



Challenges in Distributed Web Retrieval

Ricardo Baeza-Yates^{1,2}
ricardo@baeza.c1

Joint work with: C. Castillo¹, A. Gionis¹, F. Junqueira¹,
V. Murdock¹, V. Plachouras¹ and F. Silvestri³

1. Yahoo! Research Barcelona – Catalunya, Spain
2. Yahoo! Research Latin America – Santiago, Chile
3. ISTI-CNR – Pisa, Italy

1 Motivation

2 Crawling

3 Indexing

4 Query Processing

5 Caching

6 Final Remarks

Challenges in
Distributed Web
Retrieval

Ricardo
Baeza-Yates

Motivation

Crawling

Indexing

Query
Processing

Caching

Final Remarks

- 1 Motivation
- 2 Crawling
- 3 Indexing
- 4 Query Processing
- 5 Caching
- 6 Final Remarks

Web Search

Challenges in
Distributed Web
Retrieval

Ricardo
Baeza-Yates

Motivation

Crawling

Indexing

Query
Processing

Caching

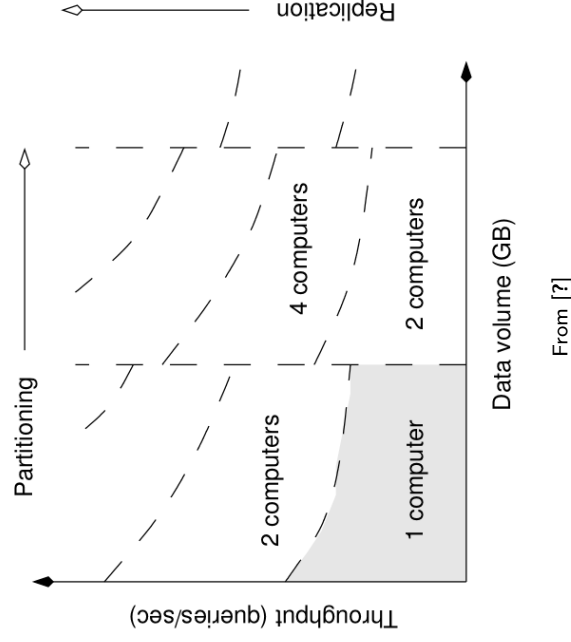
Final Remarks

One of the **most** complex data engineering challenges today:

- Distributed in nature
- Large volume of data
- Highly concurrent service

Current solution: Large replicated centralized architectures

Handling the Scalability Problem



A Quick Computation of System Size

- 20 billion Web pages implies at least 100Tb of text
- The index in RAM implies at least a cluster of 3,000 PCs
- Assume we can answer 1,000 queries per second
- 173 million queries a day imply 2,000 queries per second
- Decide that the peak load plus a fault tolerance margin is 5 times the average
- This load implies a replication factor of 10 giving a total of 30,000 PCs
- Total deployment cost of over 100 million US\$ plus maintenance cost

In 2010, being conservative, we would need over 1 million computers!

Does this make sense?

Solution: A Fully Distributed Search Engine

- Distribution decreases replication and crawling, and hence the cost per query
- We can exploit network topology
- We can exploit high concurrency and locality of queries
- We can use several layers of caching

Main design problems:

- Depends upon many factors that are seldom independent
- One poor design choice can affect performance or/and costs

Main Challenges

The search engine:

- Must return high quality results
(e.g. handle quality diversity and fight spam)
- Must be fast (fraction of a second)
- Must have high capacity
- Must be dependable
(reliability, availability, safety and security)
- Must be scalable

Challenges in Distributed Web Retrieval

Ricardo Baeza-Yates

Motivation

Crawling

Indexing

Query Processing

Caching

Final Remarks

Main Modules and Issues

	Partition	Dependability	Communication	External factors
Crawling	URL assignment	Re-crawl	URL exchanges	Web growth, Content change, Network topology, Bandwidth, DNS, QoS of servers
Indexing	Doc. partition, Term partition	Re-index	Partial indexing, updating, merging	Web growth, Content change, Global statistics
Querying	Query routing, Collection selection, Load balancing	Replication, caching	Rank aggregation, Personalization	Changing user needs, User base growth, DNS

Challenges in Distributed Web Retrieval

Ricardo Baeza-Yates

Motivation

Crawling

Indexing

Query Processing

Caching

Final Remarks

- 1 Motivation
- 2 **Crawling**
- 3 Indexing
- 4 Query Processing
- 5 Caching
- 6 Final Remarks

Crawling

- In theory it is simple: fetch, parse, fetch, parse, ...
- In practice it is difficult: implies using other people's resources (web servers' CPU and network)

Issues

- How to **partition** the crawling task?
- What to do when one agent **fails**?
- How to **communicate** among agents?
- How to deal with **external factors**?

Partitioning

- Host-based partitioning exploits locality of links
- Balance improves if large/small hosts are treated differently
- Performance improves if geographic location is considered (almost no research in this subproblem)

Consistent hashing

Allows to add and remove agents from the pool [Boldi et al., 2004]

Communication

- Host-based partitioning reduces communication
- Highly-linked URLs should be cached
- Communication with the server can be improved if server cooperates

External factors

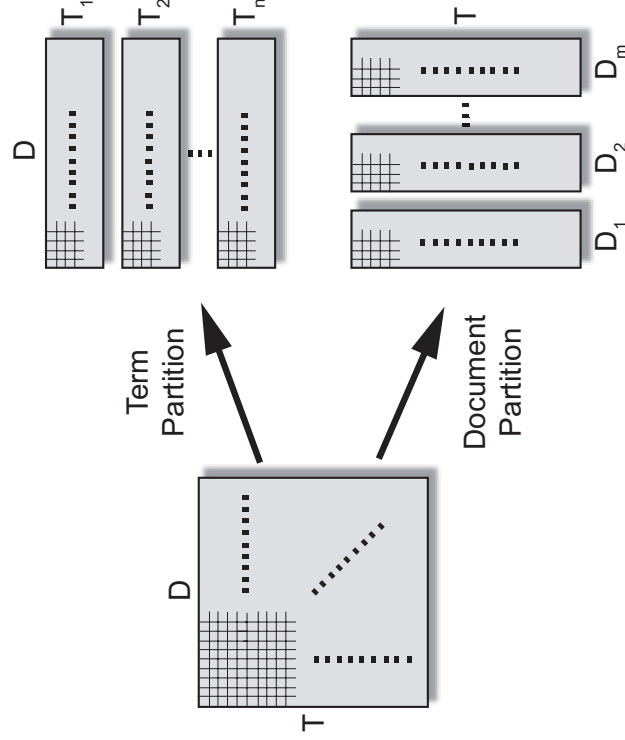
- DNS can be a bottleneck
- Varying quality of implementation of HTTP
- Varying quality of HTML coding
- Varying quality of service in general
- SPAM

- 1 Motivation
- 2 Crawling
- 3 **Indexing**
- 4 Query Processing
- 5 Caching
- 6 Final Remarks

Indexing

- Indexing is the process of building an *index* over data, in this case a collection of Web documents
- *Inverted Indexes* are typically used in IR indexes
 - *Lexicon*: contains distinct terms appearing in the collection's documents
 - *Posting Lists*: contains descriptions of occurrences of relative terms within the corresponding documents

Index and Distributed Indexing



Document Partitioning

- Split the collection into several sub-collections and index each one of them separately (corresponding to vertically slicing the $T \times D$ matrix)
- Queries are sent to all partitions and a broker merges the results
- Pros:
 - higher throughput
 - new documents are easily added to existing indexes
 - parallel load balanced query processing
- Cons:
 - high number of disk operations
 - high volume of data read from disk
 - concurrency is not exploited

Term Partitioning

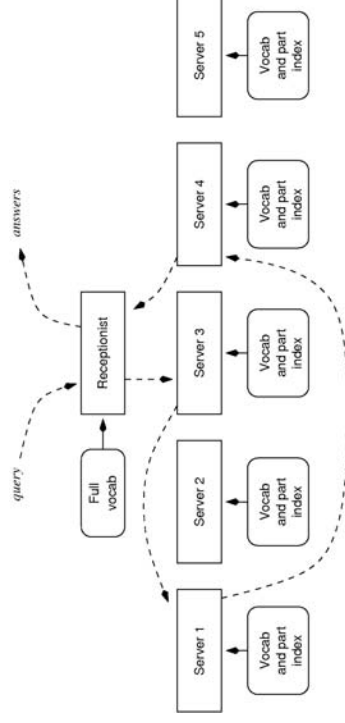
- Split terms of the lexicon and their inverted lists (corresponding to horizontally slicing the $T \times D$ matrix)
- Queries sent only to partitions having the query terms
- Pros:
 - reduced number of disk accesses
 - reduced volume of exchanged data
 - truly concurrent query processing
- Cons:
 - harder to do load balancing
 - require the entire index to be built before slicing it into partitions
 - index is harder to maintain

Not scalable with large collections

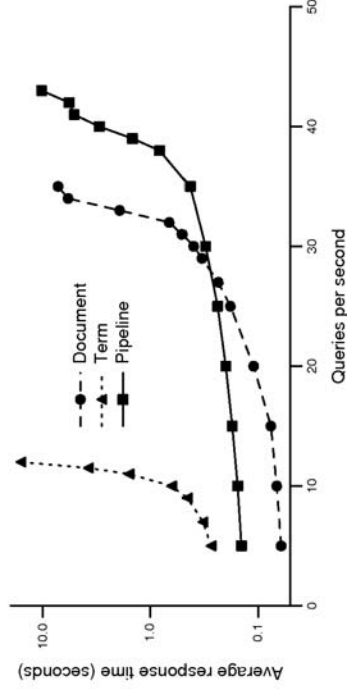
Hence, current Web search systems use document partitioning

Pipelined Query Processing

- Query routing: forward query to a pipeline of servers [Webber et al., 2006]
- Merging of results is done across the pipeline



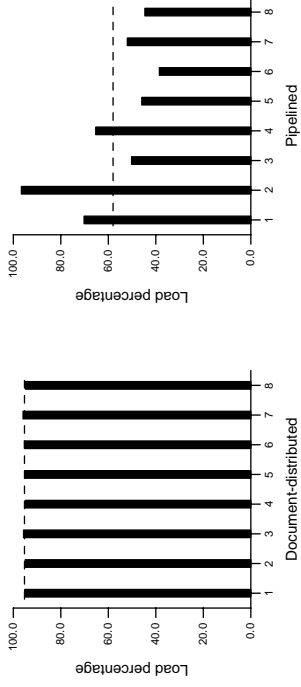
Pipelined Query Processing



There are recent results on how to improve the balance of the query load [Moffat et al., 2006]

Load Balancing Issues

- In document partitioned indexes not adopting collection selection strategies, load is almost balanced among all the query processors
- In term partitioned indexes (even using the new pipelined scheme) load balancing is an issue



- In federated document partitioned systems where collection selection is applied, balancing the load is still an unexplored issue.

Partitioning Goals

- Partitioning is the first design issue to be faced in distributed indexing
- A distributed index should allow for efficient query routing and resolution
- Reducing the number of nodes queried, is desirable too

Partitioning Techniques

- **Random partitioning**
 - documents are assigned uniformly to the partitions
- **Topical organization using clustering**
(e.g. *k*-means [Larkey et al., 2000, Liu and Croft, 2004])
 - documents are firstly clustered and then each partition is composed by one (or more) cluster(s)
- **Usage-induced partitioning**
(e.g. Query-Vector Doc. Model [Puppin et al., 2006])
 - clustering is induced by the way users interact with the index

Exploiting Usage Information

- Query logs contain features that are critical for optimizing efficiency of different parts of search engines
 - query distribution
 - query arrival time
 - click-through information
 - ...

Usage Information in Document Partitioning

- Random partitioning does not ensure load balancing [Badue et al., 2006]
- Broadcast-based systems perform unnecessary operations on sub-collections containing few or no relevant documents
- Usage-based mapping can be adopted to partition sub-collections that can be effectively discriminated upon query reception [Puppin et al., 2006]

Usage Information in Term Partitioning

- Frequency of query terms can be exploited to partition a collection with the aim of balancing the load of query processors
- *Bin-packing* approach [Moffat et al., 2006]
- *Data mining* approach [Lucchese et al., 2007]

Challenges in Distributed Indexing

- In document partitioned system we need to find partitioning strategies for enhancing collection selection performance in terms of effectiveness
- In both document and term partitioned systems it is a challenge to find effective load balancing strategies

Query Processing

System components

- Clients submitting queries
- Sites consisting of servers
- Servers are commodity computers

Query processing

- System receives a query
- Query routing: forwarding query to appropriate sites
- Merging results

Challenges

- Determine appropriate sites on the fly
- WAN communication is costly

Challenges Revisited

Large-scale systems

- Large amount of data
- Large data structures
- Large number of clients and servers

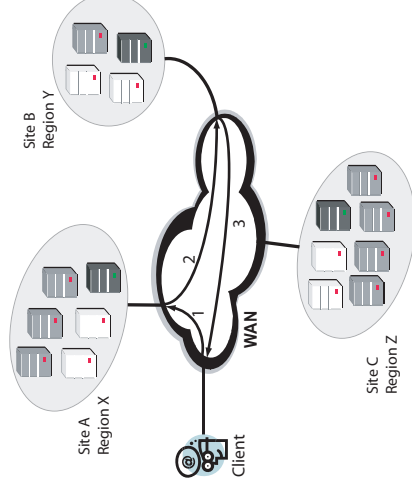
Partitioning of data structures

- Necessary due to very large data structures
- Parallel processing
- e.g. document collection split by topic, language, region

Replication of data structures

- For availability, throughput, and response time
- Conflict with resource utilization

Framework for Distributed Query Processing



- **Query processor** matches documents to the received queries
- **Coordinator** receives queries and routes them to appropriate sites
- **Cache** stores results from previous queries

Currently...

Multiple sites

- Sites are full replicas of each other
- Simple query routing: Dynamic DNS

According to the [previous framework](#), opportunity to

- Use storage resources more efficiently
- More sophisticated query routing mechanisms
- Effective partition strategies (e.g., language-based strategies)

Partitioning

Goals

- Achieve cost-effective scalability
- Reduce response times

Potential solutions

- Partition of large data structures by topic, language, etc.
- Effective query routing first to local sites, then to global sites
- Incremental presentation of results to alleviate network latencies

Dependability

Goals

- Availability of query processors
- Consistency of replicated query data (can be weak)
- Consistency of user state: e.g., personalization, user preferences

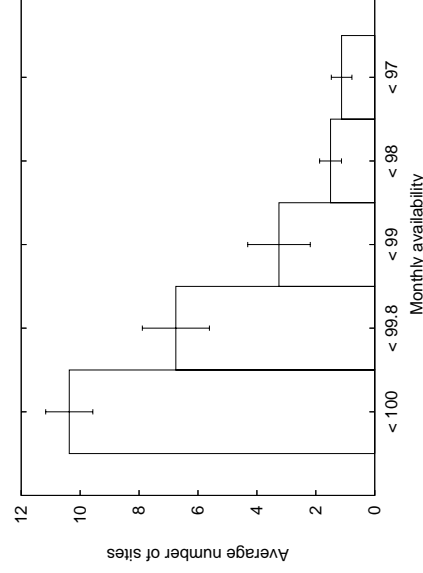
Potential solutions

- More network resources: multi-homed sites
- Replication: within and across sites
- Consistency: techniques for weak consistency (replicas eventually converge)
- Caching: improve availability when query processors are unavailable

Dependability

Achieving availability is not straightforward

- BIRN system studied in [Junqueira and Marzullo, 2005]
- Outages are quite frequent



Communication

Message latency

- Communication is costly in wide-area networks
- Latency is not negligible
- Reduced capacity of servers as the latency to process a query increases

Potential solutions

- Reduce as much as possible the number of sites contacted to process a query
- Most queries processed by sites that are close according to network distance

Caching Query Results or Postings [Baeza-Yates et al., 2007]

Caching query answers:

- 44% of queries are singletons (appear only once)
- 88% of the unique queries are singletons
- Infinite cache would achieve 56% hit-ratio

Caching postings of terms:

- 4% of terms are singletons
- 73% of the unique terms (the vocabulary) are singletons
- Infinite cache would achieve 96% hit-ratio

Note

All statistics and graphs on caching refer to a *one-year* query log from `yahoo.co.uk`

Static or Dynamic Caching of Postings

Static caching of postings (QTF) [?]

- Cache terms with the highest query log frequency $f_q(t)$

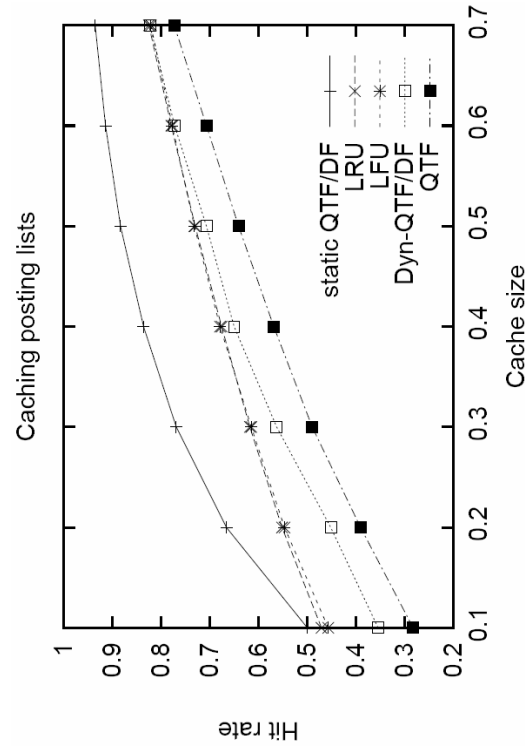
However, there is a tradeoff between $f_q(t)$ and $f_d(t)$

- Terms with high query log frequency $f_q(t)$ are good for the cache
- Terms with high document frequency $f_d(t)$ occupy too much space

Static caching of postings as a KNAPSACK problem (QTFDF)

- Cache posting lists of terms with the highest ratio $\frac{f_q(t)}{f_d(t)}$

Static or Dynamic Caching of Postings?: Static!

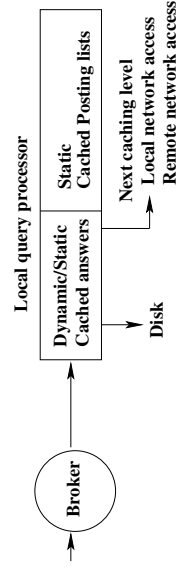


Analysis of Static Caching

Trade-offs between caching postings and answers

- As expected, caching postings results in more hits
- But caching answers is faster
- To compare we need to consider time/space parameters

Problem: Given a fixed amount of memory and the average response times for a system, how much to allocate for caching answers and how much for caching postings?



Analysis of Static Caching

Scenario 1: *Centralized retrieval system, complete/partial query evaluation, un/compressed postings*

- Postings cache can answer more queries than answers cache
- Most available memory for caching postings

Scenario 2: *WAN distributed system, complete/partial query evaluation, un/compressed postings*

- Network time dominates
- Most available memory for caching answers

Query Dynamics

- Slowly changing query dynamics makes static caching viable
- Small dynamic cache to handle query bursts

Challenges in
Distributed Web
Retrieval

Ricardo
Baeza-Yates

Motivation

Crawling

Indexing

Query
Processing

Caching

Final Remarks

- 1 Motivation
- 2 Crawling
- 3 Indexing
- 4 Query Processing
- 5 Caching
- 6 **Final Remarks**

Challenges in
Distributed Web
Retrieval

Ricardo
Baeza-Yates

Motivation

Crawling

Indexing

Query
Processing

Caching

Final Remarks

Plenty of Open Problems

- How much of each cache should be static and how much dynamic?
- Modeling the interaction of the three parts
- Adding distributed replication (now is not independent of the architecture!)
- Having a complete analytical model to explore performance-cost trade-offs

All this without compromising answer quality!

At the same time we must keep a reasonable response time

Questions or remarks?

LSDS-IR

Workshop on Large-Scale Systems for Information Retrieval

- Intersection between large-scale networked systems and information retrieval
- Focus on cutting-edge research ideas spanning the domains of systems and information retrieval
- Will be part of ACM SIGIR'07, Amsterdam
- Organizers:
 - ➊ Flavio Junqueira and Vassilis Plachouras (Yahoo! Research)
 - ➋ Fabrizio Silvestri (ISTI-CNR)
 - ➌ Ivana Podnar (University of Zagreb)
- Submission deadline: June 1, 2007
- More info: <http://www.tel.fer.hr/llds-ir/>



Badue, C., Baeza-Yates, R., Ribeiro-Neto, B., Ziviani, A., and Ziviani, N. (2006).

Analyzing imbalance among homogeneous index servers in a web search system.

Information Processing & Management.



Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Silvestri, F., and Plachouras, V. (2007).

The impact of caching on search engines.

In *Proceedings of the International ACM SIGIR Conference (to appear)*, Amsterdam, Neatherlands.



Boldi, P., Codenotti, B., Santini, M., and Vigna, S. (2004).

Ubicrawler: a scalable fully distributed web crawler.

Software, Practice and Experience, 34(8):711–726.



Junqueira, F. and Marzullo, K. (2005).

Coterie availability in sites.

In *Proceedings of the International Conference on Distributed Computing (DISC)*, number 3724 in LNCS, pages 3–17, Krakow, Poland. Springer Verlag.



Larkey, L. S., Connell, M. E., and Callan, J. (2000).

Collection selection and results merging with topically organized u.s. patents and trec data.

In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 282–289, New York, NY, USA. ACM Press.



Liu, X. and Croft, W. B. (2004).

Cluster-based retrieval using language models.

In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, NY, USA. ACM Press.



Lucchese, C., Orlando, S., Perego, R., and Silvestri, F. (2007).

Mining query logs to optimize index partitioning in parallel web search engines.

To Appear in Proceedings of The 2nd International Conference on Scalable Information Systems (INFOSCALE 2007).



Moffat, A., Webber, W., and Zobel, J. (2006).

Load balancing for term-distributed parallel retrieval.

In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 348–355, New York, NY, USA. ACM Press.



Puppini, D., Silvestri, F., and Laforenza, D. (2006).

Query-driven document partitioning and collection selection.

In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 34, New York, NY, USA. ACM Press.

Challenges in
Distributed Web
Retrieval

Ricardo
Baeza-Yates

Motivation

Crawling

Indexing

Query
Processing

Caching

Final Remarks



Webber, W., Moffat, A., Zobel, J., and Baeza-Yates, R.
(2006).

A pipelined architecture for distributed text query evaluation.
Information Retrieval.

published online October 5, 2006.

Challenges in
Distributed Web
Retrieval

Ricardo
Baeza-Yates

Motivation

Crawling

Indexing

Query
Processing

Caching

Final Remarks



Challenges in Distributed Web Retrieval

Ricardo Baeza-Yates^{1,2}

ricardo@baeza.c1

Joint work with: C. Castillo¹, A. Gionis¹, F. Junqueira¹,
V. Murdock¹, V. Plachouras¹ and F. Silvestri³

1. Yahoo! Research Barcelona – Catalunya, Spain
2. Yahoo! Research Latin America – Santiago, Chile
3. ISTI-CNR – Pisa, Italy