This document presents a set of advices that should be taken into consideration if you want to install NXE somewhere on your system.

1 Requirements

It is necessary to have the following libraires installed for NXE to work properly. They are typically all available as packages in your favorite GNU/Linux distribution (Debian for example).

- python (version > 2.3)
- LibExpat and its python binding (XML parsing library)
- OpenSSH client
- Paramiko, a python SSH2 library

Please also note that the following packages are optional in the execution of the experiments through NXE, but that portion of code should be commented out (graph generation in NXE.py) to avoid errors at the end of the execution.

- GNUPlot (version > 4.0)
- GNUPlot.py, a python binding for GNUPlot

It might also be a good idea to have Bash installed if you plan to use some of the scripts that are in the script/ directory. There isn't a lot of bashisms used but it is more convient.

2 SSH

Currently, the SSH related options in NXE are not implemented.

NXE is assuming that you are able to connect to the remote locations using a public key without a passphrase. It is mandatory to do so, ssh-agents because they are not propagated by NXE, and your experiments will fail if you don't do so.

Please also note that with the way the paramiko library connects (not a login shell), you must be very careful with the (i.e. try to avoid relative paths if you don't know what you are doing) the scripts name you are providing in the or allow the modification of variables in the sshd configuration). You also may need to create a (pseudo-)tty by yourself.

3 Cleaning up a NXE session

If you forgot to set the “cleanall” flag in your scenario file, then it might be possible that some temporary files are left at the end of the execution.

NXE is creating the following files (relative to the G5K context and the deployment method used there):

On the end-nodes, logs related files may be created in the local /tmp folder. In case the NXE session is used several times, they are also copied in the /tmp/old folder after the log retrieval, so you can retrieve a previous version of your logs there

On the frontend nodes, files related to the deployment are created in the local /tmp folder.

NXE_id_job_pid_cluster Indicates the number of the OAR reservation number for the NXE process *pid* on the cluster *cluster*

OAR_reservation_uniq_pid_cluster Provides the list of hostnames in the OAR reservation *reservation* for the NXE process *pid* on the cluster *cluster*

output_kadeploy_pid_cluster log of the deployment performed by kadeploy for the NXE process *pid* on the cluster *cluster*

On the node where the NXE process is launched, a temporary folder is created (typically /tmp/NXE_pid , but the location of the tmpfolder root can be changed using the *tmpfolder* flag in the config part of the XML input file) and is used to store any temporary files used during a run of NXE.

After the execution of a run of NXE, the log files and the result graphs are stored in an archive folder (typically the archives folder in the NXE folder, followed by the date corresponding to the start time of the instance of the NXE program, the location can be changed by used the *archivefolder* flag in the config part of the XML input file)

4 Scripts conventions

The following convention[1] is used when scripts complete their task: it is required taht they send back a string “OK!” (resp. “KO!”) if everything was allright (resp. something went wrong). When a “KO!” status is returned, NXE generally issues a warning to the user depending on the estimation of the gravity of the situation by

[1] it is necessary to do so, as most of the scripts are executed via a ssh remote command and the return code we get is not the one we are interested in

the program. The execution might abort if such a status is returned during a vital stage of the execution (*i.e.* reservation, deployment,. . .).

There is only one exception, the script used to perform the nodes reservation is required to send back the names/IP adresses of the nodes that will be involved in the scenario, one by line. If multiple columns are provided, it is assumed that the first one corresponds to a control interface of the node (used to issue the commands to the node) and the second one to do the experiments (the other columns are ignored but could be used for later usage of NXE).

The following parameters are reserved and corresponds to parameters that are send by the NXE to the scripts.

- –reset : indicates that the script receiving this command should do the reserve of its purpose and/or restore the default configuration (*i.e.* the bandwidth limitation script should remove the limitation used).
- –delete : indicates that the script should release the resources corresponding to the current NXE instance**unification avec le flag –reset ?**
- –timeout timer: indicates that the script receiving this command should stop after at most after *timer* seconds and should kill all the launched processes
- –pid pid: reference of the NXE instance used
- –cluster site: reference of the site concerned by the command (reservation + deployment)
- –nodes number: number of nodes to reserve for a given site (reservation)
- –date date: used for advanced time reservation
- –reservation-number: reusing a previous instance of NXE
- –sleep time: time to sleep before checking the nodes allocated(advanced time reservation)
- –interval string: serie of inter-arrival times (sleep) between two consecutive flows
- –size string: serie of flows size
- –target host: host that the script will try to communicate with

(best way to do is to eat every parameter and don’t care about those you don’t care ?)