Problem set 1, MPI Intro

TDT4200, Fall 2015

Deadline: 03.09.2015 at 23:59. Contact course staff if you cannot meet the deadline.

Evaluation: Pass/Fail

Delivery: Use itslearning. Deliver exactly two files:

- *yourNTNUusername_ps1.pdf*, with answers to the theory questions
- *yourNTNUusername_code_ps1.{zip |tar.gz |tar}* containing your modified versions of the files:
  - `Makefile`
  - `computeMPI.c`

General notes: Code must compile and run on the following systems:

1. `its015-XX.idi.ntnu.no` (`XX` being any of the lab machines in ITS015).
2. Vilje Supercomputer: vilje.hpc.ntnu.no

You should only make changes to the files indicated. Do not add additional files or third party code/libraries.

Part 1, Theory

Problem 1, General Theory

a) Write a table explaining Flynn's Taxonomy.

   i) Explain how, where, and why MPI fits into Flynn's Taxonomy, and why if not.

b) Shared Memory

   i) Explain how and why (and why if not), MPI fits with Shared Memory systems.

c) Distributed Memory

   i) Explain how and why (and why if not), MPI fits with Distributed Memory systems.

Problem 2, Code Theory

Answer these questions after you've finished the code implementation from Part 2.

a) Does your implementation have any communicational bottlenecks?

   i) Briefly outline the communicational bottleneck(s), if any.

   ii) Is it possible to reduce the bottleneck, and if so how?

      1) If so, outline an algorithm that will reduce the bottleneck.

b) Does your implementation have any computational bottlenecks?

   i) Briefly outline the computational bottleneck(s), if any.

   ii) Is it possible to reduce the bottleneck(s)?

      1) If so, outline an algorithm/method that will reduce the bottleneck.
      A one line formula, if found, is acceptable (this is a hard question).

Part 2, Code

The `Makefile` handed out with this problem set compiles the handed out `computeSerial.c` and creates an executable named "`serial`". `serial` implements a naive approach to sum $1/log(num)$ within the range given as input to the program. Valid input values for $start$ and $stop$ are any natural numbers $\mathbb{N}$ greater than 2, and $stop > start$. Output is only "%f\n", where f is a real (float or double) printing the sum of the range given.

For the MPI version modify the `computeMPI.c` and the makefile so that "`make all`" also creates the MPI executable named "`parallel`". Modify the given `Makefile` to compile the new executable with a new *Makefile-rule*.

Problem 1, MPI Intro

a) In the *MPI* copy `computeMPI.c`, parallelize the serial program from `computeSerial.c` using MPI.

b) Implement a "`make all`" *Makefile*-rule which compiles and executes corresponding MPI file/executable in the given `Makefile`.

c) Analyze your implementation and report the following:

i) The amount of operations performed by your program as a function of $O(n), n = stop - start$, and the amount of operations performed per process P_i in your program with $i \in [1, 10]$.

ii) The amount of MPI operations performed by your program as a function of $O(P), P = NumberOfProcesses$.

iii) The average amount of MPI operations performed per process P_i in your program with $O(P), P = NumberOfProcesses$.

iv) The maximum number of MPI operations performed by any process P_i as a function of $O(P), P = NumberOfProcesses$.

v) The difference in run-time with $P \in [1, 2, 4, 8]$ and $n = [2 \to 100, 2 \to 10^6, 2 \to 10^9]$ when run on a machine in ITS015.
Make a graph for each value of P, showing the run-time for different increments of n. Each range of n should be split into 20 steps of equal increments.

Additional details can be found in the recitation slides for this problem set.