NutchAndFronteraDesignGoals

Design Goals for How Nutch and Frontera can Work Together

Here are the possible goals of integration of Frontera and Nutch:

1. to get the best of two: Nutch is good at scale, faster on fetching/parsing, but Frontera/Scrapy is online, much easier on customization, having good docs and written in Python, 2. to ease the migration from Frontera to Nutch and opposite, 3. identify and fix design problems.

Now, few words how Nutch and Frontera could work together.

1. Nutch Fetcher can be easily used with Frontera, if it will be implemented as a service, communicating by means of Kafka or ZeroMQ and talking Frontera protocol (which is documented). Fetching involves parsing and many string operations, that could be more efficient in JVM. FetchItem would require adapter for Frontera Request, the same for ParseData. It could help Frontera users save some time on fetching, but if use case requires scraping (for broad crawling it isn't), they would need to add scraping step later. 2. Scrapy can be used as fetcher for Nutch too. We just need to figure out a way how to run Scrapy spider in Hadoop environment. Input/Output adapters, process wrapper are needed. Some interface modifications are also required to use extracted items from content in Nutch-Solr(or other Lucene based) pipeline. Scrapy is much more efficient in network operations conceptually: asynchronous select()/epoll based http client and connection pool. This can be improved in Nutch. This would allow writing/debugging of custom scraping code amazingly easy. Plus Nutch is used as a crawl frontier for Scrapy and Tika-based parsing and indexing primitives can be used for building search. 3. Frontera's DB and strategy workers can be used in Hadoop/Nutch pipeline to generate Nutch segments and read fetcher output with slight modifications. It's possible to generate quite big segment by continuously running get_next_requests() routine (meant to be used for small batches). They use low level storage, currently HBase and RDBMS are supported. Number of workers can be scaled, they're designed for this. Same problems are here, need of adapters and process wrappers. RDBMS could suffer from concurrent access, but that's solvable. This would allow to use Frontera as a crawl frontier with Nutch. It could be helpful if someone wants to implement crawling strategy in Python.

Nutch and Frontera use cases aren't completely overlap. Majority of people who look into Frontera want to crawl some small amount of websites, scrape some data from them and revisit. Sometimes they need to scale fetching (meaning no polite crawling here) or parsing/scraping part, and sometimes they need some custom prioritization or external queue management. Quite few is using it for broad crawling with Kafka and HBase.

Currently, Frontera is moving towards the ease of use: ZeroMQ transport, transport layer abstraction, standalone Frontera/Scrapy based crawler in Docker, web UI.

Major Nutch Use Cases and Future Vision that Align with Frontera

1. Deep Web Extractions - both from a crawler, and using various interactive and non-obtrusive Javascript libraries. We started with Selenium but are now looking at HTMLUntil, PhantomJS, and others. 2. Measuring Crawl Footprint - I think we need to understand better the crawl footprint, and use that information to better guide and strategize crawling. 3. Adaptive and ML-based crawling algorithms - my team is working on a Machine Learning based algorithm for crawling that leverages Naive Bayes, and RL. 4. Content Extraction from more and more formats with Tika. This is one potential area we could overlap on since there is both a Tika Python library and Nutch Python library (originating from DARPA Memex).

Seems like the more stuff we do in the Python libraries, and with Tika potentially could serve as an initial integration. As for broader crawling, I'm also interested in how Nutch and Spark can work together. Nutch over Spark is something I have a few researchers in my team working on now.