Impact of task granularity in unbalanced worloads

Xavier Valls

ROOT

Data Analysis Framework

https://root.cern

Task granularity

- Dividing the work at thread-level the traditional way (1 chunk/core) is not optimal if the workload distribution between each of these partitions is unbalanced.

- if a partition finishes earlier, will remain in idle state waiting for the remaining ones to finish → suboptimal exploitation of the hardware resources.

- Potential solution: increasing the granularity of the data partitions (creating more tasks).

Chunking in the executors

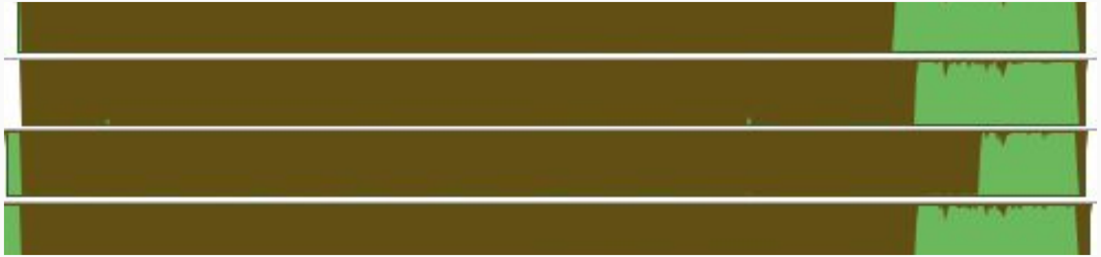
- TThreadExecutor: Available! Parameter in MapReduce to specify the number of partitions.
- TProcessExecutor: Not available yet, adapted for this study

A simple benchmark

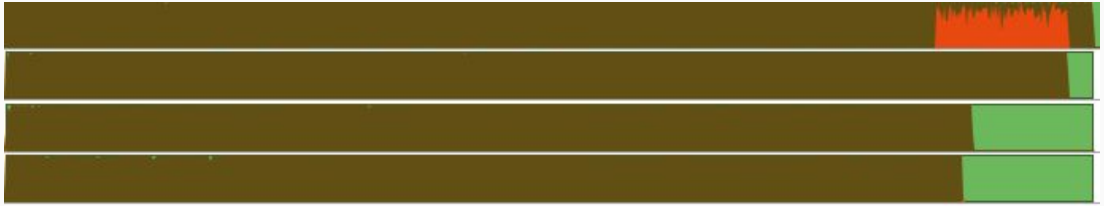
- conditional evaluation of two different implementations of the Vavilov probability distribution function (VavilovFast and VavilovAccurate)
- VavilovFast is 5x faster than VavilovAccurate
- Negative Values of the data: VavilovFast. Positive values: Vavilov Accurate
- Two different distributions of the data

Case 1: slightly unbalanced

Random Gaussian with σ = 1 and μ = −0.25 for the first third of data elements, σ = 1 and μ = 0.25, for the second third, and σ = 1 and μ = −0.75 for the remaining third. 8 ∗ 10^7 points.



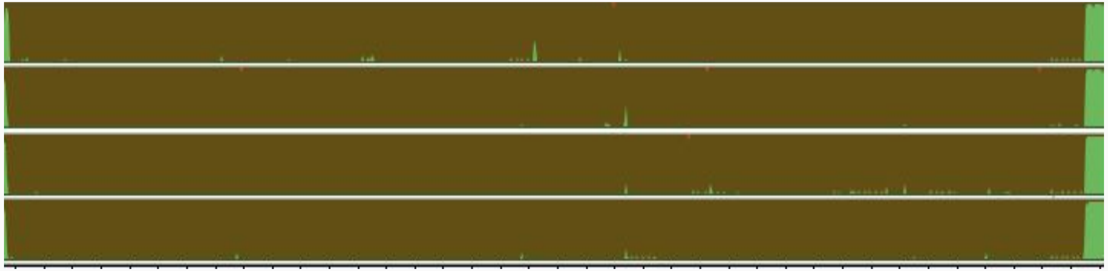
TProcessExecutor
(4 partitions, 2*10^7
elements/chunk)



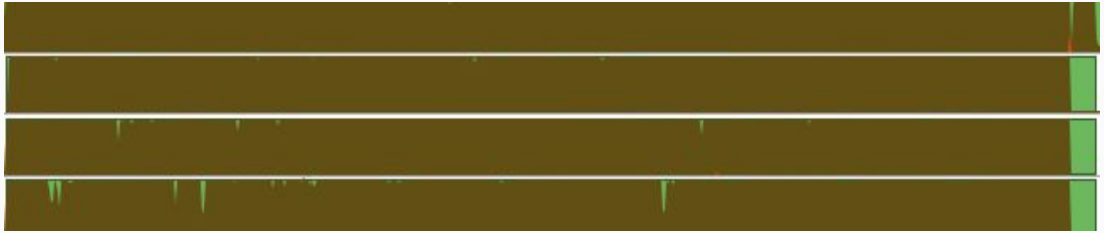
TThreadExecutor
(4 partitions, 2*10^7
elements/chunk)

Case 1: slightly unbalanced

Random Gaussian with σ = 1 and μ = −0.25 for the first third of data elements, σ = 1 and μ = 0.25, for the second third, and σ = 1 and μ = −0.75 for the remaining third. 8 ∗ 10^7 points

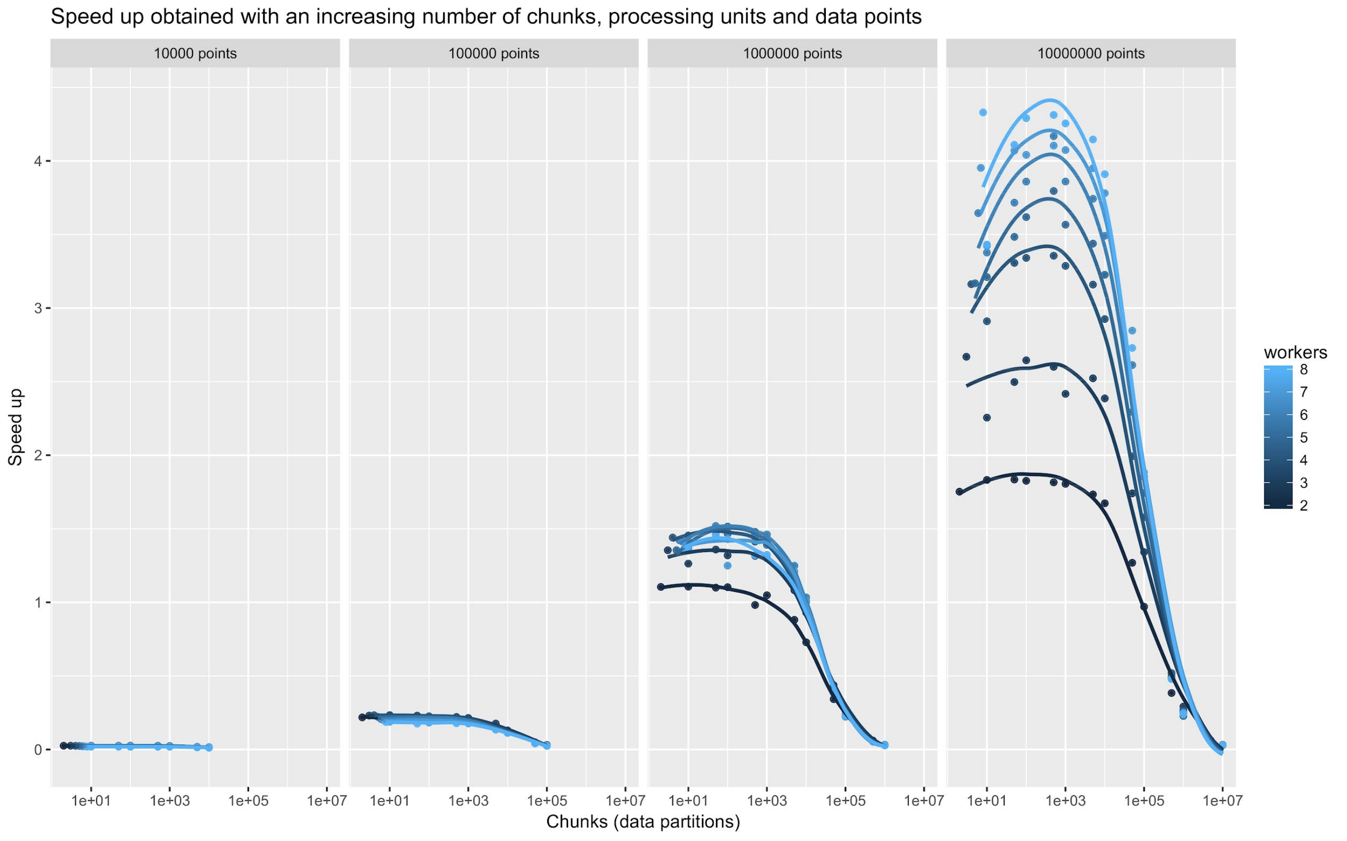


TProcessExecutor
(8000 partitions,
10^4 elements/chunk)

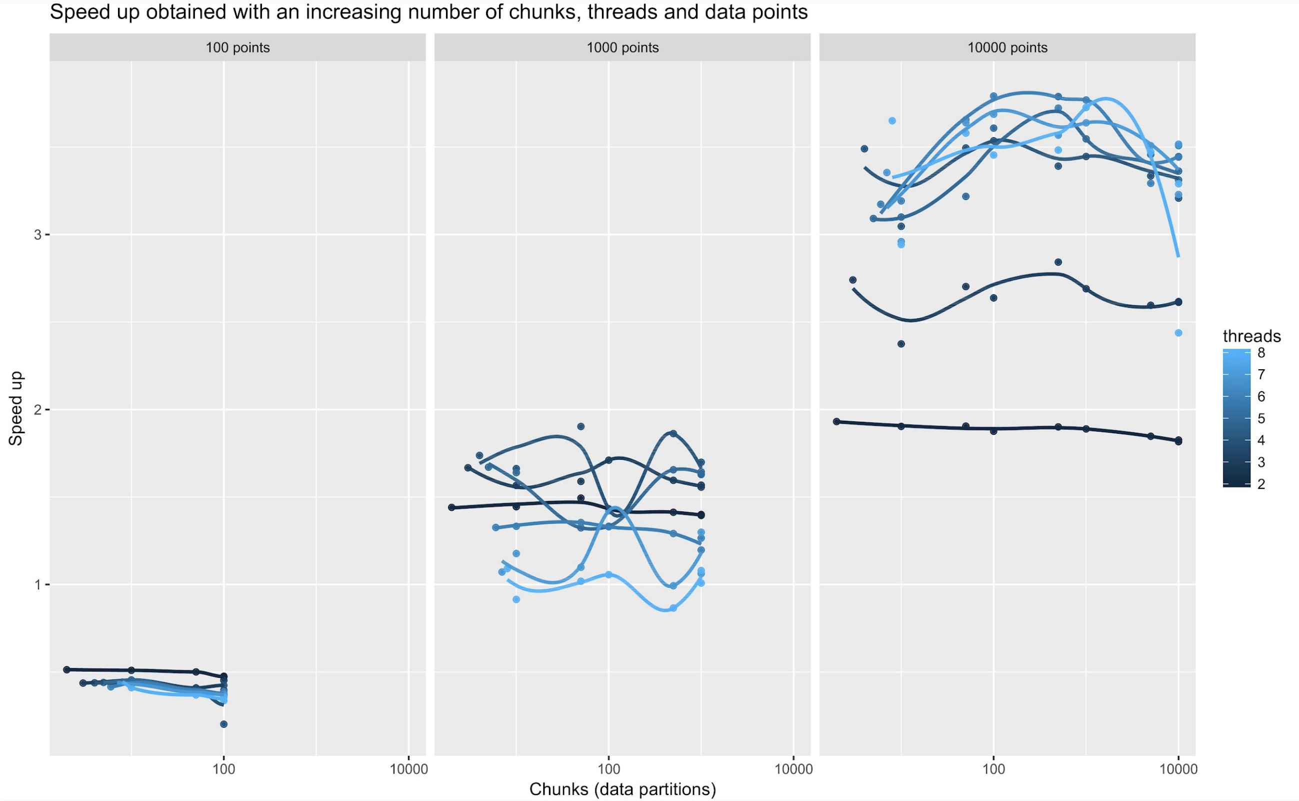


TThreadExecutor
(8000 partitions,
10^4 elements/chunk)

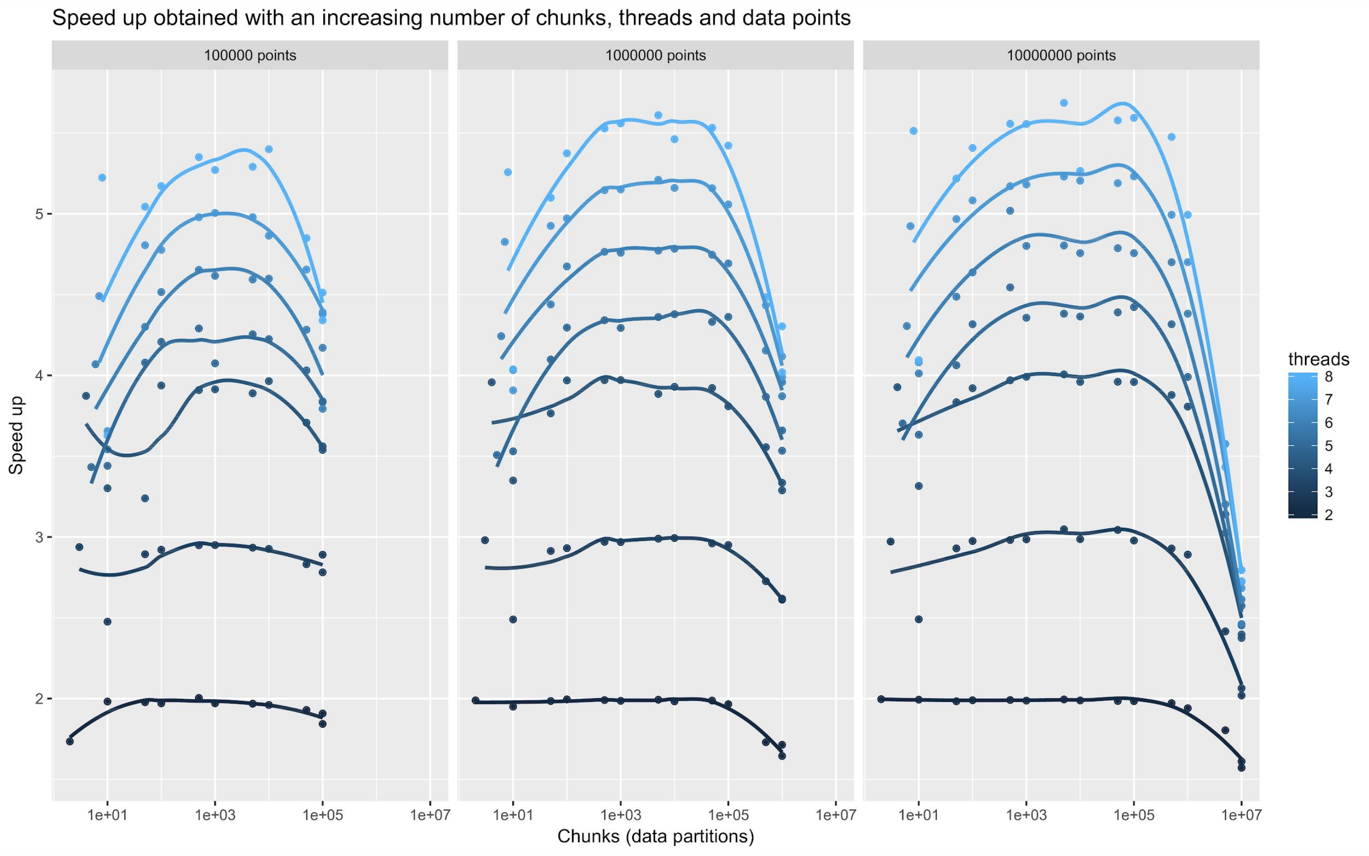
Case 2: benchmark (Multiprocess)



Case 1: benchmark (Multithread I)

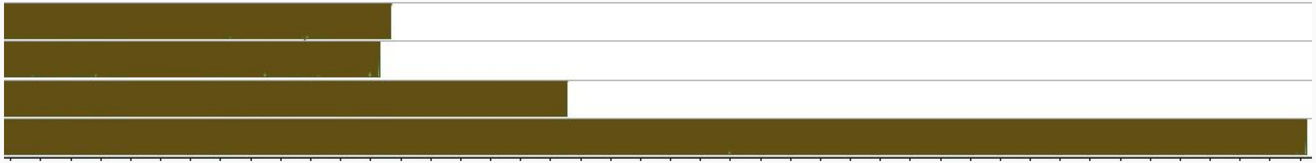


Case 1: benchmark (Multithread II)

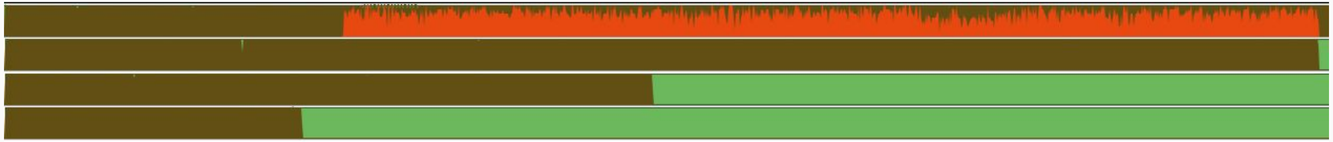


Case 2: extremely unbalanced

- first two thirds of data: VavilovFast
- last third: VavilovAccurate



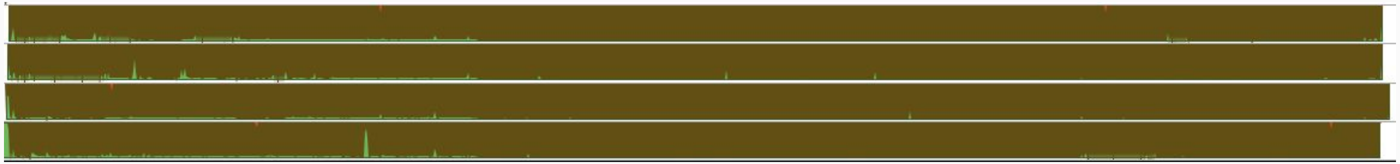
TProcessExecutor
(4 partitions, 2*10^7 elements/chunk)



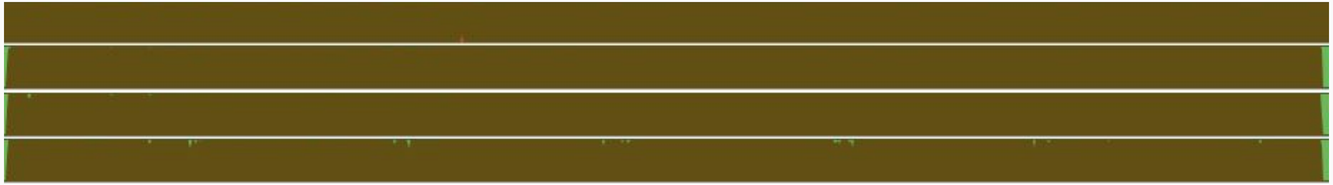
TThreadExecutor
(4 partitions, 2*10^7 elements/chunk)

Case 2: extremely unbalanced

- first two thirds of data: VavilovFast
- last third: VavilovAccurate

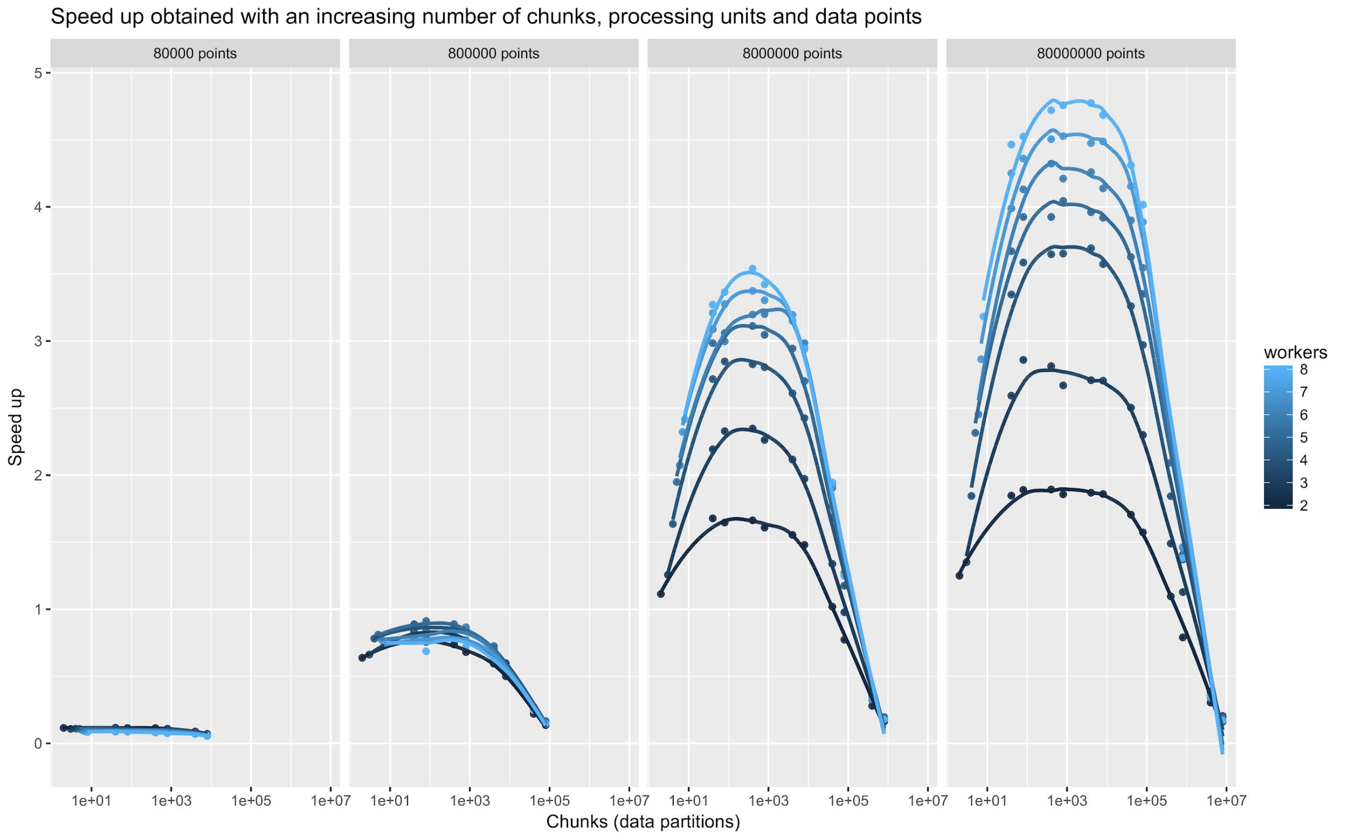


TProcessExecutor
(8000 partitions,
10^4 elements/chunk)

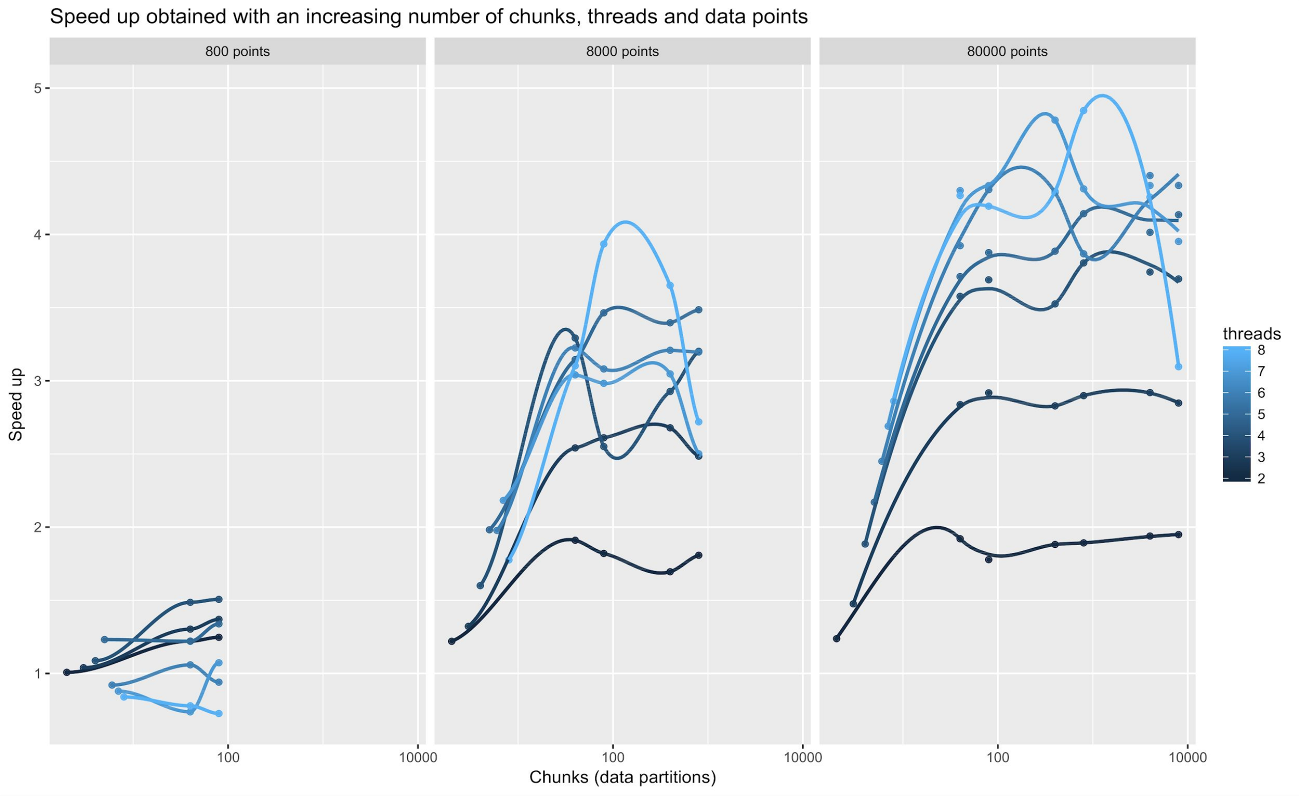


TThreadExecutor
(8000 partitions,
10^4 elements/chunk)

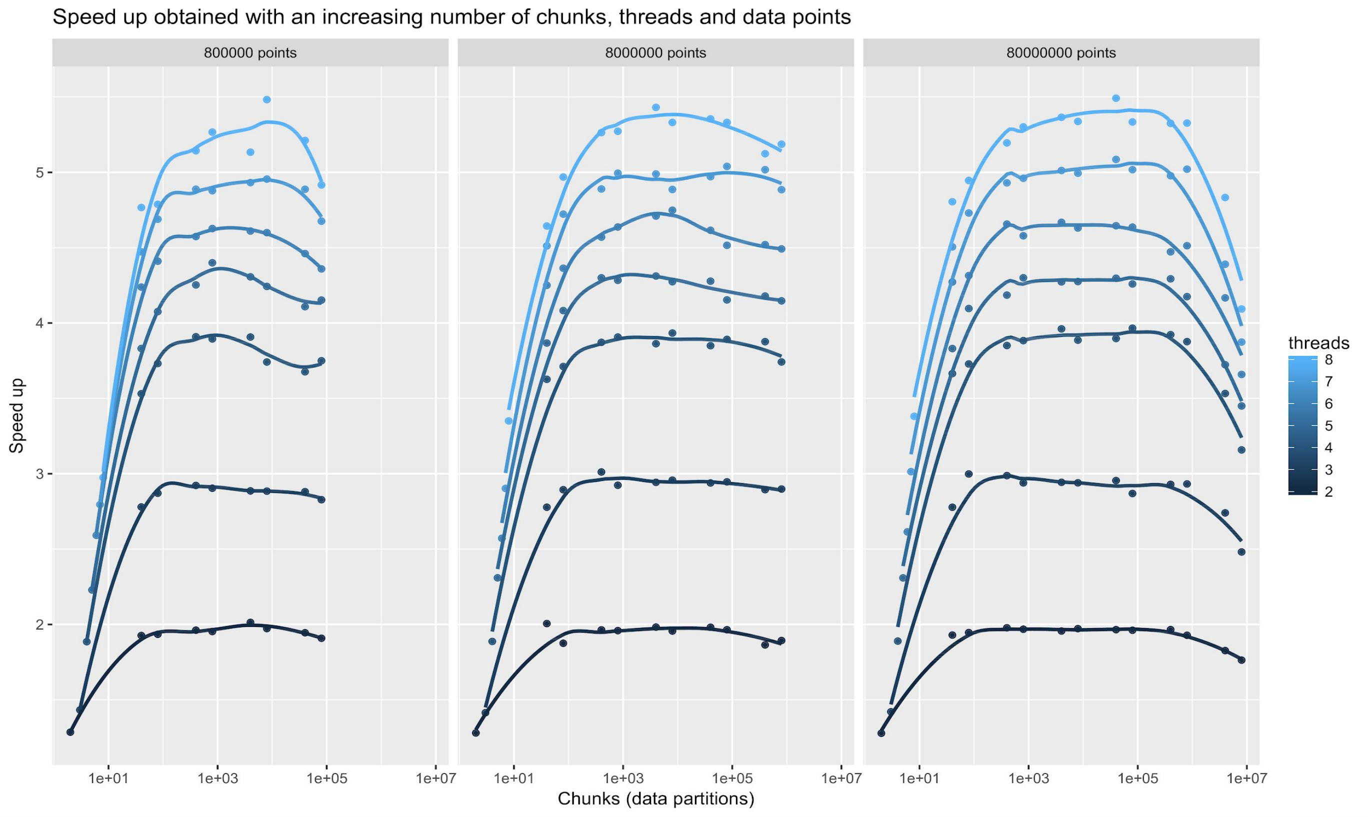
Case 2: benchmark (Multiprocess)



Case 2: benchmark (Multithread I)



Case 2: benchmark (Multithread II)



Missing

- Take more measurements to see if the hyperthreading speed up in the MT case with few data points stops fluctuating.
- Make a more fair comparison (same data points/ same points per chunk in both examples)
- Quantify in terms of task overhead.