Mobile Investigation Workstation (MIW)

Emmanuel Benazera emmanuel.benazera@xplr.com
SEEKS SAS / XPLR Software Inc, 36 rue Jacques Babinet,
31100 Toulouse France

1. Description

Steria faces very large ($> 50G$ per day) amounts of proxy log collections from which to draw useful reports. Fast generation of user reports requires fast processing and indexing of logs. This documents briefly describes and documents a proof of concept for processing the logs and building a searchable index.
The purpose of the system is to first collapse large amounts of logs into smaller amounts (i.e. by aggregating log records for every user, visited domain name, ...). Second, the system indexes those collapsed logs in such a manner as to make queries of the index very fast (i.e. below the second).
The built system comprehends:

- A configurable log format for specifying new logs as inputs to the system;
- C++ map-reduce code (see details below) for heavily multithreaded log collapsing and indexing;
- A pre-configured Apache Solr Cloud instance with four shards, acting as the search engine itself.

2. Architecture

The architecture is built on map-reduce (MR) for log processing and Apache Solr for building a distributed and scalable search engine.

2.1 Map Reduce with Metis

The architecture uses a two step process built as MR jobs:

- log collapsing;
- log indexing.

Both jobs have been implemented in C++ with Metis (`http://pdos.csail.mit.edu/metis/`). Metis takes full advantage of multi-core machines, thus leading to very high performance on a single server (though it doesn't scale horizontally across several machines).

2.2 Big Data search engine with Solr

Apache Solr Cloud is a high performance scalable search engine. It has been preconfigured for using a four shards index (i.e. an index split in four), with the ability to scale across machines.

3. Setup & Compiling from sources

C++ code is available from git repository:

```
git clone gitolite@xplr.com:miw_steria
```

(Access requires an SSH key registered with XPLR servers.)
A pre-configured Solr Cloud tarball is available from the sources above as well, follow the instructions below in order to setup a new system.

3.1 Compiling MIW

```
cd miw_steria
./configure
make
```

3.2 Setting up Solr

In miw_steria

```
tar xvjf solr-4.4.0.tar.bz2
cd solr-4.4.0
./setup.sh
```

See the Usage section for using the Solr search engine.

4. Usage

Below are the two steps

4.1 Log collapsing

Assuming that the logs are located in /home/steria/logs

```
cd miw_steria
./apps/log_compacter /home/steria/logs/* -b -o logs_out.json -j \
  -f format/yourformat -n yourapp
```

This collapses every log file in reposity /home/steria/logs using yourformat and produces a JSON file of collapsed logs, ready to be indexed with Solr. The 'yourapp' parameter tags the logs for a particular application. This is useful to mix logs from different sources and origins into a single index, and later retrieve them.
The collapsing scheme is controlled by the log format definition. See below for details.
Use

```
./app/log_compacter
```

to get the full list of options.

4.2 Log collapsing while preserving original logs

This is for collapsing logs while preserving the original data and linking them for later access from the search engine

```
./apps/log_compacter /home/steria/logs/* -b -o logs_out.json -j \
  -f format/yourformat -n yourapp -d
```

The resulting file is now larger than the original. When indexed (see sections below), this data is then automatically compressed, and can be retrieved without loss.

4.3 Tips & Performances for collapsing

This MR job runs in memory. The required memory by each MR job is directly proportional to the size of the log file it processes. For this reason it is recommended to test the log collapsing program on files of different sizes, then split the largest log files accordingly.
A convenient way to split file is to use the split program on Unix systems:

```
split -l 1500000 log1
```

will split the log1 file into smaller files of 1.5M lines each.
Another way to lower the memory trace is to limit the number of map jobs, as follows:

```
cd miw_steria
./app/log_compacter /home/steria/logs/* -b -o logs_out.json -j -m 10
```

This will tell the program to use 10 mapping jobs. Note that the output remains exactly the same indpendently of the number of mapping jobs.

4.4 Writing & Configuring log formats

Formats are held in the miw/formats repository.
To create a new format, write a JSON file describing the log fields, see domain_controller_format.json as an example. Every described fields can be attached a few parameters for guiding the log collapsing operation.

Description of the available parameters:

- key: true if field should be part of the log key. The logs are collapsed for fields that bear the exact same key combination;
- aggregated: true if field values should be aggregated. For an exact same key, other fields can be aggregated;
- aggregation: union, sum, max, count. Aggregation operation controls the aggregated fields value, typically for union of IP addresses, and sum or max of transferred bytes for example;

- processing: day, month, year, hour, minute, second. Processing allows to only retain part of a date field. Typically, we may want to use the hour from a date field as a key for collapsing all logs within every hour and compute / unionize values across other fields such as IP addresses or amount of transferred data;
- date_format: generic specification of log date format, such as %d/%m/%Y;
- preprocessing: special formats, such as *evtx* can be pre-processed in order to generate new fields that are not originally encoded as CSV.

4.5 Starting the search engine

If the Solr is already running, skip this step.
To start Solr Cloud:

```
cd solr-4.4.0
./jetty.sh start 4
```

To stop Solr Cloud:

```
cd solr-4.4.0
./jetty.sh stop 4
```

To get a count of the number of elements in the index:

```
cd solr-4.4.0
./count_results.sh
```

To erase the index:

```
cd solr-4.4.0
./clear_index.sh
./clear_index.sh
```

Note the need to call clear_index.sh twice.

4.6 Log indexing

Indexing the log is achieved with a MR job in order to easily maximize the capability of the server:

```
cd miw_steria
./app/solr_commit logs_out.json -m 10
```

This will index the collapsed logs into Solr.
It has been observed that the program can put Solr under too much stress and thus generate failures. For this reason, the program reports the number of successful calls.
In order to limit the stress on the Solr indexer, it is recommended to limit the number of mapping jobs. Thus the example above uses a 10 mapping jobs call.

4.7 Search & Queries

Once some data have been indexed, queries can be issued.
Examples of queries:

```
curl "http://localhost:8984/solr/collection1/select?q=url:*google.com*&rows=0"
```

reports records of users having visited any URL from google.com.

```
curl "http://localhost:8984/solr/collection1/select?q=username:hal&rows=0"
```

reports records of user hal.

```
curl "http://localhost:8984/solr/collection1/select?q=username:hal \
        &date=2012-02-03T00:00:00+2DAYSrows=0"
```

reports records of user hal between the 3rd and 5th of February 2012. See `http://lucene.apache.org/solr/api-4_0_0-BETA/org/apache/solr/schema/DateField.html` for more information on how to query date fields with Solr.

```
curl "http://localhost:8984/solr/select?q=appname:yourapp&wt=json&rows=0"
```

reports the number of records for logs in application 'yourapp'.
It is possible to delete the content of the app 'yourapp' from the index, as follows:

```
curl "http://localhost:8984/solr/update?commit=true" \
--data '<delete><query>appname:yourapp</query></delete>' \
-H 'Content-type:text/xml; charset=utf-8'
```

The other applications in the index remain unchanged.
Note that Solr supports the service of queries while indexing. Therefore in production, new logs can be indexed without disrupting service.

4.8 Access to original data, from search results

When collapsing logs while preserving the original data (see section 4.2), and indexing them, it is possible to retrieve these data from the results to a search query. The steps are as follows:

- Issue a search query;
- From results, look up the *id* field of the records of interest;
- To get all content related to the id value, search for the string id value + _ *content*:

  ```
  curl "http://localhost:8984/solr/select?q=id:stringid_content&wt=json"
  ```

This returns a result with field *content* in the form of an array that contains all original data from which the collapsed log was produced. These data are only stored and not indexed.

Important Notes:

- the size of the content field data can range from mega to giga bytes;
- the size of the content field data can exceed the default output buffer of Solr's Web server;
- the size of the content field can exceed the system's in-memory capacity.