International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12 – No. 9, December 2017 – www.ijais.org

BMC based Data Logger for Performance Analysis

Nikhilesh Nayak
Department of Embedded Systems,
Vellore Institute of Technology,
Vellore, Tamil Nadu, India

Dheerendra Singh Tomar
Data Centre Group, PAE
Intel Corporation, Bangalore,
Karnataka, India

Shanmugasundaram M.
School of Electronics Engineering,
Assistant Professor (Senior),
Vellore Institute of Technology,
Vellore, Tamil Nadu, India

ABSTRACT

In the present scenario, a server rack has multiple platforms attached to it, each designed to perform a different set of actions, thus, having different hardware requirements. To increase the throughput of such a platform either the hardware requirements are multiplied or the platform is replaced completely. This unoptimized method is rather expensive and inefficient. [1] This paper focuses on improving the performance of a system by providing accurate analysis and predict hardware requirements to improve overall throughput. For this, data logs are collected over a period of time which take performance data dumps of sensors connected to the platform via BMC. These sensors monitor the platform and measure its internal physical parameters. This data is then used to create a database and a training set. This set is used to train a machine learning algorithm which gives an efficient algorithm to analyze the present performance and give accurate prediction. This gives an optimal solution to increase throughput of a platform. [2]

General Terms

Machine learning, data logs, BMC, performance analysis and IPMI.

Keywords

Data logger, sensors, python, IPMI, BMC, machine learning, database and performance.

1. INTRODUCTION

A platform's performance depends heavily on the present condition of its hardware. For example temperature, fan speed, thermal trip, voltage and current readings etc. An aggregate of these parameters decides the current throughput of the platform. The current value of these parameters are measured by sensors built into the computer system and reported to the BMC. Among these sensors, some collect data on current asserted states which indicate whether any internal or catastrophic error has occurred.

A BMC (Baseband Management Controller) is a microcontroller embedded into most server boards and serves as the heart of the IPMI architecture by providing features like autonomous monitoring and recovery implemented directly into platform management system. [3] BMC monitors the platform for critical events provided by sensors and sends an alert or creates an event log when specific parameters exceed their given threshold. And thus corrective measures are taken by the user. BMC supports IPMI 2.0 for Intel Corporation server boards and platforms. [4]

The intelligent platform management interface (IPMI) allows for independent monitoring, logging and recovery implemented directly into platform management hardware and software. [5] IPMI works by interfacing with BMC, thus further enhancing its management capabilities on servers as well as allowing administrator level control like remote access to BIOS, OS console information and power management of servers. It also enables access to platform management information like remote access through LAN, inter-chassis access through Intelligent Chassis Management Bus, local access through system management software etc. [6] As it isolates the System Management Software from the hardware, hardware advancements can be made without interrupting the software.

Table 1: IPMI Functions

Sl. No.	Function	Description
1	sensor get list	List of all sensor names, current value and range
2	sensor get "sensor ID"	Display information on single sensor
3	sdr get "ID"	Get all asserted and deserted states information on a sensor
4	sdr info	Collect sensor data repository information
5	sdr list	List out all sdr assertions logs
6	sdr clear	Clear sdr logs
7	session info	Identify current session and get all information
8	sel last <count>	Get last <count> logs of sel

TensorFlow is an open-source library used for machine learning applications. [7] It creates a graphical dataset from a set of inputs and arranges them into clusters. This set of data is called training data and it's a subset of the database. After the algorithm is trained, a set of validation data is used for algorithm validation. This final algorithm is then applied on new data to get the desired output. Thus, improving efficiency and predictability. [8]

2. PERFORMANCE ANALYSIS

2.1 Data logging and Database

An application is created using python which collects sensor logs from various sensors connected to the platform over a period of time. This application is run as a service and is always collecting data in the background. These data logs are fetched using OpenIPMI tool which connects to the platform IP and keeps a log of this data. The sensor data is captured continuously with a ten minute interval between each log so that data is neither constant nor drastic. These logs are then stored in a (.csv) file for later analysis and training. When new data is taken again, it's appended in the same file. Once the database is created, it's divided into three sections: training samples, validation samples and new data samples as seen in figure 1.

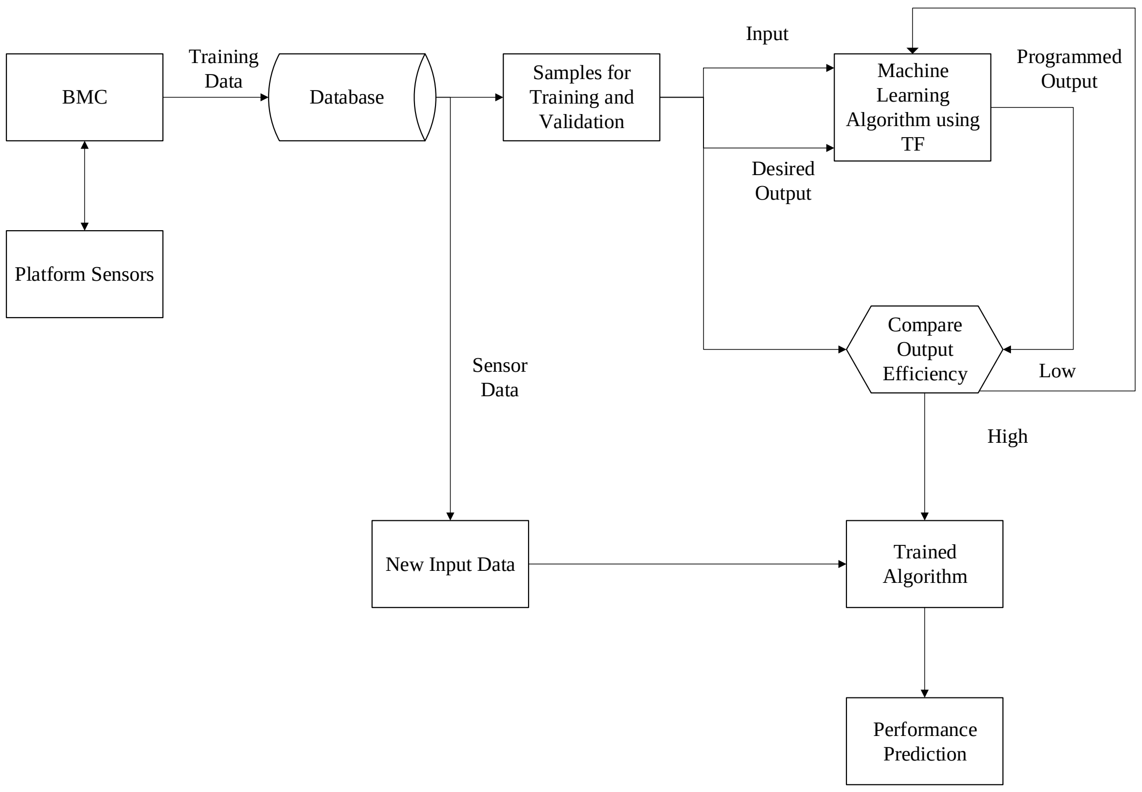

Fig 1: Performance Analysis of Platform using Machine Learning

2.2 Sensors required

Table 2: Sensor Range

Sl. No.	Name	Range
1	Processor 1 Temperature	40-50 °C
2	Processor 1 Memory Thermal Trip	20-30°C
3	Processor 1 Voltage Output	2.12-3.42 V
4	Processor 1 Current Output	20-60 %
5	Input Power	100-1428 W
6	Processor 2 Temperature	40-50°C
7	Processor 2 Memory Thermal Trip	20-30°C
8	Processor 2 Voltage Output	2.12-3.42 V
9	Processor 2 Current Output	20-60%
10	Front Panel Temperature	20-50°C

Sensor data is obtained using the sensor get "Sensor ID" function under IPMI tool. As each platform can have a different set of names for the respective sensors, sensor IDs are used to collect the data instead of sensor names. [9] As IPMI allows automatic reconfiguration of itself based on platform capabilities it eliminates the need for platform specific configuration. The sensor data records contain information on type of sensors, number of sensors, sensor threshold, type of reading, current sensor reading and range of sensor reading for each sensor.

2.3 Training and validation of machine learning algorithm

To measure the throughput of a platform based on data from multiple sensors a python script is used. This utilizes TensorFlow library to train the machine learning algorithm and create multi-dimensional clusters of data, where each sensor parameter represents a dimension. The cross-section of these dimensions gives the current throughput of the platform. A set of training data samples is used to train the system. The programmed output from the algorithm is then compared with the desired output over multiple epochs. The accuracy and efficiency of the algorithm is then calculated and if found to be below a certain threshold, the algorithm is trained again. After the algorithm is fully trained, validation samples are used to test the efficiency of the programmed output.

Machine learning algorithms differ from conventional algorithms as here instead of setting our own constraints on the algorithm and passing an input through it, an algorithm is designed and trained on itself over an initial set of data. [10] The final algorithm then replaces the program in a conventional algorithm as seen in figure 2.

The learning method used in machine learning can either be supervised or unsupervised. [11] In supervised learning the inputs and desired outputs are labelled and the algorithm is trained to find the relation between the two and then compare new unlabelled outputs with the taught ouputs to find a suitable solution. For training the algorithm a set of data is used and clusters of data are formed and categorised into multiple subgroups based on type of output. On the other hand unsupervised learninig has unlabelled data and the algorithm has to find common points among the input data. Its generally used for transactional data where its not easy to find a common relation between two inputs. Here no desired output is provided. Thus, it can look at more complex data and organise it in a meaningfull manner. [12] In this paper labelled data with sensor inputs and desied outputs is used so supervised learning is applied.

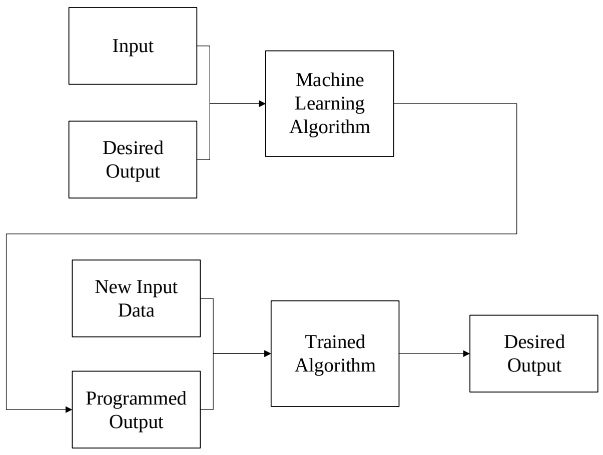

Fig 2: Machine Learning Algorithm

Python has wide variety of machine learning libraries available like TensorFlow, Keras and PyTorch. And thus proves to be effective for processing data. Keras is a high level neural network API, and can run on top of TensorFlow. Thus, making it a wrapper which allows for fast prototyping and runs seamlessly on CPU as well as GPU. [13] PyTorch on the other hand is designed to be fast and a complete re-written version of TensorFlow. But it's still in beta and a lot of the functions are not supported yet. Also while TensorFlow uses data statically by defining a graph before a model can ran, PyTorch takes a dynamic approach as data can be added on the go. [14] In this paper TensorFlow is used to implement machine learning algorithm on the database.

Here the input data is labelled as sensor data and throughput and the goal is to learn a general rule that maps relation between the two. As this has two class values, it requires binary classification. Thus, a logistic regression algorithm for machine learning is used. [15] The inputs are divided into two or more classes and the learner produces a model that assigns unseen inputs to these classes. Under TensorFlow as the data log have continuous numerical values, they are specified by means of an input builder function. [16] This function is then passed to "tf.estimator.estimator" methods like evaluate and train. These functions construct the data in the form of tensors. Tensors are a good format to represent dense data as they describe a linear relation between geometric vectors, scalars and other tensors. Once the tensors are designed, feature columns are formed from the original data. But sometimes the relation between label and feature columns is not linear. To get the fine grained relationship between the two we can use bucketization. It is the process of dividing the complete range of a continuous features into a set of consecutive packets/buckets and converting the original numerical feature into a bucket ID depending on which group it falls into.

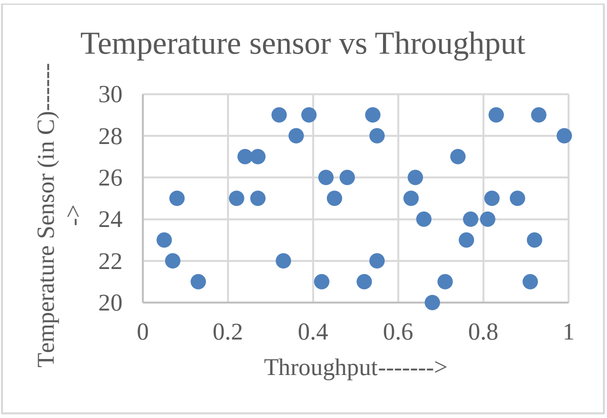

Fig 3: Temperature Sensor Data

In the first epoch of the training all sensor data points are far away from each other. After each epoch these points start converging as the Euclidean distance between them is compared and eventually over hundreds of iterations their positions become constant and they are stored and categorized in clusters as shown in figure 3. At this point the accuracy and efficiency of the algorithm is tested using new data.

2.4 Testing on new data

Once the algorithm is trained and the efficiency is over 97%, the algorithm is tested on a new set of data. This data includes the sensor input and desired throughput as seen in figure 3. If the present throughput of a system is say at 60%, to increase it to 70% any point from the 70-80% cluster is chosen which represents the exact increased requirement to achieve that throughput. The present hardware can then be replaced with the new set of hardware. This makes the algorithm a more efficient replacement to the present scenario.

3. CONCLUSION

Thus, this paper focused on improving performance of a system by applying machine learning to analyze throughput of a system and predict ways to optimally increase throughput by removing redundant hardware and adding essential hardware. Machine learning was done using TensorFlow module on python which trained a logistic regression model and created a graphical database. After the training, validation was done using a new set of data. Finally efficiency of the algorithm was checked and the process was repeated until efficiency was above 97%. The output algorithm was then used for analysis and prediction.

4. REFERENCES

[1] Intel Corporation (2012). *Intel Server Boards and Server Platforms Server Management Guide*. pp.1-7, 68-100.

[2] Intel Corporation (2017). *Intel® 64 and IA-32 Architectures Optimization Reference Manual*. 1st ed. pp.15-76.

[3] GitHub | OpenBMC. (2015). *openbmc/facebook*. [online] Available at: https://github.com/facebook/openbmc [Accessed 27 Oct. 2017].

[4] Lwn.net. (2017). *OpenBMC, a distribution for baseboard management controllers [LWN.net]*. [online] Available at: https://lwn.net/Articles/683320/ [Accessed 23 Oct. 2017].

[5] Miniard, C. and Montavista Software (2006). *IPMI – A Gentle Introduction with OpenIPMI*. 1st ed. pp.9-24, 89-94,115-143.

[6] Intel Corporation, Hewlett-Packard Company, NEC Corporation and Dell Inc. (2013). *Intelligent Platform Management Interface Specification Second Generation*. 2nd ed. pp.12-26, 207-215, 419-447.

[7] TensorFlow. (2017). *TensorFlow Linear Model Tutorial*. [online] Available at: https://www.tensorflow.org/tutorials/wide [Accessed 1 Nov. 2017].

[8] The Practical Dev. (2017). *Design Philosophy - Introduction to Tensorflow Part 1*. [online] Available at: https://dev.to/kasperfred/design-philosophy-of-tensorflow---introduction-to-tensorflow-part-1-ajp [Accessed 20 Sep. 2017].

[9] Intel Corporation (2007). *Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*. Intel Corporation. August. pp. 14–1

[10] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach.* Springer Science & Business Media, 2013.

[11] Agarwal, P. (2016). Machine Learning Toolbox. *Machine Learning and Applications: An International Journal*, 3(3), pp.25-34.

[12] Smola, A. and Vishwanathan, S. (2010). *Introduction to Machine Learning*. The Press Syndicate of the University of Cambridge, pp.165-194.

[13] Keras.io. (2017). *Keras Documentation*. [online] Available at: https://keras.io/#keras-the-python-deep-learning-library [Accessed 20 Oct. 2017].

[14] Towards Data Science. (2017). *PyTorch vs TensorFlow—spotting the difference – Towards Data Science*. [online] Available at: https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-difference-25c75777377b [Accessed 18 Oct. 2017].

[15] Brownlee, J. (2017). *Logistic Regression for Machine Learning - Machine Learning Mastery*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/logistic-regression-for-machine-learning/ [Accessed 2 Nov. 2017].

[16] TensorFlow. (2017). *Getting Started With TensorFlow | TensorFlow*. [online] Available at: https://www.tensorflow.org/get_started/get_started [Accessed 8 Oct. 2017].