CS4961 Parallel Programming

Lecture 7:
“Fancy” Solution to HW 2

Mary Hall

September 15, 2009

Administrative

- Programming assignment 1 is posted (after class)
- Due, Tuesday, September 22 before class
  - Use the "handin" program on the CADE machines
  - Use the following command:

    "handin cs4961 prog1 <gzipped tar file>"
- Mailing list set up: cs4961@list.eng.utah.edu

Using Intel Software on VS2008

- File -> New Project (C++)
- Right Click Project or Project -> Intel Compiler -> Use Intel Compiler
- Project -> Properties -> C/C++ -> General -> Suppress Startup Banner = No
- Project -> Properties -> C/C++ -> Optimization -> Maximize Speed (/O2)
- Project -> Properties -> C/C++ -> Optimization -> Enable Intrinsic Functions (/Oi)
- Project -> Properties -> Code Generation -> Runtime Library -> Multithreaded DLL (/MD)
- Project -> Properties -> Code Generation -> Enable Enhanced Instruction Set = Streaming SIMD Extensions 3 (/arch:SSE3)
- Project -> Properties -> Command Line -> Additional Options -> Add /Qvec-report:3
- Click Apply
- Your command line options should look like

```
/c /O2 /Oi /D "WIN32" /D "_DEBUG" /D "_CONSOLE" /D
"_UNICODE" /D "UNICODE" /EHsc /MD /GS /arch:SSE3 /fp:fast /
Fo"Debug/" /W3 /ZI /Qvec-report:3
```

Project 1 Assignment

- **PART I**
- Each of the files t1.c, t2.c, t3.c, t4.c and t5.c from the website cannot be vectorized by the ICC compiler. Your assignment is to produce the equivalent but vectorizable nt1.c, nt2.c, nt3.c, nt4.c and n5t.c. To determine whether or not the compiler has vectorized a loop in the code, look at the output of the compiler that results from the flag –vec-report3. If it says: "remark: LOOP WAS VECTORIZED.", then you have succeeded! Hints: There are several reasons why the above examples cannot be vectorized. In some cases, the compiler is concerned about the efficiency of the vectorized code. This may be because of the cost of alignment, concerns about exceeding the register capacity (there are only 8 128-bit registers on most SSE3-supporting platforms!) or the presence of data dependences. In other cases, there is concern about correctness, due to aliasing.

Project 1 Assignment, cont.

- **PART II**

  The code example youwrite.c is simplified from a sparse matrix-vector multiply operation used in conjugate gradient. In the example, the compiler is able to generate vectorizable code, but it must deal with alignment in the inner loop. In this portion, we want you to be a replacement for the compiler and use the intrinsic operations found in Appendix C (pp. 832-844) of the document found at: http://developer.intel.com/design/pentiumii/manuals/243191.htm. The way this code will be graded is from looking at it (no running of the code this time around), so it would help if you use comments to describe the operations you are implementing. It would be a good idea to test the code you write, but you will need to generate some artificial input and compare the output of the vectorized code to that of the sequential version.

Homework 2, Problem 1 Solution

Problem 1 (#10 in text 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 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

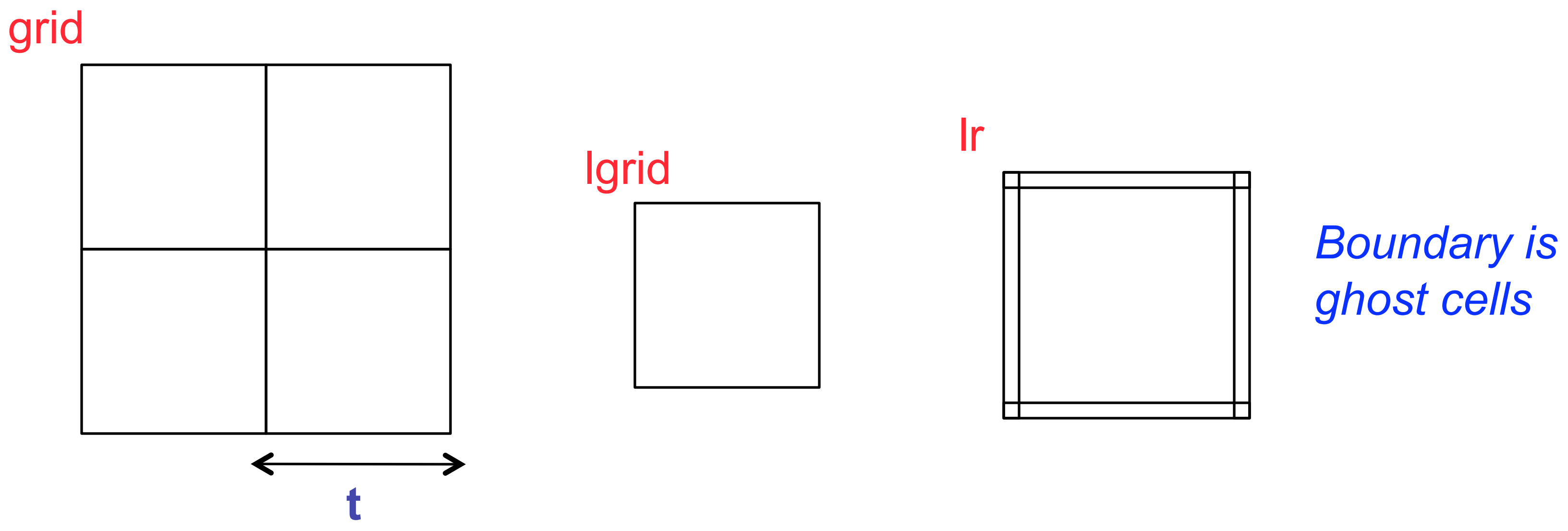
Steps 1, 3 & 8. Partition Global Grid and Iterate

```
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++;
     }
}
```

Next Time

- Parallel algorithms
- Introduction to threads