A Real-time Distributed Hardware Health Monitoring Framework

Sai Santhan Kusam Venkata, Jahnavi Bharadwaj, Gillian Dobbie, Sathiamoorthy Manoharan
University of Auckland
New Zealand

Health monitoring involves sensing, reporting, and sometimes adjusting the states of objects or nodes remotely. This paper describes the design and implementation of a real-time distributed hardware health monitoring framework, assuming a homogeneous set of hardware nodes. The framework consists of sensor components operating at the nodes, and visualization components operating at monitoring stations. A service-oriented software architecture naturally lends itself to fit such a framework. As the primary focus and case study, the paper presents the implementation of a system to monitor the status of a collection of computers, where the sensors are hardware probes and counters such as CPU probes and performance counters.

***Keywords*-health monitoring framework, hardware monitoring, service-oriented architectures.**

I. Introduction

Health monitoring is the process of getting information about different components of critical equipment. It is an important process conducted by many people on different types of equipment ranging from computers and telecommunications equipment to smart homes. Information gathered through monitoring a device can be used to ensure that the system is running efficiently and detect any faults before the system experiences downtime.

Monitoring critical equipment holds many advantages. Moreover, having the ability to monitor multiple nodes in one location drastically reduces the amount of time taken up by monitoring. A tool that is able to monitor multiple nodes simultaneously is not only beneficial for network administrators, but is also very helpful for people that are into cryptocurrency mining, home automation and monitoring, and any research based on electrical equipment. With a generic architecture, people can use such a tool to monitor sensors and other things in their house as home automation and the Internet of Things (IoT) is becoming very popular.

A tool that is able to not only provide the user a convenient way of monitoring multiple nodes in one place, on multiple devices, but also able to notify them of any issue that may arise while they are not actively monitoring is very advantageous as this will give them the ability to go and address the issues before it causes any downtime, or reduce the downtime that this issue causes. Having the capability to remotely perform maintenance activities and other important tasks is also very advantageous because this allows the user to place their equipment in hard to reach places and still perform important tasks while they are on the other side of the world.

There are currently many hardware monitoring tools on the market, but they also have many drawbacks. These tools are only capable of monitoring one node at a time, which drastically increases the amount of time an administrator has to spend checking different nodes they are responsible for. Another major disadvantage is that all of these tools are domain specific, and not modifiable, meaning they cannot be used to monitor different types of equipment simultaneously.

This paper describes the design and implementation of a hardware monitoring framework that is robust enough to monitor multiple nodes at a time, while being generic and modifiable to allow the monitoring of multiple types of devices. A tool based on this framework will be very useful to have for anyone that is in charge of monitoring equipment. This framework will also allow for activities to be performed on the equipment remotely, which can include maintenance activities and emergency procedures. As the framework allows for a central server to be used to monitor nodes, it can also be used to log information and even notify the user when a potentially problematic issue arises.

The central server as well as the nodes distribute data (aggregated or raw) to the consuming parties. Such data distribution system naturally falls in the category of service-oriented architectures, typically defined as a "set of components which can be invoked, and whose interface descriptions can be published and discovered" [1], [2]. The service interface defines the operations of the service as well as the object types and formats used in conjunction with these operations.

Older Web Services use SOAP [3], an XML format for representing objects. Modern Web Services, however, use something more efficient: representational state transfer (REST). REST is a light-weight protocol that loosely couples the service and the client consuming the service, and is based on HTTP [4], [5], [6]. There are many efficiency gains a RESTful Web Service has over a SOAP-based Web Service. For example, the messages in a RESTful service are more efficient than SOAP messages. Besides, service responses can often be cached for re-use, thus saving on server resources, network bandwidth, and latency. Many of the current service-oriented architectures, therefore, use RESTful services [7]. It will therefore be noted later that

978-1-5090-2399-8/16/$31.00

our monitoring framework too uses RESTful services to implement the distribution of data both from the nodes that are monitored, as well as from the central server that monitors these nodes.

The rest of this paper is organized as follows. Section II reviews some of the related tools that are available. Section III describes our design goals, the software architecture, and its implementation. Section IV presents a comprehensive evaluation of the framework including its user-interface. The final section concludes with a summary.

II. Related Work

As indicated earlier, there are currently many hardware monitoring tools available, and this section reviews some of the key tools. Note that these tools use domain-specific frameworks that don't suit tailoring to domains they aren't designed for.

A. Core Temp

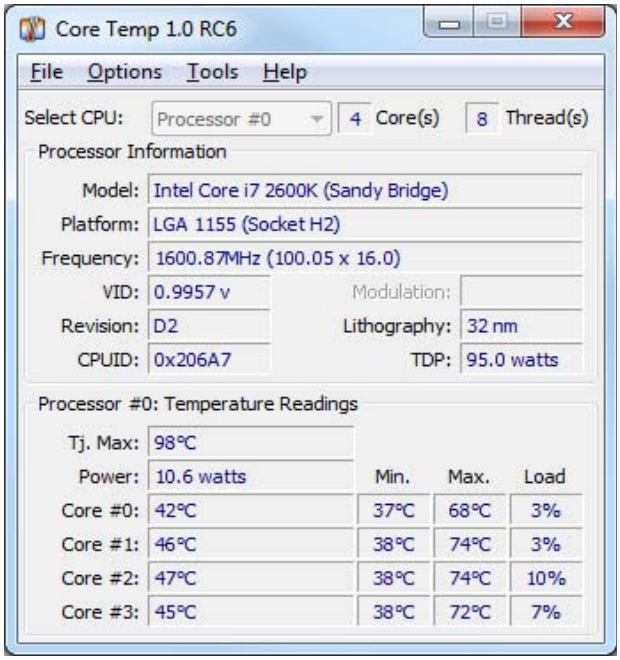

Figure 1. Core Temp Main Window

Core Temp is a compact and lightweight tool that is designed to monitor a computer's vital information [8]. It is able to retrieve information such as CPU, model, platform, frequency, lithography, and temperatures of each core with minimum and maximums and the total power drawn by the processor. All this information is gathered in real time with an API that directly interfaces with the motherboard.

Core Temps UI is a simple desktop application that displays all the essential information in a single window. They have also recently created a mobile application that lets the user monitor their computers information from their mobile device. The limitation of this system is that it only lets the user monitor data coming in from a computer, and you can only monitor one device at a time, and only on the local network unless complicated network configuration is done, which could in turn create some security issues.

Core Temp also provides a developer friendly SDK written in C++ that can be leveraged by other developers to pull information from computers. This SDK is packaged into a convenient library which can be imported into C# or C++ applications.

B. OpenEnergyMonitor

OpenEnergyMonitor is an open source community project that is focused on developing an energy monitoring and control system for domestic and industrial applications [9]. It consists of many different components that also include specialised hardware that enables the system to function currently. The software component of the system is called emoncms, which currently runs on a Windows computer and Android based smartphones.

This system provides a very user-friendly interface containing information in many forms including graphs. This system is great for monitoring energy usage and generation, but it requires the user to purchase or make specialised hardware to interface with the software components, and it is only designed to work for energy systems.

C. ASUS AI Suite 3

ASUS AI Suite 3 is a configuration and monitoring tool supplied by ASUS with all of their motherboards [10]. This suite is able to monitor many different components of the system including voltages of CPU and RAM, fan speeds, and temperatures from various different sensors from around the motherboard. It is also able to configure clock speeds of processors that support this feature, and also change voltages of different components on the motherboard.

Due to the proprietary nature of this tool, it is able to perform low level configuration of components and provide an intuitive user experience, but it is not able to be used to monitor or configure motherboards that are not manufactured by ASUS. ASUS AI Suite 3 not only provides the user with a desktop application to interact with, but also has applications for Android and iOS, which allows for local network monitoring.

D. EVEREST Ultimate Edition – PC Diagnostics

Everest is a computer diagnostics and monitoring tool developed by a Canadian firm Lavalys. It is advertised as a complete PC diagnostics software utility with features including but not limited to hardware device information, software and operating system information, hardware monitoring, speed and performance benchmarks, and system stability testing [11].

The main feature of Everest is real-time hardware monitoring. It gives the user hardware information gathered from sensors in real-time in a way that is very intuitive. Due to the complexity of the software and the number of features it has, it is very heavy on the system's resources, and the core functionality of hardware monitoring is dwarfed by the other features that this software has. However, like many of the other tools we came across in our research, Everest only functions on the local machine, which means remote monitoring is not possible, and the software is not lightweight.

E. Splash Monitoring

Splash Monitoring is a renewable energy monitoring tool. It is currently in use in many systems such as solar water heating systems, photovoltaic systems, energy management systems, and plumbing systems. It is purpose built or customized for each application, which makes it extremely well suited to the environment it is being deployed into. The system features animations and special graphics that are designed to match the environment the software is deployed into, which makes the system more user friendly.

It also requires proprietary hardware to accompany their software for the system to function correctly. For these reasons, it needs to be professionally installed, which makes the system very expensive to setup. Remote monitoring is also an option but it will add to the cost of the system.

F. eG Enterprise Server Hardware Monitoring

eG enterprise is an integrated monitoring software that is compatible with monitoring servers produced by multiple vendors. It is also compatible with multiple operating systems. Where eG excels compared to the others in the market is that it allows for remote monitoring and monitoring of several servers from a central console. eG can provide the user with information regarding how many processors are available, how the cooling units are functioning, memory availability, system faults, and information from various other sensors.

eG Hardware Monitor also has the ability to proactively alert administrators of any hardware or software issues, which can help them to proactively resolve the issue before the server goes down.

This system does have a lot of advantages for system administrators in large companies with a wide array of servers, but it is far too complicated and expensive for individuals as well as small to medium sized business. Another disadvantage of such a system is that it is domain-specific and does not offer many customization options.

III. Design and Implementation

The design of the monitoring framework follows the following goals:

- Monitor multiple nodes from a central location.

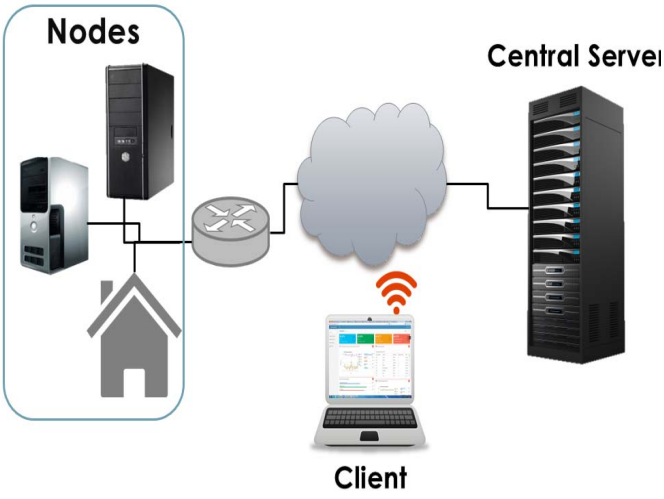

Figure 2. Distributed architecture of real-time hardware monitor framework

- Flexibility to work with different types of nodes.
- Ability to customize.
- Be user friendly and easy to configure for potential users that arent computer experts.
- Allow the user to remotely monitor the equipment they want to monitor.
- Keep a log of all the historical values to help data analysis and debugging.

Figure 2 depicts the high-level architecture of the system. The nodes interact with a central server so that the server collects the monitored data off the nodes. Clients again query the central server to obtain the collected data. Figure 3 shows the components of the high-level architecture in more detail. These components are described below.

A. Node

As shown in Figure 3, a node comprises of two different components. The node uses two components to increase the modularity and robustness of the framework. Gathering information from sensors requires administrative access. The Node Sevice which is a local OS service has the administrative rights and gathers sensor information. The information is then passed onto the Node API which is a Web application.

The Node Service is responsible for gathering hardware information from sensors. It uses an open source framework called Open Hardware Monitor to gather sensor information.

The Node API is a Web application that provides a RESTful interface to the node. This application allows the Server API and Server Service components to request information from the Node.

When it receives a GET request with a parameter, it checks what the parameter is before forwarding it to the Node Service. If the parameter is one of the permitted ones,

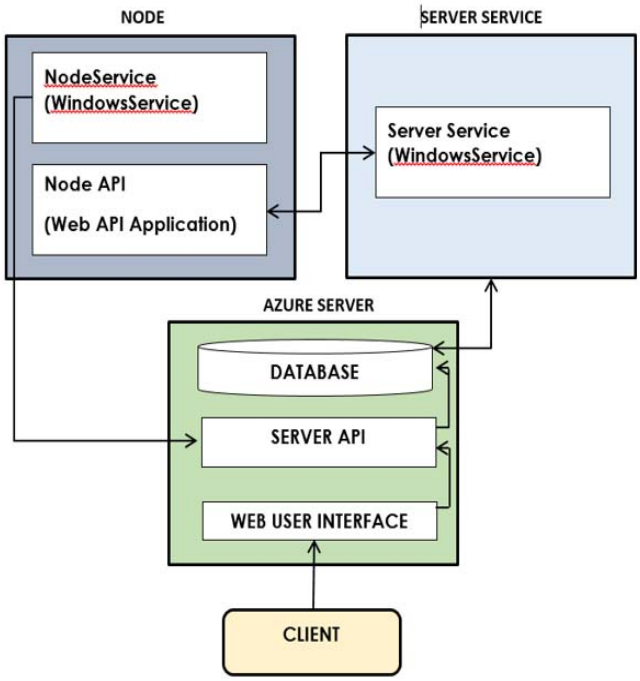

Figure 3. Architecture of the system and interaction between components

it opens a connection to a named pipe, writes the parameter to the pipe, and waits for the service to return values. These values are then turned into a JSON array before being returned to the requester.

The Node API is designed to accept tasks from the Server API to change node settings. However, this is feature is currently incomplete.

B. Server API

Server API provides a RESTful interface to access many functions of the hardware monitoring framework. To support a modular design, there are multiple controllers in the application responsible for a certain type of tasks.

C. User Interface

The graphical user interface of the hardware monitoring framework uses HTML, CSS and JavaScript. It also makes use of Bootstrap, which is a popular HTML, CSS and JS framework for developing responsive websites.

The moniotring data is gathered through the Server API using AJAX calls. Where charting is required, the user interface uses Google Charts. See Figure 4.

D. Server Service

The Server Service is an optional standalone tool created for polling each node for information and putting it into the database. This is also a service that is meant to run in the background all the time, so it was implemented as a Windows service.

The Server Service obtains a list of nodes along with their addresses and list of measurements able to be retrieved. Once this information is obtained, the service then requests each node to send its hardware parameters to the service. When it receives the parameters, it updates the node document on the database by adding the new list of parameters to the updates list and committing it back to the database.

IV. Evaluation

The system meets its design goals described in the previous section. It supports monitoring multiple nodes, and is flexible enough to accommodate for user customization.

A. Web Application Performance & Quality

There are many systems and tools that allow analysing the performance of a web application [12], [13], [14], [15].

We used an open source analysis tool called YSLOW [12]. YSLOW analyses web pages and suggests different ways to improve the performance.

Our initial YSLOW ranking was a “D”, but based on YSLOW’s recommendations, we improved the ranking to a “B”. The following is a list of improvements made.

1) Reduced the number of HTTP requests – The number of HTTP requests were reduced by combining some of the JavaScript files and some CSS files. We did not go overboard with this, though, because combining unrelated JavaScript files would decrease the code quality and coherence of the code in a single file.
2) Add “expires” tags to resources - The Expires headers for resources help the browser determine if the resources cached are still valid or it should load a newer version. If the expires tag is not configured, browsers are forced to load all resources every time the page is opened. However, a number of resources loaded from external servers (e.g. Google fonts) do not have an expiry tag, and we are not able to do anything about it because these external resources are not in our control.
3) Enable static and dynamic compression - We enabled static and dynamic compression on our server to reduce the size of the files being transmitted. Compressions improves the efficiency of the bandwidth usage.
4) Minified JavaScript files - We minified JavaScript files to further reduce the size of these files. This contributes further to improving the response times.

We could not improve the grade beyond a “B” further because of third party resources not having expiry tags, and other factors that are not in our control.

B. Heuristic Evaluation

Heuristic evaluation is a form of usability inspection technique used to identify issues with user interfaces. In particular, for evaluating the usability of the user interface we have created, we went through Nielsens heuristics, which

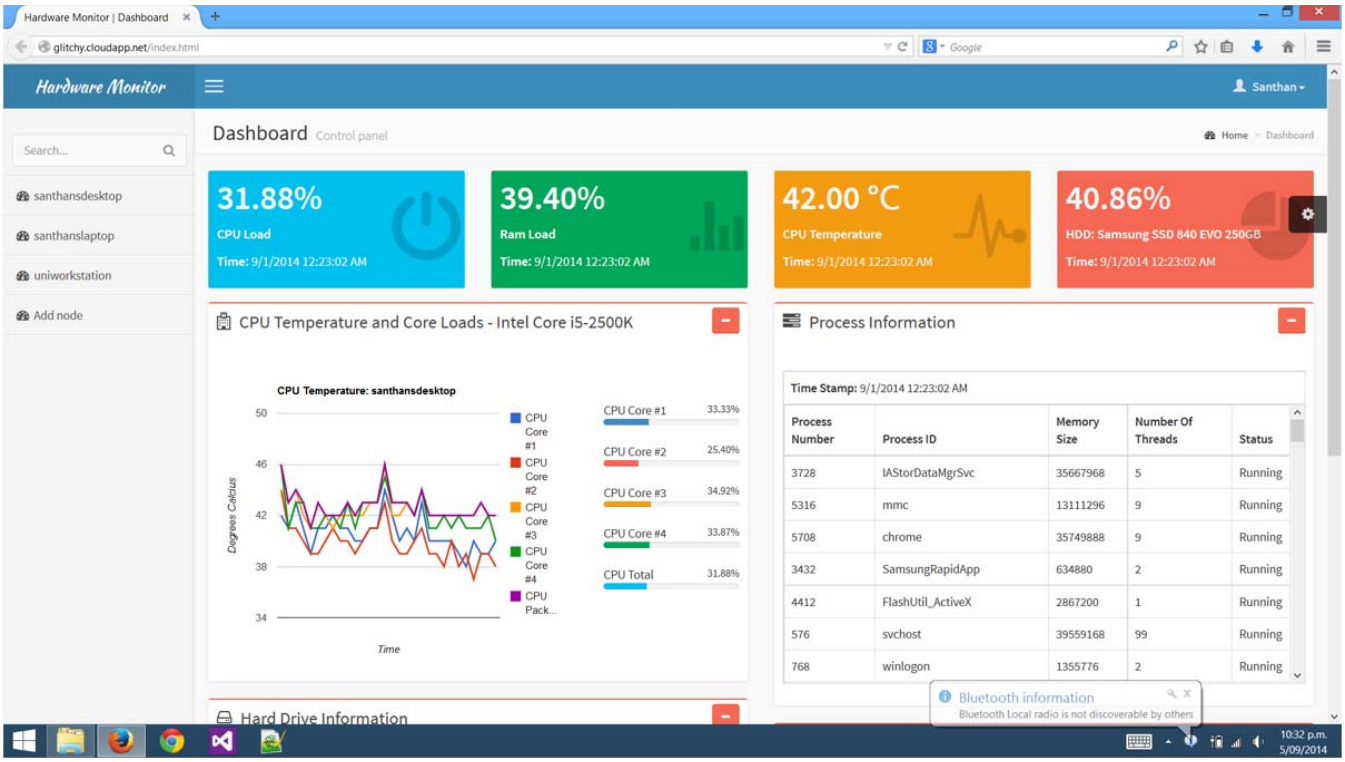

Figure 4. Screenshot of the user interface

are the most popular usability heuristics for user interface design. The list of heuristics and how they are met by our user interface is given below:

1) Visibility of system status - The time at which data from the node was gathered is always shown to the user, which will give them a status of the node.
2) Match between system and real world - We have not used any complicated diagrams or terminology in the interface. Wherever possible, we used symbols that are able to match the elements in our system to real world elements.
3) User control and freedom – This was an easy heuristic to meet because we do not actually have many states that the interface can be in. We have also designed the interface in a way that would lead the users to feeling like they are always in control of the interface and that they have the freedom to perform actions they want. We have also provided ways for them to revert actions. An example of this is that widgets that are accidentally minimised can be brought back with a click of a button.
4) Consistency and standards - Throughout the interface we have used standardised symbols and terms that are known to computer users.
5) Error prevention - when the user is adding nodes, error checking is done to ensure that information that the user has entered is in fact correct.
6) Recognition rather than recall We have used standard symbols and terms in the interface to help the users understand tasks in the interface without putting excessive cognitive load on the user.
7) Flexibility and efficiency of use - One of the most prominent features of our interface is its flexibility. What makes it flexible is that all the widgets are separated and can be moved around to different places on the interface, which allows the user to move the information they value the most to the top of the page. All information related to one node is on a single web page, which improves the efficiency.
8) Aesthetic and minimalist design - As mentioned above, all information is displayed efficiently on a single web page. The layout is also just simple widgets in a grid like layout, which is minimalistic and familiar to everyone.
9) Help users recognise, diagnose and recover from errors - If/when the user makes a mistake or the interface goes into an error state, the user is immediately informed of the error and what has caused the error. We have used plain English for these error messages, which will help the user recover from it.
10) Help and documentation - User manuals and help documentation will be provided to the users, should they need it.

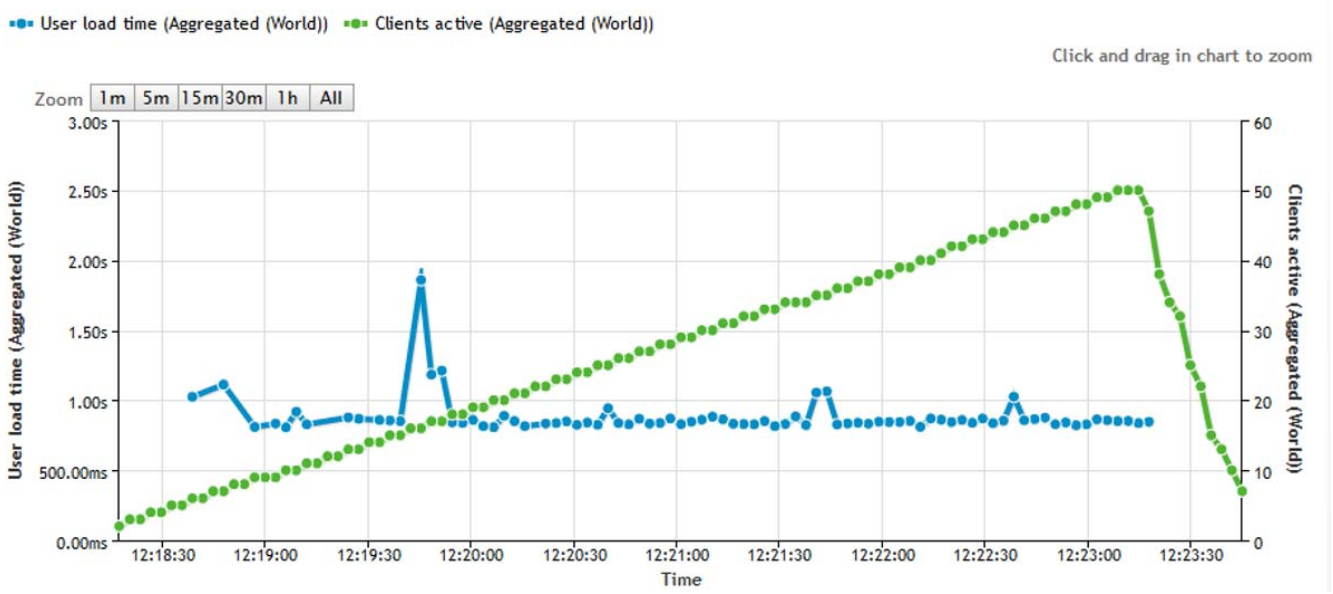

Figure 5. LoadImpact test results

C. Performance Evaluation

To evaluate the performance of our website and ServerAPI, we used a tool called LoadImpact (www.loadimpact.com) to create virtual users and stress test it. Figure 5 shows how our framework handled the load.

The graph shows that other than a little peak at 15 users, the load times of the web page and all its content stayed within 1 second, which is a very acceptable result.

V. Summary & Conclusion

This paper presented a framework for hardware health monitoring. Both heuristic and performance evaluation of the system highly rate the framework.

While the concrete implementation focussed on monitoring computer hardware, the framework is designed to be extensible and can be easily customized. Further development of the framework could include other health monitoring scenarios such as health monitoring of structures (e.g., bridges and buildings).

References

[1] H. Haas and A. Brown, *Web Services Glossary*, February 2004, W3C recommendation.

[2] D. Sprott and L. Wilkes, "Understanding service-oriented architecture," *The Architecture Journal*, pp. 10–17, January 2004.

[3] M. Gudgin *et al.*, *SOAP Version 1.2 Part 1: Messaging Framework*, 2nd ed., April 2007, W3C recommendation.

[4] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[5] J. Webber, *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly, 2010.

[6] L. Richardson and S. Ruby, *RESTful web services*. O'Reilly, 2007.

[7] T. Erl, B. Carlyle, C. Pautasso, and R. Balasubramanian, *SOA with REST: Principles, Patterns &Constraints for Building Enterprise Solutions with REST*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2012.

[8] CoreTemp, "http://www.alcpu.com/coretemp/," 2013.

[9] OpenEnergyMonitor, "http://openenergymonitor.org/emon/," 2015.

[10] ASUS, "AI suite 3," 2015. [Online]. Available: http://ai-suite-3.software.informer.com/1.0/

[11] Lavalys, "EVEREST ultimate edition - PC diagnostics," 2015. [Online]. Available: http://www.lavalys.com/products/everest-pc-diagnostics/

[12] Yahoo!, "Best practices for speeding up your web site," 2015.

[13] D. Menasce, "Load testing of web sites," *Internet Computing, IEEE*, vol. 6, no. 4, pp. 70 – 74, July/August 2002.

[14] L. Motalova and O. Krejcar, "Stress testing data access via a web service for determination of adequate server hardware for developed software solution," in *Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, vol. 1, March 2010, pp. 329–333.

[15] J. Križanić, A. Grgurić, M. Mošmondor, and P. Lazarevski, "Load testing and performance monitoring tools in use with AJAX based web applications," in *MIPRO, 2010 Proceedings of the 33rd International Convention*, May 2010, pp. 428 – 434.