An Architecture and Methods for Big Data Analysis

Bogdan Ionescu, Dan Ionescu, Cristian Gadea, Bogdan Solomon, and Mircea Trifan

Abstract Data production has recently witnessed explosive growth, reaching an amount larger than 4 zetabytes in 2013. This includes data sources such as sensors used to gather climate information, posts to social media sites containing digital pictures and videos, purchase transaction records, and cell phone GPS signals, to name a few. Not yet having more than an intuitive and ad-hoc definition, big data is challenging the IT infrastructure of all companies and organizations, forcing them to look for viable solutions. In essence, big data implies collecting, extracting, transforming, transporting, loading (ETL), classifying, analyzing, interpreting, and visualizing, among many other operations, on large amounts of structured, semi-structured, and unstructured data, in the order of a few petabytes per day, executed and terminated in critical time. This paper will introduce the architecture and the corresponding functions of a Platform and Tools implementing these challenging operations. The architecture is built around a distributed network of virtual servers called "agents", which can migrate around a network of hardware servers whenever available resources are provided or created. A Control Center makes decisions on moving the agents based on the availability of resources when needed. An example from the telecommunications industry will illustrate how the Platform is applied to this domain of big data.

Bogdan Ionescu
University of Ottawa, Ottawa, Ontario, Canada e-mail: `bsolomon@ncct.uottawa.ca`

Dan Ionescu
University of Ottawa, Ottawa, Ontario, Canada e-mail: `dan@ncct.uottawa.ca`

Cristian Gadea
University of Ottawa, Ottawa, Ontario, Canada e-mail: `cgadea@ncct.uottawa.ca`

Bogdan Solomon
University of Ottawa, Ottawa, Ontario, Canada e-mail: `bsolomon@ncct.uottawa.ca`

Mircea Trifan
University of Ottawa, Ottawa, Ontario, Canada e-mail: `mircea@ncct.uottawa.ca`

1 Introduction

Every day, an amount of 2.5 quintillion bytes of data is created around the world. At this rate, it is estimated that more than 90% of the data in the world existing today has been created in the last two years alone. Recent reports of market research companies, such as International Data Corporation (IDC) [7], mention that there is a tremendous increase in the rate of data creation, which from a rate of production of 1.8ZB ($1.8x10^{21}$B) per year in 2011 arrived at a rate of more than 4ZB per year in 2013. A doubling of the amount of data every other two years was also predicted. It is no surprise that data-intensive computing, search, and information storage, all done in almost real-time, have generated a new trend in the computing landscape. The spread of Cloud Computing and data storage options at low or acceptable costs for hosting server farms, as well as the rush for interpreting large amounts of data for predicting the advent of events of interest for business, politicians, social behaviors or endemics, all have changed the way that data was regarded and produced in the last decade.

"Big data", as it was called, is being touted as a critical challenge. It includes the processing of data to obtain analytics results, as well as its management. The term first appeared in 2011 in a Gartner Report about emerging technologies [12].

Most definitions of big data focus on the size of data in storage, although a more comprehensive definition shall encompass the variety and the velocity of data being acquired and processed. Thus the big data is defined merely by its volume, variety, velocity, and value - the four Vs. Each of them has ramifications. The *volume* is defined by terabytes, records, transactions, tales and files, the *velocity* of dealing with large amounts of information can be defined as batch, near time (soft real-time), or real-time streams, while the *variety* is defined by the character of the data such as structured, unstructured, semistructured, or all of the above, and the *value* is defined by the revenue or the results brought into the enterprise by using a big data application.

The term "big data" was created to define the collection of large amounts of data in structured, semi-structured, or unstructured formats in large databases, file systems, or other types of repositories, and the processing of this data in order to produce an analysis and synthesis of the trends and actions in real or almost real-time. Out of the above amounts of data, the unstructured data needs more real-time analysis, and bears more valuable information to be discovered, providing a more in-depth understanding of the researched subject. It is also the unstructured data which incurs more challenges in collecting, storing, organizing, classifying, analysing, as well as managing.

The successes registered by Narwhal and its team, namely, the big data analytics used by the winner of the 2012 Presidential Elections in the US, demonstrated that it is not enough to have or host a huge amount of data; rather, there is a need to know how to use it, too [2].

Industries have implicated themselves in a strong competition for providing a big data solution which can take advantage of the high potential of big data. Tier 1 companies are having products which address the challenges above by providing products based on the popular Hadoop framework [17]. Many government agencies, especially within the United States, have announced major plans to accelerate big data research and applications [14]. Despite the availability of commercially available big data storage devices and tools available from Google, Facebook, Baidu, Taobao, and Cisco (among others), the field of big data is still relatively new and requirements for big data, as well as the analytics associated with it, are not yet formalized. A good definition for big data is still being debated between main players in the field, with academia and the industry still discussing this subject. However, it is unanimously agreed that the value chain of big data consists of the following big categories of operations: data generation, data acquisition, data storage, and data analysis [5].

Many solutions for big data storage and processing have been experimented with. As such, permanent storage and management of large-scale disordered datasets, distributed file systems, and NoSQL databases are mentioned as good choices for approaching big data projects. Presently, there is a collection of technologies and tools available to experiment with various big data applications. Cloud computing infrastructures, Amazon AWS, Microsoft Azure, and data processing frameworks based on MapReduce, Hadoop, and Microsoft DryadLINQ allow running domain specific parallel applications. For example, Gunarathne et al. [8] used available cloud resources to assemble genome segments and reduce the dimension in the analysis of their chemical structure such that the analysis of 166-D datasets, which consisted of 26,000,000 data points, was accomplished.

This paper presents a cloud-based architecture and methods for a platform designed to acquire, index, collect, interpret, process, transport and store structured, unstructured, and semistructured data in order to provide the user with a customizable platform that is able to extract data and information from large data sources, as well as stream various repositories and resources. The platform is easily programmed via a graphical language using simple concepts such as data source, data destination, data converters, and data transformers. These primitive concepts allow for the establishment of a data flow which starts from data sources and moves to storage locations through data analytics interpreters, the latter being included in the data transformer group. Initially, the platform is discovering and collecting data and then indexes it in order to locate it quickly. A Hadoop mechanism is used for indexing and location. The final goal of the platform is to present the user with the results of the data analysis in an easy-to-read format such that the user can engage in decision-making across many domains without difficulty.

The platform was designed to enable the following:

(a) indexing and classifying static and dynamic data from structured, unstructured and semistructured data repositories;
(b) mining through statistical identification and discovery of complex events through analyzing and predicting data evolutions from structured, unstructured and semistructured data repositories;
(c) integration and unification of large-scale data from disparate resources and streams;
(d) analysis while maintaining data integrity and data cleansing;

(e) controlling the cloud environment for accomplishing data auditing while reporting changes of data in the controlled repositories;
(f) scalability and elasticity on discovery of new data sources;
(g) effective and meaningful decision support;
(h) continuous quality control of results.

The remainder of the present paper is organized as follows: Section 2 is dedicated to a review of existing technologies for building big data. Section 3 briefly introduces the requirements which are at the heart of the Big Data Platform architecture described in this paper. Section 4 outlines the proposed architecture for a Big Data Platform and briefly discusses key functions of such a platform. The core of the Big Data Platform is also presented in more detail. In Section 5, the architecture components are discussed in more detail. In Section 6, an application of the Big Data Platform is given. Section 7 provides the conclusions of the paper.

2 Related Work

The globalization of the economy brought with it a huge amount of information which companies, at all levels of their structural and organizational complexity, had to digest in order to set up a well-defined business plan, as well as marketing and production strategies. The immediate impact on the company IT infrastructure was overwhelming. The increasing data volumes of the early 2000s caused the storage and ~~CPU~~ technologies to be stretched over their limits by the numerous terabytes of data, thereby resulting with a scalability crisis. Enterprises had to make sound and solid business decisions to cope with the big data avalanche, as data is nowadays the new raw material of business. Today, big data is a “must have”, and so are the corresponding hardware and software technologies which can deal with it. No longer ignoring such information, companies nowadays rely on the newly discovered facts from real-time big data analysis.

Big data projects were initially derived from decision support systems, which, in an industrial wording, are known as business intelligence applications. As a consequence, a large spectrum of commercial and open source tools and libraries have been built. On the other hand, the academia has produced a rich theoretical basis for big data analysis using the strong theoretical background offered by statistics and operations research, as well as associated libraries that laid a solid foundation for data mining [10], data analytics [4], and decision making systems [16]. One of the results of the R project (among other projects) was the generation of specific languages and models, such as the R statistical language or the Predictive Model Markup Language (PMML), which allowed researchers to combine statistical methodologies (along with ready-to-use packages) in order to process massive amounts of data collected from various repositories, eventually feeding the decision support system. Forced by the nature of its business, Google had to develop hardware and software infrastructure to deal with data discovery, collection, storage and processing. On the software side, the MapReduce programming model [11] was designed and implemented. The Open Software movement reacted and produced Hadoop. The industry has also dedicated time and effort to produce big data platforms which enterprises can buy or pay to use on-the-cloud.

Tier 1 companies, such as EMC, Oracle, IBM, Microsoft, Google, Amazon, and Facebook, along with several startup companies, offer big-data solutions. Big data is presently one of the important strategic technologies. The US government produced their “Big Data Research and Development Plan” in 2012, along with a May 2014 study entitled “Big Data: Seizing Opportunities, Preserving Values” [14]. The European Community and Japan produced similar works, and even the United Nations issued a “Big Data for Development” report in which big data is seen as an efficient source of information for better protection and governing of the people [1].

Currently, the majority of big data projects, which are either financed by the industry or initiated in academic circles, rely on Hadoop. At the industrial level, Hadoop is used in applications such as spam filtering, network searching, click-stream analysis, and social recommendation. Companies including IBM [6], MapR, EMC, Cloudera, and Oracle offer commercial Hadoop execution and/or support, as well as parts or solutions for various applications. There is also a rich list of open source projects among which Hadoop is widely used [9].

Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of hardware or virtual servers, partially inspired by Google’s MapReduce and Google File System (GSF) papers. It’s framework consists of the following modules:

(a) Hadoop Common, containing libraries and utilities of the framework;
(b) Hadoop Distributed File System (HDFS), a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
(c) Hadoop YARN, the resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users’ applications;
(d) Hadoop MapReduce, a programming model for large scale data processing.

Hadoop modules were designed to be insensitive to failures. It has been successfully deployed by Yahoo, Facebook, IBM, EMC, Oracle and other big data analytics providers. The Hadoop “platform” also consists of a number of related projects such as Pig, Hive, HBase, Spark, Drill, D3.js, and HCatalog, among others.

Hadoop is deployable in an on-site datacenter, as well as in the cloud. The Hadoop cloud deployment avoids companies being trapped in specific set-up expertise. The cloud solution of Hadoop is available from Microsoft, Amazon, and Google.

As big data analysis implies setting up a mathematical environment for the application of statistical analysis methods (among which one can also include prediction), the R project [16] was developed. R is a language and environment containing modules for linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and graphical display of results. R provides facilities for data manipulation, calculation and graphical display, including:

(a) an effective data handling and storage facility,
(b) a suite of operators for calculations on arrays, in particular matrices,
(c) a large, coherent, integrated collection of intermediate tools for data analysis,
(d) graphical facilities for data analysis and display, and
(e) a programming language which includes primitives for conditionals, loops, user-defined recursive functions and input and output facilities.

The R system was written in R, a language which combines C, C++, and Fortran assets in the statistical analysis of real-time data. The R language was designed as to allow users to add functionality on an as-needed basis for each application component.

PMML [13] is an open source XML-based language designed to provide the means to describe models related to predictive analytics and data mining. PMML was developed to allow applications to describe and exchange models produced by data mining and machine learning algorithms. It supports common models such as logistic regression and feedforward neural networks. Models are described via an XML Schema. One or more mining models can be contained in a PMML document. A PMML document is an XML document with a root element of type PMML, and is a valid XML document.

Other languages such as WEKA [10] allow users to build stat-of-the-art machine learning algorithms and data pre-processing tools. In combination with KNIME [4], it allows for building using visual assembly of data and data mining pipelines. The combination of WEKA and KNIME provides a solid platform and tools for visually building complex and massive data analysis applications. Apache Mahout [3] is another Java based open source project dedicated to the creation of a machine learning library, to be used in conjunction with Hadoop, thereby providing a solid basis for algorithms for data clustering, data classification, pattern mining, dimension reduction and more.

A comprehensive survey of big data and related technologies is given in [5].

3 Requirements for the architecture of a Big Data Platform

Increasing amounts of data flooding the IT infrastructure have created serious challenges in data acquisition, storage, management and analysis. As relational databases only apply to structured data, traditional RDBMSs could not handle the huge volume and heterogeneity of big data. As big data applications encompass a complex combination of computing and networking infrastructures, a main set of requirements will consider the implementation of big data infrastructures so as to provide cost efficiency, elasticity, and smooth upgrading/downgrading solutions.

Based on the result analysis of several use cases, the following series of functional requirements have been determined as being crucial for the implementation of big data platforms:

(a) Heterogeneous data discovery and collection tools: the platform has to be capable of programmatically connecting with real-time data flows produced by databases, file systems, web pages, web sources, web services, emails, HTTP and FTP servers, data streaming sources, and specific applications such as CRP, ERP, and others;
(b) Data unification and integration tools: the platform must be able to access, collect, unify, cleanse, and store data from multiple and different data sources, and to deal with inconsistent or noisy data. The platform shall implement data pre-processing tools which shall unify the data and integrate the various data sources;
(c) Data indexing, tagging, mining and classification tools: the platform has to be capable of providing tools for data indexing, both statically and dynamically, to extract patterns and to classify the information contained in massive amounts of data;
(d) Statistical analysis tools: the platform must support different data types to be fed to statistical data analysis algorithms for calculating key statistic and stochastic parameters and provide results from a set of statistic functions such as the correlation coefficients, state estimation and prediction, and other statistical and stochastic analysis;
(e) Self-provisioning and self-optimization: the platform shall implement at least the self-provisioning and self-optimization functions of the set of self-management functions of an autonomic computing environment, as the complexity of the computing and networking infrastructure of any big data platform can very quickly reach the limits of human manageability;
(f) Interactive exploration: the platform has to provide a set of visualization techniques for visual analysis and easy interaction with the data;
(g) Intelligent decision support: the platform shall ~~provide~~ algorithms and corresponding mechanisms for domain-specific data interpretations required by the decision making systems;
(h) Manual Operation: the platform shall provide facilities for manual analysis, semi-automated decision support or fully automated systems from case-to-case;

The non-functional requirements shall include scalability and real-time or near-real-time performance.

4 An Architecture for a Big Data Platform

This section describes the architecture of a big data platform which fulfills the requirements listed in Section 3. The architecture is sketched in Figure 1.

The architecture shown in Figure 1 has, at its core, a data mining, monitoring, and management platform henceforth referred to as “M3Data”. M3Data provides a series of components which are interconnected by the Big Data Platform.

As represented in Figure 2, M3Data is endowed with connectors for various sources of information (either structured, semi-structured, or unstructured). It col-

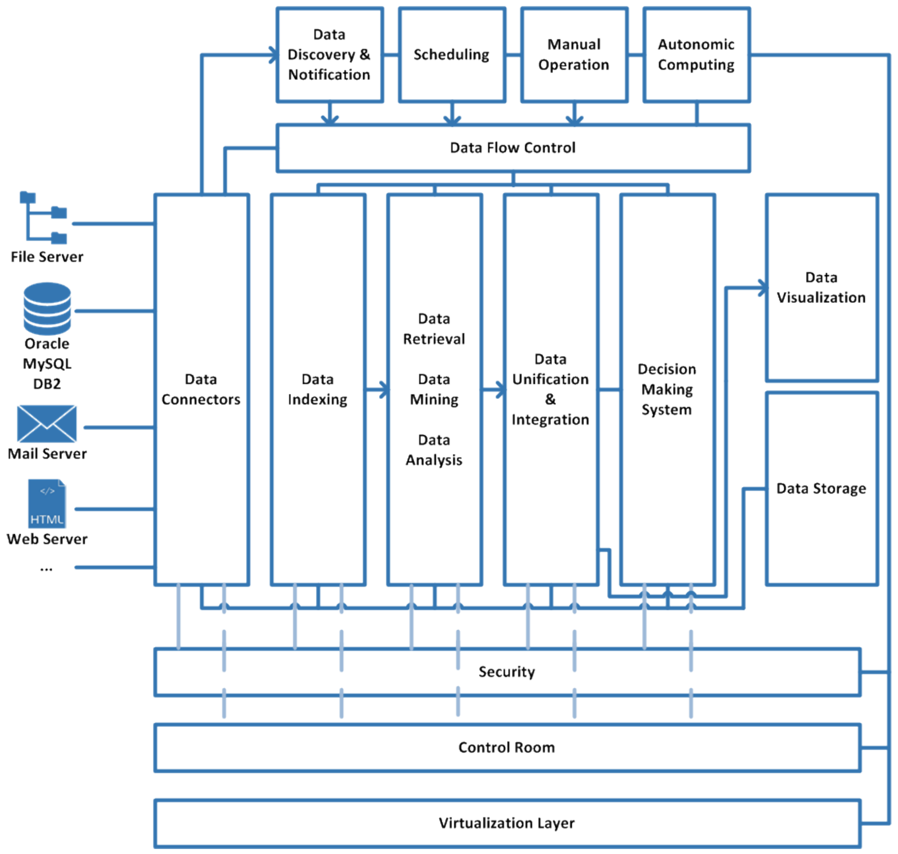


Fig. 1 Architecture of Big Data Platform

lects the data and puts it in a data flow object. It performs various processing operations on the data, from mining and analytics to extraction, integration, cleansing, and distribution of it. During all these processes, it is monitoring and logging all modifications ~~suffered~~ by the data.

The architecture of M3Data consists of six layers of software packages, as shown in Figure 3.

The layers of the core of the Big Data Platform assure the following functionality, all of which are essential to any big data application, and thus to the Big Data Platform presented in this paper:

(a) platform control: all control functions are implemented in the Control Layer of M3Data;
(b) platform security: specifically, M3Data uses the Role-Based Access Control security principles implemented in a Security Layer, which can also be coupled with an external security mechanism;

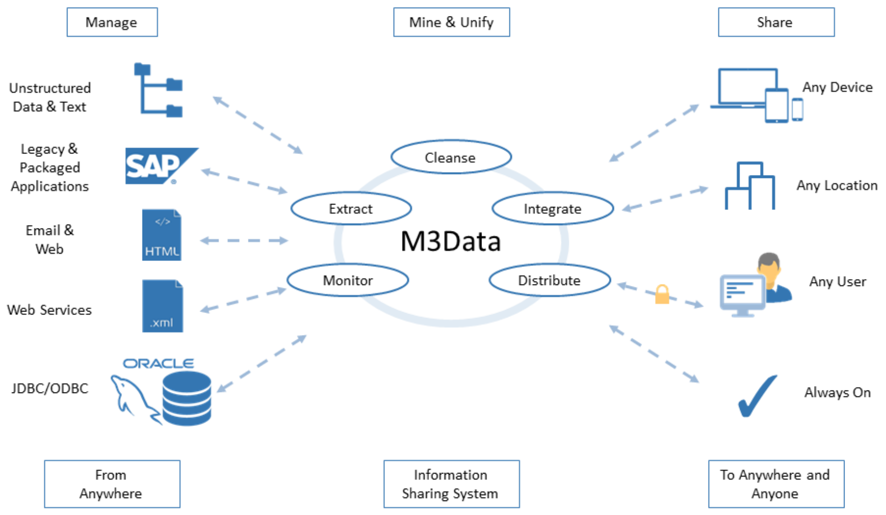


Fig. 2 What M3Data Is

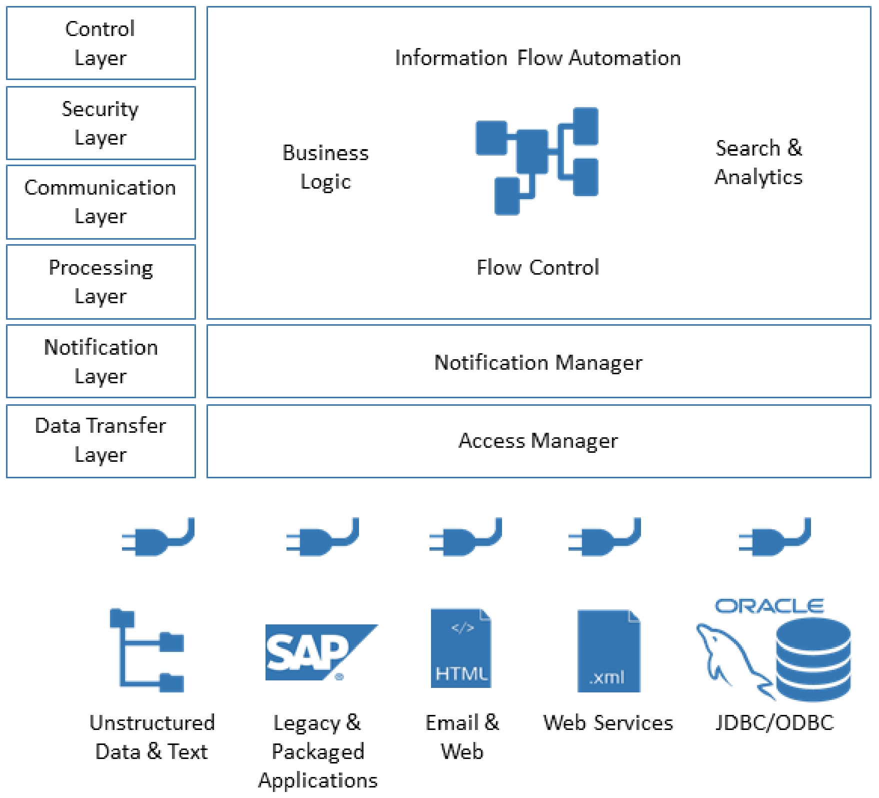


Fig. 3 How M3Data Works

(c) platform communication: the Communication Layer implements all protocols needed ~~to communicate with the data sources and inside the building blocks,~~ referred to as “agents”, of an M3Data application;
(d) platform processing: a series of mining, unification, and analysis of the data are included in the Processing Layer;
(e) platform event notification: M3Data Notification Layer implements functions for trapping, recording and passing to the Control and Processing Layer all events generated by the database triggers, OS events, emailers, etc.;
(f) data transfer: data transfer from source to destination among all data resources of the platform is implemented in the Data Transfer Layer.

The above layers all concur to the collection, transformation, and transfer of data from source to destination. One of the key modules of M3Data is a graphical programming language which allows users to cut the development time by an order of magnitude. This functionality is embedded in the Data Flow Programming Studio (DFPS) component, which contains all the graphical programming primitives and operators needed to ease the programming burden of designing and implementing a specific big data application. The user therefore only needs to translate the business rules of the application into a data flow describing the path ~~that the data has to go through~~ while various business tasks of the application run and then implement the business logic in a DFPS diagram. The look and feel of the iconic programming language, called Data Flow Programming Language (DFPL), is given in Figure 4.

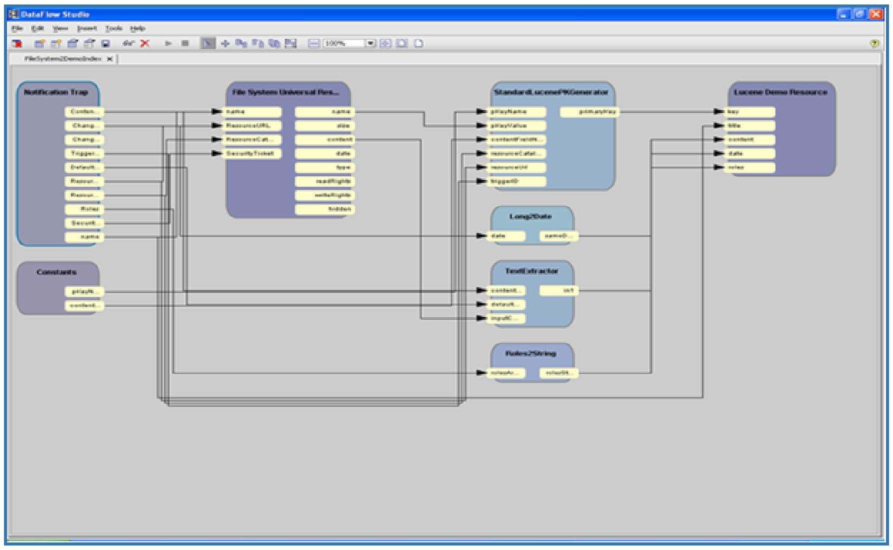


Fig. 4 Data Flow Studio Screenshot

The graphical language was implemented by following the rules and premises of graph-grammar theory. The data flow control is intrinsically contained in a DFPS diagram. A diagram can be built by using the following block diagram:

(a) resource blocks, which are iconic representations for the data repositories which sources of data for the overall application. The supported resources are DB2, ~~Informiz~~, any JDBC database, MSSQL, Oracle, and Lucene. Any of these resources can be either source or target for the data flow.
(b) converters, which can be simple or cycled. A simple converter implements a very simple data manipulation operation, while a cycled converter defines a multi-step data manipulation operation which has a well defined beginning and ending and will repeat multiple times between the end points.
(c) scripted converters, which are converters programmed via a wizard either in Java or using rules for the converters used for implementing decision support or expert systems using Drools. A special scripted converter is the conditional case converter used to evaluate conditions based on information known at runtime.
(d) subdiagrams, which are diagrams built by connecting smaller diagrams or blocks in a functional block.
(e) ~~arrows~~, which connect correctly the block in a diagram. For correctness, a diagram is automatically checked by the DFPL debugging tool.

The diagrams produced in DFPS can be inherited in other DFPS diagrams as blocks, the result being yet another diagram similar to the one showed in Figure 4. This property allows for the composition of DFPS diagrams in more powerful DFPL constructs. This composition of programming blocks is extremely practical as it simplifies the diagram complexity, which is one of the major hurdles of all graphical programming languages. In this way, applications of any degree of complexity can be built using partial diagrams, thereby helping the implementation of the application.

In what follows, a high level description of the main architectural components of the Big Data Platform is given.

5 Main Components of the Big Data Platform

This section presents a high level overview of the main components of the architecture of the Big Data Platform, as previously introduced in section 4. Namely, it describes components for data flow control, data connectors, data discovery, data notification and scheduling, data indexing and search, data mining and analysis, data unification and integration, decision support, data security, data control, data storage, data visualization, and autonomic computing.

5.1 Data Flow Control

As Big Data Platforms are required to constantly manipulate massive amounts of data (where the data often has a complex structure), the platform has to be endowed with tools that permit the administrators and users to program it in a very flexible way, thereby allowing them to interact with the data flow at every step of the operations which the data has to ~~suffer~~. Data is collected via the connectors of the platform, and is then prepared to be analyzed, ~~analyzed~~, stored and displayed. The DFPS diagrams therefore provide arrows that originate at the source of data and point towards the target repository of the data, and links together various blocks of the diagrams.

However, when data synchronization has to be done among many repositories, it is sometimes possible for two or more databases or repositories to hold the same data, and which therefore have to be kept synchronized while data is changed in one of the databases. This is made possible via triggers and the logic around them. Using the DFPS, the following big data operations can simply be inherited from the library of implemented blocks:

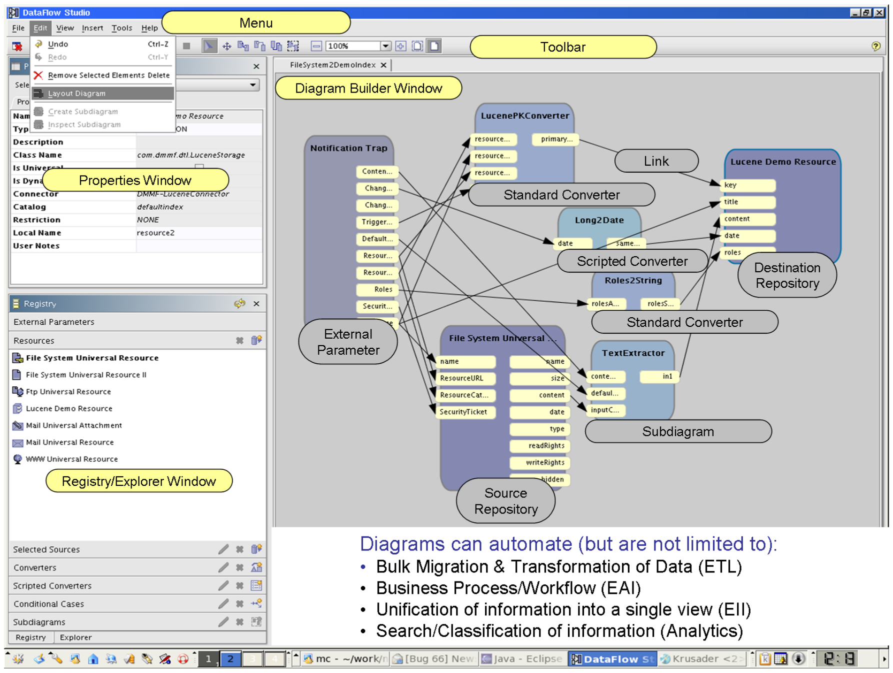
(a) accessing,
(b) cleansing and processing,
(c) distributing and migrating,
(d) monitoring and synchronizing,
(e) unification and reporting,
(f) monitoring, notifying and scheduling,
(g) searching and retrieving,
(h) classifying,
(i) logging.

Based on the above existing diagrams, more complex diagrams for big data processing can be built as shown in Figure 5.

5.2 Data Connectors

Big data applications require a platform that is properly connected to various sources of information which are to be analysed by the analytics module of the platform such that some predictions in regards to the business course that an enterprise has to adopt in order to be successful in a global competitive environment. A big data platform therefore has to provide all the connectors to the above sources of information such that any new information is acquired and interpreted in real-time or near real-time.

The Big Data Platform described in this paper contains connectors to various databases such as Oracle, DB2, Informix, Sybase, and other databases which have Java based APIs. A series of connectors are provided for additional legacy databases, some of which were programmed in the database's native programming language (usually C). There are also connectors to file systems specific for Windows, Linux,



Diagrams can automate (but are not limited to):
- Bulk Migration & Transformation of Data (ETL)
- Business Process/Workflow (EAI)
- Unification of information into a single view (EII)
- Search/Classification of information (Analytics)

Fig. 5 Using the Data Flow Programming Studio to build complex diagrams

and Unix/Solaris/AIX operating systems, as well as for email servers, web servers, web services, FTP servers and connectors for CRM, ERP, and BI applications. Any other connectors can be created and inserted in the Data Flow Diagram, where they will reside at the source level of the diagrams, as was shown in Figure 4.

5.3 *Data Discovery, Notification and Scheduling*

The platform contains modules which are in charge of data discovery of all existing sources of data, such as data streamers, file systems, databases, emailers, HTTP or FTP servers, and more. Connectors are available in the DFPS as basic blocks and are programmed using simple drag-and-drop operations on the block symbol of a diagram. The platform has to be authorized to connect with the sources of data by using corresponding access credentials to the database, file system, emailer, or other data source. A notification system prompts the platform when new data has been inserted in the repository and indexes it accordingly. The notification can trigger an action for validating the data modification or can place the notification and data in a batch mode. The latter functionality is contained in the Scheduling module of the platform.

A notification diagram can be used in the context of any data source and destination to trigger a runtime action of the Big Data Platform functions in regards to unification, mining, analytics and others. This is summarized in Figure 6.

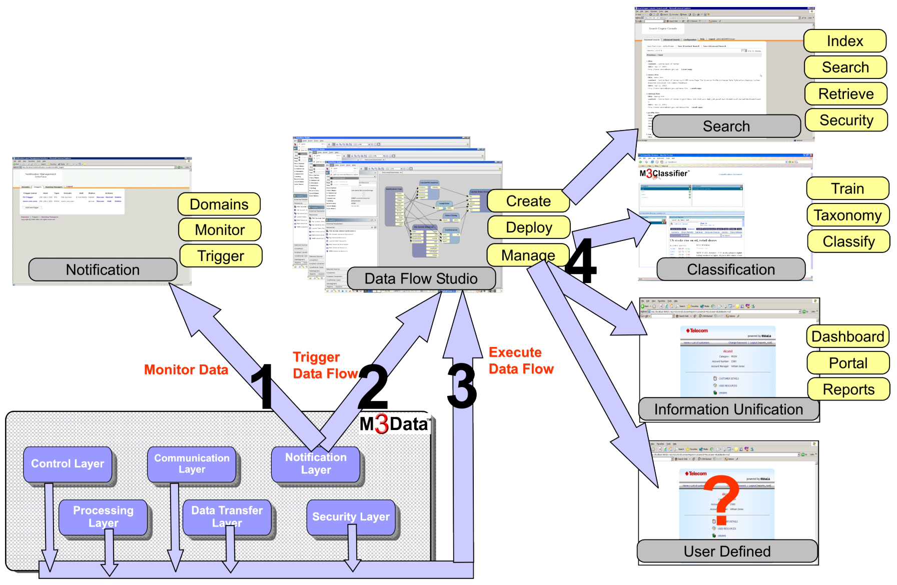


Fig. 6 Trigger a notification on a big data application; an example

5.4 *Data Indexing and Search*

The platform has powerful indexing capabilities and provides a search engine, ~~but this requires users to scan and index~~ the massive data repositories supported by the Big Data Platform. The indexing and search engine provides real-time web-style full-text search, as well as more advanced SQL-like querying capabilities. The indexing engine allows for more complex indexing of the data which accelerates the search and helps in the data classification. Data from distributed heterogeneous relational databases is retrieved along with other types of data such as files, web pages and emails. It is then indexed and displayed as needed based on search inputs or automated reports. Figure 7 illustrates the indexing and search features of the Big Data Platform.

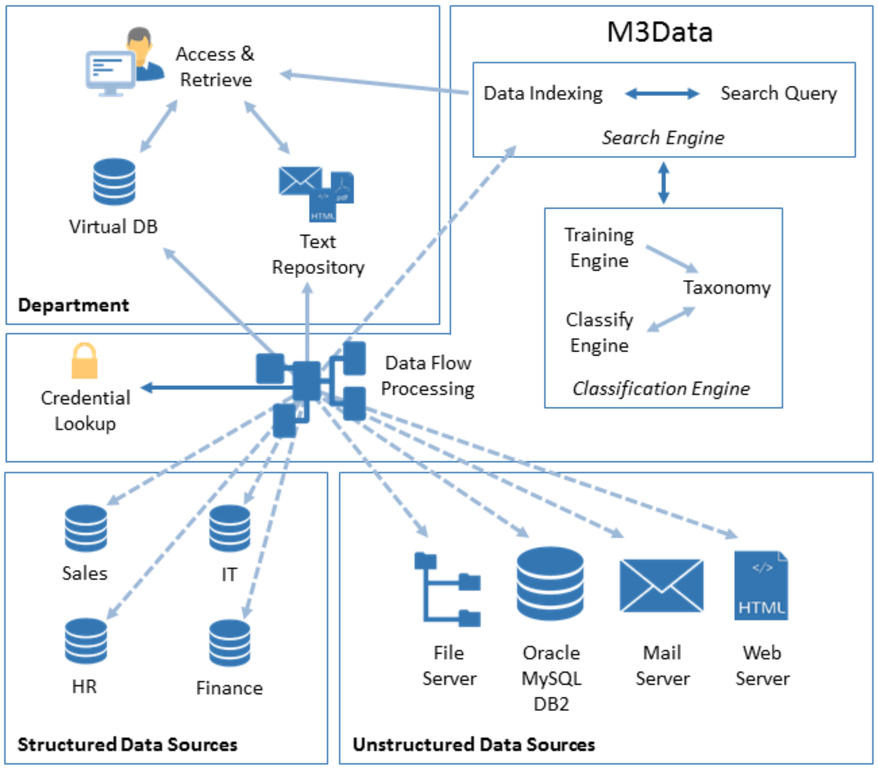


Fig. 7 Data indexing and search module

5.5 *Data Mining and Analysis*

Based on the platform's flexible search engine, a large variety of data can be inspected and explored for hidden information. The platform is capable of combining ~~SQL based~~ searches and results, providing flexible search with support for partial matches. In the case of partial matches, the results are statistically classified according to the confidence factor obtained.

5.6 *Data Unification and Integration*

The platform provides tools for the unification and analysis of massive amounts of data culled from a variety of structured, semi-structured and unstructured data sources. For unstructured data, an automatic XML schema matching tool is provided. At the XML level, a transformation of the XML schema takes place automatically for unifying data from two different text forms. For example, column items are mapped semantically into each other using a rule-based mapper. If a similarity of the items defined in the two XML description of the form are discovered, uniform values are produced (e.g for salutations, name suffixes, street abbreviation, etc). The unification tool of the platform is designed to be flexible in terms of the addition of new *transformer* blocks in the data flow programming diagram (Figure 4). The functionality of these new blocks can be programmed in Java or C/C++. By following the above procedures, WEKA-style mining modules and elements of the R language were integrated in the system.

5.7 Decision Support

The platform contains a business rule management system (BRMS) with a forward and backward chaining inference based rules engine. The BRMS module is based on Drools, which implements the Rete algorithm, and is a component of the JBoss Application Server. JBoss was used as the platform's distributed Java middleware. The decision support system of the Big Data Platform presented in this paper is therefore based on the Drools version included in the JBoss Community Edition server. This version of JBoss contains the Drools Guvnor (the Business Rules Manager), the Expert (the rule reasoning engine), the Flow (process/workflow), the Fusion (event processing/temporal reasoning) and the Planner (the automated planning).

5.8 Data Security

The security module of the platform implements a restricting system to control access for authorized users via RBAC. The platform allows users to access the information in the system if the users belong to a certain group of users, which is defined by certain roles for various job functions. The platform allows for the creation of roles, role assignment, and role and permission authorization. RBAC allows users to have differentiated access to the objects of information. If the role of the user does not have role/access privileges to obtain a specific object of information, the object is defined as non-existing for that user.

5.9 Data Control

A necessary module of a distributed system is related to the control of all aspects of storing and executing various actions by the platform on the distributed data. This resource management module is responsible for managing compute resources in clusters. It allows platform administrators to install, deploy, and upgrade all platform components. The control module keeps track of computational processes, their distribution in the system, as well as automatically redistributing them if some processes do not terminate in due time, or are interrupted. It allows platform administrators and application designers to locate, in real-time, the components (agents) of the big data application and to get information about their state. It also records all operations that take place in the system and logs them in a Journalizer for further investigation. In this way, users can audit all data movement and transformations in a big data application as needed. Figure 8 shows the main components of the Control Center of the Big Data Platform.

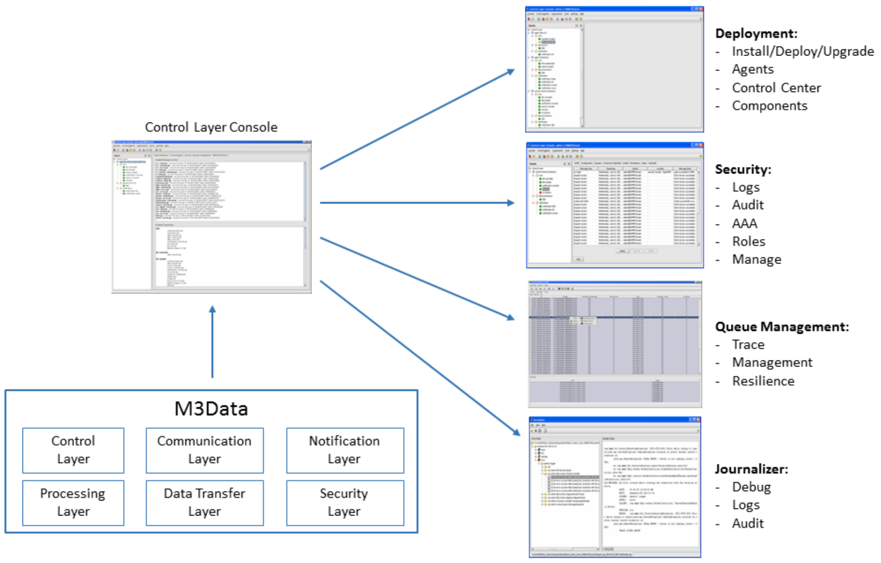


Fig. 8 M3Data Control Center

The following components make up the Control Center package:

1. The deployment component, which provides tools for the installation, deployment, and upgrading of the Big Data Platform at run time;
2. The security module, which was described in Section 5.8;
3. The queue management module, which keeps track of all processes, and their status, and implements resilience methods such that the Big Data Platform is reliable and avoids failures;
4. The Journalizer module, which keeps logs of all Big Data Platform operations for use in auditing the activities of the Big Data Platform and for debugging the Big Data functionality.

5.10 Data Storage

Although a very important component of a big data application, the data storage module is conditioned by the existence of the necessary hardware foundation that can handle very large amounts of structured, semi-structured and unstructured data sets while remaining capable of scaling in real-time in order to keep up with the growth of the data volume. It must also provide the input/output operations per second (IOPS) necessary to deliver data to the mining/analytics tools. This module comprises vast amounts of commodity servers with direct-attached storage (DAS). Redundancy is at the level of the entire compute/storage unit, and if a unit suffers an outage of any component, it is replaceable in its entirety (data included). Such a hardware/software combination is typically running Hadoop, along with NoSQL or Cassandra as analytics engines, and typically have PCIe flash storage instead of (or in addition to) regular hard drives so as to keep storage latency at a minimum. Due to the large volume of data, there is usually no shared storage in the configuration of all Storage Units.

A recent trend in the industry has been to use *Hyperscale* computing environments which are very well known from the implementations of the largest web-based operations of Google, Amazon, Facebook, Apple and others.

An alternative for big data storage is represented by a *scale-out or clustered NAS*. It uses parallel file systems that are distributed across many storage nodes. These systems can handle billions of files without the performance degradation experienced with ordinary file systems as they grow.

Another Storage Unit format which can handle a very large numbers of files is *object storage*. Object storage tackles the same challenge as scale-out NAS, namely, that traditional tree-like file systems become unwieldy when they contain a large number of files. Object-based storage gets around this by giving each file a unique identifier and indexing the data and its location. Rather than the typical kinds of file systems, this is method is more similar to how the Internet's Domain Name System (DNS) operates. Object storage systems can scale to very high capacity and large numbers of files (in the billions), and are therefore another option for enterprises that want to take advantage of big data. Object storage, however, is a less mature technology than scale-out NAS.

In general, big data storage needs to be able to handle large capacities and to provide low latency for data retrieval, thereby feeding the analytics module with data.

5.11 Data Visualization

Any big data platform has to provide a visual conclusion to the data extracted and interpreted by the analytics module. Data visualizations are sets of valuable tools in a variety of settings, allowing the creation of a visual representation of data points, reports on progress, or even the visualization of concepts for the customer segments.

This module is also external to the big data platform whose architecture is presented in this paper. By taking advantage of modern web-based graphic toolsets, open source projects (as well as companies that have this as their business model) exist that offer flexible visualization tools that are capable of handling big data. Another model for big data visualization is represented by interactive data visualization. Interactive visualization addresses the use of computers and mobile devices to drill down into charts and graphs for additional details, thereby interactively (and immediately) changing the data displayed, as well as the processing flow.

5.12 Autonomic Computing

Autonomic computing is one of the vital modules of the architecture. Autonomic computing is now a more mature technology and mainly addresses problems related to large and very large systems. It is meant to apply a supervised control of the computing environment. Autonomic computing is organized in a form of a control system for real-time processes. By using a control loop, it implements functions such as self-provisioning/configuration, self-healing, self-optimization and self-protection, effectively simulating the behaviour of a system administrator.

The autonomic computing module of the platform described in this paper is based on an autonomic computing platform proposed in [15]. The IT infrastructure of the Big Data Platform is the controlled object of the autonomic computing loop shown in Figure 9, which resolves the automated provisioning of the system over the virtual layer, taking into account constraints related to the optimal usage of computing resources while the Big Data Platform is resolving external requests.

In the Big Data Platform of this paper, the autonomic computing architecture is mapped into a real-time control loop whose elements or blocks are transducers, which include software transducers for CPU load, task response time, or think-time, all of which are provided by the middleware application server. The central decision-making system of the autonomic computing module makes decisions based on state estimation and resource availability. A controller which establishes the opportunity of provisioning yet another copy of the whole Big Data server, or using another computing resource for an agent of the Big Data Platforms, transmits this decision to an actuator which contains the entire set of deployment and configuration via regular expressions and which puts these commands into an execution list. Through the above process, the autonomic computing applied to the big data architecture presented in Figure 1 takes care of the automatic provisioning and of the optimization of the resource usage of the big data application.

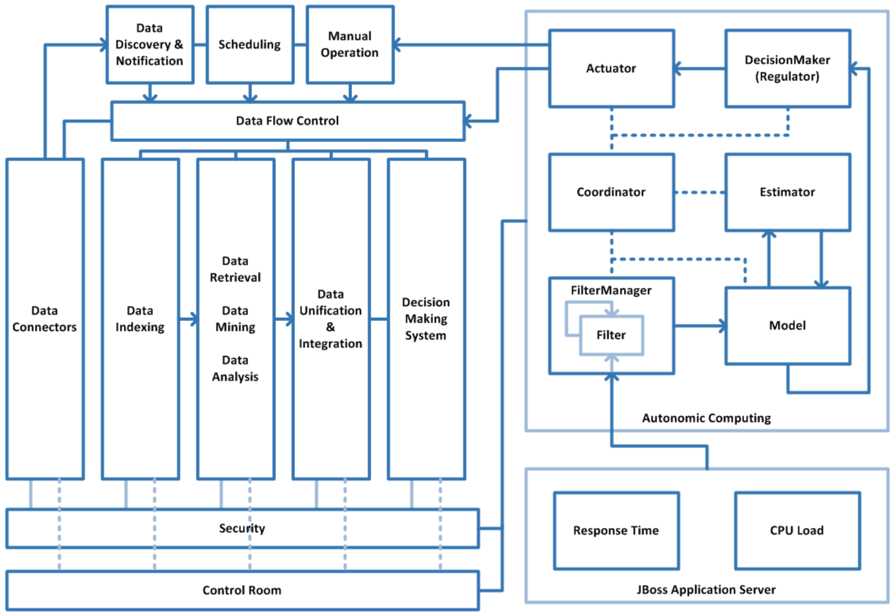


Fig. 9 Autonomic Computing environment for a Big Data Platform

6 Applications of the Big Data Platform

In a typical application, any big data platform has to provide services for three types of users: the big data platform administrator, the big data architect and/or designer, as well as for the final user/consumer of the big data platform results (who usually is not a savvy computer professional). This is represented in Figure 10.

The first deployment of the various components of the Big Data Platform presented in this paper is done by the administrator of the Platform, as shown in Figure 11. From that point on, the Autonomic Computing System takes control of the entire operation of the system in cooperation with the M3Data internal mechanisms, which take care of the distribution of agents and of processes within the entire distributed system onto which the Big Data Platform is deployed.

A test deployment of the Big Data Platform was created for a military system which implemented complex data acquisition, indexing, retrieval, unification and analysis of data collected from a series of databases, documents from a specific file system, emails, and web services. This deployment is summarized in Figure 12.

The visualization of the results obtained from the Big Data application are overlayed on top of a geographic map as shown in Figure 13.

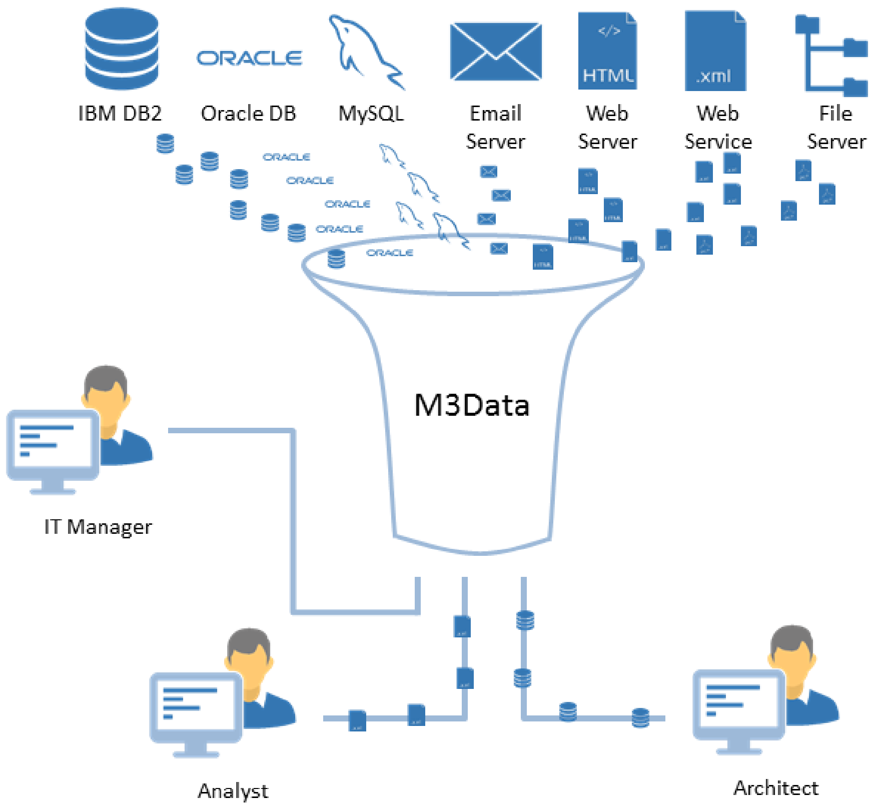


Fig. 10 M3Data Users

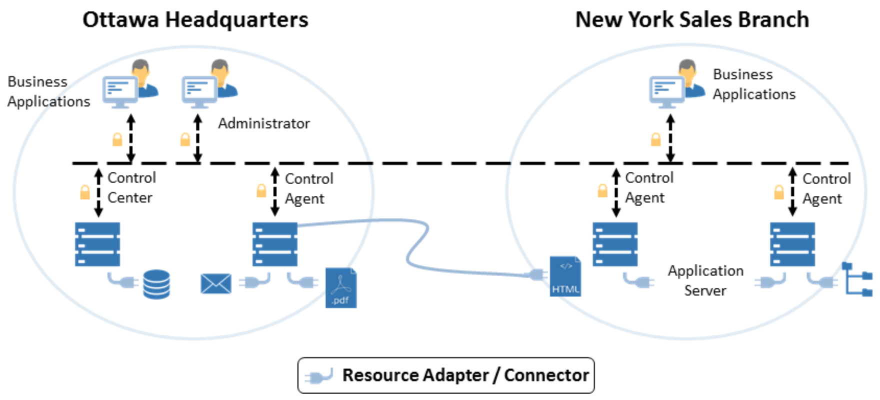


Fig. 11 Deploying a Big Data Platform

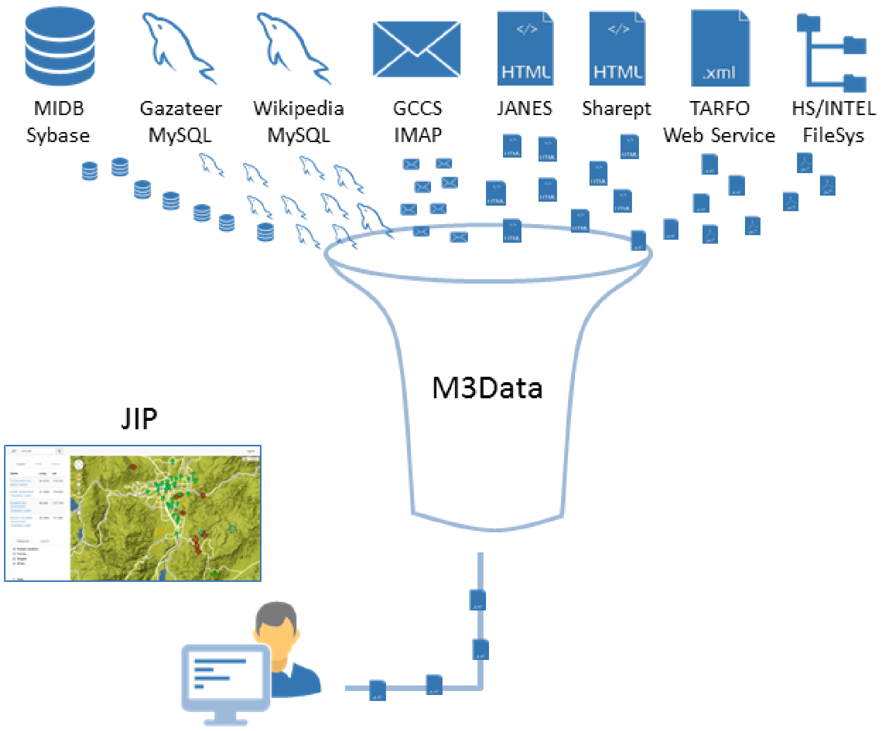
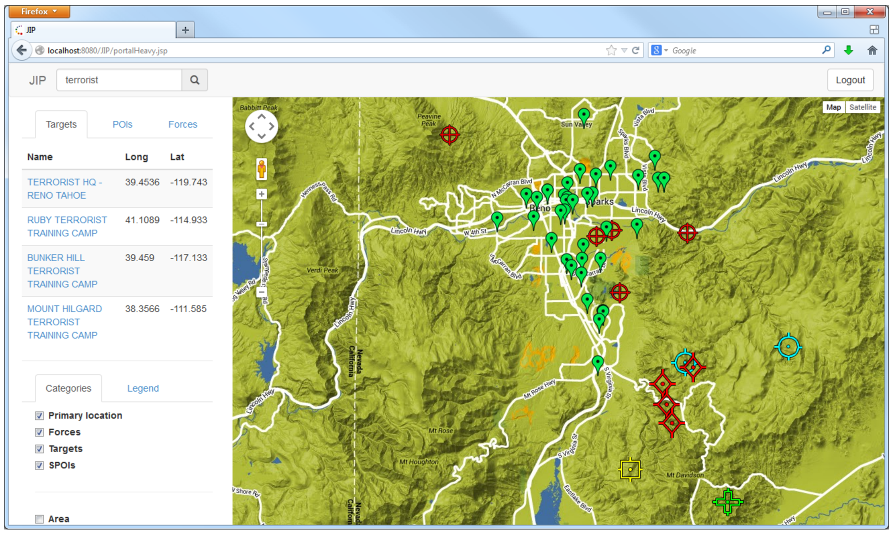


Fig. 12 An application of the Big Data Platform in the military domain



Name	Long	Lat
TERRORIST HQ - RENO TAHOE	39.4536	-119.743
RUBY TERRORIST TRAINING CAMP	41.1089	-114.933
BUNKER HILL TERRORIST TRAINING CAMP	39.459	-117.133
MOUNT HILGARD TERRORIST TRAINING CAMP	38.3566	-111.585

- [✓] Primary location
- [✓] Forces
- [✓] Targets
- [✓] SPOIs
- [ ] Area

Fig. 13 Vizualizing the results of a Big Data Application

7 Conclusions

In this paper, the requirement and corresponding architecture for a Big Data Platform have been introduced. The resulting platform is build around an Information Sharing Platform and comprises data collection components, data processing, analysis, as well as transport and visualization tools. The platform is also endowed with a graphical programming environment, a control center of the distributed system application, and automated provisioning and optimization systems. The graphical programming environment allows users to interactively modify the functionality and work-flow of various modules and functions of the Big Data Platform, thereby making it applicable for a series of applications which require collecting massive amounts of data from which hidden data has to be inferred and visualized. For an example deployment, the Big Data Platform was applied to the military domain. The flexibility, ease-of-use, robustness and other characteristics of the Big Data Platform introduced in this paper are therefore of significant value as big data continues to grow in importance in the IT industry.

References

1. Big Data for Development: Opportunities Challenges. `http://www.unglobalpulse.org/projects/BigDataforDevelopment`, 2012. [Accessed: July 2014].
2. Amazon. AWS Case Study: Obama for America Campaign 2012. `http://aws.amazon.com/solutions/case-studies/obama/`, 2012. [Accessed: July 2014].
3. Apache. Apache Mahout project. `https://mahout.apache.org/`, 2014. [Accessed: July 2014].
4. S. Beisken, T. Meinl, B. Wiswedel, L. de Figueiredo, M. Berthold, and C. Steinbeck. Knime-cdk: Workflow-driven cheminformatics. *BMC Bioinformatics*, 14(1):257, 2013.
5. M. Chen, S. Mao, and Y. Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.
6. C. Eaton, T. Deutsh, D. Deroos, D. Lapis, and P. Zikopoulos. *Understanding Big Data, Analytics for Enterprise Class; Hadoop and Streaming Data*. McGraw-Hill, Inc., 1st edition, 2012.
7. R. D. Gantz J. Extracting value form chaos. `http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf`, 2011. [Accessed: July 2014].
8. T. Gunarathne, T.-L. Wu, J. Y. Choi, S.-H. Bae, and J. Qiu. Cloud computing paradigms for pleasingly parallel biomedical applications. *Concurrency and Computation: Practice and Experience*, 23(17):2338–2354, 2011.
9. Hadoop. Hadoop Wiki: PoweredBy. `http://wiki.apache.org/hadoop/PoweredBy`, 2014. [Accessed: July 2014].
10. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.
11. D. Jeffrey and G. Sanjay. Proceedings of usenix osdi '04: Operating systems design and implementation. In *ICSOC*, pages 107–111, October 2004.
12. B. M. 2011 hype cycle special report. `http://www.gartner.com/newsroom/id/1763814`, 2011. [Accessed: July 2014].

13. PMMLorg. Pmml 4.2 - general structure. `http://www.dmg.org/v4-2-1/GeneralStructure.html`, 2014. [Accessed: July 2014].
14. J. Podesta, P. Pritzker, and E. Moniz. *Seizing Opportunities, Preserving Values*. White House Publishing, 1st edition, 2014.
15. B. Solomon, D. Ionescu, M. Litoiu, and M. Mihaescu. Towards a real-time reference architecture for autonomic systems. In *SEAMS '07: Proceedings of the 2007 International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 1–10, 2007.
16. R. D. C. Team. *R: A Language and Environment for Statistical Computing*. R Development Core Team, 1st edition, 2011.
17. T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2009.