2015 IEEE 12th International Conference on Autonomic Computing

Ranking and Updating Beliefs based on User Feedback

Industrial Use Cases

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

***Abstract*—Incorporation of user feedback in enterprise management products can greatly enhance our understanding of modern technology challenges and amplify the ability for those products to home in to user environments. In this paper we present an entropy-based confidence determination approach to process user feedback data (direct or indirect) to automatically rank and update the beliefs of any recommender system. Several examples of application of this method are discussed in the context of VMware products. Moreover, an optimization algorithm is demonstrated for adaptive thresholding of monitoring flows based on user ratings of generated alerts effectiveness.**

***Keywords —Feedback computing, recommendation systems, belief ranking, entropy, adaptive thresholds*.**

I. Introduction

With emerging challenges in automated system management for the cloud computing age, there is an unprecedented motivation in new smart software solutions with data-agnostic instrumentation. Such an approach, in particular, is adopted by virtualization and hybrid cloud leader VMware within a series of products [1], as in vRealize Operations Manager (vR Ops), Log Insight, vSphere Dynamic Resource Scheduler (DRS). Those products and their features that are mostly delivered by means of data learning methods with minimum involvement of user or expert/administrator knowledge provide a high level universality in problem solutions at the customers independently of their environment. (In this paper the words user or customer refers to both the human user of the products and also the beneficiary of actions taken by the products. For example, a host can be the customer of DRS actions). These are indispensable technologies in modern large-scale and heterogeneous cloud systems. However, these technologies have inevitable drawbacks that can be overcome with employment of new algorithmic solutions based on the feedbacks that user or expert provides during exploitation of those intelligent software systems. Statistical processing of feedback data can enhance the original data-driven analytics by enabling a closed-loop self-tuning product. Deployed products or features that either take or recommend an action to the user for automatic management purposes (based on prior beliefs in best practices or learned from its ecosystem) or whose output invokes an action from the user fall into this category. User feedback can be obtained directly, via asking the user questions (can be as simple as a "like" button), or indirectly, by following user behavior. More importantly, incorporation of a closed-loop feedback mechanism can dramatically boost the enterprise solutions by tuning themselves to the customer's usage patterns.

In general, we can think of the feedback computing as an umbrella domain for vast variety of research and application areas ranging from modern recommender systems to systems design for peta-scale compute and storage environments [2] and clouds [3]. In all cases, the feedback incorporation aims at maximizing systems throughputs. In particular, [2] discusses a feedback computing framework that can alleviate system bottlenecks for better utilization of system resources, and [3] elaborates an optimal resource allocation scheme for IaaS. While recommender systems represent a well-developed computer science area, the feedback compute in system design issues remains fragmentary and lacking a systematic treatment. To the best of our knowledge, there is no simple and unified data science framework for processing of feedback information in improvement of the above mentioned cognitive technologies that concern system and product design. Therefore, an idea arises to interpret the feedback compute in terms of validation of systems main cognitive convictions/beliefs that they operate on, employing a formalism of statistical inference with the concepts of confidence, information uncertainty, and ranking.

A data-agnostic analysis of an environment results in certain "beliefs". Beliefs can be computed or gained through prior knowledge of certain systems (e.g. when to move a virtual machine (VM) to another host within a virtualized infrastructure). These beliefs are applied as tools for systems management, in particular, in recommending the user (or many distributed users) to take actions or indicating certain status of the system such as anomalous states. Sometimes those beliefs, or equivalently, the recommendations generated based on those beliefs can be evaluated by the user or expert feedback. Moreover, the assessment feedback by the same user about the same belief can vary from one time instant to another based on many factors, such as evolution of the underlying environment, re-comprehension of the belief's influence over time, and situation-based specifics. Collecting this feedback and comprehending it by means of a second layer data-agnostic processing, we are able to come up with more reliable recommendations to the user by modifying the original beliefs. In case of multiple users of the system beliefs, those updated recommendations need to be generalized, a trade-off for all parties satisfaction. This means that those products and their features will be adequately self-tuned at the user environment in terms of smarter belief retrieval. The term "recommendation systems/engines" applied here primarily covers any recommendation generation apparatus in the sense above. This means that our research can sit on top of any recommender system.

II. Entropy-Based Confidence

Assume we have observed some beliefs from data representing the system at the user. In a data-agnostic

978-1-4673-6971-8/15 $31.00 © 2015 IEEE
DOI 10.1109/ICAC.2015.29

management approach of systems we apply those beliefs directly without measuring user perception of (confidence on) effectiveness of our recommendations based on those beliefs. If there is an opportunity to ask the user for some feedback about the observed beliefs and evaluate its statistical nature, then we are able to manage the system more effectively. Let $B_1, B_2, \ldots, B_n$ be the initial beliefs (with their data-agnostic maximum confidences equal to 1), and let $F(B_i) \equiv f_1(B_i), f_2(B_i), \ldots, f_K(B_i)$ be the feedback statistics (collected into a uniform array from possibly many distributed users of the system instances over time) on belief B_i taking value from $[0,1]$. We assume that any indirect feedback that can be tracked is also convertible to the same interval. We also assume that the feedback statistics for different beliefs are independent of each other. An extension of this model needs to account for correlated feedback statistics regarding different beliefs. By processing $F(B_i)$ we want to come up with a confidence $C(B_i)$, $i = \overline{1,n}$, of B_i that tells us the degree of the belief "strength" as a managing utility. This allows introducing an order among those beliefs in terms of their confidences converted to ranks $R(B_i)$. The ranked list of beliefs then can be applied to adjustment of basic data-agnostic usage of the beliefs.

Let the user be asked to answer a question if he/she is satisfied (and to what degree) by the recommendation related to a belief B_i. And let the user provide a feedback taking value from $[0,1]$ (or quantized to l degrees within that interval) at each time t_k when facing the belief B_i. So the feedback for B_i is a series of statistics:

$$F(B_i) \equiv \{f(t_k, B_i)\}_{k=1}^{K} \equiv \{f_k(B_i)\}_{k=1}^{K}$$

where 1 is full satisfaction and 0 is complete dissatisfaction. The "like"/"dislike" feedback option is the most extreme case in this model. Based on $F(B_i)$ we want to make a convergence evaluation in user opinion and output a confidence $C(B_i)$ which can be incorporated into the existing analysis to tune its performance. This confidence will support the degree of validity of the initial belief B_i. Our main postulates are: 1) the posting of feedback statistics is assumed to be a process with increasing degree of importance with respect to time; 2) if there is no convergence in user feedback statistics then we have to keep our basic belief system without update; 3) if there is a convergence to some degree of user feedback then the basic beliefs are updated according to confidences obtained. Therefore, we want to have a map that at each time instant t_r returns a weighted statistics of the past feedback series:

$$S(f_k(B_i)) = \sum_{r=1}^{k} w(t_r) f_r(B_i) / \sum_{r=1}^{k} w(t_r) \quad (1)$$

where the weight function $w(t)$ can be, for instance, of linear or exponential form. In exponential case it can be determined by

$$w(t_k) = 1 \text{ and } w(t_r) = e^{-(t_k - t_r)} \text{ for } r < k. \quad (2)$$

Let

$$\bar{S}(B_i) \equiv \{S(f_1(B_i)), \ldots, S(f_K(B_i))\} \quad (3)$$

be the obtained data (taking values on $[0,1]$) after (1) and (2). Now we estimate the uncertainty in the data (3) in terms of its distribution/histogram on the l intervals from $[0,1]$, applying the entropy measure:

$$H(\bar{S}(B_i)) = -\sum_{r=1}^{l} h_r \log_l h_r, \ \sum_{r=1}^{l} h_r = 1 \quad (4)$$

where h_r is the normalized frequency of values $S(f_k(B_i))$ within one of the mentioned intervals r, $r = \overline{1,l}$. Note that $0 \le H(\bar{S}(B_i)) \le 1$. Then the confidence in belief B_i can be measured by the characterized uncertainty:

$$C(B_i) = 1 - H(\bar{S}(B_i)) \quad (5)$$

where we set

$$C(B_i) = 0, \quad \text{if } H(\bar{S}(B_i)) > \bar{H} \quad (6)$$

with a threshold value $\bar{H}$. This is the case of high uncertainty in feedback data (i.e. there is no convergence in user opinion) and the corresponding confidence should be the minimum.

Now let $m(h_{\max})$ be the average of the values $S(f_k(B_i))$ from the mode $h_{\max}$: $h_{\max} = \max\{h_1, h_2, \ldots, h_l\}$. This will take into account the degree of importance in time for the values that fall into the modal column in the histogram. If entropy is within $\bar{H}$, we determine the needed confidence $C(B_i)$ by identifying the interval of the bias in uncertainty, hence, by the mode $h_{\max}$. Now we can rank the equal-weighted original beliefs by their confidences. Alternatively, based on the computed confidences, new beliefs can be designed as combination of the original and respective feedback-based beliefs within specific applications.

The rank for B_i is computed by

$$R(B_i) = \alpha + (1 - \alpha) m(h_{\max}) \quad (7)$$

where $\alpha = 1$, if $C(B_i) = 0$; $\alpha = C(B_i)$, if $C(B_i) > 0$. Thus, given (7) we have a list of beliefs ranked according to their feedback-based confidences. Further and specific usage of $C(B_i)$ depends on the application as shown in Section 4.

III. Application Use Cases

Several domains of potential applications of the framework of Section 2 are discussed below on VMware's cloud management solutions.

Root cause detection (vR Ops). Problem root cause determination is a crucial element in infrastructure management products. The recommendations for root cause localization with anomaly impact ranking are generated purely based on beliefs (say correlations of anomalies) learned from the infrastructure [4]. With the introduced feedback compute framework on user satisfaction this domain gains an additional capability for re-ranking of the initial beliefs to produce more effective recommendations.

Dimension reduction (vR Ops). Monitoring and surveillance of modern IT infrastructures is a complex task. Currently vR Ops has a strategy to apply dimensionality reduction methods, such as "principal feature analysis", to

substantially lower the complexity of computations for measured data center processes (metrics). This reduction can be pushed further with user feedback ratings against impacts of different metrics or group of metrics on the whole system. To state simply, it will result in elimination of low weight metrics from the product's analytics engine thus mitigating the high computational burden for anomaly detection through dynamic thresholds [5].

False positive alarm reduction (vR Ops). Generation of alarms and alerts of different severity is the cornerstone recommendation instrument that vR Ops applies. Sometimes the level of false positive alarms is high at specific user environments. Moreover, the alerts generated by vR Ops can indicate not only system performance issues but also change indications that may not be of interest to the user. On the way of reducing this kind of noise for the users, the feedback compute can be of immense help. The corresponding alarm and alert recommendations ranked appropriately will converge to an optimal product usage pattern at a particular IT ecosystem.

Log analysis (Log Insight**)**. An application of feedback compute might be related to anomaly detection in logs by fundamental structure extraction in terms of Dynamic Normalcy Graphs [6]. In particular, we are able to enhance the efficiency of the DNG as a causation tool via processing of user feedback statistics on correlation breakage alarms. That can be performed if we evaluate the confidences of each correlation (belief) in DNG from that statistics and apply it in computation of abnormality degree of data stream. Another problem in log analysis is the optimized task execution by users. In particular, facing an error type event the user executes some tasks to remedy the system. This information on relations of various error events and tasks can optimize the execution of those tasks now ranked appropriately as the best priority recommendations.

Automatic execution of VM migration (DRS). If the cluster is fully automated, vCenter places VM's that join the cluster on appropriate hosts and migrates running VM's between hosts, as needed, to ensure the best possible use of cluster resources. The migration threshold varying from Conservative to Aggressive controls this automatic move based on some best practices or beliefs. Tracking the hosts' behavior (which is the end recipient of the actions) in terms of their responses in migration we are then able to rank those prior beliefs and make the migration threshold dynamic, suited to the deployment environment. One such response maybe the rate of vMotion actions performed per host which can be an indication of incorrect prior belief about the suitability of the host. Another source of feedback information might be employed to revise the DRS affinity rules. Namely, the statistics from performance issues at the cluster associated with certain VMs will allow ranking the prior affinity rules for VMs. In that way, the original affinity rules would be effective in trade-off with performance of the infrastructure, thus enabling dynamic rules.

Social-media platform for virtualization management. This is an application for the proposed feedback mechanism related to the work [7] on organizing a virtual environment into a social network of its own. This kind of social interaction between humans, hosts, VM's, and servers, is a unique repository of feedback information that can be turned into valuable knowledge improving the social network tools and efficiency of such platforms. Consider the canonical design in [7], where an administrator follows the vCenter server, the latter in turn follows hosts, and the hosts follow VMs. Processing the like/dislike interaction within this hierarchy by our method we can highlight the most unbalanced social links in the network by ranking them in terms of confidences among users. That will imply recommendations towards improvement of social health with indication of relevant network sectors to better virtualization management.

IV. Adjustment of Thresholds

Hard thresholding is an effective monitoring and anomaly detection tool in management solutions [8] along with time-dependent dynamic thresholding [5],[9]. We apply the principles of Section 2 to develop a method for self-modifying hard thresholds (HT). The primary goal is to make the common expert knowledge on different processes in terms of these time-independent thresholds adaptively optimized to the user environment. Processing utility ratings of alerts that user inserts into the rating system upon violation of a manually set HT (belief), our method automatically leads to the optimal HT of the monitoring flow as the best fit for the users' experience and application. This allows generating alerts with maximum accuracy, or, alternatively, optimizing the trade-off between false positive and negative alerts. Collecting user ratings for an HT (denoted by D in upper case) alerts via appropriately asking questions about their effectiveness, we aim at adjusting the threshold. However, this adjustment is controlled by a user-defined tolerance to noise degree N (false positive level from $[0,1]$). Our prototype UI within vR Ops allows the distributed users to rate (1 to 5 stars) active alerts as how indicative they are (or perceived to be). The ratings are converted to quantitative options as *non-indicative* (0), *somewhat* (0.25), *rather* (0.5), *highly* (0.75), *perfectly* (1). As soon as sufficient statistic defined by both volume of ratings and users participation in feedback generation is available, we execute the algorithm below to compute the compromise HT based on relevant confidence C in user opinion. According to this system the weighted ratings fall into one of four buckets: the first one is $[0,0.25]$ and the last one is $[0.75,1]$. The basic assumption behind this adaptive thresholding is that the indicative power of alerts increases with their constituent data point distances from related HT's.

Algorithm (upper HT). If $C > 0$, calculate $m(h_{max})$. If $|1 - m(h_{max}) - N| \leq \delta$, no need to update the threshold, where δ is a closeness, interpreting $1 - m(h_{max})$ as the actual "error term" or noise degree. Two possible cases arise:

I. If $1 - m(h_{max}) - N < -\delta$, then this means that the acceptable noise level is low and we can add more data points under the threshold line by *moving the threshold down* to capture more issues. Since in this case we have no feedback below the HT to work with, we need an extrapolation of existing statistics above the HT with the worst case scenario, and thus optimizing the noise within one rating interval (up to 25%). The steps of the procedure are:

a. Calculate the average feedback count $n(f)$ per alert;

b. Sequentially move down the HT to its nearest data neighbor to hypothetically involve more alerts and estimate the noise expectation:

b1. Form new alerts (if any) with hypothetical feedbacks equal to the winning histogram bucket minimum (denote it by $V(r_{min})$). Take this minimum rating (for a particular new alert) $n(f)$ times;

b2. Add the hypothetical statistics from b1 to the existing feedback statistics;

c. In each iterative "nearest neighbor" move down, calculate the hypothetical error term of the statistics from b2. If the latter is close to N within δ-accuracy, the iteration stops. Otherwise it stops at hypothetical HT with closest to N error term smaller than N;

d. The procedure stops with the hypothetical HT D_{opt}. Then the adjustment of HT is performed by the formula:

$$D = (1 - C)D + CD_{opt};$$

Further optimization, if N is still far away of the actual error term, is undertaken after the next round of rating collection. Moreover, if the optimization is not possible within one bucket, the following partial modification of the above mentioned procedure is foreseen only for steps c and d, the rest of computations are identical:

c. If $m(h_{max}) - V(r_{min}) \leq \delta$, then the minimum rating assignment in b1 is modified to take the minimum rating of the left neighbor interval of the winning bucket.

d. In each iterative "nearest neighbor" move down, calculate the hypothetical confidence for the statistics of new and rated alerts. If it is not positive, jump to the next nearest neighbor point, otherwise, calculate the error term. If the latter is close to N within δ-accuracy, the iteration stops. Otherwise it stops at hypothetical HT with closest to and smaller than N error term.

II. If $1 - m(h_{max}) - N > \delta$, then it means that the noise level is higher than it is tolerable and the HT *needs to be increased*. In this case, a natural optimization task arises. Specifically, the hypothetical jump in HT should be as much as is possible to minimize the difference between N and the resulting error term while processing the feedback statistics. The steps of the procedure are:

a. Hypothetically moving D up to its nearest neighbor data point, calculate the corresponding confidence and $m(h_{max})$ based on that line (with the same alert generation rule or wait cycle value) and rest of alert ratings. If there is no convergence (confidence=0), jump to the next nearest data point above. Stop the procedure when $1 - m(h_{max}) - N < \delta$ or $1 - m(h_{max}) - N$ is the minimum achievable positive difference by the iteration.

b. If in moving up we lose enough statistics (in terms of confidence calculation) for further iterations, we stop the procedure;

c. Let the procedure stop at a HT value D_{opt}. Set

$$D = (1 - C)D + CD_{opt}.$$

So the general concept behind the HT increase is that of constructing the distribution of error terms of hypothetical beliefs (as HT's) and choosing the one with an error term closest to tolerable noise level. Lower HT case is symmetric.

Fig. 1 illustrates a customer data of 18 months with initially low HT (82% for CPU load) and noise generating alerts (because of change in data behavior). Negatively rated alerts by the user cause the algorithm to learn much higher and indicative position (94%). The confidence in user opinion for this experiment was 0.86, with actual noise 0.77 before the HT adjustment. Fig. 2 stands for a pictorial projection of user ratings on the corresponding HT alerts onto the time series data in another adjustment scenario.

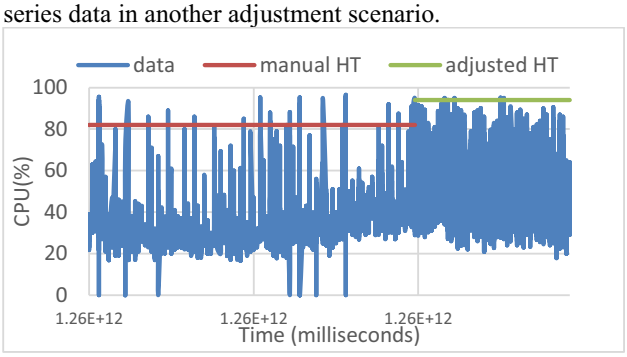

Fig. 1. Data with manual HT and its adjustment.

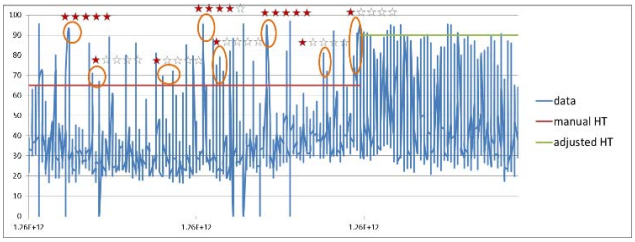

Fig. 2. User ratings of alerts.

REFERENCES

[1] VMware products. http://www.vmware.com/products.

[2] R. Gunasekaran and Y. Kim, "Feedback computing in leadership compute systems," *Proc. 9th Int. Workshop on Feedback Computing*, Philadelphia, US, June 17, 2014.

[3] R. Srikant, "Resource allocation and networking in clouds and data centers," *Proc. 9th Int. Workshop on Feedback Computing*, Philadelphia, US, June 17, 2014.

[4] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, "An anomaly event correlation engine: Identifying root causes, bottlenecks, and black swans in IT environments," *VMware T. J.* 2(1), pp. 35-45, 2013.

[5] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, "An enterprise dynamic thresholding system," *Proc. USENIX 11th ICAC*, June 18-20, Philadelhia, US, pp. 129-135, 2014.

[6] A.N. Harutyunyan, A.V. Poghosyan, N.M. Grigoryan, and M.A. Marvasti, "Abnormality analysis of streamed log data,"*Proc. IFIP/IEEE NOMS*, May 5-9, Krakow, Poland, pp. 1-7, 2014.

[7] R. Soundararajan, E. Celebi, L Spracklen, H. Muppalla, and V. Makhija, "A social-media approach to virtualization management", *VMware T. J.* 1(2), pp. 59-68, 2012.

[8] Y. Wang and Y. Mei, "Online parallel monitoring via hard-thresholding post-change estimation,"*Proc. IEEE Int. Symp. Inform. Theory*, Honolulu, HI, USA, June 29–July 4, pp. 3190-3195, 2014.

[9] D. Breitgand, M. Goldstein, E. Henis, and O. Shehory, "Efficient control of false negatives and false positive errors with separate adaptive thresholds," *IEEE Trans. Net. & Serv. Manage.* 8(2), pp. 128-140, June 2011.