University of Glasgow | School of Computing Science

GEMS: Glasgow Energy Measurement System for the Raspberry Pi Cloud

Piotr Hosa

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — March 5, 2016

Abstract

Acknowledgements

Education Use Consent

Contents

Chapter 1

Introduction

1.1 Motivation

Cloud technology has become increasingly popular in recent years. Owners of large infrastructures such as Google, Amazon and Microsoft are able to offer their customers highly scalable services at a lowering cost and no upfront investment. However, dealing with considerable amounts of data and traffic invokes a need for sizeable hardware, which imposes various challenges on its owners. These challenges include automated service provisioning, virtual machine migration and energy management among others [14].

The Glasgow Raspberry Pi Cloud (RPC), as discussed in section 1.2, is a scale model data centre. The RPC is a research and educational tool and it aims to mimic full-sized cloud infrastructures. This is incrementally achieved by the projects done on the RPC, which use leading virtualisation and cloud software such as Docker and Kubernetes.

This project focuses on developing a power monitoring system for the RPC, which is a fundamental step toward power management. Monitoring makes it possible to identify machines with abnormal power consumption, but also to compare the power efficiency of similar software frameworks. While power management is not of particular concern in a cluster of this size, implementing such a system advances the project by making the RPC more similar to full-scale data centres. In the full-scale cases energy management is highly important. Not only is it crucial for reducing operational costs but also to keeping in accordance with environmental concerns, which will only increase in the future [3].

1.2 The Glasgow Raspberry Pi Cloud

1.2.1 Overview

The Raspberry Pi project at the University of Glasgow was started in 2012 by a group of researchers in the School of Computing Science. The motivation to pursue the project was a lack of satisfactory methods that would have allowed to conduct research on data centres and cloud computing. Before the Raspberry Pi Cloud, recreating such infrastructures relied mainly on software simulations and using physical environments with a very limited number of machines. Both of these methods were aimed at reducing costs of such research systems. The RPC eliminated the financial overhead by using 56 low-cost computers and therefore allowed to model full-scale data centres more closely. Using Raspberry Pis in the project also eliminated other challenges associated with traditional data centre servers like space constraints, cooling and considerable energy used by the machines. The RPC also has the advantage of exhibiting real network traffic and running real cloud services, which is difficult to model in software simulations [11].

1.2.2 The Original Cloud

In its original version the RPC consisted of 4 racks of 14 Raspberry Pi 1 Model B computers. Each rack had a top-of-rack switch that connected to a central OpenFlow switch and then to the Internet. Additionally, there was a master node also connected to the OpenFlow switch that for system management. Each of the nodes in the rack run the Raspbian version of Debian. The first software developed for the cluster was a REST API with a web interface. The project also involved experimenting with LXC virtualisation, Hadoop and Software Defined Networking [13].

1.2.3 More Recent Projects

One of the most recent projects with the RPC involved a hardware upgrade. One rack of 14 Raspberry Pi 1 computers was replaced with Raspberry Pi 2 Model B devices, which are more powerful than the ones used previously. The new infrastructure extended the potential applications of the cluster and made it possible to experiment with Kubernetes, Google's cluster management software. In the recent months Kubernetes has been adapted to the ARM architecture of the Raspberry Pi [12] and work has been done on the Kubernetes Dashboard [6].

Before the upgrade, a team of students worked on a wake-on-LAN hardware and load balancer software for the cluster, which used Docker and SaltStack.

1.3 Related Work

1.3.1 Raspberry Pi Clusters

Since the Raspberry Pi has been released it became hugely popular in the computing community. This meant that not only the researchers at the University of Glasgow (the project is described in section 1.2), but also other research institutions took advantage of the low-cost computers.

Most notably, the cluster at Free University of Bozen-Bolzano consists of 300 Rasberry Pi 1 Model B computers. The motivation for creating this cluster was to create an inexpensive test bed for cloud computing research, hence similar to the RCP [1]. However, the Bolzano cluster's considerable size makes it a distinct cluster compared to the RPC (56 nodes), the Iridis-pi cluster (64 nodes), the Boise State University (32 nodes) and the Bradley University cluster with 25 nodes [5], [2], [10].

The Iridis-pi, which next to the RPC, is another UK-based cluster from the University of Southampton. Iridis-pi was created in response to a research and industrial demand for low-power cluster systems. After performing benchmarks on the cluster, the researchers concluded that as the cluster has relatively low computing power and communication bandwidth it is best suited as an educational resource. The researchers also highlighted the cluster's advantages over virtual and full-sized clusters, such as realistic resource utilization and network traffic, low upfront and maintenance cost and redundancy of additional hardware (e.g. cooling). This further confirmed the cluster is best suited as a teaching tool [2].

The Boise State University cluster was created by a PhD student at the institution with the purpose of developing his software in the Rasberry Pi cluster before deploying it on the destination embedded devices. This had such benefits over a full-size cluster as exclusive access and administrator privileges, which made development easier for the student [5].

The Bradley University cluster was created with the purpose to serve as a learning tool in the field of high-performance computing. The goal of the team was to design a cluster made of readily available parts, which could be assembled using a minimal number of tools. This approach was to ensure their cluster could be easily replicated, therefore making it a more popular educational tool. The researchers also insisted on developing their system in Python, which has a gentle learning curve, making it even more approachable for novice programmers [10].

From the above research projects it is visible that there is a trade-off between the Raspberry Pi's low price and its low computing power. This however makes the device well suited for use in education, which does not require high processing power like that of full-scale computer clusters, but needs low-cost solutions.

1.3.2 Power Measurement

The core element of hardware in this project is the power measuring shield created by the MAGEEC project at the University of Bristol [8]. The shield attaches to the I/O pins of the STM32F4 Discovery board and provides hardware interfaces for measuring up to 3 external sources [9]. The STM32F4 is a single-board microcontroller, which means it does not run an operating system like the Raspberry Pi. It uses pre-loaded C programs that run continuously when power is supplied to the board. The MAGEEC shield was designed with the purpose of measuring the power efficiency of compilers. Therefore, the processors that are measured in the MAGEEC project require to be embedded in boards with jumper pins that allow to intercept the CPU power supply or to be stand-alone processors such as Atmel AVR. This served the purpose of measuring the power consumption of the CPU only, excluding other components on the board. In this project the author uses the STM board to measure the power consumption of all the components in the Raspberry Pi, therefore the lack of CPU jumper pins does not pose an obstacle that would otherwise exclude the Raspberry Pi of being subject to this experiment.

In terms of power measurement specifically on the Raspberry Pi, researchers from the Technical University of Darmstadt investigated the power consumption versus CPU utilization and network traffic for a single node in their PowerPi (PP) project [4]. The hardware set-up in PP that enables power measurement works on the same principle as the one in this project, however in this instance pre-existing hardware was utilized to take the measurements (the MAGEEC power shield). Since the PP project focused on determining the relation between power consumption and CPU and network utilization alone, non-essential processes were terminated while performing the experiments. The researchers also took the measurements at night to reduce Wi-Fi interference. Their results showed mostly clear correlation between power consumption and CPU as well as network utilization, indicating increased power consumption for increased CPU load and network bandwidth.

Another them at the University of So Paulo investigated the performance and energy efficiency of two REST frameworks on the Raspberry Pi [7]. To measure power consumption, the researchers used Arduino Nano microcontroller boards. In the experiment two frameworks were compared: Axis2 and CXF. The experiment concluded that the CXF framework consumes equal or less energy than Axis2.

These examples show that research has been done in the filed of measuring the power consumption of the Raspberry Pi, as well as other devices, such as single-board microcontrollers in the MAGEEC project. The MAGEEC research influenced this project in particular, as it provided the means of taking the power measurements without the need to design new hardware and software.

1.4 Summary

The research that is described in this chapter proceeded this project and indicates two trends: the Raspberry Pi is used to create a reliable scale-model data centre infrastructures and power monitoring in devices is crucial to understanding how they use power. Although power monitoring has been performed for the Raspberry Pi, monitoring multiple nodes has not been as popular. This project aims to build on energy monitoring and extend it to multiple nodes in a Raspberry Pi cluster, creating a system that measures power consumption, as well as CPU parameters of individual nodes and makes that data visible to the user.

Chapter 2

Requirements

2.1 User Stories

This section presents on overview of how the users would use the system, as intended at the beginning of the project. The user stories were created by the author and were a basis to constructing a set of functional and non-functional requirements as described in sections 2.2.1 and 2.2.2. The full set of user stories is available in appendix A.

There are two main functions that the system fulfils, that is providing the user with an overall and instantaneous state of the cluster and providing more detailed data that can be analysed outside of the system. The user would be presented with an easily available web GUI. Other intended features included changing the parameters of the system (e.g. sampling frequency) and influencing the nodes in the system (e.g. rebooting) through the GUI.

2.2 Requirements

The following requirements have been arranged with MoSCoW prioritization, which suggests which features should be implemented first. In the course of development due to different factors the requirements have been adjusted to reflect what can be achieved in the project. These changes and design decisions are described in chapters 3 and 4. The requirements listed in this chapter are in the form they were created at the beginning of the project.

2.2.1 Functional

Functional requirements describe specific features to be implemented in the system.

The system **must** have:

1. Real-time graph in the main page that shows power, temperature and load for all nodes in the cluster
2. Alert queue in the main page that updates whenever there is an event that indicates a fault

3. A graphical representation of the cluster with color-coded states that gives an overview and allows to further investigate a node when selected
4. The detailed view for each node containing basic information about it (address, name, up time, alert rate) and also show a real-time graph for power, temperature and load for this node
5. CSV storage that keeps all the records and makes them available for later

The system **should** have:

1. The option to toggle between viewing power, temperature and load in the graphs
2. An indication of whether the power measurement board is active
3. The option to change the sampling frequency and to zoom in and out in graphs
4. The facility to reboot nodes remotely in case of node failure

The system **could** have:

1. List of pods for each node in the GUI, if the cluster is managed by Kubernetes

2.2.2 Non-Functional

The set of non-functional requirements describes the behaviour of the system, its accessibility, compatibility with existing software and operating systems and fault tolerance. The requirements for this system are described below.

The system **must** have:

1. The ability to provide the GUI for multiple clients without over-stressing the system
2. Compatibility with other devices that requires minimal configuration
3. The ability to recover from broken connections and the API failures
4. Cross-browser compatibility for the 3 most common browsers (Chrome, Firefox, IE)
5. Compatibility with such operating systems as Raspbian and Arch Linux
6. The proper structure to be easily maintainable

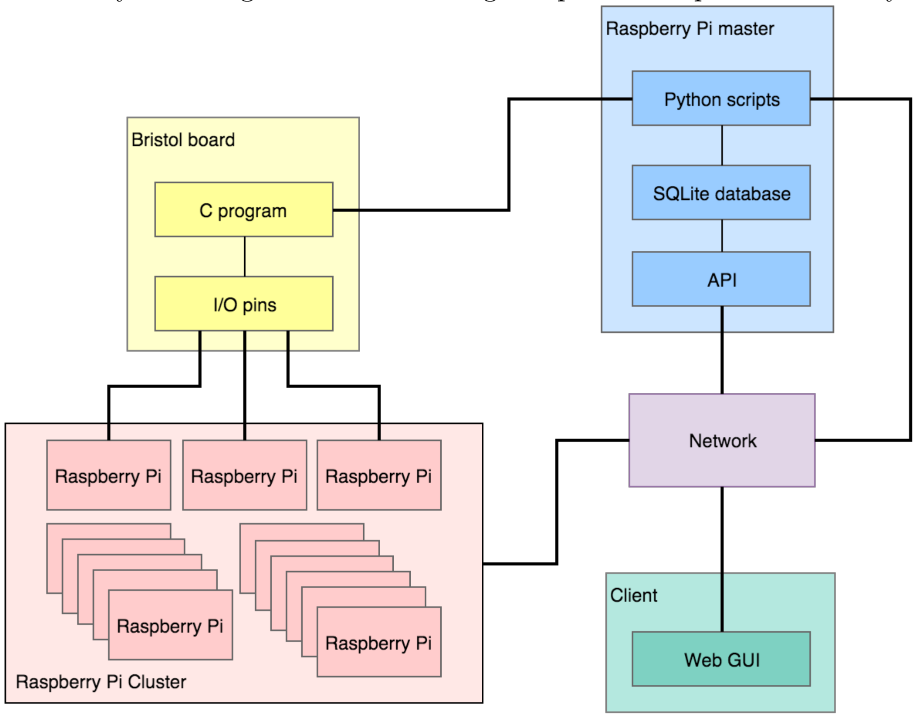
The system **should** have:

1. Compatibility with two less browsers (Safari and Opera)
2. Compatibility with other Linux distributions

Chapter 3

Design

Figure 3.1: System diagram demonstrating the power component of the system.



3.1 Hardware

1. Adapted from the MAGEEC project
2. The setup in the Glasgow PiCloud (MAGEEC wires + network)

3.2 API Server

1. Choice of OS for the Pi

Figure 3.2: THIS HAS TO BE REPLACED! Basic wiring set-up for measuring power consumption.

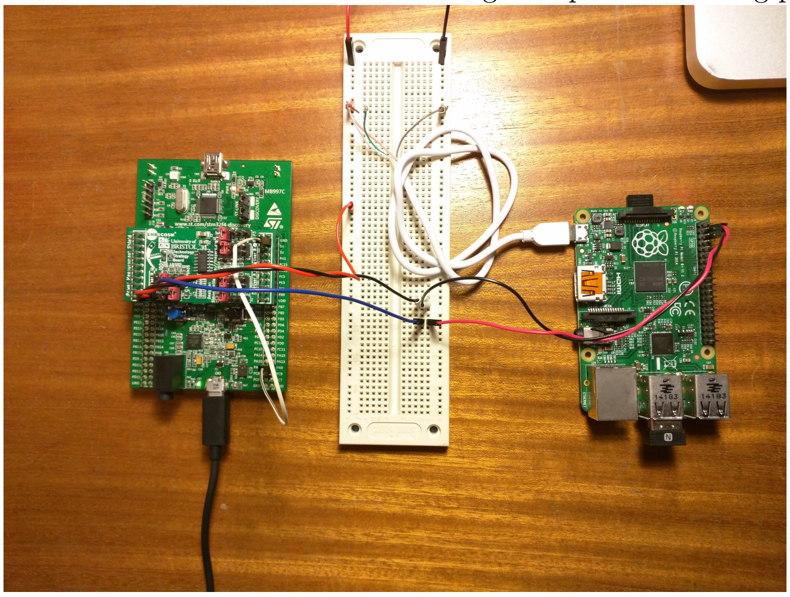
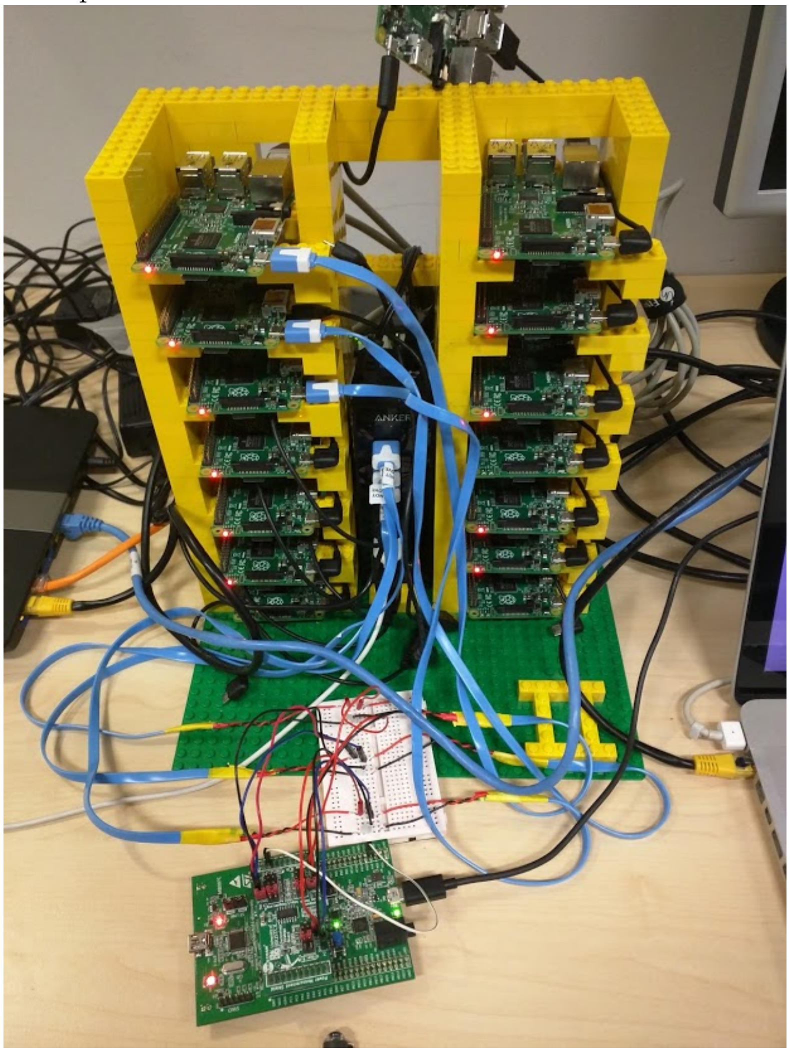


Figure 3.3: The set-up for the whole tower with three nodes connected to the Bristol board.



2. Python chosen as it is most frequently used on the Pi and therefore there is a lot of support and documentation for it.
3. Flask (and Flask restless) chosen for API. Other options available (Django, Tastypy and Sadman) but Flask is simple and popular and therefore seemed like the right choice.
4. Chose to use SQLite dabase with Flask SQLalchemy to make database access easier.

3.3 Web Client

1. Single Page Application design (rendering and logic done in client which takes processing off server; more responsive websites)
2. There is a number of frameworks available (Ember.js, ExtJS, React, Knockout) but Angular has a large community and it supports bidirectional data binding, which some of the other frameworks lack
3. Many front-end frameworks have been adapted to Angular but if they have not it is possible to do it by hand.

Figure 3.4: TO BE CHANGED! Should make this UML? AngularJS modules and their relationships.

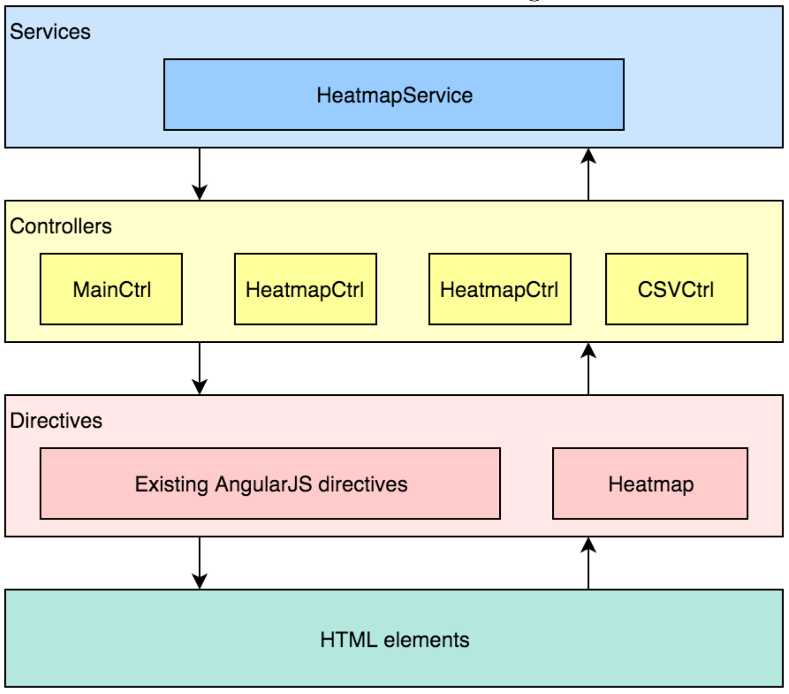
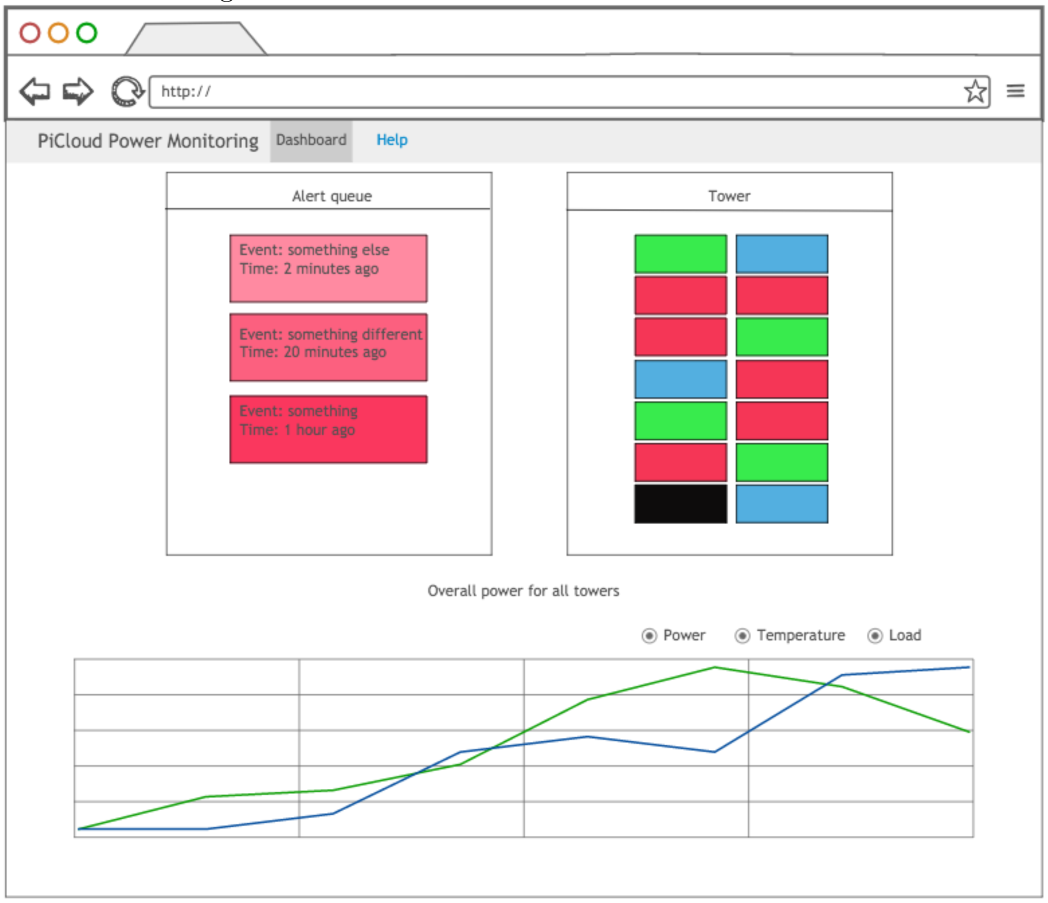


Figure 3.5: Dashboard wire frame for the web client



Chapter 4

Implementation

4.1 Logger Node

1. Logging python (based on MAGEEC pyenergy library)
2. Database and API
3. Web client
4. Salt managing scripts

4.2 Minion Nodes

1. Reporting python

Chapter 5

Evaluation

5.1 Testing

1. Python tests for scripts
2. JavaScript tests(?)

5.2 Acceptance Testing

1. Overview of testing scenario
2. Feedback from testers
3. Changes in system based on feedback

Chapter 6

Conclusion

6.1 Summary

1. What has been implemented
2. What the system can do
3. Comment on usability

6.2 Future Work

1. What other features are the next logical step
2. How do those features fit within the technology stack

Appendices

Appendix A

User Stories

As an admin
I want to see real-time data for the power consumption of the whole tower
So that I can identify the total power consumption

As an admin
I want to see real-time data for the power consumption for selected nodes (as many as possible)
So that I can determine what the power consumption is in various states (idle etc.)

As an admin
I want to see real-time CPU temperatures in my cluster in a graph
So that I know how they compare with power consumption (on the same graph)

As an admin
I want to see real-time CPU loads in my cluster in a graph
So that I know how they compare with power consumption (on the same graph)

As an admin
I want to be able to change the power, temperature and load sampling frequency
So that I can instantaneously have an overview of the data as well as a closer look whenever I want to

As an admin
I want to see a real-time diagram representation of the cluster (power, temp, load) with color coding indicating faults or extreme conditions (such as high temperature)
So that I have a good overview and can identify which nodes exhibit abnormal behaviour

As an admin
I want to see an alert queue that is separate from any other diagram and displays an alert when I need to take action (maybe email alert as well)
So that I have a clear indication when I need to respond quickly

As an admin
I want to be able to turn off or reboot nodes from the GUI
So that I have an easy way of dealing with non-responsive nodes

As an admin I want to be able to see pods running on each node in real time
So I have a better understanding for the level of the node's power consumption

As an admin
I want to be able to access log files for power usage (csv)
So that I can safely store data and analyse it later

As an admin
I want to be able to access the GUI from any browser
So that I can view the cluster state from any machine

As an admin
I want to be able to see the status of the power measurement board
So that I know whether it is providing measurements for the API

As an admin
I want to receive data (power, temperature and load) instantaneously
So that the graphs update smoothly

Bibliography

[1] Pekka Abrahamsson, Sven Helmer, Nattakarn Phaphoom, Lorenzo Nicolodi, Nick Preda, Lorenzo Miori, Matteo Angriman, Juha Rikkila, Xiaofeng Wang, and Karim et al. Hamily. Affordable and energy-efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment. *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, 2013.

[2] Simon J. Cox, James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott, and Neil S. OBrien. Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*, 17(2):349–358, 2013.

[3] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. *IEEE Network*, 29(2):56–61, 2015.

[4] Fabian Kaup, Philip Gottschling, and David Hausheer. Powerpi: Measuring and modeling the power consumption of the Raspberry Pi. *39th Annual IEEE Conference on Local Computer Networks*, 2014.

[5] Joshua Kiepert. Creating a raspberry pi-based beowulf cluster. *Boise State University*, 2013.

[6] H. McWha. Kubernetes Dashboard run as Docker image on Raspberry Pi 2., 2016.

[7] Luiz Henrique Nunes, Luis Hideo Vasconcelos Nakamura, Heitor De Freitas Vieira, Rafael Mira De Oliveira Libardi, Edvard Martins De Oliveira, Lucas Junqueira Adami, Julio Cezar Estrella, and Stephan Reiff-Marganiec. A study case of restful frameworks in Raspberry Pi: A performance and energy overview. *2014 IEEE International Conference on Web Services*, 2014.

[8] James Pallister. Energy measurement infrastructure.

[9] James Pallister, Simon Hollis, and Jeremy Bennett. Impact of different d options on energy consumption, 2012.

[10] Aaron M. Pfalzgraf and Joseph A. Driscoll. A low-cost computer cluster for high-performance computing education. *IEEE International Conference on Electro/Information Technology*, 2014.

[11] Fung Po Tso, David R. White, Simon Jouet, Jeremy Singer, and Dimitrios P. Pezaros. The Glasgow Raspberry Pi Cloud: A scale model for cloud computing infrastructures. *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, 2013.

[12] Jim Walker. How to: Kubernetes multi-node on Raspberry Pi 2s, 2015.

[13] David R. White. A Raspberry Pi cloud, 2013.

[14] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *J Internet Serv Appl*, 1(1):7–18, 2010.