Herding Clones

Mike Kershaw
Michael.Kershaw@marist.edu

August 17, 2004

Why?

- Computer Science department wanted to offer students their own servers for classwork which would be available for the entire duration of their academic career
- IBM Joint Study to use virtual servers in an academic setting

Requirements for Clones

- Very large number of Linux guests on z/VM (currently over 500)
  - Too many to create manually
  - Long life-cycle before guests are decommissioned - Up to 5 years life cycle for each server
  - Difficult to maintain kernel patches and services for large numbers of systems
- Must be flexible and modifiable by users
  - Used by students and faculty for purposes not foreseen in the original design
  - Must be able to push security updates to the clones during

Requirements for Clones

use

- Should be user-friendly
  - Not all users will be experts
  - Basic functionality and configuration should be accessible by all without compromising the ability of advanced users

Finding out what doesn't work

- First created as fully stand-alone servers for specific courses
- This design had many drawbacks
  - Could not maintain servers or fix problems
    - No mechanism for security updates
    - No mechanism for network changes
    - No method to recover forgotten passwords - Not linked to central LDAP system
  - Could not give users new software
  - Required Linux/Unix skills to configure or modify services

Finding out what doesn't work

- Required a large amount of disk space for each server - entire root FS duplicated needlessly
- Prior to guest LANs, point-to-point IUCV links were needed.
  * Requires 2 IPs per clone
  * No method to automatically allocate IPs and modify the VM TCP/IP config
  * Limited number of IUCV links per VM stack
  * Poor error recovery on IUCV - rebooting stack meant rebooting all attached linux systems

Designing the next phase

- New generation of servers designed with help from Sine Nomine consultant, Scott Courtney
- Shared root filesystem
  - Greatly reduced impact on drive space - 2.5GB shared between all systems
  - Single point of maintenance and upgrades
  - LDAP access control
  - "Known Good" configuration rollback point no matter what users change
- z/VM Guest LAN connection allows automatic creation and

Designing the next phase

hundreds of guests per LAN.

- User accounts and passwords controlled by LDAP (PAM-LDAP and NSS-LDAP)
- Users able to install their own software to writeable partition
- Central management server
  - Web-based management of user servers
  - Central database of allocated systems and IPs
  - Service monitoring and network services for clone machines

Dividing up the filesystem

- Start with a standard install with a writeable root partition
- Identify components a user would likely need to modify, and relocate the files (and optionally recompile the binaries to point to a different configuration location)
  - `/usr/local` - Contains the entire writeable filesystem, in accordance with the general guideline of installing non-system software in the local path.
  - `/home` and `/root` - Obvious first candidate, users need to be able to write to their own directories
  - `/etc` - Some configuration files need to be written to, but the bulk of the directory should remain readonly. Notable exceptions are `hosts`, `ld.so.conf` and `ld.so.cache`, which

are relocated to `/usr/local/etc`, and `mtab` which is symlinked to `/proc/mounts`

- `/tmp` - Relocated for obvious reasons. Could have been made into a shmfs/tmpfs filesystem, but memory is at a premium.

- `/var` - Let the users have syslogs

- `/dev` - A special case. Device info is needed during the initial startup, so it cannot be symlinked, but it must also be writeable. During the boot procedure, a tiny loopback filesystem from the user partition is mounted over `/dev`. This is a legacy from the 2.2.x kernel, and could be replaced with a mount-at-boot devfs 2.4.x or 2.6.x kernel.

Organizing the read-only components

- Configure PAM-LDAP and NSS-LDAP so that user maintenance is remote
- Relocate shared services to `/usr/share` for ease of maintenance
- Reconfigure or recompile shared services to refer to `/usr/local` for configuration files
- Easiest method for making maintainable services on the user segment of the filesystem is to use symlinks to replicate the directory from the shared filesystem. (See next slide)

An example service: Apache

The complete Apache directory is created in the user segment with symlinks to the shared read-only components:

```
/usr/local/apache/bin -> /usr/share/apache/bin/
/usr/local/apache/cgi-bin
/usr/local/apache/conf
/usr/local/apache/htdocs
/usr/local/apache/icons
/usr/local/apache/include -> /usr/share/apache/include/
/usr/local/apache/libexec -> /usr/share/apache/libexec/
/usr/local/apache/logs -> /var/log/apache/logs/
```

User segment configuration

- Classes may require students to reconfigure services or run different versions than those provided
- All user services configs are stored in the writeable portion
- `/usr/local/etc/rc.d/rc.local` allows users to add their own services or override existing services
- Per-host unique SSH keys generated on the first boot and stored in /usr/local/etc/
- Networking information is stored in virtual TAGs and extracted with `hcp TAG QUERY`

CMS user configuration

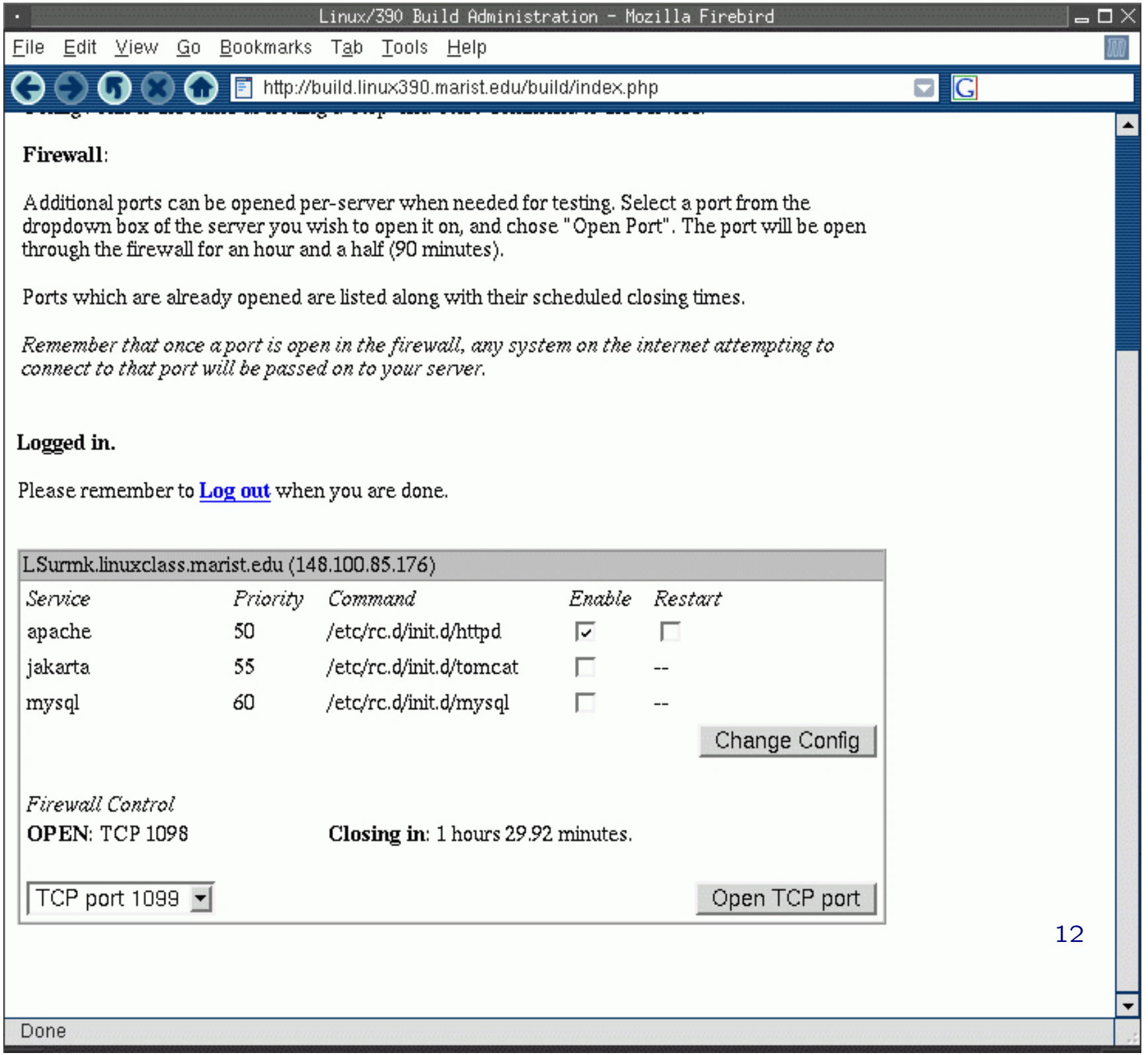
- Users are not allowed to log into CMS for their virtual machines
- All servers share a common 191 minidisk with the configuration files and `profile exec`
- `profile exec` queries user name on boot, and loads the config file defining owner, any custom init parameters, and network status

Managing virtual servers

- Customized login shell accepts service, shutdown, and restart commands sent to the console
- Able to control what pre-defined services boot. For example, apache is configured to load on all new servers, but jakarta and mysql are not.
- Able to restart a service without connecting to the server
- Able to open pre-defined ports in the firewall to their server only
- In the future, users will be able to request servers via the website

Managing virtual servers

- Web interface on the main server lets users control their servers


Linux/390 Build Administration - Mozilla Firebird
File Edit View Go Bookmarks Tab Tools Help
http://build.linux390.marist.edu/build/index.php
Firewall:
Additional ports can be opened per-server when needed for testing. Select a port from the dropdown box of the server you wish to open it on, and chose "Open Port". The port will be open through the firewall for an hour and a half (90 minutes).
Ports which are already opened are listed along with their scheduled closing times.
Remember that once a port is open in the firewall, any system on the internet attempting to connect to that port will be passed on to your server.
Logged in.
Please remember to Log out when you are done.
LSurmk.linuxclass.marist.edu (148.100.85.176)
Service Priority Command Enable Restart
apache 50 /etc/rc.d/init.d/httpd
jakarta 55 /etc/rc.d/init.d/tomcat --
mysql 60 /etc/rc.d/init.d/mysql --
Change Config
Firewall Control
OPEN: TCP 1098
Closing in: 1 hours 29.92 minutes.
TCP port 1099
Open TCP port
Done

Network access control

- Central server acts as router and DNS server for all user machines
- OSPF routing and IPtables firewalling to restrict ports to be consistent with our site policies for network access

User-Controlled services

- Users must be able to control services without being experts at Linux.

- Service control is through a seemingly complicated series of scripts and service machines, but has always worked reliably:
  1. PHP modifies the service state database to reflect the request if server is being changed to always up or always down

  2. PHP writes the request to a transaction file as `USER SERVICENAME UP|DOWN|RESTART`

  3. Privileged daemon uses `hcp` to forward the request as a message to a `PROP` service machine

User-Controlled services

4. `PROP` validates the destination, and uses `CP SEND` to write the command to the users console

5. Modified login shell accepts the command, queries the central MySQL database for the location of the service scripts, and runs them

User-Controlled services

- This convoluted path is necessary to preserve the security of the system:
  1. Users cannot directly issue `hcp` commands
  2. Only the central control server can issue a service command
  3. The central control server should not have VM privileges high enough to write to arbitrary consoles
- Latency of the complete system from web-click to service being changed is less than a second despite the many layers

Timed firewalls

- Security vs. Useability
- Users will need to open ports for special services, ie, classes writing java servers
- Ports can be opened in the firewall in hour-and-a-half durations
- Timed firewall ports fulfill two requirements:
  1. Increased security by limiting exposure time
  2. Increased user awareness of security by forcing users to manually chose to open a port
- Timed firewall entries are placed in mysql with user, end

Timed firewalls

time, and port. MySQL is overkill, but it's already running

Creating machines

This procedure will be replaced in the future with an automated system based off LDAP and the web interface.

1. Perl code on the controller accepts the user account and number of total systems requested
2. If the user already has as many machines as were requested, generation is aborted
3. Allocation script generates guest machine account names and creates database entries with the network information, ownership, etc
4. Created machines are written to transaction file
5. Transaction file is FTPd to VM

Creating machines

6. Set of REXX scripts are run against the transaction file to perform DIRMAINT, disk copy, guest LAN grants, and IPL

Starting and stopping machines

- Controller can request autolog and shutdown via messages to service machine
- Servers all run custom login shell which accepts a HALT command via the console – no root password needed for service machine to start a shutdown
- On IPL, controller begins a staged IPL of all servers in the database flagged to boot automatically. Controller IPLs 15 at a time with a 30 second delay to minimize contention for the 191 shared minidisk
- When the controller is issued a HALT command by the operators, it begins a staged shutdown of all the virtual machines, again spaced to prevent massive contention for swap IO: 500

servers with 64 meg of storage each can cause VM to stop responding for 10-15 minutes if they are all logged off at once

- Predates signal shutdown, but custom login shell provides a number of additional commands which are not available with shutdown signals

Monitoring user servers

The central database tracks all the services which should be running on the student servers. A perl script running on the central server monitors and responds to:

- Unresponsive servers. After a server has been unresponsive for 5 minutes, a halt is sent to the console, and if it continues to be unresponsive, a `CP FORCE` logoff is sent

- Down servers. If a server is down, it will be restarted

- Down services. If a service is down, the service will be restarted via the same mechanism the web page administration system uses

Creating servers on demand

In order to create servers quickly on demand in the future:

1. A pool of unassigned servers is kept in nolog status
2. Assign is carried out as normal to modify the controller database
3. A message is sent to a service machine to assign an empty server and IPL it – at this point, the server is available and running for the user, with no more delay than a `DIRM CHNGID`
4. In the background, the service machine repopulates the pool of empty servers via FlashCopy or DDR

Making secure web interfaces

Not directly related to the clone systems, but an important facet: How do you securely issue commands that require root privs? (ie, `hcp`).

- Running apache as root is BAD
- Giving apache access to run root-commands like `hcp` is also very bad.
- Directly handing user input, even sanitized user input, is usually bad.

So what can be done?

- Transaction files
- PHP writes the command to a transaction file, ie: `SERVICE`

Making secure web interfaces

```
LSURMK apache START
```

- A privileged daemon (which could be as simple as a bash script) detects the change either real-time via tail or at regular intervals

- The daemon parses the command - once removed from user input, plus a fixed "language" for commands is easy to validated.

- The daemon can then execute the command as root, with minimal delay, securely removed from an unprivileged webserver

Contact

- Email: `Michael.Kershaw@marist.edu`
- Download: `http://reason.marist.edu/~urmk/`