MP 2: Random Migrating Particles

CS 598 LVK: Parallel Migratable Objects

Fall 2014

Due date: Thursday, October 9, 10 PM

For the assignment, you will write code that simulates a set of particles moving randomly in a 2-dimensional space within a bounding box. The coordinates of the overall simulation box are between 0.0 and 100.0 along each dimension. The particles are divided among chares based on their coordinates. The chares should be organized in a 2-dimensional array of chares (using a chare array) of size *k x k*. So, each chare owns a bounding box of its own with size *100.0/k*.

Your program should generate *n* particles per chare during construction with a random (but valid, i.e. within the chare) position for particles. Your program should accept the number of particles per cell *n*, and *k* as command line parameters in that sequence.

Skeleton code: A base code for MP2 is added to your SVN repositories. Update your SVN repository by doing `svn update` and you should be able to see a folder named *mp2*. The skeleton code includes base code for Mainchare *Main*, 2-D Chare Array *Cell* and *Particle* class representing the particles the Chare Array contains. There are also comments in the skeleton code that will guide you through the assignment.

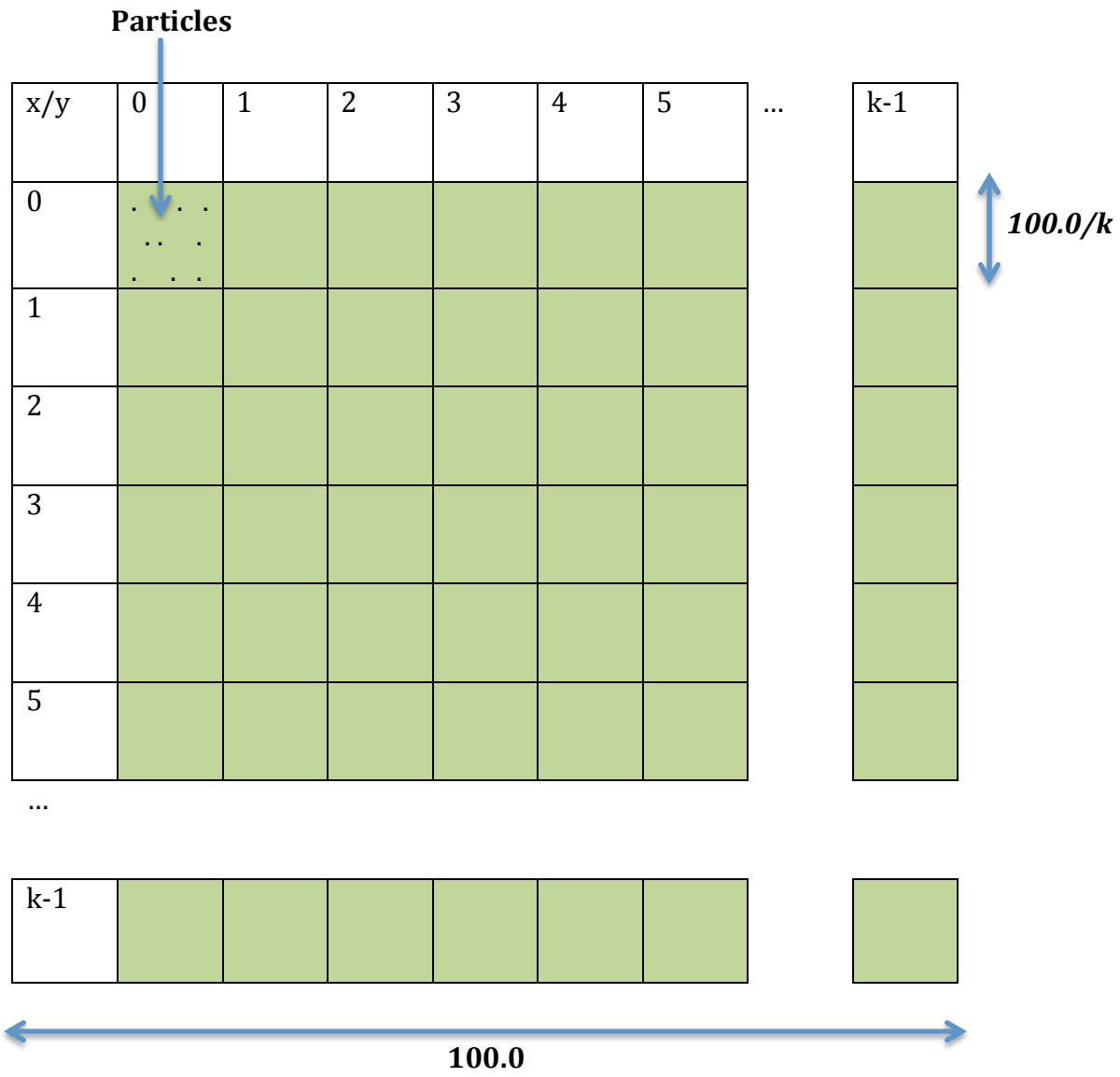
Expected Output: Your program should calculate and print to screen the maximum and total number of particles every 10 iterations. **Use the provided print function**. We will be automating the grading for this assignment and failure to use the provided function might result in a lower grade. Additionally, the simulation should not be delayed by this calculation (i.e. you should use reductions).

For testing your MP, you can use 10000 (=n) particles per chare, simulated over 100 steps and a chare array of size 16 x 16 (k=16). Experiment with different number of particles and chare array sizes as our test cases will use values in addition to the defaults.

Note: There might be multiple particles having the same *x* and *y* coordinates, especially if you increase the density of each cell. You do not need to handle this case separately; it is a valid case assumption.

Also, we have provided a default Makefile for your use. Feel free to modifiy it for testing but we will be using the default version of the Makefile to grade your

code. Therefore you should make sure your code compiles and runs using the default version.



- The pseudocode for the overall algorithm is:

```
for(iteration=0; iteration<ITERATION; iteration++){
```

- For each of the particles that belong to my chare: change its x and y coordinate by a small random amount.

- Move all the particles that do not belong to a chare's bounding box to their correct homes. Since the movement is small, this will mean communication to the eight near neighbor chares. Some of these messages may contain no particles.

- `if(iteration%10 == 0)`

    - Do reductions to calculate average and max number of particles

}

Submission:

- *SVN:* Check-in every file you modified. Also check-in one output log from execution in Taub. This should include the print statements your code makes so we can verify it is correct. Name this file taub_run.
- *Late penalty*: There will be a 20% penalty for late submissions. Only up to 1 day latency will be accepted.