# CS4230 Parallel Programming

## Lecture 9:
## Locality and Data Parallel Algorithms

Mary Hall
September 20, 2012

# Administrative

- SVD assignment status
  - Problems with speedup on water
  - I suspect OMP implementation
  - Will resolve today in some way with Axel

# More locality: Image correlation

```
… Initialize th[i][j] = 0 …

 /* compute array convolution */
 for(m = 0; m < IMAGE_NROWS - TEMPLATE_NROWS + 1; m++){
   for(n = 0; n < IMAGE_NCOLS - TEMPLATE_NCOLS + 1; n++){
     for(i=0; i < TEMPLATE_NROWS; i++){
       for(j=0; j < TEMPLATE_NCOLS; j++){
         if(mask[i][j] != 0) {
              th[m][n] += image[i+m][j+n];
         }
       }
     }
   }
 }
/* scale array with bright count and template bias */
…
th[i][j] = th[i][j] * bc – bias;
```

# Example Data Parallel Algorithm

Problem 1 (#10 in Lin/Snyder on p. 111):

The Red/Blue computation simulates two interactive flows. An n x n board is initialized so cells have one of three colors: red, white, and blue, where white is empty, red moves right, and blue moves down. Colors wrap around on the opposite side when reaching the edge.

In the first half step of an iteration, any red color can move right one cell if the cell to the right is unoccupied (white). On the second half step, any blue color can move down one cell if the cell below it is unoccupied. The case where red vacates a cell (first half) and blue moves into it (second half) is okay.

Viewing the board as overlaid with t x t tiles (where t divides n evenly), the computation terminates if any tile's colored squares are more than c% one color. Use Peril-L to write a solution to the Red/Blue computation.

## Steps for "Localized" Solution

Step 1. Partition global grid for n/t x n/t processors

Step 2. Initialize half iteration (red) data structure

Step 3. Iterate within each t x t tile until convergence (guaranteed?)

Step 4. Compute new positions of red elts & copy blue elts

Step 5. Communicate red boundary values

Step 6. Compute new positions of blue elts

Step 7. Communicate blue boundary values

Step 8. Check locally if DONE

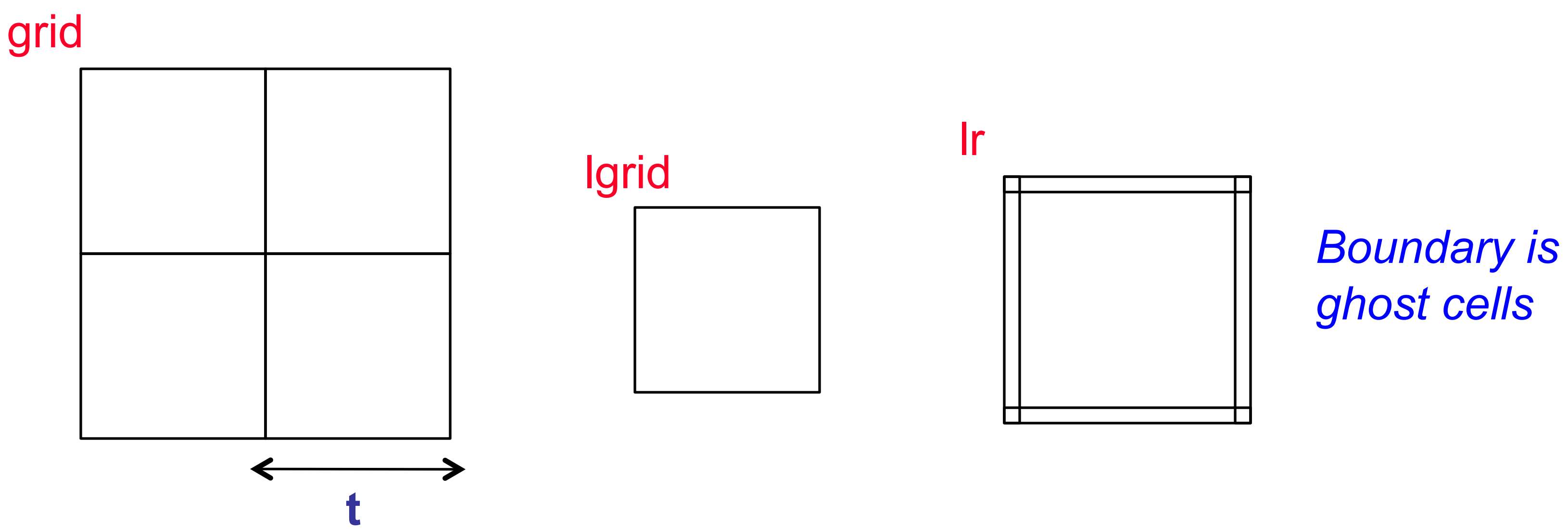
# Steps 1, 3 & 8. Partition Global Grid and Iterate

```
int grid[n,n], lgrid[t,t];
boolean gdone = FALSE;
int thr = (n/t)*(n/t);

forall (index in(0..thr-1))
    int myx = (n/t) * index/(n/t);
    int myy =(n/t) * (index % (n/t));
    lgrid[] = localize(grid[myx,myy]);
    while (!gdone) {
        // Steps 3-7: compute new locations and determine if done
        if (Bsum > threshold || Wsum > threshold || Rsum > threshold)
           exclusive { gdone = TRUE; }
        barrier;
  }
```

# Step 2: Initialization of lr the first time

```
int lr[t+2,t+2], lb[t+2,t+2];
// copy from lgrid  and grid to lr
lr[1:t,1:t] = lgrid[0:t-1,0:t-1];
lr[0,1:t] = grid[myx,myy:myy+t-1];
lr[t,1:t+1] = … ; lr[1:t,0] =…; lr[1:t,t+1] = ;
lr[0,0] = lr[0,t+1] = lr[t+1,0] = lr[t+1,t+1] = W; // don't care about these
```



# Steps 4 & 5: Compute new positions of red elts and Communicate Boundary Values

```
int lb[t+1,t+1];
lb[0:t+1,0:t+1] = W;
for (i=0; i<t+1; i++) {
   for (j=0; j<t+1; j++) {
       if (lr[i,j] == B) lb[i,j] = B;
       if (lr[i,j] == R && lr[i+1,j] = W) lb[i+1,j] = R;
       else lb[i,j] = R;
  }
}
barrier;
lgrid[0:t-1,0:t-1] = lb[1:t,1:t];  //update local portion of global ds
barrier;
//copy leftmost ghosts from grid
if (myx-1 >= 0)  lb[0,1:t] = grid[myx-1, 0:t-1];
else             lb[0,1:t] = grid[n-1,0:t-1];
```

# Steps 6 & 7: Compute new positions of blue elts and communicate Boundary Values

```
int lb[t+1,t+1];
lr[0:t+1,0:t+1] = W;
for (i=0; i<t+1; i++) {
   for (j=0; j<t+1; j++) {
      if (lb[i,j] == R) then lr[i,j] = R;
      if (lb[i,j] == B && lb[i,j+1] = W) lr[i,j+1] = B;
      else lr[i,j] = B;
  }
}
barrier;
lgrid[0:t-1,0:t-1] = lr[1:t,1:t];  //update local portion of global ds
barrier;
//copy top ghosts from grid
if (myy-1 >= 0)  lr[1:t,0] = grid[0:t-1,myy-1];
else             lr[1:t,0] = grid[0:t-1,0];
```

## Step 8: Compute locally if DONE

```
for (i=1; i<t+1; i++) {
   for (j=1; j<t+1; j++) {
         if (lr[i,j] == R) then Rsum++;
         if (lr[i,j] == B) then Bsum++;
         if (lr[i,j] == W) then Wsum++;
     }
}
```