System Installation Suite
Massive Linux Installation

Sean Dague
japh@us.ibm.com / sean@dague.net
Software Engineer
IBM Linux Technology Center

A long time ago...

- Spring 2000
- Source Forge was just announced by then VA Linux Systems (Jan 2000)
- Red Hat 6.2 released (March 2000)
- XFree86 4.0 released (March 2000)
- Linux Clusters are few (Scyld Beowulf 2 is not released until Summer)

A niche to be filled... Massive Installation

- Red Hat had an automated install process in kickstart, but many other distributions did not.
- LUI - Linux Utility for Cluster Installation v1.0 (April 2000)
  - Rich Ferri, IBM
  - Originally Red Hat only
- SystemImager v1.0 (May 2000)
  - Brian Finley, VA Linux Systems
  - Originally Red Hat and Debian only

LUI - Linux Utility for Cluster Installation

- Conceptually based on the AIX SP installation tool NIM (Network Install Manager)
- Resource Based
  - Resources provide a description of a completed machine
  - Resources included things like: RPM list, Disk table, Post install actions
  - Resources and Resource descriptions stored in a Database
- Network Based
  - Machines must be network booted (use PXE hardware or emulate with Etherboot)
  - TFTP transfers kernel over the network
  - NFS Root and USR on Resource server used to drive installation

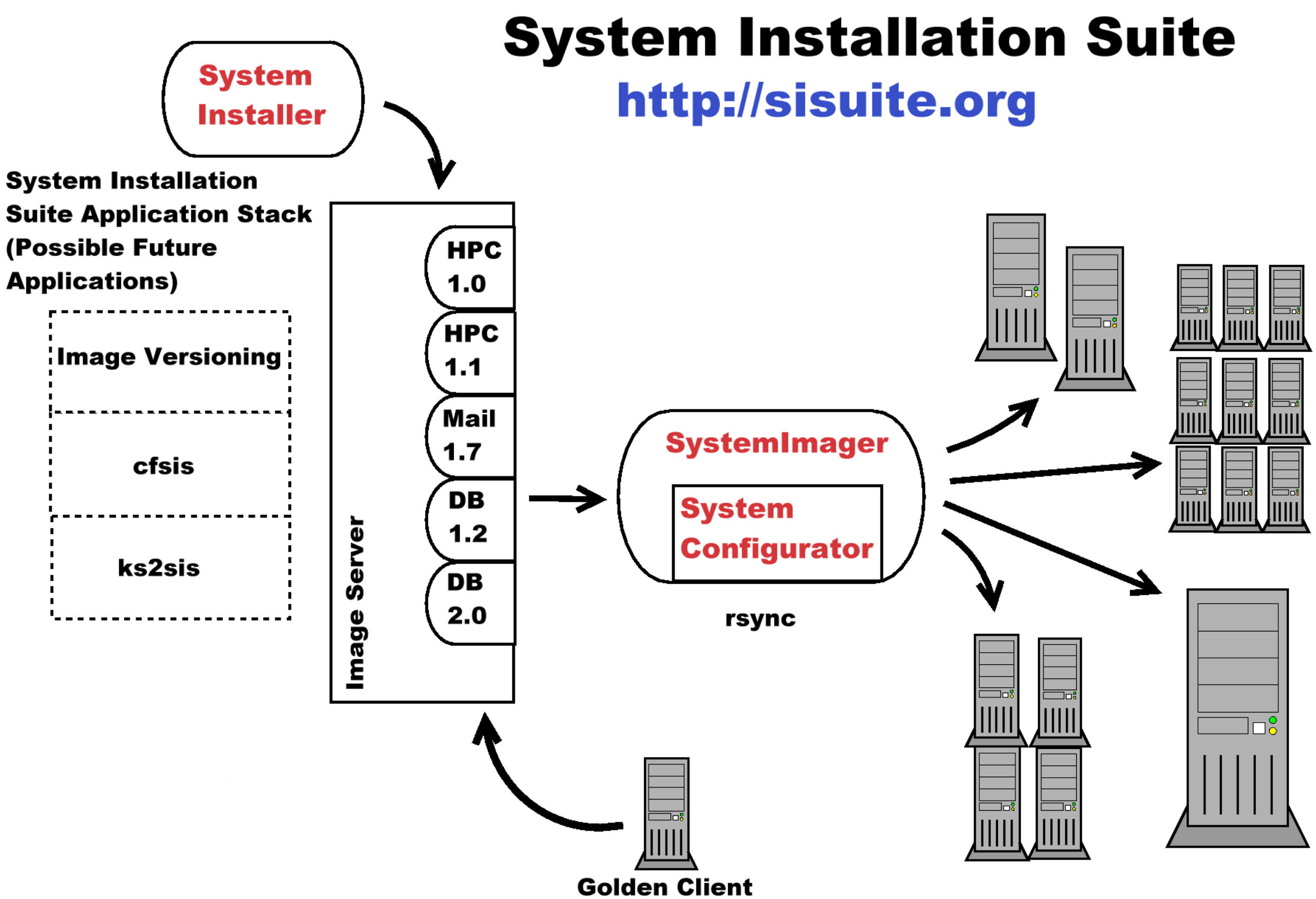
SystemImager

- Image Based
  - Images captured from a **golden client**
  - All images stored on a central server
  - Image is an complete copy of all disks on client
- Linux Based Autoinstall System
  - Autoinstall can be floppy, cd, hard drive, or network driven
  - Kernel and Initial Ramdisk drives install
  - Custom installation variables can be passed via local.cfg on floppy disk or extra options with dhcp server

System Installation Suite

- Best of both worlds
- LUI evolves into SystemInstaller - resource based image generator
- SystemImager infrastructure remains to transfer images
- System Configurator configuration API added

System Installation Suite at a glance



Rsync - my favorite protocol

- rsync is a file transfer protocol which transfers only the differences between source and target
- rsync can be secured with ssh
- the rsync process can be daemonized to give other programs access to files
- supports **modules**, symbolic names for directories of files

What is a SIS image?

- A SIS image is a directory that contains the entire contents of the root (and other) file systems of a live machine
- Stored in /var/lib/systemimager/images/$NAME
- Exported as an rsync module - $NAME
- Because it is a full file system, you can **chroot** into the image and do things like:
  - compile code (using the native tools in the image)
  - update packages
  - even run graphical programs from the image
- With SIS, images aren't as unchangeable as other imaging tools

Fetching an Image

- Very similar to SystemImager pre-SIS
- Install **golden client** however you feel like
- client must be “prepared”
  - **prepareclient** command runs on the client
  - captures all the partition information to files in /etc/systemimager
  - starts an rsync daemon which has an rsync module 'root'
- Then image is captured
  - **getimage** command runs on server
  - connects to client via rsync
  - rsyncs the contents of rsync://golden-client/root to /var/lib/systemimager/images/imagename
  - calls **mkautoinstallscript** to build the autoinstall script

Building an Image

- SystemInstaller builds images
- mksiimage command
  - takes a number of parameters including: package list, packages location
  - currently only works with rpms
  - determines “best” version to install of any package (highest version, best compatible architecture)
  - does the equivalent of
    rpm -ihv -r /var/lib/systemimager/images/imagename ...

Building an Image - Rpm fun

- Why this mostly works
  - **rpm** reorders packages during installation based on prereqs
- Scenario (MDK 8.2)
  - Package requirements
    - bash requires ld-linux.so.2 - provided by glibc
    - glibc requires /sbin/sash - provided by sash
    - sash requires nothing
  - user command: **rpm -ihv bash.rpm glibc.rpm sash.rpm**
  - effective command: **rpm -ihv sash.rpm glibc.rpm bash.rpm**
- Policy dictates that rpms are not allowed to have interactive post scripts, which helps a lot

Building an Image - Issues

- Straight rpm is too simple an approach most times
- Not all packages are made to install from ground up
- Most distribution installers have policy outside of rpm prereqs to complete initial installation.
- Many packages forget to include programs used in %post scripts in their prereq list (you can't use egrep if its not there)
- It is really hard to create 1500 packages (average distribution) that all have exactly the right prereqs.

Building an Image - Staging

- SystemInstaller has the ability to create stages where certain packages will be installed with different options
- Distro name and version are detected from the rpm pool, and appropriate default stages are used
- Scenario: (SuSE 7.2)
  - Stage 1 Flags: --nodeps --noscripts --noorder
  - Stage 1 Packages: aaa_dir aaa_skel bash db glibc sh-utils readline ncurses fileutils
  - Stage 2 Flags: --nodeps --noorder
  - Stage 2 Packages: textutils base grep diffutils aaa_base devs perl termcap
  - Then all other packages installed as before

Building an Image - Post build

- In many cases other post installs have to be run
- Because we prevented some early scripts from running, we may have to make up for it now
- Programs we know we need to run:
  - Mandrake - msec
  - SuSE - SuSEconfig

Building an Image - Adding Disks

- mksidisk command creates partition scheme info
- takes an fstab like input file and creates /etc/fstab and partitionschemes files in the image
- supports nfs file systems as well as normal ext2, swap, etc.
- mkautoinstallscript must be rerun after a disk change

Defining Clients - Addclients

- addclients is the SystemImager command for adding client definitions
- Accepts ranges of client names and builds definitions for them
- What it does:
  - modifies /etc/hosts and copy in [scripts]
  - adds symbolic link from hostname.sh to install script
- Can be run via command line or interactively

Defining Clients

Mksimachine / Mksirange

- mksimachine / mksirange are the SystemInstaller command to add client definitions
- mksirange uses addclients to do real work
- Also adds client definitions to SIS DB
- mksimachine used to list, delete, or modify clients

Auto Installation - BOEL

- Autoinstall is driven by BOEL (Brian's Own Embedded Linux)
- Consists of a special kernel and ramdisk
- Floppy, CD, Hard Drive and Network boot supported
- Because of the floppy constraint, space is extremely limited
- BOEL is Linux, so theoretically any fs type that Linux supported could be used to deploy an image (ext2, ext3, and reiserfs currently supported)

Building BOEL - Kernel

- 1.44 Meg is a seriously constrained environment
- ~780k for kernel, ~630k for initrd.gz, rest for syslinux
- kernel is a monolithic version 2.2.18
  - contains many popular network drivers
  - many popular scsi drivers
  - ext2, ext3, msdos, reiserfs, and raid support
- Kernel can be rebuilt to support other hardware

Building BOEL - Initrd.gz

- Required elements:
  glibc, busybox, rsync, mke2fs, ash, dhclient
- glibc 2.x vanilla too large for this environment
- glibc-pic used instead
  - PIC - Position independent Code
  - Allows creation of small libc with only the used parts of glibc
  - mklibs.sh from Debian used to do this
- Unfortunately, glibc-pic only shipped on Debian, so build system is currently tied to Debian.
- All other utilities (mkraid, mkreiserfs, etc) stored on the server and rsynced over on demand.

Autoinstall Process

BOEL in Control

- Kernel and Ramdisk loaded
- Autoinstall client brings up network and determines its hostname
    - local.cfg stored on floppy
    - dhclient and reverse dns
    - dhclient and server hosts file
- rsync hostname.sh from image server
    - local.cfg, dhcp server, or via options to dhcp server
- give control to hostname.sh

Autoinstall Process

Autoinstall Script

- Autoinstall Script
  - POSIX shell script run under a statically linked ash (Debian's)
  - generated by **mkautoinstallscript**
  - can be hand tuned by user
- Partitions the drives
- Formats the partitions
- Rsyncs the proper rsync module over to local disk
- Runs System Configurator
- Runs specified postinstall action

System Configurator

- Problem: image must be customized on client node to actually boot
  - Machine must boot
  - Machine must get on the network
  - Other configurations may need to happen
- Distributions have many different ways to configure the network
- Making a machine bootable is very architecture specific
- Want to export a simple API to SystemImager so that no logic must go into the autoinstall process to do this task

System Configurator

Possible Solution

- A possible solution!
- Build a huge matrix
- Map every distro, distro version, and arch to what they need
- Detect distro, distro version, and arch
- Do what the matrix tells you

System Configurator

Why the matrix solution sucks

- What happens when SuSE 8.1 comes out?
  - Mappings for 8.0
  - However 8.1 doesn't have any data in the matrix
- What about Debian Unstable, or Mandrake Cooker?
  - No real version on either of these
  - Both are constantly in flux, but often used
- What about all the minor distros which are subtle variations of Red Hat?

System Configurator

Footprinting

- A better way:
  - if /**etc**/**sysconfig**/**network-scripts** exists, generate **ifcfg-INTERFACE**
  - if /**sbin**/**lilo** and /**etc**/**lilo.conf** exist, run lilo
  - if /**etc**/**rc.config** exits, and /**etc**/**sysconfig**/**network** does not, put network variables in **rc.config**.
- Look for the underlying config files, if they exist, make them look right
- Support derivative distributions most of the time
- New versions of existing distributions will tend to be supported
- Footprints can grow organically over time

System Configurator

Chroot Trick

- System Configurator has a lot of logic in it
- Translates to a lot of code, none of which is POSIX shell.
- Its in Perl, which is too large to put on the boot media
- Going back to the server for Perl would suck
- How do we make it available to the autoinstall process?

System Configurator

Chroot Trick

- Easy, just put it in the image
- systemimager-client requires systemconfigurator (on a package level)
- Once the image is transferred to the client disk, it is fully chrootable
- Last command of autoinstall process is:

```
chroot /a/ /usr/bin/systemconfigurator --stdin --configsi <<EOF
[NETWORK]
...
EOF
```

- We get the full perl environment of the working machine to play with

System Configurator - step by step

- All stages are independently runnable
- --confighw
  - uses a PCI modules table generated from kudzu and harddrake database to relate PCI ids to kernel modules
  - modifies modules.conf (and related files) to match what it finds
- --confignet
  - creates the proper network scripts
  - also modifies resolv.conf if requested
- --configrd
  - sets up initrd if required
  - uses native initrd tools

System Configurator - step by step

- --configboot
  - Generates bootloader configuration files
  - Supports Lilo, Grub, Elilo, and EFI raw
- --runboot
  - runs the appropriate command to setup the bootloader
  - uses the on disk configuration file
- --configkeyboard (new)
  - Sets up keytables
- --configtime (new)
  - Sets up timezone

System Configurator - step by step

- --configuserexit (new)
  - allows for custom user exits
- --configsi == --confighw --confignet --runboot
- with each phase independent, user can turn of the things she doesn't need
- Example:
  - MSC.Linux using SIS to install Itanium 2 nodes
  - --confighw did the wrong thing for them
  - by changing --configsi to --confignet --runboot it works

Autoinstall finishes

- Postinstall runs
  - halt system
  - reboot system
  - beep forever (default)
- If anything goes wrong, autoinstall dumps to shell

Maintenance

- First, update the image
- Option 1: Apply patch to golden-client
  - Apply patch to running machine
  - make sure it works
  - rerun getimage on server
- Option 2: Modify image directly on the server
  - rpm -Uhv -r $imagedir updated.rpm
  - chroot $imagedir && apt-get update && apt-get upgrade
  - make sure it works

Maintenance

- Image on server, copy of image on client
- rsync modules still accessible from client
- **updateclient** - updates the client
  - runs on the client machine
  - uses an excludes file locally
  - can run bootloaders after invocation
  - can start a harddrive based autoinstall
- many directories excluded by default (/var/log, /tmp, etc)
- user must be very careful to adjust excludes to her environment

SIS in real life

- Oracle Linux Migration
  - Move Oracle's internal pap servers from number of Unix Servers to 200 - 300 Linux Intel servers
- OSCAR - Open Source Cluster Application Resource
  - Easy HPC Cluster building tool
  - Builds an installer around SIS
  - Over 20,000 downloads of 1.2.x (which uses SIS)
- CSM - Cluster Systems Management
  - IBM proprietary tool for cluster management
  - Will have SIS as an installation option in the near future

Future Directions

- Support More Filesystems
  - XFS
  - JFS
- Support Volume Management
  - LVM(2)
  - EVMS
- Modular Kernel coming soon!

Future Directions

- Multicast support
  - multicaster
  - mrsync
- Especially important for large clusters like Terra Grid and PNNL
- Should make install time O(log(n)) or so for large systems

Future Directions

- Ports, Ports, Ports
- SIS stable is x86 only
- SIS devel also supports IA64
- zSeries (S/390) port underway
- PPC (pSeries, iSeries) ports planned for this year
- PARISC (ask Dannf)?

Future Directions

- Full Support for non RPM based distros
- SystemImager and System Configurator work fine
- Debian
  - problem: packages can ask questions
  - Until debconf is available, it will be hard
  - using FAI might work
- Slackware
  - Looks like it should be easy
  - Needs SC work at the moment
- Others?

Future Directions

- More GUI work
- Tksis is the current GUI, but can only handle SystemInstaller commands
- Unified GUI for all commands, plus editors to tweak things like disk tables and packages

Future Directions

- Enterprise Features
- Robust remote logging
- Robust remote status
- Tighter integration of SIS components
- Assurance testing of images
  - possibly using UML on images
- Versioning Framework
  - being able to keep multiple versions of an image in a normalized way
  - applying a patch automatically builds a new version

Future Directions

- Non-Linux installation (really pie in the sky stuff)
- xBSD (Net, Open, Free)
  - POSIX like, so rsync will be happy
  - FS supported in Linux
  - Lots of people run large farms of xBSD clusters
- Possibly commercial UNIXes
- Possibly others
- Ground rules
  - Linux needs to be able to build the fs
  - Linux needs write access to the fs
  - The fs should have some level of POSIXness to it

Future Directions

- Non-Linux installation (really pie in the sky stuff)
- xBSD (Net, Open, Free)
  - POSIX like, so rsync will be happy
  - FS supported in Linux
  - Lots of people run large farms of xBSD clusters
- Possibly commercial UNIXes
- Possibly others
- Ground rules
  - Linux needs to be able to build the fs
  - Linux needs write access to the fs
  - The fs should have some level of POSIXness to it

The Mantra

- Do it once. Do it right. Do it for everyone.
- No one should have to reinvent massive installation again