

System level traffic shaping in disk servers with heterogeneous protocols

Eric Cano and Daniele Francesco Kruse

European Organization for Nuclear Research CERN, CH-1211 Genève 23, Switzerland

E-mail: eric.cano@cern.ch, daniele.francesco.kruse@cern.ch

Abstract. Disk access and tape migrations compete for network bandwidth in CASTORs disk servers, over various protocols: RFIO, Xroot, root and GridFTP. As there are a limited number of tape drives, it is important to keep them busy all the time, at their nominal speed. With potentially 100s of user read streams per server, the bandwidth for the tape migrations has to be guaranteed to a controlled level, and not the fair share the system gives by default. Xroot provides a prioritization mechanism, but using it implies moving exclusively to the Xroot protocol, which is not possible in short to mid-term time frame, as users are equally using all protocols. The greatest commonality of all those protocols is not more than the usage of TCP/IP. We investigated the Linux kernel traffic shaper to control TCP/ IP bandwidth. The performance and limitations of the traffic shaper have been understood in test environment, and satisfactory working point has been found for production. Notably, TCP offload engines' negative impact on traffic shaping, and the limitations of the length of the traffic shaping rules were discovered and measured. A suitable working point has been found and the traffic shaping is now successfully deployed in the CASTOR production systems at CERN. This system level approach could be transposed easily to other environments.

1. Introduction

1.1. CASTOR's disk staging and access patterns

The CERN Advanced STORAge manager (CASTOR) [1] is used to archive to tape the physics data of past and present physics experiments at CERN. The current size of the tape archive is more than 91 PB. In 2012 alone, about 30 PB of raw data were stored.

CASTOR organises data in files, which are stored in a disk staging area before migration to tape or after being recalled from tape. Each file is stored complete on one file system of one of the disk servers. The staging area is also used as a cache/work area where files are kept as long as possible. A garbage collector manages deletions. This double usage leads to a double access pattern, where the tape system competes with user access (interactive or batch processing) for accessing data in the disk servers.

A central scheduler, the stager, organises all data transfers to and from the staging areas. The tape system is scheduled with a “freight train” approach, where long lists of file transfers are sent to the tape servers for execution. The time delay between a scheduling decision and the actual transfer can be minutes. The precise location of the file on one disk server is pre-determined at scheduling time.

The stager limits the number of concurrent user accesses to disk servers with a slot system. But this does not guarantee efficient transfer from the user. Some experiment's frameworks



open many files during the whole analysis session, where the actual data transfer for a given file is just a burst in the middle of a long, mostly idle session. This led operators to choose a high slot quota, and disk servers can end up with around 300 user connections. With all of those active, tape migrations can go as low as 1 MB/s.

1.2. Competition between end users and tape system for disk server bandwidth

With the time delay between decision and actual transfer, the stager cannot guarantee an efficient data transfers for the tape server. Yet, an efficient tape transfer is fundamental as the tape drives are a scarce resource, shared by all users, and central for efficient migration to new tape media[2]. Disk pools are dedicated to a given experiment or task, so an inefficient access pattern has a controlled scope, however tape drives stuck waiting for data from disk servers have a global effect.

We therefore needed a mean to guarantee the bandwidth served to the tape drives and give them high priority. In addition, the disk servers contain a good number of disk spindles, but many of them are still connected to the network with a single 1 Gbit/s interface: the network bandwidth is then the resource being competed for.

Today's tape drives typically achieve 250 MB/s[3] [4]. The tape servers buffer files in memory, and read/write up to 3 files in parallel, if they are small enough, so we need to guarantee between $\frac{250 \text{ MB/s}}{3} = 83 \text{ MB/s}$ and 250 MB/s from any disk server. This will not be achieved fully with a single 1 Gbit/s interface, but with a fair share bandwidth as low as 1 MB/s, we had a long way to go.

The CASTOR disk servers are accessed using several protocols: RFIO, XRoot, root and GridFTP. Tape servers use RFIO, but so do some user accesses. RFIO has no means to reserve bandwidth for a given user. On the other hand the tape servers are dedicated machines. The IP addresses are hence an appropriate criterion to discriminate between tape servers and other accesses.

2. Traffic shaping as a mean to favour the tape servers

As no application level means, neither central nor disk server level can guarantee the tape bandwidth, we investigated the system level. All the CASTOR infrastructure runs on Linux [5]. The Linux kernel, beyond the classic IP chains, contains many powerful network control features, allowing detailed routing and traffic control [6]. The traffic control delays network packets on the output of an interface. As a consequence, this methodology only allows control for one direction out of two (migrations to tape, and not tape recalls), but securing the data to tape is the most important.

2.1. Traffic control objectives

We tried to guarantee 90% of the bandwidth to the tape servers in case of mixed disk and tape traffic. We also want to still provide all of the bandwidth to any class of traffic in the absence of the other.

2.2. Traffic control configuration

By default the Linux kernel uses a “priority FIFO” queuing discipline called (`pfifo_fast`). This queuing discipline prioritises the IP packets based on the type of service (TOS) bits. In order to retain a behaviour as close as possible to the system defaults, we used the priority (`prio`) queuing discipline, mimicking this default. The best effort TOS (which is where all the normal TCP traffic is) FIFO of this queuing discipline is in turn sent in a “class base queuing” queuing discipline or `cbq`. Our `cbq` configuration contains 2 classes; one taking all traffic by default, and getting 10% of the bandwidth (with possibility to borrow more from other idle classes).

A second class gets the 90% bandwidth. And finally a classifier assigns packets bound to tape servers to the second, high priority class. The classifier is a list of rules, which can select a single IP or a contiguous IP range.

In our case, there are no significant bidirectional data streams. So TCP packets carrying the ACK bit are usually small. Those packets are added to the high priority class in order to avoid penalizing incoming data transmission. Finally ssh protocol is also given high priority.

The detailed final configuration can be seen in appendix.

2.3. Test environment

The production environment does not provide a stable enough workload to provide reliable results. The workload was hence simulated with one machine identical to the production disk servers (with a single 1 Gbit/s interface) and 2 client computers simulating respectively the tape and user clients, with a variable number of streams for each. The kernel used is Linux 2.6.18-274.17.1.el5, the test disk server had dual Intel Xeon E51500 @ 2.00GHz processor, with a Intel 80003ES2LAN Gigabit card (one link out of 2 was used).

The three machines were dedicated to the test, and the bandwidth was measured through the increase of the bytes transfer statistics in `/proc/net/dev` over 2s.

3. Results

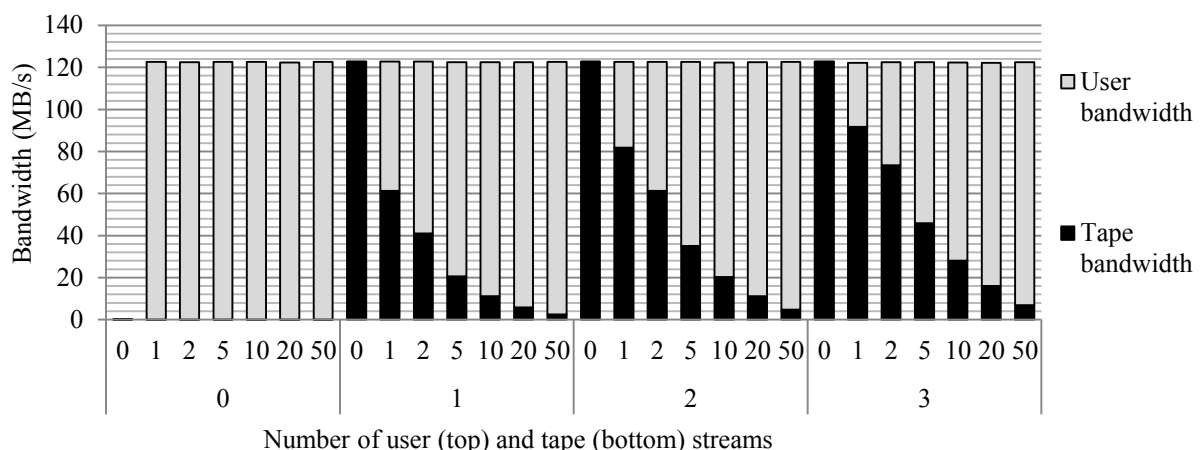


Figure 1. Default bandwidth distribution (stacked)

During tests, various parameters and queuing disciplines were tested. Most notably, both borrowing and limitation to 90% could not be achieved with other queuing disciplines.

Additionally, the usage of TCP segmentation offload (TSO) engine with the traffic shaper led to unstable bandwidth, and a bandwidth ratio close to the non-traffic shaped one. Turning off the TSO costs more in processing time: the kernel and CPU see packets of network MTU (standard Ethernet MTU of 1500 B in our case), and has to process and filter proportionality more packets. With the TSO on, a network capture showed packets of 64 kB.

The bandwidth ratio for various numbers of tape and user streams, for default set-up (where we get the expected fair share distribution), and with traffic control (TSO off) are shown in figures 1 and 2. The values are the averages over several runs of the test.

The result is not exactly the expected 90% ratio: with only one tape stream, we only reach a ratio of $\approx 65\%$, and with at least two, the ratio becomes a stable $\approx 84\%$.

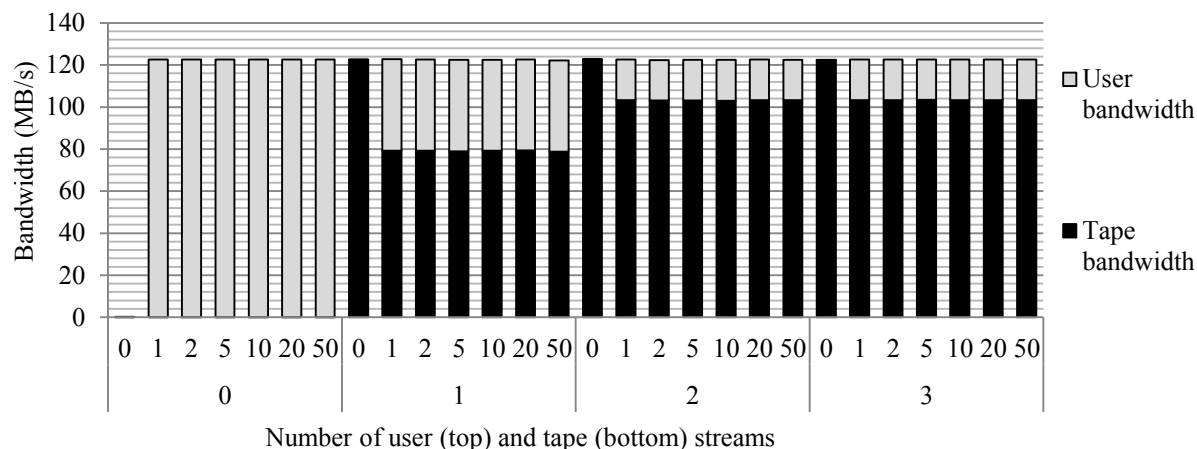


Figure 2. Traffic shaped bandwidth distribution (stacked)

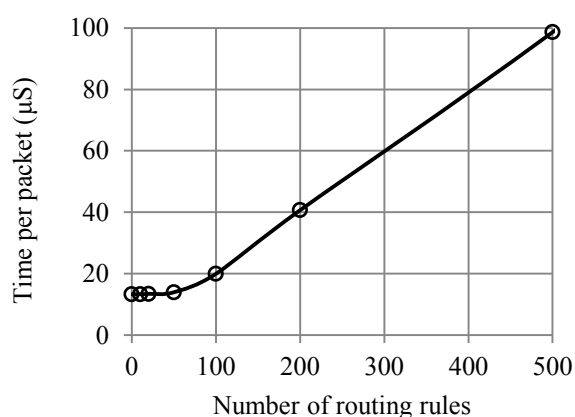


Figure 3. Time per packet against number of classification rules

The number of hosts listed in the tape host list also had an impact on performance, especially with a raised packet rate in the absence of TSO. We artificially increased the number of IP prioritization rules up to 2000 to measure their costs and assess the practical limits for the number of IP addresses. The average time per packet was deduced from the bandwidth measurements (making the assumption that the majority of packets were a full 1500 B). This measurement is shown on figure 3.

The measurements show an initial plateau where the time to process the classification rules is hidden by the time to send the packet over the wire ($\frac{1500 \text{ B}}{1 \text{ Gbit/s}} = 12 \mu\text{s}$). We then have a clean linear progression. The slope yields a processing time per rule of $\approx 200 \text{ ns}$. This

gives a budget of about 60 rules, not taking into account other processing overhead for packet processing. The measurements show that the limit is in our case closer to 50 rules at most.

With faster interfaces, the affordable number of rules decreases linearly, to the point of becoming impractical (about 5 only) for a 10 Gbit/s.

4. Deployment to production

At CERN, we had 122 tape servers at deployment time, 11 network range based rules were enough to discriminate tape traffic. The traffic control script as show in appendix has been turned into a system service and deployed on live systems. The traffic control rules can be turned on an off at any moment without disrupting the network data flow.

Long term logs of the tape activity showed that before applying the traffic shaping, many days saw a significant fraction (10 – 30%, sometimes more) of the data volume transferred to tape at less than 10% of the maximum speed. This fraction dropped to 0.5% on the worst days after applying the traffic shaping. No degradation of performance from the user's perspective was documented.

5. Conclusions

During a crunch in operations, the Linux kernel's powerful network control features proved invaluable to work around system overload. Some features had to be understood in a test environment, due to imprecise documentation, but the expected result was met.

This solution remains short term and only covers migrations to tape, but not recalls from tape. For the longer term, we are currently looking into additional tape staging area, allowing fully controlled scheduling, and distributed storage like a reliable array of inexpensive nodes (RAIN) like Ceph, where traffic would be evened out over many nodes, eliminating hotspots [7].

Appendix: the configuration script

```
1 # Turn off TCP segmentation offload: kernel sees the details
2 # of the packets routing
3 /sbin/ethtool -K eth0 tso off
4
5 # Flush the existing rules (gives an error when there are none)
6 tc qdisc del dev eth0 root 2> /dev/null
7
8 # Duplication of the default kernel behavior
9 tc qdisc add dev eth0 parent root handle 10: prio bands 3\
10     priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1
11
12 # Creation of the class based queuing
13 tc qdisc add dev eth0 parent 10:1 handle 101: cbq bandwidth 1gbit avpkt 1500
14 tc class add dev eth0 parent 101: classid 101:10 cbq weight 90 split 101:\
15     defmap 0 bandwidth 1gbit \
16     prio 1 rate 900mbit maxburst 20 minburst 10 avpkt 1500
17 tc class add dev eth0 parent 101: classid 101:20 cbq weight 10 split 101:\
18     defmap ff bandwidth 1gbit\
19     prio 1 rate 100mbit maxburst 20 minburst 10 avpkt 1500
20
21 # Prioritize ACK packets
22 tc filter add dev eth0 parent 101: protocol ip prio 10 u32\
23     match ip protocol 6 0xff\
24     match u8 0x05 0x0f at 0\
25     match u16 0x0000 0xffc0 at 2\
26     match u8 0x10 0xff at 33\
27     flowid 101:10
28
29 # Prioritize SSH packets
30 tc filter add dev eth0 parent 101: protocol ip prio 10 u32\
31     match ip sport 22 0xffff flowid 101:10
32
33 # Prioritize network ranges of tape servers
34 tc filter add dev eth0 parent 101: protocol ip prio 10 u32 match ip\
35     dst <Network1>/<bits1> flowid 101:10
36 tc filter add dev eth0 parent 101: protocol ip prio 10 u32 match ip\
37     dst <Network2>/<bits2> flowid 101:10
38 # <etc...>
```

Listing 1. Traffic control setting script

References

- [1] CASTOR homepage <http://cern.ch/castor>
- [2] Kruse D F 2013 *The Repack Challenge* Unpublished paper presented at CHEP 2013, Amsterdam
- [3] IBM enterprise tape drives <http://ibm.com/systems/storage/tape/drives/>
- [4] Oracle StorageTek T10000C Tape Drive <http://www.oracle.com/us/products/servers-storage/storage/tape-storage/t10000c-tape-drive/overview/index.html>
- [5] Scientific Linux CERN <http://cern.ch/linux/scientific.shtml>
- [6] Linux advanced routing and traffic control <http://www.lartc.org/lartc.pdf>
- [7] Weil S. A. 2007 Ceph: Reliable, scalable, and high-performance distributed storage <http://ceph.com/papers/weil-thesis.pdf>