CDF/DOC/ONLINE/PUBLIC/6693
November 2, 2003
Version 1.0

Level 3 Farm Administration Tools

N. Leonardo
Massachusetts Institute of Technology

Abstract

It is described sysadmin tools and procedures developed for maintenance, recovery and installation of the Level 3 computing farm.

Keywords: Online Linux farm, Maintenance, Recovery, Installation.

Foreword

The Level3 computing farm functions as a processor based filtering system as part of the data acquisition and trigger systems of CDF Run II.

It constitutes a computing resource formed of approx. 300, dual-processor computers under the Linux OS, continuously performing tasks of online event reconstruction and analysis.

This document contains the description of administrative tools put together with the purpose of effectively performing necessary actions over many farm nodes, such as those involved during rescue and upgrade. Farm and technical details are deliberately provided, making this a suitable guide to system administration of the Level3 farm.

It is based on general farm sysadmin and maintenance experience over the last couple of years, during which the tools have been developed and extensively used.

Contents

1 The Level 3 online computing system

1.1 The DAQ/Trigger subsystem

The Level3 system forms a critical component of the data acquisition and trigger systems of the Experiment.

A thorough description is presented in the EVB/L3 systems' Manual (CDF Note 6138) I have written some time ago [1].

1.2 Computing farm

The Level3 system is realized as a farm formed of approximately 300, dual-processor, Linux boxes.

Two of these — Gateway1 and Gateway2 — serve as proxies on behalf of the entire farm. All messages coming in and out of the system are relayed through the Gateways, which themselves do not process data directly.

The remaining are worker nodes, residing on a private network, through which data flows. These are organized in (16) subfarms, each formed of so-called Converter (1), Processors ($\sim$ 16), and Output (each serving two subfarms) nodes.

Gateway1

Runs the Event Builder proxy.

Contains two partitioned disks, employed for running and backup systems.

Partition table excerpt for local filesystems:

```
/dev/hda1            /boot
/dev/hda2            /
/dev/hda5            /tmp
/dev/hdb6            /home
/dev/hdb7            /cdf
```

The */home* filesystem is exported to the entire farm.

Level3 filter code and common products directories are mounted from external online machines.

Gateway2

Runs the Level3 proxy.

Contains two partitioned disks, employed for running and backup systems.

Partition table excerpt for local filesystems:

```
/dev/hda5            /
/dev/hda6            /tmp
/dev/hdc10           /cdf
```

Level3 filter code and common products directories are mounted from external online machines.

Converter

Contains ATM card and required modules.

Contains multi(4)-port Ethernet card and required modules.

Partition table excerpt for local filesystems:

```
/dev/hda1          /boot
/dev/hda2          /
/dev/hda4          /cdf
```

Processor

Partition table excerpt for local filesystems:

```
/dev/hda1          /boot
/dev/hda2          /
/dev/hda5          /tmp
/dev/hda6          /cdf
```

Output

Partition table excerpt for local filesystems:

```
/dev/hda1          /boot
/dev/hda2          /
/dev/hda5          /tmp
/dev/hda6          /cdf
```

Additional structure

/local

UPS/UPD [7] bootstrap are installed in this directory.

The `setups*` files are located in `/local/ups/etc/`; soft links to these setup files should exist on `/usr/local/etc/`, as e.g.

`/usr/local/etc/setups.sh` → `/local/ups/etc/setups.sh`

Individual product chain files should be located on the database directory:

- Gateways: `/cdf/products/upsdb/product_name`
- Worker nodes: `/local/ups/db/product_name`

/cdf/products

The products themselves are installed in this directory. Each product instance should contain a *ups* directory with corresponding table (where in particular the `setup` actions are defined), located as in

`/cdf/products/`*product_name*`/Linux+2.4/version/ups/`*product_name*`.table`

On the Gateways the following link is required (`/cdf` copied daily from Gateway 1 to 2):

`/usr/products` → `/cdf/products`.

Products `root` and `ooc` are installed on all farm nodes.

/cdf/level3

This directory contains the level3, relay, and executable code; the following structure is assumed by the code:

```
/cdf/level3/filter:
    bin
    calib
    cint
    control
    etc
    include
    lib
    relay
    tar
    tcl
```

/log

The following link for log files is assumed `/log` → `/cdf/log`

Products

UPD product install

Available UPD [7] products may be installed from Gateway1, using the command `upd install`. This requires passive ftp data transfer, `export FTP_PASSIVE=1`; the user should have `products` group permissions, check `/etc/group.nis`.

The following illustrates the steps necessary for upd product installation; An example of a `gcc` product instance is given.

`ssh` *user*`@b0l3pcom1`

the following should occur at shell invocation
`source /local/ups/etc/setups.sh`

`setup upd`

confirm local OS flavor `ups flavor`

check local and available distribution [7] product instances

ups list gcc

upd install gcc v3_0_1 -f Linux+2.4

UPS product creation

For creating an ups product the following guidelines are provided:

place the source code in appropriate location
`/cdf/products/`*product_name*`/Linux+2.4/`*product_version*`/`

create ups directory containing product table file as in
`/cdf/products/`*product_name*`/Linux+2.4/`*product_version*`/ups/`*product_name*`.table`

create version file in database; e.g. in Gateway1
`/cdf/products/upsdb/`*product_name*`/`*product_version*`.version`

update current chain file in database; e.g. in Gateway1
`/cdf/products/upsdb/`*product_name*`/current.chain`

Alternatively, product declaration may be done automatically by issuing ups commands. The following exemplifies this for an instance of the product `root`.

```
> cd /cdf/products/root/Linux+2.4
> mkdir v3_05_07
> cp -r [my_private_code_location]/* v3_05_07
> ups declare root v3_05_07                    \
    -f Linux+2.4                               \
    -z /cdf/products/upsdb                     \
    -r /cdf/products/root/Linux+2.4/v3_05_07   \
    -m root.table                              \
    -U $UPS_DIR                                \
    -M /cdf/products/root/Linux+2.4/v3_05_07/ups
> ups declare root v3_05_07 -g current
    -f Linux+2.4
    -z /cdf/products/upsdb
> setup root
> which root
  /usr/products/root/Linux+2.4/v3_05_07/bin/root
```

The following exemplifies this instance version file of product `root`.

```
FILE = version
PRODUCT = root
VERSION = v3_05_07
FLAVOR = Linux+2.4
```

```
QUALIFIERS = ""
  DECLARER = leonardo
  DECLARED = 2003-10-17 22.33.44 GMT
  PROD_DIR = /usr/products/root/Linux+2.4/v3_05_07
  UPS_DIR  = /local/ups/prd/ups/v4_6_3/Linux-2
  TABLE_DIR  = /usr/products/root/Linux+2.4/v3_05_07/ups
  TABLE_FILE = root.table
```

The same product instance may be undeclared issuing the command

`ups undeclare root v3_05_07 -z /usr/products/upsdb -f Linux+2.4`

REMARKS

Booting sequence

When bringing up the full farm the following booting sequence should be respected to avoid complications:

b0l3gate1

b0l3gate2

b0l3boot-vx (a.k.a. b0l3backup)

Ethernet switches

Converter, Processor, Output nodes

1.3 Maintenance & Administration

Continuous and attentive administration of the Level3 farm is a true necessity for ensuring its optimal performance, which is expected. It can certainly become an absorbing activity when perfomed by a single individual.

The farm has been functioning continuously with its full size for a couple of years now. Several tools have been developed to deal efficiently with maintenance requirements and other necessary sysadmin actions, on possibly many farm nodes.

These tools have been extensively used since, for both fixing and preventing failures which would otherwise disrupt the optimal functioning of the system, and consequently of data taking by the experiment. In spite of a few serious, large scale problems [1] diligent maintenance efforts have made them of no consequence for the best performance of the system.

Tools most convenient for — otherwise rather challenging — system operations to be performed on possibly many farm nodes, such as regular filesystem recovery, operating system upgrades, new nodes installation, exist, having been extensively tested, for use on the Level3 farm.

[1] At some point a persistent hardware failure was diagnosed in as many as half of the nodes in the farm; the problem was diligently handled through assiduous maintenance, and has allowed notwithstanding for optimal farm operation.

2 Recovery

Investigation and recovery from problems on individual and many nodes is achieved through extensive use of properly customized *L3 recovery floppies.*

These form convenient tools for executing required actions on nodes which fail to boot healthily. These actions can be instructed on a terminal connected to the node, or specified in advance on a file at a remote server.

Common actions include filesystem and code installation (typically necessary on a large number of nodes); hard drive inspection, partioning, filesystem creation, master boot record repairing; modifying boot loader and system configuration files. Other general, system supported, user specified actions are readily implementable as well.

2.1 Boot image

The recovery floppy is formed of a configurable boot image. This should contain a small Linux kernel with the proper drivers to boot the system in order to perform recovery operations, allowing as well customization in a simple fashion. The farm recovery procedure described here is general and can be in principle implemented using any such, rescue-oriented, available Linux image.

The particular rescue image which has been employed is the **tomsrtbt** disk image [3], described as a "floppy which has a root filesystem and is also bootable". This contains a very small distribution of Linux, with most commands useful for system recovery, most necessary drivers, and network connectivity capabilities, fitting a single floppy disk. The most attractive feature nevertheless is that the distribution comes with a number of scripts which can be used to readily create and transfer a modified image.

It contains on the other hand some objectionable features as well according to farm's requirements. It does not recognize the multi-port Ethernet cards used in Converter and Output nodes; this has required the use for recovery purposes of the on-board port. The current version does not contain the `sfdisk` Linux command, which would have otherwise been specially useful (it would facilitate the typical process of hard-drive partitioning+installation).

Also bothersome was the necessity of explicit login (requiring direct name/password typing at the console); this was fixed by compiling an appropriately modified `sulogin.c` code (where `getpasswd()` has been removed), enforcing static linking, and transferring it to the image.

Installation

An appropriate tar ball containing the *tomsrtbt* image can be downloaded from one of the listed [3] sites. The accompanying script `install.s` allows to transfer the raw image to a floppy disk; `fdformat` should be executed on the floppy in case it is not properly formated already.

Image customization requires a few steps. Executing `unpack.s tomsrtbt.raw` creates an expanded version of the raw image structure. Files located in the directory `tomsrtbt-`*version*`.unpacked/2/` can be modified as desired. An accordingly customized image can then be produced and installed simply by executing the scripts `buildit.s` and `install.s`, respectively, from the unpacked directory.

The Level3 customization presented is general, its dependence on image releases being minimal; although, extra functionality may become available.

2.2 *Remote* procedure

The raw rescue system is not adapted to the farm imperative of simultaneous recovery of many computers. A procedure was therefore developed allowing fast recovery of a large number of nodes in the farm, in a shortest time period – crucial for having the full farm readily operational, and not disturbing data taking. Indeed, it has allowed for safe, full-scale farm shutdowns and rebootings, with the guaranty of a reasonably short time for recovery of possibly many (typical case) failing nodes.

To decrease the time of an individual recovery operation, the original login program is replaced by a modified one wherein that process is made automatic; sleeping times, reserved for other systems' or interactive usages, are reduced or simply eliminated.

To cope with the necessity of dealing effectively with many nodes, it is essential to make the recovery procedure completely automated, whith no need for user's direct input. This is achieved by adapting the recovery system's configuration files so the desired actions are performed at the very first stages of system initialisation, while other, default actions are not executed if not necessary.

A simplified initalization table (`/etc/inittab`, the configuration file to **init**) is defined, where the instruction `::sysinit:/etc/rc.S` is included. This instructs initial execution of `/etc/rc.S`, where some image configuration scripts are sourced, and at the end of which a call to a procedure execution script is *added*. The later can contain a set of intended actions on the node, ending for example with a `/sbin/reboot` instruction. The advantage of these features is the earliest execution of the recovery actions right at initialization level, and interruption of the default initialization process itself once the intended actions have been completed. Effectively, unnecessary, time consuming actions and configurations do not occur, by avoiding definition of common *run level*'s and their execution.

Furthermore, a *remote procedure* is put together where the procedure execution script is stored on a remote server, accessible via the network. This clearly favors generality and convenient procedure development. Indeed, once such an image has been created and transferred to a portable disk, the later can then be used quite generally, for execution of whatever actions are defined on a convenient script file residing on a remote, permanently accessible node.

This is achieved in practice as follows. A script named *fixnode.all.remote.floppy* is defined, and executed at the end of the file `/etc/rc.S`, whose function is to

1. configure the network,

2. mount an `nfs` accessible location, and

3. execute a remotely located script where user-specified actions are pre-defined.

Notice also that, if after a certain number of actions has taken place (e.g. network configuration) one wishes to still have access to a shell using a console directly connected to the node, this is made possible simply by including a call to a login program. This program can be e.g. the `sulogin` alluded to before, and can be included with the image or, e.g. for lack of available space on the floppy, simply placed on the remote location.

2.3 Recovering filesystems

2.3.1 Filesystem corruption

The recovery procedure here discribed was originally developped to cope with a persistent hardware failure affecting consistently a considerable fraction of the farm. It was verified that filesystems in those nodes were being progressively corrupted with time, eventually inducing their failure. In that case, it would then interrupt data taking, requiring an expert to arrive and remove it temporarily from the online hardware database.

No practical hardware solution to this problem was found, nevertheless its effects were surpassed through persistent maintenance efforts. These involved periodic (initially weekly) rebooting of the farm, enforcing checks of all filesystems every time, and systematic recovery of those shown defective.

Track was kept of the frequency of filesystem failures over the farm, both in a recovery, automatically generated log file and node statistics page [2]. This allowed for an assignment of a failure expectation to nodes in the farm, use of which has been made in preventive replacements and posterior hardware upgrades, as well as in progressive transfer of nodes with higher failure rates to less critical, offline usages.

2.3.2 Automating procedure

The remote recovery procedure is used for repairing damaged filesystems on the Level3 farm. An image is created containing the script *fixnode.all.remote.floppy*, which is directly executed at initilization time. Its description follows below.

configuring network

A set of `IP addresses` are *reserved* in the Level3 Ethernet network for use in the recovery process. These are explicitly mentioned in the `/etc/hosts` file of the server; e.g.

```
192.168.23.240        b0l3rec0.fnal.gov b0l3rec0
...
```

To each image floppy is assigned a distinct address among this set, which it uses for accessing the private network. The following set of instructions are sufficient for configuring a network interface.

```
ripadr=192.168.22.240
ifconfig eth0 $ripadr
ifconfig eth0 broadcast 192.168.255.255
ifconfig eth0 netmask 255.255.0.0
route add -net 192.168.0.0
```

nfs-mounting

A remote node should be available as `nfs` server, with a directory containing the necessary materials for implementing the recovery procedure.

Any available, *failure-free* Linux box in the farm can in principle be used as server for the recovery process. The directory containing the recovery materials would have to be `nfs` exported; this is done by creating/editing the file `/etc/exports` with contents as in

```
/recovery    192.168.0.0/255.255.0.0(rw,no_root_squash)
```

(where the IP address/netmask pair is used to avoid access control problems for `nfs` related daemons), and issuing the commands `rpc.`**nfsd** and `rpc.`**mountd** (or just **exportfs -a** in case `nfs` is already up and running).

In the Level3 farm Gateway1 has been used for this purpose. Recovery materials are located in `/home/recovery` directory (where the `/home` directory is permanently `nfs` exported to the entire farm). Its *remote* sub-directory can be mounted as follows.

```
mkdir /mnt/extdisk
mount -t nfs 192.168.24.11:/home/recovery/remote /mnt/extdisk
```

write permissions are kept for log purposes only, and are otherwise not required.

execute remote script

Having now successfully mounted the remote recovery directory, a call to a main script (*fixnode.all.remote*) is performed.

```
/mnt/extdisk/fixnode.all.remote
```

This will be expected to end with a `reboot` instruction. However, if that is not the case, or if the remote script is not found at all, a call to a login command may be used to allow direct console input.

```
/mnt/extdisk/bin/sulogin
```

This ends the image's residing script, *fixnode.all.remote.floppy*.

2.3.3 Installing filesystems

Filesystem recovery involves a number of steps. These are instructed in the main remote recovery script (*fixnode.all.remote*). This script, which resides on the remote recovery directory in Gateway1, reads in filesystem and node information from accompanying configuration files, recreates the specified filesystems by extracting their standard contents from available tarballs also located remotely in the server; it ends with a node rebooting instruction, after writing some log information as record of what was done.

read in recovery information

Berore executing the procedure, information needs to be specified regarding the filesystem to be recovered. This is done in a file (*recover.config-remote*) residing remotely (Gateway1). It may contain the name of the filesystem to be recreated, that of the corresponding partition, as well as the name and origin of an appropriate tarball; e.g.

```
hda7   cdf_260.tar     192.168.23.60   /cdf
```

re-create filesystem

The specified filesystem can be created using the `mke2fs` Linux program, destroying the previously existing corrupted one.

If a full filesystem check (`fsck`) is desired each time the machine boots, this can be forced by adjusting the maximal mounts count between two checks to unit; alternatively, the maximal time between two checks can also be imposed. This is done using `tune2fs`, and should be instructed at this stage, before mounting the filesystem.

expanding contents

The intended contents of the associated partition may be transferred to the node by expanding an appropriate tarball, once both the remote `tar` file location and the local partition are mounted on the image's recovery system.

The procedure expects the filesytems' tar files to be in a pre-defined location (namely, the `tarballs` subdirectory); such tarball can be produced from an equivalent healthy node as in the following example.

```
ssh root@b0l3260;  cd /cdf
tar -cvpf /home/recovery/tarballs/cdf_260.tar ./*
```

updating IP address

In case the root filesystem happens to be the one recovered, node unique configuration information, as its IP address, needs to be specified.

Firstly one extracts a node's hardware identification, as the Ethernet card's MAC (Media Access Control) address; this is displayed in the **ifconfig** output following the string `HWaddr`. Then the associated IP address is extracted from a tab-delimited list of hostname, IP and MAC addresses. Such a list (*ipmaclistL3-remote*) is produced readily by executing a simple script over the farm which extracts the addresses' information from ifconfig output. It should be updated whenever there are relevant hardware modifications; including e.g. node swaping and Ethernet card replacement.

The thus found node's IP address should be used to update the Ethernet port(s) configuration settings; e.g., the file `/etc/sysconfig/network-scripts/ifcfg-eth0` for the first Ethernet port.

logging

A record of the performed recovery is entered in a log file, `stat.log`. This should contain information identifying the recovered node, filesystem, and date the procedure was performed.

checking for floppy

The procedure should terminate with a reboot instruction for the node just recovered. Before this instruction can be issued the recovery floppy needs to be removed from the node's drive, avoiding repetitive execution; the procedure halts and waits for that to happen. Checks are regularly made for the presence of the floppy; additionally, a temporary file with a suggestive name is *touched* on the server. Only once it is no longer detected does the system proceed with the final **reboot** instruction.

further

The automated procedure may be interrupted at any stage (prior to the final reboot instruction) simply by issuing a call to a login program, as the remote `sulogin`, and be continued in a interactive fashion. This may be useful in cases when e.g. several filesystems are to be installed, or for post-installation inspection purposes.

Updates to the recovery procedure are implemented remotely; thus, without the need to e.g. modifying recovery images.

2.4 Installation through recovery

The Level3 farm recovery floppy can naturally be employed for the purpose of node installation; this corresponds essentially to recoverying *all* filesystems. It has been used in cases of new node, new hard drive, and new system installations.

partitioning

Disk (re-)partitioning may be necessary, and can be conveniently done using the `sfdisk` Linux partition table manipulator. First, a file with the intended partition map should be created; this can be done by dumping the partitions of an existing, identically partitioned device, as shown below.

```
sfdisk -d /dev/hda > hda.out
```

A file is produced this way with the appropriate format to serve as input to `sfdisk` as follows.

```
/sbin/sfdisk /dev/hda < hda.out
```

The option $--force$ to `sfdisk` can be used, in which case one should be even more certain of the requested actions.

In case the hard drive to be partitioned is not (cannot be) mounted in a local root filesystem, thus rendering the `sfdisk` facility unavailable, one may use an uncustomized version of the recovery floppy (or raw *tomsrtbt*) to perform a manual partitioning using the standard `fdisk` manipulator.

multi-recovery

Once the disk has been re-partitioned as intended, the remote recovery floppy can be used to complete the installation. The associated (extended) remote script then performs the remaining, usual actions of filesystem creation (`mke2fs`), swap areas setting up (`mkswap`), filesystem mounting and contents transfer from tarballs over the network, MBR fixing, and reboot.

2.5 Generality

The recovery floppies execute whatever instructions are specified on a main remote script; this is named *fixnode.all.remote* and resides on the directory `/home/recovery/remote/` at the recovery `nfs` server (Gateway1).

The performed actions can be a simple call to a shell login program, filesystem recovery or installation as specified in previous sections, or any other specified action supported by the image's system (or by the node's system in a post recovery stage using the chroot environment).

Other commonly useful, available commands include:

- `fdisk`, for disk paritioning
- `chroot`, for executing system commands on the node
- `lilo`, for MBR repair; e.g. `hda1` being the root filesystem,
  `chroot /dev/hda1 /sbin/lilo`

3 Installation

A full node system installation becomes necessary whenever one has a new or blank hard drive, or an Operating System upgrade is to be performed.

Installation of a farm node can be achieved using an adapted recovery procedure. Indeed, such an operation has been repeatedly performed in Level3 in the past. Such a recovery floppy based installation may even be expedite in terms of time duration of the procedure, on a single node basis.

The task nevertheless of installing a large farm system like Level3 can be a challenging one. When the time comes for a full farm upgrade, for example, alternative ways may be worth examining. The so-called RedHat *kickstart* mechanism is explored and customized to farm requirements. A related floppy-*less* procedure for full farm installation is implemented.

3.1 Kickstart

Using individual Fermi RedHat Linux installation on multiple Level3 farm boxes repeatedly, as would be the case during OS upgrade, is time consuming and prone to errors and disparities.

A mechanism for automating the installs of a large number of nodes, which additionally have similar configurations, is therefore desirable to say the least. The *kickstart* [5] RedHat installation method provides in principle such a mechanism. Effectivelly, this allows the *scripting* of the otherwise interactive installation process; the latter is driven by a configuration file containing the answers to the questions that would normally be asked during a manual installation.

Most features can be specified in this single file, including: network configuration, distribution source and method, root password selection, boot loader, language, timezone, packages to be installed.

Additionally, the method offers the possibility of specifying a list of shell level commands (i.e. scripts) to be executed just before (`pre`) or after (`post`) the normal system installation. Use is made of these useful features for specific system and Level3 software installation.

As a general wisdom remark, it should be acknowledged the likelihood of sintax change and availability of even broader functionality in coming RedHat releases.

3.1.1 Image

The installation images can be dowloaded from appropriate locations [4]. The image contents can be accessed for customization purposes by transfering the kickstart image file `bootnet.img` to a device such as a floppy disk as follows.

```
fdformat  /dev/fd0
dd if=bootnet.img of=/dev/fd0H1440
mount -t msdos /dev/fd0 /mnt/floppy
```

```
ls /mnt/floppy/
  boot.msg
  general.msg
  initrd.img
  ldlinux.sys
  param.msg
  rescue.msg
  snake.msg
  syslinux.cfg
  vmlinuz
```

`vmlinuz` is the Linux kernel. The other larger file (`initrd.img`) is the initial root disk image (an ext2 filesystem compressed in a file, containing e.g. the collection of loadable kernel modules).

The file `syslinux.cfg` is the configuration file for the the *syslinux* boot loader, and the various `*.msg` files are message files which are normally displayed by the loader. The config file is relevant only for floppy based installation, in which case it needs to be customized; while the message files can be safely removed if floppy space is needed.

The process is guided by a kickstart configuration file (`ks.cfg`) which needs to be added together with the files mentioned above.

3.1.2 Configuring system installation

The appropriate kickstart configuration, **ks.cfg**, file needs to be created. This is a text file, containing a list of directives and keywords, and can be written from scratch. Alternatively, one can use the kickstart configurator application, `/usr/sbin/ksconfig`.

In the current Fermi Linux release a config file is automatically generated by anaconda; this file contains the configuration selected in the performed installation (done e.g. using a standard interactive installation), and is located at `/root/anaconda-ks.cfg` on the installed node.

The Kickstart config file is formed of the following main sections:

1. System information

2. RedHat packages to be installed

3. `pre` and `post` installation shell commands

Keywords must be in order; comment lines start with `#`.

Excerpts of the Level3 configuration file are persented next with explanations.

Networking

The installation is done via the network. A static network configuration method is used.

A set of `IP addresses` are *reserved* in the Level3 Ethernet network for use in the installation process. These are explicitly mentioned in the `/etc/hosts` file of the server; e.g.

```
192.168.23.250        b0l3ks0.fnal.gov  b0l3ks0
...
```

All the required networking information needs to be specified following the **network** keyword, in a single line.

```
network --bootproto static  --device eth0
        --ip 192.168.23.250 --hostname b0l3ks0
        --netmask 255.255.0.0 --nameserver 192.168.24.11
```

The specified information is static, it will be used during the installation process, as well as after the installation if not changed in the post configuration section.

The server from which to install and installation tree directory are specified with the **nfs** keyword.

```
nfs --server 192.168.24.11 --dir /scratch/731a.install/i386
```

Partitioning

First, the disk partition table is initialized. The **zerombr** keyword clears the master boot record, removing exisiting OS boot loader. **clearpart** removes exisiting partitions, and initializes the disk label to the default.

```
zerombr yes
clearpart --all --initlabel
```

The new partitioning is specified via the **part** keyword. This allows to specify, for each partition, the mount point, filesystem type, minimum size, disk where it is to be created, among others. The following table is being used.

```
part /boot --fstype ext3 --size=50 --ondisk=hda --asprimary
part / --fstype ext3 --size=4096 --ondisk=hda --asprimary
part /tmp --fstype ext3 --size=2048 --ondisk=hda
part swap --size=2047 --ondisk=hda --asprimary
part /cdf --fstype ext3 --size=1 --grow --ondisk=hda
```

Other options

install; make a fresh system installation (rather than upgrade)

text; perform installation in text mode (default is graphical mode)

lang en_US; set language for installation (English)

langsupport en_US; set languages to install on the system

timezone –utc America/Chicago; set system timezone (see `timeconfig`)

bootloader; set boatloader and its location; defaults are GRUB (for LILO, `--useLilo`) at *mbr*

keyboard us; set system keyboard type

mouse none; do not configure mouse for the installed system

skipx; do not configure X on the installed system

authconfig; set up the authentication options for the system (see `authconfig`)

rootpw –iscrypted XA8wIytED41RI; set the system's root password, based on previously derived encrypted form; encrypted password generated e.g. as $perl\ -e\,'printf\ \ crypt("olacomoestas",\ "XA")\ .\ "\backslash n"'$

reboot; do not ask for confirmation for rebooting after installation is completed

Package selection

The list of packages to be installed is initiated with the *%packages* command. Packages can be specified by component or by individual package name. Components are installed by giving their group name; individual packages may be installed by giving the package name, i.e. their `rpm` file name, excluding the version and platform information.

The available packages are listed in $i386/RedHat/RPMS/$ directory of the installation source. For example, the file

```
expect-5.32.2-67.i386.rpm
```

contains the rpm installation for the package named `expect`, characterized by version (5.32.2; `expect -v`), release (67), architecture (i386).

A list of installed packages is produced during installation, located at $/root/install.log$. One may take as example the package section of the anaconda automatically generated configuration file. This can be modified, but care should be payed to relative dependencies.

pre & *post* **install**

The pre and post installation sections (started with the *%pre* and *%post* interpreters) are placed at the end of the configuration file, and define shell commands to be run just before and just after the system's installation.

Pre-install commands are run in the installation's image environment.

Post-install commands are run in the change root, system's environment; commands can still be specified to run outside of the chroot environment when the option $--nochroot$ is used. The system's image is accessible on */mnt/sysimage/* from the kickstart image (nochroot) environment. Different scripting languages can be activated with the $--interpreter$ option. What is more, one can take advantage of all utilities which have been installed on the newly built Linux system.

Installation and settings of Level3 specific features can be instructed directly using **post** (or **pre**) installation scripting. An alternative is to use the post install scripting features of workgroups.

nochroot **environment**

The installer image's system is available in the *nochroot* environment. The latter is accessible in *pre*, $post--nochroot$ (and $after.rpms.nochroot.sh$, when using workgroups) interpreters.

The source $i386$ installation directory is mounted on

```
/mnt/source
```

and the installing system on

```
/mnt/sysimage
```

The workgroup materials, when these are used, are copied to

```
/mnt/sysimage/etc/$WORKGROUP/
```

where `WORKGROUP='cat /etc/workgroup'`, e.g. `CDFlevel3`.

3.1.3 Post configuration

This is the section of the configuration file initiated by the *%post* directive, run in chroot (system's) environment.

Here one specifies extra actions to be performed after the normal system installation. In particular, one can access the network, and mount an apppropriate directory.

```
mkdir /mnt/ksdir
mount 192.168.24.11:/home/recovery/kickstart /mnt/ksdir
echo "Kickstart installation performed on `/bin/date`" >  /mnt/ksdir/ks.post
...
umount /mnt/ksdir
```

The list of actions to be performed may include the following.

Update fstab:

```
echo "b0l3serv:/home     /home                   nfs    \
         auto,rw,rsize=8192,wsize=8192,soft" >> /etc/fstab
```

Modify/create general system's configuration files:

```
/etc/hosts
/etc/ntp.conf
/etc/ssh/ssh_config
/etc/ntp/step-tickers
/etc/sysconfig/static-routes
/etc/sysconfig/network
/etc/sysconfig/network-scripts/ifcfg-eth#
```

Perform runtime services settings and others:

```
/sbin/chkconfig --level 35 anacron  on
/sbin/chkconfig --level 35 gpm      on
/sbin/chkconfig --level 35 network  on
/sbin/chkconfig --level 35 sshd     on
/sbin/chkconfig --level 35 netfs    on
/sbin/chkconfig --level 35 atd      on
/sbin/chkconfig --level 35 keytable on
/sbin/chkconfig --level 35 portmap  on
/sbin/chkconfig --level 35 syslog   on
/sbin/chkconfig --level 35 nfslock  on
/sbin/chkconfig --level 35 ypbind   on
/sbin/chkconfig --level 35 crond    on
/sbin/chkconfig --level 35 kudzu    on
/sbin/chkconfig --level 35 random   on
/sbin/chkconfig --level 35 sendmail off
/sbin/chkconfig --level 35 apmd     off

/sbin/hwclock --utc --systohc
/usr/bin/updatedb
```

3.1.4 CDFlevel3 workgroup

Fermi Workgroups [6] have been designed to provide extra installation customization capabilities. In practice, this further extends (and facilitates) the *packages* and *post* installation procedure, automatically running a custom shell script at the end of the install, and providing a *storage* area accessible by the shell script. In particular, this circumvents the need to mount an extra nfs location with Level3 specific materials, as these can be placed in designed locations in the main source location for the installation.

An appropriate workgroup, **CDFlevel3** [6], has been defined, [2] and is located in the directory (at the source installation tree location)

`i386/Fermi/workgroups/CDFlevel3`

During installation this gets copied with preserved tree structure to $/etc/CDFlevel3$ (in general to /etc/‘cat /etc/workgroup‘), as instructed in the file [3]

`i386/Fermi/common/scripts/post.sh`

with preserved tree structure, thus becoming available to the customization script. It contains the following base directories.

RPMS

Contains workgroup specific rpms to be last installed (with the rpm option $--force$), before customization script is executed.

configfiles

This is a storage area; it may hold farm specific configuration files.

scripts

This is the location for customization scripts. Scripts located in this area are executed in case their names match the following, and have execution permissions:

- before.rpms.sh
- after.rpms.sh
- after.rpms.nochroot.sh

The first two are executed in a chroot (system's) environment; the last one is run in the installing system's context, the install medium location being available in $SOURCE, and the new install area in $CHROOT. In these bash scripts full paths should be specified.

comps

This file defines the groups and rpm packages to be installed during the RedHat packages installation time.

It should be merged with other `comps` files (from other workgroups) and the RedHat defined portion, in the following file.

[2]The Level3 workgroup was initially put together for farm installation purposes by A.Korn.

[3]This script normally is executed automatically. In versions 731x, there was a bug in the installer which prevented this to occur during kickstart; it thus required (as suggested to me by T. Dawson) an explicit call to the script in the post section of the kickstart configuration file:

%post - -nochroot
sh /mnt/source/Fermi/common/scripts/post.sh

```
i386/RedHat/base/comps
```

One may have the following simplest comps definition of the workgroup

```
0 --hide  CDFlevel3 {
  CDFlevel3-tag
  expect
  upsupdbootstrap
}
```

The UPS/UPD installation is completed by placing the $upsupdbootstrap - local * .rpm$ (for installation in */local*) from $RedHat/RPMS/$ in the workgroup's $RPMS/$ directory.

Farm specific configuration files, as well as necessary tarballs, are placed in $configfiles/$, to be handled by a customization file named $after.rpms.sh$ located in *scripts/*; this would substitute the *post* installation section of the $ks.cfg$ kickstart configuration file.

For RedHat release 8.0 or higher, the comps file, actually **comps.xml**, is written in XML format, for increased flexibility.

The workgroups feature is not *necessary* for Level3 installation, as alternative ways have been designed; nevertheless, it has been set up, fully tested, and is made available, with this section serving as guide, as it may be usefull for additional and administration convenience.

3.2 Installation disk

The kickstart installation may be performed by a floppy disk. This must contain a Linux kernel image and appropriate configuration files; specifically the following files are required:

vmlinuz
initrd.img
ldlinux.sys
syslinux.cfg
ks.cfg

Files `syslinux.cfg` and `ks.cfg` need to be created or customized. The latter has been described in detail in previous sections.

The floppy uses the `syslinux` boot loader; `syslinux.cfg` is its configuration file, and contains the following:

default ks : assign kickstart as default boot image
prompt 0 : do not bring up prompt at boot, or
timeout 60 : decrease delay for default image to be booted
label ks : image label (arbitrary, as long as consistent with default)

kernel vmlinuz : specify name of file containing the kernel
append : add kernel parameters, including
`ks=floppy` : find `ks.cfg` on the floppy (drive `/dev/fd0`)
`initrd=initrd.img` : name of initial root disk image file
`lang=`
`text` : keyword text enables text-based install
`ksdevice=eth0` use this network device to connect to the network
`devfs=nomount ramdisk_size=8192`

The mentioned, customized files should be transferred to a floppy disk. The installation is performed simply by booting the node with the so produced kickstart installation floppy.

During the installation process various consoles are accessible with information of what is taking place:

`Alt-F1` - installation dialog
`Alt-F2` - shell prompt
`Alt-F3` - install log (messages from install program)
`Alt-F4` - system log (messages from kernel, etc.)
`Alt-F5` - other messages

further

In another implementation, for additional functionality, the default image directives in `syslinux.cfg` may be left included; these can be specified at boot prompt, in which case this should not be disabled, and an appropriate delay (e.g., `timeout 60` for 6 seconds) should be specified.

A custom boot message screen may be displayed, by adding the keyword `display ksl3.msg` to the `syslinux.cfg` , where `ksl3.msg` is a message file. The contents of this file may be marked up, such as by adding colored words (e.g. `^O09LEVEL3^O02` is displayed in blue).

If the line `F1 ksl3.msg` is added, the message is shown when `F1` is pressed at boot time; this allows for displaying several messages, containing for example instructions associated with different kernels or configurations available.

The floppy disk kickstart procedure is most relevant for installing nodes which do not boot up properly, as the installation does not make use of a possibly pre-existing system.

3.3 Hard drive based installation

For nodes which have a running operating system, as it is the case during upgrade, a floppy*less*, hard drive based installation procedure has been set. This is most convenient, as a node or a pre-defined *list* of nodes can be installed by executing a single script, which can even be done from a remote location, e.g. from one's office!

The strategy here is to use the hard drive has the kickstart installation media; it involves the following steps: (i) transfer the necessary images and corresponding configuration files to the hard drive to be installed; (ii) configure the current node's boot loader to use the kickstart image; (iii) reboot the node. After this, if all went well, the nodes come up with a brand new installation.

Currently, LILO is used in the farm; accordingly, its configuration file, `lilo.conf`, is modified by adding the kickstart image specifications.

Assume the node to be installed has the following disk structure: the root filesystem is $hda1$; it has an extra partition, $hda6$ mounted on $/usr$; the current directory contains the kickstart image installation files, together with a proper kickstart configuration file (described in previous sections).

```
mkdir /usr/boot
ipadr=‘/sbin/ifconfig eth0 | grep inet        \
       | cut -f 12 -d " " | sed -e s/addr:/""/‘
hostn=‘hostname -s‘
todo=‘echo $ipadr --hostname $hostn‘
sed -e s/"192.168.23.250"/"$todo"/ ks.cfg-template > /usr/boot/ks.cfg
cp -f initrd.img /usr/boot/initrd-install.img
cp -f vmlinuz    /usr/boot/vmlinuz-install
cat <<EOF >>/etc/lilo.conf
  image=/usr/boot/vmlinuz-install
     label=install
     initrd=/usr/boot/initrd-install.img
     append="ks=hd:hda6/boot/ks.cfg"
     read-only
     root=/dev/hda1
EOF
/sbin/lilo
/sbin/lilo -R install
/sbin/reboot
```

3.4 Installation through recovery

Some kinds of nodes on the farm require specific installation care. This is the case for the Converter and Output nodes. These require extra or modified drivers for additional hardware (e.g., ATM card in Converters, multi-port Ethernet tulip card in both). Two possibilities are foreseen for installing these: (*i*) produce appropriate RPMs, or (*ii*) install the filesystem contents from an already properly installed, similar node. Given that these are considerably fewer than Processor nodes, and the possibly delicate task of creating proper rpm installation files, the most straightforward way is the latter.

A filesystem recovery based installation can be implemented making use as well of the linux kernel which comes with the Fermi installation. Namely, booting the Linux kernel in its rescue mode

```
boot: linux rescue
```

one is re-directed to a shell environment, allowing to perform general rescue operations. These may include disk inspection, partitioning (e.g. `sfdisk` is available), filesystem creation (e.g. `ext3` feature are available), mouting local and remote filesystems, and so on. Additionally, the `i386` source directory on the installation server is automatically mounted.

The following procedure was therefore implemented and tested.

source directory

Add to the installation source on the server the directory

```
i386/LEVEL3/
```

this becomes accessible in the recovery system's image environment under

```
/mnt/source/LEVEL3/
```

filesystem contents

Place there all source filesystem contents, e.g. in

```
i386/LEVEL3/tarballs/
```

install script

Create a recovery-like script, placing it in the `i386/LEVEL3/` recovery-based installation directory.

```
i386/LEVEL3/scripts/
```

This may include actions as the following:

- `mke2fs -j /dev/$fsname1` : create `ext3` filesystems
- `mkswap /dev/$fsswap` : set up swap areas
- `mkdir /mnt/$fsname1; mount -t ext3 /dev/$fsname1 /mnt/$fsname1` : mount filesystems
- `cd /mnt/$fsname1; tar xvf $tardir/$fstar1`

- `chroot /mnt/$rootfs /sbin/lilo` : configure boot loader
- `echo "Node installed on ‘date‘ > /mnt/$fsname1/l3install.log "`: write installation record

Once such a procedure is set up this way, the installation may be executed by performing the following steps: (*i*) (re-)partition the target disk, if necessary; (*ii*) reboot the node using the installation image in rescue mode (this can be done using a floppy disk, or an hard drive based installation, in a similar fashion as described in previous sections); (*iii*) execute the installation script previously set up,

```
/mnt/source/LEVEL3/scripts/nodeinstall.sh
```

Exiting the rescue shell will reboot the node just installed.

Note that the filesystem table should include explicit partitions locations rather than *label*'s which without further do won't be defined.

Acknowledgments

I wish to thank T. Kim, A. Korn, S. Tether for discussions; G. Gómez-Ceballos, I. Kravchenko for also dedicating attentive care to the system and level3 code expertise; the full Level3 team; and finally the online shift crew for permanently using the system.

The Level3 work here described was supported in part by the Science and Technology Foundation, Portugal.

References

[1] N. Leonardo for the EVB/L3 team, *Event Builder and Level 3 Manual for Experts*, CDF Note 6138, http://www-cdfonline.fnal.gov/evbl3shift/evbl3pager.html. 3

[2] N. Leonardo for the EVB/L3 team, *EVB/L3 experts online page*,
http://www-cdfonline.fnal.gov/evbl3shift/evbl3pager.html
Recovery, http://www-cdfonline.fnal.gov/evbl3shift/pager/recovery/
Node statistics, http://www-cdfonline.fnal.gov/evbl3shift/pager/recovery/nodelist.html
10, 30

[3] T. Oehser, *The most GNU/Linux on 1 floppy disk*, http://www.toms.net/rb/ 8

[4] The Femi Linux page, http://www-oss.fnal.gov/projects/fermilinux/
v7.3.1 distribution, http://www-oss.fnal.gov/projects/fermilinux/731/home.html
v7.3.1 ftp, ftp://linux.fnal.gov/linux/731a/
v7.3.1 images, ftp://linux.fnal.gov/linux/731a/i386/images/ 15

[5] The Official Red Hat Linux Customization Guide,
http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/index.html
15

[6] Fermi Workgroups,
http://www-oss.fnal.gov/projects/fermilinux/common/workgroups.maintainers.html
A. Korn, CDFlevel3 workgroup,
ftp://linux.fnal.gov/linux/731/i386/Fermi/workgroups/CDFlevel3/
T. Dawson, Maintaining a workgroup,
http://home.fnal.gov/d̃awson/linux/custom/maintain.workgroup.html 20, 21

[7] UPS, UPD and UPP page at Fermilab,
http://www.fnal.gov/docs/products/ups/ 4, 5, 6

UPD and UPD distribution server,
http://kits.fnal.gov/

Main ftp server containing available product releases,
ftp://ftp.fnal.gov/products/

A Recovery image configuration and procedure implementation

A.1 Configuration files

inittab

```
::sysinit:/etc/rc.S
::ctrlaltdel:/bin/reboot
::shutdown:/etc/rc.0
```

rc.S

```
#!/bin/sh
PATH=/bin
BT=1
#Read from image
mount -o remount /
mount -a
mount -o ro /dev/fd0u1722 /fl
. /fl/settings.s
update &
cp /fl/rc.* /fl/*.s /bin
gzip -d /bin/*.gz
. /bin/rc.custom
umount /fl
#Execute local script
/fixnode.all.remote.floppy
```

Note: When a PC is booted the BIOS runs its power-on-self-test (post) and then looks for more boot information in the boot sector of the available device: (i) typically in the Master Boot Record (MBR) of the hard drive, (ii) in floppy if present (and booting device order set accordingly in BIOS). The Linux kernel then loads **init**, the general process dispatcher, using `/etc/inittab` as its configuration file, where run levels are defined and the default is specified. The initialisation table consists of records of the format: `id:runlevels:action:process`. *Usually*, the first script run by init is `rc.S`, then it drops down and runs the default run level (usually 2).

In the present recovery implementation the procedure is performed just *before* run levels are executed.

A.2 Include a new binary

Below is shown how to include a new binary file as part of a customized image. The case of *sulogin* (single user login) program is provided in detail as an example.

```
Download source code; e.g. search as in:
    http://rpmfind.net/linux/rpm2html/search.php?query=SysVinit

Unpack
    > rpm -ihv SysVinit-2.78-5.src.rpm

Modify code; in sulogin.c comment out getpasswd() call
    /*
    while(1) {
        if ((p = getpasswd()) == NULL) break;
        if (pwd->pw_passwd[0] == 0 ||
        strcmp(crypt(p, pwd->pw_passwd), pwd->pw_passwd) == 0)
    */
        sushell(pwd);
    /*
        printf("Login incorrect.\\n");
    }
    */

Remove other un-necessary features; e.g. PCMCIA related

Compile and enforce static link
    > cc -Wall -O2 -D_GNU_SOURCE   -c -o sulogin.o sulogin.c
    > cc -s -static -o sulogin sulogin.o -lcrypt

Transfer executable to unpacked image directory
   > cp sulogin  tomsrtbt-[version]/tomsrtbt-[version].unpacked/2/sbin/

Call program; impose sulogin at specified run level
   > vi tomsrtbt-[version].unpacked/2/etc/inittab

      id:4:initdefault:
      si:S:sysinit:/etc/rc.S
      rc:5:wait:/etc/rc.M
      ca::ctrlaltdel:/bin/shutdown -t5 -rfn now "CtlAltDel"
      l0:0:wait:/etc/rc.0
      l6:6:wait:/etc/rc.6
      ~~:S:wait:/sbin/sulogin
      c1:5:wait:/sbin/sulogin
```

A.3 Create a recovery floppy

```
Go to Tom's Root Boot Linux page:
   http://www.toms.net/rb/

Download the software from one of the listed sites, e.g.
   ftp://www.tux.org/pub/distributions/tinylinux/tomsrtbt/tomsrtbt-[version].tar.gz

Expand the .tar.gz archive:
  > tar xvzf tomsrtbt-[version].tar.gz
```

```
A) Create raw floppy:
  > cd tomsrtbt-[version]
  > ./install

B) Create a customized floppy (e.g., include myscript):
  > cd tomsrtbt-[version]
  > ./unpack.s tomsrtbt.raw
  > cd tomsrtbt-[version].unpacked/
  > cp /mylocation/myscript 2/
  > chmod +x 2/myscript
  > ./buildit.s
  > cd tomsrtbt-[version+1]
  > ./install
```

A.4 Remote recovery procedure

The common procedure for restoring a filesystem on a Level3 node involve the following steps.

1. Go to remote recovery directory; `Gateway> cd /home/recovery/remote`
2. Specify filesystem information: check/edit file `recover.config-remote`
3. Check for existence of corresponding tar file in `tarballs` directory; otherwise create it
4. For Output and Converter nodes, re-seat Ethernet cable (A) on on-board Ethernet port
5. Check/edit recovery script if needed, `fixnode.all.remote`
6. Execute recovery script on broken node; insert remote recovery disk on floppy drive of broken node and hit reset; this will execute locally the remote script named `fixnode.all.remote` located on remote directory
7. Remove floppy when done; wait $\sim 4min$ or check `stat.log`; the procedure will wait until the floppy has been removed from the node's drive; for Output or Converter nodes, the Ethernet cable should be re-seat into its original port
8. Check that node is up and well, starting with `ping, ssh, df`; execute specific actions known to be required by certain classes of nodes
9. Distribute relay/level3 code in case the relevant filesystem (`/cdf`) has been restored (see [2] for instructions)

Details and updates are found in the recovery page in [2].

A.5 Filesystem recovery scripts

File: fixnode.all.remote
Description: main recovery script

```
#!/bin/sh

#Get MAC address of current node
mac=`/usr/bin/ifconfig eth0 | grep HWaddr | cut  -d " " -f 11`
echo "The MAC address found is $mac"

#Extract node name + ip from associated $mac in ipmaclistL3-remote table
name=`/usr/bin/grep $mac /mnt/extdisk/remote/ipmaclistL3-remote|cut -f 1 -d " "`
ipadr=`/usr/bin/grep $mac /mnt/extdisk/remote/ipmaclistL3-remote|cut -f 2 -d " "`
echo "The node name found is $name with ip address $ipadr"

#Read recovery configuration: select filesystem
fsname=`grep -v "#" /mnt/extdisk/remote/recover.config-remote|cut -f 1`
fstar=`grep -v "#" /mnt/extdisk/remote/recover.config-remote|cut -f 2`
fsip=`grep -v "#" /mnt/extdisk/remote/recover.config-remote|cut -f 3`
fsid=`grep -v "#" /mnt/extdisk/remote/recover.config-remote|cut -f 4`

#Recreating filesystem on /dev/$fsname
/usr/bin/mke2fs /dev/$fsname
#Setting max mount count to 1
/usr/bin/tune2fs -c 1 /dev/$fsname

#Mount /dev/$fsname
mkdir /mnt/$fsname
mount -t ext2 /dev/$fsname /mnt/$fsname

#Expand tarball
cd /mnt/$fsname
tar xvf /mnt/extdisk/tarballs/$fstar

#Assign correct ip-address to eth0: $fsip->$ipadr
if [ $fsid = "/" ]
then
cd /mnt/$fsname/etc/sysconfig/network-scripts
sed -e s/$fsip/$ipadr/ ifcfg-eth0 > tempo
mv tempo ifcfg-eth0
fi

#Add log info to b0l3serv:/home/recovery/stat.log
echo "Rebooted node $name; recovered $fsname aka $fsid; \
      on " `/usr/bin/date` >> /mnt/extdisk/stat.log

#Wait until recovery disk is removed
mkdir /mnt/floppy
mount -t minix /dev/fd0u1722 /mnt/floppy
```

```
flp=`df | grep floppy | wc | cut -f 1`
umount /mnt/floppy
while [ $flp -gt 0  ]; do
 sleep 5
 mount -t minix /dev/fd0u1722 /mnt/floppy
 flp=`df | grep floppy | wc | cut -f 1`
 echo "Still waiting for floppy to be removed... $flp : not zero"
 umount /mnt/floppy
 touch /mnt/extdisk/remove_floppy_to_reboot_$name
done
rm  /mnt/extdisk/remove_floppy_to_reboot_$name

#Reboot the node
sleep 2
reboot
```

Auxiliary files

The remote recovery directory is `Gateway1:/home/recovery/remote/`. A description of the main remote files follows.

- `recover.config-remote`. This file contains a map which specifies which filesystem is to be recovered, and tarball information; it is read by the main script; needs to be edited before it is executed (see header for editing instructions).

  Format:

  ```
  #fs     tarball         taredFromIP     partition
  hda1    rootfs_274.tar  192.168.23.74   /
  #hda6   usr_273.tar     192.168.23.73   /usr
  #hda7   cdf_260.tar     192.168.23.60   /cdf
  ```

- `ipmaclistL3-remote`. This file contains a list of names, IP and MAC addresses of each single node; it is read by the main script; ought to always be up-to-date.

  Format:

  ```
  b0l3c01 192.168.21.1 00:00:D1:1C:6D:CF
  ...
  b0l3001 192.168.22.1 00:E0:81:04:3E:F7
  ...
  b0l3u01 192.168.26.1 00:D0:B7:A7:95:9E
  ...
  ```

  This needs to be updated whenever an hardware alteration takes place in the farm, including: installation of a new node; node swapping; Ethernet card replacement or exchange. Output and Converter nodes use a multi-port Ethernet card which is not supported by the system on the floppy; therefore one needs to use instead the non-default (motherboard embedded) port, and it is the corresponding MAC addresses that are necessary. An updated version of the file may be readily obtained by executing a simple script as below in each node.

```
#!/bin/bash
echo "usage: getmacip_node.sh"
mac=`/sbin/ifconfig eth0 | grep HWaddr | cut -f 11 -d " "`
ip=`/sbin/ifconfig eth0 | grep inet | cut -f 12 -d " " | sed -e s/addr:/""/`
echo "`hostname` $ip $mac"
exit
```

For example, for processor nodes,

```
ssh root@b0l3pcom1
touch /home/l3proxy/recovery/newlist.txt
expect /root/scripts/allProc.exp
/home/recovery/remote/scripts/getmac_node.sh \
        >> /home/l3proxy/recovery/newlist.txt
```

and similarly for Converter and Output nodes after Etherent cable re-seating.

- `stat.log`. This is the general recovery log file; it is automatically edited each time the procedure is used, with information about the node and filesystems just recovered.
  Format:

  ```
  Rebooted node b0l3203; recovered hda6 aka /; on  Tue Sep 30 14:54:11 2003
  ```

- `tarballs/`. This directory should contain (at the time of execution) the appropriate tar file with the filesystem contents to be installed; correct description of these files should appear on the file ”recover.config-remote”; appropriate tarballs may be available on: Gateway2:tarballs/tarballs.reservoir
- `bin/`. This directory contains binaries to be executed by the recovery system; e.g. `sulogin`.
- `scripts/`. This directory contains various scripts pertaining to the procedure.
- `store/`. This is a backup directory; prior to modifying any of the files one ought to be sure that a backup version is saved.

B Install configuration and implementation

B.1 Configuration files

File: ks.cfg
Description: main kickstart configuration file

```
nfs --server 192.168.24.11 --dir /scratch/731a.install/i386
install
lang en_US
langsupport en_US
keyboard us
mouse none
text
skipx
network --bootproto static --device eth0 --ip 192.168.23.250                  \
        --netmask 255.255.0.0 --gateway 192.168.24.0 --nameserver 192.168.24.11
xconfig --card "ATI Mach64" --videoram 4096 --hsync 31.5 --vsync 50-61 \
        --resolution 1024x768 --depth 16 --defaultdesktop gnome
rootpw --iscrypted XA8wIytED41RI;
firewall --disabled
authconfig --enableshadow --enablemd5 --enablenis     \
        --nisdomain b0l3p --nisserver b0l3serv.fnal.gov
timezone --utc America/Chicago
bootloader
zerombr yes
clearpart --all --initlabel
part /boot --fstype ext3 --size=50 --ondisk=hda --asprimary
part / --fstype ext3 --size=4096 --ondisk=hda --asprimary
part /tmp --fstype ext3 --size=2048 --ondisk=hda
part swap --size=2047 --ondisk=hda --asprimary
part /cdf --fstype ext3 --size=1 --grow --ondisk=hda
reboot

%packages
@ GNOME
@ Network Support
@ Authoring and Publishing
@ Software Development
@ Kernel Development
@ Fermi Kerberos
@ Openssh Server
@ FermiStandAlone
ghostscript-fonts
balsa
mozilla-chat
compat-libstdc++
gaim
Xaw3d-devel
glade
```

```
libesmtp
ddd
xmms-gnome
libesmtp-devel
libgtop-devel
SDL
gdk-pixbuf-devel
pan
ORBit-devel
doxygen
glib2-devel
kernel-smp
lesstif-devel
libghttp-devel
pygtk-devel
tetex-xdvi
smpeg
gnome-core-devel
bonobo-devel
libcap-devel
gnome-vfs-devel
rsync
SDL_net
control-center-devel
pango-devel
openssh-askpass-gnome
GConf-devel
bonobo-conf-devel
transfig
w3c-libwww-devel
eel-devel
libglade2-devel
xfig
audiofile-devel
xpdf
licq
imlib-devel
gnome-libs-devel
unzip
usbview
gq
gv
gtk+-devel
librsvg-devel
gcc-objc
gal-devel
python2-devel
SDL_image
acroread-plugin
magicdev
w3c-libwww
libpcap
```

```
exmh
fam-devel
freetype-devel
ghostscript
zz_libg2c.a_change
memprof
XFree86-devel
Guppi-devel
lesstif
libglade-devel
xawtv
openssh-askpass
redhat-config-network
ical
guile-devel
libole2-devel
licq-gnome
oaf-devel
nedit
gnome-media
SDL_mixer
librep-devel
gnome-print-devel
atk-devel
libmng-devel
libungif-devel
libxml-devel
acroread
glib-devel
gtk2-devel
netpbm-devel
galeon
ucd-snmp
xmms
emacs
expect


%post

mkdir /mnt/ksdir
mount 192.168.24.11:/home/recovery/kickstart /mnt/ksdir

hostn=`cat /etc/sysconfig/network | grep HOSTNAME | sed -e s/HOSTNAME=//`
touch '/mnt/ksdir/$hostn.kslock'
nodelog='/mnt/ksdir/$hostn.ks.log'
touch $nodelog
echo "Kickstart hd-installation $hostn performed on `/bin/date`"   \
      >>  /mnt/ksdir/ks.log
echo "Starting l3 ks hd install of $hostn on `/bin/date`" >> $nodelog

echo "`/bin/date` : copying config files" >> $nodelog
```

```
echo "b0l3serv:/home     /home                   nfs    \
       auto,rw,rsize=8192,wsize=8192,soft" >> /etc/fstab
cp -f /mnt/ksdir/etc/ks-hosts         /etc/hosts
cp -f /mnt/ksdir/etc/ks-static-routes /etc/sysconfig/static-routes
cp -f /mnt/ksdir/etc/ks-ntp.conf      /etc/ntp.conf
cp -f /mnt/ksdir/etc/ks-ssh_config    /etc/ssh/ssh_config
cp -f /mnt/ksdir/etc/ks-sshd_config   /etc/ssh/sshd_config
cp -f /mnt/ksdir/etc/ks-step-tickers  /etc/ntp/step-tickers

echo "`/bin/date` : copying cdf.tar" >> $nodelog
if [ -d /cdf ]; then
cd /cdf
tar xfvz /mnt/ksdir/cdf/cdf_ks.tgz
cd /
ln -s /cdf/log log
cd /cdf/level3/filter/control
tar xfvz /mnt/ksdir/cdf/ks-control-vtest.tgz
cd /cdf/level3/filter/relay
tar xfvz /mnt/ksdir/cdf/ks-relay-v1_4_7.tgz
fi

echo "`/bin/date` : copying local.tar" >> $nodelog
cd /
tar xpvfz /mnt/ksdir/cdf/ks-local.tgz
#rpm -i --force /mnt/ksdir/rpms/upsupdbootstrap-2.2-8.i386.rpm
#rpm -i --force /mnt/ksdir/rpms/upsupdbootstrap-local-2.2-2.i386.rpm

echo "`/bin/date` : copying ups databases" >> $nodelog
if [ -d /local ]; then
cd /usr/local/etc
ln -s /local/ups/etc/setups.sh .
ln -s /local/ups/etc/setups.csh .
fi

#runtime services settings
/sbin/chkconfig --level 35 anacron  on
/sbin/chkconfig --level 35 gpm      on
/sbin/chkconfig --level 35 network  on
/sbin/chkconfig --level 35 sshd     on
/sbin/chkconfig --level 35 netfs    on
/sbin/chkconfig --level 35 atd      on
/sbin/chkconfig --level 35 keytable on
/sbin/chkconfig --level 35 portmap  on
/sbin/chkconfig --level 35 syslog   on
/sbin/chkconfig --level 35 nfslock  on
/sbin/chkconfig --level 35 ypbind   on
/sbin/chkconfig --level 35 crond    on
/sbin/chkconfig --level 35 kudzu    on
/sbin/chkconfig --level 35 random   on
/sbin/chkconfig --level 35 sendmail off
/sbin/chkconfig --level 35 apmd     off
/sbin/hwclock --utc --systohc
```

```
/usr/bin/updatedb

echo "`/bin/date` : done" >> $nodelog
echo "      Ending installation  of `hostname -s` on `/bin/date`" \
     >> /mnt/ksdir/ks.log
echo "...." >> /mnt/ksdir/ks.log

touch /root/l3install.log
cat <<EOF >> /root/l3install.log
 Level3 Farm Installation Procedure
 Author: Nuno Leonardo

 Node:   $hostn
 Method: hard-drive kickstart
 Date:   `/bin/date`

 Details:
EOF

cat $nodelog >> /root/l3install.log
rm '/mnt/ksdir/$hostn.kslock'
umount /mnt/ksdir
exit
```

File: syslinux.cfg
Description: floppy boot loader configuration

```
default ks
prompt  1
timeout 10
display ksl3.msg
F1 ksl3.msg
F2 boot.msg
F3 general.msg
F4 rescue.msg
label ks
  kernel vmlinuz
  append ks=floppy ksdevice=eth0 initrd=initrd.img lang= text \
         devfs=nomount ramdisk_size=8192
label linux
  kernel vmlinuz
  append initrd=initrd.img lang= devfs=nomount ramdisk_size=8192 vga=788
label text
  kernel vmlinuz
  append initrd=initrd.img lang= text devfs=nomount ramdisk_size=8192
label expert
  kernel vmlinuz
  append expert initrd=initrd.img lang= devfs=nomount ramdisk_size=8192
label nofb
```

```
  kernel vmlinuz
  append initrd=initrd.img lang= devfs=nomount nofb ramdisk_size=8192
label lowres
  kernel vmlinuz
  append initrd=initrd.img lang= lowres devfs=nomount ramdisk_size=8192
```

File: ksl3.msg

Description: marked up message to be displayed at boot time (instructions may be added).

```
^L

            ^O09Welcome to the ^O0cLevel3 Farm^O09 kickstart installation!^O07



               LL     EEEEEE  V     V EEEEEE LL      333333
               LL     EE      V     V EE     LL           3
               LL     EEEE     V   V  EEEE   LL        3333
               LL     EE        V V   EE     LL           3
               LLLLLL EEEEEE     V    EEEEEE LLLLLL  333333


                    ^O05K   I   C   K   S   T   A   R   T ^O07



                             Procedure by
                           ^O0fNuno T. Leonardo^O07
                 Massachusetts Institute of Technology



            ^O05[F1-Level3] [F2-Main] [F3-General] [F4-Rescue]^O07
```

B.2 Installation scripts

File: ks.harddrive.install.sh

Description: sets node for hard drive based kickstart installation

```
#!/bin/bash
hostn=`hostname -s`;
if   [ $hostn = "b0l3pcom1" ] ||  [ $hostn = "b0l3pcom2" ]; then
 echo "ATTENTION: action forbidden in $hostn... goodbye !";
 exit 1;
else
 echo "Node $hostn will be prepared for kickstart HD installation !"
fi
```

```
ipadr=`/sbin/ifconfig eth0 | grep inet | cut -f 12 -d " " | sed -e s/addr:/""/`
todo=`echo $ipadr --hostname $hostn`
rm -f /boot/ks.cfg
sed -e s/"192.168.23.250"/"$todo"/ /home/leonardo/l3kickstart/ks-hd.cfg > /boot/ks.cfg

cp -f /home/leonardo/l3kickstart/initrd.img /boot/initrd-install.img
cp -f /home/leonardo/l3kickstart/vmlinuz    /boot/vmlinuz-install

if [ -e /etc/lilo.conf-OLD ]; then
echo "lilo.conf will not be modified";
elif [ -e /etc/lilo.conf ]; then
cp -f /etc/lilo.conf /etc/lilo.conf-OLD;
cat <<EOF >>/etc/lilo.conf

#kickstart install image
image=/boot/vmlinuz-install
        label=install
        initrd=/boot/initrd-install.img
        append="ks=hd:hda1/ks.cfg"
        read-only
        root=/dev/hda2
EOF
else
echo "/etc/lilo.conf does not exist \!"
exit;
fi

/sbin/lilo
/sbin/lilo -R install
/sbin/shutdown -r now
exit
```

File: makefloppy.ks.sh
Description: produces node-customized kickstart installation floppy

```
#!/bin/sh
echo "Customized level3 kickstart floppy creation"
echo -n "Enter level3 node name [b0l3ks0]: "
read hostn
echo -n "Complete node IP address 192.168. [23.250]: "
read ipaddr

echo "Customizing configuration..."
if [ ! $hostn ]; then
 hostn='b0l3ks0';
fi
if [ ! $ipaddr ]; then
 ipaddr='192.168.23.250';
```

```
fi
ipaddr="192.168.$ipaddr";

echo "  Node: $hostn   IP: $ipaddr"
sed -e s/"192.168.23.250"/$ipaddr/ ks.cfg-template > ks.cfg-template1
sed -e s/b0l3ks0/$hostn/ ks.cfg-template1 > ks.cfg

echo -n "Insert floppy disk...."
read something

echo -n "Do you want to format floppy? "
read fmat
if [ "$fmat" = "yes" ]; then
echo "Start formatting disk...";
fdformat /dev/fd0H1440
elif [ "$fmat" = "no" ]; then
echo "Skipping disk formatting...";
else
echo "Please try again.";
exit 1;
fi

echo -n "Need to transfer raw image? "
read rimg
if  [ "$rimg" = "yes" ]; then
echo "Transferring image to floppy disk..."
dd if=bootnet.img of=/dev/fd0H1440
elif [ "$rimg" = "no" ]; then
echo "Skipping disk formatting...";
else
echo "Please try again.";
exit 2;
fi

echo "Starting image configuration..."
echo "Mounting disk..."
mount -t msdos /dev/fd0 /mnt/floppy
echo "Costumizing image..."
cp ks.cfg /mnt/floppy
cp syslinux.cfg /mnt/floppy
umount /mnt/floppy
echo "Done"
exit
```

Typical use and output of this script is presented next.

```
> ./makefloppy.sh

Customized level3 kickstart floppy creation
Enter level3 node name [b0l3ks0]: b0l3ks3
Complete node IP address 192.168. [23.250]: 23.253
```

```
Customizing configuration...
  Node: b0l3ks3   IP: 192.68.23.253
Insert floppy disk....
Do you want to format floppy? yes
Start formatting disk...
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
Need to transfer raw image? yes
Transferring image to floppy disk...
2880+0 records in
2880+0 records out
Starting image configuration...
Mounting disk...
Costumizing image...
Done
```

File: nodeinstall.sh

Description: script for node installation through global filesystems recovery using rescue installation image

```
#!/bin/sh

#tarballs location
tardir='/mnt/source/LEVEL3/tarballs'

##### BEGIN of script configuration
#Enter node information:
#node name (change default b0l300):
name=b0l300
#ip address (change default 192.168.23.00)
ipadr=192.168.23.00

#Enter filesystem information; tarballs located on \
# /scratch/731a.install/i386/LEVEL3/tarballs/
#filesystem 1: root
fsname1=hda2
fstar1=rootfs_246.731.tar
fstar1a=usr.731.tar
fsip1=192.168.23.46
#filesystem 2: boot
fsname2=hda1
fstar2=boot.731.tar
#filesystem 3: cdf
fsname3=hda6
fstar3=cdf.731.tar
fstar3a=ks-control-vtest.tar
fstar3b=ks-relay-v1_4_7.tar
#filesystem: swap
```

```
fsswap=hda3
#filesystem: tmp
fstmp=hda5

echo "The node to be installed is $name with ip address $ipadr"

echo "Recreating all filesystems ..."
mke2fs -j /dev/$fsname1
mke2fs -j /dev/$fsname2
mke2fs -j /dev/$fsname3
mke2fs -j /dev/$fstmp
mkswap -j /dev/$fsswap

echo "Mount all partitions..."
mkdir /mnt/$fsname1
mkdir /mnt/$fsname2
mkdir /mnt/$fsname3
mount -t ext3 /dev/$fsname1 /mnt/$fsname1
mount -t ext3 /dev/$fsname2 /mnt/$fsname2
mount -t ext3 /dev/$fsname3 /mnt/$fsname3

echo "Untarring ..."

cd /
echo "Untar directories: root"
cd /mnt/$fsname1
tar xvf $tardir/$fstar1
if [ ! -d /mnt/$fsname1/usr ]; then
mkdir /mnt/$fsname1/usr
fi
if [ ! -d /mnt/$fsname1/boot ]; then
mkdir /mnt/$fsname1/boot
fi
if [ ! -d /mnt/$fsname1/tmp ]; then
mkdir /mnt/$fsname1/tmp
fi
if [ ! -d /mnt/$fsname1/home ]; then
mkdir /mnt/$fsname1/home
fi

cd /mnt/$fsname1/usr
tar xvf $tardir/$fstar1a

cd /
echo "Untar directories: boot"
cd /mnt/$fsname2
tar xvf $tardir/$fstar2

cd /
echo "Untar directories: cdf"
cd /mnt/$fsname3
tar xvf $tardir/$fstar3
```

```
cd /mnt/$fsname3/level3/filter/control
tar xvf $tardir/$fstar3a
cd /mnt/$fsname3/level3/filter/relay
tar xvf $tardir/$fstar3b

cd /
echo "Changing IP address in ifcfg-eth0 ..."
cd /mnt/$fsname1/etc/sysconfig/network-scripts
sed -e s/$fsip1/$ipadr/ ifcfg-eth0 > tempo
mv tempo ifcfg-eth0

echo "Installed node $name; recovered $fsname1 $fsname2 $fsname3; \
    on `/usr/bin/date` " > /mnt/\$fsname1/root/l3install.log

echo "Update mbr"
umount /mnt/$fsname2
mount -t ext3  /dev/$fsname2 /mnt/$fsname1/boot
chroot  /mnt/$fsname1 /sbin/lilo

echo "Un-mounting filesystems..."
umount /mnt/$fsname1/boot
umount /mnt/$fsname2
umount /mnt/$fsname3

echo "Probing floppy disk..."
mkdir /mnt/floppy
mount -t msdos /dev/fd0 /mnt/floppy
flp=`df | grep floppy | wc | cut -f 1`
umount /mnt/floppy
while [ $flp -gt 0  ]; do
 sleep 5
mount -t msdos /dev/fd0 /mnt/floppy
 flp=`df | grep floppy | wc | cut -f 1`
 echo "Still waiting for floppy disk to be removed... $flp : not zero"
 umount /mnt/floppy
done
echo "Floppy drive is now empty."
echo "Done."
echo "The node is now being rebooted..."
sleep 2
reboot
```