A spectral clustering-based optimal deployment method for scientific application in cloud computing

Pei Fan, Ji Wang* and Zhenbang Chen

National Laboratory for Parallel and Distributed Processing,
National University of Defense Technology,
Changsha, 410073, China
E-mail: peifan@nudt.edu.cn
E-mail: wj@nudt.edu.cn
E-mail: zbchen@nudt.edu.cn
*Corresponding author

Zibin Zheng and Michael R. Lyu

Shenzhen Research Institute,
Department of Computer Science and Engineering,
The Chinese University of Hong Kong,
Hong Kong, China
E-mail: zbzheng@cse.cuhk.edu.hk
E-mail: lyu@cse.cuhk.edu.hk

Abstract: Similar to Grid computing systems, scientific applications in cloud are large scale distributed systems that are deployed on distributed cloud nodes. Scientific applications usually have a lot of communications between the nodes for deployment. Therefore conventional ranking methods are not appropriate for deploying scientific applications. The reason is ranking methods do not consider the relations between nodes. We propose a novel spectral clustering based deployment method that takes not only the computing qualities of cloud nodes into account, but also the communication performance between different nodes. Experimental results show the effectiveness of our method for improving the performance of scientific applications.

Keywords: grid computing; cloud computing; spectral clustering; optimal deployment; scientific application; communication-intensive.

Reference to this paper should be made as follows: Fan, P., Wang, J., Chen, Z., Zheng, Z. and Lyu, M.R. (2012) 'A spectral clustering-based optimal deployment method for scientific application in cloud computing', *Int. J. Web and Grid Services*, Vol. 8, No. 1, pp.31–55.

Biographical notes: Pei Fan is a PhD candidate in the School of Computer Science, The National University of Defense Technology. His main research interest focus on Cloud Computing and Fault tolerance.

Ji Wang is a Professor at the National Laboratory for Parallel and Distributed Processing in China. He is also the Deputy Director of the National Laboratory for Parallel and Distributed Processing in China. His research interests include high confidence software development, software engineering and distributed computing. He served as in program committees for many conferences including COMPSAC, APSEC, ATVA, EMSOFT, HASE, QSIC, ICTAC, ICFEM, ISoLA, and SCAM. He is on the editorial board of the Journal of Systems and Software. He has authored and co-authored more than 80 research papers in journals, conferences and workshops.

Zhenbang Chen is an Assistant Perofessor at the National Laboratory for Parallel and Distributed Processing in China. His research interests include formal methods for component-based system and Service-Oriented Computing, new network computing techniques and software verification.

Zibin Zheng is an Associate Research fellow at Shenzhen Research Institute, The Chinese University of Hong Kong. He received his PhD Degree from the Chinese University of Hong Kong in 2011. He received ACM SIGSOFT Distinguished Paper Award at ICSE'2010, Best Student Paper Award at 2010 ICWS'2010, First Runner-up Award at 2010 IEEE Hong Kong Postgraduate Research Paper Competition, adn IBM PhD Fellowship Award 2010–2011. He served as program committee member for many conferences including CLOUD, CLOUDCOMPUTING, SCC,etc. His research interests include service computing, cloud computing and software reliability engineering.

Michael R. Lyu is an IEEE Fellow and an AAAS Fellow, for his contributions to software reliability engineering and software fault tolerance. He is also a Croucher Senior Research Fellow. He received the PhD Degree in Computer Science from the University of California, Los Angeles, in 1988. He is currently a Professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks,Web technologies, multimedia information processing, and E-commerce systems. He has published over 270 refereed journal and conference papers in these areas. He served as in program committees for many conferences including HASE, ICECCS, ISIT, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WI.

1 Introduction

Scientific computing involves the usages of mathematical models and numerical solution techniques to solve scientific, social scientific and engineering problems (Vecchiola et al., 2009). Running scientific applications usually not only need high performance computing resources to perform large scale experiments but also requires high bandwidth to transport data. With the development of virtualisation

technology, cloud computing offers a new way to support scientific applications. The term 'Cloud Computing' became popular in October 2007 when IBM and Google announced collaboration in that domain (Lohr, 2007; Sims, 2007). Due to the abilities to offer flexible dynamic IT infrastructures, QoS guaranteed computing environments and configurable software services (Armbrust et al., 2010; Hayes, 2008), cloud computing becomes a hot topic. Compared with other computing platforms, cloud computing is deemed as the next generation of IT platforms and promising to be a cheaper alternative to supercomputers and specialised clusters, and a much more scalable platform than the largest common clusters or resource pools (Buyya, 2010; Foster et al., 2008; Kim, 2011). For example, for scientific applications, cloud computing proposes an alternative in which resources are no longer hosted by the computing facilities of researchers, but leased from big data centres only when needed. This feature makes cloud computing systems be able to provide a high performance and massive storage required by scientific applications, but with a lower cost. Some work of doing science on cloud already exists, such as Nimbus (Keahey and Freeman, 2008) and Cumulus (Wang et al., 2008). Especially, in Deelman (2008) shows that cloud computing offers a cost-effective solution for scientific computing.

Similar to traditional Grid computing systems (Caromel et al., 2007) and component-based systems (Cortellessa and Grassi, 2007), cloud computing systems are also composed by a number of cloud nodes (server, virtual machines or mobile devices) (Rodriguez et al., 2011). When deploying a computing-intensive scientific application (e.g., SETI@home) (Anderson et al., 2002) in a cloud, the cloud service provider can simply rank the available cloud nodes based on their QoS values and select the ones with the best performance for users. However, communication-intensive scientific applications (e.g., Message Passing Interface (MPI) applications) usually need the collaborations of cloud nodes, and there are a lot of communications between these nodes. Thus, the communication performance will impact the performance of an application very much. There are some factors that can influence the communications between cloud nodes, for example, the geography locations of cloud nodes, and the workloads of cloud nodes and bandwidth limits. Therefore, the performance of a communication-intensive scientific application is greatly affected by the network connections between the selected nodes. For such kind of applications, selecting optimal cloud nodes by using ranking-based methods is not suitable, since the communication performance between nodes needs to be considered. For example, assuming a user wants to deploy a MPI program on two cloud nodes, there are totally four available nodes in the cloud. As illustrated in Figure 1, these four candidates form a communication

Figure 1 Cloud node ranking by average response time

	A	*B*	*C*	*D*		
A	0	1	2	3		6/4
B	1	0	4	2	⇒	7/4
C	2	4	0	1		7/4
D	3	2	1	0		6/4

matrix, where each entry in the matrix is the response time between two nodes. If we rank these available candidates via the average response time, then nodes A and D will be selected for the MPI application. However, from the communication matrix, we can see that the response time between A and D is 3 s. If the user selects A and D, he or she may get a poor performance.

In this paper, we propose a spectral clustering analysis based method for scientific application deployments in cloud computing. Three algorithms are used in our method. Firstly, we construct a response time matrix, which represents the communication performance of available cloud nodes, and conduct a spectral clustering to partition the cloud nodes to different clusters based on the response time matrix. Secondly, we select the initial centroid which is a key step of the cluster analysis process. To get a better result, we proposed a density-based algorithm to select initial centroids. Thirdly, a greedy algorithm is proposed to select the optimal cloud nodes from clusters. The notations used in this paper are listed in Figure 2.

Figure 2 The notations used in this paper

Denotations	
M	response time matrix between cloud nodes
ω_{ij}	edge weight of similarity graph
d_i	degree of vertex
W	adjacency matrix
D	degree matrix
L	graph Laplacian
A	set of clusters
α	times
H	high density-areas
$\overline{H}$	low density-areas
$\overline{C_i}$	the *ith* centre of cluster
RTT	average time of clusters
$\psi(i,j)$	strength of preference between i and j
$Comm_{ij}$	communication performance of node pair
$Calc_{ij}$	computing performance of node pair

This paper is an extension of our conference paper (Fan et al., 2011). Compared with the work in Fan et al. (2011), there are following extensions:

- We refine the initial centroid selection method and redesign the method of selecting nodes from clusters. These new methods can improve the performance of the communication-intensive scientific applications on cloud system.
- Extensive real-world experiments are conducted.

The rest of this paper is organised as follows: Section 2 discusses the related work; Section 3 defines the problem of scientific application deployment; Section 4 presents the clustering analysis algorithms that include spectral clustering, initial centroid selection and greedy algorithm; Section 5 compares our method with other methods and Section 6 draws the conclusion and gives our future work.

2 Related work

Selecting the cloud nodes for a communication-intensive scientific application is important and challenging. In traditional distributed computing systems (e.g., Grid systems, P2P systems and component-based), node or component schedule and recommendation methods have been widely adopted. We divide the existing related work into these following categories.

- *Random*: In Anderson (2004), Anderson proposes a volunteer computing framework called BOINC, which uses a random method to select nodes for deployment. XtermWeb (Malecot et al., 2006) is another volunteer computing platform and also uses a random method.
- *Ranking or Rating*: RIDGE (Budati et al., 2007) is a reliability-aware system that uses the prior performance and behaviour of a node to make more effective scheduling decisions. Sonnek et al. (2007) propose schedule algorithms that employ the estimated reliability ratings of nodes for task allocations. QoS can be employed for describing the non-functional information of cloud nodes (Sun et al., 2011; Zhang et al., 2010). Based on QoS performance, a number of selection and schedule strategies have been proposed. Liu and Yang (2008) propose a ranking-oriented approach to collaborative filtering. CloudRank (Zheng et al., 2010a) is a QoS-driven component ranking framework for cloud computing. Zheng et al. (2010b) propose a component ranking method for fault tolerance in cloud applications.
- *Matching*: Condor (Fery et al., 2002) is a specialised workload management system for computing-intensive jobs, and uses matching to select resources. In Caromel et al. (2007), Denis et al. propose a random walk method to match available nodes for users.

Although it is easy to choose cloud nodes randomly, the results are often poor. Ranking or rating and matching methods merely consider node performance, without the relationship between nodes (e.g., the communication between cloud nodes), which is important for communication-intensive scientific applications. In this paper, we focus on analysing the relationship between cloud nodes to optimally deploy communication-intensive scientific applications.

Communication-intensive scientific applications are usually deployed on clusters or super computers. There are also some existing literatures for improving the performance of communication-intensive applications. Qin et al. (2010) propose a communication-aware load balancing method for improving the performance of communication-intensive applications by increasing the effective utilisation of networks in cluster environments. Taniar and Leung (2003) indicate load

balancing gives enormous impact to execution scheduling and the needs for parallel execution scheduling is reduced when load balancing achieved. Jimenez et al. (2008) present a study of the sharing policies of information loading in communication-intensive applications. Nishtala et al. (2009) shows that using one-sided communications and overlaps offers advantages for communication-intensive applications on BlueGene/P. Existing work usually conducts on clusters and is focused on load balancing. Within a small cluster, communication performance is not a big problem, because there are fast connections between nodes. However, communication-intensive scientific application on cloud is designed for scientists to collaborate, where large scale and distributed applications need to be executed across several data centres. Our work focuses on the optimal deployment of scientific applications on cloud platforms and the communication performance.

Recently, scientific applications in clouds have attracted great interests, since clouds provide an alternative to clusters, grids and supercomputers for scientists with a lower cost (Petruch et al., 2011). Hoffa et al. (2008) indicates that the communication performance is important for scientific applications in cloud computing. Ostermann et al. (2009) analyses the performance of the EC2 cloud computing services for scientific computing. Nebulas (Chandra and Weissman, 2009) uses distributed voluntary resources to build cloud. Koehler et al. (2010) integrating grid computing technologies with a Cloud infrastructure to support the scheduling of dynamic scientific workflows. GEMS (Costantini et al., 2010; Rampino et al., 2010) is a grid empowered molecular simulator on the european grid platform. The sizes of data have seen exponential growth in the past, and data has been distributed across the globe in a grid or cloud environment (Taniar et al., 2008). To processing data effectively, some literatures discuss the large data sets problem of scientific applications. Kwok et al. (2002) propose a parallel fuzzy c-Means clustering algorithm for large data sets. Cope et al. (2009) propose a data placement strategy for urgent computing environments to guarantee data robustness. NUCA (Hardavellas et al., 2009) is a data placement and replication strategy for distributed cache that can reduce data access latency. Yuan et al. (2010) propose a data placement strategy in scientific cloud workflows with respect to the dependencies of data. Different from this work, our work focuses on the optimal deployment of science applications on cloud platforms. However, for data-intensive applications, we believe our method is also feasible.

3 Problems analysis and architecture

In this section, we analyse the problem of deploying the communication-intensive scientific applications on a cloud. We also introduce the framework of cloud nodes selection.

Scientific applications run on a cloud platform, which is composed of many distributed cloud nodes, and each connection between two cloud nodes has a limited bandwidth. The cloud platform is denoted by a set of cloud nodes $C = \{p_i \mid i = 1, 2, \ldots, n\}$, where p_i is the ith cloud node. The performance of a communication-intensive application relies on the communication between cloud nodes. The response time between cloud nodes can be represented as an n by n

matrix M, called response time matrix, where b_{ij} is the response time between the nodes p_i and p_j. Apparently, b_{ij} is equals to b_{ji}.

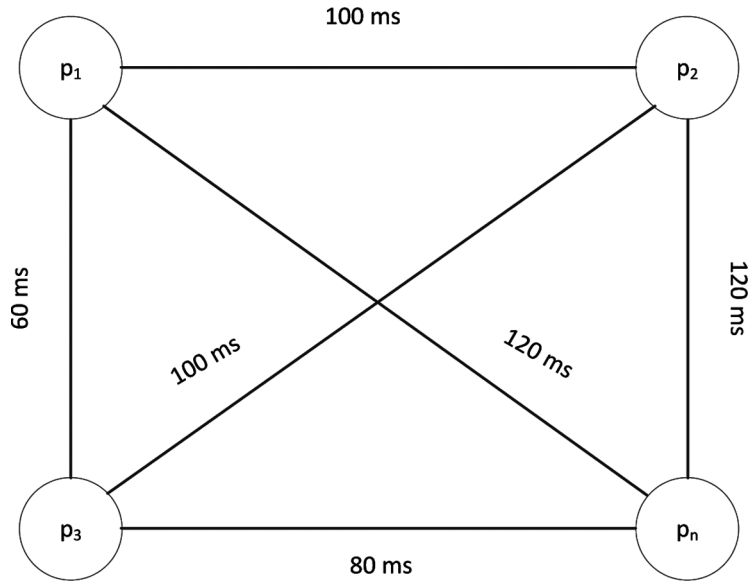
The values in M may change dynamically, since the communication performance between two cloud nodes depends on the state of each node.

$$M = \begin{bmatrix} 0 & b_{12} & \cdots & b_{1n} \\ b_{21} & 0 & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & 0 \end{bmatrix}.$$

The problem of deploying a communication-intensive scientific application can be viewed as a problem of selecting cloud nodes. If we choose nodes randomly, the performance will usually be poor. If we use a ranking-based method to select nodes according to their QoS values as indicated by Figure 1, we will also probably get a poor performance. Therefore, to get an optimal deployment of scientific applications, a new method is needed, which can not only consider the order of nodes, but also the relationship between cloud nodes.

Cloud nodes in a cloud platform form a communication graph like Figure 3, in which the value of each edge represents the response time between two nodes. Form Figure 3, we can restate the deployment problem of scientific applications as follows: we want to find a partition of the graph such that the edges between different groups have a higher values (which means that nodes in different clusters are far away from each other) and the edges with a group have a lower values (which means that nodes within a same cluster are close to each other).

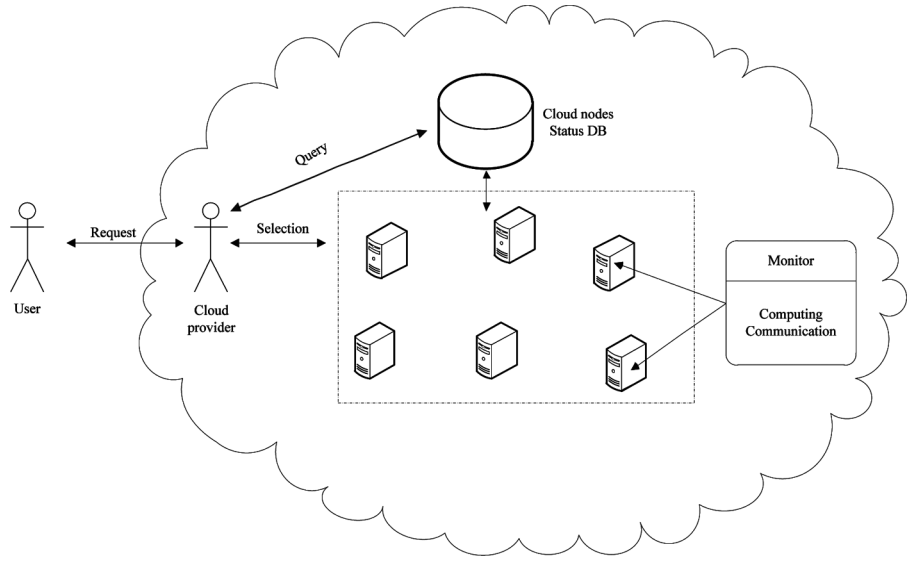
Figure 3 Communication graph of cloud nodes



For this problem, we propose an optimal cloud node selection framework that is shown in Figure 4.

- A cloud user submits an application deployment request to the cloud service provider with related information, such as computing-intensive or communication-intensive.
- The cloud service provider gets the information of available nodes that is stored in the clouds status database, and then uses our deployment method that will be introduced in Section 4 to select the candidate nodes for the user.
- Cloud node status database records the information of available nodes, including the size of free memory, heartbeat time, etc., and the information is send by each cloud nodes periodically.
- Each node runs a monitor program, which takes charge of monitoring the computing and communication performances of the cloud node. To measure the computing and communication performance of a node, we use the average value during a period as the value of each performance. More details will be introduced in Section 5.

Figure 4 Cloud nodes selection framework



4 Cluster analysis

This section presents our spectral clustering based method of cloud node selection, which is explained in three steps. At the beginning, we will introduce the spectral clustering and give the basic steps of our node selection method. There exists a key step in spectral clustering that uses a k-means algorithm to partition nodes.

Therefore, we introduce the method of selecting initial centroids in detail at the second step. Finally, we present how to select cloud nodes from the generated clusters.

4.1 Spectral clustering

Clustering is one of the most widely used techniques for exploratory data analysis, with applications ranging from statistics, computer science and biology to social sciences and psychology (Jain et al., 1999). The intuition of clustering is to separate points into different groups according to their similarities (Kaufman and Rousseeuw, 2005). In this paper, we want to separate cloud nodes into different clusters, and the nodes in a same cluster have a shorter response time.

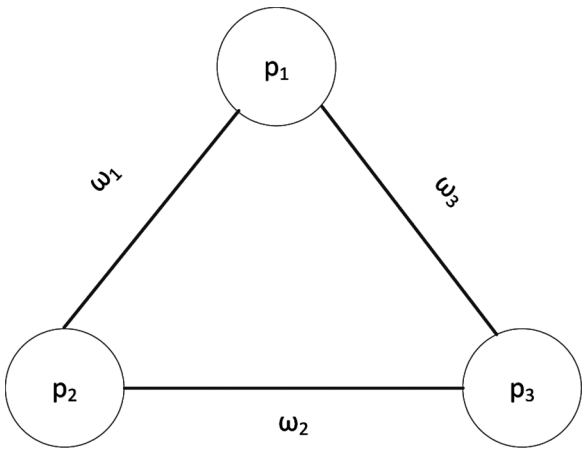
Given a cloud platform C and the bandwidth matrix M we represent the cloud nodes in the form of a similarity graph. Figure 5 shows a similarity graph. A similarity graph is an undirected graph $G = (V, E)$, where V is the vertex set and E is the vertex relation set. Each vertex v_i in V represents a cloud node. Two vertices are connected if there is a relation between these two vertices in E, and the edge is weighted by ω_{ij} to indicate the communication performance between two cloud nodes. The weight of each relations is usually standardised into the range [0, 1], and the standardisation is as follows.

$$\omega_{ij} = 1 - \frac{b_{ij} - \min b_{ij}}{\max b_{ij} - \min b_{ij}}. \tag{1}$$

From equation (1), we can observe that a higher value ω_{ij} means a shorter response time. The similarity graph can be representing as a weighted adjacency matrix W that is defined as.

$$W = \begin{bmatrix} 0 & \omega_{12} & \cdots & \omega_{1n} \\ \omega_{21} & 0 & \cdots & \omega_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{n1} & \omega_{n2} & \cdots & 0 \end{bmatrix}.$$

Figure 5 Similarity graph



If ω_{ij} is 0 it means that the vertices v_i and v_j are not connected ($\omega_{ij} = 0$, and $i = j$ mean a nodes cannot connect to itself). As G is undirected we require $\omega_{ij} = \omega_{ji}$. Let D denote a diagonal matrix defined as

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}.$$

In the matrix D, d_i is the degree of a vertex v_i and defined as

$$d_i = \sum_{j=1}^{n} \omega_{ij}. \tag{2}$$

A clustering analysis algorithm is designed to divide the cloud nodes into different clusters, where the edges between different clusters have lower weight values and edges within a cluster have higher values. Assume cloud nodes are divided into k clusters, denoted by $\{A_1, A_2, \ldots, A_k\}$ where each A_i is a set of cloud nodes and $k \geq 2$, and the clusters should satisfy the following conditions:

- $A_i \neq \emptyset$, where $i = 1, 2, \ldots, k$;
- $A_i \cap A_j = \emptyset$, where $i, j = 1, 2, \ldots, k$ and $i \neq j$;
- $\cup\{A_i, A_2, \ldots, A_k\} = V$.

The method for spectral clustering is the graph Laplacian matrix. There exists a whole field dedicated to the study of those matrices, called spectral graph theory (Spielman, 2007). Laplacian matrix is defined as

$$L = D - W. \tag{3}$$

An overview of its properties can be found in Mohar (1992), Mohar and Juvan (1997). The following proposition summarises the most important facts of spectral clustering (Luxburg, 2007).

- The smallest eigenvalue of L is 0, and the corresponding eigenvector is the constant one vector 1.
- L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$.

Using spectral clustering algorithms for finding k clusters is shown in Algorithm 1 The algorithm includes the following steps:

- Step 1 (line 1): Use response time matrix M, equations (1) and (2) to compute W and D.
- Step 2 (line 2–3): Get the graph Laplacian L via equation (3). L is an n by n matrix, and we calculate the first k eigenvectors that means the eigenvectors corresponding to the k smallest eigenvalues.

- Step 3 (line 4–6): U is the matrix containing the eigenvectors $u_1, \ldots, u_k$ as columns. In line 6, we use the k-means algorithm (Bach, 2003; Luxburg, 2007) to partition the rows of U to different clusters.
- Step 4 (line 7): Partition the cloud nodes to the relevant clusters

The basic idea of the spectral clustering can be explained as a graph cutting problem, and more details can be referred to Hagen and Kahng (1992), Shi and Malik (2000).

Algorithm 1: Spectral clustering algorithm

Input: Response time matrix M, number of k of clusters to partitioned
Output: k clusters

1 Computing W and D, then construct a similarity graph;
2 Compute the Laplacian matrix $L = D - W$;
3 Compute the first k eigenvectors $u_1, u_2, \ldots, u_k$ of L;
4 Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, u_2, \ldots, u_k$ as columns;
5 For $i = 1, \ldots, n$ let $y_i \in \mathbb{R}^k$ be the vector corresponding to the ith row of U;
6 Cluster the nodes $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into the clusters $C_1, \ldots, C_k$;
7 Cluster $A_1, \ldots, A_k$ with $A_i = \{p_j | y_j \in C_i\}$ $j = 1, \ldots, n$;

In the step 3 of the spectral clustering algorithm, a k-means algorithm to partition cloud nodes is used. Choosing initial centroids is very important for the k-means algorithm, which will be introduced in the next subsection.

4.2 Select initial centroid

Choosing initial centroids properly is a key step for cluster analysis. Although it is easy to choose all initial centroids randomly, the result would be very poor. The major problem of the random method is that the noises and outliers (e.g., cloud nodes with longer response time) may be selected, which will greatly influence the quality of the resulted clusters. In general, in a data space, the data objects in a lower density area are usually regarded as noise objects (Ester et al., 1996). In our previous work (Fan et al., 2011), we select centroids by using a density-based method, which selects the initial centroids from high-density areas.

In the density-based method, U is the matrix containing the vectors $u_1, \ldots, u_k$ as columns. u_i can be denoted by $u_i = (u_{i1}, u_{i2}, \ldots, u_{in})$, y_i is the vector corresponding to the ith row of U (cf. Algorithm 1). Y is defined as the set of y_i and denoted by $Y = \{y_i | i = 1, 2, \ldots, n\}$, and y_i can be viewed as a point denoted by $y_i = (y_{i1}, y_{i2}, \ldots, y_{ik})$. To judge whether a point is in a high-density area, we give the following definition.

Definition 1: A point y_i in a high density area should satisfy the following condition

$$\frac{1}{\alpha}\bar{u}_j \leq |y_{ij}| \leq \alpha \bar{u}_j \quad 1 \leq i \leq n, \quad 1 \leq j \leq k \tag{4}$$

where $\bar{u}_j$ is the average value of u_j and denoted by $\bar{u}_j = \sum_{i=1}^{n} |u_{ij}|$, α is the times.

Based on Definition 1, we can get a set of points that are in the high-density areas, and the rest points are in the low-density areas. Let H be the set of points in the high-density areas, and $\bar{H}$ is the set of points in the low-density areas. Apparently, $H \cup \bar{H} = Y$. We select the initial centroids from H, and impacts introduced by noise points will be eliminated by this approach. The first centroid z_1 we random select from H, and the second centroid z_2 is the point that has the greatest distance from z_1. We use response time between z_1 and z_2 to represent the distance between z_1 and z_2, and denoted by $\text{dist}(z_1, z_2)$. The third one z_3 has the greatest distance from z_1 and z_2, which should satisfy the below condition:

$$\begin{aligned}&\min\{\text{dist}(z_3, z_1), \text{dist}(z_3, z_2)\}\\&\quad = \max\{\min\{\text{dist}(y_i, z_1), \text{dist}(y_i, z_2)\} \mid y_i \in H\}.\end{aligned} \tag{5}$$

Similarly, the kth centroid z_k needs to satisfy the following condition, where $k \geq 2$.

$$\begin{aligned}&\min\{\text{dist}(z_k, z_i) \mid 1 \leq i < k\}\\&\quad = \max\{\min\{\text{dist}(y_j, z_i) | 1 \leq i < k\} | y_j \in H\}.\end{aligned} \tag{6}$$

The equation (6) implies that, in high-density areas, we select initial centroids that are far away from each other.

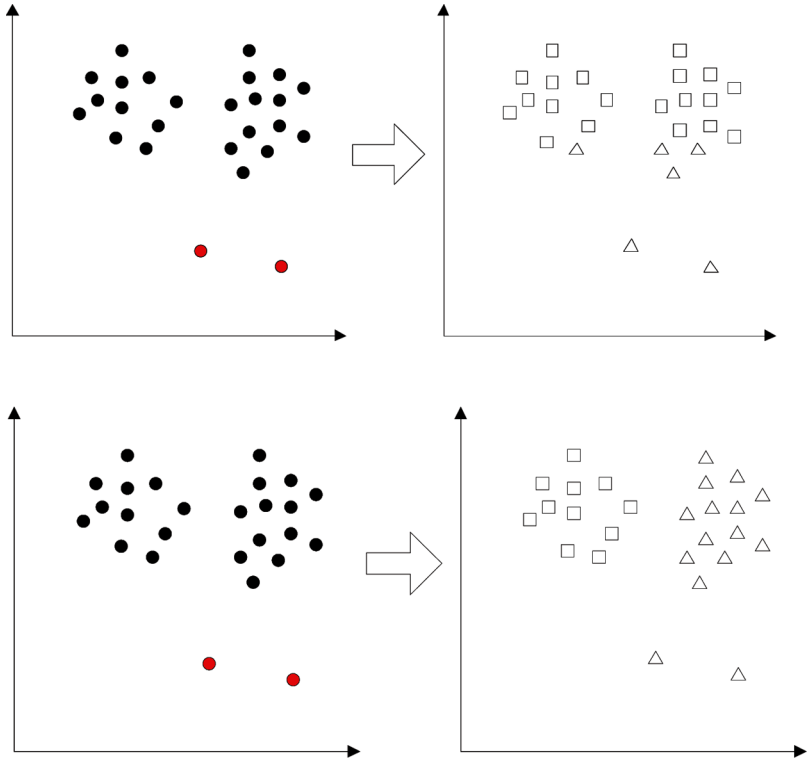
In Fan et al. (2011), after selecting initial centroids, then use the clustering method to partition all nodes into different clusters. However, there is a problem for this method in some scenarios, since the points in low-density may influence the quality of resulted cluster.

We use a simple example of two clusters in Figure 6 to demonstrate the problem, where the left figures are node distribution figures and right figures are the resulted (different shapes represent different clusters). From Figure 6, we can observe that there are two high-density areas and a low-density area. Intuitively, these two high-density areas should be in different clusters. However, as indicated in Figure 6(a), we can observe that most of nodes in two high-density areas are partitioned in a same cluster and the others in another cluster. The reason is that the red points in low-density area influence the clustering process. To address this problem, we refine the k-means algorithm, whose basic idea is to cluster the nodes only in high-density areas. Figure 6(b) shows the result of only clustering the nodes in high-density areas. Figure 8 is the detail of k-means algorithm.

The algorithm in Algorithm 2 is used at the line 6 of the Algorithm 1. In Algorithm 2, we use k-means algorithm to cluster only the points in high-density area and calculate the centre of each cluster (line 1–2). At the end of the algorithm (line 3), we use Euclidean distance to calculate the distance between each point in low-density area and the centre of each cluster, then assign the point to the closest cluster. Based on this method, the influence introduced by the points in low-density area is eliminated. The centre of a cluster C_i is denoted by $\bar{C}_i$, and defined as follows, where d is the number of the points in C_i.

$$\bar{C}_i = \frac{1}{d} \sum_{j:y_j \in C_i} y_j \tag{7}$$

Figure 6 Clustering in all nodes and high-density areas: (a) clustering all nodes and (b) clustering the nodes in high-density areas (see online version for colours)



Algorithm 2: Refined k-means algorithm

Input: set of points in high-density H, and low-density $\bar{H}$.
Output: k clusters.
1 Cluster the points in H with the k-means algorithm into clusters $C_i, \ldots, C_k$;
2 Calculate the centre of C_i, where $i = 1, \ldots, k$;
3 Calculate the distance between $y_j \in \bar{H}$ and the centre of C_i, then assign the point $y_j \in \bar{H}$ to the closet cluster;

4.3 Greedy select algorithm

After clustering, cloud nodes are divided into different clusters, we use the average response time (RTT) of a cluster to represent the average communication performance of the cluster, which is calculated as follows.

$$RTT = \frac{1}{n} \sum_{j=1}^{n} \bar{A}_{ij} \tag{8}$$

where $\bar{A}_i$ is similar calculated by equation (7). According to the RTT of each cluster, one or more suitable clusters can be selected according to a user request.

We use I to denote the set of selected cluster, each cluster is a set of cloud nodes, and the performance of cloud nodes is represented as a matrix $\psi \in \mathbb{R}^{|I|\times|I|}$, where $|I|$ is the number of the cloud nodes in I. The value of elements in matrix denoted by $\psi(p_i, p_j)$ indicates the strength of a node pair and is defined as

$$\psi(p_i, p_j) = \lambda \times Comm_{ij} + (1 - \lambda) \times Calc_{ij}. \tag{9}$$

In the above formula, $Comm_{ij}$ is the value of the communication performance between the cloud nodes p_i and p_j, $Calc_{ij}$ is the value of the computing power of the nodes, and λ is the value to tradeoff between communication performance and computing power. Note that the values of $Comm_{ij}$ and $Calc_{ij}$, should be standardised into the range [0, 1]. The method of obtain $Comm_{ij}$ and $Calc_{ij}$ will be introduced in Section 5.1. Matrix ψ is symmetric, and we set $\psi(p_i, p_i) = 0$ for each cloud node.

Given a preference function $\psi(p_i, p_j)$ that assigns a score to every pair of cloud nodes, we want to choose a set of nodes from I that agrees with the pair-wise preferences as much as possible. Let ρ be an optimal set of cloud nodes select from I. We define an objective value function.

$$V^{\psi}(\rho) = \sum_{i,j\in\rho} \psi(i, j). \tag{10}$$

Our goal is to produce a set ρ that maximises the above objective value function. One possible approach is to search through the possible set and select the optimal set ρ that maximises the value function. However, there are $C_n^{|\rho|}$ possible cases for n nodes, it is impossible to iterate all the cases when the value of n is large. To address this problem, we propose a greedy algorithm to find an approximately optimal set, and the algorithm is shown in Algorithm 3.

Algorithm 3: Greedy select algorithm

Input: a cluster set I, a matrix ψ, and a required number m of cloud nodes.
Output: an optimal set ρ.

```
F is assigned with the node set of I;
foreach p_i ∈ F do
    π(p_i) = Σ_{p_j∈F} ψ(p_i, p_j);
end
while |F| ≠ m do
    p_t is the node in F with the smallest π(p_t);
    F = F − {p_t};
    foreach p_k ∈ F do
        π(p_k) = π(p_k) − ψ(p_k, p_t);
    end
end
ρ = F;
```

Algorithm 3 includes the following steps:

- Step 1(line 1–4): for each cloud node, calculate its sum of preference values to all the other nodes. A node with less value indicates it is more likely to be excluded from the node set for the deployment.
- Step 2 (line 5–11): cloud nodes are deleted by picking the node p_t that has the minimum preference value, and the preference values of the remaining nodes are updated by removing the effect of the deleted node p_t. This step continues until the number of remain nodes is equal to m.
- Step 3 (line 12): at the end of this algorithm, the remained nodes are in F. Thus we set $\rho = F$, which means F is the approximately optimal node set.

After using this algorithm, the required cloud nodes are selected, and the cloud application can be deployed on these nodes. In the following section, we will take some experiments to justify our method.

5 Experiments

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

5.1 Experiment setup and benchmark

We carried our experiments on PlanetLab (Chun et al., 2003), 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 conduct our experiments in a realistic real world environment. Our experimental environment consists of 125 distributed nodes that serve as cloud nodes. Our framework is implemented with JDK 1.6. The schedule node and database server are also deployed on PlanetLab. In Section 4.3 we introduced $Comm_{ij}$ and $Calc_{ij}$ to represent the communication performance and computing power of a node pair, To obtain the accurate values of $Comm_{ij}$ and $Calc_{ij}$, our framework measures the response time between two nodes periodically, and uses the average response time during a period as the value of $Comm_{ij}$. The computing power of a cloud node pair is difficult to measure, since cloud nodes are usually heterogeneous. To measure computing power, we run a benchmark (e.g., calculating π) on each cloud node periodically, and use the average execution time of two nodes to as the value of $Calc_{ij}$. Our framework ran about 80 days, and we conducted above 5700 times to measure computing power and above 4560 times for communication performance.

In our experiments, we used different cloud node selection methods for two MPI benchmarks: NPB (NAS Parallel Bechmarks) and IMB (Intel MPI Benchmark) (Miguel et al., 2007). NASA NPB is a widely used MPI Benchmark, which consists of programs designed to help evaluate the performance of supercomputers. The benchmark is derived from computational fluid dynamics (CFD) applications. The

Intel MPI Benchmarks, version 3.2, are designed to measure the performance (in terms of latency and/or throughput) of the most representative MPI primitives running on a given platform. We use IMB to test the communication performance of selected cloud nodes. To compare the performance of our method against others, we use the following two metrics via for NPB and IMB, respectively.

- *Makespan for NPB*: The makespan of a job is defined as the duration between sending out a job and receiving a correct result.
- *Latency for IMB*: The latency defined as the average performance of latency when run IMB on the selected cloud nodes.

To obtain precise results, all benchmark run 3 times, and we use the average results.

5.2 Performance comparison

To justify the effectiveness of our spectral cluster based method, we compare our method with the following four methods:

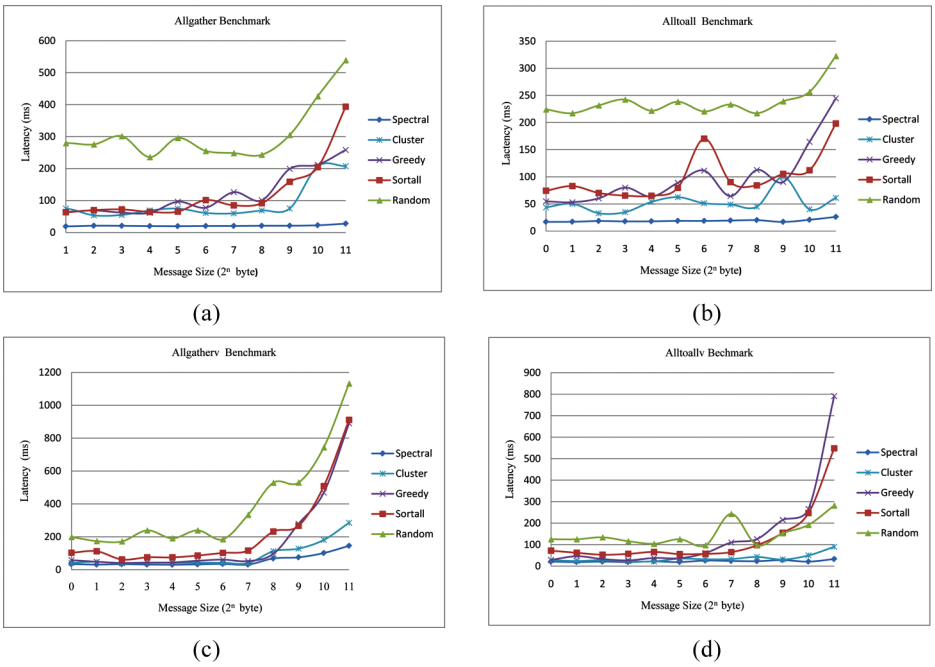
- *Random*: random-based cloud node selection method. This method is used in Anderson (2004) and Malecot et al. (2006).
- *RankAll*: using equation (9) to rank cloud nodes with respect to both the computing power and the communication performance, then select the k-highest nodes.
- *Greedy*: this method is proposed for ranking a set of cloud nodes, which directly uses equation (9) and the Algorithm 3 to rank the cloud nodes and select the k-highest nodes. This method is similar to Liu and Yang (2008) that uses greedy algorithm to select resources.
- *Cluster*: The method is proposed by us (Fan et al., 2011).

We first conducted IMB experiments. The IMB applications include three benchmarks (IMB-MPI1, IMB-EXT, IMB-I/O), and we wanted to run IMB on 8 cloud nodes. In our experiments, we used IMB-MPI1 to evaluate the communication performance between cloud nodes. IMB-MPI1 contains serials of benchmarks, and we choose ***Allgather***, ***Allgatherv***, ***Alltoall*** and ***Alltoallv*** benchmark. In IMB experiments, since IMB-MPI1 can only evaluate the latency between two cloud nodes, we set the value λ to be 1 (cf. equation (9)), and partitioned the available cloud nodes into 4 clusters. Each IMB benchmark should run with some message lengths, in our experiments, the message lengths are varied from 1, 2, 4, ..., 2048 bytes (2^0–2^{11}).

Figure 7 shows the results of running IMB benchmarks. From the figure, we have the following observation:

- In ***Allgather*** an ***Alltoall*** benchmark, spectral clustering method obtains the best communication performance, i.e., has the least latency time under all the different message sizes. Same result is gotten also in ***Allgatherv*** and ***Alltoallv***.
- In most cases, the results of cluster-based method are only worse than those of the spectral cluster method. On the contrary, the random method always gets the worst result, which fits our intuition.

Figure 7 Results of IMB: (a) allgather; (b) alltoall; (c) allgatherv and (d) alltoallv (see online version for colours)



- With the increasing of message size, the latency of each cloud nodes pair increases. In addition, we can observe that the performance of each method is obvious greatly decrease under bigger message length condition, since nodes should consume more time to transfer message.

NPB applications 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 ran the experiments on 8 or 9 cloud nodes for different benchmarks and 16 cloud nodes for all the benchmarks. In NPB experiments we partitioned cloud nodes into 4 clusters and used the following parameter setting: $\lambda = 0.5$ (NPB needs both communication and computing power). The impact of different settings of these parameters will be provided at 5.4, 5.5 and 5.6.

Table 1 show the running of the NPB benchmarks. The numbers 8, 9 and 16 indicate the numbers of the used cloud nodes. These experimental results in Table 1 show that:

- Among all the methods, spectral clustering method obtains the best performance (less execution time). The performance of cluster-based method is only worse than our method. In most cases, the random method has the worst performance. This conclusion is also valid for IMB benchmarks.
- With the increasing of the cloud nodes, the performance becomes better. However, in some cases, the performance of using 16 cloud nodes is worse

Table 1 Comparison of Makespan (s)

		Spectral	*Random*	*Rankall*	*Greedy*	*Cluster*
EP	8	21.9	137.8	65.9	57.1	34.2
	16	20.3	59.7	60.2	87.6	43.7
IS	8	60.5	83.9	720.8	685.3	174.2
	16	40.4	3097.1	1771.5	2482.8	84.7
MG	8	28.2	785.4	470.0	282.3	239.1
	16	90.7	441.8	158.3	163.0	117.5
CG	8	47.3	1420.6	605.3	689.9	320.2
	16	317.5	2526.2	1957.0	1667.3	463.1
LU	8	245.9	3057.4	3033.3	2227.6	1317.0
	16	814.3	3318.3	1784.0	1724.4	975.6
SP	9	56.0	189.6	321.0	376.9	63.3
	16	39.3	610.2	218.6	304.6	88.3
BT	9	51.9	145.8	285.2	362.4	73.9
	16	39.5	185.2	96.0	227.9	77.7

than 8 cloud nodes. It is because the nodes in PlanetLab are deployed over Internet, and the cost of communication may increase greatly with more internet connections.

- The performances of Ranking-based methods (**RankAll** and **Greedy**) are worse than that of spectral cluster-based methods, since these ranking based methods cannot consider the relations between cloud nodes.

Based on the results of IMB and NPB, we can observe that our spectral cluster based method can obtain a better performance. On the contrary, because the ranking-based methods cannot reflect the relationship between cloud nodes, the performance of them is worse.

5.3 Compare of different clustering set method

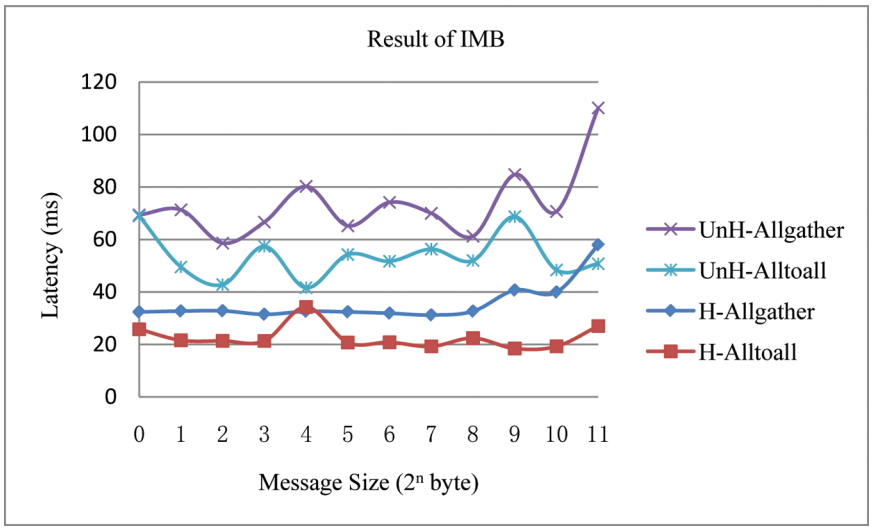
In Section 4, we introduced the high-density based method to select initial centroids and then clustering the cloud nodes in high-density areas. To justify the effectiveness of this method, we compare it with the method in Fan et al. (2011) (selecting initial centroids from high-density areas but clustering all cloud nodes). The NPB benchmarks are used, and the parameters are set the same as in those in Section 5.1. Table 2 and Figure 8 show the results. In Table 2, High means our method, and Unhigh is the method in Fan et al. (2011). From Table II, we can observe that our method gets a better performance. Unhigh method gets a similar performance in some benchmark (EP, SP and BT), but in the other benchmarks (IS, MG, CG, LU), the performance of Unhigh is worse. In Figure 8, we use Allgather and Alltoall in the IMB benchmarks. And it is obvious that our method (H-Allgather, H-Alltoall) is better.

In the result, Table 2 and Figure 8 indicate that selecting initial centroids from high-density area and then clustering in high density can be have a better performance.

Table 2 Result of NPB (s)

	EP	*IS*	*MG*	*CG*	*LU*	*SP*	*BT*
High	19.3	51.3	26.7	59.2	235.9	34.3	37.1
Unhigh	39.3	197.2	113.7	214.3	906.4	40.8	37.7

Figure 8 Result of IMB (see online version for colours)



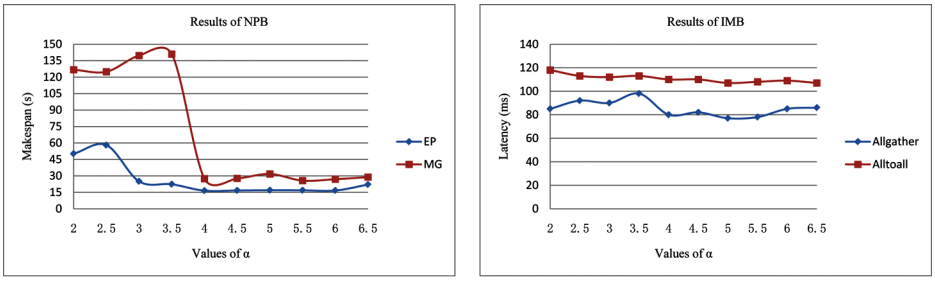
5.4 *Impact of* α

In this subsection, we will analyse the impact of the values of α (cf. equation (4)), since the value of α greatly affects the result of clustering. Therefore, it affects the deployment of cloud application. To study the impact of α, we conducted a serial of experiments. In these experiments, the values of α vary from 2 to 6.5 with a step of 0.5 and we run the benchmark on 8 cloud nodes. Figure 9(a) shown the result of EP and MG benchmarks in NPB. In Figure 9(a) we can observe that each benchmark gets the worst performance (longest execute time) when α is between 2 to 3.5. The reason is some noisy points are partitioned in high-density areas and affect the clustering result when α is set with small values. When α increase from 4 to 6.5, the execute time decrease. Figure 9(b) show the results of IMB with different α (the message size is 8192 bytes and $\lambda = 1$). We can observe a similar result to Figure 9(a) see that the results are not obvious like Figure 9(a). We can also find that Allgather and Alltoall got the worse communication performance when α is from 2 to 3.5. When α is from 4 to 6.5, both benchmarks have the best performance. From these two figures, we can observe all benchmarks obtains the best performance when α from 4 to 6.5.

5.5 *Impact of* λ

In this subsection, we analyse the impact of λ. In Fan et al. (2011), the value of λ is indicated to greatly affect the performance of communication-intensive

Figure 9 Impact of α: (a) results of NPB and (b) results of IMB (see online version for colours)

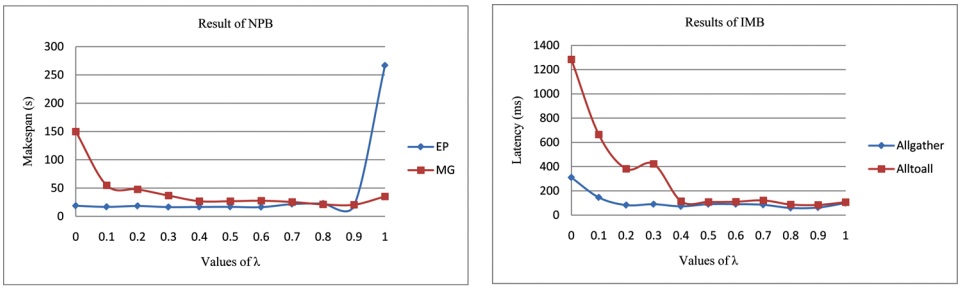


(a) (b)

cloud applications, since λ represents the tradeoff between computing and the communication abilities. However, different application should have different values of λ. To study the impact of this parameter, we also conducted a serial of experiments. In these experiments, the value of λ varies from 0 to 1 with a step of 0.1 and 8 cloud nodes we used. The benchmark which used in this subsection is same as Section 5.3. From Figure 10(a) we can observe that different benchmarks obtain different performance under a same value of λ, e.g., MG has the worst performance when $\lambda = 0$, but the performance of EP is better, On the contrary, EP gets the worst performance when $\lambda = 1$, but MG has better performance. In addition, we can found both benchmarks obtain better performance when the value of λ falls into the range [0.4–0.6].

Figure 10(b) shows the results of IMB with different value of λ (the message size is 8192 bytes). In Figure 10(b) Allgather and Alltoall obtain the worst communication performance when $\lambda = 0$, because communication performance is not considered. With the increasing of λ, the performance becomes better.

Figure 10 Impact of λ: (a) results of NPB and (b) results of IMB (see online version for colours)



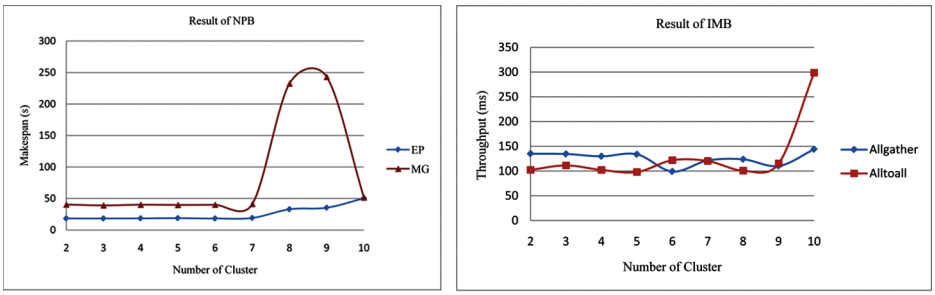
(a) (b)

From the experimental results, we can observe that the value of λ is related to the type of an application. For a computing-intensive application, λ should be set as a small value. On the contrary, λ should be set a higher value when the application is communication-intensive. Meanwhile, all benchmarks obtain an acceptive performance when λ is in [0.4–0.8].

5.6 Impact of cluster number

In this subsection, we analyse the impact of different number of clusters by ranging it from 2 to 10 with the step value 1(the cluster number should be more than 2), $\alpha = 5$, $\lambda = 0.5$ for NPB and $\lambda = 1$ for IMB, all benchmark run on 8 nodes. Figure 11(a) shows that the spectral cluster based method obtains the best performance when cluster number from is 2 to 7. When cluster number from 7 to 10, the performance gradually decreased, it is because the select nodes from clusters may across multiple clusters if the cluster number is bigger. Similar to NPB, as indicated in Figure 11(b), IMB (the message size is 8192 bytes) obtains the best performance when cluster number from 2 to 9, and the performance gradually decreased from 9 to 10.

Figure 11 Impact of cluster number: (a) results of NPB and (b) results of IMB (see online version for colours)



(a) (b)

6 Conclusion

In this paper, we propose a spectral clustering based cloud node selection method for communication-intensive scientific applications. We model the scientific applications deployment problem as a graph partition problem. Based on this problem model, we use a spectral clustering analysis method to partition cloud nodes in different groups. By taking the advantage of the spectral clustering analysis, our approach not only considers the QoS values of cloud nodes, but also the relationship (e.g., response time) between cloud nodes. Our approach systematically combines clustering analysis and ranking methods. The experimental results show that our approach outperforms other approaches.

In a communication-intensive application, nodes connect to each other with a special topology structure, e.g., a node has more connections to certain nodes, and fewer to other nodes. Currently, we have not considered the communication topology structure of communication-intensive applications. More investigations towards this direction will be our future work.

Acknowledgements

This research is supported by the National Basic Research Program (973) of China under the Grant No. 2011CB302603, and the National Science Foundation

of China under the Grant No. 60725206 and No. 61100078, and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 415311).

References

Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M. and Werthimer, D. (2002) 'Seti@home: an experiment in public-resorce computing', *Communications of the ACM*, Vol. 45 No. 11, pp.56–61.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M. (2010) 'A view of cloud computing', *Communications of the ACM*, Vol. 53, No. 4, pp.50–58.

Anderson, D.P. (2004) 'Boinc: a system for public-resource computing and storage', *GRID 2004: Proceedings of 5th International Workshop on Grid Computing*, Pittsburgh, USA, pp.4–10.

Bach, B.F. (2003) *Learning Spectral Clustering*, Computer Science Division, University of California Berkeley, UCB/CSD-03-1249.

Budati, K., Sonnek, J.D., Chandra, A. and Weissman, J.B. (2007) 'Ridge: combining reliability and performance in open grid platforms', *HPDC 2007: Proceedings of 3rd International symposium on High Performance Computing and Communications*, Monterey, USA, pp.55–64.

Buyya, R. (2010) 'Cloud computing the next revolution in information technology', *ICOMP 2010: Proceedings of 11th International Conference on Internet Computing*, Las Vegas, USA, pp.6–10.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M. and Bowman, M. (2003) 'Planetlab: an overlay testbed for broad-coverage services', *ACM SIGCOMM-Computer Communication Review*, Vol. 33, No. 3, pp.3–12.

Caromel, D., Costanzo, A.D. and Mathieu, C. (2007) 'Peer-to-peer for computational grids: mixing clusters and desktop machines', *Parallel Computing*, Vol. 33, Nos. 4–5, pp.275–288.

Cortellessa, V. and Grassi, V. (2007) 'A modeling approach to analyze the impact of error propagation on reliability of component-based systems', *CBSE 2007: Proceedings of 11th International Symposium on Internet Component-Based Software Engineering*, Massachusetts, USA, pp.140–156.

Costantini, A., Gutierrez, E., Cacheiro, J.L., Rodriguez, A., Gervasi, O. and Lagana, A. (2010) 'On the extension of the grid-empowered molecular science simulator: MD and visualisation tools', *International Journal of Web and Grid Services*, Vol. 6, No. 2, pp.141–159.

Cope, V., Trebon, N., Tufo, H.M. and Bechman, P.H. (2009) 'Robust data placement in urgent comoputing envirments', *IPDPS 2009: Proceeding of 23rd International Symposium on Parallel and Distributed Processing*, Rome, Italy, pp.1–13.

Chandra, A. and Weissman, J. (2009) 'Nebulas: using distributed voluntary resource to build clouds', *HotCloud 2009: Proceeding of 1st International Workshop on Hot Topics in Cloud Computing*, San Diego, USA, pp.2.

Deelman, E., Singh, G., Livny, M., Berrinman, G.B. and Good, J. (2008) 'The cost of doing science on the cloud: the montage exeample', *SC 2008: Proceeding of 20th International Conference on High Performance Computing*, Austin, USA, pp.50.

Ester, M., Kriegel, H., Sander, J. and Xu, X. (1996) 'A density-based algorithm for discovering clusters in large spatial databases with noise', *KDD 1996: Proceeding of 2th International Conference on Knowledge Discovery and Data Mining*, Oregon, USA, pp.226–231.

Fery, J., Tannenbaum, T., Livny, M., Foster, I.T. and Tuecke, S. (2002) 'Condor-g: a computation management agent for multi-institutional grids', *Cluster Computing*, Vol. 5, No. 3, pp.237–246.

Foster, I.R.I.T., Zhao, Y. and Lu, S. (2008) 'Cloud computing and grid computing 360-degree compared', *GCE 2008: Proceeding of 4th International Workshop on Grid Computing Environments*, Austin, USA, pp.1–10.

Fan, P., Wang, J., Zheng, Z. and Lyu, M.R. (2011) 'Toward optimal deployment of communication-intensive cloud applications', *Cloud 2011: Proceeding of 4th International Conference on Cloud Computing*, Washington, D.C., USA, pp.460–467.

Hayes, B. (2008) 'Cloud computing', *Communications of the ACM*, Vol. 51, No. 7, pp.9–11.

Hardavellas, N., Ferdman, M., Falsafi, B. and Ailamaki, A. (2009) 'Reactive nuca: near-optimal block palcement and replication in distributed caches', *ISCA 2009: Proceeding of 36th International Symposium on Computer Architecture*, Austin, USA, pp.184–195.

Hagen, L.W. and Kahng, A.B. (1992) 'New spectral methods for ratio cut partitioning and clustering', *IEEE Transactions on CAD of Integrated Circuits and Systems*, Vol. 11, No. 9, pp.1074–1085.

Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B. and Good, J. (2008) 'On the use of cloud computing for scientific workflows', *eScience 2008: Proceeding of 4th International Conference on eScience*, Indiana, USA, pp.640–645.

Jimenez, J.B, Caromel, D., Leyton, M. and Piquer, J.M. (2008) 'Load information sharing policies in communication-intensive parallel applications', in Priol, Thierry; Vanneschi, Marco (Eds.): *From Grids to Service and Pervasive Computing*, Springer, New York, USA, pp.111–121.

Jain, A.K., Murty, A.K. and Flynn, A.K. (1999) 'Data clustering: a review', *ACM Computing Surveys*, Vol. 31, No. 3, pp.264–323.

Kim, W. (2011) 'Cloud computing adoption', *International Journal of Web and Grid Services*, Vol. 7, No. 3, pp.225–245.

Keahey, K. and Freeman, T. (2008) 'Science clouds: early experiences in cloud computing for scientific applications', *CCA 2008: Proceeding of Cloud Computing and Applications*, Chicago, USA, pp.1–6.

Kaufman, L. and Rousseeuw, P.J. (2005) *Finding Groups in Data: An Introduction to Cluster Analysis; Electronic Version*, Wiley Series in Probability and Statistics, Wiley, Hoboken, NJ.

Koehler, M., Ruckenbauer, M., Janciak, I., Benkner, S., Lischka, H. and Gansterer, W.N. (2010) 'A grid services cloud for molecular modelling workflows', *International Journal of Web and Grid Services*, Vol. 6, No. 2, pp.176–195.

Kwok, T., Smith, K.A., Lozano, S. and Taniar, D. (2002) 'Parallel fuzzy c-means clustering for large data set', *Euro-Par 2002: Proceeding of the 8th International Euro-Par Conference*, Paderborn, Germany, pp.365–374.

Lohr, S. (2007) *Google and I.B.M Join in Cloud Computing Research*, October. Available at: http://www.nytimes.com/2007/10/ 08/technology/08cloud.html.

Luxburg, U. (2007) 'A tutorial on spectral clustering', *Statistics and Computing*, Vol. 17, No. 4, pp.395–416.

Liu, N.N. and Yang, Q. (2008) 'Eigenrank: a ranking-oriented approach to collaborative filtering', *SIGIR 2008: Proceeding of 10th International Conference on Research and Development in Informantion Retrieval*, Singapore, pp.83–90.

Miguel-Alonso, J., Mercero, T. and Ogando, E. (2007) *Performance of an Infiniband Cluster Running MPI Applications*, Technical report, EHU-KAT-IK-03-07.

Mohar, B and Juvan, N.T.M. (1997) 'Some applications of laplace eigenvalues of graphs', *Graph Symmetry: Algebraic Methods and Applications*, Vol 497, NATO ASI Series C, pp.227–275.

Malecot, P., Kondo, D. and Fedak, G. (2006) 'Xtremlab: a system for characterizing internet desktop grids', *HPCC 2006: Proceeding of 2th International Conference on High Performance Computing and Communications*, Munich, Germany, pp.357–358.

Mohar, B. (1992) 'Laplace eigenvalues of graphs – a survey', *Discrete Mathematics*, Vol. 109, Nos. 1–3, pp.171–183.

Nishtala, R., Hargrove, P., Bonachea, D. and Yelick, K.A. (2009) 'Scaling communicationintensive applications on bluegene/p using one-sided communication and overlap', *IPDPS 2009: Proceeding of 23rd International Symposium on Parallel and Distributed Processing*, Rome, Italy, pp.1–12.

Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T. and Epema, D.H.J. (2009) 'A performance analysis of ec2 cloud computing services for scientifc computing', *CloudComp 2009: Proceeding of 1st International Conference on Cloud Computing*, Munich, Germany, pp.115–131.

Petruch, K., Stantchev, V. and Tamm, G. (2011) 'A survey on IT-governance aspects of cloud computing', *International Journal of Web and Grid Services*, Vol. 7, No. 3, pp.268–303.

Qin, X., Jiang, H., Manzanares, A., Ruan, X. and Yin, S. (2010) 'Communication-aware load balancing for parallel applications on clusters', *IEEE Transactions on Computers*, Vol. 59, No. 1, pp.42–52.

Rampino, S., Pirani, F., Garcia, E. and Lagana, A. (2010) 'A study of the impact of long range interactions on the reactivity of $N + N2$ using the Grid Empowered Molecular Simulator GEMS', *International Journal Web and Grid Services*, Vol. 6, No. 6, pp.196–212.

Rodriguez, J.M., Zunino, A. and Campo, M. (2011) 'Introducing mobile devices into Grid systems: a survey', *International Journal Web and Grid Services*, Vol. 7, No. 1, pp.1–40.

Sonnek, J.D., Chandra, A. and Weissman, J.B. (2007) 'Adaptive reputation-based scheduling on unreliable distributed infrastructures', *IEEE Transactions on Parallel Distribute System*, Vol. 18, No. 11, pp.1551–1564.

Sims, K. (2007) *IBM Introduces Ready-To-Use Cloud Computing Collaboration Services Get Clients Started With Cloud Computing*, Available at:http://www-03.ibm.com/press/us/en/pressrelease/22613.wss.

Shi, J. and Malik, J. (2000) 'Normalized cuts and image segmentation', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp.888–905.

Spielman, D.A. (2007) 'Spectral graph theory and its applications', *FOCS 2007: Proceeding of 48th International Symposium on Foundations of Computer Science*, RI. USA, pp.29–38.

Sun, Q., Wang, S., Zou, H. and Yang, F. (2011) 'QSSA: a QoS-aware service selection approach', *International Journal of Web and Grid Services*, Vol. 7, No. 2, pp.147–169.

Taniar, D. and Leung, C.H.C. (2003) 'The impact of load balancing to object-oriented query execution scheduling in parallel machine environment', *Information Sciences*, Vol. 157, pp.33–71.

Taniar, D., Leung, C.H.C., Rahayu, W. and Goel, S. (2008) *High Performance Parallel Database Processing and Grid Databases*, John Wiley and Sons, Hoboken, NJ.

Vecchiola, C., Pandey, S. and Buyya, R. (2009) 'High-performance cloud computing: a view of scientific applications', *ISPAN 2009: Proceeding of 10th International Symposium on Pervasive Sytems, Algorithms, and Network*, Kaohsiung, Taiwan, pp.4–16.

Wang, L., Tao, J., Marcel, K., Canales, C.A., David, K. and Wolfgang, K. (2008) 'Scientific cloud computing: early definition and experience', *HPCC 2008: Proceeding of 10th International Conference on High Performance Computing and Communications*, Dalian, China, pp.825–830.

Yuan, D., Yang, Y., Liu, X. and Chen, J. (2010) 'A data placement strategy in scientific cloud workflows', *Future Generation Computer System*, Vol. 26, No. 8, pp.1200–1214.

Zhang, Y., Huang, G., Liu, X. and Mei, H. (2010) 'Integrating resource consumption and allocation for infrastructure resources on-demand', *CLOUD 2010 Proceeding of 3th International Conference on Cloud Computing*, Miami, USA, pp.75–82.

Zheng, Z., Zhang, Y. and Lyu, M.R. (2010a) 'Cloudrank: a qos-driven component ranking framework for cloud computing', *SRDS 2010: Proceeding of 29th International Symposium on Reliable Distributed Systems*, Delhi, India, pp.184–193.

Zheng, Z., Zhou, T.C., Lyu, M.R. and King, I. (2010b) 'Component ranking for fault-tolerant cloud applications', *IEEE Transactions on Service Computing*, Accept, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5959151