Running multiple instances of a script on a cluster using OpenMPI

Use cases

This method is suitable for embarrassingly parallel processing tasks: i.e. tasks which can split up to be processed in parallel with no communication between the processes.

MPI

MPI stands for “Message Passing Interface”. It a standard widely used on clusters for allowing multiple instances of a program to run across on or more machines. As its name suggests these different instances are able to communicate with each other. It essentially works like this.

1. You install a version of MPI on your cluster. CUPPA uses OpenMPI from the ubuntu repositories.
2. You write your fortran or c program including special MPI calls. An important one in most cases is `MPI_Comm_rank()` which returns an ordinal integer number to each instance
3. You can now both
    - Instruct your program to do different things depending on its rank
    - Pass data from one instance to another
4. You compile your code using a special MPI compiler
5. You run your code using another MPI program (`mpirun`). You will have to specify the number of instances (processes) and that hostnames of the machines on which you wish your command to run

Using OpenMPI to run normal programs

OpenMPI will also run normal, non-mpi programs. For example you could create the following text file called `machines`

```
cuppa01
cuppa02
cuppa03
```

then type

```
$ mpirun -np 3 -machinefile machines ls /exports/
```

The line above instantiates 3 instances of `ls /exports/` on the computers specified in `machines`. You should get back:

```
apps  temp_mnt  xraid03
apps  xraid02
apps  rsyncd.conf  xraid01
```

This is because the stdout and stderr from all 3 processes are copied to the terminal you are using. Note the order that these lines return in is not guaranteed. Note that you can change the number of processes to 6 and you'll have two processes run on each machine. You can also leave out the -machinefile switch and have all processes run on the local machine.

Using OpenMPI to parallelise normal programs

Let's investigate the environment in which our program is running. Type

```
$ mpirun -np 3 -machinefile machines env
```

The env command lists all environment variables in the pseudoshell where the mpi program runs. If you look carefully you should see some variable which has the values 0, 1 and 2 on each machine respectively. Unfortunately the exact name of the variable changes depending on the exact version of OpenMPI. On cuppa it is `OMPI_MCA_ns_nds_vpid`

```
$ mpirun -np 6 -machinefile machines env | grep OMPI_MCA_ns_nds_vpid
OMPI_MCA_ns_nds_vpid=1
OMPI_MCA_ns_nds_vpid_start=0
OMPI_MCA_ns_nds_vpid=2
OMPI_MCA_ns_nds_vpid_start=0
OMPI_MCA_ns_nds_vpid=0
OMPI_MCA_ns_nds_vpid_start=0
```

It is trivial to use this environmental variables in any scripting language. You can also access environmental variables in c or fortran. Alternatively you can create a wrapper script for compiled programs and pass the relevant parameters on the command line. Below are a few examples

bash

```
#!/usr/bin/env bash
rank=$OMPI_MCA_ns_nds_vpid
if ((rank == 0))
then
    echo I am zero
else
    if ((rank < 5))
    then
        echo I am process $OMPI_MCA_ns_nds_vpid
    fi
fi
```

perl

```
#!/usr/bin/perl
use Env;
Env::import();
$rank=$OMPI_MCA_ns_nds_vpid;
if ($rank == 0)
{
    print "I am zero\n";
}
elsif ($rank < 5)
{
    print "I am $rank\n";
}
else
{
    die "Nothing to do\n";
}
```

python

```
#!/usr/bin/env python
import os
import sys

env_rank_name = "OMPI_MCA_ns_nds_vpid"
try:
    rank = int(os.environ[env_rank_name])
except:
    raise RuntimeError, "Error, can't read environment variable %s" %
env_rank_name

if rank is 0:
    print "I am zero"
elif 1 <= rank < 5:
    print "I am %d" % rank
else:
    print "Nothing to do"
    sys.exit(0)
```

From:
http://www.atnf.csiro.au/vlbi/dokuwiki/ - **ATNF VLBI Wiki**

Permanent link:
http://www.atnf.csiro.au/vlbi/dokuwiki/doku.php/correlator/mpiscript

Last update: **2011/12/09 17:14**