A DATA EXCHANGE MESSAGING SYSTEM FOR INTERNET OF THINGS SYSTEMS

Moyosore Akinrinmade, Nalini Venkatasubramanian, Praveen Venkateswaran, Phu Nguyen, Kuo-lin Hsu
University of Texas San Antonio, Department of Computer Science, Center for hydrometeorology and Remote Sensing,
University of California Irvine

Introduction

- Internet of things(IoT) is an internetworking of different devices, so there is a collection and exchange of data through the network.
- There is no way for the data from these sources to collaborate so we designed a software that will enable collaboration and incorporation of these data sources.
- Our goal is to make a software that will incorporate data from hydro meteorological instruments like rain gauge, sensors etc. to assist hydro meteorological decisions like rainfall, flooding, etc..
- Also since it's a lot of data coming in we want the users or applications to be able to take a portion of the data without looking at the whole data .

System Architecture

Figure 1: A diagram of data exchange system flow

- The producers are the data sources e.g. rain gauge, water sensors, rainfall satellite data, data from users etc.
- The consumers are the users or applications that receive data from the producers.
- The Kafka brokers are the servers that run Kafka . Used by producers to publish data into topics. Used by consumers to pull data from topics.
- The consumers are the users or applications that receive the data from the producer based on their preferences.
- The zookeeper is used for managing the system. It notifies the publisher and subscriber of a new message in the system.

Methodology

The Kafka producer and consumers classes are designed using Java on Eclipse IDE

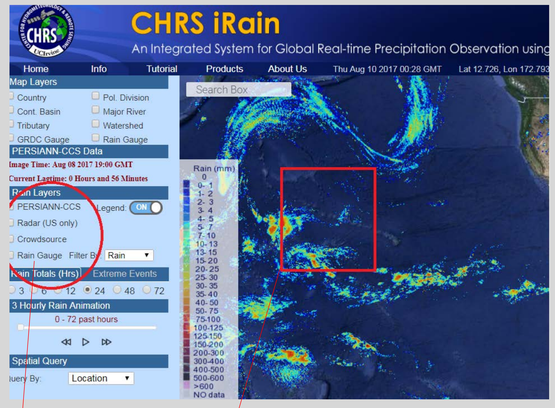
ProducerFile.java: Producer Class

1. Create a Java project and class in Eclipse
2. Save the data file in the project created
3. Import all the files needed for both Kafka and the reader
4. Open the data file and using a while loop read the data in the file, line by line into an array
5. In the while loop use the split command to split each line and put the values in a 2-dimensional array
6. Calculate coordinates of each data in the data file using the yllvalue and xllvalue given to us at the beginning of the file
7. The first coordinates are (xllvalue, (yllvalue + rows*cell size)); cell size and number of rows is given at the beginning of the file
8. Increase the xllvalue by cell size to get the next x coordinate and the (yllvalue + rows*cell size) by cell size to get the next y coordinate
9. save the coordinates as the key and the data at the coordinate as the value
10. Send the values to the consumer using the Kafka command

ConsumerFile.java: Consumer class

1. Create a Java project and class in Eclipse
2. Use the Java scanner class to ask the user for the topic they will like to connect to
3. Save the value entered in a variable and use it to connect to the topic using Kafka commands
4. Ask the user for the starting and ending coordinates, they would like to get the values
5. Calculate the number of rows and columns using rows= (y2-y1)/cellsize, column= (x2-x1)/cellsize
6. Use a for loop to generate these values and save them into a 2-dimensional array
7. Using a while loop, receive the values from the producer class and save it into record
8. In the while loop, split record.value and save the values in an array
9. Use a for loop to go through the coordinates saved in the array above, use an if statement to check if the coordinates matches with the one sent through the topics by the producer and if it matches print the coordinate and the value.

iRain Description



Data Sources (Producers)

Bounding box to show the beginning and ending coordinates of the area a user is interested in getting the rainfall values.

Figure 2 : iRain is a website designed by the center of Hydrometeorology and remote sensing UCI and it is used as the test case for the research

Results

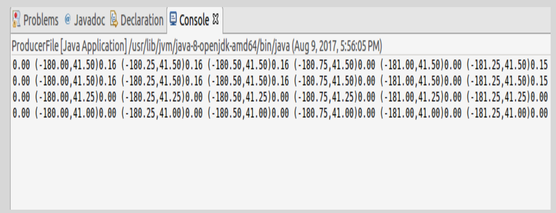


Fig 3: A screen-shot of the producer sending coordinates and the values at the coordinates to the topic broker

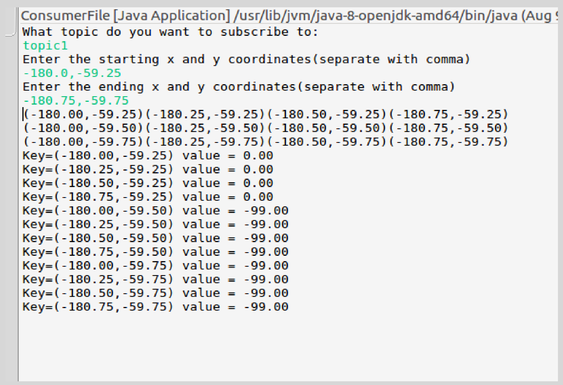


Fig 4 : A screen-shot of the consumer program generating coordinates from the bounding box given by the user and using the coordinates generated to find and print out the values at the location

Conclusion

- In conclusion, the primary purpose for the research was achieved. We created a data exchange that is faster and more efficient.
- We were able to create a system where applications can receive their data directly from the data sources without having to download the file containing the data each time they needed data.
- Users can now request only the data they need and do not need to go through the entire data file to get a section.

Future Work

- Producers and consumers should be able to receive data in real time.
- Consumers should be able to subscribe to more than one topic at a time and receive data from them at the same time or concurrently.
- Producers should receive notifications when there is a new consumer that subscribers to the topic they are sending data to .
- Consumers should receive notifications when there is a new message from the Kafka broker(server).

Acknowledgements

- Dr. Nalini Venkatasubramanian for her mentorship
- Dr. Kuo-lin Hsu, Dr. Phu Nguyen and the center for hydrometeorology and remote sensing for the resources provided and their mentorship
- Praveen Venkateswaran for his mentorship and help through out the project
- The IoT-SITY REU directors for the opportunity to take part in this research
- University of California, Irvine and Donald Bren Hall for the space and resources