Post mortem SRM indicent 10-11 April 2009

De WikiPIC, la enciclopedia libre.

Table of contents

Executive summary

On Friday 10/04/2009 (Easter) there was severe performance degradation of the SRM service at PIC offered to the WLCG collaboration for around 8 hours (from 10/04/09 18:15 UTC to 11/04/09 02:00 UTC). The problem was not completely solved until 11/04/09 11:05 UTC.

Follow-up actions

- Get insight into tuning the postgresql database for the SRM server so that it does not require the vacuum flag. There's currently an issue open with dcache developers ragarding this.

Problem detection and reporting

On the 9th of April, at 7pm, we noticed two critical alerts in Nagios. The first one was an overload on our srm server, and the second one a direct consequence of this overload (a service check timeout of the GRIDFTP_SRM_MONITORING nagios sensor). The first reaction was to send a mail notification to the responsible of the service. This did not had any impact on the availability of the site and was correlated to an increase in activity from the MAGIC experiment.

On the 10th of April, at 9pm, we got our first SAM test failing. This triggered alarms via SMS to our Manager on Duty, who followed the procedures specified for that case. Those were ineffective due to the problem being too specific.

Problem timeline/follow-up

1. The friday 10th at 11am, a phone call was made to a dcache administrator to have some directives about the procedure to follow in case of deterioration of the situation. Decision was made to do nothing until the service was not affected. The services was not yet degraded.
2. At 9pm this same day, a second phone call was made as the situation was critic (load average of the srm server at more than 80 and all the SAMtest_CE in failed) to take immediate actions.
3. In the first analysis it looked like to be a user driven DoS. As a consecuence, the user was banned.
4. User banning was not enough and we had to deny/slow down FTS transfer rate while investigation was going on.
5. Final solution (see technical analysis) was found and applied (11/04/09 @ 11:05 UTC).

Technical Analysis

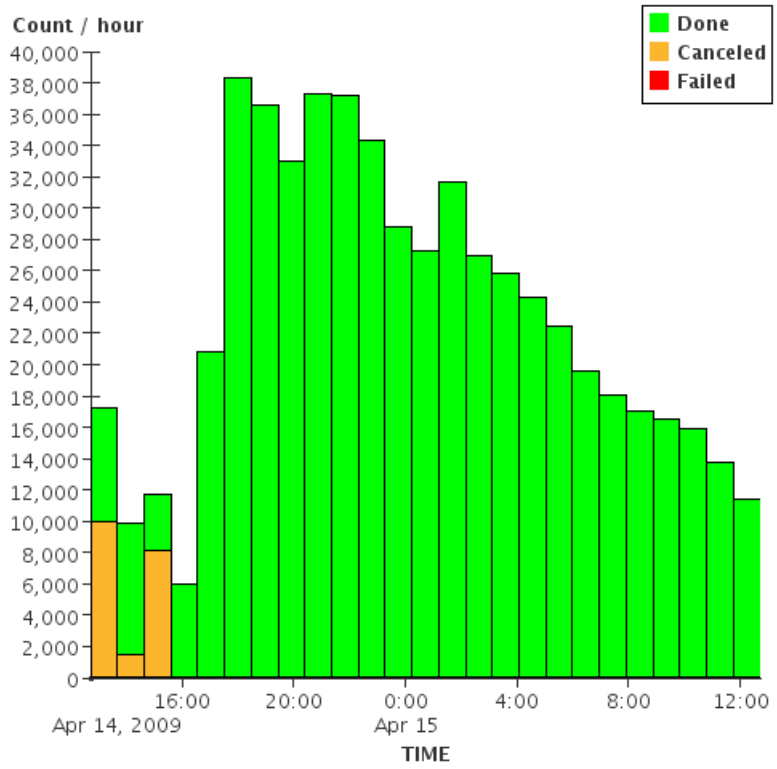
We found that the SRM was handling about 40k transfers/hour and the non-expected thing was that SRM's postgresql server process was keeping busy almost all CPU resources on the SRM server while the normal situation (at least for us) was to have Java using much more CPU than the pgsql server. Restarting the SRM server was to no use. Most of the transfers were issued by the MAGIC VO at PIC, that shares some facilities with the WLCG VOs. That situation was an extraordinary transfer of many small files. The problem was not with many simultaneous get/put operations (magic issued at most 3 of such at the same time) but the continuous submission of transfers that were making the postgresql tables become less and less tidy and, with no vacuum, became less and less efficient, making each subsequent query slower up to the point that we had an almost unresponsive SRM server.

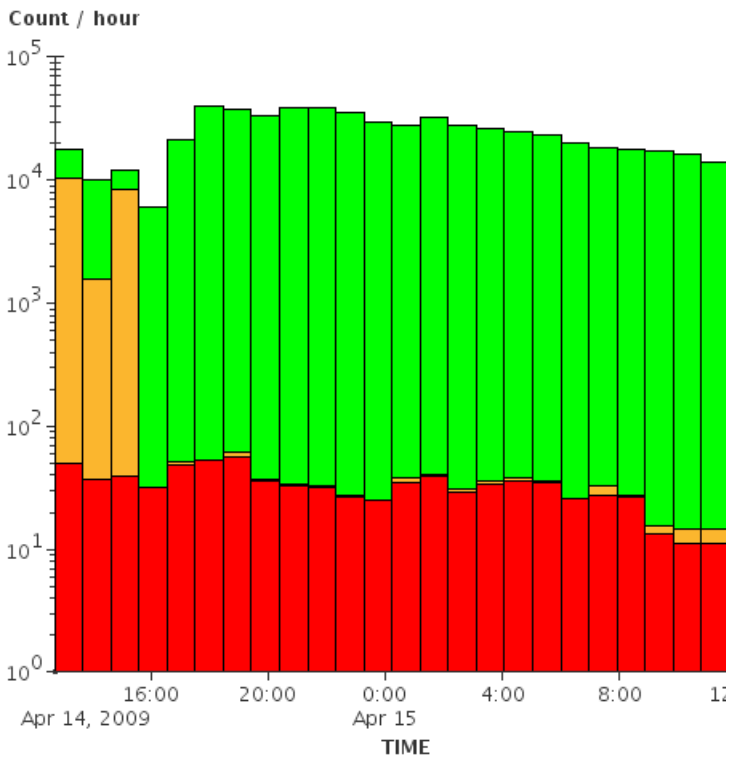
After some investigation we realized that what did the difference was having srmVacuum=false (as recommended on http://www.dcache.org/manuals/Book/config/cf-srm-srm.shtml). Setting it back to true and restarting the SRM solved the overload situation. We saw that having srmVacuum=false and restarting the SRM daemon (/etc/init.d/dcache restart) made the machine's CPU was back to 100% used and 60 loaded in a few seconds (many SRM queries timed out). After setting srmVacuum=true and restarting (11/04/09 @ 11:05 UTC) load remained stable to less than 1 with a CPU usage of about 10-30%.

The downside of having srmVacuum=true is that SRM's postgresql log warnings about "only table or database owner can vacuum it" are back:

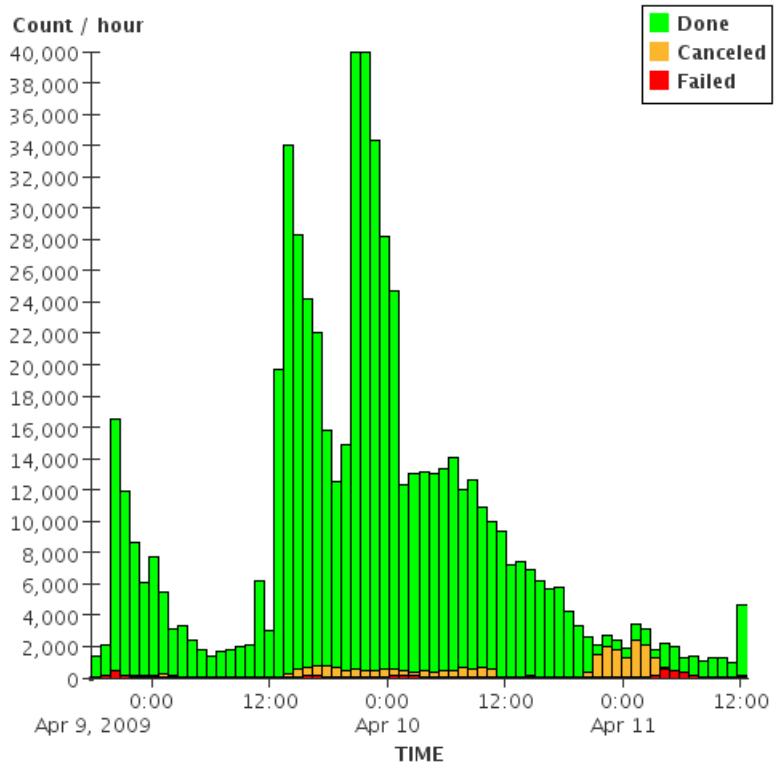
```
WARNING:  skipping "pg_authid" --- only table or database owner can vacuum it
WARNING:  skipping "pg_database" --- only table or database owner can vacuum it
WARNING:  skipping "pg_tablespace" --- only table or database owner can vacuum it
WARNING:  skipping "pg_pltemplate" --- only table or database owner can vacuum it
WARNING:  skipping "pg_shdepend" --- only table or database owner can vacuum it
WARNING:  skipping "pg_shdescription" --- only table or database owner can vacuum it
WARNING:  skipping "pg_auth_members" --- only table or database owner can vacuum it
WARNING:  skipping "pg_authid" --- only table or database owner can vacuum it
WARNING:  skipping "pg_database" --- only table or database owner can vacuum it
WARNING:  skipping "pg_tablespace" --- only table or database owner can vacuum it
WARNING:  skipping "pg_pltemplate" --- only table or database owner can vacuum it
WARNING:  skipping "pg_shdepend" --- only table or database owner can vacuum it
WARNING:  skipping "pg_shdescription" --- only table or database owner can vacuum it
WARNING:  skipping "pg_auth_members" --- only table or database owner can vacuum it
```

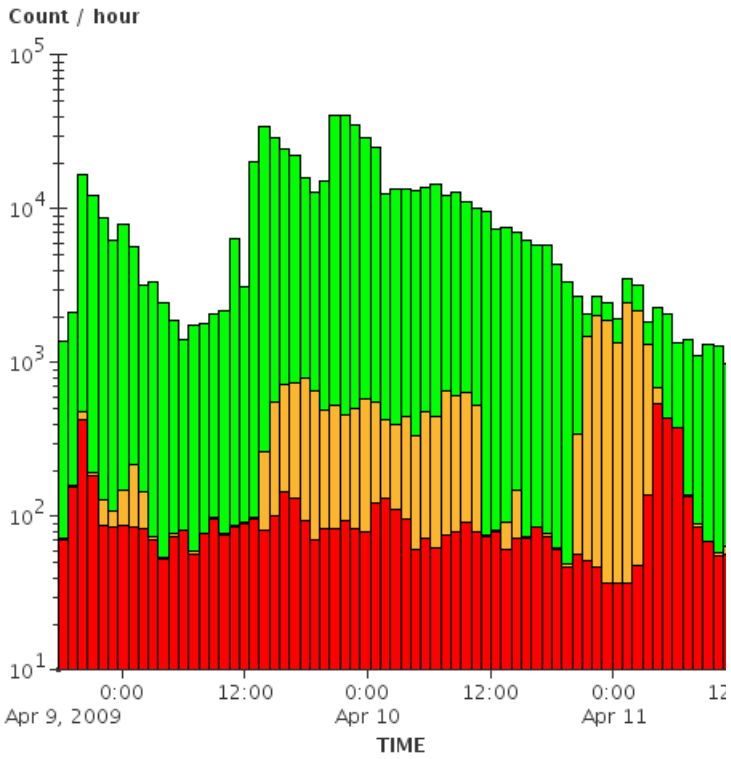
This is the expected behavior under a 40k load (we have reproduced the 40K transfers situation and have seen that there is no service degradation and postgresql does not consume all resources):



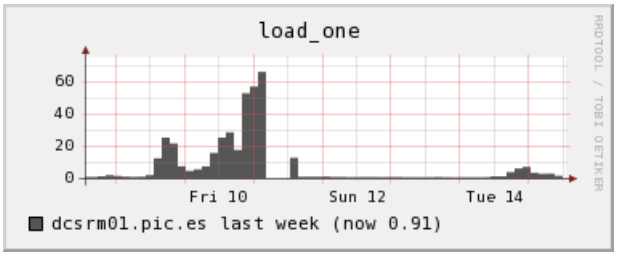


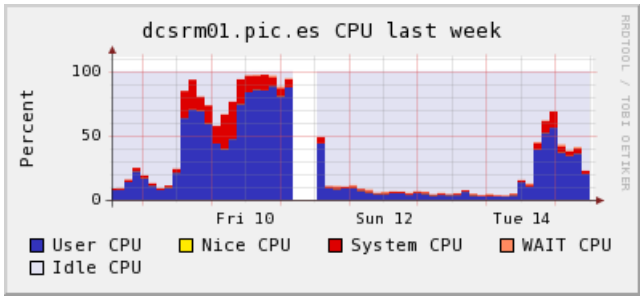
This is the behavior we got when srmVacuum was set to False:





CPU load/percent of the SRM server, including normal activity and abnormal activity periods for comparison.





We tried to check what exactly the srmVacuum flag is supposed to do via documentation at dcache.org/fermilab sites with no success. After asking dCache SRM developers, we found out the flag is simply issuing "VACUUM ANALYZE;" to SRM's postgresql data base every 6 hours.

- Esta página fue modificada por última vez el 12:53 15 abr, 2009.