Pi in the Sky

Whitney Nelson
Faculty Advisor: Mr. Bruce Chittenden
Department of Computer Science
Hampton University
100 East Queen Street Hampton, VA 23668
whitney.nelson@my.hamptonu.edu

ABSTRACT

Weather data collection and analysis is vital for earth climate studies. Approached with the problem of collecting and organizing weather data at low cost, we decided to use the credit card sized Raspberry Pi (R-Pi) computers to tackle the problem. R-Pi clusters are flexible to a) collect weather data via sensors, b) host the obtained weather data on a cloud, c) organize the weather data on a database and d) host a web server that can provide access to the weather data. While our team worked on the overall project, my role was to generate scripts for the user-server-database interactions. When a user requests data to the server, the request is serviced by a series of scripts performing different tasks. The first task is to create a connection to the database, where the weather data is stored. The second task is to transform the user request into an appropriate query based on user selection. Then, the query generated, after passing a validity test, is sent to the database server. Upon receiving the data from the database, it is again checked for validity. The final task is to format the received data for displaying as plot or tables.

CCS Concepts

• Computer systems organization → Embedded and cyber-physical systems → Embedded systems → Embedded software
• Computer systems organization → Architectures → Distributed architectures → Cloud computing

Keywords

Raspberry Pi; Cloud Storage; Weather Arduino; Database; Server

1. INTRODUCTION

The dilemma that often arises within Supercomputing is cost. One specific area that has been effected by the cost of supercomputing is weather analysis. Scientist use climate trends from geographical places all around the world to predict weather and protect the earth. Supercomputers have allowed scientist to collect massive amounts of data which help to spread knowledge about hurricanes, earthquakes, and other natural disasters. The "Pi in the Sky" project was designed to create weather stations at a low cost.

The project was first broken up into three different parts: Web Service, Cloud Storage, and Sensor. The sensors were used to read in the appropriate weather data. Once the weather data is read, it is then uploaded and remotely stored via cloud storage. This is where all the data is stored. A database is then synchronized with the cloud. As soon as data is uploaded from the sensor and sent to the cloud, it is automatically uploaded to the database. Once in the database, the data is then organized into tables. This organization allows the data to be properly indexed based upon a user request. All the code was then put on the web page and properly adjusted for a friendly user interface. The web page was then further developed to make the user's experience as pleasant as possible.

ADMI 2016, March 31-April 3, 2016, Winston-Salem, NC, USA.

2. RASPBERRRY PI

The very first step that had to be completed before building a functional Weather Station was to obtain the appropriate computer that is inexpensive but still able to handle the task at hand. It was decided that the Raspberry Pi 2 Model B (showed below in Figure 1), a low costing credit card sized computer should be used because of its four core processor with one GB RAM. The Raspberry Pi is a single board computer comprised of the following key components: four quad USB ports, Ethernet port, HDMI port, Micro USB, and 40 General Purpose Input Output pins (GPIO pins). The model cost was approximately $35. It also has a Linux based operation system (Debian). There was also a fan which was used to maintain cooling of the Raspberry Pi.

Figure 1. Raspberry Pi 2 Model B

3. SENSORS

3.1 Weather Pi Arduino

An applicable sensor module was to be determined in the same regard of cost and capabilities. The goal is to create an array of sensors using an I2C interfaces. The I2C must connects to every sensor of the Weather Station. There will also be an indirect

access created between the sensor module and the Raspberry Pi with the following common ports:

VCC -> power
GND -> ground
SCL -> Serial Clock
SDA -> Serial Data

The Weather Pi Arduino is a $25 weather board with the capability of interfacing with Raspberry Pis and Arduino micro-computers. It was developed by SwitchDocs Labs, low powered, and contains proper plugs for external sensor inputs.

3.2 Assembly of Weather Station

The Weather Pi was connected to the Raspberry Pi through the General Purpose Input Output (GPIO) pins. The other components were then soldered onto the Weather Arduino board. The Real Time Clock/Electrically Erasable and Programmable Read Only Memory (EEPROM) accurately keeps track of seconds, minutes, day, date, month, and year. It contains battery input for accurate timekeeping without main power, automatically adjust for months less than 31 days and leap year. The HTU210-F sensor (Humidity, Temperature Sensor) was the first sensor to be soldered onto the board. It has a typical accuracy of ±2% and is able to collect temperature, pressure, altitude, and humidity.

The MOD-1016 sensor (Lightning Sensor) was soldered next. This sensor can detect storms front up to 40km away and are able to distinguish if storm front are moving closer or further away. The WeatherRack-Anemometer/Wind Vane/Rain Bucket measures wind speed, wind direction, and rainfall. It required additional analog to digital converter board for operation and uses seal magnetic reed switches. The 12Bit 12C ADC+PGA ADS1015 (Analog to Digital Converter) was necessary for the WeatherRack. It has high resolution analog to digital conversion and can perform conversions at rates up to 3300 samples per second. It also includes programmable gain amplifier, which can help boost small signals to utilize the fill 12-bit range.

The Humidity Temperature Sensor and Lightening Sensor communicate to the Raspberry Pi through the I2C interface. The measurements for each weather data used is listed in Table 1 and the total cost of the Weather Station was approximately $200.

Table 2. Measurements for Weather Data

Weather Data	Measurement
Temperature	Celsius
Pressure	Kilopascals
Altitude	Meters
Humidity	Percentage
Lightning Detection	Counts number of strikes within a time window
Rain Bucket	Inches
Wind Vane	Directional Degrees
Anemometer	Miles per hour (MPH)

3.3 Sending Data to the Raspberry Pi

Every script for the sensors was written in python except for the Weather Rack. The Weather Rack was written in C. Once proper scripts were obtained, the data gathered is then sent to the Raspberry Pi to display the proper data category, measurement, date, and time.

4. CLOUD STORAGE

4.1 Owncloud

A cloud provides storage of information over an internet platform. The advantages of cloud storage is the reduction in the amount of unused computers and materials. It also allows easier access across the board. When engaging in cloud computing there are three attributes that create the SPI model: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS is the service provider which is made available to the client. PaaS is the service that is used to build off of. IaaS is the service includes the hardware, storage, and network components. Owncloud deemed to be the most appropriate cloud storage to use because unlike other cloud storage options like Dropbox, it is a self-hosted share server. Owncloud remotely stores data and the information that is uploaded can be accessed by any pi within the cluster and it is free. Owncloud version 5.0 was download and produced one error reporting that the .htacess files does not work. Once the default-ssl and 000-default files were changed. In testing the Owncloud we have determined that the current releases are broken.

Release 8 – Broken
Release 7 – Broken (later works)
Release 6 - Broken
Release 5 – Works

We did not attempt to debug the current releases, but we waited until releases were updated and then upgraded our Clouds to version 7.0 which was much more stable.

4.2 Building the cloud storage

A python library called Pyocclient was used to push files from the Raspberry Pi to the cloud. It was then remotely synced with a database on all the disk of the Raspberry Pis. Per each sensor module it can store up to eight kilo bytes per hour and 70.08 megabytes per year. The interface transfer rate ran up to five gigabytes per second (USB 3.0). Owncloud can hold up to 1000 more sensor modules and holds the capacity to store up 14 years of data. Once the Owncloud was loaded with data, the database MYSQL was imported into the python scripts so that the database could be fully populated.

5. WEB SERVICE

5.1 Setting up the Server

The Web Service is used to display the information that is to be collected. In order for each pi to "talk" to one another or share information the appropriate Message Passing Interface (MPI) must be installed. MPICH2 was downloaded for it seemed most compatible with the Linux operating system. Each pi was set up with its on individual hostname and IP address with each address connecting to the router and one another. The wireless connection was then established through the network interfaces directory. Next an apache server was downloaded in order create a directory for the web service. Hypertext Preprocessor (PHP) was then downloaded for editing and creating a particular display. This established a functional Raspberry Pi with wireless internet access that could communicate with the other pi and create files. The information must be held in a database, therefore both SQL and MYSQL was downloaded with the intention of having the information being plotted through Python.

5.2 Development of Web Service

In order to create an interaction between a client and server, a web interface must be created. The data must be viewed in a neat an attractive manner. The most appropriate way to display weather data is in graphs and tables. There also is handling that is done in case of error in the sensors. Open source software is used to include more modules and create an interactive user experience.

Lustre RSF, Drupal, and Anaconda are open sources software that were all considered to download and use. Lustre development became stuck. There was a discussion about which version of Debian supports Lustre, whether it was Debian Stretch or Debian Wheezy, the final belief was that it is Wheezy. Anaconda was then found that it cannot be ran on Raspberry Pi. Therefore, Drupal became the choice for our web server. Python was considered in developing the pages, so Numpy, Scipy, and Matplotlib were all downloaded but were found incompatible with Drupal. Altogether, it was decided that an open source software would not be used and web pages would be made with the Apache server. Drupal was used through the project but prosed several implications of unreliability.

PHP and Hypertext Markup Language (HTML) are the two languages used to extract the data based up user request. HTML works on the client side of this process which describes the browser. It is the display of the page and works as an interface that the user interacts with. When the user fills out the entire form and the submit button is hit, the HTML code is sent to the PHP code to be processed. PHP is intellectual programming language that works on the server side of this process.

First a connection between the code and the database must be made. It begins by asking the MYSQL database for permission to use its information. The username, IP address, password, database name, and table name are required information that are entered to access the database information. After the connection to the database is made, the database is queried according to the web server request. The code checks the database for the requested information and, if found, gathers all of the necessary data. Finally, the data is displayed in forms of tables (displayed in Figure 2) and graphs that display the accurate results based off the user's request.

Temperature	Pressure	Altitude	Date	Time
23	81454	1804	20150723	1250
23	81448	1804	20150723	1251
23	81446	1804	20150723	1252
23	81450	1804	20150723	1253
23	81454	1804	20150723	1254
23	81451	1804	20150723	1255
23	81447	1804	20150723	1256
23	81448	1804	20150723	1257
23	81450	1804	20150723	1258

Figure 2. Database Sample

5.3 Error Handling

The Weather Pi's sensors sometimes show error in the results by skipping a period time of data. In this case, if the user were to pick a time that was not in the database, an empty graph or table would appear. To avoid this problem, code was written for error handling which would still be able to display some sort of table or graph. When the user makes a request and results are not in the databases, a new query is then called that goes to the nearest time available therefore, displaying a table or graph for the nearest time.

5.4 Graphing

JP Graph is a graphing tool that is written in PHP and works with MYSQL. The queries that were made for the tables were used in the exact same manner for the graph. There were also graphs made that display each Weather Parameter for the current day and the last hour of data.

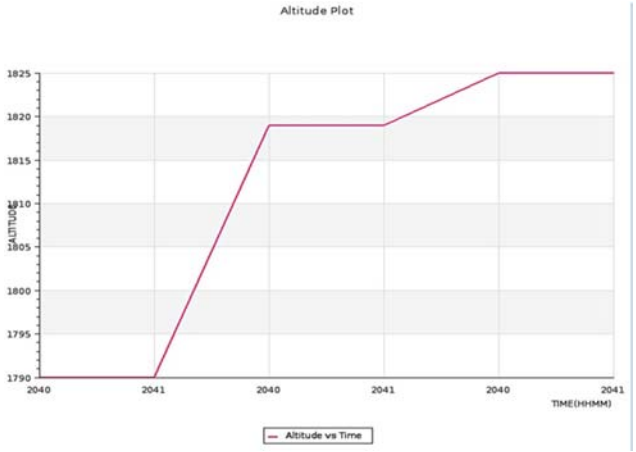

Figure 3. Graph using JP Graph

6. Overall Conclusion

All the scripts worked together to create the "Pi in the Sky" web page and were then adjusted for a friendly user interface. The Weather Stations were efficient but did not have as much flexibility in the software that could be downloaded onto it. At times the Sensors or Weather board itself could be unreliable by skipping over a specific amount of data. This was handled within the web interface scripts. Although Owncloud only held one functional version, Owncloud 7.0.1 was able to appropriately store all the data that was received from the sensors. The database was able to be utilized for all the necessary uses. Open source software seemed to be difficult to use however the Apache server was fully able to display appropriate results with charts and tables. Altogether, the project was able to effectively create a low costing Weather Stations.

7. ACKNOWLEDGMENTS

Special thanks are given to the contributors that made this project possible: National Center for Atmospheric Research, Dr. Richard Loft, Dr. Raghu Raj Prasanna Kumar, Mr. Bruce Chittenden, Rashmi Oak, Harish Ramachandran, Priyanka Sanghavi, Amogh Simha, Ian Bragg, Gaston Seneza, and Jenish Koirala.

8. REFERENCES

[1] Rousse. 2012. Reasoning about naming systems. *What is SPI model (SaaS, PaaS, IaaS)? - Definition from WhatIs.com.* (Feb. 2012). DOI=http://searchcloudcomputing.techtarget.com/definition/SPI-model.

[2] Breckenridge, C. 1996. *A Tribute to Seymour Cray*. SRC Computers, Inc. (Nov. 1996). DOI=https://www.cgl.ucsf.edu/home/tef/cray/tribute.html