Gefördert durch:

Ministerium für Innovation,
Wissenschaft und Forschung
des Landes Nordrhein-Westfalen

Einfach. Sicher. Riesig.

30.1.2017

sciebo
dieCampuscloud

Database Clusters for Sync and Share Services

Holger Angenent, Andreas Wilmer
University of Münster

> Databases

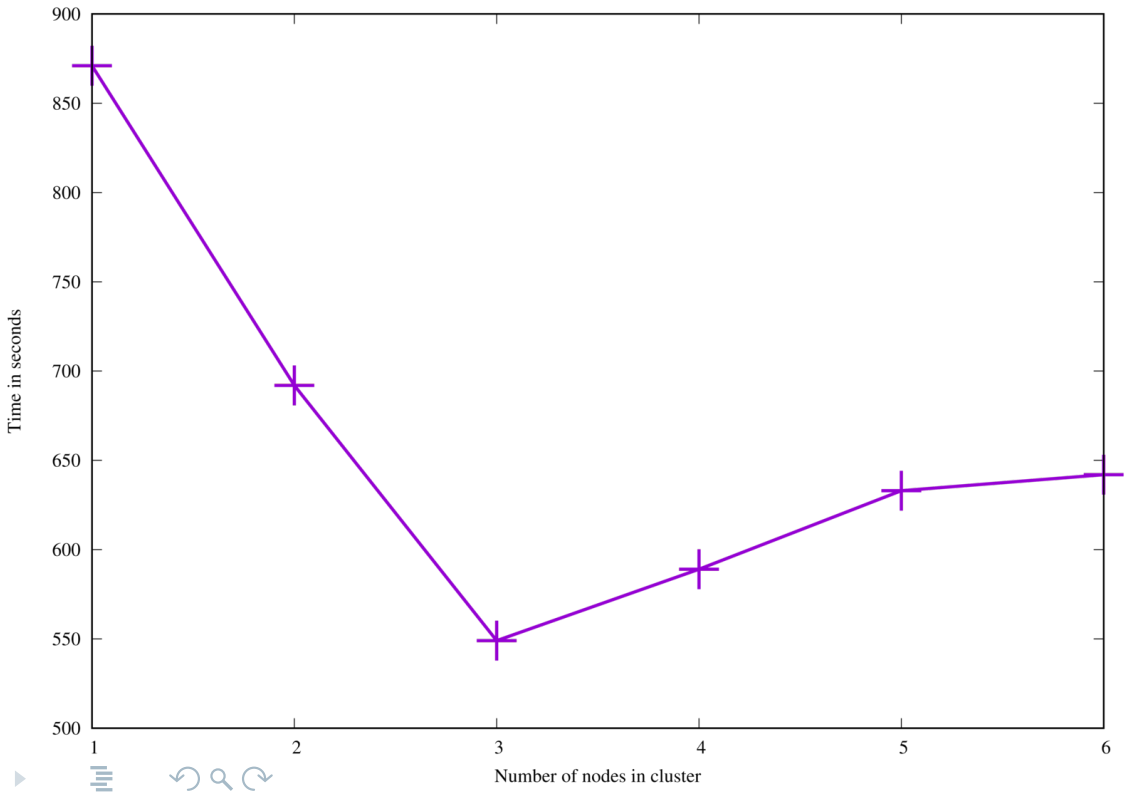
- Large sync and share installations require fast database clusters
- What number of nodes is to be desired?
- Capability to write data does not scale with number of nodes since data is written to all nodes
- On the other hand data is mostly read, so you want to have as many as possible
- But how much will writing be slowed down due to many nodes?
- Make some experiments with a test system and compare results with data from sciebo

> Setup of Test System

- 7 nodes
- 32 GB RAM
- 8 cores, 2,3 GHz, AMD Opteron Processor 2352
- Apache Server with CentOS 6 and php7 on head node
- 6 database nodes with MariaDB 5.5 and Galera
- Local HDDs
- DDR Infiniband and 1 GBit Ethernet
- owncloud 9.1.3, owncloud client 2.2.4

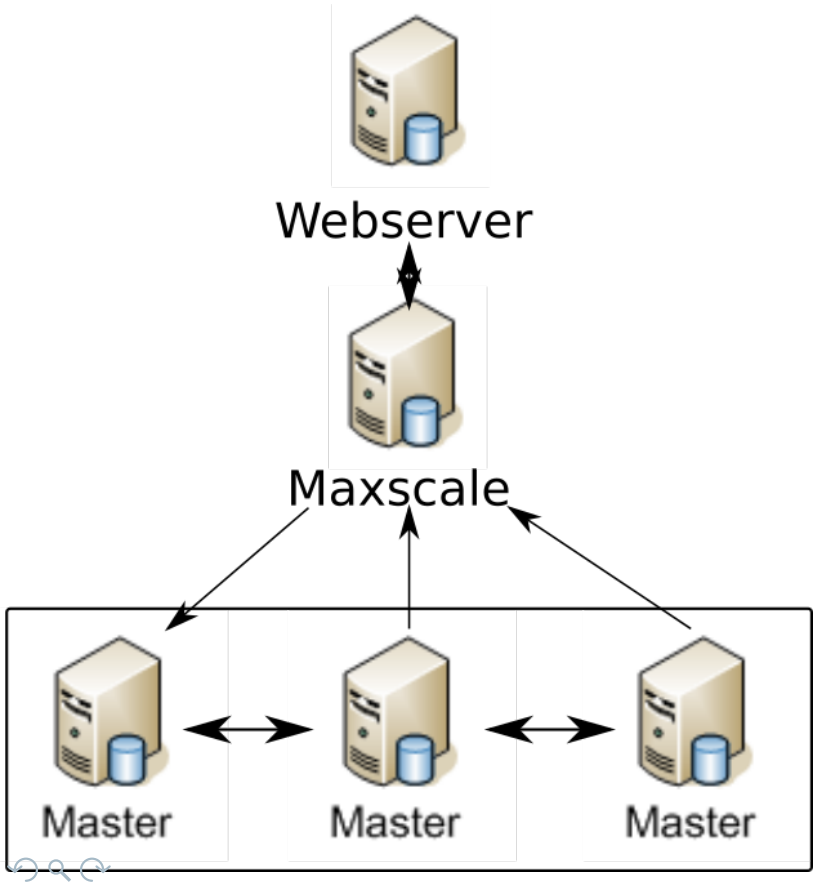
> Results from Test System

- Upload 32,000 files with 16 processes of owncloud command line client via SmashBox
- Measure required time
- Test both networks
- No other load on database, Webserver was fully utilized



> Flow Control

- A node gets work from the master node via replication and from Maxscale
- If it cannot write fast enough, it sends the master node a signal to stop sending more write requests

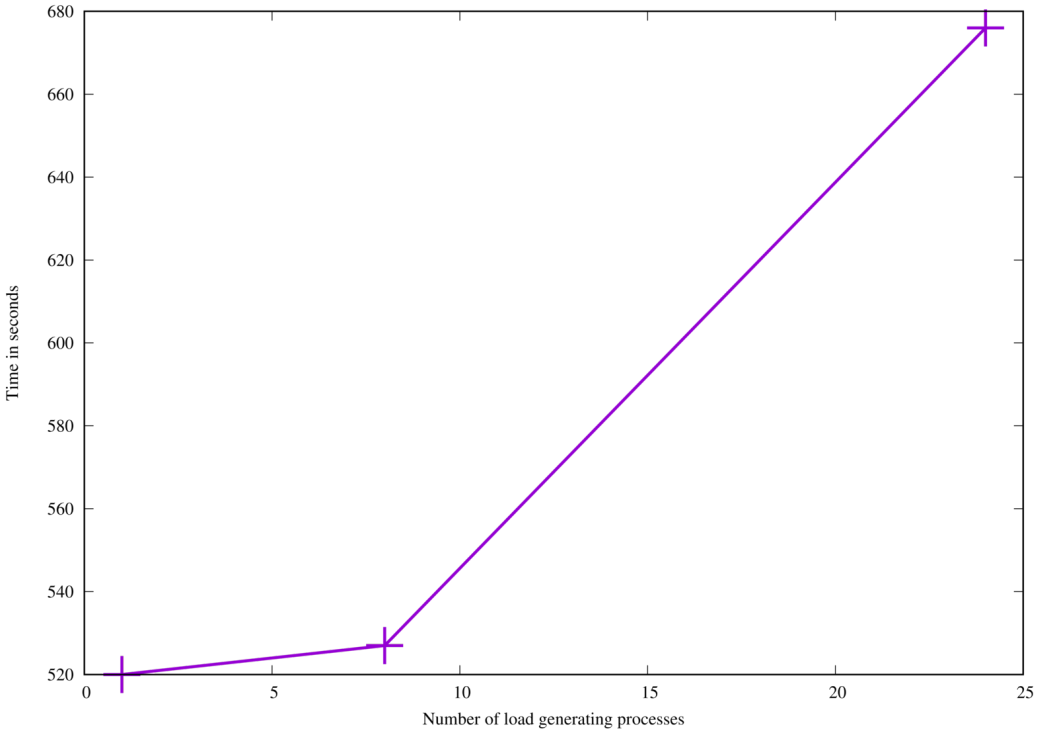


> Results from Test System

- What happenes, if there is (too much) load on the database?
- Saturate a cluster node with (nonsense) load

```
SELECT * FROM oc_filecache WHERE name like \'%'.$randword.'%\''
```

in loop



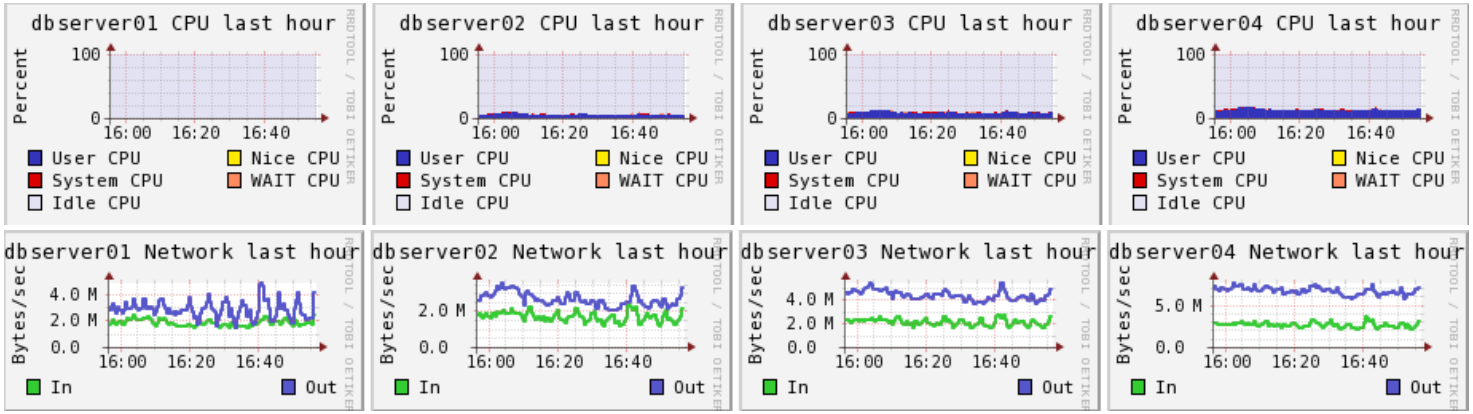

```
wsrep_local_send_queue       | 0
wsrep_local_send_queue_max   | 1
wsrep_local_send_queue_min   | 0
wsrep_local_send_queue_avg   | 0.000000
wsrep_local_recv_queue       | 1
wsrep_local_recv_queue_max   | 59
wsrep_local_recv_queue_min   | 0
wsrep_local_recv_queue_avg   | 3.431916
wsrep_local_cached_downto    | 6311594
wsrep_flow_control_paused_ns | 49293740711
wsrep_flow_control_paused    | 0.021224
wsrep_flow_control_sent      | 546
wsrep_flow_control_recv      | 550
```

3 active nodes Lock table had similar effects

sciebo
dieCampuscloud

> Results from Sciebo

- Allmost all load is on the reading nodes
- Flow control sometimes activated



wsrep_local_send_queue	0
wsrep_local_send_queue_max	1
wsrep_local_send_queue_min	0
wsrep_local_send_queue_avg	0.000000
wsrep_local_recv_queue	0
wsrep_local_recv_queue_max	1010
wsrep_local_recv_queue_min	0
wsrep_local_recv_queue_avg	0.314111
wsrep_local_cached_downto	14789926640
wsrep_flow_control_paused_ns	196037331250
wsrep_flow_control_paused	0.000220
wsrep_flow_control_sent	61
wsrep_flow_control_recv	355

With `gcs.fc_limit=500` and about 9000 active users

sciebo
dieCampuscloud

> Results from Sciebo

- The users on this cluster are separated to 8 different instances
- We saw the whole cluster getting slower, when there was too much load on one of them (probably due to flow control)

⇒ If you have already separate instances, different mySQL processes might speed up the whole thing.

How much memory is needed?

- 76 GB of metadata (according to mysqltuner.pl) for about 30k users
- Activity log for 7 days
- 9 GB could be freed by OPTIMIZE TABLE on *oc_activity*

⇒ A smaller machine would possibly do the job, too.

> Conclusion

- I would prefer a database cluster with more smaller nodes
- Find out, what could stress the database that much that it triggers flow control