Schema Independent Reduction of Streaming Log Data

Theodoros Kalamatianos and Kostas Kontogiannis

National Technical University of Athens
Dept. of Electrical and Computer Engineering
Athens, 15780, Greece
thkala@softlab.ntua.gr, kkontog@softlab.ntua.gr

Abstract. Large software systems comprise of different and tightly interconnected components. Such systems utilize heterogeneous monitoring infrastructures which produce log data at high rates from various sources and in diverse formats. The sheer volume of this data makes almost impossible the real- or near real-time processing of these system logs. In this paper, we present a log schema independent approach that allows for the real time reduction of logged data based on a set of filtering criteria. The approach utilizes a similarity measure between features of the incoming events and a set of filtering features we refer to as beacons. The similarity measure is based on information theory principles and uses caching techniques so that infinite log data streams and log data schema alterations can be handled. The approach has been applied successfully on the KDD-99 intrusion detection benchmark data set.

Keywords: software engineering · log analysis · log filtering · information theory

1 Introduction

Large software systems consist of many interconnected components. The operation of such systems is usually monitored by specialized applications that emit a wealth of information in the form of event logs. In this context, a challenging task is to understand in a tractable manner what operations are performed by the system at any given time, in order not only to understand how the system operates, but also to identify situations where the system is performing unscheduled or unexpected tasks. To date, most dynamic analysis approaches are applied offline, but for most practical applications a real-time on-line analysis is preferred. However, the sheer volume of the emitted logged data makes such an on-line analysis non tractable. The objective is thus to devise log filtering techniques that allow for the selective reduction of logged data according to specific filtering criteria. The filtering criteria can be set by the administrators and relate to specific hypotheses or analyses that need to be tested or performed.

In this paper we propose an on-line, schema independent approach that is based on information theory principles to calculate log event similarity with

the purpose of analyzing and appropriately grouping streaming events that are emitted by a system's monitoring infrastructure. The technique consists of three main steps. In the first step, which can be performed off-line, a collection of significant log features or "beacons" are selected by the user or by an automated process (e.g. a data mining process), as being significant to a use case or to a possible type of system incident. Such beacon features may be unsuccessful login attempts to a particular server, a warning for a failed transaction, transaction requests arriving at a high frequency from a group of servers, or a performance degradation alert. Beacon features are then used to generate user defined *beacon events*. In the second step of the process, event similarity values are computed between the beacon events and the incoming events in the log data set. The similarity scores are computed by comparing attribute values that are weighted by an information content coefficient. For example, if an attribute value is constant across all logged events, then this attribute value should not be considered as important for the computation of an overall similarity between two events. The result of the second step is an overall similarity value for each event with the beacon event set as a whole. In the third step of the process, a threshold is selected and events that exhibit an aggregated similarity value with the beacon set that is above the threshold value, are considered as a cohesive collection of events that correspond to a use case or an incident.

The proposed approach has three notable advantages over existing dynamic analysis approaches. First, it is schema independent and it is, therefore, readily available for collections of logged data that are emitted by different monitoring components and not conforming to the same schema or format. This eliminates the need for event schema merging or mapping, a process that is often too computationally expensive to be performed for on-line analysis. Second, the proposed approach does not require a training data set, a requirement for most approaches that are based on machine-learning techniques. Rather, the proposed approach is based on the on-line adaptive re-calculation of an information content coefficient for each attribute value, allowing thus the similarity score to be automatically adapted as the system evolves or its operational profile changes. Finally, it can be applied to streaming log data, permitting real time analysis, in contrast to most log analysis techniques that analyze stored logged data in an off-line manner.

This paper is organized as follows: Section 2 presents related work. Section 3 presents the event model. Section 4 discusses the event similarity measure and the selection of the filtered output. Section 5 presents implementation considerations, Section 6 presents experimental results, while Sections 7 and 8 provide related discussion and conclude the paper.

2 Related Work

Dynamic program analysis has been extensively used to understand the behavior of software systems. Bruegge et al. [1] proposed a framework to support dynamic analysis by source code instrumentation of systems written in C/C++. Männistö et al. [2] proposed the tool SCED, for modeling dynamic properties of object

oriented systems. Both tools require access to the source code, which might not always be the case.

In the area of data set filtering, in [3] two different data filtering and noise reduction techniques are discussed. The first is based on multiple-partitioning filtering while the second is based on iterative partitioning filtering. In [4] a technique that allows for the discovery of processes by analyzing event logs is proposed, while in [5] a log analysis technique has been used to evaluate the evolution of business models by comparing known model templates to actual models. In [6] a domain independent approach is proposed for log reduction. The main difference between the proposed approach and the one presented in [6] is that the approach discussed here uses information theory for event similarity calculation and that it can be applied in streaming log data.

In the field of log analysis by event feature reduction, [7] discusses a technique using Latent Semantic Indexing for log reduction and filtering so that root cause analysis can be tractable for systems emitting large volumes of log data. In [8] a technique to identify event features important for dynamic analysis and, in particular, intrusion detection is presented. The reduced feature sets allow for more tractable analysis to be performed. Compared to our approach, the work in [8] is fine tuned for intrusion detection and aims to reduce features as opposed to events. In [9] the Enhanced Support Vector Decision Function technique is used for selecting important features in log entries to support intrusion detection analysis. In [10] a technique based on the maximization of conditional information and weak dependence is proposed for the selection of important features in data collections. In [11] an information theoretic approach that selects an optimal set of attributes by removing irrelevant and redundant features is presented. Similarly, in [12] a hybrid approach for feature selection based on information theory as well as filter and wrapper models is presented. In [13] a two phase approach for network intrusion detection that is based on feature reduction and reasoning using fuzzy clustering and the Dempster-Shafer theory is proposed.

Overall, the main differences of our work with the related work presented in this section are that our approach is domain and schema independent and it does not require training data sets. Furthermore, it can be applied on real-time streaming data, thus allowing for on-line analysis.

3 Event modelling

3.1 Static event model

Each event e_i of the input stream $\mathcal{E} = \{e_1, e_2, \ldots e_{N_E}\}$ is defined as a set of pairs that consist of an attribute $a_j \in \mathcal{A}$ and its associated value $v_{j,i}$. More specifically, each event e_i in the input stream is defined as:

$$e_i = \{< a_j, v_{j,i} >: 1 \leq i \leq |\mathcal{E}|, 1 \leq j \leq |\mathcal{A}|\} \quad (1)$$

where $\mathcal{A} = \{a_1, a_2, \ldots a_{N_A}\}$ is the set of all attributes a_j appearing in the events of the input stream. If an event e_i does not have an attribute a_j we simply consider the value for this particular attribute to be NULL. This notation can

be used to represent any acyclic tree-like structure, such as those commonly generated by system utilities (e.g. XML, JSON), while it can be easily stored in a database, such as MongoDB.

It should be noted that this approach is particularly effective when used with *structured* data, with its precision increasing as the detail of the representation increases. While free-form attribute string values can still be used as input to the low-level primitive value distance metric calculation, several approaches have been proposed (e.g [14], [15]) to extract schema information from log files.

3.2 Stream model

The implicit insertion of NULL values allows each attribute a_j to be defined as a stream S_{a_j} of its values $v_{j,i}$: $S_{a_j} = \{v_{j,1}, v_{j,2}, \dots v_{j,N_E}\}$. Therefore the event stream can be viewed as a collection of discrete time series, one for each attribute, that are evolving in parallel.

The proposed approach is most effective when each attribute is an independent variable with no correlation to other attributes. Redundant information in the input stream may skew the results of the process by incorrectly emphasizing certain values, while reducing the perceived importance of others. The problem of normalizing redundant information streams has been studied extensively. For example, several feature selection techniques have been proposed (e.g. [10], [8]) and can be used as a pre-processing stage to remove redundant features.

4 Event analysis and filtering

4.1 Beacon event set compilation

The first phase of the proposed process involves the compilation of a cohesive beacon event set $\mathcal{B} = \{b_1, b_2, \dots b_{N_B}\}$ that serves as the filtering criterion. These events can be selected directly from the input stream or can be drafted by the operator, as a collection of *pseudo-events*, by combining features and values that the operator considers important or of interest. The system requires no information on the actual selection process, which allows the use of opaque third-party utilities.

4.2 Similarity computation

In order to compute a similarity measure of each event in the input stream with the beacon set, we propose a three stage process. The results of each stage are fed to the next one, creating a hierarchical process that is described bellow.

Stage 1 The first stage involves the dynamic determination of the importance that an attribute or a value carries in the evaluation of a final overall similarity measure. For example, an attribute for which its value is constant throughout

the input stream does not carry any significant *information content* for the computation, while attribute values that vary may carry higher *information content*. This allows a low-level similarity metric for primitive values (e.g. strings), typically described as $dist : \mathcal{E} \times \mathcal{E} \times \mathcal{A} \rightarrow \mathcal{R}$, to be used as the basis of an event-level similarity measure.

More specifically, the proposed approach attempts to determine which attributes and attribute values offer the best event *selectivity* with regard to a specific beacon set. For this purpose, a statistical similarity measure based on information theory is introduced. Each attribute a_j is considered an independent discrete random variable with an alphabet V_j of N_{V_j} possible symbols (values).

According to information theory, the entropy H of an independent variable X is a measure of the average information content of each sample of X. Likewise, the information content I is a metric of the *importance* of a variable as a whole. If X is a discrete time series of N samples with an alphabet of n symbols and a probability mass function of $p(X)$, we have:

$$H(X) = E(-log_r p(X)) = -\sum_{i=1}^{n} p(x_i) log_r p(x_i) \tag{2}$$

$$I(X) = -\sum_{i=1}^{N} log_r p(x_i) = N \cdot H \tag{3}$$

The probability $p(x_i)$ of a symbol is equal to its relative frequency within the time series. Specializing formula 3 using the individual attribute value frequencies, it is therefore possible to compute the information content of a specific attribute value $v_{j,i}$, as well as that of a discrete attribute a_j as a whole:

$$I(v_{j,i}) = -n_{j,i} \cdot log_r \frac{n_{j,i}}{N_E} \tag{4}$$

$$I(a_j) = -\sum_{i=1}^{N_{V_j}} (n_{j,i} \cdot log_r \frac{n_{j,i}}{N_E}) \tag{5}$$

The selectivity offered by each event attribute is in direct relation to the information content of the equivalent time series. Conceptually, highly repetitive attributes, such as domain-specific constants, have a limited use as *distinguishing features* between events, but they also exhibit a relatively low entropy that can be used to reduce their participation in the event comparison process. Furthermore, attributes with extreme diversity, e.g. unique identifiers, tend to skew the similarity metric, since they have a high information content despite being of limited value for determining similarity. To offset this issue, the information content $I(v_{j,i})$ contributed by each specific *attribute value* is taken into account in relation to the information content $I(a_j)$ of that particular attribute as a whole. This leads to the following definition of the *information content fractions* $IF(a_j)$ and $IF(v_{j,i})$ for an attribute and an attribute value respectively:

$$IF(a_j) = \frac{I(a_j)}{\sum_j^{N_A} I(a_j)} \tag{6}$$

$$IF(v_{j,i}) = \frac{I(v_{j,i})}{I(a_j)} \tag{7}$$

The information content fraction is a dimensionless quantity in the $[0, 1]$ range. Using it as a coefficient lessens the impact of attribute values with low overall contribution. This is especially effective for attributes with high diversity: a large number of discrete values means that each specific value has a minuscule contribution to the overall information content on its own. The resulting IF fraction will be relatively low, reducing the skew normally caused by high-diversity attributes. Using this principle, a similarity metric $SV_{i,b,j}$ can be defined for the attribute a_j values of input event e_i and beacon event b as follows:

$$SV_{i,b,j} = IF(v_{j,b}) \cdot IF(v_{j,i}) \cdot dist(i, b, j) \tag{8}$$

Stage 2 At the second stage we compute a weight W_{a_j} for each attribute a_j in the input stream $\mathcal{E}$. This attribute-specific weight is affected by the information content of the attribute and the particular contribution of its specific values in the beacon event set. This enhances the effect of less frequent values that are common within the beacon event set and may, therefore, be a distinguishing feature for the selection process.

$$W_{a_j} = I(a_j) \cdot \prod_{b \in \mathcal{B}} IF(v_{j,b}) \tag{9}$$

Stage 3 The final stage contains the evaluation of a similarity measure between two events and leverages that measure to compare each input stream event with the beacon set as a whole.

Lin [16] offers a formal definition of the concept of similarity, based on three basic intuitive tenets: (a) the more commonality two objects share, the more similar they are; (b) the more differences two objects have, the less similar they are and; (c) two identical objects should always reach the maximum similarity.

Using a weighted mean to leverage the per-attribute similarity values from the previous stages to an event-level metric $SE_{i,b}$ satisfies all three basic conditions. Additionally, it allows the attribute-level weights which are not bounded to form a bounded metric with the same range as the primitive correlation metrics:

$$SE_{i,b} = \frac{\sum_{j=1}^{N_A} (W_{a_j} \cdot SV_{i,b,j})}{\sum_{j=1}^{N_A} W_{a_j}} \tag{10}$$

The final computation involves the determination of an overall similarity SB_i of an input event with the beacon set as a whole. To procure a similarity metric between a single event and an *event set* using a metric defined between single events, it becomes necessary to reexamine the three basic principles mentioned above. More specifically, the requirement for a maximum similarity result on identical inputs is no longer satisfiable since sets and single items are not directly comparable. Our prototype implementation uses an arithmetic mean, averaging the similarities calculated for the input event with each beacon event:

$$SB_i = E(SE_{i,b}) = \frac{\sum_{b \in \mathcal{B}} SE_{i,b}}{N_B} \tag{11}$$

4.3 Filtered event group selection

Conceptually, for the reduced event set we aim to select those events with the highest similarity values. Providing a *threshold* for that selection, however, is not trivial. Real-time streams are unbounded with regard to space and impose certain latency constraints, while the amount of information that is known a priori is limited. Therefore an *adaptive* threshold selection method should be used for the last phase of the process.

In our approach, the resulting event group is computed using a dynamic threshold, which is determined by detecting significant gaps in the distribution of similarity values. More specifically, a *sketch* of the cumulative distribution function (CDF) of the similarity values is formed in constant space and time, and is updated and examined periodically as the input stream is processed. Using a rough derivative computation it is possible to detect plateaus in the CDF, which typically separate similarity value clusters. A heuristic algorithm is then used to select an appropriate threshold, by taking into account the distances between the highest valued clusters, their size and their position within the similarity value range. Similar techniques based on sub-linear data sketches have been used for approximate percentile determination on real time streams ([17], [18]).

5 Design and implementation considerations

5.1 Adaptivity of the proposed approach

Adaptivity to schema changes The prototype implementation is completely domain agnostic, with no *a priory* knowledge of the domain or the event schema. As such, it operates on the inherent assumption that only part of the actual schema has been considered at any given time. An incoming event may contain a number of previously unseen attributes that expand the existing view of the schema of the monitored domain. Each new attribute is essentially treated as a time series that only contained *null* values until its time of emergence.

However, for most event domains and monitoring systems it is reasonable to assume a *fixed number of attributes*. Therefore, the set of attributes that are known to the system at runtime will gradually converge towards a constant set, with few or no new additions after a sufficient amount of time.

Information content aging Typical statistical algorithms are generally targeted at data *sets* and not well suited for streaming input:

1. Common set-based metrics exhibit a form of *inertia* as the number of recorded data points increases. New data has a decreasing effect as time passes, gradually reducing the adaptability of the system to zero.

2. Cumulative metrics, e.g. the mean and standard deviation, suffer from numerical range and precision issues when implemented trivially on a computer.

To avoid these issues, the proposed system makes use of algorithms that *devalue* older data points with the passage of time. Daneshgaran et al. [19] provide an information theoretic definition of aging, while Cormode et al. ([20]) describe a generic exponential decay model for aggregate metrics.

Our prototype implementation synchronizes using input events as a *time reference* and ignores any other perception of time. As a result, the decay model used for the aging process, is a simple step-wise decay model. For each *cycle*, the information content currently recorded for all attributes and attribute values is multiplied by a positive *decay* coefficient no greater than 1:

$$I_{aged,t+1} = decay \cdot I_t + (I_{t+1} - I_t) \tag{12}$$

The decay coefficient should be selected with regard to the monitored system, most notably its event rate and other temporal characteristics. Typical values for the *decay* coefficient during experimentation were in the $[0.95, 1)$ range.

5.2 Space complexity and object replacement algorithms

Since real-time systems have no known limit for the length of the input stream, their space complexity should be constant, or at most sub-linear, with respect to the size of the input. For this reason, it is necessary to *approximate* the operation of the similarity determination algorithm in a space-efficient manner. From the mathematical formulas at its foundation it is clear that the values with the least impact are those with the lowest contribution of information content. Therefore, a potential approximation involves limiting the volume of retained metadata by eliminating the entries that correspond to low-impact values.

Taking the effects of the aging process into account, one can intuitively identify the low-impact values as those that (a) are infrequent or (b) have not appeared recently in the input stream. Selecting items based on frequency and recency is essentially the purpose of *object replacement algorithms*, more commonly known as *caching algorithms*. The main intent of a caching subsystem is to use a pre-determined amount of space to store results of past requests, replacing old or infrequent entries when necessary. Therefore, we can use such replacement algorithms to provide a hard limit for the space usage of the proposed system.

For our prototype implementation, we selected the *Adaptive Replacement Cache* [21] algorithm, due to its simplicity and overall performance. It provides a good balance between recency and frequency, while also being resistant to pathological behaviors and able to adapt to its input. Moreover, it has excellent runtime performance and does not need external tuning that would require domain-specific knowledge.

5.3 Runtime performance

In order to ensure the scalability of the system for monitoring large infrastructures, several approaches to parallelization are available:

– *Attribute-level parallelization*, where each processor handles a subset of the attributes within each event, based on the assumption that each attribute is a completely independent time series.
– *Beacon-level parallelization*, where the comparison of an input event with each beacon event is handled by a separate processor.
– *Event level parallelization*, where the input events are distributed to a large number of processing nodes using an appropriate load-balancing scheduler.

These methods are generally orthogonal and can therefore be employed simultaneously if necessary. The prototype implementation, supports the first two methods and is able to achieve a sustainable throughput of several thousand events per second, when making full use of a 4-processor personal computer.

6 System evaluation

For the evaluation of the proposed algorithm we used a data set generated within the scope of a competition that was held in conjunction with KDD99, the Fifth International Conference on Knowledge Discovery and Data Mining. The KDD99 data set [22] is a derivative of the DARPA Intrusion Detection Evaluation 1999 data set. As such, KDD99 events represent traffic within a computer network, while specific security breaches are attempted. The data set features about 4.8 million events and 41 distinct attributes, of both continuous and discrete types. In addition, the KDD99 data set contains embedded classification data for each event, which can serve as a *reference* for the evaluation of the quality metrics of any selection method. This alleviates the need for domain-specific tools or manual intervention for the creation of the *golden standard* against which our system will be evaluated.

The prototype implementation was written in the Java programming language and tested extensively in streaming mode. For each experiment, the used beacon events were selected randomly from a specific attack type, with the rest of the events that belong to the same attack type being the expected result.

6.1 Similarity metric evaluation

To evaluate the quality of the similarity metric, it is necessary to examine its selectivity between matching and non-matching events. Intuitively, an acceptable metric would provide results with clear *separation* between potential matches and the rest of the input, allowing the selection of the matching results by means of a simple similarity threshold. Ideally, the separating space would also not contain any noise in the form of infrequent errant values, although their effect on the overall accuracy of the system would be negligible and they could be filtered-out relatively easily using a variety of methods.

Fig.1 illustrates the aforementioned similarity gap for the events from a small range of the input stream. The plotted values were produced using a beacon set of 20 events that match the *neptune* attack type from the KDD99 data set. The

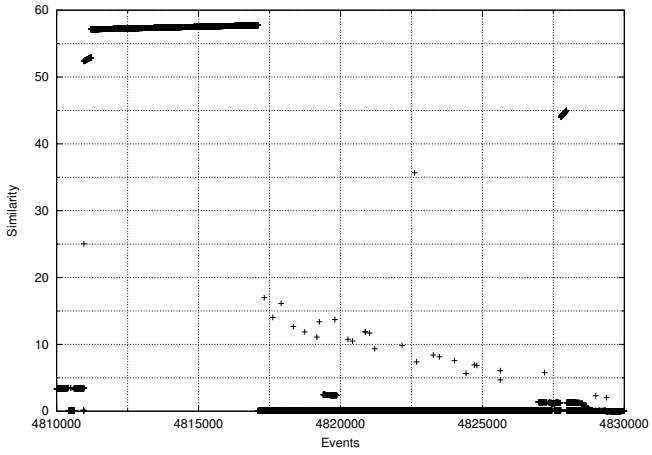

Fig. 1. Similarity values for a part of the input stream with two clusters of potential matches

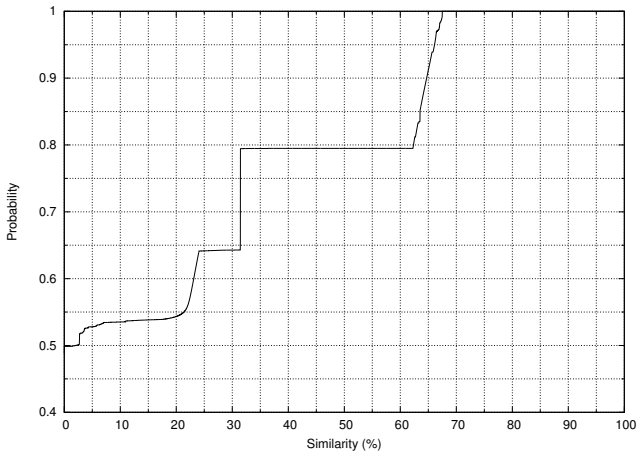

Fig. 2. Cumulative distribution function for the produced similarity values

input set contains two event clusters of the same type in this particular range, in the subranges $[4810961 - 4817099]$ and $[4827760 - 4827959]$. The corresponding similarity values rise from a baseline of about $5 - 10\%$ to values in the $40 - 60\%$ range, with virtually no noise in the range from approximately 20% to 40%, thus being in complete accordance with the golden standard.

Fig.2 depicts a plot of the cumulative distribution function (CDF) of the similarity for the same scenario, calculated over a range of $1,000,000$ events. The CDF of the similarity essentially provides the ratio of rejected events for each possible selection threshold. Sharp rises in the CDF indicate the existence of

Table 1. Experiment results - Beacon set size effect

Test	# of beacons	# of correct events	# of retrieved events	Precision (%)	Recall (%)	Reduction (%)
neptune	5	203941	203941	100.000	99.573	79.606
neptune	10	203947	203941	99.997	99.573	79.605
neptune	20	205087	204023	99.481	99.613	79.491
back	5	1999	1941	97.099	96.953	99.800
back	10	2070	1941	93.768	96.953	99.793
back	20	2045	1941	94.914	96.953	99.796
teardrop	5	993	198	19.940	99.497	99.900
teardrop	10	196	196	100.000	98.492	99.980
teardrop	20	203	196	96.552	98.492	99.980

a tight cluster around the same value, while a plateau is caused by the lack of any data points in the corresponding range. In Fig.2 a gap in the similarity values, indicated by a plateau in the CDF plot for similarities in the approximate range of $31 - 63\%$, separates the matching events from the rest of the input stream. A rather conservative threshold selection of e.g. 35% would preserve roughly 20% of the input stream, which is congruent with the ratio of *neptune*-type events in the same range.

Both figures illustrate the potential usability of the proposed similarity metric as a *distinguishing criterion* for event selection purposes. In addition, the significant similarity value difference between matching and non-matching events allows the use of simpler algorithms in the final selection stage, by limiting the need for complex noise-reduction techniques.

6.2 Quality of results

An accepted technique of assessing an information retrieval process is the measurement of *precision* and *recall* values. For our system, recall is more important than precision, since recall is critical for ensuring that the technique does not discard matching events, especially in a streaming environment where the recovery of such events might not be possible. However, precision still remains an important quality, since it relates to the reduction effected upon the size of the input, which is the final purpose of the proposed system.

For the evaluation process, the prototype implementation was subjected to a series of experiments, a subset of which is presented in table 1. Table 1 contains results from experiments performed over a range of $1,000,000$ events for three different event types and for a variable beacon event set size. The selected scenarios cover a significant event frequency range, with *neptune*-type events comprising about 20% of the input, while *back* and *teardrop* events correspond to 0.2% and 0.02% respectively. Despite this variation, the system reliably manages to retrieve over 95% of the requested events, with a precision that typically lies in the range of $90 - 95\%$. The system is able to produce acceptable results

Table 2. Experimental results - Effect of noise in the beacon set

Test	Beacon noise (%)	# of retrieved events	# of correct events	Precision (%)	Recall (%)	Reduction (%)
neptune	10	204027	204021	99.997	99.612	79.597
neptune	20	204810	204021	99.615	99.612	79.519
neptune	30	204027	204021	99.997	99.612	79.597
neptune	40	205260	204798	99.775	99.992	79.474
neptune	50	206266	204733	99.257	99.960	79.373
back	10	2070	1941	93.768	96.953	99.793
back	20	5433	1941	35.726	96.953	99.457
back	30	44540	1929	4.331	96.354	95.546
back	40	77757	1904	2.449	95.105	92.224
back	50	497826	1985	0.399	99.151	50.217
teardrop	10	389445	199	0.051	100.000	61.056
teardrop	20	497395	199	0.040	100.000	50.261
teardrop	30	497395	199	0.040	100.000	50.261
teardrop	40	497395	199	0.040	100.000	50.261
teardrop	50	497395	199	0.040	100.000	50.261

with as low as 5 beacon events, which makes it usable by human operators without the aid of additional tools. Varying the size of the beacon event set, does not generally appear to have a consistently significant effect, although in most cases we observed that larger sets resulted in a slightly higher amount of noise. The *teardrop* event type, however, is a notable exception, due to the relatively limited cohesion and higher diversity that characterizes its members. As a result, a larger beacon event set is required to allow the system to reliably establish which features characterize the *teardrop* events, as evidenced by the low precision calculated for the 5-event beacon set. In addition the small size of the *teardrop* event set aggravates the effects of any factor that negatively affects the precision of the system.

6.3 Stability assessment

To assess the stability of the system we considered its behavior with respect to the presence of random noise in the beacon event set. The evaluation is particularly important, since in real-life cases it is not always possible to define a beacon event set with absolute precision. Table 2 illustrates the experimental results of randomly selected noisy events in the beacon set. Low amounts (e.g. 10%) of noise do not generally have a significant impact, while higher amounts may have a negative effect on precision with the recall being mostly unaffected. On the other hand, event types with limited cohesion, such as the *teardrop* type, are far more susceptible to the negative effects of noise, since the beacon events feature a certain amount of noise themselves. It should be noted that these comments are only valid in the case of *random* noise. The effects of cohesive erroneous

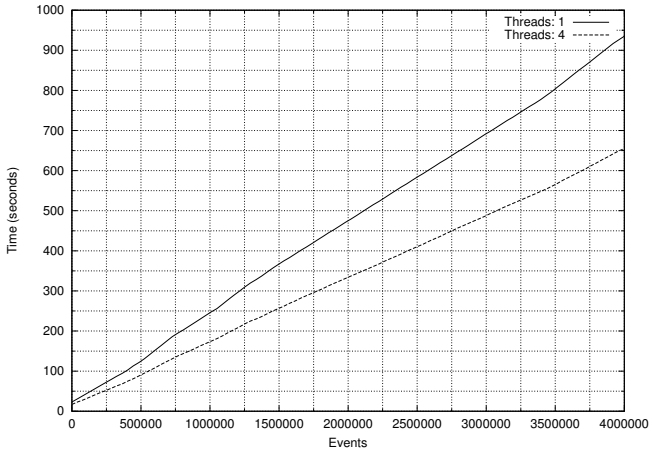

Fig. 3. Processing time versus input size for a variable number of worker threads

events in the beacon set are more severe, with the system generally selecting events that match either the requested or the erroneous event set.

6.4 Runtime performance

The runtime performance of the prototype implementation was evaluated from several aspects, in order to assess the feasibility of its deployment in production environments. On a mid-range personal computer with four processor cores, the throughput of the system was macroscopically stable and typically well in excess of $5,000$ events/second, with $10,000$ events/second being attainable for smaller beacon event sets. The use of multiple worker threads provided a notable performance increase, as seen in Fig.3, although the prototype had not been adequately optimized for multiple processors. Last, the system had quite reasonable memory requirements, being able to process the KDD99 data set with less than 96 MB of heap memory available to the Java VM.

7 Discussion

The proposed approach is mainly intended as an efficient schema-independent initial-approach analysis tool. As such, several assumptions were made which can potentially create situations where the system will misbehave:

Each attribute is examined in isolation: The potential usefulness of *combinations* of attribute values is not examined. Addressing this issue in a domain-agnostic manner requires the use of adaptive methods to avoid the introduction of operations with exponential complexity in relation to the combination size.

The system examines each event independently: No temporal relationships are established between events or attribute values. In general the detection of

systematic transitions is considered to be beyond the scope of this system, for complexity and performance reasons.

The information content determination algorithm does not take value proximity into account: In some cases it makes sense to consider as equivalent attribute values which happen to be in close numerical or lexicographical proximity.

However, the event selection approach presented in this paper is more flexible than rule-based filters, as it does not require any domain knowledge and can adapt to the input stream.

8 Conclusion

In order for on-line tractable analysis of log data streams to be performed, we should first devise techniques that allow for the selective reduction of the volume of the data that needs to be considered for each analysis. In this paper, we have presented an approach that allows for the on-line selection of logged events that are highly cohesive with respect to a particular feature set. The feature set is used to compile a set of user defined events we refer to as beacon event set. The proposed approach is based on information theory to compute a similarity measure between incoming events and the beacon set, and is schema independent. The result is a highly reduced collection of incoming events that are highly related to specific system behavior. Results obtained from the KDD99 benchmark data set indicate that the approach can tractably reduce the size of events that need be considered with respect to some important system activity such as intrusion attempts, while maintaining high recall and precision levels. Possible future work includes the extension of the approach to handle combinations of attributes for computing event similarity, the incorporation of an attribute reduction pre-processing phase so that further performance enhancements can be possible, and the investigation of techniques for adaptive threshold selection. This work has been supported by CA Labs and CA Technologies UK.

References

1. B. Bruegge, T. Gottschalk, and B. Luo, "A framework for dynamic program analyzers," in *SIGPLAN Notices*, vol. 28, no. 10. ACM, 1993, pp. 65–82.
2. K. Koskimies, T. Männistö, T. Systä, and J. Tuomi, "SCED: a tool for dynamic modelling of object systems," *University of Tampere, Dept. of Computer Science, Report A-1996*, vol. 4, p. 199, 1996.
3. T. M. Khoshgoftaar and P. Rebours, "Improving software quality prediction by noise filtering techniques," *Journal of Computer Science and Technology*, vol. 22, no. 3, pp. 387–396, 2007.
4. S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens, "Process discovery in event logs: An application in the telecom industry," *Applied Soft Computing*, vol. 11, no. 2, pp. 1697–1710, 2011.
5. R. Agrawal, D. Gunopulos, and F. Leymann, "Mining process models from workflow logs," *Advances in Database TechnologyEDBT'98*, pp. 467–483, 1998.

6. T. Kalamatianos, K. Kontogiannis, and P. Matthews, "Domain independent event analysis for log data reduction," in *IEEE Conference on Computers, Software and Applications*. IEEE, 2012, pp. 225–232.
7. H. Zawawy, K. Kontogiannis, and J. Mylopoulos, "Log filtering and interpretation for root cause analysis," in *Software Maintenance, IEEE International Conference on*, 2010, pp. 1–5.
8. G. Zargar and P. Kabiri, "Identification of effective network features for probing attack detection," in *Networked Digital Technologies, First International Conference on*, 2009, pp. 392–397.
9. S. Zaman and F. Karray, "Features selection for intrusion detection systems based on support vector machines," in *6th Consumer Communications and Networking Conference*. IEEE, 2009, pp. 1–8.
10. F. Fleuret, "Fast binary feature selection with conditional mutual information," *The Journal of Machine Learning Research*, vol. 5, pp. 1531–1555, 2004.
11. M. Last, A. Kandel, and O. Maimon, "Information-theoretic algorithm for feature selection," *Pattern Recognition Letters*, vol. 22, p. 799, 2001.
12. M. Sebban and R. Nock, "A hybrid filter/wrapper approach of feature selection using information theory," *Pattern Recognition*, vol. 35, no. 4, pp. 835–846, 2002.
13. T. S. Chou, K. K. Yen, and J. Luo, "Network intrusion detection design using feature selection of soft computing paradigms," *International Journal of computational intelligence*, vol. 4, no. 3, pp. 196–208, 2008.
14. W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 117–132.
15. M. Nagappan and M. A. Vouk, "Abstracting log lines to log event types for mining software system logs," in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. IEEE, 2010, pp. 114–117.
16. D. Lin, "An information-theoretic definition of similarity," in *15th international conference on Machine Learning*, vol. 1, 1998, p. 296.
17. Q. Zhang and W. Wang, "A fast algorithm for approximate quantiles in high speed data streams," in *19th International Conference on Scientific and Statistical Database Management*. IEEE, 2007, p. 29.
18. G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Space-and time-efficient deterministic algorithms for biased quantiles over data streams," in *Proceedings of the 25th ACM SIGMOD*, 2006, pp. 263–272.
19. F. Daneshgaran and M. Mondin, "Information aging," in *Information Theory. Proceedings., IEEE International Symposium on*, 1997, p. 38.
20. G. Cormode, S. Tirthapura, and B. Xu, "Time-decayed correlated aggregates over data streams," *Statistical Analysis and Data Mining*, vol. 2, no. 5-6, pp. 294–310, 2009.
21. N. Megiddo and D. Modha, "ARC: a self-tuning, low overhead replacement cache," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, 2003, pp. 115–130.
22. "The Knowledge Discovery & Data Mining Cup 1999 Data." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html