Designing PhelkStat: Big Data Analytics for System Event Logs

Mohammed Salman*, Brian Welch* and Joseph Tront*
*Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg

David Raymond†, Randy Marchany†
† IT Security Office
Virginia Tech, Blacksburg

***Abstract*—With wider adoption of micro-service based architectures in cloud and distributed systems, logging and monitoring costs have become increasingly relevant topics of research. There are a large number of log analysis tools such as the ELK(ElasticSearch, Logstash and Kibana) stack, Apache Spark, Sumo Logic, and Loggly, among many others. These tools have been deployed to perform anomaly detection, diagnose threats, optimize performance, and troubleshoot systems. Due to the real-time and distributed nature of logging, there will always be a need to optimize the performance of these tools; this performance can be quantified in terms of compute, storage, and network utilization. As part of the Information Technology Security Lab at Virginia Tech, we have the unique ability to leverage production data from the university network for research and testing. We analyzed the workload variations from two production systems at Virginia Tech, finding that the maximum workload is about four times the average workload. Therefore, a static configuration can lead to an inefficient use of resources. To address this, we propose PhelkStat: a tool to evaluate the temporal and spatial attributes of system workloads, using clustering algorithms to categorize the current workload. Using PhelkStat, system parameters can be automatically tweaked based on the workload. This paper reviews publicly available system event log datasets from supercomputing clusters and presents a statistical analysis of these datasets. We also show a correlation between these attributes and the runtime performance.**

***Index Terms*—Log Analysis, Data Mining, Cloud Computing, Cybersecurity, ELK Stack.**

I. Introduction

Log analysis and monitoring involve extracting important log information to identify key temporal system events. Cloud service providers such as IBM, Amazon Web Services, and Red Hat make use of log analysis extensively. Additionally, log analysis is an indispensable tool in the area of cybersecurity. With hackers using sophisticated tools and algorithms to try to break into systems, it is important for organizations to devise increasingly effective ways to protect against such attacks. Some of the most reliable and accurate tools in cybersecurity are the event and audit logs created by network devices. While log analysis initially started out as a means for troubleshooting faulty systems [1], log files are now also used for monitoring system and network activity [2].

Micro-service architectures have become quite prevalent in the IaaS (Infrastructure as a Service) industry. With the pay-as-you-go model becoming widely adopted, where the consumer pays based on resource usage, service providers must adopt a fine-grained monitoring approach. Service providers need to allocate resources (compute capacity, network transfer, persistent storage, etc.) to carry out this fine-grained monitoring, increasing the overhead of hosting these services. Log analysis tools are crucial to the accurate monitoring of these large, distributed systems.

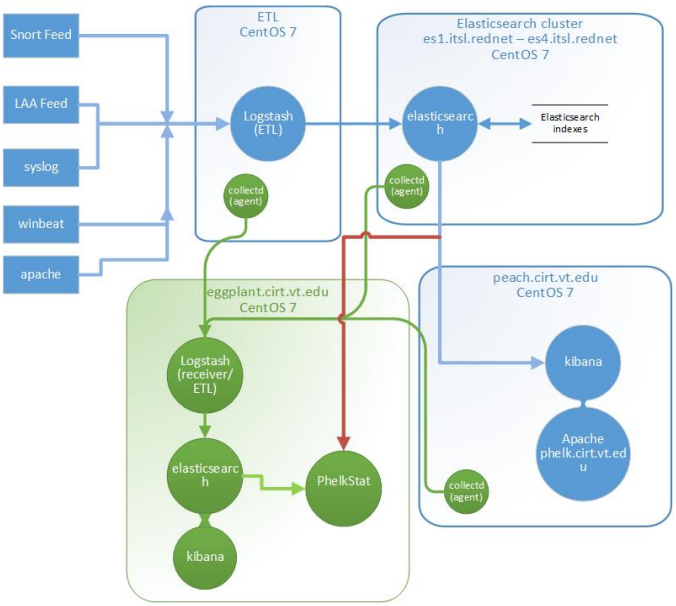

Fig. 1. Phelk Setup at ITSL

With increasing emphasis on cybersecurity, log files are now increasingly being used for threat diagnosis and mitigation [3]. Network traces and system event logs are used to build anomaly detection tools. The 2016 Internet Security Threat report [4] from Symantec shows the number of exposed identities at 423 million in the year 2015, up by 143% from 2014. More importantly, the number of web attacks blocked per day was 1.1 million, which increased by 117%. Predictive algorithms, which employ machine learning and deep learning, are increasingly used to protect against attacks. These tools use logs as the training data and predict anomalies and threats based on real-time log data.

Building log analysis tools to extract semantic log information is an extremely challenging problem. At the Information Technology Security Lab (ITSL) at Virginia Tech, we work at the intersection of big data and cybersecurity. To further

cybersecurity research, we have deployed an Elasticsearch, Logstash, and Kibana (ELK) cluster, as shown in Figure 1, to stream data from the production network at Virginia Tech. Data sources include snort data, wireless network associations, and syslog data from servers maintained by the Office of Networking and Information Systems. These data feeds are fed into a Kafka store before being ingested into Logstash, which performs Extract, Transform, and Load (ETL) operations on the data. The filtered data is then fed to an Elasticsearch cluster where it can be queried. Kibana serves as the visualization tool for these feeds. The runtime performance of the ELK Cluster (CPU utilization and I/O) is tracked using the collectd daemon, which sends the resource utilization parameters to a separate ELK instance. The current setup does not contain the PhelkStat implementation and the block shown in the diagram is where we intend to instrument PhelkStat in the setup. One of the main challenges we face is optimizing the performance of the Elasticsearch cluster and exploring other ingestion tools, such as Fluentd, in order to optimize system performance.

Log analysis is an extremely challenging problem and it is impossible for us to propose a general solution. We focus on system event logs in particular with these objectives in mind:

- Survey existing publicly maintained event log datasets; in addition to providing us training data for PhelkStat, it also serves as a collection of datasets which other researchers may find useful in the future.
- Characterize system workloads by identifying quantitative attributes such as arrival rate, event log size, event criticality level, and content of anomalous data.
- Develop a model to find the optimal configuration parameters based on the workload.
- Generate log templates using NLP techniques to create artificial, but realistic, datasets.
- Extract user-sensitive fields such as host names and IP addresses, which could be subsequently used by anonymization tools to sanitize datasets.

II. Background

We collected one month of system event log data (October 2016) from the Advanced Research Computing (ARC) group at Virginia Tech. ARC provides compute clusters to perform research. The event log data is from the New River system, a 134 node cluster with the following technical specifications:

TABLE I
ARC CLUSTER(NEW RIVER

Compute Engine	No of Nodes	Type of CPU	Cores	Memory
General	100	2 x E5-2680v3 2.5GHz(Haswell)	24	128GB
Big Data	16	2 x E5-2680v3 2.5GHz(Haswell)	24	512 GB
GPU	8	2 x E5-2680v3 2.5GHz(Haswell)	24	512 GB
Interactive	8	2 x E5-2680v3 2.5GHz(Haswell)	24	256 GB
Very Large Memory	2	4 x E7-4890v2 2.8GHz(Ivy Bridge)	60	3 TB

In Figure 2, we calculated the number of event logs being generated per minute normalized to a 24 hour cycle. We posit that the arrival rate is directly correlated to the system workload. To establish the baseline, the mean arrival rate was

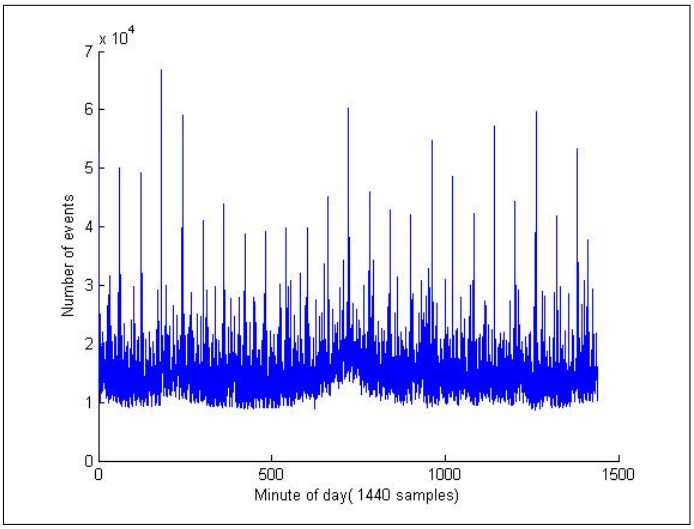

Fig. 2. Arrival Rate Distribution per minute for ARC Dataset

calculated and found to be about 6,000 samples per second. The maximum arrival rate was found to be about four times the mean value. We also evaluated the hourly and monthly variations and found the standard deviation of the workload to be about 50% of the mean value. Therefore, using the same configuration parameters without regard to the workload variation will lead to bad performance.

Another area of investigation would be the over-commitment of resources in periods where the load is less than the mean arrival rate. In a similar analysis, [5] evaluated over-commitment of resources in OpenStack and found the storage overhead to be 80%.

III. Related Work

The increasing relevance and scale of system log mining has led to a variety of approaches to derive meaning from the continual streams of logged system events. Accurate contextual log analysis has already seen practical application: Suleiman et al. describe the use of cloud log forensics to identify malicious behavior through log analysis [6].

Xu et al. detailed the strengths and weaknesses of different methods for large-scale system log processing, including tailored scripts, relational databases, and stream-based processing. Their group provided detailed analysis explaining that the temporal, real-time, and distributed nature of system log events lends itself to stream-based processing. Working from this analysis, they implemented a decision tree-based algorithm to provide service operators with correlations between log attributes. Integral to their implementation was the concept of continuous queries; they used the Telegraph Continuous Query Engine (TelegraphCQ) to perform these queries. Their work, aimed at root failure localization, provides a basis for more advanced machine learning techniques targeted at contextual analysis [7].

A large obstacle in the contextual analysis of system log events is the inconsistency in their output format. Despite

attempts to standardize log output [8], there still lacks uniformity across the many sources and hierarchical logging of modern systems. The generation of log templates allows for the distinction between variable words in logging messages and the concrete structure of the log message; Kobayashi et al. refer to these two components as variables and descriptions, respectively [9].

In order to process these different log formats and generate log templates, Natural Language Processing (NLP) techniques have been employed, such as Conditional Random Fields (CRFs). Kobayashi et al. employed a CRF-based model to attain greater than 99 percent accuracy in word-level comparison [9]. Due to the supervised nature of CRF-based algorithms, their accuracy can depend largely on the quality and variation of the provided training data; [9] noted that their model performed poorly for log events for which they had limited training data. Sufficient quantities of these training datasets can be difficult to acquire. Efforts towards dataset categorization have taken place in similar areas of research: a quantitative study of datasets has been carried out by [10], where they compare datasets in the area of computer vision and machine language.

An alternative approach to structuring log templates through NLP techniques is clustering. Clustering algorithms focus on grouping similar data [11]. The recent LogCluster algorithm, implemented and detailed by Vaarandi and Pihelgas, improves on the shortcomings of many previous clustering algorithms such as SLCT, and can be used to cluster similar log formats while identifying a disjoint set of outlier logs [12]. The real-time log data provided by current large-scale systems provides many different contexts from which analysis can be gleaned; however, each of the described techniques in this section has excelled at log analysis for a tailored purpose, such as root failure localization for a relatively narrow set of log formats [7] or the generation of log templates for data sets with a wider range of log formats [8]. These methods show promise in their respective areas of focus, but we believe they could provide a general purpose solution to the real-time log analysis dilemma if combined in some capacity.

IV. Quality Criteria for System Event logs

The quality of a dataset is heavily dependent on the system being used for data collection. Large data centers serving thousands of users are able to produce huge datasets which are adequate to sufficiently test data analysis tools. Datasets can suffer from the problem of correlated information, where the event logs contain highly redundant information. Additionally, in many cases, datasets are proprietary and cannot be accessed in the public domain. In this section, we propose a set of attributes that characterize system event log datasets. The attributes can be divided into two categories:

A. Temporal attributes

- Arrival rate distribution: The distribution of the number of samples occurring at a given instant of time. This gives us an idea of whether the events are sparsely or uniformly distributed, which can be used to characterize the system workload. Most event logs tend to be sparsely distributed with more messages being observed near an anomaly condition.
- Events size distribution: Not all event logs are of the same format. Some events specify normal routine messages while some specify critical events or contain verbose messages. We calculated the number of bytes in each event log and plotted the distribution of the number of bytes against the number of messages containing that number of bytes.

B. Spatial attributes

- Anomaly events: There is a need to carry out this analysis separately because non-anomalous events tend to contain more bytes. Furthermore, syslog events could contain multiple levels of critical events and we must estimate the number of events for each critical level.
- Message type: Event logs can be categorized into multiple categories such as sshd, kernel, or RPC, among many others. We must estimate the number of events corresponding to each category.
- Contextual log analysis: To derive semantic meaning from log messages, we can employ NLP techniques to generate log templates, using CRFs to form word associations.
- Anonymizable content: This includes the information which can be anonymized such as hostname, IP addresses, and port number. This is important because this information could be used by an anonymization tool to better tune the algorithm or to identify the actual content that needs to be optimized.

These attributes provide a set of quantitative metrics to characterize the system workload. These system metrics serve as inputs to our clustering algorithm to categorize datasets. In order to maintain a mapping between the metrics and performance, we ingest these datasets into our ELK cluster and measure runtime parameters such as CPU utilization and file I/O, among other parameters.

V. Datasets

We have collected a mix of publicly available event logs datasets and proprietary event logs from two organizations within Virginia Tech: the Office of Networking and Information Systems and the Advanced Research Computing group.

A. Publicly available datasets

The publicly available datasets being used for evaluation are shown in Table II and described below.

1) The Computer Failure Data Repository (CFDR): CFDR was started as project at CMU in 2006 to accelerate research by providing system event logs from a variety of production systems. We are using three datasets from CFDR:

- HPC4, containing three traces
  1) Spirit2

TABLE II
LIST OF PUBLIC DATASETS

Name	Time period	System type	Size(GB)	Description
Thunderbird2	2005- 2006	HPC cluster	27072	Dell System at SNL.
Liberty2	2004-2005	HPC Cluster	944	HP System at SNL.
Spirit2	2005-2007	HPC Cluster	1024	HP System at SNL.
Cray	2008	Cray systems	1.52	Event logs, console logs, and syslog from Cray XT series machines running Linux.
dartmouth/campus	2001-2004	Wireless network	1.1	Time stamped sanitized syslog records from over 450 access points over a period of 5 years.

2) Thunderbird2
3) Liberty2

A more detailed description of the dataset is give in [13].

- Cray

2) Crawdad: Started by Dartmouth University to facilitate sharing of datasets captured from wireless networks, this collection contains 119 datasets contributed by the research community. We are using the dartmouth/campus traceset [14].

B. Private Datasets

1) Advanced Research Computing (ARC): The ARC group is an organization at Virginia Tech focused on providing centralized support for research computing by building, operating, and promoting the use of advanced cyberinfrastructure. ARC maintains seven clusters, six of which contain more than fifty nodes. In collaboration with ARC, we have been able to access the system event logs of these servers for a period of one month.

2) Information Technology Security Lab (ITSL): As shown in Figure 1, the data collected consists of about twelve months of wireless and system traces. Approximately two months of data were randomly selected to carry out the analysis.

C. Design

We propose Phelkstat, a tool which calculates the attributes introduced in Section IV. Due to the disparate nature of event log formats, using a single template to parse the fields is not possible. Also, given the huge size of these datasets, we used Apache Spark as the stream-processing framework to build our tool. The experiments were performed on a four node Spark cluster. The current implementation does not generate log templates using the aforementioned CRF algorithm; rather, we generate templates based on the message types extracted from the log.

The workflow is outlined in Figure 3. Each of the functions arrival rate, arrival bytes, anomaly content and message type further generate their own RDD (Spark's Resilient Distributed Dataset) by defining their own map operations. There are three stages in the workflow:

- Stage 1: Input phase; the traceset file is loaded into an RDD. Spark also provides the option to load multiple trace files into one RDD.

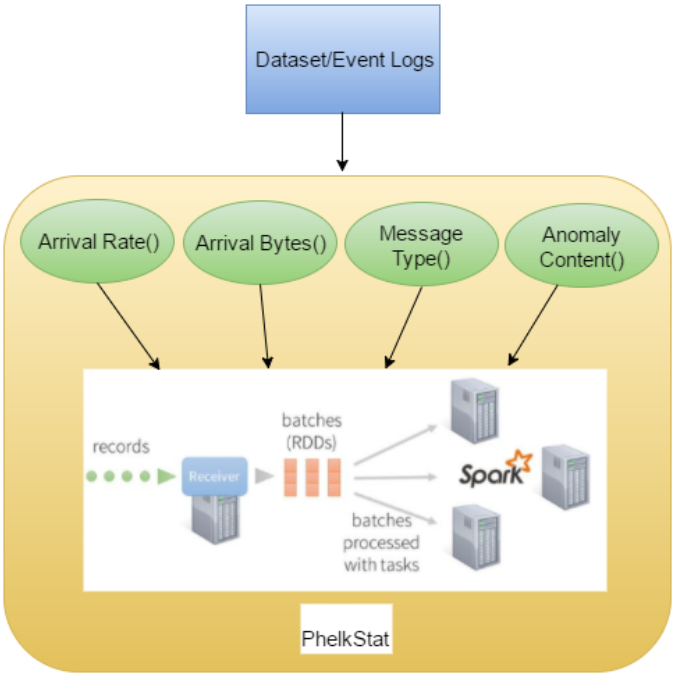

Fig. 3. PhelkStat Block Diagram

- Stage 2: Map phase; Spark stores the intermediate results in memory as RDDs. In this stage, we create separate RDDs to calculate the various attributes. The temporal attributes rely on extracting the timestamp and are evaluated by using the same RDD.
- Stage 3: Reduce phase; we combine the results from each branch of Stage 2 to aggregate the results. For the temporal parameters, we maintain a hashmap where the key is the minute of the day and the value is the attribute value. The spatial parameters are calculated by extracting the keywords from the logs.

VI. PRELIMINARY EVALUATION

A typical log event is characterized by a timestamp and an entry containing the event type, node information, and a message. An example of an event log of this format follows:

2005.01.01 sadmin1 Jan 1 00:00:07 kernel: hda: status error: status=0x00

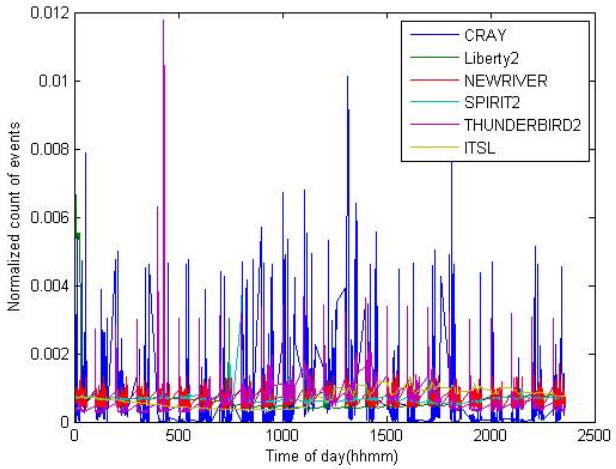

Fig. 4. Arrival Rate Distribution

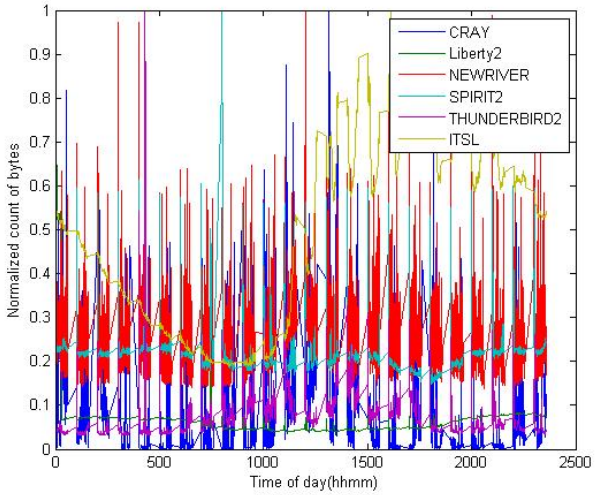

Fig. 5. Arrival Bytes Distribution

We model the event log as:

$$D = (x_1, t_1), (x_2, t_2), (x_3, t_3), ..., (x_n, tn) \quad (1)$$

where x_i represents the event log and t_i represents the timestamp, respectively.

In the estimation of the temporal attributes, we use a piecewise approach which breaks down the dataset into time sub-intervals, evaluating the attributes on each of these sub-intervals. The sub-intervals of time in a dataset D given by (1) are represented by V:

$$V_1 = Num.\ of\ samples(t_1 + \delta t)$$
$$V_2 = Num.\ of\ samples(t_1 + 2\delta t) - V_1$$
$$V = V_1, V_2, ..., V_m$$

A. Arrival Rate Distribution

This is the estimate of the number of samples for each of the sub-intervals of time. The granularity of the sub-interval is essential to maintain the normal distribution of samples. We estimate the mean value and the standard deviation of the sample distribution using the formula:

$$u = \frac{(\sum_{i=1}^{M} V_i)}{M}$$
$$Var = (V_i - u)^2$$
$$\sigma = sqrt(Var/M)$$

Figure 4 shows a plot of the event arrival rate distribution against time. As the datasets are of varying size, the count has been normalized by the total number of samples in the respective datasets. It can be seen that the Thunderbird2 system has a bursty workload with multiple variations. The Liberty2 and Spirit2 systems have similar workload variations and therefore can be grouped into the same category with regard to configuration parameters. We also see periodic stubs of activity interspersed with spikes, indicating that a load balancer has been deployed to distribute traffic among nodes.

B. Distribution of Bytes per Interval

Each event log differs in size, in terms of its number of bytes. Following a similar method to our approach for arrival rate distribution, we use:

$$B_i = Num.\ of\ Bytes(t_1 + i\delta t) - B_{i-1}$$
$$B = B_1, B_2, ...B_m, B_0 = 0$$

Figure 5 shows the arrival rate distribution in terms of bytes. The ITSL dataset has significant variation in the number of bytes arriving as the day progresses. We can infer from this that there is either some system maintenance work being performed or a set of scheduled jobs is being run. A detailed analysis of the message types at this point of time in the event log could shows us that the increase is to cron jobs being scheduled.

C. Message Type

The determination of the message type in an event log is straightforward; the message types we encountered were typically one of sshd, kernel, syslog-ng, or RPC, although we have come across other message types, as well. This information is useful for log template generation, as each message type will need to have its own template.

This message type information is maintained in a hashmap where a key is the message type (mtype) and value given by:

$$Val_k = \sum_{i=1}^{N} I(mtype = k)$$

where I(a=b) is an indicator function given by:

$$\begin{cases} 0 & a \neq b \\ 1 & a == b \end{cases}$$

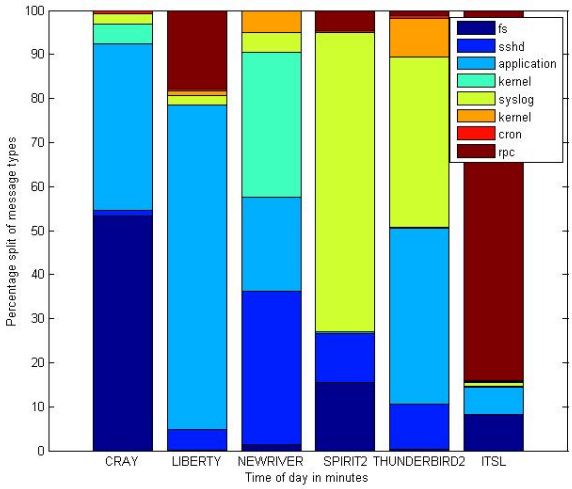

Fig. 6. Message Type Distribution

Figure 6 shows a plot of the distribution of the message type for each of the datasets previously mentioned. This information is helpful in generating artificial datasets using log templates and generating separate indices for each message type in Elasticsearch to improve performance.

D. Anomaly Content

The syslog protocol defines seven severity levels given by Emergency, Alert, Critical, Error, Warning, Notice, Info, and Debug [8]. The number of messages belonging to each category are counted for each dataset by traversing through the event logs and are outlined in Table III.

TABLE III
DISTRIBUTION OF MESSAGE CRITICAL LEVELS

Dataset	Total Values	Error	Panic	Critical	Warning	Info
CRAY	6214940	324420	14040	0	22080	5881400
LIBERTY2	265569231	4904141	20	18201	167114	160579755
NewRiver	22144269	3645133	4618	1	1236769	17257748
SPIRIT2	211212192	12722989	16	2413982	7908396	18166809
THUNDERBIRD2	272298969	76666612	102	26534	2135493	37508
ITSL	1.490667342e+09	253350	1264513232	155486153	563746	1,424,636,833

E. Runtime Analysis

In order to create a mapping between the statistical attributes and the runtime performance, we ingested the data into an ELK stack similar to the one described in Section I and measured two runtime parameters, CPU Utilization and Memory Consumption. The experiments were run for three iterations and the mean of the results has been presented here.

The effect of arrival rate and the byte rate can be explained intuitively: as the number of messages to be indexed increase, more resources are required. [15] shows that grouping messages of one type into the same index reduces the number of shards to be searched which leads to an improvement in performance.

TABLE IV
RUNTIME ANALYSIS

DataSet	Mean Arrival Rate/s	Mean CPU Utilization	Mean Memory Consumption(Gb)
CRAY	4007.416	19.72	18.084
LIBERTY2	3073.717	7.912	18.0043
NEWRIVER	2656.28	8.74163	12.9472
SPIRIT2	3151.6	7.4063	15.6784
THUNDERBIRD2	2444.583	4.5	15.44

As seen in Table IV, there is a correlation between the arrival rate and the resource consumption. There does seem to be some anomalies with regard to New River, which has a higher CPU utilization despite a lower arrival rate. The analysis presented here is only with respect to the arrival rate. We believe that the anomalies can be explained by considering other attributes such as event bytes size, message type, among others.

Another attribute that we have not taken into account is the anonymizable content. Elasticsearch can perform tokenization, where it breaks down complex strings. For example, nr-001 will be split into nr and 001, both of which will be indexed separately. The anonymized content, such as hostname and IP address, does not have to be split. PhelkStat can idenitfy anonymizable fields and the tokenization on these fields can be disabled to improve performance.

VII. Limitations and Future Work

Our current work presents a statistical evaluation of datasets along with a preliminary mapping to the runtime parameters. One of the limitations of our current framework is that once we know the optimum parameters, we need to restart the service after making the configuration changes. The next step is to tune configuration parameters of log analysis tools for a dataset and validate if that set of parameters optimizes performance with respect to another statistically similar dataset. The categorization of workloads will be conducted using unsupervised learning algorithms described in [16], [17] and [18].

With regard to implementing the Contextual Log Analysis, we plan to begin with the CRF-based algorithm used by Kobayashi et al. [9]. The LogCluster algorithm, as described in [12], will then be applied to enhance the training stage of the CRF-based algorithm. In our proposed pipeline, log training data is first processed by the LogCluster algorithm, determining clusters of log messages as well as generating a set of outliers. Additional features based on shared presence with neighboring words in LogCluster-generated clusters are then added to words in the CRF training data. Weak areas in the training data are identified by their presence in the set of outlier log events. Additionally, variation of the LogCluster algorithms support threshold parameter, which influences the number of clusters generated [12], allows for control over the impact of this pre-processing step in the trained CRF model.

VIII. Conclusion

The preliminary analysis presented here shows a correlation between the statistical parameters and the runtime evaluation after normalizing with respect to the size of the datasets.

Although the arrival rate and the byte count distribution present the same general trend, the byte distribution does show anomalies not seen in the arrival rate distribution. Using byte distribution as a feature vector in classification will help in improving accuracy. The attributes need to be assigned weights based on the type of application. For example, if the application performs Anomaly detection, it would assign a higher weight to the anomaly content attribute. On the other hand, to optimize system storage, the byte distribution would be assigned a higher priority. PhelkStat has currently been tested with ELK as the application primarily to optimize our current setup. Each application has its own configurable parameters. While optimizing for different applications does remain an area of concern, we believe that the approach we have proposed can be generalized as long as there exists a mapping between the attributes and configuration parameters of an application.As the scale of distributed systems increases, so does the need for tools that can perform anomaly and threat detection, analyze performance metrics, and provide semantic log analysis. Additionally, the real-time nature of these large-scale systems requires that these types of tools have some degree autonomy to tune their analyses to their contemporary input streams. A tool which can perform the aforementioned analyses and adequately scale with the distributed systems of cloud providers will have widespread implications, ranging from threat mitigation and detection in the cybersecurity domain to micro-service usage assessment in the cloud hosting domain. Our proposed PhelkStat will provide a base in temporal and spatial analysis from which these scalable problems can be handled.

IX. Acknowledgment

The authors would like to thank Forest Godfrey (Cray dataset) and Jon Stearley and Adam Oliner (HPC 4 dataset) for making the public datasets available. We would also like to thank the Advanced Research Computing organization at Virginia Tech for providing us event logs from their research computing cluster.

References

[1] U. Flegel, "Pseudonymizing unix log files," in *Infrastructure Security*. Springer, 2002, pp. 162–179.

[2] K. Kent, "Guide to computer security log management," 2007.

[3] A. Ambre and N. Shekokar, "Insider threat detection using log analysis and event correlation," *Procedia Computer Science*, vol. 45, pp. 436–445, 2015.

[4] "istr-21-2016-en.pdf," https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf, (Accessed on 11/23/2016).

[5] A. Anwar, A. Sailer, A. Kochut, and A. R. Butt, "Anatomy of cloud monitoring and metering: A case study and open problems," in *Proceedings of the 6th Asia-Pacific Workshop on Systems*, ser. APSys '15. New York, NY, USA: ACM, 2015, pp. 6:1–6:7. [Online]. Available: http://doi.acm.org/10.1145/2797022.2797039

[6] S. Khan, A. Gani, A. W. A. Wahab, M. A. Bagiwa, M. Shiraz, S. U. Khan, R. Buyya, and A. Y. Zomaya, "Cloud log forensics: Foundations, state of the art, and future directions," *ACM Comput. Surv.*, vol. 49, no. 1, pp. 7:1–7:42, May 2016. [Online]. Available: http://doi.acm.org/10.1145/2906149

[7] D. P. Wei Xu, Peter Bodk, "A flexible architecture for statistical learning and data mining from system log streams," in *Temporal Data Mining: Algorithms, Theory and Applications*. IEEE, January 2004. [Online]. Available: https://www.microsoft.com/en-us/research/publication/a-flexible-architecture-for-statistical-learning-and-data-mining-from-system-log-streams/

[8] R. Gerhards, "The syslog protocol," 2009.

[9] S. Kobayashi, K. Fukuda, and H. Esaki, "Towards an nlp-based log template generation algorithm for system log analysis," in *Proceedings of The Ninth International Conference on Future Internet Technologies*, ser. CFI '14. New York, NY, USA: ACM, 2014, pp. 11:1–11:4. [Online]. Available: http://doi.acm.org/10.1145/2619287.2619290

[10] F. Ferraro, N. Mostafazadeh, T.-H. K. Huang, L. Vanderwende, J. Devlin, M. Galley, and M. Mitchell, "A survey of current datasets for vision and language research," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 207–213.

[11] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in *Proceedings of the 3rd IEEE Workshop on IP Operations Management (IPOM 2003) (IEEE Cat. No.03EX764)*, Oct 2003, pp. 119–126.

[12] R. Vaarandi and M. Pihelgas, "Logcluster - a data clustering and pattern mining algorithm for event logs," in *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)*, ser. CNSM '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/CNSM.2015.7367331

[13] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. IEEE, 2007, pp. 575–584.

[14] D. Kotz, T. Henderson, I. Abyzov, and J. Yeo, "CRAWDAD dataset dartmouth/campus (v. 2009-09-09)," Downloaded from http://crawdad.org/dartmouth/campus/20090909, Sep. 2016.

[15] "Tuning data aggregation and query performance with elasticsearch on azure — microsoft docs," https://docs.microsoft.com/en-us/azure/guidance/guidance-elasticsearch-tuning-data-aggregation-and-query-performance, (Accessed on 11/23/2016).

[16] S. Balaji and S. Srivatsa, "Unsupervised learning in large datasets for intelligent decision making."

[17] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[18] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 412–419.