S3FNet-LXC README

Vladimir Adam

April 28, 2015

1 Overview

This document serves as supplementary documentation to S3FNet-LXC. For more information about TimeKeekeper, see `s3fnet-lxc/dilation-code`.

2 Repository Contents

base: This folder contains the original base S3FNet source code as well as the modifications added to realize S3FNet-LXC. See `https://s3f.iti.illinois.edu/` for the original source code.

csudp: This folder contains the code for the UDP client and server. A DML file can give LXCs individual commands and most of the sample provided DML files utilize the UDP client and server.

dilation-code: This folder contains all the relevant code for TimeKeeper. `dilation_module.h` defines how many CPU cores are used in experiment via the macro `EXP_CPUS` that needs to be redefined depending on the physical hardware. Note, its best to leave a few CPU cores for background work. `dilation-code/scripts/timekeeper-stopExperiment` executable is called before the launching an S3FNet-LXC simulation in order to make sure everything is initialized from scratch. For questions about the TimeKeeper code, contact the original author Jereme Lamps at `jereme.lamps@gmail.com`. At the time of writing of this document, this repository contains the latest version of TimeKeeper.

experiment-data: This folder will contain all the output from any given experiment into `experiment-data` when an S3FNet-LXC simulation *successfully* finishes.

kernel_compilation_config: This folder contains the `.config` file used when compiling the custom Linux kernel. Note, it has to be renamed accordingly if used for kernel compilation.

lxc-command: This folder contains the code necessary used to pass commands into LXCs via a `reader`. When an LXC is created, it is started as a daemon running the executable `reader`. `reader` is a binary which continuously listens for commands via a named pipe. When a command is set to an LXC, the command is written to an appropriate named pipe which is created in `/tmp`. See `sendCommandToLXC()` in `lxc_proxy.cc` to see how a command is sent to an LXC. Note, it is recommended to send commands that will finish executing during the course of the simulation, as the reader may not immediately flush the output out to the file. For example, `ping 10.10.0.9` will only terminate manually. For questions about the reader code, contact the original author Jereme Lamps at `jereme.lamps@gmail.com`.

lxc-scripts: This folder contains bash scripts used to create and destroy the LXCs, Linux bridges and TAP devices.

original_s3fnet_base: This folder contains the source code of the *original* base S3FNet on which S3FNet-LXC was based on. To best understand the changes made, I recommend diffing (Meld Diff Viewer recommended) the contents of base and original_s3fnet_base to see the changes.

3 Installation and Running

After checking out the source code from the repository, one must first install the modified Linux 3.10.9 kernel. (NOTE: For best results, check out the repository in the home directory. This is because the LXCs launch the UDP Client/Server executables by sending `~/s3fnet-lxc/csudp/client [args]` to an LXC). This can be done by calling `sudo dilation-code/kernel_setup.h`. That script does the following:

1. Compiles the scripts inside `dilation-code/scripts`
2. Downloads the Linux 3.10.9 into `/src` directory.
3. The Linux archive containing the source code is decompressed and the patch is applied.

For the kernel compilation, there are 2 options:

1. `cd /src/linux-3.10.9` and follow the tutorial at `http://mitchtech.net/compile-linux-kernel-on-ubuntu-12-04-lts-detailed/` to compile and install the custom kernel implementing TimeKeeper.
2. There is a kernel configuration option `(kernel_compilation_config/vlad_desktop_config)` which should have everything preconfigured. Copy `vlad_desktop_config` into `/src/linux-3.10.9` and rename it to `.config`. Optionally, it is possible to call `sudo make menu config` and rename the string appended to the end of the kernel. Then call:
   - sudo make
   - sudo make modules_install
   - sudo make install

Next open and edit `s3fnet-lxc/base/s3fnet-definitions.h` and modify **PATH_TO_S3FNETLXC** with the absolute path pointing to the root folder of the git repository. Furthermore, modify **PATH_TO_READER_DATA** with the absolute path pointing to `data` folder inside the top of the repository.

Next, and execute the bash script `./configure\_s3fnet-lxc.sh`. This will compile the TimeKeeper module, the UDP client/server inside `s3fnet-lxc/csudp/` as well as the reader which is executed by the LXCs.

Next, cd into `base` and compile S3FNet-LXC using `make fullbuild`. This will compile all of S3FNet-LXC including the DML binary. For more information about building see `https://s3f.iti.illinois.edu/usrman/installation.html#build`.

At this point, you should be able to execute `sudo make examplerun`. This will call `timeKeeper-stopExperiment()` which will clean up the experiment (assuming it was run previously). Next, it will `cd` into `s3fnet-lxc/base/s3fnet/test/lxc_tests/small_2_udp` and use the `dmlpart` and `dmlenv` to create necessary DML files. Finally, it will call `s3fnet/s3fnet` passing in the DML files which create the model. See `https://s3f.iti.illinois.edu/usrman/installation.html#running-s3f-s3fnet-experiments` to how it was done using base S3FNet. S3FNet-LXC

uses an almost identical mechanism.

At this point, the simulation will beginning running. An instance of the LXC Manager is created. Afterwards, S3FNet parses the DML files building the model and creating LXCs for any hosts that are emulated. All the LXCs are frozen at time T and the timelines begin individually executing S3F events and advancing LXCs. The simulation should run for about 80 seconds.

4 s3fnet-LXC/base/tklxcmngr

`s3fnet-LXC/base/tklxcmngr` contains the code implementing the LXC Manager as well as the LXC Proxy. It also includes `TimeKeeperFunctions.cc/h` and `utility_functions.cc/h` which originate from `dilation-code/scripts`. These files contain functionalities provided by TimeKeeper to individual control the advancement of LXCs. Note, the function `progress(...)` in `TimeKeeperFunctions.cc` was slightly modified to change from using a `select()` call to a `poll()`. This was done in order to support more than 1024 file descriptors opened by a single process.