SCore MPI

Taking full advantage of GigE

NW-GRID Training Event
26th January 2007

Cliff Addison
University of Liverpool

Overview

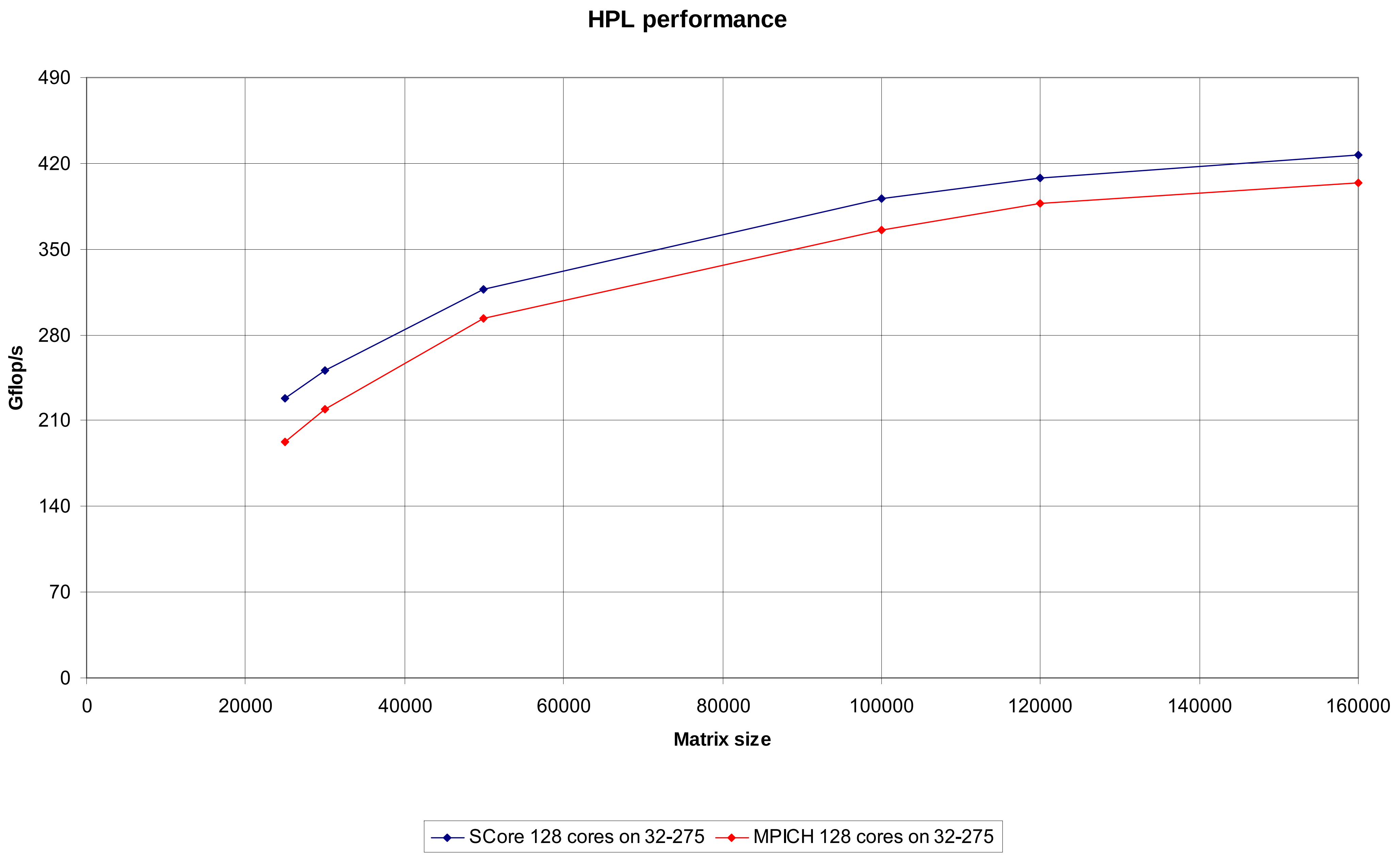
- What is and why use SCore?
- Compiling SCore MPI jobs.
- Submitting parallel jobs via mpisub.
- Writing your own qsub script.
- Themes and variations.

What is SCore?

- Advanced cluster O/S and run-time environment
  - Cluster installation & management tools
  - Integrated communication layer
  - Includes an advanced implementation of the MPI standard
    - presently based on mpich
- Commercially developed/supported by Streamline
- Original developer works for Streamline in Japan

Why SCore?

- Performance
  - Employs a TCP/IP bypass on Ethernet
    - Inter-node latencies typically around 15 μsec (vs. 25-30 μsec with regular mpich)
    - Inter-node bandwidth around 110 MB/s (vs. ~ 100 MB/s with mpich)
  - Network fail over mode allows multiple NICs to be employed.
    - latency circa 20 μsec, b'width ~ 200 MB/s
  - Intra-node performance excellent.


HPL performance
Gflop/s
490
420
350
280
210
140
70
0
0
20000
40000
60000
80000
100000
120000
140000
160000
Matrix size
SCore 128 cores on 32-275
MPICH 128 cores on 32-275

Why SCore?

- Expanded feature set (active development in Japan via PC Cluster Consortium)
  - Good checkpoint / restart mechanisms (soon to extend to multi-threaded and dynamically linked applications).
  - Robust job failure (relatively few problems with rogue processes).
  - Close integration with Sun Grid Engine.

Compiling SCore MPI jobs

- Suitable only for 64-bit binaries!!
- Default environment should put **`/opt/score/bin`** near the top of your PATH list.
- Use of compiler wrappers essential:
  - MPI library links / includes handled automatically.
  - mpif90 for PGI pgf95
  - mpicc, mpif77 and mpic++ use the GNU compilers
  - **`mpicc -compiler pgi`** to access PGI C compiler
  - **`mpicc -help`** shows command to get more information.

More on compiling

- Compiler options, library references are passed through to the associated compiler / loader

```
mpicc -compiler pgi -fastsse -tp k8-64
```

- Go faster options for PGI cc and f90:

```
-fastsse -tp k8-64
```

  - Note flag `-tp k8-64` must be included on the link line too.

- 64-bit binaries can be tricky to get right
  - With VASP (f90 code) needed to add `–Mlfs`

Using mpisub

- The mpisub script generator is the easy way to submit SCore-MPI jobs to the system.
  - Underlies the MPI job type via the sge jobmanager with globus-job-submit / globus-job-run.
  - Basic syntax easy:

    ```
    mpisub NxS exec arg1 arg2
    ```

    - N – number of nodes, S – 1, 2 or 4 cores / node

      ```
      mpisub 5x4 pi2 "< infile"
      ```

Using mpisub-2

- mpisub:
  - Generates a job submission script,
  - Submits the job to an appropriate parallel environment – is non-blocking!
  - Performs a qstat command to give you an idea of where job your lies in the queue.
  - Options further controlled using the values of
    - $QSUB_OPTIONS
    - $DEFAULTQ
    - $SGE_RESOURCES

A minimal qsub script

```
#!/bin/sh
#$ -cwd -V -j y
#$ -pe score 3
EXEC="pi_redirect arg1”
PROCS=$((NSLOTS-1))x4
scout -wait -F $HOME/.score/ndfile.$JOB_ID \
-e /tmp/scrun.$JOB_ID \
-nodes=$PROCS,  $EXEC
```

Grid Engine Options

Parallel environment is score

Executable & argument

NSLOTS – predefined = 3 here so PROCS is 8 on 2 nodes

JOB_ID – SGE job id

qsub scripts – General points

- Need to specify number of slots = nodes+1 in the parallel environment line.
- A script can be parameterised fully if move the –pe line out as a qsub argument.
- EXEC and PROC variables make the script more generic, but are not essential.
  - EXEC string is usually just the executable name
- $NSLOTS is predefined by SGE to the number of slots.
- $JOB_ID is predefined to the SGE job id.
- scout line can be copied once and left alone if use $EXEC and $PROC.

qsub scripts - Further details

- Sample qsub command for script without the –pe line:

```
qsub -pe score 7 qsub_NeedsPE_pi.sh
```

- Typical run generates 4 files (SGE output & error plus job output & error).
  - `#$ -j y` – combines error and output files
  - `#$ -o outfile` - redirect standard output
    - Similar for standard error
    - Combine above two commands - all output in `outfile`.

qsub scripts - More further details

- Sometimes need special environment variables passed to the compute nodes.
  - **`#$ -V`** → passes environment to starting process
    - scout them propagates this environment to compute nodes.
- Default location for output etc. – home directory
  - **`#$ -cwd`** → changes this to current working directory.

qsub scripts – Final points

- Example programs, Makefile, README can be found in /usr/local/examples (at least on lv1).
- mpirun can be used instead of scout to launch parallel jobs.
  - Standard arguments
    - Machinefile: `$HOME/.score/ndfile.$JOB_ID`
  - May not be as robust as scout on failed jobs!

Grid implications

- qsub commands are non-blocking operations
  - Need to capture something like Job ID and then check to see when that job is no longer on the system (e.g. build on lines like)

```
Job_ID=`qsub myscript | cut -f 3 -d " "`
qstat -u $USER | grep $Job_ID
```

  - Can use gsiscp to stage / retrieve files
  - Can use gsissh to run commands / scripts
  - Make certain remote submission needs are not satisfied using mpisub (much easier to work with!)