Control Enhancements in Existing Temperature Monitoring System

Andres Rivero
Fermi National Laboratory - Computing Section
Indiana University
arivero@umail.iu.edu

Abstract

Fermilab performs active monitoring of temperatures in their data centers using a custom developed software solution. The key feature of this software is to alert appropriate personnel when a temperature in a data center goes above or below a defined threshold. In addition, the software provides reports to allow facilities engineers to see trends. This paper describes the modifications made to enhance this software. These improvements provide Fermilab the ability to easily adjust the control temperature thresholds, as well well as extract data for analysis more effectively. These tools are now functioning in a production environment.

I. Introduction

Maintaining optimal ambient temperatures in a data center poses a challenge that must be addressed in order to protect computers containing data. The computers must remain at a certain temperature to avoid damaging the circuitry. As a consequence, Fermilab has developed systems that alerts the department of any anomalies in the temperature such as LMsensors and Metasys. These systems are in place to alert, however a visual aid was necessary to monitor the changes in trend along a day. To solve this, under the supervision of Dr. David Ritchie a system called the Temperature Monitoring System was developed. This system displays the temperatures at the remote data center location.

The project was a success. In addition to the visual element, the software combined technology from the other running systems that alert the department whether the temperature dropped below or surpassed a threshold. In the event that sensors malfunction, the alerting system is configured to notify the team until actions are taken. To prevent false positives, the department requested control tools to further enhance the existing monitoring system.

The current system lacks an intuitive and easy way to change threshold configuration data, a lack of an easy way to extract data for reporting, and a crude system to display trending data. The upgrades I performed addressed these problem in the current system.

This paper describes the methods used to solve these problems. First, I designed a simple GUI with enough functionality to meet the basic requirements and shared this with the end users. After collecting user feedback, I then enhanced the GUI and developed the code to perform the necessary tasks. I explored new technologies to solve these problems (e.g.. OpenpyXL), which lead to the successful development of these solutions.

II. Methods

The investigations to improve the monitoring system were divided into three sections. Each section explains its own methodology and obstacles encountered in order to reach a final conclusion. The sections are divided as follow: First, we explored the addition of features that allowed user-friendly manipulation of thresholds. Second, we investigated third-party libraries that could contribute with the implementation of an efficient export-

ing tool. Last, we examined the integration of Google Charts to produce graphs to modernize the aesthetics of the website.

I. Threshold Editor

To determine whether this part was feasible, I considered the specifications of the software. Since we had no physical access to the server, we used Microsoft Remote Desktop to connect remotely. Lastly, to explore what was the best approach, we gained admin privileges to the source code.

The temperature monitoring system runs in a Windows Server 2008 environment. The temperature monitoring system has services that run every 5 minutes to collect sensor information, and update the HTML files that present this information to the user. To begin, it was necessary to request proper permissions to access the source code. In addition, it was necessary to install packages in my environment (MacOS) that allowed connection to this server. These steps were not trivial, however once this was accomplished I was able to study the source code and begin to make progress on these enhancements.

To offer a solution to the stated requirements, I studied the development of graphical interface modules. The preferred library, Tkinter, is a package included with the Python compiler. Before designing the solution, I met with stakeholders to discuss various ideas. It was concluded that to build a functional tool, only five widgets from the TKinter python library were necessary. I used a drop down menu to give the user a choice of data center rooms to display. A radio button was used to select the hot or cold aisle to work with. Labels were chosen to describe the adjacent values. An input box was utilized for user input of threshold values, with buttons to allow for incrementing and decrementing the displayed value in those input boxes. Finally, a buttons are supplied to submit the values as well as clear the values.

The implementation process consisted of backing up the original system, installing the new software, and verifying correct functionality. Verification was accomplished by testing each component of the software using a test plan. When the software executed, I verified that the initial display values were correct. Next, I modified the threshold value and verified that the changes were made correctly. A final check was accomplished by examining the configuration file and verifying that the values were correct.

II. Excel Exporter

There are many third-party python modules to handle Excel sheets with python. The best option was OpenpyXL. This module works with Excel 2007 and beyond allowing the use of more rows and columns. The server, computer and way to access the source code was a constant. However, the development of this new tool required deeper understand of the data. Thus, I became acquainted with the monitoring system.

Rather than copying data manually to a spreadsheet,a method was devised to extract the data using a python script. The first steps consisted of creating a user-friendly GUI that contained all widgets needed to fulfill the requirements. Two drop down menus that displayed the room and type of sensor (hot, cold or all of them). Plus, three input boxes that allowed feeding a begin and end date, and the name of the Excel file. The "Save" button performs the magic (via the OpenpyXL library) to transfer the text data to an Excel spreadsheet.

The test suite was basic and simple. The first test was to verify that the initial data was displayed correctly by the GUI. It was important to verify that the correct list of sensors was displayed for a chosen room. To test this, I selected each room individually and verified that the results contained the sensors for that room.

III. Graphing Script

To improve the previous bar charts, we explored Google Charts to improve the presentation. The previous module used HTMLgen to build an HTML file on the fly. However, Google Charts used javascript to construct and draw column charts, as opposed to bar charts. In order to start exploring the advantages of this new technology, we activated javascript in the development machine. This was necessary for the graphs to load properly in the browser.

Since javascript was a necessity, I wrote a template file that built a simple HTML file, but also created a script to plot the graphs. The file consists of constant commands that Google Chart recognized. The data for the daily temperature of the sensor had to be set in the script. In addition to the previous graph, we decided to add a script on top of the page that allows a user to choose a seven or thirty days average. The HTML code consisted of only place holders for the graph and control filters.

To include data in a template, I implemented a module that read the template and wrote the daily data in its corresponding location. As mentioned previously, there was a location for the previous seven days, and also, for the previous thirty days. Given the template, the library copies each line in the template to a HTML file until it meets the criteria to copy the data instead. The module was in charge of averaging the temperatures at the given time slots the seven and thirty days previous to the current date. This module runs independently from other libraries the system used. But, we had to import it as to allow the webpages to be created when the system was scheduled to run.

To show that the software was graphing correctly, one had to compare the new graphs with the existing graphs. In addition, one could actually read the raw sensor data and verify that against the graphs. To check, that the averages were correct, I had to calculate the mean for random time slots throughout the previous n days.

III. Results

Most of the testing results consisted of running scripts that would verify correctness. Each script performed a different function and verified something completely unrelated to all other tools. The results confirmed that the software is performing as requested.

The threshold editor performs basic error checking, insuring that the data entered is reasonable (i.e. no characters, high threshold > low). When temperature configurations are modified via this tool, an email notification is sent to each member of the Facilities team. Thorough testing has provided results that prove that the error checking is functioning as designed.

When entering dates, validation is done to verify the correct date format. In addition, the selected dates must be logical (i.e. Start date must be before End date). If an incorrect value is supplied, the software shows a warning message and fills in default values in the appropriate fields. Thorough testing was performed that verified that the software is performing correctly.

The Excel export tool was developed assuming the sensor data was correct. One of the unexpected outcomes of developing this tool was demonstrating that certain sensors were missing blocks of data. This was unknown to the Facilities Engineering group. The cause of this missing data is currently under investigation. To work around this data integrity issue, software changes are being developed at this time in the export tool to handle this case.

Verifying the correctness in the graphing tool was easier to visualize. It has been shown

the new python module is displaying the information correctly and improving the use of space in the webpage. The calculations to compute the mean values has been manually verified. This code is currently in the development environment waiting on approvals by the stakeholders to move to production.

The Threshold Editor tool was successfully deployed to the production environment on 2014/07/15, and is now used to make changes to threshold configurations.

IV. Future Work

To further advance the system, we brainstormed what upgrades could benefit the software. The current system uses a static approach and has no correlation between website and code. This presents obstacles that make certain enhancement difficult to implement. To address this, we plan to move the data collected to a database. Moreover, we intend to use new technology to develop a dynamic software system that allows user interaction. Unfortunately, there was no time to do any work outside of planning, however, the experiences gained this summer will make this possible in the near future.

V. Conclusion

It is critical that data centers operate within a range of acceptable ambient temperatures. Fermilab uses a software system to monitor datacenter temperatures in near real-time. My work this summer has improved this system by providing mechanisms to accurately and easily change the alarm threshold values, as well as providing sophisticated graphing and data export tools that will allow for better trending analysis to proactively spot problems.

VI. Acknowledgement

I would like to express my deep gratitude to Mr. Tim Kasza, Mr. Adam Walters and the rest of the Facilities team for their important contribution in the development of these enhancement controls. Without their feedback, this project would not have been successful. Also, I would like to express my appreciation to Ms. Diane Engram, Ms. Linda Diepholz and Dr. Elliott McCrory for their assistance and having given me this wonderful opportunity.

Finally, I would like to extend my thanks to my supervisor, Mr. James Fromm, for his guidance and encouragement during this Summer. I would also like to thank Ms. Lauri L. Carpenter for the quick lessons on Google Chart API, and Django. My grateful thanks are also extended to Mr. Randolph C. Reitz for adding sense of humor to the long meetings.

References

[Google, 2014] Google Charts. Google Charts.

[Lutz, 2013] Lutz, M. (2013). Learning Python,5th Edition. *O'Reilly.*

[Flanagan, 2011] Flanaga, D. (2011). Javascript: The Definitive Guide. *O'Reilly.*

[Beazley, 2009] Beazley, D. M. (2009). Python Essential Reference, 4th Edition. *O'Reilly.*

[DeMarcus, 2007] DeMarcus, T. (2009). Implementing E-mail Alerts to the GCC Temperature Monitoring Program.

[Mukasa, 2006] Mukasa, C. (2006). Temperature Monitoring in GCC.

[Ben-Judah, 2005] Ben-Judah, S. (2005). Garnering Temperature Sensor Data to Display on a GCC Base Diagram Illustration.