Tale of Tails: Anomaly Avoidance in Data Centers

***Abstract*—Anomaly detection in cloud data centers where physical boxes host multiple virtual machines (VMs) often results in issuing performance tickets. Such tickets are routinely issued when CPU and/or RAM usage exceed predefined thresholds. Using resource usage data from production data centers that consist of more than 6K physical machines hosting more than 80VMs, we identify stochastic properties of resource usage both at the physical box and at the VM levels that illustrate how anomaly instances that result in ticket issuing are triggered. Focusing on the tail usages of CPU and RAM and their properties, we develop an algorithm for VM cloning that proactively manages to dramatically reduce anomaly instances using tail usage information. Evaluation results show that the proposed VM cloning mechanism dramatically reduces the number of CPU anomaly instances by 60%, while art-of-the-practice VM migration mechanism only achieves around 40% reduction of CPU anomaly instances. Meanwhile, the duration of CPU anomaly instances is strictly controlled below 2 time windows with the help of reactive balancing between original and cloned VMs, considering some of boxes originally experience CPU usage violation continously more than 25 time windows. Moreover, such an impressive anomaly reduction is accomplished at little cost in both terms of training (tail usage prediction via mean usage) and storage (recent one day's observation only) for the historical usage series.**

I. Introduction

Ticket issuing is widely used in today's data centers [**?**], [**?**] for performance anomaly detection. Performance tickets are issued either automatically by the system (e.g., when high resource usage is detected and signals potential deteriorated user experience) or by the users themselves (e.g., after the system is perceived slow or unresponsive). Ticket resolution is an expensive process as it usually requires manual labor for root-cause analysis [**?**]. Transient resource usage that grows beyond a predefined threshold is considered an anomaly as it threatens user performance at tails and signals unsteady system operation. Such usage fluctuations are the result of aggressive multiplexing of multiple VMs across physical servers competing for limited physical resources and the dynamic nature of each VM workload [**?**].

Fig. 1: How to determine a usage anomaly instance.

We analyze data center usage times series of a major vendor. The trace data correspond to 6K physical boxes serving more than 80K VMs over a time period of a week. Our analysis shows that anomalies instances fall into two categories: single anomaly instance (AI) where the duration of the anomaly is short and continuous anomaly instance where the duration of the anomaly is long. Figure 1 gives an overview of usage anomaly instances and how they affect the numbers of issued tickets. Resource usage is typically reported within discrete time windows (e.g., in our trace each time window equals to 15 min). While the usage series has fluctuations across time, it goes beyond the usage threshold three times, twice where it exceeds the usage threshold for a short window only (single AI), and once where it exceeds the usage threshold for a long window, a window that is a concatenation of several discreet time windows (continuous AI). Such continuous AI results in multiple consecutive tickets. In addition to the AI duration, the trace characterization points to one more important factor that distinguishes single and continuous anomaly instances: we find that not only the *duration* but also the *magnitude* of a continuous AI is much larger than the single one (see Figure **??**(c)). While the single AI may be considered relatively harmless, continuous AIs have the potential to significantly undermine the user perceived performance.

In this paper we develop a methodology for reducing CPU tail usages in data centers via *proactive* VM cloning across different physical servers aiming to eliminate or drastically reduce continuous anomaly instances in physical servers. To this end, we first do a detailed, post-hoc workload characterization study of usage time series (for both CPU and RAM) in production data centers. This characterization analysis allows us to view the statistical characteristics of usage time series and focus on the properties of their tails. The key findings of this characterization study are as follows: 1) the culprit of continuous CPU AIs is VM consolidation. 2) CPU tail usage is highly correlated to the mean CPU usage of physical servers, while and the CPU tail usage of physical servers follows a normal distribution. 3) RAM anomalies do not relate strongly to the VM consolidation level in contrast to CPU AIs. Based on these observations, we propose a three-step algorithm to reduce/remove the CPU tail usages for physical servers. First, we develop a CPU tail usage prediction method which captures both the steady state of tail distribution and the overall CPU usage level in the tenant. As a second step, based on the tail predictions of physical servers, we design a proactive VM cloning algorithm, which heuristically clones VMs and distributes the workloads across different physical servers.

The proposed VM cloning algorithm is evaluated in detail using trace driven simulation. Results are summarized as follows: 1) the proposed prediction method of tail usages is much cheaper computationally compared to accurate but expensive time-series predictions (e.g., neural networks), while balancing tail usages across boxes; 2) VM cloning achieves a ten percent higher reduction in CPU usage violations than classic load balancing with VM migration. It is also noteworthy that our method achieves CPU tail usage reduction with minimal RAM usage violation (up to 3 per cent), while migration increases RAM usage violations by 60 percent. After load balancing tail usages, not only the number of CPU AIs reduces by 60%, but also the duration of continous CPU AIs ialso dramatically drops, with mean duration no more than two time windows (30 minutes). Note that if no cloning is used, continuous AI durations can reach up to 6 hours. **Comparisons with migration? or with raw data?** (JX: raw data)

The outline of this work is as follows: **TBD** Section **??** presents related work, followed by the summary and conclusions in Section **??**.

II. Evaluation

In this section, we evaluate the proposed methodology in mitigating performance anomalies and tail usage violation for around eighty datacenter tenants, running more than 1000 boxes and 10,000 VMs. We focus on the following metrics of interests related to the datacenter dependability in terms of CPU: (i) tail target violation for boxes, (ii) anomaly instances reduction, and (iii) anomaly duration reduction. Moreover, we also present the prediction accuracy achieved by the proposed tail prediction, in comparisons with neural network based time series prediction [**?**]. Particularly, we learn form the past $W = 1$ day, predict the usages of the following day, and proactively actuate the VM cloning strategy. We develop a simulator to evaluate the proposed methodology for 4 consecutive days.

In the following, we first sketch the simulator design and present the effectiveness of the proposed methodology in reducing (continuous) AI for tenants considered. We then highlight how the proposed tail prediction scheme and VM cloning outperform alternative approaches, i.e., time series prediction and VM migration.

A. Experiment

Simulator We develop a trace-driven simulator to emulate the system dynamics and evaluate the metrics of interests. The input data of the simulator are CPU and RAM usages from VMs and boxes hosted at IBM production datacenters. We determine if anomalies occur in the granularity of 15-minute windows, which is the granularity provided in the original trace. The simulator computes the box CPU/RAM usages as the sum of its hosted VM CPU/RAM usage plus the original box-only usage, which is independent of VM activities. We generate the VM CPU usage according to the normal distribution with mean being the product of original CPU usages and split weights, in the case of VM cloning. As for the VM RAM usages, we directly use the values from the trace. Note that the simulated usages are not identical to the original trace, due to the emulated feedback from the VM cloning strategies.

Note that we resort to the last value prediction, for all statistics related to RAM usages, due to the stable behavior of RAM, for both boxes as well as VMs. For the mean box CPU usage, we also use the last value prediction.

Targets Here, we consider the targets for box CPU and RAM usages are 60%, and 90%. The reason for a higher RAM target is due to LC *Ji. please provide a reason*□. The specific tail evaluated here are 90^{th}, 95^{th}, and 98^{th} percentiles. When imposing the target value on the 95^th, we allow roughly five out nighty-six 15-minutes windows in a day that has CPU and RAM usage exceeding the target values of 60 and 90, so called anomaly instances (AI) When there are more five occurrences of such AI for a box in a day, we consider this as a case of tail target violation. In the rest of this section, we focus on presenting the average reduction of tail target violation (RTV) per tenant as well as the reduction of AI (RAI), particularly on the CPU (RTV_c, and RAI_c). The average RTV and RAI per tenant are in the range of [01].

B. Big Picture

LC *Ji, we might want to change the fig to RTV here*□ We first present an overview on the average reduction of tail target violation (RAI_c) for all eighty tenants' box CPU, in Fig **??**. The tail percentile considered here is 95. Each point there represents a tenant, whose RAI_c is shown in the y-axes, CPU availability (i.e., CPU utilization) is in x-axes, and number of boxes is represented by the the size of bubble. A lower value of x-axes shows the higher spare resource availability, and the bigger bubble implies that the tenant has a higher number of boxes. There are two visible trends in how effectiveness fo our proposed methodology in reducing AI.s One can see that we tend to achieve a higher reduction for bigger tenants and for tenants that have higher spare capacity, due to higher degree of freedom in redistributing the box.

C. Effectiveness of TRA Prediction

Here, we present how the proposed tail prediction scheme can accurately capture the box CPU tail dy-

Fig. 2: Bubble plot of CPU anomaly instance reduction for all tested tenants via VM cloning with tail target equal to $95\%ile$. Here the size of bubbles is the number of boxes in the tenant (range from 2 to 80), x-axis represents the CPU utilization in the tenant, and y-axis represents the reduction in number of CPU AI compared with the original.

namics and effectively reduce the tail target violation when integrating with VM cloning. We compare three variates of the proposed tail estimating schemes with the neural network based time series prediction(NN) [**?**] that can accurately forecast the entire trajectories of usages. LC *Ji. please correct*In order to build NN prediction models, we first need to use historical data, i.e., past three days, which is sufficiently long to capture the time dependency within the series. The three articular tail estimation schemes are based on i.e., $L = \overline{T} + \alpha S_t$, and with different values of α, i.e.,

- $\alpha = 0$, for all boxes, neutral prediction (NP),
- $\alpha = 2$, for all boxes, conservative prediction (CP),
- α_i, for tenant i boxes, the proposed tenant-resource-aware prediction (TRA).

When setting $\alpha = 0$, NP uses the mean of classified distribution for box CPU tail, whereas CP conservatively use the tail of tail distribution[1] so as to overestimate the box tail.

Prediction Accuracy. Figure **??** (a) and (b) summarize the CDF of absolute and raw percentage of prediction errors for CPU box tail for all optimization periods. The two key finding are: (i) the proposed tail estimation scheme (of $\alpha = 0$) is almost as accurate as expensive NN approach, and (ii) the proposed TRA overestimates the CPU tail, but is still more conservative than CP.

The lowest prediction errors are achieved by the NN at the average value of 20%, which is the most computationally expensive and requires long historical data for training. Given how short the training data and low computational complexity, CP can achieve very similar absolute prediction errors as NN. When looking to the distribution of raw errors, CP has slightly higher errors than NN. We regard such a finding as a positive news, as CP tends to overestimate the box CPU tail shown by more than 60 % of errors are positive. In contrast, NN tends to underestimate the box tail, indicated by the more than 70% of errors are negative.

[1]For normally distribution, mean plus two standard deviation represents the 96^{th} percentile of the distribution.

Without any sunrise, TRA and CPU have significant higher absolute and raw errors, due to their conservativeness in estimation. In terms of raw errors, only 15% of prediction errors from TRA is negative. As the ultimate objective of the tail prediction is to enable the efficient resource management, conservative prediction tends to lead to resource provisioning, which is more desirable than the situation of under-provision, leading to the high risk of performance anomaly.

(a) Absolute PCT Error (%) (b) PCT Error (%)

Fig. 3: CDF of error of tail prediction using different methods.

Reduction of Tail Target Violation To see the impact of the tail prediction schemes, particularly the proposed TRA, we combine three tail prediction schemes with VM cloning strategies. In Fig. **??**, we summarize the average reduction of CPU tail violation (RTV_c) for each tenant, for different considerations of tail percentiles and prediction schemes. A particular bar in Fig. **??** presents the distribution of tenant's average RTV_c, showing their 25, 50, and 75 percentiles. Moreover, we also draw the average values by circular dots, in each case evaluated. We can see that the proposed TRA achieves the highest average reduction per tenant (shown by higher positions of circular dots), for all three tail percentiles. Specifically, the average RTV_c under the TRA prediction scheme is around 50, whereas CP and NP can only achieve the average tail reduction per tenant around 30-40. Another worth mentioning finding is that, when increasing the tail percentiles, i.e., from 90 percentile to 98 percentile, the reduction drops slightly for all prediction schemes. This is because a more stringent performance requirement is applied, allowing only a very small number of AIs, and the potential of reducing violation by redistributing the box loads is thus lower.

This also leads to the explanation that why CP scheme outperforms NP in the case of 90 and 95 percentiles, but NP results into a better reduction in the case of 98 percentile. To better highlight the impact of different prediction schemes on mitigating tail target violation, we zoom into the performance of two tenants. One of the tenants has a lower CPU utilization, meaning high resource availability, whereas the other tenant has a higher CPU utilization. We list their average RTV_c in Table **??**. On the one hand, CP is able to reduce all tail violations for the tenant with a higher resource availability but performs poorly for the other tenant. On the other hand, NP has the opposite performance for these two tenants, arguing for the need of a prediction scheme that can adopt its conservativeness to the resource availability. Indeed, RTA prediction is enable VM cloning to achieve the highest amount of reduction for both tenants, as it uses different α values for box tail predictions.

Fig. 4: CPU usage tail target violation reduction comparison: our tenant specific tail prediction *v.s.* fixed steady state based tail prediction.

TABLE I: Average reduction in tail violation (RTV_c):a comparison with CP, NP and TRA. A case study on the tail of 95 percentile

	Tenant with higher resource	Tenant with lower resource availability
CP($alpha = 2$)	100	16.7
NP ($alpha = 0$)	75.0	50
RTA (α_i)	100	50

D. Effectivness of VM cloning

Fig. 5: CPU usage tail target violation reduction comparison: our method *v.s.* load balancing using VM migration.

(a) Reduction in CPU AI (%) (b) Duration of CPU AI

Fig. 6: Comparison of CPU AI for each tenant between VM cloning and VM migration: the tail target is 95%ile.

TABLE II: RAM violation comparison between VM cloning and VM migration.

	Mean RAM Violation Increment (%)		
	90%ile	*95%ile*	*98%ile*
VM Cloning	0	2.5	2.5
VM Migration	59.7	59.7	59.7

References

[1] I. Giurgiu, J. Bogojeska, S. Nikolaiev, G. Stark, and D. Wiesmann, “Analysis of Labor Efforts and their Impact Factors to Solve Server Incidents in Datacenters,” in *CCGrid*, 2014.

[2] R. Birke, I. Giurgiu, L. Y. Chen, D. Wiesmann, and T. Engbersen, “Failure analysis of virtual and physical machines: patterns, causes and characteristics,” in *DSN*, 2014, pp. 1–12.

[3] I. Giurgiu, A. Almasi, and D. Wiesmann, “Do you know how to configure your enterprise relational database to reduce incidents?” in *IM*, 2015.