A Map of the Clouds: Virtual Network Mapping in Cloud Computing Data Centers

Khaled Alhazmi*, Mohamed Abu Sharkh*, Daehyun Ban† and Abdallah Shami*
*Department of Electrical and Computer Engineering, Western University
*London, Canada. Email: {kalhazmi, mabusha, ashami2}@uwo.ca
† Software R&D Center, Samsung, South Korea

Abstract—Cloud computing success as a paradigm stems from the concepts its implementation was based on. Dynamic scalability and flexible pay-as-you-go pricing options attract business clients. However, efficient implementation of virtualization is the core that makes all these parts tick. To make that possible, cloud service providers need a thorough virtual network mapping system that allocates computational and network resources in a way that guarantees the quality of service conditions for clients while maximizing providers’ revenue and resource utilization. A comprehensive system is introduced to solve virtual network mapping for a set of connection requests sent by cloud clients. Connections are collected in time intervals called windows. Node mapping and link mapping are performed. Different windows size selection schemes are introduced and evaluated. Simulation results show that the Dynamic windows size algorithm achieves cloud service providers objectives in terms of served connections ratio, resource utilization and computational overhead.

Keywords—Virtual network mapping; Cloud data centers; Node and Link mapping; Window size decision.

I. INTRODUCTION

Cloud computing success as a paradigm stems from the concepts its implementation was based on. Providing the client with the ability to stretch or shrink the rented infrastructure dynamically on demand is a key advantage over the traditional plan and buy model. This feature combined with delivery over fast connections and the vast pay-as-you go pricing options explains the cloud popularity among business clients. However, the concept that made implementing all these features possible is in fact virtualization. Through virtualization, cloud service providers can offer computing power, storage, platforms, and services in a commodity based design without the clients needing to worry about low level implementation details. Clients can compute and connect without the overhead of resource management or network routing and control [1].

To achieve this objective, cloud service providers need efficient systems to comprehensively manage resource virtualization, allocation and scheduling in cloud data centers. Cloud data centers are home to thousands of servers that store, process and exchange clients’ data. Such systems will receive clients network and computational resource reservation requests and perform the mapping and scheduling of these requests. In the virtual network model adopted by many providers, clients can request Virtual Machines (VMs) of several types that differ in resource configuration. They can also request connections to be established between VMs or between a VM that resides in a public cloud and a client headquarter (private cloud). These communicating points constitute a virtual network where each client expects to maintain the agreed upon Quality of Service (QoS) conditions regardless of how many other clients share the data center resources. It is also important that dynamic network resource allocation to virtual machines is possible at all times. In case additional bandwidth is requested, it must be allocated to a running virtual machine instance to improve performance within a limited period of time. This virtual network mapping scenario poses questions like: What VM placement/mapping policy is used? What connection mapping and scheduling policy is used? How often are arriving requests processed and mapped? How are these requests prioritized?

In this work, we aim to tackle the problem of virtual network mapping in a cloud computing environment. Virtual network mapping in the context of cloud computing can be defined as finding the methodology to serve/handle requests arriving from clients continuously by forming virtual networks that are composed of multiple virtual machine instances running on servers in multiple geographically distant data centers. Virtual machine instances can be connected through virtual network links or edges that are mapped to physical (substrate) network paths.

Multiple variations of the virtual network mapping problem can be found in the literature, each with the goal of constructing the most efficient virtual network mapping (embedding or assignment in some sources) methodology.

In [2], a resource allocation model for cloud computing data centers is proposed. The proposed model consists of a network of private and public clouds. Requests arrive from clients either to reserve a VM on a public cloud, connect two VMs together or a VM to a private cloud. Multiple VM placement techniques and connection request scheduling techniques are proposed and...
evaluated. Computational and network resource are considered. Connection requests are processed individually without using any aggregation policy.

The authors of [3] introduce another virtual network embedding methodology. In their work, virtual network embedding is divided into node mapping and link mapping and the proposed virtual network embedding algorithm called VINEYard aims to coordinate node and link mapping phases in order to enhance the quality of embedding.

The authors also propose a window-based virtual network algorithm called the WiNE. Their simulation results show that collecting the virtual network requests and processing them periodically at the end of the time interval called window can be cost effective with resource utilization and can minimize the network controller processing overhead. As virtual network requests come in, the WiNE algorithm collects them in batches for a given time period (window) and analyzes the revenue potential of each request, giving higher priority to requests with the higher potential revenue. Every virtual network request has a specific time frame during which it is active. If the request lifetime is over before the end of the window, the request is ignored/dropped. The optimal window size analysis for a set of requests was not addressed.

In [4], the authors propose an embedding algorithm that accumulates multiple virtual network requests during a particular active window and processes them according to their specific requirements. In case some virtual network requests could not be addressed in a given time window, they are assigned to a queue and are considered again in the following. If a request has passed its maximum waiting time in the queue then it is dropped. For every active time window, the requests that arrived in it are addressed along with the requests that are waiting in the queue and prioritized based on the revenue they generate. Path splitting and migration features are also discussed.

In [5], optimal networked cloud mapping is formulated as a Mixed Integer Programming (MIP) problem, indicating objectives related to cost efficiency of the resource mapping procedure, while abiding by user requests and QoS conditions. A method is subsequently proposed for the efficient mapping of resource requests onto a shared substrate network connecting various islands of computing resources. A heuristic algorithm is adopted to address the problem. A Java based simulator is also introduced to evaluate the performance of the solution. A proof-of-concept realization of the proposed schema, mounted over the European future Internet test-bed FEDERICA, is implemented.

In [6], an elastic service oriented virtual network mapping methodology in cloud computing is presented. To complete virtual network request mapping onto a substrate network, two key steps are taken. These are virtual network mapping and resource allocation. Revenue is calculated based on realistic market prices.

The authors of [7] cite the difference in application bandwidth requirements and the high cost of dividing flows into different electronic channels as a motive to use a new design. Wavelength Division Multiplexing (WDM) systems are proposed in this paper as they "Employ Optical Add/Drop Multiplexers (OADMs) which allow a wavelength to either be dropped at a node or optically bypass the node’s electronics." The WDM mechanism proposed in the paper is traffic grooming. In traffic grooming, applications with less wavelength requirements can be aggregated on shared wavelength channels as a means to optimize network resources utilization. A sliding traffic scheduling model is also presented. Scheduling a request does not depend on its lifetime. The authors also propose a time window based algorithm where the network wavelength requests are divided into multiple time windows. Request with times that span over more than a time window have the option of being scheduled in alter window if the network does not have enough resources in earlier windows.

In this paper, we introduce a new virtual network mapping methodology for cloud computing data centers. Our work will involve the following:

- Provide a comprehensive methodology that covers computational and network resource requests by performing node mapping and link mapping.
- Aggregate the connection requests into virtual network requests and process these requests in a time window based manner.
- Investigate the effect of fixed and dynamic window sizes and the aggregation factor combined with virtual network mapping in a networked cloud Environment. The objective is to arrive at the optimal window size for a specific virtual network mapping problem.

The following sections are organized as follows: Detailed problem description is given in Section II. Section III presents the different time window selection techniques. The simulation environment and results are discussed in Section IV. Finally, Section V concludes the paper.

II. PROBLEM DESCRIPTION AND SOLUTION

When creating an efficient resource allocation methodology, it is critical that the resources possessed by the cloud infrastructure are properly modeled. It is equally important that the resource allocation system is continuously aware of the cloud infrastructure operational status in real time. In the scenario we are implementing, clients will reserve VMs for a specific amount of time (clients will be located in one node of the substrate network and the VM is hosted in one of the data centers). Connection requests are not known precisely in advance. The substrate network consists of data centers and client nodes. Each data center has a number of servers that can host multiple virtual machines without exceeding the server capacity. Multiple types of VMs that differ in configuration are available, and can be specified by clients. Clients request a connection between the VM and a client node. In a connection request, the client specifies the source, the destination, the requested start time, duration, requested capacity units (VM specs), and the required bandwidth. An example of a set of connection requests is shown in Table 1. As a first step of the solution, client requests will be combined based on a preset aggregation factor to form virtual network requests (VNRs). Serving these requests can be abstracted as a virtual network mapping problem where the nodes represent sources and destinations and the edges are virtual links between these nodes. Also, these VNRs will be assigned to time windows based on
the requested start time. Therefore, we can abstract a single window as a set of virtual network mapping requests during the time window. Besides, we will abstract a set of requests that span over more than one window as a set of virtual network mapping requests during these time windows. We call these requests spanning requests. After that, the system performs node and link mapping for the virtual network request on the substrate network.

Here the questions arise: How do we best choose window size in order to achieve performance and revenue objectives? In the following section we discuss the techniques we use to decide the window size. The window size selection will affect performance measures like request acceptance ratio, allocation (computational) overhead, resource utilization and revenue.

III. TECHNIQUES USED

A. Virtual Network Mapping

1) Node mapping:
To map a VNR on a physical substrate network, we first need to allocate the concerned VM to a server that has sufficient computational resources to support the VM. The node mapping algorithm used is a variation of the Node distance algorithm used in [2]. This insures a wide distribution of VMs and less connection request collision.

2) Link mapping:
Next we map the virtual link to a substrate path in order to connect the VM to the client headquarters. This link mapping is subject to satisfying the bandwidth constraints on the substrate path. The algorithm used is a greedy algorithm that allocates the request to the shortest path that has the requested amount of bandwidth available. This algorithm is efficient time-wise to minimize the computational overhead.

B. Time window selection techniques

1) Fixed window technique:
In this technique fixed window periods are defined and the connection requests are aggregated based on a predefined aggregation factor. For any given fixed window period, the connection requests are analyzed and those with the highest revenue potential are prioritized. The prioritized requests are aggregated and mapped to the substrate network and requests that cannot be mapped onto the substrate network are rejected. In addition, the maximum waiting time is also taken into account for that request. When a client sends a request for mapping they also specify a maximum allowed tardiness time period for which they are willing to wait. If a request could not be mapped within that time period it must be blocked.

2) Dynamic window technique:
In this technique, client requests are divided into sets where each set is assigned to a time window. Within a single dynamic window, the time requirements for different client requests could also overlap. In other words, two different requests accommodated within a single time window might request resources in the same period of time. A variation of this algorithm was used in the context of optical networks in [7].

Figure 1 and Figure 2 illustrate the results of running this algorithm on the sample of requests shown in Table 1. Figure 1 shows the client connection requests after dividing them into different window spaces and they all overlap in time. Figure 2 shows -in the form of an interval graph- how the connections are processed and implemented. In the interval graph, each row represents the connections that are connected with the nodes. For every row, if there is a connection is shown as “1” while a value of “0” is indicated for no connection. Figure 3 shows the virtual network requests that are abstracted in single windows and the process of virtual network mapping. Virtual network mapping involves node mapping and link mapping where connection requests are mapped to the substrate layer.

Once the window sizes have been decided, the requests are assigned to the respective windows and mapping is performed.

The steps taken are detailed in the algorithm above. After deciding the number of windows (NW) and each window size (WS), these values are used to assign connections to windows.

Next, connections that expire before the end of their assigned window are filtered and removed. Then, connections of each window are aggregated into virtual network requests (VNR) and sorted based on the potential generated revenue. The system checks if there is enough computational and network resources to map these requests. Finally, node mapping and link mapping are performed.
TABLE 1. AN EXAMPLE OF A SET OF CONNECTION REQUESTS

<table>
<thead>
<tr>
<th>Connection requests</th>
<th>Source</th>
<th>Destination</th>
<th>Start time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Client node 1</td>
<td>VM4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>Client node 1</td>
<td>VM1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>C3</td>
<td>Client node 2</td>
<td>VM3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>C4</td>
<td>Client node 3</td>
<td>VM2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>C5</td>
<td>Client node 2</td>
<td>VM4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>C6</td>
<td>Client node 2</td>
<td>VM4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>C7</td>
<td>Client node 4</td>
<td>VM3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>C8</td>
<td>Client node 4</td>
<td>VM1</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>C9</td>
<td>Client node 4</td>
<td>VM1</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>C10</td>
<td>Client node 3</td>
<td>VM2</td>
<td>11</td>
<td>4</td>
</tr>
</tbody>
</table>

I. PERFORMANCE EVALUATION

To evaluate the proposed techniques, a discrete event simulator was developed using C++.

With regards to the substrate network, the NFS network, as in Figure 4, is used in the simulation. Following similar setups to the ones in [2], the network is composed of 14 different nodes of which 3 are data center nodes and the rest are client nodes. 132 servers were used in the simulation with 44 servers in each data center. To connect the nodes in the substrate network, 21 links were used and 546 different paths were defined. Three alternate routing paths were defined for each pair of nodes.

The input contained data corresponding to 200 virtual machine instances.

For the connection requests coming from the clients, as in [3][4][5][8], their arrival rate were set according to the Poisson process varying from 1 to 5 steps with 0.5 per 100 time units. The connection request lifetime is exponentially distributed with an average of 1000 time units and the total number of connection requests is set to 3000. The maximum waiting time for each request was set to half of its lifetime. For every connection, there is a source, destination, start time, destination virtual machine specifications and requested bandwidth information. The source nodes are uniformly distributed with a client ID ranging from 0-10 and the destination nodes that represent a virtual machine number follow a uniform distribution of 1-200, given the fact that 200 virtual machine instances were used in the simulation.

With regards to node specifications, the CPU resources are uniformly distributed in the range of 50-100 and memory and storage are uniformly distributed in a range of 50-100. The available BW is set at 200 for all the links.

With regards to the requested virtual machine capacities, CPU resources are uniformly distributed for every request for a virtual machine instance in the range of 0-20. For memory and
storage, a uniform distribution is defined with a range of 0-200 and similarly with regards to BW, a uniform distribution within a range of 0-50 is defined.

II. RESULTS AND ANALYSIS:

Multiple metrics were measured during our experiments. Our main objective was to evaluate the window size decision effect on the different performance metrics. We have evaluated a no window scheme along with the dynamic window scheme based on the maximum independent set algorithm. In addition, we have evaluated fixed window scheme with multiple choices for the window size ranging from a small size (50 time units) up to a very large window size (2000 time units).

Figure 5 shows the average number of Virtual Network Requests VNRs per window. This number becomes very large, as the window size chosen grows larger. This has a direct effect on the ratio of served connections. As the number of the virtual network requests in a specific window grows, it becomes harder to find enough resources to establish a virtual network by mapping the requested nodes and links. We can also see that when using dynamic window sizes although it does not produce the lowest number of virtual network requests per window, it still yields a low number close to the numbers yielded by the fixed windows of very small size. Therefore, we notice as in Figure 6 that the served connections ratio for the dynamic window stands among the best. It was surpassed only by the results from fixed windows of a very small size (50 to 150 time units).

Figure 7 shows the average number of blocked connections. As we explained in the previous sections, all connections are aggregated with a factor of three and collected during the window and then processed at the end of the window. This means that if a connection's lifetime expires before their respective window is over, this connection will be blocked or rejected before the beginning of the mapping process regardless of the availability of the resources. The main factor that affects the ratio of blocked connections apart from connections' lifetime is the window size. As the figure shows, the number of blocked connections is very high for large window sizes while it stays in an acceptable level for small and dynamic window sizes. Again, the dynamic window size scheme performs acceptably for this metric.
Implementing virtualization in a smooth and cost effective way is crucial to the cloud service acceptance and market penetration. A challenge faced by cloud service providers is designing the resource allocation techniques that will tackle the problem of virtual network mapping. Clients send numerous requests to reserve computational and network resources and expect the Quality of service conditions they specify to be maintained through the request lifetime. One of the main features that define a virtual network mapping policy is the windows size selection scheme. Multiple window size selection schemes were evaluated and evaluated in this work. Dynamic window selection scheme was introduced in the context of virtual network mapping for cloud computing data center. After evaluating the possible window size selection techniques, simulation results showed that the Dynamic Windows Size scheme achieved all cloud service providers objectives in terms of served connection ratio, resource utilization and computational overhead. As a future step, we will further investigate the impact of the aggregation factor, different pricing options and the distributed cloud network topology on performance and revenue of the virtual network mapping requests.

III. CONCLUSION

We now examine the final metric which relates to the computational overhead expected by the network controller that performs the mapping process. The amount of computational overhead needed to map a certain amount of requests is affected by two factors in this problem. First, is the number of windows. This is based on the window size when using the fixed windows size scheme. As the windows become smaller, the number of windows needed is larger and the total computational overhead grows. Second, is the total number of VNRs in the problem. Figure 9 shows that for a specific amount of requested connections, using a small fixed window size tends to produce much higher number of VNRs than using the large fixed windows sizes and dynamic windows. This corresponds to a higher overhead which of course is not favorable by the cloud service provider. Comparing the performance metrics from acceptance, expired connections, and computational overhead we find that using the dynamic window size scheme can be the technique that shows good performances for all these metrics. Hence, using this scheme would be the best option for the cloud service providers when performing virtual network mapping.

![Graph](image-url)

**Fig. 8.** Results showing number of rejected VNRs for different window decision techniques

![Graph](image-url)

**Fig. 9.** Results showing number of virtual network requests for different window decision techniques

I. REFERENCES


