

**SCATTER-GATHER BASED APPROACH IN SCALING
COMPLEX EVENT PROCESSING SYSTEMS FOR
STATEFUL OPERATORS**

Sriskandarajah Suhothayan

(168268V)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

February 2019

**SCATTER-GATHER BASED APPROACH IN SCALING
COMPLEX EVENT PROCESSING SYSTEMS FOR
STATEFUL OPERATORS**

Sriskandarajah Suhothayan

(168268V)

Thesis submitted in partial fulfillment of the requirements for the degree Master of
Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

February 2019

DECLARATION

I declare that this is my own work and this MSc project report does not incorporate without acknowledgment any material previously submitted for degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or another medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date:

Name: Sriskandarajah Suhothayan

We certify that the declaration above by the candidate is true to the best of our knowledge and that this report is acceptable for evaluation for the CS6997 MSc Research Project qualifying evaluation.

Supervisors

.....
Dr. H. M. N. Dilum Bandara

.....
Dr. Srinath Perera

.....
Date

.....
Date

ABSTRACT

With the introduction of Internet of Things (IoT), scalable Complex Event Processing (CEP) and stream processing on memory, CPU, and bandwidth constraint infrastructure have become essential. While several related work focuses on replication of CEP engines to enhance scalability, they do not provide expected performance while scaling stateful queries for event streams that do not have pre-defined partitions. Most of the CEP systems provide scalability for stateless queries or for the stateful queries where the event streams can be partitioned based on one or more event attributes. These systems can only scale up to the pre-defined number of partitions, limiting the number of events they can process. Meanwhile, some CEP systems do not support cloud-native and microservices features such as startup time in milliseconds.

In this research, we address the scalability of CEP systems for stateful operators such as windows, joins, and pattern by scaling data processing nodes and connecting them as a directed acyclic graph. This enabled us to scale the processing and working memory using the scatter and gather based approach. We tested the proposed technique by implementing it using a set of Siddhi CEP engines running on Docker containers managed by Kubernetes container orchestration system. The tests were carried out for a fixed data rate, on uniform capacity nodes, to understand the processing capacity of the deployment. As we scale the nodes, for all cases, the proposed system was able to scale almost linearly while producing zero errors for patterns, 0.1% for windows, and 6.6% for joins, respectively. By reordering events the error rate of window and join queries was reduced to 0.03% and 1% while introducing 54ms and 260ms of delays, respectively.

ACKNOWLEDGMENT

I would like to take this opportunity to express my deep sense of gratitude and a profound feeling of admiration to my project supervisors. Many thanks go to all those who helped me in this work. My special thanks to the University of Moratuwa for giving an opportunity to carry out this research project.

I would like to gratefully acknowledge Dr. Dilum Bandara, the internal project supervisor, for his continuous guidance and support throughout the whole duration of the project, under whose supervision that I gained a clear concept of what I should do. I would also like to extend my heartfelt gratitude to Dr. Srinath Perera, the external supervisor of the project, for sharing the experiences and expertise with the project matters. Last but not least, I thank all those who like to remain anonymous although the help they provided to me was valuable.

Thank you.

TABLE OF CONTENTS

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGMENT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	viii
LIST OF ABBREVIATIONS	ix
1. INTRODUCTION	1
1.1 Background	1
1.2 Motivation	1
1.3 Problem statement	3
1.4 Objectives	3
1.5 Outline	4
2. LITERATURE REVIEW	5
2.1 Complex Event Processing Systems	5
2.2 CEP functionalities	6
2.2.1 Filtering events based on attributes	6
2.2.2 Aggregation on sliding windows	6
2.2.3 Joining multiple streams	7
2.2.4 Pattern matching and sequence detection	7
2.3 Understanding characteristic of single node CEP	8
2.4 Distributed architectures for scaling CEP nodes	9
2.4.1 Running multiple CEP nodes in a cluster	10
2.4.2 Distributing different type of queries to different CEP nodes	10
2.4.3 Distributing execution via Publish/Subscribe infrastructure	11
2.4.4 Distributing events by partitioning each stream	12
2.4.5 Distributing events as batches	14
2.5 Distributing CEP operations over multiple CEP nodes	14
2.5.1 Scalability of stateless operators	15
2.5.2 Scalability of stateful operators	16
2.6 Summary	20

3. PROPOSED SOLUTION	22
3.1 Proposed solution	22
3.1.1 Scaling window operators	22
3.1.2 Scaling pattern operators	25
3.1.3 Scaling join operators	29
3.2 Summary	32
4. IMPLEMENTATION	33
4.1 Scaling window operators	33
4.2 Scaling of pattern operators	37
4.3 Scaling join operators	40
4.4 Summary	41
5. PERFORMANCE EVALUATION	43
5.1 Data set	43
5.2 Experimental setup	44
5.3 Analysis on system scalability	45
5.3.1 Analysis on scalability of window operation	45
5.3.2 Analysis of pattern operation scalability	51
5.3.3 Analysis on scalability of join operation	54
5.4 Analysis on latency and accuracy	57
5.5 Applicability to other CEP systems	60
5.6 Summary	61
6. SUMMARY	62
6.1. Conclusion	62
6.2. Research limitations	63
6.3. Future work	64
REFERENCES	66

LIST OF FIGURES

Fig. 2.1: Overview of CEP	5
Fig. 2.2: Vertical scaling with multiple CEP nodes	10
Fig. 2.3: Distributed deployment of Oracle CEP	12
Fig. 2.4: Anatomy of Kafka Topic	13
Fig. 2.5: Scaling stateless CEP queries	16
Fig. 2.6: Parallelizing operator graph using partitions	17
Fig. 2.7: Combining partitioning and pipelining	17
Fig. 2.8: Optimization on streaming aggregation	18
Fig. 2.9: StreamCloud query parallelization strategy	19
Fig. 3.1: Scaling sliding time window	23
Fig. 3.2: Scaling pattern based on Brenna at el	26
Fig. 3.3: Scaling pattern based on stream type	26
Fig. 3.4: Scaling pattern based on distributed streams	27
Fig. 3.5: Scaling pattern by replicating distributed streams	27
Fig. 3.6: Scaling join of small and large windows	30
Fig. 3.7: Scaling join of two large windows	30
Fig. 4.1: Deployment of standard sliding time window test	33
Fig. 4.2: Deployment of scalable sliding time window test	34
Fig. 4.3: Deployment of standard pattern test	38
Fig. 4.4: Deployment of scalable patterns based on streams	38
Fig. 4.5: Deployment of scalable pattern based on distributed streams	38
Fig. 4.6: Deployment of standard join test	41
Fig. 4.7: Deployment of scalable join test	41
Fig. 5.1: Throughput of 1, 3, 5, 9 and 17 node time windows	46
Fig. 5.2: Memory consumption of 1, 3, 5, 9 and 17 node time windows	46
Fig. 5.3: CPU utilization of 1, 3, 5, 9 and 17 node time window	47
Fig. 5.4: Maximum time interval supported by the number of nodes	48
Fig. 5.5: Event consumption throughput by the number of nodes while supporting maximum time interval	48

Fig. 5.6: Average number of events stored in each window node	48
Fig. 5.7: Average bandwidth of window processing nodes (events/Sec)	49
Fig. 5.8: Average CPU utilization of window processing nodes	49
Fig. 5.9: Maximum window length supported by the number of nodes	50
Fig. 5.10: Maximum supported pattern matching duration for worse-case workload	51
Fig. 5.11: Maximum supported pattern matching duration for average-case workload	52
Fig. 5.12: Average throughput of each pattern state node	53
Fig. 5.13: Average CPU utilization of each pattern state node	53
Fig. 5.14: Maximum large window length of the join nodes	55
Fig. 5.15: Average bandwidth of the join nodes	55
Fig. 5.16: Maximum length of each join window	56
Fig. 5.17: Average join node bandwidth while holding the largest possible length window	56
Fig. 5.18: Average latency of the sliding time and length windows	57
Fig. 5.19: Average latency of simple pattern	58
Fig. 5.20: Average latency of small and large window and two large window joins	58

LIST OF TABLES

Table 2.1: Symbols used to analyze CEP characteristics	8
Table 2.2: Baseline characteristics of single node CEP engine	9
Table 2.3: Comparison on distributed architectures for scaling CEP nodes	15
Table 2.4: Characterization summary of distributed CEP operations over multiple CEP nodes	21
Table 3.1: Summary of distributed stateful CEP operations over multiple CEP nodes for streams that cannot be partitioned by a key	31
Table 5.1: Accuracy and performance analysis of window queries	59
Table 5.2: Accuracy and performance analysis of join queries	60

LIST OF ABBREVIATIONS

ATM	Automated teller Machine
CEP	Complex Event Processor
CPU	Central processing unit
DBMS	Database Management System
ESB	Enterprise Service Bus
GC	Garbage Collection
IoT	Internet of Things
JMS	Java Messaging Service
NFA	Non-deterministic Finite Automata
RDD	Resilient Distributed Dataset
TCP	Transmission Control Protocol
TPS	Transactions Per Second
XA	eXtended Architecture