Yet Another
Matrix Multiplication Case Study

Or,

What I Learned Last Summer

by

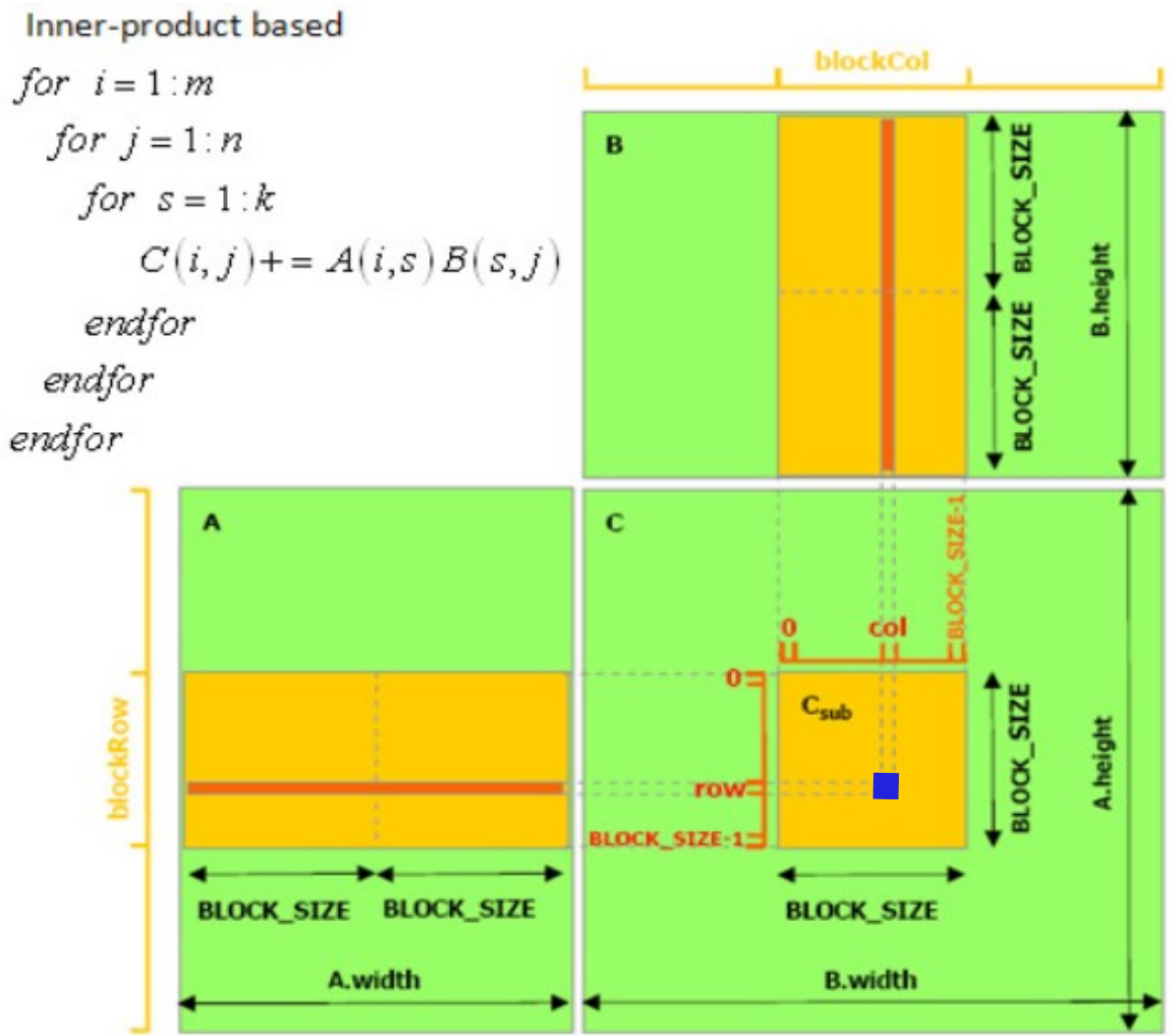
Chris McClanahan

Basic SGEMM

Square sub matrices are stored in shared memory

Each thread outputs one element of C

High occupancy



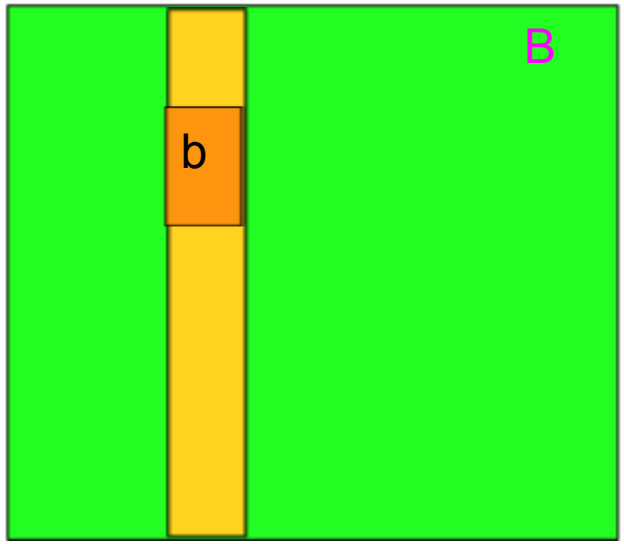
Jacket 1.4 SGEMM

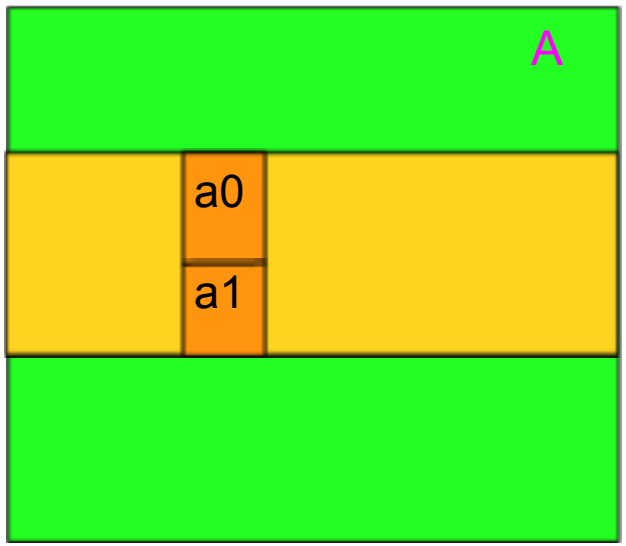
2 sub matrices output per thread block

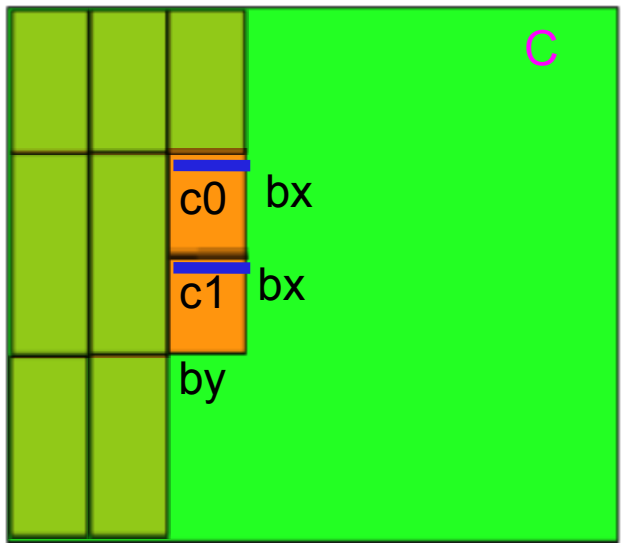
Each thread produces 2 rows of c

Non-square block sizes (bx > by)

a,c in registers | b in shared memory







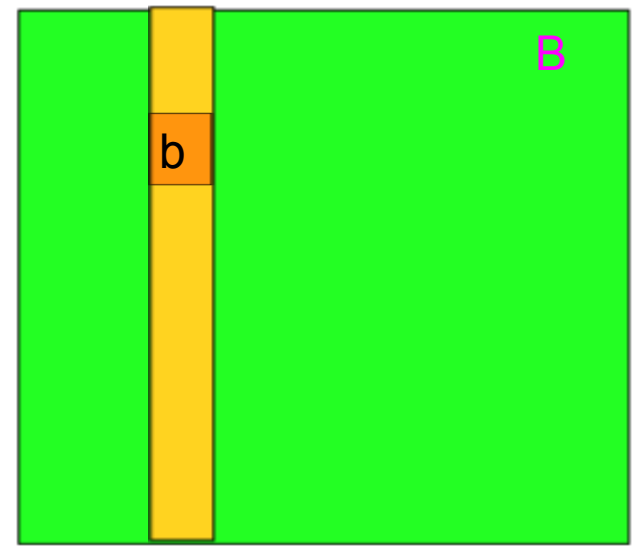
Jacket 1.5 SGEMM

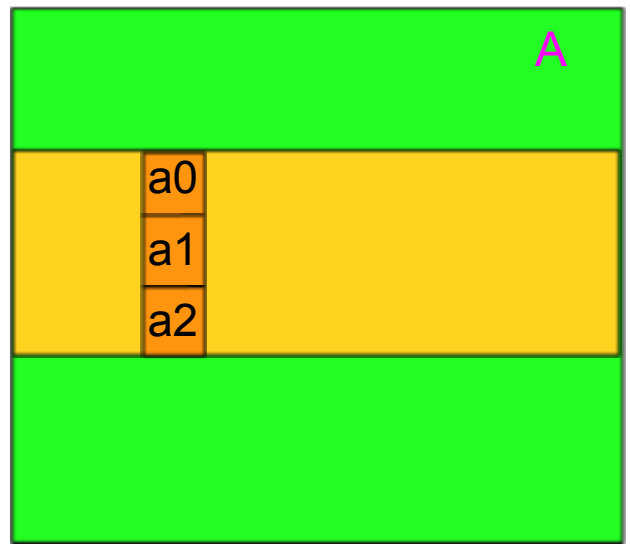
3 sub matrices output per thread block

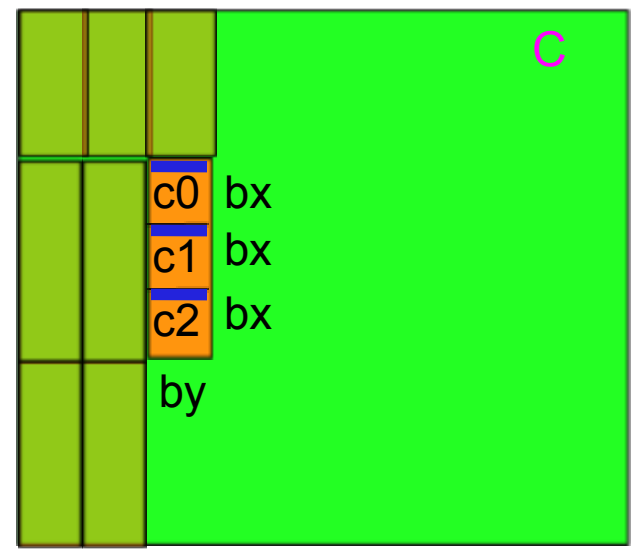
Each thread produces 3 rows of c

Non-square block sizes (bx > by)

a,c in registers | b in shared memory





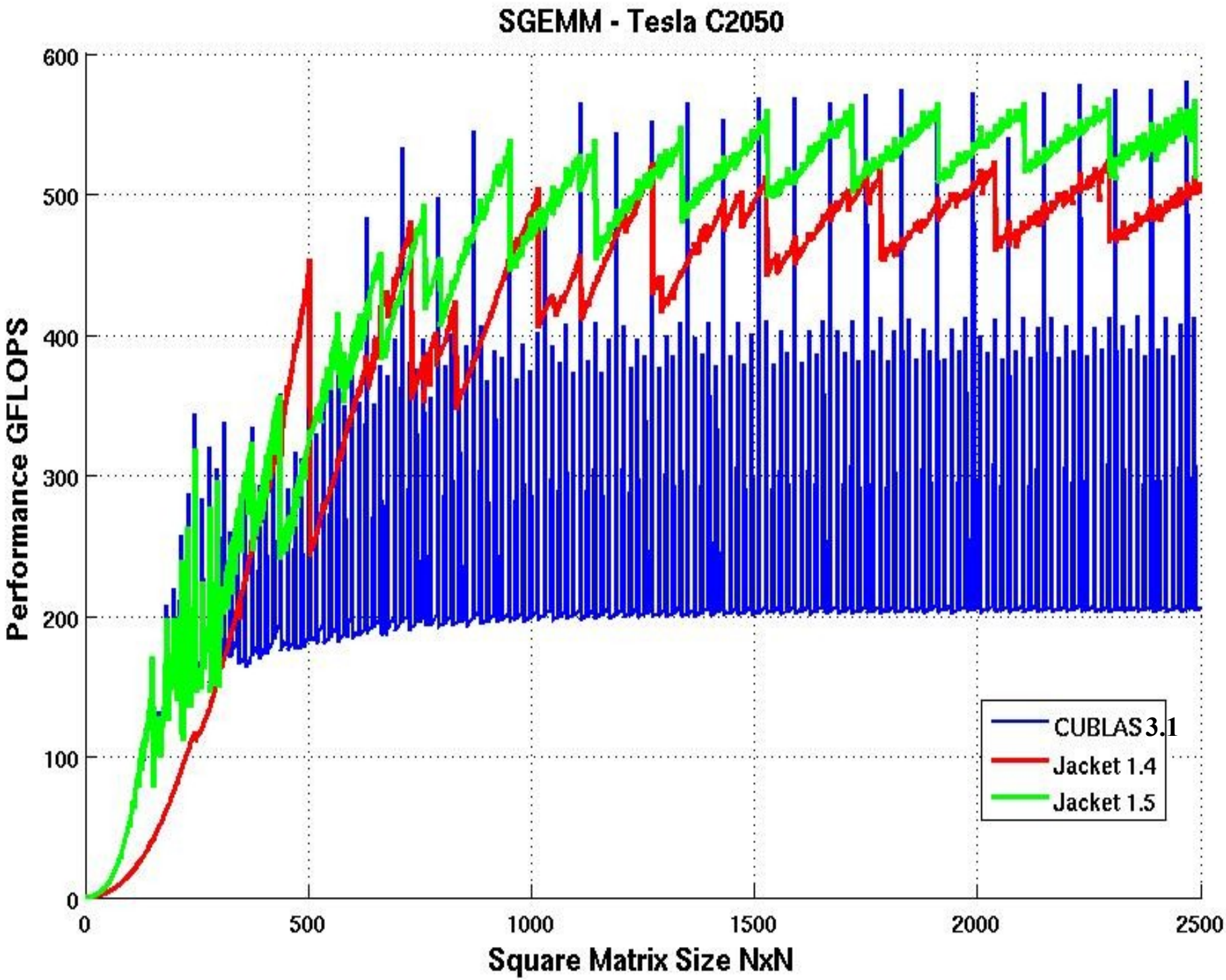


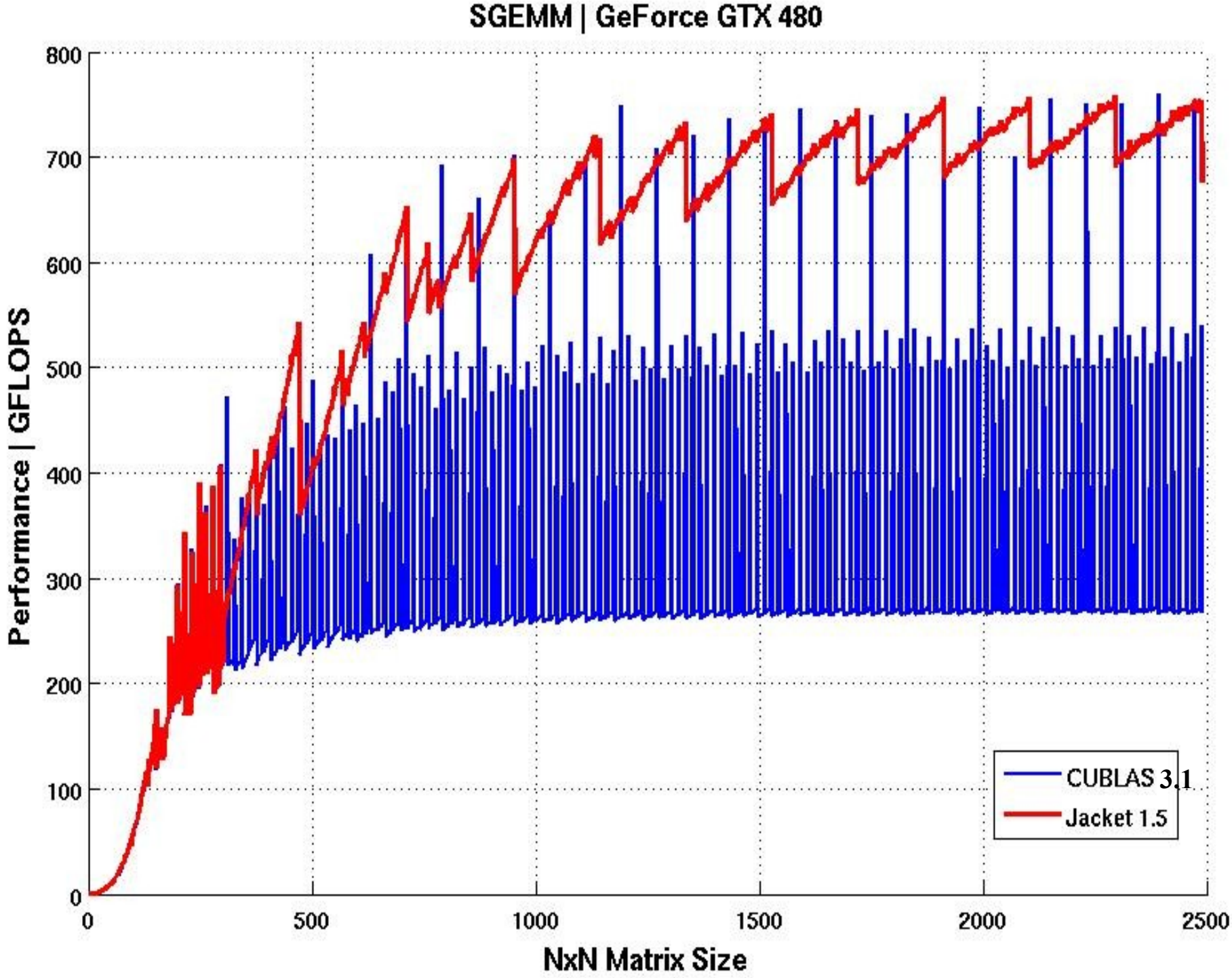
Putting registers to work

1. Load *b* from texture B → shared memory
2. Load *a* from texture A → register
3. Re-use *b* in registers
4. Perform outer product: *c* += *a* * *b* all in registers
5. Store *c* → global memory

Ex.. 3 vectors output per thread:

```
#pragma unroll
for ( i = 0; i < BLOCK_SIZE_X; ++i ) {
    float A0_reg = fetchA(iA0); iA0 += p.lda;
    float A1_reg = fetchA(iA1); iA1 += p.lda;
    float A2_reg = fetchA(iA2); iA2 += p.lda;
#pragma unroll
    for ( j = 0; j < BLOCK_SIZE_Y; ++j ) {
        bb_reg      = *(bb_ptr + j      );
        *(c0 + j) +=  (A0_reg * bb_reg);
        *(c1 + j) +=  (A1_reg * bb_reg);
        *(c2 + j) +=  (A2_reg * bb_reg);
    }
    bb_ptr += (BLOCK_SIZE_Y + 1);  // bb_ptr = &bb[i][0]
}
```


SGEMM - Tesla C2050
Performance GFLOPS
Square Matrix Size NxN
0
100
200
300
400
500
600
0
500
1000
1500
2000
2500
CUBLAS 3.1
Jacket 1.4
Jacket 1.5


SGEMM | GeForce GTX 480
Performance | GFLOPS
NxN Matrix Size
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000
2500
CUBLAS 3.1
Jacket 1.5

Heuristics

How many threads per block?
- Problem size / (block size * number of outputs per thread)

Sizes of thread blocks?
- As large as register usage allows
- Ideally a power of 2 (odd sizes can actually work, but are slower)

Shape of thread blocks? (for matrix multiply)
- Square: mathematically optimal, handles general sizes well
- Rectangle: can optimize for linear memory layout

Account for register spill?
- Watch output of --ptxas-options=-v

Chuck Norris doesn't optimize code... code runs faster because it's scared of him.

References

Vasily Volkov's papers

Lung-Sheng Chien paper

NVIDIA Forums

More Info

http://wiki.accelereyes.com/wiki/index.php/MTIMES_Benchmarks