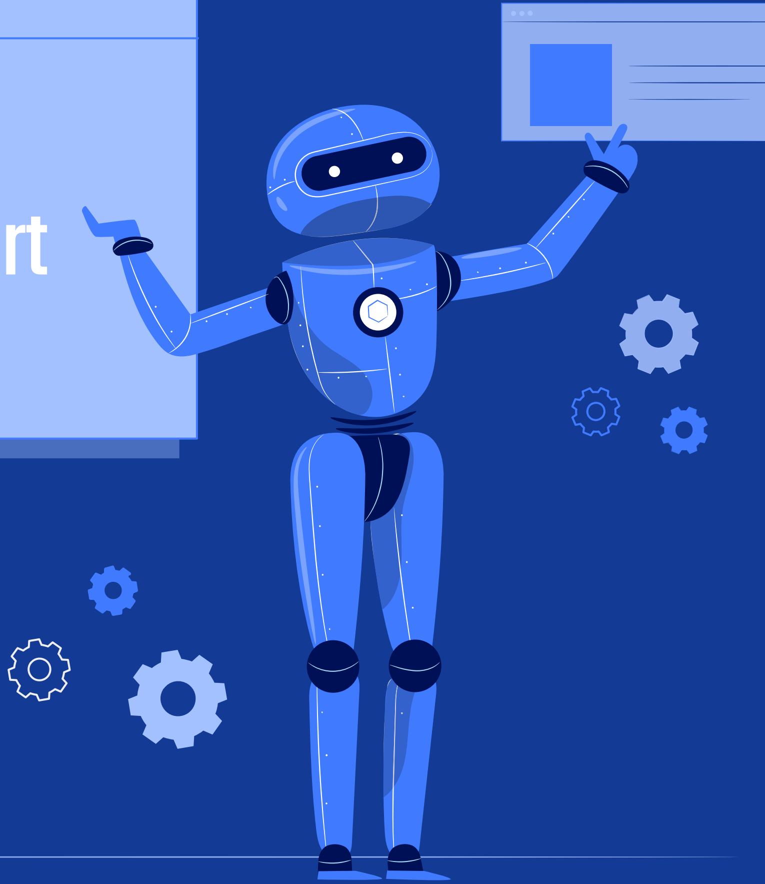
HPC project:

Batch merge and merge path sort

Astrid Legay - Marco Naguib - MAIN5



Summary

01 Review of the subject

02 Step 1 : mergeSmall_k

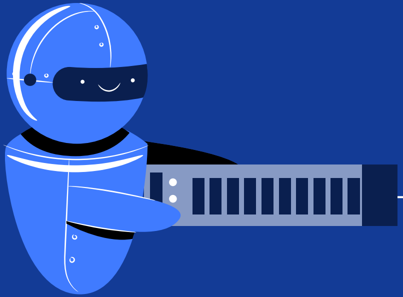
03 Step 2 : pathBig_k mergeBig_k

04 Step 3 : merge_sort

05 Step 5 : mergeSmallBatch_k

06 Applications

01 Review of the subject



Tri de tableaux

APPLICATIONS

Database
Image processing
Graph theory

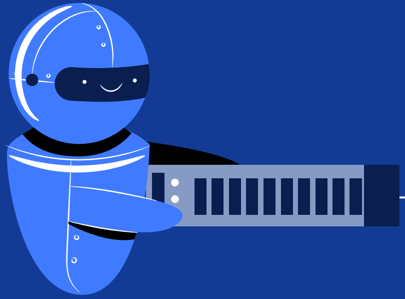
ALGORITHMS

Bubble sort
Quick sort
Merge sort

MERGESORT

Highly parallelizable thanks to divide and conquer

02 Step 1 : mergeSmall_k



Step 1

For $|A| + |B| \leq 1024$, write a kernel mergeSmall k that merges A and B using only one block of threads.

Process

01

Wrinting the algortihm A and B (sequentially)

02

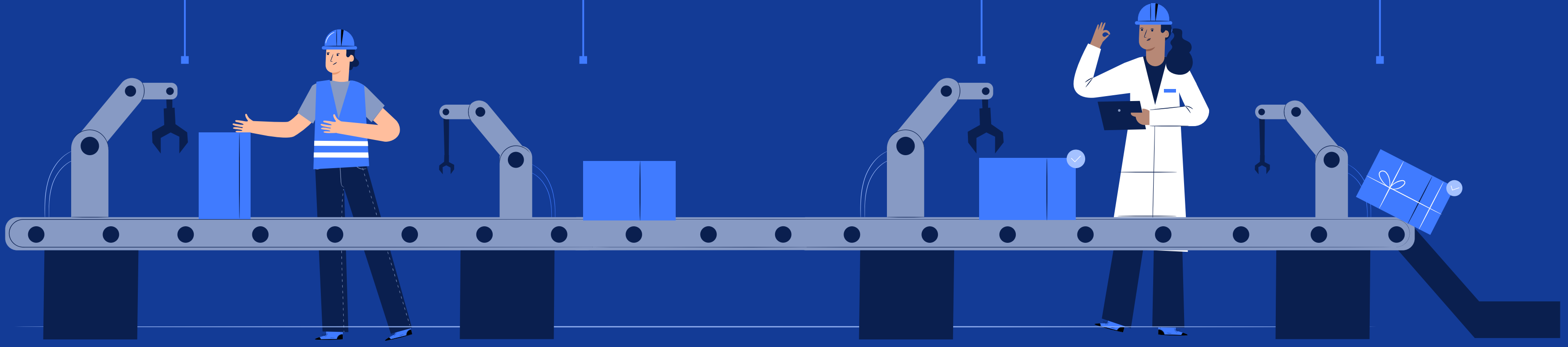
Using the functions for testing and printing

03

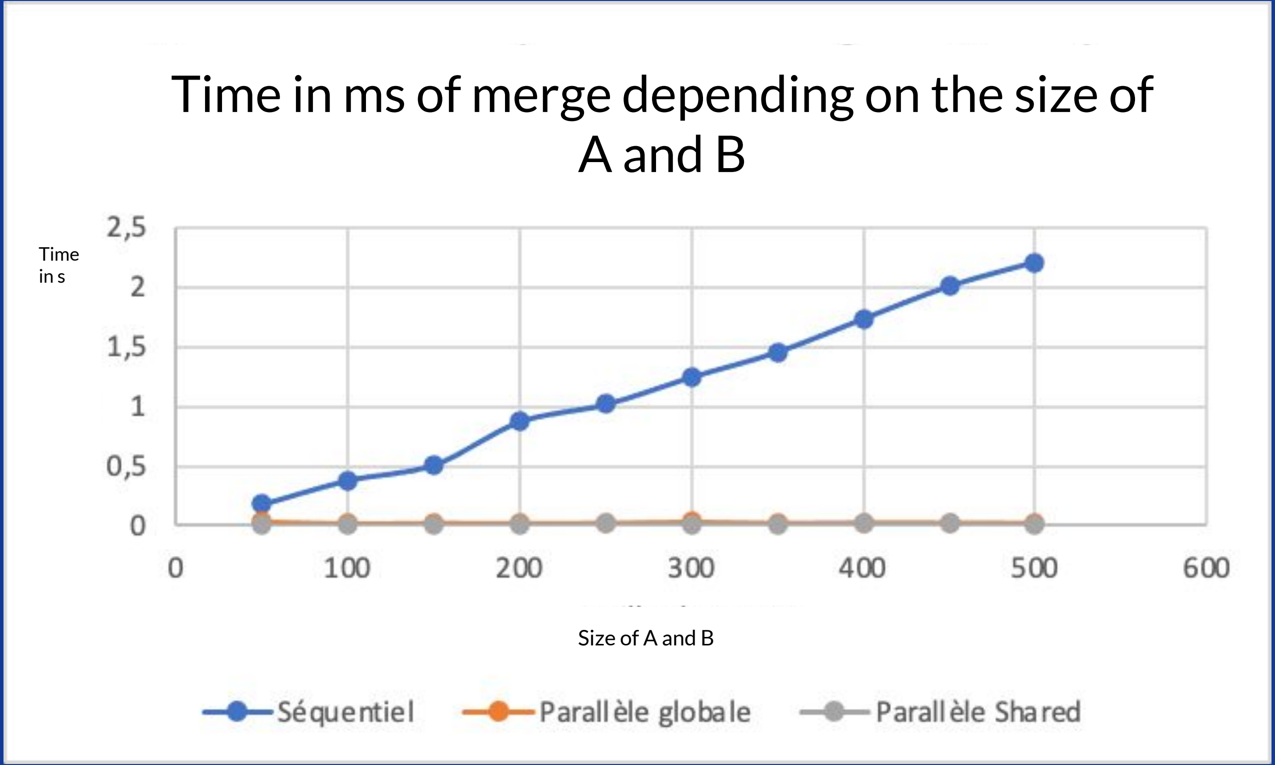
Writing MergeSmall_k on global and shared memory using Cuda

04

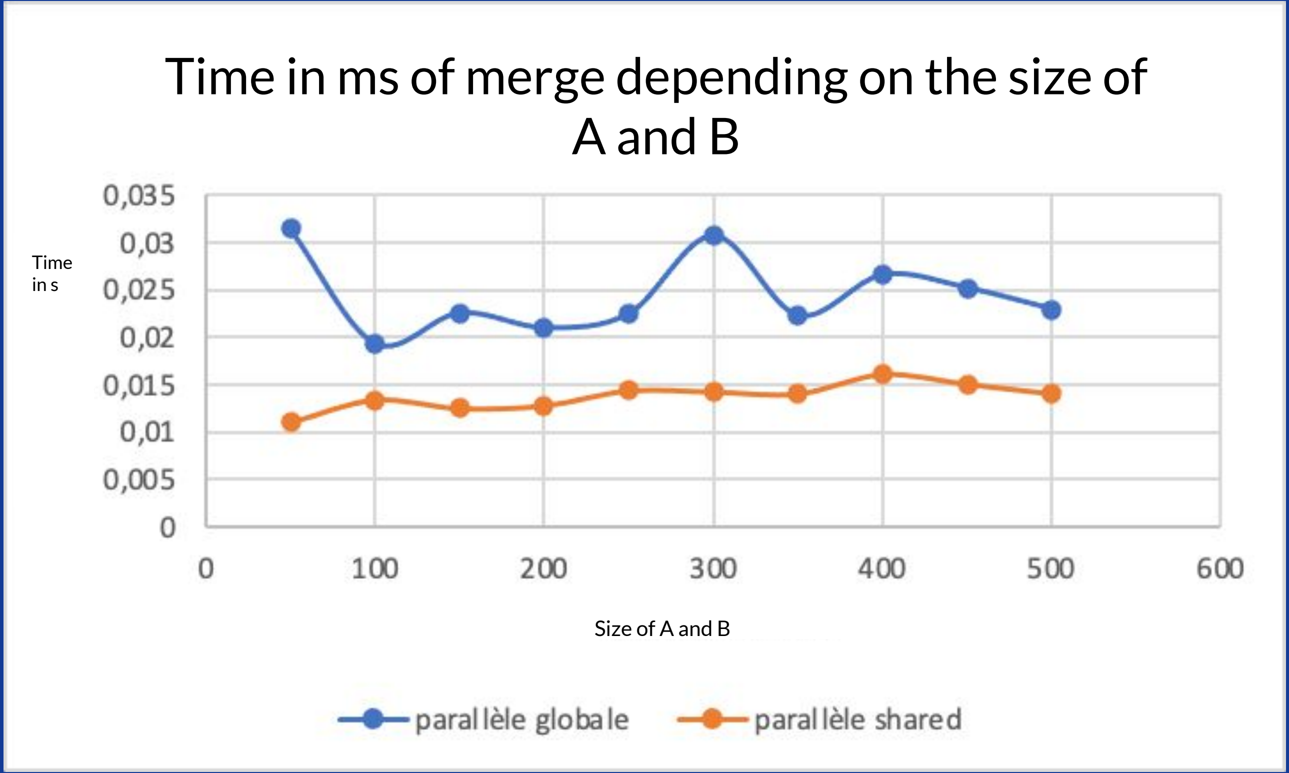
Tests and measurements



Results

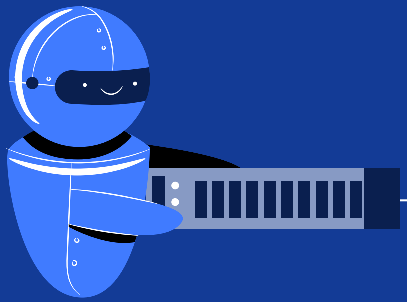

Time in ms of merge depending on the size of A and B
Time in s
2,5
2
1,5
1
0,5
0
0
100
200
300
400
500
600
Size of A and B
Séquentiel
Parallèle globale
Parallèle Shared

Global and shared memory comparison



03 Step 2 :

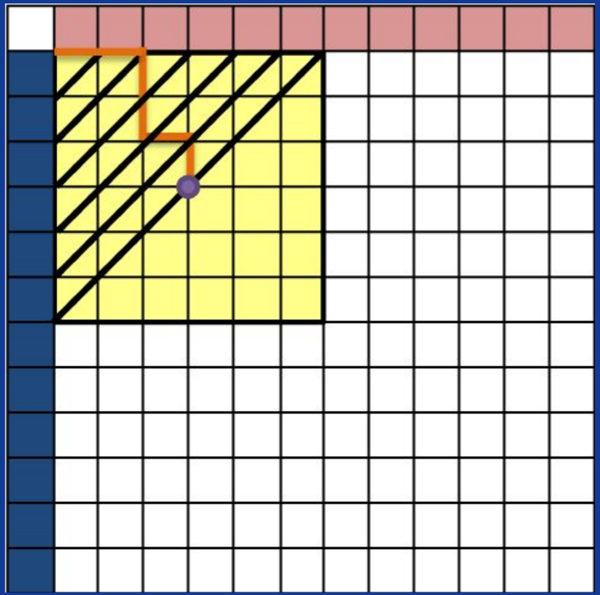
pathBig_k et mergeBig_k

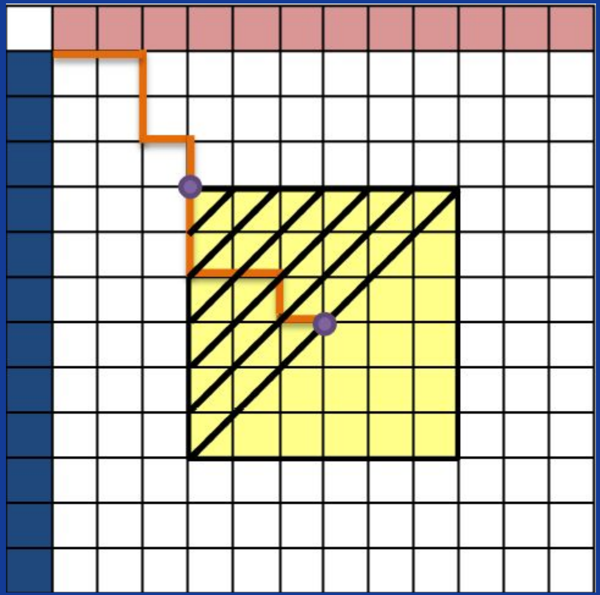


Step 2

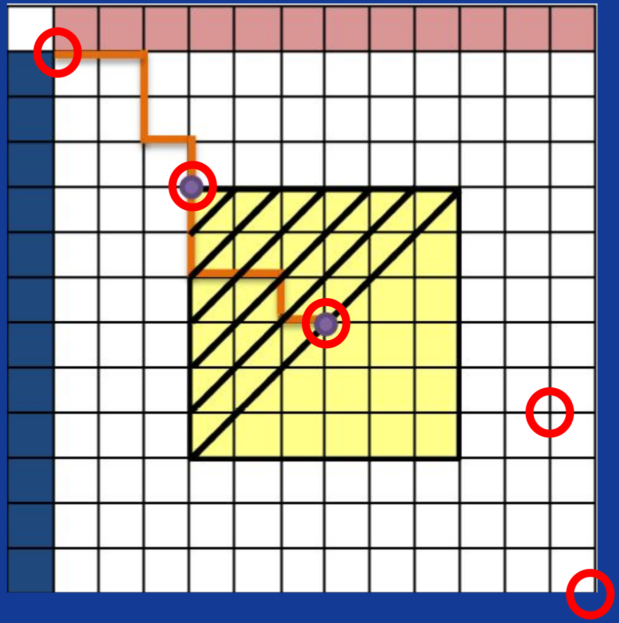
For any size ¦A¦+¦B¦ = d sufficiently smaller than the global memory, write two kernels that merge A and B using various blocks: The first kernel pathBig k finds the merge path and the second one mergeBig k merges A and B.

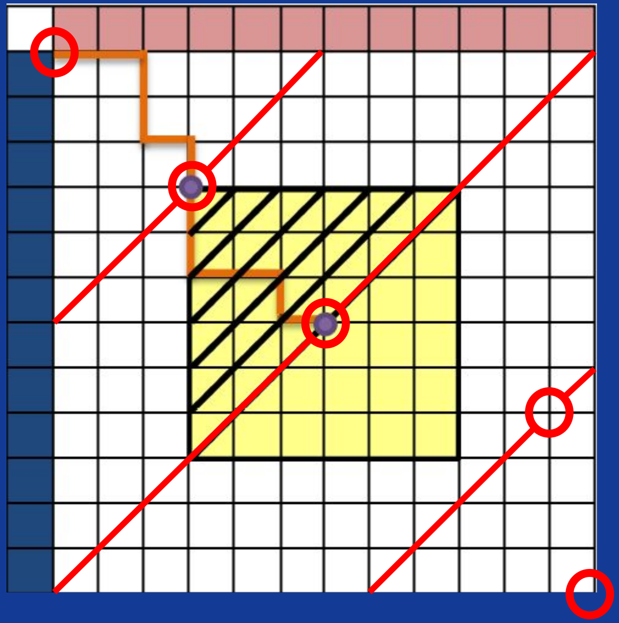
Use of sliding windows





Use of sliding windows





Process

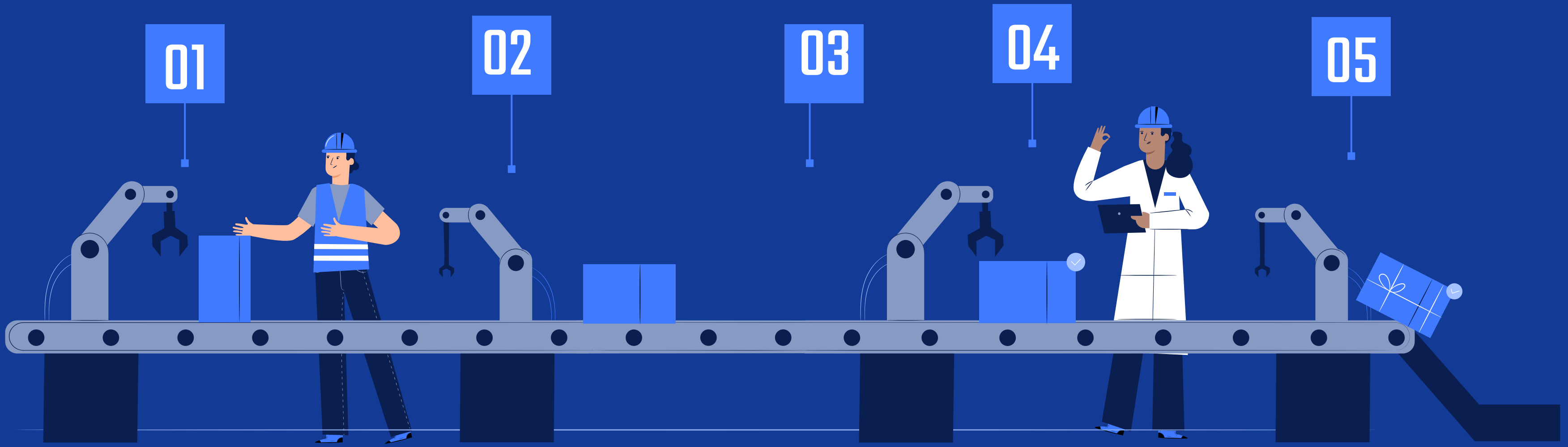
Using testing functions of question 1

Pathbig : finds the intersection between the way and the diagonal

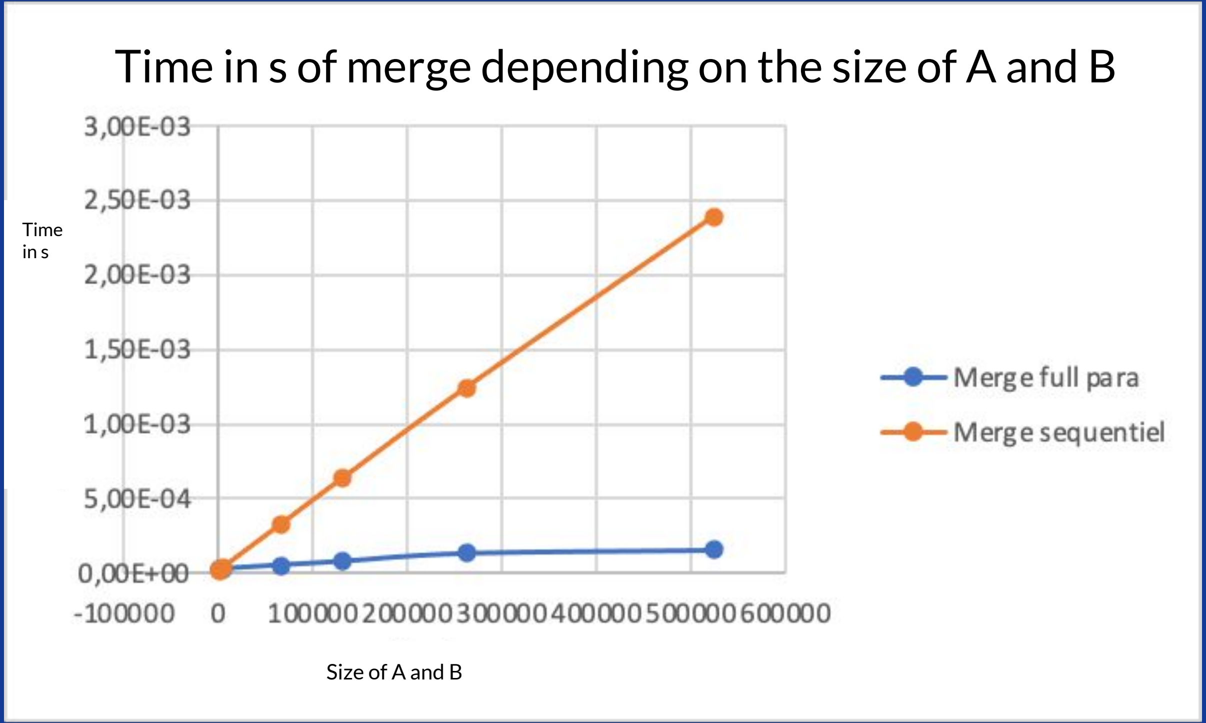
Mergebig : merge the sliding windows 1 by 1

Mergebig : merge the slinding windows in parallel

Tests and measurements



Results

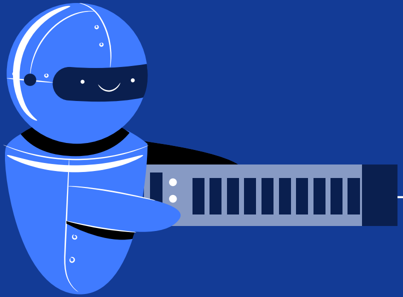

Time in s of merge depending on the size of A and B
Time in s
3,00E-03
2,50E-03
2,00E-03
1,50E-03
1,00E-03
5,00E-04
0,00E+00
-100000
0
100000
200000
300000
400000
500000
600000
Size of A and B
Merge full para
Merge sequentiel

POLYTECH SORBONNE

SORBONNE UNIVERSITÉ

04 Step 3 :

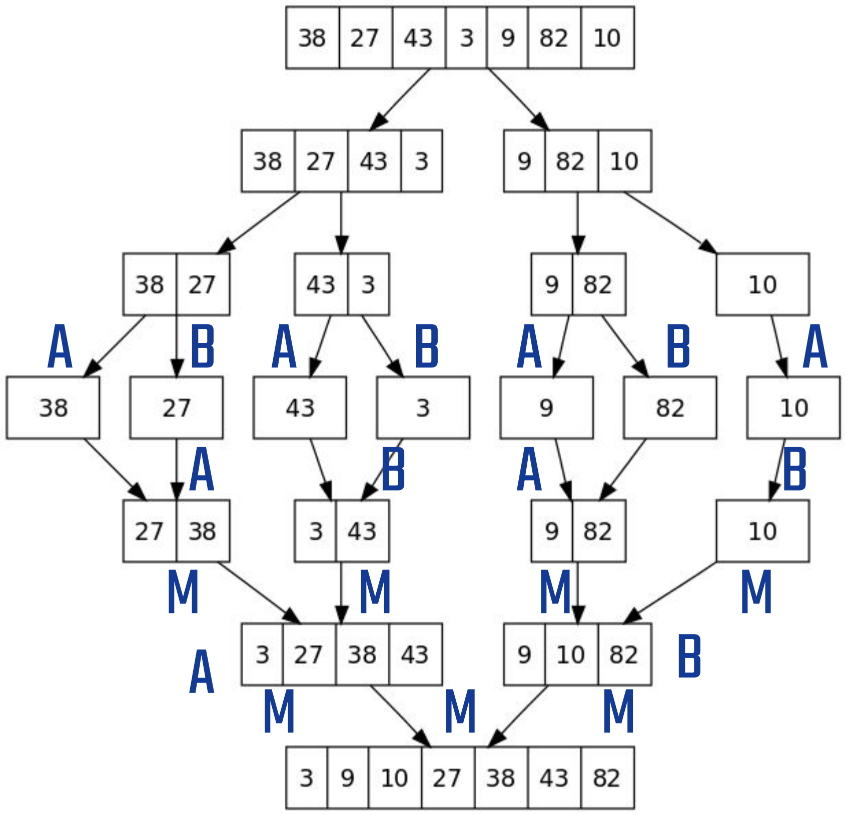
mergesort



Step 3

Looping on appropriate calls of pathBig k and of mergeBig k, write a function that sorts any array M of size d sufficiently smaller than the global memory. Give the execution time with respect to d.

Merge Sort



38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
A
B
A
B
A
B
A
38
27
43
3
9
82
10
A
B
A
B
27 38
3 43
9 82
10
M
M
M
M
3 27 38 43
9 10 82
A
B
M
M
M
3 9 10 27 38 43 82

Process

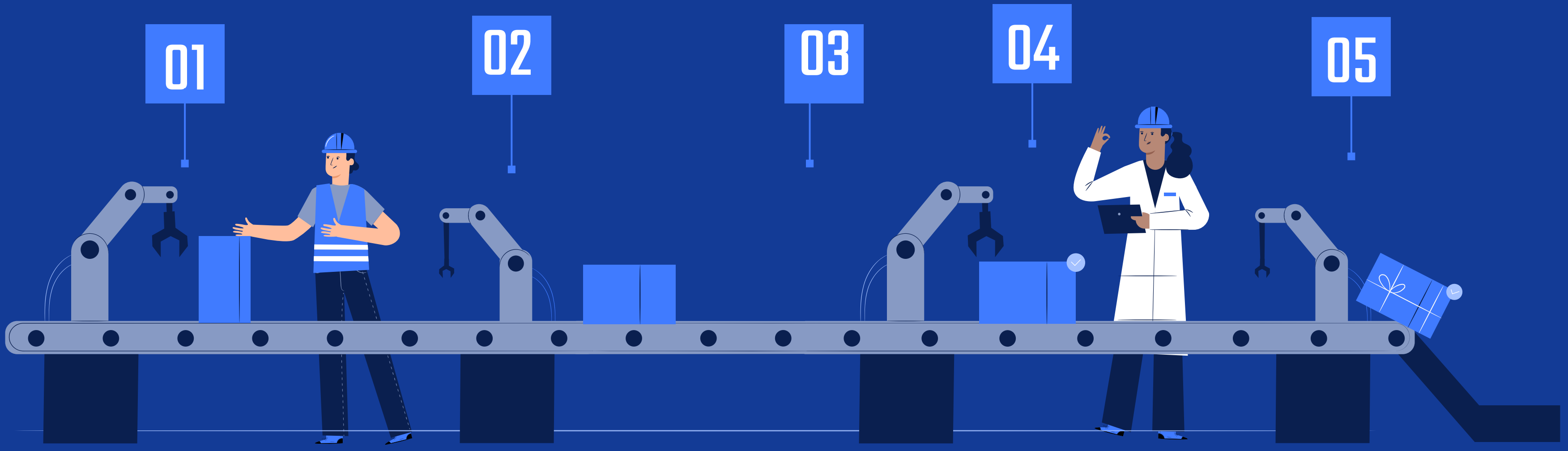
Sequential code in C

Using the functions for testing and printing

Parallelizing the sequential code to CUDA

Tests and measurements

Optimization



01

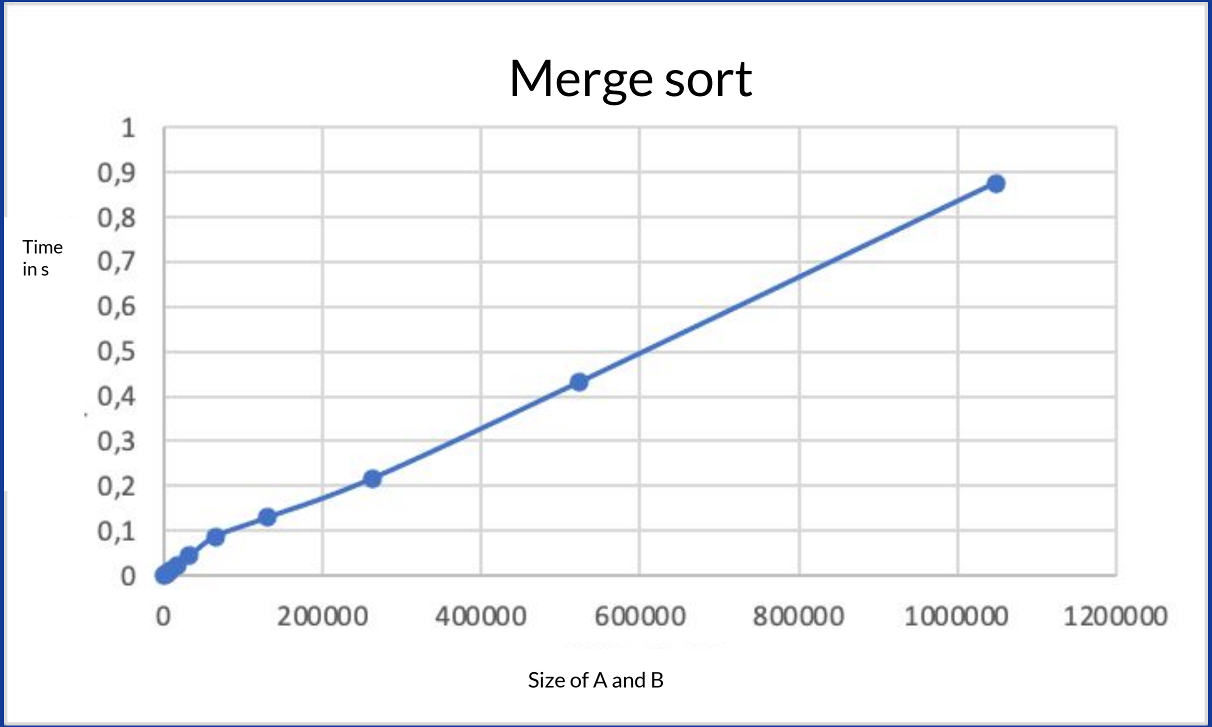
02

03

04

05

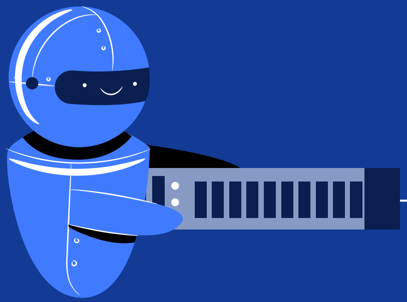
Results


Merge sort
Time in s
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
0
200000
400000
600000
800000
1000000
1200000
Size of A and B

05

Step 5 :

mergeSmallBatch_k



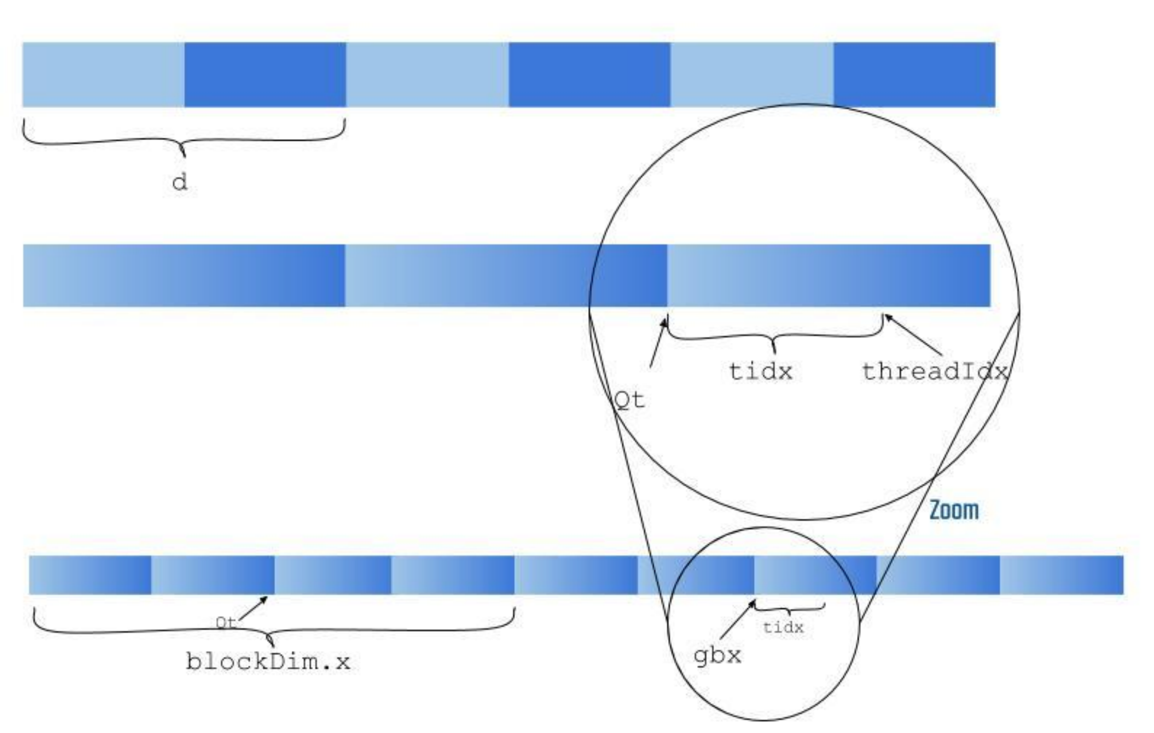
Step 4

Explain why the indices

- `int tidx = threadIdx.x%d;`
- `int Qt = (threadIdx.x-tidx)/d;`
- `int gbx = Qt + blockIdx.x*(blockDim.x/d);`

are important in the definition of mergeSmallBatch k.

Question 4



Step 5

Write the kernel mergeSmallBatch k that batch merges two by two $\{Ai\ \}1 \leq i \leq N$ and $\{Bi\ \}1 \leq i \leq N$. Give the execution time with respect to $d = 4, 8, \ldots, 1024$.

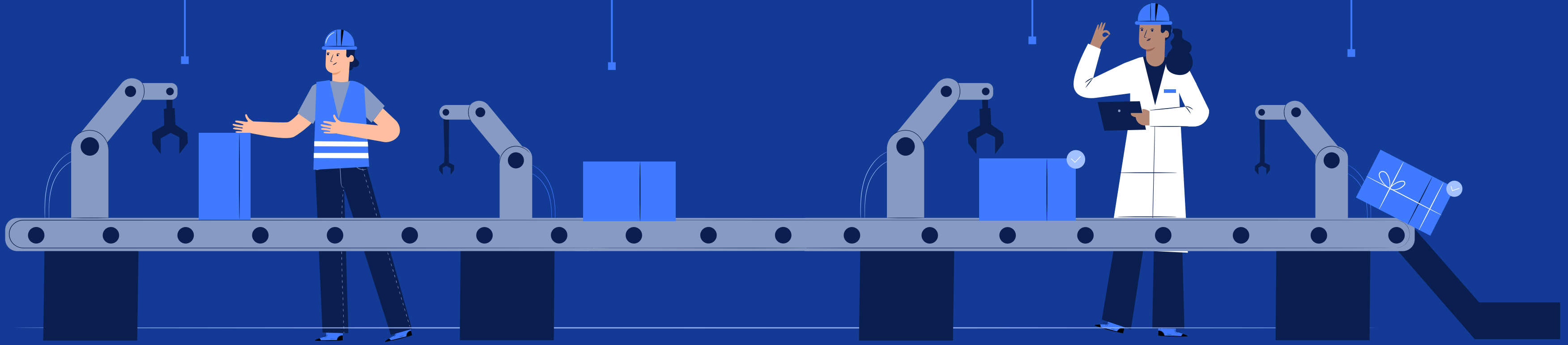
Process

01 Reusing MergeSmall

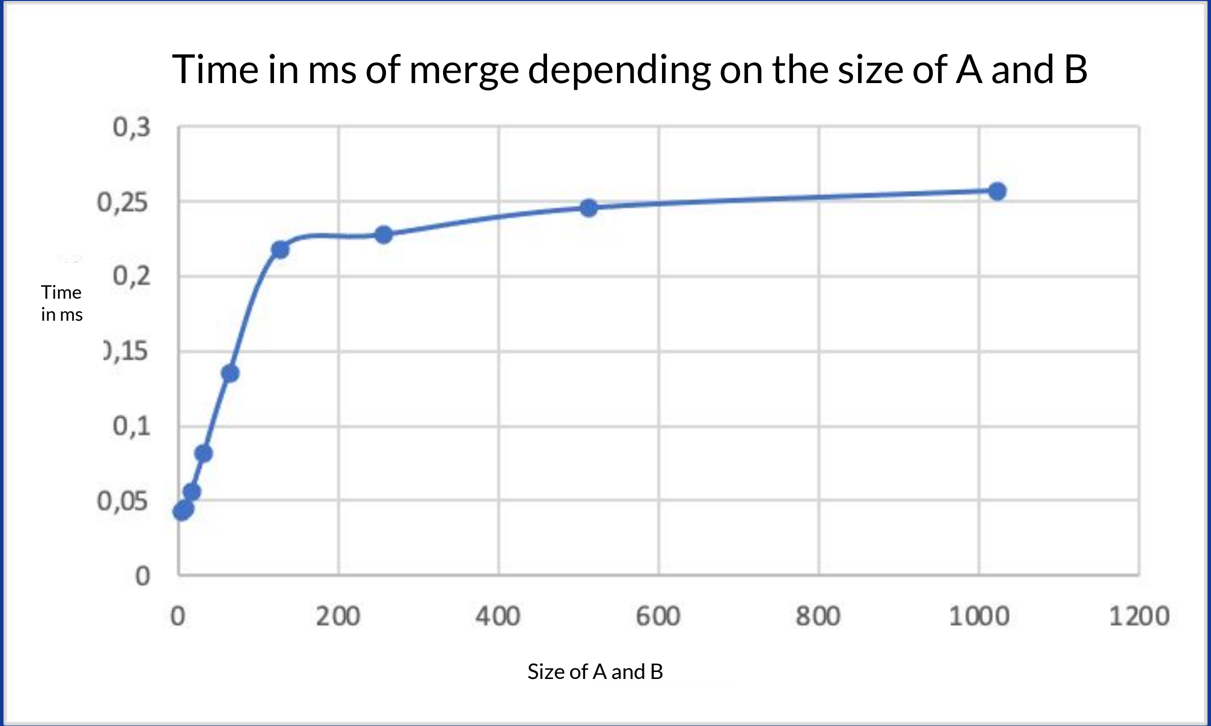
02 Using the indices in question 4

04 Tests and measurements

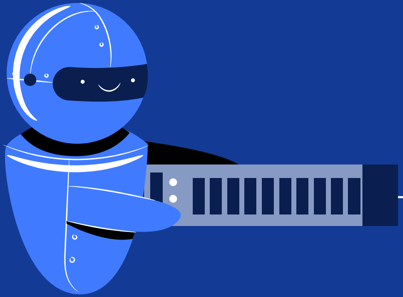
05 Optimization



Results

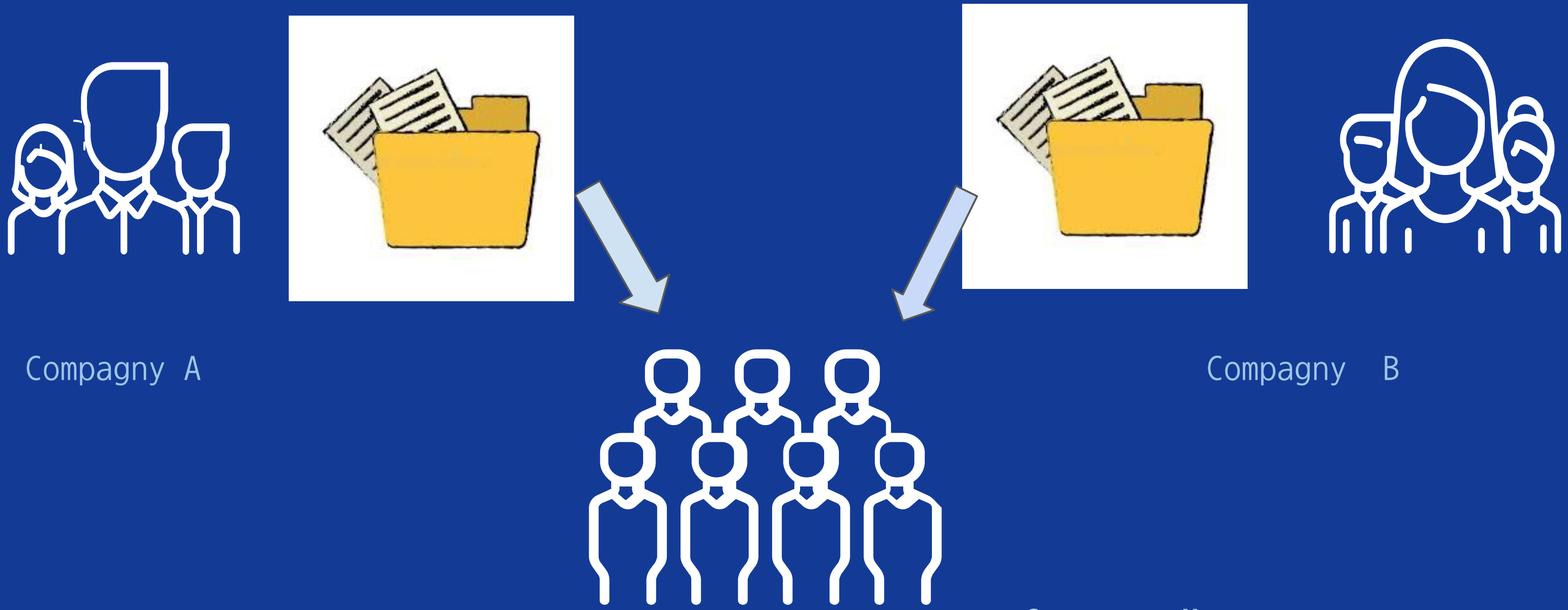

Time in ms of merge depending on the size of A and B
0,3
0,25
0,2
0,15
0,1
0,05
0
Time in ms
0
200
400
600
800
1000
1200
Size of A and B

05 Applications



From a "practical" point of view

Company Merger

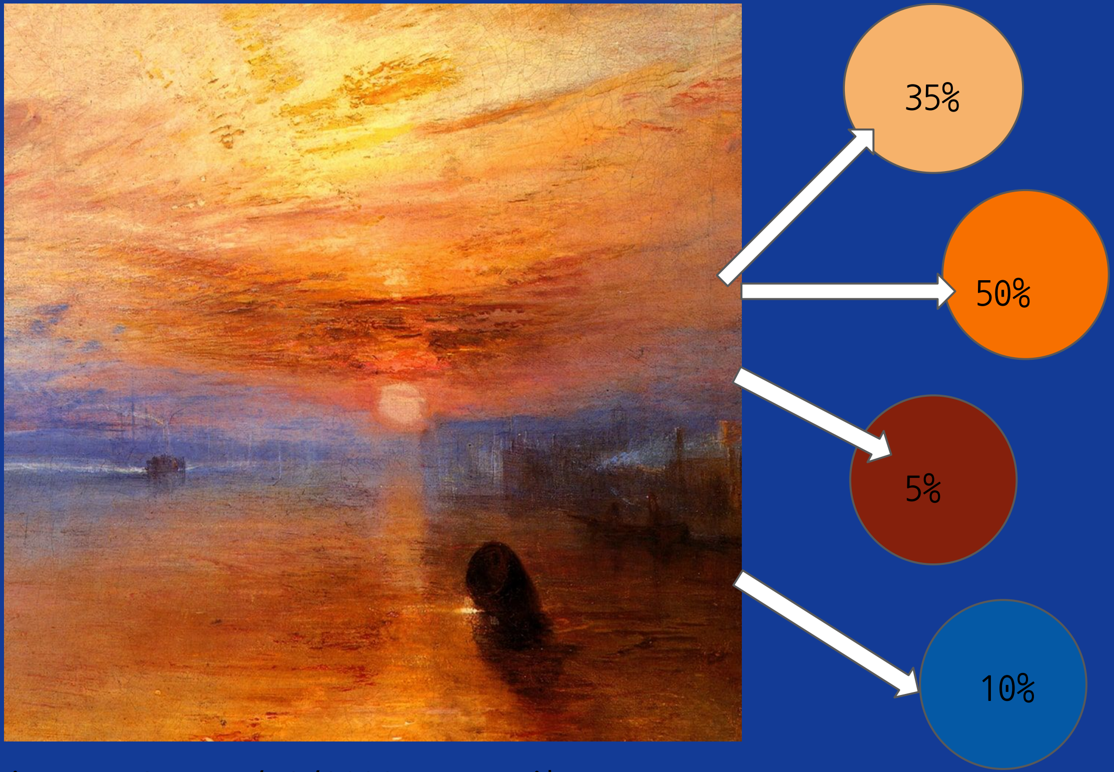


Compagny A

Compagny B

Compagny M

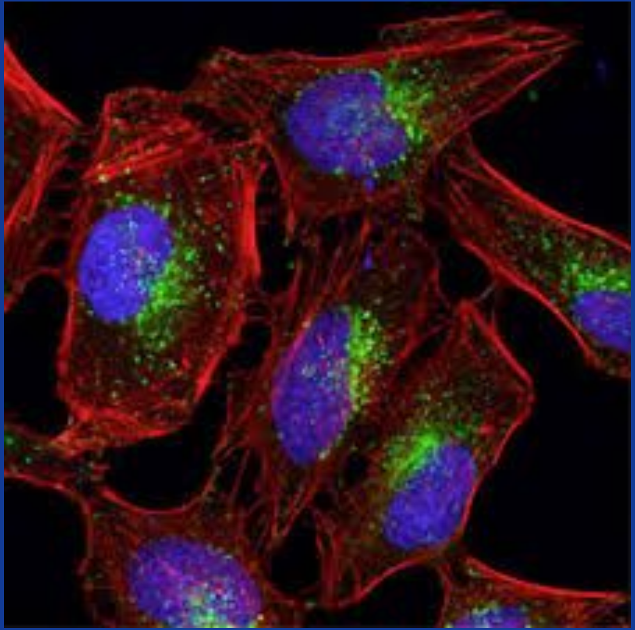
Image processing: in art

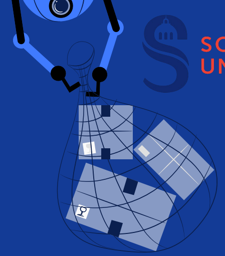


- Recognize the color palette
- Recognize the painter
- Sort by color
- Recognizing the artistic movement
- Recognize the artist's period

Image processing: in medicine

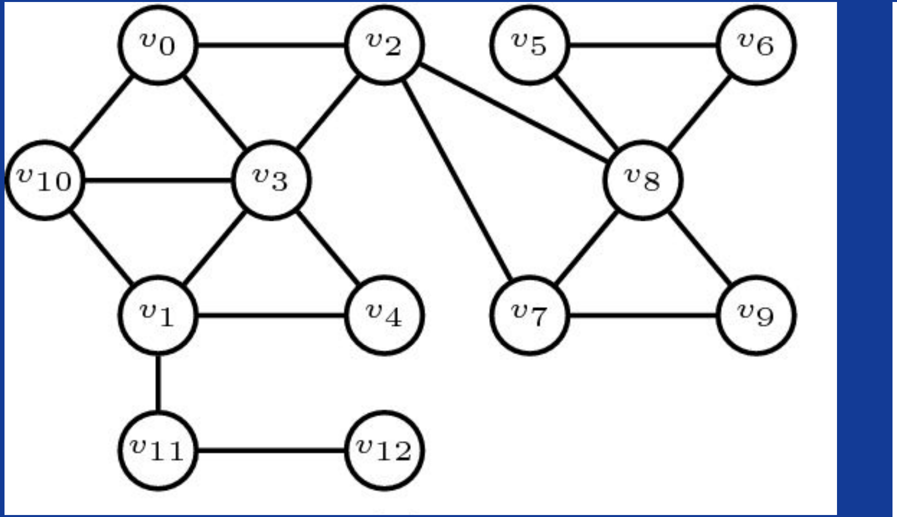
- Counting of cancer cells
- Identify how many there are

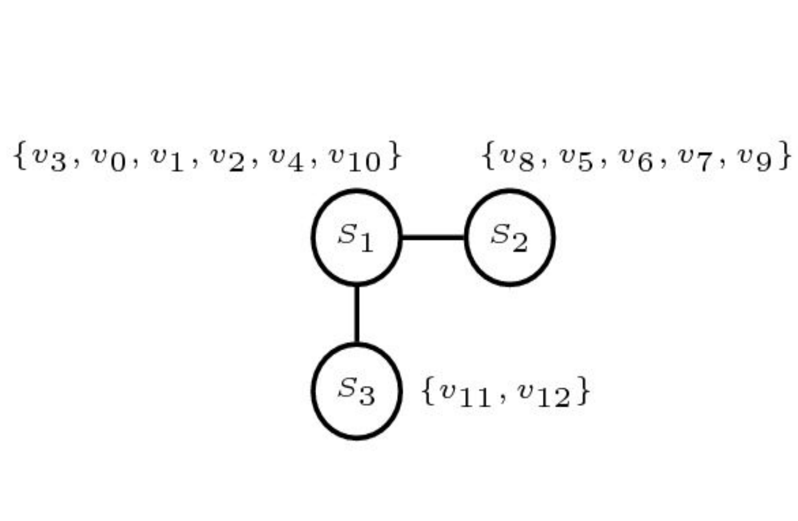




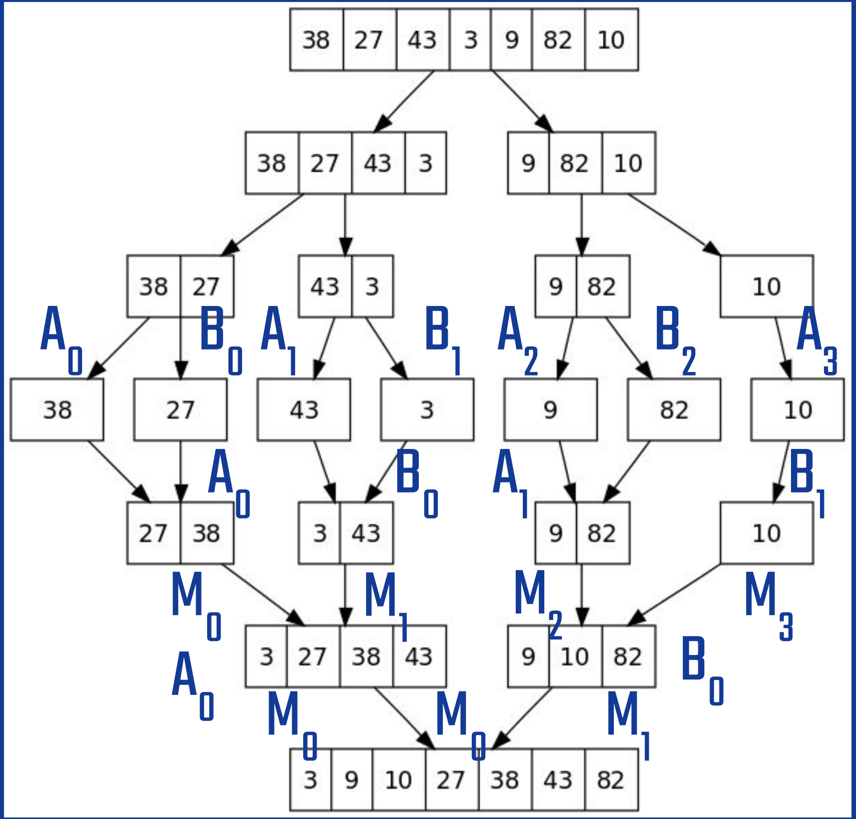
From a "technical" point of view

Web graph compression




$\{v_3, v_0, v_1, v_2, v_4, v_{10}\}$
$\{v_8, v_5, v_6, v_7, v_9\}$
S_1
S_2
S_3
$\{v_{11}, v_{12}\}$

Merge Sort with MergeSmallBatch



POLYTECH SORBONNE

SORBONNE UNIVERSITÉ

Comparaisons

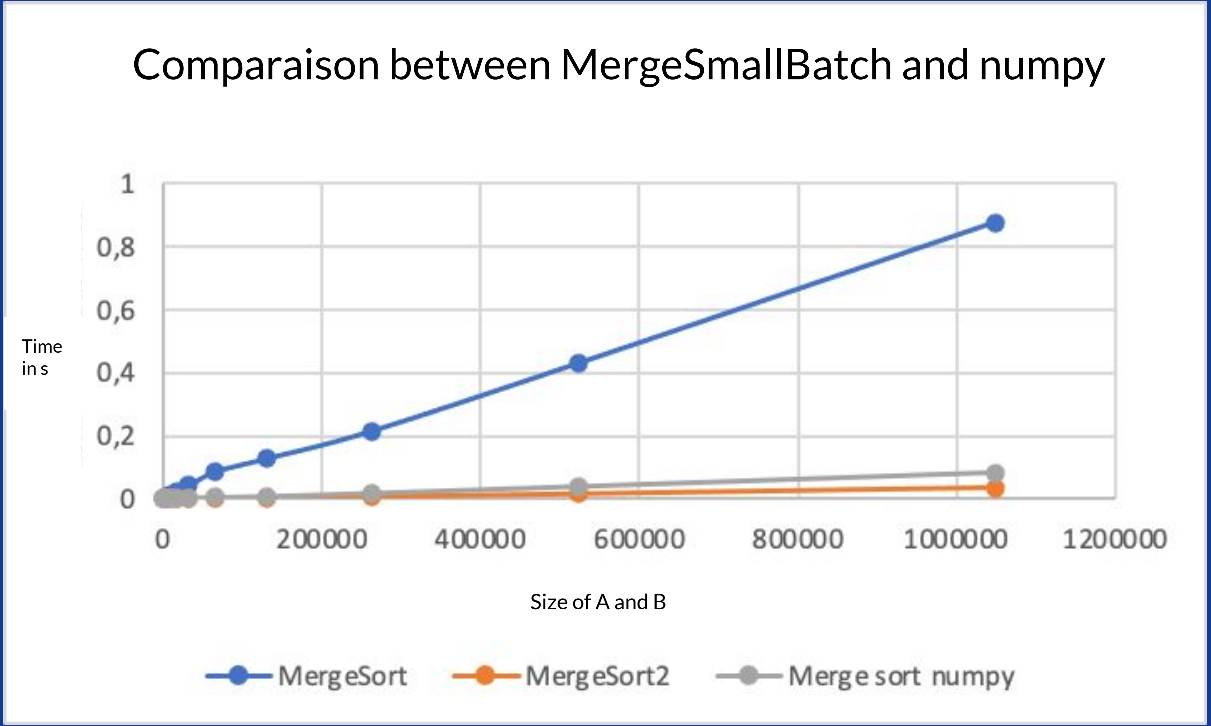
01.

Comparison between the execution times of question 3 and 5

02.

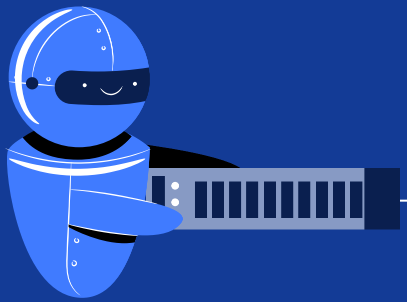
Comparaison between MergeSmallBatch and Numpy.sort()

Results


Comparaison between MergeSmallBatch and numpy
Time in s
1
0,8
0,6
0,4
0,2
0
0
200000
400000
600000
800000
1000000
1200000
Size of A and B
MergeSort
MergeSort2
Merge sort numpy

End

Conclusion



Conclusion

GPU

Discover the world of the GPU in practice

Cuda - Parallelism

Discover another way to set up parallelism

Problems

- Middle
- Infinite loop
- Managing the leftovers

Think about different possible applications

Thank you

Do you have any question ?

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Stories**