Internship:

Python Data Processing on Supercomputers for Large Parallel Numerical Simulations

Context

The field of high-performance computing has reached a new milestone, with the world's most powerful supercomputers exceeding the exaflop threshold. These machines will make it possible to process unprecedented quantities of data, which can be used to simulate complex phenomena with superior precision in a wide range of application fields: astrophysics, particle physics, healthcare, genomics, etc. In France, the installation of the first exaflop-scale supercomputer is scheduled for 2025. Leading members of the French scientific community in the field of high-performance computing (HPC) have joined forces within the PEPR NumPEx program (**https://numpex.irisa.fr**) to carry out research aimed at contributing to the design and implementation of the machine's software infrastructure. As part of this program, the Exa-DoST project focuses on data management challenges. This thesis will take place within this framework.

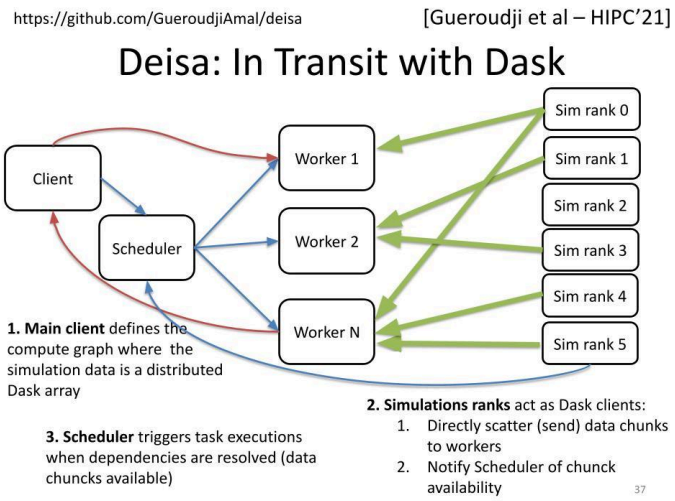
Without a significant change in practices, the increased computing capacity of the next generation of computers will lead to an explosion in the volume of data produced by numerical simulations. Managing this data, from production to analysis, is a major challenge.

The use of simulation results is based on a well-established calculation-storage-calculation protocol. The difference in capacity between computers and file systems makes it inevitable that the latter will be clogged. For instance, the Gysela code in production mode can produce up to 5TB of data per iteration. It is obvious that storing 5TB of data is not feasible at high frequency. What's more, loading this quantity of data for later analysis and visualization is also a difficult task. To bypass this difficulty, we choose to rely on the in-situ data analysis approach.

In situ consists of coupling the parallel simulation code, Gysela, for instance, with a data analytics code that processes the data online as soon as they are produced. In situ enables reducing the amount of data to write to disk, limiting the pressure on the file system. This is a mandatory approach to run massive simulations like Gysela on the latest Exascale supercomputers.

We developed an in situ data processing approach called Deisa, relying on Dask, a Python environment for distributed tasks. Dask defines tasks that are executed asynchronously on workers once their input data are available. The user defines a graph of tasks to be executed. This graph is then forwarded to the Dask scheduler. The scheduler is in charge of (1) optimizing the task graph and (2) distributing the tasks for execution to the different workers according to a scheduling algorithm aiming at minimizing the graph execution time.

Deisa extends Dask, so it becomes possible to couple an MPI-based parallel simulation code with Dask. Deisa enables the simulation code to directly send newly produced data into the worker memories, notify the Dask scheduler that these data are available for analysis, and that associated tasks can then be scheduled for execution.



Compared to previous in situ approaches that are mainly MPI-based, our approach relying on Python tasks makes for a good tradeoff between programming ease and runtime performance.

Problematic

When discussing in-situ data analysis, two primary techniques are often highlighted: **in-transit analysis** and **in-process analysis**.

In-transit analysis involves examining data while it is being transferred between systems or across various components of a distributed architecture. For instance, in large-scale simulations or scientific experiments, data is typically generated on one system (such as a supercomputer) and needs to be sent to another system for storage or further analysis. Rather than waiting for the data to reach its final destination, in-transit analysis allows for computations to be performed on the data as it moves. This approach significantly reduces overall processing time.

In contrast, in-process analysis entails analyzing data during its generation or processing by the application. Instead of waiting for an entire simulation or data generation task to finish, this technique enables concurrent processing of data throughout the ongoing task, such as during simulation steps in a scientific application. By doing so, the burden of post-processing is alleviated, as computational tasks are distributed over time.

To illustrate these techniques, consider the Gysela code. Our goal is to integrate both in-transit and in-process analyses to enhance data analytics while minimizing data transfer between systems. A common diagnostic performed on Gysela data is the global aggregation of certain fields across the entire domain. This global operation can be divided into a subdomain reduction followed by a reduced global reduction. By executing the initial reduction directly on the process where the data is generated, we can significantly decrease the volume of data transferred. This, in turn, alleviates the load on the parallel file system.

However, determining which reductions should be performed on specific resources presents a challenge, especially since we often lack prior knowledge about the types of diagnostics

that will be required. This highlights the concept of co-scheduling. In this context, co-scheduling refers to the coordinated execution of in-transit and in-process data analysis tasks to optimize resource efficiency and minimize data movement latency. By aligning the scheduling of these two processes, the system can ensure more effective utilization of resources, such as network bandwidth, CPU, and memory. This approach is particularly vital for large-scale applications, where traditional methods of moving and analyzing massive datasets can lead to significant bottlenecks.

Mission

Before putting in place a solution to automatically manage the separation of local reductions from the workflow, we need to check whether or not the overall performance can be improved by executing the local reductions in-process. The candidate will consider an artificially generated workflow in which one has the possibility to isolate local operations from the global task graph. Next, he/she needs to manually assign those local operations to be performed on the same process as the application. The local results will then be aggregated to dedicated resources for the final results. The candidate will be in charge of the performance evaluation of the whole workflow.

The successful completion of this internship could lead to a 3-year thesis, during which you will further explore the concepts already covered and conduct research work, notably the automation of co-scheduling in-situ and in-process tasks.

Main activities

After studying the state of the art, getting to grips with the architecture of PDI and Deisa, and getting familiar with the Dask environment, the candidate will study, propose, and develop innovative solutions, which he or she will publish in the best journals and conferences in the field. Within the Exa-DoST project of the NumPEx PEPR, the candidate will have privileged access to very large-scale computers for experiments. The framework developed will be tested on large-scale applications with close collaboration with CEA/DAM or/and CEA/DES. The candidate will be based at Maison de la Simulation, in close collaboration with the teams of specialists in high-performance computing and simulation at Inria Grenoble.

Technical skills

- An excellent Master degree in computer science or equivalent
- Strong knowledge of distributed systems
- Knowledge on storage and (distributed) file systems
- Ability and motivation to conduct high-quality research, including publishing the results in relevant reviews
- Strong programming skills (Python, C/C++)
- Working experience in the areas of HPC and Big Data management is an advantage
- Very good communication skills in oral and written English
- Open-mindedness, strong integration skills and team spirit

References

1. Dask - https://www.dask.org/
2. Deisa Paper: Dask-enabled in situ analytics. Amal Gueroudji, Julien Bigot, Bruno Raffin. Hipc 2021. https://hal.inria.fr/hal-03509198v1
3. Deisa Paper: Dask-Extended External Tasks for HPC/ML In Transit Workflows, Amal Gueroudji, Julien Bigot, Bruno Raffin, Robert Ross. Work workshop at Supercomputing 23. https://hal.science/hal-04409157v1
4. Deisa Code: https://github.com/pdidev/deisa
5. Ray - https://github.com/ray-project/ray
6. Damaris: How to Efficiently Leverage Multicore Parallelism to Achieve Scalable, Jitter-free I/O. Matthieu Dorier , Gabriel Antoniu , Franck Cappello, Marc Snir , Leigh Orf. IEEE Cluster 2012. https://inria.hal.science/hal-00715252