Article

v-Mapper: An Application-Aware Resource Consolidation Scheme for Cloud Data Centres †

Aaqif Afzaal Abbasi [1,2,*] **and Hai Jin** [1,*]

[1] Services Computing Technology & System Laboratory, Cluster & Grid Computing Laboratory, Big Data Technology & System Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China;

[2] International School of Software, Wuhan University, Wuhan, China;

* Correspondence: aaqif@hust.edu.cn (A.A.A); hjin@hust.edu.cn (H.J); Tel.: +86-27-8754-3529 (ext. 8033); Fax: +86-27-8755-7354

† This paper is an extended version of paper published in the 9th Asia-Pacific Services Computing Conference (APSCC'15), Bangkok, Thailand, 7-9 December 2015.

Abstract: Cloud computing refers to applications delivered as services over the Internet. Cloud systems employ policies that are inherently dynamic in nature and that depend on temporal conditions defined in terms of external events, such as the measurement of bandwidth, use of hosts, intrusion detection or specific time events. In this paper, we investigate an optimized resource management scheme named v-Mapper. The basic premise of v-Mapper is to exploit application-awareness concepts using software-defined networking (SDN) features. This paper makes three key contributions to the field: (1) We propose a virtual machine (VM) placement scheme that can effectively mitigate the VM placement issues for data-intensive applications; (2) We propose a validation scheme that will ensure that a service is entertained only if there are sufficient resources available for its execution and (3) We present a scheduling policy that aims to eliminate network load constraints. An evaluation was carried out with various benchmarks and demonstrated that v-Mapper shows improved performance over other state-of-the-art approaches in terms of average task completion time, service delay time and bandwidth utilization. Given the growing importance of supporting large-scale data processing and analysis in datacentres, the v-Mapper system has the potential to make a positive impact in improving datacentre performance in the future.

Keywords: network; systems; cloud computing; data centre; performance; software-defined; virtual machine; scheduling; admission control; application-aware

1. Introduction

Cloud computing is taking the computing world by storm. It has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased [1, 2]. One of the big promises of cloud computing is its ability to provide high quality services at a low cost, which is a result of sharing resources among many users. However, the actual ability to provide these services at a low cost critically depends on the efficiency of the resource utilization in the cloud. Despite attaining maturity in many major areas of service provisioning, cloud computing is still developing.

However, even as it rapidly develops, gaps are emerging due to the evolution of existing and new technologies, with these gaps currently being addressed by different work groups, alliances, industries and standard bodies.

In general, there are two serious issues in deploying and provisioning resources in cloud environments, namely virtual machine (VM) placement and lack of precise task scheduling schemes [3, 4]. Refined resource allocation is usually implemented by deploying VM instances and customizing their resources on demand, but this impacts the performance of VMs in completing customers' workloads. Refined resource allocation involves the management of multiple types of resources, such as the CPU, memory and I/O devices. VM placement and scheduling are considered to be critical components of cloud computing because they directly affect a provider's performance. Network operators are responsible for configuring their network resources to enforce various high-level policies and to enable them to respond to a wide range of network events. However, as demand for resources continually increases, networks are experiencing increasing resource management issues, including issues with the allocation, provisioning, requirement mapping, adaptation, discovery, brokering, estimation and modelling of resource needs [2, 5]. Solving these problems would provide significant benefits, such as increased scalability, better quality of service, optimal resource utility, reduced overheads, improved throughput, reduced latency, specialized environment, cost effectiveness and simplified interfaces [6].

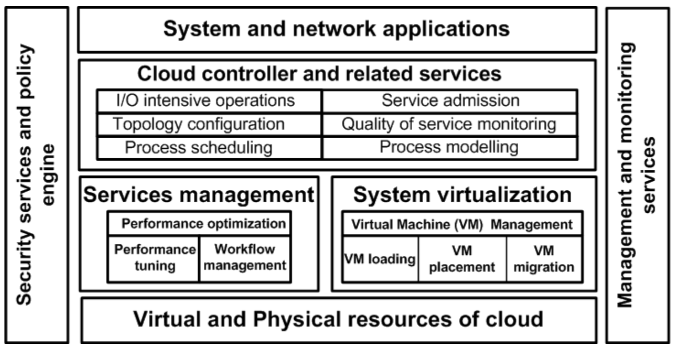
Security services and policy engine
System and network applications
Cloud controller and related services
I/O intensive operations
Service admission
Topology configuration
Quality of service monitoring
Process scheduling
Process modelling
Services management
Performance optimization
Performance tuning
Workflow management
System virtualization
Virtual Machine (VM) Management
VM loading
VM placement
VM migration
Virtual and Physical resources of cloud
Management and monitoring services

Figure 1. An overview of cloud services and applications.

Figure 1 illustrates some of the many cloud services and applications provided by data centres. The question of how to design an appropriate resource management scheme that can satisfy both providers and customers is becoming a major issue in modern cloud environments. One potential solution is that cloud vendors could offer specialized hardware and software techniques in order to deliver higher reliability, albeit this would presumably come at a higher price. This reliability could then be sold to users as part of a service level agreement. The high-availability computing community has long followed the mantra "no single point of failure", yet in terms of reliability, the management of a cloud computing service by a single company is in fact a potential single point of failure and thus a reliability concern [1, 2]. Even if a company has multiple data centres in different geographic regions using different network providers, it may have common software infrastructure and accounting systems that are at risk, or the company could even go out of business with the network going down. Today's networks provide little or no mechanism for automatically responding to this wide range of reliability issues. Nowadays, network operators are trying to implement increasingly sophisticated policies and complex tasks with a limited and highly constrained toolset basically consisting of low-level device configuration commands in a command line interface (CLI) environment [7, 8]. Here, not only are network policies low-level, they are also not well equipped to react to the continually changing network conditions. Indeed, today's state-of-the-art network configuration methods can only implement a network policy that deals with a single snapshot of the network state.

Unfortunately, mainstream operating systems are unaware of synchronization operations within multithreaded applications. This could significantly aggravate both the system throughput and fairness. In addition, network states change continually, and at present, operators must manually adjust their network configuration in response to the changing network conditions. Due to this limitation, operators tend to use external tools or build ad hoc scripts to dynamically reconfigure network devices when events occur. As a result, configuration changes are frequent, and can lead to frequent misconfigurations. State-of-the-art networks typically involve the integration and interconnection of many proprietary, vertically integrated devices [9, 10]. This vertical integration makes it incredibly difficult for operators to specify high-level network-wide policies using current technologies. Innovation in network management has therefore been limited to stop-gap techniques and measures, such as tools that analyze low-level configurations to detect errors or to otherwise respond to network events. Furthermore, proprietary software and closed developments in network devices by a handful of vendors make it extremely difficult to introduce and deploy new protocols.

In order to overcome cumbersome resource allocation challenges, the present paper proposes, presents and evaluates a novel paradigm in the field of cloud resource management named v-Mapper. v-Mapper employs applied graph theory and Fuzzy Analytical Hierarchy Process (Fuzzy-AHP) techniques to optimize network performance behaviour. This helps in providing a clean representation of VMs on a cloud mesh. It also enables system administrators to fully realize the potential of application-awareness concepts by putting them under automated control.

The specific aim of this paper is to establish the necessary baseline for a tool-supported decision support method aimed at facilitating the selection of cloud services in a multi-cloud environment. Herein, we propose v-Mapper, a new scheme that implements application-aware features to oversee VM placement, services admission and task scheduling functions in a cloud environment. V-Mapper is flexible and easy to implement. It uses attributes associated with resource management functions and evaluates the execution costs of individual applications. Based on the method proposed, herein, we also elaborate the suitability of both the method proposed and the state of the art for analyzing risks as well as for ensuring quality and cost in a multi-cloud context.

The rest of this paper is organized as follows. Section 2 discusses the related work in conventional VM management approaches. Section 3 presents background information on application-awareness and related concepts in existing cloud data centres. In section 4, we elaborate v-Mapper's functions and features in detail. Section 5 provides a comprehensive discussion on the test bed and simulation settings. In section 6, we present the performance evaluation results and carefully evaluate our solution. Section 7 concludes this paper and discusses the future work.

2. Related work

In the cloud environment, resources must be shared in such a way that a user's applications do not affect other users' applications. Therefore, the resources of all the users have to be partitioned so that they are private and secure. Our research is related to cloud resource management. In the following section, we highlight some important work performed in cloud resource administration functions.

2.1 VM management in cloud datacentres.

Applications in cloud infrastructures acquire virtual machines (VMs) from providers when necessary. The current interface for acquiring VMs from most providers, however, can be too limiting for tenants. Data-intensive applications [11, 12, 13] need to communicate with datacentres frequently, and therefore, the traffic loads among these VMs are particularly heavy. This can also increase system overhead and potentially degrade the network performance. Energy efficient placements seek to consolidate VMs for resource utilization, which can also greatly impact network performance. This can lead to situations in which VM pairs are placed on host machines whose traffic volumes are already large.

VMs typically tend to follow patterns in their resource demands. A study [14] of a large number of CPU traces from production servers showed that most of the demand traces had a high correlation with a periodic behaviour. Statistical multiplexing can also be used to exploit possible correlations among VMs. A continuous monitoring approach to study VM resource usage patterns on all running VMs has been presented in [15, 16].

Recently the problem of VM placement has been extended to include other important aspects of the data centre infrastructure, such as network traffic and storage facilities. In particular, VMs deployed in cloud infrastructures typically show a network traffic dependency that can be better accommodated by consolidating interdependent VMs in the same physical machine [10]. Interestingly, network topology and the structure of the data centre highly influence the choice of the placement under a traffic optimization goal [12]. Similar dependencies also appear between VMs and the storage resources that must serve different clients. In this case, applications requiring higher I/O throughput can be placed closer to the storage. In [11], the authors carried out some investigations in this aspect. They proposed a strategy to place a VM on the physical machine with the smallest data transfer time for the required data in consideration of the network speed.

However, supporting tenant-controlled VM placement makes it difficult, if not impossible, to perform simple resource management tasks. We believe that if application-awareness concepts were to be employed, it would help provide a simple and clean interface for datacentre administrators.

2.2 SDN role in datacentre optimization

Datacentres not only provide a flexible and reliable storage space but also support the underlying virtualization infrastructure. SDN solutions focus on the need for the use of programmability in the data centre network by encouraging the implementation of comprehensive management capabilities that can conclusively demonstrate operation compliance. This helps networks to reap a number of benefits, such as cost reduction and ease of management [17]. SDNs are reshaping the future of networks by changing the behaviour of conventional network operations. An SDN's management plane can be seen as a management-cum-control platform that accommodates traditional network management services and protocols. The Open Network Foundation (ONF) workgroup on Configuration and Management is working to enhance the operation, administration and management capabilities of the OpenFlow protocol. Proposals such as from Procera [7], Frentic [18] and Meridian [19] encourage the use of SDN solutions to enhance network management capabilities.

3. Application-awareness concepts in cloud datacentres

Conventional networking-based cloud resource administration techniques fail to address the dynamicity in the networking arena, especially ones that require a lot of human intervention to provide the network resources in an on-demand manner. Software-defined networking is a key factor driving the software-defined cloud (SDC) research community. Figure 2 presents a typical software-defined cloud environment. In order to effectively manage the middlebox processing capabilities in SDCs, the concept of outsourcing enterprise middlebox processing to the cloud was proposed in [20] with a system the authors called APLOMB (Appliance for Outsourcing Middleboxes). APLOMB delegates middlebox tasks to the cloud to ensure the ease of management and capital expenditure (CAPEX) functions.

In multi-tenant datacentres, tenants need a flexible solution for migration that can also ensure that, during the migration of unmodified workloads from enterprise networks to the service provider network, the network configurations will not change. This is an extremely challenging task and traditional networking approaches have so far failed to address this task. These concerns are addressed by the Network Virtualization Platform (NVP) in [21]. NVP has been proposed for use with multi-tenant datacentres using SDN as a base technology. It uses SDN to decouple network management from the physical infrastructure in two profound ways: first, by decoupling the tenant control planes from the physical infrastructure, and second, by decoupling the implementation of NVP from the physical infrastructure by using software switching at the edge.

The easy creation and manipulation of software-defined cloud environments enable the model-based deployment of upcoming standards, such as the Topology and Orchestration Specification for Cloud Applications (TOSCA) [22]. This also allows a model description to be built for different kinds of workloads, such as for topology patterns and management plans. TOSCA also provides a flexible combination of declarative and imperative models of workloads that can be seamlessly integrated with automation frameworks for task-level automation and optimized resource orchestration. In order to effectively monitor service-centric cloud environments, a unified solution was presented in [23], which can rapidly migrate and optimize the use of cloud resources in ways that are consistent with the service profiles and policies. The reliable implementation of application-aware concepts through SDN implementation is now an essential requirement for leading data centre solutions. We believe that a combination of open standards technology (SDN technology) with a best-of-breed hardware platform (SDN-enabled hardware peripherals) can deliver flexible, scalable and high performance networking solutions for the future networking industry.

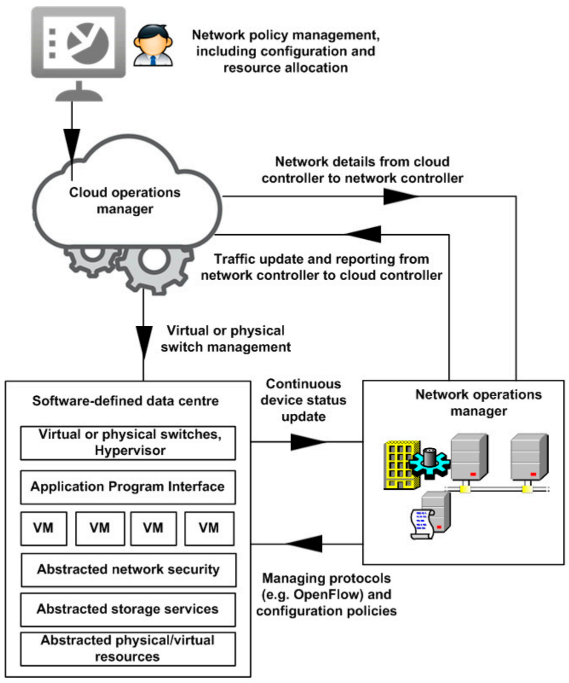
Network policy management, including configuration and resource allocation
Cloud operations manager
Network details from cloud controller to network controller
Traffic update and reporting from network controller to cloud controller
Virtual or physical switch management
Software-defined data centre
Virtual or physical switches, Hypervisor
Application Program Interface
VM
VM
VM
VM
Abstracted network security
Abstracted storage services
Abstracted physical/virtual resources
Continuous device status update
Network operations manager
Managing protocols (e.g. OpenFlow) and configuration policies

Figure 2. A view of the services and functions in a software-defined cloud data centre

4. Data centre design and the v-Mapper system model

Below we describe the v-Mapper model and its features in detail.

4.1 VM placement and the data centre model

Modern data centre architecture designs rely on the path multiplicity to achieve horizontal scaling of hosts [1, 23]. Therefore, data centre topologies are very different from conventional enterprise network topologies. Data centre topologies employ various forms of multi-rooted spanning trees to address congestion-related constraints. For the development of our model, we started by modelling a data centre network and its application jobs as a static problem. We then formulated a joint VM placement and scheduling strategy that aimed to minimize these constraints.

In our model, we consider a cloud environment consisting of two storage nodes SN1 and SN2 and three compute nodes CN1, CN2 and CN3. We also use three data packets DP1, DP2 and DP3. The modelling scheme presented is similar to that presented in [24], except for the fact that we consider a graph theoretic representation for our cloud architecture. The presented cloud scenario is mapped in Figure 3.

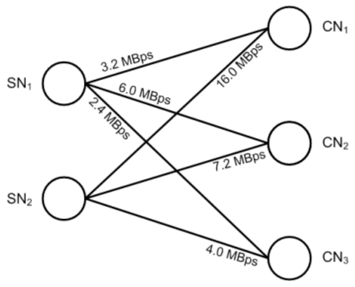

Figure 3. CN – SN relationship as a bipartite

We then interpret the nodes and edges relationship (in Figure 3) by developing an adjacency matrix:

$$B = \left[b_{ij}\right]_{n\,x\,m} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix} \quad (1)$$

By populating data in the adjacency matrix (the same way as mentioned in [24]), we obtain the following matrix:

$$B_{3,2} \begin{bmatrix} 3.2 & 16.0 \\ 6.0 & 7.2 \\ 2.4 & 4.0 \end{bmatrix} \text{MBps} \quad (2)$$

By loading the data packets (DPs) on the storage nodes (SNs), the matrix can be represented as:

$$D_{2,2} = [d_{ki}]_{2x2} = \begin{bmatrix} DP_1 & DP_3 \\ DP_2 & 0 \end{bmatrix} = \begin{bmatrix} 200 & 500 \\ 100 & 0 \end{bmatrix} \text{MB} \quad (3)$$

The response time R of an individual node can be calculated as:

$$\left[\tau_{ij}\right]_{n\times m} = \left[\frac{1}{b_{ij}}\right]_{n\times m} \quad where\ 1 \le i \le n\ and\ 1 \le j \le m \quad (4)$$

Therefore, the total response time for a CN can be calculated by using the following expression:

$$t_{CN,i} = \sum_{j=1}^{n} t_{R,ji} \quad (5)$$

We briefly present our VM placement scheme in Algorithm 1.

Algorithm 1 VM placement

```
1:  Input: Load DP on SN
2:  Output: VM placement decision
3:  Let CN denote Compute Node of a cloud network
4:  Let $t_{CN,i}$ denote service response time of a CN i
5:  Let least denote least response time of all CNs
6:  Let j denote the CN with least response time
7:  Calculate $t_R$ for each CN as
8:          $t_{CN,i} = \sum_{j=1}^{n} t_{R,ji}$
9:  $i \leftarrow 0$
10: $j \leftarrow 0$
11: least $\leftarrow t_{CN,0}$
12: while $i < n$ do
13:     if least $> t_{CN,i}$ then
14:         least $\leftarrow t_{CN,i}$
15:         $j \leftarrow i$
16:     end if
17:     $i \leftarrow i+1$
18: end while
19: Place VM on CN j
20: exit
```

4.2 VM workload admission

doi:10.20944/preprints201702.0074.v1

VM resource provisioning plays a key role in ensuring that cloud providers adequately accomplish their obligations to customers, while maximizing the utilization of the underlying infrastructure. A capable workload admission scheme requires dynamically allocating each service request with the minimal resources that are needed for acceptable fulfilment of that service requirement, leaving the surplus resources free to deploy more VMs.The decision-making process for workload admission in v-Mapper takes place through a fuzzy comprehensive evaluation [25]. The process involves a number of steps, as described below.

1) First, we determine the factor and sub-factor weights for cloud services through an evaluation index (U).

2) "Not all clouds are created equal"; therefore, we create a set of comments (V) to describe the evaluation of cloud services by using phrases like "Acceptable", "Constrained", etc. These are determined based on Saaty's 1 to 9 scale [26] to describe the preferences between alternatives as being either equally, moderately, strongly, very strongly or extremely preferred.

3) In order to provide a comprehensive assessment methodology, we create an evaluation matrix(R) from U to V, where each factor ui(i ≤ n) can be written as a fuzzy vector R_i∈μ(V). Mathematically, this fuzzy relationship can be expressed as:

$$\boldsymbol{R} = \left(r_{ij}\right)_{nxm} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{pmatrix} \tag{6}$$

The evaluated result of equation 6 should match the normalized conditions, because the sum of the weight of the vector is 1 (i.e. for every i, ri1 + ri2 + ri3 +……..+rim =1).

4) A factor assigned to a number in a computation system reflects its importance.

The greater the weight of a factor is, the more importance it has in the system. We, therefore, determine the factor weight (FW) of each factor in the evaluation index (U) system.

5) We obtain the evaluation result (E) through the product of the factor weight (FW) and the evaluation matrix (R). This can be denoted as E=FW(R) = (E1, E2, E3,…,Em). Finally, the evaluated weight now can be assigned to the respective application.

In the next step, the individual cloud node workload is computed. By representing the cloud model as an arrangement of two interdependent row vectors, v-Mapper considers a cloud scenario where the number of tasks performed by the cloud is represented as a row vector a = [a1, a2, a3,…,an] of a resource class c. Similarly, the resource occupation of individual tasks can be represented by another row vector v = [v1, v2, v3,….., vn]. Each member of the row vector v represents the resource usage for its respective element in row vector a. By considering a case where the number of service requests accepted by the cloud environment does not exceed its aggregate compute nodes, the condition can be expressed mathematically as:

$$\boldsymbol{a.v} = \sum_{i=1}^{n} a_c v_c \leq N \tag{7}$$

where N shows the number of aggregate compute nodes. Due to the varying allocation models and schemes, workload evaluation for cloud services is increasingly complex. However, applying probability to schemes with varying behaviour can address these concerns. Therefore, by employing the recursive methodology approach presented in [27, 28], we can calculate the probability p of an individual cloud node capacity q as:

$$\boldsymbol{p(q)} = \sum_{a:av=n} \frac{\alpha_1^{a_1}}{\alpha_1!}, \dots, \frac{\alpha_1^{a_n}}{\alpha_n!} n = 0,1,2,\dots,N \tag{8}$$

The measured node occupancy probability can be input to the Fuzzy-AHP for fine tuning its decision structuring. Now, the submitted workload occupancy probability in the proposed cloud system can be evaluated. To ensure it upholds the workload occupancy of the entire cloud system, v-Mapper ensures that the resource allocation is made under the cloud's permissible resource threshold limits. The workload admission control algorithm (Algorithm 2) ensures that sufficient resources are available to entertain a request. If resources are scarce, a message about the unavailability of resources is prompted. This conditional approach used in v-Mapper makes it simple yet effective to implement. In this experimental study, the available resources of a cloud are considered as a set containing the cloud's physical resources, the computational resources, the memory and the bandwidth resources.

Algorithm 2 VM workload admission

```
1:  Input: Workload admission request
2:  Output: Workload admission decision
3:  Let $res_{ava}$ denote the available resources for a VM
4:  Let $res_{req}$ denote the requested resources by a VM
5:  Let queue denote the request queue
6:  while (!queue.isEmpty())
7:  {
8:      req = queue.firstRequestResource();
9:      if ($res_{req} \leq res_{ava}$)
10:     {
11:         $res_{ava} = res_{ava} - res_{req}$
12:         queue.pop();
13:     }
14:     else
15:     {
16:         sendMessage("resources inadequate");
17:         queue.pop();
18:     }
19: }
```

4.3 VM scheduling scheme

v-Mappers' scheduling policy (for admitted applications) involves locating resources that are subject to multiple global optimization constraints and that require searching of a large pool of resources. The proposed scheduler satisfies these requirements by using a fair approach to managing workloads. Our study considered three scenarios (Figure 4) to evaluate v-Mapper's scheduling policy.

- Scenario 1: The VM hosting the applications is manually scheduled to receive batch processing of the incoming resource demands. The demands accepted by the VCPU of the VMM are synchronized before processing them.

Scenario 2: The scheduling strategy for real-time scheduling is set up in such a way that all the VCPUs contain the upcoming scheduling requests based on a load-sharing policy (LSP) [29], which optimally dispatches the incoming workloads according to the current availability of the VMs. The LSP facilitates the scheduling process by permanently engaging the minimal number of available VMs for the incoming workloads.

Scenario 3: We developed a load-sharing scheme that manages the VMs on the basis of the incoming workloads, enabling all the VCPUs to be scheduled and synchronized in response to the received workloads. This eases VM scheduling and improves the comprehensive understanding of the virtual infrastructure at both the physical and logical levels.

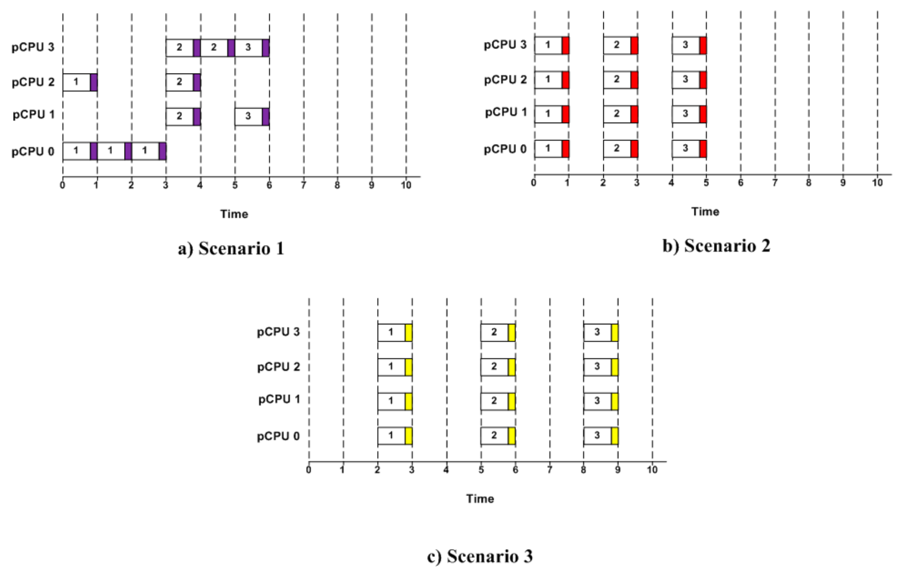

Figure 4. v-Mapper – Scheduling scenarios

Theoretical analysis: v-Mapper uses a load-sharing concept to improve fairness. Usually, scheduling policies are governed by optimizing a generalized proportional fairness objective. Therefore, compared to other scheduling techniques, our approach enforces a more stringent fairness and load balance among workloads. Specifically, it assigns tags to each request when they arrive, and then dispatches requests in a non-descending order of tags. v-Mapper also delivers a statistically fair service even when the service rate fluctuates due to varying link capacity or varying CPU processing rates, etc.

To demonstrate our load-sharing approach, let Ra be a scheduling request that carries a workload capacity Wa. This request is shared among r resources of a cloud's total available shared resources SR. The shared fair load SFL of an admission-control request Ra for each shared cloud resource SR is then defined as:

$$W_a^{SFL} = {W_a}/{r} \tag{9}$$

The shared fair load SFL of the shared resource SR for the scheduling request query Qreq can be defined as:

$$\mathrm{S}_\mathrm{R}^\mathrm{SFL} = \sum_{\mathrm{R_a} \in \, \mathrm{Q_{req}}} \mathrm{W}_\mathrm{a}^\mathrm{SL} \tag{10}$$

The scheduler prioritizes and accepts resource-intensive applications requiring maximum load-share capacity because their processing requirements fit in most with the system capacity. We then assign a priority Prq to the individual resource demand (per requesting application), whereby we consider the priority as a workload admission control request: shared fair load ratio. For the described scenario, the workload admission control request's shared fair load ratio can be expressed as:

$${Q_R}/{S_R^{SFL}} \tag{11}$$

We next define a priority function (line 1 in Algorithm 3) to ensure that the service requests (i) are treated fairly in the queue (memory buffer). We apply conditionality (line 2) for execution, where a request query can only be entertained once all the resources required for its execution are available. This also guarantees that the total resource requirements of the requesting application cannot exceed the system threshold capacity (Rt) (see line 2). We then provide individual applications with their respective priority value (Pqi) (line 3). This priority value is calculated by the priority function (equation 11). In the next step, we queue all these buffered requests (br) in descending order (lines 7–13). This is done to ensure that the requests with the smallest resource requests but higher priority value are entertained first. Finally, the sorted resource-demanding applications are forwarded for process execution (lines 15–16).v-Mapper's VM scheduling policy is presented in Algorithm 3.

Algorithm 3 VM scheduling policy

Input: *qi*: represent a new request *i*; *Rqi*: the (required) resources for a service request *i*; *Rt*: system threshold for the resource, *Pqi*: priority of request *i*; *N*: request number in a memory buffer *br*.

Output: scheduling decision

```
1:   Let $priority = \frac{Q_R}{S_R^{SFL}}$
2:   while R$qi \leq$ Rt do
3:       Assign request $q_i$ its calculated priority value
4:       Push $q_i$ to br
5:   end while
6:   /* Sort buffer requests by priority in descending order*/
7:   for$i \leftarrow 0$ to N do
8:       for$j \leftarrow 0$ to N-2-$i$do
9:           if $Pq_j < Pq_{(j+1)}$do
10:              $Pq_j \leftrightarrow Pq_{(j+1)}$
11:          end if
12:      end for
13:  end for
14:  /*Process all sorted VM requests*/
15:  for $i \leftarrow 0$ to N do
16:      Process($q_i$)
17:  end for
18:  exit
```

5. Performance test bed and simulation environment

Below, we present a detailed study on the simulation settings of our system. The datacentre architectures used in the simulation were Fat-Tree and B-Cube, while the resource distributions used in the experiments were drawn from an empirical distribution given in [30]. The study report here covers our designed baseline strategies, the simulation environment and the results (section 6).

5.1 Baseline strategies and compared scenarios

In this section, we report on the evaluation of our algorithms against various data centre topologies. To provide benchmarks for our evaluations, we considered the following baseline strategies.

- Displacement strategy (ds): The displacement strategy selects a path with minimum end-to-end node congestion.It is frequently used in VM clusters handling small data loads.
- Blind placement (bp): In a blind placement strategy. the tenant VMs adopt the path with a minimum hop count.
- Sequential placement (sp): Here, the tenant VMs are placed sequentially in a server queue, where the VMs are ordered on the basis of their increasing machine-id order (lowest to highest

order). This strategy attempts to balance the load resilience of applications by improving host subscription.

- Stochastic (Random) placement (stp): Also known as the random placement strategy, in SP, the VMs are distributed in stochastic order. The stochastic placement strategy randomly picks available host machines from within a specified availability zone for a job.

5.2 Simulation set-up and settings

We conducted the performance evaluation of the proposed framework initially by simulating the physical resources of a data centre and then by emulating their execution on a fixed number of applications and services. For this, we generated a mix of services and applications that, to the best of our knowledge, are representative of those within a real cloud environment.

We considered two types of topologies: physical topology and logical topology. To ensure optimal switching functions and traffic control, the physical cloud infrastructure used in our testing scenario consisted of eight hosts connected over seven SDN-enabled switches. We deployed our framework using an API that uses the SDN-enabled EPC Gateway to communicate with the SDN-enabled switching mesh. The simulations on the network devices and hosts were conducted by using the OpenDayLight controller and Mininet v.2.2.0 on an AMD Opteron™ 6300 Series Platform with a 16-core x86 processor.

For the logical topology, we constructed a workload emulating the DC traffic. We also ran our experiments using a real workload from large computing clusters at Huazhong University of Science and Technology. The workload contained 6 days of cluster machine activity and included details like job number, arrival, waiting and service time, CPU and memory usage traces. We compared five algorithms i.e. vmap (v-Mapper), sp-bp, sp-ds, stp-bp and stp-ds on two datacentre topologies: Fat-Tree and B-Cube. Fat-tree is a scalable design for network architecture. The unique feature of a fat-tree data centre architecture is that for any switch, the number of links going down to its siblings is equal to the number of links going up to its parent in the upper level. Therefore, the links get "fatter" towards the top of the tree. The switch in the root of the tree has the most links compared to any other switch below it. B-Cube [31], on the other hand, is a server-centric network architecture in which servers with multiple network ports connect to multiple layers of mini-switches, as shown in Figure 5. By employing a range of topology settings, traffic and data applications, we ensured maximum core switch utilization to estimate the network resources utilization.

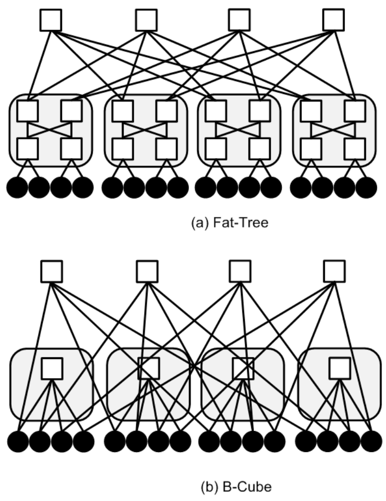

Figure 5. Topologies used in the performance evaluation.

6. Performance results and analysis

In this section, we present our findings to verify the effectiveness of the v-Mapper system.

6.1 Impact on data centre topologies

Our proposed strategy takes into account the measured (real-time) communication dependencies among VMs and the underlying data centre network topology as well as the capacity limits of the physical servers of the data centre. Its major goal is to minimize the data centre network traffic while satisfying all the server-side constraints. We assign every link in every topology the same capacity. To ensure uniformity, a synthesized traffic pattern is adopted. The VM resources are uniformly distributed and occur between range values of 0.2 and 0.8. From the results in Figure 6, vmap shows less congestion in the maximum core switch for the Fat-Tree and B-Cube topologies.

Our technique minimizes the path length between two intercommunicating VMs. This leads to an efficient and better usage of the link and switches resources. It also results in an increase in the blocking ratio of groups of VMs with a high intensive traffic rate. Therefore, the performance improvement of vmap is more significant in topologies with better connectivity and path diversity.

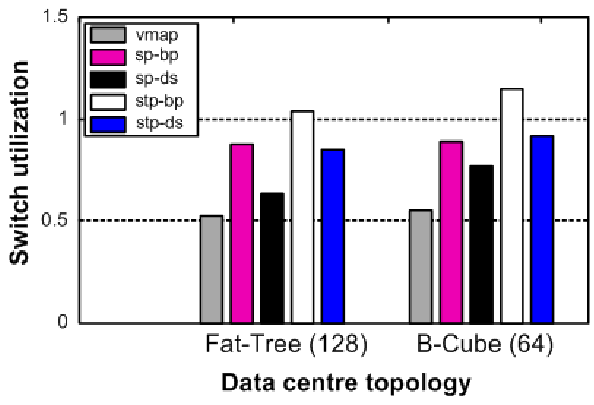

Figure 6. Performance evaluation of the algorithms with the different data centre topologies

6.2 Performance cost

Performance can be assessed under numerous aspects. Here, we investigated scenarios where the hardware resources could be fully utilized. In order to evaluate the performance cost, we evaluated the performance of all the algorithms using a real workload obtained from our computing clusters. The details of 2,500 jobs were extracted to evaluate their arrival times and flow duration statistics. These details were then catered for the Fat-Tree and B-Cube topologies. Figure 7 and Figure 8 illustrate the detailed performance cost for all the compared algorithms with static and varying data traffic between VMs, respectively. From the results, it is clearly observed that as we increase the intensity of traffic between VMs, the overall performance cost increases. This increase results in congestion among communicating VMs and the slowing down of the networks' overall performance. Due to this, the performance of stp-bp declines. This can be understood as the stp-bp strategy is dependent on hop counts; therefore, if the traffic intensity among VMs is high, it adversely affects the algorithm's performance. On the other hand, vmap works independent of these constraints and has therefore the minimum cost among all the tested algorithms in the experimental set-up.

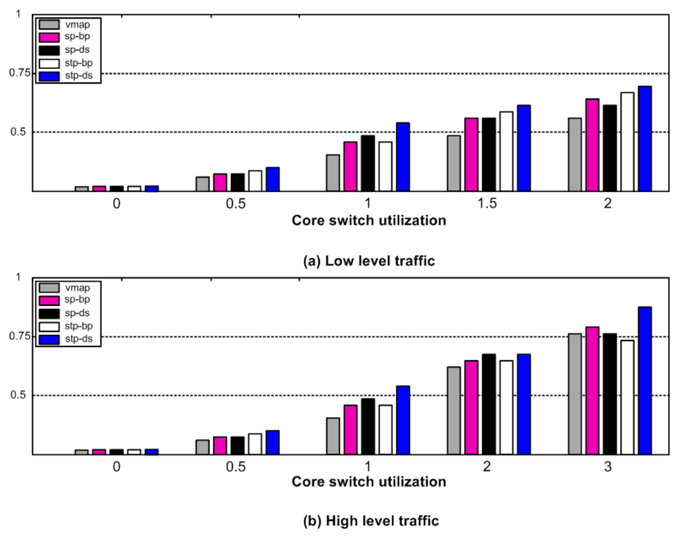

Figure 7. Performance evaluation of the algorithms using the Fat-Tree (128 node) data centre topology

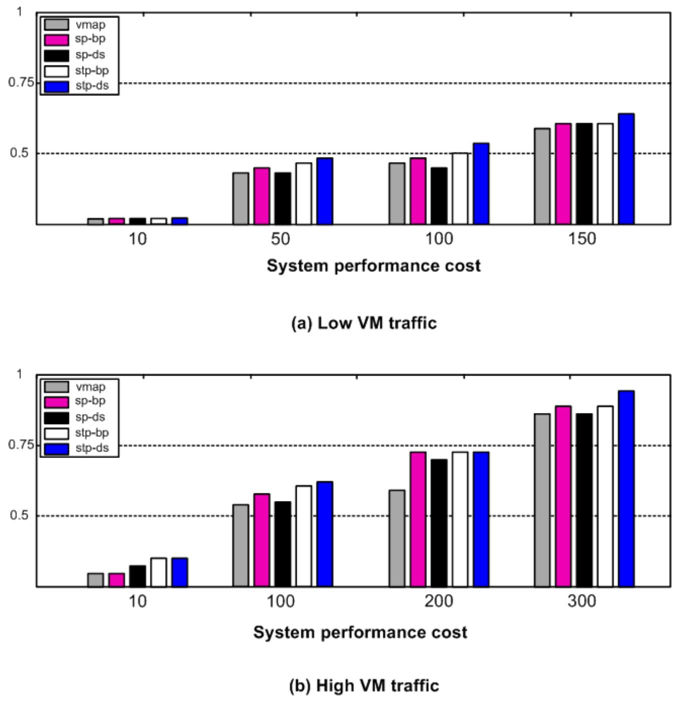

Figure 8. Performance evaluation of the algorithms by varying the traffic intensity between VMs

6.3 VM resource occupancy

The distribution of VM image data should be based on the unit of chunks, instead of on entire VM image files. Herein, we first studied the distribution of the number of instances created from individual VMs. The VM size distribution is the amount of resources required by each VM instance to carry out discrete events. This categorization provides the basis for understanding how VM images are constructed and how similar they are. Different data centre applications have their own patterns of resource consumption, e.g. CPU-intensive applications or memory-intensive applications, etc. To represent a clean and simple representation of a proposed scheme, our study defines only one type of resource consumption pattern. The evaluations were performed on 125-host B-Cube architecture with a synthesized traffic pattern. The application requests were categorized in four major categories i.e. uniform small, uniform large, random and bimodel demand. Further detail of the application request ranges employed in our experiments is given in Table 1. The results in Figure 9 demonstrate that vmap outperformed the other algorithms in the uniform small and random schemes; whereas in the uniform large and bimodal demand schemes, its efficiency showed a 30 percent improvement compared to the other algorithms. This makes vmap (v-Mapper) a simple, feasible, yet practical VM management and resource consolidation solution.

Table 1. Classification of VM resources.

Size	Request range
Uniform small	30 percent
Uniform large	100 percent
Random	0 to 100 percent
Bimodel demand	Normal distributions *N*(0:3;0:1) and *N*(0:7;0:1) with equal probability

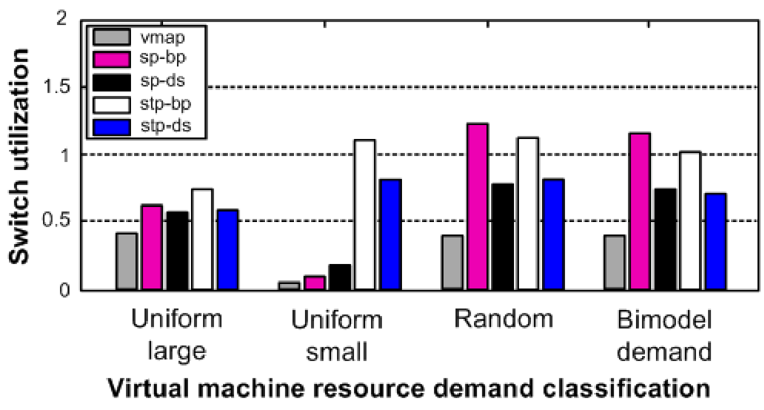

Figure 9. Switch resource utilization for the compared resource allocation schemes

7. Conclusion and future work

With the prevalence of virtualization and multi-core processing architectures, hosting VMs and their applications is becoming a challenging task in cloud data centres. In this paper, we identify the VM administration concerns in cloud datacentres. We optimized VM placement through a graph theoretic approach under controlled parameters. The representation of the VM placement problem presented in this paper and the use of matrices (from linear algebra) result from the fact that cloud systems can be represented as a network of nodes and edges forming a graph. Herein, we proposed an algorithm that solves the scheduling of applications using an online solution for a dynamic environment with changing traffic (service admission-control). By leveraging and expanding capacity approximation techniques, we propose v-Mapper, which is an efficient solution that restricts the synthesized data centre traffic traces under a spectrum of workloads. Furthermore, we demonstrated that v-Mapper offers a consistent and significant improvement over conventional benchmarks used in cloud data centres.

In future work, we plan to exploit the effects of topology changes for our framework. We also plan to investigate a best-fit approximation solution for reducing the VM overheads.

Acknowledgments: This research is supported by National Science Foundation of China under Grant No. 61232008, National 863 Hi-Tech Research and Development Program under Grant No. 2015AA01A203, and the Fundamental Research Funds for the Central Universities under Grant HUST: 2016YXZD016.

Author Contributions: Aaqif Afzaal Abbasi and Hai Jin designed the study. Hai Jin designed experiments. Aaqif Afzaal Abbasi performed experiments. Aaqif Afzaal Abbasi and Hai Jin analyzed the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abbasi, A.A.; Jin, H.; Wu, S. A Software-Defined Cloud Resource Management Framework. In Proceedings of the Asia-Pacific Services Computing Conference, Bangkok, Thailand, 7-9 December 2015.
2. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; Zaharia, M. A view of cloud computing. *COMMUN ACM* **2010**, 53, 50-58.
3. Wu, S.; Zhou, L.; Sun, H.; Jin, H.; Shi, X. Poris: A Scheduler for Parallel Soft Real-Time Applications in Virtualized Environments. *IEEE T PARALL DISTR* **2016**, 27, 841-854.
4. Wu, S.; Chen, H.; Di, S.; Zhou, B.; Xie, Z.; Jin, H.; Shi, X. Synchronization-Aware Scheduling for Virtual Clusters in Cloud. *IEEE T PARALL DISTR* **2015**, 26, 2890-2902.
5. Greenberg, A.; Hamilton, J.; Maltz, D.A.; Patel, P. The cost of a cloud: research problems in data center networks. *COMPUT COMMUN REV* **2008**, 39, 68-73.
6. Liao, X.; Jin, H.; Liu, Y.; Ni, L.M.; Deng, D. AnySee: Peer-to-Peer Live Streaming. In Proceedings of the IEEE International Conference on Computer Communications, Barcelona, Spain, pp. 1-10, 23-29 April 2006.
7. Kim, H.; Feamster, N.; Improving network management with software defined networking. *IEEE COMMUN MAG* **2013**, 51, 114-119.
8. Monsanto, C.; Reich, J.; Foster, N.; Rexford, J.; Walker, D. Composing software defined networks. In Proceedings of the USENIX Symposium on Networked Systems Design and Implementation, Lombard, USA, pp. 1-13, 2-5April 2013.
9. Jin, H.; When Data Grows Big. *COMPUTER* **2014**, 12, 8.
10. Wang, M.; Meng, X.; Zhang, L. Consolidating virtual machines with dynamic bandwidth demand in data centers. In Proceedings of the IEEE International Conference on Computer Communications, Shanghai, China, pp. 71-75, 10-15 April 2011.
11. Piao, J.T.; Yan, J. A network-aware virtual machine placement and migration approach in cloud computing. In Proceedings of the International Conference on Grid and Cooperative Computing, Nanjing, China, pp. 87-92, 1-5 November 2010.
12. Meng, X.; Pappas, V.; Zhang, L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In Proceedings of the IEEE International Conference on Computer Communications, San Diego, USA, pp. 1-9, 15-19 March 2010.
13. Wang, S.H.; Huang, P.P.W; Wen, C.H.P.; Wang, L.C.; EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined datacenter networks. In Proceedings of the IEEE International Conference on Information Networking, Phuket, Thailand, pp. 220-225, 10-12 February 2014.
14. Bobroff, N.; Kochut A.; Beaty, K.; Dynamic placement of virtual machines for managing SLA violations, In Proceedings of 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, pp. 119-128, 21-25 May 2007.
15. Gong, Z.; Gu, X.; Pac: Pattern-driven application consolidation for efficient cloud computing. In Proceedings of IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, Miami, USA, pp. 24-33, 17-19 August, 2010.
16. Calcavecchia, N.M.; Biran O.; Hadad, E.; Moatti, Y.; VM placement strategies for cloud scenarios. In Proceedings of IEEE International Conference on Cloud Computing, Honolulu, USA, pp. 852-859, 24-29 June 2012.
17. Katta, N.P.; Rexford J.; Walker, D.; Incremental consistent updates. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, Hongkong, pp. 49-54, 16 August 2013.
18. Foster, N.; Guha, A.; Reitblatt, M.; Story, A.; Freedman, M. J.; Katta, N. P.; Mosanto, C.; Reich, J.; Rexford, J.; Schlesinger, C.; Walker, D.; Harrison, R. Languages for software-defined networks. *IEEE COMMUN MAG* **2013**, 51, 128-134.
19. Banikazemi, M.; Olshefski, D.; Shaikh, A.; Tracey, J.; Wang, G. Meridian: an SDN platform for cloud network services. *IEEE COMMUN MAG* **2013**, 51, 2, 120-127.
20. Sherry, J.; Hasan, S.; Scott, C.; Krishnamurthy, A.; Ratnasamy, S.; Sekar V. Making middleboxes someone else's problem: network processing as a cloud service. *COMPUT COMMUN REV* **2012**, 42, 13-24.
21. Jin, H.; Abbasi, A.A.; Song, W. Pathfinder: Application-aware distributed path computation in clouds. *INT J PARALLEL PROG* 2016, 1-12.

doi:10.20944/preprints201702.0074.v1

22. Binz, T.; Breiter, G.; Leyman, F.; Spatzier, T. Portable cloud services using tosca. *IEEE INTERNET COMPUT* **2012**, 16, 80-85.
23. Mahindru, R.; Sarkar R.; Viswanathan, M. Software defined unified monitoring and management of clouds. *IBM J RES DEV* **2014**, 58, 1-12.
24. Chang, D.; Xu, G.; Hu, L.; Yang, K. A network-aware virtual machine placement algorithm in mobile cloud computing environment. In Proceedings of IEEE Wireless Communications and Networking Conference Workshops, Shanghai, China, pp. 117-122, 7-10 April 2013.
25. Zadeh, L. A.; Fuzzy sets. *INFORM CONTROL* **1965**, 8, 338-353.
26. Saaty, T.L; What is the analytic hierarchy process? *Mathematical models for decision support* **1988**, 109-121.
27. Bonald T.; Virtamo, J.; A recursive formula for multirate systems with elastic traffic. *IEEE COMMUN LETT* **2005**, 9, 753-755.
28. Kaufman, J.S.; Rege, K. M.; Blocking in a shared resource environment with batched Poisson arrival processes. *PERFORM EVALUATION* **1996**, 24, 249-263.
29. Cherkasova, L.; Gupta D.; Vahdat, A. Comparison of the three CPU schedulers in Xen. *PERF EVAL REV* **2007**, 35, 42-51.
30. Jiang, J.W.; Lan, T.; Ha, S.; Chen, M.; Chiang, M.; Joint VM placement and routing for data center traffic engineering. In Proceedings of the IEEE International Conference on Computer Communications, Orlando, USA, pp. 2876-2880, 25-30 March 2012.
31. Guo, C.; Lu, G.; Li, D.; Wu, H.; Zhang, X.; Shi, Y.; Tian, C.; Zhang, Y.; Wu, S. BCube: a high performance, server-centric network architecture for modular data centers. *COMPUT COMMUN REV* **2009**, 39, 63-74.