Main page
Recent changes
Server admin log (Prod)
Server admin log (RelEng)
Deployments
SRE/Operations Help
Incident status

Cloud VPS & Toolforge
Cloud VPS documentation
Toolforge documentation
Request Cloud VPS project
Server admin log (Cloud VPS)

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Incident documentation/20110926-DBandPower

< Incident documentation

Contents [hide]

Documentation by Mark Bergsma

Access switch power outage at 14:12 UTC

Chris, our Tampa data center contractor, was replacing the broken management switch msw-d2-sdtpa in sdtpa rack D2. Our management network is completely unrelated to our production network, and doesn't carry any important, production traffic. However, the management switch is mounted directly under the production network access switch (asw-d2-sdtpa). During the replacement, at 14:12 UTC, the power cable of asw-d2-sdtpa came slightly loose, as it was very close to the switch and cables being replaced. Chris didn't notice this, as all LEDs of the switch are on the other side.

As a result, the entire rack, consisting of about 40 MediaWiki application servers, lost network connectivity. Nagios complained about all servers in the rack, and due to all kinds of second order effects, many other services were (briefly) disrupted or overloaded, leading to many other Nagios downtime reports. Several Ops and Engineering members were looking at the downtime, but had difficulty pinpointing exactly what was going down.

After a few minutes, I noticed that all application servers being reported down were located in the same rack D2 (as I checked in Racktables), and that asw-d2-sdtpa was reported down by Observium (e-mail to root@ and the web interface). A quick check revealed that indeed this switch had gone down. I asked Chris on IRC if this was related to his work, which it was indeed. As soon as Chris restored power, all network connectivity and services were restored within a minute at 14:30 UTC.

The main problem with a rack of application servers going down, besides the significant reduction in capacity, is the loss of memcached caching - meaning that MediaWiki on the remaining servers will be parsing/regenerating many pages and objects, resulting in a much higher load at reduced capacity for a while. memcached redundancy/resiliency is definitely something that could need improvement for this reason.

In our newer racks and data centers, we also have better redundancy of the network, including redundant power and uplinks for access switches, so this doesn't happen so easily. If we do not intend to move Tampa any time soon, we should look at retrofitting our older racks with that setup as well.

Database master switch outage at 18:23

For the schema changes of the MediaWiki 1.18 release, Asher has been master changes for all core db clusters. Usually this goes automatically without incidents, but cluster s7 also contains the CentralAuth database, used by all wikis. Before changing masters, Asher set cluster s7 to read-only in the MediaWiki db configuration. As soon as he changed the s7 master to a new database server at 18:23, the new master got bombarded with MediaWiki "slave lag" queries, i.e. MediaWiki instances trying to determine the replication lag of the slaves compared to the master, to select which db slave to use. Normally, this check is disabled when MediaWiki is in database read-only mode, but as was discovered later, this does not apply to CentralAuth queries - possibly because these are also issued by wikis outside the s7 cluster. Therefore, the new master was immediately overloaded with these queries, and interrupted normal queries and replication traffic, bringing down the sites.

Asher investigated the situation, and killed all outstanding queries on the new master, which fixed the problem at 18:32. He also filed a bugzilla ticket #31170 to improve MediaWiki's behavior during db read-only situations. Additional measures to avoid this problem in the future could be to split off CentralAuth into its dedicated cluster on dedicated hardware (we could use some old DB servers for that.

Another database related outage at 22:41 UTC.

Apparently there was another master switch related outage at this time with db cluster s5, but I was not around at that time, and haven't had an opportunity to discuss this incident with Asher yet. As far as I understand from the IRC backlog, the MySQL instances unexpectedly hung after Asher issued the "STOP SLAVE" command to stop replication, but I don't yet know the details.

Asher, could you give us a summary on that?

Additional notes by Asher Feldman

Our master switch scripts starts off with the following course of action:

1. set the old (still current) master to read-only
2. in the case of MasterSwitch.php, kill queries on the old master running for more than 10 seconds
3. in the case of both sets of switch scripts, run "flush tables" against it, to ensure that pending transactions are committed. Unless sql_bin_log is disabled for a session, running "flush tables" on a master is replicated to the slaves.

db44, one of the two dewiki slaves had the following long running queries in the "copying to tmp table" state:

```
# User@Host: wikiadmin[wikiadmin] @  []
# Query_time: 40459.620905  Lock_time: 0.000076 Rows_sent: 684  Rows_examined: 
108189072335
SET timestamp=1317034172;
SELECT
                        MIN(rev_timestamp) AS rt, MAX(p.fr_timestamp) AS pft, 
MIN(n.fr_timestamp) AS nft
                FROM `revision`
                INNER JOIN `page` FORCE INDEX (PRIMARY) ON
                        (page_id = rev_page AND page_namespace IN 
('0','6','10','14'))
                INNER JOIN `flaggedrevs` AS p FORCE INDEX (PRIMARY) ON
                        (p.fr_page_id = page_id AND p.fr_rev_id < rev_id AND 
p.fr_timestamp < rev_timestamp)
                LEFT JOIN `flaggedrevs` AS n FORCE INDEX (PRIMARY) ON
                        (n.fr_page_id = page_id AND n.fr_rev_id >= rev_id AND 
n.fr_timestamp >= rev_timestamp)
                WHERE
                        ((rev_user = 0) AND (rev_timestamp BETWEEN 
'20080523084941' AND '20080530084941') AND ((rev_id % 23) = 0))
                GROUP BY rev_id;
```

It looks like this ~11 hour long query is being run against dewiki every several hours, with the same values in the where clause. It should be returning the same 684 rows each time after apparently examining ~100 billion (!). At the time of the attempted switch, three of these had been running for between 2-8 hours:

```
| 85340307 | wikiadmin   | | dewiki | Query   | 28531 | Copying to tmp table | 
SELECT
                        MIN(rev_timestamp) AS rt, MAX(p.fr_timestamp) AS pft, 
MIN(n.fr_timestamp) AS nft
                FROM `r |
| 85856932 | wikiadmin   | | dewiki | Query   | 21079 | Copying to tmp table | 
SELECT
                        MIN(rev_timestamp) AS rt, MAX(p.fr_timestamp) AS pft, 
MIN(n.fr_timestamp) AS nft
                FROM `r |
| 86957051 | wikiadmin   | | dewiki | Query   |  7221 | Copying to tmp table | 
SELECT
                        MIN(rev_timestamp) AS rt, MAX(p.fr_timestamp) AS pft, 
MIN(n.fr_timestamp) AS nft
```

When flush tables ran on this host, it locked all tables and essentially hung there, waiting for the multi-hour temp table writes to complete. Mysql will still happily accept connections and queries, which will be locked in "waiting for table" status. After 5 minutes, things looked like:

```
| 31026092 | system user |                       |        | Connect |   342 |
Flushing tables      | flush tables

| 87306315 | root        | | NULL   | Query    |    68 | Killing slave          |
slave stop
| 87302413 | wikiuser    | | dewiki | Query    |   342 | Waiting for table      |
SELECT /* LinkCache::addLinkObj Sk!d */
page_id,page_len,page_is_redirect,page_latest  FROM `page`  |
| 87302455 | wikiuser    | | dewiki | Query    |   342 | Waiting for table      |
SELECT /* Article::pageData  */
page_id,page_namespace,page_title,page_restrictions,pa
...
| 87306315 | root        | | NULL   | Query    |    68 | Killing slave          |
slave stop <-- hanging on flush tables
...
| 87306478 | wikiuser    | | dewiki | Query    |    68 | Waiting for table      |
SELECT /* ApiPageSet::initFromTitles */
page_namespace,page_title,page_id,page_is_red |
| 87306479 | wikiuser    | | dewiki | Query    |    68 | Waiting for table      |
SELECT /* LinkHolderArray::replaceInternal */ page_id, page_namespace,
page_title, pag |
...
```

A large number of apache workers were tied up by sending blocked queries to this db, resulting in the actual site outage. A running flush tables cannot truly be killed directly (http://bugs.mysql.com/bug.php?id=44884) - the site was restored by sending mysql on this host a sigterm, preventing it from accepting additional incoming connections and killing everything off. Mysql on db43 was then restarted, and the rest of the replication changes went smoothly.

Modifying the switch scripts to prevent the "flush tables" statement on the master from replicating, as well as automated killing of long running queries on the slaves (instead of just the old master) should prevent a recurrence of this scenario.

Category: Incident documentation