A Virtual Network Performance Optimization Strategy for Cloud-based Big Data Processing

Pan Lulu

School of Software Engineering, Chongqing University, Chongqing 401331, China

Email:942230839@qq.com

Keywords: Cloud computing; OpenStack Neutron; virtual networks; MapReduce

Abstract. With the widely use of cloud-based MapReduce frameworks, an increasing number of solutions have been proposed to optimize the performance of the cluster. Majority of the existing studies concentrate on optimizing the task scheduling or resource provisioning mechanisms to improve the platform's data processing or communication performance separately, without an overall consideration. This paper designs and implements a novel virtual network performance optimization strategy for cloud-based big data processing, which changes the topology of virtual networks to optimize MapReduce performance. We implement our strategy (named Topology Optimized Mechanism on OpenStack Neutron) using python and integrate TOMON into Neutron as a separate plug-in to realize automatic deployment of virtual networks.

Introduction

With the popularity of cloud computing, more and more big-data applications are gradually migrating to the cloud platform. The flexible service framework and "on-demand access" business model of cloud platform making the cloud-based big data applications better scalability and higher value. In addition, this model provides intensive computing resources for the small and medium enterprises without the need to spend manpower and material resources to deploy and maintain their own cluster. There are typical enterprise applications that include Google BigQuery and Amazon EMR. How to improve the structure of the cloud platform to better support the deployment of MapReduce applications on platforms is a key research point. Specifically, two major factors affect the ability of MapReduce processing. One is the network delay of data transmission between virtual machines, and the other is the data processing rate of each specific virtual machine.The paper is to optimize virtual network topology by designing multi-host agent deployment strategies and improve the performance of the upper level of MapReduce by changing the virtual network topology.

Relevant technology

Modeling and solving systems. We conduct the modeling and performance evaluation of each phase of MapReduce workflow in cloud-based platforms. Map is divided into three stages: first, the big data cut into small data blocks, and then distribute to the communication agency; second, communication agency distribute the data from first stage to corresponding virtual machine; third, the virtual machine process the data forwarding from communication agency. The expression of total time-consuming of map task execution:

$$T_{total}^{map} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} t_{ij}^{map} = N \cdot \frac{S}{B} + \frac{\sum_{i}^{x_i=0}\sum_{j=1}^{n_i}\mu_{ij}}{\mu_{total}} \cdot \frac{S}{kB} + \frac{NS}{\mu_{total}} = N \cdot \frac{S}{B} + \frac{SN_a}{kBN\gamma} + \frac{NS}{\mu_{total}}$$

Reduce is divided into three stages: first, Mapper submits the results to the corresponding communication agency. Second, all communication agency sent the gathering results to the communication agency of Reducer. Third, Reducer communication agent forwards the results to the destination virtual machine. The expression of total time-consuming of Reduce task execution:

$$T_{total}^{reduce} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} t_{ij}^{reduce}$$

$$= \frac{(N-N_a)S_r n_r}{kB} + [\frac{NS_r n_r}{B} - \frac{(N-N_a)S_r}{kB}\cdot\frac{n_r k}{n_r+k-1} - \frac{N_a S_r n_r}{B(n_r+k-1)}] + \frac{NS_r}{B}\cdot(1-\frac{N_a}{N})n_r$$

$$= \frac{(\mu_{total}-\mu_{agent})NS_r n_r}{\mu_{total}kB} + [\frac{NS_r n_r}{B} - \frac{(1-\frac{\mu_{agent}}{\mu_{total}\gamma})NS_r}{kB}\cdot\frac{n_r k}{n_r+k-1} - \frac{\mu_{agent}NS_r n_r}{\mu_{total}\gamma B(n_r+k-1)}]$$

$$+ \frac{NS_r}{B}\cdot(1-\frac{\mu_{agent}}{\mu_{total}\gamma})n_r$$

Then, we can get the optimization model:

$$T_{total} = Min \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i}(t_{ij}^{map} + t_{ij}^{reduce}) \qquad \forall i \in [1,m] \quad \forall j \in [1,n_i]$$

$$T_{span} = Min \quad (Max \quad t_{ij}^{map} + Max \quad t_{ij}^{reduce}) \qquad \forall i \in [1,m] \quad \forall j \in [1,n_i]$$

The expression of total time-consuming of MapReduce execution in steady status:

$$T_{total} = T_{total}^{map} + T_{total}^{reduce}$$

$$= \frac{NS}{B} + \frac{S(N-\gamma N_a)(N-N_a)}{kBN\gamma} + \frac{N^2 S}{\mu_{total}N - (1-\gamma)\mu_{total}N_a}$$

$$+ \frac{[(2k+1)N\gamma - (k\gamma+1)N_a]S_r n_r}{kB\gamma} - \frac{(N-2N_a)S_r n_r}{B(n_r+k-1)}$$

Finally, we get the expression of the optimal number of communication agency through solving the multi-function:

$$\begin{cases} \dfrac{\partial T_{total}}{\partial k} = 0, \quad \dfrac{\partial T_{total}}{\partial N_a} = 0 \\ \dfrac{\partial^2 T_{total}}{\partial k^2}\cdot\dfrac{\partial^2 T_{total}}{\partial {N_a}^2} - (\dfrac{\partial^2 T_{total}}{\partial k \partial N_a})^2 > 0 \end{cases} \Rightarrow k_{optimal} = \frac{NSn_r\mu_0\gamma}{|BS(1-\gamma) - (n_r+1)NS_r\mu_0\gamma|}$$

$$N_a^{optimal} = N + \frac{(k+1)NS_r n_r}{2S} + \frac{kNS_r n_r}{2S(n_r+k-1)} - \frac{kB(1-\gamma)}{2\mu_0\gamma}$$

By solving an extended 0-1 knapsack problem we can get the optimal number of communication agency, the recursive expression of the algorithm:

$$S(i,k,N_a) = \begin{cases} S(i-1,k,N_a) & if \quad |S(i-1,k,N_a) - N_a| \leq |S(i-1,k-1,N_a-n_i) + n_i - N_a| \\ S(i-1,k-1,N_a-n_i) + n_i & else \end{cases}$$

The cloud platform mapping between the optimal communication agency and virtual machine can be abstracted as a load-balancing problem, it is NP-hard. We designed a greedy algorithm to obtain an approximate solution of the problem, which result is 1.5.

Cloud-based integrated monitoring module. In the cloud computing environment, with all the hardware resources are virtualized computing, it is not convenient to query for resources' characteristic, that is be hidden. When as a cloud service provider, they must grasp the real-time status of the user's current virtual machine, including processor's speed, storage conditions and the remaining space of disk. This requires cloud platform must have a quality of monitoring user usage of the host, and monitoring technology can solve this problem perfectly.

Nagios is an open source software used for system and network monitoring. Users can not only monitor the host and the service of host follow their own demand, but also sent warnings about the state of the host via e-mail and text message to the supervisor. Figure 1 shows the list of Nagios monitoring host and monitored hosts.

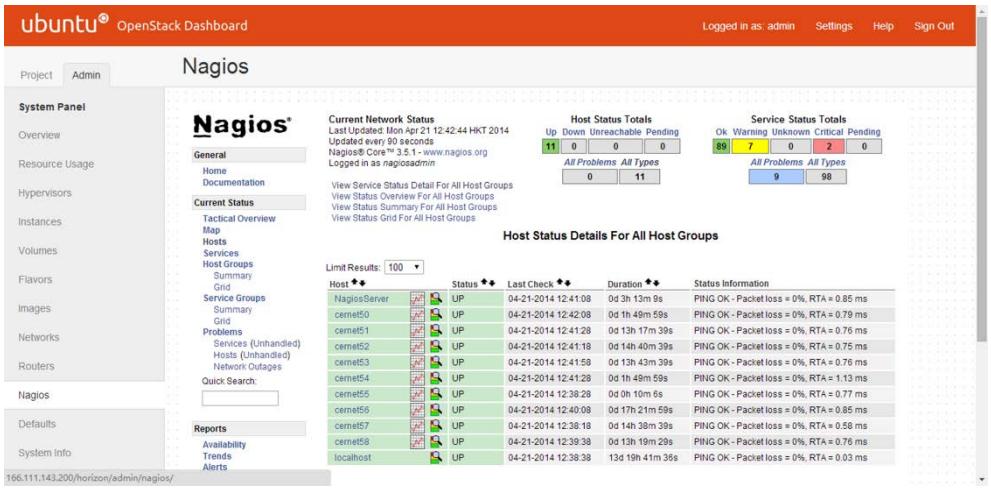

Figure1 the list of Nagios monitoring host and monitored hosts

Experiment

This experiment build a IaaS cloud computing platforms which contains 21 servers, including one control node, 20 compute nodes. We use a unified micro-host template, and then assign IaaS platform to 246 virtual machines to build a parallel computing cluster based on cloud computing. Where one virtual machine is configured as a Master node of Hadoop, other 245 units as Slave nodes of Hadoop.

Figure 2 shows the experiment result effected by virtual machine allocation communications agency strategies in Map stage. The red line use LoadBalancing algorithm to assign the virtual machine, while the blue line allocates virtual machine to the communication agency randomly. We can see that the experimental results using the algorithm LoadBalancing allocation strategy is better than the random distribution. Figure 3 shows the experiment result effected by deployment strategy of communication agency in Map stage. The red line use the Knapsack algorithm to deploy communication agency. The blue line select the position of the communication agency randomly. The green line and the black line use the greedy algorithm, which select maximum and minimum numbers of virtual machines to deploy server communication agency. The results shown in figure that the total time that using the Knapsack algorithm to deploy communication agency is optimal; while result using random deployment method is the worst in the average. Figure4 shows the experimental result effected by optimizing different conditions. The black line represents have not use any invention strategy, and the total time increases along with the amount of data. The blue line is optimizing the number of communication agency based on the black line. The green line is using the Knapsack algorithm to deploy communication agency based on the Blue Line. Last, the red line in the figure use LoadBalancing algorithm on the basis of the green line to further optimize the mapping relationship between communications agency and virtual machines. As can be seen obviously, the results get improved after using an optimization strategy and the span between the black line and the blue line is the largest, which indicates that the effect optimizing the number of communication agency is the best.

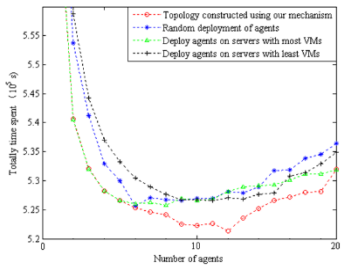

Figure2 Different allocation strategies

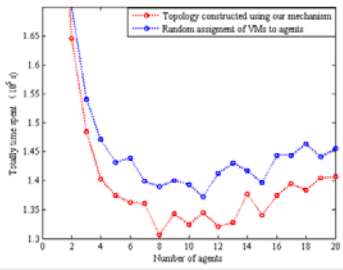

Figure 3 Different deployment strategies

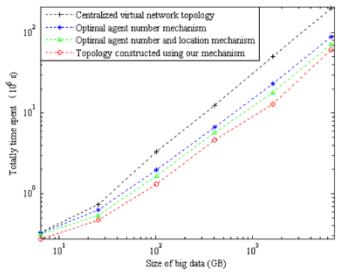

Figure 4 Optimization different conditions

Conclusion

This paper focuses on the existing research on performance optimization strategy for cloud-based big data processing and designing virtual network Multihost of Neutron, we propose a deployment strategy based on multi-agent communication of multi-objective optimization model. It achieves rapid deployment of virtual network performance optimization strategies through the Python language, and integrates monitoring module Naigos into the cloud platform to better observe and assess TOMON strategy platform brings performance improvements and to lay the foundation for further research. Meanwhile, this paper exists some problems and deficiencies. Performance optimization strategy which virtual network by changing the virtual network topology to achieve, and not go deep into the study of the distribution of the underlying virtual network bandwidth resources, and through our follow-up study found that virtual network performance bottlenecks additional sources bandwidth resources at the bottom cannot be fairly allocated.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to significantly improve the quality of this paper. The research reported in this paper has been partially supported by the Fundamental Research Funds for the Central Universities No. CDJZR1309551, the Fundamental Research Funds for the Central Universities No. CDJZR12090004.

References

[1]Yuan, Yi, et al. "On interference-aware provisioning for cloud-based big data processing." Quality of Service (IWQoS), 2013 IEEE/ACM 21st International Symposium on. IEEE, 2013.

[2]Schadt, Eric E., et al. "Computational solutions to large-scale data management and analysis." Nature Reviews Genetics 11.9 (2010): 647-657.

[3]Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[4]Zaharia, Matei, et al. "Improving MapReduce Performance in Heterogeneous Environments." *OSDI*. Vol. 8. No. 4. 2008.

[5]Chang, Hyunseok, et al. "Scheduling in mapreduce-like systems for fast completion time." INFOCOM, 2011 Proceedings IEEE. IEEE, 2011.

[6]Theja Maguluri, Siva, R. Srikant, and Lei Ying. "Heavy Traffic Optimal Resource Allocation Algorithms for Cloud Computing Clusters." arXiv preprint arXiv:1206.1264 (2012).