pISSN 1976-913X
eISSN 2092-805X

J Inf Process Syst, Vol.9, No.2, June 2013 http://dx.doi.org/10.3745/JIPS.2013.9.2.205

An Analysis of Replication Enhancement for a High Availability Cluster

Sehoon Park*, Im Y. Jung*, Heonsang Eom* and Heon Y. Yeom*

Abstract—In this paper, we analyze a technique for building a high-availability (HA) cluster system. We propose what we have termed the 'Selective Replication Manager (SRM),' which improves the throughput performance and reduces the latency of disk devices by means of a Distributed Replicated Block Device (DRBD), which is integrated in the recent Linux Kernel (version 2.6.33 or higher) and that still provides HA and failover capabilities. The proposed technique can be applied to any disk replication and database system with little customization and with a reasonably low performance overhead. We demonstrate that this approach using SRM increases the disk replication speed and reduces latency by 17% and 7%, respectively, as compared to the existing DRBD solution. This approach represents a good effort to increase HA with a minimum amount of risk and cost in terms of commodity hardware.

Keywords— High-Availability Cluster, Replication Enhancement, SRM, DRBD

1. INTRODUCTION

Various proposals have sought to increase data reliability and availability on cluster networks. Database systems have their own protection schemes for failover conditions so that they can continue to serve their applications and user processes. However, HA (High-Availability) [1] requires a high cost in that it uses a complex code in the DBMS. This also necessitates an additional setup procedure for database management and administration, as well as for the setup of some specialized hardware [2].

In this paper, we present what is termed the 'Selective Replication Manager (SRM)', which interacts with the system status of the Linux operating system. The SRM monitors the system status in real time and chooses the best values for configuring the parameters of the cluster system. These parameters are the replication buffer size, the TCP size, the bandwidth of the synchronization process, and the methods of the replication protocol. We implemented the SRM based on the Distributed Replication Block Device (DRBD) [3, 4] to create a reliable and high-availability cluster network on commodity hardware. The DRBD is built as a block device function in the recent Linux Kernel (version 2.6.33 or higher). It provides disk replication over the network and can also be easily implemented on commodity hardware. However, the synchro-

* The Seoul R&D Program (No. JP110034) supported this work. The ICT at Seoul National University provided the research facilities for this study.
Manuscript received March 28, 2012; accepted May 22, 2012.
Corresponding Author: Im Y. Jung
* Dept. of Computer Science and Engineering, Seoul National University, Seoul, Korea (tshpark@gmail.com, {iyjung, hseom, yeom}@snu.ac.kr)

nous replication protocol strongly depends on the actual writing speed of the disk and network bandwidth.

We propose several experimental considerations to increase the disk replication throughput and to reduce the disk and network latency in different hardware combinations by using the SRM with adaptively changing parameters that are based on the software. The empirical results show that SRM facilitates a 10-15% improvement in various ways.

2. BACKGROUND

The DRBD driver code is integrated into the vanilla Linux Kernel with the 2.6.33 release or higher. It is implemented as a block device to create a HA cluster network and can be easily utilized on commodity hardware, such as a storage device or a database system. It is similar to RAID-1 [5-7] mirroring, which creates an exact copy set of data on two disks. However, it is distinguished in that the entire block device can be replicated over the network.

Figure 1 describes an overview of the traditional DRBD architecture. It consists of primary and secondary devices. All of the data from the primary device will be replicated to the secondary device over the TCP/IP network for high availability and to maintain good data reliability. If a failure occurs in the primary node for any reason, the secondary device will take the place of the primary device within a few seconds without any failure detection at the application/user level. As soon as the secondary node detects a failure on the primary node, the standby device will be promoted to the active node, and the replicated device is then mounted by the DRBD to maintain the availability of the service. The DRBD works on the top of a block device, which can be a HDD or logical volumes of the LVM (Logical Volume Manager). This replication scheme for disks or databases can be executed synchronously [8]. (i.e., when any type of data has been produced on the primary node, it will be sent to the secondary node and will wait for the acknowledgement of the write completion on the disk from the secondary node.) If the primary device receives the completion *ack* notification, the data can be updated on the physical disk in order to increase data reliability. This also guarantees data consistency, as the primary disk verifies that the backup data has been written safely on the secondary device before updating it on the local disk.

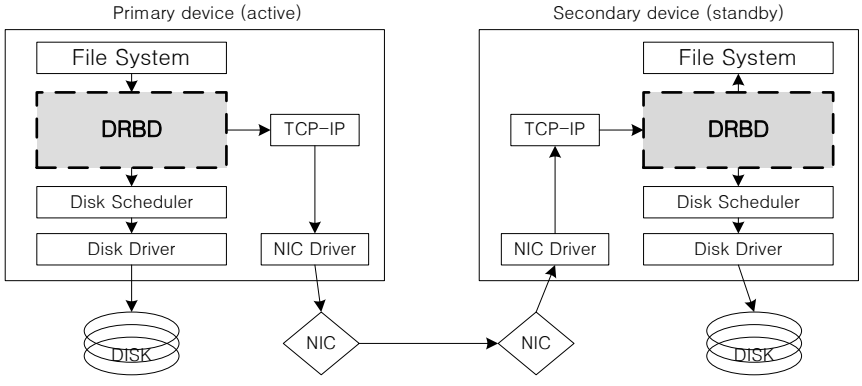

Fig. 1. DRBD architecture

Therefore, the performance of the synchronous DRBD cluster is mainly limited to the speed of the secondary disk device. For the same reason, the replication performance on the remote cluster also depends on the network bandwidth.

However, a HDD (SATA) is a mainstream disk on a commodity hardware cluster, and its random write speed is very low, at about 3.6MB/s, whereas the sequential write speed is 30.9MB/s. These figures are generated by an Iozone [9] benchmark test as writing a file set that is 1 GB in size with a record size of 4 KB. Therefore, the throughput performance of the DRBD mainly depends on the lower disk speed, especially in the case of random write operations.

We utilized various methods in an effort to increase the DRBD throughput and to optimize its latency. First, we integrated a DRAM-based SSD device as the secondary disk. The following heterogeneous combinations consist of a DRAM-based SSD with a HDD, which is referred to as DRAM-SSD in this paper. The DRAM-SSD has excellent random write performance based on the SSD architecture [10-12]. However, the DRAM-SSD is not typically recoverable in the case of a power failure because it is a DRAM-based storage device (i.e., the DRAM-SSD is volatile storage unless it has a battery pack). However, it may be better for the DRAM-SSD device to be used as a secondary device rather than as a primary device, as a standby device will not require data consistency as much as a primary device. It also has advantages in terms of the performance gain that is offered and the good failure resilience of the replication cluster. In the same experiment of the Iozone test of a HDD as a secondary device, the setup with the DRAM-SSD as a secondary device resulted in a fourfold increase in performance with a HDD attached as the primary device.

Second, we modified the maximum buffers, which the replicated block device allocates for writing data to the disk, and permitted a change to the asynchronous method selectively in accordance with the network bandwidth condition and congestion. Under most circumstances, an adaptive setting depending on the TCP size and the disk synchronization bandwidth also shows a reasonable performance enhancement with the combination of a HDD as the primary device and a DRAM-SSD device as a backup device.

3. SELECTIVE REPLICATION MANAGER (SRM)

In this paper, we propose the Selective Replication Manager, which monitors specific aspects of the system status, such as the CPU load, memory utilization rate, and network overhead.

Figure 2 shows an outline of the SRM architecture. The DRBD consists of the following two different components for replication in the kernel: the admin layer and the block device. The admin layer is a tool that is used to control and configure the block device driver, which is integrated into the DRBD system as the cluster management software.

We suggest several options for tuning the performance considering the increase in the disk throughput and the reduction in the latency.

- Asynchronous mode: rather than synchronous data replication, we evaluate the option of asynchronous data replication
- Adaptive synchronization: the network bandwidth for the data replication node
- Utilize the max-buffer size: increase to a maximum value of 8,000 from the default
- Increase the MTU size: the maximum transmission unit size of the network interface on the

TCP/IP protocol

In Figure 2, the SRM, which is located on the top of the DRBD layer, controls the admin module. At the same time, it interacts with Operating System (OS). The SRM monitors the OS status in real time and sends a request message to the DRBD admin layer to adjust its configuration selectively. For example, the SRM changes the data replication pattern to the asynchronous mode from the synchronous setting when the SRM detects that the current network bandwidth is very limited and that congestion is high. In the asynchronous mode, an active device will not wait for acknowledgement from a standby device. Instead, the primary node will start to update the disk whenever files are modified and will simultaneously write to the TCP buffer to send a data to the secondary node. This method can increase the DRBD throughput because the primary device does not have to wait for an acknowledgement from the destination node.

However, the asynchronous protocol will not check whether the data that is sent over the network from the primary device was written safely on the secondary device. Whenever the updating of data occurs on the primary device, it will update the primary disk and at the same time it sends the data to the secondary device over the TCP/IP network. Therefore, the performance of the asynchronous protocol is no longer limited by the network bandwidth or by the secondary disk speed. The asynchronous protocol can experience a reduction in data consistency. However, using the synchronous mode with a limited network bandwidth will degrade the overall system performance. When we adopt the asynchronous protocol, data loss can occur in the event of a power failure, as the primary data is not always guaranteed to be replicated on the secondary disk. All of the data sent in the buffer will be lost when the primary power fails. Therefore, it is very important to use both of the protocols selectively according to the SRM depending on the network congestion status.

For example, the SRM will choose the asynchronous protocol if it detects that the network bandwidth is temporarily low. After the network condition is recovered to normal, the SRM returns to the synchronous mode to guarantee data consistency. Another example of the SRM is when the synchronization speed is set to the maximum speed of the NIC, which is '1 Gbps' in our experiment. In this case, the network bandwidth of the full system will be occupied only by DRBD services. The SRM recognizes that other resources also need network-related I/O jobs, and therefore adjusts the DRBD synchronization bandwidth in order to meet the needs of the other resources.

Moreover, the default buffer size that the DRBD allocates for writing data to the disk can also be modified selectively by the SRM up to a maximum size of 8,000 KB from the default setting

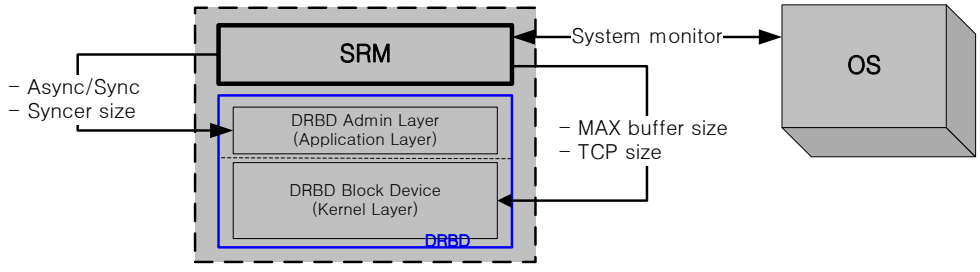

Fig. 2. SRM on a DRBD

of 2,048 KB. Increasing the size of the buffer improves the performance when the system transfers large files as compared to small files. Furthermore, this buffer size modification should be made in conjunction with a change to the MTU size of the network interface. The SRM provides a considerable advantage in terms of its ability to utilize the system resources and for its ability to improve the data replication performance. The SRM architecture is used for the DRBD configuration and can be applied to any application as a system analyzer. In Section 4, the experimental result shows that the SRM leads to a 10-15% improvement of various factors.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Environment

Our experimental setup consisted of two servers, each of which was equipped with four Quad-Core 2.53 GHz Intel Xeon processors with 24 GB of RAM. The primary server has a 1 TB 7200 rpm HDD and 8 GB of a DDR2 ECC DRAM-SSD device (Section 4.2) as a secondary disk. Both the primary and secondary systems run on CentOS release 5.7 with the 2.6.36 Linux Kernel compiled with the DRBD 8.3.12. Both servers have a 1 Gbps NIC connected by a 10 Gbps switch, and each server has an Infiniband 10 Gbps (in SDR) HCA that is directly connected by a 4X Infiniband cable (Section 4.8)

4.2 DRAM-SSD Device

We utilized a DRAM-SSD (Solid State Disk), which offers FPGA-level functionality. The parameters of the DRAM based SSD device, including the hardware-latency when writing and overwriting a page, can be found in the literature [13, 14]. The characteristics of the DRAM-SSD are described in Table 1.

Table 1. The DRAM-SSD device

Memory	DDR2 8GB ECC DRAM
Capacity	64GB
Channel	PCI Gen1 x 4
FPGA	Quartus II 10.1 (32 bit)
Performance	Read: 700MB/s, Write: 650 MB/s
Latency	Read: 48ns, Write: 150ns, Overwrite: 200ns

4.3 Disk Combinations

We used various storage disk combinations to find the best match in terms of the disk writing speed. In Table 2, we classified the disk setup from A to F in regards to the primary disk and the secondary disk. Figure 3 shows how each setup can perform in the sequential and random write as compared to the Iozone benchmark, which uses a file size that is set to 1 GB and a record size of 4 KB. We ran 100 trials per setup to generate an average write speed for each setup.

As shown in Figure 3, the first two columns (A, B) show the speed of the sequential and random write operations in the local device as references, while the others, C to F, are evaluated in the remote cluster by the DRBD. According to the graph, the DRAM-SSD device has very high

Table 2. Disk setup for primary/standby disks

Seq/Ran Write (MB/s)	Primary device	Secondary device	Setup
Local	HDD	-	A
	DRAM-SSD	-	B
DRBD	HDD	HDD	C
	DRAM-SSD	DRAM-SSD	D
	DRAM-SSD	HDD	E
	HDD	DRAM-SSD	F

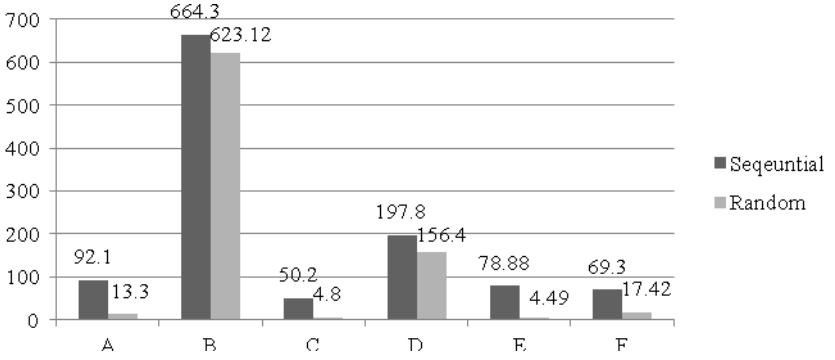

Fig. 3. Disk throughput (MB/s)

throughput for the sequential and the random write speed, as compared to the HDDs. When comparing (B) and (D), the write speed of the DRAM-SSD drops from 623.12 MB/s to 156.4 MB/s in the random write pattern on account of the replicated block device overhead and the network overhead. Here, we used the Infiniband 10 Gbps device to guarantee that the network speed would be as high as 1250 MB/s at most, as compared to 125 MB/s for the 1 GB NIC at its theoretically maximum in setup (D). According to the graph, setup (D) appears to offer the best performance in terms of the DRBD configuration, but the cost of a DRAM-SSD device is much higher compared to a HDD because it is a DRAM-based device. Hence, setup (D) should not be considered to be the best solution. For different types of device formations, specifically (E) and (F), with a DRAM-SSD as both the primary and the secondary device, the total performance is higher when a faster device is attached as a secondary device (F).

As shown in the (E) setup, in this case even the DRAM-SSD speed dropped to 77.88 MB/s from 197.8 MB/s due to the replication process to the secondary device, which is a HDD. Setup (F) also has a similar result for the sequential write operation, but it shows an improvement for the random write operation, from 4.49MB/s to 17.42MB/s. Therefore, a DRAM-SSD could be a good alternative device to use as a standby device, as it offers a performance gain despite the fact that a DRAM-SSD is a rather expensive device. In terms of the random write speed, having a DRAM-SSD as the secondary device (F) improves the speed fourfold compared to setup (E), with a HDD as the secondary device. Finally, we propose the (F) setup as having the best performance with the lowest latency and therefore used this setup for the rest of the experiment.

4.4 Adaptive Buffer Size with The HDD - DRAM-SSD

Figure 4 shows the sequential and random write speed for the (F) setup with different values of the max-buffer size, which the DRBD allocates for writing data to the disk device. Depending on the max buffer size, the speed of a sequential and a random write increases by 17.2% and 10.7%, respectively, as the max-buffer value increases from 2,048 KB as the default to 8,000 KB with the same Iozone setup that was discussed in Section 4.3. A larger buffer does not always lead to better performance, as this depends on the size of the target file used in the transmission. The SRM will calculate the best buffer size to selectively correspond to the size of files for replication to ensure high throughput and low latency.

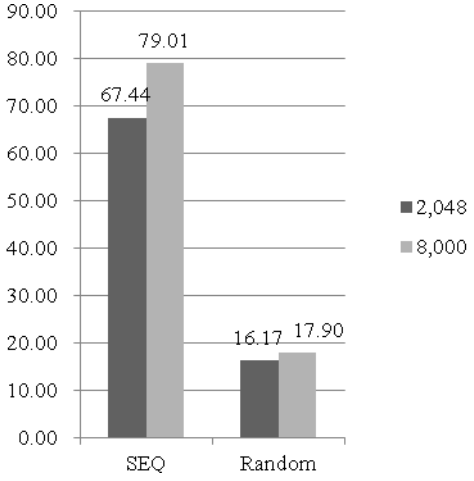

Fig. 4. Disk write speed (MB/s) with different DRBD buffer sizes

4.5 TCP-Send Buffer Size Adjustment

The TCP buffer size is another main factor influencing the performance of the SRM as it attempts to undertake replication over a network. This buffer is a memory buffer for outgoing TCP traffic [15]. Extending the size of the TCP-send buffer essentially reduces the latency, as it decreases the total number of requests involving the *send()* function on the TCP/IP protocol. The default value is 128 KB, and as the TCP buffer size is increased by the SRM. The write speed (MB/s) by Iozone with the same configuration that was discussed in Section 4.3 increases linearly to 2,048 KB, as shown in Figure 5. In our experiment, the performance gain was 14% with a size of 2 MB in (F) in combination with the SRM, but the throughput was regressed by more than 2 MB.

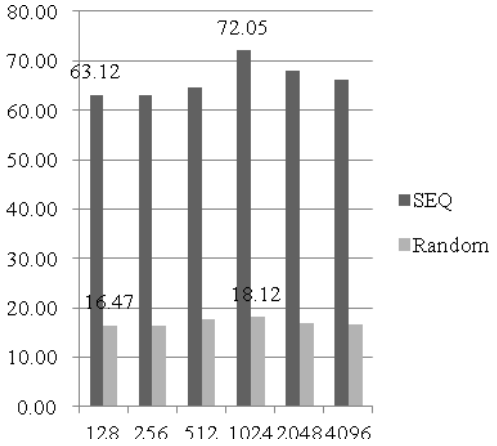

Fig. 5. Disk write speed (MB/s) with different TCP send buffer sizes

4.6 Latency Optimization

The latency test for the DRBD was carried out using 'dd' [16], which is a common Linux program that supports low-level copying and the conversion of raw data. Figure 6 shows the average time in milliseconds during the writing of a small file in the local device (shown as sh-ll-dev) and with a DRBD connected device (as shown as sh-dev). The default block size in the Linux Ext3 file system is 4096 (4 KB) and the minimum size is 512 bytes. According to the minimum size, the experiment measures the total time to write 512 bytes 100,000 times. We then compared each case (the DRBD and the local device) by calculating the average time for writing one file. The overhead data replication latency via the DRBD was only 2.7% as compared to the local drive. The first run showed somewhat higher latency due to the cold data, but overall, Figure 6 presents that the latency with setup F (HDD & DRAM-SSD) was reasonably low and does not affect the total performance in the DRBD.

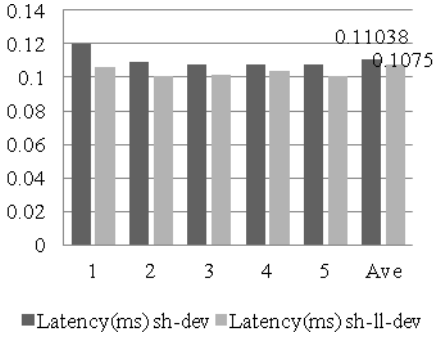

Fig. 6. Latency time for writing a file (ms)

4.7 Asynchronous and Synchronous Comparisons

We tested both protocols (async/sync) with the same hardware setup (F). However, as shown in Figure 7, there were no major differences between the protocols, as our experiment was conducted within the same datacenter configuration. The network bandwidth was large enough that we could dismiss the difference on the performance advantage of the asynchronous protocol. The latency evaluated in Section 4.6 was also very low within the cluster. Therefore, using the synchronous protocol in a high-bandwidth cluster is beneficial with the lowest overhead because in this way it is possible to increase the data consistency.

However, if the cluster is connected in a WAN (Wide Area Network) environment and the network bandwidth is somewhat limited, the asynchronous protocol will clearly increase the total throughput during the replication of the cluster. The SRM detects the network congestion in real time and applies the replication protocol in contrast to the asynchronous method. This will not only increase the replication performance, but it will also enhance the utilization of the resources of the primary machine. Figure 7 presents the synchronous and asynchronous replication results over a network running on a 1 GB NIC via a Gigabit switch with a HDD as the primary disk and a DRAM-SSD as the secondary device. Although the test was conducted in the same datacenter, the asynchronous method still has a performance gain of about 7.7% and 4.1% in the sequential and random write modes, respectively.

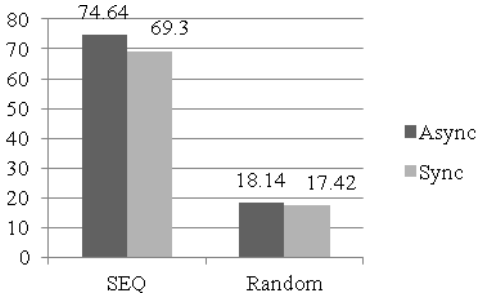

Fig. 7. Disk speed (MB/s) in the asynchronous mode

4.8 Synchronization Bandwidth Over Infiniband

We have also constructed a DRBD cluster using the Infiniband [17, 18] network setup to verify how a next-generation network device would affect the replication performance. The device we used here is the Infiniband III Lx single-port 4X Infiniband HCA [19], which has a SDR (single data rate) of 10 Gbps and a DDR (double data rate) of 20 Gbps. Because the current DRBD setup only supports the TCP/IP stack, we used an IB_IPoIB module, which converts the Infiniband device to allow it to operate with an IP module (i.e., it facilitates IP encapsulation with the Infiniband device). In a comparison of the throughput performance via a Gigabit Ethernet with the same disk combination (F), as depicted in Figure 3, Infiniband HCA gives almost the same performance except for the DRAM-SSD and DRAM-SSD (D) combination. Overall, the results using Infiniband were quite trivial due to the IP encapsulation overhead and the low throughput of the HDD performance.

5. CONCLUSION

A SRM (Selective Replication Manager) located on top of a Linux block device monitors the OS status in real time and sends a request (i.e., the optimization values) to the DRBD admin tool layer to increase the disk throughput and to reduce latency.

DRBD is an efficient replicated block device solution for creating a reliable HA cluster network using commodity hardware, which involves little or no code changes for an existing setup. To increase the overall performance, we determined that the main bottlenecks are particularly the speed of the disk device during a random write operation and the network bandwidth.

We proposed a novel hardware combination to increase the disk replication speed over a network. We adapted a DRAM-SSD as a standby device, and this offers a fourfold increase in the performance over a HDD according to the results of a random write test. Although the DRAM-SSD has a drawback because it is a DRAM-based volatile storage device, it offers a meaningful advantage as a secondary backup device. Alternating between synchronous and asynchronous protocols based on the current network condition and adjusting the maximum synchronization bandwidth by the SRM also improves the disk replication speed. The size of the max buffer and the MTU size are also factors that can be adjusted to reduce the latency during the writing of data as part of the DRBD solution.

Our experimental evaluation demonstrated that the SRM process will improve the disk replication performance and will reduce latency by a maximum of 17% and 7%, respectively, as compared to the traditional HA clusters setup.

6. FUTURE WORKS

A HDD is somewhat unsuitable for write-intensive workloads due to its slow random write speed when building a high-availability cluster. To overcome this issue, we propose a new network mirroring technique that writes in an append-only fashion, similar to the LFS [20-22] implementation. It is designed for high writing throughput, and all updates of the data and metadata are written sequentially in a continuous stream, which is known as a log.

In addition, the replicated log file structure system in the secondary device should be enabled as a failure-recovery function when the primary device fails in a high-availability cluster. Therefore, a novel method will be necessary where we convert the file system from the secondary log structure file system to the primary device, which has a regular ext2/ext3 Linux file system.

REFERENCES

[1] Linux-HA Project. [online] http://www.linux-ha.org/doc/

[2] D. Komo. Microsoft SQL Server 2008 R2 High Availability Technologies, *White Paper*. *Microsoft*, 2010.

[3] Reisner, P., Ellenberg, L., Drbd v8 Replicated storage with shared disk semantics. In: *Proceedings of the 12th International Linux System Technology Conference (Linux-Kongress)* 2005, Hamburg, Germany (October 11-14, 2005)

[4] Reisner, P., Ellenberg, L., Distributed Replicated Block Device (DRBD) documentation at http://www.drbd.org. January, 2012.

[5] D. Patterson, G. Gibon, and R. Katz, A case of for redundant arrays of inexpensive disks (RAID), *vol. 17. ACM*, 1988

[6] G. A. Gibson, Redundant Disk Arrays: Reliable, Parallel Secondary Storage. *MIT Press*, Cambridge, MA, 1992.
[7] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz and D. A. Patterson, 'RAID: High-performance, reliable secondary storage,' *ACM Computing Surveys*, 26, (2), 145-185 (1994).
[8] C. Dirik and B. Jacob. The performance of PC solid-state disks (SSD) as a function of bandwidth, concurrency, device architecture, and system organization. *In Proceedings of the 36th IEEE International Symposium on Computer Architecture (ISCA)*, 2009.
[9] "Iozone." http://www.Iozone.com, January, 2012.
[10] S.-W. Lee, B. Moon, and C. Park. Advances in flash memory SSD technology for enterprise database applications. *In Proceedings of the ACM SIGMOD*, 2009.
[11] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim. A case for flash memory SSD in enterprise database applications. *In Proceedings of the ACM SIGMOD*, 2008.
[12] P. Reisner, Distributed replicated block device, 2002, http://www.drbd.org/fileadmin/drbd/publications/drbd _lk9.pdf
[13] LEE, B.C., IPEK., MUTLU, O., AND BURGER, D. Architecting phase change memory as a scalable dram alternative. *ISCA* '09, pp.2-13.
[14] QURESHI, M. K., SPINVASAN, V., AND RIVERS, J. A. Scalable high performance main memory system using phase-change memory technology. *In Proceedings of the 36th annual ISCA* '09(2009), pp.24-33.
[15] Kelly T. Scalable TCP: improving performance in high speed wide area networks. *First International Workshop on Protocols for Fast Long-Distance Networks*, 2003.
[16] "dd" http://en.wikipedia.org/wiki/Dd_(Unix), January, 2012.
[17] Pfister, F., Gregory, A Introduction to the InfiniBand Architecture, pages 617-632. *IEEE Press and Wiley Press*, 2001.
[18] IBM Corporation. IBM InfiniBand product advance summary datasheet. http://www.chips.ibm.com/products/infiniband, August, 2001.
[19] "Infiniband III Lx single-port 4X InfiniBand HCA" http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=19&menu_section=41, January, 2012.
[20] Rosenblum, Mendel. , John K. Ousterhout. The design and implementation of a log structured file system. *In Proceedings of the Thirteenth Symposium on Operating Systems Principles*, pages 1{15, Paci_c Grove, California, October, 1991. *ACM Press*.
[21] John Ousterhout and Fred Douglis. Beating the I/O bottleneck: a case for log-structured file systems. *Operating Systems Review* 23(1):11-27, January, 1989.
[22] M. Seltzer, K. Bostic, M. K. McKusick, and C. Staelin. An Implementation of a Log-Structured File System for UNIX. *Proc. of the Winter 1993 USENIX Conf.*, San Diego, CA, January, 1993, 307-326.

Sehoon Park

Sehoon Park received the BS degree in Information and Computer Science from University of California, Irvine, USA in 2005. Since December 2005, he had been a software engineer for 5 years in Mobile Device Division of Samsung Electronics. Currently, he is a M.S. candidate in the school of Computer Science and Engineering at Seoul National University. His research interests include mobile system computing, web based offloading architecture and distributed computing system.

Im Y. Jung

Im Young Jung received the first BS degree in chemistry from Pohang University of Science and Technology in 1993 and the second BS degree in computer science from Seoul National University in 1999. And she received her MS and PhD degrees in computer science and engineering from Seoul National University in 2001 and 2010 each. Since February 2001, she had been a researcher for 3 years in the Electronics and Telecommunications Research Institute (ETRI), South Korea. Now, she is BK Assistant Faculty in Seoul National University. Her current research interests include security of data and system, distributed computing, cloud computing, e-Science grid, data management system, workflow system.

Hyeonsang Eom

Hyeonsang Eom received his B.S. degree in Computer Science and Statistics from Seoul National University (SNU), Seoul, Korea, in 1992. He received his M.S. and Ph.D. in Computer Science from the University of Maryland at College Park, Maryland, USA, in 1996 and 2003, respectively. He worked as an intern in the Data Engineering Group at Sun Microsystems, USA, in 1997. Before becoming a professor at SNU in 2005, he worked as a senior engineer in the Telecommunication R&D Center at Samsung Electronics. He is currently a Professor in the Department of Computer Science and Engineering at SNU. His research interests are in the areas of cloud computing, next-generation OS, high performance storage systems, energy efficient systems, fault tolerant systems, digital rights management and information dynamics

Heon Y. Yeom

Heon Young Yeom received his B.S. degree in Computer Science from Seoul National University, Seoul, Korea in 1984 and the M.S. and Ph.D. degrees in Computer Science from Texas A&M University, College Station, in 1986 and 1992, respectively. From 1986 to 1990, he was with Texas Transportation Institute as a Systems Analyst and from 1992 to 1993, he was with Samsung Data Systems as a Research Scientist. He joined the Department of Computer Science, Seoul National University, in 1993, where he currently is a Professor and teaches and conducts research on distributed systems, and transaction processing