Table of Contents

Throttling Xrootd **1**
Namespace Queries 1
Bandwidth 2
I/O Operations Per Second (IOPS) 3

Throttling Xrootd

Xrootd provides new capabilities to CMS end-users. They have a simpler, lower-latency mechanism to access sites directly. While this is a fantastic capability for scientists, sites need to plan accordingly to protect their storage from overload. This page discusses a few simple techniques.

Our advice is to keep a light touch when implementing throttles - sometimes throttles come at the cost of overall performance. Keep the system design such that the expected use is well below the throttle-imposed limits, and combine throttling with active monitoring of system health.

Namespace Queries

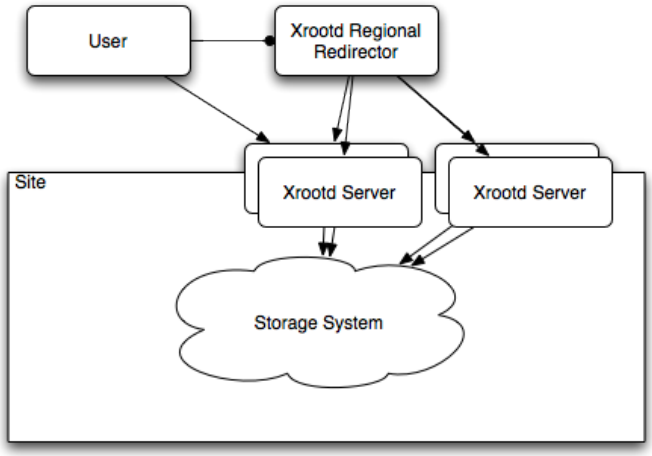
When the regional redirector searches for a file it hasn't encountered previously, it will query all sites subscribed to it to see if they have the file. This can be problematic in two cases:

1. "Runaway" user: whether by silly mistake or not, it is easier for users to perform many more queries than with SRM.
2. "Query Propagation": As the redirector queries all subscriber servers, then other peer redirectors, one user query can cause many resulting queries at a site with many Xrootd proxies.

Techniques to mitigate both are described in this section.

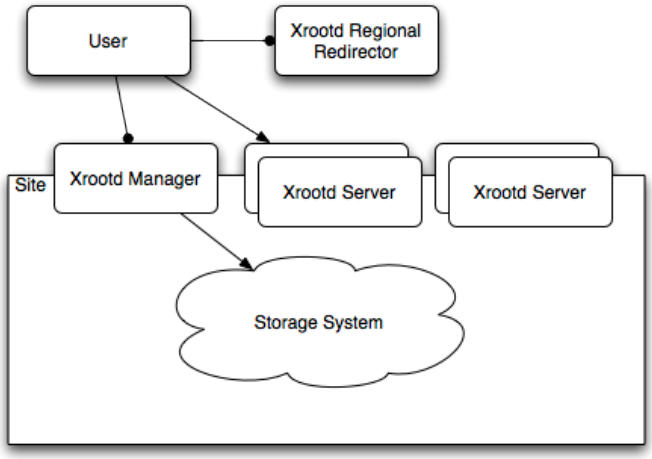
The 3.0.3 release of Xrootd comes with query throttles in the cmsd, covered here: http://xrootd.slac.stanford.edu/doc/dev/cms_config.htm, using the cms.dfs directive. The site can decide to rate-limit the number of queries performed by their cmsd and cause additional queries to queue up.

The default layout looks like this:



The site servers subscribe to the regional redirector directly and have a "backend" to the site shared filesystem. So, for each user query, a site

with 3 servers will see 3 queries to the shared filesystem. This setup, while simple and fault-tolerant, does not scale well with the number of Xrootd servers. Once a site adds a fourth server, we recommend using a manager node and use the `cms.dfs` directive. The `cms.dfs` directive allows large sites to add a separate manager in Xrootd hierarchy that queries the filesystem directly and assumes all the subscribed servers see the same shared filesystem. See the diagram below:



In this diagram, each of the site xrootd servers will use `all.manager` instead of `all.manager` in their xrootd.cfg. For each user query, there will only be one shared filesystem operation from the manager, regardless of the number of site xrootd servers. Large sites should consider adding a second manager node for fault tolerance.

A third useful throttle is the `mdhold` option of the `cms.dfs` directive. This option allows directory non-existence to be cached for a limited amount of time. Suppose a user queries for `/store/foo/bar` and `/store/foo` does not exist. Then, a subsequent query for `/store/foo/baz` will not cause a filesystem operation as the `cmsd` will recall that directory `/store/foo` does not exist.

Bandwidth

The `xrootd` daemon contains no direct bandwidth control. We believe bandwidth shaping should not be done by the daemon, but at a higher level. Two ways to do this are:

- Host-level: Use tc (http://linux.die.net/man/8/tc) to shape and schedule xrootd-based traffic on the host.
  - Sites running a 2.6.36 kernel can add the xrootd process to a cgroup so the limits are applied to the process, not a particular port.
- Network-level: Most modern managed switches can do flow shaping. This is the preferred solution, as the bottleneck is more often in the network than at the host level.

We recommend to distinguish the xrootd and gridftp traffic at your site and apply the appropriate policy.

Note: We are developing experimental support for bandwidth throttling in Xrootd; see this page for more information.

I/O Operations Per Second (IOPS)

In CMSSW-based workflows, IOPS is a greater concern than raw bandwidth. Many uses can cause a huge number of small reads, especially in older versions. Always recommend users adopt a 4_x release, preferably 4_2 or later.

There's no direct mechanism to throttle the IOPS, especially across many xrootd servers. However, one can explore using the `xrd.sched` directive (covered at http://xrootd.slac.stanford.edu/doc/prod/xrd_config.htm) to limit the number of worker threads that will perform simultaneous operations. Keep a light touch here - be aware that too draconian settings will effectively make your site broken in the view of users.

To some extent, remote-users are self throttling: as CMSSW is single-threaded a user across the continent can perform no more than 20 IOPS; a transatlantic user can do around 6 IOPS due to round trip times.

We suggest a reactive approach. You can monitor how much activity goes on through your site from our detailed monitoring collector.

Note: We are developing experimental support for IOPS and concurrency throttling in Xrootd; see this page for more information.

This topic: Main > XrootdThrottling
Topic revision: r5 - 2019-02-14 - DonataMelaikaite