Research on High Availability Architecture of Cloud Platform

Lai Xinming¹, Wang Haitao², Zhao Jing², Zhang Fan², Zhao Chao², Wu Gang²,*

¹Aisino Corporation, Beijing, China
²China National Institute of Standardization, Beijing, China
*Corresponding author
Email: wugang@cnis.gov.cn

Abstract. Cloud computing platforms have extremely requirements for high availability, and many solutions have proposed to ensure high availability of system. In view of the shortcomings of the existing solutions, we improve the high availability of the cloud computing platform based on the 3 tiers cluster environment, including the business layer cluster, the management cluster, and the main control layer. The management cluster ensures high availability of the service layer cluster. The main control layer ensures high availability of the management cluster. The main control layer uses the dock mechanism to ensure certain high availability. Through practical application, the solution achieves high availability of the cloud computing platform, achieving availability 99.99%.

1. Introduction
High availability [¹] is a critical characteristic of a system, which ensure the system run so long time relative to “100% operational” or “never failing”. High availability decides the economic effectiveness of cloud computer platform, which thousands organizations deployed their business system on it. Many researchers focus on improve the availability of system. The earliest solution deploys the business system depended on single cluster like K8S, but the most critical problem which lies in business system crashed down when any single node breakdown. To solve this drawback, researchers design and implement Two-level business cluster environment [²], the top tier is a lightweight centre scheduler, and the other tier is special application scheduler, like Hadoop, Spark, Storm. In this scenario, the whole cluster will unavailable while the top tier centre scheduler collapsed. In this paper, we put forward 3 tiers architecture to ensure the high availability of system, includes substantial control layer, management layer and business layer. The business layer cluster environment ensured by management layer, and the high availability of management layer ensured by the substantial control layer, and high availability of substantial control layer ensured by Docker mechanism. This architecture applied in system Harchet, and the testing result shows the availability time of this system reach to 99.99%.

The section of this paper as following: Section 2 represent the key technologies used in this scenario. Section 3 introduces the architecture logic and implement methods. The conclusion is given in section 4.
2. Key Technologies of the architecture implementation

2.1. Kubernetes
Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. In this paper, Kubernetes responses to deploy thousands business applications and management systems, Kubernetes make sure the high availability of these applications.

2.2. Heartbeat Mechanism
The heartbeat mechanism \(^{(3)}\) is often used in network programs. When there is no data interaction between the client and the server, the heartbeat is required to detect whether the other party is alive. Heartbeat detection can be initiated by the client or server.

The mechanism is entirely contained in the connector client and connector server components, no additional objects are involved. In addition, connector clients send notifications that the manager application can receive to be aware of changes in the status of a connection

3. Design and Implementation of Harchet Architecture
As the mentioned above, Kubernetes responses to make sure the high availability of deployed applications. This paper focus to ensure high availability of Kubernetes.

3.1. The logic architecture of Harchet
The logical architecture of Harchet like a two-tier client/server architecture. Harchet server is the controller deployed on the nodes outside business clusters. Harchet client deployed on the nodes of business cluster, make a master-slave structure. In the single business node, Prometheus collect monitoring information of business nodes, and Harchet client response to send the information and server status to the Harchet server by heartbeats. As the fig. 1 shows.

![Fig. 1 the Diagram of Logical Architecture](image)

The server in controller responses to receive data, data analysis and assign operating maintenance tasks, the operating maintenance task assigned by data analysis results. In this logical architecture, server is deployed as single node, which be formed as containerization deployment like Docker or
single binary deployment. As the single node deployment, exists single point of failure. Through the testing, the high availability reached 99.5%.

3.2. The Embedded Architecture of Harchet
Generally, 3 tier architecture used to solve the problem of single point of failure, deploy multiple instances as controller in the cluster, and run zookeeper choose master problem. Through load balance components to share the performance stress. But this approach has a drawback, monitor message feedback delay will lead to client loss be monitored if one instance breakdown. In this paper, use scheduling framework kubernetes as business cluster to solve load scheduling. The architecture focus to make sure the high availability of kubernetes, then the high availability reached 99.99%. The diagram of embedded architecture as the fig. 2.

![Fig.2 the Diagram of Embedded Architecture](image)

In this architecture, business cluster ensures high availability of applications, Harchet client run business applications and collect client status as monitors send to controller cluster by heart beats mechanism. In controller cluster, as kubernetes cluster to assure high availability of kubernetes. kubernetes cluster include scheduler, controller-manager and API server services, these three services communicate with kubelet and kube-proxy on nodes continually to maintain the working status of the whole kubernetes cluster, if the master services access a certain node failed, the node will be marked unavailable and schedule pod on longer for the node.

These components started as container by kubelet, kubelet monitoring their status. API server is the core of the cluster, response the communication of function modules in cluster. The cluster modules storage their information into etcd via API server, API server provide information interaction when acquire and operate these information.

Controller management is the management control centre of cluster, response to manage the node, pod copy, endpoint, service account and so on. If one node failed, Controller management will find the diagnosis in time and fix it automatically, to ensure the cluster working all time.

4. Conclusion
This paper introduced a method of architecture to ensure high availability of cloud computer platform, use kubernetes ensure the high availability of applications deployed on the business clusters, build controller cluster to make sure the high availability of kubernetes, through kubelet monitoring the status of each node and use heartbeats mechanism implement delivery the monitoring messages. Based on this architecture, the system high availability time reach to 99.99%.
Acknowledgments
This paper is supported by grants from National Key R&D Program of China (522016Z-4973) and China National Institute of Standardization (522018Y-5941, 522018Y-5948).

References
[1] Floyd Piedad, Michael Hawkins (2001). High Availability: Design, Techniques, and Processes. Prentice Hall. ISBN 9780130962881.
[2] http://kubernetes.io
[3] http://www.oracle.com
[4] Bell C. (2018) Introduction to High Availability. In: Introducing InnoDB Cluster. Apress, Berkeley, CA
[5] Zitan Chen, Alexander Barg, "The repair problem under connectivity constraints: Explicit MSR codes for the rack-aware model of distributed storage", Communication Control and Computing (Allerton) 2018 56th Annual Allerton Conference on, pp. 710-714, 2018.
[6] S. Sahraei, M. Gastpar, Increasing availability in distributed storage systems via clustering, 2017.
[7] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, S. Vishwanath, "Locality and availability in distributed storage", IEEE Transactions on Information Theory, vol. 62, no. 8, pp. 4481-4493, 2016.