Multiple Program Multiple Data Runs

Timothy H. Kaiser, Ph.D.
tkaiser@mines.edu

MPMD/MIMD Runs?

- In theory, MPI jobs can have different executable program for each task: Multiple Program, Multiple Data or Multiple Instruction, Multiple Data
- Why?
  - Might have different processes doing different things, for example a GUI front end
  - Can combine different languages
- Not all versions of MPI support this (MVAPICH 1)
- Real trick is telling system how to launch the job

Method for OpenMPI and MVAPICH2

- Same method works for both versions of MPI
- Create a description of your job on the fly from inside your script
- The description is a mapping of programs to cores
- Tell mpiexec/mpirun to use your description to launch your job
- We created a utility script to make it easy

Using appfiles

- Most versatile way to run a MPMD program is to use an "appfile"
- A list of nodes on which to run along with the program that is to be run on each node
- Using an appfile
  - OpenMPI
    - mpiexec --app appfile
  - MVAPICH
    - mpiexec --configfile appfile

Appfile format

- Collection of lines of the form
  - -host <host name> -np <number of copies to run on host> <program name>
- Specify different application names in your appfile for MPMD
- You can specify a node or program more than once

Examples

These two are equivalent

Appfile Example 1

```
-host compute-1-1 -np 1 myprogram
-host compute-1-1 -np 1 myprogram
-host compute-1-1 -np 1 myprogram
-host compute-1-1 -np 1 myprogram
```

Appfile Example 2

```
-host compute-1-1 -np 4 myprogram
```

Note: You should specify the full path to your program

These two are not equivalent

Appfile Example 1

```
-host compute-1-1 -np 2 aya.out
-host compute-2-3 -np 2 bee.out
```

Appfile Example 2

```
-host compute-1-1 -np 1 aya.out
-host compute-2-3 -np 1 aya.out
-host compute-1-1 -np 2 bee.out
-host compute-2-3 -np 2 bee.out
```

Difficulty and Solution

- Problem:
  - Names of the nodes that you are using are not known until after the job is submitted
  - You need to create the appfile on the fly from within your PBS script

Difficulty and Solution

- Solution:
  - The PBS variable $PBS_NODEFILE contains the name of a file that has the list of nodes on which your job will run.
  - We have created a script "match" which takes a list of nodes and a list of applications to run on those nodes and creates an appfile
- Located on Ra at /lustre/home/apps/utility/match
  - Path= /lustre/home/apps/utility:$Path
  - setenv PATH /lustre/home/apps/utility:$Path

Solution

Given your $PBS_NODEFILE and a list of programs in a file app_list the simplest usage of match is:

```
match $PBS_NODEFILE app_list > appfile
mpiexec --app appfile
```

For mvapich2 replace --app with --configfile

Match notes

- Number of applications that get launched
  - Is equal to the length of the longer of the two lists, the node file list, or the application list
  - If the lists are not the same length then multiple copies will be launched
    - Match also takes an optional replication count, the number copies of an application to run on a node
- Feel free to copy and modify the match script for your own needs

Examples

```
#get a copy of all of our nodes, each node will be
#listed 8 times
cat $PBS_NODEFILE > fulllist

#save a nicely sorted short list of nodes, each node only
#listed one time
sort -u $PBS_NODEFILE | sort -t- -n -k2 -k3 > shortlist
```

fulllist

```
compute-8-15.local
compute-8-15.local
compute-8-15.local
compute-8-15.local
compute-8-15.local
compute-8-15.local
compute-8-15.local
compute-8-15.local
compute-8-13.local
compute-8-13.local
compute-8-13.local
compute-8-13.local
compute-8-13.local
compute-8-13.local
compute-8-13.local
compute-8-13.local
compute-8-13.local
```

shortlist

```
compute-8-15.local
compute-8-13.local
```

```
/lustre/home/apps/utility/nsort $PBS_NODEFILE
```

Also works to give a sorted list

Examples

- We have two programs we are going to play with f_ex00 and c_ex00
- We have two program lists that we are going to use
    - oneprogram
        - c_ex00
    - twoprograms
        - c_ex00
        - f_ex00

match fulllist twoprograms > appfile I

```
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
```

match fulllist twoprograms > appfile I

```
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-15.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-13.local  -np  1  f_ex00
```

match shortlist twoprograms > appfile2

```
-host  compute-8-13.local  -np  1  c_ex00
-host  compute-8-15.local  -np  1  f_ex00
```

match shortlist twoprograms 2 > appfile3

```
-host  compute-8-13.local  -np  2  c_ex00
-host  compute-8-15.local  -np  2  f_ex00
```

match shortlist oneprogram 2 > appfile4

```
-host  compute-8-13.local  -np  2  c_ex00
-host  compute-8-15.local  -np  2  c_ex00
```

This will be useful for OpenMP programs

Can take names from command line

match <node list file> -p"list of programs" [<number of copies per node>]

```
match shortlist -p"c_ex01 f_ex01" 1 8
```

Run 1 copy of c_ex01 on the first node in shortlist
and 8 copies of f_ex01 on the second node