Adaptive Probing: A Monitoring-Based Probing Approach for Fault Localization in Networks

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

***Abstract*—The past research in fault localization in distributed data centers has used either monitoring or probing techniques in isolation. In this paper, we argue that effective fault localization solutions can be built by exploiting the information captured by both the techniques. Based on this concept, we propose an adaptive probing solution for fault localization where information from monitoring agents is used to adapt probing policies such that probe traffic is reduced, localization accuracy is increased, and localization time is minimized. We demonstrate the proof-of-concept through experimental results.**

Keywords-Fault Localization, Probing, Monitoring.

I. INTRODUCTION

Today's enterprise systems are increasing in scale and complexity; and are being used for serving various performance critical applications. For smooth operations of these systems, there is an increasing demand to monitor the system performance and localize any component failure in minimal time.

The past approaches for monitoring can be classified into two categories: (a) component-level passive monitoring, and (b) end-to-end probing. Passive monitoring techniques [2,5] involve deployment of monitors at each component to collect system metrics (e.g. CPU utilization, memory usage etc.) periodically. Probing-based techniques [3, 4] send test transactions (e.g. ping, trace routes) through the network to infer network health. Passive monitoring-based techniques provide fine-grained metrics, but fail to provide end-to-end view of the system. Probing-based techniques, on the other hand, can provide end-to-end metrics but introduce additional traffic in the network and fail to provide fine-grained analysis.

Most of the fault localization solutions, proposed in the past, are based on using either probing or monitoring in isolation. We argue that effective fault localization can be designed by using the information captured by both probing and monitoring agents. In this paper, we present the design of a monitoring-based adaptive probing solution for fault localization.

The ability to measure and analyze end-to-end metrics has encouraged the development of various probing-based solutions for fault localization in networks. However, the past probing-based solutions suffer from the limitation of lack of node-level view. All probing-decisions are based only on the result of probes sent in the past and do not use the information captured by the passive monitors. The probes thus selected fail to balance the inherent tradeoff between probe traffic and localization time. In this paper, we argue that effective solutions can be designed by using monitoring agents to adapt the probing policies. The adaptive probing-based solutions thus built for fault localization can (a) provide accurate fault localization, (b) minimize fault localization time, and (c) minimize the additional probe traffic created in the network.

The adaptive-probing solutions proposed in the past [3,4] use the following 2-step approach -
1. A small set of probes are sent periodically to detect the presence of failure in the network. The probes only detect the presence of a failure but do not localize the failed nodes. The selection of probes and their probing frequency should be such that (a) probe traffic is minimized and (b) detection time is minimized.
2. On detection of a failure, additional probes are sent to localize the exact failure. The selection of probes should be such that (a) fault localization accuracy is high, and (b) localization-time is minimized.

The above steps of detection and localization can be improved using the information collected by monitors. The metrics collected by the monitoring agent at each node can be used to estimate potential failure or change in steady-state of a node. Monitoring information can then

be used to adapt the probing policies for detection and localization of failure in order to improve probe-traffic, localization time, and localization accuracy.

In this paper, we present solutions for fault localization centered around the idea outlined above. Initially, the monitoring data is used to get insights into the performance properties of nodes and paths. Then the probes and the probe frequencies are determined on the basis of these monitoring insights. The selection of probes thus helps to keep both probe traffic and detection time low. Probe selection is further refined and optimized on the basis of additional insights gained from the results of past probes and monitoring agents. With the above approach, localization time is kept low and the accuracy of fault localization becomes high.

We discuss each of these problems in detail in the following sections. In particular, our focus is on examining the opportunities and the challenges in solving these problems. We present some initial ideas on our ongoing research in building a desired solution in lines stated above. We also present initial experimental results to demonstrate the proof-of-concept.

II. Related Work

Initially, for fault localization, pre-planned probing technique was used, in which pre-compiled probes were sent to localize the cause of any fault. For reducing the management traffic, active probing was introduced which divided the work into two sub-parts, viz., fault detection and fault localization reducing the number of probes. Both the pre-planned and active probing approaches are described in [4]. Fault detection was done by selecting minimum number of probes which can cover all nodes in a greedy manner and Fault localization was done by using min, max or binary search proposed in [3].

In the literature, fault localization was done in various ways through monitoring. In [5], the problem of finding root cause of failure is solved by building a dependency graph in the network. There has been very little work in combining probing and monitoring [2]. An efficient approach to dynamically adjust the monitoring level of monitors on the basis of probe results was mentioned in [1]. Considering the state of work in the area, our approach assumes significance as it attempts to balance the two techniques in solving the core problem of fault localization.

III. Problem Formulation

Figure 1 presents the basic building blocks of the proposed approach.

A. Health-check engine

Monitors are deployed on the nodes and the monitored metrics are fed to the health-check engine to estimate the health of each node. The health of a node is an indicator of a potential node failure or a change in the steady state of the node. Several design decisions need to be made at the health-check engine such as: (a) given a large number of metrics, how to choose the best representative set of metrics. (b) how to compute change in the steady state of a node? (c) how to compute abnormality of a node behavior? We discuss details of health-check engine in Section IV.

Note that node-level analysis only provides a silo-based understanding of the system. Reporting of failed nodes based purely on monitoring information can lead to many false positives and false negatives. The node level analysis needs to be supported by probing to measure end-to-end effect of the unexpected behavior of a node.

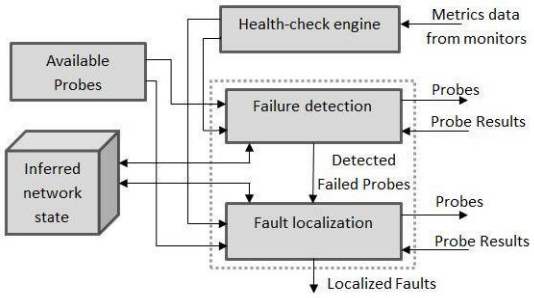

Figure 1: Proposed architecture

B. Failure detection

Probes are sent from probe stations. Probe stations are special nodes instrumented to send probes and receive probe results. Probes should be selected carefully because sending a large number of probes will create a huge burden on the already existing application traffic. The probes for failure detection are selected only to detect the presence of failure in the network and not to localize the cause of failure. The failure detection module uses the node health information to select probes and probe frequency for failure detection such that (a) any failure in the network can be detected, (b) probe traffic is minimized, and (c) failure detection time is

minimized. We discuss details of failure detection in Section V.

C. Fault localization

The fault localization component selects additional probes to be sent into the network to further analyze the nodes on the failed probe paths. Fault localization probes are invoked only in the event of presence of failure and should be designed to quickly and accurately localize the fault. The fault localization module uses the node health information to select probes such that (a) fault localization is accurate, and (b) localization time is minimized. We discuss details of fault localization in Section VI.

IV. Computing Health Measure of a Node.

A. Select the representative metrics

Monitoring agents typically collect a wide variety of system metrics including CPU and memory utilizations, disk IO, page faults, thread pool, heap size, network bytes in/out, etc. It is critical to identify a representative set of metrics in order to analyse a node health. While expert-knowledge-based selection of metrics cannot be completely avoided, some statistical filters can be applied to prune the redundant, derived, and dependent metrics. We propose to use invariant detection techniques based on Principal Component Analysis (PCA) and Classification and Regression Trees (CARTs) to identify the representative metrics.

B. Compute change in steady state of metrics

After identifying the representative metrics, the next step is to model the normal behavior of a metric. In many scenarios, standard compliance thresholds are prescribed for many system metrics such as CPU utilization, page faults, etc. In the absence of such benchmarks, we propose a simple statistical approach to compute the normal behavior of a metric based on historical data captured under the normal system conditions. Different techniques have been proposed in the past for representing normal behavior of a metric time-series, but for simplicity, we compute the mean MN(x) and standard deviation SDN(x) of a metric x on the historical data and update it periodically.

The metric *x* is then evaluated over periodic intervals to check for any deviation from normal behavior. We propose to evaluate the abnormality based on change in both mean and standard deviation. Consider $M_t(x)$ and $SD_t(x)$ as the mean and standard deviation of the metric *x* in time-interval *t*. The node abnormality can then be computed as follows:

$$\alpha*(|M_t(x)-M_N(x)|)/M_N(x) + \beta*(|SD_t(x)-SD_N(x)|)/ SD_N(x)$$

,where α and β are constants (such that $\alpha+\beta=1$) to assign different weights to change observed in mean and standard deviation.

C. Compute node abnormality

A metric for node abnormality can be computed as a function of abnormality of its representative system metrics. A conservative approach for calculating node abnormality is to compute the maximum or 90^{th} percentile of the abnormality values computed for all representative measures. In more informed scenarios, a weighted average of the abnormality values can be used to compute the node abnormality. The weights can be chosen based on available knowledge of metric importance. For instance, for a node intended to perform CPU intensive tasks the abnormality of CPU metrics will be given higher weight over the IO metrics.

V. Failure Detection

The traditional approach of failure detection has been to select a small set of probes that covers all network nodes and send these probes at a high frequency to quickly detect presence of any node failure. In this section, we demonstrate how the node abnormality information computed above can be used to intelligently select the probes and probing frequencies for failure detection, thereby resulting in less probe traffic and quicker detection.

A. Desired probing frequency for node and probe

The desired probing frequency for a probe can be recommended based on the abnormality measure of its nodes. Thus, we propose to compute path abnormality as a function of abnormalities of its nodes. Various functions can be used to compute path abnormality. A conservative approach is to choose maximum of the node abnormality values, while an aggressive approach is to choose minimum. For the following discussion, we consider path abnormality as an average of abnormality values of the nodes on the path. Thus:

$$\text{Abnormality}(P)=\text{Avg}_{n \in \{\text{Nodes}(P)\}}(\text{Abnormality}(n))$$

The path abnormality can then be used to recommend probing frequency for a probe. Node showing high abnormality should be probed at higher frequency to quickly detect potential end-to-end performance problem. Node showing consistent normal behaviour should be probed at

a lesser frequency to reduce unnecessary probe traffic. Similarly, a probe with high path abnormality is likely to pass though unhealthy nodes demanding higher probing frequency. A probe with low path abnormality passes through the nodes showing expected normal behaviour and hence can be probed at a lower frequency.

Note that a very aggressive approach of large decrease in probing frequency can harm in scenarios where a node failure causes end-to-end problems but the monitoring agents fail to capture the node's abnormal behaviour. A very large decrease in probing frequency of the perceived healthy paths can save on probe traffic but can cost increase in detection time. Hence, frequency should be set appropriately to balance such trade-off. For simplicity, we propose to use a rule-book to map range of node and path abnormality to recommended probing frequency. This rule-book can be constructed based on network properties and expert knowledge. As part of our on-going research, we are working on techniques to automate the rule-book construction and adapting the rule-book to changing network conditions.

Algorithm : Probe Selection for Fault Detection

Input: ProbeSet, Abnormality%, Abnormality to Freq. mapping of nodes.
Output: Fault Detection ProbeSet.
1. For each probe, calculate the following measures:
- *Average Probing Frequency(APF)* = Average of the desired probing freq.(DPF) of all nodes in that probe.
- *Additional Probing cost(APC)* = Sum of the additional probing costs of all nodes in that probe. It is APF – DPF if DPF<APF else 0.
- *Monitoring loss(ML)* = Sum of the monitoring losses of all the nodes in the probe. It is DPF – APF if DPF>APF else 0.
- *Coverage* = number of nodes in the probe which are not yet covered.
- *ProbeScore* = α **APC* + β **ML* + γ **Coverage*

where α, β, γ are constants whose typical values are -1, -2 and 3.
2. Include the probe which has maximum *ProbeScore* in the probeset.
3. Keep repeating Steps 1 and 2 till all nodes are covered.

Algorithm 1: Fault Detection Probe-set Selection

B. Probe Selection

The probes for failure detection should be selected on the following criteria:
(a) Coverage: This criterion indicates the number of nodes covered by a probe. The number of failure detection probes can be decreased by selecting probes that cover large number of nodes that are not covered by already selected probes.
(b) Probe frequency match: This criterion indicates the match between the recommended and the desired probing frequency at each node on the probe path. Consider a scenario where many nodes on a probe path *P* demand a lower probing frequency than the probing frequency of the probe path *P*. In such cases, the probe *P* will result in the cost of sending additional probe traffic on these nodes. Also, when many nodes demand a probing frequency higher than the frequency of the probe path, it will result in collection of less data leading to potential delayed detection of failure of these nodes. Thus, the probes passing through nodes with similar probe frequency demands should be preferred to maximize the probe frequency match. We propose to compute following two metrics to evaluate a probe: (i) *Additional probing cost* (APC): to address scenarios where path probing frequency is greater than required node probing frequency. (ii) *Monitoring loss* (ML): to address scenarios where path probing frequency is smaller than required node probing frequency.
Algorithm 1 computes *ProbeScore* for each probe based on the coverage, additional probing cost, and monitoring loss by the equation:
ProbeScore = α * APC + β * ML + γ * Coverage
α, β and γ are set such that the selected probe passes through nodes with similar desired probing frequencies. We initialize these constants as -1,-2 and 3. However, they can be set to other values based on requirements. The algorithm selects probe with Max *ProbeScore* to group nodes with similar desired probing frequencies together till all the nodes in the network are covered.
In order to capture changing network conditions, the probe selection process should be performed at periodic intervals or in the event of increase in unhealthy node count.

Algorithm : Probe Selection for Fault Localization

Input: ProbeSet, Abnormality%, Threshold, Suspected NodeSet(N)
Output: Fault Localization ProbeSet.
1. For each probe, compute the coverage as follows:
Probe Coverage = $N_{healthy}$ - $N_{unhealthy}$
2. Cover Healthy Nodes.
i. Select the probe that has largest *Probe Coverage*.
ii. Keep repeating step 2.i till all healthy nodes are diagnosed.
3. Cover Unhealthy Nodes.
i. Select the probe that has smallest *Probe Coverage*.
ii. Keep repeating step 3.i till all unhealthy nodes are diagnosed.

Algorithm 2: Fault Localization Probe-set Selection

VI. FAULT LOCALIZATION

Once a failure is detected by failure detection probes, additional probes are sent to localize the failed node(s). In this section, we demonstrate how probe selection for fault localization can be improved by using the monitoring information. We demonstrate this improvement using the example of Min and Max Search proposed as part of the adaptive probing solution in [3]. We refer to the nodes on the path of the failed failure-detection probes as the suspected nodes.
(a) Min-Search: Min Search selects a probe for each suspected node which passes through the least number of other suspected nodes. Here, a failed probe quickly localizes the failed node, but a successful probe does not significantly prune the suspected node set.

(b) Max-Search: Max Search selects probes that cover maximum number of suspected nodes. In this search, a failed probe needs additional probes to further localize the root-cause, but a successful probe significantly prunes the set of suspected nodes.

We propose to use node health information to exploit the strengths of both Min and Max search. Probes for healthy and unhealthy nodes can be selected based on Max-Search and Min-Search policy respectively. Thus, the probes selected using Max-Search policy are likely to succeed and hence will effectively prune the search space with minimal probes. The probes selected using Min-Search policy are likely to fail and will quickly localize the failed node(s). Algorithm 2 presents the proposed algorithm.

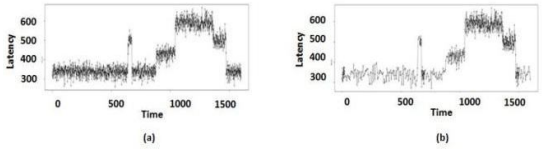

Figure 2: Latency time-series sent by (a) traditional adaptive probing (1601 probes), (b) proposed adaptive probing (776 probes)

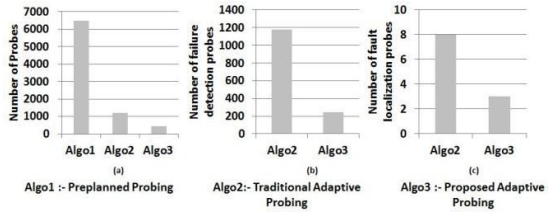

Figure 3: Probes sent by different probing approaches. (a) total probes,(b) Probes for failure detection, (c) Probes for fault localization.

VII. Experiments And Results

We simulated network topologies traffic through CSIM and evaluated the probing approaches to localize node failures. A node failure is modeled as a gradual increase in node processing latencies resulting in high end-to-end latencies of the ongoing traffic. At each node, we collected resource utilization metrics. Pings were sent as probes and latencies were calculated. We compared proposed approach with pre-planned probing [4] and adaptive probing [3].

We first demonstrate the case of a single probe path where traditional adaptive probing sends probes at a constant frequency and the monitoring-based adaptive probing adapts the probing frequency. The monitoring-based adaptive probing approach is able to capture all the states of normal and unexpected behavior even after sending significantly lower traffic (Figure 2).

We next performed experiments on a 20-node topology to localize failure of 2 nodes. All three algorithms accurately localized the failed nodes. We computed the number of probes sent by pre-planned, traditional-adaptive, and proposed probing algorithms. Figure 3(a) shows that the number of probes computed by the proposed approach (453) is significantly less than pre-planned probing (6490) and traditional-adaptive probing (1188). Figure 3(b) and 3(c) compute the number of probes sent by traditional-adaptive probing and the proposed approach for failure detection and fault localization. It can be seen that the proposed approach provides significant savings on the probe traffic.

VIII. Conclusion And Future Work

We presented an adaptive probing solution for fault localization in a network by adapting the probing policies using the information provided by monitoring agents. We presented initial ideas to infer information captured by monitoring agents, and used this information to select probes and their frequency for failure detection and fault localization. We presented the proof-of-concept through simulations.

While the proposed solution demands deployment of both monitoring and probing agents, we argue that an adaptive approach can significantly decrease the probing and monitoring demands compared to cases of using either of them in isolation. As part of our going research, we are also working on intelligently selecting minimal nodes to deploy monitoring agents. We are also working on several other aspects of this solution such as (a) different ways to compute node health and path health, (b) automating the rule-book to map path-health to probe frequency, (c) extensive experimental evaluation of proposed solution in real set-ups and simulations.

References

[1] D. Jeswani, R.K. Ghosh and M. Natu . “Adaptive Monitoring: A Hybrid Approach for Monitoring using Probing”. In *HIPC* 2010.

[2] S.-H. Han, M.-S. Kim, H. Ju, and J. Hong. The architecture of NG-MON: A passive network monitoring system for high-speed IP networks. In *DSOM* 2002, Montreal, Canada, 2002.

[3] M. Natu, A. Sethi, “Application of Adaptive Probing for Fault Diagnosis in Computer Networks”, In *NOMS*, Salvador, Brazil, 2008.

[4] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez. Adaptive diagnosis in distributed systems. *IEEE* Transactions on Neural Networks, 2005.

[5] P. Bahl , R. Chandra , A. Greenberg , S. Kandula , D. A. Maltz and M. Zhang "Towards highly reliable enterprise network services via inference of multi-level dependencies", *Proc. ACM SIGCOMM*, 2007.