RÉPUBLIQUE
FRANÇAISE
Liberté
Égalité
Fraternité

From I/O optimisation to workload-aware adaptative checkpointing

Méline Trochon, Francieli Boito, Brice Goglin, François Tessier,
Jean-Thomas Acquaviva- Inria, Bordeaux, DataDirect Network

Summary

Introduction : HPC (High Performance Computing)

- Computes heavy simulations (weather forecast, nuclear, physics related)
- High complexity (type of compute nodes, storage devices) and increase software stack
- From regional cluster (around 50 nodes) to national cluster (few thousand nodes)

Figure: Joliot Curie - TGCC/GENCI

NumPEx national project or where the funding come from

ALICE RECOQUE
BEYOND HPC

Figure: Alice Recoque - Computer Scientist (1929-2021)

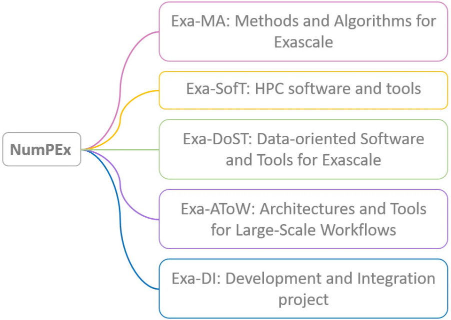


Figure: NumPEx Official website

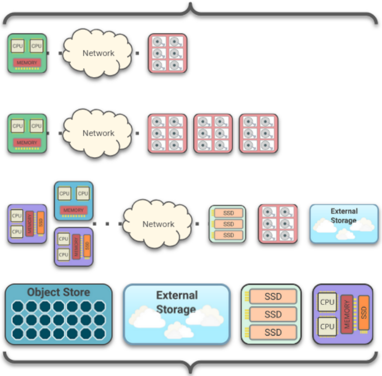
Data movement - Input/Output (I/O)

"Traditional" approach

Horizontal scaling

Vertical scaling

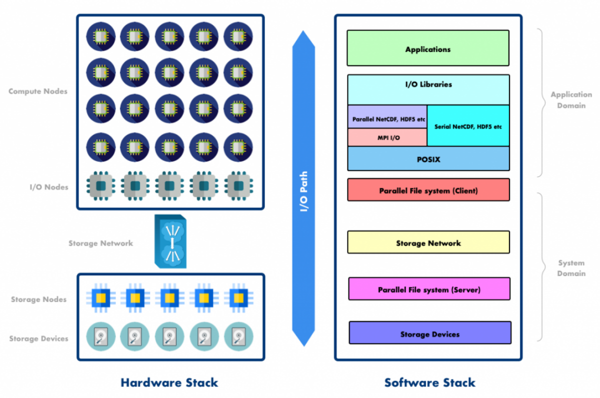
Storage **heterogeneity**



Software stack to
manage all of these

(*Lustre PFL & DNE, DAOS, Mochi, ADIOS2, Damaris, GekkoFS, BeeGFS, RADOS, CHFS…*)

Complexification of I/O path



- Lots of optimisation at several levels
- Several vendors; hard to maintain compatibility
- Difficulties to trace and profile I/O operations
- Bottlenecks can come from different layer (network switch congestion, file system overload)

Figure: Taken from https://pramodkumbhar.com/2020/03/i-o-performance-analysis-with-darshan/

Storage trends

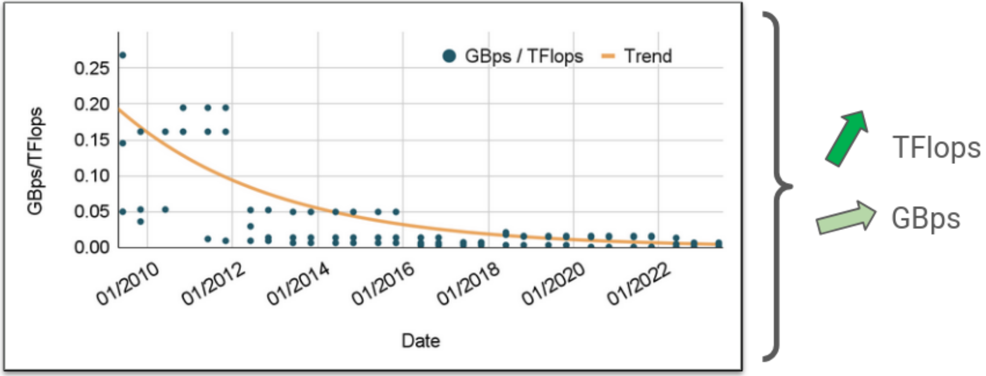


Figure: Extracted from the Top500 over the past 15 years

02

Checkpointing optimisation of a plasma turbulence simulation code

The GYSELA simulation code

- GYrokinetics Semi-Lagrangian simulation code
- The checkpoint mechanism takes a lot of time, and scales badly
- I/O operations done using PDI (Portable Data Interface)

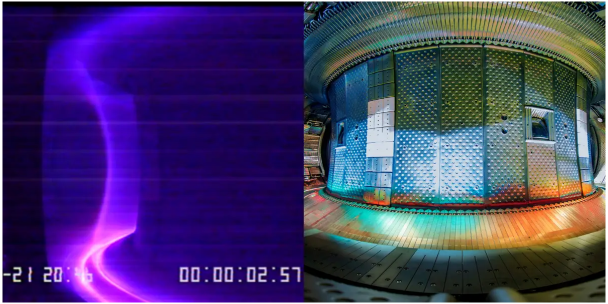
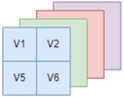


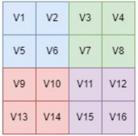
Figure: The WEST tokamak at CEA/IRFM

Checkpointing implementation



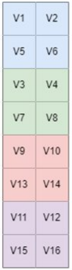
Sync (sequential):
- One file per process (approx. 10 000)
- Dim: local F size

Very fast, but not scalable and stops at exascale



PARA (Collective):
- One file per mu (approx. 100)
- Dim: total F size

Aggregated F in files, small number of files
Take much longer than sequential



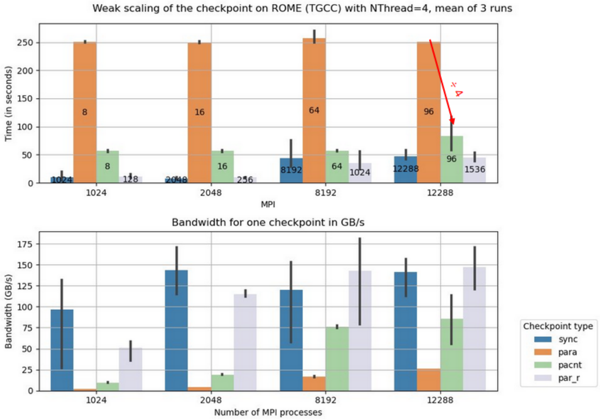
PACNT (Collective contiguous):
- One file per mu (approx. 100)
- Dim: Nproc x local F size

New implementation
Non aggregated F in files but contiguous for the processes
small number of files and fast
Improvement over PARA regarding time

PAR_R (Collective in R)
- One file per mu x nproc_theta (approx. 1 000)
- Dim: Nproc_r x local F size

New implementation
Balanced between file size and collectivity,
Similar performance to SYNC but still generate a lot of files

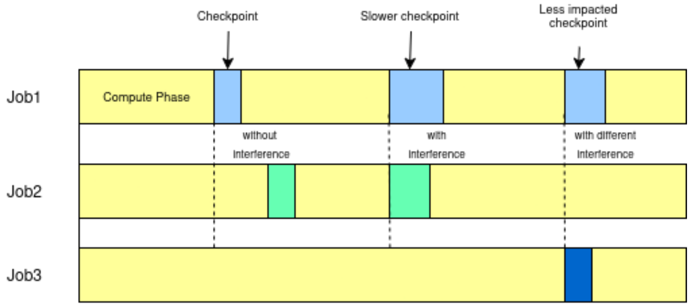
Weak scaling on Irene, up to a thousand nodes



- Tests performed on Rome partition of Irene cluster.
- Found scalable collective implementation
- **Highlights the importance of contiguous I/O operations**

03
Impact of concurrent jobs on checkpointing performance

Interference while performing checkpointing



Usual type of I/O operations

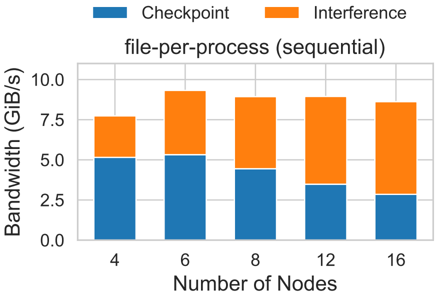
	Partitioning	File strat.	Block size	Transfer size
ckpt	8 x 8	FPP	6 GiB	8 MiB
interf.	4-16 x 4-16	FPP/SSF	3-12 GiB	4KiB - 8 MiB

Table: **Partitioning** corresponds to the number of nodes and processes per nodes. **File strat.** is either file per process (FPP) or single shared file (SSF).

Executions on two systems:

- PlaFRIM (Bordeaux), $\approx$ 40 nodes, interconnection of 100Gbit/s using **BeeGFS**;
- EuLab (an experimental cluster in CINECA), 16 nodes, interconnection of 400 Gbit/s using **Lustre**.

Results - Variating number of nodes



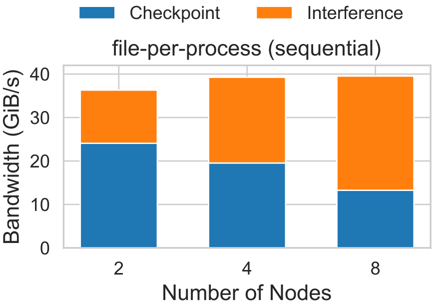


Figure: Bandwidth of `checkpoint` (bottom, blue) and `interference` (top, orange) running together, varying the **number of nodes** used by the `interference` (x-axis). The two applications are otherwise identical.

Results - Variating file strategy on PlaFRIM

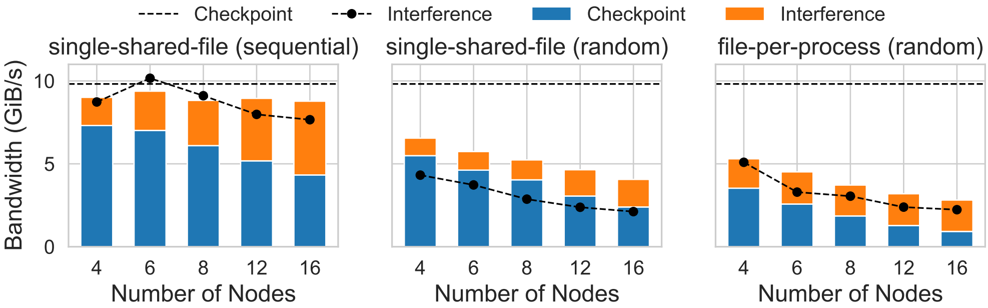


Figure: Tests performed on PlaFRIM. Bandwidth of `checkpoint` (bottom, blue) and `interference` (top, orange) running together, varying the number of nodes used by the `interference` (x-axis), which **writes using different file strategies and spatialities**. The lines depict their performance when running in isolation.

Results - Variating file strategu on Eulab

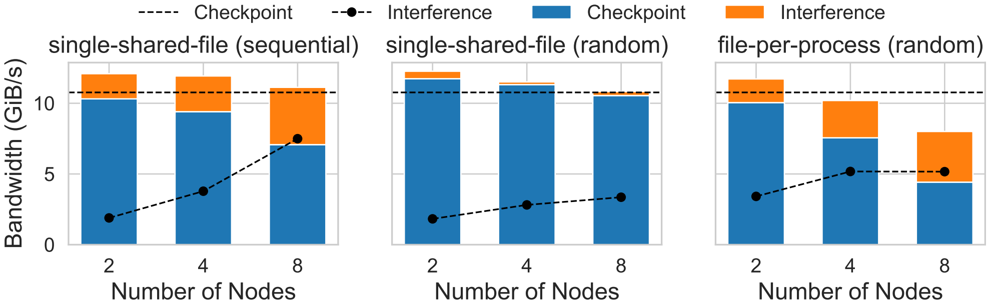


Figure: Tests performed on the eulab using hard disks. Bandwidth of `checkpoint` (bottom, blue) and `interference` (top, orange) running together, varying the number of nodes used by the `interference` (x-axis), which **writes using different file strategies and spatialities**. The lines depict their performance when running in isolation.

Conclusion from benchmark

- Some I/O patterns performs badly on their own but also **delays other I/O** operating at the same time
- The optimal I/O pattern is influenced by the file system overload
- We would like to **take leverage of file system real-time information** before performing the checkpoint.

04
Simulating adaptative checkpointing

What do we know ?

Checkpointing specifities

- Known behavior : Huge contiguous I/O operation
- Temporal flexibility: Checkpointing can be delayed
- Pattern flexibility : The I/O patterns can changed (especially how we store the data)

Get file system knowledge

- Prediction model from previous I/O operations data
- Communication to Lustre via Slurm (using Quality of service variables)
- Communications between different storage targets are costly

The simulator using Simpy

- Replicate HPC cluster I/O environment using logs from previous clusters
- Test and create algorithms to diminish checkpointing 'loss' time
- Implement and test the solution on simulation code

Conclusion

- Data movement will become a bigger bottleneck in future HPC clusters
- Application jobs needs to become more aware of their environment (heterogeneity hardware, shared resources, etc.)

Merci.