



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE

# Sequential and parallel algorithm implementations

Andrea Moscatelli

Course:

**Parallel computing**

Teacher:

**Marco Bertini**

# Outline

- **SAD in JAVA**
  - Sequential/parallel implementations
  - Performance analysis
- **Bigram/Trigram detection with CUDA**
  - CPU/GPU implementations
  - Performance analysis

# Project 1

SAD in JAVA

# SAD Technique

**SAD = Sum of Absolute Differences**

It's a very simple *pattern recognition* technique based on algebraic differences between *pattern* and *dataset*.

pattern

2	1	4
---	---	---

dataset

6	3	2	1	5	7
---	---	---	---	---	---

# SAD Technique

**SAD = Sum of Absolute Differences**

It's a very simple *pattern recognition* technique based on algebraic differences between *pattern* and *dataset*.

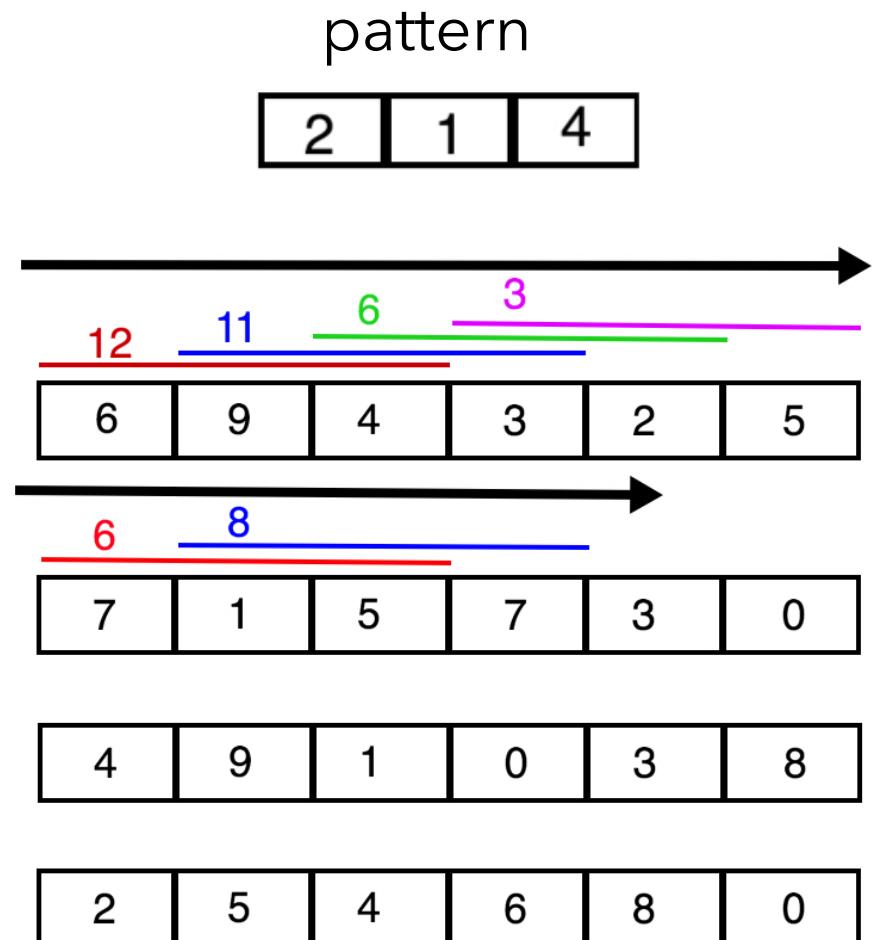
pattern
2 1 4

dataset
6 3 2 1 5 7
1° 4+2+2 = 8
2° 1+1+3 = 5
3° 0+0+1 = 1
4° 1+4+3 = 8

# Java implementation

Sequential:

```
sequentialMethod(){  
    return findBestSAD(0, datasetHeight);  
}  
  
findBestSAD (startIndex, endIndex) {  
    bestSAD = MAX_VALUE;  
    foreach i in [startIndex, endIndex]{  
        foreach j in [0, datasetLength]{  
            sad = calculateSAD(i, j);  
            if (sad < bestSAD)  
                bestSAD = sad;  
        }  
    }  
    return bestSAD;  
}
```

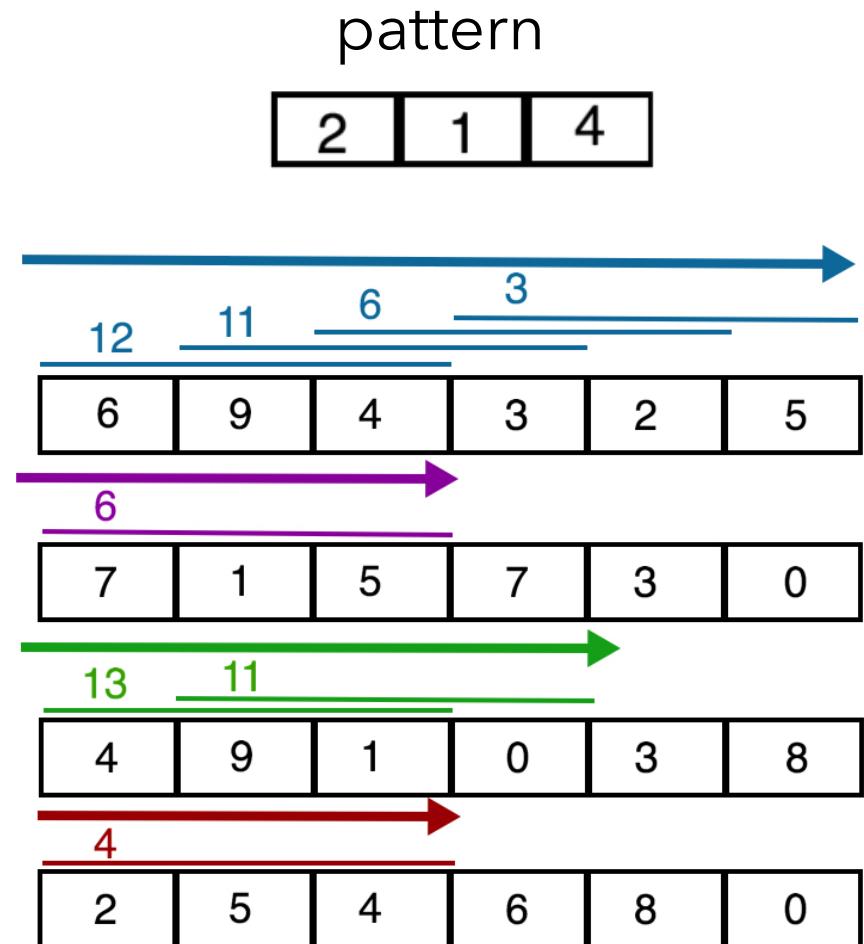


# Java implementation

## Parallel:

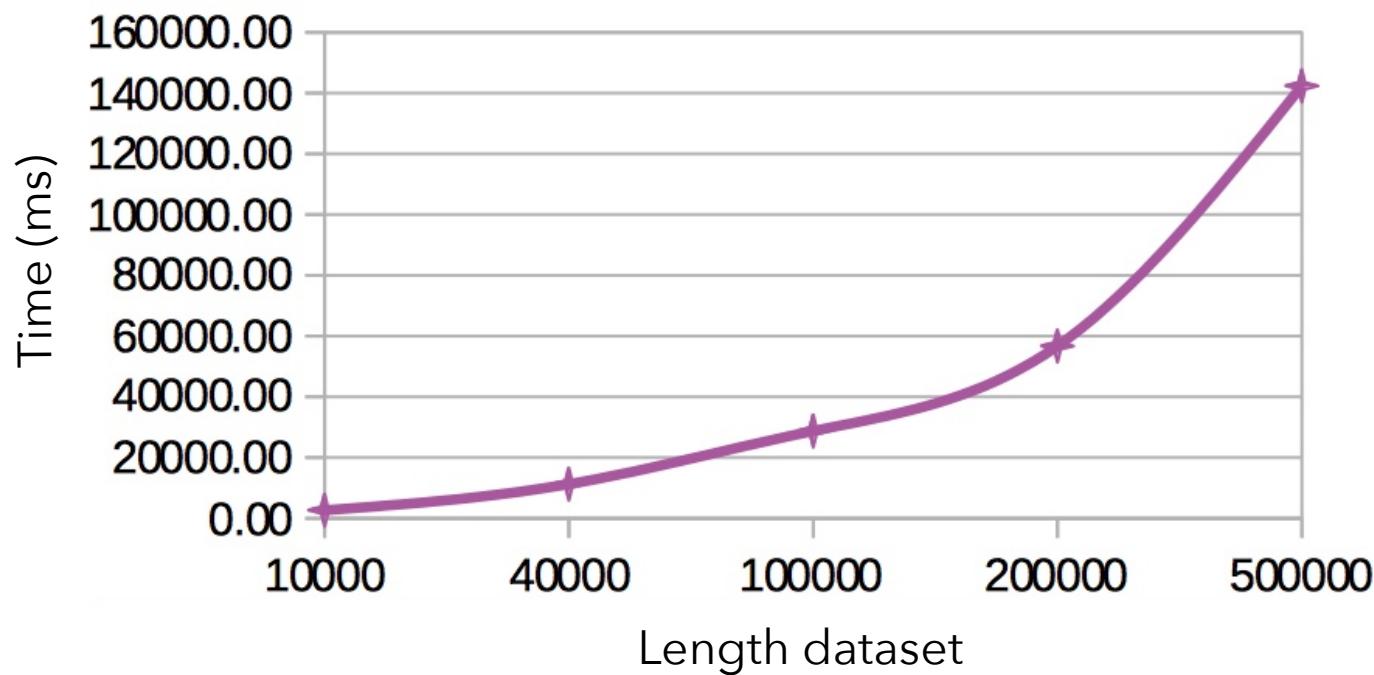
```
parallelMethod(){
    slice = datasetLength / numberOfThreads;
    bestSAD = MAX_VALUE;
    executor = new ExecutorFixedThreadPool(numberOfThreads);
    for i from 0 to n-1{
        executor.add(new Thread(
            sad = findBestSAD(i * slice, (i+1) * slice);
            bestSAD = min(sad, bestSAD);
        ));
    }
    executor.runAll();
    return bestSAD;
}

findBestSAD (startIndex, endIndex) {
    bestSAD = MAX_VALUE;
    foreach i in [startIndex, endIndex]{
        foreach j in [0, datasetLength]{
            sad = calculateSAD(i, j);
            if (sad < bestSAD)
                bestSAD = sad;
        }
    }
    return bestSAD;
}
```



# Performance

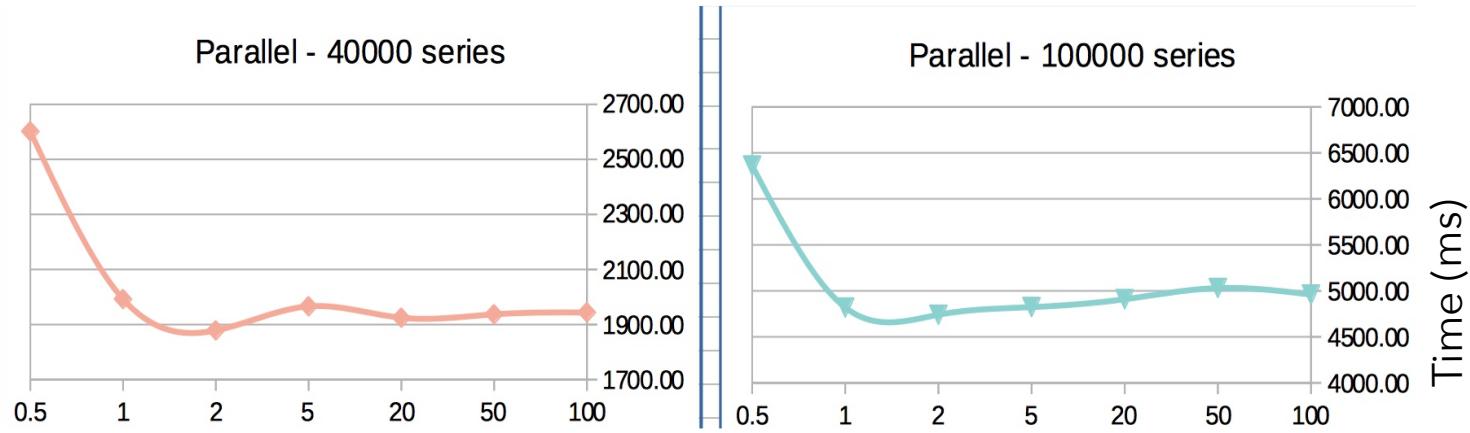
Seqential:



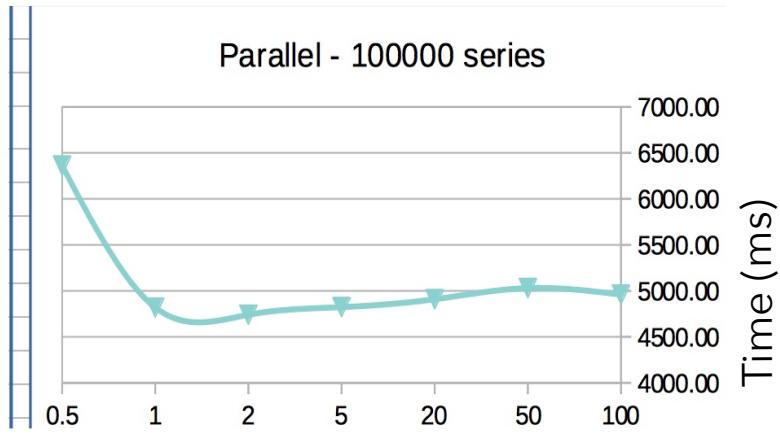
# Performance

Parallel:

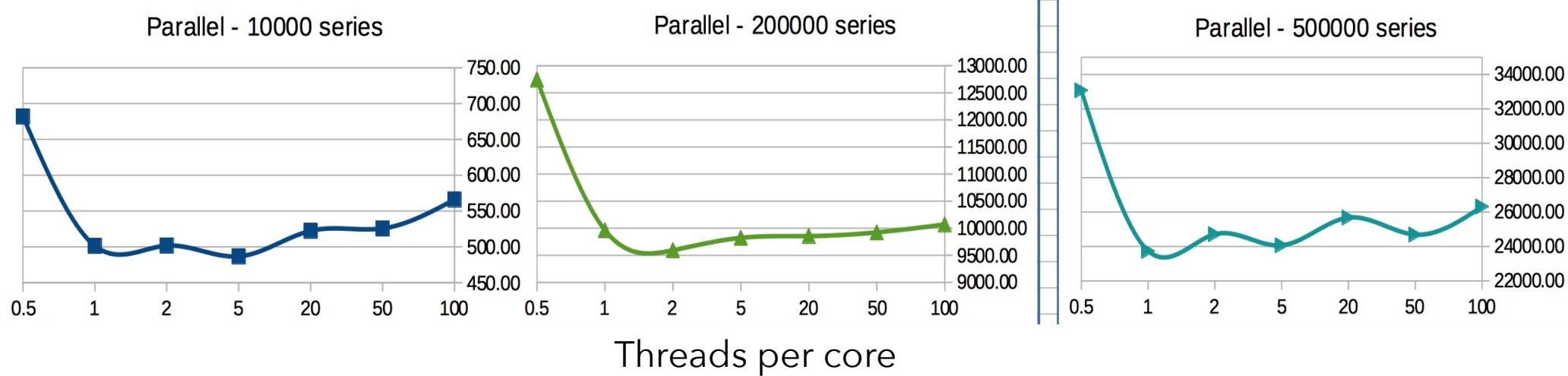
Parallel - 40000 series



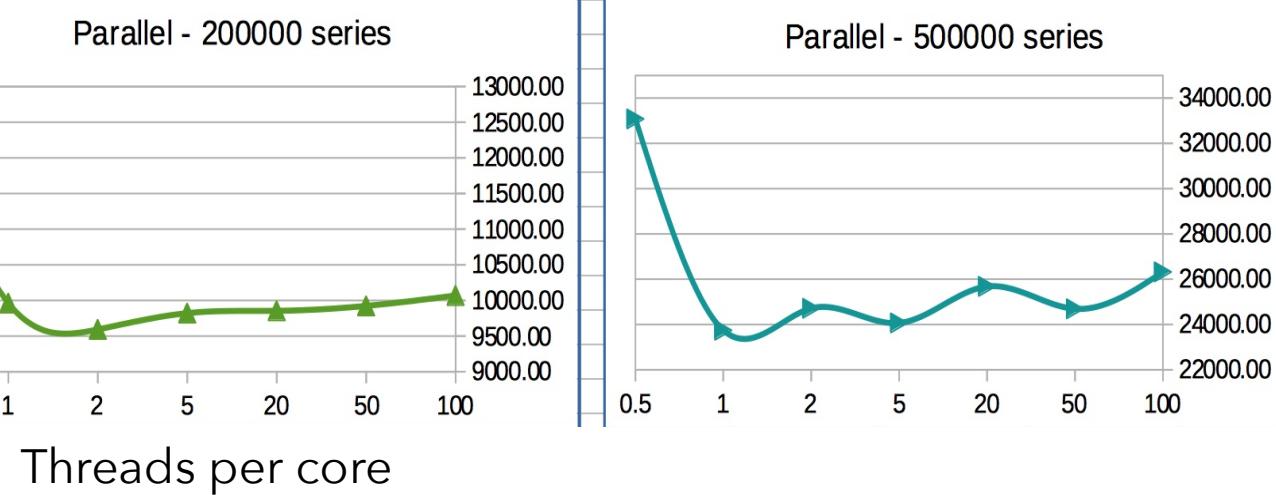
Parallel - 100000 series



Parallel - 10000 series



Parallel - 200000 series



Threads per core

# Project 2

## Bigram and Trigram in CUDA

# Bigrams and Trigrams

What's a bigram (or trigram)?

It's a couple (or triplet) of adjacent letters or word in a text.

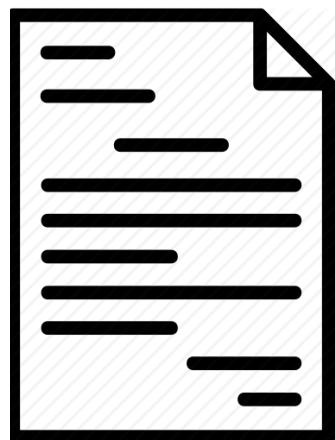
For instance:

“aabb cabb”

<b>Bigram</b>	<b>Occurrences</b>	<b>Trigrams</b>	<b>Occurrences</b>
aa	1	aab	1
ab	2	abb	2
bb	2	bbc	1
bc	1	bca	1
ca	1	cab	1

# CPU Implementation

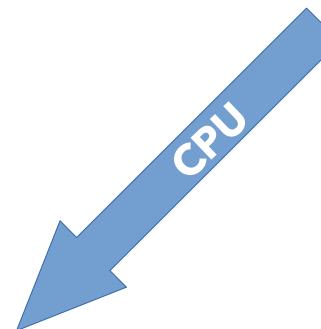
The version with the CPU scan the input text file, character by character in a sequential manner populating a map of results.



vecenturiesbutalsotheleapint  
oelectronicitypesettingremaini  
ngessentiallyunchangeditwas  
pop...

## MAP:

- AAA → 3
- AAB → 2
- AAR → 4
- BCE → 2
- BDF → 3
- ...

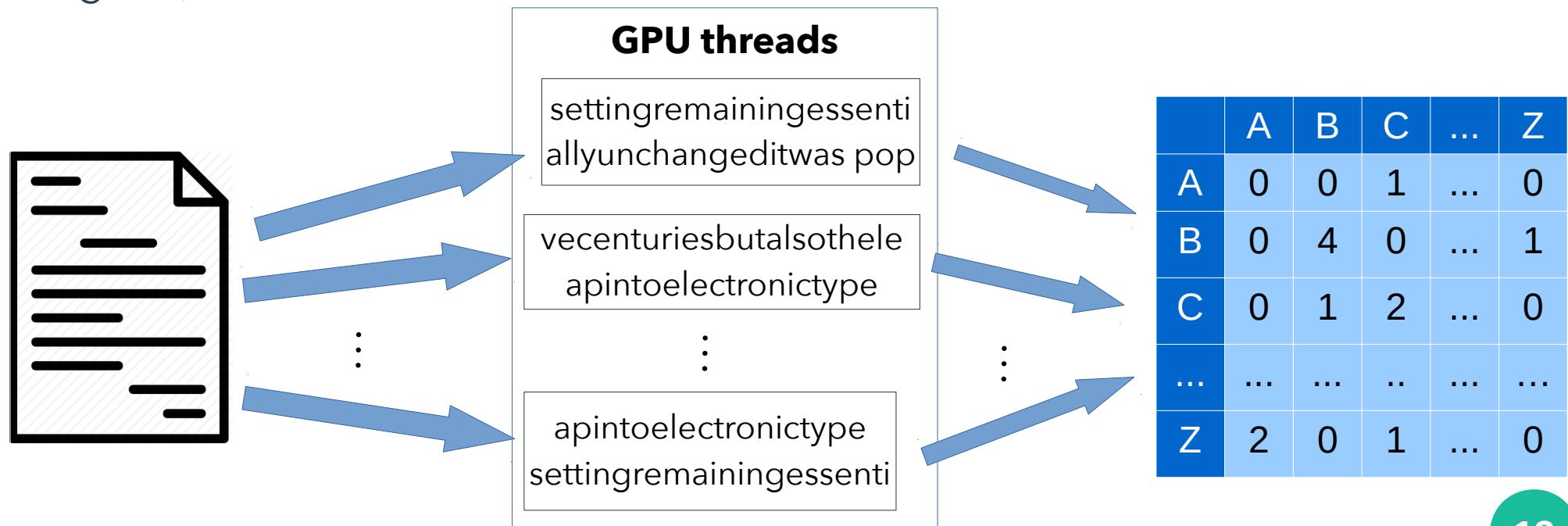


# GPU Implementation

The version with the GPU in CUDA scan the input text file, splitting it in N parts and giving them to N threads where:

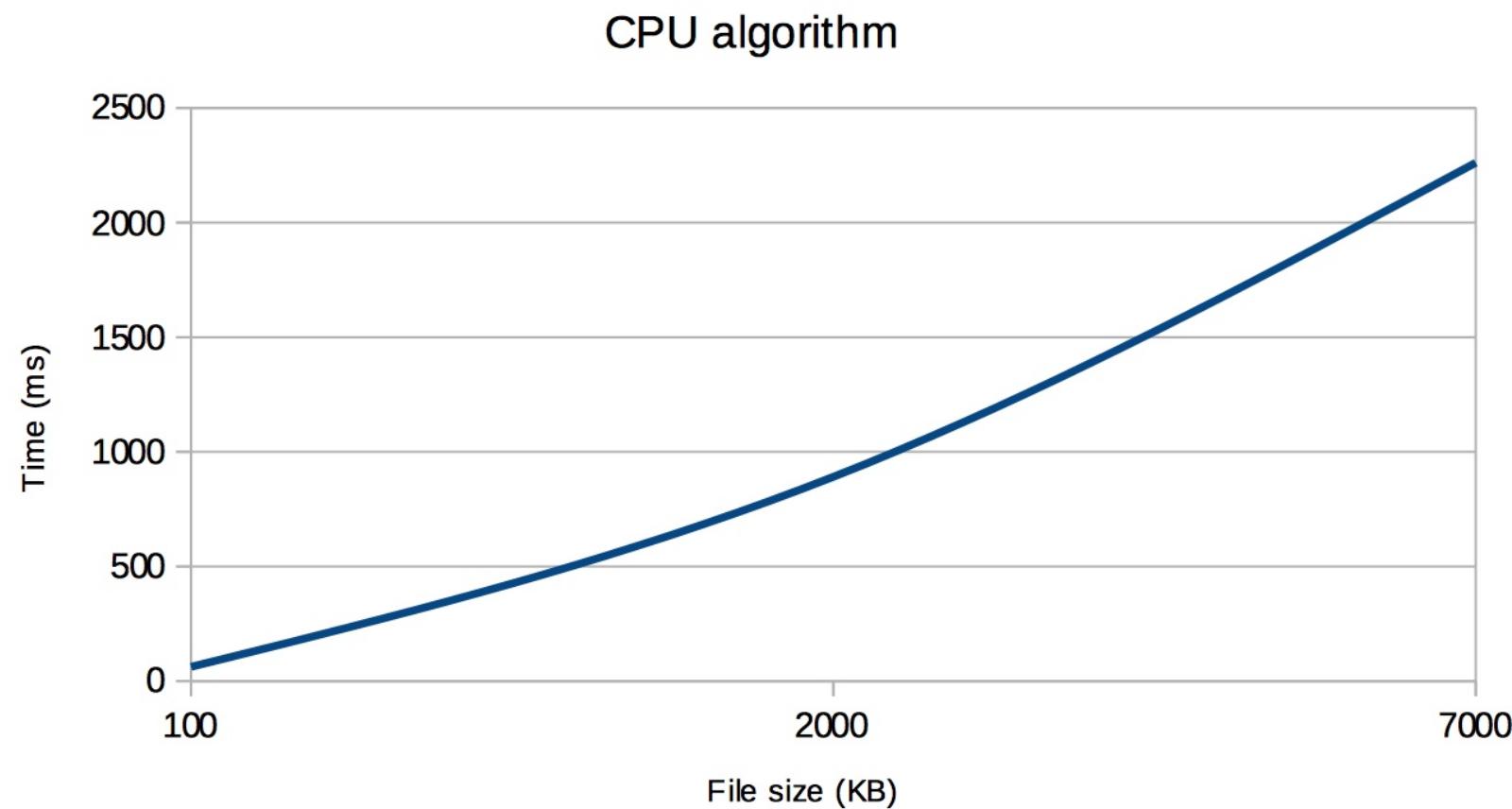
$$N = \text{GRID DIMENSION} \times \text{BLOCK DIMENSION}.$$

Each thread scan the file and populate a global matrix (or cube, in case of trigram) of occurrences.



# Performance

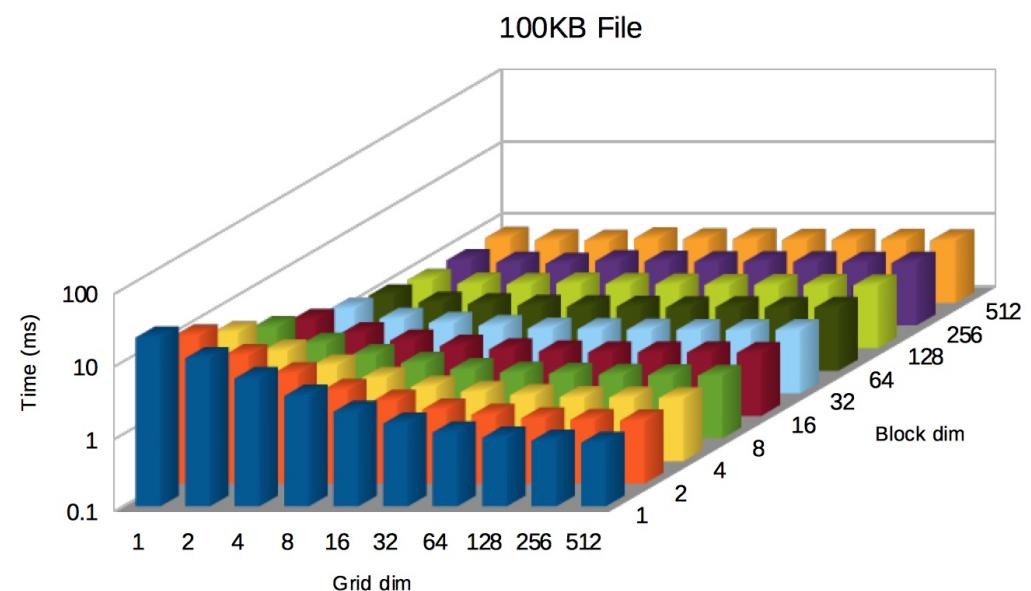
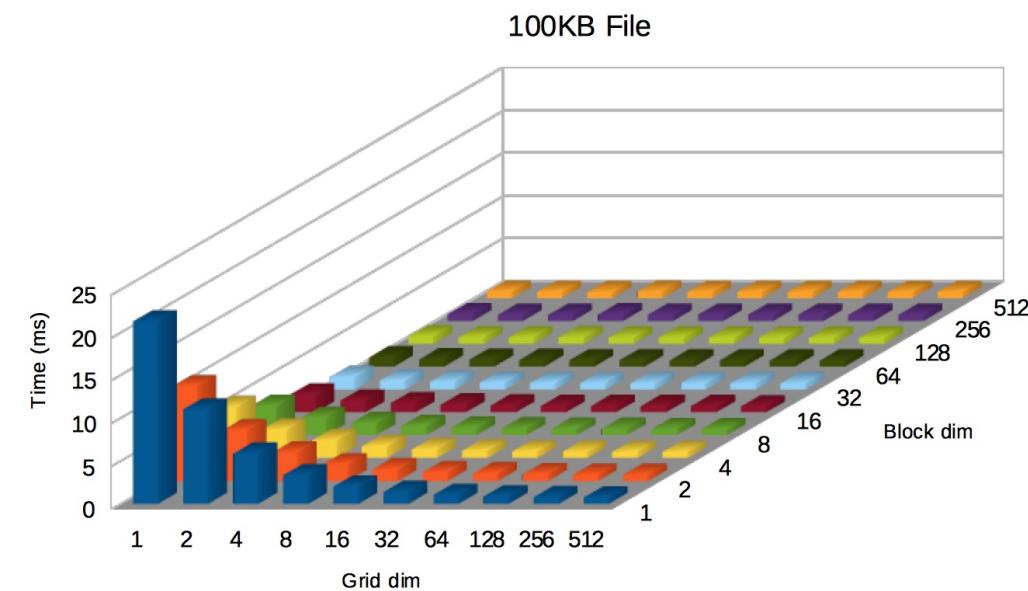
Results of the algorithm varying the size of the text, grid and blocks.



# Performance

Results of the algorithm varying the size of the text, grid and blocks.

**GPU:**

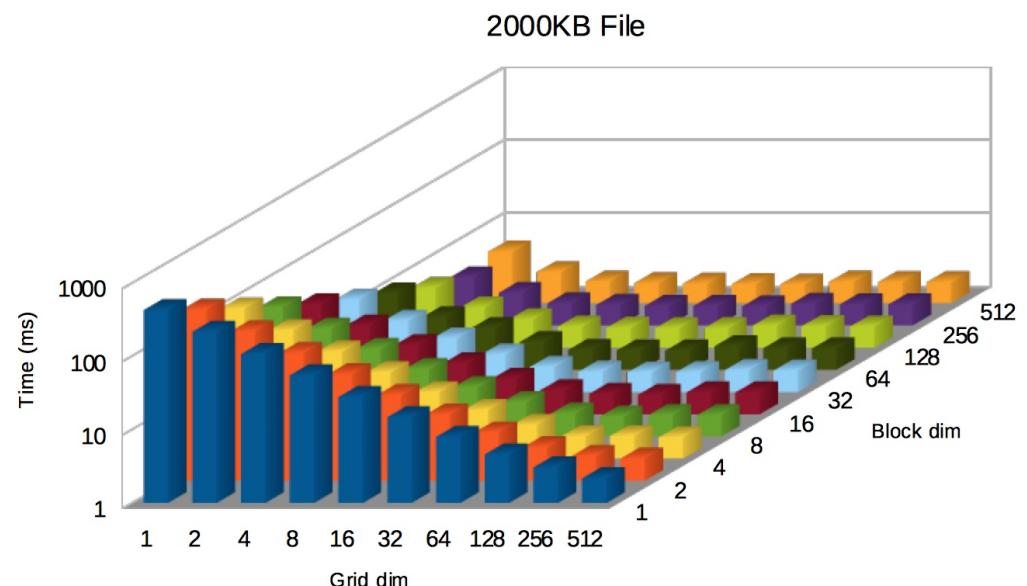
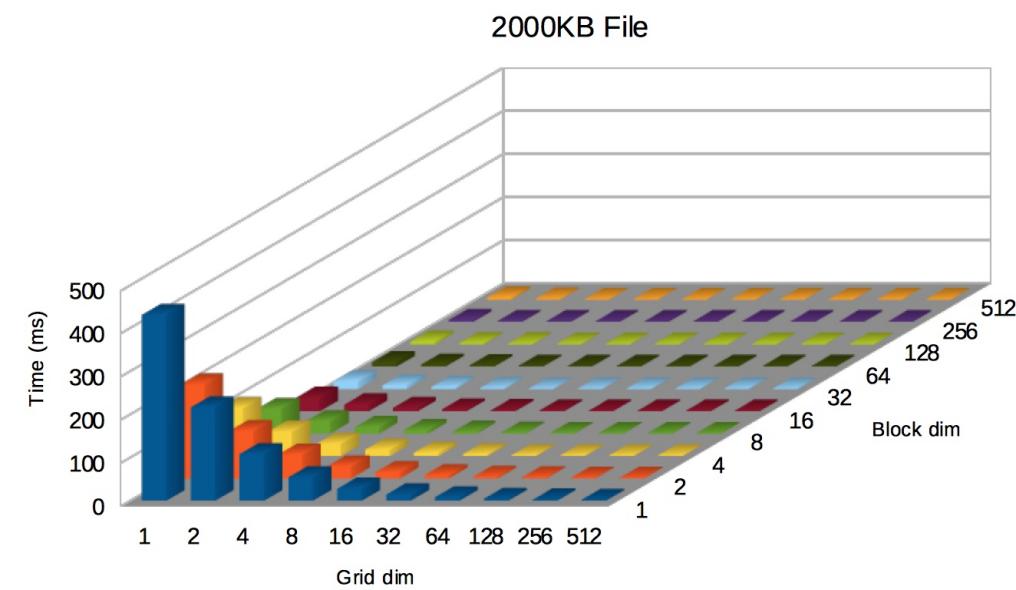


first access:  $\approx 750$ ms

# Performance

Results of the algorithm varying the size of the text, grid and blocks.

**GPU:**

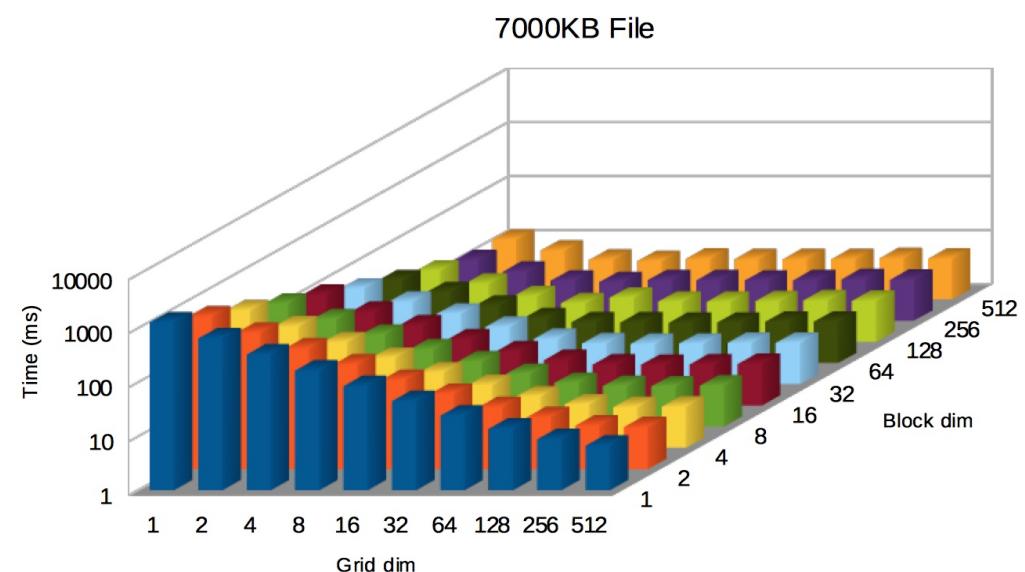
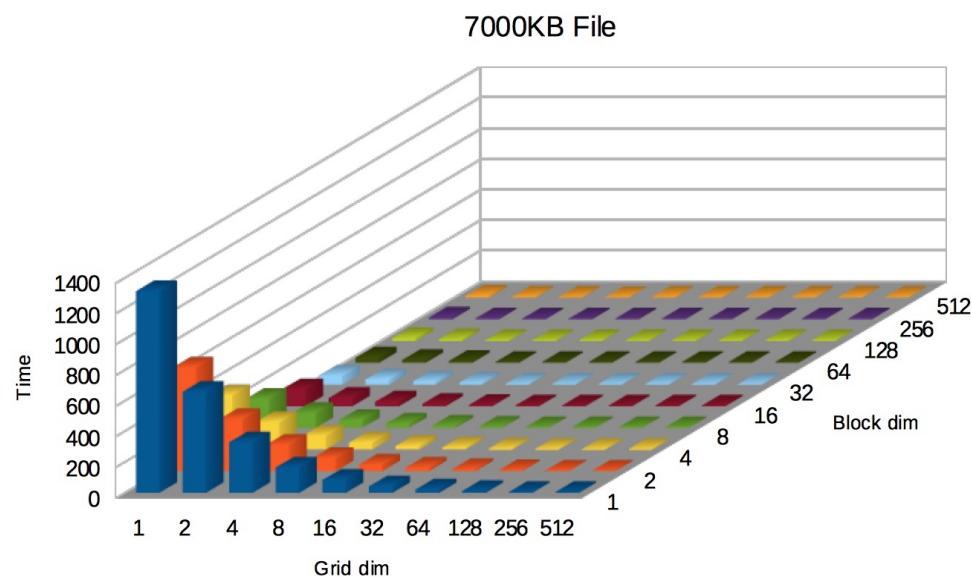


first access:  $\approx 750$ ms

# Performance

Results of the algorithm varying the size of the text, grid and blocks.

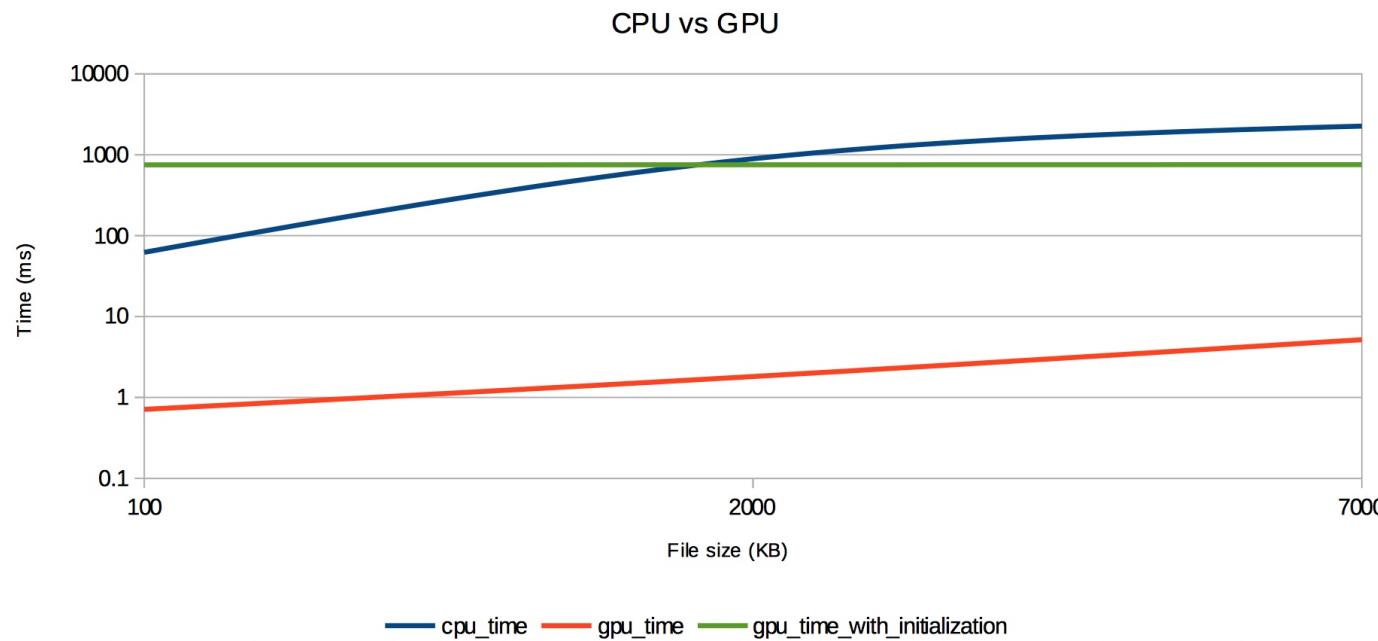
**GPU:**



first access:  $\approx 750\text{ms}$

# Conclusion

- Ignoring the first access time, GPU is always better than CPU.
- The best parallel result is around block size **512** and grid **4-8** but after a certain value the gain it's almost insignificant (for small file).
- Considering the first access time, it seems that the CPU is better till ~1800KB, after this value GPU is always better.



Thank you!