CS 347
Parallel and Distributed Data Processing

Distributed Information Retrieval
Serious Sports Fans Only $1,000,000 in Cash and Prizes!
For serious sports fans only! Play Fantasy Football!

Lycos
It's amazing where Go Get It will get you.

Find: ___ Go Get It

Enhance your search.

New Search  TopNews  Sites by Subject  Top 5% Sites  City Guide  Pictures & Sounds  PeopleFind  Point Review  Road Maps  Software  About Lycos  Club Lycos  Help
How many pages are on the Web?
Web Search Engine

- Crawling
- Indexing
- Computing ranking features
- Serving queries
Crawling

- Fetch content of web pages

Diagram:
- Init
- Get next URL
- Get page
- Extract URLs
- Seed URLs
- URLs to visit
- Visited URLs
- Web pages
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
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
\[
\begin{array}{c}
\text{url}_1 \\
\text{url}_3 \\
\text{url}_5 \\
\end{array}
\]

Agent B
\[
\begin{array}{c}
\text{url}_2 \\
\text{url}_4 \\
\text{url}_6 \\
\end{array}
\]

Agent A
\[
\begin{array}{c}
\text{url}_1 \\
\text{url}_2 \\
\end{array}
\]

Agent B
\[
\begin{array}{c}
\text{url}_3 \\
\text{url}_4 \\
\end{array}
\]

Agent C
\[
\begin{array}{c}
\text{url}_5 \\
\text{url}_6 \\
\end{array}
\]
Contravariance

Agent A

url₁
url₃
url₅

Agent B

url₂
url₄
url₆

Agent A

url₁

Agent B

url₃
url₄

Agent C

url₅
url₆

Agent A

url₁
url₃

Agent B

url₂
url₄

Agent C

url₅
url₆
Assignment

• Consistent hashing
  – Hash function: URL $\rightarrow$ 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
Assignment

A
B
url$_6$
B
A
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 failure
  – Recovering list of URLs to visit
    ▪ Checkpoints
    ▪ Communication logs

• Transient failure
  – Avoiding re-visiting URLs
    ▪ Before fetch, check with near neighbor agents
Indexing

- Build term-document index

<table>
<thead>
<tr>
<th>t₁</th>
<th>d₁</th>
<th>d₂</th>
<th>d₃</th>
<th>d₄</th>
<th>d₅</th>
<th>d₆</th>
<th>d₇</th>
<th>d₈</th>
<th>d₉</th>
<th>d₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>∙</td>
<td></td>
<td>∙</td>
<td></td>
<td></td>
<td></td>
<td>∙</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>∙</td>
<td></td>
<td>∙</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>∙</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>∙</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₅</td>
<td></td>
<td></td>
<td></td>
<td>∙</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₆</td>
<td></td>
<td></td>
<td></td>
<td>∙</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₇</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₈</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Collection

Posting for t₁
Architecture

- Web pages
- Distributors
- Map
- Indexers
- Query servers
- Reduce
- Inverted index files
- Intermediate runs
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 throughput 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
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 $d$, navigate to new page by following a random link on current page
  – With probability $(1 - d)$, restart at a random page

→ PageRank score = expected fraction of time spent at a page
Formula

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

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

- Probability of random restart at x
- Out-degree of page y
- PageRank of y, where y links to x
- PageRank of page x
Algorithm

\[ i = 0 \]
\[ p[i](x) = \frac{(1 - d)}{n} \]

repeat

\[ i += 1 \]
\[ p[i](x) = \frac{(1 - d)}{n} \]

for all \( y \rightarrow x \)

\[ p[i](x) += d \cdot p[i-1](y) / \text{out}(y) \]

until \( |p[i] - p[i-1]| < \varepsilon \)
Implementation

• Two vectors, current and next
• Initialize vectors
• Iterate over all pages \( y \), distribute PageRank from current\( (y) \) to next\( (x) \) for all links \( y \rightarrow x \)
• current = next, re-initialize next
• Go back to iteration over pages or stop
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 \(<x, \text{current}(y) \div | \text{edges}(y) |>\)
  – Also emit \(<y, \text{edges}(y)>\)
• Reduce
  – Take \(<x, \text{val}>\) and \(<x, \text{edges}(x)>\)
  – Sum \((d \cdot \text{val})\) into \(\text{next}(x)\), add \((1 - d) \div n\)
  – Emit \(<x, (\text{next}(x), \text{edges}(x)))>\)
Distribution

Map

Reduce

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

\langle x, \text{val} \rangle

\langle x, \text{val} \rangle

\langle x, (\text{next}(x), \text{edges}(x)) \rangle
Query Processing

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

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

- Partitioning
  - Parallel processing
  - Distributed storage of data
  - E.g., index partitioning

- Replication
  - Availability
  - Throughput
  - Response time
Issues

• Routing the query
  – To sites
    ▪ E.g., identical sites + routing by dynamic DNS lookup
  – Within sites

• Merging the results

• Caching
# Issues

<table>
<thead>
<tr>
<th>Document partition</th>
<th>Routing</th>
<th>Merging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All servers</td>
<td>Results selected by servers; ranking by coordinator</td>
</tr>
<tr>
<td>Term partition</td>
<td>Servers containing query terms</td>
<td>Selection and ranking by coordinator</td>
</tr>
</tbody>
</table>
Caching

• What to cache?
  – Query answers
  – Term postings
Caching

- What to cache?
  - Query answers
    - Faster response
  - Term postings
    - More hits

Query terms repeated more frequently than whole queries
Caching Policy

- Terms most frequent in queries
  → high hit ratio

- Terms most frequent in documents
  → require more cache space
    (longer postings)

- Use static caching based on query/document frequency ratio
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