TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.4, April 2014, pp. 3247 ~ 3252
DOI: http://dx.doi.org/10.11591/telkomnika.v12i4.4887

A Multi-party Decision Hot-standby Model

Congdong Lv*, Wei Ma, Xiaoyong Li
School of Computer and Information Technology, Beijing Jiaotong University, China
*Corresponding author, e-mail: congdong.lu@gmail.com

Abstract

The dual server hot-standby mechanism is often used to improve the system availability. However, in traditional dual server hot-standby models, the states of servers are seldom determined from the client's observation, and it's easy for the master server and the slave server to make wrong decisions about the state of each other, which may cause split brain. This paper presents a multi-party decision hot-standby model. In this model, the master server and the slave server determine the state of each other not only from the observation of themselves, but also from the observation of the client, which helps them make correct decision to maintain or change the service platform, so as to ensure the continuity of application. Compared with traditional dual server hot-standby models, the model suggested in this paper is more reasonable because of the involvement of the client's observation.

Keywords: *hot-standby, split brain, availability, multi-party decision*

1. Introduction

The dual server hot-standby mechanism is a common method to improve service availability of the system. VRRP [1] and HSRP [2] are both classic hot-standby protocols. Many vendors and open source organizations also launched corresponding hot-standby solutions for their own platform and products, such as Linux-HA [3]. The general principle of hot-standby mechanism is that the master server/device and the slave server/device detect and judge the available state of the other through heart beat protocol. Once the slave server confirm that the master server is unavailable, it will take over its functions and provide services for clients. The research of dual server hot-standby mechanism focuses on fault detection, service recovery, and assurance between the master server and the slave server [3-5].

The traditional dual server hot-standby mechanism may cause the risk of Split Brain [6], [7]. The Split Brain is due to the failure of the communication mechanism (or heartbeat mechanism) between the master server and the slave server. It will cause the misjudgment between the servers. Then, both them cannot consistently provide services. There are many possible causes for heartbeat mechanism fails, for example, physical line problems or the failure of the heartbeat software. Fencing and Quorum mechanism is a common way to prevent split-brain [7, 8], such as the method of using SCSI reserve and Quorum Daemon, dual state consistency synchronization [9], etc. However, these methods have many limitations. It may cause a new single point of failure [10, 11]. In addition, these methods are not common. It is difficult to apply in the dual server hot-standby mechanism of the network equipment.

In order to overcome or mitigate the split-brain risk of the traditional dual server hot-standby mechanism, we present a multi-party decision hot-standby model. The multi-party decision mechanism refers that the master server and the slave server determine the true availability status of each other not only rely on their own observations, but also rely on the clients' observations. Because the services are for the client, it is reasonable to use the client's observations as a server status determination factor. It can effectively avoid the limitations and one-sidedness of the bilateral decision mechanism to combine the master server's observations and the slave server's observation together with the client observations in the multi-party decision mechanism. In addition, the dual server hot-standby model proposed in this paper not only can be used to the hot-standby mechanism of host and database services, but also can be used for the hot-standby mechanism of network devices.

Contributions of this paper include:

Received September 3, 2013; Revised November 6, 2013; Accepted December 11, 2013

1) Introduce the client's observations of the service status and use it as an important factor for the master server and the salve server to determine the availability of each other;
2) Comprehensively analyze multiple observations of the master server, the slave server and the client to provide the basis to judge the state of system services;
3) Present a multi-party decision hot-standby model and analyze its versatility and security.

2. A Multi-party Decision Hot-standby Model

The service provides for the client. However, in the traditional dual server hot-standby model, the client's observations weren't considered to determine the service's availability [12-14]. The multi-party decision hot-standby model consists of three important decisions entities: the master server, the salve server and the client, shown in Figure 1:

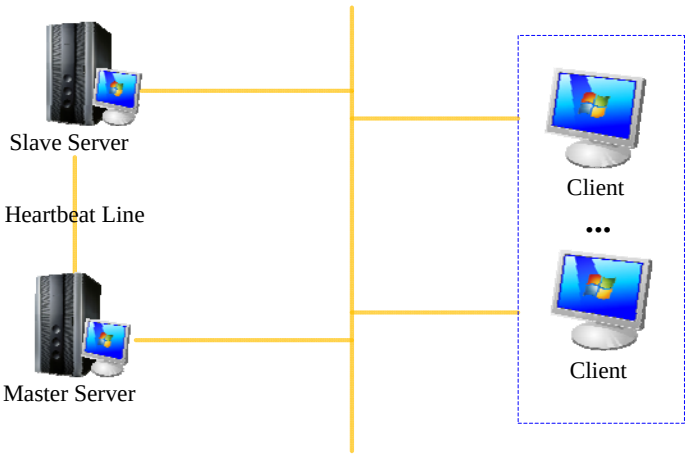

Figure 1. The Structure of the Multi-party Decision Hot-standby Model

For the participating entities of the structure in Figure 1, we use number N (=m*22+s*21+c*20) to represent one entity's communication state with the others, where m stands for the communication state with the master server, s stands for the communication state with the slave server, c stands for the communication state with the master server. The communication relationship between entities is bidirectional, that is, if the entity A can communicate with entity B, then entity B can communicate to entity A. Conversely, if the entity A cannot communicate with entity B, then the entity B cannot communicate with the entity A. The values of m, s, and c are Boolean. The value 0 indicates no communication between them. The value 1 indicates that they can communicate with each other. For each entity, assume that it always could communication with itself, and the value is 1.

The model in Figure 1 shows that all clients are a participating entity. If there is a client who can communicate with the master server or the slave server, then all clients can communicate with the master server entity or the slave server entity.

For convenience, we use M represents the master server communications form, S represents the slave server communication form and C represents the client communication form.

In the model of Figure 1, each entity may have four communication states with the other entities. For the master server, there are 7, 6, 5 and 4, where 7 means it can communicate with others; 6 indicates that it can only communicate with the slave server, but cannot communicate with the client; 5 indicates that it can only communicate with the client, but cannot communicate with the slave server; 4 means that it cannot communicate with others:

Table 1. Communication States between Entities

Entity	Communication State	Interpretation
Master Server	7	Can communicate with others.
	6	Only can communicate with the slave server, but cannot communicate with the client.
	5	Only can communicate with the client, but cannot communicate with the slave server.
	4	Cannot communicate with others.
Slave Server	7	Can communicate with others.
	6	Only can communicate with the master server, but cannot communicate with the client.
	3	Only can communicate with the client, but cannot communicate with the master server.
	2	Cannot communicate with others.
Client	7	Can communicate with others.
	5	Only can communicate with the master server, but cannot communicate with the slave server.
	3	Only can communicate with the slave master, but cannot communicate with the master server.
	1	Cannot communicate with others.

Because the communication is bidirectional, it isn't possible to randomly combine the communication state of the three participating entities in Table 1, for example, M=7 and C=3 cannot occur simultaneously because M=7 represents that the master server can communicate with others, but C=3 indicates that the client cannot communicate with the master server, that is the master server cannot communicate with the client. They are contradictory. Remove all conflicting combination, Table 2 lists all possible combinations of the communication state of the three participating entities. And it gives the interpretation.

Table 2. All Possible Combinations of the Communication State and the Interpretation

M	S	C	Interpretation
7	7	7	Normal
7	6	5	The master server can communicate with the slave server; The slave server cannot communicate with the client.
6	7	3	The master server can communicate with the slave server; The master server cannot communicate with the client.
6	6	1	The master server can communicate with the slave server; Neither the master server nor the slave server can communicate with the client.
5	3	7	The master server cannot communicate with the slave server; The slave server can communicate with the client.
5	2	5	The master server can communicate with the client; Neither the master server nor the client can communicate with the slave server.
4	3	3	The slave server can communicate with the client; Neither the slave server nor the client can communicate with the master server.
4	2	1	They cannot communicate with each other.

Here we give some definitions about the model in Figure 1.

Definition 1 (information exchange rules) entities in the dual server hot-standby model based on multi-party decision mechanism should follow the following rules:

1) Each entity save an internal data structure T. T = MSC, where M, S, C represent the communication state of the master server, the communication state of the slave server and the communication state of the client. After system initialization, T is set to 000.

2) Every time t, each entity generates M, S or C with communication state according to their network connection status and updates to T. Then he sends it to the other two entities. For instance, the master server generates the communication state M, then updates to M in T stored by his own, and finally, sends it to the slave server and the client. The slave server generates the communication state S, then updates to S in T stored by his own, and finally,

sends it to the master server and the client. The client generates the communication state C, then updates to C in T stored by his own, and finally, sends it to the master server and the slave server.

3) After each entity receives the communication state from other entities, he will update it to T. For example, the master server receives S from the slave server, it will directly update S in T stored by his own.

4) If an entity has not received the communication state of other entities within the time period e, it will set the corresponding item to 0. For example, if the master server received no communication state from the slave server within the time period e, it will set S in T stored by his own to 0.

Table 3. The Contents of each Entity's T Form under each Communication State

M	S	C	Master Server T	Slave Server T	Client T
7	7	7	777	777	777
7	6	5	765	760	706
6	7	3	670	673	073
6	6	1	660	660	001
5	3	7	507	037	537
5	2	5	505	020	505
4	3	3	400	033	033
4	2	1	400	020	001

As shown in Table 3, for each entity, in addition to no communication with other two entities (the shaded entries in Table 3), its value of T is consistent with the communication state MSC of the system, which means that even if the communication state known by each entity is not consistent with the real communication state of the system, it will not affect its properly understanding about the real communication state of the system. In fact, if an entity has no communication with other entities, then the other entities' real communication state has no meaning to it. For example, in the last two rows in Table 3, T of the master server represents that it has no communication with the slave server and the client. So regardless of existing communication between the slave server and the client (the last row but one in Table 3) or no communication between them (the last row in Table 3), the master server knows that it couldn't provide services any more.

Definition 2 (service switching rules) switching rules in the multi-party decision hot-standby model follow the following rules:

1) When T stored in the master server is 670 or 400, the master server provides no service, or continues to provide services.

2) When T stored in the slave server is 673 or 033, the slave server takes over services from the master server, or continues an alternate service.

As shown in Table 3, when T stored in the master server is 660, T stored in the slave server is also 660. In this case, the master server has no communication with the slave server. So they are all unavailable to the client, and requirement to switch the server does not exist. In a real system, in this case, the master server and slave server should send alarm information to alert managers that the network may malfunction.

3. Security Analysis

Analysis methods in other works [15] can be used for reference. To determine the dual server hot-standby model which based on multi-party decision mechanism is to work properly, we must verify whether there is situation that the master server and the slave server compete to provide services. In other words, when the slave server takes over the service, the master server is also providing services. If there is competition for providing services, it can be considered that the model exist "split-brain".

Based on the service switching rules defined in definition 2, only when T stored in the slave server is 673 or 033, the slave server is trying to take over providing services. At the same time, the T stored in the master server is 670 or 400. It is the condition for the master server out of providing services. The multi-party decision hot-standby model avoids "split-brain" between the master server and slave server.

From the one described can be seen, it is precisely because the client's participation, which makes primary and secondary servers determine the state of each other have more evidence, and thus the state of each other to make the right judgments. Client as a service object, which the master and slave server state judge has two effects: First, it establishes the principle of majority judgment advantages, thus breaking the tradition of the master and slave servers as a result of an equal number of judgments in each state there is a conflict controversy. Secondly, the client is not mainly provided from the server based on the judgment, when the client triplet T is 004 when the primary server cannot provide services to clients, so even if the state judge appears contradictory to each other will not bring more service availability issues.

4. Implementation

The multi-party decision hot-standby model has no technical obstacles in the realization. This approach in this paper is both to run agent software in the master server and the slave servers. Then select a number of clients, and also install and run agent software on clients. This agent software supports the rules in definition 1 and definition 2:

1) Entities send hello packets to each other via IP multicast and report the communication status N;

2) Entities maintain T = MSC based on the communication status of themselves and the received hello packet.

In order to ensure switching's real-time ability and reliability of the communication status, all entities need to maintain several time variables, including:

1) The hello packet's interval time is hello-time, which corresponds to the time t in definition 1. Every time t, each entity sends a hello packet to report its communication status;

2) Elements' expiration time in T is expire-time, which corresponds to the time e in definition 1. If an entity is not received communication state from others within expire-time, then set the related element to 0 in T. For instance, within the expire-time, if the master server does not receive any hello packets from the client, then the C in T is set to 0. Taking into account the network may be delayed, the expire-time must be greater than the hello-time, generally set at least that the former is the latter triple.

Since all clients are a whole participating entity, so for any client, the manner which generates client communication state C is different from the other two entities. All clients set up the communicate state form of themselves according to their communication states and other clients' hello packets, for example if a client does not communicate with the slave server, but as long as it receives any one client hello packets It can communicate with the slave server, it will update the communication state N=N+1*21. Similarly, for the master server and the slave server, as long as one of them can communicate with the client, then update the communication state N=N+1*20.

Also, the model should try to avoid the problem when the client is off line at the same time. If all clients are shut down, then both the master server and the slave server will think that they cannot communicate with the clients. To avoid this problem, when select the client, we should pay attention to the following: 1) Select the appropriate number of clients; 2) Confirm the client does not simultaneously offline or shut down; 3) Try to ensure that these clients are not connected with the same network access device at the same time, to avoid network equipment failure causes all clients while offline.

5. Conclusion and Future work

The multi-party decision hot-standby model can overcome the possible split brain in the traditional hot-standby system, and its implementation is not complicated, and can be flexible configured. It doesn't require changing the system platform and applications.

Future works include when how many clients included in the mechanism, the model is best. Also, How to choose the clients may be a good work.

References

[1] Knight S, Weaver D, Whipple D, et al. Virtual router redundancy protocol. *RFC2338*. 1998.
[2] Li D, Morton P, Li T, et al. Cisco hot standby router protocol (HSRP). 1998.
[3] Zhong C, Zhang L, Li H, et al. *Research and Implementation of Dual-Server Hot-Standby of Configuration Software*. Intelligent Control and Automation. WCICA 2006. The Sixth World Congress on. 2006; 2: 6120-6123.
[4] Chen CH, Ting Y, Lu W B, et al. *Recovery mechanism design for hot standby computer system*. Systems, Man and Cybernetics, 2003. IEEE International Conference on. IEEE. 2003; 3: 3027-3031.
[5] Nakamura S, Nakayama K, Nakagawa T. *Optimal backup interval of database by incremental backup method*. Industrial Engineering and Engineering Management. IEEM 2009. IEEE International Conference on. 2009; 218-222.
[6] Han Y. An integrated high availability computing platform. *Electronic Library*. 2005; 23(6): 632-640.
[7] Coulouris G, Dollimore J, Kindberg T. Distributed Systems: Concepts and Design Edition 3. *Paragraph*. 2001; 15: 04-04.
[8] Yang X, Wang Y, Liu Y. Design and implementation of dual-server hot-standby system for real-time database. *Jisuanji GongchengyuYingyong (Computer Engineering and Applications)*. 2012; 48(29).
[9] Kamyod C, Nielsen RH, Prasad NR, et al. *Resilience in IMS: End-to-end reliability analysis via Markov Reward Models*. Wireless Personal Multimedia Communications (WPMC). 15th International Symposium on. 2012; 564-568.
[10] Zhou G, Zhao H, Guo W. *Safety requirements analysis and performance verification of hot standby system using colored Petri-net*. Industrial Electronics and Applications (ICIEA), 8th IEEE Conference on. 2013; 656-661.
[11] Li MJ, Diao Y, Zhang Y. Design and implementation of gas disaster early-warning system based on hot-standby. *Gongkuang Zidonghua-Industry and Mine Automation*. 2013; 39(5): 19-23.
[12] Mizutani K, Takeshita H, Ishii K, et al. Experimental validation of effect of the power-saving standby mode on the backup path. *OptoElectronics and Communications Conference held jointly with International Conference on Photonics in Switching (OECC/PS)*. 2013; 1-2.
[13] Su H, Zhang Y. Distribution grid fault location applying transient zero-mode current. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(5): 883-890.
[14] Li H, Lei Z. Research on Infrared Special Facula View Measurement Method Based on Image Processing Technology. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(6): 1422-1429.
[15] Samosir AS, Sutikno T, Yatim AHM. Dynamic Evolution Control for Fuel Cell DC-DC Converter. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2011; 9: 183-190.