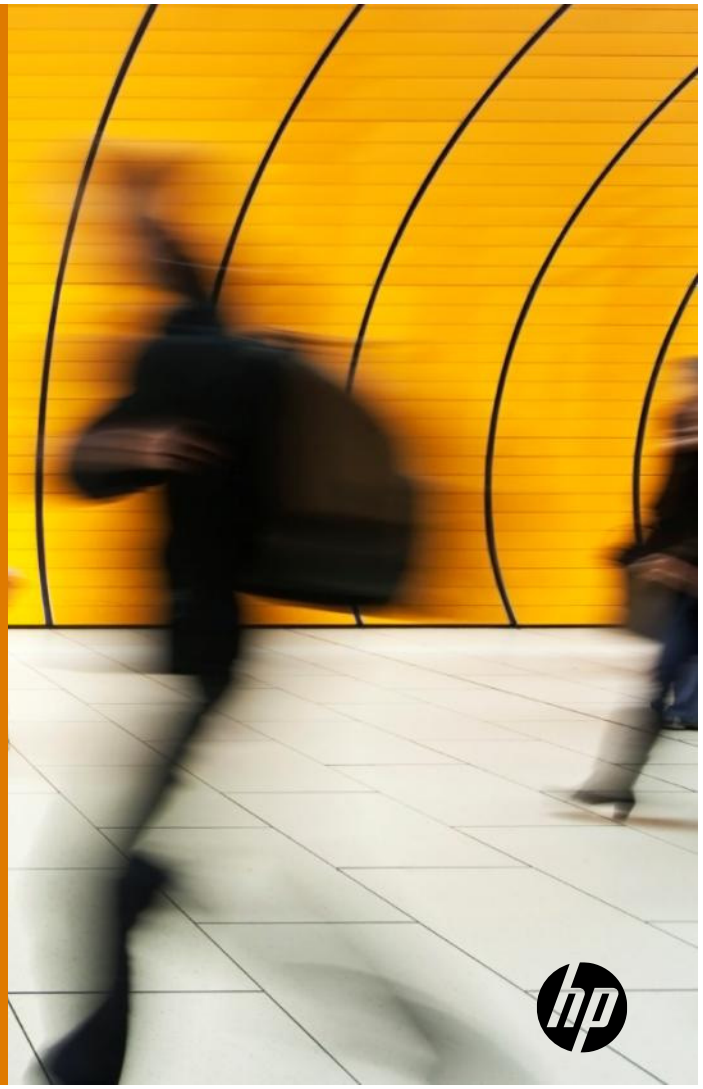
LINPACK

Cheran Sorin Cristian
EMEA HPC Competency Center



RUNNING LINPACK

The LINPACK Benchmark was introduced by Jack Dongarra.

the LINPACK Benchmark is used to solve a dense system of linear equations.

benchmark that allows the user to scale the size of the problem and to optimize the software in order to achieve the best performance for a given machine.

The result is given in number of FLOP per second.

Rpeak is the theoretical valued calculated. This shows the maximum number of FLOPS that the system might achieve.

Rmax is the number of FLOPS returned by the LINPACK benchmark.

CPUs : Theoretical performance- SERVER

Rounded values

server	chip	Number of cores	Flops/cycle	peak GFlops/server	Linpack GFlops/server	% eff.
any 2S	X5670 (2.93GHz)	12	4 per core	140	125	89%
4S	X7650(3.0GHz)	24	4 per core	288	260	89%
8S	X7650(3.0GHz)	48	4 per core	576	460	88%
any 2S	O6174(2.2GHz)	24	4 per core	211	180	85%
any 4S	O6174(2.2GHz)	48	4 per core	422	360	84%

CPUs : Performance- CLUSTER Rounded values

Calculating the Rpeak for a cluster

Cluster of 12 nodes with BL280 with processors of 2.4 GHz each

Rmax = 12 (servers) x 12 (cores) x 4 (FLOPs) x 2.4 GHz = 1382.4 FLOPs.

Expected Rpeak = DEPENDS ON THE INTERCONNECT.

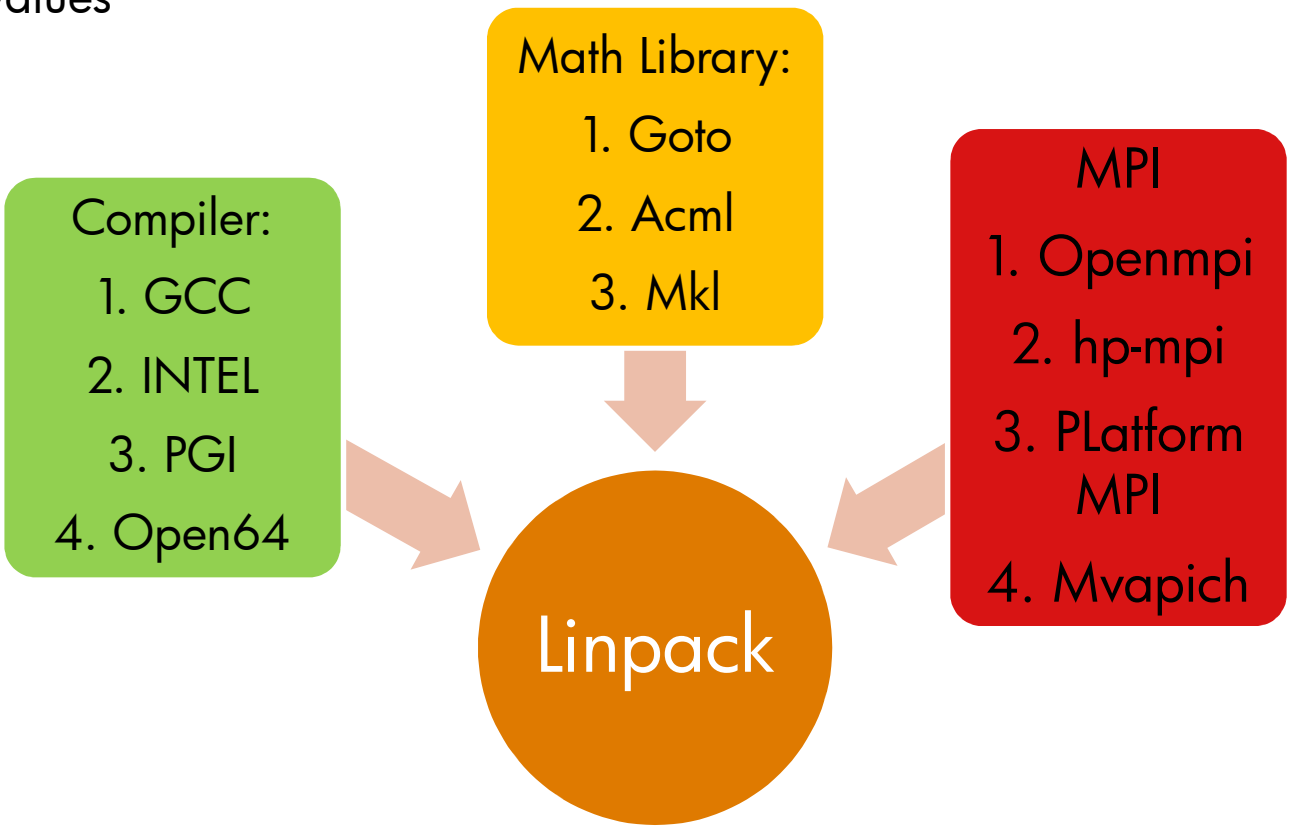
If INTERCONNECT= 1GB => 40-50% Rmax

If INTERCONNECT= 10GB => 60-65% Rmax

If INTERCONNECT= IB => 75-90% Rmax (depends on the network architecture)

COMPILING LINPACK – What do we need?

Rounded values



Linpack – Output Example

Rounded values

Server: BL280 G6 – 8 cores at 3.0

Rpeak: 8 (cores) x 4 (Flops) x 3.0 = 96 GFLOPs

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR01C2C8       28800   200     2     4             193.47              8.232e+01
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0034771 ...... PASSED
================================================================================
```

Rmax = 82.32 GFLOPs

Efficiency = 85.75%.

Linpack – Simple lets Run ☺

Normally one downloads hpl from the web

*www.netlib.org/benchmark/**hpl**/software.html*

This time around lets take it from /tmp/test/hpl.tgz

cp /tmp/test/hpl.tgz .
tar –zxvf hpl.tgz
cd hpl

Here we have written a script that compiles the linpack using different platforms

Compiling LINPACK

in hpl directory run

. [trng21@tunturi hpl]$ ./prepbin.csh
if (0 < 3) then
usage: ./prepbin.csh <gnu|open64|pgi|intel> <goto|gotop|acml|acmlp|mkl|mklp>
<impi|hpmpi|pmpi|qmpi|parampi|mpich|mpich2|mvapich|mvapich2|openmpi> [amd|intel]
exit 1
[trng21@tunturi hpl]$

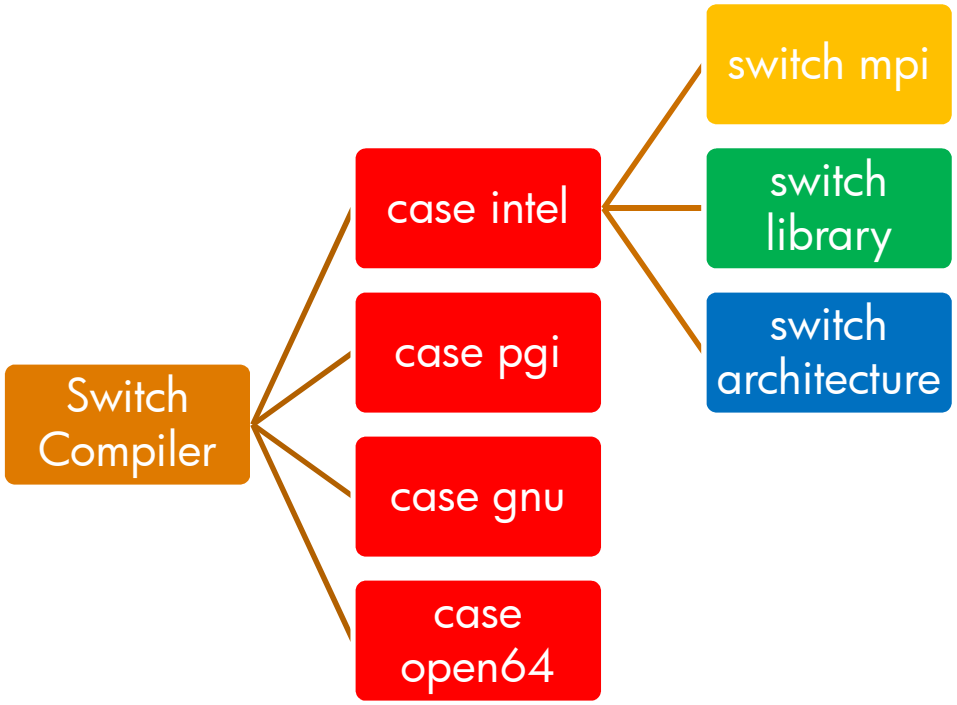
SO YOU WOULD HAVE TO CHOSE

COMPILER – MATH LIBRARY – MPI - ARCHITECTURE

prepbin.csh

It is a series of "cases"

setenv c $1 <- compiler
setenv l $2 <- library
setenv m $3 <- mpi
setenv a ""$4 <- architecture



Running LINPACK

We use the script run.csh which has as a parameter just the number of cores on which I want to run.

The Linpack is called at the end of the file:

(/usr/bin/time srun -n $np **-N 1** $o/bin/mine/xhpl)|& tee myoutput.$np

ATTENTION: we also need to set in the SCRIPT the –N 1

Use –nodelist=n[30X]

Where X is the team number

Output files

Everything will be saved in a directory of the form :

work.${np}P.${nt}T.`hostname`.`date +"%m%d%H%M"`.$$

Example:

work.2P.1T.tunturi.csc.fi.11290228.26479

Interesting file to look at is myoutput.${np}

Lets Run ☺

cd /home/<YOUR_USER_NAME>/hpl

./run.csh 1 &
./run.csh 8 &

It might fail ☺

Go back to run.csh and add on the srun line the following:

--mem-per-cpu=3072

(/usr/bin/time srun -n $np -N 1 –nodelist=n[30**X**] *- -mem-per-cpu=3072*
$o/bin/mine/xhpl)|& tee myoutput.$np