Handling LSST data: experimenting an event-driven approach

Bastien Gounon

January 27, 2020

CCIN2P3

cnrs

Context : image simulation on the grid

- ImSim: a software package simulating LSST images
  - Takes astronomical catalogs as input
  - Outputs LSST-like FITS images

- Year 4 and 5 of the survey running in the European grid via DIRAC
  - UK & French sites participate in the simulation effort (~13 sites)

Making ImSim data available for processing

- Job outputs uploaded to CC-IN2P3 LSST dCache grid SE (~20TB available)
- Total estimated output data is ~100TB
- ImSim data then ingested through the LSST processing pipeline
- Processing jobs run at CC-IN2P3, so we need to:
  - Transfer all that data from dCache to our main distributed filesystem
  - Empty the dCache pool on a regular basis to make room for incoming ImSim data

Taking advantage of dCache events

- Legacy approach: manually execute bulk copy (using Globus for example)
  - Requires human intervention
  - Difficult to estimate copy duration

- Event-driven approach: take advantage of storage events for automation
  - No human intervention
  - Copies happen in real-time, as new files are uploaded to dCache

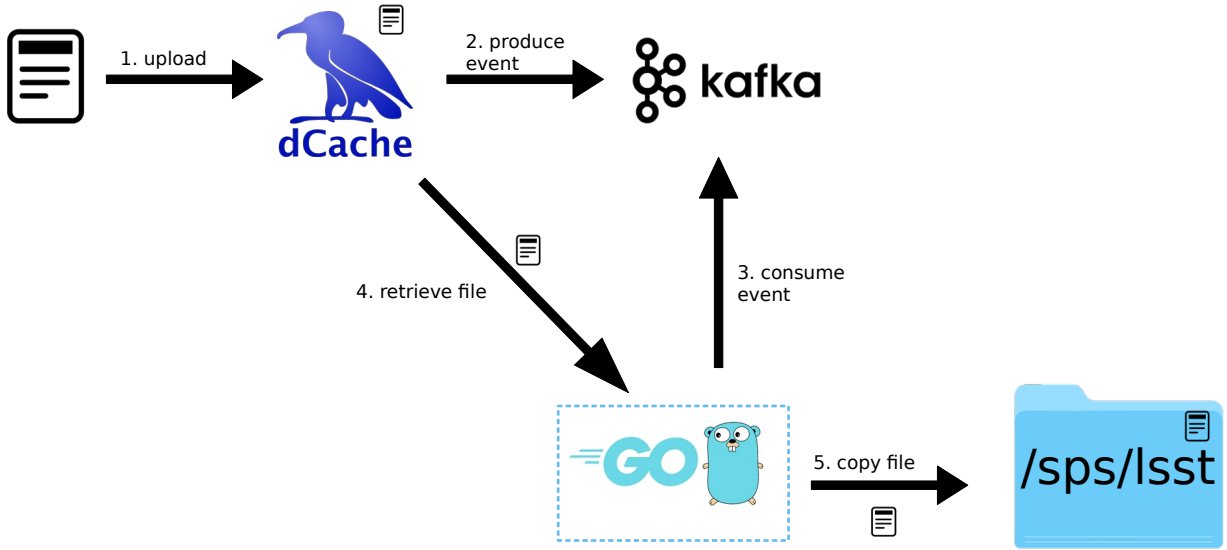
- Good opportunity to gain experience with dCache events and event-driven processing in general
- Using the existing Kafka cluster from our ESCAPE testbed

dCalqr

- Written in Go
- Source on IN2P3 GitLab: https://gitlab.in2p3.fr/bastien.gounon/dcalqr
- Available as a binary executable or a Docker container
- Goal: copy incoming dCache data to the main LSST storage for processing
- Designed for one specific use case
- Still in active development
- Portable and standalone executable, very easy to deploy and maintain
- Watch any local or remote dCache instance, from a server or laptop

What

- Consume messages from a Kafka topic
- Filter messages and identify uploads to a given directory
- Execute HTTP GET request from dCache (using x509_proxy authentication) and copy the file to a local directory



What


```
{
   "date":"2019-07-09T11:11:57.024+02:00",
   "owner":"/O=GRID-FR/C=FR/O=CNRS/OU=CC-IN2P3/CN=Adrien Georget",
   "msgType":"request",
   "clientChain":"134.158.240.106",
   "mappedGID":239,
   "cellName":"WebDAV-ccdcalitest12",
   "session":"door:WebDAV-ccdcalitest12@webdav-ccdcalitest12Domain:AAWNO/PLJUg:1562663516332000",
   "subject":[
      "UserNamePrincipal[ageorget]",
      "UidPrincipal[3915]",
      "LoAPrincipal[IGTF-AP:Classic]",
      "EntityDefinitionPrincipal[Person]",
      "FQANPrincipal[/dteam/NGI_FRANCE/sites/IN2P3-CC]",
      "GidPrincipal[239,primary]",
      "/O=GRID-FR/C=FR/O=CNRS/OU=CC-IN2P3/CN=Adrien Georget",
      "GroupNamePrincipal[lcgdteam,primary]",
      "Origin[134.158.240.106]",
      "FQANPrincipal[/dteam/NGI_FRANCE]",
      "FQANPrincipal[/dteam/NGI_FRANCE/sites]",
      "EmailAddressPrincipal[adrien.georget@cc.in2p3.fr]",
      "FQANPrincipal[/dteam,primary]"
   ],
   "transferPath":"/pnfs/in2p3.fr/data/doma/testWebdav",
   "sessionDuration":692,
   "storageInfo":"disk:doma@osm",
   "cellType":"door",
   "fileSize":119936222,
   "mappedUID":3915,
   "VERSION":"1.0",
   "queuingTime":0,
   "cellDomain":"webdav-ccdcalitest12Domain",
   "client":"134.158.240.106",
   "pnfsid":"00004F0AF7BEAE5A4CBEBF0FBF036D01F4C4",
   "billingPath":"/pnfs/in2p3.fr/data/doma/testWebdav",
   "status":{
      "msg":"",
      "code":0
   }
}
```

How

- Runs as a daemon on any machine
- Shopify/sarama Go client library for Kafka interaction
- Toml file for configuration
- Mostly built-in Go libraries for the rest
- Additional features :
  - Retry and report failed transfers
  - Adler32 checksum validation
  - Configurable maximum number of parallel transfers
  - Metrics logging (file size/throughput)
  - Keep track of the topic offset

How

```
[kafka]
topic = "billingEGEE"
servers = [ "172.17.0.37:9092", "172.17.0.111:9092", "172.17.0.63:9092",]
consumergroup = "imsimprod001"

[dcache]
url = "https://ccdcacli236.in2p3.fr:2880/lsst"
pool = "/pnfs/in2p3.fr/data/lsst"
folder = "/lsst/user/j/james.perry"
retryseconds = 1
retrytimes = 10
maxdownloads = 10
timeout = 600

[local]
proxy = "/pbs/home/l/lsstdata/software/dcalcr/x509_proxy"
capath = "/pbs/home/l/lsstdata/software/dcalcr/certificates"
downloadfolder = "/sps/lssttest/datasets/desc/DC2/Run2.2i/sim/input"
permissions = '2755'
failedtasksfile = "/pbs/home/l/lsstdata/software/dcalcr/failed.csv"
```

config file

How

```
2020/01/23 17:47:14 [UPLOAD] New file on partition 0 : /pnfs/in2p3.fr/data/lsst/lsst/user/j/james.perry/23003/23003691/fits_01188251_10.tar (f9c82897-0976-ed88-61d8-0db305308527)
2020/01/23 17:47:14 [CURRENTLY_RUNNING] 1 running tasks: [f9c82897-0976-ed88-61d8-0db305308527] (SYSTEM)
2020/01/23 17:47:15 [COPY_INIT] Creating destination file /sps/lssttest/datasets/desc/DC2/Run2.2i/sim/input/23003/23003691/fits_01188251_10.tar.part (f9c82897-0976-ed88-61d8-0db305308527)
2020/01/23 17:47:15 [COPY_START] Getting data from https://ccdcacli236.in2p3.fr:2880/lsst/lsst/user/j/james.perry/23003/23003691/fits_01188251_10.tar (f9c82897-0976-ed88-61d8-0db305308527)
2020/01/23 17:47:15 [COPY_RUN] Copying data to /sps/lssttest/datasets/desc/DC2/Run2.2i/sim/input/23003/23003691/fits_01188251_10.tar.part (f9c82897-0976-ed88-61d8-0db305308527)
2020/01/23 17:47:16 [COPY_OVER] https://ccdcacli236.in2p3.fr:2880/lsst/lsst/user/j/james.perry/23003/23003691/fits_01188251_10.tar to /sps/lssttest/datasets/desc/DC2/Run2.2i/sim/input/23003/23003691/fits_01188251_10.tar (checksum validation: OK) (f9c82897-0976-ed88-61d8-0db305308527)
2020/01/23 17:47:16 [COPY_REPORT] 217 MB copied in 379.387386ms, 4355.91 Mbps (f9c82897-0976-ed88-61d8-0db305308527)
```

logs

Scalability

- Easy horizontal scaling thanks to Kafka consumer groups
  - Kafka partitions are dynamically assigned to members of the group
  - Increase throughput by running one instance per data transfer node
  - Currently our dCache topic has a single partition, but we would like to move to multiple partitions

- Impact on dCache ?

Evolution

- Logging to ElasticSearch for advanced monitoring
- Automatic file deletion
- Templatize code to fit new workloads behind Kafka (write your own filter + the task to run for each message matched)

Issues encountered

- Hard-coded pause between message reception and HTTP GET to avoid returning a 404 error
- Requests periodically hang for a few minutes, before successfully completing all at once. Reason not identified yet, may be related to dCache load ?
- Request queue seems to fill up even though the client is limited to 10 simultaneous connections: more debug needed client side

Impressions

- >400K files and >50TB of data copied over 7 weeks, left unattended for most of the time
- Low failure rate since retry implementation (< 1%)
- No major issue with dCache nor Kafka

Conclusion and questions

- Need to explore FaaS platforms such as Apache Openwhisk
  - Pros: flexibility, run any code, centralized
  - Cons: initial configuration, maintaining the service
- Looking for more use cases to reuse our experience with dCache storage events for other usecases around Kafka

Thank you!