High availability of data using Automatic Selection Algorithm (ASA) in distributed stream processing systems

Sultan Alshamrani, Hesham Alhumyani, Quadri Waseem, Isbudeen Noor Mohamed
Faculty of College of Computers and Information Technology, Taif University, Kingdom of Saudi Arabia

Article Info

Article history:

Received Dec 1, 2018
Revised Jan 31, 2019
Accepted Feb 21, 2019

Keywords:

Automatic Selection Algorithm
Distributed Stream Processing Systems
High availability
Recovery

ABSTRACT

High Availability of data is one of the most critical requirements of a distributed stream processing systems (DSPS). We can achieve high availability using available recovering techniques, which include (active backup, passive backup and upstream backup). Each recovery technique has its own advantages and disadvantages. They are used for different type of failures based on the type and the nature of the failures. This paper presents an Automatic Selection Algorithm (ASA) which will help in selecting the best recovery techniques based on the type of failures. We intend to use together all different recovery approaches available (i.e., active standby, passive standby, and upstream standby) at nodes in a distributed stream-processing system (DSPS) based upon the system requirements and a failure type). By doing this, we will achieve all benefits of fastest recovery, precise recovery and a lower runtime overhead in a single solution. We evaluate our automatic selection algorithm (ASA) approach as an algorithm selector during the runtime of stream processing. Moreover, we also evaluated its efficiency in comparison with the time factor. The experimental results show that our approach is 95% efficient and fast than other conventional manual failure recovery approaches and is hence totally automatic in nature.

Corresponding Author:

Sultan Alshamrani,
College of Computers and Information Technology,
Taif University. P.O. Box 888 Al-Hawiya, Taif,
Zip Code: 21974, Kingdom of Saudi Arabia.
Email: susamash@tu.edu.sa

1. INTRODUCTION

With increased vogue of streams and stream processing engines, the continuous data flow management and processing has become easier for the end users. These engines process a set of continuous queries and generate the continuous output on the fly. However, these stream processing engines (SPE) are often prone to failures, which is always a big and direct threat for High Availability, Real-time stream execution and Fault Tolerance [1, 2].

Stream computing [3] involves the computations for analytic purpose in its best way using the stream processing engines (SPEs) of different types discussed in [4]. These are the engines that can generate huge and big data streams continuously on the fly in a cluster of commodity servers. Currently, the most popular recovery techniques for availability of data is active, passive and upstream but the current setup lacks the automatic selection of recovery mechanisms based on the best suit recovery policy. Since each recovery techniques have its own advantages and can be used for specific failures, therefore saving time and maintaining the efficiency can be an advantage over the previous state of the art. This will improve the overall performance in distributed stream processing engines. Our proposed automatic selection algorithm (ASA) deals with the automatic selection of a recovery algorithm without human interventions and hence saving the time and increasing the performance.

Therefore, we design an automatic selection algorithm (ASA) for choosing the best recovery technique from a group of available recovery options. We evaluated the feasibility of our approach by implementing it with IBM BlueMix Stream Application. We use the data of New York City Dot as stream data for our experiment. In most of the modern distributed processing systems, they can recover automatically from failures [5], but none gives the flexibility to choose the best recovery algorithm from a set of available recovery algorithms to automate the recovery.

Our automatic selection algorithm chooses the most appropriate recovery algorithm related to specific type of failures from a set of available algorithms during the real-time failure scenarios, which increases the performance of recovery by means of automation and by means of doing it at high speed. In this work, we focus on automation and performance of the recovery mechanisms. The automatic selection algorithm which will work as a selector to choose the best recovery algorithm based on CPU utilization criteria. We evaluate our automatic selection algorithm on distributed stream setup and find the effective results accordingly. First, we saved the intermediate results as a distributed cache for support of recovery in our experiment. Then we use our automatic selection algorithm based on a distributed cache during the real-time stream processing. The purpose of the distributed cache is to provide input streams during the failure. Only the failed node should be restarted, and other nodes can operate normally taking the input streams. We have applied the automatic selection algorithm (ASA) to distributed stream processing engine and illustrate the effectiveness of our proposed approach. Experimental results show that our approach achieves effective and faster recovery tactics than other conventional failure recovery approaches.

In summary, this paper makes the following contributions as 1) we propose an automatic selection algorithm (ASA) for selection of recovery at a single node.2) we also accelerate the recovery speed (efficiency in comparison with time factor) of ASA as an enhancement in our experiments. The rest of this paper is organized as follows: In Section 2, is the background. Section 3 describes the automatic selection algorithm. Section 4 we depict our evaluation, followed by Section 5 which gives our conclusion.

2. BACKGROUND

2.1. Data streams

Datastream is a sequence of data packets used to transmit or receive information [6]. For better understanding, we assume a schema ‘S1’ describes a data, consisting of attributes with ordered timestamp set τ. Then a stream is defined as an infinite multiset of elements < si, τi > where (si is a tuple of the stream with schema S1 and τi ∈ τ the timestamp). Furthermore, many stream elements share the common timestamp [7]. Therefore, the data stream is a continuous sequence of attribute-value tuples that all come from some pre-defined schema and continuously output in a flow [8]. Moreover, they are unpredictable and may arrive by millions per seconds [7] so are hence known as data stream processing systems (DSPSs) [9].

2.2. Stream processing systems and their applications

Data streams are received from online sources. The application domain using these streams are always used for real-time events for a plenty of purposes. Another important concern is the high availability of these systems. To provide high availability, replication and backups of the stream processing play a vital role [1], [10-12]. The stream-based applications [13] and the stream-processing concept is nowadays a letst trend for research [14]. The concern fields of the research in stream processing include digital signal processing and complex event processing [11].

2.3. Distributed stream processing systems and framework

Many stream-based applications are distributed naturally and are controlled by many different computing devices [13]. During the process of distribution, these system scales out [15]. A DSPS receive input from the end user, distribute the tasks and send back the result to end user [16]. Furthermore, the high availability, fault tolerance and scalability are critical requirement in distributed stream processing to overcome data loss [1], [13], [15], [17], [18]. A typical distributed stream-processing system should offer several benefits: It should enable high-availability in case of failures [19] and should allow multiple node scaling and balance with load spikes.

2.4. High availability in distributed stream processing systems

High availability (HA) is a critical issue for stream processing systems to provide continuous and uninterrupted operation against failures such as machine crash [20, 21]. Recovery techniques are commonly used technique to provide high availability and data processing guarantees [3, 22]. As in financial data analysis [20], they accept non-deterministic results [23]. Hence different approaches are used for the high availability purpose which provide different types of recovery semantics [8].

2.5. Recovery and types

Regular backups are always the best option to recover the lost data or to go back to the previous stable state. There are many factors involved with backups which we can't ignore and can be handy to choose the backup type for the recovery [17]. Those factors are a time of backups, runtime overheads, and cost of backups.

The range of recovery for stream processing engines ranges from ''no information loss'' to ''some information loss''. The general recovery methods are categorized as Gap Recovery, Precise Recovery and Rollback Recovery [18]. 1) Precise Recovery: In precise recovery only increase in latency is visible. 2) GAP Recovery: In Gap recovery, the loss of information is expected, and it provides the weakest guaranteed processing. 3) Rollback Recovery: In Rollback recovery, the information is preserved and not lost. It provides increase in latency [5, 18]. Rollback recovery can be achieved using Active Standby, Passive Standby and Upstream backups. A) In active standby, both primary and secondary nodes are given the same input at a given time. Active standby is best to provide repeating recovery for deterministic networks and divergent recovery for non-deterministic networks [15]. Active standby gives a shorter recovery time with a similar amount of overhead [20]. B) In passive standby, the primary task is capable to saves its state periodically to a permanent shared storage. The passive standby provides repeating recovery for deterministic networks and divergent recovery for non-deterministic networks. C) In upstream backup/amnesia, the nodes store the tuples until the downstream nodes acknowledge them. Upstream backup provides repeating recovery for deterministic networks and divergent recovery for non-deterministic networks. Hence, it gives the weaker guarantee with lower operational costs and lower network overhead [18]. Upstream backup provides gap recovery with the least overhead [24].

In Paper [8] it is concluded that each algorithm has its advantages and can be used for different types of recovery [18]. Active standby is the master for fastest recovery among all, but it costs high. It is best for the environments where fast failure recovery is needed and where minimal disruption justifies higher costs. On the other hand, the Passive standby is the master to provide precise recovery in arbitrary query networks. The Upstream backup is master and well suited for an environment where failures are infrequent and short recovery delays are can be managed. It has lower runtime overhead but a longer recovery time that mostly rely on the size of the query-network state.

However, these recovery strategies operate independently on its own and the current DSPS are therefore unable to provide all fastest recovery, precise recovery and a lower runtime overhead at the same time. There is a need to combine all the three strategies together. No doubt, there are several challenges that need to be solved to combine all these together. First, there is a need to provide an automated mechanism that would be able to identify the right recovery technique amongst the three based on the system requirements. Second, there is a need to prioritize recovery method in the event of multiple failures.

The main goal is to provide high availability and data processing guarantee. To achieve this goal, we need the best recovery mechanism in case of failures. High availability & data processing guarantees imply data should be available and safe in case of failures. To recover from the failures, we should use the best recovery guarantee and the best recovery approaches available.

We intend to use together with different recovery approaches (i.e., active standby, passive standby and upstream standby) at nodes in a distributed stream-processing system (DSPS) based upon criteria like (system requirement) e.g. OLTP requires the fastest recovery when compared to OLAP where recovery does not have to be fast). We intend to provide all the benefits of fastest recovery, precise recovery and a lower runtime overhead in a single solution. The aim is to gain benefit from all the available techniques because we know each technique has its own advantages and is used for specific propose. Our aim is to explore a new strategy that can automatically determine the most suitable recovery method based on system requirements which will enhance performance in the form of recovery speed (efficiency will be increased in comparison with time factor i.e. value change factor).

3. AUTOMATIC SELECTION ALGORITHM

3.1. Problem definition

The problem of selecting the best algorithm arises in a plenty of situations. We define our stream failure as Sf={list*, states*}. The list* denotes a list of algorithms available which can be used for recovery. The states* represents all information for failure recovery like active recovery as +, passive recovery as -, upstream recovery as 0 and a lower runtime overhead). At the same time, we evaluate the efficiency regarding time factor for Automatic selection algorithm during runtime failures.

3.2. Automatic selection algorithm

After a single node failure, we assign values to the input parameters, which is a basic requirement of the selection algorithm to work. These parameters are chosen wisely to satisfy and achieve the objectives of the problem using the simple way. We include the basic model of algorithm selection based on the criteria of the algorithms [25]. Here, the performance measure characteristic is based on the criteria parameter. We enhanced the model by increasing its performance by increasing its efficiency in comparison with the time factor. Hence this will accelerate the recovery speed.

For our experiments as mentioned previously, we want to use our designed algorithm selection in selecting the recovery algorithm from a set of available algorithms based on the CPU utilization criteria. The aim to automate the recovery selection and increase the performance by means of time feature. The results will give the best selection of algorithms and will also increase the performance of selection by means of time factor i.e. value change factor. This is an enhanced added feature to the basic selection model. The purpose of the overall experiments is to increase the high availability of data using automatic selection algorithm (ASA) in distributed stream processing systems. Therefore, we implement the basic model with enhancements on selection problems of [26]. We describe the basic abstract model in Figure 1. The overview of the flow is represented as:

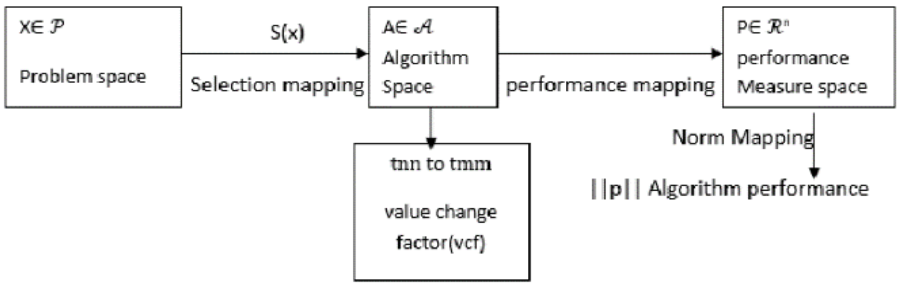


Figure 1. Segregating the basic selection algorithm parameters (criteria) with Time Factor (value change factor)

The objective is to determine S(x) to achieve have high algorithm performance.

Definitions used:

P as 𝒫=Problem space or collection
x=Member of P problem to be solved
A=Algorithm space
A as 𝒜=Member ofA, algorithm applicable to problems from P
Rn as ℛn=n-dimensional real vector space 0 f performance measures
P=Mapping from A x P to R n determining performance measure

Algorithm Selection Problem criteria for S(x):

For selection mapping S(x), we use the basic criteria for the selection and we present our basic model based on the best selection criteria match as below:

Selection based on criteria:

Choose that selection mapping B(x) which gives maximum performance for each problem:

$\|P(B(x), x)\| \leq \|P(A, x)\|$ for all $A \in \mathcal{A}$

3.3. Enhancement of time factor to increase performance

As mentioned in Figure 2, the tmm and tnn are the value change factors of performance. Its value is the time period to obtain the result (Final). Hence the total time factor value is represented as:

$$\begin{matrix} t11 & t21 & tmm \\ t12 & t22 & t2m \\ tnn & t2n & tmm \end{matrix}$$

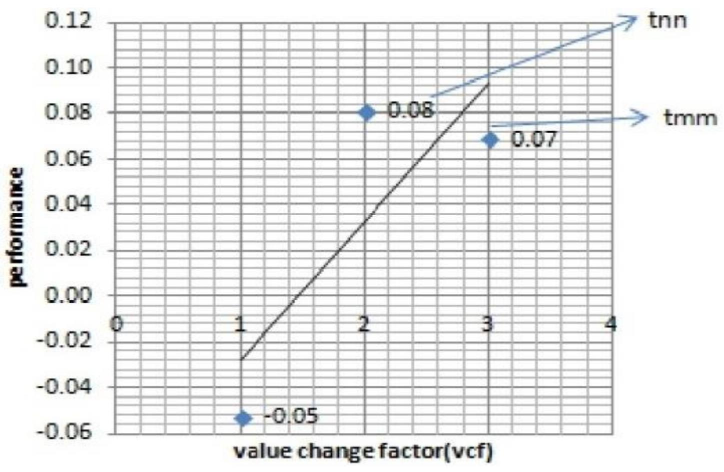


Figure 2. Performance enhancement after adding the time factor to the basic selection model

This provides the improvement for fast recovery and fast selection using criteria parameters with time factor ranging from t11 to t22 as tnn to tmm. The ranging values are already mentioned and declared. There is a difference of 10 in value in each time factor. The values of some of these parameter measures can be improved by enhancing the basic algorithm regarding time parameter using value change factor (vcf). Hence by scaling the measures to a standard value of (say 0, +1 and -1) which is based upon the type of algorithm as (active, passive or upstream). For this type of selection and enhancement, we purpose, automatic selection algorithm as:

Algorithm: Automatic Selection Algorithm

```
Input: A stream set as Sf={list*}, CPU-Count, max value set
Output: The value for {states*}, where states are denoted as +, - and 0
1 failure existence FE, vcf-value change factor
2 Sf={list*, states*}, Sf is the data stream
3 Set range for vcf
4 Value range -tmm, tnn
5 if CPU-count is < max value set
6 return 0
7 else if
8 if CPU-count > max value set
9 return +1
10 else
11 return -1
12 end
```

A common approach to this algorithm selection model is to differentiate and select the algorithm from a set of algorithms in the portfolio. This model is added to the performance of the corresponding algorithm based on the value change factor (vcf) criteria and range. The algorithm results in best selection using algorithm selector and hence enhance the performance accordingly. In Figure 3, the automatic selection algorithm selects the best option using the selector and hence enhance the performance directly. The standard scaling measures (0, +1 and -1) are choosen options for selector in the form of (active, passive or upstream) algorithms. Table 1 represents the parameter values for different types of algorithms (active, passive and upstream).

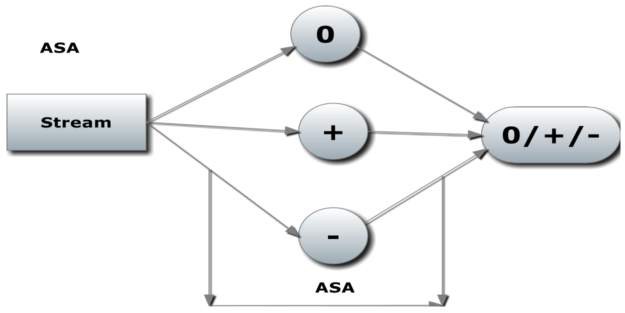


Figure 3. Algorithm selector in our experiments

Table 1.CPU and segregation based on implementation

Algorithm Type	CPU Load Per Input Rate	Max value set	CPU Load Threshold	Min value set	Basic Response time	Value Range
Active	0.5x	0.5x	5	5	100	0
Passive	9.5	9.5x	95	95	50	+1
Upstream	9.9	9.9	100	5	20	-1

Algorithm Type: Represents the type of recovery algorithm available. We have mentioned them here as active recovery, passive recovery and upstream recovery.
CPU Load Per Input Rate: The CPU load after executing this algorithm which is represented as a function f(x), where x=input rate.
Max Value Set: It is the maximum value a recovery algorithm can achieve.
CPU Load Threshold: The threshold of CPU load (in %) above which the execution time (i.e., latency per tuple) of the algorithm on this device type is expected to increase rapidly.
Min Value Set*:* It is the minimum value a recovery algorithm can achieve.
Basic Response time*:* It is a performance time parameter.

The algorithm solver selects the recovery type based on the CPU utilization and the matching type of parameters. The parameters are used to determine the equivalence of the algorithm. The average runtime of CPU utilization is counted in seconds; the number of instances solved by time limit is an enhancement. Table 2 represents the stream process CPU utilization and the value representation of algorithms used.

Table 2. Segregated values based on criteria

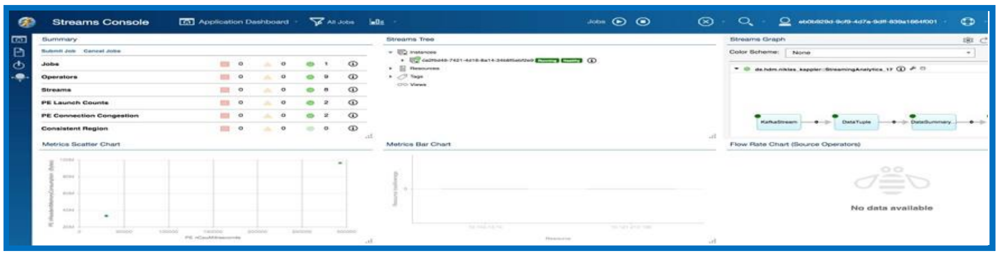
Data flow	0(common)	0(common)	0(common)
Steam process	Min CPU utilization	Max CPU utilization	Query
Possibilities	Active	Passive	upstream
Values	0	+1	-1

4. EVALUATION

Before starting our implementation, we make the following assumptions: 1) we consider the failures in the form of (software bugs, hardware errors, failures at nodes); 2) after the failure of a task node, it cannot take its upstream records or output results to the concern downstream task nodes, and its state fails categorically.

4.1. Experimental setup

Experimental setup includes a utilization of IBM BlueMix Stream Application. The NYC Traffic Sample demo data comes from a real-time traffic website created by the New York City Department of Transportation (DOT). We push the New York city Dot data source through multi-thread to Bluemix message queue. Figure 4 shows the stream console of IBM BlueMix showing our implementation.

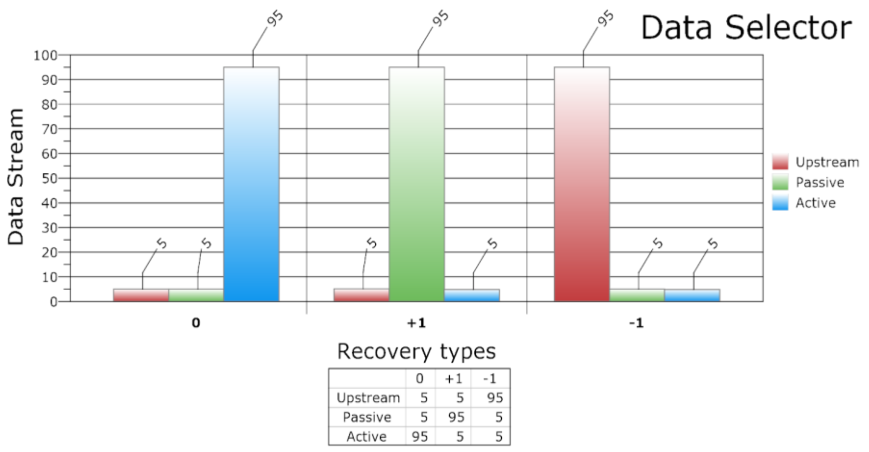


Jobs	0	0	1	ⓘ
Operators	0	0	9	ⓘ
Streams	0	0	8	ⓘ
PE Launch Counts	0	0	2	ⓘ
PE Connection Congestion	0	0	2	ⓘ
Consistent Region	0	0	0	ⓘ

Figure 4. IBM BlueMix stream implementation

4.2. Impact and evaluation of our experiment

During the runtime of the real-time traffic speed data, we stop a different number of task nodes and then assume the scenario of failures. Then we try to measure the impact of our automatic selection algorithm (ASA) simultaneously on failures. More importantly, we try to recover using three available options (Active, Passive and Upstream) by the implementation of our automatic selection algorithm (ASA). We then observe the following results: First, ASA provides a much faster impact and was the best filter in selecting the best option for recovery based on the CPU criteria. In Figure 5, the experimental results show that ASA algorithm achieved 95x faster enhancement and automated recovery than conventional manual setup.



	0	+1	-1
Upstream	5	5	95
Passive	5	95	5
Active	95	5	5

Figure 5. Result of our ASA for selection

5. CONCLUSION

The most critical requirements of big data in distributed stream processing systems (DSPS) is the high availability of data. This can be achieved by recovery mechanisms. We try to implement our automatic selection algorithm with time enhancement on all nodes. By doing this the stream processing systems were able to choose the best recovery algorithm from a set of available algorithms on the basis of CPU utilization criteria. For a future work, our algorithm for selection can be used through machine learning concepts for more accuracy.

ACKNOWLEDGEMENTS

We would like to thank Taif University, Saudi Arabia for providing the financial support on this project.

REFERENCES

[1] B. Gedik, S. Schneider, M. Hirzel and K. Wu, "Elastic Scaling for Data Stream Processing," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1447-1463, June 2014.

[2] Balazinska, M., Balakrishnan, H., & Stonebraker, M. "Load management and high availability in the Medusa distributed stream processing system". *In Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM. 2004, June. pp. 929-930
[3] Sebastian Raschka. "Model evaluation, model selection, and algorithm selection in machine learning Part I-The basics", arxiv. 11 Jun 2016.
[4] R. Ranjan, "Streaming Big Data Processing in Datacenter Clouds," in *IEEE Cloud Computing*, vol. 1, no. 1, pp. 78-83, May 2014.
[5] Kamburugamuve, S., & Fox, G. Survey of the distributed stream. Grids Ucs Indiana Edu. 2013.
[6] Golab, Lukasz, and M. Tamer Özsu. "Issues in data stream management." ACM Sigmod Record 32.2 (2003): 5-14.
[7] Repantis, T., & Kalogeraki, V. "Replica placement for high availability in distributed stream processing systems". In *Proceedings of the second international conference on Distributed event-based systems ACM*. July 2008: pp. 181-192.
[8] J. -. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker and S. Zdonik, "High-availability algorithms for distributed stream processing," *21st International Conference on Data Engineering (ICDE'05)*, Tokoyo, Japan, 2005, pp. 779-790.
[9] Hirzel, M., Soulé, R., Schneider, S., Gedik, B., & Grimm, R. "A catalog of stream processing optimizations". *ACM Computing Surveys (CSUR)*, 2014; 46(4), 46.
[10] Kamburugamuve, S., Fox, G., Leake, D., & Qiu, J. Survey of distributed stream processing for large stream sources. Grids Ucs Indiana Edu. 2013.
[11] Gu, Y., Zhang, Z., Ye, F., Yang, H., Kim, M., Lei, H., & Liu, Z. "An empirical study of high availability in stream processing systems". In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*. Springer-Verlag New York, Inc. November 2009; p. 23.
[12] Z. Zhang *et al.*, "A Hybrid Approach to High Availability in Stream Processing Systems," *2010 IEEE 30th International Conference on Distributed Computing Systems*, Genova, 2010, pp. 138-148.
[13] Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., & Zdonik, S. B. "Scalable Distributed Stream Processing". In *CIDR*. January 2003; Vol. 3: pp. 257-268.
[14] Tatbul, N., Çetintemel, U., & Zdonik, S. (2007, September). Staying fit: Efficient load shedding techniques for distributed stream processing. In *Proceedings of the 33rd international conference on Very large data bases* (pp. 159-170). VLDB Endowment.
[15] Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles ACM.* November 2013; pp. 423-438
[16] H. Wang and L. Peh, "MobiStreams: A Reliable Distributed Stream Processing System for Mobile Devices," *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, Phoenix, AZ, 2014, pp. 51-60.
[17] Q Huang, & Lee, P. P. "Toward high-performance distributed stream processing via approximate fault tolerance". *Proceedings of the VLDB Endowment*, 2016; 10(3), 73-84.
[18] Balazinska, M., Balakrishnan, H., & Stonebraker, M. "Load management and high availability in the Medusa distributed stream processing system". In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data ACM*. June 2004: pp. 929-930.
[19] J. Hwang, Y. Xing, U. Cetintemel and S. Zdonik, "A Cooperative, Self-Configuring High-Availability Solution for Stream Processing," *2007 IEEE 23rd International Conference on Data Engineering*, Istanbul, 2007, pp. 176-185.
[20] Heinze, T., Aniello, L., Querzoni, L., & Jerzak, Z. (). "Cloud-based data stream processing". *In Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems ACM*. May 2014: pp. 238-245.
[21] Stonebraker, M., Çetintemel, U., & Zdonik, S. "The 8 requirements of real-time stream processing". *ACM Sigmod Record*, 2005; 34(4), 42-47.
[22] Kwon, Y., Balazinska, M., & Greenberg, A. "Fault-tolerant stream processing using a distributed, replicated file system". *Proceedings of the VLDB Endowment*, 2008; 1(1), 574-585.
[23] Balazinska, M., Balakrishnan, H., Madden, S., & Stonebraker, M. Fault-tolerance in the Borealis distributed stream processing system. June 2005. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data ACM*. pp. 13-24.
[24] Gedik, B., Andrade, H., Wu, K. L., Yu, P. S., & Doo, M. SPADE: The system s declarative stream processing engine. *In Proceedings of the 2008 ACM SIGMOD international conference on Management of data ACM*, June 2008: 1123-1134.
[25] Kotto-Kombi, R., Lumineau, N., Lamarre, P., & Caniou, Y. Parallel and distributed stream processing: systems classification and specific issues. 2015.
[26] Rice, John R. "The algorithm selection problem." Advances in computers. Vol. 15. Elsevier, 1976. 65-118.

BIOGRAPHIES OF AUTHORS

DR. Sultan S Alshamrani is currently working as an assistant professor at Taif University in Saudi Arabia and he is the Chairman of the committee of faculty members affairs and the Chairman of the committee of Capstone Projects at the department of Information Technology. DR. Sultan got his PhD from the University of Liverpool in UK and a master's degree in Information Technology (Computer Networks) from the University of Sydney in Sydney, Australia. DR. Sultan finished his bachelor's degree in computer science from Taif University in 2007 with General Grade" Excellent" With first honor and an accumulative GPA of (4,85) out of (5,00) where considered the highest GPA in the collage

Dr. Hesham Alhumyani earned his B.Sc. (2009) in Computer Engineering from Umm Al-Qura University. His M.Sc. (2013) and Ph.D. (2016) degree was obtained in Computer Science and Engineering from the University of Connecticut - USA. He has been a member of UConn's Underwater Sensor Networks Lab from 2012 to 2016. Currently, he is an assistant professor in Computer Engineering department at Taif University – Saudi Arabia. His research interest is in Underwater Wireless Sensor Networks, optimization, Internet-Of-Things and recentrly has joined cloud computing research team at Taif University.

Quadri Waseem received the Bachelor Degree in Computer Application (BCA) from Kashmir University, J&K and the Master of Computer & Management (MCM) degree from Pune University, India. Previously, he had worked with Oracle Corporation as a Fusion Middleware Administrator.He is Currenty working as a Faculty at the College of Computers and Information Technolgy, Taif University, Saudi Arabia.His primary research interests are in Big Data, Parallel and Distributed Systems, High-Performance Computing, IOT, and Deep Learning.