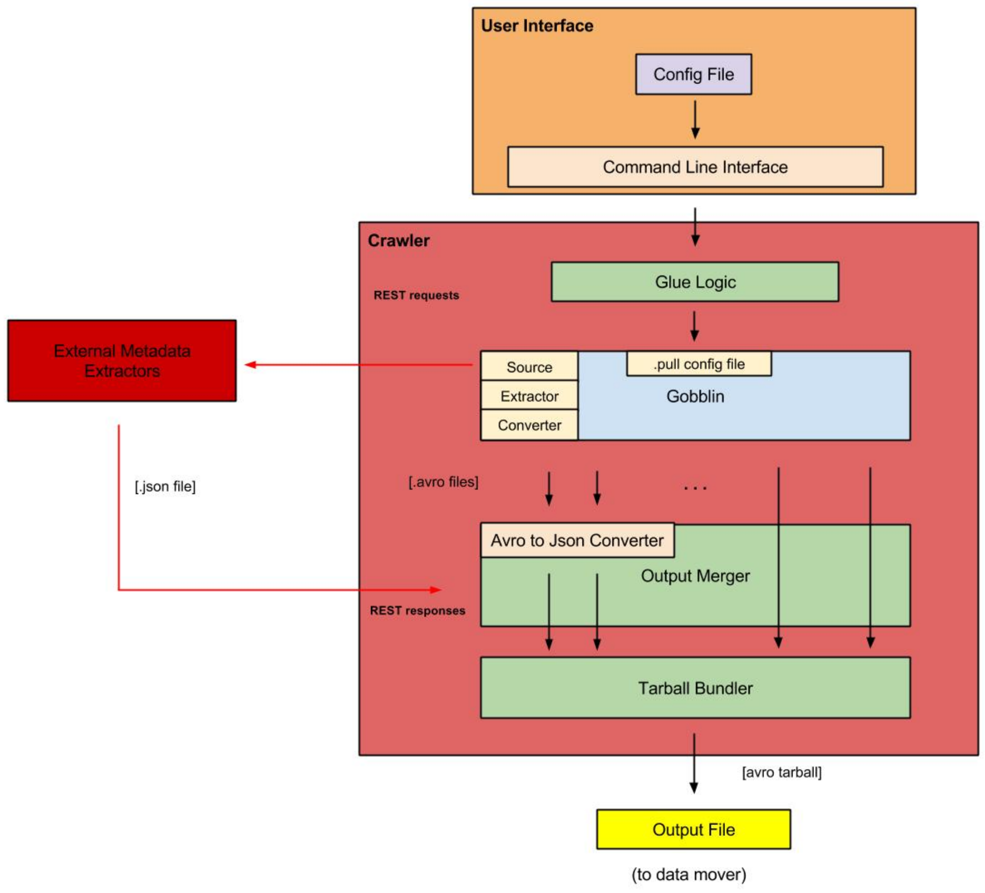
CS286A Metadata Crawler Architecture

Kyle Dillon, Ian Juch, Eric Tu

Overview

The crawler will be a wrapper built around Gobblin for our specific use case of collecting metadata from a variety of sources and outputting metadata in a standard format. To use it, we will provide a command line interface where the user should specify a config file (e.g. scheduling frequency, external metadata extractors, directories to traverse). Based on the config file, the crawler will autogenerate a Gobblin config file and start Gobblin. Gobblin will then traverse the specified directories and collect basic statistics, and if it recognizes any special files (as defined by the config file), it will send a REST request to an external metadata extractor. Output logic will then combine the metadata from Gobblin and any external REST calls, producing a single avro object for each source file. All the avro files are then compressed into a tarball and sent over to the data mover via scp.

Architecture



Description of Major Components

Gobblin:
- Will do the heavy lifting of processing the data and writing it out to a file
- Keep count of how many external REST calls are issued, and write out to a temp file
- Inputs:
  - Data
  - .pull configuration file
- Outputs:
  - Output files (.avro)

Glue logic:
- Takes the Config file as input
- Autogenerates a Gobblin .pull config file
- Launches Gobblin and sets up a process to monitor and react to new .avro files that are created by Gobblin

External Metadata Extractor:
- Expert at extracting metadata for specific types of data.
- Can handle asynchronous REST requests for metadata on a certain data source.
- Responds with a .json file of metadata.

Output Merger:
- Combines metadata from REST calls with the metadata extracted by Gobblin

Tarball Bundler:
- tar all the avro files into a single compressed tarball
- Doesn't output the final combined .avro tarball until all REST requests complete (based on count from Gobblin-produced temp file) or timeout

User Interface:
- config file
  - directories to crawl
  - destination of data mover
  - Scheduler information for periodic jobs
    - cron job syntax
  - special filetypes (csv, ipython, etc.)

Interface to Data Mover
- use scp for the file transfer

- send avro tarballs, where each avro file contains the metadata for a single source file

Pseudocode

```
# use Gobblin to iterate over files and directories recursively
For each file in directory{
    # special filetypes call out to external API's
    if(special)
       hand off to external API # probably Java

    # all filetypes get some basic stats
    get_basic_stats() # "wc" or "ls -l"
    feed metadata files to Gobblin # will convert to AVRO
}

# wait for requests to return
while(ext_api_requests_outstanding

For each converted avro file {
    convert Gobblin output from AVRO to JSON
    merge Gobblin and external API JSON files # Java JSON parser
}

tar all metadata avro files into a single tarball
scp to data mover
```

Version 2 Considerations

- Need to think about if we are overloading the source from which we are extracting metadata (could be the Gobblin source or the external REST metadata extractor)
- Make a more configurable mechanism for plugging in your own external REST API metadata extractor
- Currently, for the sake of simplicity we will skip jobs silently if the cron interval specified is shorter than the time it takes to actually do the crawl. May want to think about whether this is actually the optimal behavior.
- Need to add support for HDFS, relational databases (SQL), non-relational databases (MongoDB), and other datastores