The Case for Multi-Engine Data Analytics

Dimitrios Tsoumakos and Christos Mantas

Computing Systems Laboratory
School of Electrical and Computer Engineering
National Technical University of Athens, Greece
{dtsouma,cmantas}@cslab.ece.ntua.gr

Abstract. As big data analytics have become an important driver for ICT development, a large variety of approaches that apply these advanced technologies on a wide spectrum of applications has been introduced. In this paper we argue on the need of a multi-engine environment that will exploit the largely different models, cost and quality of the existing analytics engines. Such an environment further requires an intelligent management system for orchestrating and coordinating complex analytics tasks over the different available engines. After summarizing some of the current approaches in data analytics, we outline the structure of our envisioned *Multi-Engine Management System* and present some of the corresponding research directions in its design and development.

Keywords: Big Data Analytics, Execution Engines, Datastores, Virtualization, Modeling, Architecture, Resource Scheduling.

1 Introduction: Current State of Big Data Analytics Platforms

The field of data analytics includes techniques, algorithms and tools used to inspect collections of data to extract patterns, generalizations and other useful information. Big Data Analytics is – both from the technology point of view and its business repercussions – an important driver for ICT development and for the knowledge-based society itself. While the "data deluge" [1] is an obvious cause for this, it is also the case that businesses are moving away from purely empirical running towards running on factual information. Therefore, the term "Big Data" refers to more than the volume of data, but rather to the combination of data size, heterogeneity and processing complexity [2].

Nowadays, big data analytics is successfully applied in a multitude of diverse areas and disciplines such as risk assessment, pharmaceutics, fraud detection, epidemiology, business process effectiveness, market analysis, anti-terrorism, telecommunications, etc.

Modern datacenters host vast amounts of data that is stored over large numbers of nodes using multiple and possibly heterogeneous storage devices. The corresponding processing requires thousands or even millions of cores. Such data analysis demands a high degree of parallelism, in both storage and computation. Cloud technologies are able to provide the infrastructure to satisfy the immense resource requirements of the data and compute-intensive analytics tasks. Nevertheless, analytics applications must produce near real-time (or interactive) response times to the posed queries. This kind of efficiency is hard to achieve as it directly conflicts with:

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 406–415, 2014.

1) the size of the data,
2) its structural diversity,
3) data location, and
4) the ad-hoc nature of analytical queries.

The immense rates at which data is produced (point 1) is a fact mainly attributed to the advent of Web 2.0 and social applications, highly increased technology uptake, sensor/satellite networks, scientific advances, etc. Structural data diversity has significantly expanded as an outcome. Besides the well-understood transactional data, information has hence been produced in the form of semi-structured (XML, forms, emails, etc) or purely unstructured (blogs, tweets, multimedia, etc) data, both in already-stored and streaming modes. The combination of points 1 and 2 above justify the reality (and often necessity) of distributed data storage: As voluminous information is collected from different sources, using different schemas and at different rates, it is natural that no single site can store and serve all this diverse data. Last but not least, as the use cases for large-scale analytics increase at high rates, the amount and quality of processing required to achieve the desired output increase in complexity. Processing techniques include (among others) real-time data stream processing, interactive query evaluation (exploratory analysis), predictive analytics, complex Information Retrieval and Machine Learning methods over both text and loosely structured data, data integration, etc.

These issues have given rise to many diverse programming models, execution engines and data stores to assist with large-scale data management. The current architectural "blueprint" for data analytics is depicted in Figure 1: Diverse data sources are managed by specialized technologies that span the areas of data storage and job execution. The semantics and richness of the tasks that can be executed over the data are defined by the system's programming model.

Nowadays, such systems target specific data or process types, outperforming general-purpose solutions (e.g., a relational database) and tightly coupling the three components of Figure 1, namely the models with the execution and datastore engines. The most widespread solution for scaling applications to big data is the Map-Reduce model [3]. Map-Reduce is very effective with computations expressed as data-parallel "mapper" and "reducer" tasks that operate on different data shards and consecutively merge the results. The Map-Reduce model, through a series of open-source and customized implementations (e.g., [4,5,6,7]) is generally acknowledged as the current state-of-the-art in big-data analytics.

Many operations, however, act on irregular data that may be heterogeneous, structured or unstructured, streaming or stored in various formats. Complex analytics applications may also perform computations with dynamic dependencies (e.g., graph traversals or fixpoint computations). Such tasks cannot be easily defined in Map-Reduce, as each phase must store the intermediate graph state and redistribute it across machines before the next phase starts. To alleviate these difficulties, other specialized models and systems have been proposed [8,9,10,11]. While all these systems have had great success, they still showcase their advantages on a limited subset of applications and types of data: For instance, graph-processing engines (e.g. [8]) limit the amount of freedom in the computation at each node (or part of a graph) and fail to fully exploit possible parallelism.

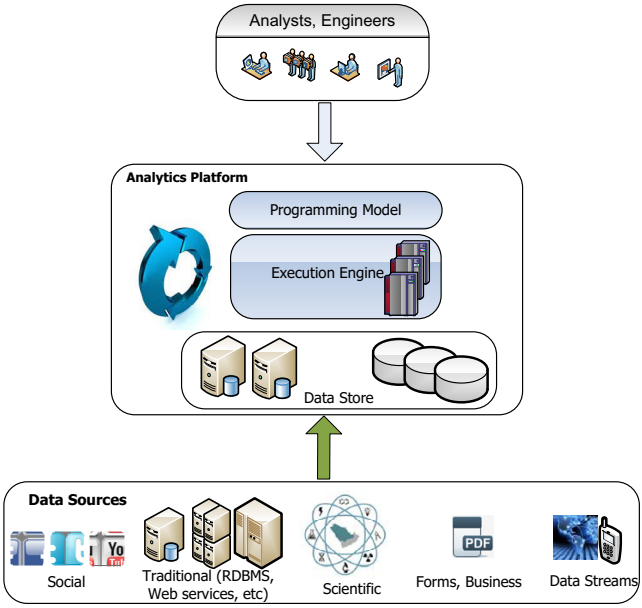

Fig. 1. Standard layering for data analytics processing over big data

Yet, modern data analytics applications are more complex than this. First, dynamic data processing is often very heterogeneous in terms of complexity and time consumption. In graph-structured data, for instance, the amount of stored data and required computation may differ from node to node. Many web-content analytics tasks can be easily parallelized and thus fit the Map-Reduce concept very well. Interactive analytics on semi-structured data requires different execution schemes respectively. It is thus suboptimal for a single model/engine to be used for all different analytics tasks. Different runtimes exhibit different cost and quality performance levels; any modern analytics platform has to be able to *fuse* models and engines of different capabilities in order to adapt to multiple job submissions.

Second, modern datacenters often include many heterogeneous storage formats, each used in multiple contexts and by different applications. This range may include traditional RDBMSs, modern column-stores, unstructured data in raw files, semi-structured data (e.g., XML, RDF), specialized stores (e.g., RDF stores), NoSQL databases, etc. Additionally some or all of the above may use central or distributed storage models. Depending on the data format, the computation of a query may differ in complexity and performance. As it is not always optimal to convert data formats and storage, for legacy or performance optimization reasons, a data analytics framework needs to support and adapt to multiple data storage formats.

Finally, a factor orthogonal to all the aforementioned points is the virtualized environment and its special performance restrictions. Avoiding expensive vendor lock-ins is a priority for many application owners; they would largely prefer being able to utilize both proprietary and popular open-source solutions. Additionally, running resource

intensive analytics tasks under heavy load and time constraints require Cloud-enabled strategies to be tightly coupled with the design and deployment of the application.

In this work, we present the case for an adaptive, intelligent analytics platform running over the Cloud. Specifically, we outline a "reference architecture" as well as the derived research directions for a software layer that will perform adaptive resource management among multiple execution engines and datastores. As no single technology has been able to demonstrate optimized performance for different load and data types, we strongly believe that efficient, cost- and quality- based fusion of alternative technologies will achieve what is currently missing, namely: Performance gains, diverse and user-controlled task execution, abstraction, flexibility and robustness. In Section 2 we briefly outline some popular current analytics approaches. In Section 3 we present our vision for a "multi-engine" data analytics platform and conclude our work.

2 Data Analytics Approaches

As the Big Data trend grows stronger in the various industries, data analytics tools evolve in order to handle large scale datasets effectively. Most of the research work is centered on adapting Hadoop [4] to the needs of time-effective data analysis and harvest both the effectiveness and agility of the Map-Reduce programming paradigm and the low query execution time of parallel DBMSs. Pig [12], Hive [13], Sawzall [14], Jaql [15] and Tenzing [16] fall in the category of offering SQL processing on top of Hadoop. Those systems implement SQL or SQL-like functionalities over the Map-Reduce model and are widely used by large enterprises in the industry.

Starfish [17] uses a sampler and a profiler in order to perform dynamic instrumentation of Map-Reduce jobs. This method uses a tracing tool in order to brake down the "data-load", "map" and "reduce" phases of the workload into sub-phases and collect run-time data of the jobs running on Hadoop. This data is used in order to automatically tune a number of configuration parameters affecting Hadoop's performance and achieve optimal execution of the workload, which is not possible to match by invoking the common rules-of-thumb that adept Hadoop users might be familiar with. The authors aim at making Hadoop a *MADDER* tool. MAD stands for Magnetism, Agility and Depth according to [18]. DER sands for Data-lifecycle-awareness, Elasticity and Robustness.

Similarly, Hadoop++ [19] tries to cleanly extend Hadoop by representing its execution pipeline as a physical execution plan and apply relational DB techniques. The first enhancement that Hadoop++ offers is indexing capability. The index, called the *Trojan Index*, is built during the data-load step of map-reduce with no penalty at query time. The second enhancement is a non-invasive join technique, called the *Trojan Join*. This technique exploits the co-partitioning of data in order to make a join of each co-partition at the "map" phase of the execution pipeline. This is far more cost effective than other join or merge operations previously proposed for Hadoop. Again, the aforementioned and other similar approaches choose Hadoop (or its variations) as the single analytics engine to operate upon and try optimize performance based on this specific choice.

Instead of building on Hadoop, ASTERIX [20] takes a different path, together with an emulation layer for existing Hadoop programs. ASTERIX tries to tackle the problem analyzing large scale structured and semi-structured data which processes in parallel,

on a shared-nothing infrastructure. The system uses a custom query language called AQL and its own runtime execution layer called Hyracks.

The system proposed in [21], executes workflows based on a generalization of the Map-Reduce programming model and operates on a key/value data store. The data processing is in the form of DAGs and is handled by the Nephele [22] computing engine. Nephele spans the DAG creating the appropriate instances of the user code and executes the workflow.

To overcome Map-Reduce's limitation for interactive data analysis, Google developed Dremel [10] and its public implementation, BigQuery. Dremel is an interactive tool that can produce results within seconds and can be used by non-programmers in order to analyze structured data, possibly in a trial-and-error, interactive, manner. Analysis on unstructured data is limited on regular expression matching. Dremel is based on columnar storage thus being able to achieve high compression rates and high scan throughput without prior indexing. While Dremel is not capable of executing complex data logic, it can act in synergy with the Map-Reduce framework to achieve that kind of functionality.

Commercial solutions for big data analysis mostly involve storage in proprietary parallel DBMSs targeted at OLAP applications. HP's offer is built on top of the IDOL engine [23] and the Vertica parallel analytics database [24]. The latter is a column-oriented DB used for data warehouses that either runs on grids of commodity servers or on the Amazon Elastic Compute Cloud (EC2). IBM also offers a data warehousing and analytics platform [25]. This product runs on custom data warehouse appliances and makes use of hardware data processing on FPGAs, being the building block for some more advanced analytics tools [26]. Very recently, IBM announced their newest "BLU Acceleration" technology for data analytics [27] that includes features such as the ability to skip over data that does not need to be analyzed, the ability to analyze data across different processors and the ability to analyze compressed data. Greenplum provides its own proprietary database [28] which offers a set of parallel functions, but also a custom Hadoop framework [29] and HAWQ [HAWQ], yet another SQL engine on top of Hadoop. Aster Data offers its own analytic management system [30]. A framework that compiles SQL into code that can be run on top of a Map-Reduce infrastructure has also been developed [31]. Microsoft is also involved in this area. It has developed HDInsight [32], a 100% Apache-compatible Hadoop distribution that integrates with its back-end products as well as a cloud-based version of it, that integrates with Azure cloud storage and the Azure SQL Database. Microsoft has also developed the Dryad distributed execution engine [33].

Summarizing this modest list of analytics platforms, we note that current approaches suffer from one or more of the following issues:

- They require or depend on proprietary tools, resulting in expensive vendor lock-ins
- They are optimized for specific data formats or data locations
- They are optimized for specific query/analytics task types

We believe that a modern analytics platform should instead base its operation on intelligent "fusion" of multiple technologies in the areas of data storage and task scheduling/execution. This allows the system to adaptively select between engines of different data models, computation models, costs and performance but also between different

technologies and possibly licensing schemes. To achieve this vision, a software platform is required to play the role of the "middleware" layer between the programming model and the low-level data and execution engines. This platform will schedule and manage compute and data tasks among the different technologies in a cloud-friendly mode, i.e., taking virtualization costs and restrictions into account. In the following section we outline a reference architecture for this "Multi-Engine Management System" by describing the major required modules, their functionality and respective research directions.

3 Multi-Engine Management System

Our vision is that analytics platforms should operate on the basis of a unified framework for managing, executing and monitoring multiple, complex jobs. Centrally positioned inside such a system lies the *Multi-Engine Management System* (MEMS). Its goal is to provide adaptive, cost-based and customizable resource management of the diverse resources available in the analytics platform (i.e., storage and run-time engines). The central notion behind MEMS is to utilize detailed models of the costs and performance characteristics of various analytics operations over the multiple engines. These models will then be used to intelligently match user-defined queries (and their policies) with the available engines. This is a challenging area that requires multidisciplinary research and development in Performance Modeling, Decision Systems, Resource Scheduling and Distributed Data Management.

Inside the lifecycle of an analytics job, MEMS will receive job description(s) from Application Management tools along with optimization policies that define the cost and quality constraints set by the user. MEMS will output a thorough execution plan along with its projected cost to the Scheduler whose task is to perform both the initial and continuous resource allocation. A runtime inspector tool will monitor and constantly evaluate/adjust models in order to alter scheduling towards more optimized execution (see Figure 2). In more detail, MEMS functionality will be performed by the following modules:

3.1 Job Parser

The goal of the job parsing tool is to receive input from the Application Management UI (user-defined analytics tasks/queries to be executed) and validate/structure it. In detail, this entails validating and registering the location (both physical and logical) of the required data; identifying the required metrics, dimensions, etc. over the data; identifying execution artifacts such as functions (average, min, max, etc), operators (joins, unions, similarity, clustering) and custom code; and finally, validating the user-defined policy. All this information must be robustly identified, structured as subtasks in a dependency graph and stored in order for the Modeling/Learning Engine module to be able to act upon it.

3.2 Modeling and Learning Engine

In order for the scheduler to choose an optimized execution plan for an analytics task that will span (a) multiple execution engines and (b) multiple data stores, a detailed cost

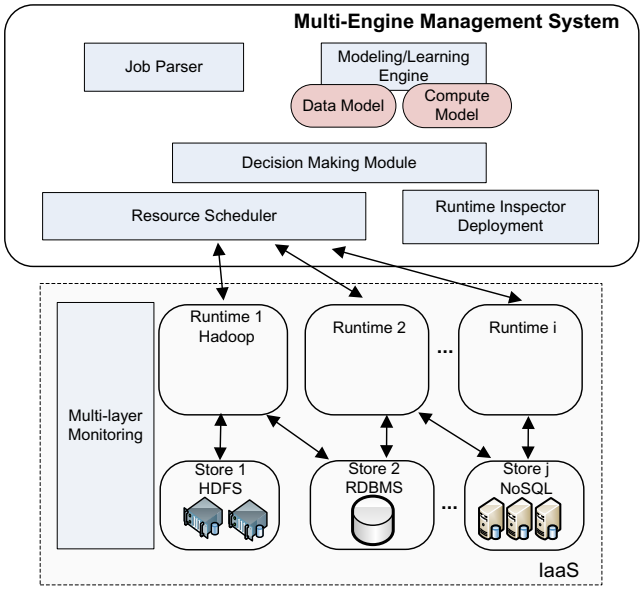

Fig. 2. A conceptual architecture for the MEMS platform

evaluation must be presented. This evaluation must detail what is the projected cost and performance (measured in time, committed resources, I/O operations, etc, but always in relation to the user-defined policies) of the parsed subtasks under variable scenarios (or deployments) of execution. These scenarios assume execution and storage engines to be variables; these variables might also impose extra variables such as the amount of committed resources per execution engine (e.g., number of worker VMs, number of allocated threads), storage size available, storage technologies (e.g., SSDs vs. traditional magnetic disks), etc. In order for such estimates to be feasible and realistic, a fusion of static models with real-time statistics from queries executed over MEMS is required. Static models will be created using detailed benchmarking; learning will be performed in real-time, as new, realistic estimates will enrich the current models both in accuracy and semantic reach. Consequently, work in this area must consider a broad range of methodologies and research activities that include benchmarking, performance modeling and learning. In addition, one must also delve into important characteristics that affect the performance of both data and compute engines such as elasticity requirements (how does altering the amount of resources allotted to a runtime affects its performance [34]) and storage architectures (e.g., shared-nothing vs. Shared-disk architectures, NoSQL databases, columnar stores and their respective advantages and disadvantages).

3.3 Automated Decision Making for Analytics Scheduling

Analytics tasks are often multi-level, resource-demanding jobs that operate on large volumes of diverse data. Their efficient execution is often of the utmost importance.

As MEMS's premise is transparent management of multiple runtimes and storage technologies available to the analytics expert, it is imperative that the assignment of tasks to the different technologies be optimized. This "intelligent" scheduling must be given central stage with a thorough study that will offer both automatic and optimal resource allocation. The problem in hand spawns several directions that must be appropriately studied. Research must consider the most fitting, among various disciplines, method for real-time, robust decision-making. The proposed algorithm must rely on the characteristics of the analytics task in hand and the model produced from the modeling and learning engine in order to schedule and orchestrate the execution of the different parts of the task. This entails deciding where each subtask is to be run, under what amount of resources provisioned, the plan for moving data to/from their current locations and between run-times (if more than one is chosen) and defining the output destinations. Equally important is the operation mode of the decision process: Specifically, the level of aggressiveness of resource commitment, the trade-offs between reactive or proactive invocation and its robustness towards oscillations.

3.4 Scheduling

In this module, methods and tools that translate high level "start runtime R using x amount of resources", "move data from engine Y to Z" commands to a workflow of primitives as understood by the specific runtimes and storage engines must be developed. This requires using standard, open-source or even custom APIs in order to allocate and free virtualized resources and also communicate with the supported platforms (i.e., store and compute engines). An additional challenge here relates to service availability and robustness that must be performed. Mechanisms that restart or reschedule jobs that have failed due to hardware or network errors must be studied. Consequently, work in this area must rely on real-time monitoring and learning mechanisms in order to quickly identify failures and bottlenecks, correlating current running times and results with previous experience and similar workloads. Work in this module must also investigate on the trade-offs of achieving availability and robustness over multiple run-times and storage technologies in a unified manner, keeping these methods transparent and efficient at the same time.

4 Conclusions

In this work, we argued on the necessity of a multi-engine framework to manage and schedule big data analytics tasks. The necessity is dictated by the largely different characteristics, cost and quality of the existing datastore and execution engines. Our vision is the design and consecutive deployment of MEMS, a middleware platform that intelligently coordinates complex jobs over collaborating technologies running on virtualized environments. In addition to the reference architecture and RTD challenges presented for MEMS, it should allow easy integration of multiple runtimes and datastores, be flexible enough to incorporate different programming paradigms and dynamically adjust its behavior relative to user-provided rules on a per-task or per-workflow basis.

References

1. The Economist: The data deluge (2010), http://www.economist.com/node/15579717
2. Ferguson, M.: Architecting a Big Data Platform for Analytics (2012), http://ibmdatamag.com/2012/10/architecting-a-big-data-platform-for-analytics
3. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. CACM 51(1) (2008)
4. The Apache Software Foundation: Apache Hadoop, http://hadoop.apache.org/
5. Liu, H., Orban, D.: Cloud mapreduce: A mapreduce implementation on top of a cloud operating system. In: 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 464–474 (2011)
6. Nokia, R.C.: The Disco Project, http://discoproject.org/
7. Amazon Web Services: Amazon Elastic MapReduce, http://aws.amazon.com/elasticmapreduce/
8. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: SIGMOD (2010)
9. Seo, S., Yoon, E.J., Kim, J., Jin, S., Kim, J.S., Maeng, S.: Hama: An efficient matrix computation with the mapreduce framework. In: Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, CLOUDCOM 2010 (2010)
10. Melnik, S., Gubarev, A., Long, J.J., Romer, G., Shivakumar, S., Tolton, M., Vassilakis, T.: Dremel: interactive analysis of web-scale datasets. CACM 54(6) (June 2011)
11. Hall, A., Bachmann, O., Büssow, R., Gănceanu, S., Nunkesser, M.: Processing a trillion cells per mouse click. Proc. VLDB Endow. 5(11) (July 2012)
12. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1099–1110 (2008)
13. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., Murthy, R.: Hive-a petabyte scale data warehouse using hadoop. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE), pp. 996–1005. IEEE (2010)
14. Pike, R., Dorward, S., Griesemer, R., Quinlan, S.: Interpreting the data: Parallel analysis with sawzall. Sci. Program. 13(4), 277–298 (2005)
15. Beyer, K.S., Ercegovac, V., Gemulla, R., Balmin, A., Eltabakh, M., Kanne, C.C., Ozcan, F., Shekita, E.J.: Jaql: A scripting language for large scale semistructured data analysis. In: Proceedings of VLDB Conference (2011)
16. Chattopadhyay, B., Lin, L., Liu, W., Mittal, S., Aragonda, P., Lychagina, V., Kwon, Y., Wong, M.: Tenzing a sql implementation on the mapreduce framework. In: Proceedings of VLDB, pp. 1318–1327 (2011)
17. Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F.B., Babu, S.: Starfish: A self-tuning system for big data analytics. In: CIDR, pp. 261–272 (2011)
18. Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: Mad skills: new analysis practices for big data. Proc. VLDB Endow. 2(2), 1481–1492 (2009)
19. Dittrich, J., Quiané-Ruiz, J.A., Jindal, A., Kargin, Y., Setty, V., Schad, J.: Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). Proceedings of the VLDB Endowment 3(1-2), 515–529 (2010)
20. Behm, A., Borkar, V.R., Carey, M.J., Grover, R., Li, C., Onose, N., Vernica, R., Deutsch, A., Papakonstantinou, Y., Tsotras, V.J.: Asterix: towards a scalable, semistructured data platform for evolving-world models. Distributed and Parallel Databases 29(3), 185–216 (2011)

21. Battré, D., Ewen, S., Hueske, F., Kao, O., Markl, V., Warneke, D.: Nephele/PACTs: a programming model and execution framework for web-scale analytical processing. In: Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 119–130 (2010)
22. Warneke, D., Kao, O.: Nephele: efficient parallel data processing in the cloud. In: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, pp. 8:1–8:10 (2009)
23. HP Autonomy: IDLE Server, http://www.autonomy.com/content/Products/products-idol-server/index.en.html
24. HP Vertica: Vertica Analytics Platform, http://www.vertica.com/the-analytics-platform/
25. IBM Inc.: IBM Netezza 100, http://www-01.ibm.com/software/data/netezza/100
26. IBM Corp.: IBM PureData System, http://www-01.ibm.com/software/data/puredata/
27. IBM Corp.: IBM Announces New Innovations to Help Organizations Benefit from the Next Natural Resource: Big Data, http://www-03.ibm.com/press/us/en/pressrelease/40768.wss
28. Greenplum, EMC Corp.: Greenplum Database, http://www.greenplum.com/products/greenplum-database
29. Greenplum, EMC Corp.: Greenplum HD, http://www.ndm.net/emcstore/greenplum/greenplum-hd
30. Teradata Corp.: Aster Big Analytics Appliance, http://www.asterdata.com/product/big-analytics-appliance.php
31. Friedman, E., Pawlowski, P., Cieslewicz, J.: Sql/mapreduce: a practical approach to self-describing, polymorphic, and parallelizable user-defined functions. Proc. VLDB Endow. 2(2), 1402–1413 (2009)
32. Microsoft Corp.: Microsoft HDInsight, http://www.windowsazure.com/en-us/manage/services/hdinsight/
33. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operating Systems Review 41(3), 59–72 (2007)
34. Tsoumakos, D., Konstantinou, I., Boumpouka, C., Sioutas, S., Koziris, N.: Automated, Elastic Resource Provisioning for NoSQL Clusters Using TIRAMOLA. In: Proceedings of the the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (2013)