## Contents

## List of Figures

## 1 Abstract

## 2 Introduction

With the advent of cloud computing, datacenters are making use of distributed applications more than ever. Companies like Google use software such as MapReduce to generate over 20 petabytes of data per day using very large numbers of commodity servers [3]. Many other companies use large scale clusters to perform various computational tasks via the the open-source MapReduce implementation, Hadoop [4], or they can possess a virtualized datacenter allowing them to migrate virtual machines between various machines for high-availability reasons. As economics change for hardware, it is likely that a scalable cloud will have the requirement to mix node types, which will lead to higher performance/capacity nodes being mixed with lower performance/capacity HDD nodes. This thesis presents an adaptive data placement method in the Nutanix distributed file system (ADSF) which will attempt to remedy the common problems found in many heterogeneous clustered file systems.

## 2.1 Motivation

A number of scenarios arise in heterogeneous Nutanix clusters that can degrade performance for an entire cluster. The currently replica disk selection logic in Stargate uses does not take into account a number of variables such as disparities in tier size, CPU power, workloads, and disk health among other things.

Considering that a write is not complete until all replicas are written, the write's performance is at the mercy of the slowest disk and node. There are several scenarios, both pathological and daily occurences, where a more robust replica placement heuristic is required. For the work in this thesis, I will focus on two orthogonal cases described below.

### 2.1.1 Interfering Workloads

An example of interfering workloads can take the form of a 3-node homogeneous cluster with only 2 nodes hosting active workloads as shown in Figure 1. In the current random selection scheme in use by the ADSF, writes are equally likely to place their replica on the other node with an active workload as they would be to place it on the idle node. This can impact performance on both the local and remote workloads as secondary writes will be slower on nodes whose resources are being utilized by their primary workloads. An adaptive replica placement scheme is needed to avoid the busy node and bias secondary replica placement on an idle node.

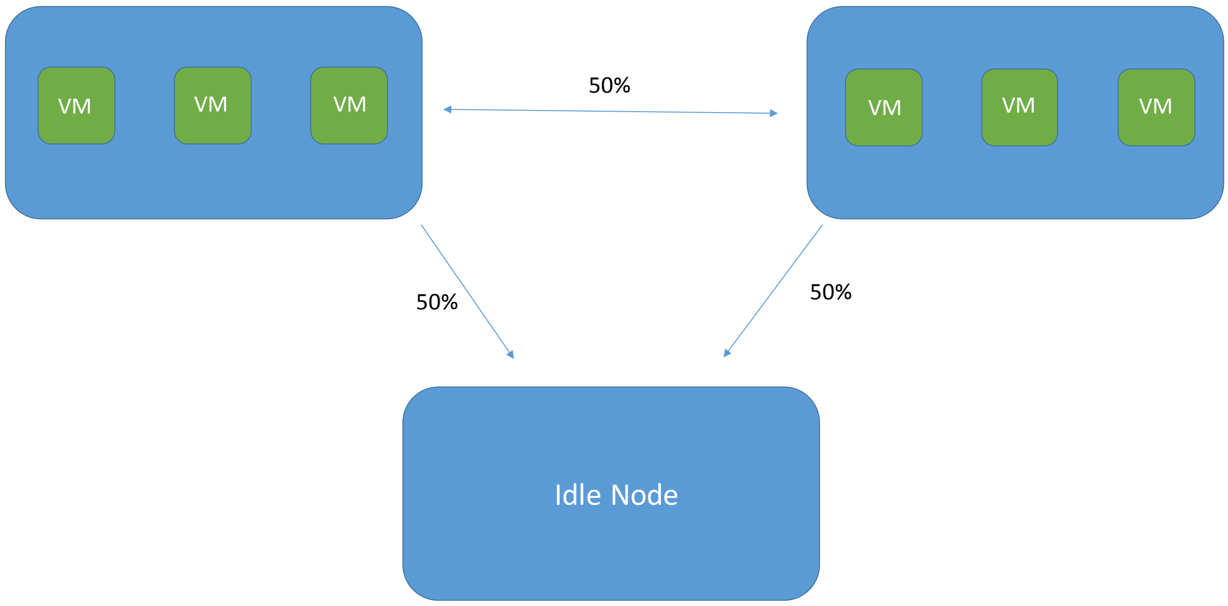

Figure 1: A cluster with identical nodes running a heterogeneous workload.

### 2.1.2 Nodes with Tier Size Disparities

A cluster containing nodes with a tier size disparity are susceptible to a skew in node fullness, even if the workload on each node is identical. This can be illustrated via Figure 2 where we have a 3-node heterogeneous cluster with 2 high-end nodes and a single weak node. Suppose these high-end nodes have 500GB of SSD tier and 6TB of HDD tier and the single weak node has only

128GB of SSD tier and 1TB of HDD tier. If 3 simultaneous workloads were to generate data such that the working sets of the workloads are 50% of the local SSD tier, the weaker node is at a significant disadvantage. Given the current NDFS replica selection algorithm, we can expect 500GB of replica traffic to flood the weak node and fill up its SSD tier well before the workload is finished. This results in an inability for the workload on the smaller node to place its primary replicas locally and forces the workload to rely on remote CVMs, increasing latency. An adaptive replica placement heuristic would mitigate this issue by taking disk usages into consideration during the placement of secondary replicas and biasing placement of secondary replicas on the nodes with more free capacity.

## 2.2 Acropolis Base System

NDFS is facilitated by a clustering of controller virtual machines (CVMs) which reside, one per node, on each server in the cluster. The CVM presents via NFS (for VMWare's ESXi [14]), SMB (for Microsoft's Hyper-V [17]), or iSCSI (for Nutanix's AHV [1]) an interface to each hypervisor that they reside on. For example, the interface provided by the CVMs to VMware's ESXi hypervisor [14] will be interfaced with as a datastore. The virtual machines' virtual disk files will reside on the Nutanix datastore and be accessed via NFS through the CVM sharing a host with the user VM. Within the CVM

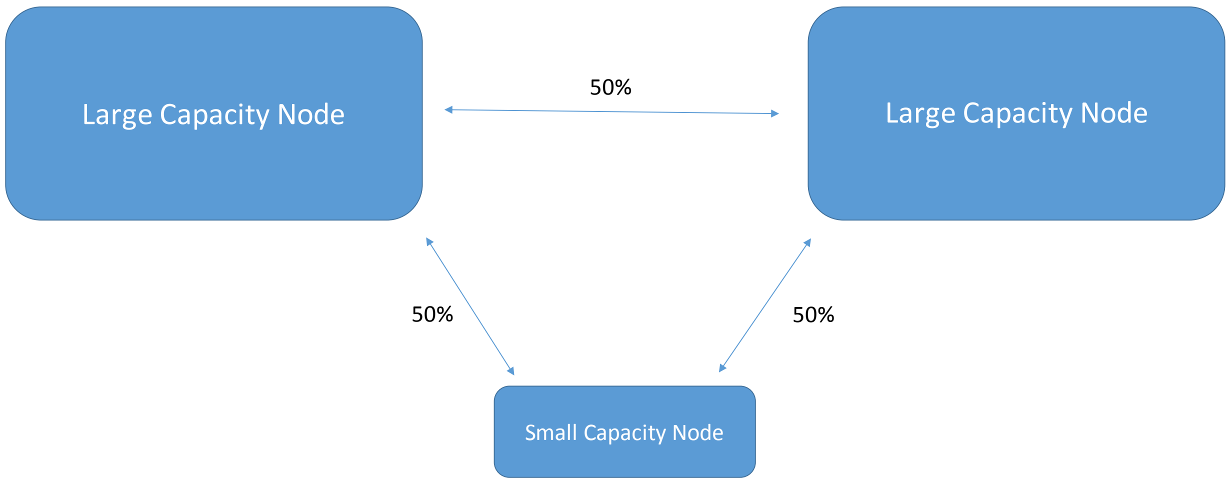

Figure 2: A cluster with nodes of varying resource capacity.

exists an ecosystem of process that make up the ADSF. This work is scoped specifically to the I/O manager process, Stargate.

## 2.3 Stargate

The Stargate process is responsible for all data management and I/O operations. The NFS/SMB/iSCSI interface presented to the hypervisor is also presented by Stargate. All file allocations and data replica placement decisions are made by this process.

As the Stargate process facilitates writes to physical disks, it gathers statistics for each disk such as the number of operations currently in flight on the disk (queue length), how much data in bytes currently resides on the disk, and average time to complete an operation on the disk. These statistics are only gathered on the local disks; however, they are then stored in a distributed database provided by another ADSF service along with the statistics gathered by every other Stargate in the cluster. These disk statistics stored in the database and are pulled periodically and are then used to make decisions on data placement when performing writes.

#### 2.3.1 Oplog and Extent Store

#### 2.3.2 Storage Tiering

#### 2.3.3 Data Replication and Fault Tolerance

#### 2.3.4 Replica Selection

## 3 Prior Work

## 4 Implementation

### 4.1 Overview

### 4.2 Fitness Values and Functions

### 4.3 Weighted Random Selection Algorithms

### 4.4 WeightedVector Class

### 4.5 Changes to the Stargate Disk Stats

### 4.6 Unit Testing

#### 4.6.1 WeightedVector Unit Tests

#### 4.6.2 ReplicaSelector Unit Tests

## 5 Evaluation and Results

The experiments below seek to measure the effect of the additive and multiplicative term fitness functions on both the tier utilization of each node and the queue lengths of disks residing on those nodes compared with uniform random selection.

## 5.1 Experimental Setup

The replica selection schemes were evaluated using a NX-1350 for evaluating the disk fullness and a NX-3-node cluster). Each node contains a single 300GB SSD and 4 HDDs 1TB in size. When evaluating the new replica disk selection framework, two heterogeneous workload scenarios are tested:

1. Two worker VMs on separate nodes running a workload with low outstanding ops.
2. Two worker VMs on separate nodes with running a workload with high outstanding ops.

## 5.2 Fio and Write Patterns

When generating I/O in these experiments, Fio is used on the worker VMs. Fio, short for Flexible IO, is an I/O workload generator that can take configuration files to specify the parameters of a test. On each worker VM, fio is used to generate a sequential write workload that completely fills the cluster's hot-tier. I choose to use exclusively sequential writes for all tests because they are the default for fio tests and the purpose of these experiments is to generate new replicas in a consistent manner. For the purposes of replica placement, the Nutanix file system does not distinguish targets based on the write pattern that generatred an extent group.

## 5.3 Tier Utilization Experiments

The tier utilization experiments define the hot-tier deviation, $d_{hottier}$, as the average SSD utilization percentage of the nodes running a workload, $u_w$, subtracted by the SSD utilization percentage of the node without a workload, $u_o$:

$$d_{hot\ tier} = \frac{u_{w1} + u_{w2}}{u_o} \tag{1}$$

Ideally, the idle node would absorb the majority of secondary replicas from the running workloads. However, uniform random selection causes only 50% of secondary replicas to go to the idle node even though it can potentially handle more work due to the fact that it does not have to service a local workload. In a uniform random replica selection scheme, we expect the nodes running a workload have to bear 100% of their own primary replicas and 50% of secondary replicas from the other worker node. This causes total SSD utilization to be skewed towards the worker nodes and for this skew to grow as the tests run. This is indicated by higher $d_{hot\ tier}$ values. We expect a more sophisticated replica selection scheme to minimize $d_{hot\ tier}$ by biasing secondary replicas towards the idle node and limiting the skew.

### 5.3.1 Low Outstanding Operation Results

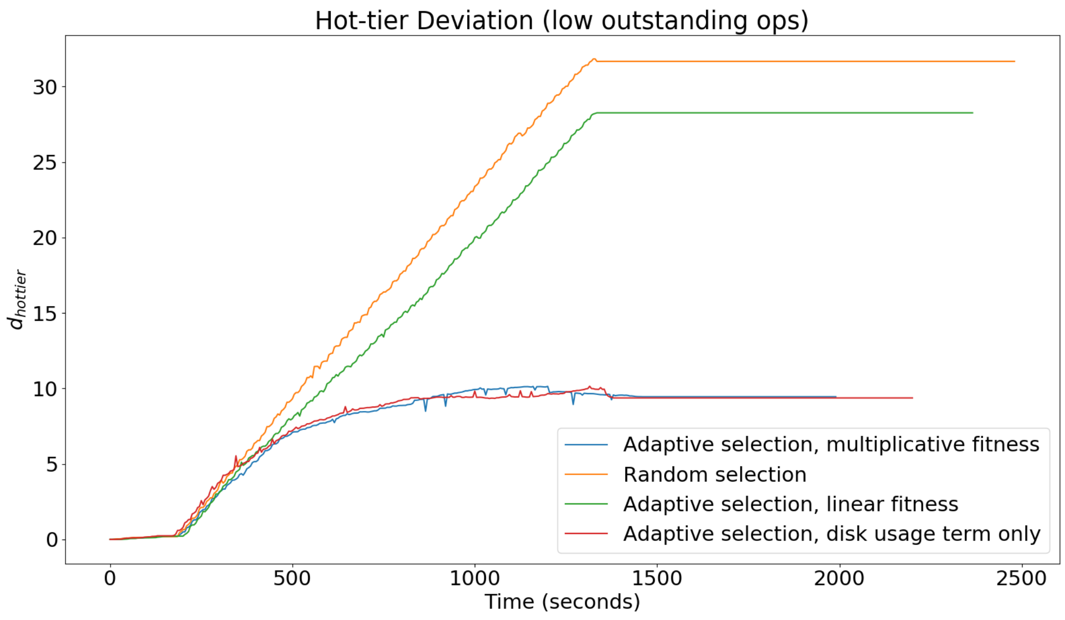

Figure 3: $d_{hot\ tier}$ values over time for low outstanding I/O operations.

Figure 3 shows the results of a workload with only a single outstanding operation. This causes the queue length reported by Stargate to be at most 1, resulting in the fitness function's queue length term to be roughly constant and approximately 1.

We can see that the additive fitness function does not minimize the hot-tier deviation as well as the multiplicative. This is because an additive fitness function's behavior varies depending on the weight chosen for the queue length term. By default, the linear fitness function gives equal weight to both the disk fullness and queue length terms; however, for one run of this experiment the queue length term was given no weight. Linear fitness that gives equal weight to both terms does not reduce skew by very much while giving no weight to the queue length term keeps all nodes' usages within 10% of each other. This is because the queue length term is contributing the maximum amount possible to the fitness value due to the consistently low queue length values. We can illustrate this by defining a disk selection bias, $b_r$, as the probability some disk, $d$, will be selected when compared with another disk, $d'$ whose utilization is 10% higher and queue length value is identical. Given a fitness function, $f$, we can calculate $b_r$ as follows:

$$b_r = \frac{f(d)}{f(d) + f(d')} \tag{2}$$

Figures 4 and 5 show the disk selection biases for very large and very small queue lengths respectively.

We can see that for an additive fitness function, the bias towards a less utilized disk decreases as the disk fullness percentages for $d$ and $d'$ increase,

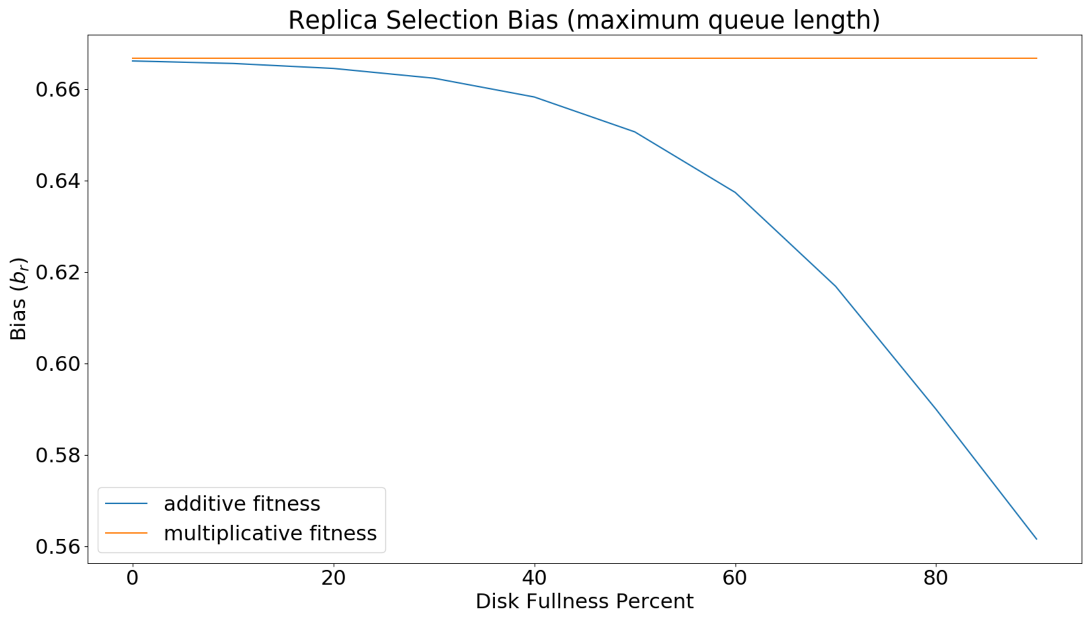

Figure 4: $b_r$ values with static queue lengths at the fitness function ceiling values.

even though they still only differ by 10% in the figure above. This is because the entire linear fitness function does not scale with each term, so we can conclude that the multiplicative fitness function is superior.

### 5.3.2 High Outstanding Operation Results

In Figure 8, we can see that both additive and multiplicative fitness functions reduce the disk fullness skew from 30% to less than 10%. Multiplicative fitness performs slightly better at minimizing $d_{hot\ tier}$ than additive fitness,

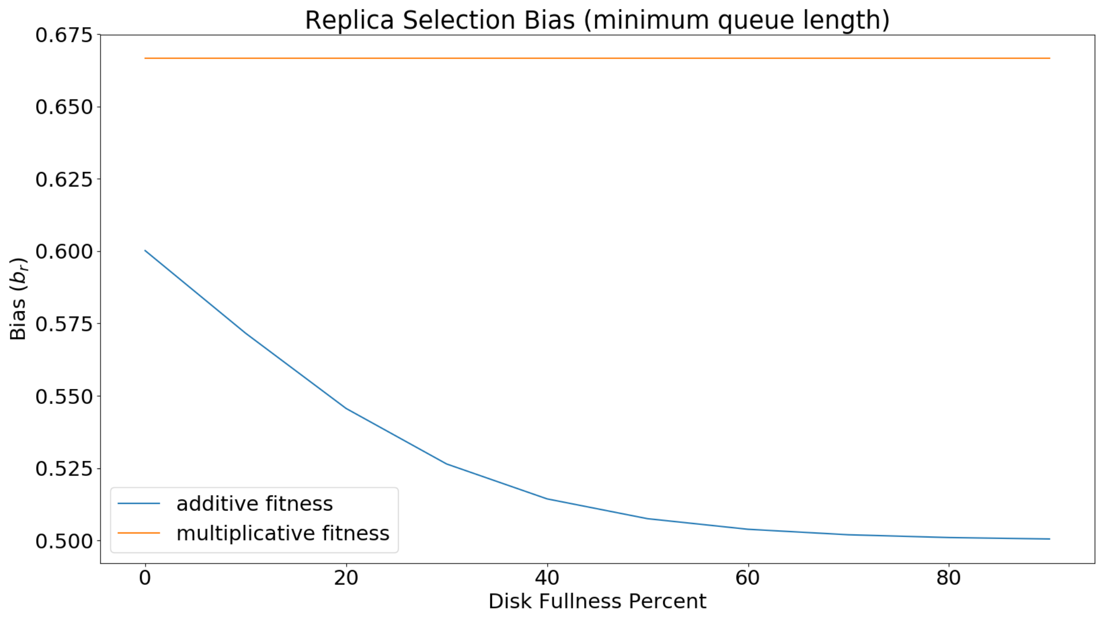

Figure 5: $b_r$ values with static queue lengths at 1.

possibly due to scaling the fitness value by the value of both the fullness and queue length terms, rather than weights.

## 5.4 Disk Queue Length Experiments

Since a low outstanding operation workload would not give useful information for measuring the effects of fitness-based replica selection on disk queue lengths, the high outstanding I/O operation experiment in the previous section was re-run for all fitness function types and for fitness function queue

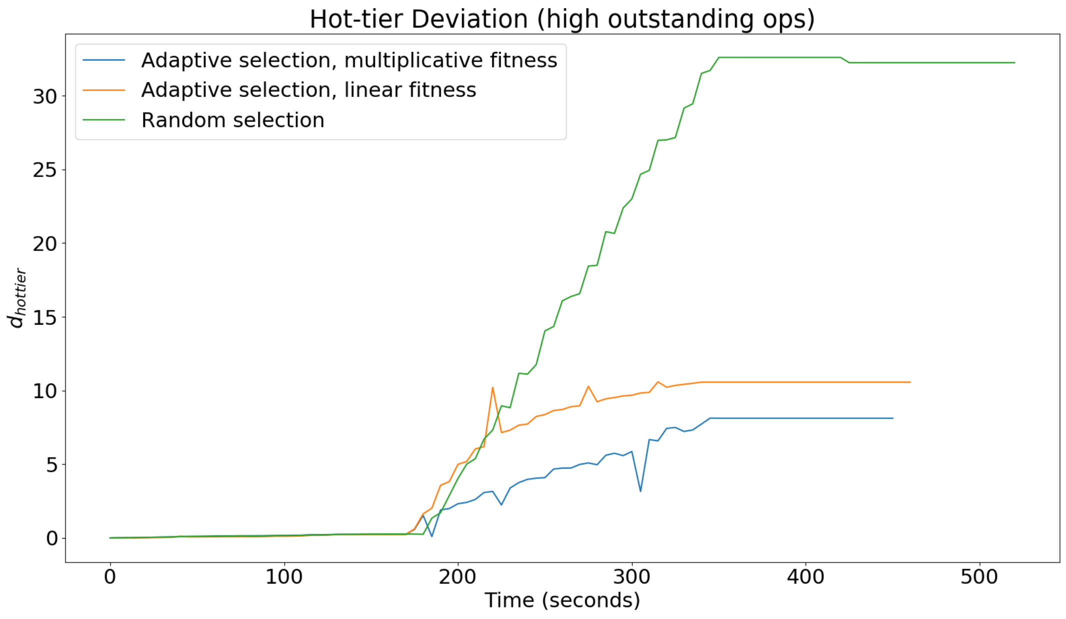

Figure 6: $b_r$ values with static queue lengths at 1.

length term ceilings of 200 and 100. Figures 7 and **??** show a reduction in queue length quartiles when fitness-based selection is used for disks on nodes that host local workloads. Lower queue length ceilings are observed to provide better results in reducing the queue lengths for the worker nodes.

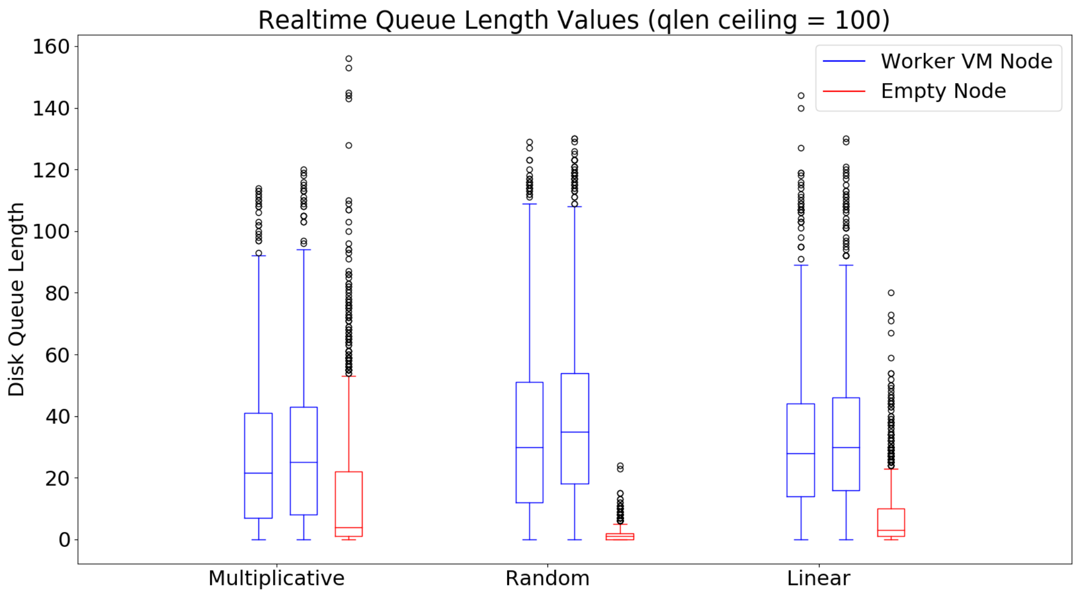

Figure 7: Queue lengths for all SSDs on the specified nodes sampled every 1 second.

# 6 User Guide

## 6.1 Re-creating Experiments

## 6.2 Scraping Data From the Nutanix Cluster

# 7 Conclusions

## 7.1 Summary

### 7.1.1 Low Outstanding Operation Test

### 7.1.2 High Outstanding Operation Test

## 7.2 Conclusion

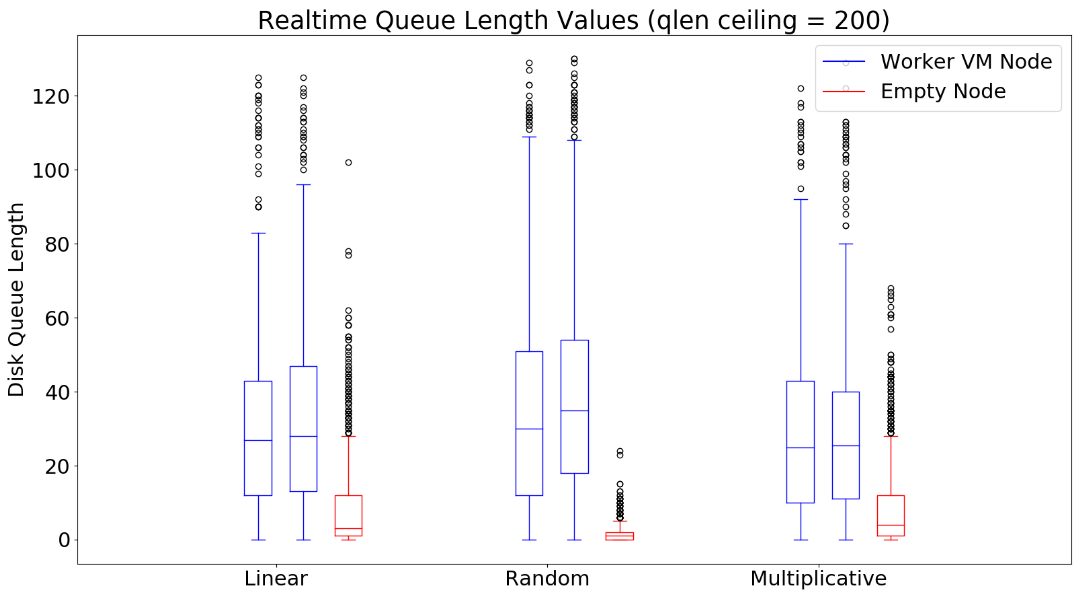

Figure 8: Queue lengths for all SSDs on the specified nodes sampled every 1 second.