Storm Crawler

Low latency scalable web crawling on Apache Storm

Julien Nioche

julien@digitalpebble.com

digitalpebble

Berlin Buzzwords 01/06/2015

digitalPebble

About myself

- **DigitalPebble Ltd**, Bristol (UK)

- Specialised in Text Engineering
  - Web Crawling
  - Natural Language Processing
  - Information Retrieval
  - Machine Learning
- Strong focus on Open Source & Apache ecosystem
  - Nutch
  - Tika
  - GATE, UIMA
  - SOLR, Elasticearch
  - Behemoth

Storm-Crawler : what is it?

- Collection of resources (SDK) for building web crawlers on Apache Storm
- *https://github.com/DigitalPebble/storm-crawler*

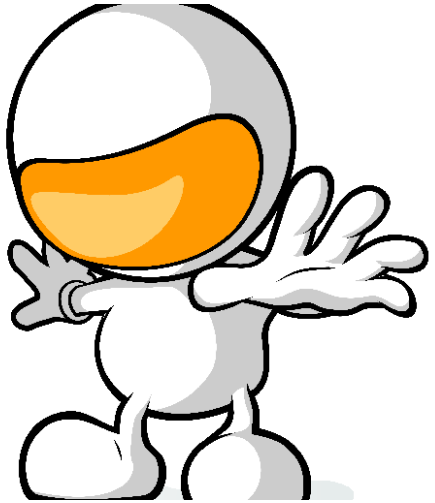
- Apache License v2
- Artefacts available from Maven Central
- Active and growing fast
- Version 0.5 just released!

=> Can scale
=> Can do low latency
=> Easy to use and extend
=> Nice features (politeness, scraping, sitemaps etc...)

What it is not

- A well packaged application
  - It's a SDK : it requires some minimal programming and configuration
- No global processing of the pages e.g.PageRank
- No fancy UI, dashboards, etc...
  - Build your own or integrate in your existing ones
  - But can use Storm UI + metrics to various backends (Librato, ElasticSearch)

Comparison with Apache Nutch



StormCrawler vs Nutch

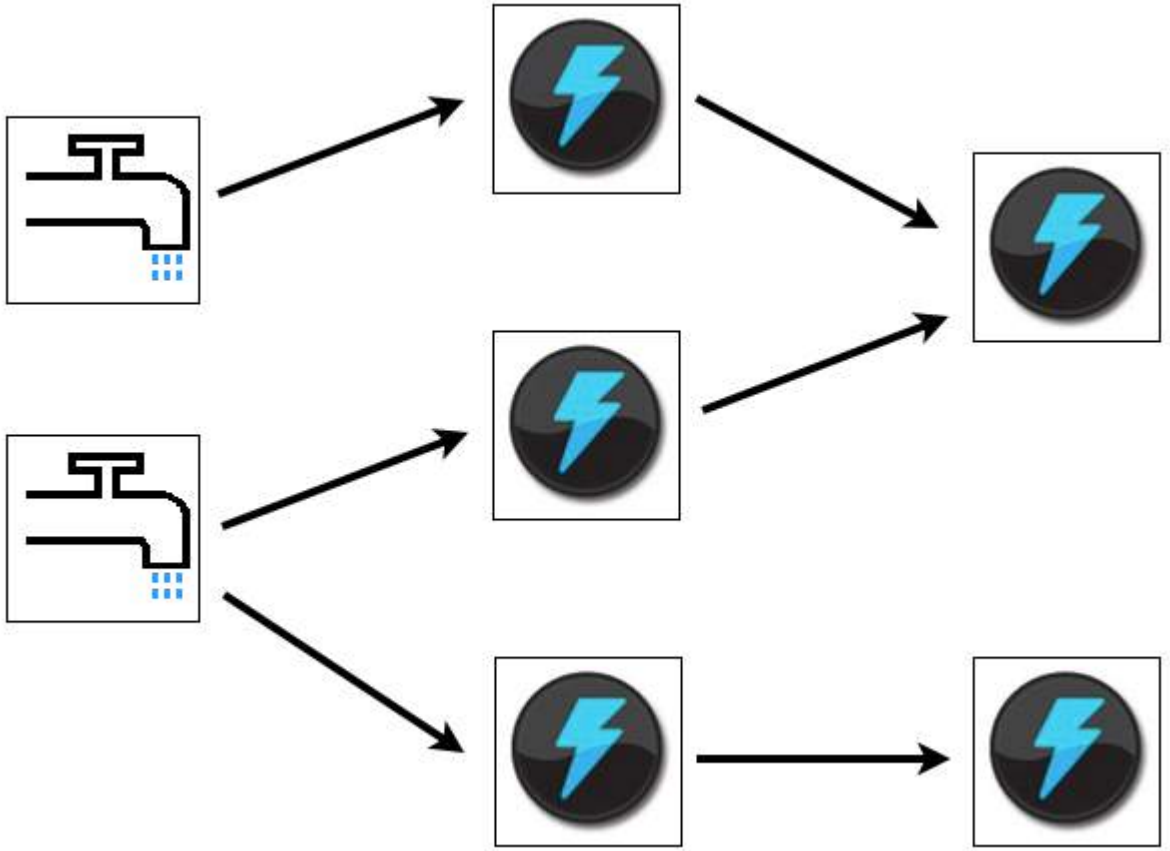
- Nutch is batch driven : little control on when URLs are fetched
  - Potential issue for use cases where need sessions
  - latency++
- Fetching only one of the steps in Nutch
  - SC : 'always fetching' ; better use of resources
- More flexible
  - Typical case : few custom classes (at least a Topology) the rest are just dependencies and standard SC components
  - Logical crawls : multiple crawlers with their own scheduling easier with SC via queues
- Not ready-to use as Nutch : it's a SDK
- Would not have existed without it
  - Borrowed some code and concepts
  - Contributed some stuff back

Apache Storm

Apache Storm

- Distributed real time computation system
- http://storm.apache.org/
- Scalable, fault-tolerant, polyglot and fun!
- Implemented in Clojure + Java
- "*Realtime analytics, online machine learning, continuous computation, distributed RPC, ETL, and more*"

Topology = spouts + bolts + tuples + streams



What's in Storm Crawler?

Resources: core vs external

- Core
  - Fetcher(s) Bolts
  - JsoupParserBolt
  - SitemapParserBolt
  - ...
- External
  - Metrics-related : inc. connector for Librato
  - ElasticSearch : Indexer, Spout, StatusUpdater, connector for metrics
  - Tika-based parser bolt
- User-maintained external resources
- Generic Storm resources (e.g. spouts)

FetcherBolt

- Multi-threaded
- Polite
  - Puts incoming tuples into internal queues based on IP/domain/hostname
  - Sets delay between requests from same queue
  - Internal fetch threads
  - Respects robots.txt
- Protocol-neutral
  - Protocol implementations are pluggable
  - Default HTTP implementation based on Apache HttpClient
- Also have a SimpleFetcherBolt
  - Need handle politeness elsewhere (spout?)

JSoupParserBolt

- HTML only but have a Tika-based one in external
- Extracts texts and outlinks
- Calls URLFilters on outlinks
  - normalize and / or blacklists URLs
- Calls ParseFilters on document
  - e.g. scrape info with *XpathFilter*
  - Enrich metadata content

ParseFilter

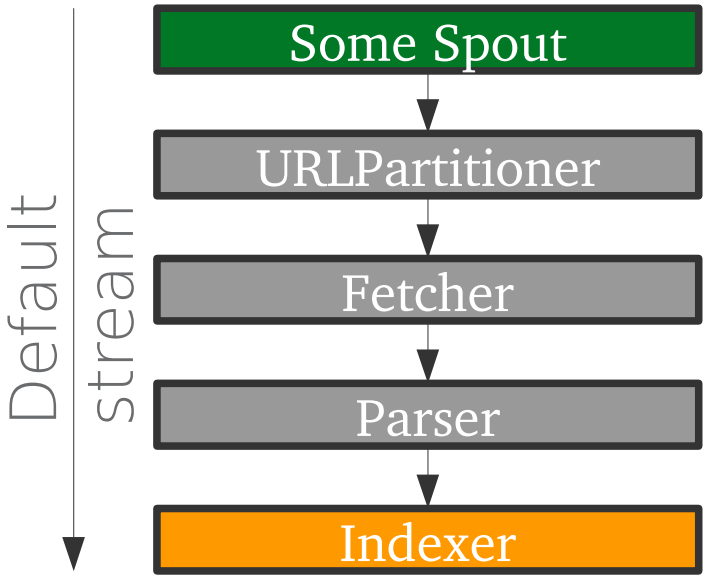
- Extracts information from document
  - Called by *ParserBolt(s)
  - Configured via JSON file
  - Interface
    - *filter(String URL, byte[] content, DocumentFragment doc, Metadata metadata)*
  - ***com.digitalpebble.storm.crawler.parse.filter.XpathFilter.java***
    - Xpath expressions
    - Info extracted stored in metadata
    - Used later for indexing

URLFilter

- Control crawl expansion
- Delete or rewrites URLs
- Configured in JSON file
- Interface
  - *String filter(URL sourceUrl, Metadata sourceMetadata,String urlToFilter)*

- ***Basic***
- ***MaxDepth***
- ***Host***
- ***RegexURLFilter***
- ***RegexURLNormalizer***

Basic Crawl Topology



(1) pull URLs from an external source

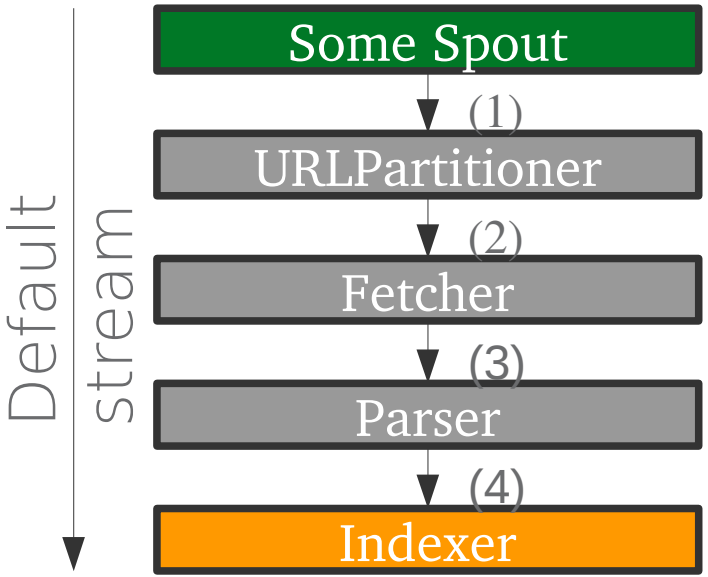
(2) group per host / domain / IP

(3) fetch URLs

(4) parse content

(5) index text and metadata

Basic Topology : Tuple perspective



(1) <String url, Metadata m>

(2) <String url, String key, Metadata m>

(3) <String url, byte[] content, Metadata m>

(4) <String url, byte[] content, Metadata m, String text>

Basic scenario

- Spout : simple queue e.g. RabbitMQ, AWS SQS, etc...
- Indexer : any form of storage but often index e.g. SOLR, ElasticSearch

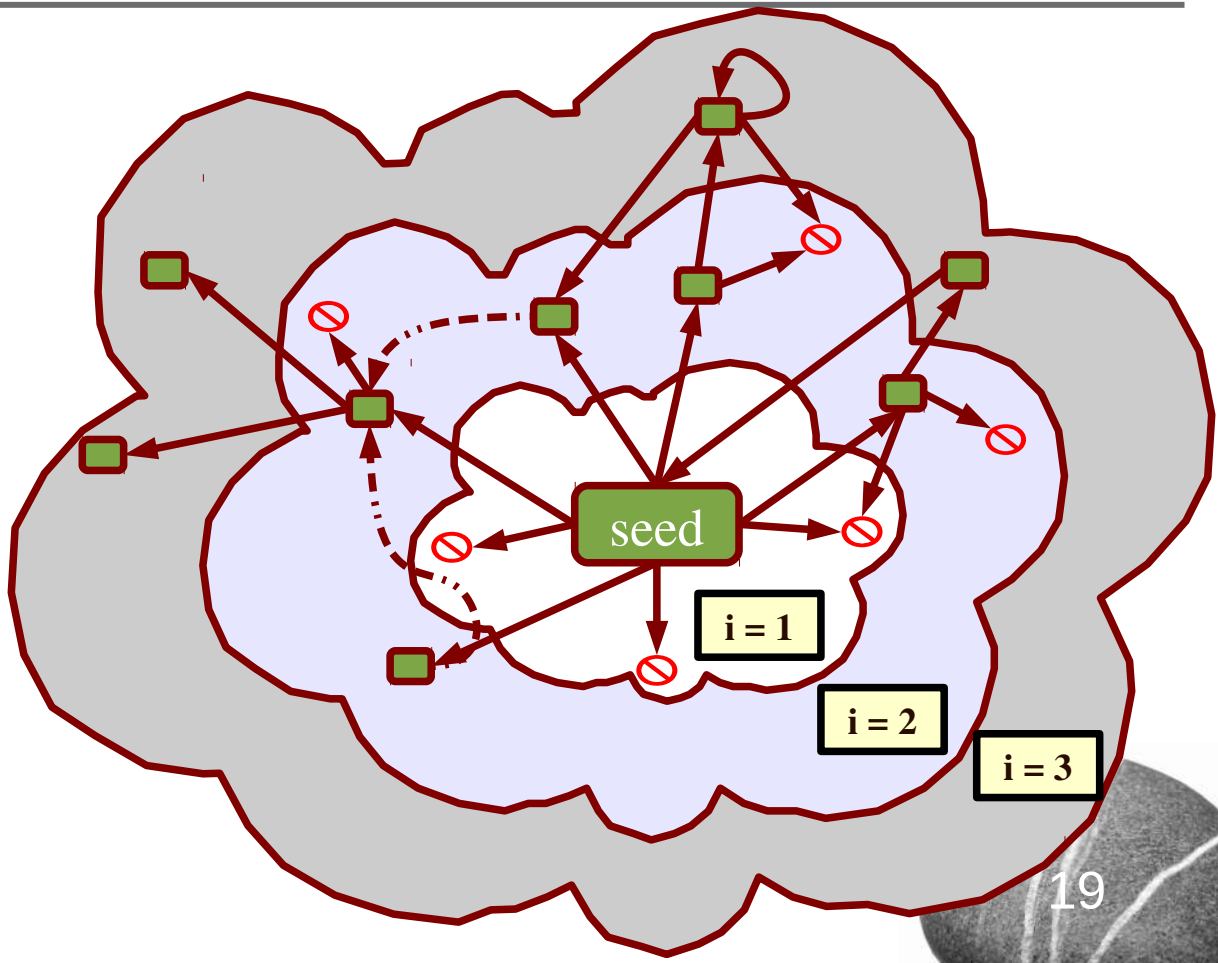
- Components in grey : default ones from SC

- Use case : non-recursive crawls (i.e. no links discovered), URLs known in advance. Failures don't matter too much.

- *“Great! What about recursive crawls and / or failure handling?”*

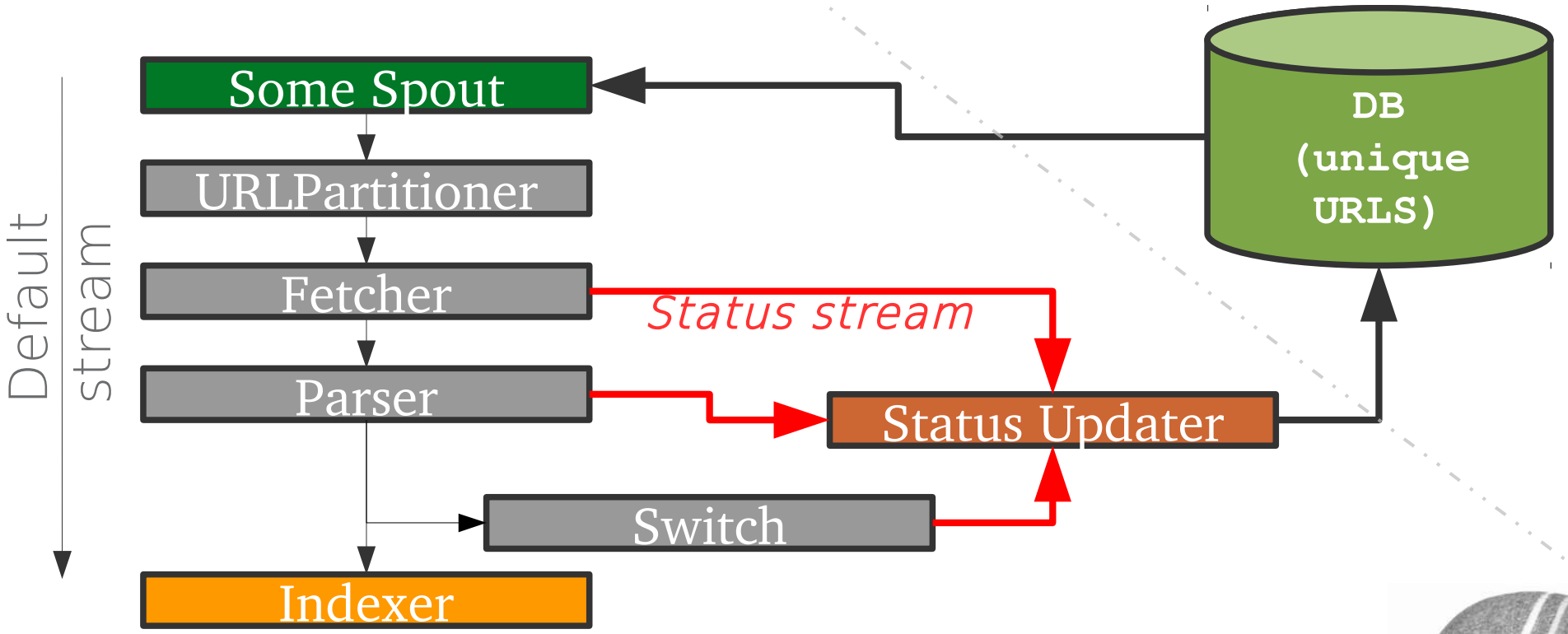
Frontier expansion

- Manual “discovery”
  - Adding new URLs by hand, “seeding”
- Automatic discovery of new resources (frontier expansion)
  - Not all outlinks are equally useful - control
  - Requires content parsing and link extraction



[Slide courtesy of A. Bialecki]

Recursive Crawl Topology



Status stream

- Used for handling errors and update info about URLs
- Newly discovered URLs from parser

```
public enum Status {
 DISCOVERED, FETCHED, FETCH_ERROR, REDIRECTION, ERROR;}
```

- *StatusUpdater* : writes to storage
- Can/should extend *AbstractStatusUpdaterBolt*

External : ElasticSearch

- IndexerBolt
  - Indexes URL / metadata / text for search
  - extends AbstractIndexerBolt
- StatusUpdaterBolt
  - extends AbstractStatusUpdaterBolt
  - URL / status / metadata / nextFetchDate in *status* index
- ElasticSearchSpout
  - Reads from *status* index
  - Sends URL + Metadata tuples down topology
- MetricsConsumer
  - indexes Storm metrics for displaying e.g. with Kibana

How to use it?

How to use it?

- Write your own Topology class (or hack the example one)
- Put resources in src/main/resources
  - URLFilters, ParseFilters etc...
- Build uber-jar with Maven
  - mvn clean package
- Set the configuration
  - External YAML file (e.g. crawler-conf.yaml)
- Call Storm

```
storm jar target/storm-crawler-core-0.5-SNAPSHOT-jar-with-dependencies.jar
com.digitalpebble.storm.crawler.CrawlTopology -conf crawler-conf.yaml
```


Topology in code

```
@Override
protected int run(String[] args) {
    TopologyBuilder builder = new TopologyBuilder();

    builder.setSpout("spout", new RandomURLSpout());

    builder.setBolt("partitioner", new URLPartitionerBolt())
            .shuffleGrouping("spout");

    builder.setBolt("fetch", new FetcherBolt()).fieldsGrouping(
            "partitioner", new Fields("key"));

    builder.setBolt("sitemap", new SiteMapParserBolt())
            .localOrShuffleGrouping("fetch");

    builder.setBolt("parse", new ParserBolt()).localOrShuffleGrouping(
            "sitemap");

    builder.setBolt("switch", new StatusStreamBolt())
            .localOrShuffleGrouping("parse");

    builder.setBolt("index", new IndexerBolt()).localOrShuffleGrouping(
            "parse");

    builder.setBolt("status", new PrinterBolt())
            .localOrShuffleGrouping("fetch", Constants.StatusStreamName)
            .localOrShuffleGrouping("sitemap", Constants.StatusStreamName)
            .localOrShuffleGrouping("switch", Constants.StatusStreamName)
            .localOrShuffleGrouping("parse", Constants.StatusStreamName);

    conf.registerMetricsConsumer(LoggingMetricsConsumer.class);

    return submit("crawl", conf, builder);
}
```

Grouping!

Politeness and data locality

Storm UI

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks
crawl	crawl-1-1425898661	ACTIVE	1m 31s	2	11	11

Topology actions

Activate | Deactivate | Rebalance | Kill

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	5640	5700	107.750	160	0
3h 0m 0s	5640	5700	107.750	160	0
1d 0h 0m 0s	5640	5700	107.750	160	0
All time	5640	5700	107.750	160	0

Spouts (All time)

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	Last error
spout	1	1	200	200	107.750	160	0			

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Error Host	Error Port	Last error
fetch	1	1	180	180	0.003	0.400	200	183.300	200	0			
index	1	1	0	0	0.004	1.250	80	1.333	60	0			
parse	1	1	4940	5000	0.126	44.250	80	31.000	80	0			
partitioner	1	1	120	120	0.000	0.000	160	0.250	160	0			
sitemap	1	1	120	120	0.001	0.500	80	0.250	80	0			
status	1	1	0	0	0.060	0.335	5020	0.222	5040	0			
switch	1	1	80	80	0.001	0.500	80	1.000	80	0			

Case Studies

- 'No follow' crawl #1: existing list of URLs only – one off
  - http://www.stolencamerafinder.com/ : [RabbitMQ + Elasticsearch]
- 'No follow' crawl #2: Streams of URLs
  - http://www.weborama.com : [queues? + HBase]
- Monitoring of finite set of URLs / non recursive crawl
  - http://www.shopstyle.com : scraping + indexing [DynamoDB + AWS SQS]
  - http://www.ontopic.io : [Redis + Kafka + ElasticSearch]
  - http://www.careerbuilder.com : [RabbitMQ + Redis + ElasticSearch]
- Full recursive crawler
  - http://www.shopstyle.com : discovery of product pages [DynamoDB]

What's next?

- Just released 0.5
  - Improved WIKI documentation but can always do better
- All-in-one crawler based on ElasticSearch
  - Also a good example of how to use SC
  - Separate project
  - Most resources already available in external
- Additional Parse/URLFilters
  - Basic URL Normalizer #120
- Parse -> generate output for more than one document #117
  - Change to the ParseFilter interface
- Selenium-based protocol implementation
  - Handle AJAX pages

Resources

- https://storm.apache.org/
- https://github.com/DigitalPebble/storm-crawler/wiki
- http://nutch.apache.org/
- "*Storm Applied*"
  http://www.manning.com/sallen/

Questions

?


Thank you