September 17th, 2015 Pre-Class Questions

Elliot Cartee

Question 0

I fell behind on the pre-class questions, and only started these questions on September 23rd. I would estimate these questions took me a little over an hour, although not continuously.

Question 1

I think I understood the slide decks and narration very well, but this is largely because I have an extensive background in numerical methods and specifically PDEs already.

Question 2

Fill in the most straightforward implementation you can think of for multiplying a compressed sparse row matrix by a vector.

I think this works, but I have not tested it or anything yet.

```
typedef struct csr_t {
    int  n;   /* Dimension of matrix (assume square) */
    int* pr;  /* Array of matrix nonzeros (row major order) */
    int* col; /* Column indices of nonzeros *
    int* ptr; /* Offsets of the start of each row in pr
                      (ptr[n] = number of nonzeros) */
    } csr_t;

    void sparse_multiply(csr_t* A, double* x, double* result) {
        int j=1;
        for(int i=0; i<n; i++) {
            result[i] = 0;
            while(j < ptr[i+1]) {
                result[i] += A[col[j]]*x[col[j]];
                j += 1;
            }
        }
}
```

Question 3

I filled out the method as follows (although I'm a little confused by the syntax)

```
double laplacian_u(double (*u)(double x, double y),
                   double h, double x, double y)
{
    double lap_x =  (u(x+h,y)-2*u(x,y)+u(x-h,y))/(h^2);
    double lap_y =  (u(x,y+h)-2*u(x,y)+u(x,y-h))/(h^2);
    return lap_x + lap_y;
}
```

Question 4

We see that this scheme updates each cell in a way that only depends on its neighboring cells at previous times. In other words, our scheme is local, so we can divide the domain into smaller sections that each processor is responsible for, including some overlapping "ghost cells". Because of these ghost cells, each processor can compute several time steps for the region it is responsible for until it needs communication from the other processors. Even with only one core available, such an organization could possibly mean that we are reusing recently computed data, and so we can keep things in cache and thus possibly make them faster.