© 2016 IJSRSET | Volume 2 | Issue 1 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Fault Tolerance Aware Low Latency Event Detection for Complex Event Processing System – A Review

A. Gokila*, R. Janani, G. T. Kalaiarasi

Department of Computer Science and Engineering, Park College of Engineering and Technology, Kaniyur, Tamilnadu, India

ABSTRACT

Data gathering in the real world environment becomes the most complex process where the multiple sensors and devices are exists. This enormous number of sensors and devices gathers the large number of data's parallely which leads to complex event processing system. Latency is the biggest problem in the handling of complex event processing system which is resolved in the existing work by introducing the pattern sensitive partitioning model in which latency of the complex event processing system can be reduced considerably with the concern of low parallelization degree. Existing work do not concentrate on handling of fault tolerance where the failure is most concerned issue in the parallel event processing system. Event failed then it is required to process it from the start which would increase the latency more. Problem is overcome in the proposed methodology by introducing the novel approach called the failure aware latency reduced complex event processing system. The experimental tests conducted were proves that the proposed methodology of this work provides better result than the existing work in terms of improved system performance.

Keywords: Latency Event Detection, Hadoop, Big Data, HDFS, RFID, GPS

I. INTRODUCTION

Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the strictures of your database architectures. To gain value from this data, you must choose an alternative way to process it.

The hot IT buzzword of 2012, big data has become viable as cost-effective approaches have emerged to tame the volume, velocity and variability of massive data. Within this data lie valuable patterns and information, previously hidden because of the amount of work required to extract them. To leading corporations, such as Walmart or Google, this power has been in reach for some time, but at fantastic cost. Today's commodity hardware, cloud architectures and open source software bring big data processing into the reach of the less well-resourced. Big data processing is eminently feasible for even the small garage startups, who can cheaply rent server time in the cloud.

\

1.1 Hadoop in Big Data

Hadoop has become a central platform to store big data through its Hadoop Distributed File System (HDFS) as well as to run analytics on this stored big data using its MapReduce component. Many of us would have certainly heard about Big Data, Hadoop and analytics. The industry is now focused primarily on them and Gartner identifies strategic big data and actionable analytics as being among the Top 10 strategic technology trends of 2013.

According to the Gartner website: 'Big Data is moving from a focus on individual projects to an influence on enterprises strategic information architecture. Dealing with data volume, variety, velocity and complexity is forcing changes to many traditional approaches. This realisation is leading organisations to abandon the concept of a single enterprise data warehouse containing all information needed for decisions. Instead, they are moving towards multiple systems, including content

management, data warehouses, data marts and specialised file systems tied together with data services and metadata, which will become the logical enterprise data warehouse.

There are various systems available for big data processing and analytics, alternatives to Hadoop such as HPCC or the newly launched Red Shift by Amazon. However, the success of Hadoop can be gauged by the number of Hadoop distributions available from different technological companies such as IBM Info Sphere Big Insights, Microsoft HD Insight Service on Azure, Cloudera Hadoop, Yahoo's distribution of Hadoop, and many more.

1.2 Introduction To The Project

Today billions of sensors and smart objects, i.e., objects with embedded electronics that enable identification, sensing, and actuation capabilities, are deployed throughout the globe, collecting data about the physical world, such as smart meters, global positioning system (GPS) sensors, and RF identification (RFID) tags. They grow in numbers and have the power to enable new applications in the areas of smart homes, smart cities, environmental monitoring, health-care, smart business and security, paving the way toward the Internet of Things (IoT). In doing so, many applications encompass a high and fluctuating number of entities or events to be monitored. For instance, in smart grid applications like demand response, hundreds of thousands of events per second is emitted from millions of consumers, with relevant load changes varying by up to a factor of seven over time. Similar fluctuations can be observed in traffic monitoring: official traffic statistics from the California Department of Transportation1 show that in one single hour ("rush hour") up to 25% of the total daily traffic volume can occur on streets. In order to support automated, timely reactions of smart objects to situations of interest occurring in the physical world, intelligence is needed that enables their consistent detection by integrating and continuously analyzing low-level sensor streams. Consistency in this regard means that neither false positives nor false-negatives should be detected. Complex event processing (CEP) is a key paradigm that helps in realizing such intelligence. It allows for specifying a graph of multiple dependent operators that step-wise transform low-level sensor data to complex events that correspond to the situations of interest, e.g., critical power grid situations or traffic jams. The CEP systems further provide means to execute the operators in a distributed manner to ensure an efficient utilization of the available resources.

However, traditional CEP systems are not yet capable of supporting low latency event detection for large-scale IoT applications. Delays in event detection can have several causes, such as delays imposed by communication, processing, and buffering of events. While for communication and processing latencies, bounds can be found that can be met with high probability, unlimited buffering delays can occur when an operator is overloaded and therefore needs to buffer an unlimited amount of events. In this regard, current CEP systems are not providing parallelization methods that can yield sufficient throughput and lack methods to adapt their resources to workload fluctuations. Thus, they cannot guarantee keeping a buffer limit and this way lack supporting low latency event detection. This shortcoming can have drastic consequences and dramatically reduces the benefits an IoT application can draw from a CEP system. Therefore, it is of critical importance to develop CEP systems that can guarantee a buffer limit even under high and fluctuating workloads.

II. METHODS AND MATERIAL

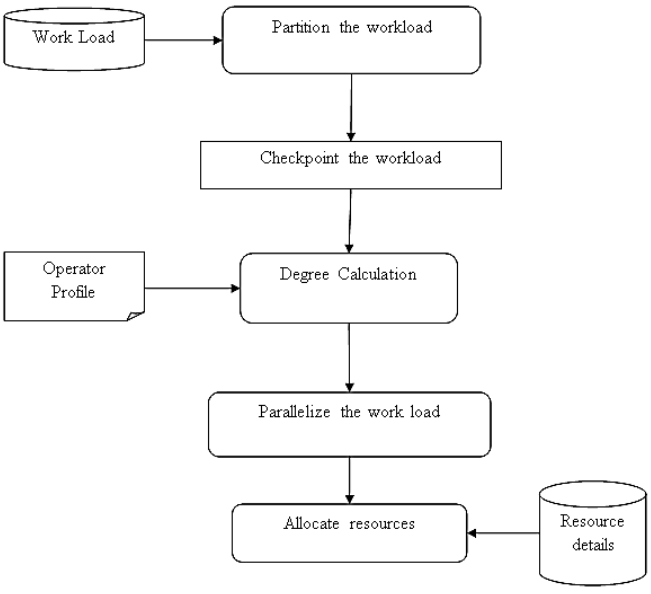

Figure 1 : System Models

A. Existing Work

The existing work make CEP operators capable of keeping a predictable buffer limit. Our contributions are

threefold. 1) It propose a novel pattern-sensitive stream partitioning model. The partitioning model allows to consistently parallelize a wide class of CEP operators and ensures a high degree of parallelism. 2) It propose methods to model the workload and dynamically adapt the parallelization degree utilizing queuing theory (QT), so that a buffering limit of each operator can be met predictably. The evaluation shows that the proposed stream partitioning methods can achieve a high throughput of up to 380 000 events per second even on commodity hardware. 3) Moreover, It is show in the context of a traffic monitoring scenario that the adaptation methods enforce even under heavily fluctuating workloads a stable parallelization degree and this way ensure that the buffering limit is met and only little over-provisioning in terms of resources is required.

B. Proposed Work

In the proposed work, the problem of providing users both the ability to see early results as motivated by online query processing is concentrated and achieve a low expected total runtime. It seeks to enable intra-query fault-tolerance without blocking and we want to do so in a manner that minimizes the expected total runtime in the presence of failures. Other objective functions could also be useful (e.g., minimize runtime without failures subject to a constraint on recovery time.) It chooses to minimize the sum of time under normal processing and time in failure recovery. This function combines high-performance at runtime with fast failure recovery into a single objective. It wants to minimize this function while preserving pipelining.

FTOpt is an optimized for our heterogeneous fault-tolerance framework. FTOpt runs as a post-processing step: it takes as input (a) a query plan selected by the query optimizer and annotates it with the fault-tolerance strategies to use. The optimizer also takes as input (b) information about the cluster resources and cluster failure model, and (c) models for the operators in the plan under different fault-tolerance strategies. FTOpt produces as output a fault-tolerance plan that minimizes an objective function (i.e., the expected runtime with failures) given a set of constraints (that model the plan). FTOpt's fault-tolerance plans have three parts: (1) a fault-tolerance strategy for each operator, (2) checkpoint frequencies for all operators that should checkpoint their states, and (3) an allocation of resources to operators. We assume a given resource allocation to operators. For a given query plan, the optimizer's search space thus consists of all combinations of fault-tolerance strategies. In this paper, we use a brute-force technique to enumerate through that search space and leave more efficient enumeration algorithms for future work. For each such combination, FTOpt estimates the expected total runtime with failures, the optimal checkpoint frequencies and, optionally, an allocation of resources to operators. It then chooses the plan with the minimum total runtime with failures.

III. RESULTS AND DISCUSSION

Design Goals

- System setup is done to handle the complex event processing system in terms of the various scheming points.
- Proposed partitioning model is to split the incoming event stream by selections, it contains the patterns to be detected, so that it refers to the model as pattern-sensitive stream partitioning.
- Workload monitoring and prediction based on operator profiling.
- Processing time of an instance often depends on the workload; operator aggregating values in a time-based window, higher event rates mean that the processed windows contain more events, so that the processing time increases.
- FTOpt runs as a post-processing step it takes as input a query plan selected by the query optimizer and annotates it with the fault-tolerance strategies.

IV. CONCLUSION

Complex event processing plays a critical role in the IoT environment which leads to an efficient handling of tasks which are submitted by the users. Decrease the latency of the system in case of presence of failures also. It is achieved by enabling the check pointing mechanism which would avoid total re-execution by check pointing the last failure point status of every complex event processing system. Latency is the biggest problem arises in the IoT environment which is tolerated in the existing work by introducing the parallel complex event processing system which will reduce the latency in the considerable manner. The objective of this work is to decrease the latency of the system in case of presence of

failures also. It is achieved by enabling the check pointing mechanism which would avoid total re-execution by check pointing the last failure point status of every complex event processing system.

V. FUTURE ENHANCEMENTS

Failure that may arise in the complex event processing system is tolerated by introducing the failure aware complex event processing system. The experimental tests conducted were proves that the proposed research work provides the better result than the existing work in terms of improved performance.

VI. REFERENCES

[1] Buchmann. A and Koldehofe. B, Eds.(2009), IT-Information Technology. Munich, Germany: Oldenbourg Verlag , vol. 51.

[2] Chlamtac. I, Miorandi. D, Pellegrini. F. D and, Sicari. S (2012), 'Internet of Things: Vision, applications and research challenges', Ad Hoc Netw.,vol. 10, no. 7, pp. 1497–1516.

[3] Cao. B, Giakkoupis. M, Prasanna. V. K, and Simmhan. Y (2011), 'Adaptive rate Stream processing for smart grid applications on clouds', in Proc. 2nd Int.Workshop Sci. Cloud Comput. (ScienceCloud'11), pp. 33–38.

[4] Cugola. G and Margara. A (2010), 'Tesla: A formally defined event specification language', in Proc. 4th ACM Int. Conf. Distrib. Event-Based Syst.(DEBS'10), pp. 50–61.

[5] Cugola. G and Margara. A (2012), 'Processing flows of information: From data Stream to complex event processing', ACM Comput. Surv. vol. 44, no. 3, pp. 15:1–15:62.

[6] Chakravarthy. S and Mishra. D(1994), 'Snoop: An expressive event specification language for active databases,' Data Knowl. Eng., vol. 14, no. 1, pp. 1–26.

[7] Charalambous. T, Fiscato. M, Kalyvianaki. E, and Pietzuch. P. (2012),'Overload management in data stream processing systems with latency guarantees', in Proc. 7th IEEE Int. Workshop Feedback Comput.,.

[8] Fernandez. R. C., Gal. A, Pietzuch. P and Weidlich. M (2014), 'Scalable stateful Stream processing for smart grids', in Proc. 8th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS'14), pp. 276–281.

[9] Jerzak. Z and Ziekow. H (2014),'The debs 2014 grand challenge', in Proc. 8th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS'14), pp. 266–269.

[10] Koch G. G, Koldehofe. B, and Rothermel. K (2010), 'Cordies: Expressive event Correlation in distributed systems' , in Proc. 4th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS'10), pp. 26–37.