Toward Optimal Deployment of Communication-Intensive Cloud Applications

Pei Fan, Ji Wang
National Laboratory for Parallel & Distributed Processing
National University of Defense Technology
Changsha,410073,P.R.China
peifan@nudt.edu.cn,jiwang@ios.ac.cn

Zibin Zheng, Michael R. Lyu
Dept. of Computer Science & Engineering
The Chinese University of Hong Kong
Hong Kong, China
{zbzheng,lyu}@cse.cuhk.edu.hk

***Abstract*—Strongly promoted by the leading industrial companies, cloud computing becomes increasingly popular in recent years. The growth rate of cloud computing surpasses even the most optimistic predictions. A cloud application is a large-scale distributed system that consist a lot of distributed cloud nodes. How to make optimal deployment of cloud applications is a challenging research problem. When deploying a cloud application to the cloud environment, cloud node ranking is one of the most important approaches for selecting optimal cloud nodes for the cloud application. Traditional ranking methods usually rank the cloud nodes based on their QoS values, without considering the communication performance between cloud nodes. However, such kind of node relationship is very important for the communication-intensive cloud applications (e.g., Message Passing Interface (MPI) programs), which have a lot of communications between the selected cloud nodes. In this paper, we propose a novel clustering-based method for selecting optimal cloud nodes for deploying communication-intensive applications to the cloud environment. Our method not only takes into account the cloud node qualities, but also the communication performance between different nodes. We deploy several well-known MPI programs on a real-world cloud and compare our method with other methods. The experimental results show the effectiveness of our cluster-based method.**

***Keywords*-Cloud nodes; Quality-of-Service; Cloud deployment; Communication-Intensive; Clustering Analysis.**

I. Introduction

Cloud computing is Internet-based computing, where shared resources (e.g., infrastructure, platform, software, data, etc.) are provided to users on-demand, like a public utility [1], [2]. Cloud applications are usually large-scale and very complex, involving a number of distributed cloud nodes. How to make optimal deployment of cloud applications is a challenging and urgent required research problem.

Similar to traditional component-based systems [3], cloud application is also composed by a number of components. When deploying a cloud application in a cloud, the application user need to select a number of cloud nodes (e.g., servers or a virtual machines) to run the cloud applications (usually software components). There are two types of common cloud applications, *i.e.,* computation-intensive applications and communication-intensive applications. In a computation-intensive application (e.g., BOINC [4]), the cloud nodes do not communicate with each other frequently. To select the optimal cloud nodes for deployment purpose, the application designer can simply rank the available cloud nodes based on their QoS values and select the best performing ones. On the contrary, in a communication-intensive application (e.g., Message Passing Interface (MPI) applications [5]), there are a lot of communications between different distributed cloud nodes. Therefore, performance of communication-intensive application is greatly affected by the network connections between the selected nodes. For such kind of applications, selecting optimal cloud nodes using ranking-based methods is not proper, since communication performance between cloud nodes needs to be considered. Designing efficient and effective cloud node selection approaches that considers node capacity as well as communications between nodes is an important task for building high-quality communication-intensive cloud applications.

Quality-of-Service (QoS) is usually employed to describe the non-functional performance of the cloud nodes, e.g., the size of free memory, number of CPUs, network bandwidth, response time, and so on. In the cloud environment, there are usually a lot of available cloud nodes. When selecting optimal cloud nodes from a set of available cloud nodes for deployment purpose, ranking-based method [6] ranks the available nodes based on their QoS values and selects the best performing ones. A drawback of the ranking methods is that these methods cannot reflect the relations between different cloud nodes. Therefore, such kind of methods cannot be applied to the communication-intensive cloud applications, whose performance is greatly influenced by the communications between the nodes in the application. For example, assuming a user wants to deploy a MPI program on a cloud and needs to select two cloud nodes for this MPI application. As illustrated in Figure 1, there are totally four available cloud nodes in the cloud. These four node candidates form a communication matrix, where each entry in the matrix is the response time between a pair of nodes. If we rank these available node candidates via their average response time, then nodes A and D will be selected as the best performing nodes for the MPI application. However,

978-0-7695-4460-1/11 $26.00 © 2011 IEEE
DOI 10.1109/CLOUD.2011.54

	A	B	C	D		
A	0	1	2	3		6/4
B	1	0	4	2	$\Rightarrow$	7/4
C	2	4	0	1		7/4
D	3	2	1	0		6/4

Figure 1. Cloud Node Ranking by Average Response Time

from the communication matrix we can see that the response time between A and D is 3 seconds, if the user (designer of the MPI application) selects A and D, he/she may get a poor performance, since the MPI application may have a large delay time between these two selected nodes.

To attack this challenge, we propose a clustering-based selection method for communication-intensive cloud applications. Our method considers not only the QoS ranking of nodes, but also the communication relations between them. Our approach takes advantages of the cluster analysis and QoS ranking for making optimal deployment for the communication-intensive cloud applications. The contribution of this paper is two-fold:

- We identify the critical problem of selecting optimal cloud nodes for communication-intensive cloud applications and propose a clustering-based method to address this problem. Based on our method, optimal cloud nodes can be efficiently and effectively determined for communication-intensive cloud applications.
- Real-world experiments are conducted to compare our method with other methods. We deploy several well-known MPI programs on a real-world cloud, *i.e.,* PlanetLab [1]. The experimental results show the effectiveness of our proposed approach.

The rest of this paper is organized as follows: Section 2 introduces motivation and system architecture; Section 3 presents our clustering-based selection method; Section 4 describes experiments; Section 5 discusses related work and Section 6 concludes the paper.

II. System Architecture

We first introduce the cloud node selection problem. As shown in Figure 2, there are a number of available distributed nodes in the cloud. Cloud user need to deploy their cloud applications on a number of optimal cloud nodes and use it. Since cloud nodes are usually distributed in different geographic locations, deploying the application on different set of nodes may obtain different level of quality. How to select a subset of optimal cloud nodes to satisfy the requirement of cloud user is an important research problem.

To attack this critical challenge, we propose an optimal cloud node selection framework. Our framework includes a centralized server managing a number of cloud nodes. The

[1]http://www.plant-lab.org

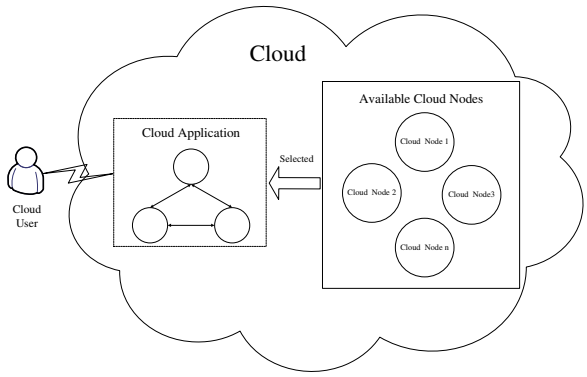


Figure 2. Motivating Example

server serves as a scheduler for different users. The server is in charge of receiving user requests, collecting the status of the nodes and scheduling cloud nodes for users. As shown in Figure 3, the steps of our optimal cloud nodes selection framework are as follows:

- **Step 1:** The cloud user submits a cloud node selection request to the server with related information, e.g., application type (*i.e.,* computation-intensive or communication-intensive), size of memory, CPU time, number of components, etc.
- **Step 2:** The request listener gets the request from the cloud user and transmits it to the scheduler.
- **Step 3:** The scheduler sends a request to the status database for getting available cloud nodes and their details QoS information.
- **Step 4:** The scheduler receives all the currently available cloud nodes from the status database, and selects the appropriate nodes by using our cluster-based cloud nodes select method, which will be introduced in Section III.
- **Step 5:** The scheduler configures the selected cloud nodes, and changes the status of nodes from free to busy.
- **Step 6:** The system send back the selected cloud nodes to the cloud user who submitted the request.
- **Step 7:** Each cloud node runs an agent program, which takes charge of monitoring the status of the cloud nodes (e.g., the size of free memory, free CPU time, etc). In addition, to get the exactly communicate information, the monitor measured the response time between nodes periodically, since the response time between nodes vary dynamically with a large range. The status received by agent will be send to the status database.

III. Cluster-Based Method

This section presents our clustering-based cloud node selection method, which is designed as a three-phase process.

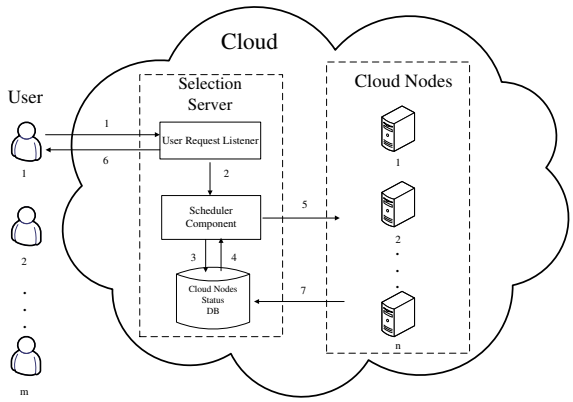


Figure 3. Component Selection Architecture

In phase 1, we introduce the method for selecting initial centroids. Then, in phase 2, the cluster analysis method is introduced. Finally, in phase 3, we present our solution for selecting cloud nodes from the generated clusters. Details of these phases are presented in Section 3.1 to 3.3, respectively.

A. Select Initial Centroid

Choosing proper initial centroids is a key step of the cluster analysis procedure. Although it is easy to choose initial centroids randomly, the cluster results are often poor. The major challenges of the random method is that the noises and outliers (e.g., cloud nodes with large response time) may be selected, which will greatly influence the quality of clusters. In general, in a data space, data objects in a lower density area are usually regarded as noise objects [7]. In this paper, we select centroids by using a density-based method, which selects the initial centroids from high-density areas.

In our approach, we use the response time between two nodes to represent the distance between them. If a node has lower response times to other nodes, it means that the node has shorter distances to the other nodes and may be in the high-density area. We define neighborhood of a cloud node p as:

Definition 1: The neighborhood of a cloud node p, denoted by $N(p)$, is defined by $N(p) = \{q \in D | dist(p,q) < DIST\}$, where $DIST$ is a threshold of response time between two cloud nodes, $D = \{p_i | i = 1, \ldots, n\}$ is a set of existing cloud nodes, and $dist(p_i, p_j)$ denotes the distance (*i.e.,* response time) between two components p_i and p_j.

Based on this definition, whether a node p is in the high-density area can be defined as:

Definition 2: A cloud node p that in the high density area should satisfy the following condition

$$Num(N(p)) > NUMBER$$

where $NUMBER$ is a threshold of the number of neighborhood nodes, and $Num(N(p))$ denotes the number of neighborhood nodes of node p.

Based on Definition 2, we can get a set of cloud nodes that are in the high-density areas and discard those cloud nodes in the low-density areas.

Let $H = \{y_i | 1 \leq i \leq m\}$ be the set of cloud nodes in the high-density areas. The initial centroids will be selected from H. The impacts introduced by noise nodes will be eliminated by this approach. The cloud node which has the largest number of neighbors is selected as the first cnetroid z_1. In other words, the first selected centroid z_1 should satisfy the following condition:

$$Num(N(z_1)) \geq max\{Num(N(y_i)) | y_i \in H\}. \quad (1)$$

We select second centroid z_2 is the node that has the greatest distance from z_1, and the third one z_3 has the greatest distance from z_1 and z_2, which should satisfy the below condition:

$$min\{dist(z_3, z_1), dist(z_3, z_2)\} =$$

$$max\{min\{dist(y_i, z_1), dist(y_i, z_2)\} | y_i \in H\}.$$

Similarly, the k^{th} centroid z_k needs to satisfy:

$$min\{dist(z_k, z_i) | 1 \leq i < k\} =$$
$$max\{min\{dist(y_j, z_i) | 1 \leq i < k\} | y_j \in H\}. \quad (2)$$

where $k \geq 2$. The Eq.(2) imply that in high-density area, we select these initial centriods that far distribute each other.

B. Cluster Analysis

Cluster analysis is useful for discovering groups and identifying interesting distributions in the underlying data. Cluster analysis can group objects (e.g., cloud nodes) based on the information found in the data describing the objects and their relations [8].

In our approach, we divide the cloud nodes into different clusters based on the response time between different nodes. Assume there are n cloud nodes distributed in a cloud, the response times between nodes can be represented as an n by n matrix, where x_{ij} is the response time between node i and node j. Apparently, x_{ij} equals to x_{ji}.

$$\begin{bmatrix} 0 & x_{12} & \cdots & x_{1n} \\ x_{21} & 0 & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & 0 \end{bmatrix} \quad (3)$$

Matrix (3) is called the response time matrix. We use p_i to represent the vector of response times from node i to other nodes. *i.e.,* $p_i = (x_{i1}, x_{i2}, \ldots, x_{in})$.

A cluster analysis algorithm is designed to divide the cloud nodes into K clusters, denoted by $C =$

$\{C_1, C_2, \ldots, C_K\}$ and $K \geq 2$. The obtained clusters satisfy the following conditions:

$$\begin{cases} C_i \neq \emptyset \quad i = 1, 2, \ldots, K \\ C_i \cap C_j = \emptyset \quad i, j = 1, 2, \ldots, K \quad and \ i \neq j \\ \bigcup_{i=1}^{K} C_i = D \end{cases} \tag{4}$$

To determine which cluster a cloud node p_i belonged to, we need to calculate the distances between p_i and different clusters. In our approach, we use the average distance between p_i and all the cloud nodes of one cluster to represent the distance between a node and a cluster. We calculate the average distance between the p_i to cluster C_k by:

$$E = \frac{1}{d} \sum_{j:p_j \in C_k} x_{ij}, \quad 1 \leq k \leq K \tag{5}$$

where d is the number of cloud nodes in the k^{th} cluster C_k. Employing Eq.(5), we can get the distances between p_i and all clusters and put cloud node p_i to the cluster that has the shortest distance value.

Our clustering algorithm for finding K clusters as shown in Algorithm 1 includes the following steps:

- Step 1 (line 1-3): select the first centriod by Definition 1,2 and Eq.(1), where F stores the initial centroids.
- Step 2 (line 4-12): Use Eq.(2) to select other centroids. For each cloud nodes in E, calculate the distance between to the cloud nodes that in F, and store the smallest distance in T(line 6-9), then select the cloud nodes as the centroid, which have the biggest value in T.
- Step 3 (line 15-18): C used to stores all K cluster sets. Before iterate, in line 13, we set $C = F$ for use the initial centroid to represent cluster. Eq.(5) used to calculate the distance between the cloud nodes and all clusters, then assign the cloud nodes to the closest cluster e.
- Step 4 (line 18-20): calculate the new center of every cluster by Eq.(7), and use the new center to replace the interrelated cluster. Step 2,3 and 4 are repeated until the RTT of every cluster don't change.

In Algorithm 1, the average response time (RTT) of cluster represents the average communication performance of cluster and calculated by:

$$RTT = \frac{1}{n} \sum_{i=1}^{n} \bar{C}_{ki}, \tag{6}$$

where $\bar{C}_k$ is the center of the k^{th} cluster and represented by a n-dimension vector. $\bar{C}_{ki}$ the i^{th} value of $\bar{C}_k$. $\bar{C}_k$ defined as

$$\bar{C}_k = \frac{1}{d} \sum_{i:p_i \in C_k} p_i \tag{7}$$

Algorithm 1: The algorithms of cluster analysis

Input: Cloud nodes set I, high-density area cloud nodes set E
Output: K clusters

1 $F = \emptyset$;
2 t=arg $max_{i \in E} Num(p_i)$;
3 $F = \{t\}$;
4 **repeat**
5 $T = \emptyset$;
6 **foreach** $i \in E$ **do**
7 d=arg $min_{j \in F} dist(p_i, p_j)$;
8 $T = T + \{d\}$;
9 **end**
10 t=arg $max_{i \in T} T(i)$;
11 $F = F + \{t\}$;
12 **until** $|F| = K$;
13 $C = F$;
14 **repeat**
15 **foreach** $i \in I$ **do**
16 e=arg $min_{j \in C} dist(p_i, c_j)$;
17 $e = e + \{p_j\}$;
18 **end**
19 **foreach** $j \in C$ **do**
20 $c_j = avg(c_j)$
21 **end**
22 **until** *RTT of every cluster don't change* ;

C. Selection

After clustering, these cloud nodes are assigned to different clusters. Therefore, we can select the cluster by the RTT of every cluster that user require. According to the RTT of each cluster, One or more suitable clusters can be selected according to the number of the requested cloud nodes.

After selecting clusters, we can rank the nodes in selected cluster by their performance, which is a tradeoff between the computing power and the communication ability:

$$perf = \lambda \times calc + (1 - \lambda) \times comm, \tag{8}$$

where $perf$ is the performance of a cloud node, and $calc$ is the value of computing power of a node, which can be calculated by the size of free memory, free CPU time, etc. In this paper the value of $calc$ is the calculated by the values of free memory and CPU:

$$calc = memory + cpu$$

The value of $comm$ is the average response time of the cloud nodes in selected cluster, which is calculate by Eq.(9). Please note that values of $calc$ and $comm$ should be standardized into the range of [0,1]. For a set of cluster that selected S and a cloud node p_i in S, $comm$ is calculate by:

$$comm = \frac{1}{|S|}\sum_{j=1}^{n} I(x_{ij}), \tag{9}$$

where $|S|$ is the number of cloud nodes in S and $|S| > 1$, and $I(x_{ij})$ defined as:

$$I(x_{ij}) = \begin{cases} x_{ij}, & p_j \in S \\ 0, & p_j \notin S \end{cases} \tag{10}$$

In Eq.(10) $p_j \in S$ means p_i and p_j are the candidate cloud nodes that will be selected. Based on Eq.(8), a number of cloud nodes that may have the best performance are selected for the user.

IV. Experiments

In this section, we evaluate our cluster-based method by real-world experiments and give a comprehensive performance comparison against other ranking methods. We first describe our experiment setup along with the benchmark, followed by the evaluation results.

A. Experiment Setup

We have deployed our experiments on PlanetLab [9], which is a global overlay network for developing and accessing broad-coverage network services. Since the nodes in PlanetLab are real machines and connected by the Internet, we can carry out our experiments in realistic real world environment. Our experimental environment consists of 100 distributed nodes which serve as cloud nodes. Our approach is implemented with JDK 1.6. The schedule node and database server are also deployed on PlanetLab nodes. In out experimental we partition cloud nodes in 4 cluster and use the following parameter settings: $\lambda = 0.5$ (Eq.(8)). $DIST = 100ms$, $NUMBER = 25$ (Definition 1 and 2). The impact of different settings of these parameters will be provided at Section 4.3 and 4.4.

In the experiments, we run different cloud node selection approaches for a MPI benchmark, called NASA NPB [10]. The reason why we choose programs in a MPI benchmark is that performance of a MPI program not only depends on the computing power of the nodes that the program runs on, but also the communication ability between the nodes. NPB is a widely used MPI benchmark, which consists of programs designed to help evaluate the performance of parallel supercomputers. The benchmark is derived from computational fluid dynamics (CFD) applications. To compare the performance of Cluster-based method against other schedule algorithm, we use the following metric via NPB:

- **Makespan:** The makespan of a job is defined as the duration between sending out a job and receiving a correct result.
- **Throughput:** The throughput of a job is defined as the total million operations per second rate (Mop/s) rate over the number of processes.

B. Performance Comparison

To study the cluster-based method performance, we compare our method with the following four methods:

- **Random :** Random-based cloud nodes selection method.
- **RankRes:** Ranking cloud nodes with respect to the free memory and CPU time. For this method, we can use the Eq.(8) by taking the value of $\lambda = 1$. Using Eq.(8) we select the k-highest nodes, where k is the number of the nodes that the user requires.
- **RankComm :** Ranking cloud nodes with respect to communication performance. Same as the RankRes method, we also use Eq.(8) but the value of λ is 0.
- **RankAll :** using Eq.(8) to ranking cloud nodes with respect to both the computing power and communication ability.

NPB applications are need to be compiled for a specific number of cloud nodes, and a given problem size. Some benchmarks (e.g., CG, MG, LU, IS, EP) can only run on a power-of-2 number of cloud nodes. The rest (SP and BT) can only run on a square number of cloud nodes. We run the experiments on 8 or 9 cloud nodes for different benchmarks and 16 cloud nodes for all the benchmarks.

Table I and Table II show the running of the different benchmarks. The numbers 8, 9 and 16 indicate the numbers of the used cloud nodes. From Tables I and Table II, we can get the following result:

Tables I and Table II show that:

- Among all the methods, our clustering-based method obtains the best performance (less execution time and high throughput). In some case, the result of clustering-based is more better than other methods (e.g., on the MG, SP, BT benchmark).On the contrary, in most case, the random select cloud nodes method got the worst result.
- In most cases, the results of *RankAll* or *RankComm* method are only worse than the clustering-based method. *RankAll* or *RankComm* cannot consider the relations between cloud nodes but is can consider one or more factors and rank it. And the result of Ranking method not stable (in some case *RankAll* better than *RankComm*, other cases *RankComm* better than *RankAll*). For these reasons, *RankAll* or *RankComm* method receives a less quality than clustering-based method.
- With increase the cloud nodes, the performance of MPI program will be increase, However, in some cases, the performance of 16 cloud nodes is lower than 8 cloud nodes, since the nodes in PlanetLab are deployed over internet, therefore the cost of communication may increase greatly with more internet connect.
- In most cases, the performance of *RankRes* method only better than Random method, since *RankRes* method

Table I
COMPARISON OF MAKESPAN (S)

	EP		IS		MG		CG		LU		SP		BT	
	8	16	8	16	8	16	8	16	8	16	9	16	9	16
Random	215.9	160.7	10977.8	5560.7	441.6	773.24	1806.0	949.9	1756.7	3495.3	506.5	519.8	498.5	230.9
RankRes	69.0	65.3	1833.4	1627.7	905.8	685.8	2063.0	1396.1	3567.3	2594.4	428.7	326.8	595.0	295.6
RankComm	122.1	176.5	1574.7	521.1	784.9	313.8	1345.2	528.0	1730.9	2037.5	401.9	154.0	276.9	113.4
RankAll	90.2	55.4	890.6	487.6	392.4	290.4	1503.4	689.4	2871.3	1739.9	326.3	258.1	354.4	394.1
Cluster	53.2	52.3	696.4	321.7	220.4	67.2	604.9	352.6	1587.2	1128.3	53.1	139.1	51.9	73.1

Table II
COMPARISON OF THROUGHPUT (MOP/S)

	EP		IS		MG		CG		LU		SP		BT	
	8	16	8	16	8	16	8	16	8	16	9	16	9	16
Random	10.29	16.2	0.01	0.02	8.90	5.03	0.83	1.58	67.91	34.13	0.19	0.22	0.46	0.99
RankRes	31.14	36.7	0.08	0.05	4.3	5.68	0.73	1.07	33.44	45.98	0.23	0.39	0.38	0.77
RankComm	12.81	12.2	0.09	0.16	4.96	12.4	1.11	2.83	69.24	58.55	0.24	0.64	0.82	2.01
RankAll	23.81	38.9	0.10	0.17	11.77	13.40	1.0	2.17	41.56	68.56	0.3	0.46	0.64	0.58
Cluster	43.1	45.6	0.12	0.57	25.33	57.91	2.56	4.24	80.12	105.75	1.82	0.74	4.4	3.12

not consider the communication performance ($\lambda = 1$), which is important to communication-intensive application.

C. Compare of initial centroids selection method

In this section, we compare our method of selecting initial centroids to random selection method. To study the impact of $NUMBER$ and $DIST$ respectively, we conduct three experiments. The first (second column of Table III), $NUMBER$ is 25 and $DIST$ by vary from 50ms to 230ms with a step value of 20. The second (third column of Table III), $DIST$ is 100ms, $NUMBER$ by vary from 15 to 33 with a step value 2. The third (fourth column of Table III), use random method. For effective compare different methods, every experiment conduct 10 times and the benchmark is EP. Table III show the result of these experiments, in Table III we represent the execute time of different method.

Table III shown in most case, our initial centroids selection method get the best performance (less execute time). In the eleventh row of Table III, represent the average makespan of all methods, the result also show that our approach get the better performance.

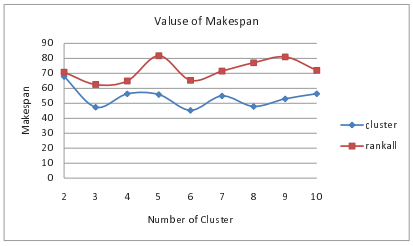
Table III
COMPARE OF INITIAL CENTROIDS SELECTION METHOD (S)

	DIST	NUMBER	Random
1	95.5	41.5	65.4
2	66.8	41.3	97.3
3	64.4	44.0	61.1
4	69.6	45.3	53.9
5	46.5	42.9	85.7
6	48.2	76.8	63.4
7	52.7	57.9	75.1
8	81.1	72.2	95.0
9	77.9	84.2	117.9
10	70.6	75.6	104.1
Average	67.3	64.9	81.9

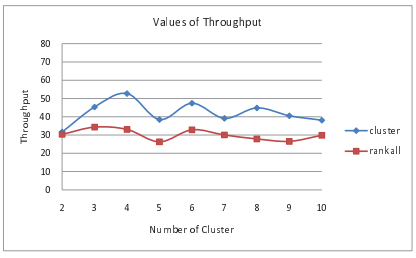
D. Impact of Parameters

1) Impact of Class number: In this section, we will analyze the impact of different number of clusters by vary it from 2 to 10 with the step value 1 (the cluster number should be more than 2), and other parameters are set same as in section 4.1. The used benchmark is EP and deployed on 8 cloud nodes. We compare the performance of using the cluster-based method with the *RankAll* method.

Figure 4 shows the makespan and throughput results when using different cluster numbers, respectively. Figure 4(a) shows that the makespan of the cluster-based method is gradually decreased and becomes stable, and the values are



(a) Makespan



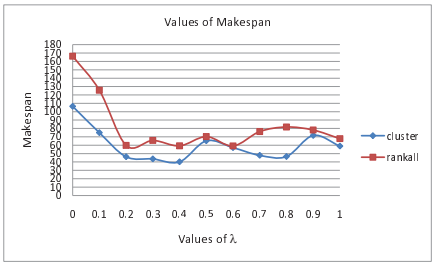
(b) Throughput

Figure 4. Impact of Cluster number

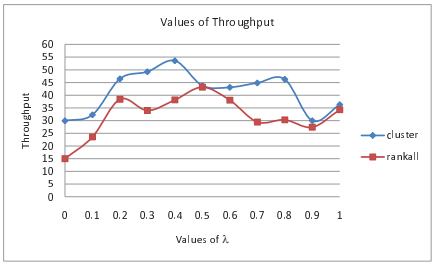
all in 50-60 seconds. Form both Figures, we can observe that: when the cluster number equals 2, the makespan and the throughput of two methods are closer. The reason is when cluster number equals 2, the most cloud nodes may be partitioned in one cluster, therefore the result of clustering-based method closer to *RankAll* method. In addition, in both figures, the clustering-based method outperforms the RankAll method.

2) Impact of λ: We change the value of λ from 0 to 1 with a step value of 0.1. Besides, we set the cluster number as 4 in this experiment. The used benchmark is EP, and the compared method is also the *RankAll* method.

Figure 5 shows the experimental results. Both methods will have the worst performance when λ is 0. The reason is only communication ability is considered, and the *RankAll* method equals the *RankComm* method when λ is 0. When λ from 0.2 to 0.4, both methods have the best performance. When λ increase from 0.4 to 1, the makespan of the cluster-based method becomes increase. In addition, we can observe that the value of λ influences the performances of both methods greatly than cluster number.



(a) Makespan



(b) Throughput

Figure 5. Impact of λ

V. Related Work and Discussion

Nodes or Components schedule and recommendation have been widely employed for volunteer computing systems, Grid systems, P2P systems, component-based systems and cloud systems [11] [12] [13] [14] [6]. QoS can be employed for describing the non-functional performance of cloud nodes[15]. Based on the cloud node QoS performance, a number of selection and schedule strategies have been proposed in the recent literature. The major approaches can be divided into three types: (1) Random appraoches (use random methods to select components). Random strategies have been employed in BOINC [4]. (2) Ranking or rating approaches (cloud nodes is ranked by the order of QoS performance). Ranking strategies have been employed in RIDGE [16] and GridEigenTrust [17]. (3) Matching approaches (matching algorithms are employed to compare the users' requirements and the QoS values of cloud nodes). Matching strategies have been employed in Condor [18]. These previous methods just consider the order of the node performance, and not consider the relationship between nodes, which is important to the communication-intensive cloud applications. In this paper, we focus on analyzing the relationship between cloud nodes to achieve optimal deployment of communication-intensive cloud applications.

To improving the performance of communication-intensive applications, a number of systems and infrastructures have been proposed [19][20][21]. Match algorithm is one of the most widely used approaches for selecting resources according to user requirements. Denis et al. [14] propose a method to select available nodes for users. However, this methods may select a number of low quality components and users should be responsible for managing node failures.

Recently, scientific applications (e.g., MPI applications) in cloud have attracted great interests, since clouds promise to be an alternative to clusters, grids and supercomputers for scientists. Christina [22] indicates the communication performance is important to scientific applications in cloud computing. There are a number of literature introducing experiences of using cloud computing for scientific applications [23] [24]. Simon Ostermann [25] analyzes the performance of EC2 cloud computing services for scientific computing. Different from these previous work, our work provides a comprehensive study on how to provide optimal deployment for the communication-intensive cloud applications.

VI. Conclusion and Future Work

In this paper, we propose a clustering-based cloud node selection approach for communication-intensive cloud applications. By taking advantage of the cluster analysis, our approach not only considers the QoS values of cloud nodes, but also considers the relationship (*i.e.,* response time) between cloud nodes. Our approach systematically combines cluster analysis and ranking methods. The experimental results show that our approach outperforms the existing ranking approaches.

In a communication-intensive application, nodes connect each other with special topology structure (e.g., a node have more connect to a certain nodes, and littler connect to other nodes).Currently, we have not considered the topology structure of communication-intensive application. More investigations are needed to consider the topology structure

in future work. In addition, the response time between nodes may depends on the sum communication load between the nodes or the state of computing and applications in each nodes. For this reason, Our future work will also include load balance for cloud nodes and fault tolerant cluster.

VII. Acknowledgement

This research is supported by the National Basic Research Program of China under Grant No.2011CB302603, the National Foundation of China under Grant NO.60725206

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, April 2010.

[2] B. Hayes, "Cloud computing," *Commun. ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[3] V. Cortellessa and V. Grassi, "A modeling approach to analyze the impact of error propagation on reliability of component-based systems," in *Proc. 10th Int'l Symp. Component-Based Software Eng.*, 2007, pp. 140–156.

[4] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Proc. 5th Int'l Conf. On Grid Computing(GRID'04)*, 2004, pp. 4–10.

[5] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface*. Cambridge, MA, USA: MIT Press, 1994.

[6] Z. Zheng, Y. Zhang, and M. R. Lyu, "Cloudrank: A qos-driven component ranking framework for cloud computing," in *Proc. 29th Int'l Symp. Reliable Distributed Systems*, 2010, pp. 184–193.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2th Int'l Conf. Knowledge Discovery and Data Mining (KDD'96)*, 1996, pp. 226–231.

[8] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis; electronic version*, ser. Wiley Series in Probability and Statistics. Hoboken, NJ: Wiley, 2005.

[9] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 3–12, July 2003.

[10] D. H. Bailey, L. Dagum, E. Barszcz, and H. D. Simon, "Nas parallel benchmark results," in *Proc. 5th Int'l Conf. High Performance Computing, Networking,Storage and Analysis (SC'92)*, 1992, pp. 386–393.

[11] P. Thysebaert, B. Volckaert, M. D. Leenheer, F. D. Turck, B. Dhoedt, and P. Demeester, "Dimensioning and on-line scheduling in lambda grids using divisible load concepts," *The Journal of Supercomputing*, vol. 42, no. 1, pp. 59–82, 2007.

[12] J. D. Sonnek, A. Chandra, and J. B. Weissman, "Adaptive reputation-based scheduling on unreliable distributed infrastructures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 11, pp. 1551–1564, 2007.

[13] D. P. Anderson and J. McLeod, "Local scheduling for volunteer computing," in *Proc. 21th Int'l Conf. International Parallel and Distributed Processing Symposium (IPDPS'07)*, 2007, pp. 1–8.

[14] D. Caromel, A. di Costanzo, and C. Mathieu, "Peer-to-peer for computational grids: mixing clusters and desktop machines," *Parallel Computing*, vol. 33, no. 4-5, pp. 275–288, 2007.

[15] X. L. Ying Zhang, Gang Huang and H. Mei, "Integrating resource consumption and allocation for infrastructure resources on-demand," in *Proc. 3th Conf. International Conference on Cloud Computing (CLOUD'10)*, 2010, pp. 75–82.

[16] K. Budati, J. D. Sonnek, A. Chandra, and J. B. Weissman, "Ridge: combining reliability and performance in open grid platforms," in *Proc. 3th Int'l Conf. High Performance Computing and Communications (HPDC'07)*, 2007, pp. 55–64.

[17] B. K. Alunkal, I. Veljkovic, G. V. Laszewski, and K. Amin, "Reputation-based grid resource selection," in *In Workshop on Adaptive Grid Middleware*, 2003, p. 28.

[18] J. Frey, T. Tannenbaum, M. Livny, I. T. Foster, and S. Tuecke, "Condor-g: A computation management agent for multi-institutional grids," *Cluster Computing*, vol. 5, no. 3, pp. 237–246, 2002.

[19] R. Buyya, D. Abramson, and J. Giddy, "An economy driven resource management architecture for global computational power grids," in *Proc. 6th Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA'00)*, 2000.

[20] N. Andrade, L. Costa, G. Germglio, and W. Cirne, "Peer-to-peer grid computing with the ourgrid community," in *Proc. 23th Brazilian Symp. Computer Networks (SBRC'05)*, 2005.

[21] I. Taylor, M. Shields, I. Wang, and R. Philp, "Distributed p2p computing within triana: A galaxy visualization test case," *In Proc. 17th Int'l Symp. Parallel and Distributed Processing (IPDPS'03)*, p. 16, 2003.

[22] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the use of cloud computing for scientific workflows," in *Proc. 4th Int'l Conf.eScience*, 2008, pp. 640–645.

[23] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific cloud computing: Early definition and experience," in *Proc. 10th Conf. High Performance Computing and Communications (HPCC'08)*, 2008, pp. 825–830.

[24] K. Keahey and T. Freeman, "Science clouds: Early experiences in cloud computing for scientific applications. cloud computing and applications," 2008.

[25] R. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "An early performance analysis of cloud computing services for scientific computing," *Technical Reprot, TU Delft, Dec 2008*.