DETECTION OF WEATHER ANOMALIES AND EVENTS OF INTEREST USING COMPLEX EVENT PROCESSING

K. A. G. Udeshani

138239G

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

October 2015

DETECTION OF WEATHER ANOMALIES AND EVENTS OF INTEREST USING COMPLEX EVENT PROCESSING

K. A. G. Udeshani

138239G

Dissertation submitted in partial fulfilment of the requirements for the degree Master of Science in Computer Science specializing in Software Architecture

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

October 2015

DECLARATION OF THE CANDIDATE & SUPERVISOR

I declare that this is my own work and this thesis/dissertation does not incorporate any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

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

……………………………………… ………………………………………..

K. A. G. Udeshani Date

The above candidate has carried out research for the Masters Dissertation under my supervision.

………………………………………… ………………………………………..

Dr. H.M.N. Dilum Bandara Date

ACKNOWLEDGEMENTS

I would like to express deep gratitude to my supervisor, Dr. H. M. N. Dilum Bandara, for his valuable support, encouragement, supervision and useful suggestions throughout this research work. His continued guidance enabled me to complete my work successfully.

Finally, I wish to express my gratitude to my family and friends for the given ideas and support to complete the work successfully.

ABSTRACT

Many natural disasters which occurred recently emphasize the importance of a system which can be used to identify changes in weather conditions. Consequently, several public organizations in Sri Lanka are planning to establish a Climate Observatory system based on open-source technologies. This research develops a Complex Event Processing based system to detect weather anomalies and events of interest to enrich a Climate Observatory system with real-time monitoring and detection capabilities.

Event Processing is, tracking and processing streams of data about the events that happen in the physical environment. Complex Event Processing (CEP) combines data from several sources to infer events and complex patterns among events in real-time. One of the key characteristic of the weather data analysis and detection is that it needs to deal with real-time data that are generated by a multitude of sensors. While several techniques are used to detect weather anomalies and events, CEP is a more suitable approach, as it is the principle technology for real-time moving data processing.

One of the objectives of this research is to demonstrate how CEP can be applied in weather anomalies and events detection. This research presents a novel weather anomalies and events detection solution based on Siddhi complex event processor. Siddhi provides the runtime to perform CEP, which is able to identify meaningful patterns, relationships and data abstractions from unrelated events and alert users about the detected patterns.

This study first analyzes the features of the existing weather detection systems and identifies the significant meteorological variables, weather sensors and use cases. The proposed solution modifies the input data before processing with Siddhi. This pre-processing step consists of two sub-systems. One sub-system pre-processes the weather sensor data. The other sub-system is used to convert radar images to 2D matrixes since Siddhi is unable to process stream of images. It applies Siddhi CEP engine for weather detection with appropriate stream definitions and query definitions. Partitions are used to process queries to gain better performance.

The proposed solution focuses on four use cases. First use case compares meteorological variables (i.e., sensor data) to predefined thresholds to identify impeding weather events. Second use case identifies weather stations with defects and suggests alternative values to replace them. Remaining two use cases find anomalies in the sensed data and identifies weather situations around a given location. These use cases provides the basic capabilities of a typical weather monitoring center such as the proposed Climate Observatory system for Sri Lanka, as they provide elementary real time, weather monitoring and detection functionalities.

The performance evaluation shows that Siddhi performs well in the specified use cases even with high input rates and large number of meteorological variables. It further identifies future enhancements to the system. The research also identified several limitations in CEP engines, particularly Siddhi, while applying them to weather anomalies and events detection.

Keywords: Complex Event Processing, Siddhi, Weather Anomalies and Events Detection, Weather Alerts, Pre-processing, Climate Observatory System, Radar, Geo-dashboard

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
APA	Action Planning Agent
ARPS	Advanced Regional Prediction System
CASA	Collaborative Adaptive Sensing of the Atmosphere
CEP	Complex Event Processing
COSTI	Coordinating Secretariat for Science Technology & Innovation
DA	Diagnosis Agent
DCAS	Distributed Collaborative Adaptive Sensing
DR	Demand Response
EPA	Event Processing Agents
ECA	Event Condition Action
EDA	Event Driven Architecture
EPN	Event Processing Network
GDS	Grid Data Service
HI	Heat Index
LDM	Local Data Manager
LEAD	Linked Environments for Atmospheric Discovery
LEAD – CI	Linked Environments for Atmospheric Discovery Cyber infrastructure
MADIS	Meteorological Assimilation Data 37 Ingest System
MDA	Mesocyclone Detection Algorithm
MOS	Model Output Statistics
NSSL	The National Severe Storm Laboratory
NWS	National Weather Service
RT	Reflective Threshold
SCIT	Storm Cell Identification and Tracking

SOA	Service Oriented Architecture
SPA	Sensor Processing Agent
SQL	Structured Query Language
SSN	Semantic Sensor Network
VCP	Volume Coverage Patterns
WRF	Weather Research Forecast
WDSS-II	The Warning Decision Support System – Integrated Information
XML	Extensible Markup Language

1. INTRODUCTION

Throughout the year we hear about the sudden changes in weather. These sudden changes cause many disasters. Sometimes fishermen were unable to return home due to strong winds and innocent lives are often lost from lightning and floods. These incidents also cause economic losses to the country. Therefore, everybody in the country is in need of an effective solution for getting informed about the sudden changes in weather.

People are interested in knowing the current state of the atmosphere of a given location and time, and they use several meteorological variables to measure the state. Typical meteorological variables for weather monitoring include temperature, air pressure, humidity, wind speed, and wind direction. Several mechanisms are used to measure these variables. For example, a rain gauge is used to measure the rainfall and a hydrometer is used to measure humidity. These meteorological variables are useful in determining both short and long term changes in the atmosphere. When these meteorological variables are used to determine atmospheric conditions at a specific place at a specific point in time (ranging from minutes to weeks), it is called *weather analysis*. When their statistical properties are used to measure long-term (ranging from months to decades) changes in atmospheric conditions, it is called *climate analysis*. Detection of weather conditions is of prime importance, as it directly affects day-to-day life. However, this is quite challenging as it requires real-time data gathering, transmission, and processing. Further it is important to identify anomalies in the sensor data reported values of the weather stations. Typically these anomalies can be either missing or incorrect sensor readings.

Complex Event Processing (CEP) based systems receive events from multiple independent simple event streams of different event sources. Complex event detection (aka. event pattern matching) is the core functionality of such a system. The user needs to provide event pattern rules in order to detect specific events. These event pattern rules can be defined in a SQL-like event processing language [1]. 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 analyse and detect thousands of events per second. Therefore, CEP technology derives intelligence from real-time event data analysis. The ability to analyse large streams of incoming events in real-time and detect relevant events makes them a suitable alternative for modern weather detection.

1.1. Motivation

The weather is vastly becoming integrated with the Sri Lankan life style, thus the ability to detect severe weather events and issue relevant notifications is very much important. There were several incidents where people faced difficulties with sudden changes of weather. As a result, the need for an accurate, trustworthy and real-time weather detection system is highly important.

To address these emerging needs, there is an interest to develop a National Climate Observatory System for Sri Lanka under the Coordinating Secretariat for Science Technology & Innovation (COSTI) initiative. The information available through this Climate Observatory will be available in appropriate forms to the public. The Climate Observatory is to be built primarily based on open-source technologies with the participation of a number of partner organizations [2]. As part of this initiative relevant stakeholders are planning to build an archive of Meteorological variables collected from many existing and new weather stations that are (to be) positioned across the country. As these meteorological variables are collected in real-time, it also provides an opportunity to perform real-time weather detection. Moreover, such early detection can be used to trigger more complicated weather algorithms that are required for more accurate detection, better forecasting and warning of complicated weather events. Therefore, it is important to be able to develop a solution to detect changing weather conditions in real-time, and be able to alert the relevant stakeholders on time. However, the proposed solution should be built on proven open source technologies to reduce the development time and cost, while increasing the accuracy and reliability.

Figure 1.1 illustrates the workflow of a typical Climate/Weather Observatory. Meteorological variables are collected by sensors attached to the weather stations and the collected data are transmitted to the Observatory using a suitable network, e.g., leased line, 3G/4G, or a satellite link.

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 tornados. Therefore, when detection of an abnormal weather condition is triggered, more complicated weather detection algorithms are run to determine further details of the event. For example, high wind conditions can lead to a tornado. So, if high wind conditions are identified during the monitoring phase, a tornado detection algorithm is executed for further confirmation. Therefore, while the monitoring phase can issue weather alerts in real-time, it is also used as the initial decision maker to initiate more complicated and resource consuming (e.g., need more computing power, memory and storage) weather detection algorithms during the third phase.

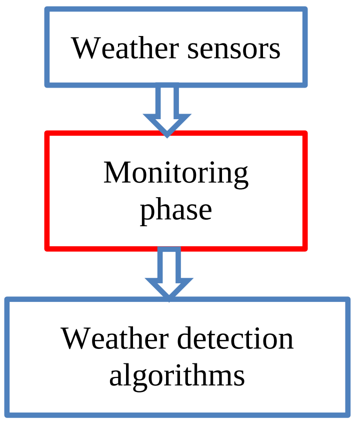


Figure 1.1 Workflow of a typical climate/weather observatory.

1.2. Problem Statement

The main objective of this research is to provide the monitoring capabilities (as seen in Figure 1.1) to a typical Climate/Weather Observatory. This is to be achieved by developing a complex event processing based weather anomalies and events detection system that is scalable in terms of the 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 should also be capable of providing solutions for common use cases found in weather detection and warning.

1.3. Research Contributions

This thesis demonstrates the application of Complex Event Processing (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 high 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 Siddhi CEP engine. The users of the system can specify necessary queries in order to identify uncertain changes in meteorological variables such as temperature, wind speed, pressure and humidity. 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 those as notifications to the relevant sub-systems (i.e., third phase of a typical Climate/Weather 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.

1.4. Outline

Chapter 2 presents the literature review. It discusses about CEP, existing weather detection systems and meteorological variables. Chapter 3 presents the research methodology. It presents the high-level architecture, use cases and the details of the proposed system. Chapter 4 presents the performance analysis and evaluation. Finally, concluding remarks, problems encountered and the future work are discussed in Chapter 5.

2. LITERATURE REVIEW

2.1. Complex Event Processing

Tracking and processing streams of information about the things that happen is called event processing. In Complex Event Processing (CEP), it combines data from several sources to infer events or patterns. The goal is to recognize meaningful events and provide a response to them with a minimum latency. Figure 2.1 shows the high-level architecture of the event-based applications where it receives events as an input stream, it processes them and generates an output stream. For example, assume the system receives events from temperature and smoke sensors. There can be predefined event patterns in order to identify a fire in advance so the system can alert necessary places to avoid difficult situations. CEP identifies complex patterns of unrelated events, event correlation and abstraction, event hierarchies and relationships between events. These relationships can be categorized as causality, membership, timing and event driven processes [3].

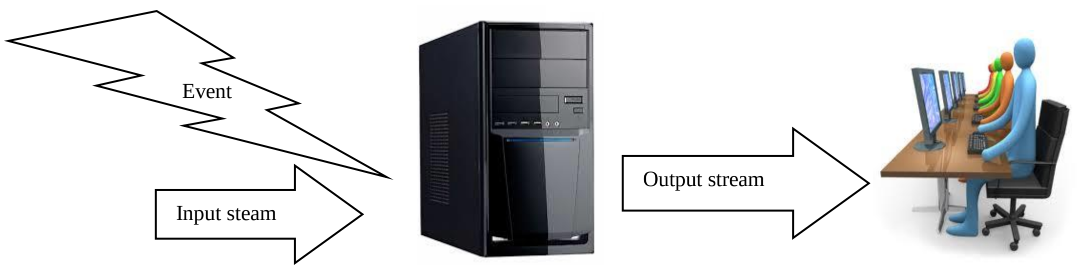


Figure 2.1 Event-based applications [4].

Figure 2.2 illustrates a high-level overview about event processing. There are several terms, which are required to know in order to understand the concept behind the CEP. Events, Event Processing Agent (EPA) and Event Condition Action (ECA) are two of them. Events can be real world or virtual. Event processing is applied to the subject system. These events can be simple events or complex events; complex events are combinations of simple events. The events can be arranged as event stream or event clouds. CEP deals with event clouds, but the event processing engine is important in both types of events. EPA filters the events and provides them to the event processing engine. The pattern/rule is defined using an SQL-like event query language. This can be categorized into three styles: composition operators, data stream query languages and production rules. Composition operators are conjunction, sequence and negation. These composition operators and nesting of expressions are used to compose single events and express complex events. Data stream query languages are based on SQL. Production rules specify actions to be taken when certain event patterns are matched. Event processing engine processes the events and notifies the users. ECA defines the actions to be taken automatically on the subject system, if the conditions are satisfied. The derived events and composite events are generated at the processing time [5].

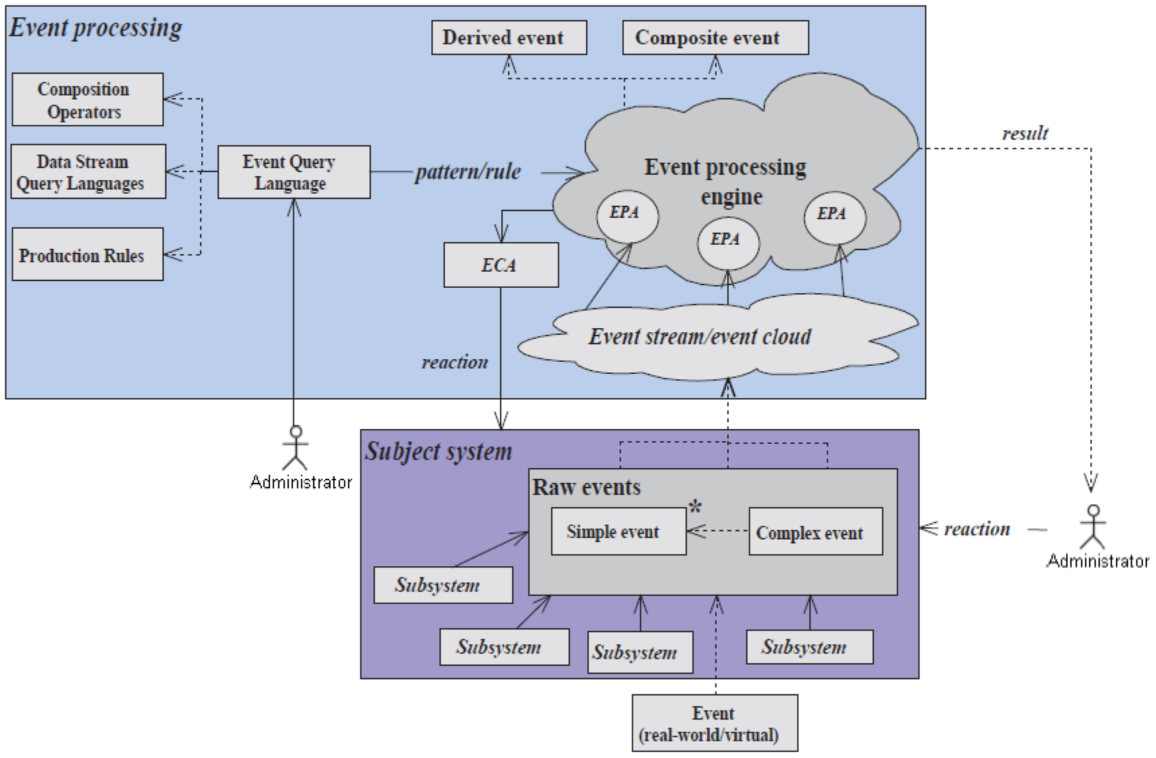


Figure 2.2 Overview of CEP [5].

Event

An event is a message or a multiple data component, which is used to define activities happening or have just happened in the physical environment [5].

Event Pattern

Event pattern defines the relationship between events. It can be a temporal relationship (e.g., “A happens before B”), causal relationship (e.g., “B happens because of A happened”), independent relationship (e.g., “there is no relationship between A and B”) or an aggregative relationship (e.g., “when events A happen that means event B also happened”) [3].

Logical operators (conjunction, disjunction or negation) and set operators (union, disjoint) are used to build an event pattern. Queries are used to define these event patterns by an Event Processing Language [3] which can be expressed as ECA (event condition rules) or as SQL-like continuous queries [6].

Time Window

The time window is used to identify the absence of events. The width of the window is specified using the number of events or the time period of an event(s). When width of window is specified in terms of the number of events it is referred to as *sliding window*. When time period of events are specified it is referred to as *time window*. In sliding window the window will gradually moves with the event notifications and the time window will process events by moving the window in event blocks [7]. Sliding windows are used to handle infinite data streams. It defines a lease time for events to consider the most recent set of events [6]. The time information of an event has three types, event occurred time, detection time and the processed time [5].

Event Pattern Rules

CEP defines correlation between events by identifying patterns. The rules are expressed by event processing languages based on event algebras. Event pattern rules define the event pattern and the corresponding actions. Event patterns specify certain situations of events and event actions are executed when the event pattern is fulfilled [6], [8], [9], [10].

The incoming events are processed with three different event pattern rules as shown in Figure 2.3. Filtering will reduce the incoming events with a guarded pattern. It filters data to detect specific conditions using simple or complex filters. First, it blocks C, E, F, H and I events since guarded pattern has defined to do so. The aggregating phase will create a combination of events from multiple sources. Further, it groups and aggregates data to produce high-level statistics and computes new data elements or transforms the data format and structure of the events. There are two aggregators they combine A, B, D and G, J events separately. Finally, there is the detecting phase of the event patterns. The unusual situations of events are caught by the detectors and they raise alerts [3]. In order to apply CEP, these functionalities are applied in a proper manner in real-time data analysis.

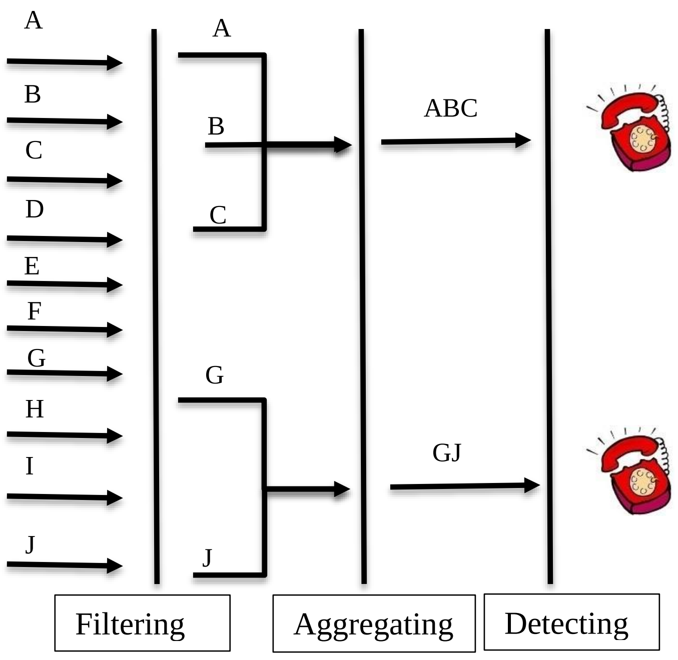


Figure 2.3 Flow of events [3].

2.2. Siddhi CEP Engine

CEP is used to detect complex conditions from specific set of low-level factors. There are several CEP engines in the market. 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. It allows initiating multiple instances and each Siddhi engine is single threaded. This CEP engine supports partitioning which allows the users to isolate the processing into small parts and speeds up the execution. Figure 2.4 shows the high-level architecture of Siddhi CEP [11].

Siddhi Architecture

Siddhi receives events through the input adapters and converts them to a common data model called tuples. The query is converted into a runtime version and deployed in Siddhi core where all the processing is done. Input events are placed in the input queues for processing; the input events which are matched with the input queries are placed in the output queues [12].

Siddhi core contains processors; the main components of processors are executors and event generators. Executors are generated by the query parser and express the query conditions to do the evaluation. These executors are arranged as a tree structure and evaluates in depth first search order. There can be many executors in a processor but only one gets executed at a time [12]

Siddhi uses a pipeline architecture where it breaks the execution into different stages using processors. It uses a publication-subscription model to move the data through this pipeline. Siddhi can implement multiple streams using a single input event queue by multiplexing them. This has improved Siddhi's performance since it does not need to monitor the intermediate events [12].

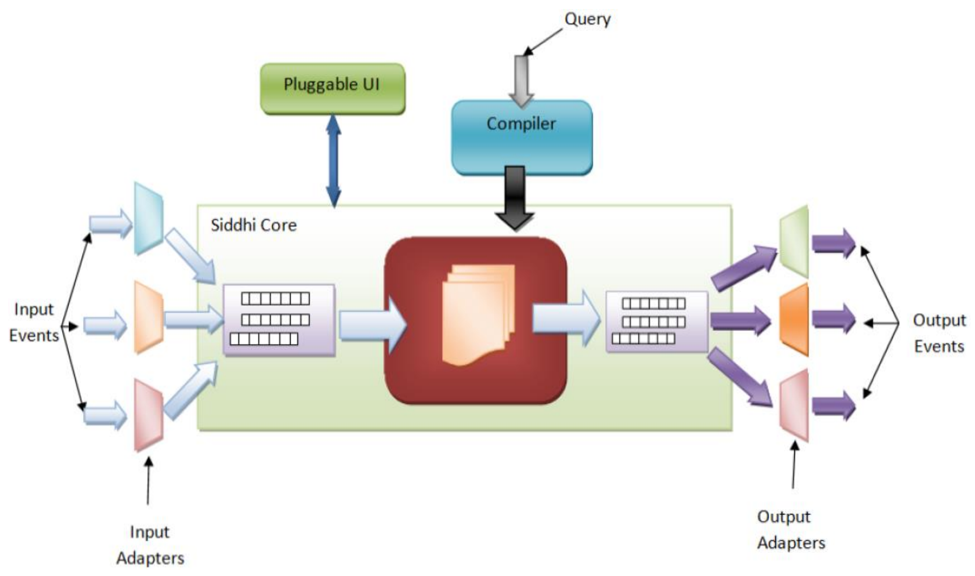


Figure 2.4 Siddhi high-level architecture [12].

These events consist of (name, value) pairs. Siddhi uses tuple data to represent events. Figure 2.5 shows a sample tuple. User has to define streams and data which belong to those streams. Users use event query language, similar to SQL, to define the queries to be applied on the incoming events [12].

`Stream id`	`Data 1`	`Data 2`	`Data 3`

Figure 2.5 Example tuple.

Siddhi Query Processing

Siddhi supports four types of queries: filtering, event windows, ordering (sequences and event patterns), (aggregation, join and split) events. These different query notations can be used to define required queries. Filter queries are used to filter the events by different conditions (>, <, =, <=, >=, !=, contains, and, or, not). Pattern queries identify event patterns like A, B, C. An alert is triggered when it receives these events with the given order. It is possible to have different events in between the given event pattern. "*Every*" operator is used in Siddhi to continuously monitor for a given pattern. The processing of these pattern queries is handled by the Pattern Processor [12].

Sequence queries are defined to fire an event when series of conditions happened one after another in a consecutive manner. Sequence processors execute the sequence queries. Currently active Executors receive the input event and check whether it can be matched with the specified conditions. If they are matched, it will send "true" and the Processor spawns a new Executor for the next state. When it is the last state the Processor will generate the output event based on the output definition. Siddhi sends the input event only to the corresponding Executor and further Siddhi eliminates duplicate states. These strategies help to improve the performance of Siddhi [12].

Siddhi supports sliding-window and batch-window based queries. These queries can be divided into time-based and length-based event queries. Sliding-window based queries keep track of the events received within a given amount of time from the current time. It helps to consider events within a limited amount of time. The length-sliding-windows keeps track of a specific number of events arrived recently. The batch-windows are similar to the sliding-windows except they perform processing as event batches. Siddhi implements windows within event queues by assigning a time window to the stream [12].

Siddhi supports avg., sum, count, max, min functions in aggregation queries. Aggregation processor will apply these on the collection of events through the window queue. In order to implement join queries it moves data from several incoming event queues to a different queue after joining (outgoing event queue). Joining processor will check whether the input events are matched with the join condition, if so send them to the next queue else keeps them to match with the future events [4].

Figure 2.6 illustrates a sample query. It joins two sensors, Sensor A and Sensor B considering the unusual situations of temperature and wind speed. Here the events arriving only to the unidirectional stream trigger the join.

```
From SensorA as b join
      SensorB p unidirectional
on temperature>40 and windSpeed>50
      select SensorId, LocationId
insert into suspectStream
```

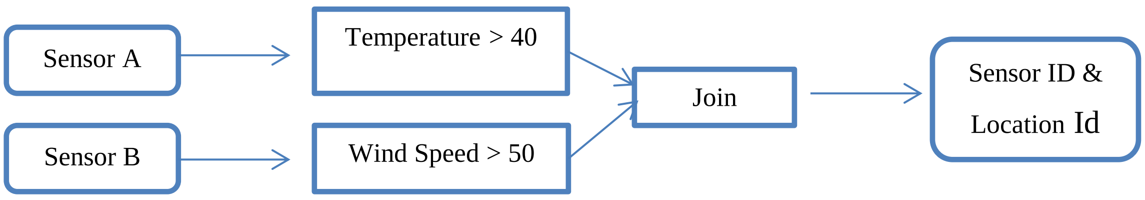


Figure 2.6	Sample query.

Applications of Siddhi

Los Angeles Smart Grid Demonstration Project uses Siddhi. It forecasts electricity demand, respond to peak load events, and help to improve the sustainable use of energy [4].

“Grand challenge” solution is explained in [13]. It presents how to map grand challenge queries into Siddhi event processing language. It further explains about implementing queries using the engine. This system deals with real-time data collected from sensors worn by players in a football game. It analyses the running speed of players, calculates the ball possession, calculates the activity in different regions of the ground, detects the hits to goal and provides running updates.

Figure 2.7 shows that if the system needs to match the sequence Q1, Q2+, Q1 where Q1 and Q2 are conditions. The sequence matches when the condition Q1 is followed by several occurrences of Q2 and another occurrence of Q1. It will listen to the Q1 when an event satisfies it, the system will create a new instance and start to listen to Q2. Likewise when it received the first event it creates a copy of the state machine and waits for a matching event. These sequences use regular expressions such as “+” and “*” [13].

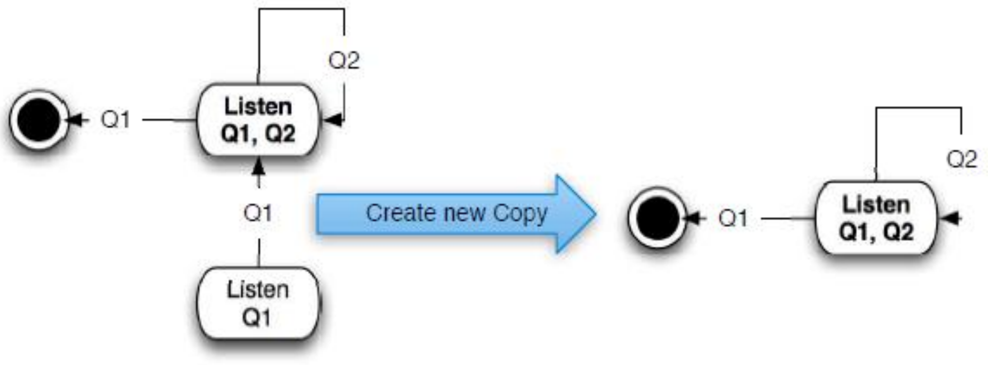


Figure 2.7 Sequence state machine [13].

2.3. The Linked Environments for Atmospheric Discovery (LEAD)

The Linked Environments for Atmospheric Discovery (LEAD) [14] is used by the meteorological higher education and research communities like US National Weather

Service. One of its objectives is to improve our understanding and ability to detect and analyze the mesoscale atmospheric situations. LEAD allows users to query the observational data, simulate and predict using numerical atmospheric models like the Weather Research Forecast (WRF), adjust data by combining observation. Further, it enables to analyze and mine the observational data and identify the relationships among them. Finally, visualize and evaluate the data and model the output using tools. The web services framework in LEAD supports automatic configuration, dynamic responses, automatic initiation of the processes and optimization to the data collection which are required with remote observing technologies. LEAD supports different kind of tools in order to achieve these capabilities using the service oriented architecture [15]. For example, when someone wants to understand why some of the severe thunderstorms produce a succession of mesocyclones and multiple tornadoes, user needs to follow some steps. First, the web-based LEAD portal will allow accessing the required Doppler radar data. It needs to filter the data where the thunderstorms were present and process them. The user can apply a data mining engine to these data sets to identify all cyclic versus non-cyclic storms and the existence of tornadoes with the surrounding environmental conditions. People can use LEAD algorithms and models to examine the probability of cyclic storm behaviour [14].

LEAD Architecture

Complex event processing enables real-time response to the weather in LEAD and plays a key role. LEAD uses the Calder as the SQL-based event processing system. LEAD uses SOA concepts at both the application and middleware level. The distributed SOA provides a secure access to complex weather forecasting models for meteorology researchers and students. Scientists can set up a weather forecast over a region and submit a workflow that will run in the future, where a SQL-based CEP query detects severe storm conditions. Vijayakumar and Plale [16] further explain about the information model for the provenance service. It contains streams and queries as its primary entities. The streams can be base streams or derived streams.

The information model shown in Figure 2.8 has six main entities. It shows different kinds of relationships among them. It shows how to append the storm information to the provenance model when someone wants to enable the prediction for different stormy conditions. It considers the size (radius) and the location (latitude and the longitude of the centre) to establish the correlation between the storm and the queries [16].

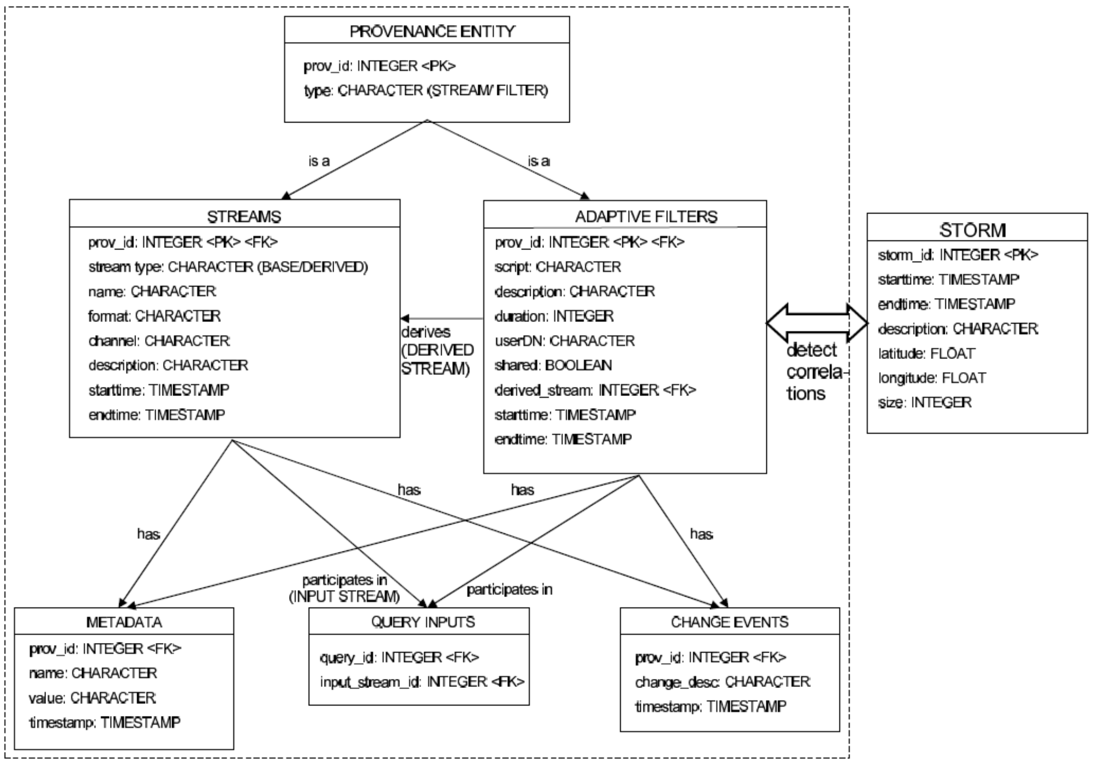


Figure 2.8 Provenance information model [16].

2.4. LEAD Cyber Infrastructure Model

One of the key research objectives of LEAD project is to present advances in cyber infrastructure (LEAD-CI) for meteorology research and education [17]. The LEAD-CI model presents a model for bridging between the physical environment and e-Science workflow systems through event processing systems.

The main purpose of LEAD-CI is to address the meteorology research challenges and process meteorological data and model output independent of format and physical location. It proposes a model to connect the physical environment and e-

Science workflow systems. Further, it proposes efficient stream mining algorithms [17].

The temporal component of severe weather events such as severe storms and tornadoes was ignored. This assumption simplifies the mining algorithms because no need to track each and every state of the weather events over time. LEAD-CI makes events processing just another web service to connect the real-time observational data in to a SOA. Figure 2.9 represents the key components of the LEAD-CI architecture. The Calder stream service is the event processing component of the LEAD-CI project [17].

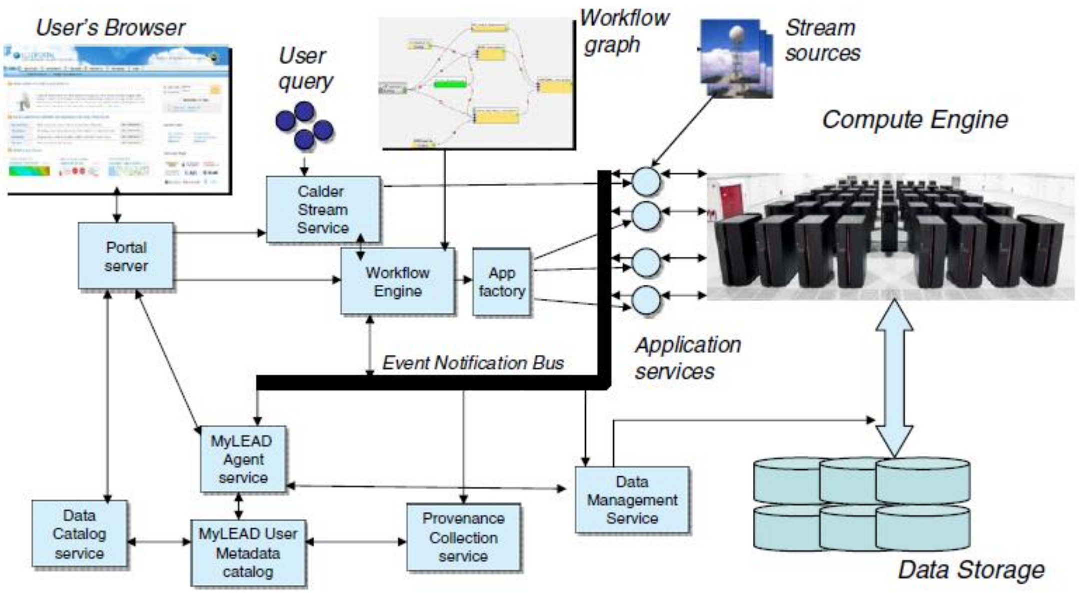


Figure 2.9 Key components of the LEAD-CI architecture [17] [18].

2.5. Collaborative Adaptive Sensing of the Atmosphere (CASA)

CASA is a Distributed Collaborative Adaptive Sensing (DCAS) system [19]. It uses the DCAS architecture to detect and predict hazardous weather using a dense network of short ranged and low powered radars [19]. CASA consists of a heterogeneous set of radars and small sensors such as pressure sensors, rain gauges and micro weather stations. CASA supports several applications as shown in Table 2.1. These applications use one or more data type from one or more radars [20].

Table 2.1 CASA applications [20].

Application	Description	No of Radars	Data types
Reflectivity	Reflectivity of clouds	1	Reflectivity
Velocity	Wind velocity	2-3	Doppler velocity, reflectivity
Network-based Reflectivity Retrieval (NBRR)	Reflectivity of clouds detected using multiple radars	3+	Reflectivity
Nowcasting	Short term forecasts of active weather events	1-3	Reflectivity
Quantitative Precipitation Estimation (QPE)	Estimating current precipitation using the intensity of rain and water droplet size	1-3	Reflectivity, differential phase, correlation coefficient
Tornado Tracking	Detect and track a tornado as it forms and moves	2+	Doppler velocity, reflectivity

Researchers can tell whether a tornado is likely to form using the measurements such as the atmospheric stability, temperature and humidity such that high instability and high humidity can lead to a tornado. CASA uses event-specific queries in a specific area of interest; for example, a location with rotating wind can lead to a tornado [20].

The CASA DCAS system includes radars and algorithms which are used for weather detection and user interfaces. IP1 is a prototype of this system which is located in Southwestern Oklahoma. The IP1's goal is to detect a tornado within 60 seconds and track their centroids. Reflective Threshold (RT) and Storm Cell Identification and Tracking (SCIT) are the detection algorithms which are used in this project. These algorithms are used to extract meteorological features of radar data [19].

2.6. CASA and LEAD.

CASA and LEAD are two complementary projects. They are used together to develop a hardware and software framework which enables real-time multiscale forecasting. It allows meteorologists to directly interact with instruments. Dynamic workflow adaptivity, dynamic resource allocation, continuous feature detection and data mining and model adaptivity are the main goals of this project [18].

Figure 2.10 explains the interaction between CASA and LEAD. CASA links radars with meteorological command and control (MC&C) module. LEAD contains a modelling loop and it executes forecast models and responses to weather conditions. The data storage tools are required to automate data staging and data collection. Monitoring tools are used to enhance the reliability and fault tolerance [18].

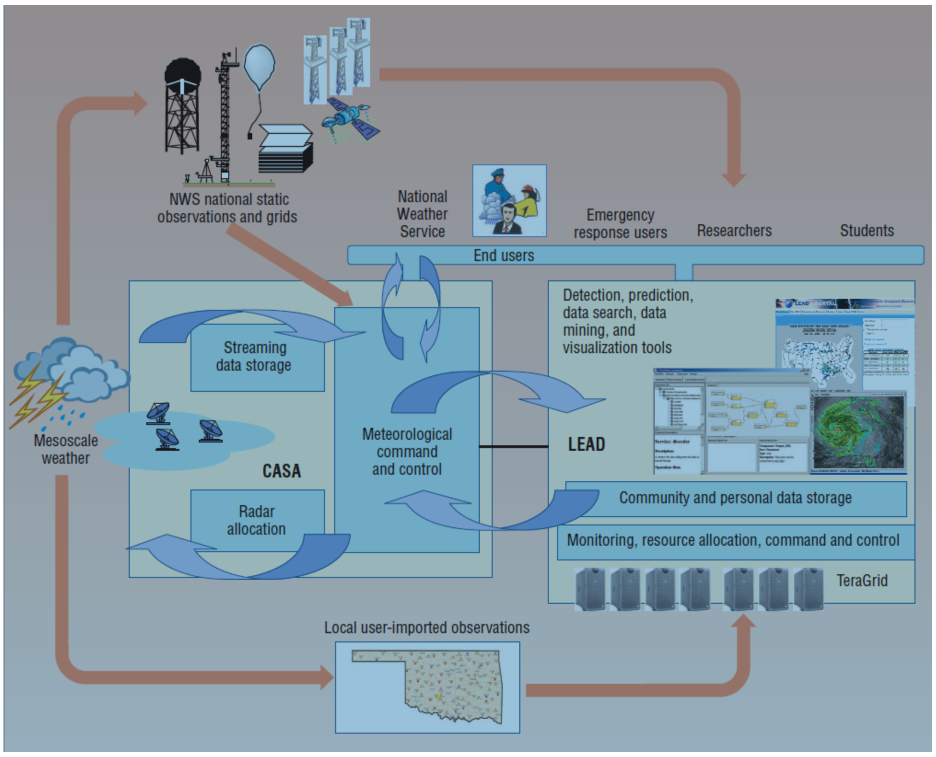

NWS national static observations and grids
National Weather Service
Emergency response users
Researchers
Students
End users
Streaming data storage
Detection, prediction, data search, data mining, and visualization tools
Mesoscale weather
CASA
Meteorological command and control
LEAD
Radar allocation
Community and personal data storage
Monitoring, resource allocation, command and control
TeraGrid
Local user-imported observations

Figure 2.10 CASA and LEAD interaction [18].

This system has several features such as distributed, collaborative and adaptive (DCAS). Distributed refers to the use of many small and inexpensive radars. Collaborative is the coordination of beams from multiple radars to achieve greater sensitivity. Adaptive refers the ability to dynamically reconfigure the radars [18].

2.7. Warning Decision Support System – Integrated Information (WDSS-II)

This system is developed to test newly developed severe weather detection algorithms. It uses data from WSR-88D radars. There are four main categories of WDSS-II software components. First component reads the available data streams and prepare them for further use. The second component allows developers to access and manipulate the various data through an Application Program Interface (API). The third component contains meteorological algorithms and applications to analyze the data and provide information for forecasters. The final component can be used to view raw input data and algorithm output [21].

WDSS-II supports severe weather forecaster requirements and provides tools for the analysis and diagnosis of several conditions such as rotation, hail, wind speed, lightning and precipitation. It provides automated algorithms that operate on data from multiple radars [21].

Figure 2.11 shows the creation of diagnostic products using automated algorithm in real-time. The ellipses show the real-time applications and rectangles shows the diagnostics products which can be used in weather analyses. Level II is the high resolution Doppler radar data. National Lightning Detection Network (NLDN) provides lighting flash data. The incoming data from these sources are process as NetCDF or XML files to create the diagnostics products using some automated algorithms. The CASA project uses WDSS-II software’s linear buffer publish/subscribe mechanism to distribute radar data among various feature detection [22].

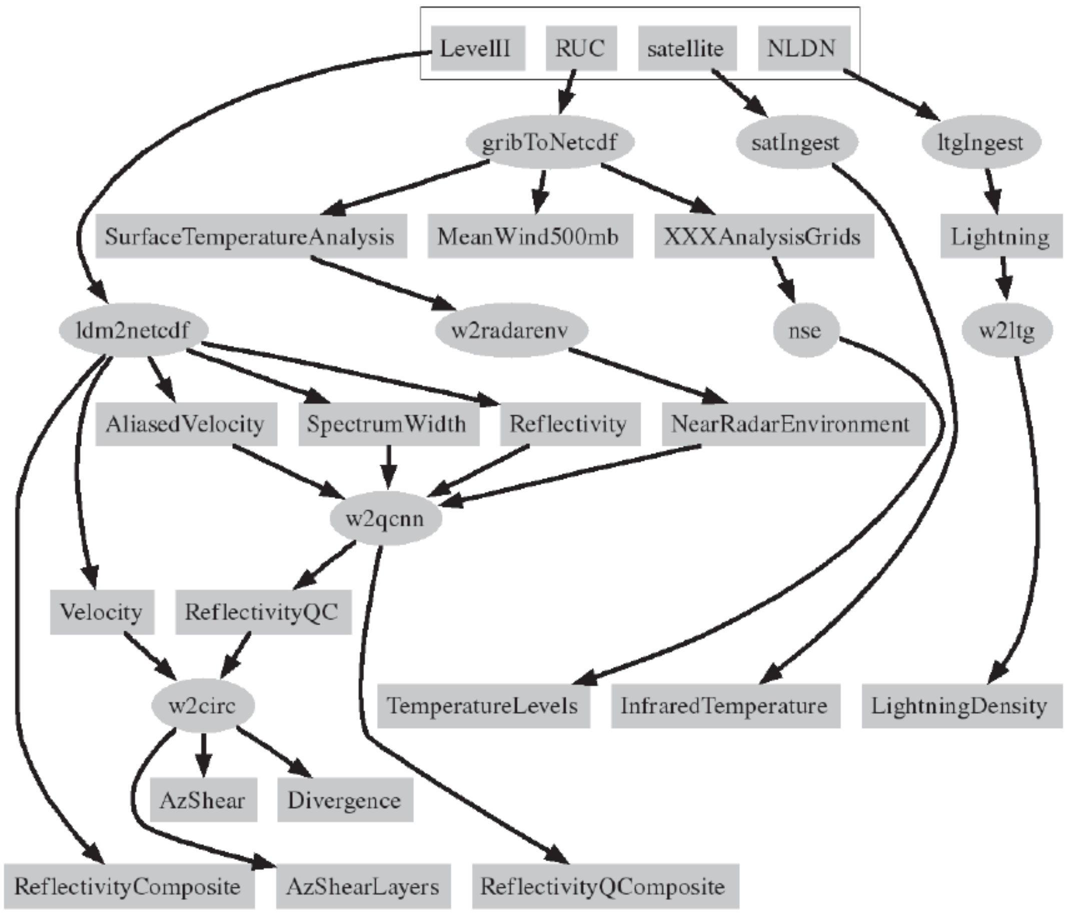


Figure 2.11 Creation of diagnostic products using automated algorithms in real-time [22].

2.8. Calder System

The Calder [23] [24]is a distributed events processing system with a centralized service to accept query requests, optimizes and deploys the queries. This stream processing system provides access to stream data for different applications and it follows the Service Oriented Architecture (SOA).

Figure 2.12 presents the architectural components of the Calder. The Calder is composed with data management and query management sub-systems. These two sub-systems communicate through the pub-sub system. The data management sub-system comprises of four services. The Calder architecture is illustrated in Figure 2.13. Grid Data Service (GDS) is used to submit continuous queries. Query planner service selects an execution plan for the query by decomposing the query into fragments and distributes it to the query execution engines on different computational hosts [25]. The query planner transforms the SQL query into an intermediate representation. The stream rowset service is a buffer between timely streams and programs. It contains a ring buffer of event data per active query in the etwork so there can be thousands of ring buffers active simultaneously. The input streams enter into the system through a pub-sub system. The arriving events are analysed with the depth first traversal of the query tree through the query operators such as select, project and join. The resulting streams are stored in the ring buffer. The stream registry service captures domain specific metadata in streams which is registered with the Calder system [23].

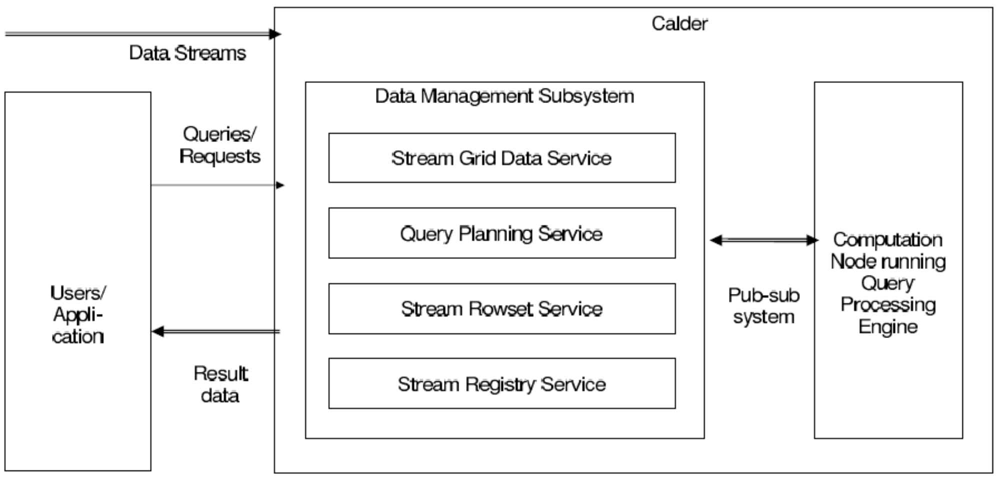


Figure 2.12 Calder architectural components [23] [24].

Single instance of Calder system can spawn multiple internal services and query processor engines to handle the workload [24]. Calder supports SQL-like continuous queries with normal constructs and it supports special constructs like EXEC, START and EXPIRE. Exec is to execute the user defined functions, START is to specify the start time of query and EXPIRE is to specify the end time of a query [23]. Query 1 is a sample filtering and mining query. It is applied on NexRad Level II Doppler radar data to detect the vortex pattern where the intensity value exceeds a specific threshold then it will issue a response trigger [25].

```
 SELECT * FROM NexRad Level II
 WHERE southBound >= "28.00"
       and eastBound <= "-89.00"
       and northBound <= "31.00"
       and westBound >= "-91.00"
 EXEC_FUNC MDA_Algorithm
START "2006-03-24T00:00:00.000-05:00"
 EXPIRE "2006-03-25T00:00:00.000-05:00";
```

Query 1 Sample query

Query processing engine accepts queries and converts them to the compiled code and it runs at each computational host in the network.

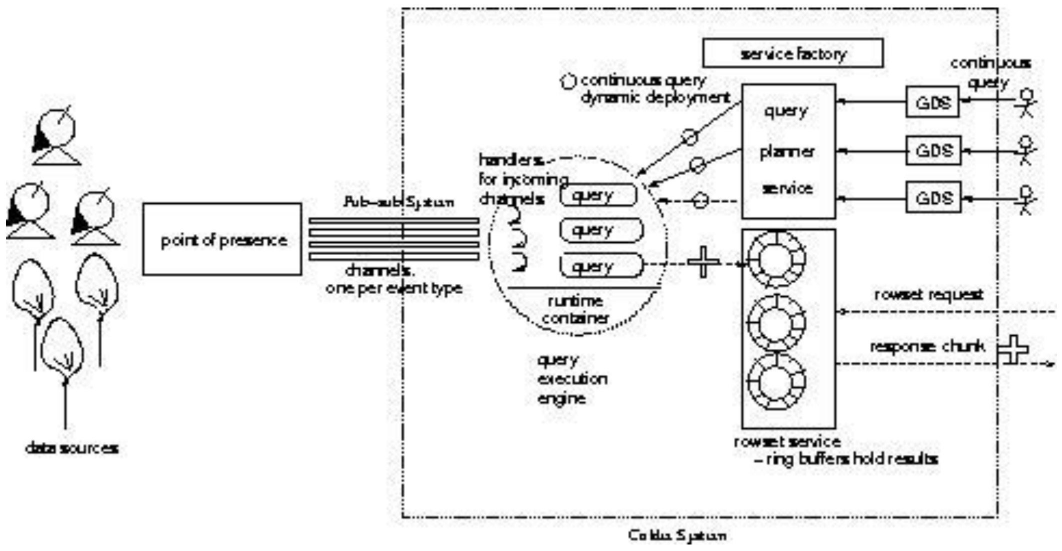


Figure 2.13 Calder architecture [25].

Calder system is applied in LEAD which is a meteoroidal forecasting model. In LEAD the incoming data is extracted as XML events. Calder converts these XML events into the internal C format for processing and the results are converted back to XML [25].

MDA algorithm is a feature detection algorithm, which is used to identify candidate mesocyclone features. ADaM classifier is used to determine a true mesocyclone feature and Calder generates a WS-Notification message to the forecast simulations [25]. In one of its projects the WS-Messenger is a publisher-subscriber system which is used to establish the communication between the Calder components and the provenance service. It uses a database to catalogue the subscriptions. So it is possible to restore the WS - Messenger server from crashes [16]. Figure 2.14 shows how the Calder is applied in the LEAD project, which is very much related to the proposed technique.

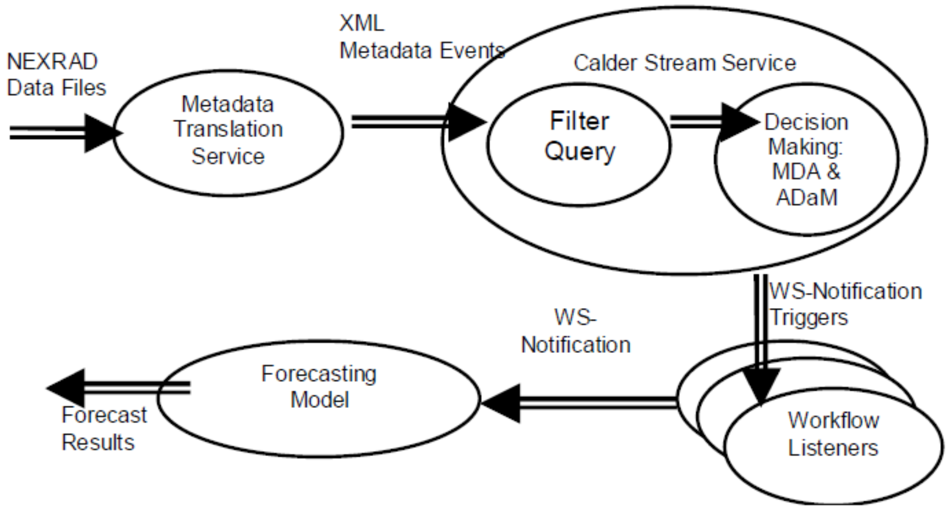


Figure 2.14 Application of Calder in LEAD [25] .

2.9. Weather Detection

2.9.1. Introduction

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, hurricane, typhoons and ice storms like natural disasters. These weather conditions occur due to air pressure differences between different places [26]. Different kinds of techniques are used to detect sudden changes in weather which can cause major disasters. Early detections of such situations help humans in several aspects. This section describes some weather detection algorithms, meteorological variables, weather detection sensors, and data sources in order to provide the background knowledge of weather detection technologies

Tornado Detection

Particular patterns of Doppler weather radar data are used to detect tornadoes. These radars measure the velocity and the radial direction of the winds in a storm to spot the rotations of it [27] .

Mesocyclones Detection

The rotating updraft or downdraft structures inside severe thunderstorms are called mesocyclones. This is usually 2-6 miles in diameter which is much larger than a tornado [27]. Over 90% of mesocyclones are accompanied by severe weather such as tornadoes or large hail hence it is important to detect mesocyclones. The mesocyclone signatures appear as couplets of incoming and outgoing velocities in radars. Mesocyclones use a velocity signature known as a Rankine Vortex for the detection process. The national severe storm laboratory mesocyclone detection algorithm (NSSL – MDA) is one such algorithm which identifies a broader spectrum of mesocyclones and has an improved probability of mesocyclone feature detection. [17].

Storm Detection

A threshold value is used in storm detection. The data points with intensities higher than the specified threshold are identified for further processing. Lightning detectors indicate electrical activity and weather radar indicates precipitation to detect storms. LEAD project uses a flexible storm detection algorithm based on user defined thresholds [26].

2.9.2. Weather Detection Sensors

There are many weather detection sensors which can be used to detect changes in meteorological variables. Figure 2.15 shows the road weather maintenance system where the different kinds of weather detection sensors are used to collect data and pass them to the Weather Monitoring Station. Several authorities will acquire that information and provide alerts to relevant agencies about difficult situations. These applications help people in their day-to-day life and reduce the possibility of accidents. The following list describes the available weather detection sensors in the road weather maintenance system [28].

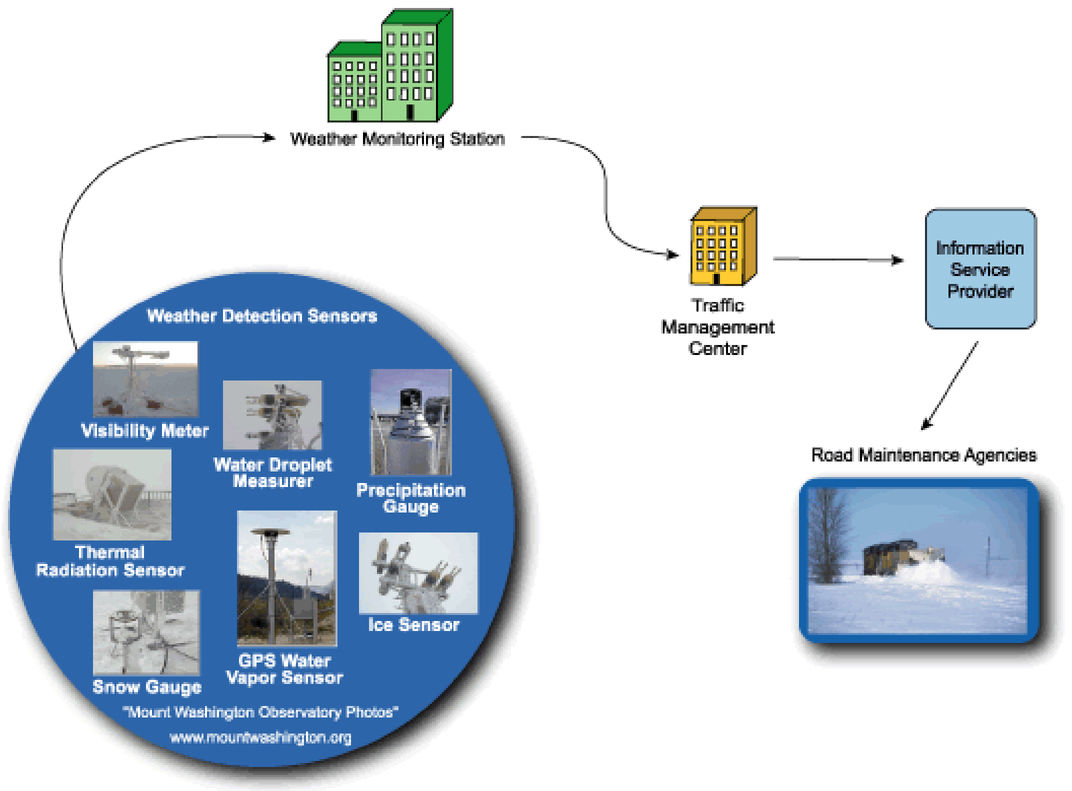

Weather Monitoring Station
Traffic Management Center
Information Service Provider
Road Maintenance Agencies
Weather Detection Sensors
Visibility Meter
Water Droplet Measurer
Precipitation Gauge
Thermal Radiation Sensor
Ice Sensor
GPS Water Vapor Sensor
Snow Gauge
"Mount Washington Observatory Photos"
www.mountwashington.org

Figure 2.15 Road weather maintenance system [28].

Weather detection sensors [28]:

1. Ice sensor – It senses the ice conditions by measuring ice crystals and water droplets.
2. Precipitation Gauge – This measures the precipitation and visibility conditions.
3. Water droplet measurer – This measures water droplet size, diameter and concentration.
4. Visibility meter – This measures cloud, precipitation and visibility conditions
5. Thermal Radiation Sensor – This detects thermal radiative emissions from cloud water and ice crystals.
6. Snow Gauge – This detects and determines the snowfall rate.
7. GPS water vapor sensor – This analyzes water vapor content in the atmosphere.

2.9.3. Meteorological Variables

Several meteorological variables have been introduced in different projects. Precipitation, wind, temperature and cloud cover are used in [29]. Reflectivity, differential phase, correlation coefficient and Doppler velocity are introduced in [20]. Relative humidity is used to predict rain in [30]. 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 [31].

2.9.4. MADIS (Meteorological Assimilation Data Ingest System)

MADIS works as a meteorological observational database and a data delivery system covering the globe. Figure 2.16 shows the overall architecture of MADIS.

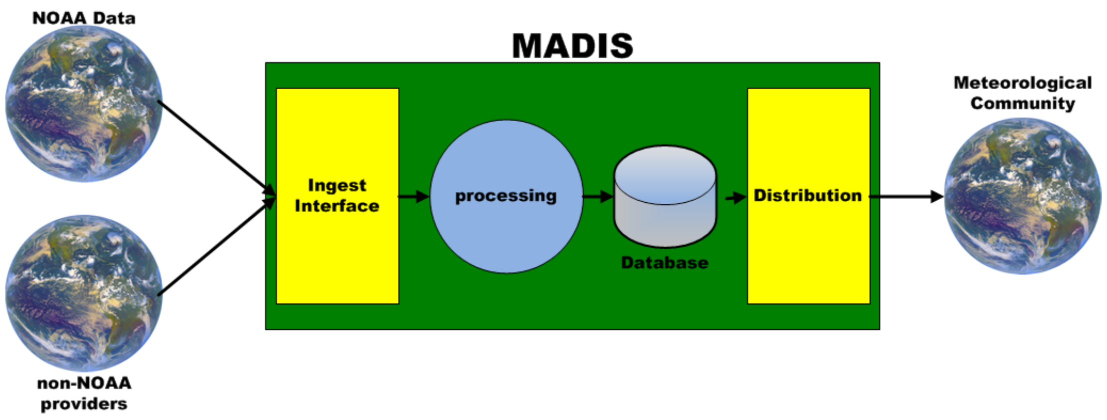
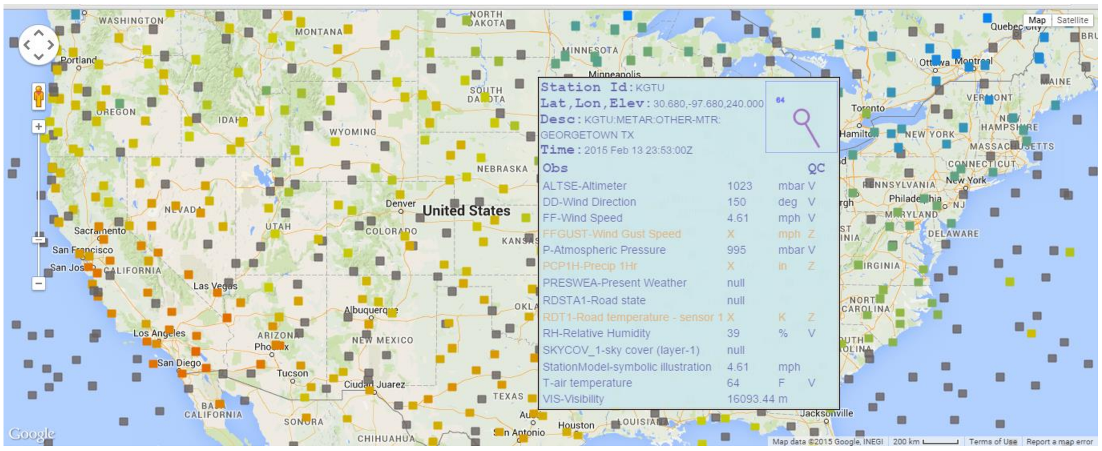


Figure 2.16 Components of MADIS [32].

NOAA data sources and non-NOAA providers ingest data to the MADIS. It decodes these observational data and converts to a common format. Finally, the meteorological community can access these data from MADIS observational database [32].

It is possible to receive real-time data from MADIS. It is recommended to use Text/XML viewer accounts since the system needs data on demand. If it is required a continuous data feed then data can be accessed via ftp or LDM (Unidata's Local Data Manager) to gain a high performance [32].

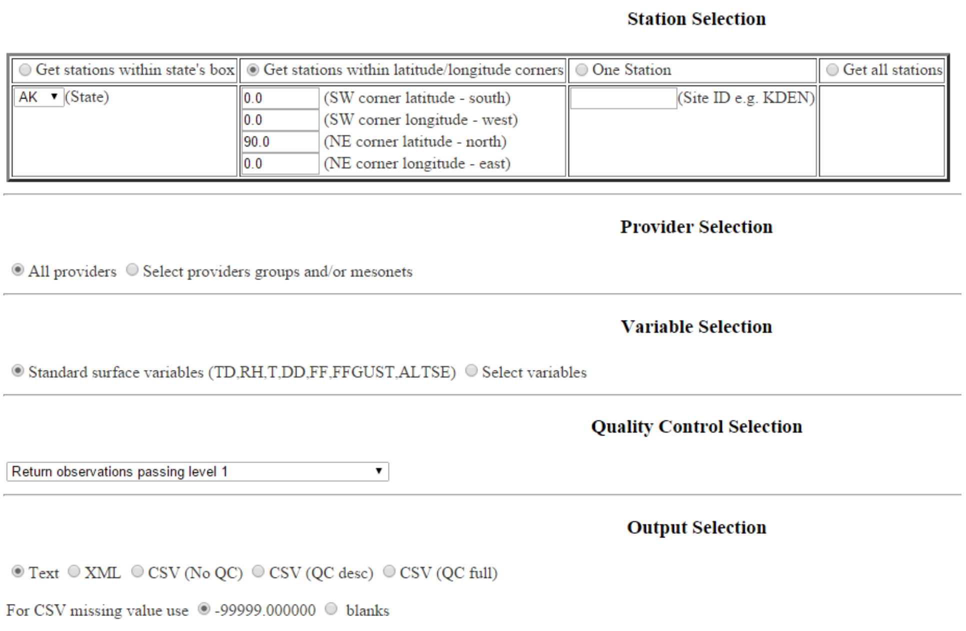
The API of the MADIS allows users to easily access to the observations and quality control information. The geographic coverage of this data set is over North and Central America and Hawaii. Figure 2.17 shows the distribution of the weather stations. These files contain data from 15 minutes before a given hour and 44 minutes after the hour. So the data are segmented into hourly files. User can specify the missing value indication as a blank or a numeric value (-99999). User needs to specify station and observation types, quality control (QC) choices, domain and time boundaries as seen in Figure 2.18 [32].



Station Id: KGTU
Lat,Lon,Elev: 30.680,-97.680,240.000
Desc: KGTU:METAR:OTHER-MTR:
GEORGETOWN TX
Time: 2015 Feb 13 23:53:00Z

Obs			QC
ALTSE-Altimeter	1023	mbar	V
DD-Wind Direction	150	deg	V
FF-Wind Speed	4.61	mph	V
FFGUST-Wind Gust Speed	X	mph	Z
P-Atmospheric Pressure	995	mbar	V
PCP1H-Precip 1Hr	X	in	Z
PRESWEA-Present Weather	null		
RDSTA1-Road state	null		
RDT1-Road temperature - sensor 1	X	K	Z
RH-Relative Humidity	39	%	V
SKYCOV_1-sky cover (layer-1)	null		
StationModel-symbolic illustration	4.61	mph	
T-air temperature	64	F	V
VIS-Visibility	16093.44	m	

Figure 2.17 Weather station display [32].



Station Selection

○ Get stations within state's box	[✓ ◉] Get stations within latitude/longitude corners	○ One Station	○ Get all stations
AK ▾ (State)	0.0 (SW corner latitude - south) 0.0 (SW corner longitude - west) 90.0 (NE corner latitude - north) 0.0 (NE corner longitude - east)	(Site ID e.g. KDEN)	

Provider Selection

[✓ ◉] All providers ○ Select providers groups and/or mesonets

Variable Selection

[✓ ◉] Standard surface variables (TD,RH,T,DD,FF,FFGUST,ALTSE) ○ Select variables

Quality Control Selection

Return observations passing level 1 ▾

Output Selection

[✓ ◉] Text ○ XML ○ CSV (No QC) ○ CSV (QC desc) ○ CSV (QC full)

For CSV missing value use [✓ ◉] -99999.000000 ○ blanks

Figure 2.18 MADIS Meteorological Surface Text/XML Viewer.

2.9.5. Weather Detection Algorithms in LEAD

Mesocyclone Detection Algorithm (MDA)

The National Severe Storm Laboratory (NSSL) MDA is an automated mesocyclone signature detection algorithm. The designed Mesocyclone Detection Algorithm uses this velocity signature, which represents incoming and outgoing radial velocity [17].

Storm Detection Algorithm (SDA)

This algorithm is similar to an image thresholding algorithm. The data points with intensities higher than a provided threshold are retained. This SDA uses the region growing technique and build 3D volumes from these retained pixels. The volumes that meet the minimum size criteria are kept since storms must have a minimum size and spatial volume. This algorithm provides the spatial location, maximum reflectivity, size and depths of the storm as the output [17]

This MDA and the SDA can be used together to identify storms and tornadoes.

2.9.6. Non-precipitation Weather Event

A meteorological phenomenon such as wind, extreme heat or cold is defined as a non-precipitation weather event. These products issued by the US National Weather Service (NWS) weather forecast offices are described in [33]. The multi-tiered concept of the NWS non-precipitation weather warning program explains the awareness of the event. It has three concepts Outlook, Watch and Warning/Advisory. An *Outlook* indicates that a hazardous non-precipitation weather event may develop. A *Watch* indicates that the risk of a hazardous non-precipitation weather event has increased, but unable to provide any certain information. A *Warning* indicates that an event is occurring or has a high probability of occurrence. When a warning is issued there is a threat to people and their property. An *advisory* is used for less serious conditions [33]. We can use the proposed system to generate warnings since they indicate the events which are currently occurring. Table 2.2 describes the non-precipitation weather warning types.

Table 2.2 Non-precipitation warning products.

Warning	Description
Dust Storm Warning	Widespread or localized blowing dust reducing visibilities to ¼ mile or less. Sustained winds of 25 mph or greater are usually required.
Excessive Heat Warning	Heat Index (HI) values forecast to meet or exceed locally defined warning criteria for at least two days (Typical values: 1) Maximum daytime HI>=105°F north to110°F south and 2) Minimum night time lows >=75°F).
Extreme Cold Warning	Operational in Alaska only. When forecast to occur for at least three consecutive days: Shelter temperature of -50˚F or colder and air temperature remains below -40˚F up to the 700-mb level.
Freeze Warning	Minimum shelter temperature is forecasted to be 32°F or less during the locally defined growing season.
Hard Freeze Warning	Minimum shelter temperature is forecasted to be 28°F or less (slightly lower or higher based on local criteria) during the locally defined growing season.
High Wind Warning	Wind speeds forecast to meet or exceed locally defined warning criteria. (Typical values are sustained wind speeds of 40 mph or greater lasting for 1 hour or longer, or winds of 58 mph or greater for any duration).

Wind Alerts

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. Beaufort wind force scale is to classify the wind alerts. Table 2.3 shows a sample classification [33].

Table 2.3 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

Temperature Alerts

Excessive Heat Warnings are issued when it achieves the specific criteria which vary among the countries. This specific criteria depends on the climate variability and the effect of excessive heat on the local population. If the maximum daytime temperature is above 41 ^{o}C to 43 ^{o}C and the minimum night time temperatures is above 24 ^{o}C then it is considered as a typical Heat Index (HI) value. It considers this extreme HI values for at least two days to issue a temperature alert [33].

2.9.7. Characteristics of Radar Images

An individual block in a radar image is called a *pixel*, *bin* or *gate*. The radar uses *range* and *azimuth* for measuring location on the radar. Range means the distance in nautical miles (a nautical mile is equal to 1.15 regular miles) from the radar site. Azimuth means the angle between the radial that points to the true north and the radial that points to the pixel of interest. A colored pixel which is called an echo or return represents the detected data. An echo can be a large group of pixels. These characteristics are visualized in Figure 2.19 [34].

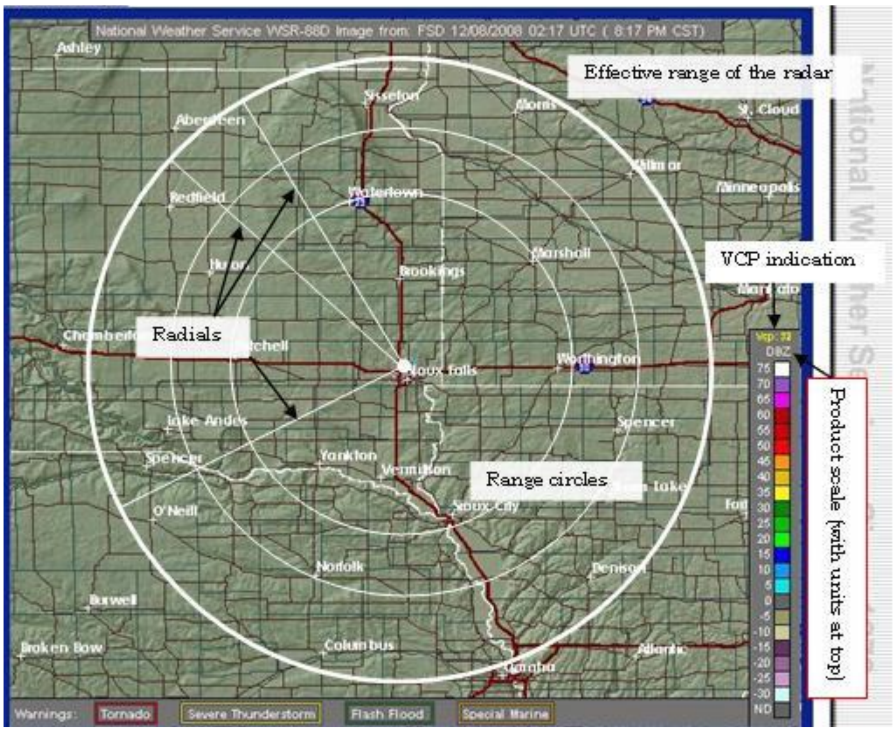

National Weather Service WSR-88D Image from: FSD 12/08/2008 02:17 UTC (8:17 PM CST)
Effective range of the radar
VCP indication
Radials
Range circles
Product scale (with units at top)
Warnings: Tornado Severe Thunderstorm Flash Flood Special Marine

Figure 2.19 Sample radar image.

Base Reflectivity

Base reflectivity is a radar product which displays the amount of energy that has returned to the radar. This is measured in dBZ (i.e., decibels relative to reflectivity Z). The scale defines the strength of returns to the radar with colors. The base reflectivity shows echoes when the radar energy bounces back to it [34].

A reflection of a wave can be formed because of a boundary between warm, moist air and cool, dry air. This boundary will be seen as a very narrow line of light reflectivity. It is common to see storms form directly on boundaries when the atmosphere is unstable and thunderstorms are likely to form [34].

This can be used to identify precipitation in radar images. A soft computing model for nowcasting of Yes/No rain situations is presented in [35] using Doppler weather radar reflectivity imageries.

Base Velocity

Base velocity is a radar product, which defines the average wind speed of the detected particles. This can be measured in knots (kts). Base velocity can be used to understand the things which are happening in the atmosphere. It can be used to determine the amount of wind shear (i.e., change of velocity with height). It can be used to detect the amount of rotation present in the storms, whether it is clockwise or cyclonic [34].

Volume Coverage Patterns (VCPs)

We can see the abbreviation VCP with a number on some radar displays. It describes the patterns which the radar scans the atmosphere. The radar may scan at up to 14 different elevation angles depending on the VCP. VCP is labeled as a two or three digit number (1, 2 or 3 only) [34].

3. RESEARCH METHODOLOGY

3.1. High-Level Architecture

The main objective of this study is to apply CEP in weather anomalies and events detection. This system monitors sensor data and match them with a predefined set of queries. It detects unusual weather events and provides alerts for further verification. The high-level architecture of the proposed system is presented in Figure 3.1. The system is mainly divided in to 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 [32] and compares with predefined scalar values. Radar image based weather detection considers reflectivity values of radar images and compares them with a threshold.

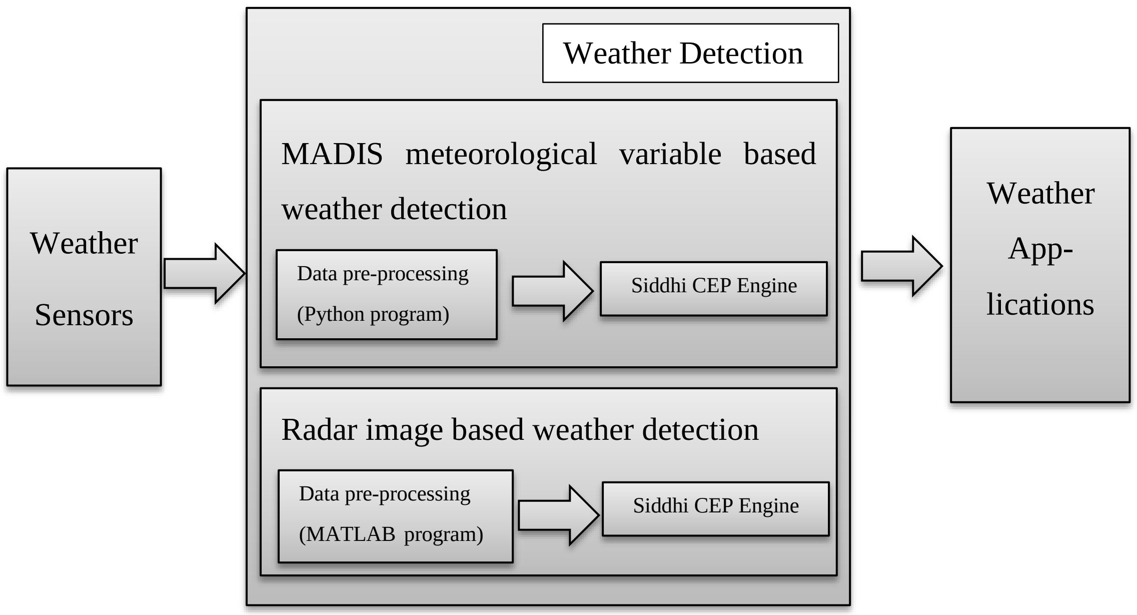

Weather Detection
Weather Sensors
MADIS meteorological variable based weather detection
Data pre-processing (Python program)
Siddhi CEP Engine
Radar image based weather detection
Data pre-processing (MATLAB program)
Siddhi CEP Engine
Weather App-lications

Figure 3.1 The proposed architecture.

The system gets two input types, surface observation datasets and radar images. These inputs are pre-processed separately in order to make them ready for the detection process. A Python program is used to pre-process surface observation datasets which are obtained from the MADIS. A MATLAB program is used to pre-process the radar images and prepare a 2D matrix (240x240) to apply as input events.

These inputs are used as the incoming events of the CEP engine. The system creates Siddhi manager with specific configurations. Siddhi provides partitions as a feature to increase the performance of the processing. The program has used partitions to process the weather station information separately. There are two Siddhi managers, one is for MADIS meteorological variable based weather detection and the other is for radar image based weather detection. Siddhi CEP engine identifies event patterns according to the predefined queries and sends alerts to necessary weather applications for further analysis.

3.2. Development Environment

This system is developed using the Java 1.8 platform. Data pre-processing parts are done using Python and MATLAB environment. Siddhi version 3.1.0 is used to identify event patterns and act on them in real-time. It can process more than 2.5M events per second on single server commodity hardware. Siddhi supports a large number of queries via partitioning into different servers and it is horizontally scalable to support very large event volumes [11]. MATLAB 7.1 is used to do the image pre-processing works. A Python 2.7.3 based program is developed to download and pre-process MADIS data.

3.3. MADIS Meteorological Variable Based Weather Detection

MADIS meteorological variable based weather detection considers certain values of meteorological variables and compares with predefined scalar values to do the weather detection and warning.

3.3.1. Meteorological Variables

Different meteorological variables have been introduced in different projects. According to the literature review some of the most relevant meteorological variables are identified such as relative humidly, pressure, air temperature, wind direction, wind speed and accumulated precipitation. Our system is able to detect the significant changes of the selected meteorological variables using the Siddhi CEP engine.

3.3.2. Weather Sensors

This study was made possible in part due to the data made available to the National Oceanic and Atmospheric Administration by several providers. Users can access real-time data or an online archive of saved real-time data. Text/XML viewer has been used since this study deals with saved real-time data and that is available only with surface observation datasets. The system have considered relative humidity (RH), air temperature (T), wind direction (DD), accumulated precipitation – 1h (PCP1H), wind speed (FF), Elevation (ELEV), latitude (LAT) and longitude (LON). A Python program is used to download the available data for a given date, time and a location. It saves that information in separate text files with the date and time.

3.3.3. Data Pre-processing

The downloaded data contained unwanted html tags and duplicates. So they were needed to be pre-processed before proceeding with further calculation. A Python program is used to remove the html tags and clear the data files. The system has downloaded these data into several files and needed to combine these files without any duplicates. Figure 3.2 explains the pre-processing steps. This stage makes the raw data ready for the event pattern matching process. This output file helps to simulate weather sensors. It contains the timestamp and the values of the selected meteorological variables. So the system is able to read data from this file and use in Siddhi as the input events.

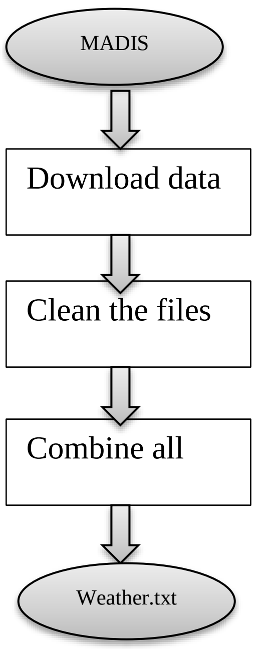


Figure 3.2 Pre-processing steps.

3.3.4. Stream Definition

The weather stream is used to implement the MADIS meteorological variable based using Siddhi CEP engine. It defines the input event pattern with necessary information.

```
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 attribute list of the weather stream is explained in Table 3.1. The timestamp is used to add the temporal aspect to data. These sensor data are 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 3.1 Weather stream definition.

Attribute	Data type	Description
Timestamp	Double	Reading time of the particular data
Wsid	String	Weather station id
Prov	String	Provider of weather data
subPro	String	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

3.4. Radar Image Based Weather Detection

Radar images play a major role in the weather detection field. We need to analyse these complex radar images in order to identify suspicious pixels on them. The proposed system processes these radar images and identifies precipitations pixels in a given radar image. It needs to pre-process these radar images and convert them to arrays since Siddhi is unable to process images directly. Section 3.4.1 explains a sample scenario of the radar image based weather detection using the proposed system.

3.4.1. Image Pre-processing

Siddhi is unable to process images hence the system needs to modify the input data. Siddhi manager can manipulate arrays so a 2D array will contain the radar image data. The radar image will be converted to a 2D array (240x240) using a MATLAB program. Figure 3.3 illustrates the key steps of the program and it is further explained in Figure 3.4 Each element of this array represents a reflectivity value.

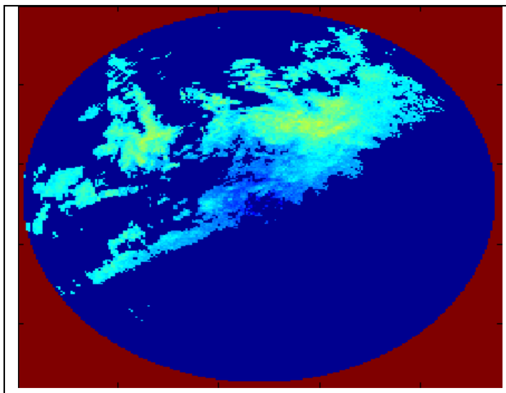
	01 102 107 99 92 102 102 94 82 0 0 88 97 102 94 92 101 106 100 100 91 87 95 0 0 0 85 89 95 104 100 95 101 101 105 105 95 95 88 0 0 0 0 85 89 95 104 102 …
(a) Input image	(b) Matrix with reflectivity values

Figure 3.3 Reflectivity matrix.

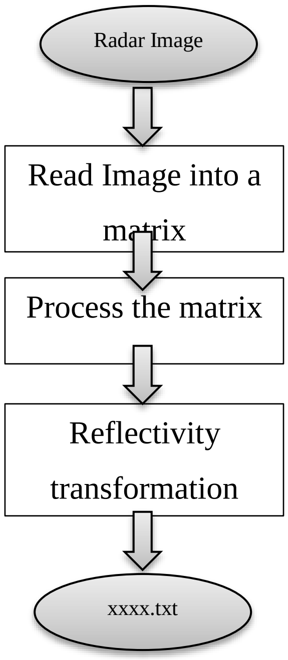


Figure 3.4 MATLAB program.

3.4.2. Stream Definition

The following radar stream is used to implement the radar image based weather detection using the Siddhi CEP engine. It defines the input event pattern with the necessary information:

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

Table 3.2 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 3.2 Radar stream definition.

Attribute	Data type	Description
Timestamp	Double	Reading time of the particular data
Matrix	String	240x240 matrix which contains the reflectivity values of the radar image

3.5. Use Cases

We have used four use cases to cover the common scenarios found in the weather detection and warning system. First use case mainly compares an input value with a threshold value while other use cases process the location specific weather information. The following sub sections explain each of these use cases with more details.

1. Comparing sensor data with predefined thresholds
    a. Comparing meteorological variables with scalar values.
    b. Identifying suspicious pixels of radar images.
    c. Building queries with multiple meteorological variables.
2. Identifying weather stations with defects and suggest alternative values.
3. Identifying anomalies in weather data.
4. Identifying nearby weather situation in a given location.

While several approaches have been used to implement these use cases such as neural networks [31], model output statistics [30] and probabilistic extreme forecast index [29]. However this study presents how Siddhi can be applied in weather anomalies and events detection to solve these use cases. Separate queries are written to implement these use cases in Siddhi. Use cases 1.a, 1.c, 2, 3 and 4 are implemented for MADIS meteorological variable based weather detection and 1.b is implemented under the radar image based weather detection.

3.5.1. Comparing Sensor Data with Predefined Thresholds

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 were written to identify anomalies on these variables by comparing with scalar values which have been introduced in Chapter 2 – Section 2.9.6.

As an example, Query 2 defines the partitions that consider Beaufort classification of window speeds (see Table 2.3). 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.

```
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 Define partitions

Query 3 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.

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

Query 3 Wind alert

Identifying Suspicious Pixels of Radar Images

Radar image based weather detection considers reflectivity values of radar images and compares them with a threshold. Section 3.4 explains more about this use case under the radar image based weather detection.

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. The Query 4 represents a wind alert and it triggers when a high wind condition towards the North East direction is identified.

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

Query 4 Wind alert

3.5.2. Identifying Weather Stations with Defects and Suggest Alternative Values

When we process sensor data we may find defects which can be either 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.

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 Equation 3.1 [38].

$$
\begin{aligned}
\mathit{Distance} = \mathit{ACOS}(&\mathit{COS}(\mathit{RADIANS}(90\text{-}\mathit{Lat1}))\ {*}\mathit{COS}(\mathit{RADIANS}(90\text{-}\mathit{Lat2})) \\
&+ \mathit{SIN}(\mathit{RADIANS}(90\text{-}\mathit{Lat1}))\ {*}\ \mathit{SIN}(\mathit{RADIANS}(90\text{-}\mathit{Lat2})) \\
&{*}\mathit{COS}(\mathit{RADIANS}(\mathit{Lon1}\text{-}\mathit{Lon2})))\ {*}\ 6371
\end{aligned} \qquad (3.1)
$$

Where, Lat1, lon1 are the locations of the first weather station and Lat2, 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 the equation 3.1.

GetIsNearTime

This function is used to compare the time difference between two weather stations. To propose alternative values it needs to compare the both reported times, but the timestamp is not always equal. Therefor 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 is used four hours as the time window. Further, it can notify whether there is any technical errors at the first weather station. Missing values are defined with -99999, so the 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 a same time and suggest alternative 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;
```

Query 5 Find missing values.

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.

```
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; ");
```

Query 6 Find alternative values.

3.5.3. 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 in order 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.

```
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;
```

Query 7 Identify anomalies.

3.5.4. Identifying Nearby Weather Situation of a Given Location

General population are interested in knowing the current weather situation at a given location. Different types of wind alerts are defined in chapter 2 - Table 2.3. For an example, if someone wants to know whether a particular area is being issued a weather warning, then the system allows querying the current wind speed with the specific range.

3.6. Summary

This chapter mainly explained about the methodology of the proposed system. It is divided into two parts; MADIS meteorological variable based weather detection and the radar image based weather detection. It further presents four use cases which can be used to verify common scenarios of a weather anomalies and events detection system. Each use case is explained with sample queries so it helps to understand them properly.

The system has used Complex Event Processing technologies to implement these main use cases. Siddhi CEP engine is used with the available features. Additionally it consists sub modules to do the pre-processing and test data generation. These sub modules are developed using Python and MATLAB.

4. PERFORMANCE EVALUATION

Next, we evaluate the accuracy and the performance of the proposed weather anomalies and events detection system. It evaluates the main use cases which are defined in Section 3.5. Further, it evaluates the performance of the system with respect to the event detection time. The system should be able to work when the numbers of events, weather stations, product count or the number of queries are increased; hence, the main concern of this section is to find out whether the proposed system can handle a large number of events at once and accurately, when they arrive at a higher input rate.

4.1. Emulation Setup

Emulation setup is presented in Figure 4.1. 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.

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.

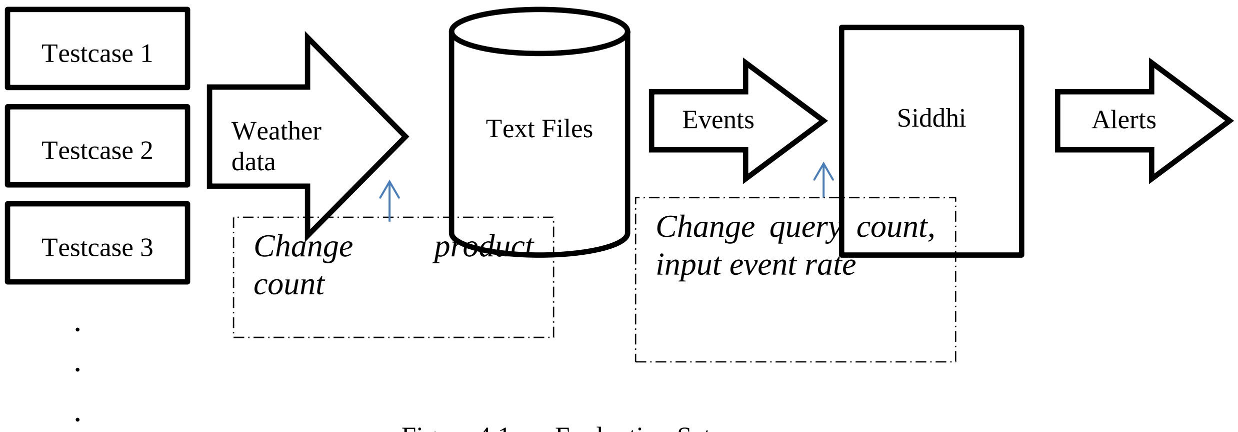
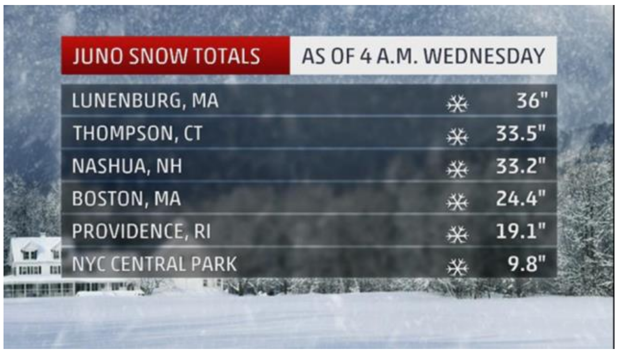


Figure 4.1 Evaluation Setup.

4.1.2. Test Data

The system had considered storm ‘Juno’ for the verification [39]. The US National Weather Service has dropped all winter storm and blizzard warnings for Juno, which was pounded on 26 – 27th January 2015. Several locations in Massachusetts had picked a large amount of snow. Most severe coastal flooding occurred in eastern Massachusetts and wind speed was 50-80 mph. Figure 4.2 shows the top snowfall totals from Winter Storm Juno. This incident was used to test all the meteorological variable values.



JUNO SNOW TOTALS	AS OF 4 A.M. WEDNESDAY
LUNENBURG, MA	36"
THOMPSON, CT	33.5"
NASHUA, NH	33.2"
BOSTON, MA	24.4"
PROVIDENCE, RI	19.1"
NYC CENTRAL PARK	9.8"

Figure 4.2 The top snowfall totals from Winter Storm Juno **[39]**.

Around 500 weather stations of the Massachusetts (MA) state were considered for testing the use cases. These details of the test data vary with the use case so that information is given under each use case description.

The downloaded data from MADIS had been used to evaluate the results. The test data file contains meteorological data from multiple weather stations. The timestamp was used to add the temporal aspect to the input events, but some test cases have ignored this timestamp in order to consider the highest input event rate. Radar image weather detection part was verified with a sample collection of radar images from [40].

4.2. Comparing Sensor Data with Predefined Thresholds

4.2.1. 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. For an example, Figure 4.3 shows how the temperature goes down due to the storm Juno on 27 the January 2015 at D9545.

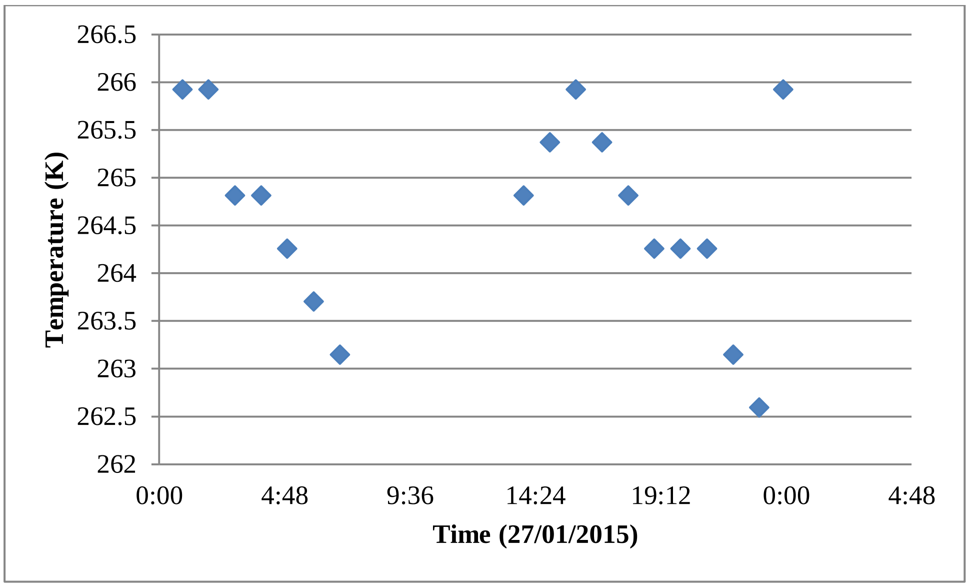


Figure 4.3 Temperature data.

The proposed system was able to identify that the temperature has gone down to 262.5K at 22:58 pm (UTC). Several test cases were executed with 13,063 input events. Table 4.1 shows the total number of detections of the each test case.

Table 4.1 Test results.

Test Case	Actual value	System Value
Temperature < 263 K	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. Table 4.2 indicates the event detection count distribution among the weather stations and it concludes that the system was able to detect weather events successfully.

Table 4.2 Total event count.

Weather station ID	Event detection count
44020	2
44013	11
AR250	24
AR511	24
C0863	14
C1532	11
C3008	5
C3094	3
…	…

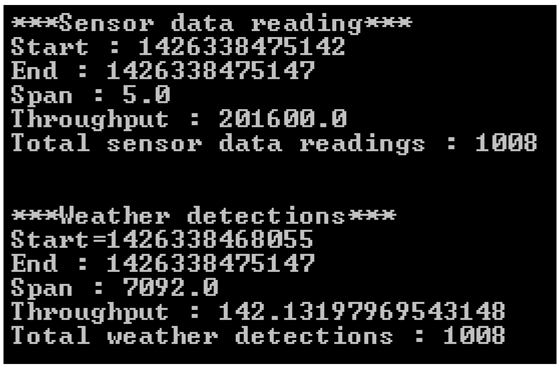
4.2.2. Identifying Suspicious Pixels of Radar Images

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.

Scenario: The input event consist a 240x240 matrix of double values. Threshold of the precipitation is '1'. The following query was used to look for precipitation pixels in the radar images [40].

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

Sample of 1,008 radar images had been used in the image-based weather detection. Figure 3.3 in Chapter 3 shows a sample radar image. 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. Figure 4.4 indicates that total sensor data readings was same as the total weather detections.



```
***Sensor data reading***
Start : 1426338475142
End : 1426338475147
Span : 5.0
Throughput : 201600.0
Total sensor data readings : 1008


***Weather detections***
Start=1426338468055
End : 1426338475147
Span : 7092.0
Throughput : 142.13197969543148
Total weather detections : 1008
```

Figure 4.4 Image based weather detection.

4.2.3. 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 MA were reported high wind conditions with a relative humidity as 100% on 27th January 2015. Table 4.3 contains the test results.

Table 4.3 Test results.

Time (UTC)	Weather station	Wind speed	Relative humidity
13:56	KHYA	16.4622	100
9:52	KCQX	13.3755	100

4.3. Identifying Weather Stations with Defects and Suggest Alternative Values

The MADIS dataset had missing values for some of the meteorological variables. Figure 4.5 shows two nearby weather stations; FSKM3 (42.109, -72.124) and AR824 (42.130, -72.098). 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 4.4 contains the test results. Further there can be a technical fault at FSKM3.

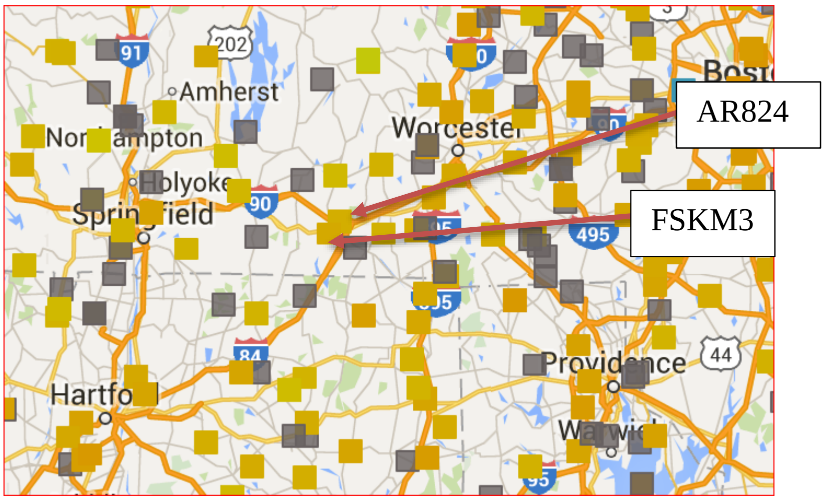


Figure 4.5 Nearby weather stations.

Table 4.4 Test results.

Time	FSKM3	AR824	Expected Value	Actual Result
0.40	-99999	51	51	51
2.00	-99999	59	59	59
2.05	-99999	56	56	56

4.4. 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 used to find nearby weather stations and their readings at the same time.

The weather station FSKM3 had reported temperature as 262K at around 13:30 PM on 26th January 2015. This use case was tested by comparing nearby weather station values. So the result was that AR824 also had reported the same value for the temperature at that time. Table 4.5 contains the test results of the weather station AR824.

Table 4.5 Test results.

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

4.5. Identifying Nearby Weather Situation of a Given Location

If someone wants to find out the current weather situation of Massachusetts as of January 27, they can query the weather data given the location information. When user search for the temperature values of the specific weather stations user can get the relevant values as in Table 4.6. Further, it can find out that the KHYA weather station has being issued high wind warnings at 13.56PM (UTC) (see Table 4.3).

Table 4.6 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

4.6. Detection Time with Different Workloads

4.6.1. Increasing Number of Queries

This test case had been executed with a 648,350 events from 25th January 23:58 pm to 27th January 22:58 pm (UTC). The results are shown in Table 4.7. Timestamp was ignored, so that we could play the entire dataset as arriving within a very short time. In this case it took 13 ms to feed the data which had 648,350 samples. Figure 4.6 depicts that the total detection time increases with the number of queries. But the system was able to handle the workload effectively.

Table 4.7 Comparison of the detection time with the query count.

No of queries	Total detection time (ms)	No of detections
1	11,906	32,000
2	12,852	256,000
3	13,719	288,000
4	14,321	288,000
5	14,872	320,000

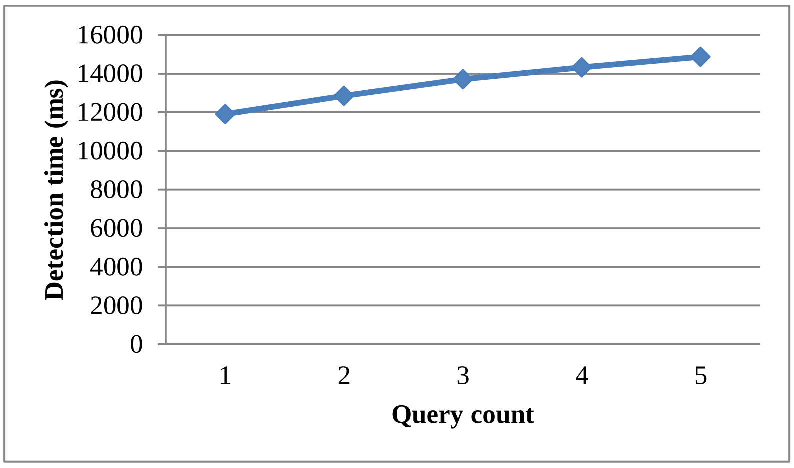


Figure 4.6 Comparison of the detection time with the query count.

4.6.2. Increasing Number of Weather Stations

The purpose of this test case was to compare the time taken to read these data and the time taken to detect abnormal weather readings while increasing the weather station count. Table 4.8 shows the test results. It is obvious in Figure 4.7 that the reading time and the detection time had increased with the increasing weather station count. The system was able to match the sensor readings with the predefined queries, hence the total number of detections were accurate. The time difference between the sensor reading time and the detection time was almost same such that it did not depend on the weather station count.

Table 4.8 Comparison of reading time and the detection time with the input event count.

No of input events	No of detections	Total reading time (ms)	Total detection time (ms)	Time difference (ms)
415	2	1,864	1,004	860
830	4	3,746	2,921	825
1660	8	7,544	6,625	919
3319	16	15,114	14,288	826

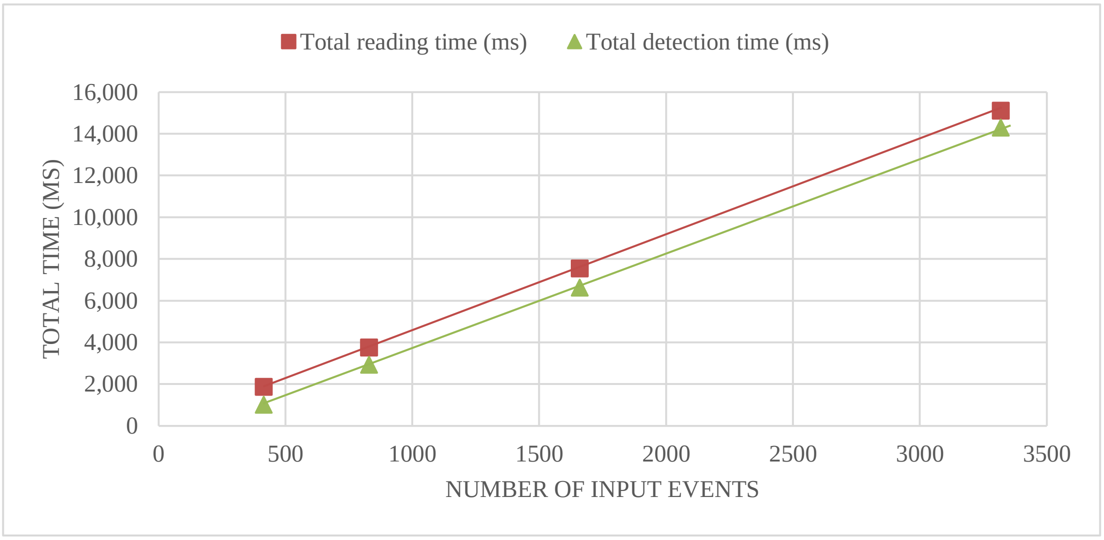


Figure 4.7 Comparison of reading and detection time with the input event count.

4.6.3. Increasing Input Product Count

Entire product list of MADIS was considered to find how the system performs with a large set of meteorological products. Queries were written to identify anomalies with the basic meteorological variables. Though the system performance doesn't change significantly with the number of product count.

4.6.4. Increasing Input Event Rate

The main concern of this test case was to increase the input event rate and check the performance of the system. The test data file was created using MATLAB. Hourly data from 25th January 23:58 pm to 27th January 22:58 pm (UTC) was used to create the test file. The original count of test data was 48. The MATLAB program created another sequence of data between the original data using interpolation.

The test case was executed with 648,350 input events within 51.6 seconds therefore, the input event rate was 12,572. Total event detections were 345,424 within 50.6 seconds and the throughput was 12,802. This concludes that Siddhi performs well with higher input rates and it can handle large number of data at a time.

4.7. Detection Delay

This test case is used to verify whether the system is able to identify the patterns with the minimum latency. So the reported time gap between receiving and identifying a specific event was 1 ms when the system has 11 parameters.

4.8. Calder vs Siddhi

Calder and Siddhi both are stream processing systems and they use SQL-like event processing languages. Siddhi can implement multiple streams using a single input event queue by multiplexing them. It has the ability to dynamically add new data formats and user defined functions. Further it sends the input event only to the corresponding executor and Siddhi eliminates duplicate states. Siddhi had used these techniques to improve its performance so it confirms the relevance of Siddhi CEP engine in this study.

4.9. Summary

First, we evaluated the accuracy of the four use cases. It compared meteorological variables with predefined thresholds to identify impending weather events, identified weather stations with defects and suggested alternative values, identified anomalies in sensor data and identified weather situations around a given location. These use cases were tested with the winter storm “Juno”. The system managed to successfully identify sensor data.

Second, several test cases were executed to evaluate the performance of the system. The event detection throughput was calculated with different number of event products and input counts. To increase the system’s workload, a larger number of queries and a larger input event rates were considered. The system performed well in each test case where the detection delay was around 2 ms.

5. CONCLUSIONS

5.1. Summary

The novel contribution of this thesis is to provide the monitoring phase capabilities to a typical Climate/Weather Observatory (see Figure 1.1) 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, as well as 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, which were 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 such as the proposed Climate Observatory system for Sri Lanka, 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.

Several approaches have been used to do weather anomalies and events detection but CEP can be identified as the most suitable contemporary technique to implement these use cases. Neural network based weather detection system [31] needs to train separate neural for each and every use case, but using Siddhi CEP engine we can easily add a new query to enhance the functionalities. Several queries can be defined parallel to implement these use cases rather than doing multiple comparisons sequentially.

The LEAD project was a very closely related to the research area, which uses CEP in weather detection. 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. 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.

5.2. Problems Encountered

It was required 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 as explained in Chapter 2.

We faced several limitations when trying to use several kind of weather data together. For an example we found lightening 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.

5.3.1. Enhancements to CEP Engines

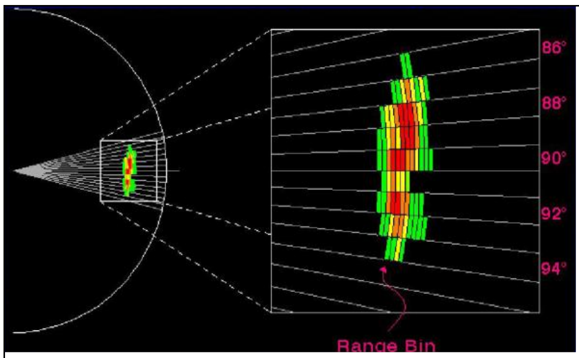
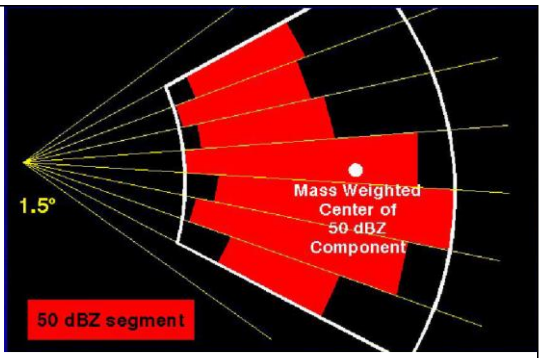
Some improvements have been identified, which are needed to be done not only in Siddhi but also in other CEP engines. Currently, Siddhi needs to support type object in stream definitions so it will be easy to send any type of object. It is required to improve object type into specialization like arrays and implement inbuilt array manipulation functionalities within Siddhi. Then the Siddhi can provide custom functions for array operations such as getArrayElement (array, index), min, max, and hasValueGreaterThan(..). Siddhi does not have in built functions to do image processing functionalities. Currently the images are required to be converted to arrays before apply with Siddhi.

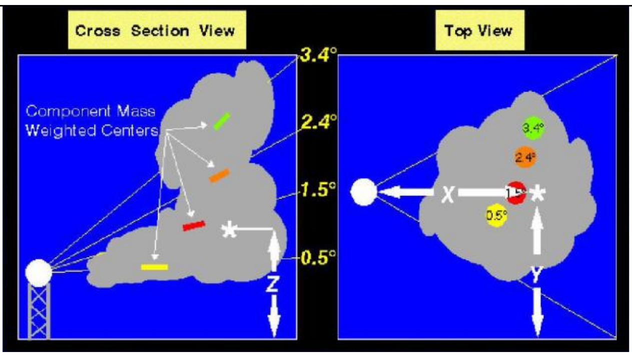
5.3.1. Storm Cell Identification and Tracking

We can implement weather detection algorithms using CEP since Siddhi allows to create user defined functions. Storm cell identification and tracking is such an algorithm. This algorithm can be used to identify storm cells. It has four sub functions: storm cell segments, storm cell centroids, storm cell tracking and storm position forecast. A storm cell is a three dimensional region, which has reflectivity values above a significant value [36].

Storm cell segments

This algorithm combines the individual range bins into storm segments along the radial. Figure 5.1 (a) shows a storm segment, which is a run of contiguous range bins (each with 1 deg x .54 nm) along a radial with reflectivity values greater than or equal to a specified threshold.

	 1.5° Mass Weighted Center of 50 dBZ Component 50 dBZ segment
(a)	(b)



(c)

Figure 5.1 Storm cell Identification (a) Storm segment (b) Mass weighted center (c) Storm centroid (* is the centroid).

Storm centroids

This algorithm groups cell segments into components, computes the components' attributes, vertically correlates the components into cells and computes the cell's attributes. A centroid is the mass weighted center of a three dimensional region of significant reflectivity as shown in Figure 5.1 (c).

Storm cell tracking

This algorithm monitors the movement of storm cells by matching storms from current volume scan to the next volume scan.

Storm position forecast

This algorithm will predict the future centroid locations based on a history of their movements. It computes the forecast movement using the linear least squares extrapolation of the storm's previous positions.

Siddhi can be used to identify the storm cell segments where it basically compares the reflectivity values with specified threshold value (see Appendix A) [37] [41].

5.3.2. Geo-dashboard in Siddhi

Geo-dashboard is introduced by WSO2 and it can be used to process spatial data. Users can see the movements of spatial objects in real-time with their rotations in the space. Spatial object data is transferred as a Geojson[] object from the CEP websocket output adapter to the web browser in the CEP geo-dashboard. Geojson point is used to place a marker on the map. This point can be used to update the location of the incoming events. They have identified several features of this geo-dashboard such as speed alerts, proximity alerts, geo-fencing and history playback. Proximity alerts are generated when two or more objects become close in their proximity by a predefined value from the user as seen in Figure 5.2.

This geo-dashboard can be applied in a weather monitoring system. Movement of rain or a wind can be simulated and it can be used to detect the sudden weather changes in nearby by places [42].

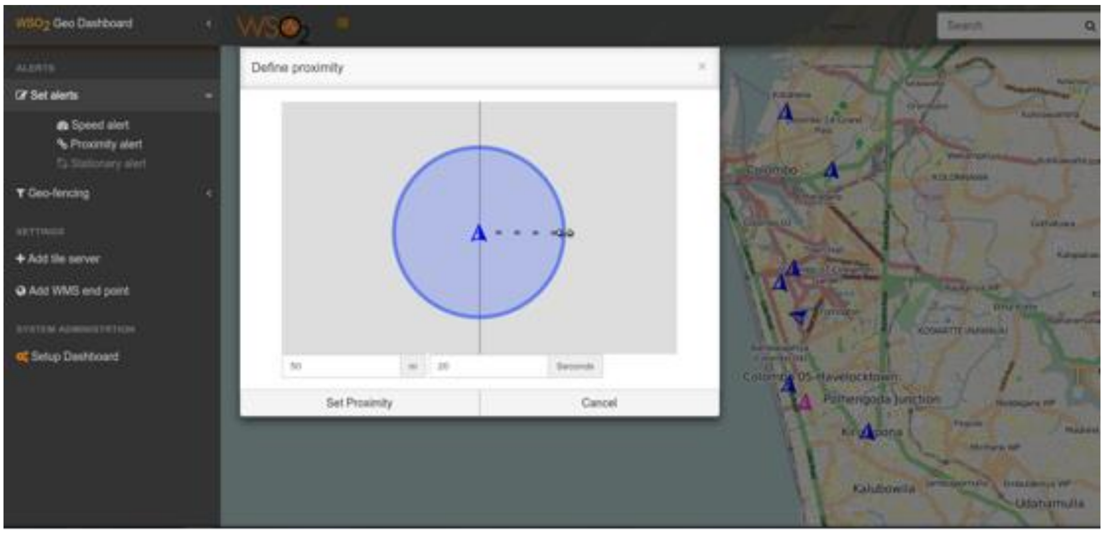


Figure 5.2 Geo-dashboard.

6. REFERENCES

[1] 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),* vol. 8, no. 15, Oct. 2013.

[2] Coordinating Secretariat for Science Technology & Innovation, "Developing a national climate observatory system for Sri Lanka," Nov. 2015.

[3] N. Mluseux, J. Mattioli, C. Laudy, and H.Soubaras, "Complex event processing approach for strategic intelligence," In Proc. *FUSION 2006, no 200*, July 2006.

[4] S. Perera. (2011, Dec 05). *A second look at complex event processing* [Online]. Available: http://www.slideshare.net/hemapani/Siddhi-a-second-look-at-complex-event-processing-implementations

[5] L. J. Fulop, G. Toth, R. Racz, J. Panczel, T. Gergely, and A. Beszedes, "Survey on complex event processing and predictive analytics," July, 2010.

[6] J. Dunkel, "On complex event processing for sensor networks," In Proc. *9th IEEE International Symposium on Autonomous Decentralized Systems (ISADS),* pp. 249-255, Athens, Mar. 2009.

[7] Q. Zhou, Y. Simmhan and V. Prasanna, "On using semantic complex event processing for dynamic demand response optimization," In Proc. *CORR abs/1311.6146*, 2013.

[8] K. Tailor and L. Leidinger, "Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks," In Proc. *8th Extended Semantic Web Conference,* pp. 285-299, 2011.

[9] J. P. Calbimonte, H. Jeung, O. Corcho and K. Aberer, "Semantic Sensor data search in a large scale federated sensor network," In Proc. *4th International Workshop on Semantic Sensor Networks,* pp. 14-29, 2011.

[10] J. P. Calbimonte, H. Jeung, O. Corcho and K. Aberer, "Enabling query technologies for the semantic sensor web," *International Journal on Semantic Web and Information Systems,* pp. 43-63, 2012.

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

[12] S. Suhothayan , I. S. Narangoda, S. Chathuranga, K. Gajasinghe, S. Perera, and V. Nanaykkara, "Siddhi: A second look at complex event processing architectures," In Proc. *IEEE Gateway Computing Environments Workshop (GCE),* 2011.

[13] “Implementing DEBS grand challenge using WSO2 Siddhi CEP engine,” unpublished.

[14] 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.

[15] K. K. Droegemeier, "Transforming the sensing and numerical prediction of high-impact local weather through dynamic adaptation," In Proc. *Phil. Trans. R. Soc. A (2009) 367,* pp. 885–904, Dec. 2008.

[16] N. N. Vijayakumar and B. Plale, "Tracking stream provenance in complex event processing systems for workflow-driven computing," In Proc. *EDA-PS Workshop,* 2007.

[17] X. Li, B. Plale, N. Vijayakumar, R. Ramachandran, S. Graves, and H. Conover, "Real-time storm detection and weather forecast activation through data mining and events processing," *Earth Science Informatics* , vol. 1, no. 2, pp. 49-57, 2008.

[18] B. Pale et al., *"CASA and LEAD: Adaptive cyber infrastructure for real-time multiscale weather forecasting," IEEE Computer,* vol. 39, no. 11, pp. 56-64, 2006

[19] M. Zink, E. Lyons, D. Westbook, J. Kurose, and D. Pepyne, "Closed-loop architecture for distributed collaborative adaptive sensing of the Atmosphere: Meteorological Command & Control," *International Journal of Sensor Networks(IJSNet), InderScience,* 2007.

[20] 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,* vol. 57, no. 16, pp. 3235-3248, Nov. 2013.

[21] K. Hondl, "Capabilities and components of the Warning Decision Support System - Integrated Information (WDSS-II)," In Proc. *American Meteorological Society Annual Meeting,* Long Beach, 2003.

[22] V. Lakshmanan, "The warning decision support system – integrated information," *Weather and Forecasting,* vol. 22, no. 3, pp. 596-612, 2007.

[23] N. N. Vijayakumar, Y. Liu and B. Plale, "Calder query grid service: Insights and experimental evaluations," In Proc. *CCGrid Conference,* 2006.

[24] N. N. Vijayakumar and B. Plale, "Towards low overhead provenance l in near real-time stream filtering," In Proc. *IPAW,* 2006.

[25] N. N. Vijayakumar, B. Plale, R. Ramachandran, and X. Li, "Dynamic filtering and mining triggers in mesoscale meteorology forecasting," In Proc. *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'06),* Denver, Aug. 2006.

[26] Wikipedia contributors. (2015, Sep). *Weather* [Online]. Available:

https://en.wikipedia.org/wiki/Weather.

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

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

[29] F. Lalaurette, "Early detection of abnormal weather conditions using a probabilistic extreme forecast index," *Quarterly Journal of the Royal Meteorological Society, 129,* pp. 3037- 3057, Mar. 2006.

[30] H. R. Glahn and D. A. Lowry, "The use of Model Output Statistics (MOS) in objective weather forecasting," *Journal of Applied Meteorology,* vol. 11, pp. 1203-1211, 1972.

[31] M. Hayati and Z. Mohebi, "Temperature forecasting based on neural network approach," *World Applied Sciences Journal,* vol. 2, no. 6, pp. 613- 620, 2007

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

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

[34] J. Duda, "How to use and interpret Doppler weather radar".

[35] D. Dutta et al., "Nowcasting of Yes/No rain situations at a station using soft computing technique to the radar imagery," *Indian J Radio & space phys,* Apr. 2010.

[36] U.S. Department of Commerce National Oceanic and Atmospheric Administration, 2006: Doppler Radar Meteorological Observations. Federal Meteorological Handbook, 11.

[37] Johnson et al., "The storm cell identification and tracking algorithm: An enhanced WSR-88 D Algorithm," *Weather Forecast (USA),* 1998.

[38] BlueMM. (2007, Jan). *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.

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

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

[41] A. S. Lanpher, "Evaluation of the storm cell identification and tracking algorithm used by the WSR---88D," 2012.

[42] WSO2 team. (2015, Jan 21). *Geo spatial data analysis using WSO2 complex event processor* [Online]. Available: http://wso2.com/library/articles/2015/01/article-geo-spatial-data-analysis-using-wso2-complex-event-processor-0/.

APPENDIX A: STORM CELL SEGMENTS

```
BEGIN ALGORITHM (STORM CELL SEGMENTS)
1.0 DO FOR ALL (radials of the elevation scan)
     1.1 DO FOR ALL (THRESHOLDS (Reflectivity))
     1.1.1 DO FOR ALL (SAMPLE VOLUMEs of the current radial)
     1.1.2 IF (REFLECTIVITY FACTOR(Sample Volume) is greater than
                or equal to THRESHOLD (Reflectivity))
            THEN
     1.1.2.1 Begin or continue POTENTIAL CELL SEGMENT
     1.1.2.2 IF (Beginning POTENTIAL CELL SEGMENT)
              THEN
     1.1.2.2.1 COMPUTE (beginning RANGE(Segment))
              END IF
     1.1.2.3 COMPUTE (ending RANGE(Segment))
     1.1.2.4 Reset NUMBER OF DROPOUTS to zero.
     1.1.3 ELSE IF (REFLECTIVITY FACTOR(Sample Volume) is greater
                than or equal to (THRESHOLD (Reflectivity) -
                THRESHOLD (Dropout Reflectivity Difference)) AND
                (continuing POTENTIAL CELL SEGMENT))
            THEN
     1.1.3.1 COMPUTE (NUMBER OF DROPOUTS)
     1.1.3.2 IF (NUMBER OF DROPOUTS is greater than
                  THRESHOLD (Dropout Count))
                  THEN
     1.1.3.2.1 End POTENTIAL CELL SEGMENT
              END IF
     1.1.4 ELSE IF (Continuing POTENTIAL CELL SEGMENT)
                THEN
     1.1.4.1 End POTENTIAL CELL SEGMENT
            END IF
     1.1.5 IF (POTENTIAL CELL SEGMENT is ended)
             THEN
     1.1.5.1 COMPUTE (LENGTH(Segment))
     1.1.5.2 IF (LENGTH(Segment) is greater than or equal to
                THRESHOLD (Segment Length(Reflectivity
                Threshold)))
                THEN
     1.1.5.2.1 Label POTENTIAL CELL SEGMENT a CELL SEGMENT
              END IF
          END IF
       END DO
     END DO
     1.2 DO FOR ALL (THRESHOLDS(Reflectivity))
     1.2.1 DO FOR ALL (CELL SEGMENTS for this THRESHOLD
                (Reflectivity))
     1.2.1.1 COMPUTE (maximum REFLECTIVITY FACTOR(Segment))
     1.2.1.2 COMPUTE (MASS WEIGHTED LENGTH(Segment))
     1.2.1.3 COMPUTE (MASS WEIGHTED LENGTH SQUARED(Segment))
     1.2.1.4 COMPUTE (NUMBER OF SEGMENTS(Reflectivity Threshold))
     1.2.1.5 WRITE (maximum REFLECTIVITY FACTOR(Segment))
     1.2.1.6 WRITE (MASS WEIGHTED LENGTH(Segment))
     1.2.1.7 WRITE (MASS WEIGHTED LENGTH SQUARED(Segment))
     1.2.1.8 WRITE (beginning RANGE(Segment))
     1.2.1.9 WRITE (ending RANGE(Segment))
     1.2.1.10 WRITE (AZIMUTH)
     1.2.1.11 WRITE (THRESHOLD (Reflectivity))
           END DO
     1.2.2 WRITE (NUMBER OF SEGMENTS(Reflectivity Threshold))
     END DO
END DO
2.0 COMPUTE (average DELTA AZIMUTH)
3.0 WRITE (ELEVATION)
```

```
4.0 WRITE (average DELTA AZIMUTH)
END ALGORITHM (STORM CELL SEGMENTS)
```

Beginning RANGE(Segment)

RSbeg = SVbeg*SVL – SVL/2

Where,

RSbeg = The beginning RANGE(Segment), the RANGE(Slant) to the front (closest to the radar) of the first *sample volume* of a *cell segment*, in km

SVbeg = The first *sample volume* of a *cell segment*

SVL = The length (in slant range) of a sample volume in km

Ending RANGE(Segment)

RSend = SVend*SVL + SVL/2

The ending RANGE(Segment), the RANGE(Slant) to the back of the last *sample volume* of a *cell segment*, in km.

Where,

SVend = The last *sample volume* of a *cell segment*

Number of DROPOUTS

ND = ND + 1

The number of contiguous *sample volumes* with a *reflectivity factor* less than the *threshold* (Reflectivity) by less than or equal to the *threshold* (Dropout reflectivity difference)

LENGTH (Segment)

LEN = RSend – Rsbeg

Maximum REFLECTIVITY FACTOR (Segment)

$$\mathrm{DBZE}avg_k = [\sum_{j=k-INT(\frac{RA}{2})}^{j=k+INT(\frac{RA}{2})} DBZE_j]/RA$$

$$DBZEmax = DBZEavg_k \text{ if } DBZEavg_k \geq DBZEmax$$

Where,

DBZE = The REFLECTIVITY FACTOR(Sample volume), the effective radar reflectivity factor of a sample volume, in dBZe

DBZEavg = The average reflectivity factor of a group of *sample volumes*, in dBZe.

RA = The REFLECTIVITY AVERAGE FACTOR, the number of *sample volumes* used for determining the maximum (average) reflectivity factor (3).

INT is a function whose magnitude is the largest integer that does not exceed the magnitude of the argument. Index j is constrained to the interval [SVbeg, SVend].

MASS WEIGHED LENGTH (Segment)

The mass density weighted length of a *cell segment*, in kg/km^2

$$\mathrm{MWL} = \sum_k [\,(\mathrm{MSV}_k)(\mathrm{RS})]$$

Where MSV = (MWF)(PIN)

Where PIN is computed from the relation ZE = (MMF)(PIN)$^{\mathrm{PIE}}$, and ZE = 10$^{(\mathrm{DBZE}/10)}$

If DBZE > MRM, DBZE = MRM, in km.

Where RS = RANGE (Slant), the slant range to the center of a *sample volume*

MWF = MASS WEIGHTED FACTOR, a factor used in computing the mass of a *sample volume* (53 x 10^3), in (hr)(kg)/(km^{2}m^2).

MMF = The MASS MULTIPLICATIVE FACTOR