Numerical Scientific Computing.
2nd Hand-in

Vasiliki Ismiroglou

Regarding the use of dask arrays.

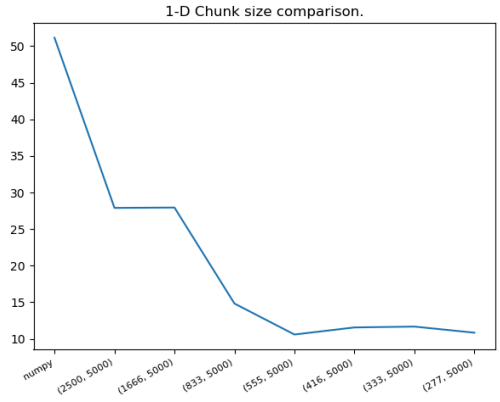
Dask arrays are calculated lazily and they are only a better option to numpy arrays specifically for the task of parallelizing processes, doing calculations in chunks at a time etc. For that reason, the only dask array in the code is the complex grid itself.
The map_blocks function is used to map the mandelbrot function to every chunk of the grid individually. Within the mandelbrot function, using dask arrays was found to be unoptimal. Masking is not possible due to the lazy nature not providing defined array shapes, and early stopping is also not meaningful if the check is not calculated on the spot.

Dask-Distributed on Strato

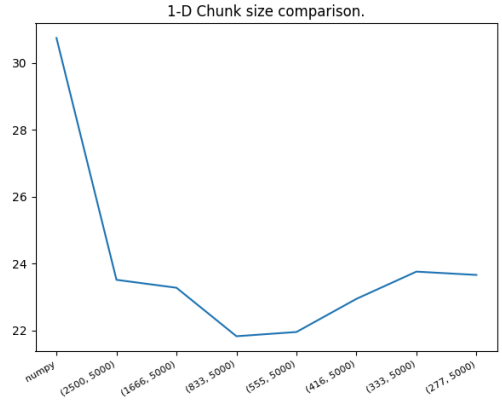
The main script used for distributed computing includes the base numpy implementation as well as a distributed across different chunk sizes. It was run both on Strato, using 4 instances of 4 VCPU cores each, and locally. Both results are from a 5000x5000 grid, run for 100 iterations without early stopping. The y axis displays time in [s].

In the local implementation, the processing time hits a minimum at 833x5000 which corresponds to 6 chunks, equal to the number of available cores. As the chunk size decreases, more overhead is introduced. In the Strato implementation we have access to a total of 16 cores. However, the speed does not seem to be increasing after 9 chunks. Instead, it remains relatively the same all the way up to 18 chunks. That can potentially have something to do with network introduced bottlenecks.

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
worker3	Ubuntu 20.04 (Focal Fossa)	10.92.1.218	AAU.CPU.d.4-12	VasLaptop	Active	AAU	None	Running	0 minutes	Create Snapshot
worker2	Ubuntu 20.04 (Focal Fossa)	10.92.1.247	AAU.CPU.d.4-12	VasLaptop	Active	AAU	None	Running	0 minutes	Create Snapshot
worker1	Ubuntu 20.04 (Focal Fossa)	10.92.1.185	AAU.CPU.d.4-12	VasLaptop	Active	AAU	None	Running	14 minutes	Create Snapshot
scheduler	Ubuntu 20.04 (Focal Fossa)	10.92.1.192	AAU.CPU.d.4-12	VasLaptop	Active	AAU	None	Running	24 minutes	Create Snapshot



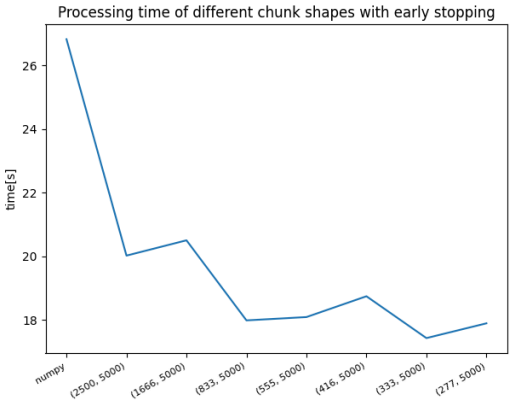
Results on Strato

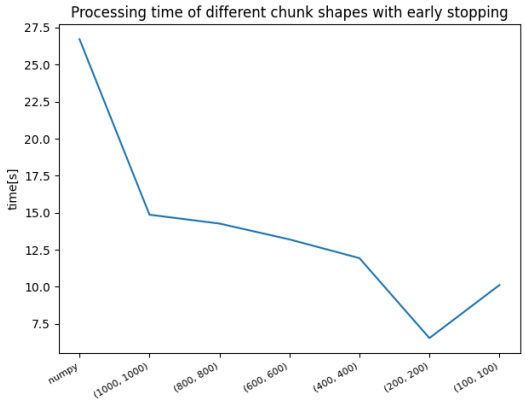


Results on local machine

Optimizations

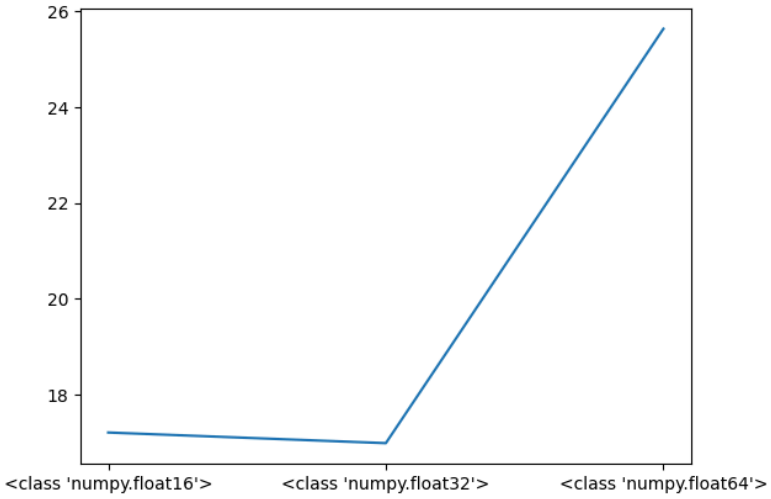
More chunk sizes were tested locally, with the addition of an early stopping optimization, that stops the iteration of a chunk when all its elements have reached the threshold. We can see here that with the addition of early-stopping the processing time does not increase after 6 chunks but continues decreasing until a very small chunk size. That can be explained by the fact that a smaller chunk will exit the iterations added by the optimization much faster than a larger one. At the same time there is a huge dip at chunk_size=(200,200) which can be explained by the chunk fitting fully in the L2 cache (3,0 MB).





Data types

The final experiment involved checking the difference in performance between different data types. Specifically, float16, 32 and 62 were used which resulted in complex 32, 64 and 124 respectively. The mandelbrots were plotted for all of them to make sure that there was no change in the actual result. The processing times are visualized bellow. float16 and float32 require the same amount of time, that can be justified by the fact that the cpu has single precision processing units and not half precision, making their processing the same. As for the float64 it is expected to be slower just due to size.


26
24
22
20
18
<class 'numpy.float16'>
<class 'numpy.float32'>
<class 'numpy.float64'>