Exercise 4: Pattern-Driven Optimization
July 10, 2014

What are the optimization strategies we could consider to alleviate each of the following potential performance bottlenecks.

- Multi-core (parallelization, synchronization)
- shared memory (sharing of data)
- Cache locality (reuse of data)
- Registers (fast access of data)
- CPU (computation efficiency)

Let's analyze the following kernels, identify their performance bottlenecks and optimization strategies. Try compose an optimization specification for each kernel and sketch the final optimized code.

1. Dense matrix-matrix multiplication kernels
   Kernel1:

```
  int i,j,k;
/*@;BEGIN(Nest1=Nest)@*/  for (j = 0; j < n; j ++)
  for (i = 0; i < n; i ++)
     c[j*n+i] = beta*c[j*n+i];
/*@;BEGIN(Nest2=Nest)@*/  for (k = 0; k < n; k ++)
/*@;BEGIN(Nest4=Nest)@*/  for (j = 0; j < n; j ++)
/*@;BEGIN(Nest3=Nest)@*/  for (i = 0; i < n; i ++)
    c[j*n+i] += a[k*n+i] * b[j*n+k];
```

2. Dense matrix-vector multiplication

```
   int i,j;
/*@; BEGIN(nest1=Nest) @*/
   for (i = 0; i < M; i += 1)
   {
     Y[i] = beta * Y[i];
/*@; BEGIN(nest2=Nest) @*/
     for (j = 0; j < N; j += 1)
     {
         Y[i] += A[j*lda+i] * X[j];
     }
   }
```

3. Sparse matrix-vector multiplication

```
   int row, j;
   /*@;BEGIN(Nest1=Nest)@*/
   for (row=0; row<n_rows; ++row) {
      /*@;BEGIN(Nest2=Nest)@*/
      for (j = rowstart[row]; j < rowstart[row+1]; j++) {
         dst[row] += val[j] * src[col[j]];
      }
   }
```

4. 7-point stencils

```
double fac;
double *temp_ptr;
int i, j, k, t;
fac = 6.0/(A0[0]*A0[0]);

/*@;BEGIN(Nest1=Nest)@*/for (t = 0; t < timesteps; t++) {
  /*@;BEGIN(Nest2=Nest)@*/for (k = 1; k < nz - 1; k++) {
    /*@;BEGIN(Nest3=Nest)@*/for (j = 1; j < ny - 1; j++) {
      /*@;BEGIN(Nest4=Nest)@*/for (i = 1; i < nx - 1; i++) {
        Anext[Index3D (nx, ny, i, j, k)] =
          A0[Index3D (nx, ny, i, j, k + 1)] +
          A0[Index3D (nx, ny, i, j, k - 1)] +
          A0[Index3D (nx, ny, i, j + 1, k)] +
          A0[Index3D (nx, ny, i, j - 1, k)] +
          A0[Index3D (nx, ny, i + 1, j, k)] +
          A0[Index3D (nx, ny, i - 1, j, k)]
          -  A0[Index3D (nx, ny, i, j, k)] *fac ;
      }
    }
  }
  temp_ptr = A0;
  A0 = Anext;
  Anext = temp_ptr;
}
```