A Computing Capability Allocation Algorithm For Cloud Computing Environment

Wenxin Hu
Computer Center
East China Normal University
Shanghai, China
wxhu@cc.ecnu.edu.cn

Jun Zheng, Xiayu Hua, Yaqian Yang
Computer Center
East China Normal University
Shanghai, China
jzheng@cc.ecnu.edu.cn

***Abstract*—For several special features in the environment of cloud computing, which may be quite different from the centralized computing infrastructure currently available, the existed method of resource allocation used in the grid computing environment may not be suitable for these changes. In our paper, a new allocation algorithm based on Ant Colony Optimization (ACO) is proposed to satisfy the needs of Infrastructure as a Service (IaaS) supported by the cloud computing environment. When started, this algorithm first predicts the capability of the potentially available resource nodes; then, it analyzes some factors such as network qualities and response times to acquire a set of optimal compute nodes; finally, the tasks would be allocated to these suitable nodes. This algorithm has shorter response time and better performance than some of other algorithms which are based on Grid environment when running in the simulate cloud environment. This result is verified by the simulation in the Gridsim environment elaborated in the following section.**

Keywords-cloud computing, grid, ant colony optimization, recourse allocation, GridSim

I. INTRODUCTION

Ant colony optimization algorithm (ACO) was proposed by Dorigo in early 1990s in his research papers [1] [2]. A few years later, multi-agent systems has been published [3], as well as a new general-purpose heuristic algorithm as a novel nature-inspired metaheuristic method for the solution of hard combinatorial optimization (CO) problems ([4] and [5]). ACO stimulates the process of food searching of real ants by using artificial ones. Although the research on ACO algorithm has made significant progress and ACO has been applied widely, the algorithm still has deficiencies to be improved, such as premature convergence and difficult to determine the control parameters. In this paper, this problem is taken into account and solutions have been denoted to fix them in the algorithm.

Cloud computing derives from a long history of research and development on various approaches to IT outsourcing, in which customers draw from a utility provider's pool of capacity on a pay-as-you-go basis as an alternative to run their own infrastructure [6]. This architecture is originated from the grid computing, and a new kind of shared basic architecture, which combines the huge system pool together and provides several storage and compute resources to the users by means of Internet [7].As the predecessor of cloud computing [8], grid computing has become to the basic structure or the skeleton of this new innovation, and on the other hand, cloud computing is the commercial implementation of grid computing [9]. However, huge difference is existed between them, which is described in [10] and this is the main motivation for claiming this algorithm.

II. THE FEATURES AND GOALS OF THIS ALGORITHM

Cloud computing affords many kinds of abstract resources or services. All of that can be divide into three levels, that is, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In this paper ,we place our algorithm on the IaaS in such a framework that mentioned in [11] level which encapsulated all of the user data and sittings into user images and stored spread around in the storage points of cloud environment in the form of user slices.

Considering the limitation of profitability, the effective network bandwidth in cloud environment may be far narrower than that in grid. For example, a traditional enterprise-class computing infrastructure should have 10G hypervelocity Ethernet, and then HPC can directly connect with the Storage Area Network (SAN). In contradict, in cloud environment, such as Elastic Compute Cloud (EC2) system in Amazon Company, there is only 250M effective network bandwidth in a single line. What's more, all the virtual servers should share this only connection to access their storage system Simple Storage Service (S3).

III. ALGORITHM DESCRIPTION

Based on the features and needs described above, we claim the resource allocation algorithm as follow.

A. Workflow of computing resource allocation

Like the storage space allocation strategy in the memory or cache of a PC, the client requirement of the hardware infrastructure following the content of agreement should be allocated dynamically from the node pool, which means to allocate resources to the user all according to the instant need of the client program instead of the commission described in the agreement all the time, which can make the entire facility more efficient and prevent to produce useless nodes.

Following architecture mentioned by the Map/Reduce strategy which is described in [12], each unit in the cloud computing node cluster is composed by a master task tracker node and several slave job tracker nodes. This structure is brief in Fig. 1.

Published by Atlantis Press, Paris, France.

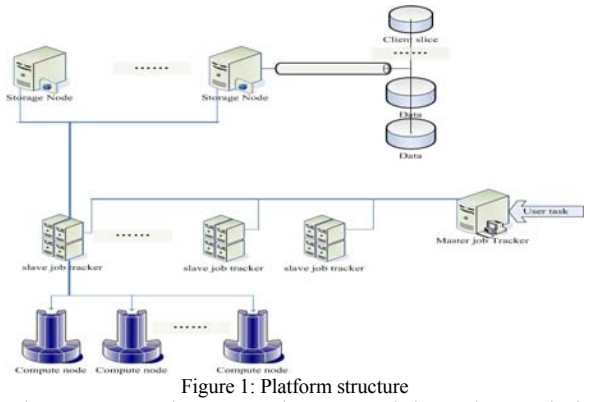

Figure 1: Platform structure

When a user task comes, the master job tracker node is responsible for all assignments, of which data resources may be contained by the user image slices spreading in different storage nodes of their slave task tracker node. The master node supervises their executions, and is reactive for the failed ones. The slave node only responses to the tasks the master node is distributed to. When a slave node receives assignments sent by the master tracker node, it will begin to search suitable computing nodes as its storage nodes. In the first step, this slave node begins to scan the useable computing resource it has. If it can satisfy the user's needs which are guaranteed by the provider, slave node will allocate them first to assignments from master node as local computing resource has high priority. Otherwise, it will discover resources in the cloud environment. These discovering should be done in a certain scale to save the network cost. Ultimately, if no resource has been found, slave tracker will announce the master tracker to move the user image slice to other slave trackers. The algorithm in this paper is mostly work in this nod node.

B. *Decision condition of computing resource quality*

Take all slave points into an undirected graph, V is an aggregation of the slave trackers, E is the network connected all slave trackers together. Searching a suitable computing node is equally to find an optimal way e ∈ V followed these guide lines:

(1) anticipated execution speed: time cost(e), that means the anticipated execution speed of the next assignment in the end point of way e.

(2) network bandwidth: bandwidth(e), that means the maximum network bandwidth offered by the way e.

(3) network delay: delay(e), that means the maximum delay in way e.

The regulation function of choosing resource and its limitation conditions are as follow:

$$res(e) = \frac{A * time\_cost(e) + C * delay(e)}{B * delay(e)} \quad (1)$$

$$S.T \begin{cases} time\_cost(e) < TL \\ bandwidth(e) > EL \\ delay(e) < DL \end{cases} \quad (2)$$

The choose of the resources and paths is to find a way that makes res(e) as small as possible and fit the three regulation conditions which have three Weight coefficients: A, B and C. TL EL and DL are their boundary regulation conditions and will have different value in different environments.

C. *Prediction of times next assignment would cost in each computing re- source*

In the entire environment, isomerism is a vital characteristic. That means, the construction, hardware and software details, capabilities, and throughout performances will be quite different. Furthermore, the situation of the network status is also complicated, so the payload of a certain line in certain time is incognizable. As the network in the environment of cloud computing is far narrower than traditional grid computing, this feature will also be changed by a large margin. As the result, the calculation of time a point needs to finish its assignments will be quite difficult. However, to allocate high efficiency and low cost computing resource to an assignment will enormously improve the entire performance. Besides, each execution would have high sensitiveness on the process ability of the node as the granularity of each task is extremely tiny. In this way, the speed prediction of potential points is quite necessary in an allocation action. Taking these characterizes described above into consideration, we design a speed prediction model that use the history record to forecast the next execution. As the payload of a computing point in the last execution is recorded, and current payload is acquirable, we design this prediction model as follow:

$$EV_m^{\alpha_{k+1}}(k+1) = \frac{\alpha_{k+1}}{\alpha_k}((1-\delta)EV_m^{\alpha_k}(k) + \delta RV_m^{\alpha_k}) \quad (3)$$

In this equation, $EV_m^{\alpha_{k+1}}$ denotes to the k-th time execution of the m-th computing resource point, the unit of which can be MIPS. α_k means the payload of the k-th prediction. $RV_m^{\alpha_k}$ represents the number m computing resource in the k-th time execution in the practice. δ is a parameter that is used to adjust the proportion of experiment value and prediction value to make the result more credible. In each computing point, system will record the last execution speed in the practice and use it to calculate the next possible execution speed by concerning with the last prediction value. At the same time, system will also record the value of α as it is a very important parameter, which has parameterized features such as the real CPU occupation percentage, assignments quantity or threads quantity.

D. *Use ant colony optimization algorithm to search the most suitable com- puting resource*

The entirely cloud environment is unknown, because the details of the resources are incognizable and the construction of network topology is mutable. In this situation, the location and quality of the computing resource points are incognizable to the storage points. Using ant

colony optimization algorithm, we can discover the computing resources in the unknown network topology, and choose the one or several most suitable ones to allocate to the user assignments, until the needs of user are totally met. When this discovering begins, slave point first send queries, that is, the "ants" in the algorithm, and these "ants" will follow the formula that choosing the point with the probability as more pheromone, more possibility. Furthermore, those ants will leave a certain dose of pheromone on the point that will be passed.

E. The choose of the next stop

In the initial status, the pheromones of each link on the points in the network topology are all the same. The formula of the possibility that the ant k choose the point I in the time d is as follow:

$$p_k^{ij} = \begin{cases} if\ q<q_0, \max_{n \notin avids(k)} \\ else, turn\ to\ function(5) \end{cases} \quad (4)$$

$$p_k^{ij} = \begin{cases} if\ j \notin avid(k), \dfrac{\{\frac{(\tau_{ij}(d))^{\alpha}}{(L_{qij}(t))^{\beta}}\}}{\{\sum_{n \notin avid(k)} \frac{(\tau_{ij}(d))^{\alpha}}{(L_{qij}(t))^{\beta}}\}} \\ else, 0 \end{cases} \quad (5)$$

In this formula, τij(d) denotes to the pheromone chroma in point j by the forward ant in point i in the time d, means the probability of choosing point j by the No. k ant in the point i. avid(k) is the avoid list of ant k. $L_{qij} = \frac{C * delay(e_{ij})}{B * bandwidth(e_{ij})}$ is the quality of the network between point i and j. α and β are the relatively Weight coefficient of the pheromone and the network quality. To avoid the quick converging which stops at the partly optimum solution, we set two random parameters $q \in [0,1]$ and $q_0 \in [0,1]$ to control the probability of the ants to choose the point which has the strongest pheromone chroma for the next stop.

F. Renewal of the pheromone chroma

In order to reflect the change of the pheromone chroma in each point, we use local renew method to modify the chroma. All slave points in which the pheromone chroma is not zero should be renewed itself following the equation as follow.

$$\tau_{ij}(t+1) = \rho * \tau_{ij}(t) \quad (6)$$

If this point has produced or received the backward ant, it should renew itself following this equation as follow:

$$\tau_{ij}(t+1) = \rho * \tau_{ij}(t) + (1-\rho) * D * \frac{1}{\sum(e_{ij})} \quad (7)$$

In that, ρ denotes to the volatility intensity of the pheromone. $\tau_{ij}(t+1)$ means the pheromone chroma in the point j detected by the point i in the time t+1. When this value is smaller than a certain value m, τ_{ij} would return to zero automatically. D is a effect enlarge modulus to put a premium on the pheromone chroma which is contained by the point on the suitable resource path referenced the resource restriction function res(e) in order to let the ants concentrate to this path a bit fast.

G. Two kind of ant data structure

Following the design of the ant described in [13], the data structure of the forward ant is in Tab. 1.

TABLE Ⅰ: Forward ant structure

Ant ID	Regulation conditions	Set of band-width	Set of dither	Set of delays	Way point Set	Avoidable node set	Scale of contentt	Source node	Hop time	Life
Ant ID	res(e) s.t	Bs	Ss	Ds	Ns	Avid	I	DN	Hops	Hp

The data structure of the backward ant is in Tab. 2.

TABLE Ⅱ: Backward ant structure

Ant ID	Actual Band-width	Actual shaking	Actual Delay	Set of Waypoints	Prediction Of computing capability	Source node
Ant ID	B	S	Ds	Ns	EV	FN

IV. WORKFLOW OF THE ALGORITHM

- Produce a forward ant in the storage point.
- Using (4) and (5), the forward ant chooses the next step point.
- When a forward ant A_f get into a slave point N_j, this point will set itself into the avoid list of this ant, and use the restrictive conditions carried by this ant and (3) to calculate to get the result that whether itself is the available point itself. If this slave point fit those conditions, it will mark itself, modify the pheromone chroma, and produce a backward ant A_b. A_b will fall back along the same road with all the information about the path and this computing resource. The end point of the path points list in this backward ant is an effective point which has available computing resource of the discovery in this tern. Meanwhile, this backward ant will gather the bandwidth information on the way. To prevent quick converging C_a threshold value parameter f is set to make the available point whose forward access frequency has passed this threshold value in order to produce the backward ant in a certain probability.
- Point N_j will modify the restrictive conditions A_f carried. The bandwidth or other factors used in this allocation will be decreased. A_f will continue going when the modified condition is still meet the need of the assignment, or it will be stopped in this point.
- Each point with the nonzero pheromone chroma, should use the function (6) and (7) renew itself in an unanimous time interval.
- If the health point is less than the value "hops" or it has no way to go (such as all the neighbor points have been set into their avoid list), this ant will die out automatically. Consequently, the health point

(HP) actually determined the scope the ant can reach.

- With the information the backward ant carried, the source point will build available resources collection, which will be sorted by available degree.
- When the found resources are not enough to meet user's need, turn to step one. Otherwise, the algorithm will stop, these resources will be allocated by using the available resources collection and in the allocation, and one rule which should be followed is that more resources should be allocated to the high efficiency assignments to decrease the payload of the network.
- When the algorithm stopped, if all founded resources are not enough to meet the user's needs promised in the SLA, move the data slices in this storage point to another should be considered.

V. SIMULATION RESULTS

Here we use GridSim [14] to implement our simulation. To simulate the cloud resource environment, we rebuild the grid resource brokers component in this toolkit in order to simulate the interaction between platform level and unified resource level in cloud instead of the relationship between collective level and resource level in grid. we overwrite the GIS class with several existed RegionalGISWithFailure classes to generate our own topology, that contain 10 thousand computing resources with random working states and connection relationships at the same time. In this class, we implement our algorithm. Just as the gridsim can only process one job (which named gridlet in this toolkit) in one process element (which named PE in GridSim), we directly use PE to be the gauge of computing capability.

We abandon the currently resource management policy used in the GridSim system. This policy maintains a table to record the entire resource using states, which may lead an unacceptable amount of communication flow. This is because when each allocating transaction happens, the table should represent the newest situation of each resource point, which means almost all relevant points should send its status back to the table holder when a job has come. Otherwise, in the cloud the topology of resource node is unknowable for the using situation, the scale effect, and varying condition of network connectivity. The algorithm presented in this paper uses self-explore mechanism which is suitable for this kind of opaque surrounding and produces a small traffic flow of ant transmission and responses of using information from only the selected nodes.

Compare with the original allocation method used in the GridSim and some of the grid system, the algorithm presented in this paper uses less network resource to finish the allocation. Additionally, when the whole scale become larger and the amount of valuable nodes become less, the advantage is obvious. Details are represented in the Fig. 2 which shows the communication message amount of these two mechanisms. The Fig. 3 shows the time cost of the two strategies used to handle the same quality and number of short jobs with 250M bandwidth. In this graph, we can see the line of the TableRecord mechanism has an obvious point after which the cost time soared sharply. The reason of the existence of this point is the depletion of the network bandwidth.

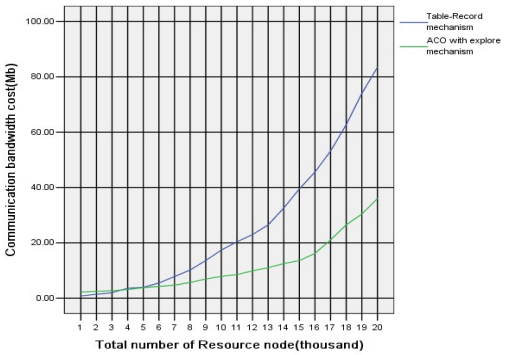

Figure 2: Communication message amount in bandwidth

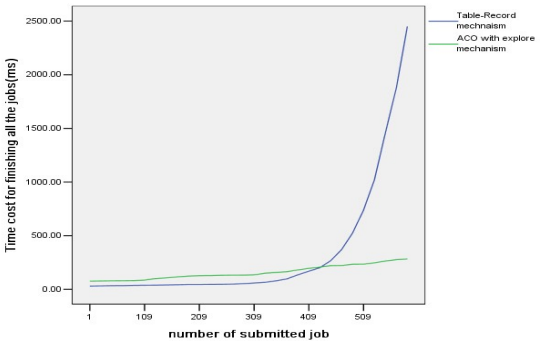

Figure 3: the time cost in two mechanism

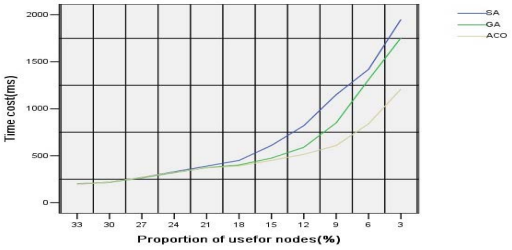

Figure 4: Comparison of these three allocation algorithm

We also do the peer comparison among GA, SA and ACO allocation algorithms. Simulation of the situation explores and searches the available points in 1000 ones to allocate 50% of them. For instance, the percentage of available point is 20%, and the total quality is 200 points. So the success is defined as searching and allocating 100 points.

Fig. 4 is the contiguous curve line of this simulation.

VI. CONCLUTIONS

As the limitation of cost, cloud computing environment is far more complicated and changeable, and since it should provide service such as Saas or Paas, the requirements of the infrastructure are more rigors. With the dynamic, shareable and large-scale characteristic in consideration, our algorithm allocates computing resource dynamically to the user assignment slices. According to the simulate results, we can

see this algorithm performs well in the job of searching and allocating resources.

REFERENCES

[1] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Dipartimento di Elettronicae Informatica, Politecnico di.

[2] M. Dorigo, Optimization, learning and natural algorithms, Italian, Ph. D. dissertation, Politecnico di Milano.

[3] M.Dorigo, V.Maniezzo, A.Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics vol.26, no.1, 1996, pp.29-41.

[4] M. Dorigo, T. Stutzle, The ant colony optimization metaheuristic: Al gorithms, applications, and advances, International Series in Operations Research and Management Science , 2003, pp.251–286.

[5] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys (C-SUR) , vol35 , no.3, 2003, pp.268–308.

[6] H. Lim, S. Babu, J. Chase, S. Parekh, Automated control in cloud computing: challenges and opportunities, in: Proceedings of the 1st workshop on Automated control for datacenters and clouds, ACM New York, NY, USA, 2009, pp.13–18.

[7] B. Ohlman, A. Eriksson, R. Rembarz, what networking of information can do for cloud computing, in: WETICE '09, IEEE International Workshops, 2009, pp. 78–83.

[8] H. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, A. Chien, The microgrid: a scientific tool for modeling comutational grids, Scientific Programming, vol.8, no.3, 2000, pp.127-141.

[9] H. Liu, D. Orban, Gridbatch: Cloud computing for large-scale data- intensive batch applications, in: Cluster Computing and the Grid, 2008. CCGRIDf08. 8th IEEE International Symposium on, 2008, pp. 295–305.

[10] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid comput- ing 360-degree compared, in: Grid Computing Environments Workshop, 2008. GCE'08, 2008, pp. 1–10.

[11] G. Mc Evoy, B. Schulze, Using clouds to address grid limitations, in: Proceedings of the 6th international workshop on Middleware for grid computing, ACM New York, NY, USA, 2008.

[12] H. Yang, A. Dasdan, R. Hsiao, D. Parker, Map-reduce-merge: simplified relational data processing on large clusters, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, 2007, p. 1040.

[13] P. D. ru YUAN yanbo, mproved QoS Routing Algorithm Based on the AntNet, MINI-MICRO SYSTEMS, vol.27.

[14] R. Buyya, M. Murshed, Gridsim: A toolkit for the modeling and simula- tion of distributed resource management and scheduling for grid mputing, Concurrency and Computation: Practice and Experience, vol.14, 2002, pp.1175-1220.