

HPCA project

Batch Sort and Merge Path Sort

RUN TIME



Problem to solve

Is it better to sort **using GPU** rather than **using CPU**?

variable names :

- ❖ A,B and M are integer array
- ❖ A,B are sorted
- ❖ M contains all results
- ❖ M is not sort at the beginning

TABLES OF CONTENTS

01

MERGE ON A BLOCK

Of two sorted array,
A and B

02

MERGE ON SEVERAL BLOCKS

Of two sorted array,
A and B

03

“MERGE SORT” ON GPU

tree approach

04

SORT WITH BATCHES

tree approach, batches
and streams

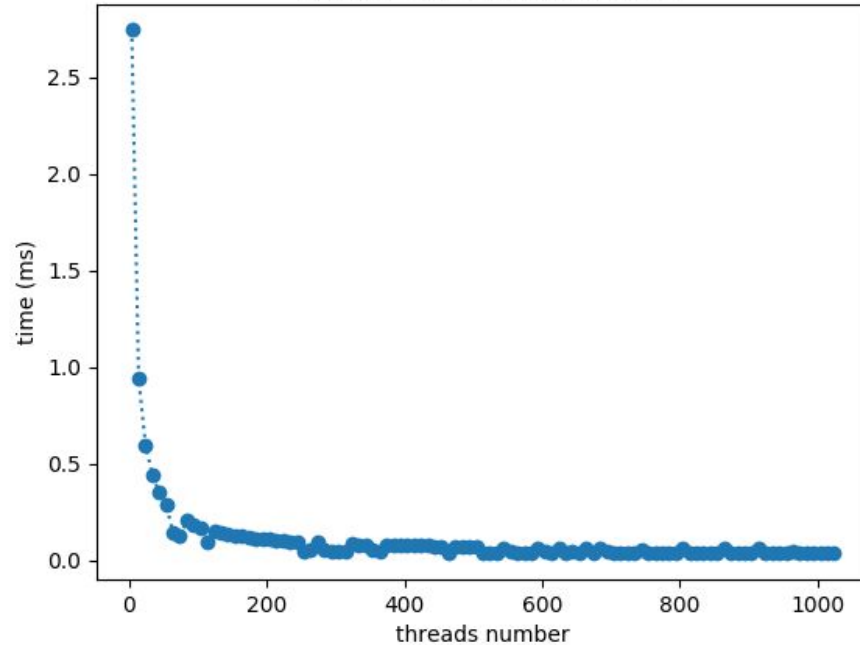


01

MERGE ON ONE BLOCK

ALGO mergeSmall_k

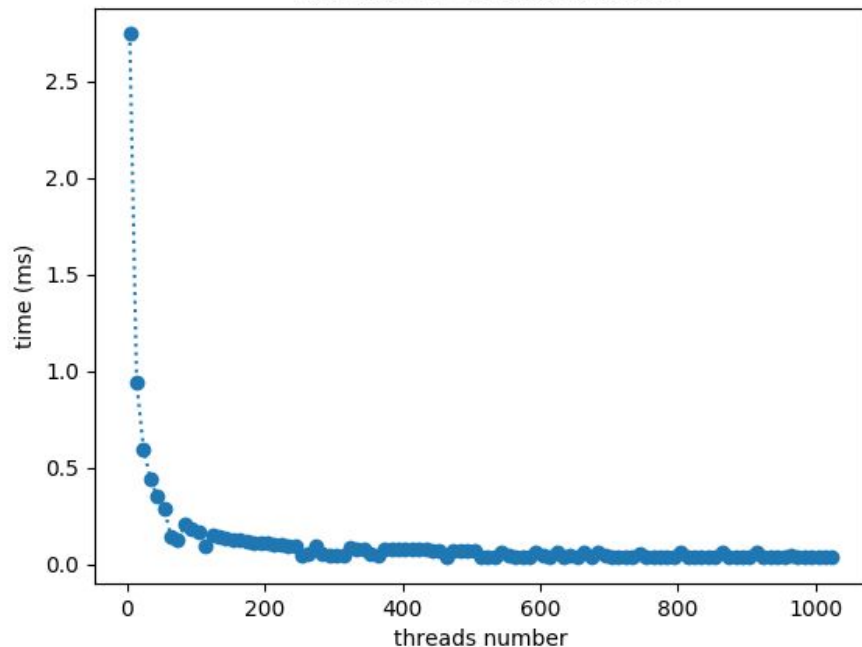
Evolution of the execution time of mergeSmall_k
in relation to threads number



KEY ELEMENTS

ALGO mergeSmall_k

Evolution of the execution time of mergeSmall_k
in relation to threads number

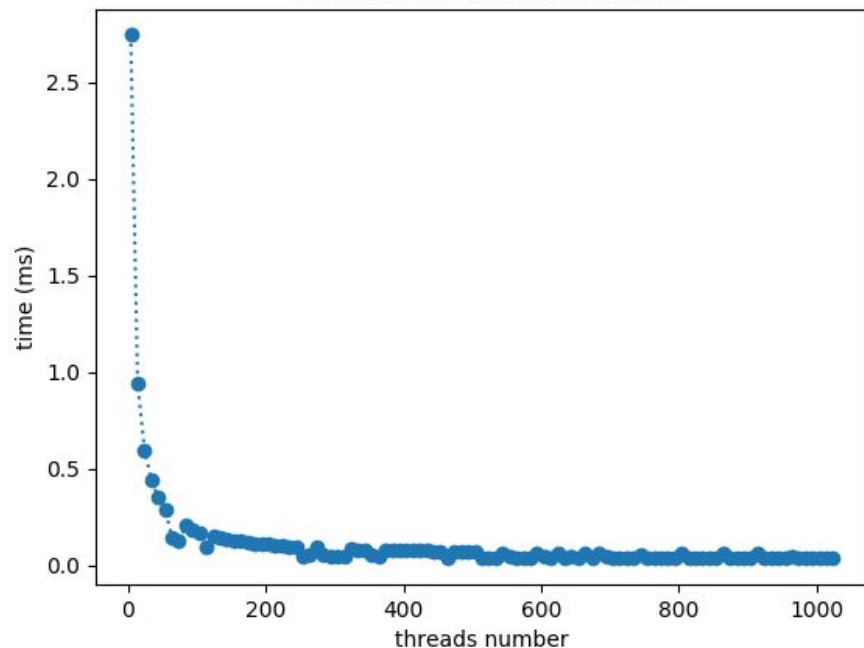


KEY ELEMENTS

- As the number of threads increases, the execution time decreases.

ALGO mergeSmall_k

Evolution of the execution time of mergeSmall_k
in relation to threads number

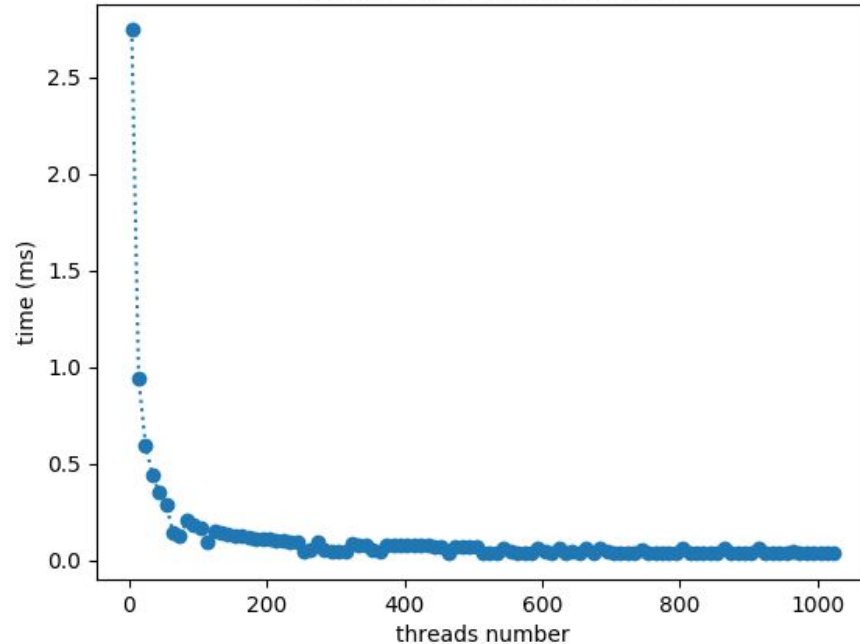


KEY ELEMENTS

- As the number of threads increases, the execution time decreases.
- Small difference between execution time without mentioning the use of shared memory and imposing the use of shared memory

ALGO mergeSmall_k

Evolution of the execution time of mergeSmall_k
in relation to threads number

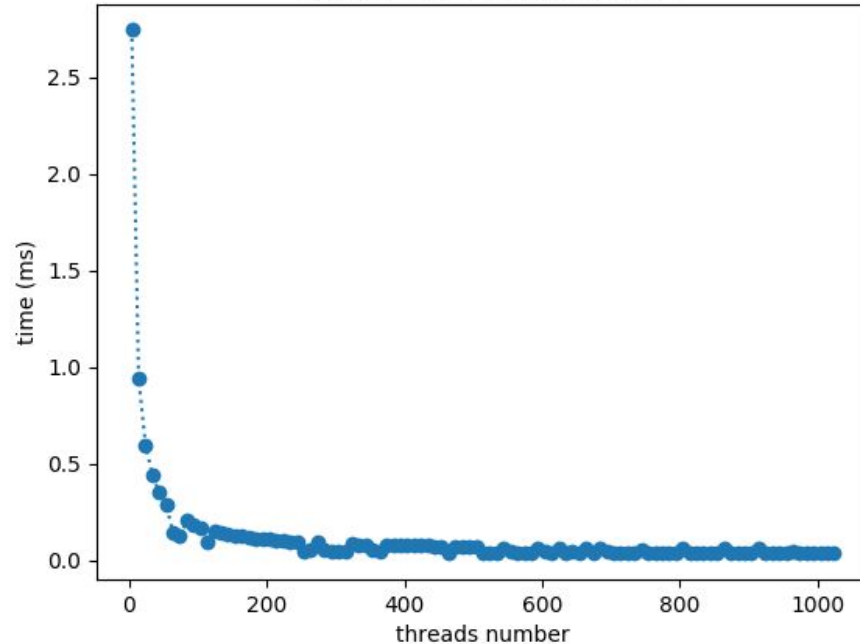


COMMENTS

- As the number of threads increases, the execution time decreases.
- Small difference between execution time without mentioning the use of shared memory and imposing the use of shared memory
- Limit of 1024 elements

ALGO mergeSmall_k

Evolution of the execution time of mergeSmall_k
in relation to threads number



KEY ELEMENTS

- As the number of threads increases, the execution time decreases.
- Small difference between execution time without mentioning the use of shared memory and imposing the use of shared memory
- Limit of 1024 elements
- Sequential code faster : 0.01379ms vs. 0.03517ms (difference negligible)

An abstract graphic on a black background. A vertical yellow line descends from the top center. Along this line and slightly to the left and right are several yellow squares of varying sizes. Some squares are partially cut off by the edges. There are also a few small white squares scattered in the upper corners.

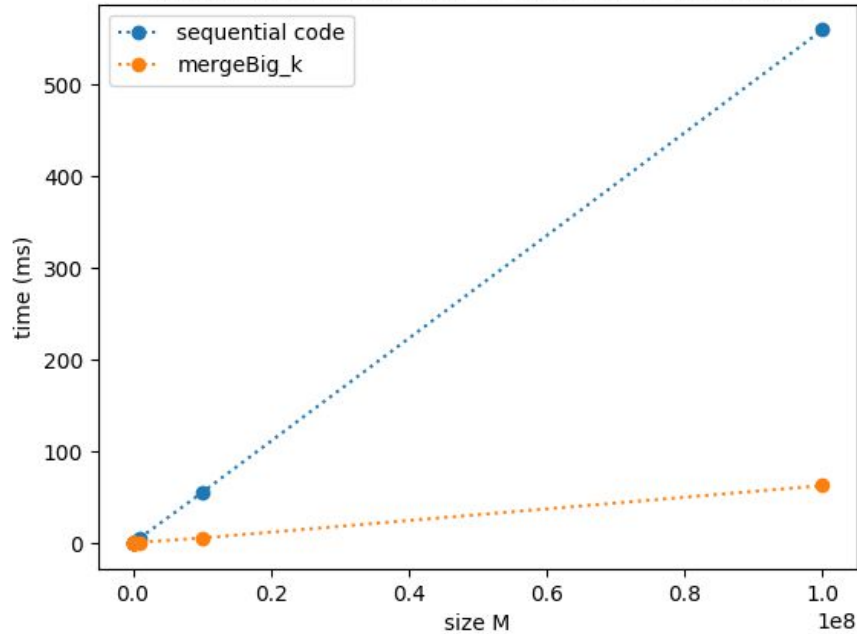
02

MERGE ON SEVERAL BLOCKS

ALGO mergeBig_k

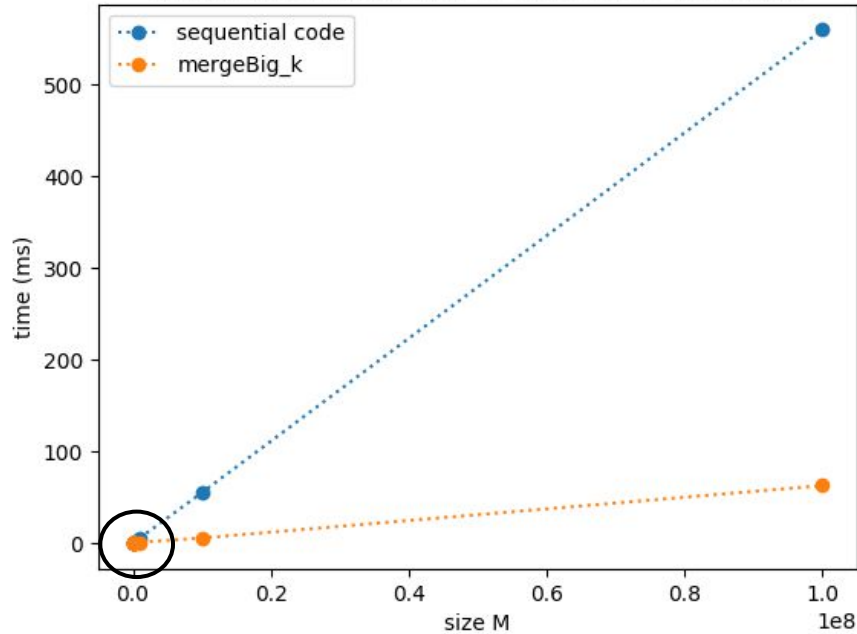
KEY ELEMENTS

Evolution of the execution time of mergeBig_k and the sequential code in relation to the number of elements to be merged



ALGO mergeBig_k

Evolution of the execution time of mergeBig_k and the sequential code in relation to the number of elements to be merged

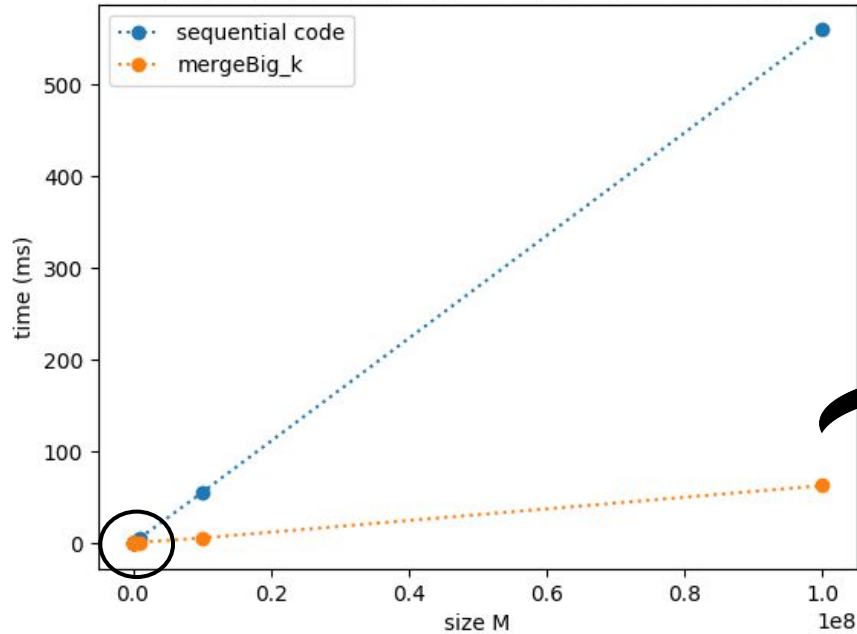


KEY ELEMENTS

- Interesting to switch to GPU with sizes of more than 150 000

ALGO mergeBig_k

Evolution of the execution time of mergeBig_k and the sequential code in relation to the number of elements to be merged

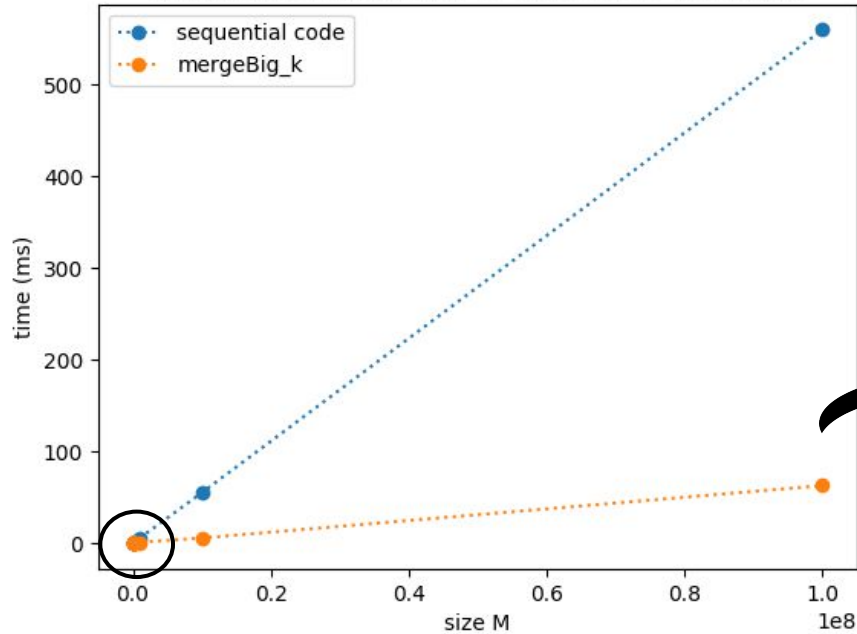


KEY ELEMENTS

- Interesting to switch to GPU with sizes of more than 150 000

ALGO mergeBig_k

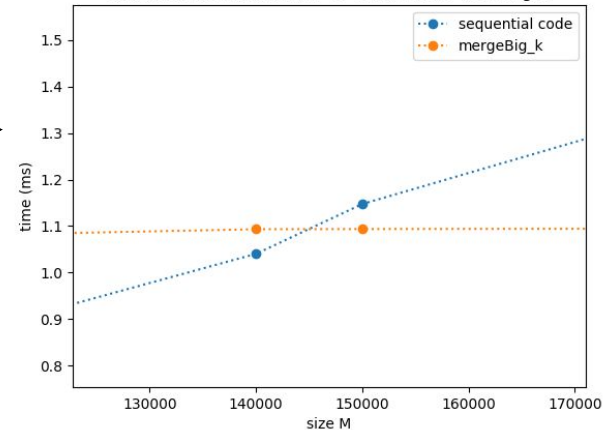
Evolution of the execution time of mergeBig_k and the sequential code in relation to the number of elements to be merged



KEY ELEMENTS

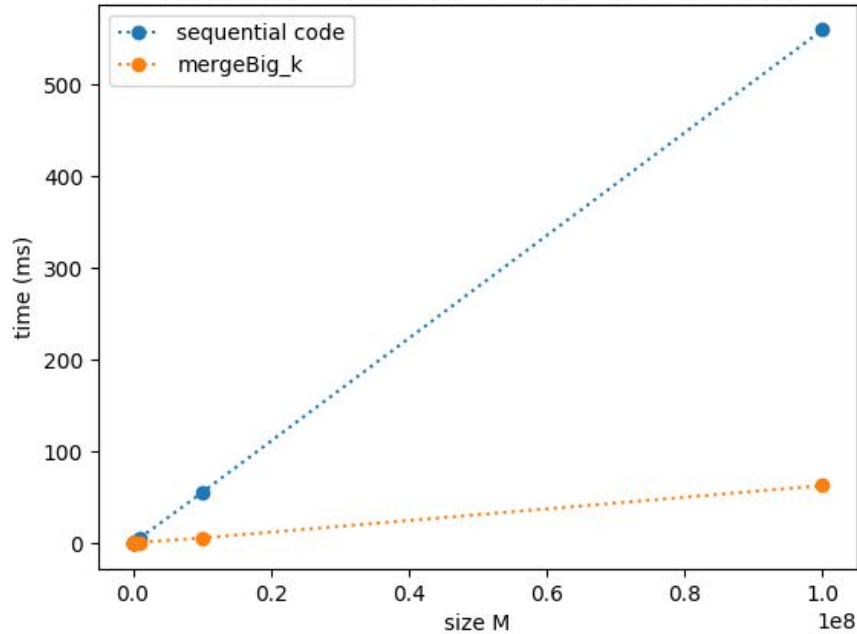
- Interesting to switch to GPU with sizes of more than 150 000

Evolution of the execution time of mergeBig_k and the sequential code in relation to the number of elements to be merged



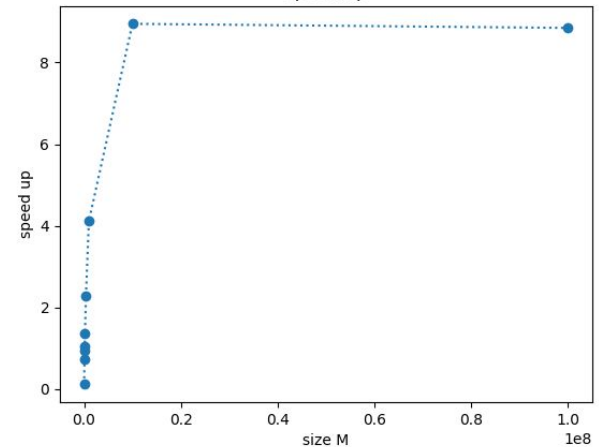
ALGO mergeBig_k

Evolution of the execution time of mergeBig_k and the sequential code in relation to the number of elements to be merged



KEY ELEMENTS

- Interesting to switch to GPU with sizes of more than 150 000
- Speed up of more than 8.



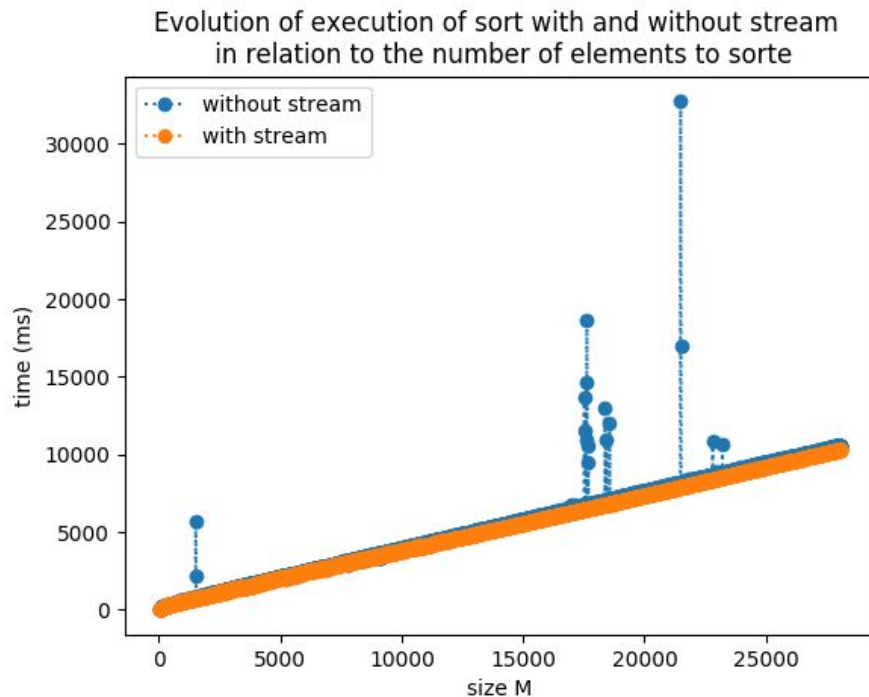


03

“MERGE SORT” ON GPU

SORT ON GPU

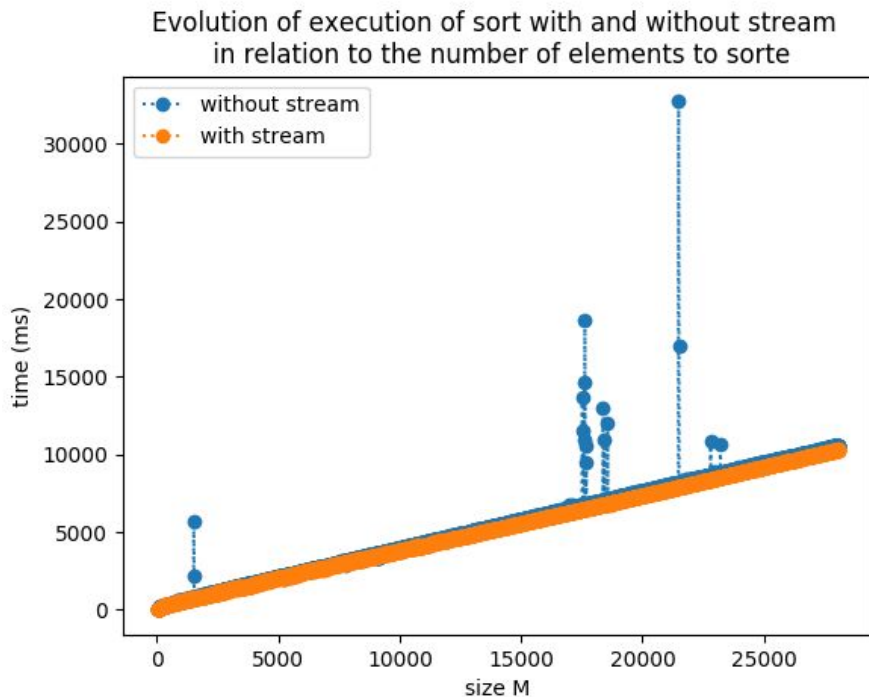
KEY ELEMENTS



SORT ON GPU

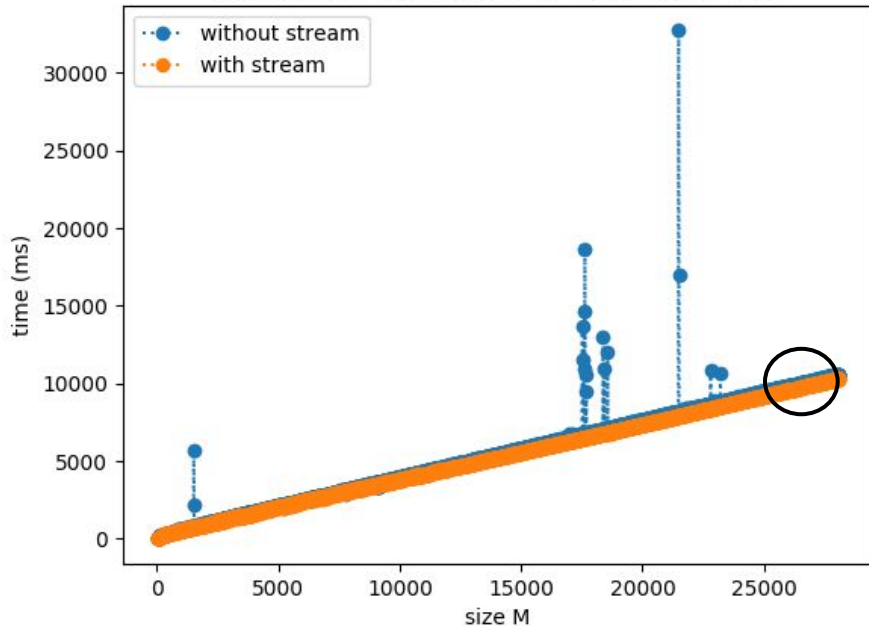
KEY ELEMENTS

- Better runtime stability with streams



SORT ON GPU

Evolution of execution of sort with and without stream
in relation to the number of elements to sort



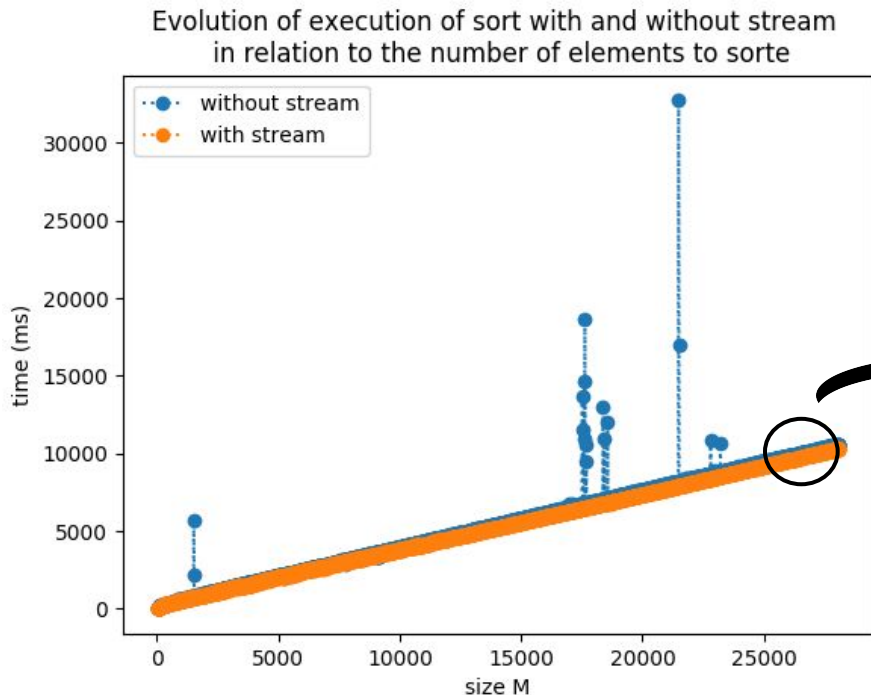
KEY ELEMENTS

- Better runtime stability with streams
- Slight improvement in execution time with streams (streams need larger and longer computations to be profitable)

SORT ON GPU

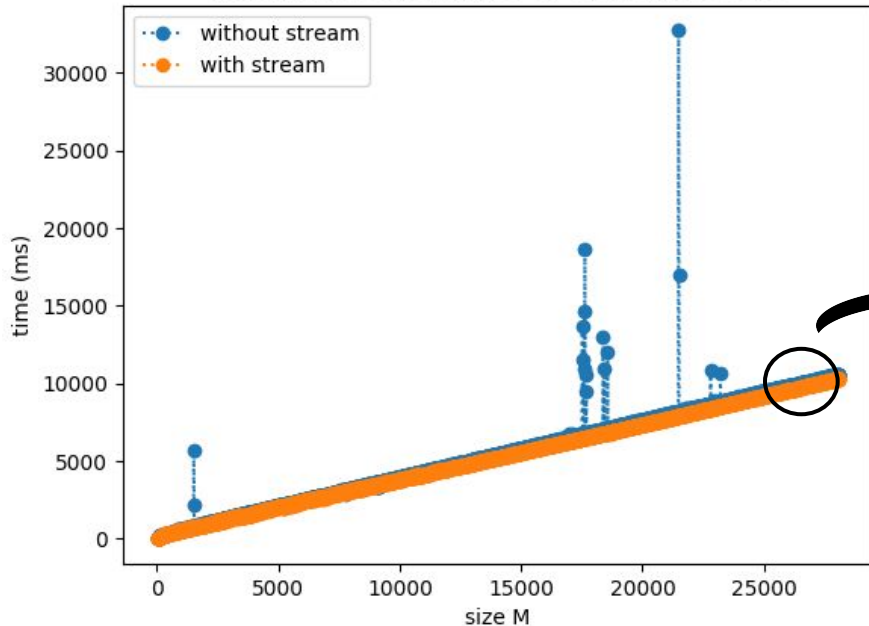
KEY ELEMENTS

- Better runtime stability with streams
- Slight improvement in execution time with streams (streams need larger and longer computations to be profitable)



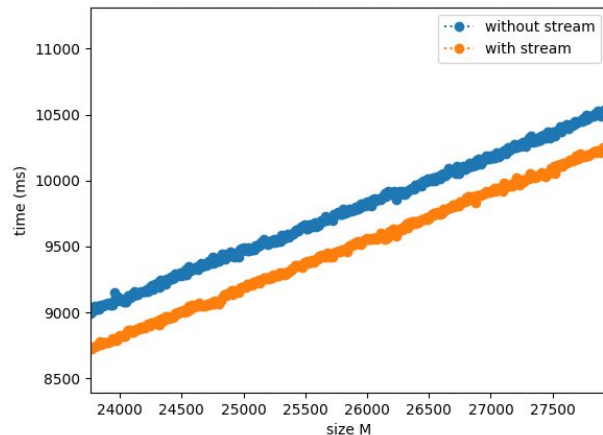
SORT ON GPU

Evolution of execution of sort with and without stream
in relation to the number of elements to sort



KEY ELEMENTS

- Better runtime stability with streams
- Slight improvement in execution time with streams (streams need larger and longer computations to be profitable)



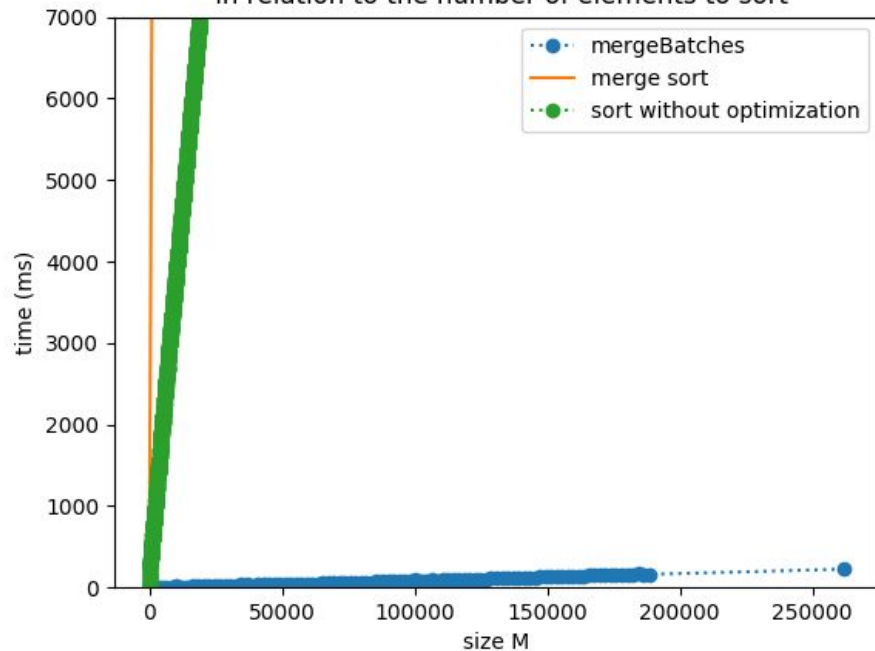


04

SORT WITH BATCHES

SORT GPU WITH STREAM

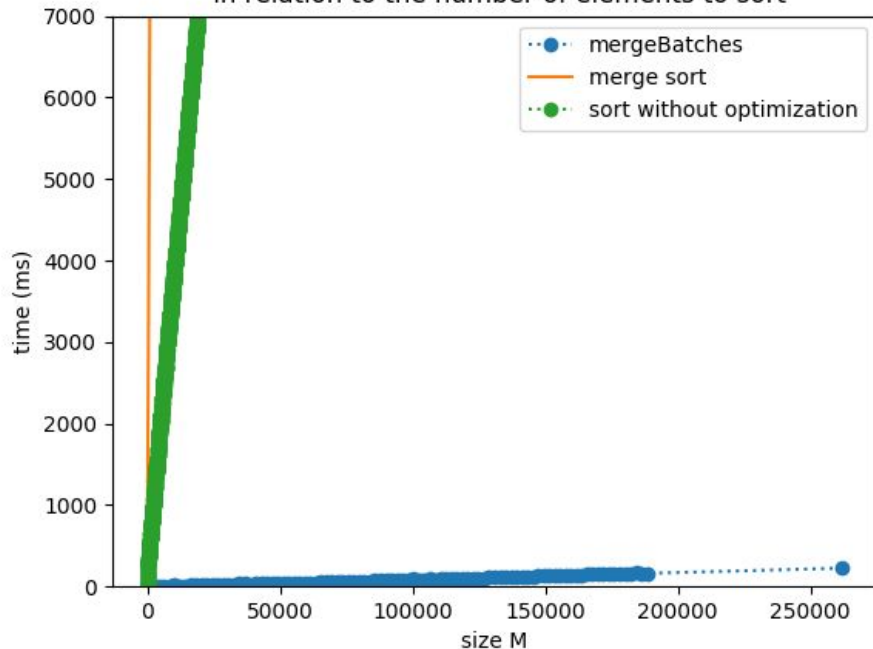
Evolution of execution time of mergeBatches and other sort in relation to the number of elements to sort



KEY ELEMENTS

SORT GPU WITH STREAM

Evolution of execution time of mergeBatches and other sort in relation to the number of elements to sort

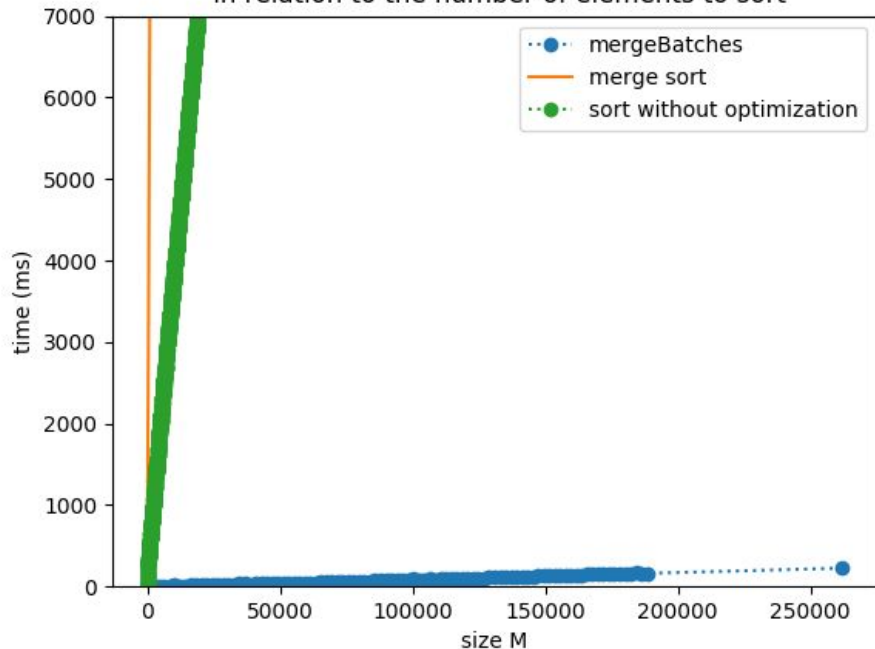


KEY ELEMENTS

- Improvement in execution time compared to merge sort and sort on GPU without optimization

SORT GPU WITH STREAM

Evolution of execution time of mergeBatches and other sort in relation to the number of elements to sort

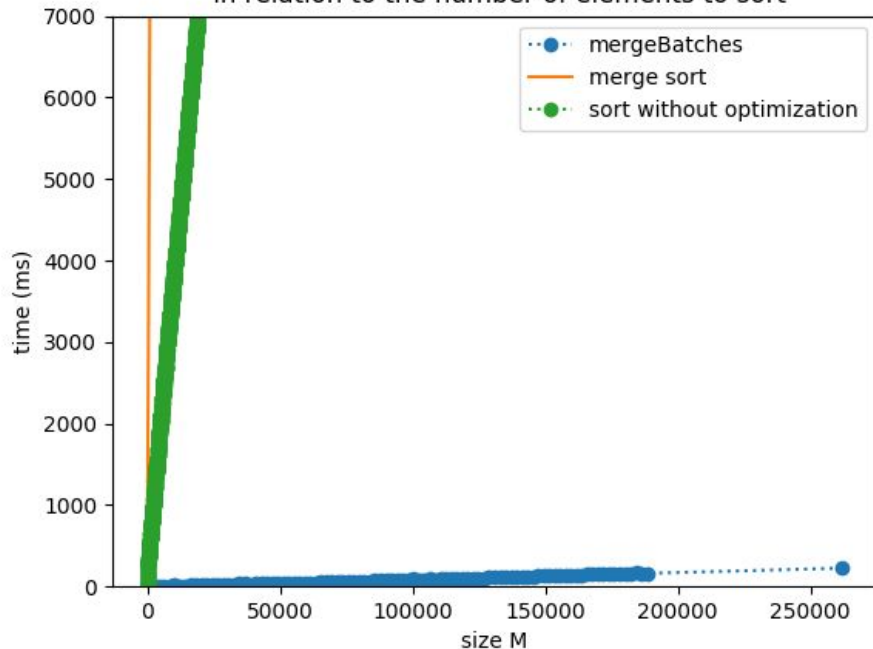


KEY ELEMENTS

- Improvement in execution time compared to merge sort and sort on GPU without optimization
- The runtime of mergeBatches increase but really slowly compare to merge sort or sort without optimization (400 speed up to run)

SORT GPU WITH STREAM

Evolution of execution time of mergeBatches and other sort in relation to the number of elements to sort

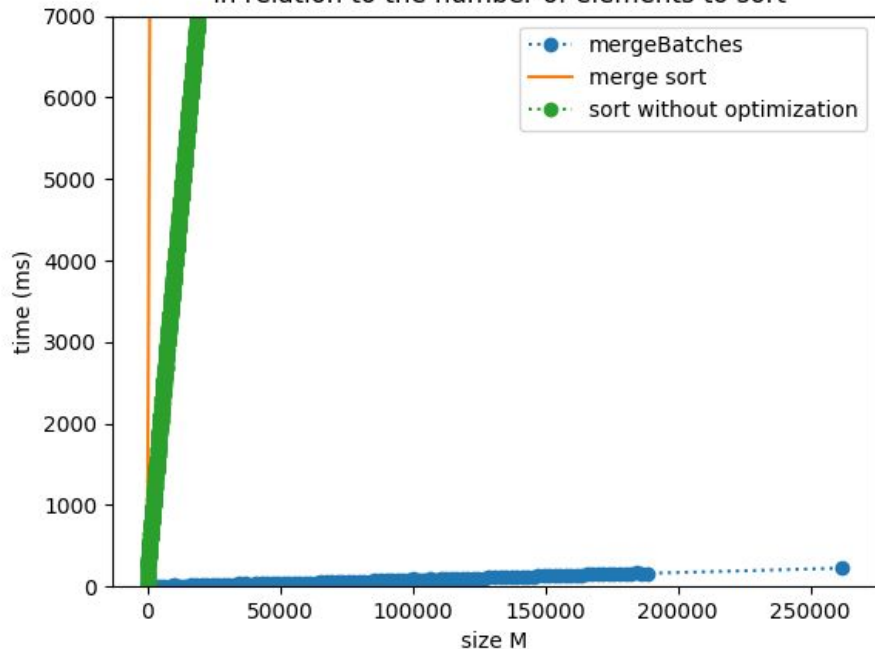


KEY ELEMENTS

- Improvement in execution time compared to merge sort and sort on GPU without optimization
- The runtime of mergeBatches increase but really less than merge sort or sort without optimization (400 speed up to run)
- Importance of adapting the advantages of small sizes (shared and mergeSmall_k algorithm)

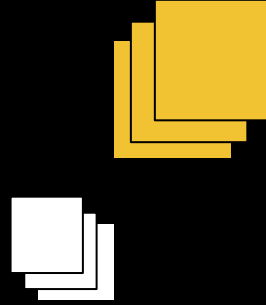
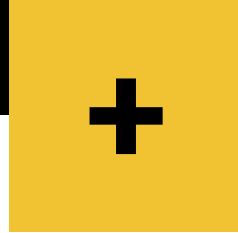
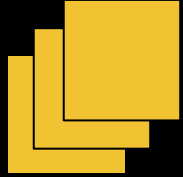
SORT GPU WITH STREAM

Evolution of execution time of mergeBatches and other sort in relation to the number of elements to sort



KEY ELEMENTS

- Improvement in execution time compared to merge sort and sort on GPU without optimization
- The runtime of mergeBatches increase but really less than merge sort or sort without optimization (400 speed up to run)
- Importance of adapting the advantages of small sizes (shared and mergeSmall_k algorithm)
- Benefit of parallelizing kernels with large compute (streams)



REAL CASE

Why sort an array?

- ❖ Some data search algorithms need to have array sorted (like dichotomous)

Why sort an array?

- ❖ Some data search algorithms need to have array sorted (ex dichotomous)
- ❖ To insert an element in an array

Why sort an array?

- ❖ Some data search algorithms need to have array sorted (ex dichotomous)
- ❖ To insert an element in an array
- ❖ Accelerated data filtering algorithms

Why sort an array?

- ❖ Some data search algorithms need to have array sorted (ex dichotomous)
- ❖ To insert an element in an array
- ❖ Accelerated data filtering algorithms
- ❖ Improvement of data readability

Why sort an array?

- ❖ Some data search algorithms need to have array sorted (ex dichotomous)
- ❖ To insert an element in an array
- ❖ Accelerated data filtering algorithms
- ❖ Improvement of data readability
- ❖ Ranking

3	1	2
---	---	---

Why sort an array?

- ❖ Some data search algorithms need to have array sorted (ex dichotomous)
- ❖ To insert an element in an array
- ❖ Accelerated data filtering algorithms
- ❖ Improvement of data readability
- ❖ Ranking
- ❖ Ordering experiment value otherwise



THANK YOU
