Conditional Name Server Identifier

- CoreDNS

Project Proposal for Google Summer of Code 2018

1. Student Information

Background

Name: Jiacheng Xu
University: Swiss Federal Institute of Technology in Lausanne (EPFL)
Major: Master student in Computational Science and Engineering

Related experience:
Reconfiguration, Security and Testing in a Ring Replication Protocol. Sep. 2017 - Present
Supervised by Prof. Rachid GUERRAOUI EPFL, Distributed Programming Laboratory

- Current work: Writing unit and integrated tests to increase the system reliability.
- Implemented a signature protocol by applying DSA to guarantee the message security.
- Accommodated the reconfiguration by modifying the common case protocol of the system.
- Implemented a reconfiguration protocol based on a conventional all-to-all broadcast primitive.

Contact Information

Email: jiacheng.xu@epfl.ch
Phone: +41 786642289
Github: https://github.com/JiachengXu

2. Introduction

In distributed TensorFlow, identifying the nodes without domain name collision is a big challenge. CoreDNS supports DNS Name Server Identifier (NSID) which allow a DNS server to identify itself. So we can deploy CoreDNS for every node in the distributed TensorFlow cluster to solve this problem. There are two ways to achieve this goal. One way is to set up a distributed Key-Value store like zookeeper or etcd, and another way is to assign each node with an order based on the timestamp.

My GSoc work aims to implement one of the approaches above.

3. Deliverables

Below I structure the contributions to this project to several core parts:

Distributed K-V store approach:

- Set up an etcd or zookeeper for distributed store domain name.
- Integration with CoreDNS.

Timestamp-based approach:

- Design of Self-identify protocol based on timestamp.
- Self-identify protocol implementation code.
- Integration with CoreDNS.

For both approaches:

- Unit tests.
- Documentation.

I listed deliverables of two possible implementations as above, and I prefer to implement the second one which is based on the timestamp of the cluster (the more detailed reason is in the “Implementation” part). And for both approaches, I would apply the test-driven technique to manage the development. And documentation should also be fulfilled during the process.

4.Implementation

As the project description mentioned above, there are two approaches to avoid the name collision in distributed TensorFlow.

4.1 Distributed Key-Value Store

This approach is about setting up a distributed key-value store like etcd to manage the possible domain name. Once the nodes are up, finding a possible domain name concurrently from the K-V store. So they can avoid name collision. This way is straightforward but needs additional DevOps to setup and maintain K-V store.

And this approach is not difficult to implement. I think the main steps are as follows:

- **Set up an etcd for the distributed TensorFlow.** In this step, I would look into the documentation of etcd and set up a basic key/value store for possible domain name map for the cluster.
- **Integration with CoreDNS.** Then I would combine the K-V store with the CoreDNS, which means after the node self-identify itself, it should set its information to the K-V store so that other nodes can avoid domain name collision by getting information

from the etcd. So in this step, two main functionality should be implemented: the first one is a setter for nodes to put information to the K-V store. And another is a getter for nodes to check the information from the K-V store. So during the start time, the node should acquire a lock from the etcd, and then check the K-V store to see if the domain name it proposed is valid and then set its own domain name in the K-V store. After that, the lock should be released.
- **Handle failure nodes.** When the failure nodes are replaced by the new nodes, the new nodes should take over the previous domain name. So when the new node is up, it gets the information from the K-V store and then takes the domain name.

This approach is mainly about integrating an etcd with the system which certainly requires more resources (DevOp).

4.2 Self-identify based on timestamps

Another way is to establish a sequence of the nodes within the cluster. For example, every node can have a startup time before it is up, and in this period, we can assign the domain name based on the timestamp. In this way, it can save resources for K-V store. The implementation steps are as follows:

- **Literal research and thinking.** The startup time might be too long if there are thousands of nodes in the cluster. It is not smart if we assign the domain name for the node one by one based on the timestamp.Instead of doing this, I would find a proper way to assign nodes concurrently. For example, if the nodes have some common information, for example,if they all know the assigned order, then they can be assigned concurrently. But this still requires more attention and research.
- **Implement the identity protocol.** After I find a reliable information for every node to identify itself, then I would implement this protocol. I think the implementation is similar to the first approach. But now the nodes should get their assignment information from timestamp (or other information, like the id of the node in the cluster.). So the get method is needed. But we don't need the setter anymore because there is no place and no need to keep the whole identify information for the system.
- **Handle failure nodes.** I think this depends on which information the nodes self-identify based on. For example, if the identification is based on the timestamp, the new node can get the whole timestamp from the cluster and find the slot for itself.

Overall, since the second approach can save DevOps and it is more interesting, I would start with implementing the second approach, and it is also possible to fall back to the first one.

5. Timeline

Timespan	**Activity**
23 April - 14 May	*Community Bonding Period:*

	● Getting familiar with the structure and source code of CoreDNS. ● Doing research and communicating with mentor for the timestamp - based approach.
Actual Start of the Work Period	
14 May - 28 May	● Design and proposal a timestamp-based identify protocol .
28 May - 11 June	● Implement the protocol in Go ● Start writing unit tests. ● Start writing documentation.
11 June - 15 June	First evaluations: ● Getting timestamp or other information like id of the node in the cluster should be done.
15 June - 9 July	● Integrate with the CoreDNS. ● Keep writing tests and documentation.
9 July - 13 July	Second evaluations: ● Self-identify should work for the no failure distributed TensorFlow.
13 July - 6 August	● Handling the node failures case. ● Add more tests and documentation. ● Fall back to the first approach (depends on the result of the second evaluation).
6 August - 14 August	● Submit Code and Evaluations.

6. Availability

I will be based in Lausanne, Switzerland during the summer. Therefore, I will be working in GMT +2 time zone. And I will be available **35 - 40 hours per week**. Since our summer vacation is from June to September, so I believe that I have enough time to complete the project.

7. About Me

I am a second-year master student at Swiss Federal Institute of Technology in Lausanne (EPFL), majoring in Computational Science and Engineering. I am passionate about distributed system infrastructure because the distributed system is more similar to the storage system in the real world. And because the outburst of the data science, the reliability of the distributed system is crucial, so I desire to work in this field and try to improve the performance of the distributed system.

I am an Open source lover, and GSoC provides me a chance to make contributions to open source projects with mentorship from great developers all over the world. I believe it is amazing and I definitely can learn quite a few cutting-edge techniques from that.

Overall, I am looking forward to working with interesting people in the CoreDNS, and I will try my best to complete the project.