Task Farming For Embarrassingly Parallel Processing

Ivan Girootto – igirotto@ictp.it

Information & Communication Technology Section (ICTS)
International Centre for Theoretical Physics (ICTP)
Multi-core system Vs Serial Programming

Xeon E5650 hex-core processors (12GB - RAM)
Multi-core system Vs // Programming

Xeon E5650 hex-core processors (12GB - RAM)
Ivan Girotto
igirotto@ictp.it

Task Farming For Embarrassingly Parallel Processing
I don’t know about // Programming

School on Parallel Programming and Parallel Architecture for HPC

ICTP Scientific Calendar 2016, stay posted!!!
... but I’m lucky!!

- I am working on an embarrassing parallel problem
- I can divide the work in independent tasks (no communication) that can be performed in parallel
- Quite common in Computer Graphics, Bioinformatics, Genomics, HEP, anything else requiring processing of large data-set, sampling, ensemble modeling
Single Program on Multiple Data

- performing the same program (set of instructions) among different data
- Same model adopted by the MPI library
- A parallel tool is needed to handle the different processes working in parallel
- The MPI library provides the `mpirun` application to execute parallel instances of the same program
\$ mpirun -np 12 my_program.x
[igirotto@mynode01 ~]$ mpirun -np 12 /bin/hostname
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02

**PATH name common to all processes!!**
Parallel Operations in Practice

• Parallel reading and computing in parallel is always allowed
• Parallel writing is extremely dangerous!
• To control the parallel flow each process should be unique and identifiable (ID)
• The OpenMPI implementation of the MPI library provides a series of environment variables defined for each MPI process
**OMPI_COMM_WORLD_SIZE** - the number of processes in this process' MPI Comm_World

**OMPI_COMM_WORLD_RANK** - the MPI rank of this process

**OMPI_COMM_WORLD_LOCAL_RANK** - the relative rank of this process on this node within its job. For example, if four processes in a job share a node, they will each be given a local rank ranging from 0 to 3.

**OMPI_UNIVERSE_SIZE** - the number of process slots allocated to this job. Note that this may be different than the number of processes in the job.

**OMPI_COMM_WORLD_LOCAL_SIZE** - the number of ranks from this job that are running on this node.

**OMPI_COMM_WORLD_NODE_RANK** - the relative rank of this process on this node looking across ALL jobs.

http://www.open-mpi.org
In Python

```python
import os
myid = os.environ['OMPI_COMM_WORLD_RANK']
[...]
```

In BASH

```bash
#!/bin/bash
myid=${OMPI_COMM_WORLD_RANK}
[...]
[igirotto@mynode01 ~]$ mpirun ./myprogram.[py/sh...]
```
Possible Applications

• Executing multiple instances on the same program with different inputs/initial cond.
• Reading large binary files by splitting the workload among processes
• Searching elements on large data-sets
• Other parallel execution of embarrassingly parallel problem (no communication among tasks)
Conclusions

• Task Farming is a simple model to parallelize simple problems that can be divided in independent task

• The *mpirun* application aids to easily perform multiple processes, includes environment setting

• Load balancing remains a main problem, but moving from serial to parallel processing can substantially speed-up time of simulation