

Balancing Volume, Quality and Freshness in Web Crawling

Ricardo Baeza-Yates and Carlos Castillo

Center for Web Research

DCC / UCHILE

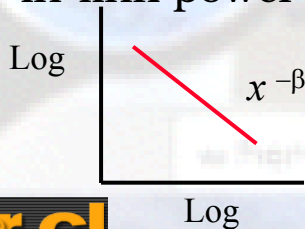
2002

Agenda

- Motivation
- Search Engines
- Web Crawlers
- Crawling
- Crawling problems
- Software architecture
- Implementation details
- Concluding Remarks

The Web

- Web: Largest public repository of *data* (more than 3 billion static pages)
- Today, there are more than 40 million Web servers
- Well connected graph with out-link and in-link power law distributions



Self-similar &
Self-organizing

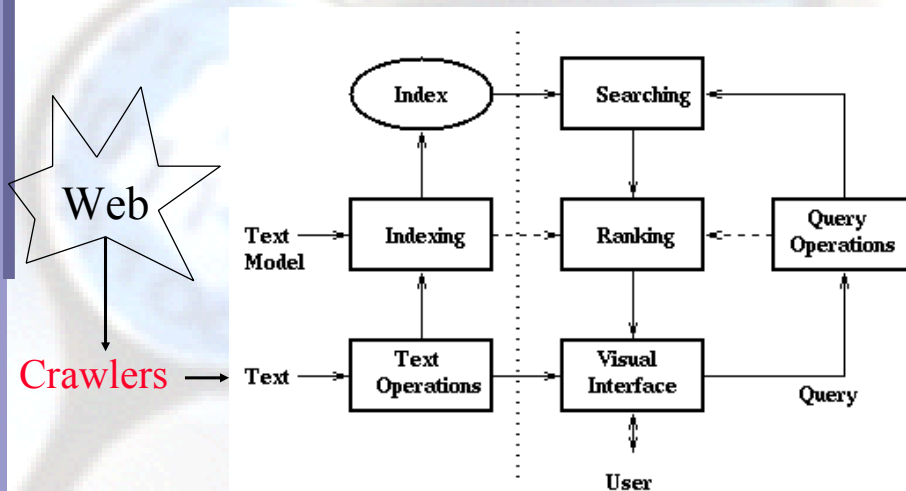
Web Retrieval

- Challenge: find information in this data
- Problems:
 - volume
 - fast rate of change and growth
 - dynamic content
 - redundancy
 - organization and data quality
 - diversity, etc...

Motivation

- Main problem of the Web: scalability
- Search engines are one of the most important tools
- Crawling the Web is the current resource bottleneck
- Our main motivation: WIRE project (Web IR Environment)
- Can we do it better?


Centralized Architectures



Crawling

- Crawlers: page recollection and parsing
- Crawlers do not crawl! They are static
- Bottlenecks: available bandwidth and CPU
- They use the wrong paradigm: pulling
- Pulling vs. pushing

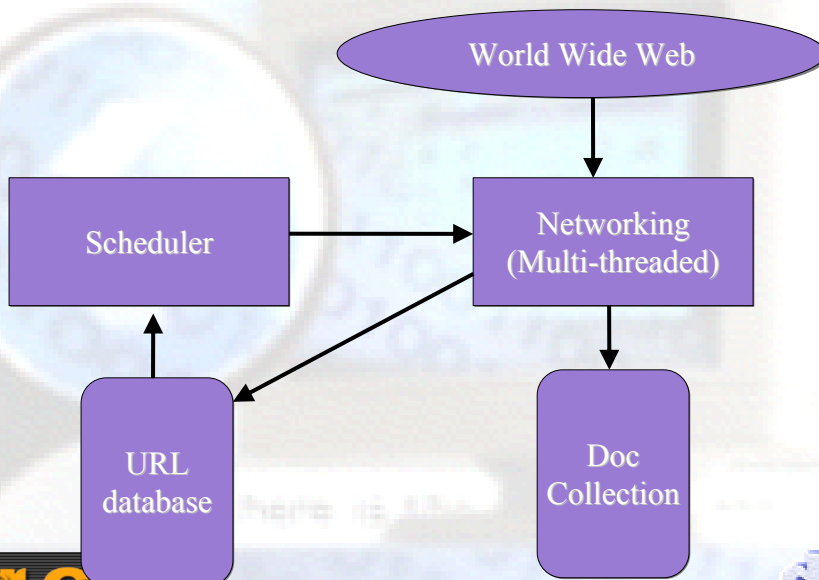
First Generation

- RBSE Spider
 - Web of 100,000 pages
- Crawler of the Internet Archive
- Web Crawler 
- Common aspects:
 - Avoid rejection from webmasters
 - Try to discover new pages, pages are scarce

Second Generation

- Mercator, Sphinx
 - Extensible architectures
 - Multi-protocol, multi-purpose
- Lycos, Excite, Google
 - Distributed (farm) crawlers
- Parallel crawlers

Standard Architecture



Main Goals

They depend on the **index** of a search engine:

- The index should contain a large number of objects that are **interesting** for the users
- The index should **accurately represent** a real object on the Web
- Generate a representation that **captures** the most significant aspects of the object using the **minimum** amount of resources

Main Issues

Interesting object?

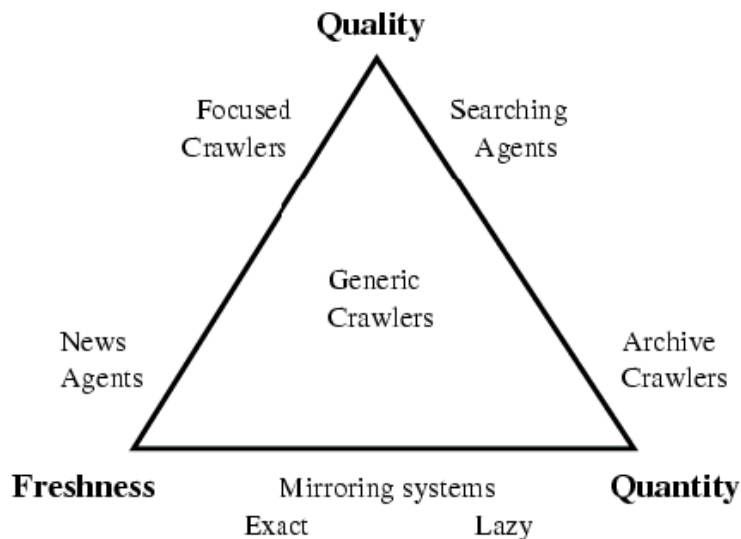
- **Quality**: intrinsic semantic value of the object. Possible estimators:
 - content analysis (e.g. vector model)
 - link analysis (e.g. Pagerank)
 - similarity to a driven query (focused crawlers, searching agents)
 - usage popularity
 - location based (depth, geography, etc.)

Main Issues

Accurate content of the objects?

- **Quantity:** as many objects as possible
 - Accurate content depends on the size and format of the representation
 - Accurate object through time?
- **Freshness:** how recent is the object
 - Web updates are common, half of the Web is less than 6 months old
 - Freshness can be estimated for most Web servers

Taxonomy



Conflicting Goal

- Refreshing a page or bringing a new page?
 - However new pages are only found in updated or new pages
 - Trade-off between quantity (more objects) and quality-freshness (interesting and up-to-date objects)
- Secondary Goals
 - Resource usage and politeness

Scheduling

- Keep collection fresh
 - More important pages must be kept fresher
- Use the network to the maximum extent
- Do not overload servers (robot politeness)
- Discover new pages (updated pages first)
- All this depends on the type of crawler

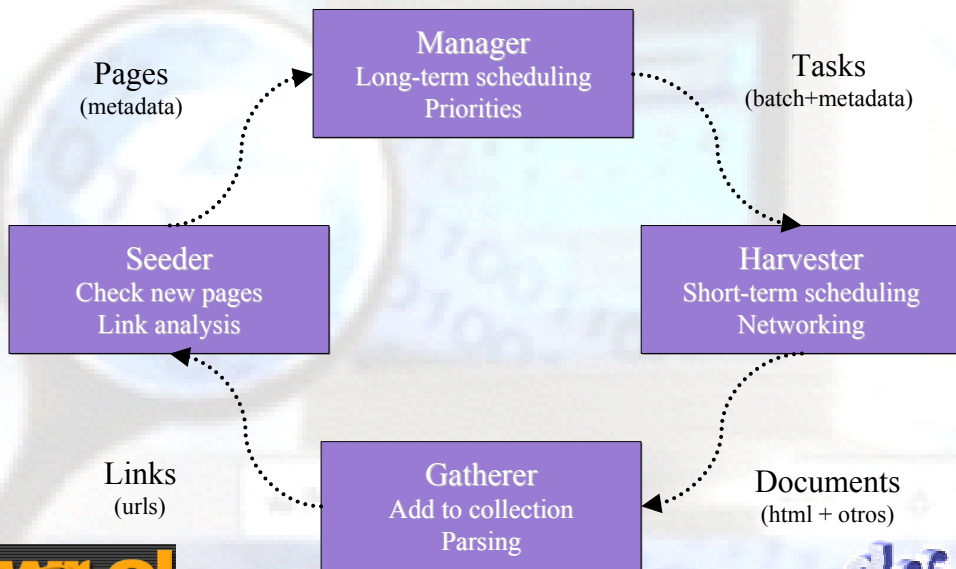
Technical Problems

- DNS is a bottleneck
- Duplicates, slow servers
- Crawler traps
- Dynamic pages and session-ids in URLs
- Threads implementations
 - For example, for constant $C = p k t$ which is the optimal number of processors p running k processes with t threads each?
 - Simulation is difficult, experimenting too

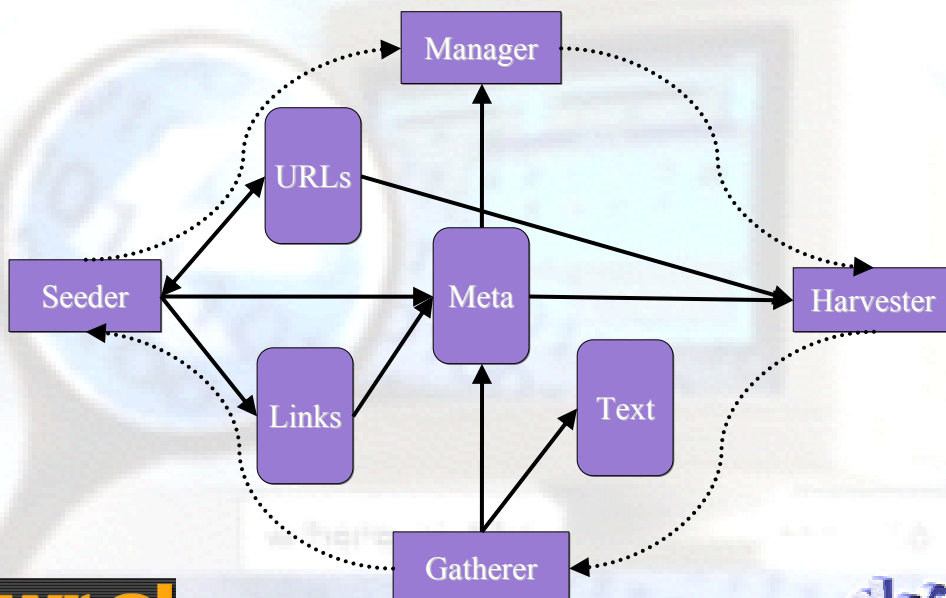
Our Approach

- Goals depend on the index knowledge
 - crawlers should be tightly integrated with the index
- Page scheduling:
 - Long-term: quality-freshness vs. quantity
 - Short-term: resource usage and politeness
- Approach that we use in WIRE

High-Level Architecture



Indices



Non Trivial

- Share-nothing distributed architecture
- Performance: how good is the result, how fast is accomplished
- No benchmarks, no standard measures
 - Pages per processor cycles, bandwidth, and time?

Manager

Collection

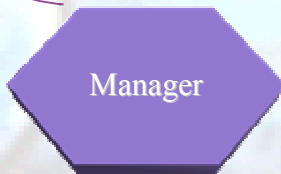
N

score=0.7
freshness=0.9
priority = 0.07

score=0.3
freshness=0.4
priority=0.21

score=0.8
freshness=0.1
priority=0.56

score=0.2
freshness=0.8
priority=0.04



Batch for the harvester

$n \ll N$

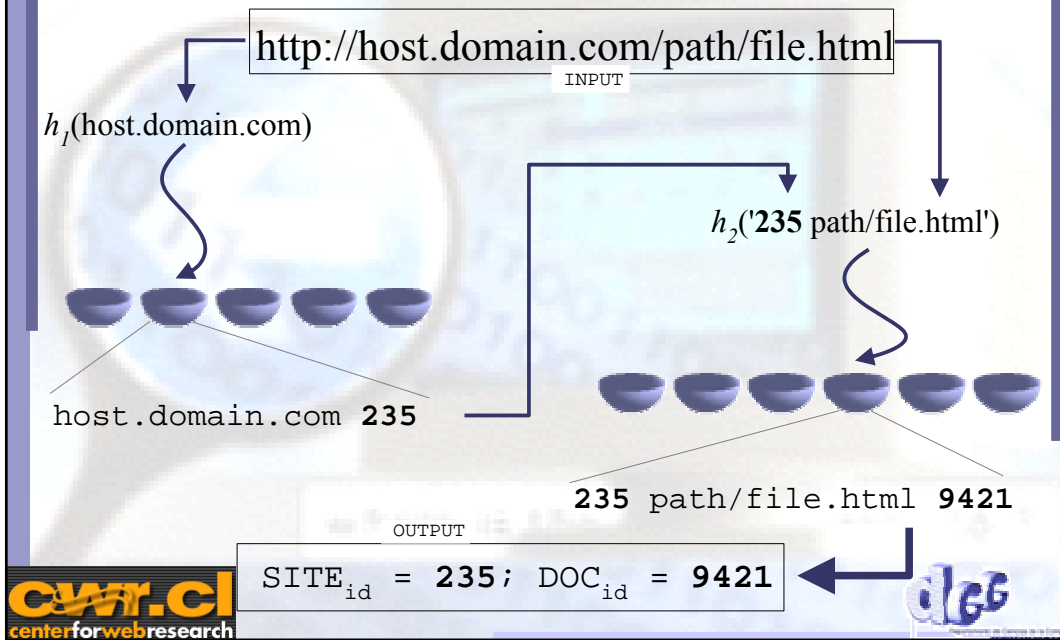
score=0.3
freshness=0.4
priority=0.21

score=0.8
freshness=0.1
priority=0.56

$$\text{priority} = \text{score} \times (1 - \text{freshness})$$

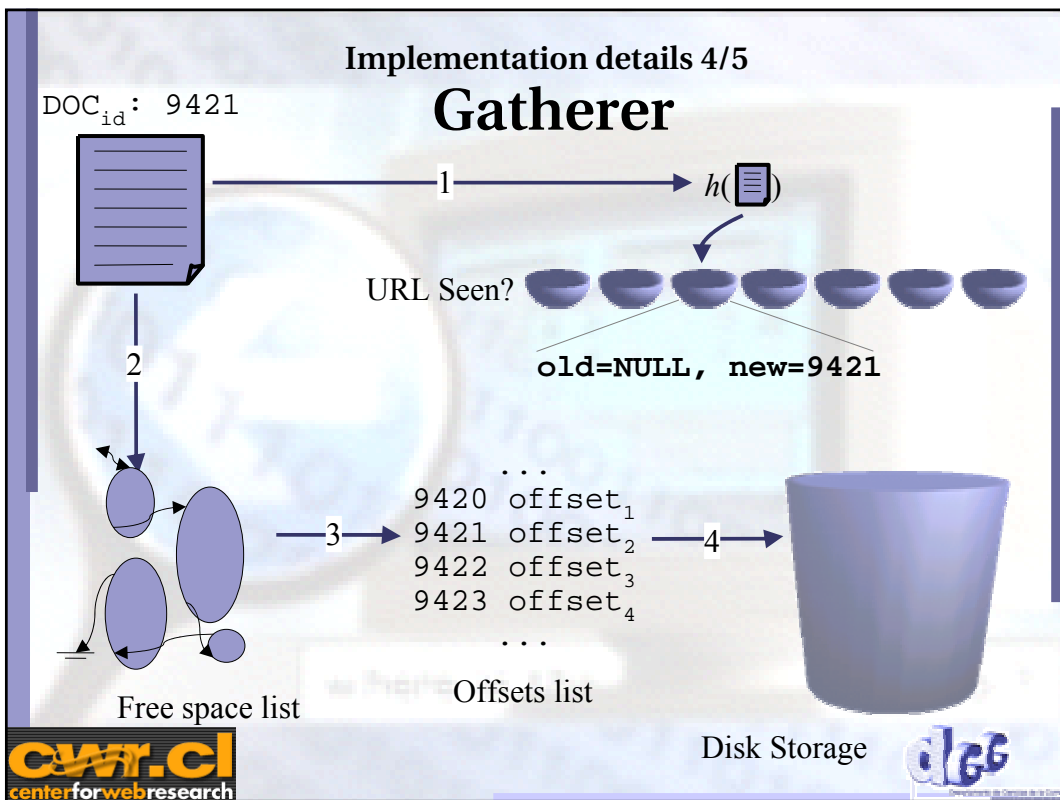
Implementation details 3/5

URL resolving (Seeder)

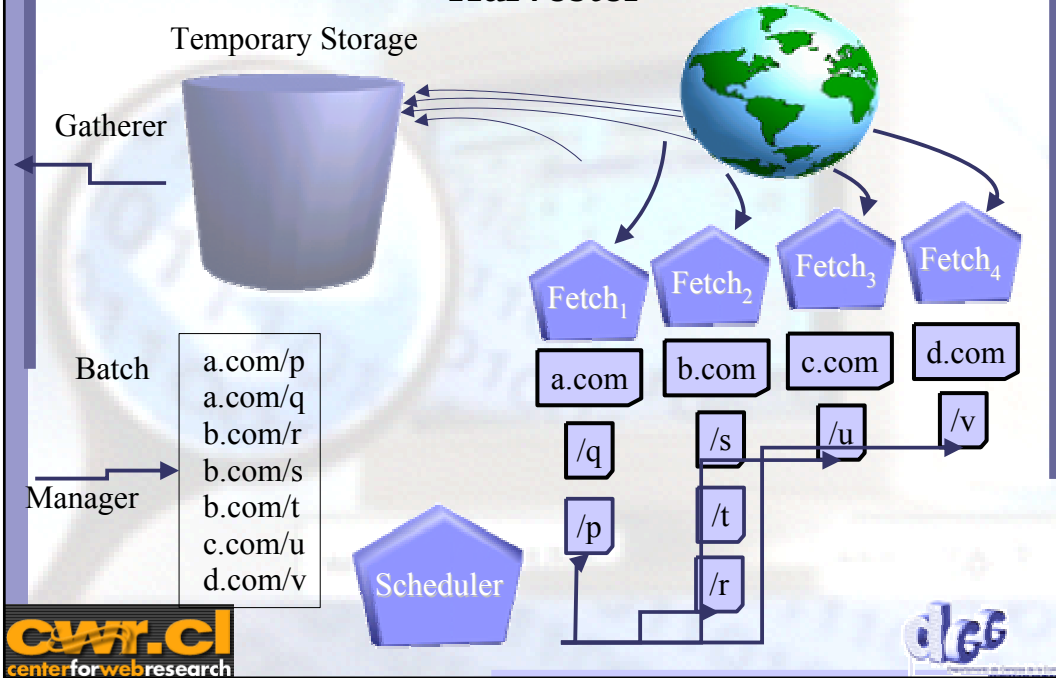


Implementation details 4/5

Gatherer



Harvester



Concluding Remarks

- Prototype currently under evaluation
- Future: Agents and pushing?
- Real networks are dynamic
 - Crawlers do not care about network topology... but could (and should!)
 - Redundancy and fault tolerance is based on randomness
 - How to avoid changing the information market in the wrong (almost any!) direction?

Questions, comments? ...