

Homework

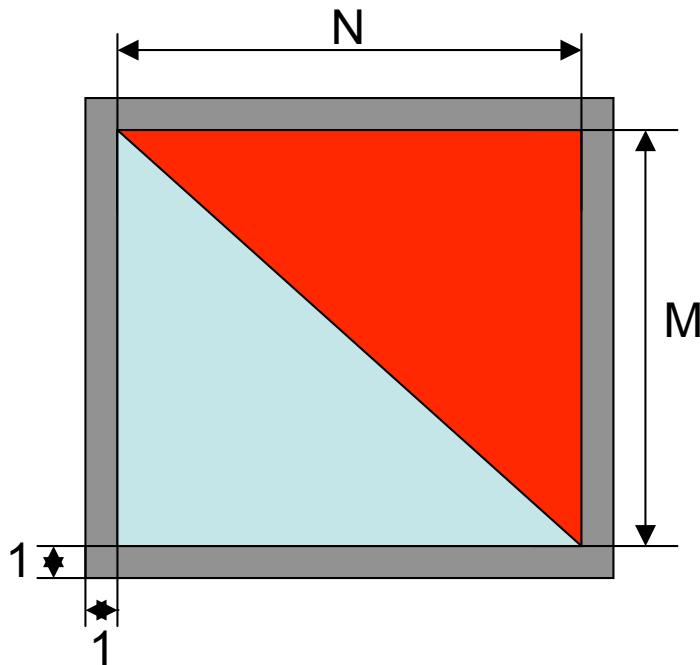
Due date: February 18th 2008

Homework

- How: by email at bosilca@eecs.utk.edu
 - A MD5 and a tar (gzipped) file.
 - Don't forget to comment your code.
 - The submission should compile and work on the cluster downstairs.
 - Provide a HOWTO and/or a README.
 - Add a pdf file with the discussion and results from the Part 2 and Part 3.

Homework – Part 1

- Definition
 - Ghost region: is a region of data around a matrix used in some mathematical algorithms to keep data from the neighbors.



1. Create a data-type describing the ghost region of a subset of the matrix starting at position (x, y) with a size (sn, sm) .
2. Create a data-type describing the upper-bound triangular matrix (the red part) excluding the diagonal
3. How can we transpose a matrix using data-types ?

Homework – Part 2

- Create a matrix of size N by M , and distribute it over a grid of processes P by Q .
- Using the data-type created in the first part of the homework, exchange the ghost regions between the neighbors.
- What is the most efficient way of doing the exchange ? Measure the performance of your different approaches, and report them in the document attached to your submission.

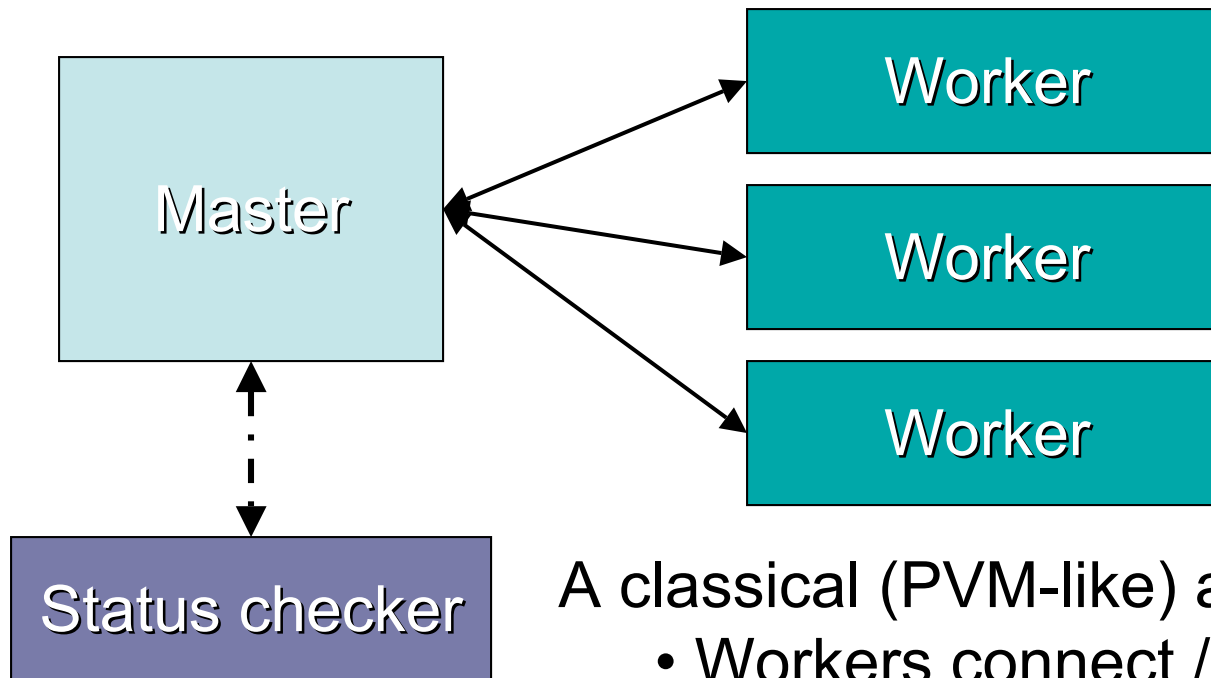
Homework – Part 3

- Create the master / slave framework described on the next slides, respecting the following constraints:
 - Each entity (master, slave, status checker) is a separate MPI application.
 - Do not use MPI_Spawn.
 - The work distributed by the master consist on a unique random number between 1 and 10, and represent the amount of seconds the worker have to sleep before requesting another piece of work.
 - The status checker print the total number of tasks completed as well as the number of tasks pending.
 - **Do it in the simplest possible way !!!!**

Advanced Master / Slave

- Use all of the MPI topics covered
 - Accept / connect
 - Multi-threaded application
- To create a generalized master / slave framework
 - Adapt the messages exchanged between processes to distribute work and return results
 - Fill in slave functions to do the work

Framework Architecture



A classical (PVM-like) approach

- Workers connect / disconnect at run time

New feature: master is multi-threaded to minimize the response latency

Framework Architecture

Master

Connect / Accept
thread

- Accept new connections
- Add the new workers to the work pool
- Handle the status report

Work distribution
thread

- Distribute the available work between the workers
- Manage the workload between clients

Results
Management
thread

- Asynchronously handle the result (e.g., save to file)
- Manage memory used for the result

Framework Architecture

