October 2005

DB2 Information Management Software

DB2 Universal Database (DB2 UDB) for Linux with DRBD and Heartbeat:

A Low-Cost Open Linux High Availability Solution

By

Anand Subramanian (anands@ca.ibm.com)
Melody Ng (mysng@ca.ibm.com)

IBM Toronto Software Lab

Contents

1.0 Introduction

1.1 Objective

This paper discusses a low-cost high availability (HA) solution for the IBM® DB2® Universal Database™ product (DB2 UDB) Version 8.2 on the Linux® platform using the open source Linux-HA project (Heartbeat) and Distributed Replicated Block Device (DRBD) software packages.

1.2 Background

Over the last few years, the increasingly global nature of modern enterprises has led to a ubiquitous demand for increased availability in most business-critical systems and applications such as enterprise databases, data marts, data warehouses, e-mail servers, and application servers.

In information technology (IT), HA refers to systems that are architected to provide business continuity through the seamless failover of systems and applications to standby systems if primary systems fail. HA solutions include technologies that help to minimize the time in which the system is unavailable while supporting failover with minimal manual intervention in case of a disaster situation. A good HA solution must take into account various important factors that include failure detection time, failover time, the manual intervention required to bring up the systems, business and application continuity, etc. [1]

1.3 Overview

This paper discusses an open HA solution for DB2 UDB, using the Heartbeat package [2] with DRBD [3]. Both Heartbeat and DRBD are open source software released under the GNU Public License (GPL). These packages together can be used to build scalable and highly available cluster applications integrated with DB2 UDB database on the Linux operating system. For details of these packages, refer to section "2.0 DB2 UDB with DRBD and Heartbeat."

The performance of the DB2 UDB / Heartbeat / DRBD combination is examined according to criteria such as performance, availability, and cost-effectiveness. The workload used against the solution being examined is created using a proprietary workload generator toolkit. The workload characteristics generated by this toolkit are similar to those from a TPC-C [4] workload that stresses disk I/O throughput.

For this evaluation, a fairly standard system configuration with limited database and system tuning is used. Some basic performance statistics of running DB2 UDB with DRBD are collected as a measure of performance of the solution. Since Heartbeat enables automatic failover and does not directly impact DB2 UDB runtime performance, the focus of this paper is more on DRBD. However, Heartbeat is used in this setup to ensure that the integrated HA solution consisting of Heartbeat and DRBD is well understood when used with DB2 UDB. Detailed performance metrics for DB2 UDB for Linux with DRBD are not included here; that complex topic is beyond the scope of this paper.

2.0 DB2 UDB with DRBD and Heartbeat

2.1 DB2 UDB

DB2 UDB is the industry's first multimedia, Web-ready relational database management system powerful enough to meet the demands of large corporations and flexible enough to serve medium-sized and small e-businesses. DB2 UDB, as a database management system, delivers a flexible and cost-effective platform on which to build robust on demand business applications. DB2 UDB further leverages system resources with broad support for open standards and popular development platforms such as Java™ 2 Platform, Enterprise Edition (J2EE), Microsoft® .NET, etc. The DB2 product family also includes solutions tailored for specific needs, such as business intelligence, and advanced tooling backed by industry-leading performance figures and reliability.

2.2 DRBD

DRBD is a block device or disk replication technology that can be viewed as network RAID-1. It can be used effectively to mirror a whole block device via a network onto another block device. Thus, DRBD involves two block devices, one labeled the primary (local) and the other labeled secondary (remote/backup/standby). Every write operation to the primary local device is written to disk and also sent to the other host across the network to be written to the secondary device. The remote host (secondary) writes the data to its configured disk. If the primary node fails, then the secondary node can take over in a typical failover scenario.

DRBD currently supports three modes of synchronization with the use of one of the following three protocols:

- Protocol A: Write I/O is reported as completed if it has reached the local disk and local TCP send buffer. [3]
- Protocol B: Write I/O is reported as completed if it has reached the local disk and remote buffer cache. [3]
- Protocol C: Write I/O is reported as completed if it has reached the local and remote disk. [3]

Through specifying one of the above protocols, you can balance performance with the level of data protection required in your environment. Protocol C is used in the setup of this paper because it guarantees full transactional semantics, which is important to DB2 UDB transactions. For more information about the differences among Protocols A, B, and C, refer to the DRBD Web site: http://www.drbd.org.

2.3 Heartbeat

Heartbeat is a fundamental part of the High-Availability Linux project and provides core cluster management services, including membership, communication, resource monitoring and management services, IP address takeover, etc. Heartbeat version 1.2.3 supports multiple IP addresses and a simple two-node primary/secondary model. When used with DB2 UDB in a cluster environment, multiple Heartbeat pairs each consisting of two nodes can be configured to support larger clusters. With the newly released version 2.0.0 of Heartbeat, the two-node size limit of the cluster is lifted. [5]

2.4 System Implementation

In the solution being examined, DRBD can be viewed as a resource that is controlled by Heartbeat. DRBD uses the underlying physical disk as a metadevice. This metadevice is essentially a DRBD resource that is used by all applications to access the disk. When the DRBD resource is made “active” on a particular node, it means that the disk configured as a DRBD resource is now accessible and ready for I/O operations.

As of the current version of DRBD, a disk can be mounted only from the primary node; mounting concurrently in read-only mode from the secondary node is not allowed. This is a limitation by design. If more than one node is concurrently modifying the distributed devices, it becomes very complex to decide which part of the device is up-to-date on which node and what blocks of the device need to be resynchronized in which direction. If the aim is to allow access to the data from multiple nodes concurrently, consider using a shared file system instead.

When using DB2 UDB with DRBD, the failover time related to unmounting the DRBD device from the original primary node and mounting it on the secondary node can be minimized by setting up the instance home directories of the DB2 instance on a local file system on each node. By creating the DB2 instance (and the instance home directory) on the local file system on both the primary and secondary nodes, and by having the DB2 instances already started on each respective node, the DB2 instance startup time can be eliminated in case of a failover.

3.0 System Configuration & Evaluation

The next three sections outline the system configuration used in this paper. For details of installing or configuring Heartbeat and DRBD, refer to [2], [3].

3.1 Physical Overview of System Configuration

The following figure shows the physical overview of the setup used in this evaluation.

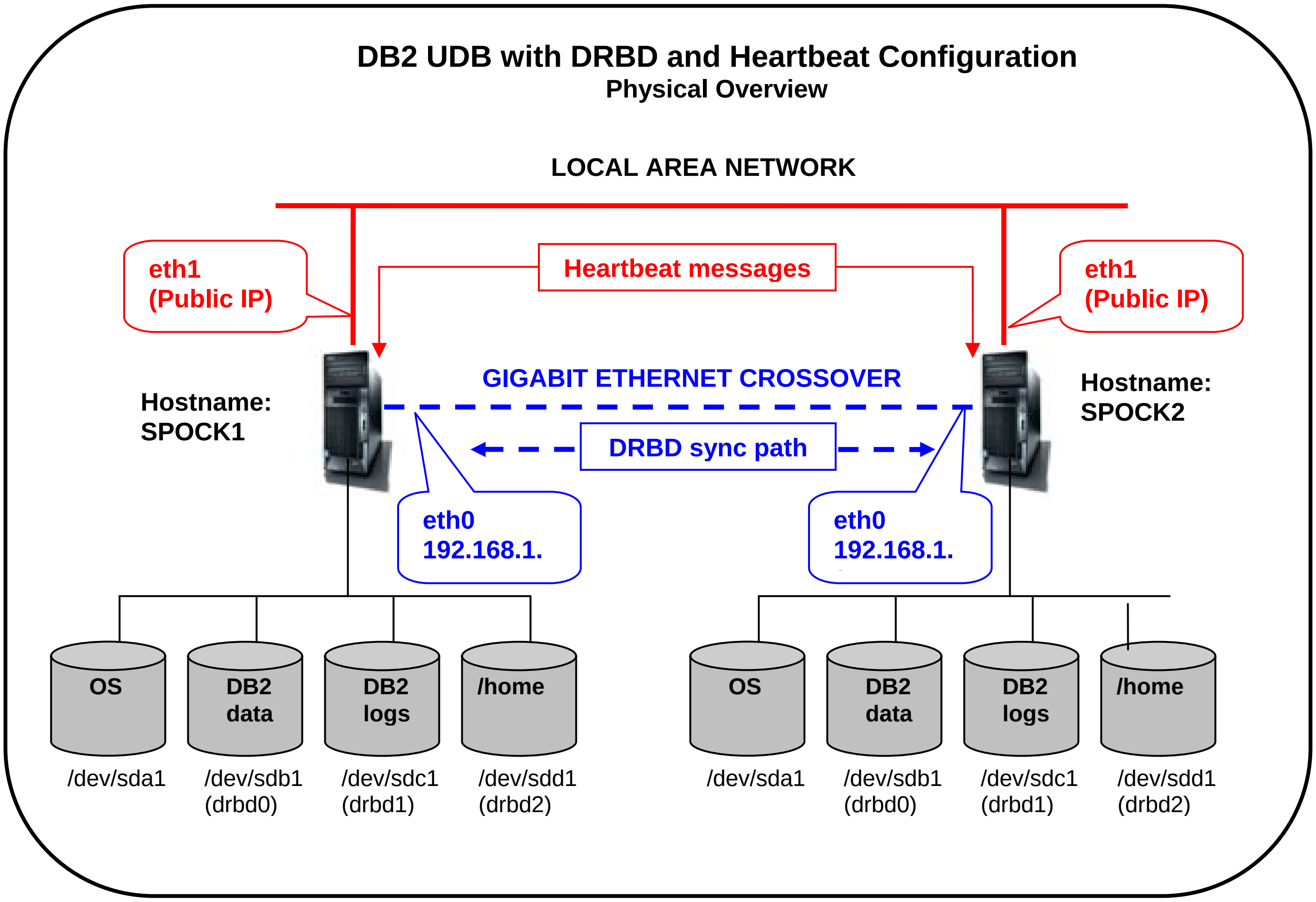


Figure 1. DB2 UDB with DRBD and Heartbeat Configuration: Physical Overview

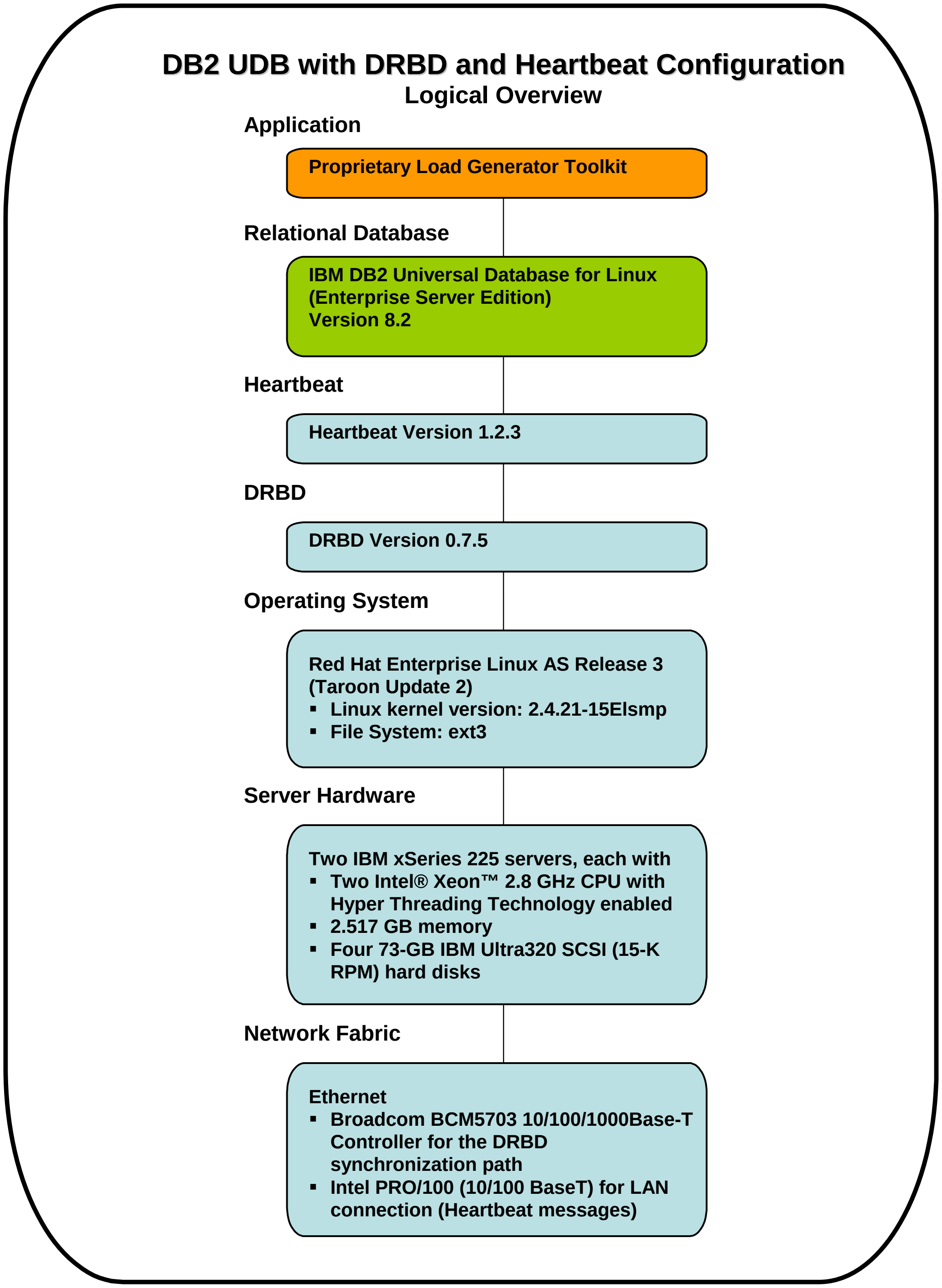
Two IBM eServer™ xSeries® 225 servers (SPOCK1 and SPOCK2) were used in a primary/secondary model. SPOCK1 is the primary node in the cluster and SPOCK2 is the secondary node.

Each server has four local Small Computer System Interface (SCSI) disks: one containing the Linux operating system (/dev/sda1), one for DB2 database data (/dev/sdb1), one for DB2 UDB transaction logs (/dev/sdc1), and one for the home directories of users on the system (/dev/sdd1), including the DB2 instance home directory. The last three disks are set up with DRBD as DRBD devices drbd0, drbd1, and drbd2, respectively. Note that this disk configuration is for test purposes only. In a production environment, many more disks and other storage options and configurations are supported by DB2 UDB.

The two servers have two links of communication between each other. The Heartbeat messages between the two go over a simple Ethernet connection on the public local area network. The DRBD synchronization between the two servers is done via a crossover Gigabit Ethernet in a private network.

3.2 Logical Overview of System Configuration

The following figure gives the logical overview of the setup used.



DB2 UDB with DRBD and Heartbeat Configuration

Logical Overview

Application

Proprietary Load Generator Toolkit

Relational Database

IBM DB2 Universal Database for Linux
(Enterprise Server Edition)
Version 8.2

Heartbeat

Heartbeat Version 1.2.3

DRBD

DRBD Version 0.7.5

Operating System

Red Hat Enterprise Linux AS Release 3
(Taroon Update 2)

- **Linux kernel version: 2.4.21-15Elsmp**
- **File System: ext3**

Server Hardware

Two IBM xSeries 225 servers, each with

- **Two Intel® Xeon™ 2.8 GHz CPU with Hyper Threading Technology enabled**
- **2.517 GB memory**
- **Four 73-GB IBM Ultra320 SCSI (15-K RPM) hard disks**

Network Fabric

Ethernet

- **Broadcom BCM5703 10/100/1000Base-T Controller for the DRBD synchronization path**
- **Intel PRO/100 (10/100 BaseT) for LAN connection (Heartbeat messages)**

Figure 2. DB2 UDB with DRBD and Heartbeat Configuration: Logical Overview

3.3 Test Configuration Parameters

DB2 UDB provides a rich set of tuning parameters that can be configured to obtain optimal performance. Note that the intention here is not to extract the best database performance but to evaluate the use of DRBD/Heartbeat as a Linux disk replication and failover solution for use with DB2 UDB. Hence, better performance tuning for higher throughput numbers (even in the same setup) are beyond the scope of this paper.

The following DB2 database configuration parameters were changed as described below:

Buffer pool size (pages)	*(BUFFPAGE) = 250 000*
Maximum storage for lock list (4 KB)	*(LOCKLIST) = 1000*

3.4 Test Runs & Results

In the following sections, we take a look at the experimental results and the test scenarios:

(i) Throughput Tests

To compare the throughput of the system with and without DRBD, the workload generator was started on the secondary node (spock2) with 5 clients running concurrently for one hour. Statistics were gathered by running the workload eight times on the same configuration. Four runs were done with DRBD and another four runs were done without DRBD. The application throughput is measured by taking the average of the throughput in four runs for the scenario with and without DRBD. The final result is reported as the relative ratio of transactions per second (tps).

	Relative Ratio of Throughput
Without DRBD	1
With DRBD	0.908

(ii) Failover Tests

DB2 UDB and the workload were started on the primary node (spock1). DB2 UDB was failed over to the secondary node (spock2), and the original primary (spock1) became the secondary node. (In other words, a role-switch of the primary and secondary node took place.) The workload continued to run from spock2 with no interruptions.

A secondary node failure test, as described in test (iii b) below, was performed. DRBD was restarted on the secondary (spock1) and DB2 UDB was failed back to spock1. No throughput issues were noticed. DRBD resynchronization time was found to be negligible, as expected, because of its “intelligent” resynchronization mechanism, which synchronized only the blocks that had changed and not the entire disk.

(iii) Other Tests

(a) DRBD communication failure test: DB2 UDB was running on the primary node (spock1); the clients were started on the primary node as well. In the middle of the run, the DRBD communication link between the two nodes was intentionally broken. There were no outages, and only a momentary dip in throughput was observed. The slight dip in throughput was attributed to the wait time for a DRBD timeout when the communication link was broken.

(b) Secondary node failure test: DB2 UDB was running on the primary node (spock1); the clients were started on the primary node as well. In the middle of the run, the power to the secondary node (spock2) was turned off. There were no outages, and no performance degradation occurred. DRBD on the primary node did not report any errors due to the abrupt failure of the secondary node. The secondary node was brought online again, and resynchronization with the primary node was done to ensure mirroring of the two block devices. The resynchronization completed in only a few seconds. (Note that the synchronization speed was changed using the following command to enable faster resynchronization: “drbdsetup /dev/drbd0 syncer –r 100000”.)

4.0 Analysis

This section takes a look at some of the factors important to the understanding of the experimental setup used and issues with respect to using DRBD with DB2 UDB for Linux in production environments.

4.1 Disk and Network Bandwidth

DRBD essentially provides disk replication across a network. Therefore, its performance largely depends on the I/O bandwidth of the physical hard drives used and the network bandwidth between the primary and secondary nodes. The main concerns in such a setup are performance and availability (failover), both of which are guided by the disk and network I/O throughput.

The term *disk write bandwidth* refers to the aggregate write bandwidth of the disks on either the primary or secondary node. In cases where the two machines are identical, the aggregate write bandwidth of each machine is identical as well. However, when the machines are not identical, the aggregate disk write bandwidth would refer to the lesser of the disk write bandwidths of the two machines. With respect to DRBD, it is important to assess whether the network bandwidth between the nodes is sufficiently large when compared to the disk write bandwidth.

4.2 Synchronization Mode

DRBD is used in full-synchronous mode (Protocol C) in this setup. This setup guarantees full transactional semantics, which is important for DB2 UDB transactions. Note that with near-synchronous (Protocol B) and asynchronous (Protocol A) modes, the performance of DB2 UDB with DRBD may further improve. Since consistency issues and data integrity exposures may occur when using Protocol A or B, this paper concentrates on Protocol C. Note that, even when DRBD is used in full-synchronous mode, the resynchronization time required is negligible because of the intelligent re-synch mechanism used by DRBD.

4.3 Performance

Appendix A documents the iostat results for the workloads run both with and without DRBD. Observe that the number of disk writes is higher without DRBD than with DRBD; this finding translates to slightly higher write throughput in the non-DRBD case. This is expected behavior because DRBD has to complete two write operations (one at the local primary node and the other at the secondary node). In addition, DRBD needs to update its metadata (bitmap file) on disk to indicate the disk portions that have changed.

The write performance measurements depend on the bottleneck in the given setup – which could be the disk throughput, the disk controller, the bus interface, or the network bandwidth throughput, among other factors. It is possible that the effects of all such factors have not been isolated. However, that is a tricky and complex issue that is beyond the scope of this paper. The workload used for the tests is highly disk I/O bound, stressing both disk and DB2 UDB throughput. As evident from the iostat results in Appendix A, this is an intense workload from the DB2 UDB perspective for such a simple configuration. Such a load surpasses many real-life workloads, especially for the configuration considered here.

Write performance in DRBD also depends on other DRBD parameters, such as the *sndbuf-size* value, which we have not explored. Performance could also be improved by the use of higher maximum transmission unit (MTU) if the network interface supports it. (These are referred to as jumbo frames.) [6]

4.4 Limitations

In general, data growth in DB2 UDB can be accommodated dynamically by the database system through the addition of table spaces to the database or additional containers to existing table spaces in the database. With respect to DRBD, this data growth means that the database, table spaces, and containers need to be created on a disk for which DRBD is set up. If these database objects are created on new disks or file systems that have not been pre-configured with DRBD, then the database has to be stopped to allow DRBD to be stopped and restarted.

Another potential limitation in such a solution is adding additional hardware, for example, hard disks. However, changes to the hardware typically involve some down time anyway, regardless of the use of DRBD. In addition, adding disks is an infrequent occurrence, especially in an OLTP type system, so this is not a problem.

Finally, as mentioned before, in the current version of DRBD, only the primary node can mount the DRBD device and access it as a resource. The secondary node is not allowed to mount the resource, even in read-only mode.

5.0 Conclusion

The cost effectiveness of a solution based on DRBD / Heartbeat with DB2 UDB for Linux provides a practical high availability alternative for organizations seeking to minimize their capital investment in specialized high availability hardware and software. The results in this paper along with the configuration and setup information clearly indicate that the open Linux-HA solution combines a relatively simple setup with reasonable performance.

Backed by the power and flexibility of the Linux operating system, the IBM DB2 Universal Database for Linux product is the database of choice for any mission-critical applications.

6.0 References

[1] Eaton, Chris and Enzo Cialini, *High Availability Guide for DB2*. IBM Press, 2004.

[2] http://www.linux-ha.org

[3] http://www.drbd.org

[4] http://www.tpc.org/tpcc/

[5] http://www.columbia.edu/acis/networks/advanced/jumbo/jumbo.html

[6] http://linuxha.trick.ca

Appendix A: iostats Results for the Setup Used

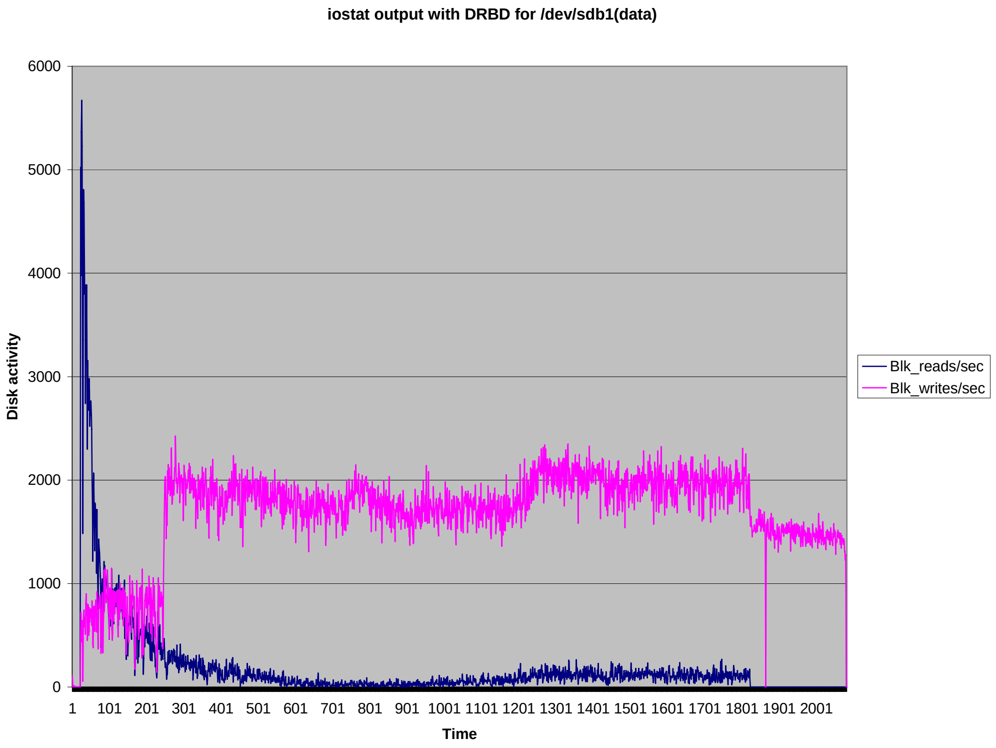


Figure 3. iostat output: DB2 UDB with DRBD (for data files)

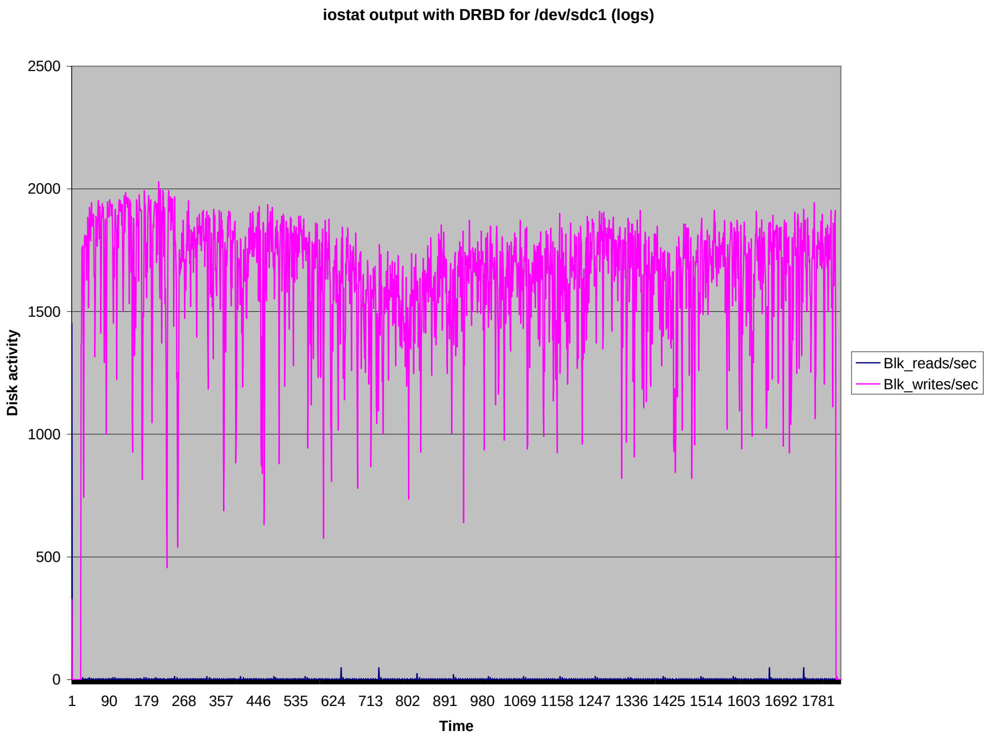


Figure 4. iostat output: DB2 UDB with DRBD (for log files)

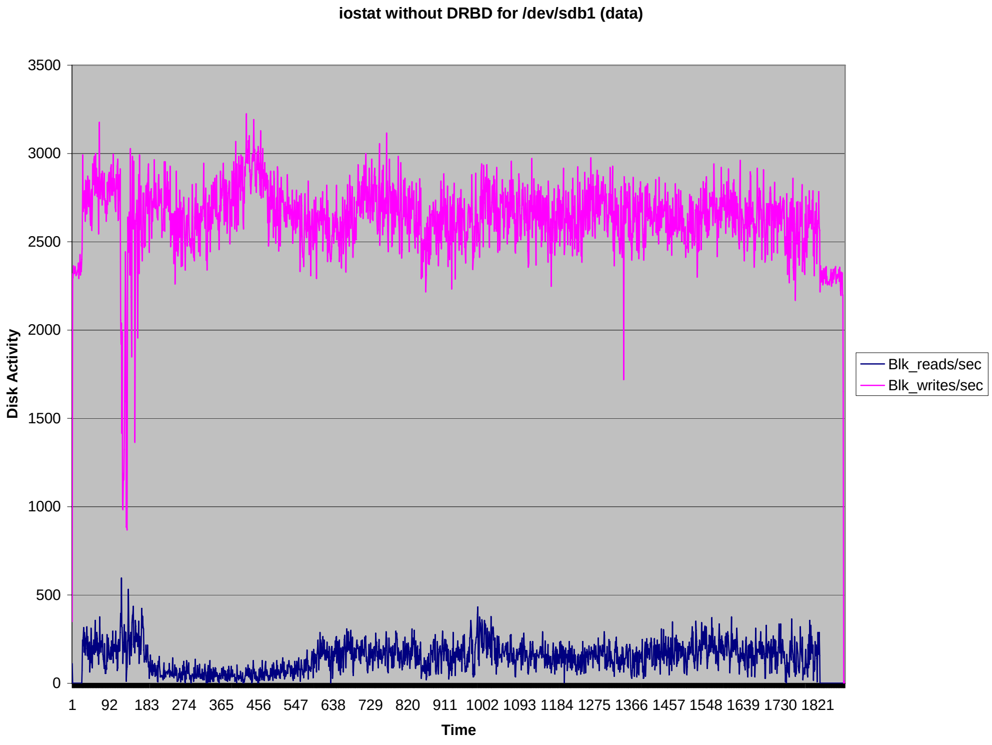


Figure 5. iostat output: DB2 UDB without DRBD (for data files)

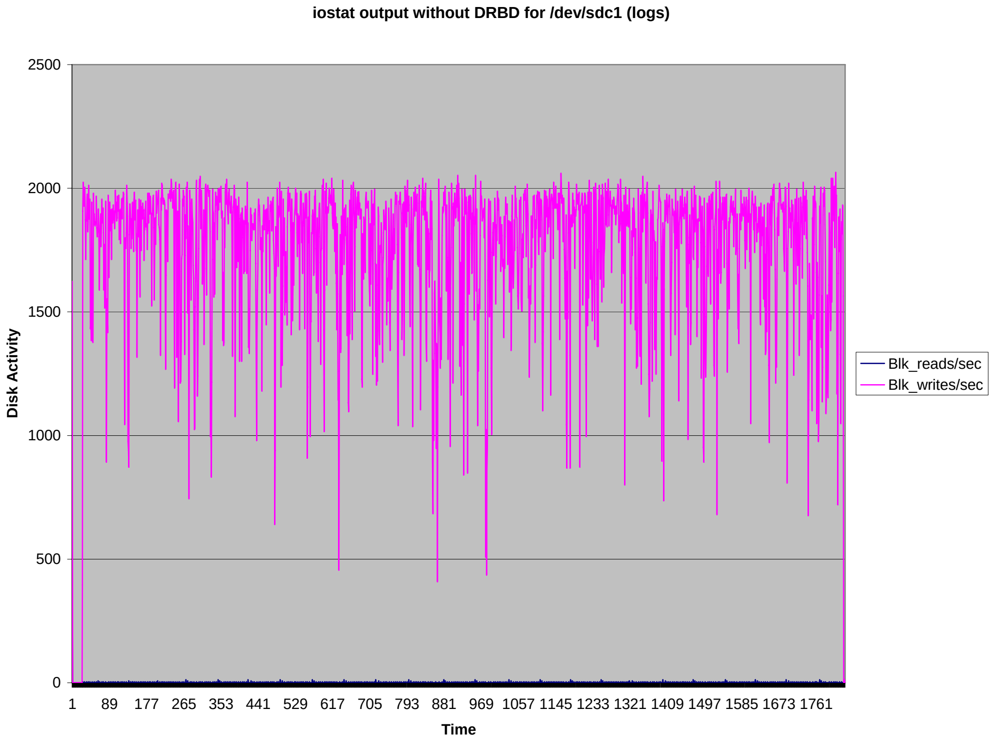


Figure 6. iostat output: DB2 UDB without DRBD (for log files)

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

Printed in United States of America
10-05

IBM makes no warranties or representations with respect to the content hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. IBM assumes no responsibility for any errors that may appear in this document. The information contained in this document is subject to change without any notice. IBM reserves the right to make any such changes without obligation to notify any person of such revision or changes. IBM makes no commitment to keep the information contained herein up to date.

The information in this document concerning non-IBM products was obtained from the supplier(s) of those products. IBM has not tested such products and cannot confirm the accuracy of the performance, compatibility or any other claims related to non-IBM products. Questions about the capabilities of non-IBM products should be addressed to the supplier(s) of those products.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.