Detection of Weather Anomalies and Events of Interest Using Complex Event Processing

K. A. Gayanthika Udeshani and H. M. N. Dilum Bandara

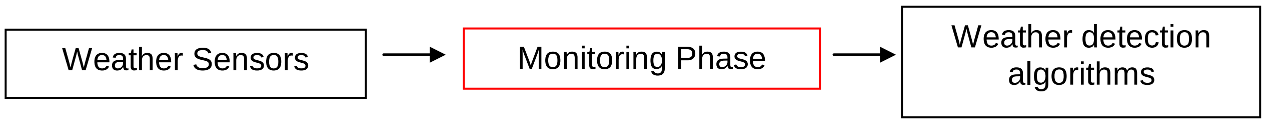
1. Introduction

Detection of weather conditions is of prime importance, as it directly affects day-to-day life. Several meteorological variables are used to measure the current state of the atmosphere at a given location and time. These meteorological variables are useful in determining both short and long-term changes in the atmosphere. However, this is quite challenging as it requires real-time data collection, transmission, and processing. Further, it is important to identify anomalies in the sensor data such as missing or incorrect readings from weather stations. This research aims at detection of these weather anomalies and events of interest using Complex Event Processing (CEP).

Figure 1 illustrates the workflow of a typical weather/climate observatory system. Meteorological variables are collected by sensors attached to the weather stations and the collected data are transmitted to the observatory system using a suitable network, for example, leased line, 3G/4G, or satellite links. The monitoring phase focuses on data pre-processing and identifying weather circumstances. This phase monitors the input sensory data and looks for abnormal values. While it is relatively easier to detect weather phenomena such as increased temperature or heavy rain, further processing is required to detect more complicated events such as storms and tornadoes.

The primary objective of this research is to provide the monitoring capabilities (as seen in Figure 1) to a typical weather/climate observatory. This is to be achieved by developing a CEP-based weather anomaly and event detection system that is scalable in terms of functionality, number of sensors, and meteorological variables. The proposed weather monitoring and detection system acts as an early warning system, as well as a trigger for the execution of complicated and resource-consuming weather detection algorithms. The system is also capable of providing solutions for common use cases found in weather detection and warning.

Figure 1. Workflow of a typical climate/weather observatory



This thesis demonstrates the application of CEP to weather anomalies and events detection systems. It applies an existing CEP engine, namely Siddhi, rather than recreating the basic CEP functionalities that are readily available. Siddhi is selected as it is an open source CEP engine, which is considered as one of the higher performing CEP engines around and capable of processing millions of events per second. Through this research we have identified a suitable subset of meteorological variables to receive continuous streams of sensor readings. These raw data streams may contain erroneous/faulty data, or some of the periodic data samples may be missing. Therefore, pre-processing algorithms are introduced to clean the incoming data streams. These incoming streams are fed into the Siddhi CEP engine. The users of the system can specify necessary queries to identify uncertain changes in meteorological variables, such as temperature, wind speed, pressure, and humidity. The Siddhi CEP engine is capable of processing these queries and matches them with the input data streams to identify relevant patterns. Once an interesting anomaly or an event is detected, the system generates an alert(s) and sends it/them as notifications to the relevant sub-systems (i.e., the third phase of a typical weather/climate observatory) for further processing and issuing warnings.

This research also suggests several changes to CEP engines, particularly Siddhi, to make it more suitable for real-time weather detection systems in different viewpoints such that we can detect more complicated weather patterns and achieve high performance with necessary changes of weather data representations in CEP.

The rest of the paper is organized as follows. Section 2 presents the related work; it discusses CEP, existing weather detection systems, and meteorological variables. Methodology is presented in Section 3. It presents the high-level architecture, use cases and the details of the proposed system. Section 4 presents the performance analysis and evaluation. Finally, concluding remarks, problems encountered, and the future work are discussed in Section 5.

2. Related work

While several approaches have been used to address weather anomalies and events detection, CEP can be identified as the most suitable contemporary technique to implement these use cases. For example, the neural network based weather detection system[11] needs to train a separate neural for every use case. However, using the Siddhi CEP engine we can easily add a new query to enhance the functionalities. Furthermore, several queries can be defined parallel to implement these use cases rather than doing multiple comparisons sequentially.

The LEAD project was very closely related to this research area, which uses CEP in weather detection[1]. They had used Calder as the CEP engine. We used Siddhi as the CEP engine which is an open source CEP engine, and it can be applied in weather anomalies and events detection easily. Moreover, Calder lacks the ability to dynamically add new data formats and user-defined functions, whereas Siddhi has the capability of adding user-defined functions easily. The system was able to implement the use cases by only adding custom functions without having to change the Siddhi codebase, and that concludes the effectiveness of the extension points of Siddhi CEP engine.

2.1 Complex Event Processing

CEP based systems receive events from multiple independent simple event streams of different event sources[2]. Complex event detection (aka. event pattern matching) is the core functionality of such a system. The user needs to provide event pattern rules to detect specific events. These event pattern rules can be defined in a SQL-like event processing language[2]. The CEP engine listens to incoming events and detects event patterns matching with the specified queries, and then sends alerts to relevant systems. Most CEP engines can analyze and detect thousands of events per second. Therefore, CEP technology derives intelligence from real-time event data analysis. The ability to analyze large streams of incoming events in real-time and detect relevant events makes them a suitable alternative for modern weather detection.

There are several CEP engines in the market[1]. These will provide the runtime to perform the CEP. Siddhi is one of the open source CEP engines which can process millions of events per second. Siddhi is implemented as a Java library that allows initiating multiple instances. Each Siddhi engine is single threaded. Further, this CEP engine supports partitions which allows the users to isolate the processing into sub-processors and speed up the execution[7].

2.2 Weather detection

Weather is the state of the atmosphere at a particular place and time, where it concerns hot or cold, dryness, cloudiness, and rain. Common weather phenomena on earth include wind, rain, snow, dust storms and cloud. Less common events are tornadoes, hurricanes, typhoons and ice storms (natural disasters). These weather conditions occur due to air pressure differences between different places[3]. Different kinds of techniques are used to detect sudden changes in weather, which can cause major disasters[4]. Early detection of such situations can help humans in several ways. There are several weather detection algorithms, which can be applied in weather detection[5].

Different types of scales are used to define weather alerts depending on the wind speed and the nature of the area. Saffir-Simpson hurricane category scale is used to describe the hurricanes in the Atlantic Ocean and Northern Pacific Ocean of the International Date Line. Enhanced Fujita Scale is used to describe the tornadoes in the United States and Canada [6]. Beaufort Wind Force Scale is to classify the wind alerts. Table 1 shows a sample classification – extracted from [6].

Table 1. Beaufort classification of wind speed (aka Beaufort Wind Scale)

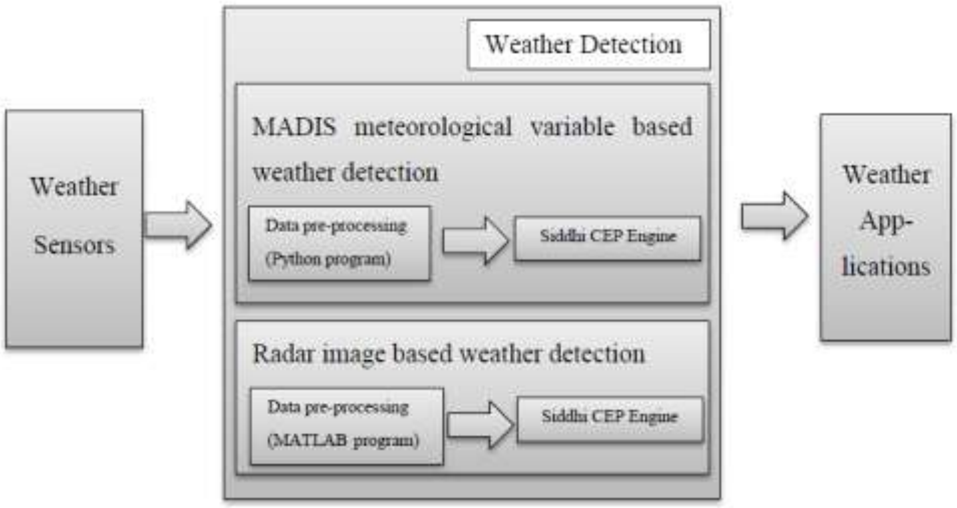
Wind Speed (mps)	*Beaufort Number*	*Alert*
11.2 – 17.4	6-7	Wind warning
17.5 – 24.6	8-9	High wind warning
24.7 – 33	10-11	High wind warning
33.1 – 49.2	12-13	High wind warning
Over 49.2	14-16	Extreme wind warning

While many different meteorological variables have been introduced, precipitation, wind, temperature and cloud cover are used in most of the weather applications [8]. Reflectivity, differential phase, correlation coefficient, and Doppler velocity are introduced in [9]. Relative humidity is used to predict rain in [10]. Wind speed, wind direction, dry bulb temperature, wet bulb temperature, relative humidity, dew point, pressure, visibility and amount of cloud with some daily meteorological variables such as gust of wind, mean temperature, maximum temperature, precipitation, mean humidity, mean pressure, sunshine, radiation and evaporation are used in [11]. According to the literature the most relevant meteorological variables are relative humidly, pressure, air temperature, wind direction, wind speed, and accumulated precipitation. The proposed system is able to detect the significant changes of the selected meteorological variables using the Siddhi CEP engine.

3. Methodology

The high-level architecture of the system is presented in Figure 2. The system is mainly divided into two sub-parts based on the input types: meteorological variable based weather detection and the radar image based weather detection. Meteorological variable based weather detection considers certain values of meteorological variables extracted from MADIS (Meteorological Assimilation Data 37 Ingest System) [12] and compares with predefined scalar values. Radar image based weather detection considers reflectivity values of radar images and compares them with a threshold.

Figure 2. The high-level architecture of the proposed system


Weather Detection
MADIS meteorological variable based weather detection
Data pre-processing (Python program)
Siddhi CEP Engine
Radar image based weather detection
Data pre-processing (MATLAB program)
Siddhi CEP Engine
Weather Sensors
Weather App-lications

3.1 Data preprocessing

In the meteorological variable based weather detection, the data may contain unwanted tags and duplicates. Hence, data needs to be pre-processed before proceeding with further calculation. This stage makes the raw data ready for the event pattern matching process, where the values of selected meteorological variables are time stamped.

Radar images play a major role in weather detection, where colors of each pixel reflect the intensity of weather parameters like cloud density, precipitation, and wind speed at a given location. Therefore, it is important to analyze these complex radar images to identify suspicious pixels on them. The proposed system processes radar images and identifies precipitation pixels. It needs to pre-process these radar images and convert them to arrays, as the Siddhi CEP engine is unable to process images directly. Therefore, we converted the radar image to a 2D array (240x240) using a MATLAB program, before feeding into the CEP engine.

3.2 Stream definition

The weather stream is used to implement the meteorological variable based weather detection using Siddhi CEP engine. It defines the input data stream as follows:

```
define stream WeatherStream (timestamp double, wsid
     string, prov string, subPro string, rh double, pressure
     double, temp double, precip double, dd double, ff double,
     precip double, lat double, lon double)
```

The attributes of the WeatherStream is listed in Table 2. The timestamp is used to add the temporal aspect to data. These sensor data were available from different providers and weather stations. Weather station id, provider and the sub-provider are used to distinguish the location of the data. The study has considered relative humidity, pressure, temperature, accumulated precipitation, wind direction and the wind speed as the significant meteorological variables. Further, latitude and longitude are used to identify the location of weather stations.

Table 2. Weather stream definitions

Attribute	*Data Type*	*Description*
timestamp	Double	Reading time of a particular data
wsid	String	Weather station id
prov	String	Provider of weather data
subPro	Strubg	Sub-provider of weather data
rh	Double	Relative humidity (%)
pressure	Double	Station pressure (P)
temp	Double	Air temperature (K)
precip	Double	Accumulated precipitation 1 hour (m)
dd	Double	Wind direction (deg)
ff	Double	Wind speed (m/s)
lat	Double	Latitude
lon	Double	Longitude

The following radar stream is used to implement the radar image based weather detection using the Siddhi CEP engine.

```
define stream RadarStream(timestamp double, matrix string)
```

Table 3 lists the attributes of this radar stream. The timestamp is used to add the temporal aspect. Matrix is a 2D double array, which contains the reflectivity values of the radar image. Siddhi allows sending any object type in the stream and it checks the object type at the time it is being used. The stream is defined with the type string since the type array is not allowed. Therefore, the system sends the array to the stream and uses custom functions to manipulate them.

Table 3. Radar stream definitions

Attribute	*Data Type*	*Description*
timestamp	Double	Reading time of a particular data
matrix	String	240x240 matrix containing reflectivity values of radar image

3.3 Use cases

This proposed solution supports four use cases that cover the common scenarios found in weather anomalies, events detection, and warning systems. The first use case compares an input value to a predefined threshold while other use cases process the location-specific weather information. Next, we present how CEP can be applied in weather anomalies and events detection to solve these use cases. Moreover, separate queries are written to implement these use cases in Siddhi.

3.3.1 Comparing sensor data with predefined thresholds

3.3.1.1 Comparing meteorological variables with scalar values

This use case is implemented using the basic meteorological variables such as relative humidity, temperature, pressure, precipitation, wind direction, and speed. Queries are written to identify anomalies on these variables by comparing with scalar values.

As an example, Query 1 defines the partitions that consider the Beaufort classification of wind speeds (see Table 1). Beaufort classification partitions/labels incoming wind speed data based on a set of thresholds. These partitions can be used to increase the performance of the system when it deals with a larger number of inputs.

Query 1: define partitions

```
define partition WindSpeed by
range ff < 17.4 as 'WIND WARNING',
range ff >= 17.4 and ff <= 49.2 as 'HIGH WIND',
range ff > 49.2 as 'EXTREME WIND';
```

Query 2 checks the wind speed of the input events, to see whether they exceed 17.4 meters per second. If so, it creates a wind alert with weather station id and the time. This alert could be used to notify other systems.

Query 2: wind alert

```
from WeatherStream [ff > 17.4]
select wsid, timestamp
insert into windAlerts
partition by WindSpeed;
```

3.3.1.2 Identifying suspicious pixels of radar images

We need to analyze these complex radar images to identify suspicious pixels on them. The proposed system processes these radar images and identifies precipitations pixels in a given radar image. First, it converts the image to a matrix and process it. Each element of this matrix represents a reflectivity value corresponding to a pixel value. Radar image based weather detection considers these reflectivity values and compares them with a predefined threshold.

Scenario: The input event consists of a 240x240 matrix of double values. The threshold of the precipitation is ‘1’. Query 3 was used to look for precipitation pixels in the sample radar images.

Query 3: radar alert

```
from RadarStream [sample:getIsPrecipitation(1, matrix)]
select matrix
insert into radarAlerts;
```

3.3.1.3 Building queries with multiple meteorological variables

Certain weather detections are based on a combination of weather circumstances; thus, this use case considers combinations of meteorological variable values to identify such weather circumstances. As an example, wind speed and the wind direction can be used to issue wind alerts. Query 4 represents a wind alert, and it triggers when a high-wind condition towards the North East direction is identified.

Query 4: wind alert

```
from WeatherStream [ff > 17.4 and dd > 30 and dd < 60]
select wsid, timestamp
insert into windAlerts
partition by WindSpeed;
```

3.3.2 Identifying weather stations with defects and suggest alternative values

Sensor data may have defects such as missing values or incorrect values. Several queries are defined to check for weather stations with defects based on the location of the weather station. These queries compare the values of nearby weather stations within a circular area. These missing values indicate either there is a technical fault in the particular weather station or those missing values need to be replaced with the nearby weather station values. The following use cases mainly focus on weather anomalies detection.

GetIsNearStation

This function can be used to compare two locations and return whether they are situated with a significant distance. Distance between two weather stations is calculated using the latitude and longitude as the following equation [13]:

```
Distance =  ACOS(COS(RADIANS(90-Lat1)) *COS(RADIANS(90-Lat2))
            + SIN(RADIANS(90-Lat1)) *SIN(RADIANS(90-Lat2))
            *COS(RADIANS(Lon1-Lon2))) * 6371
```

Where, Lat1 and Lon1 are the locations of the first weather station and Lat2 and Lon2 are the locations of the second weather station. Given the latitude and longitude of two weather stations, it can find out whether the distance between both is within a specified limit using this equation.

GetIsNearTime

This function is used to compare the time difference between two weather stations. To propose alternative values, it needs to compare both reported times, but the timestamp is not always equal. Therefore, this function is used to match the time with a deviation value (900 s).

When there are missing values in a particular weather station for a time period of four hours, Query 5 will check nearby weather station's temperature value. This system is tested with hourly data and it is better to consider at least four consecutive missing values. Hence, it uses four hours as the time window. Further, it can notify whether there are any technical errors at the first weather station. Missing values are defined with -99999, so Query 5 compares the temperature value with -99999 and identifies whether the value is missing or not. It filters the values of nearby weather stations at the same time and suggests alternative values.

Query 5: find missing values

```
from WeatherStream [temp < -90000.0] #window.time(240 min)
as A join WeatherStream [temp > -90000.0] as B
on   sample:getIsNearStation(B.lat,   B.lon,   A.lat,   A.lon)   and
sample:getIsNearTime(A.timestamp, B.timestamp) select A.wsid, B.wsid,
B.temp
insert into tempAlerts
partition by WeatherStation;
```

When the user identifies that a particular weather station (e.g., D9545) does not have values for temperature, it can find alternative values from nearby weather stations. Query 6 returns the values from nearby weather stations when there is a missing value in the D9545.

Query 6: find alternative values

```
from WeatherStream [temp < -90000.0 and wsid == 'D9545']
as A join WeatherStream [temp > -90000.0] as B
on   sample:getIsNearStation(B.lat,   B.lon,   A.lat,   A.lon)   and
sample:getIsNearTime(A.timestamp, B.timestamp) select B.wsid, B.temp
insert into missingAlerts; ");
```

3.3.3 Identifying anomalies in weather data

Similarly, when there is any value which deviates from the expected value for any meteorological variable, it compares with nearby weather stations to identify anomalies. Query 7 will check whether the reported value as 100% for the relative humidity is acceptable by comparing with nearby weather stations.

Query 7. Identify anomalies

```
from WeatherStream [rh == 100.0] as A join
WeatherStream [rh == 100.0] as B on sample:getIsNearStation(B.lat,
B.lon, A.lat, A.lon) and sample:getIsNearTime(A.timestamp,
B.timestamp) select B.wsid, B.timestamp " +
insert into rhAlerts;
```

3.3.4 Identifying a nearby weather situation in a given location

Users are interested in knowing the current weather situation at a given location. Different types of wind alerts are defined in Table 1. For example, if someone wants to know whether a particular area is being issued a weather warning, then the system allows for querying the current wind speed with the specific range.

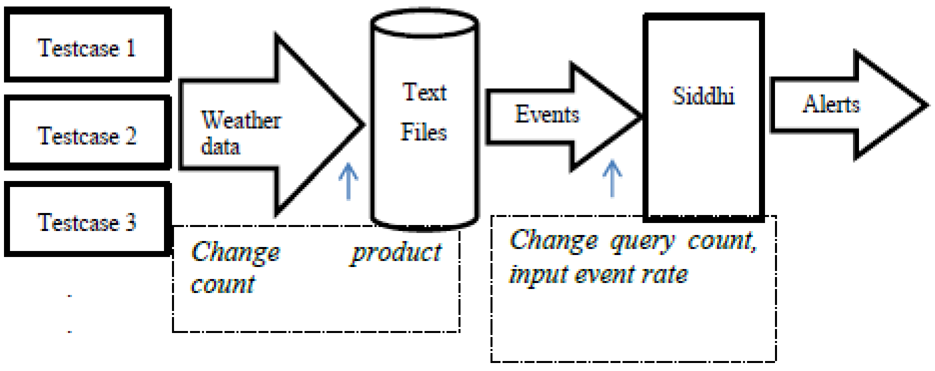
4. Performance analysis

The performance and accuracy of the proposed weather anomalies and events detection system were confirmed using a recent weather incident (winter storm "Juno") [14]. The system was able to match the input events with the predefined queries successfully. These use cases worked effectively with the existing performance of Siddhi, and the system was able to handle more than 10,000 events per second.

4.1 Emulation setup

Emulation setup is presented in Figure 3. The system is being tested in a single computer environment. Siddhi CEP engine is being used to test the defined test cases with the sample weather data. The product count, query count, and the input event rate are considered to test the system performance.

Figure 3. Emulation setup



4.1.1 System Setup

Hardware: This system is tested in a single computer with Intel i5 – 4200U CPU running at 1.6 - 2.3 GHz, 4 GB of RAM, and 64-bit Windows 8 Pro.

Software: The system is developed using Siddhi version 3.1.0 in a Java 1.8 environment. Apache Ant 1.9.6 is used as the build tool. It has used MATLAB 7.1 to pre-process radar images and to create a large sample data file using interpolation. The Python 2.7.3 environment is used to download and pre-process MADIS data.

4.1.2. Test Data

The system had considered storm 'Juno' for the verification [14]. The US National Weather Service had dropped all winter storm and blizzard warnings for Juno, which pounded on the 26-27th January 2015. Several locations in Massachusetts had picked up a large amount of snow. Most severe coastal flooding occurred in eastern Massachusetts, and the wind speed was 50-80 mph.

4.2. Comparing Meteorological Variables with Scalar Values

The MADIS API allows downloading weather sensor data from weather stations of state MA on 27th January 2015. D9545 is one of the weather stations in this area.

The test data file contains 13,063 numbers of sensor data readings of relative humidity, pressure, temperature, accumulated precipitation, wind direction and the wind speed for that day. The proposed system was able to identify that the temperature had gone down to 262.5K at 22:58 pm (UTC). Several test cases were executed with 13,063 input events. Table 4 shows the total number of detections of each test case.

Table 4. Test results

Test Case	*Actual Value*	*System Value*
Temperature < 263K	2,645	2,645
Wind speed > 17.4 (high wind)	42	42
Relative humidity == 100.0	168	168

This relative humidity was reported as 100.0% for multiple times from several weather stations. The summation of these individual event detection counts is equal to the total event count.

4.3 Identifying suspicious pixels of radar images

In order to identify suspicious bins of the radar image, it compared each element of the array with a threshold value. It was needed to process the matrix in a sub-function since the Siddhi CEP engine does not support direct array manipulation functionalities.

A sample of 1,008 radar images [15] had been used in the image-based weather detection. These samples contained precipitation pixels; therefore, the total number of detections was 1,008. It took 7,092 ms seconds to match these input events with the query.

4.4 Building queries with multiple meteorological variables

This use case was verified using the wind speed and relative humidity data. This winter storm was pounded with heavy snow, high winds, and coastal flooding. The system identified that most of the weather stations in the Massachusetts area were reporting high wind conditions with a relative humidity as 100% on January 27, 2015. Table 5 contains the test results.

Table 5. Test results

Time (UTC)	*Weather Station*	*Wind Speed*	*Relative Humidity*
13:56	KHYA	16.4622	100
9:52	KCQX	13.3755	100

4.5 Identifying Weather Stations with defects and suggest alternative values

The MADIS dataset had missing values for some of the meteorological variables. The system considered two nearby weather stations: FSKM3 (42.109, -72.124) and AR824 (42.130, -72.098) (See Figure 4). FSKM3 contained missing values of relative humidity on January 27; whereas AR824 reported sensor readings throughout the same day. Therefore, this scenario was applied to find alternative values for missing relative humidity values of FSKM3 from AR824 weather station. Table 6 contains the test results. Further, there can be a technical fault at FSKM3.

Figure 4. Nearby weather stations

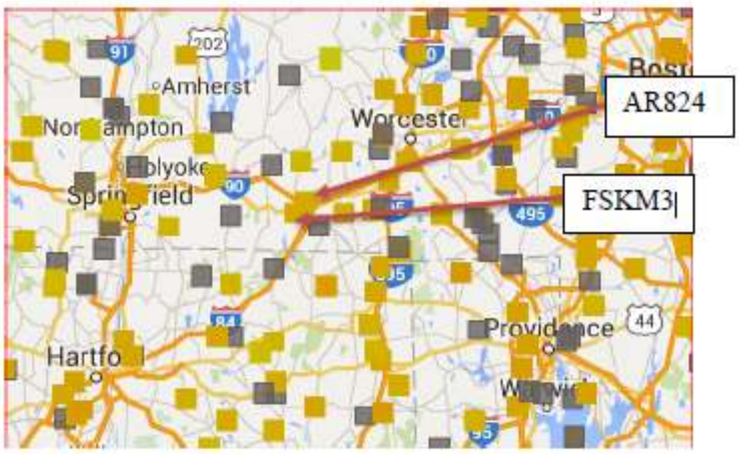


Table 6. Test results

Time	*FSKM3*	*AR824*	*Expected Value*	*Actual Result*
t0.40	-99999	51	51	51
2.00	-99999	59	59	59
2.05	-99999	56	56	56

4.6 Identifying anomalies in weather data

Detected high wind and heavy snow conditions can be verified with nearby weather station information. For this use case also, we considered the weather stations from Massachusetts. Longitude and latitude are helpful in tracking the locations of the weather stations and are used to find nearby weather stations and their readings at the same time.

The weather station FSKM3 had reported the temperature as 262K at around 13:30 PM on January 26, 2015. This use case was tested by comparing nearby weather station values. Therefore, the result was that AR824 also had reported the same value for the temperature at that time. Table 7 contains the test results of the weather station AR824.

Table 7. Test results

Time (UTC)	*Temperature (K)*
12:45	262.0389
14:00	262.5945
14:55	263.15

4.7 Identifying nearby weather situation of a given location

If a user wants to find out the current weather situation of Massachusetts as of January 27, he/she can query the weather data given the location information. When a user searches for the temperature values of the specific weather stations, the user can get the relevant values as in Table 8. Further, they can find out that the KHYA weather station has issued high wind warnings at 13.56PM (UTC) (see Table 5).

Table 8. Test results

Time (UTC)	*Weather Station*	*Reported Value (K)*	*Result*
6:58	C5897	261.483	261.483
6:55	D9545	263.15	263.15
6:00	FSKM3	264.15	264.15
6:00	VTDOT	261.26	261.26

5. Conclusion

5.1 Summary

The novel contribution of this thesis is to provide the monitoring phase capabilities to a typical weather/climate observatory (see Figure 1) using the idea of Complex Event Processing. The idea of Complex Event Processing is relatively simple to use. It adds agility to a weather detection system by allowing it to detect primitive weather events and anomalies by simply writing a SQL-like query, add dynamic queries on the fly, and enables scalability in terms of number of methodological variables, weather stations, and queries. Moreover, such a system can scale to high arrival rates of sensor readings.

This system was divided into two parts considering the input data: surface observational data, and radar image data. The basic weather detection scenarios were created based on these input event types. It presented several use cases and demonstrated how to apply Siddhi CEP engine to solve these use cases. It compared meteorological variables with predefined thresholds to identify impending weather events, identify weather stations with defects and suggest alternative values, identify anomalies in sensor data and identify weather situations around a given location. These use cases address the basic needs of a typical weather monitoring center, as it provides basic real-time weather monitoring and detection functionalities.

The performance and accuracy of the proposed weather anomalies and events detection system were confirmed using a recent weather incident (winter storm "Juno"). The system was able to match the input events with the predefined queries successfully. These use cases worked effectively with the existing performance of Siddhi, and the system was able to handle more than 10,000 events per second.

5.2 Problems encountered

It was necessary to process radar images since they were used in several weather detection algorithms. There were no functions to manipulate images in Siddhi thus the system converted these images into 2D arrays and used Siddhi extensions to process arrays. Currently, Siddhi does not support inbuilt array manipulation functions such as "get the maximum value of the array" and "get number of rows/columns" so the system had to use traditional array manipulation functions to process these 2D arrays.

This system can be used as an early weather monitoring system. In order to confirm these conditions further, it requires using complex and resource-consuming weather detection algorithms. Furthermore, we faced several limitations when trying to use several kinds of weather data together. For example, we found lightning data from WSI weather data, but we could not find temperature information at the same locations.

5.3 Future work

Meteorology is a vast research area. This system has implemented four key use cases to achieve the main goal of the study, but more use cases can be defined for further verifications.

To better support weather data analysis, CEP engines require several improvements. Currently, Siddhi CEP supports type object in stream definitions enabling the ability to handle any type of object. It is necessary to improve object type into specialization like arrays and implement inbuilt array manipulation functionalities within Siddhi. Then Siddhi can provide custom functions for array operations such as getArrayElement (array, index), min, max, and hasValueGreaterThan(). Furthermore, the images are required to be converted to arrays before applying with Siddhi. Therefore, it is necessary to add image processing features to Siddhi CEP engine.

References

[1]. K. K. Droegemeier et al. Linked environment for atmospheric discovery (LEAD): A cyber infrastructure for mesoscale meteorology research and education. In *Proc. 20th Conf. on Interactive Information Processing Systems for Meteorology, Oceanography and Hydrology.* Seattle. Jan. 2004.

[2]. Y. Sokha, K. Jeong, J. Jonghyun, and W. Joe. A complex event processing system approach to real-time road traffic event detection. *Journal of Convergence Information Technology (JCIT),* 8(15), Oct. 2013.

[3] Wikipedia contributors. (2015, Sep). Weather [Online]. Available: https://en.wikipedia.org/wiki/Weather.

[4]. *Weather applications - weather detection* [Online]. Available: http://fresno.ts.odu.edu/newitsd/ITS_Serv_Tech/weather_app/ weather_applications_Weather_Detection.html.

[5]. *Tornado Detection* [Online]. Available: https://www.nssl.noaa.gov/education/svrwx101/tornadoes/detection/.

[6]. National Weather Service. (2011, Nov). *WFO non-precipitation weather products specification [PDF]*. Available : http://www.nws.noaa.gov/directives/sym/pd01005015curr.pdf.

[7]. *WSO2 Complex Event Processor* [Online]. Available: http://wso2.com/products/complex-event-processor/.

[8]. F. Lalaurette. Early detection of abnormal weather conditions using a probabilistic extreme forecast index. *Quarterly Journal of the Royal Meteorological Society, 129,* pp3037- 3057, Mar. 2006.

[9]. H. M. N. D. Bandara and A. P. Jayasumana. Distributed, multi-user, multi-application, and multi-sensor data fusion over named data networks. *Computer Networks,* 57(16), pp3235-3248, Nov. 2013.

[10]. H. R. Glahn and D. A. Lowry. The use of Model Output Statistics (MOS) in objective weather forecasting. *Journal of Applied Meteorology,* (11), pp1203-1211, 1972

[11]. M. Hayati and Z. Mohebi. Temperature forecasting based on neural network approach. *World Applied Sciences Journal,* 2(6), pp613- 620, 2007.

[12]. National Oceanic and Atmospheric Administration. (2010. *Meteorological Assimilation Data Ingest System* [Online]. Available : http://madis.noaa.gov/.

[13]. Blue, M. M. (2007). *Excel formula to calculate distance between 2 latitude, longitude (lat/lon) points (GPS positions)* [Online]. Available: http://bluemm.blogspot.com/2007/01/excel-formula-to-calculate-distance.html.

[14]. L. Lam, et al. (2015). *Winter storm Juno hammering New England* [Online]. Available:http://www.weather.com/storms/winter/news/winter-storm-juno-blizzard-boston-nyc-new-england.

[15]. [Online]. Available: https://dl.dropbox.com/u/37693320/VisualizationExercise1/Virring.zip.