Double-precision Matrix Multiply on CUDA

Parallel Computation (CSE 260), Assignment 2

Andrew Conegliano (A53053325) Matthias Springer (A99500782) GID G-338-665

February 6, 2014

0.1 Assumptions

The following assumptions apply within this work.

- All matrices are square matrices.
- All matrices consist of double-precision floating point values (64-bit doubles).

0.2 Notation

In this work, we use the following notation and variable names.

- n is the size of one dimension of the matrices involved. I.e., every matrix has n^2 values.
- When referring to matrices, A and B denote the source matrices and C denotes the target matrix, i.e. $AB = C$.
- b_i, b_j and b_k denote the block size for each *dimension* i, j, and k, respectively[1].

1 Runtime Environment

We optimized our DGEMM implementation for a specific runtime environment. All benchmarks and performance results are based on the following hardware and software.

1.1 Hardware

1.2 Software

2 Basic Algorithm

[1] We tried multiple levels of blocking and it is evident from the context which level of blocking we are refering to.

```
for i = 0 .. n - 1
     for j = 0 .. n - 1
          cij = C[i*n + j]

          for k = 0 .. n - 1
               cij += A[i*n+k] * B[k*n+j]

          C[i*n+j] = cij
```

Figure 1: Naive matrix multiplication algorithm.