Nutch: an Open-Source Platform for Web Search

Doug Cutting
Internet Archive
doug@nutch.org

Abstract

Nutch is an open-source project providing both complete Web search software and a platform for the development of novel Web search methods. Nutch is built on a distributed storage and computing foundation, such that every operation scales to very large collections. Core algorithms crawl, parse and index Web-based data. Plugins extend functionality at various points, including network protocols, document formats, indexing schemas and query operators.

1. Introduction

Nutch is an open-source project hosted by the Apache Software Foundation [1]. Nutch provides a complete, high-quality Web search system, as well as a flexible, scalable platform for the development of novel Web search engines. Nutch includes:

- a Web crawler;
- parsers for Web content;
- a link-graph builder;
- schemas for indexing and search;
- distributed operation, for high scalability;
- an extensible, plugin-based architecture.

Nutch is implemented in Java and thus runs on many operating systems and a wide variety of hardware.

2. Architecture

Nutch has a set of core interfaces implemented by plugins. Plugins implement such things as network protocols, document formats and indexing schemas. Generic algorithms combine the plugins to create a complete system. These algorithms are implemented on a distributed computing platform, making the entire system extremely scalable.

3. Distributed Operation

Distribution operation is built in two layers, storage and computation.

3.1 Nutch Distributed File System (NDFS)

Storage is provided by the the Nutch Distributed File System (NDFS) which is modeled after the Google File System [2] (GFS). NDFS provides reliable storage across a network of PCs. Files are stored as a sequence of blocks. Each block is replicated on multiple hosts. Replication and fail-over are handled automatically, providing applications with an easy-to-manage, efficient file system that scales to multi-petabyte installations.

For small deployments, without large storage requirements, Nutch is easily configured to simply use a local hard drive for all storage, in place of NDFS.

3.2 MapReduce

MapReduce is Nutch's distributed computing layer, again inspired by Google [3]. MapReduce, as its name implies, is a two-step operation, *map* followed by *reduce*. Input and output data are files containing sequences of key-value pairs.

During the map step, input data is split into contiguous chunks that are processed on separate nodes. A user-supplied map function is applied to each datum, producing an intermediate data set.

Each intermediate datum is then sent to a reduce node, based on a user-supplied *partition* function. Partitioning is typically a hash function, so that all equivalently keyed intermediate data are all sent to a single reduce node. For example, if a map function outputs URL-keyed data, then partitioning by URL hash code sends intermediate data associated with a given URL to a single reduce node.

Reduce nodes sort all their input data, then apply a user-supplied *reduce* function to this sorted map output, producing the final output for the MapReduce operation. All entries with a given key are passed to the reduce function at once. Thus, with URL-keyed data, all data associated with a URL is passed to the reduce function and may be used to generate the final output.

The MapReduce system is robust in the face of machine failures and application errors. Thus one may reliably run long-lived applications on tens, hundreds or even thousands of machines in parallel.

A single-threaded, in-process implementation of MapReduce is also provided. This is useful not just for debugging, but also to simplify small, single-machine installations of Nutch.

4. Plugins

An overview of the primary plugin interfaces is provided below.

4.1 URL Normalizers and Filters

These are called on each URL as it enters the system. A URL normalizer transforms URLs to a standard form. Basic implementations perform operations such as lower-casing protocol names (since these are case-independent) and removing default port numbers (e.g., port 80 from HTTP URLs). If an application has more knowledge of particular URLs, then it can easily implement things such as removal of session ids within a URL normalizer.

URL filters are used to determine whether a URL is permitted to enter Nutch. One may, for example, wish to exclude queries with query parameters, since these are likely to be dynamically generated content. Or one may use a URL filter to restrict crawling to particular domains, to implement an intranet or vertical search engine.

Nutch provides regular-expression based implementations of both URL normalizer and URL filter. Thus most applications need only modify a configuration file containing regular expressions in order to alter URL normalization and filtering. However, if, e.g., an application needs to consult an external database in order to process URLs, that may easily be implemented as a plugin.

4.2 Protocol Plugins

A protocol plugin is invoked to retrieve the content of URLs with a given scheme, e.g., HTTP, FTP, FILE, etc. A protocol implementation, given a URL, returns the raw, binary content of that URL, along with metadata (e.g., protocol headers).

4.3 Parser Plugins

Parser plugins, given the output of a protocol plugin (raw content and metadata), extract text, links and metadata (author, title, etc.). Links are represented as a pair of strings: the URL that is linked to; and the "anchor" text of the link.

Nutch includes parsers for formats such as HTML, PDF, Word, RTF, etc. Since Web content is frequently malformed, robust parsers are required. Nutch currently uses the NekoHTML [4] parser for HTML, which can successfully parse most pages, even those with mismatched tags, those which are truncated, etc.

The HTML parser also produces a XML DOM parse tree of each page's content. Plugins may be specified to process this parse tree. For example, a Creative Commons plugin scans this parse tree for Creative Commons license RDF embedded within the HTML. If found, the license characteristics are added to the metadata for the parse so that they may subsequently be indexed and searched.

4.4 Indexing and Query Plugins

Nutch uses Lucene for indexing and search. When indexing, each parsed page (along with a list of incoming links, etc.) is passed to a sequence of indexing plugins in order to generate a Lucene document to be indexed. Thus plugins determine the schema used; which fields are indexed and how they are indexed. By default, the content, URL and incoming anchor texts are indexed, but one may enable other plugins to index such things as date modified, content-type, language, etc.

Queries in Nutch are parsed into an abstract syntax tree, then passed to a sequence of query plugins, in order to generate the Lucene query that is executed. The default indexing plugin generates queries that search the content, URL and anchor fields. Other plugins permit field-specific search, e.g., searching within the URL only, date-range searching, restricting results to particular document types and/or languages, etc.

5. Algorithms

Generic algorithms are implemented in terms of the plugins outlined above, in order to perform user-level tasks such as crawling, indexing etc. Each algorithm, except search, is implemented as one or more MapReduce operations. All persistent data may be stored in NDFS for completely distributed operation.

5.1 Crawling

The crawling state is kept in a data structure called the *crawldb*. It consists of a mapping from URLs to a CrawlDatum record. Each CrawlDatum contains a date to next fetch the URL, the status of the URL (fetched, unfetched, gone, etc.), the number of links found to this URL, etc. The crawldb is bootstrapped by inserting a few root URLs.

The Nutch crawler then operates in a cycle:

1. generate URLs to fetch from crawldb;
2. fetch these URLs;
3. parse the fetched content;
4. update crawldb with results of fetch and new URLs found when parsing.

These steps are repeated. Each step is described in more detail below.

5.1.1 Generate

URLs are generated which are due to be fetched (status is not 'gone' and next fetch date is before now). This set of URLs may further be limited so that only the top most linked pages are requested, and so that only a limited number of URLs per host are generated.

5.1.2 Fetch

The fetcher is a multi-threaded application that employs protocol plugins to retrieve the content of a set of URLs.

5.1.3 Parse

Parser plugins are employed to extract text links and other metadata from the raw binary content.

5.1.4 Update

The status of each URL fetched along with the list of linked URLs discovered while parsing are merged with the previous version of the crawldb to generate a new version. URLs which were successfully fetched are marked as such, incoming link counts are updated, and new URLs to fetch are inserted.

5.2 Link Inversion

All of the parser link outputs are processed in a single MapReduce operation to generate, for each URL, the set of incoming anchor texts. Associating incoming anchor text with URLs has been demonstrated to dramatically increase the quality of search results. [5]

5.3 Indexing

A MapReduce operation is used to combine all information known about each URL: page text, incoming anchor text, title, metadata, etc. This data is passed to the indexing plugins to create a Lucene document that is then added to a Lucene index.

5.4 Search

Nutch implements a distributed search system, but, unlike other algorithms, search does not use MapReduce. Separate indexes are constructed for partitions of the collection. Indexes are deployed to search nodes. Each query is broadcast to all search nodes. The top-scoring results over all indexes are presented to the user.

6. Status

Nutch has an active set of users and developers. Many sites are using Nutch today, for both intranet and vertical search applications, scaling to tens of millions of pages. [6] Nutch's search quality rivals that of commercial alternatives [7] at considerably lower costs. [8] Soon we hope that Nutch's public deployments will include multi-billion page search engines.

The MapReduce-based version of Nutch described here is under active development. In the course of Summer 2005 we expect to index a billion-page collection using Nutch at the Internet Archive.

7. Acknowledgments

The author wishes to thank The Internet Archive, Yahoo!, Michael Cafarella and all who contribute to Nutch.

8. References

[1] http://lucene.apache.org/nutch/

[2] Ghemawat, Gobioff, and Leung, *The Google File System*, 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October, 2003, http://labs.google.com/papers/gfs.html

[3] Dean and Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004, http://labs.google.com/papers/mapreduce.html

[4] Clark, *CyberNeko HTML Parser*, http://people.apache.org/~andyc/neko/doc/html/index.html

[5] Craswell, Hawking, and Robertson, *Effective site finding using link anchor information*, Proceedings of ACM SIGIR 2001, http://research.microsoft.com/users/nickcr/pubs/craswell_sigir01.pdf

[6] http://wiki.apache.org/nutch/PublicServers

[7] http://www.nutch.org/twiki/Main/Evaluations/OSU_Queries.pdf

[8] http://osuosl.org/news_folder/nutch