Nutch2Architecture

Note this document is EXTREMELY outdated. If you are able to contribute documentation here then please contact us on dev@nutch. In the meantime please check out the other pages about Nutch2 FrontPage#Nutch_2.0

Overview

- Reuse of existing Nutch codebase
  - While some things will change this architecture is more of a refactor than a complete re-write. Much of the existing codebase including plugin functionality should be reused.
- Remove the plugin framework
  - After some experimenting, DI using spring or another similar framework presents problems. Good news is that we can achieve the same thing using the configuration objects from hadoop along with creating new instances using reflectionutils. This is more service locator than dependency injection but it still gives us the same benefits.
  - Have the ability to change the jobconfiguration settings for tools. This can be accomplished through some type of properties file on the classpath and would be useful for testing, for example the ability to switch out an outputformat to see the output in text format.
  - Have mock objects that make it easy to test jobs.

Data Structures

- CrawlBase
  - Url CrawlState
  - CrawlState
    - Current state fields
    - CrawlHistory is a list of CrawlDatum objects ordered by reverse date
  - CrawlDatum has Metadata
- CrawlList
  - Url CrawlHistory
  - Separate from CrawlBase for Multiple concurrent crawls
- FetchedContent
  - Url BytesWritable, FetchStatus
  - FetchStatus would be a status of the fetch, error codes, any fetch information. This would then be translated by another tool back into the CrawlBase. FetchStatus has Metadata.
- ParsedContent
  - Url MapWritable
  - [MapWritable] would contain Text Writable or Writable[] and would allow the parsing of all different types of elements within the content (href, headers, etc.)
- Processing
  - Processing would take the ParsedContent and translate that into multiple specific data parts. These data parts aren't used by any part of the system except Scoring.
  - Processing would be specific functions including updating the CrawlBase, peforming analysis on ParsedContent, Integration of data from other sources.
  - Some processors would translate content into formats needed by scorers.
  - Processors are not constrained by specific data structures to allow flexibility in analysis, updating, blocking or removal, and other forms of data processing. The only requirement is scoring programs must be able to access processing output data structure in a one to one relationship.
- Scoring
  - Url Field
  - Url Float
  - Field is a name, value(s), and score, being Text, Text, and Float respectively.
  - The fields become the fields that are indexed with the scores becoming field boosts.
  - Scoring takes the specific data parts from analysis and outputs the above formats.
  - Field needs lucene semantics.
  - Indexing
    - Indexing indexes Fields for a document according to the field values and boosts. Indexing does not determine either field values or boost values.
    - Indexing aggregates document boosts to create a final document score.

Tools

- Injector
  - Injects data sources into the CrawlBase creating new CrawlBase, CrawlHistory objects.
  - This could also be used to update the status or change the state of Urls in the CrawlBase manually.
- Generator
  - Creates CrawlLists from the CrawlBase
  - Filters could be created to run on only a subset of the CrawlBase Urls.
- Fetcher
  - Fetches CrawlLists objects and creates FetchedContent objects.
- UpdateCrawl
  - Updates the CrawlBase Urls with the FetchedStatus objects of the FetchedContent.
  - This does not add new Urls to the database, only updates current Urls.
- Parser

    - Creates ParsedContent objects from FetchContent sources.
    - Run multiple different parsers based on different conditions.
- Processors
    - New Url Processor
        - A processor which updates the CrawlBase with new urls parsed from ParsedContent sources.
    - Html Processor
        - Does specific processing on Html sources from ParsedContent.
    - Link Processor
        - Creates a specific database of Url Inlinks and Outlinks.
    - BlackList Processor
        - Processor which removes urls and their content from being indexed if they are on a blacklist.
    - Other Processors
        - There should be many other tools here that perform specific functions such at language identification, handling redirected urls and scoring, etc.
- Scorers
    - Html Scorer
        - Scores Html analysis
        - Link Scorer
            - Create a page-rank type score from the Link Analysis.
        - Other Scorers
- Indexer
    - Create Lucene indexes from multiple Scoring objects.
- Query tools

Supporting Infrastructure

- Shard management
    - Perhaps a separate project to be shared by Lucene, Solr, and Nutch.
    - Nutch shards need other content besides indexes, summaries and links for example.
- Cluster management
- Automated job streams for nutch processes
- Build and command line scripts
    - Allow packaging of all core and third-party contrib jars to run on a standard hadoop cluster.
    - People should be able to create an extension and drop in a jar and it just runs, no need to deploy jar manually to all slaves.
- Full unit testing suite, documentation and tutorials
    - Maybe a book would be good. We definitely need documentation to lead a person from installation to extension.