Scalability Assessment of Complex Boolean Queries

Jefferson Heard, Joseph Prokop, David Grossman, Ophir Frieder
Information Retrieval Laboratory, Dept. of Computer Science, Illinois Institute of Technology
10 W. 31st. St.
Chicago, IL 60611
{heard,prokop,dagr,ophir} @ir.iit.edu

Keywords: Search and Retrieval

Abstract
Complex Boolean queries specify exact logical constraints for document relevance. Unlike for natural language based queries, presently no benchmark exists to evaluate systems executing lengthy Boolean queries (including wildcard requests as well as proximity searches). We developed a set of queries specifically designed to test the scalability of a complex Boolean information retrieval system. We used our benchmark to evaluate a commercial search engine with data modeled on a system in current production use. Several indexing configurations were studied. Our experimental findings show that surprisingly the shared index configuration is more scalable than the distributed index configuration.

1 Introduction
Complex Boolean Information Retrieval (CBIR) systems are highly accurate as the nature of relevance can be formulated precisely using, potentially complicated, logical expressions. As applications using complex Boolean queries often involve many Boolean operations and search massive data collections, lengthy query processing times are common.

Our task was to evaluate the effects on processing times of various index configurations for a commercial CBIR engine under a typical expected load of a defense contractor application. When designing the experiments, however, we found that there was no existing collection of queries or data that could serve as a benchmark to evaluate CBIR systems. Thus, we collected a terabyte of data and created a set of 837 queries to rigorously test I/O and CPU usage. A significantly large subset of these queries is available as the IIT Scalability Assessment for Boolean Retrieval Engines (ISABRE) benchmark at http://ir.iit.edu/scalability/isabre. Those queries specific to the commercial engine under evaluation are not included. The data used was provided by Paracel.

The tests covered three separate collections and used a broad set of parameters. The parameter combinations were selected based on the effect they should have on overall performance and to test the interaction of various parameters with one another.

We modeled our testing based on past information systems performance evaluations. Two such studies include a distributed performance evaluation of the INQUERY engine [1] and that of a large-scale finance application [2]. These evaluations considered metrics such as the rate of queries handled, memory and CPU usage, and several other metrics. Given space limitations only a high-level overview of our findings are reported here.

2 Experiments

2.1 Query Collections
The difficulty with all current benchmarks for large collections of data is that they are all TREC oriented. TREC queries are natural language based, with no Boolean structure, and an arbitrary Boolean representation of them might not test a CBIR system fully. Instead, we developed the IIT Scalability Assessment for Boolean Retrieval Engines (ISABRE). ISABRE is a combination of keywords drawn from TREC topics planted in a set of structured Boolean queries that are already known to provide a rigorous test in a production environment. We took 837 query structures (no keywords from the original queries were retained) that our customer provided us and built queries from the TREC topics. These queries were known to be intensive in terms of either or both of I/O and CPU usage. The customer has also tested the original queries against the production system that our experiments simulated and found the benchmark to be accurate. Currently, we are testing a second CBIR system using this benchmark.

In translating the queries, we made sure that against our dataset there were roughly the same proportion of queries
returning no results and the same proportion of wildcards, etcetera, so as to most closely approximate the original test query set.

2.2 Testing

Our distributed architecture consists of 15 identical machines, each equipped with two AMD Athlon MP 2400+ processors, 2GB main memory, and two 78GB 7,8KRPM SCSI disks. A dedicated copper gigabit switch connects all machines. Our shared architecture uses the same machines, but in addition, a configured Polyserve Matrix Server connects them to RAID over a dedicated 2-gigabit interface.

In both architectures, all nodes issue queries. In the shared architecture, each issues a subset of the total set of queries against the entire shared index. In the distributed architecture, each machine issues all the queries on their own index subset. This means that the distributed architecture may well do more work than the shared architecture, since each index subset must be hit equally for all queries, and if an N document maximum is requested, then up to N must be returned by each query processing engine, and merged. We believe this is the primary reason behind the shared system's performance advantage. Queries are run with a constant number always executing (until the end when the last queries are allowed to finish running but no new ones are started).

Testing parameters were selected based on the manufacturer's suggestions. Where interaction was deemed likely between parameters, the variations for each parameter were cross-tested. Our core test groups consist of: a control group, an intensive query group, a group that varies memory and server process count, a group varying number of documents retrieved, a group testing a higher number of parallel query tasks, a group where the concurrency is increased from 5 to 500 concurrent queries in queue, and a group that tests which provided query load balancing algorithm is best. Our mid-sized collection saw throughputs in the shared architecture stay relatively stable as compared to the distributed one, which dropped dramatically. Throughput at each quartile of performance was nearly equal for the two architectures, varying on average as little as half a query per second between the two.

Our largest collection actually saw an increase in performance for the shared architecture, with a modest drop in performance for the distributed architecture. Even for the best settings for each collection, the distributed architecture performed at half a query per second slower than the shared architecture. The average queries per second per architecture for the various collection results are summarized below:

<table>
<thead>
<tr>
<th>Collection Size</th>
<th>Shared</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>100GB</td>
<td>3.00</td>
<td>18.88</td>
</tr>
<tr>
<td>320GB</td>
<td>2.61</td>
<td>3.65</td>
</tr>
<tr>
<td>1TB</td>
<td>4.04</td>
<td>1.40</td>
</tr>
</tbody>
</table>

4 Analysis and Conclusions

In general, the performance of the shared architecture over the different collection sizes did not change much, suggesting the RAID was a bottleneck for small collections, but less so for the larger ones, which allowed such factors as cache coherence, Polyserve's load balancing and pre-fetching, and network saturation to come into play in the comparison between systems. The distributed architecture's performance dropped considerably as more and more I/O was done on the system for the larger collections. At one terabyte, the performance graphs cross and the shared system outperforms the distributed system consistently, if only slightly. It would be interesting to find out at what point the shared architecture bogs down.

Acknowledgments

We gratefully acknowledge Vicki Holt and Darryl Howard in the review of our study, test plan, and results. The figure uses icons available free from http://art.gnome.org

References