Abstract—Self-awareness is an important attribute for any system to have before it is capable of self-management. A system needs to have a continuous stream of real-time data to analyze to allow it to be aware of its internal state. To this end, previous approaches have utilized system performance metrics and system log data to characterize system internal state. In using system logs to characterize system internal state, the computation of strongly correlated message types is necessary. In this work, we show that strongly correlated message types can be easily discovered without much computation. Our work explores a natural behaviour of system logs where system log data partitioned using source and time information contain correlated message types. We demonstrate how the groups of partitions, which contain correlated message types, can be found by clustering the partitions based on their entropy-based information content. We evaluate our method using cluster cohesion, cluster separation and cluster conceptual purity as metrics. The results show that our proposed method not only produces well-formed clusters but also clusters that can be mapped to different alert states with a high degree of confidence.

Index Terms—Algorithms; Autonomic Computing; Networked Systems; System Management; Modeling and Assessment

I. INTRODUCTION

An autonomic computer system is a system which is, to a certain degree capable of self-management [1]. As most of today’s computing infrastructures require high availability, the need for self-managing systems becomes apparent when fault resolution is required. System administrators need to be able to resolve system problems quickly despite the complexity of the systems they manage. Having systems that can provide pointers to possible causes/solutions is therefore desirable [2], [3].

For a computer system to be capable of self-management, it needs to possess four attributes [4]. These are self-awareness, self-situation, self-monitoring and self-adjustment. Self-awareness refers to the system’s ability to be informed about its internal state. To achieve self-awareness, the system needs to be able to differentiate between its different internal states. Previous work has leveraged on either the collection of system performance metrics [2] or system logs [5], [6], [7] as sources of data, which could be analyzed by a system to make it aware of its internal state. System logs are usually good indicators of system state as they contain reports of events that occur on the several interrelated components of complex systems [3].

To use system logs to characterize internal state, it is necessary to compute the correlated message type sequences in the log. Message types are textual templates, which abstract the natural language messages in system logs. In this work, we explore the use of entropy-based information content clustering of system log partitions as a means of discovering system state. Such a system log partition will contain log data from a single source on the network over a unit period of time. Our results show that such partitioning of system logs leads naturally to partitions, which contain correlated message types, a previously unknown property of system logs. We demonstrate how these correlated message types can easily be discovered through the grouping of the log partitions into conceptual clusters using their entropy-based information content scores [8]. Conceptual clusters are clusters where objects in a cluster can be described by a concept, not just based on their distance from each other.

We evaluate our method using three measures, namely, cluster cohesion, cluster separation and cluster conceptual purity, on datasets derived from the system logs of four High Performance Cluster (HPC) systems [9]. The results show that not only are the clusters obtained by our proposed information content based clustering method well formed but they can also be described conceptually with a high level of confidence with regards to the different alert types, which have been identified by system administrators in the log data.

In the following, we discuss concepts important to understanding our work and previous work in Section 2, while Section 3 discusses entropy-based analysis of system logs in-depth. Section 4 discusses the methodology of the proposed method and the experiments, whereas the results of those experiments and the generalization of the proposed method are discussed in Section 5 and Section 6, respectively. Finally, conclusions are drawn and the future work is discussed in Section 7.

II. BACKGROUND AND PREVIOUS WORK

The goals of any automatic system monitoring application in regard to the goal of self-healing in autonomic systems...
can usually be summarized into one of three inter-related goals. These goals are System Characterization, Error/Failure Identification and Failure Prediction. To be able to create applications that will be capable of automatically monitoring systems with the above goals in mind, we must have access to continuous streams of real-time data from the systems. Such data must explicitly or implicitly contain information about the state of the system and its components. System logs meet this criterion.

A literature review shows that significant effort has been carried out by various researchers on how to use system logs for all the goals listed above [5], [6], [7]. On the other hand, Nodeinfo is an alert detection method based on the entropy-based information content analysis of system logs [10]. Our work builds on this entropy-based information content approach for system monitoring [8]. We demonstrate how an entropy based approach can be extended to the discovery of system state. Our work shows that the decomposition of messages in event logs based on source and time information naturally leads to event log partitions, which contain correlated message types. This property of event logs is the major contribution of this work. The entropy-based information content clustering of event logs makes this behaviour evident and thus makes the discovery of correlated message types, which are indicative of system states possible without much computation. To achieve this, we build conceptual clusters using the portions of the system log that have similar information content. We present this approach in more detail in the following.

III. ENTROPY-BASED ANALYSIS OF SYSTEM LOGS

Entropy-based analysis of system logs was proposed in [10] and has been employed in the task of unsupervised alert detection. We refer to alerts as events (or group of events) in a system log that are symptomatic of failure or require the attention of an administrator. An entropy based approach to alert detection has been shown to scale to large datasets [11] and achieve an F-Measure detection accuracy of up to 100% leading to an effective false positive rate (FPR) of 0% in the best case [8]. Entropy based alert detection in system logs attempts to identify partitions of the event log, which are more likely to contain alerts than others. The partitions are created by decomposing the event log using source and time information. The partitions used in our work are referred to as nodehours. A nodehour is basically one hour of log information produced by a single node on the network.

Since entropy based approaches are based on a “Similar Computers, Similar Code, Similar Logs” assumption, a logical pre-processing step is to partition the contents of the system log based on the similarity of their source. In previous work, similarity was based on the functionality of the nodes in the HPC files used in the evaluations [10], [11], [8]. After partitioning, the rest of the process is as follows:

- **Step-1:** Calculate an entropy based information content score for each term that appears in the system log.
- **Step-2:** Calculate the information content score for nodehours of the system log, based on the information content of the terms that occur in it.
- **Step-3:** Create a ranking of the nodehours based on their information content score.

There are various approaches that follow the above 3-Step process to the entropy-based analysis of system logs [8]. We utilize the NodeinfoPlus Uniq approach in our work. Details of this approach can be found in [12].

A. System State Discovery using Entropy-Based Information Content

In this section, we describe the intuition behind our method of extending an entropy-based approach to system state discovery and the datasets used in our work. In our evaluation of entropy-based alert detection, we tested the approach(es) on four HPC logs. These system logs are Blue-Gene/L (BGL), Liberty, Spirit and Thunderbird(Tbird), which are part of a set of HPC system logs, publicly available in the USENIX Computer Failure Data Repository [13]. These datasets were collected on HPC systems with varied configurations and usage patterns; hence results obtained using this data should be generalizable. Some statistics for these system logs can be found in Table I. The events in these system log datasets have been previously labelled as alerts and non-alerts by domain experts, giving us ground truth when evaluating our results. More details of the hardware architecture, configuration, characteristics, log collection methods and alert identification policies of these datasets and the systems that produced them can be found in [9].

In order to ensure the basic assumption for entropy-based analysis of event logs, i.e. “Similar computers correctly executing similar work should produce similar logs”, we separated the messages in the datasets based on the functional roles of the nodes that produced them. This results in fourteen categories. These categories are listed in the first column of Table II. The *-Other categories are not functional groupings of messages but consist of all messages that could either not be placed in any of the other categories or have ambiguous source information. The statistics of the resultant datasets is detailed in Table II.

In our evaluations, a strong clustering of nodehours around single information content score values was observed, see Fig. 1. Such clustering of values could be considered odd, since information content scores are real numbers that theoretically can take any value in the range $[0, \infty)$. The graphs in Fig. 1 show select scatter plots for nodehours from two of the datasets shown in Table II. In each graph, the y-axis represents the information content score for a nodehour using NodeinfoPlus Uniq, while the x-axis represents each individual
TABLE II

<table>
<thead>
<tr>
<th>SYSTEM LOG DATA FUNCTIONAL GROUPING STATISTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td># Events</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>BGL-Compute</td>
</tr>
<tr>
<td>BGL-I/O</td>
</tr>
<tr>
<td>BGL-Link</td>
</tr>
<tr>
<td>BGL-Other</td>
</tr>
<tr>
<td>Liberty-Compute</td>
</tr>
<tr>
<td>Liberty-Admin</td>
</tr>
<tr>
<td>Liberty-Other</td>
</tr>
<tr>
<td>Spirit-Compute</td>
</tr>
<tr>
<td>Spirit-Admin</td>
</tr>
<tr>
<td>Spirit-Other</td>
</tr>
<tr>
<td>Tbird-Compute</td>
</tr>
<tr>
<td>Tbird-Admin</td>
</tr>
<tr>
<td>Tbird-SM</td>
</tr>
<tr>
<td>Tbird-Other</td>
</tr>
</tbody>
</table>

nodehour sorted according to their information content score. We can see the clustering in all the graphs, while being most pronounced with the BGL-Link category.

![BGL_Link]

(a) BGL- Link Category

![Third_SM]

(b) Third- SM Category

Fig. 1. Scatter Plot of Nodehours (x-axis) vs. information content scores (y-axis) for select node functionality categories. The plot differentiates between alert nodehours and normal nodehours, the Third-SM category has no alert nodehours. Nodehours are sorted based on information content score in the plot.

In the following, we try to explain what could be responsible for this observation. Consider a set of distinct objects $X$, which you wish to sample (with replacement) and distribute into a number of bins (each bin acting as a bag, which can contain several instances of the same object from $X$), with the following constraints:

1) If the number of bins is $n$, then $|X|$ should be $<< n$.
2) If $Y_i$ is the set of unique objects in bin $i$, then $|Y_i|$ should be $<< |X|$ for most $i$.

If the sampling from $X$ described above is carried out even with a random sampling method, it is easy to see how we could end up with several bins containing the same set of distinct objects. Constraint 2 effectively reduces the number of possible distinct object combinations that can exist in any bin, while constraint 1 ensures that the chance for a combination to repeat itself (within a bin) is high. If the number of possible combinations of objects from $X$ given constraint 2 is less than $n$, a combination repetition is guaranteed.

If we take $X$ to be set of message types that exist in a system log and the bins as the nodehours in the system log, then we can apply the process described to the system log analysis domain with one major difference; the sampling from $X$ will follow a Pareto distribution rather than a random distribution. Previous work [5] has shown that the distribution of messages in system logs typically follows a Pareto distribution. This means that the sampling of objects from $X$ will be biased in such a way that a small subset of the objects would be sampled more frequently than others. This biased sampling should accentuate the result of having several bins containing the same set of distinct objects.

We conjecture that the process described above is responsible for the strong clustering of nodehours around a single information content score, which is in a way a hash value for the set of unique message types in a nodehour, so we can link a distinct information content score to one or more sets of unique message type combinations. Previous work suggests that temporal filtering of system log messages could be beneficial for system log analysis [14] and NodeinfoPlus_Uniq represents a form of implicit temporal filtering of system log messages.

From Table II, we can see that the constraints we described...
earlier hold true for our datasets. The “# Msg-Types” column represents \( |X| \), the “# Nodehours” column represents the number of bins while the “Msg-Types/Nodehour (Max)” and “Msg-Types/Nodehour (Avg.)” columns represent the maximum and average number of message types that can be found in each nodehour, respectively. Based on these observations, we formulate the following hypotheses:

- Nodehours with the same information content score (based on NodeInfoPlus_Uniq), contain the same unique set of message types.
- Information content scores, which occur frequently, represent nodehours, which contain strongly correlated message types. These strongly correlated message types in turn represent a system activity or state.

Our work in this paper was carried out to investigate these hypotheses.

IV. METHODOLOGY

In this section, we describe the conceptual clustering technique we developed based on the hypotheses above. This technique is a contribution of this work. We also discuss the methods to evaluate the quality of the clusters formed using our proposed technique.

A. Information Content Based Clustering

A detailed description of the information content clustering approach and the properties that differentiate it from previous approaches can be found in [12]. Essentially the algorithm creates each cluster as a bin, which can be described using the tuple \((ICS, \text{“MaxEntropyMsgType”})\), where \( ICS \) is an information content score value and \( \text{“MaxEntropyMsgType”} \) is the ID of the message type with the maximum entropy value among all the message types that have instances in the nodehour. For the first hypothesis in Sec. III-A to be true, two nodehours with the same information content score should at least have the same message type with maximum entropy, given that this message type would probably be the highest contributor to the information content score. Hence, the addition of \( \text{MaxEntropyMsgType} \) to the description. All nodehours with the same values for the tuple will end up in the same cluster.

B. Evaluations

In this section, we describe the methods we used in evaluating the quality of the clusters formed by the proposed technique. It should be noted that since the information content score associated with each nodehour is a real number, we set a precision level of 10 decimal places when testing the equality of the information content scores in our evaluations. This is important for the reproducibility of our results. We also considered only clusters, which had \( \geq 10 \) nodehours in them, arguing that clusters with fewer nodehours might only have resulted by mere coincidence. Our experiments were carried out using all the fourteen datasets listed in Table II.

We evaluated the clusters formed at the end of our experiments using three measurements, namely i.e. cluster cohesion, cluster separation and cluster conceptual purity. Our methods for deriving the cluster centroids and calculating cluster distance are detailed in [12].

Cluster cohesion : This measures the degree of similarity of the members of a cluster. This is measured using the \( F_{\text{within}} \) statistic as defined in Eq. 1. It represents the standard deviation within a cluster, using the centroid as mean. In Eq. 1, \( \mu_x \) represents the centroid of cluster \( X \), while \( x_i \) represents the \( i \)th nodehour in cluster \( X \).

\[
F_{\text{within}}(X) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} d(\mu_x, x_i)}
\]  

Cluster separation : This measures the degree of similarity between different clusters. This is measured using the Distance statistic as defined in Eq. 2. It represents the distance between two clusters \( X \) and \( Y \).

\[
\text{Distance}(X, Y) = d(\mu_x, \mu_y)
\]  

Conceptual Purity : We assume that the proposed method produces conceptual clusters, i.e. clusters linked to concepts with some level of confidence. We attempt to measure the degree to which the clusters formed meet this criterion using the Conceptual Purity measure. The datasets used in our experiments provide us the ground truth with respect to the alert concepts that exist in these datasets, in the form of the alert categories assigned to each event in the log [9]. A string is added to the beginning of each line in the log data to indicate the type of alert to which the event relates. A “-” in front of a line in the log data indicates that the event does not relate to any know alert type. To this end, we use this information to measure the conceptual purity of the alert clusters, i.e. clusters that contain a majority of alert nodehours. The process defined in Algorithm 1 determines the ratio of nodehours in an alert cluster, which contain the signature for an alert category. It measures the degree to which the alert nodehours in a cluster can be linked to alert category signatures, which appear in nodehours in the cluster. A value of 1 for this ratio implies that all the alert nodehours in a cluster can be linked to the same alert categories. Therefore the cluster can conceptually be linked to the alert categories with 100% confidence. We can therefore repeat this process for all alert categories whose signatures appear in the nodehours of a cluster and do the same for all the clusters. The average of these values represents the conceptual purity with respect to the alert categories defined in the system log. In this case, it is not our intent to detect alerts. We use alerts to evaluate conceptual purity with respect to the available ground truth. We are unable to do this for non-alert clusters as no ground truth exists for non-alert system states in the data sets evaluated. However, the cluster cohesion and cluster separation evaluations provide strong evidence to support the position that results obtained for the evaluation of alert states can be extended to normal states. We can also
confirm that the majority of nodehours in non-alert clusters do not contain alerts.

Algorithm 1 This pseudo-code describes our method for determining the degree to which the signature of an alert category can be linked to an alert cluster.

**Input:** Set $M_{alert}$ of messages types that can be linked to an alert category
Cluster $S_{alert}$ whose conceptual purity with respect to the alert category we want to determine.

**Output:** Alert Category to Alert Cluster Ratio. [Range [0,1]]

```plaintext
1: ratio_sum = 0
2: count = 0
3: Determine the number $n$ of nodehours in $S_{alert}$
4: for each message type $m$ in $M_{alert}$ do
5:  Determine the number $x$ of nodehours in $S_{alert}$ that contain $m$
6:  if $x > 0$ then
7:     temp_ratio = $\frac{x}{n}$
8:     ratio_sum = ratio_sum + temp_ratio
9:     count++
10: end if
11: end for
12: cluster_ratio = $\frac{ratio\_sum}{count}$
13: Return(cluster_ratio)
```

V. RESULTS

The results generally show that by using our proposed information content based clustering method we were able to produce well formed and conceptually meaningful clusters. Detailed statistics of the clusters formed using the proposed approach can be found in [12]. The results show that on average, we are able to cluster ~92% of nodehours into clusters of size $\geq 10$ using this approach.

In the following sections, we discuss our results on the goodness of the clusters formed using (i) internal measures (cohesion and separation) and (ii) ground truth (alert categories). Finally, we discuss how the results validate our hypotheses.

A. Internal Measures

The results also demonstrate that on average the clusters formed show an average $F_{within}$ measure of $\sim 0.01$ indicating tightly formed clusters and an average $Distance$ measure of $\sim 0.82$ indicating that the clusters are well separated. Details of the $F_{within}$ and $Distance$ results for each of the datasets can be found in [12].

B. Ground Truth

The ability of our approach to separate alert states from normal states for select node categories is illustrated in Fig. 2. In each graph in Fig. 2, each line on the y-axis represents a nodehour cluster which contains at least one alert nodehour. The colors in each line represents the distribution of alert nodehours (red) to normal nodehours (blue). These graphs indicate that our clustering method was able to separate normal nodehours from alert nodehours to a good degree. Note that there are other clusters not shown in the graphs in Fig 2, which consist entirely of normal nodehours.

The conceptual purity ratio attempts to measure the degree to which the clusters are able to differentiate between different alert categories, i.e. if one of the nodehours in a cluster contains the signature for a specific alert category, to what degree do the other nodehours also show the signature for the alert category. The results show an average conceptual purity ratio of $\sim 0.96$ with regard to the alert categories. Details of this evaluation for each of the datasets can be found in [12]. These results overall help to validate the hypotheses stated in Sec. III-A.

VI. DISCUSSION

In this section, we discuss the implications of our results and our ideas about how this method can be generalized.

We outlined in Section III-A the constraints on the distribution of message types across nodehours, which allow our method to work. These constraints have to be met for the proposed method to work. These datasets are from four different HPC systems that are different in configuration, installation location and usage. Despite this, the constraints applied to them all. This gives a strong support that these constraints are generalizable to HPC systems and possibly be to other systems that are similar such as Data-centers.

The first constraint can be achieved by the utilization of sufficiently large datasets i.e. data sets that result in sufficient bins of the system log. The second constraint on the other hand, likely results due to the Pareto sampling of messages and the choice of the time window for the partitions used. Since the Pareto property is a characteristic of the log data, it is generalizable to most system logs. Therefore, meeting the second constraint would only require careful selection of an appropriate time window for the partitions. A time window of one hour (from a nodehour) was utilized in our evaluations and was able to generalize for the 14 datasets used. Previous work states that correlated messages in system logs could occur within time windows of between 1 second to 1 day [15]. A more generalized approach would be to vary this time window with the inter-arrival rate of messages in the
system log. However, once patterns have been extracted from the system log, they can be applied without recourse to the time window that is used to extract them.

For the proposed method to work, it is necessary for the events in a system log to be separable by source and time. This is required for the information content calculations. This method cannot be applied to a system log where source and time information is not available.

On the other hand, this method can be generalized to a system log, which contained only messages for a single node or computer. The proposed technique can fit to this scenario because it is possible to leverage other system log fields to act as sources for the events in the log. The proposed approach ignores the order in which messages occur in a nodehour in determining patterns. It could be interesting to investigate how taking this into consideration could affect the results. An obstacle to this investigation is the fact that clocks among several computers are usually not well synchronized to the level of precision that is required for such analysis [5], [16].

VII. CONCLUSIONS AND FUTURE WORK

In this work, we demonstrate how the decomposition of system log messages using source and time information naturally leads to grouping of correlated message types within the partitions. We also demonstrate how these correlated message types can be discovered using the entropy-based information content of the partitions (nodehours) as a means of clustering the partitions. We evaluate our method using 14 datasets derived from the functional decomposition of system logs of four HPC systems. The results show that the resulting clusters are well formed i.e. having high internal cohesion and high external separation. We also show that with a high level of confidence we could conceptually map the clusters to different alert categories. While our results are only tested on alert behavior, the proposed approach has potential to be extended to normal behavior.

The well formed nature of the clusters produced by the proposed method lends credence to the assertion that these clusters potentially mirror the different states of the system. The metrics we have used to evaluate our results, using ground truth on alert states and internal measures, provide sufficient evidence that these clusters are useable for describing the state of a system at a point in time. These states can be used to learn a Finite State Automaton (FSA), which mirrors the normal workflow of the system [7]. Hence our results have practical applications for enhancing the self-awareness and self-monitoring capabilities of a system, which are important characteristics for an autonomic system [4].

Future work will focus on testing our approach on non-HPC system logs and on determining an appropriate method for choosing a time window for the partitions used in the approach. Further analysis of the clusters produced by our method would also be interesting. This would be a bid to further validate them as indicators of system state and the building of models of state transition. Since our method demonstrates potential to separate alert states from normal states, it will be interesting to see how information from these clusters can be used to improve the task of alert detection.

ACKNOWLEDGEMENTS

This research is supported by a Natural Science and Engineering Research Council of Canada (NSERC) Strategic Project Grant. This work is conducted as part of the Dalhousie NIMS Lab at http://www.cs.dal.ca/projectx/.

REFERENCES