Streaming Big Data Processing in Datacenter Clouds

Rajiv Ranjan
Commonwealth Scientific and Industrial Research Organization, Australia

Despite clear technological advances, research challenges must be solved to realize a standard large-scale, QoS-optimized platform for managing streaming big data analytics ecosystem.

Welcome to the inaugural Blue Skies column of IEEE's flagship cloud computing magazine. This column intends to provide an in-depth analysis of the most recent and influential research related to cloud technologies and innovations, focusing on streaming big data processing in datacenter clouds.

Big Data Computing Paradigm

Today, we live in a digital universe in which information and technology are not only around us but also play important roles in dictating the quality of our lives. As we delve deeper into this digital universe, we're witnessing explosive growth in the variety, velocity, and volume of data[1,2] being transmitted over the Internet. A zetabyte of data passed through the Internet in the past year; IDC predicts that this digital universe will explode to an unimaginable eight Zbytes by 2015. These data are and will be generated mainly from Internet search, social media, mobile devices, the Internet of Things, business transactions, next-generation radio astronomy telescopes, high-energy physics synchrotron, and content distribution. Government and business organizations are now overflowing with data, easily aggregating to terabytes or even petabytes of information.

The above examples demonstrate the rise of big data applications, in which data has grown unrestrainedly. Conventional data processing technologies are now unable to process this data within a tolerable elapsed time. Such applications generate datasets that don't fit the data processing model frameworks of traditional relational databases (such as Oracle, MySQL, and DB2) and data mining (such as Microsoft Excel, Matlab, and R). Relational databases operate on archived data in response to queries such as "commit a credit card transaction" (as in e-commerce). That is, the data processing technologies are designed to maintain an efficient and fault-tolerant collection of data that's accessed and aggregated only when users issue a query or transaction request (and thus the data must be archived prior to processing).

In contrast, all state-of-the-art implementations of data mining algorithms operate by loading the whole training dataset into the main (RAM) memory of a single machine or simple machine clusters that have static processing and storage capacity configurations. This approach has two key problems.[3–6]

2325-6095/14/$31.00 © 2014 IEEE

First, the data can simply grow too big over time to fit into the available RAM. Second, most of big data applications produce data spread across multiple distributed data sources (including streaming sources). Moving all the datasets to a centralized machine is thus expensive (due, for example, to network communication and other I/O costs), even if we assume that the machine has a super-large RAM to hold all the data for processing. Further, when the data mining algorithms' computational complexity exceeds the available RAM, the algorithms don't scale well and they never finish or are unable to process the whole training dataset.

To process data as they arrive, the paradigm has changed from the conventional "one-shot" data processing approach to elastic and virtualized datacenter cloud-based data processing frameworks that can mine continuous, high-volume, open-ended datastreams. This advancement is broadly supported by two key technologies.

Data Mining/Application Programming Frameworks

Data mining and application programming frameworks enable the creation of a big data analytics application architecture. Broadly, these frameworks can be classified into four categories. Large-scale data mining frameworks, such as GraphLab,[3] FlexGP,[6] Apache Mahout (http://mahout.apache.org), and MLBase,[7] implement a wide range of data mining algorithms—including clustering, decision trees, latent Dirichlet allocation, regression, and Bayesian—that can mine datasets in parallel by leveraging distributed set of machines.

Distributed message queuing frameworks, such as Amazon Kinesis (https://aws.amazon.com/kinesis) and Apache Kafka (http://kafka.apache.org), provide a reliable, high-throughput, low-latency system of queuing real-time datastreams. Data application programming frameworks, such as Apache Hadoop (http://hadoop.apache.org) and Apache Storm (http://storm.incubator.apache.org), write applications that rapidly process massive amounts of data in parallel on large sets of machines. To speed up the data mining algorithms, these frameworks simplify the process of distributing the training and learning tasks across a parallel set of machines. The frameworks also automatically take care of low-level distributed system management complexities, such as task scheduling, fault management, interprocess communication, and result collection.

Finally, NoSQL database frameworks, such as MongoDB (www.mongodb.org), HyperTable (http://hypertable.org), Cassandra (http://cassandra.apache.org), and Amazon Dynamo (http://aws.amazon.com/dynamodb), allow data access based on predefined access primitives such as key-value pairs. Given the exact key, the value is returned. This well-defined data access pattern results in better scalability and performance predictability that is suitable for storing and indexing real-time streams of big datasets. These frameworks can scale more naturally to ad hoc and evolving large datasets, as NoSQL databases don't require fixed table schemas or support expensive join operations.

Datacenter Clouds

The second key technology is datacenter clouds,[8–10] which promise on-demand access to affordable large-scale resources in computing (such as multicore CPUs, GPUs, and CPU clusters) and storage (such as disks) without substantial upfront investment.

Datacenter cloud services are a natural fit[1,4,6] for processing big datastreams, because they allow data mining algorithms (and underlying application programming and database frameworks) to run at the scale required for handling uncertain data volume, variety, and velocity. However, to support a complicated, dynamically configurable big data ecosystem, we need to innovate and implement novel services and techniques for orchestrating cloud resource selection, deployment, monitoring, and QoS control.

> Today, we live in a digital universe in which information and technology are not only around us but also play important roles in dictating the quality of our lives.

Big Data Analytics Ecosystem

As Figure 1 shows, a big data ecosystem's high-level architecture consists of three main components or layers:

- *Data ingestion* accepts data from multiple sources, such as online services and back-end system logs.
- *Data analytics* consists of many systems—such as stream/batch processing systems and scalable machine learning frameworks—that ease implementation of data analytics use

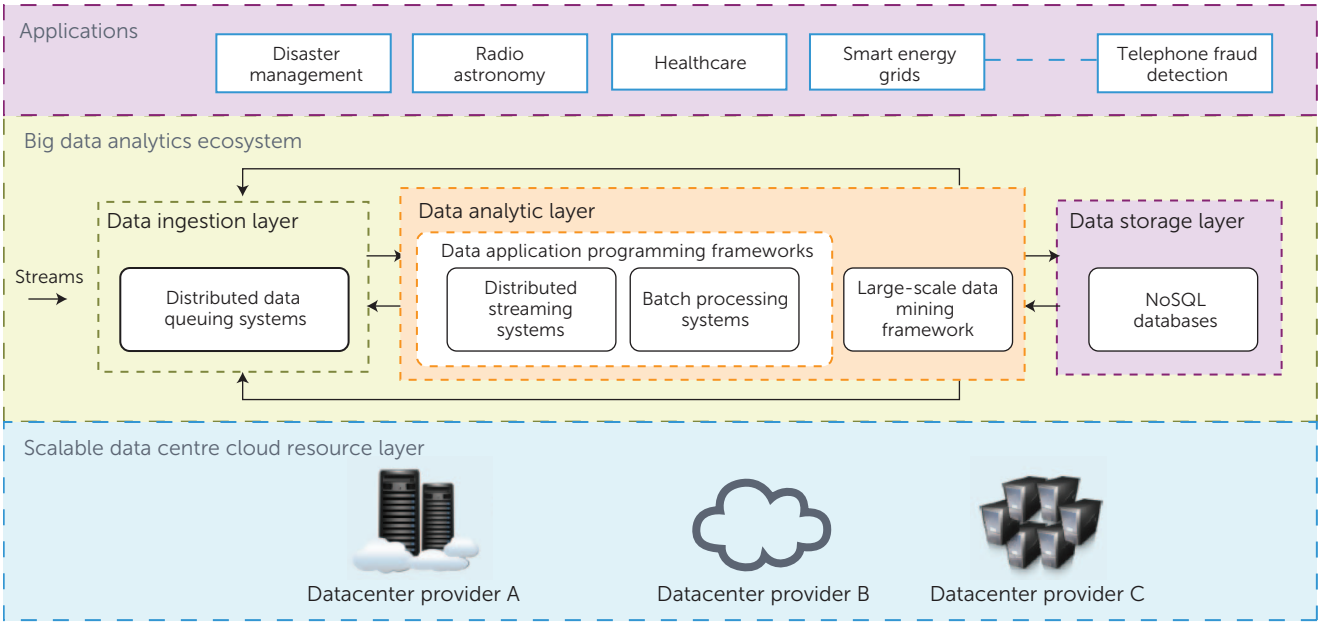

FIGURE 1. A high-level architecture of large-scale data processing service. The big data analytics architectures have three layers—data ingestion, analytics, and storage—and the first two layers communicate with various databases during execution.

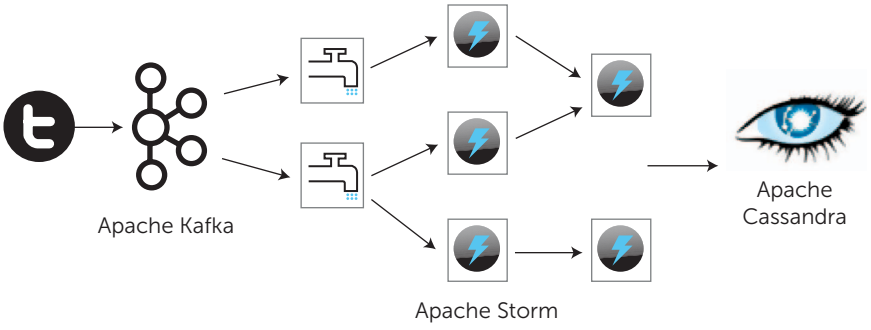

FIGURE 2. A simple instance of large-scale datastream-processing service. The example service consists of Apache Kafka (data ingestion layer), Apache Storm (data analytics layer), and Apache Cassandra Systems (data storage layer).

cases such as collaborative filtering and sentiment analysis.

- *Data storage* consists of next-generation database systems for storing and indexing final as well as intermediate datasets.

The first two layers talk with different databases during execution and, where required, persist or load the data in or from a database. The simple architecture in Figure 1 offers a snapshot of real ones; we encourage passionate readers to also investigate the Lambda Architecture.[11]

Recently, each architectural layer changed dramatically in terms of the software stack when services such as Yahoo!, Twitter, and LinkedIn released open source solutions for dealing with big data. Figure 2 shows an example of the new architecture: Apache Kafka serves as a high-throughput distributed messaging system, Apache Storm as a distributed and fault-tolerant real-time computation, and Apache Cassandra as a NoSQL database. It's not surprising that real-time stream-processing systems are just one building block in the big data ecosystem; computing arbitrary datasets via arbitrary queries demands a variety of tools and techniques.

Open Source Real-Time Stream Computation Frameworks

Although the stream-processing concept is not new, the available open source stream-processing systems are quite young and a silver bullet solution doesn't exist. Therefore, picking an appropriate platform for (near) real-time stream processing is a nontrivial task given the number of offers and their multiple features. To ease this process, we created an initial list of criteria: architecture, language support, integration with other technologies, and documentation and community support.

We divide the architecture dimension into centralized, distributed, and parallel distributed systems. Central-

ized in-memory streaming systems are suitable for handling queries with low-latency requirements, and they don't produce much intermediate state data. For example, Esper (http://esper.codehaus.org) is a streaming system with a centralized architecture that runs on a single node and keeps everything (states, operators, and so on) in memory. However, if the continuous queries have a large window size and might entail millions of tuples per second, a better answer is found in systems with a parallel-distributed architecture—such as Apache Samza (http://samza.incubator.apache.org)—which let you partition the streams and parallelize operators' execution across a cluster of machines.

Language support refers to the frameworks' flexibility in letting your team develop an application using your choice of language. Apache Storm is a good example here; it supports both JVM and non-JVM languages.

Another salient dimension is *technology integration*—specifically, the availability of ready-to-plug libraries for connecting the system to various in-line technologies. For example, Apache Kafka is a high-throughput distributed in-memory messaging system that complements every stream-processing system and has a ready component for this integration, which is a trump card.

The last (but not least) criterion is the framework's *documentation and community support*, which lets developers employ APIs easily. Here, Esper and Apache Storm have adequate documentation support, which is only expected to grow as more and more end-users adopt these systems. Table 1 gives an overview of state-of-the-art open source systems and how they meet the criteria.

Given more space, we would expand the criteria list to include more technical features such as dynamic rebalancing, state management, fault-tolerance, built-in monitoring, and metric reporting capabilities. However, in-depth analysis of overriding open source or commercial frameworks is beyond this column's scope.

Open Challenges and Research Directions

Despite the clear technological advances in machine learning, big data application programming frameworks, and datacenter clouds, we've yet to realize a standard large-scale, QoS-optimized platform as a service-level software for managing a streaming big data analytics ecosystem. Future efforts will focus on solving the following research challenges.

Understanding an Optimal Analytics System

It's not yet clear how to build an optimal big data application architecture given the abundance of existing frameworks that offer competing functionalities for large-scale data mining, distributed message queuing, data application programming, and NoSQL databases. Frameworks such as Apache Mahout implement several data mining algorithms, but it's not clear which is most suitable for processing given both historical and streaming big data in a distributed and parallel setting. Similarly, some data application programming frameworks such as Apache Hadoop are suitable for handling historic data, while others like Spark Streaming[12] or Apache S4[13] are better suited to streaming data. Similar complexities exist in choosing NoSQL databases, especially for heterogeneous (structured and unstructured) datatypes. Therefore, we must develop a solid scientific foundation and decision-making technique that can help us select these key functionalities based on the big data's nature (that is, its volume, variety, and velocity).

Monitoring and Managing End-to-End QoS

Guaranteeing QoS for large-scale data processing across multiple layers and various computing platforms is a nontrivial task. The QoS for each computing platform in the ecosystem isn't necessarily the same; key quality factors include throughput and latency in distributed messaging system, response time in the batch processing platform, and precision recall in the scalable data mining platform. Therefore, it is not yet clear

- how these QoS could be defined coherently across layers;
- how the various measures should be combined to give a holistic view of the stream of data flows end-to-end; or
- how optimal optimization would be realized in cases with large sets of variables and constraints, such as with heterogeneous resources, bursty workloads, and so on.

To this end, future research efforts must take an end-to-end QoS view of the ecosystem and develop techniques and frameworks that cater to all components rather than treating them as silos.

Provisioning Datacenter Cloud Resources for Real Time Analytics

Handling large volumes of streaming and historical data—ranging from structured to unstructured and numerical to micro-blog datastreams—is challenging because its volume is heterogeneous and highly dynamic. Although datacenter clouds offer abundant resources, they don't support QoS-driven autonomic resource provisioning or deprovisioning in response to changes in the 3Vs (that is, in the big data application's behavioral uncertainties).

The datacenter cloud resource provisioning's uncertainty[14–16] has two

Table 1. Capability Analysis of Recent Open Source Stream-Processing Systems.

	Architecture	Language Support	Integration	Documentation
Esper	Centralized in-memory	Java .NET Declarative SQL-like query language	API for integrating functionalities	Well-documented API and a thorough reference architecture that covers all features with clear-cut examples Active community mailing list
Apache Samza	Parallel-distributed	Java Virtual Machine (JVM) languages	Managed by Apache Yet Another Resource Negotiator (YARN) resource manager (Storm-YARN) Apache Kafka	Limited documentations and examples
Spark Streaming[12]	Parallel-distributed	Scala Java SQL-like query language (Shark)	Integrated scalable machine learning library (MLlib) Integrated graph processing algorithms Apache Kafka Apache Flume Twitter ZeroMQ Message Queuing Telemetry Transport (MQTT)	Limited documentations and examples
Apache Storm	Parallel-distributed	JVM and non-JVM languages Higher-level programming model (Trident)	Managed by apache YARN resource manager (Storm-YARN) Apache Kafka Kestrel RabbitMQ Java Messaging Services (JMS) Apache HBase (Storm-HBase) Twitter Machine learning integration with TridentML library	Well-documented APIs and online tutorials Several books Active community
Apache S4[13]	Parallel-distributed	JVM and non-JVM languages	Apache YARN	Limited documentations and examples

aspects. First, from a big data application's perspective, it's difficult to estimate workload behavior in terms of the data volume to be analyzed, data arrival rate, datatypes, data processing time distributions, and I/O system behavior. Second, from a datacenter resource perspective, without knowing the big data's requirements or behaviors, it's difficult to make decisions about the size of resources to be provisioned at any given time. Furthermore, the availability, load, and throughput of datacenter resources can vary in unpredictable ways, due to failure, malicious attacks, or network link congestion. In other words, we need reasonable workload and load resource performance prediction models when making provisioning decisions for datacenter resources that host instances of data mining algorithms, distributed message queuing systems, data application programming frameworks, and NoSQL databases.

Ensuring End-to-End Security and Privacy

Data stored on (and processed by) cloud resources and big data analytics ecosystem components aren't secured at finer granularity levels. The application data managed by these resources

and components are vulnerable to theft, because adversaries can gain access to private data and malicious database administrators might capture or leak data. Hence, research efforts must focus on developing techniques that efficiently support the following:

- end-to-end data encryption and decryption without causing additional query and data processing overhead (time and space);
- execution of various traditional SQL queries—such as equality checks, order comparisons, aggregates, and joins—or NoSQL queries over encrypted data; and
- preservation of data security and privacy at each lifecycle stage (including creation, ingestion, analytics, and visualization).

Developing techniques that can ensure end-to-end stream security and privacy remains a challenging research problem.

This column also welcomes high-quality position, survey, and review papers from cloud computing and related research areas. Future contributions might also include in-depth reports on innovative research projects in academia and research institutions, cloud and big data challenges at leading international conferences, and open source cloud computing projects.

Acknowledgements

I thank Omer Rana (Cardiff University), Lizhe Wang (Chinese Academy of Sciences), and Alireza Khoshkbarforoushha (Australian National University) for providing and discussing their viewpoints on research areas related to this column. I also thank Khoshkbarforoushha for his instrumental input in the compilation of Table 1.

References

1. X. Wu et al., "Data Mining with Big Data," *IEEE Trans. Knowledge and Data Eng.*, vol. 26, no. 1, 2013, pp. 97–107.
2. W. Fan and A. Bifet, "Mining Big Data: Current Status, and Forecast to the Future," *SIGKDD Explorations Newsletter*, vol. 14, no. 2, 2013, pp. 1–5.
3. Y. Low et al., "Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud," *Proc. Very Large Database Endowment*, vol. 5, no. 8, 2012, pp. 716–727.
4. S.R. Upadhyaya, "Parallel Approaches to Machine Learning—A Comprehensive Survey," *J. Parallel Distributed Computing*, vol. 73, no. 3, 2013, pp. 284–292.
5. D. Peteiro-Barral and B. Guijarro-Berdiñas, "A Survey of Methods for Distributed Machine Learning," *Progress in Artificial Intelligence*, vol. 2, no. 1, 2013, pp. 1–11.
6. O.C. Derby, *FlexGP: a Scalable System for Factored Learning in the Cloud*, doctoral dissertation, Dept. Electrical and Computing Eng., MIT, 2013.
7. T. Kraska, "MLBase: A Distributed Machine-Learning System," *Proc. Sixth Biennial Conf. Innovative Data Systems Research*, 2013; www.cidrdb.org/cidr2013/Papers/CIDR13_Paper118.pdf.
8. M. Armbrust et al., "A View of Cloud Computing," *Comm. ACM*, vol. 53, no. 4, 2010, pp. 50–58.
9. D.A. Patterson, "Technical Perspective: The Data Center Is the Computer," *Comm. ACM*, vol. 51, no. 1, 2008, pp. 105–105.
10. L. Wang et al., eds., *Cloud Computing: Methodology, Systems, and Applications*, CRC Press, 2011.
11. N. Marz, *Big Data: Principles and Best Practices of Scalable Real-Time Data Systems*, O'Reilly Media, 2013.
12. M. Zaharia et al., "Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters," *Proc. 4th Usenix Conf. Hot Topics in Cloud Computing*, 2012; www.cs.berkeley.edu/~matei/papers/2012/hotcloud_spark_streaming.pdf.
13. L. Neumeyer et al., "S4: Distributed Stream Computing Platform," *Proc. IEEE Int'l Conf. on Data Mining Workshops*, 2010, pp. 170–177.
14. R.N. Calheiros, R. Ranjan, and R. Buyya, "Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments," *Proc. 40th Int'l Conf. Parallel Processing* (ICPP 11), 2011, pp. 295–304.
15. J. Schad et al., "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," *Proc. Very Large Database Endowment*, vol. 3, nos. 1–2, 2010, pp. 460–471.
16. A. Iosup et al., "On the Performance Variability of Production Cloud Services," *Proc. IEEE/ACM Int'l Symp. Cluster, Cloud, and Grid Computing* (CCGrid 11), 2011, pp. 103–113.

RAJIV RANJAN *is a senior research scientist and Julius Fellow at the Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia. His research interests include cloud computing, big data, and quality of service (QoS) optimization in distributed systems. Ranjan has a PhD in computer science and software engineering from the University of Melbourne. Contact him at rajiv.ranjan@csiro.au and http://www.ict.csiro.au/staff/rajiv.ranjan.*