Analysis of Cloud Resource Optimal Allocation Strategy Based on Bayesian Algorithm

Xiaoxiao Fan1,*

1ShangHai Golden Education, 200080, ShangHai, People R China.

*Corresponding author e-mail: m1869976091@shge.org

Abstract. With the increasing popularity of big data technology, the scale of computer network system is larger and more complex, especially the rise of cloud technology, to a large extent, subverts many concepts of traditional computer network, which also means that the traditional modeling technology has been difficult to meet the accurate modeling of modern computer network. In order to manage and schedule cloud resources more efficiently, people have proposed various technical schemes, among which Bayesian optimization is one of the most concerned technologies. Bayesian optimization is a global optimization method, which can quickly find the optimal solution even in the calculation process without closed form and high complexity, and the difficulty of optimal allocation of cloud resources is just here. The experimental results show that the proposed Bayesian optimization method can find the optimal configuration in a finite number of times, which is obviously better than the traditional grid search and random search methods.

Keywords: Bayesian Optimization, Cloud Network, Resource Allocation, Optimal Solution

1. Introduction

The basic framework of Bayesian optimization theory is to set a broad model for unknown functions, constantly collect data, modify the prior setting, and update the posterior probability[1]. In the process of data acquisition, according to the collection function to ensure the correct direction, so as to optimize the calculation speed. It can be seen that the process does not need to model the data, and has good applicability. As long as the priori properties match the real scene, better results can be obtained[2]. There is no one method that can be applied to all problems. Therefore, what kind of prior knowledge is used needs to be analyzed in detail. Moreover, even if there is a big difference between the prior model selection and the actual situation, because the posterior probability is constantly updated, it can finally show some basic properties consistent with the actual situation.
In today's era, it has become a habit for organizations or researchers to run big data applications in the cloud [3]. Pay on demand is far better than purchasing equipment to deploy private servers in terms of convenience and cost saving. However, how to choose the appropriate cloud configuration for different big data applications is a problem. This problem is abstracted into an optimization problem, which can be improved by using Bayesian optimization algorithm OK. For example, in the adaptive problem of video stream, data-driven deep reinforcement learning method can be used to solve the problem. However, how to adjust the parameters of the neural network and get the best experimental results? The cost of using the typical grid search method is too high, and the random search method cannot guarantee the best parameters, then Bayesian is used. Both the advantages of the two methods can be combined to ensure the automatic parameter adjustment.

Bayesian optimization has a very broad prospect in the network resource allocation [4]. Of course, it is undeniable that in each specific problem, the Bayesian optimization algorithm needs to be adjusted accordingly, so as to be applicable to specific problems, such as the selection of a priori, the selection of collection functions, and so on. At the same time, how to model the problem is also a problem. The traditional precise modeling is how to model the relevant performance of interest, so as to solve the problem by using Bayesian optimization algorithm.

2. Overview of Optimal Cloud Configuration

2.1. The Necessity of Optimal Cloud Configuration

Running big data analysis tasks on cloud servers has become the key to almost all data analysis industries. In order to support various use cases, various technologies have been developed to facilitate data processing, such as map reduce, SQL language, deep learning, principal component analysis [5, 6, 7], etc. The execution environment structure of these big data analysis tasks is similar, namely virtual machine (cluster). However, different big data analysis tasks have different emphasis on resource requirements, such as CPU intensive tasks or IO intensive tasks, and virtual machine (cluster) resources (CPU) Speed, memory size and speed, hard disk size, network condition, etc.) requirements are different. It is not easy to implement unified cloud configuration (virtual machine type and number) for different big data analysis tasks. It is very important to choose appropriate cloud configuration for different analysis tasks for service quality and business cost control [8]. Bad cloud configuration selection will make the cost of analysis task increase significantly, even more than ten times of the optimal configuration cost.

2.2. Difficulties in Achieving Optimal Configuration

It is not easy to choose the appropriate cloud configuration for different big data analysis tasks. On the contrary, it is a relatively difficult work, which is embodied in the following aspects:

(1) Performance modeling. Since the time consumed by a task is non-linear with the resources on the cloud virtual machine [9], a fixed number of CPU cores are used to run spark regression tasks. When the memory reaches 256gb, the running time begins to decline. Therefore, allocating more than 256gb of memory to the task will improve the running time. Before that, increasing memory seems to have no effect. At the same time, the network situation in cloud configuration is dynamic, such as network bandwidth jitter, which makes it impossible to accurately model the problem.

(2) Cost model. Naturally, it can be thought that the cloud configuration with the highest performance for any task can certainly make the task run time the shortest and achieve the optimal performance [10]. However, in the actual production application, more resources will lead to a large amount of money. Therefore, in order to minimize the money cost, it is necessary to balance the contradiction between resource requirements and running time. When the regression task is run on spark, the money cost
decreases with the increase of the number of clusters used, which indicates that the decrease of the running time of the task is enough to offset the cost caused by the increase in the number of clusters.

(3) Scalability. There are many types of big data analysis tasks, such as CPU intensive tasks or I/O intensive tasks, and each task type requires different resource allocation ratios[11]. Terasort task uses low memory instance is the optimal configuration (minimum cost). Because terasort is a CPU intensive task, CPU speed is the key. As long as the memory is not particularly small, it has little impact on the task overhead. On the contrary, when running regression tasks on spark, as the memory available to each CPU core increases, the overhead begins to decrease rapidly at 4GB. Because the regression task is memory intensive, a large amount of intermediate data needs to be saved in the operation process. If the memory is insufficient, it can only be saved to the hard disk. The I/O speed of the hard disk is far less than that of the memory, resulting in the operation time The cost is very large. To establish a model for a specific task requires a long time of mathematical modeling and debugging parameters. Therefore, to accurately model each task, the consumption of human resources is huge. Therefore, the idea of performance modeling is not desirable[12].

(4) The problem of random search. Using random search is a good idea, but the random search algorithm has a high cost. In the worst case, this simple brute force search needs to traverse all the cloud configuration lists to get the optimal configuration, which consumes a lot of time and money[13]. In order to reduce the search time, we can improve the random search process, such as reducing the search dimension, searching only CPU / memory at a time, then searching the number of CPU cores of each machine, and then searching the cluster size. However, this kind of search algorithm may lead to the failure of searching the optimal configuration.

3. Optimal Allocation Method of Cloud Resources Based on Bayes

3.1. Overall Scheme Design
In the process of searching for the optimal cloud configuration, due to the lack of information available, it cannot be optimized through continuous trial and error[14]. Therefore, this feature is very suitable for using Bayesian optimization algorithm. The algorithm cost is very sensitive to the number of sampling points, so it is necessary to get the optimal or near optimal results within a few samples. One of the key problems is to stop the calculation in time. One of the methods is to set the number of iterations, but the optimal value may not be obtained. A better way is to set a threshold and stop the algorithm when the calculation result is less than a certain threshold.

Figure 1 shows the search process of performance model configuration, initialize less cloud configurations, run some big data analysis tasks on these cloud configuration instances, input specific configuration information and task running time into Bayesian model, and then dynamically select the cloud configuration of the next running task according to Bayesian algorithm, and then feedback the running time to Bayesian model. When the index is raised enough, stop the algorithm. It is an approximation of the performance model, which can get the optimal or near optimal cloud configuration. This method allows some uncertainties, so it has a good control of the cost, and has a certain degree of scalability.
Figure 1. Flow chart of optimal configuration search algorithm

3.2. Implementation Architecture of The Model

This paper implements the framework shown in Figure 2, which is divided into four parts

(1) Search controller. This component takes the user's specific application, objective function (cost or running time), constraints (longest running time, cost, maximum or minimum cluster size) as input to form a candidate cloud configuration list, and then pass it into Bayesian optimization component. At the same time, the search controller will first create a virtual machine through the cloud controller, and then install the specific application and its input into the virtual machine. Running in the simulation machine. The search controller also needs to monitor the state of Bayesian Optimization components to decide whether to stop the optimization.

(2) Cloud monitoring component. This component is mainly responsible for collecting task feedback and passing task feedback (mainly runtime) to Bayesian optimization component for decision-making. This component is lightweight and usually only runs when the task is completed.

(3) Bayesian Optimization driven. This component is based on the third-party Bayesian optimization system and only needs to be modified. At the same time, Bayesian Optimization driver uploads a series of candidate cloud configurations in the optimization process to the cloud service provider and runs through the cloud controller.

(4) Cloud controller. Cloud controller is a layer of encapsulation that unifies various cloud service providers. Each cloud service provider provides a customized API for virtual machine creation and destruction, virtual network creation and destruction, mirror virtual machine, and list instances. In order to facilitate management, the API is unified and easy to use through one layer of encapsulation. At the same time, the cloud controller also includes the function of uploading commands to a virtual machine through SSH.

After the implementation of the whole framework, the big data tasks can be stored in a specific folder, and then the purchased cloud service provider's secret key is added to the configuration file. The main program can automatically create a virtual machine on the cloud server according to the cloud configuration, upload the big data task, get feedback, destroy the virtual machine, update the model, and select the next cloud configuration to repeat the process. It is an automatic process, and with the increase of the execution times of the same task, it tends to choose the best cloud configuration for the task.
3. Application of Bayesian Optimization Algorithm

In order to implement Bayesian optimization algorithm, we first need to introduce the Markov chain Monte Carlo method (MCMC), and then describe the implementation of Bayesian optimization algorithm based on Gaussian process. MCMC is a method to estimate the posterior distribution of interest parameters by random sampling in probability space[15]. In the field of Bayesian inference, MCMC is a commonly used method to find the posterior probability of some unusual distributions.

4.1. Monte Carlo Method

At first, Monte Carlo method was used to solve some integral equations which are not easy to solve. For example, \( \int_a^b f(x)dx \) is difficult to solve. Monte Carlo method can be used to simulate the approximate value. A simple simulation method is to randomly sample \( n \) points \( x_1, x_2, \ldots, x_{n-1} \) between \( [a, b] \), respectively calculate \( f(x_i) \), and then use their mean value to represent the value \( f(x) \) in the interval \( [a, b] \), so the approximate solution of the integral equation is

\[
\frac{b-a}{n} \sum_{i=0}^{n-1} f(x_i)
\]

However, this method implies an assumption that \( x \) is uniformly distributed between intervals \( [a, b] \), and in most cases, it is not so simple. Assuming that the distribution is \( p(x) \), it can be calculated as follows:

\[
\theta = \int_a^b f(x)dx = \int_a^b \frac{f(x)}{p(x)}p(x)dx \approx \frac{1}{n} \sum_{i=0}^{n-1} f(x_i)
\]

The far right of the formula is the general formula of Monte Carlo method.

Now the problem is how to get the \( x \) sample points with the distribution of \( p(x) \). For common distribution, we can get it directly by formula. But for the uncommon distribution, a feasible way is to get the distribution sampling point of the sample through accept reject sampling. The accept reject sampling method assumes a common distribution \( q(x) \) (such as Gaussian distribution), and then reject some samples according to a certain method to achieve the purpose of approaching distribution \( p(x) \). But for high-dimensional distribution, it is difficult to find such a \( q(x) \)

4.2. Markov Chain

Figure 2. Implementation Diagram
Markov chain assumes that the state transition probability at a certain time only depends on the previous state. To use precise mathematical language to describe it is to assume that there exists a state sequence

\[ \cdots, X_{t-2}, X_{t-1}, X_t, X_{t+1}, \cdots \]

And the state at time \( t + 1 \) only depends on the state of time \( t \), that is

\[ P(X_{t+1} | \cdots, X_{t-2}, X_{t-1}, X_t) = P(X_{t+1} | X_t) \]

Therefore, the Markov chain can be determined only by knowing the transition probability between any two states.

The state transition matrix of Markov chain model is its core. If Markov chain can converge, no matter what distribution it starts from, it tends to the same distribution after finite state transition. This distribution is called the stationary distribution of Markov chain.

If the Markov chain state transition matrix of a certain stationary distribution can be obtained, the sample set of the stationary distribution can be easily sampled. Assuming that \( P \) is the state transition matrix, any initial probability distribution is \( \pi_0(x) \), the probability distribution after the first round of Markov chain state transition matrix is \( \pi_1(x) \), after the I round is \( \pi_I(x) \), after the nth round, the stable distribution \( \pi(x) \) can be obtained. For each distribution \( \pi_i(x) \), we can obtain

\[ \pi_i(x) = \pi_{i-1}(x)P = \cdots = \pi_0(x)P^i \]

The sampling process can be summarized as follows:

1. Input the state transition matrix \( P \) of Markov chain, set the number of state transition \( T \) and the number of sample points required \( n \).
2. The initial value \( a \) is obtained by sampling from any simple probability distribution \( x_0 \).
3. From \( t = 0 \) to \( t + n - 1 \): \( x_{t+1} \) is obtained by sampling from the conditional distribution \( P(x_{t+1} | x_t) \).
4. The final sample set \( \{x_T, x_{T+1}, \cdots, x_{T+n-1}\} \) is the sample set corresponding to the required stationary distribution.

### 4.3. Simulation Analysis

In order to verify the algorithm in this paper, we introduce a common test function, Branin function

\[ f(x) = \left( x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10(1 - \frac{1}{8\pi}) \cos(x_1) + 10 \]

The range of arguments to this function is \( x_1 \in [-5, 10], x_2 \in [0, 15] \), optimal value \( f(x^*) = 0.397887 \), where \( x^* = (-\pi, 12.275) \) or \((\pi, 2.275)\) or \((9.4278, 2.475)\). Then, the actual available Bayesian optimization algorithm is used to optimize the Branin function, and the optimization path is shown in Table 1. It can be clearly seen that Bayesian optimization can quickly get the optimal solution after a few iterations.

**Table 1.** Bayesian Optimization of Branin function path

| Branin Value   | x1     | x2     |
|---------------|--------|--------|
| 505.000965   | -5     | -5     |
| 26.627243    | 5      | 5      |
| 19.779172    | 4.223383 | 5.362742 |
| 9.847092     | 4.213910 | -0.511606 |
| 2.548846     | 3.813333 | 1.554687  |
| 0.759394     | 3.281030 | 2.686759  |
| 0.584543     | 9.390747 | 2.970106   |
| 0.465718     | 3.179018 | 1.998851  |
| 0.424777     | 9.382747 | 2.304093  |
| 0.399038     | 3.136473 | 2.111031  |
| 0.398018     | 9.424416 | 2.486095  |
| 0.424777     | 9.382747 | 2.304093  |
| 0.399038     | 3.136473 | 2.111031  |
| 0.398018     | 9.424416 | 2.486095  |
5. Conclusions
Bayesian optimization technology has been proved to have good performance for unknown and expensive function global optimization. The performance evaluation is generally divided into optimization speed and global optimality. In the scenario of searching for optimal cloud configuration, especially online, enough observation points are the most important decision factors. Global optimization can be appropriately abandoned, but however, when looking for a set of optimal hyper parametric scenarios for machine learning, it is more necessary to consider whether it is globally optimal due to the background of offline scenarios. Of course, because of the definition of Bayesian optimization, we first establish a sufficiently large prior assumption in the unknown space. By constantly sampling and updating the posterior probability, each iteration can compress the prior hypothesis and make it more and more fit the real function. At the same time, using the collection function to guide each sampling has its own property of optimization time.

After that, we use the Bayesian network optimization method to solve the problem of cloud optimization, and then we use the Bayesian network optimization method to solve the problem. The application of resource allocation shows good performance. In fact, due to the existence of Gaussian process (GP), the time complexity of Bayesian optimization algorithm is closely related to the number of observation points. Specifically, there is a cubic growth relationship between them. This means that Bayesian optimization algorithm still has great challenges in the application of large-scale automatic parameter adjustment problems. In order to realize the off-line calculation to online calculation, it is necessary to convert the cubic growth relationship to the linear growth relationship. Of course, to achieve this goal, there are still many technical difficulties that need to be further studied. It is hoped that the results of this paper can provide a useful reference for the research in related fields.

References
[1] Honglin Li, Payam Barnaghi, Shirin Enshaeifar, Frieder Ganz. Continual Learning Using Bayesian Neural Networks[J]. IEEE transactions on neural networks and learning systems. 2020
[2] Hao J, Zhang BB, Yue K, Wang J, Wu H. Performance Measurement and Configuration Optimization of Virtual Machines Based on the Bayesian Network[J]. CLOUD COMPUTING AND SECURITY, PT II. 2017: 641-652.
[3] Singh, R., Manitsas, E., Pal, B. C., Strbac, G. A Recursive Bayesian Approach for Identification of Network Configuration Changes in Distribution System State Estimation[J]. Power Systems, IEEE Transactions on. 2010, Vol. 25(No. 3) 1329-1336.
[4] Chen Lijun, Wang Chang. Research and Simulation of Failure-Aware Algorithms for Trusted Cloud Resource Scheduling[J]. Computer Simulation. 2020, (10): 334-337, 356.
[5] Wu Yuewen, Wu Heng, Ren Jie, et al. Heuristic Based Resource Provisioning Approach for Big Data Analytics in Cloud Environment[J]. Journal of Software. 2020, 31(6): 1860-1874.
[6] Geng Zhenwei, Chen Xueqin, Wang Xinyun. A Resource pool fault Diagnosis Method based on Bayesian Networks[J]. Yunnan Electric Power. 2017, 45(3): 92-94, 99.
[7] Jia Meng, song Chengxiang, Li Yang. RESOURCE SCHEDULING STRATEGY BASED ON BAYESIAN MODEL[J]. Journal of Shandong Normal University(Natural Science). 2013, (4): 20-23, 27.
[8] Hsu, C.-J, Nair, V., Freeh, V.W., Menzies, T.. Arrow: Low-level augmented Bayesian optimization for finding the best cloud VM(Conference Paper)[J]. Proceedings - International Conference on Distributed Computing Systems. 2018: 660-670.
[9] Shyam, Gopal Kirshna, Manvi, Sunilkumar S.2,. Virtual resource prediction in cloud environment: a Bayesian approach[J]. Journal of Network and Computer Applications. 2016: 144-154.
[10] Yuvaraj, G., Nizar Ahamed, M., Siva Rama Lingham, N., Jayaprakash, D.. Optimizing best cloud service using the Bayesian personalized ranking framework (Article) [J]. International Journal of Engineering and Technology (UAE). 2018, Vol.7(No.1Part1): 579-581.

[11] Lu Han, Xianjun Shi, Taoyu Wang. Bayesian Network Model Test Configuration Method based on Genetic and Binary Discrete Particle Swarm Combination Algorithm [J]. Journal of Physics: Conference Series. 2020: 012007.

[12] Shankar Sankararaman, Kyle McLemore, Sankaran Mahadevan, Samuel Case Bradford, Lee D. Peterson. Test Resource Allocation in Hierarchical Systems Using Bayesian Networks [J]. AIAA Journal. 2013, Vol. 51(No.3)

[13] Gao Xiaoguang, Yang Yu, Guo Zhigao. Learning Bayesian networks by constrained Bayesian estimation [J]. Systems Engineering and Electronics. 2019, 30(3): 511-524.

[14] YANG Yu, GAO Xiaoguang, GUO Zhigao. Finding optimal Bayesian networks by a layered learning method [J]. Journal of Systems Engineering and Electronics. 2019, 30(5): 946-958.

[15] Akkarajitsakul, K., Hossain, E., Niyato, D. Distributed resource allocation in wireless networks under uncertainty and application of Bayesian game (Article) [J]. IEEE Communications Magazine. 2011, Vol. 49(No.8): 120-127.