Pattern Detection in Unstructured Data

An Experience for a Virtualized IT Infrastructure

Mazda A. Marvasti, Arnak V. Poghosyan, Ashot N. Harutyunyan, and Naira M. Grigoryan
VMware
{mazda;apoghosyan;aharutyunyan;ngrigoryan}@vmware.com

***Abstract*—Data-agnostic management of today's virtualized and cloud IT infrastructures motivates statistical inference from unstructured or semi-structured data. We introduce a universal approach to the determination of statistically relevant patterns in unstructured data, and then showcase its application to log data of a Virtual Center (VMware's virtualization management software). The premise of this study is that the unstructured data can be converted into events, where an event is defined by time, source, and a series of attributes. Every event can have any number of attributes but all must have a time stamp and optionally a source of origination (be it a server, a location, a business process, etc.) The statistical relevance of the data can then be made clear via determining the joint and prior probabilities of events using a discrete probability computation. From this we construct a Directed Virtual Graph with nodes representing events and the branches representing the conditional probabilities between two events. Employing information-theoretic measures the graphs are reduced to a subset of relevant nodes and connections. Moreover, the information contained in the unstructured data set is extracted from these graphs by detecting particular patterns of interest.**

***Index Terms*—Fault management, unstructured data, pattern detection, event correlation, directed graph.**

I. Introduction

The volume of unstructured data in modern world grows rapidly. Statistical inference in those data sets is an important scientific and technological challenge. It is also an issue directly related to data-agnostic management systems, such as automated fault management in modern virtualized data centers and in the cloud. A data-driven approach here provides a universal (context-independent), fully automated, and proactive remediation framework in comparison with the expert knowledge involvement approach unfeasible for voluminous IT infrastructures with high level of transiency. What this paper demonstrates perfectly fits into the current IT management landscape in terms of problem localization, anomaly risk and impact assessment via monitoring tools.

We introduce a mathematical model converting the raw unstructured data into "event" data that enables probabilistic graph-based analysis of the environment. This analysis is purely statistical in nature without any contextual knowledge about the underlying physical phenomena. We also apply our method to a semi-structured text-based log data of Virtual Center (VC), and demonstrate detected patterns of special value. The method can be superior in practice to other methods with complex analytics applying semantic learning to extract knowledge and structuring the data for further data mining. Examples of works within that framework are [5] and [6]. In particular, Ostrowski [5] claims the goal of identifying semantic content towards classification of unstructured text. The research reported in Shariff et al [6] aims at retrieving the structured information out of unstructured data using feature extraction, analyzing it syntactically, organizing the analyzed data into entities, rules, associations, facts.

The methodology employed here relies on information measures applied on the extracted event space to create a Directed Virtual Graph (DVG) of events and important "correlations" for efficient pattern recognition purposes. This is a generic and non-rule-based technology in comparison to vast variety of traditional event correlation techniques (e.g. http://www.softpanorama.org/Admin/Event_correlation/).

Below we describe how the unstructured raw data can be converted into "events", where an event is a set of attributes one of which is used as a proximity measure. An attribute is defined as any property assigned to an event (such as criticality, user name, datacenter ID, etc.).

Introduce

$$\overline{E_i} \equiv \{m, r, A_j\}, E_i \equiv \{r, A_j\}, A_j = \{a_1, a_2, ..., a_n\}$$

where $\overline{E_i}$ is the i-th event, m is the attribute used for proximity measure (time for example), r is an optional attribute used for presentation perspective (event source for example), and A_j is the j-th attribute package containing n attributes ($n = f(j)$). Essentially this says that the number of attributes can differ between the various attribute packages and is a function of each attribute package. Furthermore, E_i is the set of events that exactly match r and p percent of attributes A_j of event $\overline{E_i}$ regardless of m. This in fact will be the nodes of the DVG discussed below.

The attributes associated with events need to be first examined to ensure they are not properties that uniquely identify an event (for example Event ID which is a unique property for every event). The reason for this identification is that we are seeking patterns within the event space and uniquely identifying attributes do not contribute any information in this identification. In fact we can generalize and ignore all attributes within an event space whose count of unique

978-3-901882-50-0

values is greater than p_a percent of the total number events. We can further impose other conditions upon the attribute set that allows for a better representation of structure within the event space. For example, we may impose rules such as 1 out of every k values for an attribute must have the same value. These rule sets (and others) allow for elimination of non-contributing attributes in pattern identification.

II. Directed Graph and Probability Space

The construction of the directed graph is based upon events and probabilities between event pairs. The nodes of the graphs represent an event state and the connections between two nodes represent the conditional probability of the two event pairs. In general the joint probability of events (E_i, E_j) can be computed by the formula

$$P(E_i, E_j \mid \Delta m) \equiv \frac{\left\|\{E_i, E_j\}\right\|}{\sum_{i=1}^{N} \|E_i\|}$$

where Δm is the maximum proximity gap (span of time for example) where events i and j are considered to be coincident, $\left\|\{E_i, E_j\}\right\|$ is the cardinality of the set $\{(E_i, E_j)\}$ considered to be coincident within Δm, and N is the total number of events (count of all E_i). The prior probability for event E_i can be computed using:

$$P(E_i) \equiv \frac{\|E_i\|}{\sum_{i=1}^{N} \|E_i\|}.$$

Applying Bayes theorem one can then obtain the conditional probability of E_i given the occurrence of E_j, namely

$$P(E_i \mid E_j, \Delta m) = \frac{P(E_i, E_j \mid \Delta m)}{P(E_j)}.$$

In essence, the above formulations determine the probability that an event will occur along with the probabilities that two specific events occur within proximity Δm. Once the events and the various probabilities are known for a system, the event graph can be constructed as shown in Figure 1. This represents the initial graph of all the events and probabilities between events (P_{ij} on the edges stand for $P(E_i, E_j \mid \Delta m)$). The next task is to reduce this graph in a way to preserve all the important aspects while reducing the total number of variables.

III. Information Measures

Probabilistic information measures play a central role in graph reduction. The mutual information [1] contained in correlation between two different events as random variables (RV) is defined by:

$$I(E_i, E_j) = \sum_{l,m} P(E_{i_l}, E_{j_m}) \log \frac{P(E_{i_l}, E_{j_m})}{P(E_{i_l}) P(E_{j_m})}$$

where indices l and m count the particular realizations of events E_i and E_j, respectively. Then the mutual information between two events constructed within our model is

$$I(E_i, E_j) = \log \frac{P(E_i, E_j)}{P(E_i) P(E_j)}.$$

In this form it is a measure of independence for those events. In case of RV events, to weigh the impact of an event E_j on a set (neighborhood) of correlated events $E_i(1), E_i(2), ..., E_i(N_i)$, a complex event denoted by $E_i^{N_i}$, the conditional entropy measure can be applied:

$$H(E_i^{N_i} \mid E_i) = -\sum_{E_i^{N_i}} P(E_i, E_i^{N_i}) \log P(E_i, E_i^{N_i} \mid E_i)$$

where $P(E_i, E_i^{N_i})$ and $P(E_i, E_i^{N_i} \mid E_i)$ are concise notations for the joint probability $P(E_i, E_i(1), E_i(2), ..., E_i(N_i))$ and conditional probability $P(E_i, E_i(1), E_i(2), ..., E_i(N_i) \mid E_i)$, respectively. In our particular case, the following combinatorial formula weighs the impact introduced by an event E_i (it is not normalized and employs the "probability flow" from E_i):

$$F(E_i) = P(E_i) \sum_{l=1}^{N_i} P(E_i(l) \mid E_i).$$

This kind of impact measure can be applied also to anomaly events data of a system for problem root cause and bottleneck identification purposes.

IV. Graph Reduction Algorithm

Obtaining the final representative DVG of the correlations in unstructured data consists of several steps:

a) Correlation graph establishment. Based on the computed probabilities, we generate the directed graph of events (Figure 1 depicts a complete graph of pair-wise correlations).

b) Graph reduction with statistically derived parameters. Assume a user defined parameter $\varepsilon \in [0,1]$ which regulates the sensitivity of graph reduction. It allows the user to introduce a control on tradeoff between the complexity and accuracy of the analysis. Compute the mutual information for each pair (i, j) and classify those values according to their signs. Let $Q^+_{0.25}$ and $Q^+_{0.75}$ be the 0.25 and 0.75-quantiles of positives, with similar notations for negative data set. The algorithm performs a graph reduction by eliminating "unessential" correlation edges, applying the whiskers model (one option), where the inter-quartile range $(Q^+_{0.75} - Q^+_{0.25})$ is an important criterion. Namely, if $I(E_i, E_j) < \Delta^+$ for $I(E_i, E_j) \geq 0$ or $I(E_i, E_j) > \Delta^-$ for $I(E_i, E_j) < 0$, then the edge connecting the node i to j is eliminated, where Δ^+ (and similarly Δ^-) is defined by $\Delta^+ = Q^+_{0.25} - (0.5 + \varepsilon)(Q^+_{0.75} - Q^+_{0.25})$. Figure 1 also illustrates dash-lined edges removable according to the above conditions that result in a loss of graph connectivity.

c) Connectivity test. To determine whether the elimination of low correlations results in loss of connectivity of the original graph (and hence in a series of sub-graphs), a knowledge interesting in terms of pattern detection, we perform a test on its adjacency matrix. A classical bit-wise OR-ing algorithm from [2] can be applied to the rows of this matrix to detect the connectivity of the graph. The sub-graphs can be identified by a *flood fill* coloring algorithm [3].

d) Conditional probability assignment. Now that the various graphs have been identified, the joint probability (first assigned to the graph edges) can be replaced by the conditional probability between nodes. The direction of the edge between two nodes is indicated using the following convention: given nodes E_i, E_j, and the conditional probability $P(E_i \mid E_j, \Delta m)$ draw a line pointing from E_j to E_i.

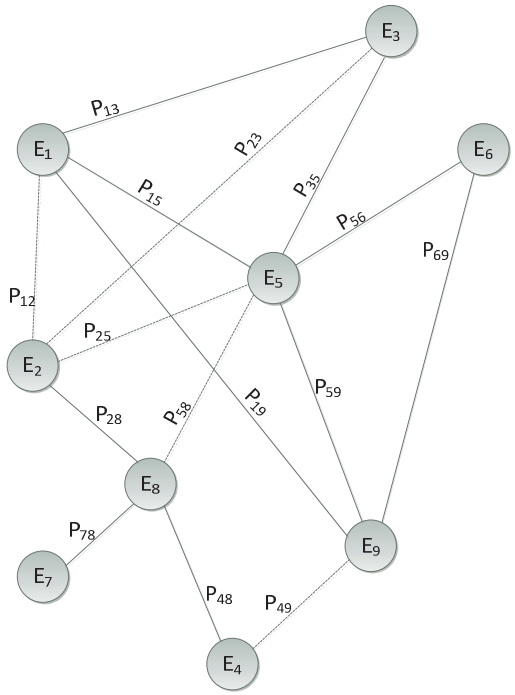

Fig. 1. Initial graph with ignorable connections.

The above steps produce a series of graphs containing the most statistically relevant information. Each graph represents a bundle of events that have some relationship to each other. Any events not belonging to any graph can be eliminated as spurious (or non-contributing) data. The problem set has thus been reduced to an examination of a series of finite graphs, each representing a correlated set of information. These graphs can also be used to extract further information about the data set. Below some specific examples of what types of information may be extracted is presented.

V. DVG-BASED ANALYSIS

Some nodes and paths in the graph can be of special interest to the knowledge miner. In particular, the following list represents one set of analysis that can be performed on the DVG's.

A. Criticality Patterns

Based on a DVG the following categories of criticality are defined and revealed:

- *Critical Node* is a node that has prior probability that is equal to or greater than the value of an upper threshold (see nodes B, G, K and P in Figure 2 for upper threshold equal to 0.9). What this reveals are events with a tendency to be more prevalent than other events.
- *Root* is a node that has no "incoming" arrows, in other words, it is historically an impacting only node (see node D in Figure 2). What this reveals are events that are sources of other events (downstream events).
- *Critical Path* is a sequence of impacting nodes with extremely high conditional probabilities on the connections (see the path BRSTU in Figure 2). What this reveals are sequences that events follow (when one occurs a sequence of events occurs).
- *Extreme Path* is a Critical Path with nodes that have prior probability that is equal to or greater than the value of an upper threshold (see the path HBAO in Figure 2 where the upper threshold is equal to 0.5). What this reveals are sequences that are highly probable in their appearance.
- *Critical sector* of some magnitude M (defined by its nodes volume) is a connected sub-graph with the joint probability connections all higher than some value.

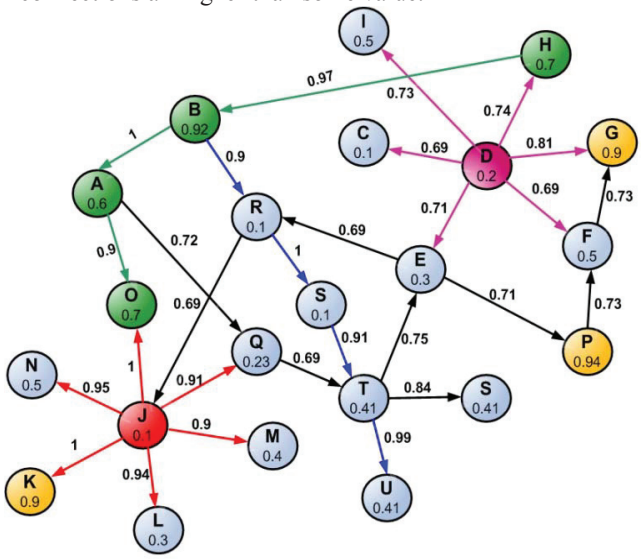

Fig. 2. Some patterns in DVG.

B. Black Swans

Black Swan is a node which has very low prior probability, but extremely high impacting ability on a large set of neighbors. The main steps in black swan type events identification are the following (see the philosophy by [4]).

Black Swan Node determination. Black Swan Node (BSN) is a rare event-node that has a small prior probability and large number of strong outgoing connections. Figures 2 and 3 show one (J) and two BSNs (J and D), respectively.

Black Swan Event determination. Black Swan Event (BSE) is a set of BSNs that covers a large set of nodes (green nodes

on Figure 3). In the same figure the two BSNs impact 72% of the system. What this reveals are events that rarely occur, but when they do they generate a lot of other events.

Based on those steps the risk of a run-time BSE can be estimated.

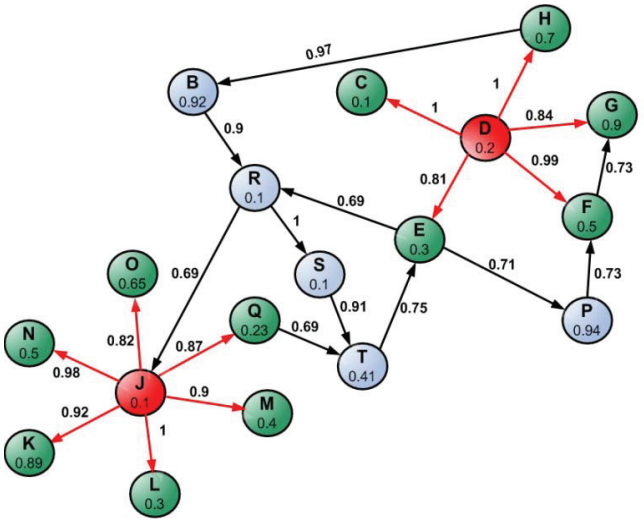

Fig. 3. Black swan type event.

VI. Implementation

The algorithm was coded in Java and enabled to consume multiple data files simultaneously and provide correlation across events. The implementation has not been optimized for scale, however, scalability numbers are included below as a point of reference.

A. Applying to VC Events

A test (without any special experimental setup) was performed on a Virtual Center (VC) diagnostic dump of 610,000 events as production data of an IT enterprise consisting of thousands of virtual devices servicing custom applications. No prior knowledge of the servers or contextual understandings of the alerts were obtained (nor necessary). The results of analysis are depicted in Figure 4. They show a DVG with three disconnected graphs for the event set. Table I shows the relevant event data where the ID column indicates the ID which is associated in the relationship graphs. The source column indicates the server ID associated with the event. Not all the attributes (abbreviated as A) are displayed to preserve space for the more relevant attributes. Also the results that are shown are not exhaustive but more representative of the overall result set. In this table the boldfaced alerts are those found with an attribute of "error" (E). The other available attributes were "warning" (W) and "info". Note that to limit the results only warning and error type events associated with another error type event are displayed. Attribute 7 points out geographic locations such as Palo Alto (PA) and North America Remote Sites (NARS).

Observation of results.

1. The occurrence of error event 23 always precedes that of event 22.
2. The occurrence of warning event 21 signals the pending occurrence of error events 7 and 6 with a probability of 0.63 and 0.69, respectively. This can be used to forewarn of potential error conditions prior to their occurrence.
3. Occurrence of error event 7 always signals the pending occurrence of error event 6.
4. Error event 12 tends to generate a series of warning events (13, 14, 17, 18, 19, 20) with at least a 0.82 probability.
5. Warning event 1 tends to be a popular follow up to other events with a moderate probability. Using the black swan type events determination algorithm we can see that event 12 is a very good candidate for BSN. Putting a little bit of context into this it shows which error events tend to have a cascade effect of generating other events.

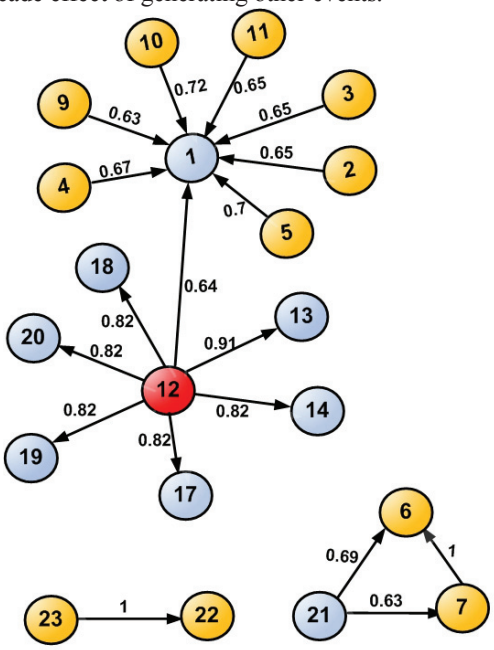

Fig. 4. VC events graph.

B. Applying to Task and VC Events Data

The second test of this algorithm consisted of consuming the same events data as mentioned above and also a set of task data which describes tasks performed by an operator. Since we deal with a semi-structured text-based data, the link between those data sets is the common attribute name. Here the proximity is defined by a common host name, data center (DC) name, and other attributes. The datasets time range was from June 12, 2010 to July 27, 2010 and contained over 370,000 events and over 16,000 tasks. To reduce the scope of the presentation only events with a categorization of "warning" or "error" are displayed here. Table II shows the various tasks of interest and Table III the relevant events. The abbreviation CS in those tables stands for "Computer Source.

Observation of results. Figure 5 shows the first obtained relationship between the tasks and the events. The orange colored nodes are tasks and the blue colored nodes are events. The interesting facts about this figure are:

1. When task 19 is executed 86% of the time it generates event 20.

2. When the above condition occurs, task 18 is then executed.
3. Also it seems that a third of the time when event 20 is generated it leads to event 48 to be generated (a potential cause perhaps of why task 19 failed).

Figure 6 shows an entirely different type of a behavior:

1. Task 39 (host.StorageSystem.rescanVmfs for host osdc-ph3-vmg1-3) is triggered and it leads to a host of other tasks (34, 35, 36, 37, 41).
2. 21% of the time when task 39 is executed it leads to event 27 which is a warning type event for a particular system.
3. It seems that the StorageSystem.rescanVmfs are a set of tasks that execute in order on different set of systems.

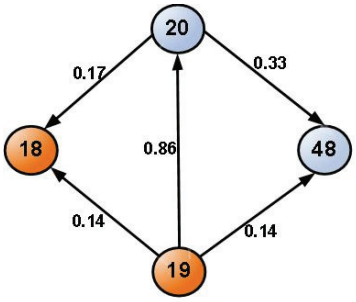

Fig. 5. Tasks and events.

Figure 7 shows the following:

1. Task 29 always precedes task 31 for the same host.
2. Interestingly, 13% of the time that event 30 is generated, it leads to the above tasks to be executed.

Figure 8 shows a very simple relationship between a task and an event:

1. Task 25 leads to event 24, 97% of the time.
2. It seems that this should happen 100% of the time but it doesn't which may mean a loss of events (just pure conjecture on our part).

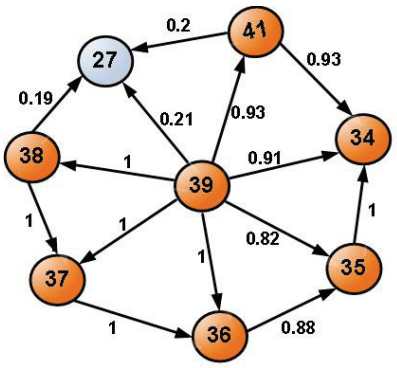

Fig. 6. Impact of task 39.

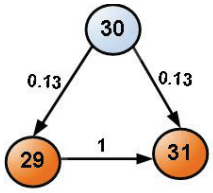

Fig. 7. Task 29 always precedes task 31.

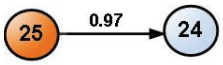

Fig. 8. Task 25 leads to task 24.

The observations or conclusions made above are purely based on the correlations and potential meaning behind the events and tasks. A person more familiar with the dataset may be able to arrive at other conclusions if appropriately armed with the correlation information.

Table I. Information about the VC events.

Event ID	Source	A1	A3	A5	A6	A7
6	**dc-vmg2-3**	**vim.event.HostReconnectionFailedEvent**	**E**		**dc-vmg2**	**PA**
2	**wdc-loadtest-vmg1-7**	**vim.event.HostConnectionLostEvent**	**E**		**wdc-loadtest-vmg1**	**NARS**
3	**wdc-loadtest-vmg1-5**	**vim.event.HostConnectionLostEvent**	**E**		**wdc-loadtest-vmg1**	**NARS**
4	**wdc-loadtest-vmg1-1**	**vim.event.HostConnectionLostEvent**	**E**		**wdc-loadtest-vmg1**	**NARS**
5	**wdc-loadtest-vmg1-4**	**vim.event.HostConnectionLostEvent**	**E**		**wdc-loadtest-vmg1**	**NARS**
7	**dc-vmg2-3**	**vim.event.HostCnxFailedBadUsernameEvent**	**E**		**dc-vmg2**	**PA**
9	**wdc-loadtest-vmg1-3**	**vim.event.HostConnectionLostEvent**	**E**		**wdc-loadtest-vmg1**	**NARS**
10	**wdc-devtest1-10**	**vim.event.VmFailedMigrateEvent**	**E**	**ora-keytmp2-ebs-d1**	**wdc-devtest1**	**NARS**
11	**wdc-devtest1-13**	**vim.event.VmFailedMigrateEvent**	**E**	**ora-qa-ksedw-d1**	**wdc-devtest1**	**NARS**
12	**dc-vmg2-3**	**vim.event.DrsResourceConfigureFailedEvent**	**E**		**dc-vmg2**	**PA**
22	**wdc-uat-vmg1-12**	**vim.event.DrsResourceConfigureFailedEvent**	**E**		**wdc-uat-vmg1**	**NARS**
23	**wdc-uat-vmg1-15**	**vim.event.DrsResourceConfigureFailedEvent**	**E**		**wdc-uat-vmg1**	**NARS**
1	osdc-ph3-vmg2-1	vprob.vmfs.resource.corruptondisk	W		osdc-ph3-vmg2	PA
13	osdc-ph3-wopsvmg1-4	vprob.vmfs.heartbeat.timedout	W		osdc-ph3-wopsvmg1	PA
14	osdc-ph3-wopsvmg1-2	vprob.vmfs.heartbeat.timedout	W		osdc-ph3-wopsvmg1	PA
17	osdc-ph3-wopsvmg1-4	vprob.vmfs.heartbeat.recovered	W		osdc-ph3-wopsvmg1	PA
18	osdc-ph3-wopsvmg1-2	vprob.vmfs.heartbeat.recovered	W		osdc-ph3-wopsvmg1	PA
19	osdc-ph3-wopsvmg1-1	vprob.vmfs.heartbeat.recovered	W		osdc-ph3-wopsvmg1	PA
20	osdc-ph3-wopsvmg1-1	vprob.vmfs.heartbeat.timedout	W		osdc-ph3-wopsvmg1	PA
21	dc-vmg2-3	vim.event.VmOrphanedEvent	W	ora-dtruong-test3	dc-vmg2	PA

Table II. Various tasks.

ID	Source	Entity ID	CS ID	DC ID	Entity Name	VM ID	Description ID
18	2251	5282	149	131	adm-col-test-2	5282	Drm.ExecuteVMotionLRO
19	1200	1226	166	131	portal-vmwperf-wlp-6	1226	vim.VirtualMachine.migrate
25	4951	4826	2943	131	ora-ent-r5u5-x86_64-template	4826	vim.VirtualMachine.clone
29	11325	11325	2943	131	dc-vmg2-2		vim.host.StorageSystem.rescanAllHba

31	11325	11325	2943	131	dc-vmg2-2		vim.host.StorageSystem.rescanVmfs
34	5058	5058	149	131	osdc-ph3-vmg1-6		host.StorageSystem.rescanVmfs
35	2365	2365	149	131	osdc-ph3-vmg1-9		host.StorageSystem.rescanVmfs
36	2251	2251	149	131	osdc-ph3-vmg1-7		host.StorageSystem.rescanVmfs
37	2113	2113	149	131	osdc-ph3-vmg1-5		host.StorageSystem.rescanVmfs
41	2305	2305	149	131	osdc-ph3-vmg1-8		host.StorageSystem.rescanVmfs
42	2472	2472	149	131	osdc-ph3-vmg1-1		vim.host.StorageSystem.refresh
43	11336	11354	11334	131	centos5-x86-64-template	11354	vim.VirtualMachine.clone
44	11336	11994	11334	131	dummy-vm	11994	vim.VirtualMachine.clone
45	4815	4048	2943	131	ubuntu-804-64-template	4048	Drm.ExecuteVmPowerOnLRO
49	632	632	65	38	wdc-devtest1-10		vim.host.StorageSystem.computeDiskPartitionInfo
50	632	632	65	38	wdc-devtest1-10		vim.host.DatastoreSystem.createVmfsDatastore

Table III. Tasks relevant events.

ID	Source	Event Type	Category	CS ID	DC ID	VM ID	Host Name
10	1683	vprob.vmfs.heartbeat.recovered	W	178	131		osdc-ph3-wopsvmg1-3
11	1681	vprob.vmfs.heartbeat.recovered	W	178	131		osdc-ph3-wopsvmg1-2
12	1563	vprob.vmfs.heartbeat.timedout	W	178	131		osdc-ph3-wopsvmg1-1
13	1798	vprob.vmfs.heartbeat.timedout	W	178	131		osdc-ph3-wopsvmg1-6
14	1563	vprob.vmfs.heartbeat.recovered	W	178	131		osdc-ph3-wopsvmg1-1
15	1798	vprob.vmfs.heartbeat.recovered	W	178	131		osdc-ph3-wopsvmg1-6
16	3112	vim.event.HostConnectionLostEvent	E	119	38		wdc-loadtest-vmg1-4
20	1200	vim.event.VmFailedMigrateEvent	E	166	131	1226	osdc-ph3-vmg2-2
21	1230	vim.event.HostConnectionLostEvent	E	119	38		wdc-loadtest-vmg1-5
22	1052	vim.event.HostIsolationIpPingFailedEvent	E	119	38		wdc-loadtest-vmg1-1
23	435	vim.event.DrsResourceConfigureFailedEvent	E	93	38		wdc-devtest3-3
24	4951	vim.event.MigrationResourceWarningEvent	W	2943	131	4826	dc-vmg2-1
26	4951	vim.event.VmCloneFailedEvent	E	2943	131	4826	dc-vmg2-1
27	2113	vprob.vmfs.journal.createfailed	W	149	131		osdc-ph3-vmg1-5
28	11336	vim.event.VmMessageWarningEvent	W	11334	131	11359	aburnett-esx1
30	4951	vim.event.HostConnectionLostEvent	E	2943	131		dc-vmg2-1
48	1200	vprob.vmfs.resource.corruptondisk	W	166	131		osdc-ph3-vmg2-2

VII. Algorithm Scalability

Figure 9 shows the total time in minutes to make a run vs. the total number of events processed (note the log scale on both axis). The regression line in this graph depicts the function $t = cE^{1.70}$, where t is time to complete, E is the number of events processed and c is the constant of proportionality. This shows that this algorithm is of order 1.70 with respect to the total number of events processed.

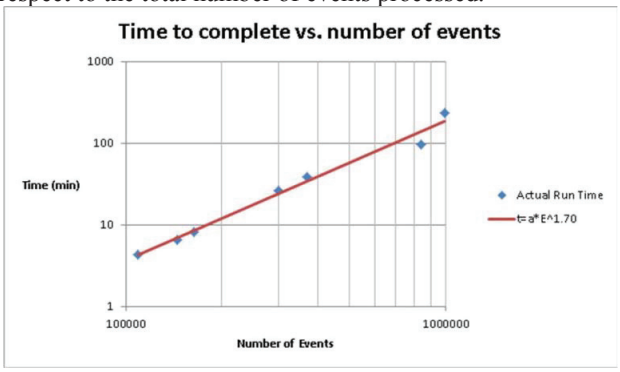

Fig. 9. Time complexity in number of events.

VIII. Conclusions

In this paper we showed a new formulation to obtain information from any data that can be turned into the format of (timestamp, source, attributes). Terming each entry as events we were then able to obtain a Directed Virtual Graph (DVG) showing relationships and conditional probabilities between event pairs. Once a DVG has been obtained other algorithms (Black Swan, Critical Node, Critical Path, etc.) can be used to derive specific conclusions regarding the data set. This technique was applied to a VC diagnostics dump of 610,000 events. The DVG of this data set showed various relationships between error type events. It also identified events that can lead to a cascade of other events. We believe the application of this technique to other data sets can provide valuable information in terms of relationships and consequence of occurrence.

Reducing the wheat from the chaff is the first step of any large data set analysis and this formulation can become the first level filter as it requires no contextual knowledge of the data and applies no rules.

References

[1] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley 1991.

[2] Narsingh Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall 1974.

[3] S.S. Skiena, *The Algorithm Design Manual*, Springer 2008.

[4] N.N. Taleb, *The Black Swan: The Impact of the Highly Improbable*, Random House 2007.

[5] D.A. Ostrowski, A framework for classification of unstructured data, *IEEE Int. Conf. Semantic Computing*, Berkeley, CA, USA, September 14-16, pp. 373-377, 2009.

[6] A.A. Shariff K, M.A. Hussain, and S. Kumar, Leveraging unstructured data into intelligent information – analysis and evaluation, *Int. Conf. Information and Network Technology*, IPCSIT, vol. 4, IACSIT press, Singapore, pp. 153-157, 2011.