A topology-aware method for scientific application deployment on cloud

Pei Fan*

China HuaYi Broadcasting Corporation,
Fuzhou 350001, China
Email: peifan@nudt.edu.cn
*Corresponding author

Zhenbang Chen and Ji Wang

National Key Laboratory for Parallel and Distributed Processing,
School of Computer Science,
National University of Defense Technology,
Changsha 410073, China
Email: zbchen@nudt.edu.cn
Email: wj@nudt.edu.cn

Zibin Zheng and Michael R. Lyu

Shenzhen Research Institute,
The Chinese University of Hong Kong,
Nanshan District, Shenzhen, China
and
Department of Computer Science & Engineering,
The Chinese University of Hong Kong,
Hong Kong, China
Email: zbzheng@cse.cuhk.edu.hk
Email: lyu@cse.cuhk.edu.hk

Abstract: Nowadays, more and more scientific applications are moving to cloud computing. The optimal deployment of scientific applications is critical for providing good services to users. Scientific applications are usually topology-aware applications. Therefore, considering the topology of a scientific application during the development will benefit the performance of the application. However, it is challenging to automatically discover and make use of the communication pattern of a scientific application while deploying the application on cloud. To attack this challenge, in this paper, we propose a framework to discover the communication topology of a scientific application by pre-execution and multi-scale graph clustering, based on which the deployment can be optimised. In addition, we present a set of efficient collective operations for cloud based on the common interconnect topology. Comprehensive experiments are conducted by employing a well-known MPI benchmark and comparing the performance of our method with those of other methods. The experimental results show the effectiveness of our topology-aware deployment method.

Keywords: topology-aware; communication topology; scientific applications; deployment; cloud computing.

Reference to this paper should be made as follows: Fan, P., Chen, Z., Wang, J., Zheng, Z. and Lyu, M.R. (2014) 'A topology-aware method for scientific application deployment on cloud', *Int. J. Web and Grid Services*, Vol. 10, No. 4, pp.338–370.

Biographical notes: Pei Fan received his PhD degree from the School of Computer Science at National University of Defense Technology in China. His main research interests focus on cloud computing and service-oriented computing.

Zhenbang Chen is an Assistant Professor 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.

Ji Wang is currently a Professor in the Department of Computer Science, National University of Defense Technology in China. He is also the Director of the Department of Computer Science. His research interests include high confidence software development, software engineering and distributed computing. He served in Programme 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.

Zibin Zheng is an Associate Research Fellow at Shenzhen Research Institute, 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 ICWS'2010, First Runner-up Award at 2010 IEEE Hong Kong Postgraduate Research Paper Competition and IBM PhD Fellowship Award 2010–2011. He served as a Programme Committee Member for many conferences including CLOUD and SCC. 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 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.

This paper is a revised and expanded version of a paper entitled 'Topology-aware deployment of scientific applications in cloud computing' presented at the IEEE Conference on Cloud Computing, Hawaii, USA, 24–29 June 2012.

1 Introduction

Scientific computing usually needs huge computing resources to carry out large-scale scientific experiments. In addition, the data transportation in scientific experiments requires a high bandwidth. Nowadays, a lot of scientific applications are deployed in grid systems because they have a high performance and massive storage. Traditional grids are based on batch-queue mechanisms that execute programs according to a pre-established scheduling policy, and they offer scalability by increasing the number of utilised computing nodes (Evoy et al., 2011). However, building a grid system is extremely expensive and it is not available for scientists all over the world to use. Recently, cloud computing has been under a growing spotlight as a possible solution for providing a flexible, on-demand computing infrastructure for scientific applications (Hoffa et al., 2008; Gunarathne et al., 2011). 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, a much more reliable platform than grids, and much more scalable platform than the largest common clusters or resource pools (Buyya et al., 2000; Foster et al., 2008). However, the nature of distributing and latency/bandwidth diversity of cloud nodes makes deploying and executing the scientific applications over clouds a challenging problem.

There are three kinds of methods for deploying applications on cloud: random based, ranking based and clustering based. A random method selects cloud nodes randomly. A ranking method will rank available cloud nodes based on their Quality of Service (QoS) values and select the best ones. Ranking methods are usually used for computation-intensive applications, but are not appropriate for communication-intensive applications [e.g. Message Passing Interface (MPI) programs] (Fan et al., 2011). The reason is a ranking method cannot consider the communication performance between cloud nodes. For deploying communication-intensive applications, clustering-based methods (Fan et al., 2011; Fan et al., 2012a) are proposed. The basic idea of a clustering-based method is to cluster the cloud nodes that have a good communication performance together to deploy an application.

Scientific applications can usually be decomposed into interdependent components, connected according to a specific topology and capable of exploiting different types of computational resources: this is what we call topology-aware applications (Bar et al., 2009). However, current deployment methods rarely consider the communication topology information of deployed applications. Thus, in the general case, for a clustering-based method, an application may continuously communicate back and forth between clusters, with a significant impact on performance. Therefore, we need to consider topology information when deploying scientific applications. A few approaches try to use the topology information to improve the performance of systems (e.g. Bar et al., 2009). These approaches usually need users to describe a topology for a deployed application. However, this requirement is not practical in cloud computing, since the scientific application may not be developed by the user, and even the sources may be not available. In this paper, we propose a topology-aware framework to automatically discover topology information and use the topology information during deployment.

Collective operations are often the critical factor determining the ultimate performance of parallel scientific applications. Therefore, how to optimise the collective operations is a challenge for researchers. Existing approaches usually optimise collective operations via implementation of collective operations algorithm (Almási et al., 2005) or

MPI library (Hoefler et al., 2011). However, implementation of collective operations algorithm or MPI library is difficult to a cloud user, since the cloud user may not be the developer of a scientific application. In order to optimise collective operations in cloud environment, we analyse the common interconnect topology of collective operations and deploy scientific applications based on these common interconnect topology.

This paper is an extension of our conference paper (Fan et al., 2012b). Compared with the work in Fan et al. (2012b), the following extensions can be found:

- We analyse the common interconnect topologies of collective operations and propose a topology-aware method to optimise collective operations. Instead of improving the collective operations algorithms directly, our topology-aware method optimises collective operations based on their common interconnect topologies.
- For determining the appropriate value of logical topology structures, we analyse the relation between a logical topology and its physical topology structures, and present appropriate logical topology structures via experiments.
- More extensive real-world experiments have been conducted.

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

2 Related work

Recently, scientific applications in clouds have attracted great interests, since clouds provide an alternative to clusters, grids and supercomputers for scientists at a lower cost (Petruch et al., 2011). For deploying applications or services on distributed systems or clouds, a number of approaches have been proposed. BOINC (Anderson, 2004) is a volunteer computing framework, which uses a random method to select the nodes for deployment. Although it is easy to choose nodes randomly, the performance is often poor. 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 the 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; Zheng et al., 2010; Zheng et al., 2011). CloudRank (Zheng et al., 2010) is a QoS-driven component ranking framework for cloud computing. Zheng et al. (2012) propose a component ranking method for fault-tolerant cloud applications. Kang et al. (2011) propose a user-experience-based mechanism to redeploy cloud services. These approaches are ranking-based methods and usually used for computing-intensive applications. Zheng and Lyu (2013) propose two personalised reliability prediction approaches of web services and recommend the best web services to the user based on the prediction result. Scientific applications usually have a lot of communications between the involved nodes. However, ranking 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. The sizes of data have grown exponentially in the past, and data have been distributed widely in a grid or cloud

environment (Taniar et al., 2008). In order to improve the performance of data-intensive applications, Yuan et al. (2010) propose a matrix-based *k*-means clustering strategy for data placement in scientific cloud workflows. SSS (Nakada et al., 2012) is a MapReduce-based stream processing system, which is capable of processing the streams of large-scale static data. Sailfish (Rao et al., 2012) is a new MapReduce framework for large-scale data processing. Different from the preceding existing work, our work focuses on the optimal deployment of the scientific applications on cloud platforms and the communication performance. However, for data-intensive applications, we believe our method is also feasible.

There are also existing literatures for improving the communication performance of the applications in cloud. Hoffa et al. (2008) indicate that the communication performance is important for the scientific applications in cloud computing. Ostermann et al. (2009) analyse the performance of the EC2 cloud computing services for scientific computing. Koehler et al. (2010) present a service-oriented infrastructure that integrates grid computing technologies with a cloud infrastructure to support the scheduling of dynamic scientific workflows. P4P (Xie et al., 2008) uses application layer traffic optimisation to control the traffic between applications and network providers. Bessai et al. (2012) propose three bi-criteria complementary approaches to tackle the allocation and scheduling workflow problems in cloud environments. Chronos (Kapoor et al., 2012) is an architecture to reduce data centre application latency especially at the tail. Based on the cross-service information and user locations, Kang et al. (2012) propose a latency-aware co-deployment mechanism for cloud-based services. We propose (Fan et al., 2011; Fan et al., 2012a) a framework that considers the node relations and uses clustering analysis to deploy communication-intensive applications. In an Infrastructure-as-a-Service (IaaS) cloud, resources or services are provided to users in the form of Virtual Machines (VMs). Therefore, several schedule algorithms based on virtual machines are proposed. Chen et al. (2011) propose a static method for tasks placement and scheduling based on virtual machines. Li et al. (2012) present a resource optimisation mechanism in heterogeneous IaaS federated multi-cloud systems, which enables pre-emptable task scheduling. Compared with our work in this paper, the existing work does not consider the communication topologies of the scientific applications in cloud computing, and a poor performance or overload may occur in some scenarios when using these methods.

There are also some existing literatures for running or deploying applications with respect to topology information. In Lacour et al. (2005), a generic application description model is proposed for the automatic deployment of the applications on computational grids. Bar et al. (2009) design a topology-aware grid middleware to schedule the topology-aware applications in grids. Coti et al. (2009) propose a topology-aware approach to deploying MPI applications in grid. In Bhatele et al. (2009), an API for topology-aware task mapping is introduced. Chang et al. (2012) present a placement algorithm that exploits the Euclidean triangular inequality property of network topologies. All of these approaches need users or developers to describe the communication patterns of scientific applications, and then map or schedule tasks on nodes. However, it is not practical for cloud users to provide communication patterns. Compared with the existing methods, we use pre-execution and clustering analysis to get topology information automatically.

For many parallel programs, such as MPI program, collective operations are critical. The speed of MPI collectives is, needless to say, often the critical factor determining the ultimate performance of the applications. To improve the performance, a number of

collective algorithms have been proposed. In Kumar et al. (2008) and Kandalla et al. (2010), some topology-aware collective communication algorithms are presented for large-scale clusters. Träff (2002) uses a topology mechanism to implement MPI. Hoefler et al. (2011) propose a new scalable process topology interface for MPI 2.2. Faraj and Yuan (2005) present a system that produces efficient MPI collective communication routines. Almási et al. (2005) implement a number of optimised collective operations on BlueGene/L systems. These approaches focus on how to implement the underlying library or collective operations. Different from previous works, our work focuses on how to provide an optimal deployment for collective operations. We have analysed the common interconnect topologies of collective operations and use the topology information to optimise collective operations.

3 Motivation and architecture

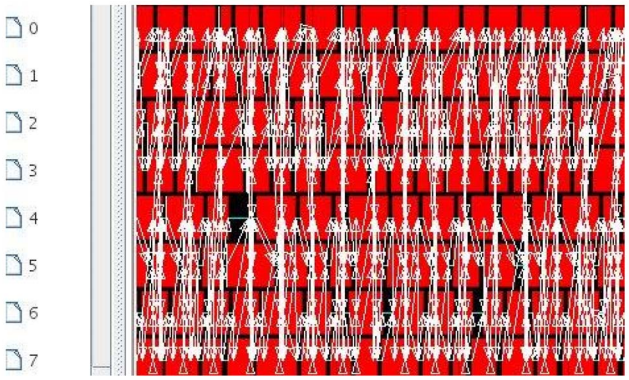
3.1 Motivation

Here we describe a topology-aware application that we will use for testing our topology-aware deployment method.

A scientific application usually needs collaborations of computing nodes, and there are a lot of communications between these nodes. An example of a communication information graph of an MPI application is shown in Figure 1. The numbers on the left are the node numbers (this application was deployed on eight nodes) and on the right is the communication graph (*x*-axis represents time). White arrows represent the messages exchanged between nodes. The application shown in Figure 1 has been parallelised in a manner that combines eight nodes to conduct this MPI application, and Figure 1 shows the following:

- The communications in the first four nodes are frequent, and the same situation happens in the last four nodes. However, the communications between these two groups are obviously less.
- Based on the communication information, we can partition the first four nodes into the same communication topology structure and the rest of the nodes into another structure.

Figure 1 The communication graph of an MPI application (see online version for colours)



In order to deploy a scientific application on cloud, we can use clustering methods to select nodes, since a clustering method can reflect the relations between nodes and partition the similar nodes (low latency between nodes) into the same cluster. However, the result of a clustering method would be very poor in the following scenarios.

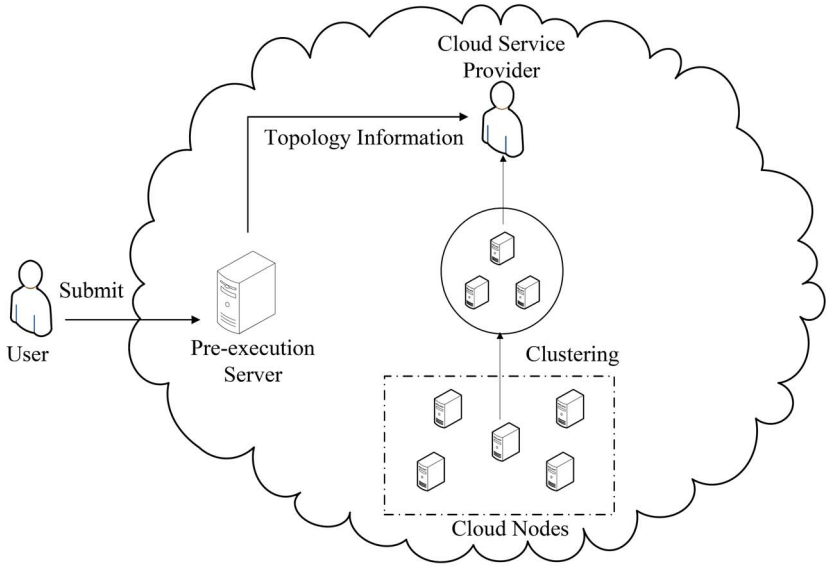
- *Choose nodes from multi-clusters:* the nodes in a same topology structure may be from different clusters if the cloud service provider selects the nodes across multi-clusters. For example, in Figure 1 nodes (0–3) should be selected from one cluster. However, in an across clusters scenario, nodes 1 and 2 may be selected from one cluster, but the rest of the nodes from another cluster, which may lead to poor performance.
- *Overload:* Taniar and Leung (2003) indicate that overload has been one of the major problems in a distributed and parallel environment. All selected nodes may be in the same cluster when using a ranking or clustering method. Under this situation, if users deploy several applications on these nodes, overload will happen.

In order to address the aforementioned problems, we propose a topology-aware framework to deploy scientific applications on cloud based on communication topology. Our method can take into account not only the communication performance between nodes but also the communication topology of a scientific application. The details of this framework will be introduced in the following section.

3.2 Topology-aware node selection framework

Figure 2 shows the architecture of our proposed topology-aware method for deploying scientific applications on cloud. The workflow of our framework is as follows:

Figure 2 Topology-aware deployment framework



- A cloud user submits an application to the cloud environment. This application will be sent to the pre-execution server. The pre-execution server takes charge of discovering and analysing the communication topology of the application. To ensure effectiveness of pre-execution phase, we employ a method that reduces the problem size of the application when pre-executing the application (Ananth et al., 2003). After pre-execution, the communication information will be recorded, based on which the topology can be extracted. In our experiments (Section 5) using MPI programs, we develop a slog-2 logfile (Chan et al., 2008) analysis tool that can discover the communication topology of an MPI program based on the MPI slog2sdk (SLOG-2 software development kit) (Chan et al., 2008), which can record the message exchanges of an MPI program when it is running. More details will be introduced in Section 4.1.
- Each cloud node runs a monitor program, which takes charge of monitoring the computing and communication performances of the cloud node. To precisely measure the computing and communication performances of the cloud node, we use the average value during a period as the value of each performance. According to the communication performance, cloud nodes will be partitioned into different clusters via clustering analysis.
- Based on the communication topology of a scientific application, the cloud service provider can map the topology of the cloud node clusters with the topology of the application, and select nodes from appropriate clusters. More details of node partition and selection will be introduced in Section 4.2.

4 Deploy method

This section presents our topology-aware method for deploying scientific applications in cloud, which is explained in two steps. First, we will introduce how a multi-scale clustering algorithm can be used to discover the communication topology of a scientific application. Next, we will use a spectral clustering method to partition cloud nodes into different clusters and present how cloud nodes from the generated clusters can be selected with respect to the topology information.

4.1 Logical topology discovery

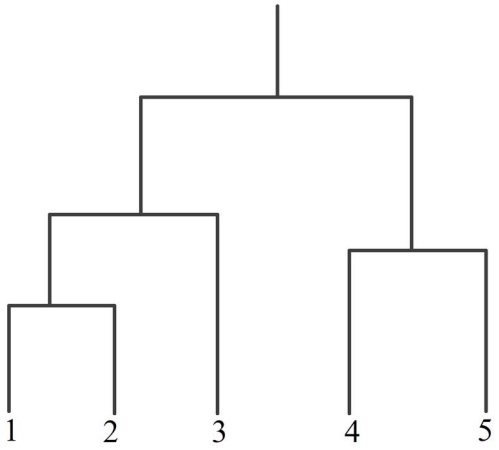
The communication pattern of a scientific application can be modelled by an undirected (weighted) graph, and it is assumed that two adjacent nodes in the graph have some communications. An undirected graph is usually represented as an adjacency matrix, each entry of which represents the communication frequency between the node pair. For example, the following is an adjacency matrix of a scientific application that is deployed on four nodes:

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array}\begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 3 & 1 & 0 \\ 3 & 0 & 0 & 1 \\ 1 & 0 & 0 & 3 \\ 0 & 1 & 3 & 0 \end{pmatrix} \end{array} \tag{1}$$

From equation (1), we can observe that there have been a lot of communications in the node pair (*A*, *B*) or (*C*, *D*), and less in (*A*, *C*) and (*B*, *D*). In this paper, we formulate the communication topology discovery problem as a graph clustering problem: we want to find the structure of adjacency, in which nodes are joined together in a tightly knit structure (which means that the nodes within the same structure have more communications with each other). And there are only looser connections between structures, which mean the nodes in different structures have less communications.

Usually, a graph clustering algorithm partitions a set of nodes into *k* groups, where *k* is an input to the algorithm. Therefore, we should know the value of *k* before using a graph clustering algorithm to discover topology. However, this assumption is not practical for cloud computing, since a cloud user or provider may not be the developer of the applications to be deployed. To attack this challenge, we use a hierarchical clustering algorithm (Jain et al., 1999). A hierarchical clustering algorithm does not assume any particular number of clusters. Instead, a desired number of clusters can be obtained by 'cutting' the dendrogram at a proper level (Jiang et al., 2004). An example of dendrogram is shown in Figure 3. The results of a hierarchical clustering algorithm are often improved with refinement algorithms, which iteratively reassign nodes to different clusters (Karypis et al., 1999). In this subsection, we use a multi-scale refinement algorithm (Noack and Rotta, 2009; Hadany and Harel, 2001) to discover a communication topology. The details of graph clustering and clustering criteria will be introduced in the following.

Figure 3 Dendrogram graph



An undirected graph *G* is defined as (*V*, *E*), where *V* is the node set and *E* is the edge set. The weights of edges are defined by a total function $f: V \times V \rightarrow N$. For an undirected graph, $f(u, v) = f(v, u)$, where $u, v \in V$. The degree of a node *v*, denoted by deg(*v*), is defined as the total weight of its edges, i.e. $\sum_{u \in V} f(v,u)$. The degree of a nodes set deg(*C*) is defined as $\sum_{u \in C} \deg(u)$; and the weight of two node sets, $f(V_1, V_2)$ is defined as $\sum_{u \in V_1, V \in V_2} f(u,v)$. A merging operation assigns to each cluster pair (*C*, *D*) a real number

called merging priority, and thereby determines the order in which an algorithm merges cluster pairs. In this paper, we use *weight density* as a merge prioritisation to merge cluster pairs. The weight density of a cluster pair is defined as

$$\frac{f(C,D)}{\deg(C)\deg(D)} \tag{2}$$

Informally, we denote a subgraph as a graph cluster if it has many internal edges and few edges to the remaining. This can be formalised by defining a measure for the coupling between subgraphs, such that a smaller coupling indicates a better clustering.

Modularity is a quality measure for graph clustering. Newman (2004) proposes a modularity measure of the coupling for k disjoint sets of nodes, which is defined in equation (3):

$$Q(V_1,\ldots,V_k)=\sum_{i,j=1}^{k}\left(\frac{cut(V_i,V_j)}{|E|}-\frac{\deg(V_i)^2}{\deg(V)^2}\right) \tag{3}$$

In equation (3), $|E|$ is the number of edges, $cut(V_i,V_j)$ is the sum of the weights of the cut wedges, the first term is the fraction of all edges that are within V_i and the second term is the expected value of this quantity (Noack, 2007). It can easily be verified that merging two clusters C and D increases the modularity by the following equation:

$$\Delta Q_{C,D}:=\frac{2f(C,D)}{f(V,V)}-\frac{2\deg(C)\deg(D)}{\deg(V)^2} \tag{4}$$

In addition, moving a node v from its current cluster C to another cluster D increases the modularity, which is explained by equation (5):

$$\Delta Q_{v\to D}:=\frac{2f(v,D)-2f(v,C-v)}{f(V,V)}-\frac{2\deg(v)\deg(D)-2\deg(v)\deg(C-c)}{\deg(v)^2} \tag{5}$$

Our multi-scale algorithm for discovering a logical topology has two stages: first, use a hierarchical algorithm to partition nodes into clusters; second, use a refinement algorithm to refine the clusters. The algorithm is shown in Algorithm 1.

Algorithm 1: Multi-scale graph clustering algorithm

Input: Adjacency matrix *M*, Edge set *E*, Node set *N*

Output: Topology structure that includes *k* groups

```
1  bMerge=true;
2  while bMerge do
3      bMerge = false;
4      Use Equation 2 to calculate weight densities;
5      E = Sort(E) //based on weight density;
6      foreach e ∈ E do
7          if e.weight density < atedges/atparis then
```

```
8                Break
9            end
10           if e.startnode or e.endnote merged then
11               Continue
12           end
13           n = Merge e.startnode and e.endnote;
14           bMerge=true;
15           N = N – {e.startnode} – {e,endnote};
16           N = N + {n};
17       end
18       Calculate new adjacency matrix M, Edge set E;
19   end
20   l=level of dendrogram;
21   for l from l_max – 1 to 1 do
22       repeat
23         (v, D) ← best node move;
24         if ΔQ_{v←D} > 0 then
25             Move node v to the cluster D
26         end
27       until ΔQ_{v→D} ≤ 0;
28   end
```

- Step 1 (4–5): Calculate weight density via equation (2), and then descend edge rankings based on weight densities.
- Step 2 (lines 6–18): If the start node and end node of an edge have been not merged, the algorithm will merge these two nodes when the weight density of the edge is greater than $atedges/atparis$, where $atedges$ is the sum of the edge weights of the graph and $atparis$ is the square of the sum of the node weights. Then, calculate the new adjacency matrix M and edge E, until no more clusters can be merged.
- Step 3 (lines 20–23): l is the level of dendrogram (cf. Figure 3) and equal to the loop count in Step 2. In line 23, the algorithm uses equation (5) to calculate the modularity of moving a node to another cluster and selects the best node move (v, D). Here the best node move is a move with the largest modularity increase (cf. equation 5).
- Step 4 (lines 24–27): Move v to D, if (v, D) is the best move and $\Delta Q_{v\to D} > 0$. The process will be repeated until $\Delta Q_{v\to D} \le 0$.

Step 2 of Algorithm 1 means two nodes merge when $\Delta Q_{C,D} > 0$ (equation 4); the proof can be seen in the following.

Proof: Two nodes merge when the weight density of the edge is greater than $atedges/atparis$ in step 2. Hence,

$$\frac{f(C,D)}{\deg(C)\deg(D)}>\frac{atedges}{atparis}$$

where $atedges = f(V, V)$ is the sum of the weights of all edges and $atparis = \deg(V)^2$ is the square of the degree of all the nodes. Hence,

$$\frac{f(C,D)}{\deg(C)\deg(D)}>\frac{f(V,V)}{\deg(V)^2}$$

since all the items are positive numbers. Hence,

$$\frac{f(C,D)}{f(V,V)}>\frac{\deg(C)\deg(D)}{\deg(V)^2}$$

Hence,

$$\frac{2f(C,D)}{f(V,V)}-\frac{2\deg(C)\deg(D)}{\deg(V)^2}>0$$

which means two nodes merge when $\Delta Q_{C,D}>0$.

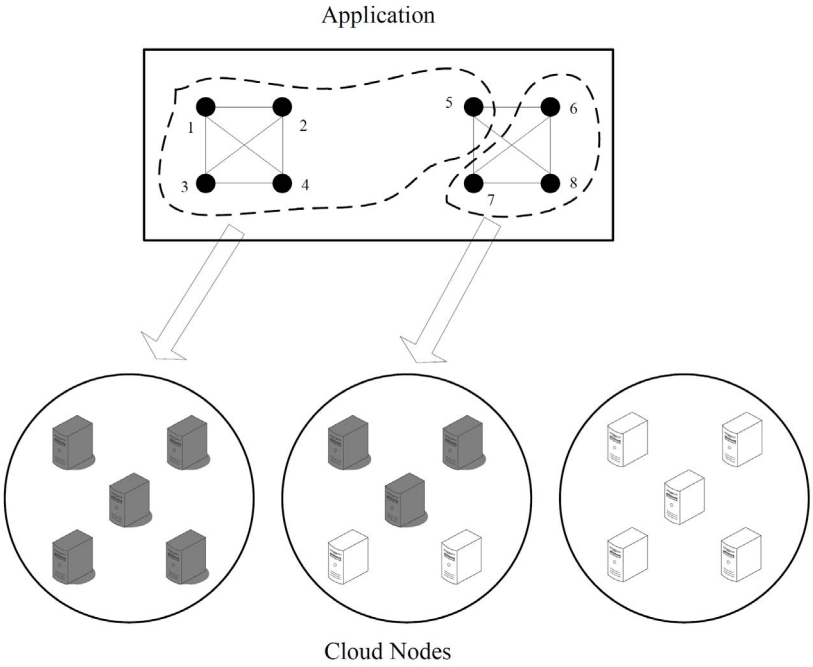
The multi-scale algorithm can generate the topology structure of a scientific application automatically. In the next section, we will explain how the physical topology of the nodes in cloud can be discovered, and then map a logical topology to a physical topology.

4.2 Physical topology discovery

Topology-aware deployment requires the information of two aspects: the logical topology (or communication topology) and the physical topology (or cloud node topology). In the previous subsection, we describe the method to obtain a logical topology. This subsection introduces the method of obtaining the physical topology of cloud nodes.

If we restrict the communications in applications only between neighbour nodes, we can have a better utilisation of the available bandwidth. Therefore, we want to have the physical topology of cloud nodes, and the nodes that are close to each other will be in the same topology structure. The nodes of different clusters will have a higher latency. Thus, if we denote the latency relations of the nodes in cloud as an adjacency matrix, the physical topology discovery problem can also be formulated as a graph clustering problem. In our previous works (Fan et al., 2011; Fan et al., 2012a), we propose a spectral clustering-based method to discover the topology of cloud nodes. In this paper, we use the discovery method in Fan et al. (2012a) to get the topology of cloud nodes.

Figure 4 Map tasks without considering topology

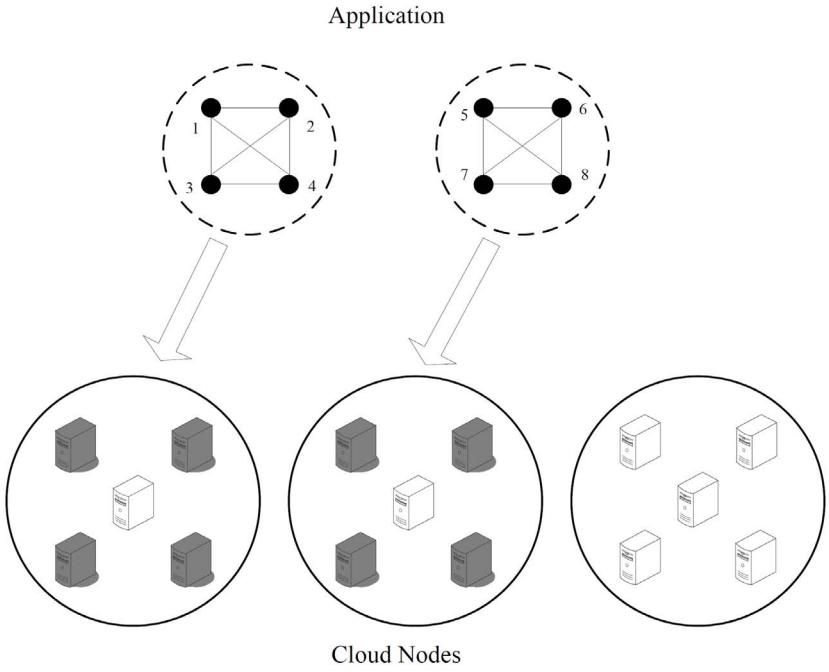


After getting a physical topology, cloud nodes are partitioned into different clusters. Then, we can map the task to these nodes for deployment. In the map phase, we may obtain poor performance if we do not consider the communication topology. We use a simple example in Figure 4 to demonstrate the problem. In Figure 4, the application has eight tasks and should be deployed on 15 cloud nodes. We can observe the communications between these tasks can be partitioned into two topology structures, each of which includes four tasks, and cloud nodes are partitioned into three clusters, each of which has five nodes. In this scenario, nodes should be selected from different nodes clusters, since the number of nodes in each cluster is smaller than the number of tasks. If we use the map method that does not consider the communication topology (e.g. the method introduced in Fan et al., 2012a), it will select all the nodes from the first cluster for deploying tasks 1–5, and three nodes from the second cluster for deploying the tasks 6–8. However, in this scenario, there have been a lot of communications between the first cluster and the second cluster, since task 5 deployed on the first cluster has a lot of communication with tasks 6–8. Therefore, mapping tasks to cloud nodes without considering topology information may lead to a poor performance.

In order to address the problem, we select the nodes for deployment based on the logic topology information of an application. As shown in Figure 5, we first rank these three clusters based on the performance of these clusters and select the first two clusters. Then, we select four nodes from the first cluster for deployment in tasks 1–4, and select four nodes from the second cluster for deployment in tasks 5–8.

We use a greedy algorithm (Fan et al., 2012a) to rank the generated clusters. After mapping and greedy ranking, the required cloud nodes of deploying an application can be selected.

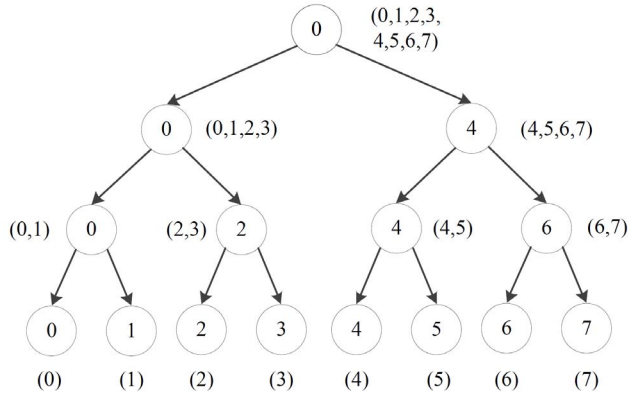
Figure 5 Select nodes based on topology structures



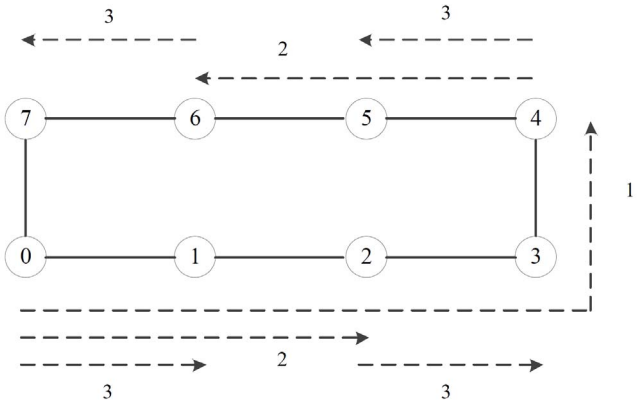
4.3 Collective operations

Collective communication is an important and frequently used component of parallel scientific applications. A study conducted at the Stuttgart High-Performance Computing Center shows that 45% of the overall time on their Cray T3E is spent on MPI routines (Coti et al., 2009; Rabenseifner, 2004). Collective operations have been studied and optimised during the last decades. A hierarchical broadcast algorithm is presented in Cappello et al. (2001). Furthermore, topology information can be used to organise point-to-point communications within collective operations. MPICH-G2 (Karonis et al., 2000) uses topology-discovery features to implement a topology-aware hierarchical algorithm. These approaches are focused on implementing and optimising MPI library and collective operations. However, cloud users may not be the developer of the applications, and a cloud platform usually has a special environment (e.g. OpenMPI) (Gabriel et al., 2004). Thus, it is difficult for users or providers to optimise collective operations. To attack this challenge, we analyse the common topologies of collective operations and use topology information and hierarchical models to optimise collective operations.

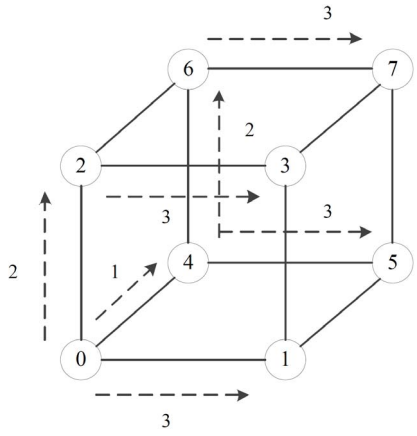
Figure 6 MPI_Scatter operations on different topologies. (a) Balance binary tree; (b) ring; (c) hypercube; (d) mesh



(a)

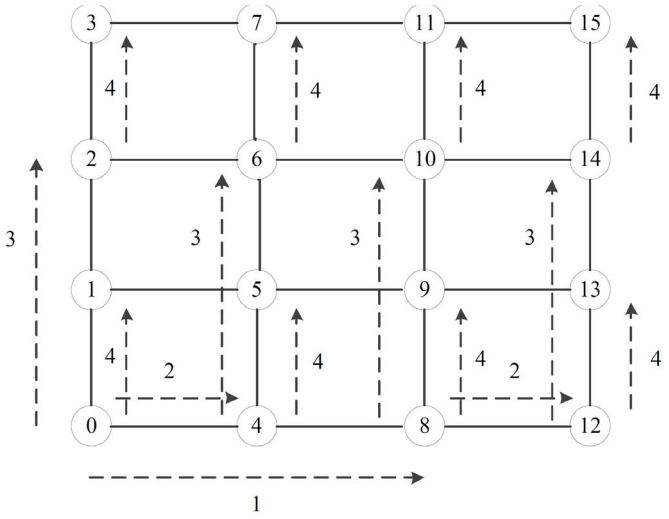


(b)



(c)

Figure 6 MPI_Scatter operations on different topologies. (a) Balance binary tree; (b) ring; (c) hypercube; (d) mesh (continued)



(d)

We introduce a variety of topologies usually used for collective operations before describing our optimisation method. Figure 6 shows four topologies (balance binary tree, ring, hypercube and mesh) used to implement Scatter operations. Figure 6(a) displays the detail communication steps for a Scatter operation on eight-node tree. In Figure 6(a), the source node (node 0) contains all the messages. The messages are identified by the labels of the destination nodes. In the first communication step, the source transfers half of the messages to node 4. In subsequent steps, each node that has some data transfers half of it to a related node (e.g. node 0 sends data to node 2, and node 4 to node 6) that has not yet received any data. There is a total of log$N$ (N is the number of nodes) communication steps. The communication steps of the Scatter operations on ring and hypercube topologies are similar to those of balance binary tree. Figure 6(d) shows a Scatter operation on a 16-node mesh, which also have log$N$ communication steps. From Figure 6, we can observe that the communication of MPI_Scatter can be organised as a hierarchical model, where messages are transmitted form the top level to the lowest level, and the nodes can be partitioned into different topological groups based on the communications. For example, in Figure 6(a), we can partition eight nodes into two topological groups (left subtree of root node and right subtree of root node), each of which has four nodes. Therefore, instead of implementing special collective operation algorithms, we can use the hierarchical model and the topology information for optimisation.

In order to use the topology information for collective operations, we have the following rules for mapping nodes to the structures and the corresponding communicators: (a) each topological structure is assigned with a set of nodes which are close to each other; (b) for all the groups, a master is defined (e.g. rank 0 is the master of

all groups) and a sub-master is defined within some groups (e.g. rank 4 is a sub-master of a group); (c) the nodes within some topology structures have special ranks, e.g. the nodes mapped in a given topology structure have consecutive in-Scatter operations, because the process with rank 0 sends messages to the processes with ranks 1 and 2, and the process with rank 2 sends message to the process with rank 3. Next, we use three collective operations as examples to show our optimisations.

MPI_Scatter: We start a Scatter operation at the process of rank 0 (root node) and conduct the operation in a hierarchical way: a root node (rank 0) is defined for topology structures and a sub-root is defined in each topology structure. Messages are sent from root node to sub-root nodes. Each sub-root node then sends the messages to its 'sub-structure' nodes, until the nodes at the lowest level are reached.

MPI_Bcast: Broadcast operations are quite similar to that of Scatter operations. The communication patterns of one-to-all Broadcast and Scatter are identical. The only differences are the size and the content of messages. Therefore, we can conduct a broadcast operation in the same way as that of Scatter operations.

MPI_Alltoall: Alltoall is a generalisation of one-to-all broadcast, and all the nodes involved simultaneously initiate a broadcast. We can use a hypercube to conduct Alltoall operations. In hypercube mode, communications take place along a different dimension of the eight-node hypercube in each step. In every step, pairs of nodes exchange their data and double the size of the messages to be transmitted in the next step by concatenating the received messages with their current data. For example, in Figure 6(c), data are simultaneously exchanged in these four node pairs {(0,1),(2,3),(4,5),(6,7)}, and then communications take place in {(0,2),(1,3),(4,6),(5,7)}. At the last step, each node pair in {(0,4),(1,5),(2,6),(3,7)} exchanges data.

4.4 Grain of logical topology structures

In Sections 4.1 and 4.2, we obtain the logical and physical topology structures. We define the number of logical topology structures as $k_{\log}$ and the number of physical structures as k_{phy}. Instead of inputting $k_{\log}$ before running, multi-scale algorithm can obtain an adequate $k_{\log}$ automatically. However, the value of $k_{\log}$ may not be proper in some scenarios when mapping a logical topology to a physical topology, since the value of k_{phy} will greatly influence the result of mapping. For example, after obtaining logical and physical topologies, $k_{\log} = 4$ and $k_{phy} = 4$. Thus, the four logical topology structures are mapped to the four physical topology structures, which means we select the nodes from four physical topology structures for deploying scientific applications. However, the third or fourth cluster has a poor performance (descending cluster ranking). Thus, the nodes that have poor performance may be selected. To address this problem, we analyse the relation between $k_{\log}$ and k_{phy}, and there are two scenarios of mapping a logical topology to a physical topology:

- $k_{\log} > k_{phy}$: intuitively, the value of $k_{\log}$ should be reduced to adapt to the value of k_{phy}.
- $k_{\log} \leq k_{phy}$: this scenario is more complex than $k_{\log} > k_{phy}$. In this scenario, we need to determine whether to reduce the value of $k_{\log}$ to adapt to the physical topology structures to avoid the problem mentioned earlier.

In this paper, we focus on the second scenario, i.e. $k_{\log} \leq k_{phy}$. In this scenario, we determine whether to reduce the value of $k_{\log}$ with respect to the value of $k_{\log}/k_{phy}$: if $k_{\log}/k_{phy}$ is indeed less than a special value, the topology-aware method will be better than other methods. For determining an appropriate value of $k_{\log}/k_{phy}$, we conduct a series of experiments that compare our topology-aware method with other methods in different values of $k_{\log}/k_{phy}$. Based on these experiments, we obtain the appropriate value of $k_{\log}$. More details of experiment results will be introduced in Section 5.2.

5 Experiments

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

5.1 Experiment setup and benchmark

We carried out our experiments on PlanetLab (Chun et al., 2003), which is a global overlay network for developing and accessing broad coverage network services. Our experimental environment consists of 100 distributed nodes that serve as cloud nodes. Our framework is implemented with JDK 1.6. In Section 2.2, we mentioned that there is a monitor program running on every cloud node for evaluating the communication and computing performances. To obtain the accurate values of communication performance, our framework measures the response time between two nodes periodically and uses the average response time during a period as the value of performance. The computing power of a cloud node pair is difficult to measure, since cloud nodes are usually heterogeneous. For measuring computing power, we run a benchmark (e.g. calculating *pi*) on each cloud node periodically and use the average execution time of two nodes as the value of computing power. Our framework ran for about 133 days, and we conducted above 9576 times to measure computing power and above 7660 times for communication performance.

Our experiment includes four parts: first, in the case of selecting nodes across multiple clusters, we compare the performance of our topology-aware method against others; second, we conduct a series of experiments for determining the appropriate value of $k_{\log}/k_{phy;}$ third, we deploy multiple collective operations applications to justify that our topology-aware method is also effective to be deployed in collective operations; and fourth, with respect to the load performance, we deploy multiple applications and compare the load performance of our method with those of others. Our experiments use two MPI benchmarks: NAS Parallel Benchmarks (NPB) and Intel MPI benchmark (IMB) (Miguel-Alonso et al., 2007). NPB were derived from Computational Fluid Dynamics (CFD) applications. NPB are a small set of benchmark programs (e.g. CG, MG and BT) designed to compare the performance of parallel computers and are widely recognised as a standard indicator of computer performance. The IMB (Version 3.2) performs a set of MPI performance measurements for point-to-point and global communication

operations for a range of message sizes. The generated benchmark data fully characterise the performance of a distributed system, including node performance, network latency and throughput.

5.2 Performance comparison

To justify the effectiveness, we compare our topology-aware method with the clustering-based methods (Fan et al., 2011; Fan et al., 2012a) that only use clustering analysis to select nodes for an application without considering the topology information.

We use the following two metrics in this experiment:

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

We first use a pre-execution (cf. Section 3.2) and logical topology discoverer (cf. Section 4.1) method to get the topology structures of the programs in NPB. Table 1 shows the topology structure numbers of these programs. The first row of Table 1 displays the numbers of the nodes used for deployment. The problem size and the number of cloud nodes must be assigned when the NPB programs are compiled. In NPB, some NPB programs (e.g. CG and MG) 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. Therefore, SP and BT cannot be deployed on eight nodes. In our experiments, the problem size of CG and MG is Class A, the BT and SP are Class B. The entities in Table 1 are the numbers of topology structures (e.g. the number of topology structures is two when CG is deployed on four nodes) obtained by the logical topology discovery algorithm (cf. Section 4.1). In our experiment, we partitioned cloud nodes into three clusters (which means $k_{phys} = 3$).

Table 1 The number of topology structures of NPB programs

	Deployed on 4	*Deployed on 8*
CG	2	2
MG	2	2
SP	2	–
BT	2	–

In each experiment, we select a small set of nodes randomly from 100 nodes (e.g. as shown in Table 2, we select seven or eight nodes randomly from 100 nodes). Usually, selecting nodes is across multi-clusters. In order to obtain precise results, all benchmarks were run ten times, and we use the average result. In Tables 2 and 3, topology means our topology-aware method and untopology is the method introduced in Fan et al. (2012a) that does not consider the communication topology of a scientific application. These results in Tables 2 and 3 show that for most of the programs, our topology-aware method performs better than untopology method (less execution time and high throughput). The reason is our method deploys applications with respect to their communication topologies.

Table 2 Makespan of different method(s)

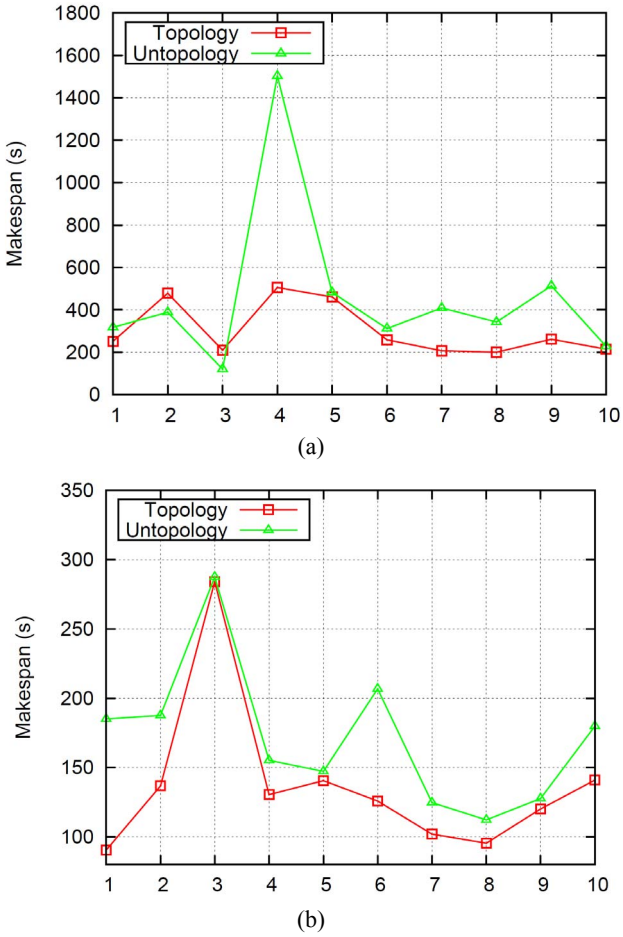
		Topology	*Untopology*
CG.4	7	164.9	268.9
	8	110.0	222.1
MG.4	7	202.0	248.2
	8	130.7	189.1
BT.4	7	81.2	95.5
	8	72.9	83.1
SP.4	7	125.1	130.2
	8	94.5	100.6
CG.8	14	304.6	461.5
	15	195.9	289.1
MG.8	14	136.6	171.4
	15	121.9	170.7

Table 3 Throughout of different method (Mop/s)

		Topology	*Untopology*
CG.4	7	14.18	7.38
	8	18.62	13.14
MG.4	7	23.43	17.88
	8	35.73	30.39
BT.4	7	3.07	2.47
	8	4.18	3.77
SP.4	7	0.92	0.86
	8	1.06	1.04
CG.8	14	5.56	4.75
	15	9.27	6.37
MG.8	14	35.2	24.4
	15	34.4	24.2

Figure 7 shows the detail results of running CG.8 and MG.8 ten times. In most cases, the topology-aware method has a better performance. On the contrary, the untopology method may have a very poor performance in some cases (e.g. the execute time of CG.8 is about 1500 s in the fourth execution). The reason is the nodes in some communication topology structures may be selected from different clusters. Therefore, the communications between nodes have a poor performance.

Figure 7 Details of ten results. (a) Details of CG.9; (b) details of MG.9 (see online version for colours)



5.3 *Grain of logical topology structures*

In this section, we will conduct a series of experiments to determine the grain of logical topology structures. The first thing we should do is to demonstrate the necessity of determining the grain of logical topology structures.

We deploy the programs in NPB on 16 nodes. The benchmark in this experiment is CG.16, and the number of topology structures is four ($k_{\log} = 4$). We randomly select 30 nodes from all the nodes and then partition these nodes into four clusters (which means the number of physical topology structures is four). Figure 8(a) shows the detailed results of executing ten times when the number of logical topology structures equals to four. From Figure 8(a), we can observe that the effectiveness of our topology-aware method is not obvious (the average makespan of the topology method is 1272.7 s and that of untopology method is 1118 s). The reason is that the number of logical topology

structures is bigger and the selected nodes are distributed in the clusters that have poor performance. For example, in this experiment, because the number of topology structures is four, the nodes are selected from four clusters. However, the third or the fourth cluster has a poor performance (descending cluster ranking). For justifying this reason, we set the number of topology structures to be two, which can be obtained by merging the (first, second) and the (third, fourth) topology structures. Figure 8(b) shows the results after changing the number of topology structures. We can observe that in most cases, the topology method is better than the untopology method (the average makespan of the topology method is 426.7 s and that of the untopology method is 575.9 s). The reason is the nodes will be selected from the first and the second clusters for deployment when the logical topology structures are merged into two, and the nodes selected from the first and the second clusters have better performance than those from the third and fourth clusters. Therefore, the topology method will be better than the untopology method.

Figure 8 The results of CG.16 in different number of topology structures. (a) The number of topology structures is four; (b) the number of topology structures is two (see online version for colours)

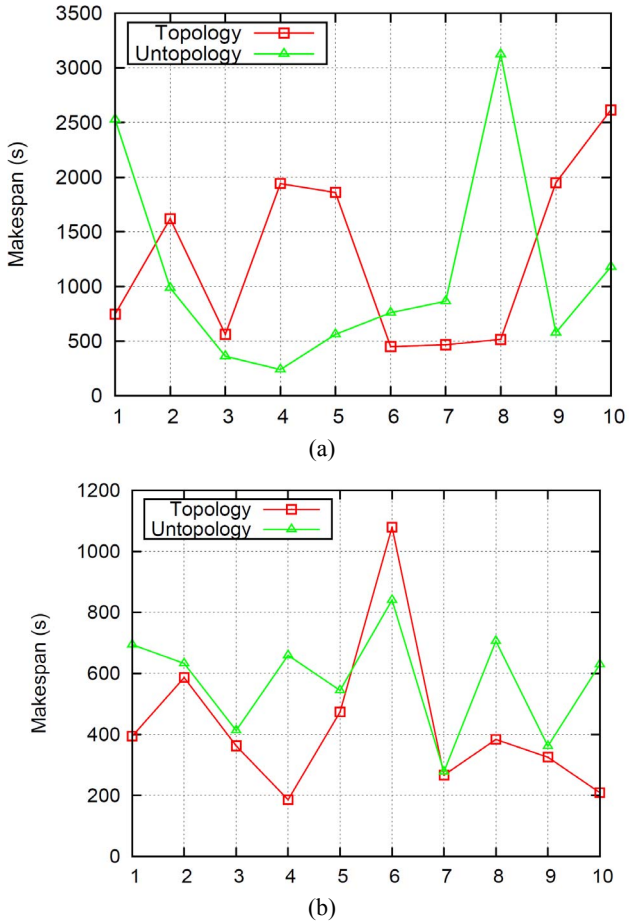
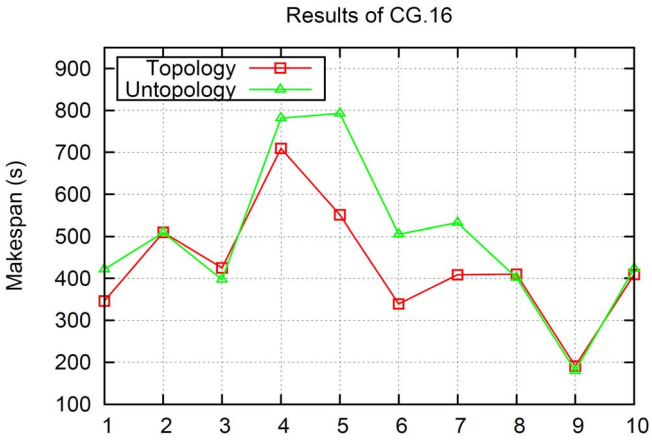


Figure 9 The result of $k_{\log} > k_{phy}$ (see online version for colours)



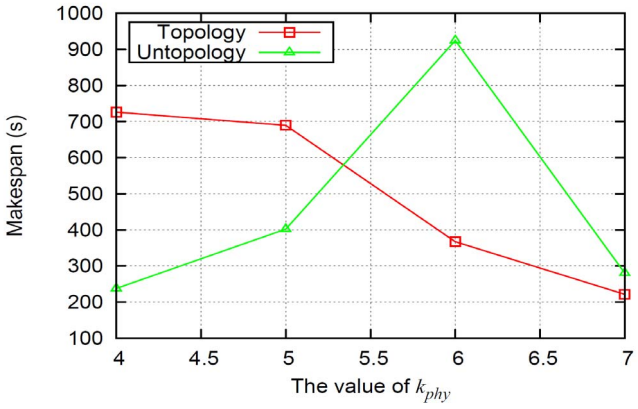
As mentioned in Section 4.4, there are two scenarios when mapping a logical topology to a physical topology. Firstly, the logical topology structures need to be reduced when $k_{\log} > k_{phy}$. In this scenario, we randomly select 30 nodes from all the nodes and set $k_{\log} = 4$ and $k_{phy} = 3$. We deploy CG.16 on 16 nodes and run ten times. We reduce the value of $k_{\log}$ to two by merging the (first, second) and (third, fourth) topology structures. Figure 9 shows the detail results of CG.16 run ten times. In most cases, topology-aware method is better than untopology method (the average makespan of topology method is 430.1 s and that of untopology method is 494.7 s). This experiment shows our topology-aware method can obtain better performance after decreasing the value of $k_{\log}$ to adapt k_{phy} when $k_{\log} > k_{phy}$.

It is difficult to determine $k_{\log}$ when $k_{\log} \leq k_{phy}$. As mentioned earlier, we make the decision with respect to the value of $k_{\log}/k_{phy}$. To study the impact of $k_{\log}/k_{phy}$, we vary the values of k_{phy} from 4 to 7 with a step value of 1, and we set $k_{\log}$ as 4 and 2 in this experiment. The MPI programs used in this experiment are CG.16 and MG.16. All benchmarks were run 3 times, and we use the average result. Firstly, we set $k_{\log} = 4$, and Figure 10 shows the details of the results of the experiment.

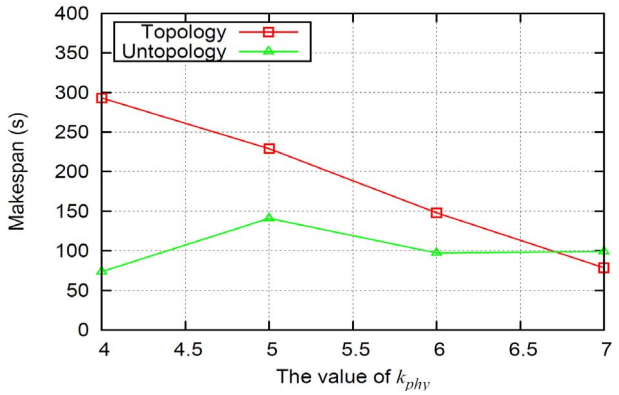
Figures 10(a–b) show the CG.16 and MG.16 results of makespan. Figure 10 shows the makespan of topology-aware method decreases when the value of k_{phy} increased (means $k_{\log}/k_{phy}$ decreased) from 4 to 7. In addition, we can observe that the performance of the topology method is worse than that of the untopology method when $k_{\log}/k_{phy} \geq 0.8$ (k_{phy} in [4–5]) and better when $k_{\log}/k_{phy} \leq 0.67$ (k_{phy} in [6–7]). This is because when $k_{\log}$ is fine-grained (e.g. $k_{\log} = 4$) and k_{phy} has a small value, it will map a logical topology to the clusters that have bad performance.

We conduct another experiment to study the impact of $k_{\log}/k_{phy}$. In this experiment, we reduce $k_{\log}$ to 2 that is obtained by merging the (first, second) and (third, fourth) topology structures. Figures 11(a–b) show the results of CG.16 and MG.16. Figure 11 shows that, in most cases ($k_{\log}/k_{phy} \leq 5$), the topology-aware method has a better performance than the untopology method. The reason is logical topology structures are mapped to the first two physical topology structures that have better performance. From Figures 10 and 11, we can observe that the performance of the topology-aware method is better than that of the untopology method when $k_{\log}/k_{phy} < 0.6$. Therefore, we can reduce the value of $k_{\log}$ to make $k_{\log}/k_{phy} < 0.6$ for obtaining a better performance when $k_{\log} \leq k_{phy}$.

Figure 10 Impact of k_{log}/k_{phy} when $k_{log} = 4$. (a) The result of CG.16; (b) the result of MG.16 (see online version for colours)

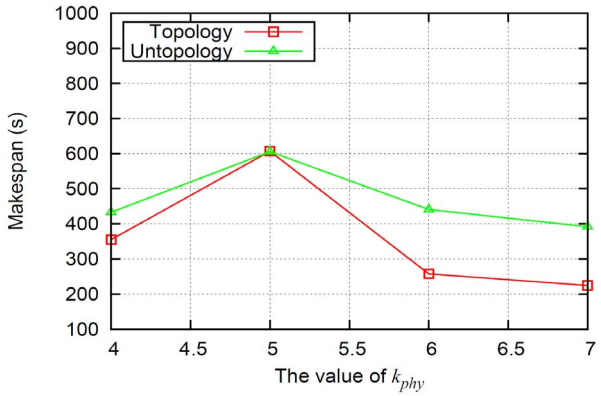


(a)



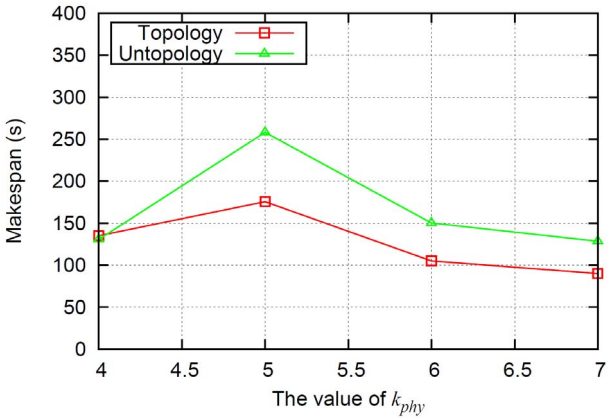
(b)

Figure 11 Impact of k_{log}/k_{phy} when $k_{log} = 2$. (a) The result of CG.16; (b) the result of MG.16 (see online version for colours)



(a)

Figure 11 Impact of k_{log}/k_{phy} when $k_{log} = 2$. (a) The result of CG.16; (b) the result of MG.16 (see online version for colours) (continued)



(b)

5.4 Collective operation experiments

In Section 4.3, we introduced the use of topology information and hierarchical models to optimise collective operations. To justify the effectiveness of this method, we compare it with the untopology method in Fan et al. (2012a). We use IMB benchmark in this experiment. IMB benchmark contains serials of benchmarks, and we choose Scatter, Reduce, Alltoall, Bcast and Barrier collective programs. We randomly select 14 and 15 nodes from all the nodes. These nodes are partitioned into three clusters. We deploy these collective operations applications on eight nodes based on the topology information and the hierarchical model that we introduced in Section 4.3. Each IMB benchmark should be run with some message lengths, except Barrier. In our experiments, the message lengths are varied: 2, 4, …, 1024 bytes (2^1–2^{10}). In order to obtain precise results, all benchmarks were run five times, and we use the average result.

Figures 12 and 13 show the results of running collective benchmarks. From the figures, we have the following observations:

- In Scatter and Reduce benchmarks, the topology method obtains the best communication performance, i.e. has the least latency time under all the different message sizes. The same result is obtained in Alltoall, Bcast and Barrier as well (Table 4).
- With the increasing of message size, the latency also increases, since nodes need more time to transfer messages.

Instead of implementing collective operation algorithms, our optimisation method analyses the common topologies of collective operations and uses the topology information and the hierarchical model to optimise collective operations. The experimental results show the effectiveness of our topology-aware deployment method.

Figure 12 Results of collective operations when the number nodes is 15 (see online version for colours)

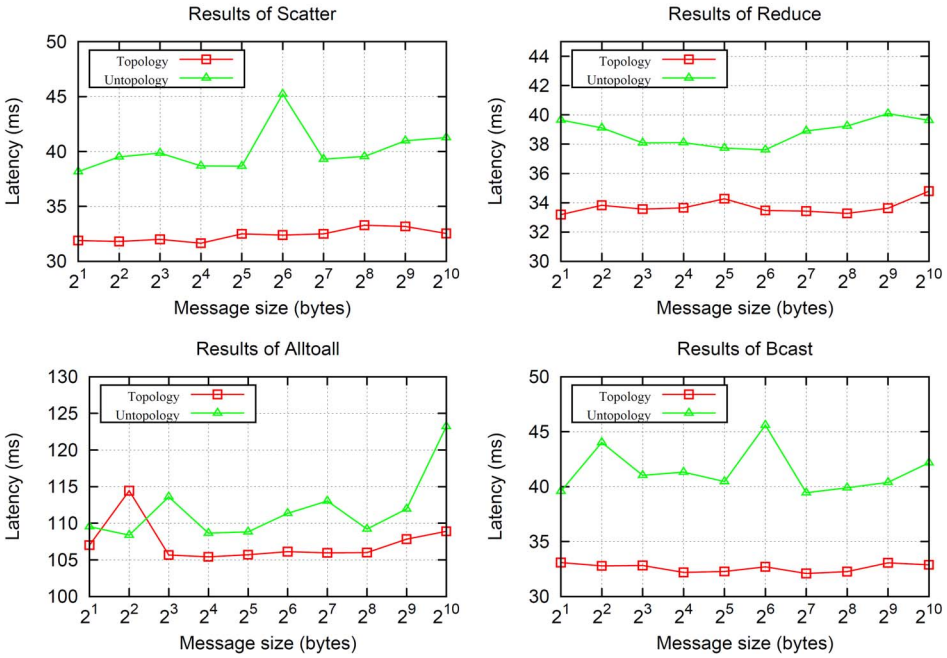


Figure 13 Results of collective operations when the number of nodes is 14 (see online version for colours)

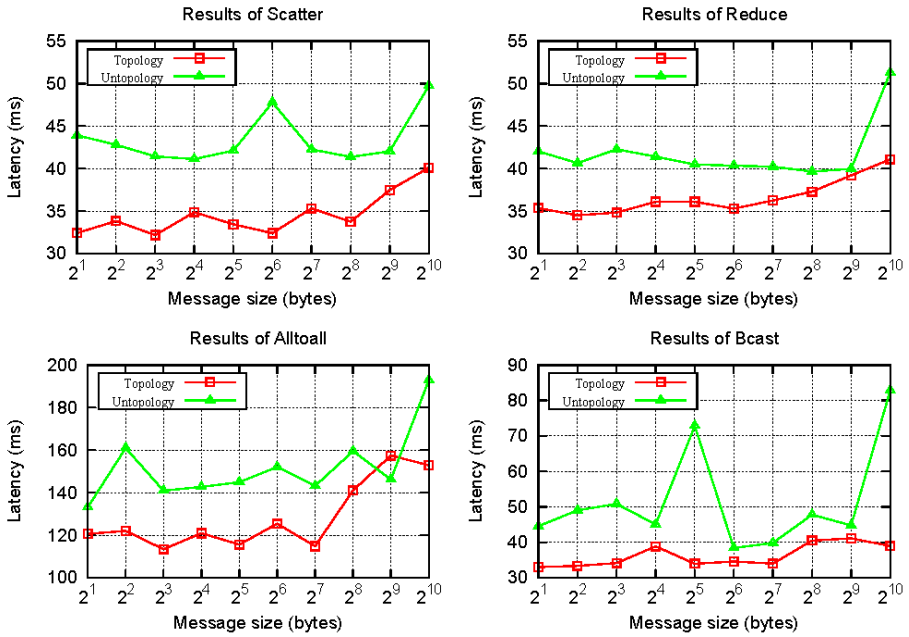


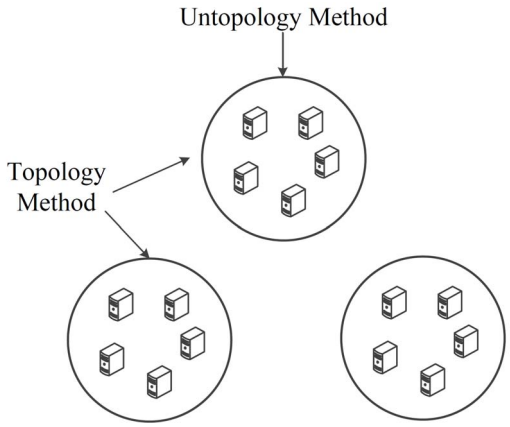
Table 4 Result of barrier (ms)

	15		*14*	
	Topology	*Untopology*	*Topology*	*Untopology*
1	123.9	149.7	135.8	315.4
2	115.1	145.3	140.9	141.2
3	133.8	140.3	128.9	163.8
4	140.3	128.9	144.8	199.5
5	130.4	205.8	142.6	151.7
Average	128.7	154.0	138.6	194.3

5.5 Load experiment

In this subsection, we analyse the load performance when deploying multiple applications on cloud nodes. As mentioned before, the topology method deploys an application based on the communication topology (cf. Figure 5). In this experiment, we use the topology method to deploy multi-applications on two clusters, and use the untopolgy method to deploy the same applications on one cluster (the untopology method cannot consider the communication topology of an application and only uses the best cluster). The deployment processes of these two methods are shown in Figure 14.

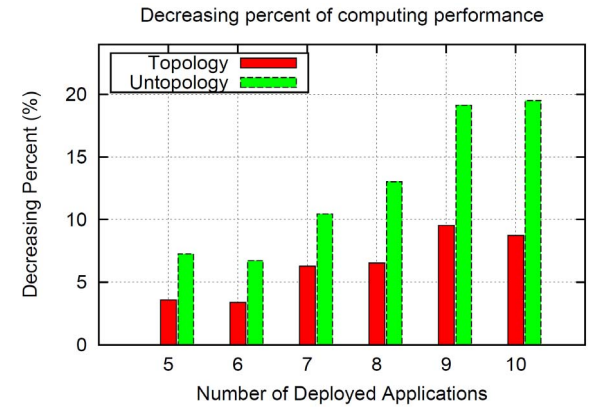
Figure 14 Process of deployment



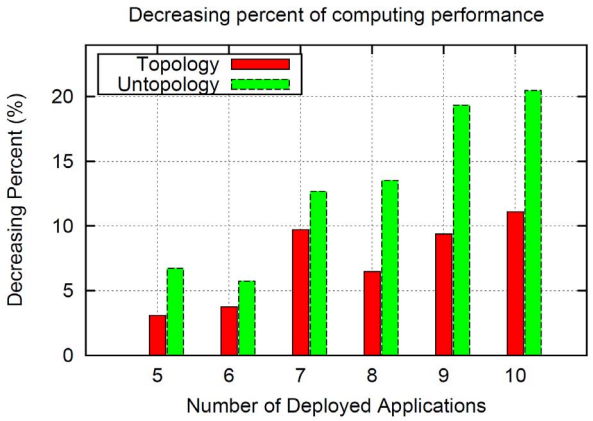
In order to obtain precise results, we partitioned all the nodes into three and four clusters for two methods. The benchmarks used in this subsection are CG.4, CG.8 and CG.16. The number of topology structures of all benchmarks is two, which means we will deploy these applications on two clusters by using the topology-aware method. We change the number of deployed applications from five to ten with a step value of 1. The metrics used in this experiment are the values of decreasing percentages of communication performance and computing performance. Running more scientific applications needs more cloud resources, such as bandwidth and CPU. Therefore, the communication performance and the computing performance will be decreased. Since the topology method deploys applications in multiple clusters, we use the average decreased

percentage as the performance of decreasing use of the topology method. Figures 15 and 16 show the results of decreasing percentages of computing and communication performances.

Figure 15 Decreased percentage of computing performance. (a) Number of clusters is three; (b) number of clusters is four (see online version for colours)



(a)



(b)

Figure 15(a) displays the results when partitioning all nodes into three clusters (cf. Section 4.2), and Figure 15(b) shows the results of partitioning nodes into four clusters. These results show that:

- In all cases, the topology method obtains a lower value of decreasing percentage of computing performance. The reason is these applications are deployed on multiple clusters when using the topology method. This procedure can be viewed as a load balancing procedure.
- With the increasing number of deployed applications, the computing performance is decreased gradually. The reason is more resources are consumed after deploying more applications.

Figure 16 Decreased percentage of communication performance. (a) Number of clusters is three; (b) number of clusters is four (see online version for colours)

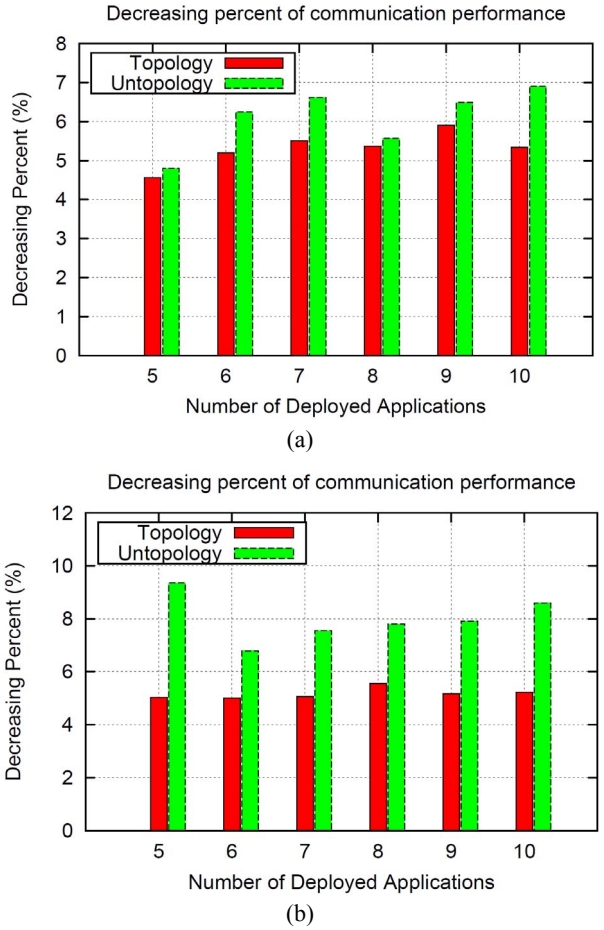


Figure 16 shows the results of decreasing percentages of communication performance, and the results are similar to those of the computing performance experiment.

6 Conclusion and future work

In this paper, we propose an automatic topology-aware deployment method for the scientific applications in cloud. By taking advantage of pre-execution and multi-scale clustering algorithms, our approach does not need cloud users to provide the communication patterns of applications. After obtaining topology information, an application will be deployed on cloud optimally. Extensive experiments are carried out, and the experimental results show that our method outperforms the existing un-topology methods.

In a cloud environment, scientific applications and cloud nodes have a special topology. Currently, we have not considered the user experiences when deploying application in cloud. Our next step includes the study of a user-collaboration-based method for deployment. In addition, online optimisation is importing schedule tasks on IaaS cloud system. Our future work also includes the study of online schedule technology.

Acknowledgements

This research is supported by the National Basic Research Program (973) of China under Grant No. 2011CB302603, and the National Natural Science Foundation of China under Grant Nos. 61100078 and 61161160565, and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CUHK 415311, N_CUHK405/11 of the NSFC/RGC Joint Research Scheme, No. 415410 of the general scheme).

References

Almási, G., Heidelberger, P., Archer, C., Martorell, X., Erway, C.C., Moreira, J., Steinmacher-Burow, B. and Zheng,Y. (2005) 'Optimization of MPI collective communication on BlueGene/L systems', *Proceedings of the 19th International Conference on Supercomputing*, 20–22 June, Cambridge, MA, USA, pp.253–262.

Ananth, G., Anshul, G., George, K. and Vipin, K. (2003) *Introduction to Parallel Computing*, 2nd ed., Addison-Wesley, Boston, MA.

Anderson, D.P. (2004) 'BOINC: a system for public- resource computing and storage', *Proceedings of the 5th International Workshop On Grid Computing*, 8 November, Pittsburgh, USA, pp.4–10.

Bar, P., Coti, C., Groen, D., Hérault, T. and Swain, M.T. (2009) 'Running parallel applications with topology-aware grid middleware', *Proceedings of the 5th International Conference on e-Science*, 9–11 December, Oxford, UK, pp.292–299.

Bessai, K., Youcef, S., Oulamara, A., Godart, C. and Nurcan, S. (2012) 'Bi-criteria workflow tasks allocation and scheduling in cloud computing environments', *Proceedings of the Proceedings of the 5th IEEE International Conference on Cloud Computing*, 24–29 June, Honolulu, HI, pp.638–645.

Bhatele, A., Bohm, E.J. and Kalé, L.V. (2009) 'Topology aware task mapping techniques: an API and case study', *Proceedings of the 14th International Symposium on Principles and Practice of Parallel Programming*, 14–18 February, Raleigh, NC, pp.301–302.

Budati, K., Sonnek, J.D., Chandra, A. and Weissman, J.B. (2007) 'Ridge: combining reliability and performance in open grid platforms', *Proceedings of the 3rd International Symposium on High Performance Computing and Communications*, 27–29 June, Monterey Bay, CA, pp.55–64.

Buyya, R., Abramson, D. and Giddy, J. (2000) 'An economy driven resource management architecture for global computational power grids', *Proceedings of the 6th International Conference on Parallel and Distributed Processing Techniques and Applications*, 26–29 June, Las Vegas, NV, USA, pp.1–9.

Cappello, F., Fraigniaud, P., Mans, B. and Rosenberg, A.L. (2001) 'HiHCoHP: toward a realistic communication model for hierarchical hyperclusters of heterogeneous processors', *Proceedings of the 15th International Symposium on Parallel and Distributed Processing*, 23–27 April, San Francisco, CA, pp.42.

Chan, A., Gropp, W. and Lusk, E. (2008) 'An efficient format for nearly constant-time access to arbitrary time intervals in large trace files', *Scientific Programming*, Vol. 16, Nos. 2–3, pp.155–165.

Chang, F., Viswanathan, R. and Wood, T.L. (2012) 'Placement in clouds for application-level latency requirements' *Proceedings of the Proceedings of the 5th IEEE International Conference on Cloud Computing*, 24–29 June, Honolulu, HI, pp.327–335.

Chen, X.J., Zhang, J. and Li, J.H. (2011) 'A method for task placement and scheduling based on virtual machines', *KSII Transactions on Internet and Information System*, Vol. 5, No. 9, pp.1544–1572.

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.

Coti, C., Hérault, T. and Cappello, F. (2009) 'MPI applications on grid: a topology aware approach', *Proceeding of 15th European Conference on Parallel and Distributed Computing*, 25–28 August, Delft, The Netherlands, pp.466–477.

Evoy, M., Giacomo, V., Schulze, B. and Garcia, E.L.M. (2011) 'Performance and deployment evaluation of a parallel application on a private cloud', *Concurrency and Computation: Practice and Experience*, Vol. 23, No. 17, pp.2048–2062.

Fan, P., Wang, J., Zheng, Z. and Lyu, M.R. (2011) 'Toward optimal deployment of communication-intensive cloud applications', *Proceedings of the 4th International Conference on Cloud Computing*, 4–9 July, Washington, DC, USA, pp.460–467.

Fan, P., Wang, J., Chen, Z., Zheng, Z. and Lyu, M.R. (2012a) 'A spectral clustering-based optimal deployment method for scientific applications in cloud', *International Journal of Web and Grid Services*, Vol. 8, No. 1, pp.31–55.

Fan, P., Chen, Z., Wang, J., Zheng, Z. and Lyu, M.R. (2012b) 'Topology-aware deployment of scientific applications in cloud computing', *Proceedings of the Proceedings of the 5th IEEE International Conference on Cloud Computing*, 24–29 June, Honolulu, USA, pp.319–326.

Faraj, A. and Yuan, X. (2005) 'Automatic generation and tuning of MPI collective communication routines', *Proceedings of the 19th Annual International Conference on Supercomputing*, 20–22 June, Cambridge, MA, pp.393–402.

Foster, I.R.I.T., Zhao, Y. and Lu, S. (2008) 'Cloud computing and grid computing 360-degree compared', *Proceedings of the 4th International Workshop on Grid Computing Environments*, 12–16 November, Austin, TX, pp.1–10.

Gabriel, E., Fagg, G.E., Bosilca, G., Angskun, T., Squyres, J.M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R.H., Daniel, D.J., Graham, R.L. and Woodall, T.S. (2004) 'Open MPI: goals, concept, and design of a next generation MPI implementation', *Proceedings of the 11th European PVM/MPI User's Grop Meeting*, 19–22 September, Budapest, Hungary, pp.97–104.

Gunarathne, T., Wu, T.L., Choi, J.Y., Bae, S.H. and Qiu, J. (2011) 'Cloud computing paradigms for pleasingly parallel biomedical applications', *Concurrency and Computation: Practice and Experience*, Vol. 23, No. 17, pp.2338–2354.

Hadany, R. and Harel, D. (2001) 'A multi-scale algorithm for drawing graphs nicely', *Discrete Applied Mathematics*, Vol. 113, No. 1, pp.3–21.

Hoefler, T., Rabenseifner, R., Ritzdorf. H., Supinski, B.R., Thakur, R. and Träff, L. (2011) 'The scalable process topology interface of MPI 2.2', *Concurrency and Computation: Practice and Experience*, Vol. 23, No. 4, pp.293–310.

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', *Proceedings of the 4th International Conference on eScience*, 7–12 December, Indianapolis, IN, pp.640–645.

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.

Jiang, D., Tang., C. and Zhang. A. (2004) 'Cluster analysis for gene expression data: a survey', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 11, pp.1370–1386.

Kandalla, K.C., Subramoni, H., Vishnu, A. and Panda, D.K. (2010) 'Designing topology-aware collective communication algorithms for large scale InfiniBand clusters: case studies with scatter and gather', *Proceedings of the 22nd International Symposium on Parallel and Distributed Processing*, 19–23 April, Atlanta, GA, pp.1–8.

Kang, Y., Zheng, Z. and Lyu, M.R. (2012) 'A latency-aware co-deployment mechanism for cloud-based services', *Proceedings of the 5th IEEE International Conference on Cloud Computing*, 24–29 June, Honolulu, HI, pp.630–637.

Kang, Y., Zhou, Y., Zheng, Z. and Lyu, M.R. (2011) 'A user experience-based cloud service redeployment mechanism', *Proceedings of the 4th International Conference on Cloud Computing*, 4–9 July, Washington, DC, pp.227–234.

Kapoor, R., Porter, G., Tewari, M., Voelker, G.M. and Vahdat, A. (2012) 'Chronos: predictable low latency for data center applications', *Proceedings of the 3rd ACM Symposium on Cloud Computing*, 14–17 October, San Jose, CA, pp.1–14.

Karonis, N.T., Supinski, B.R., Foster, I.T., Gropp, M., Lusk, E.L. and Bresnahan, J. (2000) 'Exploiting hierarchy in parallel computer networks to optimize collective operation performance', *Proceedings of the 14th International Symposium on Parallel and Distributed Processing*, 1–5 May, Cancun, Mexico, pp.377–382.

Karypis, G., Han, E. and Kumar, V. (1999) *Multilevel Refinement for Hierarchical Clustering*, Technical Report TR-99-020, Department of Computer Science, University of Minnesota, Minneapolis, MN.

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

Kumar, R., Mamidala, A.R. and Panda, D.K. (2008) 'Scaling Alltoall collective on multi-core system', *Proceedings of the 22nd International Symposium on Parallel and Distributed Processing*, 14–18 April, Miami, FL, pp.1–8.

Lacour, S., Pérez, C. and Priol, T. (2005) 'Generic application description model: toward automatic deployment of application on computational grids', *Proceedings of the 6th International Conference on Grid Computing*, 13–14 November, Seattle, WA, pp.284–287.

Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X. and Gu, Z. (2012) 'Online optimization for scheduling preemptable tasks on IaaS cloud systems', *Journal of Parallel Distributed Computing*, Vol. 72, No. 5, pp.666–677.

Miguel-Alonso, J., Mercero, T. and Ogando, E. (2007) *Performance of an InfiniBand Cluster Running MPI Applications*', Technical Report, EHU-KAT-IK-03-07.

Nakada, H., Ogawa, H. and Kudoh, T. (2012). 'Stream processing with BigData: SSS-MapReduce', *Proceedings of the 4th International Conference on Cloud Computing Technology and Science*, 3–6 December, Taipei, Taiwan, pp.618–621.

Newman, M.E.J. (2004) 'Analysis of weighted networks', *Physical Review E*, Vol. 70, No. 5, p.056131.

Noack, A. (2007) 'Energy models for graph clustering', *Journal of Graph Algorithms and Applications*, Vol. 11, No. 2, pp.453–480.

Noack, A. and Rotta, R. (2009) 'Multi-level algorithms for modularity clustering', *Proceedings of the 6th International Symposium on Experimental Algorithms*, 4–6 June, Dortmund, Germany, pp.257–268.

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 scientific computing', *Proceedings of the 1st International Conference on Cloud Computing*, 19–21 October, 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.

Rabenseifner, R. (2004) 'Optimization of collective reduction operations', *Proceedings of the 4th International Conference on Computational Science*, 6–9 June, Krakow, Poland, pp.1–9.

Rao, S., Ramakrishnan, R., Silberstein, A., Ovsiannikov, M. and Reeves, D. (2012) 'Sailfish: a framework for large scale data processing', *Proceedings of the 3rd ACM Symposium on Cloud Computing*, 14–17 October, San Jose, CA,, pp.1–14.

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.

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.

Träff, J.L. (2002) 'Implementing the MPI process topology mechanism', *Proceedings of the 16th International Conference on Supercomputing*, 16–22 November, Baltimore, USA, pp.1–14.

Xie, H., Yang, YR., Krishnamurthy, A., Liu, Y. and Silberschatz, A. (2008) 'P4P: provider portal for applications', *Proceedings of the 24th ACM SIGCOMM Conference on Data Communication*, 17–22, Seattle, WA, pp.351–362.

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

Zheng, Z. and Lyu, M.R. (2013) 'Personalized reliability prediction of web services', *ACM Transactions on Software Engineering and Methodology*, Vol. 22, No. 2, pp.1–28.

Zheng, Z., Zhang, Y. and Lyu, M.R. (2010) 'Cloudrank: a QoS-driven component ranking framework for cloud computing', *Proceedings of the 29th International Symposium on Reliable Distributed Systems*, 31 October–3 November, New Delhi, India, pp.184–193.

Zheng, Z., Hao, M., Lyu, M.R. and Irwin, K. (2011) 'QoS-Aware web service recommendation by collaborative filtering', *IEEE Transactions on Services Computing*, Vol. 4, No. 2, pp.140–152.

Zheng, Z., Zhou, T.C., Lyu, M.R. and King, I. (2012) 'Component ranking for fault-tolerant cloud applications', *IEEE Transactions on Service Computing*, Vol. 5, No. 4. pp.540–550.