Chapter 5:
Advanced usage of Rucio

Monitoring, Popularity, Rebalancing

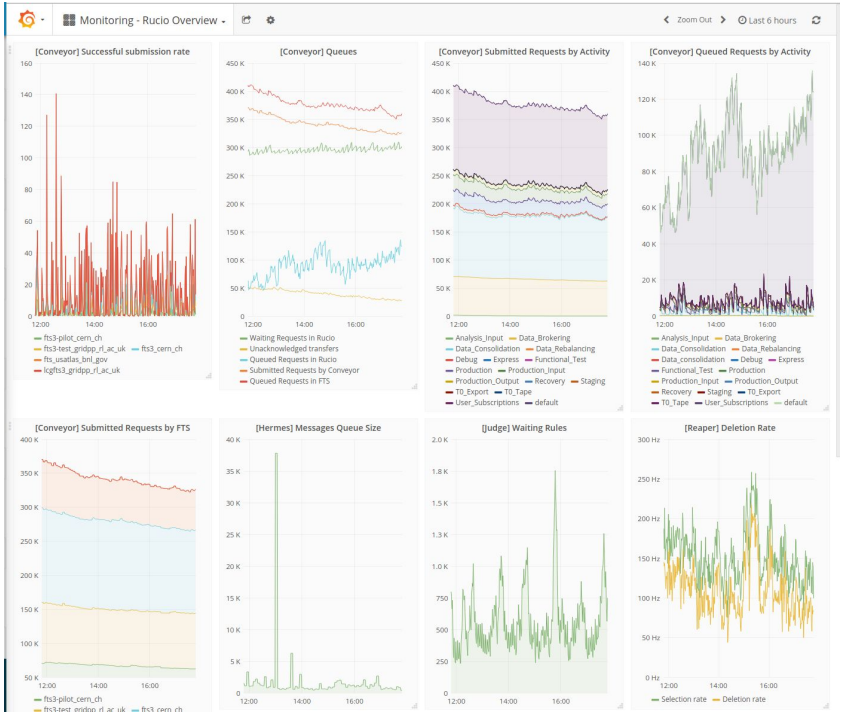
Overview

- Everything mentioned in this talk is optional and does not have to be used!

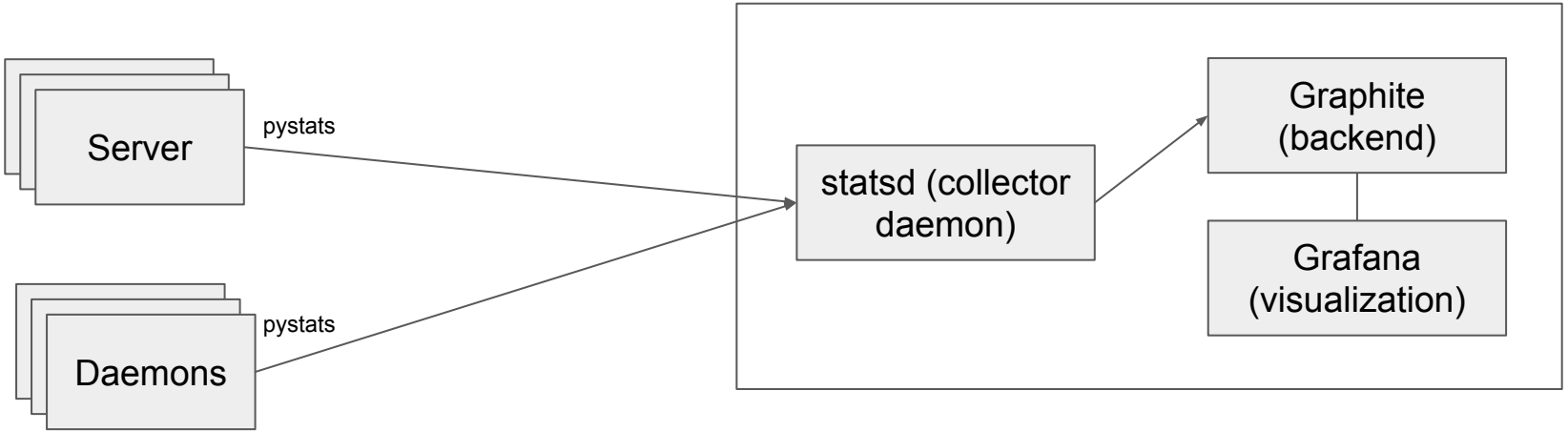
- Monitoring
  - Monitoring of internal metrics
  - Transfer/deletion monitoring dashboard
- Popularity
  - Popularity based deletion
  - Popularity based replication
- Data Rebalancing
  - Automatic rebalancing of data to free space on full sites

Internal Monitoring (Graphite/Grafana)

- Used to monitor internal performance metrics of Rucio servers and daemons, e.g., server response time, submission rates, queued messages, etc.
- Each Rucio component has built-in calls to send metrics to a central Graphite server
- Additionally there are probes that check subsystem states
- Two possible front-ends available
  - Built-in Graphite dashboards
  - Grafana dashboards (see picture)



Internal Monitoring Architecture



- Easy to set up
  - Docker images available both for Graphite and Grafana
  - Otherwise a manual install is also easily possible
  - Then only need to configure Rucio to point to the Graphite host and data will automatically start to flow in
  - Dashboard templates for Grafana can be provided

Events

- Way to generate messages in Rucio to communicate to other systems and for monitoring
- Produced by different subsystems, e.g.,
  - Transfers completed or failed by transfer daemon
  - Deletion completed or failed by deletion daemon
- Messages daemon takes car of distributing the messages
- Supports different types of delivery: e.g., ActiveMQ, email …
- ActiveMQ can be used as a buffer and to distribute the messages to different consumers

Example Event

t	payload.account	-
t	payload.activity	Data Rebalancing
#	payload.bytes	1,065,577
t	payload.checksum-adler	5c824f06
t	payload.checksum-md5	-
t	payload.dst-rse	BNL-OSG2_DATADISK
t	payload.dst-type	DISK
t	payload.dst-url	srm://dcsrm.usatlas.bnl.gov:8443/srm/managerv2?SFN=/pnfs/usatlas.bnl.gov/BNLT0D1/rucio/mc15_13TeV/dd/15/EVNT.07917213._203457.pool.root.1
#	payload.duration	12
t	payload.guid	-
t	payload.name	EVNT.07917213._203457.pool.root.1
t	payload.previous-request-id	-
t	payload.protocol	srm
t	payload.reason	
t	payload.request-id	7ecd6d9b5de64bb6b89ee03d4c4dfd56
t	payload.scope	mc15_13TeV
t	payload.src-rse	TAIWAN-LCG2_DATADISK
t	payload.src-type	DISK
t	payload.src-url	srm://f-dpm001.grid.sinica.edu.tw:8446/srm/managerv2?SFN=/dpm/grid.sinica.edu.tw/home/atlas/atlasdatadisk/rucio/mc15_13TeV/dd/15/EVNT.07917213._203457.pool.root.1
🕐	payload.started_at	February 27th 2018, 14:07:31.000
🕐	payload.submitted_at	February 27th 2018, 13:54:11.000
t	payload.tool-id	rucio-conveyor
t	payload.transfer-endpoint	https://fts.usatlas.bnl.gov:8446
t	payload.transfer-id	b6b9cb4c-ad94-506a-82ee-fb7ae492303b
t	payload.transfer-link	https://fts.usatlas.bnl.gov:8449/fts3/ftsmon/#/job/b6b9cb4c-ad94-506a-82ee-fb7ae492303b
🕐	payload.transferred_at	February 27th 2018, 14:07:43.000
t	tags	ites
t	type	transfer-done

Traces

- Client reports to monitor data access
- Rucio provides a central server endpoint to collect traces
- Clients can be easily instrumentalized to send traces without big impact on the client
- For ATLAS the traces are sent by
  - pilots for grid job accesses
  - Command line tools for local uploads/downloads
  - WebUI for downloads
- Server forwards the messages to ActiveMQ
- From there can be handled like events

Example Trace

t	appid	3846364849
#	catStart	1,519,741,991.094
t	clientState	DONE
t	dataset	data15_13TeV:data15_13TeV.00276336.physics_Main.merge.AOD.r7562_p2521_tid07709524_00
?	duid	⚠ -
t	eventType	get_sm_a
t	eventVersion	pilot3
t	filename	AOD.07709524._000022.pool.root.1
#	filesize	3.015GB
t	guid	FA5BE042BC79D447BA373BA4D404770A
t	hostname	wn001.rc.cuhk.edu.hk
t	ip	10.68.8.22
t	localSite	HK-LCG2_DATADISK
t	pq	ANALY_HK-LCG2_ARC
t	protocol	gfalcopy
#	relativeStart	1,519,741,991.096
t	remoteSite	HK-LCG2_DATADISK
t	scope	data15_13TeV
t	stateReason	OK
t	taskid	2
#	timeEnd	1,519,742,013.702
#	timeStart	1,519,741,975.974
t	traceId	b489496ad8ab41ed8ebcdb90833c2d9f
t	traceIp	::ffff:137.189.4.133
🕒	traceTimeentry	February 27th 2018, 15:34:08.406
#	traceTimeentryUnix	1,519,742,048.406
#	transferStart	1,519,741,991.096
t	type	rucio-trace
t	url	srm://se01.atlas.cuhk.edu.hk:8446/srm/managerv2?SFN=/dpm/atlas.cuhk.edu.hk/home/atlas/atlasdatadisk/rucio/data15_13TeV/f5/0e/AOD.07709524._000022.pool.root.1
t	usr	d5081bac388de4604adc174ae8fde8c7
t	usrdn	/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=gangarbt/CN=660595/CN=Robot: Ganga Robot/CN=proxy
t	uuid	32ad6bd6fe3f4b6e0d7c4c80d351c8aa
#	validateStart	1,519,742,008.99

Monitoring

- Transfer and deletion monitoring can build using open source technologies
- Events and traces can be streamed to Elasticsearch using tools like logstash
- From there they can be used for different use cases:
  - Dashboards for global transfer monitoring
  - Detailed transfer failure monitoring
  - Site monitoring
  - Popularity monitoring

Advanced Transfer/Deletion Monitoring

- Advanced workflow possible If the amount of events become too much to handle directly
- In ATLAS also extra information like topology needed
- ATLAS DDM dashboard based on online processing using Apache Kafka and Spark to aggregate data and enrich
- Written to InfluxDB which is faster for time series
- Grafana used for visualization

ATLAS DDM Dashboard (Latest data)

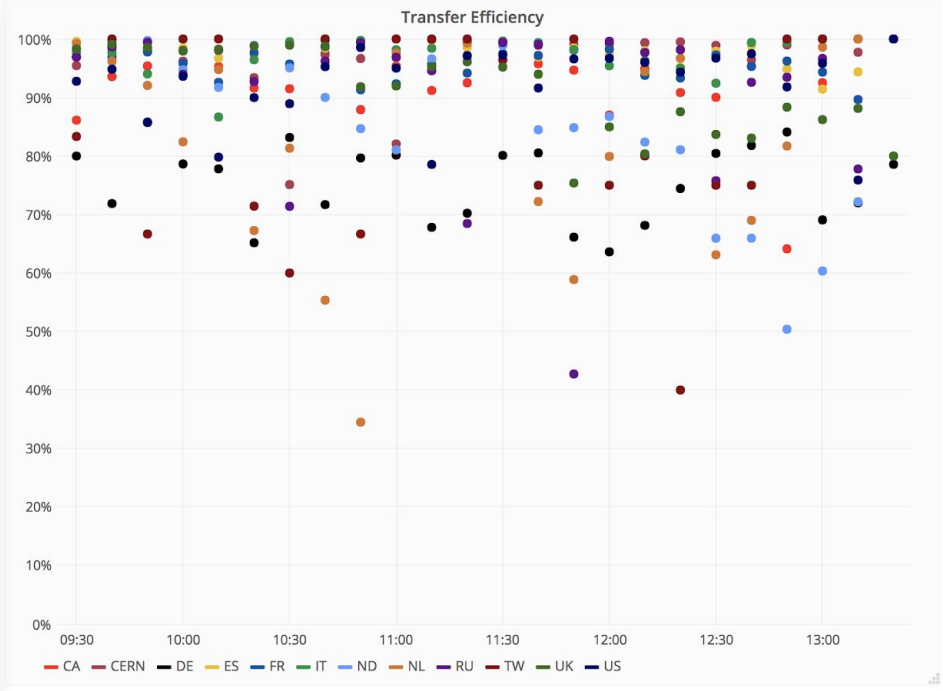
Zoom Out | Last 4 hours UTC

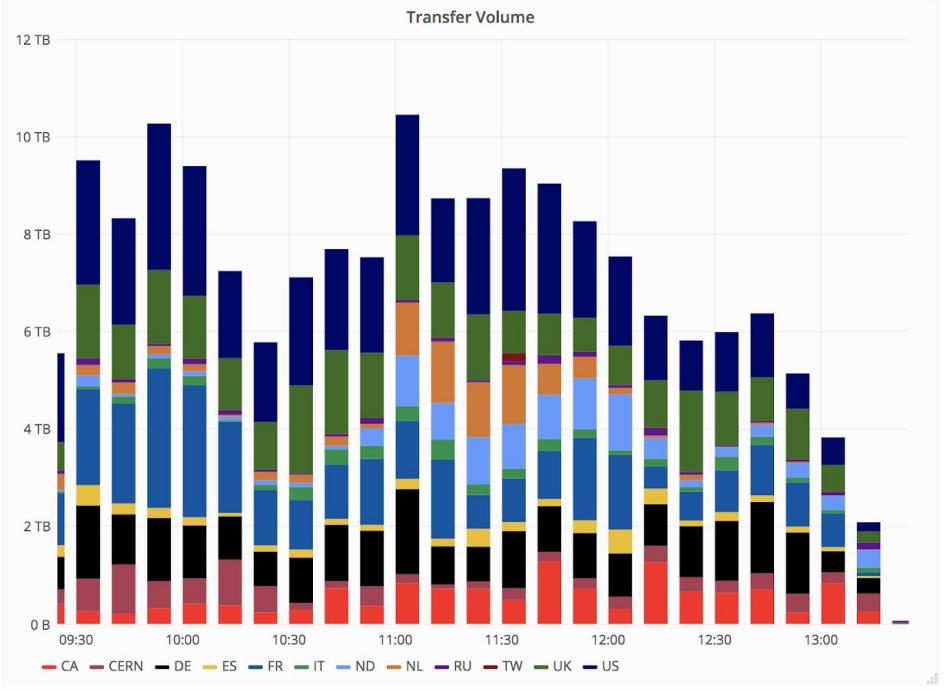
Group by dst_cloud | Binning auto

Activity: Analysis Input + Data Brokering + Data Consolidation + Data Rebalancing + Deletion + Express + Functional Test + Production Input + Production Output + Recovery + T0 Export + T0 Tape + User Subscriptions + default + Staging | Source country All

Source site All | Destination country All | Destination site All | Filters + | Matrix Columns dst_cloud | Matrix Rows src_cloud

Transfers





Popularity based deletion

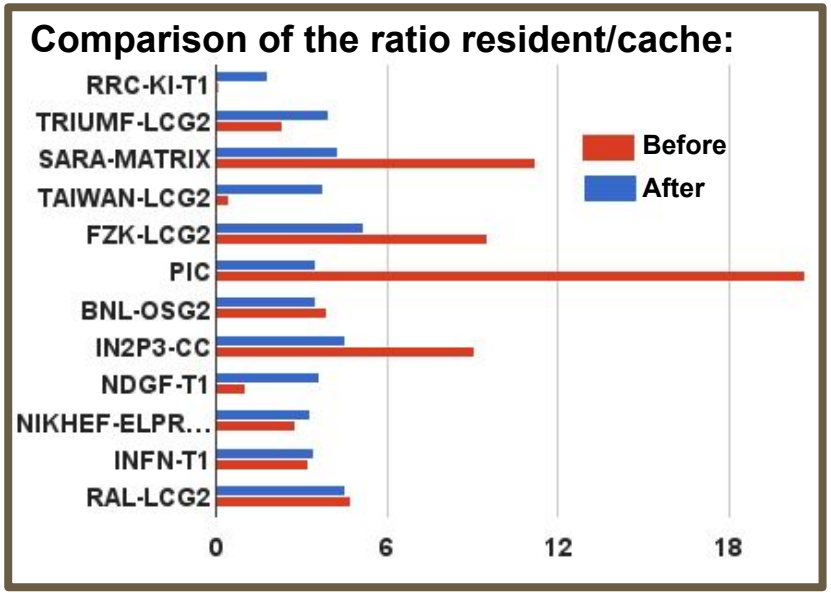
- Daemon available that reads traces from ActiveMQ
- Updates the last accessed date for file and dataset replicas
- When space is needed at site deletion agent ranks by last accessed data
- Remove least used replicas first
- I.e., a replica can stay at an RSE if continued to be used and enough space available

Popularity based replication

- Daemon to automatically create replicas of popular data
- Identifies popular data by scanning input dataset of new jobs
- Then can use different metrics to check if new replica could be needed:
  - Past popularity using traces
  - Already available replicas
  - Network bandwidth between possible source and destination sites
  - Free space
  - Computing resources
- Registered new rule with limited lifetime in Rucio

Automatic data rebalancing

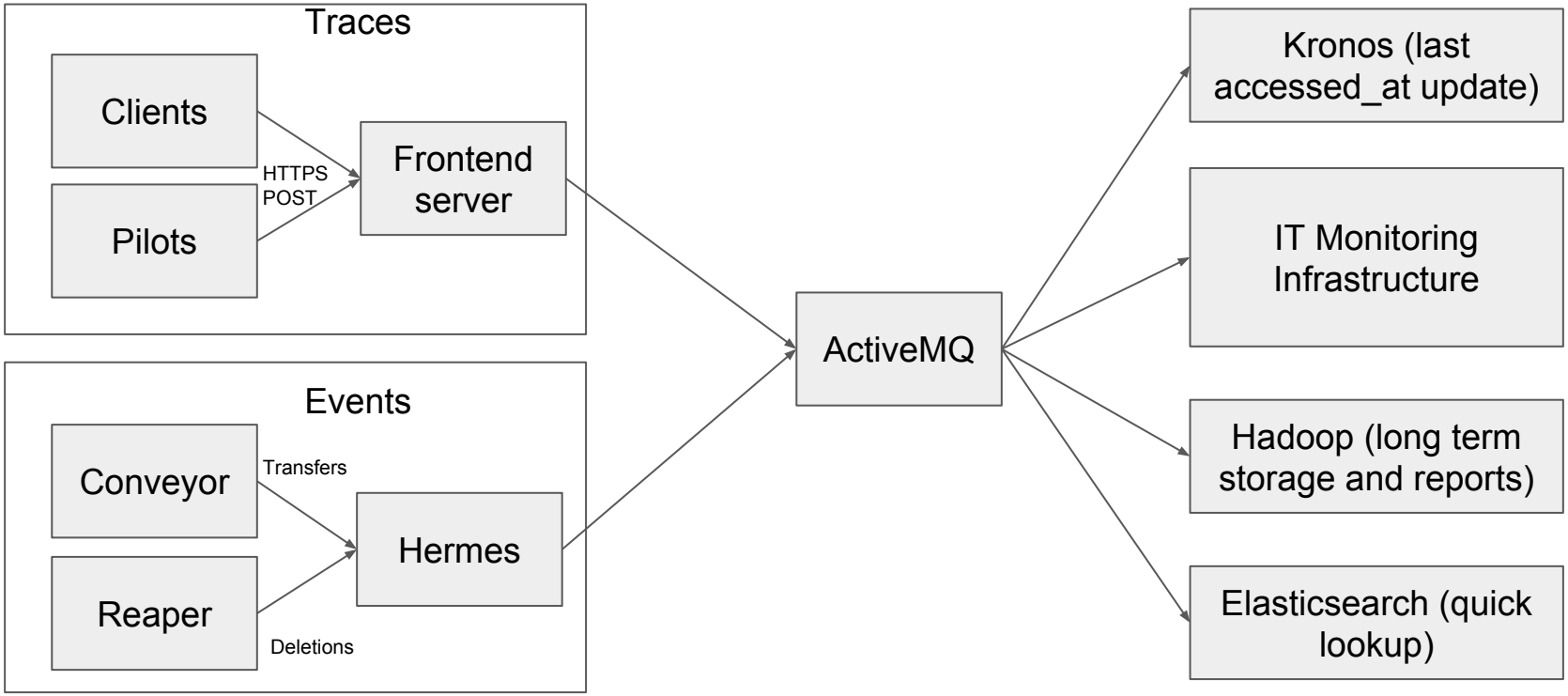
- Daemon to automatically rebalance data from RSEs that are running full to free RSEs
- Usually this is a cumbersome tasks but it is a good candidate for automation
- Data is selected based on policies, age, etc.
- Interesting metric to measure is the ratio between resident and cache data



Questions?

If you have a question but don't get the chance to ask it directly during the session, you can do it here: https://goo.gl/BdSGoC

Traces / Events architecture



Monitoring Architecture