Building a Scalable Knowledge Space on the Cloud: Initial Integration and Evaluation

Delsey Sherrill, Jonathan Kurz, and Craig McNally
MIT Lincoln Laboratory
{dsherrill, jonkurz, cmcnally}@ll.mit.edu

Application

As part of the DoD's net-centric data strategy, significant efforts have focused on standardizing formats and consolidating access to ISR sensor products. Meanwhile, the collective knowledge of human analysts, in the form of finished intelligence products, continues to be captured in weakly structured document formats such as PowerPoint and then disseminated over ad hoc channels such as email. As a result, analysts rarely benefit from current knowledge of other units unless they are subscribed to the same email distribution lists or have access to the same file shares. Further, transfer of knowledge to an incoming unit is often limited by face-time with the departing unit. Even if inheriting a digital archive, the new unit lacks tools to easily discover archival knowledge relevant to their mission.

MIT Lincoln Laboratory has developed the Structured Knowledge Space (SKS) system for extracting entities, spatiotemporal references, and relationships from weakly-structured intelligence documents and making them discoverable over the network using similar tools as for sensor data.

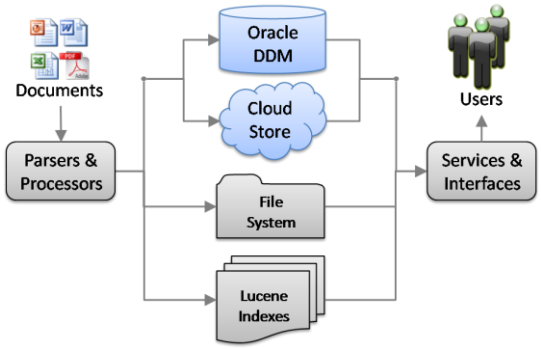

Figure 1: SKS system diagram.

Architecture

Figure 1 shows the overall flow of information through SKS. Documents are received in their original form through manual upload, automated report feeds, and email. Current the system handles the most common reporting formats such as Microsoft Office and PDF. The text is extracted from its original binary format and processed to validate its uniqueness, verify its classification label, and enrich its metadata by recognizing named entities and spatiotemporal references. The processing results are persisted in an Oracle database using a dimensional data model (DDM) [1]. In addition the text and metadata are indexed for full-text search using Lucene [2], an open-source search engine in Java.

On the retrieval side, human analysts use the web search interface to formulate searches for documents of interest, view the results, and select individual documents to download and read in greater depth. In addition, SKS RESTful web services provide an API for third-party tools to access SKS data and resources for presentation in their own applications.

Computation

The computations performed by SKS in response to a search request are divided among three main components. First, a query to Lucene returns a list of document identifiers most relevant to the query. Given the identifier list, a second component calculates facet counts, and a third component converts that list into human readable search results.

Searching in Lucene involves scanning its inverted indexes for matches to the current query, computing a relevance score for each match, and then sorting the matches by relevance score and returning them to the caller. Lucene uses a number of optimizations that enable sub-second keyword search performance on indexes containing millions of documents.

Computing the facet count involves finding the most common named entities by category (e.g., persons, locations, and organizations) within a set of documents. The purpose of facets is to summarize what entities are related to the user's search, and to enable rapid drill-down to the documents mentioning a particular entity. Most demanding is the computation of facet counts over all documents in the system, when there is no active search, as happens when the SKS search page is first opened.

Formatting the search results involves a straightforward lookup of each document's metadata and raw text. The raw text is needed to generate a highlighted context snippet for each search result based on the current query.

Cloud Integration

While the current storage architecture and computational resources have met SKS requirements to date, there are compelling reasons to explore alternatives for the future. Cloud computing has gained a following in recent years because it replaces expensive shared memory hardware and proprietary database software with cheap commodity hardware clusters and open source software. By integrating cloud computing with SKS, we hope to deliver comparable (if not superior) search performance to users at a lower long term cost.

For the initial integration efforts, we are making use of our organization's existing high-performance computing infrastructure known as LLGrid [3]. Recently a dynamic cloud deployment capability has been developed for LLGrid. As shown in Figure 2, the deployed cloud stack include a block-based distributed file system, Hadoop DFS [4], and a secure data access layer to the cloud triple store.

Our approach to integrating the cloud stack with SKS is an incremental one. Rather than attempt to replicate the full data model in the cloud, we have instead focused on the two key text retrieval functions that are supported by the Oracle DDM: facet computation and search result formatting, and have implemented a simple data model in the cloud store for just those two functions.

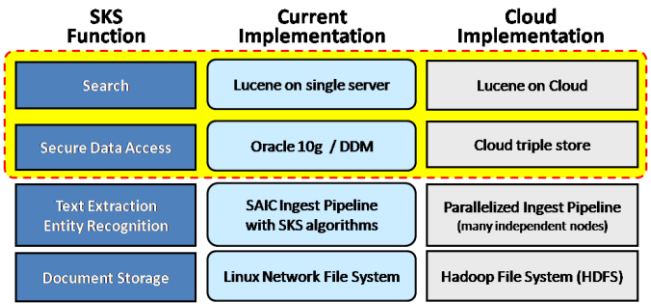

Figure 2: Cloud stack (highlighted portions to be evaluated).

The other major integration point for cloud computing in SKS is the Lucene search engine, which runs within a single JVM under the current implementation. Several open source projects are underway to port Lucene into the cloud computing environment. We have begun experimenting with Katta [5], which pairs Lucene with the Hadoop cloud platform. We intend to explore other alternatives as well, including Lucandra, which is based on the Cassandra open source cloud stack [6].

Evaluation

Comparing two systems with radically different storage architectures presents a considerable challenge due to the multitude of possible software configuration parameters and hardware platforms. Our plan is to load identical data sets into both systems, run the same sets of search queries, and take a "best foot forward" approach to the comparison. That is, each system will be tuned independently and then the comparison will be based on the best configuration available for each of the systems. The timings of components that are common to both systems will be separated from the timings of the components that differ across systems.

We plan to benchmark the Oracle and cloud systems at exponentially increasing levels of loading (from 500k to at least 10M documents total) by measuring the overall search request timing and the timings of search subcomponents for every test request. These measures will also guide exploration of the parameter space of possible software and hardware configurations for each individual system.

Overall search request timings will be recorded from requests to the SKS search application generated by JMeter, an open source web application benchmarking tool [7]. JMeter automates the request process, enabling tests of thousands of queries at a time under controlled conditions. Subcomponent timings for Lucene search, facet computation, and result formatting will be recorded in the server logs for each JMeter test that is run.

Results

Preliminary results from the timing of SKS search service requests on the current Oracle-based system are shown in Figure 3. The database and application were run on separate servers, and JMeter requests were sent from a third machine. The document load at test time was 500,000, and the test requests consisted of 200 unique keyword queries, all of which were known to occur in at least one document.

The horizontal axis indicates total time per request, and the vertical axis shows the timing for each subcomponent in the request. Note that the facet computation and result formatting components run concurrently, so that their contribution to the total time is a maximum rather than a sum. The Lucene search component is run prior to these two components and thus its contribution to the search request timing is additive.

The strong correlation between the Lucene component timing and the total request timing indicates that Lucene is the primary driver of search times. The underlying causes of Lucene search variance are still under investigation, though we have ruled out frequency of the search keywords within the document corpus as a factor. The other components are relatively constant across searches, with facet computation about 10 ms slower on average than result formatting.

It remains to be seen whether these patterns will hold for larger document loads on the current and cloud implementations of SKS. Our future investment in cloud technology depends crucially on demonstrating that its scalability in the context of our application domain is at least on par with scalability of the current system. We look forward to reporting these results in the coming months.

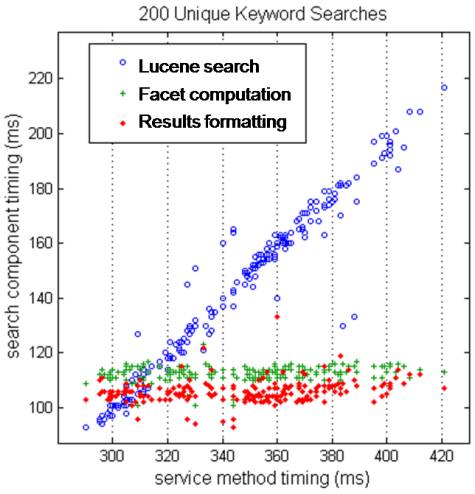

Figure 3: Preliminary search performance results.

References

[1] R. Kimball, "A dimensional modeling manifesto," *DBMS,* vol. 10, pp. 58-70, 1997.

[2] Lucene, http://lucene.apache.org

[3] N. T. Bliss, R. Bond, J. Kepner, H. Kim, and A. Reuther, "Interactive Grid Computing at Lincoln Laboratory," *Lincoln Laboratory Journal,* vol. 16, pp. 165-216, 2006.

[4] Hadoop, http://hadoop.apache.org

[5] Katta, http://katta.sourceforge.net/

[6] Cassandra, http://cassandra.apache.org/

[7] JMeter, http://jakarta.apache.org/jmeter