LCFG, Datagrid Extra Bits and Bobs.

Steve Traylen

13th March 2002

1 Introduction

A few people have been asking over the last couple of weeks all the extra things that I did with LCFG at RAL to get it working in our environment. So this is some of the experiences that I have had with it and in particular what is needed to get a datagrid testbed running.

This not meant to replace the datagrid documentation. I hope this encourages us to get testbeds up and running quickly. Between us if we share problems I expect we can get things going soon. The software between sites is so similar I expect we will all hit the same problems.

So this is a collection of some missing bits from the documentation and the interesting bits from the existing documentation. Some of it is obvious, some of it less so.

I am extremely happy to help where I can to get people going. I have tried to keep things brief so ask if something does not make sense.

2 Creating an LCFG Server

I think most people have managed to install the LCFG box okay. It is just a Redhat 6.2 with a few extra RPMS and a large repository of datagrid RPMS.

The instuctions I found most useful was the cookbook[1] method written by Julian Blake.

Start the procedure by installing RedHat 6.2 as normal. You need at least 4 Gigs to be available in */opt/local*. Then create the RPM repository.

[1] `http://datagrid.in2p3.fr/distribution/datagrid/wp4/installation/doc/server-install-cookbook/linuxinst26.html`

2.1 RPM Repository

Julian's document was very much written from the perspective of someone being at CERN with access to the AFS volumes and such like. The bits which you need to change are the bits that concern downloading the RPMS from datagrid.

The repository for datagrid RPMS is at `http://datagrid.in2pr.fr/distribution/`. Unfortunately the directory structure in this web repository is not the same structure as repository structure that LCFG expects. I do not know why this is?

So to create a repository I forget sections 3,4 and 5 from Julian's notes and used my own script, `createRepository.sh.txt`. Not particularly clever it just maps a directory on a web server to a directory on your file system and downloads everything in it and does some tidying up.

This script also also uses a Makefile called `Makefile-genheaders.txt` Both of these may well still need editing for your local site. Check before you run them.

In the repository, for each rpm, say *gcc-2.8.i386.rpm*, there is also a file required called *.gcc-2.8.i386.rpm*. This file is also created by the createRepository script.

It is a pain the directory structure is not the same, it would make a life a lot easier if it was. I will get in touch with the relevant people. RSync, FTP or NFS access would also make mirroring simpler as well.

This script and many other things require wget be configured to work , if behind a forced proxy then you need to add an equivalent of *http_proxy = http://wwwcache.rl.ac.uk:8080/* to */etc/wgetrc*.

2.2 LCFG Server RPMS

Now install the LCFG rpms on the server as described at the start of section 8 in Julian's Document.

Carry on to the end of Julian's document except for section 16 about starting mkxprof (Make XML profile) as a daemon. Somewhere elsewhere, I forget where, this is no longer recommended and I agree.

I will add some stuff below about the details of the mkxprof compiler.

3 XML Configuration

To re-cap the basic LCFG idea is that every machine is completely defined by an XML file. There is exactly one XML file for each machine. You should in theory

never have to look at this XML file or even care what XML is but I have had to check the odd value in there at times when trying to work out what is going on.

This XML file only has a depth of about a three levels and every bit of machine configuration is produced from an (object, attribute, value) triplet.

For example the *xntpd* object has an attribute of *serverdomain* which in my case has a value of *rl.ac.uk*. Also *xntpd* has an attribute of *servers* in my case *ntp0*. This example configuration appears in the XML file as.

```
<xntpd>
  <serverdomain>rl.ac.uk</serverdomain>
  <servers>ntp0</servers>
</xnptd>
```

Using LCFG comes down to two distinct tasks. The first is to create this XML file on the server. The second is to convert this XML file into the actual configuration on each machine. The XML file is distributed with an apache server.

3.1 XML File Creation

For each machine you should have a top level configuration file. This is something like. */var/obj/conf/profile/source/gppui* for a machine called gppui.

For the following work in the directory */var/obj/conf/profile/source*

All of the configuration files you now have in */var/obj/conf/profile/source* are completely out of date and need updating against the CVS repository at Marianne.

To do this loosely follow the instructions [2] but what you want to do exactly is add the following to your */.ssh/config* file.

```
Host datagrid.in2p3.fr
Protocol 1
Port 2222
PasswordAuthentication yes
RSAAuthentication no
Compression no
User anoncvs
```

Set an environment variable, *CVS_RSH=ssh ; export CVS_RSH*

Check out edg-release version 1.1.1

[2] http://datagrid.in2p3.fr/HOWTO/CCIN2P3CVSaccessHOWTO-3.html

cvs -d datagrid.in2p3.fr:/cvs checkout -r v1_1_1 edg-release

(Presently v 1.1.1 has not been released and tagged in CVS, you could get the current one by leaving out *-r v1_1_1* but remember you do not want to go past *v1.1.1* once it has been tagged or else you will be entering development releases. I expect v1.1.1 will be tagged in a matter of days).

This will create a directory edg-release containing all the LCFG files that you require.

You should copy everything from *edg-release/source* to */var/obj/conf/profile/source/* and everything from *edg-release/rpmlist/* to */opt/local/linux/6.2/rpmcfg*.

Copy the example User Interface file *testbed004* to a file called gppui or whatever you are going to call your user interface. Edit this file and change the hostname in it. The user interface is the simplest machine and the best to start with.

If you look at gppui the top level file that defines this particular machine you will see something like this.

```
/*
  gppui
  ================================================
  A USER INTERFACE SAMPLE
*/

/* Host specific definitions */
#define HOSTNAME gppui

/* Some useful macros */
#include "macros-cfg.h"

/* Site specific definitions */
#include "site-cfg.h.legnaro"

/* Linux default resources */
#include "linuxdef-cfg.h"

/* LCFG client specific resources */
#include "client_testbed-cfg.h"

/* Computing Element specific resources */
#include "UserInterface-cfg.h"
```

It is worth reading these files , most of the configuration within these can be guessed.

One small problem here in that a lot of the documentation about individual objects

and their attributes, eg xntpd, is not available until you get a client up and running. Very soon I will ask someone to place this documentation on line for datagrid. It is on the LCFG site[3] as well but there is no guarantee that these objects are at the same version.

This individual machine file,*gppui* must be converted into `gppui.xml` and this is done using *mkxprof -v -A gppui*. This is the only flags I have ever used with *mkxprof*. There are lots more and the program has a man page though I have never had need to read it yet.

The command *mkxprof* does two things, applies a C pre processor (CPP) to gppui to execute all the #include and #define statements within this file.

This C pre processed file is then converted to the XML file gppui.xml assuming there were no errors. The errors at this point are usually along the line of not being able to locate a file if you have changed things.

Other things that you defiantly need to change are the contents of *site-cfg.h.legnaro* to *site-cfg.h.ral* and edit the include line in *gppui* that points to this file. As for what the values in this file should be most of them are local to your site but ask for those ones that are causing problems.

At this point I would attempt an install of the client box gppui with the floppy disk that you made earlier.

The format of the files that are included within gppui are a custom format to LCFG. Check the man page for *rdxprof* (Read XML Profile) but for our earlier xntpd example we have the two lines.

```
xntpd.serverdomain      rl.ac.uk
xntpd.servers           ntp0
```

so it is

```
object.attribute        value
```

Absolutely all configuration in LCFG is defined by this format in these files. This would be the case other than the fact CPP allows you to use *#define* and this echoed back later as its value. So

```
#define NTPBOX ntp0
xntpd.servers           ntp0
```

is the same as above

This does make life confusing as there are two layers of configuration. The initial cpp language and then the lcfg language which is what is processed into the XML.

[3] `http://www.lcfg.org/doc/\#compdoc`

4 Overriding Datagrids Configuration

The idea is that all site configuration should be in *site-cfg.h.ral* and everything else remain static.

This is not true and very quickly you realize that things in say *client-testbed.h* are not correct for your site.

An obvious example is the *auth.rootpwd*. The value of this is the hash of your root password which you will want to set locally. You could edit client-testbed.h and enter your own hash. This will work but a problem arises when datagrid release a new client-testbed.h then you will have to merge the differences together which is fairley horrible.

Instead add a line to *gppui* such as `#include`"ral-extras.h". In the *ral-extras.h* you can add a line

```
+auth.rootpwd                    56jasdkGT21Z1
```

The + at the start of this line says override anything that was set earlier.

As an aside here more that one person has asked how to create a hash of a password. The way I do it for a password *password*.

```
 perl -e 'printf crypt("password","W2")."\n"'
```

the second argument to crypt is the salt and is your choice but must be 2 chars. There may be a better way and I am nervous around telling people how to set their root passwords.

In fact I have a lot of other *#includes* which are local to my site, one for UIs , one for WNs, ... You may find that you may need more as well, it is a matter of practice. I suppose at some point we could have a UK-site.h or something eventually.

So my *gppui* file in fact looks looks like this.

```
/*
  ================================================
  A USER INTERFACE SAMPLE
*/

/* Host specific definitions */
#define HOSTNAME gppui05

/* Some useful macros */
```

```
#include "edg/macros-cfg.h"

/* Site specific definitions */
#include "ral/site-cfg.h.ral"

/* Linux default resources */
#include "edg/linuxdef-cfg.h"

/* LCFG client specific resources */
#include "edg/client_testbed-cfg.h"

/* Computing Element specific resources */
#include "edg/UserInterface-cfg.h"

#include "ral/ral-extras.h"
#include "ral/ui-ral-extras.h"

/* Local Home Directories */

/* This is only on this machine and will eventually
  go so we put this here.  */
EXTRA(nfsmount.nfsmount) gpphome
+nfsmount.nfsdetails_gpphome /home/gpp gppmast:/home/gpp rw
```

So I have all of the datagrid supplied configuration in a directory *edg* and all of my configuration supplied in directory *ral*. There is nothing correct about it this way, it is just my way.

Another method for achieving the same thing that I think NIKHEF is done is to use CVS to maintain your differences to the checked out datagrid version. If you are happy with CVS this may be good for you do things as well.

5 Site Configuration

You can now do everything to apply local configurations to your boxes while maintaining the core of the software release and updates from the European datagrid.

Now we go through some of specific configurations that you may have to tackle.

5.1 Ethernet Cards

There are two issues here, the Ethernet driver on the floppy disk and the one that is configured to be loaded at reboot.

I require an e1000 driver for my card so I need to get *alias eth0 e1000* into my */etc/modules.conf* file somehow.

I want my configuration to be preserved if there is a new datagrid release so I want to add this *ral-extras.h* file.

```
+update.modlist                e1000
+update.mod_e1000              alias eth0 e1000
```

This configuration says replace the list of modules I want to load with e1000 and then use the correct line in the *modules.conf*. Note the entry e1000 here is just a label. What is above is exactly the same as.

```
+update.modlist                  zebedee
+update.mod_zebedee              alias eth0 e1000
```

but makes less sense of course.

5.2 Boot Process

The boot process for LCFG should go like this:

- Floppy disk boots.
- Kernel Loads.
- A DHCP request finds the name of the LCFG server, eg *gpplcfg*
- NFS Mount *gpplcfg:/opt/local/linux/installroot/6.2* as the root file system.
- /etc/dcsrc is run. Thats gpplcfg:/opt/local/linux/6.2/installroot/etc/dcsrc
- This scripts downloads the XML profile and starts to interpret it.
- Partition the disk
- Install RPMS
- Reboot twice and configure some things on the way.

You may find a problem determining the name of your lcfg server and if so just hard code *T151=http://gpplcfg.gridpp.rl.ac.uk/* into the */etc/dcsrc/* script.

If you are having problems determining your DHCP server or you do not have a DHCP server you can get around a lot these problems by passing options to the kernel at boot time. As a minimum you need.

```
LILO: linux root=/dev/nfs init=/etc/dcsrc
```

This is what is hard coded into the floppy disk to cause a root file system to be mounted.

If you have to there are a load more options[4] that can be passed to the kernel such as which DHCP server to use, what IP address to use, the name of DNS server and so on.

If the driver you need for your floppy is not compiled into the kernel then you need to compile one up. There is a minimum[5] set of things you will have to have compiled into your kernel.

If building a kernel is proving to be difficult or its not worth it because you only have two machines then use a standard Redhat bootnet disk. At the LILO prompt just run *LILO: linux rescue*. Answer the questions and then you will get a prompt. At this point you can run the following to get things going.

```
# mkdir /tmp/mnt
# mount gpplcfg:/opt/local/linux/ /tmp/mnt
# chroot /tmp/mnt/installroot/6.2
# /etc/dcsrc
```

If you do this you will have hard to code the T151 variable into the dcsrc script on the LCFG box.

5.2.1 Partitions

The partition software can only create primary partitions but more importantly it can only delete primary partitions. If the box you are trying to install on already has extended partitions then you need to get rid of them somehow. A Redhat rescue disk is one solution.

[4] `http://www.linuxhq.com/kernel/v2.2/doc/nfsroot.txt.html`
[5] http://www.linuxhq.com/kernel/v2.2/doc/nfsroot.txt.html

5.2.2 Kernel Version

You may find that although you have an SMP box that you are only running a single processor kernel after the LCFG install. Or like me you may need to run a completely different kernel because of some hardware you have.

To force LILO to load a particular kernel use.

```
+update.lilokernel              /boot/vmlinuz-2.2.19-6.2.12.ral_1smp
```

This will load which ever kernel you want. There is a correct way about getting the SMP kernel to be default kernel, look through the iteam mailing list to find out. It would be a lot of hard work for not a lot though.

5.3 RPM Lists

For each machine there is also a list of RPMS to be installed on that machine.

5.3.1 Which List of RPMS

In the profile for each machine there a few attributes to the update object that define which is the set of RPMS to be installed.

They are

```
+update.rpmcfg                  client_testbed
+update.rpmcfgdir               /export/local/linux/6.2/rpmcfg
#define RPMDIR                 /export/local/linux/6.2/RPMS
+update.rpmdir                  RPMDIR/release:\
RPMDIR/updates:\
RPMDIR/external:\
RPMDIR/LCFG:\
RPMDIR/globus2_beta21:\
RPMDIR/security:\
RPMDIR/globus2_config:\
RPMDIR/WP1:\
RPMDIR/WP2:\
RPMDIR/WP3:\
RPMDIR/WP4:\
RPMDIR/WP5:\
RPMDIR/WP6:\
RPMDIR/WP7:\
RPMDIR/WP8/Alice:\
```

```
RPMDIR/WP8/Atlas:\
RPMDIR/WP8/CMS:\
RPMDIR/WP8/LHCb:\
RPMDIR/WP9:\
RPMDIR/WP10:\
RPMDIR/apps_common
```

The *update.rpmcfg* and *update.rpmcfgdir* attribute define the location of a file that contains a lists of RPMS to be loaded onto this particular machine. This file is also preprocessed with a C pre processor and so can optionally and probably should contain *#include* statements to include blocks of rpms. The *update.rpmdir* attribute gives a colon delimited path to the RPMS repositories. If you add an extra repository you must change this line.

Remember that for *gppui* say *+update.rpmcfg* may be defined in every file that is included from *gppui*. Its the last value of *+update.rpmcfg* that counts though. Obviously this is true for all attribute values pairs.

5.3.2 Testing the Consistency of an RPM List

If you have a bunch of RPMs it is a good idea to check that they can live together before you precede.

There is Makefile in */opt/local/linux/6.2/rpmcfg* that is supposed to this but it it not that useful. If you want to test the consistency of a file *UI-rpm* which is list of RPMS that you want to verify could be installed where *UI-rpm* may optionally contain some #include statements.

```
$ mkdir -p /tmp/UI-rpm/var/lib/rpm
$ mkdir -p /tmp/UI-rpm/dev
$ mknod  /tmp/UI-rpm/dev/null  c 1 3
$ rpm --initdb --root /tmp/UI-rpm
$ RPMPATH=/opt/local/linux/6.2/RPMS
$ /usr/lib/cpp -traditional -undef UI-rpm > UI-rpm.rpmcfg
$ updaterpm -r /tmp/UI-rpm -R $RPMPATH/updates:$RPMPATH/release -s UI-rpm.rpmcfg
```

Obviously you need to expand the path list for the RPM repositories to the same as your *update.rpmdirs* attribute is set to in your profile for the machine that you intend to use this list for. Updaterpm has an easy man page.

So really there are two files that define a machine. The profile file in */var/obj/conf/profile/source* and the rpmlist in */opt/local/linux/6.2/rpmcfg* although the profile points at the rpmcfg file.

5.4 The End

I will add to this as much as I can over the next few days as I review what it was that I did myself on the various boxes. The things I have included so far concern how to get something running rather than a testbed. That can follow...

I will just send some individual HOWTOs to the tb-support[6] list.

Obviously people are welcome to add answers to any problems they encountered while getting LCFG to go.

[6] http://www.jiscmail.ac.uk/lists/tb-support.html