Redbooks Paper

Erich Amrehn
Joachim Jordan
Frank Kirschner
Bill Reeder

# Server Consolidation with Linux for zSeries

## Introduction

This paper provides an introduction to server consolidation with Linux for zSeries. It covers the following areas:

- The different types of consolidations, their pros and cons.
- Technical example of a server farm under z/VM, using sharing and cloning techniques.
- How to map a UNIX or Intel server farm to a Linux under z/VM environment.
- Example farms: File serving with Samba and Web serving with Apache.
- A brief introduction to TCO.

## Consolidation types

The main drive behind consolidation is to lower the total cost of ownership (TCO). There are many aspects of the TCO that can justify server consolidation. Server consolidation can:

- Increase utilization of server hardware, software, and networks.
- Lower total costs including employee cost, floor space, and energy.
- Increase availability, lower downtime costs.
- Lower overall cost by consolidation of test and development servers.
- Permit faster deployment of new servers.

While it is possible to quantify the benefits of a server consolidation with a TCO calculation, in many cases this may not be necessary. For example, if floor space is a limiting factor, and thus further growth will require constructing a completely new building, the cost savings of just this single factor can be intuitively sufficient to justify a server consolidation.

## The different types of consolidation

The most common point of view on consolidation is to look at it from an application perspective. Using this approach, the three basic types of consolidation can be distinguished as: centralization, physical consolidation, and application integration. An overview of the three types follows.

### Centralization

Servers, distributed over several different cities or buildings, are consolidated at one or a few locations. Personnel can be consolidated and work done more efficiently, raising the ratio of the number of servers per employee. Servers dependent on a local resource, like fax servers (which have to be connected to the telephone line in the location) generally cannot be consolidated. Many servers were distributed (that is, in different locations) in the past to save network bandwidth to the centralized computer centers, and also because of availability problems of the network connections to the computer centers. With servers in the distributed locations, users could still work even if the network connections to the centralized computers were down. Now, however, network bandwidth is much more reliable and getting cheaper every year. Therefore, consolidation of these servers can be feasible today. Figure 1 illustrates the concept of centralization.

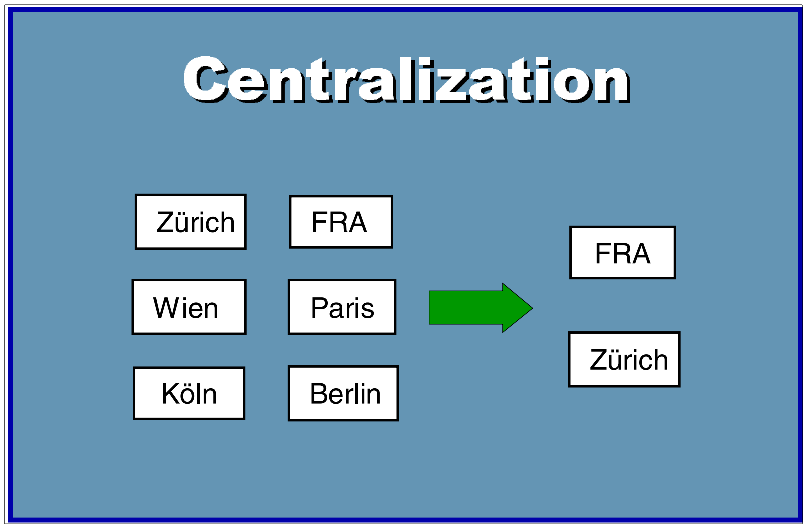

*Figure 1 Centralization*

### Physical consolidation

Servers running the same applications can be consolidated by merging them together on one server. File servers are a good example of this: the users and their directories can be easily copied from several servers to one server.

Physical consolidation works best when the servers being consolidated are generally of low utilization. (Typically less than 5% utilization is seen on Intel, and less than 13% on RISC systems. See the Scorpion whitepaper for details.) Another factor that supports physical consolidation is if the candidate servers have not been very recently updated to newer, faster machines. Savings will be seen in the areas of lower operations cost, people costs, and improved disaster recovery.

If the hardware being replaced is new and fast hardware, one alternative is to redeploy it for some other use. Reducing the number of systems and application installations from many to one can significantly lower the expense and effort of administration and maintenance. In addition, depending on the software license models, consolidation can also save software licence fees. Figure 2 illustrates the concept of physical consolidation.

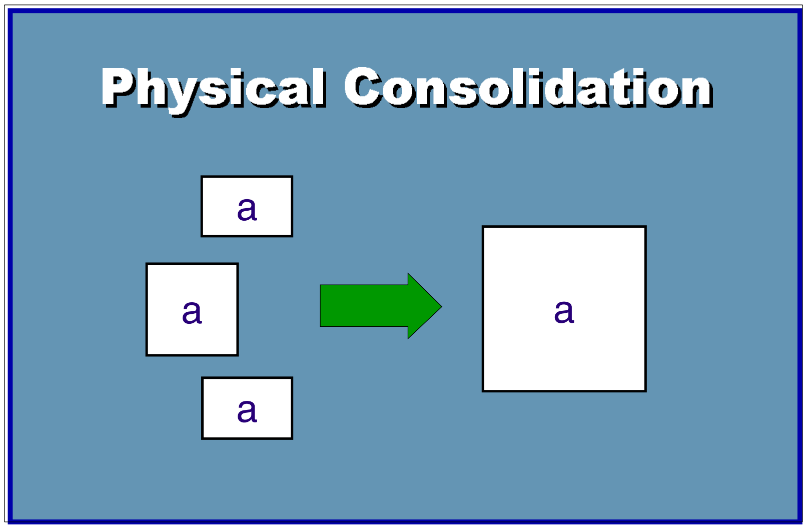

*Figure 2 Physical consolidation*

## Application integration

This is a consolidation of different servers running different applications to one server. Complying with the UNIX paradigm, which states: “one application per server,” an Internet presence with Web, ftp, e-mail, news, and DNS services would have to run on 5 servers. If these servers are consolidated, the services will then run on one server. This is illustrated in Figure 3.

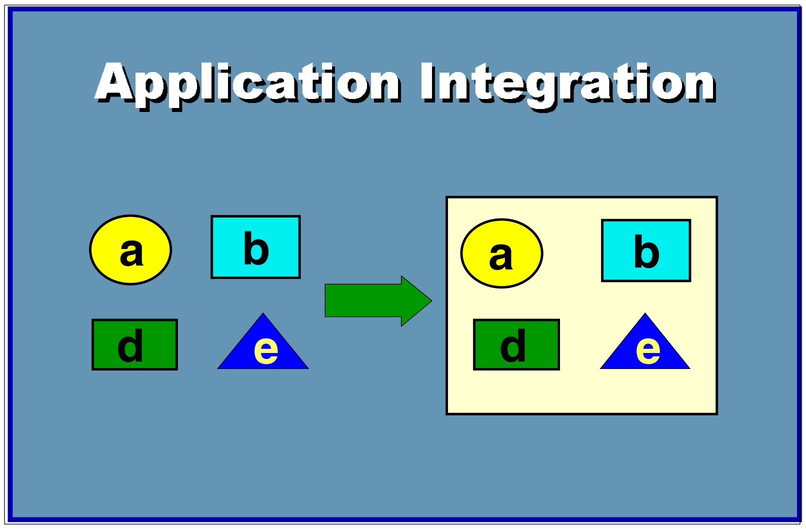

*Figure 3 Application integration*

One consideration with respect to application integration is that isolation of the different workloads, both for security and workload balancing purposes, has to be ensured. Without good separation among the applications, a software bug in one of the services could cause the whole server to crash, bringing down all the other services, too.

Application consolidation can achieve savings similar to those realized by physical consolidation. On the other hand, savings generated by lowering the number of operation system images may be lost by expenses incurred to establish the necessarily complex setup for isolation.

## Consolidation from an infrastructure perspective

Looking at the consolidation scenarios from an infrastructure perspective, the issues related to physical and application consolidation are somewhat different. Taking an infrastructure approach, the two scenarios are not distinguished by the type of applications consolidated, but instead by the operation environment requirements. The scenarios of physical or application consolidation can be refined to either full or virtual consolidation, as described in the following sections.

### Full consolidation (n-to-1)

With full consolidation, several servers, running either the same or different applications, are consolidated on one server and one operating system image.

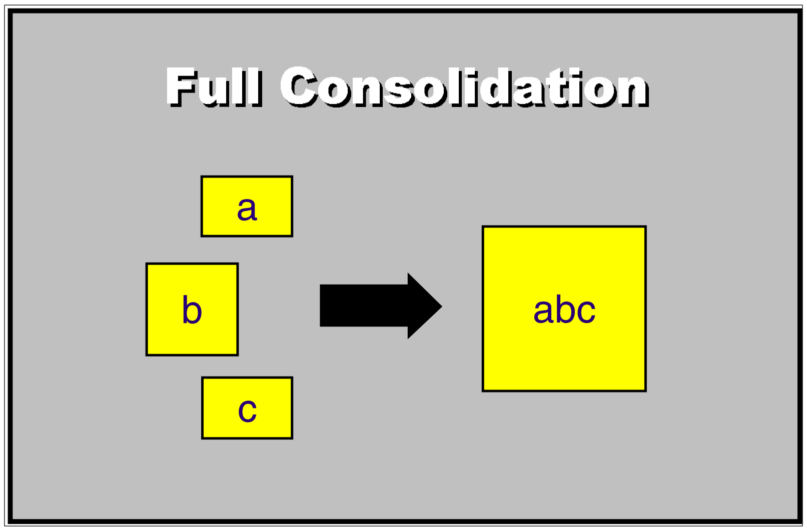

*Figure 4 Full consolidation*

There are several reasons why a full consolidation may be difficult, or not feasible at all:

- Security: Depending on the applications and the functionality of the operation system, a strong enough security separation may only be possible with separate servers.
- Stability: Some of the applications may not run very well together on one system. For the same application on the servers, this could be because the software does not support several instances running on the same system. In addition, there is the danger of one application using up all the system resources or crashing the operating system, and by that, disabling all the other applications, too.
- Maintenance: While merging servers and reducing the number of operating systems in a full consolidation lowers the maintenance effort, different service levels and versions of the application software may require applications to run on separate servers.
- Flexibility: Applications will change. This means that new versions of applications will be used, which may require a different operating system environment, additional or changed third-party software, more resources or higher service levels. Furthermore, applications will be removed or replaced over time. This is not a problem for a distributed server environment, but it requires good planning and thorough testing to ensure the quality of the solutions in a fully consolidated environment.

- Support: Many software vendors are very specific about support on applications and platforms. As a result, the customer may risk losing support of any kind from the software vendor because of the necessary changes the customer makes to implement a consolidated environment.

**Important:** For these reasons, for many servers the best approach will be to do a *virtual* consolidation with Linux for zSeries.

### Virtual consolidation (n-to-n)

In virtual consolidation, servers are consolidated into the same number of operating system images running on one box by using the virtualization technology of z/VM on zSeries. z/VM virtualization technology allows sharing the available hardware between the guest systems, leading to better utilization of the hardware. Resource allocation is secure, transparent, and fully dynamic. This is illustrated in Figure 5.

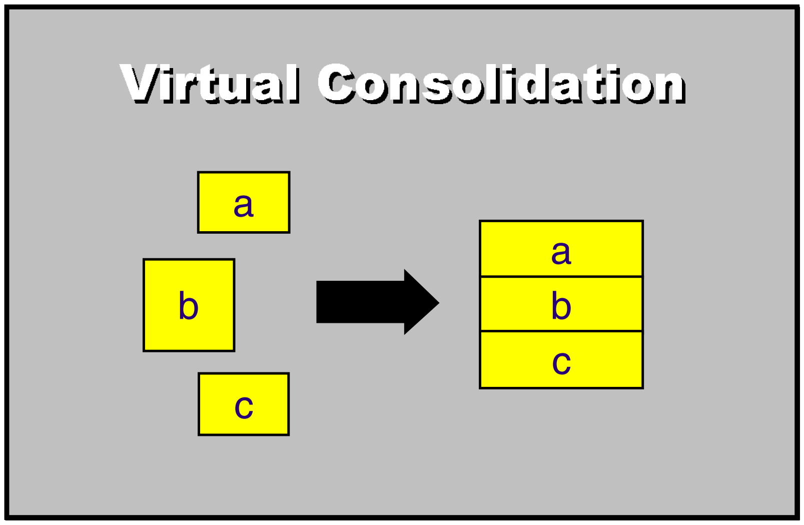

*Figure 5 Virtual consolidation*

Even with the number of operating systems installed staying the same, administration and maintenance cost can still be reduced by using cloning techniques. Deployment of new servers is very fast, because they are only logical definitions and can share the already available hardware resources.

## Things to keep in mind for a consolidation with Linux for zSeries

There are some points to be aware of when investigating the potential for a server consolidation:

- ISV software: There are several ISV software packages that are not yet available for zSeries Linux. A consolidation on Linux for zSeries is only possible if software is either available, or migration to an available alternative is acceptable.
- Network: Consolidation, especially of distributed servers, may require more network bandwidth from the computer center to the distributed locations.
- Personnel cost savings: These come mostly from changes in the processes and organization of a company. If a company wants to do everything exactly the same way on Linux for zSeries as they do it on distributed servers, the personnel cost saving may be small.

- Depreciation: Only in a very few cases will the gains achieved by a server consolidation justify replacing the existing hardware and software and the resulting loss of the net depreciated value.
- Leasing or other contracts: They can create additional costs in a server consolidation. For example, server leases have to be paid, even if the servers are moved to Linux on zSeries

### Issues that help drive server consolidation

The following issues are among those that support server consolidation in many specific cases.

- Server farm growth

  Normally, the size of a server farm does not stay flat, but grows, sometimes very rapidly. It may be very feasible to manage and consolidate future growth by changing the strategy now. It is also very easy to migrate the already existing servers into the consolidated environment if reasons like depreciation are no longer valid.
- Increased availability of a consolidated server farm

  The consolidation of the different servers of a Web application can lead to a synergy effect of higher availability. This is especially true for a consolidation on Linux for zSeries, since not only the servers, but also network and storage will be consolidated and simplified. This reduces complexity, making the environment more robust and increasing availability.
- Streamlining of a server farm environment

  Many historically grown parts of a server farm will have to be reconsidered and streamlined for a consolidation. This will increase efficiency of operations and thereby lower cost.

## How to map a UNIX/Intel farm to a Linux for zSeries farm

In most consolidation exercises it is necessary to determine how many distributed servers can be consolidated onto a zSeries machine running z/VM and Linux. Here is a “quick” way of estimating that may help. This is merely a rule of thumb approach, *not* an official IBM sizing. It is intended only as a guideline to conceptualize and to help determine if the workloads being examined are a reasonable fit for zSeries. To do this we need to know the utilization of the distributed servers, number of processors, MHz rate of the servers, and the workload-factor (WLF).

These machines achieve an actual ratio of throughput which is workload-dependent. The workload dependency, called WLF in our model, should be used to adjust for differences in architecture between server platforms. It can be used to accommodate such values as:

- Application path length
- Application specifics (Java, PHP, WAS, DB, and so on)
- IO values for storage
- IO values for the network
- Cache and memory efficiencies dependent on architecture

Figure 6 shows that there is no fixed relationship between different CPUs (SUN, Intel and zSeries, to name a few). Rather there is a more or less large dependency on the type of workload (WLF) as well as the utilization of the servers. So the ratio or WLF will vary by application type and utilization.

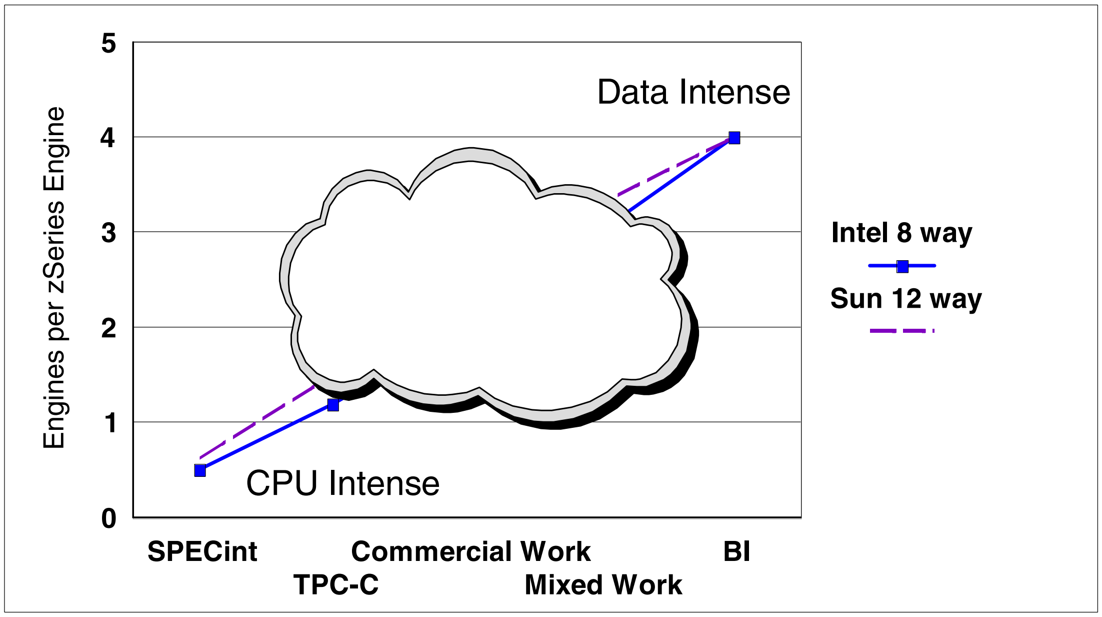

*Figure 6 CPU comparison at 100% CPU utilization for SUN, Intel and zSeries*

Figure 6 shows that in a best case we will be able to handle the workload of 3 to 4 SUN/Intel CPUs (100% utilized) with one zSeries CPU. This also means, in a case where the CPU utilization on distributed servers (like SUN/Intel) is, for example, 10% instead of 100%, that a zSeries is able to handle the workload of about 30 to 40 SUN/Intel CPUs with a similar clock speed as the zSeries.

You can apply the following “logic” to get some base numbers:

$$\text{Processors}_z = (\text{Utilization}_{other} * \text{Processors}_{other} * \text{MHz}_{other}) / (\text{MHz}_z * \text{WLF})$$

In this formula the assumption is that the zProcessor runs at 100% utilization.

**Note**: The workload calculation shown will quickly give you a generalized value of the processor numbers that can be consolidated. It also shows where a specific workload should NOT be considered for consolidation. An example of a bad fit would be an application such as animation rendering, which is running on a 2 GHz Intel processor, at 100% CPU utilization. It can be shown quickly that this workload would not be a cost-efficient candidate for consolidation onto a zSeries processor. Keep in mind that the zSeries is a Data-Processing server.

**Note:** This quick “sizing” method is not an official IBM sizing. For precise sizing information contact your IBM representative and get an official sizing (SIZE390).

This calculation is for quick estimates only. *Do not* use this for a final sizing. Given the simple nature of the equation, do not move the %CPU utilization up beyond the recommended values, unless there is *actual* data to support this.

Another way to get a feeling for the possible utilization is to look at the network utilization of a server farm. Here is an example to explain what is meant by this:

- Consider a Web server farm of 100 servers
- The company network backbone is a Gigabit Ethernet network, which translates to linespeed bandwidth of 125 MB/s.

The SpecWeb99 benchmark (`http://www.spec.org/osg/web99/`) is a fairly simple and mostly static HTML-oriented benchmark and will not simulate the complexity of most real Web applications, but it gives enough information for a first analysis of Web environments. It measures how many parallel connections of around 50 KB/s a system can handle. Available on the Web site is also a list of submitted measurements. One measurement entry is for the IBM eServer xSeries 343, a 2-way 1,26 GHz PentiumIII system (see details at `http://www.spec.org/osg/web99/results/res2002q2/web99-20020318-00182.html`). It can handle up to 3416 parallel SpecWeb99 connections of 50KB/s, or 166 MB/s bandwidth in summary!

This means that a 3/4 utilized system can saturate the line speed of a Gigabit Ethernet network or in the example, the whole company network backbone. Additional network traffic generated for everything other than the Web server is not even included, either! Even assuming a 10 times higher CPU utilization for a real Web application containing dynamic parts (for example, in Java), the average server in the server farm will only have a low utilization.

Sample simple calculation (not a official sizing):

- 100 Intel single CPU machines
- Clock speeds of 200 MHz
- 5% CPU Utilization
- Putting the workload onto a z800 running at 80% CPU utilization
- The workload can be handled with 2 - 3 processors (CPUs).

## Server Farming with z/VM and Linux for zSeries

This section shows an example on how a cloned and shared Linux server farm under z/VM can be set up and how maintenance can be done.

### Initial setup for the cloning environment

First we describe how to set up a cloned and shared environment, including the issues to consider in planning, and items to be careful of.

#### z/VM setup

The z/VM setup is not really covered in detail here since it is already described in detail in several other sources, such as the IBM Redbook "ISP/ASP solutions” or “Cloning Linux Images in z/VM” written by Tung-Sing Chong.

To minimize the effort of defining the z/VM guest definitions for the Linux clones, always map disks to the same addresses for the guests. In the setup used here, the network is implemented using one Linux system as the router. This system exclusively owns one GB Ethernet card and is connected to the Linux farm clones via IUCV. Generation of network definitions for a new clone can be easily automated with some scripting.

The basic concept is the same as using a VM TCPIP stack as the router. Some very good examples for that are shown in the documents mentioned previously.

#### Linux setup

The first step in setting up the Linux cloning environment is to decide where the system will be split into a read-write and a read-only part. The easiest way to do this is to just move the /opt and /usr directory tree onto a read-only file system. Here is an example showing the space usage from the "/" directory:

```
donald09:/ # du -h -s  *
6.2M   bin     3.3M    boot    0       cdrom
24k    dev     22M     etc     0       floppy
20k    home    9.5M    lib     16k     lost+found
12k    media   12k     mnt     105M    opt
514M   proc    876k    root    9.7M    sbin
32k    tmp     697M    usr     21M     var
```

Some directories can be ignored since they do not contain any installed files, like /proc for the proc file system or /dev. Others, such as /cdrom, /floppy, /mnt, and /media are mount points containing no data. This means only 9% of the data are in other directories than /opt and /usr. With the root ( /, not to mistaken for /root) directory in the read-write file system, the user of the Linux system clone still has a maximum amount of flexibility, while sharing 91% of the system read only.

On the other hand, this also has some impact on system administration and maintenance. Having more parts of the operating system on the read-only file system means that a user can only mess up a much smaller area of the system by mistake. It can even be limited down to the clone-specific level, so that a user can only destroy the configuration of his own applications.

For maintenance, it has to be considered that many packages put binary files in /lib, /bin, /usr, and /sbin. The configuration information is normally stored in files in /etc. Dynamic data like log files and database files are normally created in /var.

This has the consequence that in an environment where only directories like /tmp, /var, /home, /root, and parts of /etc are copied to a private read-write file system, it will be much easier to put update rpms into the system, as the file replace or add file not only in /usr and /opt, but also in /lib, /bin, or /etc. Therefore, we decided on the configuration shown in Figure 7.

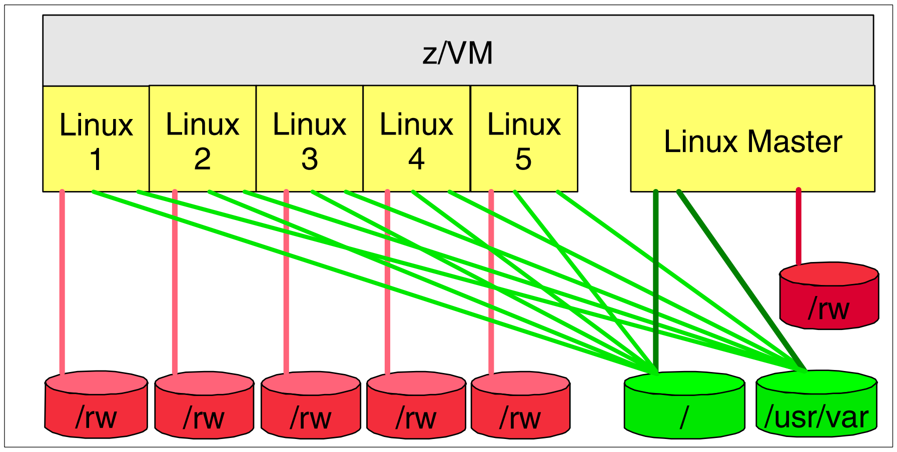

*Figure 7 System setup*

This is the directory of our read-only / file system:

```
drwxr-xr-x   2   root  root   4096    May   7 17:12 bin
drwxr-xr-x   3   root  root   4096    May  13 13:25 boot
drwxr-xr-x   6   root  root   12288   May  13 11:56 dev
drwxr-xr-x   25  root  root   4096    May  13 13:27 etc
```

```
lrwxrwxrwx   1   root  root   8       May 10 09:48 home -> rw/home/
drwxr-xr-x   6   root  root   4096    May  8 13:59 lib
drwxr-xr-x   4   root  root   4096    May  7 17:07 media
drwxr-xr-x   2   root  root   4096    May 10 09:48 mnt
drwxr-xr-x   11  root  root   4096    May  7 17:11 opt
dr-xr-xr-x   43  root  root   0       May 13 13:29 proc
lrwxrwxrwx   1   root  root   8       May 10 09:49 root -> rw/root/
drwxr-xr-x   11  root  root   4096    May 13 16:25 rw
drwxr-xr-x   4   root  root   4096    May 10 13:01 sbin
lrwxrwxrwx   1   root  root   7       May 10 09:49 tmp -> rw/tmp/
drwxr-xr-x   17  root  root   4096    May 13 16:26 usr
lrwxrwxrwx   1   root  root   8       May 13 16:25 var -> /rw/var/
```

The necessary read-write directories, such as /tmp, /var, /home, and /root were moved to a second disk, which was mounted as read-write. Also, /usr/local was redirected to the read-write disk. Here is the /rw directory:

```
drwxr-xr-x    2    root  root   4096    May 13 13:29 lib
drwxr-xr-x    2    root  root   4096    May 13 13:29 bin
drwxr-xr-x    2    root  root   4096    May 13 13:29 usr
drwxr-xr-x    2    root  root   4096    May 13 13:29 sbin
drwxr-xr-x    2    root  root   4096    May 13 13:29 dev
drwxr-xr-x    4    root  root   4096    May 13 13:27 etc
drwxr-xr-x    3    root  root   4096    May  7 17:07 home
drwxr-xr-x    14   root  root   4096    May 13 16:25 local
drwx------    4    root  root   4096    May 13 09:21 root
drwxrwxrwt    8     root  root  4096    May 13 13:35 tmp
drwxr-xr-x   17    root  root   4096    May  7 17:12 var
```

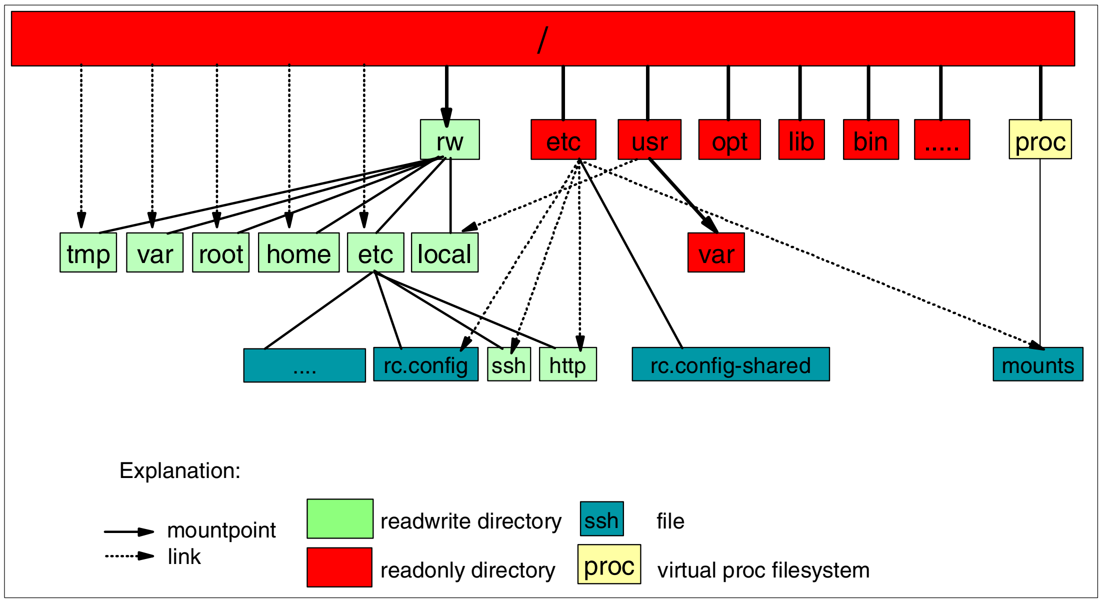

*Figure 8 Directory tree*

This setup used another disk mounted under /usr/var, shared read-only between all clones, to store common configuration files and maintenance jobs. It is explained in more detail in "Maintenance in a cloned and shared environment" on page 16.

## The general cloning setup

As shown in Figure 7 on page 9, we decided to have one master image, where we did the initial installation and which was subsequently used for administration and maintenance tasks. We had three disk copies of the master installation. The first copy was for the master instance as the working operating system. The second copy was used for testing of changes in the configuration and maintenance scripts. The third copy was the master copy that was linked to the clones. When a configuration change or a maintenance job was thoroughly tested, the test disks were copied to the master copy disks. The clones picked the new disks up the next time they were rebooted. This also makes it possible to have several copies of the disk sets, and therefore, if a maintenance update breaks the system, a system administrator can easily switch back to a previous service level by linking to an older set of disks.

A Linux system, especially a standard SuSE installation, is not prepared for a split into a read-write and a read-only part, especially if the read-only part also includes the /etc directory. Therefore, there had to be some additional changes made. Other distributions, such as Debian, are more easily split into a read-write and a read-only part. Now let's have a closer look at the modifications to the SuSE installation.

## The /etc/rc.config SuSE configuration file

The SuSE-specific rc.config file stores most of the configuration information. It is read by all SuSE startup and configuration scripts. It is also used in the first startup script boot in /etc/init.d, which is executed when the read-write file system is not already mounted. It also contains individual information for each clone, such as the TCP/IP configuration.

To resolve this issue we decided to use two copies. One is stored under a different name, like rc.config-shared in /etc in the shared read-only file system, and the other copy is in /rw/etc, where the individual changes for the clones can be made. The scripts can find this copy via a symbolic link in /etc:

```
lrwxrwxrwx  1 root  root 17  May  22 10:49 rc.config -> /rw/etc/rc.config
```

The only file which will need the rc.config-share file is the /etc/init.d/boot startup script. Therefore, this line has to be changed:

```
. /etc/rc.config
```

has to be changed to:

```
. /etc/rc.config-shared
```

At least three more changes to the rc.config file have to be made:

First, disable the SuSEconfig scripts because they will not work in this kind of environment:

```
ENABLE_SUSECONFIG="no"
```

And, for the TCP/IP dynamic configuration, one line has to be added at the end:

```
. /rw/etc/rc.config-tcpip
```

This is explained “Changes to the TCP/IP setup” on page 12.

The parameters for the syslog have to be changed in the script so that the syslog uses a device in /rw/dev:

```
SYSLOGD_PARAMS="-p /rw/dev/log"
```

These are only some basic changes. For a more sophisticated environment there may be additional changes necessary.

## The /etc/init.d/boot script

This script is the first one to be executed at startup. One change necessary was already described in the previous section.

The SuSE boot script contains a section where the checking and mounting of the file systems is done. The root file system is mounted as read-only for the checks; if no error occurs, the last step is to mount it as read-write. This will not work here, as the root file system is a read-only file system. Therefore, all these commands had to be commented out:

```
# mount -n -o remount,rw /
```

The status of the mounted file systems is stored in /etc/mtab. With the new Linux 2.4.x kernels, a better idea is to use /proc/mounts. Therefore, and also to avoid the problem with /etc being read-only, we placed a symbolic link from /etc/mtab to /proc/mounts.

```
lrwxrwxrwx    1 root     root  12 May 13 10:48 mtab -> /proc/mounts
```

The SuSE boot script contains a line that deletes leftover files from the last time the system was up. We don't have to worry about that since our root file system is read-only and there will never be any leftovers. Therefore we also comment this out:

```
 #  rm -f /etc/mtab* /etc/nologin /nologin /fastboot
```

At every startup, the SuSE boot script executes the zic program, which makes a link for the configuration file /etc/localtime, containing the time zone setting. Actually, it is sufficient to do this only the first time, and then the file can remain unchanged as long as the time zone of the server does not change. Therefore, this line can be commented out:

```
#    /usr/sbin/zic -l $TIMEZONE
```

These are only the basics steps necessary to make the /etc/init.d/boot script work in a shared environment. For a real production system, around three quarters of this script, especially the parts inherited from the Intel Desktop history, should probably be cleaned up. Depending on the usage of the clones and the details of how the shared environment is set up, more changes may be necessary. The rc.config file is already missing in the SuSE 8.0 professional version for Intel, so there is hope that in future versions of the SuSE Linux Enterprise Server this may be much easier to modify for a shared environment.

### *Changes to the TCP/IP setup*

The TCP/IP setup is individual for each Linux clone, but it is something which should not be changed by the clone user. Our approach was to have some scripts in the read-only file system that automatically configure TCP/IP at startup.

Here is a small perl script that uses the **cpint** tool (open source, available in the Internet) to find out the VM guest name. Then it looks into a file on the shared read-only file system where, in this example, for each clone there is an entry containing the guest name, the IP address, and the peer IP address. As previously mentioned, the clone's network connection is an IUCV connection to another Linux router system. The script uses the information in the table to write a small config file to /rw/etc/rc.config-tcpip, which overlays the values stored in /rw/etc/rc.config.

```
#!/usr/bin/perl

use strict;

my($HCP) = '/sbin/hcp';
my($HOSTNAMES) = '/usr/var/config/hosts';
my($CONFIG_LOCAL) = '/rw/etc/rc.config-tcpip';
my($DEBUG) = 0;
my($guest) = '';
my($ip) = '';
my($peer) = '';

my($GUESTNAME) = split(/ /,`$HCP query userid`);
print("HOST: $GUESTNAME\n") if $DEBUG;

open(HOSTNAMES,"< $HOSTNAMES") || \\
  die("Could not open $HOSTNAMES: $!\n");
while(<HOSTNAMES>)
{
    chomp;
    ($guest, $ip, $peer) = split;
    last if ($guest eq $GUESTNAME);
}
close(HOSTNAMES);

print("Guestname: $guest\n") if $DEBUG;
print("IP:        $ip\n") if $DEBUG;
print("Peer:      $peer\n") if $DEBUG;

open(CONFIG_LOCAL,"> $CONFIG_LOCAL") || \\
  die("Could not open $CONFIG_LOCAL: $!\n");
print(CONFIG_LOCAL  \\
  "#Automatically generated configuration file. Do not edit!\n");
print(CONFIG_LOCAL qq{NETCONFIG="_0\"\n} );
print(CONFIG_LOCAL qq{IPADDR_0="$ip"\n} );
print(CONFIG_LOCAL qq{IFCONFIG_0="$ip pointopoint $peer"\n} );

close(CONFIG_LOCAL);
```

This uses the file /usr/var/config/hosts, a simple text file shared in the read-only file system. It is placed on the additional disk for administration and maintenance and contains one line with configuration information for each clone. Similarly, there could be more files containing other individual information for the clones that should be administered and stored centrally, for example, information about additional disks to mount.

```
DONALD09        9.164.178.29    9.164.178.1
FAILURE         9.99.99.1       9.99.99.2
```

The z/VM guest name has to be in upper case.

The configuration file created by the perl script looks like this:

```
#Automatically generated configuration file. Do not edit!
NETCONFIG="_0"
IPADDR_0="9.164.178.29"
IFCONFIG_0="9.164.178.29 pointopoint 9.164.178.1"
```

This file will be read after the rc.config values and will overlay the values stored there. The perl script has to be executed before the start of the network. This approach has several advantages. The information is stored and can be maintained in one place. The script will be executed at every boot. This means that if a guest has been moved to another router or has got a new IP address, only the general configuration file has to be changed. And finally, the scripts all reside in read-only protected space. Therefore, even if a clone user breaks the network setup, everything will be set to normal after a reboot.

A similar script can also be used to set up the route.conf file, and after the network device is started, DNS commands can be used to get the name and the domain of the clone and set up the /etc/hosts and /etc/HOSTNAME files accordingly. For that, the following files were redirected to the read write file system:

```
lrwxrwxrwx    1 root     root  16 May 13 13:10 HOSTNAME ->     /rw/etc/HOSTNAME
lrwxrwxrwx    1 root     root  18 May 13 13:27 route.conf ->   /rw/etc/route.conf
lrwxrwxrwx    1 root     root  13 May 13 13:11 hosts ->        /rw/etc/hosts
```

## Disk setup

The disk mounting was done in two steps. The disks that are common to all clones, like the shared disks and the basic read-write disk, were listed in /etc/fstab and automatically mounted by the SuSE startup scripts. Therefore, the /etc/fstab remained on the read-only root file system.

Additional private disks and disks not common to all clones were mounted by a script using a technique similar to that used for TCP/IP setup. Common disk address ranges were used for the private disks, too. This allowed us to list the disks in the parmfile for the **zipl** command. Then we used a script to read out the disk information and mount the disks later on in the boot process.

If a company does not want to configure all disks with the same virtual addresses in the z/VM guest configuration and therefore also in the Linux kernel parmfile, this command can be used to dynamically add disks in a script:

```
echo "add device range=devno-range" >> /proc/dasd/devices
```

## Add-on for individual startup scripts

A clone user want to be able to install additional applications and packages. There has to be a possibility to add scripts to the startup process.

This was done by adding a script /etc/init.d/run-rw to the SuSE startup:

```
#! /bin/bash
### BEGIN INIT INFO
# Provides: additional installes
# Required-Start: $network
# Required-Stop: $network
# Default-Start: 3 5
# Default-Stop: 0 1 2 6
# Description: Start additional stuff
### END INIT INFO
# mounts for special chroot enviroments
mount --bind /bin /rw/bin
mount --bind /lib /rw/lib
mount --bind /sbin /rw/sbin
mount --bind /usr/sbin /rw/usr/sbin
mount --bind /usr/bin /rw/usr/bin
mount --bind /usr/lib /rw/usr/lib

# execute indivdual stuff

/usr/local/bin/run-parts/rw/etc/init.d/
```

This script uses the program **run-parts**, which is not part of the SuSE distribution. It is part of the debianutils package of the Debian distribution; Red Hat has a clone in the crontabs package. We used the Red Hat package, where the program is a shell script which can be easily modified for additional needs. This program executes the scripts found in the directory provided as parameter. The comment in the beginning allows use of the **insserv** command of the SuSE installation to easily include the script into the SuSE rc.d startup process.

### Changes to the passwd db installation

The clone user will need to be able to create new users and will want to have his/her own passwords. Therefore, the files containing this information will be moved to the /etc/rw directory:

```
lrwxrwxrwx    1 root     root           14 Jun  5 16:45 passwd -> /rw/etc/passwd
lrwxrwxrwx    1 root     root           14 Jun  5 16:48 shadow -> /rw/etc/shadow
lrwxrwxrwx    1 root     root           14 Jun  5 16:45 group -> /rw/etc/group
lrwxrwxrwx    1 root     root           14 Jun  5 16:48 gshadow -> /rw/etc/gshadow
```

The commands **useradd**, **userdel**, **usermod**, **groupadd**, **groupdel**, **groupmod** and **passwd** use these files. To ensure that there is no corruption happening, they create lock files and backup files before a change in /etc, which in our setup is read-only and will cause them to fail without additional adaptation.

One way to do this is to adapt the source code and to move the backup and lock files to directory on the read-write disk. We chose another alternative, using a chroot environment for the execution of these commands.

There are some modifications necessary to prepare the system. The steps we used were to:

- Copy the /etc/skel directory to /rw/etc
- Copy the /etc/pam.d directory to /rw/etc
- Copy the file /etc/nsswitch.conf to /rw/etc
- Create some empty directories: /rw/sbin, /rw/lib, /rw/usr/bin, /rw/usr/sbin and /rw/usr/lib
- Set the suid flag for suidperl: chmod s+u /usr/bin/suidperl
- Create a directory /usr/var/bin, where we put some wrapper scripts for the commands listed above
- Put the directory /usr/var/bin into the first position of the PATH variable in /etc/profile in the following two lines:
  - test "$UID" = 0 && \ PATH*=/usr/var/bin:/*usr/local/sbin:/sbin:/usr/sbin:$PATH
  - test "$UID" = 0 || PATH="=/usr/var/bin:$PATH:."

We then put the following wrapper script wrapcmd into the /usr/var/bin directory:

```
#!/usr/bin/suidperl
my(@allowed_commands) = ('passwd','chsh','chfn','useradd',
'userdel','usermod','groupadd','groupdel','groupmod');
$ENV{PATH}='/usr/local/bin:/usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin';
chroot("/rw") || die("could not chroot to /rw: $!\n");
# set euid back to the calling uid
($>,$)) = ($<,$();
# get the basename of the command
my($command) = $0;
$command =~ s/^.*\///;
for (@allowed_commands)
```

```
{
    if ($_ eq $command)
    {
        exec($_, @ARGV);
    }
}
print("Command '$command' is not allowed to execute.\n");
```

This perl program first changes the root environment to /rw, then checks if the called program is in the list of allowed programs, and if yes, executes it. Any lock and backup files will now be allocated in /rw/etc. The script must have set the suid flag.

To use this script for several commands, we just added now some symbolic links to the /usr/var/bin directory:

```
lrwxrwxrwx    1 root     root           7 Jun 20 14:20 groupadd -> wrapcmd
lrwxrwxrwx    1 root     root           7 Jun 20 14:20 groupdel -> wrapcmd
lrwxrwxrwx    1 root     root           7 Jun 20 14:20 groupmod -> wrapcmd
lrwxrwxrwx    1 root     root           7 Jun 20 14:45 passwd -> wrapcmd
-rwsr-sr-x    1 root     root         241 Jun 20 14:19 wrapcmd
lrwxrwxrwx    1 root     root           7 Jun 20 14:20 useradd -> wrapcmd
lrwxrwxrwx    1 root     root           7 Jun 20 14:20 userdel -> wrapcmd
lrwxrwxrwx    1 root     root           7 Jun 20 14:20 usermod -> wrapcmd
```

Because we put this directory at the first position of the PATH, it will be executed instead of the original binaries.

These are the basic commands for the administration of the passwd database. There are some additional commands we did not test, but it should be possible to wrap them accordingly and put them in the list of allowed commands.

### Additional changes

There may be some more changes necessary, depending on how much of the system configuration has to be read-write accessible to the clone users.

Configuration files or directories can be redirected like:

```
lrwxrwxrwx    1 root     root  14 May 13 13:08 httpd -> /rw/etc/httpd/
lrwxrwxrwx    1 root     root  12 May 13 13:09 ssh -> /rw/etc/ssh/
```

The /etc/httpd directory contains the configuration of the Web server, which will be individual for each clone.

It is also necessary to have an individual directory for the secure shell server sshd. The keyrings of the server, which are unique for each clone, are stored in the /etc/ssh subdirectory.

Similar changes will be necessary for other applications, such as MySQL, Postgres, or Squid.

## Maintenance in a cloned and shared environment

The split of the system into a private read-write file system and a read-only file system affects how maintenance has to be done. The rpm tool stores the information about the installed packages in /var/lib/rpm/. This means that a clone user can also install additional rpm packages, if the packages do not write files to the read-only part of the directory tree.

## RPM usage by the Linux clone user

As an example, we used the MySQL rpm package of the SuSE distribution:

```
rpm -qlp mysql.rpm
```

This shows the files in the package and the path where they would be installed.

The files in this package will, by default, be installed in /usr and /etc. In this setup, these directories are a part of the shared read-only file system and not write accessible for the clone user.

With the **`--relocate`** option, these paths can be changed:

```
rpm -i --relocate /usr=/usr/local --relocate /etc=/usr/local/etc mysql.rpm --badreloc
```

Now /usr will be translated to /usr/local and /etc to /usr/local/etc. If the package is not built as being relocate-able, the **`--badreloc`** option is used to enforce it. RPM packages normally also include pre- and post-install scripts. These can be printed using:

```
rpm -qp mysql.rpm --scripts
```

It most cases the automatic execution of the scripts will fail in this kind of environment with most of the operation system residing on a read-only file system. This means that the necessary pre- and post-install actions have to be done manually. For the rpm installation, the execution of the scripts can be disabled with the command option **`--noscripts`**.

## Maintenance of the shared system

As described, this rpm database cannot be used for common maintenance updates of the read-only shared system. Therefore, there have to be two rpm databases: one for the private usage of the clone and one for the shared base system. This can be set up after the installation of the master system. In this example, the original database in /var/lib/rpm was left there as private copy. A copy of the database in /usr/var/lib/rpm will be used for the master system maintenance.

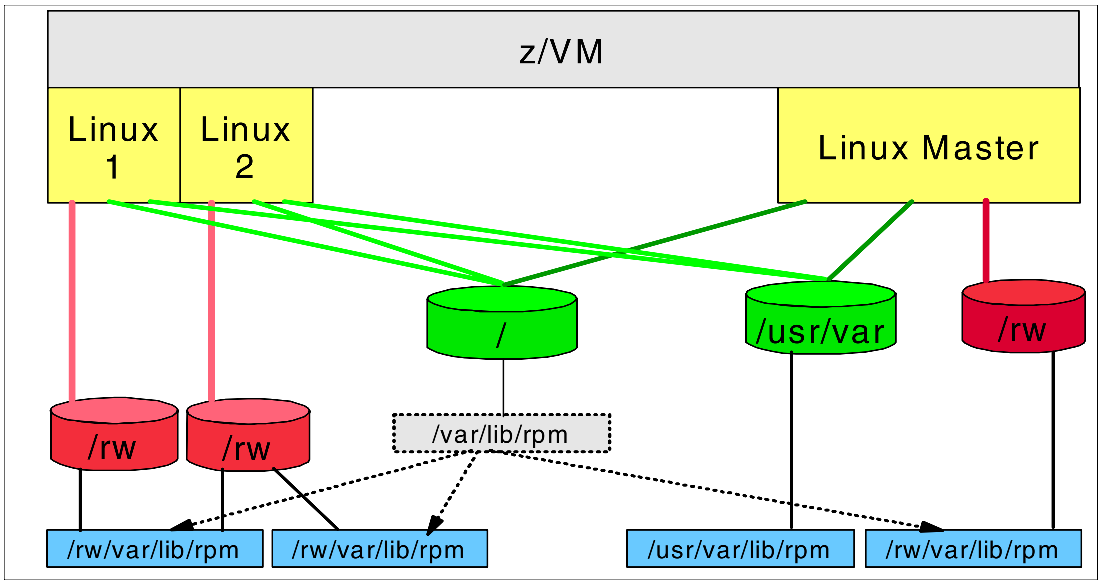

*Figure 9 rpm database setup*

The system administrator can now install a new or update an existing package with the **`--dbpath`** option to use the shared rpm db. Any updates of the system disk should not be done on the shared disk, but on a private copy. Then the VM directory entries can be changed to point to the new master disk as described in “The general cloning setup” on page 11. This also allows the administrator to keep different maintenance levels for the system disk.

Later on, when the clone reboots at its scheduled maintenance window, the guest should be configured to use the new VM guest definition and the clone will automatically have the new maintenance level. To update the private copy of the rpm database use the following approach.

The rpm command has a option to only update the database but not do any changes on the file system:

```
rpm -i mysql.rpm --justdb
```

We used this to update the private rpm database copies of the clones. This can be done automatically at startup. For this, several jobs were added to the environment. Here is an overview:

- run-maintenance is executed in the rc.d startup process and calls.
- run-post-scripts checks for new maintenance directories and executes any scripts found in the directories.

In /etc/init.d/, run-maintenance was created and added to the rc startup process with the **insserv** command:

```
#! /bin/bash
### BEGIN INIT INFO
# Provides:       maintenance
# Required-Start: $local_fs
# Required-Stop:  $local_fs
# Default-Start:  2 3 5
# Default-Stop:
# Description:    Start the Apache httpd daemon
### END INIT INFO

/usr/var/maint/bin/run-post-jobs
```

It executes the script run-post-jobs, which checks for new maintenance jobs and starts them.

```
#!/bin/sh
# Name of the control file.
# It contains the last job-id that has been run.
CONTROL_FILE=/var/lib/maint/post-job-id

# Directory that contains the sub directories with the
 post-install-scripts
JOB_DIR=/usr/var/maint/jobs

PATH=/usr/local/bin:/usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin

if [ ! -f $CONTROL_FILE ]
then
   echo 0 > $CONTROL_FILE
fi

job_id=`cat $CONTROL_FILE`

for i in $JOB_DIR/*
```

```
do
   nr=`basename $i`
if [ $job_id -lt $nr ]
then
   run-parts $JOB_DIR/$nr
       job_id=$nr
        echo $job_id > $CONTROL_FILE
fi
done
```

This script reads the name of the latest maintenance level executed from the file /var/lib/maint/post-job-id. It then looks in /usr/var/maint/jobs for directories that have a name greater than the name in the control file. The algorithm used in the scripts only works for numbers, not for entries containing other characters types. So we chose the layout yyyymmddnn. yyyy stands for the year, mm for the month, dd for the day and nn is an additional number to distinguish maintenance done in one day. Here is a sample directory:

```
donald09:/usr/var/maint/jobs # ll
drwxr-xr-x    2 root     root         4096 May 22 11:48 2002052201
drwxr-xr-x    2 root     root         4096 May 22 11:49 2002052202
```

One of the jobs should always contain the update of the individual clones' rpm databases by calling rpm with the option **--justdb**. The other jobs should contain the post-install actions. This can also be used to centrally administer the clone configurations that reside in the individual read-write parts.

The script then uses the **run-parts** command, which has already been described, to execute all scripts in the directories. After executing the maintenance scripts in one directory, the control file is changed to reflect the new level.

## Monitoring of maintenance jobs

It should be verified that the jobs did not fail. Here are two example techniques illustrating how this can be done. In general the jobs should send an e-mail to a maintenance e-mail address to inform about success or failure of the maintenance. Here is a small sample shell script:

```
#! /bin/sh
mutt -s test maintenance@central <<EOF
Maintenance 2002052201 successfull applied!
EOF
```

This can be used as an skeleton to be included in the maintenance jobs.

For e-mails not only delivered locally, sendmail has to be installed and configured to forward messages to other servers.

For more information about why the maintenance failed, we sent the output of the maintenance jobs to a central syslog facility. For that, the /etc/syslog.conf must be changed:

```
local0.*                @central
```

This will now send all syslog entries for the facility local0 to the syslog server central. Message lines can be sent from the maintenance jobs by using the **logger** command:

```
logger -p local0.info -t clone5 "This is the message"
```

The option **-t** should contain the name of the clone to distinguish the messages of the different clones and it is also possible to use a different priority in the **-p** option, for example, **local0.err, local0.warn**.

The goal of this section was to give an introduction to how a shared and cloned environment with Linux for zSeries and z/VM might look. It was never intended to be a production-ready setup, but only to give some ideas and skeletons for how a real setup can look, and how, in general, it can work.

## Example file server consolidation with Samba

Samba is an implementation of the Microsoft SMB/CIFS protocol. It can be used to offer the file, print, and authorization services on a Linux/UNIX server. For a detailed discussion of Samba, refer to the IBM Redbook *Samba Installation, Configuration, and Sizing Guide* or visit the Web site of the Samba project at:

http://www.samba.org

A Windows NT 4-server environment consists of 3 kinds of servers: primary domain controllers (PDC) and the backup domain controllers (BDC), which are used for authentication, for logon and access control to the resources; and file and print servers. The current version of Samba, 2.2.4 works well with the Windows NT 4-domain concept.

Samba can replace all 3 kinds of servers. It is also possible to mix Windows servers and Samba servers, for example, having Samba file and print servers connect to a Windows PDC/BDC, or having a Samba PDC doing authentication for a Windows file/print server. This makes it very easy to migrate because the servers can be moved one at a time instead of having to switch the whole environment in one step. It is not possible to mix Windows PDCs with a Samba BDC or the other way around.

Windows NT 4 distinguishes between two types of logon servers: primary and backup domain controllers. Samba does allow multiple, redundant logon servers (domain controllers) in a domain, but does not implement the master(PDC)/slave(BDC) user account replication protocol required to support Windows NT 4 backup domain controllers in the same domain with a Samba domain controller.

Windows 2000 domain controllers use a complex, proprietary multi-master replication protocol. In most cases you should not try to mix Samba and Windows 2000 domain controllers in the same domain.

A migration is best started by moving the file and print servers. Samba can connect to the Windows PDC/BDC servers (via Winbind). The PDC/BDCs can either remain on Windows or be moved to Samba, too. Windows clients can use Samba as logon servers.

If multiple Samba logon servers are used, the user information is stored in an LDAP server as back end. If the PDCs are also migrated to Samba, our opinion is that the file and print server should then not connect to the Samba logon servers via Winbind, but instead do the authentication check directly against the LDAP backend. Another alternative to LDAP is the usage of NIS.

The successor of the Windows NT 4 server environment uses Active Directory Server (ADS) for authentication. This is a protocol related to LDAP that integrates NT-style RPCs with LDAP and Kerberos. The currently available Samba version 2.2.4 does not support Active Directory LDAP and Kerberos interfaces, though it can still participate as a file and print server in an Active Directory Domain, even in Native Mode. The ADS LDAP and Kerberos support is planned to be included in the Samba 3.0 version.

A developer code version can already be downloaded and the plan for version 3.0 is to become available in fall 2002. A Samba 3.0 implementation is able to work as a file and print server and can connect to an ADS for authentication and use Kerberos for authentication to an ADS server. A future release will likely support replacement of an ADS domain controller by a Samba server, too.

In a production environment, a Samba server should be set up to use a journaling file system and access control lists (ACL) for Windows NTFS-like security. This is, for example, possible with a Linux system running on a kernel 2.4.17+ plus ACL patches and Samba 2.2.4.

These are POSIX ACLs, which differ slightly from the Windows NT version. In Windows NT, there are two more functions available: “append only” and “write/edit but not delete.” Administration of user accounts on Samba file servers can be mostly done with the Windows administration tools, but there are some slight differences in functionality. Alternative administration interfaces, such as webmin, which offers a Web browser interface, are available.

### Samba and LDAP

Samba can be combined with an LDAP server as a backend for authentication. This allows you to merge authentication for Java applications, UNIX and Windows in a unique way. Most UNIX systems nowadays can use LDAP for user authentication, and Java has an interface to LDAP, too.

Some Java application servers, like IBM WebSphere, have complete LDAP integration. Therefore, Web applications, UNIX systems, and—via Samba PDCs—even Windows clients and file/print servers could use the same LDAP server as a central authentication authority.

## Example environment

Here is an example of a file and print server farm running on Windows and a mapping to a Samba server farm running on Linux for zSeries. This can be used as a skeleton for mapping a real-life environment.

Assumptions: Normal Windows servers are running with a utilization of less than 5%. File and print serving tends to be more I/O intensive.

### Windows Environment

105 servers, 5 PDCs, 30 print servers and 70 file servers

- Hardware
  - 5 PDCs: 500 MHz P4, 512 MB RAM, 1 FastEthernet adapter and 30 GB disk
  - 30 print servers: 2x 500 MHz P4, 1 GB RAM, 1 FastEthernet adaptor and 60 GB disk
  - 70 file servers: 500 MHz P4, 512 MB RAM, 1 FastEthernet adapter and 120 GB disk
  - 10x 19" racks with, each for 11 servers plus network switch
- Power
  - 115 x 300W = 34,5 KW
- Space
  - 10x 19" racks floor space
- Software
  - 105 Windows NT installations
- Network

  - 10 network switches, with 11x FastEthernet ports and a GB-Ethernet port to for the connection to the company backbone network.
- People:
  - 6 system administrators (~15 servers per administrator)

### Consolidated Samba on Linux for zSeries environment

105 Linux instances running under z/VM

- Hardware
  - 4 zSeries IFLs on existing HW, 32 GB RAM, 2 GB Ethernet adapters, for disk 12 ESS storage system with 8 TB.
- Power and space
  - 7KW for one ESS storage system
  - One ESS storage system, floor space similar to 4x 19" racks.
  - No additional space and power for zSeries system.
- Software
  - Cost for a Linux distribution (and consider, the Debian distribution is available for free)
  - z/VM for 4 IFLs
- Network
  - 2 shared GB Ethernet adapters already mentioned in Hardware
- People
  - In a highly optimized environment, 3 system administrators

## Web server

Simple Web serving on Linux represents a very good place to begin server consolidation. Web servers running on UNIX-based servers can be moved directly. Web server content can consist of many components:

- Static content: Directory structures and files (HTML files, xml files, JPEG/GIF pictures, programs, and so forth)
- Dynamic content: Basic CGIs written in Phyton, Perl, PHP, or C; Java-based components like servlets, JSPs, EJBs; and Microsoft proprietary ASPs

In a migration these components will have to be moved to the Web servers running on Linux. Should the customer be using Active Server Pages (ASPs) from Microsoft the Web serving scenario is somewhat more complicated. ASPs can be converted to run using Java. There is software available  from Halcyon that will convert the code to run under Java. For a complete discussion contact Halcyon directly at:

`www.halcyonsoft.com`

One area where easy conversion is problematic is when the ASPs point to a database. If the database is either Oracle or DB2, the databases can be moved to Linux zSeries and run directly. If the customer is using SQL Server there are several choices: leaving the database on the existing Microsoft-based server, or moving the database to either Oracle or UDB.

If the customer has written their own DLLs that are called by the ASPs, the code has to be ported to Linux. Most DLLs are written in either C or C++ and would need to be ported and tested. As a second option there are consultant services, including those from IBM Global Services, that can assist in porting the DLLs to the Linux/Apache platform.

The ultimate direction should be to move to Java Server Pages (JSPs), which offers a platform-agnostic approach.

## Example environment

Here is the mapping of an environment based on Sun rack-mounted servers doing intranet Web serving to a Linux Web server farm running under z/VM on zSeries. This can be used as a skeleton to understand how a real-life server farm will map to a Linux on zSeries environment.

Technical assumptions: Some of the workload will be static HTML, which is mostly I/O, and some dynamic HTML generated by PHP, Java/JSP, or Perl, which will be mix of CPU and I/O. For utilization, we assume most of these Web servers run at around 10% CPU.

### UNIX environment: 50 Intranet Web servers

- Hardware
  - 50 Sun 220R system: 2x Ultrasparc II 300 MHz, 512 Mb RAM, 2 FastEthernet cards, 18 GB disk
- Power
  - 50 x 300W = 15 KW
- Space
  - 5x 19" racks floor space
- Software
  - Solaris is included with the HW
- Network
  - 5 FastEthernet Network switches plus cables
- People
  - 4 system administrators

### Linux for zSeries Environment: 50 Linux instances running under z/VM

- Hardware
  - 4 IFLs, 16 GB RAM, 1 GB Ethernet adapter and 512 GB disk space
- Power
  - None additional, add to existing zSeries and ESS systems
- Space
  - None additional
- Software
  - Cost for a Linux distribution (For example, Debian is free.)
  - z/VM for 4 IFLs
- Network
  - 1 shared GB Ethernet adapters, already mentioned in hardware
- People

  – 3 system administrators

## TCO how-to

This section provides an introduction to total cost of ownership, and describes how this can be calculated for a server farm and its mapping to a Linux for zSeries farm.

The components that make up TCO are:

- Hardware:
  - Servers
  - Disk
  - Network
  - System Management
  - Racks (+cable)

  Hardware components will vary greatly between architectures being compared for solving the same customer problem. It is important to keep in mind that in a server consolidation model, lowering total cost while maintaining specific performance/service levels are the main objectives. Due to differences in architecture and performance (TPC or MIPS) numbers between architectures, it is important to correctly size the comparative workloads.

- Software
  - Operating system
  - Linux SW support
  - System management
  - Database
  - Application

  Software costs make up a very large component in the TCO model. In a distributed model *most* of the ISV software cost is based on the number of processors (CPUs), independent of whether the machine is running at 3% or 100%. Depending on the Linux for zSeries based pricing, there can often be savings in ISV software.

- People
  - Full-time equivalents

  People costs need to be examined carefully to make sure that the cost per person and the ratio of IT staff per server reflects the customer situation. There is a lot of geographic adjustment that needs to be made to this variable since people costs tend to differ greatly. It is frequently a very sensitive area when working with data.

- Occupancy
  - Area
  - Utilities

  There is a lot of variability in floor space and utility costs in different geographies. When looking at floor space costs, consider whether there are constrains to the floor space. If yes, acquiring more space will probably have a very high initial cost. This large additional cost can often weigh heavily in the total cost analysis.

- Migration

  Migration costs are costs attributed to moving from the existing environment to the new proposed environment.

- Downtime

Downtime is an aggregated number based on expected downtime of each of the different architectures, and based on industry numbers for the cost of downtime. Cost of downtime numbers should be examined as a means of demonstrating the degree of customer satisfaction issues that may occur within the different architectures. Downtime numbers are valuable for highlighting the hidden costs; however, rarely is cost of downtime presented in the final TCO analysis.

There are several TCO advantages to consolidating distributed servers with Linux on zSeries:

- Distributed servers requires many discrete servers: in even the smallest environment there is a test, a production and a backup server. With zSeries these can all be virtualized.
- Security can be enhanced by protecting the Kernel and most of the binaries on a z/VM read-only minidisk.
- Administrative costs of managing the cloned and shared environment are lower because fewer personnel are needed than is the case for discrete servers.
- Upgrades to the server environment are easier to implement, again through cloning and sharing of environments.
- A company that is out of floor space will have to build a new data center if they continue along the current distributed path. This can be avoided with server consolidation.
- Large software charges from distributed software vendors can be avoided.
- Disaster recovery plans can be improved.
- Excess capacity on an existing mainframe can be utilized.

### A sample TCO using the CIO View tool

The CIO View tool was developed over the past year to help customers analyze their total cost alternatives. The cost numbers generated by the tool are based on industry averages. In the best of all worlds the customer can work in conjunction with IBM and/or business partners to refine the overall analysis. Generally in working together there are factors that can show up which will have a large impact on the overall decision.

### Accessing the CIO View tool

The CIO View tool can be purchased directly from IDC. To get more information and to see some sample TCO information go to:

`www.cioview.com`

IBMers and selected business partners also have access to the CIO view tool, and can work with customers to perform an assessment. More information can also be found at:

`www-1.ibm.com/servers/eserver/zseries/campaigns/z800tco_tool.html`

## Summary

When considering the consolidation of a server farm, it is important to understand how the existing farm can be mapped to the Linux for zSeries environment, the pros and cons of a full versus a virtual consolidation, and the advantages of a cloned and shared environment with Linux under z/VM versus a distributed environment.

In most business environments, some good candidates for server consolidation are:

- File/print servers
- Database servers

- Web servers
- Network infrastructure
- E-mail

These servers represent about 60% of the existing server base. In addition to the TCO advantage of a consolidation, there are benefits in the areas of disaster recovery, security, rapid deployment of new servers, and reduced project implementation time.

## References

IBM Redbook: *Linux on IBM eServer zSeries and S/390: ISP/ASP Solutions*
http://publib-b.boulder.ibm.com/Redbooks.nsf/9445fa5b416f6e32852569ae006bb65f/33fa4bee2e91435585256ac600602f20?OpenDocument&Highlight=0,ISP%2FASP

IBM Scorpion Study
http://www.ibm.com/servers/eserver/zseries/library/whitepapers/scorpion.html

“Cloning Linux Images in z/VM” by Tung-Sing Chong
http://www.vm.ibm.com/devpages/chongts/tscdemo.html

Standard Performance Evaluation Corporation, SpecWeb99
http://www.spec.org/osg/web99/

SpecWeb99 benchmark study
http://www.spec.org/osg/web99/

CIO View tool
http://www.cioview.com/linux/IBMPreView/

IBM Redbook: Samba Installation, Configuration, and Sizing Guide:
http://publib-b.boulder.ibm.com/Redbooks.nsf/9445fa5b416f6e32852569ae006bb65f/7fb18fa101b54b328625688500562954?OpenDocument&Highlight=0,samba

Samba homepage
http://www.samba.org

SuSE Homepage
http://www.suse.com

Debian Homepage
http://www.debian.org

CPint tool written by Neale Ferguson
http://linuxvm.org/penguinvm/programs/

Halycon Homepage
www.halcyonsoft.com

## The team that wrote this paper

**Erich Amrehn** is a certified Senior IT Specialist at the EMEA Technical Marketing Competence Center (TMCC), Boeblingen, Germany. Before joining the TMCC, he worked as a project leader at the  International Technical Support Organization, Poughkeepsie Center. During that time, he wrote redbooks and taught Linux topics worldwide. Before joining the

ITSO in 1998, he worked as a technical consultant to the IBM System/390 division for e-commerce on S/390 in Europe, the Middle East, and Africa. He also has 13 years of VM experience in various technical positions in Germany and other areas in Europe and worldwide.

**Joachim Jordan** is an IT architect at the IBM Technical Marketing Competence Center(TMCC), Boeblingen, Germany. He joining the IBM lab in 1997, and spent several years in z/OS development. He then moved to the TMCC, working on customer projects focused on z/OS, UNIX System Services, and WebSphere. Since 2000, his focus has been on Linux and OSS.

**Frank Kirschner** is a consultant and IT specialist for the trustsec IT solutions GmbH (Germany). He has worked with Linux since 1992, and is currently working for the IBM Technical Marketing Competence Center (TMCC) in Boeblingen(Germany). He is one of the maintainers of the Linux Debian distribution port for zSeries. His experience also includes security, OSS and programming in Java, Perl and C.

**Bill Reeder** is a Certified Specialist in eServer and Linux Consolidation. He is a lead on the Americas Team for zSeries. During his five years with IBM, Bill helped to create the ALIGN server consolidation methodology and has helped customers do total cost analysis and design cost effective solutions. For the past two years Bill has focussed on Linux solutions for consolidating real-world business workloads onto IBM eServers.

# Notices

This document created or updated on July 11, 2002.

Send us your comments in one of the following ways:
- Use the online **Contact us** review redbook form found at:
  **ibm.com**/redbooks
- Send your comments in an Internet note to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ  Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

## Trademarks