High Availability cluster with DRBD

Gregory Steulet . Consultant . 22.10.2008

Nowadays many systems need to be high available. A solution to provide such availability on the storage level could be DRBD (Data Replication Block Device). What kind of usage can fit to DRBD? Could we trust this technology and finally how does DRBD impact the performances? With these questions in mind we deeply analyzed DRBD at Trivadis.

What is DRBD?

DRBD is an open source product made by an innovative Austrian company called LINBIT. This product is designed to build high availability clusters on Linux system. This is achieved by mirroring a whole block device via a (dedicated) network. Basically you could see it as a network RAID 1 (mirroring) over two nodes in the free distribution. The current version of DRBD is 8.2.6 and our tests have been performed on this release.

A second version of DRBD, DRBD+ is commercialised by LINBIT. This version allows to build a two nodes cluster and enables to perform a third, asynchronous mirroring for a high-availability cluster over an unlimited distance.

DRBD supports three kinds of replication mode:

- **Protocol A:** For high latency networks. Write I/O is reported as completed as soon as it reached local disk and local TCP send buffer
- **Protocol B:** For lower-risk scenarios. Write I/O is reported as completed as soon as it reached local disk and remote TCP buffer cache
- **Protocol C:** For most cases, preserves transactional semantics. Write I/O is reported as completed as soon as it reached both local and remote disks.

In our tests only protocol C has been considered as it is by far the most used and most secure.

Possible applications

DRBD provides two basics kinds of synchronous replication, master-slave (active-passive) and multi-master (active-active). In the active-passive case operations are only allowed on the master because the slave device is unmounted. This mode guarantees that only one cluster node manipulates the data at a given point in time.

Several kinds of application can benefit from a DBRD architecture such as a web server, ftp server or even a database server. A server running DRBD in active-passive mode could be used as a failover cluster.

Since version 8.0, DRBD provides the active-active configuration mode which could be used within a load-balanced cluster (both nodes of the cluster are in use). However this mode requires the use of a shared cluster file system such as GFS (Global File System) or OCFS2 (Oracle Cluster File System) as it is possible to have concurrent accesses against the same data.

DRBD has no load balancing functionality, to benefit from the active active mode, the use of a load balancer or a cluster aware application is mandatory.

High-availability

As we speak about high-availability we also speak about downtimes. There are two major kinds of downtime; planned downtime and unplanned downtime. Unplanned downtime can have several causes, like power outage, human error, data corruption, software or hardware errors, etc... The planned downtime can be due to maintenance or routine operations such as upgrade, space management or system reconfiguration. In this section we focus on how DRBD handles maintenance operations and split brain situations, those are two causes of downtime in the DRBD field.

Note that for general information about High Availability, each one is free to read the first pages of the Trivadis white paper about this topic:
www.trivadis.com/uploads/tx_cabagdownloadarea/Trivadis_HA_white_paper_release_2_2_final.pdf

Maintenance operations

Before having a look on some maintenance operations it could be necessary to know how to monitor the current status. For that you can use one of the following commands:

- cat /proc/drbd
- /etc/init.d/drbd status
- drbdadm cstate/state/dstate
- cat /var/log/messages | grep drbd

```
a10:/etc/init.d # ./drbd status
m:res      cs          st               ds                 p  mounted  fstype
1:rVot01   Connected   Primary/Primary  UpToDate/UpToDate  C  /u101    ocfs2
2:rVot02   Connected   Primary/Primary  UpToDate/UpToDate  C  /u102    ocfs2
```

This output shows two resources that are both connected (cs column), the configuration mode is active-active (st column) and data are up to date on both nodes (ds column). In addition this status provides the mount points (/u101,/u102) and the used file system (ocfs2).

DRBD allows to achieve a lot of maintenance operations without impacting the system availability. For example adding resources can be done online by following the procedure below:

1. Fdisk, partprobe your disks (max. 15 logical partitions on Linux) on both hosts
2. Create the Logical Volumes on both hosts
3. Create the appropriate entries in the /etc/drbd.conf file on both hosts
4. Reload your DRBD config, you might use the --dry-run first
5. Start the new DRBD device on both hosts
6. Force one site to be primary, and then the resynchronisation starts
7. Set the second site to primary as well, create the FS and mount them. Ready!

DRBD also supports online resize of a resource. However before growing or shrinking a file system online three criterias must be fulfilled:

1. The resource must be on logical volume management subsystem, such as LVM or EVMS
2. The resource must currently be in the “Connected” connection state
3. The file system must support online growing or shrinking

Even rolling upgrade is possible in a primary-secondary configuration mode by upgrading the secondary node first. A DRBD rolling upgrade implies to:

1. Putting the standby server offline
2. Upgrading DRBD on the standby node
3. Bringing the standby server online
4. Doing the failover
5. Upgrading the former active

Finally DRBD allows to reconfigure resources while they are up and running by making the necessary changes to the /etc/drbd.conf file and issuing the “drbdadm adjust all” command on both nodes.

Split Brain

Split brain is a term coming from medical field and describes a state where the link between both hemispheres of the brain is severed. In computer science the cluster split brain is based on this definition. However DRBD split brain definition slightly differs. In DRBD, a split brain is (according to LINBIT) "*a situation where, due to temporary failure of all network links between cluster nodes, and possibly due to intervention by a cluster management software or human error, both nodes switched to the primary role while disconnected*". To summarize, the split brain is the fact of having two primary resources while the network between these resources failed. Loss of connectivity between hosts in DRBD is referred to as a cluster partition. For convenience and to avoid confusion the term split brain in this article refers to the DRBD split brain definition.

Although a split brain situation occurs when there is no more connection between two primary resources it's also possible to get a split brain situation in an active-passive configuration. If the connection between both servers is down, it is still possible to switch a resource from passive to active, and therefore to obtain a DRBD split brain. DRBD detects this situation as the connection between hosts becomes available again. As soon as the split brain has been detected DRBD stops the replication.

```
Feb 10 17:01:53 node1 kernel: drbd0: Split-Brain detected, dropping
connection!
```

You have two ways to manage such a situation: manually and automatically. Manually by discarding data on one node or automatically by using one of the algorithms provided by DRBD. The possible algorithms will depend on the configuration (0,1 or 2 primary) when the split brain is detected. In addition the choice of an algorithm will strongly depend on the applications which are running over DRBD. Among the possible algorithms there is

1. Discarding modifications made on the younger primary
2. Discarding modifications made on the “older” primary
3. Discarding modifications on the primary with fewer changes
4. Graceful recovery from split brain if one host has had no intermediate changes
5. Discarding modifications on the current secondary

Therefore it is possible to manage the “after split-brain” situation, but what about the possibilities of avoiding it? Dopd (DRBD outdate-peer deamon) in conjunction with heartbeat provides a solution by outdating the resources on the secondary node through another network path. An outdated resource cannot become primary anymore as long as the connection is not re-established and as long as some maintenance operations have not been performed. In order to enable dopd you will need to append on both nodes the following lines in your /etc/ha.d/ha.cf

```
respawn hacluster /usr/lib/heartbeat/dopd
apiauth dopd gid=haclient uid=hacluster
```

Reloading heartbeat and modifying drbd.conf as below

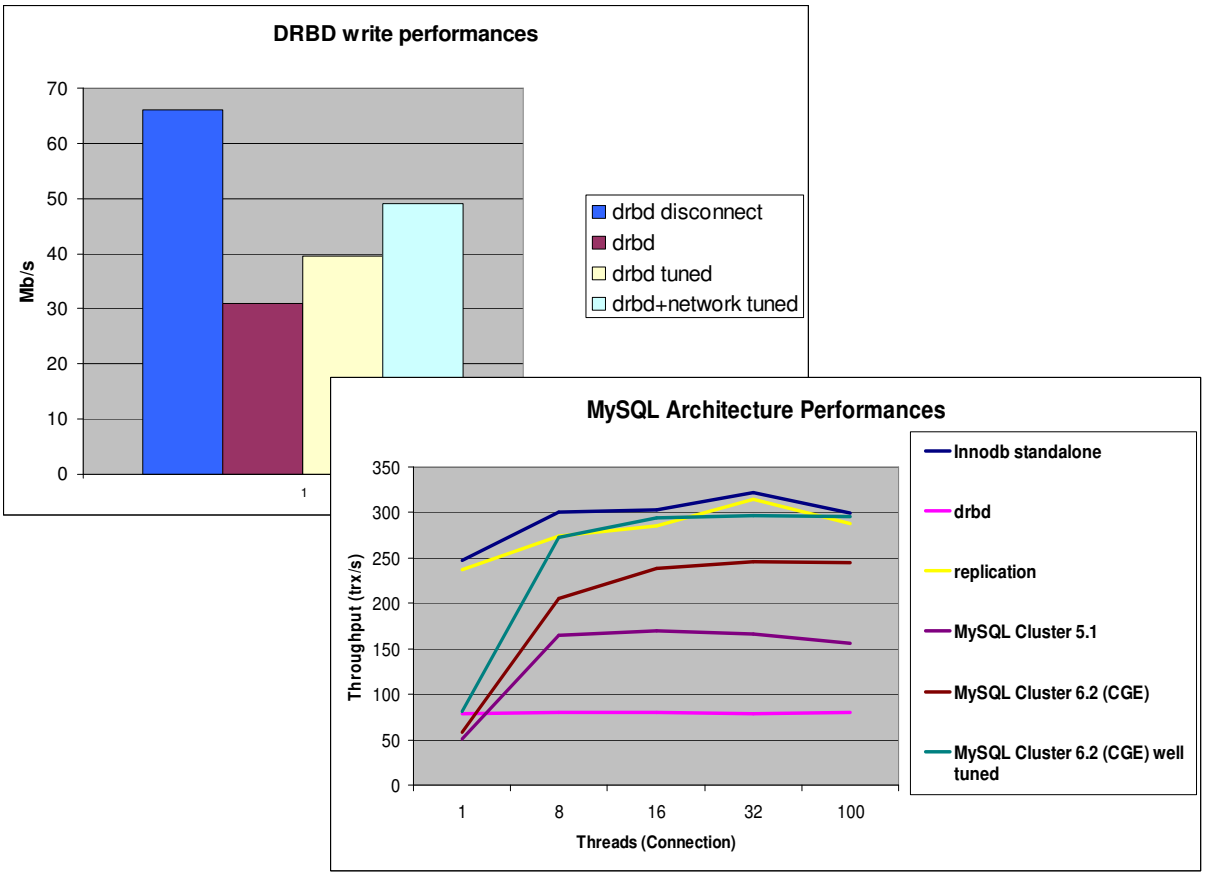
```
resource rDrbd01 {
handlers {
      outdate-peer "/usr/lib/heartbeat/drbd-peer-outdater"; }
disk { fencing resource-only; }
...
}
```

Modifications will take effect after restarting or adjusting your drbd configuration

```
drbdadm adjust rDrbd01
```

Performances

Depending on which protocol is used A, B or C the performances could be more or less impacted. All Read operations are serviced from local hard drive even in active-active mode, however writes suffer from DRBD overhead. In our tests we used protocol C. Two kinds of tests have been made, a pure write test with “dd” and a second with MySQL and SysBench. Of course these results fully depend on the architecture and on the software but the proportions should be kept in a “standard” architecture.



The first graph (DRBD write performances) shows how DRBD in a primary/secondary mode could impact write performances if you do not pay attention to the settings (difference between blue and purple column). After spending some time on DRBD and kernel settings a significant improvement can be obtained (difference between blue and light blue column).

The second graph (MySQL Architecture Performances) compares several high-availability solutions for MySQL. The highest throughput is done with MySQL 5.1 with innodb as storage engine. Regarding MySQL Cluster, tests have been done with three nodes architecture including two NDB nodes with dedicated network. DRBD in a primary secondary mode is stuck at about 80 transactions/second whatever the number of connection. Those poor results are due to the fact that MySQL 5.1 do lot of serialized synchronous writes.

The major problem with write performances in DRBD are mostly due to disk and network latency, so if you plan to use DRBD and you have some important write performance prerequisites you should have a look on Battery Backed Write Cache (BBWC) controller and Infiniband or Dolphin interconnect. Dolphin interconnect and LINBIT announced there partnership on September 12th. Dolphin Express low latency protocol will be supported with version 8.2.7 which should be available in a few weeks.

How could I tune DRBD?

There is no silver bullet that allows a significant improvement in every kind of situation. However there is a range of DRBD parameters that can affect writes performances like:

- **Activity Log size** (al-extents)
- **Unplug watermark** (unplug-watermark)
- **Maximum I/O request buffers allocated** (max-buffer)

It's also possible to change some network kernel parameters such as rmem_default, rmem_max, wmem_default, wmem_max, tcp_mem, tcp_wmem, tcp_rmem. Anyway before changing those parameters have a look on the application requirements and make sure you gathered the required know-how.

Conclusion

DRBD provides a reliable and stable host based mirror solution. Installation as well as configuration is a straightforward process, even if you have to know what you do of course. In addition this solution if free in a two nodes configuration and MySQL supports it officially.

On the other hand, DRBD is only working on Linux. The other operating systems sell their own solution. For the moment SuSE Linux Enterprise supports only version 0.7 and there is no support provided by Red Hat. As told in the performance section you could encounter some serious write performance degradation if your application is strongly write oriented.

Finally, even if it sounds obvious, please perform a lot of tests before going in production, This is anyway a best practice in each high availability project. In order to know if this solution fits well to your requirements in terms of availability and performances, testing is mandatory. In addition do not be fooled there is no product that can give high-availability. The high-availability is a concept, lot of best practices, good habits before a product.

Trivadis supports several customers for High Availability architecture. As for any High Available project (MySQL replication MySQL Cluster, Oracle Dataguard or RAC, Vertas Storage Fundation,…) the complexity of a cluster architecture shouldn't be underestimated. Several steps are very important in such a project: concept, documentation, testing/validating, and so on…

Thanks to William Sescu for his active participation and contribution in this paper, to Michael Wirz for his contribution in the project and Yann Neuhaus for his expertise in the field of high-availability and contribution in this paper. Without them this article wouldn't have been possible.

Gregory Steulet
Oracle Certified Professional 10G
MySQL Cluster 5.1 Certified
Trivadis SA
Rue Marterey 5 Tel: +41-21-321 47 00
CH-1005 Lausanne Fax: +41-21-321 47 01
Internet: www.trivadis.com Mail: info@trivadis.com

Literatur und Links…

http://www.drbd.org/users-guide/users-guide.html
http://blogs.linbit.com/florian
http://www.mysql.com/products/enterprise/drbd.html
http://www.mysqlperformanceblog.com
http://sysbench.sourceforge.net/
http://www.mysql.org