RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

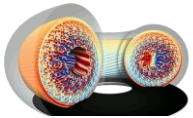
cea cnrs Inria

FRANCE 2030
PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

ExaDoST - Work Package 1 Breakout Session with Gysela

WP Leaders:
Francieli Boito (Université de Bordeaux) &
François Tessier (Inria Rennes)

The Gysela case study for WP1



- Ckpt time x3 with interference
- From 500k cores:
  - ~10TB of data written
  - 25k files
  - 30’ min ckpt time (5GBps)

Goal: Improve checkpointing performance

1. To improve the way Gysela writes the checkpoint files
   - This started during Méline Trochon's internship - some promising results but did not explore HDF5 compression and sub-files.
     - Some improvement may be required in PDI
     - **We'll hire an intern to continue this work (Inria TADaaM)**
   - Run it with TOTO to profile the low-level I/O behavior caused by different parameters
   - Asynchronous checkpointing could be explored now (with the C++ version)
     - Stage data in the CPU memory
     - Or I/O directly from GPU to the PFS?
     - **Damaris plugin for PDI? VeloC plugin for PDI?**

The Gysela case study for WP1

Goal: to improve checkpointing performance

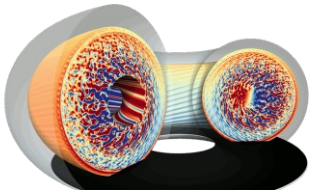
2. To mitigate contention in the access to the shared storage
   - Topic of Méline Trochon's PhD thesis (DDN/Inria)
   - Requirement: I/O benchmarks/mini-apps
     - Initial analysis - to characterize interference - using IOR benchmark to mimic checkpointing
     - **We will create and distribute a Gyselalib++-based I/O benchmark that only performs the checkpoint phases (similar to US-developed HACC-IO, VPIC-IO, etc…)**
   - **The idea: to use information from the file system to decide when to checkpoint**
     - Implementation is not yet clear, could be a PDI plugin.
     - some OST placement things will be done in TOTO
   - **Benefit Gysela but also any application that checkpoints!**

Detailed comments

Motif inspired by Gysela (from our previous AG)

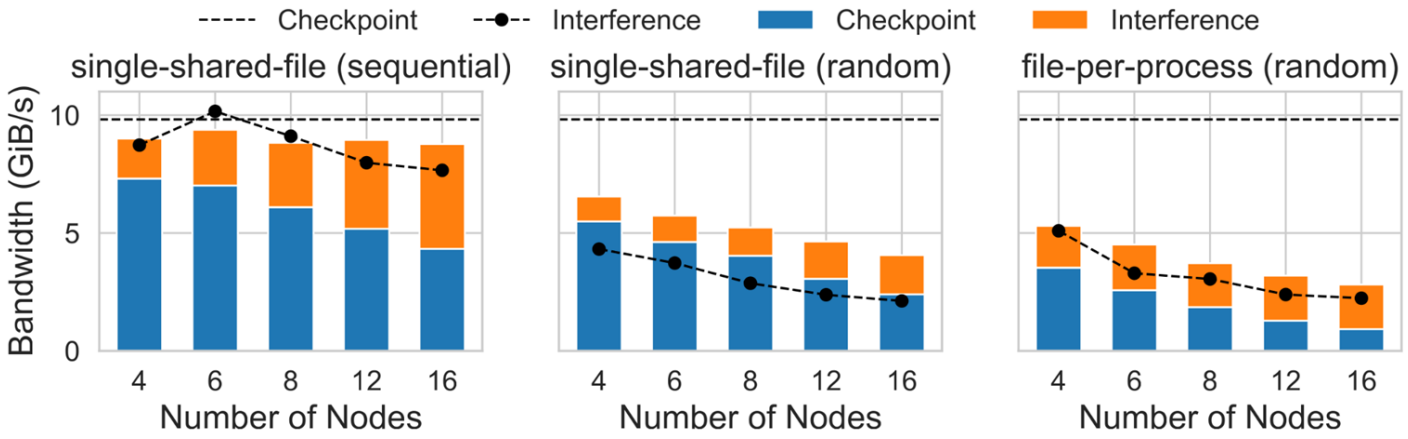
The application periodically writes a checkpoint file, that is used for fault tolerance reasons only. The very last checkpoint file is also used to restart a next simulation, so it is mandatory. The application runs with a few processes per node (one per GPU or per NUMA domain, for example), and the amount of data written corresponds to ¼ of the node memory (each process in the node will write a part of it). The data is not contiguous in memory, but contiguous writes could be done with the cost of writing more data (including “ghost cells”). Processes may have the same amount of data or not.

- the application uses all the node RAM, so solutions that require memory copies are not acceptable
- to dedicate compute resources for I/O could not be possible right now, but maybe in the future when using GPU (CPU resources could be used to advance I/O)
- they would like to be able to restart with a different number of processes
- predictability of checkpointing time at the end is an issue
- hundreds of TB to a few PB of checkpointing data at exascale



The impact of concurrent I/O on checkpointing

- Large experimental campaign to characterize the impact of concurrent jobs on checkpoint operations
- Found that shared-file patterns tend to receive less of the available bandwidth
- Identified "toxic" patterns that degrade the global system bandwidth



Méline Trochon, Jean-Thomas Acquaviva, Francieli Boito, Brice Goglin, François Tessier, Luan Teylo. ***On the Impact of Interference from Concurrent Jobs on Checkpointing Performance****, pre-print, 2025.*

Ongoing Work and Discussion Topics

- Continue the investigation of interference
  - Checkpointing time 3 times longer with concurrent jobs
  - Ex: from 500'000 cores, the checkpointing phase takes ~30 minutes for writing ~10TB of data into 25k files (~5,5 GB/s)
  - Objective: Test other infrastructures, metadata-intensive operations
- For experiments on I/O (including interference), we use IOR benchmark to mimic the Gysela motif
  - We can already compile/run Gysela (using Giux)
  - New C++-based version of Gysela
  - Objective: Develop a Gysela I/O kernel based on Gyselalib++ to reproduce the I/O behavior of the app
    - Make a 5D or 6D case (4D case already available). Increasing dimensions makes checkpoints larger. Also, PDI has to be configured carefully so the processes will write contiguously
    - We could advertise it as a FR/EU I/O kernel for the I/O community

- In what does the new C++ version differ?
  - Gysela uses the whole memory of the system, so asynchronous checkpoint was not an option in the Fortran (CPU-based only) version, but it is now with the C++ version.
  - Could the checkpoint go from GPU memory directly to the file system? See GPUDirect Storage by NVidia
- New storage tiers could help
  - Node-local storage could be used for staging data before checkpointing but no that common on recent supercomputers
  - Will Alice Recoque feature intermediate storage tiers? Xavier (GENCI): “Count on ephemeral storage.”
- PDI + checkpointing: an API that stores a pointer to the data then use plugins to store it
  - Gysela uses PDI to store data already
  - PDI arbitrates between plugins to keep data consistency
  - There is an FTI plugin, but it is no longer maintained so we need something else. An option is using Damaris plugin to copy data into memory and then doing I/O asynchronously. A VeloC plugin could also be an option.
  - Objective: Study of the best option to delegate the checkpoint/restart process (work with the Gysela, PDI and Damaris teams)
  - Todo: Talk with Amina Guermouche (Inria Bordeaux) about asynchronous checkpointing

- Data format
  - Use both HDF5 and netCDF
  - Gysela moved to netCDF which is better for integration with Dask and post-execution analysis
- Data layout, sub-filing strategy, compression, … [Méline's internship work]
  - An important thing was the number of files being generated by directory
    - sub-filing from HDF5 could help - from the application point of view, there is a single file, but HDF5 creates many actual files
  - Compression is one of the things to be explored to accelerate I/O and reduce storage footprint
    - HDF5 is able to do that, but Méline had problems making it work (compatibility, compiler version, etc.).
    - One can have a double precision data structure and declare the HDF5 dataset as a simple precision one.
  - Todo: Hire an intern to explore that

- Tracing memory usage of I/O would be interesting to see how it behaves,
  - Very hard to trace memory usage by HDF5: logically it allocates huge memory spaces although it is not actually used. A tracer would see a huge memory usage by HDF5 which would be inaccurate.
  - If we can fix the previous point: Can be extended to netCDF as well, as it uses HDF5. As long as the APIs expose the option…
- Use of ExaDoST-developed tools
  - TOTO: can profile the I/O behavior according to different parameters
  - Objective: do experiments with Gysela to explore the space of parameters for PDI, HDF5, etc., That could give us insights of their impact and help explain the poor I/O performance of Gysela
  - Todo: write an internship proposal (work at TADaaM)

Notes during the meeting (November 5 2025)

checkpointing taking 30minutes

~10TBs per checkpoint

how many times? at the very least 1, at the end of simulation, for a 10h simulation, they do every hour or so

asynchronous checkpointing? maybe in the C++ version

10TB in 30minutes is only ~5 GiB/s, so where is the bottleneck? a huge number of files, apparently, from the results of Méline's internship.

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

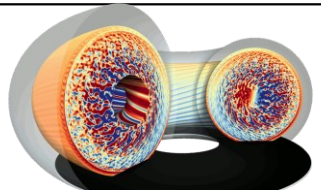
cea cnrs Inria

FRANCE 2030
PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

ExaDoST - Work Package 1
Breakout Session with Gysela from 2024

WP Leaders:
Francieli Boito (Université de Bordeaux) &
François Tessier (Inria Rennes)

Session 2 (Gysela) notes



- original: checkpoint file is file-per-process
  - 22784 mpi ranks -> same number of files, total of 16.2 TB for the largest scale they tested
  - done with PDI
  - they moved to parallel HDF5 to do shared-file
- there is also some output that is saved, they want to compress it smartly with AI
- contacts: Dorian Midou and Kevin Obrejan
- not one process per node (as in Julien's description), but a few MPi ranks per node, and they all write. There will typically be one rank per gpu, or one rank per NUMA domain. In total the distribution function uses ¼ of te node RAM, if there are multiple processes they will each write a portion of it.
- ¼ memory in total, may be gpu
- 5d array of double (they call it the "distribution function"
- each process gets an equal size, but it may not be the case in the future
- this happens every N timesteps
- keep two sets of files
- synchronous, don'twant to dedicate compute resources fr I/O, maybe for gpu. they tried with fti and a house-made solution, it was not worthy it. even using a local ssd was not worthy it
- minimum is one checkpointing at the end, the rate depends on the time between failures. it is the start of a new simulation to save time, starting already from a certain steady state
- it can be any format
- it'sn ot possible to restart with a different scale, but they would like to
- ghost cells: for now it's like 2 every 32 points for each dimension, but it will be different in future versions (not clear how it will be)
- there are always at the limit of memory, no memory copies are tolerated
- the application works with two versions of the 5d array, updating one at atime. So that means that if there was a way of doing asynchronous I/O, that would be possible without memory copy, because the array that is being written is not going to be modified for a sub-timestep. But that is quite a small time scale, a few seconds maybe
- the 5d data is just for checkpointing, it's never ever used for anything else (they make 3d reduced data they write and it's very fast, and they use that for analysis)

Notes during the session

New functionality in slurm where we can send a signal to a job saying they should stop

wp2 proposes that checkpointing happens as a plugin for pdi/damaris, so it can use the same core that is already dedicated for analytics. That wll prevent a situation where there are a thousando dedicated cores to a thousand different functionality

in this sense it would be better to take fti (that is no longer supported) as a part of the pdi project, rather than using veloc