CERN openlab

Developing an EOS Data Transfer Monitoring

CERN openlab Summer Student Lightning Talk session

Srobona Ghosh

Supervisor : Cédric Caffy

14 / 09 / 2022

Introduction

Project Topic

- Analyze, implement and put in place an EOS data transfer monitoring tool
  - extract the amount of files/bytes read/written from the EOS report log
    - for a specific period of time
    - per protocol
    - per EOS instance

Parts of the Project

1. Creation of the entire project workflow from an eosreport to generation of dashboards
2. Replay of history for all eosreports from 1st January 2022
3. Deployment of an automated task to run the statistics every day

EOS and EOS Report Logs

EOS

- Distributed disk-based storage system
- Stores vast amounts of physics data and user files
- Manages 700 PB of raw storage
- Delivers 2.5 EB to CERN experiments per year

EOS Report Log Files

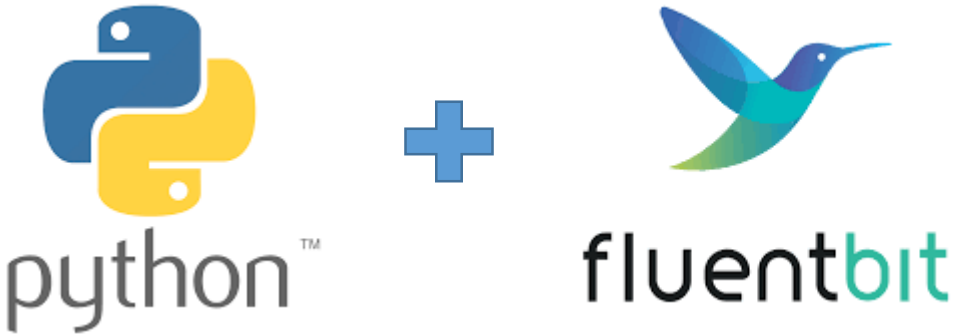
- Generated by each EOS headnode under
  - /var/eos/report/<YEAR>/<MONTH>/<YEAR><MONTH><DAY>.eosreport
- Provides information
  - files/bytes read/written/created/updated/deleted
- Used for data analysis, problem investigation and security forensics.
- Each line of the report logs looks like this:

log=583657ac-e2bf-11ec-ab9c-a4bf0179653a&path=/eos/lhcb/user/r/roneil/MightyTB/2020octoberdesy/extern/mupix8_daq/firmware/pcie/ip_compiler_for_pci_express-library/pciexpx8f_confctrl.v&fstpath=/data16/0001de4c/48fb9f64&ruid=1&rgid=1&td=1.115610:1482@st-096-gg561yoq&host=st-096-dd905d40.cern.ch&lid=1048834&fid=1224449892&fsid=12484&ots=1654207199&otms=132&cts=1654207199&ctms=171&nrc=1&nwc=0&rb=225344&rb_min=225344&rb_max=225344&rb_sigma=0.00&rv_op=0&rvb_min=0&rvb_max=0&rvb_sum=0&rvb_sigma=0.00&rs_op=0&rsb_min=0&rsb_max=0&rsb_sum=0&rsb_sigma=0.00&rc_min=0&rc_max=0&rc_sum=0&rc_sigma=0.00&wb=0&wb_min=0&wb_max=0&wb_sigma=0.00&sfwdb=0&sbwdb=0&sxlfwdb=0&sxlbwdb=0&nfwds=0&nbwds=0&nxlfwds=0&nxlbwds=0&ot=0.355&rt=74.23&rvt=0.00&wt=0.00&osize=225344&csize=225344&delete_on_close=0&prio_c=2&prio_l=4&prio_d=1&forced_bw=0&ms_sleep=0&sec.prot=sss&sec.name=eos&sec.host=eos&sec.vorg=-&sec.grps=-&sec.role=-&sec.info=-&sec.app=eos/drain&tpc.dst=st-096-gg561yoq.cern.ch&tpc.src_lfn=/replicate:48fb9f64

EOS Report Logs Parsing, Aggregating and Generating the Statistics

Problem

- Read the EOS report
- Extract statistics
  - Files/bytes read/written
    - per EOS instance
    - per protocol (xrootd, http…)
    - Per day/month/year



Solution Used

Python script+fluent-bit process

- The Python script
  - reads the eosreport file
  - applies the filters
  - parses into key-value pairs
  - aggregates the data by performing summation
  - outputs the aggregated data to the eosreports_statistics.txt file
- The fluent-bit-data-sending process
  - fluent-bit
    - fast and lightweight log processor, stream processor and forwarder
    - has in-built filters and pre-configured parsers
    - allows backpressure handling and temporary buffer storage
  - add the “eos instance” to the data coming from the eosreports_statistics.txt file
  - sends that data to the buffer influxDB database for persistence

Persistence Of The Statistics: Storing The Aggregated Data In A Database

Problem

- Storing the statistics and the aggregated data efficiently
- Should a database be used?
- Types of databases to be used?

Solution Used

- **InfluxDB**
  - purpose-built for storing and quickly performing analysis on large volumes of time-series data
  - a timestamp identifies a single point in any given data series
  - primary key is pre-set to time always
- Two InfluxDB databases in the same instance were used
- influxDB-datatransfer-buffer
  - contains every row coming from the fluent-bit-data-sending process
  - "buffer database"
    - queries are ran on this database to aggregate and send data to the influxDB-datatransfer-prod database
- influxDB-datatransfer-prod
  - final production database
    - containing the total amount of files/bytes read/written by
      - each eos instance
      - each protocol
      - each day
    - will be queried by Grafana to display the plots

Visualizing The Aggregated Data

Problem

- Visualizing the data in a nice monitoring setup
- How the data visualization should look like?
- Visualization tool is to be used: Grafana or Kibana?

Solution Used

- **Grafana**
  - open-source analytics and interactive visualization tool
  - supports many data sources
  - provides charts, graphs, and alerts for the web
  - allows users to create complex monitoring dashboards using interactive query builders
- vastly used as a visualization tool for monitoring tasks at CERN IT.
- InfluxDB can be easily used as a data source

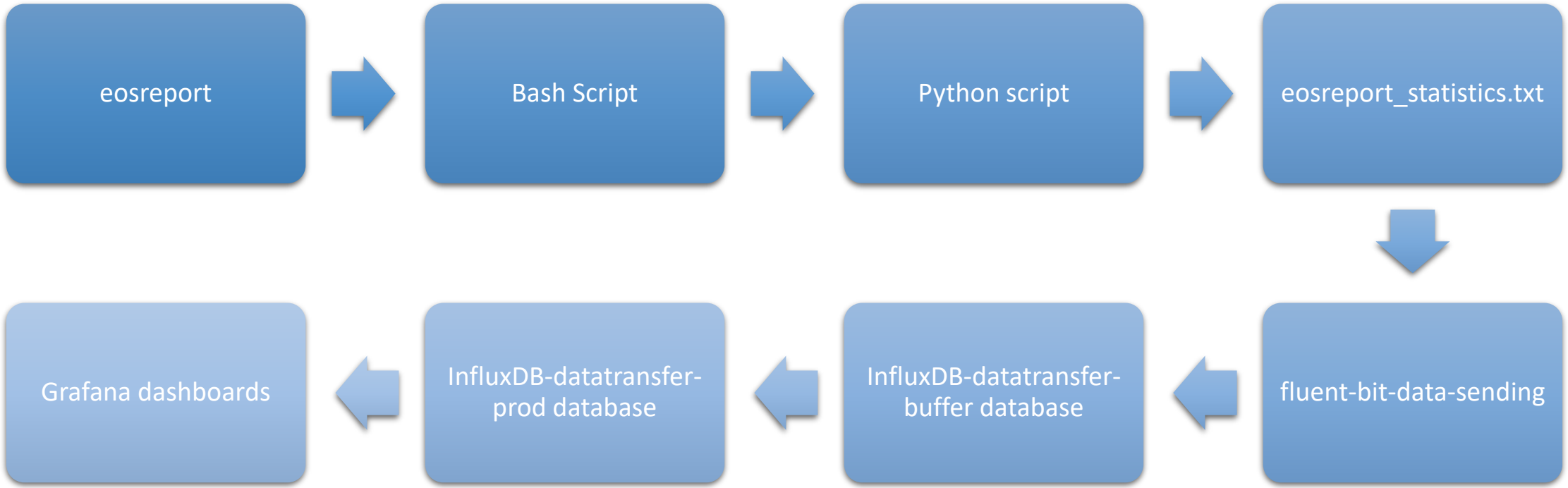
Replaying The Historical Data From 1st January 2022

Most important part of this project!

- Backlog ingestion of the data: Generation of the statistics of all the eosreports from 1st January 2022.
- A **bash script** is used to automate this history replay.
- It takes eos instance, a year and/or a month and/or a day as inputs.
- The bash script
  - copies the zipped eosreport file
    - of each day
    - each headnode
    - each instance
  - extracts it
  - provides it to the python script
    - to execute over the eosreport file
    - to do the calculation and aggregation
    - to append it to the text file that is read by fluent-bit-data-sending process
  - deletes the copied unzipped eosreport file after the Python execution is over

Summary

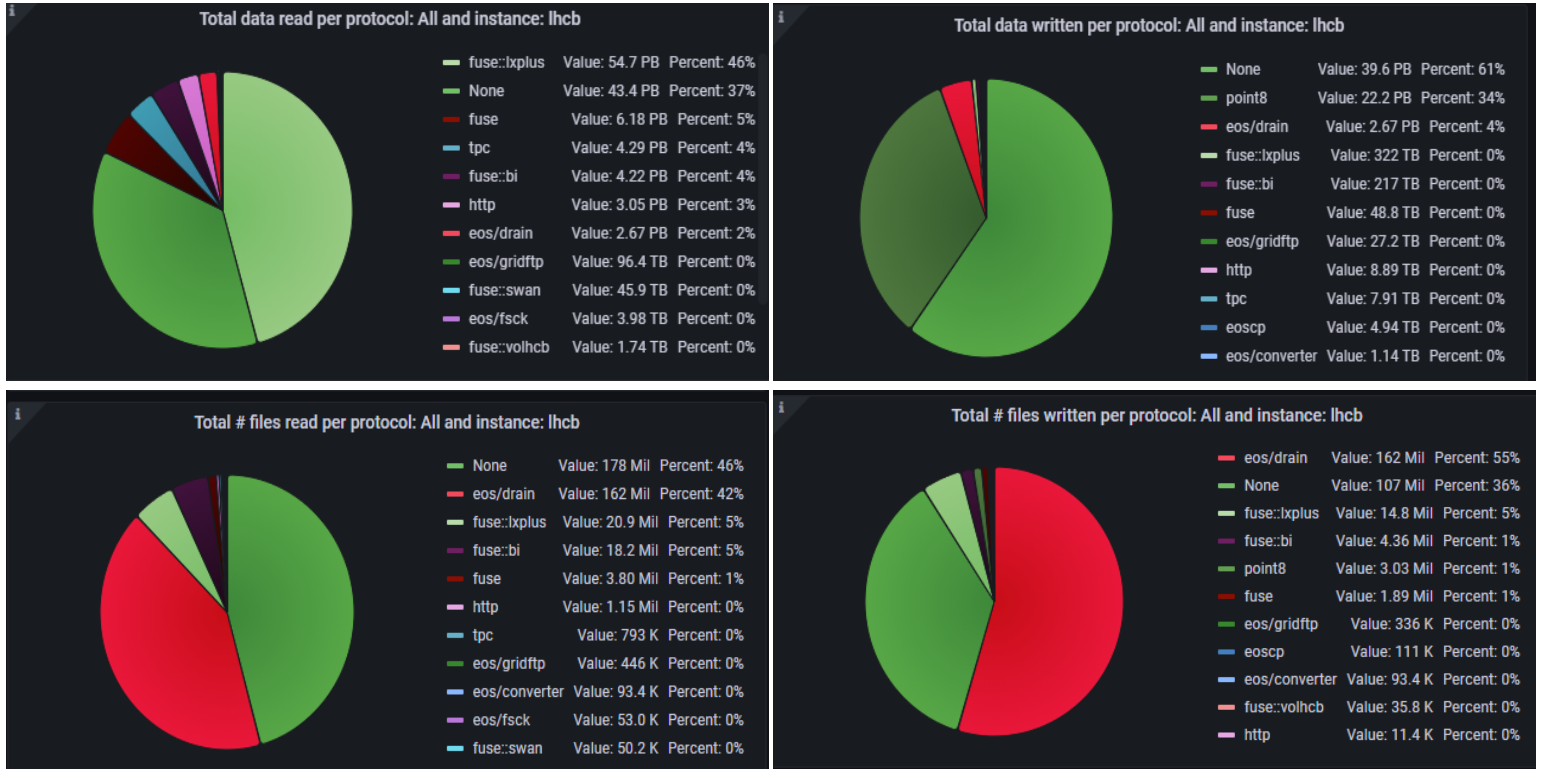
The entire process pipeline from one eosreport file to building Grafana dashboards.



Results

…and some snaps!

- Files used in this project can be found in this GitLab link here
- The entire workflow was implemented
  - an eosreport file➡ Grafana dashboards
- Replay of the historical data from the beginning of 2022 was successfully done
- Grafana dashboards are created
  - keeping protocol and eos instance as variables
  - the amount of files/bytes read/written,
    - per eos instance
    - per protocol (xrootd, http…)
    - per day



Pie-charts generated from the eosreport files of the EOSLHCb instance for all protocols from the 1st of January 2022 until the 9th of August 2022

Conclusion

…and what needs to be done!

- The aim of that project
  - create the EOS Data Transfer Monitoring tool that will allow the EOS operators to see, in one click and look
    - the amount of files/bytes read/written in an EOS instance within a specific period of time.
- Technologies used to accomplish this task
  - Python
  - Bash
  - fluent-bit
  - InfluxDB
  - Grafana
- This monitoring setup can be
  - automated and deployed in production to obtain the required statistics of the current day eosreport files
    - using a cron job
  - this part of the project has not been done yet.
- Finally, this monitoring tool could be
  - distributed as an RPM to deploy all the related scripts and configuration files in production on an EOS headnode
    - using the Puppet configuration management tool
  - this may be done in the future

References

- https://indico.cern.ch/event/1040723/contributions/4371444/attachments/2259661/3839856/FluentBit-ASFD-10-06-2021.pdf
- https://en.wikipedia.org/wiki/Grafana
- https://docs.fluentbit.io/manual/
- https://docs.influxdata.com/influxdb/v2.0/
- https://grafana.com/docs/
- https://monit-grafana.cern.ch/d/000000036/eos-control-tower?orgId=22&refresh=5m
- https://home.cern/science/computing/storage
- https://gitlab.cern.ch/dss/eos

QUESTIONS?

sg85@iitbbs.ac.in