Adaptive Monitoring: A Hybrid Approach for Monitoring using Probing

Deepak Jeswani *†, R. K. Ghosh†, Maitreya Natu ‡
*Student author
†Indian Institute of Technology, Kanpur, India
‡Tata Research Development and Design Centre, Pune, India
Email : djeswani@iitk.ac.in; rkg@iitk.ac.in; maitreya.natu@tcs.com

***Abstract*—Data centers, now a days, typically deploy monitoring agents to collect various performance metrics across several data center components. An important first step in this respect is to arrive at a set of monitoring metrics and the level or frequency of monitoring. Monitoring each and every metric at a high frequency (e.g. every second), would produce very large size of monitoring logs. Storing as well as analyzing such logs pose many challenges. That apart, there is a high probability that certain interesting data would get buried under very large data-sets and escape critical analysis. On the other hand, monitoring a few metrics at a low frequency aggravates the risk of losing important information relating to events of interest. In this paper, we propose a novel approach to apply probing-based adaptation of the monitoring tools. We present algorithms to analyze probe performance which enable us to derive monitoring recommendations. We compare traditional monitoring with adaptive monitoring, and show that the latter method significantly decreases the volume of monitoring data, without loosing those pertaining to interesting events.**

I. Introduction

The continuous evolving nature of today's data centers has led to their incremental and unplanned growth. As a result, the data center operators lack a complete understanding of the underlying system behaviour. The problem is aggravated further with increasing number of performance critical applications relying on services of these data centers. Therefore, data centers need to be continuously monitored for various metrics such as performance, capacity, etc. The data collected from such monitoring are then analyzed to diagnose and heal performance and capacity problemswhile maintaining seamless uninterrupted data services.

Various tools and techniques have been proposed for monitoring the operations of data centers. These techniques can be broadly classified under two: (i) component-level passive monitoring-based techniques, and (ii) end-to-end probing-based techniques. The passive monitoring techniques typically involve deployment of agents (eg., Tivolli, Nagios) at each machine to periodically collect various performance metrics such as CPU utilization, page faults, etc. While the tools based on passive monitoring focus on individual component, the probing-based tools compute the end-to-end metrics such as latency, throughput, etc. Probing-based tools [5], [1], [6] send test transactions (such as pings, HTTP requests, etc.) through the system and analyze their performance. Most of the earlier work in using both passive monitoring [3], [2] as well as probing has been in isolation. In this paper, we argue that these techniques can be used in combination to leverage eachother's effectiveness. Our primary focus here is on using probing to improve the effectiveness of monitoring tools. We believe that similar solutions can be built for the other case of using monitoring information to improve probing.

Monitoring tools provide a sufficient and rich amount of information about the behaviour of a server. These tools provide various metrics ranging from per-processor performance metrics, to network traffic metrics, to application-level performance metrics. The tools also have provisions to allow selection of monitoring metrics and the frequency of monitoring. However, one of the biggest problems faced by the data center operators is making a decision on what metrics to monitor at what time? Monitoring all the metrics at a very high frequency (e.g. every second) produces enormous amount of monitoring logs. Storing as well as analyzing such logs pose several challenges. Furthermore, interesting data tends to get buried in such large data-sets and may possibly escape careful analysis. On the other hand, monitoring very few metrics at a low frequency incurs the risk of losing important information and events of interest. In the rest of paper, we use the term *monitoring*

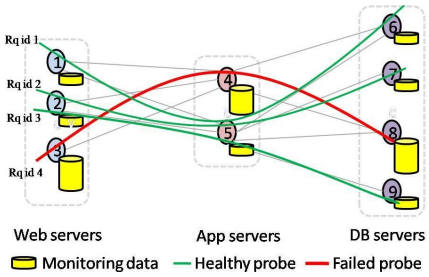


Fig. 1. An example data center.

level to refer to the aggressiveness of monitoring. A low monitoring level refers to monitoring fewer metrics at a low frequency. A high monitoring level, on the other hand, refers to monitoring of large number of metrics at a high frequency.

In this paper, we argue that the information obtained through probes can be used to address the above concerns and improve the effectiveness of per-machine monitoring. The end-to-end metrics collected by a probe are an indicative of the likely health of the components serving the request. Hence, a probe result can be used to set the appropriate monitoring levels of these components. Consider an example data-center of an equity trading plant (Figure 1) where several requests for equity trades and the market updates are processed each day. Each request passes through several processing steps using various components. An increase in the end-to-end latency of serving a particular request (request 4 in Figure 1) is a likely indication of performance problem at one or more components serving the request. Furthermore, different requests demand different resources, e.g., CPU intensive requests, IO intensive requests, database intensive requests, etc. Thus, a poorly performing request can provide insights into the health of not just the component but also the specific resources within the component. In this paper, we exploit this observation and propose an adaptive-monitoring solution where we use the information from probe results to set monitoring levels of individual components in the data-center.

Various challenges need to be addressed in order to build such a solution. For instance, how to select probes? How to analyze various metrics obtained from probe results to derive a recommendation for monitoring level ? How to compute a monitoring-level for a component from the monitoring recommendation obtained from multiple probes? How frequently should the monitoring-levels be changed?

The key contributions of this paper are as follows:

- A novel adaptive monitoring algorithm that uses end-to-end probe for deriving monitoring levels of individual components.
- An experimental evaluation to demonstrate that the generated monitoring data (through the above adaptive monitoring algorithm) successfully captures all interesting properties while retaining a significantly low volume monitoring data.

II. Design Rationale

Building a solution for adaptive monitoring involves following four major steps:

1) Selection of probes:: An important initial problem to address is the selection of right set of probes. Probes can be ongoing system transactions or customized synthetic traffic. The probes should be selected such that the monitoring recommendations can be provided to all components of interest by analyzing the probe results. A lot of earlier work on probing can be useful in addressing this problem.

2) Analysis of probe performance:: An important step of adaptive monitoring is the analysis of the probe performance metrics. Adaptive monitoring requires analysis of these end-to-end metrics to infer the health of various components serving the probe. The analysis needs to capture various events such as sudden changes, gradual changes, and deviation from the normal behavior.

3) Deriving monitoring recommendations from probe performance:: The analysis of end-to-end metrics provides insights into component health. This analysis thus provides enough indication on criticality of monitoring a specific component. These insights need to be translated to monitoring levels.

4) Setting monitoring levels:: Once the monitoring recommendations are derived, the monitoring agents at the component need to be re-tuned to the new monitoring level. The solution for this step makes various decisions such as: how frequently should the monitoring levels be changed? Monitoring levels of which set of components should be changed together?, etc.

This paper primarily focuses on step 2 and 3. For clarity, we make simplistic assumptions that the probe selection is done using domain knowledge. We present algorithms for analysis of probe results and mapping end-to-end probe analysis to component-level monitoring recommendations.

III. Proposed Approach

For the sake of clarity, let us first examine a simple scenario where only a single probe is sent through the system and a single end-to-end metric is collected. After

explaining the details of the algorithm, we address the complex case where multiple such probes are analyzed. Suppose, for example, a probe passes through three components and the end-to-end latency of the probe is measured. All components are equipped with monitoring agents that monitor resources such as CPU utilization, memory, page faults, database calls, file IO time, etc.

A. Analysis of probe results

Step 1- Aggregation of probe results: To avoid responding to noise and transient fluctuations in the collected end-to-end metric, we remove outliers and aggregate the end-to-end metrics by computing the mean of all values collected over a time window w. Thus, over a period of time, a time-series T of the end-to-end metric is built, where each point T_i in this time-series is aggregated over a time-window w. We use the standard sliding window mechanism where the active window slides over time and windows overlap each other. Later on, we present heuristics to dynamically change the window size and the amount of window overlap.

Step 2- Compute the deviation from the normal behavior: The presence of abnormal behavior in the end-to-end metric is evaluated next. The expected value for the end-to-end metric can be obtained using domain knowledge (e.g., SLAs) or by analyzing historical data. Keeping the expected normal values as the benchmark, we analyze if the observed value deviates from the expected normal behavior. Given the normal value to be T_{nor} and the observed value to be T_{cur}, the deviation of the observed value from the normal behavior is computed as follows:

$$Dev(T_{cur}) = abs(T_{cur} - T_{nor}) \tag{1}$$

Step 3 - Compute the rate of change: The rate of change provides insights into the severity of the change and the likely future values of the metric. For instance, the rate of change can differentiate between a linear increase and an exponential increase in the latency. Also, it can differentiate a scenario of 50% increase from a 5% increase in latency. Given the value T_{cur} observed in current time window and the value T_{prev} observed in previous time window, we compute the rate of change as follows:

$$Rate(T_{cur}) = abs(T_{cur} - T_{prev})/T_{prev} \tag{2}$$

B. Deriving monitoring recommendations

The analysis performed on the end-to-end metrics of a probe then needs to be mapped to a recommended monitoring level. A rule-book approach is proposed to map the two metrics $Dev(T_{cur})$ and $Rate(T_{cur})$ to the monitoring levels $ML(Dev(T_{cur}))$ and $ML(Rate(T_{cur}))$ respectively. The domain knowledge and standard practices along with an analysis of historical data can be used to build such rule-books. In addition to this, learning mechanisms can be used to adjust the rule-book settings over time. We do not discuss further details of rule-book creation due to lack of space. We compute the monitoring level for a time window T_{cur} based on rule-book recommendations and monitoring level of the previous time window T_{prev}. The monitoring level is then calculated as follows:

$$ML(T_{cur}) = ML(T_{prev}) + (ML(Dev(T_{cur})) \pm ML(Rate(T_{cur})))/2 \tag{3}$$

where, the addition or the subtraction of $ML(Rate(T_{cur}))$ depends whether rate of change is negative or positive. Note that by incorporating the previous monitoring level in the computation of next monitoring level, we build monitoring levels on the previous decisions. Instead of analyzing the full history of the probe performance, the most recent monitoring level provides best representation of the past inferences.

C. Adjustment of the window of probe result analysis

The standard sliding window mechanism has been used to construct time-windows. The collected end-to-end metric values are aggregated over these time-windows. There are two metrics to be considered here: (i) the size of the time-window, (ii) the amount of overlap of the current window with the previous window. The size of the time-window controls frequency of analyzing the probes to compute monitoring recommendations. The amount of overlap controls the extent of historical data to be used while deriving monitoring recommendations. For the problem of adaptive monitoring we propose to dynamically change the window size and amount of overlap using the following heuristics.

A probe observing a steady behavior need not be checked very frequently for adjusting the monitoring levels of components through which it passes. Furthermore, in such scenario, the consecutive windows tend to be similar in nature. Hence, if the monitoring level of *k* consecutive time-windows is observed to be same, then the window size of *(k+1)* time-window is increased and overlap is decreased. We propose to increase the window size in an additive manner to ensure a conservative increase. We propose to decrease the amount of overlap in a subtractive manner.

On the other hand, if the probe observes a fluctuating behaviour, the monitoring levels need to be tuned in

a more cautious manner. If the monitoring levels of *k* consecutive time-windows observe changes, then the time-window is decreased. To act quickly to changes we propose to decrease window size in an aggressive manner by performing a multiplicative decrease. In such scenario, the amount of overlap is kept high to resist noise. We propose to increase the amount of overlap in an additive manner.

D. Addressing multi-path scenario

As a component can serve more than one probe, We next present our approach to incorporate multiple monitoring level recommendations to derive a consolidated recommendation.

Different probes being served by the component may provide different monitoring recommendations. However, the confidence in the monitoring recommendation provided by a probe can be computed based on the length of the probe. When a component *C* is served by *k* probes, we assign a weight to the monitoring recommendation of each probe. The weight is inversely proportional to the length of the probe. We then compute a weighted average of the recommendations to derive a consolidated recommendation.

Consider a scenario where 3 probes pass through a node N. The number of nodes on these probes are 10, 5, and 2 and the monitoring level recommendations of these probes are 2, 3, and 5 respectively. The monitoring level for node N is then derived as follows:

$$ML(N) = (1/10 * 2) + (1/5 * 3) + (1/2 * 5) = 3.3 \quad (4)$$

IV. Experimentation and Results

In this section we present an experimental evaluation to demonstrate the correctness of the proposed approach.

A. Effectiveness in capturing changes and steady states

Figure 2 presents various scenarios of changes and the steady states in the end-to-end latency time-series. Figure 2 also shows the derived monitoring levels for each of the time-series. The rate of increase in the end-to-end latency is higher in Figure 2(a) than in Figure 2(b). This property is correctly captured and reflected in the computed monitoring levels. Figure 2(c) shows another case where the end-to-end latency shows a periodic behavior of an increase followed by a decrease. The monitoring levels in Figure 2(c) show that an increase in latency is quickly captured resulting in a quick increase in monitoring levels. The algorithm also effectively captures the normal behavior of the probe and decreases the monitoring level on observing normal values.

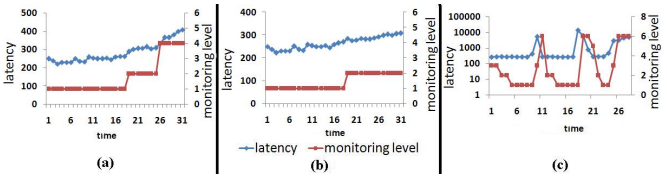
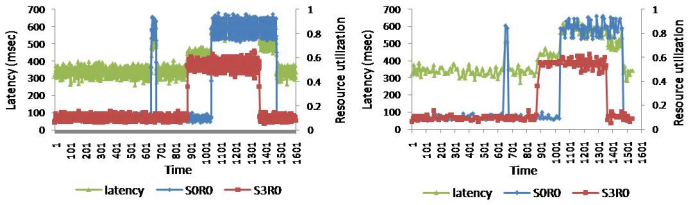


Fig. 2. Different scenarios of variations observed in end-to-end latency and the corresponding derived monitoring level recommendations.

B. Comparison of traditional monitoring and adaptive monitoring

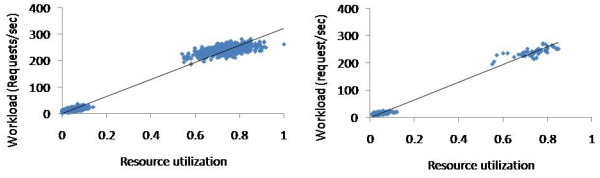
We now compare the traditional monitoring and the adaptive monitoring under two criteria: (i) amount of monitoring data collected, (ii) accuracy of analysis. We simulated a three-tier data center using CSIM [4]. The topology consists of 6 servers S0 through S5. We modelled each server and its resource using the Machine Repairman Model. At each server we collect four resource utilization metrics namely, database lock time, CPU utilization, available memory, and disk write time referred to as R0, R1, R2, and R3 respectively. In the case of traditional monitoring, these metrics are collected every 3 seconds. In the case of adaptive monitoring, the frequency of metric collection varies depending on the monitoring level recommendations. We generated end-to-end probes through the data center and collected probe results to analyze end-to-end performance. We next present the results of two very common analysis operations namely *fault localization* and *headroom analysis*. Through these experiments we demonstrate that adaptive monitoring collects significantly less amount of monitoring data than traditional monitoring while maintaining the accuracy levels of traditional monitoring.

1) Fault Localization: In this experiment, the monitoring logs are used to perform fault localization. Multiple faults are inserted into the system (increase in database lock time R0 of Server S0 and Server S3, referred to as S0R0 and S3R0 hereafter). These faults result in increase in end-to-end latency. Figure 3(a) and Figure 3(b) present the data collected by traditional and adaptive monitoring respectively along with the observed end-to-end latency. The sudden increase in the values represent the failure regions. The number of data points collected by traditional monitoring was 1602 while that collected by adaptive monitoring was only 164. It can be seen from Figure 3(c) that even though the data points collected by adaptive monitoring is almost 10% of that of traditional monitoring, the former successfully captures all interesting events captured by the latter monitoring



	First fault (Correlation coefficient)	Second fault (Correlation coefficient)	Data points
Traditional monitoring	SORO (0.91)	S3RO (0.77)	1602
Adaptive monitoring	SORO (0.92)	S3RO (0.80)	164

(c)



	Linear model	Capacity	Data points
Traditional monitoring	workload = 322.14*request + 0.144	276.16	2881
Adaptive monitoring	workload = 324.62*request + 0.524	266.17	252

(f)

Fig. 3. (a, b, c) Traditional monitoring, adaptive monitoring and statistics for fault localization. (d, e, f) Traditional monitoring, adaptive monitoring and statistics for headroom analysis.

and removes redundant data.

We use a standard technique for fault localization based on event correlation. We compute correlation coefficient of all resource metrics with end-to-end latency. The metrics with very high correlation-coefficient are reported as likely causes. While more sophisticated techniques have been presented in literature, we use this technique to demonstrate the proof of the concept. As shown in Figure 3(c), both adaptive as well as traditional monitoring correctly identify the inserted failures as the likely causes. The adaptive monitoring, thus, builds a monitoring log that is 10% the size of the log built using the traditional monitoring, without compromising on the accuracy of the fault localization analysis.

2) Headroom analysis: In the next experiment, we use the monitoring logs to compute the resource headroom. The headroom of a resource represents the amount of available resource and the workload that can be supported by that resource. We build regression models between workload and resource utilization using monitoring logs obtained from both traditional monitoring and adaptive monitoring. Figure 3(d) and Figure 3(e) show the model built using traditional monitoring and adaptive monitoring respectively. Figure 3(f) shows the equation of the linear models. It can be seen that the models and capacity estimation are very similar for the two logs. Adaptive monitoring thus significantly reduces data log size (252 vs. 2881) without compromising on the model accuracy.

V. Conclusion and Future Work

In this paper, we propose a novel approach to use probing-based solutions to adapt the monitoring tools. We present algorithms to analyze the performance of probes. We present the techniques to derive monitoring recommendations from the analysis of probes. We present a comparison of the traditional monitoring with adaptive monitoring on the basis of (i) amount of data collected and (ii) accuracy of analysis performed on the logs generated by the two logs. We demonstrate that adaptive monitoring significantly decreases the volume of collected monitoring data, without loosing any interesting events.

Our future plans include addressing various issues such as (i) automatic building of rule-book , (ii) giving user to choose probes, etc. We also plan to evaluate the proposed approach by deployment of a prototype tool on a real-world data center. Like adaptive monitoring, we also plan to explore the possibility of adaptive probing to improve the end-to-end probing solutions based on the insights obtained from component monitoring.

References

[1] A. Frenkiel and H. Lee. EPP: A framework for measuring the end-to-end performance of distributed applications. In *Performance engineering 'Best Practices' conference, IBM Academy of Technology*, 1999.

[2] L. P. Gaspary and E. Canterle. Assessing transaction-based Internet applications performance through a passive network traffic monitoring approach. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.*, 2004.

[3] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju, and James Won-Ki Hong. The architecture of NG-MON: A passive network monitoring system for high-speed IP networks. In *13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2002, Montreal, Canada*, 2002.

[4] http://www.mesquite.com/. Csim 20 development toolkit for simulation and modeling.

[5] N. Hu and P. Steenkiste. Towards tunable measurement techniques for available bandwidth. In *Bandwidth Estimation Workshop (BEst03), San Diego, CA*, 2003.

[6] M. Natu and A. S. Sethi. Application of adaptive probing for fault diagnosis in computer networks. In *IEEE/ IFIP Network Operations and Management Symposium, Salvador, Brazil*, Apr. 2008.