TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 10, October 2014, pp. 7430 ~ 7437
DOI: 10.11591/telkomnika.v12i8.5433

An Elastic Data Intensive Computing Cluster on Cloud

Zhaolei Duan*, Xueli Wu
College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Dongfeng Road, Zhengzhou, Henan Province, China
*Corresponding author, e-mail:duanzl@zzuli.edu.cn

Abstract

In order to meet the increasing processing demand of data intensive computing, an elasticdata intensive computing cluster EDICC is presented. Using local resource and cloud resource at the same time, EDICC has high availability and reliability, especially when flash crowd happens. EDICC could acquire resource from cloud when system is overloaded and release unnecessary cloud resource after load down. Experimental results show, comparing withtraditional data intensive computing system, EDICC could achieve outstanding performance and higher resource efficiency.

Keywords*: data intensive computing, load balancing, cloud computing*

1. Introduction

Data intensive computing is the new direction and hot spot of high performance computing research in recent years. Data intensive computing could achieve cross regional and cross platform data processing, massive data processing and analyzing, design and decision-making assisting. Research results show, in big business and research institute, the increase speed of data need to be processed is 4 times the increase speed of hardware processing capability. Facing the explosive increase of data amount, we must find an effective solution to meet the demand of data processing and acquire valid information from massive data. Currently, the main method to accomplish data intensive computing and meet growing processing demand is using computing system which has high enough computing capability, such as mainframe, super computer and computer cluster.

How to build computing system for data intensive computing, how to make data intensive computing system's software and hardware be highly scalable and reliable, is the focus of current data intensive computing research. In traditional data intensive computing system, there always is the situation that system load exceed system processing capability. In this situation, the system performance will decrease significantly, even worse, system will enter suspended animation, cause service unavailable. A typical example is the train ticket booking system in spring festival. Statistics shows, during spring festival of 2012, daily network hits of China train ticket website "12306" exceeded 1.0 billion. There was network congestion in "12306" website from 6 o'clock in the morning. During spring festival of 2013, the congestion situation of "12306" website became worse, daily network hits exceeded 10.0 billion. This situation, of course, violated the principle of high availability and high reliability of data intensive computing system, must be avoided. Unfortunately, existing data intensive system has no enough flexibility of resource supply. When there is a burst in workload growth, such as flash crowd in website, traditional data intensive system could not handle it effectively. Even if use processor and other hardware resource as more as possible when build data intensive system, along with rapid increase of data amount, available resource in data intensive system will be exhausted eventually. Therefore, in order to handle continued and rapid growth of resource demand in data intensive system, we must think of system scalability from the very beginning of system design.

As a high scalabilitysolution, cluster is a common scheme for data intensive computing system. Traditional computer cluster design, achieve its scalability through adding new node into system, which requires more system cost. Using traditional cluster to build universal data intensive computing system has obvious shortcoming. For example, in order to handle some application, we increase system investment, enhance system computing power, after this

Received December 23, 2013; Revised July 29, 2014; Accepted August 17, 2014

application is done, system no longer need all processing capability, part of system resource will be idle and part of system cost is wasted. Ideally, we need a data intensive computing system, in which there are enough processing capability and other relative resource for all customers and applications, on the other hand, there is no too much idle and wasted resource at the same time. In any moment, resource in system could be utilized effectively, and all customers' service quality will be guaranteed. In other word, when design universal data intensive computing system, we need a scalable and flexible system architecture, make system have enough resource to meet the dynamic processing demand, and improve system resource utilization efficiency. Apparently, there is a contradiction between improving resourceutilization and ensuring customers service quality. In traditional data intensive computing system, usually use load balancing strategy to relieve and avoid this contradiction. But because data processing demand is growing endlessly, eventually this contradiction is unavoidable. Recently , along with the development of cloud computing, the concept of "acquiring resource on demand" is presented, point out a new direction to solve above contradiction.

In this paper, we present a new elastic data intensive computing cluster—EDICC. In EDICC, we use hybrid resource supply mode: EDICC could use local resource and cloud resource at the same time. EDICC is part in cloud: when system is in heavy load, it uses cloud resource to increase available resource amount and guarantee service quality. EDICC is part out of cloud: it uses local resource to enhance system reliability and security, keep service available even if cloud resource is unavailable. In this way, EDICC could achieve outstanding performance and resource efficiency comparing with traditional data intensive computing clusters.

2. Current Research and Relate Work

Data intensive computing system which has fixed number of resource, usually Current research and Relate Work use load balancing strategy and resource dispatching strategy to guarantee system service quality. Along with increase of the task number in system, available resource will be in short supply, the effect of load balancing strategy and resource dispatching strategy become more and more weak, system service quality will decline inevitably.

In traditional data intensive computing system, to handle the situation that resource demand exceeds supply, there are several ways: admission control, service downgrade, dynamic increasing server resource. These methods could be divided into two kinds: first kind method, solve problem of lacking resource supply through reducing some users' or all users' service quality. Second kind method, solve the problem through increasing the amount of available resource in system. Service downgrade method only avoids the problem of lacking resource supply in some degree. Increasing server resource method is limited by the total amount of resource in server farm, we could move resource from one application to another, but there still will be problem of lacking resource supply when the total resource demand exceeds total amount of available resource in server farm. Therefore, traditional data intensive computing system could not eliminate the possibility of lacking resource supply. When the processing demand bursts, such as flash crowd happens, this possibility willraise significantly, eventually cause sharp decline of system availability and reliability. Some data intensive computing system only facing the problem of lacking resource supply in some special period of time, for example, China train ticket booking system ("12306" website) in spring festival. In this situation, although we could purchase enough software and hardware resource in advance, but this solution is very uneconomic. On the other hand, along with the continuing growth amount of data and computing, even if there are enough resource in system, more resource always will be needed eventually. So, how to make data intensive computing system has enough resource supply elasticity, and control system cost at the same time, is the core problem when design data intensive computing system.

In the 90's of the last century, the concept of grid computing is presented, describing such a technology: allow customer acquire computing resource on demand, just like using electricity and water resource in daily life, pay the cost according to the amount of resource they used. Although grid computing research make considerable progress in more than a decade, but the original goal of grid computing is not fully realized. Current grid computing is still limited in sharing resource among research institutes, users is resource provider and resource consumer at the same time.

Recently, cloud computing bring us new dawn. The article [1] and [2] introduce key technologies of cloud computing. In a word, cloud computing is “from cloud, to cloud”. Cloud is Internet, “from cloud” means customer could get required resource through Internet; “to cloud” means computing result will be sent back to customer through Internet. Any customer could get required resource through cloud computing and achieve application on cloud. In cloud computing, data is stored in cloud, applications and services are deployed on cloud, fully utilize computing power of data center to serve customer. Comparing with 10 years ago, the need for massive data processing is more and more urgent, thus need more and more computing power. Government facility, research institute, and Industry entertainment, all begin to use cloud computing power to meet their continuing increasing computing need.

Along with the development and popularization of cloud computing, more and more researchers begin to do research using cloud resource. NIR [3] presented by Zhou Yang is an open source cloud enabled content based image retrieval system. Content based image retrieval is high computation task because of the algorithm computation complexity and big amount of data. NIR using cloud resource to build image retrieval system, is easy to extent and flexible to deploy.

Tim Dornemann [4] presented a scheme of BPEL workflow engine basing on cloud computing. The basic idea of this scheme is using virtual machines in Amazon’s EC2 (Elastic Compute Cloud) to provide new hosts and handle peak load situations in BPEL workflow system. The presented system supports on demand resource provisioning. Experimental results for a computationally intensive video analysis application show this solution is feasible and efficient.

Existing data intensive computing clusters lack of flexibility on resource provision. When the system is overloaded, the performance become poor, or even worse, the service will be unavailable. Unfortunately, only using load balancing strategy could not solve this problem. So, basing on cloud computing, we propose an elastic data intensive computing cluster EDICC. In EDICC, we could solve this problem fundamentally, and guarantee the high availability and reliability of our data intensive computing cluster.

3. System Architecture

In our elastic data intensive computing cluster, back ends of cluster can be divided into two groups: local node and cloud node. Local nodes are constructed using local resource, they are essential component of EDICC. Local nodes are always part of EDICC unless it could not work normally. Cloud node is built using cloud resource, they are same with local node functionally. The difference between local node and cloud node is the number of cloud node is changing according to the system load status and available resource amount. Fig. 1 shows the structure of EDICC.

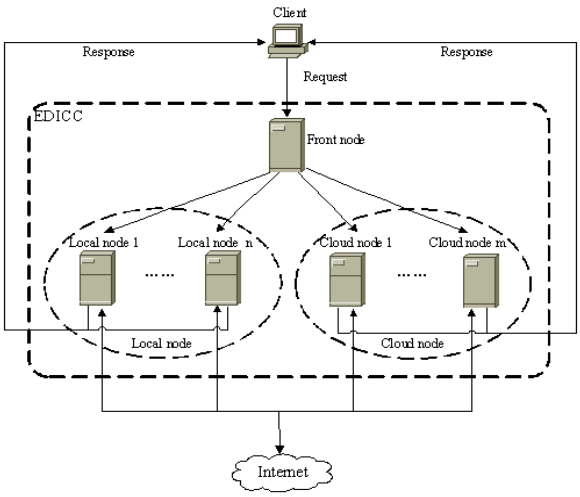

Figure 1. The Architecture of EDICC

In EDICC, cluster has n+m back ends, including n (n>0) local nodes and m (m>=0) cloud nodes. From the aspect of resource supply, our EDICC could use local resource and cloud resource at the same time, namely, EDICC has a hybrid resource supply mode. Our elastic data intensive computing cluster in cloud environment could combine local resource and cloud resource seamlessly. Use cloud resource to insure service quality and system performance when system is overloaded. Use local nodes to guarantee the reliability of EDICC.

4. Performance Index

In Figure 2, we simulate data intensive computing cluster, choose web cache service as the application, use NLANR's log to test performance of traditional data intensive computing clusters with different fixed node numbers. The result shows, along with the increase of node number, the average response time is reduced.

Obviously, more heavy the system load is, more improvement the increase of node number will bring. When the node number exceeds some degree, the improvement become unobvious, this mean the system has enough resource to handle current load. Basing on above analysis, we have an idea: We could adjust node number in Data intensive computing cluster. When system is in heavy load, we could increase available node. When system load down, we could decrease node number.

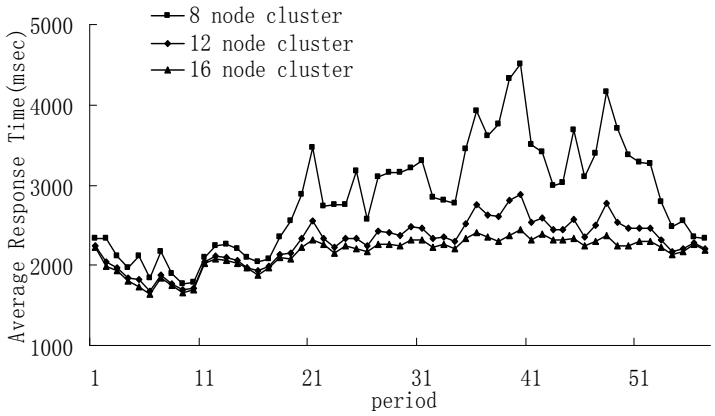

Figure 2. Respose Time of Traditional Data Intensive Computing Cluster

In order to realize our idea, we should choose a performance index firstly. Basing on this index, we could decide whether the EDICC needs more resource to handle current system load. When we evaluate the performance of web cache service, response time is usually taken into account in the first place. Because of the cache characteristic, there are always some requests which are not in cache should be sent to original server. These requests is so-called miss request, their response time is affected by network condition and original server process capacity. When there is network congestion or overloaded original server, miss requests' response time will increase noticeably even if the load in EDICC is low. In other words, we could not use response time as performance index directly. So, we proposed a relative performance index R to represent the performance of EDICC. This relative performance index R is described as follow: R is ratio of hit requests' average service time to miss request' average original server response time in given period.

$R = \frac{AvgHitTime}{AvgMissTime}$, AvgHitTime is the average service time of all hit requests in EDICC, AvgMissTime is the average original server response time of all miss requests in EDICC.

Before using performance index R, we need set threshold, when R exceed the threshold, add new node to EDICC, otherwise remove node from EDICC. How to decide threshold is an important problem. We must maintain the system performance in an acceptable level. At the same time, we should reduce the frequency of node number's alteration, to avoid

performance jitter. In order to achieve above goals, We set a threshold range [t1, t2]. When R<t1, release cloud resource if there are cloud nodes in EDICC. When R>t2, acquire cloud resource, build cloud node to increase capacity of EDICC.

5. Deployment of Cloud Node

Providers of resource in the cloud, such as Amazon EC2 or the Science Clouds, enable users to acquire resource on demand. Resource in cloud usually provided in the form of virtual machines. Virtualization is the basic technology in cloud and users could fully control virtual machines. Acquire resource from cloud, construct virtual machine, and then install user specified application on virtual machine, this procedure is the deployment of user application in cloud.

Vmplant [5] uses a typical deployment method: use a image configured with basic environment to create virtual machine, then install and configure application. This method support fine grain deployment, could install different application as user need, but affected by application install and configure time, the deployment time may be relative long.

Another deploy method is using fully configured image in which application is already installed to create virtual machines. The shortcoming of this method is lack of flexibility, only suitable for the situation that needn't to change application. Despite above limitation, this method is simple and deployment time is short.

After acquire resource from cloud, we need to build cloud node and add it into our elastic data intensive computing cluster. This means we should deploy application in cloud. From the aspect of elastic data intensive computing cluster, if we allow a cloud node take hours to deploy application, EDICC will not be able to handle system overload situation timely. So, we need fast deployment in seconds or faster. Obviously, using basic image to create virtual machine, then install application, this method could not meet our need. In our elastic data intensive computing cluster, software environment and configure on different node are almost identical, except IP address. Therefore, we use an image fully install cluster and application software to create virtual machine, construct cloud node for EDICC.

6. Experimental Verification

In order to verify performance of EDICC, we have implemented a trace-driven simulation of data intensive computing cluster. We select squid access log of NLANR to be the trace in simulation of web cache application, compare EDICC with traditional data intensive computing clusters which have fixed node number.

In experiment, threshold of R is set to [0.3, 0.7], the sample period of R is 60 seconds, and the initial node number (local node number) of our EDICC is 8.

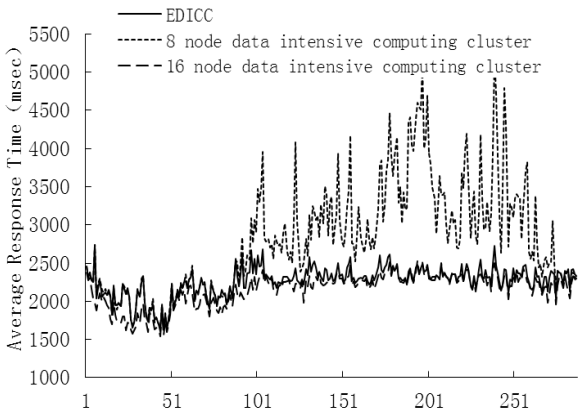

Figure 3. Performance of EDICC

Figure 3 shows the comparison result of performance among EDICC and traditional data intensive computing cluster with 8nodes and 16 nodes. From experiment beginning to 100th period, the response time curve of EDICC is almost the same with 8 nodes traditional data intensive computing cluster. When the system load begin to increase, 8 nodes traditional data intensive computing cluster could not provide enough resource, its response time has notable increase. EDICC could get more resource from cloud, and enhance system capacity dynamically, so response time in EDICC is kept in a relative low level. Comparing with 16 node traditional data intensive computing cluster, in the beginning 100 periods, EDICC's response time is little higher because EDICC has only 8 nodes. After 100th period, EDICC add cloud node into system dynamically, its response time is almost the same with 16 node traditional data intensive computing cluster.

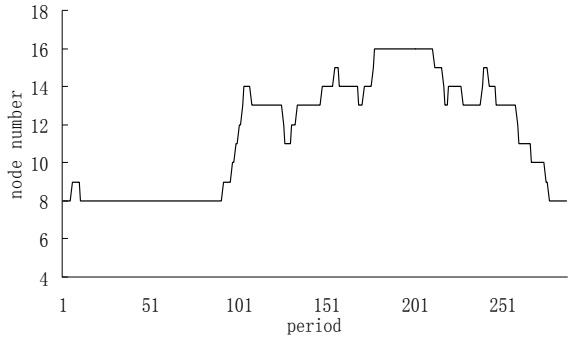

Figure 4. Node Number in EDICC

Figure 4 illuminates the node number change of EDICC in experiment. From the node number curve, we can tell since about 100th period, node number in EDICC begin to increase, this means EDICC start to acquire resource from cloud. The max node number in EDICC is 16 in our experiment. Analyzing Figure 3 and Figure 4, we can get a conclusion: EDICC has higher resource efficiency than 16 node traditional data intensive computing cluster, and could achieve almost same performance. In other words, comparing with traditional data intensive computing cluster, EDICC could improve service quality and enhance resource efficiency at the same time.

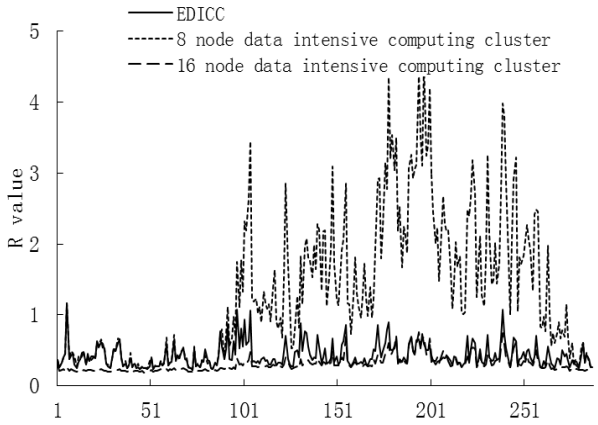

Figure 5. Comparision of R Value

Figure 5 shows change of performance index R's value in experiment. The change of R value is consistent with change of response time in data intensive computing cluster. When system lack of resource, R value will increase, when system has idle resource, R value will

decrease. The R value fluctuates according to the load in data intensive computing cluster. Range of R value's fluctuation affected by amount of available resource. If there is enough available resource, R value will fluctuateunobviously,otherwise R value will vary intensively. Among all three curves in Figure 5, curve of 16 nodes traditional data intensive computing cluster is the smoothest one, and curve of 8 nodes traditional data intensive computing cluster is the roughest one. In EDICC, because resource supply is dynamically adjust according system performance, R value's curve is very close to 16 nodes traditional data intensive computing cluster's curve. This proves R is an eligible performance index for data intensive computing cluster.

7. Security and Reliability

Because we use cloud resource in elastic data intensive computing cluster, it is inevitable that we will bring cloud security issues into our EDICC. Security issues is a hot research field in cloud computing.

In order to avoid the influence of security problem in cloud computing, we present a solution for EDICC. The main idea is: divide client or request into groups with different security priority. For example, we can set some IP address range as high security priority client, or set requests to special destination addresses or URLs as high security priority request. When EDICC receive requests with high security priority, send them to local nodes which have higher security configuration, using local node to guarantee the security in EDICC.

Similar to security in EDICC, for reliability issue in EDICC, we can give client or request different priority. When cloud resource is unavailable, use local node to serve high priority request firstly.

8. Conclusion

In this paper, we present an elastic data intensive computing cluster-EDICC. In EDICC, we use "on demand" cloud resource to construct new cluster nodes when load exceed system capacity, and release unnecessary cloud resource when load down. At the same time, use local resource to guarantee security and reliability in our data intensive computing system. In this way, we built a high available and high reliable data intensive computing system. Experiments' results show, comparing with traditional data intensive computing clusters which have fixed node number, our elastic data intensive computing cluster could accomplish outstanding performance, higher resource efficiency and lower system cost.

Acknowledgements

This work was supported by the Science and Technology Plan of Zhengzhou (No. 131PPTGG411-5), the Science and Technique Research Program of Henan Educational Committee (No. 14A520022), the Ph.D. Scientific Research fundation of Zhengzhou University of Light Industry.

References

[1] Chen Kang, Zheng Wei Min. CloudComputing: System Instances and Current Research. *Journal of Software*. 2009; 20(5): 1337-1348.

[2] Luo Jun-zhou, JinJia-hui, Song Ai-bo. Cloud Computing: Architechture and Key Technologies. *Journal on Communications*. 2011, 32(7): 3-21.

[3] Zhuo Yang, Sei-ichiro Kamata, AlirezaAhrary. *NIR: Content based image retrieval on cloud computing,* Proceedings of IEEE International Conference on Intelligent Computing and Intelligent Systems. IEEE Computer Society, Shanghai, China. 2009; 556-559.

[4] Tim Dornemann, Ernst Juhnke, Bernd Freisleben. *On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud.* Proceedings ofthe 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE, Shanghai, China. 2009: 140-147.

[5] Ivan Krsul, Arijit Ganguly, Jian Zhang. *VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing.* Proceedings ofthe ACM/IEEE SC2004 Conference on Supercomputing. IEEE Computer Society, Pittsburg, PA, USA. 2004: 1-12.

[6] Hideo Nishimura, Naoya Maruyama, Satoshi Matsuoka. *Virtual Clusters on the Fly - Fast, Scalable, and Flexible Installation*, Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Rio De Janeiro, Brazil, 2007: 549 – 556.
[7] Marios D Dikaiakos, George Pallis, Pankaj Mehra. Cloud Computing：Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing*. 2009; 13: 10-13.
[8] Wang Peng, Meng Dan1, Zhan Jianfeng. Review of Programming Models for Data-Intensive. *Journal of Computer Research and Development*. 2010; 47(11): 1993-2002.
[9] Zheng Pai, Cui Li-Zhen, Wang Hai-Yang. A Data Placement Strategy for Data-Intensive Applications in Cloud. *Chinese Journal of Computers*. 2010; 33(8): 1472-1480.
[10] Bicer Tekin, Chiu David, Agrawal Gagan. *A framework for data-intensive computing with cloud bursting*. Proceedings of the 2011 IEEE International Conference on Cluster Computing. 2011: 169-177.