What is Analytic Infrastructure and Why Should You Care?

Robert L Grossman
University of Illinois at Chicago
and Open Data Group
grossman@uic.edu

ABSTRACT

We define analytic infrastructure to be the services, applications, utilities and systems that are used for either preparing data for modeling, estimating models, validating models, scoring data, or related activities. For example, analytic infrastructure includes databases and data warehouses, statistical and data mining systems, scoring engines, grids and clouds. Note that, with this definition, analytic infrastructure does not need to be used exclusively for modeling but simply useful as part of the modeling process. In this article, we discuss the importance of analytic infrastructure and some of the standards that can be used to support analytic infrastructure. We also discuss some specialized analytic infrastructure applications and services, including applications that can manage very large datasets and build models over them and cloud based analytic services.

1. INTRODUCTION

If your data is small, your statistical model is simple, your only output is a report, and the analysis needs to be done just once, then there are quite a few statistical and data mining applications that will satisfy your requirements. On the other hand, if your data is large, your model is complicated, your output is not a report, but instead a statistical or data mining model that needs to be deployed into operational systems, or parts of the work need to be done more than once, then you might benefit by using some of the infrastructure components, services, applications and systems that have been developed over the last decade to support analytics.

In this article, we refer to these types of components, services and applications as analytic infrastructure. We will define analytic infrastructure a bit more precisely below. We discuss the importance of analytic infrastructure and some of the standards that can be used to support analytic infrastructure. We also discuss some specialized analytic infrastructure applications and services, including applications that can manage very large datasets and build models over them and cloud based analytic services.

2. DEFINING ANALYTIC INFRASTRUCTURE

The rest of this article will be clearer if we define some terms. Define *analytics* to be the process of using data to make decisions based upon models. By *models* we mean statistical, mathematical or data mining models that are derived from data ("empirically derived") using techniques that are generally accepted by the community ("statistically valid").

As is standard, we divide the process of building a model into several steps. We use the term *data preparation* or data shaping for the process that takes the available data and transforms it to produce the inputs to the models (features). We use the term *modeling* (in the narrow sense) or *model estimation* for the process that, given a dataset of features, computes the model, which usually involves estimating the parameters of the model. The application that produces the model is sometimes called a *model producer*. Finally, after the model is estimated, it is usually *validated*.

Once a model is produced, *model consumers* take a model and data to produce scores. Often this is called *scoring*. Finally, scores are often *post-processed* using business rules to produce alerts or recommendations, or used as an input for a model or for additional data shaping.

In this article, we define *analytic infrastructure* to be the applications, services, utilities and systems that are used for either preparing data for modeling, estimating models, validating models, scoring, or related activities. For example, analytic infrastructure includes databases and data warehouses, statistical and data mining systems, scoring engines, grids and clouds. Note that with this definition analytic infrastructure does not need to be used exclusively for modeling but simply useful as part of the modeling process.

3. STANDARDS FOR ANALYTIC INFRASTRUCTURE

For data mining applications that consist of several services or components, standards provide agreed upon interfaces for accessing the services or components. This is particularly important if you ever need to change one or more services or cmponents in the application.

Standards for describing statistical and data mining models are also important. These types of standards provide an application and system independent format for persisting and interchanging models. Data mining projects often generate several models and providing life cycle management for these models is easier if there is a standard format for models. The models in this format can then be persisted in a repository or otherwise managed. Having a standard format for models is also useful for business continuity purposes and for compliance related requirements.

Broadly speaking, there are two basic approaches that have

Step	**Inputs**	**Outputs**
Preprocessing	dataset (data fields)	dataset of features
Modeling	dataset of features	model
Scoring	dataset (data fields), model	scores
Postprocessing	scores	actions

Table 1: The table contains some of the steps when building and deploying analytic models that are directly relevant to analytic infrastructure.

been used to capture an analytic process.

The first approach is to think of the analytic process as a workflow and to think of the workflow as being described as a directed acyclic graph (DAG), where the nodes are arbitrary code and the edges indicate that the output of one program is used as an input to another program. The advantage of this approach is that it is very general, easy to understand, and relatively simple to process. A disadvantage is that from an archival point of view, over time computer code can present difficulties: the necessary environment to execute it may not be available for example.

Another approach is to agree via a standardization process on the meaning of the necessary elements to define a statistical model and then to specify these in the description of the model. For example, a regression tree model can specify that a regression tree consists of nodes and nodes are associated with predicates, etc. Then to describe a specific model, one just needs to specify the assignments for all the elements in the model. The advantage of this approach is that the description of a model is quite explicit, and, in principle, will be just as easy to understand ten years later as it was when it was created. Another advantage is that updating a model does not require testing new code, but instead just requires reading new parameters for the model. This approach can dramatically reduce the time to deploy new models.

The second approach is the approach taken by the Predictive Model Markup Language (PMML) [5], which defines several hundred XML elements that can be used to describe most of the common statistical and data mining models.

4. THREE IMPORTANT INTERFACES

Table 1 describes some of the steps in the data mining process that are most relevant for analytic infrastructure.

Interfaces for producers and consumers of models. Perhaps the most important interface in analytics is the interface between components in the analytic infrastructure that produce models, such as statistical packages, which have a human in the loop, and components in the analytic infrastructure that score data using models and often reside in operational environments. Recall that the former are examples of model producers, while the latter are examples of model consumers. The Predictive Model Markup Language or PMML is a widely deployed XML standard for describing statistical and data mining models using XML so that model producers and model consumers can exchange models in an application independent fashion.

Many applications now support PMML. By using these applications, it is possible to build an open, modular standards based environment for analytics. With this type of open analytic environment, it is quicker and less labor-intensive to deploy new analytic models and to refresh currently deployed models.

Interfaces for producers and consumers of features. Since Version 2.0 of PMML was released in 2001, PMML has included a rich enough set of transformations that data preprocessing can be described using PMML models. Using these transformations, it would be possible to use PMML to define an interface between analytic infrastructure components and services that produce features (such as data preprocessing components) and those that consume features (such as modeling applications). This is probably the second most important interface in analytics.

Interfaces for producers and consumers of scores. The current version of PMML is Version 4.0 and the PMML working group is now working on Version 4.1. One of the goals of Version 4.1 is to enable PMML to describe post-processing of scores. This would allow PMML to be used as an interface between analytic infrastructure components and services that produce scores (such as scoring engines) and those that consume scores (such as recommendation engines). This is probably the third most important interface in analytics.

Today, by using PMML to describe these interfaces, it is straightforward for analytic infrastructure components and services to run on different systems. For example, a modeler might use a statistical application to build a model, but scoring might be done in a cloud, or a cloud might be used for preprocessing the data to produce features for the modeler.

5. ANALYTIC INFRASTRUCTURE FOR WORKING WITH LARGE DATA

Today, it is still a challenge to build models over data that is too large to fit into a database. One solution is to use grids [7]. Today, grids have two main limitations. First, they were not designed for working with large data per se, but rather, to support virtual communities. Second, many programmers found the MPI-based programming model rather difficult to use.

An alternative approach that overcomes both of these limitations was outlined in a series of three Google technical reports [8], [6] and [3]. The architecture consists of a distributed storage system called the Google File System (GFS) [8] and a parallel programming framework called MapReduce [6]. GFS was designed to scale to clusters containing thousands of nodes and was optimized for appending and for reading data. Contrast this to a traditional relational database that is optimized to support safe writing of rows with an ACID semantics and usually supported by a single computer (perhaps with multiple cores) over a RAID file system.

A good way to describe MapReduce is through an example. Assume that GFS manages nodes i spanning several, perhaps many, racks. MapReduce assumes that the data consists of a key-value pair. Assume that node i stores web pages $p_{i,1}$, $p_{i,2}$, $p_{i,3}$, …, p_{i,i_j}. Assume also that web page p_i contains words $w_{j,1}$, $w_{j,2}$, $w_{j,3}$, …. A basic structure important in information retrieval is an inverted index, which is a data structure consisting of a word followed by a list of web pages

$$(w_1; p_{1,1}, p_{1,2}, p_{i,3}, \ldots)$$

$$(w_2; p_{2,1}, p_{2,2}, p_{2,3}, \ldots)$$

$$(w_3; p_{3,1}, p_{3,2}, p_{3,3}, \ldots)$$

with the properties:

1. The inverted index is sorted by the word w_j;
2. If a word w_j occurs in a web page p_i, then the web page p_i is on the list associated with the word w_j.

A mapping function processes each web page independently, on its local storage node, providing data parallelism. The mapping function emits multiple <key, value> pairs (<word, page_id> in this example) as the outputs. This is called the Map Phase.

A partition function $m(w)$, which given a word w, assigns a machine labeled with $m(w)$, is then used to send the outputs to multiple common locations for further processing. This second step is usually called the Shuffle Phase.

In the third step, the processor $m(w_i)$ sorts all the <key, value> pairs according to the key. (Note that there may be multiple keys sent to the same node, i.e., $m(w_i) = m(w_j)$.) Pairs with the same key (keyword in this example) are then merged together to generate a portion of the inverted index <w_i: $p_{x,y}$, ...)>. This is called the Reduce Phase.

To use MapReduce, a programmer simply defines a parser for input records (called a Record Reader) and a Map, Partition, Sort (or Comparison), and Reduce functions and the infrastructure takes care of the rest.

Since many applications need access to rows and columns of data (not just bytes of data provided by the GFS), a GFS-application called BigTable [3] that provides data services that scale to thousands of nodes was developed. BigTable is optimized for appending data and for reading data. Instead of the ACID requirements of traditional databases, BigTable chose an eventual consistency model.

Google's GFS, MapReduce and BigTable are proprietary and not generally available. Hadoop [11] is an Apache open source cloud that provides on-demand computing capacity and that generally follows the design described in the technical reports [8] and [6]. There is also an open source application called HBase that runs over Hadoop and generally follows the BigTable design described in [6].

Sector is another open source system designed for data intensive computing [13]. Sector was not developed following the design described in the Google technical reports, but instead was designed to manage and distribute large scientific datasets, especially over wide area high performance networks. One of the first Sector applications was the distribution of the 10+ TB Sloan Digital Sky Survey [10]. Sector is based upon a network protocol called UDT that is designed to be fair and friendly to other flows (including TCP flows), but to use all the otherwise available bandwidth in a wide area high performance network [9].

On top of the Sector Distributed File System is a parallel programming framework called Sphere that can invoke user defined functions (UDFs) over the data managed by Sector. From the perspective of Sphere's UDFs, MapReduce is simply the concatenation of three specific, but very important UDFs, namely Map, Shuffle and Reduce UDFs as described above. These specific UDFs are available with the Sector distribution, as well as several others.

As measured by the MalStone Benchmark [12], Sector is approximately twice as fast as Hadoop.

Table 2 contains a summary of GFS/MapReduce, Hadoop and Sector.

6. CLOUD-BASED ANALYTIC INFRASTRUCTURE SERVICES

Over the next several years, cloud-based services will begin to impact analytics significantly. Although there are security, compliance and policy issues to work out before it becomes common to use public clouds for analytics, I expect that these and related issues will all be worked out over the next several years.

Cloud-based services provide several advantages for analytics. Perhaps the most important is elastic capacity — if 25 processors are needed for one job for a single hour, then these can be used for just the single hour and no more. This ability of clouds to handle this type of surge capacity is important for many groups that do analytics. With the appropriate surge capacity provided by clouds, modelers can be more productive, and this can be accomplished in many cases without requiring any capital expense. (Third party clouds provide computing capacity that is an operating expense, not a capital expense.)

Packaging. One of the advantages that clouds instances provide is that all the required system software, application software, auxiliary files, and data can be integrated into a single machine image [1]. This can sometimes significantly simplify the process required to prepare data, build models, or score data.

Preparing data. For interactive exploration of small datasets, for preparing a single dataset for analysis, and related tasks, single workstations with desktop applications usually work just fine and are probably the easiest to use.

On the other hand, for the pipelined analysis of multiple datasets with the same pipeline, a cloud that provides on demand computing instances offers several advantages. By pipeline here, we mean a sequence or workflow of computations where the inputs of one computation are the outputs of one or more prior computations. These computations occur, for example, quite frequently in bioinformatics.

Once the pipeline is prepared and an appropriate computing instance is defined, handling new datasets in the future is quite simple. When a new dataset arrives, simply invoke a new computing instance, supply the dataset, and save the results of the analysis.

With this approach, computing instances can easily manage surge requirements that arise when multiple datasets must be processed at the same time. To handle such a surge, simply invoke a new computing instance for each dataset so that the datasets can be processed in parallel.

MapReduce for clouds that provide on-demand computing capacity has proven to be particularly effective for preparing data in which large numbers of relatively small records have to be prepared for modeling, such as web pages [6], transactional business data [2], and network packet data.

Building models. For very large datasets, several traditional statistical and data mining algorithms have been re-

Design Decision	Google's GFS, MapReduce, BigTable	Hadoop	Sector
data management	block-based file system	block-based file system	dividing data into segments & using native file system
protocol for message passing with system	TCP	TCP	Group Messaging Protocol
protocol for transferring data	TCP	TCP	UDP-Based Data Transport (UDT)
programming model	MapReduce	MapReduce	User defined functions, MapReduce
replication strategy	at the time of writing	at the time of writing	periodically
security	not mentioned	not yet	HIPAA capable
language	C++	Java	C++

Table 2: Some of the similarities and differences between Google's GFS/MapReduce, Hadoop and Sector.

formulated as MapReduce jobs, which enables Hadoop and similar systems to be used for computing statistical models on very large datasets (see, for example [4]).

MapReduce and similar programming models enable basic statistical models to be built on very large datasets with very little effort, something that was not so easy just a short time ago. For example, the technical report [2] describes how a statistical model to identify potentially compromised web sites was built on log files containing over 10 billion records on a 20 node cluster using just a few lines of MapReduce code.

Once a source of data is well understood, it is sometimes required to estimate models for different instantiations of the data. For example, this is a common scenario for some analytic models used in online systems. Here a separate model may be required for each visitor to the web site or each user of an online application. In this case, given data from the user, an on-demand computing instance can be invoked to build a visitor-specific or user-specific model.

Scoring models. As mentioned above, once a model has been estimated, it is usually straightforward to apply this model to a file or stream of data to produce scores (outputs of the model). To say it another way, it is usually easier to develop a model consumer than it is to develop a model producer.

Given a dataset and a model described in PMML, it is straightforward to create an on-demand computing instance that can score the data using the model. An architecture like this that i) separates model producers and model consumers and ii) scores data using on-demand computing instances provides three important advantages:

1. First, with this approach, the computing instance can be pre-configured so that essentially no knowledge is required by the system scoring the data other than the name of the pre-configured computing instance, the name of the data file, and the name of the PMML file. In contrast, estimating a statistically valid model using a statistical or data mining package often requires considerable expertise.

2. Second, by using on-demand computing instances for scoring, it is straightforward to invoke multiple computing instances so that multiple datasets can be scored at the same time (in other words to handle surges).

3. Third, it is also straightforward to score very large datasets by partitioning them and scoring each partition in parallel using on-demand computing instances. It is often not straightforward to estimate models by partitioning the data, but it is almost always easy to score large datasets by partitioning.

7. SUMMARY

In this article, we used the term analytic infrastructure to refer to the tools, services, applications and platforms that support the analtyic process.

Except for the most basic models, building models requires specialized software. For example, for many years, databases have been an important component of analytic infrastructure: both to manage data, as well as to help prepare data for modeling. More recently, a variety of other analytic tools and services are becoming more common, including scoring engines to deploy models, workflow systems for managing analytic tasks that need to be repeated, and cloud-based services for a variety of tasks, including data preparation, modeling, and scoring.

With the proper analytic infrastructure, models can be built more quickly, deployed more easily, and updated more efficiently. For these reasons, the proper analytic infrastructure is often the difference between a successful and unsuccessful analytic project.

8. REFERENCES

[1] Amazon. Amazon Web Services. http://aws.amazon.com, 2009.

[2] J. Andersen, C. Bennett, R. L. Grossman, D. Locke, J. Seidman, and S. Vejcik. Detecting sites of compromise in large collections of log files. *submitted for publication*, 2009.

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *OSDI'06: Seventh Symposium on Operating System Design and Implementation*, 2006.

[4] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-Reduce for machine learning on multicore. In *NIPS*, volume 19, 2007.

[5] Data Mining Group. Predictive Model Markup Language (pmml), version 4.0. http://www.dmg.org, 2009.

[6] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[7] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, California, 2004.

[8] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, New York, NY, USA, 2003. ACM.

[9] Y. Gu and R. L. Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks*, 51(7):1777—1799, 2007.

[10] Y. Gu, R. L. Grossman, A. Szalay, and A. Thakar. Distributing the sloan digital sky survey using udt and sector. In *Proceedings of e-Science 2006*, 2006.

[11] Hadoop. Welcome to Hadoop! http://hadoop.apache.org/core/, 2008.

[12] MalGen: A utility for generating synthetic site-entity log data for testing and benchmarking data intensive applications. http://code.google.com/p/malgen, 2009.

[13] Sector. http://sector.sourceforge.net, 2008.