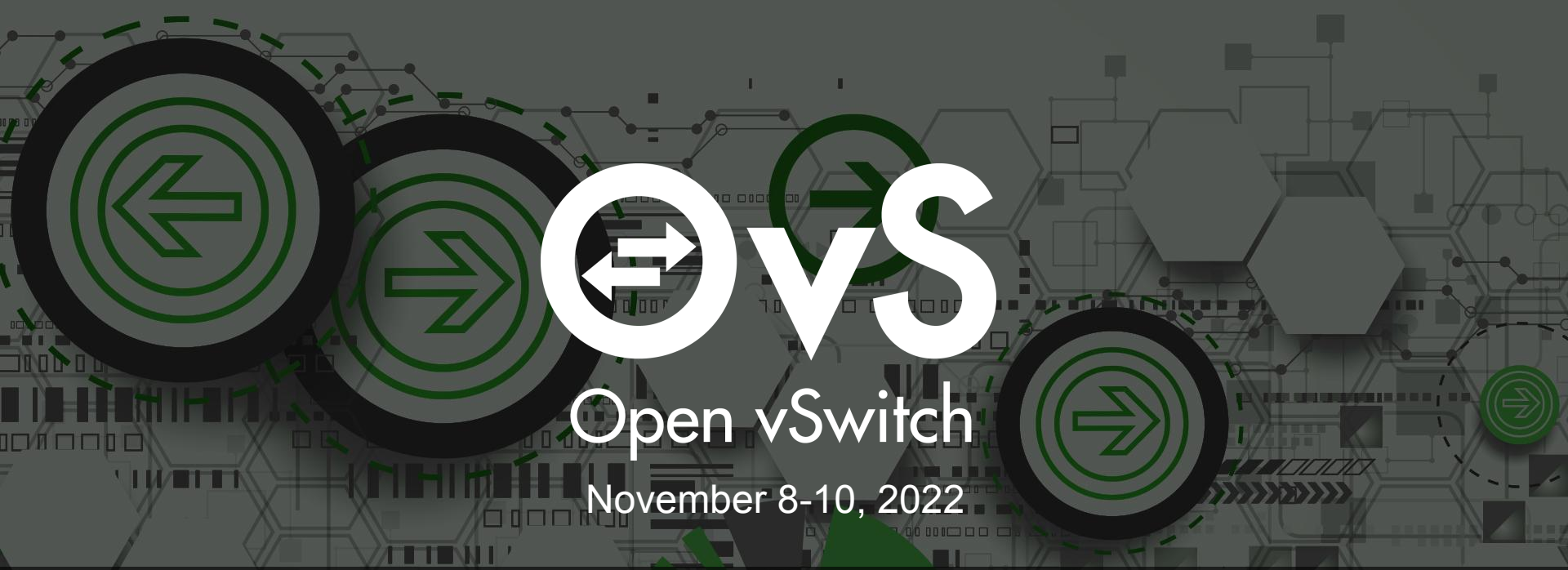
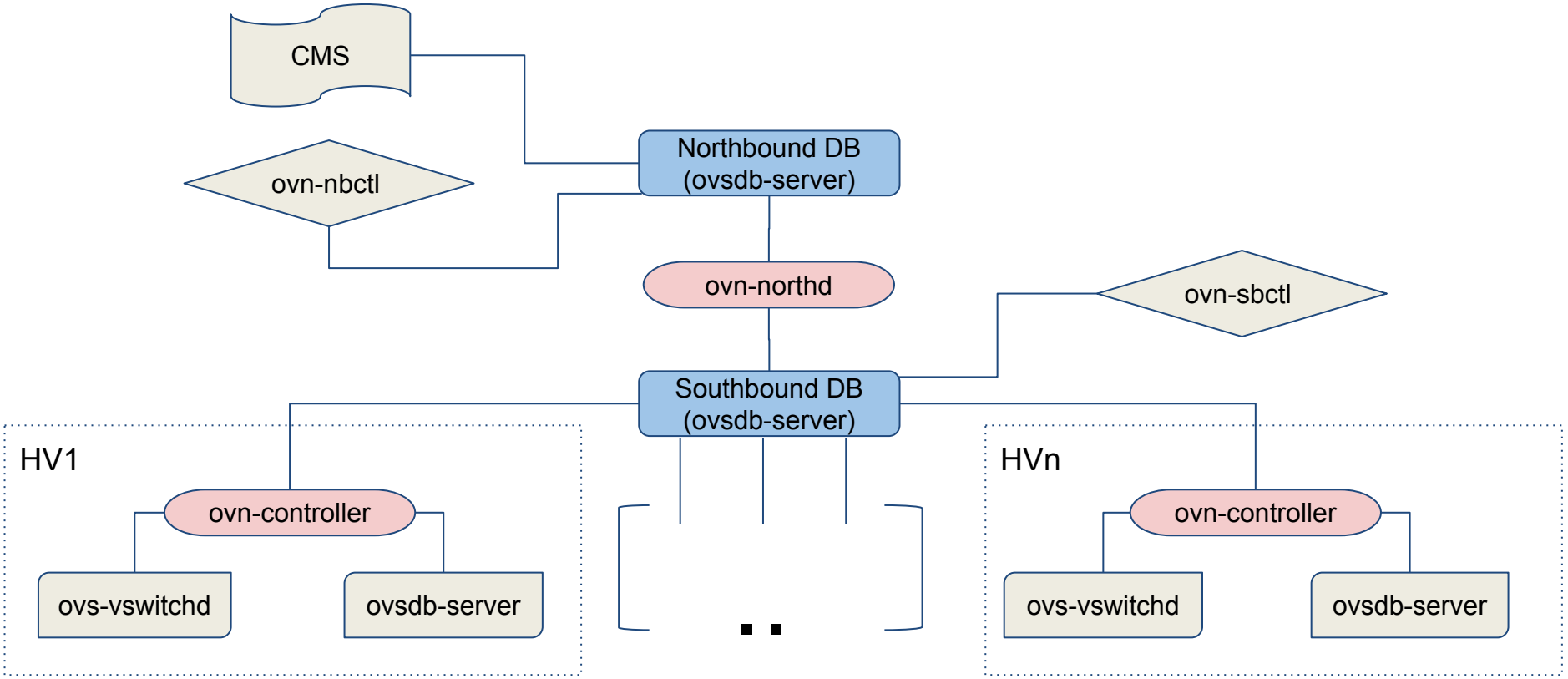


OVSDB performance updates '22
Testing with ovn-heater

Ilya Maximets, Red Hat

OVN Architecture Overview



Recap of 2021 performance improvements

“OVSDB: Performance and Scale Journey '21”
https://www.openvswitch.org/support/ovscon2021/#sT22

- Database file format changes in v2.15
- Algorithmic optimizations targeting operations on sets
- Deduplication of objects in memory
- …

Results for clustered databases:

- x15 performance improvement
- x6 memory consumption improvement

All changes are available in OVS v2.17

Do we need more performance?

- More performance is always better!

  But ...

Do we need more performance?

- More performance is always better!

  But ...

- **OVSDB** server did **not** appear as **a** performance **bottleneck** in any of our performance and scale tests since last year.

Do we need more performance?

- More performance is always better!

  But ...

- **OVSDB** server did **not** appear as **a** performance **bottleneck** in any of our performance and scale tests since last year.

Focus this year is on:

- Improving performance in corner cases.
- Chasing anomalies.
- Improving general stability.

Do we need more performance?

- More performance is always better!

  But ...

- **OVSDB** server did **not** appear as **a** performance **bottleneck** in any of our performance and scale tests since last year.

Focus this year is on:

- Improving performance in corner cases.
- Chasing anomalies.
- Improving general stability.

There are still some general performance improvements though.

ovn-heater – testing closer to a real world

Fully synthetic benchmarks are important, but not always representative of the “real world”.

ovn-heater – testing closer to a real world

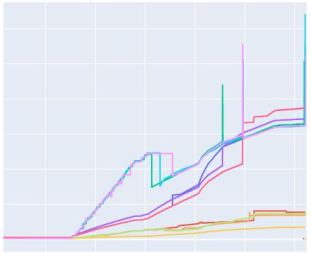
Fully synthetic benchmarks are important, but not always representative of the “real world”.

What is ovn-heater ?

- Testing tool designed to mimic behavior of a CMS (mostly ovn-kubernetes) on top of pure OVN setup.
- Under the hood - ovn-fake-multinode, expanded to work on actual multi-node setup.
- Can generate different workloads depending on a test scenario.
- Can gather performance data / measure how long it takes to provision resources.

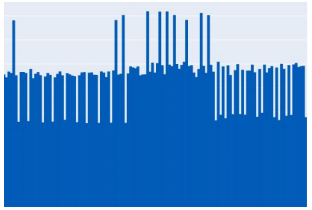
ovn-heater – testing closer to a real world

Fully synthetic benchmarks are important, but not always representative of the “real world”.



What is ovn-heater ?

- Testing tool designed to mimic behavior of a CMS (mostly ovn-kubernetes) on top of pure OVN setup.
- Under the hood - ovn-fake-multinode, expanded to work on actual multi-node setup.
- Can generate different workloads depending on a test scenario.
- Can gather performance data / measure how long it takes to provision resources.



- And it plots nice charts!

https://github.com/dceara/ovn-heater

Tes scenario

- 250-density-heavy test scenario:
  - **250** fake nodes
  - **13.750** logical ports (10 + 45 ports per node)
  - **16.875** load balancers
  - Port creation rate: ~20 per second

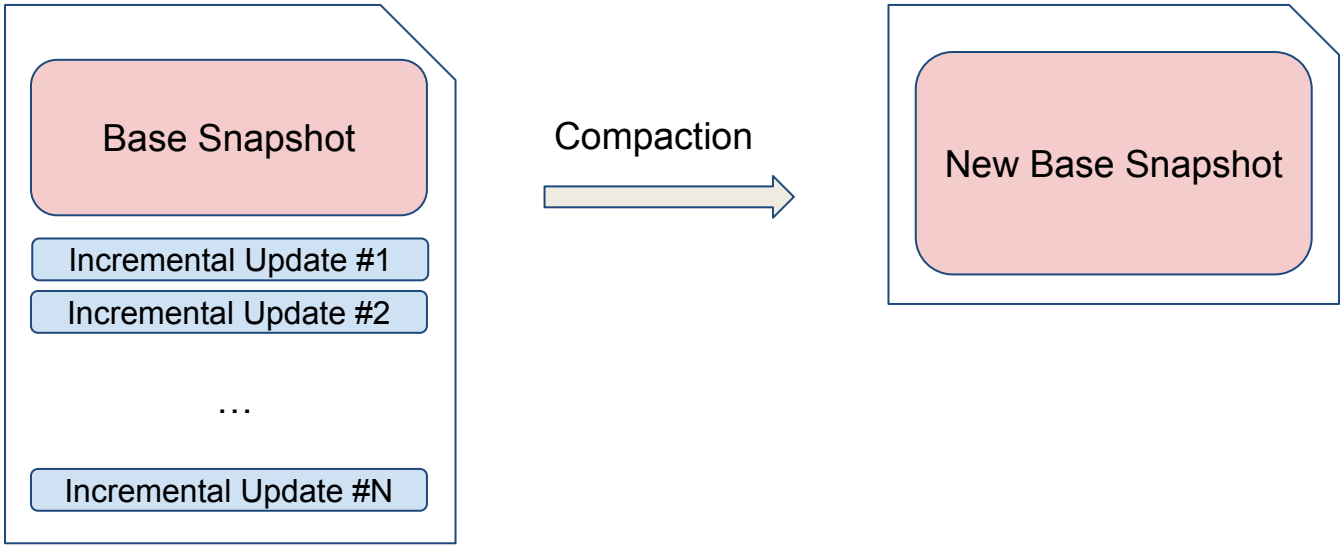
- Configuration:
  - Clustered database (RAFT, 3 servers)
  - ovn-monitor-all = true

Southbound DB CPU usage OVS 2.17 + OVN 22.06

Northbound DB CPU usage OVS 2.17 + OVN 22.06

Database compaction

- On-disk database files are incremental
- Easy way to add changes to a persistent storage at the cost of potentially infinite file growth.



Database compaction problem

- With the growth of a database, compaction can take significant amount of time:

```
ovsdb|INFO|OVN_Southbound: Database compaction took 8332ms
timeval|WARN|Unreasonably long 8345ms poll interval (7097ms user, 1221ms system)
```

- During that time the database server is not responsive.

Database compaction problem

- With the growth of a database, compaction can take significant amount of time:

```
ovsdb|INFO|OVN_Southbound: Database compaction took 8332ms
timeval|WARN|Unreasonably long 8345ms poll interval (7097ms user, 1221ms system)
```

- During that time the database server is not responsive.

- Partially solved by transferring the leadership to other cluster members.
  - Clients are notified about leadership change.
  - Leader-only clients (ovn-northd) can reconnect to a new leader.
  - Not leader-only clients (ovn-controller) **will not reconnect**.

Database compaction: Solution?

- Move compaction to a different thread?
    - Internal data structures are not thread-safe.
    - Need a way to allow database updates while the other thread is working on the same data to generate a new base snapshot.

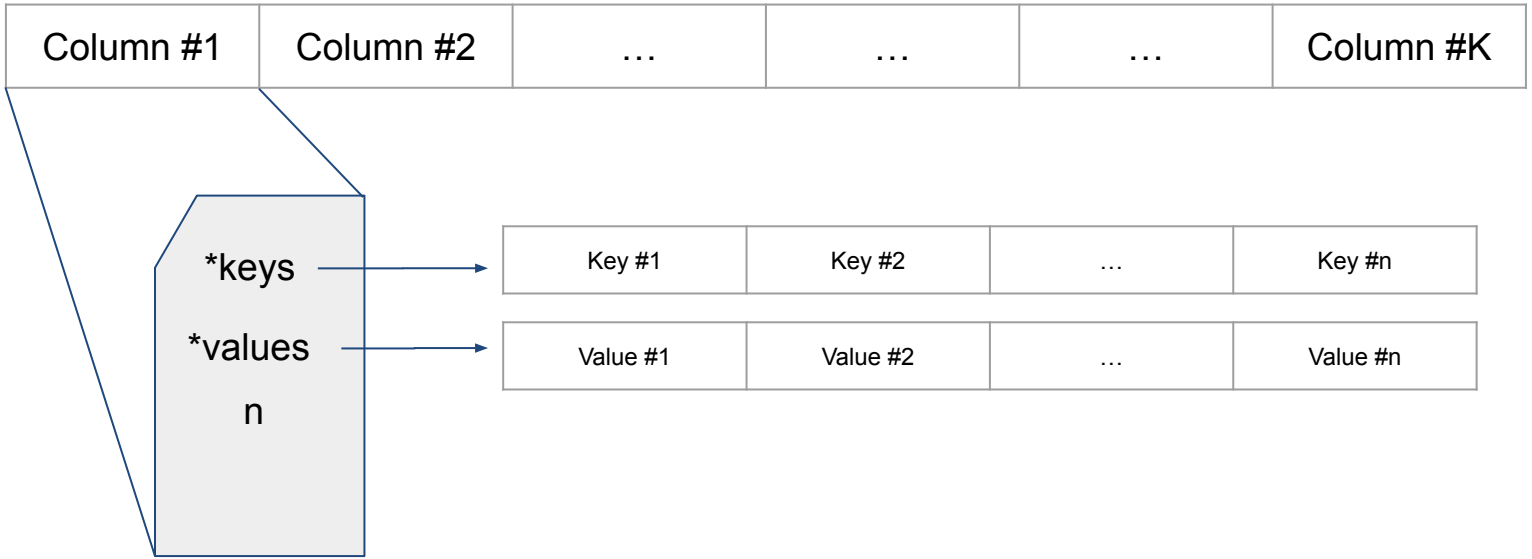
Database compaction: Solution?

- Move compaction to a different thread?
    - Internal data structures are not thread-safe.
    - Need a way to allow database updates while the other thread is working on the same data to generate a new base snapshot.

Copy on Write!

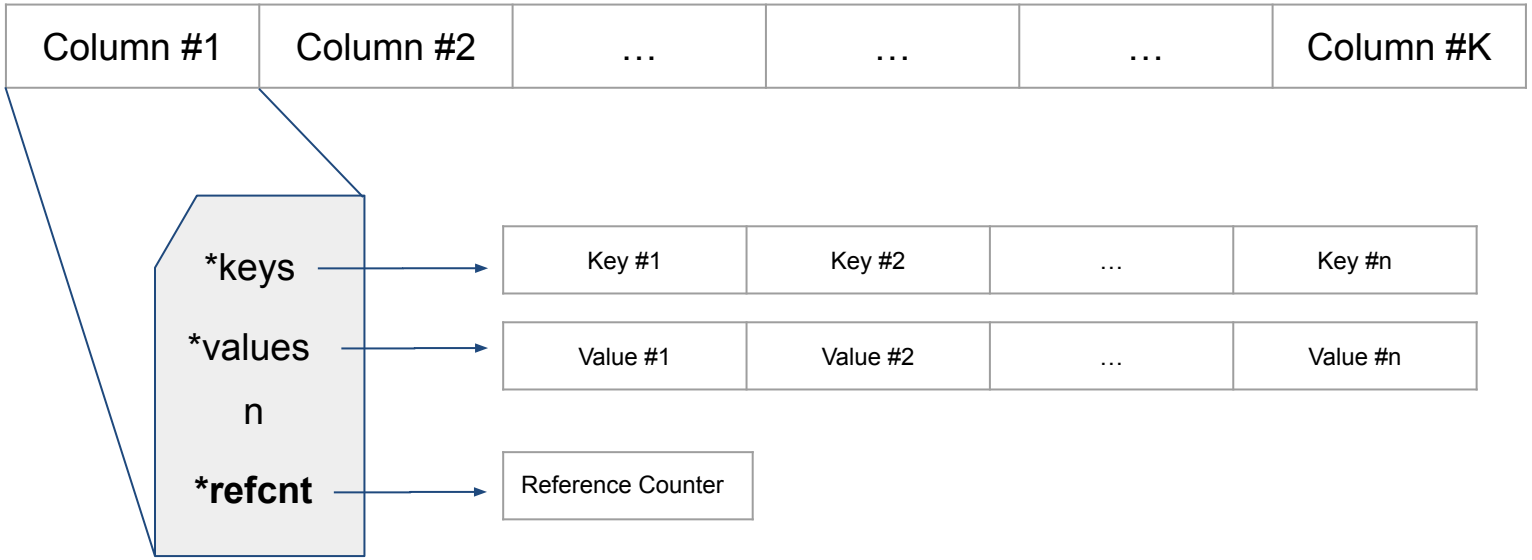
Database compaction: Solution

- Database Row:



Database compaction: Solution

- Database Row:



Database compaction: Solution

Workflow:

1. Main thread creates a shallow copy of the whole database.
2. Main thread starts a compaction thread with a copy of the database.
3. Compaction thread prepares a final serialized JSON representation of the database.
4. Main thread writes resulted database snapshot to the new file.
5. Main thread writes all the changes from the RAFT log that appeared after the creation of the shallow copy.

Database compaction: Solution

- Compaction thread only reads the data.
- Main thread creates a copy of the column every time it needs to modify it.
- No locks or other special synchronization is required!

Database compaction: Solution

- Compaction thread only reads the data.
- Main thread creates a copy of the column every time it needs to modify it.
- No locks or other special synchronization is required!

Bonuses:

- Reference counting mechanism for datum objects significantly reduced memory duplication across the process.
- Simple **copy of the column doesn't cost CPU time**!
  - This saved about **40%** of CPU time during the normal operation.

Database compaction: Results

- With the parallel compaction (from a larger 500-node test):

```
ovsdb(compaction16)|DBG|OVN_Southbound: Compaction thread started.
ovsdb(compaction16)|DBG|OVN_Southbound: Compaction thread finished in 4704 ms.
ovsdb|DBG|OVN_Southbound: Database compaction took 579ms
                                    (init: 168ms, write: 411ms, thread: 4704ms)
```

- In the case above we have 2 poll intervals:
  - 168 ms to create a shallow copy of the database and start the thread.
  - 411 ms to write resulted snapshot on disk.

- Compaction thread took 4.7 seconds to prepare a snapshot.

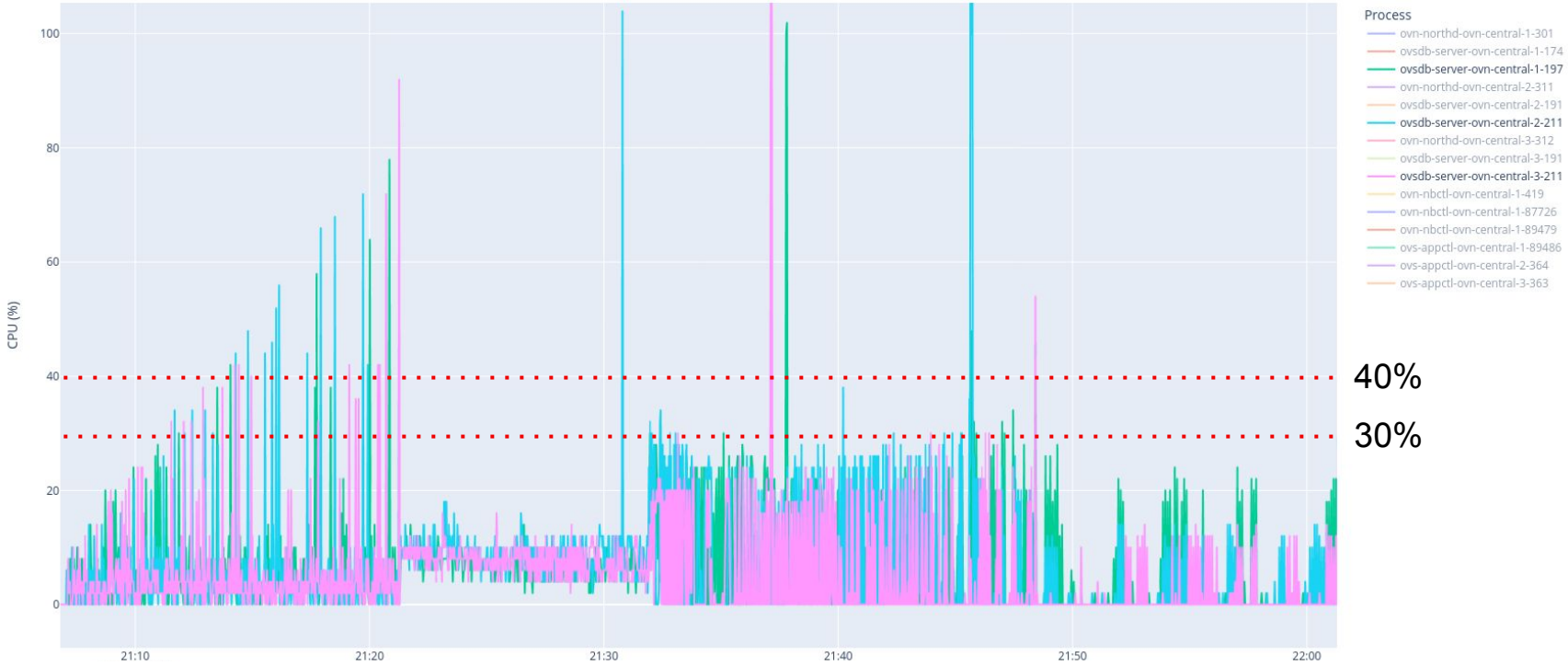
Other notable performance optimizations in OVS 3.0

- Improved performance of JSON parser. [Rosemarie O'Riorden]
  - Up to **40%** speed up in parsing JSON strings.

- New flag for the IDL clients - `OVSDB_IDL_WRITE_CHANGED_ONLY`. [Dumitru Ceara]
  - A way to avoid sending columns that didn’t change in the transaction.

- Use of a faster SHA1 implementation from OpenSSL library. [Ilya Maximets]

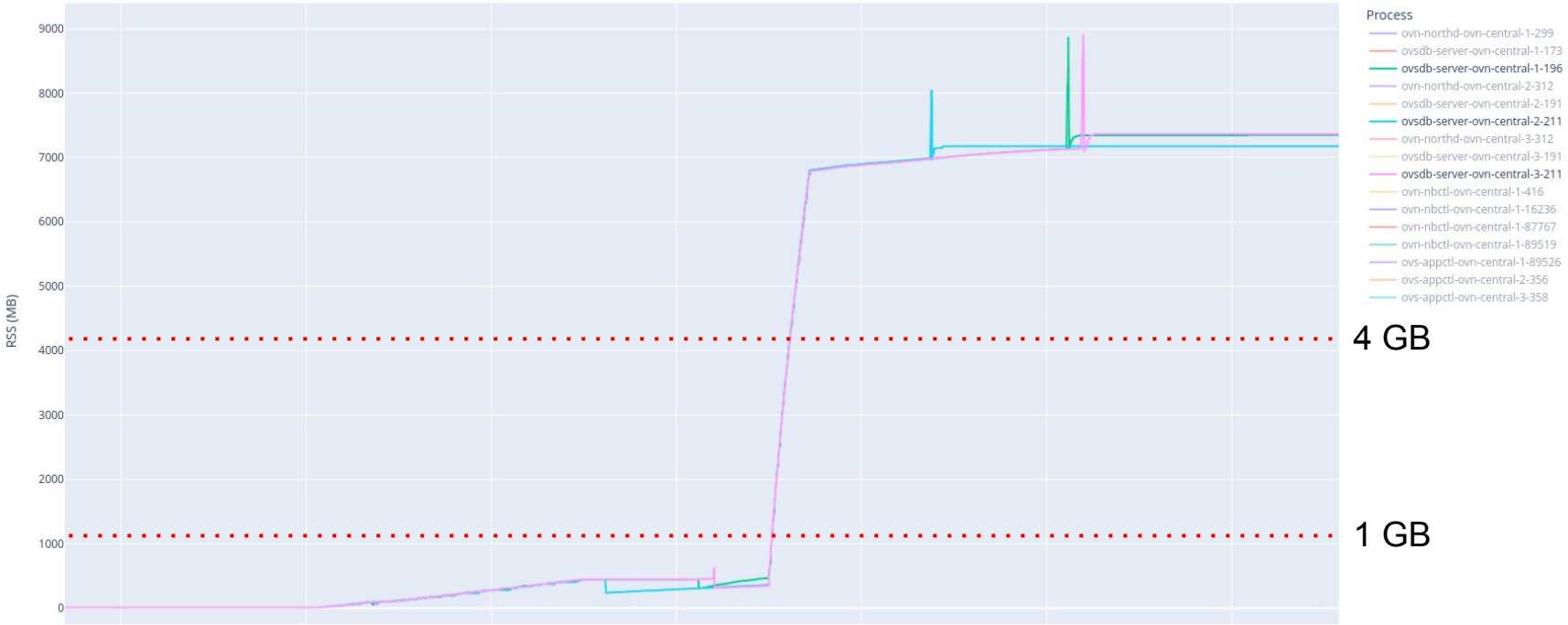
Southbound DB CPU usage OVS 3.0 + OVN 22.06



Southbound DB CPU usage OVN 22.06

Elephant in the room. OVS 2.17 + OVN 22.06

Southbound DB Memory Usage (RSS)

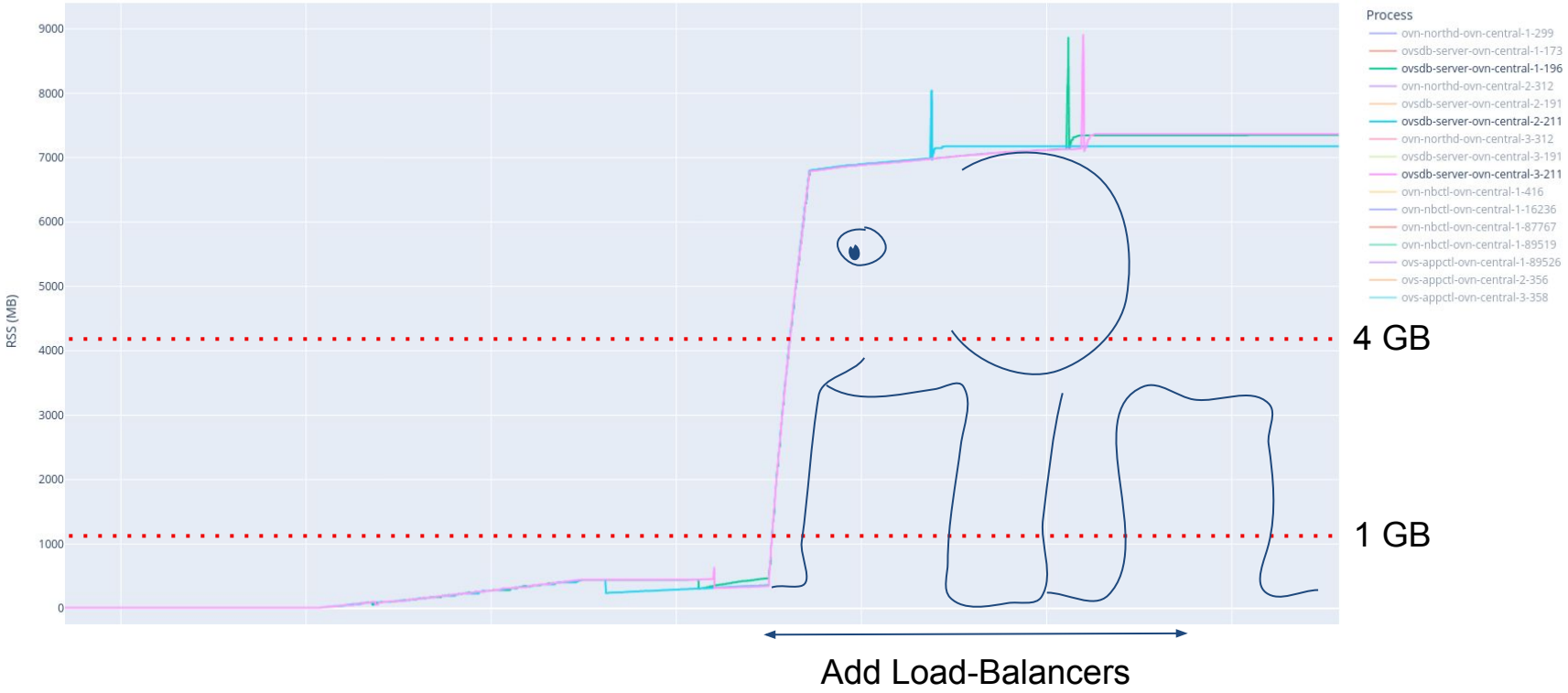


Elephant in the room. OVS 2.17 + OVN 22.06

Southbound DB Memory Usage (RSS)

Elephant in the room. OVS 2.17 + OVN 22.06

Southbound DB Memory Usage (RSS)



Southbound DB: memory usage

- Running the test under valgrind we got following result:

```
--------------------------------------------------------------------------------
  n        time(i)         total(B)    useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 20 1,011,495,832,314  11,610,557,104  10,217,785,620 1,392,771,484        0

    88.00% (10,217,785,620B) (heap allocation functions) malloc/new/new[]
    ->70.47% (8,181,819,064B) 0x455372: xcalloc__ (util.c:121)
      ->70.07% (8,135,785,424B) 0x41609D: ovsdb_weak_ref_clone (row.c:66)
        ->70.07% (8,135,785,424B) 0x41609D: ovsdb_row_clone (row.c:151)
          ->34.74% (4,034,041,440B) 0x41B7C9: ovsdb_txn_clone (transaction.c:1124)
          | ->34.74% (4,034,041,440B) 0x41B7C9: ovsdb_txn_add_to_history (transaction.c:1163)
          |   ->34.74% (4,034,041,440B) 0x41B7C9: ovsdb_txn_replay_commit (transaction.c:1198)
          |     ->34.74% (4,034,041,440B) 0x408C35: parse_txn (ovsdb-server.c:633)
          |       ->34.74% (4,034,041,440B) 0x408C35: read_db (ovsdb-server.c:663)
          |         ->34.74% (4,034,041,440B) 0x406C9D: main_loop (ovsdb-server.c:238)
          |           ->34.74% (4,034,041,440B) 0x406C9D: main (ovsdb-server.c:500)
          |
          ->34.74% (4,034,041,440B) 0x41B7DE: ovsdb_txn_clone (transaction.c:1125)
            ->34.74% (4,034,041,440B) 0x41B7DE: ovsdb_txn_add_to_history (transaction.c:1163)
              ->34.74% (4,034,041,440B) 0x41B7DE: ovsdb_txn_replay_commit (transaction.c:1198)
                ->34.74% (4,034,041,440B) 0x408C35: parse_txn (ovsdb-server.c:633)
                  ->34.74% (4,034,041,440B) 0x408C35: read_db (ovsdb-server.c:663)
                    ->34.74% (4,034,041,440B) 0x406C9D: main_loop (ovsdb-server.c:238)
                      ->34.74% (4,034,041,440B) 0x406C9D: main (ovsdb-server.c:500)
```

Southbound DB: memory usage

- Running the test under valgrind we got following result:

```
--------------------------------------------------------------------------------
  n        time(i)         total(B)    useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 20 1,011,495,832,314  11,610,557,104  10,217,785,620 1,392,771,484        0

    88.00% (10,217,785,620B) (heap allocation functions) malloc/new/new[]
    ->70.47% (8,181,819,064B) 0x455372: xcalloc__ (util.c:121)
      ->70.07% (8,135,785,424B) 0x41609D: ovsdb_weak_ref_clone (row.c:66)
        ->70.07% (8,135,785,424B) 0x41609D: ovsdb_row_clone (row.c:151)
```

- There is no need to copy weak reference tracking objects to the transaction history!

Southbound DB: memory usage

- Alternative solution - change the database schema:

```
"Load_Balancer": {
     "columns": {

         "datapaths": {
             "type": {"key": {"type": "uuid",
                              "refTable": "Datapath_Binding"},
                      "min": 0, "max": unlimited"}},
         "datapath_group":
             {"type": {"key": {"type": "uuid",
                               "refTable": "Logical_DP_Group"},
                       "min": 0, "max":1}},

     "isRoot": true},
```

Southbound DB: memory usage

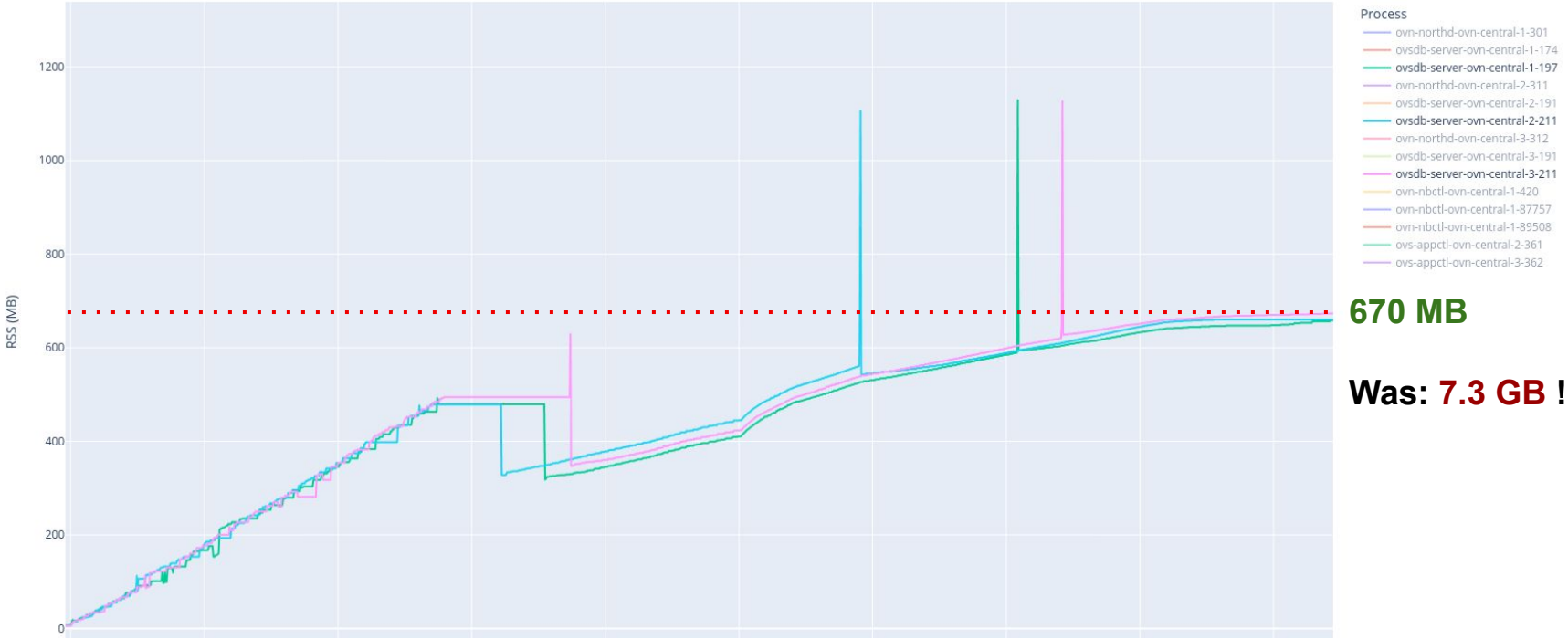
- Alternative solution - change the database schema:

```
"Load_Balancer": {
     "columns": {

          "datapaths": {
              "type": {"key": {"type": "uuid",
                               "refTable": "Datapath_Binding"},
                       "min": 0, "max": unlimited"}},
          "datapath_group":
              {"type": {"key": {"type": "uuid",
                                "refTable": "Logical_DP_Group"},
                        "min": 0, "max":1}},

          "isRoot": true},
```

- Both solutions are valid and have their own benefits.
- So, using both!

Memory usage. OVS 3.0 + OVN 22.09 (both solutions)

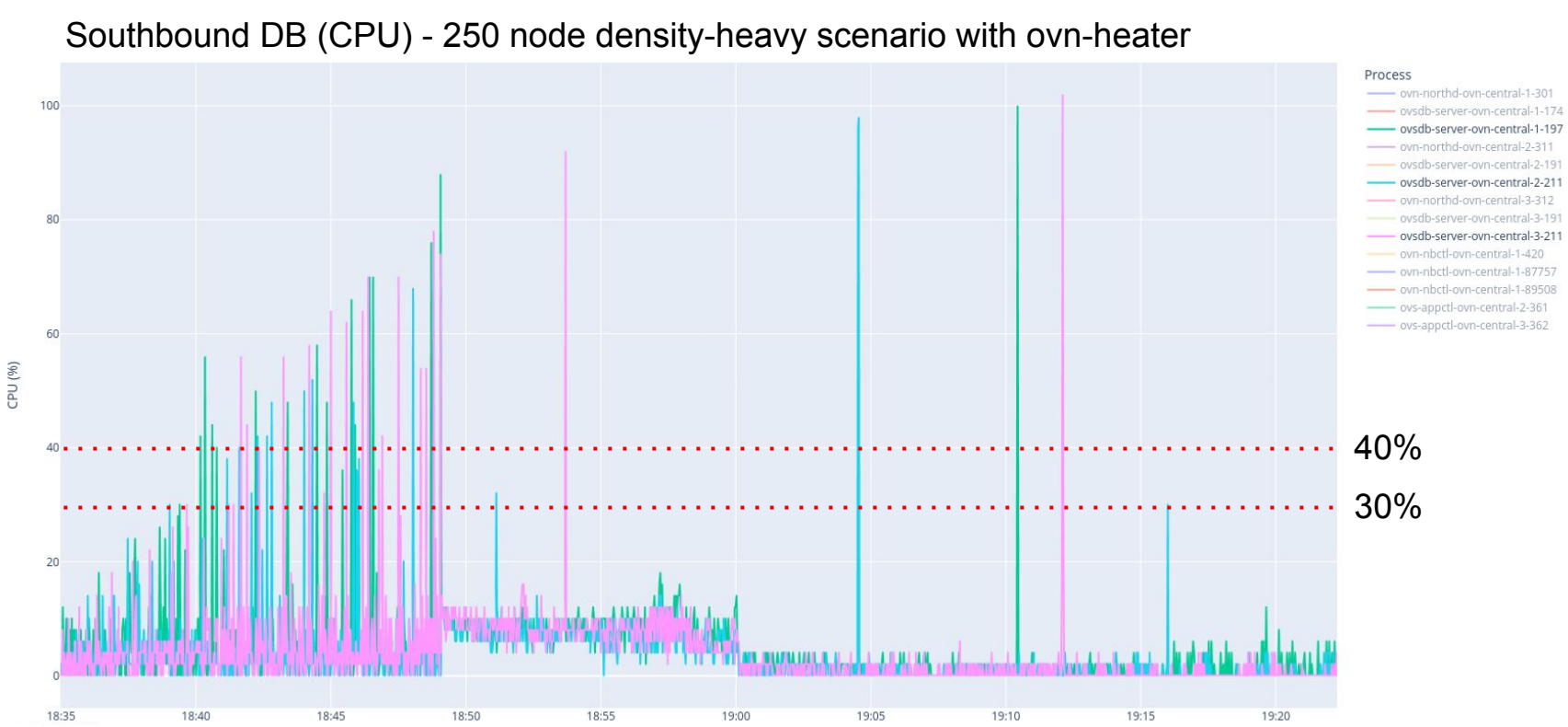
Southbound DB (RSS) - 250 node density-heavy scenario with ovn-heater



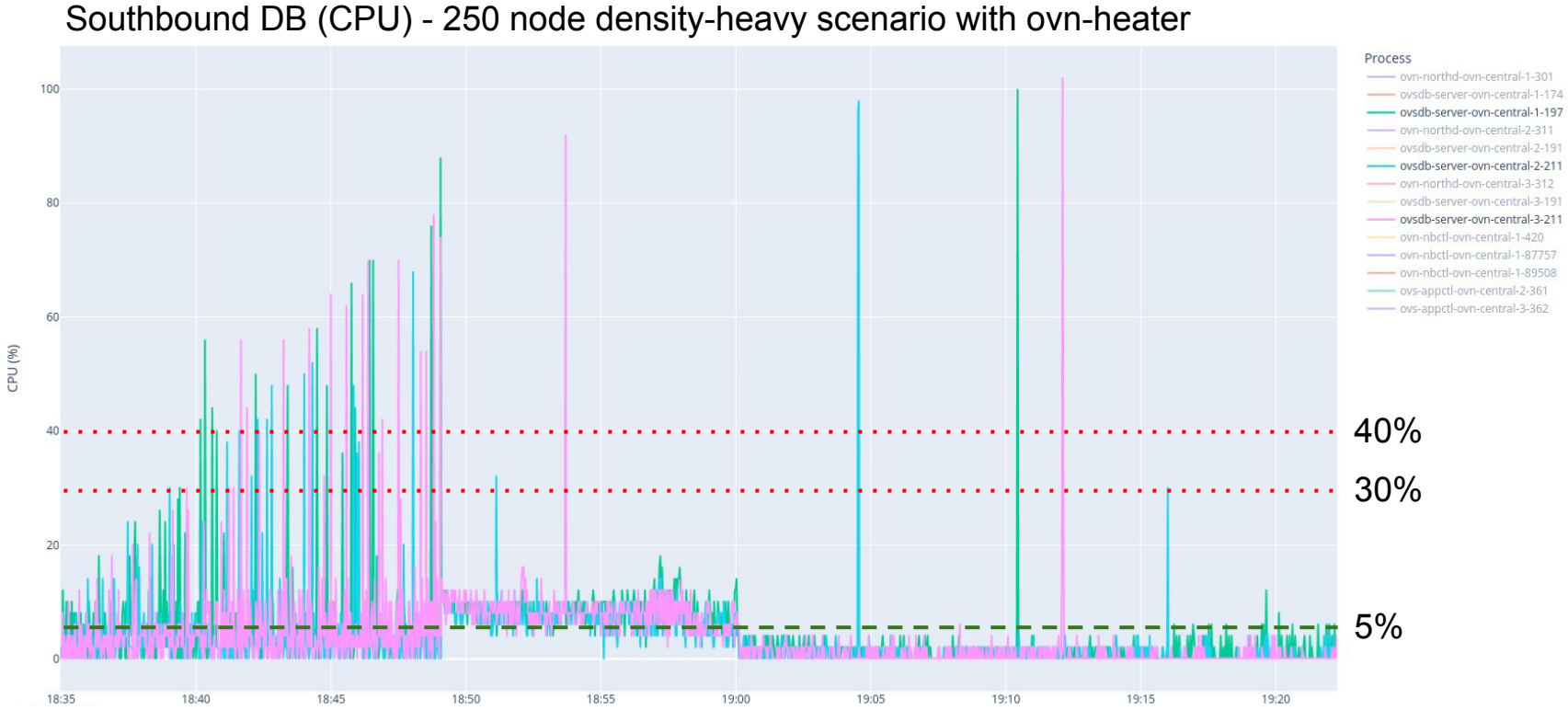
670 MB

Was: 7.3 GB !

Southbound DB CPU usage OVS 3.0 + OVN 22.09



Southbound DB CPU usage OVS 3.0 + OVN 22.09



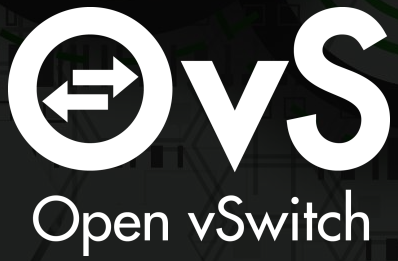
OVSDB Relay Updates

- Added transaction history support, a.k.a. fast re-sync.
- Improved performance of column mutations.

Other notable changes

- Bug fixes (backported to all stable branches):
  - Fixed transaction double commit problem on leadership transfer.
  - Fixed an issue with too frequent unnecessary compactions.

- Added support for logging changes in particular tables. [Dumitru Ceara]
  - `ovs-appctl -t ovsdb-server ovsdb-server/tlog-set DB:TABLE on|off`



Thanks!

Email: i.maximets@ovn.org