Optimizing Apache Nutch For Domain Specific Crawling at Large Scale

Luis A. Lopez, Ruth Duerr, Siri Jodha Singh Khalsa
luis.lopez@nsidc.org
http://github.com/b-cube
IEEE Big Data 2015, Santa Clara CA.
Overview

- **BCube** is a building block of NSF’s EarthCube, for the past 6 months we’ve been crawling the Internet trying to gather all possible data that’s relevant to the geosciences.

- Our focus is to discover **scientific datasets and web services** that may contain geolocated data. Mainly structured information (xml, json, csv)
Understanding the problem [Focused Crawling]

Big Search Space With Very Sparse Data Distribution

- Billions of web pages
- Most content is not scientific data
- Scientific data is not well advertised

Solution

A good(enough) scoring algorithm

Acceptable Performance With Limited Resources

- Scalable stack
- Handles TB of data
- Distributed processing
- Fault tolerant
- Uses commodity hardware

Apache Nutch!

... Hard Problems

- Content Duplication
- Semantics
- Robots.txt
- Remote Servers Performance
- Malformed Metadata
- Bad Web Standards Implementation
- Cost
<table>
<thead>
<tr>
<th>Previous Work</th>
<th>Our Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Finding the scoring algorithm that performs 10% better</td>
<td>● To understand where the bottlenecks are</td>
</tr>
<tr>
<td>● Implementing an in house (not open sourced) crawler</td>
<td>● To use an open source project</td>
</tr>
<tr>
<td>● Focusing on a specific type of data</td>
<td>● To improve fetching times</td>
</tr>
<tr>
<td>● Measuring performance on thousands of pages (sometimes just hundreds)</td>
<td>● To modify the crawler for focused crawls</td>
</tr>
<tr>
<td></td>
<td>● To use “the cloud” to lower operational costs</td>
</tr>
<tr>
<td></td>
<td>● To verify what happens at large scale</td>
</tr>
</tbody>
</table>
Scoring

PageRank Like Scoring

Focused Crawl Scoring
BCube Customizations

Nutch

1. Injector
2. Generator
3. Fetcher
4. Parser
5. Updater
6. Link Inverter
7. LinkDB
8. CrawlDB

Solr

6. Indexer (Lucene)
7. Index

filters, normalizers, plugins (Nutch)
BCube Plugins

**Parse-rawcontent**
Indexes the unparsed content of a document

**parse-bayes**
Scores pages using an online naive-bayes classifier

**index-xmlnamespaces**
Indexes all the namespaces used in xml documents

**index-links**
Indexes inlinks and outlinks of a document

**index-bcube-extras**
Indexes HTTP responses

**index-bcube-filter**
Discards documents using mime types or substring matching

**parse-tika**
fixed critical bug that blocked us from parsing valid XML files
BCube Filtering

url regex, suffix

bayes

scoring

filters: mime type, url regex.

Fetcher

Nutch DB

Lucene
Problems...
Performance Degradation

- Crawl-Delay
- Sparse data distribution
- Duplicated content
- Slow servers
- “The tar pits”
- Variable cluster performance in the cloud
- Idle CPU time

https://wiki.apache.org/nutch/OptimizingCrawls
Good Scoring, Let’s Celebrate!... **not so soon.**

The Tar Pit

- Keep indexing relevant documents from the same sites prevented us from discovering new ones.
- Well scored documents are still relevant and we should index them.
- How often these sites are updated should be taken into account.
Content Duplication... at large scale.


<table>
<thead>
<tr>
<th>Domain</th>
<th>Documents fetched</th>
</tr>
</thead>
<tbody>
<tr>
<td>fr.climate-data.org</td>
<td>212142</td>
</tr>
<tr>
<td>bn.climate-data.org</td>
<td>209257</td>
</tr>
<tr>
<td>de.climate-data.org</td>
<td>203279</td>
</tr>
<tr>
<td>en.climate-data.org</td>
<td>197716</td>
</tr>
</tbody>
</table>
Improving Performance

- Robots.txt
- Crawl-Delay
- Large files:
  - .ISO .HDF etc.
- SSD vs HDD
- Fetching Strategy
- Crawling at different speeds.
- Filtering out file extensions using Nutch’s suffix-regex filter
- SSD instances on AWS
- Generate a limited number of links per host and distribute the fetch on as many nodes as possible.
## Nutch + BCube

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value</th>
<th>BCube Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>crawldb.url.filters</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>db.update.max.inlinks</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>db.injector.overwrite</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>generate.max.count</td>
<td>-1</td>
<td>1000</td>
</tr>
<tr>
<td>fetcher.server.delay</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>fetcher.threads.fetch</td>
<td>10</td>
<td>128</td>
</tr>
<tr>
<td>fetcher.threads.per.queue</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>fetcher.timelimit.mins</td>
<td>-1</td>
<td>45</td>
</tr>
</tbody>
</table>

![Vanilla vs BCube Indexing Performance](image)
Conclusions and Future Work

- There are major issues in focused crawls that can only be reproduced at large scale.
- Some issues cannot be addressed by improving the focused crawl alone.
- We can implement mitigation techniques effectively to alleviate the problems under our control.
- Apache Nutch can scale and be used for focused crawls.
- Optimize scoring algorithm using the link graph and content context.
- Develop a computationally efficient mechanism for dynamic relevance adjustment.
- Automate cost effective cluster deployments.
- Use the latest selenium plugins in Nutch for specific use cases.
- More...
References

BCube at Github

https://github.com/b-cube

Apache Nutch

https://nutch.apache.org/

Common Crawl Project

https://commoncrawl.org/

The Science of Crawl: Deduplication of Web Content

http://bit.ly/1Gg32Hh