Weekly 3 – MPHP

Tycho Canter Cremers

Task 1

Task 2

A.

Why is the outer loop not parallel?

Because acum is declared outside the loop the same memory location is accessed and written to in different iterations possibly at the same time. So “B[i,0] = accum;” has a possibility of using the wrong value. To solve it you should privitize accum and declare it inside the loop.

```
1. for i from 0 to N-1 // outer loop
2.        accum;
3.        accum = A[i,0] * A[i,0];
4.        B[i,0] = accum;
5. for j from 1 to 63 // inner loop
6.        tmpA = A[i, j];
7.        accum = sqrt(accum) + tmpA*tmpA;
8.        B[i,j] = accum;
```

Is the inner loop parallel?

Again, no. Because accum is, once again declared outside the loop. It could be written too, non sequentially, in another iteration resulting in the wrong value at B[i,j]. Additionally, sqrt is not an associative operation, so can’t be used in parallel in this case.

Can the inner loop be re-written as a composition of parallel operators?

No, because taking the square root of accum is not associative.

If line 6 is re-written as `accum = accum + tmpA*tmpA,` would it be possible now to rewrite the inner loop as a composition of bulk operators?

Now you can. It’s basically a map to calculate **`tmpA*tmpA`** and then a scan.

D.

When you use the make file to run both of the original Program and the transformed program I get this output. The Transformed program is almost 10 times faster than the original, non-transposed one.

```
[mns267@a00332 Task1-2]$ ./matrixMult_2
Original Program on GPU runs in: 46255 microsecs
GPU PROGRAM   VALID!
[mns267@a00332 Task1-2]$ ./matrixMult_3
Transformed Program on GPU runs in: 5154 microsecs
GPU PROGRAM   VALID!
```

Task 3

d.

```
./matrixMult
Sequential Naive version runs in: 5393490 microsecs
GPU version runs in: 5729 microsecs
Performance= 374.84 GFlop/s, Time= 5729.000 microsec 64 64
VALID RESULT!
```

Above is the output for the code ran with the tiled kernel. The tiled parallel version is almost 1000 times faster than the sequential version.

```
./matrixMult
Sequential Naive version runs in: 5409844 microsecs
GPU version runs in: 17280 microsecs
Performance= 124.28 GFlop/s, Time= 17280.000 microsec 64 64
VALID RESULT!
```

Running the code with the normal kernel runs about a little more than 3 times slower than the tiled kernel. But still a lot faster than the sequential version.