ISSN No. 0976-5697

Volume 6, No. 2, March-April 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Performance Evaluation of Microsoft StreamInsight as a Complex Events Processing Platform

Ramakrishnan Iyer
Research and Innovation Tech CoE
IGATE Global Solutions Ltd.
Mumbai, India

Radharaman Mishra
Research and Innovation Tech CoE
IGATE Global Solutions Ltd.
Mumbai, India

Abstract: Internet of Things (IoT) is one of the most-hyped emerging areas of technology in recent times. With things communicating with each other over the internet, it is only natural to expect that the volumes of the data collected and transmitted will become massive. Therefore, analysing such explosive amounts of data in near real-time, to generate actionable information out of these, is an imperative. Complex Event Processing (CEP) refers to the technology that enables near real-time processing of such data streams, without having the need to store and retrieve them from databases. Microsoft StreamInsight is a product that enables Complex Event Processing. In this paper, we talk about how Microsoft StreamInsight can be used to quickly develop a flexible high performance CEP solution. We have given an overview of "IStreamAnalytics", a platform that we have built using StreamInsight. Further, the performance of StreamInsight as a CEP solution is evaluated through a series of experiments.

Keywords: Microsoft StreamInsight, Complex Events Processing, Internet of Things

I. INTRODUCTION

Due to the availability and affordability of wireless data communication for personal and enterprise uses, enterprises are witnessing an exponential increase in volume of data flowing through its systems. This influx of data is going to rise higher and higher as the number of connected devices increases. According to an estimate by Cisco IBSG the number of connected devices would be touching 50 billion by 2020[1]. Traditional database technology platforms and related development approaches are not designed to handle this unprecedented inflow of data for deriving information that can be used for meaningful business purposes. Enterprises need a platform that will enable them to go through these data streams, analyze them in near real time and provide insight into it, while ensuring good performance. This has led to the emergence of Complex Event Processing (CEP) technology.

CEP is a technology for high-throughput, low-latency processing of event streams, including data coming continuously from systems, devices and sensors. CEP will play an important role in various verticals such banking, financial services, manufacturing, retail, telecommunication and healthcare etc. which produce high volumes of streaming data and need real-time analysis. Given the huge market potential for CEP, many players are entering into CEP market products such as Esper, HStreaming Cloud, Microsoft StreamInsight, Oracle CEP, SAP Sybase Event Stream Processor, StreamBase Systems' Event Processing Platform etc. IGATE has developed IStreamAnalytics as a platform using Microsoft StreamInsight with various modules that can be used to develop a CEP solution using the Microsoft .Net framework.

II. KEY CONSIDERATIONS FOR A CEP IMPLEMENTATION

With the limitation of some analytical tools needing to store large volumes of data, CEP has emerged as an alternative for real-time detection and event based systems to analyze large volumes of data quickly. The overall approach of a CEP implementation has profound technological and architectural implications such as:

- Ability to read the data from multiple sources of data where the format of the input data could be different.
- Events processed in memory as it occurs to avoid any storing of data.
- Configurable business rules defined by a business analyst with no need to know the nitty-gritties of the technology.
- Data aggregation and analysis to provide high level summary, trends and statistics.
- Ability to detect certain patterns in events and trigger off actions that needs to be taken.
- Guarantee good performance, high availability (24x7) and scalability, since the number of connected devices might increase drastically.

III. IGATE ISTREAMANALYTICS SOLUTION

A. A Generic Platform for Processing Complex Events

We investigated whether Microsoft StreamInsight[2] can be leveraged as a technology platform for building intelligent solutions with simple to medium complexity, which could be a technology enabler for use cases in the context of Internet of Things.

Our objective was to look at the StreamInsight from two different perspectives:

- Observe its capability to consume high volume of events data in a very short span of time. We used plain command line based .NET programs both for brevity and to minimize the overheads.
- Look at StreamInsight as a way to build an application platform surrounding it for enterprise level scenarios. We developed a multilayered platform to visualize enterprise scenarios. We designed a tiered architecture to see how the complexity in architecture impacts and also to see how such architectures can be built.

Standardization is the way to generalization. We developed IStreamAnlaytics as a platform for building CEP based solutions using Microsoft StreamInsight. Our approach considers the following premises as a foundation:

- The event data should be encapsulated in a class that serves various business scenarios, termed as payload in StreamInsight. We had to ensure that all the events will have the fields like event source, some field identifying the data it carries, the event data for the field and the date and time of the event (there could be a start and end values for the date and time in certain cases).
- The business rules are created in xml format and defined separate from the code and represented clearly using the logical operators (e.g., setting the threshold values for the events). The rules were given specific names and provided with alert messages that would be raised in case of a violation.
- For most of the practical scenarios, the sources generating the events (sensors etc.) won't be accessing the StreamInsight server directly. Hence, there has to be a gateway that would accept the events, do basic validations, filter and then pass it to the StreamInsight engine ensuring quality of data.

Following diagram shows the high level architecture components of the IStreamAnalytics solution:

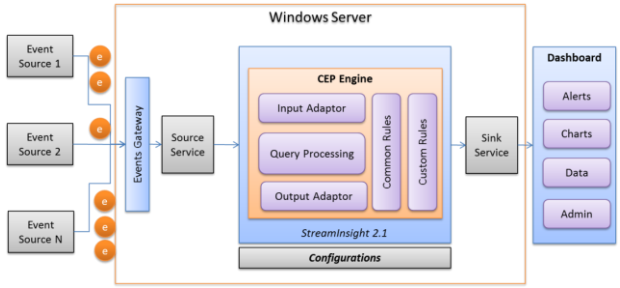

Figure 1: IStreamAnalytics Architecture

Two more significant design decisions that we took were the model (type) of the event and the mode of handshake for the StreamInsight with the source and sink (StreamInsight names destination as sink). StreamInsight allows processing of point events (that have just a start time stamp and it is considered to last on tick in time), interval events (having both the start and the end time stamp) and edge events (it has both the start and end time stamp, but they come separately)[3]. We selected point model for two reasons, its similarity with the sensor behavior and for the sake of simplicity.

Thus, we have 5 different building blocks for the overall solution. Here is a short description for all of them:

B. Architecture Components and Configurations

The overall architecture of the solution has to be in line with the development and integration framework provided by StreamInsight[4]. We leveraged the Source, Sink and Adaptors based model to build the IStreamAnalytics platform.

- Event Gateway: This is the contact point for the various input sources (e.g., device sensors-speedometer of a vehicle, device on a patient, smart meters etc.). It does the necessary transformation to make the payload standard and sends the event to the Source Service.
- Source Service: It does the integration between the Event Gateway and the CEP engine. All the events that are sent from the Gateway reach to StreamInsight through the "Source Service".
- CEP Engine: The actual CEP program that runs within the StreamInsight. The engine will house two types of rules, pre-built (generic rules specified in the XML files) and custom rules (non-generic complex application specific rules not covered by the pre-built rules).
- Sink Services: It does the integration between the CEP engine and the Dashboard by sending the query results and the events data to the Dashboard.
- Dashboard: Shows the data and the query results (alerts, aggregates etc.) generated by the CEP.

Here is a sample of the simple rules using XML syntax:

```
<Parameter>
      <Name>vehicle_speed</Name>
      <MaxValue>60</MaxValue>
      <MinValue>10</MinValue>
      <Alert-message>
            Vehicle speed: crossing threshold value
            (allowed between 11-59)
      </Alert-message>
      <RuleName>RuleWithinThreshold</RuleName>
</Parameter>
```

This rule applies to an event that comes with a field identifier name "vehicle_speed". The rule has a name "RuleWithinThreshold" and it states that if the value goes below 10 or above 60, then an alert needs to be triggered.

The "Alert-message" can be used for displaying the notification in the Dashboard.

The rules are set in sync with the payload structure. As long as the source data can be encapsulated in the payload format and rules specified in the given XML format, the CEP engine will process it and any threshold violation will get triggered, which will be displayed in the "Dashboard" almost instantaneously.

Taking it to be of further use we wrote a few more type of rules, namely:

- Check for only minimum or only maximum values
- Raise alert as soon the value falls within the specified range
- Define threshold on the aggregates (for the example when the total distance travelled in last 24 hour by a vehicle goes beyond a certain limit)

C. Design Considerations

Our intention was to develop a CEP solution that can be flexible enough to work with multiple sources of data in any format and test various types of scenarios that could be implemented with StreamInsight. The high level architecture components were designed as independent modules and integrated through services. Additionally this approach allowed us to run multiple instances of the "Event Gateway" and "Dashboard" applications in parallel. Following

illustration depicts the various scenarios that we were able to create and test validating the flexibility and scalability of our approach:

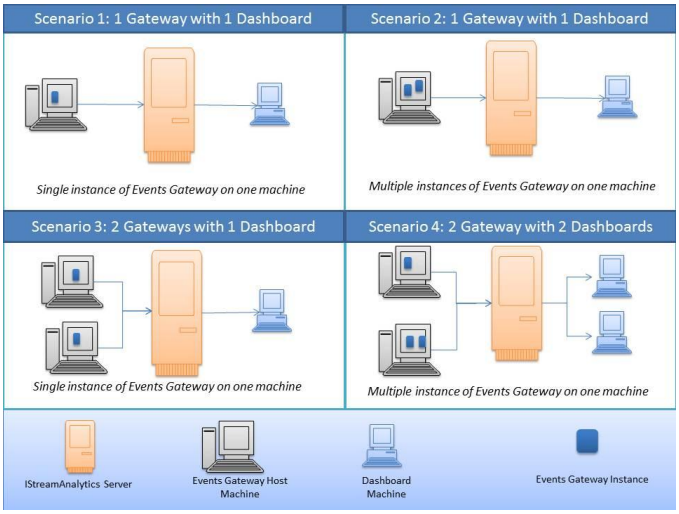

Figure 2: IStreamAnalytics Test Scenarios

IV. PERFORMANCE EVALUATION

With respect to ensuring a good performance for a CEP solution, some key points to factor in are:

- System needs to process large volumes of data published at a high speed, which will need a matching infrastructure
- Effective, meaningful and cost effective mining and analysis of the input events needs a robust analytics platform
- Since the number of connected devices might increase drastically, the architecture needs to be scalable
- Time is very critical in Complex events processing, since there is a need to respond to critical situations almost at near real-time
- Considering the high volume of data coming in at high speed, need to ensure the data quality
- High network bandwidth is required to read all the raw data generated by millions of connected devices

To evaluate the performance of StreamInsight, we set some key objectives such as:

- To find the number of records processed every second which will not depend on the number of records sent for processing
- To evaluate the performance based on the multiple sources of data sent
- To find how StreamInsight handles the concurrency of events
- To find how the complexity in the architecture impacts performance

A. Running the Performance Test

Our objective was to evaluate different types of scenarios that could be mixed to articulate architectures depending upon the business and performance needs. We tested the following scenarios:

Check StreamInsight throughput

Large number of events were posted to StreamInsight and persisted in some form without doing much processing on it. We created a simple StreamInsight program that leveraged the text and database adaptors for connecting with the source and the sync. This scenario was tested for two different cases, one when the output data (that was essentially same as of the data posted) was written to a text file and the other when it was stored into a database. This test was to run to check the StreamInsight's claim for handling the high volume scenarios and also to get an insight into measure how the overall time increase as the processing is added.

Check StreamInsight throughput with simple processing

Large number of events being posted to StreamInsight and persist it in some form, StreamInsight was asked to calculate aggregate for the events data periodically. We created a simple StreamInsight program that leveraged the text and database adaptors for connecting with the source and the sync. This scenario was again tested for two different cases, one when the output data (aggregate value) was written to a text file and the other when it was stored into a database. This test in contrast to the first one was to get some clue into how a simple calculation impacts the overall processing time.

Check StreamInsight throughput for complex architectures

Last scenario was to test how the StreamInsight based CEP solution will perform when complicated multi-layer architecture is in place. IStreamAnalytics platform, which we have explained above, was built to simulate this scenario. And as we have seen it has various components and services to generate data, send it to the CEP Server, and show it to the user.

This test scenario needs some detailed explanation as given below:

Simulating the Sources

CEP solutions built on top of StreamInsight could process thousands of events every second. Keeping the limitations (infrastructure and time etc.) in mind we kept a moderate goal in front of us; generate a handful of events (up to 50) every second. Events details, along with a generation date time, were kept in flat files that the "Event Gateway" can read and send to the StreamInsight through "Source Service". Keeping the date time stamp in the file allows spacing the events in time but it also restricts in creating a real life scenario where the events could be bubbling. Keeping the objective of trying out as many scenarios as possible we also allowed the source service to replace the event date time with the date time when it is being sent to StreamInsight, if required.

Dealing with multiple Sources

CEP solutions will often be receiving events that will be coming from more than one source. It would not be unlikely that multiple events were generated at same point of time from one or more devices. These events will be assumed to be duplicates, only one of them will be considered a valid event for a device although all of them could be potentially a valid event. There can be several approaches to tackle this problem.

Create multiple streams for different sources and combine them together (using UNION on all of the sources) within the StreamInsight. This approach could be workable when there are a limited number of fixed sources. If the number of sources is changing or too large this would be complex. Apart from that scenarios when the sources are not providing any data could complicate the solution.

Add tick level differences to the event time stamp: StreamInsight can differentiate the event times at the tick level (10 million ticks add up to a second). The event gateway can keep a counter, that increment itself every time there is a new event and resets itself every second (or when there is change in the time). We are assuming that the events are arriving with time precision in seconds. This tick is added to the event time stamp and that makes it unique to StreamInsight. Since there can be up to 10 million ticks in a second we have a very large number of records that could be made unique. For certain cases a refined approach could be applied. The payload could have two date time fields. One of them will hold the date time that the event source provided and second could be the date time generated by the "Event Gateway" that is the current date time. "Event Gateway" can capture the time when it is about to send the event and add the tick variation to it. This approach will ensure that the events are made unique without losing the information about the time when they were actually generated.

Handling the Concurrency of the Events

CEP solutions by nature have to deal with large number of events. The timing of the event is important because, unless specified otherwise, the events will need to be ordered as per time. Overriding this basic rule, although possible, brings its own possible complications and challenges. The approach and the solution developed for handling the multiple sources that makes the events unique across the time scale, worked for this issue as well. Since the event's time has been added with a tick variation, even the events having the same date and time stamps don't have the matching date time stamp at the tick level. This keeps the original time stamp of the data unaltered at the HH:MM:SS level, while still making them unique.

Managing the time for aggregations

Usually we calculate the aggregate to know the cumulative value over a period of time. In CEP scenarios there could be cases where apart from the period of time, when the aggregation happened also might be critical. Consider a hypothetical scenario where we are calculating the sum of the temperature received from all the engine parts and also want to know exactly when the temperature was detected to cross a threshold value. Adding to the complexity, assume that the processing is being executed not in real time, rather is based on the input feeds that have been received from the engine sensors. When this data is fed to the CEP engine, the aggregations will go as defined but the aggregation queries will not record the time of the aggregation correctly. As a solution this time could be the time stamp of the last event from the event window and needs to be captured and reported along with the aggregate data. When the stream is analyzed in real time and CEP receives the data almost instantaneously, this time could be the time of aggregation done by the CEP itself.

After reading the events file in a defined format and sending the events to the StreamInsight and before displaying them in a dashboard (these activities will remain more or less similar in any of the approach), here is what we did within the CEP solution:

- Integrate the event source with the event generator which will be reading data based on a format specified in the xml file
- Define and bind with the sinks (destination, the dashboard)
- Collect the events as they arrive
- Execute LINQ queries on the event stream for:
  - Finding alerts based on defined rules in xml files
  - Calculating aggregates based on defined parameters in xml files
  - Executing temporal comparisons which will need to be custom developed
  - Transmitting the results including the event details to the dashboard
- Perform the cleanup once the execution is over

All this was covered within just a few LINQ queries[5] (a scripting language provided with Microsoft StreamInsight) and some code for managing the configurable rules, besides a few lines of code for setting up the source and sinks. In addition to what can be configured, StreamInsight provides as in-built functions:

- Detection of the latest event as it arrives
- Automatic execution of the LINQ queries every time a new event is there, including the aggregates
- Ability to move backward and forward in time in the streams and incorporating the altered events into the LINQ queries
- Easy binding with the source and sink adaptors.

As planned, designed and developed we created following two different sets of configurations (data files, their definitions, set of rules etc.):

- First, dealing with the engine parameters transmitted through the sensors of a vehicle while driving
- Second, real time inventory related feeds from a chain of retail stores

We tried to run the IStreamAanalytics with both the configuration files (and not a single line of code change). As expected the "Dashboard" displayed the appropriate alerts based on the data successfully.

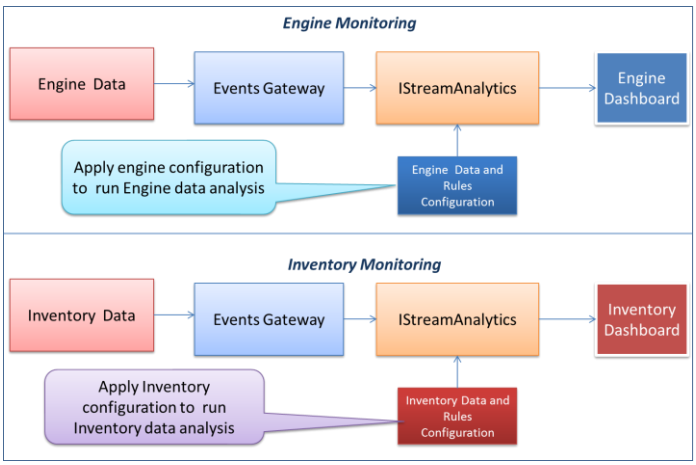

Figure 3: Running IStreamAnalytics for different business cases

B. The Results

Following table captures the results of the first two performance scenarios, where we checked for the StreamInsight throughput without any processing and with a simple aggregate calculation.

Table I. First Two Performance Scenarios

Scenario	Machine No.	Number of Records processed	Total Time taken in Milli Seconds	Records Processed Per Second	Comments
Scenario 1	M1	500	264	1894	Events stored to text file without processing
Scenario 2	M1	5211	1702	3062	Events stored to text file without processing
Scenario 3	M1	20000	6437	3107	Events stored to text file without processing
Scenario 4	M1	35500	11700	3034	Events stored to text file without processing
Scenario 5	M1	500	1730	289	Events stored to DB without processing
Scenario 6	M1	5211	17115	304	Events stored to DB without processing
Scenario 7	M1	20000	66468	301	Events stored to DB without processing
Scenario 8	M1	35500	156117	227	Events stored to DB without processing
Scenario 9	M1	500	143	3497	Aggregate calculations stored to text file Aggregate calculated every 10 seconds
Scenario 10	M1	500	136	3676	Aggregate calculations stored to text file Aggregate calculated every 20 seconds
Scenario 11	M1	500	127	3937	Aggregate calculations stored to text file Aggregate calculated every 30 seconds
Scenario 12	M1	500	338	1479	Aggregate calculations stored to DB Aggregate calculated every 10 seconds
Scenario 13	M1	500	280	1786	Aggregate calculations stored to DB Aggregate calculated every 20 seconds
Scenario 14	M1	500	218	2294	Aggregate calculations stored to DB Aggregate calculated every 30 seconds

Notes:

- M1 is the machine where StreamInsight is running. The data files and SQL Server Database is also running on the same machine.
- The machines configuration: Intel Core 3.10 GHz Processor with 2.85 GB of RAM

A quick look at the table reveals that the number of records processed every second is fairly consistent for each of the scenarios.

- Data pushed to text file (scenarios 1, 2, 3 and 4)
- Date pushed to the database (scenarios 5, 6, 7 and 8)
- Aggregates stored to the text file (scenarios 9, 10 and 11)
- Aggregates stored to the database (scenarios 12, 13 and 14)

We can draw the following conclusions:

- As our test data shows StreamInsight can consume thousands of records and perform simple calculations on them every second, when running on a simple desktop machine. We can hereby confirm that it will be able to handle higher number of records and complex business logic on a server class high end machine.
- Adaptors are important, as when the data is directly fed from a text file to and written into flat file it performs better than the scenario when the data is stored into a database.

Following table depicts a snapshot of the third test scenario where a layered architecture was in place and business rules were more complicated:

Table II. Third Test Scenarios

Scenario	Machine No.	Number of Records processed	Total Time taken in Seconds	Records Processed Per Second	Comments
Scenario 1	M1	1000	22	45	One file running on M1
Scenario 2	M1	5000	108	46	One file running on M1
Scenario 3	M1	20000	428	47	One file running on M1
Scenario 4	M1	35500	749	47	One file running on M1
Scenario 5	M1	50000	1111	45	One file running on M1
Scenario 6	M1	75000	1621	46	One file running on M1
Scenario 7	M1	25000			Two files running in parallel on M1.
	M1	25000	935	53	
Scenario 8	M2	20000	461	43	One file running on M2

Notes:

- M1 is the machine where StreamInsight is running. This machine also hosts the "Source Service", "Destination Service" and "Dashboard" components.
- M2 is a different machine in the network and connects to the StreamInsight instance running on M1.
- Both the machines have the same configurations (Intel Core 3.10 GHz Processor with 2.85 GB of RAM)

If we look closely at the data here are a few useful observations:

- The number of records processed every second is fairly constant irrespective of the total number of records.
- In the "Scenario 7", there is an improvement in the performance when two instances of the "Events Gateway" application are running together simultaneously processing two different files.
- In the "Scenario 8", when the file is being processed on M2 that connects to the StreamInsight instance running on M1 through the "Source Service". Note that the service is still running on M1.
- The overall performance drops significantly in a layered architecture because we are now able to push far lesser number of records to the StreamInsight in a given unit of time.

Here are some of the important observations from the overall exercise.

- Loosely coupled adaptor based architecture is required for the enterprise level CEP solutions. It allows separation of concerns and hence optimization of the architecture flexibility. But it also adds a lot of overheads. If the throughput related requirements are big, a tradeoff approach has to be found.
- Since the StreamInsight can process thousands of records in a second the gateway application/ service and adaptors that will feed the events to it have to be designed especially to receive and send the expected inflow of events.
- The gateway and adaptors should be designed for handling the scenarios where multiple instances of them could be deployed to share the workload.
- The events files could be read in sequence and that put a limitation on the number of records that could be pushed in a second. To break this barrier the event generator was designed in a way that multiple instances of it could be made running and each can read a separate file and connect to the same source service. In the enterprise context, architects will need to pay special attention to the flow of incoming events, how fast they will be arriving, whether there will be a peak time for them, whether they will need some processing before entering CEP engine, whether the servers have enough capacity to process them, etc.
- The approach of payload having two date time fields where one was based on the event source and the second one based on the current date time suited real-time customer requirements better since it allows accommodating a very large number of events with unique time stamps and doesn't add any additional complexities in the process. One of our use cases dealt

with a scenario where each event was supposed to be unique and should not get ignored. There could be business cases where events with matching details (especially the date-time stamp) will be considered as duplicates and hence need not be processed. Depending upon the nature of the events this approach might need to get adjusted.

These observations elucidate the importance of designing the application architecture that suits the business requirements. Apart from that the design of the data provider applications and adaptors that connect the sources and sinks (destinations) to the CEP engine are the critical factors having impact on the performance and stability of the overall solution. StreamInsight capabilities, when combined together with a suitable architecture, can definitely deliver high performing CEP solutions.

V. CONCLUSION

CEP has opened up a tremendous opportunity for enterprises to leverage the large amounts of data and bring a new transformation in how to detect any change in parameters or data, use this information to trigger new events to help save costs and better decision making. StreamInsight capabilities when combined together with the suitable architecture can definitely deliver high performing CEP solutions.

VI. REFERENCES

[1] The Internet of Things How the Next Evolution of the Internet Is Changing Everything http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[2] Microsoft StreamInsight https://msdn.microsoft.com/en-in/sqlserver/ee476990.aspx

[3] Introducing Microsoft StreamInsight http://web.cecs.pdx.edu/~tufte/410-510DS/readings_files/Microsoft%20CEP%20Overview.pdf

[4] StreamInsight Server Architecture https://msdn.microsoft.com/en-us/library/ee391536(v=sql.111)

[5] A Hitchhiker's Guide to Microsoft StreamInsight Queries http://blogs.msdn.com/b/streaminsight/archive/2012/08/01/a-hitchhiker-s-guide-to-streaminsight-2-1-queries.aspx