

A Distributed Blog Search Platform

Ian Fischer, Elias Torres
Harvard University
{fischer,torres}@fas.harvard.edu

Our project is to design, implement and evaluate a distributed search platform based on Hadoop [1], an open source implementation of Google's MapReduce [2] and GFS (distributed file system) [3]. Additionally, we would like to perform a comparative analysis with the results from Cobra [4], a blog aggregator and content-based filterer.

1 Introduction

Blogs are the latest form of communication on the Internet today. In their rawest forms they are online dairies published on the web in reverse chronological order. Their contents are easily managed by lightweight content management systems that have enabled a large number of authors varying in technical abilities to publish their thoughts on the Web. The collection of blogs on the Web is commonly referred to as the blogosphere; Technorati [5] reports that they are tracking 50 million blogs and it is currently doubling approximately every six months. More important than the blog creation rate is the blog posting rate estimated at 1.6 million posts a day [6]. As expected with the large number of publishers and content, the number of blog readers grows each day, but unfortunately it is not easy to find and track content in the blogosphere.

Naturally, a flurry of Internet startups have formed delivering services in blog content discovery and tracking, but unfortunately most of the technical and system details are not publicly available. There has been a small number of papers that suggest algorithms for specific queries and analysis on the blogosphere, but even fewer papers that outline a complete blog aggregator system. Our goal is to propose an alternative design to Cobra and to document and evaluate it.

2 Related Work

2.1 Cobra

Cobra is a blog aggregator and content-based filtering system. It uses a three-tiered network of *crawlers* that scan web feeds, *filters* that match crawled articles to user subscriptions, and *reflectors* that provide users with an RSS feed containing search results. Cobra adds a novel service provisioning technique that decides the minimal amount of physical resources needed to host a Cobra network based on the list of blogs to crawl.

The **crawler service** in Cobra makes use of well-known features in HTTP such as ETags and Last-Modified headers to reduce the bandwidth needed to update the feeds. Additionally, the simple fact of having a centralized crawler amortizes the network usage on behalf of a large number of users. Cobra also assigns source feeds to crawlers based on DNS latency to determine network locality and reduce end-user latency.

Cobra's **filter service** makes use of a clever matching algorithm proposed by Fabret *et al.* [7] The algorithm is a two-phase algorithm that has the advantage that words mentioned in multiple subscriptions are only evaluated once. The paper also mentions support for disjunctive queries by injecting separate conjunctive queries. The results from the evaluation show the ability to match 1 million queries in under 10ms.

The **reflector service**'s jobs are to receive matching articles and deliver RSS feeds to the users interested in that subscription. There could be any number of reflector nodes and users are assigned to a specific reflector. The crawlers send a full copy of the article and re-compute the matching algorithm to see which subscriptions it originally matched.

2.2 Nutch

Nutch [8] is an open source search engine project at Apache. It has built-in support for crawling the web and building the necessary indexes to support users submitting searches via a web application. It makes use of Hadoop for distributed crawling and index building, but does not provide an out-of-the-box blog search experience. In fact, there are many companies starting their search sites based on Nutch such as blogdigger.com [9] and mozdex.com [10]. Our goal is to extend Nutch capabilities by adding a specific set of mappers and reducers that would provide significant value to blog subscribers.

2.3 BlogPulse

Glance *et al.* [11] have documented interesting findings by automatically finding trends in the blogosphere. In addition to trends, they maintain daily lists of key persons, key phrases, and key paragraphs on their website: blogpulse.com [12]. The authors started with a small list of 22,000 blogs and grew it by monitoring a website maintained by Radio Userland where blog systems send a ping message every time they post a new message.

The paper mainly highlights the features of their toolkit, Analyst Workbench. The toolkit has many components that are used by their site to do corpus creation, indexing, phrase finding, trending and data mining. The authors document exciting results but restrict themselves to a results table and do not go into their algorithm details or system design and implementation. We hope to combine some of the BlogPulse functions with a focus on system design to show how such a large blog sensing system could be implemented.

3 Project Description

Our project will take advantage of the MapReduce programming model for distributed computing introduced by Google and later implemented by Hadoop. We will use MapReduce to build a general purpose blog aggregator and query system. The major goals of the system are to handle a large amount of feeds and user subscriptions, and to provide a flexible mechanism to implement new analysis algorithms on the crawled data sources.

The project will be made up of a series of MapReduce classes that will perform several jobs from crawling to building user search results feeds. First, we will implement a basic crawler module for Hadoop that makes use of a new mechanism to incorporate non-Java programs into a Hadoop workflow called HadoopStreaming. The program will be written using Universal Feed Parser [13], one of the most capable feed readers available today, which is written in Python. The module will not only fetch feeds but will additionally normalize them into the Atom Feed format.

More importantly, we need to build a module similar to Cobra's keyword matcher in order to run against our crawled store. The Map function would sift through large chunks of the crawled feeds and output feeds that satisfy the users' subscriptions. However, we would like to implement at least a second program that requires querying at the minimum pairs of feeds in order to obtain the desired results. This module will help us show how effective the MapReduce programming model is at implementing more complex types of queries.

Finally, we would like to compare our results with that of Cobra and hopefully show that our distributed approach will still be within a reasonable time-bound; e.g., within a few minutes of a blog being updated, or within a second of a user requesting a new topic.

In order to motivate our project, we will build our system to handle, at the least, the following operations: finding the top web links (i.e., non-blog) associated with a given topic; and finding the political blogs by searching for names of the politicians running in the national congress races.

4 Novel Aspects

Our project design is different than Cobra as it is less stream-oriented and more focused on the programming model to process the source feeds. We would like to use the MapReduce primitive to implement all novel areas of the system so that new ones can easily be added and existing ones replaced. For example, we would like to perform the locality-aware distribution of source feeds and the pre-processing of users' subscriptions as MapReduce programs.

A missing function in Cobra is new blog discovery. By

using a distributed file system of source feeds, we can run other programs in parallel that analyze the crawled content for links, adding the findings as links with more meta-data such as new blog, external web page, existing blog, etc.

A place in the crawler where there is room for small improvement is in the use of a hash function to detect changes in either the feed as a whole or on a per-entry level. In order to detect changes in Atom for example, one only needs to compare the atom:id and the atom:updated elements in the entry in order to know whether something changed; otherwise it should be legal to ignore changes at the document level. This is a better approach since hash computation is inadequate to properly detect changes, unless a proper XML canonicalization mechanism has been applied in the first place.

5 Hurdles

One of the hurdles we expect to encounter is how to store blog entries – using MapReduce requires a certain amount of static data in a distributed file system, so we will certainly be storing some amount of historical data; it is not yet clear how much is the correct amount, however. As the number of blog entries we store approaches one per blog, we approach a streaming system, but we lose the ability to handle historical queries. As it approaches storing all entries, we start needing significant amounts of storage, and we potentially dramatically increase latency, depending on the queries.

Another potential hurdle is the use of MapReduce – properly phrasing our algorithms as map and reduce functions may occasionally prove challenging, but this is imperative to making our system adequately distributed.

6 Timeline

1. Nov - Week 1: Implement a basic URL injector and crawler using Hadoop
2. Nov - Week 2: Implement a basic keyword matching mapper/reducer
3. Nov - Week 3: Implement a person finder
4. Nov - Week 4: Implement a blog discovery mapper/reducer
5. Dec - Week 1: Deploy system on PlanetLab or Harvard
6. Dec - Week 2: Collect results and stats
7. Dec - Week 3: Write-up our results

References

- [1] Hadoop: A distributed computing platform. <http://lucene.apache.org/hadoop/about.html>.
- [2] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementation*, December 2004.
- [3] Howard Gobioff, Shun-Tak Leung, and Sanjay Ghemawat. The google file system. In *Symposium on Operating System Principles*, October 2003.
- [4] Ian Rose, Rohan Murty, Peter Pietzuch, Jonathan Ledlie, Mema Roussopoulos, and Matt Welsh. Cobra: Content-based filtering and aggregation of blogs and rss feeds.
- [5] Technorati. <http://www.technorati.com/>.
- [6] David Sifry. The state of the blogosphere. <http://www.sifry.com/alerts/archives/000436.html>.
- [7] Françoise Fabret, H. Arno Jacobsen, François Llirbat, João Pereira, Kenneth A. Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2):115–126, 2001.
- [8] Nutch: open source web-search software. <http://lucene.apache.org/nutch/about.html>.
- [9] Blogdigger. <http://www.blogdigger.com/>.
- [10] Mozdex. <http://www.mozdex.com/>.
- [11] N. Glance, M. Hurst, and T. Tomokiyo. Blogpulse: Automated trend discovery for weblogs. In *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, 2004.
- [12] Blogpulse. <http://www.blogpulse.com>.
- [13] Universal feed parser. <http://www.feedparser.org/>.