Big Data – Web Originated Technology meets Television

Bhavan Gandhi and Sanjeev Mishra
ARRIS

Abstract

The development of Big Data technologies was a result of the need to harness the ever-growing zettabytes of data created and consumed on the Internet. Web content, social networking, and user generated content in the form of blogs, images, and videos are just a few examples of the nature of unstructured data that don't fit into the traditional relational database models (RDBMS). In a similar manner, the television eco-system is evolving beyond just the delivery of linear video content to the television. Video experiences are evolving into more complex systems that support delivery of linear and on-demand content on multiple portable devices, capable of local and network DVR, and even supporting targeted content recommendation and advertisement placement. Harnessing Big Data technologies from the Internet world and bringing it into the TV space allows for deeper understanding of user behavior and system performance. This paper provides an overview of Big Data Technologies, and it gives examples an architectural overview of how these technologies can be used in an operator's eco-system.

INTRODUCTION – WHAT IS BIG DATA?

Web Originated

Terms typically used to characterize Big Data are volume, velocity, and variety [1, 2, 3]. Volume refers to the sheer amount of data that currently exists and is being created continuously. Velocity is the speed with which this data can be processed and analyzed for a meaningful use. Variety is simply the different types of data that can be collected. From the perspective of the Internet, zettabytes (2^{70} bytes) of data are created and stored in the form of images, documents, tweets, videos, maps, social interactions, etc. This data has to be analyzed and classified with speed so that end users can find and retrieve relevant information to accomplish their tasks. Much of this data is either unstructured, or in cases where there is inherent structure, the variety of the data leads to the need to support multiple structures.

The advent of the Internet and the volume of data that needs to be managed, stored, and analyzed led to the emergence of modern day Big Data technologies. Yahoo and Google have developed breakthrough technology on processing Internet scale disparate data and storing the processed results on distributed commodity servers to enable high availability and scalability. The Google File System (GFS) [4] and Big Table [5] showcase early pioneering work that was developed for storing data at Google. This is the pre-cursor to some of the mainstream, open source big data related technologies that are being developed and evolved by the open source community like Apache. Apache HBase [6] and Apache Hadoop Distributed File System (HDFS) [7] are open source equivalents of Big Table and GFS respectively.

Given that companies like Google, Yahoo, and Facebook are storing petabytes (2^{50} bytes) of data, there is a need to access this vast repository of data to process the information and derive meaning quickly. MapReduce is a framework that works on top of HDFS and HBase; it allows for parallel access, incremental, and distributed processing of huge amounts of data to derive useful meaning that can be used by subsequent services or applications. Hive and Pig are higher-level languages that allow for programming or scripting of MapReduce jobs. Researchers at Facebook developed Hive to allow SQL-like queries against their Big Data sets [8, 9]. Pig was developed by Yahoo staff to

reduce the complexity of programming MapReduce jobs.

Big Data for Television

The television eco-system is migrating from dedicated hardware / software solutions to more flexible sets of services that allow customized and interactive experiences for end-consumers. The power of Big Data technologies from the web world can be harnessed to create measurable and customized experiences in the television space. This is especially true of experiences that are migrating from the single primary screen, in-home, linear viewing experiences to multi-screen, on-the-go, on-demand content consumption experiences. There are essentially three stakeholders that benefit from Big Data and corresponding data science: the operator, the content provider, and the end consumer. The operator benefits by having meaningful information for monitoring and optimizing system performance for capacity planning and measuring impact. The content provider can use this information to optimize and understand how their programming is being used by the end-consumer. Finally, the end consumers benefit from better interactive and targeted experiences that are more meaningful for them or their household.

A well-instrumented TV eco-system mirrors the web world in terms of volume, velocity, and variety of data that is collected, albeit not at the same scale. Volume of data is collected from loosely coupled systems and modules that span application, control, and delivery services. With disparate services comes data variety. All collected data has to be analyzed with velocity to provide meaningful and useful information to subsequent applications impacting user interaction, or systems impacting operator systems. The analyzed results can be provided through application programming interfaces (APIs), or through visualizations and operational dashboards.

ANALYTICS SYSTEM ARCHITECTURE

Analytics systems are comprised of three major components: data collection, storage, and analysis. A high level architecture of a general analytics system is given in Figure 1.

Data Collection

The key to any analytic system is collecting data, which is federated across a number of services within an eco-system. This requires instrumenting network, client services and applications, or equipment to generate and report events, which are then sent to an event intake mechanism. From an applications and a services perspective, event reporting should be a lightweight process where the complexities of ingesting events are abstracted away from the event collection probes.

At ARRIS, we have implemented an event-reporting library (in Java) that abstracts away the event collection logistics from the instrumented service. The instrumented service needs to

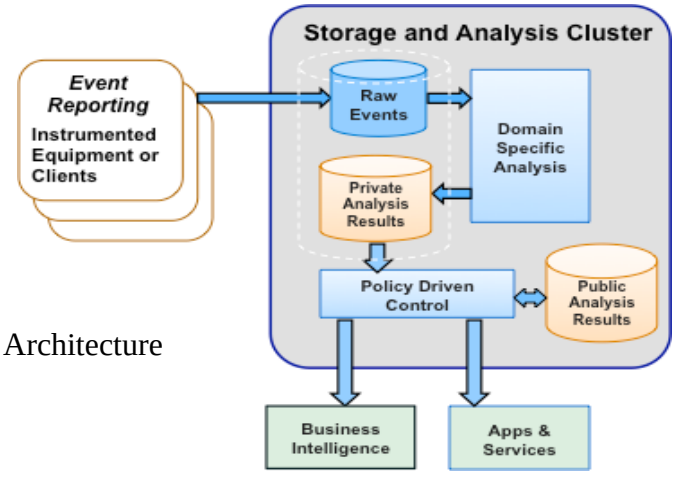

Figure 1 - High Level Analytics Architecture

define the events that are meaningful within its context and "report" them either using an abstracted library or directly to an event intake mechanism. Keeping this lightweight for the instrumented services, it is best to collect raw or minimally processed events, hence placing much of the compute burden on the analytics system.

We are currently using Spring XD [10] as our real-time event intake mechanism. It is an open source project being developed under the Spring framework umbrella. Spring XD, a distributed event intake mechanism supporting HTTP, TCP, and Syslog among others, allows for processing of these events prior to storing and persisting. We define one Spring XD stream per event type or event source.

Storage & Persistence

Once events are ingested, the question becomes one of the database technologies that the system should employ. The answer is quite simply a function of system requirements and the types of data that are being housed. The choices are either relational database management systems (RDBMS - SQL), or non-relational (NoSQL).

For storing and subsequently processing events from a variety of sources, where there is no obvious cohesive relational structure, a non-relational database such as HBase or files stored directly on HDFS indexed with Hive are obvious choices. As previously mentioned, this stems from Big Data technologies from the Internet. This allows for storing raw or minimally processed events for subsequent domain specific analysis. The result of any analysis or processing that is done on the data can also be stored in the same DBMS

The flexibility of Hadoop makes it a great choice for systems collecting information from loosely coupled, disparate sources, where processing typically needs to be done across the events to derive meaningful domain specific insight.

Data Analysis & Processing

Domain specific analysis needs to be done to accomplish targeted goals of the system. This is application dependent and determined by an articulated set of key performance indicators. For example, in the web domain, search and retrieval end up being important tasks. More complex tasks may include either content or social recommendations. Similarly, in the TV space, the data science applied determines the efficacy of the solution.

A Hadoop / HDFS system is designed to run MapReduce jobs for pulling and processing information. Higher-level languages that allow MapReduce scripting are Pig, Hive, and Scalding. Processing may be a simple statistical processing, correlating, and tabulating information across various individual services, or it may require machine learning techniques to do predictive analysis to profile human or system behavior; predicting user behavior facilitates end-user interaction, or predict system behavior / trends to circumvent system limitations.

A typical data analysis system houses sensitive user or subscriber information, which has to be protected and secured. A policy-based access control mechanism is imperative for ensuring that subscriber-specific information is exposed only to authorized external applications or business intelligence tools

Analysis and processing information have three primary goals:

- Create visuals and dashboards to provide business intelligence,
- Provide application-programing interfaces for downstream usage to drive system behavior or user enhancements, and
- Derive atomic and aggregate data sets on which additional analysis and exploration can be performed.

This is the data science or intelligence that rides on top of the Big Data technologies to derive business value.

ANALYTICS FOR DIGITAL AD INSERTION

Dynamic ad insertion (DAI) is a prime example of an application where Analytics plays a differentiating role increasing operator efficiency, measurement, and relevant ad targeting. Following the principles and architecture outlined previously, a data analytics system for DAI is comprised of instrumenting data sources for event collection, event intake, persistence / storage, and analysis. The results of the analysis are exposed via an application interface or they are exposed to a business intelligence tool for visualizing system information through appropriate dashboards.

Figure 2 is an architecture that is specifically designed for a DAI system. In this example, we are following an SCTE-130 based approach for DAI targeting multi-screen devices based on subscriber / user profiles [11].

The sources of event information are:

- ADS – Ad Decision Service for capturing ad placement request, response and notification events
- ADM – Ad Decision Manager for content delivery events. In the case of HLS, this may include manifest related information that is delivered to a client / application.
- CIS – Content Information Service for processed content and ad metadata
- POIS – Placement Opportunity Information Service for ad placement location events.
- Clients – Where possible actual client / application events are collected to get actual content and ad consumption information directly from the consumer facing application.

In addition to these operational events, bulk data imports are needed to correlate collected events with actual subscribers and programming. For this, we collect Subscriber Management System (SMS) information and guide information (EPG).

The events and bulk data are ingested via an

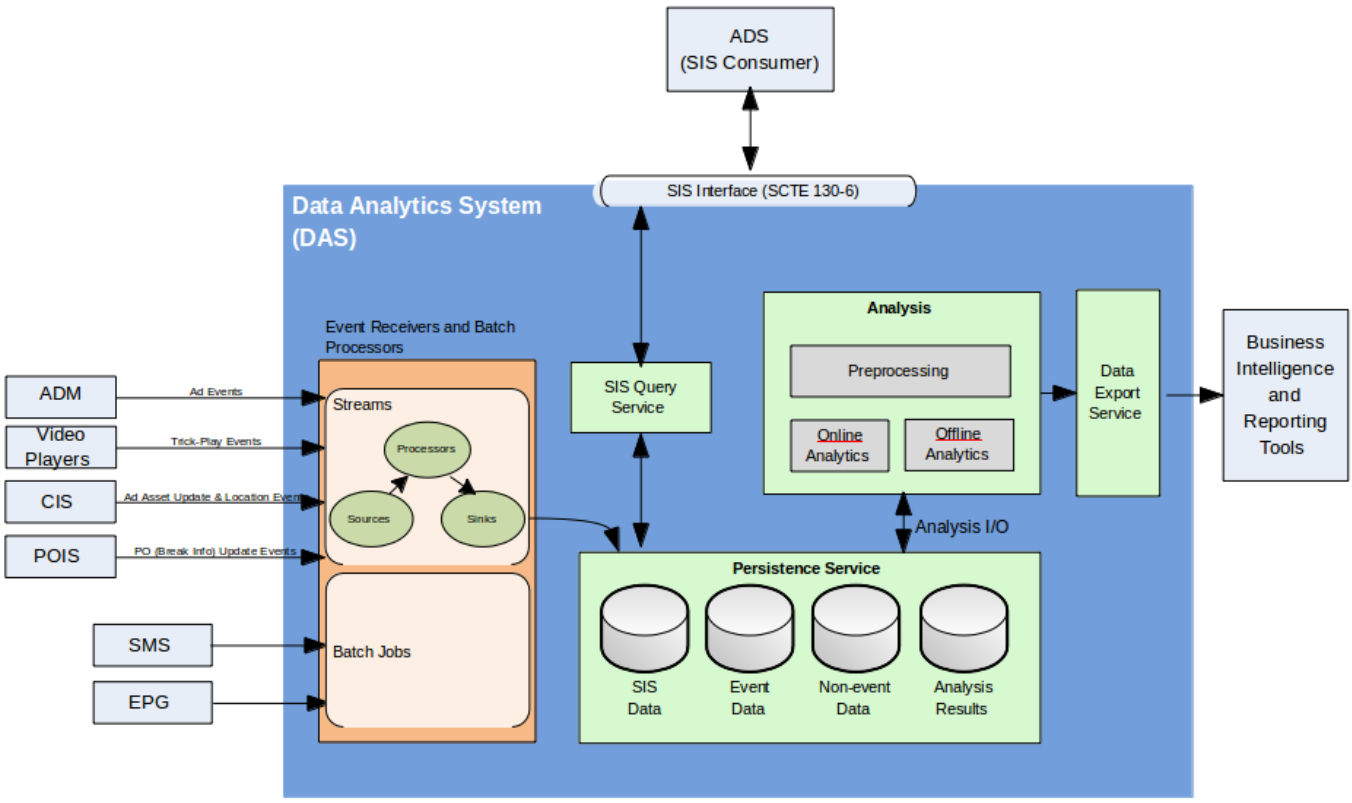

Figure 2 - Analytics Architecture for DAI

event intake module, which has the ability to do some filtering and processing of incoming data before storing it into our Persistence / Storage service. The collected information is processed using a variety of statistical techniques to derive user / usage profiles, and to collect historical information about DAI system usage. The user profiles are exposed via an SIS compliant interface so ADS systems can do appropriate ad targeting based on this user information. Historical performance and measurement information for content and ad delivery is visualized through appropriate dashboards for Business Intelligence.

In a typical DAI analytics system, the value is in:

- Classifying system behavior and usage. This has operational benefits for understanding how the system is being used. This is useful for operators, content providers, and advertisers.
- Characterizing subscribers so that appropriate insight is provided when targeting specific ads for better effectiveness.
- Understanding network and system limitations by analyzing and getting insight into trends of system usage and comparing them to current capacity has system is able to support. This has implications on scalability, storage, and bandwidth management.

DEPLOYMENT MODELS

The architecture supports various deployment models, either dedicated hardware-based or cloud (public or private). To enable cloud deployment, the services are developed to be hardware agnostic and can be scaled up or down as needed. Except in certain cases such as database interactions, the communication between different services is restricted to REST or SOAP (with preferably JSON, or XML, encoding) for simplicity. As new nodes can be added or removed, services are addressed by a proxy service.

The architecture is analogous to any Internet-scale web application: a distributed n-tier system. Unlike a standard web application that handles millions of requests/response from users, the DAI application handles millions of events from different components of the TV eco-system, which we denote as event sources. So as not to add additional burden on the event sources, we ensure that events are consumed asynchronously, and any additional filtering or

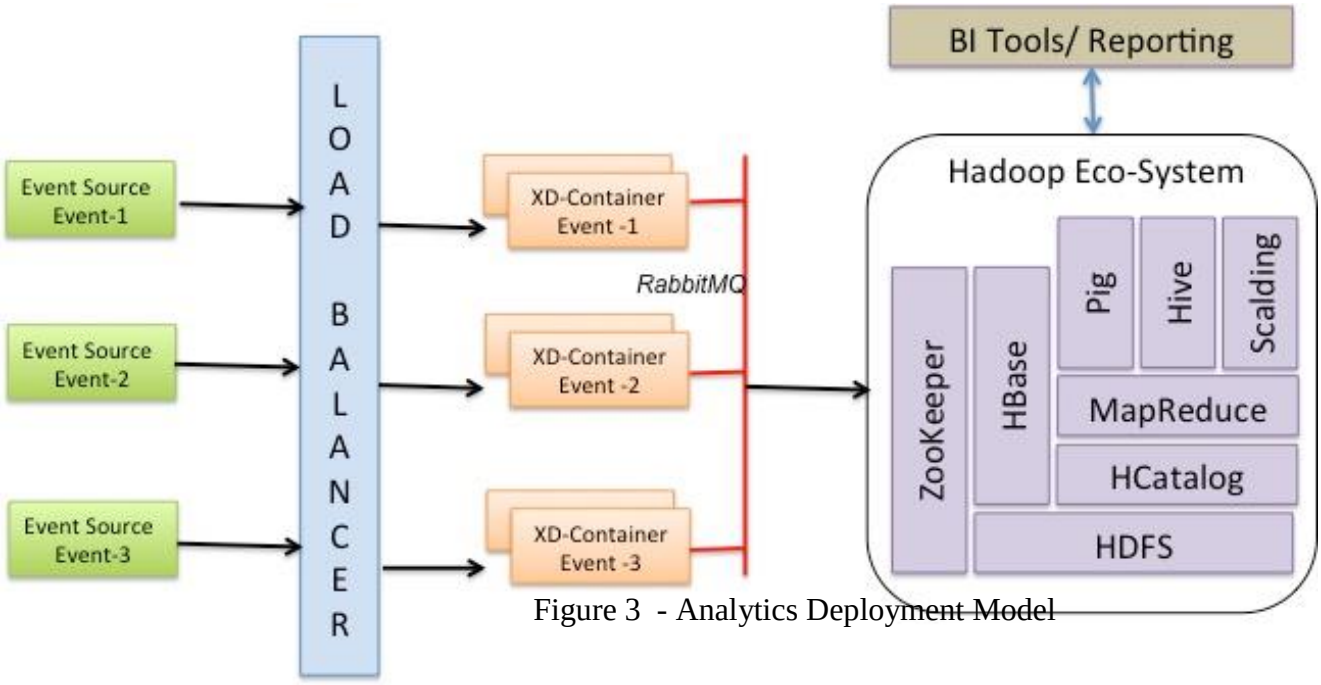

Figure 3 - Analytics Deployment Model

processing burden lies on the event consumers.

Special attention is given to high availability and fault tolerance throughout the system and especially at event consumer and storage/persistence layers. The Spring XD layer is deployed behind a software load balancer that allows us to deploy replica of each Spring XD container – one container per event type. Similar approach is taken at storage layer. Moreover, the Hadoop infrastructure (HDFS/HBase etc.) is already equipped with high availability and fault tolerance. Figure 3 depicts a typical deployment model.

ACKNOWLEDGEMENTS

This paper represents work that is being done by the Applied Research Center in ARRIS' Network & Cloud business.

CONCLUSION

Big Data technologies were developed primarily to target specific needs of the Internet. However, as our TV delivery systems grow more complex and as the technologies evolve to be more web-like, there is direct applicability and flexibility in using these technologies for television systems. A dynamic digital ad insertion system is just one example that can employ Big Data with appropriate analysis to provide more meaningful user interaction and targeting, operator measurement, and drive network efficiencies.

REFERENCES

1. Arthur, L. (2013, August 15). What is big data?. Retrieved from http://www.forbes.com/sites/lisaarthur/2013/08/15/what-is-big-data/

2. Couts, A. (2012, September 25). Breaking down 'big data' and Internet in the age of variety, volume, and velocity. Retrieved from http://www.digitaltrends.com/web/state-of-the-web-big-data/

3. Laney, D. (2001). 3D data management: Controlling data volume, velocity, and variety. In Application Delivery Strategies. Stamford, CT: META Group, Inc.

4. Ghemawat, S., Gobioff, H., & Leung, S. (2003, October). The Google file system. ACM - SOSP '03, Bolton Landing, NY.

5. Chang, F., Ghemawat, S., Hsieh, W., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., & Gruber, R. E. (2006, November). Bigtable: A distributed storage system for structured data. OSDI'06: Seventh symposium on operating system design and implementation, Seattle, WA.

6. Apace hbase. (2014, March 17). Retrieved from http://hbase.apache.org

7. Apache hadoop. (2014, March 14). Retrieved from http://hadoop.apache.org

8. Thusoo, A., Sarma, S. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Wyckoff, P., & Murthy, R. (2009, August). Hive - a warehousing solution over a map-reduce framework. ACM - VLDB '09, Lyon, France.

9. Apache Hadoop, Hive, and Pig on Google compute engine. (n.d.). Retrieved from https://cloud.google.com/developers/articles/apache-hadoop-hive-and-pig-on-google-compute-engine

10. Spring xd. (2014). Retrieved from http://projects.spring.io/spring-xd/

11. American National Standard, Society of Cable & Telecommunications Engineers (2011). Digital program insertion – advertising systems interfaces: Part 1 – advertising systems overview (ANSI/SCTE 130-1 2011)