Universiteit van Amsterdam
BSc Informatica
Dr Adam Belloum

HPC and BigData

2014/2015

Tutorial DAS4

Objectives: In this first tutorial students will learn some basic features of the DAS4 supercomputer. This tutorial will focus on the details needed in the hands-on of the workshop on MPI: learn to set up your working environment, the rules of using the DAS4 systems, run simple application example using DAS4 queuing systems, and run simple CUDA or OpenCL program on the DAS4 special node containing accelerators.

For more details about DAS4 consult the DAS4 web page[1]

1. DAS-4 User Accounts
    a. Password management:
        - Where student should change their password?
    b. DAS4 File System management:
        - What is the user default quota?
        - How to use Scratch to get extra space?
    c. System admin:
        - Who to contact in case of problem?
    d. Access to DAS4:
        - How to log DAS4 using ssh?
2. Usage policy
    a. What is the default run time for jobs?
    b. How and when to execute long running programs?
    c. Where to run application program (compute nodes vs. head node)*?*
    d. What are the consequences of not following the rules?

3. Job Execution
    a. How to use Grid Engine (SGE) batch queuing system? Follow the steps of SGE/MPI example
    b. How to use Prun user interface? Prun/MPI example
4. CUDA and OpenCL
    a. How to setup the environment for running CUDA /OpenCL programs?

[1] **Source**: http://www.cs.vu.nl/das4/home.shtml in the section Users

SURF SARA

UNIVERSITY OF AMSTERDAM

MPI hands-on

1. Learn about how to set the mpi environment
    - Read Guideline job submission on DAS4
    - Link http://www.cs.vu.nl/das4/jobs.shtml

2. Create, and compile an MPI "Hello world" program
    - Uses the appropriate MPI include file
    - Identifies task 0 as the "master" task
    - Initializes the MPI environment
    - Gets the total number of tasks
    - Gets the task's rank (who it is)
    - Gets the name of the processor it is executing on
    - Prints a hello message that includes its task rank and processor name
    - Has the master task alone print the total number of tasks
    - Terminates the MPI environment

    - **Hint:** Try first alone if you cannot use the helloworld.c available on the hpc.uva.nl/Downloads

3. Run MPI "Hello world" program

    3.1. Run the helloworld using the command mpirun command
    - you should be able to run the hellow world with only specify the number of processes "`-np`"
    - **Hint**: help can be find on http://www.open-mpi.org/faq/

    - **Question:** Explain what did happen? Where your jobs have been executed?

    3.2. Run the helloworld using the command mpirun command with more options
    - You can select the hosts on which you want to run your processes
    - **Hint**:
        - Read about the option of the mpirun "`--hostfile`" on http://www.open-mpi.org/faq/
        - If the program does not run, try to read about the mpirun "`–prefix`" on http://www.open-mpi.org/faq/

    - **Question:** Explain the difference with (3.1)? why do with need the prefix option?

    3.3. Run the helloworld using the using SGE
    - **Hint:**
        - Read about http://www.cs.vu.nl/das4/jobs.shtml

    - **Question:** Explain how SGE has helped you in running your MPI program?

    3.4. Run the helloworld using the prun
    - **Hint:**
        - Read about http://www.cs.vu.nl/das4/jobs.shtml

    - **Question:** what prun can do more than SGE?

4. Create, and compile an MPI "Hello world" program that exchange messages using blocking send/receive routines[2]

   - Assuming you were able to create a successful "hello world" MPI program in copy your source file to a new file and call it something like helloBsend.c
   - If you were not successful, you can use the provided helloworld.c.
   - Edit your new helloBsend source file and modify it to do the following - after the master task has printed the number of tasks, but before MPI_Finalize:

     Have each task determine a unique partner task to send/receive with. One easy way to do this:

     ```
     if (taskid < numtasks/2) then partner = numtasks/2 + taskid
     else if (taskid >= numtasks/2) then partner = taskid - numtasks/2
     ```

   - Each task sends its partner a single integer message: its tasked
   - Each task receives from its partner a single integer message: the partner's tasked
   - For confirmation, after the send/receive, each task prints something like:

     `"Task ## is partner with ##"`

     where ## is the taskid of the task and its partner.

[2] This exercise is part of the Message Passing Interface (MPI) tutorial given by Blaise Barney at the Lawrence Livermore National Laboratory
https://computing.llnl.gov/tutorials/mpi/exercise.html#Exercise2