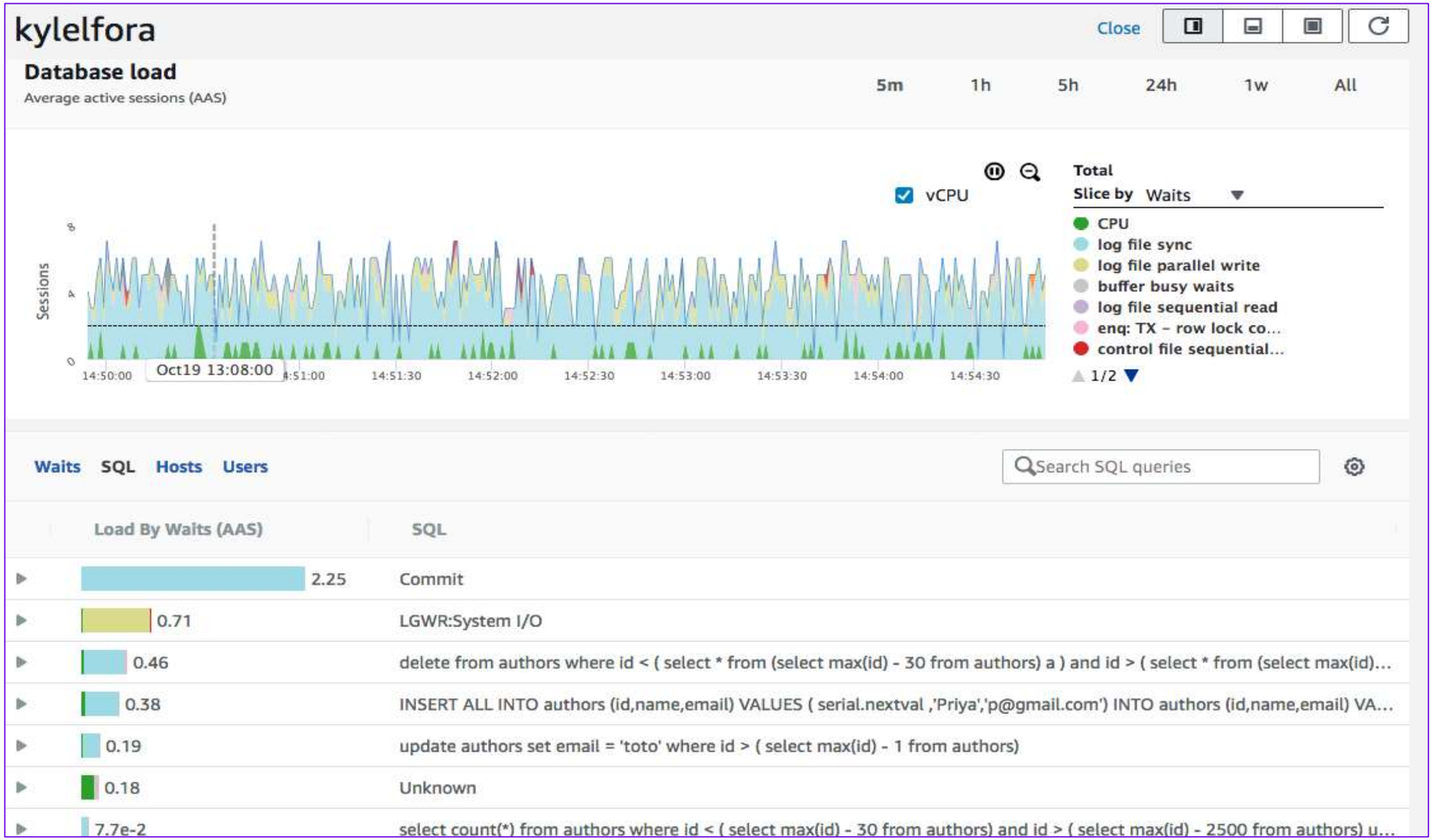
Postgres—Five users



MySQL—Five users



Oracle—Five users



What’s available

What is available

- Available
  - Engines
    - Amazon Aurora PostgreSQL
    - Amazon Aurora MySQL 5.6 1.17.3 and higher
    - Amazon RDS Postgres 10
    - Amazon RDS MySQL 5.6.41+ and 5.7.22+
    - Amazon RDS Oracle
  - Functionality
    - DB load chart
    - Top N table
  - Wait, user, host, SQL
  - API/SDK
  - Long-term data retention
  - Alerts through Amazon CloudWatch

What is Coming

- Available
  - Engines
    - Amazon Aurora PostgreSQL
    - Amazon Aurora MySQL 5.6 1.17.3 and higher
    - Amazon RDS Postgres 10
    - Amazon RDS MySQL 5.6.41+ and 5.7.22+
    - Amazon RDS Oracle
  - Functionality
    - DB load chart
    - Top N table
  - Wait, user, host, SQL
  - API/SDK
  - Long-term data retention
  - Alerts through Amazon CloudWatch
- Coming
  - Engines
    - Amazon Aurora MySQL 5.7
    - Amazon RDS for MariaDB
    - Amazon RDS SQL server
  - Functionality
    - SQL execution plan
    - SQL stats
    - OS and DB statistics