Parallel and Distributed Algorithms and Programs
TP n°1 - Getting hands dirty with MPI

and so dirty will they become...

Julien Braine
julien.braine@ens-lyon.fr

Laureline pinault
laureline.pinault@ens-lyon.fr

11/09/2017

Part 1

Bitonic arrays

Definition 1 *We call* ***V-shaped*** *a sequence which is either increasing and then decreasing or decreasing and then increasing. Thus, sequences* $\langle 2, 3, 7, 7, 4, 1\rangle$ *and* $\langle 12, 5, 10, 11, 19\rangle$ *are V-shaped.*

Definition 2 *We call* ***bitonic*** *a sequence which is a circular shift of a V-shaped sequence. For example, sequences* $\langle 4, 1, 2, 3, 7, 7\rangle$ *and* $\langle 11, 19, 12, 5, 10\rangle$ *are bitonic.*

Question 1

a) Open the script `gen-bitonic-array.py` and use it to generate a bitonic array (V-shaped in practice) in a file named `bitonic-array.txt`

Part 2

Sequential sort of bitonic arrays

We suggest to sort bitonic arrays with the following algorithm. The algorithm is as follows :

1. Let a be a bitonic array of size 2^k
2. Let $start = 0, size = 2^k$
3. For each $i \in [start, start + size[$, if $a[i] > a[i + size/2]$, swap cells i and $i + size/2$ in a .
4. If $size \neq 1$ do step 3 with $start = start, size = size/2$ and $start = start + size/2, size = size/2$.

The proof of correctness of this algorithm relies on the following invariant (we do not ask to show the correctness) : after step 3, the subarrays $A1 = a[start...start + size/2[$ and $A2 = a[start + size/2...start + size[$ have the following properties :

1. All elements of $A1$ are smaller than all elements of $A2$
2. $A1$ and $A2$ are bitonic

The sorting algorithm is written in file `sort-bitonic.c`.

Question 2

a) In file `sort-bitonic.c` edit the function `sort_sequential` and write this algorithm sequentially (recursive).

b) Test it on a few examples using `gen-bitonic-array.py`. (read the code of `sort-bitonic.c` to figure out how)

Part 3

Parallel sort of bitonic arrays

We wish to write a parallel algorithm using MPI for our bitonic sort. The key idea consists in parallelizing step 3. The algorithm can be described as follows :

1. We launch a number of MPI processes equal to the number of cells in the array to be sorted
2. $step = 2^k$
3. Each process number i receives exactly cell with index i : they do not own their own copy of the array (using MPI Scatter)
4. Each process i "talks" with process $i + step/2$ (using MPI Send and MPI Receive) so that process i receives the minimum and process $i + step/2$ receives the maximum of cells i and $i + step/2$.
5. $step = step/2$
6. if $step \neq 1$, redo 4
7. Retrieve the complete array (using MPI Gather)

Question 3

a) Understand the proposed parallel algorithm and convince yourself that it has same functionality

b) Implement it in the function `sort_parallel`

c) Check that it works on examples !

Part 4

Let's analyze our algorithms

Question 4

a) What is the complexity of the sequential algorithm (number of comparaisons) ?

b) What is the parallel complexity of the parallel algorithm (for simplicity, the number of comparaison of the process that does the most)

c) What is the complexity in communication of the parallel algorithm (for simplicity, the number of messages exchanged between processes overall)

d) Prove the correctness of the sequential algorithm

Part 5

Let's improve our algorithms

Question 5

a) How can you use bitonic sort to sort non bitonic arrays ?

b) Implement it and test it !

c) Make it work on arrays which have non power of two size

Part 6

Do not forget to keep a copy of the precious code you developed for later!