Observing the Consistency of Distributed Systems

Deepthi Devaki Akkoorath
University of Kaiserslautern, Germany
akkoorath@cs.uni-kl.de

Viktória Fördős
Erlang Solutions Ltd, Hungary
viktoria.fordos@erlang-solutions.com

Annette Bieniusa
University of Kaiserslautern, Germany
bieniusa@cs.uni-kl.de

Abstract

In distributed Erlang systems temporary network issues are very likely; and can affect the consistency of the system. If the system is a distributed data store relying on a weak consistency model, its data stored may diverge as a result of connectivity issues. Even worse, there is no indicator of the divergence. In this paper we present our initial work on divergence metrics and give a preliminary evaluation on exposing divergence to the system operator.

Categories and Subject Descriptors H.3.4 [*Systems and Software*]: Distributed systems

Keywords Data store, weak consistency, divergence metrics

1. Introduction

Antidote[1] is an open-source platform written in Erlang/OTP. It allows implementing highly-scalable distributed key-value stores that feature conflict-free replicated data types (CRDTs)[6]. Antidote supports the experimental implementation for inter- and intra- data center data replication, and provides benchmarks for comparable evaluation of distributed consistency protocols. For example, the Cure protocol [1] provides atomic multi-key writes, snapshot reads, and data partitioning. Antidote offers different types of consistency, ranging from strong consistency to eventual consistency.

Under strong consistency, updates are forwarded to and acknowledged by all replicating nodes before returning to the caller. Therefore, updates can be totally ordered, and all nodes have the same system state. Under weaker notions of consistency, updates are accepted by a subset of the replicating nodes and only later forwarded to the other nodes in the systems. While this enables partition-tolerance and high throughput, there is a delay until updates are visible in other replicas. This results in a short period where the states diverge on the replicas. The applications that are built on weakly consistent data stores may tolerate small divergence. But large divergence might yield to unintuitive results. Monitoring divergence helps to spot potential issues related to replication protocols, e.g. due to a high replication factor or under increasing load.

[1] `https://github.com/SyncFree/antidote`

Erlang'16, September 23, 2016, Nara, Japan
© 2016 ACM. 978-1-4503-4431-9/16/09...
http://dx.doi.org/10.1145/2975969.2975975

WombatOAM[2] [2] is an operation and maintenance tool for systems running on the BEAM virtual machine. It detects anomalies and early warning signs, and presents them in the form of alarms, so that the problems can be resolved before they cause service disruption. It provides the Erlang centric view of the system, full visibility of the historical metrics and notifications, live information about running systems and the drive to use this information to troubleshoot issues during and after outages. The heart of WombatOAM are the WombatOAM plugins that run on the managed node and give insight into popular Erlang & Elixir applications.

Contributions In this paper we present our initial work on measuring divergence of data stored by the Antidote nodes. We describe a divergence metric, and discuss how the metric characterizes the current divergence of a replicated system. We extended WombatOAM with an Antidote plugin for divergence measurement and conducted some preliminary experiments. The evaluation provides a fine-grained insight into how the Antidote nodes behaved under varying load.

2. Measuring divergence

Divergence is a property of the data store that indicates how different the value of an object in a replica is compared to that in other replicas. Given a global view of all updates on an object at all replicas until a given time, the divergence of the object replica can be quantified as the number of missing updates in the replica. However, in order to measure it, we need to have the complete information about all updates in all replicas, which requires synchronization. What we are interested in, is to measure the divergence locally in a replica with the limited information it has about other replicas.

We, therefore, propose another metric: *staleness*. Staleness indicates how "old" the information received from other replicas is. It is measured as the difference between the observed timestamps from other replicas and the local timestamp. The difference indicates potentially missing updates with a timestamp between the observed and the local time. We can measure only *potential* staleness this way, because it is not possible to know whether there is actually a missing update or not. Nevertheless, it is feasible in many systems to assign timestamps from a loosely synchronized clock such as using NTP [5], and keep track of the time when it was last synchronized.

We measure divergence for each Antidote data center (DC) using the metric potential staleness. For each DC, there is a *global stable time* which is a vector that denotes the version of a data which is read by any transaction initiated at that time. Each entry of the vector denotes the timestamp last observed from the corresponding DC. It is updated when it receives an update or a heartbeat (to indicate there were no updates) from another DC [1]. The staleness

[2] `https://www.erlang-solutions.com/products/wombat-oam.html`

is then calculated as the difference between the current local timestamp and the minimum timestamp entry in the global stable time vector. The value of staleness has a lower bound bounded by the network delay between DCs.

We wanted to minimize the negative impact put on Antidote that can be caused by observing the divergence [4]. Thus, we a) separated the divergence calculation into a new Erlang process and b) expose the measured divergence via an Exometer [3] histogram. As a result, the calculation remains private and under the control of Antidote that cannot be overused by third party tool. Therefore, it does not put a visible impact on the core of Antidote. Also, third party tools can retrieve the value of the Exometer metric at any rate, since this has zero impact on Antidote.

2.1 Monitoring using WombatOAM

To have a better insight into the Antidote nodes, we developed a new WombatOAM plugin for our experiments. Besides automatically collecting Exometer metrics, the plugin exposes the two most important divergence metrics from operation's perspective: the maximum and the median values of the potential staleness exposed by the Antidote nodes. Based on the historical metrics, it can then be analyzed how the system behaved during peaks.

When the divergence related to an Antidote cluster grows too much, the system may not be able to function properly causing partial service disruptions or a major outage. To prevent the financial loss and reputation damage, the operation team wants to resolve such incidents as soon as possible. Here, the plugin can warn the operation team about the anomaly before the issue escalates. As the very early warning sign, it raises an alarm when the maximal potential staleness captured on the Antidote nodes raises above the limit the system can tolerate. This situation may not result in an outage, but it shows that the system is not perfectly balanced. On the contrary, when the medians are above this limit, service disruptions are very likely to occur. At that time another alarm will be raised by the plugin indicating the last chance of the operation team to resume their system back to normal.

3. Evaluation

We deployed Antidote nodes and WombatOAM in Grid'5000 to separate machines. The setup consists of 2 DCs, each hosting 8 Antidote nodes. Each DC has a full replica of the store. The objects are distributed among the nodes in a DC. Load was generated using various number of client threads (16, 32, 64) changed in every 10 minutes. 1:1 read:write operations were performed for the first 30 minutes and 3:1 read:write operations for the second 30 minutes.

By studying the data collected by WombatOAM (at a sampling frequency of 1 min), we made the following observations. First, both the medians (Figure 1) and the maxima (Figure 2) measured on the nodes at the same time are almost equal. No serious outlier appears, which implies that the nodes behave the same. The average median is 90 ms while the average maximum is 290 ms. Second, the peaks in the maxima always happen before the peaks in the medians. Third, we observed that the maxima correlate with the sum message queue lengths of all Erlang processes running on the Antidote nodes (Figure 3). Considering that the changes in the message queue lengths are the direct effect of the varying load put on the Antidote nodes, we can conclude that the load affects the potential staleness measured on the Antidote nodes.

4. Outlook

We are planing to investigate more metrics measuring the divergence of data stored by the Antidote nodes. Next, we will move our focus to other distributed data stores, such as Riak or Mnesia, trying to generalize and to apply our metrics to other vertices.

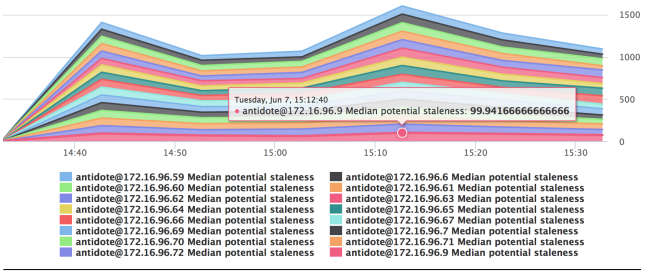

Figure 1. Stacked graph showing median potential staleness.

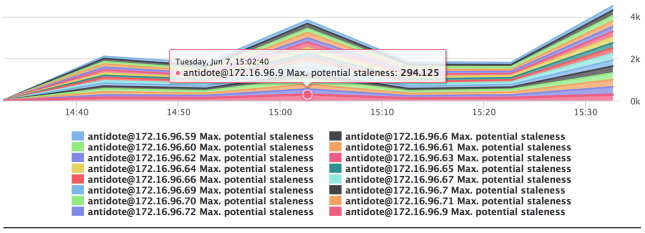

Figure 2. Stacked graph showing maximum potential staleness.

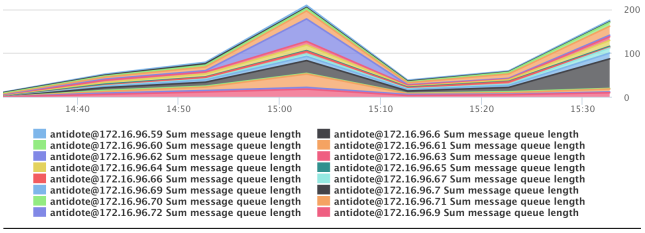

Figure 3. Stacked graph showing sum message queue lengths.

Acknowledgments

This research is supported in part by European FP7 project 609551 SyncFree (20132016). Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] D. Akkoorath, A. Z. Tomsic, M. Bravo, Z. Li, T. Crain, A. Bieniusa, N. Preguica, and M. Shapiro. Cure: Strong semantics meets high availability and low latency. In *Proceedings of 36th IEEE International Conference on Distributed Computing Systems, ICDCS*, June 2016.

[2] N. Chechina, K. MacKenzie, S. Thompson, P. Trinder, O. Boudeville, V. Fördős, A. Ghaffari, C. Hoch, and M. M. Hernandez. Evaluating Scalable Distributed Erlang for Scalability and Reliability. *Under submission*, April 2016.

[3] Feuerlabs. Exometer - Erlang instrumentation package. `https://github.com/Feuerlabs/exometer`, 2016. Accessed 4 June 2016.

[4] T. Mytkowicz, P. Sweeny, M. Hauswirth, and A. Diwan. Observer Effect and Measurement Bias in Performance Analysis. Computer Science Technical Reports CU-CS-1042-08, University of Colorado, Boulder, 2008.

[5] NTP. The network time protocol. `http://www.ntp.org`, retrieved Jun 2016.

[6] M. Shapiro, N. M. Preguiça, C. Baquero, and M. Zawirski. Convergent and commutative replicated data types. *Bulletin of the EATCS*, 104: 67–88, 2011.