A Comprehensive Informative Metric for Analyzing HPC System Status using the LogSCAN Platform

Yawei Hui*, Byung Hoon Park†, Christian Engelmann‡
Computer Science and Mathematics Division, Oak Ridge National Laboratory
*huiy@ornl.gov, †parkbh@ornl.gov, ‡engelmannc@ornl.gov

***Abstract*—Log processing by Spark and Cassandra-based ANalytics (LogSCAN) is a newly developed analytical platform that provides flexible and scalable data gathering, transformation and computation. One major challenge is to effectively summarize the status of a complex computer system, such as the Titan supercomputer at the Oak Ridge Leadership Computing Facility (OLCF). Although there is plenty of operational and maintenance information collected and stored in real time, which may yield insights about short- and long-term system status, it is difficult to present this information in a comprehensive form. In this work, we present system information entropy (SIE), a newly developed metric that leverages the powers of traditional machine learning techniques and information theory. By compressing the multivariant multi-dimensional event information recorded during the operation of the targeted system into a single time series of SIE, we demonstrate that the historical system status can be sensitively represented concisely and comprehensively. Given a sharp indicator as SIE, we argue that follow-up analytics based on SIE will reveal in-depth knowledge about system status using other sophisticated approaches, such as pattern recognition in the temporal domain or causality analysis incorporating extra independent metrics of the system.**

***Index Terms*—Metrics, Cloud computing, Visual analytics**

I. Introduction

Along with the ever increasing sophistication of high performance computing (HPC) systems, reliability, availability and serviceability (RAS) logging systems have been accordingly designed to record extensively large volumes of system information that are stored majorly in log files. These log files hold data collected from various hardware and software monitoring components, reflecting system booting records, hardware failures, memory allocation/usage anomalies, network status, file system performance, configuration and run-time information of user applications, and many more aspects. The relative significance of these informative indicators changes over time as the HPC system status evolves with independent external driving forces of various natures. For example, running a well-tested user application would incur much fewer errors than running the same application when it is still in its early implementation stage, and the contrast and change in the system status would correspond accordingly. In many cases, RAS researchers choose a few well-defined system variables/features and build analysis around them without a broader consideration of the full picture from the perspective of system evolution. In contrast, this paper introduces a solution to summarize the system information that is immune to specific selections of representative variables/features and grasps the evolving system status in a flexible yet comprehensive way.

In order to meet many challenges imposed by the volume and the complexity of the log files accumulated at the Oak Ridge Leadership Computing Facility (OLCF) for the Titan supercomputer, we have developed a big data analytical platform – Log processing by Spark and Cassandra-based ANalytics (LogSCAN) – that provides flexible and scalable data acquisition, transformation and computation on the Titan log files [1], [2]. Taking in a range of different Titan log files such as "console" outputs, "consumer" reports and "netwatch" transcripts, LogSCAN uses several sophisticated log parsers to pre-process these files and the results are stored in multiple Cassandra database tables. By allowing multiple users to access these tables via different computational schemes (e.g., Jupyter notebooks and Scala applications), LogSCAN enables users to manipulate the data in accord with their unique perspectives and carries out computational workflows with their preferred analytical platform/packages within the same programming environment.

Coming with the power of a multi-user platform like LogSCAN is the diverse representation of data sets that are exposed to the analytic workflow. We notice that most of the representations can be abstracted in a general form of a 2D data table, with each row being a record and each column a feature. For instance, there are many ways to organize the event logging information of the Titan supercomputer. One could use the pristine format of the event records in which a single entry contains basic information like the time and location of the event. Or one could transform either the whole event history or part of it into a refined format in which every record has a much richer set of features derived from user-defined filters/algorithms. Since the analytic goals and specific computational schemes vary among researchers, it is practically impossible to predict the layout of the data set being eventually fed into a workflow. With the change of context in

which the analysis is being conducted, one or a few features for the same set of records may become irrelevant (or relevant). For example, users might be interested in the investigation of the system's temporal evolution (as our work in this paper). In this case, the relative significance of the included features will vary over time and users may find it hard to identify a fixed set of features for their entire analysis.

One goal of LogSCAN is to provide an effective means to summarize the system status regardless of the way a user manipulates the original logging information in a specific workflow. As long as the data set is in compliance with the general layout of record vs. feature, we show with convincing examples in this article that there is a universal computing paradigm for a metric to describe a system's status. This metric – system information entropy (SIE) – is designed to mitigate the problem caused by arbitrary choices of feature sets while bringing in a comprehensive system status indicator that leverages the powers of traditional machine learning techniques and information theory. SIE is easily calculated and, while highly compressed, sensitively traces the change of a system.

In the following sections, we first briefly review related work in Section II. In Section III, we give implementation details of LogSCAN relevant to this work and describe the essentials of the SIE definition in Section IV. We continue in Section V by showing several usage cases of SIE as a system status metric. An extensive discussion will be given in Section VI and a concise summary of our report in Section VII.

II. Related Work

Many recent works [3]–[6] leveraged the increasing power provided by the innovation in machine learning research. Being able to define a set of target system properties which supposedly represent the key characteristics of the system, researchers have found correlations among these properties and interpret/predict the system behaviours according to their implications. For example, Gainaru et al. [5], [7] took a hybrid approach by combining signal analysis to characterize events (with three pre-defined types of signals: periodic, noise and silent) and data-mining algorithms to find patterns between them. No matter what the start point is (logging information [3], [8], resource usage [6], performance characteristics [6], [8], etc.), the first crucial task is to create reliable indicator(s) representing the overall (or partial) system status.

There has been a continuous effort on quantitatively characterizing the system status based on the analysis of the logging information. In closely related works [4], [9]–[11] to our study, many authors applied similar algorithms – principal component analysis (PCA) and information entropy (IE) – in their approaches. Depending on the exact target of the analysis, however, some of the consequences after applying indiscriminately the dimensionality techniques focused on reducing the dataset may negatively impact the results by eliminating critical features in anomaly detection [6], [10]. We provide an easily-computed comprehensive system status indicator which takes no pre-assumption about the relative significance of the system properties chosen for investigation. Such a system indicator, we argue, will facilitate follow-up analytics and reveal in-depth knowledge about system status using other sophisticated approaches, such as pattern recognition in the temporal domain or causality analysis incorporating extra independent metrics.

III. Data Preparation

Introduced in Section I, LogSCAN is designed to process the logging information collected from Titan, a HPC at OLCF manufactured by Cray with 27 petaflops of theoretical peak performance and consists of 18,688 AMD Opteron 16-core CPUs, 693.5 TiB total memory and 18,688 NVidia Tesla K20X GPUs that are spread over 400 hundred cabinets (4 nodes per blade, 24 blades per cabinet). Many implementation details of LogSCAN, deployed at the Compute and Data Environment for Science (CADES) at ORNL, have been presented in a separate article [2]. Without much redundancy, we provide here a brief supplementary description of LogSCAN and demonstrate its typical data wrangling process for the data manipulation and transformation.

A. Application Platform

LogSCAN consists of an Apache Cassandra cluster with 16 worker nodes and two independent analytic Apache Spark clusters named "logcat" and "bh-cloud", respectively. "logcat" is a 16-node Spark cluster deployed on top of the Cassandra cluster maximizing data locality, while "bh-cloud" consists of three "fat" nodes residing in a different subnetwork, which we designate to run highly customized jobs requiring long executions. In terms of total computational resources, "logcat" and "bh-cloud" have comparable amount of cores and memory [2].

In our study, we found that benefits from data locality provided by "logcat" are significant when interactive jobs access the Cassandra cluster frequently. On the another hand, long-time analysis running on "bh-cloud" could safely ignore the amortized penalty when making few but computationally intensive queries to the Cassandra cluster at the beginning. For the rest of the analysis, it simply executes iterative tasks around one or a few objects stored in memory.

Depending on the actual computing paradigm, users of LogSCAN may prefer using one cluster over the other when configurations of hardware platforms and software stacks are of concern. For example, instead of reading repeatedly small portions of the data table (which is mostly the case in an interactive Spark job), one may choose to read the whole table from the Cassandra cluster and cache it in memory. This happens more likely as the user submits the application via a jar which performs iterative computations in order to best reduce the amortized I/O cost. A caveat associated with this approach, though, is that it significantly increases the need for computational resources. For example, it increases the possibility of running out of memory, as observed on the "logcat" cluster.

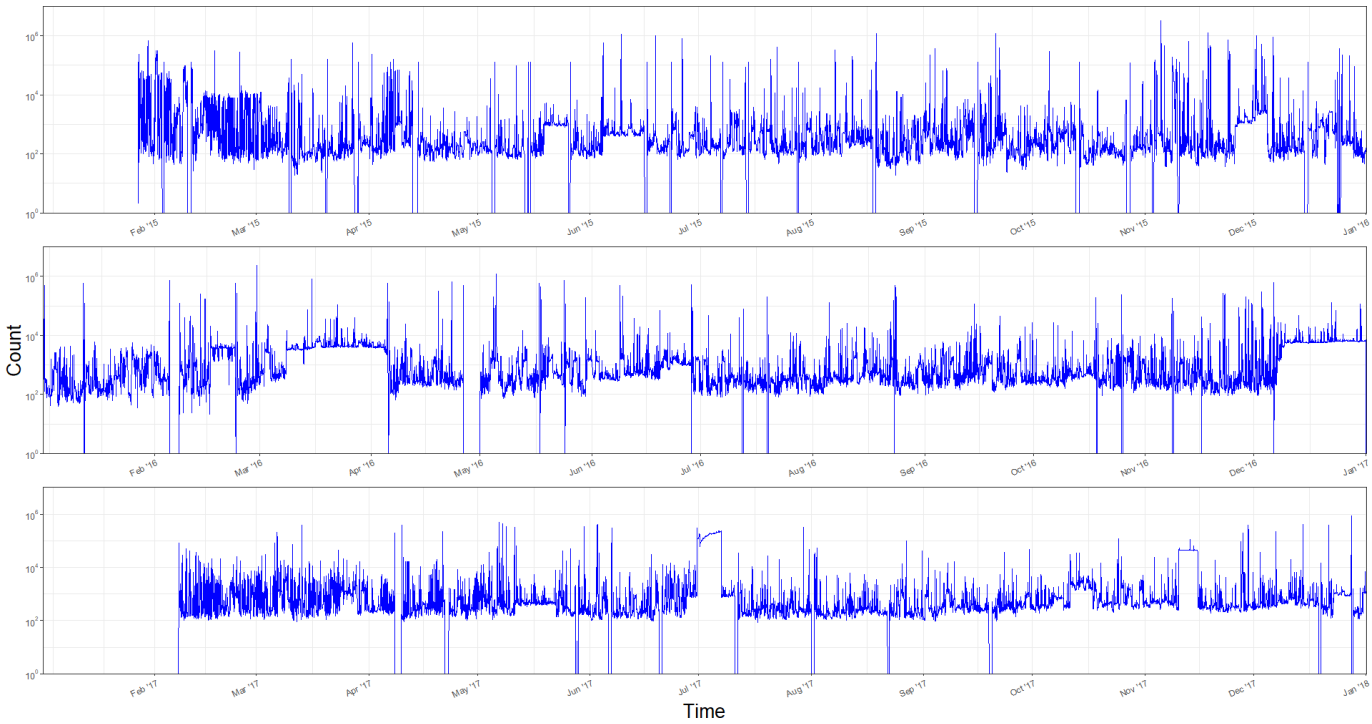

Fig. 1: The total event counts accumulated in calendar year 2015, 2016 and 2017. The resolution in time is by hour, which means the count number plotted at any given time is the total counts recorded on all Titan nodes within the past hour.

B. Data Reduction

LogSCAN's parsers take as input several prominent types of Titan log files and the processed records are transformed into system events. Each event is stored as a single record in a specific Cassandra table and uniquely marked, according to its logging nature, by a combination of event properties. The most crucial selection of properties considers: 1) event time, i.e., when being registered in the logging system (e.g., at "2016-03-03 10:38:54"); 2) event type, i.e., what is the plausible cause (e.g., Graphics Engine Error (GEE), or Out of Memory (OOM)); and 3) event source, i.e., where being registered in terms of the physical position in Titan's architecture (e.g., on the node "c8-3c1s6n1" – for the nodal naming convention of the Titan architecture see [2]).

Counts	Percentage	ID	Description
10	0	1	DVS Confusion
2,998,492	2	2	NVRM Xid
5,671,348	4	3	Machine Check Exception (MCE)
1,229	0	4	NVRM DBE
49	0	5	Unknown GPU Error (UGE)
302,969	0.2	6	Graphics Engine Error (GEE)
5,732	0	7	Kernel Panic
782,337	0.5	8	Out of Memory (OOM)
16,938,194	11.6	9	HWERR
1,215,780	0.8	10	Seg. Fault
43,268,141	29.5	11	Lustre
31,498,746	21.5	12	LNet
992,997	0.7	13	LNet Error
42,809,426	29.2	22	Lustre Error
146,485,450	100		

TABLE I: The non-zero event types and their occurrences in Titan's logs between January 2015 to March 2018.

LogSCAN's parsers have been so far designed to recognize 22 event types (see Table I) and will be extended to include more in future versions. The total number of unique event records from the 3-year logging history of Titan stored in the Cassandra cluster is around 146.5 million, each represented by a single row in the event table as shown in Table II. Statistically speaking, from a beginning time at "2015-01-27 06:19:54" to the end of log records at "2018-03-09 09:11:42", four event types - "Lustre", "Lustre Error", "LNet", and "HWERR" - dominate with their total counts (in millions) of 43.3, 42.8, 31.5, and 16.9, respectively. All other event types accumulate to numbers of order of magnitudes lower. To show a panorama view of the whole event data set, we plot in Figure 1 the total event counts accumulated in calendar year 2015, 2016 and 2017, respectively, with the time resolution of an hour.

Although the general definition of the SIE (introduced in the next section) as a system metric can be applied to any data organized into a 2D table of record vs. feature, we emphasize that, the focus of this study is to develop a temporally quantifiable low-dimensional metric in a form of a time series in order to describe the evolutionary history of the system. It is obvious that the crude form of the event table described above is inadequate for this purpose and a properly justified combination of time-window and time-step is needed

Time	Event Type	Source
2015-01-28 05:00:42	3	c10-4c2s6n0
2015-01-28 05:52:18	22	c7-2c1s6n0
2016-04-05 13:48:16	12	c20-0c1s4n3
2016-08-24 01:00:35	11	c2-1c0s1n2
2015-12-16 03:19:36	9	c3-5c0s4n2
2017-12-07 10:05:01	11	c18-7c0s2n1

TABLE II: An example of the event table after the data wrangling process. In total, the table contains 146.5 million events logged on all Titan nodes during the period of January 2015 to March 2018. For any specific translation from a numeric event type to its literary name (such as "3" representing "Machine Check Exception" (MCE)), see Table I for definitions.

to guarantee meaningful statistics when calculating SIE in such a circumstance.

Looking at Table II, one can see there may be several events of different types registered on different Titan nodes (or "event source") at each time stamp recorded in the table. By "pivoting" (a Spark dataframe operation that dominates the data manipulation cost in our analysis) on the event table according to distinctive event types, we acquire the event history for each registered event source. Theoretically, the event table at any time (in a natural unit of a second) can be represented by a Spark dataframe, where each row is a specific source while each column a specific event type. In reality, however, it is computationally infeasible and practically unintuitive to use a dataframe with such a fine time resolution to inspect long-term system status. Instead, to reflect the compressed history of the system status in a time series, we apply a time window to the event table along the observed temporal dimension and accumulate events within this window for each combination of event source and type. Practically, we choose to advance an hour-wide window by steps of a minute. The resulting event table is represented as a $M \times P$ Spark dataframe with M event sources and P event types as illustrated as an example in Table III.

Taking a closer look at Table III, we see that one of the columns – event source – is actually a composite feature in a sense that it can be further split to provide more details about the source. For example, the first record in Table III shows that all the events registered within the past hour occur at Titan node "c0-2c2s1n2", which represents "Cabinet Row 0 by Column 2, Chassis 2, Slot 1 and Node 2" (see [2] for the nodal naming convention). In fact, as being revealed in the next section, the calculation of SIE does not dictate any specifics about how one arranges the data set in the format of record vs. feature. To demonstrate the generality of SIE as a system metric, we create a second data layout other than the original shown in Table III by re-arranging the table in such a way that the event source is mapped one-on-one to a pair of X and Y coordinates in a 2-D nodal layout map, mimicking the floor map of Titan. More specifically, if denoting a set of grid coordinates [$Cabinet\_Row$, $Cabinet\_Column$, $Chassis$, $Slot$, $Node$] which could be explicitly read out from

Source	Machine Check Exception	Out of Memory	Seg. Fault	Lustre
c0-2c2s1n2	1	0	0	0
c10-4c0s0n0	0	0	0	4
c10-4c0s3n0	0	1	0	3
c10-4c1s6n1	0	0	3	1

TABLE III: An example of the time-windowed event table for one hour starting from "2016 Jan. 27, 05:00:00" and ending at "2016 Jan. 27, 06:00:00". For an illustrative purpose, only 4 rows are shown out of a total of 400.

the source name, one way to perform the source-to-coordinate map is:

$$\begin{aligned} X &= 12 * Cabinet\_Row + 4 * Chassis + Node \\ Y &= 8 * Cabinet\_Column + Slot \end{aligned} \tag{1}$$

Here, the choice of mapping scheme is quite arbitrary and can be replaced with any other scheme deemed suitable. The direct output of the mapping is the addition of two new properties – X and Y in the nodal map – for each record in the event table. Considering all the records in a given event table, such as Table III, they could be pinned down at [X, Y] in a new 2D data table with their cell values filled in with the counts from either a single event type or several types combined. Any cell at a position where no event has been registered will simply take a value of zero. As the input to the computation of SIE, this new 2D layout (referred to as "Nodal_Map") will possess a dimensionality of [300, 64], as determined by the mapping scheme in Equation 1. It is quite different from the original layout (referred to as "Source_Type"), as it has a varying dimensionality. In Figure 2, we illustrate the nodal layout for five event types and the combined total at the time of "2015-02-09 01:49:54" in Titan's logging history.

IV. Representative Model of System Status

Given a collection of log files, the information of the system status could be extracted and stored in a 2D data table as demonstrated in Section III-B. The general format of these data tables is record vs. feature. If taking into account a finite time resolution, the study of the historical evolution of system status will introduce a third dimension – time. For example, if considering the layout "Source_Type" defined in Section III-B (also demonstrated in Table III), we have assembled, at any chosen moment for a pre-defined time window, a 2D data table with records along the direction of event source (19,200 distinctive nodes) and features of event type (22 distinctive types). We then advance this process by a fixed time step and the whole history of system status can be represented as a 3D data set. It is important to note that the 2D table of record vs. feature could take a variety of manifestations as, in our cases, of different layouts – "Source_Type" and "Nodal_Map". From alternative perspectives in how to represent the information of system status, one should be able to create many new layouts for this 2D table.

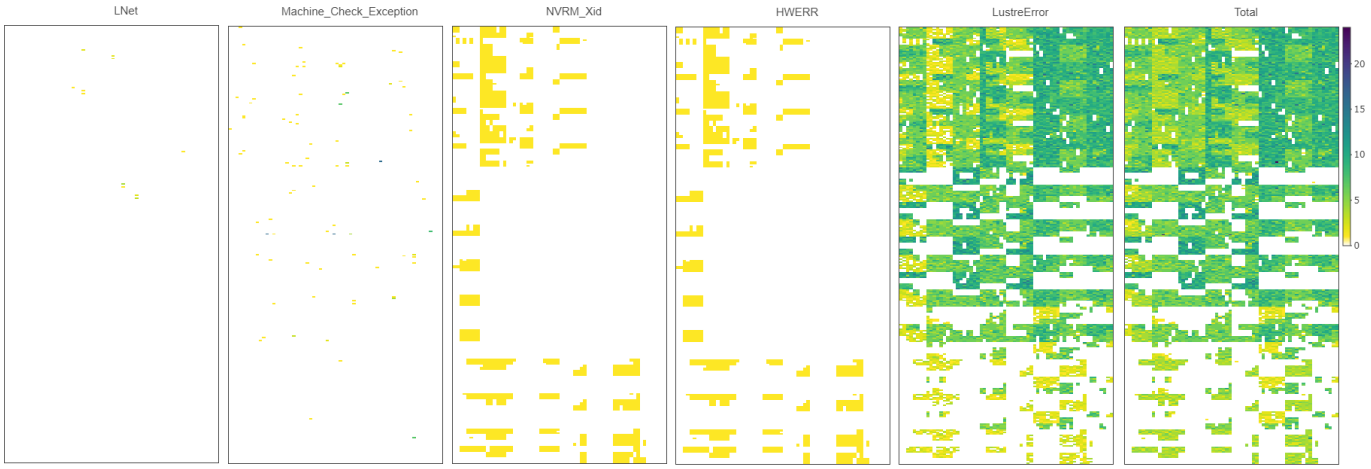

Fig. 2: The nodal layouts for every event type and the combined total for the past hour prior to "2015-02-09 01:49:54". Each layout has dimensions of [300, 64] in pixel and each pixel represents a unique Titan node with its coordinates $[X, Y]$ translated from its source name via Equation 1. The count numbers are plotted with common color bar with a top taken from the maximum total counts among all nodes.

Provided such a 3D data set, the key question is how to come up with a quantifiable low-dimensional metric, which incorporates as much system information as in a high-dimensional representation. The solution we propose in our study is to compress the history of system status along dimensions of record and feature in a representative metric (i.e., SIE) and put it in a time series. Two widely used algorithms – Principal Component Analysis (PCA) [12] and Information Entropy (IE) – have played key roles in our design of the SIE. The applications of both, PCA and IE, in acquiring SIE are independent of the structural details of the data table, thus hold great potential for generalization in other similar system status analyses.

A. Principal Component Analysis (PCA)

In many cases of PCA applications, the algorithm has been used to reduce data dimensionality for subsequent analysis. If a 2D data set is represented in a record vs. feature format, one can use PCA to find the, so-called, principal components (PCs) in a (usually) high-dimensional space spanned along the feature directions of the data set. Mathematically speaking, the non-zero set of PCs is a linear combination of the original features and generally considered to best represent the originals by maximizing the feature variance along the directions of the identified PCs. Instead of focusing on reducing the dimensionality of the data set, in our study, we are more interested in the overall degree of feature correlation which can be calculated with the relative variances among all PCs. Among many standard methods to obtain feature variances along PCs, we choose to factorize the co-variance matrix by certain matrix decomposition procedure. Since LogSCAN includes a Spark cluster as its computational framework, we naturally select the highly optimized Singular Value Decomposition (SVD) in the Spark MlLib package to compute feature variances along PCs and define a probability vector out of these PC variances as:

$$\xi_i = \frac{\sigma_i}{\sum_1^k \sigma_i} \tag{2}$$

where σ_i (i from 1 to k) is the i-th variance calculated from k eigenvalues of the SVD decomposition.

Once the variance probability $\vec{\xi}$ is calculated for a certain 2D table of record vs. feature, the original 3D data set is transformed into a 2D space spanned by time and $\vec{\xi}$. To store system status in a more compacted way, we go to our second procedure of information compression.

B. Information Entropy (IE)

Used as a numeric indicator of the average amount of non-redundant system information, Information Entropy (aka. Shannon entropy) was first developed by Shannon in his seminal paper for information theory [13]. In our design of SIE, we use information entropy as an information compression algorithm to mathematically quantify the information content of a random signal with a certain probability distribution.

Given the variance probability vector $\vec{\xi}$ obtained from Section IV-A, we define the Information Entropy, H, via an analogy to the Boltzmann's formula:

$$H = -\sum_1^k \xi_i log_b(\xi_i) \tag{3}$$

Here the definition of IE takes a general form of, so-called, "b-ary entropy", while, in our study, the logarithm with base 10 has been adopted. At any given moment, the variance probability vector is transformed into a value of H, which renders the compressed 2D data set of time vs. $\vec{\xi}$ from Section IV-A as a time series of $H(t)$. As demonstrated in the following sections, $H(t)$ sensitively responds to varieties

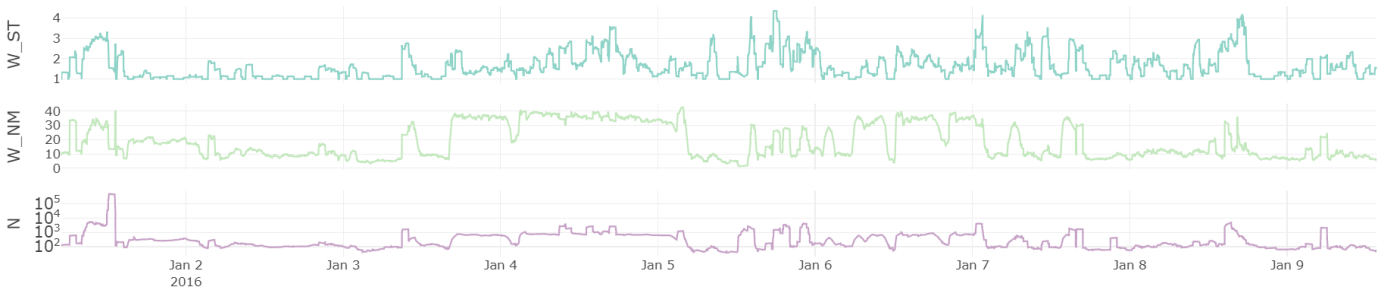

Fig. 3: A historical panorama of system status plotted with 3 time series. The top panel represents the SIE for the "Source_Type" layout, while the he middle panel represents the SIE for the "Nodal_Map" layout (see Section III-B). The bottom panel shows the total event counts accumulated within a one hour window prior to the given time spot. All three plots share the same time resolution of one minute and use distinctive color codes, i.e., green for "Source_Type", blue-green for "Nodal_Map" and purple for total counts, which are kept consistently for all plotting in our paper.

of driving forces in the system which shape the general behavior over time in a comprehensive manner, especially as the dimensionality of the feature space increases with the complexity of the system being monitored.

Equipped with $H(t)$, we now define a general metric of the system status – System Information Entropy:

$$W(t) = b^{H(t)} \tag{4}$$

Here b is the specific logarithmic base used in the definition of H, and it takes the value of 10 in our work. The preference of using W as SIE over H can be justified with the following two extreme examples. Suppose there is a distribution of $\vec{\xi}$ that only a single PC variance dominates all others and H will be very small with a minimum of $H = 0$ according to Equation 3. In this case, the value of W will approach 1, which simply means all observed features are highly correlated and could imply they share a single source of driving force (e.g., a single user or a large-scale application). On the other hand, if $\vec{\xi}$ takes an even distribution among all PC variances, H will approach its maximum value of $H = log_b(k)$. Accordingly, W will be close to k, which strongly indicates that each feature is driven by an independent source (e.g., multiple users or various applications utilizing different system resources). In summary, W is an effective indicator of the number of independent features the system possesses at a given time.

V. Application

The Titan log data used in our study was collected from early 2015 to early 2018. For the purpose of simplicity and clarity, to demonstrate the SIE (or W) as a comprehensive indicator of the system status, we selectively pick a few snapshots in segmented times. The criterion for our choices is arbitrary as long as the illustrations showcase exemplary applications in the chosen time intervals. In the following, we first show a historical panorama of the system status and then zoom in to some interesting time intervals to see how the SIE sheds new lights on interpreting system status in contrast to what we have learned from simple event counts.

A. Overall Feature Recognition

In Figure 3, we show 3 time series from both SIEs (W) and total counts (N). While for a given time spot, the fix-sized accumulation window determines that N is the same for either of the event table layout, a dramatic difference exists in W for the two layouts. For simplicity, we use W_{ST} and W_{NM} respectively to represent the SIEs for the two layouts (where "$_{ST}$" stands for "Source_Type" and "$_{NM}$" for "Nodal_Map").

B. System Information Entropy (SIE)

At a first glance, we could see both W_{ST} and W_{NM} respond accordingly to N at large scale during this period of time from "2016-01-01 05:14" to "2016-01-09 13:38", while differences between either one of them and the total counts are unmistakably standing out at many medium to small time scales. The overall system status started with a few "violent" jumps showing on all three curves prior to "2016-01-01 15:00", where W_{ST} and W_{NM} share a similar profile which is much less extreme comparing to N. Before reaching about "2016-01-03 09:00", the system ran in a relatively stable status which could be seen with both W_{ST} and N, while the "activeness" of W_{NM} reveals the sources of the events recorded in this period of time were changing quite dramatically. Following on, we see all three curves took a few jumps before reaching a common "plateau" which spans one and a half days (from "2016-01-03 16:00" to "2016-01-05 05:30", approximately). Taking a closer look, we can see the two Ws respond to N in a rather different way. For example, around "2016-01-04 01:40" to "2016-01-04 02:40", both Ws registered a similar dip in values as did N. However, at between "2016-01-04 08:50" to "2016-01-04 17:40", only W_{ST} responded significantly to N while W_{NM} remained docile. By "2016-01-05 19:30", all three curves resigned from the relatively stable "plateau" and entered a long-term oscillating status until reaching the end of the time interval under investigation.

From the crude observation conducted on the large scale features presented by both W_{ST} and W_{NM}, we see that they did catch up with N, the original information source regarding system status. However, as a combined and compressed system

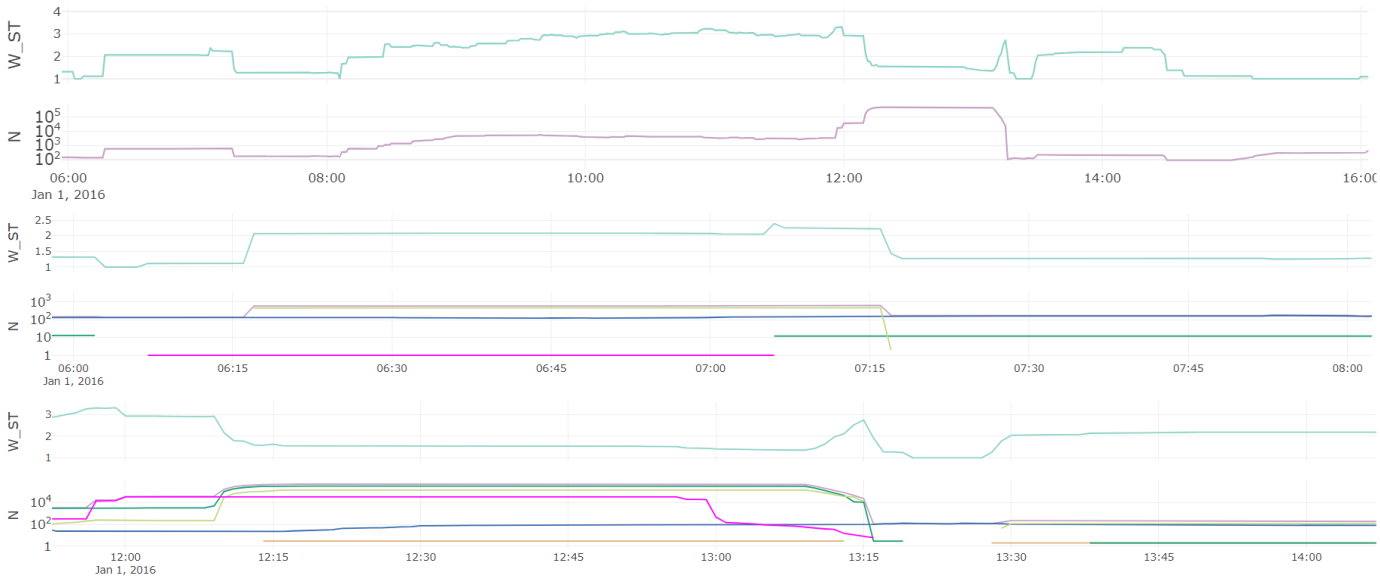

Fig. 4: A series of close-ups from Figure 3 for the layout "Source_Type". On the top panel, we plot the time series of the information entropy in form of W and the total counts N. The zoom-in takes a time interval from approx. "2016-01-01 06:00" to "2016-01-01 16:00". On the bottom panel, two plots are drawn for illustrative examples described in the text (see Section V-C). In addition to the total counts, the event count for each non-zero event type is drawn in these two plots, as well. The color codes for event types are: "Lustre", "LNet", "Machine Check Exception" (in short, "MCE") "LustreError" and "HWERR".

metric, we expect W could provide more information about system status. To verify this point, we need to take a much closer look at their temporal behaviors.

C. *"Source_Type" Layout*

We start our scrutiny on detailed SIE by first exploring the "Source_Type" layout. In Figure 4, we take several close-ups of the same time series shown in Figure 3 at various scales. The top panel is a zoom-in of Figure 4 with the time interval around "2016-01-01 06:00" to "2016-01-01 16:00", where W and the total counts N contrast in two time locations.

The first happens at around "2016-01-01 07:06" when N barely has any change while W shows an abrupt jump in its value. Before this moment, W shows a nearly constant value around 2.1, which practically indicates there are slightly more than two independent features representing the system status. At the moment of the spike, either new features appear in the system, or existing features become significant, or both. Also the system status has been altered in an indecisive way because the value of W changed only by a small fraction. Following the spike, a drop of W indicates a similar process in the opposite direction happens, i.e., either features cease to exist in the system, or a less number of features become dominant, or both. Referring to the definition of the "Source_Type" layout, the feature is essentially total counts for each event type accumulated in the time window on Titan nodes. On the middle panel of Figure 4, the plot shows a further zoom-in from the plot on the top panel. In different colors, we plot the total counts for all event types with non-zero values. Right before the jump, there are three features (i.e., event types in this case) - "LNet", "Machine Check Exception" (in short, "MCE") and "Lustre Error". The W, taking a value of 2.1, accurately represents the system status with the fact that "LNet" and "MCE" dominate "Lustre Error". Upon the jump, a new feature ("Lustre") appears in the system and gives the W a small boost. For a very short time, all four features exist in the system until "Lustre Error" disappears. This new change in the system status manifests in form of a very small drop in the W which is not surprising given the contribution from "Lustre Error" is initially quite small.

The second analysis focuses on a time interval from approximately "2016-01-01 12:00" to "2016-01-01 14:00". As briefly mentioned in Section V-A, there is a common "plateau" for both W and N. However, at the end of the "plateau", N shows a sharp drop while, in contrast, W takes a big jump followed by a sharp drop within a small period of time. Taking a closer look at the time series shown on the bottom panel in Figure 4, we see there are five features (i.e., "Lustre", "LNet", "MCE" "Lustre Error" and "HWERR") present before the jump. Since "Lustre" and "LNet" dominate over others to a great extent, it's not surprising that the W does not change much before the jump. Staring after "2016-01-01 13:10", both "Lustre" and "LNet" take a downturn and approach in value the other features, while "MCE" stays quite constant. The changes of collective relative significance continue to drive the W to a higher range until "LNet" disappears abruptly and "Lustre" takes such a sharp drop in value. Correspondingly, the W takes a downturn and eventually rests on a number merely

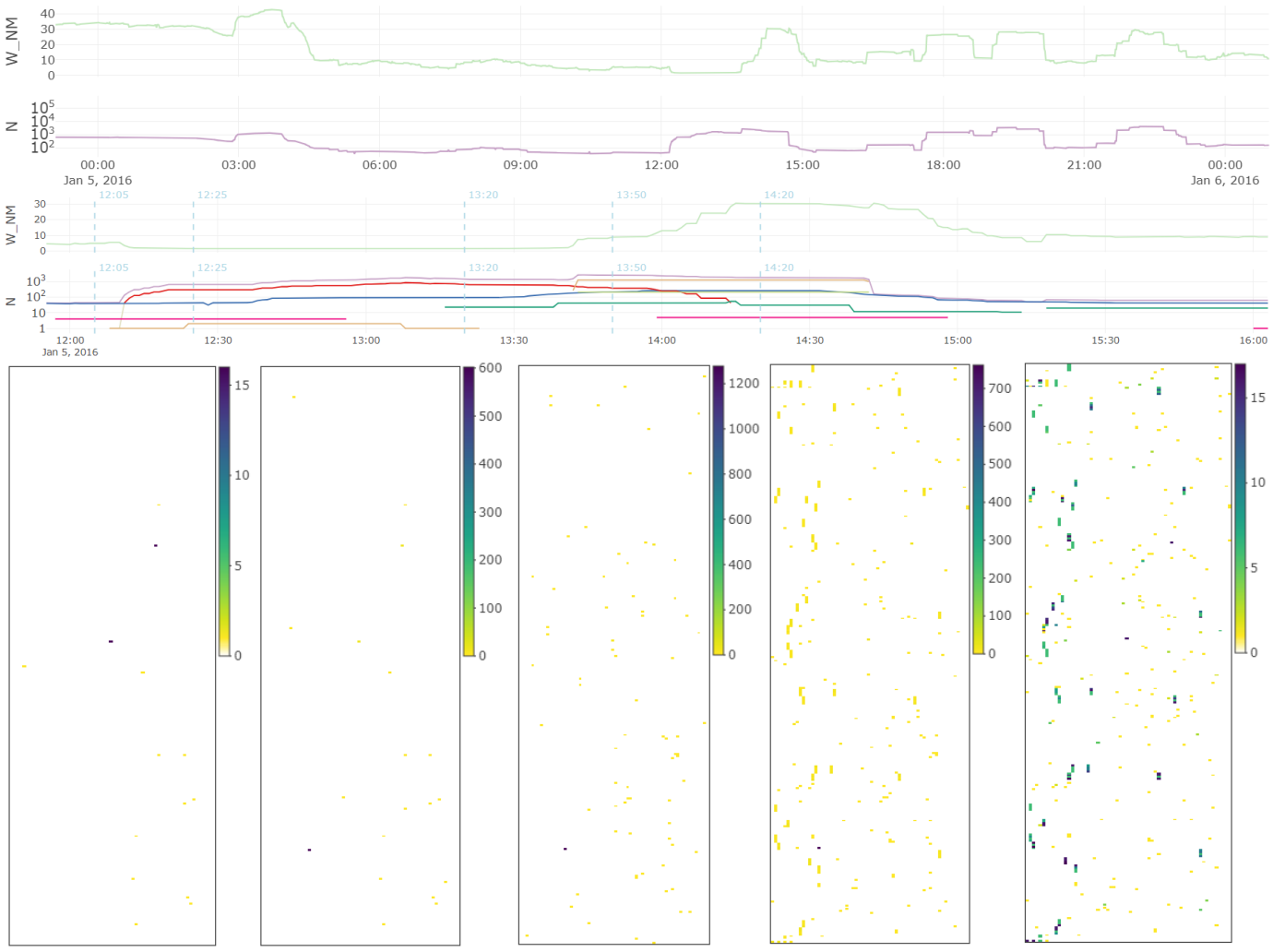

Fig. 5: A close-up for SIE of "Nodal_Map" layout is shown for a time interval from "2016-01-05" to "2016-01-06" on the top panel, while a zoom-in shown on the middle panel focuses on a smaller time span of "12:00" to "16:00". To distinguish among different event types when drawing the event count, the color codes used are: "Machine Check Exception" (in short, "MCE"), "Seg Fault", "HWERR", "NVRM Xid", "Graphics Engine Error" (in short, "GEE"), "Lustre", and "LNet". For almost all the time from "12:10" to "14:14", the number of event counts for "NVRM Xid" and "GEE" are identical, resulting in the overlap curves with only "GEE" shown explicitly. On the bottom panel, heat maps are plotted at specific times (chosen and marked on the zoom-in SIE plot) at "12:05", "12:25", "13:20", "13:50" and "14:20". In comparison to Figure 2, the color scales vary dramatically from moment to moment due to the range of the total counts recorded.

above one which reflects the system status is dictated by a single dominant feature ("MCE").

From both cases, we see the SIE (W) faithfully represents system status in a very comprehensive and sensitive manner. In the next Section, we demonstrate that the same analysis approach can be applied to another layout (i.e., "Nodal_Map") which reveals different aspects of the system's characteristics.

D. "Nodal_Map" Layout

In comparison to the "Source_Type" layout, "Nodal_Map" presents the event table in a different perspective. In this layout, the features are directly associated with the Titan nodal positions which are mapped to a 2D matrix with dimensions [300, 64], instead of using a multi-dimensional grid (to be exact, a 5D grid, which is impractical for any effective visualization, see Section III-B). Consequently, the SIE calculated in this layout can behave quite differently, as briefly discussed in Section V-A. Overall, W in "Nodal_Map" has a much larger dynamic range for its values due to the fact that there are much more potential independent features in this layout (64 nodal map columns) comparing to "Source_Type" (theoretically, only 22 event types could be possibly registered at the same time).

In our close-up study on the "Nodal_Map" layout, as shown in the first plot on the top panel of Figure 5, we choose a

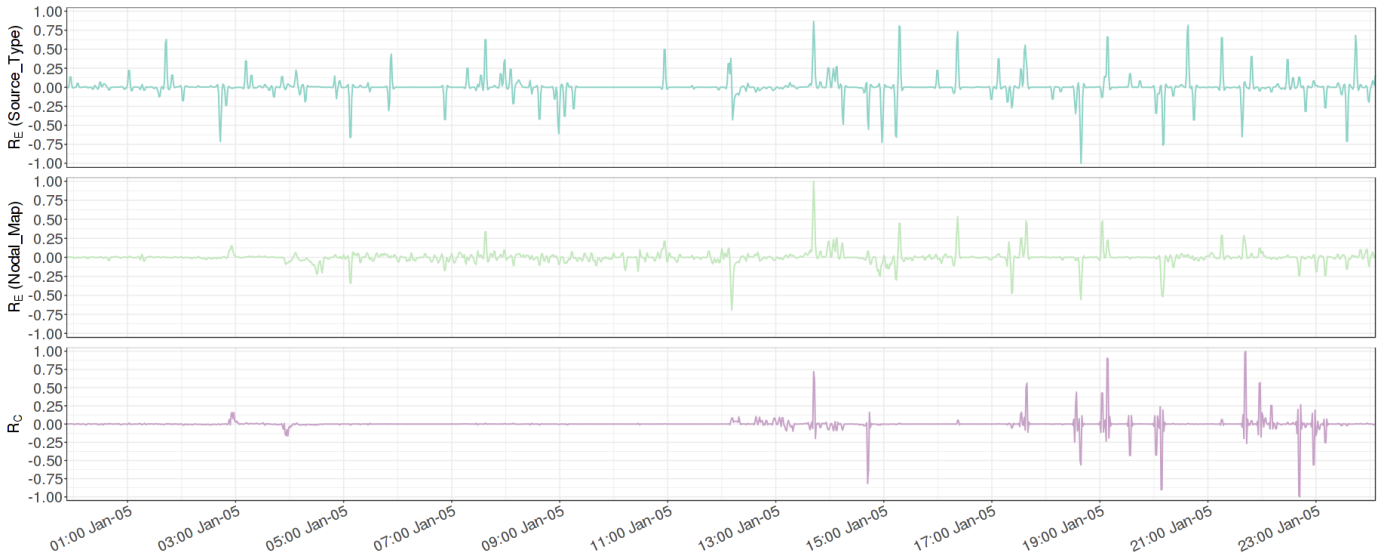

Fig. 6: Two examples of the change rates for the information entropy (R_E on the top and middle panels for layouts of "Source_Type" and "Nodal_Map", respectively) and the total counts (R_C on the bottom panel).. The relative change rates are normalized with respect to the maximum values of R_E and R_C within the time interval, respectively, which explains why there is only one reading of 1.0 for either of R_E or R_C. The time interval (from about 2016-01-05 00:00:00 to 2016-01-06 00:00:00) is chosen to coincide with that used in the detailed discussion on the zoom-in panel shown in Figure 5 (see Section V-D).

small time interval ("2016-01-05" to "2016-01-06"). In most part of the time series, W follows up with the variation of N. There is one segment in time (approximately from "12:00" to "16:00") raising quite a bit of interest in our investigation. A further zoom-in is plotted on the middle panel in Figure 5. Before "12:10", W essentially "copies" the behavior of N for an extended amount of time. After this turning point, we see a substantial increase in the total counts contributed by event types "NVRM Xid" and "Graphics Engine Error" (in short, "GEE"). However, the W drops to a value close to 1, which indicates the system enters a status where a few hot spots appear (meaning very localized events are recorded along the time). The trend keeps on for both W and N until at approx. "13:40". After that moment, without significant increase of event counts (actually, it is slightly decreasing instead), the W shows a steady march for higher and higher values until it reaches its top number of 30 around "14:15". This simply tells us that the localized hot spot(s) disappear and the system is returning back to a "random state" with a higher entropy.

Since the features in the layout are nodal map columns, we choose several representative moments marked by vertical lines in the plot on the middle panel. We then draw, at these moments, the corresponding heat maps on the bottom panel for total event counts, meaning all event types combined. The heat map at "12:05" shows a relatively quiet moment of the system where no specific location on the nodal map records events beyond "normal". In the second heat map at "12:25", we see a dramatic change of the system status in which a single hot spot appears on Titan node "c4-1c0s7n1" with a dominant total counts of 602. This critical change explains exactly why the W drops sharply from above 5 to almost 1. Following on to the third heat map in the middle of the bottom panel, we can see there are other event types entering the picture at very dispersed nodal positions, while the hot spot created by "NVRM Xid" and "GEE" is intensified with more counts registered by these two event types. The net effect on W is that W keeps almost unaltered. After "13:40", there is a new significant change in the total counts which introduces a new event type "HWERR". Unlike "NVRM Xid" and "GEE", counts from "HWERR" are not localized at all (easily seen in the fourth heat map), which naturally contributes to the W jump observed at "13:40". This trend keeps pushing on until after "14:15" when the relative dominance of the hot spot created by "NVRM Xid" and "GEE" totally die out. By then, the W reaches its local maximum indicating the system is running quite evenly in terms of nodal distribution and this description of system status is satisfyingly verified by referring to the last heat map.

VI. Discussion

As a natural extension in analyzing a time series, we calculate time derivatives of the SIE (R_E) described in the two applications in Section V. For comparison, we also calculate the change rate of the total counts (R_C). When calculating R_E, the derivative of the time series of $H(t)$ has been used instead of $W(t)$. This choice is taken to keep the current gradient analysis in line with future analysis, such as evolution of dynamic systems (suggested in the following). Mathematically, the two choices of computing R_E differ by a scale factor of W.

In Figure 6, we show R_E for both layouts of "Source_Type" and "Nodal_Map" and R_C for total counts. The time interval

is chosen so that it mostly coincides with that used in the detailed discussion on the middle panel shown in Figure 5 (see Section V-D). The change rates plotted in Figure 6 are normalized with the maximal values of R_E and R_C within the time interval, respectively, which explains why there is only one reading of value 1 for either of R_E and R_C. The first impression from Figure 6 is that R_E is much more sensitive, in comparison to R_C, to the outside signals (no matter what their natures are). When giving definitions of layouts "Source_Type" and "Nodal_Map" in Section III-B, we know "Source_Type" possesses a much smaller degree of freedom (DOF) in the feature space (22 possible event types) when comparing to "Nodal_Map" which has 64 potential event source positions localized to Titan's nodal columns. This discrepancy in DOF may explain the observed higher sensitivity level in R_E for "Source_Type" when comparing to "Nodal_Map". Due to the normalization of the change rates, we expect the sensitivity levels of R_E vs. R_C will change with different time and/or time span. However, overall we observe a higher sensitivity level in R_E than R_C.

Zooming into further details on these plots, we identify a general pattern that we call the "limited window effect" (LWE). This artificial effect is purely created by the design of how to represent the evolution history of the system without substantially harming the statistical meanings of any analysis output (see Section III-B). In one of the two significant manifestations of LWE, we see a pair of mirrored features which is highly anti-symmetric about the base line of value "0". This is especially obvious in the R_C plot in Figure 6 where, with every significant positive change rate, there is a counterpart in the negative half space. For example, one pair of these "mirrored features" on the bottom panel in Figure 6 appears with the positive R_C right before "2016-01-05 03:00" and the negative counterpart after an hour of time. Compared to the R_C plot, LWE imposes a much lesser footprint in the plots of R_E, which is likely attributed to the fact that information entropy as a comprehensive status metric is diversely driven by many system variables other than dictated by a single source of information.

Besides the "mirrored features", there is another LWE which accounts for the smearing of feature details within a fixed time window due to a prominent event. Looking back at Figure 4 or Figure 5, one can easily identify typical step-like features with a width of approximately 1 hour (i.e., our chosen size of time window). As shown with the calculation of R_E and R_C, taking the time derivative can efficiently eliminate this smearing effect of LWE.

VII. Conclusion

Without pre-assumption about the relative significances among system properties, we design a metric which reflects the system status in a concise form of a time series. It has been demonstrated that this system metric comprehensively yet sensitively summarize the overall system characteristics without compromising on computational efficiency. By adopting SIE as a robust system status metric, we envision further applications, such as pattern recognition, causality analysis in conjunction with other independent system metrics (hardware health indicators, user job submission patterns, etc.), and similarity analysis as a dynamic system. Generally speaking, any analysis approach that takes a time series as its ultimate input could be applied to SIE, and its application will create much needed insight into the evolution of HPC system status over time.

Acknowledgment

This work is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, program manager Lucy Nowell, under contract number DE-AC05-00OR22725, and by the Compute and Data Environment for Science (CADES) facility and the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is managed by UT Battelle, LLC for the U.S. DOE (under the contract No. DE-AC05-00OR22725).

References

[1] B. H. Park, S. Hukerikar, R. Adamson, and C. Engelmann. "Big Data Meets HPC Log Analytics: Scalable Approach to Understanding Systems at Extreme Scale", IEEE International Conference on Cluster Computing (CLUSTER), 2017, pp. 758–765.

[2] B. H. Park, Y. Hui, S. Boehm, R. A. Ashraf, C. Layton, and C. Engelmann. "A Big Data Analytics Framework for HPC Log Data: Three Case Studies Using the Titan Supercomputer Log", IEEE International Conference on Cluster Computing (CLUSTER), 2018, in publication.

[3] S. Di, R. Gupta, M. Snir, E. Pershey, and F. Cappello. "LOGAIDER: A Tool for Mining Potential Correlations of HPC Log Events", 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2017, pp. 442–451.

[4] A. Goudarzi, D. Arnold, D. Stefanovic, K. B. Ferreira, and G. Feldman. "A Principled Approach to HPC Event Monitoring", Proceedings of the 5th Workshop on Fault Tolerance for HPC at eXtreme Scale - FTXS '15, ACM Press, New York, New York, USA, 2015, pp. 3–10.

[5] A. Gainaru, F. Cappello, M. Snir, and W. Kramer. "Fault prediction under the microscope: A closer look into HPC systems", IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, 2012, pp. 1–11.

[6] O. Tuncer, E. Ates, Y. Zhang, A. Turk, J. Brandt, V. J. Leung, M. Egele, and A. K. Coskun. "Diagnosing Performance Variations in HPC Applications Using Machine Learning", ISC 2017: High Performance Computing, Springer, Cham, 2017, pp. 355–373.

[7] A. Gainaru, F. Cappello, S. Trausan-Matu, and B. Kramer. "Event Log Mining Tool for Large Scale HPC Systems", 17th International Conference, Euro-Par 2011 Parallel Processing, E. Jeannot, R. Namyst, and J. Roman (Eds.), Springer, Berlin, Heidelberg, 2011, pp. 52–64.

[8] W. Yoo, M. Koo, Y. Cao, A. Sim, P. Nugent, and K. Wu. "Performance Analysis Tool for HPC and Big Data Applications on Scientific Clusters", Conquering Big Data with High Performance Computing, Springer International Publishing, R. Arora (Eds.), 2016, chapt. 7, pp. 139–161. 52009,

[9] S. Fu. "Performance Metric Selection for Autonomic Anomaly Detection on Cloud Computing Systems", IEEE Global Telecommunications Conference - GLOBECOM 2011, 2011, pp. 1–5.

[10] Q. Guan, and S. Fu. "Adaptive Anomaly Identification by Exploring Metric Subspace in Cloud Computing Infrastructures", IEEE 32nd International Symposium on Reliable Distributed Systems, 2013, pp. 205–214.

[11] Z. Lan, Z. Zheng, and Y. Li. "Toward Automated Anomaly Identification in Large-Scale Systems", IEEE Transactions on Parallel and Distributed Systems, 2010, vol. 21(2), pp. 174-187.

[12] K. Pearson F.R.S. "LIII. On lines and planes of closest fit to systems of points in space", The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1901, vol. 2(11), pp. 559–572.

[13] C. E. Shannon. "A mathematical theory of communication", The Bell System Technical Journal, 1948, vol. 27(3), pp. 379–423.