dc_inject.py

Data Challenge 24 Workshop

2023-06-09

Mario.Lassnig@cern.ch

Lessons from DC21

- Injecting extra transfers on top of regular experiment traffic
  - is easy
  - is difficult

- It's easy, because
  - We know what we want to do
  - Rucio takes care of the hard work, it's just adding transfer rules
  - There's already some data lying around that can be transferred

- It's difficult, because
  - bash scripts with while loops, parsing CSV, calling CLIs, etc… are not very robust
  - Selection of data to transfer via `grep` and `vim` is … very manual
  - There's a lot of links and injecting with `rucio add-rule` is slow, easily causing storage overflows
  - Required operator attention too high

- Can we improve the injection?
  - Under the assumption that our target metric is the **1h mean throughput**
  - Multiple experiments are using Rucio … one injection tool to rule them all

dc_inject.py

- Available at https://gitlab.cern.ch/atlas-adc-ddm/dc_inject
  - Please submit your improvements!
- Address experiences from DC21
  - Wave-like injection pattern
  - Getting rid of transferred data
  - Rate attenuation
  - Universal chaos

```
usage: dc_inject.py [-h] [--injection-interval INJECTION_INTERVAL]
                    [--rule-lifetime RULE_LIFETIME] [--big-first]
                    [--fudge-factor FUDGE_FACTOR]

Inject data transfers into Rucio to match average Mbps/hour

optional arguments:
  -h, --help            show this help message and exit
  --injection-interval INJECTION_INTERVAL
                        Injection interval in seconds (default=900, 1...3600).
  --rule-lifetime RULE_LIFETIME
                        Rule lifetime in seconds for immediate purging
                        (default=7200, 1...86400)
  --big-first           Inject big DIDs first, otherwise smaller DIDs go first
                        (default)
  --fudge-factor FUDGE_FACTOR
                        Increase the injected amount by this percentage, given
                        as a float, to account for universal chaos (default=0,
                        0...1)
```

How does it work?

- Retrieve unique datasets per source
- Create link configuration
- Run the tool!
  - nohup & `tmux` are useful

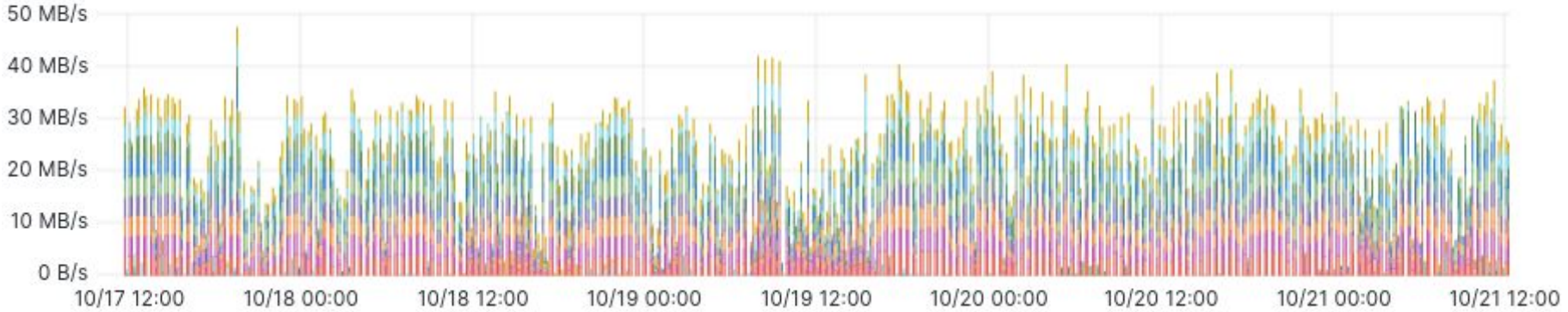
```
SELECT scope || ':' || name || ',' || bytes
FROM (
SELECT a.scope, a.name, a.bytes, count(*)
FROM atlas_rucio.dataset_locks a,
(
SELECT scope, name, bytes, ROUND(bytes / length) AS avg_file_size
FROM atlas_rucio.dataset_locks
WHERE rse_id = atlas_rucio.rse2id('&1')
AND state = 'O'
AND bytes > 0
AND length BETWEEN 1 AND 1000
AND bytes / length> 100000000
) b
WHERE a.scope = b.scope
AND a.name = b.name
GROUP BY a.scope, a.name, a.bytes
HAVING count(*)= 1
);
```

```
$ cat config.csv | shuf -n 25 | sort
AGLT2_DATADISK,BNL-OSG2_DATADISK,20
BNL-OSG2_DATADISK,AGLT2_DATADISK,20
BNL-OSG2_DATADISK,IN2P3-CC_DATADISK,20
BNL-OSG2_DATADISK,INFN-T1_DATADISK,20
BNL-OSG2_DATADISK,TRIUMF-LCG2_DATADISK,20
CERN-PROD_DATADISK,IN2P3-CC_DATADISK,40
CERN-PROD_DATADISK,NDGF-T1_DATADISK,40
CERN-PROD_DATADISK,TRIUMF-LCG2_DATADISK,40
DESY-ZN_DATADISK,FZK-LCG2_DATADISK,20
INFN-ROMA1_DATADISK,INFN-T1_DATADISK,20
INFN-T1_DATADISK,INFN-ROMA1_DATADISK,20
INFN-T1_DATADISK,RAL-LCG2-ECHO_DATADISK,20
INFN-T1_DATADISK,TRIUMF-LCG2_DATADISK,20
MWT2_DATADISK,BNL-OSG2_DATADISK,20
NDGF-T1_DATADISK,FZK-LCG2_DATADISK,20
PIC_DATADISK,BNL-OSG2_DATADISK,20
PIC_DATADISK,NDGF-T1_DATADISK,20
RAL-LCG2-ECHO_DATADISK,INFN-T1_DATADISK,20
SARA-MATRIX_DATADISK,BNL-OSG2_DATADISK,20
SWT2_CPB_DATADISK,BNL-OSG2_DATADISK,20
TRIUMF-LCG2_DATADISK,FZK-LCG2_DATADISK,20
TRIUMF-LCG2_DATADISK,INFN-T1_DATADISK,20
TRIUMF-LCG2_DATADISK,RAL-LCG2-ECHO_DATADISK,20
UKI-LT2-QMUL_DATADISK,RAL-LCG2-ECHO_DATADISK,20
UKI-NORTHGRID-MAN-HEP_DATADISK,RAL-LCG2-ECHO_DATADISK,20
```

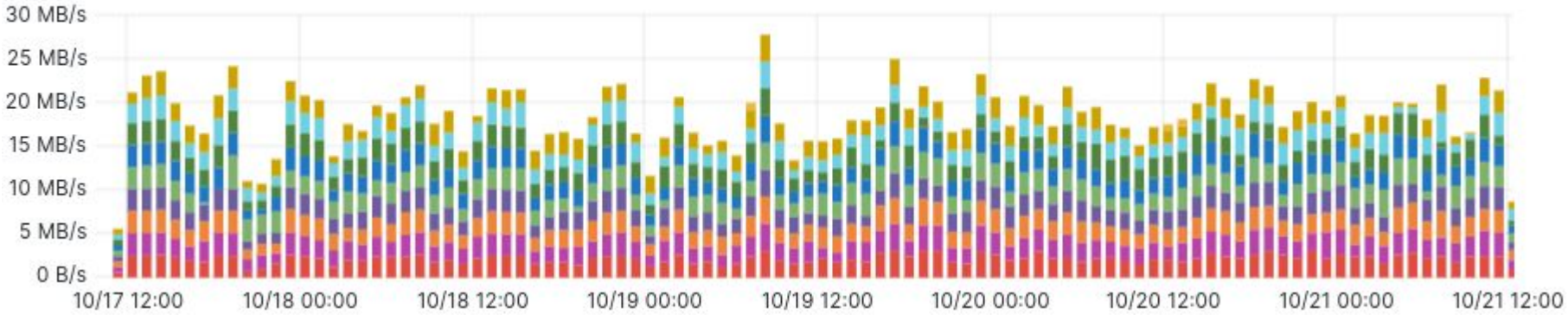
```
$ cat CERN-PROD_DATADISK.lst | shuf -n 5 | sort
mc16_13TeV:mc16_13TeV.302344.MadGraphPythia8EvtGen_A14NNPDF23LO_HVT_Agv1_VcWH_lvqq_m0800.deriv.DAOD_HIGG5D2.e7778_e5984_s3126_r10201_r10210_p4310_tid27018629_00,762627958
mc16_13TeV:mc16_13TeV.410084.MadGraphPythia8EvtGen_A14NNPDF23LO_ttgamma80_noallhad.deriv.DAOD_HIGG5D3.e4418_e5984_s3126_r10724_r10726_p4613_tid32714701_00,53487622097
mc16_13TeV:mc16_13TeV.506194.MGPy8EG_Zee_FxFx_3jets_HT2bias_CFilterBVeto.deriv.DAOD_JETM6.e8382_e7400_s3126_s3136_r10724_r10726_p5037_tid31518039_00,1613745557451
mc16_13TeV:mc16_13TeV.520920.MGPy8EG_A14N23LO_EWpMSSMRun2_EWFilt_0_102245.deriv.DAOD_SUSY1.e8435_e7400_a875_r10724_r10726_p3990_tid33332684_00,406230222
mc23_13p6TeV:mc23_13p6TeV.515016.MGPy8EG_A14N23LO_HNL12p5_ctau0p1_mumumu.deriv.DAOD_PHYS.e8529_e8528_s4159_s4114_r14799_r14811_p5855_tid34907741_00,259937706
```

First trial run :: 10 Mbit T0->T1

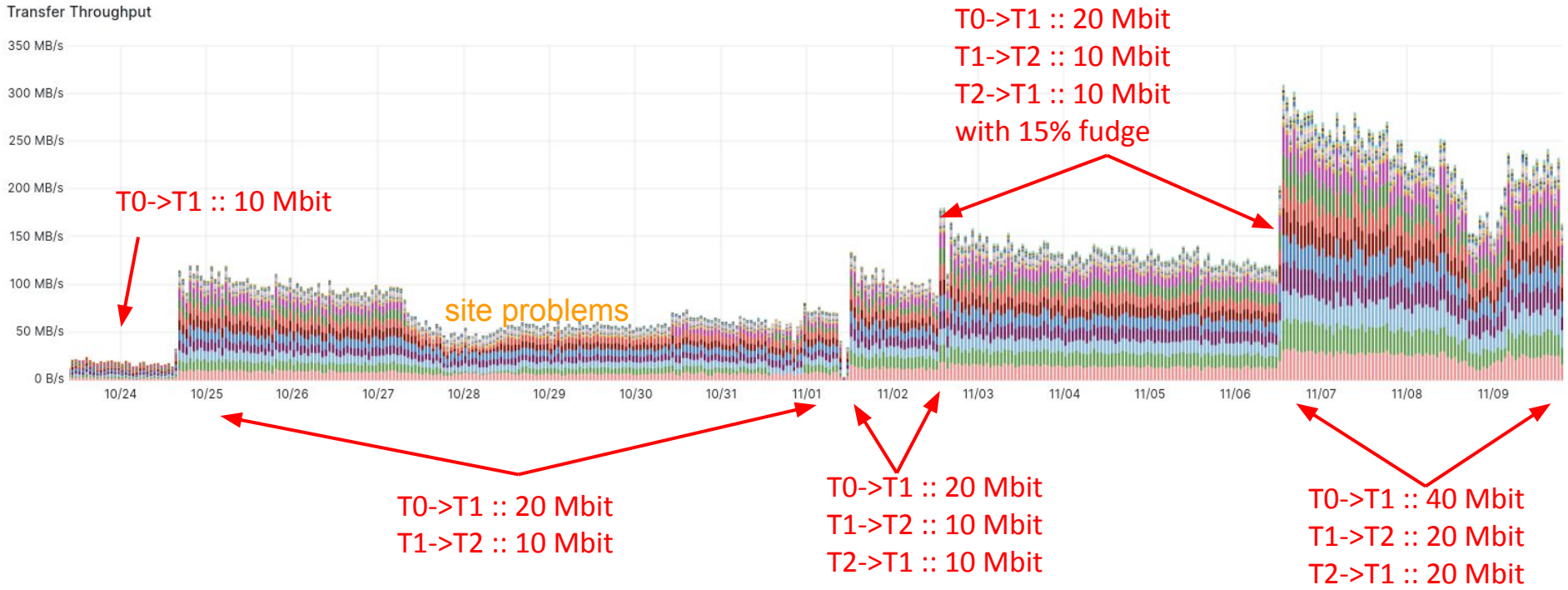
Transfer Throughput (10 min bins)



Transfer Throughput (1h bins)



More configuration variations



Last but not least

- First observations
  - Throughput decay noticeable on sites with not enough uniques
    - Need to reduce rule lifetime
  - Fudge factor
    - Useful if average dataset size is not quite right for the wanted throughput
    - Not useful if you only have large datasets, won't be selected anyway
  - Rate configuration is static
    - Needs operators to have a look every now and then, configure the "top-up" rate, and restart the tool
    - but we could dynamically get it from MONIT! Will be there for DC24
  - Some improvements necessary w.r.t. reuse of datasets across restarts
    - To reduce "rate catch-up" time
- dc_inject.py
  - It's not a good name :-)
  - Not even AI can create a logo for it!
  - If you have suggestions, let us know!