DDF Benchmarking at LAB

Systèmes Electroniques et Informatique Instrumentale

SE2I Projects:

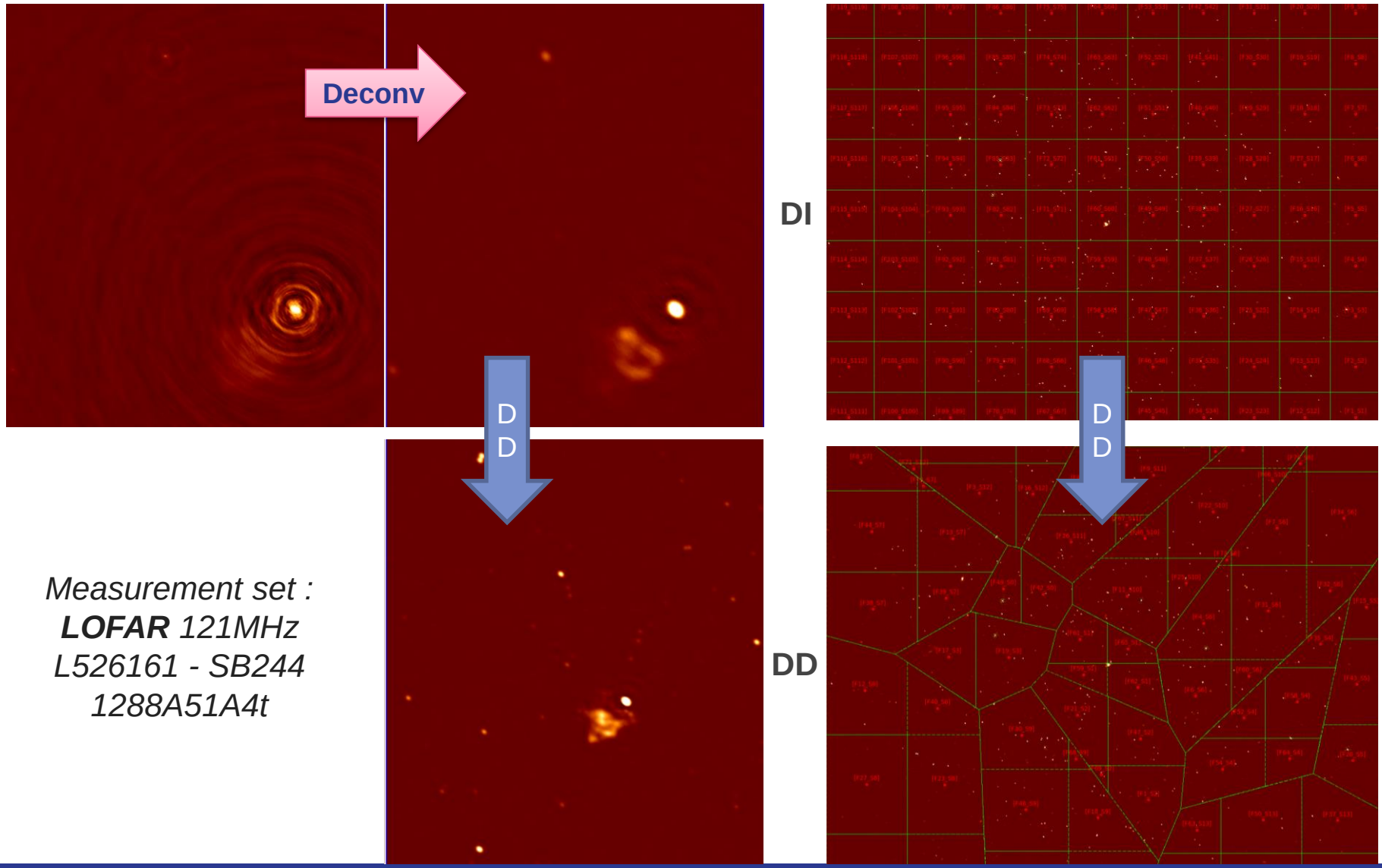
- **Exploring the solar system**

ChemCam (Curiosity), *SuperCam* (Perseverance), *MIRS* (MMX), ExoMars

- **Ground instrumentation**

Atacama Large subMillimeter Array (ALMA + WSU 2030), *SPIAKID*

Example of execution on the ALCOR server



Measurement set :
***LOFAR** 121MHz*
L526161 - SB244
1288A51A4t

Example of execution on the ALCOR server

- CPU AMD EPYC 7H12 64-Core Processor (128 logical cores at 2,6GHz max) + 1 TiB RAM
- Initial use of a Singularity container to port the application to our machine

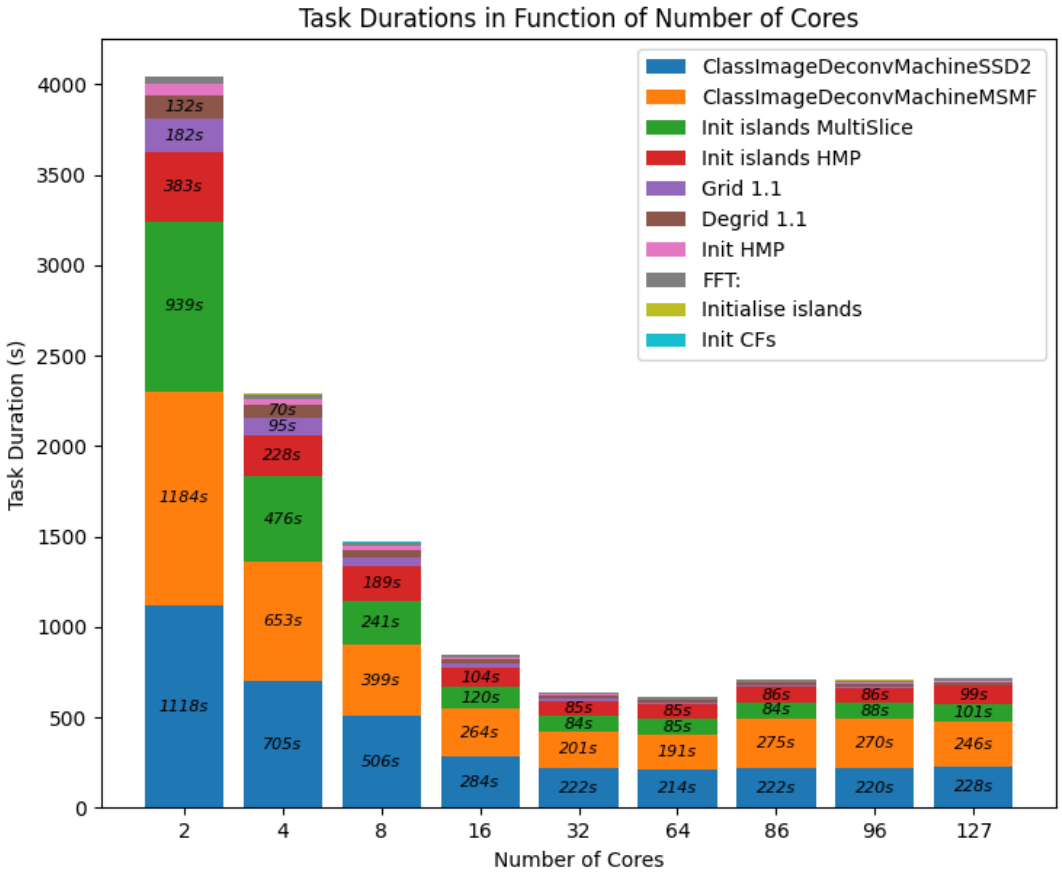
```
[Singularity] root@alcor:$ DDF.py Test.deeper.parset --Output-Name Test.deeper.AP --Cache-Reset 1 --Cache-PSF auto --
ions-DDSols DD0 --DDESolutions-SolsDir None
 - 09:50:45 - AsyncProcessPool             | Fixing parent process to vthread 0
 - 09:50:45 - AsyncProcessPool             | Worker affinities not specified, leaving unset
/usr/local/src/DDFacet/DDFacet/Data/ClassVisServer.py:607: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if DATA["sort_index"] is not None and DATA["Weights"] is not 1:
DDFacet version is 0.6.0.0-1941b8e5
Using python package located at: /usr/local/src/DDFacet/DDFacet
Using driver file located at: /usr/local/src/DDFacet/DDFacet/DDF.py
            _________________           _
            |  _   \  _   \  ___|         | |
            | | |  | | |  | |_  __ _  ___ ___| |_
            | | |  | | |  |  _/ _` |/ __/ _ \ __|
            | |/ /| |/ /| || (_| | (_|  __/ |_
            |___/ |___/ \_| \__,_|\___\___|\__|


 - 09:50:45 - DDFacet                      | starting DDFacet (/usr/local/src/DDFacet/DDFacet/DDF.py Test.deeper.pars
Dirty auto --Predict-InitDicoModel Test.deeper.DicoModel --DDESolutions-DDSols DD0 --DDESolutions-SolsDir None)
 - 09:50:45 - DDFacet                      |    version is 0.6.0.0-1941b8e5
 - 09:50:45 - DDFacet                      |    working directory is /home/antsa/Téléchargements
 - 09:50:45 - DDFacet                      | Successfully read parset Test.deeper.parset, version 0.2
 - 09:50:45 - DDFacet                      | Checking system configuration:
```

- Benchmarked execution

DDF.py Perf.parset --Output-Name Perf-{numcores} --Output-Mode Clean --Deconv-MaxMajorIter 1 --Data-ColName DATA --Parallel-NCPU={numcores}

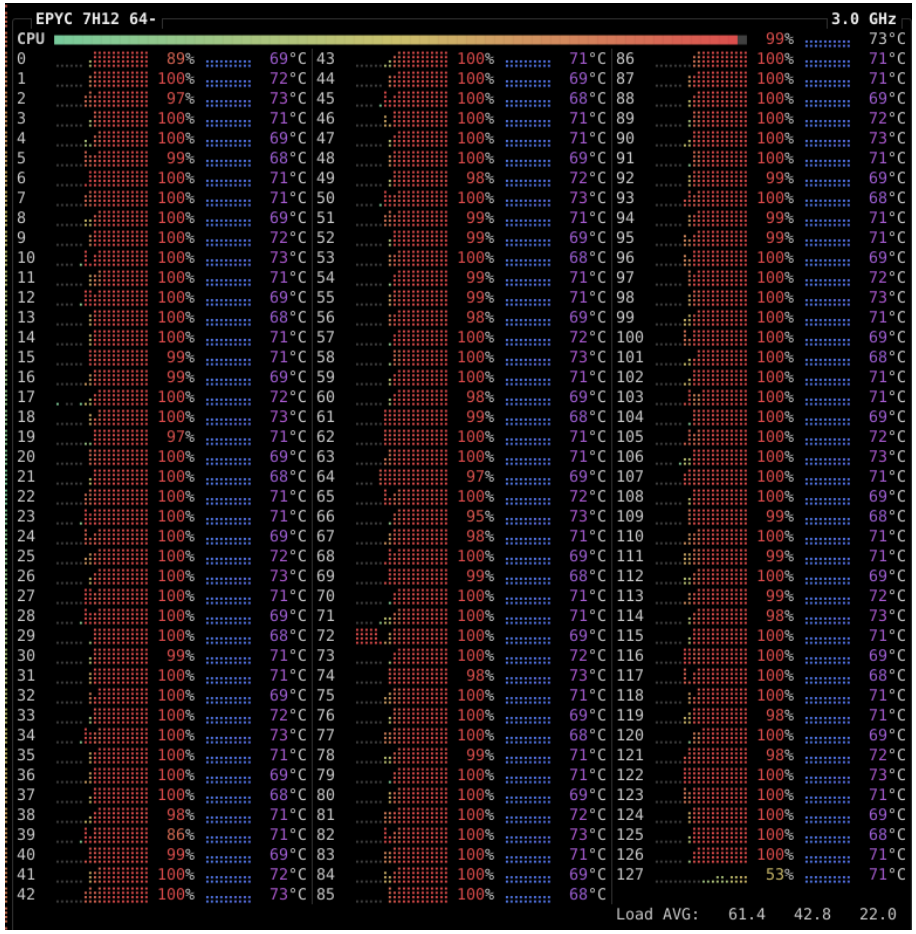
Profiling script (python)

- Use of DDFacet logs to profile the execution time of the various stages
- Performance degradation > 64 cores: NUMEXPR_MAX_THREADS variable?
- Evolution of the duration of the various DDFacet stages as a function of the number of cores



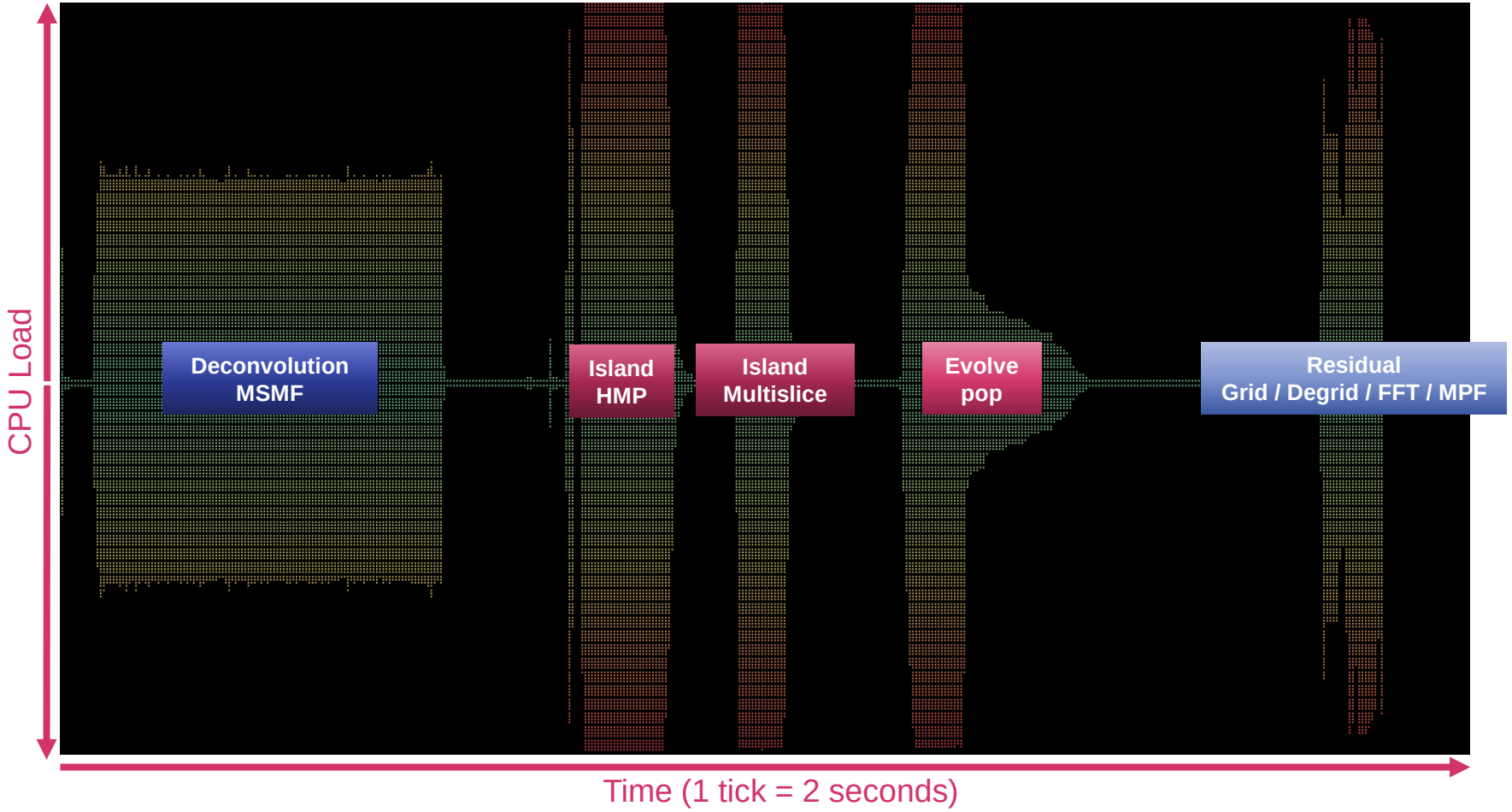
Profiling with Btop

- Exploitation of the 128 logical cores of the CPU during the execution on ALCOR (LAB)
- CPU : AMD EPYC 7H12 64-Core Processor
- Profiling : Btop++ (https://github.com/aristocratos/btop)



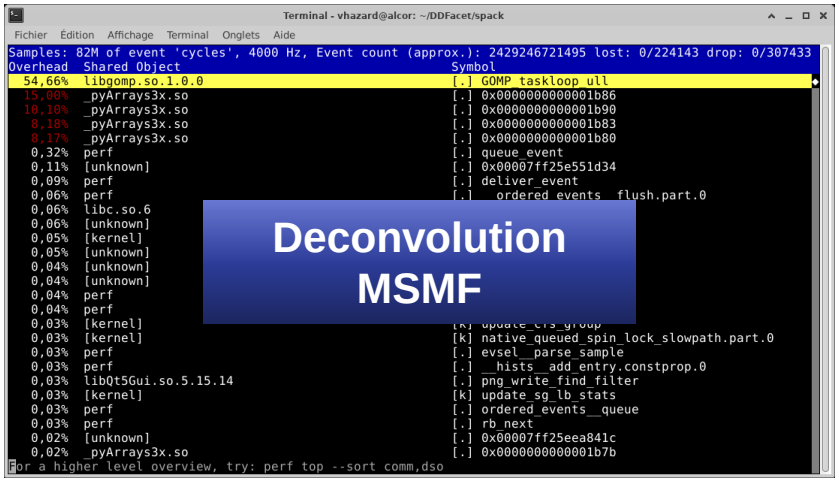
Profiling with Btop

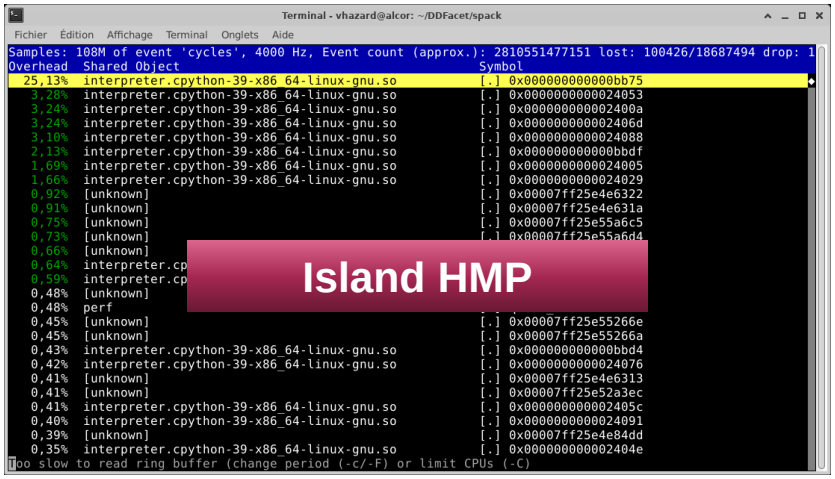
- CPU core utilisation over time
- Labelling the different stages of the DDFacet Pipeline

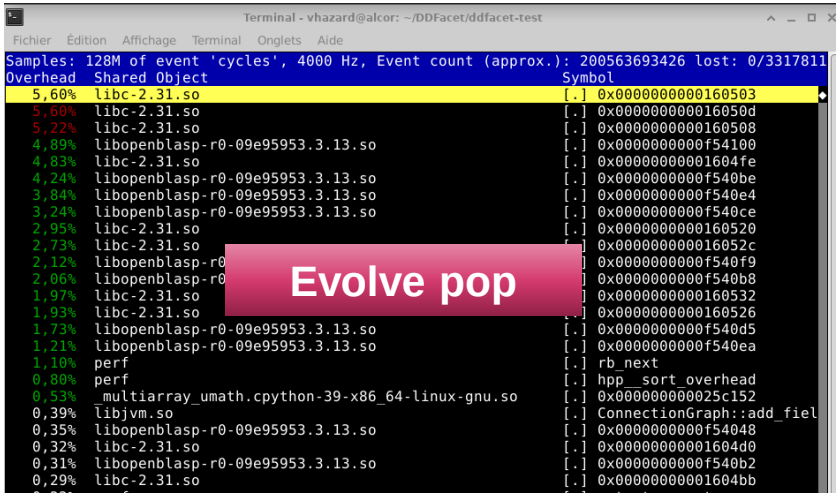


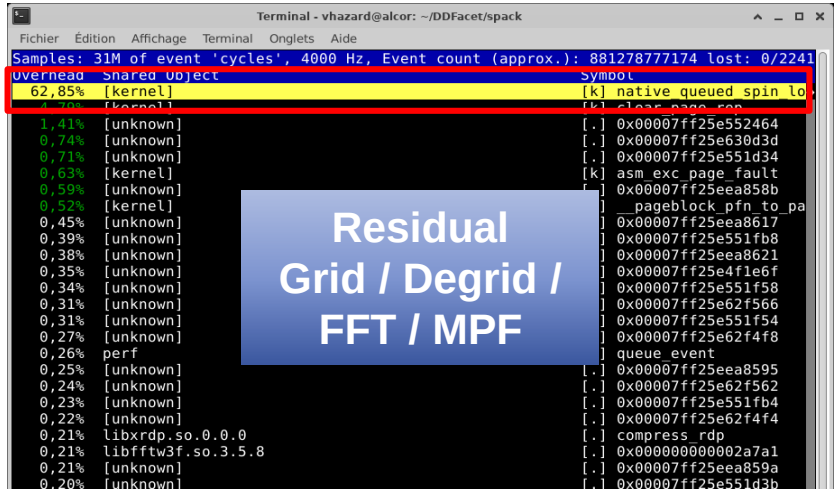
Profiling with Perf

- CPU consumption broken down by low-level function call, depending on DDF current stage









Profiling with BenchMonSPC

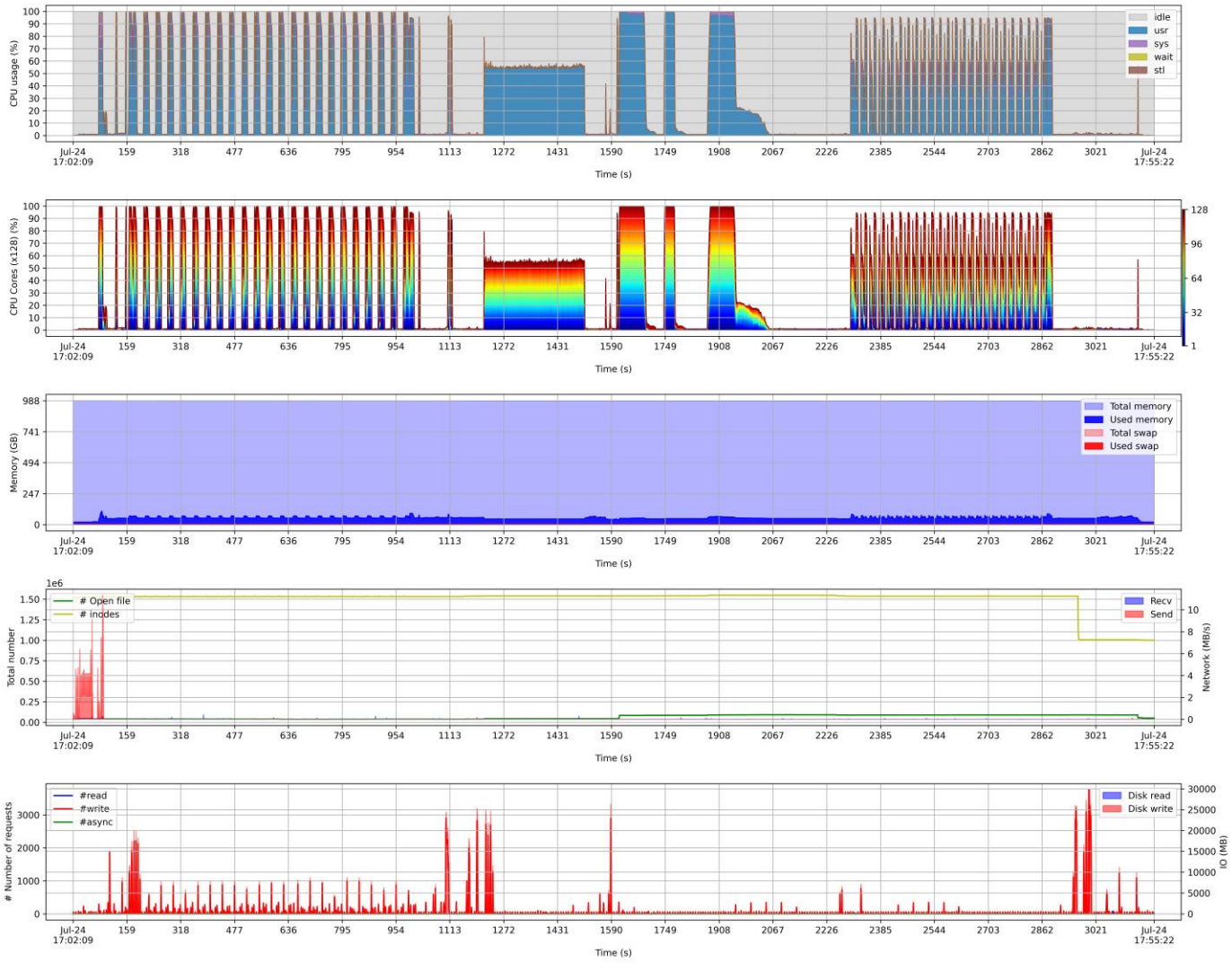
Profiling CPU Load, Memory, Disk, & Files

BenchMonSPC
Auteurs :
Anass Serhani
Shan Mignot

Data set :
24MS from LOFAR
L526161 - SB244

Server :
ALCOR (DAS06)

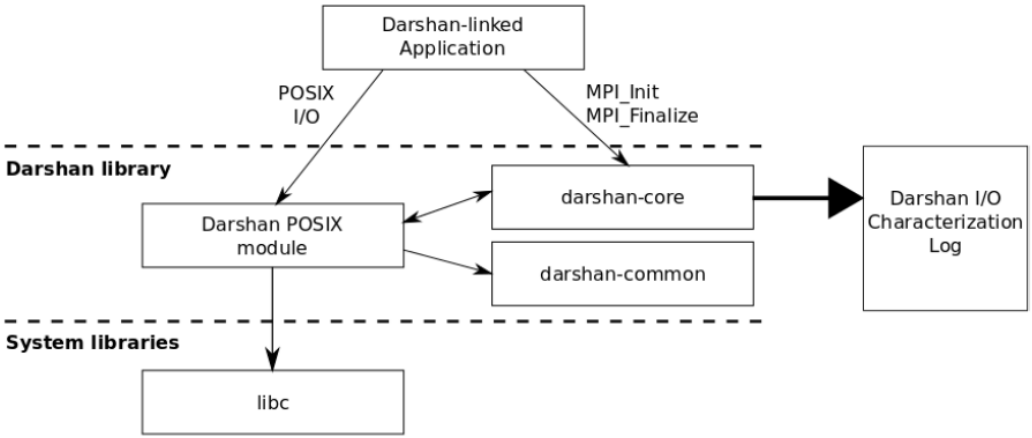
DDFacet
50-minute run



I/O Profiling

- Iheb Becher (M2 internship at LAB & INRIA) worked to use *Darshan* and profile IOs with a trace report, using dynamic instrumentation

- Darshan's *libdarshan* library intercepts system function calls, relying on two main components: *darshan-core* and *darshan-common*.



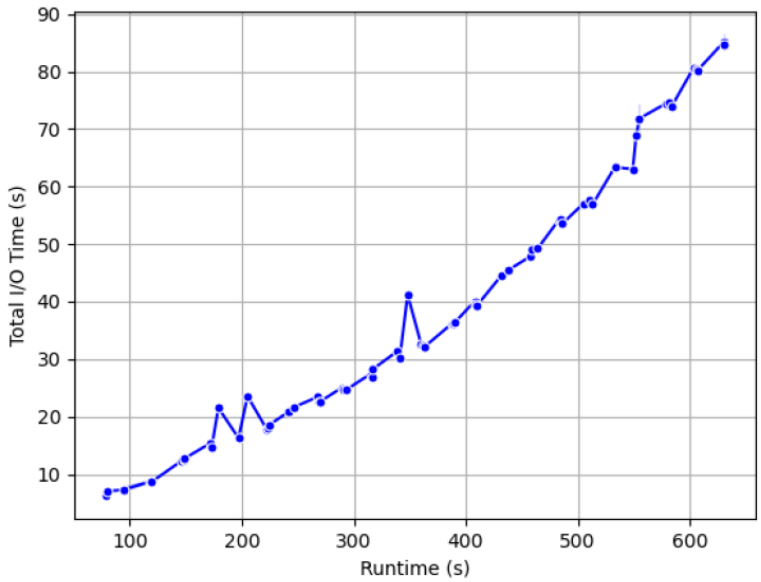
Darshan execution environment

- Darshan Python API (PyDarshan):

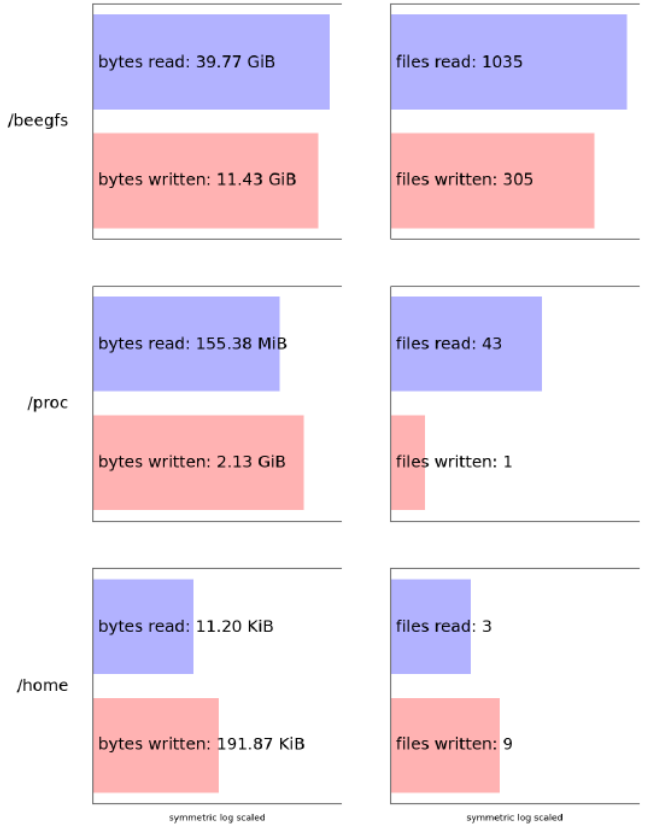
  - Read Darshan trace files (.darshan)
  - Accessing metadata
  - Exploration of I/O modules

I/O Profiling

- Preliminary I/O profiling results on a single DDFacet run (10-minute execution)



I/O time of DDFacet relative to total execution time



Data access model by file system

- Note: this doesn't necessarily mean there's an I/O bottleneck, as DDFacet executes asynchronously

- Darshan fails to capture all write operations by DDFacet, as it underestimates the total cache file size (e.g. 51.64 GB vs. 11.43 GB) measured by a system script

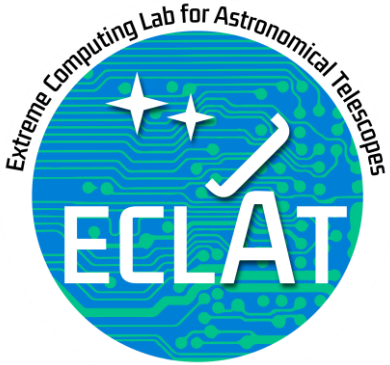
Software deployment

- Several deployment solutions have been tested so far for the DDF pipeline, as an alternative to the use of Singularity containers (Spack & Guix)
- The main challenges are deploying code to ensure reproducible environments and optimizing execution performance on the target machine

- ⌛ **Spack** deployment with the help of David Guibert (Eviden). This development is at standstill, error compiling sources (github.com/dguibert/spack/tree/dg/ddfacet-busy-week)
- ☑ Development of **Guix** package to deploy DDFacet, work carried out by Olivier Aumage (INRIA - STORM). Functional and deployed on PlaFRIM and ALCOR

What do to next?

- Coordinate efforts between different labs & industry within ECLAT
- New HR: contract hire to join the CSSD team at LAB for 2025 to work on this particular topic (as part of NumPEx)
- There's still work to be done on software deployment, on both DDF and ICAL pipelines
- Understanding the algorithms and orchestration of imaging software remains a weak spot



Spack vs Guix

- Python packages
- Does not require root to build
- Does dependency resolution
- Binaries are relocatable

- Lightweight
- Easy to deploy with root access
- More precise descriptions for large environments