Parallel and Distributed Information Retrieval

Murad Kamalov
Outline

- Introduction
- Parallel and Distributed Information Retrieval
  - Query throughput
  - Query response time
- P2P Information Retrieval
  - Chord
- Conclusions
Background

- MIMD - Multiple Instruction stream Multiple Data stream
  - Inter-process communication using shared memory
- Distributed system - separate machines
  - Inter-process communication using TCP/IP
- Search Index
  - Structure to facilitate fast information retrieval

<table>
<thead>
<tr>
<th></th>
<th>Term 1</th>
<th>Term 2</th>
<th>Term 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 1</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Document 2</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
IR, Parallel and Distributed IR

- Information Retrieval (anno 1880)
  - Searching Documents
  - Searching Information within Documents
  - Searching metadata about Documents

- Parallel and Distributed IR
  - Improving query throughput
  - Improving query response time
Improving query throughput

- Broker == Load Balancer
- **Search Engines** either share search index or have their own copy of one
- Applicable on both MIMD architectures and distributed systems
Improving Query response time

- Broker
  - Splits query to sub-queries and sends them to multiple search engines
  - Aggregates search results
Improving Query response time (2)

- Two ways to improve query response time
- Documents: \{D1, D2, D3\}

**Case 1:** execute query on different documents

\[
\text{result} = R_1 \cup R_2 \cup R_3
\]

**Case 2:** split query

\[
\text{query} = Q_1 \cup Q_2 \cup Q_3 \quad \text{result} = R_1 \cup R_2 \cup R_3
\]
Case 1: Example

- D1 = **Happiness** is a feeling characterised by stupidness, weird, well being, or **joy**
- D2 = **Happiness** is feeling of **joy**.
- D3 = The pursuit of **Happiness**
- QUERY = "happiness joy"
Case 2: Example

- D1 = *Happiness* is a feeling characterised by stupidness, weird, well being, or *joy*.
- D2 = *Happiness* is a feeling of *joy*.
- D3 = The pursuit of *Happiness*.
- QUERY = "happiness joy"

RESULT = \{D1, D2, D3\} ∪ \{D1, D2\} = \{D1, D2\}
Query Processing

- Select collections to search
- Distribute query to selected collections
- Evaluate query at distributed collections in parallel
- Combine results from distributed collections into final result

How to partition documents into collections?
How to select collections to search?
Collection Partitioning

- Replicate collections across all search servers
  - Search Engine {D1,D2,D3,D4,D5}
  - Search Engine {D1,D2,D3,D4,D5}
  - Search Engine {D1,D2,D3,D4,D5}

- Random distribution
  - Search Engine {D1,D5}
  - Search Engine {D3}
  - Search Engine {D2,D4}

- Semantic distribution
  - Subject specific
    - Search Engine {D1,D2} - Information Retrieval
    - Search Engine {D3,D4} - Cryptology
  - Alphabetical
    - Search Engine {D5} - Internet of Things
Source Selection

• Specifies, how broker selects search engines to which query will be sent

• All collections equally likely to contain document
  o random partitioning
  o significant semantic overlap

• Collection ranking
  o semantic distribution
P2P Information Retrieval

- Network consists out of peers.
- Documents are distributed among the peers

Chord
  - P2P protocol for retrieving documents from ring of equal peers
  - Fully decentralised
  - Widely used in DHT implementations
  - Ring represents ID space of length $2^m$
  - Each node has m-bit ID = SHA1(IP address)
  - Each document has m-bit ID (SHA1 based)
  - Each node maintains routing table with at most m entities
  - IDs of documents and nodes are in the same namespace
Chord: routing table example

$\begin{align*}
    & m = 6 \\
    & \text{ID space } = 2^6 = 64 \\
    & \text{routing entry } = N + 2^{i-1}, \text{ for } j = 1..m
\end{align*}$

- In our example $N = 8$
- Thus, we get following routing table

<table>
<thead>
<tr>
<th>$j$</th>
<th>Eq.</th>
<th>ID</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$8 + 2^0$</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>$8 + 2^1$</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>$8 + 2^2$</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>$8 + 2^3$</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>$8 + 2^4$</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>$8 + 2^5$</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>
Chord: Document lookup example

- Send request for document 45, to node 0
- Each node is responsible for documents which ID's are smaller/equal to his own ID and bigger than ID of previous node
- E.g. Node 48 is responsible for documents in range 41..48
Conclusions

- IR is large area with many subareas
- Solutions based on Parallel and Distributed IR are around for a while
  - We reviewed some of them
- Recently a lot of attention was paid to P2P technologies for Information Retrieval
  - Cheaper cost of deployment
References

- Modern Information Retrieval, Chapter 9, Parallel and Distributed IR, book by Ricardo Baeza-Yates and Berthier Ribeiro-Neto
- Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. Ion Stoica, Robert Morris
Thank You!