TEST Engineering & Management

May-June 2020
ISSN: 0193-4120 Page No. 4657-4660

Server Monitoring Using RAFT Algorithm

[1]Aman Srivastava, [2]Asad Ahmad, [3]Akash Hadagali, [4]Thirumagal E
[1,2,3]BTech, [4]Professor, C&IT, Reva University, Bengaluru

Article Info
Volume 83
Page Number: 4657-4660
Publication Issue:
May-June 2020

Article History
Article Received:*19 November 2019*
Revised: *27 January 2020*
Accepted: *24 February 2020*
Publication: *12 May 2020*

Abstract

Raft is a sophisticated algorithm that is majorly used for leader election and log replication across the servers in distributed system to achieve the same state and fault tolerance. Currently, raft algorithm is used for server health checks in distributed systems but not for monitoring. Beside only sending the health status we are sending the CPU and memory utilization of the server. Raft isolates the key components of accord, for example, pioneer political decision, log replication, and security, and it upholds a more grounded level of coherency to diminish the quantity of states that must be thought of. Results from a client study exhibit that Raft is simpler for understudies to learn than Paxos. Raft likewise incorporates another instrument for evolving the bunch participation, which uses covering greater parts to ensure wellbeing.

Keywords: *Raft, Server monitoring, Log replication, distributed system.*

1. Introduction

RAFT is an algorithm that is intended to be straightforward. It's proportionated to Paxos in adaptation to non-critical failure and execution. The thing that matters is that it's disintegrated into moderately autonomous subproblems, and it neatly addresses every single significant piece required for down to earth frameworks. We trust Raft will make agreement accessible to a more extensive crowd, and that this more extensive crowd will have the option to build up an assortment of greater accord-based frameworks than are accessible today.

RAFT is comparable from various perspectives to existing consensus algorithms, yet it has a few novel highlights:

• Strong Leader: Raft utilizes a more grounded type of initiative than different accord calculations. For instance, log passages just stream from the pioneer to different servers.

This streamlines the administration of the repeated log also, makes Raft more obvious.

• Leader political race: Raft utilizes randomized clocks to choose pioneers. This includes just a limited quantity of instrument to the pulses previously required for any agreement calculation, while settling clashes essentially and quickly.

• Membership changes: Raft's instrument for changing the arrangement of servers in the group utilizes another joint accord approach where the larger parts of two unique arrangements cover during advances. This permits the bunch to keep working ordinarily during design changes

Ease of use:

This Application of server health check provides the administrator the benefit to monitor performance of server and its utilization from his phone at any time. The App related to it can be easily installed on any android or ios device from where this real time monitoring can be done anytime. The app just has a single page interface which shows all the different instances of the server along with level of their utilization with the help of very good animation. The level of CPU and memory utilization of all these server nodes are also shown.

This Application also has a elaborative desktop variant as well. This type of GUI based implementation makes the work of monitoring and health check very easy and fast and helps in reducing the downtime of the server.

Performance:

As the application is totally based on raft implementation it has similar performance as that of other algorithms like paxos. Here the elected leader has full power for log replication. Here in Raft full control of log entries are with leader, so it can be said that log's move in only one direction in this application. Due to the availability of mobile application the speed of getting notified related to any change in the health status of the server is increased which leads to low downtime of the server and help the organization to provide better user experience

May-June 2020
ISSN: 0193-4120 Page No. 4657-4660

RAFT has less message types than some other calculation for accord-based log replication that we are mindful of. For instance, we tallied the message types VR what's more, Zookeeper use for fundamental accord and enrollment changes (barring log compaction and customer connection, as these are about autonomous of the calculations). VR what's more, Zookeeper each characterize 10 distinctive message types, while Raft has just 4 message types (two RPC demands furthermore, their reactions). Pontoon's messages are more thick than different calculations', however they are easier aggregately. Also, VR and Zookeeper are depicted in terms of transmitting whole logs during pioneer changes extra message types will be required to streamline these systems with the goal that they are down to earth

2. Related Works

Currently RAFT is being used in CONSUL which runs on a minimum of three servers in server/client style approach. The three servers are used for quorum however in a home environment we can run it on just one.

Prometheus is another open source tool which uses RAFT as a monitoring system. It has a number of HTTP API's with the help of which it can request unprocessed data and evaluate PromQL queries.

3. Existing Works

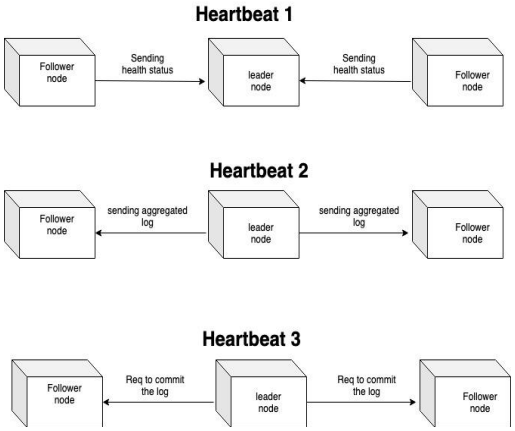
Right now, it is being utilized by two open-source devices CONSUL and Prometheus. In CONSUL the operators register on the servers in the wake of giving a rundown of administrations that are running on it. It utilizes three servers for a majority. Prometheus is a measurement based observing framework that is intended to monitor the general framework conduct and health. CONSUL runs on at least three servers in server/customer style approach. The three servers are utilized for majority anyway in a home domain we can run it on only one. Prometheus uses RAFT as an observing system. It has various HTTP API's with the assistance of which it can demand natural information and assess PromQL questions.

4. Proposed Works

Objectives:

1. To help the companies to keep a track of their servers and the applications running on it.
2. To send alert to the system admin in case of an unhealthy server
3. To monitor the web servers for security, speed as well as user load.
4. To protect the server from a possible failure.
5. To monitor the performance and operations of components and equipment at granular level including CPU and memory utilization.
6. Develop test cases that capture all potential server failures.

Block diagram:



Algorithm:

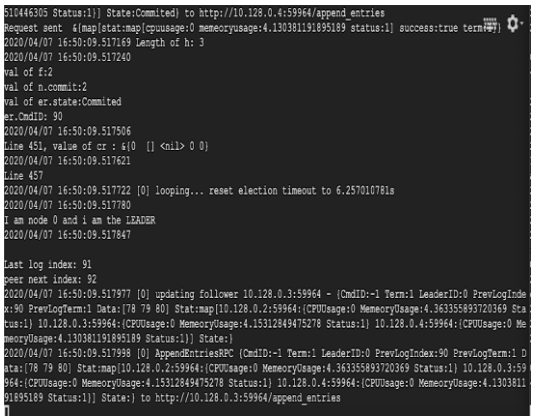
RAFT is the primary algorithm used by this server monitoring software. It is used mainly because it governs the Consensus taking place between different server instances.

RAFT implement's consensus by first electing a leader by-election and then giving the leader the complete responsibility for managing the log. A new leader is chosen every time when the existing one fails. Generally, a raft cluster has multiple server instances that allow the system to tolerate failure till the majority of the instances are up and working. Out of these multiple instances, only one is elected as a leader while others behave as passive members. RAFT algorithm is a successor of the Paxos consensus Algorithm which was very difficult to implement so RAFT replaced it. The Leader node has complete responsibility of log management in the system. Whenever there is a problem on the leader node a new election takes place and a new leader is chosen all together.

A single leader selection increases the speed of log entries, As there is only one entity to to make choices related to main system functionality.

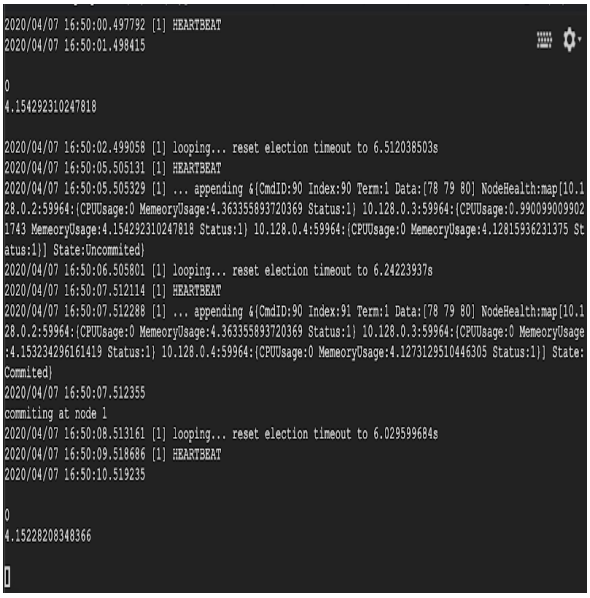
5. Experimental Results and Analysis

Analysis (running raft on cluster of three nodes)
Leader node

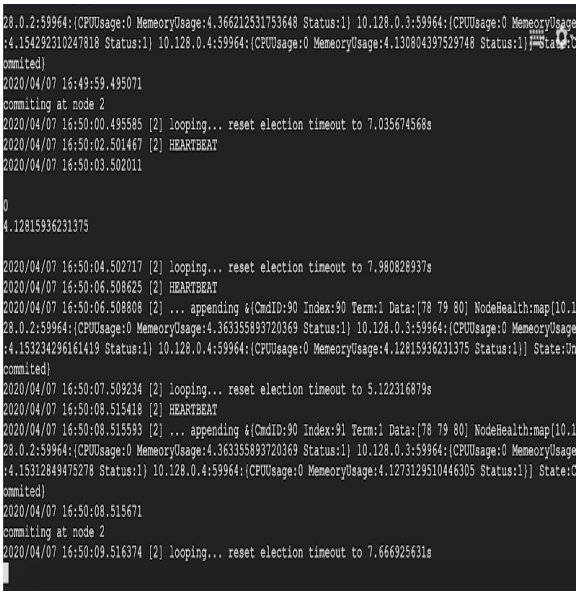


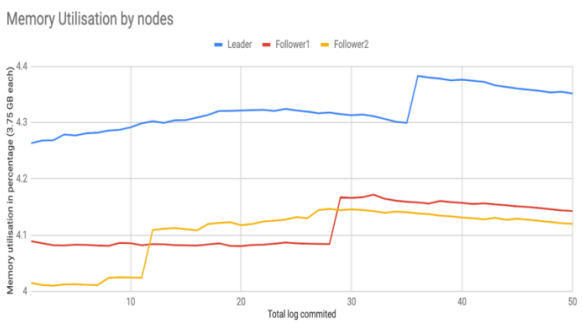
May-June 2020
ISSN: 0193-4120 Page No. 4657-4660

Follower node1



Follower node2





The memory gets piled up with logs but can be flushed once the logs are committed but if there exist more servers in a system then they are required to have more memory. As shown above the leader node will require more memory since all the logs are aggregated by the leader node.

6. Conclusion

Calculations are frequently planned with rightness, proficiency, or potentially compactness as the essential objectives. Although these are for the most part commendable objectives, we accept that understandability is similarly as significant. None of different objectives can be accomplished until engineers render the calculation into a commonsense usage, which will go amiss from what's more, develop the distributed structure. Except if engineers have a profound comprehension of the calculation and can make instincts about it, it will be hard for them to hold its attractive properties in their execution. Right now tended to the issue of disseminated agreement, where a broadly acknowledged however invulnerable calculation, Paxos, has tested understudies and engineers for numerous years. We built up another calculation, dependent on Raft and helps in server monitoring which we have demonstrated to be more justifiable than using Paxos. We additionally accept that Raft gives a superior establishment for framework building. Utilizing understandability as the essential plan objective changed the manner in which we moved toward the structure of Raft; as the structure advanced we got ourselves reusing a couple of systems more than once, for example, deteriorating the issue and streamlining the state space. These procedures not just improved the understandability of Raft yet in addition made it simpler to persuade ourselves regarding its accuracy.

7. Acknowledgement

Our appreciation goes to Prof.Thirumagal E, Professor of computer Sciences, the university of Reva for her strong ongoing support for the research, so that, amid a maze of possibilities and Shortcomings, we can achieve a planned objective.

References

[1] Ivan Loire – A simple node.js service monitor “https://github.com/iloire/watchmen” Last commit 31 Mar 2020

[2] Diego Ongaro and John Ousterhout of Stanford University - In Search of an Understandable Consensus Algorithm “https://raft.github.io/raft.pdf ”, July 2014

[3] Heidi Howard, Malte Schwarzkopf,Anil Madhavapeddy, and Jon Crowcroft SIGOPS Operating Systems Review, January 2015.

[4] Heidi Howard, University of Cambridge, Computer Laboratory, UCAM-CL-TR-857, July 2014.

[5] Diego Ongaro of Stanford LogCabin source code “http://github.com/ logcabin/logcabin.”

[6] JUNQUEIRA, F. P., REED, B. C., AND

May-June 2020
ISSN: 0193-4120 Page No. 4657-4660

SERAFINI, M. Zab,"High-performance broadcast for primary-backup systems". In Proc. DSN'11, IEEE/IFIP Int'l Conf. on Dependable Systems & Networks (2011), IEEE Computer Society, pp. 245–256.

[7] Raft consensus algorithm website. http://raftconsensus.github.io

[8] MAZIERES D. Paxos made practical. "http: //www.scs.stanford.edu/dm/home/ papers/paxos.pdf", Jan. 2007.