Eviden SW HPC R&D/Data & Energy Efficiency

Eviden projects & tools for large applications efficiency optimization

Philippe Couvée, Mathieu Stoffel 28/11/2024

an atos business

Introduction/portfolio

DDFacet IO analysis

ARGOS

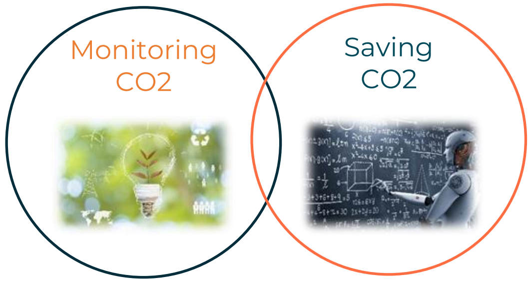
Potential collaborations

1 Introduction/portfolio

Data & Energy Efficiency Domain

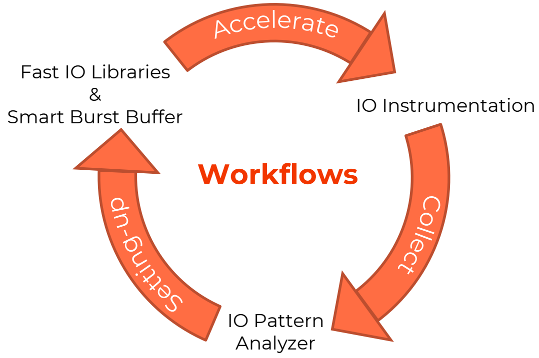
4 R&D teams, 51 engineers & PhD students

- Focus on tools for users & sysadmins in ***production***
- Make optimal usage of allocated resources



Smart Energy Management
Dynamic Power Optimizer

Smart Energy Management
Energy Optimizer



Products Portfolio

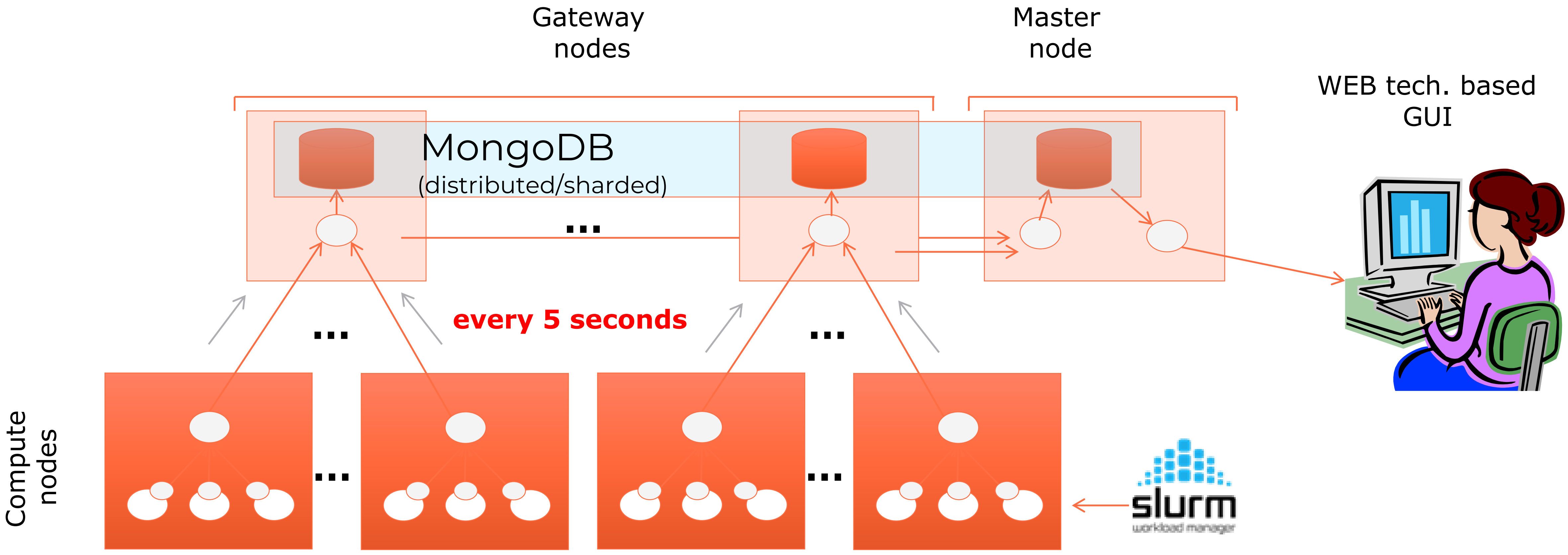
	Collect data	Analyze	Optimize
MPI, OMP System	Light Weight Profiler	-	-
Data Management	IO Instrumentation	IO Pattern Analyzer	Fast IO Libraries Flash Accelerators
Energy Management	Energy Optimizer	-	Dynamic Power Optimizer Power Capping (in EO)
Carbon Footprint	Alumet (containers, serveurs)	Real Time Carbon footprint evaluation	Slurm power saving Cloud VM control

2 DDFacet I/O analysis

IO Instrumentation Architecture & Components

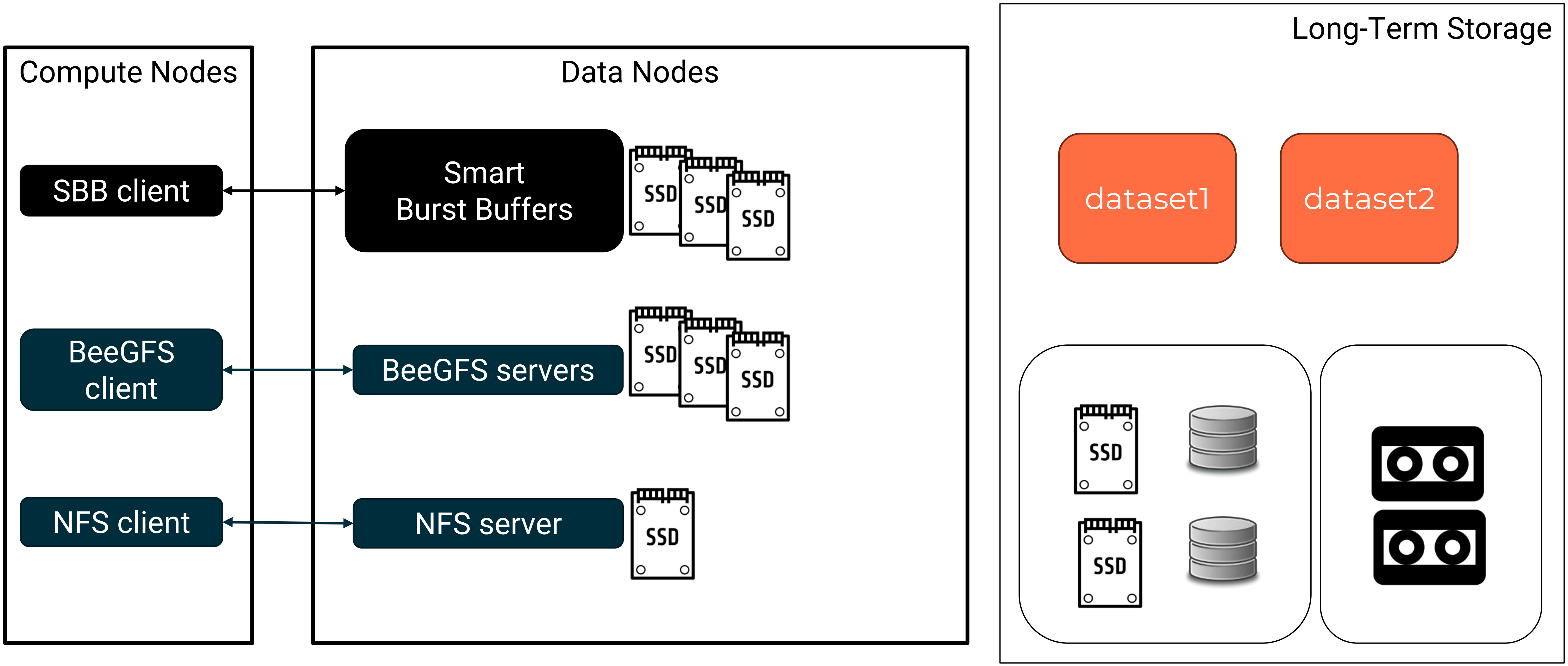
Designed for scalability

- **80+ counters, very low overhead, summary & time series**



Ephemeral I/O Services & Datasets

IO-SEA project outcome



DDFacet, MPI version

On 25 nodes

- Deployed and run DDFacet MPI

DDFacet parameters : –Cache-Dirty auto –Output-Name ddfacet –CF-Nw 100 –CF-wmax 50000

–Data-ColName DATA –Data-Sort 1 –DeconFluxThreshold=0.0035 –Deconv-MaxMinorIter=10000

–Facets-DiamMax 1.5 –Facets-DiamMin 0.1 –Facets-NFacets=11 –FreqNDegridBand 1 –Image-Cell

1.5 –Image-NPix=10000 –Output-RestoringBeam 12.0 –RIME-DecorrMode=FT –Weight-ColName=None

–Misc-ConserveMemory 1 –Output-Mode=Clean –Parallel-Affinity disable –Parallel-NCPU=64 –

Cache-VisData off

Figure 9: Sky region where the observation used during the internship was made

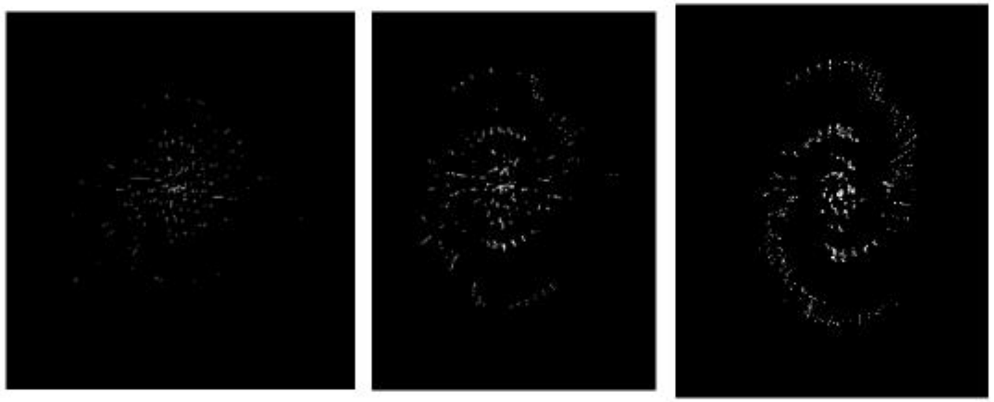


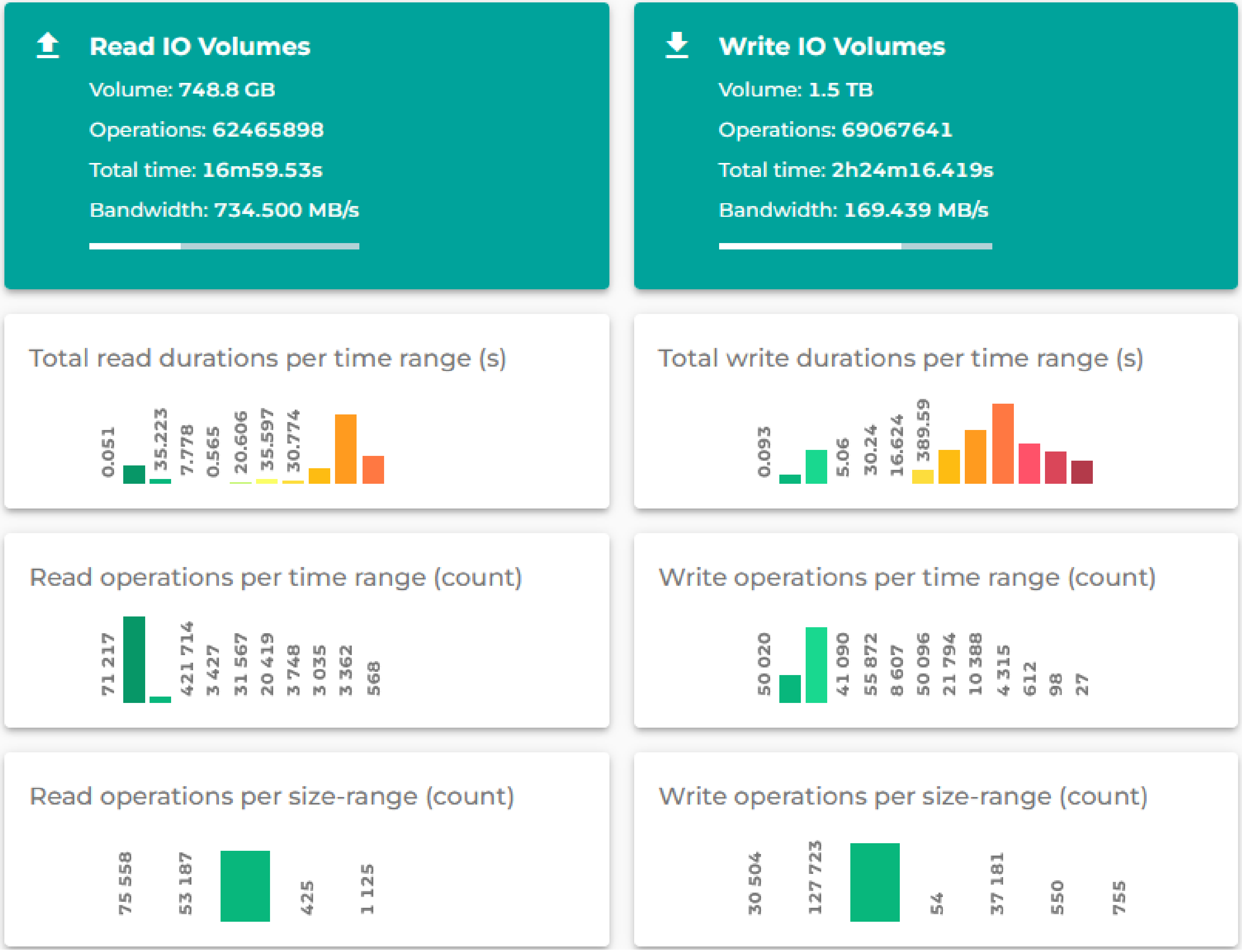
Figure 10: one galaxy in the sky image generated by DDFacet using different amounts of MS a- 12 MS b- 24 MS c- 48 MS

IO Instrumentation Example

DDFacet on Lustre

- **DDFacet run on 25 compute nodes**
  - **750 GB read**
  - **1.5 TB Write**

- **Run Time : 25m55s**
- **On a « standard » Lustre file system (~4GB/S)**



Computation time
Nodes: 25
Duration: 25m55s
Cumulated time: 4w19h6m40s

EVIDEN

IO Instrumentation Example

DDFacet on Lustre through Smart Burst Buffer

- **DDFacet run on 25 compute nodes**
  - **750 GB read**
  - **1.5 TB Write**

- **Run Time : 26m15s**
- **READ much longer, due to « cache misses » on the datanode**

Computation time

Nodes: 25

Duration: 26m15s

Cumulated time: 4w1d4h

Read IO Volumes

Volume: 751.4 GB

Operations: 62719706

Total time: 40m41.986s

Bandwidth: 307.718 MB/s

Write IO Volumes

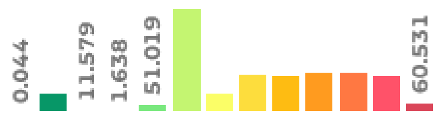
Volume: 1.5 TB

Operations: 69321491

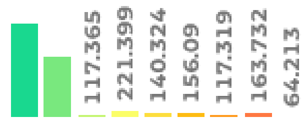
Total time: 1h28m37.572s

Bandwidth: 276.673 MB/s

Total read durations per time range (s)



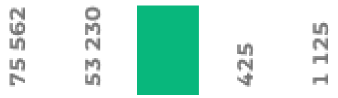
Total write durations per time range (s)



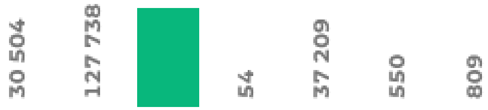
Read operations per time range (count)

Write operations per time range (count)

Read operations per size-range (count)



Write operations per size-range (count)



IO Instrumentation Example

DDFacet on Lustre through Smart Burst Buffer with dataset prefetch

- DDFacet run on 25 compute nodes
  - 750 GB read
  - 1.5 TB Write

  - Run Time : 26m15s
  - Contention on the datanode...

Read IO Volumes

Volume: 750.8 GB

Operations: 62655518

Total time: 8m46.572s

Bandwidth: 1425.800 MB/s

Write IO Volumes

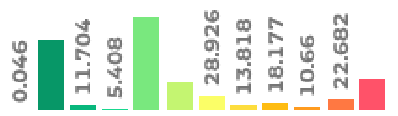
Volume: 1.5 TB

Operations: 69257331

Total time: 1h45m56.74s

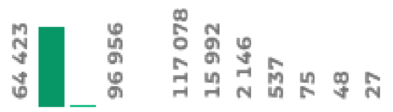
Bandwidth: 231.340 MB/s

Total read durations per time range (s)



Total write durations per time range (s)

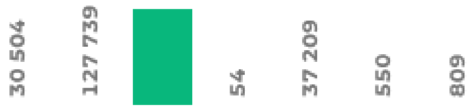
Read operations per time range (count)



Write operations per time range (count)

Read operations per size-range (count)

Write operations per size-range (count)



Computation time

Nodes: 25

Duration: 26m15s

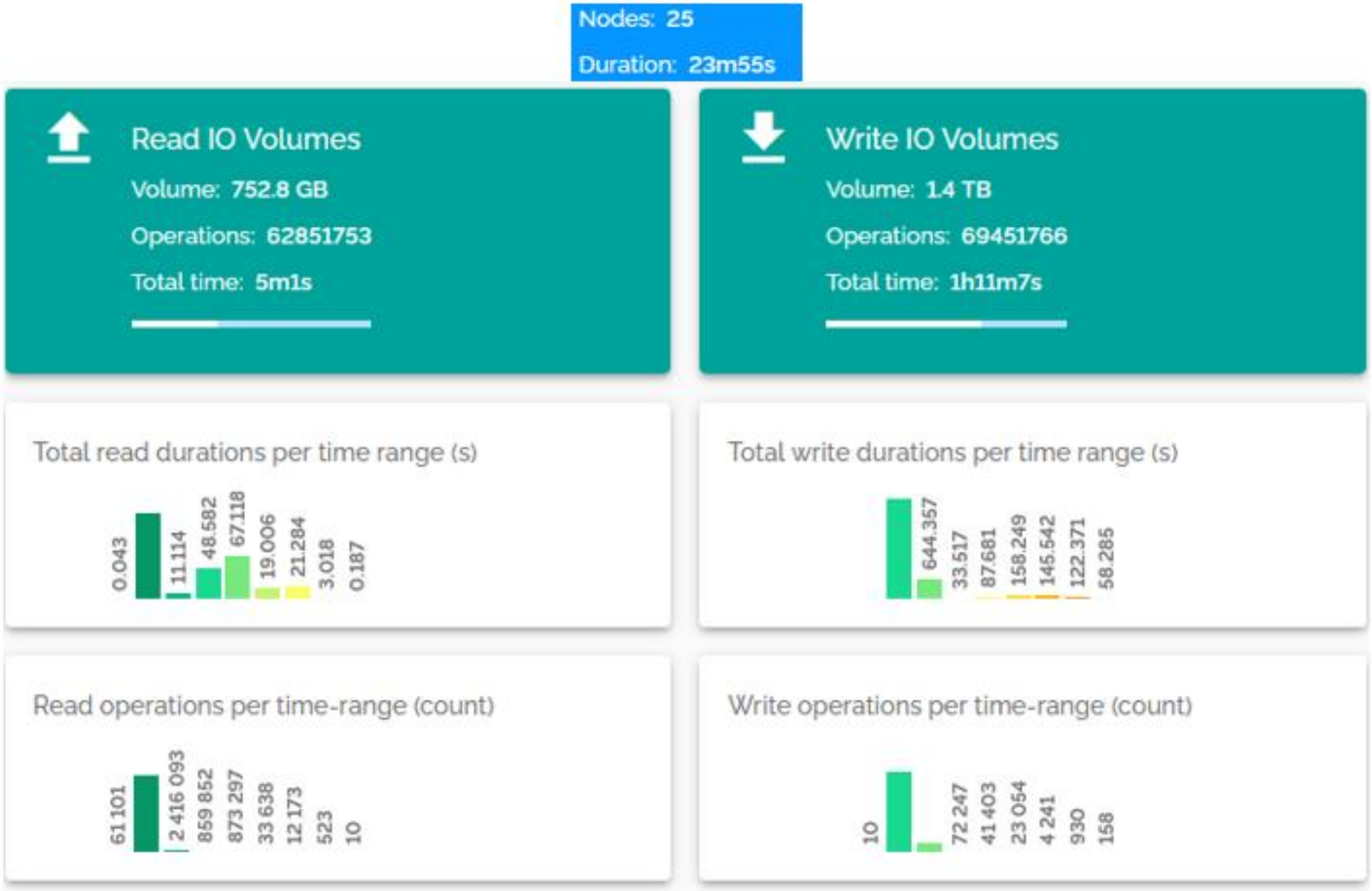
Cumulated time: 4w1d4h

IO Instrumentation Example

DDFacet on Lustre through Smart Burst Buffer with dataset prefetch (datanode configuration tuned)

- **DDFacet run on 25 compute nodes**
  - **750 GB read**
  - **1.5 TB Write**

  - **Run Time : 23m55s minutes**
  - **On a « standard » Lustre file system (4GB/S)**



3 ARGOS

ARGOS motivations

Simplify product offering

- Address customer demand for user oriented tools
- Rationalize and simplify product offering
- « Source Code Available » licence

Enable complex analysis

- Multi sources data collection (energy, CPU, memory, MPI, IO, …)
- Archive data & metrics
- Converged interfaces (GUI, APIs)
- AI based analytics to make complex analysis & generate *recommendations*

Optimize run time efficiency

- Act through *levers*
  - Existing: Energy efficiency (DPO, Capping), I/O
  - New: **Memory**, MPI, placement, AAPC…

EVIDEN

ARGOS product positioning

- Focus on ***application behavior*** in **production conditions**

  - Low overhead : necessarily not « exhaustive »
  - Not a tool for developers in dev phase, to debug and optimize applications
    - There are already many good tools for that purpose
  - No need for source code, no recompilation….
  - Automation, configurability

Argos Roadmap

Rationalize software offer by merging all advanced features into a single product

Performance Studio 1.0

- Target: December 2024
- Release objectives:
  - System metric collection only (CPU, MPI, memory)
  - CLI + API (no GUI)
  - Report generation (JSON + human readable)

Performance Studio 2.0

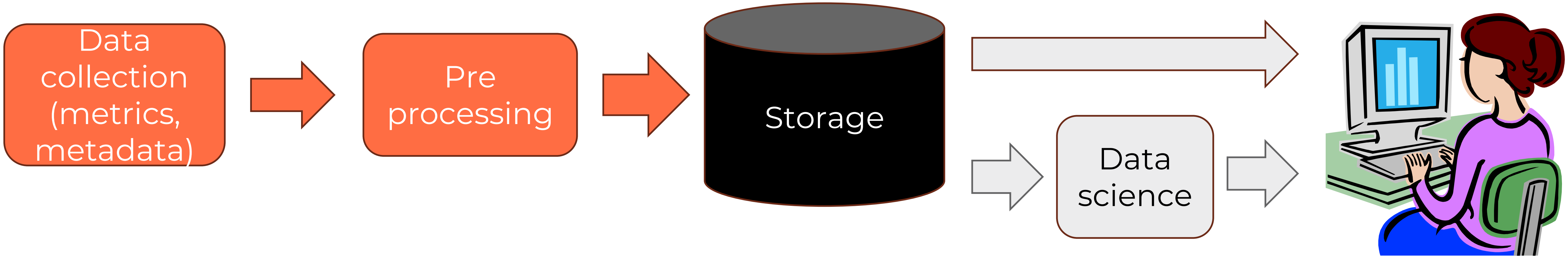
- Target: H1 2025
- Release objectives:
  - Dedicated GUI
  - I/O, Energy metrics
  - Optimization tools
  - I/O, energy

Performance Studio 3.0

- Target: H2 2025
- Release objectives:
  - More admin features
  - Performance impact analysis
  - Analytics

ARGOS Architecture, high level view

Data collection, analysis & run time optimization of large HPC/IA jobs in production



10K+ sources compute nodes and Infrastructure

Near real time data collection & agregation

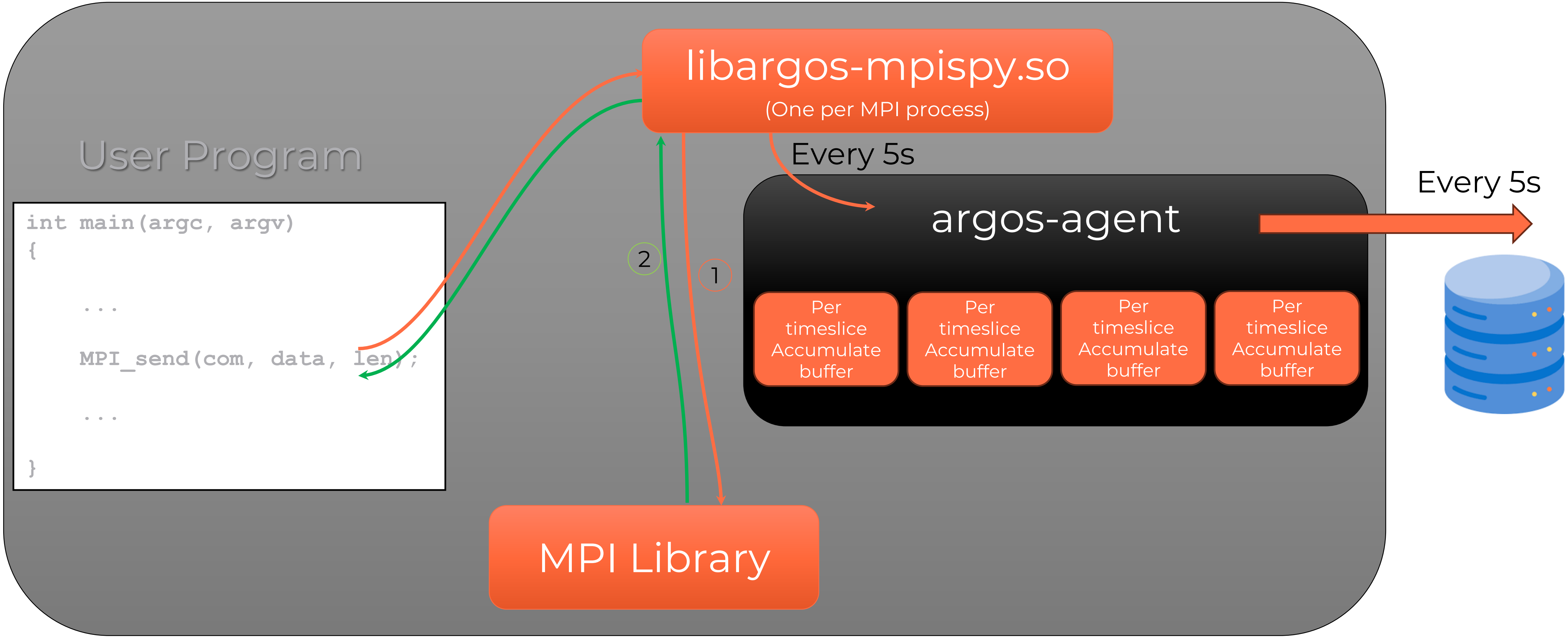
MongoDB based
No data aging

Data Analysis

Reports
GUI
Role based access

ARGOS V1.0 : MPI Interception & instrumentation

PMPI based



ARGOS V1.0 Collective MPI metrics

MPI Collective Metrics

Collective Operations	1664
Collective Time	938ms 587274ns
Incoming Bytes	52.66 KiB
Outgoing Bytes	52.66 KiB
Synchronization Operations	0
Synchronization Time	0 ns
Total Collective Calls	1664
Total Time in Collective Communications	938ms 587274ns

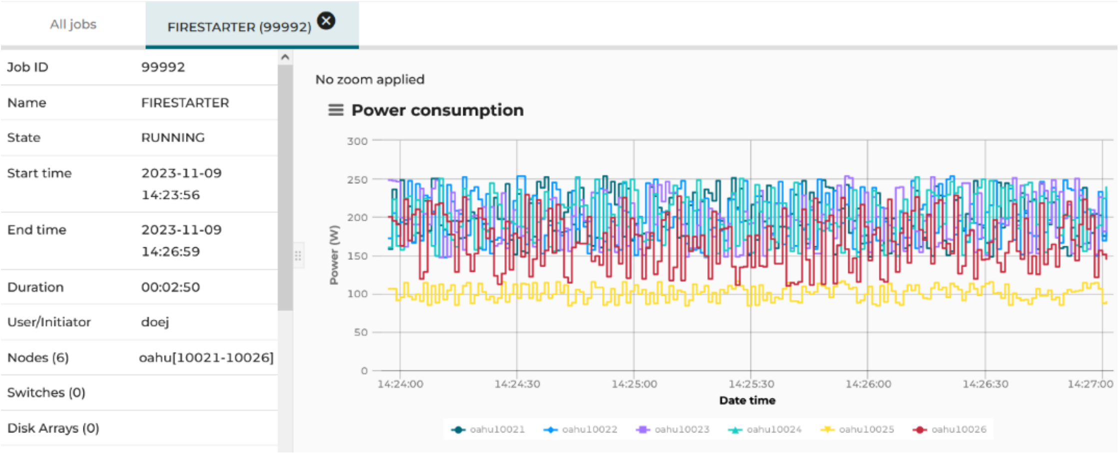
EVIDEN

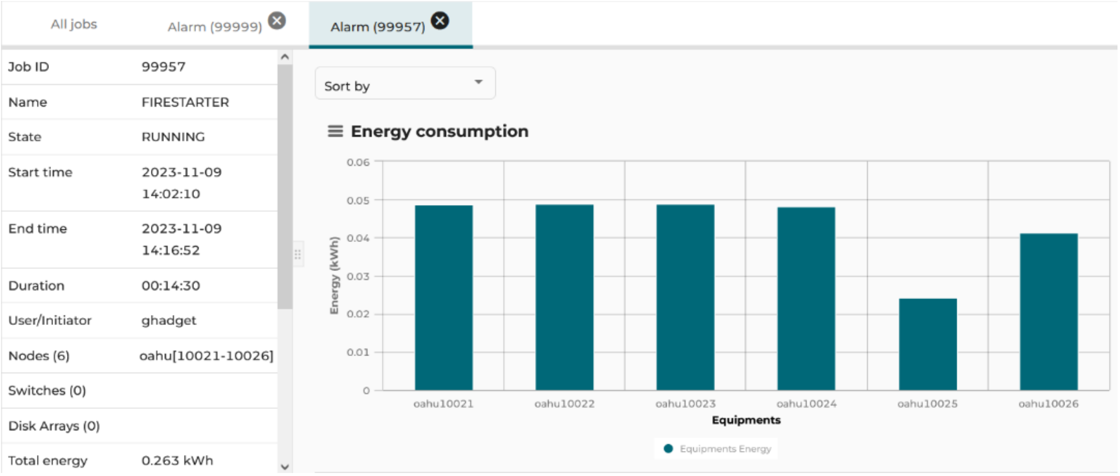
4 Potential collaborations...

ARGOS V2.0 : Energy monitoring

The first step to mastering is observing - out-of-band monitoring with Energy Optimizer (EO)

- Out-of-band sampling of the power consumption of several types of components: mostly compute nodes, a few network switches, …
- On current and legacy hardware:
  - 1-second sampling period
  - Monitoring data stored in DB for up to 5 years (with different resolutions based on how old are the data)
  - For compute nodes, through the Redfish service of the BMC
- For (some) future hardware (**design in progress**):
  - ~100 values sampled at 10 Hz stored on the BMC
  - Retrieved every few seconds (TBD) and injected in DB
- CLI, REST-API, and GUI to access the data

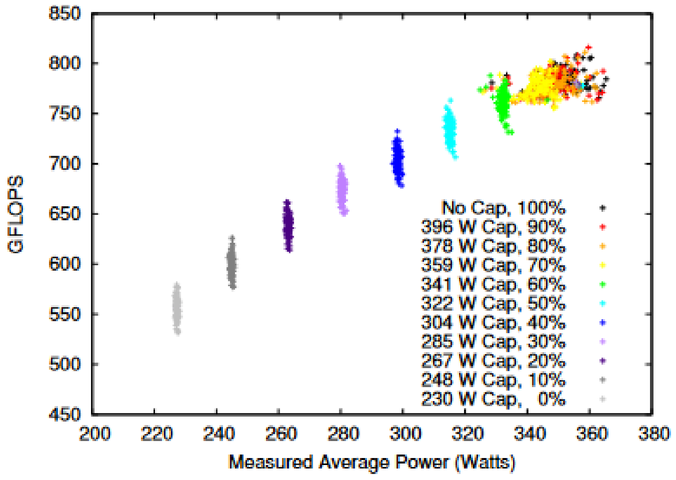




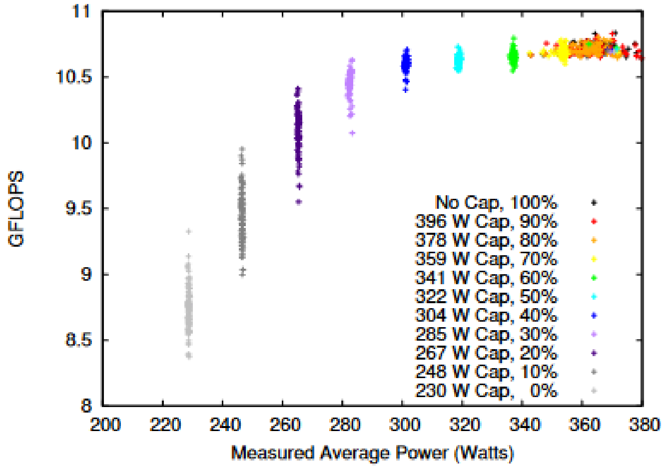
ARGOS V2.0 : Energy monitoring

A use-case for the power/energy monitoring data

- Even homogeneous compute nodes exhibit different performance/power consumption trade-offs
- For instance, the graphs on the right (extracted from [1]) showcase the performance distribution of 100 homogeneous nodes under power caps for HPL and HPCG
- If the power caps allowing boost frequencies to be used are set apart, nodes can exhibit around 7.5% of performance variability for the enforced same power budget
- SKA might (will?) need power capping capabilities to cope with the "power supply perturbations"
- First use-case of monitoring:
  - Divide nodes in bins regarding performance under power cap
  - Preferentially use the most performant nodes when under power cap



(b) HPL: Power Cap Sweep



(d) HPCG: Power Cap Sweep

[1]Pedretti et al. - *A Comparison of Power Management Mechanisms: P-States vs. Node-Level Power Cap Control*, 2018, 10.1109/IPDPSW.2018.00117

ARGOS V2.0 : Power capping

Power capping capabilities of Energy Optimizer (EO)

EO implements power capping capabilities for a broad range of BullSequana compute nodes, including the X2000 and XH3000 ones.

Power caps are set out-of-band, through the BMC of the nodes and those capabilities are also exposed through CLI, REST-API and GUI.

On top of the enforcement of power caps, some work have been started to extend EO with refined features regarding power capping, among which ***Application-Aware Power Capping (AAPC)***:

- The more power-eager the jobs, the more performance degradations under power cap they display
- Redirect power budgets between jobs depending on their power-eagerness to decrease the global performance degradation when compared to the standard approach Fair-Sharing Power Capping (FSPC)
- PoC implemented (more details on the next slide) and patent under review
- Other criteria could be used to redirect power budgets between jobs, for instance job priority

ARGOS V2.0 : Power capping

More details about Application-Aware Power Capping (AAPC)

PoC implemented and deployed on a 12-node partition of kiwi (an Eviden R&D system):

- The same schedule of jobs (figure on the right) was executed without power capping (Baseline), and with a global power budget to be enforced either by AAPC or FSPC
- Some of the jobs contained by the schedule are power-eager, some display less power-eagerness, and some exhibit an intermediate behaviour
- The below table contains the results (averages on seven repetitions):
  - Power budgets redirected toward power-eager jobs for AAPC
  - Globally, less performance degradations induced by AAPC when compared to FSPC

Power capping strategy	Relative TtS increase when compared to Baseline
AAPC	+3.99%
	+3.93%
	+1.17%
FSPC	+10.7%
	+3.28%
	+0.451%

- The x-axis is the time spent since the start of the schedule (in seconds) – first 12 minutes are displayed
- A mix of standard HPC benchmarks was used to submit jobs

EVIDEN

ARGOS V2.0 : Optimize the power-efficiency of HPC applications

A few words about Dynamic Power Optimizer (DPO)

Dynamic Power Optimizer (DPO) is a runtime tool designed to run on the compute nodes, in parallel of an HPC application to optimize its power-efficiency (FLOP/s / W).

No need to annotate or recompile the target HPC application. Integrated with Slurm through a SPANK plugin to start it when a job begins and terminate it when the latter ends:

```
srun --dpo=yes (… Slurm options …) /path/to/my/app (… App parameters …)
```

DPO resorts to fine-grain monitoring of a set of events to build metrics representative of how well the executed HPC application uses the computing resources of the node.

<u>It then scales the frequency of the CPUs *(GPU support in the roadmap)* based on those metrics</u>:

- Lower frequencies means less computing power and less power consumption
- If during a short phase the application does not need the maximal frequency to reach its nominal level of performance, then scaling the frequency down allow for energy savings at minimal performance cost
- When the short phase stops, the frequency is scaled up to its nominal value to avoid significant impact on performance

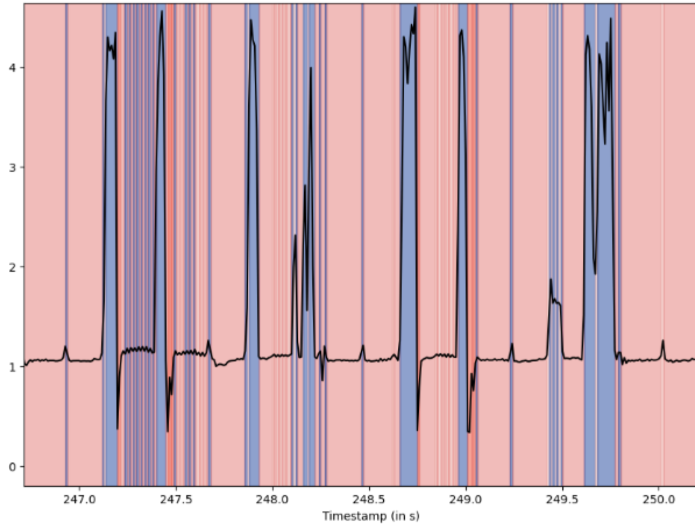
ARGOS V2.0 : Optimize the power-efficiency of HPC applications

Some results with DPO and an invitation

To validate the support of the X2140 blade (2 x Intel Sapphire Rapids 8480+ CPUs) in DPO, some experiments were performed with HPCG.

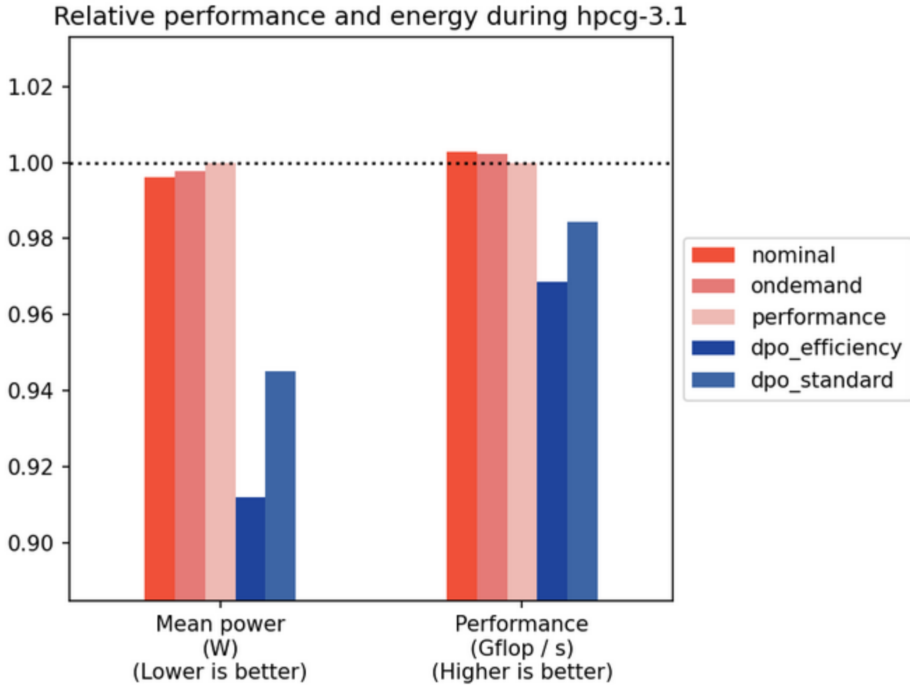
<u>In the top right corner</u>:

- The evolution of metric IPC (Instruction retired Per reference Cycle) as monitored by DPO (10 ms sampling period) is shown
- In red, phases for which the frequency is scaled down
- In blue phases for which it is scaled back up



<u>In the bottom right corner</u>:

- Comparison of performance and energy consumption of HPCG when executed with the ACPI CPU frequency governors `performance` (baseline) and `ondemand`, the constant nominal frequency, and DPO with two configurations named “standard” and “efficiency”
- With the “efficiency” configuration, DPO reduces the mean power consumption by 8.8% for a performance decrease of 3.1%. **That is 6.2% more FLOP/s / W!**



ARGOS V2.0 : Optimize the power-efficiency of HPC applications

Some results with DPO and an invitation

The configuration of DPO can be fine-tuned for a specific HPC application being executed on specific hardware.

We already collaborated with application developers to fine-tune the configuration of DPO for production HPC applications such as the Integrated Forecasting System (ECMWF), TerrSysMP (JSC), and GROMACS (Groningen University) in the context of EU projects. We got some good results!

If you are interested in testing DPO on an application, do not hesitate to reach us (e-mails address on the last slide ☺).

EVIDEN

Invitation: Explore a new lever on memory

Internship to start exploration in March 2025

Memory hungry applications

- Need to allocate more nodes than needed from CPU/GPU resource point of view
  - ➢ Allocated CPUs won't be fully used

CPU hungry applications

- Need to allocate more nodes than needed from memory resource point of view
  - ➢ Allocated Memory won't be fully used

Example of power consumption (W) of 2 modern blades in idle mode and running HPL

Blade	type	cpu	gpu	nic	mem	other
xh3140_idle	CPU	288.2	0	14.9	56.6	46.2
xh3140_max	CPU	988.2	0	21.9	188.7	103.5
xh3145_idle	CPU+GPU	247.1	600	24	41.1	103.8
xh3145_max	CPU+GPU	847.1	2400	32.7	147.1	235.5

- Extend virtual memory spaces with a new ephemeral IO service
  - Based upon xmap, a kernel module developed by FORTH

- Limit energy consumption by switching off part of the memory on allocated nodes
  - Based upon linux hot plug features

Thank you

philippe.couvee@eviden.com
mathieu.stoffel@eviden.com