High Performance Cluster Monitoring System

Conference Paper · November 2018

DOI: 10.23919/APSIPA.2018.8659536

7 authors, including:

Lam Nguyen
Foreign Trade University
25 PUBLICATIONS **379** CITATIONS
SEE PROFILE

Charles Peck
Earlham College
20 PUBLICATIONS **205** CITATIONS
SEE PROFILE

High Performance Cluster Monitoring System

Xunfei Jiang, Tuguldur Baigalmaa, Lam Nguyen, Daiki Akiyoshi,Eli Ramthun, Niraj Parajuli, Charles Peck
Earlham College, Richmond, IN 47374, USA
E-mail: {jiangxu, tbaiga13, ltnguyen14, dakiyo14, ebramth15, nparaj14, charliep}@earlham.edu Tel: +01-765-9831708

***Abstract*—System monitoring is an important basis for system modelling and improvement. For achieving higher efficiency in performance and lower energy consumption in cluster systems, we design a monitoring system that tracks the performance and temperature of clusters for education and research purposes. The total energy consumption of clusters can be estimated by using the performance and temperature data. Specifically, in our proposed system, all real-time data (including temperature and activities of components of computing nodes) is collected and stored in Round-Robin Databases (RRDs). These data can be visualized or downloaded through a friendly user interface for further analysis. Moreover, our system also provides users with a powerful runtime comparison feature, which allows users to compare the performance of a running experiment with historical experimental results without waiting for the completion of experiments. The data visualization and user interfaces in the monitoring system are demonstrated by using an experiment on our cluster system.**

I. Introduction

Cluster systems establish a new approach that provides high-performance computing environments with low costs. The cluster computing environments have been widely used in a variety of scientific research projects and real-world applications. For example, Oracle RAC utilizes cluster systems for providing high-performance, scalable and reliable services [1]. Google built large scale clusters by combining thousands of commodity-class computers for web search services [2].

In addition to the ability of high performance computing, cluster systems, as well as other large scale computing systems, also consume large amount of energy. According to a report, data centers consumed nearly 4.5 billion dollars in electricity in 2006, which contributed 1.5% of total electricity consumption in the United States [3]. To minimize the energy cost, many studies have been conducted to investigate the tradeoffs between performance and energy efficiency. Assaf *et al.* developed an energy-aware prefetching method for hybrid storage systems by utilizing the advantage of low energy cost in solid state disks [4]. Chavan *et al.* proposed a thermal-aware file assignment strategy for reducing the cooling cost of storage systems [5]. Jiang *et al.* characterized the thermal behaviors of computing components, and proposed a thermal model of storage systems for cooling cost estimation [6].

Since energy monitoring systems are able to provide preliminary data for analyzing the characteristics of energy consumption and validate the accuracy of energy models, we propose an energy monitoring system for clusters. Our system collects activities and temperature of major components in clusters, and provides a friendly interfaces to users for data visualization and comparison. Through the proposed system, users can easily manage their experimental results, and export them for further analysis. More importantly, our system also delivers an online comparison tool, which allows users to improve their energy efficient solutions by comparing with historical data without waiting for the completion of the running experiments.

The major contributions of our system are summarized below.

- The proposed system captures a variety of performance related and energy efficiency related data in cluster systems.
- All data can be visualized through friendly user interfaces.
- The data collection process can be customized for different scenarios.
- The collected data can be compared with historical data at runtime.

The rest of this paper is organized as follows: Section II surveys related work. The design of the proposed system is illustrated in Section III. The implementation details and experimental results are presented in Section IV. Section V concludes this paper.

II. Related Work

We review previous work related to data visualization and software in system monitoring.

A. Data Visualization

Data visualization is an important module in a monitoring system. It provides an immediate view of the system status in a convenient way. D3.js (Data Driven Documents) is one of the most popular data visualization library [7]. All charts and diagrams are visualized and rendered by using HTML, CSS, and SVG techniques. For our project, we need to display activities and temperature of computing components in clusters through figures, and provide diverse interfaces for user interactions. Allowing users to enlarge figures for detailed information is one of features supported in our system. D3.js supports interactive SVG figures efficiently at small-scale; however, our system always collects large

amount of data on high performance clusters, which cannot be effectively supported by D3.js.

RRDtool is an open source tool of high-performance data logging and graphing for time series data. It can be easily integrated with script-based applications [8]. The data is managed in a Round-Robin manner; old data will be overwritten by new data when the data space is full. In our project, RRDtool satisfies our requirements because we do not need to record the system status all the time. Instead, users should specify the time periods they are interested, and then we can save data in the specified time periods. Moreover, RRDtool also provides interfaces to generate figures in various formats (e.g., PNG, SVG, and EPS), and compare data collected in different time periods or across nodes. These two features enable us to display and compare experimental results for users.

B. Software for monitoring

Ganglia is one of the most popular monitoring systems for high performance computing systems [9]. It uses XML for data representation, XDR for compact and portable data transmission, and RRDtool for data management and visualization. Users can add metrics by using gmetric (a module in Ganglia). It also provides metric comparison across nodes; however, customized comparison is not supported well.

Although most cluster monitoring systems were designed for administrators to track the system status, there were a few monitoring systems that were proposed for ordinary users. Li and Zhang developed an HPC cluster monitoring system for Grid computing and utility computing clusters that enable users to find available computing nodes and customize the HPC facility for better performance [10]. Moore *et al.* presented a monitoring system that collects a subset of cluster monitoring data for users to make better use of computing resources on clusters [11]. These two systems aim at increasing the usability of cluster nodes in order to improve the performance; however, online customized data analysis was not supported. Users have to download and analyze the data at a local system.

There are also some monitoring systems were designed for monitoring performance of specific jobs or applications [12][13]. Gómez-Iglesias *et al.* designed a tool that provides the most critical information for running an application efficiently on HPC systems [12]. The information provided forms a complete view of the application's interaction with the system resources. The tool was designed to be scalable and have minimal impact on application performance, and includes support for different accelerators. However, this system stores all the collected data in plain text and focuses on single job analysis. And all users of the system have access to the collected data.

We propose a monitoring system that not only supports to visualize the activities and temperature of clusters, but also allows users to conduct online data comparison for analysis. In addition, our system protects users data that only registered users are able to access the archived data requested by themselves.

III. Design

In this section, we illustrate the design of the proposed monitoring system. As Fig. 1 displays, our system consists of a data collection module and data visualization module. The data collection module runs at background for collecting and processing data at runtime; while the data visualization module interacts with users through friendly user interfaces.

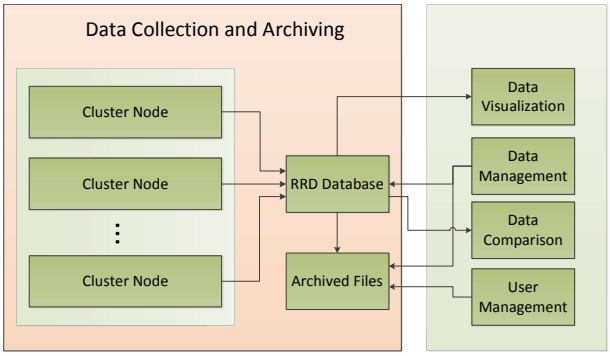


Figure 1: Framework of our cluster monitoring system.

A. Data Collection

Our system collects both utilization and temperature of major components from computing nodes. The computing components include processors, memory, hard drives, and Graphics Processing Units (GPUs). Moreover, our system also collects the inlet and outlet temperature of chassis of computing nodes for estimating the cooling cost of the cluster system. All collected data is processed and then written to Round-Robin Databases.

For convenient comparison and analysis, our system supports user specified archiving. To satisfy a variety of scenarios, a single data collection job from a user can be customized by specifying interested metrics, computing nodes, and data collection time periods and time interval. And the data collection module will schedule all jobs for execution.

B. Data Visualization

The module of data visualization focuses primarily on visualizing experimental results to users. Our system supports real-time and historical data visualization. The expiration of real-time data is configurable in our system. For example, the real-time database can be set to serve data collected in the last one hour, and all previous data can be accessed in the historical database.

C. Data Management

Experimental data for a user is managed independently from other users. After the data is persisted by the data collection module, users can (1) visualize the data by using our data visualization interfaces; (2) export the data to their local systems for further analysis; (3) remove the data if it will not be used anymore. Furthermore, the system supports searches for archived data by using experiment names, computing nodes, metrics, and execution time. The search results could be sorted by experiment names, the start time and finish time of experiments, and etc.

D. Data Comparison

To assist researchers for online data analysis, our system can visualize the comparison of experimental results for users. In the comparison process, users can specify computing nodes, the time periods of experiments, and metrics used in the comparison.

E. User Management

The proposed system has an internal user management module that supports user registration and experiment customization. A registered user can specify interested metrics, nodes, data collection time periods, and data collection intervals for their experiments. Data collected in experiments is stored separately. Administrators can manage all experimental results while ordinary users can only access their own data.

IV. Implementation and Demonstration

In this section, we present the implementation details as an example to demonstrate how the proposed system can be used for monitoring cluster systems.

A. Testbed

All experiments were conducted on a homogeneous cluster of 12 nodes. Each node was equipped with two Intel(R) Xeon E5530 processors, 16 GBytes memory, a Seagate ST380215AS 80G hard drives, and a Tesla C1060 GPU. All nodes were running under CentOS 5.11, and connected by GigaBit Ethernet network.

Temperature sensors were deployed on chassis of computing nodes for capturing the inlet and outlet temperature of all nodes. A MiniGoose [14] was installed to collect data from these temperature sensors. We also launched programs at backend to monitor the utilization and temperature of major components on computing nodes. Modern processors, hard drives, and GPUs had interior temperature sensors that periodically report their temperature. The temperature of GPUs were detected by using nvidia-smi [15]. The utilization of CPUs and hard drives were captured by *iostat* utility tool.

All data was collected once per second in our experiments. Real-time expiration time was one hour, and the historical database stored data for the past 7 days. For better performance in data access and visualization, we created a RRD database file for each component on a node.

B. Data Visualization

Experimental data was visualized and displayed to users through web-based interfaces. We used an open-source project AdminLTE [16] as the framework of our web-based user interfaces, and all figures were generated under the help of jsrrdgraph [17].

Fig. 2 displays an example of visualizing the real-time results on a single node. The figure has two y-axes, and all temperature and utilization data is highlighted in different colors. It is worth noting that we use average values for cases if multiple cores or hard drives are equipped on a single node. The figures are SVG file which could be scaled by mouse scrolling while mouse down. In addition, this web page will be refreshed once per 5 seconds automatically in order to reflect the latest changes.

Figure 2: An example of real-time data visualization for node AS1.

In our implementation, we also support the visualization of temperature and utilization data of components in separate figures. User could specify the nodes for display. In default, all data for all nodes will be display in this page. Fig. 3 is an example of CPU temperature for a single node; each line represents the temperature of a CPU core. In the legend area, the most recent value, max value and average value of the last hour are also displayed. If users are interested in comparing a subset of the CPU cores, they could deselect the other check-boxes at the right side of the figure for a better view.

Figures of memory, hard drive, GPU, and inlet/outlet temperature are also available[1] in the same page. All real-time data can be downloaded from this page by any user.

To support further analysis, all historical data can be visualized in the historical data page. Users can access the

[1]All other temperature/utilization figures are not displayed in this paper.

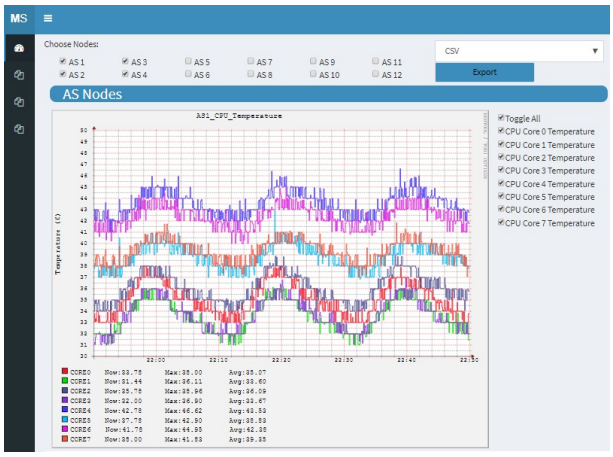


Figure 3: An example of visualization of CPU temperature for node AS1.

data that is not available anymore on the real-time data visualization page. Due to large amount of data collected on each node, this page only displays figures for a specific node at a time for reducing the workload in figure generation and rendering. However, users are allowed to switch back and forth among nodes in our system by clicking the buttons for specific nodes on the top of the page.

C. Data Management

As shown in Fig. 4, a user interface for data collection job configuration was implemented in our system.

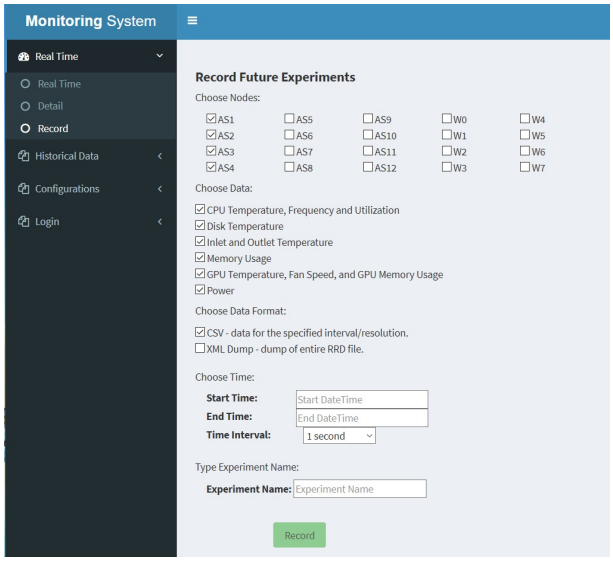


Figure 4: User interface for data collection configuration.

Users can collect data for future experiments by specifying computing nodes, and types of metrics they are interested in. CSV and XML are two file formats currently supported by our system. A unique experiment name and the execution time of experiments are the other two required configuration parameters for a data collection job. We have a validation process that verifies the parameters before a job is submitted, and an error message will be prompted if the configuration is invalid.

In addition to job submission, we also allow users to archive experimental data from our historical databases. The user interface of archiving data is the same to the one of job submission except that the start and end time must be in the time periods of our historical databases.

D. Data Comparison

A useful feature that enables users to potentially improve their systems is data comparison. Our data comparison tool supports two types of comparison. Users can either compare experimental data in different time periods on specific nodes, or compare experimental data on different nodes in specific time periods. User could choose multiple data metrics in both comparisons. Figures will be generated in the server side and downloaded to client side for a shorter response time.

Fig. 5 shows an example of comparing GPU metrics among four nodes (Node 1-4) in the same time period. User could specify different time interval when comparing data from multiple nodes.

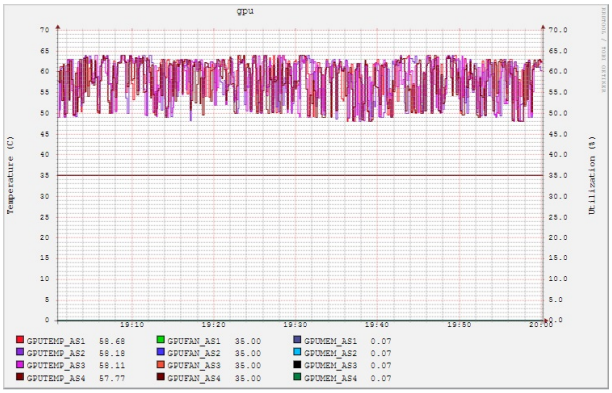


Figure 5: Comparison of GPU of multiple nodes at the same time period.

On the other hand, Fig. 6 visualizes the results of two experiments that tracks CPU temperature on a specific node. The two experiments started at different times on the same node. We were interested in the comparison of the data for these two experiments in their first 10 minutes. Our system automatically aligns the experimental results by the start time for a fair comparison.

E. User Management

Our system only accepts registered users for data archiving, and the experimental results of a user are managed

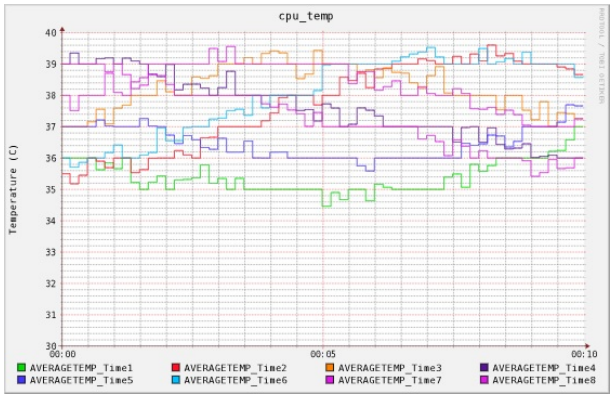


Figure 6: Comparison of CPU temperature in different time period on the same node.

independently to the ones of other users. Users can only view and download the archived experiments performed by themselves after log-in; while administrators can access all data in the system.

Fig. 7 displays an example of searching experimental data limited by the execution time of experiments. A logged in user specifies the start time and/or end time, all archived experiments that happened between these two time will be shown. Fig. 8 shows the experimental results filtered by keywords. These two searches are performed by an administrator; so that the unique experiment names, usernames, and experiment configuration parameters are displayed in the search result. Furthermore, user could also search for experiments by specifying multiple searching criteria: the keywords, start time and end time.

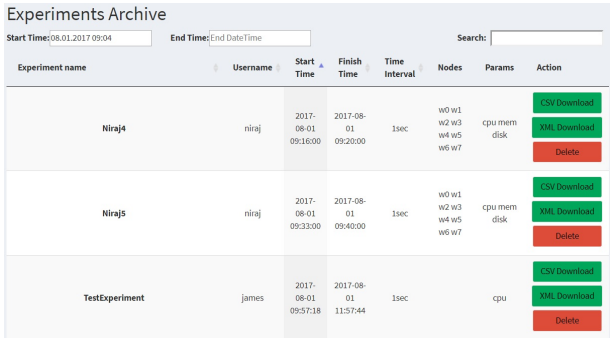


Figure 7: Search experimental results by specifying the start time of the experiments.

In addition, as shown in Fig. 9, the search results can be sorted by any of the experiment name, username, the start or finish time for convenience.

V. Conclusion and Future Work

In this paper, we proposed a cluster monitoring system that tracks the performance and temperature of major computing components in cluster systems. This system not only allows users to collect experimental data in a convenient way, but also enables them to compare their experimental results for potential system improvement. Two types of data comparisons are supported by our system. Users can either compare data of specific metrics in different time periods on the same node, or compare data of specific metrics in a specific time period on different nodes. All experimental data and comparison results are visualized by figures auto-

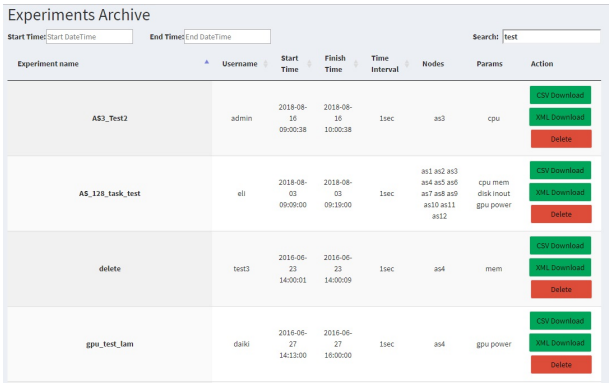
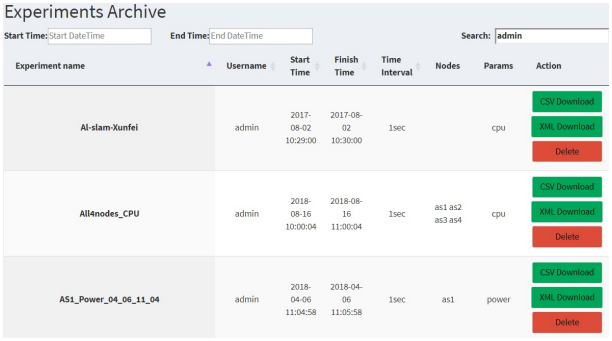
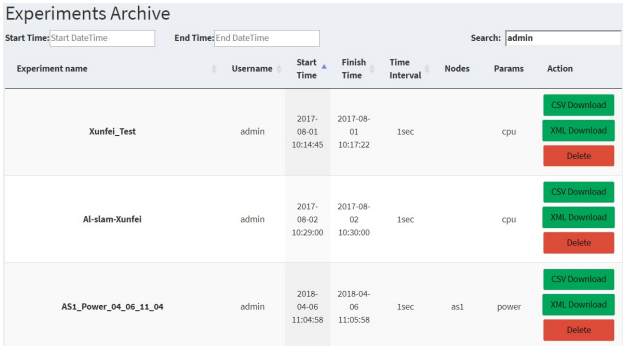


Figure 8: Search experimental results by using keywords.



(a) An example of sorting the search result by experiment names in an ascending order.



(b) An example of sorting the search result by start times of experiments in an asending order.

Figure 9: Examples of sorting search results.

matically generated by our system. This system has been used in our research in the fields of energy modeling and energy efficiency improvement. It improved the efficiency of data collection significantly in our research and enabled us to do preliminary data analyze based on the real time data visualization.

In our future plan, we will improve the system by developing the following functions. (1) Add supporting of comparison for more complex scenarios (e.g., comparison of experiments on different nodes in different time periods). (2) An uploading feature will be added, which allows users to upload data in compatible formats for online comparison and analysis. (3) More administration functions will be developed. For example, administrators of the system can create database files or launch background metric data collection programs through the web-based interfaces. (4) Display advanced information to assistant the online analysis of real time data. (5) Extend the data management module to support the management of recording jobs and archiving jobs. For example, recording jobs that has not been started could be paused/resumed or canceled by users who created them or by administrators. When a recording job is finished, a notification will be sent to the user who created this job. For archiving jobs, an estimation of the job execution time will be prompted to the user so that no repeated job status check is needed from the user.

ACKNOWLEDGMENT

This research was supported by Harry Todd Costello Fund, Lemann Student/Faculty Collaborative Research Fund, Stephen and Sylvia Tregidga Burges Endowed Research Fund, Scantland Summer Collaborative Research Gift, and Ford/Knight Collaborative Research Fund from Earlham College.

REFERENCES

[1] M. Vallath, *Oracle Real Application Clusters.* Digital Press, 2004.

[2] L. A. Barroso, J. Dean and U. Holzle, “Web search for a planet: The Google cluster architecture," in *IEEE Micro*, vol. 23, no. 2, pp. 22-28, March-April 2003.

[3] U. E. P. Agency. “Report to congress on server and data center energy efficiency”. Technical report, August 2007.

[4] M. M. Al Assaf, X.-F. Jiang, M. R. Abid, and X. Qin. “Eco-Storage: A Hybrid Storage System with Energy-Efficient Informed Prefetching”. in *Journal of Signal Processing Systems*, 72(3), pp. 165-180, 2013.

[5] A. Chavan, M. I. Alghamdi, X.-F. Jiang, J.-F. Zhang, M.-H. Jiang, and M.-K. Qiu. “TIGER-Thermal Aware File Allocation in Storage Clusters,” in *IEEE Transactions on Parallel and Distributed Systems*, pp(99), 2015.

[6] X.-F. Jiang, J. Zhang, X. Qin, M.-H. Jiang, and J.-F. Zhang, “Thermal Modeling and Management of Storage Systems in Data Centers,” in *Handbook on Data Centers*. Springer, pp. 915-944, 2015.

[7] M. Bostock, V. Ogievetsky, and J. Heer, “D3: Data-Driven Documents " in *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011).

[8] RRDtool. “https://oss.oetiker.ch/rrdtool/.”

[9] ganglia. “http://ganglia.sourceforge.net/.”

[10] M. Li and Y. Zhang, “Hpc cluster monitoring system architecture design and implement,” in *2009 Second International Conference on Intelligent Computation Technology and Automation*, volume 2, pp. 325-327, Oct 2009.

[11] C. L. Moore, P. S. Khalsa, T. A. Yilk, and M. Mason, “Monitoring high performance computing systems for the end user,” in *2015 IEEE International Conference on Cluster Computing*, pp. 714-716, Sept 2015.

[12] A. Gómez-Iglesias, C. Rosales, and T. Evans, “Practical Monitoring of Resource Utilization for HPC Applications,” in *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale (XSEDE16)*. ACM, New York, NY, USA, Article 49, 8 pages.

[13] T. Röhl, J. Eitzinger, G. Hager and G. Wellein, "LIKWID Monitoring Stack: A Flexible Framework Enabling Job Specific Performance monitoring for the masses," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Honolulu, HI, 2017, pp. 781-784.

[14] MiniGoose II User Manual, “http://www.itwatchdogs.com/datasheets/MiniGoose_II_User_Manual_v1_05.pdf.”

[15] nvidia-smi, “http://developer.download.nvidia.com/compute/cuda/6_0/rel/ gdk/nvidia-smi.331.38.pdf.”

[16] AdminLTE, “https://github.com/almasaeed2010/AdminLTE.”

[17] jsrrdgraph, “https://github.com/manuelluis/jsrrdgraph/.”