DESIGN AND IMPLEMENTATION OF A DATA STREAM MANAGEMENT SYSTEM WITH ADVANCED COMPLEX EVENT PROCESSING CAPABILITIES

A Thesis

by

Erdi Ölmezoğulları

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Engineering

Özyeğin University
June 2013

DESIGN AND IMPLEMENTATION OF A DATA STREAM MANAGEMENT SYSTEM WITH ADVANCED COMPLEX EVENT PROCESSING CAPABILITIES

Approved by:

Professor İsmail Arı, Advisor
Department of Computer Engineering
Özyeğin University

Professor Hasan Sözer
Department of Computer Engineering
Özyeğin University

Professor Ekrem Duman
Department of Industrial Engineering
Özyeğin University

Dr Salih Ergüt
AveaLabs
Avea

Date Approved: 19 June 2013

To My Parents, Brother and Close Friends

ABSTRACT

The world has seen proliferation of data stream applications over the last years. These applications include computer network monitoring, Radio Frequency Identification (RFID)-based supply chain and traffic management systems, e-trading, online financial transactions, web click-streams, some mobile communication applications, and civilian or military applications using sensor networks. All of these applications are considered "mission-critical" by related organizations and require real-time stream processing to detect simple or complex events, so that strategic decisions can be made quickly. An emerging system architecture called Data Stream Management System (DSMS) is well-suited to address the analysis needs of emerging data stream applications. DSMS forms the basis for our project and allows processing of high-speed data streams with different continuous queries. In this thesis, we present design and implementation details of a data stream management system with advanced complex event processing (CEP) capabilities. Specifically, we add "online" Association Rule Mining (ARM) and testing capabilities on top of an open-source DSMS system and demonstrate its capabilities over fast data streams. Our most important findings show that online ARM can generate (1) more unique rules, (2) with higher throughput, (3) much sooner (lower latency) than offline rule mining. In addition, we have found many interesting and realistic musical preference rules such as **"George Harrison ⇒ Beatles"**. We demonstrate a sustained rate of 15K rows/sec per core. We hope that our findings can shed light on the design and implementation of other fast data analytics systems in the future.

ÖZETÇE

Son yıllarda dünyada veri akışı uygulamalarında hızlı bir artış görülmüştür. Bu uygulamalara örnek olarak bilgisayar ağ gözleme sistemleri, radyo frekanslı kimlik tanıma (RFID) temelli tedarik zinciri ve trafik yönetim sistemleri, e-ticaretler, çevrimiçi finansal işlemler, web (örün) tıklama-akışları, bazı mobil komünikasyon uygulamaları ve sensör ağları kullanan sivil-askeri uygulamalar verilebilir. Bütün bu uygulamalar etkileri açısından ilgili kurumlarca "kritik görev" addedilmekte ve gerçek-zamanlı işleme tabi tutularak içlerindeki basit ve karmaşık olayların hızla bulunması istenmektedir. Amaç stratejik kararların çabuk alınmasıdır. Projemize temel teşkil eden Veri Akışı Yönetim Sistemleri (VAYS) mimarisi yüksek hızlı akışların farklı sürekli sorgularla bellek içinde hızla işlenebilmesini sağlamakta ve ortaya çıkan yeni uygulama alanlarının veri analiz ihtiyaçlarına daha iyi cevap verebilmektedir.

Günümüzde bilişim dünyası faydalı bilgiye ulaşma yolunda "büyük veri" problemleri (verinin kütlesi, hızı, çeşitliliği, tutarsızlığı) ile baş etmeye çalışmaktadır. Bu makalede, büyük veri akışları üzerinde İlişkisel Kural Madenciliği'nin (İKM) daha önce literatürde yapılmamış bir şekilde "çevrimiçi" olarak gerçeklenme detayları ile başarım bulguları paylaşılacaktır. En önemli bulgularımız çevrimiçi kural çıkarımı sayesinde: (1) çevrimdışı kural çıkarımından çok daha fazla kuralın, (2) çok daha hızlı ve etkin olarak, ve (3) çok daha önceden hesaplanabileceği gösterilmiştir. Ayrıca müzik tercihlerine uygun **"George Harrison ⇒ The Beatles"** gibi pek çok ilginç ve gerçekçi kural bulunmuştur. Sonuçlarımızın ileride diğer büyük veri analitik sistemlerinin tasarım ve gerçeklemesine ışık tutacağını ummaktayız.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest appreciation and sincere gratitude to my advisor to Dr. İsmail Arı for giving me the opportunity to work with in Cloud Computing Research Group in Özyeğin University. I believe that I have improved my skills and perspective thank to his suggestions, supports, comments and deepest knowledge during master education, since 2011. I have realized so many times that he is a brilliant and innovative academician, and this thesis would not be completed without his considerations and helps.

I would like to thank to all of my instructors in Özyeğin University for their innovative lectures and special attentions during masters. Since I had learned new innovative richest practical and theoretical subjects in computer science. They taught me new, innovative, rich practical and theoretical subjects in CS.

I would like to thank to also my friends in Cloud Computing Research Groups, labs people, and all of graduate students in Özyeğin University. Especially, I would like to share special thanks with my close desk friend, Uğur Koçak, since he motivated me with his jokes, considerations, and coffee breaks in my last term. Now, I have perfect morale thanks to his help.

Last of all to you my precious family, who always support and encourage me to reach the best I can. Love you.

I would also like to send special thanks to our sponsors. My masters research has been partially sponsored by European Union FP7 Marie Curie Program BI4MASSES Grant, Turkish National Institute of Science and Technology (TUBITAK) Grant 190E194, IBM Shared University Research program, Turkish Telecomm and Avea Labs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Computer systems and the middleware connecting these systems generate increasingly more messages among themselves to interact. Financial sector, defense industry, health-care sector, transportation, telecommunication & communication, military, etc. are all establishing their critical operations around real-time systems. Results of one or more operation can affect other operations. Thus, the dependencies of causes & effects of operations have to have to be turned into patterns and knowledge using real-time intelligent systems. The obtained knowledge plays a crucial role on fast decision making. The business people try to answer questions such as;

- *Which are the most popular items (e.g., products, brands, stock etc.) ?,*
- *Which items are associated with which other items?,*
- *How do certain conditions and actions (e.g., advertisement, web recommendations, and extraordinary economical conditions) affect item popularities and relations?*

Thus, in order to effectively make valuable decisions, Business Intelligence (BI) systems are developed and Complex Event Processing (CEP) engines usually find a place near these systems in the Enterprise Data Warehouse (EDW) architecture.

1.1 Enterprise Data Warehouse (EDW)

The Business intelligence (BI) [8] technologies provide predictive and/or historical operational insights. EDW architecture is composed of Extract Transform Load (ETL), Online Analytical Processing (OLAP), data mining, Complex Event CEP, Business

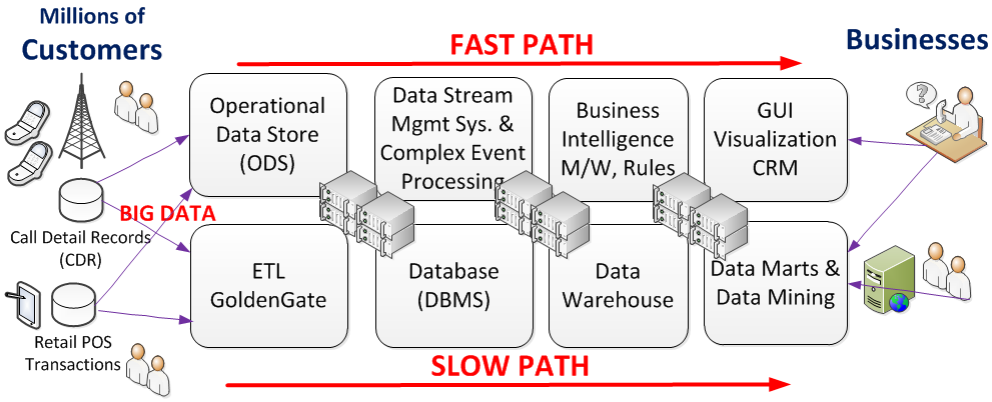


Figure 1: Traditional Enterprise Data Warehouse (EDW) Architecture paths.

Performance Management (BPM), text mining, and other predictive or prescriptive BI tools [9] as shown in Figure 1.

The data received from various resources have to be first processed by *Extraction, Translation, and Load (ETL)* operations. After *"extraction"* the data is converted to new desired formats by different *"translation"* techniques, such as cleaning, encoding, sorting, and aggregations, etc.. Later, the data formatted are load to destinations, such as Data Warehouse (DWH) and Data Marts (DM) by *"loading"* operations.

Figure 2 shows another event-driven data processing architecture called Continuous, Heterogeneous Analysis Over Streams (CHAOS) by Gupta et al.[1]. CHAOS focuses on muti-dimensional analyses, i.e. real-time OLAP cube operations over streams and displaying of results by real-time visualization. In this thesis, we add stream mining capabilities to CEP engines, which is complemetary to OLAP features added by CHAOS.

1.2 Event Driven Architecture (EDA)

The EDA model has been used in various software systems before: (e.g., simple events, implemented in graphical user interface event-driven programs, message-driven applications or application integration scenarios). "After computer scientists had understood the basic principles of events for decades, events have been widely used in

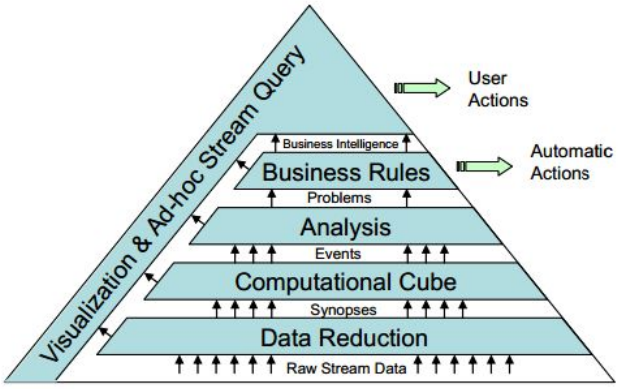


Figure 2: Conceptual Pyramid of Continuous, Heterogeneous Analysis Over Streams (CHAOS) [1]

system software, such as operating systems and network and system management (NSM) tools[10]." Therefore, "The EDA is a design paradigm, in which a software component executes in response to receiving one or more event notifications. EDA is more loosely coupled than the client/server paradigm because the component that sends the notification does not know the identity of the receiving components at the time of compiling [11]." as defined by the Gartner researcher. EDA is constructed on two main components, the cause called *"event"*, and effect is called an *"action"* [12]. An event can be defined as a significant change in states [13]. Its most common and important relationships are time, cause, aggregation) among itself [14]. Let us consider Figure 3, an instance of EDA system, which has A and B events.

Time It denotes the order of events. For example, *event A* occurs before *event B*.

Cause It denotes the dependence of relationship between activities in EDA system. For example, If the activity that signified *event A* had to happen in order for the activity that signified *event B*, then A caused B.

Aggregation It denotes the abstraction. For example, If *Event A* signifies an activity that consists of the activities of a set of events, *B1, B2, B3* then A is an aggregation of all the events in B.[14]

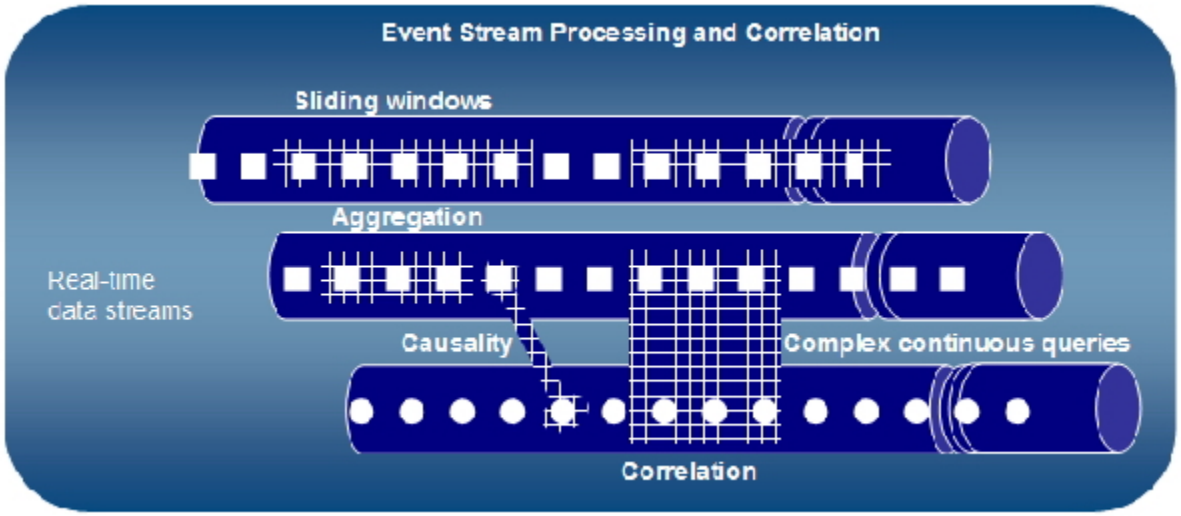


Figure 3: Time, Causality, and Aggregation [2]

Other EDA related concepts and principles are as follows:

Simple Event Processing A notable event occurs, and then, action is initiated. "Simple event processing is commonly used to drive the real-time flow of work-taking lag time and cost out of a business [12]." In other words, ECA (Event-Condition-Action)[1] systems support simple event processing.

Stream Event Processing Both ordinary and notable events occur, and then, events are filtered to raise a significant business event. It is commonly used to drive the real-time flow of information for decision making [12].

Complex Event Processing The notable and ordinary events together with different event types can occur in longer time spans. The event correlation may be causal, temporal or spatial. Therefore, CEP is commonly used to detect and respond to business anomalies, threats by event pattern detection and matching techniques. In other words, CEP requires sophisticated event interpreters [12].

[1] "Event Condition Action (ECA) is a short-cut for referring to the structure of active rules in EDA and active database systems [15]."

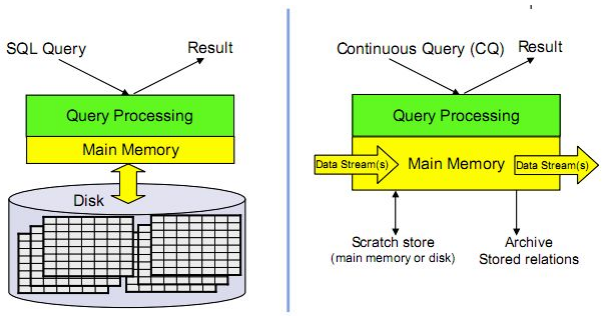


Figure 4: Database Management Systems (DBMS) vs. Data Stream Management Systems (DSMS)

1.3 Data Stream Management Systems(DSMS)

Data Stream Management Systems (DSMS) that can be classified as EDA systems as referred in previous section 1.2. An instance of DSMS consists of four fundamental different components, such as *"events"* (E), *"queries"* (Q), *"windows"* (W), and *"alerts"* (A). On-the-flight data that is incoming from sensors is called *"event"*. Entire operations such as *sampling, storing, routing, enrichment, transformation, matching, parsing*, including *filtering, and pattern detection* can be applied on "events" by using Event Processing (EPL) " continuous queries"

EPL is semantically similar to Structured Query Language (SQL), and both of them are declarative languages. In contrast to DBMS, in DSMS data is temporally stored in main memory in main memory according to be different kinds of "windows". The fundamental differences of between DBMS and DSMS that are shown in Figure 4. In order to handle nonstop continuously incoming "events" from multiple resources over different mediums, real-time ad hoc containers like queue structures that are used with queries as temporarily container. The container structures are called "window" in DSMS. The various kinds of "window" (e.g., time/size based sliding and tumbling) are used by the registered queries. When critical conditions or patterns are detected and caught by the corresponding "query", it triggers "events" and alerts according to conditions and its situations.

Briefly, DSMS benefits mainly some common advantages that are explained in the followings:

- In order to reduce I/O latency and to perform high performance computing, continuous querying is performed on main memory.
- Flexible queries can be run on systems in real-time easily.
- In order to return fast result, infinite and rapidly changing data streams are handled in limited main memory.
- Since querying is in main memory, it is a volatile data storage.
- Events' causes & effects relationships can be instantly monitored by using specific queries (e.g., pattern detection).
- In order to obtain outdated knowledge from raw streaming data, it can be observed on DSMS within real-time, without any data stored in DBMS.

Most DSMS design issues included reducing memory consumption or using it effectively. In this way, data compression and summarization techniques [16] are developed to overcome challenges in implementation phase. *Data-based techniques and task-based techniques* for reducing memory consumption are explained as follows:

Data-Based Techniques It summarizes dataset, and it chooses a subset from the incoming stream to be evaluated. Various data-oriented techniques, such as sampling, load shedding and sketching of data are developed for these goals. After this stage, the reduced data represents by synopsis data structures and aggregation later on.

Sampling Machine Learning techniques and statistical computing have been used according to derived loss function.

Load Shedding If incoming data bursts main memory, sequence of data may be dropped by using different technique (e.g., randomly and semantic drop [17]). Since data is chucked out, its applicability plays most critical roles on stream computing.

Sketching A subset of the features randomly are projected, and thus, irrelevant dimensions can be reduced. For instance, the most popular technique, PCA (principal component analysis) can be used to get better solution.

Synopsis Data Structures In order to select random data points, histograms, wavelets like FFT or sketching are applied moving data in stream. However, these techniques cannot exactly reflect accurate properties of data. It is briefly used to learn general information about incoming data over streams.

Aggregation Statistical means and variance calculations are used to summarize incoming data in this approach.

Task-Based techniques Since DSMS/CEP unveils a new concept in stream computing, traditional existing techniques are tried to revamp, and transformed for stream computing.

Approximation Algorithms Those are used to find approximate solutions to optimization problems that are often associated with NP-hard problems [18]. The algorithms can return an approximate solution with error bounds [16].

Sliding Window It is used to analyze the most recent data items over streams to summarize versions of the old ones [16].

Algorithm Output Granularity The algorithm is used to cope with fluctuating very high data rates according to the available memory and the

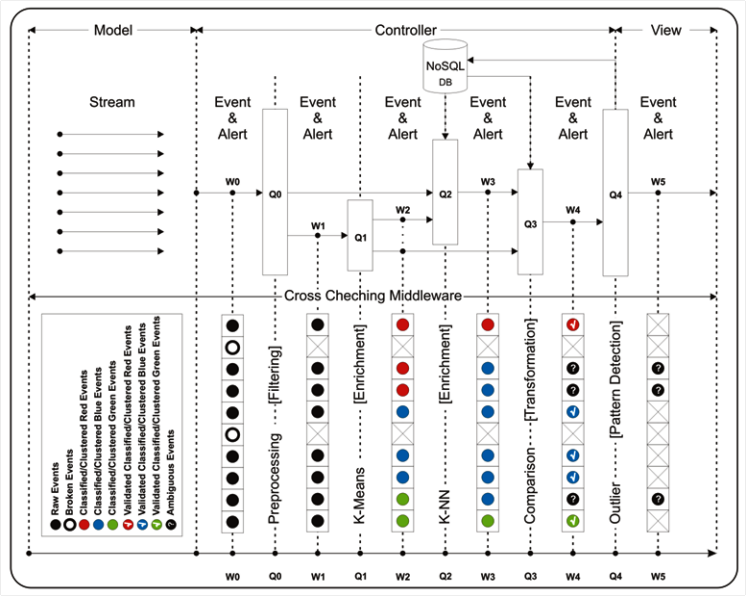


Figure 5: MVC Architecure in DSMS

processing speed represented in time constraints. It consists of fundamental three stages, such as adaptation of resources after mining on data for the first two stages, merging of generated knowledge for the last stage. Algorithm Output Granularity (AOG) has been used in clustering, classification and frequency counting to cope with challenges.

Addition to pros and cons, each EDA system had been constructed according to Model, View, Controller (MVC) design paradigm, and also DSMS has been developed by using the MVC pattern. In an instance of DSMS, Model, View and Controller respectively correspond to Event, Alert, and Query as shown in the Figure 5. Our system architecture has been also developed according to aspects of the MVC pattern. Furthermore, in this thesis, entire processes, including preprocessing and data mining algorithms had been implemented on EPL query-plan. For instance, most popular data mining techniques, such as Association Rule Mining (e.g., Apriori, and FP-Growth)[19] is one of them. We have implemented continuous queries as new aggregation function in EPL over Esper.

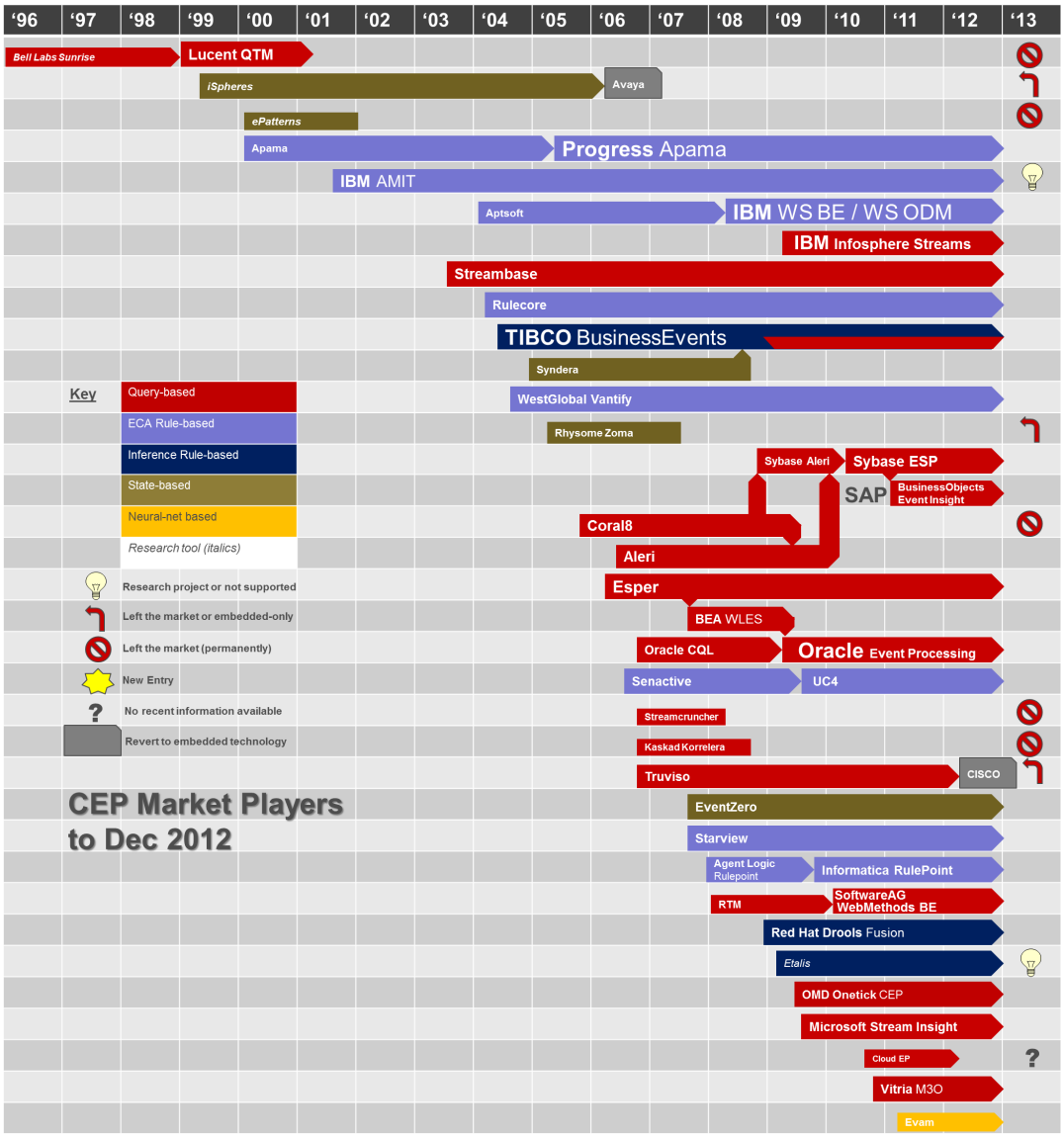


Figure 6: Evolution of CEP and DSMS [3]

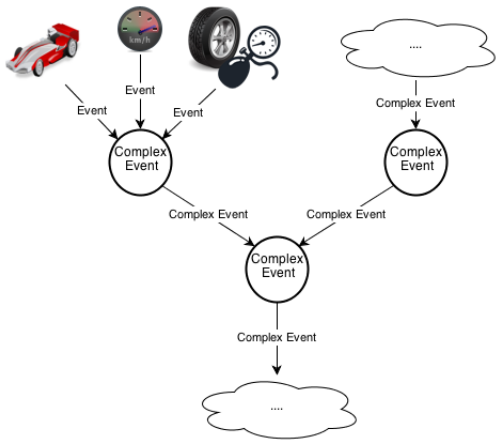


Figure 7: Complex Event Processing (CEP) scenarios

1.4 Current CEP Market and Products

There are a lot of DSMS/CEP engines in stream computing market. In general, their evolutions, market surveying, and commons vs. different properties are illustrated in Figure 6 including the most popular engines such as Avaya, Avaya, Progress-Apama, IBM Streams, Streambase, TIBCO, Eper, etc.. During the writing of this thesis (as of 15 June 2013) TIBCO acquired Streambase and Software AG purchased Apama from Progress. In our study, The Esper DSMS/CEP engine have been used to develop new stream computing (e.g., statistical and data mining algorithms).

1.5 Complex Event Processing (CEP) Scenarios

Let us assume that there is a car, and it has several sensors as depicted in Figure7. Its sensors measure respectively tire pressure, speed of the car, and detect if someone sits on a seat or leaves a seat. According to mounted three sensors on car, let us define some scenarios over them like in the followings.

1^{st} Case :

Action : The tires moves from 45 psi to 41 psi over 15 minutes.

Events : *"lossOfTirePressure"*, and speed event are generated by system.

2^{nd} Case :

Action : The tires drops from 45 psi to 20 psi in 5 seconds.

Events : *"blowOutTire"*, speed event are generated by system.

3^{rd} Case :

Action : The car suffers a blown tires within a very short time, the car may leave road and strike a tree and the driver is thrown from the car.

Events : *"blowOutTire", "zeroSpeed", and "driverLeftSeat"* $\rightarrow$ *"occupantThrownAccident"* are generated by system.

The 3^{rd} case illustrates a composite event (e.g., *"occupantThrownAccident"* called event B) that consists of a group of simple events (e.g., *"blowOutTire", "zeroSpeed" and "driverLeftSeat"* called event A). And also, the relationship between is define as $A \rightarrow B$. In this way, complex events can be generated using of composite events and simple events, and then, interpreted by systems. Thus, various independent events in designed time-critical [20] systems can be used to easily perform intelligent external monitoring application on them, such as supply chain management, surveillance and facility management, healthcare, etc..

1.6 Data Stream Mining

Stream computing, also known as data stream processing has unveiled as a new processing concepts [21], such as incremental algorithms, sublinear algorithms, one-pass and multi-pass algorithms. It has capabilities of processing incoming data streams from tremendous numbers of sensors in a real-time fashion. Since incoming data rate fluctuates wildly, data stream applications have to have low latency that is impossible due to computational finite/limited resources [22]. Due to these, data stream processing faces [21] some challenges. The concept of *"data stream mining"* did not exist before 2000s. Association Rule Mining (ARM) algorithms such as Apriori [23] and FP-Growth [24] were designed for *"offline"* or traditional data mining. Our study utilizes these ARM algorithms, but applies them over streams by integrating them

with a CEP engine and testing their performance with different sliding windows. The tools which claim *"real-time analytics"* capabilities mostly depend on an offline rule extraction phase (i.e. *the training*) using historical data followed by online pattern sequence detection (i.e. *the testing or scoring phase*). We mine the rules themselves in real-time to be able to detect changes in trends and not just detect the existence of already mined historical rules.

Some of the seminal work on pattern and rule mining over streams [25][26][27] only talk about the challenges, models and issues without any specifics of implementation. Issues mentioned include the need for one-pass algorithms and incremental updates among windows. The former issue (one-pass) arises only when the incoming stream rate is much faster than what the stream mining algorithm can process and the latter issue (incremental updates) is only needed if we want to exactly imitate the offline version of the mining algorithm. We demonstrate in this thesis that neither of these may be an issue for many real-life applications.

Frequent pattern mining [25] is not exactly the same task as rule mining, since finding rules requires an additional confidence calculation. In this thesis, we only focus on doing ARM over streaming text log data and do not address processing of already stored (offline) data or encoded media (image and audio/video) types. We also do not discuss other fields of stream mining including stream clustering and stream classification. These topics constitute interesting future work.

ARM over streams can be regarded yet another way of reducing raw data size to extract useful information. Other data-based techniques for reducing raw data size without losing the patterns carried inside include sampling, load shedding, sketching, synopsis and aggregations. An overview of these techniques can be found in [28]. Aggregations over streams can include basic counting and summations as well as more complex mean, variance and correlation [19] calculations. StatStream [29] is among the first in literature to discuss scalable stream correlations by using Discrete Fourier

Transform (DFT) coupled with approximation techniques. In our previous works, we also applied product-moment correlation techniques to spatio-temporal data and shown that such correlations may not strictly guarantee spatial locality [19] [30].

CHAPTER II

BIG DATA

2.1 Big Data Challenges & Big Data Opportunities

1 ZettaByte (Trillion GB) of data exists in the digital universe today, and it is steadily increasing due to sensors, network devices, mobile technologies, servers, etc. as shown in Figure 8. For instance, "Akamai (Content Distribution Network) analyzes 75 million events per day to better target advertisements."[1].In 2008, "Google was processing 20000 TB of data (20 PB) per a day"[2]. In order to cope with fast growing data challenges, new technologies are developed as new solutions.

The challenges of big data management are described as 4Vs *(Volume, Variety, Velocity, and Veracity)* that are briefly explained below.

Volume The volume challenge is defined as the huge sizes of data to be stored and queried.

Velocity The velocity is defined that as speed of in-flight data as it moves from raw data sources to analysis destinations.

Variety The variety is defined as increasing types, format (e.g., structured and unstructured) of data.

Veracity Data are incoming over various medium from different sources within real-time, second by second. Therefore, data may be irrelevant and contain errors (e.g., broken, out of order, missing values, and wrong values). That is, the veracity is defined as accuracy or truthfulness of data.

[1]A Comprehensive List of Big Data Statistics, http://wikibon.org/blog/big-data-statistics/
[2]http://techcrunch.com/2008/01/09/google-processing-20000-terabytes-a-day-and-growing/

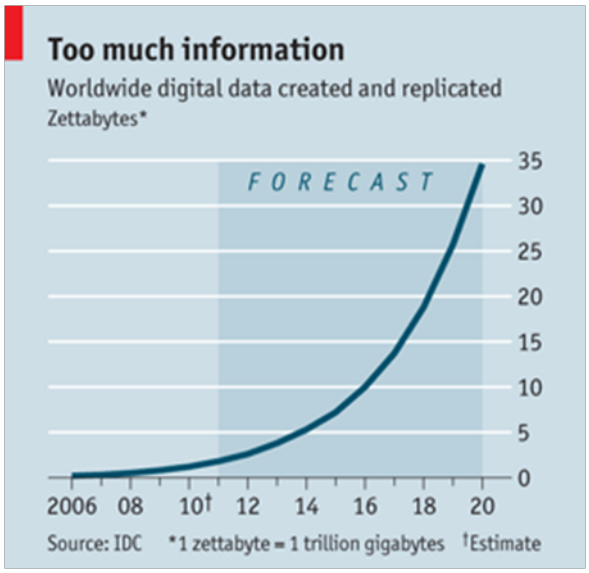


Figure 8: Digital data replication in wordwide, year by year [4]

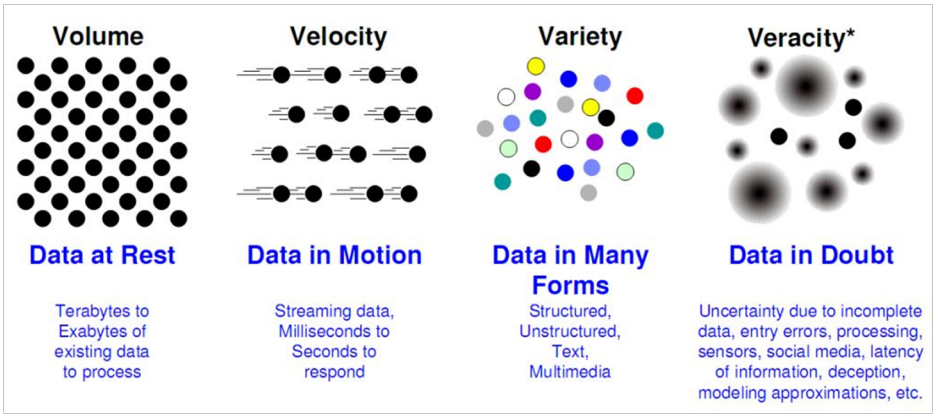


Figure 9: Big Data 4Vs Dimensions [5]

Today, in order to reduce the effects of 4Vs, robust approaches and systems are developed, but some additionally catastrophic problem occurs according to the dependence on fast data dimension. The importance of fast data and its relationship with other dimensions in big data will be laid out in section 2.2.

2.2 Importance of Fast data in Big Data

The term *"Big Data"* is used to refer to challenging data management problems that arise due to the high Volume, Velocity, Variety, and Veracity (4Vs) of the data [31]. "*Fast data* is the *velocity* part of big data" said Tony Baer and thus coined the term. Today, many organizations including social networks, telecommunication operators, media networks, financial institutions and governments generate Terabytes of data each day and attempt to insert this high *"volume"* data into databases that already contain Petabytes. Even worse, data is replicated by different departments for analysis. Data comes from different sources such as sensors, mobile phone call data records, web/system logs, which also results in a wide *"variety"* of data to be processed and stored (e.g., unstructured text, semi-structured XML-JSON, structured CSV, or binary audio/video data). Since "5 Exabytes of data is generated every two days", and its "80% growth is in unstructured form", traditional database management systems cannot cope with this mixed variety of data very well. New distributed data processing frameworks such as Apache Hadoop [32] (with HDFS and MapReduce paradigms) and the NoSQL <*key,value*> stores (e.g., HBase[3] , Cassandra[4] , MongoDB[5], etc.) have been invented to cope with the high volume and variety of data (i.e. big data batches) within the last decade. In addition to these, Mahout that is extended project from Hadoop used in Data Mining solutions to extract unknown critical knowledge by various innovative companies (e.g., Facebook, Twitter, Linkedin, LastFm, etc.).

[3]Apache Hbase project, http://hbase.apache.org/
[4]Apache Cassandra project, http://cassandra.apache.org/
[5]MongoDB by 10gen, http://www.mongodb.org/

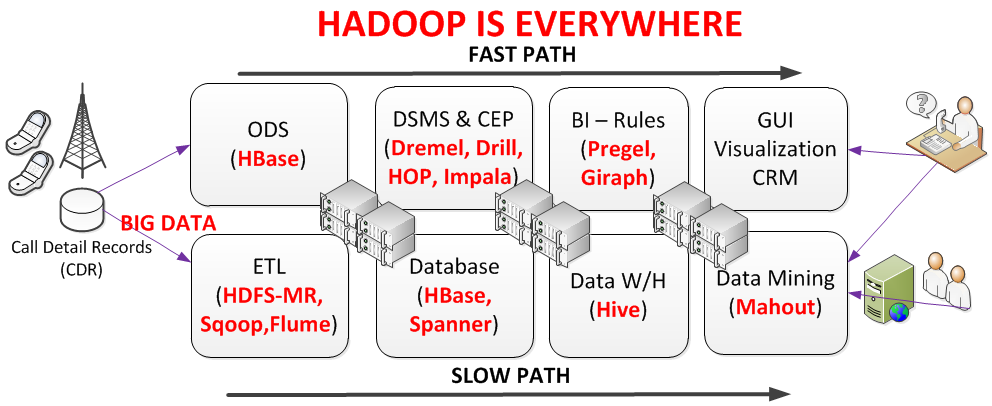


Figure 10: Hadoop and related Apache projects are taking over the enterprise data warehouse (EDW) architecture in both slow and fast paths.

In order to made accelerator appliance such as IBM Netezza[6], Hadoop and similar distributed software that has been combined on hardware, such as Multi-core Network Processor Units (NPU), FPGA, etc.. Therefore, data have been handled to be evaluated and analyzed by Mahout/Hadoop and similar project that eplot big data opportunities.

However, they were neither designed to process data in-flight or fast streaming data, nor were they designed for doing interactive or looping analytical studies. Recently, this deficiency is also being addressed via projects that would be placed in the fast path of the enterprise data warehouse (EDW) architecture as shown in Figure 10. Just to name a few, these projects include Google Dremel [33], Apache Drill, and Hadoop Online Prototype (HOP).

While volume and variety had always been a challenge for data management, its increasing *"velocity"* is the new issue of the 21st century. Attempting to store these data first to analyze them later creates additional costs, unwanted delays to actionable information, and loss of business opportunities. Fortunately, there are now tools to process data on-the-fly as it moves from distributed sources of collection (e.g., sensors and routers) to selected destinations. Since the number of data sources and

[6]IBM Netezza Data Warehouse Appliances, http://www-01.ibm.com/software/data/netezza/

sampling frequencies is steadily increasing, processing in-flight data in-memory is still a challenge. As a result, enterprises increasingly employ Data Stream Management Systems (DSMS) [34] [35] and further want to extend them with complex online analytical and mining capabilities [36]. The resulting software tools are sometimes called Complex Event Processing (CEP) engines in the literature [2]. There are now distributed versions of these continuous stream analysis frameworks such as Yahoo S4[7] and Twitter Storm[8] , in which each query plan operator is placed in a separate processing engine. The benefits of using CEP engines for stream analytics are at least three-fold:

- They can eliminate unwanted data early in the pipeline, saving further CPU, memory, storage and energy costs.
- They can turn raw data into actionable information quickly, thus helping businesses catch profitable opportunities or avoid losses due to fraud or operational inefficiencies.
- They can catch transient or emerging patterns, which never show up in an offline data mining analysis.

[7]Yahoo (and Apache) S4 project http://incubator.apache.org/s4/
[8]Twitter Storm, http://storm-project.net/

CHAPTER III

RULE MINING OVER STREAMS

3.1 Motivation: Why is "online"rule mining critical?

We are at an age, where the trends do not hold for long. Therefore, the temporal aspects of recommendations are extremely important. Unfortunately today, when the streaming data volumes increase, the reaction of data analysts is to increase the minimum support and confidence thresholds to obtain fewer rules with stronger lift (refer to Section 4 for details). Yet, rules that present themselves with high lift values over a long time period (e.g., a month) may have become outdated by the end of that period. For example ice-cream, cold drinks, and plastic cups will be extremely popular for the hottest month of the year, just as sand-bags will be popular before a hurricane. However, there will be no sales opportunity after the trend is gone. These temporally emergent patterns occur even faster for online sales or in stock markets where millions of transactions occur every second. Stream rule mining can extract a trend such as *"when stocks HPQ and MSFT go down by >1%, DELL follows"* within a single hour or day. Assume a recent trend that emerges only in a certain timeframe. Its confidence value may not be the highest globally, but its local business value may be quite high. Our suggested systems are designed to catch these rules timely as they occur. The rest of the thesis is as follows. Section 2 describes the related work and compares them against our proposed methodology. Section 3.2 gives an overview of the ReCEPtor system architecture. Section 3.3 describes the rule mining algorithms. Section 3.3.4 gives performance analysis and results.

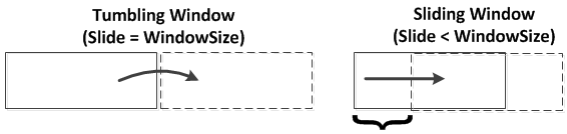


Figure 11: Tumbling vs. Sliding Window semantics

3.2 ReCEPtor System Architecture

DSMS engines [34][35] provide effective queuing, scheduling, time and count-window support, and fast in-memory processing of high-speed, continuous, unbounded data streams. They parse, optimize and execute queries written in declarative languages such as Event Processing Language (EPL) in Esper [2]. EPL syntax and semantics are quite similar to that of Structured Query Language (SQL) in databases, but there are additional clauses such as WINDOWs to support sliding or tumbling window-based analysis over data streams. Figure 11 shows these two types of windows. Sliding-time windows are used to buffer event tuples whose occurrence times fall within a certain time period (e.g., last 1 minute) and to replace events that are older than the time window. The window will move or "slide"in time with a period that is usually smaller than the size of the window and therefore the two event epochs overlap. Similarly, sliding-count windows buffer the last X events. Tumbling windows on the other hand jump to the next epoch by moving as long as the window size, while there is no data overlap between the two event epochs. Other types of windows include Landmark and Damped [26], where the former considers events from a past landmark time until present and the latter gives more weights to recent events. EPL queries can be used for continuous filtering (e.g., ***SELECT*** *x,y* ***FROM*** *Stream <x,y,z>* ***WHERE*** ...) as well as aggregations: algebraic (COUNT, SUM, AVERAGE) or holistic (MIN, MAX). Complex aggregation functions such as TOP-K, DISTINCT, QUANTILES, and SKYLINE can also be implemented. We slightly modified some of the open source ARM algorithms and integrated them with Esper CEP engine. We also implemented new EPL clauses for these algorithms so that they can be used just as easily as any

other window-based (sliding or tumbling) aggregation query. Figure 12 shows the components and high-level architecture of our data stream analytics system. High-speed raw data streams are first subjected to Select and Project operators inside the DSMS to get rid of tuples unwanted further in the data pipeline. The data volume can be reduced row-wise (Select) and/or column-wise (Project). Basically, the preprocessing stage of the stream mining is done at this stage. The rest of the complex analytics and mining clauses are implemented inside the CEP system, which is deployed either as a private or public cloud system. Data from streams can be correlated among each other or with data stored in persistent repositories such as DBMS and NoSQL systems. NoSQL ("Not-only SQL") is a generic term for highly-scalable, distributed data storage and processing engines coupled with a declarative or procedural language for analytics on top. NoSQL systems are used in support of the event queues to extend the correlation windows. New CEP operators developed in this and recent previous work are denoted with the *butterfly* sign in Figure 12 aggregate (correlation) and stream mining (Apriori & FP-Growth). The statistical results and alerts are sent to Business Process Management systems and visualization tools for further actions. The feedback loop shown at the CEP system output denotes the registration of rules learned through predictive mining back into the CEP engine for faster descriptive processing. For example, an association rule such as *(A & B* $\Rightarrow$ *C)* can be registered as a sequence <<*A,B*>,*C*>. Over time the system will be more sensitive against complex events, patterns or rules that it has seen before (similar to reinforcement learning), hence the name ReCEPtor was given to it.

3.3 Association Rule Mining (ARM) Over Streams

This section describes implementations of popular ARM algorithms, Apriori and FP-Growth, over data streams and their use for real-time rule generation. Assume there are items or itemsets $\boldsymbol{I} = \{i_1,\ i_2\ \ldots i_m\}$, in a database $\boldsymbol{D}$. An association rule is

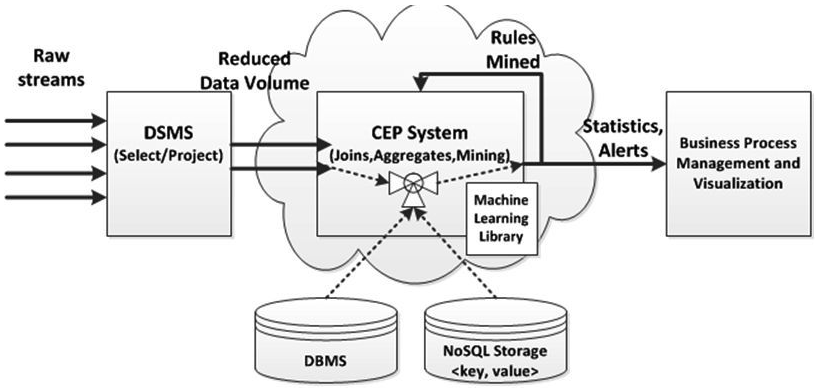


Figure 12: Data stream analytics and mining system architecture.

denoted as $\boldsymbol{X} \Rightarrow \boldsymbol{Y}\ (S, C)$, where X & Y are two disjoint frequent itemsets (i.e. X $\subset$ I, Y $\subset$ I and X $\cap$ Y = $\emptyset$). The total *support* **S** and *confidence* **C** values of the itemsets are calculated using Equ. (1) and Equ. (2). Support of an itemset is the percentage of records in a database that contain that itemset (either X, Y, or both). *Confidence* of the above rule is calculated as the percentage of records that contain X that also contain Y.

$$Support(X \Rightarrow Y) = \frac{s(X \cup Y)}{\mid D \mid} \tag{1}$$

$$Confidence(X \Rightarrow Y) = \frac{s(X \cup Y)}{s(X)} \tag{2}$$

In Apriori [23], all items in the database are transformed into transactions with as transaction id (TID) and their support values are calculated using Equ. (1). Next, items with ($support < minSupport$) threshold values are eliminated. The remaining items are used to construct itemsets (i.e. all subsets) with $\langle 2, 3, ..., n \rangle$ tuples combinatorically. The confidence values for each itemset is calculated using Equ. (2) and itemsets with ($confidence > minConfidence$) threshold are saved as current rules. Apriori algorithm - although it is among the first invented ARM algorithm and the most famous one- is computationally costly because of its combinatorial approach. If the transaction count in the database is $\boldsymbol{N}$, unique item count is $\boldsymbol{k}$, the total itemset

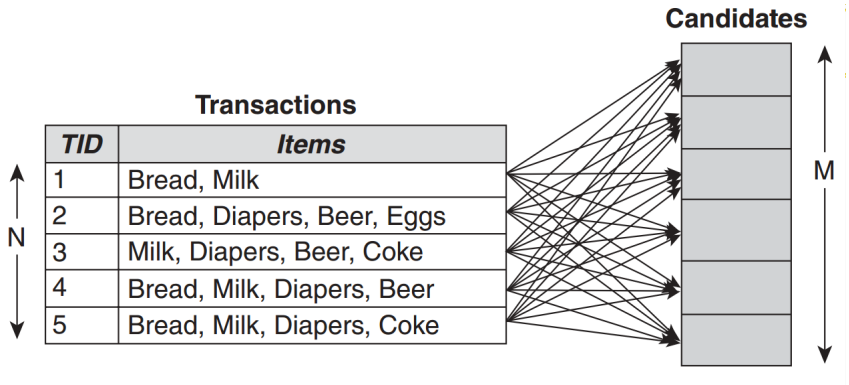


Figure 13: Construction of the candidates from transactions [6].

count is $\boldsymbol{M}=2^k$ *-1*, and maximum transaction length is w, then the time complexity of Apriori is $\mathbf{O}\ (\boldsymbol{N.M.w})$ [6], which depends on the unique item count, *k*, exponentially.

FP-Growth [24] is another popular algorithm designed to address the performance problems of Apriori. FP-Growth also uses the *minSupport* and *minConfidence* threshold values, but it builds the candidate set differently from Apriori. It first orders items in the database based on frequency, stores them in a sorted *FrequencyList* or *Flist*, and prunes the list using the *minSupport* threshold. It creates a Frequent-Pattern tree or *FP-tree* data structure after pruning. The rules are extracted from the FP-tree using the minConfidence value.

3.3.1 Methodology & Experiment Setup

We obtained the Java implementations of the Apriori and FP-Growth algorithms from the Association rule library of the famous machine learning tool called Weka [37] and integrated these algorithms in Esper engine which is also Java based. Later we integrated a second, more-efficient version of FP-Growth algorithm called SPMF (Sequential Pattern Mining Framework) by Phillip Fournier-Viger into ReCEPtor. One needs to implement a custom aggregation function in Esper (AggreationSupport class) to add new operators. We implemented this interface for each algorithm. The EPL continuous query for Apriori looks like this:

```
SELECT Apriori(parameters, table.feature1, table.feature2)
FROM event.win:length(5)
AS table
```

The parameters we used to test Wekas Apriori algorithm inside Esper were as follows: '**-N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1**', where N is the number of rules to output for each window, T is the metric type by which to rank rules (0=confidence — 1=lift — 2=leverage — 3=Conviction), C is minimum metric score (e.g., min confidence = 0.9) of a rule, U/M are the upper/lower bounds for minimum support (defaults = 1.0 and = 0.1), D is the delta by which the minimum support is decreased in each iteration (default = 0.05), S is the significance level, and c is the class index (default = last). In dynamic streaming environments fixed manual settings could lead to either too many or too few rules to be extracted. Therefore, we initially selected to leave this U/M bounds adjustment to be dynamically made by the Weka system. Table.feature1 is the transaction schema and Table.feature2 is the item schema. The EPL continuous query for FP-Growth similarly looks like this:

```
SELECT FPgrowth(parameter, table.feature1, table.feature2)
FROM event.win:length(5)
AS table
```

The parameters we used to initialize Wekas FP-Growth algorithm inside Esper were as follows: **'-P 2 -I -5 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.7'**, where N, T, C, D, U/M are the same as Apriori; P is the attribute index for binary attributes in normal dense instances (default index 2 for sparse instances is used), I is the maximum number of items to include in large items sets (and rules) (default = -1, i.e. no limit.).

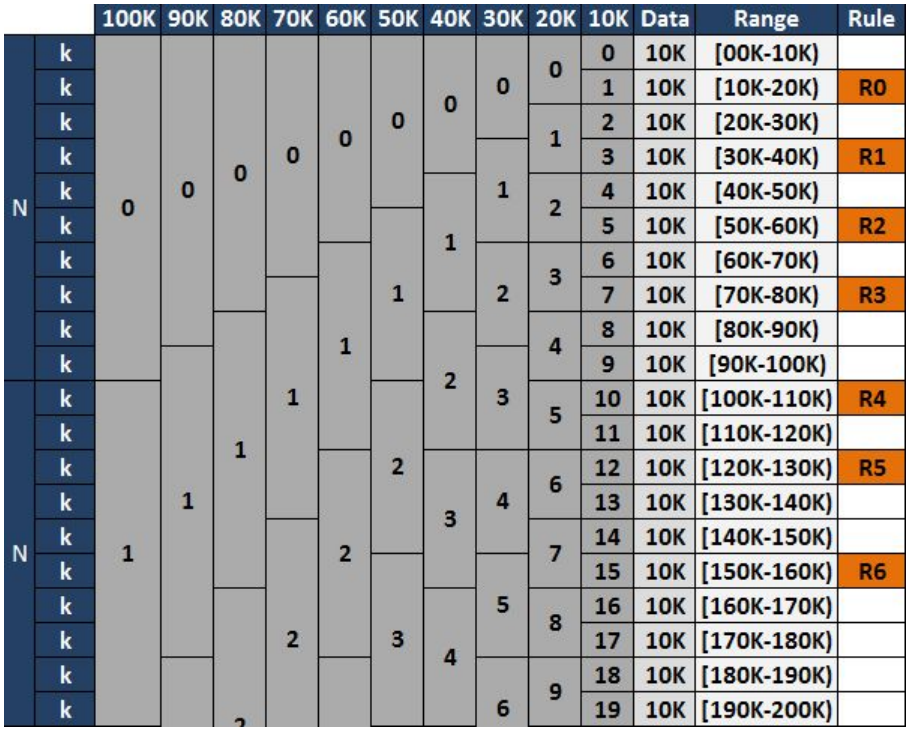
3.3.2 Transactions and Sessions

Most ARM studies use examples from the retail shopping domain: "a customer who buys X and Y also buys Z". In this domain a transaction can be clearly defined by a shopping receipt. How about unbounded sessions such as online music streaming? How can we define a transaction?

A time-based "transaction"or session in LastFM listening dataset is defined as the list of songs listened by each user during the window of analysis. This view of session or transaction is analogous to the one-time visit of a customer when he/she goes to a retail store and buys a set of items. Just like the same customer may visit the retail store later to buy another set of items, the LastFM music user is assumed to listen to another set of songs if he/she has a batch of records in other windows.

Figure 14 illustrates the relations of tumbling windows of 10K (*N=10,000*) through 100K row count sizes with each other. A rule mining analysis can be done either on 10K tumbling windows, or 20K, etc. to represent a session. We actually tried even smaller windows sizes (only 100 or 1000) and upto the complete workload size (to simulate offline processing). We found that windows that are too small do not generate any rules and windows that are too big will take too long to process similar to an offline data mining operations. Before a window slides or tumbles to the nest epoch, rules that are found in the current window have to be published immediately. We plotted the transaction counts for different window sizes in the results section. Figure14 also shows where the extracted sample rules first appeared: for example R0 first appeared in [10K-20K] interval, R1 in [30k-40k] and so on. The goal should be to publish these rules as early as possible, so that the required business actions can be taken immediately.

The minSupport and minConfidence values can also affect the rule count results for different windows sizes. We fixed them after the first comparison of Apriori and Wekas FP-Growth (more about these in Section 5).



		100K	90K	80K	70K	60K	50K	40K	30K	20K	10K	Data	Range	Rule
N	k	0	0	0	0	0	0	0	0	0	0	10K	[00K-10K)	
	k										1	10K	[10K-20K)	R0
	k									1	2	10K	[20K-30K)	
	k								1		3	10K	[30K-40K)	R1
	k							1		2	4	10K	[40K-50K)	
	k						1				5	10K	[50K-60K)	R2
	k					1			2	3	6	10K	[60K-70K)	
	k				1			2			7	10K	[70K-80K)	R3
	k			1			2		3	4	8	10K	[80K-90K)	
	k		1			2		3			9	10K	[90K-100K)	
N	k	1			2		3		4	5	10	10K	[100K-110K)	R4
	k			2				4			11	10K	[110K-120K)	
	k		2			3	4		5	6	12	10K	[120K-130K)	R5
	k				3			5			13	10K	[130K-140K)	
	k			3		4	5		6	7	14	10K	[140K-150K)	
	k		3		4			6			15	10K	[150K-160K)	R6
	k			4		5	6		7	8	16	10K	[160K-170K)	
	k		4		5			7			17	10K	[170K-180K)	
	k			5		6	7		8	9	18	10K	[180K-190K)	
	k		5		6			8			19	10K	[190K-200K)	

Figure 14: Sessionization using tumbling windows: window sizes, their item ranges and the rules founds (R0-R6)..

3.3.3 LastFM Dataset and Preprocessing

LastFM data contains information about 1000 people (lastfm-1K dataset) [38] listening to songs in the LastFM databases. There are approximately 75,000 unique artists, a few hundred thousand unique songs, and millions of transactions in this 3GB dataset from 2005-2011. Briefly, the fields include <*user-id, timestamp, artist-mbid, artist-name, song-mbid, song-title*>. In the preprocessing phase, we first cleaned the records with missing artist information and removed the time-song fields which did not contribute to the rule extraction. This process was done offline and our ongoing work includes doing preprocessing online as well. We used count-based sliding windows. Finally, we had two features of the dataset (<*user-id, artist-mbid*>). Since the Apriori algorithm uses high amounts of memory, just for that algorithm, we further trimmed the data to include users that listened to more than 100 songs and songs that have been listened more than 3000 times overall. This resulted in 967 unique

users listening to 1105 unique artists. This work only reports the performance studies from 5.7 million rows belonging to year 2008 data, but a wider range study including 18 million rows from 2007-2008-2009 were also done (similar results were obtained and thus not reported).

3.3.4 Performance Results

The questions we tried to answer in our experiments are as follows:

1. Can we find transient rules by online rule mining that do not appear in offline rule mining?

2. Is online rule mining more process-efficient and does it generate the results faster?

3. Are the rule results generated by two strategies same, similar or different?

4. Can we learn rules (whether local or globally valid) earlier and publish them with reduced latency?

The analysis results in this section answer these questions. The results given in Table I show that Wekas FP-Growth algorithm integrated into Esper (for finding Top10 rules in 5.7M row count) runs 75x-613x faster than the Wekas Apriori algorithm over the same data stream (61/0.81 and 226/0.37). This is in line with most previous work [26], since FP-growth avoids the iterative, combinatorial candidate generations calculated by Apriori. Due to its poor performance, we later excluded Apriori from the online ARM comparisons. The results for other analysis and comparisons with Apriori are given in our previous related work [19], therefore we skip details here.

After the elimination of Apriori due to unsatisfactory performance in stream processing, we fixed and used *minSupport=0.01*, *minConfidence=0.9* values. As described above these parameters can be changed dynamically by the system when no

Table 1: Results Of Offline Analysis (i: instances, a: attributes)

Algorithm	**[i:967 a:1105]**	**[i:1105 a:967]**
Apriori	61.403s	226.502s
FPGrowth	0.811s	0.369s

rules our found, but we fixed them to be able to compare the performances and published rules of two similar FP-Growth (Weka and SPMF) algorithms. As the window lengths increase, both the number of items (M) and the transaction counts (N, w) per window increase. Figure 14 shows the transaction sizes grouped per user. We can clearly see that longer windows will have more transactions (e.g., 100K $\rightarrow$ 550) and shorter windows less transactions (e.g., 10K $\rightarrow$ 300). Since the support and the confidence values use transaction counts, it will be harder items to find support in larger windows. Therefore, many local rules that find support in their respective windows do not appear when a global analysis is conducted. The minThreshold values can be decreased as window sizes increase, but in that case the computation times will also increase. Figure 15 displays the total number of rules (unique vs. non-unique) found by different tumbling window sizes at the end of the 5.7 million rows. The numbers decrease as windows sizes increase as the minSupport value was kept constant. 10K windows generate 4736 rules (3894 unique) whereas 100K windows generate 570 rules (527 unique). We can also see that the number of non-unique and unique rules found respective windows to not different much, which is a strong indicator for the locality of rules.

$$Jaccard(A, B) = \frac{s(A \cap B)}{s(A \cup B)} \tag{3}$$

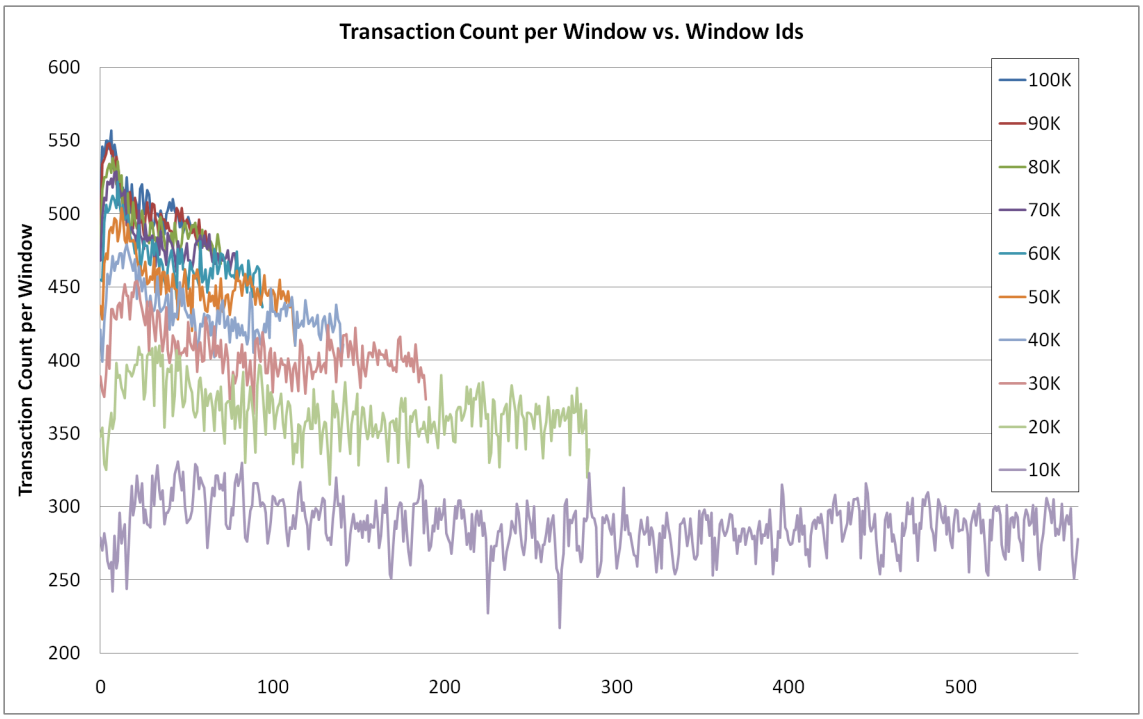


Figure 15: Transaction counts for different tumbling window sizes (Y-axis) given per window id (X-axis).

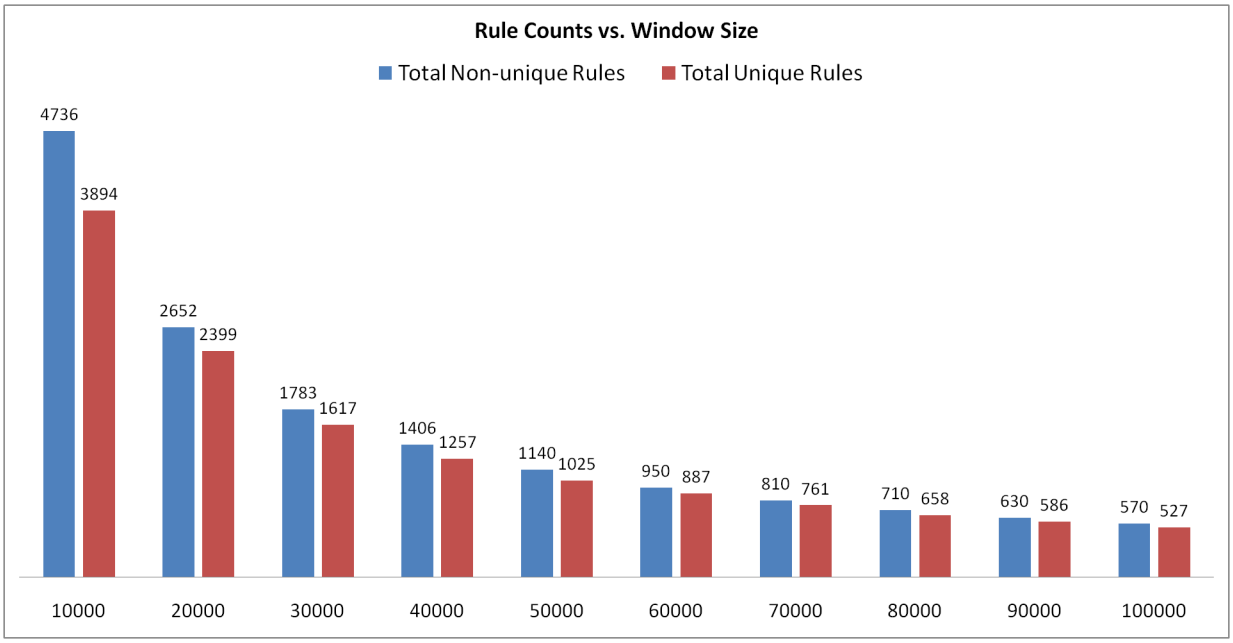


Figure 16: This Figure shows the total non-unique and unique rule counts found by the analyses that use different tumbling window sizes.

	10K	20K	30K	40K	50K	60K	70K	80K	90K	100K
10K	1,00	0,11	0,07	0,05	0,04	0,04	0,03	0,03	0,02	0,02
20K	0,11	1,00	0,13	0,10	0,07	0,06	0,04	0,03	0,02	0,02
30K	0,07	0,13	1,00	0,13	0,09	0,08	0,06	0,04	0,04	0,03
40K	0,05	0,10	0,13	1,00	0,14	0,10	0,07	0,05	0,04	0,04
50K	0,04	0,07	0,09	0,14	1,00	0,16	0,12	0,09	0,07	0,07
60K	0,04	0,06	0,08	0,10	0,16	1,00	0,16	0,11	0,09	0,06
70K	0,03	0,04	0,06	0,07	0,12	0,16	1,00	0,16	0,12	0,09
80K	0,03	0,03	0,04	0,05	0,09	0,11	0,16	1,00	0,16	0,11
90K	0,02	0,02	0,04	0,04	0,07	0,09	0,12	0,16	1,00	0,14
100K	0,02	0,02	0,03	0,04	0,07	0,06	0,09	0,11	0,14	1,00

Figure 17: Jaccard similarity of rules found by different window sizes.

$$
\begin{aligned}
Precision(A,B) &= \frac{s(A \cap B)}{s(A)} \\
Recall(A,B) &= \frac{s(A \cap B)}{s(B)} \\
T(A,B) &= \frac{1}{Precision(A,B)} + \frac{1}{Recall(A,B)} \\
F1Score(A,B) &= \frac{2}{T(A,B)}
\end{aligned} \tag{4}
$$

To understand how the rules found in Figure 15 by different tumbling window size compare to other window sizes we conducted detailed set-based ($\cup$,$\cap$,-,$\subseteq$,$\subset$,$\supset$) similarity and difference analysis. For similarity measure we used both the Eq. 3 and Eq. 4, but we report only Jaccard in Figure 17 as the results were quite similar. The matrix in Figure 17 is symmetric as expected with a band of 1s in the diagonals as it is the same set. Note that the Jaccard value drops sharply after this to 0.15 range for comparisons of windows with 10K size difference down to 0.02 for 90K size difference. This finding suggests that the unique rules found by the 10K window analysis in Figure 16 are 89% different (only 11% similar) to the rules found by the 20K window analysis. We do not show that matrices for set differences and set intersections for brevity.

We found that only two popular rules were common among the analysis using different window sizes. These rules were (Rule1) **George Harrison** $\Rightarrow$ **The Beatles** (i.e. people who listen to George Harrison also listen to the beatles) and (Rule2)

	Rule-1		Rule-2		Rule-3		Rule-4	
Window	Frequency	Window Id	Frequency	Window Id	Frequency	Window Id	Frequency	Window Id
10K	4	{274,371,499,522}	4	{50,87,337,555}	0	-	0	-
20K	1	{54}	1	{25}	0	-	0	{223}
30K	3	{91,97,123}	1	{29}	1	{55}	0	-
40K	3	{68,101,132}	2	{2,84}	1	{29}	1	{111}
50K	4	{54,58,81,106}	3	{18,21,67}	2	{33,39}	1	{22}
60K	5	{45,67,78,87,88}	1	{15}	0	-	1	{74}
70K	1	{53}	2	{1,15}	1	{28}	0	-
80K	4	{50,60,64,65}	1	{11}	1	{4}	0	-
90K	2	{41,57}	2	{10,12}	0	-	0	-
100K	4	{21,37,40,51}	1	{9}	1	{3}	1	{4}

Figure 18: Rules found by analysis in all window sizes (1-2) and other popular rules (3-4) that did not appear just as frequently as 1-2.

Hilary Duff ⇒ Britney Spears. George Harrison, is the guitarist of The Beatles who continued to sing similar songs and apparently continued to get liked by the Beatles fans. Similarly Hilary Duff and Britney Spears have similar singing styles and do have a similar fan base. Figure 18 shows which rule was found in which window. Rule 1 & 2 appear frequently in different window ids of analysis with all window sizes. Rule 3 and Rule 4 are also popular, but do appear in fewer many window sizes and ids. These rules are **"Iron & Wine and Coldplay ⇒ Radiohead"** (Rule3) and **"Blackfield ⇒ Placebo"** (Rule4). The important thing to note is that we can pinpoint rules as they emerge and publish them without any delay.

Note that different ARM algorithms would find the same set of rules, therefore the results of the analyses in Figures 15 → 18 would be the same. But how do their performances compare. We have seen before that Apriori was inapplicable due to its exponential time complexity. Therefore, we implemented and integrated a second, more-efficient version of the FP-Growth algorithm called SPMF (Sequential Pattern Mining Framework) into ReCEPtor and compared it with the FP-Growth implementation in Weka. The results are given in Figures 19, 20 All results were obtained on an IBM server with Intel Xeon processors, but only a single core was used. We hope to report performance results of parallel execution with Espers HA (high availability) mode in the future. In Figure 10, we can see that SPMF is 6-7x

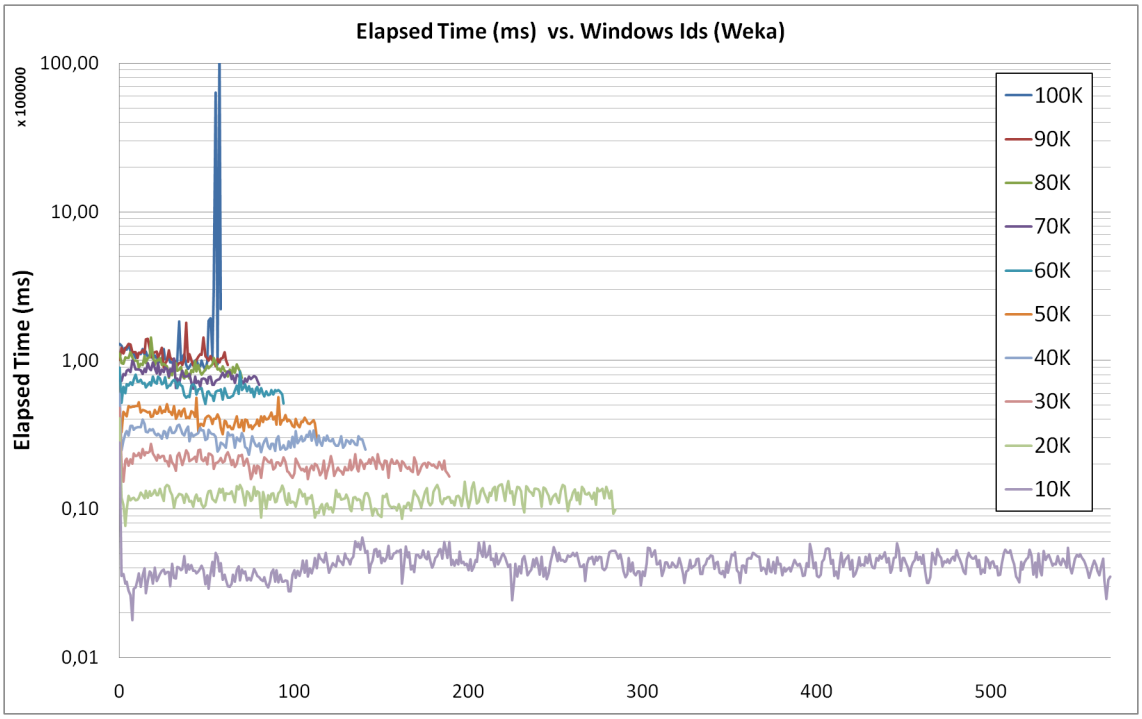


Figure 19: Per window executions times of the FP-Growth implementation in Weka

faster than Wekas FP-Growth (e.g., compare 10K window size), but the performance results of Wekas implementation are more stable. SPMF can vary widely, especially as window sizes increase to 100K.

Figure 21 shows the total time for different windows sizes and the results are quite critical. First of all, it shows that the offline rule mining operation that becomes impossible with thousands of users and millions (5.7M) of rows becomes possible and feasible with an online analysis as we have done in ReCEPtor. Note that a years worth of LastFM data was mined in a few minutes. We see that smaller window sizes can be processed much faster as window sizes decrease and yet lots of unique rules will still be found as we have demonstrated above. SPMF is 7x faster than Wekas FP-Growth for the 10K windows, but the difference reduces to 2x for 90K windows and almost equals for the 100K. The fastest analysis in Figure 21 shows us that it is possible to process 5.7 Million rows in 6.42 minutes in a single core giving us an immense rate of 887K rows/min per core or 15K rows/sec per core. Note that this is not a simple stream

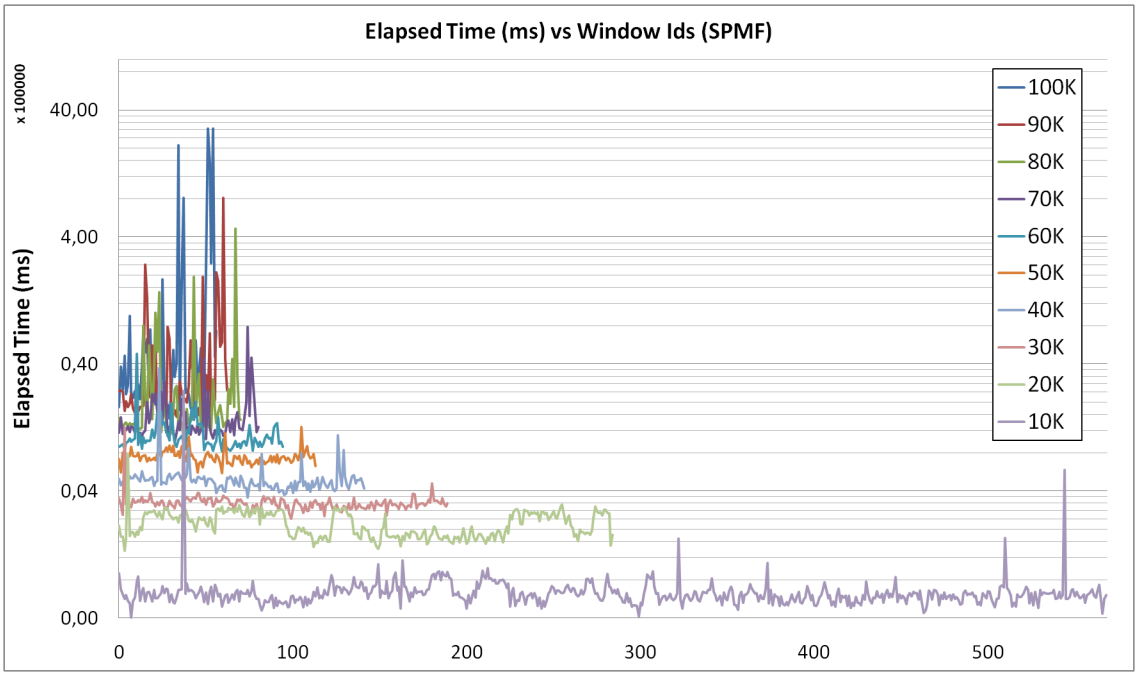


Figure 20: Per window executions times of the FP-Growth implementation in SPMF

filtering operation, but a complex algorithmic calculation. This is the rate at which we can sustain ARM over fast streams. Many streaming applications do not generate data this fast. Yet, as the support and confidence calculations are embarrassingly parallel we can easily sustain much higher rates with multi-core servers. We estimate that we can potentially mine rules over a fast stream of 600K rows/sec using a 40 core IBM symmetric multi-processor (SMP) server, but the claim bears a proof which we leave as future work. However, our performance findings here render the strong assumptions about the need for "one-pass stream mining"algorithms over fast data questionable (whether they are really needed or not).

Figure 22 show the memory usage of the SPMF FP-Growth rule mining algorithm, which was more efficient than Wekas both in time and space. First, even the largest memory used (3.17GB) is easily obtained on any commodity server even laptops-today. All the stream processing is done in-memory therefore this analysis was required and useful. Larger window sizes require more memory for holding the data and creating the candidate sets as can be expected. The increased memory usage

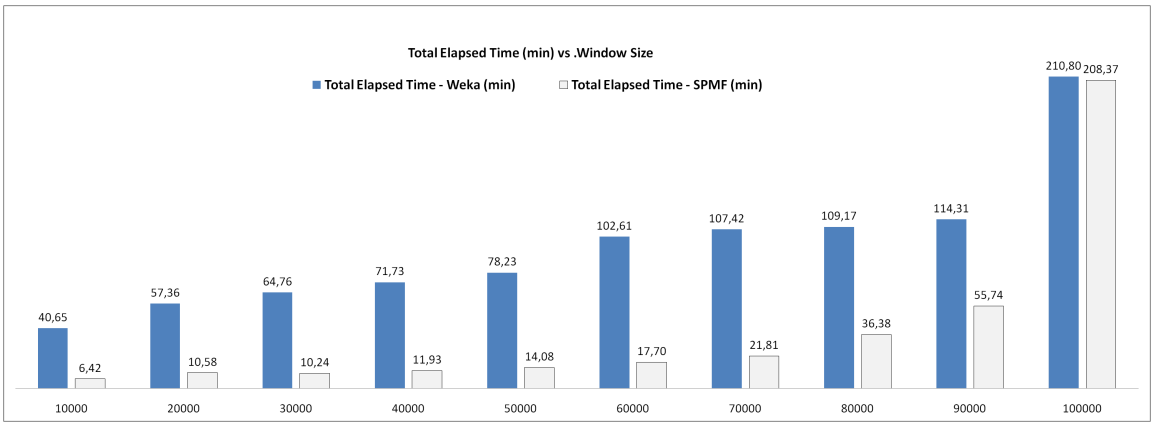


Figure 21: Total processing times of two FP-Growth algorithms (Wekas and SPMF) for different window sizes

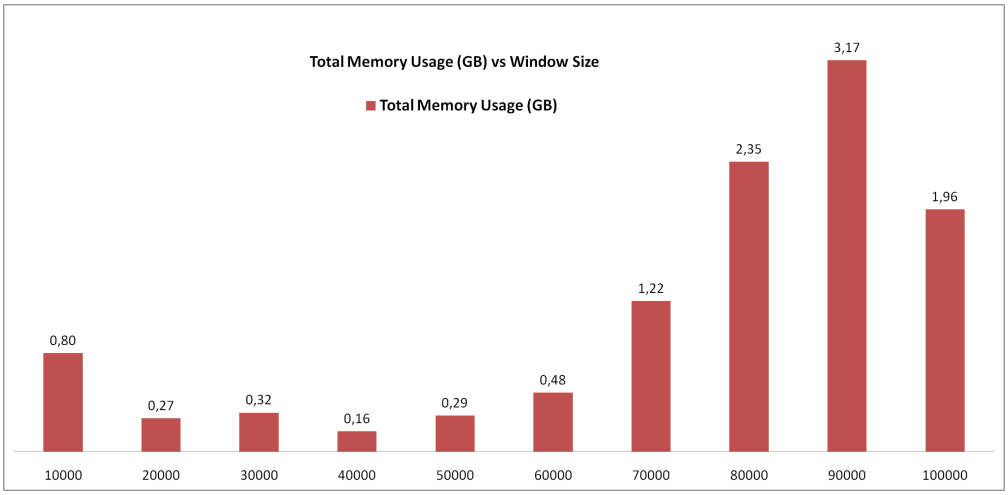


Figure 22: Memory usage of SPMF FP-Growth algorithm for different tumbling window sizes

towards smallest window sizes are due to the increased number of rules generated as shown in Figure 16

During our research we discussed the use of incremental updates among windows and even briefly tested the outcomes. However, the disadvantages were two-fold:

1. We were keeping more state around, thus increasing the memory usage and
2. The additional states kept turned a real-time analytics approach into the imitation of an offline rule mining algorithm, which we did not desire.

CHAPTER IV

COMBINATORIAL TESTING TECHNIQUES FOR CEP ENGINES

Testing practice has a crucial role during the design, implementation and integration of software, hardware and complex systems composed of them. The cost of failures, caused by bugs that could not be detected and fixed early in the process increase in a multiplicative way, and adversely affect the overall projects costs. However, trying to do comprehensive tests generating correct outputs is also costly both time-wise and money-wise. Combinatorial Testing Techniques (CTT) have been a preferred method in software testing due to their quantifiable case coverage guarantees and appropriateness for automation. We observed that, Complex Event Processing (CEP) engines - commonly used today for real-time analysis over critical, high-volume signal processing applications (e.g., mobile communication, sensors, radar) - are NOT being systematically tested with approaches such as CTT. In this chapter, we uniquely show applicability of CTT to CEP for fast creation of continuous query test suites and present promising results.

4.1 Introduction to Software Testing

In order to design reliable systems testing is performed to reduce faults from designing phase until releasing the system to end-user. According to NIST Report, published by NIST in USA [7], many faults that are detected at middle and last phases, such as integration and testing of software projects, as shown in Figure 23. Since fixing faults are the more expensive and costly at the end of last development phase, the faults should be detected in middle-forward of development phase. 40% of development

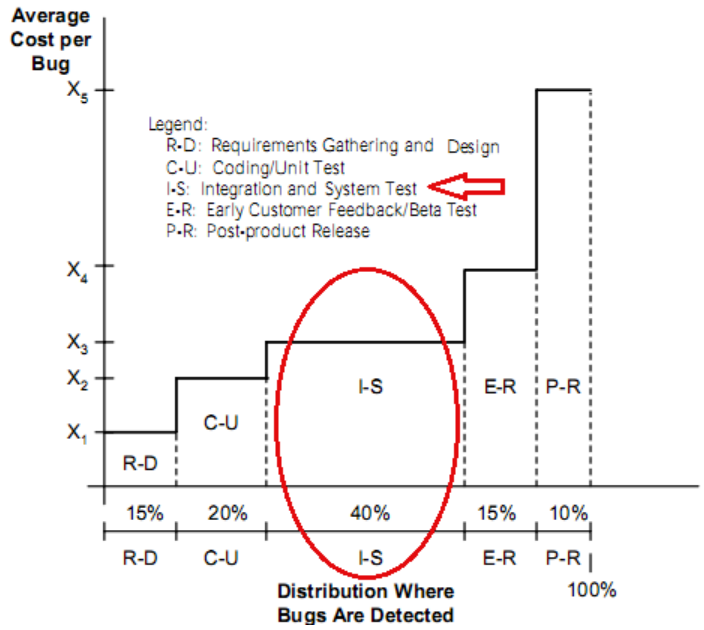


Figure 23: Averages and costs were depicted that the faults of which could be detected and fixed occurred in software phases [7]

phases (I-R) are composed of system integration and testing phases as denoted in Figure 23. According to the highest average in Figure 23, the faults that are detected in last phases can adversely affect the entire total of software costs due to a long testing phase. Thus, comprehensively convenient testing has to be done and high performance systems.

4.2 Motivations for Testing CEP Systems

Software behavior depends on parameters, such as system inputs, configuration settings, state variables and many other factors, which can affect performance testing of a complex system. If every parameter and all possible values for each parameter is considered for the variable space, then testing all combinations will be impossible.

Real-time "mission-critical" systems and applications especially have to be effectively tested within short time. The testing has to be automatically performed on the system. However, the comprehensively testing on CEP engines and DSMS, could not be observed in prior literature. Therefore, we focus on the implementation and

application of automatic test techniques on CEP that will be explained in next sections. In order to present new solutions (e.g., suitable and measurable for test case coverage in testing phase) to test CEP engine, CTT had been chosen and investigated as detailed in Section 4.3. Instead of generating all combinations of test cases, CTT can be preferred to considerably reduce number of test cases and duration of testing. Since CTT are used to generate subsets accordingly given input variables, defined on systems and using various CTT, quality of test coverage and its results can be qualified. The results are laid out section 4.6.

Goals of our studies and motivations are below;

- To generate fast test cases/suites on CEP systems.
- To detect anomalies on different environments and systems for CEP engine by using various CTT. In this way, to perform performance testing, and
- As future work, completed automatic testing framework for CEP system will be developed as a smart system that can recommend convenient Event Processing Language (EPL) queries to users on the basis of performance results of generated and executed test cases.

4.3 Combinatorial Test Techniques (CTT)

As CEP/DBMS engines were described in the introduction of the thesis. In this section, specific test strategies, such as Combinatorial Testing Techniques (CTT) are applied to queries to test their performance.

In AGENDA (A test GENerator for Database Applications) study [39],[40], error logs and records in RDBMS are interpreted by specific Lexical Analyzer. Test cases are generated to reduce repeatedly occurring errors according to interpretations,and then new and best fitting queries are suggested to the user. In QAGen (Query-Aware Test Database Generation) test automation [41], inputs of database schema

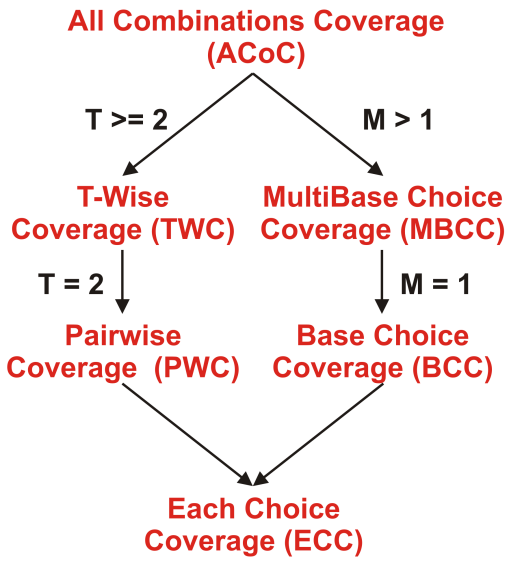


Figure 24: Subsumption Relation of CTT

are analyzed, and then, testing queries are generated consciously with restricted query properties. Test vectors [42] are created by test drivers that perform tests by using pairwise test techniques in AETG (Automatic Efficient Testcase Generator). In contrast to these test automation tools, this study is performed on Esper [2] CEP/DBMS engines instead of RDBMS.

In order to generate various subsets, the cartesian product is mainly used to create well defined test cases cases as illustrated in Figure 24. All Combination Coverage (ACoC) Technique covers subsets of test cases to generate whole combination of test cases. Pairwise Coverage (PWC) Technique covers just pairwise combinations. PWC is subset of T-Wise Coverage (TWC) Technique, and also, Base Choice Coverage (BCC) Technique is subset of Multibase Choice Coverage (MBCC) Technique. However, Each Choice Coverage (ECC) Technique is separately subsumed by PWC and BCC. All these testing strategies are detailed as below.

In this study, test subsets are generated by using five various techniques[43] according to variable space, such as data fields of streams, and number of queries, etc. in systems.

All Combinations Coverage (ACoC): Every possible combination for all of parameter of values has to be covered by it. For example, if we have three parameters P1 = (A, B), P2 = (1, 2, 3), and P3 = (x, y), then, all combinations coverage requires 12 tests: (A, 1, x), (A, 1, y), (A, 2, x), (A, 2, y), (A, 3, x), (A, 3, y), (B, 1, x), (B, 1, y), (B,2, x), (B, 2, y), (B, 3, x), (B, 3, y)

Base Choice Coverage (BCC): For each parameter, its one of the possible values is designated as a the base choice. For instance, a target parameter is chosen from space. Expect for being chosen parameter, the rest of parameters' combinatorial sets are generated. Finally, in order to apply BCC, the target chosen parameter is inserted into each subset of test cases generated in previous step.

MultiBase Choice Coverage (MBCC): At least one, and one more possibly values of variables, designated for each parameter. MBCC can be defined as generic version of BCC.

Each Choice Coverage (ECC): Each parameter value have to be covered in at least one test case.

Pairwise Coverage (PWC): Given any two parameters, every combination of them has to be covered in at least one test case.

T-wise Coverage (TWC): Given any t parameters, every combination of them/its t parameter(s) has to be covered in at least one test case.

4.4 Applying CTT on CEP

In contrast to generating many test cases for analysis, CTT is used for execution of restricted amount of test cases. In this study, in order to measure performances of EPL, test cases for EPL are automatically generated by CTT. Our goal is to apply performance measurability effectively and faster. In order to implement EPL query

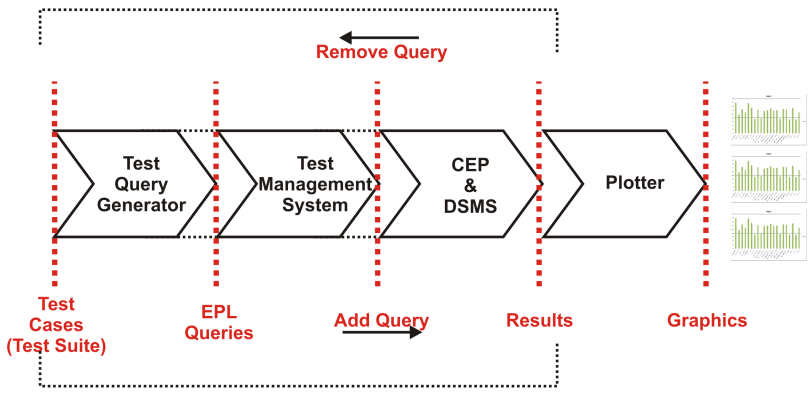


Figure 25: Structure of main system

optimizations, and detect anomalies of usage of query in CEP engine, the performance results are observed . The overall system architecture is illustrated in Figure 25.

The system is composed of three main modules that are integrated in the system.

Test Query Generator: Test cases for being chosen concerning EPL queries in the test suite can be configured by using a shell script. Generated test cases are stored as XML file format.

Test Management System: Test cases concerning EPL queries in configuration file are used to dynamically execute all queries, and observe performance results in EsperTech CEP & DSMS engine [2] .

CEP & DSMS: In order to determine concerning incoming events over streams, and analyze over them, it is used to connect streams on the system. It also tries to make sure that sources such as CPU, memory can be effectively consumed.

The structure of test configuration is designed by a XML file and used by CEP engine. In Figure 26, "users" are denoted by U_k, "queries" are denoted by Q_i, Q_j linked to "users", and schema of "stream" is denoted by S_n that is associated with "data" fields denoted by D_m.

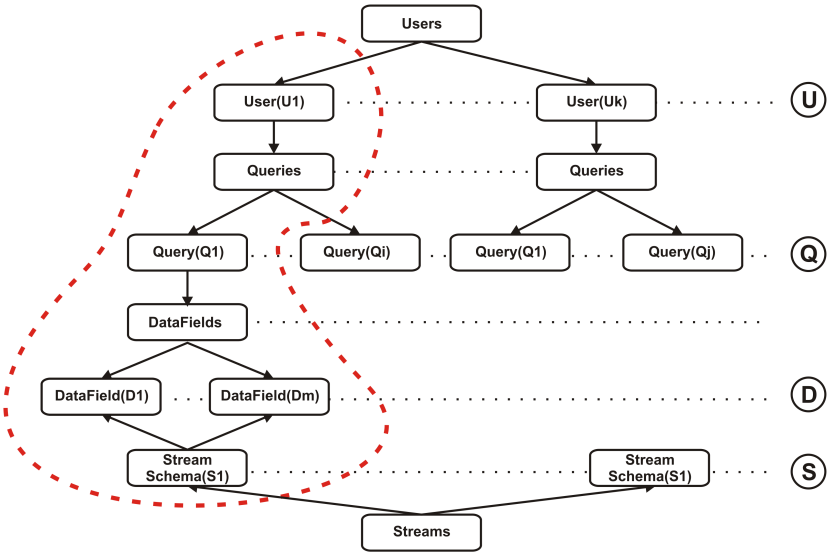


Figure 26: Unique test configuration file.

Since the structure of configuration file is associated with itself as a tree structure, it ensures that many various test case combinations are generated. As depicted in Figure 24, CTT methods, such as ACoC, BCC, MBCC, ECC, PWC, TWC are separately applied on *U, Q, D, S* levels and their branches of tree are illustrated in Figure 26 Likewise, although also each level has combinatorial structure in itself, especially "users" and "queries" are assumed to be uniform in our assumptions will be laid out next sections. Finally, test cases for XML configuration on CEP engine, will be generated by using CTT according to assumptions, such as uniform structures, using just "data fields" in "queries". For these reasons, number of all combinations will be calculated by *s(TT)* denoted in formula 5. *S(D)* can dramatically affect the number of test cases generated because of a parameter in power of 2.

$$s(TT) = s(U) * s(Q) * 2^{s(D)} \tag{5}$$

Since all combinations test cases have a large space, some assumptions are made as below instead of generating of all combinations.

- There are 4 different type of queries abbreviated as SPJA *(Selection, Projection,*

Join and Aggregation) in EPL . If the whole of type of queries insert into test cases generation, number of test combinations will dramatically increase. Thus, test cases are generated and tested by just using projection queries.

- In order to reduce the combination space, red dashed line, as shown in Figure 26, CTT are applied just on "data fields" D_m in query S_1.

- The working space has 1 "user", 1 "query" and 5 "data fields" related with "query" according assumption (s(U) = 1, s(Q) = 1, s(D) = 5)

Therefore, 32 different combinations are generated according to assumptions and formula 5.In other words, it covers over all test cases to test according to our requirements. The reduced combination space is smaller than real system spaces. For instance, lets calculate number of test cases that will be generated . An instance of system has 100 users, 10 queries per user, 10 different data fields per query, and working time let be 5s per continues query. *(s(U)=100, s(Q)=10,s(D)=10, t=5s)*

In order to obtain 100% coverage from generated test cases according to given inputs (variables and their values), 1 024 000 test cases have to be created manually and/or automatically. In that case, this test suite will take a total of 60 days to finish to be executed. Consequently, test framework automation provides some benefits, such as reducing large scaled space by using CTT. In this way, new test approaches are ensued for EPL queries over CEP/DSMS.

$$s(TT) = s(U) * s(Q) * s(D) \tag{6}$$

4.5 Experiment Results and Discussion

In this experiment, data sets are composed of 5 data fields *(<bus id, latitude, longitude, velocity, date>)* and sampled per 20ms from the GPS data on transportation

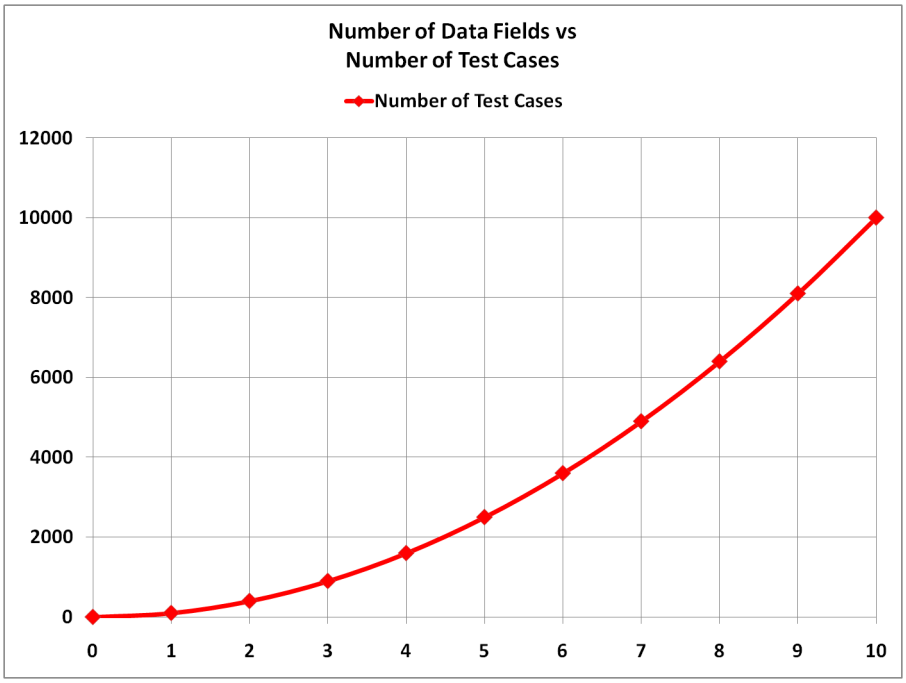


Figure 27: Exponentially increasing of number of test cases.

collected by IETT (Istanbul Municipality Bus Transport Agency)[1].

The generated test cases are tested on a laptop with Intel i5 (4 Cores), 2.67GHz CPU, 3GB RAM and Windows-7 64-bit OS. We have two goals: (1) one of them is to try to monitor main performance results, (2) to detect anomalies over them. In this way, both minimum test cases in one test suite and maximum coverage provide efficient testing for CEP/DSMS and users.

Throughput results of the system, sorted according to used coverage values of CTT, as shown in Figure 28, which are obtained by using various CTT. In contrast to ACoC technique, when comparing both throughput result and average throughput result, rest of techniques, except for ACoC technique has approximately similar results as plotted in Figure 28. According to the throughput results, test cases are tested in same time, as well as generated test cases for ACoC seems and behaves as load testing. Even though ACoC leads to the best coverage. Its throughput result is higher

[1]IETT, http://www.iett.gov.tr/en

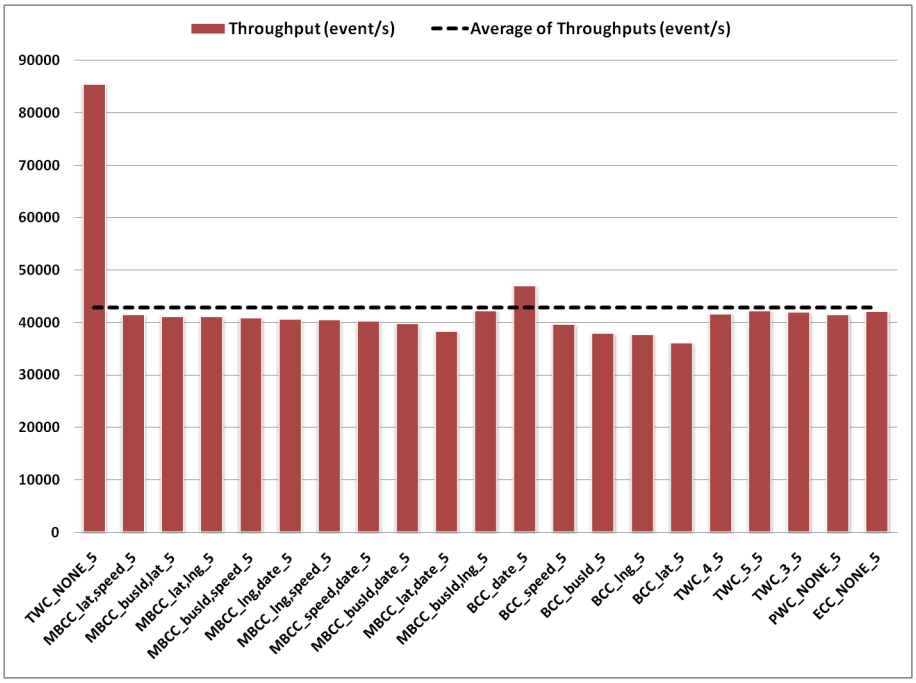


Figure 28: CTT - Throughput (event/s)

than average throughput. Hence, it would like to be preferred. Thus, the rest of the testing strategies can be preferred for systematically and effectively testing.

Figure 29 shows that latency results are varying according to chosen CTT. By contrast, coverage vs. variable space for each generated test case, as shown in Figure 30. Figure 28, Figure 29 show throughput and latency results that are compared for each CTT, meanwhile, their test coverage is shown in Figure 30. According to results as an ideal test technique, ECC can be figured out from them. However, representing fundamentals of instance of system by using small test cases that can not be applicable. In other words, it can be can be potentially dangerous for system testing.

ACoC has the highest coverage, and also, TWC (for T=5) has the lowest coverage that are observed from performance results. In other words, "data field" have 5 "fields" that are compatible with TWC (T=5). During our experiment, the extremely lowest throughput or/and highest latencies would be able to be anomalies that was not observed. Therefore, according to performance results, EsperTech CEP [2] can

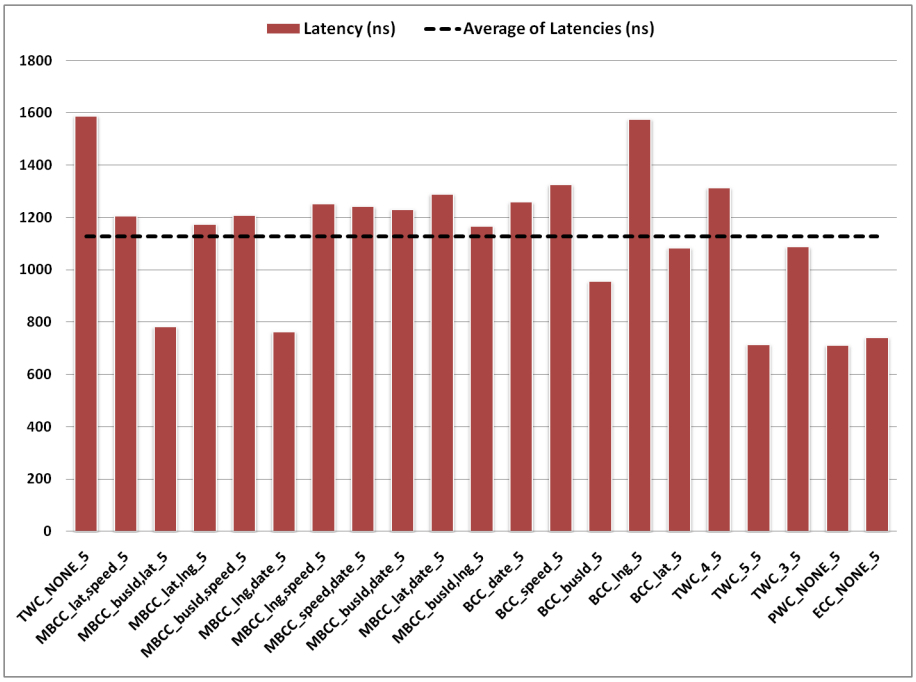


Figure 29: CTT - Latency (ns)

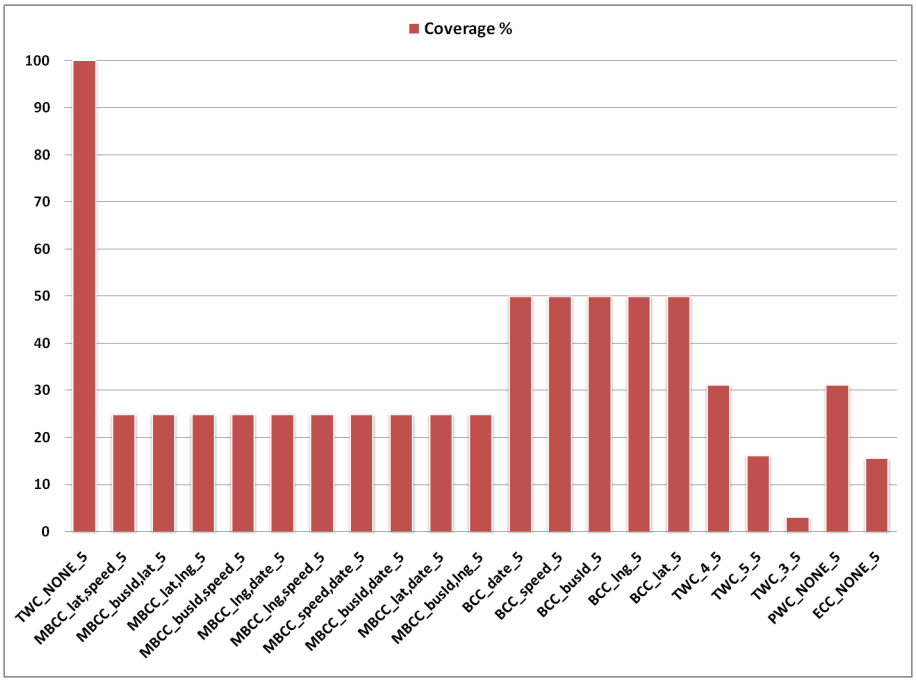


Figure 30: CTT - Latency (%) (ns)

be declared as “reliable”.

4.6 Conclusion and Future Work

The number of test cases that can be generated using the “data field” (D_m) exponentially grows. In order to reduce this number, CTT are can be applied according to the results in this study. In this way, in order to test approximately whole properties of system, test cases can be generated to cope with the challenge regarding the testing huge of variable space. Benefits of using CTT are to reduce huge variable space and to obtain restricted meaningful performance and coverage result. Application of a test framework automation with using CTT that is unique study on CEP/DSMS. Since the automation provides some benefits, such as generating continues EPL queries, fast and easily interpretation for performance results, it can be preferred to generate test cases in CEP/DBMS. Importance of reliability and testability of CEP/DBMS were emphasized by this study. In future work, in order to implement smart query recommendation to suggest best queries to system users, CTT’s query generator ability will be combined with coverage and performance results. It will be a cloud service, and then, maybe it can be presented to users via web.

CHAPTER V

HARDWARE-ACCELERATED DATA STREAM PROCESSING

Some operations such as querying and reporting, can take a long time in DWH and DM. Some specific queries (e.g., complex SQL queries, etc.) regarding BI operations cause tremendously large number of I/O requests on traditional DBMS in EDW. In order to improve performances (e.g., analyze data and get fast respond within real-time) of systems, various approaches are developed to reduce operational and analytical delays. IBM Netezza[1], hardware-software appliance was developed for this purpose. The IBM appliance is mainly constructed on a group of FPGA[2], and it includes specific software, such as IBM InfoSphere BigInsights[3], IBM InfoSphere Streams[4] integrated on FPGA to accelerate system. In contrast to FPGA, Graphics Processing Unit (GPU) cards are used as hardware-accelerators to cope with the same problems instead of using CPU technologies. Likewise, multi-core Network Processor Card (NPU) produced by Cavium[5] can be used. It has MIPS64 OCTEON architecture that can concurrently process network packets, Link Layer to Application Layer in Open Systems Interconnection (OSI) model. NPU is used in network infrastructures, such as Load Balancing, to overcome Distributed Denial Service of Attack (DDoS Attack), Deep Packet Inspection (DPI), Packet Sniffing (PS), etc.. In this study, simple DPI application, integrated string searching algorithm is performed

[1]IBM Netezza Data Warehouse Appliances, http://www-01.ibm.com/software/data/netezza/

[2]Field-programmable gate array, http://en.wikipedia.org/wiki/Field-programmable_gate_array

[3]IBM InfoSphere BigInsights, http://www-01.ibm.com/software/data/infosphere/biginsights/

[4]IBM InfoSphere Streams, http://www-01.ibm.com/software/data/infosphere/stream-computing/

[5]Cavium, http://www.cavium.com/

on Cavium NPU card. DPI application is chosen as a research area for this study, since it is more applicable to use in stream computing.

5.1 Challenges

Until releasing of the project, we have spent a lot of time and physical effort to launch the NPU card.

- To compile Linux kernel on Octeon/MIPS (Fixed wrong/broken Makefiles and file paths, mounted a bigger USB disk in order to fact that Cavium's compact disk had enough space to build OS. To Boot build OS using our USB flash disk)
- To obtain new laptops as client user & Gateways/DNS settings from the university and permissions to run Cavium Card on real IT environments.
- To develop sniffer program work as a real DPI application over the Internet.
- To modify and fix broken DPI application that was running as Brute-Force for new searching algorithm such as Boyer-Moore

5.2 System Architecture and Hardware Settings

In order to program, MPAC (Multi-Processor and Communication) API developed by Caviums Engineers in C based programming language, is indeed required to perform concurrent network operations on Multi-Cored NPU card. MPAC is used in this study. Cavium NPU card has two different modes, such as Simple Executive Standalone (SE-S) and Simple Executive User-Mode (SE-UM) to run application over itself. It has no running and built Operation System (OS) over itself, the embedded operating system has to be built to launch applications over it. Although applications require OS to run over it in SE-UM, by contrast, SE-S does not require this. In order to easily launch network drivers on DPI, SE-UM is preferred in this study. In addition to hardware's properties, embedded Linux operating system has Busybox

utility package composed of small executable files to easily manage hardware. "It provides replacements for most of the utilities usually found in GNU fileutils, shellutils, etc."[6]. In order to launch the card, networking commands & tools (bootoctlinux, brctl, ifconfig, mount, dhcpd, etc.) in Busybox have been referred.

DPI can be briefly thought as a simple searching application that run over an instance of an NPU card to find out desired words over the stream data pass-through network layers, such as layer 7 to layer 2 in Open Systems Interconnection (OSI) model. Since data flows from network to other network, through using network infrastructures, its performance plays crucial roles on searching and filtering over streaming data. The challenge can be solved with hardware or software approaches. Our approach is to execute string searching algorithm on DPI. Although Boyer-Moore string searching algorithm is used in this study, in order to compare results with Brute-Force string searching algorithm, two of them are performed on the Cavium NPU card.

In this study, CN57xx NPU Card, MIPS64 architecture as depicted in Figure31 is used. Its properties are as shown in Figure32. It has 8-12 cnMIPS cores, up to 900 MHz on a single chip that can be processed up to 21.6 Billion MIPS64 instructions per second by using 2M L2 cache[7].

Our system architecture is designed as in Figure 33. Two different LANs in the university connect with gateway denoted by GW in Figure 33. It has the TCP/IP communication properties that are given in Table 2

Table 2: Gateway properties

IP	Subnet	Gateway	DNS
10.100.40.100	255.255.255.0	10.100.40.254	10.100.10.101

There are 2 different computers, such as tester and manager laptop in the system

[6]Busybox, http://www.Busybox.net/
[7]Cavium NPU Card,http://caviumnetworks.com/

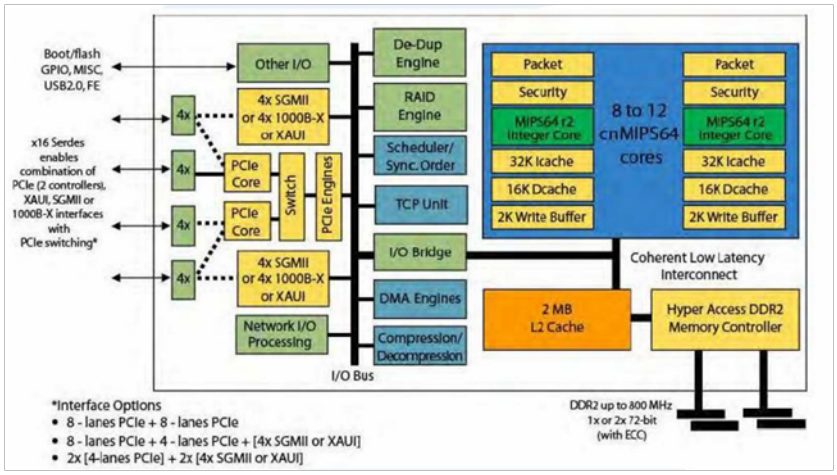


Figure 31: Architecture of Cavium NPU Card (CN57XX)

OCTEON Plus CN57XX Processor Family

Device	cnMIPS Cores	Performance: Max. Available Instructions Per Second	Option: SSP	Option: SP	L2 Cache	Networking Interfaces	PCI-Express	Memory IO w/ECC	Package
CN5740	8	14.4B	Y		2MB	2x [4x SGMII or 1x XAUI]	2x [x4 or x8 Lanes]	DDR2 up to 800 MHz 1x or 2x 72-bit wide	1217 FCBGA
CN5745	10	18.0B	Y		2MB				
CN5750	12	21.6B	Y		2MB				

Device Options:
Device Speed Grade (600 LP = 600 MHz Low Power, 600 = 600 MHz, 750 = 750 MHZ, 800 = 800 MHz, 900 = 900 MHz)
Option code for device family listed below:

SSP = Secure Storage Processor: Includes RAID, encryption, compression/decompression, networking, TCP acceleration and QoS
SP = Storage Processor: Includes RAID, compression, networking, TCP acceleration and QoS

Figure 32: Properties of Cavium NPU Card

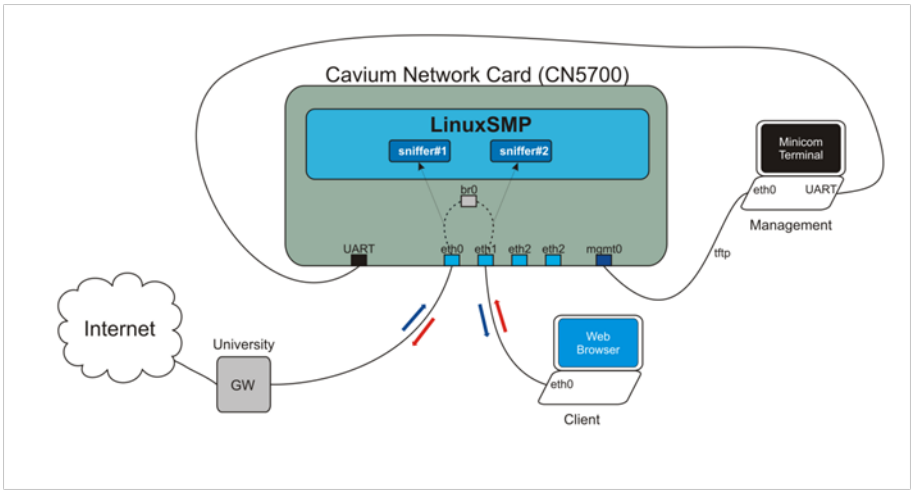


Figure 33: View of Simple Real DPI Settings in University Network

architecture. Setting of the tester machine, web browser client on it is done accordingly TCP/IP communication information above. Tester machine works as a dummy user to test the system. The other one, manager machine, is used to manage (e.g., to load file, to execute program, to launch card) the system, Cavium NPU Card via minicom terminal over *UART-UART* connection is established with NPU. The *eth0-mgmt0* connection is used to establish internet and use TFTP (Trivial FTP) over card by manager machine. These connections set according to the settings, below.

In order to complete TFTP the settings,"/etc/minirc.ttyUSB0" file has to be modified as detailed below.

```
# Machine-generated file - use "minicom -s" to change parameters.
pr port             /dev/ttyUSB0
pu baudrate         115200
pu bits             8
pu parity           N
pu stopbits         1
```

In addition, "/etc/xinetd.d/tftp" file has to be modified as detailed below.

```
#/etc/xinetd.d/tftp
#
#       Erdi Olmezogullari
#       erdi.olmezogullari@ozu.edu.tr

service tftp
{
    socket_type     = dgram
    protocol        = udp
```

```
        wait            = yes
        user            = root
        server          = /usr/sbin/in.tftpd
        server_args     = -s /tftpboot
        disable         = no
        per_source      = 11
        cps             = 100 2
        flags           = IPv4
    }
```

In order to launch Cavium NPU Card, the following commands are prompted on command line.

```
$bash: minicom w ttyUSB0
$bash: tftboot 0 vmlinux.64
$bash: bootoctlinux 0 coremask=0xfff
```

The two different mediums, LANs, have to be connected using a network bridge or router. The virtual network bridge can be provided by Busybox commands. The bridge between two different mediums, such as denoted by *eth0* and *eth1* interfaces on NPU card that is established by using commands of Busybox. Those interfaces are connected by bridge *br0*. The sniffing operation ??? In order to connect different LANs, the following commands are prompted on command line. Thus, setting of each component in system is done.

```
$bash: ifconfig eth0 0.0.0.0
$bash: ifconfig eth1 0.0.0.0
$bash: brctl addbr br0
```

```
$bash: brctl addif br0 eth0

$bash: brctl addif br0 eth1

$bash: brctl br0 up
```

5.3 Implementation of Simple DPI Application

In this study, two kinds of string searching algorithms,such as Brute-Force[8], Boyer-Moore[9] are used to implement a simple DPI application to improve the performance calculation. In order to detail the implementation of simple DPI application, let us consider that the tester machine would like to search something over the Internet via "www.google.com". In that case, HTTP requests, such as searching from the tester machine (eth0) send to eth1, and then, requests illustrated by "red arrows" sends through-pass br0 to eth0 in system architecture as shown in Figure 33. The responses illustrated by "blue arrows" return from www.google.com over gateway denoted by GW. Since NPU card holds place between to different LANs, every request and its responses will respectively pass through $eth0 \rightarrow eth1 \rightarrow br0 \rightarrow eth0 \rightarrow GW \rightarrow Internet$. In a policy-based (or black-listed) Internet access system, each ethernet plug-in, such as eth1 (incoming/request) and eth0 (oncoming/response) on the card can be sniffed by a simple DPI application. Our simple DPI application is used over them. Therefore, HTTP requests may be stoped from going to a destination in case it includes a certain word in the HTTP packets.

The differences of brute-force and Boyer Moore string searching are shown in Figure 34 and 35. Obviously, Boyer Moore is more efficient than brute-force string searching algorithm. In the best case, complexity of Boyer Moore is $O(n/m)$ instead of brute-force, $O(nm)$.

Since preprocessing operation processes concerning string at the beginning of the

[8]Brute-Force, http://www-igm.univ-mlv.fr/ lecroq/string/node3.html#SECTION0030
[9]Boyer-Moore, http://www-igm.univ-mlv.fr/ lecroq/string/node14.html#SECTION00140

- no preprocessing phase;
- constant extra space needed;
- always shifts the window by exactly 1 position to the right;
- comparisons can be done in any order;
- searching phase in $O(m \times n)$ time complexity;
- $2n$ expected text character comparisons.

Figure 34: Feature of Brute-Force Algorithm

- performs the comparisons from right to left;
- preprocessing phase in $O(m + \sigma)$ time and space complexity;
- searching phase in $O(m \times n)$ time complexity;
- $3n$ text character comparisons in the worst case when searching for a non periodic pattern;
- $O(n/m)$ best performance.

Figure 35: Feature of Boyer-Moore Algorithm

searching in Boyer Moore algorithm, it performs better than other searching algorithms, including the brute-force, worst cases algorithm and its details are shown in Figure 35.

Since fast streaming data is coming from concerning network areas, and it can be handled by efficient algorithm such as Boyer Moore. In real-time an application, simple DPI can be performed, by using Boyer the Moore algorithm, on Cavium NPU Card.

5.4 Integration of DPI Application on Card

The Real system architecture and running simple DPI application on it are depicted in Figure 36 and 37.

In Figure 38 and 37, the laptop, on the right side is the client accessing the Internet shown in Figure 31. The other laptop, manager machine, is connected via an UART cable (minicom terminal) to the Cavium NPU card and shows that the card is able to detect the word “Civanlar” inside the HTTP requests/responses with the Brute-Force

Figure 36: View of Simple Real DPI Systems in University Network

Figure 37: View of Close Up of Connections on the Card

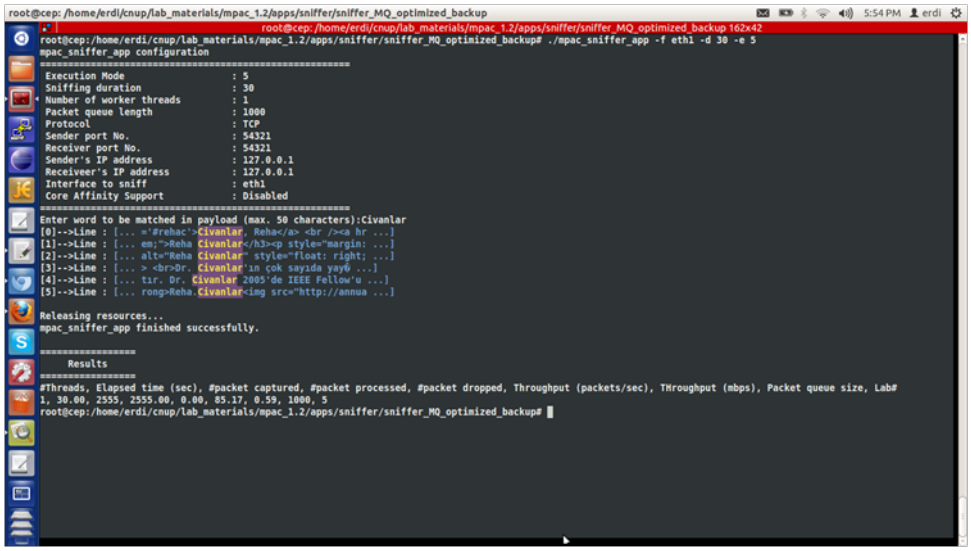


Figure 38: Screenshots of Simple DPI with Brute-Force Searching Algorithm on Development Environment

and Boyer-Moore Algorithm. You can see the screenshots in Figure 38 and Figure 39. The incoming and outgoing packets via HTTP are sniffed with "Civanlar" search key over the DPI application on multi-core network card that is shown using Brute-Force searching algorithm.

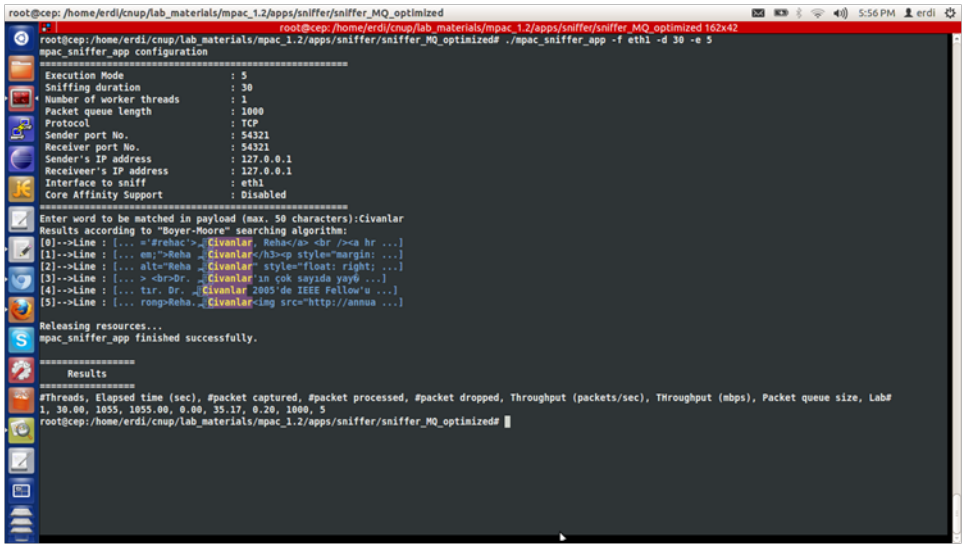


Figure 39: Screenshots of Simple DPI with Boyer-Moore Searching Algorithm on Development Environment

5.5 Conclusion and Futurework

Java Virtual Machines (JVM) is installed on Cavium NPU Card for MIPS64 architecture. Thus, any java-based program, a data processing engine, could be run on our setup. Potential applications include OpenFlow[10] testbed & controller with Cavium and EsperTech Complex Event Processing (CEP). After that, high performance tests can be directly measured on Cavium NPU Cards multi-cores.

[10]What is OpenFlow?,http://www.openflow.org/wp/learnmore/

CHAPTER VI

CONCLUSION

Conducting on-the-fly analytics over big and fast data is the newest IT challenge. In this thesis, we presented implementation details and performance results of *ReCEPtor*, our system for "online" Association Rule Mining (ARM) over big and fast data streams. We made critical comparisons of "online" vs. "offline" data mining. Specifically, we implemented and tested Apriori and two different FP-Growth algorithms (from Weka and SPMF) inside Esper CEP engine and compared their performances using LastFM social music site data at different tumbling window sizes. We found that with online ARM we can generate many unique rules, with a sustained throughput up of 15K rows/sec, and lowest possible latency for rule publications (as they emerge) compared to offline rule mining. We have found many interesting and realistic musical preference rules such as **"George Harrison $\Rightarrow$ Beatles"** and reported some of them here. We also implemented a new testing framework for CEP engines and described an NPU-based hardware acceleration scheme for CEP in this thesis. We hope that our findings can shed light on the design and implementation of other fast data analytics systems in the future.

Bibliography

[1] C. Gupta, S. Wang, I. Ari, M. C. Hao, U. Dayal, A. Mehta, M. Marwah, and R. K. Sharma, "Chaos: A data stream analysis architecture for enterprise applications.," in *CEC* (B. Hofreiter and H. Werthner, eds.), pp. 33–40, IEEE Computer Society, 2009.

[2] EsperTech, "Espertech Inc. Event Stream Intelligence." `http://www.espertech.com/`, May 2013.

[3] P. Vincent, "The tibco blog: The cep market 2009: a brief history lesson." `http://www.thetibcoblog.com/2009/07/31/the-cep-market-2009-a-brief-history-lesson/`, July 2009.

[4] T. Economist, "The leaky corporation." `http://www.economist.com/node/18226961`, Feb 2011.

[5] IBM, "Why big data?." `http://almaden.ibm.com/colloquium/resources/Why\%20Big\%20Data\%20Krishna.PDF`, March 2004.

[6] P. N. Tan, *Association Analysis: Basic Concepts and Algorithms*, ch. 6. March 2006.

[7] NIST, "NIST planinng report, the economic impact of inadequate infrastructure for software testing, 02-3," tech. rep., NIST, May 2002.

[8] R. Olivia, *Business Intelligence Success Factors: Tools for Aligning Your Business in the Global Economy*. 2009.

[9] Wiki, "Business intelligence BI." `http://en.wikipedia.org/wiki/Business_intelligence`, May 2013.

[10] W. R. Schulte, "The growing role of events in enterprise applications." `http://www.gartner.com/id=399662`, July 2003.

[11] I. Glossary, "Event-driven architecture (eda)." `http://www.gartner.com/it-glossary/eda-event-driven-architecture/`, July 2003.

[12] B. Michelson, "Event-driven architecture overview," *Patricia Seybold Group, Feb*, 2006.

[13] M. K. Chandy, "Event-driven applications: Costs, benefits and design approaches," 2006.

[14] J. Plummer and J. Johnson, "Complex event processing," April 2008.

[15] Wikipedia, "Event-condition-action (eca)." `http://en.wikipedia.org/wiki/Event_condition_action`, March 2013.

[16] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *SIGMOD Rec.*, vol. 34, pp. 18–26, June 2005.

[17] N. Tatbul, U. Çetintemel, S. B. Zdonik, M. Cherniack, and M. Stonebraker, "Load shedding in a data stream manager," in *VLDB*, pp. 309–320, 2003.

[18] Wikipedia, "Approximation algorithm." `http://en.wikipedia.org/wiki/Approximation_algorithm`, May 2013.

[19] I. Ari, E. Olmezogullari, and O. F. Celebi, "Data Stream Analytics and Mining in the Cloud," *IEEE DaMiC*, 2012.

[20] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, (New York, NY, USA), pp. 407–418, ACM, 2006.

[21] Y. Ahmad, "Lecture notes: Data stream processing." `http://www.sdss.jhu.edu/~budavari/tamas/teaching/2013-01/PraSci-22-Yanif-streams-overview.pdf`, May 2013.

[22] A. Ishii and T. Suzumura, "Elastic stream computing with clouds.," in *IEEE CLOUD* (L. Liu and M. Parashar, eds.), pp. 195–202, IEEE, 2011.

[23] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. 20th Int. Conf. on Very Large Data Bases (VLDB)*, 1994.

[24] C. Borgelt, "An Implementation of the FP-growth Algorithm," *ACM Workshop of Open Source Data Mining Software, (OSDM)*, pp. 1–5, 2005.

[25] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities; Data Mining: Next Generation Challenges and Future Directions," *AAAI/MIT*, 2003.

[26] N. Jiang and L. Gruenwald, "Research issues in data stream association rule mining," *In SIGMOD Record*, vol. 35, pp. 1–5, March 2006.

[27] P. S. Yu and Y. Chi, "Association Rule Mining on Streams," *Encyclopedia of Database Systems*, 2009.

[28] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining Data Streams," *ACM SIGMOD Record*, vol. 34, no. 2, 2005.

[29] Y. Zhu and D. Shasha, "Statstream: Statistical Monitoring of Thousands of Data Streams in Real Time," *Proc. of VLDB*, 2002.

[30] I. Ari and O. F. Celebi, "Finding Event Correlations in Federated Wireless Sensor Networks," *Federated Wireless Sensors and Systems Workshop (FedSenS), In Conj. with IWCMC*, 2011.

[31] IBM, "What is Big Data?." http://www-01.ibm.com/software/data/bigdata/, May 2013.

[32] Apache, "Apache hadoop project." http://hadoop.apache.org/, April 2013.

[33] S. Melni and et al, "Dremel: Interactive Analysis of Web-Scale Datasets," *Proc. Of The VLDB Endowment*, vol. 3, pp. 1–16, 2010.

[34] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems," *ACM PODS*, vol. 3, June 2002.

[35] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik., "Aurora: A new model and architecture for data stream management," *VLDB Journal*, vol. 12(2), August 2003.

[36] IBM, "IBM Infosphere Stream Mining Toolkit." http://www-01.ibm.com/software/data/infosphere/streams/, May 2013.

[37] Weka, "Weka Machine Learning Toolkit." http://www.cs.waikato.ac.nz/~ml/weka/, May 2013.

[38] O. Celma, "Last.fm Dataset-1K Unique Users." http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html, May 2013.

[39] E. Tang, P. G.Frankl, and Y. Deng, "Test coverage tools for database applications," 2006.

[40] D. Chays, Y. Deng, P. G. Frankl, and E. J. Weyuker, "An AGENDA for testing relational database applications," in *Software Testing, Verification and Reliability*, p. 2004, 2004.

[41] E. Lo, C. Binnig, D. Kossmann, M. T. zsu, and W.-K. Hon, "A framework for testing DBMS features.," *VLDB J.*, vol. 19, no. 2, pp. 203–230, 2010.

[42] K. Burr and W. Young, "Combinatorial test techniques: Table-based automation, test generation and code coverage," in *Proceedings of the Intl. Conf. on Software Testing Analysis and Review*, pp. 503–513, West, 1998.

[43] Y. Lei, "Lecture on software testing and maintenance." http://crystal.uta.edu/~ylei/cse4321/data/combinatorial.pdf, August 2011.

[44] D. Anderson, "Extraction, translation, and load (ETL) technologies part1." http://dbbest.com/blog/extract-transform-load-etl-technologies-part-1/, December 2012.

[45] M. Hemphill, "Building business intelligence BI." http://exonous.typepad.com/mis/2004/03/building_busine.html, March 2004.

[46] U. Fayyad, G. Piatetsky-shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, pp. 37–54, 1996.

[47] Wikipedia, "Data mining." http://cra.org/ccc/docs/init/bigdatawhitepaper.pdf, May 2013.

[48] I. Glossary, "Complex event processing (cep)." http://www.gartner.com/it-glossary/complex-event-processing, May 2013.

[49] T. BusinessEvents, "Complex event processing." http://www.tibco.com/products/event-processing/complex-event-processing/default.jsp, May 2013.

[50] D. Gyllstrom, E. Wu, H. jin Chae, Y. Diao, P. Stahlberg, and G. Anderson, "Sase: Complex event processing over streams," in *In Proceedings of the Third Biennial Conference on Innovative Data Systems Research*, 2007.

[51] L. Golab and M. T. Özsu, "Issues in data stream management," *SIGMOD Rec.*, vol. 32, pp. 5–14, Jun 2003.

VITA

Erdi Ölmezoğulları was born in İzmir on 4^{th} of June, 1986. He received a B.Sc. degree in 2008 1^{st} ranked in Geophysical Engineering, as well as he has a B.Sc. degree in Computer Engineering (Double Major) in 2010 from Dokuz Eylül University, İzmir, Turkey.Then, He worked approximately 4 months in software company as a software engineer, İzmir. After he had left from company, He started M.Sc. program on Computer Engineering in Özyeğin University, İstanbul. He has worked in Cloud Computing Research Group(CCRG) on Complex Event Processing and Stream Data Mining under supervision of Dr. İsmail Arı since 2011.