SP2

Using the IBM SP2 at OSC

The Ohio Supercomputer Center (OSC) provides supercomputing services to Ohio colleges, universities, and companies.

One of the high performance computing systems available at OSC is the IBM SP2. IBM SP2. The current SP2 configuration consists of eight wide nodes each possessing a POWER2 processor with a 256 KB data cache and a 32 KB instruction cache. 512 MB of main memory is available for nodes 01 through 09 (see numbering scheme below) and 256 MB for nodes 11 through 15. Each node possesses 8 GB of disk space.

An exceptional feature of the SP2 is its High Performance Switch (HPS). This interconnection network between nodes provides a low latency, high bandwidth communication medium for parallel applications.

To access the SP2, **`telnet`** to **`oscsp.osc.edu`** .This is node 01 (**`oscsp01`**) by default. The nodes on the SP2 are labeled by the odd numbers from **`oscsp01`** to **`oscsp15`**. All interactive work should be done from node 01. Note that each user's home file area on the SP2 (`/home`) is mounted from the Cray Y-MP2E. The nodes form a group known as pool 1. Type **`jm_status -P`** for a listing.

Compiling on the SP2

Compilers for Fortran, C and C++ are available on the IBM SP2. Your choice of compiler will depend on whether you choose MPL, MPI, PVMe or HPF as your programming style (see **Programming Styles**).

MPL (Message Passing Library)

```
mpcc prog.c
```

```
mpxlf prog.f
```

Options accepted by **`xlf`** or **`cc`** may be used here as well. The C++ compiler is **`mpCC`**.

MPI

LAM (Local Area Multicomputer) is available for developing parallel programs using MPI.

PVMe

```
cc (xlf) prog.c (prog.f) -L/usr/lpp/ssp/css/comm_lib -lpvm3 (-lfpvm3) -bI:/usr/lpp/p
```

where *comm_lib* is the communication library used and is either `libus` or `libip`. Fortran programs must link the `pvm3` **and** `fpvm3` libraries. The compiler **`f77`** is identical to `xlf`.

HPF (High Performance Fortran)

```
xlfhpf -qhpf -O prog.f -us
```

The communication library is specified by the `-us` or `-ip` option.

Executing Programs

MPL

Execution is via POE (see **Programming Styles**).

```
poe a.out
```

PVMe

1.) Start the PVMe daemon.

```
pvmd3e 4 -share cpu &
```

allocates 4 processors for example.

2.) Run the program on one of the nodes allocated in step 1. If your local host is not one of the allocated nodes, you must use `rsh` *`hostname`* `a.out`.

3.) **Halt the PVMe daemon!**

Upon finishing, type `pvm`. At the `pvm>` prompt type `halt`.

HPF

Execution is via POE. For example, using four processors:

```
poe a.out -procs 4
```

Batch Requests

IBM's **LoadLeveler** (**LL**) is the batch queuing system on the SP2. Batch jobs are submitted using

```
llsubmit job.cmd
```

where `job.cmd` is a LL command file. Here is a sample command file for an MPL job using the US communication library and four processors. User files are located in `/home/work`.

```
# This is a comment line
#@ min_processors=4
```

```
#@ max_processors=4
#@ initialdir=/home/work
#@ output=my.data
#@ error=my.err
#@ job_type=parallel
#@ requirements=Adapter==``hps_user'')
#@ class=POE
#@ environment=MP_EUILIB=us
#@ queue
/usr/bin/poe a.out
```

For a PVMe batch job, set `class=PVMe` and change the last line above to

```
/usr/lpp/pvm3/pvmd3e -exec a.out
```

`MP_EUILIB` is not relevant to PVMe.

Monitoring jobs

To monitor batch requests use `llq`. To cancel batch jobs use `llcancel` *Id* where *Id* is from `llq`'s output. Use `xloadl` to monitor and submit jobs using the X Window interface. Use `jm_status -j` for general info.

Programming Styles

The primary programming model for the SP2 is message passing. A shared memory model may be implemented using HPF.

Message Passing

Parallel processing is accomplished by making calls to one of three sets of libraries: MPL, PVMe or MPI (via LAM).These libraries allow your code to run as a number of parallel tasks on the SP2 nodes.

MPL

(Message Passing Library) You must set up your execution environment before running your MPL program. This environment is called **POE** (Parallel Operating Environment). At a minimum you **must** set the following five environment variables

```
MP_PROCS=number of tasks
MP_RMPOOL=1
MP_EUILIB=ip or us
MP_EUIDEVICE=en0 or css0
MP_RESD=yes
```

Using `MP_EUILIB=us` provides the best performance but only one job per node may run in US at any given time. Type `man poe` for details.

PVMe

PVMe is IBM's PVM **enhanced** for the HPS. To port existing PVM code to the SP2, relink with the PVMe libraries. Specify include files and working directory for PVMe:

Fortran: `include 'fpvm3.h'`

C: `#include ''pvm3.h''`

export PVMEPATH=/home/work

Debugging/Performance Analysis

The tools **`dbx`**, **`pdbx`** and **`xpdbx`** are available for debugging programs. The Visualization Tool (**`vt`**) provides visualization and performance monitoring. Consult the man pages for details.

Documentation

OSC Tech | Search | Consult | Computing

http://oscinfo.osc.edu/computing/sp2/