Linux Based Disaster Recovery Solutions

Eurosec'2006
Session S7A

James E.J. Bottomley
SteelEye Technology, Inc.
James.Bottomley@steeleye.com

3 April 2006

Abstract

The value of preserving data against potential disasters via the use of backup tapes (and their transport to an off-site location) has been well understood for a long time. However, the basic problem with such a plan is that even though it preserves valuable (and often irreplaceable) data, it doesn't necessarily encompass a method for getting your business back on track after a disaster. Unfortunately, the usual highly available solution of having a duplicate installation elsewhere and bringing up your services there in the event of a disaster at the primary site was beyond the budget of most small and medium businesses. Now, however, with the advent of managed hosting and other ISP services, it is possible to negotiate a disaster recovery site for a pay as you go premium which is within budget. To take advantage of this, all SMBs have to do is to select a method of getting their data from their primary server to the ISP backup site.

1 Introduction

In order to understand exactly what Disaster Recovery Solution you need, you must first understand what "Disaster Recovery" actually means. The term has a variety of uses, from a simple tape backup regimen all the way up to a continuous replicating disaster availability system. This can only be defined if you have an understanding of the disaster tolerance criteria for your business, which is loosely defined as how much data are you willing to lose, what is the maximum time before your business needs to be up and running and how much are you prepared to pay to achieve this. Since the cost of disaster recovery goes up pretty fast as the first two decline, cost usually ends up being the gating item.

2 Sizing Network Pipes

In order to achieve a continuously replicating disaster recovery solution, you need a network link between two separate sites over which the data can be replicated. Obtaining a dedicated net-

work pipe for this purpose usually ends up being the single most expensive (and recurring) item in the Disaster Recovery budget. Since network bandwidth is so expensive, it is vital to understand what your current (and future) requirements for transmission are. One of the cardinal pitfalls is to choose a pipe, for cost reasons, that is actually to narrow for your continuous replication needs. Therefore, it is vital to run an assessment within your enterprise of which of your applications need continuous backup and what the characteristics of the data turn over are for them. For most actual applications, this data can only be reliably obtained by setting up a monitor within the application itself. Although it can, theoretically, also be obtained from simulations, the observation from the field is that simulations rarely account for everything and often underestimate the data turn over figures.

3 Replication Characteristics

In every network replication scenario, in order to pack as much data into the pipe as possible (and therefore utilise it up to a realistic bandwidth maximum, like 90%) it is required that the replication mode be asynchronous (this means that the rate at which data is stuffed into the pipe is decoupled from any application requirements about data safety (the only guarantee is ordering, meaning that if the aplication goes down, there will be in-flight data, acknowledged as completed to the aplication, which does not make it safely to the replica volume).

Additionall, there is a risk that the actual network connection itself will fail. In that case, the changes made to the local volume but not transmitted to the replica must be logged for later transmission when the network connection is restored. In the common implementations, there are two types of logging: transaction and intent, each with their own advantages and pitfalls. Since the only available type of logging in the linux open source solutions is intent, we shall only describe the operation of this type.

An intent log is basically a simple bitmap, one for each chunk (chunks may be any size, but usually they are kept at around 4-256kB) a bit set to one indicates the corresponding chunk to be empty, and must be replayed in its entirety, and a zero usually indicates clean (no replay necessary). The advantage of a bitmap is that it's usually small (a tiny fraction of the space used to store the data) and the primary disadvantage is that it has no ordering information, so on a bitmap replay the remote replica doesn't contain a viable copy of the data until the replay is entirely completed.

4 Linux Replication Solutions

Once you have a network pipe, there are a variety of replication solutions to choose from. The ones we shall focus on are the open source solutions, but there are also one or two closed source solutions from various vendors.

4.1 `md/nbd`

This implementation, which is sponsored by SteelEye Technology, is very much in the spirit of open source component re-use. Simply put, it creates a replication system by utilising two readily available components within Linux: `md` which is the Multiple Device subsystem to create a RAID-1 (mirror) and `nbd` the Network Block Device to make one of the legs of the mirror network remote from the system. The advantage of this approach is that mirroring technology in

an operating system kernel is complex and hard to get right, so reusing existing, working and tested components to create a network mirror is a great benefit in ensuring reliability. Additionally, `md/nbd` is the only open source solution to be incorporated into the Linux Kernel itself[1]

4.2 `drbd`

This is the Distributed Replicating Block Device. It is a completely self contained replication solution (i.e. written from scratch, not starting with existing components). It has almost identical capabilities to `md/nbd`; however, it is not currently part of the Linux Kernel (it is available as a separate, open source, add on module). It is currently supported by SUSE (and is thus always found in the SLES Distribution).

5 Conclusions

Enterprise Level replication today is very viable with Linux and can be built with completely open sourced components. However, unless you really know what you are doing, there are a large number of implementation subtleties which can be difficult to get right (and expensive if you get them wrong). Thus, it is often better, at least for the initial deployment, to engage with a company whose subject matter expertise is in Disaster Recovery and replication.

[1] This is as of kernel version 2.6.14, when the last pieces of asynchronous mirroring and intent logging went into the `md` subsystem