CS 378: Programming for Performance
Assignment 3: Loop transformations
Due: March 3rd

February 23, 2015

Late submission policy: Submission can be at the most 2 days late. There will be a 10% penalty for each day after the due date (cumulative).

You have used performance counters to measure the metrics that impact performance. You have learnt a way to programmatically solve computational science equations. In this assignment, we will focus on improving the single-thread performance of a solution to the 2-dimensional heat diffusion equation, by using loop transformations. The code for that can be found at:
http://www.cs.utexas.edu/~pingali/CS378/2015sp/assignments/heat2d.c.

For each loop transformation (including the baseline):

- Measure the performance metrics that are relevant and correlate with execution time.
- Compile your code in ICC with flags '-O3 -fp-model precise'.
- Submit your run to the job scheduler on Stampede at TACC - use the 'serial' queue. Since the values you obtain will depend a lot on the machine you use, you must use Stampede for the numbers you report.
- Verify that the output matches that of the baseline.

Plot the execution time and performance metrics of the different loop transformations and analyze them.

Baseline

The baseline kernel has this loop structure:

```
for t from t_min to t_max
  for i from i_min to i_max
    for j from j_min to j_max
      S(t, i, j)
```

Loop reordering 15 points

In the first assignment, you had experimented with loop reordering for matrix multiplication code. In this case, interchange only the inner two loops to have such a loop structure:

```
for t from t_min to t_max
  for j from j_min to j_max
    for i from i_min to i_max
      S(t,i,j)
```

Loop unrolling 15 points

Unroll the innermost loop to have such a loop structure:

```
for t from t_min to t_max
  for i from i_min to i_max
    for j from j_min to j_max in increments of 4
    {
      S(t,i,j)
      S(t,i,j+1)
      S(t,i,j+2)
      S(t,i,j+3)
    }
```

Loop tiling 15 points

Block or tile the innermost two loops to have such a structure:

```
for t from t_min to t_max
  for ii from i_min to i_max in increments of 32
    for jj from j_min to j_max in increments of 32
      for i from ii to ii+31
        for j from jj to jj+31
          S(t,i,j)
```

Tile Sizes 30 points

The code above uses a tile size of 32x32. Experimentally determine a tile size that performs the best. Report at least 5 variants (preferably those with distinctive behavior), including the fastest one.

Fastest 25 points

Validity: Is any loop transformation valid? For example, can we interchange t and i loops in the baseline kernel? You will find that this will yield incorrect results. One way to verify the validity of a transformation is to execute the transformed code and compare the output with that of the original code. We will study a systematic way to determine validity of transformations later in class.
Combining transformations: As you might have realized by now, loop transformations can be combined and used simultaneously. Loop tiling itself is a combination of stripmining and loop reordering.
Goal: Write a very fast 2-dimensional heat diffusion equation solver. Your code must work for any input size. You may use any of the techniques described above or from the literature but your code should not call any external library. Your code should be compiled with "icc -O3 -fp-model precise heat2d.c -o heat2d" and run with "./heat2d 2>out_heat2d". Report its execution time and performance metrics in the plots. Briefly describe your transformations and optimizations in the report. You will be graded on performance. You will not get any points if we are not able to reproduce your results for the given input size and verify correctness for a randomly chosen input size.

Deliverables

Submit (to canvas) your fastest performing code and a report (PDF) containing all plots and analysis.