Toward Automated Testing of Geo-Distributed Replica Selection Algorithms *

Kirill Bogdanov
KTH Royal Institute of Technology
kirillb@kth.se

Miguel Peón-Quirós
University Complutense of Madrid (UCM)
Spain
mikepeon@gmail.com

Gerald Q. Maguire Jr.
KTH Royal Institute of Technology
maguire@kth.se

Dejan Kostić
KTH Royal Institute of Technology
dmk@kth.se

ABSTRACT

Many geo-distributed systems rely on a replica selection algorithms to communicate with the closest set of replicas. Unfortunately, the bursty nature of the Internet traffic and ever changing network conditions present a problem in identifying the best choices of replicas. Suboptimal replica choices result in increased response latency and reduced system performance. In this work we present GeoPerf, a tool that tries to automate testing of geo-distributed replica selection algorithms. We used GeoPerf to test Cassandra and MongoDB, two popular data stores, and found bugs in each of these systems.

CCS Concepts

•**Software and its engineering** → *Software testing and debugging;* •**Networks** → *Wide area networks;*

Keywords

Symbolic Execution, Replica Selection Algorithms

1. INTRODUCTION

Today, many popular services are deployed in cloud environments such as Amazon EC2[1], Microsoft Azure[3], and Google Cloud[2]. These services are used by hundreds of millions of users who are spread across the globe. For reasons of performance, reliability, and survivability many such services are replicated across geo-distributed datacenters. Ensuring low latency response time in such an environment is both highly important and difficult [8, 7]. To achieve lower response times, clients of these services often use *replica selection algorithms* to choose a replica or set of replicas they will contact.

Unfortunately, Internet traffic is bursty and routing can change. This directly affects end-to-end characteristics of a wide-area network paths such as latency, bandwidth and packet loss rate. Therefore, in order to operate in such conditions, replica selection algorithms need to continuously measure the networks state and use past history in order to anticipate future changes. All of these factors make replica selection very hard.

To better understand the underlying conditions, we performed detailed measurements of application-level round-trip latencies among all the Amazon EC2 regions. Our measurements demonstrate high variability in network conditions over time. Figure 1 shows round-trip measurements over a period of one day from the Ireland EC2 datacenter to datacenters in all other EC2 regions. From the point of view of Ireland, the set of nearest replicas based on the network latency change. Moreover, this demonstrates the need for dynamic adaptation to ever-changing conditions, as despite running over a dedicated infrastructure, any static configuration of replicas would be suboptimal.

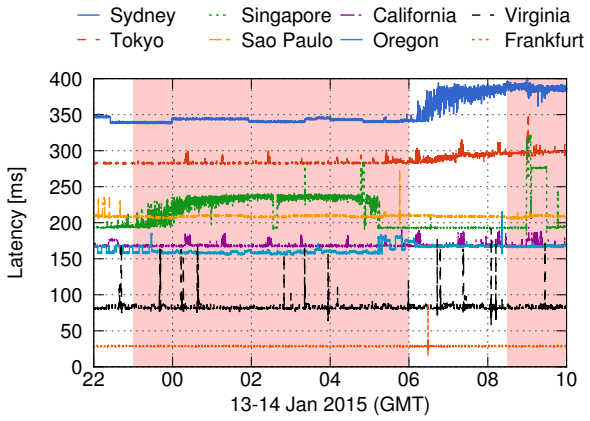

Figure 1: RTT measurement over UDP from the Ireland EC2 datacenter to 8 other EC2 regions (after low pass filter). Red background highlights the time of the day when the order of replicas change (Singapore-São Paulo).

*Work funded by ERC project 259110. Miguel performed his work while working at IMDEA Networks.

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.

DOI: http://dx.doi.org/10.1145/2785956.2790013

It is extremely difficult to identify errors in replica selection algorithms. First, such errors usually do not lead to critical system failures. Second, often it is hard to determine the optimal choices of replicas in the absence of up-to-date full global knowledge. Debugging is difficult due to the number of potential bug causes, such as problems in latency sampling, math calculations, replica selection logic, etc.

All of these factors significantly complicate testing of replica selection algorithms. To perform comprehensive testing would require trying numerous network topologies with variable latency, bandwidth, and loss rate characteristics. Moreover, one would need to predict duration, intensity, and frequency of changes of these network characteristics. Therefore, it is unlikely that simple unit testing would be able to uncover all possible errors. Instead, we need a *systematic testing tool*.

Next, we describe GeoPerf, our tool for automating the process of testing replica selection algorithms. We believe that it is the first such tool.

2. GEOPERF

We choose to apply symbolic execution [6], because it performs systematic code path exploration. It uses an automatic constraint solver to derive concrete input values that will cause a particular code path within a program to be executed. As an input we use network latencies that could be observed by the replica selection algorithm under test, while code paths represent different replica decisions made by the algorithm.

Each replica choice is defined by the set of different code paths that can lead to that choice. Under normal execution a single code path is determined by the system state (i.e., past and current set of network RTTs, time, random numbers) and the implementation of the replica selection algorithm itself. However, by using symbolic execution we can explore alternative code paths that lead to different replica choices given the symbolic state of the system. This allow us to generate latency inputs that could force a replica selection algorithm to choose one or another subset of nodes. However, to reason about correctness of such choices we need to know the ground truth. For this purpose, GeoPerf provides a simplified replica selection mechanism that represents the minimum bound on the achievable latency. This algorithm act as a comparison point to the target algorithm and allows us to evaluate individual replica choices.

First we develop a controlled environment where we can generate, monitor, and replay events deterministically. Second, we isolate target replica selection algorithm from the system under test. Then we use our controlled environment to compare two replica selection algorithms while both algorithms share one view of the network. GeoPerf uses symbolic execution to determine if two algorithms make different choices under identical network conditions.

GeoPerf is based on our own discrete event-based simulator, which contain a set of geo-distributed nodes connected via wide-area network paths with variable latency. We simulate arrival of client requests and use replica selection module, first to monitor the state of network paths and then to periodically choose a set of nodes to serve incoming requests. We use GeoPerf to create two instantiations of the simulator and run them in parallel in the identical environments with synchronized clocks, and deterministic pseudo-random number generators. One simulation is configured to use the reference replica selection algorithm and the other the algorithm under test.

We use the symbolic execution engine to explore possible code paths and to generate latency inputs that define the network conditions among the nodes. The purpose of this exploration is to identify a sequence of latency values that can expose a weakness in the target replica selection algorithm. This is done by demonstrating that the target algorithm repeatedly exhibits inferior performance (i.e., choices) in the simulated environment.

3. EVALUATION

Using GeoPerf we found bugs in replica selection algorithm of two popular geo-distributed data stores, Cassandra[5] and MongoDB[4]. In addition, we quantify the impact of the bugs that GeoPerf found. Specifically, we replay the trace of the latencies we collected across Amazon EC2 using GeoPerf's event simulator, and compute the median time that is wasted due to the bugs. Figure 2 shows the comparison between the buggy and fixed versions of Cassandra's replica selection algorithms, it demonstrates that over 20% of all requests were affected by the bug. The median loss for 10% of all requests is above 50 ms.

A video clip showcasing the work: http://prophet.ssvl.kth.se/wp-content/uploads/2015/05/Kirill-src.mp4

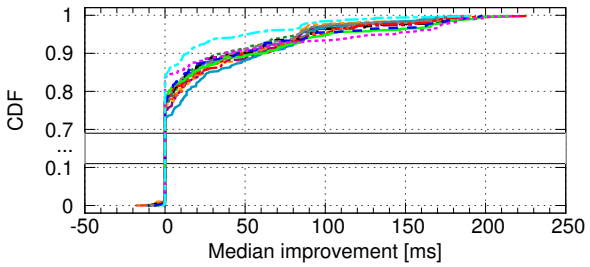

Figure 2: The CDFs of medians of the requests completion time differences between buggy and fixed versions of the Cassandra's replica selection algorithms (EC2 latency trace replay via GeoPerf). The figure contains 14 CDFs, one for each day of the trace of latency samples.

4. REFERENCES

[1] Amazon ec2. http://aws.amazon.com/ec2/.

[2] Google cloud. https://cloud.google.com/.

[3] Microsoft azure. http://azure.microsoft.com/.

[4] Mongodb. http://www.mongodb.org/.

[5] Apache. Cassandra. http://cassandra.apache.org/.

[6] Cadar, C., Dunbar, D., and Engler, D. R. KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs. In *OSDI* (2008).

[7] Dean, J., and Barroso, L. A. The tail at scale. *Commun. ACM 56*, 2 (Feb. 2013), 74–80.

[8] Decandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. Dynamo: Amazon's Highly Available Key-value Store. In *SOSP* (2007).