Distributed Web Crawling, Indexing, and Search

Ricardo Baeza-Yates
B. Barla Cambazoglu

Yahoo! Research
Barcelona Lab

Outline

- Overview of Web search
- Web crawling
- Query processing
- Indexing
- Federated and P2P search
- References

Web Search

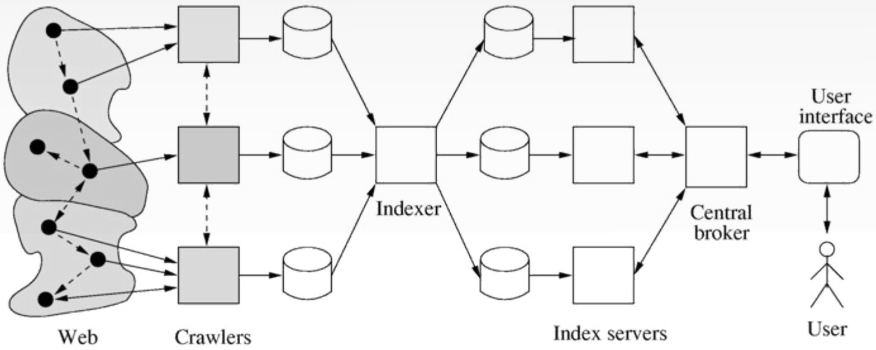
- Dimensions in Web search
  - Web
  - users
  - queries
  - search engines
- Web
  - today, there are more than 130 million Web servers
  - Web is the largest data repository (estimated as 100 billion pages)
  - well-connected graph with out- and in-link power law distributions
- Users
  - culturally and educationally diverse
  - little patience (few queries posed & few answers seen)

Web Search

- Queries
  - very short (inherent to users or due to the query language?)
  - different goals
    - informational
    - navigational
    - transactional
- Search engines
  - indices typically contain terabytes of data
  - hundreds of millions of queries served every day (thousands of queries per second)
  - a query must be evaluated under 300 ms
  - massive hardware infrastructures

Components of a Search Engine

- Three main components in a search engine
  - crawling
  - indexing
  - query processing



Web Crawling

- Web crawling is the process of locating, fetching, and storing the pages on the Web
- The computer programs that perform this task are referred to as Web crawlers or spiders
- A typical Web crawler
  - starts from a set of seed pages,
  - locates new pages by parsing the downloaded seed pages,
  - extracts the hyperlinks within,
  - stores the extracted links in a fetch queue for retrieval,
  - continues downloading until the fetch queue gets empty or a satisfactory number of pages are downloaded.

Web Crawling Architectures

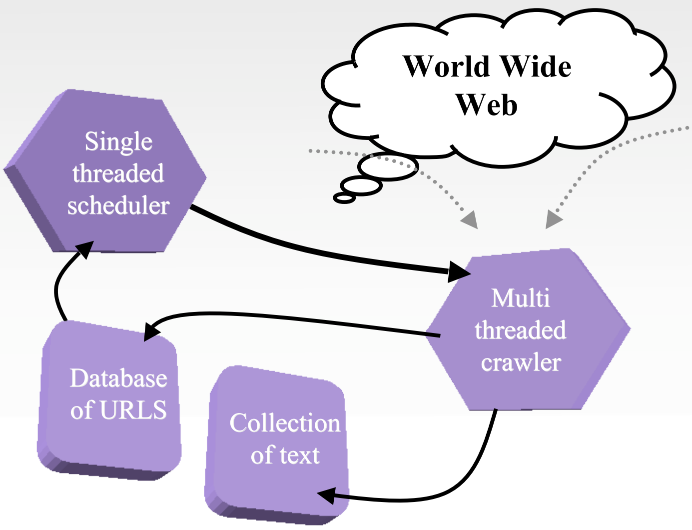
YAHOO!

- Sequential
  - single computer
  - not scalable
- Parallel
  - multiple computers, single data center
  - not scalable in terms of network
- Geographically distributed
  - multiple computers, multiple data centers
  - scalable, but has overheads

- 7 -

Sequential Web Crawling

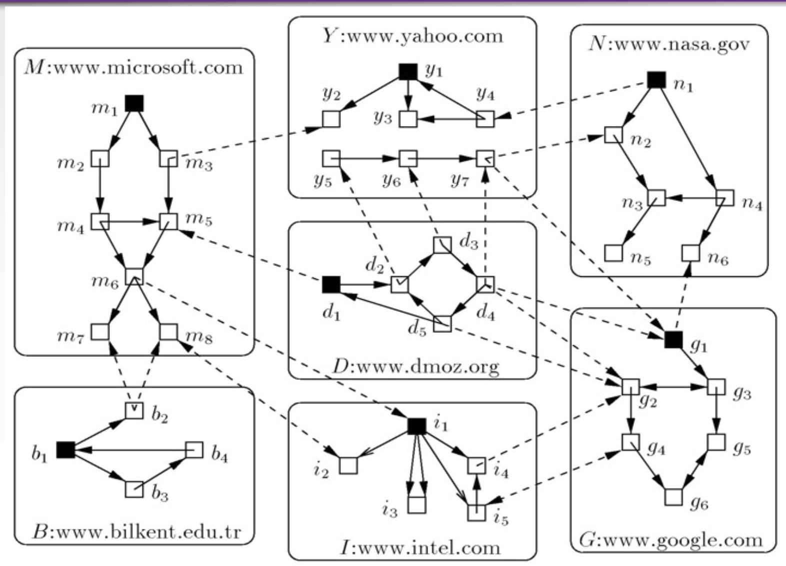
YAHOO!



- 8 -

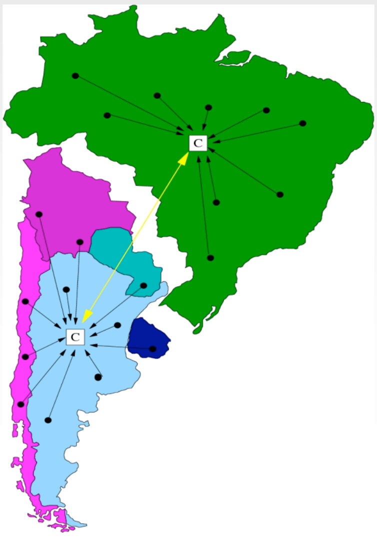
Parallel Web Crawling

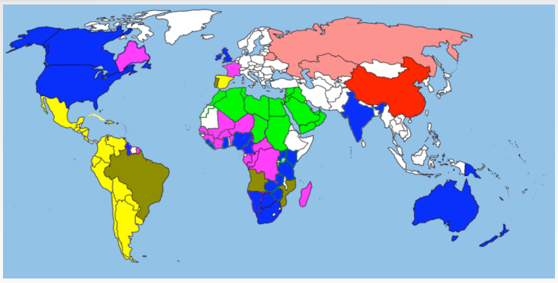
YAHOO!

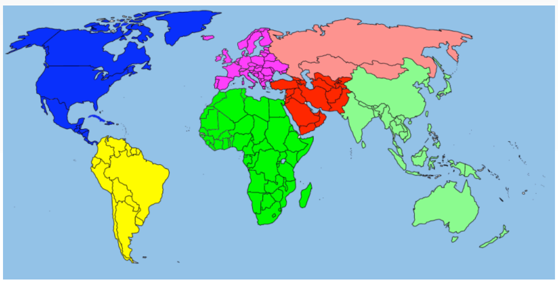


Geographically Distributed Web Crawling

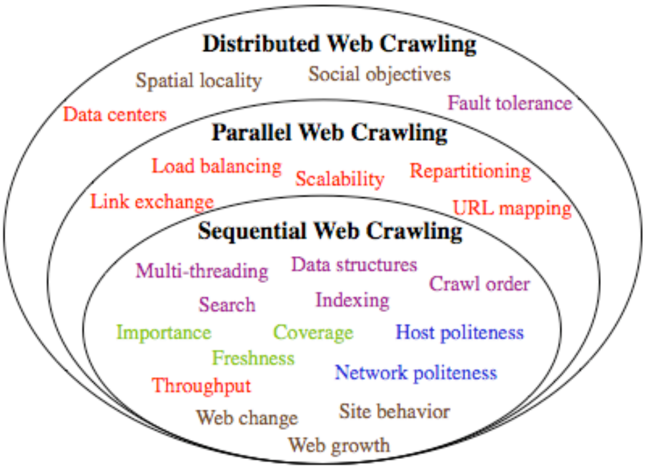
YAHOO!





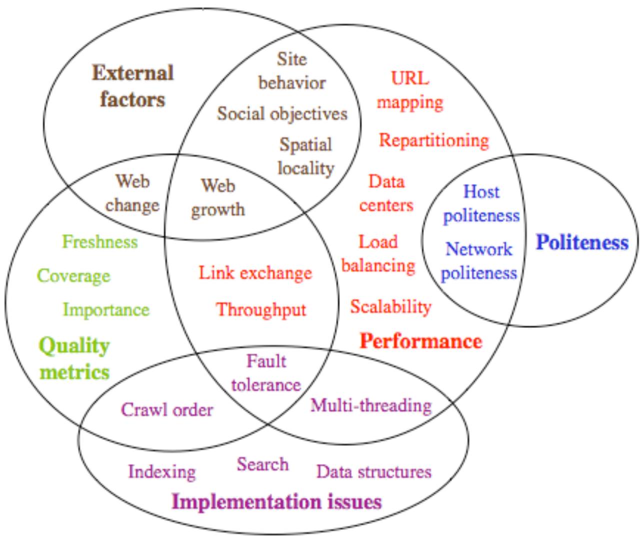


An Architectural Classification of Concepts



A Topical Classification of Concepts

- Quality metrics
- External factors
- Performance
- Implementation issues
- Politeness



Quality Metrics

- Coverage: The percentage of the Web discovered or downloaded by the crawler.
- Freshness: Measure of out-datedness of the local copy of a page relative to the page's original copy on the Web.
- Page importance: Percentage of important or popular pages in the repository.

External Factors

- Web growth
- Web change
- Site behavior
  - hostile sites (e.g., spider traps, infinite domain name generators)
  - spam sites (e.g., link farms)
  - sites with restricted content (e.g.,robot exclusion),
  - unstable sites (e.g., variable host performance, unreliable networks)
- Social objectives
  - language
  - country
  - region
  - interest
- Spatial locality

Performance

- Throughput: Download rate in bytes per second
- Load balancing
  - content stored
  - bytes downloaded
  - requests issued
- Scalability in terms of the number of
  - pages
  - crawlers
  - data centers
- Link exchange
- URL mapping
- Repartitioning
- Data centers

Implementation Issues / Politeness

- Multi-threading
- Crawl order
- Data structures
  - queue of the URLs to be downloaded
  - list of the URLs seen
  - local DNS cache
  - cache of robots.txt files
- Fault tolerance
- Indexing
- Search
- Host politeness
- Network politeness

Benefits of Distributed Web Crawling

- Higher crawling throughput
  - spatial locality
  - low latency
- Improved network politeness
  - less overhead on routers
- Resilience to network partitions
  - better coverage
- Increased availability
  - continuity of business
- Better coupling with distributed indexing/search
  - reduced data migration

Challenges in Distributed Web Crawling

- Web partitioning/repartitioning: the problem of finding a Web partition that minimizes the overheads in distributed Web crawling
  - minimization objectives
    - page download times
    - link exchange overhead
    - repartitioning overhead
  - constraints
    - coupling with distributed search
    - load balancing
- Data center placement: the problem of finding the optimum geographical placement for a fixed number of data centers
  - geographical locations are now objectives, not constraints
  - optimum number of data centers

Challenges in Distributed Web Crawling

- Link classification
  - may need to identify
    - language
    - region
    - interest to a community
  - may utilize
    - site content
    - link connectivity of the site
    - IP databases
  - multi-language sites
    - overlap

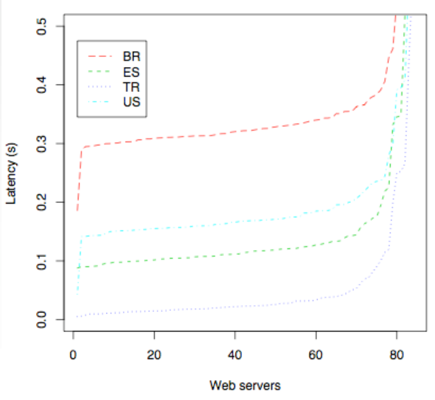
Challenges in Distributed Web Crawling

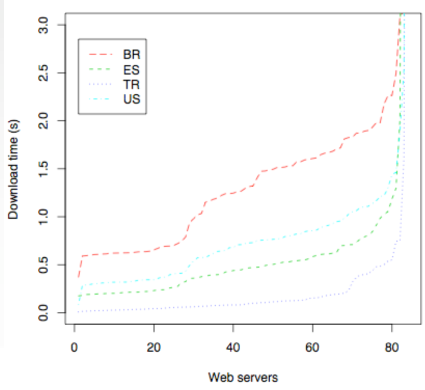
- Coupling with indexing/search
  - data may be moved to
    - a single data center
    - replicated on multiple data centers
    - partitioned among a number of data centers
  - decisions must be given on
    - what data to move (e.g., pages or index)
    - how to move (i.e., compression)
    - how often to move (i.e., synchronization)

Experiments on Throughput Performance

- Network access statistics over the .edu domains
  - using a customized echoping version
  - over one week
- Eight crawled countries
  - US, Canada
  - Brazil, Chile
  - Spain, Portugal
  - Turkey, Greece
- Four crawling countries
  - US
  - Brazil
  - Spain
  - Turkey

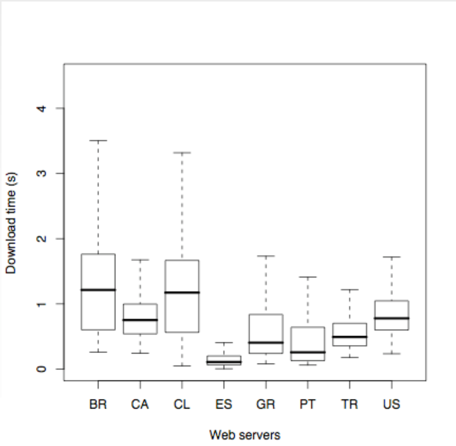
Spatial Locality

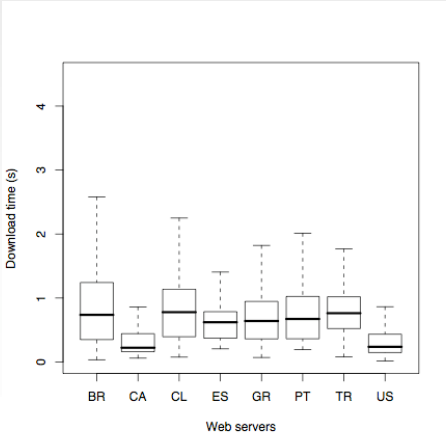




Crawler Performance

YAHOO!

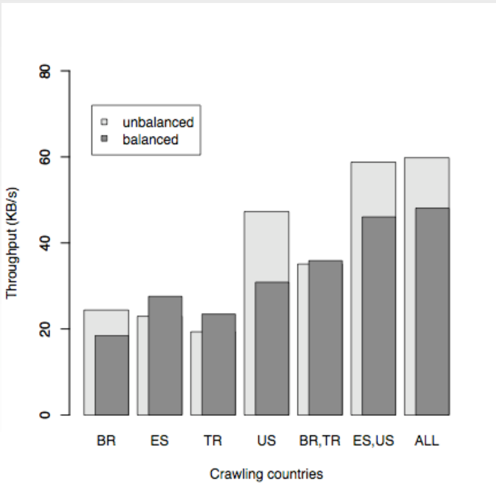




- 23 -

Throughput

YAHOO!



- 24 -

Observations

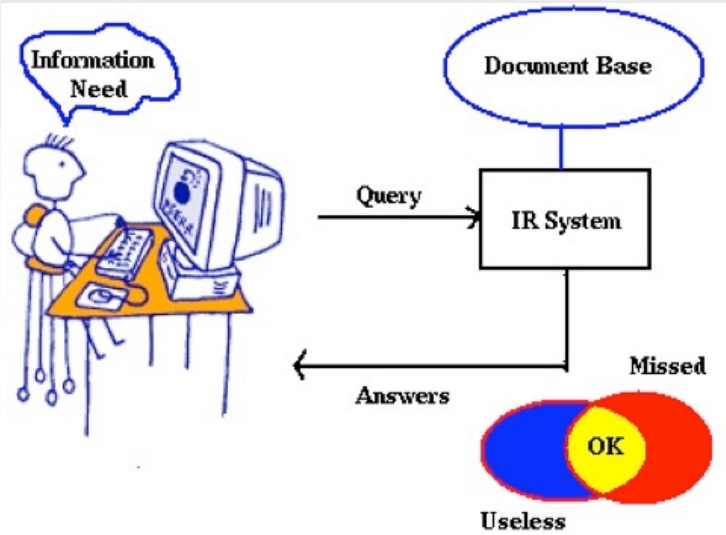
YAHOO!

- There should be at least one data center in US as US has
  - more Web pages
  - better network infrastructure
- For the current distribution of pages, centralized Web crawling seems to be still feasible
- A complete feasibility analysis requires work in both theory & practice
  - on the theoretical side, appropriate cost models should be developed.
    - financial costs (operational, maintenance, revenue)
    - scalability (number of data centers, number of crawlers per center, the network bandwidths)
    - performance (download, link exchange, repartitioning times)
  - on the practical side, the trends in the Web should be followed

- 25 -

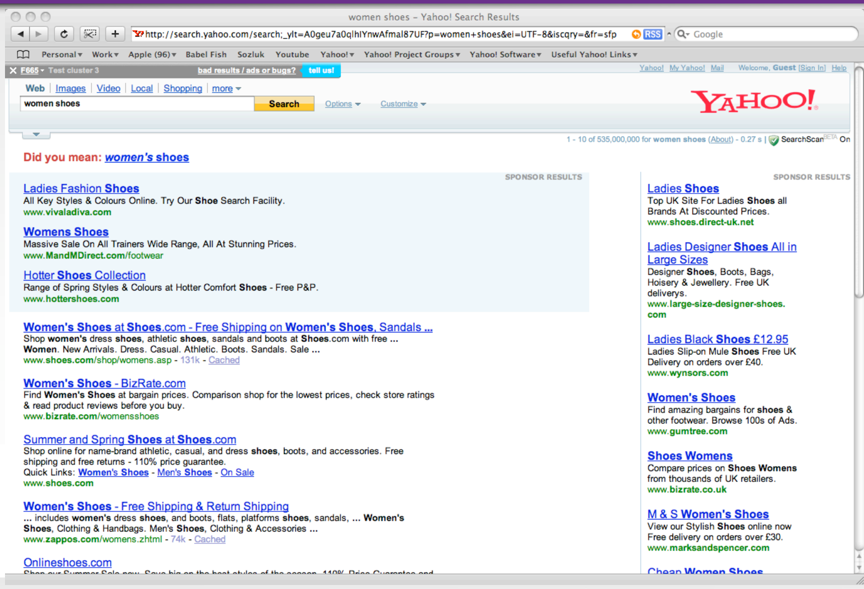
Query Processing

YAHOO!



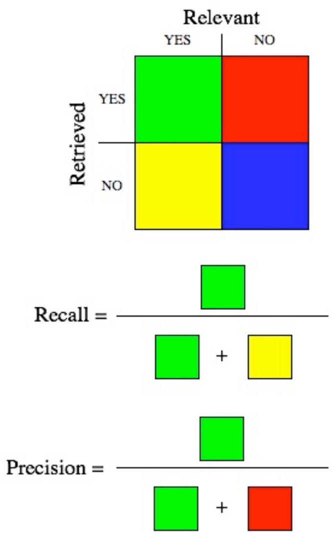
- 26 -

Query Processing



Ranking

- Two important measures
  - recall
  - precision
- Most important features
  - Content (e.g., tf-idf)
  - URL (e.g., site importance)
  - Link (e.g., PageRank)
  - Spam (e.g., porn)
  - Click



Background on Parallel Architectures

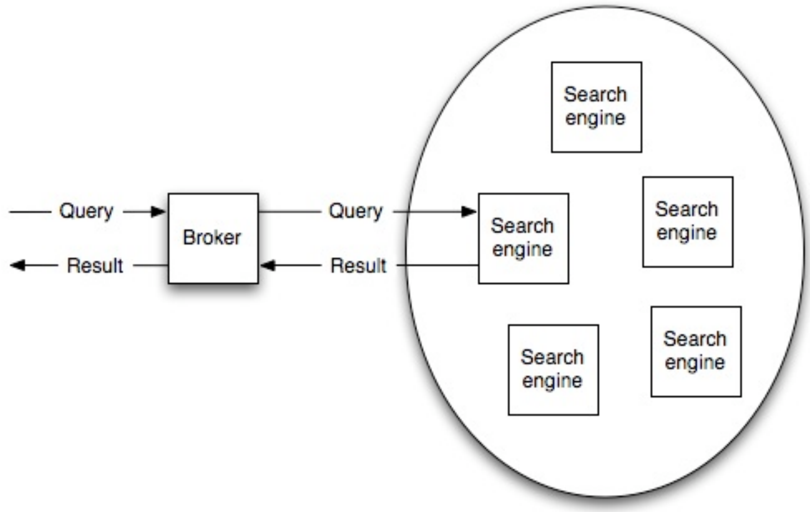
		Data stream	
		Single	Multiple
Instruction stream	Single	SISD classical	SIMD simple
	Multiple	MISD (rare)	**MIMD** many SISD

MIMD Architectures

- Can be
  - tightly coupled (shared memory)
  - loosely coupled (distributed memory)
- Distributed-memory architectures
  - many computers interacting via network
  - PC Clusters
  - very loosely coupled
  - more coarse-grained programs
- Two ways a retrieval system can exploit a MIMD machine
  - parallel multitasking (inter-query parallelism)
  - partitioned parallel processing (intra-query parallelism)

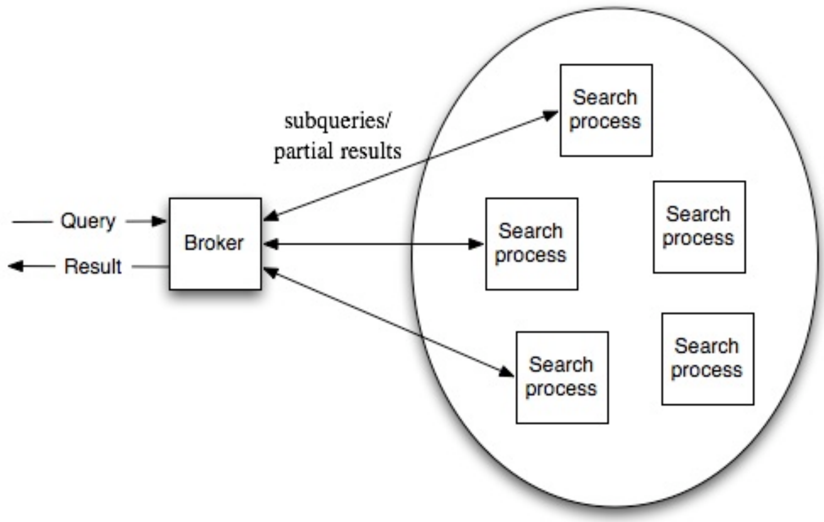
Inter-query Parallelism

YAHOO!



Intra-query Parallelism

YAHOO!



Indexing

- An inverted index is a representation for the document collection over which user queries will be evaluated

- Alternatives
  - signature files
  - suffix arrays

- An inverted index is composed of two parts
  - a set of inverted lists
    - a set posting entries
      - document id
      - word score
      - word positions
  - an index into these lists

Indexing

A sample document collection

Document	Text
1	Pease porridge hot
2	Pease porridge cold
3	Pease porridge in the pot
4	Pease porridge hot, pease porridge not cold
5	Pease porridge cold, pease porridge not hot
6	Pease porridge hot in the pot

Indexing

YAHOO!

Dictionary	Inverted Lists					
cold	<2,1>	<4,1>	<5,1>			
hot	<1,1>	<4,1>	<5,1>	<6,1>		
in	<3,1>	<6,1>				
not	<4,1>	<5,1>				
pease	<1,1>	<2,1>	<3,1>	<4,2>	<5,2>	<6,1>
porridge	<1,1>	<2,1>	<3,1>	<4,2>	<5,2>	<6,1>
pot	<3,1>	<6,1>				
the	<3,1>	<6,1>				

Inverted Index Partitioning

YAHOO!

- There are two possible methods for partitioning an index
  - Term-based partitioning
    - T inverted lists are distributed across P processors
    - each processor is responsible for processing the postings of a mutually disjoint subset of inverted lists assigned to itself
    - single disk access per query term
    - multiple accumulators communicated per document
  - Document-based partitioning
    - N documents are distributed across P processors
    - each processor is responsible for processing the postings of a mutually disjoint subset of documents assigned to itself
    - multiple disk accesses per query term
    - single accumulator communicated per document

Term-Based Partitioning

YAHOO!

- Parallel index construction
  - documents are assigned to processors
  - each indexing process generates a batch of inverted lists
  - a merge step is performed to create the final, global index
- Query processing
  - query is decomposed into terms
  - each term is sent to a processor holding the corresponding inverted list
  - each processor that received a subquery creates an accumulator list with partial document scores and returns them to the broker
  - broker combines partial document scores and creates a final answer set

Term-Based Partitioning

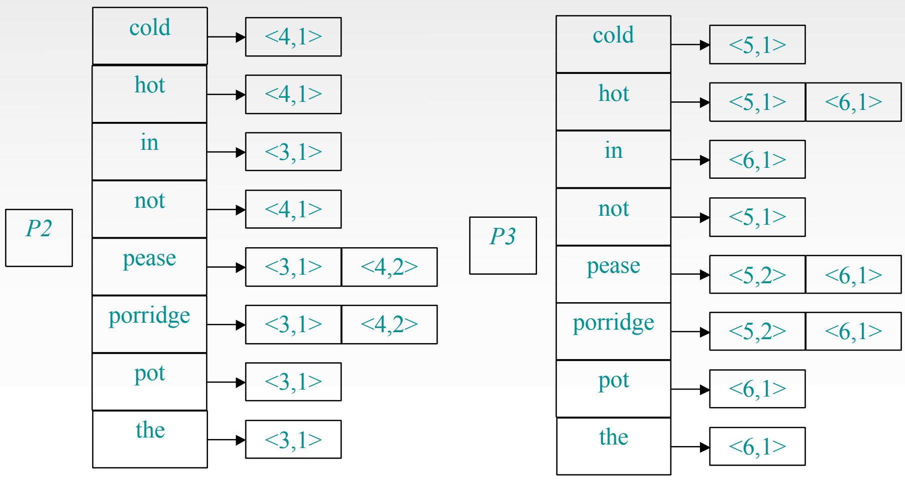
YAHOO!

P1	cold	<2,1>	<4,1>	<5,1>			
	hot	<1,1>	<4,1>	<5,1>	<6,1>		
	in	<3,1>	<6,1>				
P2	not	<4,1>	<5,1>				
	pease	<1,1>	<2,1>	<3,1>	<4,2>	<5,2>	<6,1>
	porridge	<1,1>	<2,1>	<3,1>	<4,2>	<5,2>	<6,1>
P3	pot	<3,1>	<6,1>				
	the	<3,1>	<6,1>				

Document-Based Partitioning

- Parallel index construction
  - each processor concurrently processes its document collection and creates a local index
  - processors exchange their local index statistics with others
  - a merge step is performed to form the global index statistics on all processors
- Query processing
  - broker sends the query to all processors
  - each processor evaluates the query on its local index producing a partial answer set
  - the broker combines the partial answer sets into a final answer set

Document-Based Partitioning



Comparison of Partitioning Schemes

Document-Based Partitioning	Term-Based Partitioning
■ Disk space consumption: More space consumption since the index file has to be replicated. Balanced disk space consumption.	■ Disk space consumption: Less space consumption since the index file is partitioned. Unbalanced disk space consumption.
■ Number of disk accesses: More disk accesses: the number of inverted list accesses for a given term is equal to the number of disks containing the inverted lists.	■ Number of disk accesses: Fewer disk accesses: for a given term, there is only a single disk access.
■ Load distribution: Though there are more disk accesses, they are concurrent.	■ Load distribution: The parallelism is limited by number of terms in the query.
■ I/O time: Shorter inverted lists. Hence, the resulting I/O time could be less.	■ I/O time: Longer inverted lists. The I/O time depends on the size of the longest posting entry.

In Practice…

- The state-of-the-art in search engines is document-based partitioning
- This is simpler to build and update
- Low inter-query-processing concurrency, but good load balance
- Low throughput, but high response time
- High throughput is achieved by replication
- Easier to maintain
- More fault tolerant

Federated Query Processing

YAHOO!

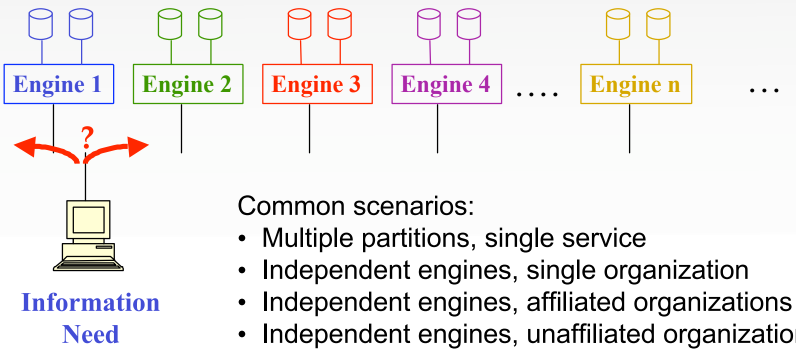
- Federated search is the process of searching multiple, online databases with potentially contextually independent content usually by means of a portal logically unifying these databases.

- This type of search is different than distributed text retrieval
  - databases are autonomous
  - semantically disjoint content
  - partial content overlap
  - infrastructure is heterogeneous

Federated Query Processing

YAHOO!

- This kind of search involves three steps
  - collection selection
  - query processing
  - result aggregation



Common scenarios:
- Multiple partitions, single service
- Independent engines, single organization
- Independent engines, affiliated organizations
- Independent engines, unaffiliated organizations

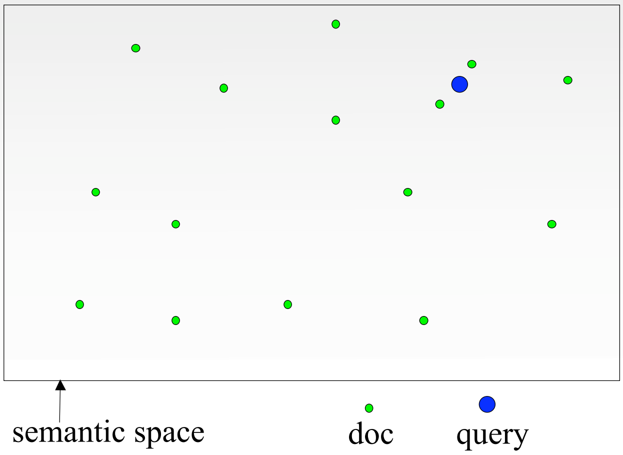
Issues Addressed

- Site description
  - content
  - search engine
  - services
- Resource ranking
  - ranking resources by how likely to contain desired content
- Resource selection
  - selecting the best subset from a ranked list
- Searching
  - interoperability
- Result merging
  - merging a set of document rankings
    - different underlying corpus statistics
    - different search engines

Peer-to-Peer Search

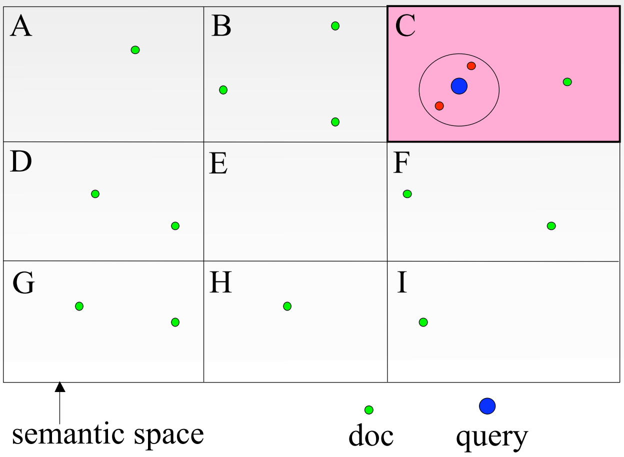
- P2P search is performed over a set of search agents by forwarding the search request between them and collecting results on the way. Agents are typically
  - many,
  - autonomous,
  - very dynamic.
- In practice, mostly for retrieval of
  - music,
  - video
- Traditional approaches
  - centralized
  - flooding
- More advanced approaches
  - CAN, Chord, Pastry, Tapestry, pSearch
  - scalable, fault-tolerant, self-organizing

Peer-to-Peer Search



- 47 -

Peer-to-Peer Search



- 48 -

Open Problems

- New retrieval models
- New ranking techniques
- More on indexing & searching
- Quality evaluation (Web, XML)
- Geographically distributed IR architectures
- More on P2P
- Spam detection
- Multimedia retrieval
- Grid computing

References

- Modern Information Retrieval, R. Baeza-Yates & B. Ribeiro-Neto, Addison-Wesley, 1999.
- Managing Gigabytes: Compressing and Indexing Documents and Images, I.H. Witten, A. Moffat, and T.C. Bell .Morgan Kaufmann, San Francisco, second edition, 1999.
- Modeling the Internet and the Web: Probabilistic Methods and Algorithms, Pierre Baldi, Paolo Frasconi, and Padhraic Smyth, John Wiley & Sons; May 28, 2003.
- Mining the Web: Analysis of Hypertext and Semi Structured Data, Soumen Chakrabarti, Morgan Kaufmann, 2002.
- Websites:
  - http://www.searchenginewatch.com/
  - http://www.searchengineshowdown.com/