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.

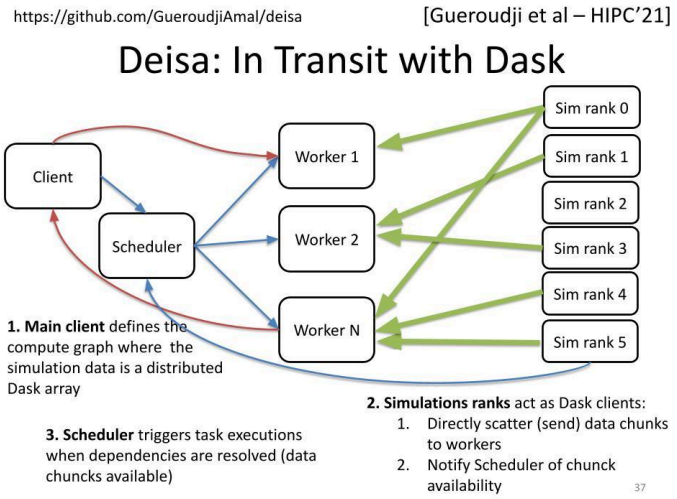
Without a significant change in practices, the increased computing capacity of the next generation of computers will lead to an explosion in the volume of data produced by numerical simulations. Managing this data, from production to analysis, is a major challenge.

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.

The goal of this PhD work is to investigate solutions to:

- Improve task placement and thus performance enabling tasks to be scheduled in process (into the simulation processes), in situ (running on external processes but on the same compute nodes that also run the simulation code), and in transit (on dedicated nodes different from the simulation nodes). Running closer to the simulation reduces the need for data movements but can potentially steal resources (CPU, GPU, network, memory, cache) from the simulation and slow it down. Dask task graph optimization is a good starting point to develop such approaches.
- Enable more diverse and flexible data processing patterns for Dask in situ:
    - data processing tasks are triggered when detecting some specific events in the data;
    - changes to some simulation internal parameters during runtime as a result of certain analytics tasks.
    - enabling task graphs combining classical analytics with deep neural networks-based analysis.

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

The candidate will begin the thesis by conducting a comprehensive study of the state-of-the-art in relevant areas, focusing on in-situ, in-transit, and in-process data analysis. Early on, they will gain proficiency in using PDI Deisa and familiarize themselves with the Gysela code.

To dive into the core subject of the thesis, the candidate will examine how to separate local data reduction from the overall workflow. They will analyze the task graph generated by Dask, the underlying library of Deisa, and conduct a static analysis to determine which tasks should be executed in-process. Applying graph theory will be crucial in this stage to identify the appropriate tasks.

Once the local tasks are defined, the candidate will implement routines within the PDI Deisa plugin to handle these local operations in the same process as the simulation. In the final phase, they will expose the locally reduced data to Deisa's dedicated I/O processes using remote procedure calls, facilitating the aggregation of data for final reduction.

Additionally, the candidate will investigate solutions to automate the above processes, ensuring that compute resources are efficiently scheduled based on the workload. The ultimate goal will be to optimize the entire workflow, improving performance and resource management.

Main activities

The candidate will undertake the thesis work by thoroughly studying existing research and developing innovative solutions to efficiently manage the integration of in-transit and in-process data analysis. This research will address critical challenges in optimizing data

workflows, and the novel solutions devised are expected to significantly enhance performance in large-scale computing environments. The outcomes of this work will be submitted for publication in top-tier journals and presented at prestigious conferences within the high-performance computing (HPC) community.

As part of the Exa-DoST project within the NumPEx PEPR initiative, the candidate will have privileged access to cutting-edge, large-scale computing infrastructure, enabling robust experimentation and testing. The framework developed will be rigorously evaluated on large-scale applications in close collaboration with renowned research entities such as CEA/DAM and/or CEA/DES. These collaborations will provide the candidate with opportunities to work on real-world HPC challenges at the frontier of scientific research.

The candidate will be based at Maison de la Simulation, working closely with interdisciplinary teams of experts in HPC and advanced simulation from Inria Grenoble, ensuring a dynamic and supportive research environment. This collaborative setting will foster innovation and ensure that the research contributes to the state of the art in both academic and industrial contexts.

Technical skills

- An excellent Master's degree in computer science or equivalent
- Strong knowledge of distributed systems
- Knowledge of 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

Benefits

- Subsidized meals
- Up to 75% reimbursed public transport
- Possibility of teleworking and flexible working hours
- Professional equipment available (videoconferencing, loan of computer equipment, etc.)
- Social, cultural, and sports benefits
- Access to professional training
- Social security
- Up to 9 weeks of paid leave

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

Annexe