CS 347
Parallel and Distributed
Data Processing

Distributed
Information Retrieval

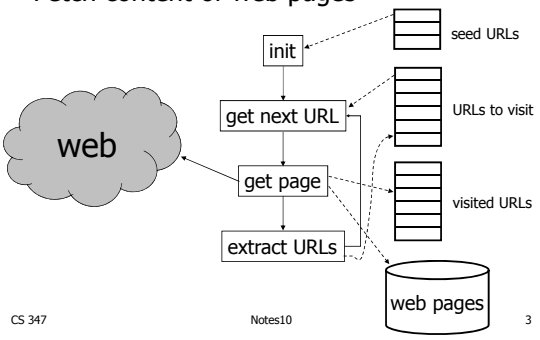
Hector Garcia-Molina
Zoltan Gyongyi

Web Search Engine

- Crawling
- Indexing
- Computing ranking features
- Serving queries

Crawling

- Fetch content of 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

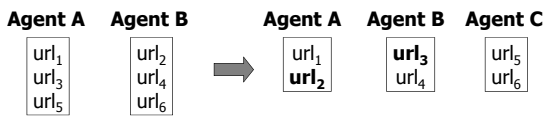
Issues

- Load at crawler
  - Variance of fetch latency/bandwidth
  - **Parallelization and scalability**
    - Multiple agents
    - Partitioning URL lists
    - Communication between agents
    - Recovering from agent failure

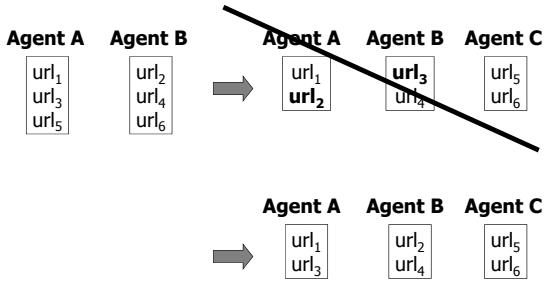
Crawl Partitioning

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

Contravariance



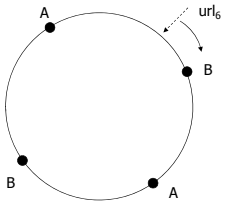
Contravariance



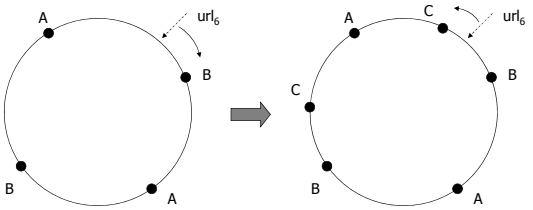
Assignment

- 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

Assignment



Assignment



- Balancing ✓
- Contravariance ✓

Crawl Partitioning

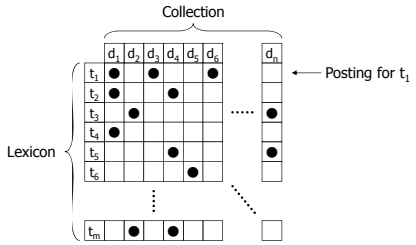
- Ideas
  - URL normalization
    - E.g., relative to absolute URL
  - Host-based partitioning
    - Reduces communication between agents
    - Small vs. large hosts
  - Geographic distribution

Fault Tolerance

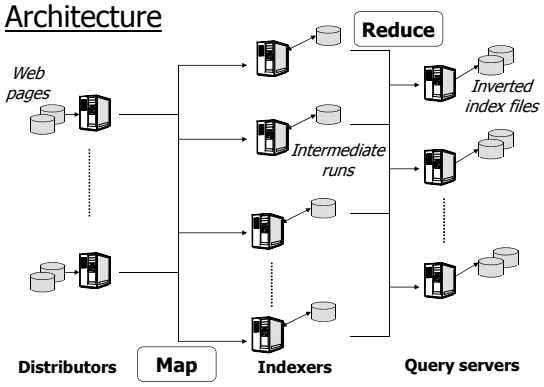
- Repartitioning ✓
- Permanent failure
  - Recovering list of URLs to visit
    - Checkpoints
    - Communication logs
- Transient failure
  - Avoiding re-visiting URLs
    - Before fetch, check with near neighbor agents

Indexing

- Build term-document index



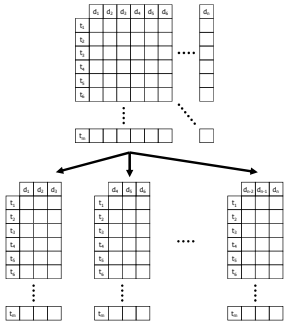
Architecture



Issues

- Index partitioning
  - Efficient query processing
    - Query routing
    - Result retrieval

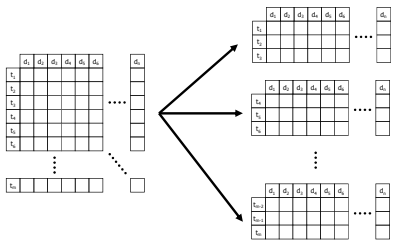
Document Partitioning



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

$$p(x) = d \cdot \sum_{y \to x} p(y) / out(y) + (1 - d) / n$$

Formula

Probability of random restart at x

Out-degree of page y

$$p(x) = d \cdot \sum_{y \to x} p(y) / out(y) + (1 - d) / n$$

PageRank of y, where y links to x

PageRank of page x

Algorithm

$i = 0$

$p^{[i]}(x) = (1 - d) / n$

repeat

  $i \mathrel{+}= 1$

  $p^{[i]}(x) = (1 - d) / n$

  for all $y \to x$

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

until $| p^{[i]} - p^{[i-1]} | < \varepsilon$

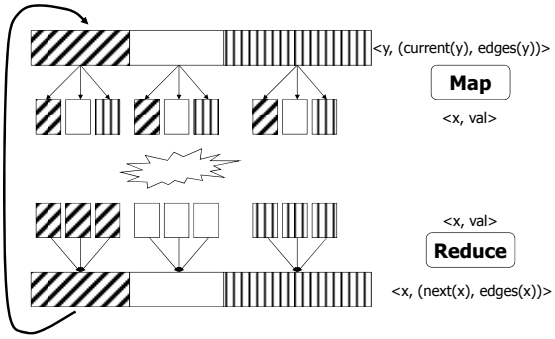
Implementation

- Two vectors, current and next
- Initialize vectors
- Iterate over all pages y, distribute PageRank from current(y) to next(x) for all links y→x
- current = next, re-initialize next
- Go back to iteration over pages or stop

Distribution

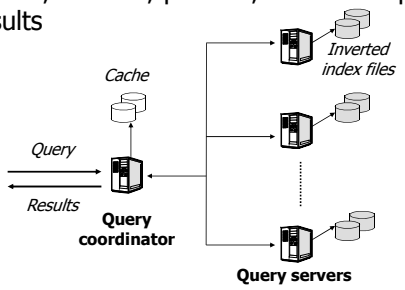
- MapReduce for each iteration i
- Map
  - Take <y, (current(y), edges(y))>
  - For each y→x in edges(y)
    emit <x, current(y) / | edges(y) |>
  - Also emit <y, edges(y)>
- Reduce
  - Take <x, val> and <x, edges(x)>
  - Sum (d · val) into next(x), add (1 – d) / n
  - Emit <x, (next(x), edges(x))>

Distribution



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

	Routing	Merging
Document partition	All servers	Results selected by servers; ranking by coordinator
Term partition	Servers containing query terms	Selection and ranking by coordinator

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 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