The disk input/output performance with virtual private servers is known to be suboptimal at best. This is largely due to the fact that virtualisation leads to the usage of shared resources and in particular shared disk access. In this research we look at the possibility of improving disk reading performance by using the read-balancing option of DRBD v8.4.1. We conducted tests in three environments, a local controlled environment, a remote WAN environment and a remote LAN environment. While DRBD was found to be successful in improving the disk reading speed in certain cases in the local controlled environment, we could not achieve such results in the actual hosting environments.

I. INTRODUCTION

Virtualisation of computing resources has made possible the easy utilization of computational power by using so called Virtual Private Servers. A Virtual Private Server (VPS) is a virtual machine (VM) that is provided as a service by a hosting provider. As such, VPS instances use shared resources, which means that they make use of time-sharing techniques to access the underlying physical resources. This also applies to disk drive usage.

As multiple virtual machines attempt to access a single physical disk in a time-shared manner, each of the VMs only gets a fraction of the total input-output capabilities of the physical disk. This leads to slower disk reads and writes which in turn results in degraded performance when disk I/O is required.

These problems are not unique to virtualised environments, as such issues may also occur when multiple applications perform I/O-heavy disk operations simultaneously on a single physical machine. However, the problem is exacerbated in a virtualised environment due to not only different applications accessing a single disk, but also multiple virtual machines.

DRBD, which stands for Distributed Replicated Block Device, is a storage distribution system [1]. DRBD employs principles similar to those backing RAID 1, but unlike RAID 1, DRBD operates in a networked environment. DRBD mirrors data on the block-level, writing to several devices over a network connection and is often used to create offsite backups. What makes DRBD interesting in the context of disk I/O optimization is its capability to perform read balancing by using several techniques (round-robin, read striping, etc). This capability is relatively new for DRBD because it is made available since DRBD v8.4.1 which was release in February 2012[2].

A. Research Question

Based on the considerations outlined above, we formulated the following research question:

Can DRBD be used to improve disk input/output performance on Virtual Private Servers?

Related to the formulated research question, we outlined the following subquestions:

- What is the extent of disk I/O performance degradation on virtual machines when comparing it to physical disk access?
- What is the effect of using DRBD with virtual machines is a LAN environment?
- What is the effect of using DRBD with virtual machines is a WAN environment, where latency is higher?
- Does DRBD provide a significant increase in performance?

We believe that we can answer the research question by researching and answering each of the identified subquestions.

B. Related Work

In 2002, Reigner et al.[3] laid the foundation of the theoretical work behind DRBD, describing a very early version of the software – its architecture, communication protocols and other concepts. With regards to performance, the paper focuses on showing that there is no performance loss because of using DRBD for high availability. However, since the software was in its infancy at the time of writing the paper, its findings are mostly relevant from a historical point of view.
In 2007, Lars Ellenberg et al.\[4\] gives a more mature view of the software. However, the paper does not focus on performance gains possible.

In 2009, Groot et al.\[5\] discuss possibilities for virtual machine migration using DRBD. Several other papers discuss networked RAID-like solutions for achieving either high availability or increased performance\[6\]\[7\]. None of these focus on improving the performance of virtual machines.

II. THEORY

In this section we provide the required theoretical background for understanding this report. We give theoretical definitions to the most significant terms and concepts used within the report.

A. Server virtualisation

**Virtualization** is the concept of simulating a physical computer system by using software. This simulated environment is then exposed to an Operating System, which can run within.

By using software to represent hardware, one achieves a high level of control and granularity – while simulating a physical computer system, one may allow the simulated system to use only a fraction of the resources of the actual physical system. This way, a very powerful server can allow the running of multiple Operating Systems on the same hardware simultaneously. The combination of the simulated physical hardware with the Operating System running on that hardware is referred to as Virtual Machine (VM).

Related to virtualisation, a **hypervisor** is a software tool, which provides the virtualised execution environment for the virtual machine to run in. The hypervisor is responsible for controlling and maintaining the execution environment and to limit the amount of resources a virtual machine may use. Hosting providers setup servers with a virtualisation hypervisor installed which controls the access to the physical hardware between the virtual machines(VMs) installed. Most common hypervisors support the allocation of dynamic memory to each VM and use sparse storage for the hard drive images. This saves up space on the host as only the amount of data that is used inside the VM is allocated to for the image instead of the whole size of the disk.

B. RAID

RAID is a storage technology to combine multiple disks into one single logical storage, hence then name Redundant Array of Independent Disks (RAID). The hard drives are connected to a RAID controller which will create an abstraction of the physical disks and present one logic array to the Operating System. The RAID controller works on the block level and, depending on the configuration used, reads and writes from multiple disks for performance or redundancy. There are multiple configurations one can use in creating a RAID. The most common used configurations are striping data across disks (RAID 0), mirroring (RAID 1) or striping with distributed parity data (RAID 5 or 6). Also a combination of multiple can be used like striping over mirrored disks (RAID 10).

C. DRBD

DRBD, which stands for Distributed Replicated Block Device, is a storage distribution system which performs RAID 1 like technology over a network. Which means it can do block level replication of data to a remote disk. DRBD is most used on high availability clusters to remove the single point of failure in case a SAN or NAS fails. Furthermore it has the advantage that DRBD can carry out reads locally instead of reading over the network from a SAN or NAS. Since the latest version of DRBD, v8.4.1, it supports read-balancing to configure where read preference on which disk it should use for read requests. This option could be used to increase read performance in case the local disks fails or is busy.

D. Performance testing

To measure the performance of a hard drive a program can be used to measure the time needed for a single read or write requests. Since the testing program runs within user space and the Operating System also performs requests to the disk an average of the tests should be used. Also a disk is optimized in such a way that it will write data that belongs to a certain file near each other which increases read and writing speeds. So there is a big difference in performance of reading a block of data and reading the same amount of data which is scattered over the disk.

A tool which demands to our requests is the open-source tool Bonnie++\[8\]. This tool can perform single read and write requests as well as blocks of data, Bonnie++ also measures the time that is needed for the disk to seek for a certain data block (random seek time). The results Bonnie++ produces are an average of multiple performed
tests to give accurate results, it also chooses file sizes of double the host RAM to eliminate the caching of files in memory.

III. EXPERIMENTAL METHODS

In this section we explain how our different testing environments are created which we used for testing the DRBD performance on virtual private servers.

A. Restrictions & Assumptions

In our tests we make use of two different environments we started with the baseline tests in our local environment where we controlled both the host server and the created virtual servers. We repeat the tests later at virtual private servers located at different hosting providers, we consider the remote environment out of our control since we don’t know anything about the host server.

We made the assumption that our local environment is considered stable and will run the tests once. But as the state of the remote servers is unknown we will perform multiple tests in that setup.

B. Preparing DRBD

DRBD is integrated into the mainline Linux kernel. However, the read-balancing feature we wanted to use, was introduced in version 8.4.1 of DRBD which was released in February 2012. That version of DRBD had not been merged into the version of the kernel that was being used with our distribution of choice (Ubuntu 11.04-server x64). We decided that it would be best to compile the DRBD software from source and to package it into a .deb package, so that we can reuse the compiled version on our other VMs.

The compilation of DRBD is a straightforward process, which is described in detail in the DRBD online documentation[2]. We followed the recommended process for compilation. Also the packaging of DRBD into a .deb package is facilitated by existing DRBD packaging tools which is also documented in the online documentation. After compiling both the DRBD modules and utilities and packaging it in a .deb package we distributed this on our VMs and installed DRBD successfully. More details about compiling and packaging DRBD can be found in the Appendix B.

C. Local Setup

The first stage of testing was performed within a local setup. In this local setup the project team was the one that administered the virtual private servers. To simulate a real VPS environment, we also generated load on the file system of the hosting server by running multiple virtual machines, which performed disk I/O.

1. Creating the Local Environment

For the local environment we used two physical servers with hostnames statler and peach. On these servers we installed XEN 4.1 as hypervisor and created multiple virtual servers(VMs) on the hosts. On these VMs we performed our tests. To simulate the environment of a real hosting provider where multiple VMs are running on the same host, we created additional VMs and generated disk I/O. Since we are aiming at improving disk I/O on cheap virtual private servers we created VMs which where allocated the following resources:

- 25 GiB hard-disk space
- 256 MiB RAM
- One CPU core

The created virtual machines are listed in Table I. All virtual machines where installed with Ubuntu 11.04 server x64 as operating system and the disk image was stored at a LVM volume group located on the local disk.

<table>
<thead>
<tr>
<th>Server Hostname</th>
<th>VPS Hostname</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>statler</td>
<td>vps1-statler</td>
<td>145.100.107.98</td>
</tr>
<tr>
<td>statler</td>
<td>vps4-statler</td>
<td>145.100.107.101</td>
</tr>
<tr>
<td>statler</td>
<td>vps5-statler</td>
<td>145.100.107.102</td>
</tr>
<tr>
<td>peach</td>
<td>vps1-peach</td>
<td>145.100.107.130</td>
</tr>
<tr>
<td>peach</td>
<td>vps2-peach</td>
<td>145.100.107.131</td>
</tr>
<tr>
<td>peach</td>
<td>vps3-peach</td>
<td>145.100.107.132</td>
</tr>
</tbody>
</table>

TABLE I: Local Virtual Machines

2. Generating Disk Load

In a “real” VPS environment the disk I/O capabilities of virtual private servers are limited because of the fact that multiple virtual machines are attempting access to the shared disk resource. To properly simulate such an environment we needed to generate disk I/O activity on the host machine.
We decided to do this by instantiating three virtual machines on each physical server. Of those, one was used for performing tests, while the others were used to generate load on the host machine. The generation of load was done by running a benchmarking tool for random periods of time on each of the two load-generating virtual machines. That way actual usage of a moderately loaded server is simulated – with periods of high disk I/O activity and periods of no disk I/O activity.

We used the tool blogbench for this simulation. Blogbench[9] tries to reproduce the load of a real-world busy file server by performing random reads, writes and rewrites in multiple threads to the disk. We executed this tool with randomized parameters by running a simple wrapper script which is included in the Appendix C.

3. DRBD configuration

On the testing VMs on statler (vps1-statler) and peach (vps1-peach) DRBD was installed and a DRBD volume was created across those two VMs. Since DRBD operates on the block level and thus requires a block device to create the volume on, we had to mount an additional block device. This due to the fact that DRBD uses the whole disk and thus the disk which contains the operating system could not be used. Because you can’t mount an additional hard drive on a hosted VPS we mounted a 5 GiB file on a local loopback device and used that for our DRBD volume.

Now on both VMs we configured DRBD in a single-primary mode where the we used /dev/loopback0 as block device for the volume. We then configured vps1-statler to be the primary and verified that the volume was synced to vps1-peach.

D. Remote Setup

For our remote setup we used three virtual private machines at two different hosting providers. Two where located at PCextreme[10] and one at Leaseweb[11], both servers where considered cheap (under 10 EUR per month) and similar hardware. See Table II for the technical details of the servers. All servers we’re running Ubuntu 10.04 server x64 as operating system and we used the compiled DRBD modules and utilities to install DRBD. We then created a loopback block device as described in the local setup section and add added this to the DRBD configuration.

With DRBD installed on all VPS, we then created two different DRBD volumes. One between the two PCextreme VPS servers and one between a PCextreme and Leaseweb VPS. We ensured that the two VPS at PCextreme where located at different hosts and where using a different storage device. By using two VPS at the same hosting provider we assume that the network latency between the machines is lower compared to a setup which involves a WAN link.

By using those two setups we are able to conclude if read-balancing over a WAN link is still feasible, and if this can come with a performance gain.

E. Tests

To test the performance of the disks and on the DRBD volume (when using different configuration settings), we decided to use a benchmarking tool called bonnie++ [8].

As a baseline test, we’ll measure the performance of the local VPS disk without using DRBD. After that, we’ll set up DRBD and measured it’s performance and how that changes when using different read-balancing settings. DRBD allows for different read-balancing settings, we will use all and see which one performs best:

- **prefer-local**, DRBD reads mostly from the local disk
- **prefer-remote**, DRBD reads mostly from the remote disk
- **least-pending**, DRBD reads from the disk that has the less pending DRBD operations
- **when-congested-remote**, DRBD reads mostly from the local disk; however, if the local disk is busy it will read from the remote
- **32K-striping**, DRBD reads 32KiB-chunks of data from both disks in an alternating manner
- **512K-striping**, DRBD reads 512KiB-chunks of data from both disks in an alternating manner
- **1M-striping**, DRBD reads 1MiB-chunks of data from both disks in an alternating manner

For full details about the method of performing these tests, refer to Appendix C.
IV. RESULTS

The results from the performed tests can be divided in three categories – local, remote LAN and remote WAN. In our local test environment, we only performed a single test for each setting. We decided to do so, as the local environment was under our complete control. As we simulated the load on our systems in a stable manner, the possibility of outliers in the results was low.

However, in the remote environment, we performed the tests multiple times, as the environment was not under our control. Also, these tests were run in two sessions at different times of the day. By running the tests multiple times, we attempted to estimate a more realistic value for each metric. By running the tests at different times of the day we attempted to identify any possible external influences caused by daily peaks in the usage of other VPS hosts in the same environment. A list with most relevant results tables can be found in Appendix A.

A. Local Results

The results from the local setup we obtained after running all of the tests sequentially. The data is shown in table III.

B. Remote LAN Results

The test results for the case when DRBD was not used are shown in Table IV. The results for the read-balancing settings prefer-local, prefer-remote, least-pending, when-congested-remote, 32K-stripping, 512K-stripping, 1M-stripping we used in DRBD can be found in tables V, VI, VII, VIII, IX, X, XI respectively.

V. ANALYSIS

The analysis of the test results was performed in two stages. We first analyzed the results from the local setup. Afterwards, we calculated the mean and the standard deviation of the values obtained from testing the remote environment. For these results, we used the sample standard deviation, calculated as per Formula (1).

\[ s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2} \]  

(1)

A. Local Setup

The results from the local setup are given in Table III.

As a baseline test, we ran out testing tool when the local disk was not loaded. We measured a reading speed of 121 MiB/s with a read latency of 110 ms. Afterwards, we stressed the disk by simulating actual server usage and could measure a disk reading speed of 25 MiB/s with a reading latency of 454 ms.

The test results from running DRBD show a significant increase of the reading speed when comparing reading from the busy local disk with using DRBD’s prefer-remote setting – we measured a disk reading speed of 60 MiB/s, which constitutes a 240% increase of the reading performance. The read latency is also significantly lower – 41 ms.

The results also show a slight increase in the read speed over the busy local disk when using DRBD’s when-congested-remote setting – we measured an improvement of about 2%. However, in this case, the read latency is significantly higher – we measured a latency of 601 ms, which is actually worse than the measured latency when reading from the stressed local disk (454 ms).

This analysis hints that it is feasible to use DRBD to increase the disk reading performance. However, the test results are obtained in a highly controlled environment – our local setup, and only serve to show that DRBD can, in certain cases, increase disk performance.

B. Remote Setup

As a baseline, we measured the disk reading speed to be about 502 MiB/s (with a standard deviation \( s = 33.3 \) MiB/s) without using DRBD. The measured average reading latency was found to be 8 ms with a standard deviation \( s = 1.2 \) ms.

With regards to disk reading speed, the analysis of the test results from the remote LAN environment (given in Appendix XII) shows that DRBD does not lead to an increase of the disk performance with any of the tested settings. The measured mean disk reading speed when reading from the local disk (without DRBD) is about 20% higher than the highest measured disk reading speed with DRBD – the highest average speed we could measure was about 401 MiB/s with a standard deviation \( s = 13.6 \) MiB/s.

With regards to disk access latency, the analysed results show that the latency is lowest when either reading from the local disk (without DRBD) – 8 ms, or when using DRBD with the prefer-remote (4.5 ms) and when-congested-remote (6.4 ms) settings.

However, by using the standard deviation of the samples, we could measure the variability of the obtained re-
results, and thus, judge the accuracy of the results. When analysing the calculated values for the standard deviation, we observed that the standard deviation is quite high for all measured operations. For the whole set of values, the standard deviation ranges from 2.95% to 176% of the measured average.

By inspecting the raw test data, we can see that the measured values are not consistent and vary greatly in many cases. This variation led us to conclude that the obtained test results are not reliable and do not accurately show the performance of DRBD in an actual VPS environment.

VI. CONCLUSION

From the analysed results of the DRBD’s performance in the local setup (Sec. V), we can conclude that DRBD has a measurable positive effect on the disk reading speed. At least two of the read-balancing options are shown to be effective in the particular setup that we constructed – prefer-local and when-congested-remote.

However, after performing the same tests in an actual hosting environment – with two virtual private servers set up to use DRBD, we could not achieve to verify our earlier results. As explained in Sec. V, the obtained test results from the remote environment could not be considered reliable enough for a conclusion to be made about DRBD’s capability of improving disk reading speed and general input/output performance.

We believe that the very nature of the shared environment in which virtual private servers run is the root cause of the widely varying test results that we obtained. While the initial positive results were obtained when performing tests in a highly controlled local environment, the hosted VPS environment changes under the influence of various factors, including activity on other VPS hosts, infrastructure management, time of the day and others. We believe that these factors are contributing to the varying degree of performance of DRBD and to our inability to correctly judge its capabilities.

Furthermore, we believe that there is room for improvement in the workings of DRBD’s read-balancing option. As this is the first released implementation of the option within the DRBD software suit, we believe that not all possibilities have been explored as of the time of writing.

VII. SUGGESTIONS FOR FURTHER RESEARCH

We believe that DRBD could be improved to include more read-balancing settings, which determine which of the disks is fastest to respond for a particular read and perform the read from it. After the read-balancing setting has matured within DRBD, a similar research may lead to more conclusive results.

users-guide/re-drbdconf.html.
Appendix A: Results & Analysis

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>No Load, No DRBD</td>
<td>86739 K/s</td>
<td>143 ms</td>
<td>121753 K/s</td>
</tr>
<tr>
<td>prefer-remote</td>
<td>9909 K/s</td>
<td>441 ms</td>
<td>60798 K/s</td>
</tr>
<tr>
<td>when-congested-remote</td>
<td>10853 K/s</td>
<td>1941 ms</td>
<td>26028 K/s</td>
</tr>
<tr>
<td>No DRBD</td>
<td>11756 K/s</td>
<td>4442 ms</td>
<td>25824 K/s</td>
</tr>
<tr>
<td>prefer-local</td>
<td>9370 K/s</td>
<td>2843 ms</td>
<td>14253 K/s</td>
</tr>
<tr>
<td>least-pending</td>
<td>10027 K/s</td>
<td>1943 ms</td>
<td>12691 K/s</td>
</tr>
<tr>
<td>512K-striping</td>
<td>10907 K/s</td>
<td>1747 ms</td>
<td>577 K/s</td>
</tr>
</tbody>
</table>

TABLE III: Test results from local setup, ordered by disk-read speed (bonnie++)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>13523 K/s</td>
<td>1948 ms</td>
<td>477587 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>13939 K/s</td>
<td>1844 ms</td>
<td>484703 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>11086 K/s</td>
<td>1845 ms</td>
<td>551020 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>10920 K/s</td>
<td>2447 ms</td>
<td>496539 K/s</td>
</tr>
</tbody>
</table>

TABLE IV: Baseline test results from remote LAN setup – DRBD not used (bonnie++)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>4011 K/s</td>
<td>4745 ms</td>
<td>202334 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>4555 K/s</td>
<td>5741 ms</td>
<td>376816 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>5017 K/s</td>
<td>4347 ms</td>
<td>413644 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>5384 K/s</td>
<td>3541 ms</td>
<td>92588 K/s</td>
</tr>
</tbody>
</table>

TABLE V: Test results from remote LAN setup – DRBD with prefer-local (bonnie++)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>4440 K/s</td>
<td>5740 ms</td>
<td>412945 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>4687 K/s</td>
<td>4844 ms</td>
<td>404862 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>5502 K/s</td>
<td>3645 ms</td>
<td>381574 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>5319 K/s</td>
<td>3945 ms</td>
<td>406221 K/s</td>
</tr>
</tbody>
</table>

TABLE VI: Test results from remote LAN setup – DRBD with prefer-remote (bonnie++)
<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>4925 K/s</td>
<td>5051 ms</td>
<td>413816 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>4789 K/s</td>
<td>3549 ms</td>
<td>381522 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>4869 K/s</td>
<td>6349 ms</td>
<td>380711 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>3919 K/s</td>
<td>7346 ms</td>
<td>170638 K/s</td>
</tr>
</tbody>
</table>

**TABLE VII:** Test results from remote LAN setup – DRBD with least-pending (bonnie++)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>5022 K/s</td>
<td>4941 ms</td>
<td>384938 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>4978 K/s</td>
<td>4643 ms</td>
<td>385837 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>4543 K/s</td>
<td>4543 ms</td>
<td>390961 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>4928 K/s</td>
<td>6169 ms</td>
<td>426875 K/s</td>
</tr>
</tbody>
</table>

**TABLE VIII:** Test results from remote LAN setup – DRBD with when-congested-remote (bonnie++)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>5312 K/s</td>
<td>3654 ms</td>
<td>394677 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>4768 K/s</td>
<td>4940 ms</td>
<td>319004 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>4839 K/s</td>
<td>6954 ms</td>
<td>445508 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>4928 K/s</td>
<td>5346 ms</td>
<td>415411 K/s</td>
</tr>
</tbody>
</table>

**TABLE IX:** Test results from remote LAN setup – DRBD with 32K-striping (bonnie++)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>4570 K/s</td>
<td>4949 ms</td>
<td>160113 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>5166 K/s</td>
<td>4181 ms</td>
<td>336166 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>4531 K/s</td>
<td>5701 ms</td>
<td>78316 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>4972 K/s</td>
<td>4142 ms</td>
<td>291639 K/s</td>
</tr>
</tbody>
</table>

**TABLE X:** Test results from remote LAN setup – DRBD with 512K-striping (bonnie++)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed</td>
<td>Latency</td>
<td>Speed</td>
</tr>
<tr>
<td>Test 1</td>
<td>4466 K/s</td>
<td>5554 ms</td>
<td>390531 K/s</td>
</tr>
<tr>
<td>Test 2</td>
<td>4695 K/s</td>
<td>4444 ms</td>
<td>409275 K/s</td>
</tr>
<tr>
<td>Test 3</td>
<td>4750 K/s</td>
<td>7247 ms</td>
<td>409075 K/s</td>
</tr>
<tr>
<td>Test 4</td>
<td>4764 K/s</td>
<td>6844 ms</td>
<td>162772 K/s</td>
</tr>
</tbody>
</table>

**TABLE XI:** Test results from remote LAN setup – DRBD with 1M-striping (bonnie++)
<table>
<thead>
<tr>
<th>Setting</th>
<th>Disk Write</th>
<th>Disk Read</th>
<th>Random Seek</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speed (s)</td>
<td>Latency (s)</td>
<td>Speed (s)</td>
</tr>
<tr>
<td>No DRBD</td>
<td>12367 K/s (1585)</td>
<td>1971 ms (190)</td>
<td>502462 K/s (33302)</td>
</tr>
<tr>
<td>prefer-remote</td>
<td>4987 K/s (504)</td>
<td>4543 ms (946)</td>
<td>401400 K/s (13681)</td>
</tr>
<tr>
<td>when-congested-remote</td>
<td>4867 K/s (220)</td>
<td>4763 ms (201)</td>
<td>397152 K/s (19991)</td>
</tr>
<tr>
<td>32K-striping</td>
<td>4945 K/s (248)</td>
<td>5224 ms (1360)</td>
<td>393650 K/s (53962)</td>
</tr>
<tr>
<td>1M-striping</td>
<td>4668 K/s (138)</td>
<td>5972 ms (1304)</td>
<td>342913 K/s (120415)</td>
</tr>
<tr>
<td>least-pending</td>
<td>4625 K/s (474)</td>
<td>5573 ms (1645)</td>
<td>336672 K/s (111757)</td>
</tr>
<tr>
<td>prefer-local</td>
<td>4741 K/s (593)</td>
<td>4593 ms (914)</td>
<td>271345 K/s (150653)</td>
</tr>
<tr>
<td>512K-striping</td>
<td>4810 K/s (310)</td>
<td>4743 ms (738)</td>
<td>216558 K/s (118659)</td>
</tr>
</tbody>
</table>

**TABLE XII:** Analysis results from remote LAN setup with standard deviation shown, ordered by disk-read speed (bonnie++)
Appendix B: Compiling and Packaging DRBD

The procedure below gives the steps performed to build and install DRBD from source on both our local and our remote experimental setup systems. Although this procedure was performed on a Ubuntu 11.04 system, this procedure is generic and can apply to other Debian-based distros. The main thing that will need to be taken into account in that case is the kernel version.

1. Obtaining the Linux Headers

We need the Linux kernel headers to be able to build a kernel module:

```
apt-get install linux-headers-2.6.38-13-server
```

2. Configuring the Build

We specify that we want to build a kernel module with `--with-km`:

```
./configure --with-km --sysconfdir=/etc --localstatedir=/var
```

3. Building the DRBD Utilities and Kernel Module

We specify the path to the Linux headers with `KDIR=/lib/modules/2.6.38-13-server/build`:

```
make KDIR=/lib/modules/2.6.38-13-server/build
```

4. Verifying the Module

After building, we want to check whether the module has been built successfully and is a valid kernel module:

```
/sbin/modinfo drbd/drbd.ko
filename: drbd/drbd.ko
alias: block-major-147-*
license: GPL
version: 8.4.1
description: drbd – Distributed Replicated Block Device v8.4.1
author: Philipp Reisner <phil@linbit.com>, Lars Ellenberg <lars@linbit.com>
srcversion: E0922D7BF69292DEE285573
depends: libcrc32c
vermagic: 2.6.38-13-server SMP mod_unload modversions
parm: minor_count: Approximate number of drbd devices (1–255) (uint)
parm: disable_sendpage: bool
parm: allow_oos:DONT USE! (bool)
parm: proc_details:int
parm: enable_faults:int
parm: fault_rate:int
parm: fault_count:int
parm: fault_devs:int
parm: usermode_helper:string
```
5. Verifying the Module

The .deb build process rebuilds all of DRBD’s sources, so you can perform the configure step and then skip to here if you only want to build the .deb packages. By performing the previous steps, however, we verified that the build process is successful and the built binaries are valid.

To build the .deb package, we need to clone the GIT repository for DRBD and copy the debian folder, which contains utilities for building DRBD into .deb packages. The folder then has to be copied in the source tree for the version that we are using.

Before creating the packages, we need to install the required prerequisites for the creation of .deb packages in general:
```
apt-get install debhelper debconf-utils docbook-xml docbook-xsl dpatch xsltproc
```

After performing these steps, the packages can be built with the following command:
```
dpkg-buildpackage -b -uc
```

The -b flag instructs dpkg-buildpackage to create a binary-only package for the DRBD utilities. This is not possible for the kernel module, as the .deb package for the module will be a source package.

The -uc flag instructs dpkg-buildpackage to create unsigned .deb packages.

The resulting .deb packages are:
```
drbd8-module-source_8.4.1-0_all.deb
drbd8-utils_8.4.1-0_amd64.deb
```

6. Installing the .deb Packages

As the OS we are installing on has to build the module from the source .deb package, there are certain prerequisite software packages that need to be installed. This step is dependent on the version of the kernel that is running on the host system:
```
apt-get install \
    module-assistant linux-headers-2.6.38-13 linux-headers-2.6.38-13-server \
    bsdmainutils debhelper dpatch gettext groff-base html2text intlttool-debian \
    libcroco3 libmail-sendmail-perl libpipeline1 libsys-hostname-long-perl \
    libunistring0 libxml2 man-db patchutils po-debconf sgml-base xml-core
```

Afterwards, we need to manually install the .deb packages:
```
dpkg -i drbd8-module-source_8.4.1-0_all.deb
dpkg -i drbd8-utils_8.4.1-0_amd64.deb
```

This step installs the DRBD utilities, but the module still needs to be built from the source provided by the `drbd8-module-source_8.4.1-0_all.deb` package:
```
module-assistant auto-install drbd8
```

After the module build successfully, we can load it into the running kernel:
```
modprobe drbd
lsmod | grep drbd
```
```
drbd 308688 0
libcrc32c 12644 1 drbd
```

When the module is loaded, we can verify that it is operating correctly by checking the contents of `/proc/drbd`:
```
cat /proc/drbd
```
```
version: 8.4.1 (api:1/proto:86-100)
GIT-hash: 91b4c048c1a0e06777b5f65d312b38d47abaea80 build by root@amuru,
          2012-03-21 13:56:56
```
Appendix C: Testing

1. Generating Load

To generate load in the local setup, we used blogbench with randomized parameters to simulate the disk load on an actual busy server. The script is shown below.

```bash
#!/bin/bash

BBENCH=./blogbench
while [ : ]; do
    ITER='echo $RANDOM | cut -c 1-1'
    READERS='echo $RANDOM | cut -c 1-1'
    COMMENTERS='echo $RANDOM | cut -c 1-1'
    echo Running Blogbench...
    $BBENCH -d /tmp -i $ITER -r $READERS -w 1 -W 1 -c $COMMENTERS -s 6

    echo Cleaning up...
    rm -rf /tmp/blog*
done
```

2. Automated Testing

To automate the process of testing under different DRBD settings, we created a simple script, which performs a set of tests, then adjusts the DRBD settings and performs the tests again. The script can be found below:

```bash
#!/bin/bash

cat /proc/drbd >/tests/prefer-local1-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/prefer-local1
sleep 10
sync
cat /proc/drbd >/tests/prefer-local1-drbd
cat /proc/drbd >/tests/prefer-local2-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/prefer-local2
sleep 10
sync
cat /proc/drbd >/tests/prefer-local2-drbd
dd if=/dev/drbd1 of=/dev/null bs=1M count=1024 2>/tests/prefer-local-dd

sed -e 's/prefer-local/prefer-remote/' /etc/drbd.d/r0.res > /tmp/r0.tmp
mv /tmp/r0.tmp /etc/drbd.d/r0.res
drbdadm adjust r0

cat /proc/drbd >/tests/prefer-remote1-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/prefer-remote1
sleep 10
sync
cat /proc/drbd >/tests/prefer-remote1-drbd
cat /proc/drbd >/tests/prefer-remote2-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/prefer-remote2
sleep 10
sync
cat /proc/drbd >/tests/prefer-remote2-drbd
```
dd if=/dev/drbd1 of=/dev/null bs=1M count=1024 2>/tests/prefer-remote-dd

sed -e 's/prefer-remote/least-pending/ /etc/drbd.d/r0.res > /tmp/r0.tmp
mv /tmp/r0.tmp /etc/drbd.d/r0.res

drbdadm adjust r0

cat /proc/drbd >/tests/least-pending1-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/least-pending1
sleep 10
sync
cat /proc/drbd >>/tests/least-pending1-drbd
cat /proc/drbd >/tests/least-pending2-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/least-pending2
sleep 10
sync
cat /proc/drbd >>/tests/least-pending2-drbd
dd if=/dev/drbd1 of=/dev/null bs=1M count=1024 2>/tests/least-pending-dd

sed -e 's/least-pending/when-congested-remote/ /etc/drbd.d/r0.res > /tmp/r0.tmp
mv /tmp/r0.tmp /etc/drbd.d/r0.res

drbdadm adjust r0

cat /proc/drbd >/tests/when-congested-remote1-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/when-congested-remote1
sleep 10
sync
cat /proc/drbd >>/tests/when-congested-remote1-drbd
cat /proc/drbd >/tests/when-congested-remote2-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/when-congested-remote2
sleep 10
sync
cat /proc/drbd >>/tests/when-congested-remote2-drbd
dd if=/dev/drbd1 of=/dev/null bs=1M count=1024 2>/tests/when-congested-remote-dd

sed -e 's/when-congested-remote/32K-stripping/ /etc/drbd.d/r0.res > /tmp/r0.tmp
mv /tmp/r0.tmp /etc/drbd.d/r0.res

drbdadm adjust r0

cat /proc/drbd >/tests/32K-stripping1-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/32K-stripping1
sleep 10
sync
cat /proc/drbd >>/tests/32K-stripping1-drbd
cat /proc/drbd >/tests/32K-stripping2-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/32K-stripping2
sleep 10
sync
cat /proc/drbd >>/tests/32K-stripping2-drbd
dd if=/dev/drbd1 of=/dev/null bs=1M count=1024 2>/tests/32K-stripping-dd

sed -e 's/32K-stripping/512K-stripping/ /etc/drbd.d/r0.res > /tmp/r0.tmp
mv /tmp/r0.tmp /etc/drbd.d/r0.res

drbdadm adjust r0

cat /proc/drbd >/tests/512K-stripping1-drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/512K-stripping1
sleep 10
sync
cat /proc/drbd >>/tests/512K–striping1–drbd
cat /proc/drbd >>/tests/512K–striping2–drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/512K–striping2
sleep 10
sync
cat /proc/drbd >>/tests/512K–striping2–drbd
dd if=/dev/drbd1 of=/dev/null bs=1M count=1024 2>/tests/512K–striping–dd

        sed -e ‘s/512K–striping/1M–striping/’ /etc/drbd.d/r0.res > /tmp/r0.tmp
        mv /tmp/r0.tmp /etc/drbd.d/r0.res
        drbdadm adjust r0

cat /proc/drbd >/tests/1M–striping1–drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/1M–striping1
sleep 10
sync
cat /proc/drbd >/tests/1M–striping1–drbd
cat /proc/drbd >/tests/1M–striping2–drbd
/usr/sbin/bonnie++ -d /mnt/drbdvol/ -u root -q >/tests/1M–striping2
sleep 10
sync
cat /proc/drbd >/tests/1M–striping2–drbd
dd if=/dev/drbd1 of=/dev/null bs=1M count=1024 2>/tests/1M–striping–dd