A Reproducible Benchmark for P2P Retrieval

Thomas Neumann Matthias Bender Sebastian Michel Gerhard Weikum
Max-Planck-Institut für Informatik
66123 Saarbrücken
GERMANY
{neumann,mbender,smichel,weikum}@mpi-inf.mpg.de

ABSTRACT

With the growing popularity of information retrieval (IR) in distributed systems and in particular P2P Web search, a huge number of protocols and prototypes have been introduced in the literature. However, nearly every paper considers a different benchmark for its experimental evaluation, rendering their mutual comparison and the quantification of performance improvements an impossible task.

We present a standardized, general purpose benchmark for P2P IR systems that finally makes this possible. We start by presenting a detailed requirement analysis for such a standardized benchmark framework that allows for reproducible and comparable experimental setups without sacrificing flexibility to suit different system models. We further suggest Wikipedia as a publicly-available and all-purpose document corpus and finally introduce a simple but yet flexible clustering strategy that assigns the Wikipedia articles as documents to an arbitrary number of peers. After proposing a standardized, real-world query set as the benchmark workload, we review the metrics to evaluate the benchmark results and present an example benchmark run for our fully-implemented P2P Web search prototype MINERVA.

1. INTRODUCTION

Motivation

Peer-to-Peer (P2P) systems have been a hot research topic for several years. Receiving public attention mainly in the context of file-sharing applications, such as Napster or Gnutella, the P2P paradigm is rapidly making its way into a number of serious (and more legal) application classes, including numerous aspects of data management, due to its prospect to concurrently utilize resources (e.g., computational power, bandwidth offerings, or storage supplies) by a huge - typically a priori unlimited - number of nodes. This sparks great expectations with regard to scalability, efficiency, and resilience to failures and dynamics. For example, applications like SETI@Home [33] or the Screensaver-Lifesaver Project [29] use the parallel computing power to watch out for extra-terrestrial life or interactions between molecules and cancer-causing targets, respectively, and ask you to "donate" your computer's idle time for a good cause. BitTorrent [5] is a P2P file distribution application that can distribute large amounts of data without incurring the the corresponding consumption in costly server and bandwidth resources. Distributed data management systems distribute the functionality of regular data management system over many sites, connected by a computer network.

A particularly intriguing class of distributed applications are information retrieval (IR) systems. An unmanageable number of systems and protocols have been designed to efficiently and effectively distribute the task of not only sharing, but *finding* information in huge data corpora, either by deliberately trying to distribute the mode of operation of centralized systems (most prominently top-k style search algorithms [9, 27]), or by entering virgin soil and designing completely novel ways to obtain all relevant information in the network [24, 1, 19, 30, 4, 10, 34, 37]. The latter applications, in addition to the potential performance and scalability advantages named before, typically also want to benefit from the intellectual input of a large user community, as a human user is the driving force behind every node in the network.

While comparing the performance (both concerning result quality and response time) of IR systems is already a challenging task for centralized systems, for distributed IR systems it is even more difficult to reproduce the experimental results being published. There exist a small number of widely deployed benchmarks for centralized IR systems consisting of both a document corpus and accompanying queries, such as a 1.2 million document collection obtained from a web crawl of the .gov internet domain, used by the Text Retrieval Conference TREC [35] for its yearly competition of IR systems. This collection has, also outside the actual conference, become a de-facto standard for centralized IR system benchmarking; it makes different systems comparable and allows for a quantification improvements. However, it is not freely available.

For distributed IR, however, no such de-facto standard has evolved so far. Some researchers in the field of P2P IR have tried to adapt the .gov benchmark to a distributed setting; however, as no method to assign the documents to peers has been standardized, a large number of often either ad-hoc or non-reproducible assignments have been used to evaluate P2P IR systems. The large variety of different assignment approaches renders the mutual comparison and the

EXPDB 2006 Chicago, Illinois, US.
Proceedings of the First International Workshop on Performance and Evaluation of Data Management Systems (EXPDB 2006), June 30, 2006, Chicago, Illinois, USA

quantification of improvements an impossible task.

Contribution

We suggest a standardized benchmark framework for P2P IR systems, based on the Wikipedia corpus, which is freely available for download and easy to process. We introduce a simple, reproducible, but yet powerful and flexible clustering method that assigns documents to a variable number of peers. We further suggest a number of real-world Web queries, taken from Google's *Zeitgeist* archive [2], that adequately model the behavior of human users. Finally, we introduce a standardized cost model that allows for a computational model to compare and to quantify the efficiency and the scalability of approaches.

The remainder of this paper is organized as follows. After reviewing related work in Section 2, we review the requirements for a P2P IR benchmark in more detail in Section 3, focusing on a suitable data corpus, a meaningful and flexible strategy to distribute the data across peers, and desired characteristics for benchmark queries. Section 4 discusses our suggested method for assigning documents to peers in detail. Section 5 revisits the necessary metrics to evaluate a P2P IR system based on our benchmark and introduces a simplified cost model. A benchmark example for our P2P Web Search engine MINERVA is shown in Section 6, before Section 7 concludes this work and points at future research directions.

2. RELATED WORK

2.1 Benchmarks for Centralized IR systems

For centralized IR systems, there exist a small number of de-facto standard benchmarks, such as the yearly competitions conducted by the Text Retrieval conference TREC [35]. However, the benchmark is only available to registered participants of the conference. Other benchmarks repeatedly deployed in the literature include the INEX benchmark [20] for centralized XML retrieval, or retrieval on the DBLP corpus of scientific publications [12].

2.2 Benchmarks for Distributed Systems

A number of benchmarks for distributed systems has been proposed (e.g., [11]), but they mostly focus on the behavior of the underlying P2P architecture as an overlay network architecture. However, they fall short of providing an application scenario and, thus, a benchmark of P2P information retrieval approaches. Similarly, the mobile networking community has invested some effort into standardizing benchmarks, that, however, also mostly center around aspects of the network layer rather than the application layer (e.g., [14]).

Being influenced by information retrieval in centralized systems, a substantial fraction of authors in the field of distributed IR evaluate their approaches by partitioning well known centralized IR benchmark collections, such as the one provided by TREC, into (overlapping or disjoint) fragments. However, the assignment of documents to peers is not standardized and performed differently by the authors, rendering the comparison of experimental results a bothersome task.

The only community of distributed retrieval, to our knowledge, that has so far seen substantial efforts into standardized application-level benchmarks is the distributed image retrieval community (e.g., [16]).

2.3 Distributed IR

In the context of distributed information retrieval there exist a number of techniques, e.g., CORI [8], GlOSS [15], or the decision-theoretic framework [31], that deal with the problem of collection selection, i.e. the task of finding appropriate collections for a particular query. These strategies are usually based on statistics that reflect the wealth and quality of a peer's corpus. Recently proposed methods [18, 3, 26] consider the overlap between collections while searching for the most promising peers.

A second challenge in distributed IR is query result merging, i.e., the proper consolidation of query results obtained from autonomous sources (with possibly incompatible scores) into one combined result list. For cooperative environments, i.e. scores and documents statistics are provided by peers, one popular approach is the algorithm proposed by Kirsch [21]. A series of publications [23] describes a CORI merging strategy, which heuristically combines both resource and document scores.

2.4 P2P IR

In recent years, distributed information retrieval systems based on Peer-to-Peer (P2P) architectures are increasingly receiving attention [24, 1, 19, 30, 4, 10, 34]. These systems usually consider a collaboration of peers where each peer stores a subset of the globally available documents. The network organization differs from system to system but all consider the peers having arbitrarily created document collections. The setup concerning the information retrieval is somewhat similar to the setup in metasearch engines, but has additional aspects because of high network dynamics and the intended total number of available data sources.

2.5 (Distributed) Top-k Queries

There exist a number of approaches to efficiently compute top-k style queries over distributed index lists, i.e., that try to adapt the same techniques used in centralized IR systems to a distributed environment, e.g., TPUT [9] or KLEE [27]. They typically consider index lists for a certain attribute to be stored at (exactly) one site (peer). These algorithms usually do not rely on a certain assignment from *documents* to peers but only on complete *index lists* for the particular terms involved in a query.

3. REQUIREMENT ANALYSIS

In order to make the experimental results obtained by different prototype P2P IR systems comparable or in order to quantify the improvements realized by additional add-ons to existing systems, the benchmark setup has to be *comparable*. In order to verify the results obtained by other research groups, the benchmark must be *reproducible*. This includes in particular, but is not limited to

- Document corpus: The document corpus must be uniquely identifiable and readily available to the public.
- Assignment of documents to peers: The method used to distribute the document corpus over the peers must be easily reproducible.
- Queries: The nature of the queries has a high impact on experimental results. For example, the number of terms in a query highly influences its execution cost.

- Metrics: Even if the benchmark framework used to conduct the experiments is comparable, the results are only comparable if the same metrics have been used to quantify the benchmark results, e.g., the underlying network cost model or the measure used to quantify result quality.

3.1 Data Corpus

Obviously, comparing the experimental results of different P2P IR systems is only possible if the experiments were conducted using the same document corpus. While there exist a number of standardized document corpora for centralized IR systems, they have so far fallen short of being standardized corpora for P2P IR retrieval experiments, because they lack a central detail: designed for centralized IR retrieval, they do not come with an assignment of documents to different peers. In fact, to pretend that they have reproducible and comparable results, many researchers in the field of P2P IR retrieval have nevertheless used these collections, assigning documents to peers in different fashions: randomly, by host, based on the link structure, classified by content, by popularity, or various other aspects. As these assignments were rarely made public (with some exceptions [7]), the results were hardly comparable or reproducible.

The document corpus used by the text retrieval conference TREC, which was obtained from a web crawl of the .gov internet domain, has some further shortcomings: it is not publicly available and its content does not reflect some general-purpose topics (e.g., contemporary music or movies), with a substantial fraction of documents being official publications with legal, historical, or scientific background.

With P2P systems becoming popular in the context of various file sharing applications, some researchers have also tried to use collections that were harvested from crawls of such networks, and typically consist of files names[1] and some annotational metadata, e.g. [28]. While providing a natural and real-world assignment from content to peers, such a corpus is, in our eyes, ill-suited for information retrieval. For example, considering a query for a particular song like "Madonna - American Music", there is no obvious ranking among different files of that song.

Instead, we suggest the publicly available download of the Wikipedia database as of Jan 25, 2006 [38] as a standard document corpus. It is a real-world corpus (not synthetically created), sufficiently large ($>$ 800,000 documents), it covers a virtually unlimited number of general-purpose topics and special interests, there are links between the articles (allowing for the adoption of link-analysis techniques, such as PageRank [6] or HITS [22]), and it comes in an XML format that can easily be parsed for further individual processing.

3.2 Data Placement

When evaluating the performance of distributed information retrieval systems, the data placement among the peers has a great impact on the resulting performance, both regarding query execution cost and result quality. For example, consider the following popular system design of peers sharing their personal indexes, where queries are locally executed on a judiciously chosen small subset of all peers and eventually merged by the query initiator (e.g., [4, 37]). The number of peers that have to locally execute the query in order to retrieve a certain fraction of all globally available relevant results (and, thus, the query execution cost) evidently highly depends on distribution of data over the peers. If the assignment of data to peers is in a completely random fashion and without data replication, for example, it is almost impossible to identify a small number of peers that together contain a reasonable fraction of relevant results, as the relevant results are expected to be uniformly spread over all peers. As another extreme case, consider the existence of mirrored collections, i.e., peers that share the same local index. Even if their indexes are expected to deliver a large number of relevant results, including both peers in the query execution would unnecessarily cause additional cost, as the second peer would not be able to add any previously unseen results.

Also, realistically there is no "one-fits-all" assignment of documents to peers. Different P2P IR systems are designed for a large variety of application scenarios, intended peer popularities, and assumptions on the existing data placement. Thus, fixing a particular partitioning of any document corpus will inevitably lead to hesitations by one or another research group and would too much limit our vision of a standardized general benchmark. Instead, we need a data placement strategy that is easy to implement and that exposes unrestricted flexibility of assigning documents to an arbitrary number of peers, and all this cast into as few and intuitive parameter settings as possible. Such an approach also allows for an experimental study to easily run several experiments with varying number of peers and degrees of data replication, e.g., to evaluate the asymptotic behavior. Also, we envision that in such cases a certain amount of preprocessing (e.g., indexing the documents or computing global statistical measures) can be re-used for all subsequent experiments.

One important aspect of the resulting document partitioning that has already briefly been mentioned before is the degree of data replication, i.e., how many collections a particular document is assigned to. Because of their inherent churn, i.e., peers leaving and entering the system at their own discretion and on short notice, P2P systems often try to enforce (or simply assume) a certain degree of data replication, so that all data remains accessible at any time with a high probability. Also, data replication often serves as a mean of load balancing, such that accesses to popular data items can be satisfied by different resource suppliers. A natural degree of mutual overlap, for example, is inherent in a scenario in which peers (i.e. users) autonomously download (copy, cache, ...) data items of personal interest. Studies in such an environment have shown a skewed distribution of data items, with popular items (e.g., recent mp3 files) being present at a substantial fraction of all peers. Thus, one of the key requirements for a P2P retrieval benchmark is the ability to control the resulting overlap between collections. However, as some IR systems do not consider overlapping collections but assume disjointly partitioned collections, the desired benchmark should also allow for the creation of disjoint partitions of the document corpus.

We postulate the following requirements for a suitable data placement strategy:

- The data distribution should be *reasonable*, i.e., in a way that it could well appear in the real-world.

[1]Using the files themselves would evidently create intellectual property issues.

- The data distribution should be *complete*, i.e., all data items should eventually be available at at least one peer.

- The data distribution should be *uniform*, i.e., all peers should have the same expected size.

One could argue that the requirement of uniform peer sizes is unrealistic itself. But in order to enable also a theoretic evaluation of P2P IR approaches, it allows for easy statistical analysis.

3.3 Queries

Suitable benchmark queries, in our eyes, have to fulfill two major requirements:

- Fit the document corpus: obviously, it does not make sense to execute queries for recent events or starlets on a corpus that is unable to contain any data relevant to this topic.

- Real-World: as we try to model the system behavior of a P2P IR system that is intended to be used by human beings, we should not use synthetically generated queries that can never fully reflect the nature of real-world queries.

- Significance: in our eyes, it is not meaningful to use benchmark queries that do not well reflect the typical workload of an information retrieval system. Instead, our benchmark should focus on statistically significant queries.

We propose the usage of queries taken from Google's *Zeitgeist* archive[2]. To obtain a sufficiently large, diverse, and significant sample, we propose the usage of all queries reported by the British google.co.uk department for 2005 at the time of this publication. Thus, the Wikipedia community was able to create content for any of the topics addressed by the queries. For reference, we list all queries in Appendix A.

For the future, we envision a second benchmark query set, consisting of particularly difficult queries.

We will revisit the **benchmark metrics** in Section 5.

4. DATA PLACEMENT

4.1 Distributing Data

One obvious way to assign web data to different peers is to use simulated crawls, i.e., each peer performs a crawl of a given size on the data collection starting from individual start pages, following the hyperlinks of documents, and uses the set of visited pages as its data set. While this produces realistic results, the results are hard to control. First, controlling the amount of overlap produced the crawls is difficult. Second, the crawls might not discover all pages (because of missing hyperlinks or insufficiently chosen peer size), so that some documents will be left out. If these documents are added to peers in a post-processing step, the result will no longer be realistic, as the peers might contain unconnected documents that could not have been obtained by a Web crawl. Third, a uniform distribution of data cannot be guaranteed when crawling. Still, the crawling scenario gives reasonably realistic partitionings, therefore the desired partitioning algorithm should partition the data at least roughly like a crawler.

[2]http://www.google.com/zeitgeist

Cluster($G = (V, E)$,k)
  number all $v \in V$ (lexicographically)
  create k clusters $C_i = \{v_i | \frac{|V|(i-1)}{k} \leq i < \frac{|V|i}{k}\}$
  do
    find a $v_i \in C, v_j \in C'$ with
      $out_{C'}(v_i) > in(v_i) \wedge out_C(v_j) > in(v_j)$
    ordered by $mout(v_i)\downarrow$,i$\uparrow$,$out_{C'}(v_i)\downarrow$,$C'\uparrow$,$mout(v_j)\downarrow$,j$\uparrow$
    place v_i in C' and v_j in C
  while suitable v_i and v_j could be found
  return $\{C_1 \dots C_k\}$

Figure 1: Greedy clustering algorithm

An attractive data placement strategy is clustering. Clustering can be done by looking at the content (which is possible for almost all kinds of data sets) or, for linked data, by looking at the graph structure. Clustering according to the graph structure has the advantage that it mimics the behavior of a crawler: Documents that point to each other are likely to be put in the same cluster. The advantage of clustering over crawling is that clustering allows for a much finer control of the resulting data distribution.

[13] shows that the web is self-organizing in the sense that web communities are formed automatically. These communities can be easily identified by considering the link distribution within these pages. It is shown that web pages have more links to other pages inside their community than to pages that are outside their community.

While a standard clustering technique like k-means [25] could be used, it does not reflect the behavior of a crawler very well. Graph-based clustering strategies like [17] cluster the data such that the number of edges between clusters in minimized, which means that data linking to each other is placed in the same cluster. This is a reasonable clustering that is a good approximation to a crawler behavior. Unfortunately, finding the optimal clustering is NP-hard and the existing clustering algorithms usually violate some of our requirements. Therefore we propose a simple greedy cluster algorithm that can be used to partition the data suitably.

The clustering algorithm interprets the documents as vertices in an undirected graph: Each document is represented by a vertex and each link forms an edge between nodes. To evaluate the quality of the clustering for a given vertex, we define three metrics:

$in(v)$ = #edges from v to other nodes in the same cluster
$out_C(v)$ = #edges from v to nodes in cluster C
$mout(v)$ = $\max_{v \notin C} out_C(v)$

The clustering algorithm is shown in Figure 1. It takes an undirected graph and the number of clusters as input and produces a suitable clustering. In a first step it labels all vertices by numbering them, using the label later for an initial clustering and as a tie breaker. This labeling could be done arbitrarily, however to get reproducible results we number the vertices according to the lexicographic order of the document titles or URLs. The labels are used to create the initial clusters.

Now the clustering is improved repeatedly finding pairs

DataPlacement($G = (V, E)$,peers,topics,chunks,overlap)
  p=$\lfloor$ peers / topics $\rfloor$
  s=chunks-overlap
  assert $p \in \mathbb{N} \wedge s > 0$
  T=Cluster(G,topics)
  for all $t \in T$
    H=Cluster($G_{|t}$,$p * s$)
    for all $0 \leq i < p$
      $C_i = \cup_{i*s \leq j < i*s+s+overlap}\ H[j \bmod p * s]$
    $C_t = \cup_{0 \leq i < p}\ \{C_i\}$
  return $\cup_{t \in T} C_t$

Figure 2: Data placement algorithm

of vertices such that the number of links between clusters decreases if the two vertices switch the clusters. If such a pair can be found, they are placed in the other cluster respectively and the next pair is determined. If no pair can be found the algorithm stops. Swapping two vertices instead of moving one vertex guarantees that the cluster sizes remain constant. To guarantee reproducible results, the algorithm defines a preference if multiple pairs could be swapped (the order by line): The algorithm prefers to fix the vertex with the high gain (*mout*) and tries to swap it with the vertex in the target cluster that has the most links to the current cluster. If these numbers are ambiguous, it prefers vertices and clusters with lower ids. This vertex selection strategy can be implemented easily by using priority queues for the clusters.

4.2 Creating Overlap

The clustering algorithm allows for a reasonable partitioning of data, but using each of the results clusters directly as one peer's local collection produces peers without overlap. While this might be intented in some cases, peers will usually contain overlapping data, as they acquire data independently. Unfortunately there is currently no established way to model overlap. Therefore we present a plausible but simple method of creating overlap here. Future work should develop a more realistic model, however our model is realistic enough to test different peer selection strategies.

Our basic assumption is that overlap primarily occurs between peers that are interested in the same topic. Furthermore we assume that overlap does not occur on the level of single documents but for groups of (related) documents. This makes sense (assuming a crawler-based acquisition of data) because two peers starting their crawl from a page within the same community will end up with an overlapping group of documents, as pages in a community point to each other.

To create an overlapping data distribution, the documents are first clustered into larger topics using the clustering algorithm introduced above. The documents in each topic are then clustered into smaller related chunks, using the same clustering algorithm on each topic cluster. These *chunks* are assigned to peers using a sliding window technique: Each peer is assigned a certain number of chunks and the next peer shifts the window a certain number of steps, creating any desired degree of overlap.

The algorithm is shown in Figure 2. It first calculates the numbers of peers per topic (p) and the size of the unique part of the first peer in a topic (s). Note that p must be a natural number and s a positive number ($s = 0$ would imply identical peers inside a topic, this could be accepted also). Then the documents are clustered into the desired number of topics. The documents in each topic are then clustered into the correct number of chunks, such that each of the p peers inside the chunk can be assigned *chunks* chunks, overlapping in *overlap* chunks with the next one. The data placement inside a topic t is stored in C_t. After all topics are processed, the final data placement is returned as union of all C_t.

By controlling the parameters *chunks* and *overlap*, any desired amount of overlap can be produced. While these parameters can easily be justified (and calculated), the parameter *topics* is somewhat arbitrary. Quantifying *topics* is difficult. If the number is set too low, unrelated peers end up share data, if it is set too high, the peers end up highly specialized. In Section 6 we used 100 topics for the Wikipedia corpus. This seems to be a reasonable choice, as Wikipedia's own categorization contains about 130 top-level categories[3].

5. BENCHMARK METRICS

5.1 Performance Measures

Result Quality

Precision and recall are the basic quality measures used in evaluating the quality of the result returned by an (distributed) information system. More specifically,

- **Recall** is the fraction of the number of returned relevant documents to the total number of relevant documents.
- **Precision** is the fraction of the number of returned relevant documents to the total number of relevant

These measure require a query-specific classification of documents in a set of relevant and non-relevant documents. This classification is usually done manually and thus requires a non-negligible human effort.

In the absence of human relevance assessments, one can use the so called **relative recall**, i.e. the ratio between the number of relevant documents returned by the distributed system (as a function of the number of peers involved in the query) and the total number of relevant documents available in the system. In distributed information systems it is often argued that this measure reflects the quality of the returned results w.r.t. to a hypothetically created centralized information system.

Efficiency Measures

- **Query Response Time**: This is the time a potential user has to wait for the query results and consists of network transfer cost, local computation, and local IO latency. Note we are interested in measuring the total execution; if several requests) run in parallel, we need to account for the longest one.
- **Network Resource Consumption**: With this measure we try to assess the system's ability to handle queries in an efficient way, i.e. requiring only a reasonable amount of available network resources. We

[3]http://en.wikipedia.org/wiki/Wikipedia:Browse

are interested in the total number of messages and the total number of bytes transmitted during query processing.

5.2 Modeling System and Network Costs

Running the experiments over multiple nodes in a network is inherently vulnerable to interference from other processes running concurrently and competing for cpu cycles, disk arms, and network bandwidth. To avoid this and produce reproducible and comparable results for algorithms ran at different times, we propose simulating disk IO latency and network latency, which are dominant factors for the total execution time. Specifically, we assume a random disk IO to incur a disk seek and rotational latency of 9 ms, plus a transfer delay dictated by a transfer rate of 8MB/s. For network latency we assume typical round trip times (RTTs) of packets and transfer rates achieved for larger data transfers between widely distributed entities [32]. We assume a packet size of 1KB with a RTT of 150 ms and use it to measure the latency of communication phases for data transfer sizes in each connection up to 1KB. When more data it sent, the additional latency is given by a "large" data transfer rate of 800 Kb/s. This number is an average throughput value (using one stream – one cpu machines) obtained from experiments conducted for wide area network throughput, sending 20MB files between SLAC nodes (Stanford's Linear Accelerator Centre) and nodes in Lyon France [32] using NLANR's iPerf tool [36].

6. EXAMPLE BENCHMARKS

We present an example application of the proposed benchmark, using our MINERVA P2P Web search engine. We created 1,000 peers using our proposed data placement algorithms: We first clustered the complete Wikipedia corpus into 100 disjoint partitions. In a next step, we created 1,000 peers such that each documents is contained in 3 peers. Thus, we used the following parameters: *topics*=100, *peers*=1,000, *chunks=3*, *overlap=2*. Manual inspection of some peers suggested that the resulting data distribution is indeed plausible.

We have used a standard query routing (aka peer selection) strategy that considers for each peer and query term the number of documents in the peer's collection that contains the term.

Quality

In Figure 3 we report on the relative recall for different number of queried peers. As one can see, the query routing works remarkably well: we retrieve $\sim 37\%$ of the globally relevant documents when combining the local results from 10 peer, i.e., only 1% of the total number of peers. This perfectly fits our goal to achieve a high information quality while causing only a small communication overhead. As mentioned earlier, this excellent result is possible since for each query there is only a small number of peers that store relevant information. This is due to the data driven assignment from documents to peers, given by the clustering algorithm that, from our point of view, is a more realistic assumption than a random assignemt from documents to peers. Furthermore, as shown in Figure 3, we achieve a relative recall of 80% with 50 peers queried. This indicates that our distributed system provides all salient features of a distributed search engine and at the same time is extremely efficient and achieves the result quality of a centralized search engine that uses the complete available knowledge with contacting only around 5% of all peers.

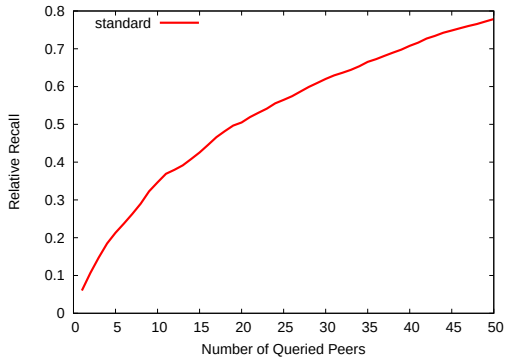

Figure 3: Recall Performance for the complete Benchmark (99 Queries)

Network Traffic

Metadata retrieval: In total the query initiator issued 146 PeerList requests with in total 122, 218 retrieved posts. Each posts consists of the contact information IP and PORT ($4 + 2$ bytes), and the number of documents (2 bytes) for a particular term as the quality of service information, thus the query initiator retrieves metadata of 977, 744 bytes for the complete benchmark of 99 queries. Note that the average PeerList length is 837 and that we always retrieve the full PeerList for a term that is involved in the query. This is the naive approach and there are obvious optimization techniques like using a top-k algorithm for the metadata retrieval or transferring only the top-50 Posts for a single keyword query if we are interested, for instance, in finding the top-50 most promising peers per query.

Query Execution: Now assume that for each query the query initiator sends the query to the top-50 peers that return their top-100 documents. Each document is represented by a URL of 30 bytes (average) and a score (floating point number of 4 bytes). This results in a network traffic of $\sim$ 16.8MB. It is important to note that the number of peers that are contacted during query execution is a tuning parameter and, here, set to 50 as an illustration. Obviously, reducing the number of contacted peers to 10 will decrease the network load during query executing by a factor of 5 and at the same time will provide reasonably good results (cf. Figure 3).

Overall, the retrieval of metadata and contacting 50 peers for 99 queries cause a total network traffic of $\sim$ 17.8MB, i.e. an average network traffic of $\sim$ 178kB per query. Note that we can further reduce this number by limiting the number of returned documents per peer from 100 to, e.g., 20, which has shown to be sufficient in real-world systems.

Number of Messages

Metadata retrieval: During execution of the 99 benchmark queries, the query initiator retrieves 146 PeerLists

from the MINERVA directory, causing $2 * 146 = 292$ messages (requests and corresponding replies together).

The actual query execution requires $2 * 50 * 99 = 9900$ messages, i.e. the number of messages in order to send the queries to the selected peers that eventually send back their local query results. So the total number of required messages is $10,192$.

Note that these numbers report only on the traffic caused by the workload of the 99 query. It does not consider the traffic caused by background processes like DHT stabilization messages or messages to build / rebuild the Minerva directory.

Overall, considering result quality and communication cost, our system shows a remarkably good performance. The $\sim$ 178kB network traffic per query when involving 50, or $\sim$ 35kB when involving 10 peers, are well within today's bandwidth opportunities, and the quality of the query results involving only a small number of peers is almost as good as the results obtained from a hypothetical centralized system that has global knowledge.

7. CONCLUSION

We have suggested a standardized benchmark framework for P2P IR systems, based on the Wikipedia corpus and Google *Zeitgeist* queries, using a reproducible, but yet powerful and flexible clustering method that assigns documents to a variable number of peers.

As a next step to a general-purpose P2P IR benchmark, we plan to include a model for network dynamics, i.e., peers joining and leaving the systems continuously. Also, we plan to obtain manually derived relevance feedback for all queries in our benchmark for all documents in the document corpus.

8. REFERENCES

[1] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. In *ISWC 2004*.

[2] G. Z. Archive. http://www.google.co.uk/press/zeitgeist/archive.html.

[3] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving collection selection with overlap awareness in p2p search engines. In *SIGIR05*.

[4] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: Collaborative p2p search. In *VLDB*, pages 1263–1266, 2005.

[5] BitTorrent. www.bittorrent.com/.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[7] J. Callan. http://hartford.lti.cs.cmu.edu/callan/data/.

[8] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR1995*, pages 21–28. ACM Press, 1995.

[9] P. Cao and Z. Wang. Efficient top-k query calculation in distributed networks, PODC 2004.

[10] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *HPDC*, 2003.

[11] D. A. P. David Oppenheimer and J. M. Hellerstein. Decentralized systems need decentralized benchmarks. Technical Report UCB/CSD-03-1234, EECS Department, University of California, Berkeley, 2003.

[12] DBLP. http://www.informatik.uni-trier.de/∼ley/db/.

[13] G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization of the web and identification of communities. *IEEE Computer*, 35(3):66–71, 2002.

[14] M. Gerla, C. Lindemann, and A. Rowstron. P2p manet's - new research issues. In M. Gerla, C. Lindemann, and A. Rowstron, editors, *Perspectives Workshop: Peer-to-Peer Mobile Ad Hoc Networks - New Research Issues*, number 05152 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/213>.

[15] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, 1999.

[16] N. Gunther and G. Beretta. Benchmark for image retrieval using distributed systems over the internet: Birds, 2001.

[17] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Inf. Process. Lett.*, 76(4-6):175–181, 2000.

[18] T. Hernandez and S. Kambhampati. Improving text collection selection with coverage and overlap statistics. pc-recommended poster. WWW 2005.

[19] S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2p-diet: An extensible p2p service that unifies ad-hoc and continuous querying in super-peer networks. In *SIGMOD*, 2004.

[20] Initiative for the Evaluation of XML Retrieval (INEX). http://inex.is.informatik.uni-duisburg.de/2006/.

[21] S. T. Kirsch. Distributed search patent. u.s. patent 5,659,732.

[22] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[23] L. S. Larkey, M. E. Connell, and J. P. Callan. Collection selection and results merging with topically organized u.s. patents and TREC data. In *CIKM*, pages 282–289, 2000.

[24] J. Lu and J. P. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM*, pages 199–206, 2003.

[25] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[26] S. Michel, M. Bender, P. Triantafillou, and G. Weikum. Iqn routing: Integrating quality and novelty in p2p querying and ranking. In *EDBT*, pages 149–166, 2006.

[27] S. Michel, P. Triantafillou, and G. Weikum. Klee: A framework for distributed top-k query algorithms. In *VLDB*, pages 637–648, 2005.

[28] National University of Singapore. http://www.comp.nus.edu.sg/∼p2p/.

[29] NFCR. http://www.nfcr.org/default.aspx?tabid=274.

[30] H. Nottelmann, G. Fischer, A. Titarenko, and

A. Nurzenski. An integrated approach for searching and browsing in heterogeneous peer-to-peer networks. In *HDIR 2005*.

[31] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR*, pages 290–297. ACM Press, 2003.

[32] D. Salomoni and S. Luitz. High performance throughput tuning/measurement. http://www.slac.stanford.edu/grp/scs/net/talk/High_perf_ppdg_jul2000.ppt. 2000.

[33] SETI@Home. http://setiathome.berkeley.edu/index.php.

[34] T. Suel, C. Mathur, J. wen Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. In *WebDB*, 2003.

[35] Text Retrieval Conference (TREC). http://trec.nist.gov.

[36] A. Tirumala et al. iperf: Testing the limits of your network. http://dast.nlanr.net/projects/iperf/. 2003.

[37] Y. Wang, L. Galanis, and D. J. de Witt. Galanx: An efficient peer-to-peer search engine system. *Available at http://www.cs.wisc.edu/∼yuanwang*.

[38] Wikipedia. http://download.wikimedia.org/enwiki/20060125/enwiki-20060125-pages-articles.xml.bz2.

APPENDIX

A. ZEITGEIST QUERIES

3
50 cent
abi titmuss
alton towers
angelina jolie
argos
arsenal
bbc
bbc news
bbc sport
beyonce
big brother
bitesize
blink 182
brad pitt
brian mcfadden
british airways
britney spears
cars
cats
cbbc
cbeebies
celebrity love island
charlie and the chocolate factory
charlotte church
crazy frog
currency converter
david beckham
dictionary
dog
dogs
dolphins
eastenders
easter
easyjet
ebay.co.uk
eminem
expedia
face party
fish
florida
flowers
football
france
friends reunited
games
girls aloud
good charlotte
gorilla
grand national
green day
harry potter
ibiza
inland revenue
jennifer ellison
jennifer ellisson
jessica alba
johnny depp
jordan
kenya
lingerie
little britain
live8
liverpool
london
london marathon
love hearts
madagascar
manchester united
mcfly
michael jackson
multimap
o2
oasis
orange
paintball
paris hilton
paula radcliffe
playboy
pope
red nose day
robbie williams
ryanair
simpsons
spain
star wars
t mobile
tesco
the killers
the wedding crashers
train times
tsunami
u2
usher
valentines day
valentines' gifts
vodafone
war of the worlds
wimbledon