Workload Characterization for Stability-As-A-Service

Vaishali Sadaphal and Maitreya Natu
Systems Research Lab, Tata Research Development and Design Centre (TRDDC), Pune.

***Abstract*—*Stability As A Service* is critical for data centers and workload characterization is one of the essential services within *Stability As A Service*. Different solutions have been proposed in the past for characterizing various system properties, however most of these approaches often require high levels of instrumentation and intrusiveness. This makes their use difficult in real-world production systems. In this paper, we argue that many interesting insights can be derived to characterize system workload simply by analyzing the transaction logs captured at different layers in the system such as access logs at application servers, SQL logs at database servers, among others. We present two approaches that analyze various dimensions and measures of the transaction logs in order to characterize workload. We demonstrate the effectiveness of the proposed approach through real-world case-studies and show how the findings of the two approaches complement each-other.**

Keywords: Stability As A Services, Performance and Capacity Management, Workload Characterization

I. INTRODUCTION

Today's computing requirements make data centers large and complex. After the initial infrastructure and application planning, the data centers keep evolving and growing to accommodate newer requirements. The continuously evolving nature and the ad-hoc growth of the data centers increases the complexity manifold. In one of the leading telecommunication companies, we observed the number of users multiplying every 15 days resulting in addition of infrastructure components, and installation of more application instances. Due to continuous evolution and ad-hoc growth, the data centers become brittle. Even a small change in an application or infrastructure carries a risk of breakdown and instability. Compromising stability of the system and breach of SLAs (Service Level Agreements) result into incurring large amount of financial losses. All these factors have made *Stability As A Service* critical for data centers. Goal of *Stability As A Service* is to ensure zero down-time, provide consistent performance, and quickly detect and resolve any instability.

A quintessential requirement for providing *Stability As A Service* is the understanding of the as-is state of system operations. The data center operators need to understand the workload patterns, performance areas, system hotspots, bottlenecks, among others. In this paper, we address one part of this puzzle by focusing on the problem of workload characterization. We present algorithms to demonstrate how the state of the art logs can be analyzed in a systematic, efficient, and automated manner to extract various properties of system workload. Workload characterization can be exposed as a service to provide the analysis of system workload properties at various layers. This service can extract properties such as workload patterns, heavy hitters, anomalous behavior, etc. The service can be used in the overall offering of *Stability As A Service* to obtain insights to better understand system behavior, do base-lining of as-is behavior, identify heavily-used and poor-performing areas, and plan for growth and optimization.

Problems related to workload characterization have been addressed in the past [11], [6]. Solutions have been proposed based on techniques such as analysis of paths followed by requests, generation of resource signatures for the request types, observing temporal communication patterns using frequent subgraph discovery, etc. Effectiveness of most of these solutions depends on the availability of wide instrumentation and use of intrusive techniques. However, the data center administrators are reluctant to introduce intrusive solutions in the production environment because of the risk of modification of system behaviour. This makes many prior solutions difficult to deploy in real-world operations. A practical solution for workload characterization is to perform analysis using logs that are commonly captured in most operational systems with minimal instrumentation and intrusiveness. The workload is commonly captured at various layers in a system hierarchy in the form of access logs. For instance, the application servers store the transaction logs, database servers store SQL logs, and disks capture the read and write operations. We argue that analysis of these logs, obtained by simple, non-intrusive, at-a-point monitoring solutions, can also provide a practical and easy-to-use solution while providing many useful insights.

In this paper, we target analysis of semi structured or structured application monitoring logs. These logs are parsed to construct a multi-column dataset where each column maps to a metric being monitored. Each row contains the value of the observed metric at a time instance. The metrics are classified into two types viz. measures and dimensions. The measures are numeric representatives of various system properties such as latency, arrival rate, etc. The dimensions are categorical attributes that provide structured labeling information to the measures. The dimensions thus provide a mechanism to filter, group, and label the measures. For instance, consider a web transaction log that contains an entry of each request received by the web server containing the request time stamp, HTTP method (GET/POST), URL, client IP address, and the time to respond. In this log, the time to respond metric is a measure while HTTP method, URL, and client IP address are the dimensions. In this paper, we present algorithms to analyze

 ISBN: 978-1-61208-215-8

hidden relationships between these measures and dimensions. Extracting such relationships helps answering questions such as (1) which set of requests should be optimized to result in maximum improvement, (2) requests with which dimension value require largest amount of resource, etc.

The workload or application monitoring log typically contains a dump of all types of activities observed in the system. Various details of the activities such as workload patterns, heavy hitters, anomalous behavior, etc., tend to get lost in the monitoring logs. For instance, in the example of web transactions log, all types of transactions are reported in a single log. Various properties of these individual groups of transactions are thus lost in this collective log. This paper addresses the problem of understanding the workload characteristics captured in the monitoring logs.

Existing literature in the area of data-mining can be leverages to analyze the dimensions and measures of these logs. Techniques for clustering, feature selection, subgroup discovery, detection of components in Gaussian mixture models, can be leveraged for developing workload characterization services. In the following, we present two approaches for workload characterization:

Consider a database P, with N records, where each record has a measure L and k dimensions $D = \{D_1, D_2, \ldots, D_k\}$. Each dimension D_i consists of its domain $Domain(D_i) = \{d_1^i, \ldots, d_n^i\}$ that represent all possible values of the dimension D_i. A dimension-value pair is referred as $(D_i = d_j)$. A descriptor θ denotes a given record type which consists of one or more dimension-value pairs.

Multi-modal analysis (MMA): Using the observation that the real world processes follow Gaussian distributions, we discover Gaussian distributions in the measure L to characterize the workload with respect to a measure. Given these modes of measure values, we next use the dimension information to tag each mode with a descriptor θ that best represents the records belonging to a mode. *Interesting subset discovery (ISD):* In contrast to the multi-modal analysis, in this approach we first explore the space of dimensions. We first identify possible combinations of dimension descriptors, identify the set of records that are explained by the dimension-descriptors, and then use the measure L to compute interestingness of the subset.

Both approaches complement each other in their findings as follows:

- While multi-modal analysis analyzes the entire spectrum of measure values, interesting subset discovery only focuses on the extreme ends of interestingly high and low measure values.
- Multi-modal analysis tries to find the dominant property and tries to describe that property with a representative dimension-value pair. The set of descriptors, thus identified, most often represent large sets. Interesting subset discovery, on the other hand, identifies the sets that are interestingly different in their statistical properties. The set of descriptors, thus identified, represent anomalies an most often represent small and medium sets.

We discuss the approach for multi-modal analysis in Section II and interesting subset discovery in Section III. We demonstrate the utility of the proposed approaches through real-world case-studies in Section V. We summarize our contributions in Section VI.

II. MULTI-MODAL ANALYSIS (MMA)

In order to develop this approach, two sub-problems need to be solved - (1) how to construct modes for a workload measure?, and (2) how to identify descriptors that best describe the requests that belong to one mode? We address these two problems below:

A. Construction of modes

Given a measure L with n values and mean and standard deviation as $< \mu, \sigma >$, we identify the set M consisting of m modes $M = \{M_1, M_2, \ldots, M_m\}$ that best classify the values of L. A mode M_i is a set of values in measure L that belong to one mode or Gaussian distribution. A 3-tuple $< \mu_i, \sigma_i, w_i >$ describes each mode M_i where μ_i, σ_i is the mean and standard deviation of values in mode M_i and w_i is the weight defined as the fraction of points belonging to the mode M_i given by $\frac{|M_i|}{n}$ where $n = \sum_{i=1}^{i=m} |M_i|$.

We use Expectation maximization algorithm [5] to estimate the parameters of the modes such as mean and standard deviation. Host of algorithms exist to identify number of modes in a given set of values [12]. However, the modes produced by these algorithms need to be refined for various reasons, such as (1) Many modes may be very similar in their mean values and are required to be merged. (2) Modes may have very large standard deviation and they need to be split to form multiple modes with smaller standard deviation. The algorithms proposed in the past require manual tuning of a parameters that merge or split the modes.

We address this problem by defining a self-tunable threshold T_{merge} on the basis of which we merge or split the modes. We use coefficient of kurtosis to tune the threshold T_{merge}. The coefficient of kurtosis is a statistical measure of values that signifies the peakiness of a distribution. For Gaussian distributions coefficient of kurtosis is known to be near-zero. We compute coefficient of kurtosis of the entire set of values and infer if the distribution is uni-modal or multi-modal. We merge the modes aggressively in case the computed values of kurtosis is near zero assuming that the distribution in uni-modal else we assume that the distribution is multi-modal and adopt a conservative approach while merging the modes. Another way is to compute coefficient of kurtosis for each identified mode and split the mode in case the distribution is inferred to be multi-modal on the basis of value of coefficient of kurtosis.

B. Identifying mode-descriptors

Given a set M of modes, we find the descriptor $\theta = (D_i = d_j)$ that has largest probability of explaining the values of measure L observed in a mode. We present two approaches to identify mode-descriptors. We first present a

score-based approach that assigns scores to each dimension descriptor to find the best descriptor. We then present a less compute-intensive feature-selection-based approach that uses classification and regressions trees to identify descriptors. The basic idea is to evaluate each dimension on how well can its descriptors describe all modes.

1) Score-based approach: We first compute the probability that the descriptors $(D_i = d_j)$ explains a mode and then present an approach to compute score of dimension using score of descriptors.

Computing probability that a descriptor explains a mode: The probability p_{ij}^x that a descriptor $(D_i = d_j)$ explains a mode $M_x \in M$is computed by identifying the percentage of values in M_x that hold property $D_i = d_j$.

Formally, if r_k is the k^{th} record, $Value(r_k, D_i)$ is the value in the dimension field D_i of the record r_k, and $Value(r_k, L)$ is the value in the measure field L of record r_k. then,

$$p_{ij}^x = \frac{|M_x = \{d_j | D_i = d_j)|}{N} \quad (1)$$

where M_x is the set of records in which $Value(r_k, D_i) = d_j$.

Computing score of a dimension: The score of a dimension D_i indicates how well is each of the mode described by the descriptor of dimension D_i. In order to calculate the score of a dimension D_i, first the probability of best descriptor of D_i is calculated for each mode. For a mode M_x the best D_i descriptor $(D_i = d_j)$ is defined as the descriptor with maximum probability p_{ij}^x. The score of a dimension D_i is then calculated by taking an average of the probability of the best descriptor for each mode.

The score-based approach computes the score for each dimension using the above approach. The dimension with maximum score is chosen as the best dimension. For each mode, the descriptor of the best dimension that has maximum probability is selected as the best descriptor.

2) Feature selection based approach: The score-based method is computationally intensive since it requires inspecting the records for every value of every dimension. We next propose an efficient algorithm that uses Classification and Regression Trees (CARTs) [3] to identify the best dimension to describe all the modes. The basic idea is to assign a class label to each record based on the mode to which the record belongs. We then use CART to identify dimensions that best classify these class labels. CART is constructed such that the class labels form the leaves and the dimension descriptors form the intermediate nodes.

CARTs can be used in two ways: One approach is to construct the entire CART and select the dimension that dominates across the non-leaf nodes. A simpler and more efficient approach is to select the dimension used as the root node. While building a CART, the root node is chosen as the node that provides the largest improvement in the classification accuracy of the leaf nodes.

Incorporating multiple dimensions: In certain cases, no single dimension best describes all the modes. To address such scenarios, we extend the proposed approach to analyze multiple dimensions to identify the best descriptor for every mode. We do this by identifying multiple candidate dimensions that can act as classifiers. Instead of using only one dimension with largest improvement in classification accuracy in CARTs, we propose to also use next few candidate dimensions with respect to the improvement in classification accuracy. We then identify the best descriptors across these selected multiple dimensions.

III. Interesting Subset Discovery (ISD)

The basic idea behind the proposed approach is to exploit the workload descriptors such as URLs, client IPs, and other dimensions of the dataset to construct subsets and then use the workload measures such as workload, performance, throughput, and other measures to define interestingness of the subsets. In order to develop this approach, two sub-problems need to be solved - (1) how to compute interestingness of a subset? (2) how to efficiently navigate through the large search space of all possible descriptors and their combinations? We address these two problems below:

This problem is similar to the subgroup discovery problem, which is well-known in data mining [2], except that the *class label* column (e.g. response time) in continuous in our case, rather than a finite discrete set. In the past, we have used the problem of interesting subset discovery in the domain of ticket analysis for IT infrastructure support [10]. In this paper, we present an application of interesting subset discovery for workload characterization and demonstrate how its findings complement the findings of multi-modal analysis.

A. Interestingness of a subset

Let descriptor θ denote a given record type and let $D[\theta]$ be the set of record of type θ. We build multisets $L[\theta]$ and $\overline{L}[]$ consisting of only the values of measure L for the records in the set $D[\theta]$ and its complement $\overline{D}[\theta])$ respectively. We use the 2-sample 2-tailed Student's t-test to compare the multisets $L[\theta]$ and $\overline{L}[\theta]$. Student's t-test makes a null hypothesis that both these sets of L values are drawn from the same probability distribution. It computes a t-statistic for two sets X and Y ($L[\theta]$ and $\overline{L}[\theta]$ in our case) as follows:

$$t = (X_{mean} - Y_{mean}) / \sqrt{(S_x^2/|X| + S_y^2/|Y|)}$$

S_X, S_Y denote the unbiased estimators of the standard deviations of the values in X and Y The denominator is a measure of the variability of the data and is called the *standard error of difference*. Another quantity called the p-value is also calculated. The p-value is the probability of obtaining the t-statistic more extreme than the observed test statistic under null hypothesis. If the calculated p-value is less than a threshold chosen for statistical significance (usually 0.05), then the null hypothesis is rejected; otherwise the null hypothesis is accepted. Rejection of null hypothesis means that the means of two sets do differ significantly. A positive t-value indicates that the set X has higher values than the set Y and negative t-value indicates smaller values of X as compared to Y.

 ISBN: 978-1-61208-215-8

B. Construction of subsets

Thus given a specific descriptor, we can use the t-test to decide whether or not the descriptor defines an interesting set. The main question now is how to systematically and efficiently search the space of all possible sets to identify interesting sets.

We build subsets of records in an incremental manner starting with level 1 subsets and increase the descriptor size in each iteration. The subsets built in first iteration are level 1 subsets. These subsets correspond to the descriptors $(D_i = u)$ for each dimension $D_i \in D$ and each value $u \in DOM(D_i)$. The subsets built at level 2 correspond to the descriptors $\{(D_i = u), (D_j = v)\}$ for each pair of distinct dimensions $D_i, D_j \in D$, for each value $u \in DOM(D_i)$ and $v \in DOM(D_j)$.

The brute-force approach is to systematically generate all possible level-1 descriptors, level-2 descriptors, ..., level-k descriptors. For each descriptor θ construct subset D_θ of D and use the t-test to check whether or not the subsets L_θ and $\overline{L_\theta}$ of their measure values are statistically different. If yes, report D_θ as interesting. Clearly, this approach is not scalable for large datasets, since a subset of N elements has 2^N subsets. We next propose various heuristics to limit the exploration of the subset space.

1) The size heuristic: The *t-test* results on the subsets with very small size can be noisy leading to incorrect inference of interesting subsets. Small subset sizes are not able to capture the properties of the record dimensions represented by the subset. Thus by the size heuristic we apply a threshold M_s and do not explore the subsets with size less than M_s.

2) The goodness heuristic: While identifying interesting subsets of records that have performance values greater than the rest of the records the subsets with the performance values lesser than the rest of the records can be pruned. In the web transaction records case, as we are using the case of identifying the requests that perform significantly worse than the rest of the requests in terms of the *response time*, we refer to this heuristic as the goodness heuristic. By the goodness heuristic, if a subset of records show significantly better performance than the rest of the records then we prune the subset. We define a threshold M_g for the goodness measure. In the case of the web transaction records database with *response time* as the performance measure, a subset is pruned if the *t-test* result of the subset has a *t-value* < 0 and a *p-value* $< M_g$.

3) The p-prediction heuristic: A level k subset is built from two subsets of level $k-1$ that share a common $k-2$ level subset and the same domain values for each of the $k-2$ dimensions. The p-prediction heuristic prevents combination of two subsets that are statistically very different, where the statistical difference is measured by the *p-value* of the *t-test*. We observed that if the two level $k-1$ subsets are statistically different mutually, then the corresponding level k subset built from the two sets is likely to be less different from the rest of the data.

Consider two level $k-1$ subsets D_{θ_1} and D_{θ_2} of the database D. Let the *p*-values of the *t-test* ran on performance data of these subsets and that of the rest of data are p_1 and p_2 respectively. Let p_{12} be the mutual p-value of the *t-test* ran on the performance data L_{θ_1} and L_{theta_2}. Let D_{θ_3} be the level k subset built over the subsets D_{θ_1} and D_{θ_2} and p_3 be the p-value of the *t-test* ran on the performance data L_{θ_3} and $\overline{L_{\theta_3}}$. Then the p-prediction heuristic states that *if*$(p_{12} < M_p)$ *then* $p_3 >$ *min*(p_1, p_2), where M_p is the threshold defined for the p-prediction heuristic. We hence do not explore the set D_3 if $p_{12} < M_p$.

4) Beam search strategy: We also use the well known beam search strategy [4], in that after each level, only top b candidate descriptors are retained for extension in the next level, where the beam size b is user-specified.

5) Sampling: The above heuristics reduce the search space as compared to the brute force based algorithm. But for very large data set (in the order of millions of records) the search space can still be large leading to unacceptable execution time. We hence propose to identify interesting subsets by performing sampling of the data set and using the above mentioned heuristics on the samples. The algorithm then retains only the most frequently occurring subsets in results obtained from several samples.

C. Algorithm for interesting subset discovery

Based on the above explained heuristics, we present Algorithm ISD for discovery of interesting subsets in an efficient manner. The algorithm builds a level *k* subset from the subsets at level *k-1*. A level *k-1* descriptor can be combined to another level *k-1* descriptor that has exactly one different dimension-value pair.

Before combining two subsets, the algorithm applies the p-prediction heuristic and skips the combination of the subsets if the mutual *p*-value of the two subsets is less than the threshold M_p. The subsets that pass the p-prediction heuristic test are tested for their size. Subsets with very small size are pruned. The remaining sets are processed further to identify records with the dimension-value pairs represented by the subset-descriptor. The interestingness of this subset of records is computed by applying the *t-test*. The interesting subset-descriptors are identified in the result subset L.

The algorithm then applies the goodness heuristic on each of the level k subset-descriptors to decide if the subset descriptor should be used for building subset-descriptors in subsequent levels.

IV. Related work

Workload characterization has been addressed in various different ways in the past. Graph mining techniques have been used to characterize requests based on similarity of request paths [6]. Requests have been characterized by identifying signature of resource demands of requests using machine learning techniques such as blind source separation [11]. We present a practical approach to provide a first-cut understanding of the system using the basic and most commonly available transaction logs. The insights captured complement the previous approaches. Also, the results can provide guidelines for

 ISBN: 978-1-61208-215-8

capturing more information in order to use other sophisticated techniques.

The problem of automatically discovering interesting subsets is well-known in the data mining community as *subgroup discovery*. Much work in subgroup discovery [1], [7], [9], [8], is focused on the case when the domain of possible values for the performance measure column is finite and discrete. In contrast, we focus on the case when the domain of the performance measure column is continuous. Also, many quality measures are used to evaluate the interestingness of subgroups and to prune the search space. A new feature of our approach is the use of Student's t-test as a measure for subgroup quality.

Unlike subgroup discovery and other techniques such as identifying anomalies, we propose to analyze the complete spectrum of the measure values. Furthermore, most of existing approaches construct groups from the perspective of dimensions. In contrast, we propose to form groups based on the measure values and then identify attributes describing these groups.

V. Application on real-world case-studies

We applied the proposed algorithms in various real-world case-studies. In this section, we present how the proposed algorithms successfully derived many useful insights across a wide-variety of case-studies. We have masked or not disclosed some part of the datasets due to privacy reasons.

A. Analysis of web-transactions

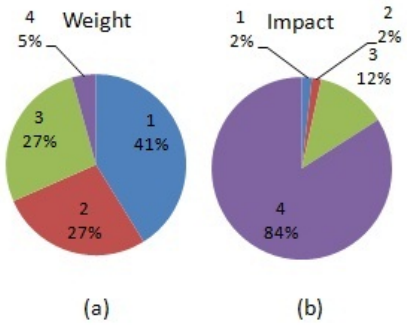
We present a case-study of transactional system, where a data center hosts the IT system of an on-line retail system. The monitoring log of a transactional system contains information about various workload and performance properties. We show how workload characterization service can be used to mine these logs and derive meaningful insights and actionable recommendations for performance management of the system. These insights can thus contribute to the higher-level objective of *Stability As A Service*. During on-line shopping clients perform various operations such as browsing, comparison of items, shopping, redeeming of vouchers, etc. Each request received by the data center is associated with various attributes such as client IP address, Host name, date and time of request, URL name, etc. The requested URL can be further split to obtain derived attributes. For instance a URL *http://abc.com/retail/AddToCart.jsp* can be split to extract *http://abc.com*, *retail*, *AddToCart* and *jsp*. Similarly date and time of the request can be split to derive more attributes such as Day of the week, Date of the Month, Month of the year, etc. Each request is associated with a performance measure of response time.

Figure 1 presents the multimodal analysis of the transactions log. The response time values of transactions belong to four different modes. The weight and impact of these modes are shown in Figure 1(a) and Figure 1(b) respectively. As can be seen, there exists a small set (5%) of requests that have very high response time (3653ms), and this set forms the highest impacting set. The next high impact set consists of 27% requests and mean response time of 86.36ms. Figure 1(c) further characterizes these modes by identifying dominant properties of the requests falling in each mode. The mode with highest response time (3635ms) contains 67% requests with $ResourceURL = Service3$. The mode with second highest response time (86.36ms) contains 48% requests with $ResourceURL = Service2$. Figure 1(c) presents more such insights to further characterize modes based on Resource URL, Resource group, and User ID. In this case study, improving Service 3 would have largest impact on decreasing the average latency of the system. Furthermore, requests with User ID=2 and Resource group = Type 2 are indicative of performance bottlenecks and demand further investigations.

Figure 2 (g) presents the discovered interesting subsets in the transactions log with respect to response time. Interesting subsets are discovered with respect to high and low response time as shown in Figure 2(g). Results show the descriptor and the statistics for each set. These are also shown in the form of plots in Figure 2 (a,b,c,d,e,f). The results show (1) percentage of requests belonging to the set, (2) average response time of the requests in the set. (3) average response time of the requests in the complement of the set, and (4) the probability of statistical similarity, p value, between the set and its complement. The algorithm identifies that the requests (0.27% of total requests) with $ResourceURL = Service5$ have significantly high response time (1070.69ms) than the rest of the requests (179.61ms). Other interesting sets with high response time are $ResourceURL = Service4$, $ResourceURL = Service3$. The algorithm also identifies sets described by multiple dimensions such as requests with $Createdby = CID1\ and\ UserId = UserId3$. Similarly, requests that perform significantly better than the rest of requests are also identified in Figure 2. For instance, interesting subsets of requests with low response time are $ResourceURL = Service6$, and $Resourcegroup = Type1$.

The results of multi modal analysis characterize the entire spectrum of the values of response time by grouping the response time values into different modes. ISD, on the other hand, focuses only on the two ends of the spectrum by identifying subsets with significantly high and low response times. The two approaches thus have some common results such as $ResourceURL = Service3$ and $Resourcegroup = Type2$ are identified as high response time requests. Similarly, requests with $Resourcegroup = Type1$ are identified as requests with low response time. However, the two results also complement each other. For instance, MMA shows that the entire range of response time values when grouped in four modes, can be explained by $ResourceURL = Service1$ (low values), $ResourceURL = Service2$ (intermediate values), and $ResourceURL = Service3$ (high values). ISD complements this results by further exploring the two ends. It identifies additional descriptors that explain very high and very low response times. For instance, $ResourceURL = Service5$, $Createdby = CID1 and UserId = UserId3$ show significantly high response time, and $ResourceURL = Service6$, $ResourceURL = Service7$ show significantly low response

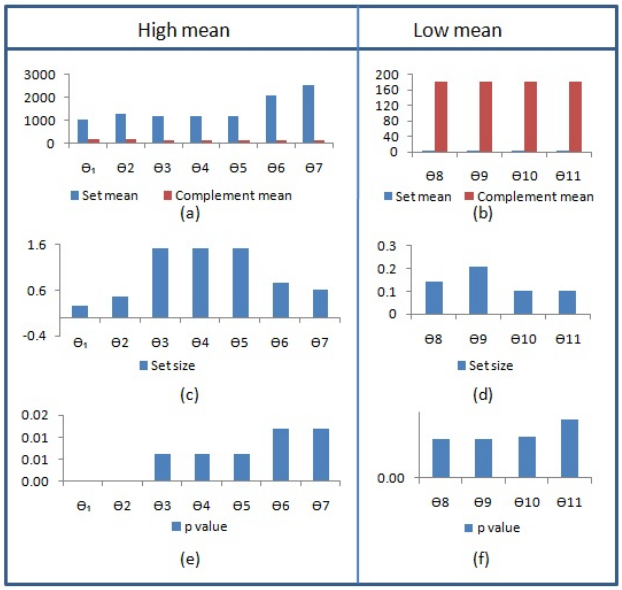
 ISBN: 978-1-61208-215-8



Mode	Weight	Mean	Impact	Resource URL	Resource group	UserID
1	41.14	8.16	335.72	Service1 (21%)	Type 1 (87%)	UserId 1 (21%)
2	27.29	11.37	310.29	Service1 (27%)	Type 1 (68%)	UserId 1 (18%)
3	27.25	86.35	2353.44	Service 2 (48%)	Type 1(59%)	UserId 1 (33%)
4	4.31	3653.38	15757.14	Service 3 (67%)	Type 2 (67%)	UserId 2 (36%)

(c)

Fig. 1. Multi-modal analysis of transactions log. (a) Modes by weight, (b) Modes by impact, (c) Mode characteristics.



Low/High	Set descriptor	Set descriptor	Set size	Set mean	Complement mean	p value
High	Θ_1	resource_url:Service5	0.27	1070.69	179.61	6.2E-18
High	Θ_2	resource_url:Service4	0.47	1309.35	176.72	1E-15
High	Θ_3	resource_group:Type2	1.52	1218.51	166.02	0.00621
High	Θ_4	resource_url:Service3	1.52	1218.51	166.02	0.00621
High	Θ_5	resource_group:Type2, resource_url:Service3	1.52	1218.51	166.02	0.00621
High	Θ_6	created_by:CID1, usrid:UserId3	0.76	2094.2	167.38	0.01182
High	Θ_7	from_ip:IP1	0.61	2573.63	167.43	0.01192
Low	Θ_8	resource_url:Service6	0.14	2.44	182.3	1.8E-47
Low	Θ_9	resource_url:Service7	0.21	2.46	182.42	1.8E-47
Low	Θ_{10}	resource_url:Service8	0.1	2.42	182.24	1.9E-47
Low	Θ_{11}	resource_url:Service9	0.1	2.75	182.23	2.7E-47

(g)

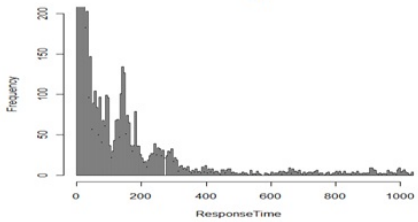
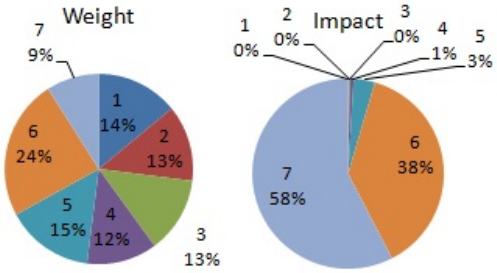


Fig. 2. ISD analysis of transactions log. (a,c,e) Sets with high mean: (a) Set mean and Universe mean, (c) Set size, (e) p value (b,d,f) Sets with low mean: (a) Set mean and Universe mean, (c) Set size, (e) p value (g) Tabular form of ISD analysis result, (h) Histogram of response time of requests.

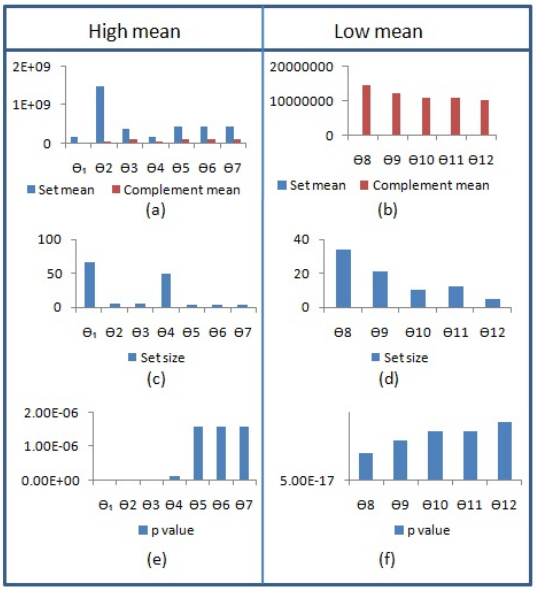


Mode	Weight	Mean	Impact	Product	Location
1	14	0.1	1.449996	ITM (27%)	HK (29%)
2	13	163.58	2126.476	BKT (23%)	HK (27%)
3	13	816.73	10617.46	EXC (14%)	ERP (26%)
4	12	2250.64	27007.7	CR (13%)	HK (33%)
5	15	10203.46	153051.9	ITM (18%)	LN (24%)
6	24	66400.78	1593619	ORD (22%)	HK (29%)
7	9	270129.8	2431168	ORD (35%)	ERP (32%)

(c)

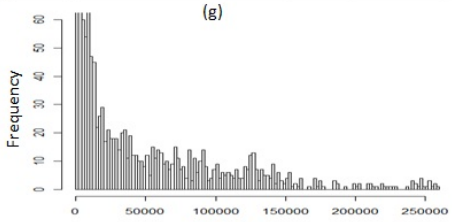
Fig. 3. Multi-modal analysis of workload log. (a) Modes by weight, (b) Modes by impact, (c) Mode characteristics.

 ISBN: 978-1-61208-215-8



Low/High	Set descriptor	Set descriptor	Set size	Set mean	complement mean	p value
High	Θ_1	Type:CS	65.83	147889394	127182.02	5.6E-17
High	Θ_2	Product:VE	5.2	1476871140	21719480.24	5E-13
High	Θ_3	Product:VT	5.2	350690403	83501855.02	8.6E-09
High	Θ_4	Application:GEMS	48.48	162351695	36271953.71	1.2E-07
High	Θ_5	Application:EMS	3.84	425826668	84284201.37	1.6E-06
High	Θ_6	Instance:EMS	3.84	425826668	84284201.37	1.6E-06
High	Θ_7	Application:EMS, Instance:EMS	3.84	425826668	84284201.37	1.6E-06
Low	Θ_8	Instance:SHARED	34.17	127182.02	147889394	5.6E-17
Low	Θ_9	Application:App1	20.7	38665.78	122808529.7	5.9E-17
Low	Θ_{10}	Product:ITM	10.44	3002.85	108746301.8	6.1E-17
Low	Θ_{11}	Location:NY	12.25	27240.15	110992182.6	6.1E-17
Low	Θ_{12}	Product:BKT	4.43	227.76	101915222.5	6.3E-17

(g)



(h)

Fig. 4. ISD analysis of transactions log. (a,c,e) Sets with high mean: (a) Set mean and Universe mean, (c) Set size, (e) p value (b,d,f) Sets with low mean: (a) Set mean and Universe mean, (c) Set size, (e) p value, (g) Tabular form of ISD analysis result, (h) Histogram of workload.

time.

These results can provide powerful levers to contribute to *Stability As A Service* as follows:

1) The analysis can provide opportunities for optimization and improvement by identifying requests that can have highest impact on the overall performance. For instance, improvement of Service 3 would result in highest impact on the overall improvement.
2) It identifies performance bottlenecks. For instance, requests with User ID=2 and Resource group = Type 2 are identified as performance bottlenecks.
3) It allows estimation of the contribution of individual request types and the impact of their improvement on the overall system performance stability.
4) Analysis on different log dimensions enables localization of the performance problems to specific URLs, Client IPs, location, time, among others.

B. Analysis of application workload

We next present the analysis of workload observed at the application tier of a loan-processing application over a period of one month. The application monitoring logs consist of per-day workload observed at the application tier along with various details of the requests such as application type, location, the requested service, etc. The dimensions in this log can be used to analyze the workload properties of the system.

Figure 3 presents the multi-modality analysis of the workload log on the basis of amount of workload. The observed workload values belong to 7 different modes. The smallest amount of workload is observed in mode 1 with average workload of 0.1 requests per day. This workload type is dominated by Product = ITM (27%), Location = HK (29%). Mode 7 represents heavy workload of 270,000 requests per day. This workload type is dominated by Product = ORD (35%), Location = ERP (32%).

Figure 4(g) presents the interesting subsets are discovered with respect to high and low workloads. Results show the descriptors and the statistics for each set. Workload with $Type = CS$ (65% of total records) is significantly higher (147889393.97) than the rest of the workload (127182.02). Other sets with high workload are described by $Type = CS$, $Product = VE$, $Application = GMS$, among others. It also identifies sets described by combinations of dimensions such as $Instance = NYD, and Location = NYD$. Similarly, sets with significantly low workload are also identified in Figure 4. For instance, $Instance = SHARED$, $Product = ITM$, $Location = HK$.

Comparing the ISD and MMA results, it can be seen that both results share some common findings. For instance, Both identified the sets $Product = ITM$, and $Location = HK$ to have significantly low workload. Some findings, on the other hand, complement each-other. For instance, MMA identifies $Product = ORD$ to describe the high workload sets. MMA thus identifies the most dominating high workload component. ISD, on the other hand, complements this result by providing other not-so-dominating high workload components such as $Product = VE$, $Product = VT$. These sets are not very large to describe an entire mode, but nevertheless have significantly higher workload.

These insights can be used for efficient resource allocation such that the resources to serve request types with low workload can be rationed while the request types with heavy workload can be supplied adequate resources. Better load-balancing and workload distribution policies can also be derived using

 ISBN: 978-1-61208-215-8

such analysis. In the given case-study, resources at location HK for serving requests with Product = ITM or BKT can be rationed while sufficient resources need to be provided at the location ERP to serve requests with Product = ORD. Also, in order to avoid location bottlenecks recommendations can be given to investigate the possibility of migrating the workload from Location ERP to other locations.

These insights can be used for capacity analysis such as efficient resource allocation, load balancing etc. For instance,

1) The resources to serve request types with low workload can be rationed while the request types with heavy workload can be supplied adequate resources. In the given case-study, resources at location HK for serving requests with Product = ITM or BKT can be rationed while sufficient resources need to be provided at the location ERP to serve requests with Product = ORD.
2) Better load-balancing and workload distribution policies can also be derived using such analysis. A workload distribution policy can be quickly worked out on the basis of modes such that resource requirements of low and high workload are met. In the given case-study workload of product VE can be moved to resources that service workload of product BKT.
3) In order to avoid location bottlenecks, recommendations can be given to investigate the possibility of migrating the workload from Location ERP to other locations.

VI. Conclusion

Stability As A Service is critical for data centers and workload characterization is one of the essential services within *Stability As A Service*. Different solutions have been proposed in the past for characterizing various system properties, however most of these approaches often require high levels of instrumentation and intrusiveness. This makes their use difficult in real-world production systems. In this paper, we argued that many interesting insights can be derived to characterize system workload simply by analyzing the transaction logs captured at different layers in the system such as access logs at application servers, SQL logs at database servers, among others. We presented two approaches that analyze various dimensions and measures of the transaction logs in order to characterize workload. We demonstrated the effectiveness of the proposed approach through real-world case-studies and showed how the findings of the two approaches complement each-other.

References

[1] M. Atzmueller and F. Puppe. Sd-map: a fast algorithm for exhaustive subgroup discovery. In *Proc. PKDD 2006*, volume 4213 of *LNAI*, pages 6 – 17. Springer-Verlag, 2006.

[2] M. Atzmuller. *Knowledge intensive subgroup mining: Techniques for automatic and interactive discovery*. Aka Akademische Verlagsgsellschaft, 2007.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth Intl., Belmont, CA, 1984.

[4] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[6] Y. Jin, E. Sharafuddin, and Z.-L. Zhang. Unveiling core network-wide communication patterns through application traffic activity graph decomposition. In *SIGMETRICS*, 2009.

[7] B. Kavšek, N. Lavrač, and V. Jovanoski. Apriori-sd: adapting association rule learning to subgroup discovery.

[8] N. Lavrač, B. Cestnik, D. Gemberger, and P. Flach. Subgroup discovery with cn2-sd. *Machine Learning*, 57:115 – 143, 2004.

[9] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with cn2-sd. *Journal of Machine Learning Research*, 5:153 – 188, 2004.

[10] M. Natu and G. Palshikar. Discovering interesting subsets using statistical analysis. In G. Das, N. Sarda, and P. K. Reddy, editors, *Proc. 14th Int. Conf. on Management of Data (COMAD2008)*, pages 60–70. Allied Publishers, 2008.

[11] A. Sharma, R. Bhagwan, M. Choudhury, L. Golubchik, R. Govindan, and G. M. Voelker. Automatic request characterization in internet services. In *1st HotMetrics Workshop, Association for Computing Machinery*, 2008.

[12] M.-H. Zhang and Q.-S. Cheng. Determine the number of components in a mixture model by the extended ks test. In *Journal Pattern Recognition Letters, Volume 25 Issue 2, 19 January 2004 , Elsevier Science Inc. New York, NY, USA*.

 ISBN: 978-1-61208-215-8