16.3. What Is High Availability?

High Availability (**HA**) encompasses the components and configuration required to keep an application functioning, even in the event of a failure. **HA** can be expanded to deal with the failure of an application, an operating system, a computer, a data center, or even a city. The major components usually involved in making a system highly available include:

- Storage.
- Network connections.
- Application monitoring.
- Resource monitoring.

High Availability is based on the principle of no single point of failure. The goal is continuous operation. Each component is examined for failure points and each are replicated. A control or monitoring program is used to verify the status of all of the component parts.

16.4. HA Stack

The traditional view of **High Availability** focuses on the duplication of resources: the introduction of a second or backup system to take over processing in the event of a failure on the primary system.

Often times, the focus is directed to the control and monitor application, as it is a key component. There are several applications:

- **Pacemaker.**
- **Heartbeat.**
- **Corosync.**
- **Red Hat Cluster Suite.**
- **Veritas Cluster Server.**

Regardless of the control program used, there are some fundamentals to consider.

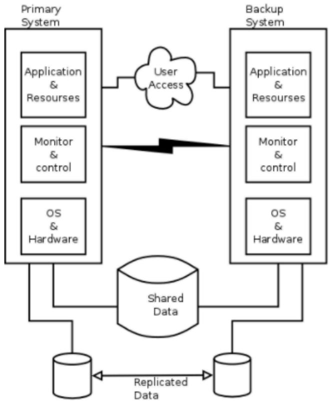

Figure 16.1: HA Stack

16.5. Applications

Applications are the reason we have the computer in the first place. If an application fails, it can be restarted on the same node or on a different node.

When an application fails, it may trigger a series of events. To switch an application to another computer (node), all necessary resources (**OS**, **Storage**, **Network**) must be in place to support the application.

Resource requirements include:

- Application installed on a second node.
- Storage required for application to be available.
- Network connection to be available.

16.6. Networking

Networking is a critical component of **High Availability**. Most services are accessed by a network.

Normally, an **IP** address is associated with a single network adapter; this is a single point of failure. It may be possible to use bonding or teaming to create an adapter that has multiple hardware components.

If the failure is further up the network stack, it will have to be abandoned and another network stack put into service.

Switching Network stacks usually means switching to another computer and taking the **IP** address to the new system. If that occurs, it is most likely that more components will have to be moved.

16.7. Storage

Most filesystems and storage devices are not multi-system access. The storage may be physically connected to more than one system at a time, but care must be taken so that the filesystems are not accessed by two (or more) systems at the same time.

There are some options for multi-system storage. If remote mirroring is considered, there will be two copies of the filesystem on different nodes. Access to the storage must still be controlled.

When switching storage to another system, some storage devices are multi-connected (**SAN** or **iscsi**).

Network-based filesystems have the advantage of being able to be accessed (read and write) by more than one node. The challenge with network-based storage is that it is generally slower than locally-attached storage.

16.8. Remote Disk Mirroring (DRBD)

One example of remote disk mirroring or remote block device is the **DRBD** project.

The **DRBD** driver code was added to the **Linux** kernel as of 2.6.33. The **DRBD** module intercepts the local writes heading for a storage device and mirrors them to a remote location via **TCP/IP**.

With identical copies of the filesystem, the application can restart on the remote mirror storage.

More information is available at: http://drbd.linbit.com.

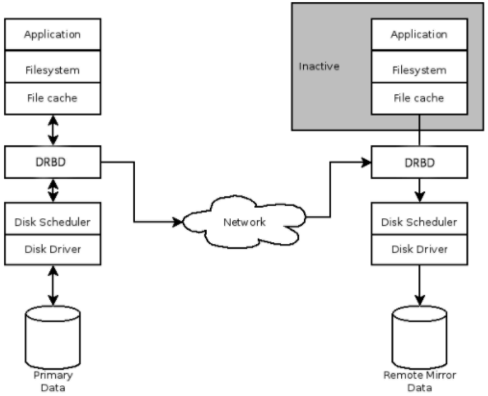

Figure 16.2: DRBD