Scientific Tools and Techniques - Exercises

Practical details

login to PLX:
ssh -l username login.plx.cineca.it

Username and Password: Available from demonstrator

Interactive PBS session:

```
qsub -l select=1:ncpus=4:mpiprocs=4,walltime=20:00 -q private -A
train_cnov2014 -W group_list=train_cnov2014 -I
```

Example PBS job script:

```
#PBS -l walltime=00:30:00
#PBS -l select=1:ncpus=4:mpiprocs=4
#PBS -N myjob
#PBS -o job.out
#PBS -e job.err
#PBS -q private
#PBS -W group_list=train_cnov2014
#PBS -A train_cnov2014

cd $PBS_O_WORKDIR
module load autoload openmpi
mpirun ./myexecutable
```

Suggested Exercises

1. Performance Analysis of a program (e.g. DL_POLY2) with Scalasca
2. Trace profiling with
    a. extrae/paraver
    b. mpirun -trace
3. Debugging a program with Totalview.

Exercise 1. Performance analysis of a program with SCALASCA

Step 1. Copy or generate the source

For example, if we are using the DL_POLY example, copy it from the build directory:

```
cp -r
/cineca/prod/build/applications/dl_poly/1.9/openmpi--1.3.3--intel--11
.1--binary/BA_WORK/dl_class_1.9 .
```

Step 2. RE-compile with the scalasca compiler wrapper.

```
cp build/MakePAR source/Makefile
```

Edit makefile:

```
$(MAKE) FC="skin mpif90" LD="skin mpif90 -o" \

module load autoload openmpi/1.4.4--gnu--4.5.2
module load scalasca/1.4.1_openmpi--1.4.4--gnu--4.5.2
cd source
make
```

Step 3. Run program

```
cd $PBS_O_WORKDIR
# copy input files for DL_POLY
cp
/gpfs/scratch/userinternal/aemerson/corsi/tools-and-techniques/ex2b/*
.

module load profile/advanced
module load autoload openmpi/1.4.4--gnu--4.5.2
module load scalasca/1.4.1_openmpi--1.4.4--gnu--4.5.2

exe=$HOME/dl_class_1.9/execute/DLPOLY.X

scalasca -analyze mpirun -np 4 $exe
```

Step 4. Analyse the results directory

scalasca -examine epik_DLPOLY_4_sum

Exercise 2a. Trace profiling with extrae and paraver

Step 1. Set up a batch job with the LD_PRELOAD command.

You can use, for example, the DL_POLY molecular dynamics program:

```
module load profile/advanced
module load extrae

TRACE="tracef.sh"

# Assuming FORTRAN source
cat<<EOF>$TRACE
#!/bin/sh

export EXTRAE_CONFIG_FILE=extrae.xml
export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitracef.so

## Run the desired program
\$*
EOF
chmod u+x $TRACE

module load autoload dl_poly/1.9
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/cineca/prod/compilers/openmpi/1.6.3
/gnu--4.7.2/lib
exe="DLPOLY.X"
mpirun -np 4 $TRACE $exe
```

xxxx

Step 2. Prepare the extrae profiling

```
cp $EXTRAE_HOME/share/example/MPI/extrae.xml .
```

Step 3. Run the job

```
qsub job.pbs
```

Step 4. Prepare the trace profile for paraver

```
module load extrae
mpi2prv -f TRACE.mpits -o output.prv
```

Step 5. Run paraver and load the profile

```
module load paraver
wxparaver output.prv
```

Exercise 2b. Trace profiling with IntelMPI and ITAC

Step 1. Re-compile program with IntelMPI

The program to be analyzed needs to be compiled with InteMPI so if using the DL_POLY2 example of Exercise 2a you must copy the source code and re-compile it.
(the module version is with GNU and OpenMPI).

```
cp -r
/cineca/prod/build/applications/dl_poly/1.9/openmpi--1.3.3--intel--11
.1--binary/BA_WORK/dl_class_1.9 .
cd  dl_class_1.9/source
module load  autoload intelmpi
make intel
```

Step 2. Copy the input files and prepare a batch job

Run the newly compiled program with PBS, after copying the input files.

```
cp
/gpfs/scratch/userinternal/aemerson/corsi/tools-and-techniques/ex2b/*
.

#PBS -l select=1:ncpus=4:mpiprocs=4
#PBS -l walltime=0:30:00
```

```
#PBS -q private
#PBS -A train_cnov2014
#PBS -N jobname
#PBS -W group_list=train_cnov2014

cd $PBS_O_WORKDIR
exe="dl_class_1.9/execute/DLPOLY.X"
module load autoload intelmpi
source
/cineca/prod/compilers/intel/cs-xe-2013/binary/itac/8.1.0.024/intel64
/bin/itacvars.sh
mpirun -trace $exe
---------
qsub job.pbs
```

Step 4. Run batch job and analyze trace file

After the run, analyze the trace with the traceanalyzer GUI.

```
source
/cineca/prod/compilers/intel/cs-xe-2013/binary/itac/8.1.0.024/intel64
/bin/itacvars.sh
traceanalyzer  DLPOLY.X.stf
```

Exercise 3. Debugging a program with totalview

We recommend you download and install the RCM client for this exercise:

http://www.hpc.cineca.it/content/remote-visualization-rcm

Step 1. Launch RCM from the client and open a terminal session.

Step 2. Create a directory and copy the program files:

```
cp -r
/gpfs/scratch/userinternal/aemerson/corsi/tools-and-techniques/poisso
n_training .
```

Step 3. Compile with a suitable compiler. Make sure you have specified the -g flag in the Makefile

```
module load autoload openmpi
make
```

Step 4. Launch a PBS job (see above) with these command lines:
(check the DISPLAY variable before launching qsub)

```
export DISPLAY=node97:8
cd $PBS_O_WORKDIR
module load totalview
mpirun -n 4 -tv ./poisson.exe
```

Try and find the program line causing the deadlock.