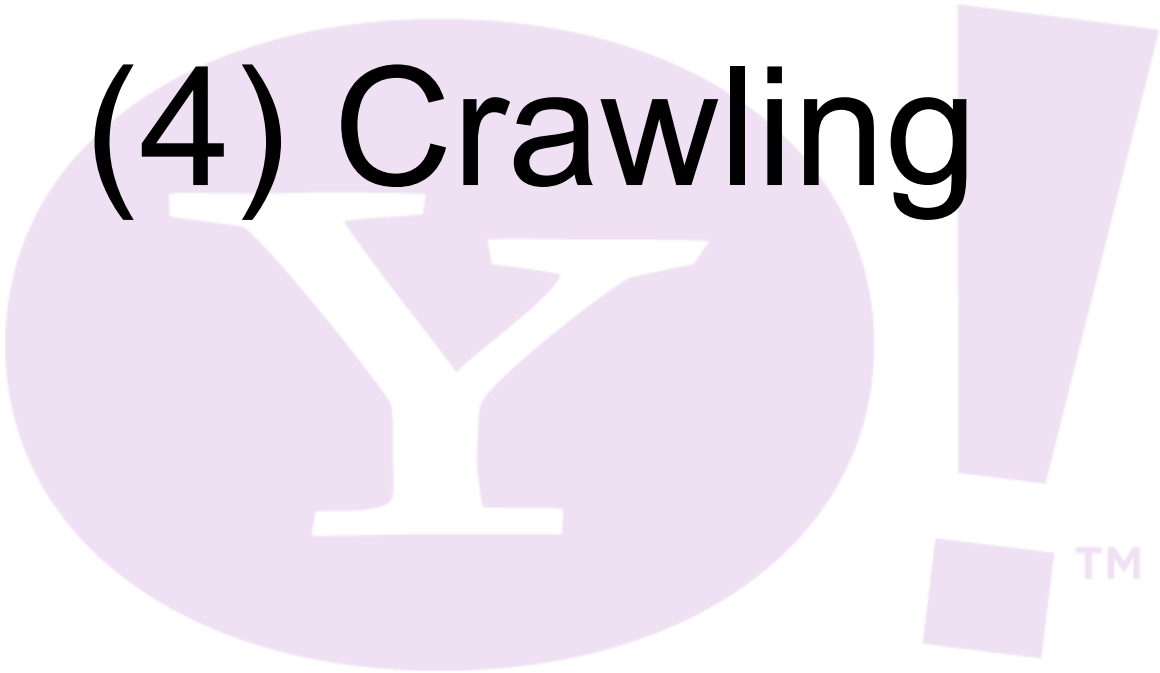
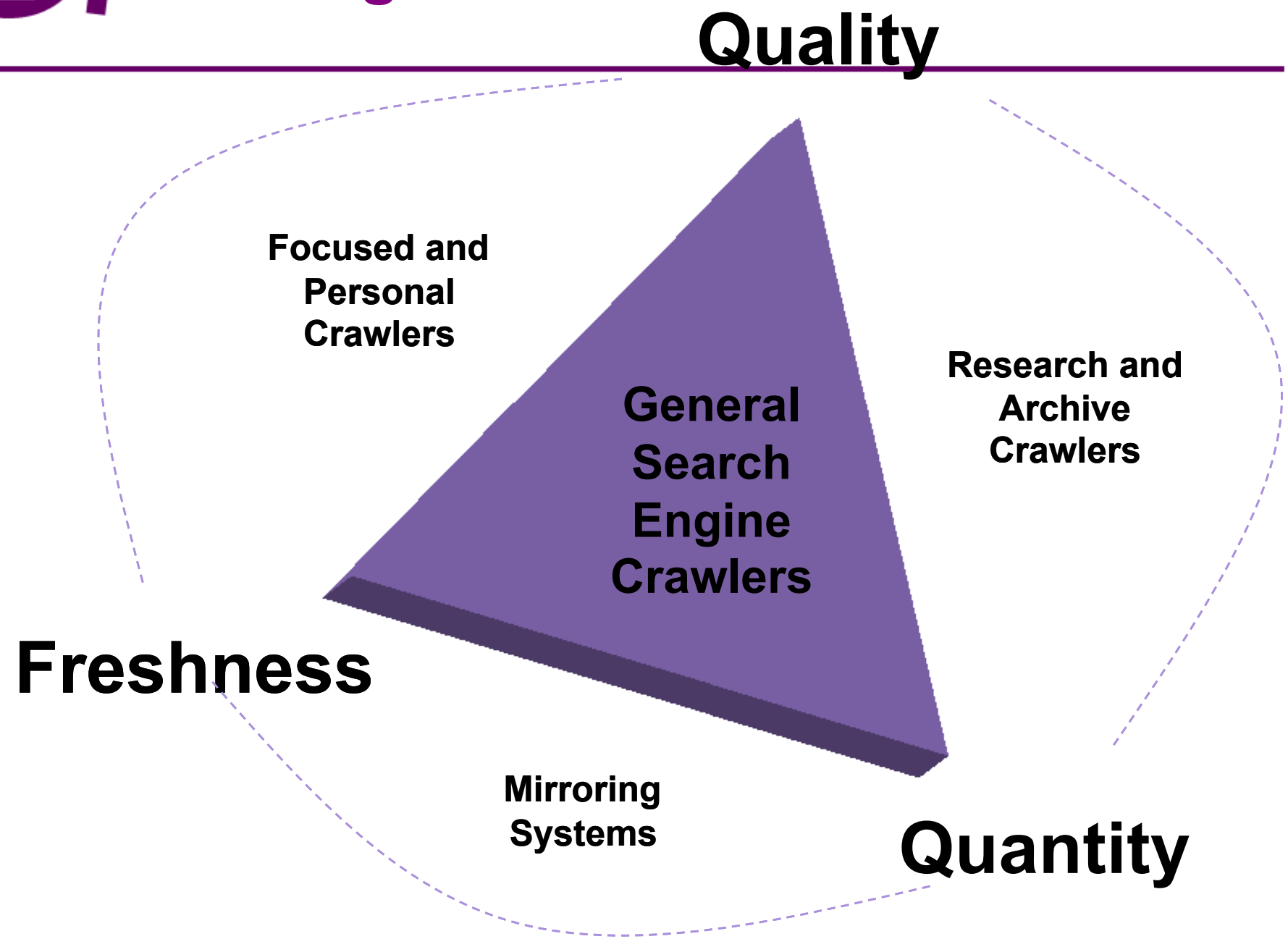
(4) Crawling

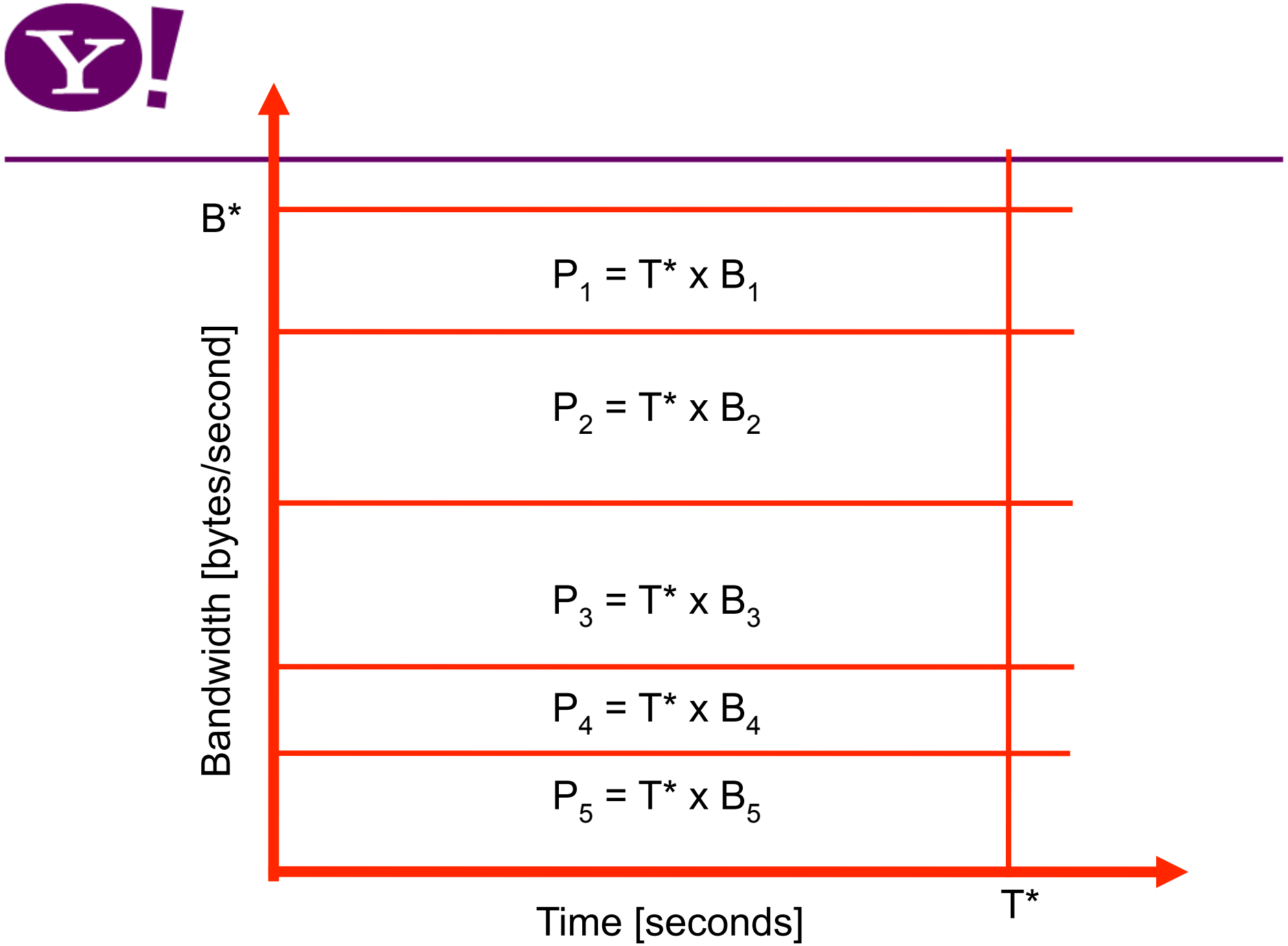


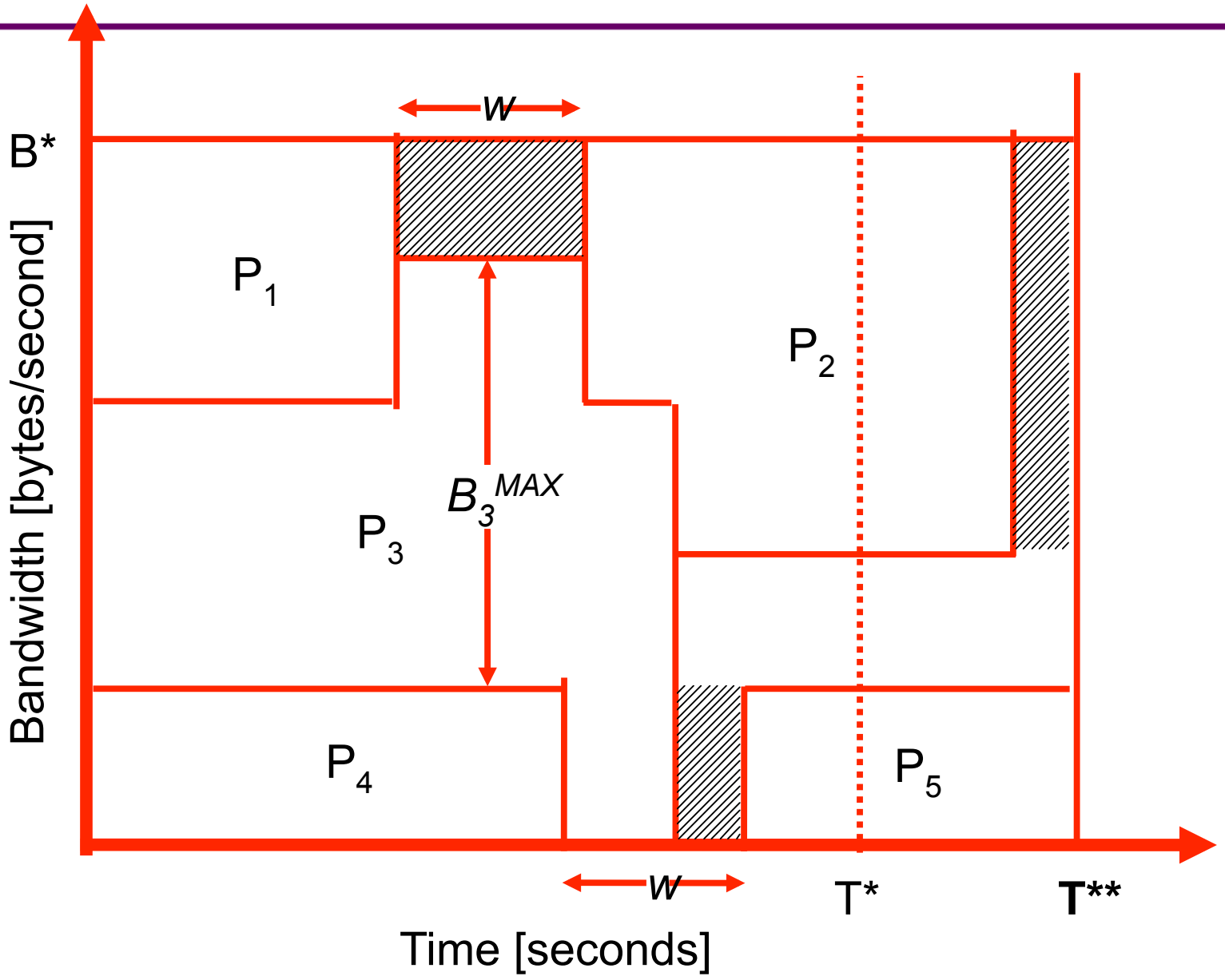
Crawling

- NP-Hard Scheduling Problem
- Different goals
- Many Restrictions
- Difficult to define optimality
- No standard benchmark

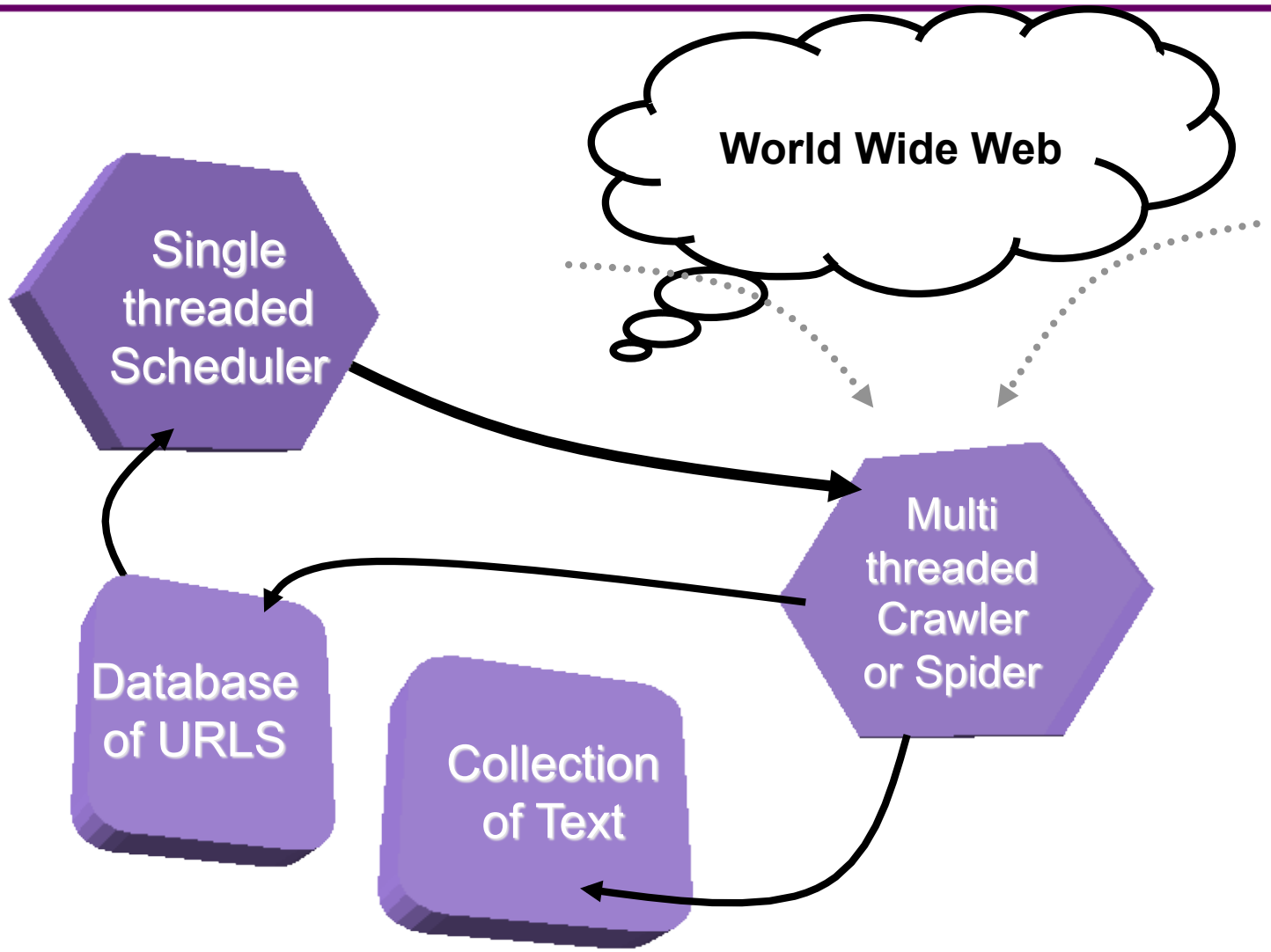
Crawling Goals

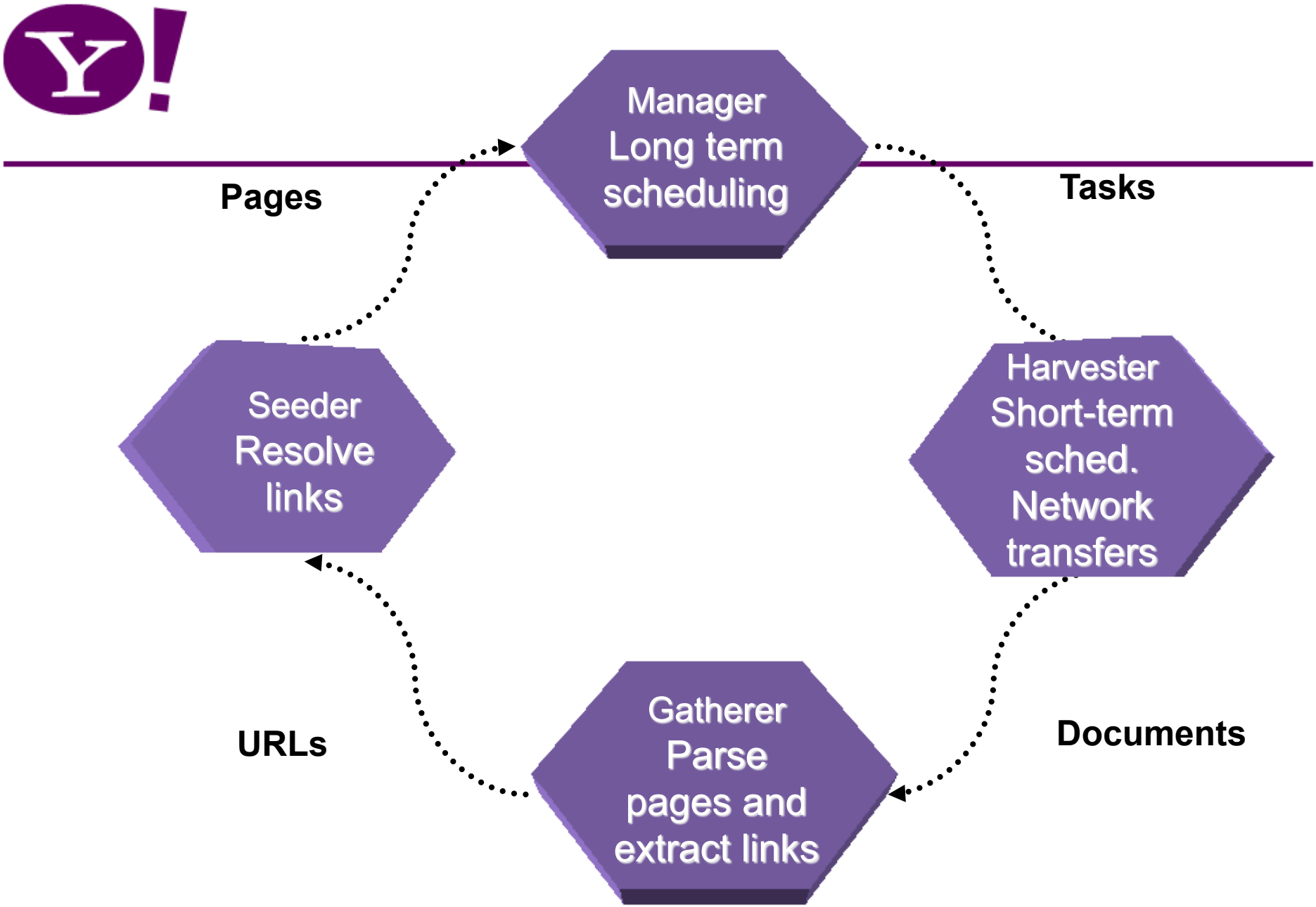



B*
Bandwidth [bytes/second]
P1 = T* x B1
P2 = T* x B2
P3 = T* x B3
P4 = T* x B4
P5 = T* x B5
T*
Time [seconds]

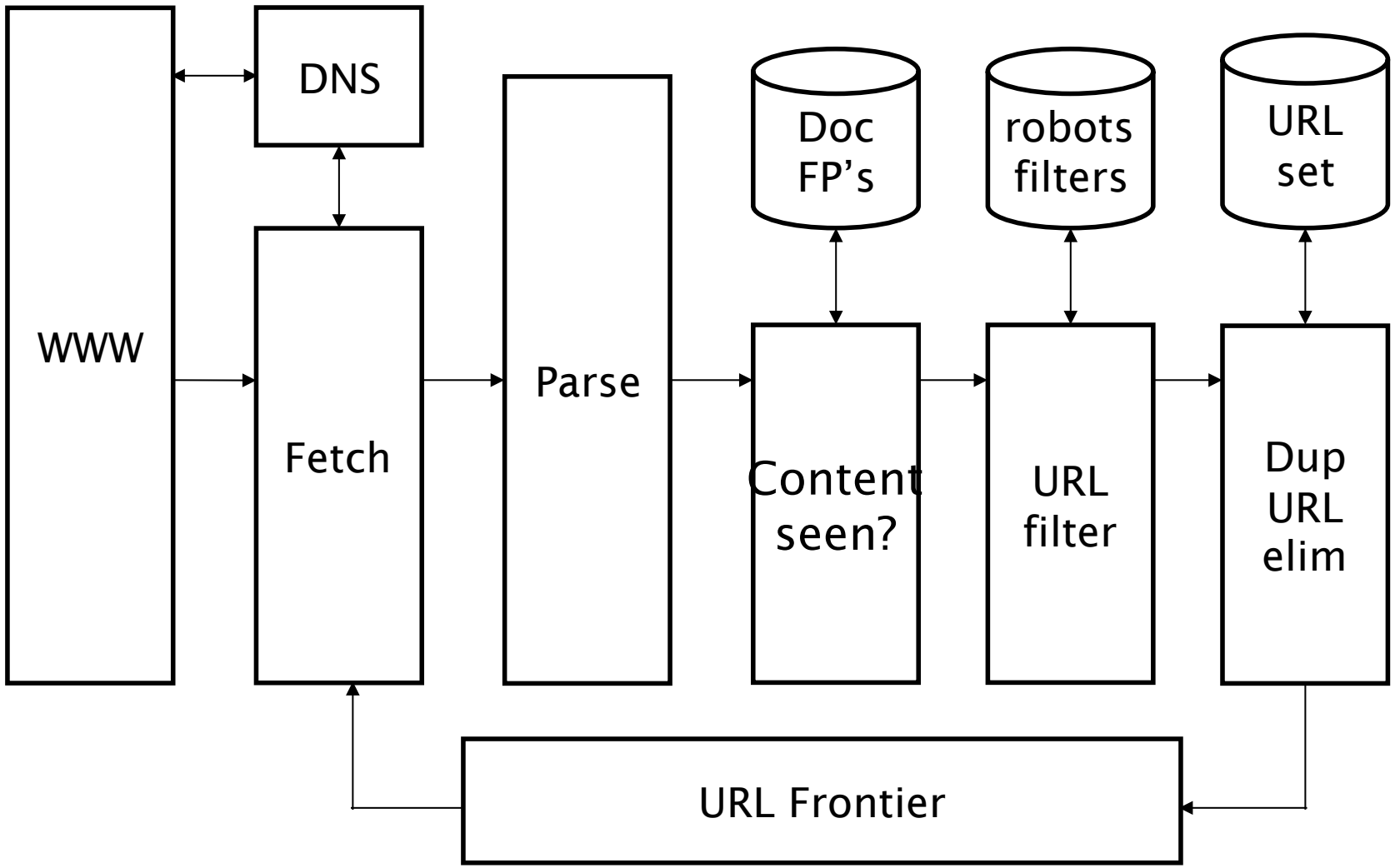

w
B*
Bandwidth [bytes/second]
P1
P2
P3
B3 MAX
P4
P5
w
T*
T**
Time [seconds]

Software Architecture

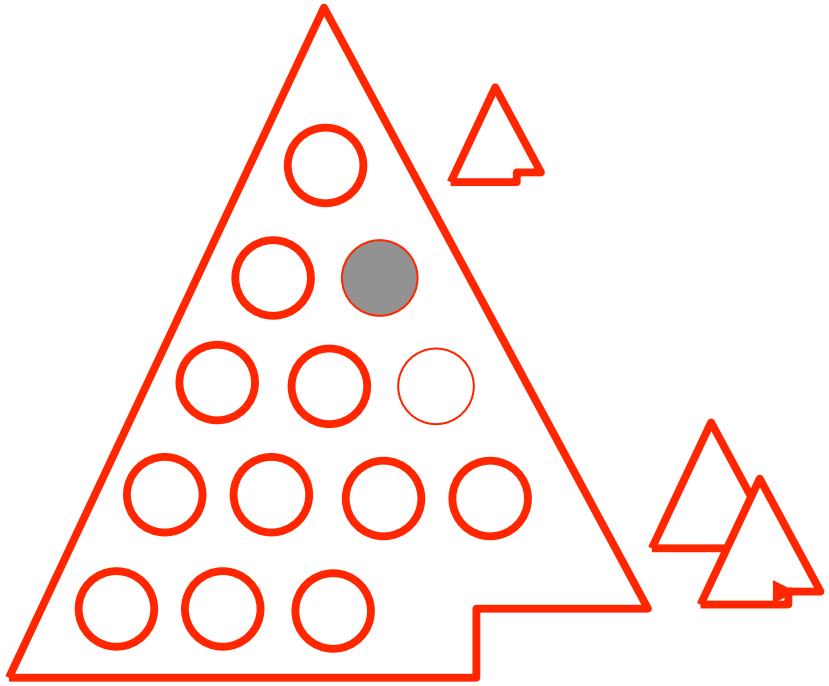



Manager
Long term
scheduling
Pages
Tasks
Seeder
Resolve
links
Harvester
Short-term
sched.
Network
transfers
URLs
Gatherer
Parse
pages and
extract links
Documents

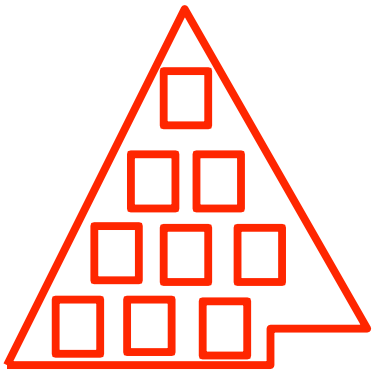
Basic Crawl Architecture



Priority Queues



Queue of Web sites
(long-term scheduling)



Queue of Web pages
for each site
(short-term scheduling)

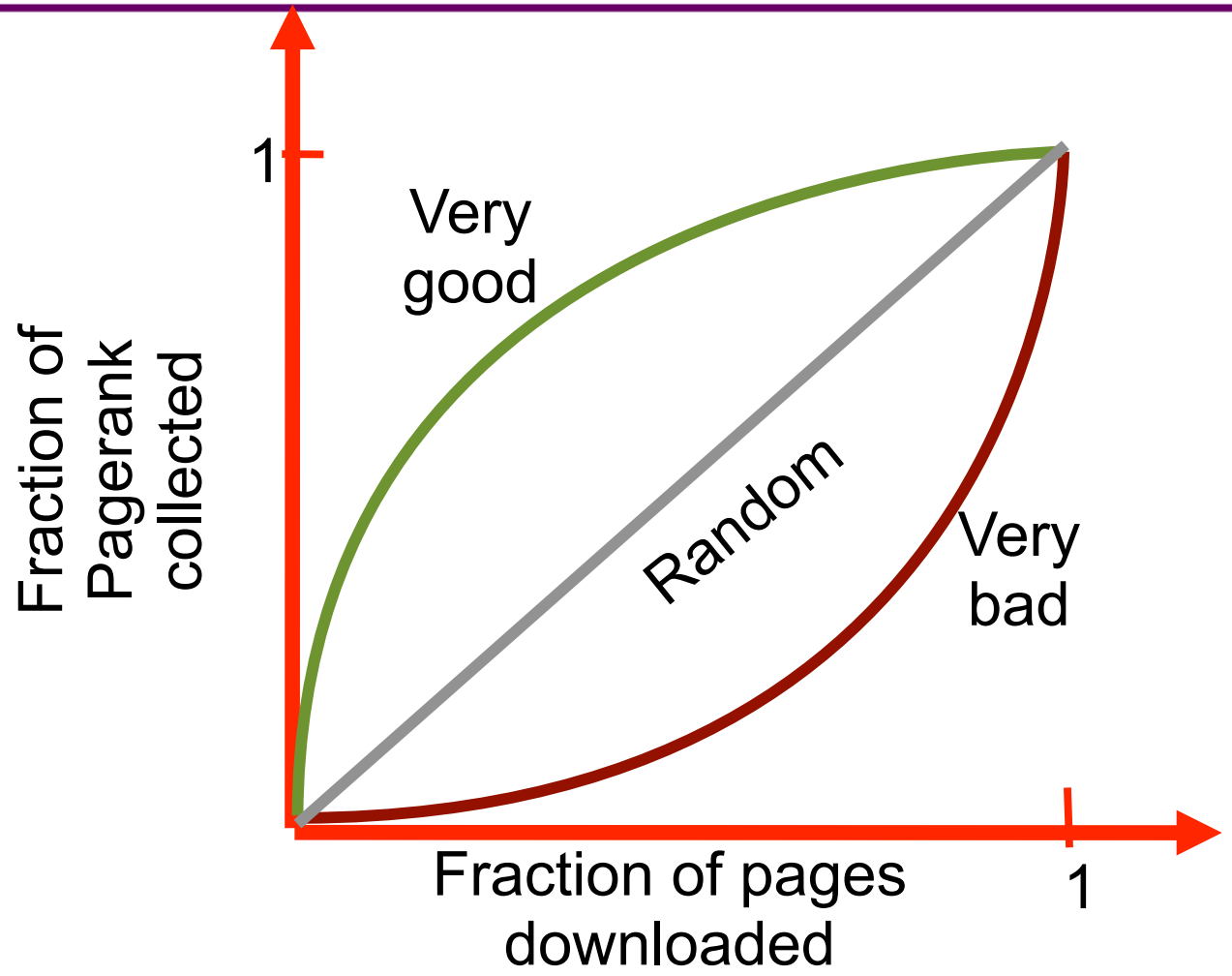
Formal Problem

- **Find a sequence of page requests *(p,t)* that:**
  - Optimizes a function of the volume, quality and freshness of the pages
  - Has a bounded crawling time
  - Fulfils politeness
  - Maximizes the use of local bandwidth
- **Must be on-line: how much knowledge?**

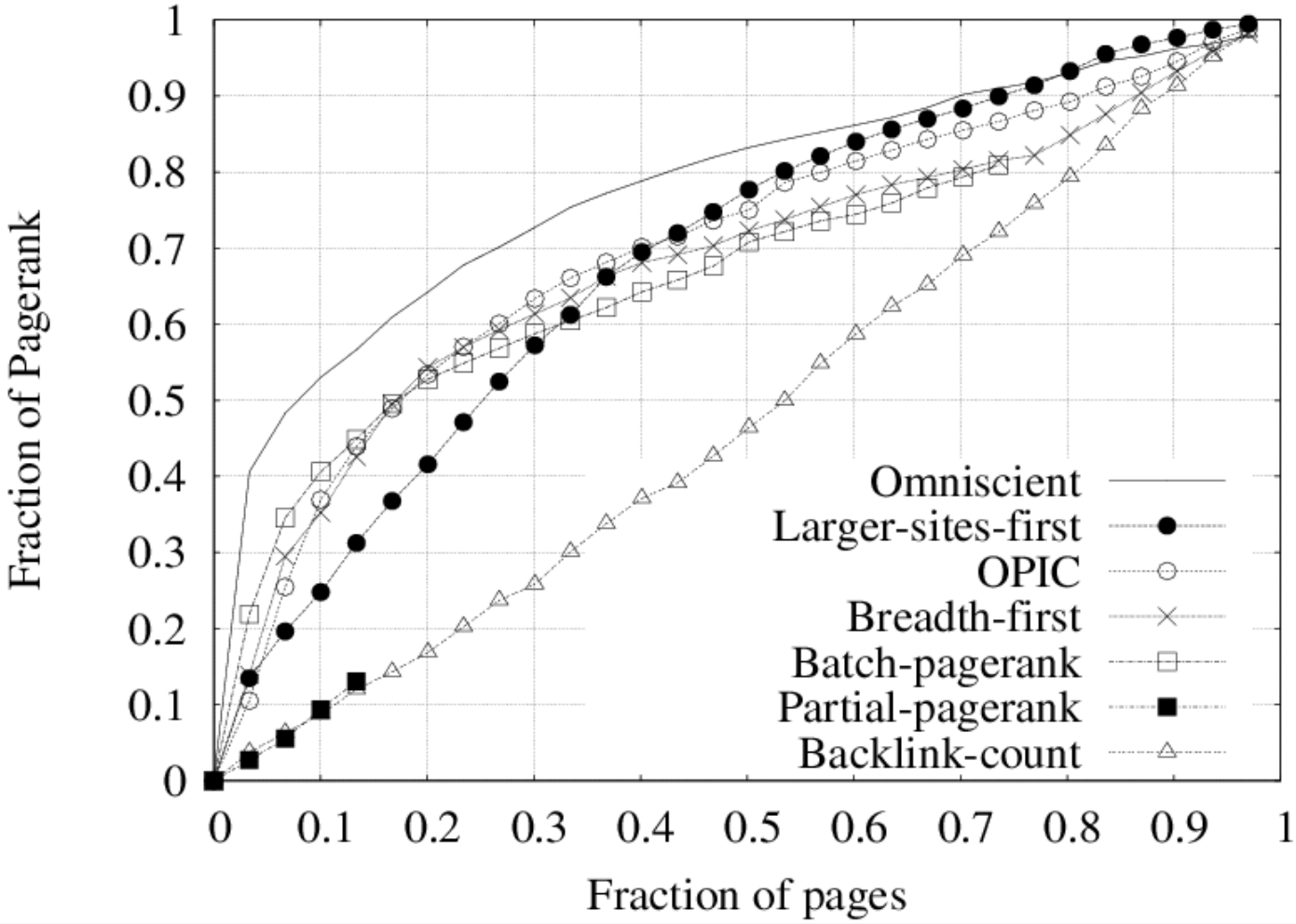
Crawling Heuristics

- **Breadth-first**
- **Ranking-ordering**
  - PageRank
- **Largest Site-first**
- **Use of:**
  - Partial information
  - Historical information
- **No Benchmark for Evaluation**
  - Use simulation

Comparing crawling algorithms

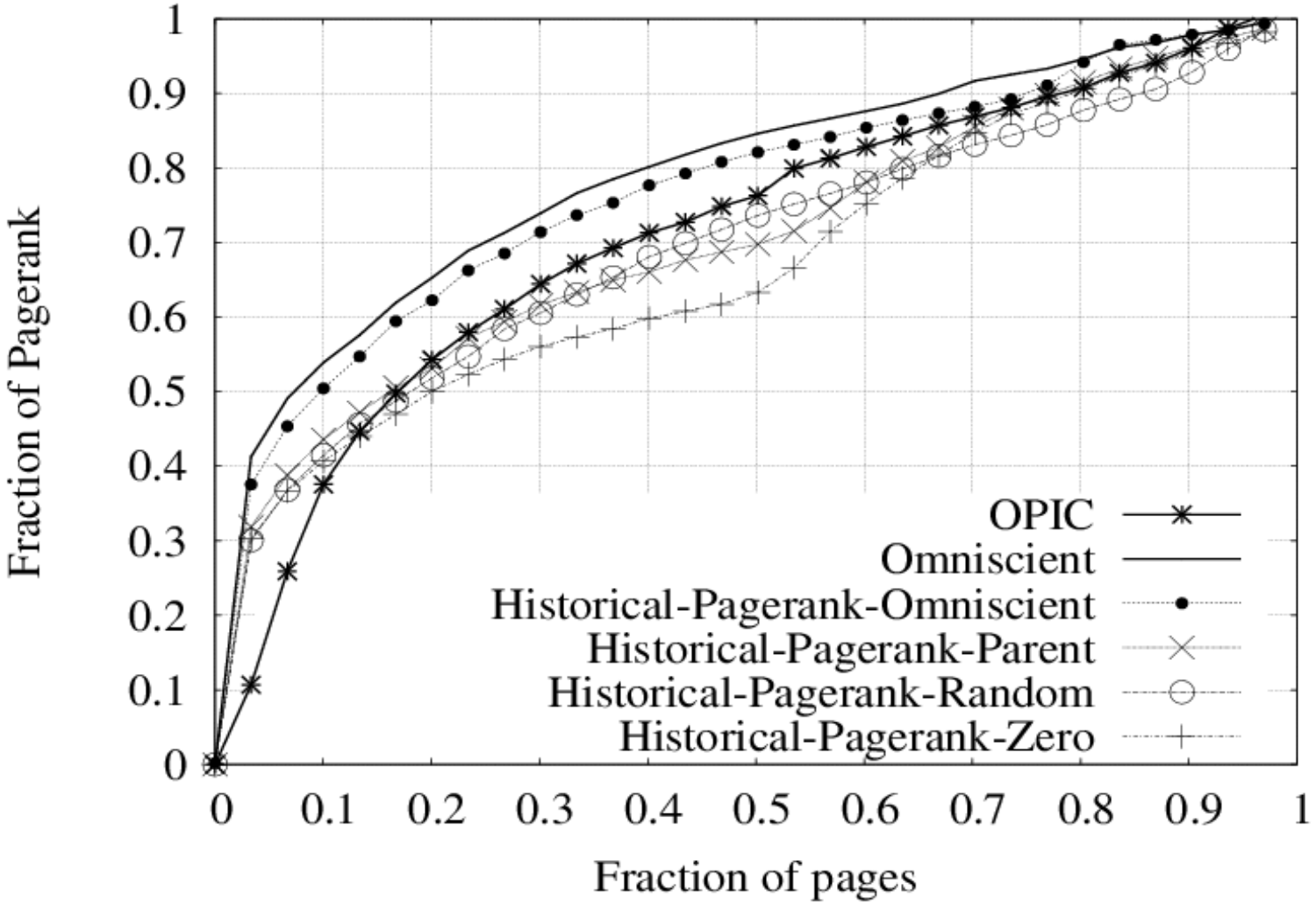


No Historical Information

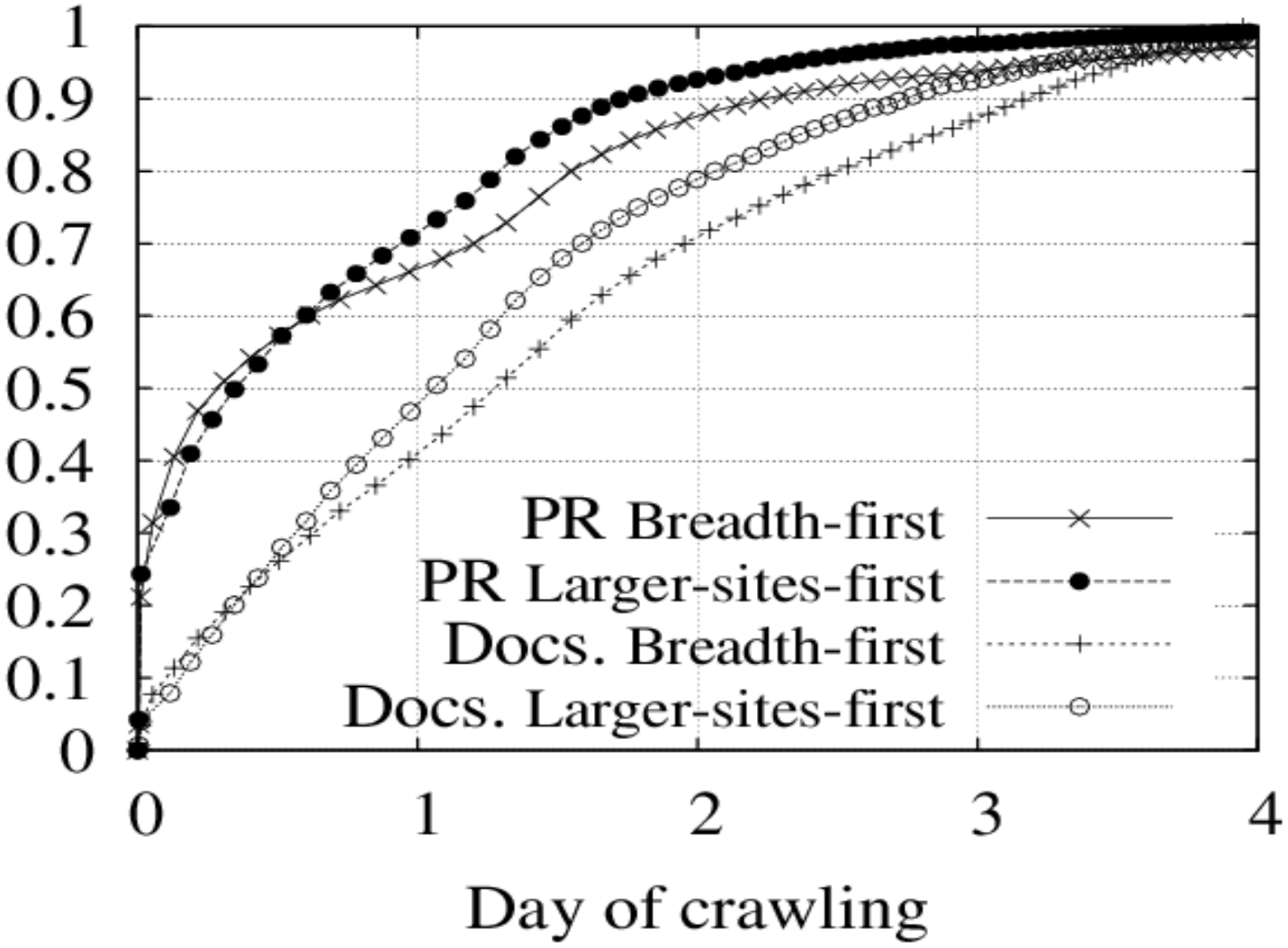


Baeza-Yates, Castillo, Marin & Rodriguez, WWW2005

Historical Information



Validation in the Greek domain

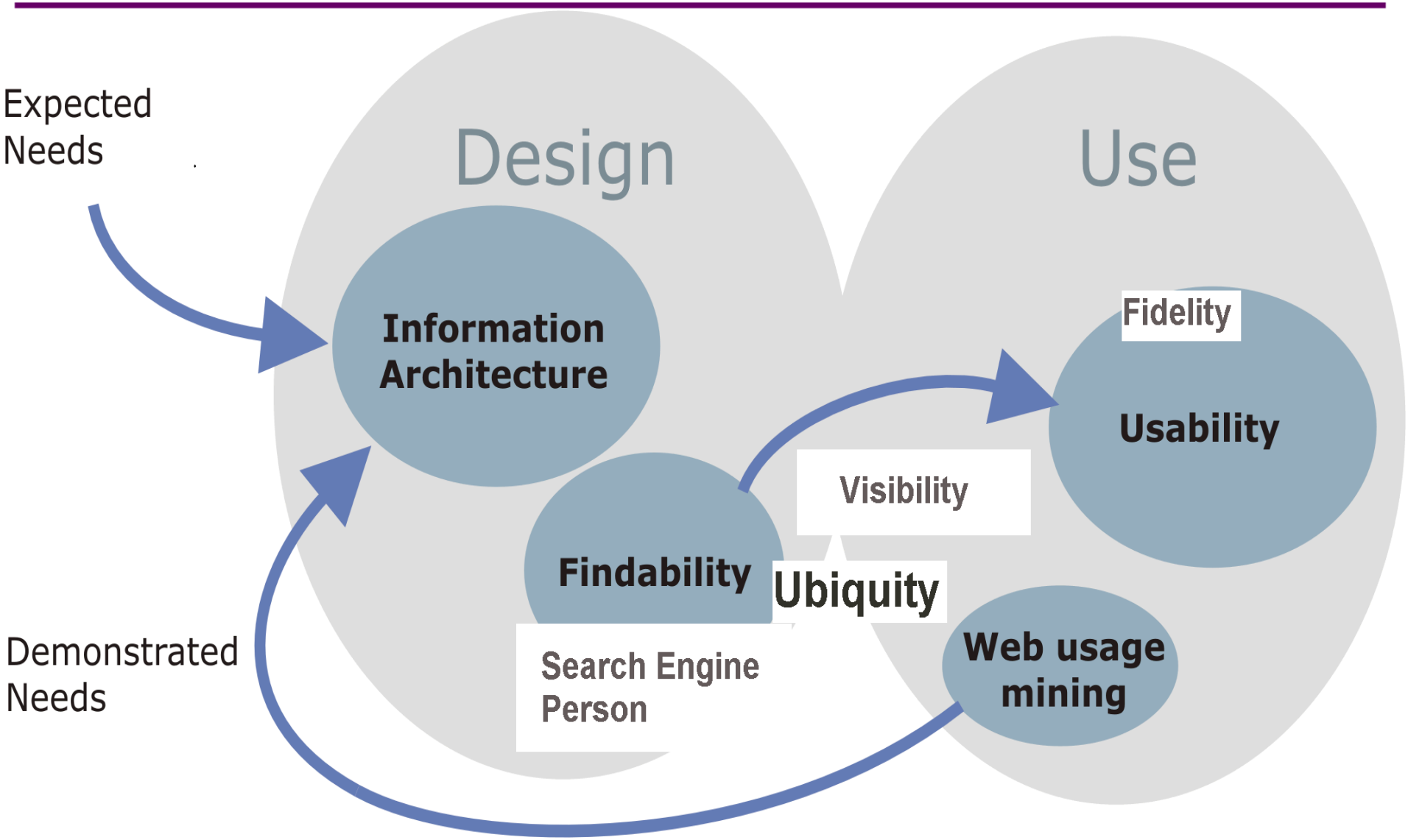


(5) Final Remarks

Young research field

- The Web is scientifically young
- The Web is intellectually diverse
- The technology mirrors the economic, legal and sociological reality
- Search is evolving to “task completion” and implicit search
- Plenty of open problems

Web Design and Search



Main Open Problems

Large scale usage data is key BUT

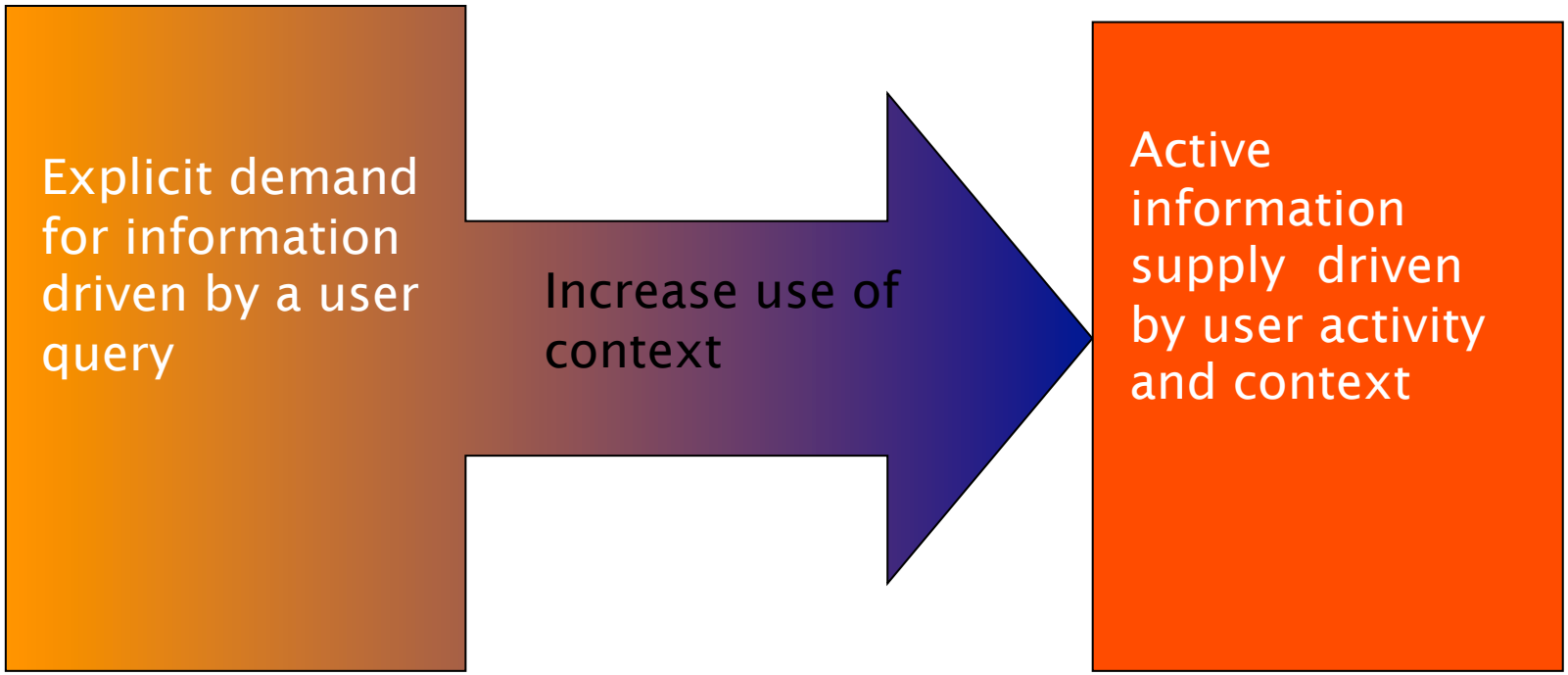
Usage data at a very large scale	
over larger and larger populations over longer and longer periods of time	
Personalization	**Privacy**
More data via larger communities, makes data less personalized *wisdom of crowds does not work well on small corpora*	Over personalization endangers privacy Long-term logs endanger privacy

The New frontiers

- **Front-end and user experience**
  - The most probable reason for users to switch between quasi-equivalent engines is a better user experience

- **Depart from the rectangle/ranked list paradigm**
  - Get rid of queries? Implicit search
    - Content delivery is one flavor
    - But in general, why should we even have to formulate a query?

What's next? Fourth generation: **From Information Retrieval to Information Supply**



Bibliography – General

- **Modern Information Retrieval**

  by R. Baeza-Yates & B. Ribeiro-Neto, Addison-Wesley, 1999. Second edition in 2011.

- **Introduction to Information Retrieval**

  by C. Manning, P. Raghavan & H. Schuetze,

  Cambridge University Press, 2008.

- **Web Search: The Role of the Users.**

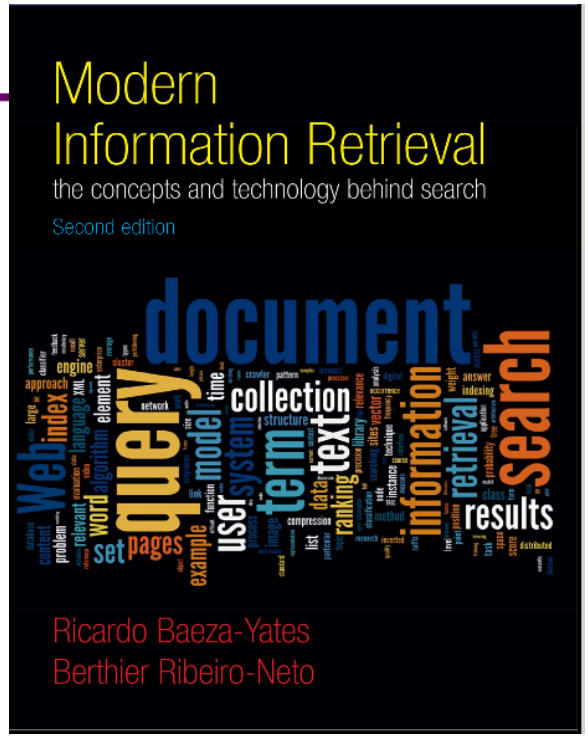
  by R. Baeza-Yates & Y. Maarek, 2011.

  Tutorial at SIGIR 2011 and other conferences.

- **Websites:**
  - http://www.searchenginewatch.com/
  - http://www.searchengineshowdown.com/

- **Main upcoming conferences:**

  SPIRE 2011, October, Pisa, Italy
  WSDM 2012, February, Seattle, USA
  ECIR 2012, April, Barcelona, Spain
  ACM SIGIR 2012, July, Portland, USA



Distributed Web Search

Ricardo Baeza-Yates
Yahoo! Research
Barcelona, Spain & Santiago, Chile

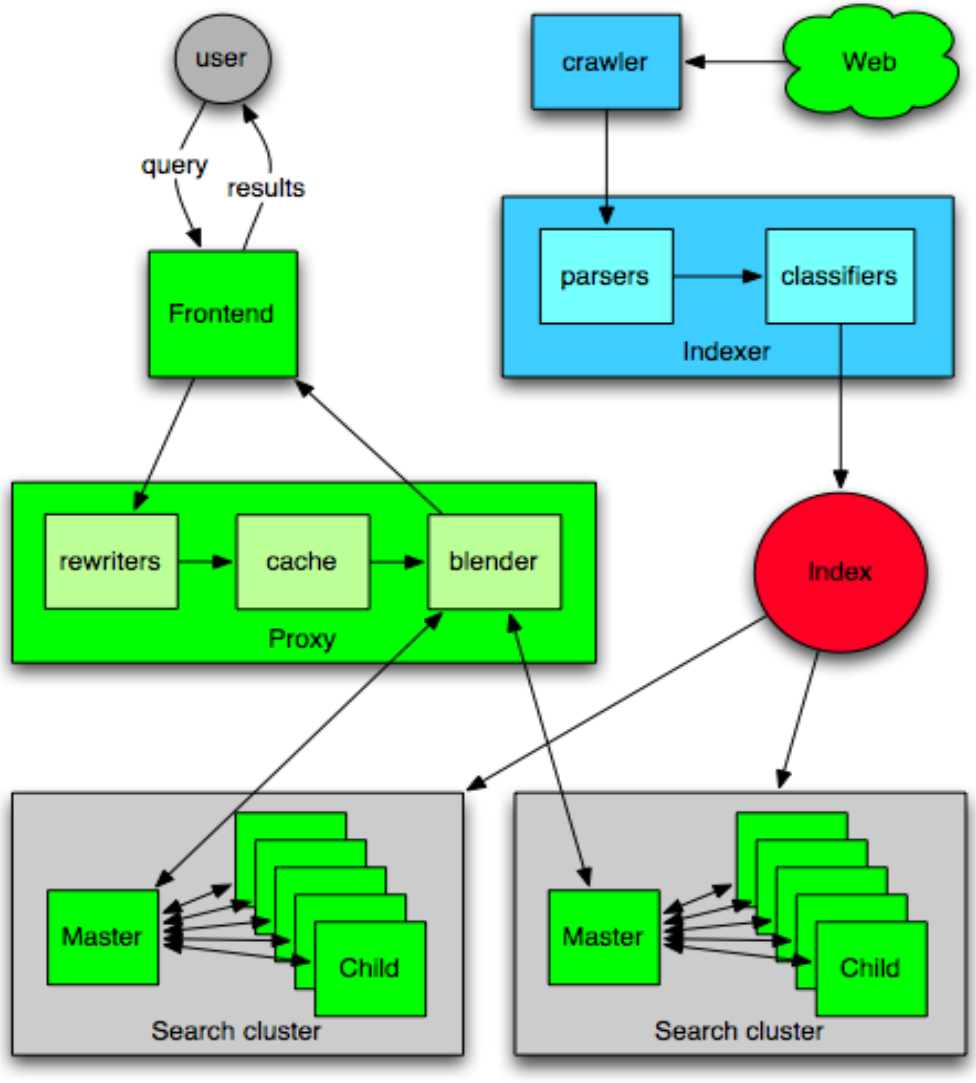
ESSIR 2011, Koblenz, Germany

Agenda

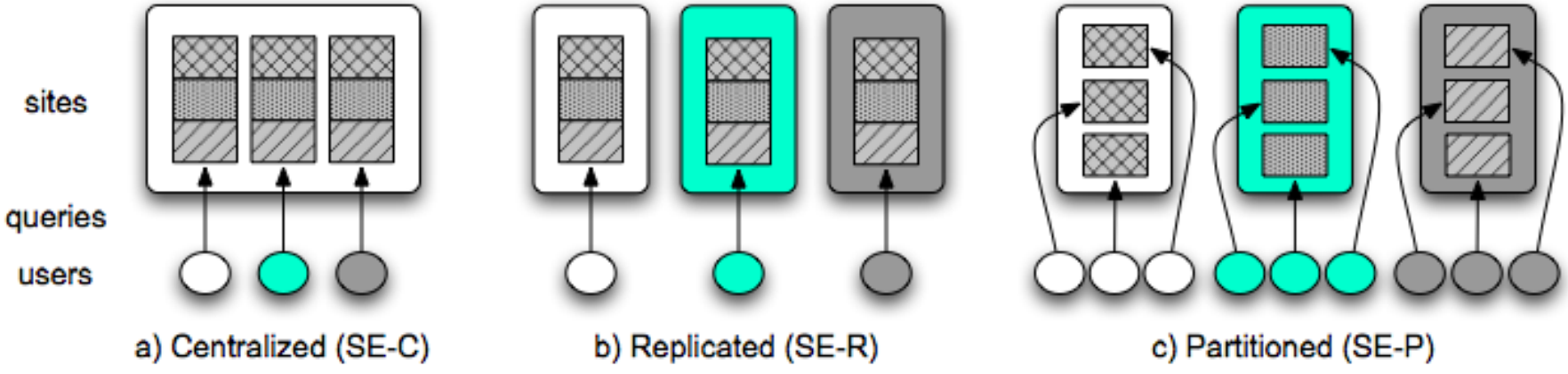
- Challenges
- Crawling
- Indexing
- Caching
- Query Processing

A Typical Web Search Engine

- Caching
  - result cache
  - posting list cache
  - document cache
- Replication
  - multiple clusters
  - improve throughput
- Parallel query processing
  - partitioned index
    - document-based
    - term-based
  - Online query processing



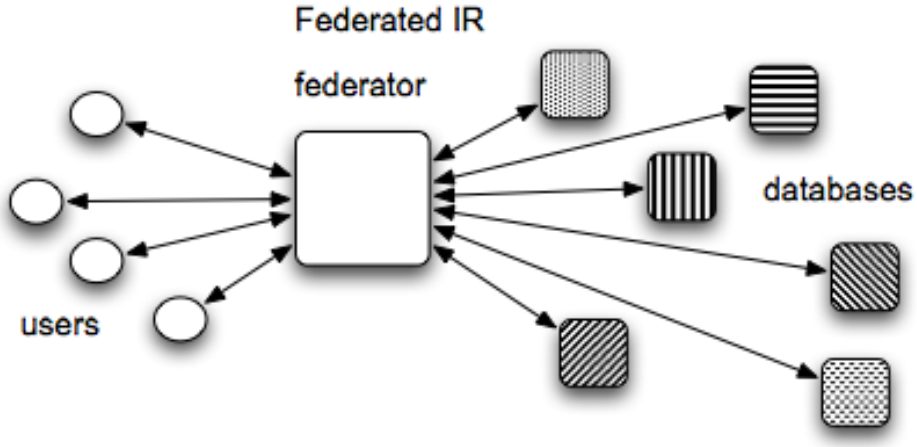
Search Engine Architectures



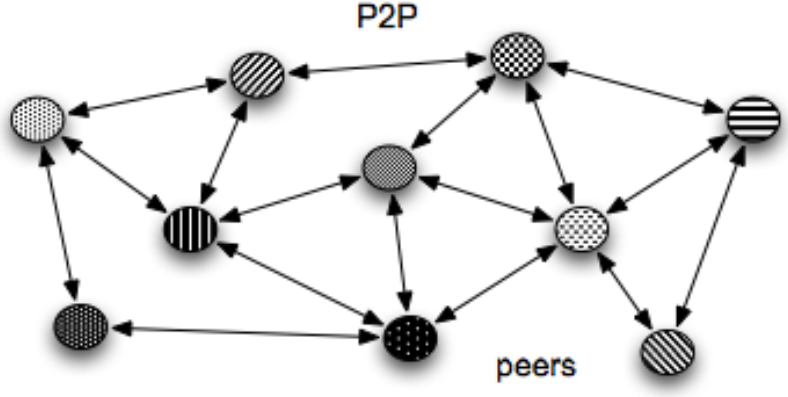
- **Architectures differ in**
  - **number of data centers**
  - **assignment of users to data centers**
  - **assignment of index to data centers**

Related Distributed Search Architectures

- **Federated search**
  - **autonomous search sites**
  - **no explicit data partitioning**
  - **heterogeneous algorithms and resources**
  - **no dedicated network**



- **P2P search**
  - **high number of peers**
  - **dynamic and volatile peers**
  - **low cost systems**
  - **completely autonomous**



System Size

- 20 billion Web pages implies at least 100Tb of text
- The index in RAM implies at least a cluster of 10,000 PCs
- Assume we can answer 1,000 queries/sec
- 350 million queries a day imply 4,000 queries/sec
- Decide that the peak load plus a fault tolerance margin is 3
- This implies a replication factor of 12 giving 120,000 PCs
- Total deployment cost of over 100 million US$ plus maintenance cost
- In 201x, being conservative, we would need over 1 million computers!

Questions

- Should we use a centralized system?
- Can we have a (cheaper) distributed search system in spite of network latency?

- Preliminary answer: **Yes**
- Solutions: caching, new ways of partitioning the index, exploit locality when processing queries, prediction mechanisms, etc.

Advantages

- Distribution decreases replication, crawling, and indexing and hence the cost per query
- We can exploit high concurrency and locality of queries
- We could also exploit the network topology
- Main design problems:
  - Depends upon many external factors that are seldom independent
  - One poor design choice can affect performance or/and costs

Challenges

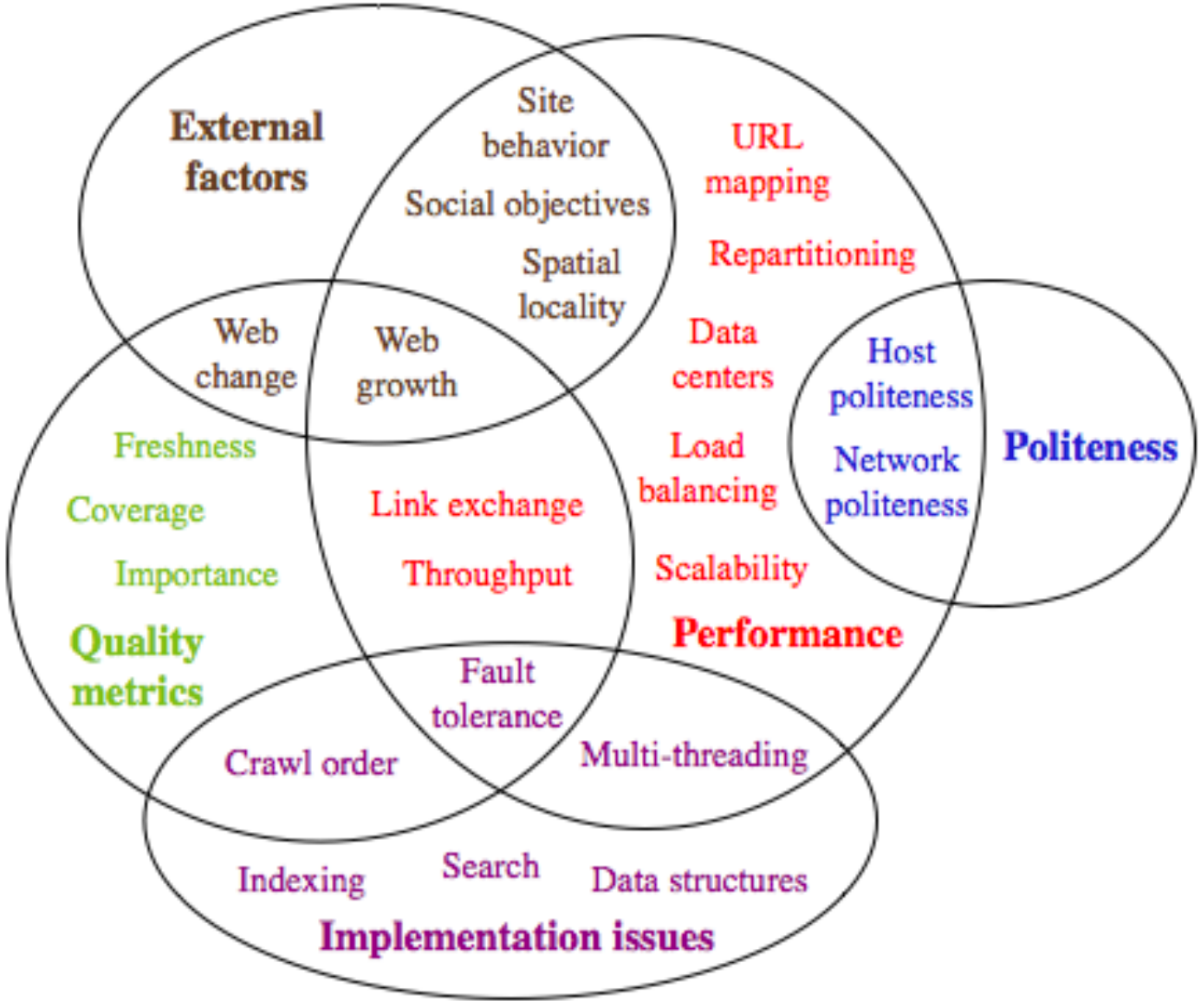
- Must return high quality results
  (handle quality diversity and fight spam)
- Must be fast (fraction of a second)
- Must have high capacity
- Must be dependable
  (reliability, availability, safety and security)
- Must be scalable

Crawling

- Index depends on good crawling
  - Quality, quantity, freshness
- Crawling is a scheduling problem
  - NP hard
- Difficult to optimize and to evaluate
- Distributed crawling:
  - Closer to data, less network usage and latency

Too Many Factors

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

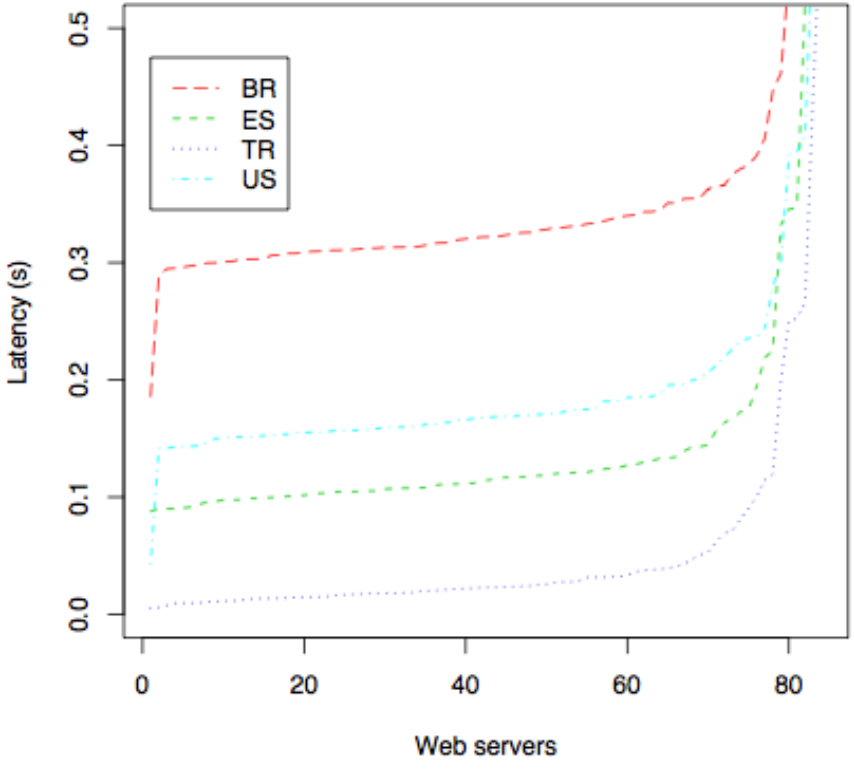


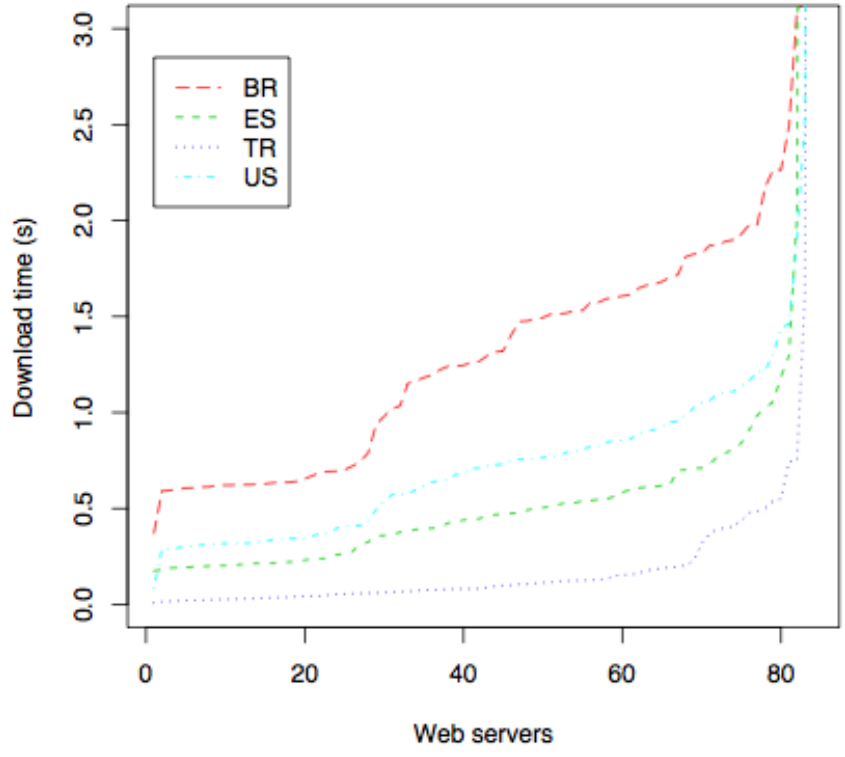
YAHOO!

Experimental Setup

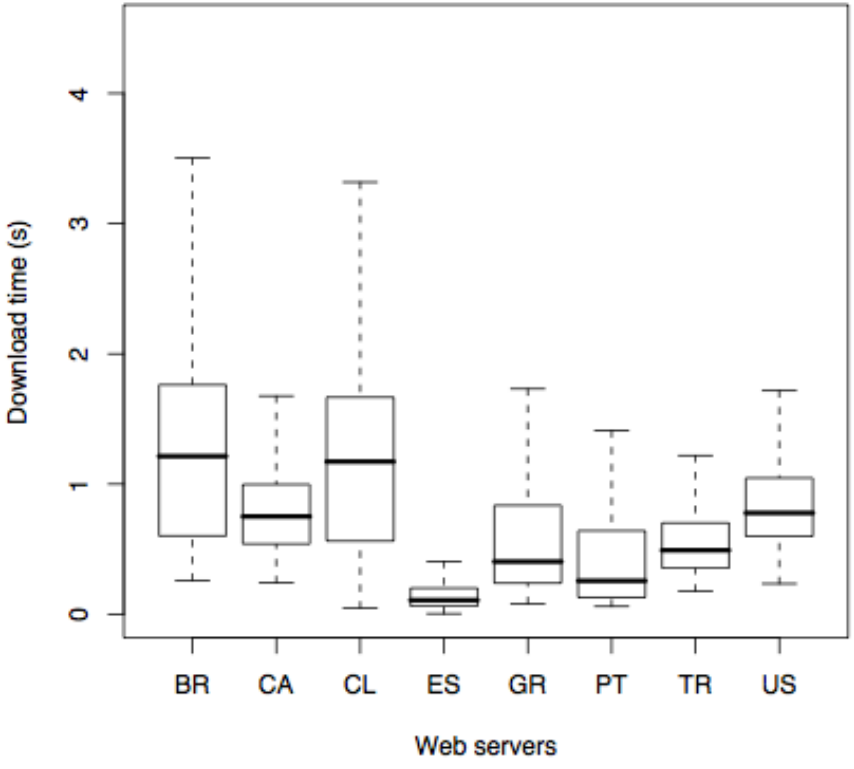
- Network access statistics over the .edu domains
  - using a customized echoping version
  - over one week
- Eight crawled countries
  - US, Canada
  - Brazil, Chile
  - Spain, Portugal
  - Turkey, Greece
- Four crawling countries
  - US
  - Brazil
  - Spain
  - Turkey

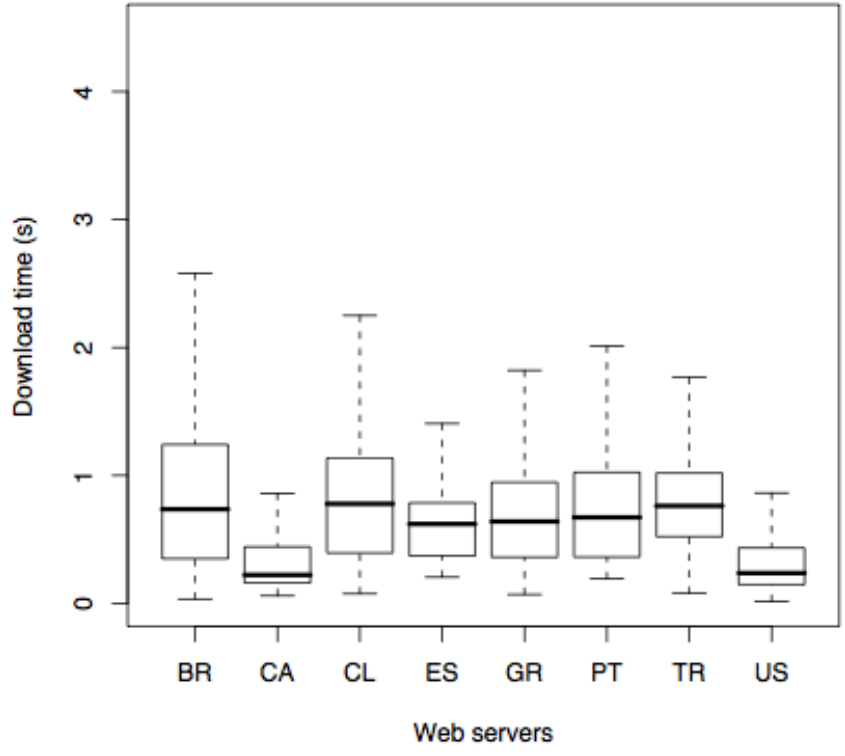
Experimental Results





Experimental Results



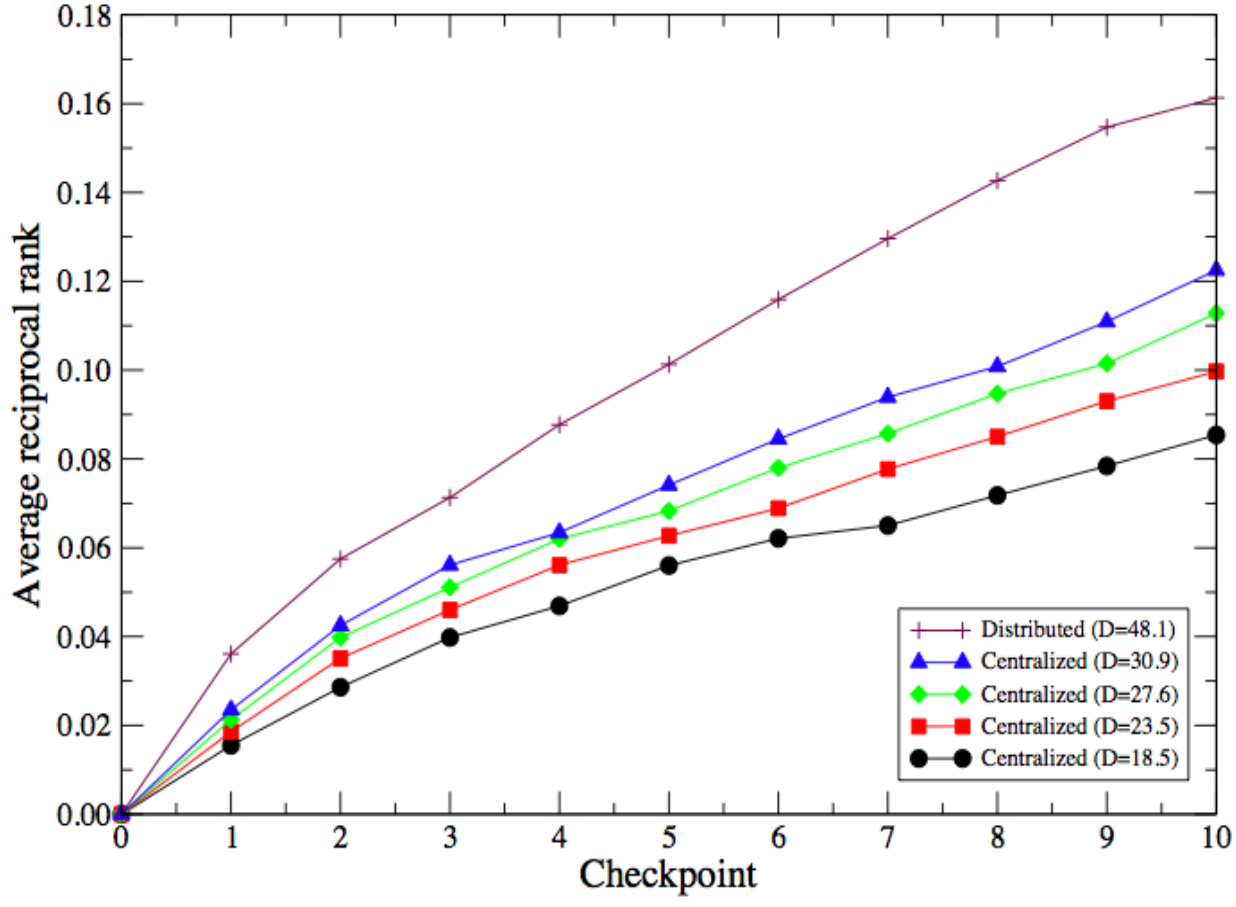


Impact of Distributed Web Crawling on Relevance [Cambazoglu et al, SIGIR 2009]

- Objective: See the impact of higher page download rates on search quality
- Random sample of 102 million pages partitioned into five different geographical regions
  - location of Web servers
  - page content
- Query sets from the same five regions
- Ground-truth: clicks obtained from a commercial search engine
- Ranking: a linear combination of a BM25 variant and a link analysis metric
- Search relevance: average reciprocal rank

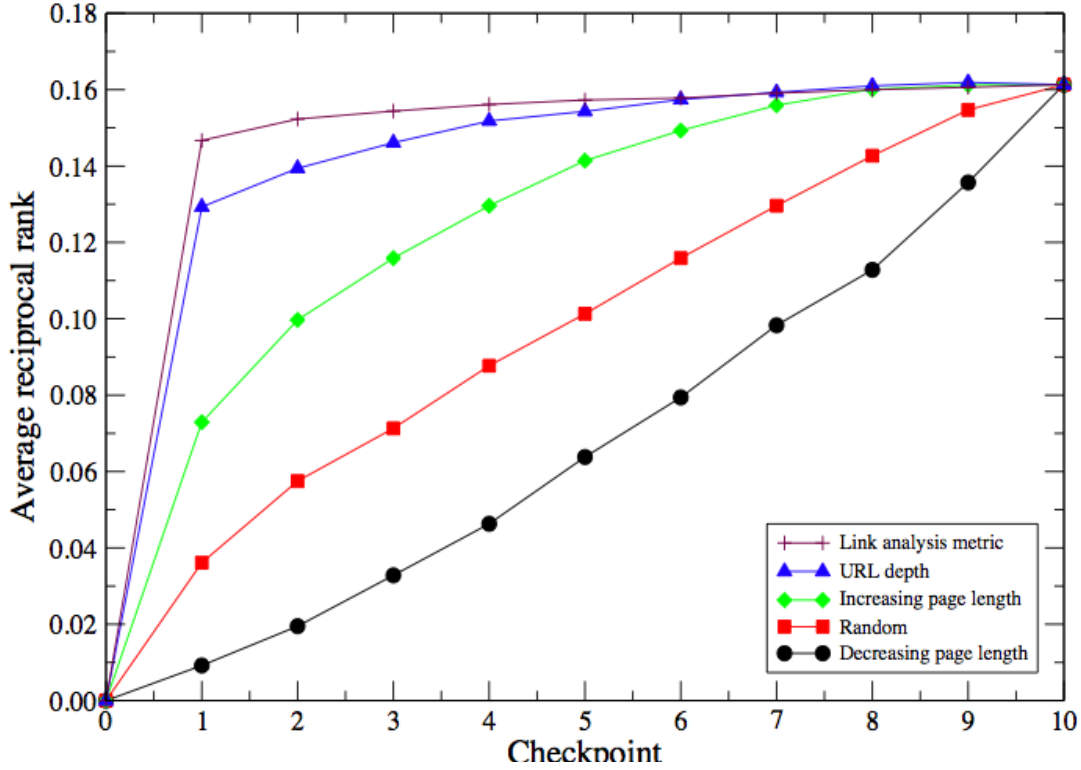
Impact of Download Speed

- Distributed crawling simulator with varying download rates
  - distributed: 48 KB/s
  - centralized:
    - 30.9 KB/s (US)
    - 27.6 KB/s (Spain)
    - 23.5 KB/s (Brazil)
    - 18.5 KB/s (Turkey)
- Checkpoint *i*: the point where the fastest crawler in the experiment downloaded 10*i* % of all pages
- Crawling order: random



Impact of Crawling Order

- **Varying crawling orders:**
  - **link analysis metric**
  - **URL depth**
  - **increasing page length**
  - **random**
  - **decreasing page length**
- **Download throughput: 48.1 KB/s**



	D			
Ordering strategy	18.5	23.5	27.6	30.9
Decreasing page length	0.041	0.058	0.072	0.086
Random	0.085	0.100	0.113	0.123
Increasing page length	0.130	0.143	0.150	0.154
URL depth	0.153	0.156	**0.158**	**0.159**
Link analysis metric	**0.156**	**0.158**	0.157	**0.159**

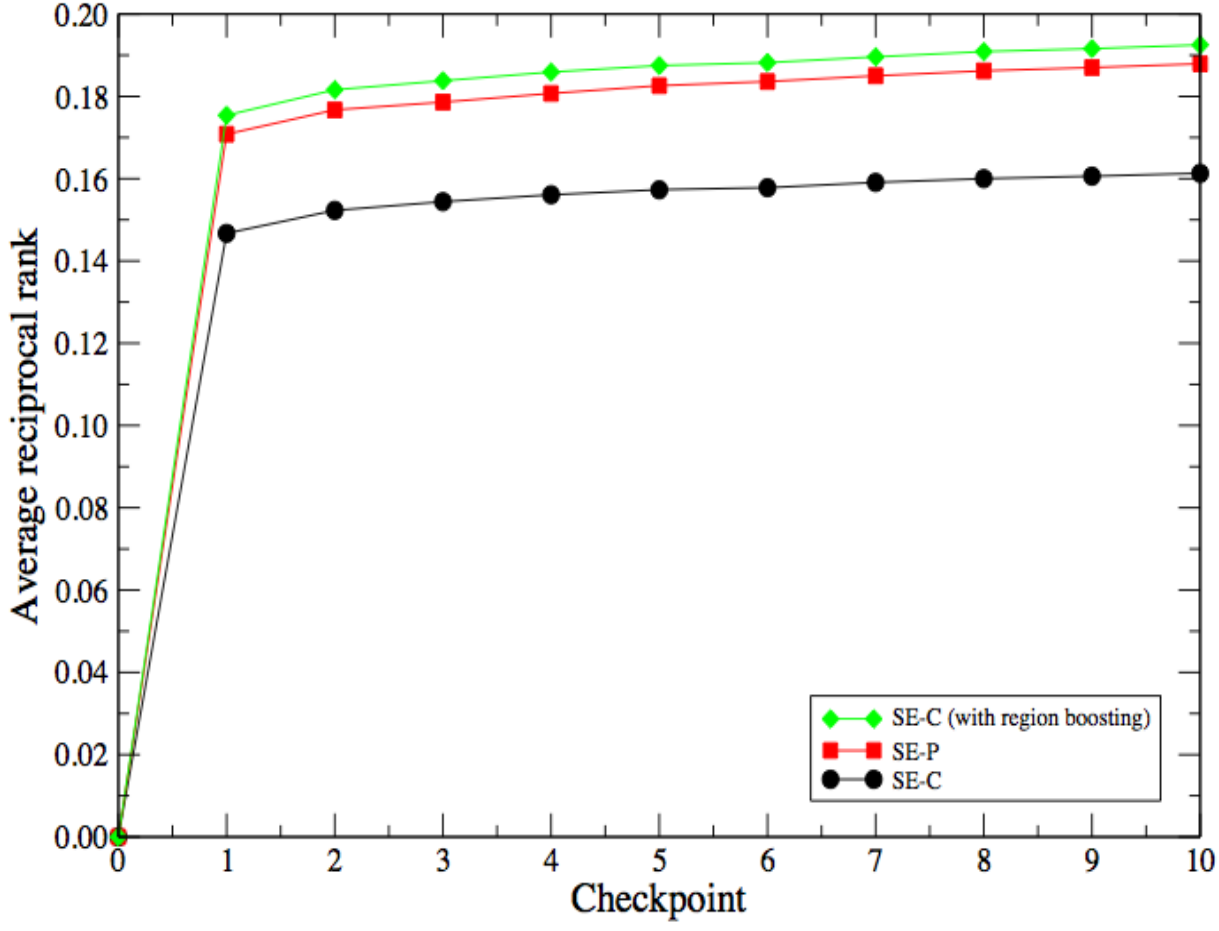
Impact of Region Boosting

- **Region boosting**
  - **SE-C**

  (with region boosting)
  - **SE-P**

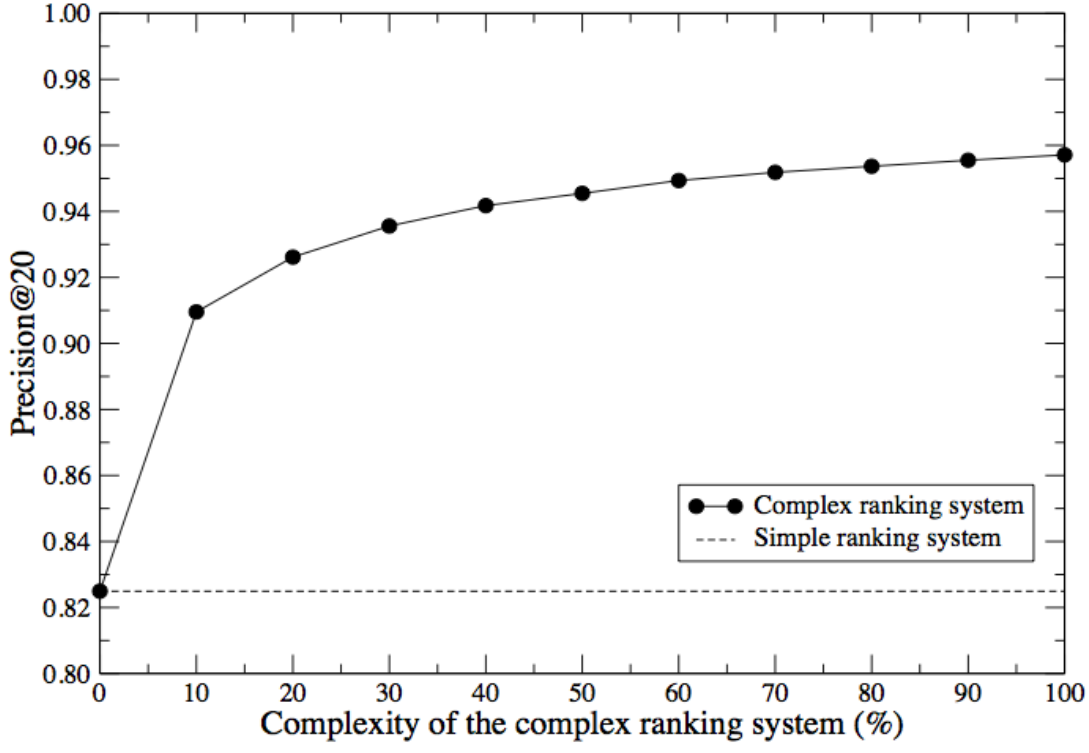
  (natural region boosting)
  - **SE-C**

  (without region boosting)
- **Download throughput: 48.1 KB/s**



Search Relevance (Cambazoglu et al, SIGIR 2009)

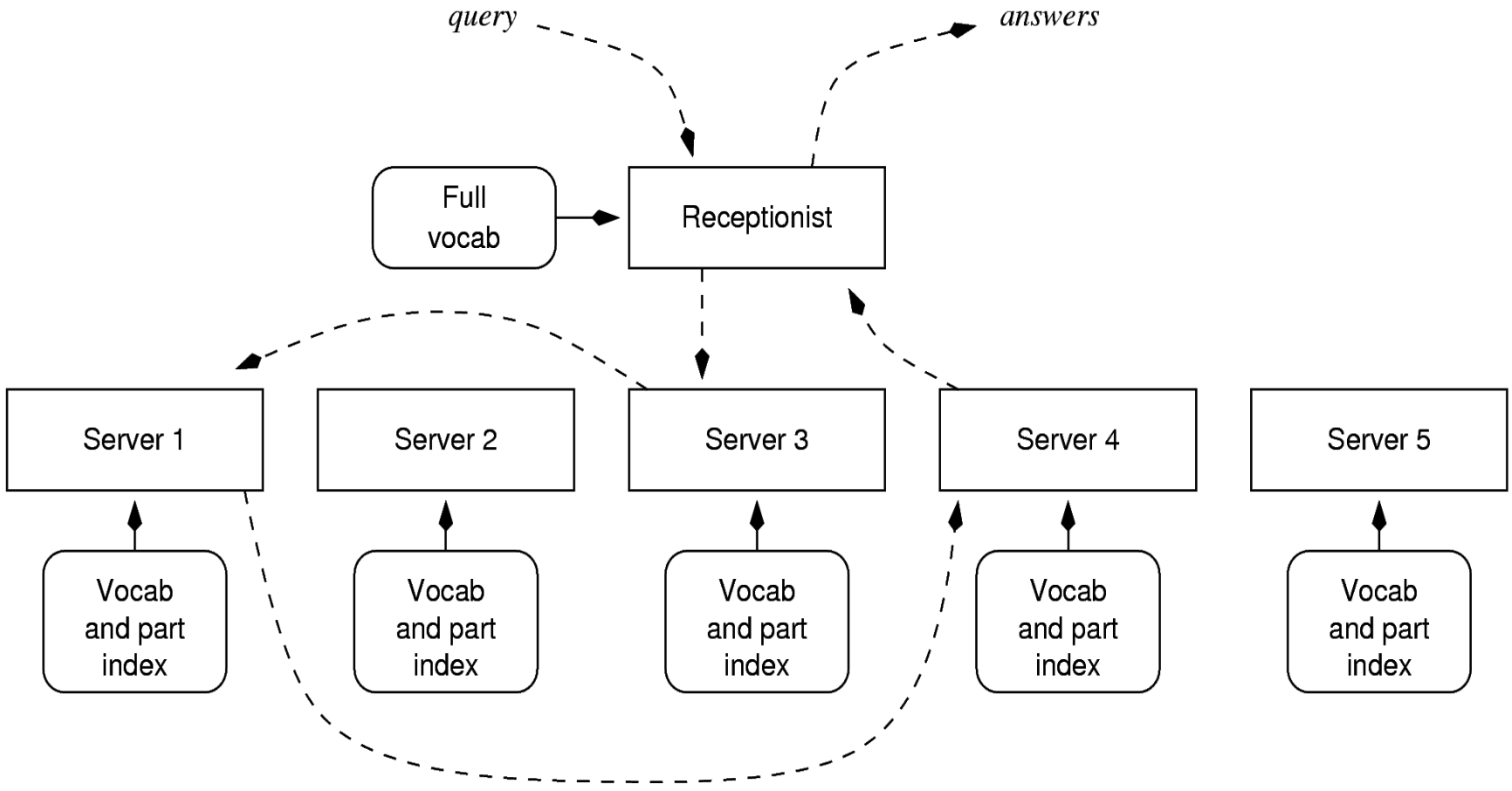
- Assuming we have more time for query processing, we can
  - relax the "AND" requirement
  - score more documents
  - use more complex scoring techniques
    - costly but accurate features
    - costly but accurate functions
- Ground-truth: top 20 results
- Baseline: linear combination of a BM25 variant with a link analysis metric
- A complex ranking function composed of 1000 scorers



Indexing

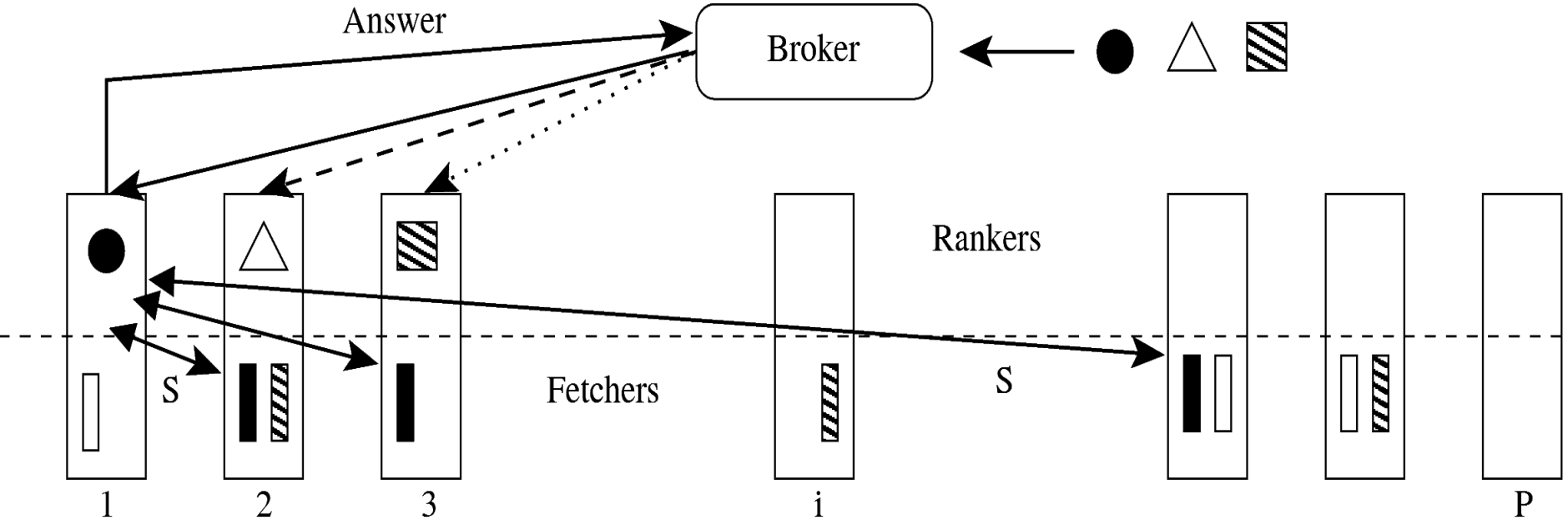
- Distributed: the main **open** problem?
- Document partitioning is natural
- Mixing partitionings:
  - Improves search
  - Does not improve indexing
- More on collection selection?
  - Puppin *at al,* 2010

Query Processing: Pipelining



Term partitioning case, Moffat et al, 2007

Query Processing: Round Robin



Works for both partitionings

Marin et al, 2008

Caching basics

- A cache is characterized by its size and its eviction policy
- *Hit* : requested item is already in the cache
- *Miss* : requested item is not in the cache

- Caches speed up access to frequently or recently used data
  - Memory pages, disk, resources in LAN / WAN

Caching

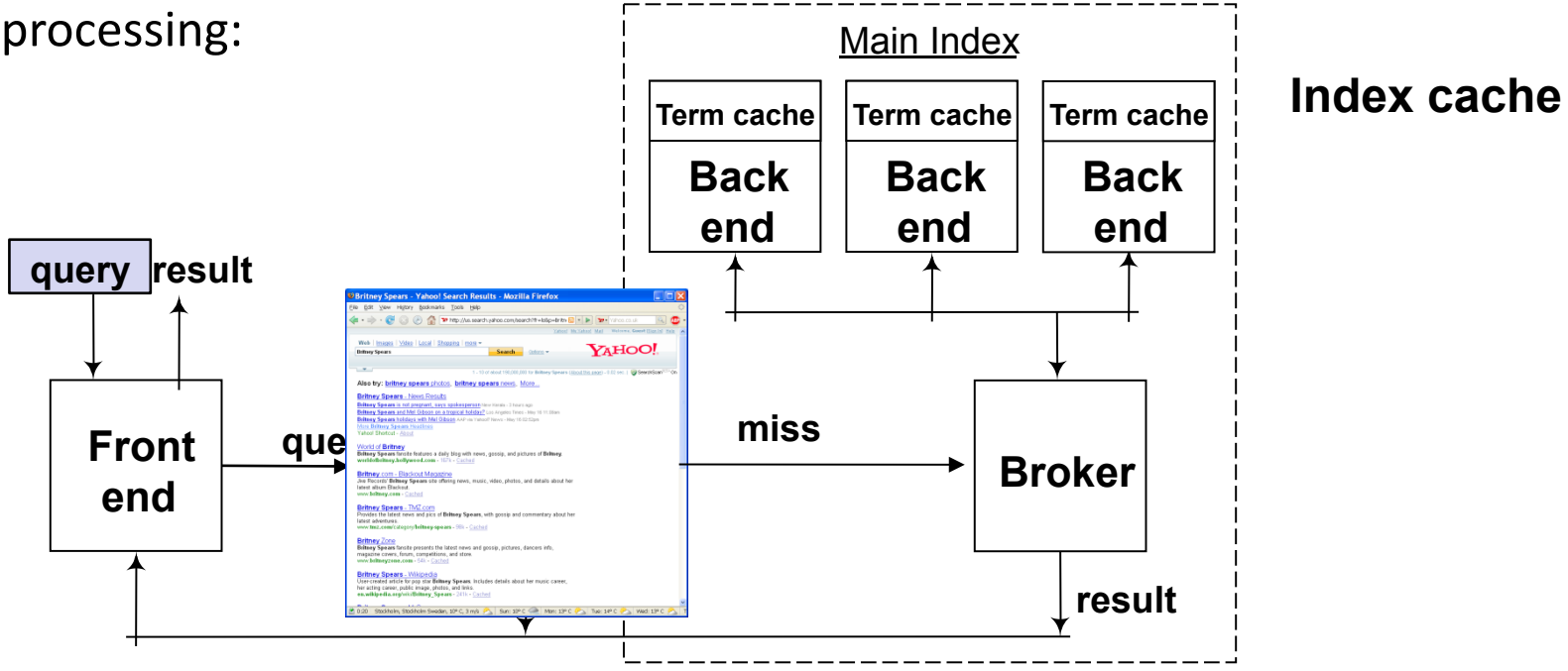
- Caching can save significant amounts of computational resources
  - Search engine with capacity of 1000 queries/second
  - Cache with 30% hit ratio increases capacity to 1400 queries/second
- Caching helps to make queries “local”
- Caching is similar to replication on demand
- Important sub-problem:
  - Refreshing stale results (Cambazoglu *et al*, WWW 2010)

Caching in Web Search Engines

- Caching **query results** *versus* caching **posting lists**
- **Static** *versus* **dynamic** caching policies
- Memory allocation between different caches
- Caching reduce **latency** and **load** on back-end servers
- Baeza-Yates et al, SIGIR 2007

Caching at work

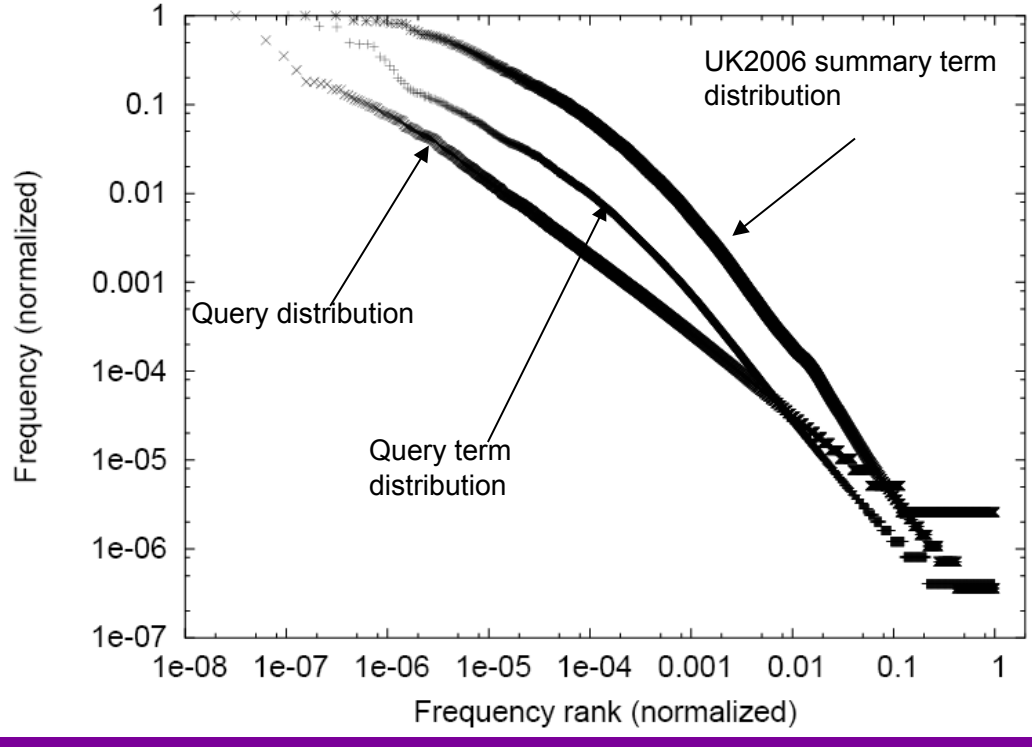
Query processing:



- Caching reduce **latency** and **load** on back-end servers

Data Characterization

- 1 year of queries from Yahoo! UK
- UK2006 summary collection
- Pearson correlation between query term frequency and document frequency = 0.424



What you write is NOT what you want

Caching Query Results or Term Postings?

- Queries
  - 50% of queries are unique (vocabulary)
  - 44% of queries are singletons (appear only once)
  - Infinite cache achieves 50% hit-ratio
    - Infinite hit ratio = (#queries – #unique) / #queries
- Query terms
  - 5% of terms are unique
  - 4% of terms are singletons
  - Infinite cache achieves 95% hit ratio

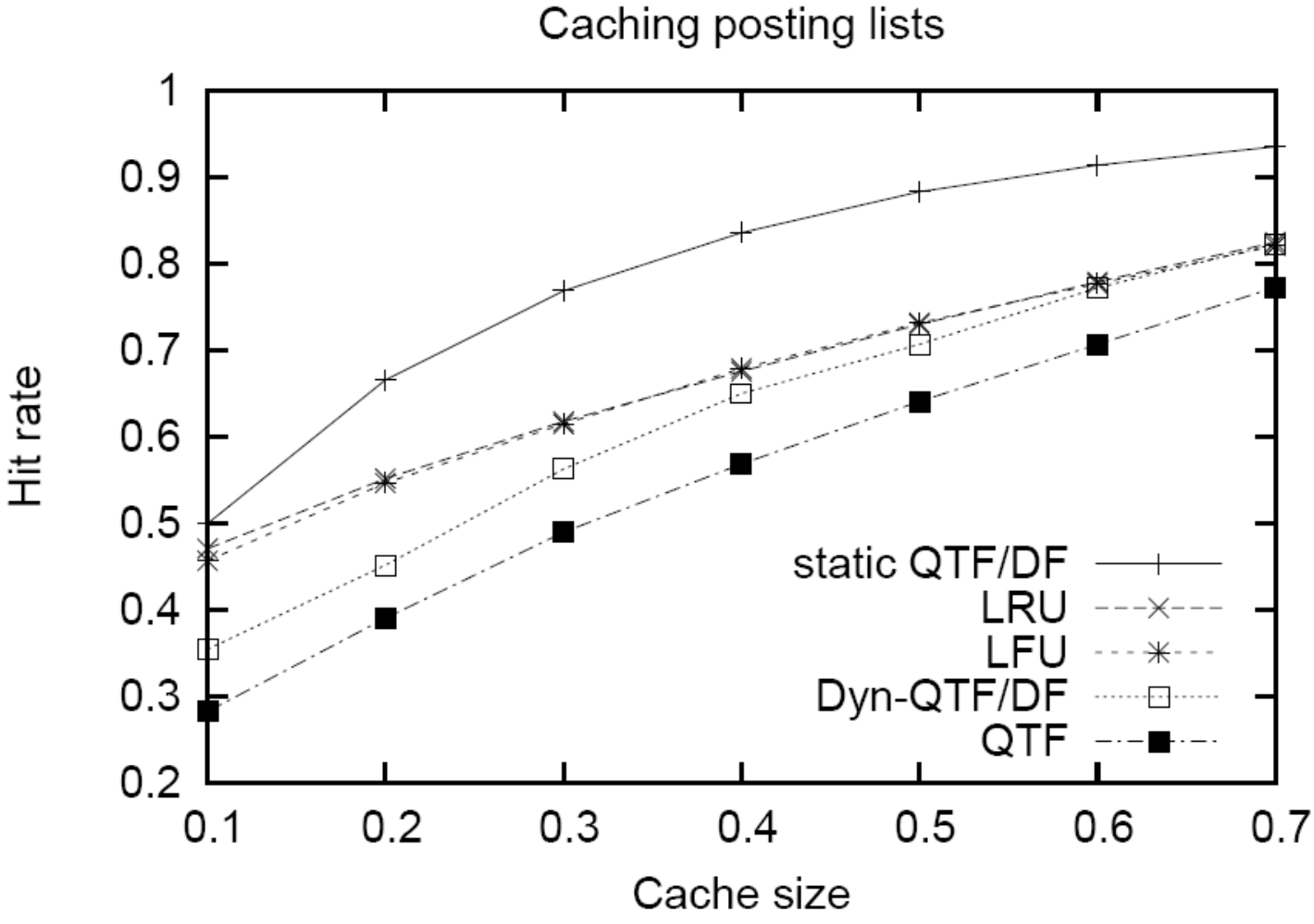
Static Caching of Postings

- QTF for static caching of postings (Baeza-Yates & Saint-Jean, 2003):
  - Cache postings of terms with the highest $f_q(t)$
- Trade-off between $f_q(t)$ and $f_d(t)$
  - Terms with high $f_q(t)$ are good to cache
  - Terms with high $f_d(t)$ occupy too much space
- QTFDF: Static caching of postings
  - Knapsack problem:
  - Cache postings of terms with the highest $f_q(t)/f_d(t)$

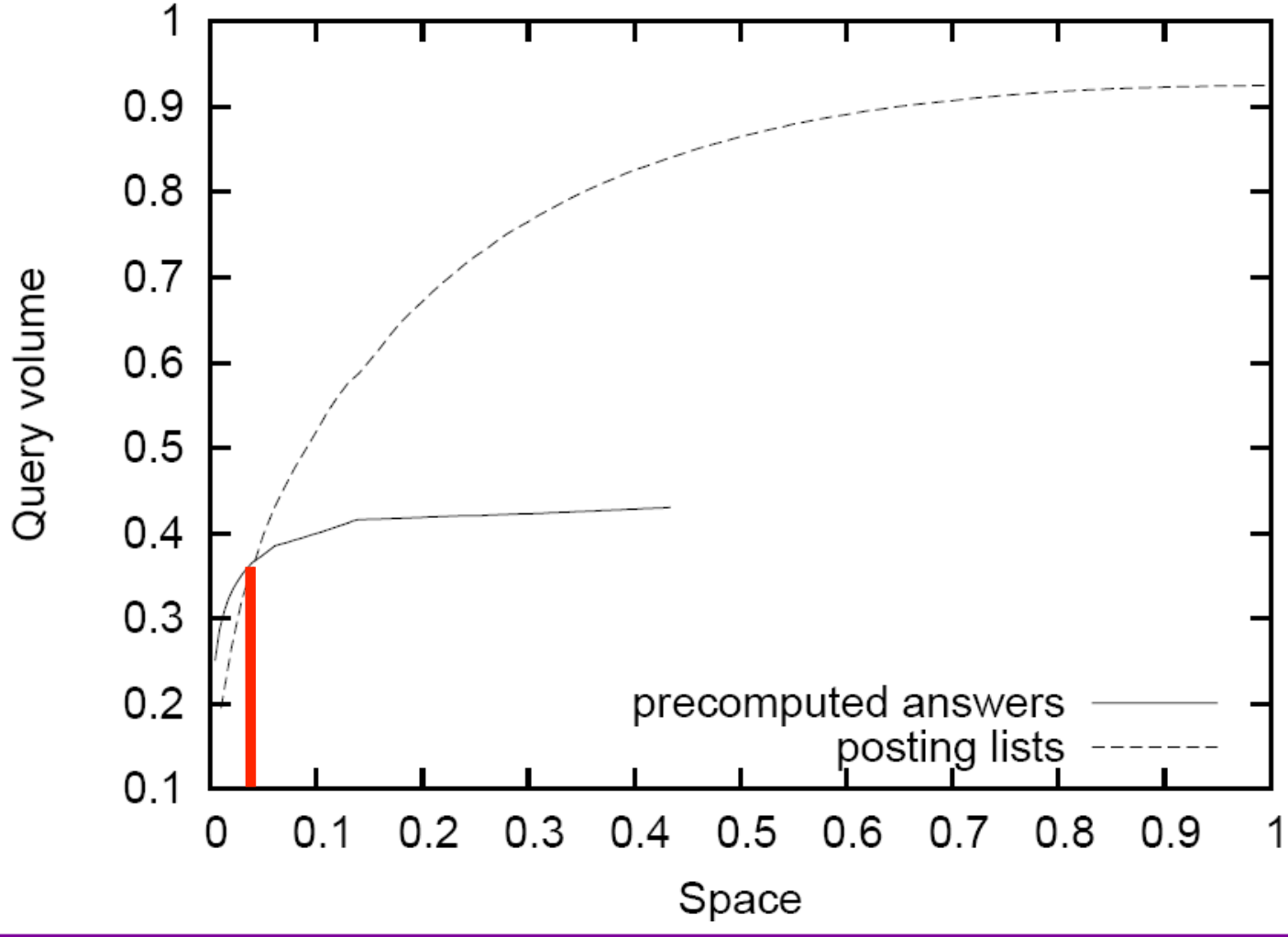
Evaluating Caching of Postings

- Static caching:
  - Q_{TF} : Cache terms with the highest query log frequency $f_q(t)$
  - $Q_{TF}D_F$: Cache terms with the highest ratio $f_q(t) / f_d(t)$
- Dynamic caching:
  - LRU, LFU
  - Dynamic $Q_{TF}D_F$: Evict the postings of the term with the lowest ratio $f_q(t) / f_d(t)$

Results


Caching posting lists
Hit rate
Cache size
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
static QTF/DF
LRU
LFU
Dyn-QTF/DF
QTF

Combining caches of query results and term postings



Experimental Setting

- Process 100K queries on the UK2006 summary collection with Terrier
- Centralized IR system
  - Uncompressed/compressed posting lists
  - Full/partial query evaluation
- Model of a distributed retrieval system
  - broker communicates with query servers over LAN or WAN

Parameter Estimation

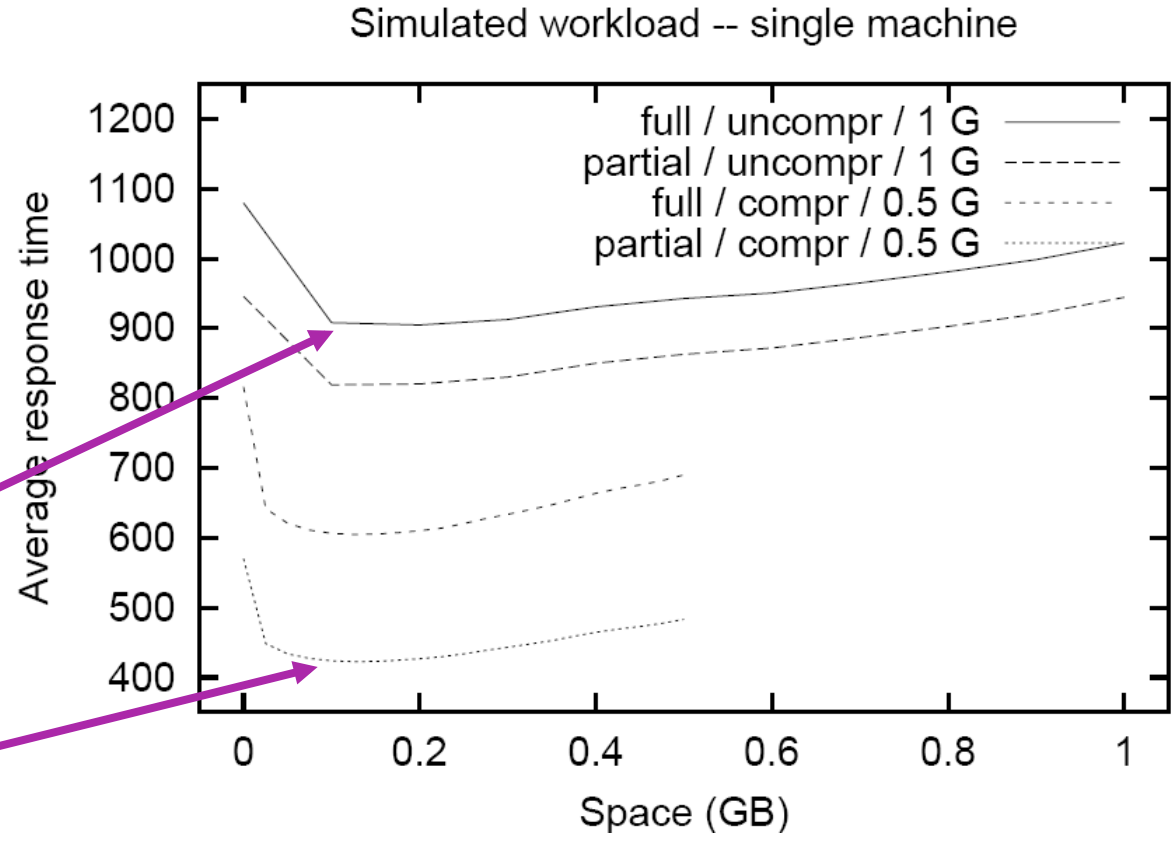
- The average ratio between the time to return an answer computed from posting lists and from the query result cache is:
  - TR_1 : when postings are in memory
  - TR_2 : when postings are on disk
  - M is the cache size in answer units
    - A cache of query results stores $N_c=M$ queries
  - L is the average posting list size
    - A cache of postings stores $N_p=M/L= N_c/L$ posting lists

Parameter Values

	Uncompressed Postings (L=0.75)		Compressed Postings (L'=0.26)	
Centralized system	TR_1	TR_2	TR_1'	TR_2'
Full evaluation	233	1760	707	1140
Partial evaluation	99	1626	493	798
WAN system	TR_1	TR_2	TR_1'	TR_2'
Full evaluation	5001	6528	5475	5908
Partial evaluation	4867	6394	5270	5575

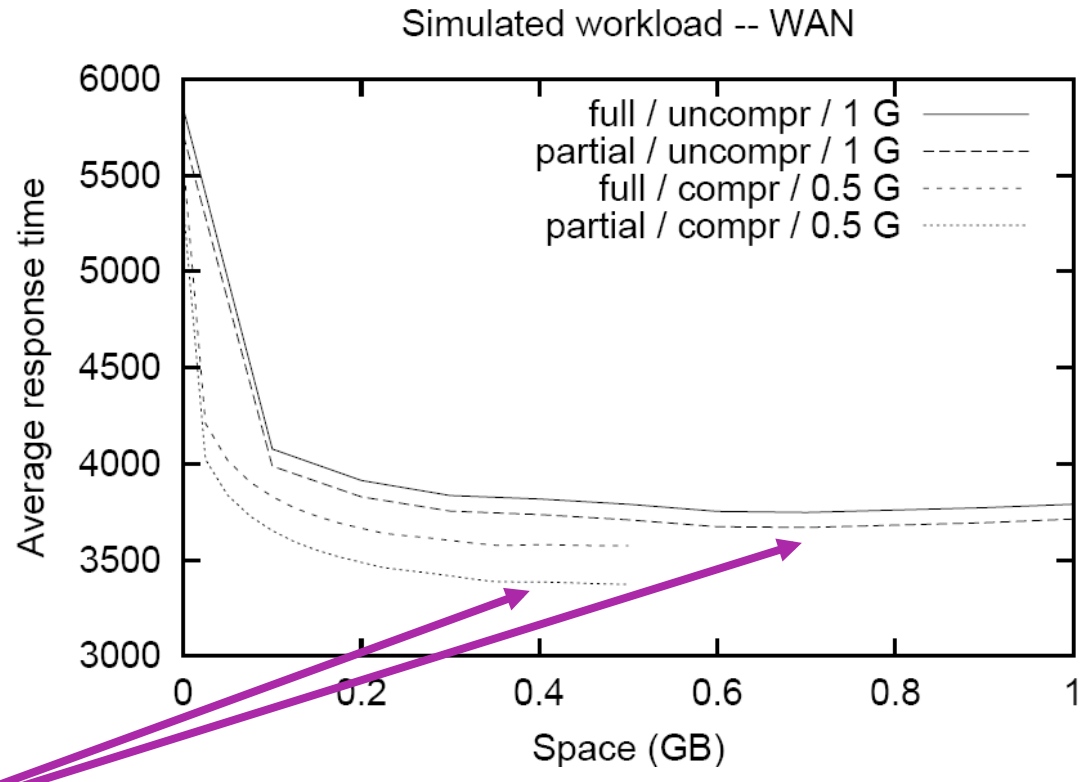
Centralized System Simulation

- Assume M memory units
  - x memory units for static cache of query results
  - M-x memory units for static cache of postings
- Full query evaluation with uncompressed postings
  - 15% of M for caching query results
- Partial query evaluation with compressed postings
  - 30% of M for caching query results



WAN System Simulation

- Distributed search engine
  - Broker holds query results cache
  - Query processors hold posting list cache
- Optimal Response time is achieved when most of the memory is used for caching answers

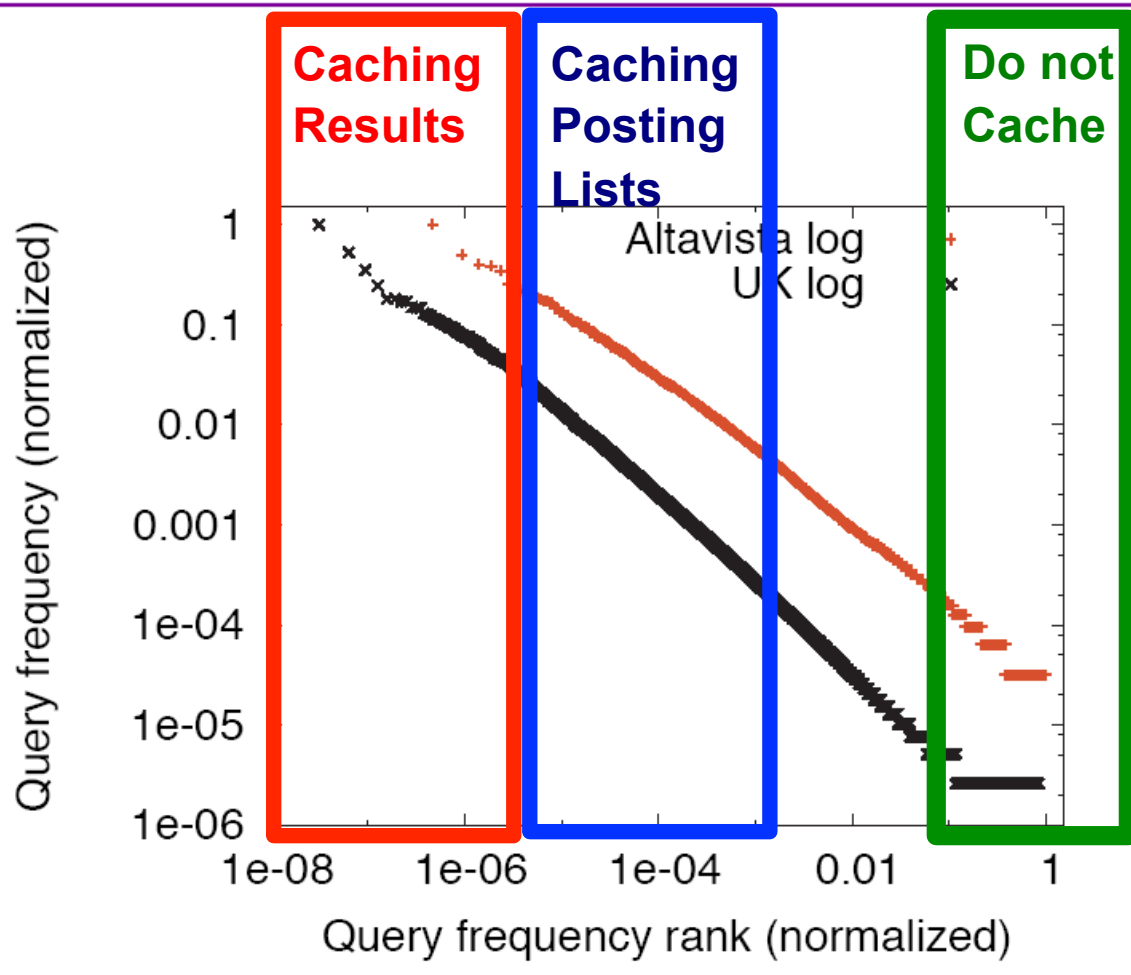


Query Dynamics

- Static caching of query results
  - Distribution of queries change slowly
  - A static cache of query results achieves high hit rate even after a week
- Static caching of posting lists
  - Hit rate decreases by less than 2% when training on 15, 6, or 3 weeks
  - Query term distribution exhibits very high correlation (>99.5%) across periods of 3 weeks

Why caching results can't reach high hit rates

- AltaVista: 1 week from September 2001
- Yahoo! UK: 1 year
  - Similar query length in words and characters
- Power-law frequency distribution
  - Many infrequent queries and even singleton queries
- No hits from singleton queries



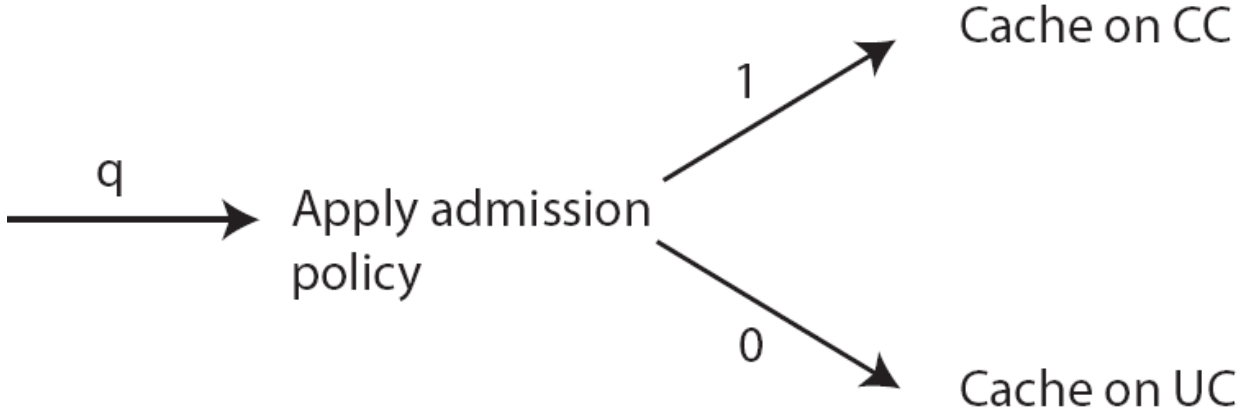
Benefits of filtering out infrequent queries

- Optimal policy does not cache singleton queries
- Important improvements in cache hit ratios

Cache size	Optimal		LRU	
	AV	UK	AV	UK
50k	**67.49**	**32.46**	59.97	17.58
100k	**69.23**	**36.36**	62.24	21.08
250k	**70.21**	**41.34**	65.14	26.65

Admission Controlled Cache (AC)

- General framework for modelling a range of cache policies



- Split cache in two parts
  - Controlled cache (CC)
  - Uncontrolled cache (UC)
- Decide if a query q is frequent enough
  - If yes, cache on CC
  - Otherwise, cache on UC

Baeza-Yates et al, SPIRE 2007

Why an uncontrolled cache?

- Deal with errors in the predictive part
- Burst of new frequent queries
- Open challenge:
  - How the memory is split in both types of cache?

Features for admission policy

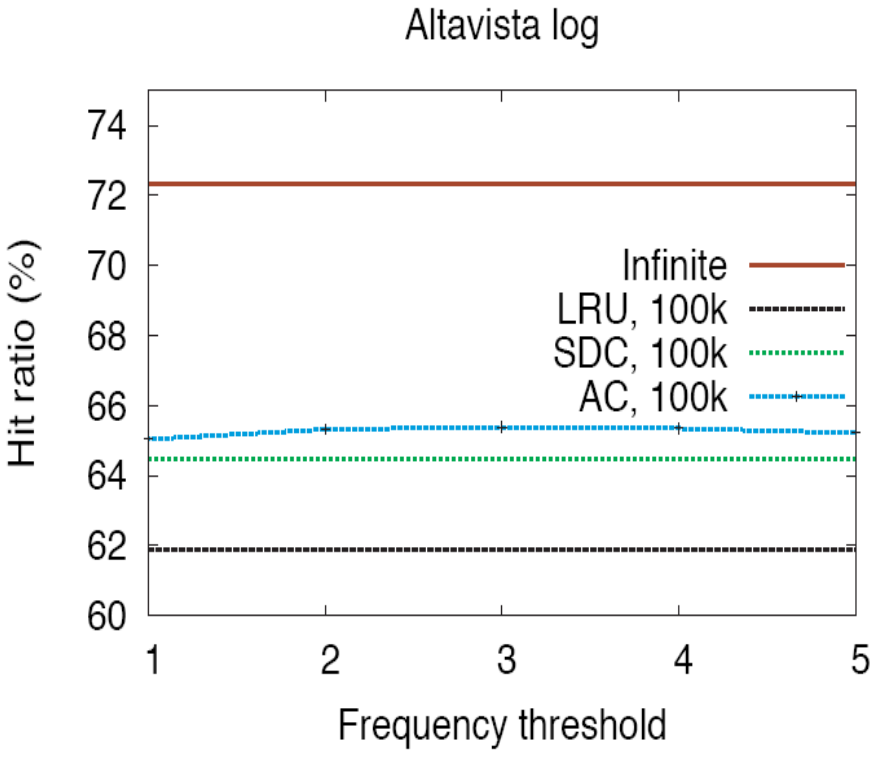
- Stateless features
  - Do not require additional memory
  - Based on a function that we evaluate over the query
  - Example: query length in characters/terms
    - Cache on CC if query length < threshold
- Stateful features
  - Uses more memory to enable admission control
  - Example: past frequency
    - Cache on CC if its past frequency > threshold
    - Requires only **a fraction** of the memory used by the cache

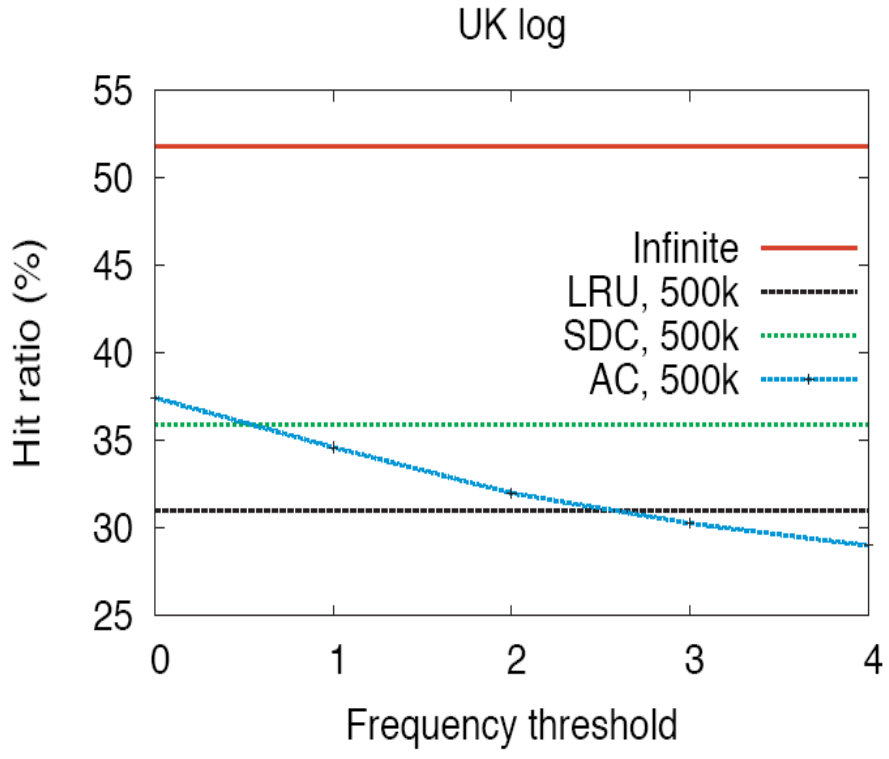
Evaluation

- AltaVista and Yahoo! UK query logs
  - First 4.8 million queries for training
  - Testing on the rest of the queries

- Compare AC with
  - LRU: Evicts the least recent query results
  - SDC: Splits cache into two parts
    - Static: filled up with most frequent past queries
    - Dynamic: uses LRU

Results for Stateful Features





Results for Stateless features

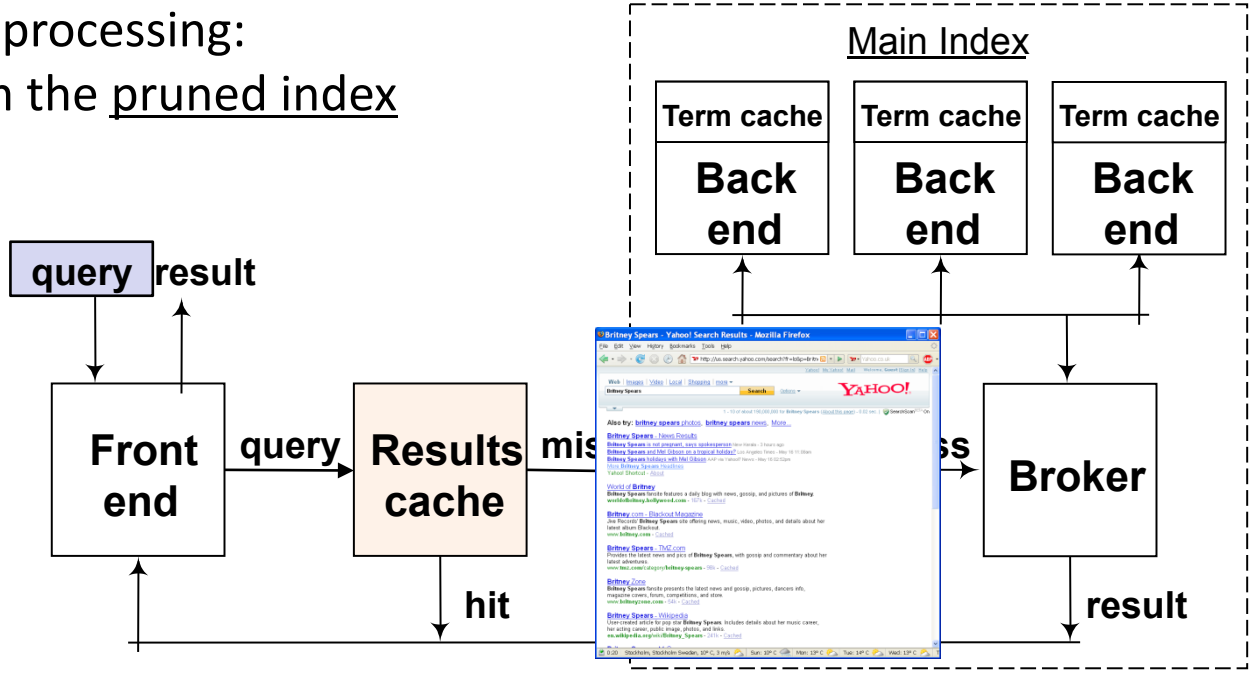
- AC with stateless features outperforms LRU
- Stateless features offer high recall but low precision

	AV		UK	
Infinite	72.32		51.78	
Sizes	50k	100k	100k	500k
LRU	59.49	61.88	21.03	30.96
SDC	**62.25**	**64.49**	**29.61**	**35.91**
AC k_c=10	60.01	59.53	17.07	27.33
AC k_c=20	58.05	62.36	22.85	32.35
AC k_c=30	56.73	61.91	21.60	31.06
AC k_c=40	56.39	61.68	21.19	30.53
AC k_w=2	59.92	62.33	23.10	32.50
AC k_w=3	59.55	61.96	21.94	31.47
AC k_w=4	59.18	61.60	21.16	30.51
AC k_w=5	59.01	61.43	20.81	30.02

Index Pruning

Query processing:

3. from the pruned index

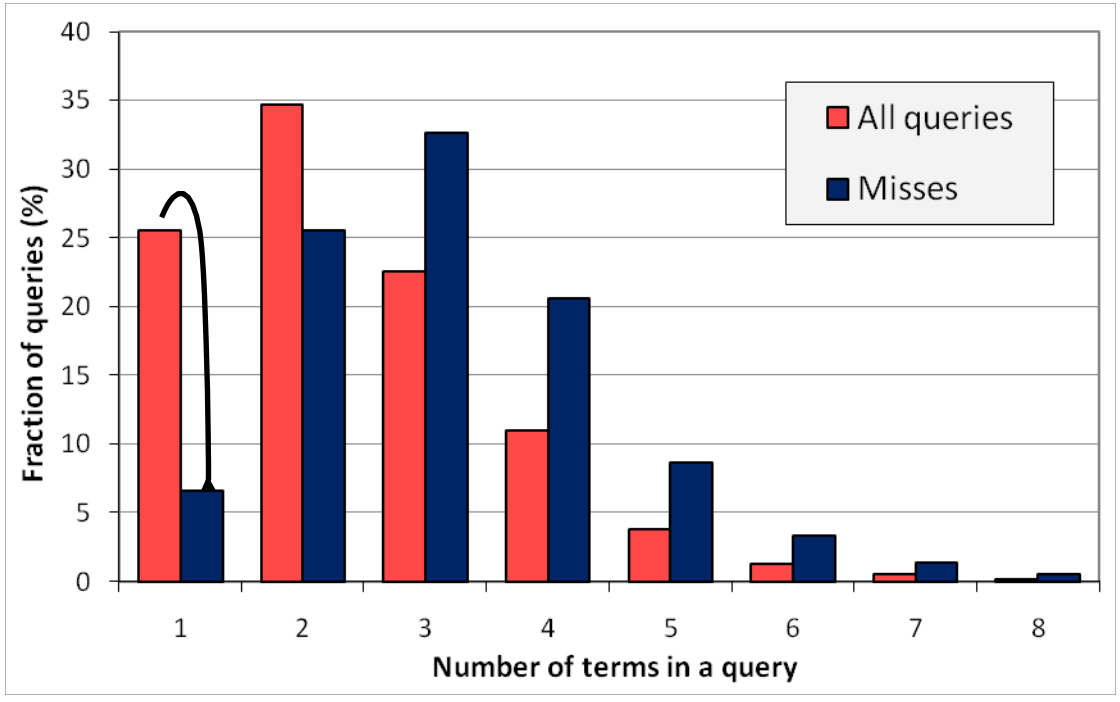


Alternative to term cache

- Results caching and index pruning together
- … to reduce **latency** and **load** on back-end servers

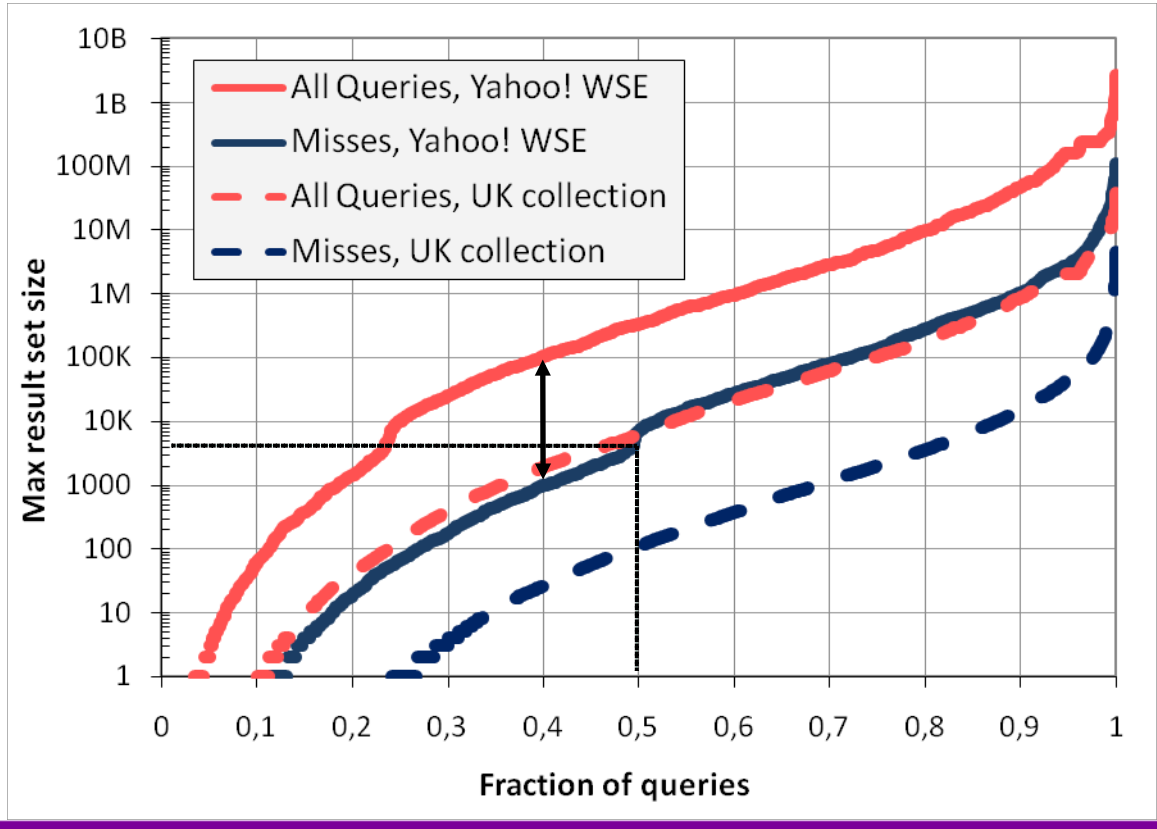
All queries vs. *Misses*: Number of terms in a query

- Average number of terms for *all queries* = **2.4**
  , for *misses* = **3.2**
- Most single term queries are hits in the results cache
- Queries with many terms are unlikely to be hits



All queries vs. *Misses*: Query result size distribution

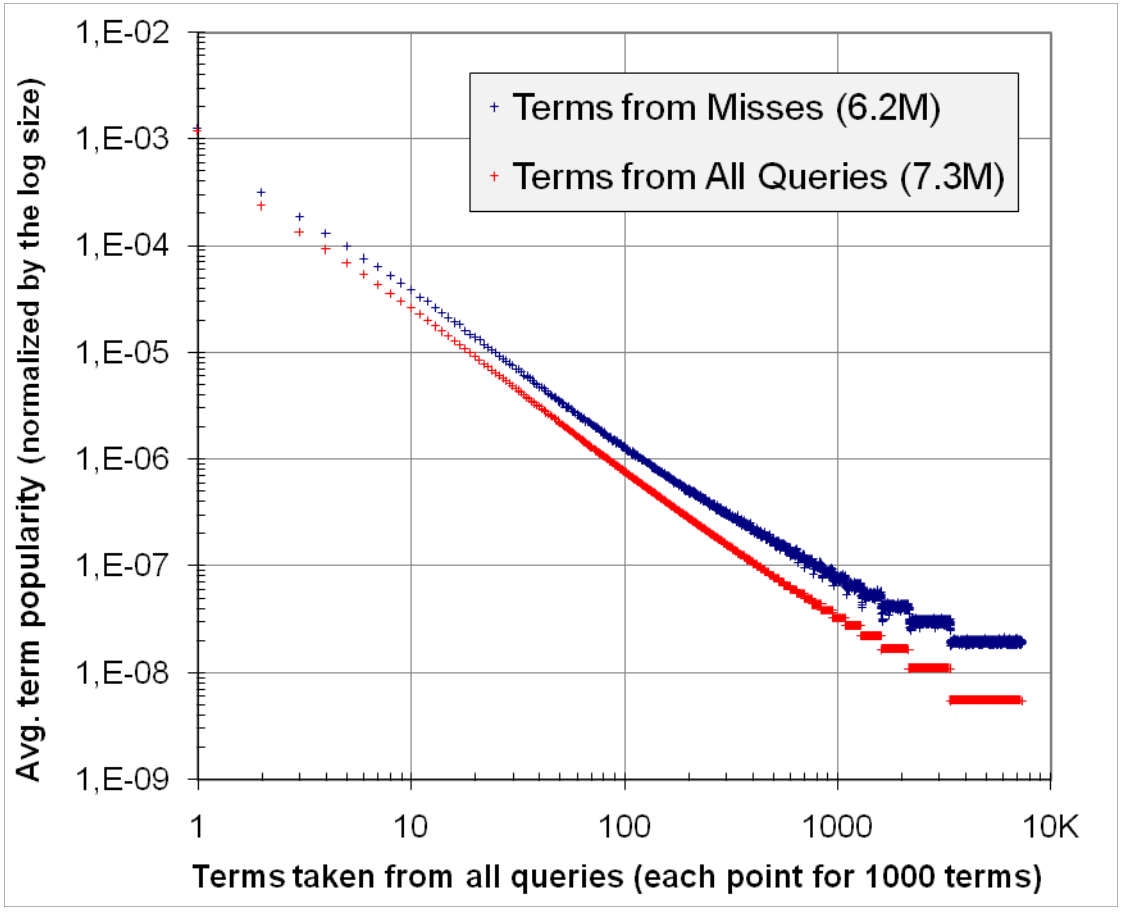
- Randomly selected **2000** queries from *all queries* and *misses*:
- Avg. result size for *misses* is **~100** times smaller than for *all queries*
- Approx. half of the *misses* returns less than **5000** results – SMALL!
- Similar results with a "small" UK document collection (78M)



All queries vs. *Misses*: Term popularity distribution

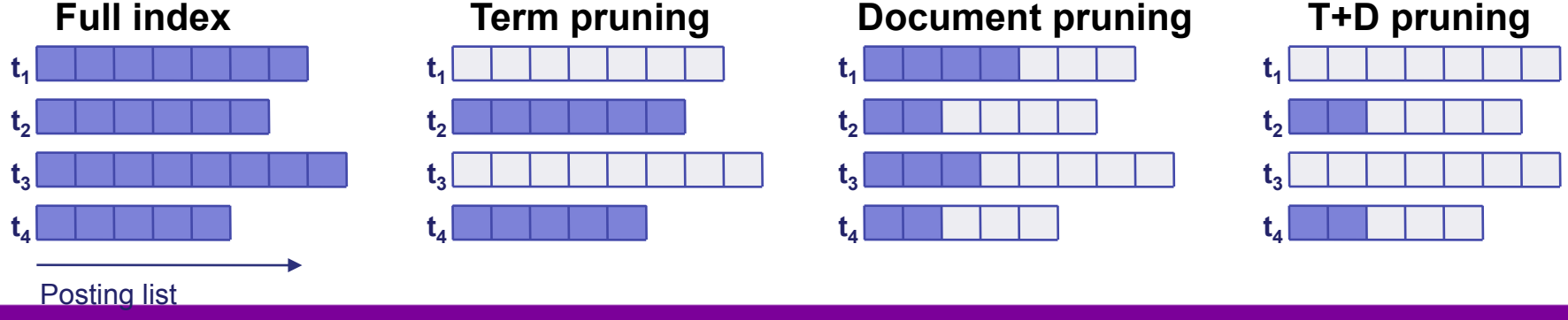
- Each point -> avg. popularity of **1000** consecutive terms
- Popularity is nor-malized by the size of the log
- The order of terms for *misses* is the same as for *all queries*
- Term popularity **does not** change much!

Log sizes: 185M – *all queries*, 41M - *misses*



Static Index Pruning (Skobeltsyn et al, SIGIR08)

- Smaller version of the main index after the cache, returns:
  - the top-*k* response that is *the same* to the main index's, or
  - a *miss* otherwise.
- Assumes Boolean query processing
- Types of pruning:
  - **Term pruning** – full posting lists for selected terms
  - **Document pruning** – prefixes of posting lists
  - **Term+Document pruning** – combination of both



Analysis of Results

- **Static index pruning**: addition to results caching, not replacement
  - **Term pruning** performs well for *misses* also

  => can be combined with results cache
  - **Document pruning** performs well for *all queries,* but requires high Pagerank weights with *misses*
  - **Term+Document pruning** improves over document pruning, but has the same disadvantages

- **Pruned index** grows with collection size

- Document **pruning** targets the same queries as **result caching**

- **Lesson learned:** Important to consider the interaction between the components

Locality

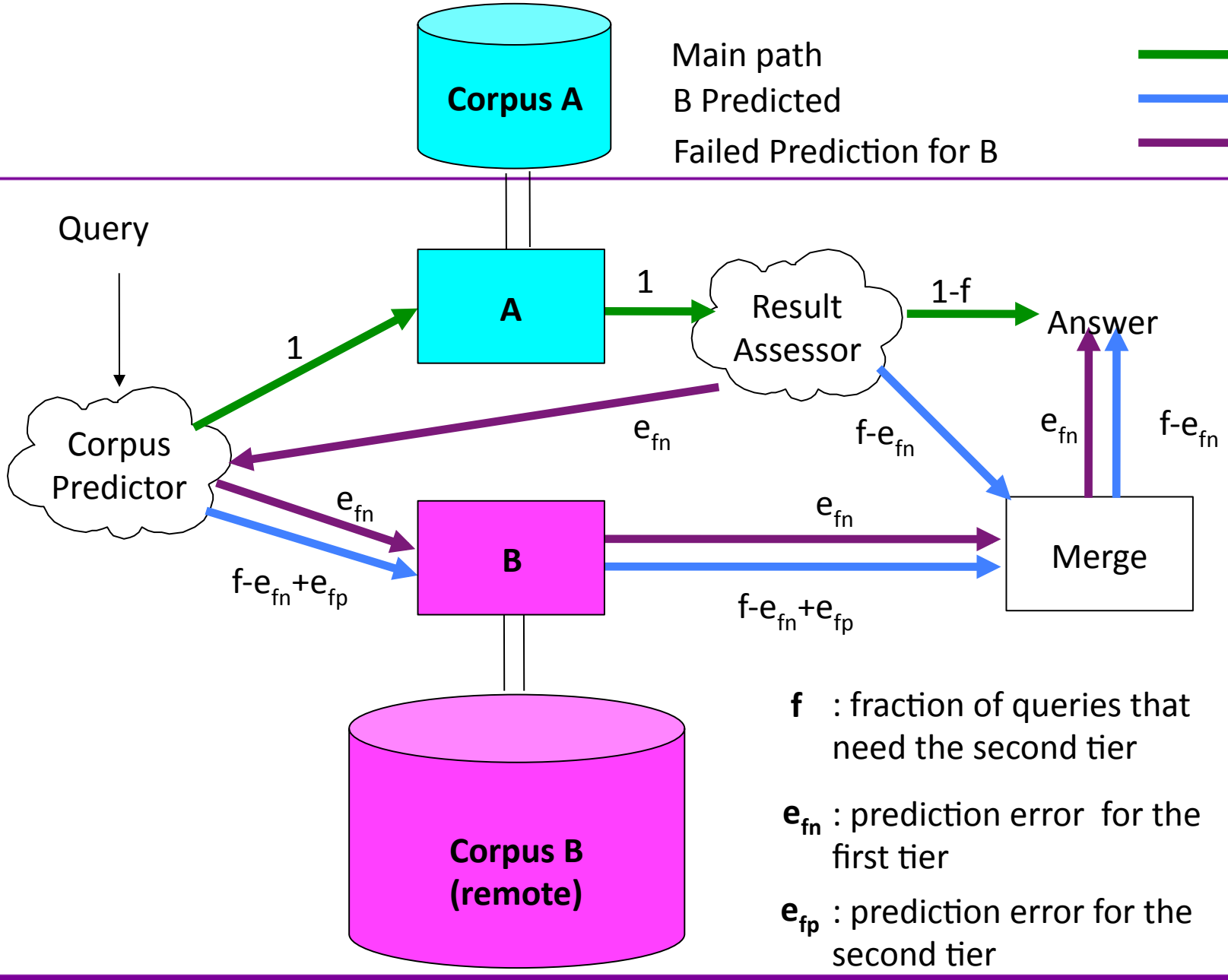
- Many queries are local
  - The answer returns only local documents
  - The user clicks only on local documents
- Locality also helps in:
  - Latency of HTTP requests (queries, crawlers)
  - Personalizing answers and ads
- Can we decrease the cost of the search engine?
- Measure of quality: same answers as centralized SE

Tier Prediction (Baeza-Yates et al, SIGIR 2009)

- Can we predict if the query is local?
  - Without looking at results and
  - increasing the extra load in the next level
- This is also useful in centralized search engines
  - Multiple tiers divided by quality
- Experimental results for
  - WT10G and UK/Chile collections

Motivation: Centralized Systems

- Traditionally partitioned corpora searched in serial, say two tiers
  - Second tier searched when first tier results are unsatisfactory
  - First tier faster and often sufficient
  - If second tier required, system is less efficient
- Better: search both corpora in parallel
- Best: predict which corpora to search


Corpus A
Main path
B Predicted
Failed Prediction for B
Query
A
1
1
Result Assessor
1-f
Answer
Corpus Predictor
1
e_{fn}
$f-e_{fn}$
e_{fn}
$f-e_{fn}$
e_{fn}
B
e_{fn}
Merge
$f-e_{fn}+e_{fp}$
$f-e_{fn}+e_{fp}$
Corpus B (remote)
f : fraction of queries that need the second tier
$\mathbf{e_{fn}}$: prediction error for the first tier
$\mathbf{e_{fp}}$: prediction error for the second tier

Trade-off Analysis (Baeza-Yates et al., 2008)

$$\begin{aligned} T_P &= T_S - (f - e_{FN})t_A \\ &= T_{min} + e_{FN}\ t_A \end{aligned}$$

$$\Delta T = \frac{f - e_{FN}}{1 + f\ t_B/t_A} \quad \Delta C = \frac{e_{FP}}{f(1 + C_A/C_B)}$$

Is it worth it?

$$\frac{T_S}{T_P} > \frac{C_P}{C_S}$$

$$R_C = \frac{C_A}{C_B} \propto \frac{Size(A)}{Size(B)}\ \frac{t_B}{t_A} = \beta\ R_T$$

$$\beta > \frac{e_{FP}}{f - e_{FN}} \qquad e_{FN} < f - \frac{e_{FP}}{f + e_{FP}}$$

Experimental Results

- Centralized case:

	Random	Centralized
Classifier Accuracy	0.714 ±0.008	0.789±0.009
Precision	n/a	0.983±0.006
Recall	na	0.265±0.022

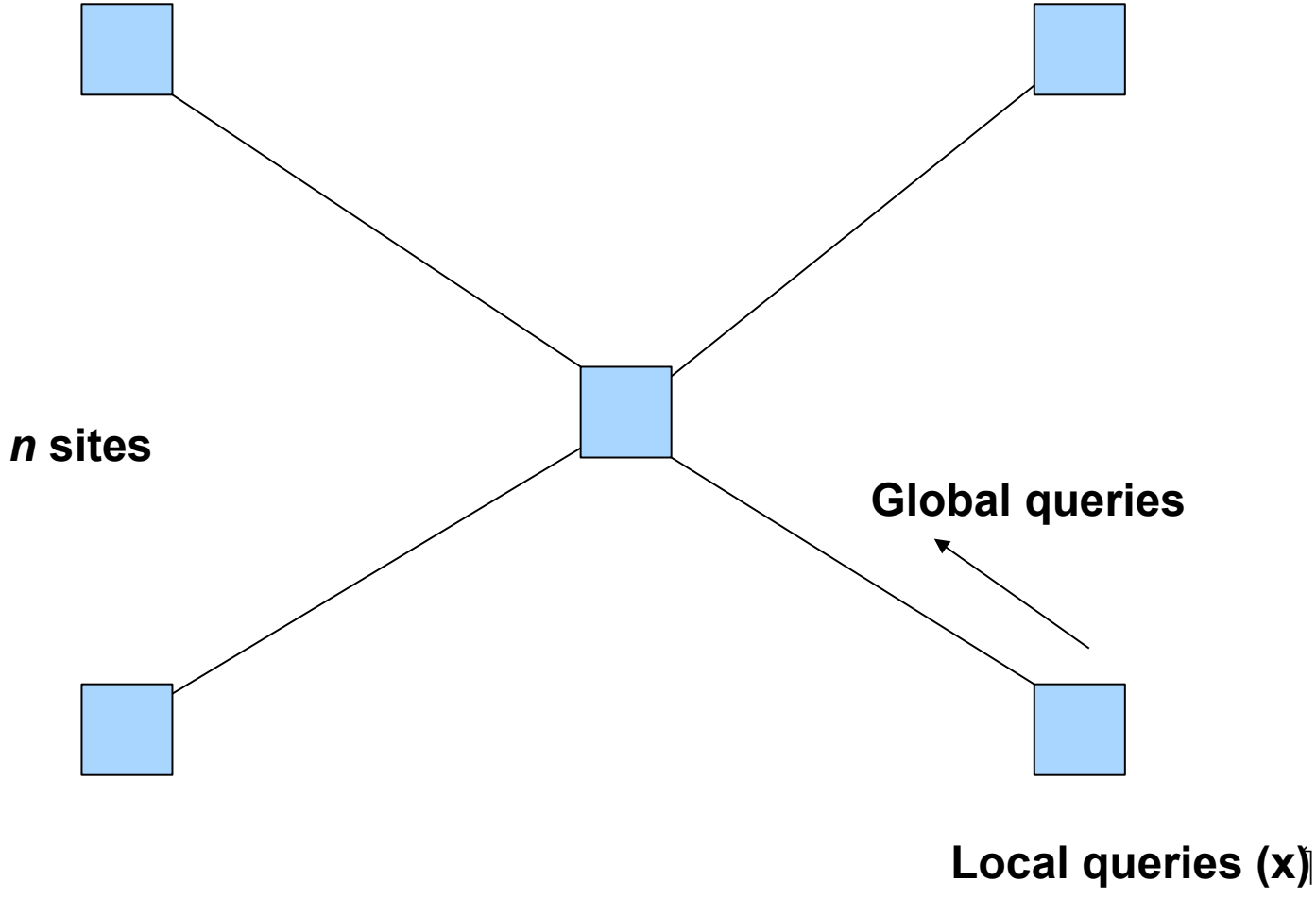
- Distributed case:

	Random	Distributed
Classifier Accuracy	0.539 ±0.006	0.776±0.006
Precision	n/a	0.675±0.006
Recall	n/a	0.991±0.003

Tier Prediction Example

- Example:
  - System A is twice faster than System B
  - System B costs twice the costs of System A
- Centralized case:
  - 29% faster answer time at 20% extra cost
- Distributed case:
  - 15% faster answer time at 0.5% extra cost
- In both cases the trade-off is worth it

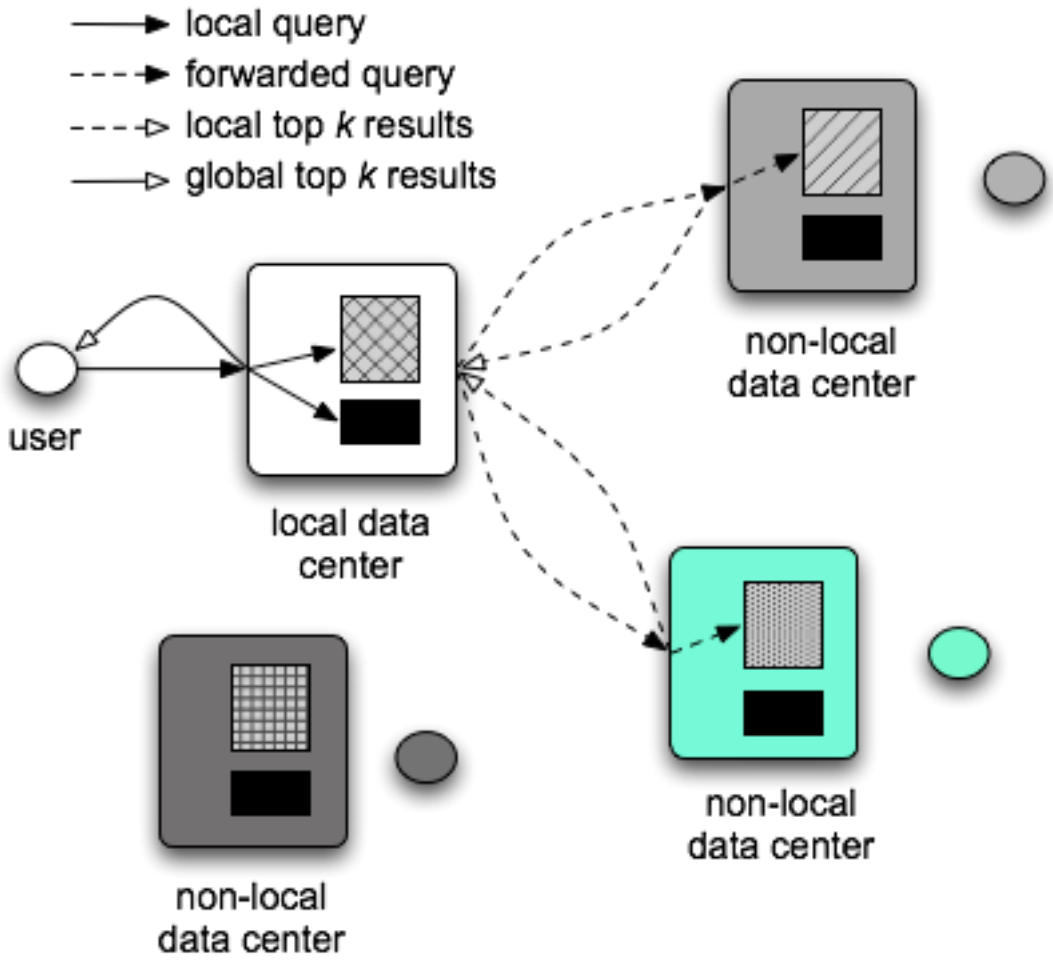
Star Topology (Baeza-Yates et al, CIKM 2009 Best paper award)



Multi-site Web Search Architecture

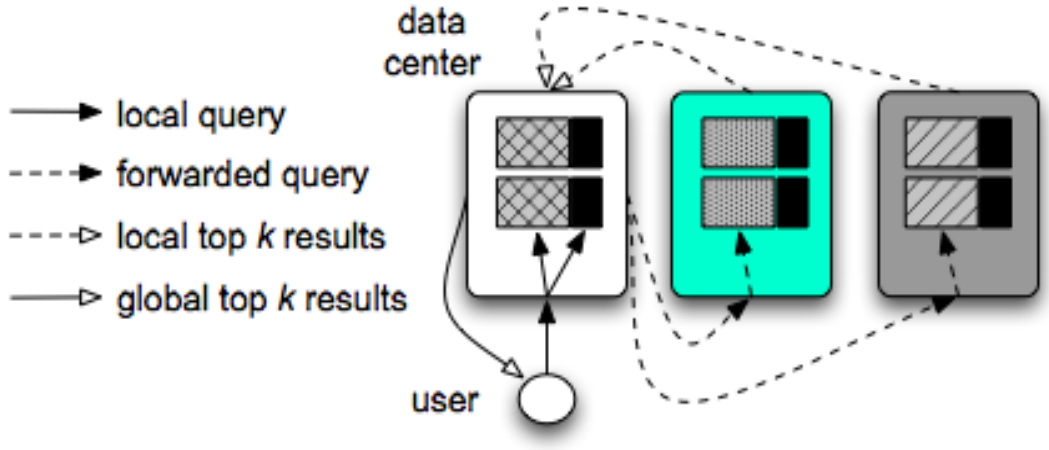
Key points

- multiple, regional data centers (sites)
- user-to-center assignment
- local web crawling
- partitioned web index
- partial document replication
- query processing with selective forwarding



A Search Engine Architecture with Partial Index Replication and Query Forwarding

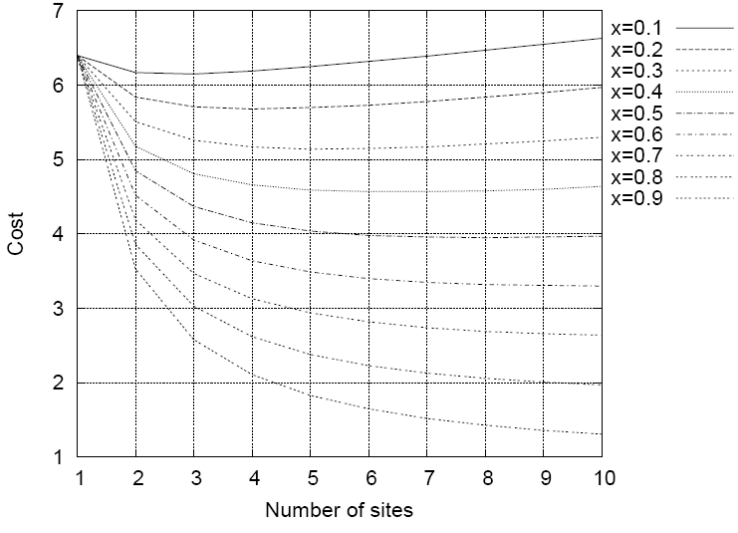
- **Features**
  - several data centers
  - users are assigned to local data centers
  - documents
    - partitioned
    - partially replicated
  - queries
    - locally processed
    - forwarded on-demand

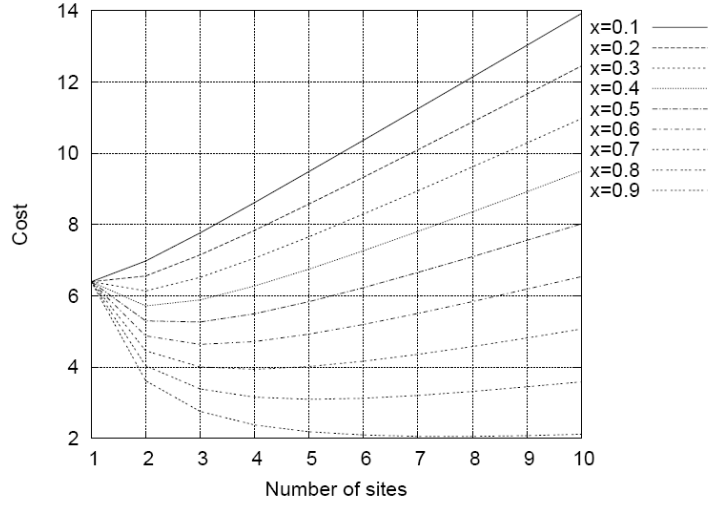


- **Parameters**
  - fraction of replicated index: β
  - fraction of queries forwarded: α
  - avg. # of sites a query is forwarded: γ
  - local queries are processed over an index of size: $I (1 - \beta) / S + \beta$
  - remote ($\gamma \alpha$) queries are processed over an index of size: $I (1 - \beta) / S$

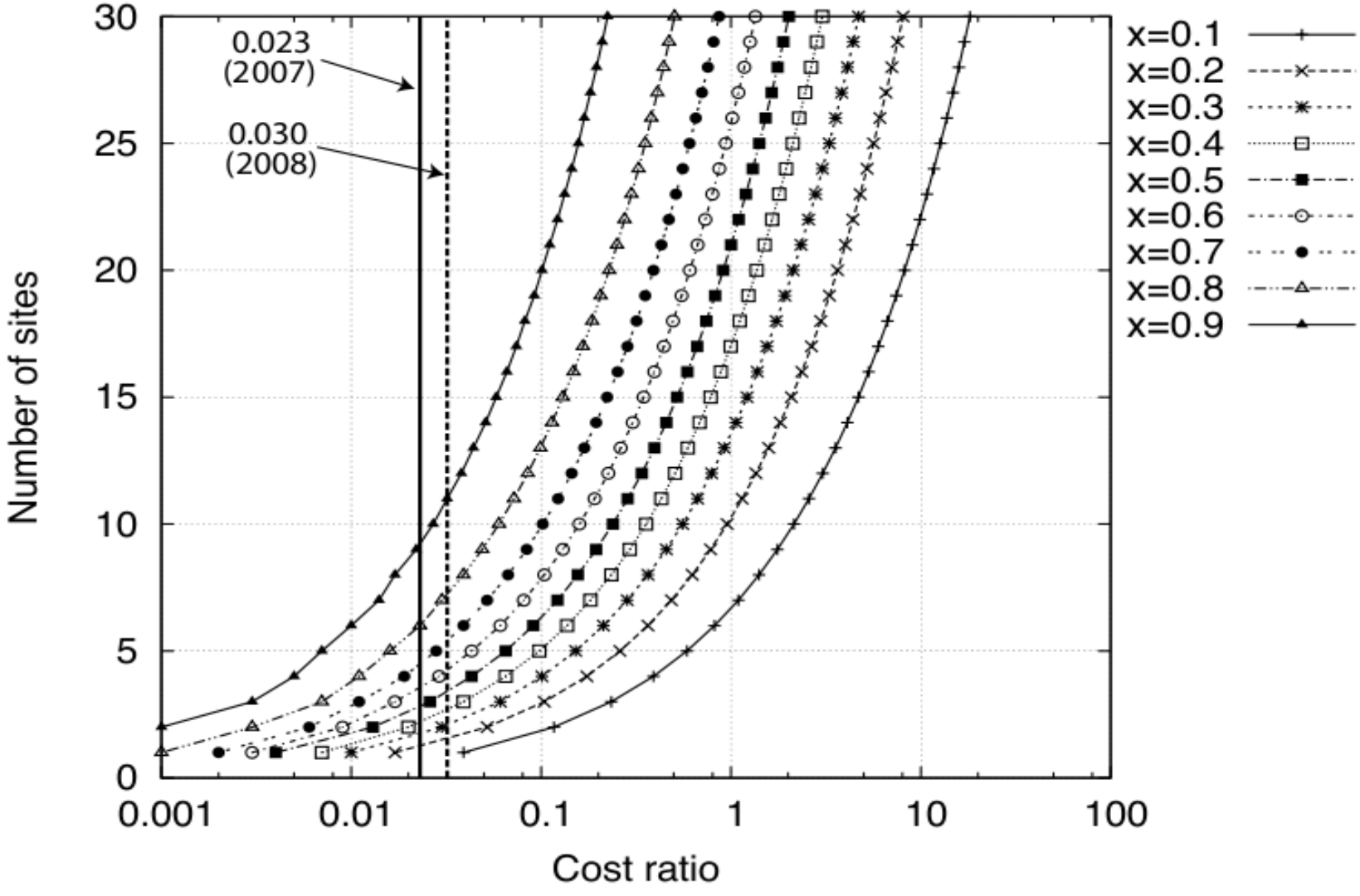
Cost Model

- Cost depends on **Initial cost**, **Cost of Ownership over time**, and **Bandwidth over time**.
- Cost of one QPS
  - *n* sites, *x* percentage of queries resolved locally, and relative cost of power and bandwidth 0.1 (left) and 1 (right)



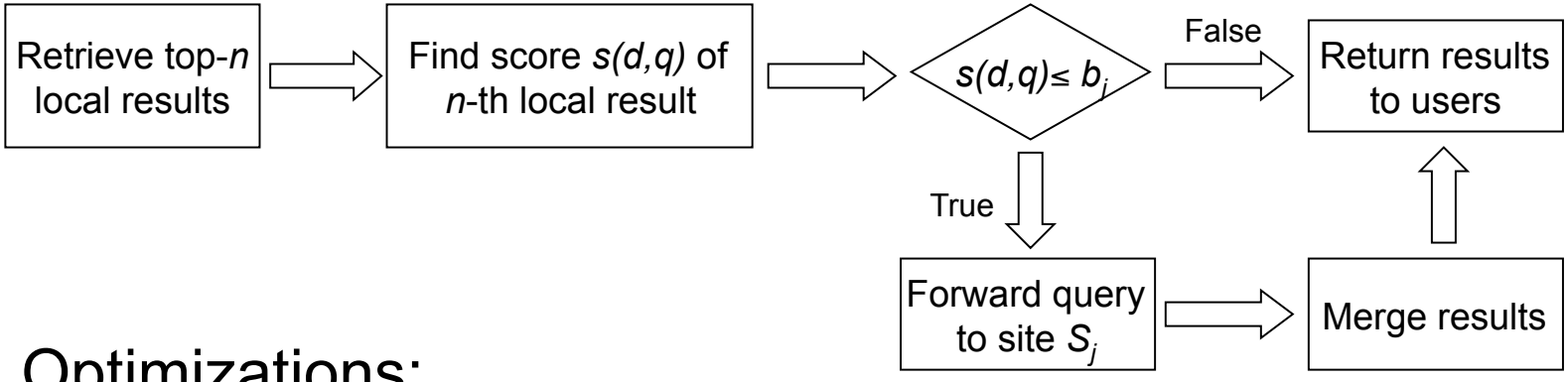


Optimal Number of Sites



Query Processing

- Site S_i knows the highest possible score b_j that site S_j can return for a query
  - Assume independent query terms
- Site S_i processes query q:

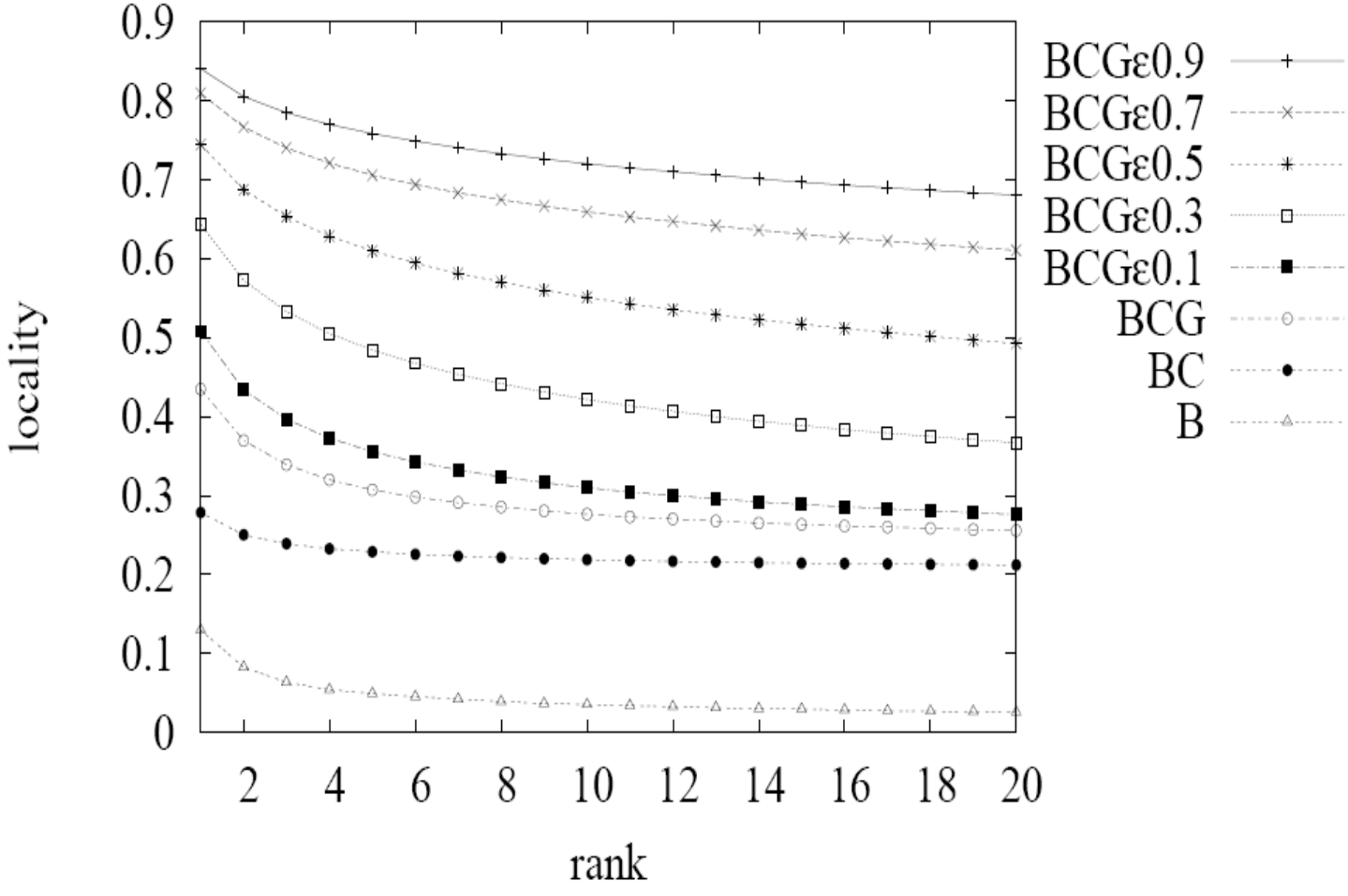


- Optimizations:
  - Caching
  - Replication of set G of most frequently retrieved documents
  - Slackness factor ε replacing b_j with $(1-\varepsilon)b_j$

Query Processing Results

- Locality at rank *n* for a search engine with 5 sites

- For what percentage of query volume, we can return top-*n* results locally



Cost Model Instantiation

- Assume a **5-site** distributed Web search engine in a **star topology**
- Optimal choice of central site S_x : site with **highest traffic** in our experiments
- Cost of distributed search engine relative to cost of centralized one

Query Processing	*Power Cost*	*Bandwidth Cost*	*Cost of distributed / Cost of centralized*
B	1.483	0.019	1.502
BC	1.278	0.016	1.294
BCG	1.156	0.013	1.169
BCG$\epsilon_{0.1}$	1.103	0.012	1.115
BCG$\epsilon_{0.3}$	0.970	0.010	**0.980**
BCG$\epsilon_{0.5}$	0.835	0.008	**0.843**
BCG$\epsilon_{0.7}$	0.719	0.006	**0.725**
BCG$\epsilon_{0.9}$	0.652	0.005	**0.657**

Improved Query Forwarding

(Cambazoglu et al, SIGIR 2010)

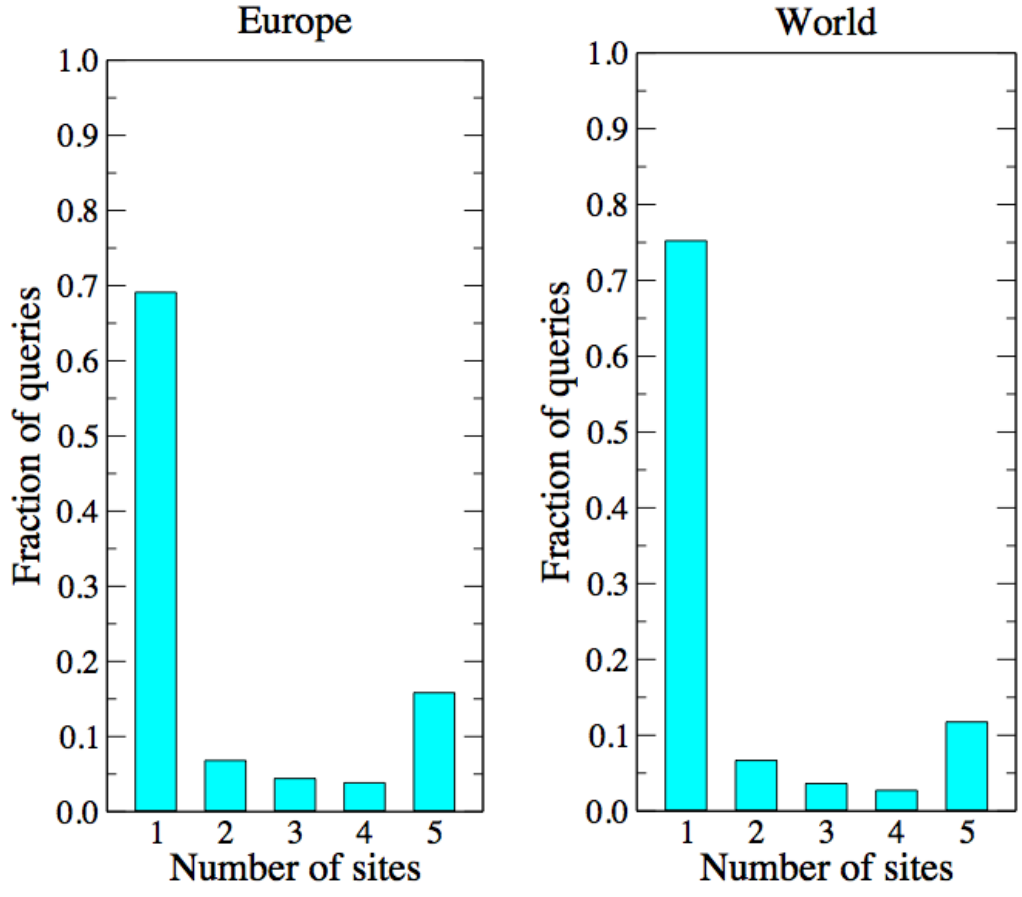
- Ranking algorithm
  - AND mode of query processing
  - the document score is computed simply summing query term weights (e.g., BM25)
- Query forwarding algorithm
  - a query should be forwarded to any site with potential to contribute at least one result to the global top *k*
  - we have the top scores for a set of off-line queries on all non-local sites
- Idea
  - set an upper bound on the possible top score of a query on non-local sites using the scores computed for off-line queries
  - decide whether a query should be forwarded to a site based on the comparison between the locally computed *k*-th score and the site's upper bound for the query

Experimental Setup

- Simulations via a very detailed simulator
- Data center locations
  - scenarios:
    - low latency (Europe): UK, Germany, France, Italy, Spain
    - high latency (World): Australia, Canada, Mexico, Germany, Brazil
  - assumed the data centers are located on capital cities
  - assumed that the queries are issued from the five largest city in the country
- Document collection
  - randomly sampled 200 million documents from a large Web crawl
  - a subset of them are assigned to a set of sites using a proprietary classifier
- Query log
  - consecutively sampled about 50 million queries from Yahoo! query logs
  - queries are assigned to sites according to the front-ends they are submitted to
  - first 3/4 of the queries is used for computing the thresholds; remaining 1/4 is used for evaluating performance

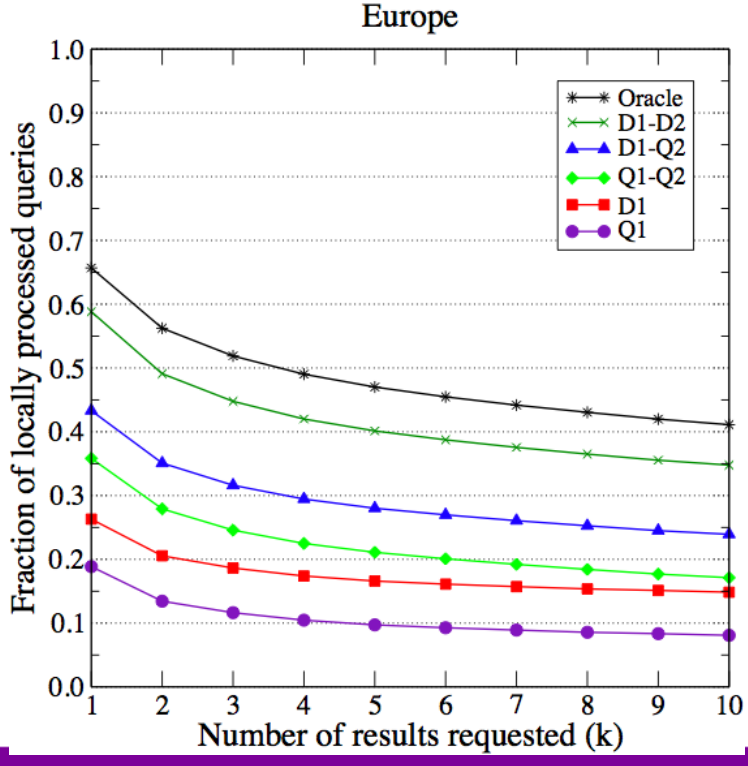
Locality of Queries

- Regional queries
  - most queries are regional
  - Europe: about 70% of queries appear on a single search site
  - World: about 75% of queries appear on a single search site
- Global queries
  - Europe: about 15% of queries appear on all five search sites
  - World: about 10% of queries appear on all five search sites

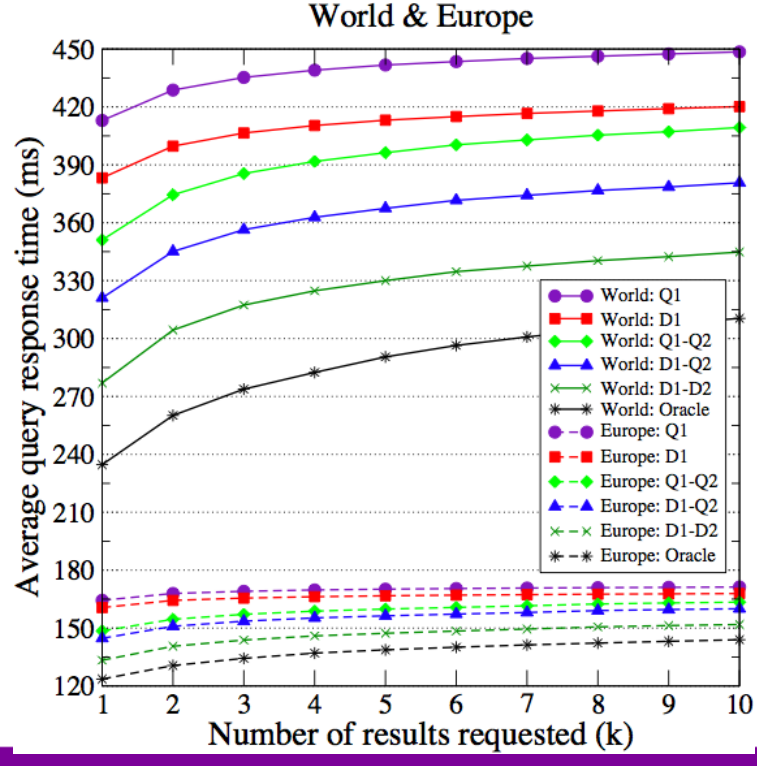


Performance of the Algorithm

- Local queries
  - about a quarter of queries can be processed locally (D1-Q2)
  - 10% increase over the baseline
  - oracle algorithm can achieve 40%

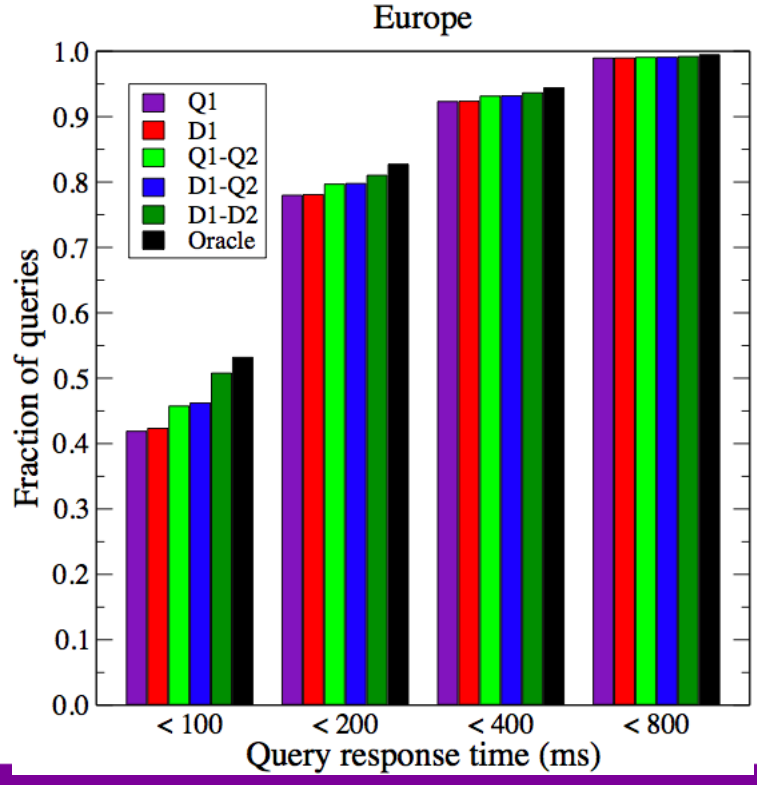


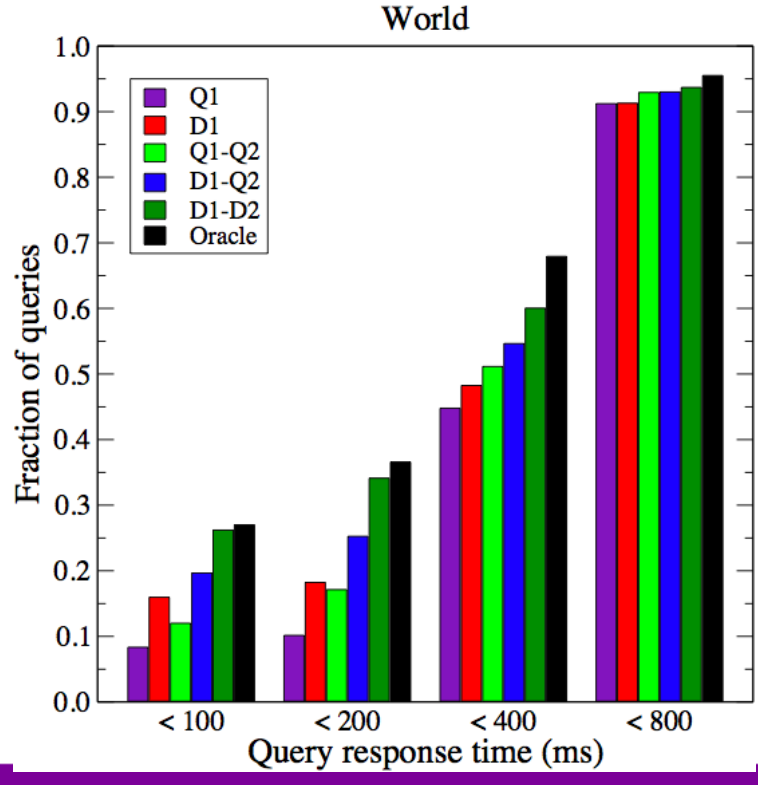
- Average query response times
  - Europe: between 120ms–180ms
  - World: between 240ms–450ms



Performance of the Algorithm

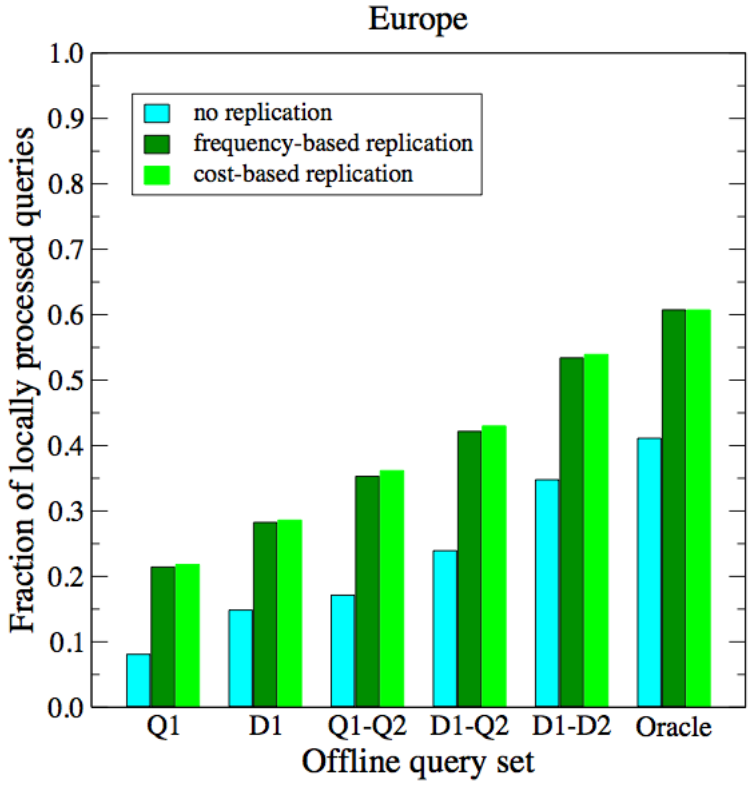
- Fraction of queries that are answered under a certain response time
  - Europe: around 95% under 400ms
  - World: between 45%–65% under 400ms

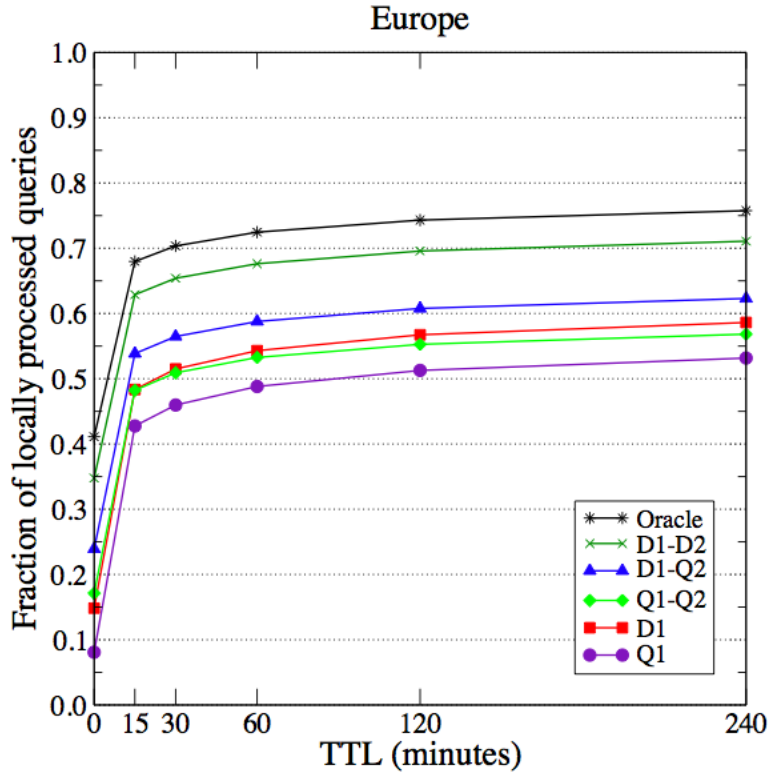




Partial Replication and Result Caching

- Replicate a small fraction of docs
  - prioritize by past access frequencies
  - prioritize by frequency/cost ratios
- Result cache
  - increase in local query rates: ~35%–45%
  - hit rates saturate quickly with increasing TTL

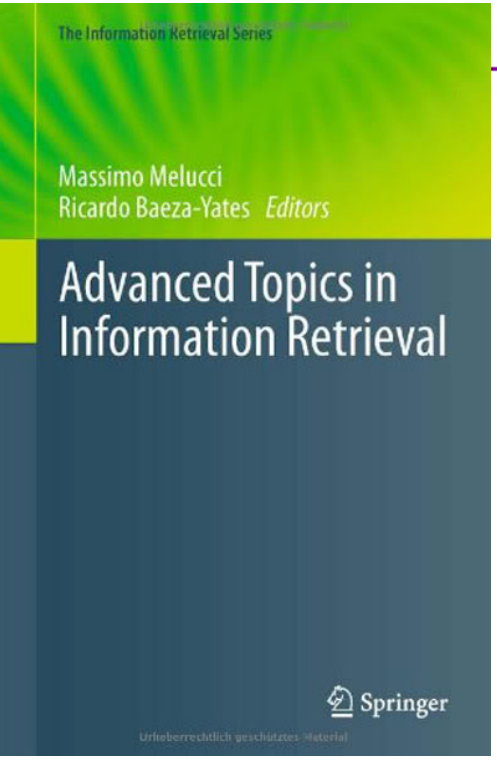




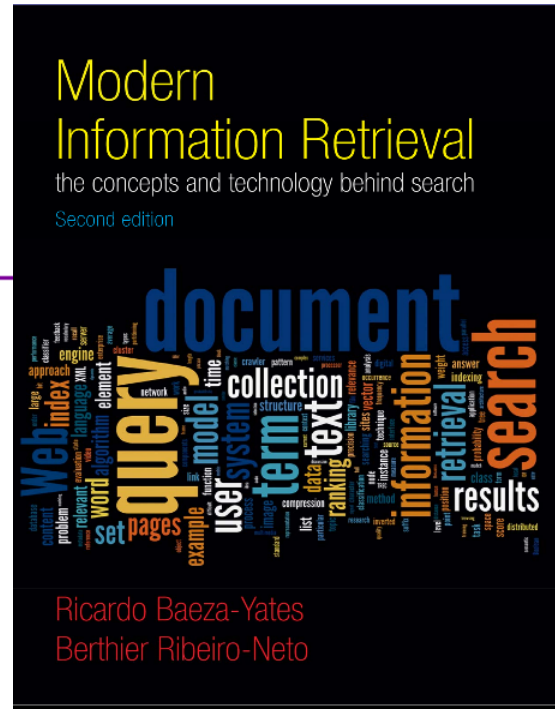
Conclusions

- By using caching (mainly static) we can increase locality and we can predict when not to cache
- With enough locality we may have a cheaper search engine without penalizing the quality of the results or the response time
- We can predict when the next distributed level will be used to improve the response time without increasing too much the cost of the search engine
- We are currently exploring all these trade-off's

Thank you!



Second edition appeared in 2011



Questions?

rbaeza@acm.org

SPIRE 2011, October, Pisa, Italy
WSDM 2012, February, Seattle, USA
ECIR 2012, April, Barcelona, Spain
ACM SIGIR 2012, July, Portland, USA