Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu); Class: MW 14:00-15:50 @ UTS 308; Office Hr: M 13:00-14:00 & by appt.

Programming Assignment 4
(Due Wednesday, 6/3/15)

This assignment is to practice programming in the new parallel programming language, Chapel. You'll write data-parallel shared-memory programs, and use the domain mapping feature to convert them to PGAS-style distributed-memory programs.

For this assignment, all students will write the same set of programs. However, CS515 students are required to implement one extra set of functions within some of these programs (see below). This assignment carries a total of 100 points.

Preparation

If you haven't done so, run `addpkg` on the Linux system to add Chapel 1.11.0 to your environment. Set the environment variable `GASNET_SSH_SERVERS` to a listing of the Linuxlab machine names. For example, in C shell you can do

```
linux> setenv GASNET_SSH_SERVERS "adelie african catron chatham ..."
```

Set a second environment variable, `GASNET_SPAWNFN`, to `S`:

```
linux> setenv GASNET_SPAWNFN S
```

It tells Chapel to map locales to SSH nodes. This variable can also be set to `L`, which will map all locales to the current host node, allowing a multi-locale program be simulated on a single node.

Download the file `assign4.zip` to your CS Linux account and unzip it. You'll see several program files:

- `domMap.chpl` — a sample domain map program
- `jacobi.c` — a sequential Jacobi program in C
- `jacobi-shm.chpl` — a shared-memory Jacobi program in Chapel
- `Makefile` — for compiling programs

Part 1. Domain Mapping (20 points)

The file `domMap.chpl` contains a demo program for domain mapping. An $n \times n$ domain is both block-distributed and cyclic-distributed (n is default to 8). An array with elements set to their locale IDs are defined over the domain. When executed with four locales,

```
linux> ./domMap -nl 4
```

it produces the following output:

```
Block Array:            Cyclic Array:
0 0 0 0 1 1 1 1         0 1 0 1 0 1 0 1
0 0 0 0 1 1 1 1         2 3 2 3 2 3 2 3
0 0 0 0 1 1 1 1         0 1 0 1 0 1 0 1
0 0 0 0 1 1 1 1         2 3 2 3 2 3 2 3
2 2 2 2 3 3 3 3         0 1 0 1 0 1 0 1
2 2 2 2 3 3 3 3         2 3 2 3 2 3 2 3
2 2 2 2 3 3 3 3         0 1 0 1 0 1 0 1
2 2 2 2 3 3 3 3         2 3 2 3 2 3 2 3
```

The numbers are locale IDs of the corresponding domain elements.

Your task is to write a new program, `domMap2.chpl`, which treats the boarder elements of the domain and the inner elements of the domain differently. For a boarder element, the program simply displays a 0; for an inner element, it displays the element's locale ID, as before. When executed with four locales,

```
linux> ./domMap2 -nl 4
```

the new program should produce the following output:

```
Block Array:            Cyclic Array:
0 0 0 0 0 0 0 0          0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0          0 3 2 3 2 3 2 0
0 0 0 0 1 1 1 0          0 1 0 1 0 1 0 0
0 0 0 0 1 1 1 0          0 3 2 3 2 3 2 0
0 2 2 2 3 3 3 0          0 1 0 1 0 1 0 0
0 2 2 2 3 3 3 0          0 3 2 3 2 3 2 0
0 2 2 2 3 3 3 0          0 1 0 1 0 1 0 0
0 0 0 0 0 0 0 0          0 0 0 0 0 0 0 0
```

Try to find a simple solution with domain manipulation. You should make sure that your program works with different values of n and different numbers of locales.

Part 2. Jacobi and Gauss-Seidel (60 points)

This part consists of three tasks. In each task, you are asked to implement the Gauss-Seidel iteration method for Laplace Equation. If you are a CS515 student, you need to implement two versions of the method, the "natural index ordering" and the "red-black ordering". If you are a CS415 student, you can choose one of these two versions to implement.

1. The file `jacobi.c` contains a function, `jacobi()`, which implements the Jacobi iteration for Laplace Equation. Your task is to write two functions in the same style to implement the Gauss-Seidel iteration:
   - `gauss_seidel()` — Gauss-Seidel iteration with natural index ordering
   - `red_black()` — Gauss-Seidel iteration with red-black ordering

   Copy `jacobi.c` to a new file, `laplace.c`; add the two new functions to it; and modify the `main()` function to include additional calls to the new functions. Run and test your program. Obverse the convergence rates of the different iteration methods.

2. The file `jacobi-shm.chpl` is a shared-memory version of the the Jacobi iteration for Laplace Equation, written in Chapel. Read and understand the program first. Your task is similar, *i.e.* write two functions in the same style as the function `jacobi()` to implement the Gauss-Seidel iteration. Copy `jacobi-shm.chpl` to a new file, `laplace-shm.chpl`; add the two new functions to it; and modify the `main()` function to include calls to the new functions. Run and test your program.

3. This last part is to convert your shared-memory program, `laplace-shm.chpl`, to a PGAS-style distributed-memory program, `laplace-distr.chpl`, by using Chapel's domain mapping feature. Both the Jacobi function and the Gauss-Seidel function(s) need to be converted.

   Use block distribution for the domain map. There are two domains in the program, `D` and `Inner`, think clearly which one should be used as the base for domain map. Note that the program structure and the computation steps do not need to change.

   You need to find a way to verify that the mesh point updates are indeed computed distributively according to the block distribution. Include proper `writeln` statements in the program for this purpose. Guard them with the `verbose` flag, as in the given program.

Possible Alternative to Part 2

As promised, you have the option to implement a different algorithm of your own choice, instead of the programs specified in Part 2 above. (You still need to do Part 1.)

Important Note: If you choose this alternative, you need to talk to me about your algorithm and to get an approval. In general, you need to implement an algorithm that can be domain-mapped to a PGAS-style distributed-memory program. Possible candidates include other numerical algorithms and some sorting algorithms.

Additional Information on Chapel

The Chapel package is located in `/pkgs/chapel` on the Linux system. You may go there to find documents and examples. For further information on the language, you may visit the official Chapel website `chapel.cray.com`.

What to Turn In

As before, write a two-page summary documenting your experience with this assignment (**20 points**). Did you observe different convegence rates between Jacobi and Gauss-Seidel methods? Are the results consistent across the sequential, the shared-memory, and the distributed-memory versions? How do their performance compare? etc.

Make a `zip` or `tar` file containing your four programs and the summary report. Use the Dropbox on the D2L site to submit your assignment file.

Important Note: In past assignments, some students did not follow the naming requirement specified in the assignment handouts for their submitted programs, which caused inconvenience in my conducting automated testing. I've been lenient on this issue so far. For this assignment, however, if your programs are not named correctly, *i.e.*, `laplace.c`, `laplace-shm.chpl`, and `laplace-distr.chpl`, points will be deducted.