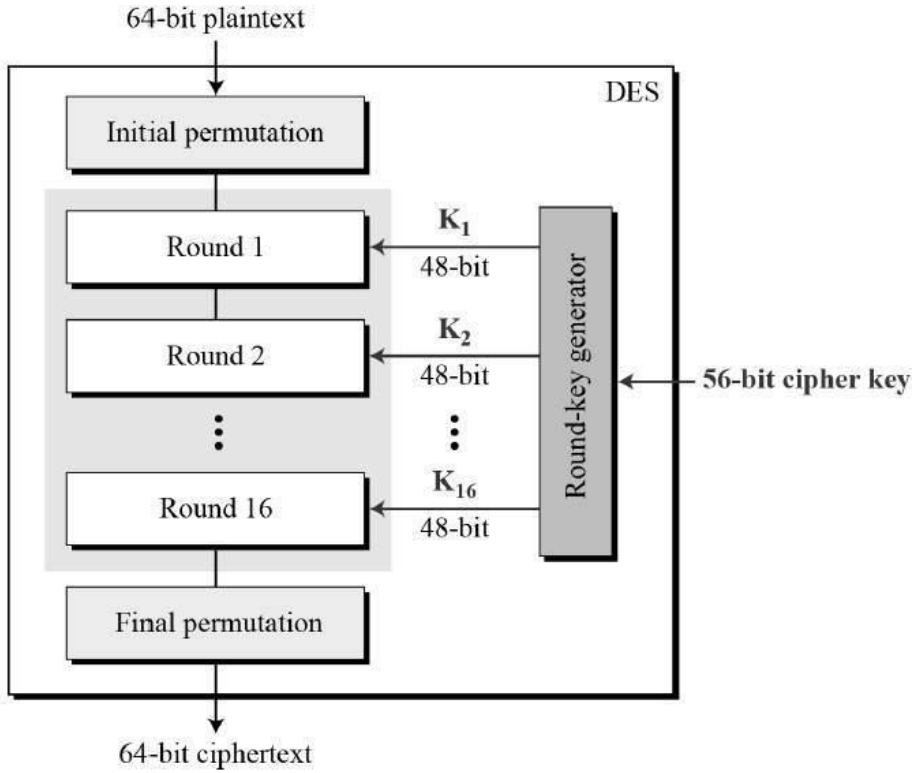
PARALLELIZATION OF DES ALGORITHM

RICCARDO FIORINI, MARIELLA MELECHI, ERICA RAFFA

ALGORITHM INTRODUCTION



- Plain text is read from a text file and stored in a string
- The initial text is divided into string depending on the number of threads
- Each thread simultaneously encrypt/decrypt a part of the text

CPU SYSTEM INFORMATION

Processor: AMD Ryzen 7 Mobile 4800H

Cores: 8

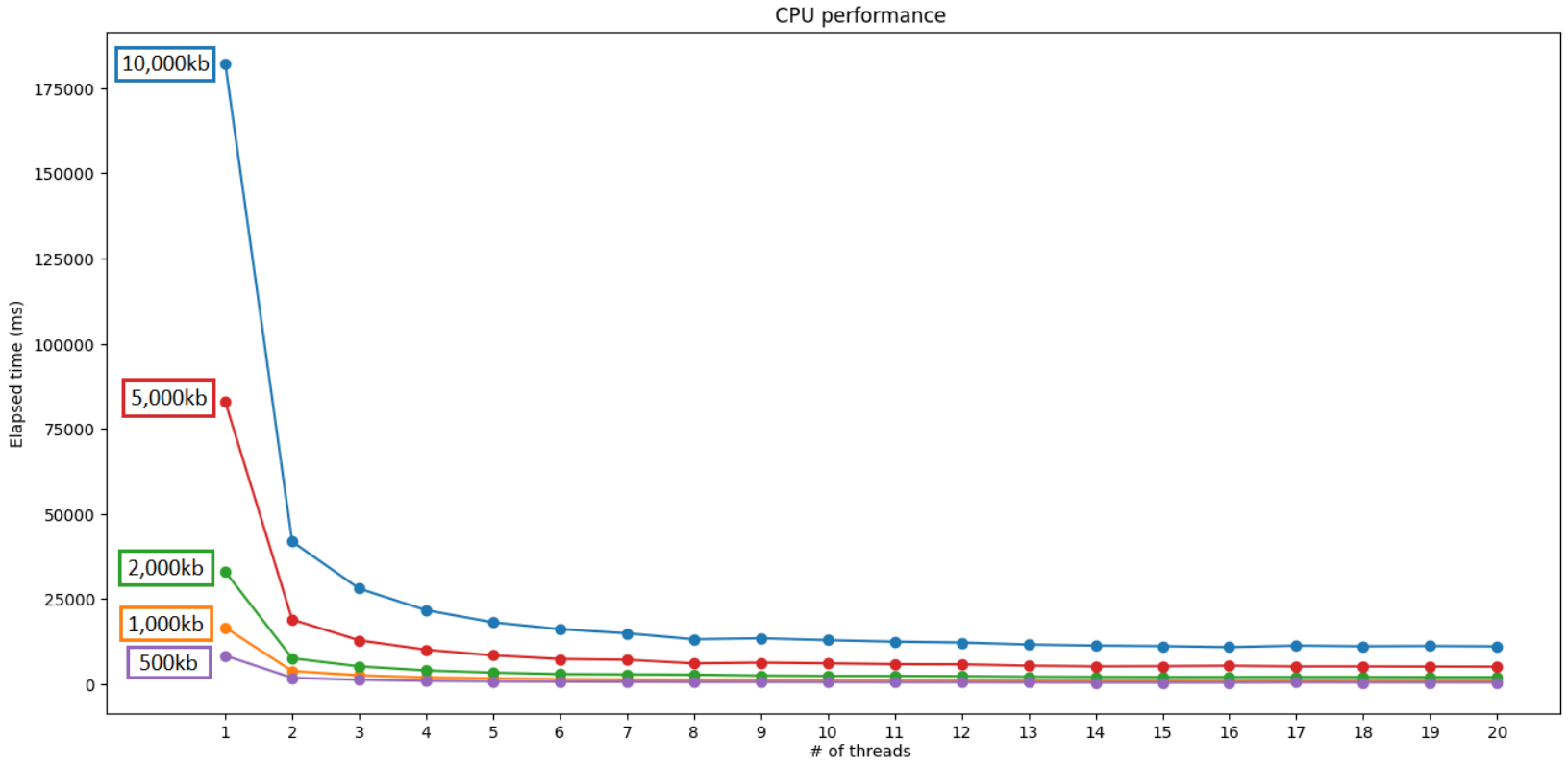
Hardware Threads: 16

Cache:

L1 Data	8 x 32 KBytes	8-way
L1 Inst.	8 x 32 KBytes	8-way
Level 2	8 x 512 KBytes	8-way
Level 3	2 x 4 MBytes	16-way

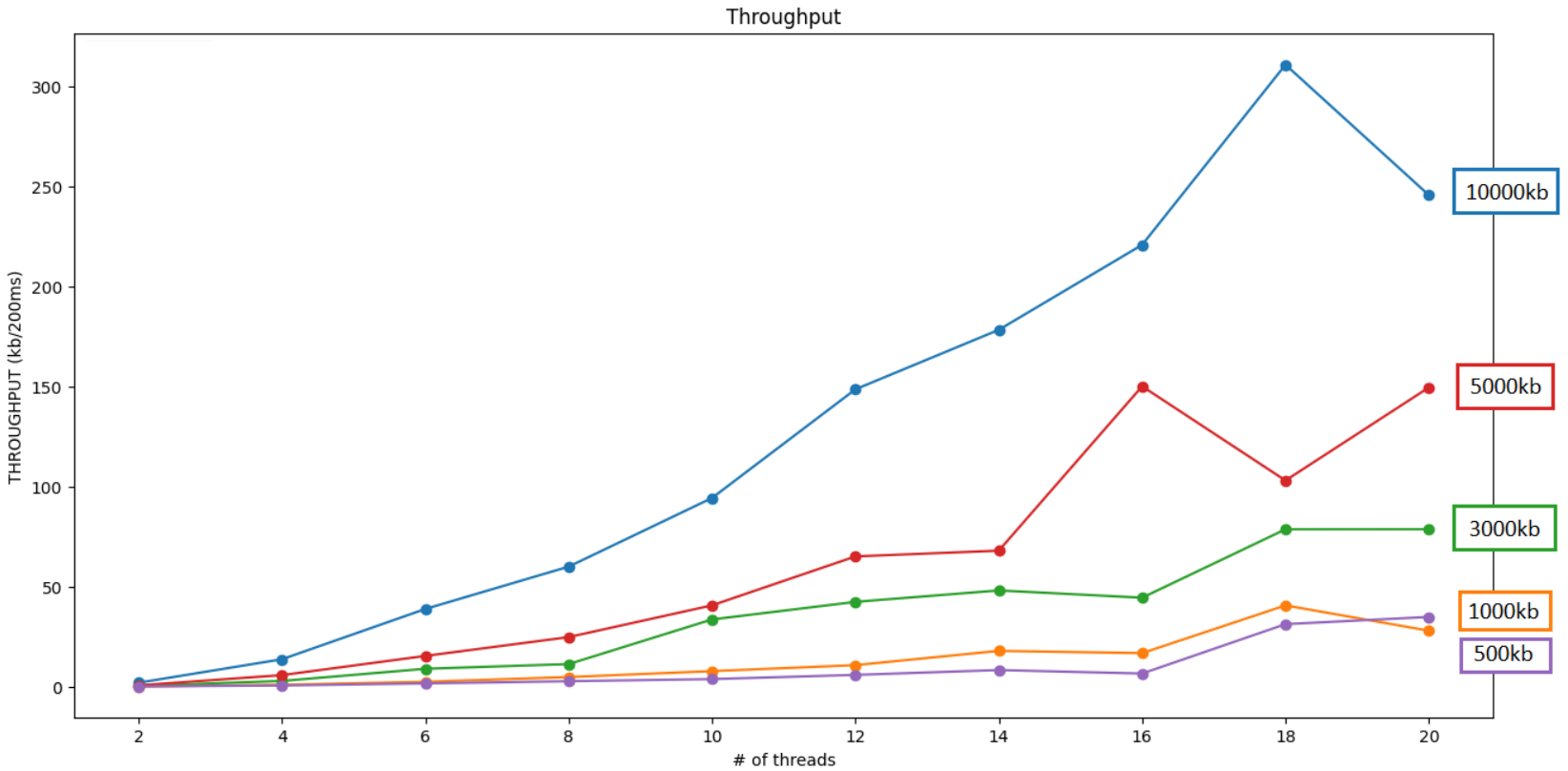
ELAPSED TIME VS NUMBER OF THREADS

- **Mean Elapsed Time** computed with 30 runs per thread
- **#threads => 16** : elapsed time decreases
- The algorithm is highly **scalable**



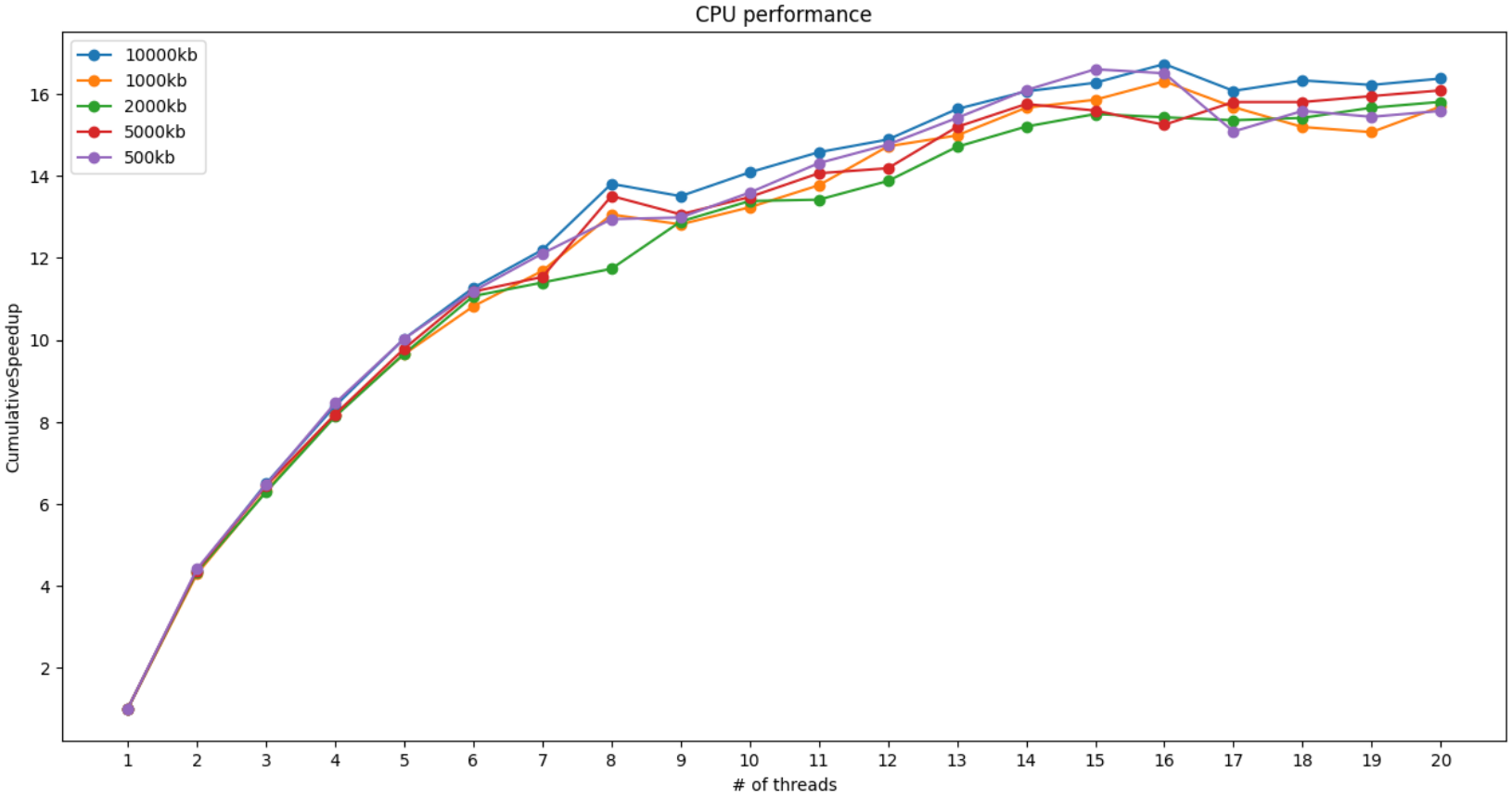
THROUGHPUT VS NUMBER OF THREADS

- **Mean Throughput** computed with 30 runs per thread
- **#threads => 16** : throughput increases



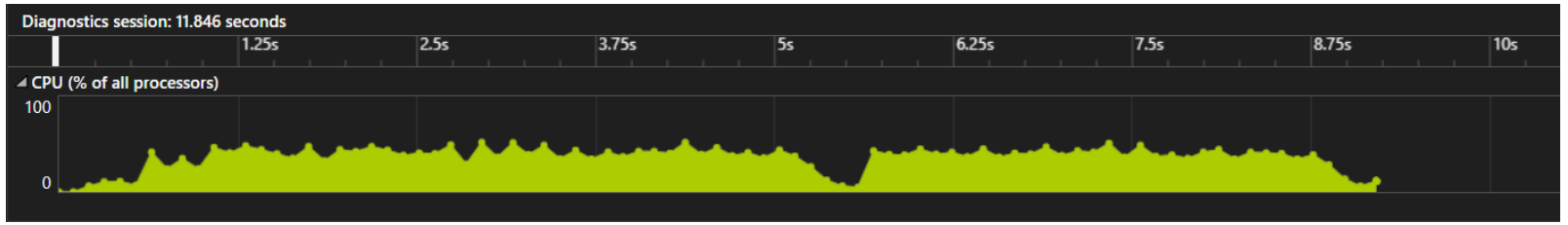
SPEEDUP VS NUMBER OF THREADS

- **Cumulative-Speedup** computed with the Mean Elapsed Time
- **#threads => 16** : speedup increases

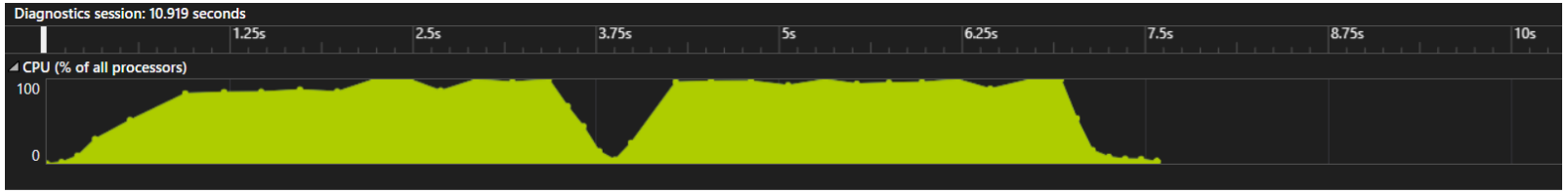


CPU UTILIZATION

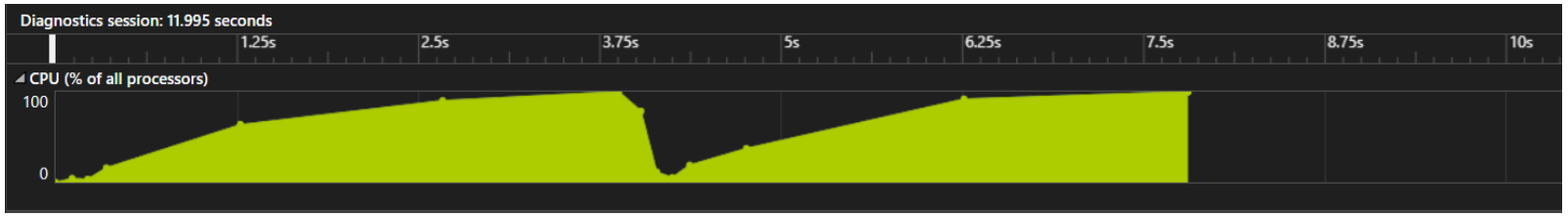
- 8 Thread



- 16 Thread



- 100 Thread



- **# thread ~16** :
  - max CPU Utilization
  - shortest elapsed time
- **# thread >> 16** :
  - CPU Utilization decreases
  - elapsed time increases

CPU utilization retrieved from Visual Studio Profiler

BOTTLENECK

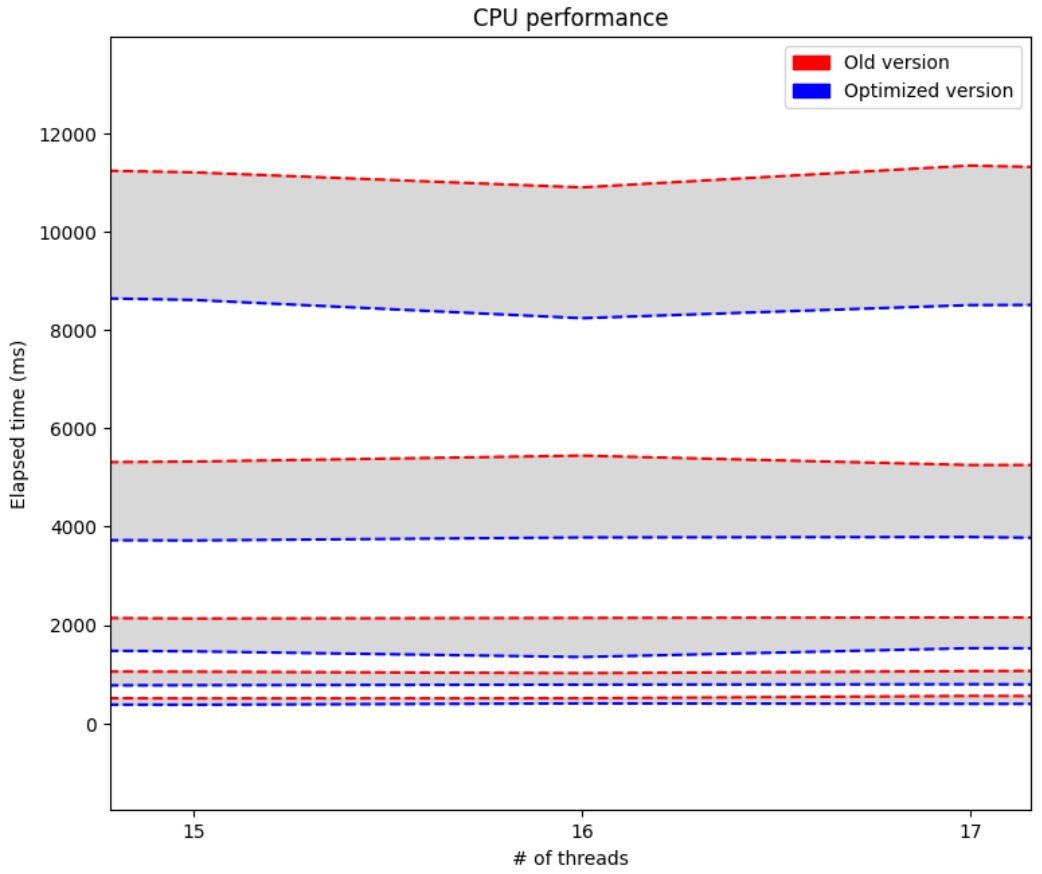
- Miss-rate
- Pipeline stall

Front-End Bound	20.7%
Memory Bound	78.7 %
LLC Miss Count	11,700,637

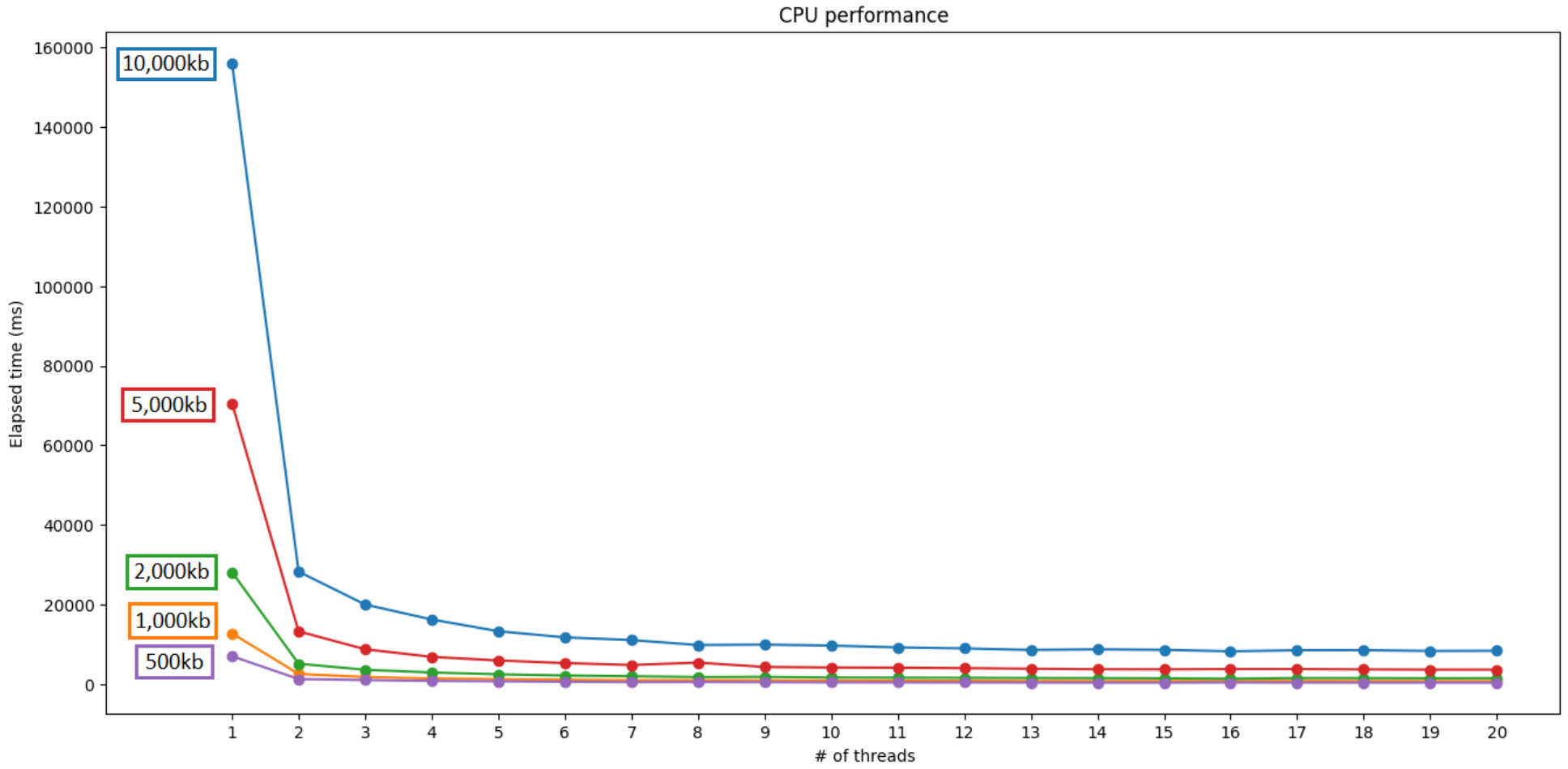
Data retrieved from AMD uProf Profiler

CPU OPTIMIZATION

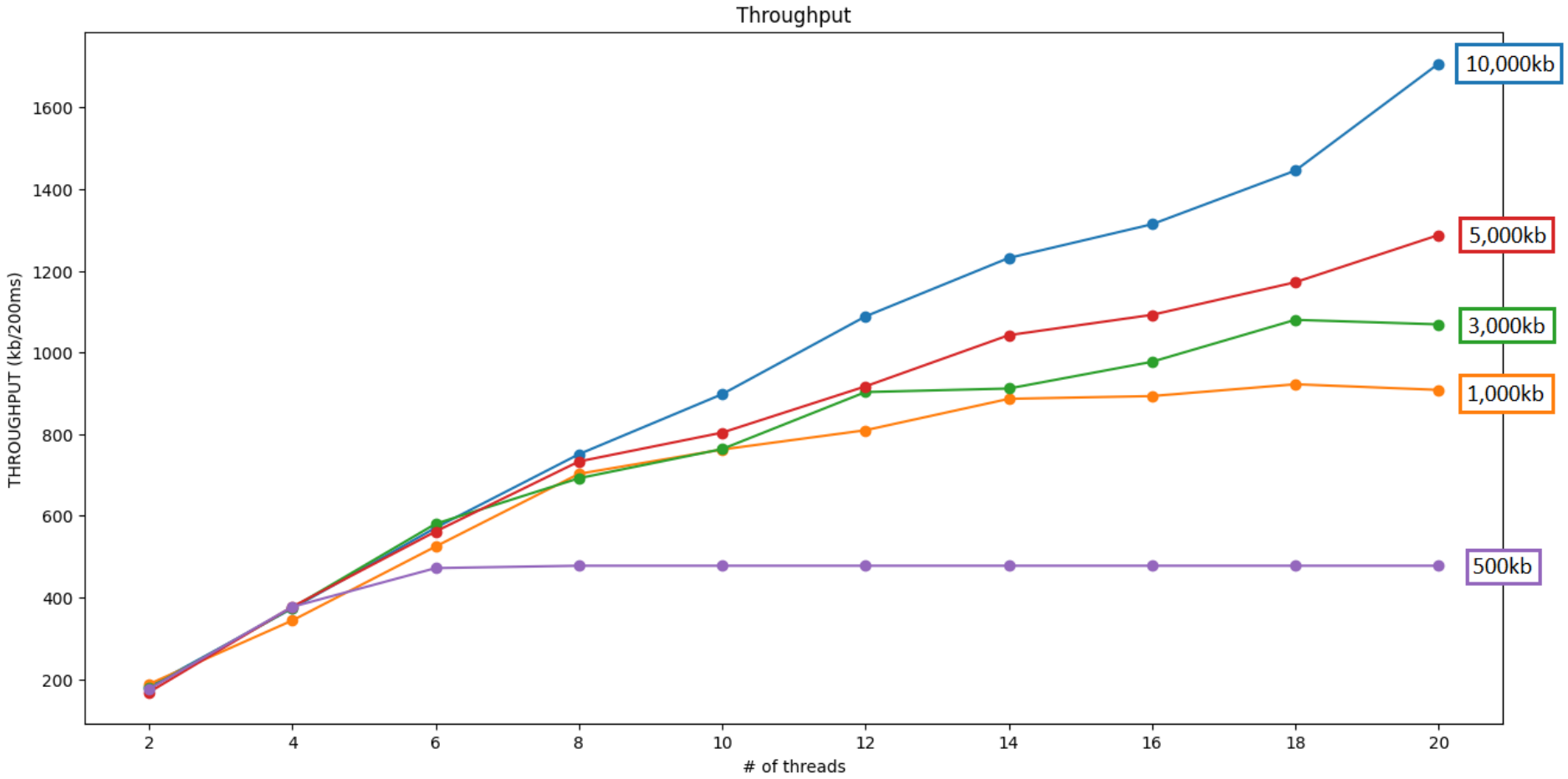
- **GOAL:**
  - Reduce the execution time for both small and big files
- **RESULTS:**
  - Performance improved
  - Execution time reduced of ~35%



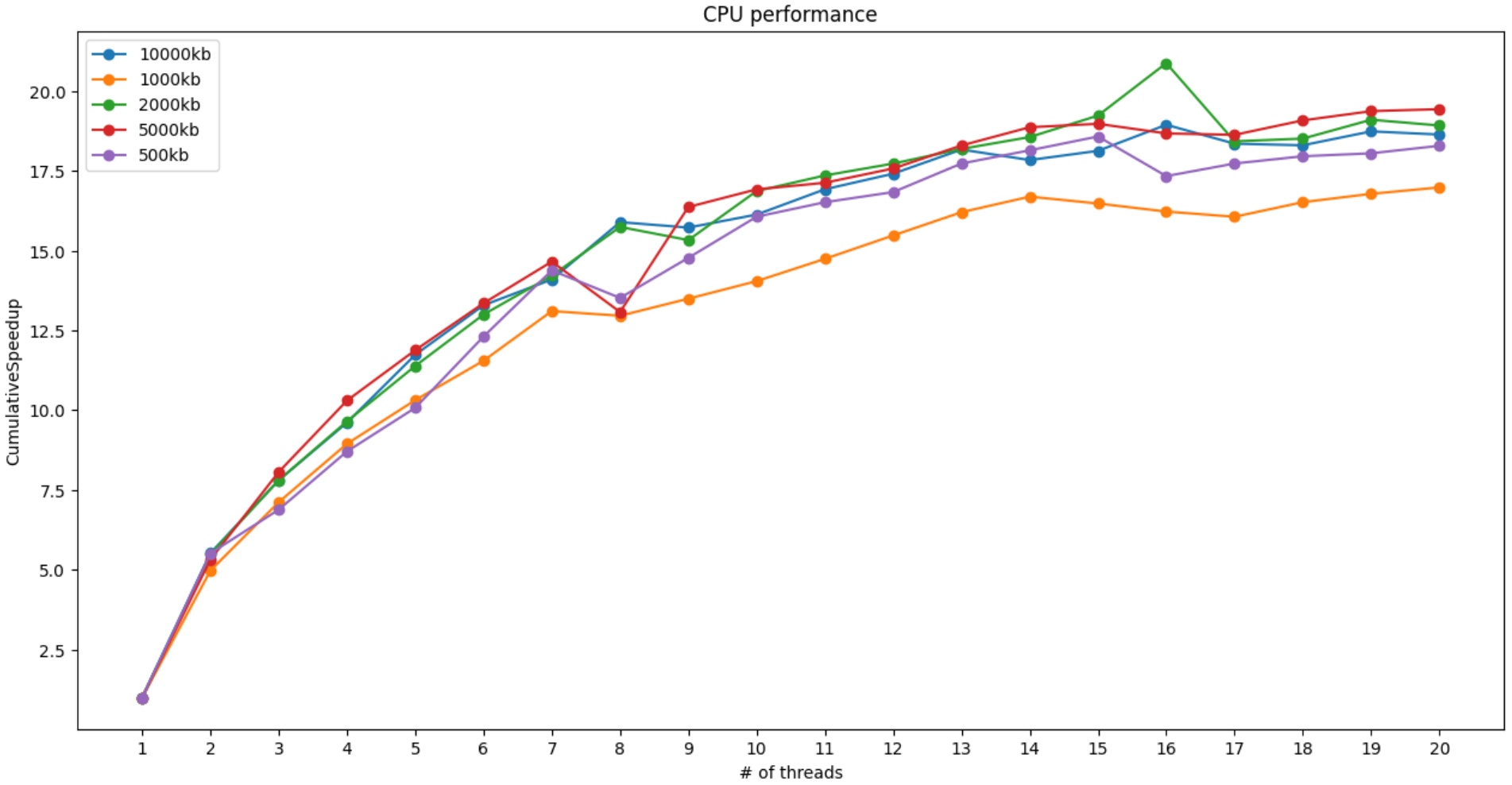
OPTIMIZATION: ELAPSED TIME VS NUMBER OF THREADS



OPTIMIZATION: THROUGHPUT VS NUMBER OF THREADS



OPTIMIZATION: SPEEDUP VS NUMBER OF THREADS



CPU OPTIMIZATION: BOTTLENECK

- Miss-rate
- Pipeline stall

Front-End Bound	24.0%
Memory Bound	73.6 %
LLC Miss Count	6,500,364

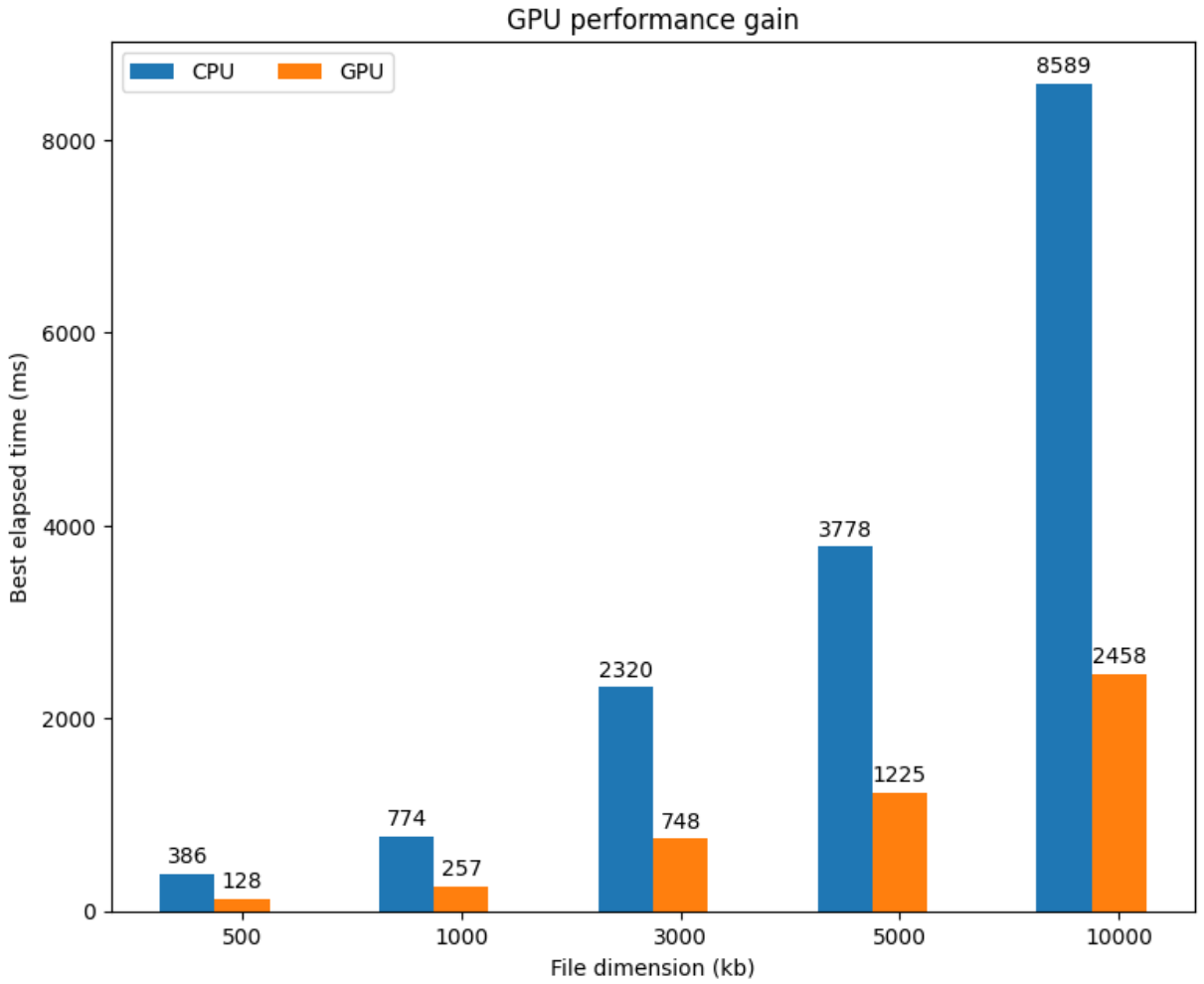
Data retrieved from AMD uProf Profiler

- emplace_back()
- reserve()
- Logical XOR

CPU OPTIMIZATION: IMPLEMENTATION

GPU OPTIMIZATION

- **GOAL:**
  - Reduce the elapsed time of the CPU implementation
- **RESULTS:**
  - Performance improved
  - Elapsed time reduced of ~70%



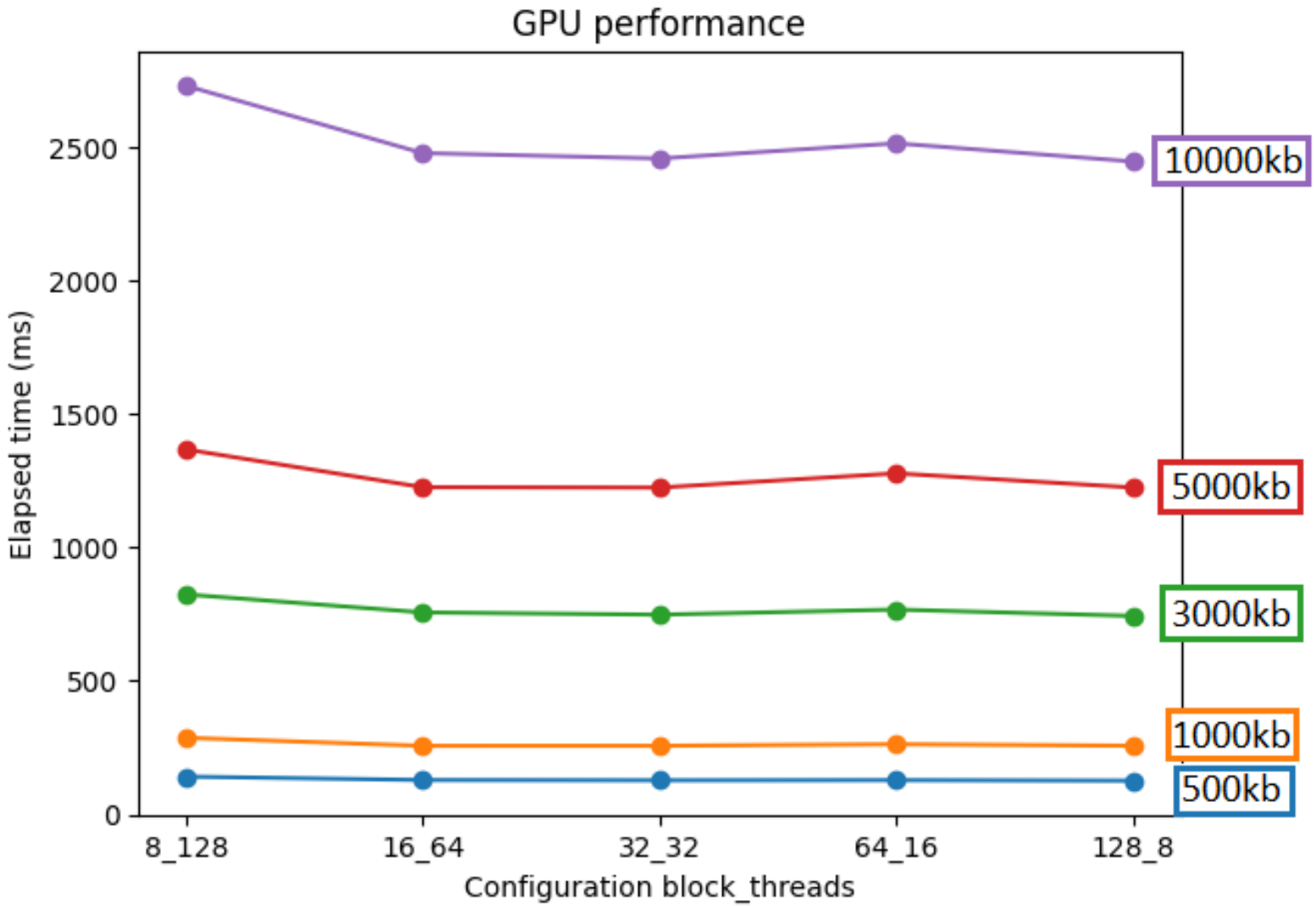
GPU SYSTEM INFORMATION

Global Memory :	4096 Mbytes
# Cuda Cores :	1024 CUDA Cores
GPU Max Clock Rate :	1485 MHz
Memory Clock Rate :	6001 MHz
Memory Bus Width :	128-bit
L2 Cache Size :	1048576 bytes
Warp Size :	32
# Threads / Multiprocessor :	1024
# Threads / Block :	1024
Max size of Thread Block (x, y, z) :	(1024, 1024, 64)
Max size of Grid Size (x, y, z) :	(2147483647, 65535, 65535)

Data retrieved from CUDA Device Query (Runtime API)

ELAPSED TIME VS GRID CONFIGURATION

- **Mean Elapsed Time** computed with 30 runs per configuration
- Overall **performance improvement**



GPU OPTIMIZATION: BOTTLENECK

- Memory-Bound Kernel
- Warp stall: high memory access time reduces warp utilization
- "More blocks" >> "Bigger blocks"

File Size	Compute Throughput (%)	Memory Throughput (%)
500 Kb	5.43	75.82
1000 Kb	5.48	77.90
3000 Kb	5.50	78.45
5000 Kb	5.57	79.99
10000 Kb	5.57	80.49

Data retrieved from NVIDIA Nsight Compute

Shared memory

Distributed workload

GPU OPTIMIZATION: IMPLEMENTATION

CONCLUSIONS

- Big improvement for DES algorithm (with parallel implementation)
- ~35% faster with CPU implementation and ~70% faster with GPU implementation (considering elapsed time)
- Maximized utilization (for CPU configuration)
- Overall performance improvement with GPU optimization