Getting Started

Tutorials

To new developers: If you want to begin to develop on Nutch do not forget to get started looking at the Hadoop source code. Hadoop is the platform that Nutch is implemented on. In order to understand anything about how Nutch works you need to also understand Hadoop.

What are the Hadoop primitives and how do I use them? Why are they there (what functionality do the add over regular primitives)?

These primitives implement the Hadoop Writable interface (or WritableComparable). What this does is gives Hadoop control over the serialization of these objects. If you look at the higher level Hadoop File System objects like ArrayFile you will see that they implement the same interfaces for serialization. Using these primitive types allows the serialization to be done in the same way as higher order data structures such as MapFile.

How does the Hadoop implementation of MapReduce work?

1. First you need a JobConf. This class contains all the relevant information for the job. Information that you need to ensure that you include in the JobConf include: 2. Then you need to submit your job to Hadoop to be run. This is done by calling JobClient.runJob. JobClient. runJob submits the job for starting and handles receiving status updates back from the job. It starts by creating an instance of the JobClient. It continues to push the job toward execution by calling JobClient.submitJob 3. JobClient.submitJob handles splitting the input files and generating the MapReduce task.

How are Nutch's Files layed out?

As a user you have some control over how files are layed out. I am going to show how a directory structure might work. The root for a crawl is /crawl. This directory can be called what ever you want and I will assume that it is in your nutch directory but it does not have to be. Remember that these directories really hold data for MapFile files. Mapfiles are layed out like <key,value>. Inside of /crawl there are several subdirectories:

/indexes - This directory holds the index that is generated by calling bin/nutch index. The directory must be called /indexes for the NutchBean to work.
/segments - When you generate a list of urls to crawl using bin/nutch generate a new segment is generated. The segments name is a time stamp. More on this in a minute.
/linkdb - This holds a list of pages and their links. This is used by the indexer to get incoming anchor text.
/crawldb - This holds a list of all URLs that Nutch knows about and their status. This includes any errors that occurred when retrieving the page.

The segment directory:

Inside of each segment there are the following directories:

/content
/crawl_fetch
/crawl_generate
/crawl_parse
/parse_data
/parse_text

These directories contain all the data for all the pages in each segment. Lets go through the directories.

/content This directory contains the raw pages for each downloaded URL. Format <Url, Content>

/crawl_fetch

/crawl_generate This directory contains a list of urls to download as part of the segment.

/crawl_parse

/parse_data This directory contains metadata about a page. Format <Url,ParseData>

/parse_text

How do I open Nutch's data files?

You will need to interact with Nutch's files using Hadoop's MapFile and SequenceFile classes. This simple code sample shows opening a file and reading the values.

```
MapFile.Reader reader = new MapFile.Reader (fs, seqFile, conf);

        Class keyC = reader.getKeyClass();
        Class valueC = reader.getValueClass();

        while (true) {
            WritableComparable key = null;
            Writable value = null;
            try {
                key = (WritableComparable)keyC.newInstance();
                value = (Writable)valueC.newInstance();
            } catch (Exception ex) {
                ex.printStackTrace();
                System.exit(-1);
            }

            try {
                if (!reader.next(key, value)) {
                    break;
                }

                out.println(key);
                out.println(value);
            } catch (Exception e) {
                e.printStackTrace();
                out.println("Exception occured. " + e);
                break;
            }
        }
```

Tutorials

- CountLinks Counting outbound links with MapReduce