Anomaly Detection in BlueGene/L Supercomputer Logs Using Random Forest Ensemble Machine Learning Algorithm

Mehul Bhargava
Master of Data Science
University of British Columbia
Kelowna British Columbia Canada
mehulbhargava@gmail.com

Navdeep Singh Saini
Master of Data Science
University of British Columbia
Kelowna British Columbia Canada
navdeepsingh3094@gmail.com

ABSTRACT

Data is growing at a high pace, which causes problems like slow computational speed, storing, processing, or transporting data, and eventually the reason for the development of supercomputers. One of them is BlueGene/L which can reach high operating speed, with very less power consumption. It's not easy to detect anomalies in such a large computer system. This paper focuses on the development of a model to automate this process of detecting anomalies. We have worked on an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) in Livermore, California, with 131,072 processors and 32,768GB of memory. The log contains alert and non-alert messages identified by alert category tags. This paper describes the process of parsing the log data, data exploration, modeling, and finally coming up with the results of anomaly detection which tells if a message in the log anomaly is or not. Since this is a classification problem, we have focused more on classification algorithms like Random Forest.

Key words: BlueGene, data, algorithm, random forest, logs, anomaly.

1. INTRODUCTION

Anomaly detection in BlueGene supercomputer logs is a task to develop a machine learning model which will classify the log messages into anomaly and non-anomaly.

Anomaly detection is the detection of abnormal behaviour in a dataset. It includes the identification of rare items [6], observations or events which are totally different from most of the data and are unusual. Here, we have classified the data as anomaly or non-anomaly.

BlueGene is a supercomputer developed and designed by IBM that can reach operating speeds in the petaFLOPS (PFLOPS) range, with low power consumption. It can accommodate as many as 128K processors.

Dataset: BlueGene logs dataset used in this project is the data with the log messages generated by the supercomputer with different properties like id, time, date, etc. It is an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) in Livermore, California, with 131,072 processors and 32,768GB of memory. The log contains alert and non-alert messages of 214.7 days identified by alert category tags. In the first column of the log, "-" indicates non-alert messages while others are alert messages [5].

In this type of data, the anomaly is less, because of this reason they are called unbalanced data. It gets difficult to apply modelling algorithms, as there are chances of improper modelling of the data and getting incorrect results. The learning algorithm performs poor in this case. The log dataset needed to be converted into a proper format to perform pre-processing and other operations on it [4], [7]. We first converted the log dataset into a data frame and exported it into a CSV file and performed various operations. The dataset consists of a total of 4,747,963 messages (rows) which is difficult to handle. We have filtered it to around 1,33,800 rows.

Goal: The goal is to classify data as anomaly and non-anomaly, so we have used a Random Forest classifier. The Random Forest classifier gives results based on the average of results from multiple decision trees to increase the

accuracy. The paper also analyses some other research related to this domain, and we have focused to develop a novel approach and improving what has already been done.

2. LITERATURE REVIEW AND BACKGORUND

2.1 An Overview of the BlueGene/L Supercomputer

NR Adiga, G Almasi, GS Almasi, Y Aridor, R Barik, D Beece, R Bellofatto, and others (2002) [1] came up with research which gives us an overview of of the BlueGene/L Supercomputer. They discovered that it's a largely parallel system of 65,536 nodes and is based on a new architecture that exploits system-on-a-chip technology to deliver target peak processing power of 360 teraFLOPS (trillion floating-point operations per second). The research gives a description of control systems, system packaging, nodes, network, software support, several programming models, architecture, etc. This research doesn't provide any idea of anomaly present in the logs, and there is no machine learning model proposed to identify the anomalies and other metrices related to that.

2.2 BlueGene/L Failure Analysis and Prediction Models

Yinglung Liang, Yanyong Zhang, A. Sivasubramaniam, M. Jette and R. Sahoo (10th July 2006) [2] proposed three models for failure detection, which can predict around 80% of the memory and network failures, and 47% of the application I/O failures. They have highlighted the importance of failure detection and gave a brief description of types of failures (software or hardware). They worked on a data of more than 100 days which is RAS event logs from BlueGene/L. Basically they found the properties of failure events and correlation between fatal and non-fatal events. In the models they didn't focus much on the anomalies and their detection. It is also important to detect anomalies other than the software and hardware failures and applying various models on them.

2.3 Filtering Failure Logs for a BlueGene/L Prototype

Y. Liang, Y. Zhang, A. Sivasubramaniam, R. K. Sahoo, J. Moreira and M. Gupta (25th July 2005) [3] suggested a model for failure detection and reducing the logs data set, for more convenient and efficient computations on it. They worked on the logs data of around 84 days and performed a three-step algorithm for categorizing and extracting failure events. Applied various types of filtering on the data like temporal filtering for removing duplicate reports and eventually removing failure reports of the same type of error across different locations. Using this approach, they were successfully able to reduce the logs by removing over 99.96% of the original entries. The filtering methods they have used are more time consuming and not generic.

2.4 Fault-aware Job Scheduling for BlueGene/L Systems

J. Oliner, R. K. Sahoo, J. E. Moreira, M. Gupta and A. Sivasubramaniam (2004) [4] put up two new job-scheduling algorithms that are more effective than previously developed algorithms. They have developed these algorithms considering failures while scheduling the jobs. They have also evaluated the impact of these algorithms on average bounded slowdown, system utilization, and average response time. They observed and concluded that their study and development of the new algorithms was successful for improving the performance of the BlueGene/L system. This research can be improved by also covering other system software and programming environment parameters, including operating system and memory management parameters while making scheduling decisions.

2.5 What Supercomputers say: A Study of Five System Logs

Adam Oliner and Jon Stearley (2007) [5] did research on five different supercomputer logs, with the aim that the research will be useful for future research on these logs. They have worked on BlueGene/L (131072), Thunderbird (9024), Liberty (512),), Spirit (1028), and Red Storm (10880). They have provided the basic idea for all of the five logs, the efficient and effective way to collect this data, ways to identify alerts, filtering algorithms, and analysis. The research was proposed in 2007 and came out to be very useful in many research and projects.

2.6 Background: There is always a need for a novel idea and an advanced approach for those tasks which are not completely accomplished. Analyzing other related works and looking at the need for models for anomaly detection in the era of growing data, we developed this project.

3. MOTHODOLOGY AND WORKFLOW

The workflow consists of the following steps:

3.1 Exploring the Dataset

For our research project we selected the BlueGene Logs Dataset. It is an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) in Livermore, California, with 131,072 processors and 32,768GB memory. The log contains alert and non-alert messages identified by alert

category tags. In the first column of the log, "-" indicates non-alert messages while others are alert messages. The label information is amenable to alert detection and prediction research. It has been used in several studies on log parsing, anomaly detection, and failure prediction.

3.2 Loading/Log Parsing and Dataframe Conversion

For reading the log dataset, it was first parsed via UTF-8 Encoding converting into the list object. After the list object, it is then converting into the two-dimensional Data frame object. The converted Data Frame Tabular datatype is further wrangled to drop some of the unnecessary columns. The columns which were finally selected for future data processing were the anomaly status, the Message or the Block ID, the message for that load. The resulting set was then loaded into the excel sheet.

3.3 Excel Sheet Operations

For our research project we selected the BlueGene Logs Dataset. After loading the filtered data into the excel, the next step is to use extract some keywords by looking at the messages and exploring. Based on that, we came across 15 keywords, and we used Excel to count the frequencies of those individual for every message in that data points. Therefore, we got to know the occurrences which will be using as the predictor or the input variables for our model to run.

3.4 Advanced Wrangling and Final Data Preparation

The next process is to again reload the dataset into the python code editor with the additional frequency column for the selected words. The key concept here is we need to group those words based on the Common Block ID as we need to run our model based on sequence of events which are under one common message or Block ID. We used predefined Python libraries and functions like group by to achieve that and then our dataset was ready for running the model where our Predictors were the count of occurrences for the unique block ID and the anomaly status as the response.

3.5 Model Selection

As the response in our project is whether the system is either an anomaly or not an anomaly, so it is basically a typical classification machine learning problem. In our Statistical and Machine Learning Paradigm, we have various predefined classifiers such as K-Nearest Neighbors, Support Vector Machine, Bagging, Boosting, Random Forest or even Multiplayer Perceptron's, Convoluted Neural Networks. Out of above common models, we first started with the Random Forest Classifier for training our dataset and luckily it worked well at one go.

3.5.1 Random Forest Advantages

- It reduces variances and overfitting problems as it adds randomness to the models with a greater number of random trees.
- It can be used for both classification and regression problems.
- The decision tree of node splitting is based on the subset of features not on all features, leading to more diverse trees.
- As the dataset was quite huge in our case, so we were worried about feature scaling and normalization, but in case of Random Forest it's not required as its not based on distance calculation.
- It is not impacted that much by outliers, as this is the real-world dataset, it can be impacted by outliers, but in case of Random Forest it may impact on one tree but not on all combination of trees.
- As we selected bunch of keywords based on our intuition, it may have some nonlinear or weird relationship which can be a cumbersome task to deal with some of the models, but in case of Random Forest, this doesn't seem to be an issue and it can outperform many other curve-based algorithms.

3.6 Model Training

The major work in the whole project is to split the dataset on the training and testing dataset and then further train the model on the training set, and further test the accuracy on the test set. We used the simple approach; however, we could have used the validation set instead of the test set for tuning our parameters, in deciding on the number of trees, but this was the case in case our test accuracy was low so we would have introspected ways to improve the model by running it on maybe 10 CV fold for further tuning of the parameters. But at first go, we went ahead just by training the model by splitting it into 75 to 25 ratio meaning 75% to the training set and the remaining 25% to the test set. For training the model we used the function random forest classifier, which is part of sklearn Python package, we will then use the fitted model for test our predictions on the test set and use it further for calculating the accuracy score or if further needed confusion matrix or other accuracy metrices like precision, recall F1-Score.

3.7 Model Accuracy

When we ran the model on the test set predictions and compared it for our test accuracy, we got the accuracy score of around 99.14%. This seems to be good at first glance, but we need to note in our whole dataset, the number of anomalies is less, around 20% of the total

observations, so in the training set. Therefore, it's kind of unbalanced set-in terms of response. Thus, to make sure our model gives accurate predictions on the future sets, we needed to compute confusion matrix and other classification scores like Precision, Recall and F1-Score.

4. CONCLUSION AND RESULTS

If we see the above results, we can see that the Precision, Recall, F1-Score are also looking accurate, also the confusion matrix is able to predict around 99% of the anomalies on the test set which looks quite brilliant. Refer Figure 1 and Figure 2. This model overall working fine on the current dataset. However, as part of future scope, we can try running other models like Deep learning/ Neural Networks to test whether it can give better results or not.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	14226
1	0.95	0.99	0.97	2298
accuracy			0.99	16524
macro avg	0.97	0.99	0.98	16524
weighted avg	0.99	0.99	0.99	16524

Figure 1 : Result image 1

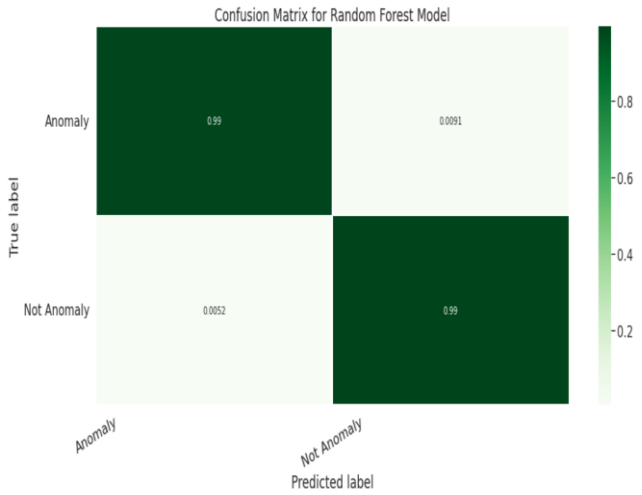

Figure 2 : Result image 2 – Confusion Matrix

REFERENCES

[1] N. R. Adiga et al., "An Overview of the BlueGene/L Supercomputer," SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, 2002, pp. 60-60, doi: 10.1109/SC.2002.10017.

[2] Yinglung Liang, Yanyong Zhang, A. Sivasubramaniam, M. Jette and R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models," International Conference on Dependable Systems and Networks (DSN'06), 2006, pp. 425-434, doi: 10.1109/DSN.2006.18.

[3] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. K. Sahoo, J. Moreira and M. Gupta, "Filtering failure logs for a BlueGene/L prototype," 2005 International Conference on Dependable Systems and Networks (DSN'05), 2005, pp. 476-485, doi: 10.1109/DSN.2005.50.

[4] A. J. Oliner, R. K. Sahoo, J. E. Moreira, M. Gupta and A. Sivasubramaniam, "Fault-aware job scheduling for BlueGene/L systems," 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings., 2004, pp. 64-, doi: 10.1109/IPDPS.2004.1302991.

[5] A. Oliner and J. Stearley, "What Supercomputers Say: A Study of Five System Logs," 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), 2007, pp. 575-584, doi: 10.1109/DSN.2007.103.

[6] F. Allen et al., "Blue Gene: A vision for protein science using a petaflop supercomputer," in IBM Systems Journal, vol. 40, no. 2, pp. 310-327, 2001, doi: 10.1147/sj.402.0310.

[7] K. D. Ryu et al., "IBM Blue Gene/Q system software stack," in IBM Journal of Research and Development, vol. 57, no. 1/2, pp. 5:1-5:12, Jan.-March 2013, doi: 10.1147/JRD.2012.2227557.