StormCrawler

Low Latency Web Crawling on Apache Storm

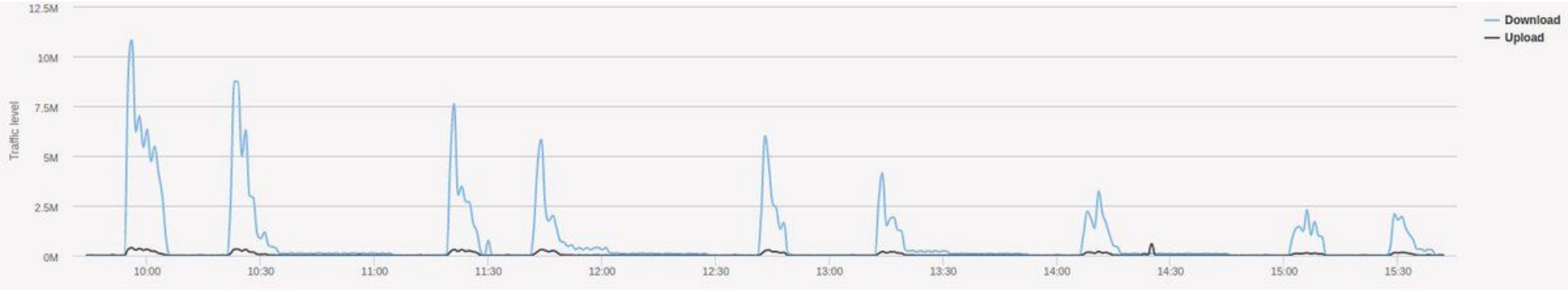
Julien Nioche
julien@digitalpebble.com
@digitalpebble
@stormcrawlerapi

digitalPebble

About myself

- **DigitalPebble Ltd**, Bristol (UK)
- Text Engineering
  - Web Crawling
  - Natural Language Processing
  - Machine Learning
  - Search
- Open Source & Apache ecosystem
  - StormCrawler, Apache Nutch, Crawler-Commons
  - GATE, Apache UIMA
  - Behemoth
  - Apache SOLR, Elasticsearch

A typical Nutch crawl



Wish list for (yet) another web crawler

- Scalable / distributed
- Low latency
- Efficient yet polite
- Nice to use and extend
- Versatile : use for archiving, search or scraping
- Java

… and not reinvent a whole framework for distributed processing

=> Stream Processing Frameworks

APACHE
STORM™

History

Late 2010 : started by Nathan Marz
September 2011 : open sourced by Twitter
2014 : graduated from incubation and became Apache top-level project

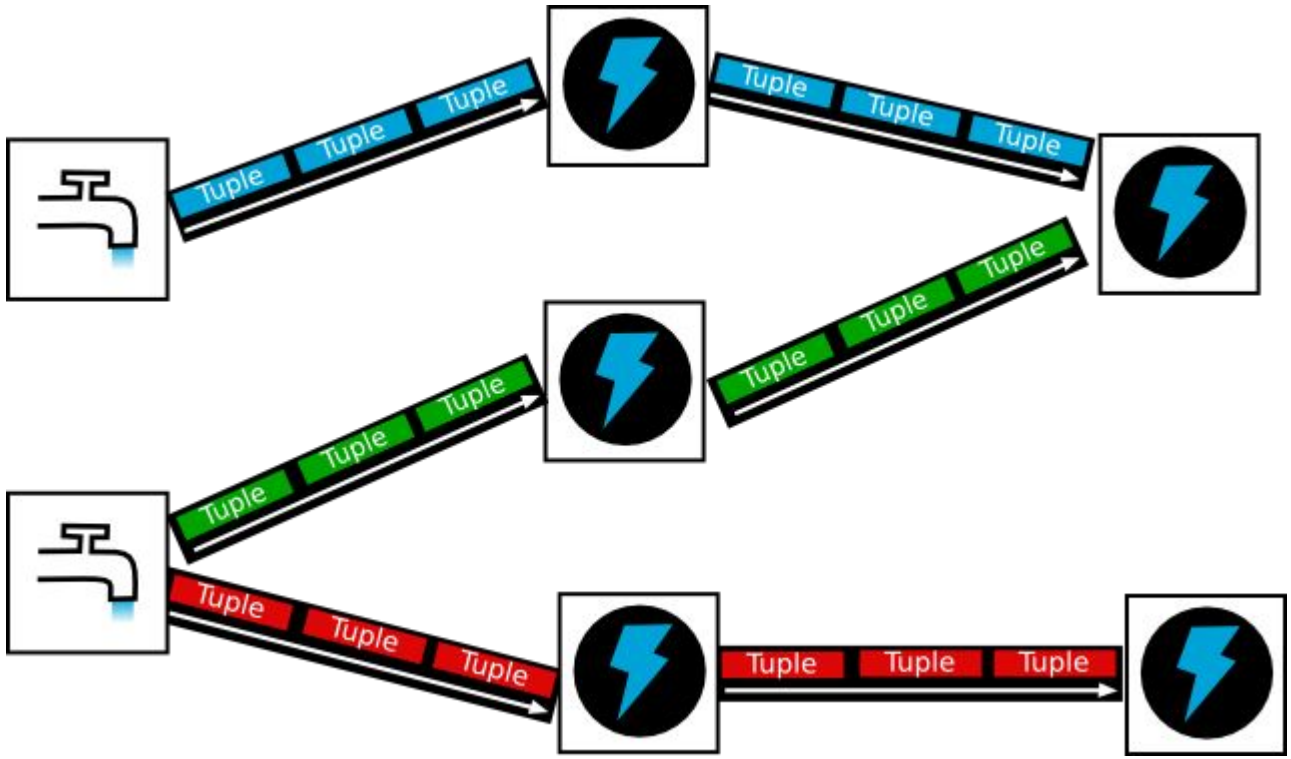
http://storm.apache.org/

0.10.x
1.x => Distributed Cache , Nimbus HA and Pacemaker, new package names
2.x => Clojure to Java

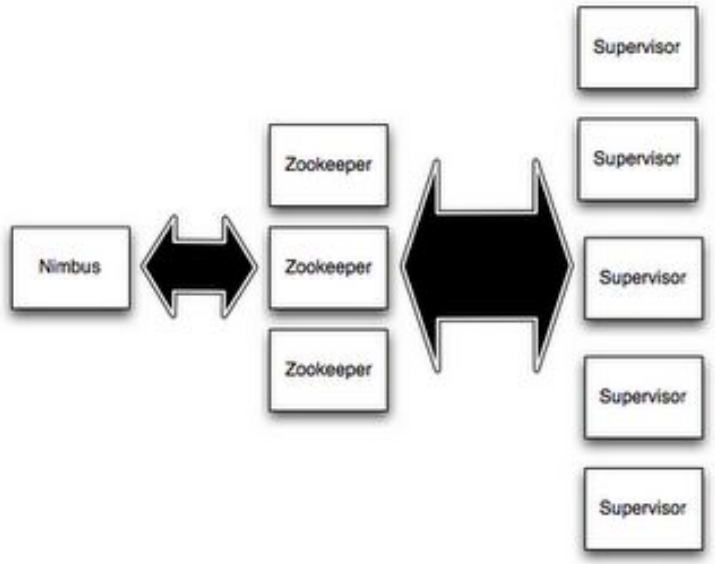
Current stable version: 1.0.2

Main concepts

Topologies
Streams
Tuples
Spouts
Bolts

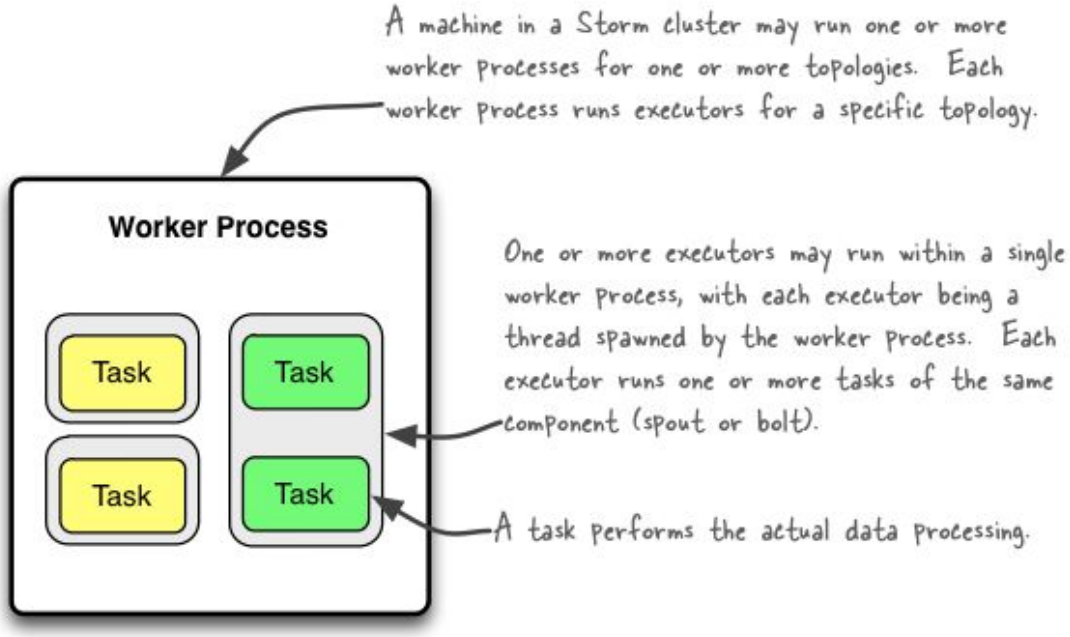


Architecture



http://storm.apache.org/releases/1.0.2/Daemon-Fault-Tolerance.html

Worker tasks



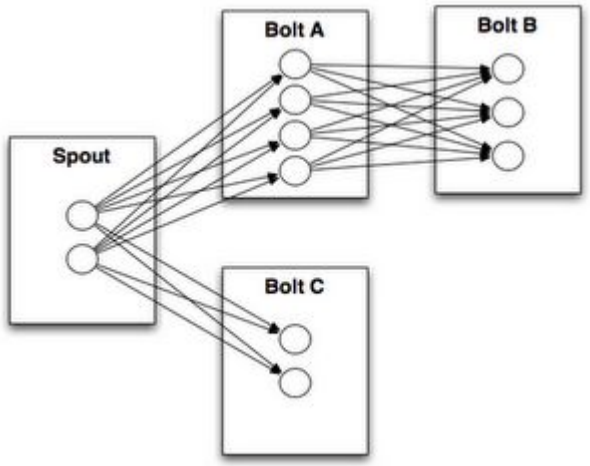
http://storm.apache.org/releases/1.0.2/Understanding-the-parallelism-of-a-Storm-topology.html

Stream Grouping

Define how to streams connect bolts and how tuples are partitioned among tasks

Built-in stream groupings in Storm (implement CustomStreamGrouping) :

1. Shuffle grouping
2. Fields grouping
3. Partial Key grouping
4. All grouping
5. Global grouping
6. None grouping
7. Direct grouping
8. Local or shuffle grouping



In code

From *http://storm.apache.org/releases/current/Tutorial.html*

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("sentences", new RandomSentenceSpout(), 2);
builder.setBolt("split", new SplitSentence(), 4).setNumTasks(8)
        .shuffleGrouping("sentences");
builder.setBolt("count", new WordCount(), 12)
        .fieldsGrouping("split", new Fields("word"));
```

Run a topology

- Build topology class using TopologyBuilder
- Build über-jar with code and resources
- *storm* script to interact with Nimbus
  - `storm jar topology-jar-path class ...`
- Uses config in *~/.storm,* can also pass individual config elements on command line
- Local mode? Depends how you coded it in topo class

- Will run until kill with *storm kill*

- Easier ways of doing as we'll see later

Guaranteed message processing

- Spout/Bolt => Anchoring and acking

  outputCollector.emit(tuple, new Values(order));

  outputCollector.ack(tuple);

- Spouts have **ack** / **fail** methods
  - manually failed in bolt (explicitly or implicitly e.g. extensions of `BaseBasicBolt`)
  - triggered by timeout (Config.TOPOLOGY _ MESSAGE _ TIMEOUT _ SECS - default 30s)
- Replay logic in spout
  - depends on datasource and use case
- Optional
  - Config.TOPOLOGY_ACKERS set to 0 (default 1 per worker)

See http://storm.apache.org/releases/current/Guaranteeing-message-processing.html

Metrics

Nice in-built framework for metrics

Pluggable => default file based one (`IMetricsConsumer`)

```
org.apache.storm.metric.LoggingMetricsConsumer
```

Define metrics in code

```
this.eventCounter = context.registerMetric("SolrIndexerBolt",
        new MultiCountMetric(), 10);
```

…

```
context.registerMetric("queue_size", new IMetric() {
    @Override
    public Object getValueAndReset() {
        return queue.size();
    }
}, 10);
```

Topology actions

Activate | Deactivate | Rebalance | Kill | Debug | Stop Debug | Change Log Level

Topology stats

Window	Emitted	Transferred	Complete latency (ms)
10m 0s	800	860	59998.500
3h 0m 0s	130115	135295	39446.914
1d 0h 0m 0s	1367573	1410730	41303.305
All time	1840500	1902120	40273.848

Spouts (All time)

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Er
spout	10	10	164440	164440	40273.848	158440	3220		

Showing 1 to 1 of 1 entries

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed
feed	1	1	1285640	1347260	0.000	7.962	124040	2.812	124020	0
fetch	1	1	168560	168560	0.000	0.048	161700	15100.497	161680	0
partitioner	1	1	161780	161780	0.000	0.018	161700	0.021	161680	0
ssb	1	1	60080	60080	0.000	0.019	62040	0.031	62040	0
status	1	10	0	0	0.000	0.021	1329460	478.042	1329440	0
warc	1	1	0	0	0.000	3.000	69260	3315.981	131080	0

Logs

- One log file per worker/machine
- Change log levels via config or UI
- Activate log service to view from UI
  - Regex for search
- Metrics file there as well (if activated)

Things I have not mentioned

- Trident : Micro batching, comparable to Spark Streaming
- Windowing
- State Management
- DRPC
- Loads more ...

-> find it all on http://storm.apache.org/


STORMCRAWLER


18

Storm Crawler

http://stormcrawler.net/

- First released in Sept 2014 (started year earlier)
- Version : 1.2
- Release every 1 or 2 months on average
- Apache License v2

- Built with Apache Maven
- Artefacts available from Maven Central

Collection of resources

- Core
- External
  - Cloudsearch
  - Elasticsearch
  - SOLR
  - SQL
  - Tika
  - WARC
- Third-party

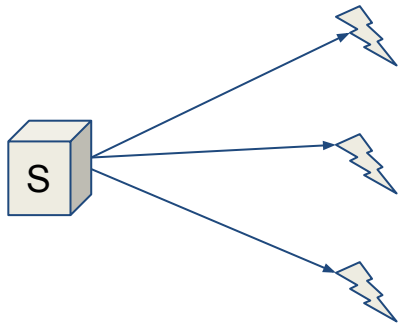

STEP
BY
STEP

Step #1 : Fetching

FetcherBolt

Input : <String, Metadata>
Output : <String, byte[], Metadata>

```
TopologyBuilder builder = new TopologyBuilder();

…
// we have a spout generating String, Metadata
tuples!

builder.setBolt("fetch", new FetcherBolt(),3)
  .shuffleGrouping("spout");
```



Problem #1 : politeness

- Respects robots.txt directives (http://www.robotstxt.org/)
  - Can I fetch this URL? How frequently can I send requests to the server (*crawl-delay*)?
- Get and cache the Robots directives
- Then throttle next call if needed
  - `fetchInterval.default`

Politeness sorted?

No - still have a problem : spout shuffles URLs from same host to 3 different tasks !?!?

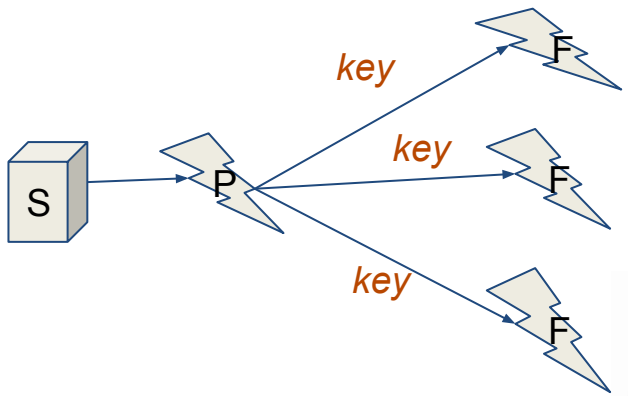
URLPartitionerBolt

Partitions URLs by host / domain or IP

`partition.url.mode: “byHost” | “byDomain” | “byIP”`

```
public void declareOutputFields(OutputFieldsDeclarer declarer) {
       declarer.declare(new Fields("url", "key", "metadata"));
   }
```

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

    builder.setBolt("fetch", new FetcherBolt(),3)
        .fieldsGrouping("partitioner", new
Fields("key"));
```



digitalPebble

Fetcher Bolts

SimpleFetcherBolt

- Fetch within `execute` method
- Waits if not enough time since previous call to same host / domain / IP
- Incoming tuples kept in Storm queues i.e. outside bolt instance

FetcherBolt

- Puts incoming tuples into internal queues
- Handles pool of threads : poll from queues
  - `fetcher.threads.number`
- If not enough time since previous call : thread moves to another queue
- Can allocate more than one fetching thread per queue

Next : Parsing

- Extract text and metadata (e.g. for indexing)
  - Calls ParseFilter(s) on document
  - Enrich content of Metadata

- ParseFilter abstract class

```
public abstract void filter(String URL, byte[] content, DocumentFragment doc, ParseResult parse);
```

- Out-of-the-box:
  - ContentFilter
  - DebugParseFilter
  - LinkParseFilter : `//IMG[@src]`
  - XPathFilter

- Configured via JSON file

ParseFilter Config => resources/parsefilter.json

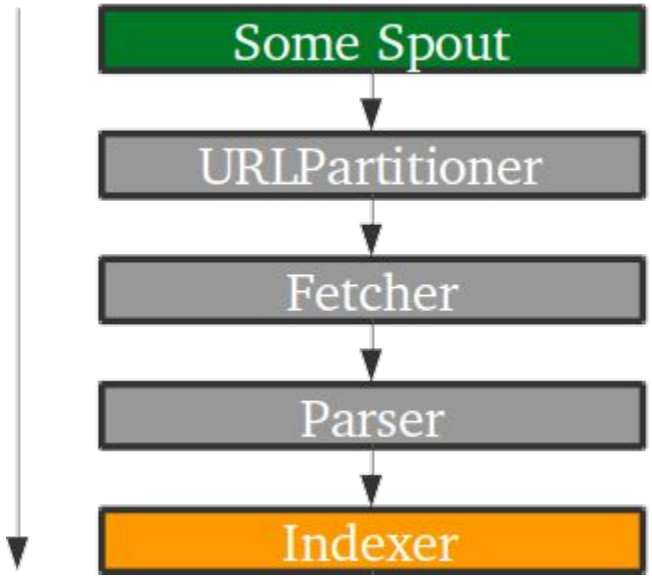
```
{
 "com.digitalpebble.stormcrawler.parse.ParseFilters": [
   {
     "class": "com.digitalpebble.stormcrawler.parse.filter.XPathFilter",
     "name": "XPathFilter",
     "params": {
       "canonical": "//*[@rel=\"canonical\"]/@href",
       "parse.description": [
           "//*[@name=\"description\"]/@content",
           "//*[@name=\"Description\"]/@content"
        ],
        "parse.title": "//META[@name=\"title\"]/@content",
        "parse.keywords": "//META[@name=\"keywords\"]/@content"
     }
   },
   {
     "class": "com.digitalpebble.stormcrawler.parse.filter.ContentFilter",
     "name": "ContentFilter",
     "params": {
       "pattern": "//DIV[@id=\"maincontent\"]",
       "pattern2": "//DIV[@itemprop=\"articleBody\"]"
      }
   }
 ]
}
```

Basic topology in code

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

builder.setBolt("fetch", new FetcherBolt(),3)
    .fieldsGrouping("partitioner", new Fields("key"));

builder.setBolt("parse", new JSoupParserBolt())
    .localOrShuffleGrouping("fetch");

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

localOrShuffleGrouping : stay within same worker process => faster de/serialization + less traffic across nodes (byte[] content is heavy!)

Summary : a basic topology



(1) pull URLs from an external source

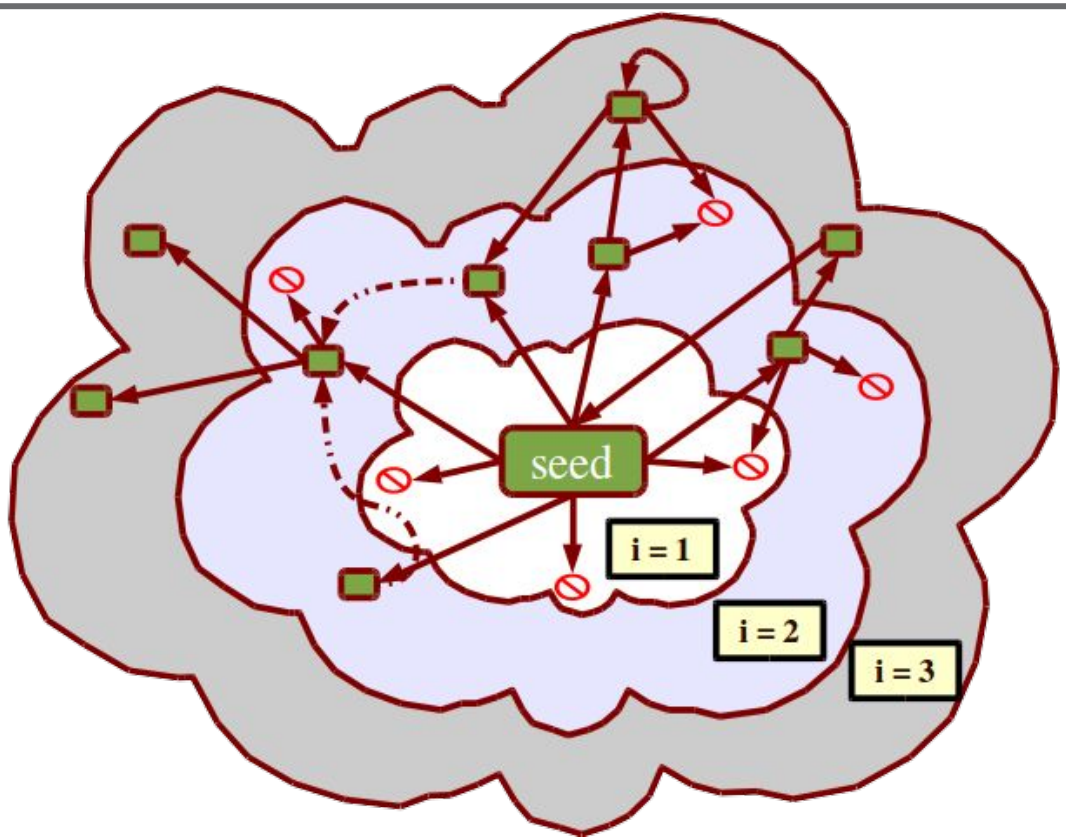
(2) group per host / domain / IP

(3) fetch URLs

(4) parse content

(5) index text and metadata

Frontier expansion



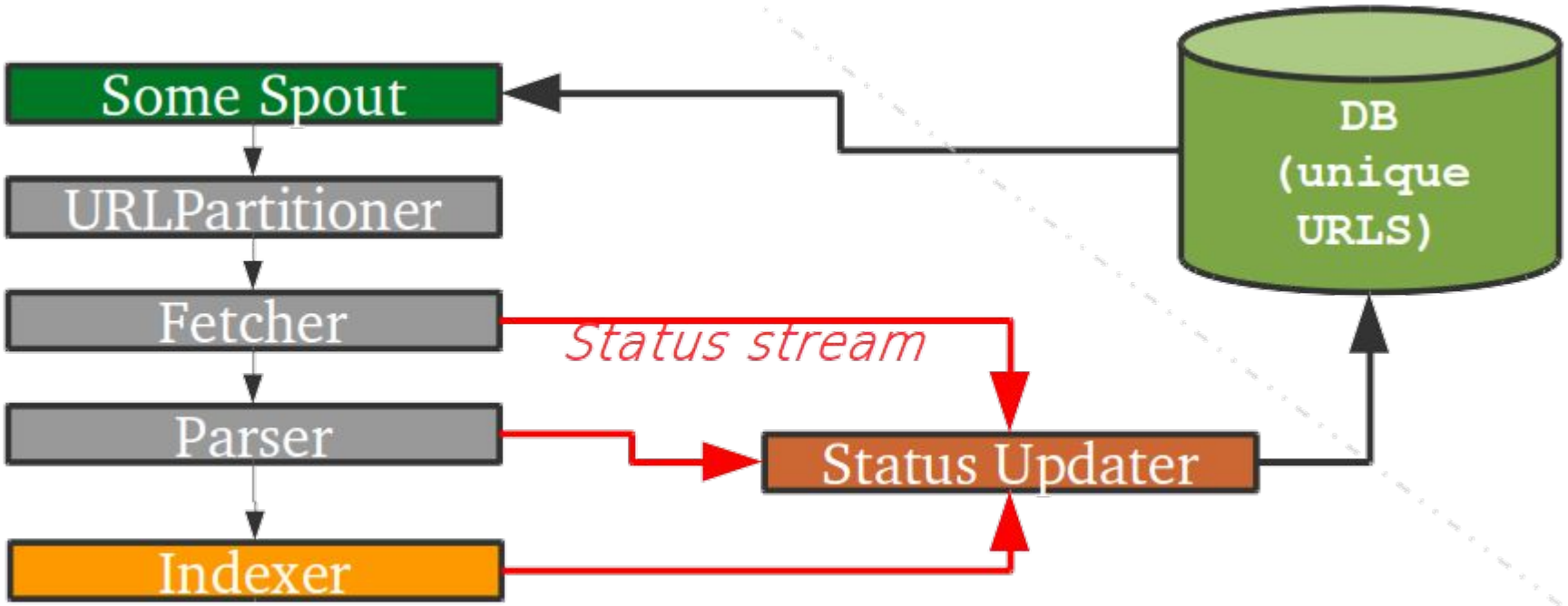
Outlinks

- Done by ParserBolt
- Calls *URLFilter(s)* on outlinks to normalize and / or blacklists URLs
- *URLFilter* interface

```
public String filter(URL sourceUrl, Metadata sourceMetadata, String urlToFilter);
```

- Configured via JSON file
- Loads of filters available out of the box
    - Depth, metadata, regex, host, robots ...

Status Stream



Status Stream

```
Fields furl = new Fields("url");
builder.setBolt("status", new StdOutStatusUpdater())
       .fieldsGrouping("fetch", Constants.StatusStreamName, furl)
       .fieldsGrouping("parse", Constants.StatusStreamName, furl)
       .fieldsGrouping("index", Constants.StatusStreamName, furl);
```

Extend AbstractStatusUpdaterBolt.java

- Provides an internal cache (for DISCOVERED)
- Determines which metadata should be persisted to storage
- Handles scheduling
- Focus on implementing `store(String url, Status status, Metadata metadata, Date nextFetch)`

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

Which backend?

- Depends on your scenario :
  - Don't follow outlinks?
  - Recursive crawls? i.e can I get to the same URL in more than one way?
  - Refetch URLs? When?

- queues, queues + cache, RDB, key value stores, search

- Depends on rest of your architecture
  - SC does not force you into using one particular tool

Make your life easier #1 : ConfigurableTopology

- Takes YAML config for your topology (no more ~/.storm)
- Local or deployed via *-local* arg (no more coding)
- Auto-kill with *-ttl* arg (e.g. injection of seeds in status backend)
- Loads `crawler-default.yaml` into active conf
  - Just need to override with your YAML
- Registers the custom serialization for Metadata class

ConfigurableTopology

```
public class CrawlTopology extends ConfigurableTopology {

   public static void main(String[] args) throws Exception {
       ConfigurableTopology.start(new CrawlTopology(), args);
   }

   @Override
   protected int run(String[] args) {
      TopologyBuilder builder = new TopologyBuilder();
       // Just declare your spouts and bolts as usual
       return submit("crawl", conf, builder);
   }
}
```

Build then run it with

```
storm jar target/${artifactId}-${version}.jar ${package}.CrawlTopology -conf crawler-conf.yaml -local
```

Make your life easier #2 : Archetype

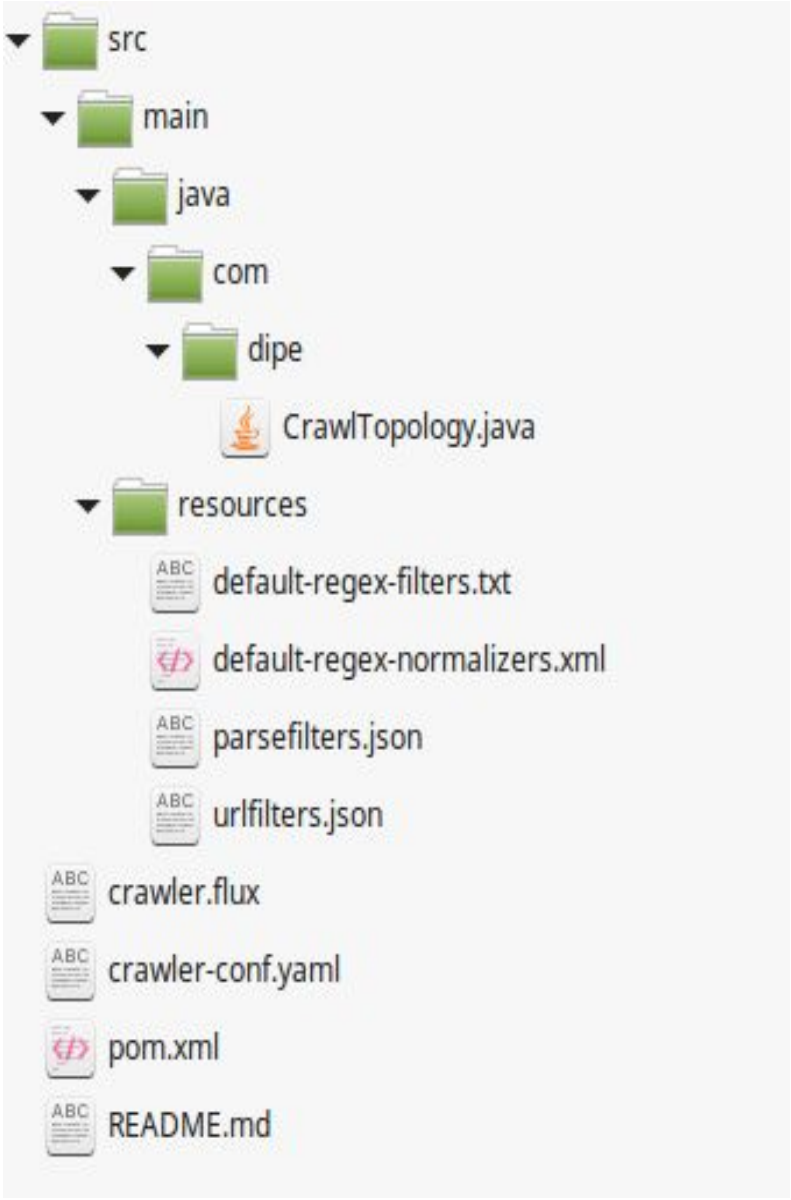
Got a working topology class but :

- Still have to write a pom file with the dependencies on Storm and SC
- Still need to include a basic set of resources
  - URLFilters, ParseFilters
- As well as a simple config file

Just use the archetype instead!

```
mvn archetype:generate -DarchetypeGroupId=com.digitalpebble.stormcrawler
-DarchetypeArtifactId=storm-crawler-archetype -DarchetypeVersion=1.2
```

And modify the bits you need


src
main
java
com
dipe
CrawlTopology.java
resources
default-regex-filters.txt
default-regex-normalizers.xml
parsefilters.json
urlfilters.json
crawler.flux
crawler-conf.yaml
pom.xml
README.md


38

Make your life easier #3 : Flux

http://storm.apache.org/releases/1.0.2/flux.html

Define topology and config via a YAML file

```
storm jar mytopology.jar org.apache.storm.flux.Flux --local config.yaml
```

Overlaps partly with *ConfigurableTopology*

Archetype provides a Flux equivalent to the example Topology
Share the same config file

For more complex cases probably easier to use *ConfigurableTopology*

External modules and repositories

- Loads of useful things in *core*
  => *sitemap, feed parser bolt, …*
- External
  - Cloudsearch (indexer)
  - Elasticsearch (status + metrics + indexer)
  - SOLR (status + metrics + indexer)
  - SQL (status)
  - Tika (parser)
  - WARC

elasticsearch

- Version 2.4.1
- Spout(s) / StatusUpdater bolt
  - Info about URLs in *'status'* index
  - Shards by domain/host/IP
  - Throttle by domain/host/IP
- Indexing Bolt
  - Docs fetched sent to *'index'* index for search
- Metrics
  - DataPoints sent to *'metrics'* index for monitoring the crawl
  - Use with Kibana for monitoring

Use cases

weborama

- Streaming URLs
- Queue in front of topology
- Content stored in HBase

stolencamerafinder

"Find images taken with camera X since Y"

- URLs of images (with source page)
- Various sources : browser plugin, custom logic for specific sites
- Status + metrics with ES
- EXIF and other fields in bespoke index
- Content of images cached on AWS S3 (>35TB)
- Custom bolts for variations of topology : e.g. image tagging with Python bolt
  - using Tensorflow

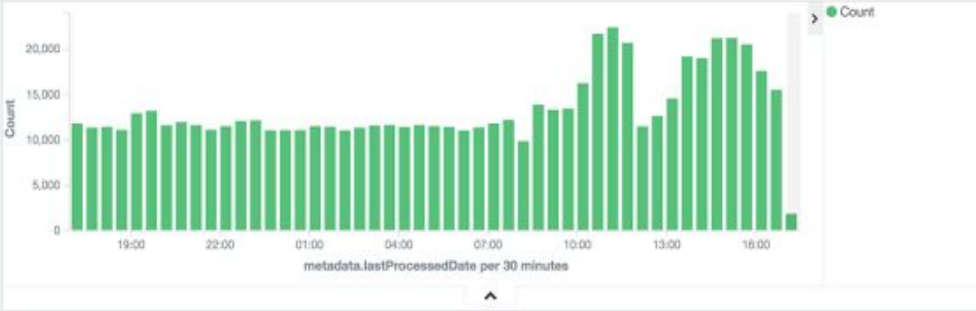
kibana Discover Visualize Dashboard Settings

Status Dashboard *

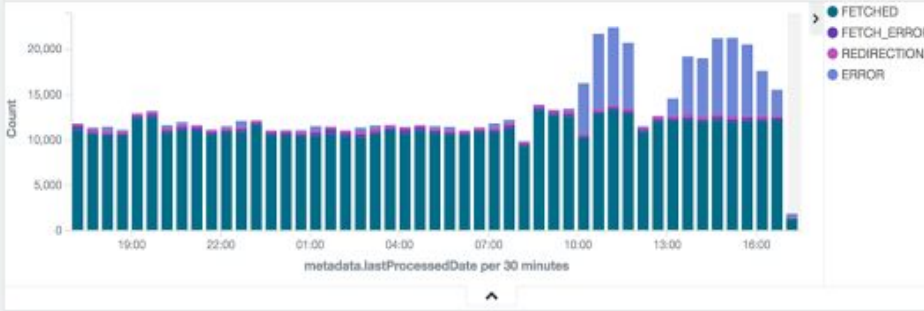
Crawler Notice

Welcome - note the Crawler Journal records config and development changes that may be useful when interpreting changes in the data.

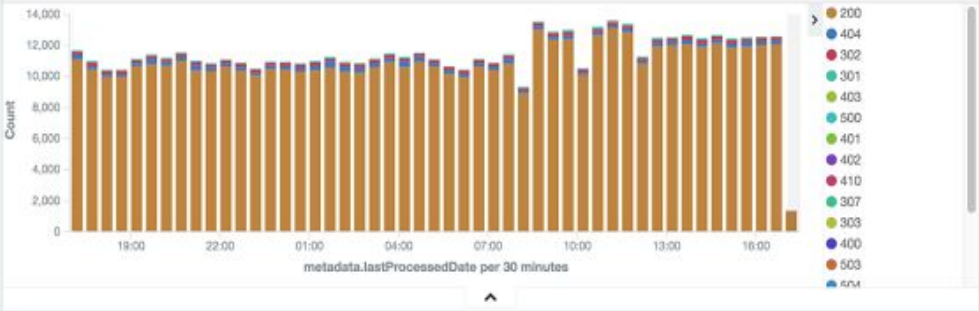
Status - raw processed



Status - processed (by status)



Status - processed (by status code)

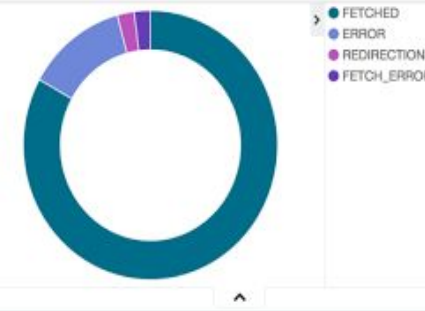


Processed Status URLs

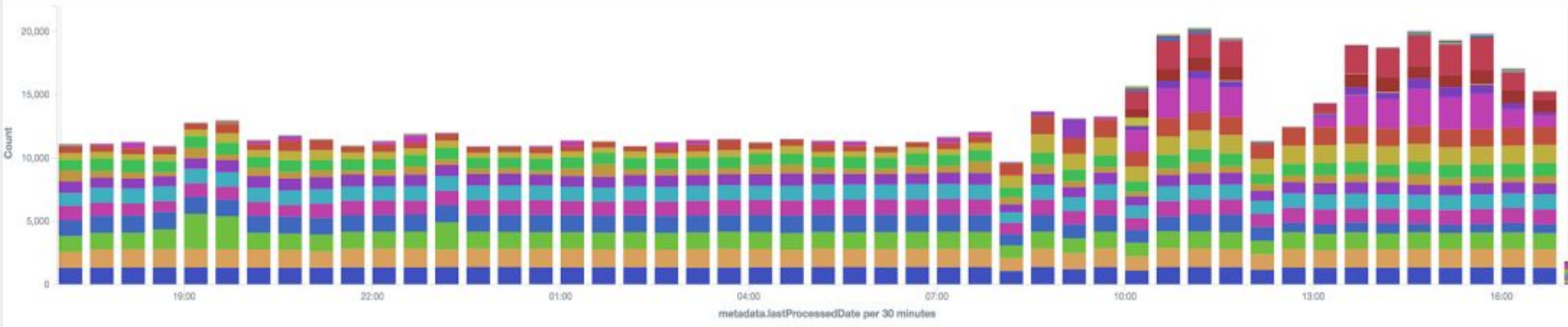
655,213

PROCESSED URLs

Processed Status



Status - processed (by hostname)



Common Crawl

News crawl

7000 RSS feeds <= 1K news sites

50K articles per day

Content saved as WARC files on Amazon S3
Code and data publicly available

Elasticsearch status + metrics

WARC module

Next?

- Selenium-based protocol implementation (#144)
- Elasticsearch 5 (#221)
- Language ID (#364)
- ...

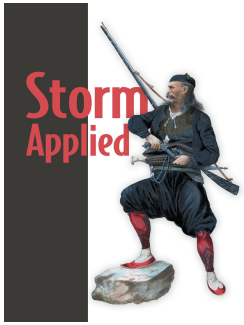
References

http://stormcrawler.net

https://github.com/DigitalPebble/storm-crawler/wiki

http://storm.apache.org

Storm Applied (Manning)
http://www.manning.com/sallen/




Thank you