One Ring to rule them all, One Ring to find them,
One Ring to bring them all, and in the darkness bind them …

How is database performance doing today?

How is database performance doing today?

- Cache buffer chains
- Latch contention
- …
- It's going really good … :-)
- How to answer with a single number?

The Ultimate Answer to Life, The Universe, and Everything.

(Douglas Adams)

How is database performance doing today?

- Cache buffer chains
- Latch contention
- …
- It’s going really good … :-)
- How to answer with a single number?
- The Ultimate Answer to Life, The Universe, and Everything?
- Possible?
- Meaningful?

Make IT
2022

Boris Oblak

One indicator to rule them all

Abakus Plus d.o.o.

- History
  - From 1992
  - ~20 employees
- DBA Applications
  - DejaVu
  - ARBITER
  - APPM
- Enterprise Applications
  - Document Management
  - Newspaper Distribution
  - Flight Information System

- Services
  - OS & Network admin
  - DBA, Programming
- Infrastructure
  - > 20 years of experience with

    High Availability on GNU/Linux
- Hardware
  - Servers, SAN, ceph,
  - Firewalls,
  - Backup Server

AS Abakus As na disku.

dejavu Data at your service.

Abakus and Oracle

- Oracle database on linux
  - Abakus: 1995
    (Oracle 7.1.5, Forms 3.0)
  - Oracle: 1997
- Parallel execution
  - Abakus: 2004
    (SIOUG 2004: Vzporedno Izvajanje operacij s PL/SQL – Boris Oblak)
  - Oracle: 2007
    dbms_parallel_execute

APPM

Abakus Plus
Performance
Monitor

- For Oracle Database Standard Edition
- Made by DBAs for DBAs
- Temporal performance comparison
- Resource allocation optimization
- Database performance tracking
- Performance bottleneck optimization

Abakus

Backup server

supports Oracle Databases and OLVM VMs

- **Backup**
  takes no time
- **Recovery**
  data recovery is almost instant
- **Disk space**
  backed up data takes up minimal amount of disk space
- **Availibility**
  data is always available and always in view
- **Security**
  backed up data can not be deleted without support personnel intervention
- **Alternative uses**
  BI analysis / reporting / DB upgrade verification / R&D testing / seamless business continuation

www.abakus.si

References

Gorenjska Banka

GENERALI Zavarovalnica
GENERALI GROUP

Ljubljana Airport

EKDIS
Ekspresno. Ekonomično.

REPUBLIKA SLOVENIJA
MINISTRSTVO ZA OBRAMBO

NOVA BANKA

MILENIJUM OSIGURANJE

KONTROLA ZRAČNEGA PROMETA SLOVENIJE

Iskra®

Hidria

Mestna občina Ljubljana

skbbanka
otp group

triglav

ANDRITZ

jata emona
LJUBLJANA

UNIVERZITETNA PSIHIATRIČNA KLINIKA LJUBLJANA
University Psychiatric Clinic Ljubljana

BANKA SLOVENIJE

SAVARe

MERKUR

TRELLEBORG
Trelleborg Slovenija, d.o.o.

SODO
SISTEMSKI OPERATER DISTRIBUCIJSKEGA OMREŽJA Z ELEKTRIČNO ENERGIJO

NLB Vita
Življenjska zavarovalnica

PRVA

MAGNETIK d.o.o.

ISS PEST MANAGEMENT SOLUTIONS

Mercator

MM KARTON

studio item

Blubit

TIKO
TOVARNA KOVINSKE OPREME

ZAVOD ZA ŠPORT RS PLANICA

PH Primorska hranilnica

GOODYEAR DUNLOP
SAVA TIRES

SAVA HOTELS & RESORTS

hit alpinea
Kranjska Gora

CS CENTROSINERGIJA

PANTEON® GROUP

Lonia

PRONET
CHOOSE THE FUTURE

LASERLINE

ORACLE®

ROS d.o.o.

INFOTRANS®

PARK POSTOJNSKA JAMA

ADRIA ANKARAN
HOTEL & RESORT

Database performance

- sql_id: elapsed time,
- job: elapsed time,
- entire instance?
  - without measuring wall time and elapsed time?

- How is DB behaving today?
- By how much will new HW speed up a DB?
- What kind of HW will make DB run twice as fast?
- Change HW or hire a DBA?

At Any Point in Time

- You are doing:

Something

At Any Point in Time

- You are doing:
    - something:
        - – shopping, exercising, feeding your pets, preparing a meal, driving to a destination, working, talking, studying, … you are doing something.

Nothing

At Any Point in Time

- You are doing:
  - something:
    - shopping, exercising, feeding your pets, preparing a meal, driving to a destination, working, talking, studying, … you are doing something.
  - nothing:
    - waiting on something:
      - sleeping, waiting for the pizza delivery, waiting in the market, waiting for coffe to brew, waiting for inspiration to write a code, waiting in line at the post office, …

At Any Point in Time

- Working.
- Waiting.
  - Does Not Necessarily Correlate With Inefficiency.
  - Your life is going to naturally have some wait time build into it (and that’s ok).

Oracle Server Process

- Something (Executing Code – Burning CPU)
- Nothing (Waiting – NOT Burning CPU)

Oracle Server Process

Executing Code

ON CPU

Oracle Server Process

Waiting

Wait Event

Waiting to read block into the buffer cache

Waiting on DBRW to write dirty blocks

Waiting on a row – lock

…

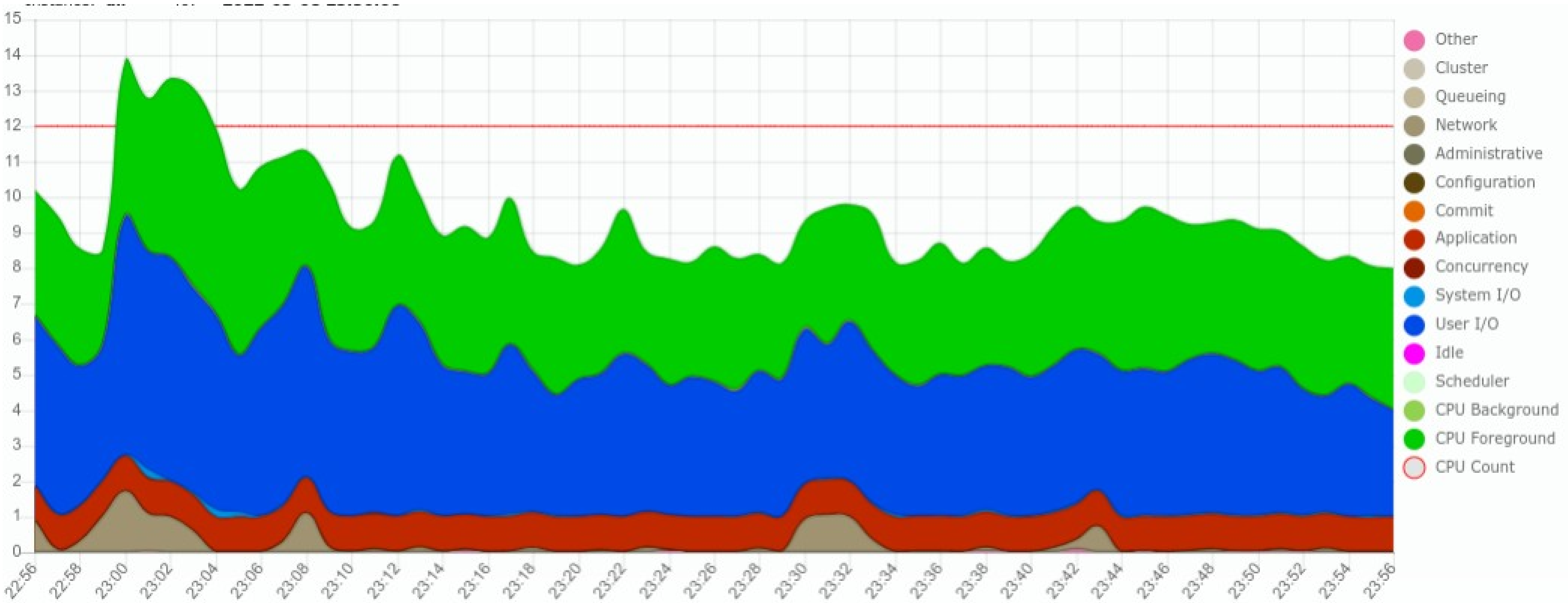
Oracle Server Process

- ON CPU
- Waiting
  - Waiting to read block into the buffer cache
  - Waiting on DBRW to write dirty blocks
  - Waiting on a row – lock
  - …
- Idle
  - Waiting for some work to be assigned
    - SQL*Net message from client?
    - …

Most Healthy Queries

- Spends Some Time Waiting and
- Some time on CPU.
- Having some wait time is not bad.
- Your »regular life« has some wait time too.

Most Healthy Queries



AS
Abakus
As na disku.

dejavu Data at your service.

User Experience

Response time

Response Time #0

- Unit of work (LIO)
- Time = »Working Time« + »Wait Time«
- Response time = »Time« / »Units« (Time per one Unit - ms/LIO)

https://method-r.com/wp-content/uploads/2017/07/Why-You-Should-Focus-on-LIOs-Instead-of-PIOs.pdf
https://blog.pythian.com/do-you-know-if-your-database-slow/
https://blog.orapub.com/20181204/do-direct-path-reads-count-as-logical-reads.html

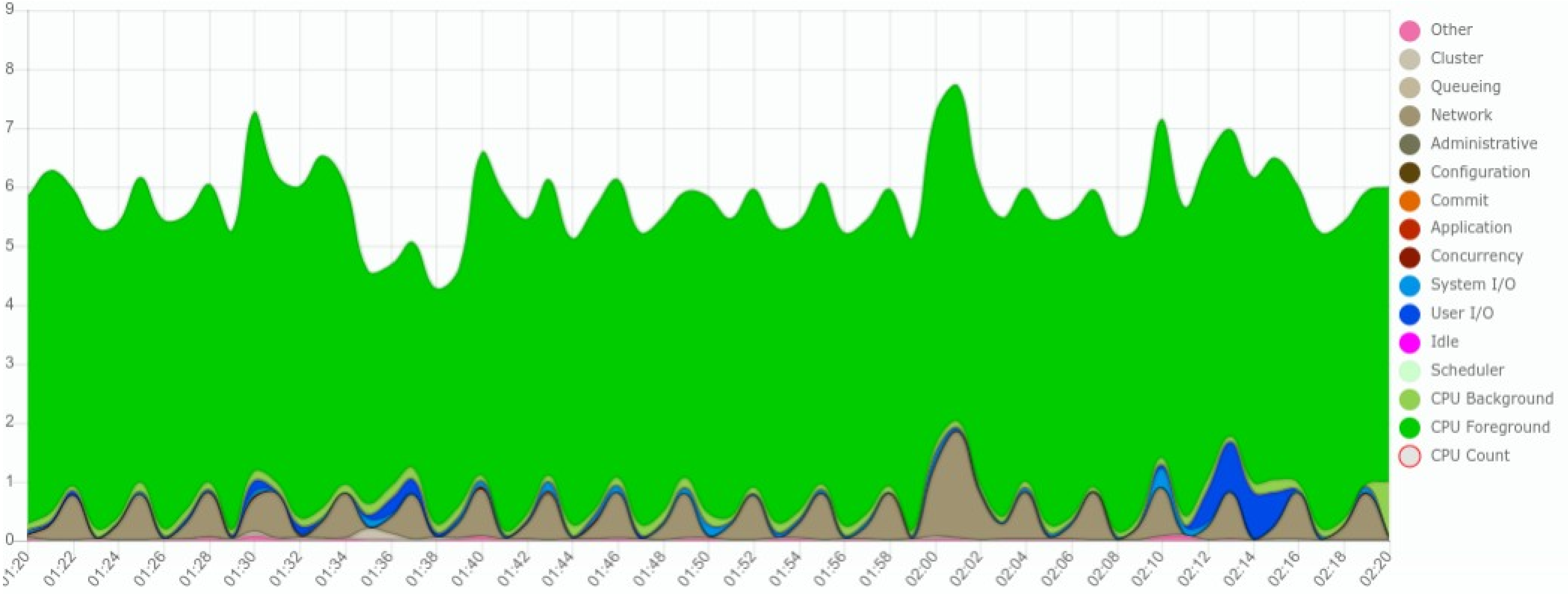
Response Time #0

- Time to complete operation
  - »Units to be done« * »Response Time«

Response time #1

- In a perfect world – most ON CPU: min Rt

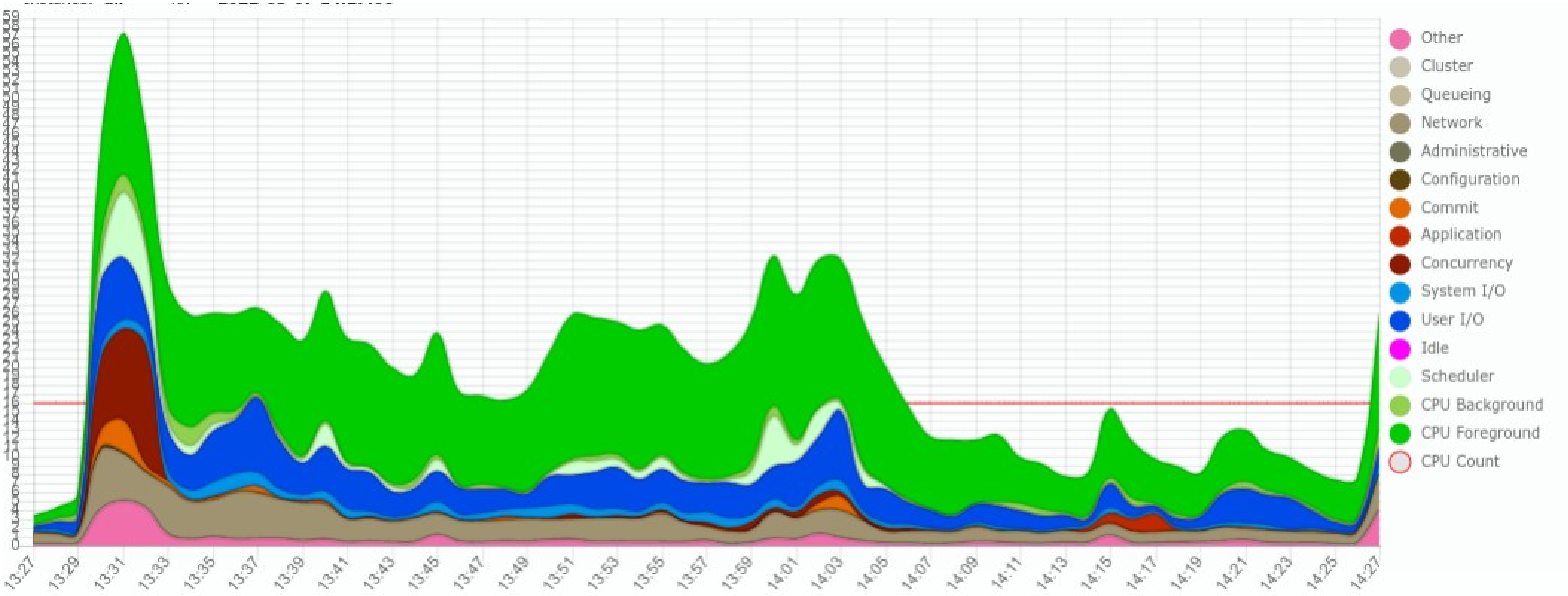
Ideal Response time



AS Abakus As na disku.

dejavu *Data at your service.*

Real Response time



As na disku.

dejavu Data at your service.

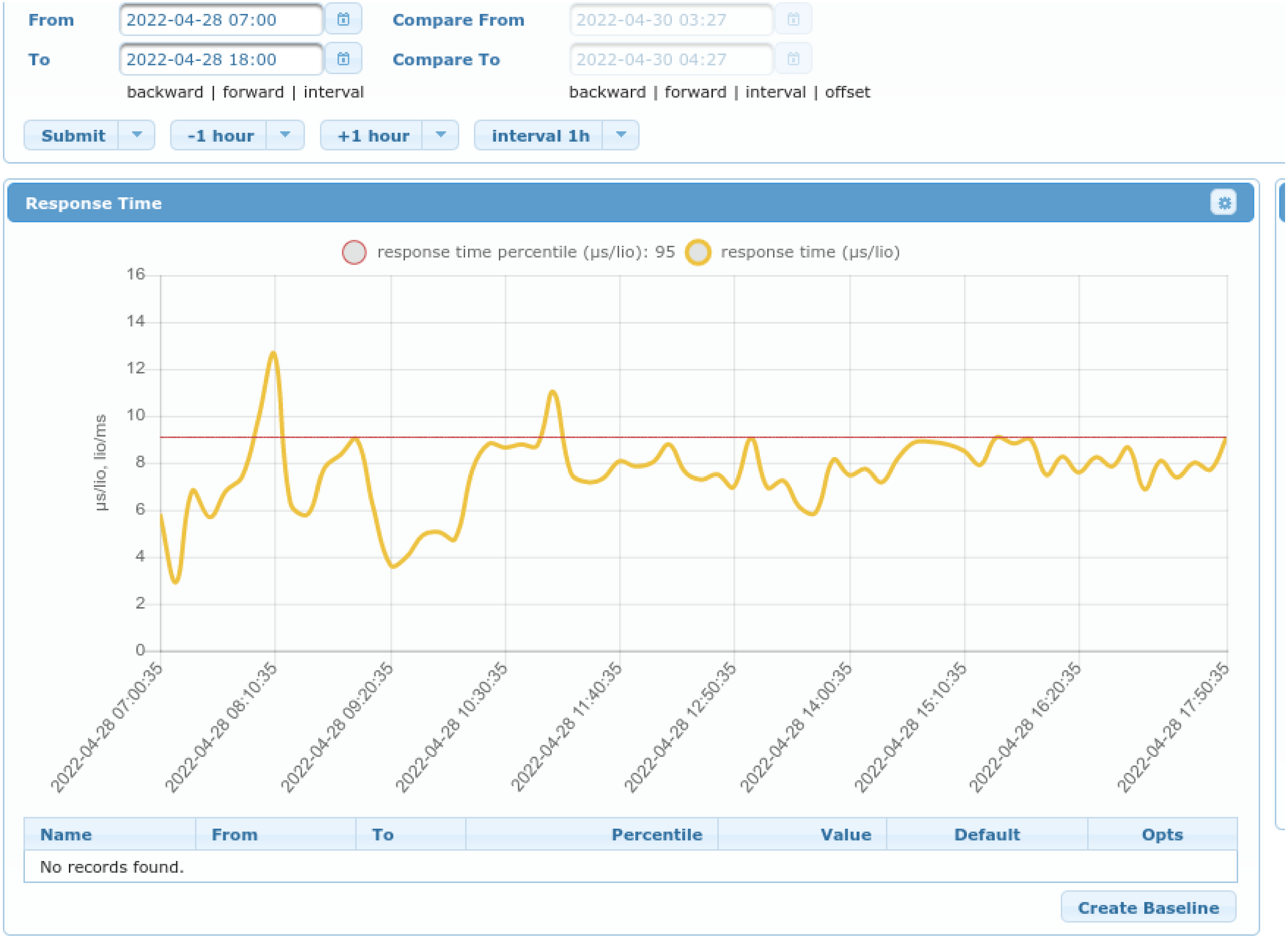
Response time #1

- In a perfect world – most ON CPU: min Rt,
- Average – unreal, in more than 50% SQL will run longer,
- Real:
    - snapshot of an »acceptable« case,
    - baseline (e.g. Rt covering 95% of all cases),
    - standard deviation.
- Calculate response time baseline when database performs »acceptable«.
    - Carve it in stone.

Baseline

Carve Response Time in stone.

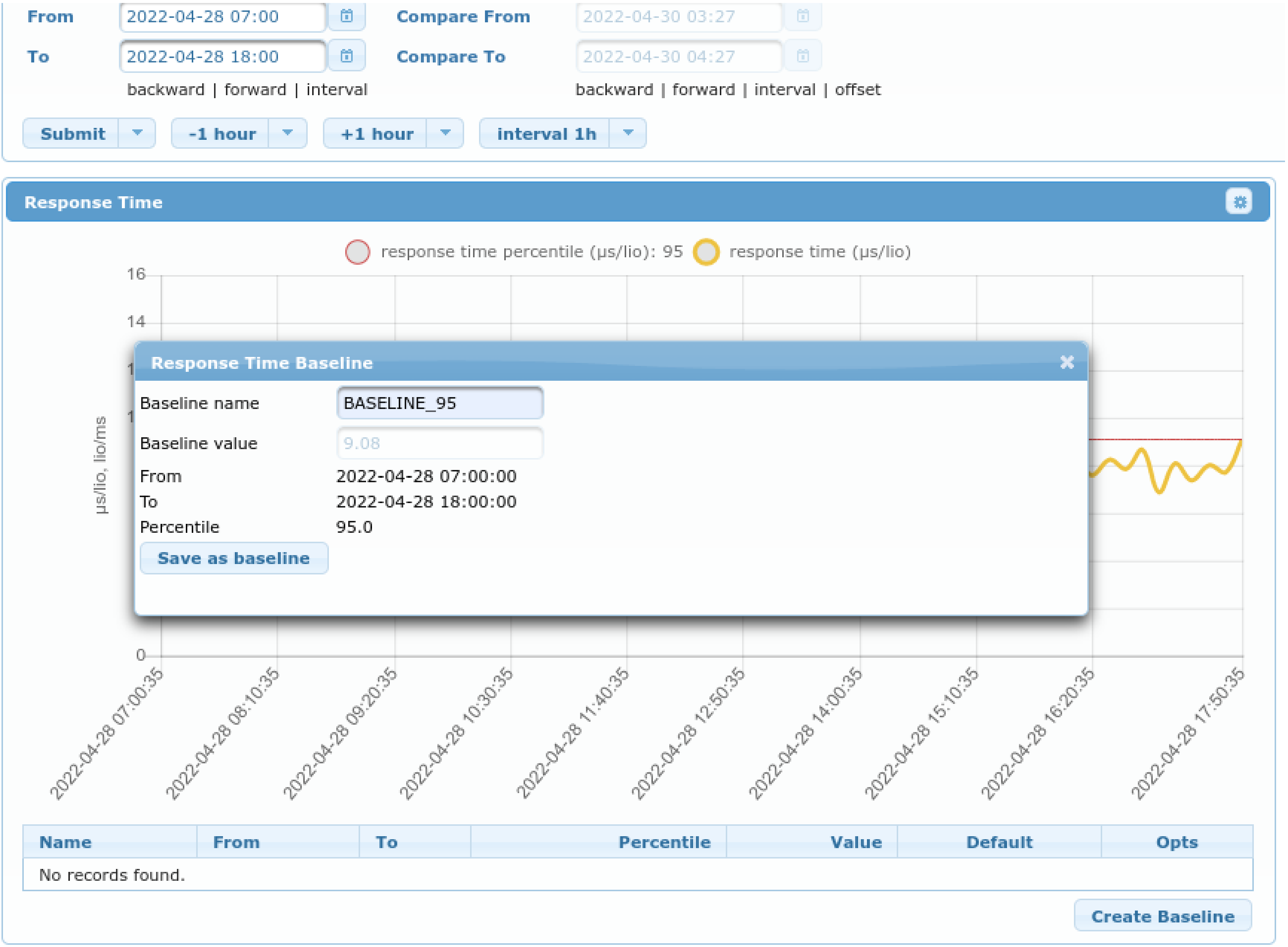
Baseline



From	2022-04-28 07:00	Compare From	2022-04-30 03:27
To	2022-04-28 18:00	Compare To	2022-04-30 04:27
	backward | forward | interval		backward | forward | interval | offset

Name	From	To	Percentile	Value	Default	Opts
No records found.						

Baseline



From 2022-04-28 07:00

To 2022-04-28 18:00

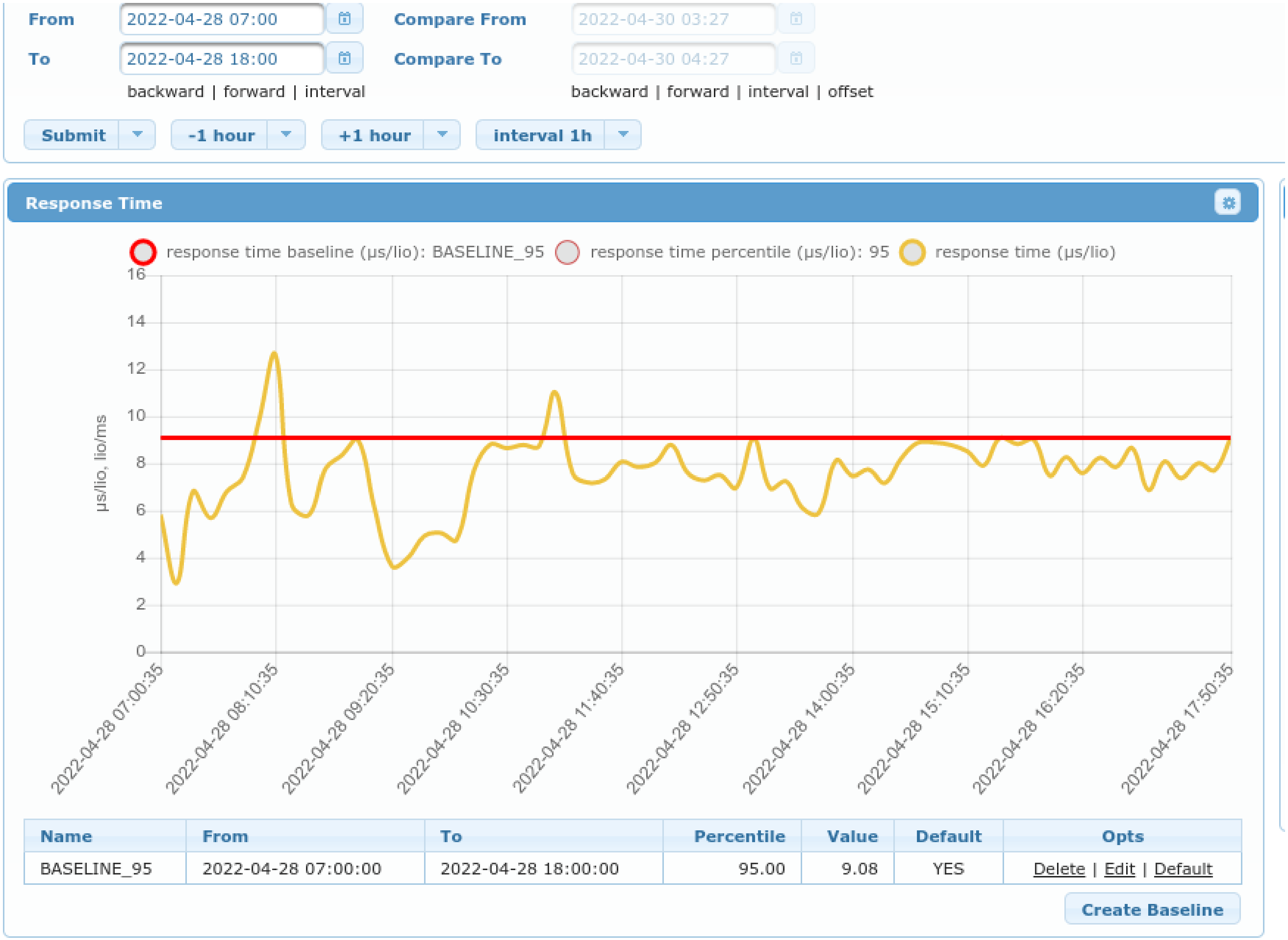
Compare From 2022-04-30 03:27

Compare To 2022-04-30 04:27

Baseline name	BASELINE_95
Baseline value	9.08
From	2022-04-28 07:00:00
To	2022-04-28 18:00:00
Percentile	95.0

Name	From	To	Percentile	Value	Default	Opts
No records found.						

Baseline



From	2022-04-28 07:00	Compare From	2022-04-30 03:27
To	2022-04-28 18:00	Compare To	2022-04-30 04:27
	backward | forward | interval		backward | forward | interval | offset

Name	From	To	Percentile	Value	Default	Opts
BASELINE_95	2022-04-28 07:00:00	2022-04-28 18:00:00	95.00	9.08	YES	Delete | Edit | Default

Work and Time

- Oracle Work:
  - ON CPU.
  - Wait.

Work and Time

- Oracle Work:
  - ON CPU.
  - Wait.

DB time

Unit of Work: LIO (logical IO)

- LIO processing is the number-one bottleneck for many busieness processes.
- LIO consumes two of system’s most expensive resources: CPU and latches.

https://method-r.com/wp-content/uploads/2017/07/Why-You-Should-Focus-on-LIOs-Instead-of-PIOs.pdf
https://blog.pythian.com/do-you-know-if-your-database-slow/
https://blog.orapub.com/20181204/do-direct-path-reads-count-as-logical-reads.html

Work and Time

- Oracle Work:
  - 9 min ON CPU.
  - 1 min Wait.
- time = DB time = 10 min.
- work = 3.000.000 LIO
- (10 * 60 * 1000) / 3.000.000 = 0,02 ms/LIO.
- Time to process single LIO = 0,02 ms!
- This will be our indicator.

How can we use it

- When number of LIO increases, DB time increases (more work = more LIO & more DB time).
- Relationship between LIO and DB time is linear.
- Indicator (ms/LIO) remains more or less the same.
- Until system get’s too busy!
- If indicator increases … may have a problem!

Get the data

- AWR (EE), statspack, Abakus APPM, …
  - DB time
  - statistic: »session logical reads«
- Running system:
  - DB time: `SELECT value FROM v$sys_time_model WHERE stat_name = 'DB time';`
  - LIOs: `SELECT value FROM v$sysstat WHERE name = 'session logical reads';`

Tests

- Take sample.
- Run load.
- Take sample.
- Calculace deltas.

Calculation

- Samples (convert all time to milliseconds):
  - wall time: delta wall time (ms).
  - DB time: delta DB time (ms).
  - LIO: delta LIO.

  - workload = DB time / wall time.
  - response time = DB time / LIO.
  - throughput = LIO / wall time.

DIY sampler

- v$sys_time_model.
- v$sysstat.
- (v$system_event).
- drill down:
  - v$sess_time_model.
  - v$sesstat.
  - (v$session_event).
- v$sysmetric.

Tests #1

- server (vm hypervisor): 12-CPU
- 4-CPU virtual machine
- Oracle 19c (19.14.0) database
- tests:
  - parallel = 1
  - parallel = 2
  - parallel = 4
  - parallel = 8

Tests #2

- Java, parallel threads.
  - (DBMS_SCHEDULER, bash, ...).
- Test SQL:
  - prepare:
    - CREATE TABLE t_samples AS SELECT * FROM dba_objects; -- source data
  - test:
    - in endless loop:
      - INSERT INTO global_temporary_table SELECT * FROM t_samples;
      - COMMIT; -- clear inserted data

Tests #3

- Testcase 1: empty machine – only database (test name = NORMAL).
- Testcase 2: overloaded VM:
  stress --cpu 4
  (testname LOCAL LOAD)
- Testcase 3: overloaded server (vm hypervisor)
  stress --cpu 12
  (testname HOST LOAD)

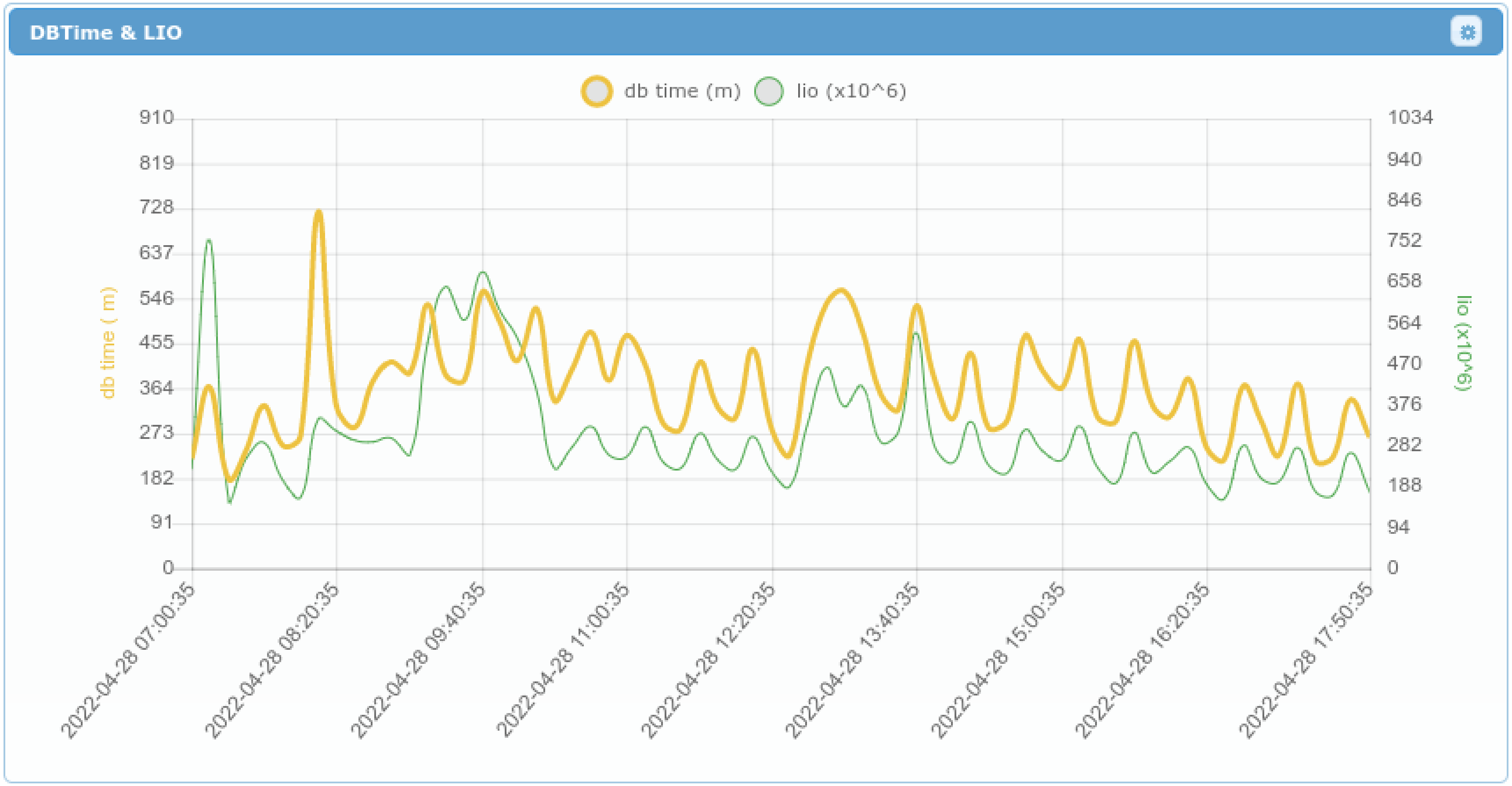
Test: NORMAL (data)

Threads	Wall Time(ms)	DB time(ms)	Workload	Throughput (LIO/ms)	Response Time (ms/LIO)
1	300000	304468,34	0,9719	56,00671	0,018121
2	300000	603450,95	1,9131	109,91091	0,018301
4	300000	1204773,67	3,8307	210,24094	0,019101

Response Time

- rate of work_done and work_time is linear.
- as work_done increases so does the work_time.
- Response Time is constant.

Work done and work time



Abakus As na disku.

dejavu Data at your service.

Response Time

- rate of work_done and work_time is linear.
- as work_done increases so does the work_time.
- Response Time is constant.
- until ...

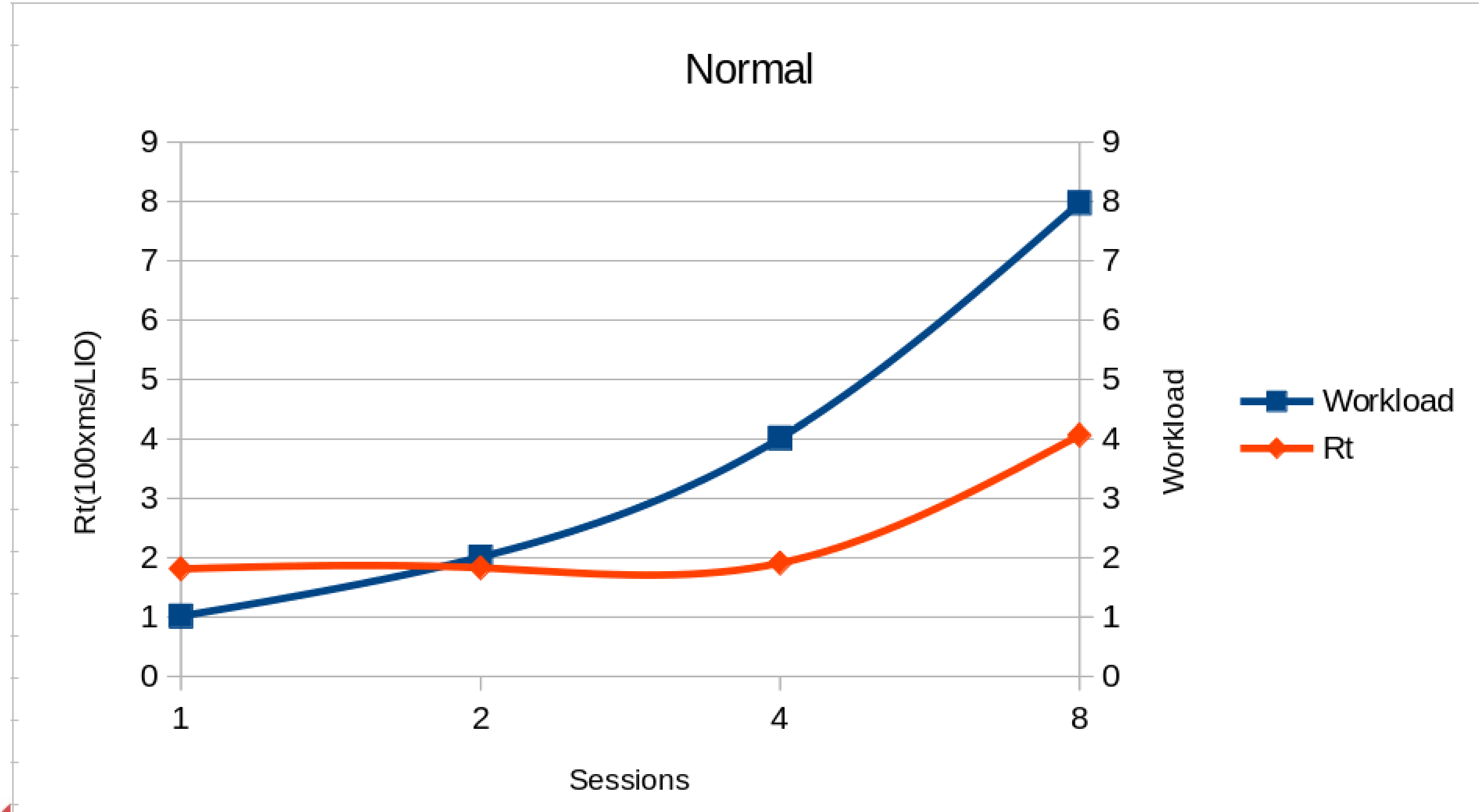
Response Time

- rate of work_done and work_time is linear.
- as work_done increases so does the work_time.
- Response Time is constant.
- until ...
- the system get’s too busy.

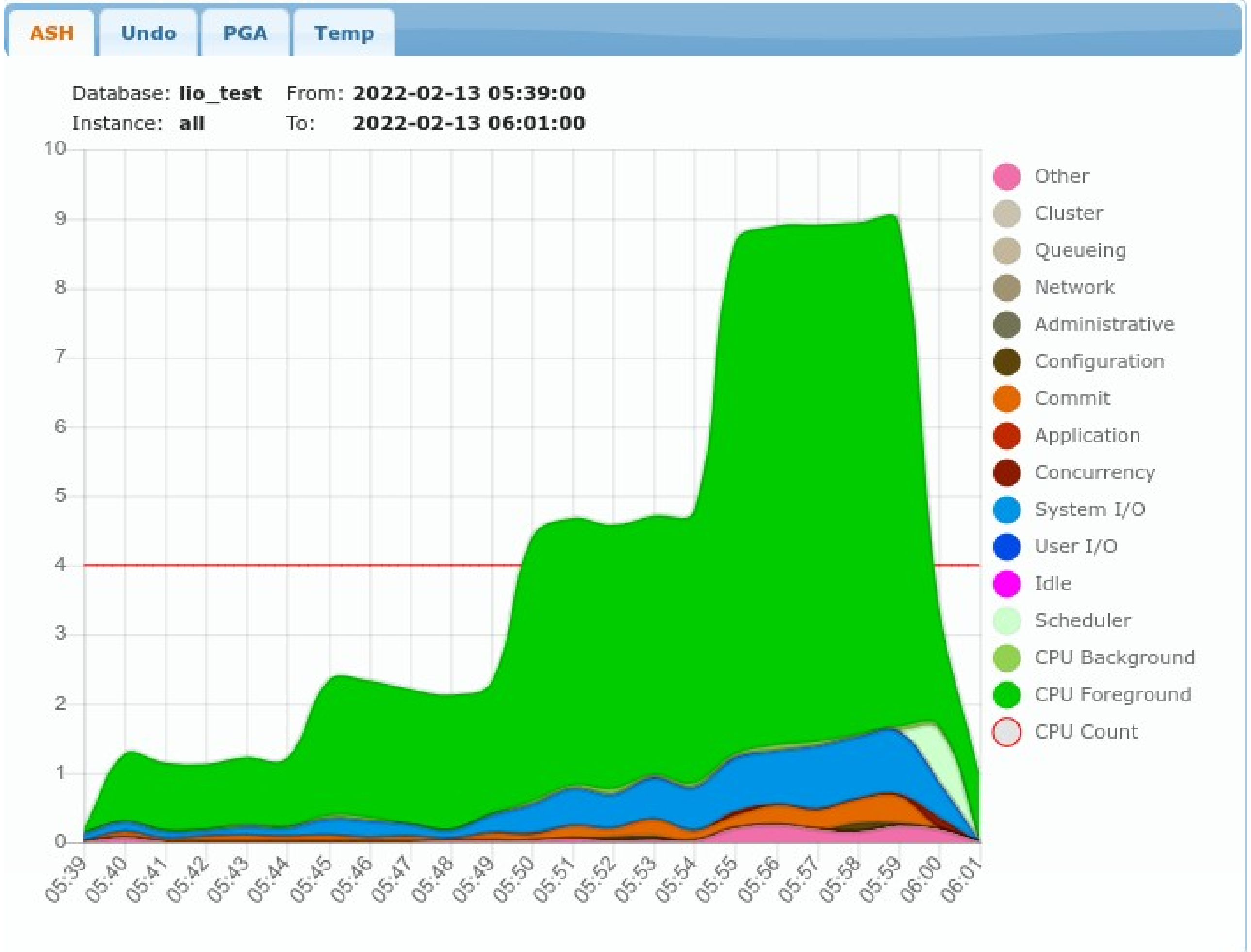
Test: NORMAL (data)

Threads	Wall Time(ms)	DB time(ms)	Workload	Throughput (LIO/ms)	Response Time (ms/LIO)
1	300000	304468,34	0,9719	56,00671	0,018121
2	300000	603450,95	1,9131	109,91091	0,018301
4	300000	1204773,67	3,8307	210,24094	0,019101
8	300000	2392955,59	7,6253	196,23527	0,040648

Test: NORMAL (graph)



Test: NORMAL (ash)



ASH | Undo | PGA | Temp

Database: **lio_test** From: **2022-02-13 05:39:00**
Instance: **all** To: **2022-02-13 06:01:00**

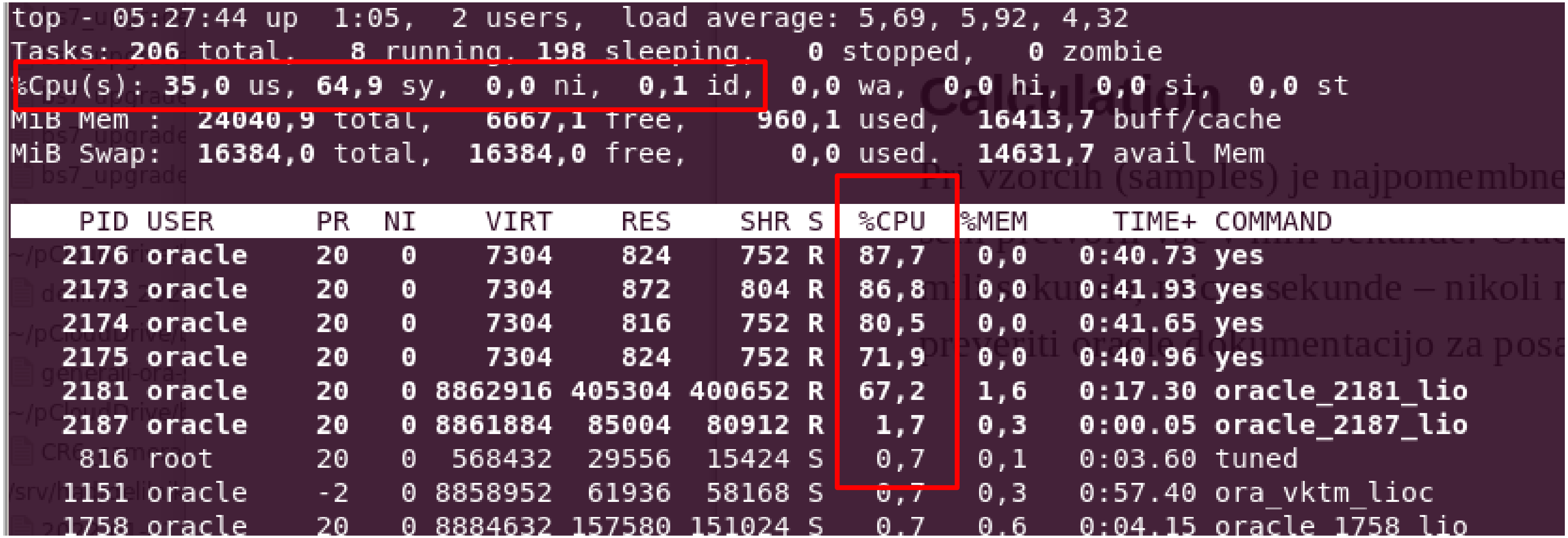
Work Done (LIO): 16.802.014

AS Abakus As na disku.

dejavu Data at your service.

LOCAL_LOAD (top)

```
$ for i in $(seq $(getconf _NPROCESSORS_ONLN)); do yes > /dev/null & done
```



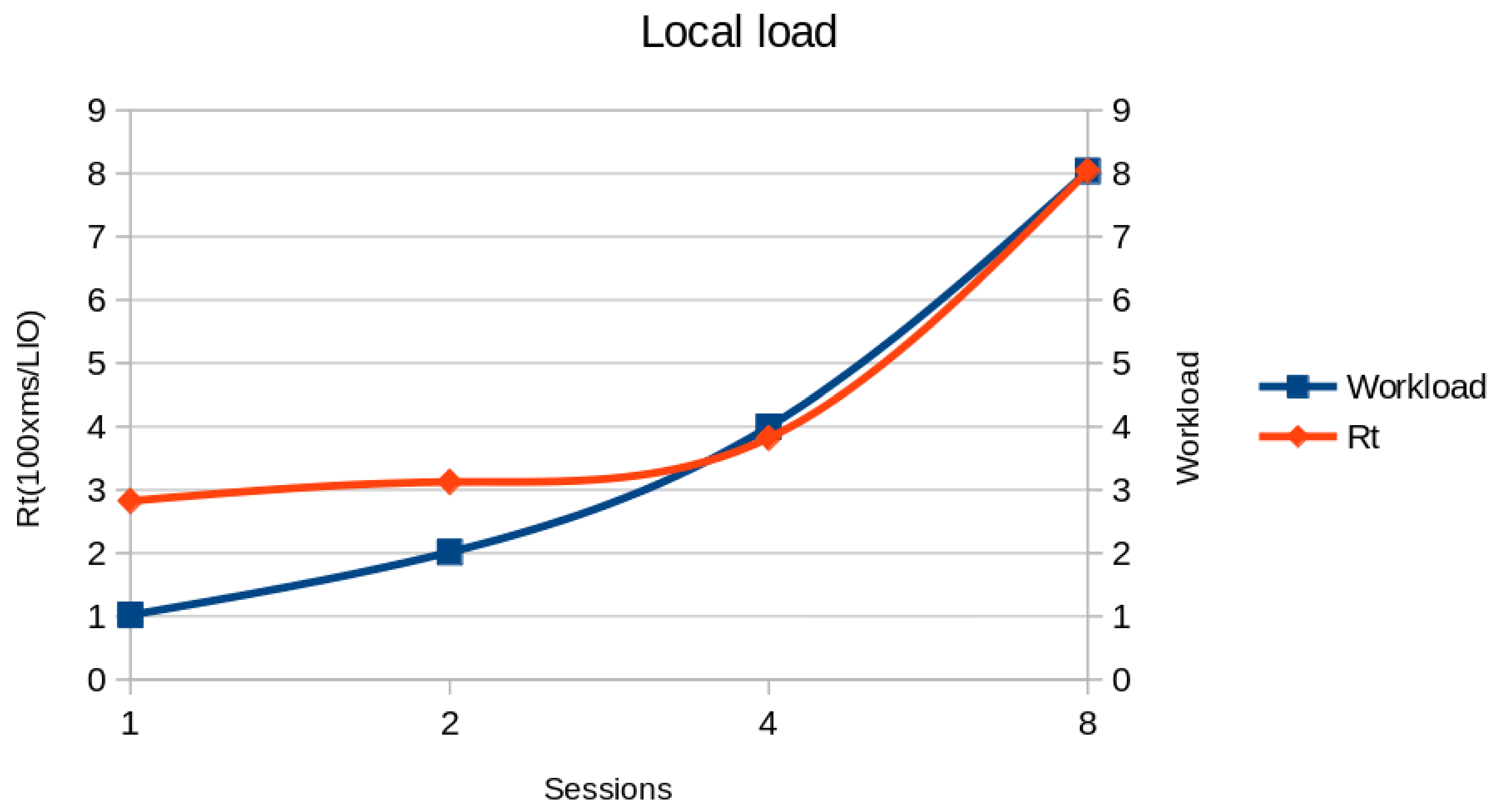
```
top - 05:27:44 up  1:05,  2 users,  load average: 5,69, 5,92, 4,32
Tasks: 206 total,   8 running, 198 sleeping,   0 stopped,   0 zombie
%Cpu(s): 35,0 us, 64,9 sy,  0,0 ni,  0,1 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :  24040,9 total,   6667,1 free,    960,1 used,  16413,7 buff/cache
MiB Swap:  16384,0 total,  16384,0 free,      0,0 used.  14631,7 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2176 oracle    20   0    7304    824    752 R  87,7   0,0   0:40.73 yes
 2173 oracle    20   0    7304    872    804 R  86,8   0,0   0:41.93 yes
 2174 oracle    20   0    7304    816    752 R  80,5   0,0   0:41.65 yes
 2175 oracle    20   0    7304    824    752 R  71,9   0,0   0:40.96 yes
 2181 oracle    20   0 8862916 405304 400652 R  67,2   1,6   0:17.30 oracle_2181_lio
 2187 oracle    20   0 8861884  85004  80912 R   1,7   0,3   0:00.05 oracle_2187_lio
  816 root      20   0  568432  29556  15424 S   0,7   0,1   0:03.60 tuned
 1151 oracle    -2   0 8858952  61936  58168 S   0,7   0,3   0:57.40 ora_vktm_lioc
 1758 oracle    20   0 8884632 157580 151024 S   0,7   0,6   0:04.15 oracle_1758_lio
```

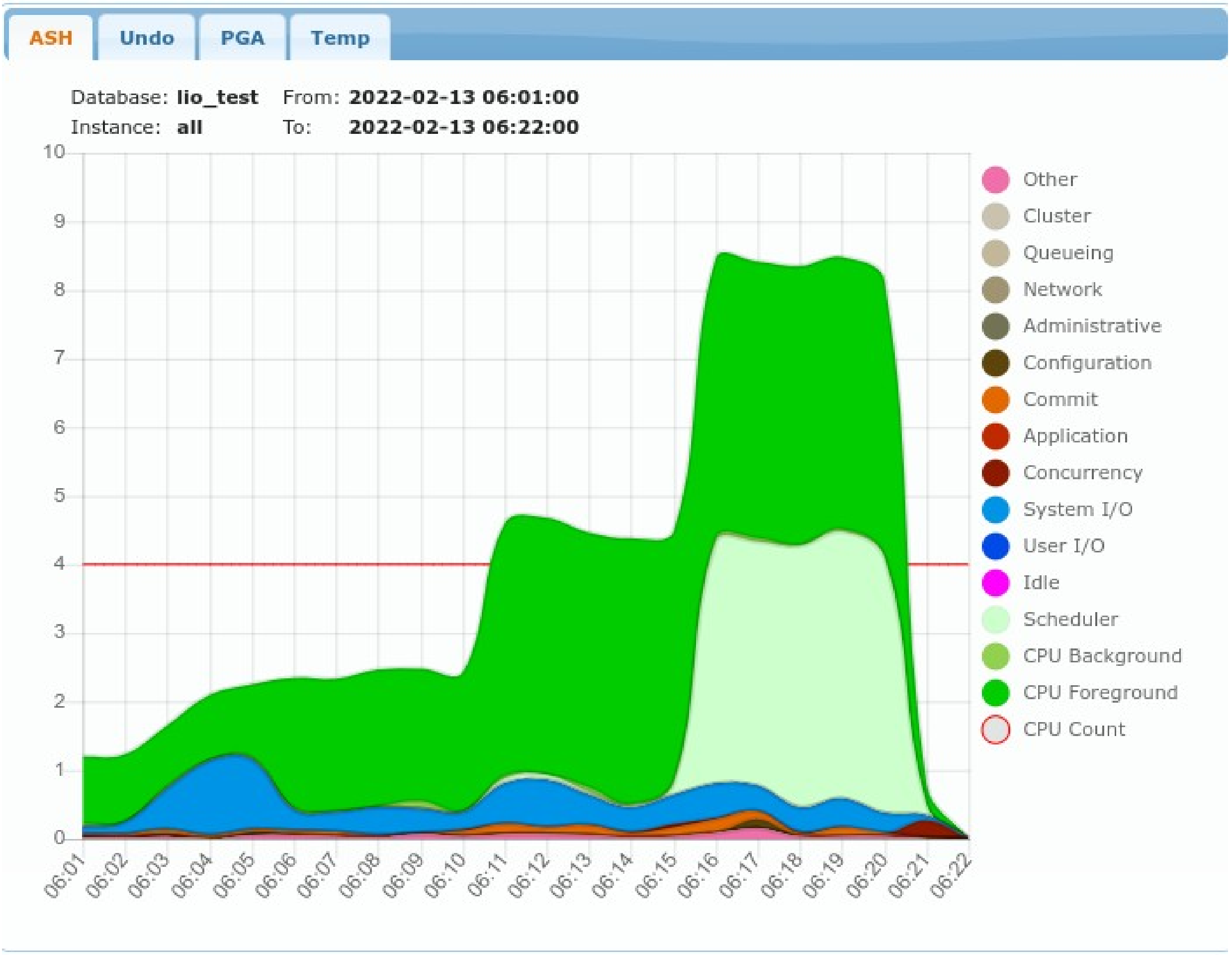
Test: LOCAL_LOAD (data)

Threads	Wall Time(ms)	DB time(ms)	Workload	Throughput (LIO/ms)	Response Time (ms/LIO)
1	300000	306109,6	1,02037	36,11619	0,028252
2	300000	604716,22	2,01572	64,5119	0,031246
4	300000	1199513,86	3,99838	104,62531	0,038216
8	300000	2410920,08	8,0364	99,94473	0,080408

Test: LOCAL_LOAD (graph)



Test: LOCAL_LOAD (ash)



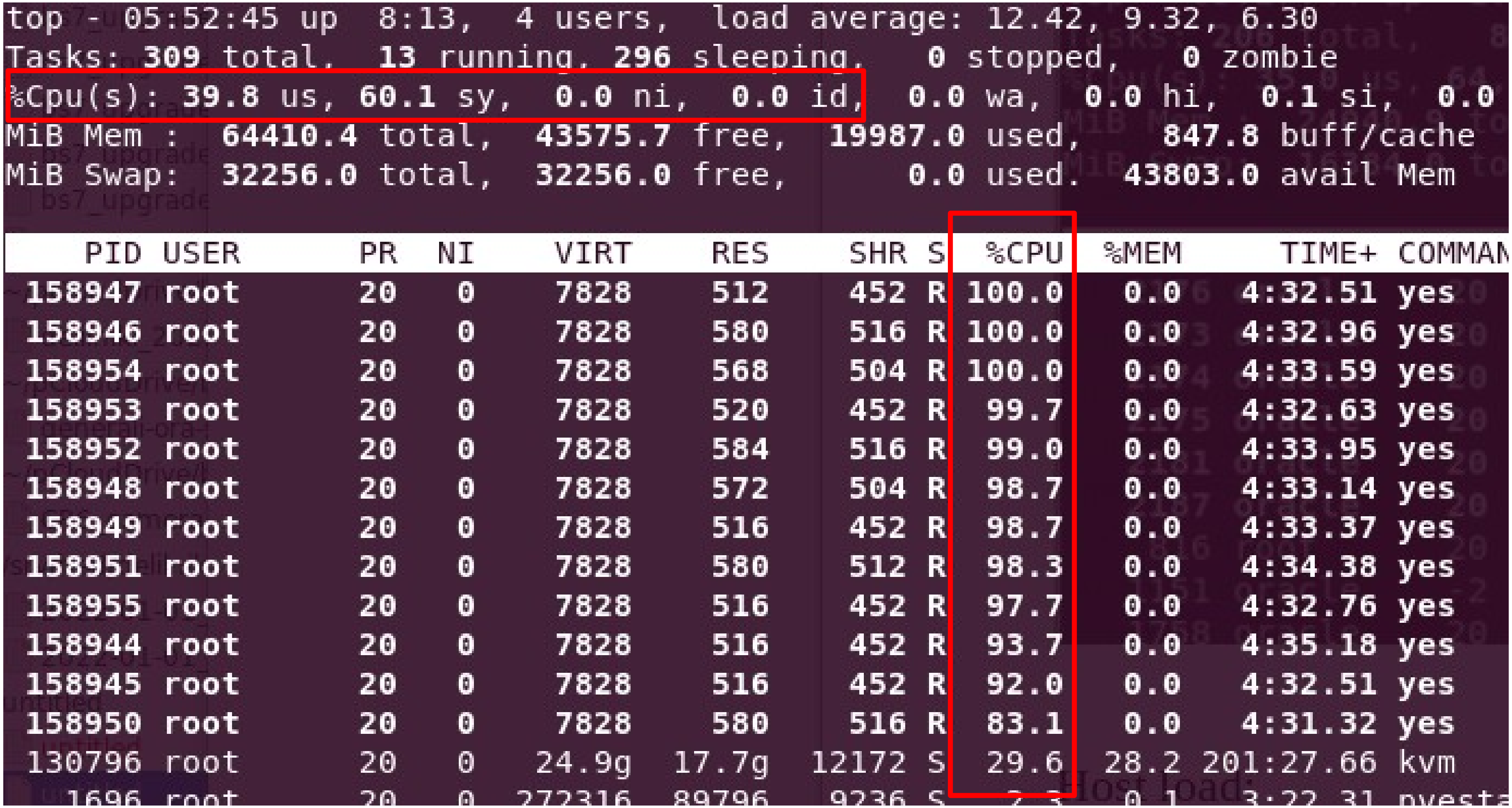
Database: **lio_test** From: **2022-02-13 06:01:00**
Instance: **all** To: **2022-02-13 06:22:00**

Work Done (LIO): 10.834.856

AS Abakus As na disku.

dejavu Data at your service.

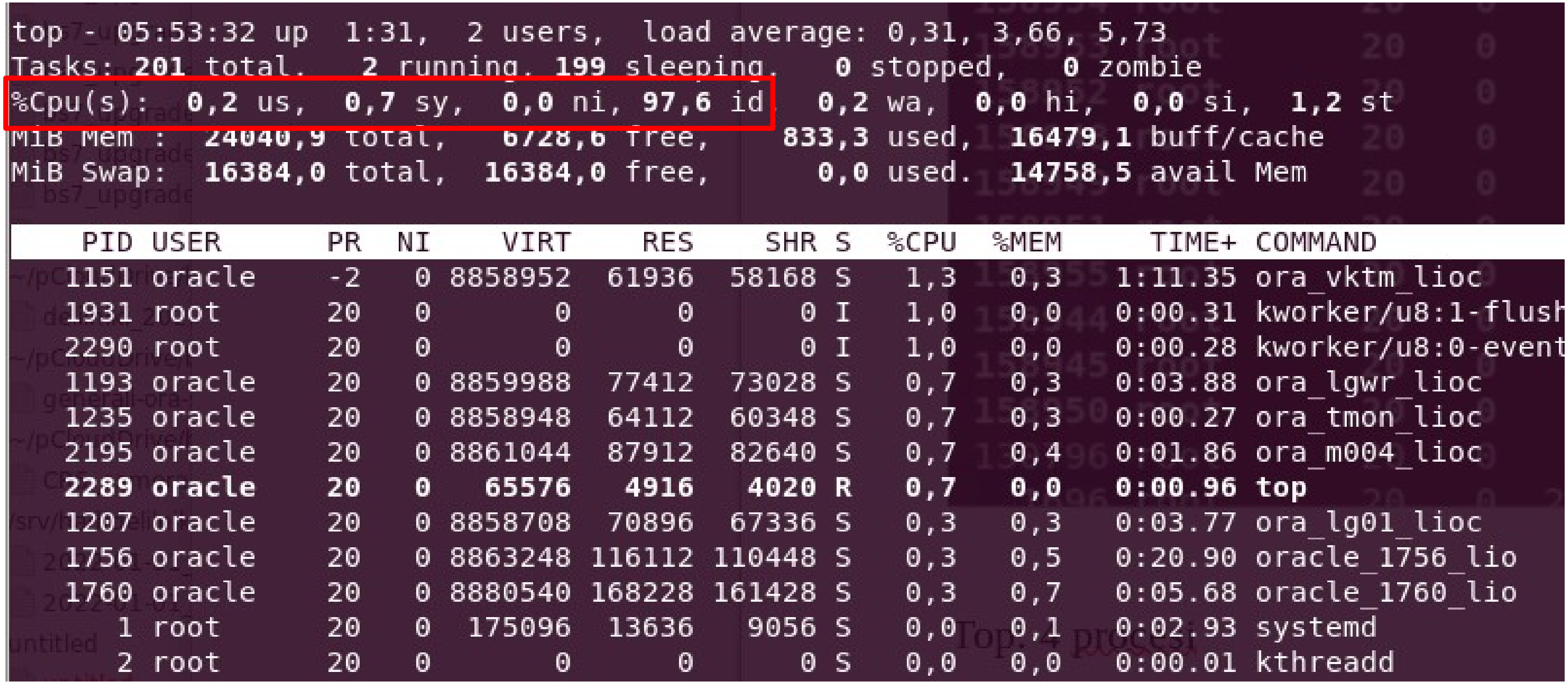
HOST_LOAD (top)



```
top - 05:52:45 up  8:13,  4 users,  load average: 12.42, 9.32, 6.30
Tasks: 309 total,  13 running, 296 sleeping,   0 stopped,   0 zombie
%Cpu(s): 39.8 us, 60.1 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0
MiB Mem :  64410.4 total,  43575.7 free,  19987.0 used,    847.8 buff/cache
MiB Swap:  32256.0 total,  32256.0 free,      0.0 used.  43803.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAN
 158947 root      20   0    7828    512    452 R 100.0   0.0   4:32.51 yes
 158946 root      20   0    7828    580    516 R 100.0   0.0   4:32.96 yes
 158954 root      20   0    7828    568    504 R 100.0   0.0   4:33.59 yes
 158953 root      20   0    7828    520    452 R  99.7   0.0   4:32.63 yes
 158952 root      20   0    7828    584    516 R  99.0   0.0   4:33.95 yes
 158948 root      20   0    7828    572    504 R  98.7   0.0   4:33.14 yes
 158949 root      20   0    7828    516    452 R  98.7   0.0   4:33.37 yes
 158951 root      20   0    7828    580    512 R  98.3   0.0   4:34.38 yes
 158955 root      20   0    7828    516    452 R  97.7   0.0   4:32.76 yes
 158944 root      20   0    7828    516    452 R  93.7   0.0   4:35.18 yes
 158945 root      20   0    7828    516    452 R  92.0   0.0   4:32.51 yes
 158950 root      20   0    7828    580    516 R  83.1   0.0   4:31.32 yes
 130796 root      20   0   24.9g  17.7g  12172 S  29.6  28.2 201:27.66 kvm
   1696 root      20   0  272316  89796   9236 S   2.3   0.1   3:22.31 pvesta
```

HOST_LOAD (VM)



```
top - 05:53:32 up  1:31,  2 users,  load average: 0,31, 3,66, 5,73
Tasks: 201 total,   2 running, 199 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,2 us,  0,7 sy,  0,0 ni, 97,6 id,  0,2 wa,  0,0 hi,  0,0 si,  1,2 st
MiB Mem :  24040,9 total,   6728,6 free,    833,3 used,  16479,1 buff/cache
MiB Swap:  16384,0 total,  16384,0 free,      0,0 used.  14758,5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1151 oracle    -2   0 8858952  61936  58168 S   1,3   0,3   1:11.35 ora_vktm_lioc
   1931 root      20   0       0      0      0 I   1,0   0,0   0:00.31 kworker/u8:1-flush
   2290 root      20   0       0      0      0 I   1,0   0,0   0:00.28 kworker/u8:0-event
   1193 oracle    20   0 8859988  77412  73028 S   0,7   0,3   0:03.88 ora_lgwr_lioc
   1235 oracle    20   0 8858948  64112  60348 S   0,7   0,3   0:00.27 ora_tmon_lioc
   2195 oracle    20   0 8861044  87912  82640 S   0,7   0,4   0:01.86 ora_m004_lioc
   2289 oracle    20   0   65576   4916   4020 R   0,7   0,0   0:00.96 top
   1207 oracle    20   0 8858708  70896  67336 S   0,3   0,3   0:03.77 ora_lg01_lioc
   1756 oracle    20   0 8863248 116112 110448 S   0,3   0,5   0:20.90 oracle_1756_lio
   1760 oracle    20   0 8880540 168228 161428 S   0,3   0,7   0:05.68 oracle_1760_lio
      1 root      20   0  175096  13636   9056 S   0,0   0,1   0:02.93 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.01 kthreadd
```

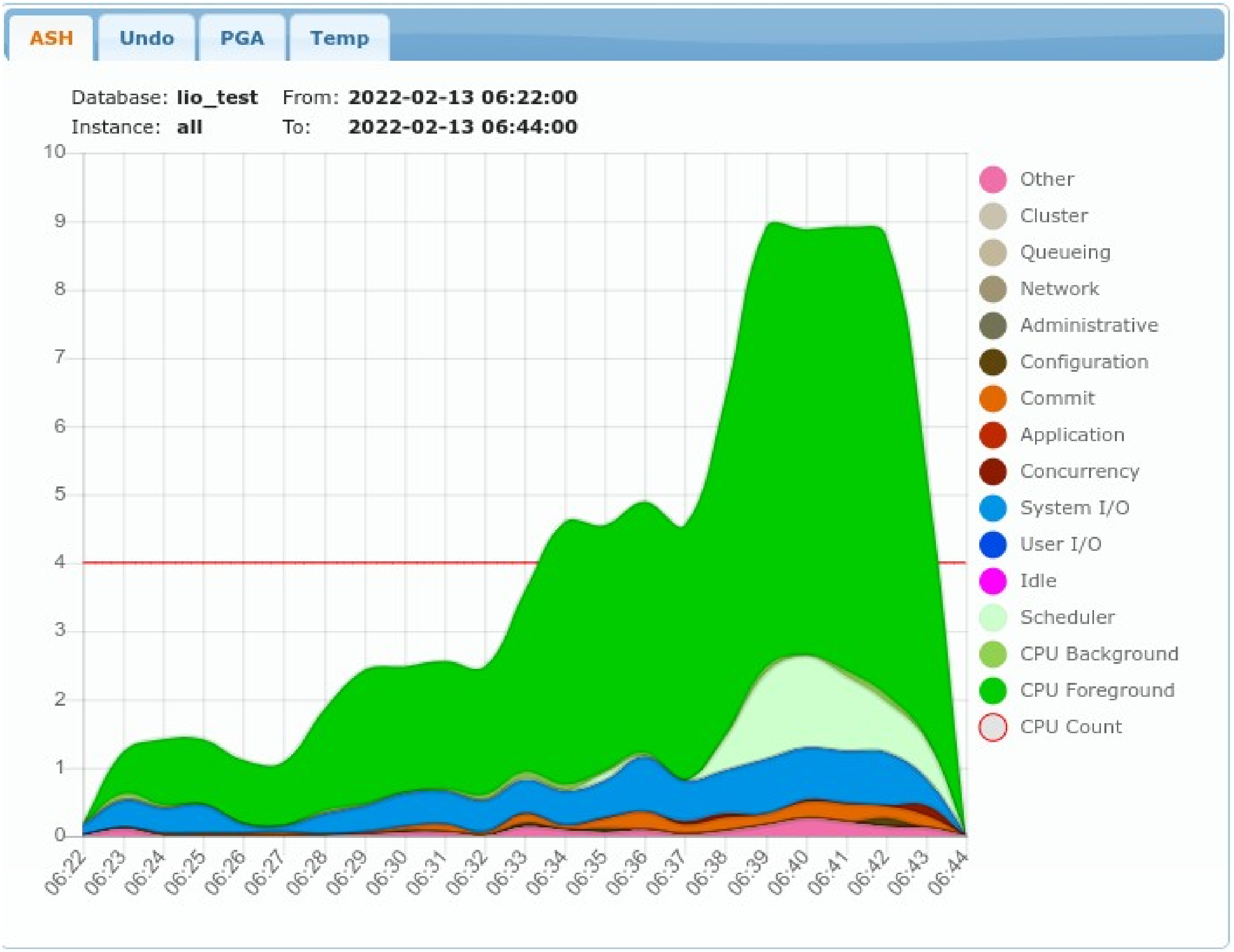
AS Abakus As na disku.

dejavu Data at your service.

Test: HOST_LOAD (data)

Threads	Wall Time(ms)	DB time(ms)	Workload	Throughput (LIO/ms)	Response Time (ms/LIO)
1	300000	315294,26	1,05098	37,46018	0,028056
2	300000	619339,74	2,06447	76,47778	0,026994
4	300000	1206753,46	4,02251	156,00989	0,025784
8	300000	2412776,96	8,04259	170,87222	0,047068

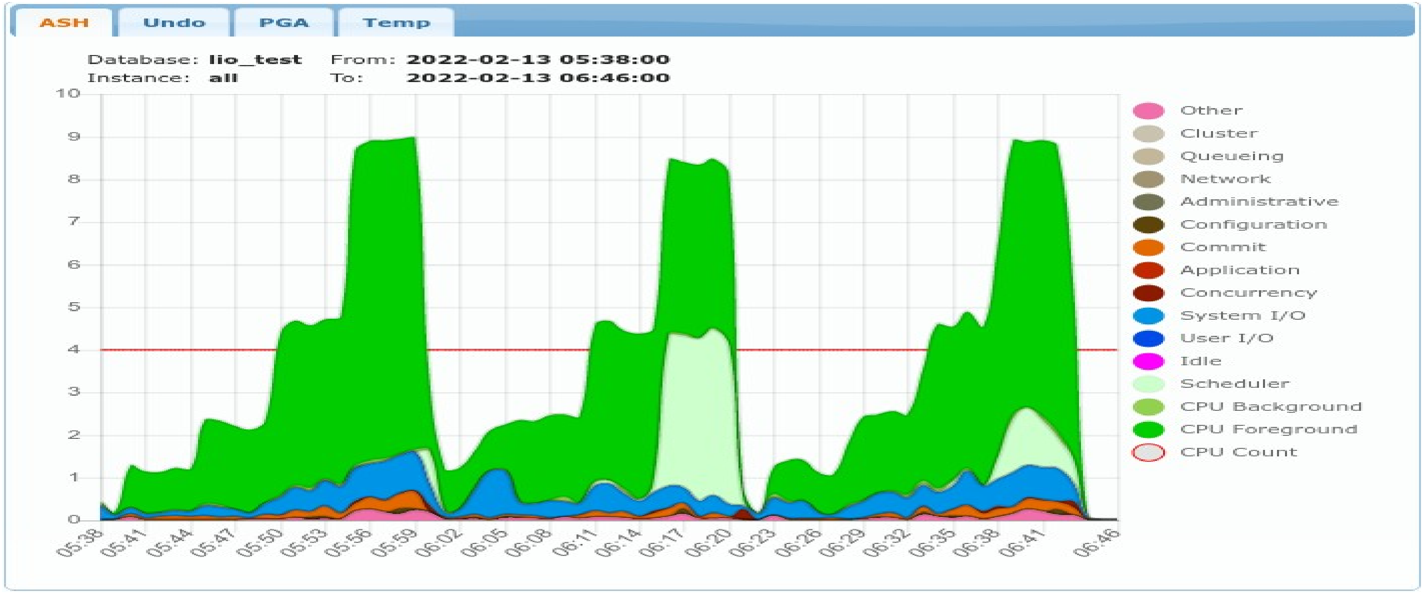
Test: HOST_LOAD (ash)



Database: **lio_test** From: **2022-02-13 06:22:00**
Instance: **all** To: **2022-02-13 06:44:00**

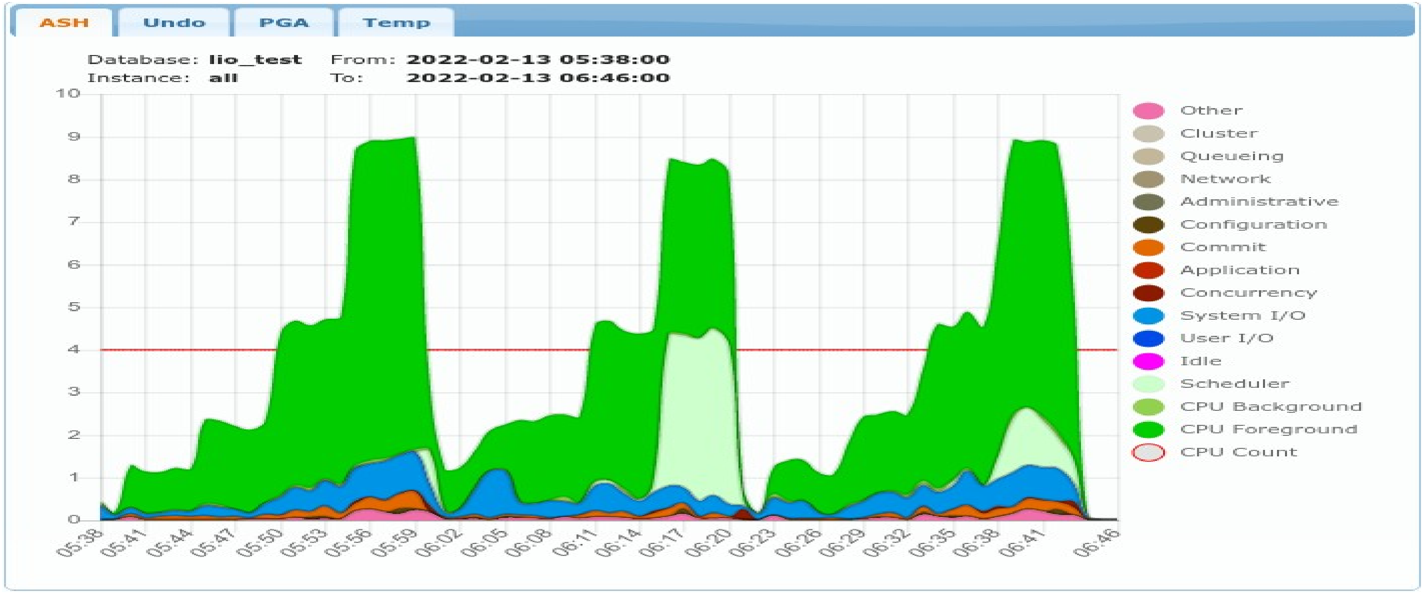
Work Done (LIO): 13.860.714

ASH

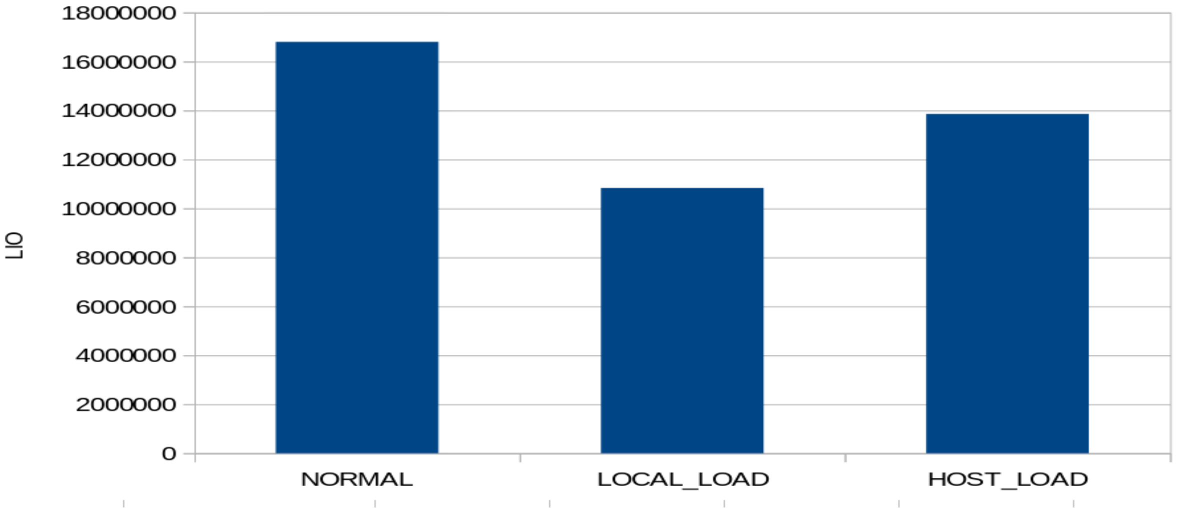


Database: **lio_test** From: **2022-02-13 05:38:00**
Instance: **all** To: **2022-02-13 06:46:00**

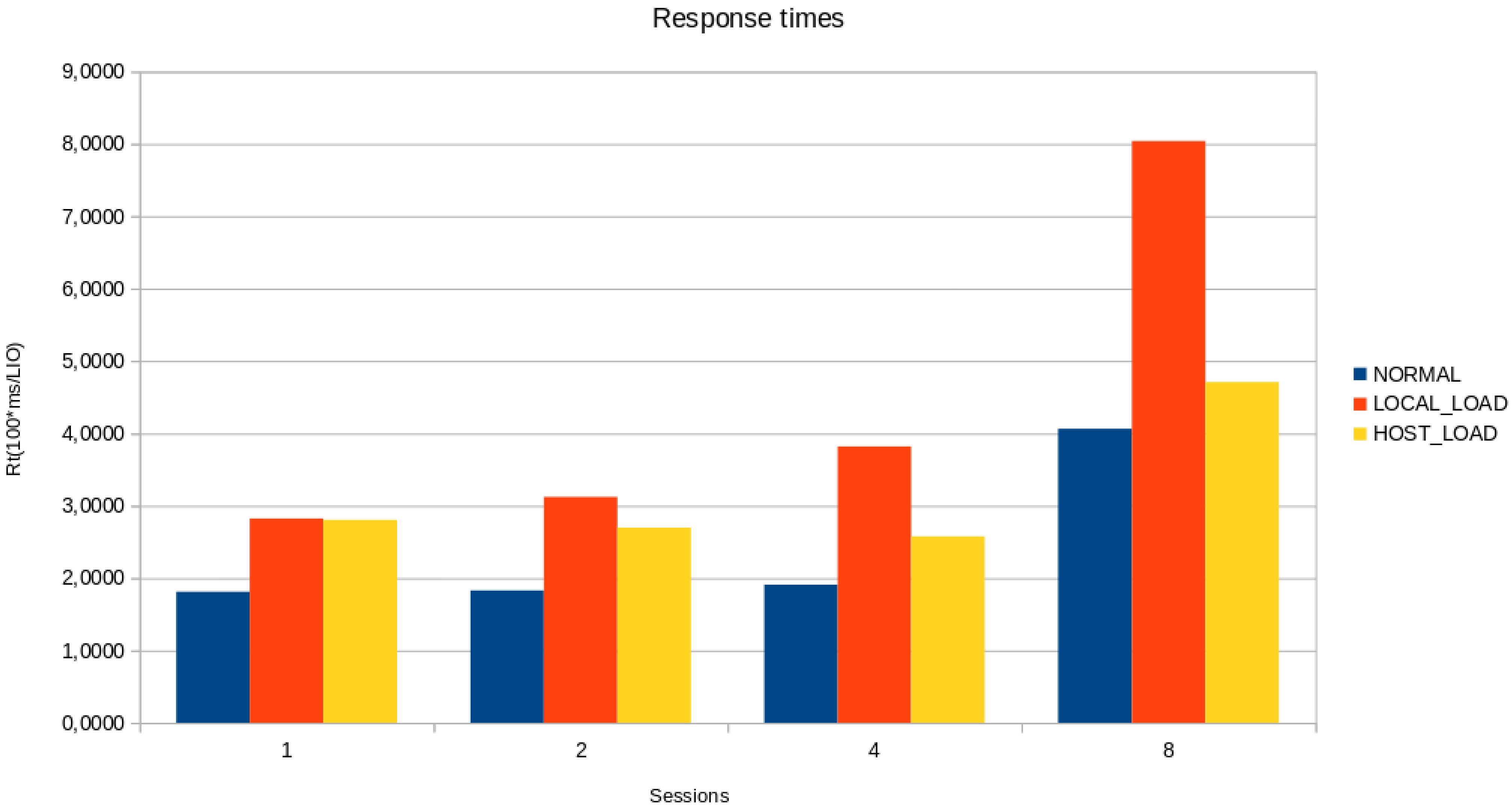
ASH



Database: lio_test From: 2022-02-13 05:38:00
Instance: all To: 2022-02-13 06:46:00



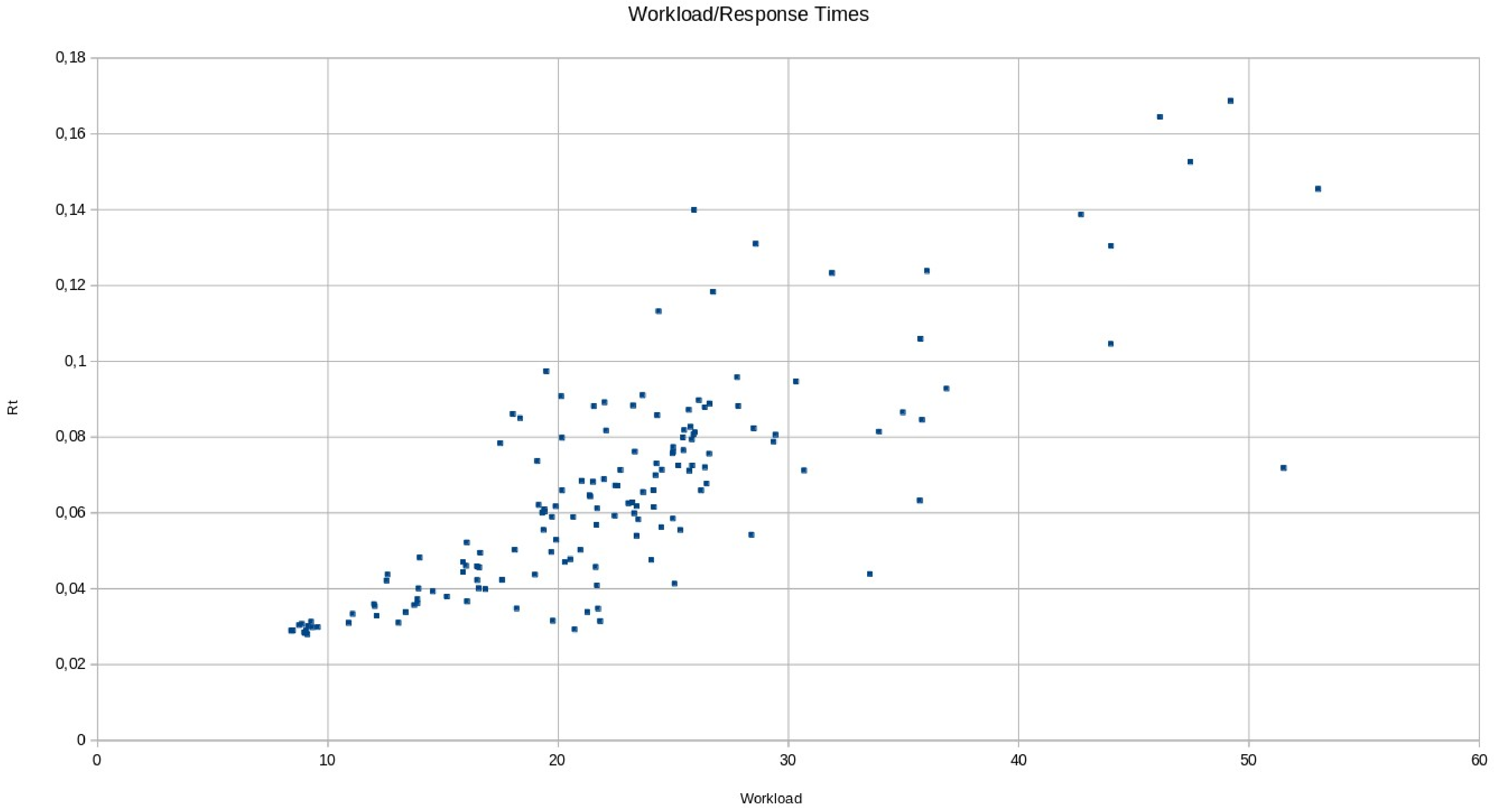
Load tests - compare



Production samples

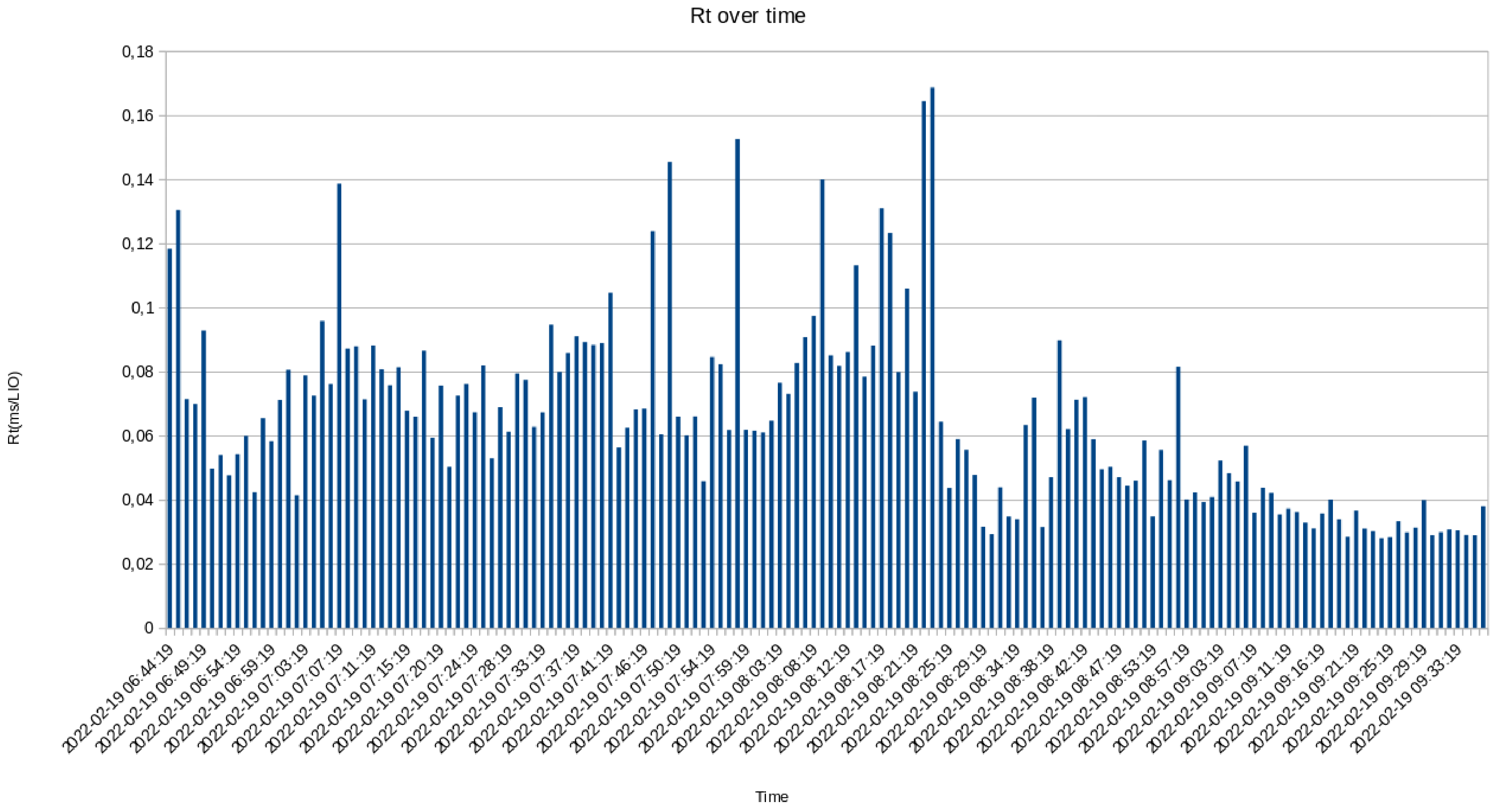
- Exadata, Oracle 11.2.0.4 EE
  - cpu_count = 12.

Production samples – #1

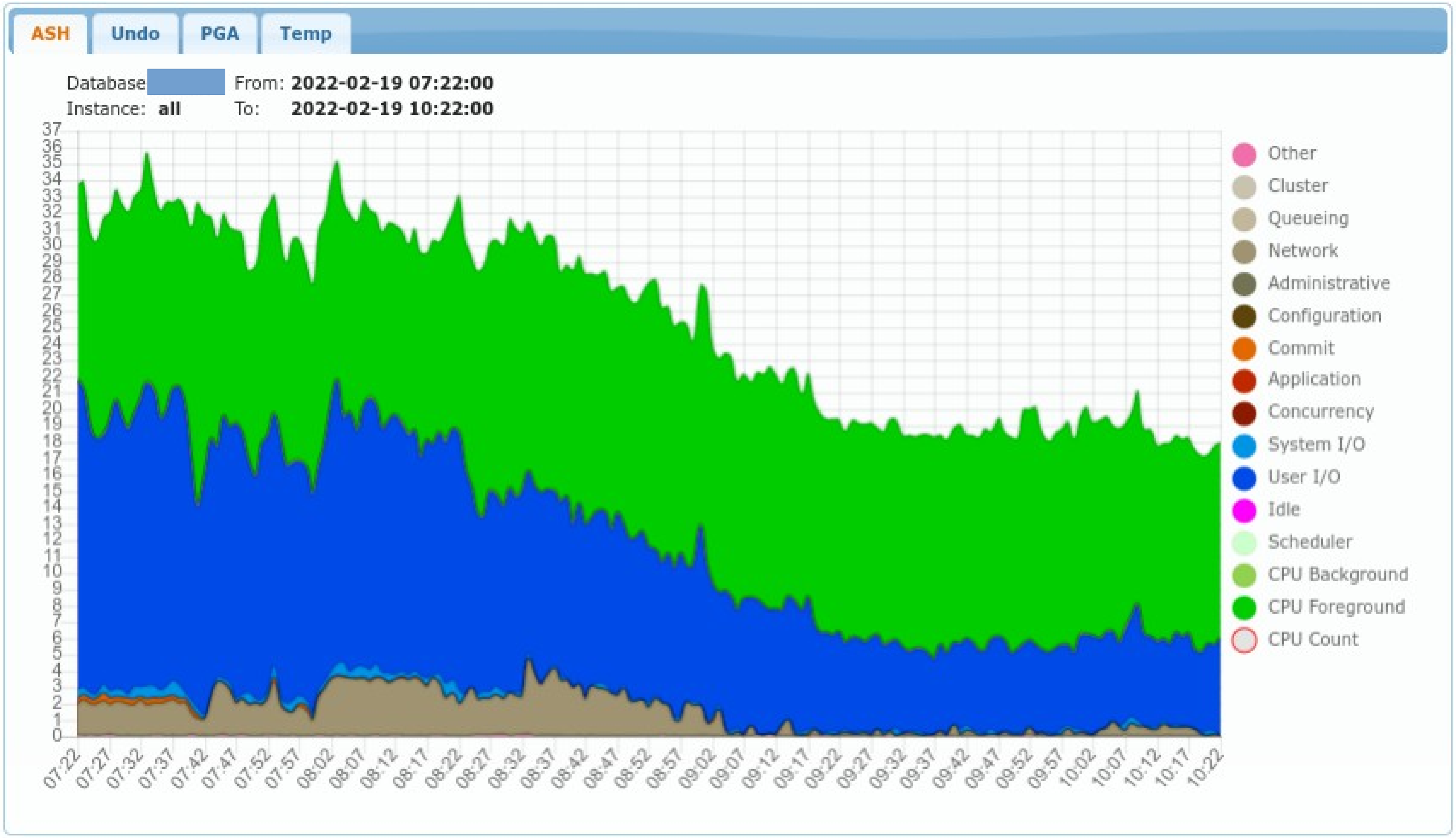


Data at your service.

Production samples – #1 (timeline)



Production samples – #1 (ash)



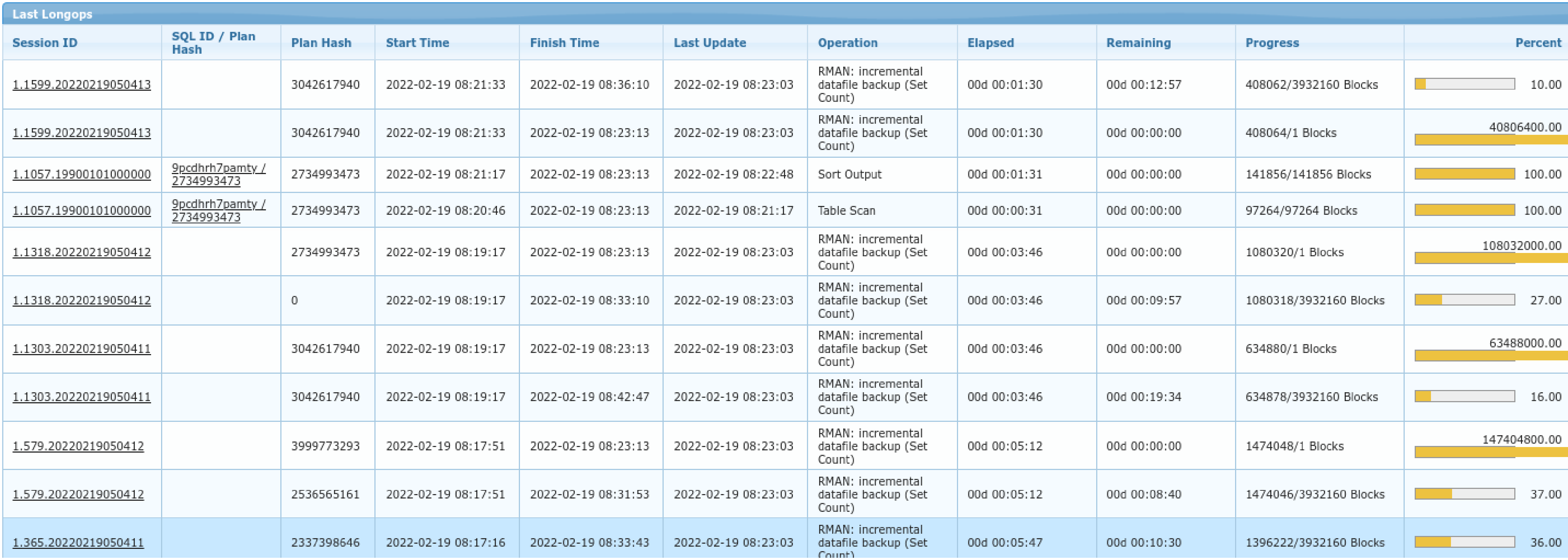
Database [redacted] From: **2022-02-19 07:22:00**

Instance: **all** To: **2022-02-19 10:22:00**

AS Abakus As na disku.

dejavu *Data at your service.*

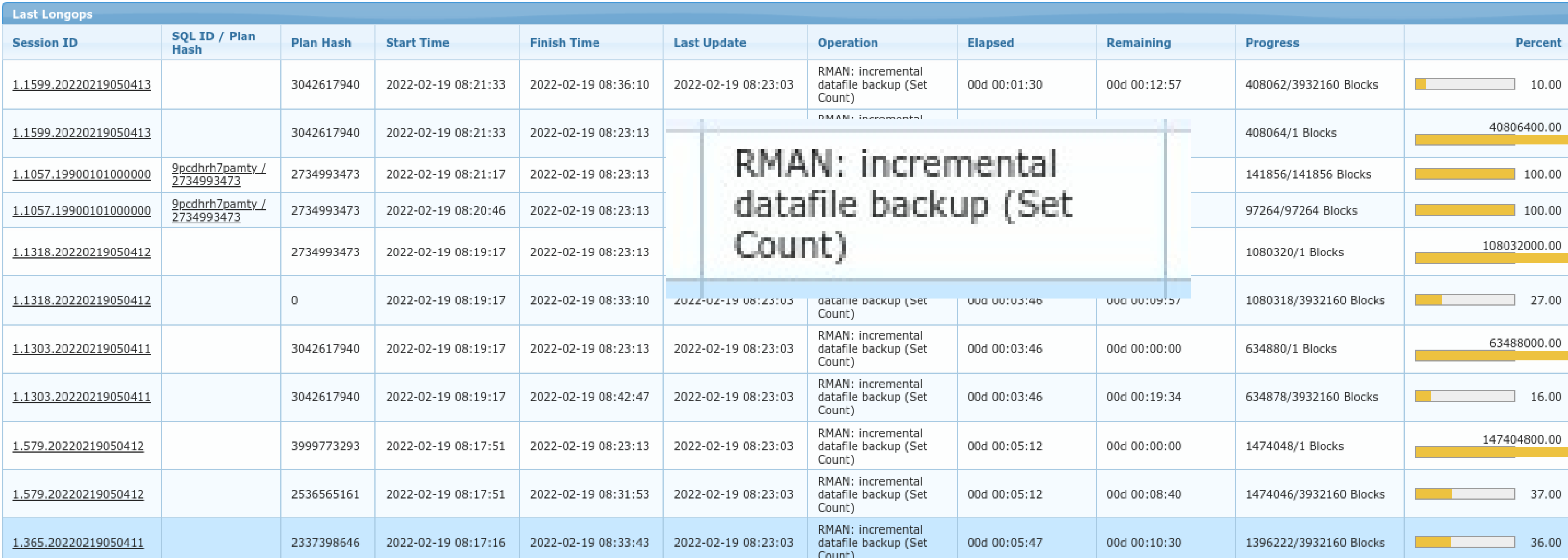
Production samples – #1 (ash)



Last Longops

Session ID	SQL ID / Plan Hash	Plan Hash	Start Time	Finish Time	Last Update	Operation	Elapsed	Remaining	Progress	Percent
1.1599.20220219050413		3042617940	2022-02-19 08:21:33	2022-02-19 08:36:10	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:01:30	00d 00:12:57	408062/3932160 Blocks	10.00
1.1599.20220219050413		3042617940	2022-02-19 08:21:33	2022-02-19 08:23:13	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:01:30	00d 00:00:00	408064/1 Blocks	40806400.00
1.1057.19900101000000	9pcdhrh7pamty / 2734993473	2734993473	2022-02-19 08:21:17	2022-02-19 08:23:13	2022-02-19 08:22:48	Sort Output	00d 00:01:31	00d 00:00:00	141856/141856 Blocks	100.00
1.1057.19900101000000	9pcdhrh7pamty / 2734993473	2734993473	2022-02-19 08:20:46	2022-02-19 08:23:13	2022-02-19 08:21:17	Table Scan	00d 00:00:31	00d 00:00:00	97264/97264 Blocks	100.00
1.1318.20220219050412		2734993473	2022-02-19 08:19:17	2022-02-19 08:23:13	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:03:46	00d 00:00:00	1080320/1 Blocks	108032000.00
1.1318.20220219050412		0	2022-02-19 08:19:17	2022-02-19 08:33:10	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:03:46	00d 00:09:57	1080318/3932160 Blocks	27.00
1.1303.20220219050411		3042617940	2022-02-19 08:19:17	2022-02-19 08:23:13	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:03:46	00d 00:00:00	634880/1 Blocks	63488000.00
1.1303.20220219050411		3042617940	2022-02-19 08:19:17	2022-02-19 08:42:47	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:03:46	00d 00:19:34	634878/3932160 Blocks	16.00
1.579.20220219050412		3999773293	2022-02-19 08:17:51	2022-02-19 08:23:13	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:05:12	00d 00:00:00	1474048/1 Blocks	147404800.00
1.579.20220219050412		2536565161	2022-02-19 08:17:51	2022-02-19 08:31:53	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:05:12	00d 00:08:40	1474046/3932160 Blocks	37.00
1.365.20220219050411		2337398646	2022-02-19 08:17:16	2022-02-19 08:33:43	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:05:47	00d 00:10:30	1396222/3932160 Blocks	36.00

Production samples – #1 (ash)

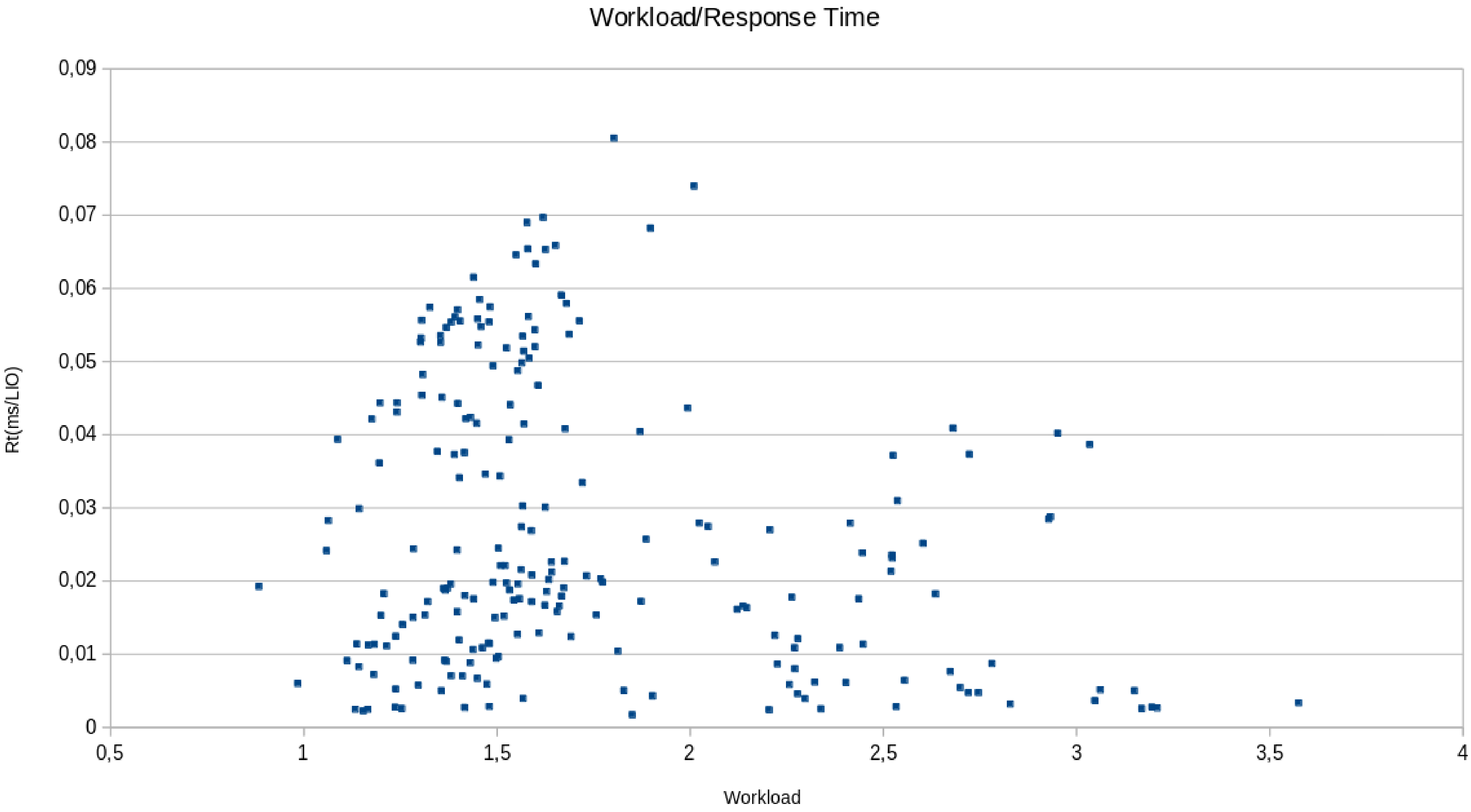


Last Longops

Session ID	SQL ID / Plan Hash	Plan Hash	Start Time	Finish Time	Last Update	Operation	Elapsed	Remaining	Progress	Percent
1.1599.20220219050413		3042617940	2022-02-19 08:21:33	2022-02-19 08:36:10	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:01:30	00d 00:12:57	408062/3932160 Blocks	10.00
1.1599.20220219050413		3042617940	2022-02-19 08:21:33	2022-02-19 08:23:13		RMAN: incremental			408064/1 Blocks	40806400.00
1.1057.19900101000000	9pcdhrh7pamty / 2734993473	2734993473	2022-02-19 08:21:17	2022-02-19 08:23:13					141856/141856 Blocks	100.00
1.1057.19900101000000	9pcdhrh7pamty / 2734993473	2734993473	2022-02-19 08:20:46	2022-02-19 08:23:13					97264/97264 Blocks	100.00
1.1318.20220219050412		2734993473	2022-02-19 08:19:17	2022-02-19 08:23:13					1080320/1 Blocks	108032000.00
1.1318.20220219050412		0	2022-02-19 08:19:17	2022-02-19 08:33:10	2022-02-19 08:23:03	datafile backup (Set Count)	00d 00:03:46	00d 00:09:57	1080318/3932160 Blocks	27.00
1.1303.20220219050411		3042617940	2022-02-19 08:19:17	2022-02-19 08:23:13	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:03:46	00d 00:00:00	634880/1 Blocks	63488000.00
1.1303.20220219050411		3042617940	2022-02-19 08:19:17	2022-02-19 08:42:47	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:03:46	00d 00:19:34	634878/3932160 Blocks	16.00
1.579.20220219050412		3999773293	2022-02-19 08:17:51	2022-02-19 08:23:13	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:05:12	00d 00:00:00	1474048/1 Blocks	147404800.00
1.579.20220219050412		2536565161	2022-02-19 08:17:51	2022-02-19 08:31:53	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:05:12	00d 00:08:40	1474046/3932160 Blocks	37.00
1.365.20220219050411		2337398646	2022-02-19 08:17:16	2022-02-19 08:33:43	2022-02-19 08:23:03	RMAN: incremental datafile backup (Set Count)	00d 00:05:47	00d 00:10:30	1396222/3932160 Blocks	36.00

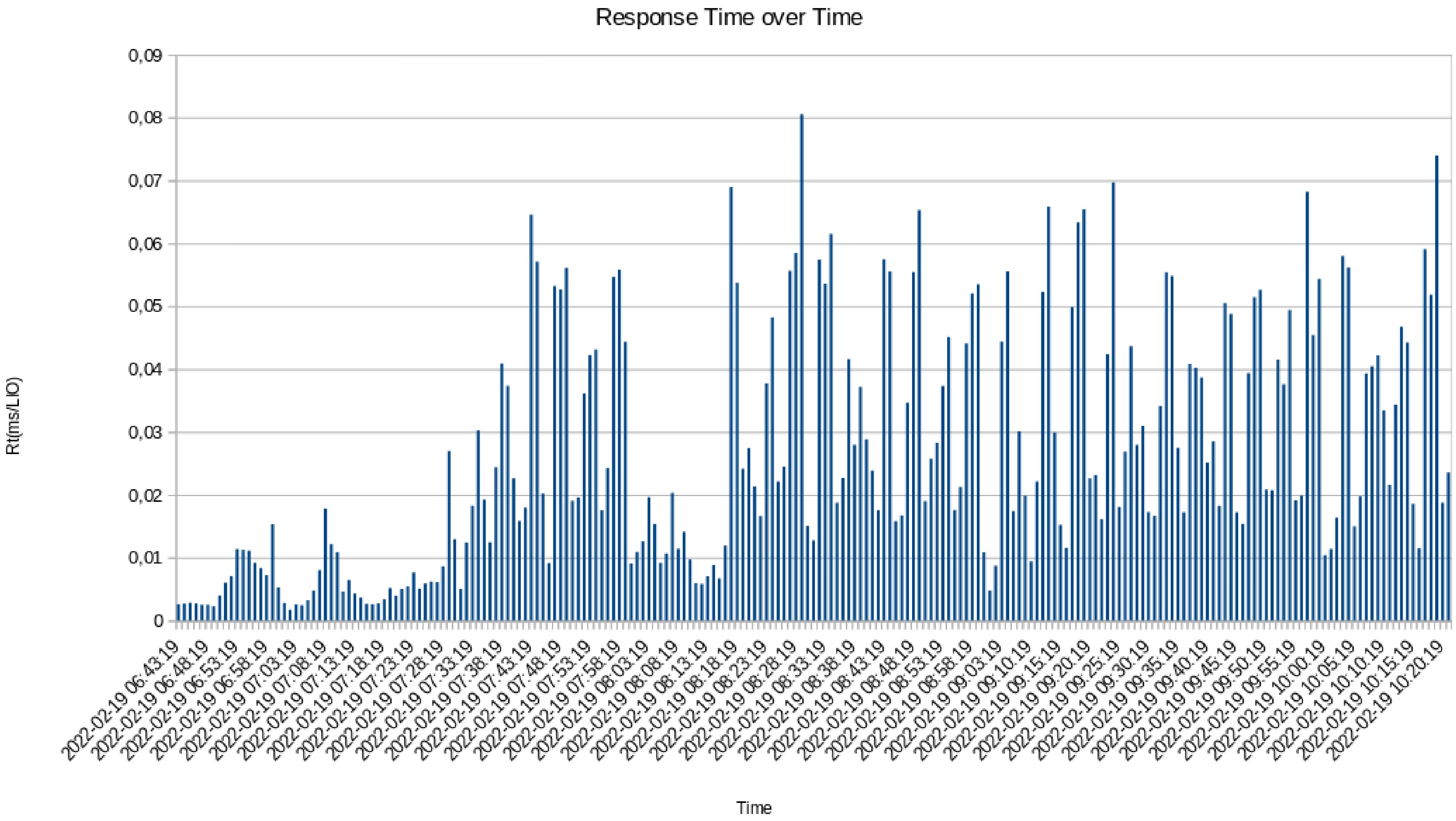
RMAN: incremental datafile backup (Set Count)

Production samples – #2

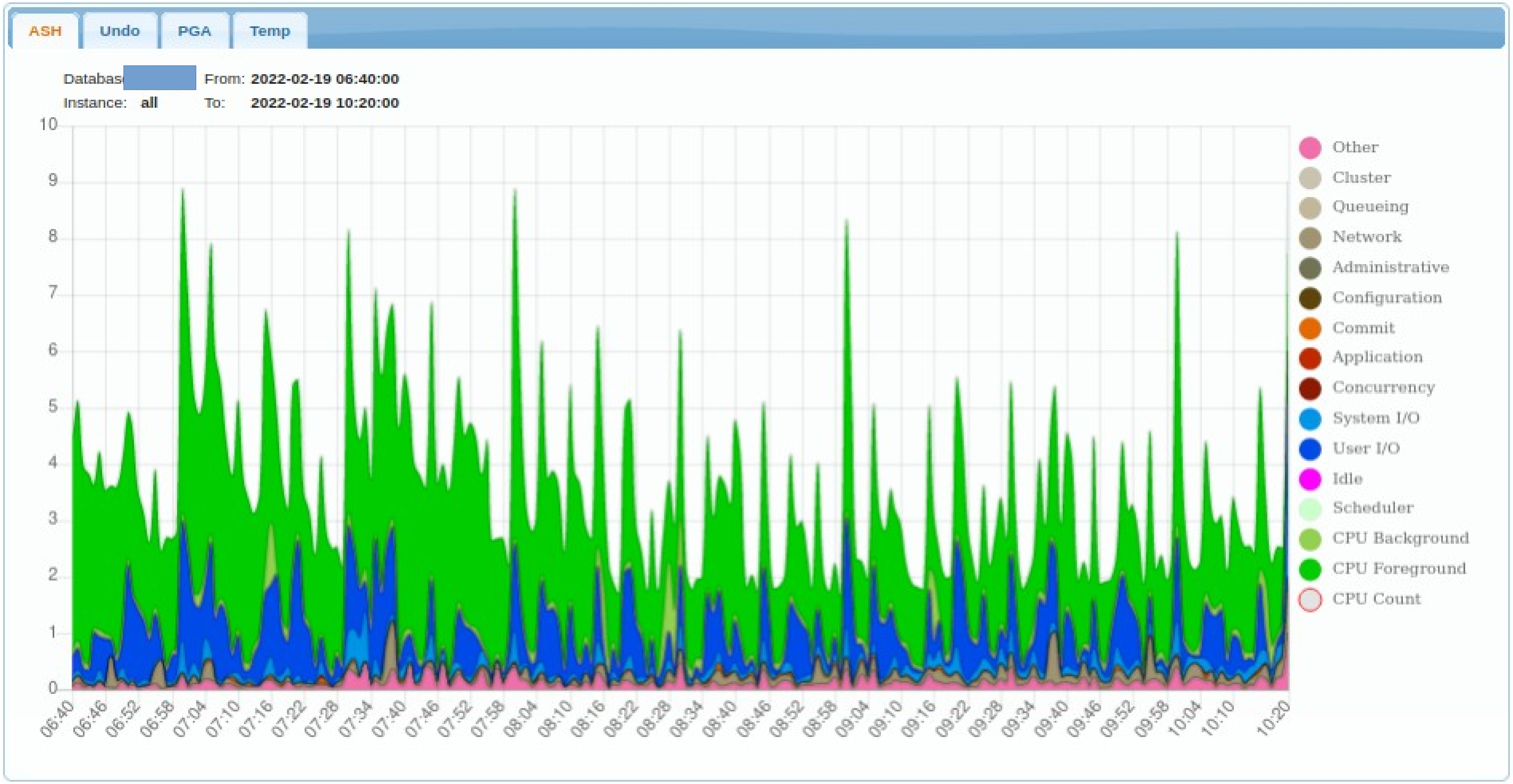


Data at your service.

Production samples – #2



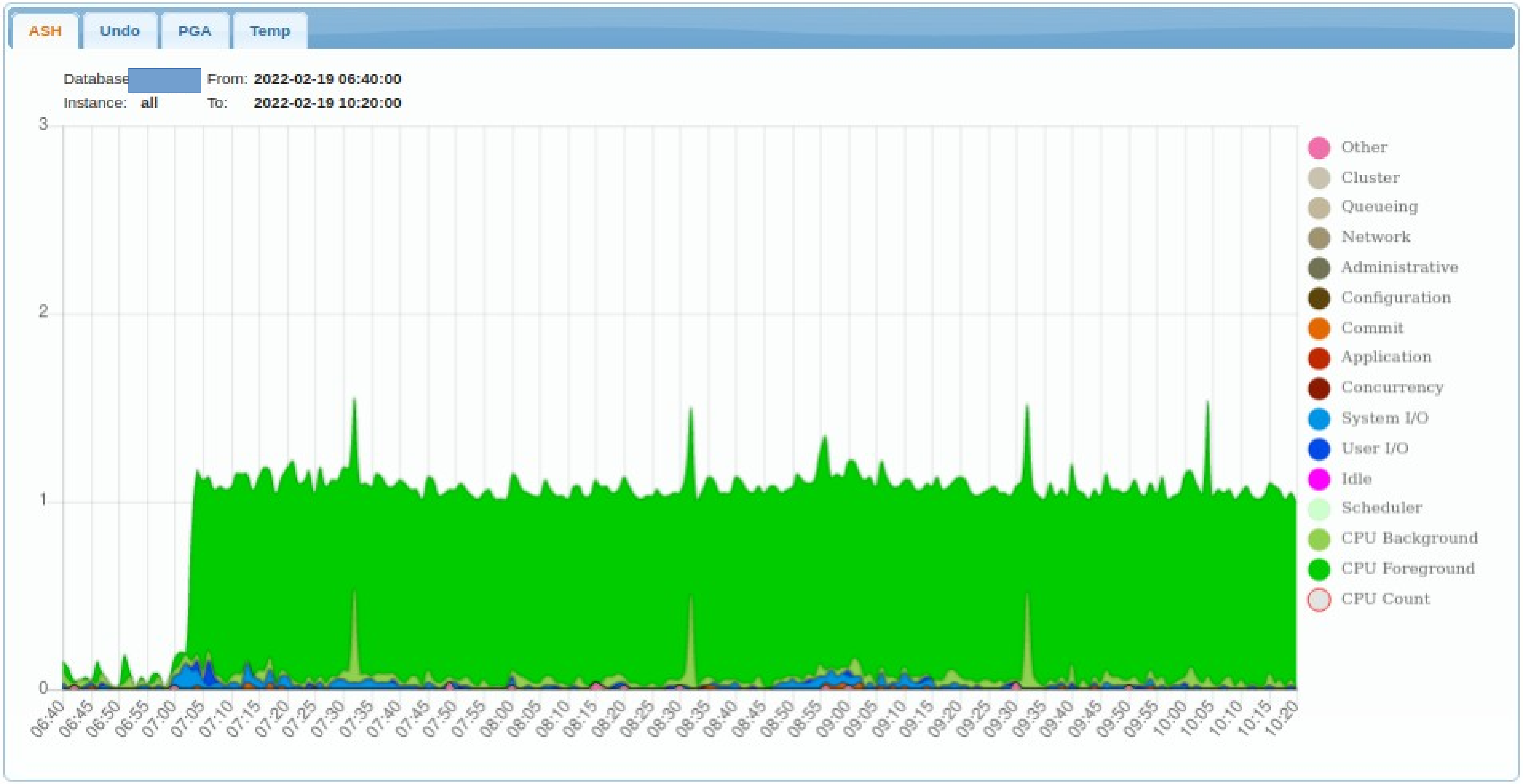
Production samples – #2



Databas[redacted] From: 2022-02-19 06:40:00

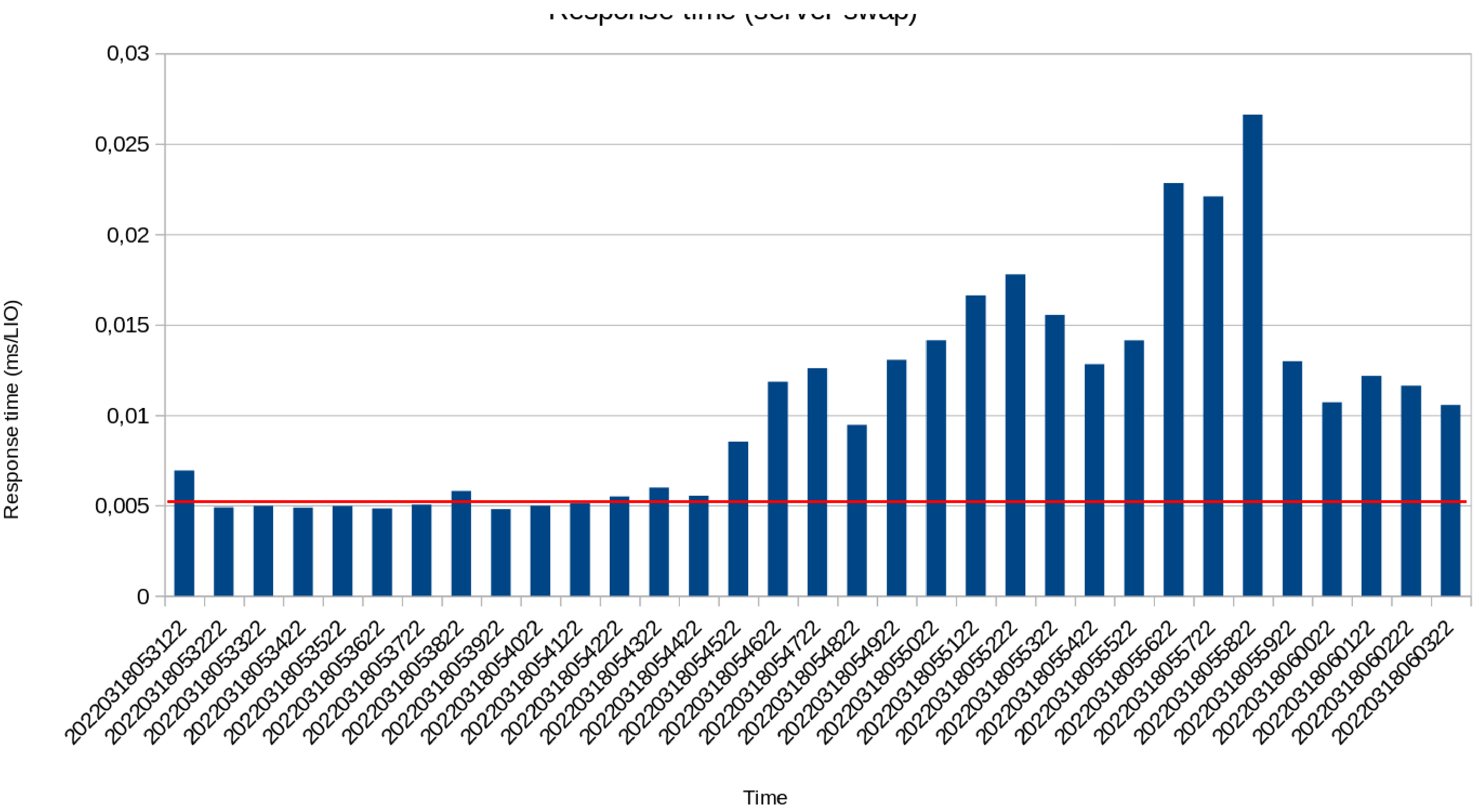
Instance: all To: 2022-02-19 10:20:00

Production samples – #2



Database [redacted] From: 2022-02-19 06:40:00
Instance: all To: 2022-02-19 10:20:00

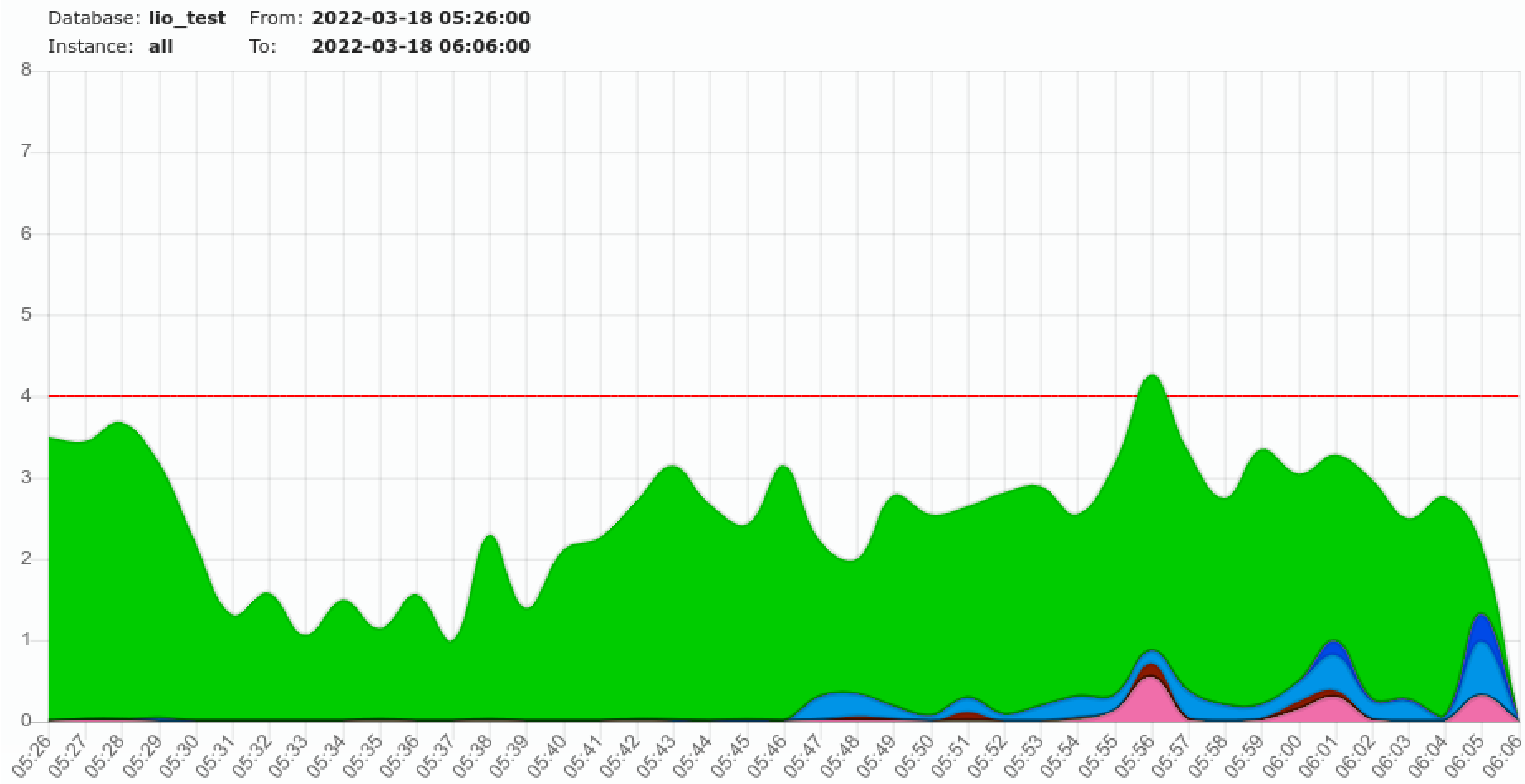
Production samples – #3



AS Abakus As na disku.

dejavu Data at your service.

Production samples – #3

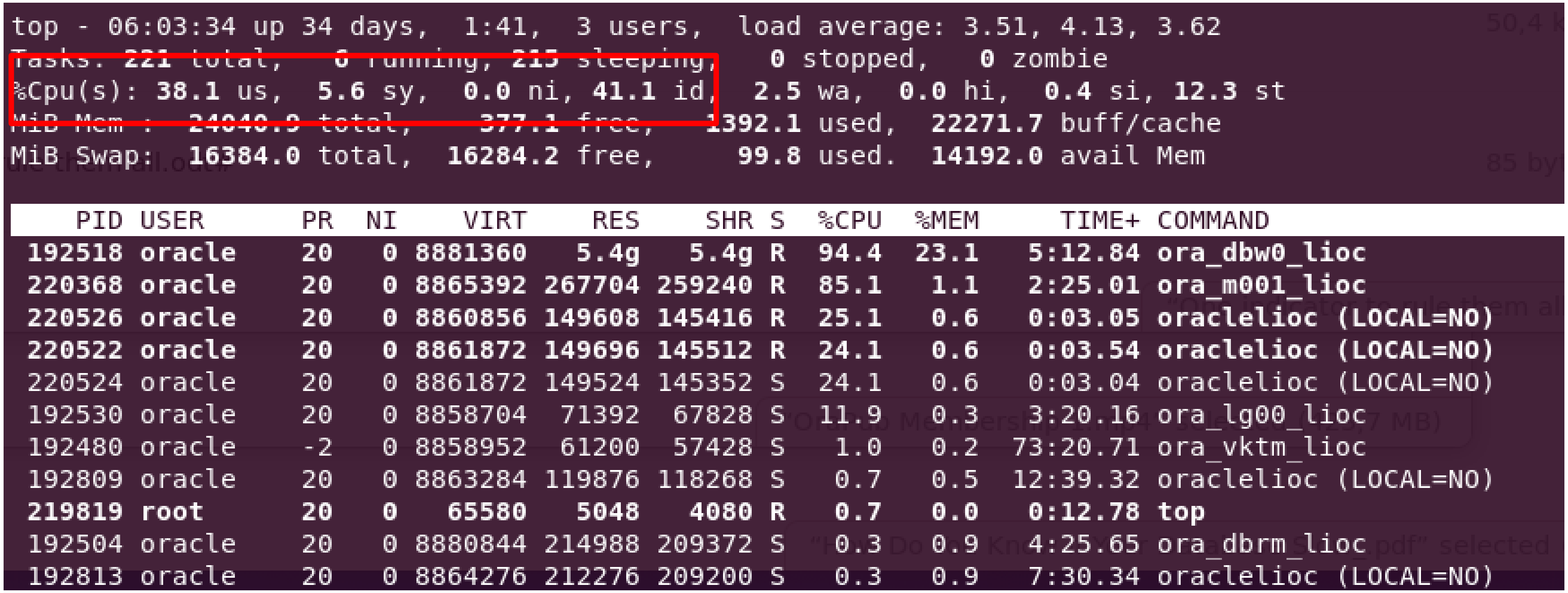


Database: **lio_test** From: **2022-03-18 05:26:00**
Instance: **all** To: **2022-03-18 06:06:00**

Abakus As na disku.

dejavu Data at your service.

Production samples – #3

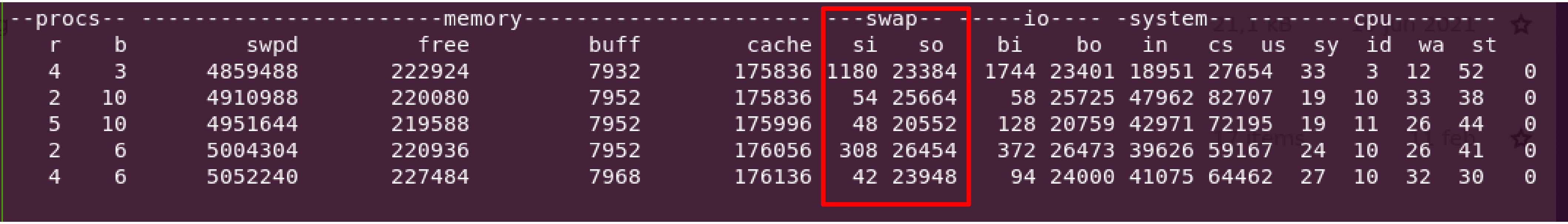


```
top - 06:03:34 up 34 days,  1:41,  3 users,  load average: 3.51, 4.13, 3.62
Tasks: 221 total,   6 running, 215 sleeping,   0 stopped,   0 zombie
%Cpu(s): 38.1 us,  5.6 sy,  0.0 ni, 41.1 id,  2.5 wa,  0.0 hi,  0.4 si, 12.3 st
MiB Mem :  24040.9 total,   377.1 free,   1392.1 used,  22271.7 buff/cache
MiB Swap:  16384.0 total,  16284.2 free,     99.8 used.  14192.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 192518 oracle    20   0 8881360   5.4g   5.4g R  94.4  23.1   5:12.84 ora_dbw0_lioc
 220368 oracle    20   0 8865392 267704 259240 R  85.1   1.1   2:25.01 ora_m001_lioc
 220526 oracle    20   0 8860856 149608 145416 R  25.1   0.6   0:03.05 oraclelioc (LOCAL=NO)
 220522 oracle    20   0 8861872 149696 145512 R  24.1   0.6   0:03.54 oraclelioc (LOCAL=NO)
 220524 oracle    20   0 8861872 149524 145352 S  24.1   0.6   0:03.04 oraclelioc (LOCAL=NO)
 192530 oracle    20   0 8858704  71392  67828 S  11.9   0.3   3:20.16 ora_lg00_lioc
 192480 oracle    -2   0 8858952  61200  57428 S   1.0   0.2  73:20.71 ora_vktm_lioc
 192809 oracle    20   0 8863284 119876 118268 S   0.7   0.5  12:39.32 oraclelioc (LOCAL=NO)
 219819 root      20   0   65580   5048   4080 R   0.7   0.0   0:12.78 top
 192504 oracle    20   0 8880844 214988 209372 S   0.3   0.9   4:25.65 ora_dbrm_lioc
 192813 oracle    20   0 8864276 212276 209200 S   0.3   0.9   7:30.34 oraclelioc (LOCAL=NO)
```

VM - top

Production samples – #3



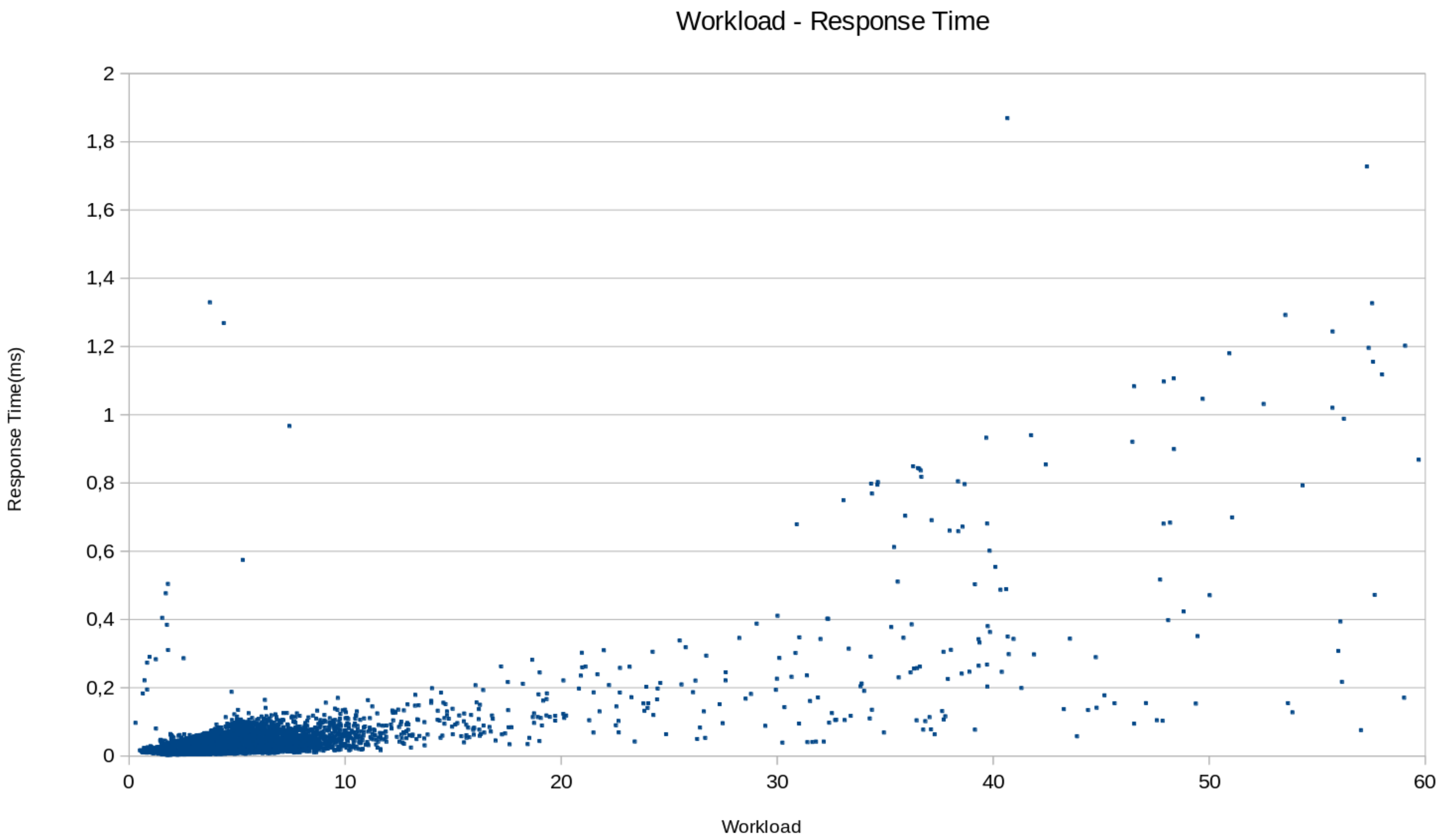
```
--procs-- -----------------------memory---------------------- ---swap-- -----io---- -system-- --------cpu--------
   r    b         swpd         free         buff        cache   si   so    bi    bo   in   cs  us  sy  id  wa  st
   4    3      4859488       222924         7932       175836 1180 23384  1744 23401 18951 27654  33   3  12  52   0
   2   10      4910988       220080         7952       175836   54 25664    58 25725 47962 82707  19  10  33  38   0
   5   10      4951644       219588         7952       175996   48 20552   128 20759 42971 72195  19  11  26  44   0
   2    6      5004304       220936         7952       176056  308 26454   372 26473 39626 59167  24  10  26  41   0
   4    6      5052240       227484         7968       176136   42 23948    94 24000 41075 64462  27  10  32  30   0
```

Server (vm hypervisor) swap

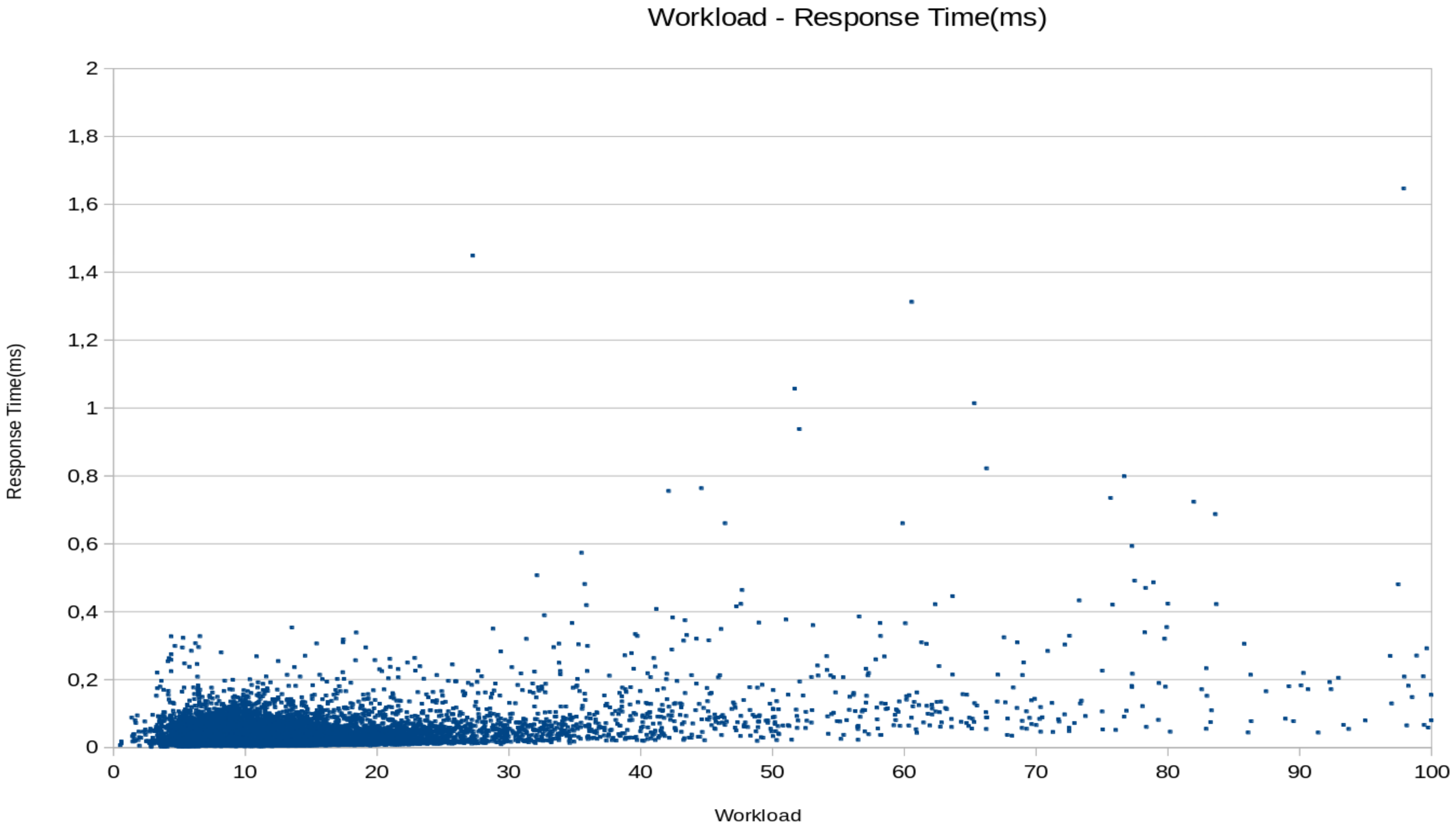
AS Abakus As na disku.

dejavu Data at your service.

Production samples – #4



Production samples – Sample #5



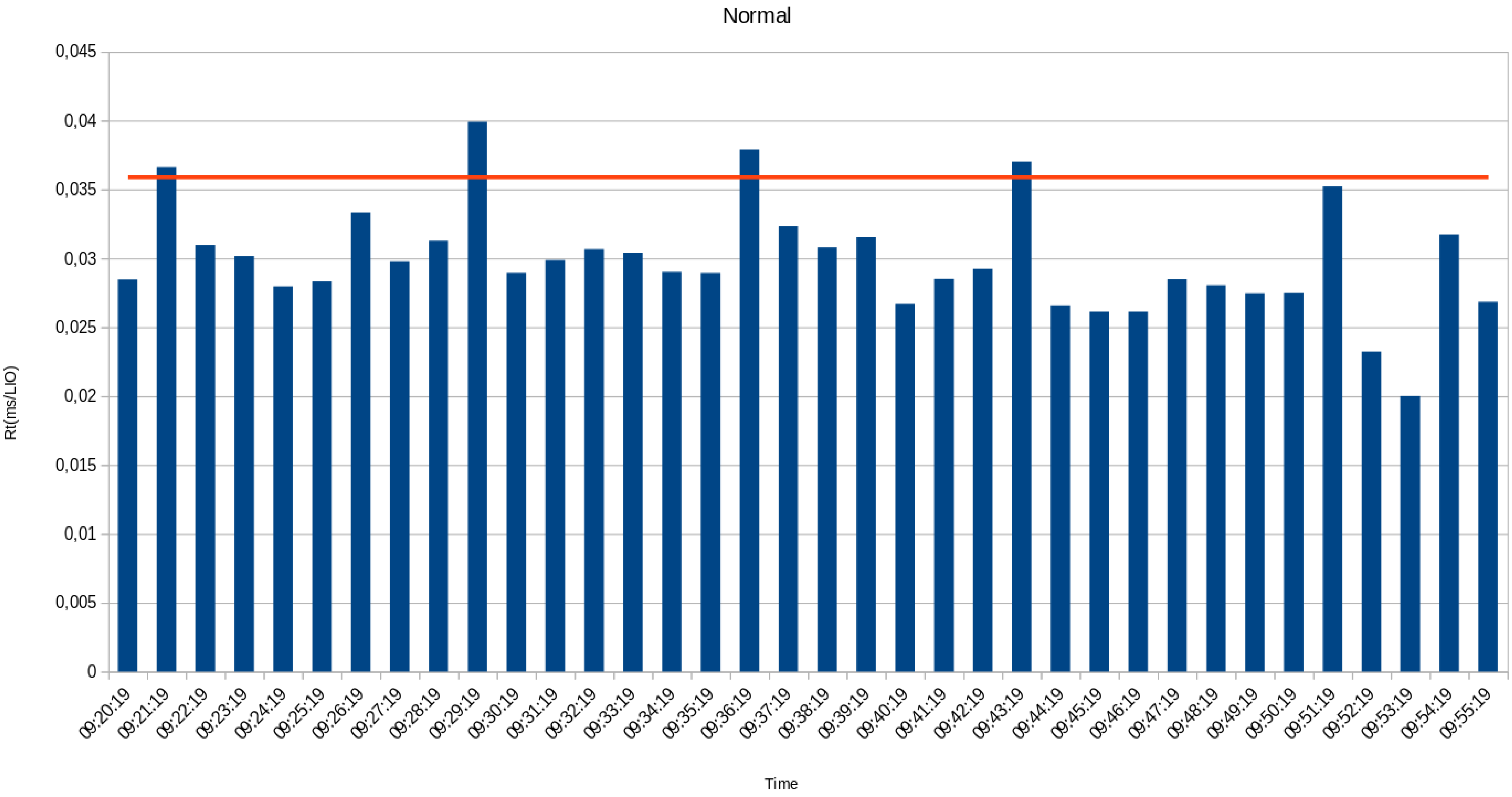
Collecting Samples

- AWR – Enterprise Edition.
- DIY sampler.
- Abakus APPM.
- 1h frequency (too infrequent).
- 10 min frequency (best experience – own tests).

Interpreting Response Time

- Rt depends on hardware.
- Baseline (when database performs well).

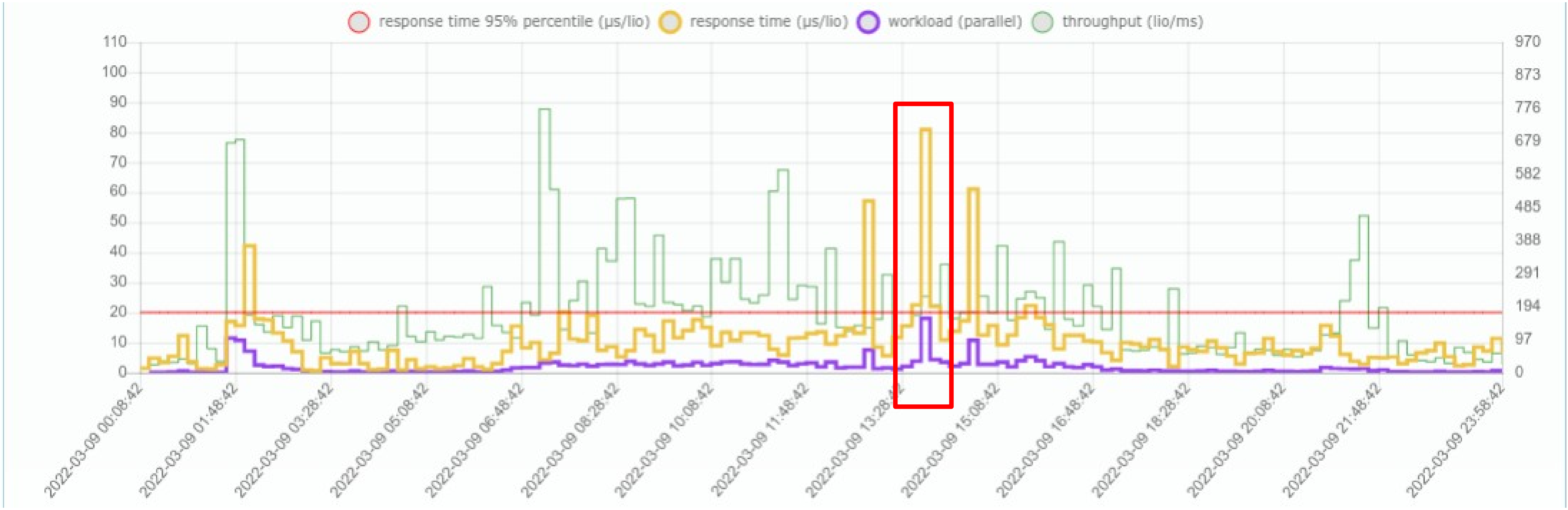
Baseline


Normal
Rt(ms/LIO)
0,045
0,04
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
09:20:19
09:21:19
09:22:19
09:23:19
09:24:19
09:25:19
09:26:19
09:27:19
09:28:19
09:29:19
09:30:19
09:31:19
09:32:19
09:33:19
09:34:19
09:35:19
09:36:19
09:37:19
09:38:19
09:39:19
09:40:19
09:41:19
09:42:19
09:43:19
09:44:19
09:45:19
09:46:19
09:47:19
09:48:19
09:49:19
09:50:19
09:51:19
09:52:19
09:53:19
09:54:19
09:55:19
Time

Monitoring

- Monitoring database activity.
- React when Rt is over baseline.
  - drill down into session:
    - – v$sess_time_model, v$sesstat
      - not in AWR.
    - – DIY samplers (on logoff triggers).
    - – Abakus APPM.

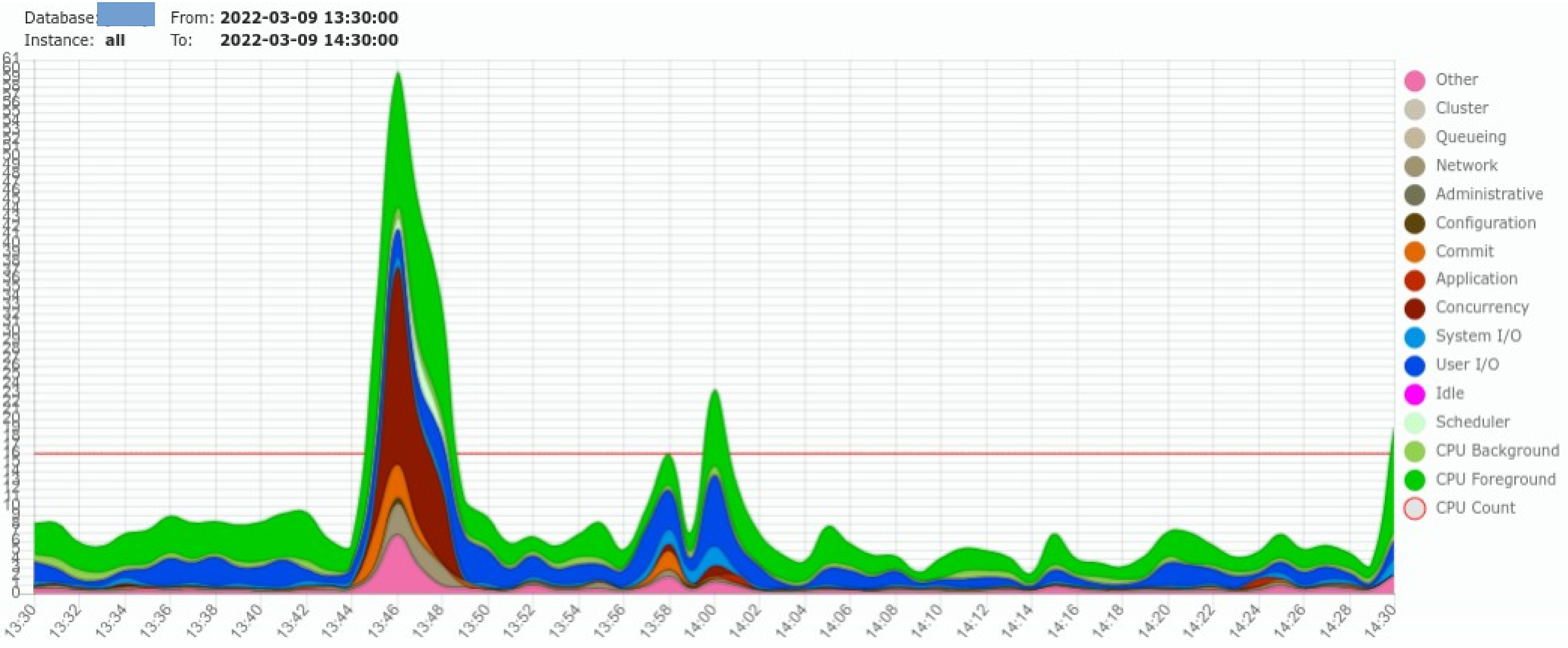
Example #1 (APPM)



Abakus As na disku.

dejavu Data at your service.

Example #1 (APPM)



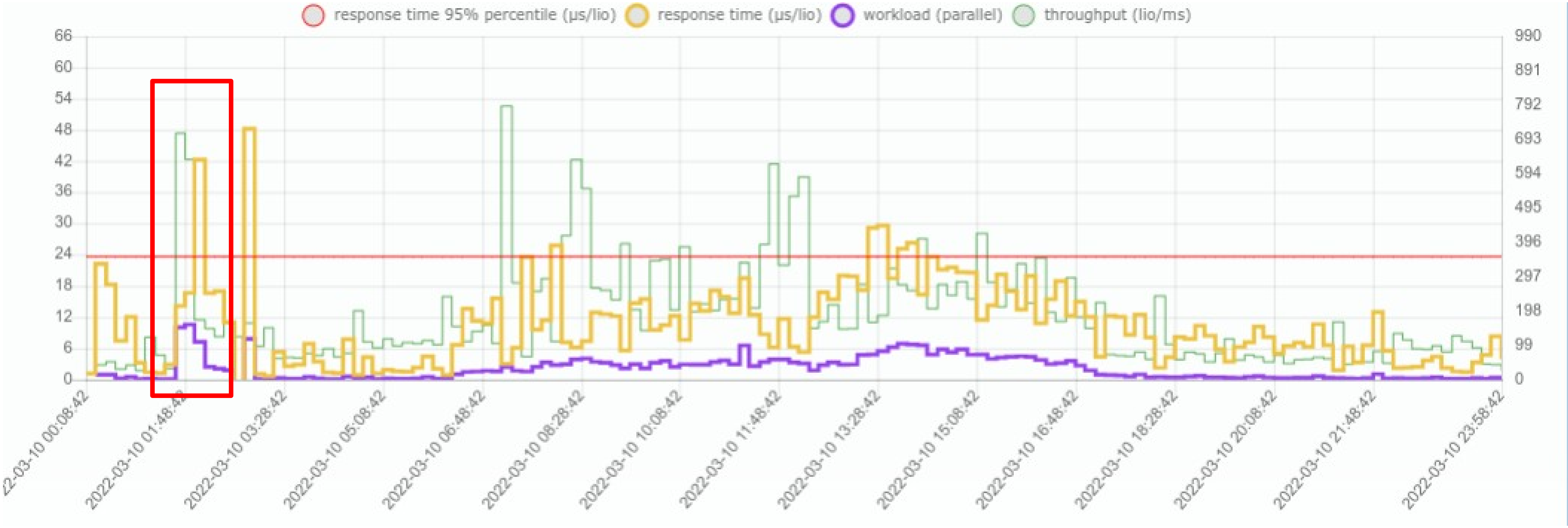
Database: [redacted] From: **2022-03-09 13:30:00**

Instance: **all** To: **2022-03-09 14:30:00**

AS Abakus As na disku.

dejavu *Data at your service.*

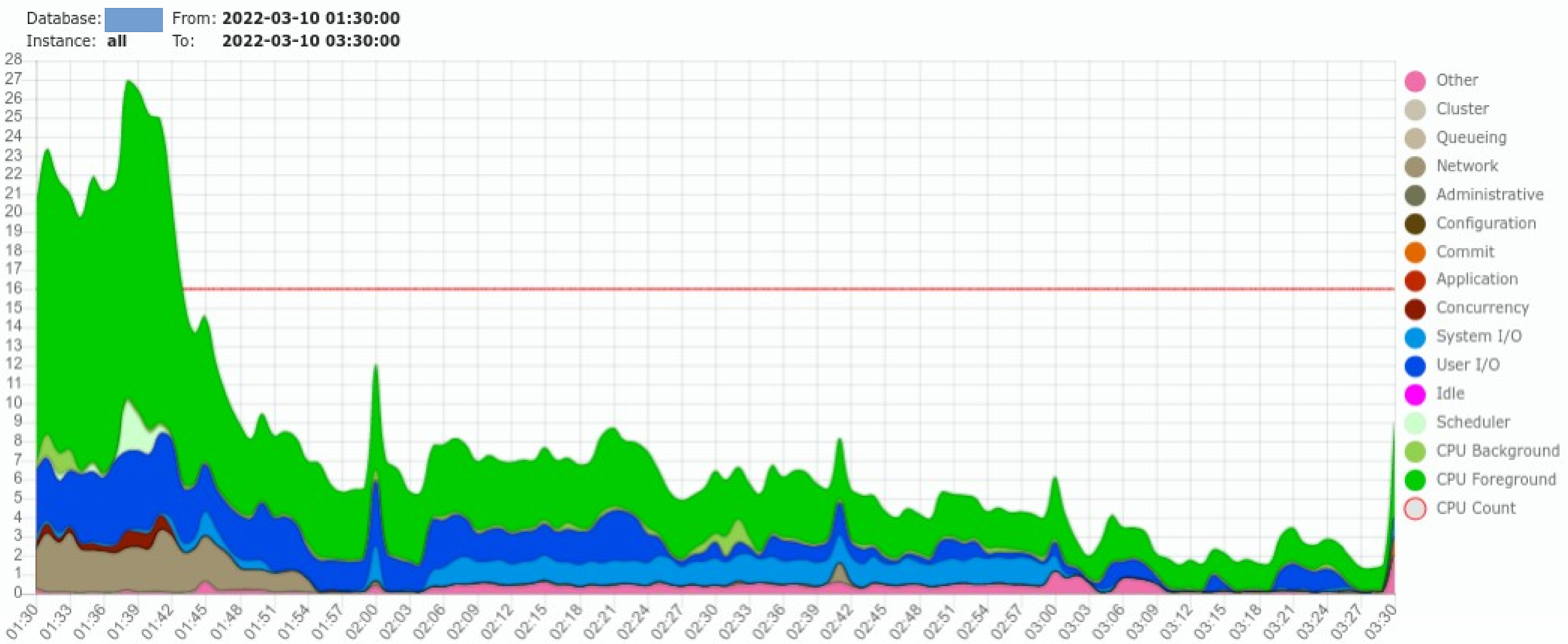
Example #2 (APPM)



AS Abakus As na disku.

dejavu Data at your service.

Example #2 (APPM)



Database: [redacted] From: **2022-03-10 01:30:00**

Instance: **all** To: **2022-03-10 03:30:00**

I’m not interested in LIO, I’m interested in the duration of an SQL statement.

Calculate execution time

- Calculate minimal execution time based on Response time:
  - clone production database.
    - (see Abakus DejaVu)
  - execute new SQL with autotrace enable.
    - (LIO = consistent gets + db block gets)
  - Min_elapsed = LIO * Rt.
    - be aware of parallelism!

Calculate execution time

```
set timing on
set autotrace traceonly

select count(*) from <TABLE> ...

Elapsed: 00:00:00.51

Statistics
----------------------------------------------------------
          0  db block gets
      28313  consistent gets
          1  rows processed
```

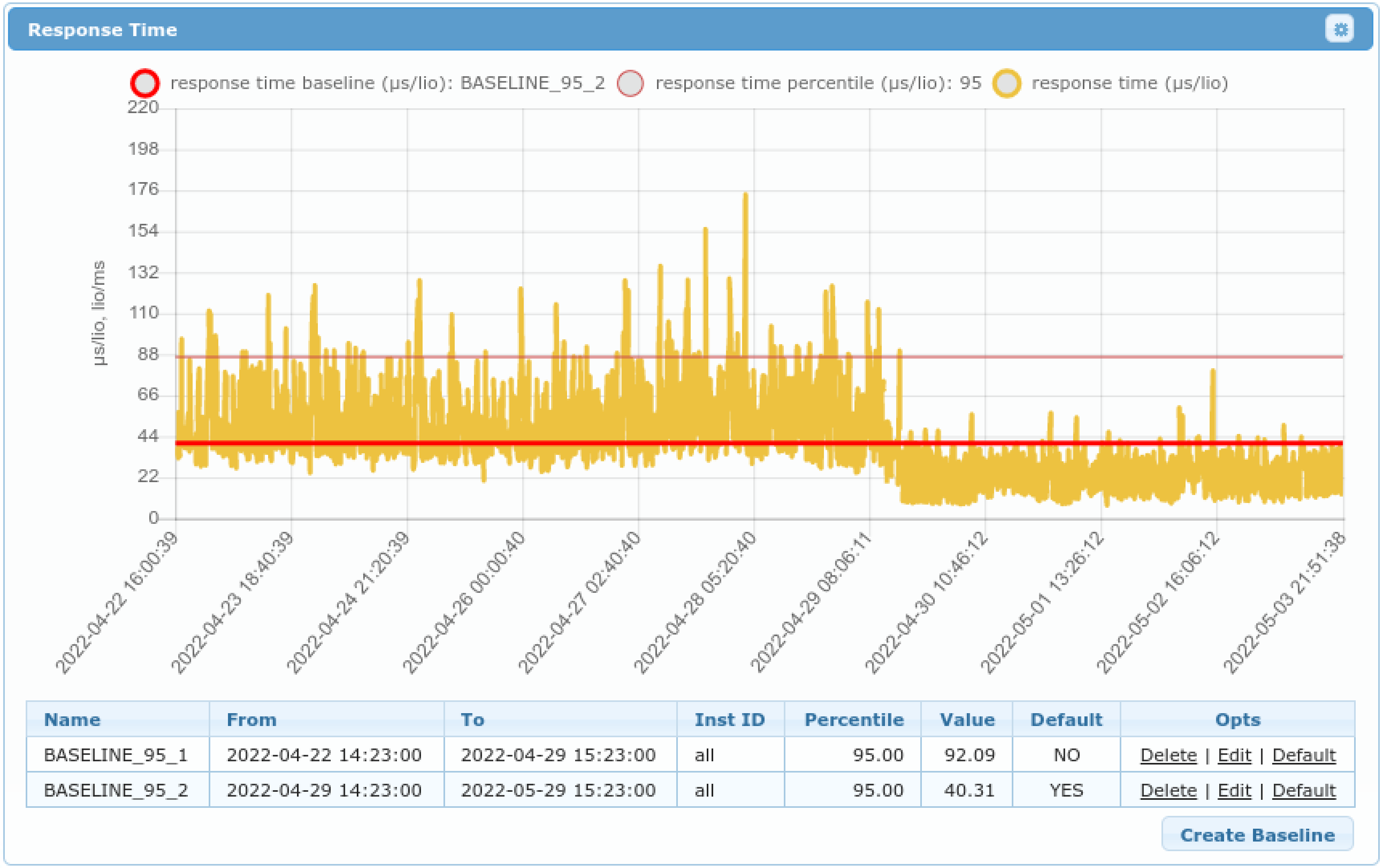
Calculate execution time

- BASELINE_95: Rt = 0,020 ms/LIO
- LIO = 28313
- Execution time = LIO * Rt / 1000
  - Execution time(95): 0,56626 sec

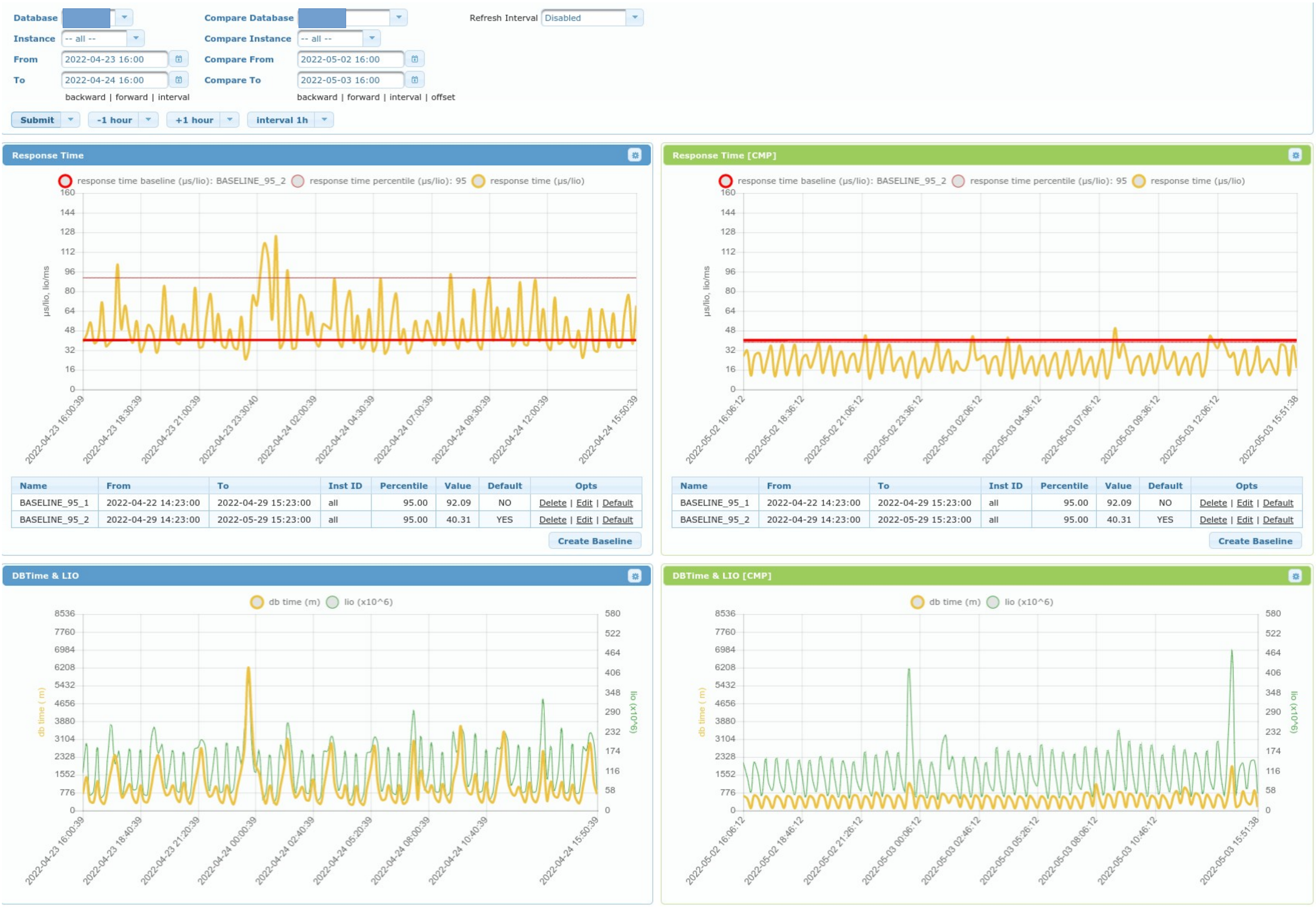
Hardware changes

- Hardware changes (CPU, RAM, ...),
- Run tests on new HW,
- Calculate (sample) Response Time,
- Compare Response Time with production.
  - Abakus APPM.

Hardware changes



Name	From	To	Inst ID	Percentile	Value	Default	Opts
BASELINE_95_1	2022-04-22 14:23:00	2022-04-29 15:23:00	all	95.00	92.09	NO	Delete | Edit | Default
BASELINE_95_2	2022-04-29 14:23:00	2022-05-29 15:23:00	all	95.00	40.31	YES	Delete | Edit | Default



Database [redacted]
Instance -- all --
From 2022-04-23 16:00
To 2022-04-24 16:00
backward | forward | interval

Compare Database [redacted]
Compare Instance -- all --
Compare From 2022-05-02 16:00
Compare To 2022-05-03 16:00
backward | forward | interval | offset

Refresh Interval Disabled

Name	From	To	Inst ID	Percentile	Value	Default	Opts
BASELINE_95_1	2022-04-22 14:23:00	2022-04-29 15:23:00	all	95.00	92.09	NO	Delete | Edit | Default
BASELINE_95_2	2022-04-29 14:23:00	2022-05-29 15:23:00	all	95.00	40.31	YES	Delete | Edit | Default

Name	From	To	Inst ID	Percentile	Value	Default	Opts
BASELINE_95_1	2022-04-22 14:23:00	2022-04-29 15:23:00	all	95.00	92.09	NO	Delete | Edit | Default
BASELINE_95_2	2022-04-29 14:23:00	2022-05-29 15:23:00	all	95.00	40.31	YES	Delete | Edit | Default

Abakus
As na disku.

dejavu *Data at your service.*

Threats

- Can be fooled?

```
SELECT COUNT(*)
  FROM sys.obj$ a
  JOIN sys.obj$ b
    ON a.owner# = b.owner#
  JOIN sys.obj$ c
    ON b.owner# = c.owner#
  JOIN sys.obj$ d
    ON c.owner# = d.owner#
 WHERE rownum <= &1;
```

One indicator to rule them all

- DB performance tracking.
- External DB and VM load awareness.
- HW change impact testing.
- Database upgrade.
- New application installed.
- More users.
- Cannot be tricked unlike “buffer cache hit ratio”.

ORA-03113: end-of-file on communication channel

?

http://www.abakus.si/

AS Abakus As na disku.

dejavu Data at your service.

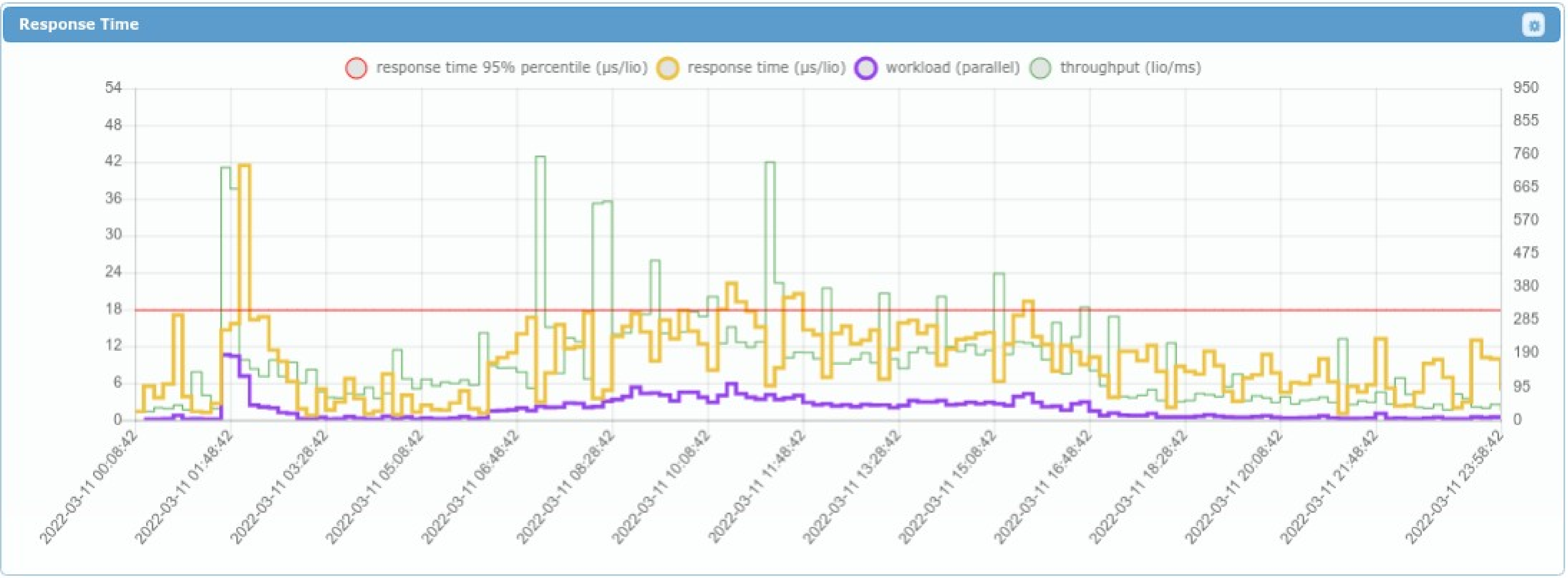
New query execution time

- POC:
  - deploy test DB as production clone (Abakus DejaVu, Snapshot Standby, ...),
  - run the query and get number of LIO (Units),
  - caluculate run time in production environment:
    - »Prod Time« = »Units« * »Response Time(PROD)«.

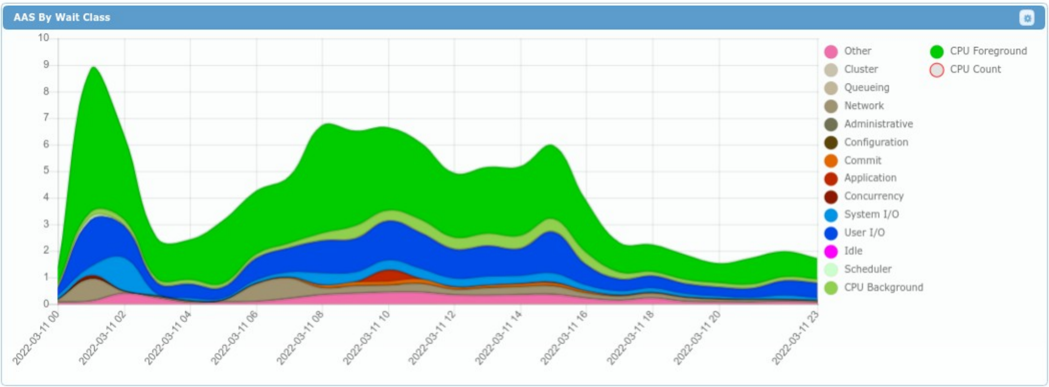
HW changes

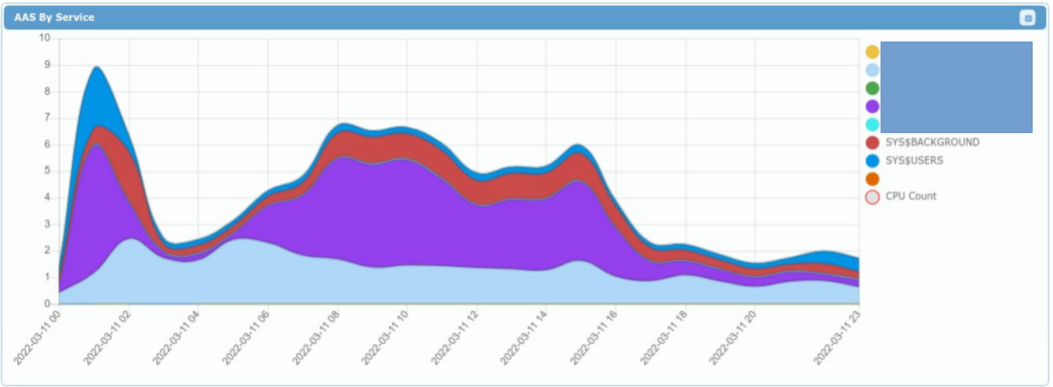
- CPU bound,
- I/O bound,
- POC:
  - calculate response time,
  - calculate new execution time:
    - »New Time« = »Units(PROD)« * »New Response Time«.

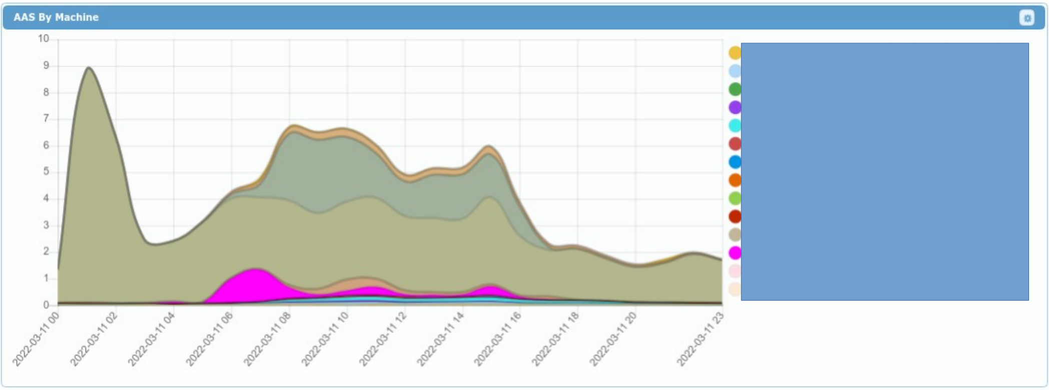
Bad, normal or good?

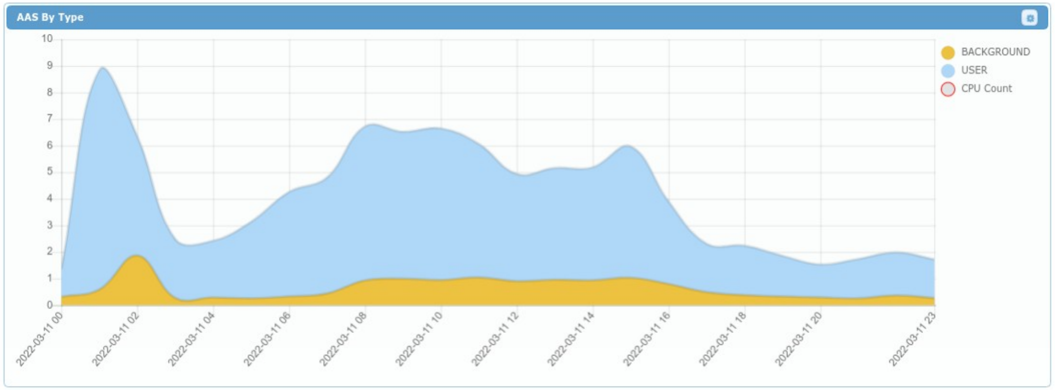


Real Response time









SQL

```
WITH snapshots AS
 (SELECT snap_id
        ,sample_time
        ,extract(hour FROM(sample_time - sample_time_prev)) * 60 * 60 +
         extract(minute FROM(sample_time - sample_time_prev)) * 60 +
         extract(SECOND FROM(sample_time - sample_time_prev)) wall_time_s
    FROM (SELECT snap_id
                ,end_interval_time sample_time
                ,lag(end_interval_time, 1) over(ORDER BY end_interval_time) sample_time_prev
            FROM dba_hist_snapshot s)
   WHERE sample_time_prev IS NOT NULL),
gtt_stats AS
 (SELECT snap_id
        ,stat_name
        ,VALUE / 1000 AS stat_value_ms
        ,lag(VALUE, 1, 0) over(ORDER BY snap_id) / 1000 AS stat_value_ms_prev
    FROM dba_hist_sys_time_model tm
   WHERE stat_name = 'DB time'
  UNION ALL
  SELECT s.snap_id
        ,s.stat_name
        ,s.value stat_value_ms
        ,lag(VALUE, 1, 0) over(ORDER BY snap_id) AS stat_value_ms_prev
    FROM dba_hist_sysstat s
   WHERE s.stat_name LIKE 'session logical reads')
SELECT snap_id
      ,sample_time
      ,wall_time_s
      ,delta_db_time_ms / (wall_time_s * 1000) workload
      ,delta_db_time_ms / 1000 delta_db_time_s
      ,delta_lio / (wall_time_s * 1000) throughput_lio_per_ms
      ,delta_db_time_ms / delta_lio response_time_ms_per_lio
  FROM (SELECT s.snap_id
              ,s.sample_time
              ,s.wall_time_s
              ,dbt.stat_value_ms - dbt.stat_value_ms_prev delta_db_time_ms
              ,dbl.stat_value_ms - dbl.stat_value_ms_prev delta_lio
          FROM snapshots s
          JOIN gtt_stats dbt
            ON dbt.snap_id = s.snap_id
           AND dbt.stat_name = 'DB time'
           AND dbt.stat_value_ms_prev != 0
          JOIN gtt_stats dbl
            ON dbl.snap_id = s.snap_id
           AND dbl.stat_name = 'session logical reads'
           AND dbl.stat_value_ms_prev != 0);
```

SQL

```
gtt_stats AS
 (SELECT snap_id
        ,stat_name
        ,VALUE / 1000 AS stat_value_ms
        ,lag(VALUE, 1, 0) over(ORDER BY snap_id) / 1000
AS stat_value_ms_prev
    FROM dba_hist_sys_time_model tm
   WHERE stat_name = 'DB time'
  UNION ALL
  SELECT s.snap_id
        ,s.stat_name
        ,s.value stat_value_ms
        ,lag(VALUE, 1, 0) over(ORDER BY snap_id) AS
stat_value_ms_prev
    FROM dba_hist_sysstat s
   WHERE s.stat_name LIKE 'session logical reads')
```

SQL

```
SELECT snap_id
      ,sample_time
      ,wall_time_s
      ,delta_db_time_ms / (wall_time_s * 1000) workload
      ,delta_db_time_ms / 1000 delta_db_time_s
      ,delta_lio / (wall_time_s * 1000) throughput_lio_per_ms
      ,delta_db_time_ms / delta_lio response_time_ms_per_lio
  FROM (SELECT s.snap_id
              ,s.sample_time
              ,s.wall_time_s
              ,dbt.stat_value_ms - dbt.stat_value_ms_prev
delta_db_time_ms
              ,dbl.stat_value_ms - dbl.stat_value_ms_prev delta_lio
          FROM snapshots s
          JOIN gtt_stats dbt
            ON dbt.snap_id = s.snap_id
           AND dbt.stat_name = 'DB time'
           AND dbt.stat_value_ms_prev != 0
          JOIN gtt_stats dbl
            ON dbl.snap_id = s.snap_id
           AND dbl.stat_name = 'session logical reads'
           AND dbl.stat_value_ms_prev != 0);
```