

Large Scale Computation of Quantiles using **MELISSA**

A. Ribés¹, T. Terraz², Y. Fournier¹, B. Iooss¹ and B. Raffin²

EDF Lab¹

INRIA Grenoble²

Context

Exascale machines will shortly become a reality. But so far only very few applications are able to take benefit of the level of parallelization these machines provide. Ensemble runs may require to execute from thousands to millions of the same simulation, making it an extremely compute-intensive process that will fully benefit from those machines. However, the large amount of data generated is a strong I/O bottleneck if these intermediate data are saved to disk.

An alternative consists in processing these data in transit as proposed by the Melissa framework. But this approach requires one-pass data processing algorithm to limit the amount of memory needed. We propose a one-pass algorithm for quantiles, relying on the stochastic quantile algorithm of Robbins-Monro. We led a massive experiment executing 80000 parallel simulations on a 6M cell mesh, processing on-line 288TB of data that thus did not need to be saved to disk. Because we bypassed the I/O bottleneck, we were able to compute ubiquitous quantiles, i.e. at all cells and timesteps.

MELISSA

To manage the simulation runs as well as the in transit computation of iterative statistics, we developed a full framework, called Melissa, built around a fault tolerant parallel client/server architecture. Melissa (Modular External Library for In Situ Statistical Analysis) is an open source framework that relies on a three tier architecture (Fig. 1):

<https://melissa-sa.github.io>

The Melissa Server aggregates the simulation results and updates iterative statistics as soon as a new result is available. The Melissa clients are the parallel simulations, providing their outputs to the server. Melissa Launcher interacts with the batch scheduler and server, for creating, launching, and supervising the server and clients. Once a simulation starts, it connects to the server and forwards the output data available at each timestep (or at a sufficiently resolved sampling in time) for updating statistics. These data can then be discarded.

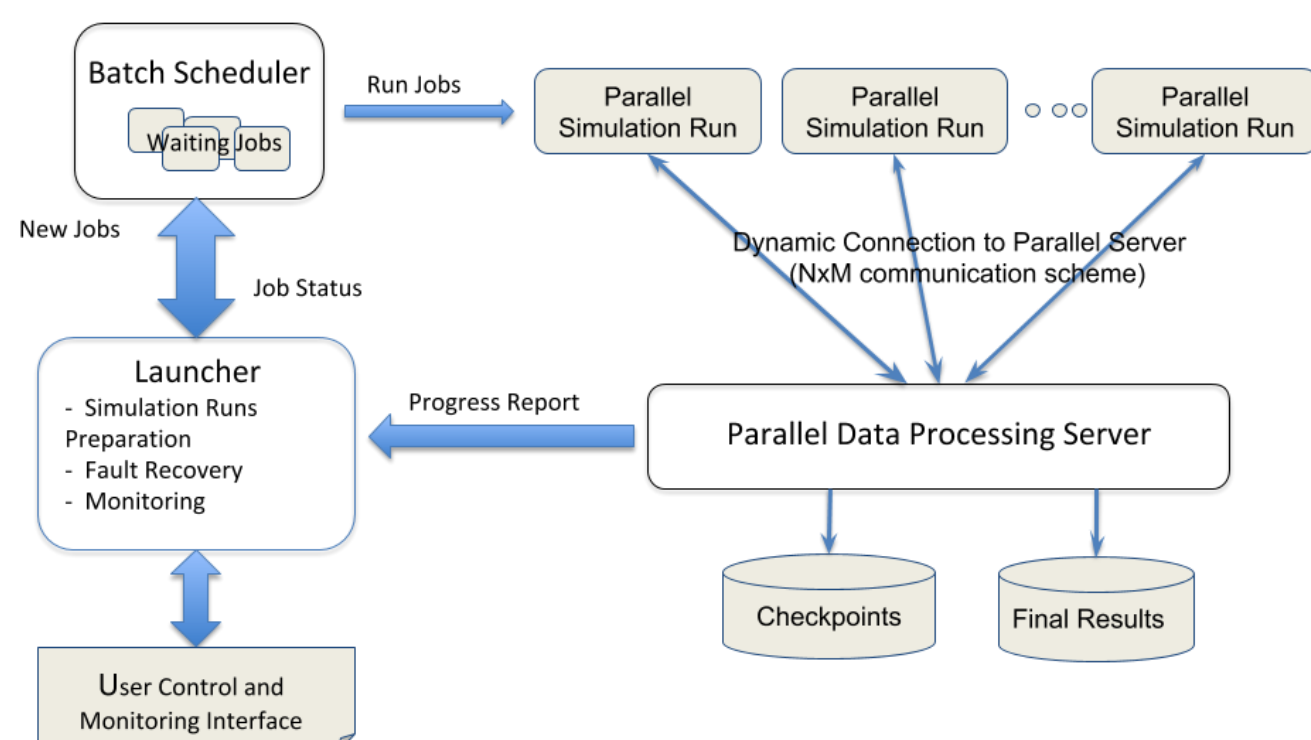


FIGURE 1

The launcher oversees the execution in tight link with the batch scheduler. The job scheduler regulates the number of simulation jobs to run according to the machine availability, leading to an elastic resource usage. The parallel server, started first, process incoming data as soon as received from the connected simulations. A fault tolerance mechanism automatically restarts the failing simulation runs or a failing parallel server.

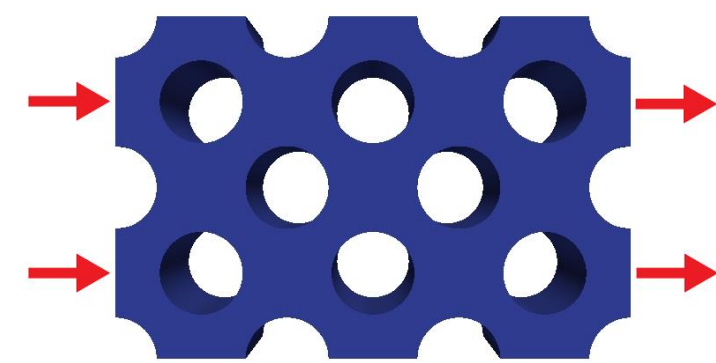


FIGURE 2

Use case: water flows from the left, between the tube bundle, and exits to the right.

Experimental results

We validated our implementation on a fluid mechanics use case simulating a water flow in a tube bundle (fig. 2). The mesh is composed of 6002400 hexahedra. We generated a **multi-run sensitivity study** by simulating the injection of a tracer or dye along the inlet, with 2 independent injection surfaces, each defined by three varying parameters:

1. dye concentration on the upper inlet,
2. dye concentration on the lower inlet,
3. width of the injection on the upper inlet,
4. width of the injection on the lower inlet,
5. duration of the injection on the upper inlet,
6. duration of the injection on the lower inlet.

The results presented on this poster ran on Occigen, ranked 70th at the top500.org of June 2018. We led a massive experiment executing **80 000 parallel simulations on a 6M cell mesh**, processing on-line **288TB of data** that thus did not need to be saved to disk. Because we bypassed the I/O bottleneck, we were able to compute ubiquitous quantiles, i.e. at all cells and timesteps.

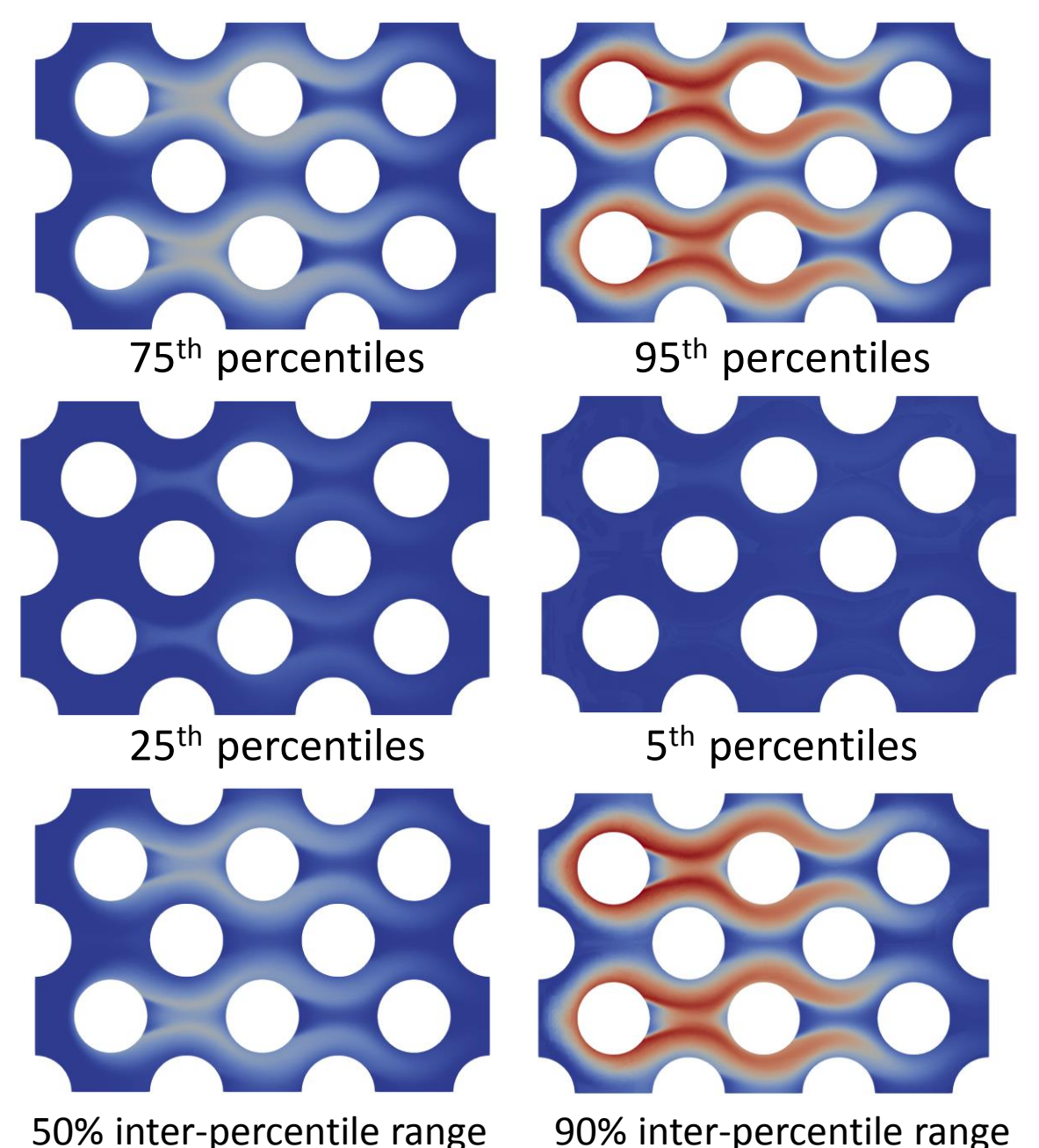


FIGURE 3

Percentiles and inter-percentile ranges maps on a slice of the mesh at timestep 80. The top line corresponds to the percentiles while the bottom line corresponds to inter-percentile ranges. All maps share the same scale, between zero (blue) and maximum (red).

$$q_{n+1} = q_n - \frac{1}{n\gamma} (1_{Y_{n+1} \leq q_n} - \alpha)$$

Robbins-Monro estimator