

# RedDatabase 4.0 in Enterprise



Berlin, 2019

# Firebird Conference 2019

Berlin, 17-19 October



# IBSurgeon



# Contents

---

- Description of the loaded system
- Tricks
- System improvements

# Description of the loaded system

---

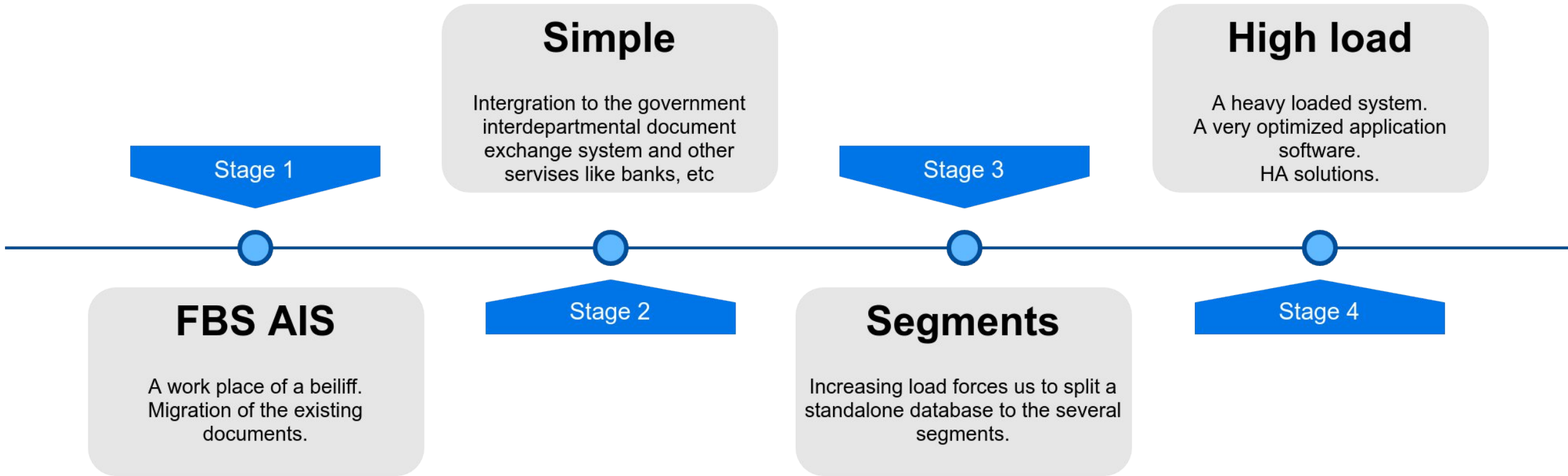
# Experience

---

- **Federal Bailiff Service (FBS)**
  - ~**3000** installations of RedDatabase. One's in every Russian city
  - **10 billions** of documents annually
  - **24/7 mode**
  - **100TB** in the main cluster of the central databases
- **Interdepartmental document exchange system of FBS (IDES)**
  - the most loaded subsystem
  - typical database size is **3 TB**
  - a database has tables
    - with **12 columns** and
    - contains up to **1.5 billion records**.
  - every day about **3 million new records** are inserted or updated
  - A query with search takes **less than 100 ms**.
  - about **2000 clients concurrently** use the database.

# Short history of IDES

---



# Requirements to IDES

---

1

Availability



2

Security



3

Performance

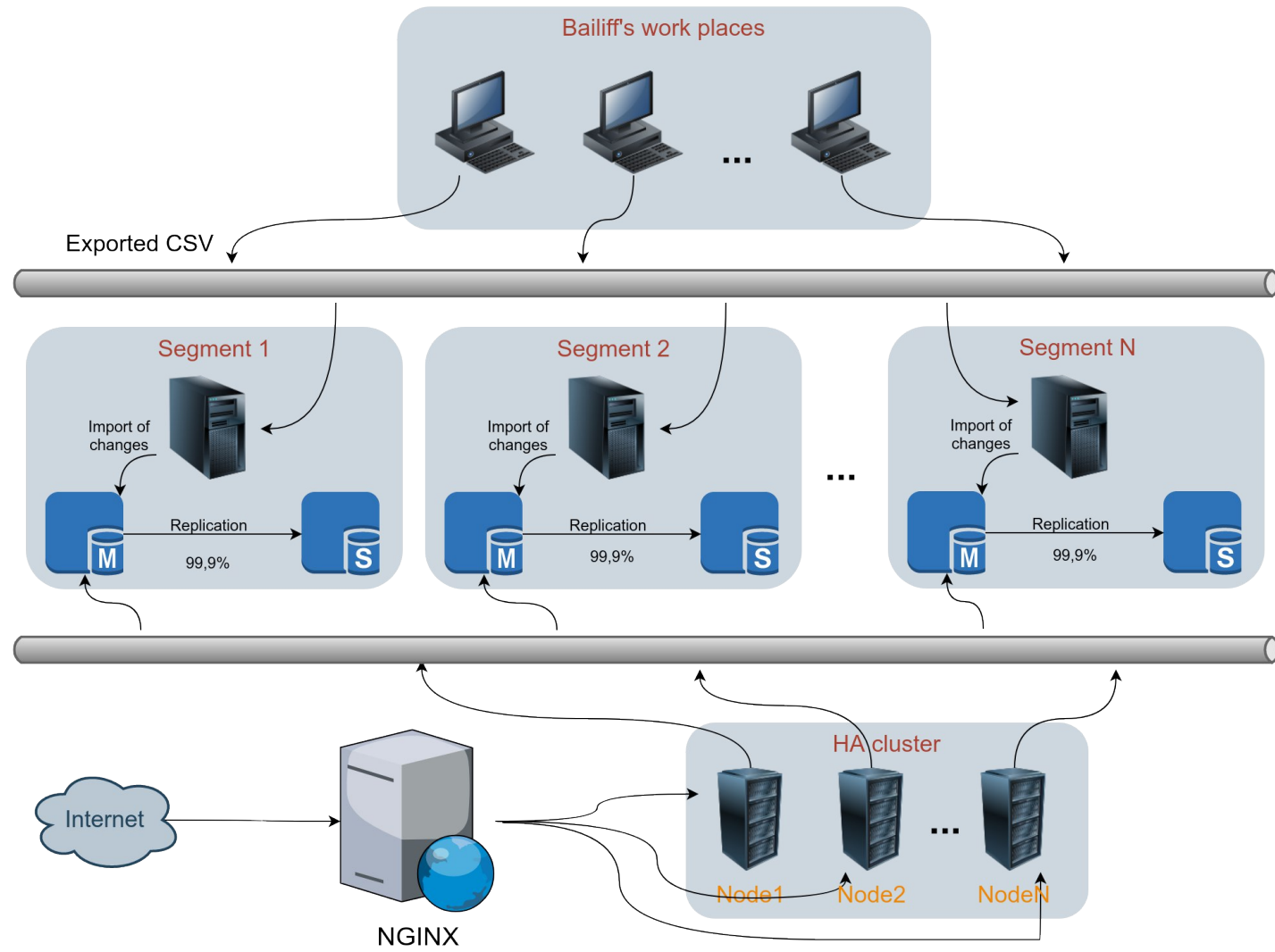


4

Reliability



# Architecture of IDES





# A role of the segments

---

- Distribute load between several databases
- Collect necessary data to localize processing
- Every database is supported independent to other segments
- There is no a single point of failure
- It's more easy to maintain

# Example of heavy loaded segment

```
roman.simakov@mvv-reestr:/opt/RedDatabase/bin
roman@roman-ubuntu: ~/prj/fb/firebird/gen/De... x  roman@roman-ubuntu: ~/prj/testgen x  roman.simakov@mvv-reestr:/opt/RedDatabase/... x +

Д 1 [||||| 86.9%] 21 [||||| 85.4%] 41 [||||| 74.8%] 61 [||||| 66.9%]
С 2 [||||| 85.7%] 22 [||||| 78.5%] 42 [||||| 82.8%] 62 [||||| 67.3%]
3 [||||| 80.5%] 23 [||||| 82.7%] 43 [||||| 67.1%] 63 [||||| 68.8%]
4 [||||| 78.6%] 24 [||||| 81.8%] 44 [||||| 64.5%] 64 [||||| 57.2%]
5 [||||| 77.4%] 25 [||||| 72.1%] 45 [||||| 67.1%] 65 [||||| 60.3%]
6 [||||| 77.4%] 26 [||||| 76.1%] 46 [||||| 63.1%] 66 [||||| 58.2%]
С 7 [||||| 74.7%] 27 [||||| 67.9%] 47 [||||| 73.2%] 67 [||||| 56.4%]
8 [||||| 75.0%] 28 [||||| 66.5%] 48 [||||| 58.9%] 68 [||||| 58.9%]
9 [||||| 72.6%] 29 [||||| 65.2%] 49 [||||| 56.2%] 69 [||||| 58.5%]
Д 10 [||||| 76.1%] 30 [||||| 83.1%] 50 [||||| 54.5%] 70 [||||| 51.3%]
11 [||||| 74.8%] 31 [||||| 84.5%] 51 [||||| 64.3%] 71 [||||| 58.9%]
12 [||||| 72.3%] 32 [||||| 70.1%] 52 [||||| 60.6%] 72 [||||| 58.0%]
И 13 [||||| 70.7%] 33 [||||| 67.9%] 53 [||||| 43.9%] 73 [||||| 50.3%]
14 [||||| 66.5%] 34 [||||| 100.0%] 54 [||||| 51.9%] 74 [||||| 39.0%]
15 [||||| 63.9%] 35 [||||| 71.8%] 55 [||||| 48.1%] 75 [||||| 42.1%]
р 16 [||||| 54.8%] 36 [||||| 60.6%] 56 [||||| 42.3%] 76 [||||| 38.1%]
17 [||||| 75.9%] 37 [||||| 57.7%] 57 [||||| 34.2%] 77 [||||| 39.0%]
18 [||||| 100.0%] 38 [||||| 47.7%] 58 [||||| 35.0%] 78 [||||| 29.6%]
3 19 [||||| 50.0%] 39 [||||| 100.0%] 59 [||||| 40.1%] 79 [||||| 7.0%]
20 [||||| 69.9%] 40 [||||| 79.0%] 60 [||||| 44.6%] 80 [||||| 15.3%]
Mem [||||| 105974/516241MB]
3 Swp [||||| 263/524287MB]
Tasks: 143, 958 thr; 6 running
Load average: 37.24 36.27 32.15
Uptime: 179 days(!), 21:40:27

Учетная запись для сайта и мобильного приложения Аэроэкспресс/Account for the Aeroexpress...
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
54964 root 20 0 593M 292M 73152 R 101.0 0.1 1h02:27 /opt/RedDatabase/bin/rdb_inet_server
70106 root 20 0 468M 167M 73040 S 101.0 0.0 14h13:52 /opt/RedDatabase/bin/rdb_inet_server
60060 root 20 0 468M 167M 73040 R 101.0 0.0 8:46.51 /opt/RedDatabase/bin/rdb_inet_server
70455 root 20 0 593M 292M 73152 S 100.0 0.1 16h35:09 /opt/RedDatabase/bin/rdb_inet_server
60918 root 20 0 455M 154M 73164 R 100.0 0.0 0:32.89 /opt/RedDatabase/bin/rdb_inet_server
70475 root 20 0 455M 154M 73164 S 100.0 0.0 13h14:24 /opt/RedDatabase/bin/rdb_inet_server
70294 root 20 0 505M 172M 73096 S 82.0 0.0 14h34:29 /opt/RedDatabase/bin/rdb_inet_server
60600 root 20 0 505M 172M 73096 R 82.0 0.0 2:50.87 /opt/RedDatabase/bin/rdb_inet_server
37613 root 20 0 427M 104M 48744 S 25.0 0.0 5:59.43 /opt/RedDatabase/bin/rdb_inet_server
60167 root 20 0 427M 104M 48744 S 24.0 0.0 0:40.32 /opt/RedDatabase/bin/rdb_inet_server
69466 t-mvv 20 0 61.8G 11.0G 20436 S 13.0 2.2 43h07:06 /usr/java/default/jre/bin/java -Djava.util.logging.config.file=/mvv/tom
53646 root 20 0 406M 85496 49316 S 8.0 0.0 1:42.51 /opt/RedDatabase/bin/rdb_inet_server
41303 root 20 0 324M 67232 48728 S 8.0 0.0 4:01.25 /opt/RedDatabase/bin/rdb_inet_server
60780 root 20 0 324M 67232 48728 S 8.0 0.0 0:06.46 /opt/RedDatabase/bin/rdb_inet_server
60694 root 20 0 406M 85496 49316 R 6.0 0.0 0:04.28 /opt/RedDatabase/bin/rdb_inet_server
60982 roman.sim 20 0 111M 3148 1272 R 5.0 0.0 0:00.82 htop
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
```

# Tricks

---

# General tuning

---

- LCX containers
  - The fastest path between the engine and disk
- tmpfs - a special file system in Linux for temporary files
  - it's suitable for temporary files used in the sorting operations
  - can use swap if necessary
  - useful for big RAM volumes
- transparent hugepage
  - Slowdown memory allocation and usage
  - DB allocates small pages
  - It's better to disable
    - > echo 'madvise'> /sys/kernel/mm/redhat\_transparent\_hugepage/defrag
    - > echo 'madvise'> /sys/kernel/mm/redhat\_transparent\_hugepage/enabled

# Limits

---

- Max process:
  - On RHEL/CentOS - 1024 by default
  - For classic -  $1024 / 5$  threads = ~200 connections
- Max open files:
  - On RHEL/CentOS - 1024/4096
  - Every classic process open ~20 files
  - Limit is ~200 connections
- To extend:
  - Carefully edit `/etc/security/limits.conf`
  - Can be changed online
  - OOM is much more destructive

# Binding processes to physical CPU

---

## Problems:

- OS can move a process to another NUMA nodes
- The process memory migrates too
- Only for very intensive modifying (not for selects!!!)

## Solution 1:

- To tune an OS process scheduler CFS:
  - **sched\_min\_granularity\_ns**
  - **sched\_wakeup\_granularity\_ns**
- It didn't work for us

## Solution 2:

- **To bind** RedDatabase processes to a physical CPU
- It works fine

# How to find a problem with NUMA

---

- System time >> User time
- High **iowait** (uninterruptible sleep) in top **without** queues to I/O device
- Very fragmented memory (hit and miss are of **the same order of magnitude**)
- It's necessary to analyze **every thread** but not the process

# Example (numastat)

---

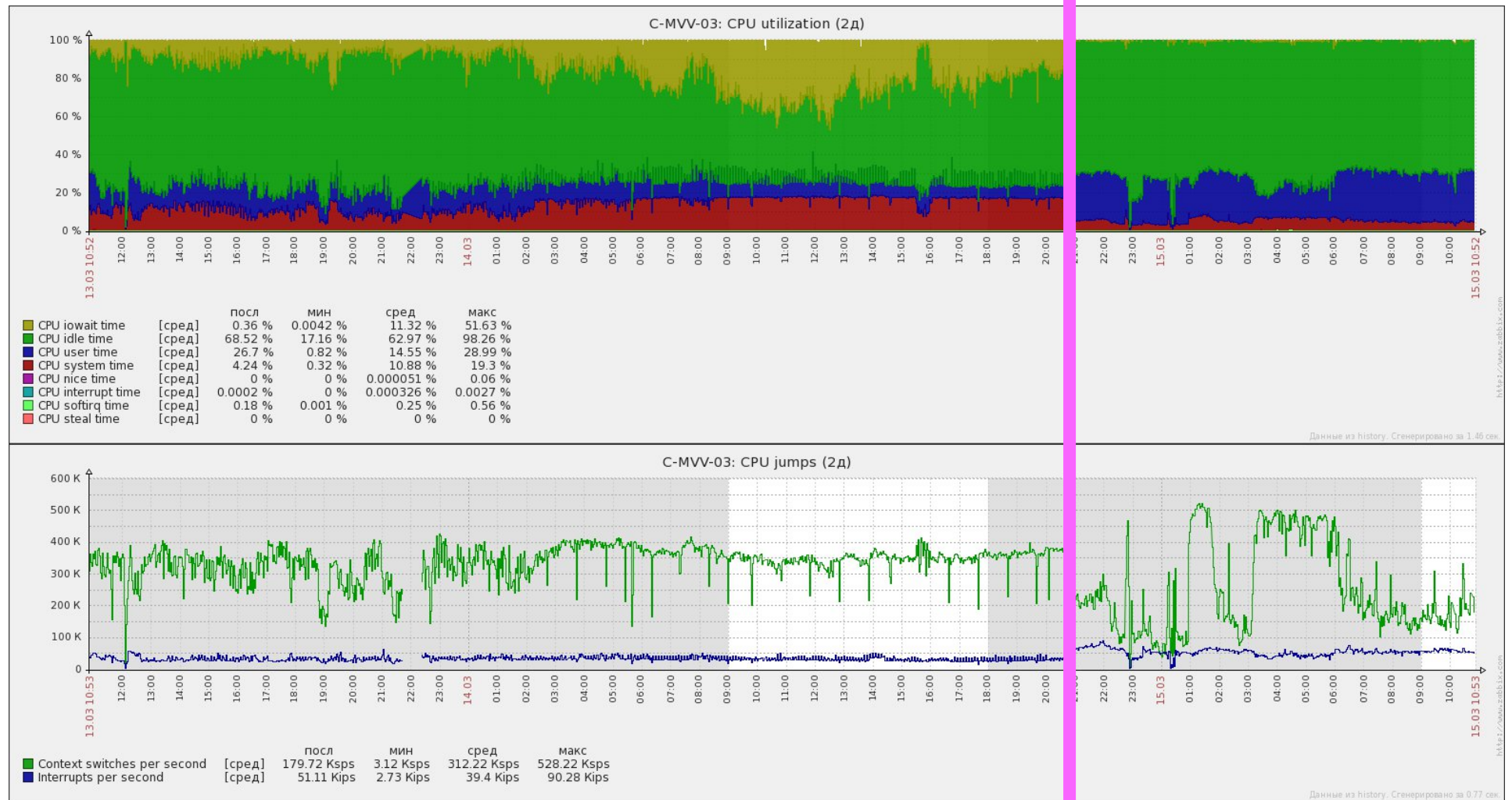
```
# numastat
                node0          node1          node2          node3
numa_hit        35483991701    25907973778    43439858254    23399971622
numa_miss       13235884421    16606874744    16180699231    18803917332
-----
numa_foreign    17220754371    11119698073    24867593917    11619329374
interleave_hit          53690          53720          53709          53739
local_node      35483981570    25907909025    43439790189    23399907916
other_node      13235894552    16606939497    16180767296    18803981038
```

```
# numastat
                node0          node1          node2          node3
numa_hit        224300115716    293940768830    289461636335    296555251853
numa_miss       516913107       380035968       451756983       486206474
-----
numa_foreign    371950817       457785062       488683152       516493501
interleave_hit          53615          53592          53603          53624
local_node      224299280174    293940256731    289461093452    296554599720
other_node      517748649       380548067       452299866       486858607
```



# Example (graphics)

- System time is **less**
- User time is **more**
- iowait time almost **absent**



# Fast backup/restore. Method.

---

- Run tmux and open 2 sessions
- In the first run:

```
gbak -v -y backup.log -g -b db stdout | lzop -1 -o db.fbk.lzo
```

- In the second run:

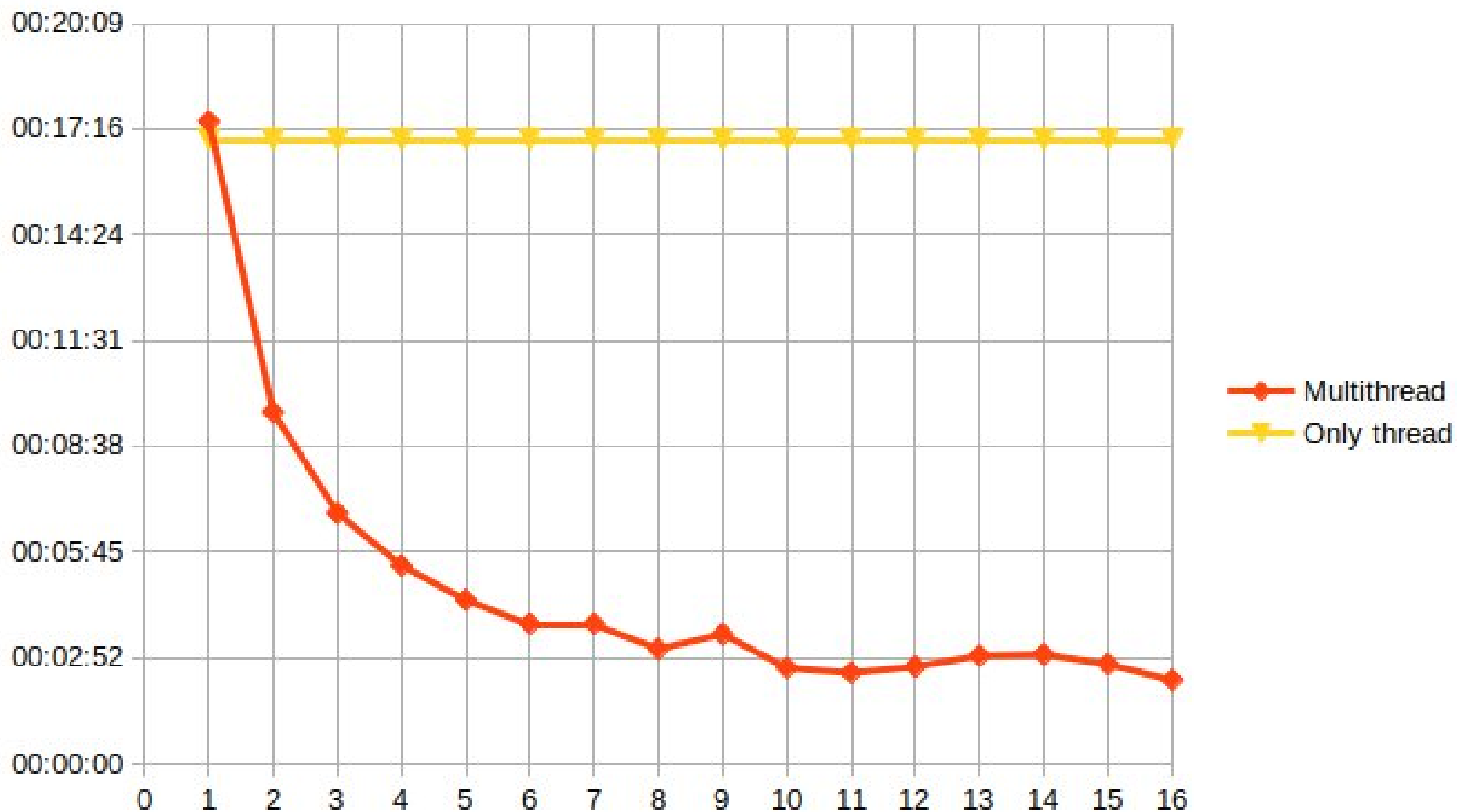
```
tail -n+0 -f db.fbk.lzo | lzop -dc | gbak -v -y restore.log -o -i -r stdin db-br
```

- Watch the end of process via restore.log and kill the second command
- Use **plum** tool ([github.com/NeoZX/plume](https://github.com/NeoZX/plume)) to activate indecies concurrently

```
plume -u sysdba -p masterkey -t <N> -d localhost:db-br
```

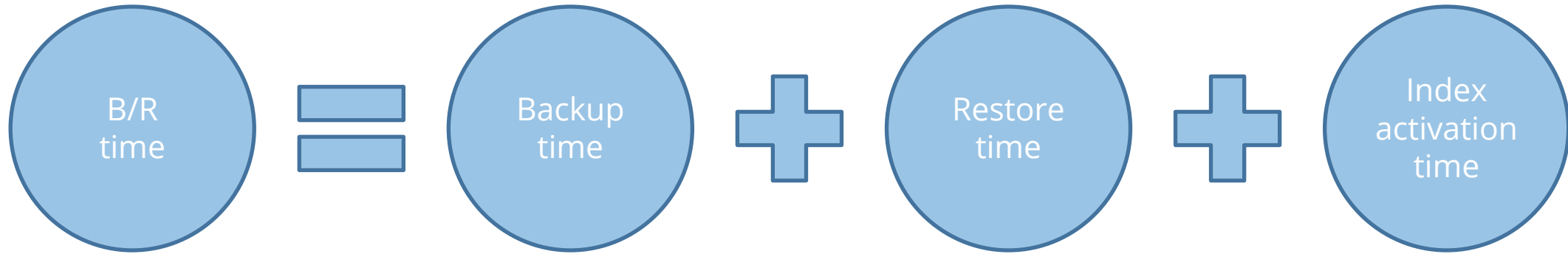
- N - a number of threads
- -ig, -n, etc can be used like gbak

# Fast backup/restore. Results.



# Fast backup/restore. Conclusions.

---



**10x** faster index activation  
**3x** faster B/R

# gbak/nbackup via fifo

---

- Enterprise backup tools like bacula (<https://www.bacula.org/>)

- GBAK

```
>mkfifo -m 0660 /tmp/database.fbk && chgrp firebird /tmp/database.fbk && \  
gbak -g -b localhost:database /tmp/database.fbk && rm /tmp/database.fbk
```

- NBACKUP

```
>mkfifo -m 0660 /tmp/database.${LEVEL}.nbk &&  
chgrp firebird /tmp/database.${LEVEL}.nbk && \  
nbackup -b ${LEVEL} localhost:database /tmp/database.level.nbk && \  
rm /tmp/database.gbk
```

- Now bacula can backup /tmp/database.\*

# System improvements

---

# Snapshot request & Garbage collection

---

- Patch
  - Originally implemented by Nikolay Samofatov (Red Soft)
  - Improved and merged by Vlad Khorsun (Firebird Foundation)
- Keep only necessary record versions
- Run request in snapshot

More details in Vlad's talks

<https://firebirdsql.org/file/community/conference-2016/statement-level-read-consistency.pdf>

**New transactions' features and changes in garbage collection in Firebird 4**

# Improvements in restore via GBAK

---

- Errors in Restore:
  - is bad (many hours and oopps)
  - can be postfixed
  - must be warnings
  - like indicies
- How it works:
  - RDB\$PROCEDURE\_BLR is set to NULL for failed objects
  - It keeps dependencies
  - Later the failed should be recreated



# Improvements in restore via GBAK. Example.

```
create procedure P1_ROUND (NUM float)
returns (RES float)
as
begin
  RES = round(NUM);
  suspend;
end^

create procedure P2
as
  declare variable VAR float;
begin
  VAR = (select RES from P1_ROUND(4.4564));
end^

alter procedure P1_ROUND (NUM float, SCALE
integer)
returns (RES float)
as
begin
  RES = round(NUM, SCALE);
  suspend;
end^
```

```
gbak:restoring stored procedure P1_ROUND
...
gbak:restoring stored procedure P2
...
gbak:committing metadata
gbak: ERROR:invalid request BLR at offset 50
gbak: ERROR:      Input parameter mismatch for procedure P1_ROUND
...
gbak:Exiting before completion due to errors
gbak:stopped at Thu Oct  3 11:00:11 2019
```

```
gbak:restoring stored procedure P1_ROUND
...
gbak:restoring stored procedure P2
...
gbak:committing metadata
gbak: ERROR:Error while parsing procedure P2's BLR
gbak: ERROR:      invalid request BLR at offset 50
gbak: ERROR:      Input parameter mismatch for procedure P1_ROUND
...
gbak: WARNING:Database is not online due to failure to restore one or
more objects.
gbak: WARNING:Run gfix -online to bring database online.
gbak:stopped at Tue Oct  8 16:48:00 2019
```

# Improvements in restore via GBAK. Fixing.

---

```
select RDB$PROCEDURE_SOURCE from RDB$PROCEDURES where RDB$PROCEDURE_BLR is NULL;  
select RDB$FUNCTION_SOURCE from RDB$FUNCTIONS where RDB$FUNCTION_BLR is NULL;  
select RDB$TRIGGER_SOURCE from RDB$TRIGGERS where RDB$TRIGGER_BLR is NULL;  
...  
ALTER PROCEDURE/FUNCTION/TRIGGER ...
```

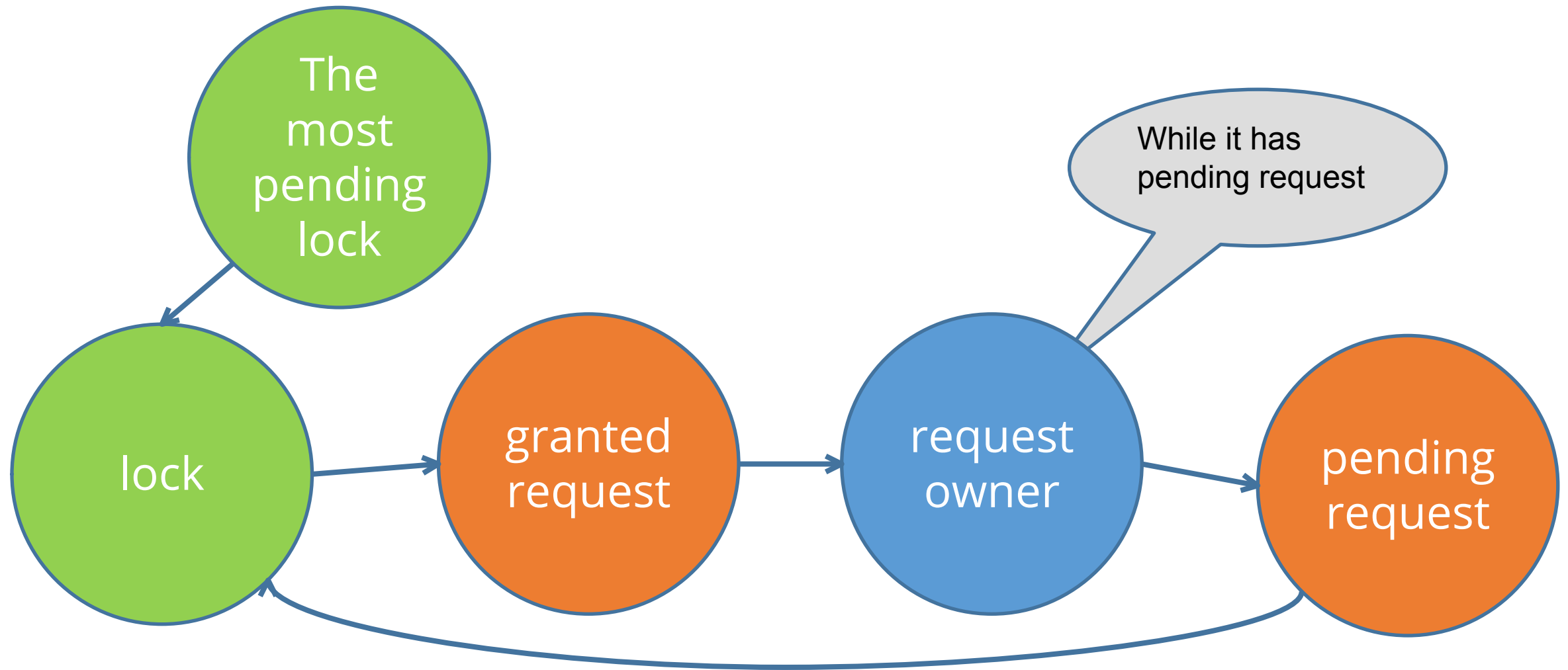
# Improvements in fb\_lock\_print

---

- fb\_lock\_print -q <seconds>
  - find potential dead owner and request
  - wait the number of seconds
  - find potential dead owner and request again
  - equal requests very often mean the owner is hang
  - gdb helps.

# Dead owner search algorithm

---



# Binary trace file

---

- All events
- The fastest way
- Low overhead

- Big file

# Binary trace file. Usage.

---

- `fbtrace.conf`

database

```
{  
    # Do we trace database events or not  
    enabled = 1  
    # Log file format (0 - text, 1 - binary, 2 - system)  
    format = 1  
    ...
```

- `In database`

```
CREATE TABLE <table_name> EXTERNAL [FILE] <filespec> ADAPTER 'fbtrace'  
[(<col_defs>)]
```

- If `col_defs` are absent a default set will be used

# Binary trace file. Example.

---

- Select every SQL statement looking for PREPARE STATEMENT event type
- Join with every FINISH STATEMENT on statementID
- Filter only queries longer then 20 ms.
- Aggregate

```
SELECT LLL.SQL, SUM(L.PERF_TIME), AVG(L.PERF_TIME), COUNT(L.PERF_TIME)
FROM log L JOIN
    (SELECT cast (LL.STMT_SQL as varchar(32000)) as SQL, LL.STMT_ID FROM log LL
     WHERE LL.EVENT_TYPE = 'PREPARE STATEMENT') LLL
  ON LLL.STMT_ID = L.STMT_ID
WHERE L.EVENT_TYPE = 'FINISH EXECUTE STATEMENT' and L.PERF_TIME > 20
GROUP BY 1
ORDER BY 2 desc
```

# Binary trace file. Result.

SQL	SUM	AVG	COUNT
<code>select sum(pl.delta_amt) amt, pl.submit_date carry_date, doc.doc_date, doc.doc_number, doc.opertype_id,</code>	549207	61023	9
<code>select sum(bh.day_credit) as Financing from expbudget exp join budgrest br on (br.budgetline_id=exp.budgetline_id) join account a on</code>	443561	62	7042
<code>select sum(bh.day_credit) as Financing from expbudget exp join budgrest br on (br.budgetline_id=exp.budgetline_id) join account a on</code>	427189	71198	1



# Load in a laboratory - big challenge

---

- Record API calls
- Emulate load profile based on trace
- Load applications which uses a database
- **Your ideas?**

# Questions?

---

[www.red-soft.ru](http://www.red-soft.ru)

[www.reddatabase.ru](http://www.reddatabase.ru)

