HW06 Task1 a

About Level 1, Level 2 and Level 3:

1. **Level 1 Functions:**
   These operations pertain to vector-vector operations and encompass tasks such as vector addition, subtraction, dot products, and various element-wise operations. These operations are conducted on one-dimensional arrays, commonly known as vectors. They are typically less computationally intensive, involving fewer arithmetic operations. Level 1 functions exhibit lower memory and data transfer demands as they primarily operate on vectors.
2. **Level 2 Functions:**
   These operations are classified as matrix-vector operations and encompass tasks such as matrix-vector multiplication, solving linear systems, and orthogonalizing vectors. These operations involve both a matrix and a vector. Level 2 functions are typically more computationally demanding than Level 1 operations but less so than Level 3 operations. They require a moderate amount of memory and data transfer.
3. **Level 3 Functions:**
   These belong to the category of matrix-matrix operations, which encompass tasks like matrix multiplication and updating matrix products. These operations involve two matrices. Level 3 functions are the most computationally intensive, demanding a significant number of multiplications and additions. They also have the highest memory and data transfer requirements.

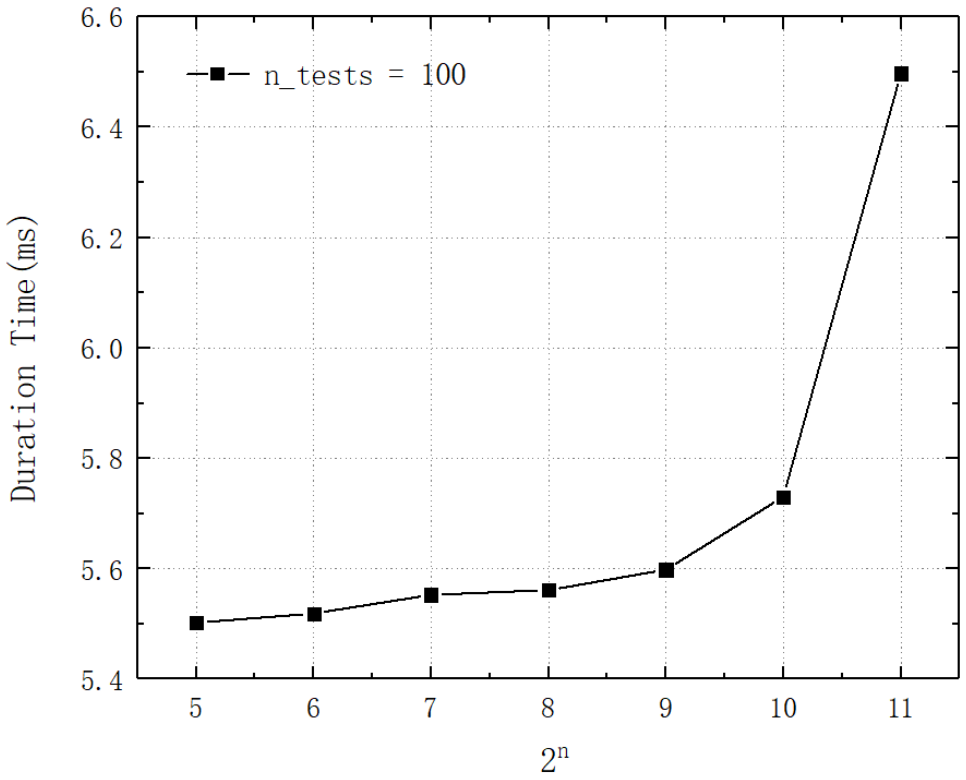
Here are some key considerations for the categorization of BLAS functions:

1. **Computational Complexity:**
   BLAS functions exhibit varying levels of computational complexity. Level 1 functions typically involve element-wise operations with low computational complexity. Level 2 functions encompass matrix-vector operations with moderate complexity. Level 3 functions perform matrix-matrix operations and are usually the most computationally intensive. Categorizing them into three levels helps developers optimize and parallelize these functions more effectively.
2. **Memory Access Patterns:**
   BLAS functions at different levels have distinct memory access patterns. Level 1 functions primarily operate on vectors, while Level 3 functions require simultaneous access to two matrices. This difference influences the optimization of data transfer and memory access.
3. **Hardware Characteristics:**
   BLAS libraries are used on various types of hardware, including CPUs and GPUs. The grouping of functions can be optimized based on the performance characteristics of the target hardware. For example, GPU-based parallel computing may benefit from parallelization of Level 3 functions, while CPU cache performance may impact the performance of Level 1 and Level 2 functions.

HW06 Task1 b

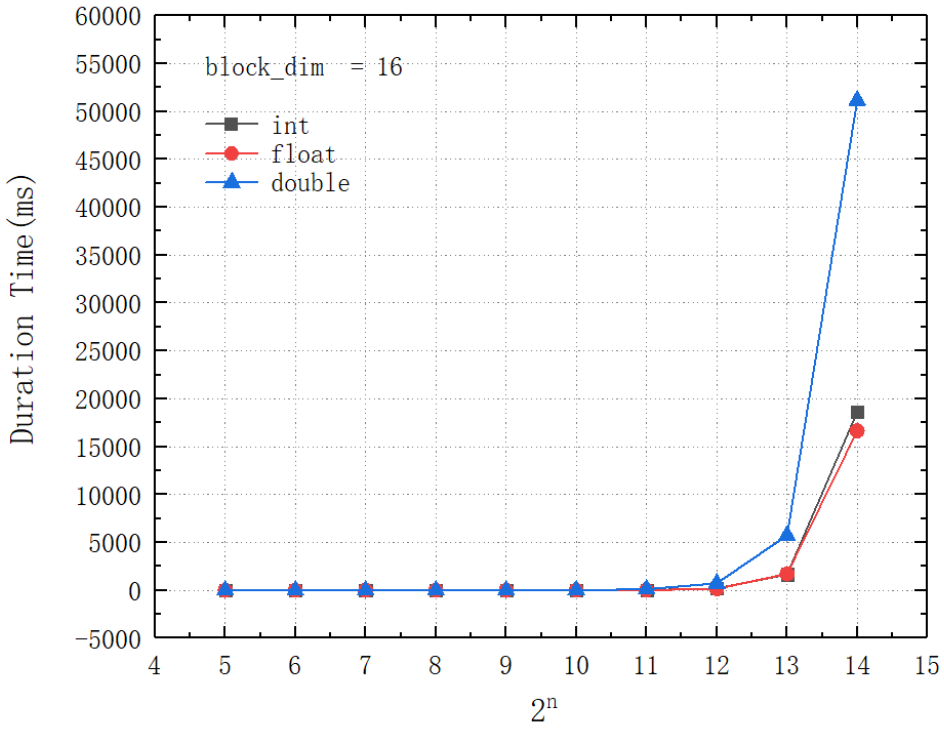
1. **Symmetric Matrix-Vector Multiplication (Level 2):**
   This specialized function assumes that the input matrix is symmetric, indicating that it is identical to its transpose. Leveraging this symmetry, the function optimizes computations by reducing the count of multiplications and additions needed. Rather than separately computing both A * x and A^T * x, it calculates only one of them, effectively halving the computational workload.
2. **Triangular Matrix Solve (Level 2):**
   Another specialized function assumes that the input matrix is triangular, either in the upper or lower triangular form. Triangular matrices exhibit all-zero elements below or above their main diagonal. This function streamlines the computation process for solving linear systems involving triangular matrices. It eliminates redundant operations with zero elements, leading to a substantial reduction in computational complexity.

HW06 Task1 e



HW06 Task1 f

HW05 Task1 b

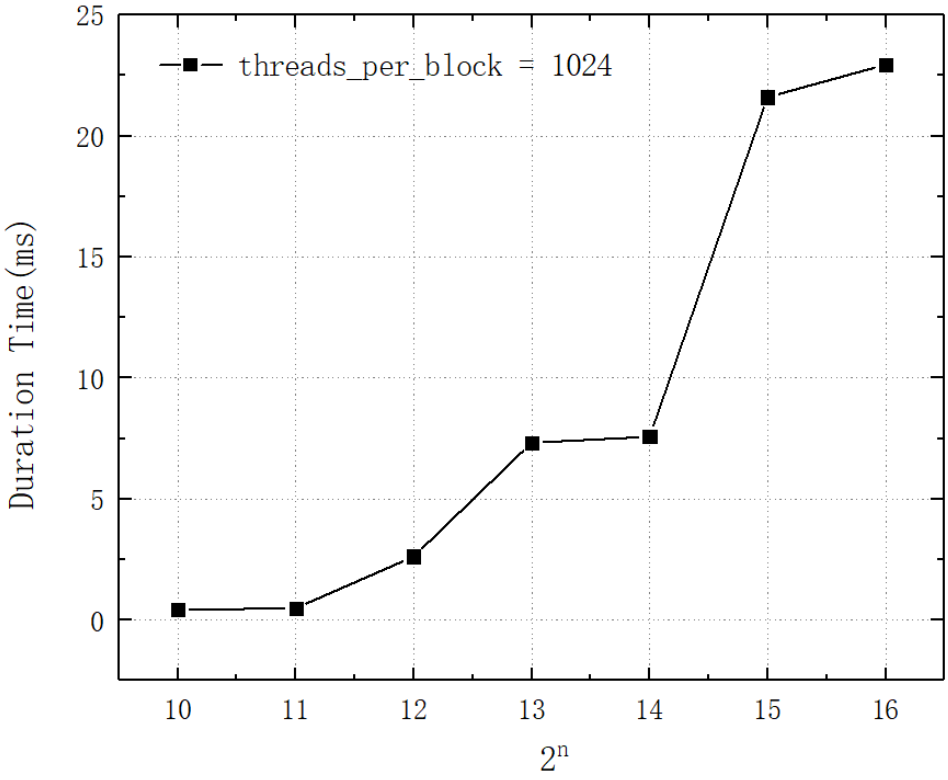


The matrix multiplication code using shared memory in HW05, involving a tiled-based approach to utilize shared memory, can be more efficient for smaller problem sizes where the overhead of launching the cuBLAS library might outweigh the computation. This is because shared memory reduces the memory latency and improves data reuse, making it beneficial for smaller matrix sizes that can fit within the shared memory and take full advantage of these optimizations. However, cuBLAS is highly optimized and designed to fully exploit the computational capabilities of GPUs. For larger problem sizes, where the matrix dimensions are substantial, cuBLAS is often significantly faster than hand-optimized shared memory matrix multiplication. This is because cuBLAS uses specialized algorithms and is optimized for various GPU architectures.

HW06 Task2 a

```
3.07993
0.451328
```

HW06 Task2 b


threads_per_block = 1024
Duration Time(ms)
25
20
15
10
5
0
10
11
12
13
14
15
16
2^n