Using Bayesian Networks to Reduce SLO Violations in a Dynamic Cloud-based Environment

Agam G. Dua

Tandon School of Engineering
New York University
Brooklyn, New York
Email: agam@nyu.edu

Aspen Olmsted

Fisher College
Boston, Massachusetts
Email: aolmsted@fisher.edu

***Abstract*—As more organizations move critical infrastructure to the cloud and leverage features like auto-scaling to grow according to the customer demand, we see a new set of challenges specific to this class of dynamic, distributed systems. In this paper, we propose a model leveraging Bayesian networks to help in the diagnostics of these systems during failures to considerably shorten the time to localize the cause of Service Level Objectives violations. The model subsequently reduces the violation duration by reducing the Mean Time To Resolution.**

Keywords–Bayesian network, Machine Learning, Cloud Computing, Auto scaling, Service Level Objective, Availability.

I. INTRODUCTION

The move of modern software to the cloud has been increasing over the past decade, and organizations are migrating more distributed systems to execute in the cloud environment. One of the reasons for this migration is the advanced custom auto-scaling abilities [1] provided by the cloud vendors. While deploying distributed systems has become a lot easier, improvements like these make it radically different from the older model of deploying systems on a mostly static infrastructure and introduces its own set of challenges.

However, businesses must continue to be actively mindful of the availability that their users expect. Most organizations design deployments around a set of metrics known as Service Level Objectives (SLOs). We define SLOs in terms of performance, reliability, and availability of the application and quantify the SLOs in metrics such as downtime, error rates, end-to-end request latencies, etc. An example latency metric would be to expect an average of 200ms response time over 5 minutes for a server side HTTP application. Exceeding this threshold would be considered an SLO violation. The expectations for a well-engineered application is high availability, i.e., infrequent SLO violations. This infrequency, and many metrics that are recorded for each system make it especially complicated to detect, localize and fix the system during a violation. This complication can result in the Mean Time To Resolution (MTTR) being unacceptable to the stakeholders of the system.

This paper focuses on the automated localization of the problem in a distributed system with each service leveraging shared infrastructure, such as network equipment, resource capacity, and even a shared database. We assume that an issue has been detected in at least one part of the distributed system. We do not specifically attempt to surface the root cause of the problem. However, we expect that by localizing the problem automatically, the MTTR decreases significantly. The decrease comes by allowing further human intervention to determine the root cause faster. In the proposed strategy, we leverage Bayesian networks and as a custom reactive probing framework that observes the state of a subset of previously hidden nodes in the Bayesian network.

The paper's organization is as follows: Section II describes the related work and the implementations of current methods. In Section III, we provide a motivating example where this application is useful. Section IV details the underlying framework and the methodology, along with the results, while Section V concludes and describes future work.

II. RELATED RESEARCH

There is an existing body of literature that tackles the problem of automated diagnosis of SLO violations in distributed systems, which broadly categorizes the diagnosis into two parts. The first part is localizing the issue to a specific subset of the system. Zhang, et al. [2] focus on response time problems caused by abnormally slow services, and use Bayesian networks to diagnose the issues. This approach's primary focus is using the response time of individual observed services and total end-to-end response time to infer time taken by unobserved (uninstrumented) services. A limitation of this model is that the localization's granularity is only up to a specific service, which itself could be a complex system and hard to debug. The research assumes that parts of the system which are not instrumented to report SLO violations of their own. Our research will aim at yielding a more granular diagnosis by introspecting services and their dependencies.

Cohen et al. [3], attempt to correlate system metrics in a distributed system with the SLO violations. They explicitly do not use application metrics, focusing instead on system-level metrics from the server such as CPU time in user mode, disk read frequency, etc. where each metric they use is specific to the system of a particular application, enabling a more granular localization of the problem. However, they require training the classifier on past data which is hard to come by since SLO violations are infrequent in a well-engineered system. Our research leverages Bayesian networks, where the prior probabilities are calibrated by a domain expert who has access to past data. Furthermore, the study mentioned above does not consider the cloud platform, which can be responsible for a separate class of SLO violations related to newer features they provide.

 ISBN: 978-1-61208-778-8

The second part of the problem is root cause analysis, which can be computationally intensive and therefore is not viable for large volumes of data or compromises accuracy due to many metrics and a large number of data points for the metrics. Natu et al. [4] apply feature selection to prune the search space of irrelevant and redundant metrics. Our approach does not attempt to prune the search space but does try and glean as much information as possible from the available metrics while maintaining focus on solving the first part of the problem.

III. Motivating Example

When an error occurs in a dynamically scaled distributed system SLO violations are often caused by time spent collecting the information needed to understand the root cause. In the case of the services we are researching, we consider a response latency over 400ms to be an SLO violation as this has been demonstrated to cause user impact.

We measure this SLO on a subset of the distributed system that corresponds to synchronous operations that directly impact the "real-time" user experience. We do not consider scheduled or deferred jobs in this. For example, we observe the time to load a specific URL on the website, or a view in the mobile application, but we do not consider a violation of a queued email sending job.

The following graph shows the full duration of SLO violations by time period that have been recorded in official postmortems in the company:

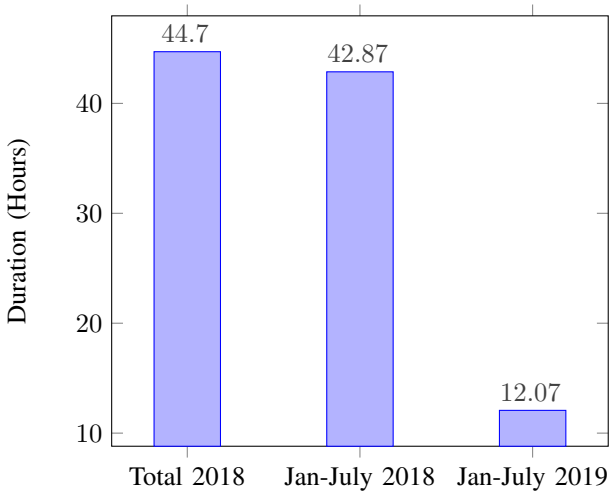

Figure 1. SLO Violations in Hours.

In particular, there was one incident where the cloud provider the company infrastructure is deployed on was facing issues in one subnet, which prevented scaling new servers to accommodate user load. It took 6 hours to mitigate user impact, 2.5 hours (˜42%) of which was spent in localizing the issue to that particular subnet. In this incident, we were alerted that we had scaling issues, without further specificity.

IV. Methodology and Empirical Evidence

To form the model, one should reason about what the SLO violation in such a distributed systems setup in the cloud could be caused by:

- *Database issues*: If the database was under load, e.g. due to too many queries per second, or other availability concerns.
- *Application Errors*: If there were application errors due to a bug or application level dependency issues.
- *Resource starvation*: If the servers were being limited by CPU, memory, network, etc.
- *Bad deployment*: If there was an issue with the deployment process itself.

Similarly, resource starvation can be caused by buggy code or scaling issues. Here, the buggy code would likely be unrelated to the business logic of the application. An example of such a problem one might encounter is a memory leak by the application not correctly freeing the memory allocated for an operation. In a database-backed application, as we are examining, this can be discovered in a bug in the database access layer where too many connections are open, tying up the resources of both the application and the database server.

Scaling issues are best described as the service's inability to receive more capacity, despite the metrics indicating a need for this extra provisioning. An example here would be when a service exceeds the aggregate CPU utilization threshold over a cluster of hosts for a service and triggers the cloud configuration scaling but is denied extra capacity.

Furthermore, resource starvation can be caused by buggy code or scaling issues, and scaling issues can be caused by:

- *Cloud limits*: If the cloud resource limits set by an agreement with the company and the cloud vendors was hit.
- *Recent configuration change in the infrastructure*: If a potential new bug was introduced.
- *Infrastructure or external dependency issue*: If there was an issue with other services or infrastructure that we depend on for scaling up.

With this in mind, a causal, directed acyclic graph on which the Bayesian network would be built was created to reflect the infrastructure:

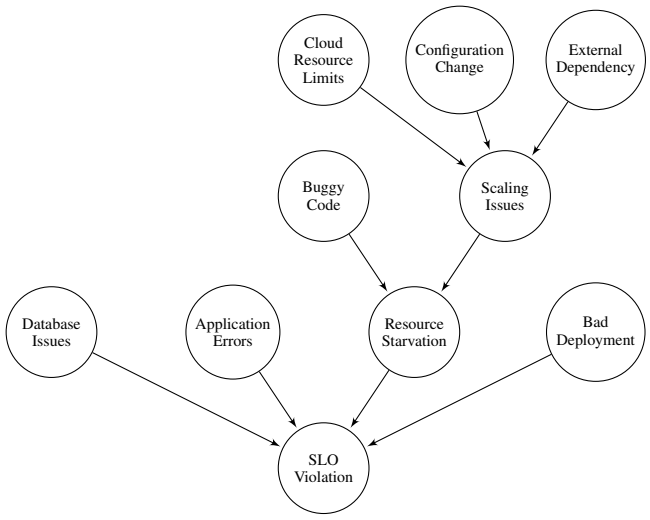

Figure 2. Causal, Directed Acyclic Graph for the Bayesian Network

The model's implementation used the pomegranate python library [5], and a series of implemented checks to make some of the nodes "observed". The checks linked into various monitoring systems and ruled out cloud limits, recent configuration changes, database issues, and application errors. This was determined by leveraging APIs (Application Programming Interfaces, in this case over the HTTPS protocol) of the monitoring systems and comparing with the conditions for the nodes in the Bayesian network, e.g., whether a deployment occurred in a period of time that correlates with the timeline of the incident.

The nodes for which information is not available are known as unobserved nodes and form the crux of the model. With the help of the pomegranate library, the output of the model results in the unobserved nodes of the Bayesian network being associated with their updated probabilities, given the information gathered from the monitoring systems, i.e., the observed nodes. These calculated probability values show with reasonable certainty that the issue was an external dependency or infrastructure issue and that deploy issues and errors are unlikely. This also indicates that there is a high chance the alerted scaling issues are causing an SLO violation. The specific probabilities generated with the above mentioned methodology associated with the nodes in the Bayesian network can be seen in Table I.

TABLE I. CALCULATED PROBABILITIES OF OUTCOMES

Node Name	*P(True)*	*P(False)*
Resource Starvation	0.9899687033177891	0.10031296682210913
SLO	0.9810724570630289	0.018927542936970975
External Dependencies	0.6070287539936117	0.39297124600638844
Bad Deployment	0.10035812797969593	0.8996418720203041
Application Errors	0.09677684818274046	0.9032231518172594

V. CONCLUSION AND FUTURE WORK

In this paper, we addressed the problem of minimizing SLO violations in an organization's infrastructure. We argued that using Bayesian networks and leveraging past data to assist with localizing of the problem can drastically reduce the Mean Time To Resolution of incidents. In this paper, we addressed this issue of minimizing SLO violations by designing a Bayesian network that incorporates causal relations and is initialized by a subject matter expert leveraging past data and experience with the system. We demonstrated in a specific type of incident that the model could correctly determine the cause and provide alternative paths in decreasing order of likelihood of occurrence.

In the future, we will prove the model can be generalized across a variety of incidents, and not just the specific motivating example in this paper. Furthermore, the model should be able to update itself with new data over time, so the relevance of the prior probabilities defined by a subject matter expert will decrease. Eventually, a pluggable architecture can be provided where the prior probabilities can be generated by automation leveraging historical data in the various monitoring systems.

REFERENCES

[1] "Big Day for Amazon EC2: Production, SLA, Windows, and 4 New Capabilities," 2008, URL: https://aws.amazon.com/blogs/aws/big-day-for-ec2/ [accessed: 2020-03-03].

[2] R. Zhang, S. Moyle, S. Mckeever, and A. Bivens, "Performance problem localization in self-healing, service-oriented systems using bayesian networks abstract," in *Proceedings of the ACM Symposium on Applied Computing*, 01 2007, pp. 104–109.

[3] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons, "Correlating instrumentation data to system states: A building block for automated diagnosis and control." in *Proceedings of the 6th conference on Symposium on Operating Systems Design —& Implementation - Volume 6*, 01 2004, pp. 231–244.

[4] M. Natu, S. Patil, V. Paithankar sadaphal, and H. Vin, "Automated debugging of slo violations in enterprise systems," in *2010 2nd International Conference on COMmunication Systems and NETworks, COMSNETS 2010*, 02 2010, pp. 1 – 10.

[5] J. Schreiber, "pomegranate," 2016, URL: https://github.com/jmschrei/pomegranate/ [accessed: 2020-03-03].

 ISBN: 978-1-61208-778-8