February 2015

Geography Aware Virtual Machine Migrations and Replications for Distributed Cloud Data Centers

Sakif Shahriar Pritom
The University of Western Ontario

Supervisor
Dr. Hanan Lutfiyya.
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

Recommended Citation

Pritom, Sakif Shahriar, "Geography Aware Virtual Machine Migrations and Replications for Distributed Cloud Data Centers" (2015). *Electronic Thesis and Dissertation Repository*. 2682.
https://ir.lib.uwo.ca/etd/2682

GEOGRAPHY AWARE VIRTUAL MACHINE MIGRATIONS AND REPLICATIONS FOR DISTRIBUTED CLOUD DATA CENTERS

(Thesis format: Monograph or Integrated Article)

by

Sakif Shahriar Pritom

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

Cloud computing provides access to computing resources for a fee. Client applications and services can be hosted in clouds. Cloud computing typically uses a network of data centers that are geographically dispersed. The distance between clients and applications is impacted by geographical distance. The geographical distribution of client requests can be random and difficult to predict. This suggests a need to reconsider the placement of services at run-time through migration. This thesis describes a framework based on software-defined networking (SDN) principles. It demonstrates algorithms that are periodically executed and determine candidate services to migrate and replicate as well as target data centers to migrate to and replicate to and an evaluation. The evaluation shows that effectiveness of the algorithms.

Keywords

Cloud Computing, Data Centers, Virtual Machine Migration, Virtual Machine Replication, Software Defined Networking.

Acknowledgments

First and foremost, I would like to express my heartiest gratitude to the Almighty, most gracious, and most merciful, for providing me with the ability and patience to accomplish this thesis successfully.

I would like to thank Dr. Hanan Lutfiyya for her constant guidance and supervision. Without her strong motivation, valuable inspiration and esteemed guidance and support, this work could not have been completed.

I would like to thank my wife, father, mother, and sister for their constant support and encouragement. I would also like to thank all my teachers, friends and relatives for their valuable advices and inspirations.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Appendices

Chapter 1

1 Introduction

1.1 Background

A data center is a collection of networked computational and storage resources. Cloud computing uses a network of data centers to provide access to computing resources for client applications on a pay-per-usage basis. Access to computing resources is enabled by virtualization technology, which allows access to computing resources by renting virtual machines (VMs). Applications and services can be hosted in data centers through the use of virtual machines [19].

Cloud computing offers infrastructure, platform and software as a service to its clients on a pay per usage basis. Many of the small, large and medium scale organizations are shifting their infrastructure into cloud in order to reduce operational and maintenance cost and overhead. Various web based service and applications are offered through Cloud Computing. Cloud computing has been widely adopted with Cisco predicting that by 2017 nearly two thirds of all workloads will be processed in the cloud [19].

Clients of the applications communicate with the services hosted in the VMs. Many applications have clients all over the world. An application is expected to provide faster access and transmission of data to its clients if it is geographically close to its clients, as some of the research work suggests that geographical distance has impact on quality of service (QoS) [7,8,12]. In order to provide a faster access and data transfer, applications which have clients all over the world should be hosted in a data center, which is on average close to its clients geographically.

1.2 Problem Statement

A cloud service provider that manages multiple and geographically distributed data centers needs to carefully consider the placement of VMs. The VMs should be placed in data centers that are close to its clients. The quality of service for clients is dependent on the geographical distance of the VM from client. The work described in Jain et al. [7], Lampe et al. [8], and Satyanarayanan et al. [12] show that geographical distances impacts latency. Satyanarayanan et al. [12] and Lampe et al. [8], illustrate that latency has an impact on the user experience for interactive and gaming applications. This suggests that for a client, the time it takes to receive a response to a request is impacted by the distance of the client from the data center, which is currently hosting the VM that the client is sending its request to. To some degree the work also suggests that the impact of latency increases with higher amounts of data being transferred.

However, the clients of application services hosted by the cloud provider are not always known a priori and the geographical distribution of requests from clients can be random and difficult to predict. Web based services and applications such as social networking, online gaming, virtual tour guide, mail clients etc. have clients from a variety of locations and countries. For such applications, the complete client list is not known as a priori. Clients may be anywhere in the world. An application may have worldwide popularity with a high number of clients but the distribution of clients from different parts of the world may not be even. For example, the US has the highest number of users in the world for both Facebook and Twitter. However, China has the largest population in the world and its population is more than four times of that in the US and yet China is not found in top ten Facebook or Twitter user countries by number of users as they use local social networks such as Qzone and Renren [22, 23, 24].

In recent years there has been a huge growth in mobile applications [48]. Mobile devices have limited storage capacity, slow processors and limited battery life. One approach for enabling users of mobile devices to be able to use resource-intensive applications is to have part of the application run on remote servers, which may be part of a cloud infrastructure. The application component that executes on the remote server typically has high computational needs and/or requires access to data [28]. These high computational needs are not easily available on the mobile device. Regardless of the network distance between the cloud infrastructure and the mobile device, the use of a remote service is well suited for mobile device applications with relatively little data to be transferred. However, long distances between a mobile device and remote services make this approach unsuitable for applications that require larger amounts of data to be transferred and/or have a high level of interactiveness with the user.

Popular mobile device platforms such as Android, iOS, Windows etc. are widely used all over the world and so a cloud based mobile application can have users from all over the world [29]. Many such applications are available in online application markets, e.g., Google Play Store, Apple App, store for installation. As an example, a cloud based encyclopedia or English to English dictionary can be downloaded and installed by anyone at any time, so the complete client list is not possible to retrieve a priori before launching the application in market. This implies that an initial placement may not be optimal over time since the clients and the distribution of clients will probably change.

There is a need to monitor communication patterns between the VMs and their corresponding clients in order to update the client list of the VMs. If needed VMs could be migrated to other data centers if it is determined that migrating the VM to some other data center will result in reduced time to access and transfer data. With an increase in workload and clients, a replica of the VM can be also beneficial if the VM owner is willing to pay for the extra VM. The new replica of the VM should be also placed in a data center that will reduce the access and data transfer time of its clients significantly.

1.3 Thesis Focus

There is a body of work that considers geographical distance in reducing latency and thus improves performance. For example, Alicherry et al. [2] minimizes the maximum distance between data centers hosting multiple dependent VMs. Little work considers the distance of VMs from its users.

This thesis focuses on decreasing the average distance of the clients from the VMs that the client is using through VM migration. Two new VM migration algorithms were developed that reduce time to data access and transfer for the clients of a VM by migrating the VM to a suitable data center that reduces the average geographical distance.

When warranted replication of a VM is used. Replicating a VM and hosting the replicated VM in the same data center that hosts the primary VM does not help to reduce the time to access and transfer data for the clients, as all the requests will be served from the same data center. We have also proposed a VM replication algorithm that creates a copy of the VM in a suitable data center. The replication algorithm has similar goals to the migration algorithms.

We exploited the programmable network architecture of Software Defined Networking (SDN) to propose a framework to deploy our algorithms in the network of connected data centers.

1.4 Thesis Outline

The thesis is organized as follows: Chapter 2 describes the related and relevant works about this area. Chapter 3 describes our proposed framework for deploying our proposed algorithms. Chapter 4 describes our proposed algorithms for VM migration and replication in order to reduce load-distance of the clients from the VM. Chapter 5 describes the experiments and presents the results of the experiments. Chapter 6 discusses conclusions and future work.

Chapter 2

2 Related Work

This thesis describes a novel approach to dynamically placing VMs geographically close to its clients. There is a body of work that considers geographical distance in reducing latency and thus improves performance. For example, Alicherry et al. [2] minimizes the maximum distance between data centers hosting multiple dependent VMs. Little work considers the distance of VMs from its users. The proposed approaches to address the problem stated in Chapter 1 are based on a Software Defined Networking (SDN) framework. This chapter describes data center organization and Software Defined Networking in Section 2.1 and 2.2. Section 2.3 describes innovative work that uses SDN to facilitate data centers and network management. Sections 2.4 and 2.5 illustrate some of the uses of VM migration and VM replication Section 2.6 describes some of the work that focuses on reducing latency. Section 2.7 discusses and summarizes the related work.

2.1 Background on Data Centers

A *data center* is a pool of connected computational, storage and network resources. The data center network has a pivotal role for the overall performance, efficiency and effectiveness of the data center as it connects the data center computing and storage resources to each other. A *cloud* refers to a set of interconnected data centers.

Data center networks are typically organized as layers where each layer is assigned a specific functionality. The layered model facilitates the management of the data center network. For example, troubleshooting is easier when the network is segmented.

A data center network use one of these layered models: three-tier network, fat tree network or DCell Network [30]. Three-tier data center networks are the most widely used [31]. In a a three-tier data center network, servers in racks are directly connected to a switch that are called either edge layer switches or Top of Rack (TOR) switches. The aggregation switches are responsible for interconnecting multiple edge layer switches together. Core layer switches are the root of the tree architecture and each core switch is connected to every aggregation switch to ensure fault tolerance and to prevent a single point of failure. A major disadvantage of the three-tier data center network is that it is extensively power hungry and it suffers from oversubscription of available bandwidth. [30]. Figure 1 depicts a three tier data center network.

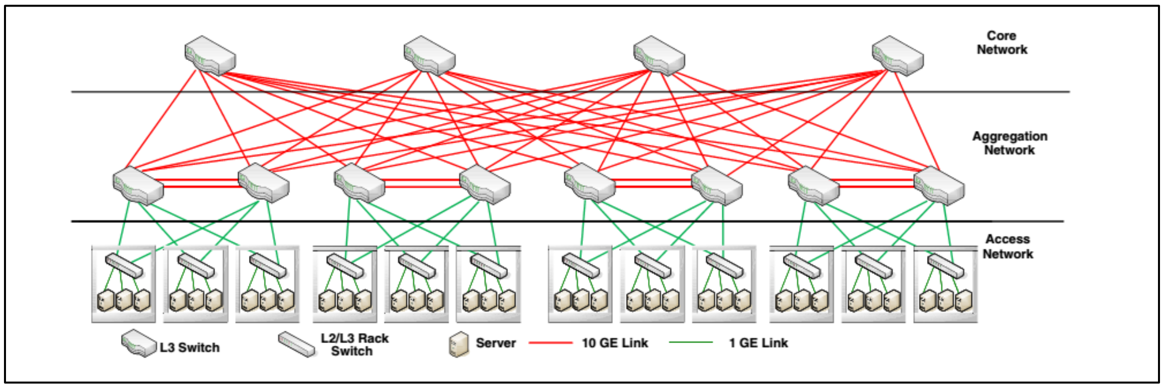


Figure 1: A three-tier data center network [30]

The fat tree architecture was proposed as a mechanism to increase end-to-end bandwidth with a lower cost and lower energy consumption [30]. The network structure is composed of *n* pods where each pod contains *n* servers and *n* switches. For each pod, switches are organized in aggregation and edge layers with *n*/2 switches in each layer. Every lower layer switch in the pod is connected to *n*/2 hosts in the pod and *n*/2 upper layer switches of the pod. There are $(n/2)^2$ core level switches, each connected to one aggregation layer switch from each of the pods [30]. Each aggregation switch is connected to a core layer switch Figure 2 depicts a fat tree data center network for *n*=4.

In a DCell data center network, the data center is organized as a hierarchy of cells or pods. It is regarded as a highly scalable as well as complicated structure [30]. The building block of the system is called $DCell_0$. A $DCell_0$ unit consists of *n* servers and a mini network switch [30]. Each server is connected to the switch of its cell and a server of another cell of the same level. Higher levels of cells are built by connecting multiple lower level cells. Each $DCell_{n-1}$ is connected to all other $DCell_{n-1}$ within the same $DCell_n$. As an example, $DCell_1$ is comprised of multiple $DCell_0$ units where each $DCell_0$ unit is connected to all of the other $DCell_0$ units within the same $DCell_1$. Figure 3 illustrates a DCell network where $DCell_0$ has two servers and a switch.

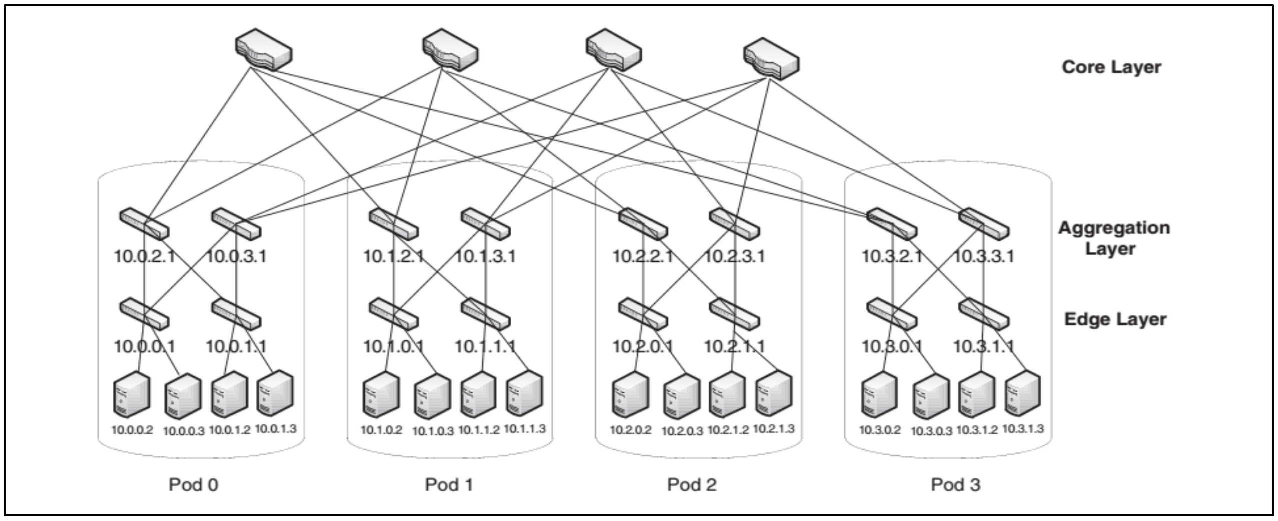


Figure 2: Fat tree based data center network [30]

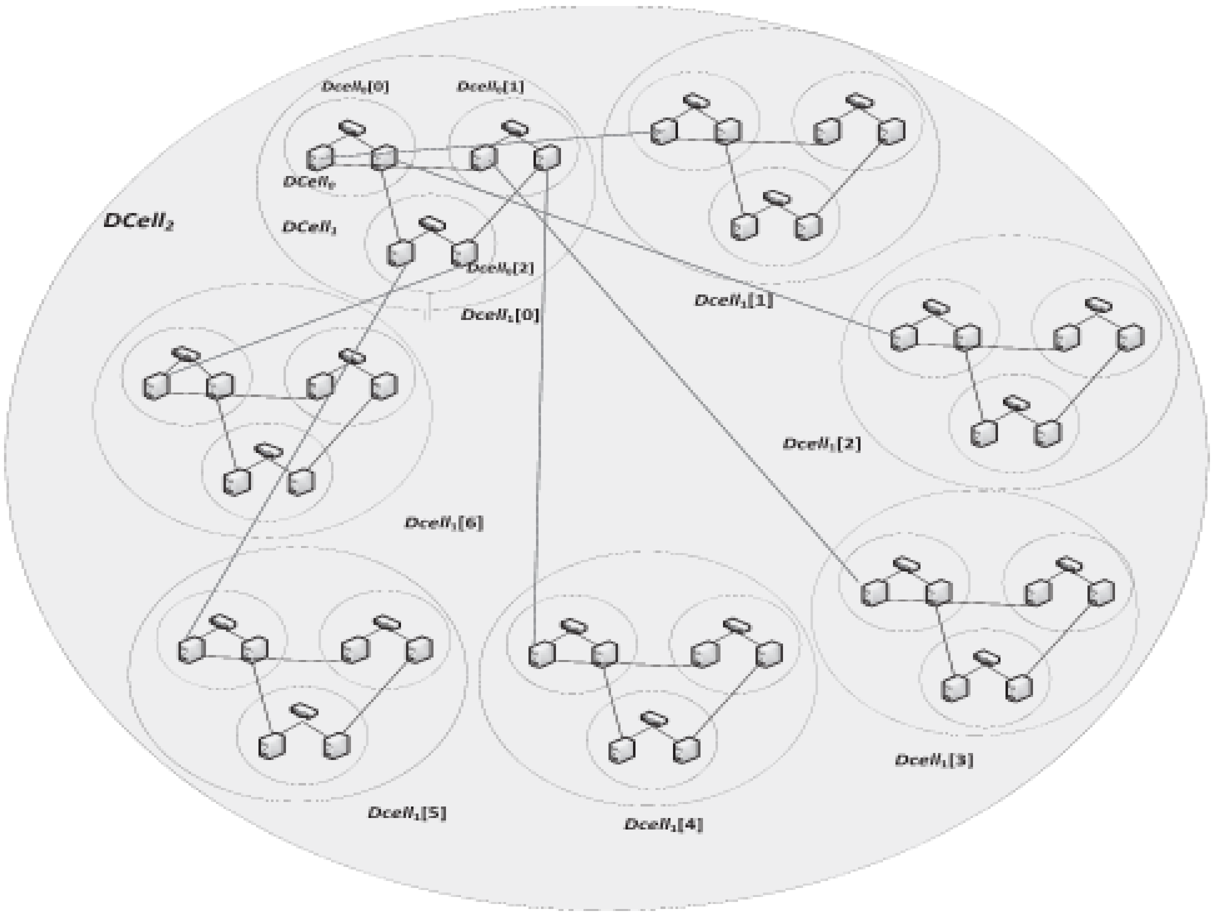


Figure 3: Fat DCell data center network [30]

Data centers are expected to provide sufficient resources to client applications while minimizing power consumption. Resource allocation algorithms for data centers should consider policies for allocating virtual machines (VMs), the physical distribution of servers, adequate networking bandwidth and the physical location of data.

2.2 Background on Software Defined Networking (SDN)

Computer networks typically have these network devices: (i) Routers forward data packets between computer networks; and (ii) Switches that link network segments. Each network device has a control plane that makes decisions on the next communication link to be used to transfer data, and a data plane that is responsible for forwarding the packet according to the control logic. Traditional network devices have the control plane and data plane coupled to each other. Software Defined Networking is a networking architecture where control logic is physically decoupled from the data plane.

Software Defined Networking (SDN) allows the network administrator to handle lower level network services and functionalities through abstraction [4]. SDN requires special network elements and an SDN controller to be deployed in the network. The role of an SDN controller is to provide the intelligence of the network where other network elements are used only for packet forwarding. An SDN controller has control over the data plane or the forwarding elements. SDN controllers can be physically distributed or centralized but logically centralized. With this physical separation of the control plane from the data plane, SDN aims to control network devices programmatically. A comparison of control and data elements between traditional networking and SDN is depicted in figure 4.

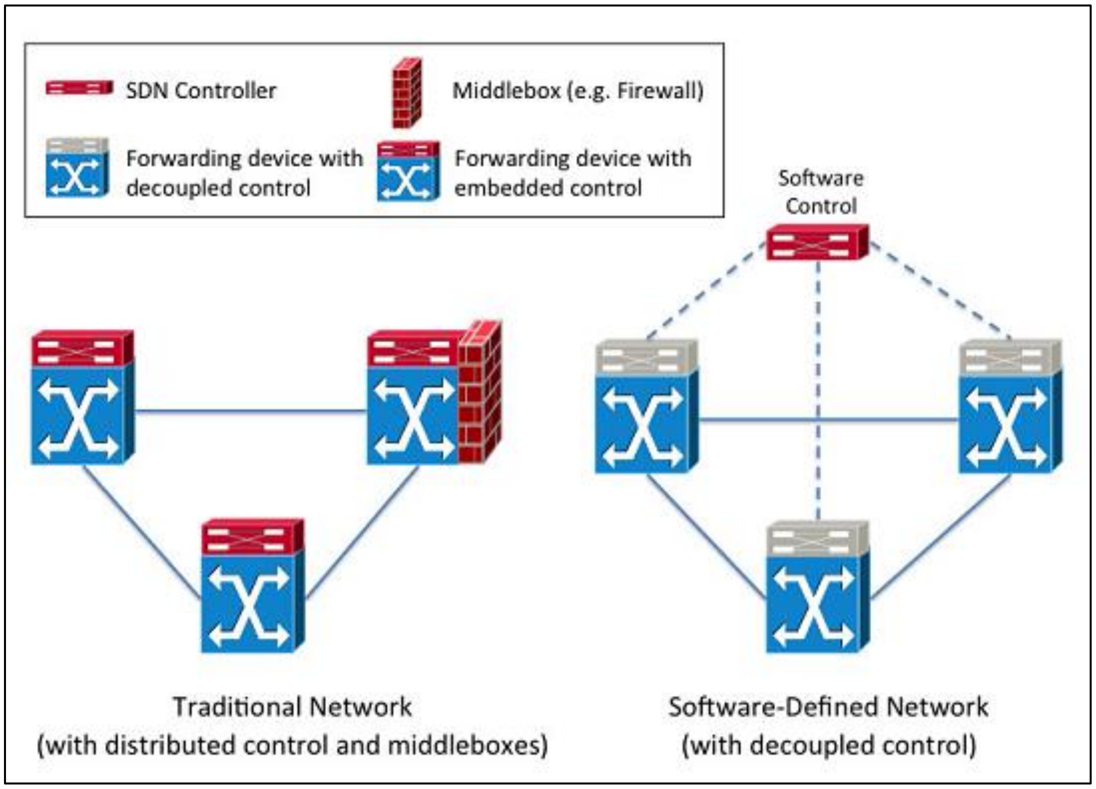

SDN Controller
Middlebox (e.g. Firewall)
Forwarding device with decoupled control
Forwarding device with embedded control
Software Control
Traditional Network (with distributed control and middleboxes)
Software-Defined Network (with decoupled control)

Figure 4: Illustration of control and data elements in traditional networks and Software Defined Networks [4]

Network administrators provide the network policies, e.g. maximum throughput or minimum delay, via the controller through a programmatic approach. The controller communicates with underlying network elements. Controllers need special protocols to communicate with underlying network elements and the network elements must have the mechanism to receive instructions from the controller to act accordingly. Two of the protocols found in the literature are the Openflow [33] and ForCES (Forwarding and Control Element Separation) [32].

The controller can be compared to an operating system or network operating system. Applications are written on top of the controller and communicate with the controller by a specific API. Network administrators are users of those applications and enforce management policies through a Graphical User Interface. The controller has modules that receive upper level instructions that are converted into Openflow commands and hence communication takes place between controller and the forwarding network elements.

The deployment of SDN can facilitate computer networking in numerous ways by allowing a wide range of services and applications to be deployed. The controller is able to communicate with a set of network elements and hence it can monitor the network status and receive information on network traffic, link or device failure, inclusion and exclusion of devices etc. Moreover, an SDN controller can monitor the network on a per flow basis. This information can be exploited to deploy new and innovative services such as power optimization of the network or a more efficient load balancing etc.

2.2.1 Architecture of SDN

An SDN architecture has three basic layers: Application layer, Control Layer and Infrastructure Layer. The network is programmed through the applications in the Application layer, where applications use an SDN controller (Control Layer) to configure network devices (Infrastructure Layer). Thus, the Control Layer has to communicate with the Application Layer as well as the Infrastructure Layer. Communication between the Application Layer and Control Layer is known as Northbound communication while Control Layer to Infrastructure Layer communication is known as Southbound communication. Figure 5 depicts the SDN architecture.

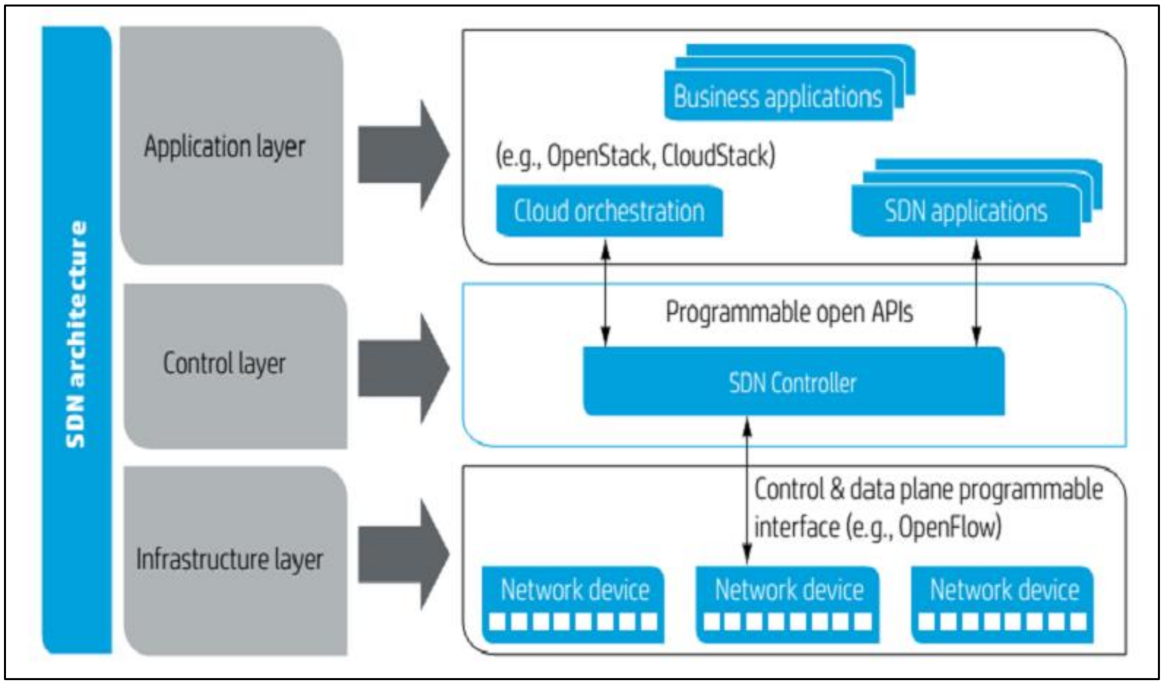


Figure 5: SDN Architecture [34]

The Application layer is the topmost layer of this three layered architecture. The application layer of SDN consists of various network applications designed for both various services or to enforce management policies. Applications use SDN controllers to achieve desired network behavior. Applications receive network level information and statistics such as packet arrival rate, link or device failure etc., specify their requirements programmatically to the SDN controller and receive feedback on its desired action. An application can be written to enforce network management policies e.g., balance the load on all the servers in the data center. The range of possible applications is huge with the potential for a great deal of innovation in networking [4].

The Control layer consists of SDN Controllers. The SDN controller translates requirements of network applications (e.g. a network application may request a shortest available path to be used) to configure the forwarding devices. The controller needs special protocols to communicate with the forwarding devices. The controller is compared to a Network Operating System as its functionalities mimics the functionality of an operating system. The infrastructure layer consists of the network devices able to receive and process the instructions from an SDN controller. To support SDN architecture, specialized hardware and devices are required which are dynamically programmable through an SDN controller. These devices do not have a control plane as the control part is consolidated in the SDN controller and act only as forwarding devices.

2.3 Related Work for Data Center and Network Management through SDN

There has been considerable research that focuses on supporting the functionality of data center management and network management through SDN. This section will describe some of the research work conducted in order to facilitate network management, data centers and clouds using SDN.

2.3.1 Load Balancing through SDN

Many online services, e.g. webserver, search engine, social networking, replicate services across multiple physical servers. The use of replication allows for better reliability and greater throughput. A front-end load balancer directs each client request to one of the servers with the goal of balancing traffic across the servers. Dedicated load balancers are expensive, become overloaded since all requests go through the load balancer and are a single point of failure. SDN and Openflow can be used for load balancing for both structured networks such as data centers [16] and unstructured networks such as enterprise and campus networks [35].

OpenFlow provides an approach to load balancing based on packet-handling rules installed by an SDN controller. However, installing separate rules for each connection may lead to a huge number of rules in switches and may result in a heavy load for the switches and the controller. Openflow does not support hashing for load balancing and thus Wang et. al. [16] proposed to exploit the use of wildcard rules for a scalable solution for handling large volumes of traffic and proposed an algorithm for load balancing which automatically adjusts to changes in load balancing policies.

The proposed Openflow load balancer described in Wang et al. [16] proactively installs wildcard rules in the switches to direct requests for large groups of clients without involving the controller. Dynamic redistribution of traffic only requires the installation of new rules in switches.

Handigol et. al. [35] proposed the “Plug-n-Serve”, a load balancing scheme for unstructured network such as enterprise and campus network. “Plug-n-Serve” is based on customized flow routing and attempts to minimize the response time through effective load balancing across the servers. Plug-n-Serve considers loads on servers and available routing paths in order to minimize response time of servers. A prototype of the proposed load balancer was tested in the Computer Science building at Stanford University. The test includes the scenarios of inclusion/removal of servers, varying the traffic load and changing the load balancing algorithm.

2.3.2 Intra Data Center Management through SDN

In a data center, almost 70% of the power consumption is due to servers and about 10% of the power is consumed for energy conversion and data center maintenance. However, network elements (e.g., switches, routers, links) consume almost 20% of the power [37]. Thus power saving practices that involve network devices could result in notable power savings. The general practice with the network resources is to always keep devices on. However network traffic is dynamic and changes over months, weeks, days and even hours. Power consumption of network devices are not energy proportional which means that the network devices consume energy at a rate that does not vary much even as network traffic load varies greatly [37]. Thus significant energy savings cannot be done by changing network demand.

ElasticTree [37] provides a mechanism that dynamically adjusts a set of active network elements, e.g., links and switches, such that the current network load can be satisfied with minimum power consumption. This is defined as the *minimum powered network subnet*. The minimum power network subset is the set of network elements which must be kept activated in order to satisfy network performance and fault tolerance goals while keeping the power consumption minimal. ElasticTree is based on three logical modules as follows: optimizer, routing, and power control. The average network power savings by applying ElasticTree to the data center network is 25% to 40%, which implies up to 8% of overall consumed energy of data center.

Another approach is seen with Hedera [38]. If more than one long lived flow passes through same output port, there will be a drastic drop in bisection bandwidth[1] utilization as both the flows will be transmitting through the same link simultaneously. However, intelligent flow placement could prevent the collision of flows in the same link. Hedera introduces a new dynamic flow scheduling algorithm which is scalable and maximizes the utilization of bisection bandwidth [38]. Hedera collects flow information from switches and computes non-conflicting paths for all flows exploiting the redundant paths.

Hedera is based on three basic steps. (1) It detects large flows (a flow that consists of a large number of packets) at the TOR switches. (2) It estimates the natural demand[2] of large flows. (3) It uses placement algorithms to compute non-overlapping paths for them, and then these paths are installed on the Openflow switches through an SDN controller. To distribute the traffic in as many as possible core switches, a packet's path is non-deterministic from edge to core switch, and is deterministic returning from the core switches to its destination edge switch.

2.3.3 SDN based Data Center Management Middleware

Researchers have proposed middleware that needed to support dynamic resource allocation. Several examples are Meridan [3], LiveCloud [39] and NOX [14].

The proposed middleware allow cloud service providers to provide complex services where customers are provided with network layer constructs and can configure the virtual topology for deployment of their applications. Deployment of such complex applications and dealing with numerous resources (e.g. computing, storage, network etc.) makes data center management difficult and challenging, as well as requires sophisticated network

---

[1] The bisection bandwidth of a network is the minimum bandwidth along all possible bisections of the network.

[2] A TCP flow's natural demand is defined as the rate it would grow to in a fully non-blocking network, such that eventually it becomes limited by either the sender or receiver Network Interface Card speed.

and resource management architecture. Any management scheme should address issues related to visibility, orchestration and provision. Visibility refers to the continuous monitoring of network states, resources and quality of service. Orchestration refers to Quality of Service (QoS) aware resource allocation and coordination. Traditional solutions may suffer due to limited support for network resources in orchestration as most of the cloud management lack in integration of network resources. SDN based network and resource management scheme can provide more programmability and fine grained monitoring and control over the network, computing and storage resources.

2.3.4 Policy Enforcement in Networks through SDN

Network applications can have various Quality of Service (QoS) requirements such as latency, throughput, jitter etc. Service Level Agreements (SLA) is established to ensure QoS requirements of such applications are maintained. Policy based network management is a network management framework where a network manager uses Service Level Agreements and service level objectives to deduce network level policies which afterwards are mapped into device level primitives. Traditional policy based network management approaches require installation of specialized software or hardware component in the network. Bari et. al. [40] proposed PolicyCop which is an autonomic QoS policy enforcement framework exploiting the network programmability of SDN.

PolicyCop consists of a Data Plane, Control Plane and Management Plane. The underlying network is required to be built from OpenFlow switches. PolicyCop was used for reacting to link failures and throughput requirements to being satisfied at runtime. Experiments show that PolicyCop is able to re-route the flows to an alternate route in case of a link failure. When two or more flows with specific throughput requirements collide in a single link, it can re-route the flows to some alternate route in order to restore the required throughput.

2.4 Virtual Machine Migration

VM migration is the strategy of transferring a VM from one host machine to another. Some of the works uses VM migration to optimize power efficiency, minimize traffic between dependent VMs or to prevent service level agreement (SLA) violation etc.

Shrivastava et. al. [15] proposed a VM migration strategy in data centers that considers the network topology and the dependencies between VMs in order to decrease network traffic between VMs. The work focuses on using VM migration to reduce network traffic between VMs rather than considering multiple clients, multiple data centers and client data access and transfer time.

Foster et. al. [6] proposed a method for dynamically switching between management strategies where they used VM migration in order to prevent Service Level Agreement (SLA) violation as well as keeping the power consumption of the data center minimum. The resource demand of a VM is dynamic and changes with time, which indicates that static allocation of VMs over servers, may cause the server to be over utilized or underutilized at different time stamp. This work focuses on migration VMs from underutilized hosts in order shut down the host in order to save power. The proposed strategy also migrate VMs from over loaded hosts to prevent SLA violation.

Hirofuchi et. al. [42] proposed a live migration of VMs over wide area network. Live migration of virtual machines can affect the I/O performance of the running service. This works aims to minimize the impact on I/O performance while migrating the VM over a wide area network.

2.5 Virtual Machine Replication

VM replication is the process of creating a replica of a VM, usually done to distribute load across multiple VMs to present performance issues due to overloaded VMs as well as preventing a single point of failure. VM replication can improve fault tolerance, reliability as well as accessibility of a service. The workload of processing client requests is shared among the VMs.

Keller et. al. [43] suggests that when the combined resource needs of the VMs of a physical host exceeds the resource availability (stressed situation), VM migration and replication can be used to improve the situation. The conducted experiments suggest that when a VM is in a stressed situation, VM replication is preferred over VM migration and VM migration is preferred over no action.

Goudarzi et. al. [44] proposed a possible use of VM replication for energy efficiency in cloud computing. Creating multiple copies of a VM and distributing workloads between these VMs helps in reducing resource requirements of each copy of a VM. This work suggests that such VMs with low resource requirements can be managed to utilize the servers more efficiently in order to reduce energy consumption.

Yuyang Du et. al. [46] proposed a VM replication strategy to improve reliability of a running VM. The work proposes to frequently replicate the state of a primary VM and thus the replicated VM can take over for any failure or crash of the primary VM.

2.6 Reducing Latency to Virtual Machines

Alicherry et al. [2] proposed their algorithm, which aims to minimize the maximum distance between data centers hosting multiple dependent VMs. Their algorithm considers the inter data center distance between distributed data centers and aims to minimize the distance between any two data center for dependent VM placement in multiple data center. This work does not consider the location of clients. This approach is aimed to reduce inter VM communication time, but overlooks the communication time between clients and corresponding VMs.

Sharkh et. al. [1] proposed two heuristic algorithms to reduce the average tardiness of a client’s request of connecting a VM to another VM or a client node.   The work described in [1] attempts to schedule resources (CPU, memory, storage) minimizing the average tardiness of the request as well as minimizing resource blocking. The work proposes the possible use of an SDN controller in order to keep track of the resources as well as handling newly arriving requests. However, multiple clients and the geographical location of the client were not considered.

Piao et. al. [9] assumed that data accessed by VMs are stored in distributed federated storage. This work also assumes that different hosts in a single data center have different access times for different storage units. Based on the requirements of accessing storage of the applications running on VMs, they proposed a VM placement and a VM migration algorithm in order to reduce time to access storage. It does not consider multiple clients nor access or data transfer time of the clients.

None of the work considers multiple clients from different geographical location communicating with the same VM.

2.7 Discussion

Various web based services and applications are shifting into cloud data centers, which have multiple clients from different geographical regions. Some of the work in the literature considers the distance between dependent VMs, as well as distance between VMs and data storage. However, after the computation is done by the VM, the output is eventually sent back to the client. As a result, the distance between the clients and the VM is expected to play a vital role in order to provide a faster service, especially for data intensive applications. To the best of our knowledge, none of the related work considers the distribution of requests from clients in making migration or replication decisions.

Chapter 3

3 Proposed System Framework

This chapter describes a framework for a system that identifies potential candidate VMs to migrate or replicate to target data centers. This work assumes that a geographic area is partitioned into N geographical regions denoted by R_0, …, R_{n-1}, where region R_i is served by data center DC_i. Each request is classified into a region based on the origin of the request. A request from a client entering the data center results in a flow, where a flow is defined as a sequence of packets traversing a network that share a set of header field values. The classification is used as input to an algorithm that determines if a VM should be migrated or replicated to a different data center. Figure 6 depicts the framework. The framework assumes the use of Software-Defined Networking (SDN). SDN can be used to develop innovative management system e.g., [3,10,14].

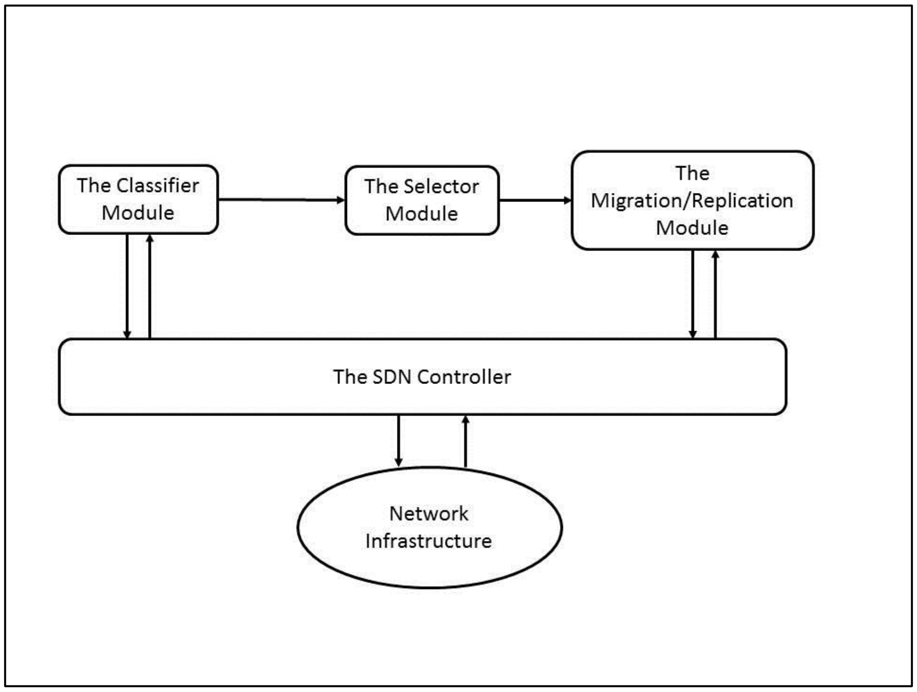


Figure 6: Proposed System Framework

Our proposed framework comprised of an SDN controller, the classifier module, the selector module, the migration module and the replication module. Either the migration module or replication module is used to support migration or replication respectively along with the rest of the modules. Section 3.1 describes the functionality of an SDN controller in the framework. Section 3.2 describes the classifier module. Section 3.3 outlines the selector module. Section 3.4 and 3.5 describes the migration module and the replication module respectively.

3.1 SDN Controller

Software Defined Networking (SDN) is a network architecture where the control plane is physically decoupled from the data plane and thus the network is directly programmable through a controller, known as an SDN controller [4]. An SDN controller controls a set of network devices through a well-defined API. Routers and switches have flow tables with entries that specify conditions (rules) that if true result in defined actions. Conditions can use wildcards on source and destination IP addresses to reduce the number of flow entries [16].

The proposed framework assumes that there is an SDN controller associated with each data center. A boundary router receives all flows to the data center. For the first packet of the first flow that arrives from a region, the packet is forwarded to the controller since there is no flow rule installed. A flow entry for that region and destination is sent to the boundary router by the SDN controller. A flow entry includes statistics related to the packets that are received. This allows for the flow of a load to be determined. This does not scale for large systems as noted by Wang et al. [16]. Data centers would require a huge flow table. Wang et al. proposes the use of a wildcard to reduce the number of entries. However, this results in loss of flow information. Wette [17] proposes an approach that requires an extension of OpenFlow to address this issue.

An SDN controller can update forwarding tables of switches and routers and redirect the flows to support live migration of VMs [16]. If a migration is triggered, the SDN controller of the data center that currently hosts the VM will alter the flow tables and redirect the flows to the target data center in order to prevent any loss in communication between clients and the VM, where the SDN controller of the target data center will install corresponding flow entries for the incoming VM. If a replication is triggered, the SDN controller of the target data center will install corresponding flow entries in the switches and routers of the target data center to support routing to the newly created VM.

3.2 The Classifier Module

This module is responsible for classifying each flow to a geographical region. One possible approach to identifying a region is to use the source IP address of the flow. An SDN controller can be used to gather flow information as described in [17].

The Classifier module collects information on the flows and the origin[3] and the destination[4] IP of each flow from the boundary router of the data center through the SDN controller. It maintains a separate list containing the origin, load and region of each flow for every VM in the data center in order to classify each flow into a region. To identify the region by source/destination IP of the flow, a pre-existing geographical vicinity list of countries and corresponding regions that states the rule for classification in a predefined manner based on geographical vicinity is maintained. In such a list, a matching rule is provided of IP addresses for region classification and thus region is determined locally avoiding the network overhead. Assume that, the cloud service provider has three data centers in Canada, India and Australia. This implies three regions. Flows originating from any country will by classified into the region which is geographically nearest to the origin using the pre-existing geographical vicinity list.

The pre-existing vicinity list will contain the list of countries and corresponding regions based on geographical vicinity. Upon arrival of a flow, the country of origin of the flow can be obtained by from the IP address of the flow [47]. The vicinity list should have possible corresponding entries for each country that results in a specific region. Thus the region of a flow can be obtained using the IP address.

---

[3] By origin of a flow, we refer to the source IP address of a flow which is destined for a VM in the data center.

[4] By destination of a flow, we refer to the destination IP address of a flow which is originated from a VM in the data center.

3.3 The Selector Module

The selector module is responsible for identifying the potential candidate VMs for migration using the classification provided by the classifier module. The module can also suggest a target data center for creating a replica of the VM at the discretion of the VM owner (Another replica of the VM may improve response time but incurs additional cost). Chapter 4 presents several algorithms that use the *load-distance* metric, which is calculated for each flow. The load-distance for a flow from a VM_i is the product of the distance between VM_i and the requestor that initiated the flow and the load. If the set of flows associated with VM_i is f_i then f_{ij} represents a single flow. This load-distance metric can be calculated for any flow, f_{ij}, and is calculated as product of the load and the distance between the VM and origin of the flow. Assume there are M flows that communicate with a VM and the distances are d_{i0}, …. $d_{i,M-1}$ and the loads are l_{i0} … $l_{i,M-1}$. The *average load-distance* is calculated as follows:

$$\sum_{j=0}^{j=M-1} (d_{ij} * l_{ij}) \Big/ M$$

The product is referred to as the weighted distance.

As noted earlier the geographical distance impacts latency, which may impact the user experience. The distance in itself is not sufficient since it does not take into account the number of packets sent, which we use to represent the load of a flow. Distance can be measured as the hop count of the source IP of a flow from the data center or the turn-around time, and the number of packets is information that can be gathered from a switch. The Selector module invokes the migration module with the information of potential candidate VMs and their corresponding target data centers to carry out a live migration of the candidate VMs in corresponding target data centers. The VM can be replicated given that the owner will pay the additional cost of the new VM (which is a clone or replica of the primary VM). In such cases, the selector module will invoke the Replication module to create a replica of the VM in the target data center.

3.4 The Migration Module

The migration module receives the set of candidate VMs for migration. For each candidate VM, the migration module invokes a migration request to the resource managers of the target data center. The resource managers of the target data center checks the resource availability of the data center to determine whether it can support the migration. If the target data center acknowledges that it has sufficient resources, and thus can accommodate the candidate VM, the migration module initiates the migration through the use of the SDN controller that programmatically removes the matching forwarding entries of the switches for migrating VMs and redirects the flows to the appropriate data center. Both data centers update their resource status after each migration takes place. This requires inter SDN controller communication as shown in Figure 7. However, a migration will not be possible if the target data center does not have sufficient computing, storage or network resources for hosting the migrating VM.

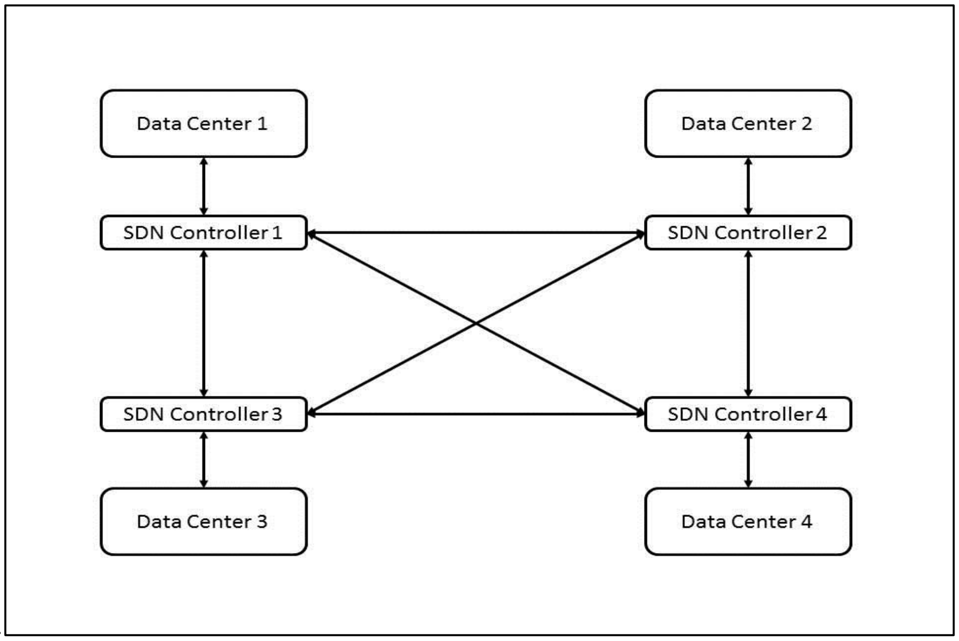


Figure 7: Interconnected SDN controllers of data centers

3.5 The Replication Module

The Replication module is responsible for creating a replica of the candidate VM in a target data center. The replication module receives the set of candidate VMs along with corresponding target data centers for replication. For each candidate VM, the replication module invokes a replication request to the resource managers of the target data centers The resource managers of the target data center checks the resource availability of the data center to determine whether it can host a copy of the candidate VM. If the target data center acknowledges that it has sufficient resources, and thus can accommodate a replica of the candidate VM, the replication module of the target data center creates a replica of the candidate VM. Target data centers update their resource status after each replication takes place. Like the migration module, the replication module also requires inter SDN controller communication as prior to the replication taking place, there is a need to know that whether the target data center has the required resources to host a copy of the candidate VM. A replication is not be possible if the target data center does not have sufficient computing, storage or network resources for hosting a replica of the candidate VM.

Chapter 4

4 Proposed Algorithms

This chapter presents two algorithms for selecting VMs to migrate and target data centers and one algorithm for replicating a VM. All three algorithms are executed periodically. Sections 4.1 and 4.2 describe the migration algorithms, Section 4.3 describes the replication algorithm and Section 4.4 discusses and summarizes the proposed algorithms.

4.1 Selector Algorithm 1 for Migration

This algorithm considers a VM to be a candidate VM if there is a prediction that moving the VM to another data center would improve the client experience by moving the VM to data center in a region that is considered, on average, closer to its clients.

This algorithm takes as input the set of virtual machines and set of flows F, that occurs during a time period. The set of flows associated with virtual machine, v_i, is denoted by f_i and f_{ij} represents a single flow in f_i. The output, MC, is a set of pairs, where each pair is a virtual machine to be migrated and the data center that the virtual machine is to be migrated to. The function *load* returns the load of the input flow. The function *region* can either take as input a flow or a VM. For a flow, the function returns the region of the flow's origin and for a VM it returns the region of the data center that the VM is located in. The function *distance* calculates the distance between two regions.

The algorithm calculates for each virtual machine the average load-distance, which is denoted by d_0. The **for** loop starting at line 7 sums the weighted load distance of flows to virtual machine, v_i. This allows for the calculation of the average load distance in line 13.

The variables, C and L, represent counter and load lists where C[j] represents the number of flows that originate in region j, and L[j] represents the load generated by the requests originating in region j. These counters are initialized in Lines 3 through 5. For each virtual machine lines 7 through 12 determines the number of requests for each region and the load from each region.

Selector Algorithm 1

```
    Input: F,V
    Output: MC
1.  for each $v_i$ in V
2.     $d_0 \leftarrow 0$
3.     for $j \leftarrow 0, N-1$ do
4.        $C[j], L[j] \leftarrow 0$
5.     end for
6.     $r_{vm} \leftarrow region(v_i)$
7.     for each flow $f_{ij}$ of $f_i$
8.        $r_{flow} \leftarrow region(f_{ij})$
9.        $d_0 \leftarrow d_0 + distance(r_{flow}, r_{vm}) * load(f_{ij})$
10.       $C[r_{flow}] \leftarrow C[r_{flow}]+l$
11.       $L[r_{flow}] \leftarrow L[r_{flow}]+load(f_{ij})$
12.    end for
13.    $d_0 \leftarrow d_0 / | f_i |$
14.    $t = i \ni \forall j \ C[j]L[j] \leq C[i]L[i], \ 0 \leq i,j \leq N-1$
15.    $d_1 \leftarrow 0$
16.    for each flow $f_{ij}$ of $f_i$
17.       $r \leftarrow region(f_{ij})$
18.       $d_1 \leftarrow d_1 + distance(r, t) * load(f_{ij})$
19.    end for
20.    $d_1 \leftarrow d_1 / | f_i |$
21.    if $d_0 > d_1$
22.       $MC \leftarrow MC \cup \{(v_i, t)\}$
23.    end if
24. end for
```

Algorithm 1: Selector Algorithm 1

The product of the flow counter and load counter reflects the traffic originating from a region. Higher values suggest many packets are flowing from the region. This is used to select the data center of the region with the maximum value of the product as a possible target data center for migration (Line 14). This in itself is not sufficient for selecting a target data center since a high value could be the result of many flows or perhaps few flows with heavy load. Essentially this represents load from a region, but does not consider distance. The potential target data center selected in Line 14 requires a calculation (Lines 15 to 19) of the average load distance of the flows of the time period, t_i, that assumes that the VM being considered for migration is in the data center selected in Line 14. This uses the flow information gathered in the time period. If the average load-distance using the data center selected in Line 14 is less than the current data center then the data center in Line 14 is chosen to migrate the VM to (Lines 21 to 23). Otherwise, the virtual machine is not migrated.

4.2 Selector Algorithm 2 for Migration

The algorithm described in Section 4.1 considers only one data center while the algorithm described in this section considers all data centers. The inputs and output are the same as that for Selection Algorithm 1. Lines 1 to 8 of Selector Algorithm 2 calculate the average load distance for a virtual machine. The calculation is similar to that of Selection Algorithm 1. Lines 10 to 22 of Selector Algorithm 2 is similar to Lines 16 to 20 of Selector Algorithm 1 except that the calculation is done for each data center and the data center selected as the target data center is the one with the smallest average load distance.

```
Selector Algorithm 2
Input: F,V
Output: MC
1.  for each $v_i$ in V
2.     $d_0 \leftarrow 0$
3.     $r_{vm} \leftarrow region(v_i)$
4.     for each flow $f_{ij}$ of $f_i$
5.         $r_{flow} \leftarrow region(f_{ij})$
6.         $d_0 \leftarrow d_0 + distance(r_{flow}, r_{vm}) * load(f_{ij})$
7.     end for
8.     $d_0 \leftarrow d_0 / | f_i |$
9.     $t = r_{vm}$
10.    $d_1 \leftarrow 0$
11.    for each $DC_k$ in DC
12.       $d_m \leftarrow d_0$
13.       for each flow $f_{ij}$ of $f_i$
14.           $r_{flow} \leftarrow region(f_{ij})$
15.           $d_m \leftarrow d_m + distance(r_{flow}, k) * load(f_{ij})$
16.       end for
17.       $d_m \leftarrow d_m / | f_i |$
18.       if ($d_m < d_1$)
19.           $d_1 \leftarrow d_m$
20.           $t \leftarrow k$
21.       end if
22.    end for
23.    if $d_0 > d_1$
24.        $MC \leftarrow MC \cup \{(v_i, t)\}$
25.    end if
26. end for
```

Algorithm 2: Selector Algorithm 2

4.3 Selector Algorithm for VM Replication

The Selector Algorithm for VM Replication is similar to the algorithm described in Section 4.1 that identifies a target data center. However, rather than predicting the load-distance if a VM is migrated, it predicts the load-distance if a copy of the VM is replicated in a target data center. To calculate the predicted load-distance, it assumes that a flow will be served by the VM which is geographically closest to the origin of the flow. This algorithm also calculates the predicted load ratio for each VM which specifies the predicted distribution of load between the primary and the replica VM. This algorithm is executed along with the migration algorithm when the owner of the VM agrees to rent extra VM.

Lines 1 to 15 of Selector Algorithm for VM Replication calculate the average load distance of a virtual machine and identify a target data center. This is similar to Selector Algorithm 1. The calculation of the predicted load distance is different from Selector Algorithm 1 in Selector Algorithm for VM replication, as the flows are expected to be distributed among the original and the replicated copy of the VM. The predicted load-distance is calculated assuming that, after a replica of the VM is hosted in the target data center, client requests will be served by the VM which is geographically the closest to the client (Assuming that the DNS will return the IP address of the closest VM [49]).

In lines 16-18, variables l_c, l_t, load_ratio are initialized to 0. These variables represent the predicted load in the original VM, the predicted load in the replicated VM and the load ratio obtained from dividing l_c by l_t respectively. Lines 19 to 29 calculate the average load distance of the flows that assumes that the VM will be replicated in the potential target data center selected in Line 14 of the flows received in the time period, t_i. The distance of the flow's origin to the original data center and flow's origin to the target data centers are compared. The smaller distance is weighted by the flow load and added to d_1. This approach means that the calculated average load distance is based on flows being sent to the nearest data center. The expected load for the potential target data center and the current data center are also calculated (Lines 23 and 26).

Selector Algorithm for VM Replication

```
Input: F,V
Output: MC
1.  for each $v_i$ in V
2.    $d_0 \leftarrow 0$
3.    for $j \leftarrow 0, N-1$ do
4.      $C[j], L[j] \leftarrow 0$
5.    end for
6.    $r_{vm} \leftarrow region(v_i)$
7.    for each flow $f_{ij}$ of $f_i$
8.      $r_{flow} \leftarrow region(f_{ij})$
9.      $d_0 \leftarrow d_0 + distance(r_{flow}, r_{vm}) * load(f_{ij})$
10.     $C[r_{flow}] \leftarrow C[r_{flow}]+1$
11.     $L[r_{flow}] \leftarrow L[r_{flow}]+load(f_{ij})$
12.   end for
13.   $d_0 \leftarrow d_0 / |f_i|$
14.   $t = i \ni \forall j\ C[j]L[j] \leq C[i]L[i],\ 0 \leq i,j \leq N-1$
15.   $d_1 \leftarrow 0$
16.   $l_c \leftarrow 0$
17.   $l_t \leftarrow 0$
18.   $load\_ratio \leftarrow 0$
19.   for each flow $f_{ij}$ of the $f_i$
20.     $r_{flow} \leftarrow region(f_{ij})$
21.     if $(distance(r_{flow}, r_{vm}) > distance(r_{flow}, t))$
22.       $d_1 \leftarrow d_1 + distance(r_{flow}, t) * load(f_{ij})$
23.       $l_t \leftarrow l_t + load(f_{ij})$
24.     else
25.       $d_1 \leftarrow d_1 + distance(r_{flow}, r_{vm}) * load(f_{ij})$
26.       $l_c \leftarrow l_c + load(f_{ij})$
27.     end if
28.   end for
29.   $d_1 \leftarrow d_1 / |f_i|$
30.   $load\_ratio = l_c / l_t$
31.   if $d_0 > d_1$
32.     $MC \leftarrow MC \cup \{(v_i, t)\}$
33.   end if
34. end for
```

Algorithm 3: Selector Algorithm for VM Replication

A VM replication in the target data center can reduce the load-distance for its flows (hence reduces data access and transfer time for its clients) along with serving the primary goals of VM replication such as increasing reliability, preventing single point of failure and distribution of load. Moreover, an SDN controller can forward the flows to the replicated VM in case of a failure or crash of the primary VM and vice versa.

Line 30 calculates load ratio as l_c/l_t. The load ratio provides an idea of how the load is expected to be distributed between the VMs if shortest path routing is applied (We assume that the geographically closest VM has the shortest path from the origin of the flow). From the load ratio, the cloud provider can determine the processing capacity and resource requirement of the replica VM. Lines 31 to 34 are the same as lines 21 to 24 of Selector Algorithm 1. A cloud resource manager can use the load-ratio to adjust resources allocated to the primary VM and its replica. The predicted load ratio could result in a decision results in a migration and VM replication. If the predicted load-ratio suggest that most of the requests will be served by either of the VM rather than a uniform distribution, only migration can be beneficial rather than using two VMs. However, if the VM is stressed with overload and a single VM is not sufficient to serve its clients, a replication along with a migration can be triggered to the target data center when the distribution of load is not uniform. When more than one VM is hosted by a single data center, the load can be distributed between the VMs by the method described in [16].

4.4 Summary

This Chapter presented two possible migration algorithms and a possible replication algorithm. The use of the migration and replication algorithms depends on various factors e.g., the willingness of the owner to pay for additional VMs, expected distribution of the load between the primary VM and its replica etc. For example, a migration algorithm may only be used if the owner is not willing to pay for an additional VM. Although not a primary focus of the thesis it is possible to apply both migration and replication algorithms that could result in a replica and a migration when the predicted load-ratio is not uniform between the VMs.

Chapter 5

5 Experimental Design and Results

This chapter describes the performance of the algorithms proposed in chapter 4. Section 5.1 describes the simulator environment, section 5.2 describes the simulation parameters and scenarios, section 5.3 describes simulation metrics for evaluation, section 5.4 describes the experimental results of VM migration algorithms (Selector Algorithm 1 and Selector Algorithm 2) and section 5.5 describes the experimental results of the VM replication algorithm (Selector Algorithm for Replication).

5.1 Simulation Environment

A simulation tool was developed using the Java programming language on Netbeans IDE. The simulation tool has the output described for Selector Algorithms 1 and 2 and Selector Algorithm for Replication. A geographic area is represented using a grid. The grid space is divided into regions based on the position of the data centers based on Voronoi diagrams [27]. With a Voronoi diagram, if N seed points are given, the given space is divided into N regions corresponding to the seed points in such a way that all points in a region is closest to the seed point of its own region. We used the location of the data centers as seed points to divide the grid space into regions.

Flows can be randomly generated by randomly generating the origin, load, data center and VM. The distance of a flow to a data center is calculated using the Euclidian distance of the origin of the flow from the data center. The simulation tool randomly selects load based on a range from .1 to 1 for each flow, where 1 represents the maximum possible load of a flow and .1 represents the minimum possible load of a flow.

5.2 Simulation Parameters and Scenarios

The simulation parameters include the number of data centers, the number of VMs to be hosted in a data center, and the range of the number of flows for each region. The number of flows is randomly generated but the number is within the range. The values that the simulation parameters may have are presented in Table 1. The dimensions of the grid are 2000x2000. Space considerations do not allow us present all the results in this chapter but results not in this chapter are in the appendices. The results presented in this chapter are representative.

Number of DC	5	10	15	20	25
Number of VMs per DC	5000	2500	1500	1000	200
Number of requests per region	5000 to 10000		10000 to 20000		20000 to 30000

Table 1: Simulation Parameters

Several scenarios were considered: (1) Scenario 1 assumes that all requests are randomly generated; (2) Scenario 2 assumes all flows come from a single region; (3) Scenario 3 divides the grid into four equal quadrants with all flows from the upper left quadrant; (4) Scenario 4 assumes that 50% of the total flows are from the upper left quadrant of the grid and rest of the flows are randomly generated from the entire grid space.

5.3 Metrics for Evaluation

We used three metrics for evaluation for our experiments as described below.

Percentage of Expected Improvement*:* For each VM in a data center, the *percentage of expected improvement* is defined as: $((d_0-d_1)\ /d_0)*100\%$. This metric provides a measure of how much of the current load-distance will be reduced if the VM is migrated to the target data center.

Percentage of candidate VMs identified: Selector Algorithm 2 is designed to search all data centers (region) to find the data center that is expected to best improve load-distance. If N_1 is the number of VMs identified for migration by Selector Algorithm 1 and N_2 is the number of VMs identified for migration by Algorithm 2 then the *percentage of candidate VM identification* is calculated as: (N_1/N_2)*100%. A higher percentage is indicative of Selector Algorithm 1's success in identifying candidate VMs for migration.

Load Ratio: The *load ratio* represents the expected distribution of load between the primary VM and its replica. This metric is used to compare the Selector Algorithm for VM replication and a random replication strategy (described later in this chapter). After a replica of the VM is created in the target data center, if the primary VM is expected to serve a total load of l_1 and the replicated VM is expected to serve a total load of l_2, then the expected load ratio is calculated as (l_1/l_2). The load ratio provides insight on the load between the primary VM and its replica.

5.4 Results of VM migration algorithms

We describe the experimental results of Selector Algorithm 1 and Selector Algorithm 2 in this section. Each experiment is repeated 100 times with the results presented reflecting the average of the repetitions.

The first set of experiments focused on evaluating the algorithms when Scenario 1 is assumed. Figures 8, 9 and 10 present a comparison of the proposed algorithms based on percentage of expected improvement for 5000 VMs, 1000 VMs and 200 VMs respectively, and for a range of data center sizes. The number of flows is randomly generated and within the range of 20,000 to 30,000 for each region. The x-axis represents the number of data centers and the y-axis shows the percentage of expected improvement in load-distance obtained through migration. For each data center, the bar to the left (blue) represents Selector Algorithm 2 and the bar to the right (red) represents Selector Algorithm 1.

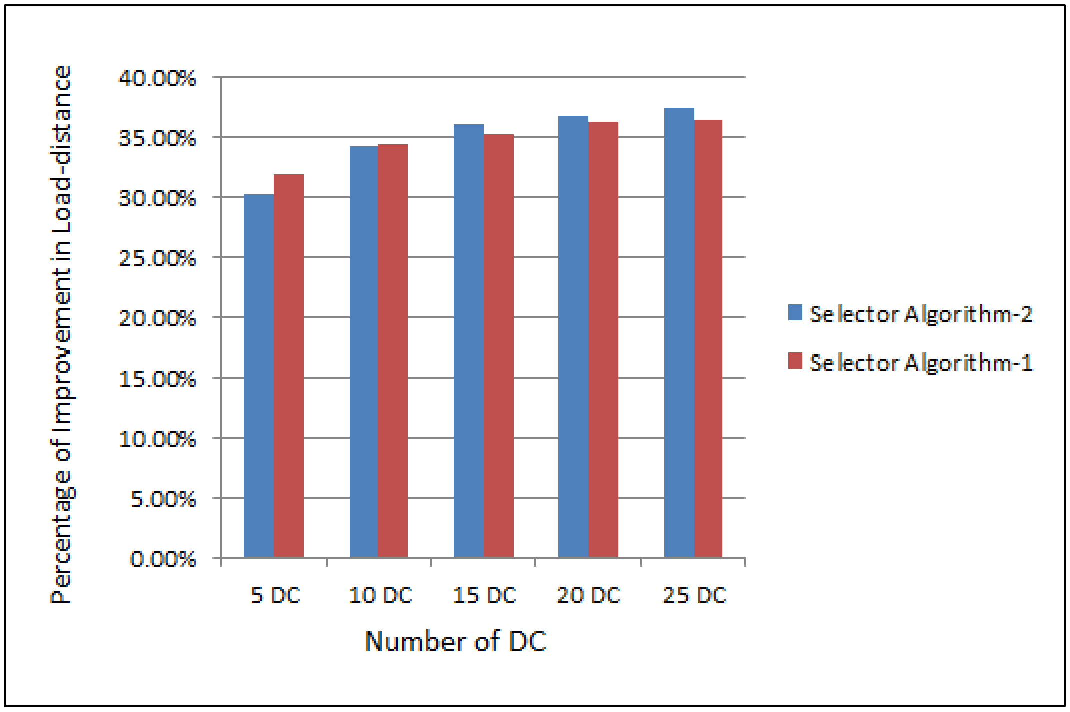


Figure 8: Scenario 1. Each data center has 5000 VMs

Figure 8 shows that for 5000 VMs in a data center both algorithms are similar in the percentage of expected improvement of load-distance.

Figure 9 also shows that with 1000 VMs in a data center that both algorithms are similar in the percentage of expected improvement of load-distance. The t-test ($p < 0.05$) suggests that the difference between the two algorithms is statistically insignificant with respect to the percentage of improvement in load-distance metric.

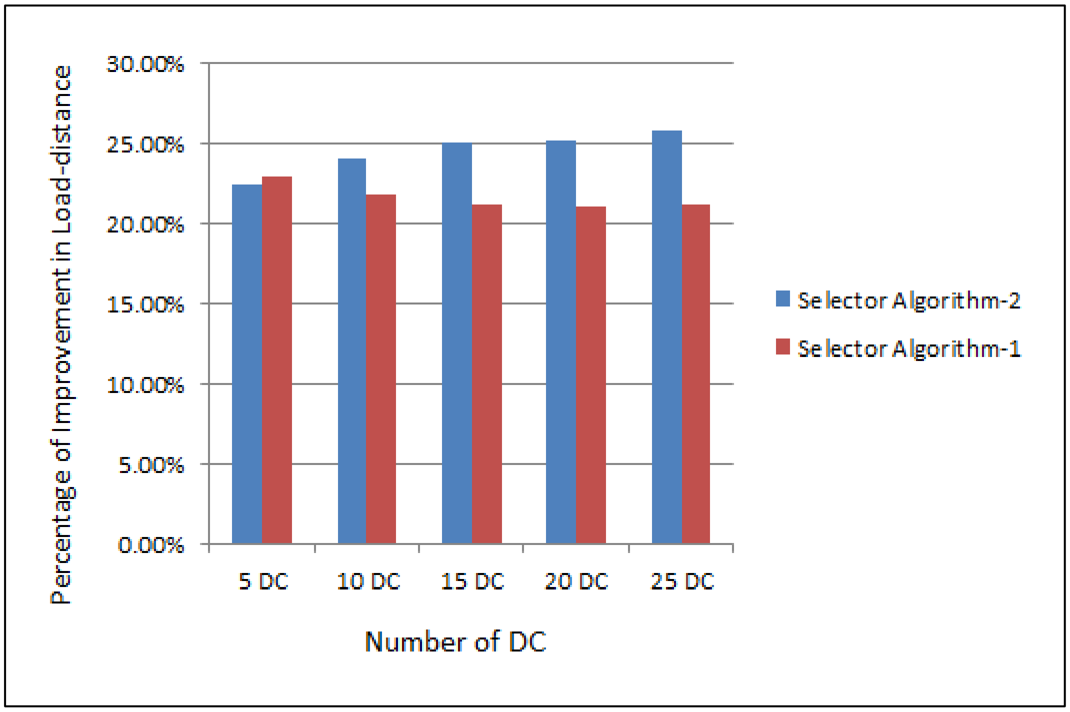


Figure 9: Scenario 1. Each data center has 1000 VMs

Figure 10 shows that with 200 VMs in a data center that as the number of data centers increases the gap in the expected percentage of expected improvement of load-distance for Selector Algorithm 1 and Selector Algorithm 2 increases.

The reason relates to the number of flows per region. The number of flows per region is in the same range (20000 to 30000). This means that the number of flows per virtual machines increases with the decrease in the number of virtual machines. For example, if the number of flows generated is in the range of 20000 to 30000 and there are 200 VMs in the data center, then each VM receives on average 25 times more flows than a data center that assumes 5000 VMs. The origin of a flow is randomly generated and thus more flows increases the likelihood that more regions are originators of flows to a VM. This prevents a single region from monopolizing a VM and thus decreases the effectiveness of Selector Algorithm 1. This is also reflected in Figure 11, where we see that the percentage of candidate VMs identified by Selector Algorithm 1 for migration decreases as the number of VMs per data center decreases.

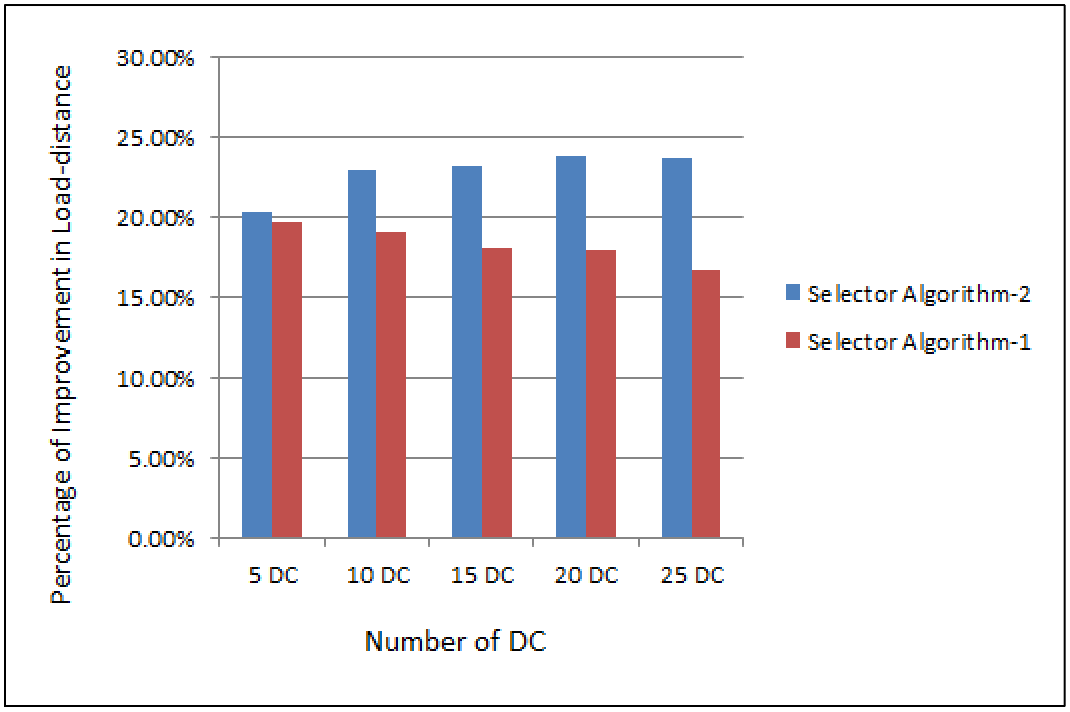


Figure 10: Scenario 1. Each data center has 200 VMs

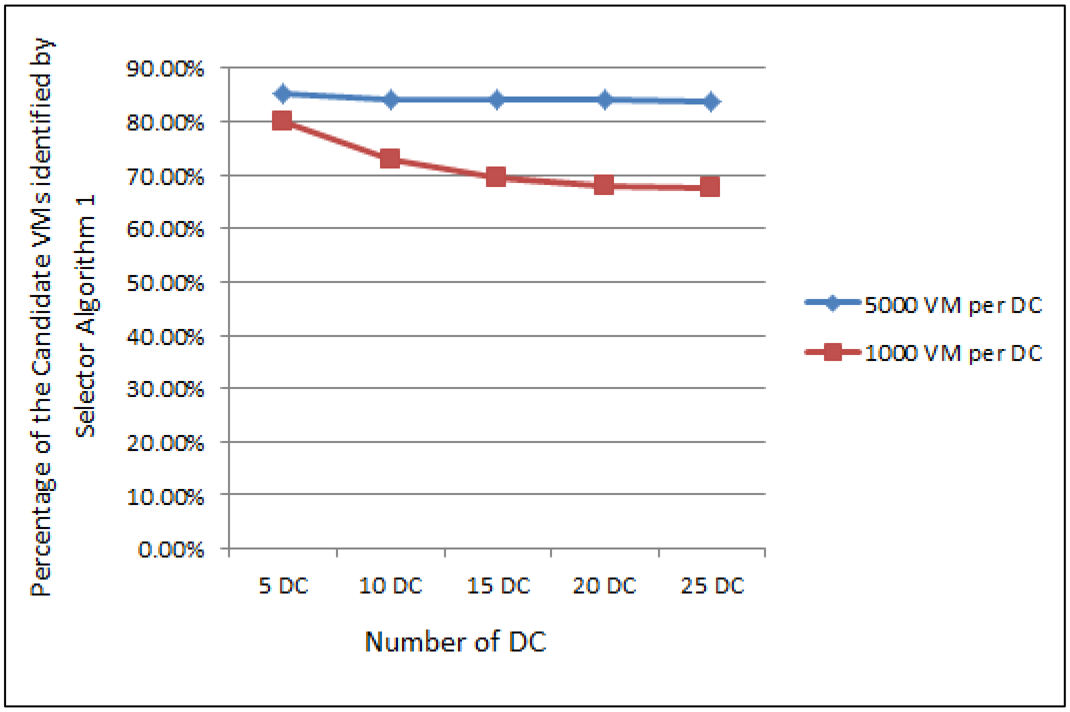


Figure 11: Scenario 1. Percentage of candidate VMs for migration identified by Selector Algorithm-1

This led to further experiments where we varied the number of flows to VMs in a data center. Selector-Algorithm 1 continued to decrease in effectiveness with more flows per data center. The rest of the experiments representing the percentage of improvement and the percentage of candidate VMs identified can be found respectively in Appendix A and Appendix B.

Figures 8,9,10 show that the percentage of expected improvement in load-distance is higher as the number of VMs per data center increases. As the number of data centers increases for the same number of flows, there is less likelihood that a region dominates.

With scenario 2 flows originate from a single region but all other simulation parameters are the same as the experiment described for the first scenario. Regardless of the simulation settings, both algorithms showed an expected improvement in load-distance that is in the range of 88% to 90%. Figure 12 shows that with 1000 VMs per data center and 20000 to 30000 flows per region, both algorithms output an 88% to 90% improvement in load-distance. Figure 13 shows that with 200 VMs per data center and 20000 to 30000 flows per region, both algorithms output an 88% to 90% improvement in load-distance. Scenario 2 suggests that if the flows for a VM are localized to the granularity of a region then there is little difference between Selector Algorithm 1 and Selector Algorithm 2 regardless of the number of data centers and flows per VM. Scenario 2 also suggests that when all the flows are generated around a single region, increment of flows per VM does not affect the performances of any of the algorithms.

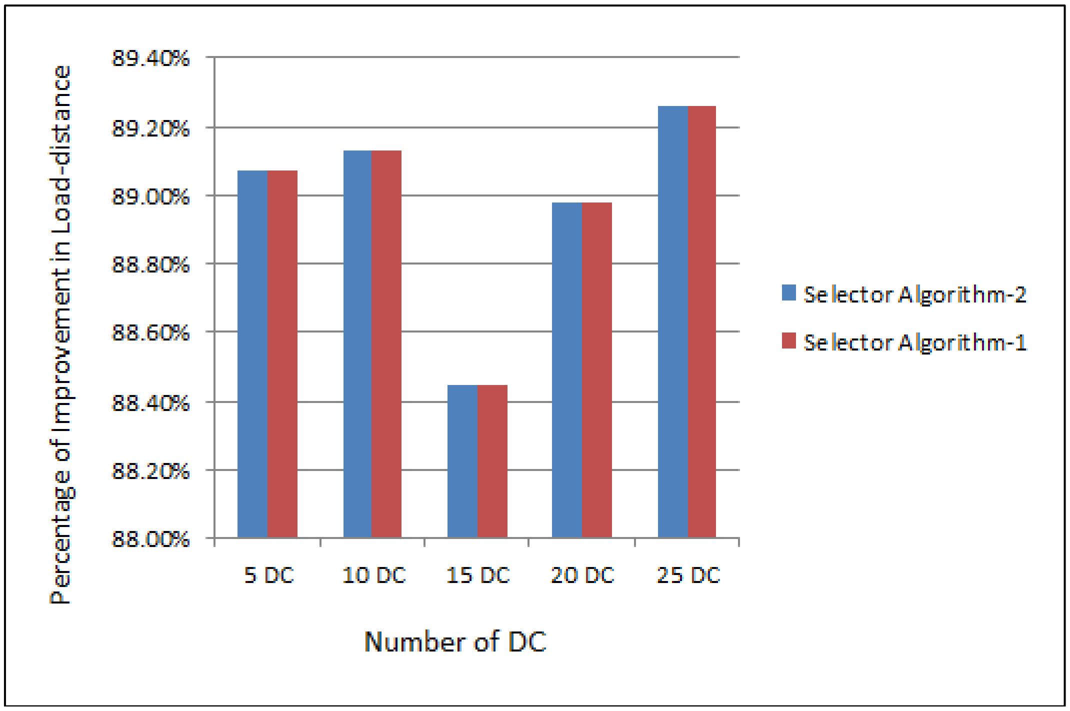


Figure 12: Scenario 2: Each data center has 1000 VMs

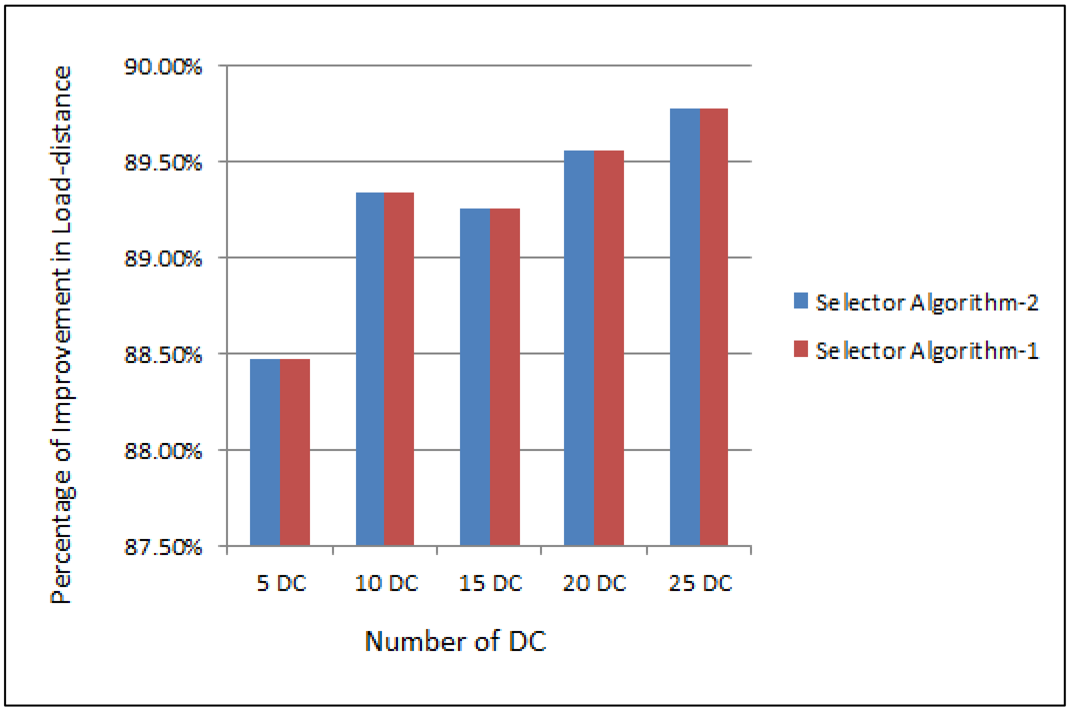


Figure 13: Scenario 2: Each data center has 200 VMs

With scenario 3, all flows are from the upper left quadrant of the grid, which is one-fourth of the entire grid space, but all other simulation parameters are the same as described for the first scenario. The results show that regardless of the number of data centers, both algorithms were close with respect to the percentage of expected improvement of load distance. Figure 14 represents an improvement of 53% to 74% in expected load-distance for 1000 VMs per data centers and figure 15 represents 57% to 72% of improvement in load distance for 200 VMs per data center. In cases we used 20,000 to 30,000 flows per region. This suggests that even at the granularity of a quadrant that there is sufficient localization for Selector-Algorithm 1 to be effective. The performance of the algorithms also does not deteriorate with the increase of flows per VM.

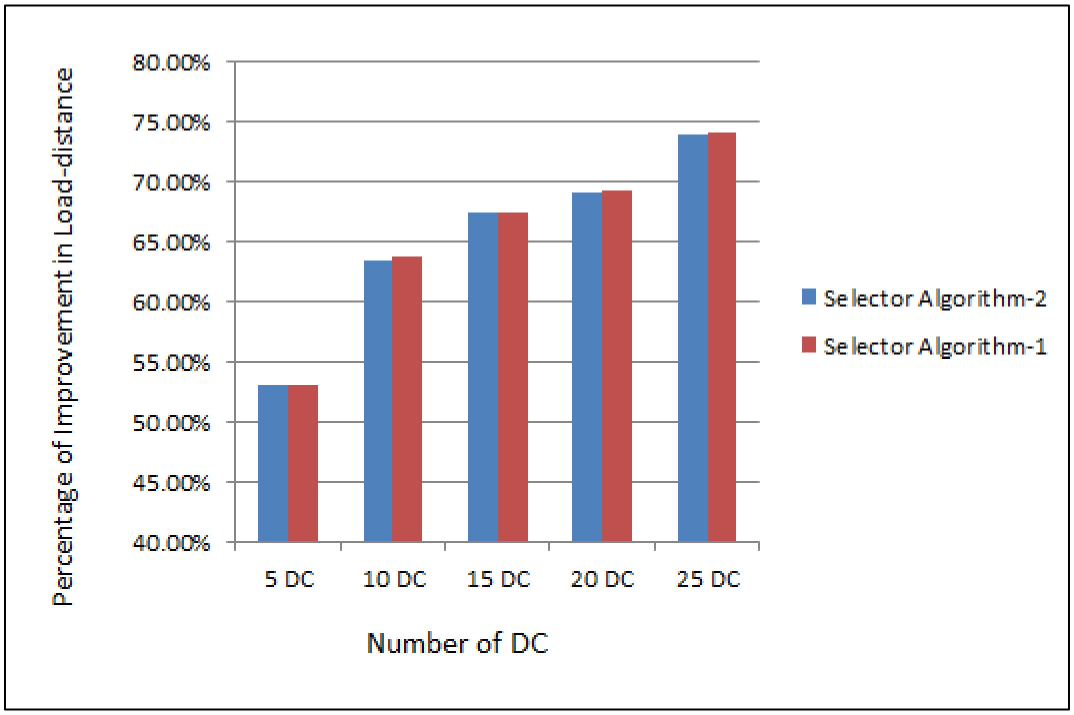


Figure 14: Scenario 3: Each data center has 1000 VMs

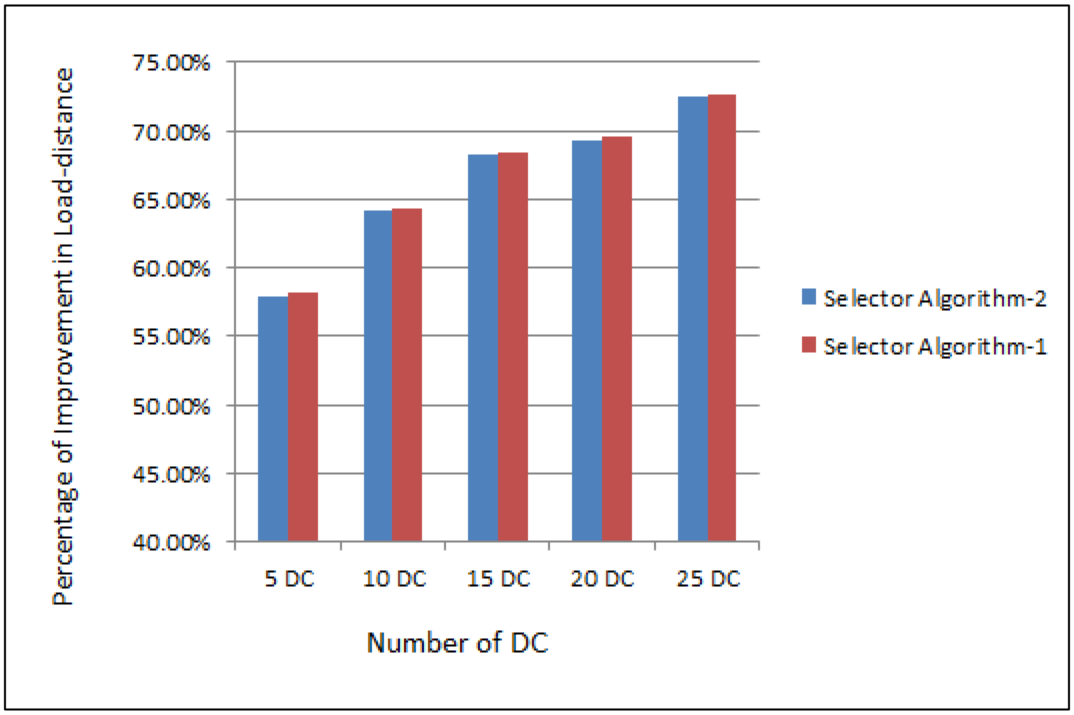


Figure 15: Scenario 3: Each data center has 200 VMs

In scenario 4, 50% of the flows originated from the upper left quadrant of the grid. The origins of the rest of the flows were generated randomly from the entire grid space. All other simulation parameters are the same as scenario 1. As we can see in figure 16 and 17, both algorithms have a similar percentage of improvement in expected load-distance. An improvement of 33% to 37% in expected load-distance was observed for both 1000VMs per data center (Shown in figure 16) and 200 VMs per data center (Shown in figure 17). Scenario 4 represents a localization of flows although not as strong of localization as seen in Scenarios 2 and 3. However, the flows from a region still dominate regardless of the number of flows for a VM, which increases with fewer VMs.

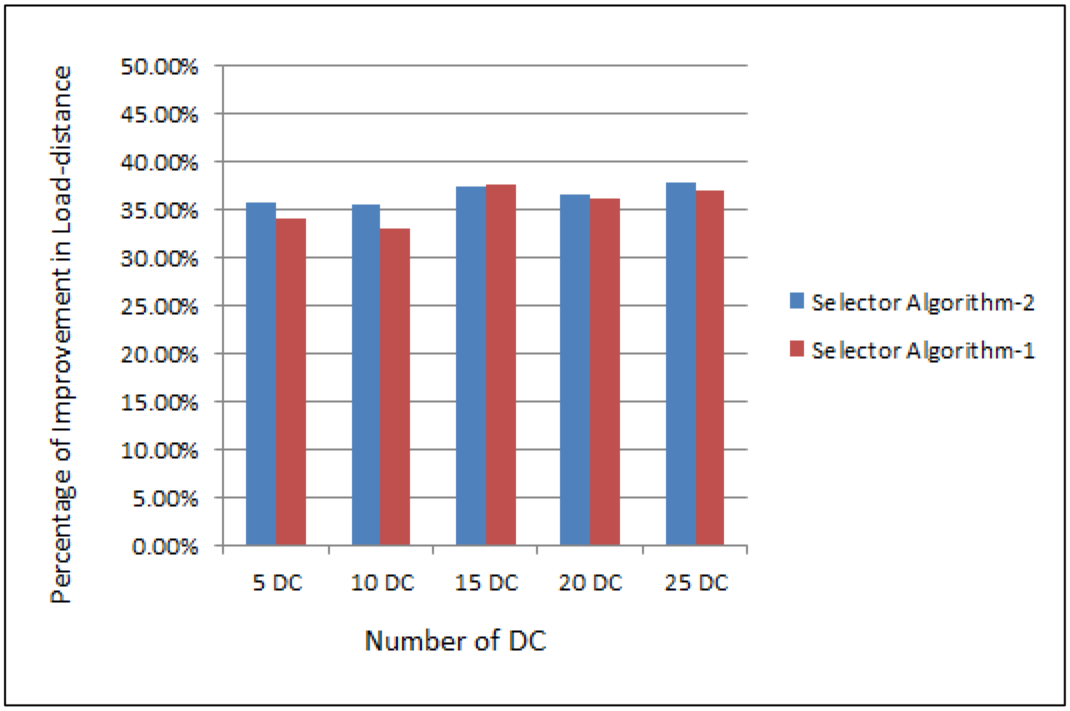


Figure 16: Scenario 4: Each data center has 1000 VMs

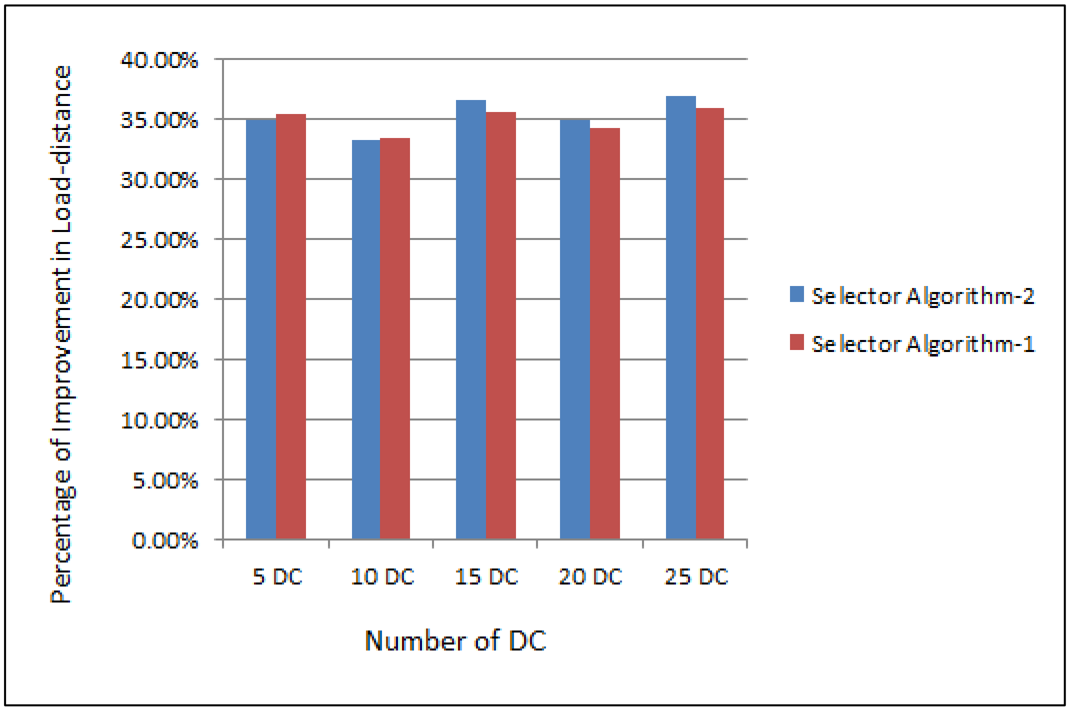


Figure 17: Scenario 4: Each data center has 200 VMs

The experiments of scenario 4 suggest that when a region dominates in number of flows, both algorithms have a similar output with reasonable improvement in load-distance. The performance of the algorithms also does not deteriorate with the increase of flows per VM.

Further investigation is seen with Figure 18, which represents a comparison between the percentage of the candidate VMs identified by Selector Algorithm 1 for 1000 VMs per data center in scenario 3 and scenario 4. The line representing 50% of flows from the upper left quadrant (ULQ) represents scenario 4 and the other line represents scenario 3. We observed that for scenario 3, the percentage of the candidate VMs identified by Selector Algorithm 1 is very close to 100% where for scenario 4, it ranged from 91% to 95%.

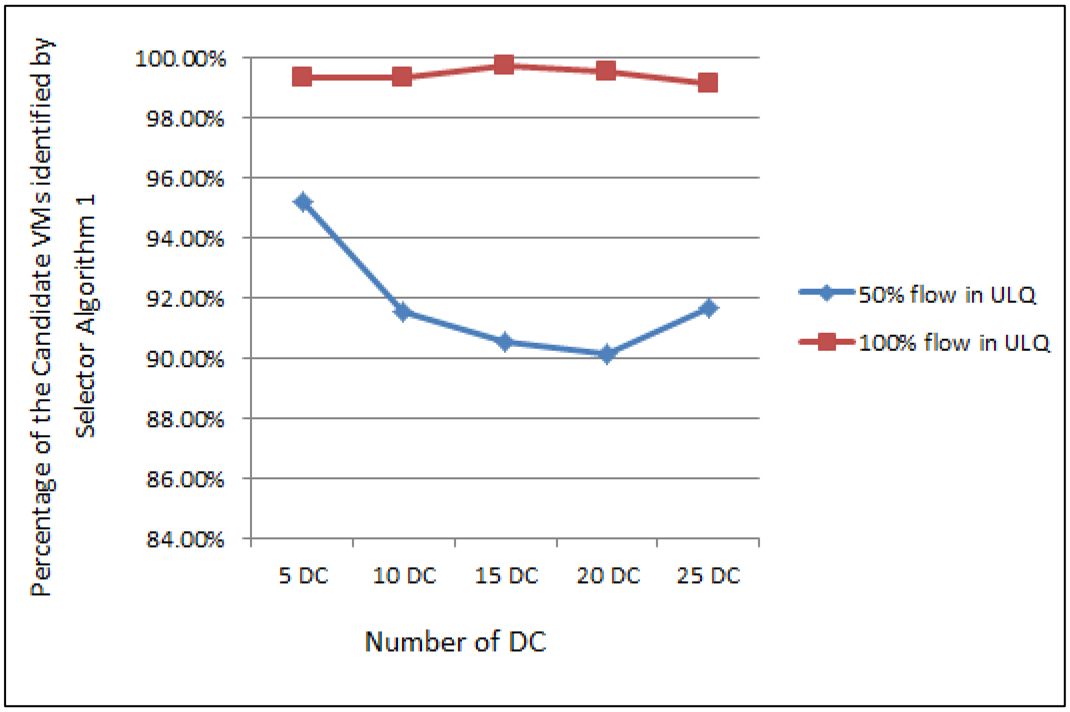


Figure 18: Scenarios 3,4: 1000 VMs per DC

Figure 19 compares the percentage of candidate VMs identified by Selector Algorithm 1 for scenarios 3 and 4 where the number of VMs per data center is 200. We observed that for scenario 3, the percentage of the candidate VMs identified by Selector Algorithm 1 is very close to 100% where for scenario 4, it ranges from 90% to 95%. These results show that Selector Algorithm 1 is more effective in scenarios 2, 3, 4 than scenario 1. This suggests that the efficiency of Selector Algorithm 1 for identifying candidate VM increases when flows in some regions dominate over others.

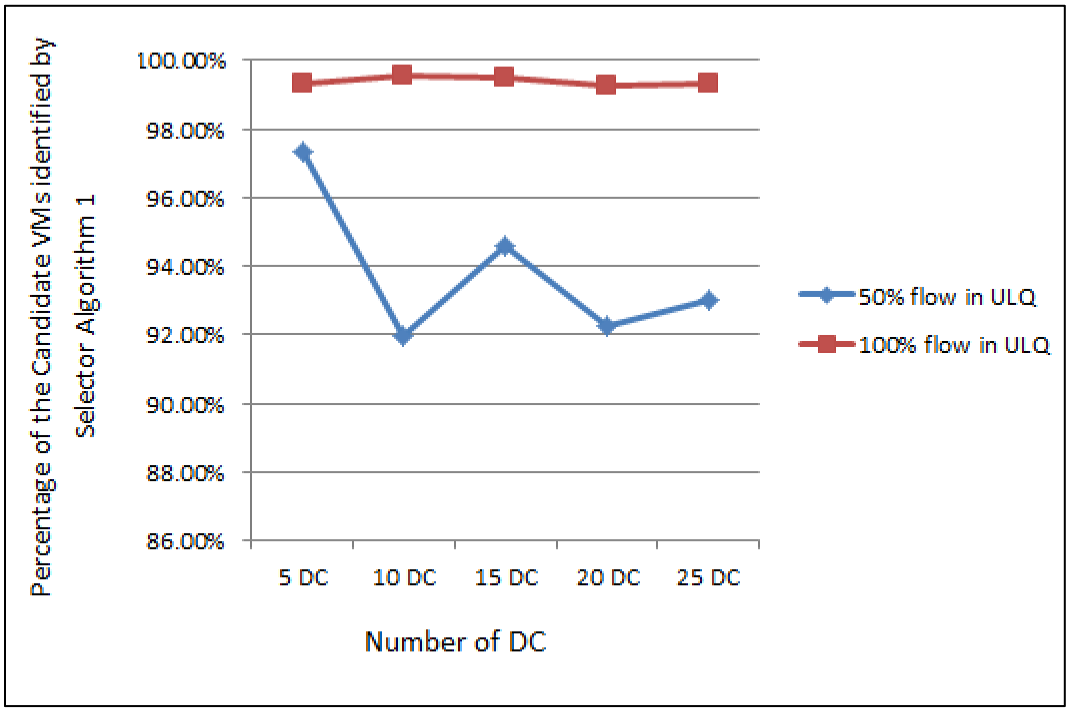


Figure 19: Scenarios 3,4: 200 VMs per DC

5.5 Results of VM replication algorithm

Each experiment described in this section is repeated 100 times with the results presented reflecting the average of the repetitions. This section describes the observed results of replication algorithm. To illustrate the effectiveness of the replication algorithm, the results of the proposed replication algorithm are compared with a random replication strategy. With random replication, a target data center is chosen randomly among the rest of the available data centers to host a replica of the VM and observe the resulting load-distance when requests are served from their closest VM. Creating a replica of the VM in the current data center for load distribution is a common technique, but this technique will not reduce the access and transfer time for clients as the requests will be served by the VMs from the same data center.

The first set of experiments focused on evaluating the replications algorithms when Scenario 1 is assumed. Figures 20, 21 and 22 present a comparison between the VM replication between the proposed algorithms and a random replication based on percentage of expected improvement for 5000 VMs, 1000 VMs and 200 VMs respectively, and for a range of data center sizes. The number of flows is randomly generated and within the range of 20,000 to 30,000 for each region.

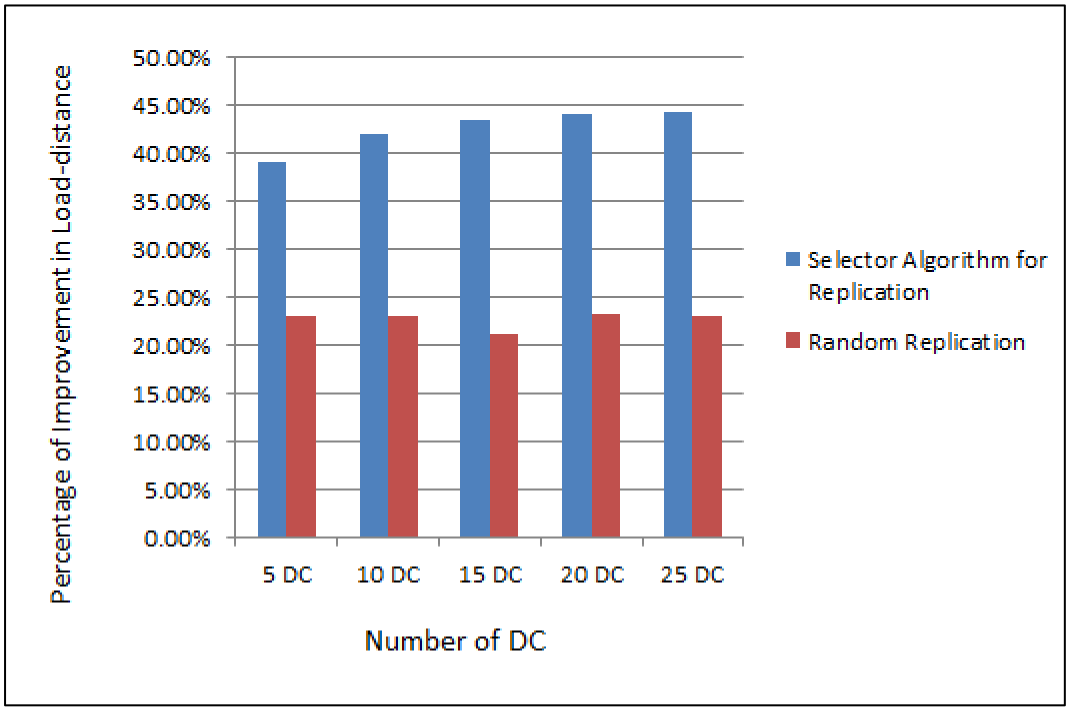


Figure 20: Scenario 1. Each data center has 5000 VMs

Figures 20, 21, 22 shows that the proposed algorithm out-performs the random replication with respect to the percentage improvement in average load distance compared to the random replication regardless of the number of data centers. The performance of the selector algorithm for replication declines with a decrease in the number of VMs per data center. As the number of VMs per data center decreases, each

VM tends to receive more flows from each region, preventing a single region from dominating. Thus the difference between our proposed algorithm and random replication decreases. This is similar to the migration algorithms.

If the VM owner attempts to create a replica of their VM in order to decrease load over a single VM as well as to prevent a single point of failure, in the context of load-distance it is better to replicate it in some other data center. As we can see that even random replication provides a constant near around 25% improvement in load-distance, while a replica in the same data center only would share the load, but the load-distance would remain unaltered.

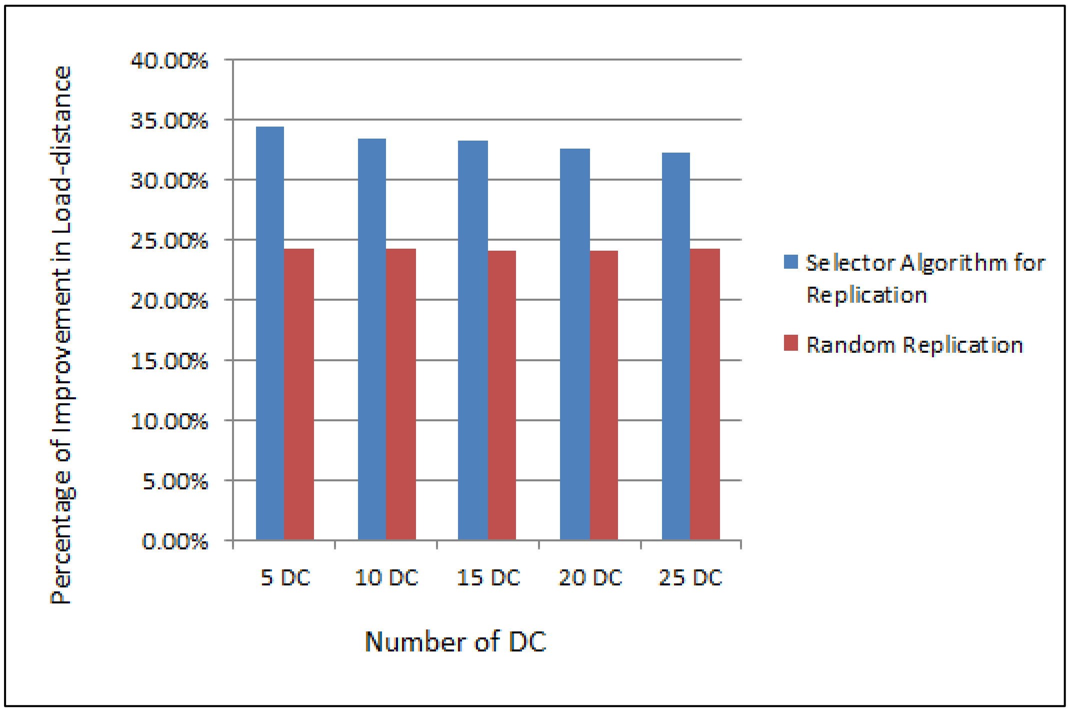


Figure 21: Scenario 1. Each data center has 1000 VMs

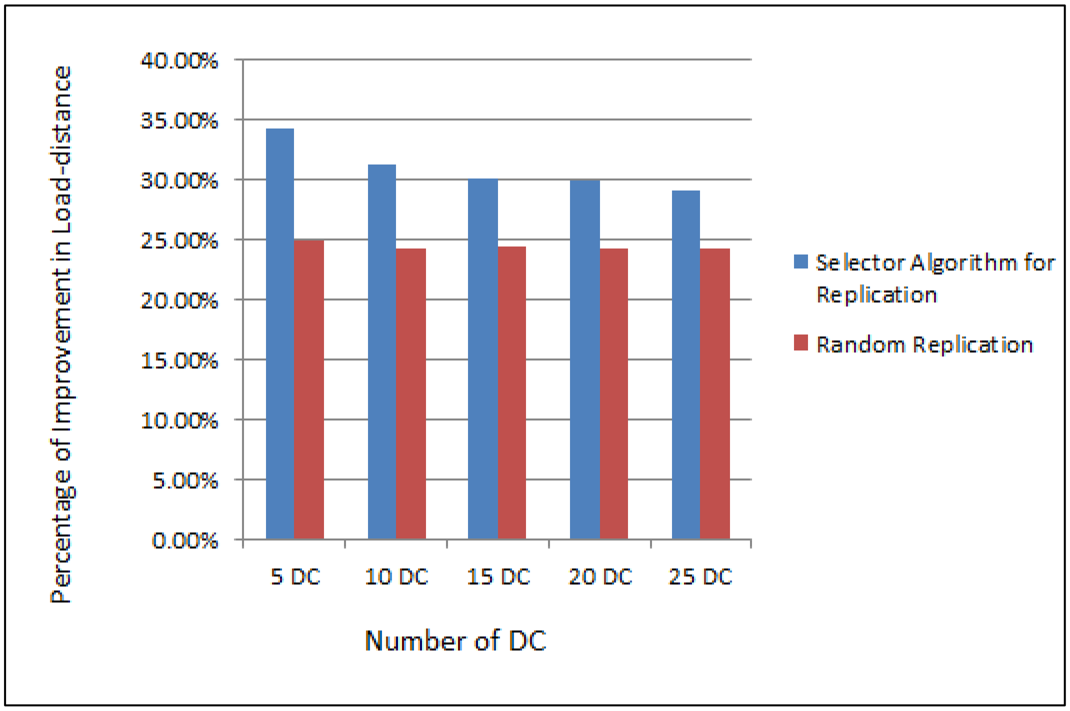


Figure 22: Scenario 1. Each data center has 200 VMs

We also observe the distribution of load between the primary VM and the replicated VM for scenario 1. Figure 23 depicts the distribution of load between the VMs through the load ratio for 200 VMs per data center. We observe for the proposed algorithm that for the most part as the number of DCs increase, the load ratio is close to 1, which represents that with the proposed algorithm the load is distributed among the VMs almost equally. With random replication, the primary VM has to serve almost twice as the number of flows served by the replicated VM. We also observed similar results for 1000 VMs per data center and 5000 VMs per data center. Detailed experiments on percentage of improvement and distribution of load can be found respectively in Appendix C and Appendix D.

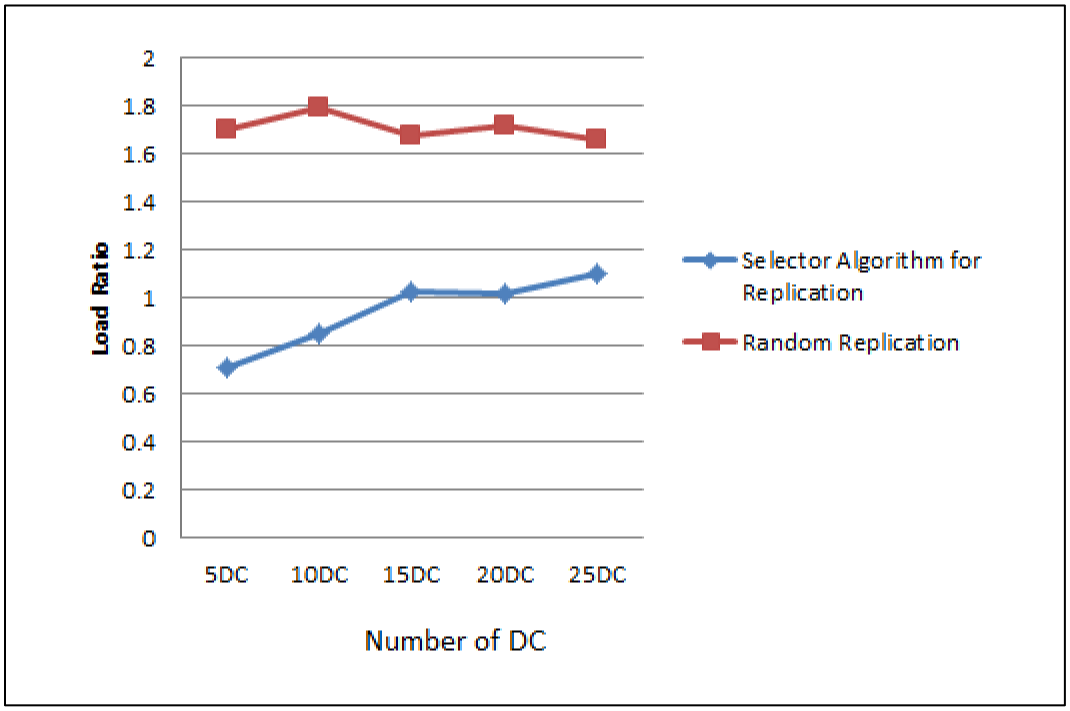


Figure 23: Scenario 1. 200 VMs per DC

For Scenario 2, the flows are generated from a single region, and thus its application would result in a replica of the VM in the data center of that single region. The average load distance does not change. A replica may be useful for other reasons but it does not reduce average load distance. Similar comments apply for Scenario 3.

In scenario 4, the performance of the proposed algorithm for 1000 VMs and 200 VMs is presented in figures 24 and 25, where 20,000 to 30,000 flows per region are generated. We observe that the performance of our algorithm is significantly higher than the random replication. We also observe that unlike scenario 1, decreasing the number of VMs per data center does not affect the performance of our algorithm with respect to average load-distance. This indicates that when flows from few regions dominate over others, the performance of proposed replication algorithm is unaffected regardless of the simulation parameters.

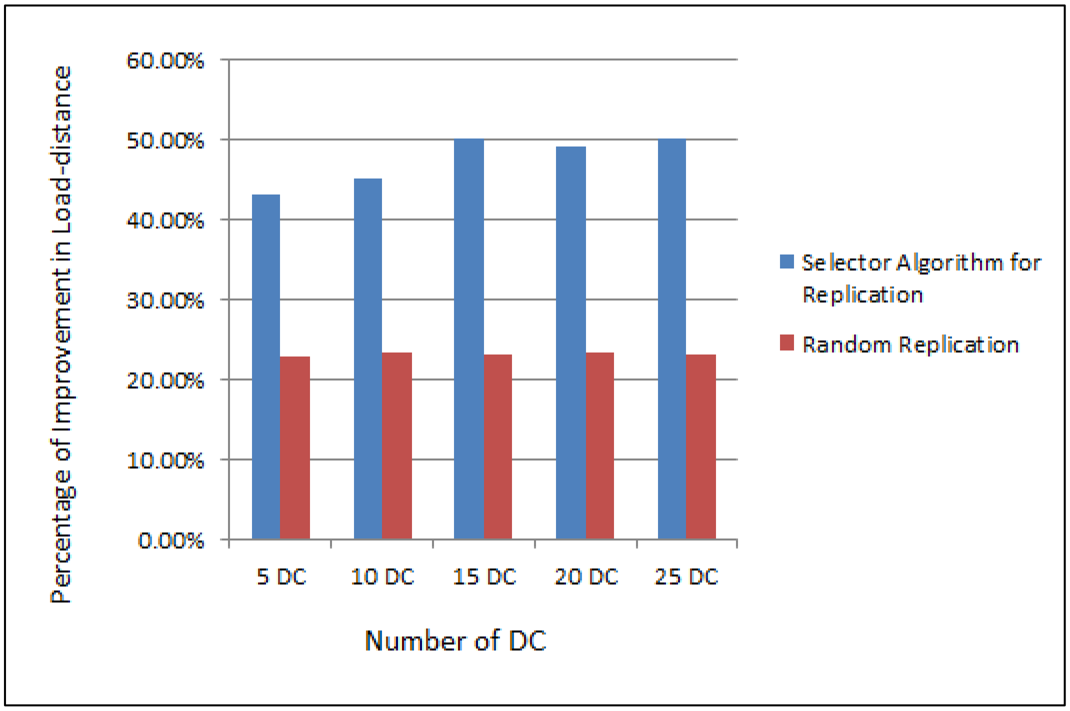


Figure 24: Scenario 4. 1000 VM per DC

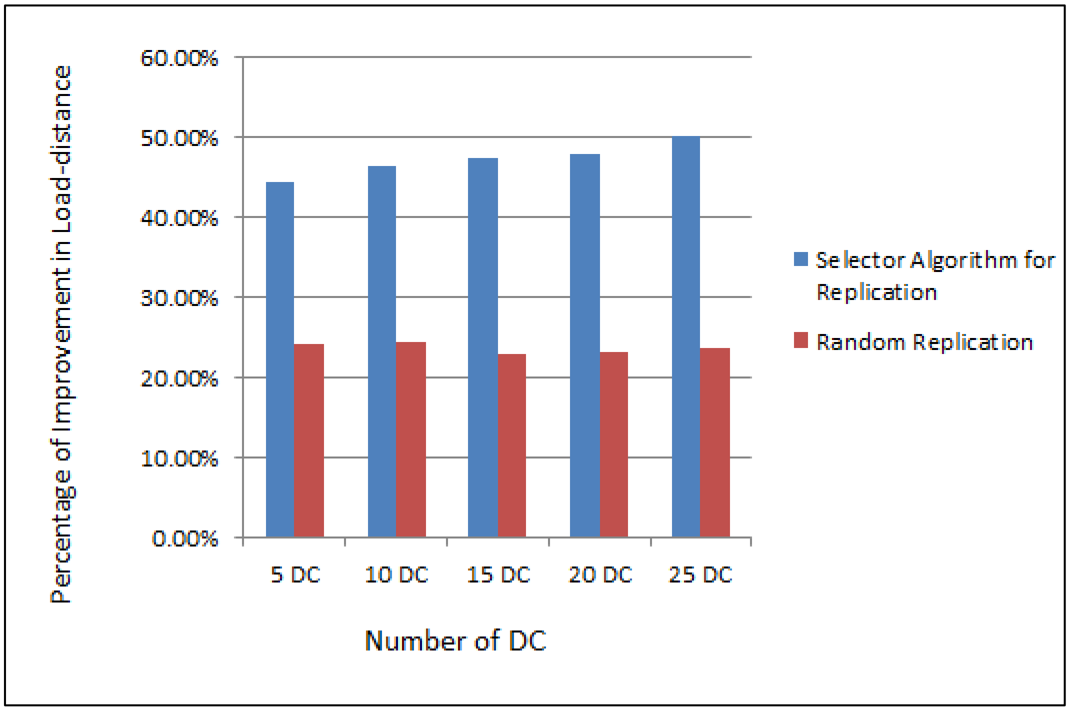


Figure 25: Scenario 4. 200 VMs per DC

Figure 26 depicts the distribution of load between the VMs through the load ratio metric for 200 VMs per data center. For our proposed algorithm the replicated VM is expected to receive almost twice as flows of the primary VM. As the VM is replicated in a region with a high density of flows, and we assume that all flows are served from the VM geographically closest to it, the replicated VM receives more flows. However with random replication, the situation is even worse as the primary VM is expected to serve almost three times the number of flows expected to be served by the replicated VM.

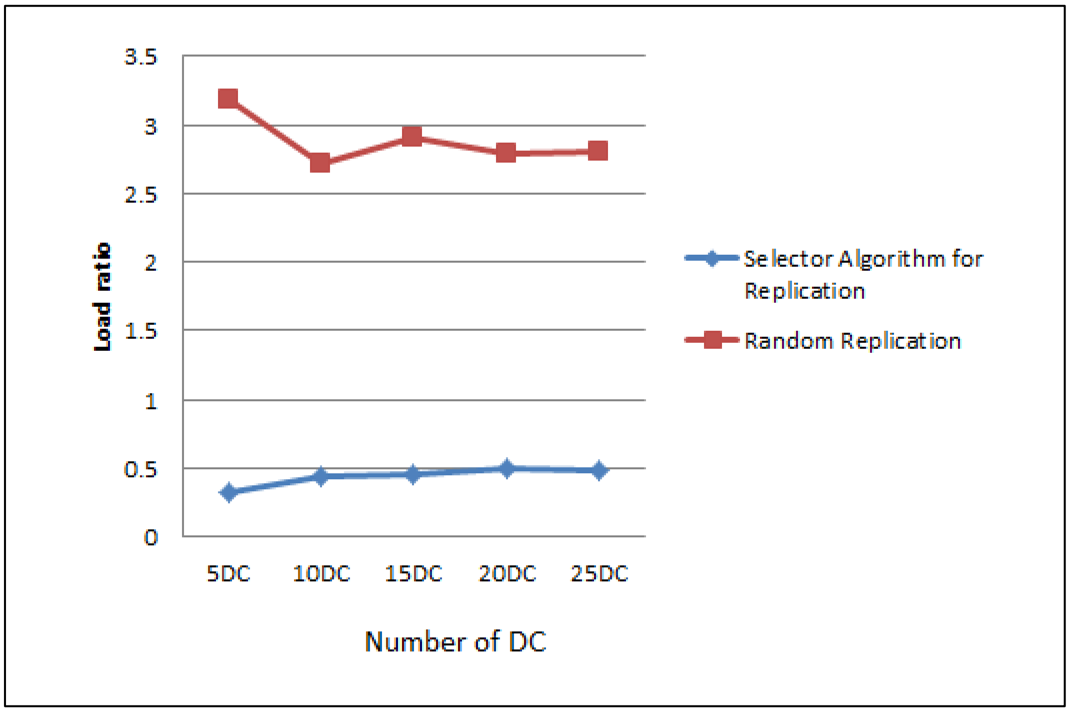


Figure 26: Scenario 4. 200 VMs per DC

5.6 Discussion on the Experiments

We observe from the experiments that both migration and replication can be effective in reducing average load-distance.

We observed for scenario 1 that when flows are randomly distributed over regions and each VM receives a fairly high number of flows, the effectiveness of both our migration and replication algorithms decreases, and Selector Algorithm 2 tends to outperform

Selector Algorithm 1. As flows are evenly distributed between regions, it becomes less likely to find a data center resulting in an average load-distance that is drastically better than the others. However, both of our migration algorithms are able to provide far better results when some region or few of the regions dominate in the number of flows over others. The effectiveness of both the migration algorithms are not affected as well as both of the migration algorithms continue to provide similar output with any simulation parameter when granularity of localization is used for the flows. However, as even for the flows without granularity of localization, expected improvement of load-distance is near around 20%, which is not insignificant.

We also observed that intelligent VM replication can be effective in order to reduce load-distance. We know that VM replication is commonly used in data centers to prevent single point of failure as well as distribute loads among the VM. Creating a replica of the VM in the same data center cannot reduce the load-distance. However, creating a replica of the VM in even in any other random data center can even reduce the load-distance up to 25% on top of fulfilling other purposes (e.g. preventing single point of failure, increasing reliability) of the VM replication stated.

Chapter 6

6 Future Work and Conclusion

With the advancement of web based services and applications, we observe a global trend of users and clients. As such applications with clients from different regions are emerging, as well as cloud is being a preferred platform to host applications, it requires necessary steps to keep the VMs closer to its clients as close as possible in order to ensure faster access and transmission of information.

We have presented distance aware VM migration and replication algorithms in this thesis. We have also presented a framework that can support our algorithms to deploy in large scale data centres. For N regions and M flows, Selector Algorithm 2 requires more computation since an average load-distance is calculated for each region while Selector Algorithm 1 does this for one region. The results show that Selector Algorithm 1 is effective especially in there is some localization of the origin of flows. Our replication algorithm shows that VM replication can be effective for reducing load-distance when flows are distributed across the regions. Future work is described in rest of this chapter.

Applying threshold in Migration and Replication Decision: We would like to apply a minimum threshold for migration and replication decision in the algorithms, thus no VM gets migrated or replicated unless its predicted improvement in load-distance is greater than the threshold value. Applying a threshold will halt migrations with lower improvement factors. We would like to experiment the outputs of our simulation parameters by varying threshold from 10% to 50%.

Using Past History as Predictor of Future Distribution: This work assumes the flow information in time period, t_i , is a predictor of flow information in t_{i+1}, which is known as Naïve approach of Forecasting.. This often works but there is a need to incorporate trend detection [50].

Algorithm Variations: Currently the algorithms fail if the target data center is not able to accommodate the VM. Selector Algorithm 1 could be modified such that a set of regions is considered. A region would be in this set if its product of the counters exceeds a threshold value. If none of the data centers associated with the regions can accommodate the VM then the algorithm would fail. A similar approach can be taken with Algorithm 2 except that this algorithm would create a set of potential data centers based on the predicated average load distance satisfying some criteria.

Switching Algorithms: Both algorithms work well under different circumstances. Using the work in Foster et. al. [16], we will develop a strategy for switching algorithms.

Incorporating CPU load: We have used only number of packet in flows to represent load. However, usage of CPU is not proportional to number of packets transferred as some of the flows may require a few information to pass but a lot of computation to do and vice versa. We plan to figure out a way in order to incorporate CPU load related to a flow.

Multiple Replications: We intend to modify the VM replication algorithm in order to support multiple replications in multiple data centers. Based on the maximum number of replications possible, we intend to identify a set of multiple target data centers. We would evaluate the algorithm when multiple replications take place. For those VMs with extreme load from multiple regions will require multiple replica VMs to increase reliability and distribute load among VMs.

Algorithms that permit Migration and Replication: Chapter 4 describes how the load ratio could be used to support a decision that VM should be migrated and replicated. We investigate algorithms for doing so and corporate CPU load and multiple migrations.

References

1. Abu Sharkh, Mohamed, Abdelkader Ouda, and Abdallah Shami. "A resource scheduling model for cloud computing data centers." *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*. IEEE, 2013.

2. Alicherry, Mansoor, and T. V. Lakshman. “Network aware resource allocation in distributed clouds." *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012.

3. Banikazemi, Mohammad, et al. " Meridian: an SDN platform for cloud network services." *Communications Magazine, IEEE* 51.2 (2013): 120-127.

4. Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti, “ A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks”, IEEE Communications Surveys and Tutorials.

5. Feamster, Nick, Jennifer Rexford, and Ellen Zegura. “The road to SDN: an intellectual history of programmable networks." *ACM SIGCOMM Computer Communication Review* 44.2 (2014): 87-98.

6. Foster, Graham, et al. " The Right Tool for the Job: Switching data centre management strategies at runtime." *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013.

7. Jain, Ankur, and Joseph Pasquale. “Internet Distance Prediction Using Node-Pair Geography." *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*. IEEE, 2012.

8. Lampe, Ulrich, et al. “Assessing Latency in Cloud Gaming." *Cloud Computing and Services Science. Springer International Publishing*, 2014. 52-68.

9. Piao, Jing Tai, and Jun Yan. “A network-aware virtual machine placement and migration approach in cloud computing." *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*. IEEE, 2010.

10. Qiao, Ying, and Hussein Mouftah. “Using an overlay network to manage the renewable energy in residential areas." *Mobile and Wireless Networking (MoWNeT), 2013 International Conference on Selected Topics in. IEEE,* 2013.

11. Raghavendra, Ramya, Jorge Lobo, and Kang-Won Lee. “Dynamic graph query primitives for sdn-based cloudnetwork management." *Proceedings of the first workshop on hot topics in software defined networks*. ACM, 2012.

12. Satyanarayanan, Mahadev, et al. “The case for vm-based cloudlets in mobile computing." *Pervasive Computing, IEEE 8.4* (2009): 14-23.

13. Shen, Yuanhong, et al. “Geographic Location-Based Network-Aware QoS Prediction for Service Composition." *Web Services (ICWS), 2013 IEEE 20th International Conference on. IEEE,* 2013.

14. Tavakoli, Arsalan, et al. “Applying NOX to the Datacenter." *HotNets*. 2009.

15. Shrivastava, Vivek, et al. “Application-aware virtual machine migration in data centers." *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011.

16. Wang, Richard, Dana Butnariu, and Jennifer Rexford. “OpenFlow-based server load balancing gone wild." (2011).

17. Wette, Philip, and Holger Karl. “Which flows are hiding behind my wildcard rule?: adding packet sampling to openflow." *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. ACM,* 2013.

18. Amazon data centers around the world, Available: https://maps.google.com/maps/ms?gl=us&ie=UTF8&t=m&oe=UTF8&msa=0&msid=216293427982568528796.0004be23831c2406d9a8e&dg=feature

19. Cisco Global Cloud Index: Forecast and Methodology, 2012–2017

20. Map of all google data centers locations, Available: http://royal.pingdom.com/2008/04/11/map-of-all-google-data-center-locations/

21. 10 Great Cloud Applications and Service, Available: http://www.salesforce.com/uk/socialsuccess/cloud-computing/10-great-cloud-applications-services-smes.jsp

22. Leading countries based on number of facebook users, Available: http://www.statista.com/statistics/268136/top-15-countries-based-on-number-of-facebook-users/

23. Exploring use of Twitter around the world, Available: http://www.sysomos.com/insidetwitter/geography/

24. A guide to Chinese social media, Available: http://blog.webcertain.com/a-guide-to-chinese-social-media-what-to-do-and-how-to-do-it/17/02/2014/?gclid=CNre4peA58ACFahaMgod5FYAnw

25. Network Bandwidth and Latency, Available: http://compnetworking.about.com/od/speedtests/a/network_latency.htm

26. Retrieving country information from IP address, Available: http://www.codeproject.com/Articles/28363/How-to-convert-IP-address-to-country-name

27. Definition of Voronoi Diagram, Available: http://en.wikipedia.org/wiki/Voronoi_diagram

28. Hoang T. Dinh, Chonho Lee, Dusit Niyato and Ping Wang, “A survey of mobile cloud computing: architecture, application and approaches”, Wireless communication and Mobile computing, Wiley Online Library, 2011.

29. Our Mobile Planet, Available: http://think.withgoogle.com/mobileplanet/en/downloads/

30. K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C.-Z. Xu, and A. Y. Zomaya, "Quantitative Comparisons of the State of the Art Data Center Architecture," Concurrency and Computation: Practice and Experience, vol. 25, no. 12, pp. 1771-1783, 2013.

31. Data Center Network Architectures, Available: http://en.wikipedia.org/wiki/Data_center_network_architectures

32. A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. “Forwarding and Control Element Separation (ForCES) Protocol Specification”, RFC 5810 (Proposed Standard),March 2010.

33. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks”, ACM SIGCOMM Computer Communication Review, 38(2):69–74, 2008.

34. Business White Paper from HP, “Prepare for Software-Defined Networking”, Available: www2.hp.com.

35. Nikhil Handigol, Srinivasan Seetharaman, Mario Flajslik, Nick McKeown, Ramesh Johari , "Plug-n-Serve: Load-balancing Web Traffic using OpenFlow", Sigcomm 2009.

36. H OPPS, C. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992, IETF, 2000.

37. Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yannis Yakoumis, Puneet Sharma, Sujata Banerjee, Nick McKeown, "ElasticTree: Saving Energy in Data Center Networks", NSDI 2010.

38. Mohammad Al-Fares, Sivasankar Radhakrishnan et. al. "Hedera: Dynamic Flow Scheduling for Data Center Networks", NDSI 2011.

39. Wang, Xiang, et al., "LiveCloud: A lucid orchestrator for cloud datacenters", IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), 2012.

40. Md Faizul Bari et al., "PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks", SDN for Future Networks and Services (SDN4FNS), IEEE, 2013.

41. Claudia, and Riccardo Lancellotti, "Automatic virtual machine clustering based on bhattacharyya distance for multi-cloud systems." Proceeding of the 2013 international workshop on Multi-cloud applications and federated clouds. ACM, 2013.

42. Hirofuchi et. al., A Live Storage Migration Mechanism over WAN for Relocatable Virtual Machine Services on Clouds", International Symposium on the Cluster Computing and the Grid, 2009.

43. Gaston Keller, Hanan Lutfiyya. "Replication and Migration as Resource Management Mechanisms for Virtualized Environments", Sixth International Conference on Autonomic and Autonomous Systems, 2010.

44. Hadi Goudarzi and Masud Pedram. “Energy-Efficient Virtual Machine Replication and Placement in a Cloud Computing System”, 5th International Conference on Cloud Computing, IEEE, 2012.

45. What is Cloud Hosting? Available: http://www.interoute.com/what-cloud-hosting

46. Yuyang Du, Hongliang Yu. “ Paratus: Instantaneous Failover via Virtual Machine Replication ”, Eigth International Conference on Grid and Cooperative Computing, GCC, 2009.

47. Geolocation Providers, Available: http://whatismyipaddress.com/geolocation-providers

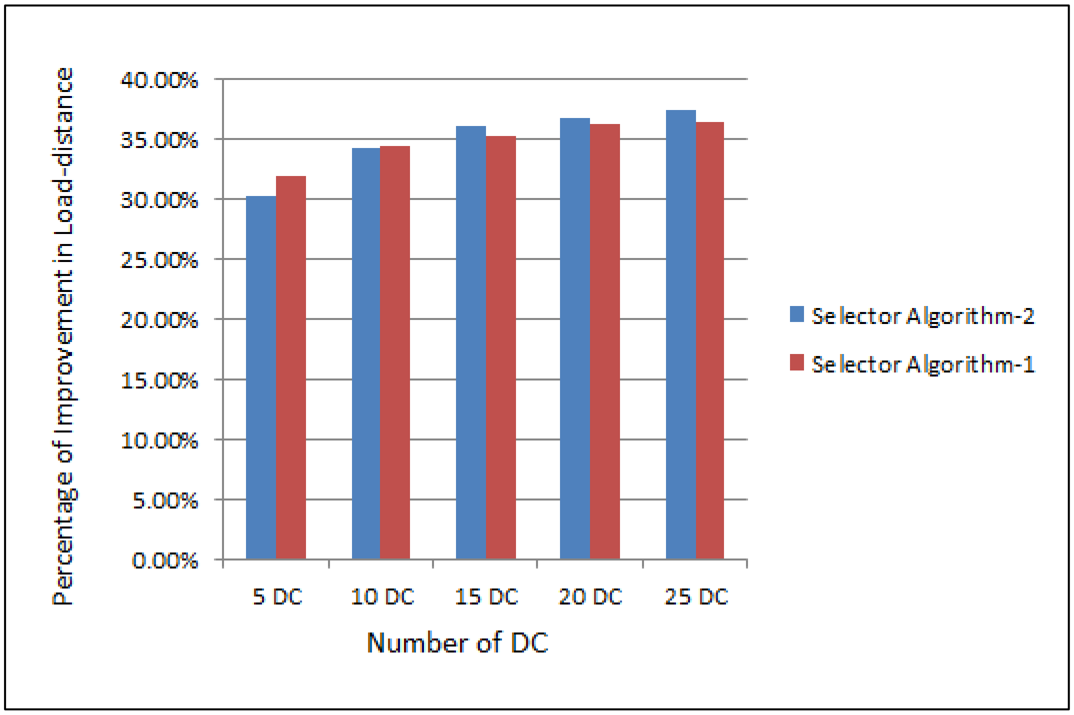
48. Mobile Applications Futures 2013-2017, Available: http://www.portioresearch.com/en/major-reports/current-portfolio/mobile-applications-futures-2013-2017.aspx

49. How Content Delivery Network Works, Available: http://www.nczonline.net/blog/2011/11/29/how-content-delivery-networks-cdns-work/

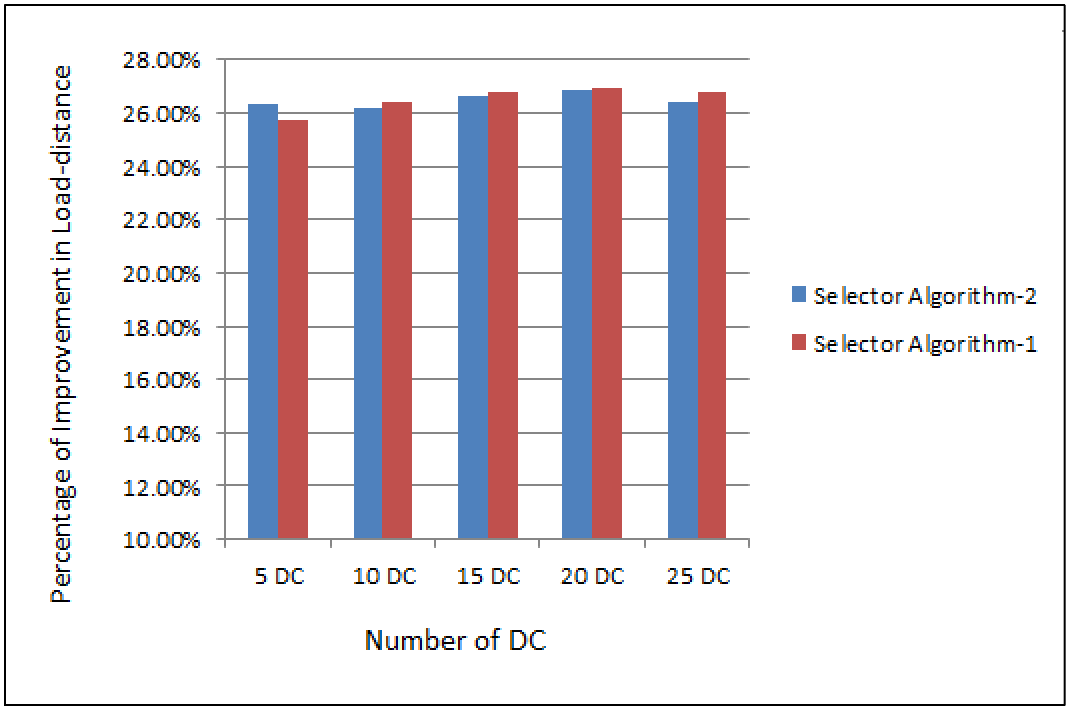
50. Naïve Forecast Definition, Available : http://www.scmfocus.com/demandplanning/2012/03/naive-forecast/

Appendices

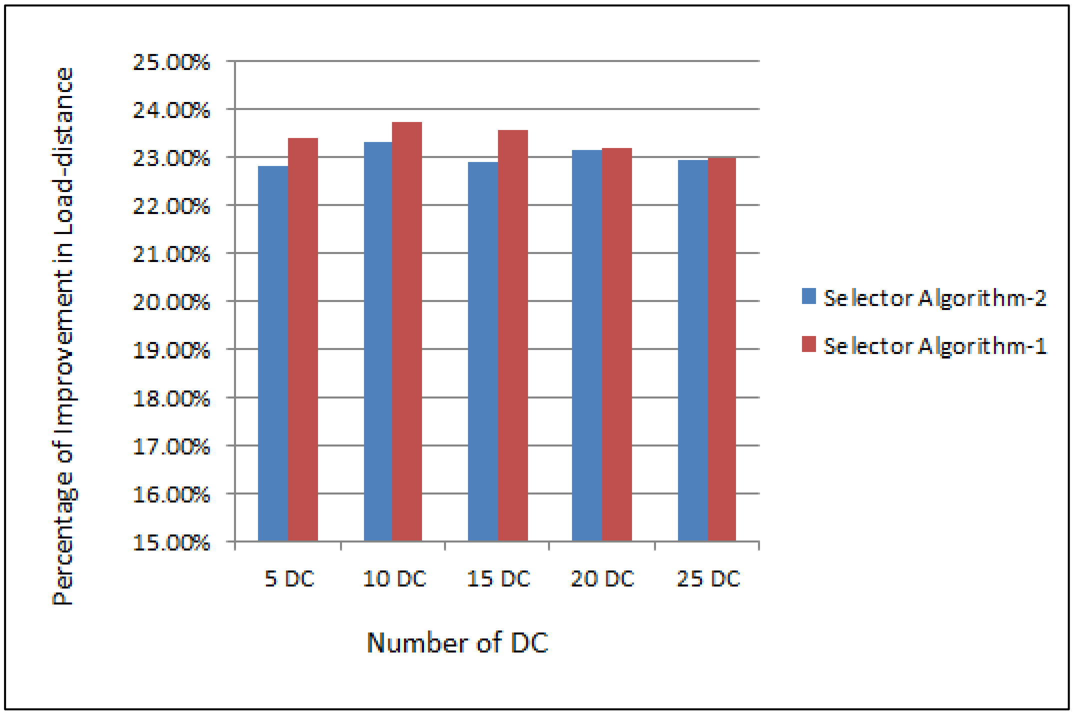
Appendix A: Percentage of Expected Improvement for Migration Algorithms



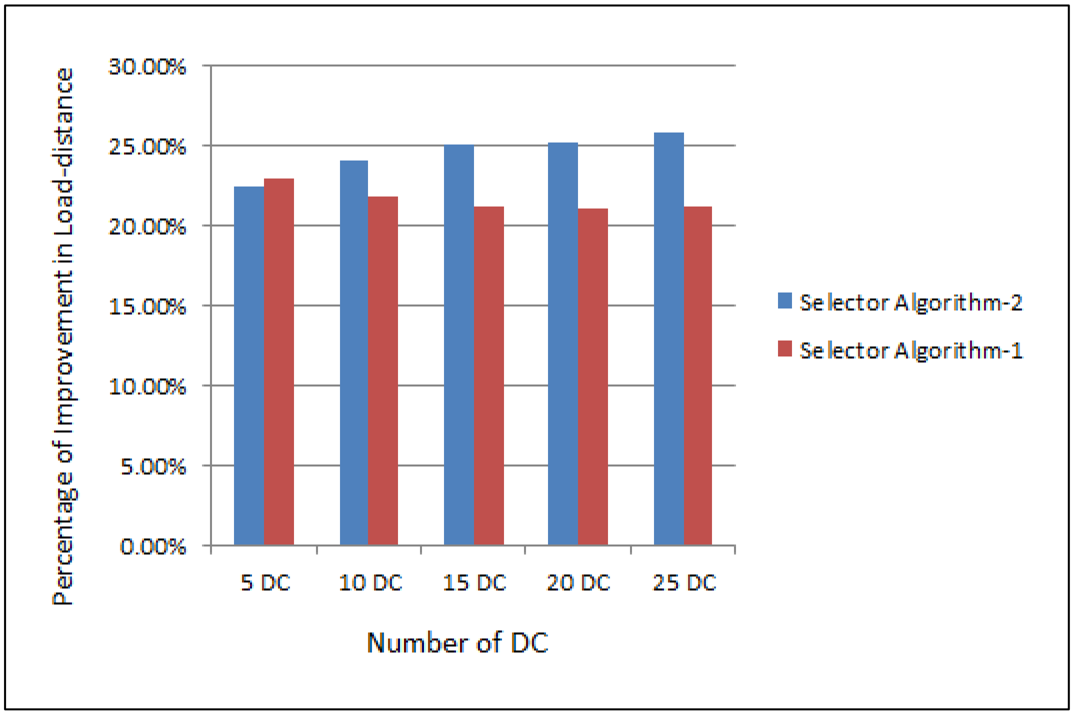
Appendix A Figure 1: 5000VMs/DC, 20K to 30K flows per region, Scenario 1



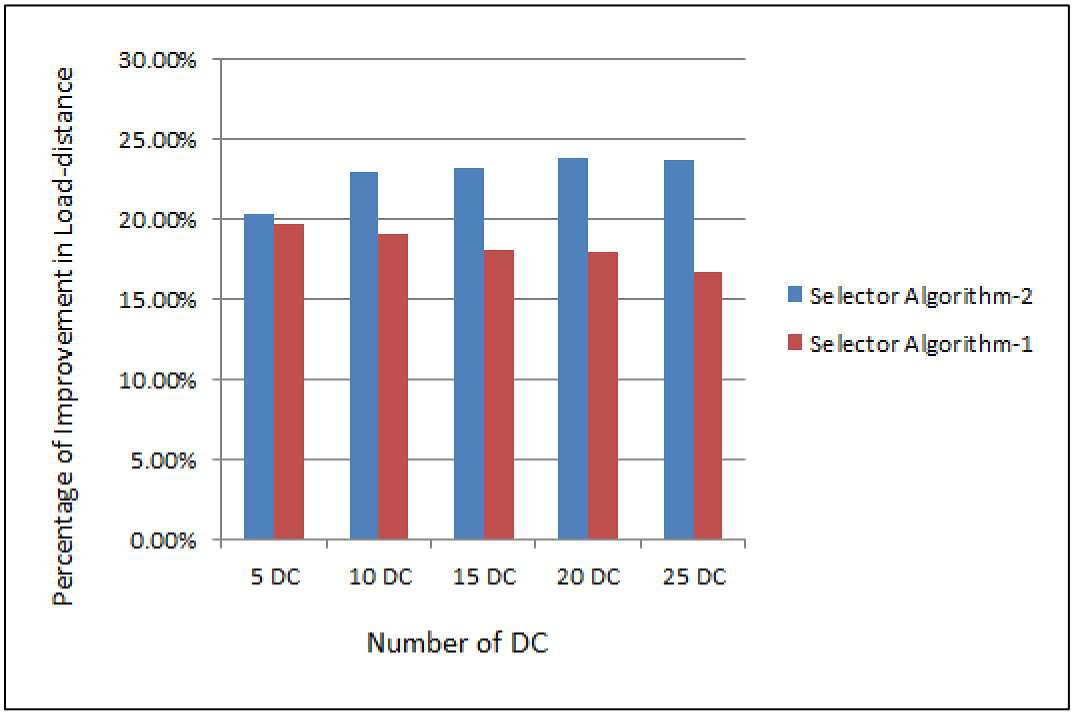
Appendix A Figure 2: 2500VMs/DC, 20K to 30K flows per region, Scenario 1



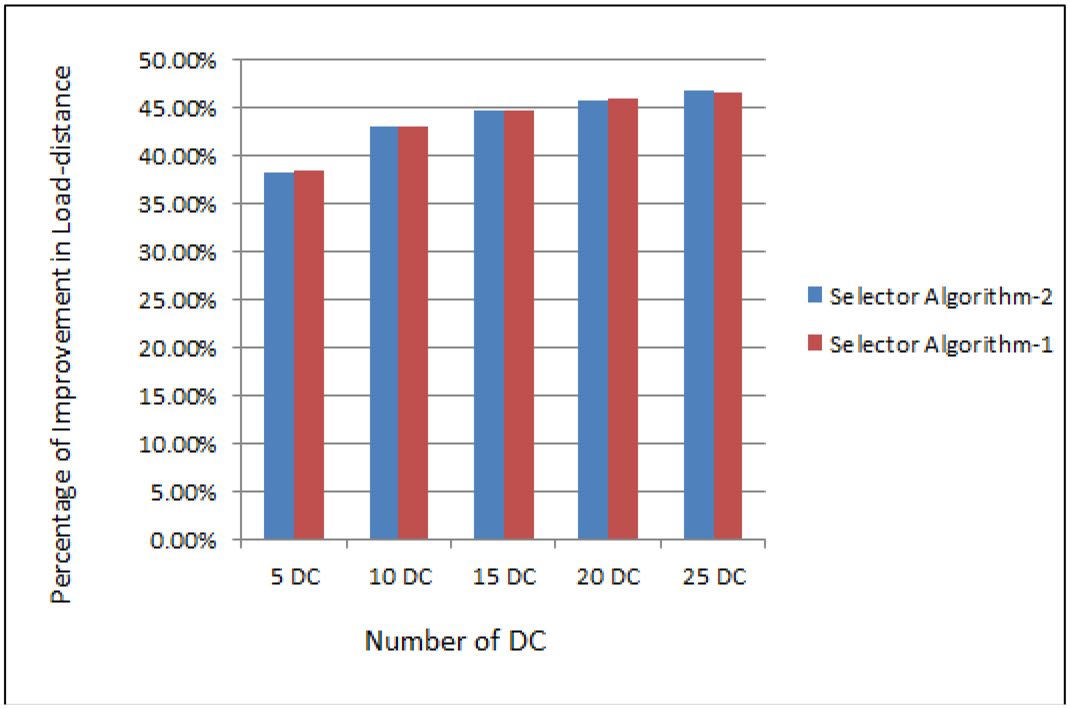
Appendix A Figure 3: 1500VMs/DC, 20K to 30K flows per region, Scenario 1



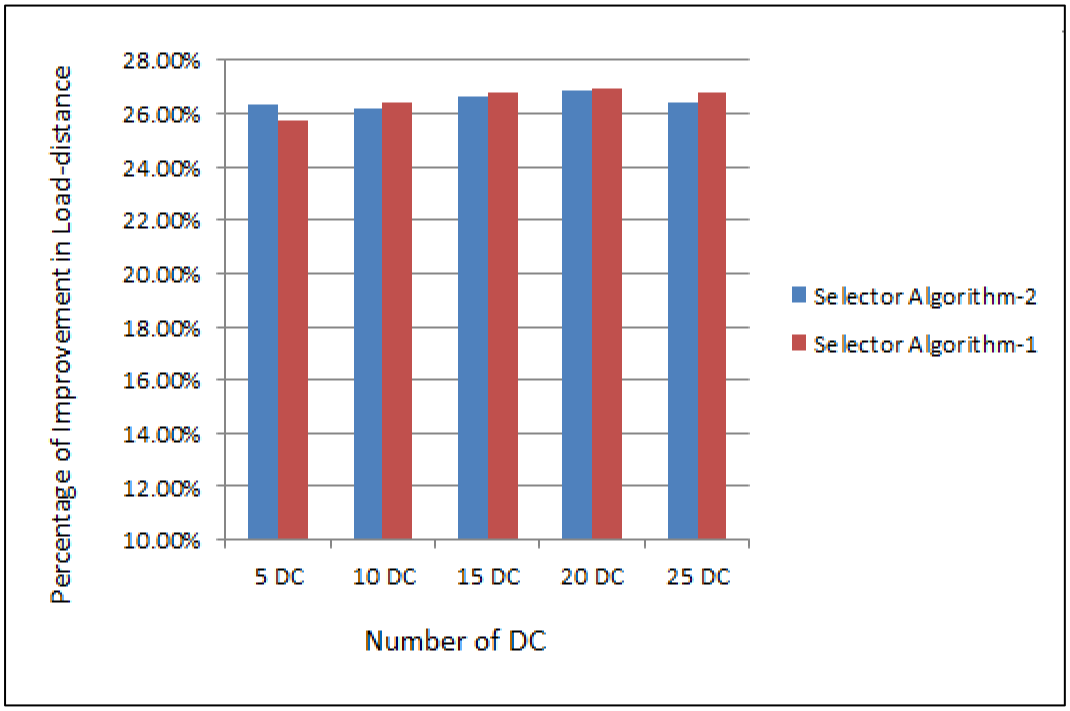
Appendix A Figure 4: 1000VMs/DC, 20K to 30K flows per region, Scenario 1



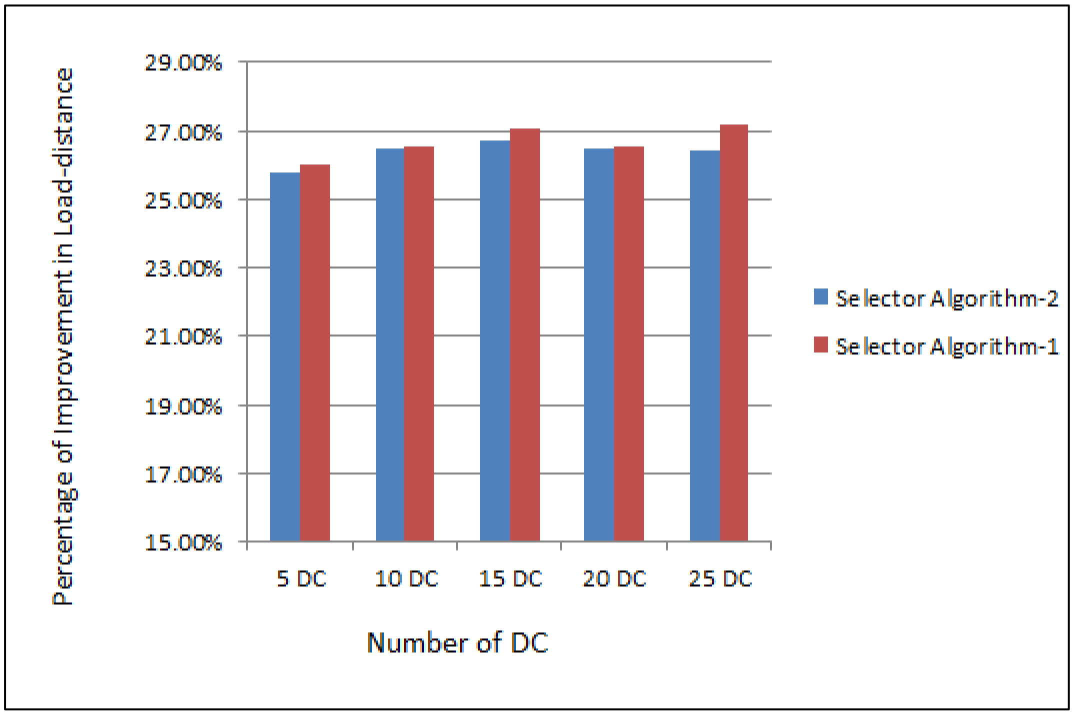
Appendix A Figure 5: 200VMs/DC, 20K to 30K flows per region, Scenario 1



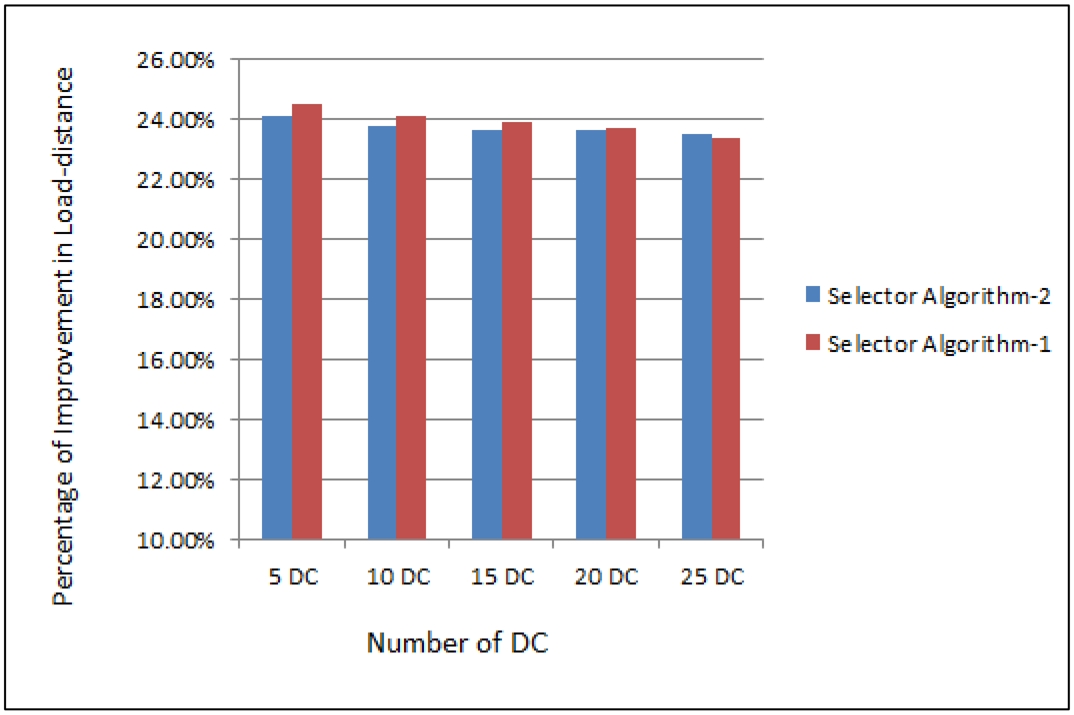
Appendix A Figure 6: 5000VMs/DC, 10K to 20K flows per region, Scenario 1



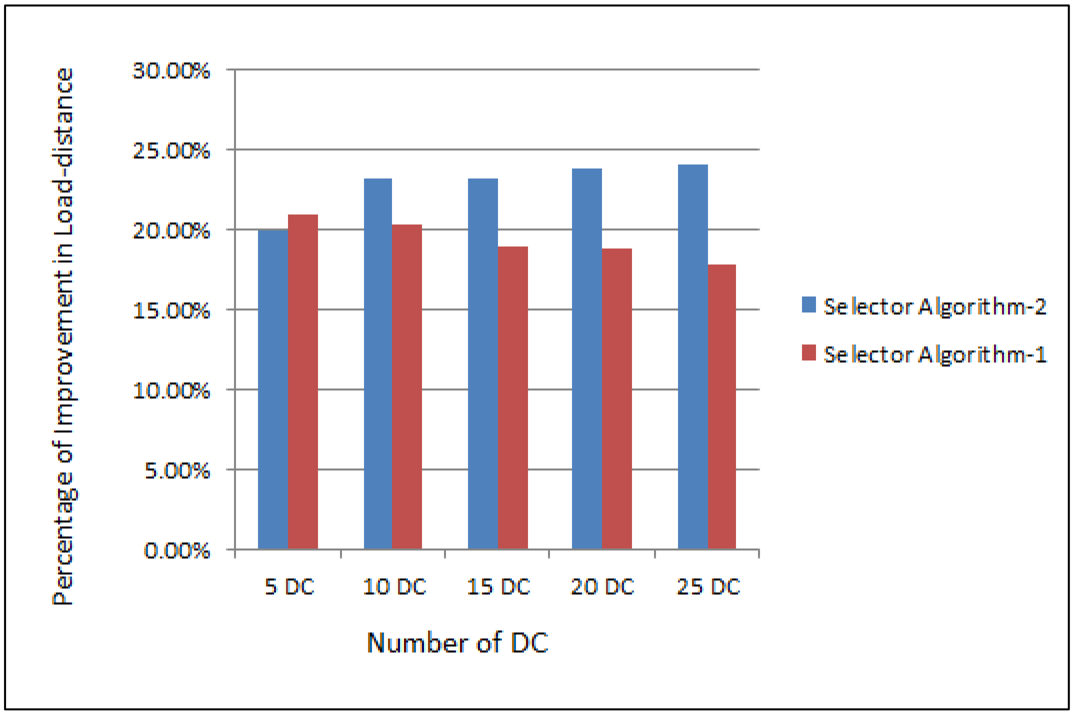
Appendix A Figure 7: 2500VMs/DC, 10K to 20K flows per region, Scenario 1



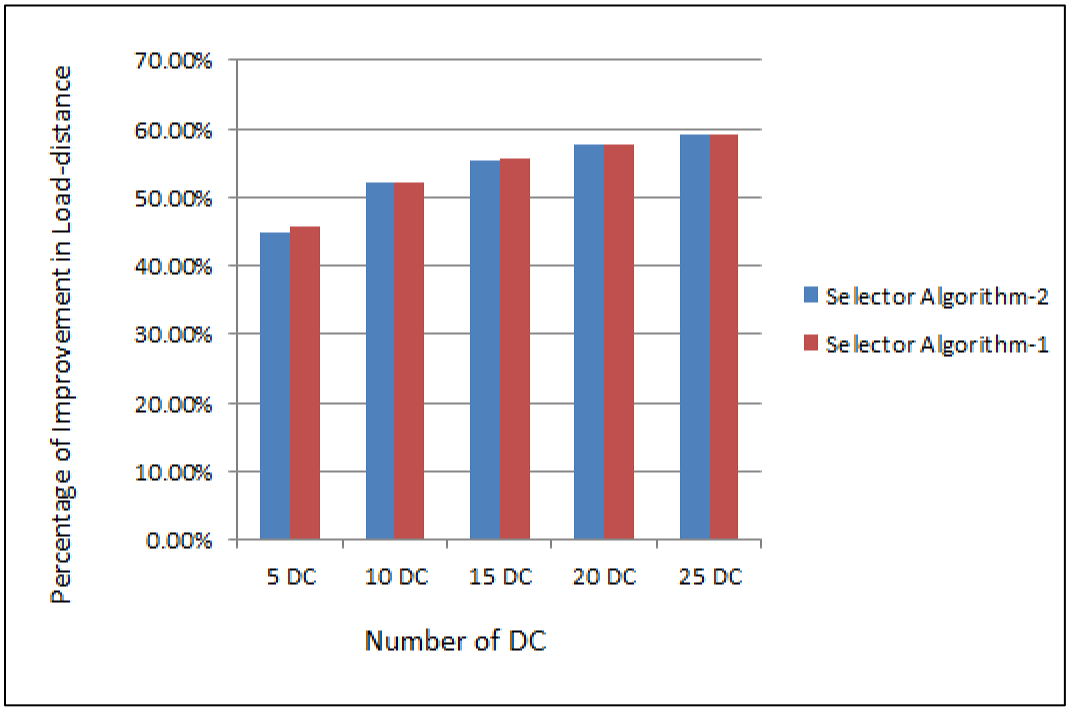
Appendix A Figure 8: 1500VMs/DC, 10K to 20K flows per region, Scenario 1



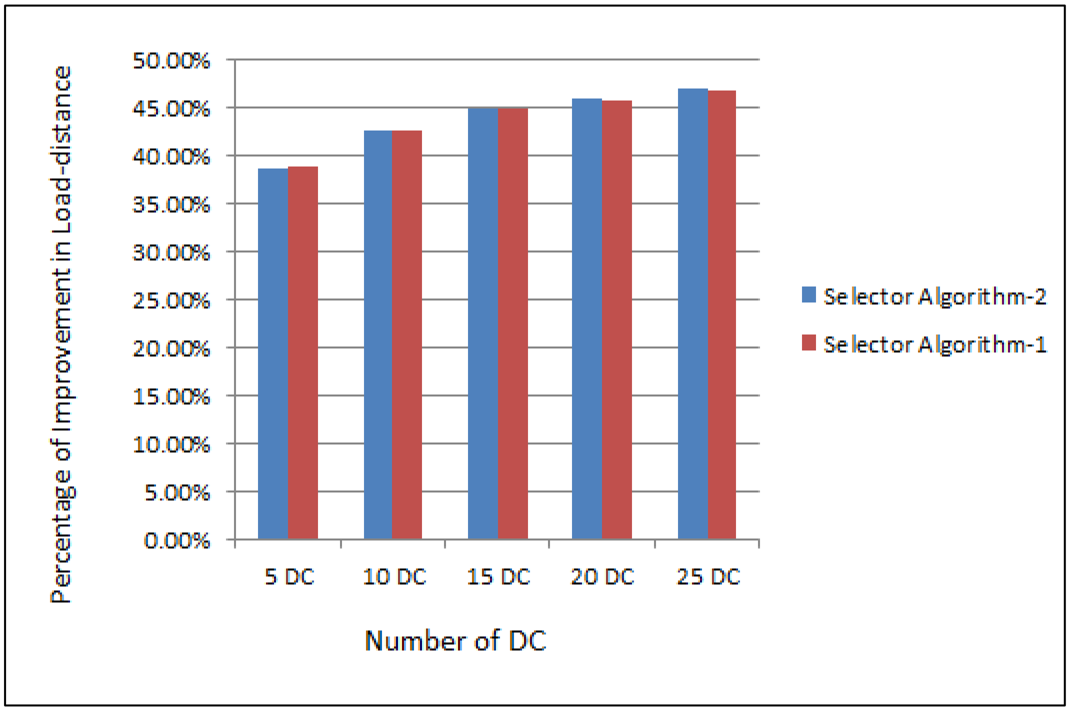
Appendix A Figure 9: 1000VMs/DC, 10K to 20K flows per region, Scenario 1



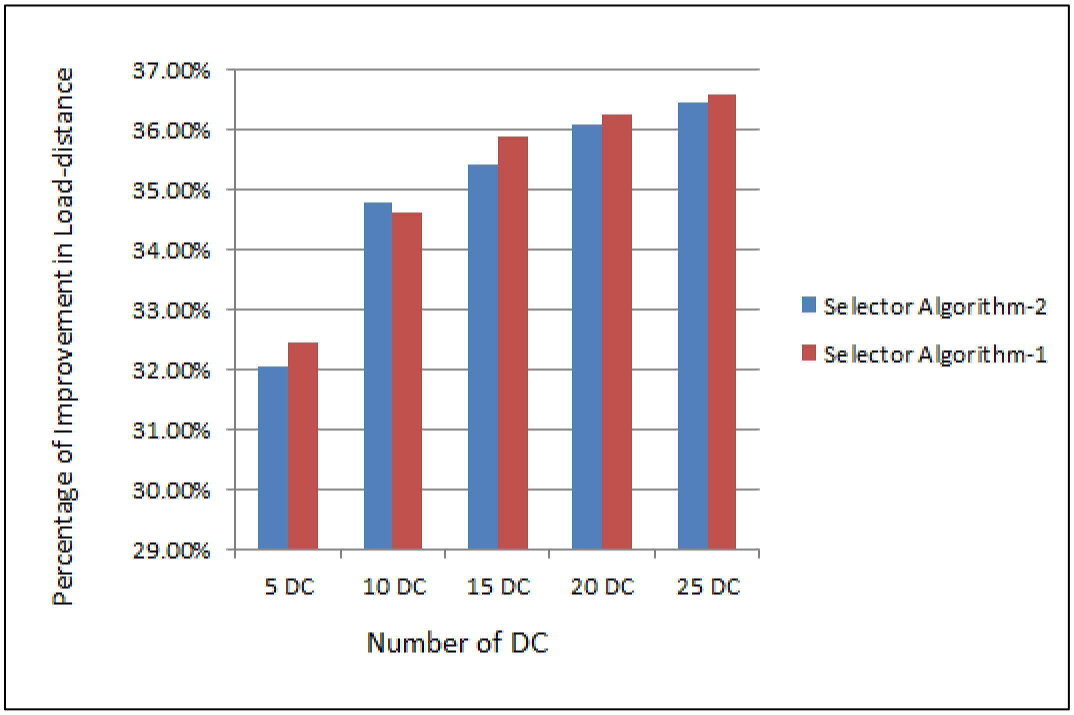
Appendix A Figure 10: 200VMs/DC, 10K to 20K flows per region, Scenario 1



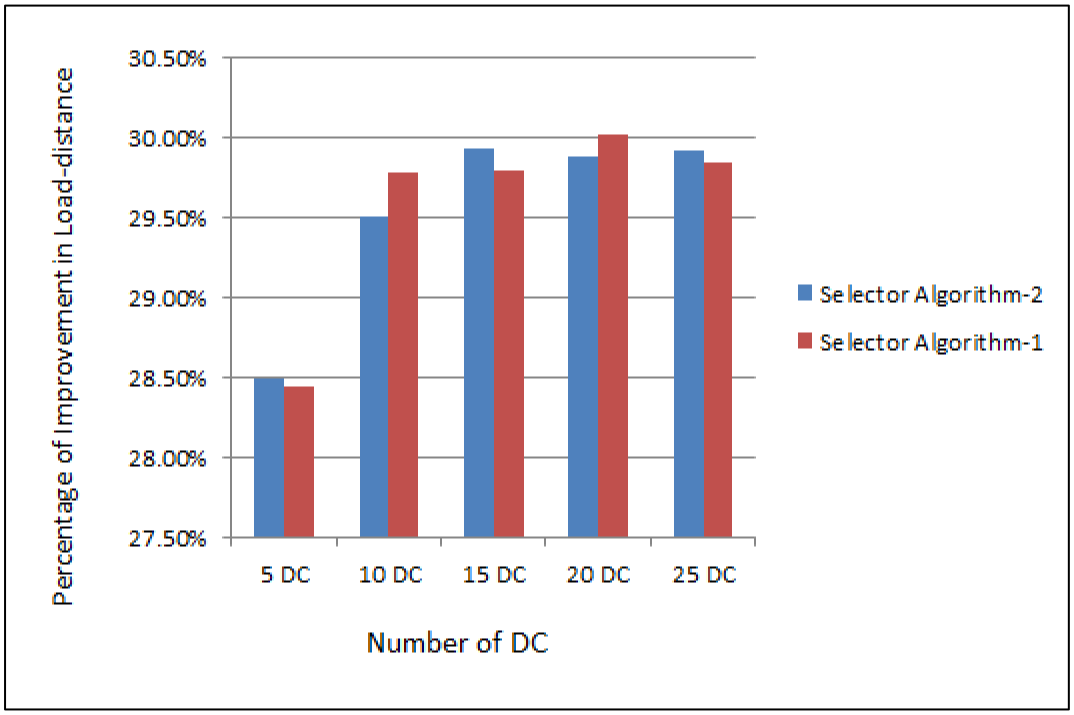
Appendix A Figure 11: 5000VMs/DC, 5K to 10K flows per region, Scenario 1



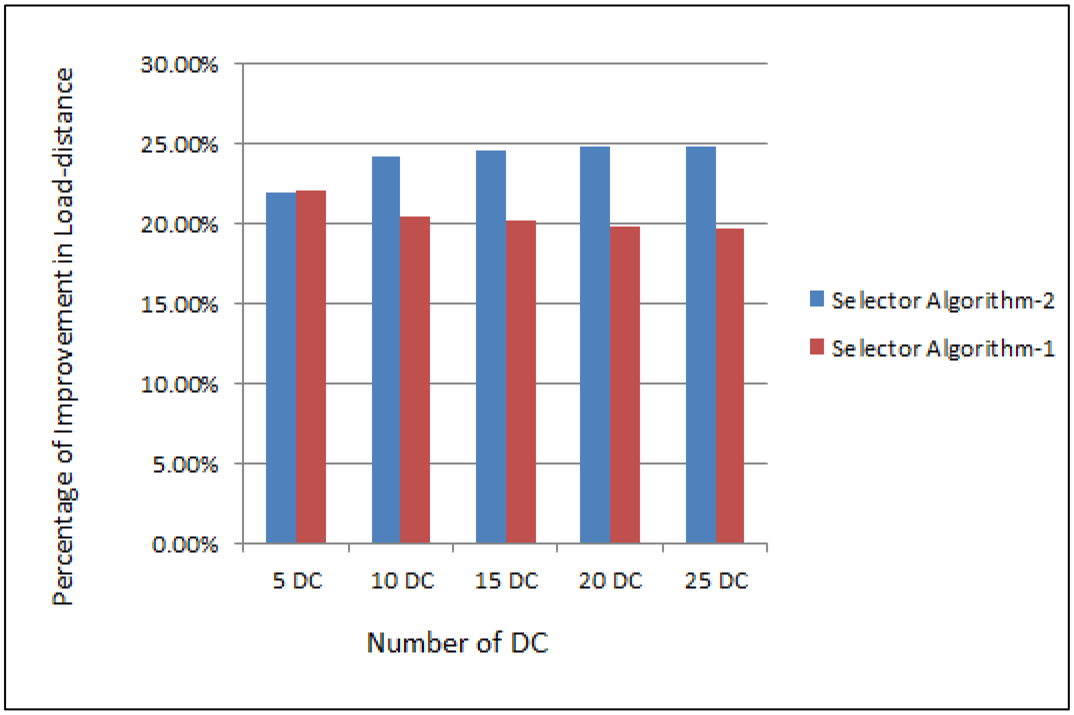
Appendix A Figure 12: 2500VMs/DC, 5K to 10K flows per region, Scenario 1



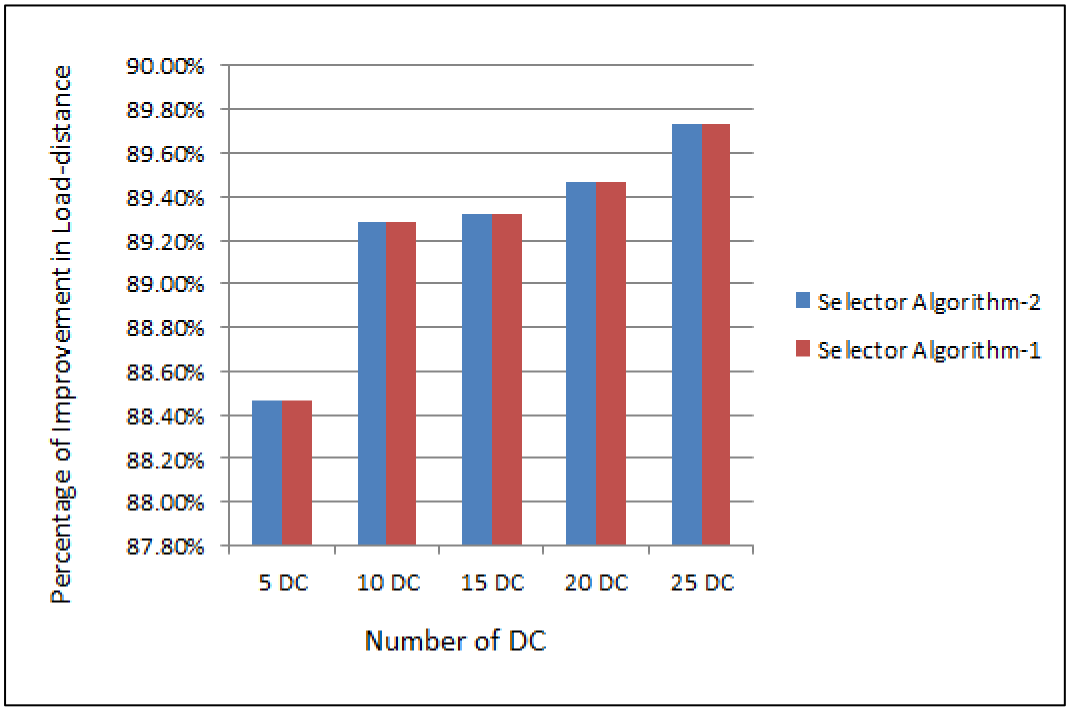
Appendix A Figure 13: 1500VMs/DC, 5K to 10K flows per region, Scenario 1



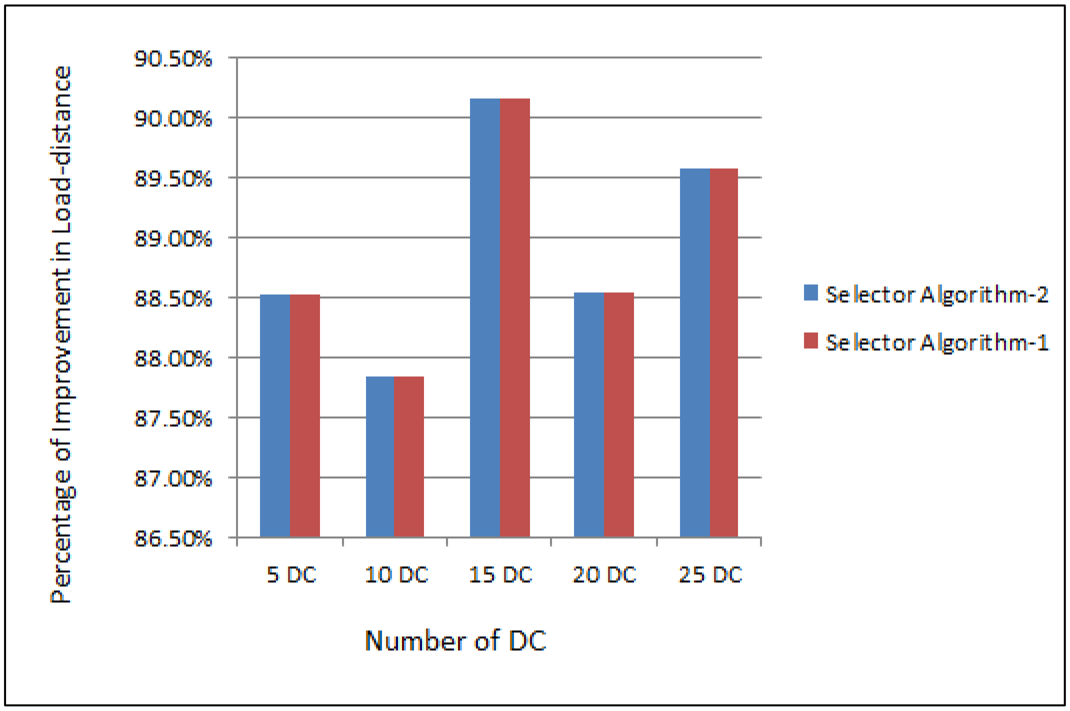
Appendix A Figure 14: 1000VMs/DC, 5K to 10K flows per region, Scenario 1



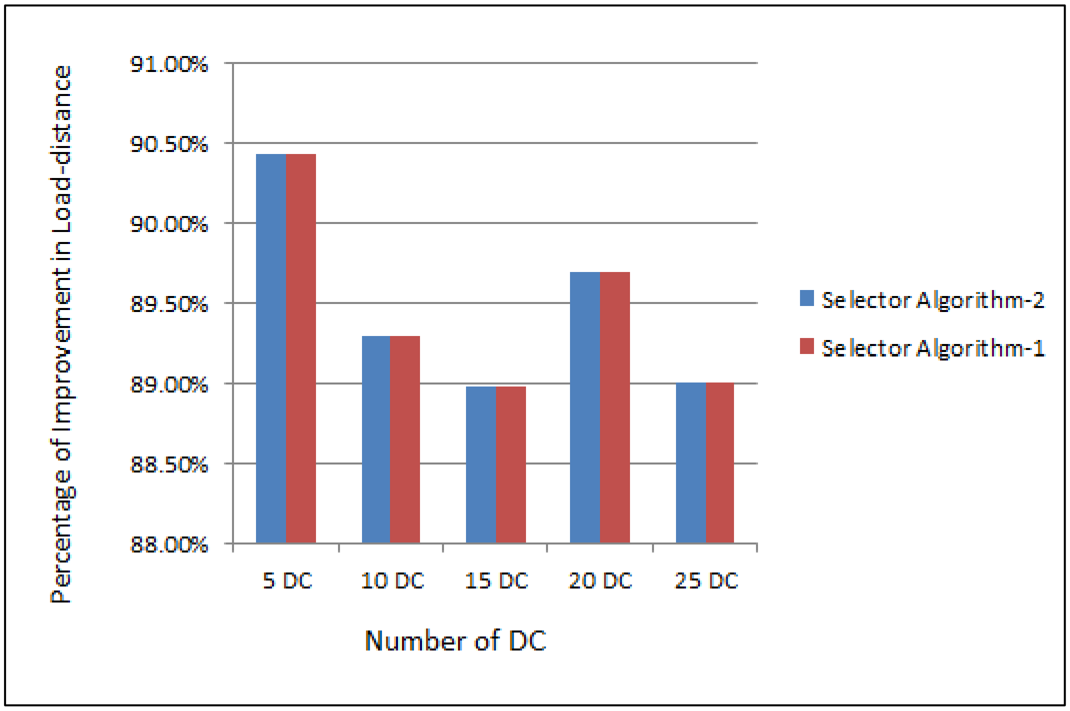
Appendix A Figure 15: 200VMs/DC, 5K to 10K flows per region, Scenario 1



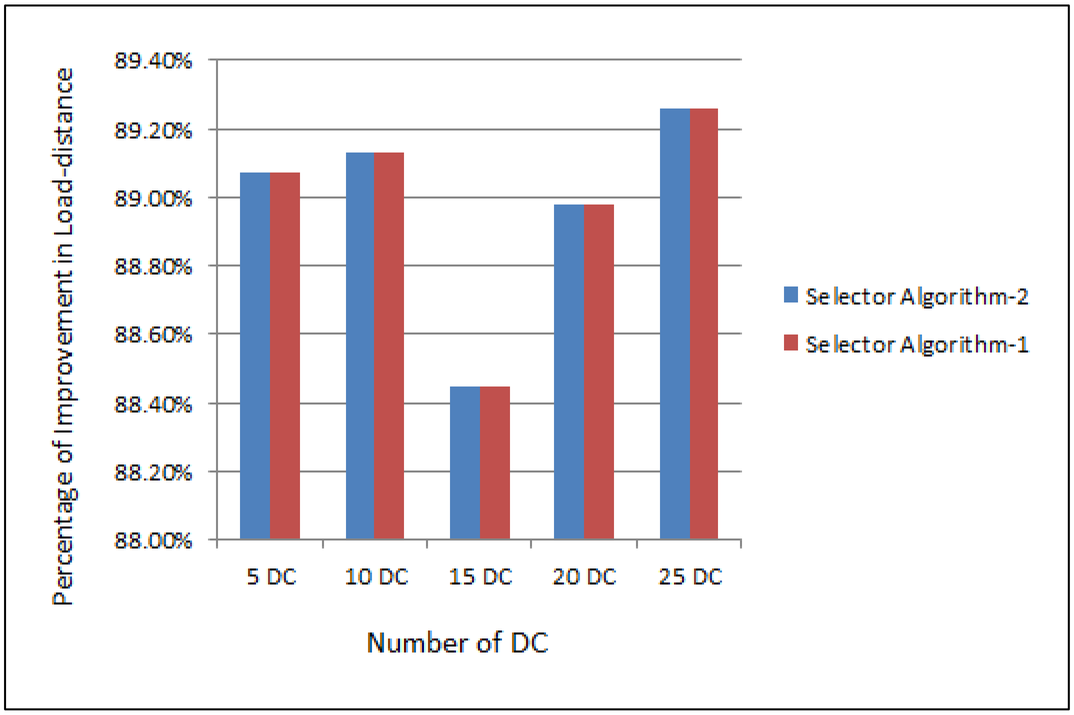
Appendix A Figure 16: 5000VMs/DC, 20K to 30K flows per region, Scenario 2



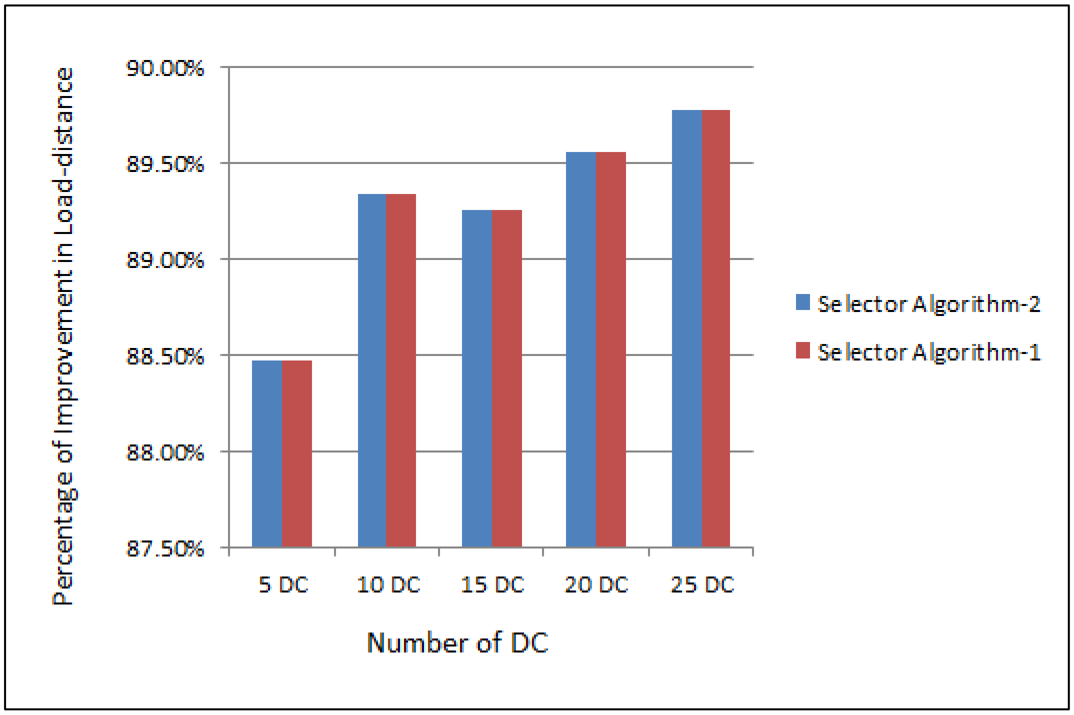
Appendix A Figure 17: 2500VMs/DC, 20K to 30K flows per region, Scenario 2



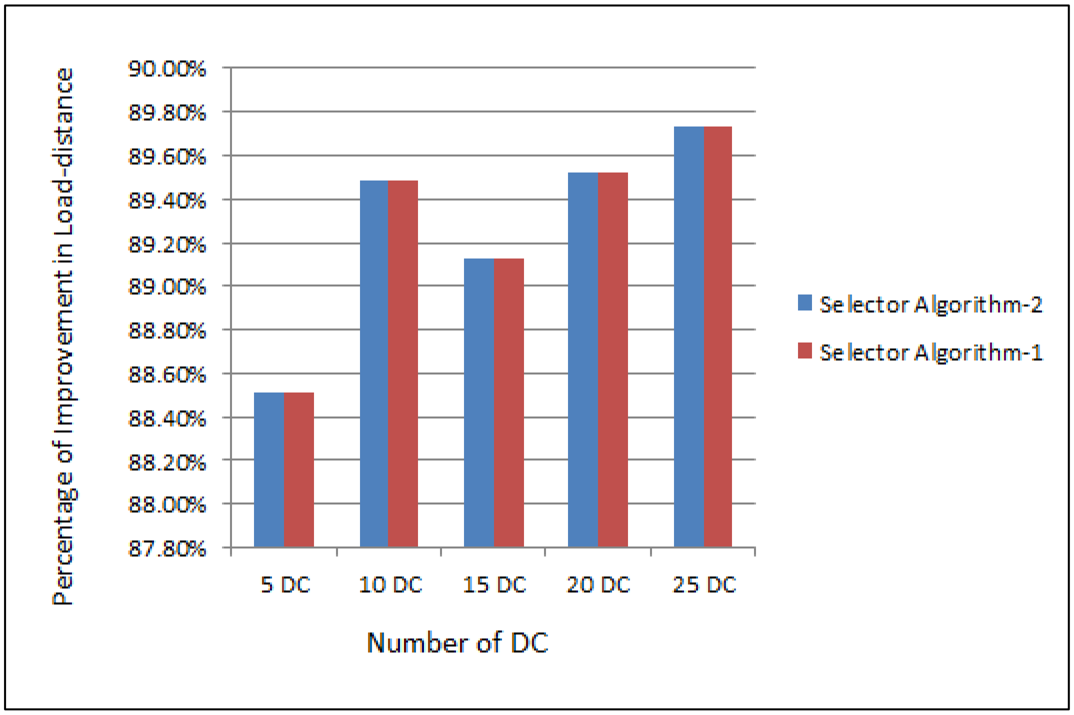
Appendix A Figure 18: 1500VMs/DC, 20K to 30K flows per region, Scenario 2



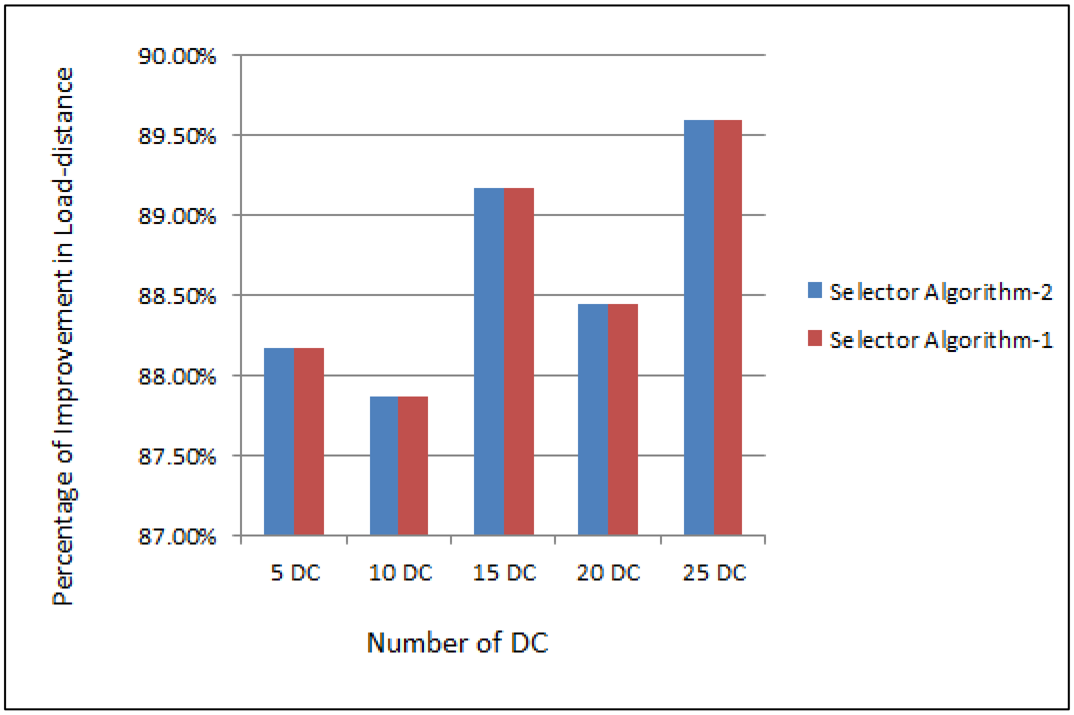
Appendix A Figure 19: 1000VMs/DC, 20K to 30K flows per region, Scenario 2



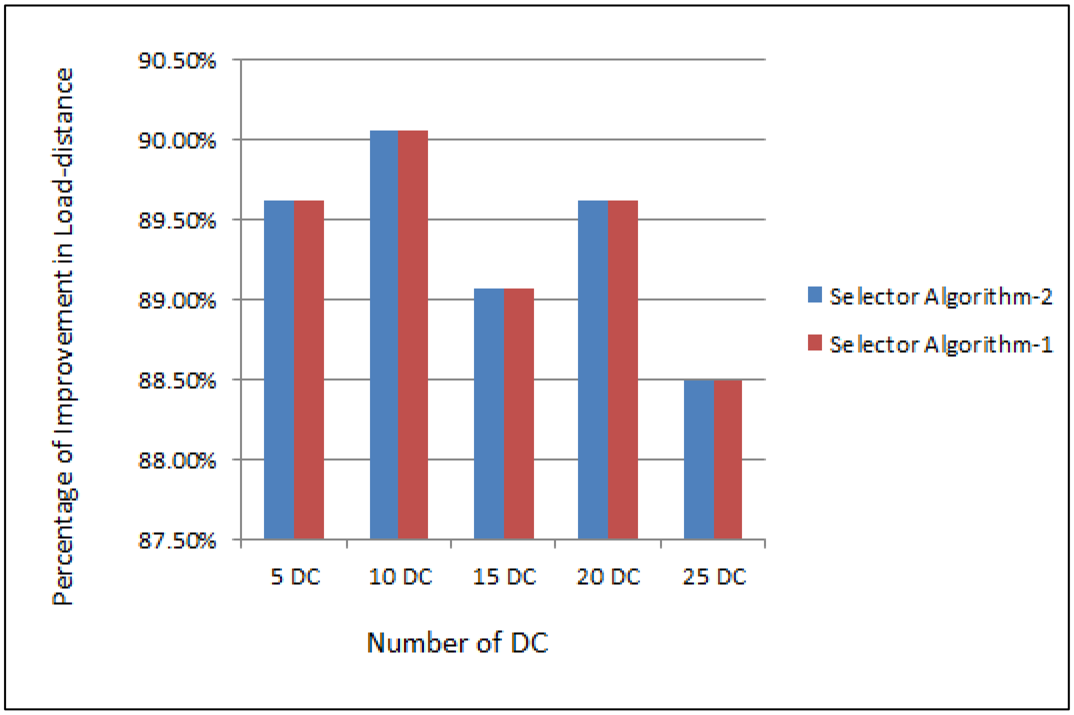
Appendix A Figure 20: 200VMs/DC, 20K to 30K flows per region, Scenario 2



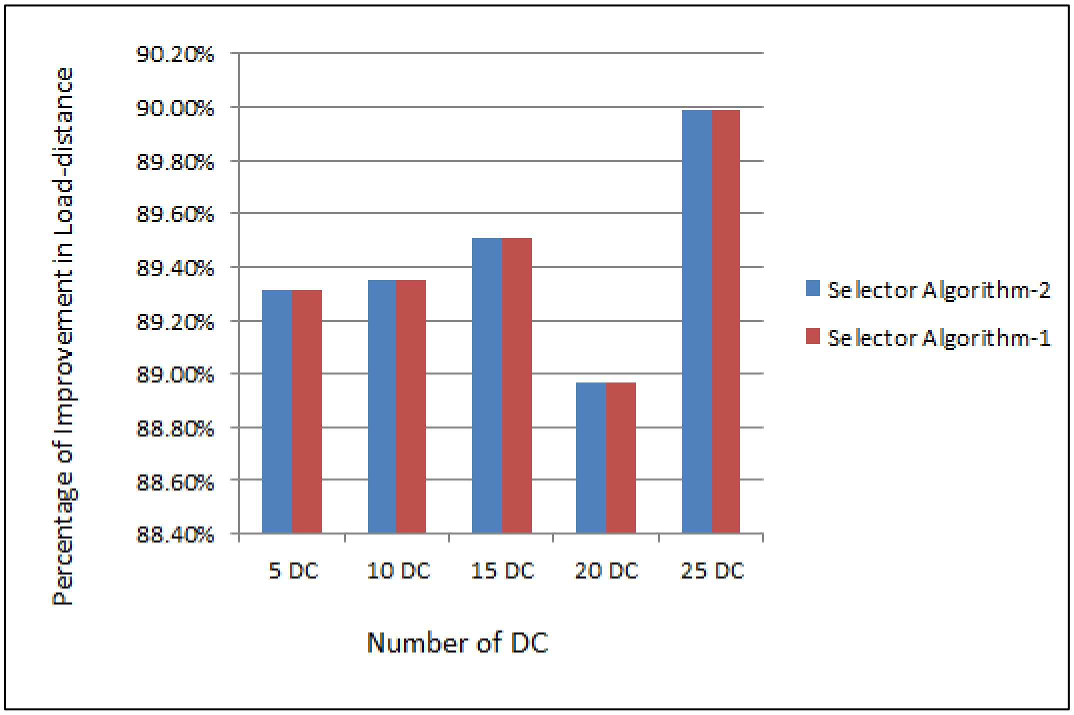
Appendix A Figure 21: 5000VMs/DC, 10K to 20K flows per region, Scenario 2



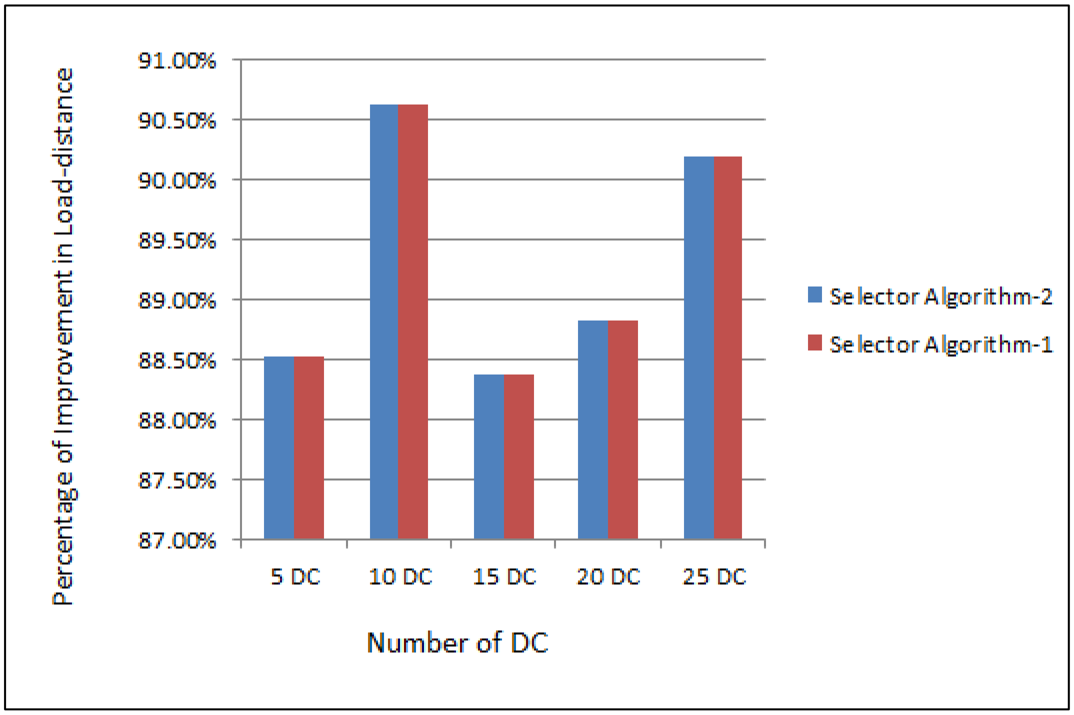
Appendix A Figure 22: 2500VMs/DC, 10K to 20K flows per region, Scenario 2



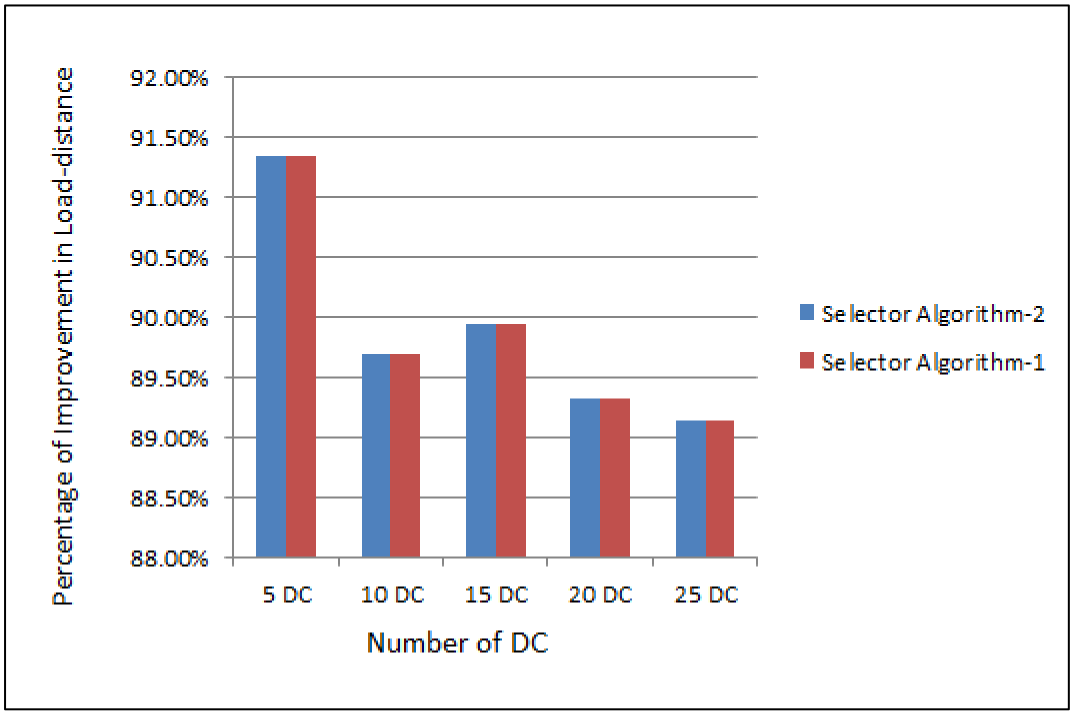
Appendix A Figure 23: 1500VMs/DC, 10K to 20K flows per region, Scenario 2



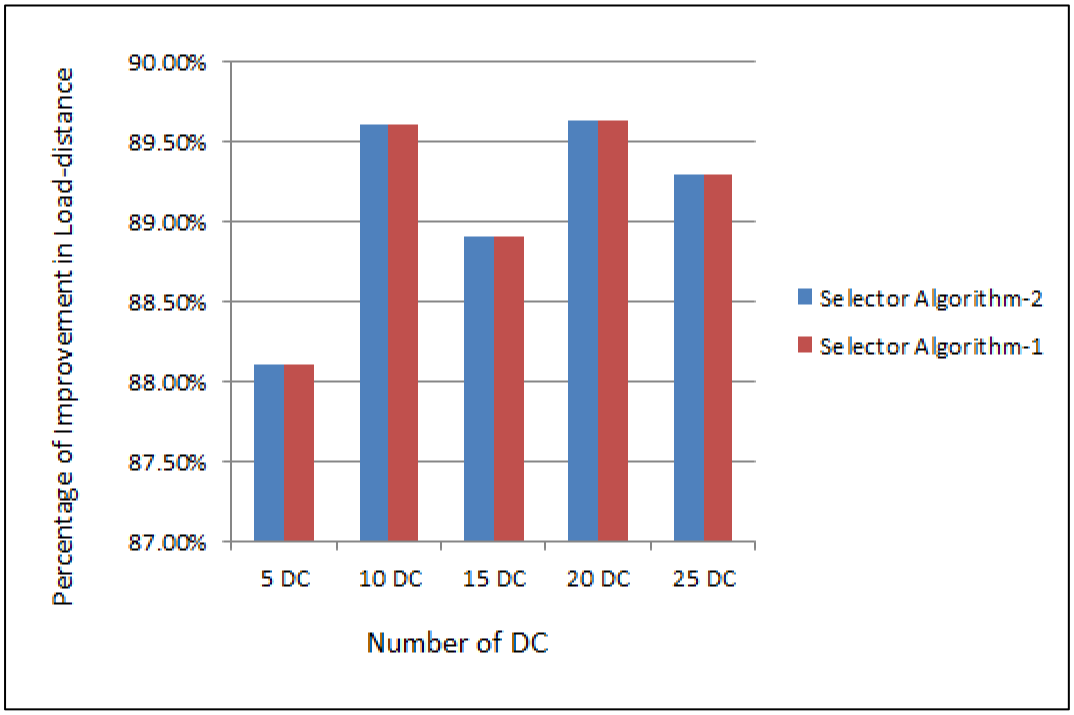
Appendix A Figure 24: 1000VMs/DC, 10K to 20K flows per region, Scenario 2



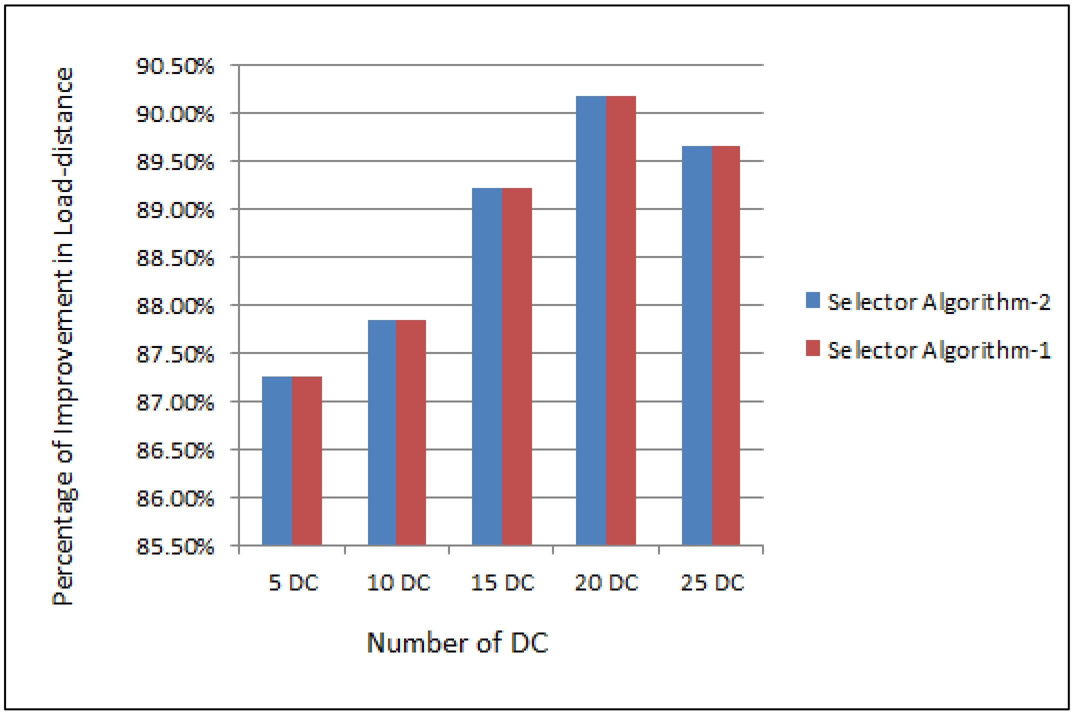
Appendix A Figure 25: 200VMs/DC, 10K to 20K flows per region, Scenario 2



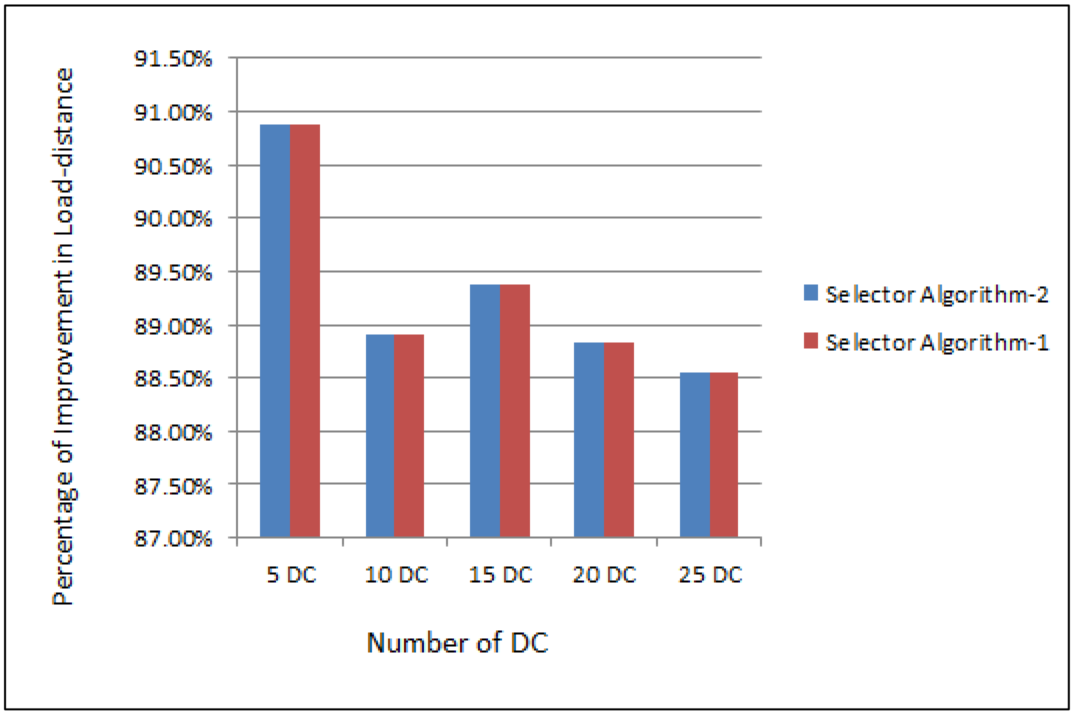
Appendix A Figure 26: 5000VMs/DC, 5K to 10K flows per region, Scenario 2



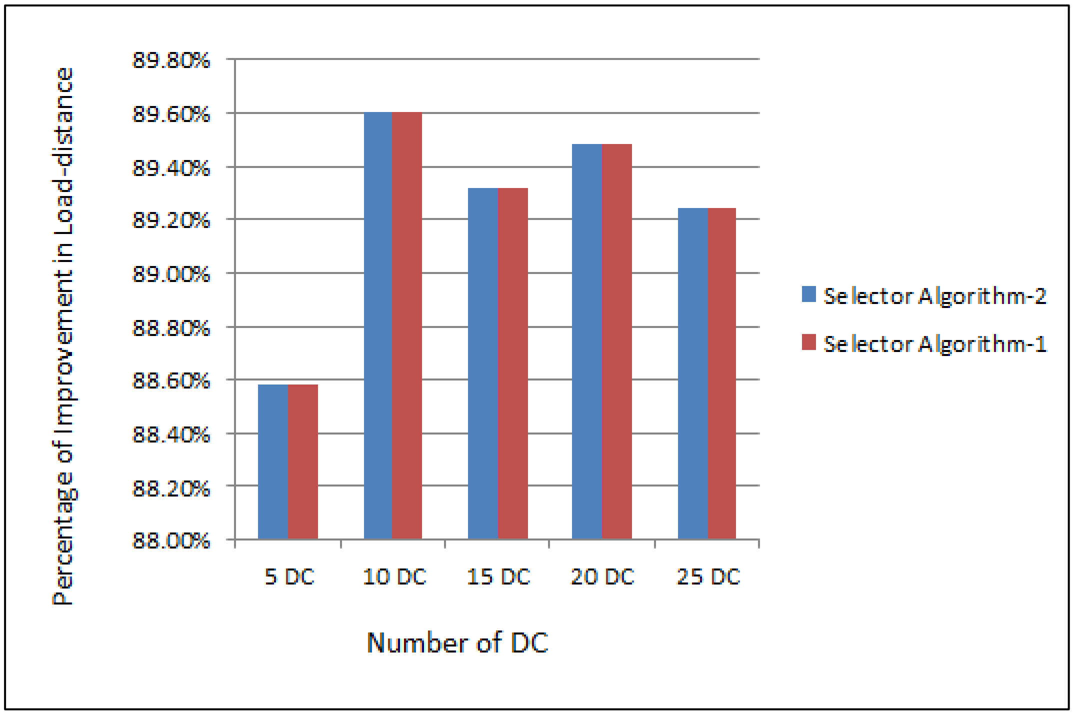
Appendix A Figure 27: 2500VMs/DC, 5K to 10K flows per region, Scenario 2



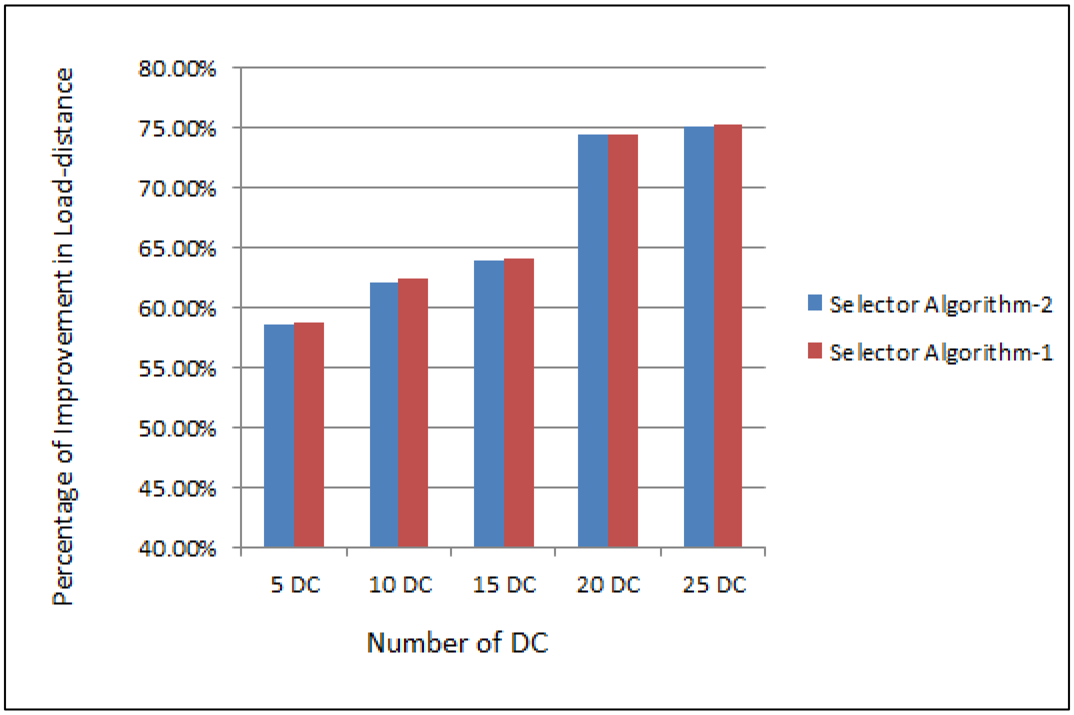
Appendix A Figure 28: 1500VMs/DC, 5K to 10K flows per region, Scenario 2



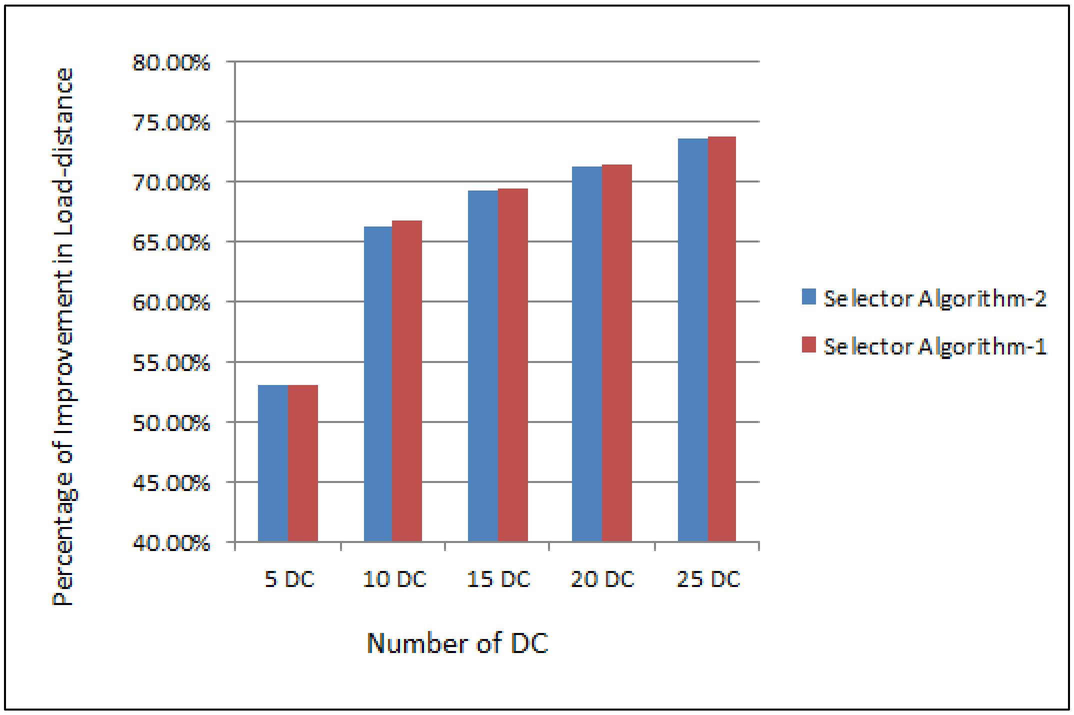
Appendix A Figure 29: 1000VMs/DC, 5K to 10K flows per region, Scenario 2



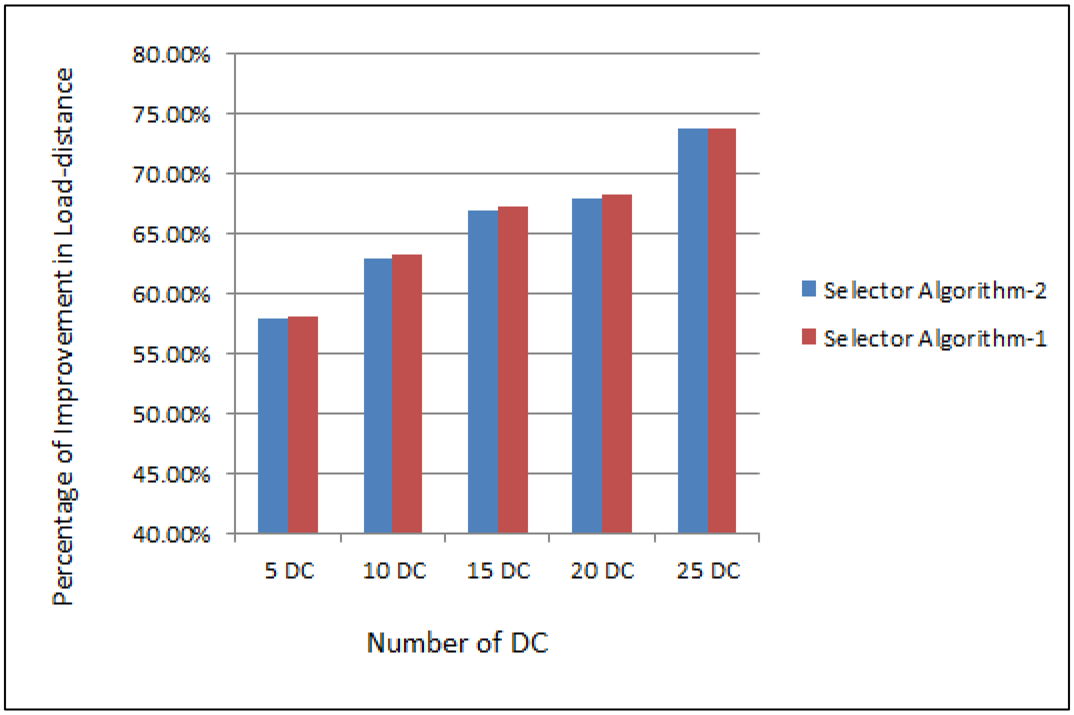
Appendix A Figure 30: 200VMs/DC, 5K to 10K flows per region, Scenario 2



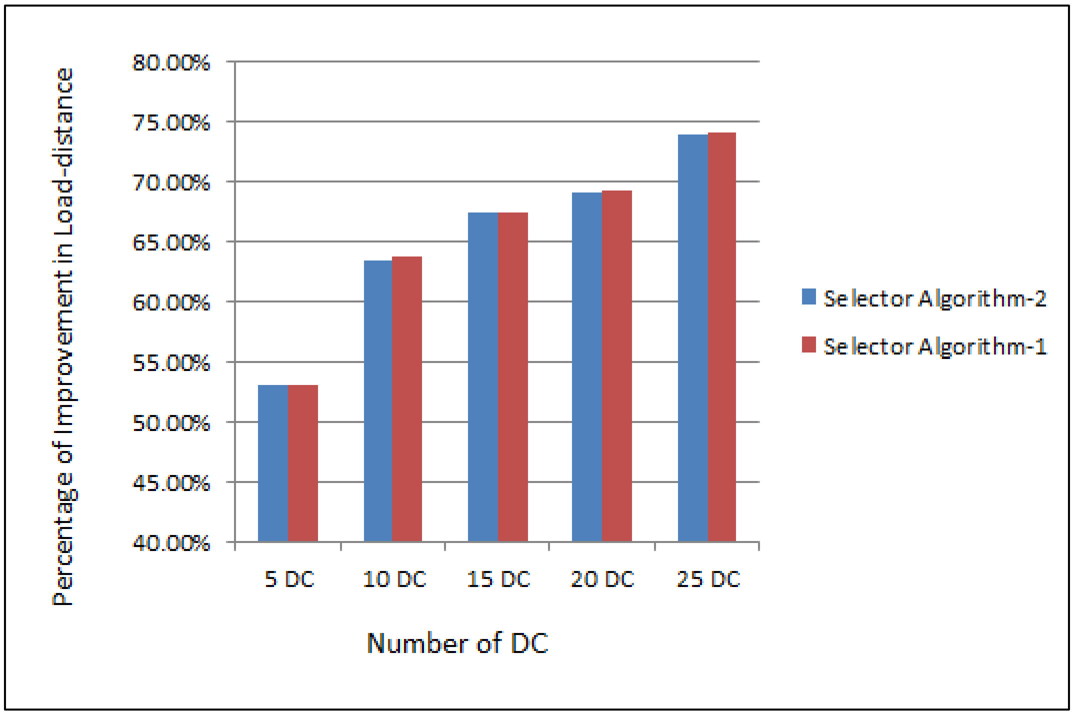
Appendix A Figure 31: 5000VMs/DC, 20K to 30K flows per region, Scenario 3



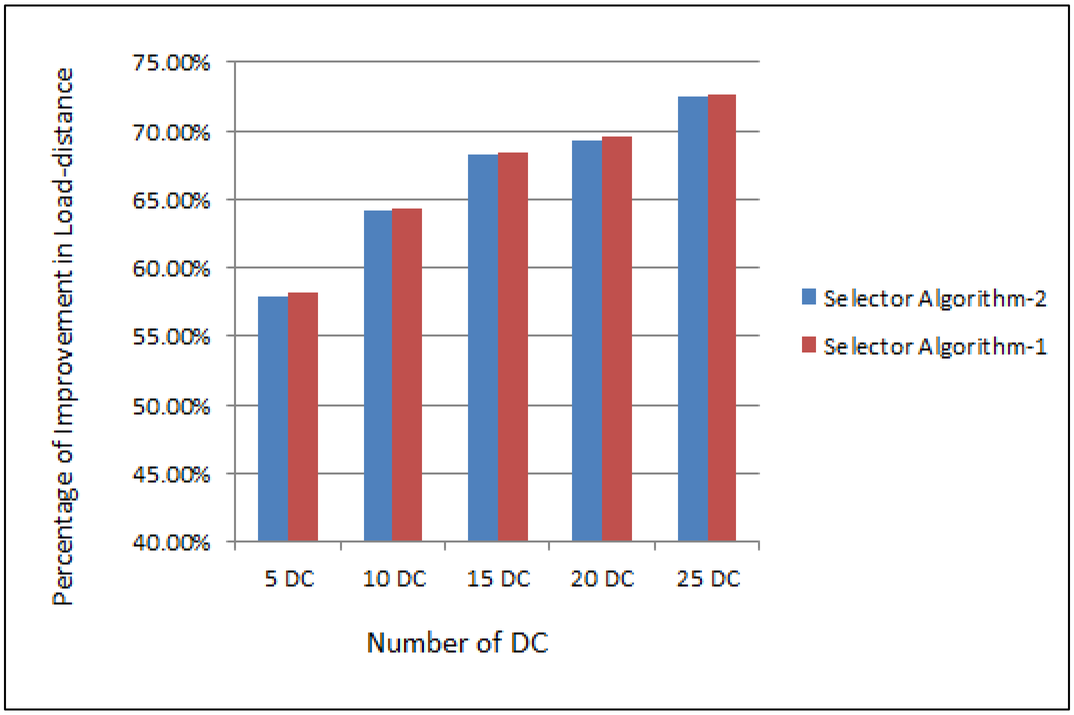
Appendix A Figure 32: 2500VMs/DC, 20K to 30K flows per region, Scenario 3



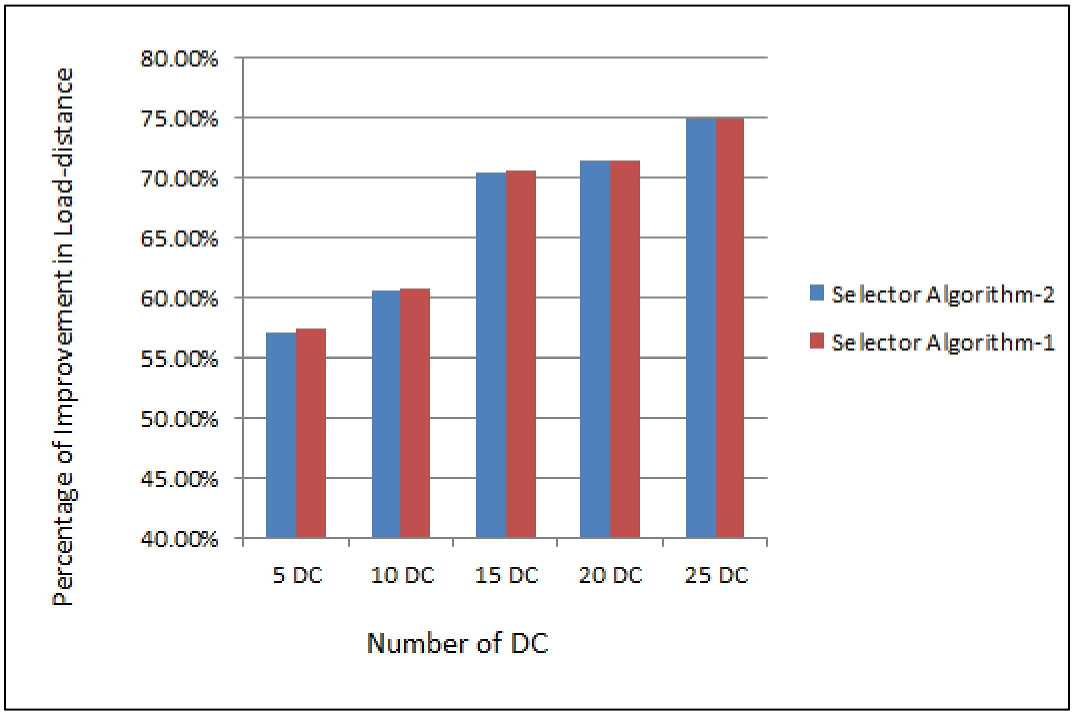
Appendix A Figure 33: 1500VMs/DC, 20K to 30K flows per region, Scenario 3



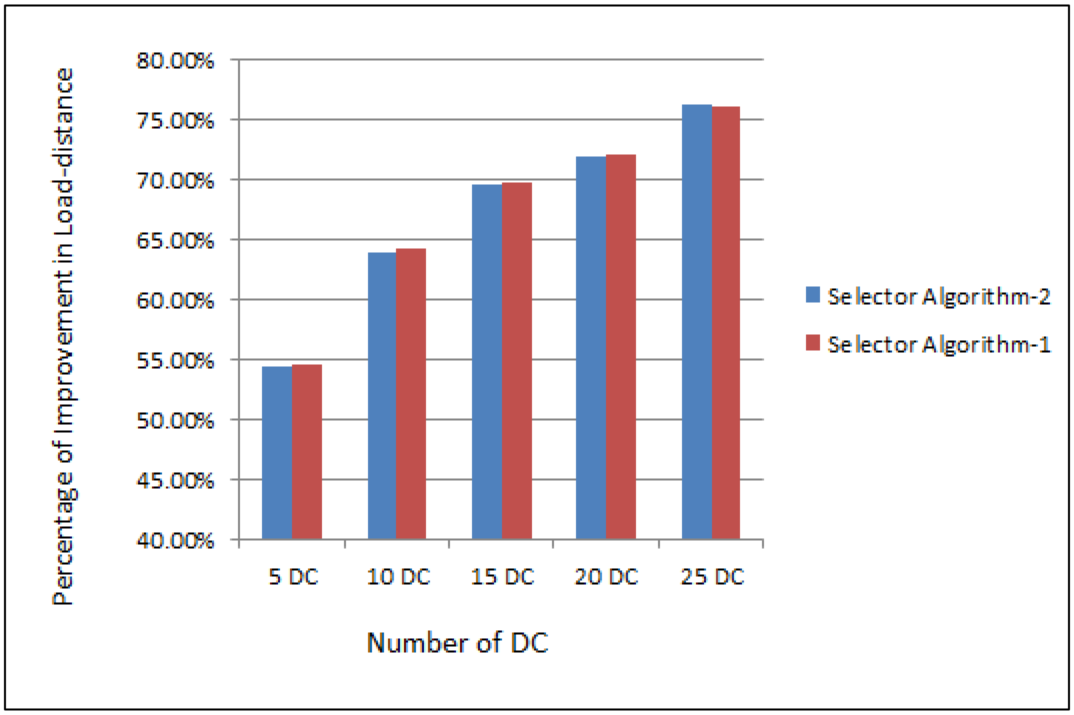
Appendix A Figure 34: 1000VMs/DC, 20K to 30K flows per region, Scenario 3



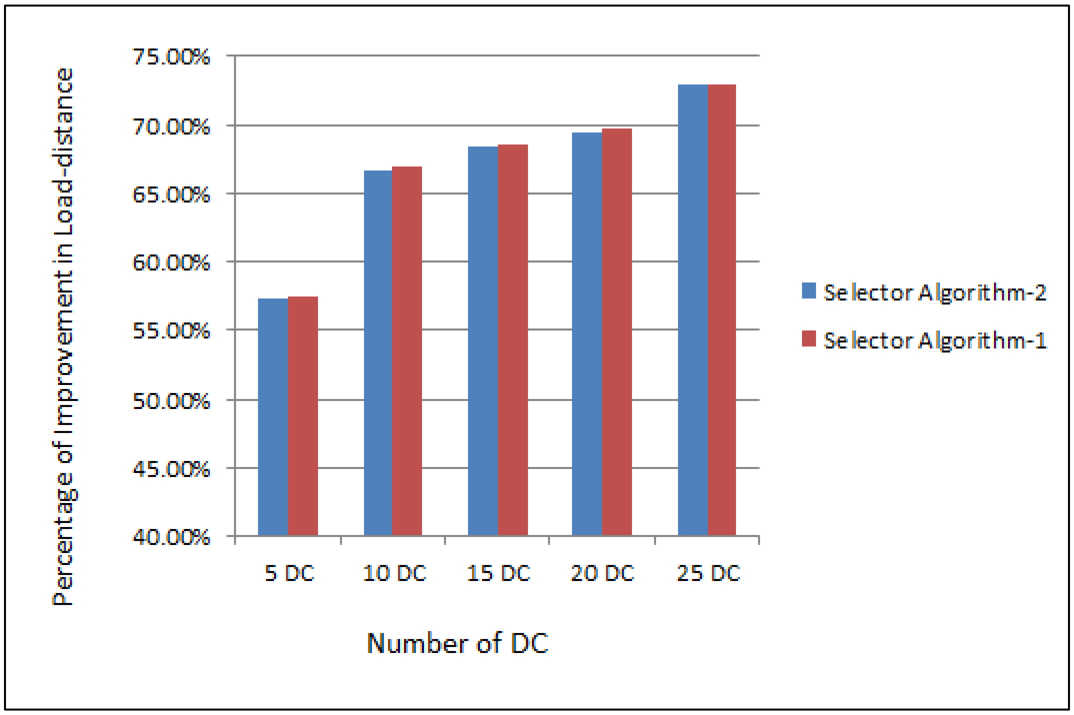
Appendix A Figure 35: 200VMs/DC, 20K to 30K flows per region, Scenario 3



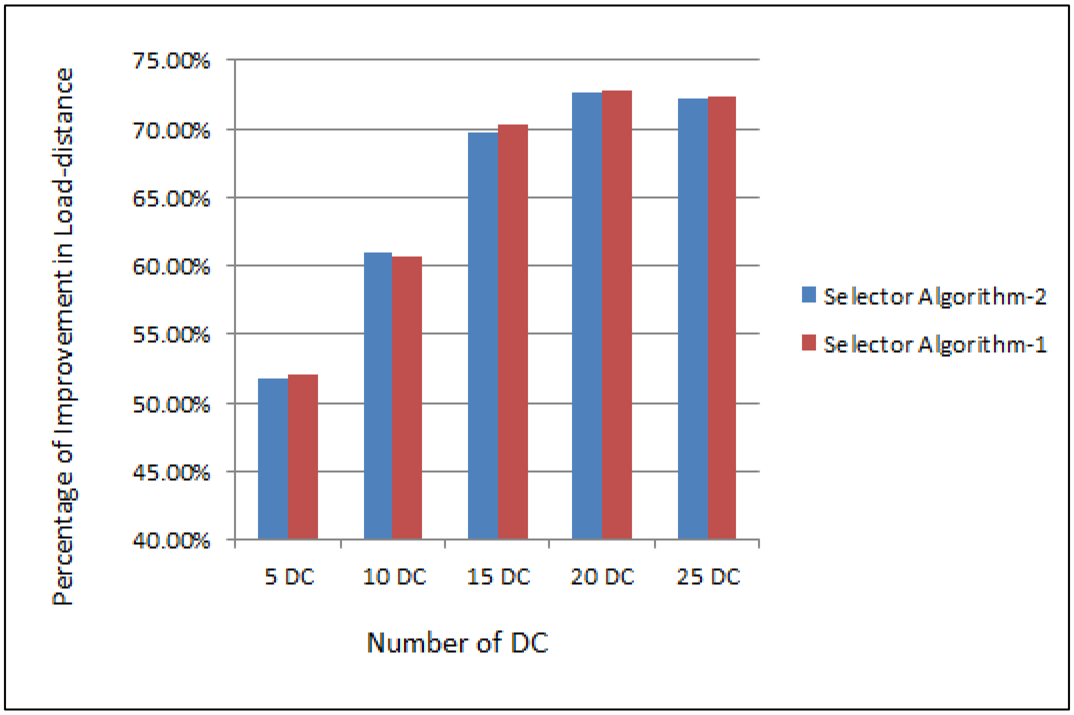
Appendix A Figure 36: 5000VMs/DC, 10K to 20K flows per region, Scenario 3



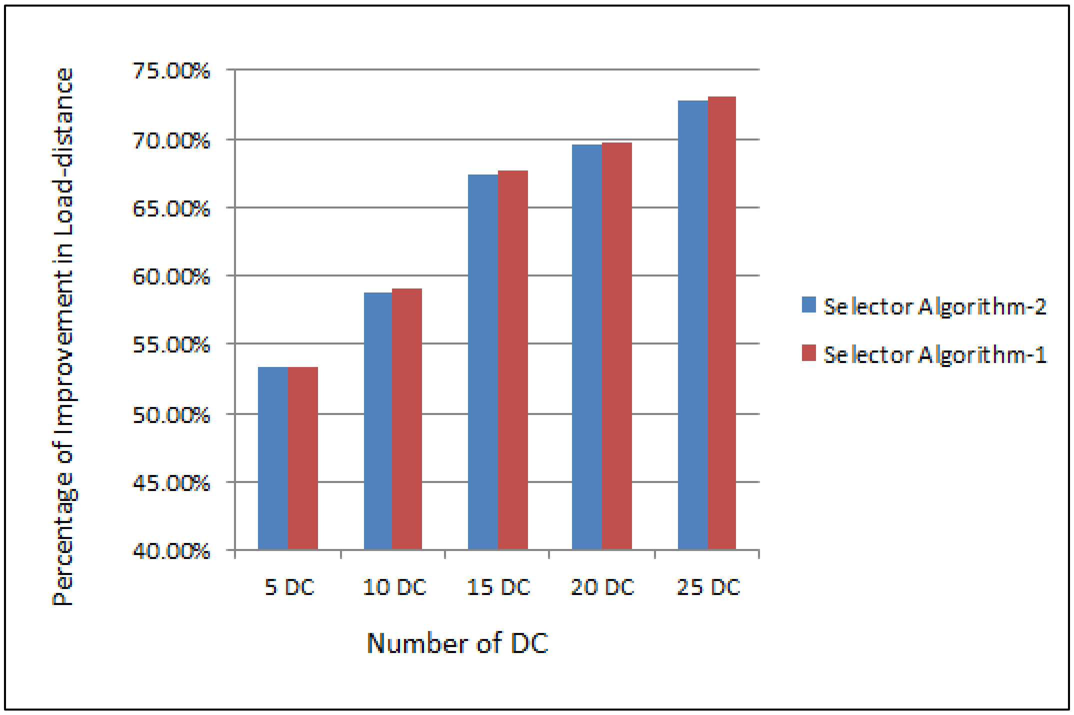
Appendix A Figure 37: 2500VMs/DC, 10K to 20K flows per region, Scenario 3



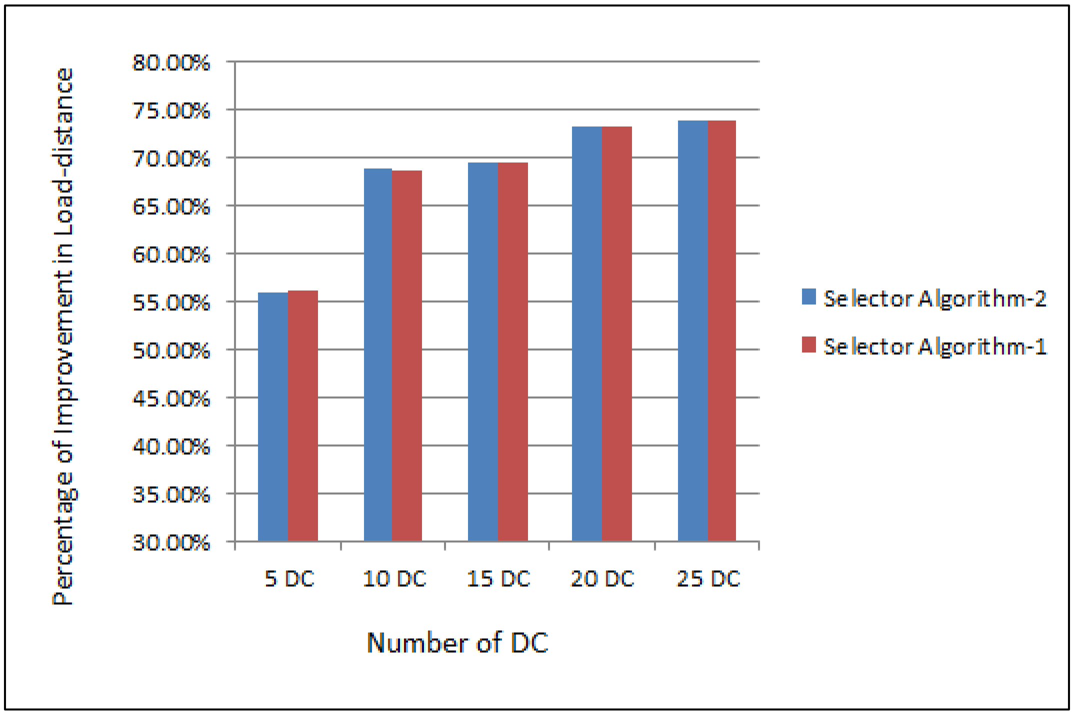
Appendix A Figure 38: 1500VMs/DC, 10K to 20K flows per region, Scenario 3



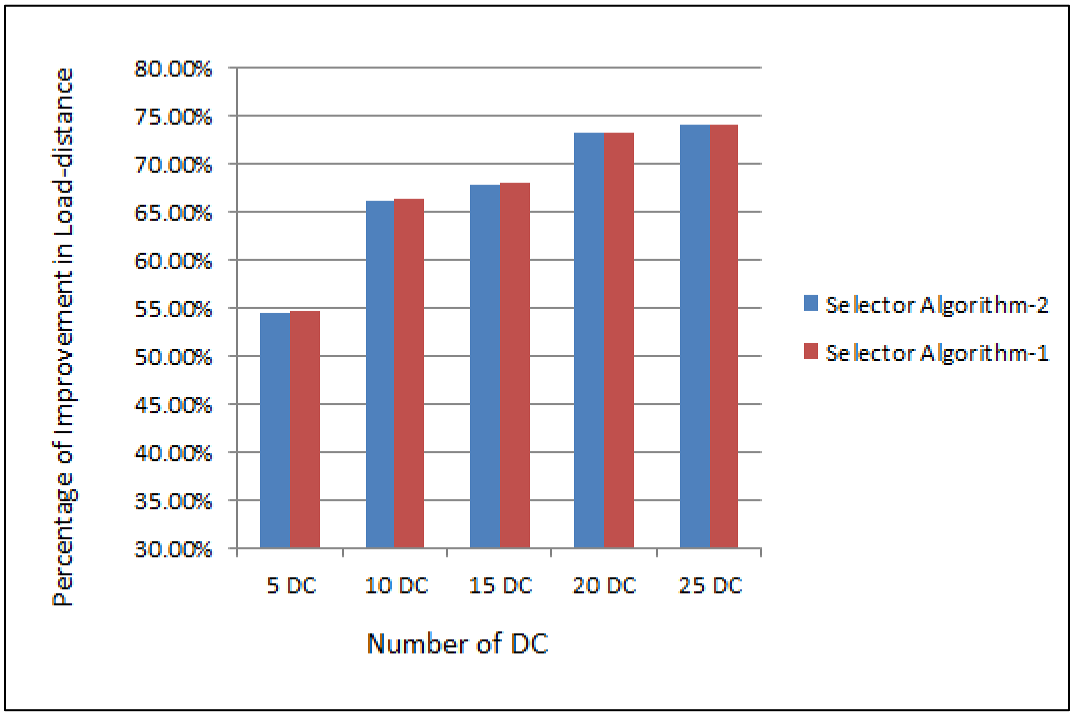
Appendix A Figure 39: 1000VMs/DC, 10K to 20K flows per region, Scenario 3



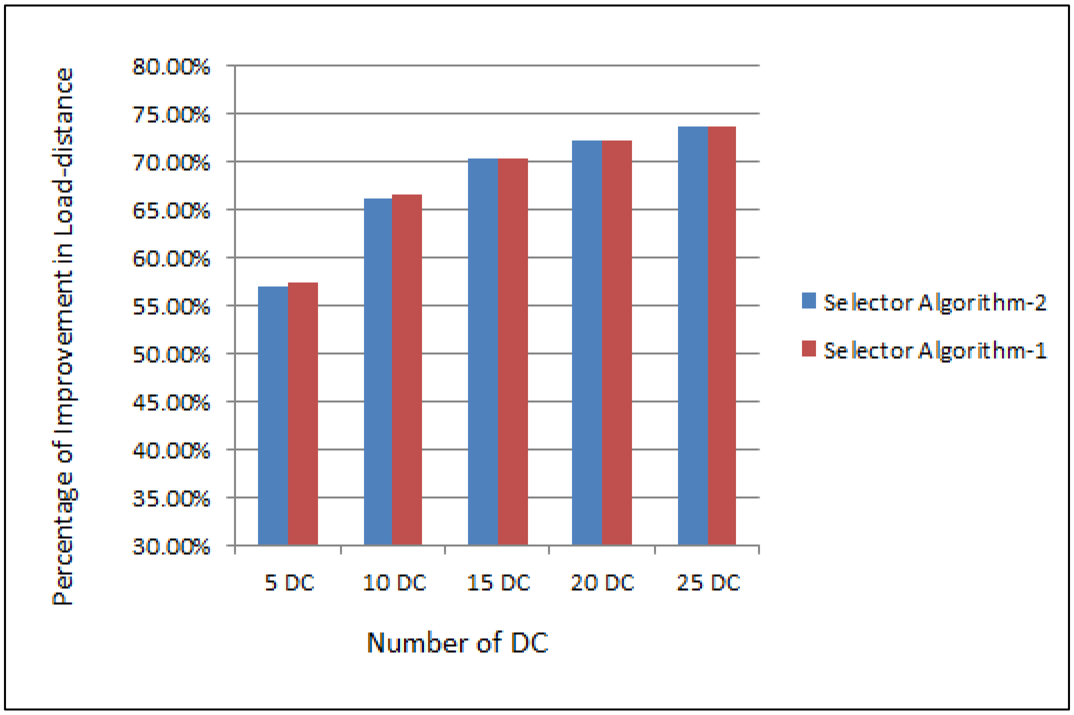
Appendix A Figure 40: 200VMs/DC, 10K to 20K flows per region, Scenario 3



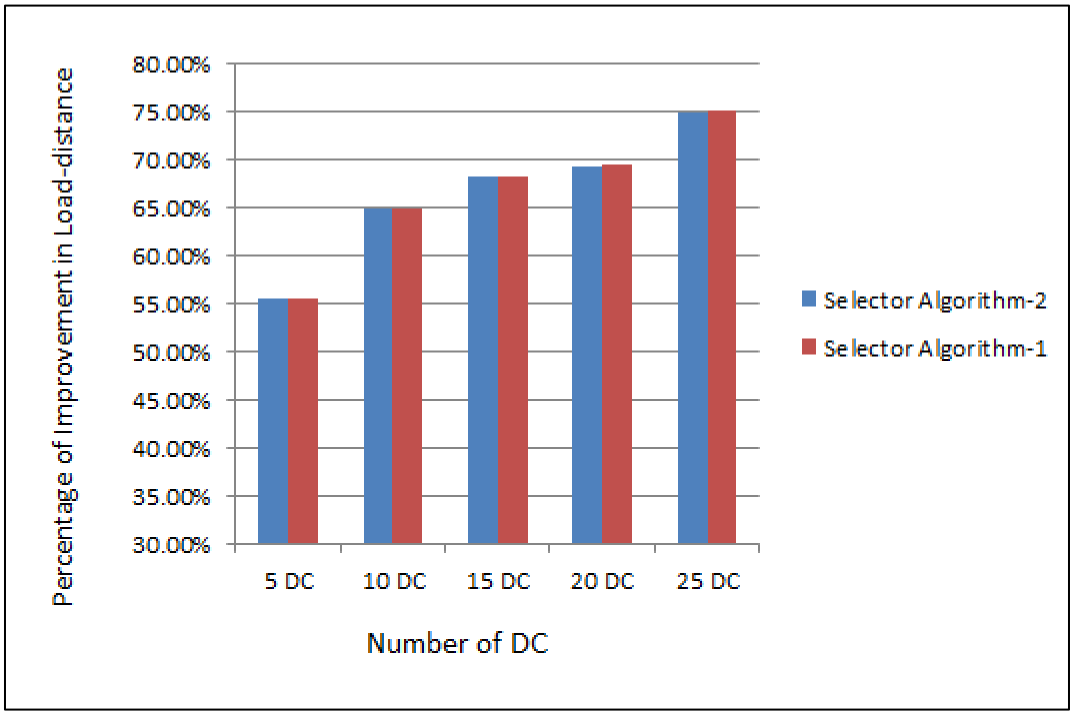
Appendix A Figure 41: 5000VMs/DC, 5K to 10K flows per region, Scenario 3



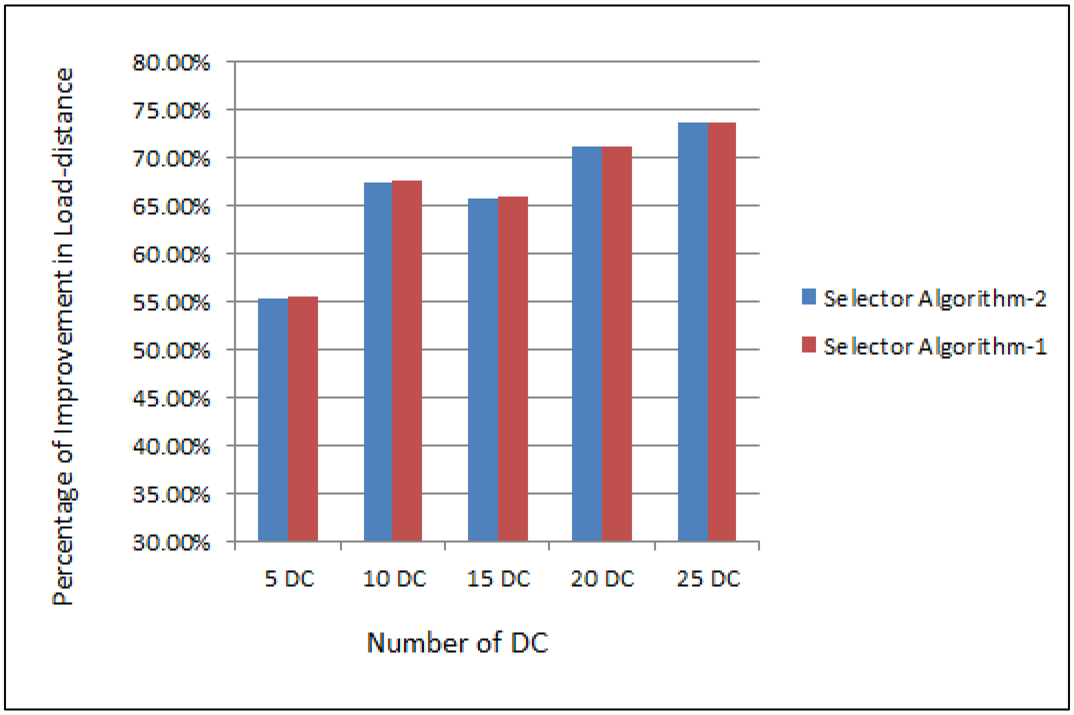
Appendix A Figure 42: 2500VMs/DC, 5K to 10K flows per region, Scenario 3



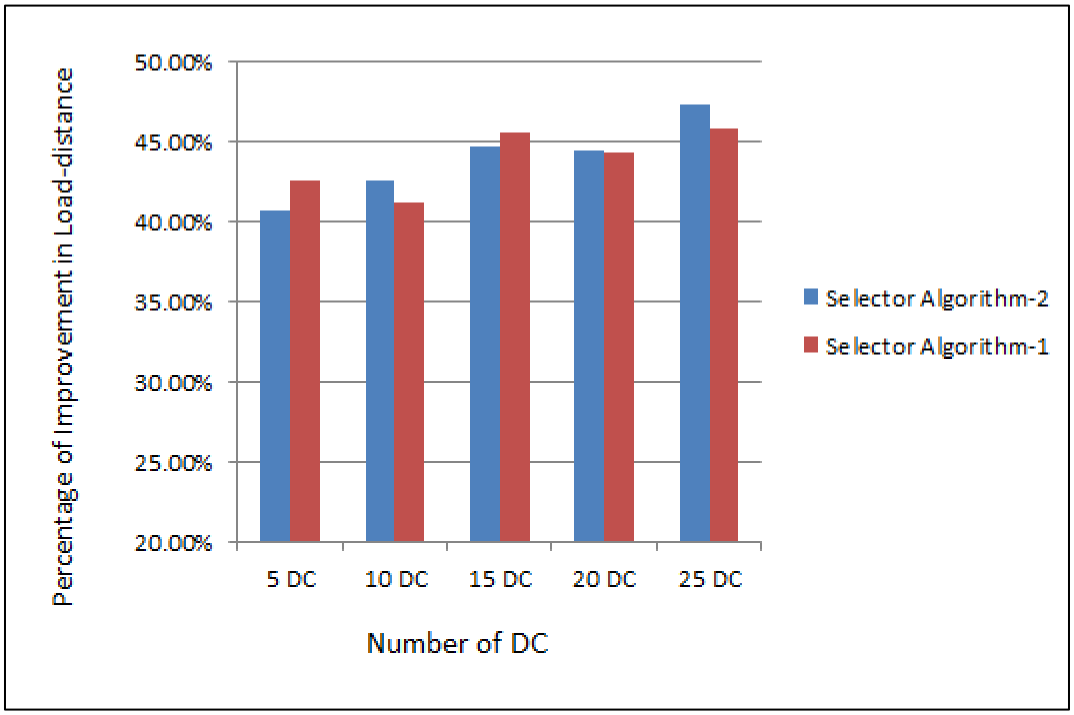
Appendix A Figure 43: 1500VMs/DC, 5K to 10K flows per region, Scenario 3



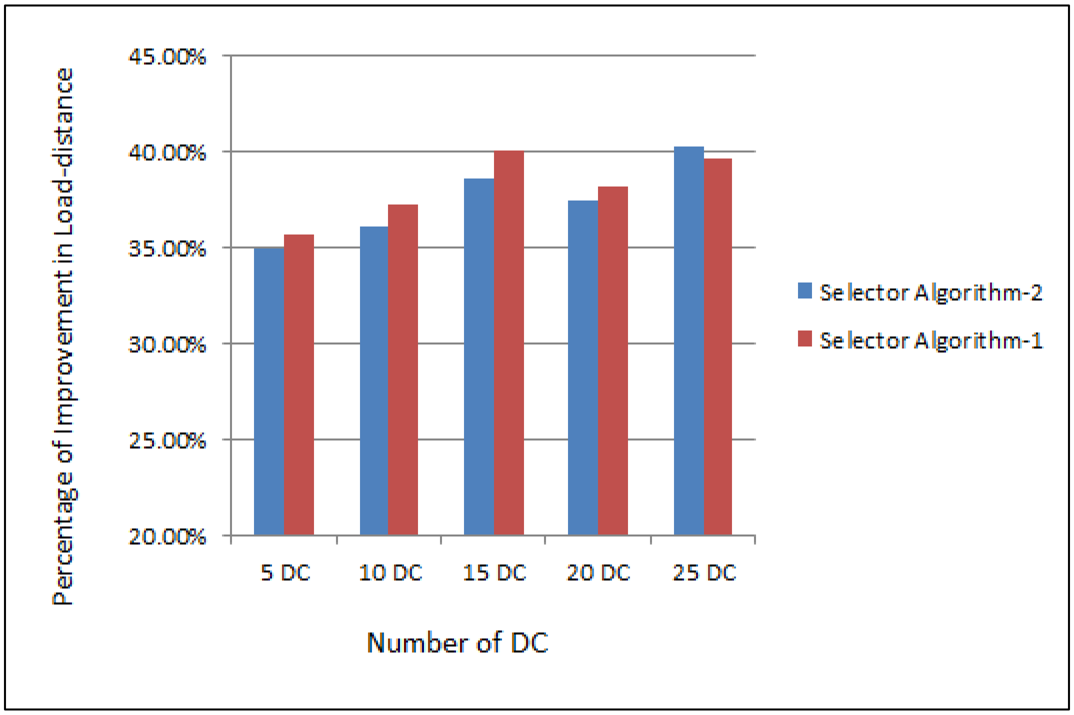
Appendix A Figure 44: 1000VMs/DC, 5K to 10K flows per region, Scenario 3



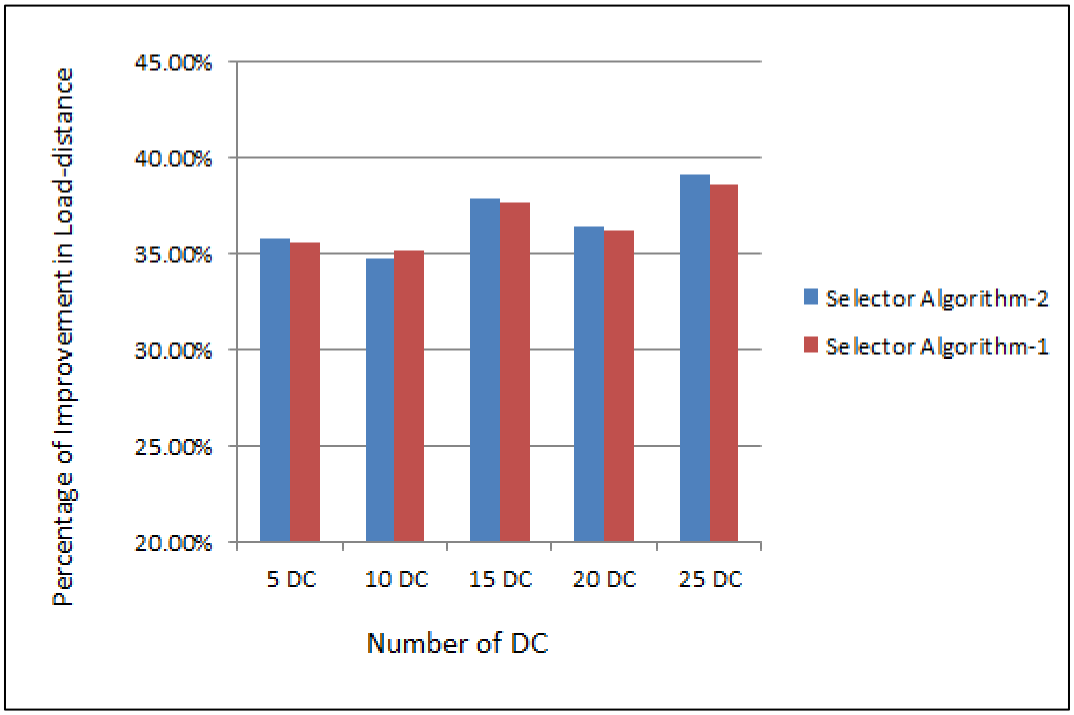
Appendix A Figure 45: 200VMs/DC, 5K to 10K flows per region, Scenario 3



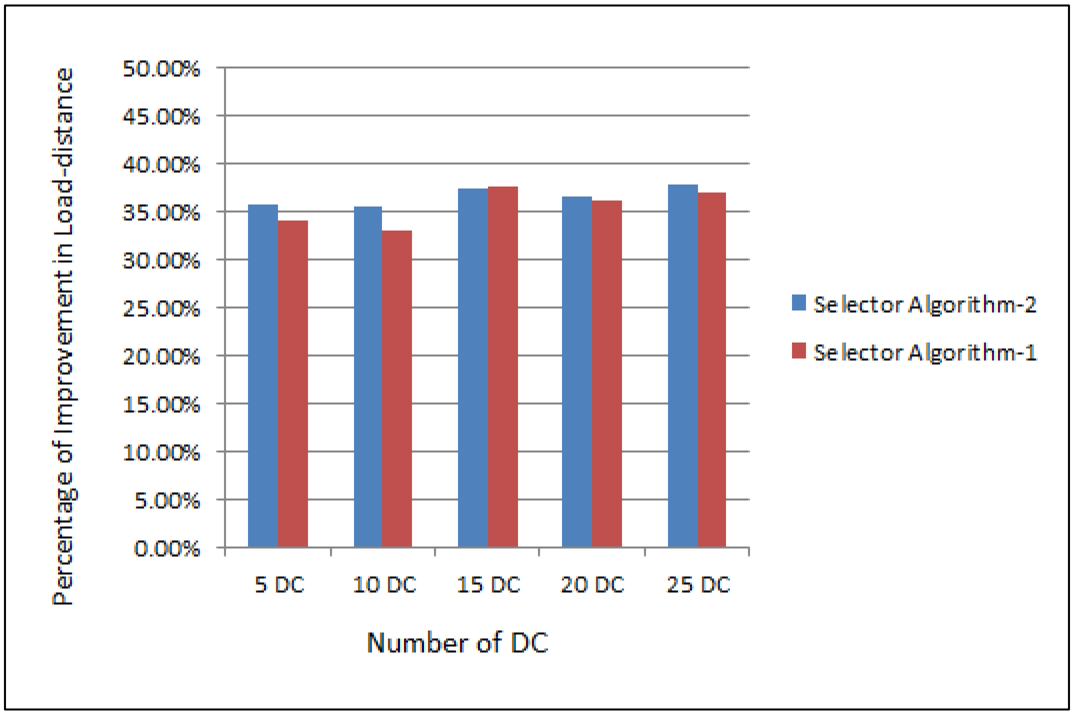
Appendix A Figure 46: 5000VMs/DC, 20K to 30K flows per region, Scenario 4



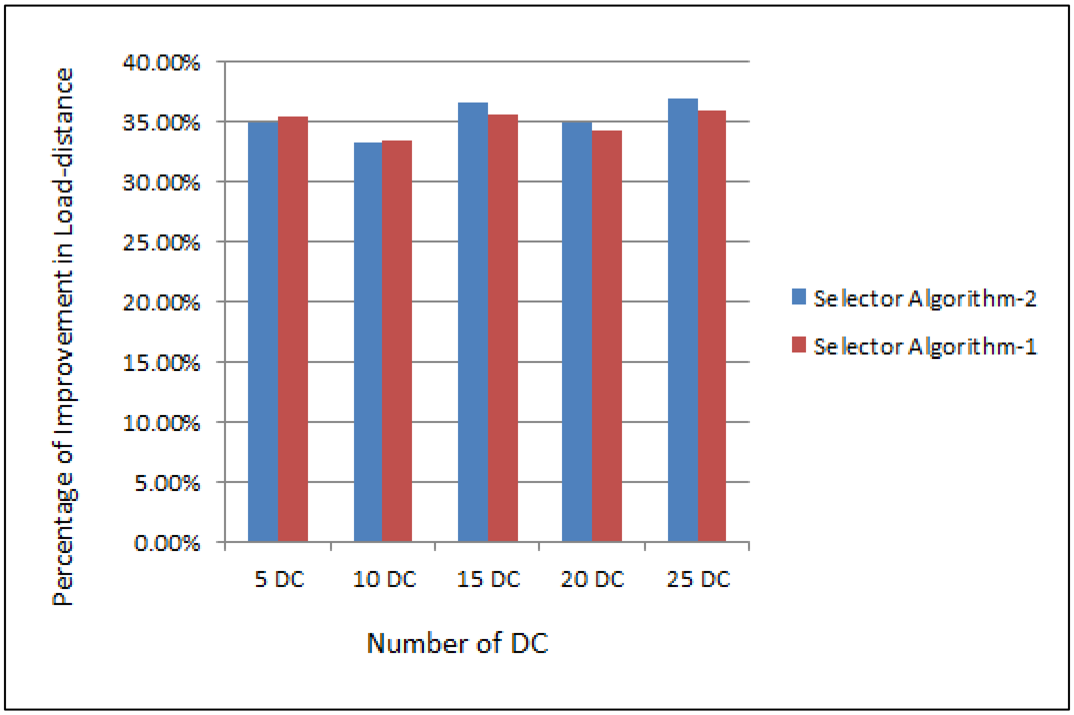
Appendix A Figure 47: 2500VMs/DC, 20K to 30K flows per region, Scenario 4



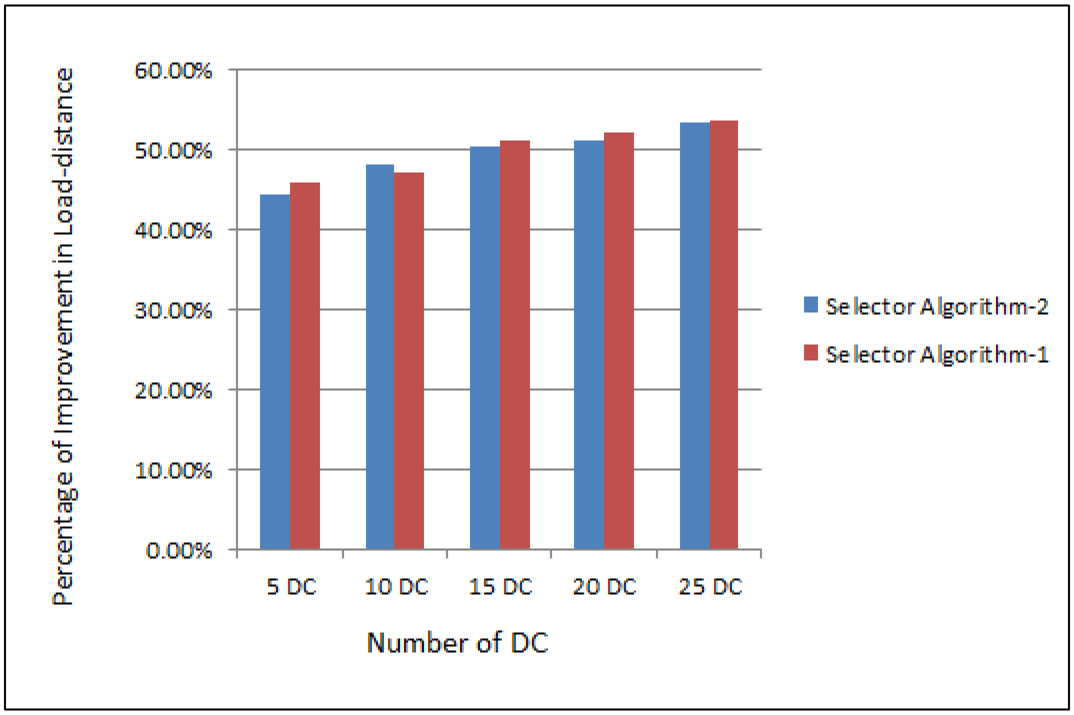
Appendix A Figure 48: 1500VMs/DC, 20K to 30K flows per region, Scenario 4



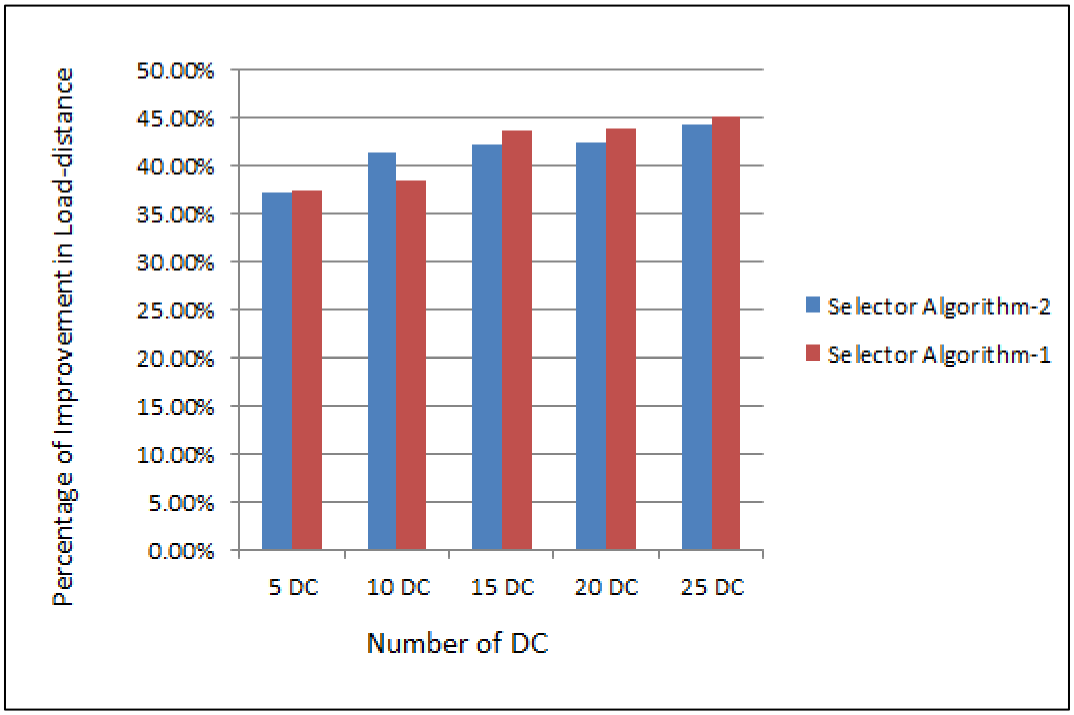
Appendix A Figure 49: 1000VMs/DC, 20K to 30K flows per region, Scenario 4



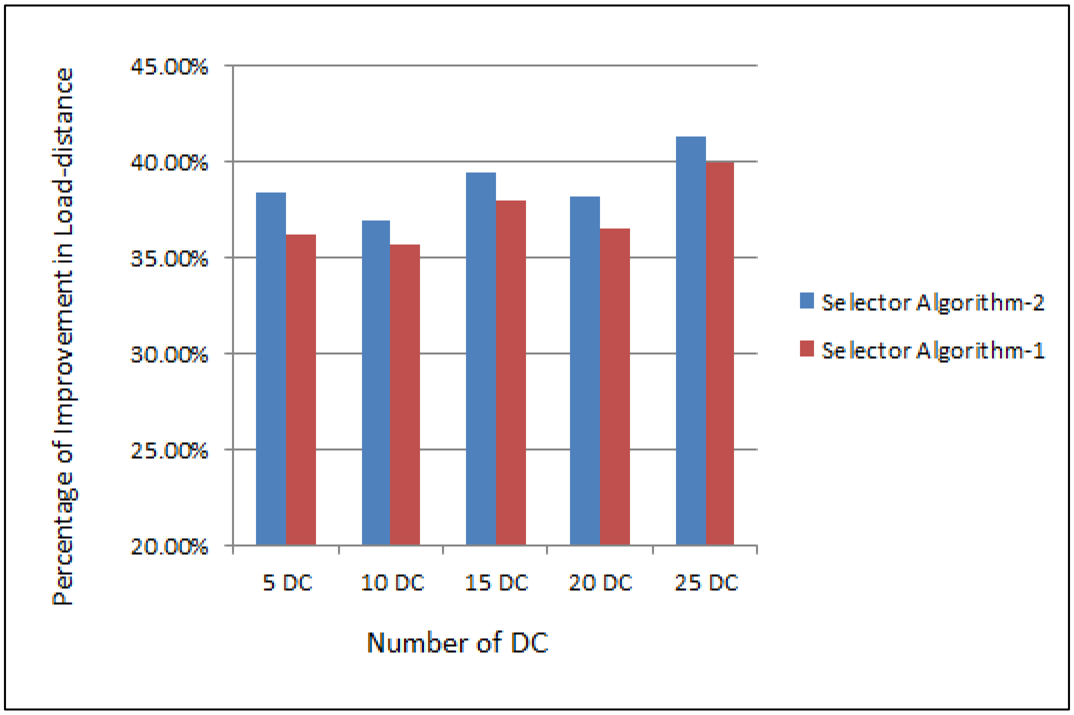
Appendix A Figure 50: 200VMs/DC, 20K to 30K flows per region, Scenario 4



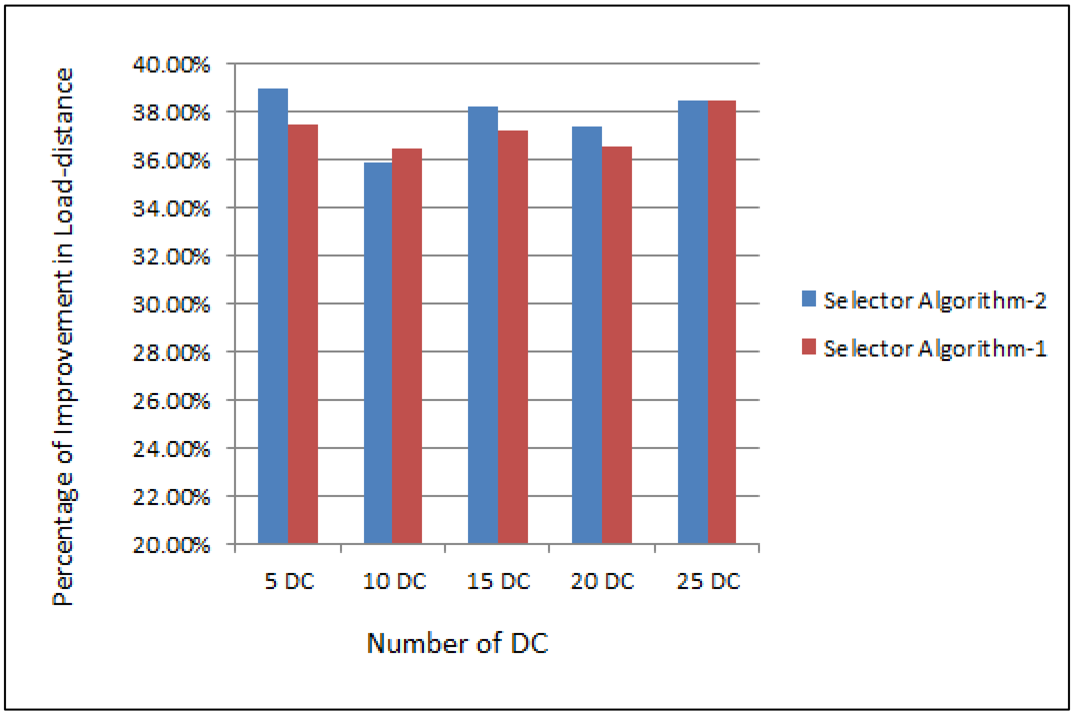
Appendix A Figure 51: 5000VMs/DC, 10K to 20K flows per region, Scenario 4



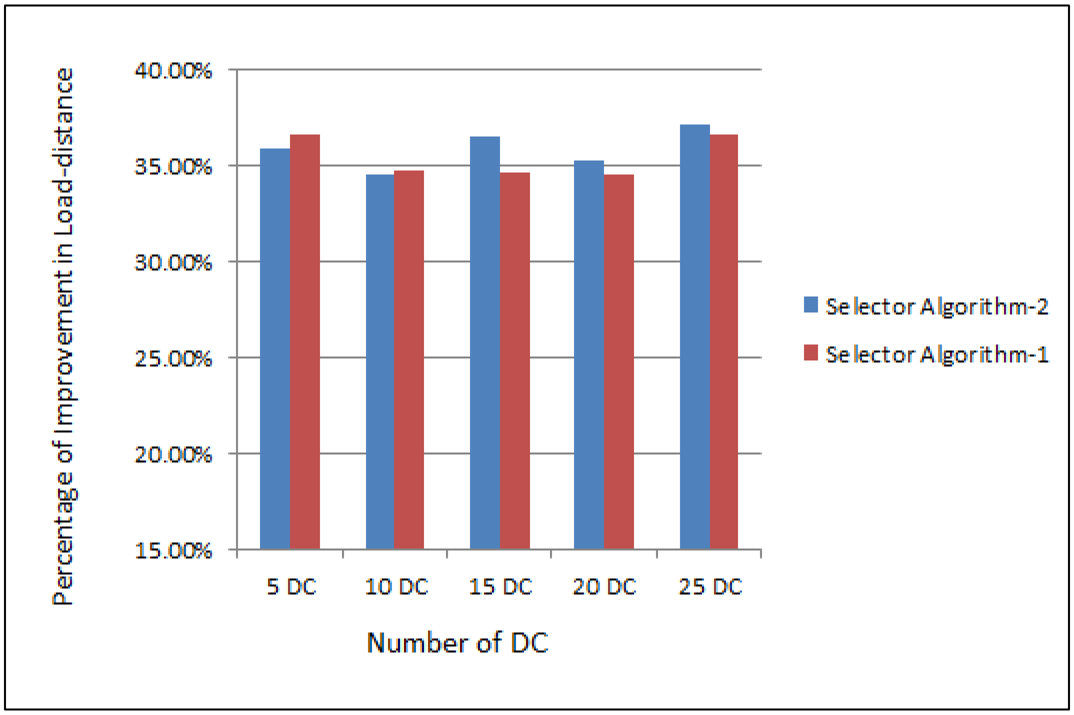
Appendix A Figure 52: 2500VMs/DC, 10K to 20K flows per region, Scenario 4



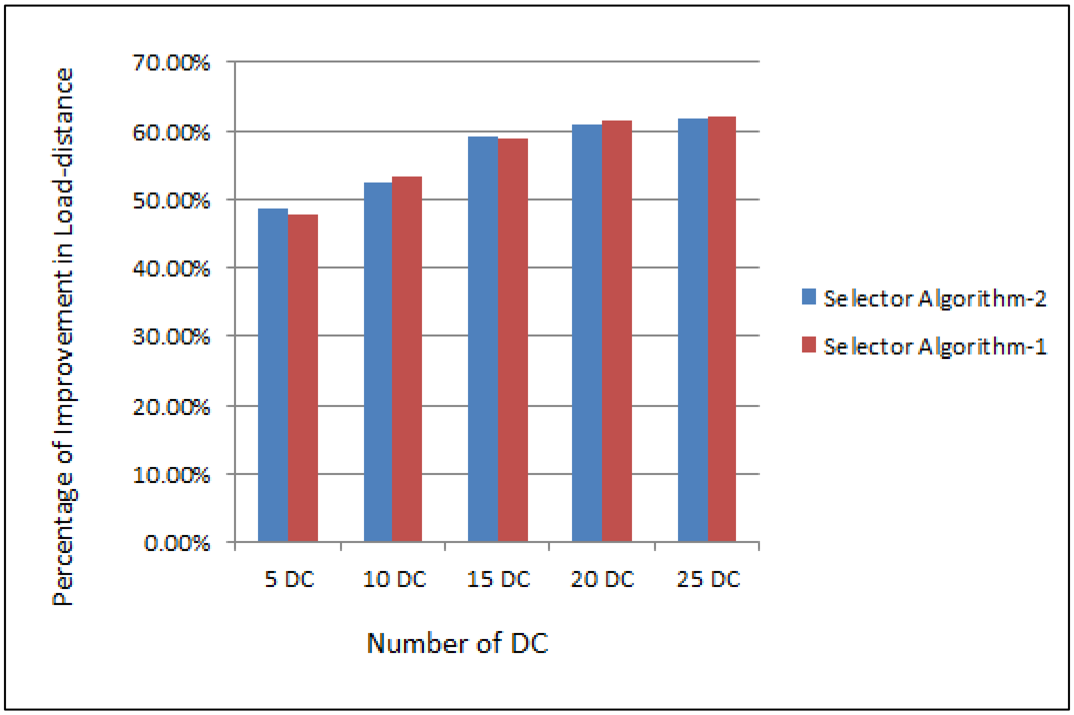
Appendix A Figure 53: 1500VMs/DC, 10K to 20K flows per region, Scenario 4



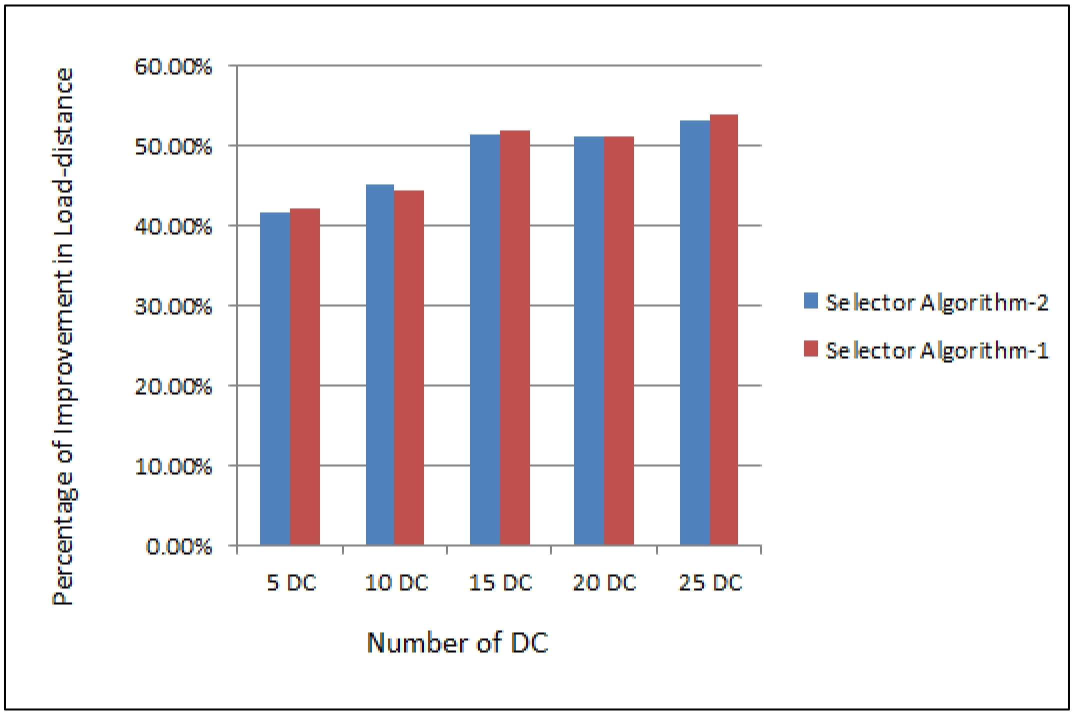
Appendix A Figure 54: 1000VMs/DC, 10K to 20K flows per region, Scenario 4



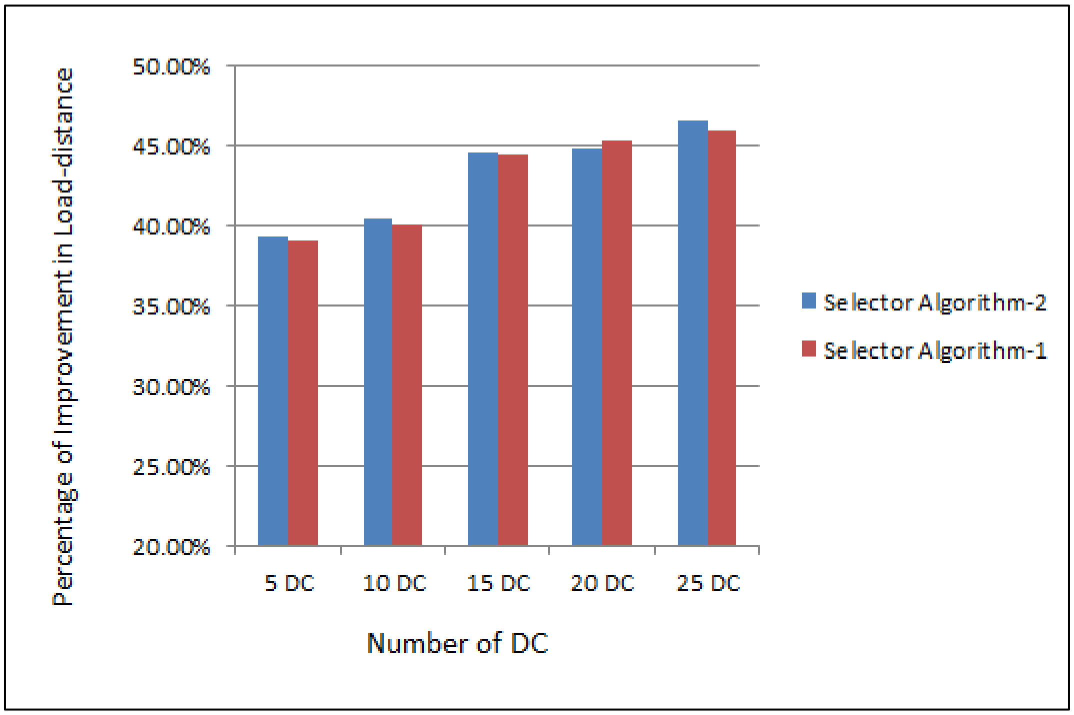
Appendix A Figure 55: 200VMs/DC, 10K to 20K flows per region, Scenario 4



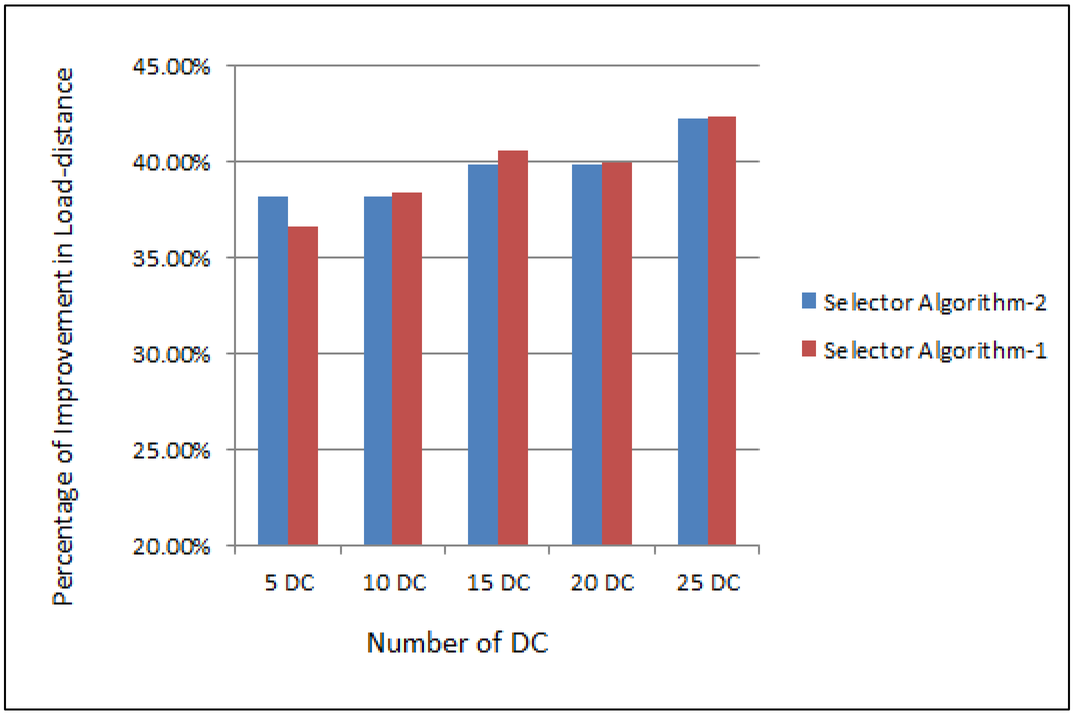
Appendix A Figure 56: 5000VMs/DC, 5K to 10K flows per region, Scenario 4



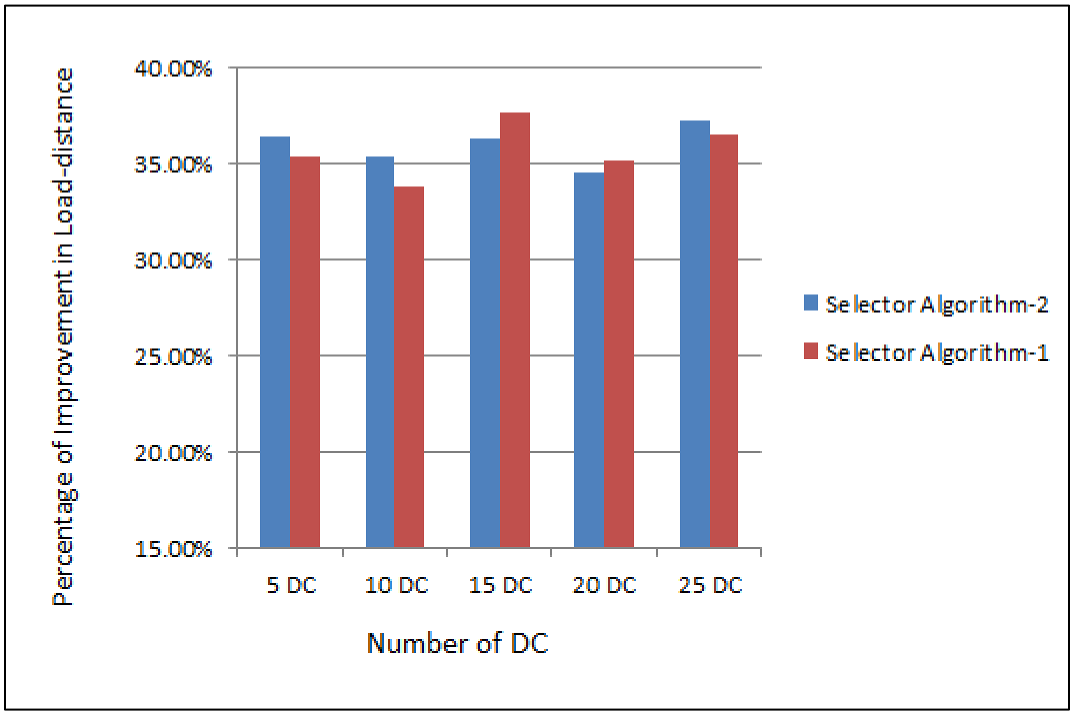
Appendix A Figure 57: 2500VMs/DC, 5K to 10K flows per region, Scenario 4



Appendix A Figure 58: 1500VMs/DC, 5K to 10K flows per region, Scenario 4

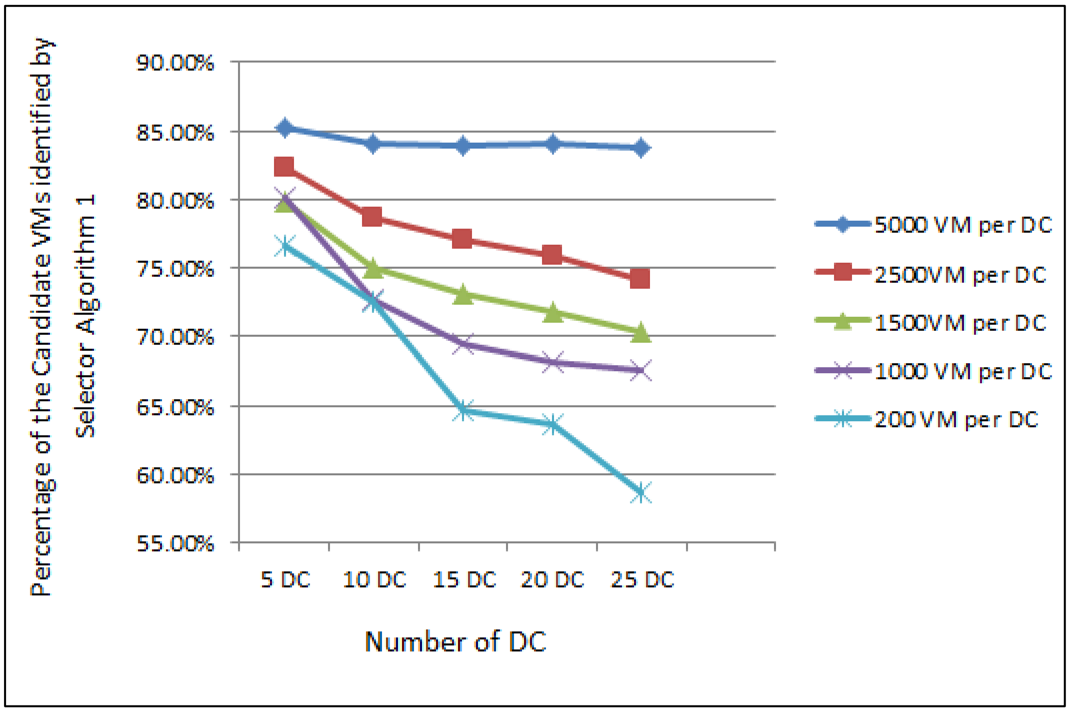


Appendix A Figure 59: 1000VMs/DC, 5K to 10K flows per region, Scenario 4

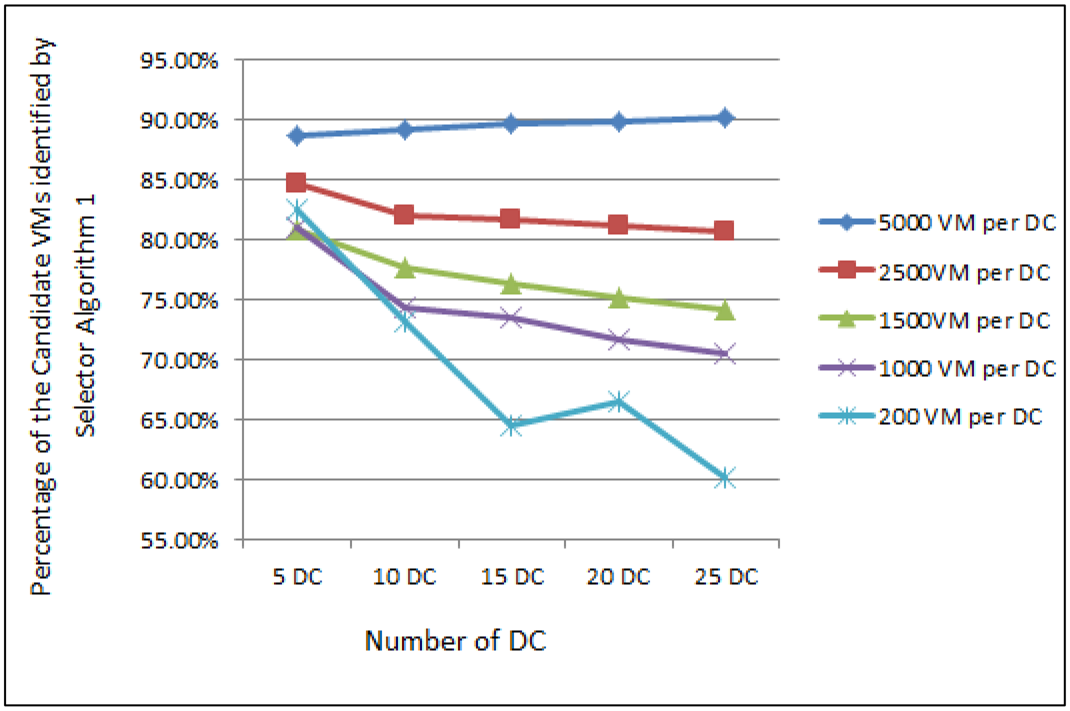


Appendix A Figure 60: 200VMs/DC, 5K to 10K flows per region, Scenario 4

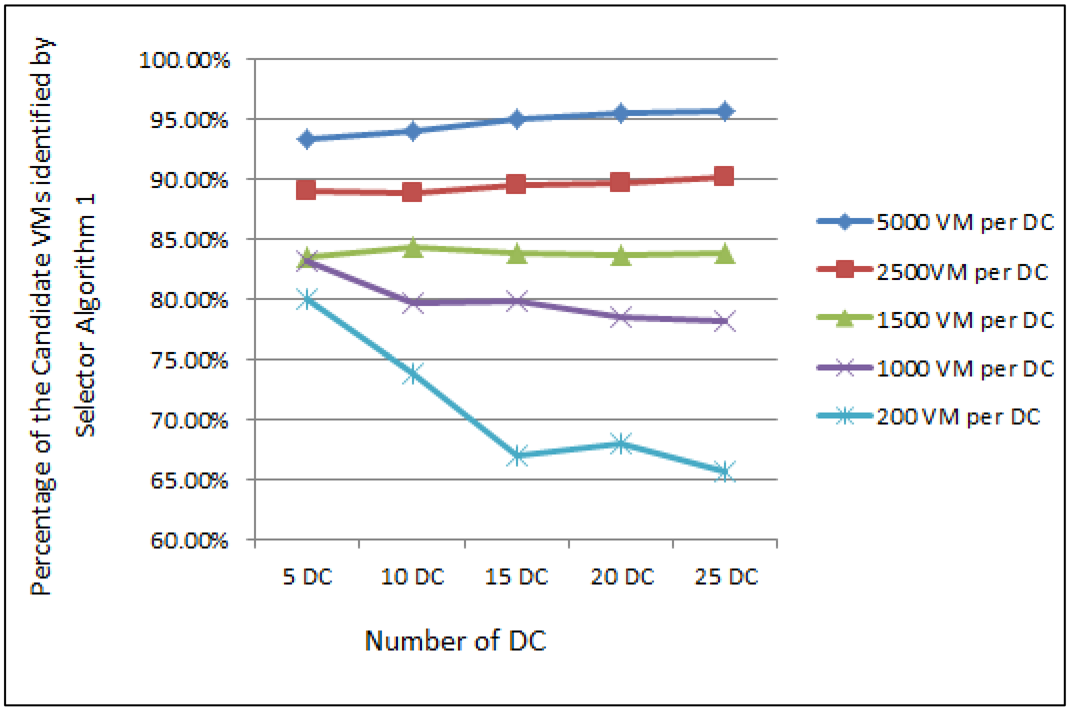
Appendix B: Percentage of candidate VM identified for Migration Algorithms



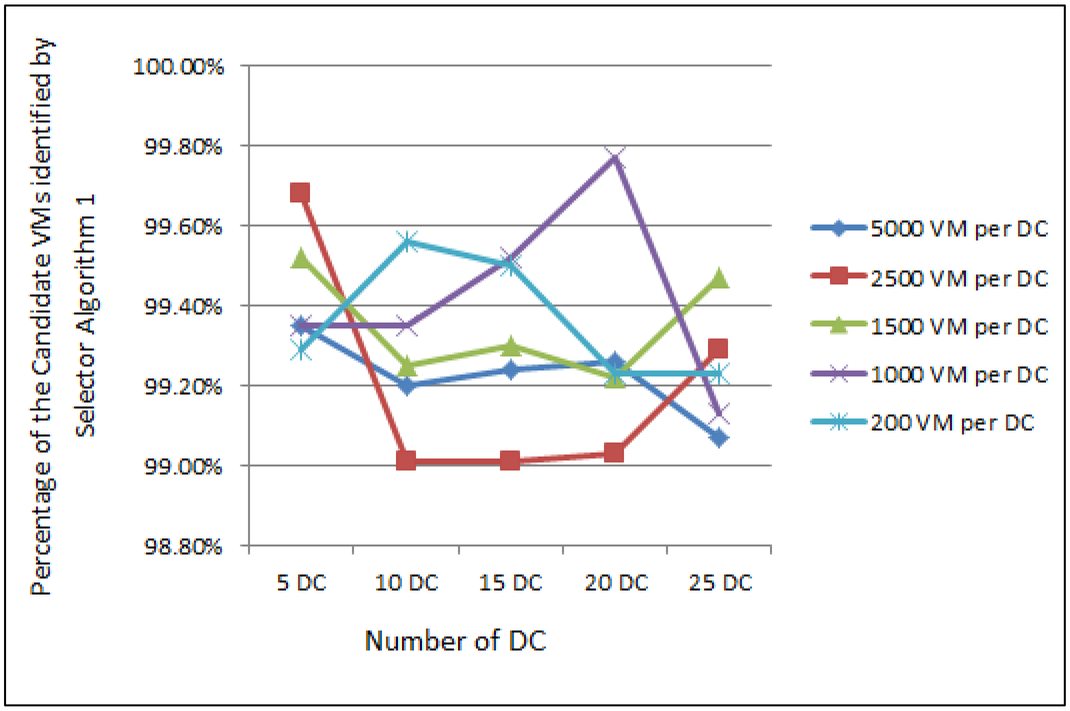
Appendix B Figure 1: 20K to 30K flows per region, Scenario 1



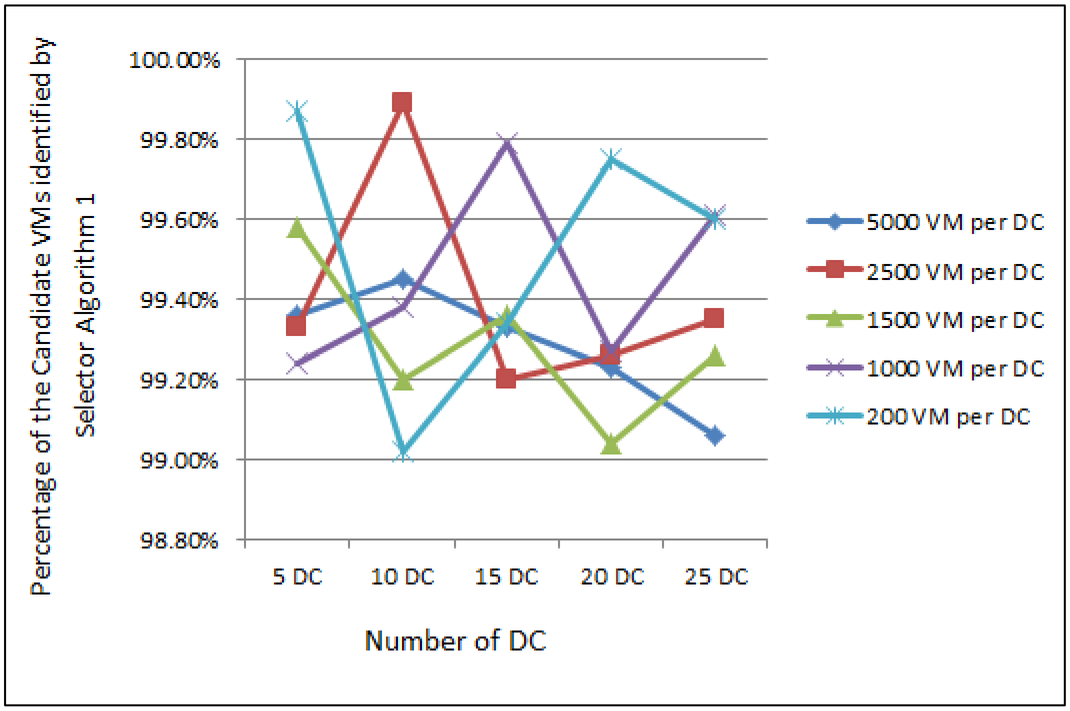
Appendix B Figure 2: 10K to 20K flows per region, Scenario 1



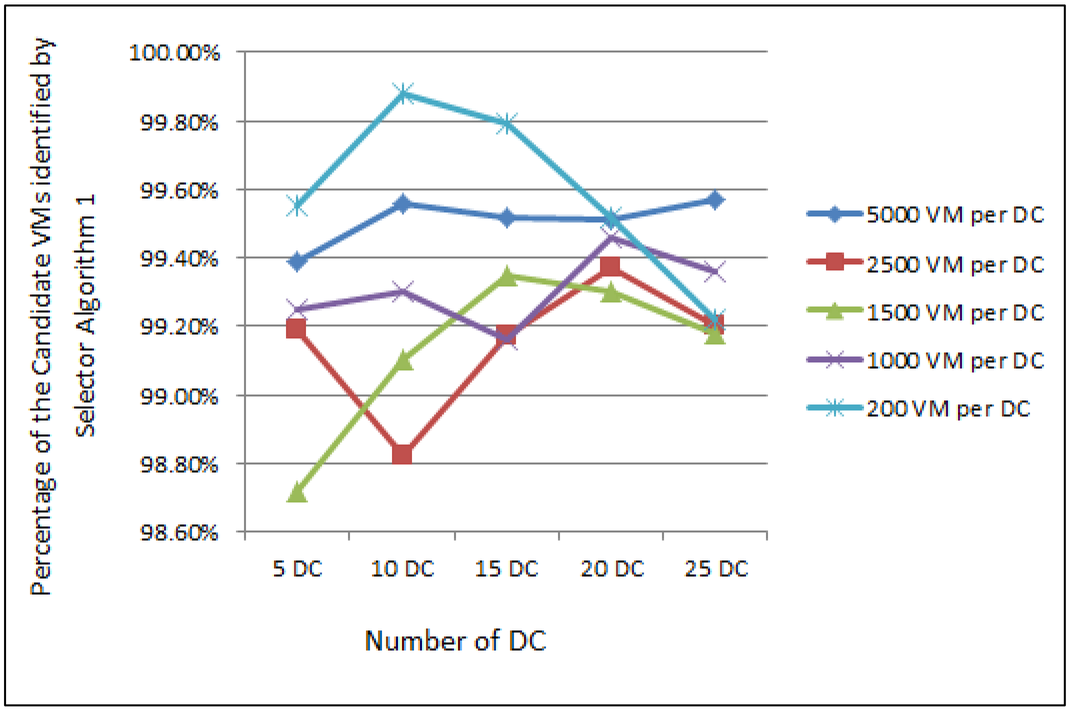
Appendix B Figure 3: 5K to 10K flows per region, Scenario 1



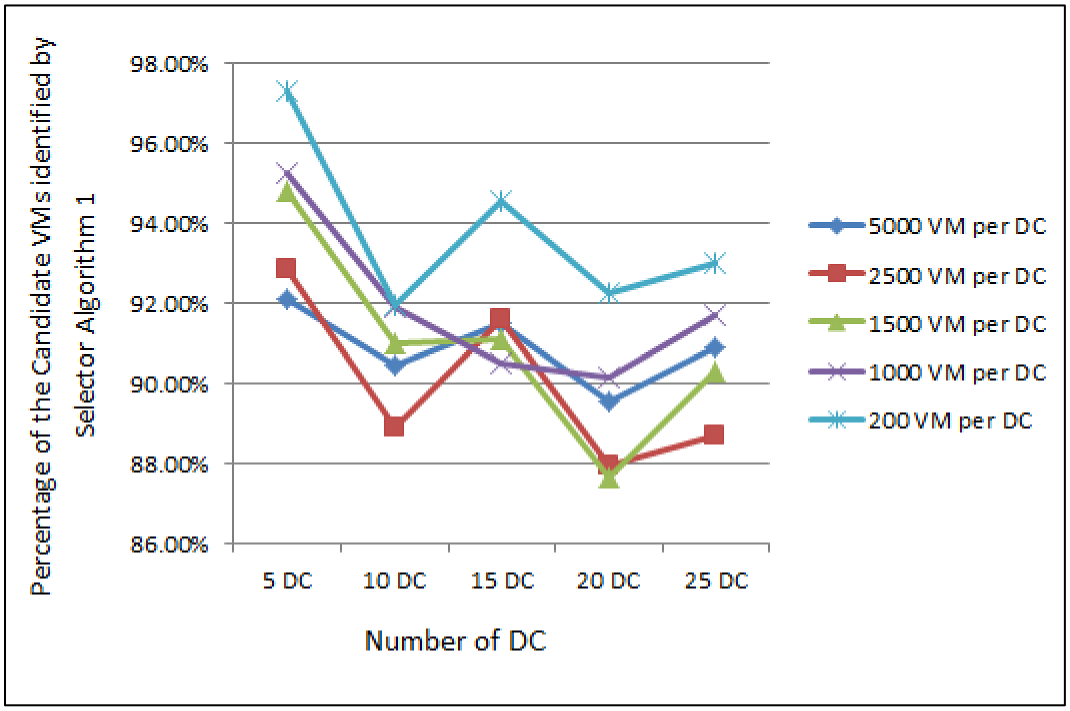
Appendix B Figure 4: 20K to 30K flows per region, Scenario 3



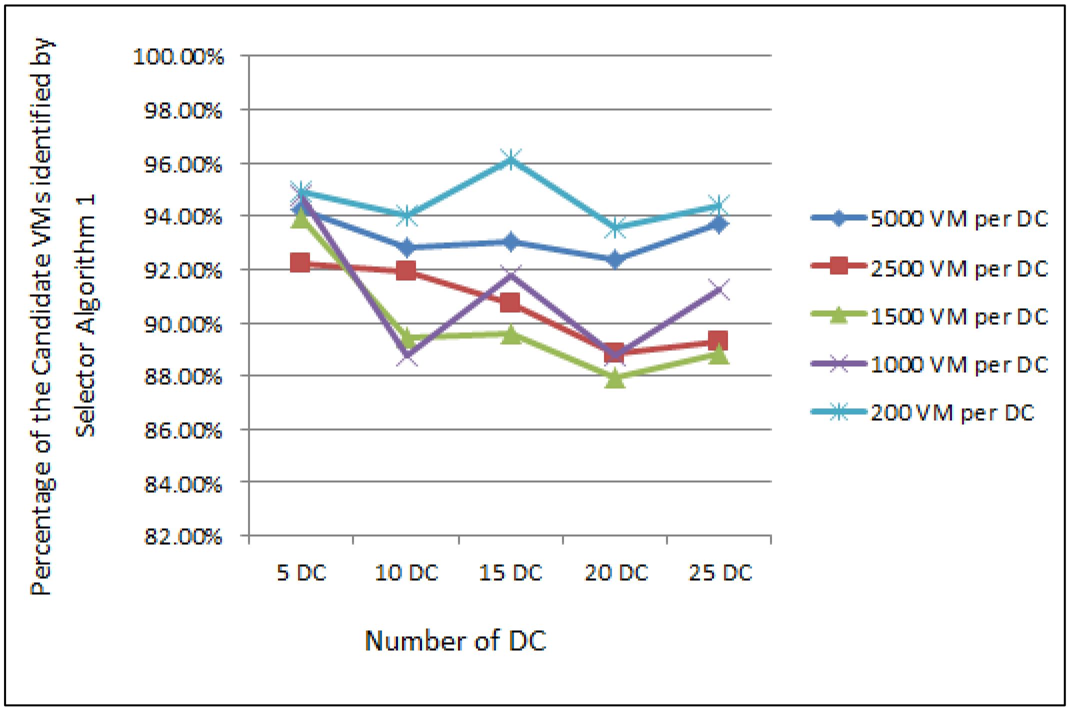
Appendix B Figure 5: 10K to 20K flows per region, Scenario 3



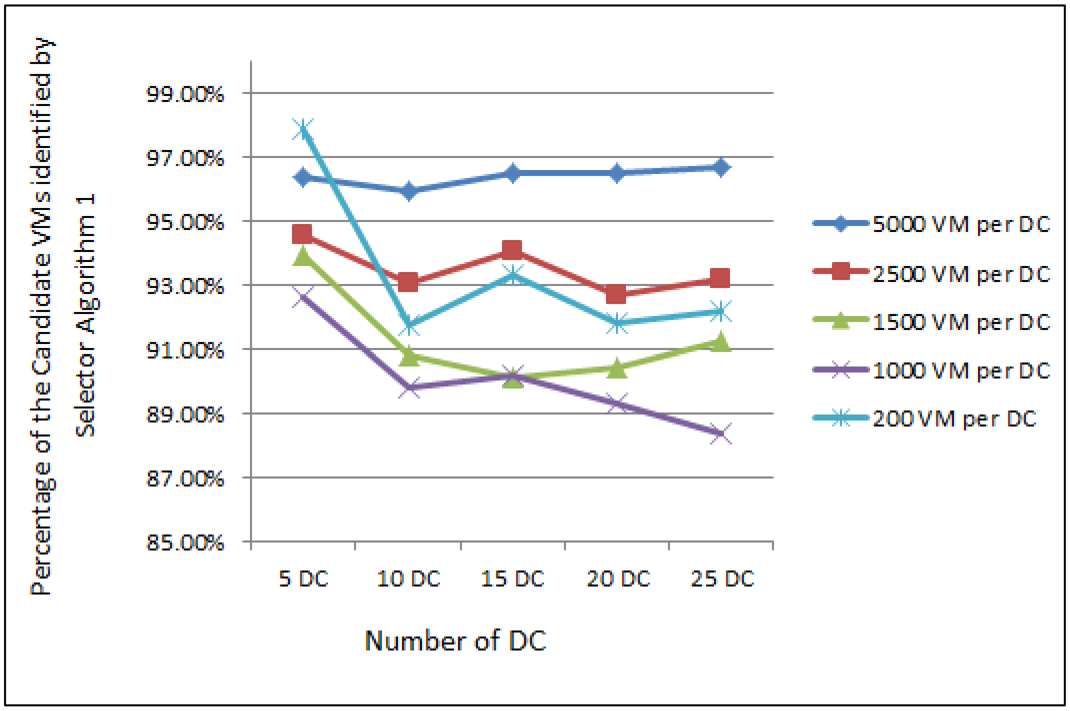
Appendix B Figure 6: 5K to 10K flows per region, Scenario 3



Appendix B Figure 7: 20K to 30K flows per region, Scenario 4

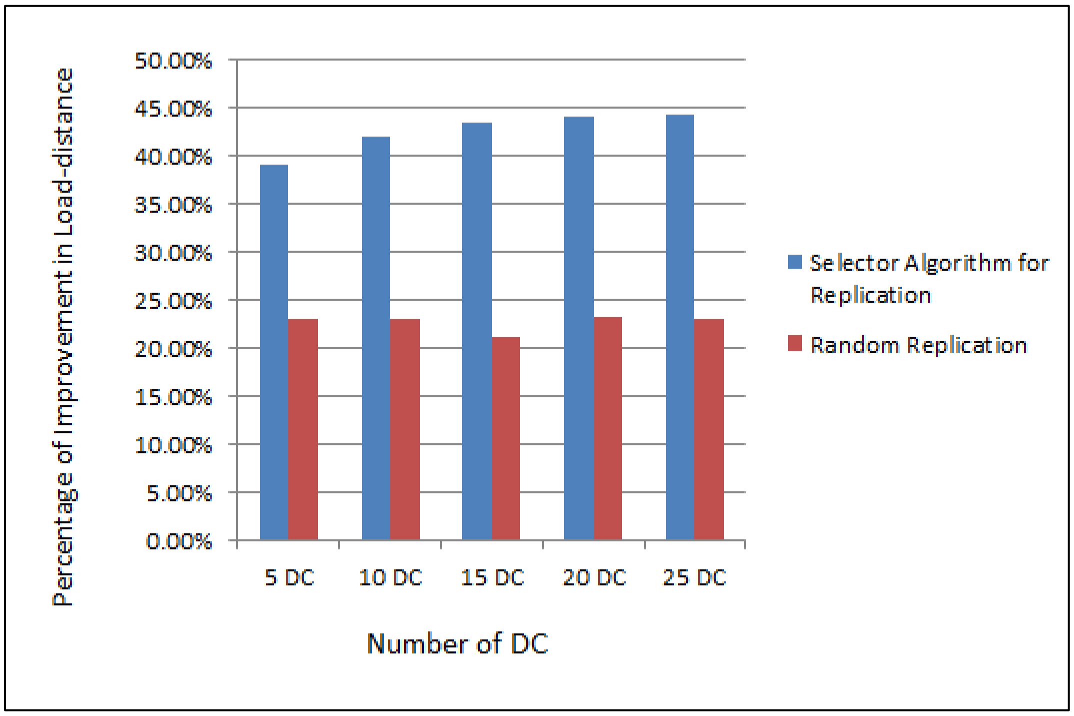


Appendix B Figure 8: 10K to 20K flows per region, Scenario 4

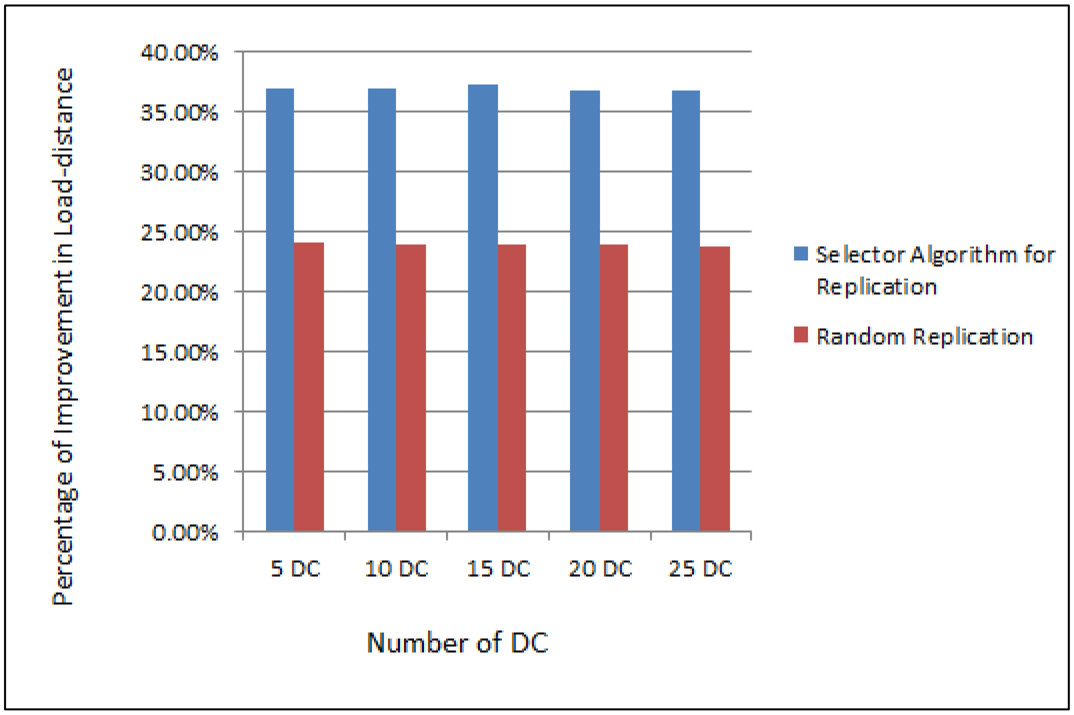


Appendix B Figure 9: 5K to 10K flows per region, Scenario 4

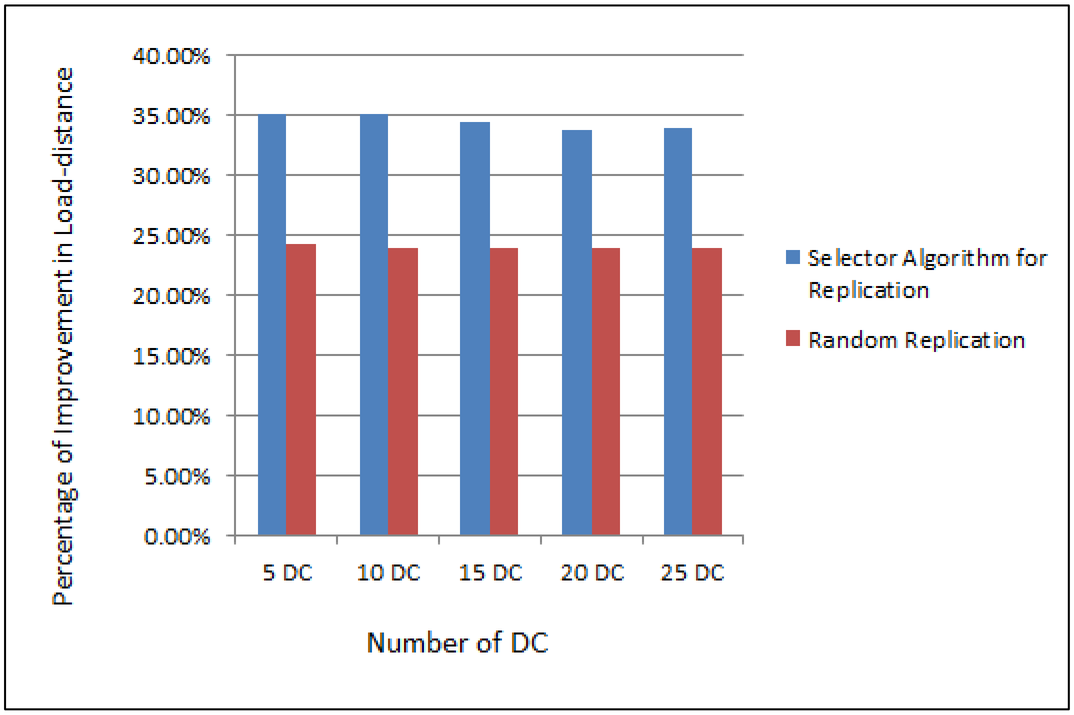
Appendix C: Percentage of Expected Improvement for Replication Algorithm



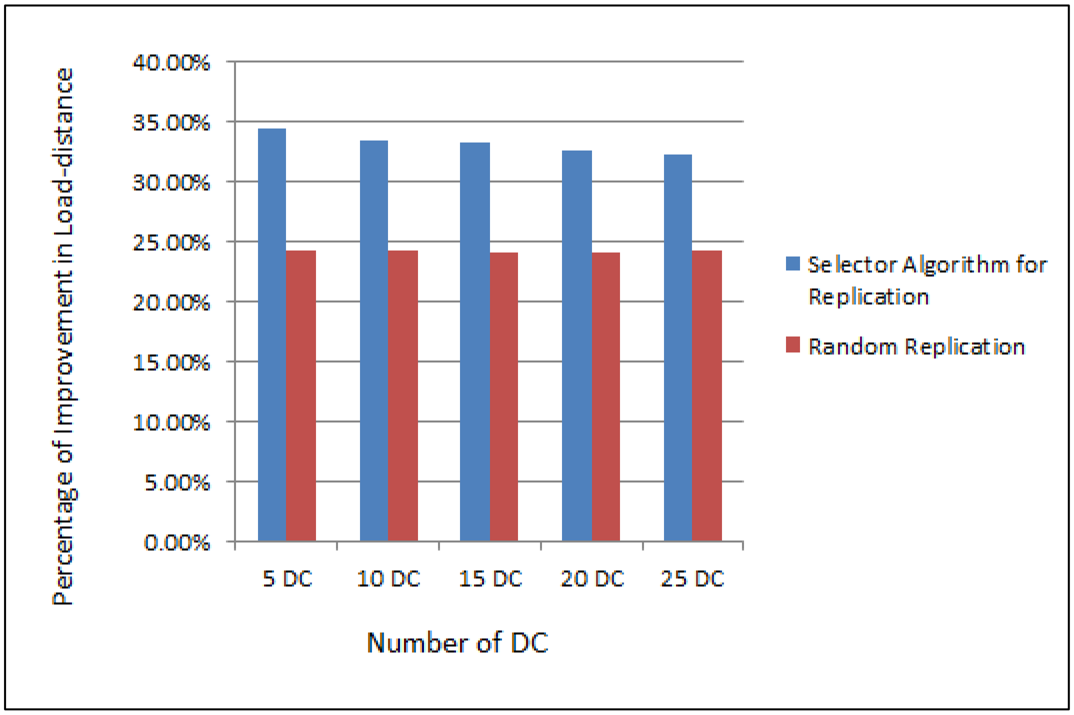
Appendix C Figure 1: 5000VMs/DC, 20K to 30K flows per region, Scenario 1



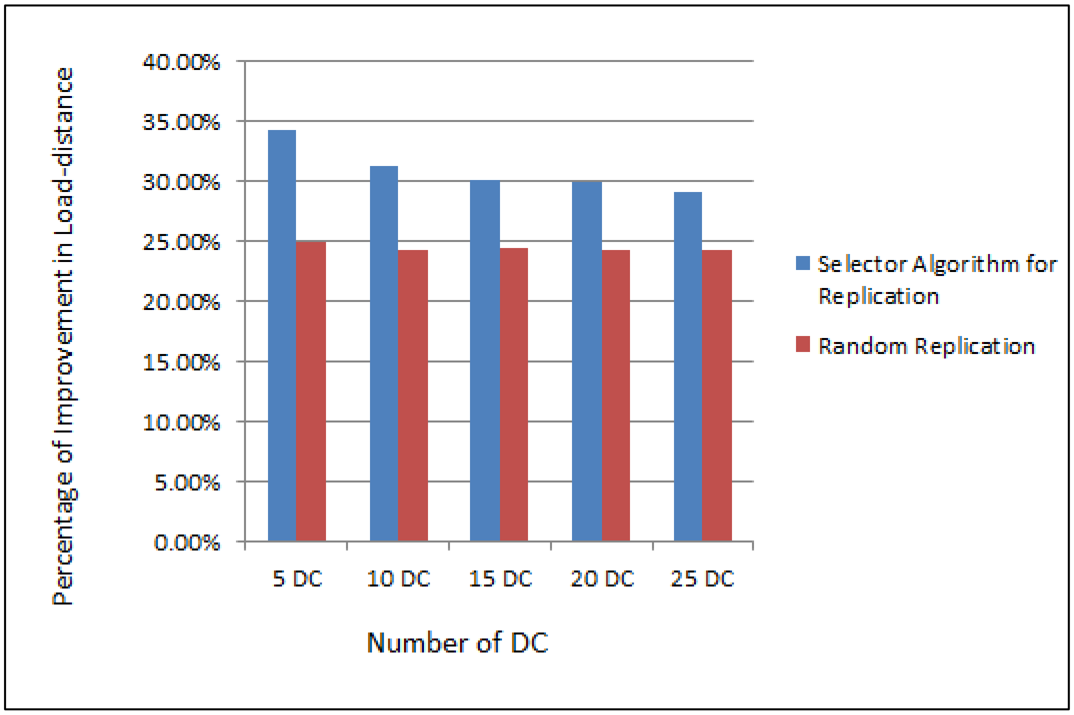
Appendix C Figure 2: 2500VMs/DC, 20K to 30K flows per region, Scenario 1



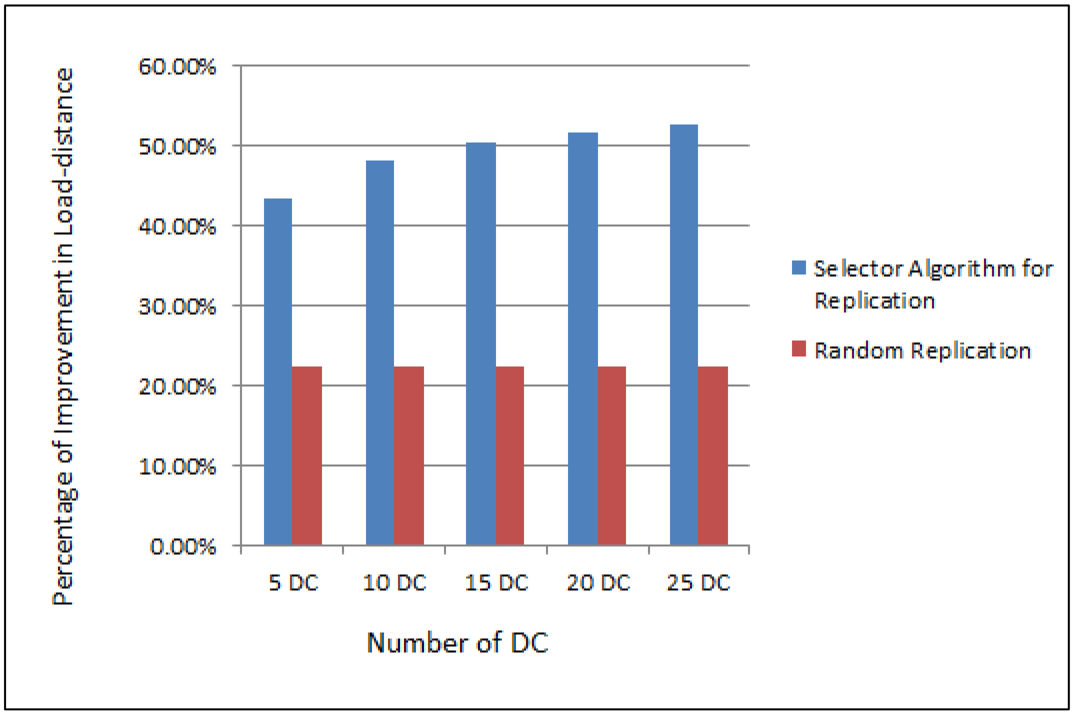
Appendix C Figure 3: 1500VMs/DC, 20K to 30K flows per region, Scenario 1



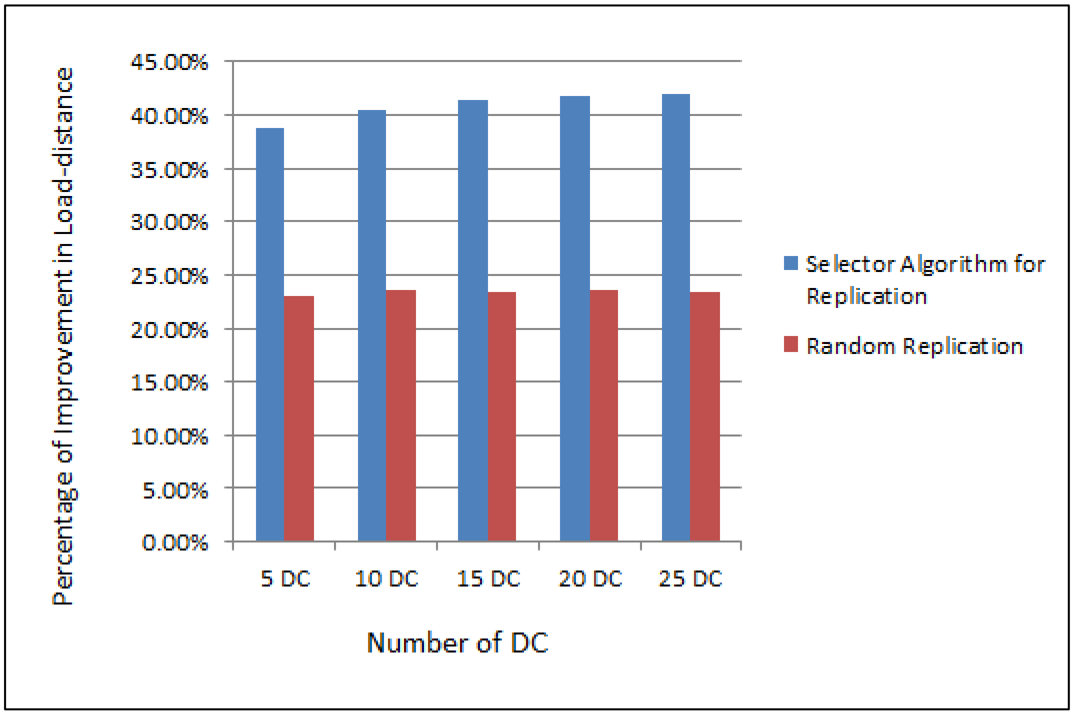
Appendix C Figure 4: 1000VMs/DC, 20K to 30K flows per region, Scenario 1



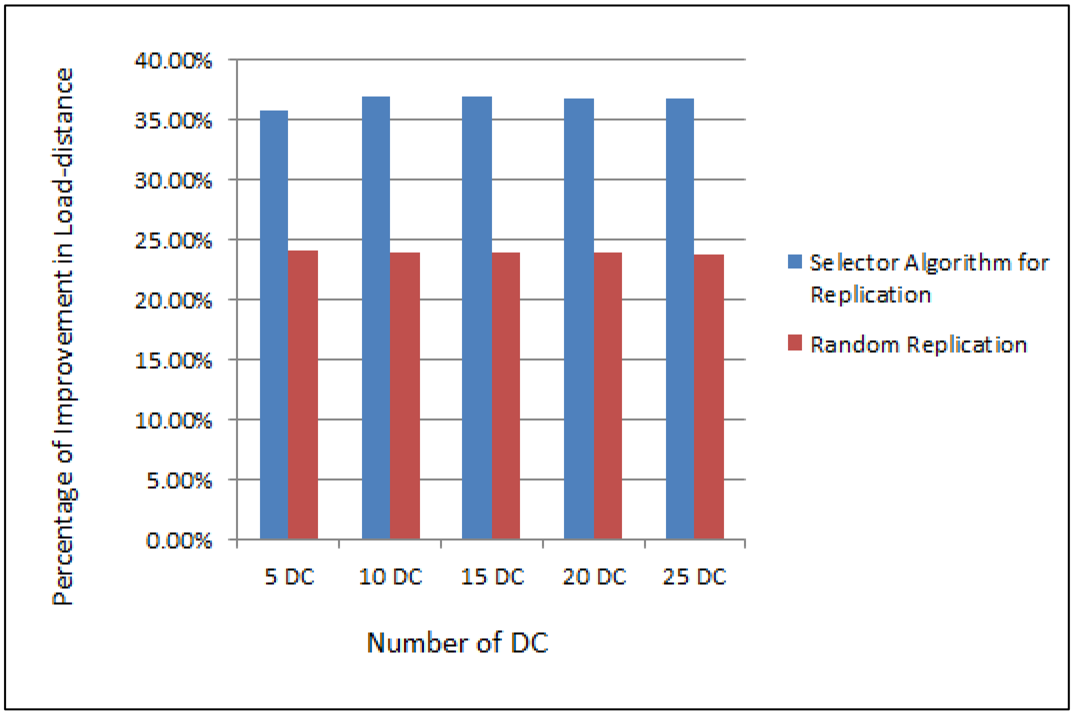
Appendix C Figure 5: 200VMs/DC, 20K to 30K flows per region, Scenario 1



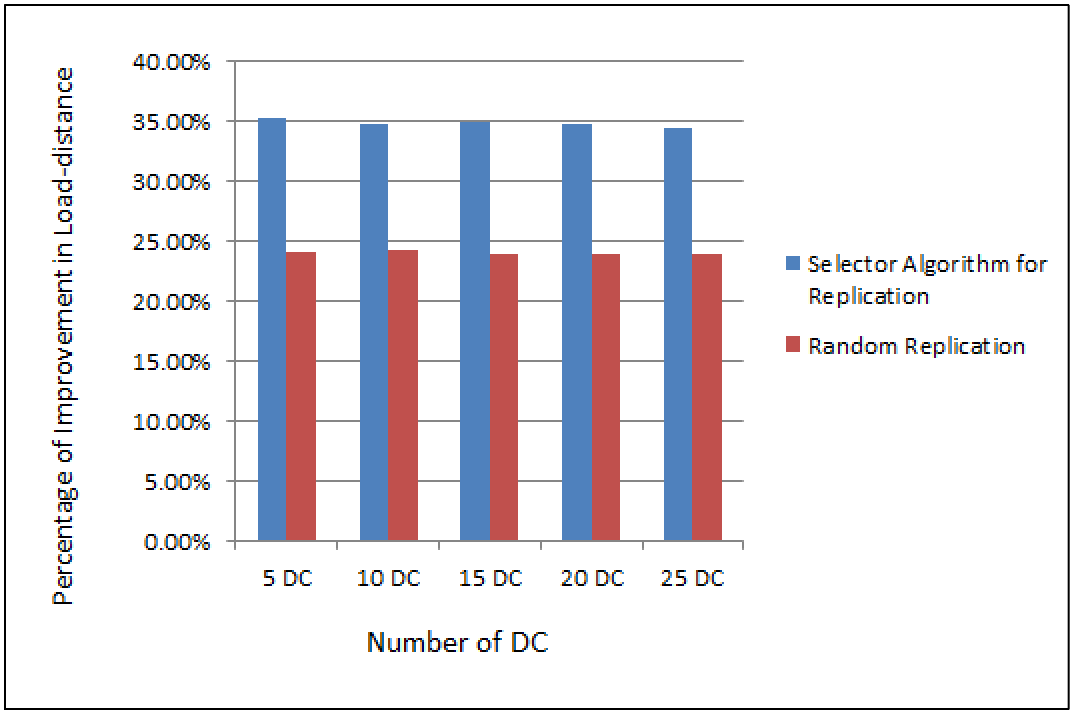
Appendix C Figure 6: 5000VMs/DC, 10K to 20K flows per region, Scenario 1



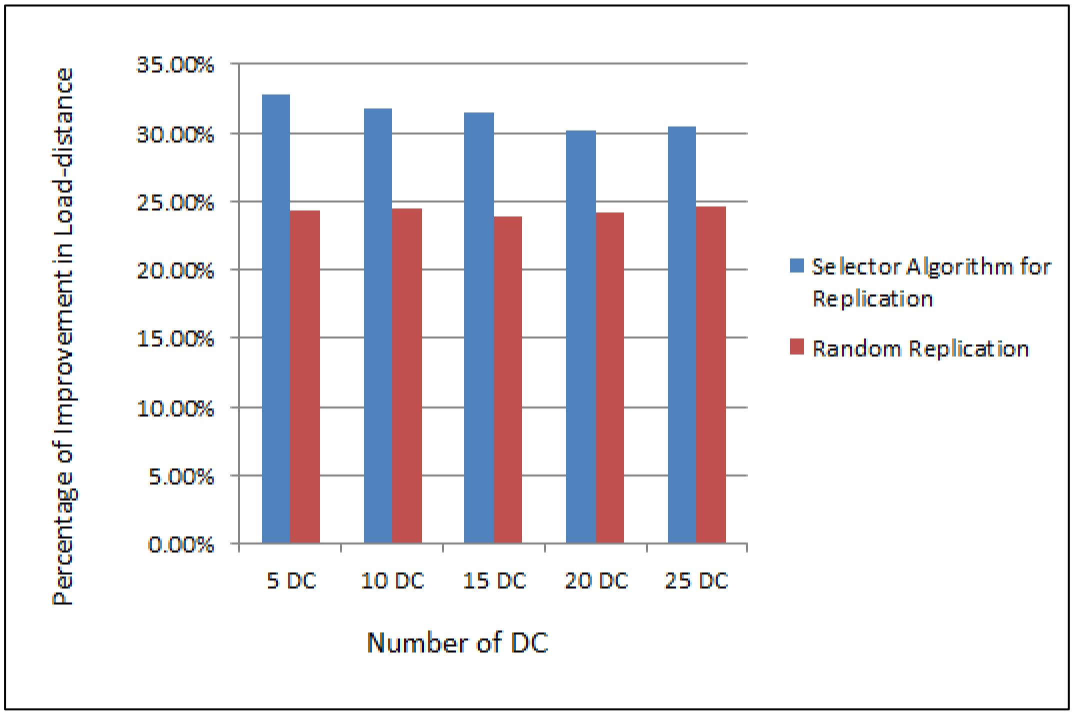
Appendix C Figure 7: 2500VMs/DC, 10K to 20K flows per region, Scenario 1



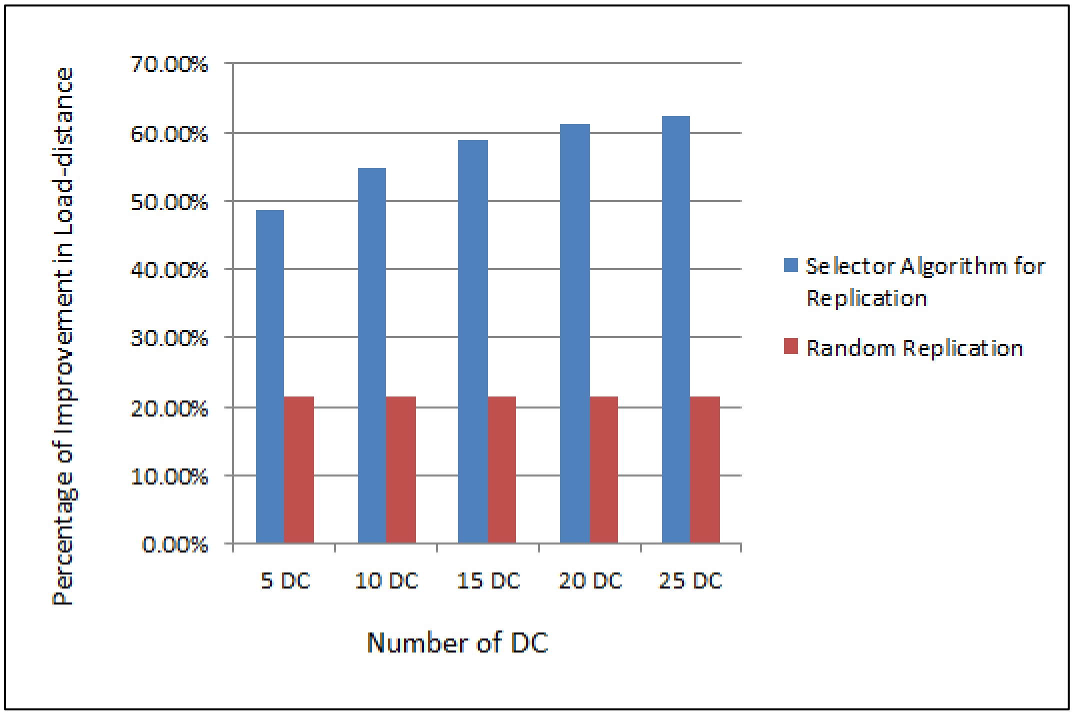
Appendix C Figure 8: 1500VMs/DC, 10K to 20K flows per region, Scenario 1



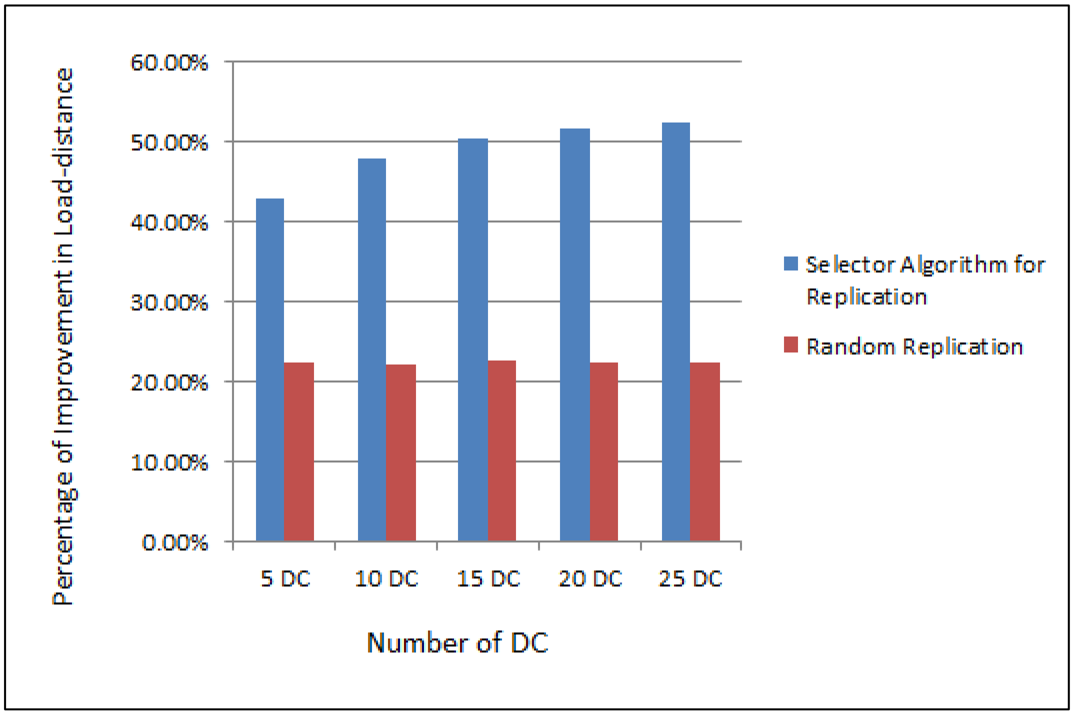
Appendix C Figure 9: 1000VMs/DC, 10K to 20K flows per region, Scenario 1



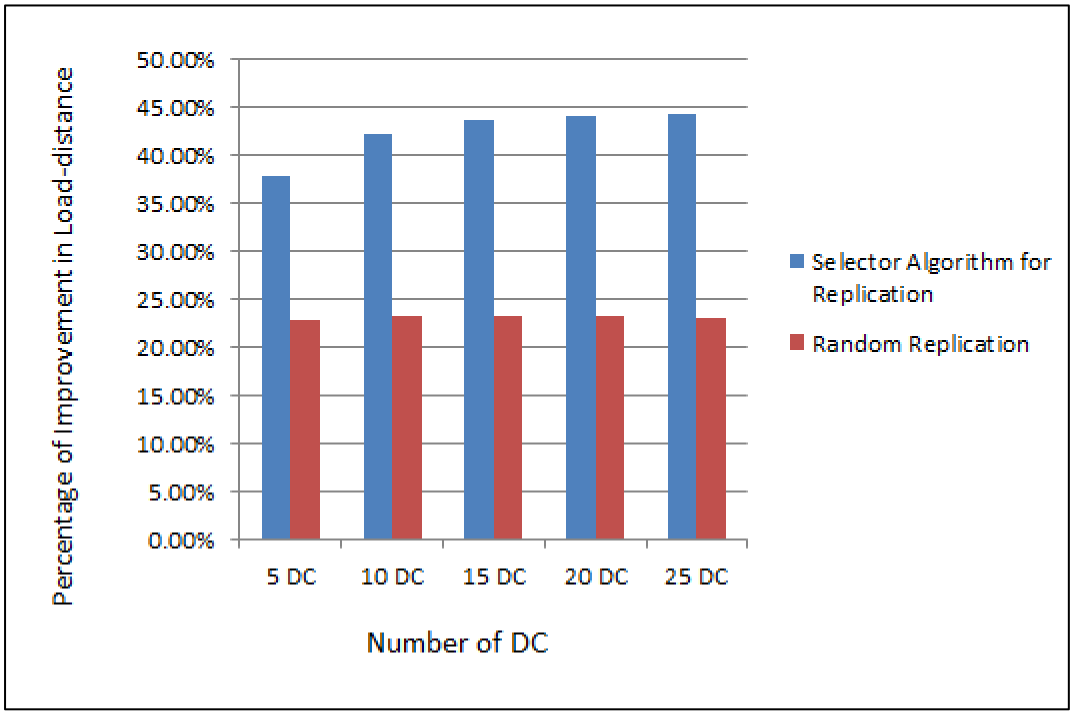
Appendix C Figure 10: 200VMs/DC, 10K to 20K flows per region, Scenario 1



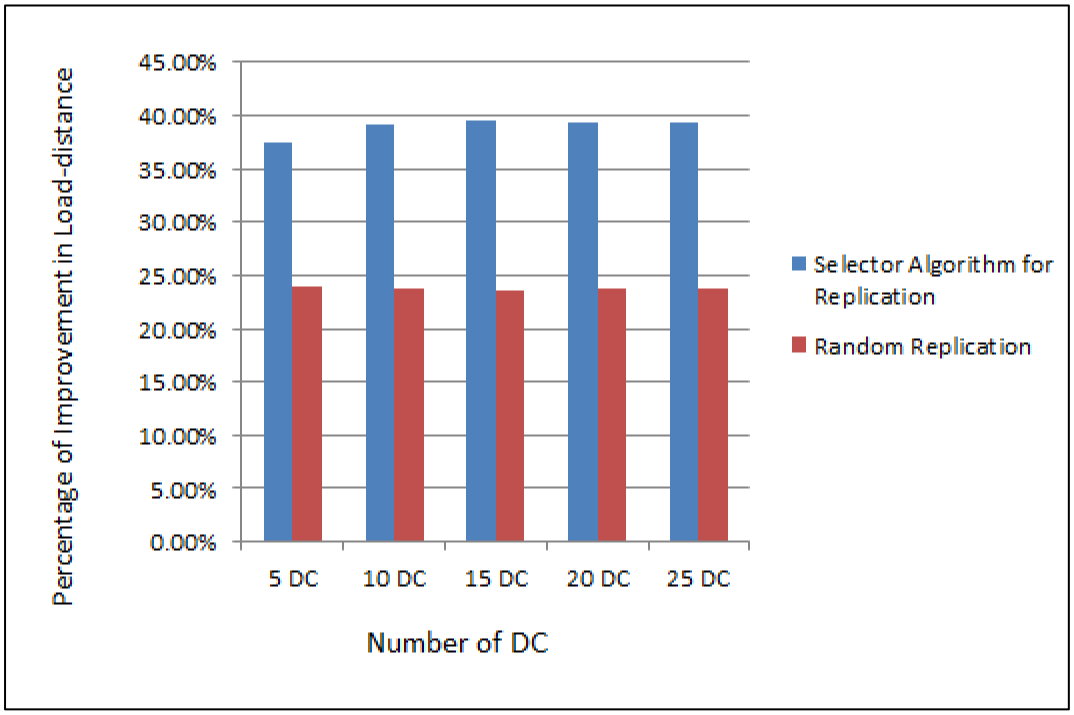
Appendix C Figure 11: 5000VMs/DC, 5K to 10K flows per region, Scenario 1



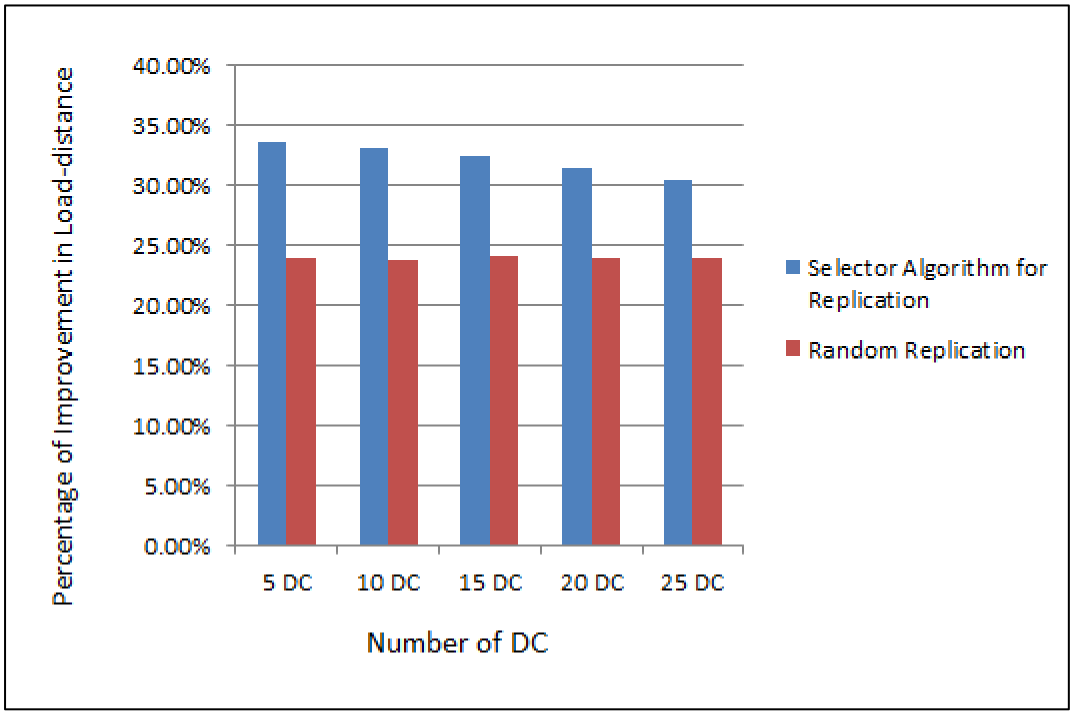
Appendix C Figure 12: 2500VMs/DC, 5K to 10K flows per region, Scenario 1



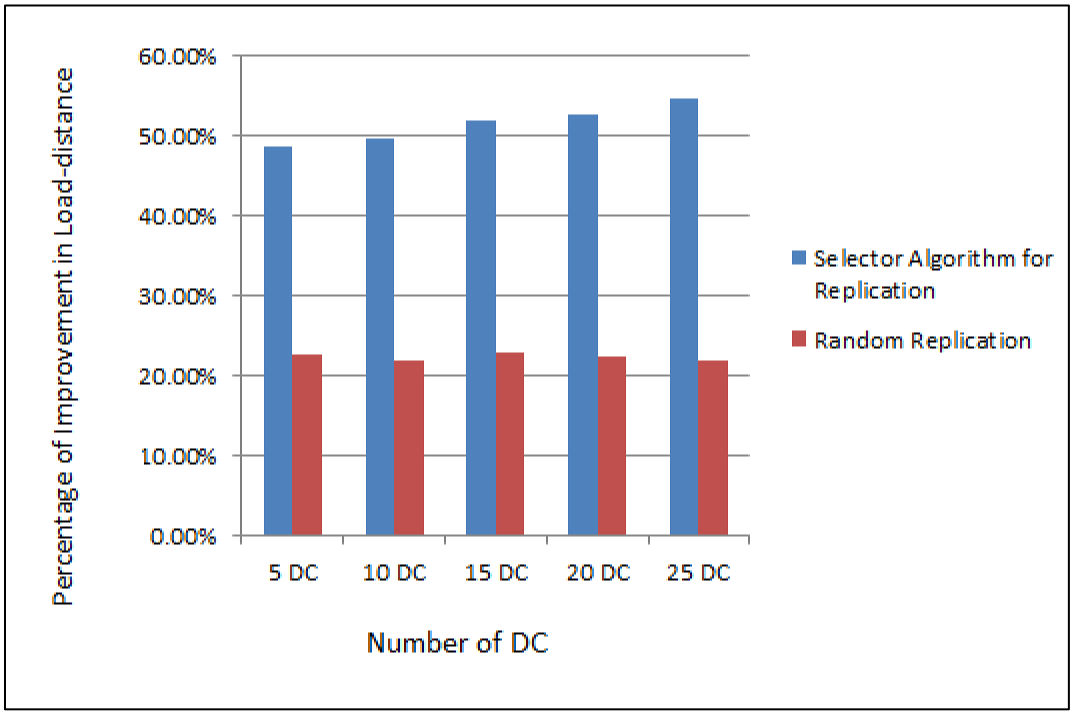
Appendix C Figure 13: 1500VMs/DC, 5K to 10K flows per region, Scenario 1



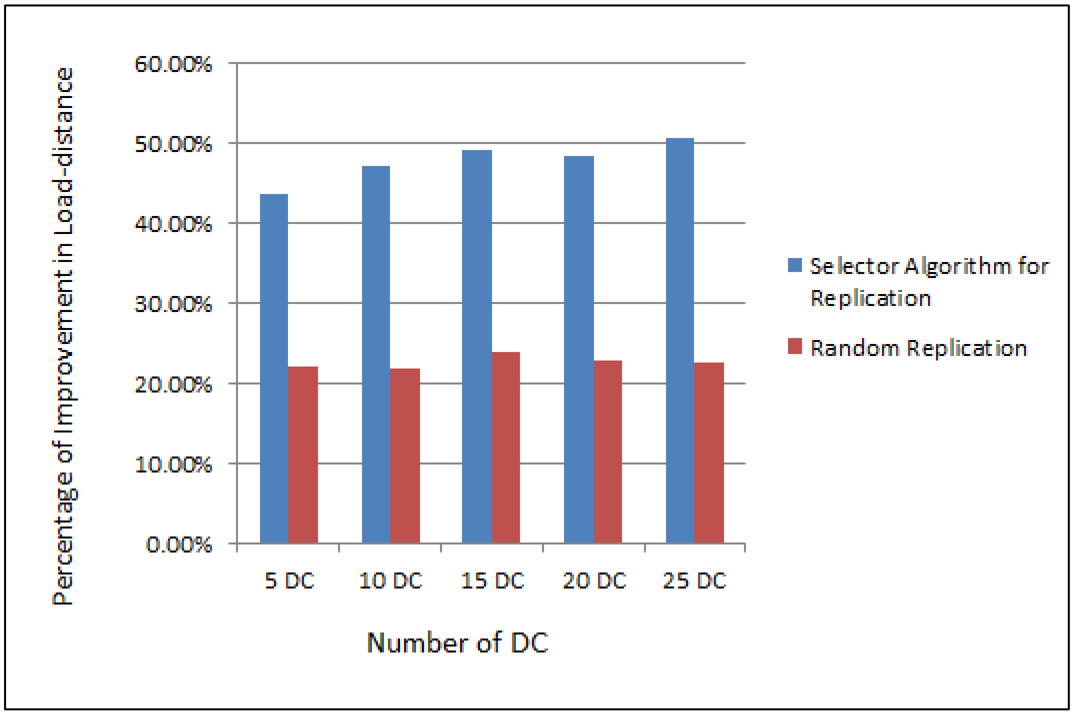
Appendix C Figure 14: 1000VMs/DC, 5K to 10K flows per region, Scenario 1



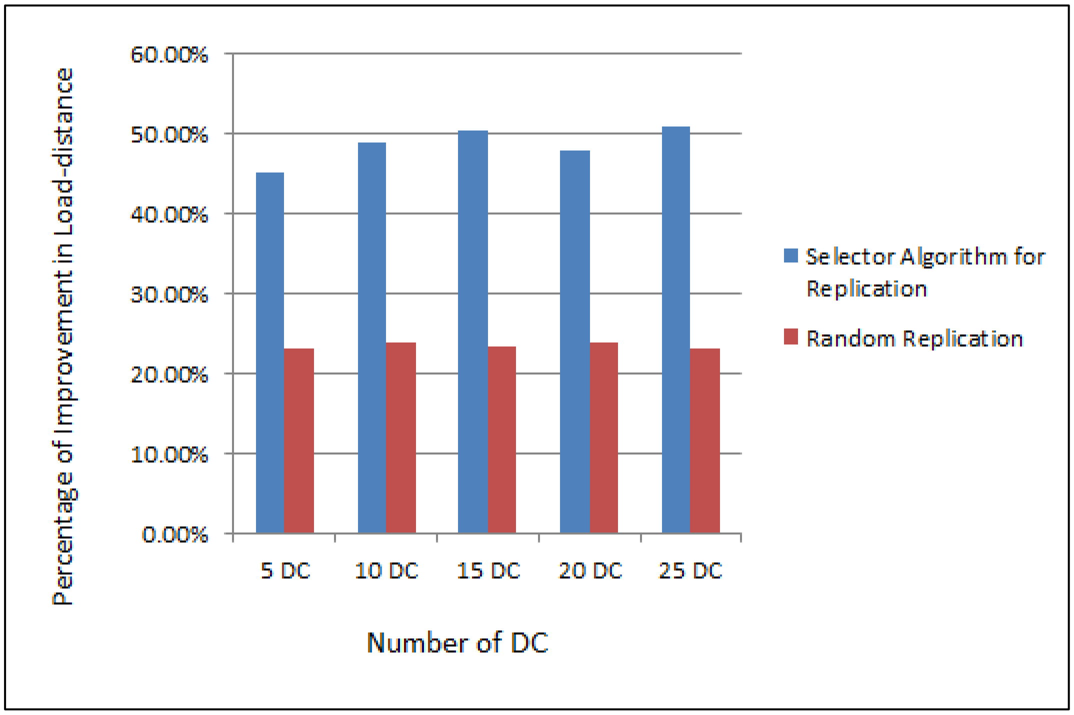
Appendix C Figure 15: 200VMs/DC, 5K to 10K flows per region, Scenario 1



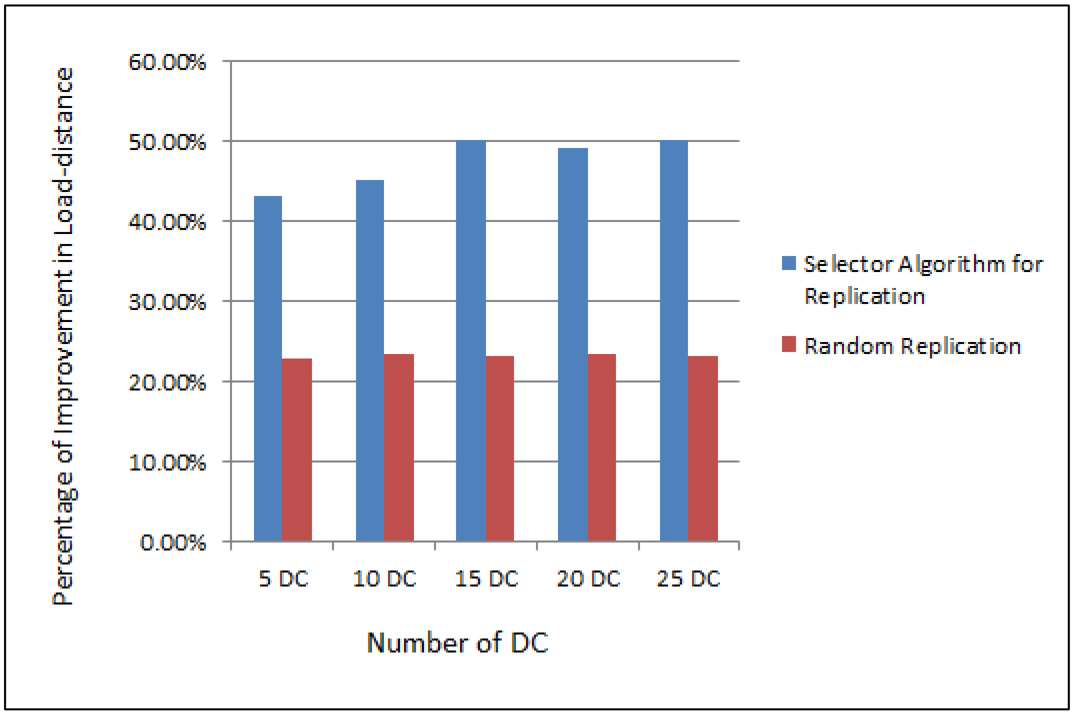
Appendix C Figure 16: 5000VMs/DC, 20K to 30K flows per region, Scenario 4



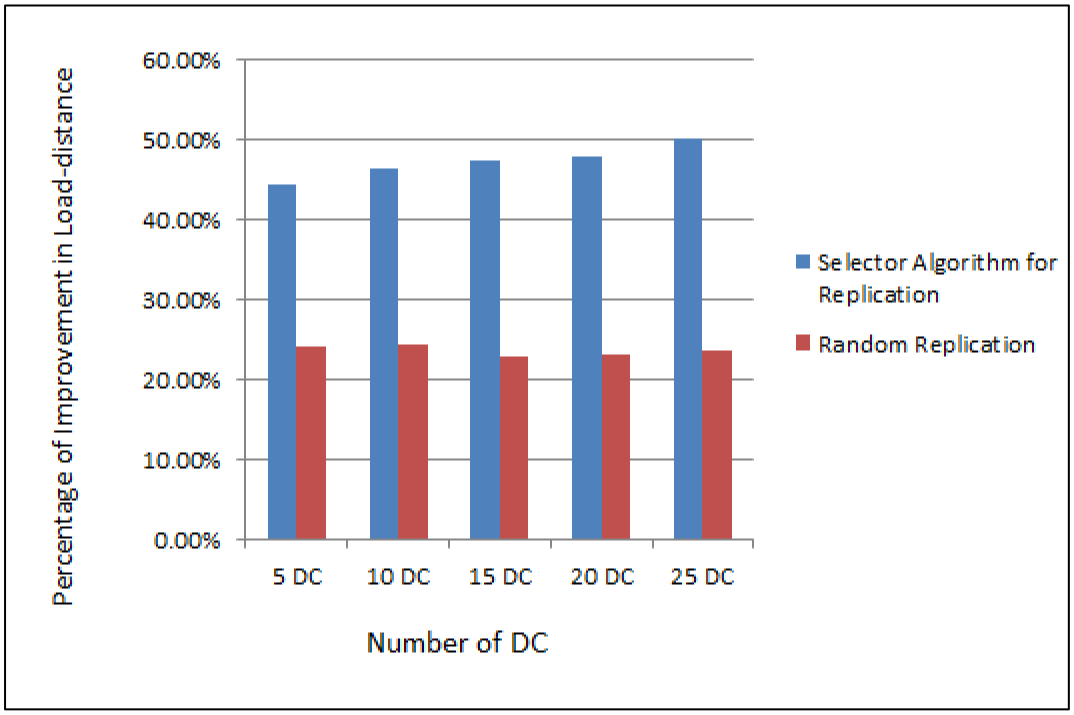
Appendix C Figure 17: 2500VMs/DC, 20K to 30K flows per region, Scenario 4



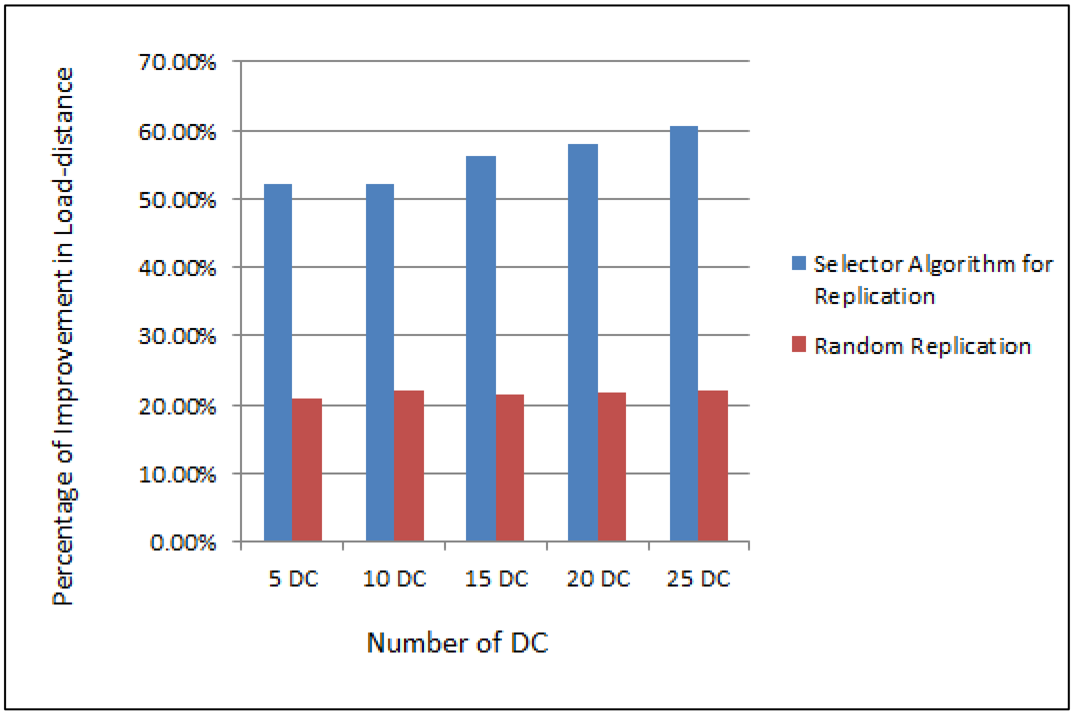
Appendix C Figure 18: 1500VMs/DC, 20K to 30K flows per region, Scenario 4



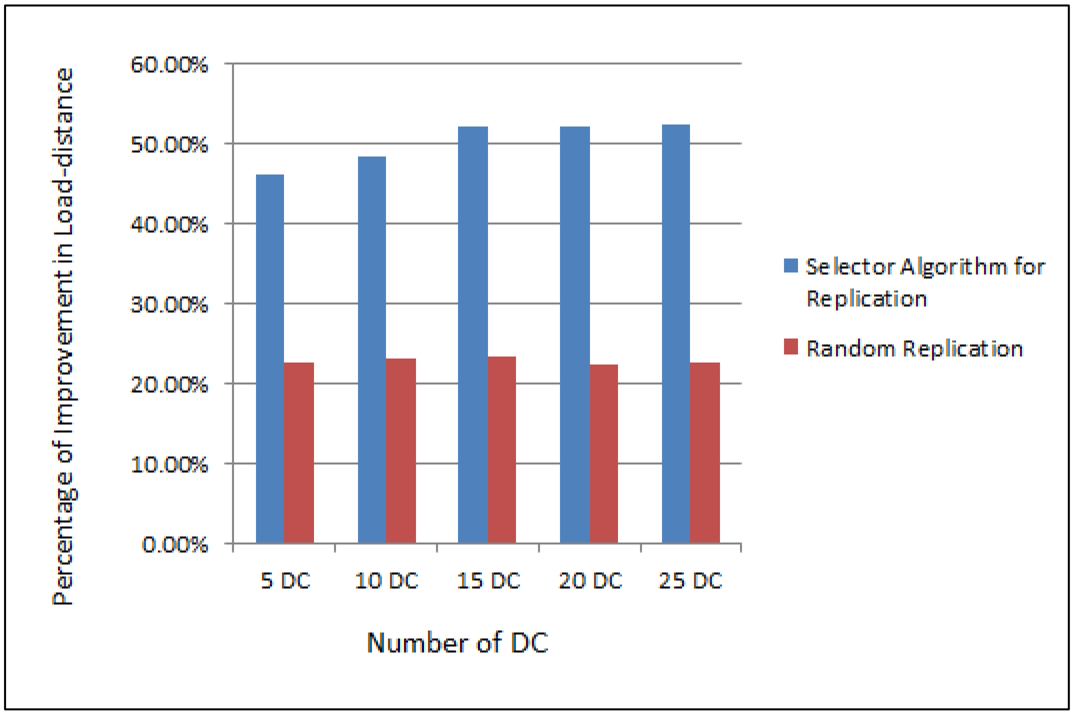
Appendix C Figure 19: 1000VMs/DC, 20K to 30K flows per region, Scenario 4



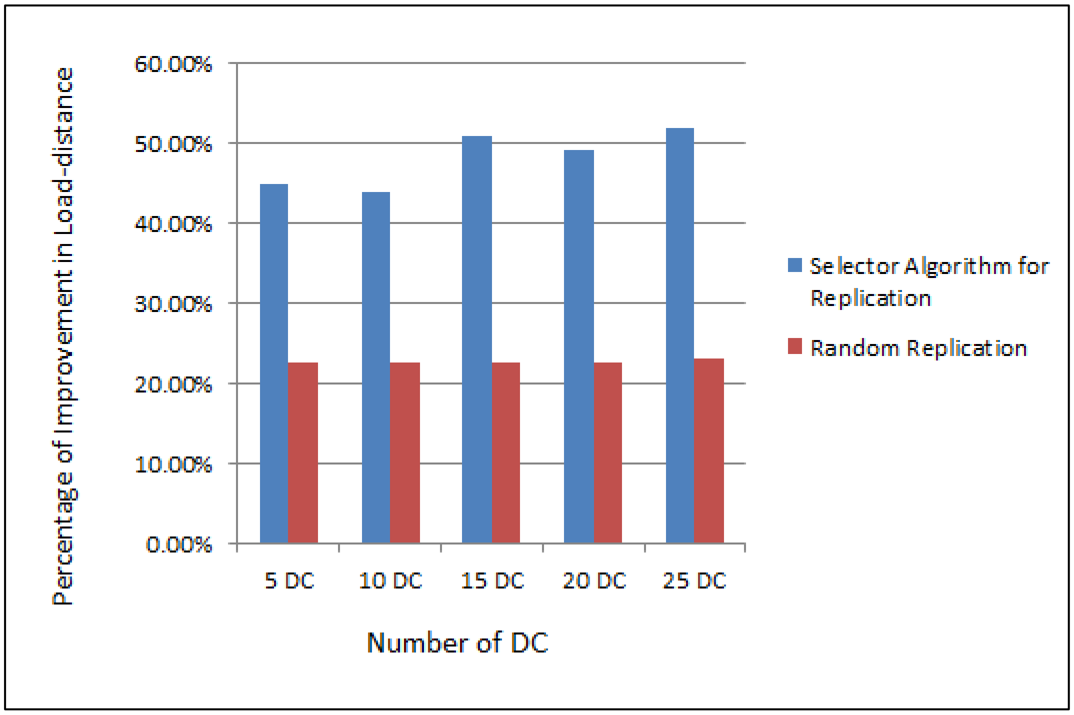
Appendix C Figure 20: 200VMs/DC, 20K to 30K flows per region, Scenario 4



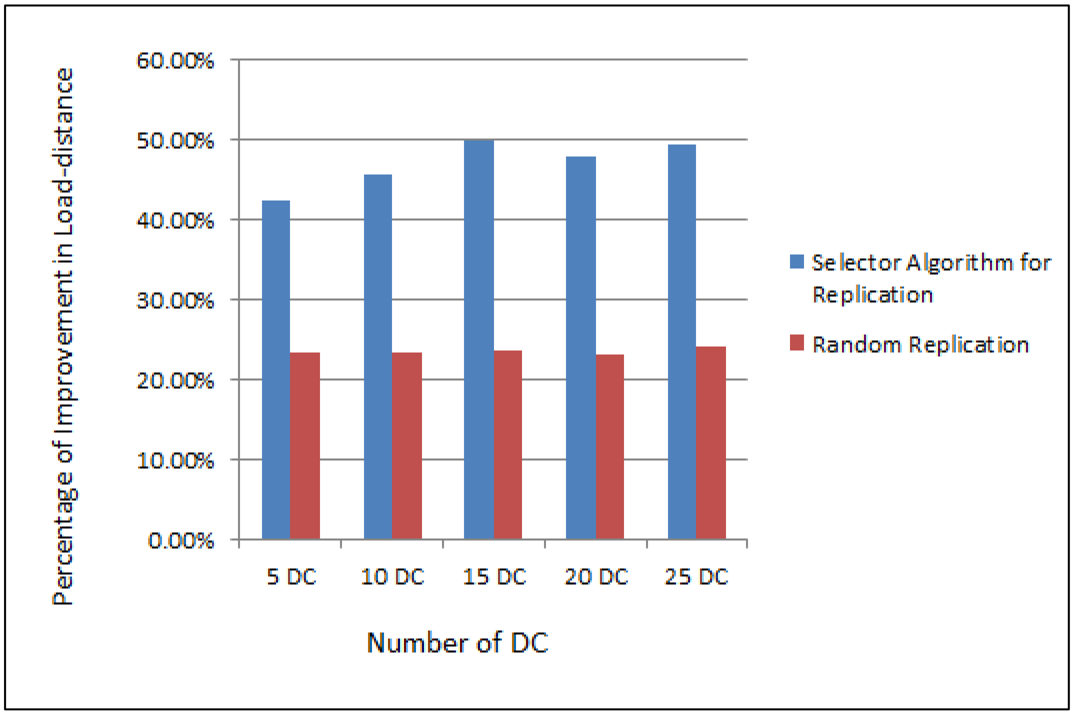
Appendix C Figure 21: 5000VMs/DC, 10K to 20K flows per region, Scenario 4



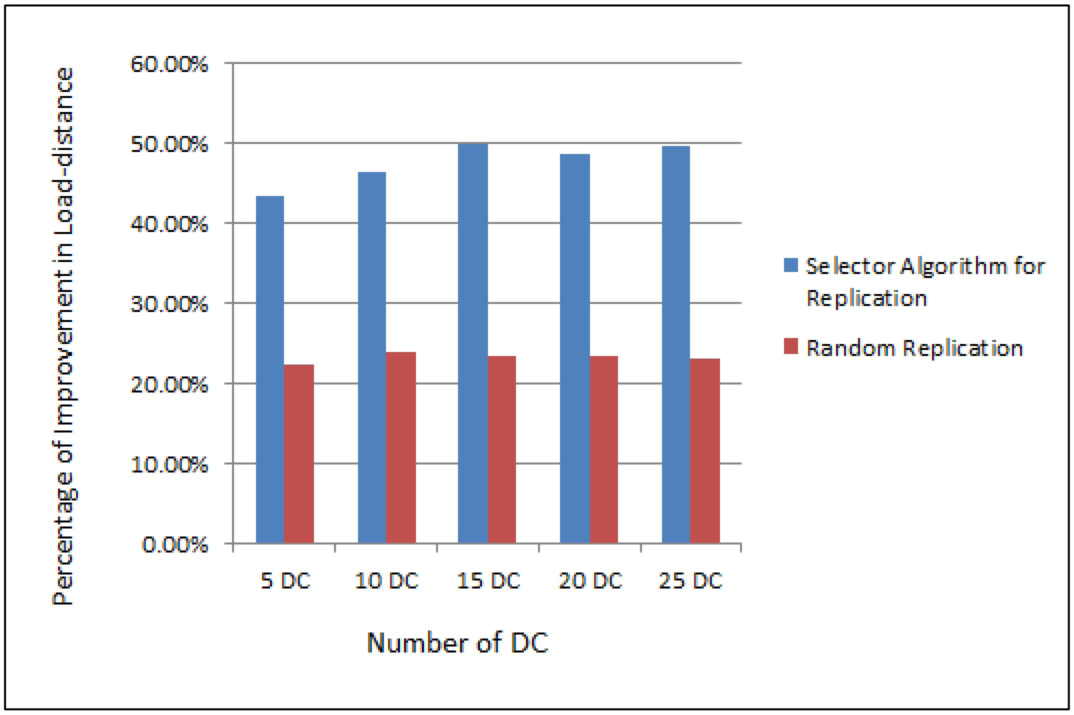
Appendix C Figure 22: 2500VMs/DC, 10K to 20K flows per region, Scenario 4



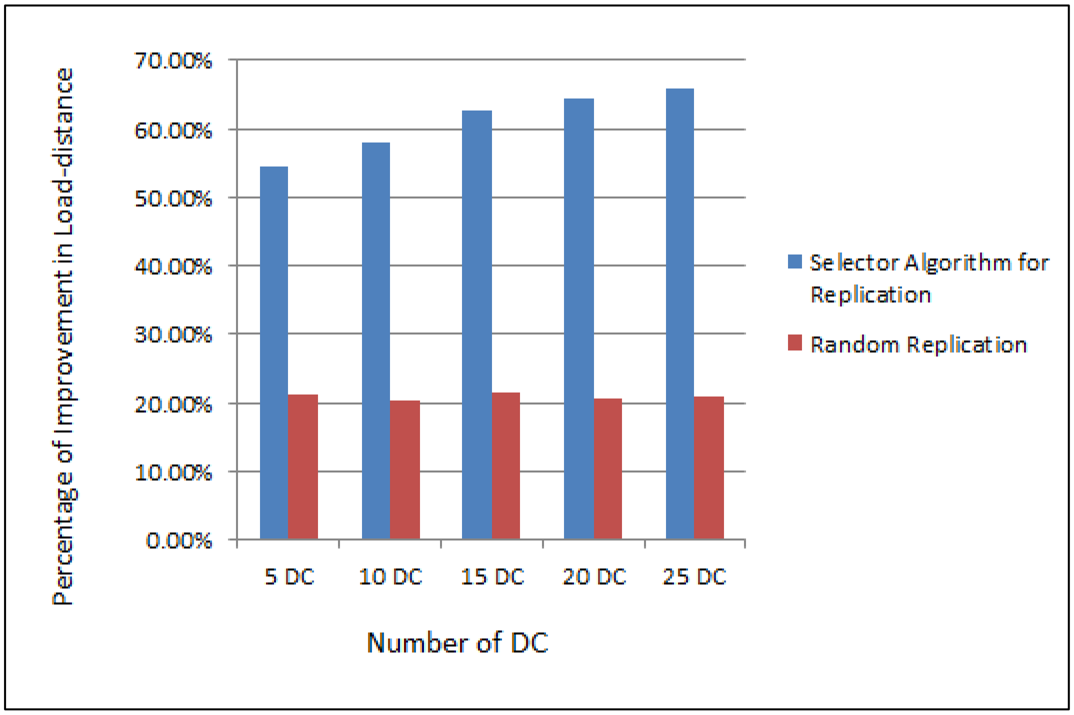
Appendix C Figure 23: 1500VMs/DC, 10K to 20K flows per region, Scenario 4



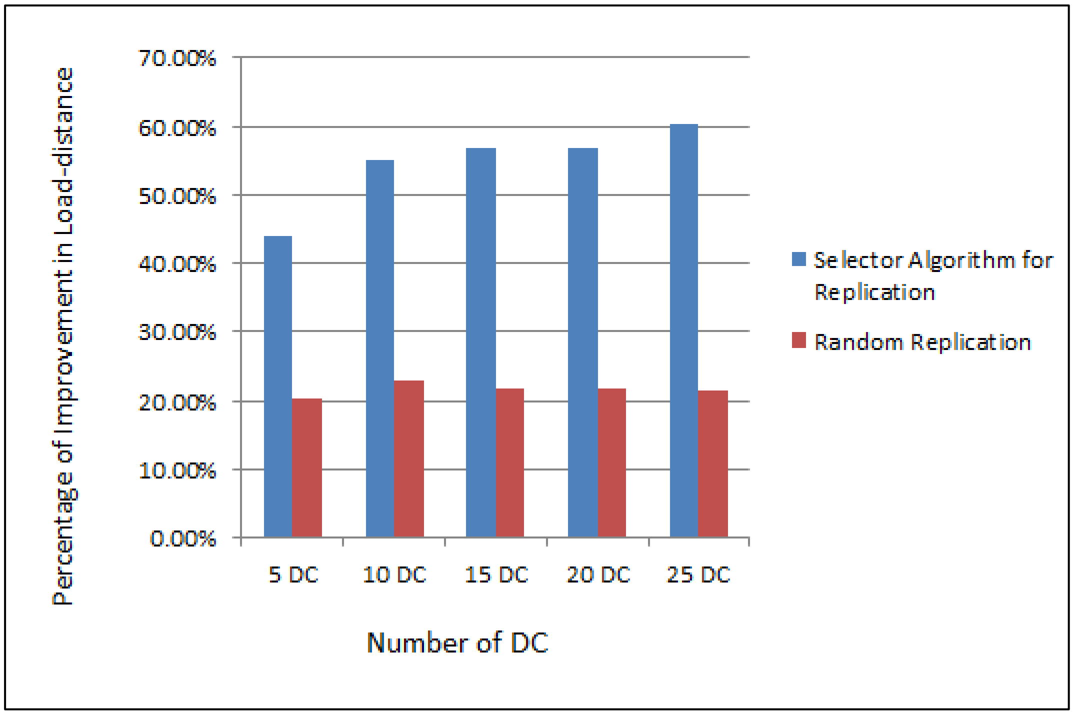
Appendix C Figure 24: 1000VMs/DC, 10K to 20K flows per region, Scenario 4



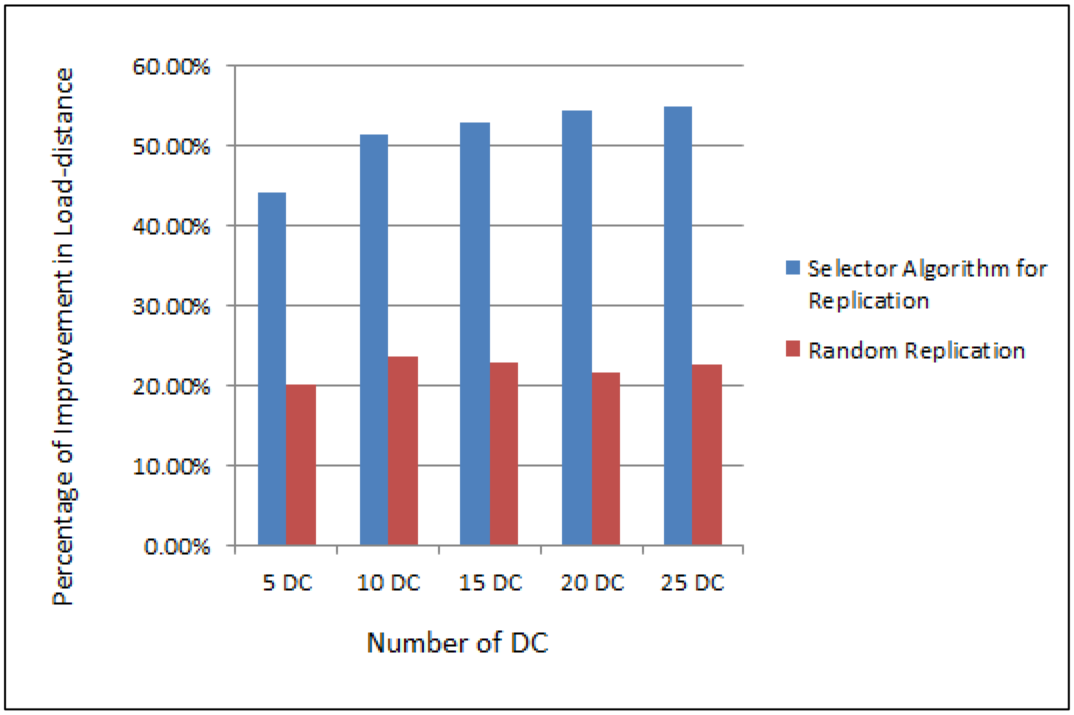
Appendix C Figure 25: 200VMs/DC, 10K to 20K flows per region, Scenario 4



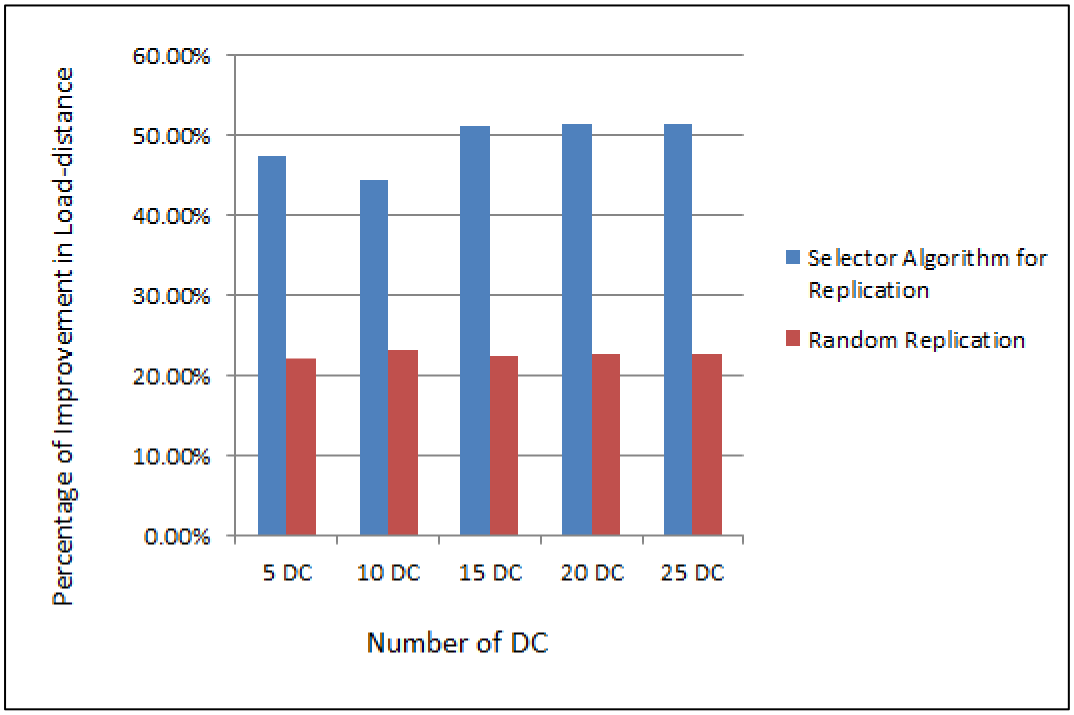
Appendix C Figure 26: 5000VMs/DC, 5K to 10K flows per region, Scenario 4



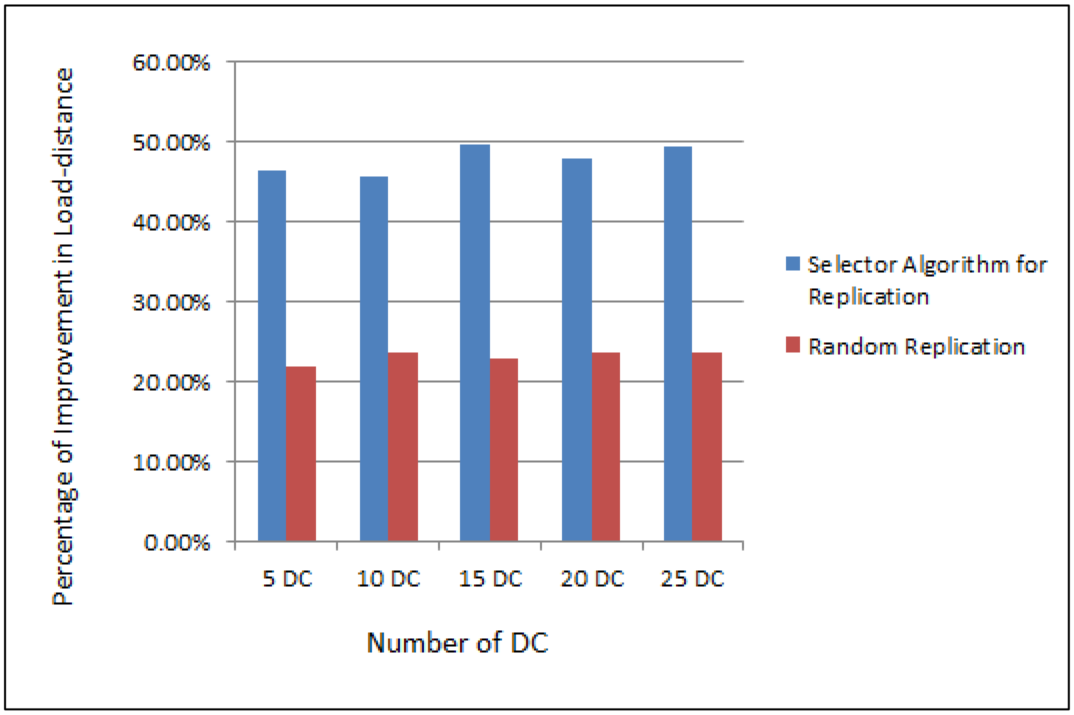
Appendix C Figure 27: 2500VMs/DC, 5K to 10K flows per region, Scenario 4



Appendix C Figure 28: 1500VMs/DC, 5K to 10K flows per region, Scenario 4

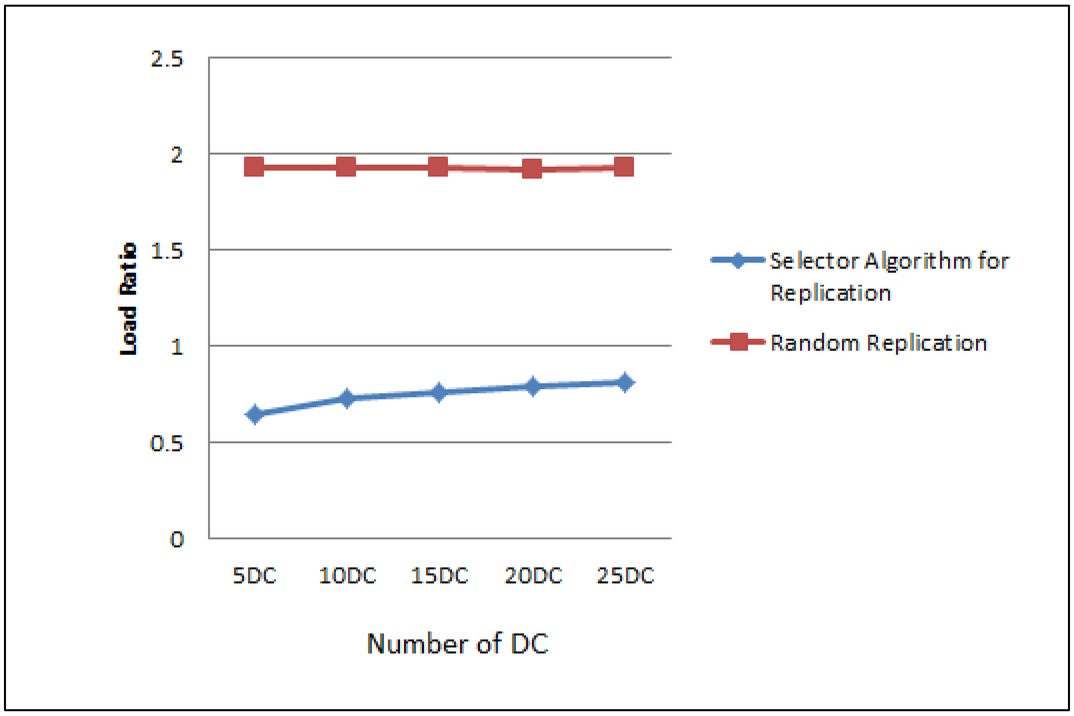


Appendix C Figure 29: 1000VMs/DC, 5K to 10K flows per region, Scenario 4

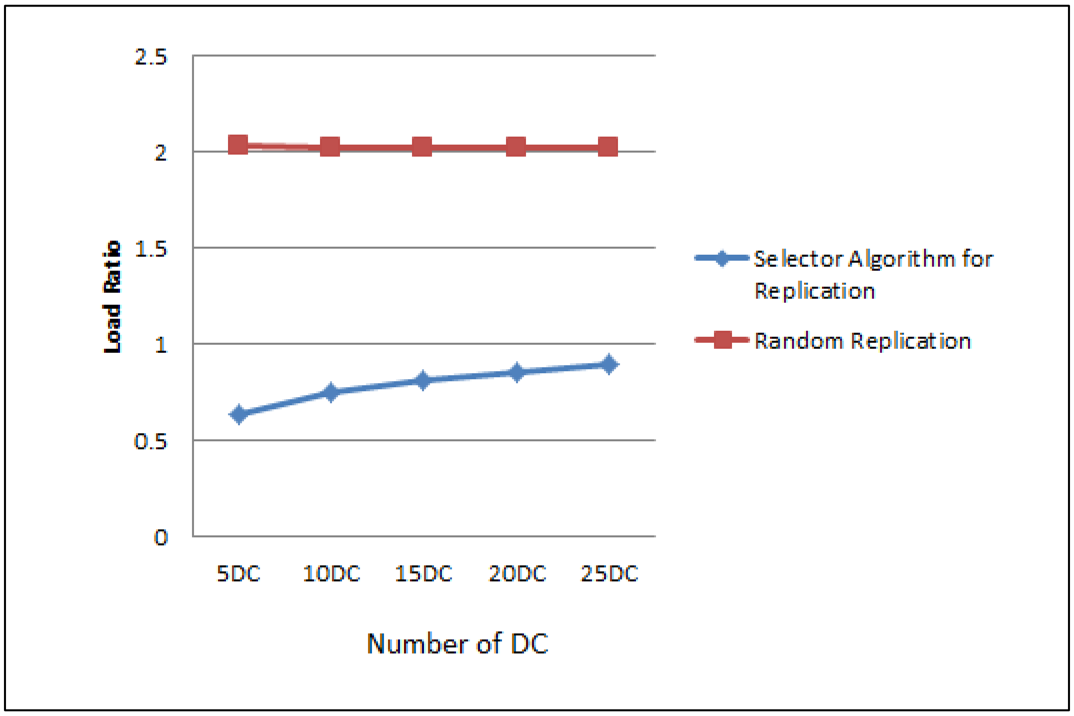


Appendix C Figure 30: 200VMs/DC, 5K to 10K flows per region, Scenario 4

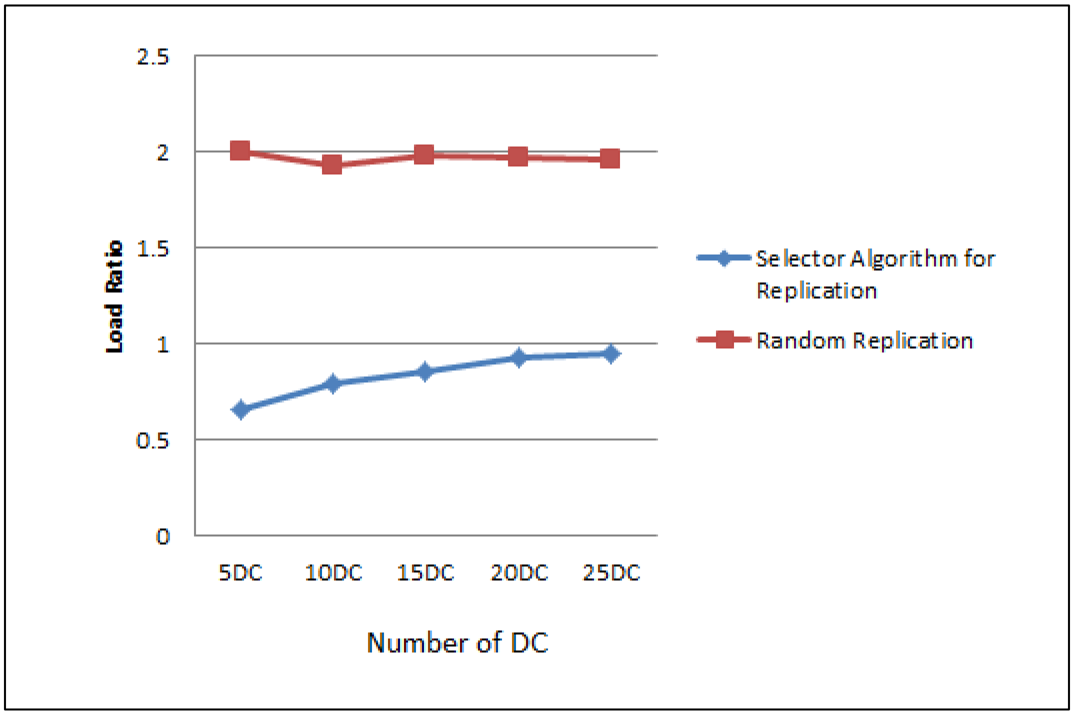
Appendix D: Load Ratio for Replication Algorithm



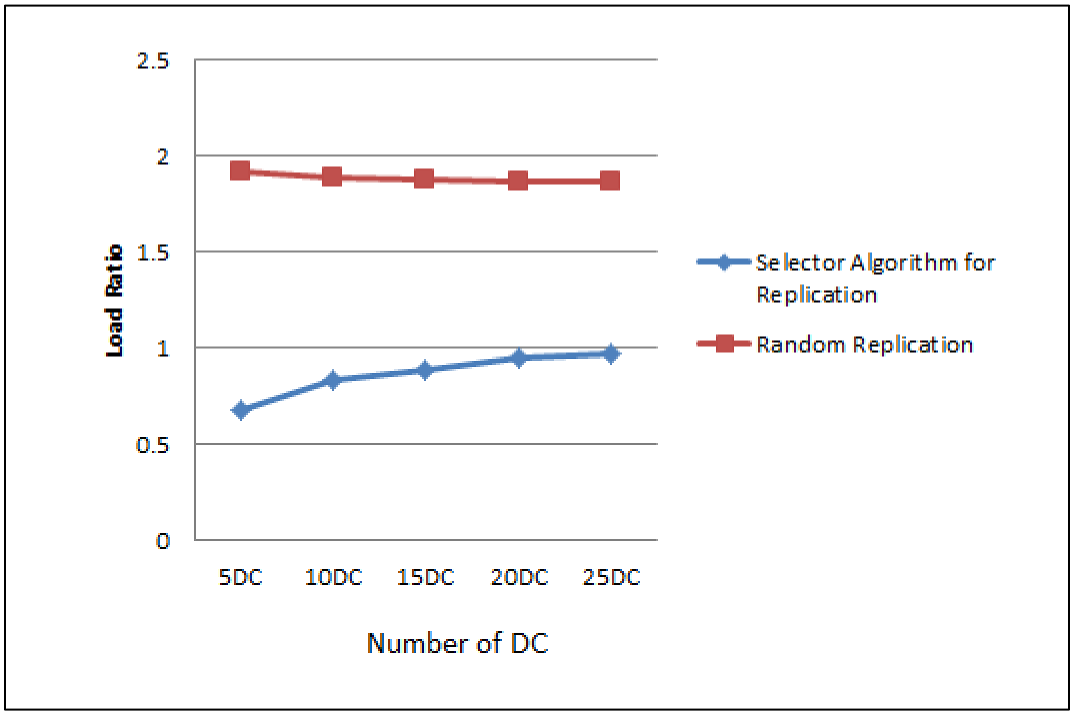
Appendix D Figure 1: 5000VMs/DC, 20K to 30K flows per region, Scenario 1



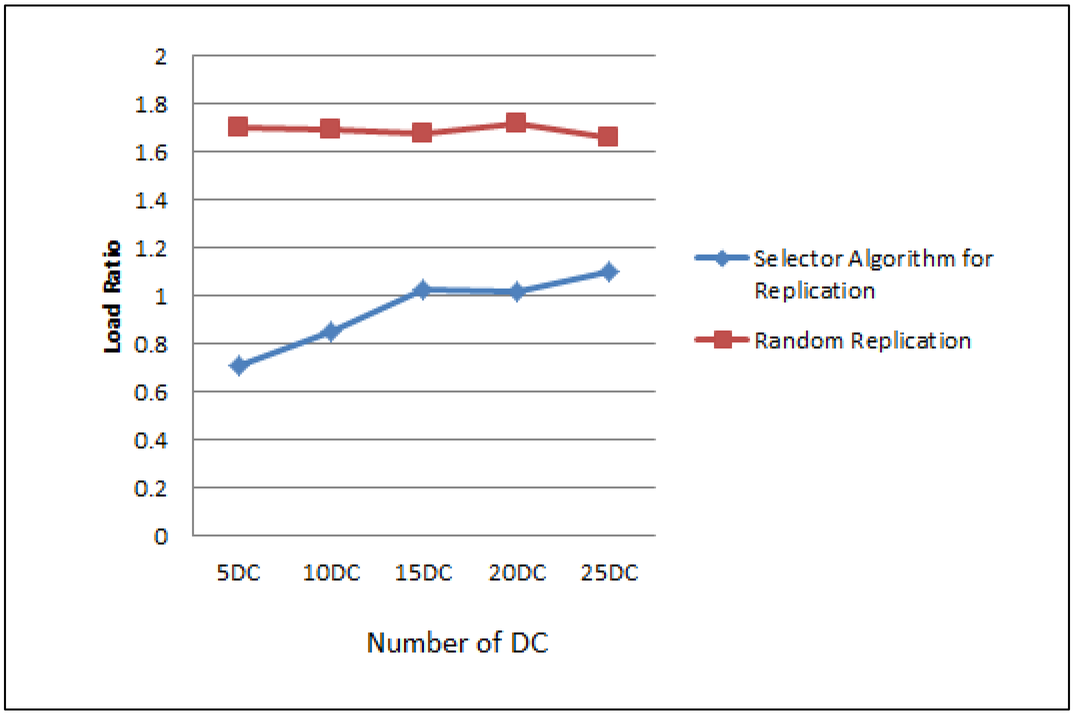
Appendix D Figure 2: 2500VMs/DC, 20K to 30K flows per region, Scenario 1



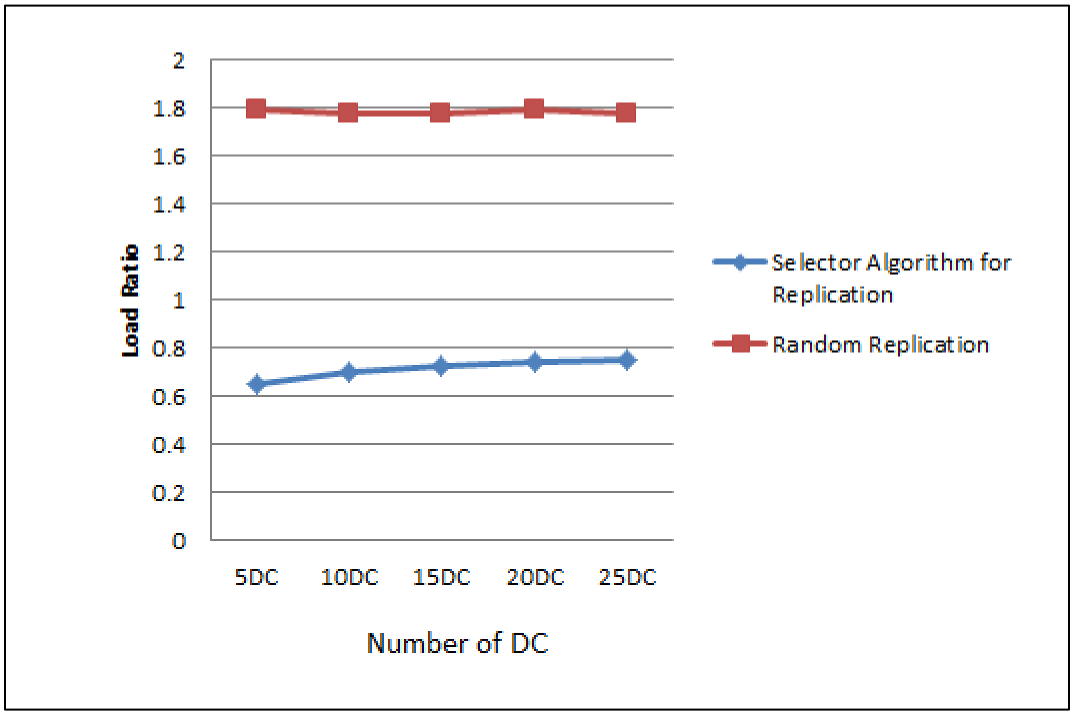
Appendix D Figure 3: 1500VMs/DC, 20K to 30K flows per region, Scenario 1



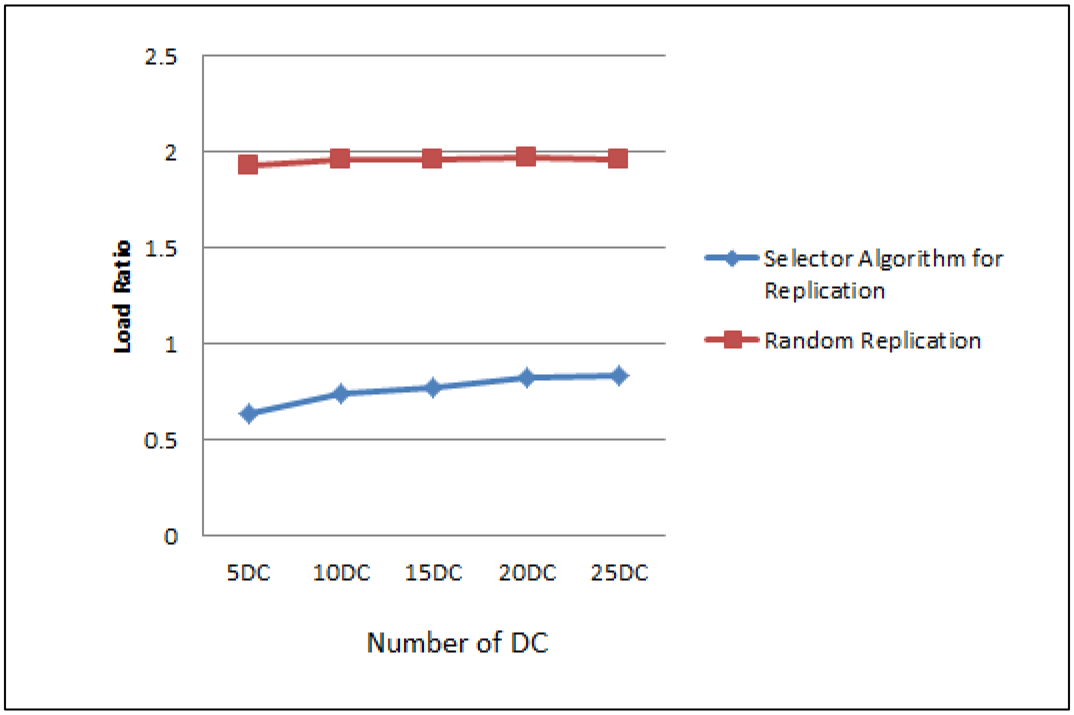
Appendix D Figure 4: 1000VMs/DC, 20K to 30K flows per region, Scenario 1



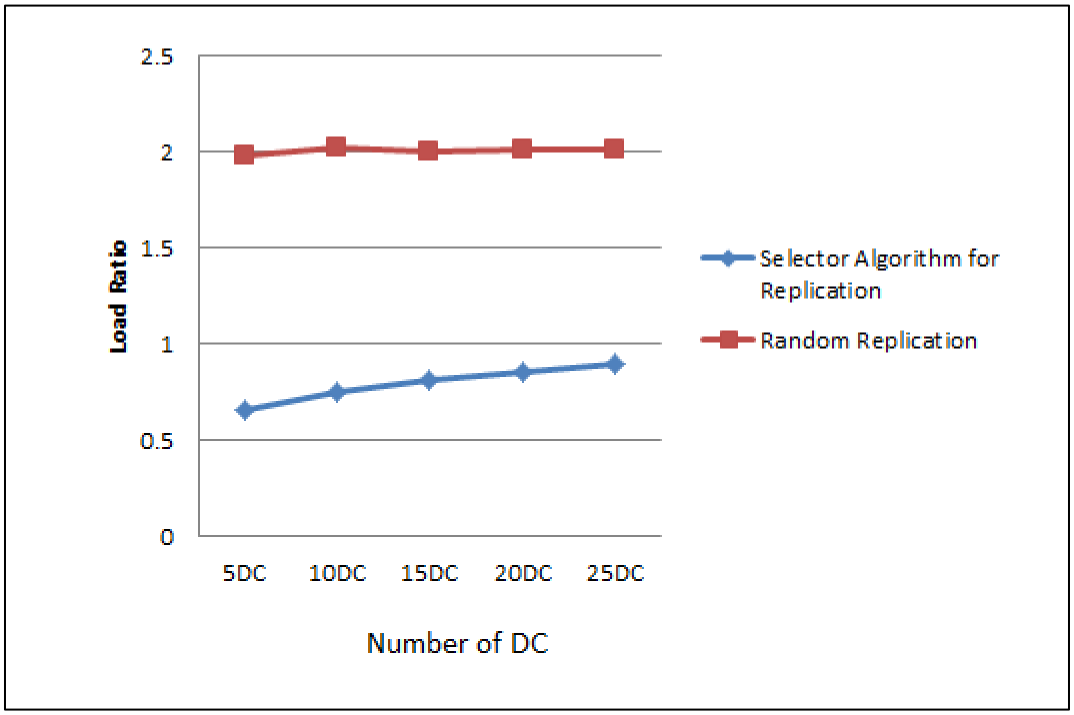
Appendix D Figure 5: 200VMs/DC, 20K to 30K flows per region, Scenario 1



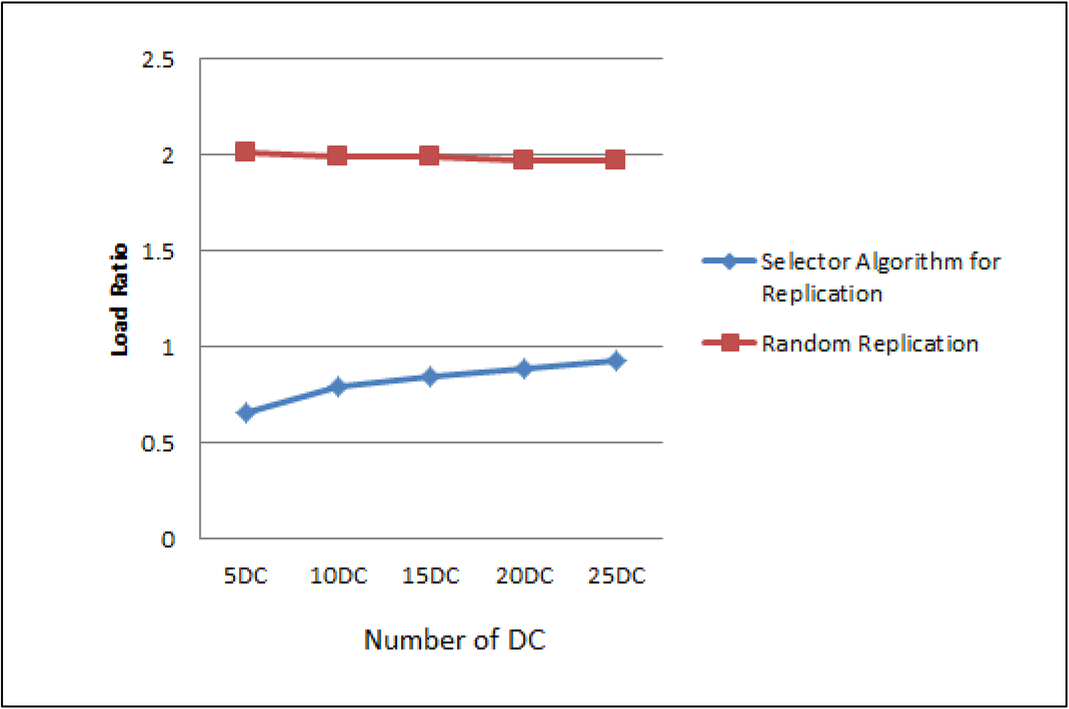
Appendix D Figure 6: 5000VMs/DC, 10K to 20K flows per region, Scenario 1



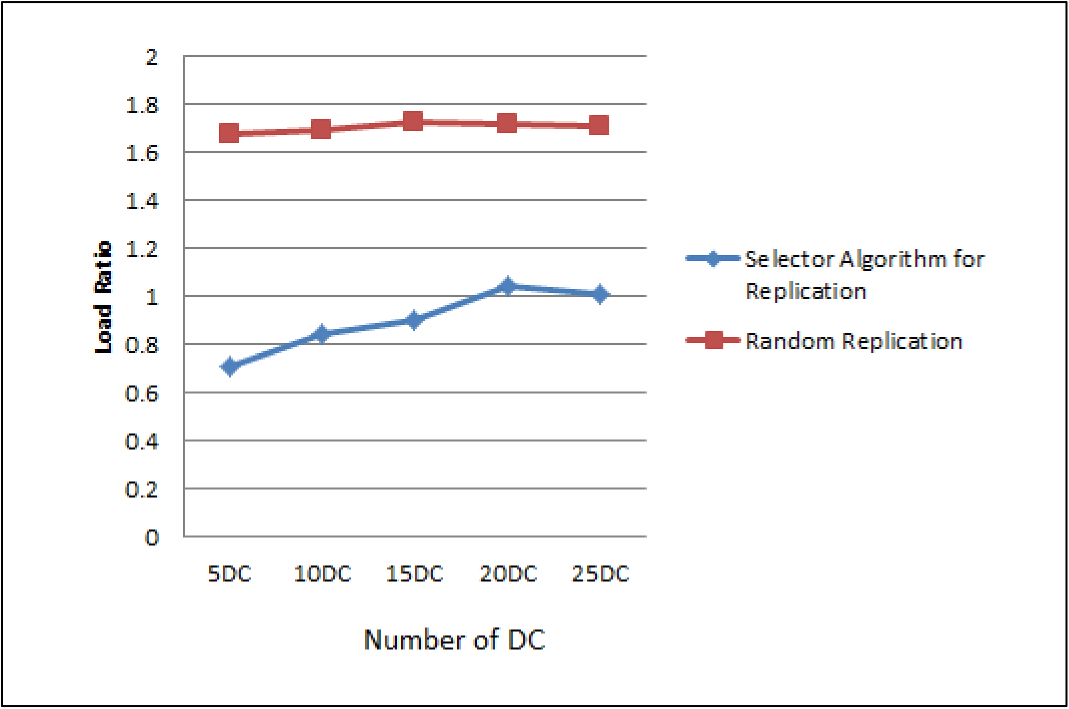
Appendix D Figure 7: 2500VMs/DC, 10K to 20K flows per region, Scenario 1



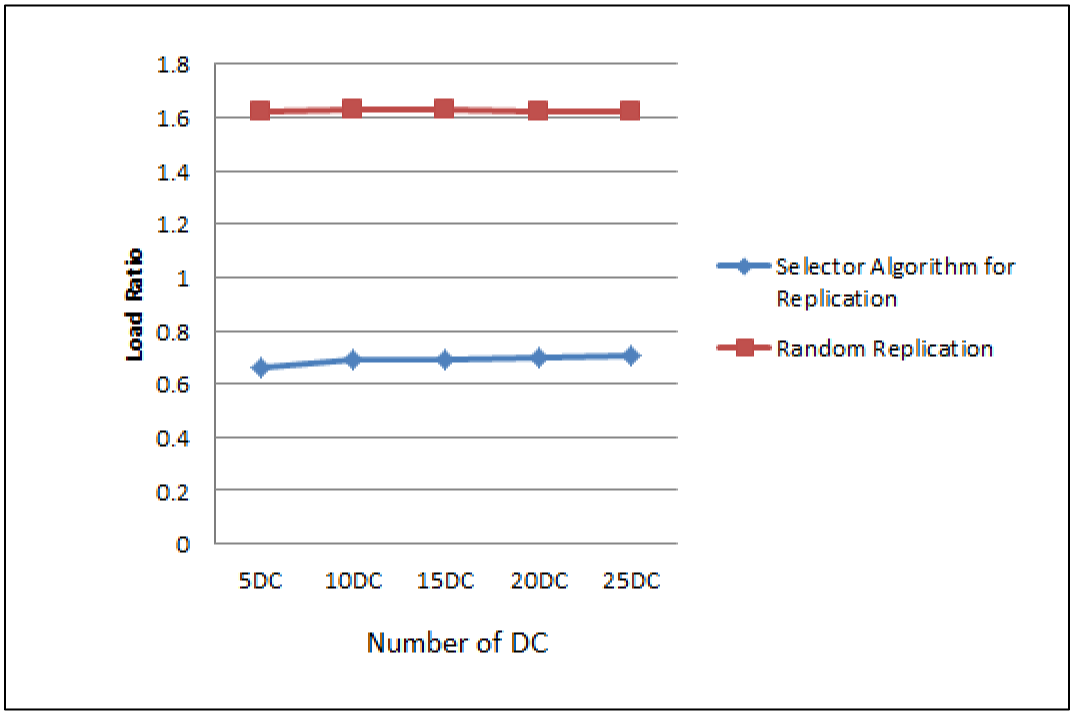
Appendix D Figure 8: 1500VMs/DC, 10K to 20K flows per region, Scenario 1



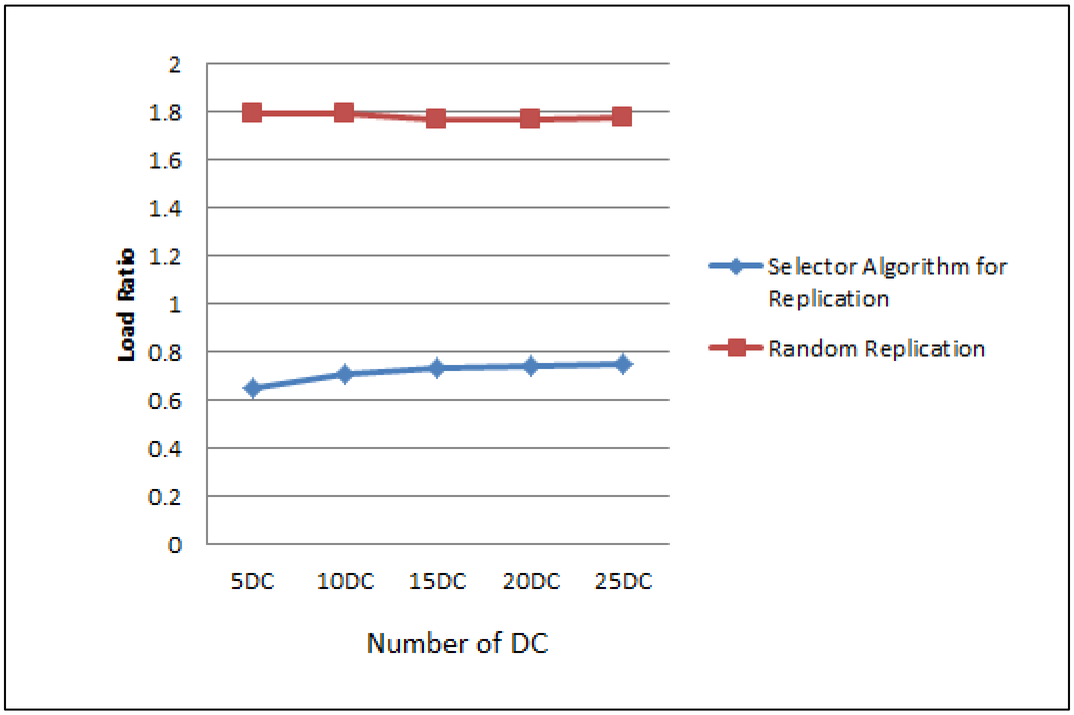
Appendix D Figure 9: 1000VMs/DC, 10K to 20K flows per region, Scenario 1



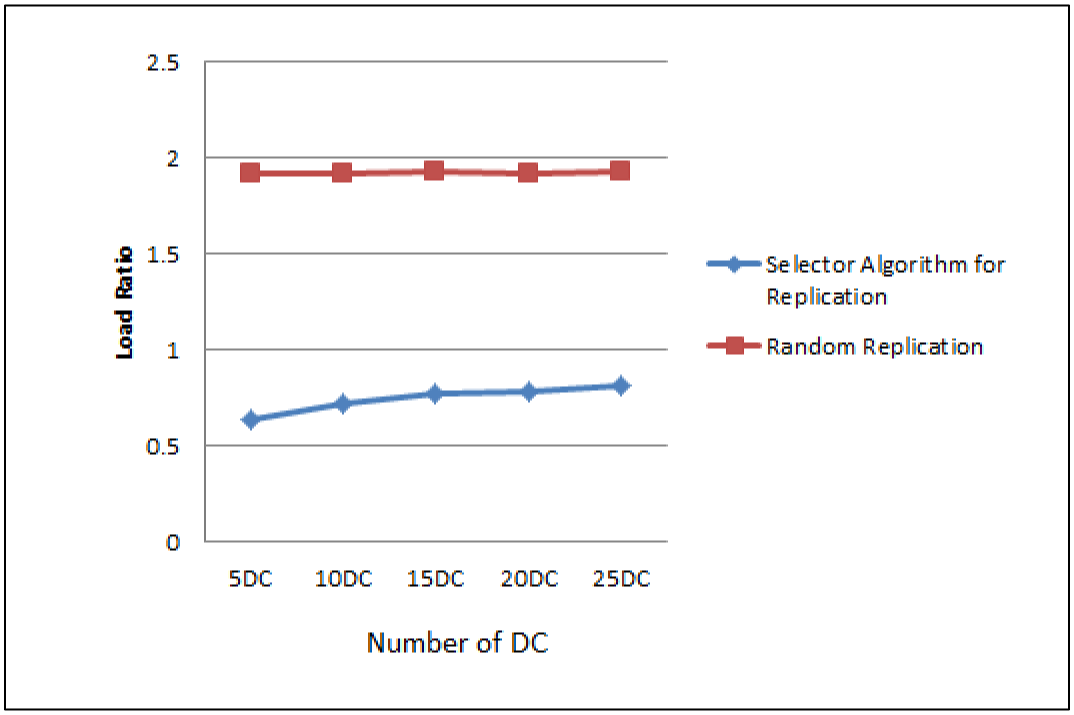
Appendix D Figure 10: 200VMs/DC, 10K to 20K flows per region, Scenario 1



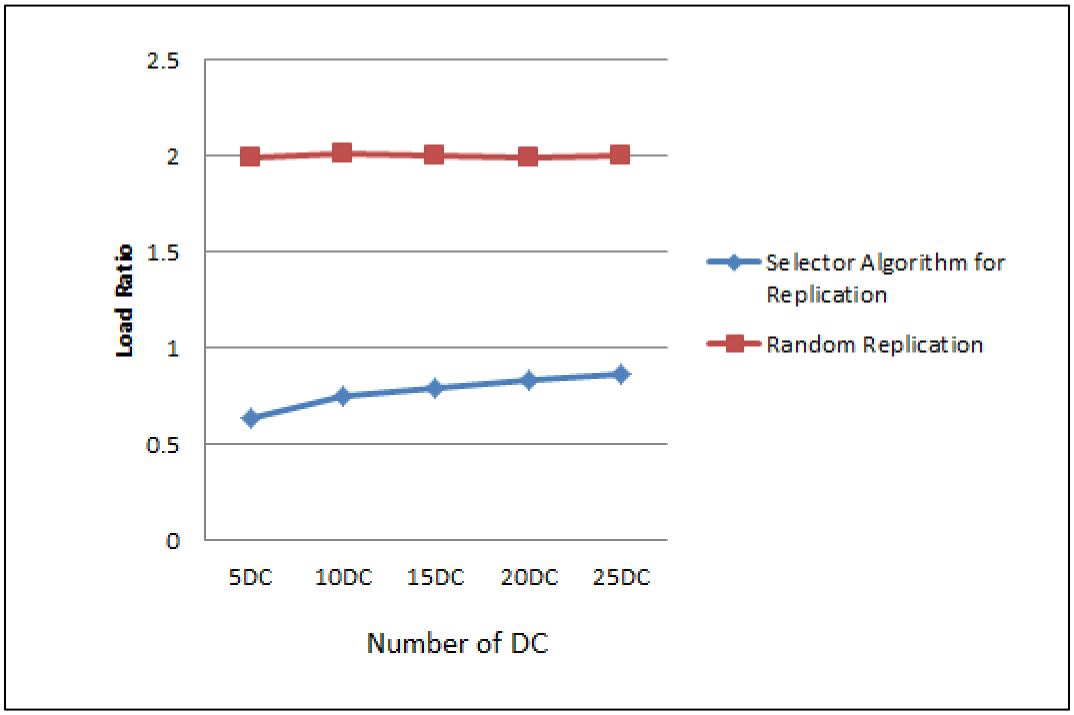
Appendix D Figure 11: 5000VMs/DC, 5K to 10K flows per region, Scenario 1



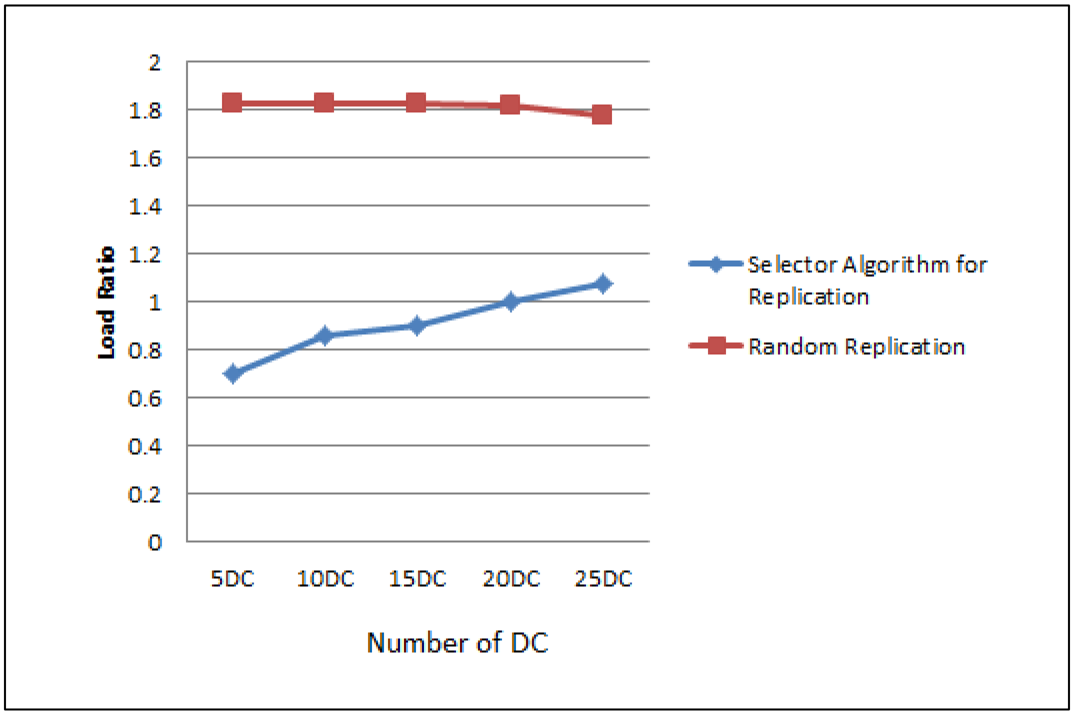
Appendix D Figure 12: 2500VMs/DC, 5K to 10K flows per region, Scenario 1



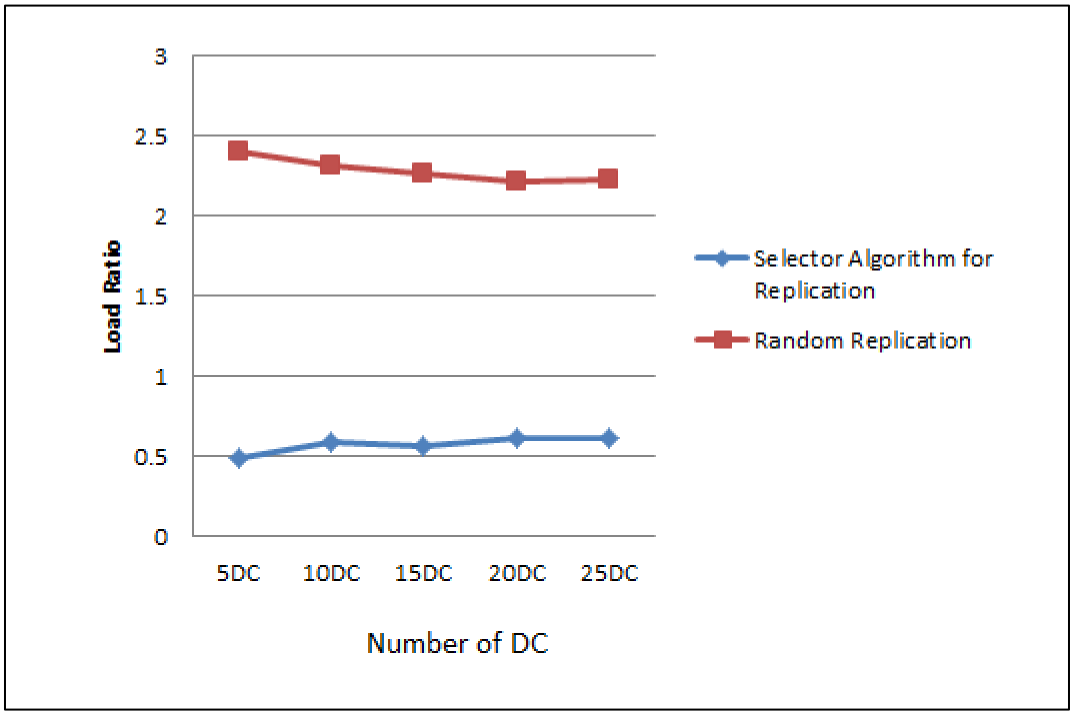
Appendix D Figure 13: 1500VMs/DC, 5K to 10K flows per region, Scenario 1



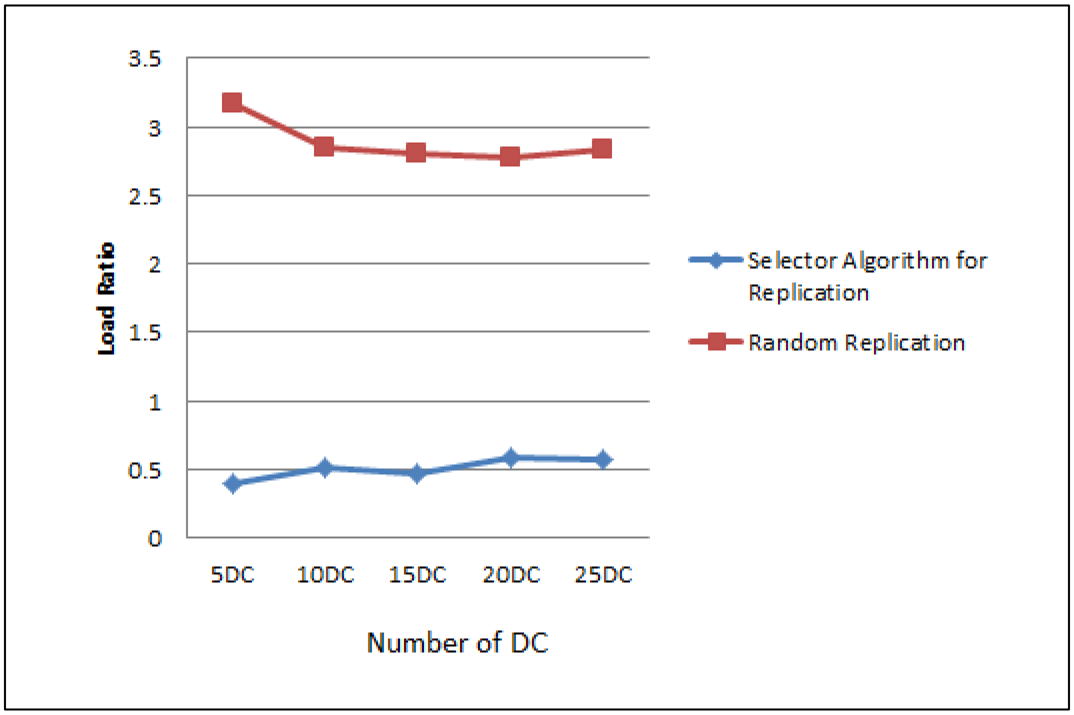
Appendix D Figure 14: 1000VMs/DC, 5K to 10K flows per region, Scenario 1



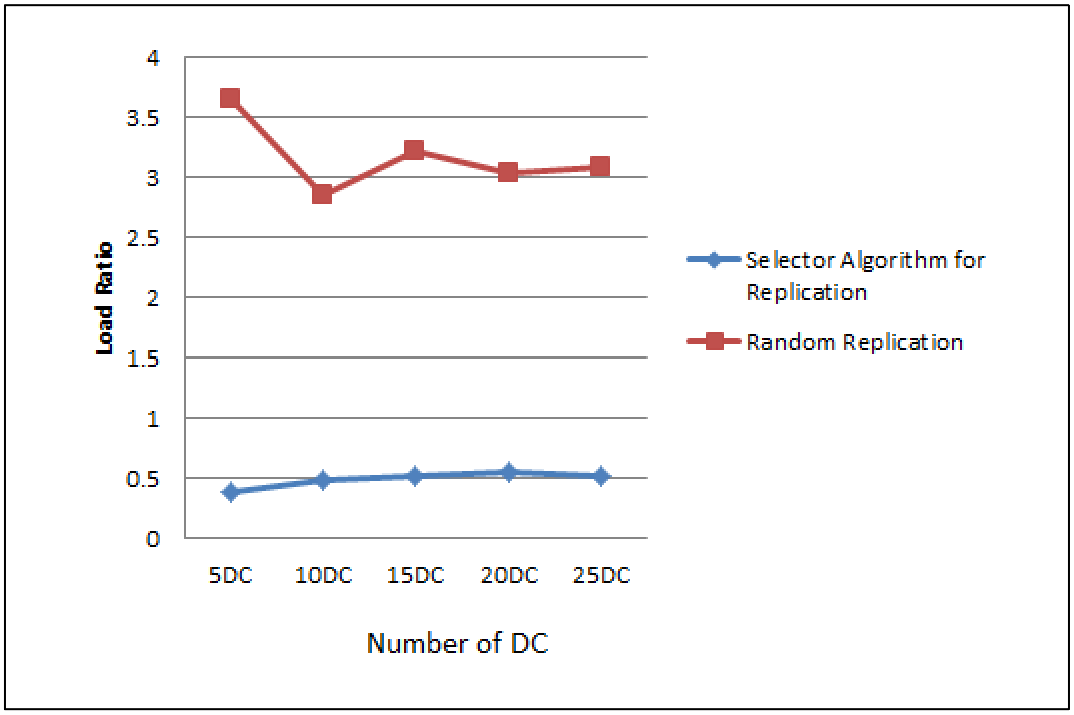
Appendix D Figure 15: 200VMs/DC, 5K to 10K flows per region, Scenario 1



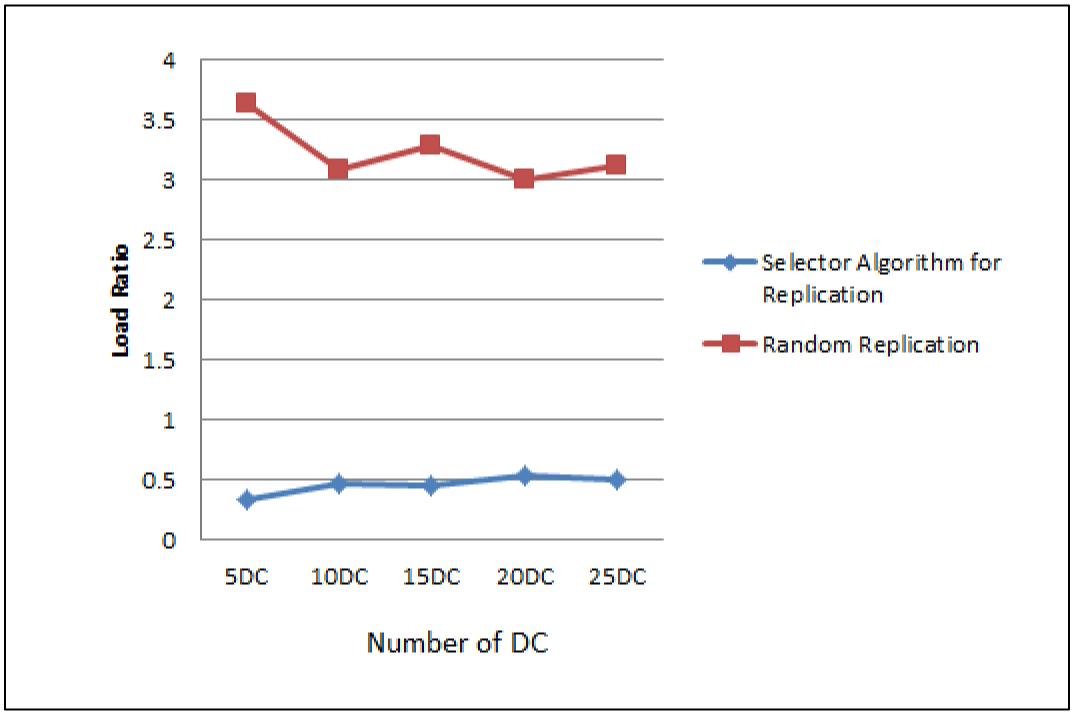
Appendix D Figure 16: 5000VMs/DC, 20K to 30K flows per region, Scenario 4



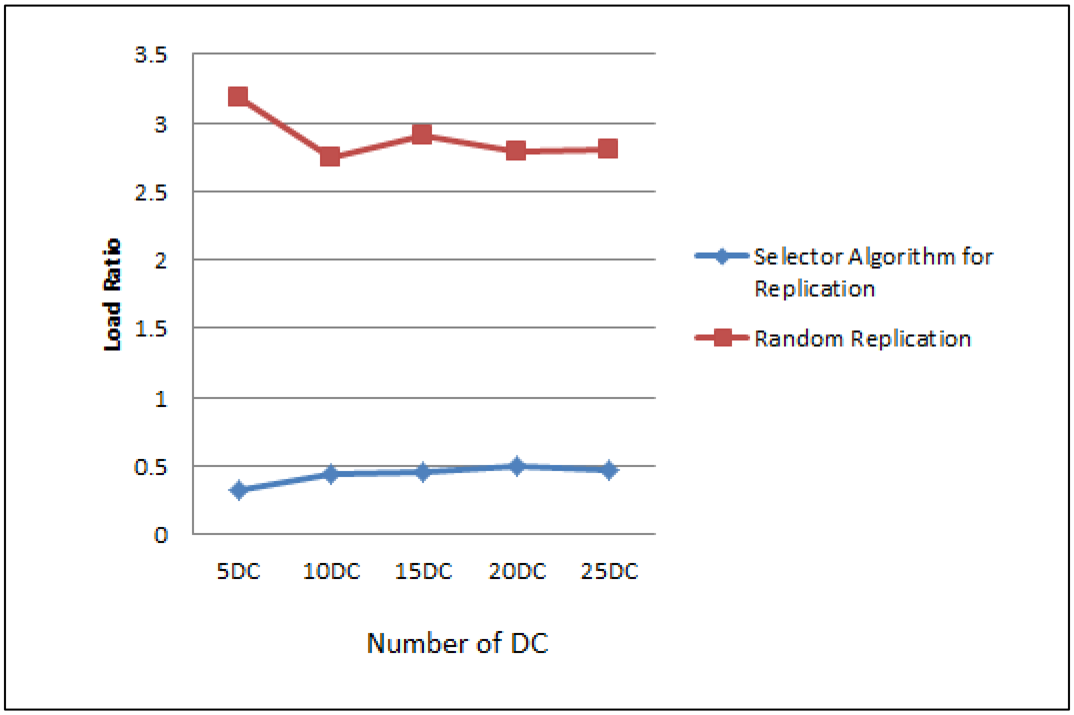
Appendix D Figure 17: 2500VMs/DC, 20K to 30K flows per region, Scenario 4



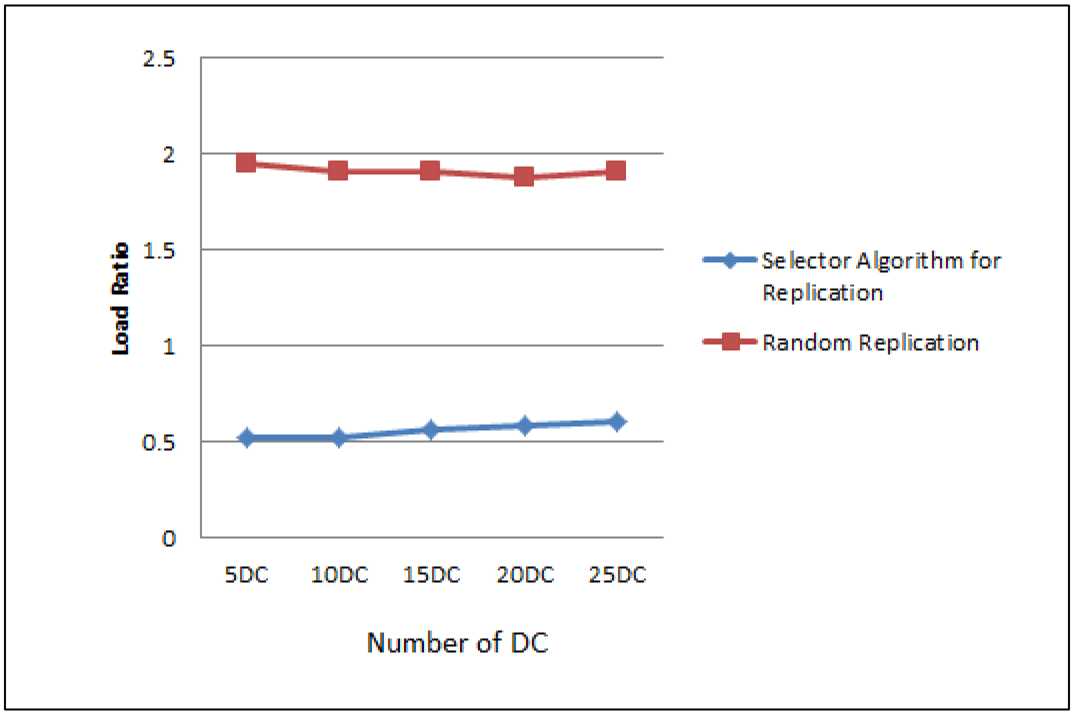
Appendix D Figure 18: 1500VMs/DC, 20K to 30K flows per region, Scenario 4



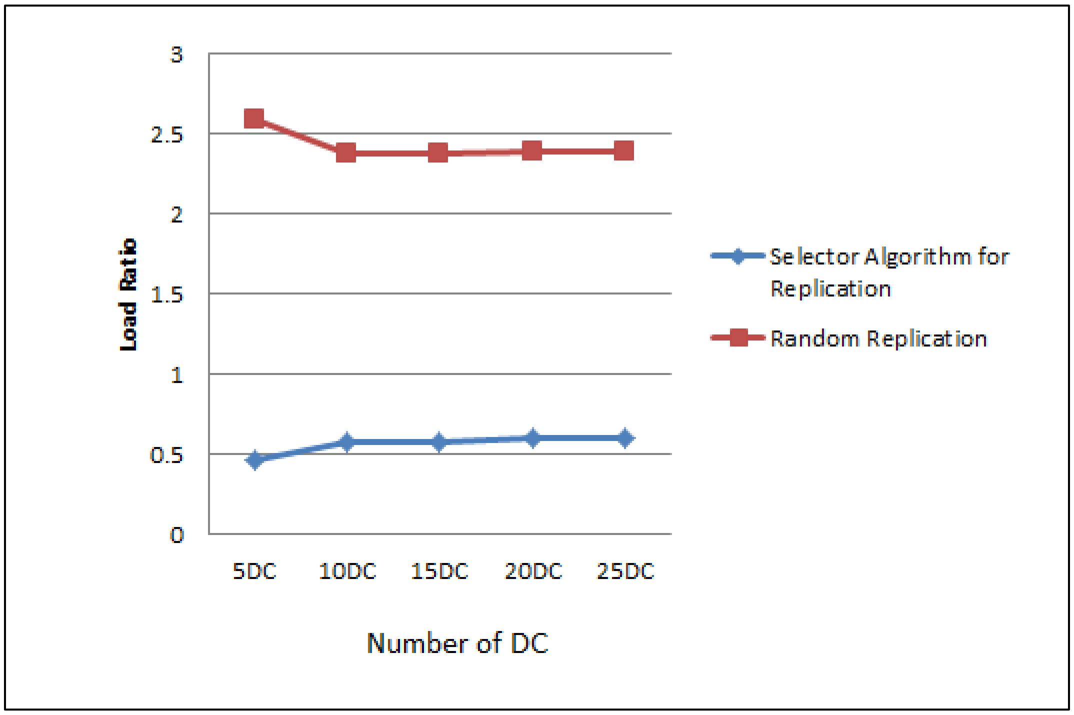
Appendix D Figure 19: 1000VMs/DC, 20K to 30K flows per region, Scenario 4



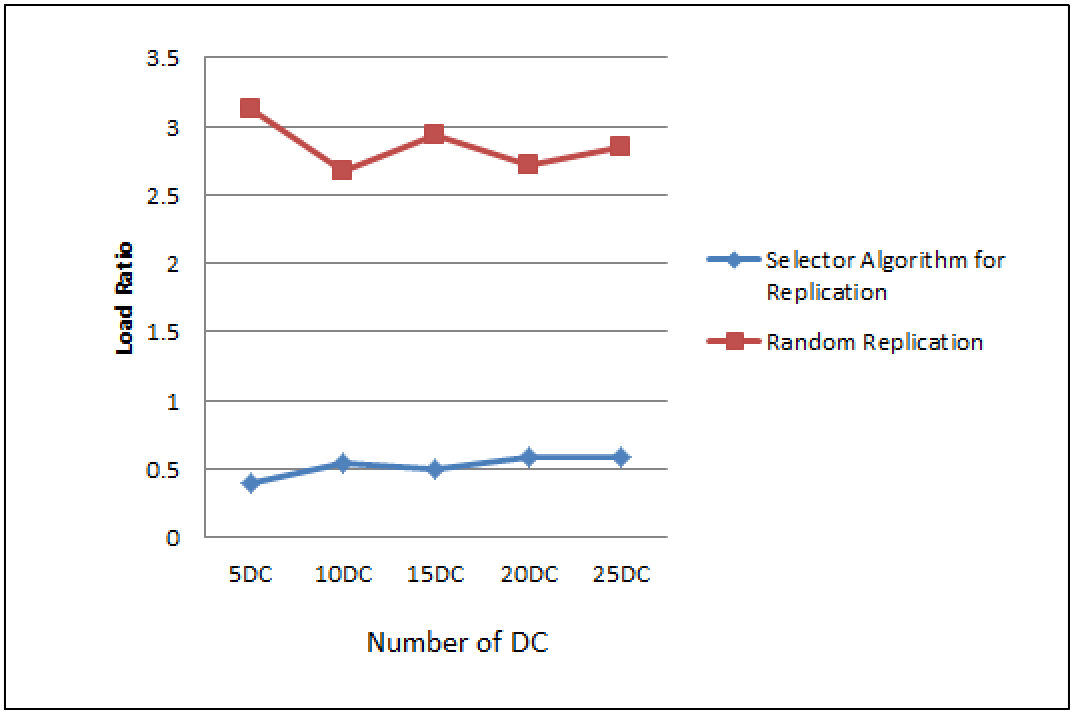
Appendix D Figure 20: 200VMs/DC, 20K to 30K flows per region, Scenario 4



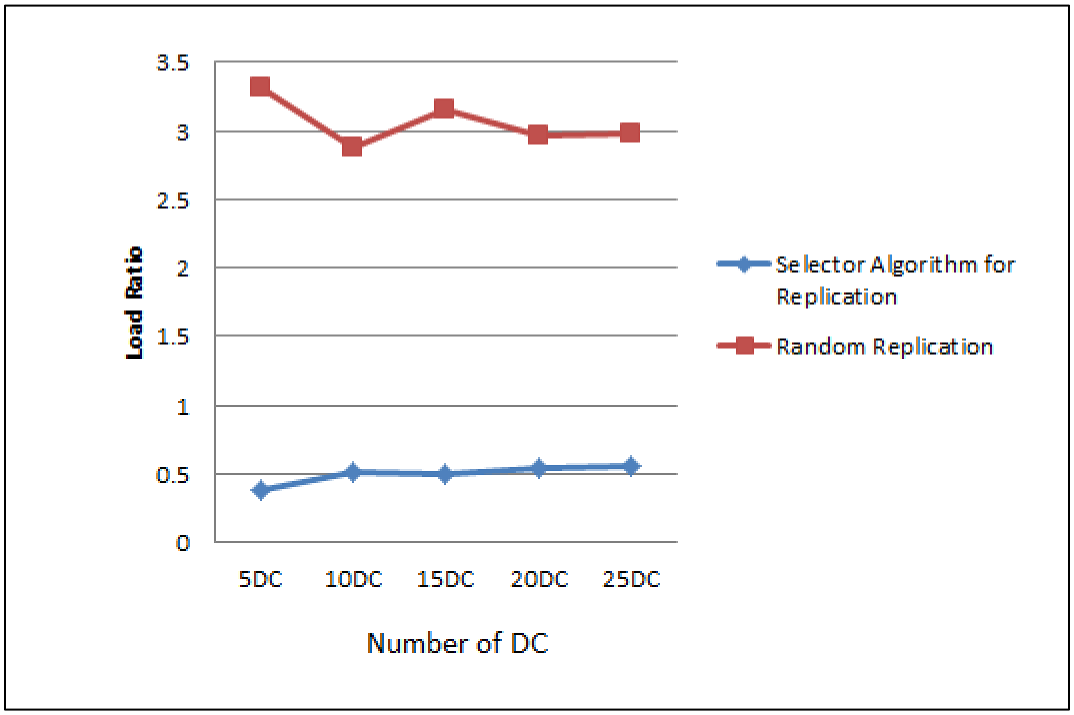
Appendix D Figure 21: 5000VMs/DC, 10K to 20K flows per region, Scenario 4



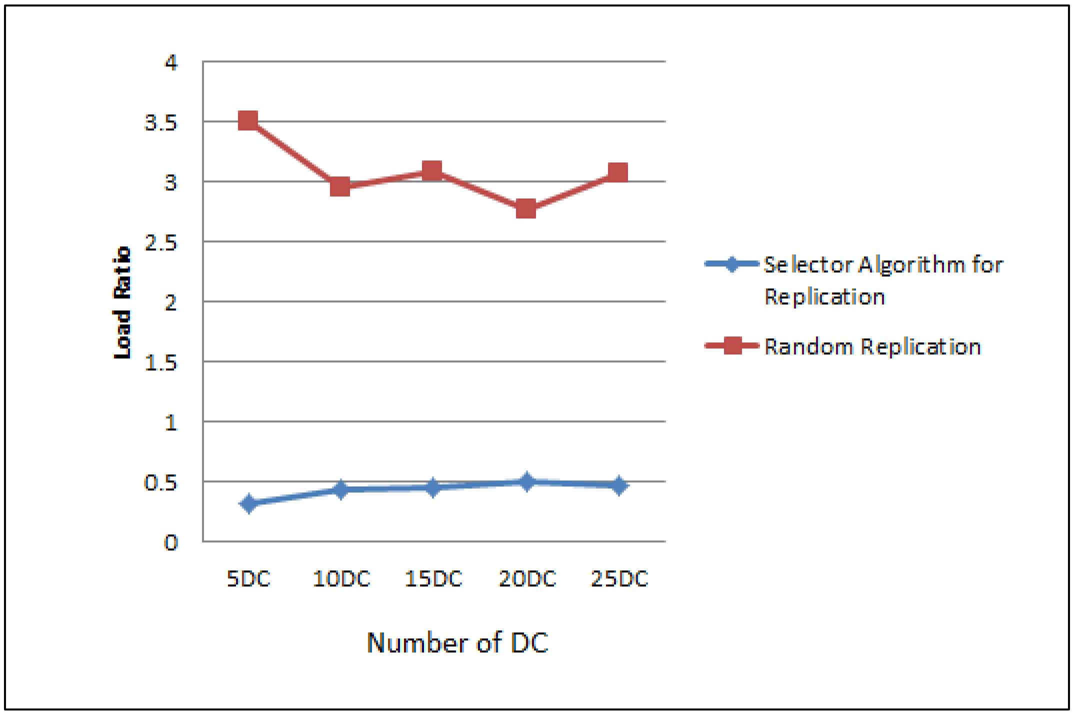
Appendix D Figure 22: 2500VMs/DC, 10K to 20K flows per region, Scenario 4



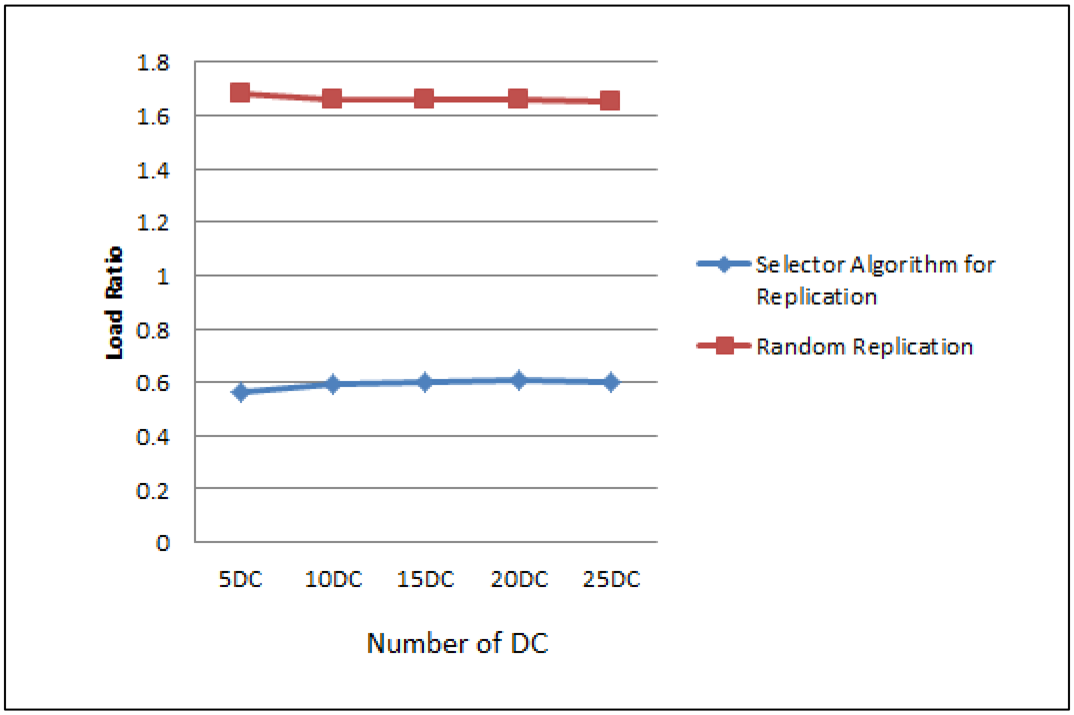
Appendix D Figure 23: 1500VMs/DC, 10K to 20K flows per region, Scenario 4



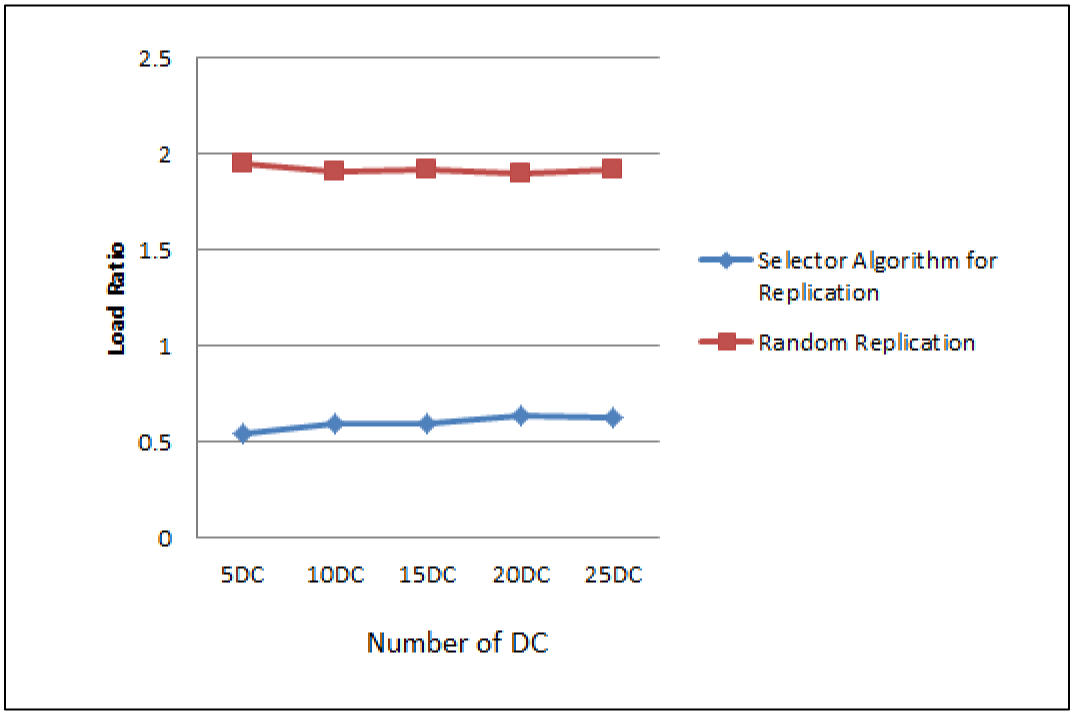
Appendix D Figure 24: 1000VMs/DC, 10K to 20K flows per region, Scenario 4



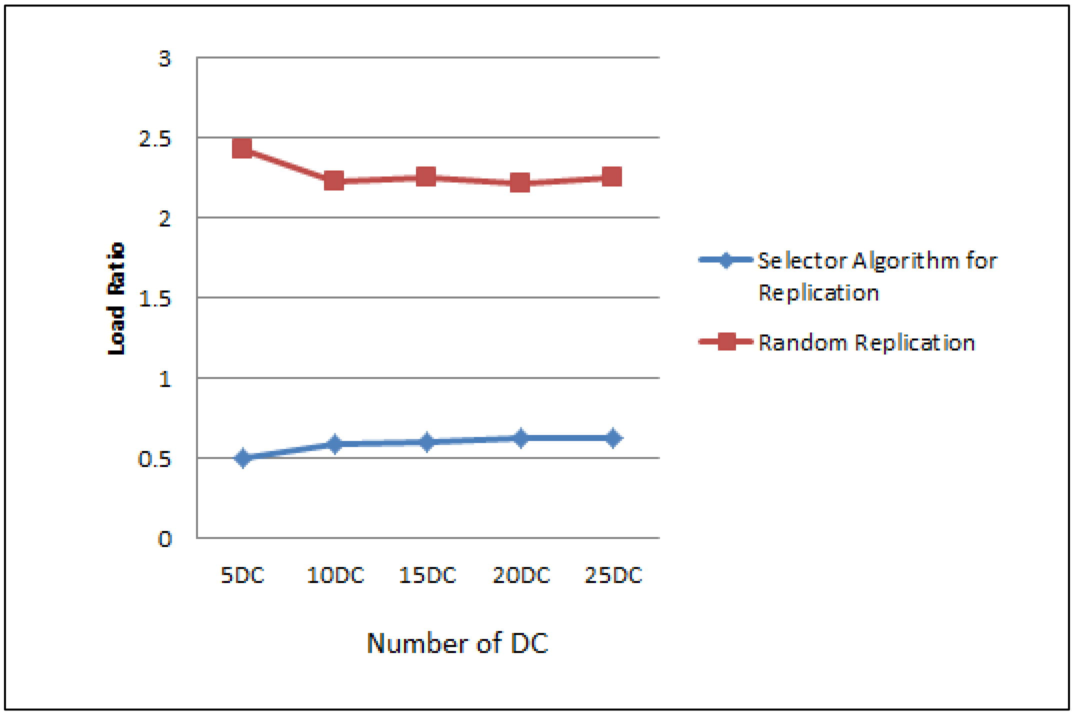
Appendix D Figure 25: 200VMs/DC, 10K to 20K flows per region, Scenario 4



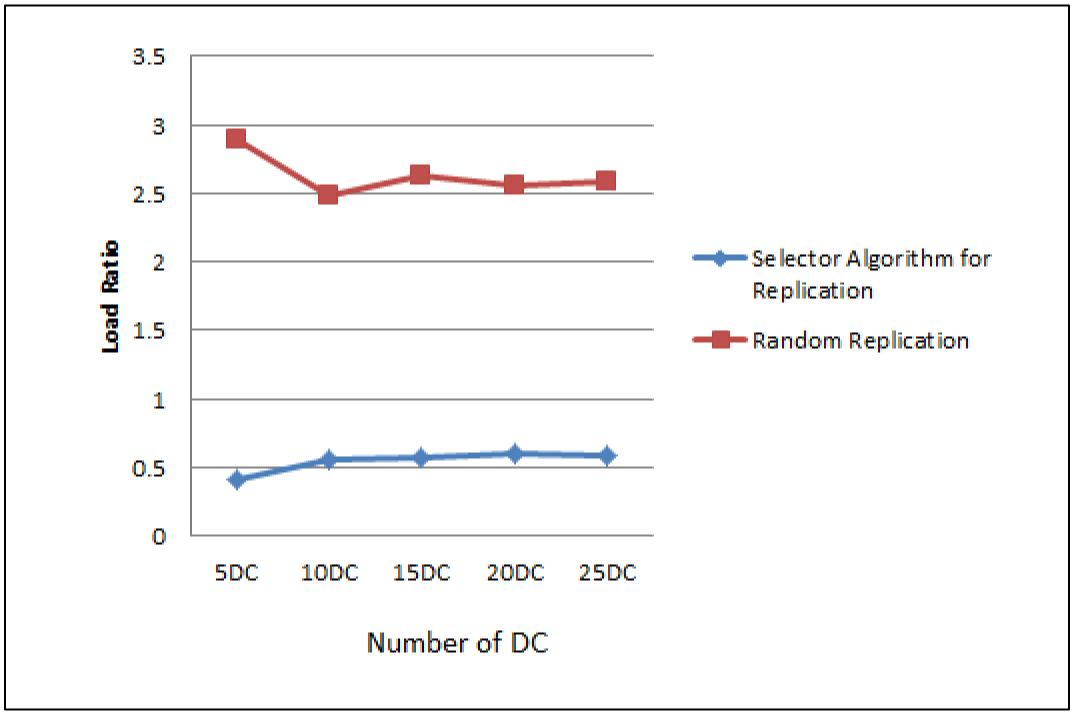
Appendix D Figure 26: 5000VMs/DC, 5K to 10K flows per region, Scenario 4



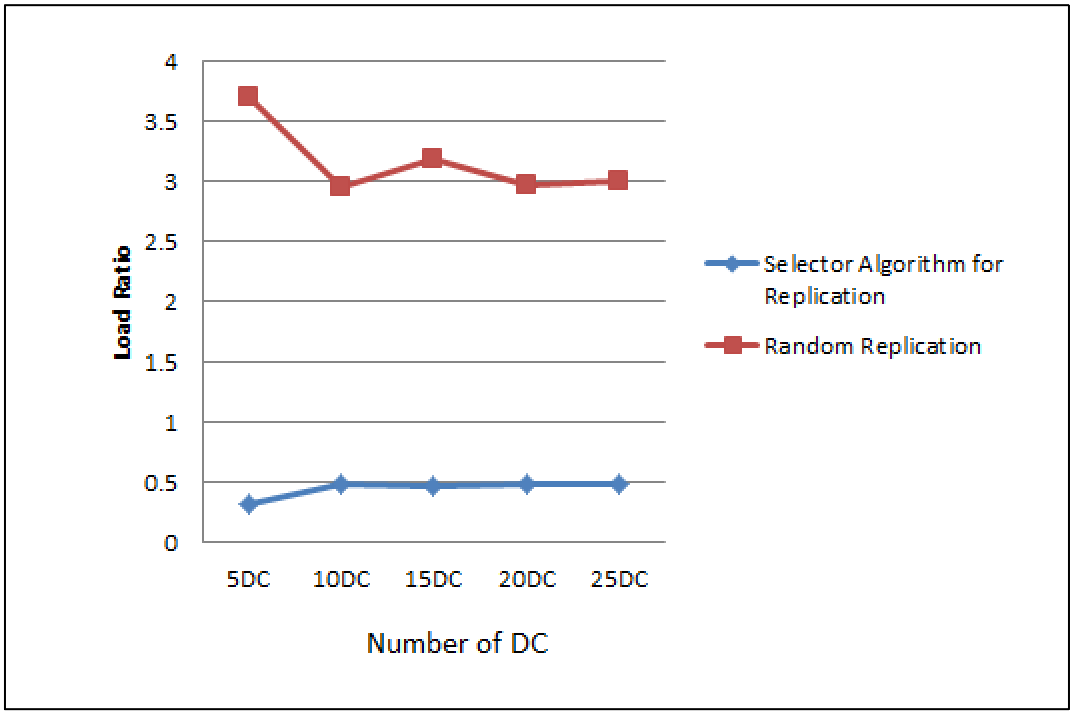
Appendix D Figure 27: 2500VMs/DC, 5K to 10K flows per region, Scenario 4



Appendix D Figure 28: 1500VMs/DC, 5K to 10K flows per region, Scenario 4



Appendix D Figure 29: 1000VMs/DC, 5K to 10K flows per region, Scenario 4



Appendix D Figure 30: 200VMs/DC, 5K to 10K flows per region, Scenario 4

Curriculum Vitae

Name:	Sakif Shahriar Pritom
Post-secondary Education and Degrees:	Bangladesh University of Engineering and Technology (BUET) Dhaka, Bangladesh. 2007-2012 B.Sc.
	The University of Western Ontario London, Ontario, Canada 2013-2014 M.Sc.
Honours and Awards:	Western Graduate Research Scholarship (WGRS) 2013-2014
Related Work Experience	Teaching Assistant The University of Western Ontario 2013-2014