A Survey of the State-of-the-art in Event Processing

Otávio M. de Carvalho, Eduardo Roloff, Philippe O. A. Navaux

UFRGS - Federal University of Rio Grande do Sul
Parallel and Distributed Processing Group
{omcarvalho,eroloff,navaux}@inf.ufrgs.br

Abstract

Applications that require real-time or near-real-time processing of high-volume of data streams are pushing the limits of traditional data processing infrastructures. These event-based applications include market feed processing and electronic trading on financial markets, network and infrastructure monitoring, cloud computing applications, fraud detection, and command and control in military environments. Furthermore, great changes were caused by cheap micro-sensor technology. This ubiquity of sensors in the real world can lead to a big field of novel monitoring and control applications with high-volume and low-latency processing requirements.

This survey aims to review the state-of-the-art in event processing systems. A set of the most significant weaknesses and limitations is discussed at a high level, and we also outline requirements that a system should meet to excel at a variety of event processing applications.

1. Introduction

Applications that require real-time or near-real-time processing functionalities are changing the way that traditional data processing systems infrastructures operate. They are pushing the limits of current processing systems forcing them to provide better throughputs with the lowest possible latencies.

The main problems to be solved nowadays are not primarily focused on raw data, but rather in the high-level intelligence that can be extracted from it. As a response, systems were developed that can filter, aggregate and correlate data, and notify interested parties about its results, abnormalities, or interesting facts.

The latest advance in such systems is the development of high performance complex event processing (CEP) engines that are capable of detecting patterns of activity from continuously arriving data.

On the other hand, the distributed processing ecosystem today is mostly focused on Apache Hadoop [10]. This system proved that the development of large scale distributed processing systems on the cloud is possible. After understanding that it was possible to develop such systems, better approaches were proposed using Hadoop's infrastructure, focusing on improving the performance of these kinds of systems. The aim was not to limit them only to batch processing, but evolve them to systems of real-time processing.

During the development of applications that aim to achieve better throughputs using Hadoop, its bottlenecks were exposed, proving that it is not the best platform for certain kinds of intensive data processing systems, being better for workloads that are more batch processing oriented. Aiming improvements in the fields in which Hadoop failed, new approaches to distributed processing were proposed, focusing each time more on reliable processing systems that are not heavily bounded by intensive processing workloads.

These efforts generated a convergence between event processing systems and distributed processing systems, moving to merge those fields. Today's event processing systems have been focusing on distributed ways for data processing. On the other hand, distributed processing systems were trying to increment their platforms with complex tools to analyze the data processed by them.

2. Background and Motivation

The development of the area starts with DSMSs (data stream management systems), such as TelegraphCQ [4] and Aurora/Borealis [2] [1], which are similar to DBMSs (database management systems), but focused on managing continuous data streams. Also, in contrast to DBMSs, they execute a continuous query that is not only performed once, but is permanently executed until it is explicitly stopped.

On the other hand, we have CEP systems, which are

event processing systems that combine data from multiple sources to infer events or patterns that suggest more complicated situations. These systems are represented broadly by traditional content-based publish-subscribe systems like Rapide [16] and TESLA/T-Rex [5] [6].

In the evolution of those systems, a process of convergence between DSMSs systems and CEP systems had generated intersections between those fields, complicating more the characterization of them in distinct and clear groups.

Aiming to solve this naming problems, efforts were done to group all those kinds of systems into a common terminology. The term Information Flow Processing (IFP) [17] was coined to refer to an application domain in which users need to collect information produced by multiple sources, to process it in a timely way, in order to extract new knowledge as soon as the relevant information is collected.

However, a process of change in the event processing field can also be perceived related by the incorporation of distributed processing capabilities into the existent tools, related mainly to the creation of MapReduce programming model. As soon as the bottlenecks present in the model were exposed [19], new systems were created aiming to process the high amounts of data produced by big data applications. This process has generated a convergence between information processing systems and distributed processing systems, that were derived or inspired by the MapReduce papers. As well as the information processing systems had aggregated characteristics of distributed processing systems, many distributed processing system are aggregating information flow processing capabilities into their platforms. These changes are making it harder to explain the differences between them, because they are merging into tools that offer characteristics of both of them.

The main motivation of making a survey into the field of event processing systems are the recent initiatives of using their ideas and tools with distributed processing systems. These systems are showing that the approaches existent until now are not quick enough for certain kinds of information processing, mainly those that require high throughputs over big mounts of data.

3. Event Processing Systems

The current initiatives that aim to improve the existent event processing systems are described below. We can see in Table 1 those systems characterized in three distinct ways: Distributed systems, complex event processing systems and database stream management systems. Also, we explain if they offer relational database management systems, in order to show that some systems aim to offer broader capabilities than event processing.

Google Photon is a distributed system for joining multiple continuously flowing streams of data in real-time with high scalability and low latency, where the streams may be unordered or delayed. The system tolerates infrastructure degradation and datacenter-level outages without any manual intervention. Photon guarantees that there will be no duplicates in the joined output (at-most-once semantics) at any point in time, that most joinable events will be present in the output in real-time (near-exact semantics), and exactly-once semantics eventually. Photon is deployed within an advertising system to join data streams, such as web search queries and user clicks on advertisements.

Apache YARN is a new framework that facilitates writing arbitrary distributed processing frameworks and applications. YARN provides the daemons and APIs necessary to develop generic distributed applications of any kind, handles and schedules resource requests (such as memory and CPU) from such applications, and supervises their execution. YARN's execution model is more generic than the earlier MapReduce implementation. YARN can run applications that do not follow the MapReduce model, unlike the original Apache Hadoop MapReduce (also called MR1). It is the familiar MapReduce execution underneath, except that each job now controls its own execution via its own ApplicationMaster taking care of the execution flow (such as scheduling tasks, handling speculative execution and failures, etc.). It is a more isolated and scalable model than the MR1 system where a singular JobTracker does all the resource management, scheduling and task monitoring work.

StreamBase CEP is a complex event processing platform for the rapid building of systems that analyze and act on real-time streaming data for instantaneous decision-making, and combines a rapid application development environment, an ultra low-latency high-throughput event server, and connectivity to real-time and historical data. It was founded in 2003 to commercialize the research done by the Aurora Project, and they support interconnection with Hadoop since 2012. Using this interconnection, they can use Apache Flume to gather information, and Apache Hadoop to perform the batch jobs. In this work, it represents several commercial CEP applications, such as Progress Apama, IBM Infosphere Streams and Oracle CEP.

Apache Chukwa is a scalable distributed system for monitoring and analysis of log-based data. Log processing was one of the original purposes of MapReduce. Unfortunately, Hadoop is hard to use for this purpose. Writing MapReduce jobs to process logs is somewhat tedious and the batch nature of MapReduce makes it difficult to use it with logs that are generated incrementally across many machines. Furthermore, its filesystem still does not support appending to existing files. Chukwa is a Hadoop subproject that bridges the gap between log handling and MapReduce. Some of the durability features include agent-side replication of data to recover from errors.

Apache Flume is a distributed system for collecting log data from many sources, aggregating it, and writing it to the Hadoop filesystem. It is designed to be reliable and

Name	Release Year	Description	DBMS	DSMS	CEP	Distributed
Google Photon [3]	2013	Distributed stream processing system		●		●
Walmart Muppet [14]	2012	Distributed event processing system		●		●
StreamDrill [23]	2012	Stream processing system			●	
SAP HANA [8]	2011	In-memory database ‡	●	●		●
Apache Storm [15]	2011	Distributed stream processing system		●		●
Apache YARN [25]	2011	Distributed general-purpose processing system †	●	●	●	●
Apache Flume [9]	2011	Distributed stream processing system ‡		●		●
Apache Kafka [13]	2011	Distributed stream processing system ‡		●		●
Apache S4 [18]	2011	Distributed event processing system.		●		●
Apache Chukwa [20]	2010	Distributed stream processing system †		●		●
HStreaming [11]	2010	Distributed stream processing system †		●		●
AMPLab Spark [26]	2010	Distributed general-purpose processing system ‡	●	●		●
VoltDB [24]	2010	In-memory distributed database ‡	●	●		●
Esper [7]	2006	Complex Event Processing System			●	
StreamBase CEP [22]	2003	Distributed complex event processing system ‡			●	●
SQLstream [21]	2003	Distributed stream processing system ‡		●		●

† Tools based on Hadoop's infrastructure
‡ Tools that can interact with Hadoop's infrastructure

Table 1. List of event processing tools and his main characteristics

highly available, while providing a simple, flexible, and intuitive programming model based on streaming data flows. Flume and Chukwa share similar goals and features. However, there are some notable differences. Flume maintains a central list of ongoing data flows, stored redundantly in Zookeeper [12]. In contrast, Chukwa distributes this information more broadly among its services.

SAP HANA is a general purpose and ANSI standards-compliant in-memory database. Because of its design, it allows transactional and OLAP reporting in a single system. It is deployable as an on-premise appliance, or in the cloud, best suited for performing real-time analytics, and developing and deploying real-time applications.

Apache Storm is a scalable, fault-tolerant distributed computation system. Similar to tools like Apache S4, it provides a distributed stream processing system, based on a particular architecture, formed by elements called bolts and spolts, that together form a processing in topology design. They argue that the system does for realtime processing what Hadoop did for batch processing.

Apache Kafka is a distributed publish-subscribe messaging system. It is designed to provide high throughput persistent messaging that is scalable and allows for parallel data loads into Hadoop. Its features include the use of compression to optimize I/O performance and mirroring to improve availability, scalability and to optimize performance in multiple-cluster scenarios.

AMPLab Spark is a general-purpose open source in-memory cluster computing system. Its main difference is that its system is based on an in-memory model, providing better throughputs than Hadoop-based systems. They offer a set of tools that resembles various tools on the Apache YARN ecosystem, such as Shark (an Apache Hive similar tool), GraphX (an Apache Giraph similar tool) and Spark Streaming (a tool for stream processing similar to Apache Storm).

VoltDB is an ACID-compliant in-memory database. It represents a new type of databases that focus on maintain the guarantees that traditional relational databases offer, but also provides a scalable and fault-tolerant system.

Apache S4 is a general-purpose distributed computation system. It focuses on providing a middleware for the development of applications that process continuous unbounded streams of data, and a way similar to tools like Apache Storm. As well as many other systems, it uses Apache Zookeeper to coordinate its distributed jobs in a fault-tolerant way.

Walmart Muppet is a stream processing tool, similar to Apache S4 and Apache Storm. It focuses on distributed parallel processing of streams of continuously flowing data, proposing a model called MapUpdate, which is a MapReduce based model for stream processing.

HStreaming is a general-purpose distributed data analytics platform for streaming data. It runs over Hadoop and aims to provide real-time processing over streams in a more performant manner than Hadoop's common MapReduce jobs.

SQLstream is an in-memory data-analytics platform. It operates in a similar way as SAP Hana and AMPLab Spark, but focusing mainly on stream processing. As VoltDB, the system provides queries in standard SQL language, and executes the queries before the data arrives in data warehouse systems.

Esper is a complex event processing system. The system enables rapid development of applications that process large volumes of data, that could be incoming messages or real-time streamings.

Streamdrill is a complex event processing system. It is oriented for real-time data analytics, focusing mainly on the top-k problem. The problem consists of continuously updating a user generated query with streaming tuples, generating a trending list with k updated entries.

4. Conclusions and Future Works

In this paper, we discussed the evolution in the event processing field. We showed that the path of development of these tools has changed since by integrating the MapReduce model into the event processing domain.

However, it is not possible to determine if these implementations will converge into greater sets of tools of general-purpose systems, offering specific capabilities for each kind of dataset to be processed. We could perceive that general-purpose distributed systems are aiming to offer information flow processing into their toolsets, as well as event-oriented systems are offering or aiming to offer distributed processing capabilities to their systems.

What we can perceive until now is that the existing set of tools for information flow processing are growing in the last years, driven mainly by the needs of the growing big data oriented applications.

Our future work will focus on the performance comparison between tools described in this survey paper, aiming to expose their bottlenecks and potentials to be applied to high performance information flow processing applications.

References

[1] D. J. Abadi, Y. Ahmad, Balazinska, et al. The design of the borealis stream processing engine. In *CIDR*, volume 5, pages 277–289, 2005.

[2] D. J. Abadi, D. Carney, Çetintemel, et al. Aurora: a new model and architecture for data stream management. *The VLDB Journal - The Int. Journal on Very Large Data Bases*, 12(2):120–139, 2003.

[3] R. Ananthanarayanan, V. Basker, S. Das, A. Gupta, et al. Photon: Fault-Tolerant and Scalable Joining of Continuous Data Streams. In *Proc. of SIGMOD Int. Conf. On Management Of Data*, pages 577–588, New York, NY, USA, 2013.

[4] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah. Telegraphcq: continuous dataflow processing. In *Proc. of the 2003 ACM SIGMOD Int. Conference on Management of Data*, pages 668–668. ACM, 2003.

[5] G. Cugola and A. Margara. TESLA: a formally defined event specification language. In *Proc. of the Fourth ACM Int. Conf. on Distributed Event-Based Systems*, pages 50–61. ACM, 2010.

[6] G. Cugola and A. Margara. Complex event processing with t-rex. *Journal of Systems and Software*, 85(8):1709–1728, 2012.

[7] esper. `http://esper.codehaus.org/`. Accessed in July 2013.

[8] F. Färber, S. K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner. SAP HANA Database: Data Management For Modern Business Applications. *ACM Sigmod Record*, 40(4):45–51, 2012.

[9] Apache Flume. `http://flume.apache.org/`. Accessed in July 2013.

[10] Apache Hadoop. `http://hadoop.apache.org`. Accessed in July 2013.

[11] hstreaming. `http://www.hstreaming.com/`. Accessed in July 2013.

[12] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: Wait-Free Coordination for Internet-Scale Systems. In *Proc. of the 2010 USENIX Conf. on USENIX Annual Technical Conf.*, volume 8, pages 11–11, 2010.

[13] J. Kreps, N. Narkhede, and J. Rao. Kafka: A Distributed Messaging System for Log Processing. In *Proc. of the NetDB*, 2011.

[14] W. Lam, L. Liu, S. Prasad, A. Rajaraman, Z. Vacheri, and A. Doan. Muppet: Mapreduce-style processing of fast data. *Proceedings of the VLDB Endowment*, 5(12):1814–1825, 2012.

[15] J. Leibiusky, G. Eisbruch, and D. Simonassi. *Getting Started With Storm*. O'Reilly Media, Inc., 2012.

[16] D. C. Luckham, J. J. Kenney, L. M. Augustin, J. Vera, D. Bryan, and W. Mann. Specification and analysis of system architecture using rapide. *Software Engineering, IEEE Transactions on*, 21(4):336–354, 1995.

[17] A. Margara and G. Cugola. Processing flows of information: from data stream to complex event processing. In *Proc. of the 5th ACM Int. Conf. on Distributed Event-based Systems*, pages 359–360. ACM, 2011.

[18] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed Stream Computing Platform. In *IEEE Int. Conf. on Data Mining Workshops*, pages 170–177. IEEE, 2010.

[19] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. A Comparison of Approaches to Large-Scale Data Analysis. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 165–178. ACM, 2009.

[20] A. Rabkin and R. Katz. Chukwa: A system for reliable large-scale log collection. In *Proceedings of the 24th international conference on Large installation system administration*, pages 1–15. USENIX Association, 2010.

[21] sqlstream. `http://www.sqlstream.com/resources/`. Accessed in July 2013.

[22] StreamBase: Complex Event Processing. `https://www.streambase.com/wp-content/uploads/downloads/datasheets/StreamBase\_Brochure\_General\_Overview.pdf`. Accessed in July 2013.

[23] streamdrill. `https://streamdrill.com/`. Accessed in July 2013.

[24] L. VoltDB. VoltDB Technical Overview, 2010.

[25] Apache YARN. `http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html`. Accessed in July 2013.

[26] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing With Working Sets. In *Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing*, pages 10–10, 2010.