Notes 12: Distributed Information Retrieval
Web Search Engine

Crawling
Indexing
Computing ranking features
Serving queries
Crawling

Fetch the content of web pages

Web

Initialize

Get the next URL

Fetch page

Extract URLs

Seed URLs

URLs to visit

Visited URLs

Web pages
Crawling Issues

Scope and freshness
- Not enough space/time to crawl “all” pages
- Page importance, quality, and update frequency
- Site mirrors and (near) duplicate pages
- Dynamic content and crawler traps

Load at visited web sites
- Rules in robots.txt
- Limit number of visits per day
- Limit depth of crawl
Crawling Issues

Load at crawler

Variance of fetch latency/bandwidth

*Parallelization and scalability*

Multiple agents
Partitioning URL lists
Communication between agents
Recovering from agent failure
Crawl Partitioning

Requirements
Each URL assigned to a single agent
Locally computable URL-to-agent mapping
Balanced distribution of URLs across agents
Contravariance
Contravariance

Agent A
url₁
url₃
url₅

Agent B
url₂
url₄
url₆

Agent A →
url₁
url₂

Agent B
url₃
url₄

Agent C
url₅
url₆
Contravariance

Agent A
\[
\begin{array}{c}
url_1 \\
url_3 \\
url_5 \\
\end{array}
\]

Agent B
\[
\begin{array}{c}
url_2 \\
url_4 \\
url_6 \\
\end{array}
\]

Agent A
\[
\begin{array}{c}
url_1 \\
\text{url}_2 \\
\end{array}
\]

Agent B
\[
\begin{array}{c}
url_3 \\
url_4 \\
\end{array}
\]

Agent C
\[
\begin{array}{c}
url_5 \\
url_6 \\
\end{array}
\]

Agent A
\[
\begin{array}{c}
url_1 \\
url_3 \\
\end{array}
\]

Agent B
\[
\begin{array}{c}
url_2 \\
url_4 \\
\end{array}
\]

Agent C
\[
\begin{array}{c}
url_5 \\
url_6 \\
\end{array}
\]
URL Assignment

Consistent hashing
Hash function: URL → agent
Each agent “replicated” $k$ times
Each replica mapped randomly on unit circle
Mapping persistent across agent restarts
Lookup: map URL on unit circle; find closest live replica
URL Assignment

A

url₆

B

A

B
URL Assignment

Balancing ✔
Contravariance ✔
Crawl Partitioning

Ideas
URL normalization
   E.g., relative to absolute URL
Host-based partitioning
   Reduces communication between agents
   Small vs. large hosts
Geographic distribution
Fault Tolerance

Repartitioning ✔
Permanent failures
  Recovering list of URLs to visit
  Checkpoints
  Communication logs
Transient failures
  Avoiding re-visiting URLs
  E.g., before fetch, check with near neighbor agents
Web Search Engine

Crawling ✔
Indexing
Computing ranking features
Serving queries
Indexing

Build the term-document index

Collection

Lexicon

Posting for \( t_1 \)
Architecture

Web pages → Distributors → Map → Indexers → Intermediate runs → Reduce

Reduce outputs:
- Inverted index files
- Query servers
Indexing Issues

Index partitioning
Efficient query processing
   Query routing
   Result retrieval
Document Partitioning
Document Partitioning

Split the collection of documents

**Advantages**
- Easy to add new documents
- Load balanced
- High processing throughput

**Disadvantages**
- Communication with all query servers
Term Partitioning
Term Partitioning

Split the lexicon

Advantages
Reduced communication with query servers

Disadvantages
More processing before partitioning
Adding new documents is hard
Load balancing is hard
Processing parallelism is limited by query length
Advanced Partitioning

Topical partitioning using clustering
Documents clustered by term-similarity
Partitions made up of one or more clusters

Usage-induced partitioning
Queries extracted from logs
Documents clustered by query-similarity
Partitions made up of one or more clusters
Web Search Engine

Crawling ✔
Indexing ✔
Computing ranking features
Serving queries
Ranking Feature Computation

Parallel/distributed computation tasks
- Text/language processing
- Document classification/clustering
- Web graph analysis
Example: PageRank

Link-based global (query-independent) importance metric

Random surfer model
Start at a random page
With probability $d$, visit a new page following a random link on the current page
With probability $(1 - d)$, restart at a random page

$\Rightarrow$ PageRank score $\sim$ expected fraction of time spent at a page
PageRank Formula

\[ p(x) = d \times \sum_{y \rightarrow x} \frac{p(y)}{\text{out}(y)} + \frac{1 - d}{n} \]
PageRank Formula

\[ p(x) = d \times \sum_{y \rightarrow x} \frac{p(y)}{\text{out}(y)} + \frac{(1 - d)}{n} \]

Out-degree of page y

Probability of random restart at x

PageRank of x

PageRank of y, where y links to x
PageRank Algorithm

\[ i = 0 \]
\[ p[i](x) = (1 - d) / n \]

repeat
  \[ i += 1 \]
  \[ p[i](x) = (1 - d) / n \]
  for all \( y \rightarrow x \)
    \[ p[i](x) += d \times p[i-1](y) / \text{out}(y) \]
until \[ |p[i] - p[i-1]| < \varepsilon \]
PageRank Implementation

Two vectors, current and next

Initialize vectors
Iterate over all pages $y$, distribute PageRank from $\text{current}(y)$ to $\text{next}(x)$ for all links $y \rightarrow x$
current = next, **re-initialize** next
Go back to iteration over pages or stop
PageRank Distribution

MapReduce for each iteration $i$

Map

Take $< y, (\text{current}(y), \text{edges}(y))>$
For each $y \rightarrow x$ in $\text{edges}(y)$
emit $<x, \text{current}(y)/|\text{edges}(y)|>$
Also emit $<y, \text{edges}(y)>$

Reduce

Take $<x, \text{val}>$ and $<x, \text{edges}(x)>$
Sum $(d \times \text{val})$ into $\text{next}(x)$, add $(1 - d)/n$
Emit $<x, (\text{next}(x), \text{edges}(x))>$
PageRank Distribution

\[
\langle y, (\text{current}(y), \text{edges}(y)) \rangle
\]

Map
\[
\langle x, \text{val} \rangle
\]

Reduce
\[
\langle x, (\text{next}(x), \text{edges}(x)) \rangle
\]
Web Search Engine

Crawling ✔
Indexing ✔
Computing ranking features ✔
Serving queries
Query Processing

Locate, retrieve, process, and serve query results
Serving Architecture

Multiple sites connected by WAN
  Site = coordinator + servers + cache

Partitioning
  Parallel processing
  Distributed storage of data
  Based on index partitioning

Replication
  Availability
  Throughput
  Response time
Serving Issues

Routing the query

To sites

E.g., identical sites + routing by dynamic DNS lookup

Within sites

Merging the results

Caching
<table>
<thead>
<tr>
<th>Serving Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Routing</strong></td>
</tr>
<tr>
<td><strong>Document partition</strong></td>
</tr>
<tr>
<td><strong>Term partition</strong></td>
</tr>
</tbody>
</table>
Caching

What to cache?
Query answers
Term postings
Caching

Query terms repeated more frequently than whole queries

What to cache?

Query answers
  → Faster response
Term postings ✔
  → More hits
Caching

What to cache?

Query answers
  → Faster response

Term postings ✔
  → More hits

Query terms repeated more frequently than whole queries

Done as well in practice
Caching

Policy
Terms most frequent in queries
  → High hit ratio
Terms most frequent in documents
  → Requires more cache space (longer postings)
Summary

Crawling
  Partitioning: balancing and contravariance
  Consistent hashing

Indexing
  Document, term, topical, and usage-induced partitioning

Computing ranking features
  PageRank with MapReduce

Serving queries
  Routing queries, merging results, and caching postings