Rucio Pig
Use Cases & Examples

IT Analytics WG
10.12.2014

Data sources

- Apache Server / Rucio Daemon logs
  - read directly from log file and continuously streamed via Flume to HDFS
  - simple text log files
- Traces
  - send to ActiveMQ broker and continuously streamed via Flume to HDFS
  - text file with one JSON encoded dictionary per trace
- Oracle Dumps:
  - daily Sqoop dumps of most important tables to HDFS
  - bz2 compressed, tab-separated text files

Daily Data volume

- Logs: ~ 23 GB
- Traces (depends on load):
  - 6.000.000 entries
  - ~5 GB
- DB dumps:
  - DIDs: 550.000.00 entries
  - Rules: 7.500.000 entries
  - Replicas: 690.000.000 entries
  - Dataset Locks: 8.000.000 entries
  - RSEs: 700 entries
  - Total Volume: ~16GB bz2 compressed

Use cases

- log files:
  - storage and simple cat / grep operations
  - (log file analysis)
- traces:
  - update of last access time of files/datasets
  - popularity reports
- DB dumps:
  - daily reports for operations / site admins for consistency checks
  - file replicas / unique files per storage endpoint
  - primary / custodial dataset replicas
  - number of replicas per dataset / last access times

Example

- Generate a list of unique replicas of a file for all storage endpoints
  - Filter all files which only have one replica
  - split for (non-) deterministic storage endpoints
  - for the deterministic storage endpoints generate the path with a UDF
  - merge everything together again
  - join with rse table to get storage endpoint names
  - store back onto HDFS in multiple files. Split per storage endpoint

Input

```
rses = LOAD '/user/rucio01/dumps/$CURRENT_DAY/rses/part-m-*.bz2' USING PigStorage('\t')
AS (
 id: chararray,
 rse: chararray,
 rse_type: chararray,
 deterministic: chararray,
 volatile: chararray
);

replicas = LOAD '/user/rucio01/dumps/$CURRENT_DAY/replicas/part-m-*.bz2' USING
PigStorage('\t') AS (
 scope: chararray,
 name: chararray,
 rse_id: chararray,
 bytes: long,
 state: chararray,
 lock_cnt: long,
 adler32: chararray,
 created_at: chararray,
 accessed_at: chararray,
 path: chararray
);
```

Filter non-unique replicas

```
-- group per file and count the number of replicas
group_reps = GROUP d_reps BY (scope, dsn);
count_reps = FOREACH group_reps GENERATE group.scope, group.dsn, d_reps,
COUNT(d_reps) as num_reps;

-- filter out all non-unique replicas
filter_unique = FILTER count_reps BY num_reps == 1;

-- there is only one entry left in bag, so flatten it
flatten_reps = FOREACH filter_unique GENERATE FLATTEN(d_reps);

-- just for convenience
unique_reps = FOREACH flatten_reps GENERATE d_reps::rse_id as rse_id,
d_reps::scope as scope, d_reps::dsn as dsn, d_reps::checksum as checksum,
d_reps::fsize as fsize, d_reps::creationdate as creationdate, d_reps::path
as path;
```

Generate path for non-deterministic RSEs

```
-- get all replicas on deterministic rses
filter_det = FILTER unique_reps BY (path is null);

-- there shouldn't be any, but better safe than sorry
filter_dsn_scopes = FILTER filter_det BY (dsn is not null and scope is not
null);

-- create the path from scope and dsn with udf
get_path = FOREACH filter_dsn_scopes GENERATE rse_id, scope, dsn,
checksum, fsize, creationdate, rucioudfs.GETPATH(scope, dsn) as path;


-- get all replicas on non-deterministic rses
filter_nondet = FILTER unique_reps BY path is not null;

-- now all replicas have a path, so put them together again
union_det_nondet = UNION get_path, filter_nondet;
```

GETPATH UDF

```
public class GETPATH extends EvalFunc<String>
{
    public String exec(Tuple input) throws IOException {
        if (input == null || input.size() == 0)
            return null;
        try{
            String scope = (String)input.get(0);
            String name = (String)input.get(1);
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(scope.concat(":").concat(name).getBytes());
            byte[] digest = md.digest();

            String md5_1 = String.format("%02x", digest[0] & 0xff);
            String md5_2 = String.format("%02x", digest[1] & 0xff);

            String corrected_scope = scope;
            if (corrected_scope.startsWith("user") || corrected_scope.startsWith("group")) {
                corrected_scope.replace(".", "/");
            }
            return corrected_scope.concat("/").concat(md5_1).concat("/").concat(md5_2).concat
("/").concat(name);
        }catch(Exception e){
            throw WrappedIOException.wrap("Caught exception processing input row ", e);
        }
    }
}
```

Get RSE names and store

```
-- read in the rses
d_rses = FOREACH rses GENERATE id as rse_id, rse;

-- join replicas and rses to get the rse name
join_reps_rses = JOIN union_det_nondet BY rse_id, d_rses BY rse_id;

-- generate the final output schema
joined_output = FOREACH join_reps_rses GENERATE d_rses::rse,
union_det_nondet::scope, union_det_nondet::dsn, union_det_nondet::
checksum, union_det_nondet::fsize, union_det_nondet::creationdate,
union_det_nondet::path;

-- store on disk, split result into one file per rse
STORE joined_output INTO 'reports/$CURRENT_DAY/unique_replicas_per_rse'
USING org.apache.pig.piggybank.storage.MultiStorage
('reports/$CURRENT_DAY/unique_replicas_per_rse', '0', 'bz2', '\\t');
```

Overall volume and

- Input:
  - replica table: ~12 GB
  - rse table: 15 KB
- Output:
  - ~600 bz2 files
  - ~9 GB
- Runtime:
  - ~30 minutes

Ad-hoc analysis (last week)

Find set difference between two large tables

- both tables
  - ~600 million rows each, 2x11GB bz2 (2x90GB raw)
- sqoop dump from Oracle
  - ~4 hours, highly skewed parallelism
- time to write Pig script with full outer join and validate on sample
  - ~20 minutes
- time to run Pig script on full dataset
  - ~10 minutes

Obsolete use case: DQ2 Accounting

Same workflow

- retrieve pre-cooked dump from Oracle
  - 4GB, 20 minutes
- run dump through 35 different Pig scripts to generate different summaries and reports
  - 2GB, 30 minutes
  - regex_extract major source of CPU load
  - optimising regex increased runtime down from 6 hours