Processamento e Recuperação de Informação

Distributed Information Retrieval

Departamento de Engenharia Informática
Instituto Superior Técnico

1º Semestre
2018/2019

Outline

Bibliography

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, Modern Information Retrieval, 2nd edtion. Chapter 10.

Outline

Why Distributed IR?

- The volume of online content today is staggering and it has been growing at an exponential rate
- On at a slightly smaller scale, the largest corporate intranets now contain several million Web pages
- As document collections grow larger, they become more expensive to manage
- In this scenario, it is necessary to consider alternative IR architectures and algorithms
- The application of parallelism and distributed computing can greatly enhance the ability to scale IR algorithms

A Taxonomy of Distributed IR Systems

		Distributed	
	Non-dist.	One system	Various systems
Internal	standard	parallel	parallel
Local area		cluster-based	local federated
Broadband		distributed	federated

Outline

Document Partitioning

- Document partitioning slices the document-term matrix horizontally, dividing the documents among the subtasks
- The N documents in the collection are distributed across the P processors in the system
- During query processing, each parallel process evaluates the query on N/P documents
- The results from each of the sub-collections are combined into a final result list

Term Partitioning

- In term partitioning, the matrix is sliced vertically
  - It divides the indexing items among the P processors
- In this way, the evaluation procedure for each document is spread over multiple processors
- Other possible partition strategies include divisions by language or other intrinsic characteristics of the data
- It may be the case that each independent search server is focused on a particular subject area

Collection Partitioning

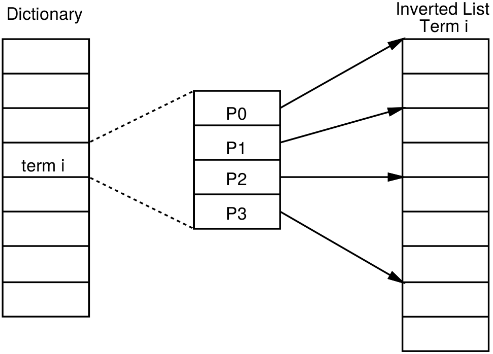
- Available when the distributed system is centrally administered
- A first option is the replication of the collection across all search servers
- The second option is random distribution of the documents
  - This is appropriate when a large document collection must be distributed for performance reasons
- The final option is explicit semantic partitioning of the documents
  - Here the documents are often already organized into semantically meaningful collections
- A broker routes queries to the search servers and balances the load on the servers

Inverted Index Partitioning

- Document partitioning
  - Logical
  - Physical
- Term partitioning

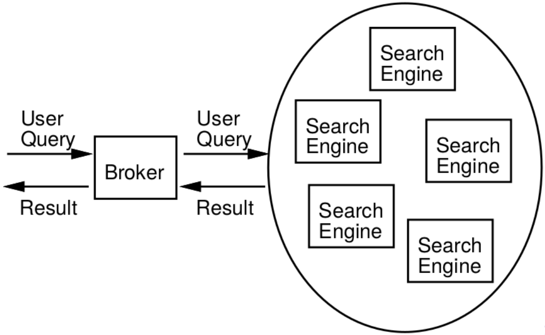
Logical Document Partitioning

- Uses the same inverted index as in the original algorithm
- Each dictionary entry includes P pointers into the corresponding inverted list
- The j-th pointer indexes the block of documents of the sub-collection of the j-th processor



Physical Document Partitioning

- Each sub-collection has its own inverted index and the processors share nothing during query evaluation
- Each processor evaluates the query on its portion of the document collection, producing a intermediate hit-list
- The P intermediate hit-lists can be merged efficiently using a binary heap-based priority queue



Physical Document Partitioning (cont.)

- Each process may require global term statistics in order to produce globally consistent document scores
- Two basic approaches:
  1. Compute global term statistics at indexing time and store these statistics with each of the sub-collections
  2. Process the queries in two phases:
     1. Term statistics from each of the processes are combined into global term statistics
     2. The broker distributes the query and global term statistics to the search processes

Comparison

- Logical document partitioning requires less communication than physical document partitioning
    - Thus, it is likely to provide better overall performance
- Physical document partitioning, on the other hand, offers more flexibility
    - E.g., document partitions may be searched individually
- The conversion of an existing IR system into a parallel system is simpler using physical document partitioning

Term Partitioning

- In term partitioning, the inverted lists are spread across the processors
- Each query is decomposed into items and each item is sent to the corresponding processor
- The processors create hit-lists with partial document scores and return them to the broker
- The broker then combines the hit-lists according

Issues in Term Partitioning

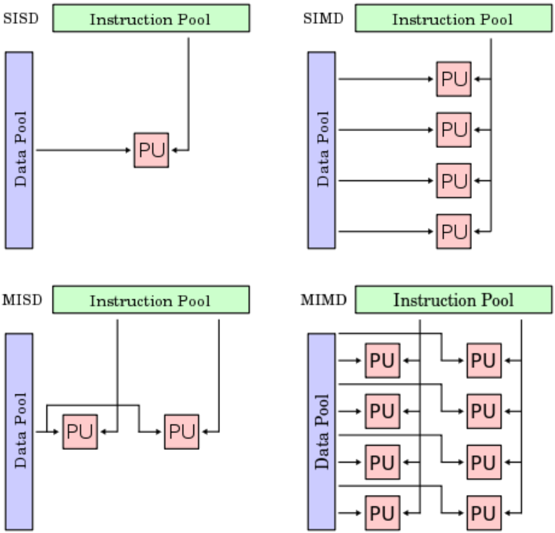
- The query load is not necessarily balanced, and thus part of the concurrency gains are lost
- Hence, the major goal is to partition the index such that:
  - The number of contacted processors/servers is minimal; and
  - Load is equally spread across all available processors/servers
- We can use query logs to split the index vocabulary among the processors to achieve the goal above

Overall Comparison

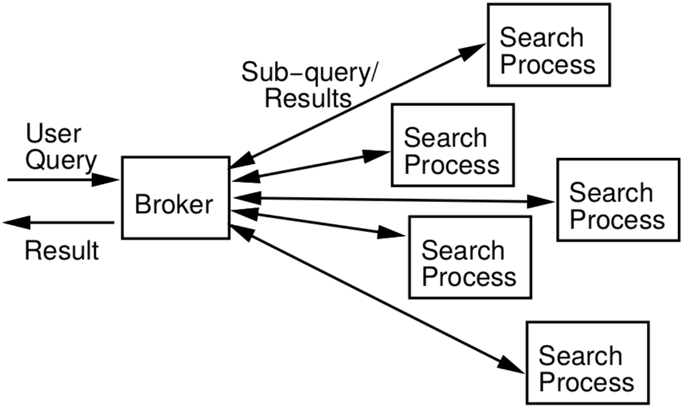
- Document partitioning affords simpler inverted index construction and maintenance than term partitioning
- Assuming each processor has its own I/O channel and disks, document partitioning performs better
- When terms are uniformly distributed in user queries, term partitioning performs better

Outline

Parallel Computing Architectures



Parallel IR on MIMD Architectures



Issues with Parallel IR on MIMD

- Care must be taken to properly balance the hardware resources on the system
- Search processes running on the different processors can perform I/O and compete for disk access
  - A bottleneck at the disk will be disastrous for performance and could eliminate the throughput gains
- In addition to adding more disks to the computer, the index data must be distributed over the disks
  - At one extreme, replicating the entire index on each disk eliminates disk contention at the cost of increased storage requirements and update complexity
  - Alternatively, heavily accessed data can be replicated and less frequently accessed data can be distributed
- Yet another approach is to install a disk array and let the operating system handle partitioning the index

Outline

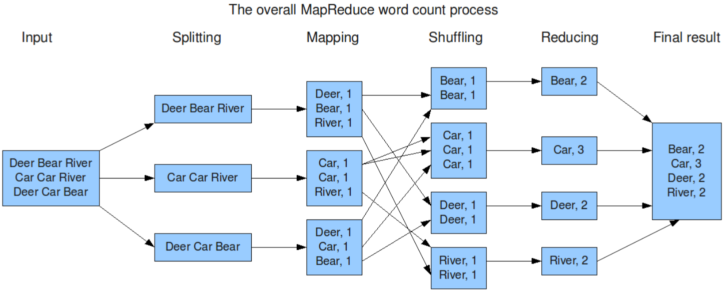
Cluster Computing

- A cluster of servers is a distributed system that has many computers, all physically close and usually connected through a fast local area network
  - As local networks become faster, a cluster presents behavior that resembles that of a parallel machine
- Load balancers are special nodes that balance the load among different machines

Cluster-based IR

- To program a cluster, there are middleware software such as:
  - MPI (Message Passing Interface)
  - PVM (Parallel Virtual Machine)
- Another possibility is the map-reduce parallel computing paradigm
- Current research is focused on extending the power of the map-reduce paradigm

Map-reduce


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Outline

Distributed Computing

- Distributed computing uses multiple computers connected by a network to solve a single problem
  - A distributed computing system can employ a heterogeneous collection of processors in the system
- The cost of inter-processor communication is considerably higher in a distributed computing system
- In distributed computing each processor has its own local memory

Key Issues

- Four main issues:
  - Partitioning deals with data scalability and, in a large IR system, implies partitioning the document collection and the index
  - Communication deals with processing scalability, which in our case is query processing
  - A system is dependable if its operation is free of failures
  - The external factors are the external constraints on the system
- Applications usually involve:
  - Computation and data that can be split into coarse grained operations, and
  - Relatively little communication is required between the operations
- Parallel information retrieval based on document partitioning fits this profile well

Modules of a distributed IR system

Module	Key Issues		
	Communication	Dependability (synchronization)	External factors
Indexing	Reindexing	Partial indexing Updating Merging	Content growth Content change Global statistics
Querying	Caching Replication	Personalization Rank aggregation	Changing user needs User base growth

Indexing

- One way to partition the index across the query processors is to consider the topics of the documents
  - Routing the queries according to their topic involves identifying the topics of both documents and queries
  - However, topic distribution might have a negative effect on the performance of the distributed retrieval system
- Partitioning the index according to the language of queries is also a suitable approach
  - A challenge in routing queries using language is the presence of multilingual documents such as in the Web
  - In addition, queries can be multilingual, involving terms in different languages
- So far, few papers suggest approaches to build an inverted index in a distributed fashion (comparatively)

Dependability

- If enough index servers fail, then the service as a whole also fails
- In some systems it is crucial to have the latest results for queries and content changes very often
  - In this case, it is important that the index data available at a given moment reflects all the changes in a timely fashion
- If a server of the system fails, it is impossible to recover the content of that server unless it is replicated
  - If this is not the case, then a possible inefficient way to recover is to rebuild the entire index
  - Another possibility would be to make the partitions partially overlapping
  - Document partitioned systems are more robust with respect to servers failures

Communication

- The distributed merge operations of the indexing process can impact the communication among servers
- Indexes are usually rebuilt from scratch after each update of the underlying document collection
- This update operation usually requires locking the index, jeopardizing the whole system performance
- Terms that require frequent updates might be spread across the servers, thus amplifying the lockout effect

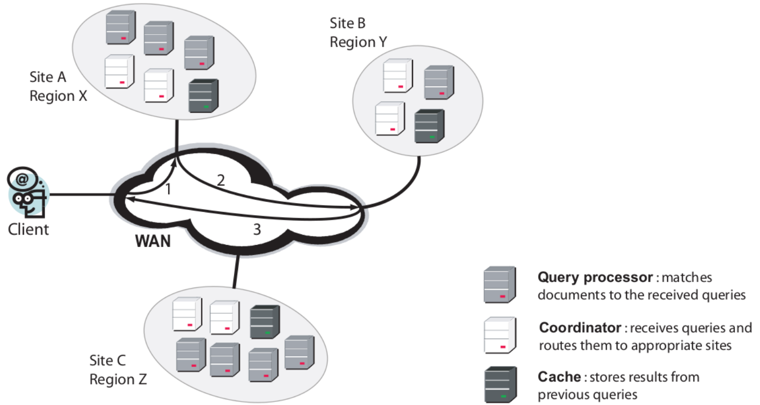
External Factors

- In a document partitioned IR system is necessary to compute values for some global parameters such as
  - the collection frequency, and
  - the inverse document frequency of a term
- There are two possible approaches:
  - One can compute the final global parameter by aggregating all the local statistics available after the indexing phase
  - The problem of computing global statistics can be moved to the system broker

Query Processing

- Resources to allocate to process a given query:
  - A coordinator makes decisions on how to route the queries to different parts of the system
  - The query processors hold index or document information
  - Cache servers can hold results for the most frequent or popular queries
    - They can reduce query latency and load on servers
- One or more servers implement each of these components
  - We can add more physical servers to increase the overall system capacity
- As these servers can be in different physical locations, we call site to each group of collocated servers

Query Processing Architecture



Dependability

- Due to the large amount of data they handle, it is challenging to determine good replication schemes
  - Having all query processors storing the same data, the system achieves the best availability level possible
  - This is likely to impose a significant overhead
- If a query processor is temporarily unavailable, we can serve cached results
- Consistency is also often a very important goal for online systems
  - There are techniques from distributed algorithms to implement fault-tolerant services

Communication

- If the index includes the position of terms, the communication overhead between servers increases
  - In such a case, the position information needs to be compressed efficiently
- In the case of a document partitioned system, query processors send the query results to the coordinator
  - The coordinator may become a bottleneck while merging the results from a large number of query processors
- Sometimes, the query processing involves adaptation of the search results according to the interests of the user
  - Each user profile represents a state, which must be the latest state and be consistent across replicas

External Factors

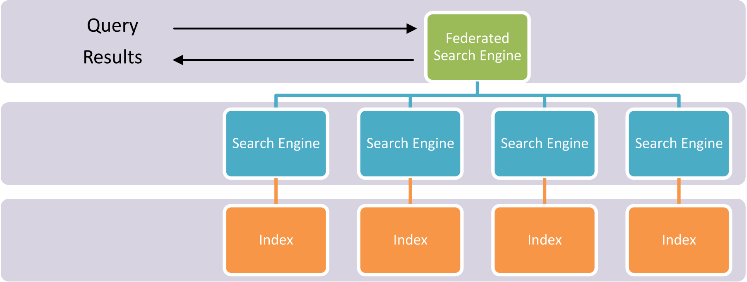
- User behavior is an external factor, which cannot be controlled by the IR system
    - For example, the topics the users search for have slowly changed in the past
    - Can also affect the performance of caching policies

Outline

1. Introduction
2. Data Partitioning
3. Parallel IR
4. Cluster-based IR
5. Challenges in Distributed IR
6. Federated Search
7. Retrieval in Peer-to-Peer Networks

Federated Search Engine

- A federated search system relies on a collection of heterogeneous servers to answer user queries



Engineering Issues

- defining the search protocol for transmitting requests and results,
  - obtain information about a search sever
  - submit a search request for one or more databases using a well defined query language
  - receive search results in a well defined format
  - retrieve items identified in the search results
  - Standards: Z39.50, STARTS, OpenSearch, ...
- designing a server that can efficiently accept a request and initiate a thread
- designing a broker that can submit asynchronous search requests to multiple servers and combine the results

Algorithmic Issues

- how to distribute documents across the distributed search servers
- how to select which servers should receive a particular query
- how to process the queries and combine the results from the different servers

How to merge the results?

- The simplest approach is to combine the ranked hit-lists using round robin interleaving
- The most accurate technique for merging ranked hit-lists is to use accurate global term statistics
  - Not always available
- Use rank-merging techniques (we have seen those on a previous lesson)

Outline

Peer-to-Peer Networks

- A peer or node is an arbitrary computer which, when connected to the Internet, joins a peer-to-peer network, conforming a peer-to-peer (P2P) system
- IR algorithms can take advantage of resources distributed across Internet, in particular file sharing

Retrieval in Peer-to-Peer Networks (1)

- Peer-level document collection descriptions can be used to identify nodes that can process the query
  - These descriptions guide the peer-selection process and the document retrieval from the selected peers
  - Resources are ranked by their likelihood to return relevant documents and top-ranked resources are selected
- Document-level indexing approaches typically distribute the complete index in a structured P2P network
  - This approach faces significant scalability problems caused by the high traffic costs

Retrieval in Peer-to-Peer Networks (2)

- Top-k query processing has been employed to solve the problem of extensive bandwidth consumption
  - Terminate the processing of a query when the top- k results obtained so far are correct
- Use hybrid index partitioning
  - All peers are clustered in groups and the indexing technique employs term partitioning within the groups

Questions?