Setting up a Linux Cluster
(for use with Telemac)

last modified: 04.11.2003

comments and questions to:
Oliver Göthel
goethel@hydromech.uni–hannover.de

1. Software

The following software was used for the examples in this manual:

Suse Linux 7.1/7.3 (www.suse.com)

–Network File System (NFS)

–Network Information Service (NIS)

–Secure Shell (SSH)

MPICH 1.2.5 (www–unix.mcs.anl.gov/mpi)

NAG Fortran 95 (www.nag.co.uk or www.nag.com)

It doesn' has to be Suse Linux. Any other Linux distribution (e.g. RedHat) will work as well. The Software for the Network File System, Network Information System and the Secure Shell is part of the Linux distribution. The MPI software can be downloaded from the internet (don't use the one which comes with the Linux distribution).

2. Installation and Configuration

The following software needs to be installed:

on the server:

–NFS–server

–NIS–server

–MPICH

–Compiler f95 and gcc

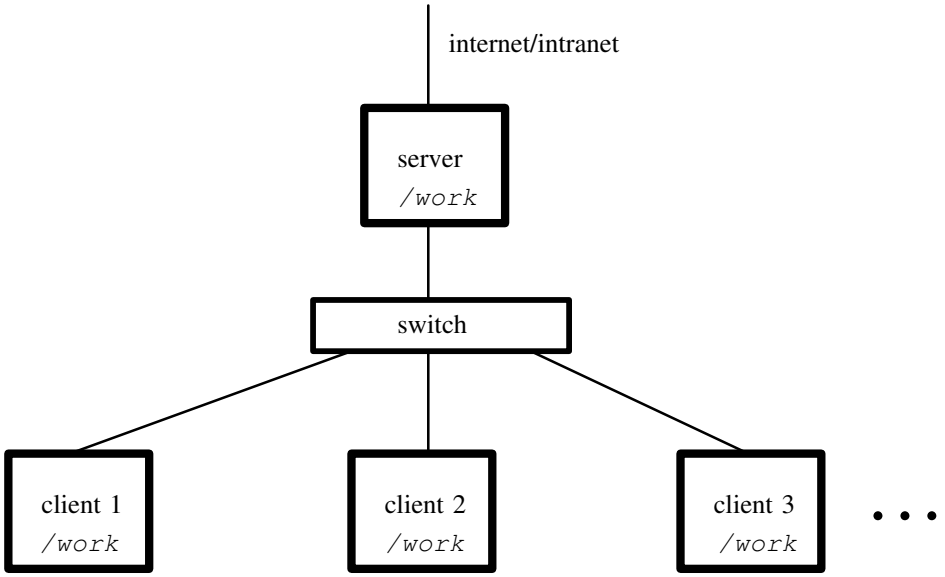
on the clients:

–NIS–client

–NFS–client (is already a part of the system –> doesn't need to be installed)

2.1 Configuring Linux

The structure of the cluster may look like this:



The directory */work* is located on the server and then exported with NFS to the clients so that */work* is available all over the cluster.
The Network Information Service is used to spread the details of user–accounts (username, password, homedirectory etc.). Configure the server as NIS–server which exports the *passwd*– and the *group*–file to the clients. The clients are configured very easy by just activating the client software and then telling the client the IP–address of the server and the cluster domain which must be defined when the NIS–server is configured.
It's advisable to use yast2 (Suse) to configure NFS/NIS on server and clients. The Linux manual may help with more details on this topics.
After configuring NFS and NIS it's time for a test. A user should be able to login on the clients with the same username and password as on the server. The homedirectory must be located in the directory which is available over NFS (e.g. */work*).

2.2 Secure Shell

In order to start a program on the server and execute it on a client, the remote login without prompting for a password has to work. In the homedirectory is a directory called *.ssh*. If a keyfile (*identity*, *id_rsa* or *id_dsa*) has already been generated, the public

keyfile (*identity.pub, id_rsa.pub* or *id_dsa.pub*) just needs to be copied to a file called *authorized_keys* in the same directory. If there's no keyfile, run the following command:

```
ssh-keygen -t rsa1
```

This will generate a keyfile for SSH protocol version 1. After copying *identity.pub* to *authorized_keys*, a remote login without prompting for a password should be possible. At the first login, the user will be asked if the key should be accepted. This appears only at the first login as mentioned. If for any reason problems occur, the verbose/debug mode of SSH may help (example):

```
ssh -v client1
OpenSSH_2.9.9p2, SSH protocols 1.5/2.0, OpenSSL 0x0090602f
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Seeding random number generator
debug1: Rhosts Authentication disabled, originating port will not be
trusted.
debug1: restore_uid
debug1: ssh_connect: getuid 0 geteuid 0 anon 1
debug1: Connecting to client1 [10.1.1.1] port 22.
debug1: temporarily_use_uid: 0/0 (e=0)
debug1: restore_uid
debug1: temporarily_use_uid: 0/0 (e=0)
debug1: restore_uid
debug1: Connection established.
debug1: read PEM private key done: type DSA
debug1: read PEM private key done: type RSA
debug1: identity file /root/.ssh/identity type 0
debug1: identity file /root/.ssh/id_rsa type -1
debug1: identity file /root/.ssh/id_dsa type -1
debug1: Remote protocol version 1.99, remote software version
OpenSSH_2.9p1
debug1: match: OpenSSH_2.9p1 pat ^OpenSSH
debug1: Local version string SSH-1.5-OpenSSH_2.9.9p2
debug1: Waiting for server public key.
debug1: Received server public key (768 bits) and host key (1024 bits).
debug1: Host 'client1' is known and matches the RSA1 host key.
debug1: Found key in /root/.ssh/known_hosts:1
debug1: Encryption type: 3des
debug1: Sent encrypted session key.
debug1: Installing crc compensation attack detector.
debug1: Received encrypted confirmation.
debug1: Trying RSA authentication with key '/root/.ssh/identity'
```

```
debug1: Received RSA challenge from server.
debug1: Sending response to host key RSA challenge.
debug1: Remote: RSA authentication accepted.
debug1: RSA authentication accepted by server.
debug1: Requesting pty.
debug1: Requesting shell.
debug1: Entering interactive session.
Last login: Tue Oct 28 13:17:41 2003 from server
Have a lot of fun...
client1:~
```

With this detailed information occuring errors can be debugged.

2.3 MPICH

After downloading and unpacking the mpich archive, the software needs to be configured and compiled. Assuming that the package should be installed in */opt/mpi* and the NAG Fortran compiler is used, the following commands (executed in the archive directory) will build and install MPICH:

```
./configure --prefix=/opt/mpi --with-f95nag -rsh=ssh

make

make install
```

MPI can be tested after installing by using the examples which come along with the package. For more details see the installation manual of MPICH.

3. Telemac

3.1 The file `systel.ini`

The compiler–/ and MPI–configuration of Telemac is done in the file `systel.ini`. Compiler– and linkerflags need to be set in this file as well as the path of the MPI–commands. Here's an example of a configuration for Telemac working with the NAG compiler and MPI located in `/opt/mpi`:

```
[nag]
DIRLIB=nag
# Options du compilateur NAG:
FC_NAM="f95"
FC_OPT_OBJEXT="o"
#
FC_OPT_COMPIL=" -O4 -w -v -c "
FC_OPT_DEBUG=" -w=obs -v -g -gline -g90 -c "
FC_OPT_PROFILE=" -O4 -v -pg -c "
FC_OPT_INCLUDE="-I "
FC_OPT_OTHERS=
#
LK_NAM="f95"
LK_OPT_NORMAL=
LK_OPT_OUTNAME=" -o "
LK_OPT_DEBUG=" -g90 "
LK_OPT_PROFILE=" -pg "
LK_OPT_OTHERS=" -O4 -Bstatic "
#
LIB_NAM=ar
LIB_OPT_LIBEXT="a"
LIB_OPT_OUTNAME="cru"
LIB_OPT_OTHERS=
LIB_RANLIB="ranlib"
#
RUN_DEBUG="dbx90 "
RUN_PROFILE=
#
FC_MPI="/opt/mpi/bin/mpif90 "
LK_MPI="/opt/mpi/bin/mpif90 -O4 -Bstatic -o <EXE>  <OBJS>  <LIBS>"
LIBS_MPI=
RUN_MPI="/opt/mpi/bin/mpirun  -nolocal  -machinefile  mpirun.txt  -np  <N>
<EXE>"
#END
```

3.2 Running Telemac in parallel

In order to run a parallel computation, the keyword `PARALLEL PROCESSORS=[number of processors]` has to be defined in the case–file.
Another configuration file is `mpi_telemac.conf`. It must be located in the work–directory from where the job is started. The machines, on which the job can run, are defined in this file. It can look like this:

```
6
client1 1
client2 1
client3 1
client4 1
client5 1
client6 1
```

The first line is the total number of processors. All other lines show the name of a machine in the cluster with it's number of processors.

Telemac can now be started an (hopefully) will run in parallel mode.