Dynamic Probes and Linux Trace Toolkit

Dynamic Probes and Linux Trace Toolkit are now available on zSeries (s/390) Linux.

Description:

Dynamic Probes (dprobes) is a tool that can be used to insert software probes, dynamically into executing code modules. When a probe is fired, a user written probe-handler is executed. The probe-handler is a program written in an assembly-like language, based on the Reverse Polish Notation (rpn). Instructions are provided to enable the probe-handler to access the hardware registers, system data structures and memory. The data may be collected and written to klog or traced using the Linux Trace Toolkit, depending on options available and / or specified.

Linux Trace Toolkit (ltt) is a suite of tools designed to extract program execution details from the Linux operating system and interpret them. Specifically, it enables its user to extract processor utilization and allocation information for a certain period of time.

Ltt can be used in conjunction with dprobes as the logging component for probe data captured. It provides a visualizer component which allows for easy viewing of the data.

This website includes information on where to obtain more information on ltt and dprobes for zSeries (s/390) Linux, consolidated kernel patches for ease of installation, and examples of how to use dprobes and ltt together for application debugging.

Current Versions: dprobes 3.6.3; ltt 0.9.5

License: dprobes - GPL; ltt - GPL

Author of S/390 patches: Michael Grundy - dprobes;Theresa Halloran - ltt.

Status: Seperate kernel patches for dprobes and ltt available at websites below. Special patches are linked to here for ease of installation and support with the SuSE SLES-7 2.4.7 Linux kernel.

Obtaining and installing dprobes and ltt for zSeries (s/390) Linux: Dprobes and ltt both require a patch to the Linux kernel, in addition to their other components. Each package may be obtained seperately from the websites below. As part of each package, a kernel patch is provided. For ease of installation a set of well behaved kernel patches (no conflicts or reject resolution necessary) are referenced here.

Dprobes home page: http://oss.software.ibm.com/developer/opensource/linux/projects/dprobes/
Linux Trace Toolkit home page: http://www.opersys.com/LTT/

Things you should download:

- **LTT 0.9.5 package:** http://www.opersys.com/ftp/pub/LTT/TraceToolkit-0.9.5a.tgz
- **Dprobes 3.6.3 package:** http://oss.software.ibm.com/developer/opensource/linux/projects/dprobes/releases/dprobes-v3.6.3.tar.gz
- **LTT 2.4.7-SuSE patch:** http://opersys.com/ftp/pub/LTT/ExtraPatches/patch-ltt-0.9.5-linux-2.4.7-SuSE.gz

Dynamic Probes and Linux Trace Toolkit

The extra patches will apply to the SuSE 2.4.7 kernel without any conflicts.

Installing the kernel patches:

The following procedure for installing dprobes and ltt on a SuSE kernel (SLES 7.2) is provided:

> ***Note: We use a FTP install configuration, details will vary slightly for NFS or other install media***

Install the SuSE kernel source.

- Bring up YAST (you must be logged in as root)
- Go into Package Management
- Enter userid and password on FTP settings screen (FTP install only)
- Choose "Change or create configuration"
- In the Series selection panel, choose series **d** (Development)
- On the package selection panel, choose Kernel-Source
- Hit F10 to finish; if there are package dependencies, you may have to hit F10 again to resolve them.
- When you return to the Installation menu, choose "Start installation".

> ***Note: We use a fairly simple build process working in /usr/src. We've put in some safegaurds for you in case things don't work out as planned. The instructions also take into account that SLES-7 shipped with some OCO modules (e.g. The OSA token ring driver) and attempt to avoid any kernel version problems. If you run into problems, and have followed the instructions carefully, the recovery instructions should be able to bail you out quickly.***

Before we start patching, configuring and building your new kernel, let's take a moment and back up some important files.

Change directories (cd) into /lib/modules. You will see a directory: 2.4.7-SuSE-SMP. This contains all the loadable kernel modules. Since we will be making and installing a new set of kernel modules, we should back this directory up.

Type:

cp -a 2.4.7-SuSE-SMP 2.4.7-SuSE-SMP.save

Now move over to /usr/src. Expand the Dprobes package by typing:

tar xzf dprobes-v3.6.3.tar.gz

This will create a dprobes-v3.6.3 directory tree. Then expand the LTT patch by running:

gunzip patch-ltt-0.9.5-linux-2.4.7-SuSE.gz

Go back to /usr/src/linux now. You can apply the DProbes/LTT patches by running:

patch -p1 < ../dprobes-v3.6.3/patches/dprobes-v3.6-247-SuSE-core-1.patch

```
patch -p1 < ../dprobes-v3.6.3/patches/dprobes-v3.6-247-SuSE-s390-1.patch
patch -p1 < ../patch-ltt-0.9.5-linux-2.4.7-SuSE
```

The patches should lay down with no warnings on a stock SuSE 2.4.7 kernel. If you have added any other patches, there may be some fuzz or offset warnings. If you got any reject messages, the kernel won't build correctly in the current state. Rejects can be caused by other patches being applied prior to this patch, usually they can be resolved quickly by viewing the source file and the rej file.

After the patches have been applied you'll need to configure and build the kernel. SuSE includes a nice little feature in their kernels: A copy of the running kernels configuration in /proc/config.gz.

You may have a different configuration you wish to use, which is fine, just copy that into the /usr/src/linux directory and skip over the steps where we create the config file.

Change directories (cd) to /usr/src/linux. Create a copy of the current configuration by running

```
gzip -dc /proc/config.gz > .config
```

Then this config file will need to be updated with the trace and dprobes options.

```
make oldconfig
```

All the config options set in that file will scroll by, you will be stopped for two new options:

```
Kernel events tracing support (CONFIG_TRACE) [N/y/m/?] (NEW) y
...
IBM Dynamic Probes (CONFIG_DPROBES) [N/y/?] (NEW) y
```

Answer **y** to both options.

The next step is to run

```
make dep
```

Then:

```
make
make modules
```

If you have made it this far without any errors, you are ready to install your new kernel. Let's start by installing the modules by typing:

```
make modules_install
```

Now we will actually install the kernel. SuSE has a nice layout that makes it easy to organize different kernel versions/configurations. We're going to keep with that theme and do the following (as root):

```
cd /boot
mkdir k_dp-ltt
cd k_dp-ltt
cp /usr/src/linux/arch/s390/boot/image .
cp /usr/src/linux/arch/s390/boot/*.boot .
cp /usr/src/linux/System.map .
cp /usr/src/linux/include/linux/autoconf.h .
cp /usr/src/linux/include/linux/version.h .
cp /usr/src/linux/.config .

cd ..

rm kernel
ln -s k_dp-ltt kernel
zipl
```

That's it. IPL the system (init 6 is a nice way). You're original kernel will still be in the /boot/k_deflt directory if you need to recover or go back to your original setup. (Reverting to your original kernel is the same as recovering, except you won't have to boot up the installation kernel to do so).

If you're system didn't boot properly jump down to the recovery section, otherwise it's time to build the dprobes and ltt usermode programs. Detailed instructions are included with the dprobes and ltt distributions, but we'll give you a quick note to help you along. To make the dprobes command line utility must have flex and bison installed. To build the visualizer component of LTT you need to have the GTK development packages installed. (Through YaST install glib packages, gnome-libs, and gnome-libs-devel).

When kernel mods attack (How to recover)

Boot off the SuSE installation kernel

If you are running under VM you can complete this task from the console and don't have to configure networking. Choose 0) No Network. If you are on an LPAR or can't stand to work from the console, configure networking and then telnet to your machine.

Install the dasd driver
```
modprobe dasd dasd=<device number that contains root>
```
Mount the volume you are going to recover, you may want to run fsck on that device first
```
mount /dev/dasda1 /mnt
```
chroot to the dasd you are recovering, helps keep things straight in the mind
```
chroot /mnt
```
Go into the /boot directory and reset the kernel symlink to the stock kernel
```
cd /boot
rm kernel
ln -s k_deflt kernel
```
Then run zipl to set the dasd up for booting the stock kernel
```
zipl
```

break out of the chroot environment (on a 3270 connection type the caret character separately)

```
^d
```

Then just make sure everything is written out to disk, unmount and halt.

```
cd /
sync
umount /mnt
halt
```

You are now ready to ipl the system with the stock kernel back in place.

Installing dprobes (non-kernel portion):
Download the dprobes distribution from: (you should already have done this)
http://oss.software.ibm.com/developer/opensource/linux/projects/dprobes/releases/dprobes-v3.6.3.tar.gz

Unpack the tarball, and go to the dprobes-v3.6.3/cmd directory. Just run **make** and **make install** to get the dprobes command line utility and man files installed. You may have to install yacc and bison, if they aren't already installed on your machine.

To test out your newly created dprobes command line interface you will need to get the drpobes modules installed and the dprobes device file created. All commands must be run as root. To install the dprobes module run:

```
modprobe dp
```

To create the dprobes device file, issue the following commands:

```
rm -f /dev/dprobes
MAJOR=`cat /proc/devices | awk '$2=="dprobes" {print $1}'`
mknod /dev/dprobes c $MAJOR 0
```

Then you can do a basic test by issuing:

```
dprobes -q -a -x
```

It should return the message: **No information returned**. Any other message will indicate a problem.

Installing ltt (non-kernel portion):

Download the TraceToolkit-0.9.5.tgz from: http://www.opersys.com/LTT/downloads.html

The installation instructions for ltt are on-line at: http://www.opersys.com/LTT/Help/ltt-installation.html

The instructions are very simple and are included here. To install the user portion of ltt, after un-zipping and un-tarring the package, issue:

```
./configure
make
make install
```

createdev.sh

Running dprobes and ltt:

Description: To run dprobes and ltt together, the basic sequence of things is:
- Issue ltt command to start tracing of custom type traces (which is what dprobes traces are).
- Insert the probe point you want to trace into the code.
- Drive the scenario to hit the probe point and the trace will be collected.
- Remove the probe.
- Turn off ltt tracing. (kill the tracedaemon).
- View / analyse trace record using ltt visualizer and / or data dump facility.

Examples:

To start the tracedaemon to collect dprobes traces issue:
tracedaemon -eCSTM -eNEWEV /dev/tracer <trace-filename> <proc-filename>

Note: <trace-filename> and <proc-filename> are filenames of where you want to put the output from the ltt trace.

Creating a test probe point handler and inserting it into a usermode program:

We'll start by creating a probepoint definition file. Using your favorite editor create a file named **utest.probe** and paste the following code into it:

```
name = "foofoo"
modtype = user
major = 1
jmpmax = 32
logmax = 100

offset = funky
opcode = 0x90bf
minor = 1
pass_count = 0
max_hits = 1000

push w, 0x7fffffff
push r, pswaddr
and
log u32, 0x01
push r, pswmask
log u32, 0x01
exit
```

This test probe point is set to fire on the first instruction of the function *funky()*. We have to specify 16 bits of the opcode and operands so that dprobes can double check that we are inserting the probe where we expected. In this case we have 0x90bf, which is the first halfword of a RS instruction format (STAM 11,15,...). The probe point handler is defined in the last section of the file. In this example when the probepoint fires we will push the value 0x7fffffff onto the stack, push the psw address value onto the stack and then and the two values together (the result going back on to the stack). The psw address with the high order bit stripped off is then logged. The psw mask value is then retrieved and logged. Detailed information about the format of a probe point definition file can be found by typing:

man dprobes.lang

Now we need to create our test program. Again using your editor of choice, create a file named foofoo.c and paste in the following code:

```
#include<stdio.h>


void funky(int i);
main(void)
{
        int i;

        printf("main  0x%08x\n", &main);
        printf("funky 0x%08x\n", &funky);
        getchar();
        for (i=0; i<1; i++)
                funky(i);
        printf("Halt for next run 0x%08x\n", &funky);
        printf("next instruction? 0x%08x\n", *((unsigned long *)&funky));
        getchar();
        for (i=0; i<1; i++)
                funky(i);
        return 1;
}

void funky(int i)
{
        printf("int i=%d\n",i);
}
```

foofoo.c can be compiled as follows:

gcc -o foofoo foofoo.c

Note that we haven't compiled with the -g option, this isn't necessary for dprobes like it is for other debuggers

Now we can go ahead and insert the probe. The following command line specifies the probe point definition to compile and insert (-i). We've also specified some additional information to collect (-l): the user id, the process id and the PSW. The CPU number is collected by default. Then finally we are indicating that logged information will go to LTT. The default is to log to the kernel log.

dprobes -i utest.probe -l UID PID PSW --log-target LTT

In this case, running program foofoo will drive the probe to hit (Well, actually when you hit enter after foofoo pauses). foofoo pauses after printing out a hex dump of the next instruction word, hitting enter allows it to continue to the point where the probe will fire.

Data is collected, so remove the probe:
dprobes -r -n foofoo
Turn off ltt tracing: ps -A, kill the pid for the tracedaemon.

tracevisualizer brings up the gui; select trace and proc files. Visualizer uses X-Windows so you must have an x-windows server on your PC and must set your DISPLAY variable.

Document Authors:
Mike Grundy
Theresa Halloran
Karin Laeben