Computational Challenges in the Analysis of Large, Sparse, Spatiotemporal Data

Ian Kelley
Information School
University of Washington
Seattle, U.S.A.
ikelley@uw.edu

Joshua Blumenstock
Information School
University of Washington
Seattle, U.S.A.
joshblum@uw.edu

ABSTRACT

The pervasive sources of data in today's networked computing environment provide many innovative opportunities, from mining patterns of individual behavior, to enabling data-intensive approaches for scientific discovery, to supporting new kinds of personal interactions and experiences. Passively collected metadata can also be mined for a variety of social analysis. However, due to the vast size and diversity of these data resources, they can pose serious computational challenges to researchers and analysts.

This paper highlights several of the key challenges involved in efficiently collecting, storing, and analyzing datasets consisting of millions of sparse files with spatial, temporal, and network features. We focus on the computational issues faced in analyzing Call Detail Records (CDRs), the metadata (i.e., log files) passively collected by mobile phone operators about transactions on their telecommunications networks. CDRs and related data provide a rich foundation for research in fields ranging from anthropology and sociology to electrical engineering and urban planning. After describing the data and its challenges, we present our current framework for computational analysis, and discuss opportunities for future work.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; K.4.0 [**Computers and Society**]: General

Keywords

Call Detail Records; CDR; Computational Social Science; Spatiotemporal; MapReduce

1. INTRODUCTION

As "big data" become increasingly ubiquitous, new computational paradigms emerged for processing its large volumes of heterogeneous and unstructured data. Parsing, transforming, and ultimately analyzing these datasets can be challenging. The data can often be too large for traditional relational databases, and the computational and memory requirements for processing it are often beyond the resource capabilities of even the most high-end machines. One way around this problem is to split analysis into small and discrete elements that can be run as independent jobs in a distributed computing environment.

Computing paradigms such as MapReduce [5] evolved directly out of efforts to tackle this problem and provide ways of chaining data-processing tasks together and aggregating their results. Since its release in 2007 as an open-source implementation of MapReduce, Apache's Hadoop[1] has developed into a generic architecture for task provisioning and data analysis, with a variety of middleware and software packages providing additional functionality and simplified programming interfaces. The Hadoop ecosystem, and its related projects, have been very successful in creating a scalable data processing framework for iterative data querying, while concurrently reducing program complexity and development time.

Certain aspects of the MapReduce model match very closely to the data analysis tasks we are undertaking on Call Detail Records (CDRs), especially queries that equate to simple `GROUP BY` statements and read data from only local storage or fast networks. However, for more complex and deeper analysis of the data, such as machine learning problems that can involve spatial, temporal, and network features of the data, other tools and middleware infrastructures are needed.

This goal of this paper is to two-fold, first to discuss our efforts to find and deploy efficient methods and infrastructures for analyzing large-scale spatiotemporal network data, and second to facilitate a broad discussion about future trends in data processing methods that include the integration of additional data sources to form more complex analysis pipelines.

The remainder of the paper is organized as follows: section 2 gives the motivation for analyzing CDRs and the kinds of information they can provide; section 3 discusses the data and processing challenges these types of records pose; section 4 describes key computational infrastructures used for this type of data mining and analysis; section 5 outlines our current data processing workflow; section 6 provides insight into the limitations of current architectures and our future directions, and section 7 concludes.

[1] https://hadoop.apache.org

DIDC'14, June 23, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2913-2/14/06 ...$15.00.
http://dx.doi.org/10.1145/2608020.2608025.

2. DATA AND APPLICATIONS

A growing number of computing applications produce large and sparse datasets with spatial, temporal, and network features. Examples include Twitter, where an individual's post messages can have timestamps and location information [4]; Facebook, where a great deal of activity on the social graph can be located in space and time [6]; and several other common sources including search queries, web server logs. Such data are widely used in research literature, with applications in the social sciences, computer science and engineering, and several other disciplines.

2.1 Call Detail Records

Here, we focus on one common instance of such spatiotemporal network data: Call Detail Records (CDRs). CDRs make a good use-case for exploring the computational challenges that arise in the analysis of large, sparse, spatiotemporal data, as they can provide a very informative source of about individuals and populations, particularly when spanning a long time frame and combined with auxiliary data sources.

A typical mobile phone log record will contain metadata including: *an anonymized identifier of the transaction originator*, *an anonymized identifier of the transaction recipient*, *the date and time*, *the duration*, *the call type (e.g., voice and SMS*, *the nearest cell phone tower to the transaction originator*, *the nearest cell tower to the transaction recipient*, and any error codes or additional provider information, such as network statistics. Note that cell phone numbers and other private information are typically anonymized by operators prior to distribution, yet they retain unique hashes, which allows common CDR lines to be grouped and evaluated.

2.2 Representative Applications

One of the goals of our group is to mine Call Detail Records for insight into patterns of human behavior, such as migration [2] and ethnic segregation [2] to processes of social and economic development [1]. In order to conduct this analysis, we require efficient methods and infrastructures for computing on CDR data. This includes not only the application and creation of algorithms and data workflows that can effectively process the data, but also investigating, using, and pushing distributed platforms in directions that facilitate such investigations.

One powerful feature of CDRs is that transactions can be aggregated and tracked through time. For example, the location and frequency of both collective and individual calls can be combined into daily, weekly, or monthly aggregates, and then compared against other similar datasets (e.g., individual vs collective sums, and similarly individuals vs other individuals). Through comparing different features of individual and collective behavior, "phone user" categories can be created and then tracked over time. These types of analysis can be useful for tracking and comparing many phenomenon, ranging from studying time-series physical mobility and migration to the effect of marketing strategies on product update.

Simple, concrete operations that are commonly performed on the data include:

- Simple `GROUP BY` queries, e.g., to determine the most visited location by an individual i in an arbitrary window $[t_m, t_n]$.
- Repeated calculation of graph metrics, e.g., to discretize the structure of the continuously evolving network into a large number of snapshots, whereby algorithms like PageRank and degree distributions can be quickly computed for each snapshot.
- More sophisticated metrics of individual mobility that go beyond observing existing data and require millions of repeated computations on small subsets of 100–1000 entries.
- New methods to simulate data that reflects the data generating process of the original data, but that alleviates potential concerns regarding privacy and confidentiality.
- Markov chain Monte Carlo and Bayesian methods for sampling and optimization.

3. DATA PROCESSING CHALLENGES

The analysis of large-scale spatiotemporal network data creates several challenges that, in our experience, are not fully addressed by any single computational framework. Before discussing the middleware commonly used for analyzing these datasets, we outline here a few of the issues we have encountered during our processing of CDRs.

3.1 Quantity and Size

Due to the large size and quantity of the input files, it is difficult to efficiently manage the data storage and computational resources needed to effectively analyze these records. For example, although a year of phone records from one large provider in a moderately sized developing nation might be considered a relatively small dataset, it can easily represent millions of input files, terabytes of data, and be represented by in several heterogeneous data formats.

Unfortunately, even the data obtained from the same provider does not come in a single homogeneous form. Data is collected in a time-series and stored as it is created. It is therefore logged and archived in whatever format is prevalent at the time. As phone networks expand, the infrastructure is often upgraded, meaning new hardware is installed and new data formats are written to log files, while some errors are fixed and new bugs appear. The end-result of this process is a heterogeneous set of metadata files, with varying levels of internal inconsistencies.

For example, a raw dataset spanning roughly a year of CDR data from a large provider in a developing country of around 50 million people might be on the order of 10 TB, while another dataset spanning many more years but from a country with a fraction of the population might be an order of magnitude smaller. This is because the number of input files and log size depends heavily on the pervasiveness, maturity, and use of the phone network, since every transaction generates at least one line in a log file. The size and density of individual log files also can vary greatly, ranging from dozens of large data dumps to millions of smaller files. Fortunately, these primary input files are usually in a plain text format, enabling them to be efficiently compressed (over 85%) for long-term storage after they are pre-processed into a uniform file format.

3.2 Data "Munging"

The process of transforming raw data into a structured, analyzable format is described in §5 and outlined in Figure 3.

This process involves first organizing the data into a coherent structure so that it can be parsed, a step that creates a second set of primary data with no loss of information. After this stage, the data can be prepped for statistical analysis, a process that involves cleaning and regrouped it into discrete categories that map well to the needed input for data processing jobs (e.g., Hadoop).

Formatting the data so that it can run efficiently in Hadoop not only entails cleaning the data so only relevant columns are present, but also grouping it into larger input data files. This is partially due to the Hadoop Distributed File System's (HDFS) [7] inefficiencies when storing and processing large numbers of small files. In this process, columns are pruned to contain only the relevant fields, call records are paired, cell tower location information is integrated, anomalies are identified, headers and other "noise" are removed, and the data is grouped mostly into aggregates (e.g., monthly files). After this step is complete, the data is ready to be staged to the HDFS, where it is replicated to several nodes for compute efficiency, tripling its footprint.

Data captured at scale by sensors has many records that are duplicated, blank, inaccurate, ill-formatted, or garbled. Within Call Detail Records, there are two main problems: the first is missing data, where one or more call record types are missing for a given day, week, or month; the second is "dirty" data, where there are anomalies in the logs. Anomalies are classified as when we have records for the day, but some critical information is either missing from some of the individual call record lines (e.g., phone numbers or tower information), ill formatted, or otherwise misleading.

For example, in one dataset, approximately 20% of all the data within the time frame was completely missing (e.g., a week of log files was missing in the middle of month A, while several days of a particular log type where missing in month B). In the data that is available, tens of millions of anomalies were identified in the CDR files. These were cases where MSSIDNs (i.e., phone numbers) were not properly formatted (e.g., they contained A-Z characters) or were missing completely, or no tower information was available, or several other identified errors.

In addition to issues with the data itself, necessary auxiliary input data can also be "dirty." For example, information about the network infrastructure itself can be missing, making it difficult to get complete lists of tower locations and thereby pinpoint GPS coordinates. These issues of missing and incorrect data must be managed, either through excluding those records, if they are very few and insignificant, or alternatively, building models to interpolate missing data from what information is available.

3.3 Sparsity

One common use of CDRs is to approximate the location of an individual at a given point in time. This information can be found by searching for the cell phone tower that a phone connected to at a moment close to the requested time. However, despite their being a large number of aggregate logs during a particular time, individual data can be very "sparse." This is primarily because for the majority of the time during a given day, most individuals are not actively making a phone call (and therefore not providing any location metadata). This creates a problem when trying to determine individual activity within a small time-window, with aggregates (e.g., individual daily averages, or conversely, collective hourly averages) much easier to obtain.

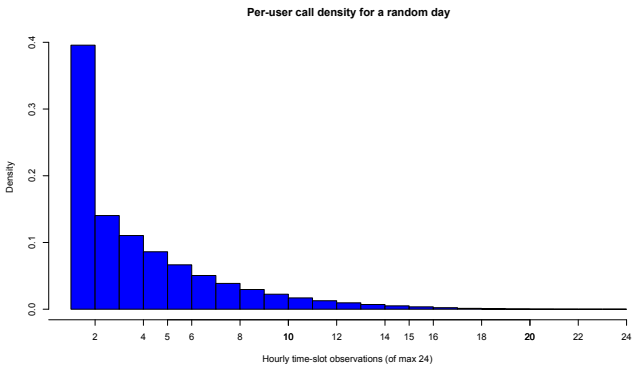

Figure 1: Number of unique hours of activity for subscribers on a single day

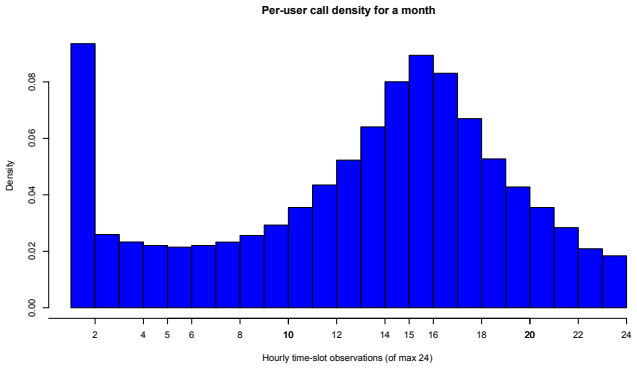

Figure 2: Number of unique days of activity for subscribers in a single month

Figure 1 gives an indication of one aspect of this sparsity, showing the distribution of the total number of active hours in a day when activity is observed for individuals. This includes all phone-based transactions such as calls and SMSes. As can be seen from the figure, most subscribers had very few distinct hours with any type of phone network connection. This is either due to a low frequency of calls, or the clustering of all activity into a small core-set of hours. In either scenario, the figures shows that even at this relatively high level of granularity (data grouped into hourly buckets), the mean number of hours used per day is only 4.2 (with a standard deviation of 3.31). Thus, there is infrequent information gathering for most users, as their daily behavior lacks intensive telecommunications activity. This makes determining and tracking location on this level of detail (hourly) a difficult task.

This leaves a very sparse dataset that makes granular mobility and usage analysis on a day-basis difficult. On a monthly-basis, the data becomes less sparse, while concurrently less informative. Figure 2 shows how the same analysis distributes over a month, with a mean of 13.3 active hours

and a standard deviation of 6.23. This can be interpreted to mean that most people never placed calls outside of the same core (yet not necessarily sequential) 13 hour time period, even when taking every transaction during the month into account. However, even though this information is less informative than the daily call logs, it can become useful in larger analysis types that incorporate other data sources [3].

4. COMPUTATIONAL MIDDLEWARE

The data storage and processing architecture in the Apache Hadoop ecosystem provides many powerful tools for data analysis. At its most fundamental level, Hadoop segments large computational problems into discrete chunks that can be processed within the resources available on a single computational node. These nodes are then clustered together to perform aggregate tasks and provide a tightly-coupled and scalable processing environment. The core Hadoop middleware presents an integrated system for data storage and processing, with the Hadoop Distributed File System (HDFS) providing the core data infrastructure, while Apache Yarn (Yet Another Resource Negotiator) coordinates job placement and result collection, centrally managing a (generally fixed) set of processing nodes.

Slight deviations from this integrated model are possible through, for example, the replacement of reliance on prestaged HDFS files with more robust input data streaming directly from highly-scalable online storage systems (e.g., Amazon's Simple Storage Service (S3)). However, in these cases, some of the same limitations apply, such as prior knowledge of the location of input data that is readily available on the designated storage cloud. Additionally, the Hadoop infrastructure performs best with relatively large (e.g., $\geq$128 MB) files, with significant performance degradation when processing is performed on numerous small files.

Hadoop's MapReduce engine provides mechanisms for partitioning datasets and performing data scanning and `GROUP BY` queries. This is very useful when calculating summations and generating different views of monolithic data sets, but it quickly becomes limiting when attempting the equivalent of an SQL join operation or performing iterative calculations such as PageRank or running machine learning algorithms. To facilitate these types of analysis, numerous complimentary middleware packages have been developed, such as Hive[2], Pig[3], Spark[4], and Shark[5].

5. CURRENT DATA PROCESSING PIPELINE

To analyze Call Detail Record input files, significant preprocessing and re-formatting is required before the data can even be deployed into a comprehensive computing infrastructure. Figure 3 depicts a typical data processing pipeline. Firstly, data acquisition is a barrier, as CDRs are not freely available and contain extremely private and sensitive information. Once data-sharing agreements have been negotiated, the sheer size of the data is a problem for transport, as are as the security implications and potential network limitations of transferring the data from its originating location.[6]

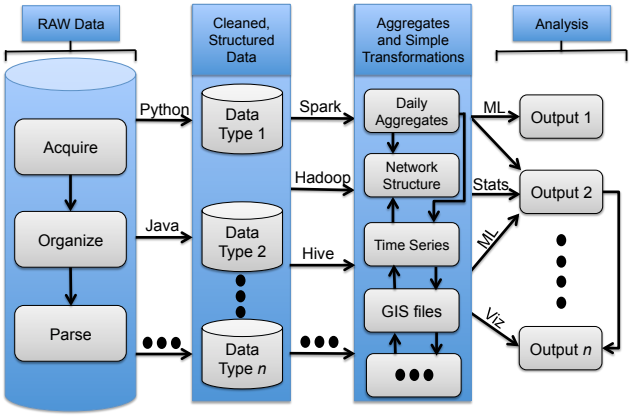

Figure 3: Data Processing Workflow.

Once the CDR files are obtained, they need to be compiled into a coherent structure, a process described in §3.2. This is often done with an array of scripts that parse through the input files and formats, binning found files into their new locations on secure disk.

To facilitate our data processing needs, our lab setup a five node cluster, where each node is equipped with 132 GB of RAM and 4 TB of HDFS storage. For our current analysis of CDR files, we are making use of MapReduce, Spark and Hive, with plans to test Shark and Mahout capabilities in the near future. Basic MapReduce has been useful for calculating many different metrics, such as geographic Center of Gravity estimations, network usage statistics, individual user behavior and call patterns, and robust error detection and data cleaning.

Spark has been leveraged for its ability to prove large "shared memory" and caching that facilitates iterative calculations. To-date, Spark has been used to compute indegree, outdegree, total degree, PageRank, connected components, and triadic closure, as well as isolating CDRs that fall within a certain geographic radius threshold corresponding to an event (e.g., an earthquake). Lastly, Hive has been used to create *ad hoc* queries against the dataset (e.g., selects and joins), providing an easy-to-use SQL scripting interface that is then translated into a workflow of MapReduce jobs. More details about the different types of current and future analysis are given in §2.2.

Once the data has been processed, and the relevant information created and extracted, it is generally in a small-enough form that it can be encrypted for transfer to external partners or imported into `R` on our servers for further analysis and visualization. Within `R`, the output data can be easily grouped, sorted, subsetted, analyzed, and plotted. For visualizations that benefit from location information, Geographic Information Systems (GIS) data can be imported, enabling data to be accurately plotted on country maps ei-

[2]http://hive.apache.org
[3]http://pig.apache.org
[4]http://spark.apache.org
[5]http://shark.cs.berkeley.edu

[6]In the case of many of our CDR files, they were located on tape backup in developing countries. This necessitated that the data first be pulled from tape archives and be copied to hard drives, before it could be physically flown on new drives to our lab.

ther as individual points or as color-coded aggregates within Voronoi cells or other regional districting.

6. LIMITATIONS AND FUTURE DIRECTIONS

The Hadoop data-processing ecosystem provides a useful environment for analyzing large datasets, allowing input data that cannot fit within the computational or memory limitations of single machines to be relatively easily distributed for analysis. However, its core middleware becomes inefficient when large numbers of files are encountered or the computational infrastructure is not highly centralized and actively managed. The architecture and its underlying filesystem are also tightly coupled, making it best suited to run on dedicated hardware with fast interconnects, precluding the inclusion of distributed components like those leveraged by resource scavenging systems.

Data processing can sometimes be moved to commercial Cloud offerings to lessen infrastructure setup and maintenance costs, but such solutions often do not fit with the security restrictions placed on sensitive data or the financial structure of research grants. Additionally, contractual obligations to CDR providers can require that data remain off the cloud and that all computations occur on local networks. In addition to the infrastructure setup and maintenance overhead associated with running a data-processing infrastructure, there is also a learning curve associated with the effective use the software middleware layers, many of which are in a constant state of development, and feel very "alpha" in their maturity.

Coupled with the need for a more robust data-processing environment that can help to eliminate some of the steps shown in Figure 3, or at least allow them to be integrated into a single environment, is the need to develop methods to find and inject auxiliary input data into data analysis pipelines. This is essential for (semi-)automated and *ad hoc* investigations that include other "big data" sources, especially if one wants to integrate real-time data streams into on-going calculations.

Current data investigations are just the beginning of what can be tried, with the most exciting investigations to be performed when multiple heterogeneous data sources can be merged together to create large meta-analysis projects. However, even small time-series datasets can require extensive storage and computational resources to analyze, requiring weeks to compute on a single machine and several days on a small cluster. Datasets are becoming larger and more pervasive; the question isn't if they will be analyzed, but rather if analysis can be performed easily using the current data processing architectures or if new systems will be developed to meet this challenge.

7. CONCLUSION

Many classes of data analysis problems involve very large and sparse graphs with spatial and temporal attributes. In this paper, we have introduced some of the complexities in analyzing Call Detail Records (CDRs). These records contain data that has been passively collected through the daily use of mobile phone networks, such as call and location information. This type of data is very rich in its features and utility and can be leveraged by a variety of data investigations due to its multifaceted use across research domains. However, the data itself is very sparse, and often "dirty" when first received, making any direct integration problematic and necessitating much storage and computationally-intensive pre-processing. Additionally, due to size and privacy considerations of the data, it is difficult to share or use the data without the careful extraction or obfuscation of key information. This process must be executed in such a way as to minimize exposure of sensitive information which can often still be gleaned even from anonymized datasets.

In this paper we have discussed some of the issues that arise when analyzing these datasets, and we have reflected on the computational and storage environments that currently represent today's best tools for this type of investigation. We have also shown how the current data storage ecosystem in Hadoop is somewhat ill-suited for certain aspects of our data analysis goals, requiring us to rearrange our data to "fit" efficiently into the HDFS storage architecture. The current methodology of pre-determining and pre-staging input data to a fixed and highly centralized file system for processing is at times obtuse and can be limiting when multiple (small) data sources need to be merged for analysis.

It is our hope that this paper has highlighted some computational and network needs that arise when analyzing large and sparse datasets, and will lead to new discussions and further innovation in computing paradigms that fulfill the data and computational challenges inherent in these types of data.

8. REFERENCES

[1] J. Blumenstock and N. Eagle. Mobile divides: Gender, socioeconomic status, and mobile phone use in rwanda. In *Proceedings of the 4th ACM/IEEE International Conference on Information and Communication Technologies and Development*, ICTD '10, pages 6:1–6:10, New York, NY, USA, 2010. ACM.

[2] J. Blumenstock and L. Fratamico. Social and spatial ethnic segregation: A framework for analyzing segregation with large-scale spatial network data. In *Proceedings of the 4th Annual Symposium on Computing for Development*, ACM DEV-4 '13, pages 11:1–11:10, New York, NY, USA, 2013. ACM.

[3] F. Calabrese, G. Di Lorenzo, and C. Ratti. Human mobility prediction based on individual and collective geographical preferences. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 312–317, Sept 2010.

[4] Z. Cheng, J. Caverlee, and K. Lee. You are where you tweet: A content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 759–768, New York, NY, USA, 2010. ACM.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[6] K. Lewis, J. Kaufman, M. Gonzalez, A. Wimmer, and N. Christakis. Tastes, ties, and time: A new social network dataset using facebook.com. *Social Networks*, 30(4):330 – 342, 2008.

[7] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10, May 2010.