Design Principles for Effective Knowledge Discovery from Big Data

Edmon Begoli, James Horey

Computational Sciences & Engineering Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
{begolie, horeyjl}@ornl.gov

***Abstract*— Big data phenomenon refers to the practice of collection and processing of very large data sets and associated systems and algorithms used to analyze these massive datasets. Architectures for big data usually range across multiple machines and clusters, and they commonly consist of multiple special purpose sub-systems. Coupled with the knowledge discovery process, big data movement offers many unique opportunities for organizations to benefit (with respect to new insights, business optimizations, etc.). However, due to the difficulty of analyzing such large datasets, big data presents unique systems engineering and architectural challenges. In this paper, we present three system design principles that can inform organizations on effective analytic and data collection processes, system organization, and data dissemination practices. The principles presented derive from our own research and development experiences with big data problems from various federal agencies, and we illustrate each principle with our own experiences and recommendations.**

I. Introduction

Knowledge Discovery from Data (KDD) [1] refers to a set of activities designed to extract new knowledge from complex datasets. The KDD process is often interdisciplinary and spans computer science, statistics, visualization, and domain expertise (Figure 1). In recent years, large quantities of data have become increasingly available at significant volumes (petabytes or more). Such data have many sources including online activities (social networking, social media), telecommunications (mobile computing,call statistics), scientific activities (simulations, experiments, environmental sensors), and the collation of traditional sources (forms, surveys). Consequently KDD has become strategically important for large business enterprises, government organizations, and research institutions. However, effectively producing knowledge from massive datasets remain challenging, especially for large enterprise organizations comprised of multiple sub-organizations (each of whom may have their own internal processes, formats, etc.). Effective KDD therefore requires effective organizational and technological practices to be in place. Specifically, knowledge discovery processes are comprised of:

- Data collection, storage and organization practices
- Understanding and effective application of the modern data analytic methods (including tools)
- Understanding of the problem domain and the nature, structure and meaning of the underlying data

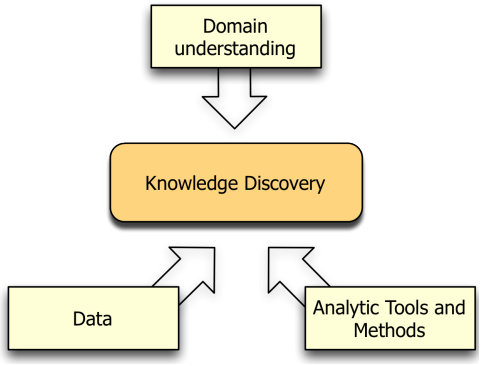

Fig. 1. Elements of the knowledge discovery process.

This paper outlines empirically derived principles for the establishment of effective architectures for knowledge discovery over big data.

II. Background

The principles described in this paper are derived from the experiences and outcomes of various real world projects at Oak Ridge National Laboratory (ORNL). ORNL collaborates with several state and federal agencies on big data projects [2]; typically ORNL receives data, is expected to analyze this data with domain experts, and to present the results via multiple avenues (i.e., web interface, reports, etc). Often the types of analysis to be performed are not explicitly defined and ORNL must explore a variety of potential methods. On occasion, ORNL is also asked to re-evaluate the current state of an agencies' internal big data architecture and strategy. In a recent example, an agency approached ORNL to develop new platforms for comprehensive and flexible data analysis. In addition to improving current analytic and business intelligence functions, the initiatives objective was to modernize, simplify, and streamline data collection, organization, and analysis of nationally significant datasets. The agency's existing methods were considered costly (using many proprietary components), somewhat antiquated (using older mainframe systems), and unable to meet rapidly increasing demand. The rest of the paper describes some key lessons learned in the development of our own infrastructure and the development of infrastructure for other agencies. We believe that the application of some

core principles can yield economical, comprehensive, flexible, and secure solutions for the federal government's big data needs.

III. Principles

Knowledge discovery, as any form of discovery, is serendipitous. Discovery of new knowledge might occur if the right factors are present and aligned. Factors such as intuition, acuteness, and probability of observation are difficult to control. Others, such as comprehensibility and organization of data, its layout, availability of proper tools, and domain expertise are controllable. Our design principles are largely concerned with maximizing the controllable factors and thereby enabling researchers to explore, analyze, and interact with data in as easy manner as possible.

A. Principle 1: Support a Variety of Analysis Methods

Knowledge discovery and modern data science employs methods from distributed programming, data mining, statistical analysis, machine learning, visualization, and human-computer interaction (amongst others). These methods often employ vastly different tools and techniques. For example, programmers may use Java to write distributed computation (i.e., Hadoop [3]), while statisticians may feel more comfortable using R, SAS, etc. Many analysts will interact with SQL during the lifetime of the application. Depending on the nature of the analysis, different tools and expertise may be applied at different times. Indeed, it has been our experience that it is better to support a variety of tools rather than forcing users to use a limited set of tools (that many may be unfamiliar with, etc.). The architecture must therefore support a variety of methods and analysis environments. In the following, we detail some specific methods that are frequently used in our projects.

1) Statistical Analysis: Statistical analysis is concerned with both summarizing large datasets (i.e., average, min, etc.) and in defining models for prediction. In our experience, such analysis is often the first step in understanding the data. However most statistical tools (i.e., R, SAS) prefer to compute over numerical and categorical data organized in a tabular, column-oriented fashion. This often requires an extensive parsing and organization step, especially for unstructured datasets. In our current systems, we provide a variety of statistical tools, including column-oriented relational databases (i.e., SQL), R, and Python (i.e., NumPy, SciPy).

2) Data Mining and Machine Learning: Data mining is concerned with automatically discovering useful models and patterns in large datasets. Data mining consists of a large set of statistical and programmatic techniques (i.e., clustering, topic discovery, etc.). Machine learning employs both data mining and statistical techniques (amongst others) with the explicit goal of enabling machines to understand a set of data. Techniques may be supervised (i.e. with human assistance) or unsupervised. Often these techniques are used in conjunction with statistical methods for elucidating complex relationships (non-linear). For these tasks, we provide tools including MADLib [4](in conjunction with EMC Greenplum [5]) and the Hadoop Mahout library.

3) Visualization and Visual Analysis: Visual analytics is an emerging field in which massive datasets are presented to users in visually compelling ways with the hope that users will be able to discover interesting relationships. Visual analytics requires generating many visualizations (often interactively) across many datasets. For our large visualization systems [6] and web-based interactive systems [7], we employ a combination of high bandwidth file systems and pre-computed analytic artifacts stored as visualization friendly objects such as JSON representations.

B. Principle 2: One Size Does Not Fit All

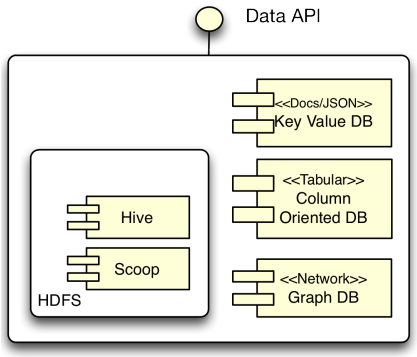

Fig. 2. Knowledge Discovery Architecture Reference Implementation - KD Fabric

In addition to providing a variety of data analysis methods, a comprehensive KDD architecture must supply a means of storing and processing the data at all stages of the pipeline (from initial ingest to serving results). While a single storage mechanism may suffice for small data volumes (i.e., local filesystem), this is more problematic for large-scale data analysis. Historically many organizations have relied on large relational databases to accomplish this. However, we argue that different types of analysis and the intermediate data structures required by these (e.g. graphs for social network analysis) call for specialized data management systems (Figure 2). Others have also recognized that the time of the single style database that fits all needs is gone [8].

1) Data Preparation and Batch Analytics: Data preparation is the first step in the analytics pipeline (Figure 3). At this stage, the data may contain errors, missing values, and is often in an unusable format (i.e., a compressed binary format). In our experience, Hadoop is an ideal tool for this stage. Hadoop is a collection of Java-based open source software inspired by Google's BigTable [9], Google File System [10] and MapReduce [11]. It includes a MapReduce component (for distributed computation) and a scalable storage component, Hadoop File System (HDFS), that can often replace costly SAN devices. Hadoop sub-projects such as Hive and HBase offer additional data management solutions for storing structured and semi-structured data sets. In our systems we rely on HDFS as a

data landing platform and use Hive as our batch-oriented data warehouse.

2) Processing Structured Data: Often the product of the data preparation stage is a set of highly structured, relational data. Although Hadoop can process such data (via Hive), we have found distributed analytic databases [12] to be useful for storing and analyzing such data. These databases (i.e., EMC Greenplum, HP Vertica [13]) are highly optimized for large reads, aggregations, and statistical processing. They often store data in a column-oriented fashion (vs row-oriented) and distribute data over multiple machines to scale large sizes. Because these systems employ SQL for querying, these databases can also serve as backends for mainstream Business Intelligence software (and thus simplifying visual interaction).

3) Processing Semi-structured Data: Not all data can be easily modeled using relational techniques. For example, hierarchical documents, graphs, and geospatial data. Such data is extremely useful for social network analysis, natural language processing, and semantic web analysis. We provide HBase [14] and Cassandra [15] for hierarchical, key-value data organization. For graph analysis, we employ both open-source tools (e.g., Neo4j [16]) and proprietary hardware solutions (e.g., Cray's uRiKa platform [17]). Finally, for geospatial data we employ open-source tools (e.g., PostGIS, GeoTools) and proprietary tools (e.g., ESRI software).

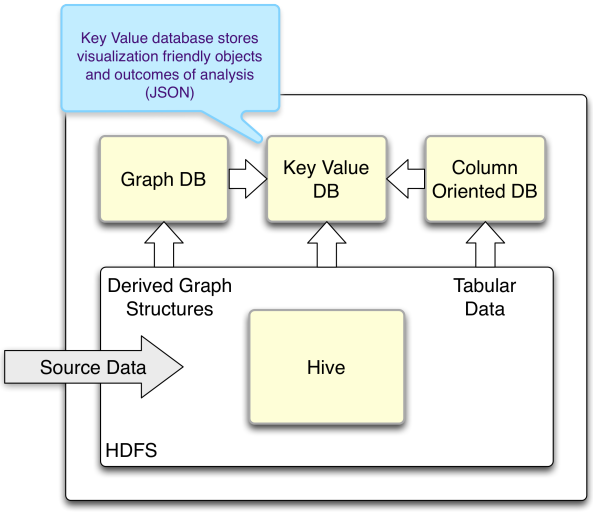

Fig. 3. Transformation of data through specialized datastores.

C. Principle 3: Make Data Accessible

While the previous principles address data analysis and organization, the final principle addresses the end product (which is often highly summarized data and insights). Specifically, it has been our experience that making the results accessible and easy to understand is paramount. Three approaches we have used to accomplish this are using open, popular standards, adoption of lightweight, web-oriented architectures, and exposing results via a web-based API.

1) Use Open, Popular Standards: Presenting results to users involves a complex set of software (databases, application servers, and web interfaces). While this space has been involving quickly, it has been our goal to use well supported frameworks. For example employing open-source databases (i.e., PostgreSQL), application servers (i.e., Java, Node.js), and nimble Javascript frameworks (jQuery, etc.). Each of these systems should communicate using standard protocols (i.e., REST, ODBC/JDBC, JSON).

2) Use Lightweight Architectures: Users demand rich, interactive experiences when using web interfaces. While traditional methods of building web services suffice (i.e., J2EE), our experience has been that new lightweight architectures (i.e., Rails [18], Django [19], Node.js [20]) simplify the construction of web applications. This, in turn, means that we can create rich applications quickly and on demand. Since many of these applications rely on open-source tools, we can be confident that our applications will be able to run on a variety of platforms.

3) Expose Results using an API: While downloading results in a document makes sense for many scenarios, users now demand more flexible methods to interact with data systems. Specifically users now expect rich web-enabled APIs to download, interact, and visualize data. Actual processing may happen on the server or in the client (via Javascript). By exposing data via an API, it becomes easier for disparate systems to interact and enables users to create additional analysis tools. In that regard, we were inspired by Yahoo's BOSS Web Services [21] as a good model of how to expose rich web service APIs. Currently, we offer data feeds as OData [22] compliant web services that represent either enumerable data entities or results of the analytical post-processing. For each instance of an API we provide interface documentation and extensive examples. As part of this effort, we have learned that, in order to foster rapid adoption of API based architecture, it is paramount to offer extensive documentation describing both API usage and the underlying datasets.

IV. Implementation

We have integrated these design three principles into our own knowledge discovery infrastructure. We use this infrastructure to address various federal agency needs and to serve as a reference implementation for agencies that may want their own infrastructure. When ORNL executes work for others, the data provided for us typically arrive in a raw format (often from their own internal systems) that's not necessarily amenable to analysis (i.e., Cobol EBCDIC). Depending on the data volume, the data is either transmitted over the internet (using a secure channel) or via encrypted NAS drives. We process this data using Hadoop and may perform initial analysis using Hive. For data that requires additional structured analysis (i.e., SQL), we place the data into a distributed database. For other data formats, we use our cloud computing platform (CloudStack [23]) to instantiate necessary datastores such as Cassandra. Other cloud platform also enables us to create virtual machines for ad-hoc analysis,

parsing, and visualization. In accordance to the principles outlined, all datasets are exposed via RESTful APIs using the OData standard.

V. Future Work

Although our infrastructure is used for real-world applications, we treat these systems as a research platform and expect it to continuously evolve as the state-of-the-art advances. We have developed our knowledge discovery principles during the course of implementing these applications and standing up our own systems. However, there is still much to do and many open architectural questions. Some immediate ones include:

- How do we take advantage of cloud computing to instantiate big data services in an optimal manner (i.e., to reduce cost, maximize performance)?
- How do we automate and formalize the process of instantiating the entire data analysis pipeline?
- How do we track provenance and handle security as the data flows through the analysis pipeline?
- What additional storage and analysis systems do we need? For example, do we need a Hadoop-for-graphs? What is the role of in-memory systems?

We are currently examining these questions at ORNL while continuing to further refine our existing systems.

VI. Conclusions

The big data movement has energized the software architecture world, and it has introduced complex, interesting questions for the community. As organizations continue to collect more data at this scale, formalizing the process of big data analysis will become paramount. In this paper, we introduced several principles that we believe can guide organizations into developing a sound, useful, and flexible data analysis pipeline. These principles are a result of experience and lessons learned from our own big data applications at ORNL. We have instantiated these principles in our own infrastructure and have found the principles to be useful guides.

References

[1] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in knowledge discovery and data mining*, 1996.

[2] T. Kalil, "Fact sheet: Big data across the federal government," *Office of Science and Technology Policy, Executive Office of the President*, March 2012.

[3] T. White, *Hadoop: The definitive guide*. Yahoo Press, 2010.

[4] J. Hellerstein, C. Ré, F. Schoppmann, Z. Wang, E. Fratkin, A. Gorajek, K. Ng, C. Welton, X. Feng, K. Li, *et al.*, "The MADlib analytics library or MAD skills, the SQL," 2012.

[5] "Greenplum database community edition." [Online]. Available: http://www.greenplum.com/products/community-edition

[6] A. Sorokine, J. Daniel, and C. Liu, "Parallel visualization for GIS applications," in *Proceedings GeoComputation*, 2005.

[7] "d3 - JavaScript based visualization library." [Online]. Available: http://mbostock.github.com/d3/

[8] M. Stonebraker and U. Cetintemel, "One size fits all: An idea whose time has come and gone," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, 2005, pp. 2–11.

[9] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers, "Big data: The next frontier for innovation, competition and productivity," *McKinsey Global Institute, May*, 2011.

[10] S. Ghemawat, H. Gobioff, and S. Leung, "The google file system," in *ACM SIGOPS Operating Systems Review*, vol. 37, 2003, pp. 29–43.

[11] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[12] D. Abadi, S. Madden, and N. Hachem, "Column-Stores vs. Row-Stores: how different are they really?" in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 967–980.

[13] "Hp vertica database community edition." [Online]. Available: http://vertica.com/community

[14] "Apache hbase," http://hbase.apache.org.

[15] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, Apr. 2010.

[16] "Neo4j graph database." [Online]. Available: http://neo4j.org

[17] "Cray yarcdata urika appliance." [Online]. Available: http://www.cray.com/products/Urika.aspx

[18] "Ruby on rails," http://rubyonrails.org.

[19] "Django," http://www.djangoproject.com.

[20] "Node.js," http://nodejs.org.

[21] "Yahoo! search build your own service (BOSS)." [Online]. Available: http://developer.yahoo.com/search/boss/

[22] "Open data protocol." [Online]. Available: http://www.odata.org

[23] "Cloudstack." [Online]. Available: http://www.cloud.com