Reprinted from the

Proceedings of the Linux Symposium

July 23th–26th, 2003
Ottawa, Ontario
Canada

Reliable NAS from Dirt Cheap Commodity Hardware

Benjamin C.R. LaHaise
`bcrl@kvack.org`

Abstract

To many of us, the integrity of our data is paramount. Unfortunately, the most cost effective commodity hardware available tends to omit important features like ECC memory, or only provides it at a substantial cost premium. To address this, an approach that makes use of concepts taken from NAS and clustering, combined with a novel trick to obtain a CRC on data blocks from existing ethernet NICs is used. The resulting code is built on top of the existing Linux Network Block Device, network drivers and async io. Details ranging from design considerations to performance tuning and implementation provide an interesting story on how to attain a much easier, and cheaper, reliable storage solution.

1 Introduction

Anyone who uses computers can tell you that they will inevitably break. From software bugs, to hardware errata, or just plain old age, the components of systems will always fail at some point in their deployed lifespan. Hardware vendors try to address various subsets of these failures, but frequently only in high end systems. Even in the case of hardware specifically designed to cope with a common mode of failure, RAID controllers are lost when a failed SCSI device locks the bus.

Over the last few years, low end hardware has come to provide capabilities well beyond the needs of many server systems. Relatively cheap IDE drives cost less than a third the amount for a comperable SCSI device. As an end user, that difference in the price performance ratio raises more than a few eyebrows. Several companies sell products into this niche, yet there are a number of failure modes which their devices fail to address. Data corruption is still possible when disks make use of non-error correcting memory, or firmware bugs exist (what body of software is bug-free?). So what is a user that wishes to make use of the price point of commodity hardware to do if their data is truely valuable?

Take one kernel hacker, random bits of hardware, several discussions over beer at conferences, this problem, and mix. The result is `netmd.o`.

2 What is Netmd?

Netmd at its core is a hybrid between the Linux RAID implementation (known as md) and the network block device (known as nbd), and is quite similar to DRBD. DRBD allows users to mirror disks over a network, yet it runs under the assumption that the underlying hardware is reliable and will detect any errors which corrupt user data. Netmd changes this by making use of an end to end CRC on sectors to check if any of the BEs have unknowingly corrupted read or write data. All told, netmd is much easier to use than either drbd or nbd while providing guarantees about data integrity.

3 Design Criteria

The front end system runs the netmd kernel module. To provide the greatest likelyhood of the software being correct, simplicity is strong goal for the resulting code.

suffered from a simple single bit error, or more gross errors resulting in so called fractured blocks (ie, the case where the sector number itself is corrupt and the wrong data is read back from disk). The basic assumption netmd makes is that your front end hardware is reliable and the bulk of errors will come from the back end systems which house all of the disks. A front end system running NetMD looks very much like an NBD, but provides reliability guarantees.

The simplest configuration for NetMD (see Figure 3) consists of a front end system and three back end systems networked via a switch or hub. A minimum of three back ends are required for NetMD to establish quorum and provide the ability to hot swap one back end at a time. Additional BEs can be added to improve the performance and reliability of a NetMD setup (such as in Figure 3).

Unlike nbd, which operates over TCP, NetMD instead operates over UDP to take advantage of the packet nature of the underlying network. This allows writes to be optimized using multicast to deliver data to all BEs simultaneously. BEs may also snoop read requests to rapidly reply if the data is still available in its local cache.

In fact, the main data integrity feature of NetMD comes from the ethernet packets which are used to transmit read and write requests over the network. All ethernet frames are protected from transmission errors by a trailing 32 bit CRC. Many ethernet cards are able to record the CRC of an incoming packet. By compensating for the header of each packet, the data CRC can be extracted at low CPU cost.