CSE 5441
Programming Assignment 1
Due Friday, 2/23/2018, 9:00AM

For this programming assignment, you are to perform loop transformations to improve the performance of each of the following codes, for sequential execution on one core. Experimental results should be presented for execution on one core of the Oakley system at OSC. For this exercise interactive execution on a login node of Oakley will be sufficient (no batch jobs need be submitted).

Template codes (pa1-p1.c, pa1-p2.c, pa1-p3.c) will be provided on Carmen for each problem, with a base code and a second copy of the code that you should modify. A comparison of results from the baseline and the modified code will also be automatically performed by the provided template code and any differences reported. For pa1-p1.c, PAPI instrumentation for gathering hardware counter data will also be included in the template (you will need to add similar instrumentation for the other two codes). You may make any changes to the codes to improve performance, as long as the correctness tests pass.

1. (35 points) In some applications, there is a need to perform two matrix-vector products: $z = Ax$ as well as $y = A^T x$. The following code shows one way to implement it. But its performance is much less than desired. Perform loop transformations to improve performance (target performance with gcc: 1.4 GFLOPs and with icc: 2.2 GFLOPs; somewhat higher performance is feasible for both compilers than the specified target).

```
double x[N],y[N],z[N],A[N][N];
int i,j;
 for(i=0;i<N;i++)
  for(j=0;j<N;j++)
  {
   y[j] = y[j] + A[i][j]*x[i];
   z[j] = z[j] + A[j][i]*x[i];
  }
```

2. (30 points) In the following code, the sum of squares of elements is needed along the middle dimension of a 3D array. Make changes to improve performance (target is 1.5 GFLOPs; I don't think much higher is easy).

```
int i,j,k;
double sum,x[N][N][N],y[N][N];
for(i=0;i<n;i++)
 for(k=0;k<n;k++)
 {
  sum = 0.0;
  for(j=0;j<n;j++)
   sum += x[i][j][k]*x[i][j][k];
  y[i][k] = sum;
  }
```

3. (35 points) The following code is an example of a tensor contraction, a higher dimensional generalization of matrix multiplication. Use loop transformations to improve performance. Target is 1.5 GFLOPs for gcc and 2 GFLOPs for icc (somewhat higher performance should be feasible; I'd be interested in seeing how much higher anybody is able to achieve on this code).

```
int i,j,k,l;
double x[N][N][N],y[N][N][N],z[N][N];
for(i=0;i<n;i++)
 for(j=0;j<n;j++)
  for(k=0;k<n;k++)
   for(l=0;l<n;l++)
    z[l][k] += x[l][i][j]*y[i][j][k];
```

Present results both with the Intel icc compiler (icc -O3) as well as the GNU gcc compiler (gcc -O3).

Submit a detailed report explaining the main reason for low performance of the provided reference version, what you did to improve performance, and the improvements achieved. Use hardware counter reports as well as vectorization reports in attempting to explain the impact of code changes.

Upload your source code for each problem on Carmen, with suitable comments in the source code. Also bring a hard-copy of your report to class.