MINERVA Infinity:
A Scalable Efficient Peer-to-Peer Search Engine

Sebastian Michel
Max-Planck-Institut für Informatik
Saarbrücken, Germany
smichel@mpi-inf.mpg.de

Peter Triantafillou
University of Patras
Rio, Greece
peter@ceid.upatras.gr

Gerhard Weikum
Max-Planck-Institut für Informatik
Saarbrücken, Germany
weikum@mpi-inf.mpg.de
Vision

• Today: Web Search is dominated by centralized engines (“to google”)
  - censorship?
  - single point of attack/abuse
  - coverage of the web?

• Ultimate goal: “Distributed Google” to break information monopolies

• P2P approach best suitable
  – large number of peers
  – exploit mostly idle resources
  – intellectual input of user community
Challenges

- large scale networks
  - 100,000 to 10,000,000 users
- large collections
  - > 10^{10} documents
  - 1,000,000 terms
- high dynamics
Questions

• Network Organization
  – structured?
  – hierarchical?
  – unstructured?

• Data Placement
  – move data around?
  – data remains at the owner?

• Scalability?

• Query Routing/Execution
  – Routing indexes?
  – Message flooding?
Overview

• Motivation (Vision/Challenges/Questions)
• Introduction to IR and P2P Systems
• P2P- IR
• Minerva Infinity
• Network Organization
• Data Placement
• Query Processing
• Data Replication
• Experiments
• Conclusion
Information Retrieval Basics

Document

Terms

# of terms (term frequency)

5 x
7 x
4 x
**Top-k Query Processing:** find $k$ documents with the highest total score

**Query Execution:** Usually using some kind of threshold algorithm*:
- sequential scans over the index lists (round-robin)
- (random accesses to fetch missing scores)
- aggregate scores
- stop when the threshold is reached

E.g. Fagin’s algorithm
TA or a variant without random accesses

* B+ tree on terms

Index lists with $(DocId: tf*idf)$ sorted by Score
P2P Systems

• Peer:
  – “one that is of equal standing with another”
    (source: Merriam-Webster Online Dictionary)

• Benefits:
  – no single point of failure
  – resource/data sharing

• Problems/Challenges:
  – authority/trust/incentives
  – high dynamics
  – ...

• Applications:
  – File Sharing
  – IP Telephony
  – Web Search
  – Digital Libraries
Structured P2P Systems based on Distributed Hash Tables (DHTs)

- “structured” P2P networks
- provide one simple method:

  \[ \text{lookup} : \text{key} \rightarrow \text{peer} \]

- CAN [SIGCOMM 2001]
- CHORD [SIGCOMM 2001]
- Pastry [Middleware 2001]
- P-Grid [CoopIS 2001]

robustness to load skew, failures, dynamics
Chord

- Peers and keys are mapped to the same cyclic ID space using a hash function.
- Key $k$ (e.g., $\text{hash(file name)}$) is assigned to the node with key $p$ (e.g., $\text{hash(IP address)}$) such that $k \leq p$ and there is no node $p'$ with $k \leq p'$ and $p' < p$. 
Chord (2)

- Using **finger tables** to speed up lookup process
- Store pointers to few distant peers
- Lookup in $O(\log n)$ steps
Overview

• Motivation (Vision/Challenges/Questions)
• Introduction to IR and P2P Systems
• P2P- IR
• Minerva Infinity
• Network Organization
• Data Placement
• Query Processing
• Data Replication
• Experiments
• Conclusion
P2P - IR

• Share documents (e.g. Web pages) in an efficient and scalable way
• Ranked retrieval
  – simple DHT is insufficient
Possible Approaches

- Each peer is responsible for storing the COMPLETE index list for a subset of terms.

Query Routing: DHT lookups
Query Execution: Distributed Top-k

[TPUT '04, KLEE '05]

capacity overload of peers with highly frequent / popular terms (data load AND query load)
Possible Approaches (2)

- Each peer has its own local index (e.g., created by web crawls)

**Query Routing:**
1. DHT lookups
2. Retrieve Metadata
3. Find most promising peers

**Query Execution:**
- Send the complete Query and merge the incoming results

capacity overload of peers with:
- highly frequent terms
- high-quality collections
Overview

• Motivation (Vision/Challenges/Questions)
• Introduction to IR and P2P Systems
• P2P- IR
• Minerva Infinity
• Network Organization
• Data Placement
• Query Processing
• Data Replication
• Experiments
• Conclusion
Minerva Infinity

• Idea:
  – assign (term, docId, score) triplets to the peers
    • order preserving
    • load balancing
  – hash(score)+
    hash(term) as offset
  – guarantee 100% recall
Hash Function

• Requirements:
  – Load balancing (to avoid overloading peers)
  – Order preserving (to make the QP work)

• One without the other is trivial ...
  – Load balancing: apply a pseudo random hash function
  – Order preserving:
    \[
    \frac{S - S_{\text{min}}}{S_{\text{max}} - S_{\text{min}}} \times N
    \]

• Both together is challenging ...
Hash Function (2)

- Assume an exponential score distribution
- Place the first half of the data to the first peer
- The next quarter to the next peer
- and so on …
Term Index Networks (TINs)

- Reduce # of hops during QP by reducing the number of peers that maintain the index list for a particular term

→ Only a **small** subset of peers is used to store an index list.
How to Create/Find a TIN

• Use $u$ Beacon-Peers to bootstrap the TIN for term $T$

$$p = \frac{1}{u}$$

For $i=0$ to $i<n$ do

- $id = \text{hash}(t, i\times p)$
- if $(i>0)$ use $\text{hash}(t,(i-1)\times p)$ as a gateway to the TIN
- else node with $id$ creates the TIN

End for

Beacon nodes act as gateways to the TIN
Publish Data / Join a TIN

- Peer with id = hash(t, score) not in the TIN for term t
- Randomly select a **beacon** node
  (Beacon nodes act as gateways to the TIN)
- Call the join method
- Store the item (docId, t, score)
Query Processing

Data Peers

Coordinator

2-keyword Query

Alternative: Collect data and send in one batch.
QP with Moving Coordinator

3-keyword Query

MINERVA Infinity: A Scalable Efficient P2P Search Engine
Data Replication

- **Vertical:** Replicate data inside a TIN via a ‘reverse’ communication.

- **Horizontal:** Replicate complete TINs
Experiments

Test bed:
10,000 peers

Benchmarks:

- **GOV**: TREC.GOV collection + 50 TREC-2003 Web queries, e.g. *juvenile delinquency *

- **XGOV**: TREC.GOV collection + 50 manually expanded queries, e.g. *juvenile delinquency youth minor crime law jurisdiction offense prevention *

- **SCALABILITY**: One query executed multiple times

........
Experiments: Metrics

Metrics

• Network traffic (in KB)

• Query response time (in s)
  - network cost (150ms RTT, 800Kb/s data transfer rate)
  - local I/O cost (8ms rotation latency + 8MB/s transfer delay)
  - processing cost

• Number of Hops
Scalability Experiment

• Measure time for a different query loads.
  – identical queries
  – inserted into a queue

![Graph showing the relationship between query load and total execution time for Minerva Infinity and no parallel processing.](chart.png)
Experiments: Results

MINERVA Infinity: A Scalable Efficient P2P Search Engine
Conclusion

• Novel architecture for P2P web search.
• High level of distribution both in data and processing.
• Novel algorithms to create the networks, place data, and execute queries.
• Support of two different data replication strategies.
Future Work

- Support of different score distributions
- Adapt TIN sizes to the actual load
- Different top-k query processing algorithms
Thank you for your attention