What can we do with the tools developed in this wp?

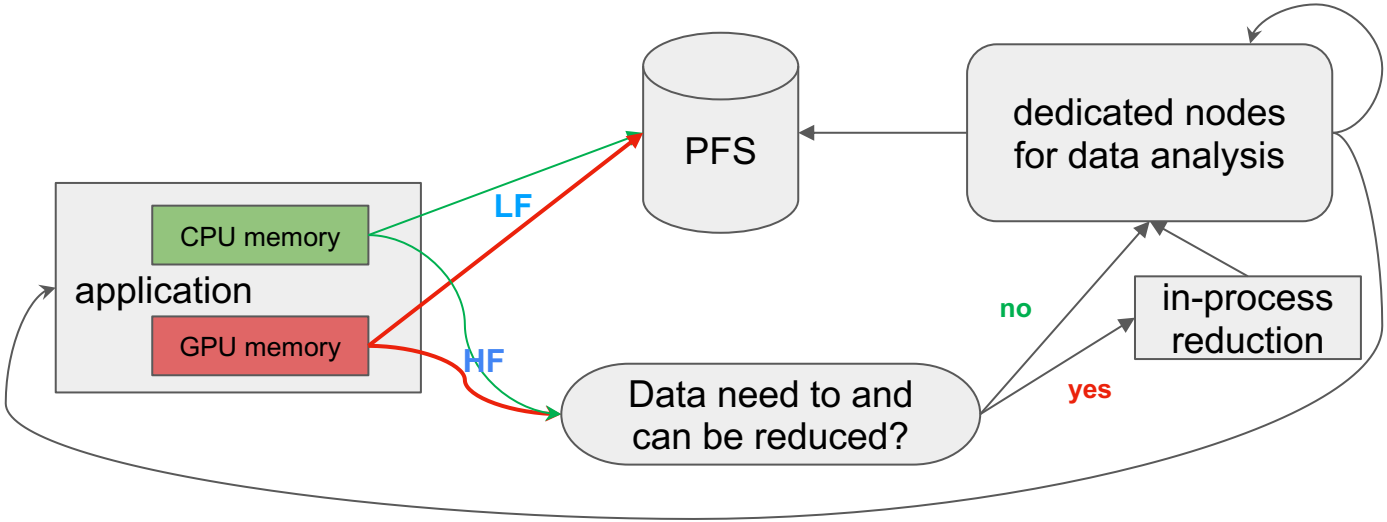
- Application running on GPU. IO-tools can fetch directly device data and, according to in-situ analysis workflow, decide whether or not copy data to host.

- Application which generates significant amount of data. We can have relatively high frequency writing, with reduced data. We can have in-proc local reduction, and global reduction on dedicated IO nodes.

- We can launch data analysis for monitoring, and depending on the results, launch other diagnostics and/or adjust the compute resources.

- Detect, with AI for instance, rare events in simulation, write only data within the interest zone, follow the evolution of the events.

- Checkpointing, CPU/GPU, synchro/asynchro, at low frequency.
- in-situ & in-transit data analysis, at high frequency
- perform in-process reduction
- according to the workload of data analysis, adjusting resources between application and analysis
- according to the results of data analysis
  - launch new analysis
  - feedback for restarting
  - write to PFS small amount of data, at very high frequency



Feedback

- currently: application → in-situ
- future: application ⇄ in-situ

GPU data

- currently: PDI & Damaris do not support GPU data directly. H2D MemCpy performed at application side.
- future: fetch GPU data directly
- benefits: AI on the same GPU to detect events, and to trigger IO and in-situ.

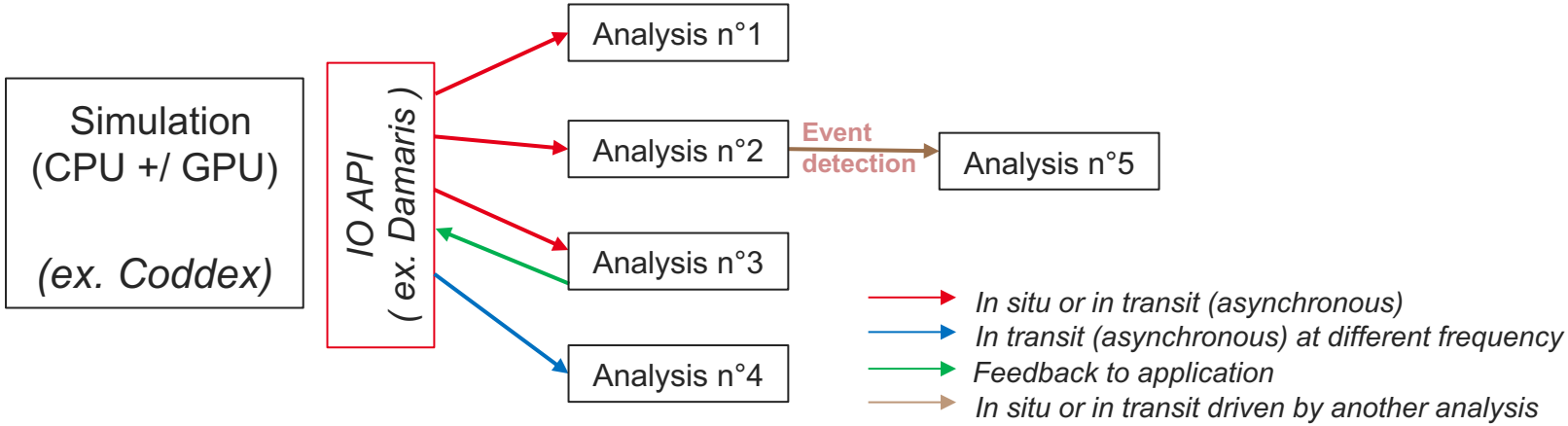
Data volume

- currently: Deisa get what is exposed via PDI
- future: perform in-proc data reduction to reduce tension on FS

PDI

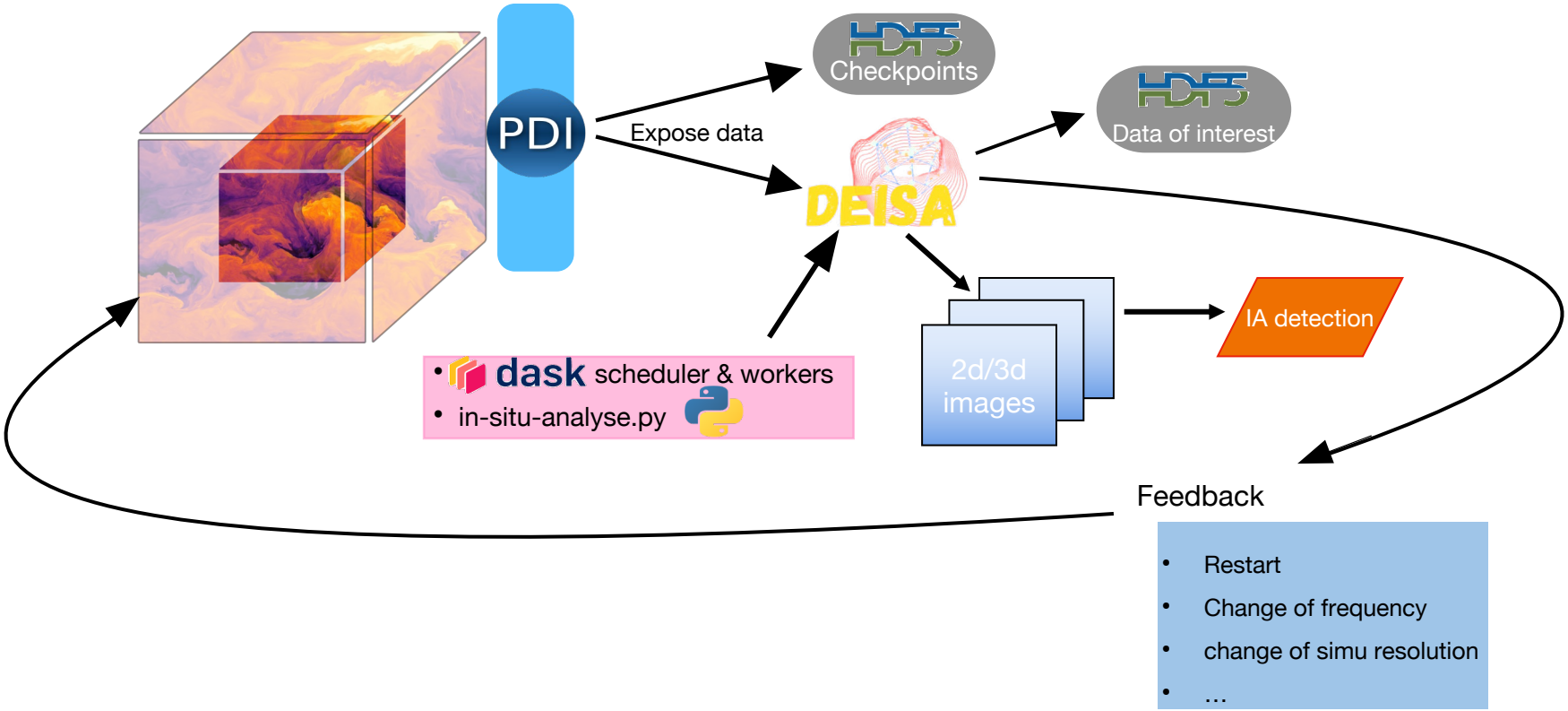
- currently: mode "push".
  - Data received from application;
  - PDI check if the data is needed by any plugin;
  - if yes, invoke the plugin
- future: mode "pull"
  - Identify in advance all results;
  - Identify which data will be necessary for the results;
  - Require data from application

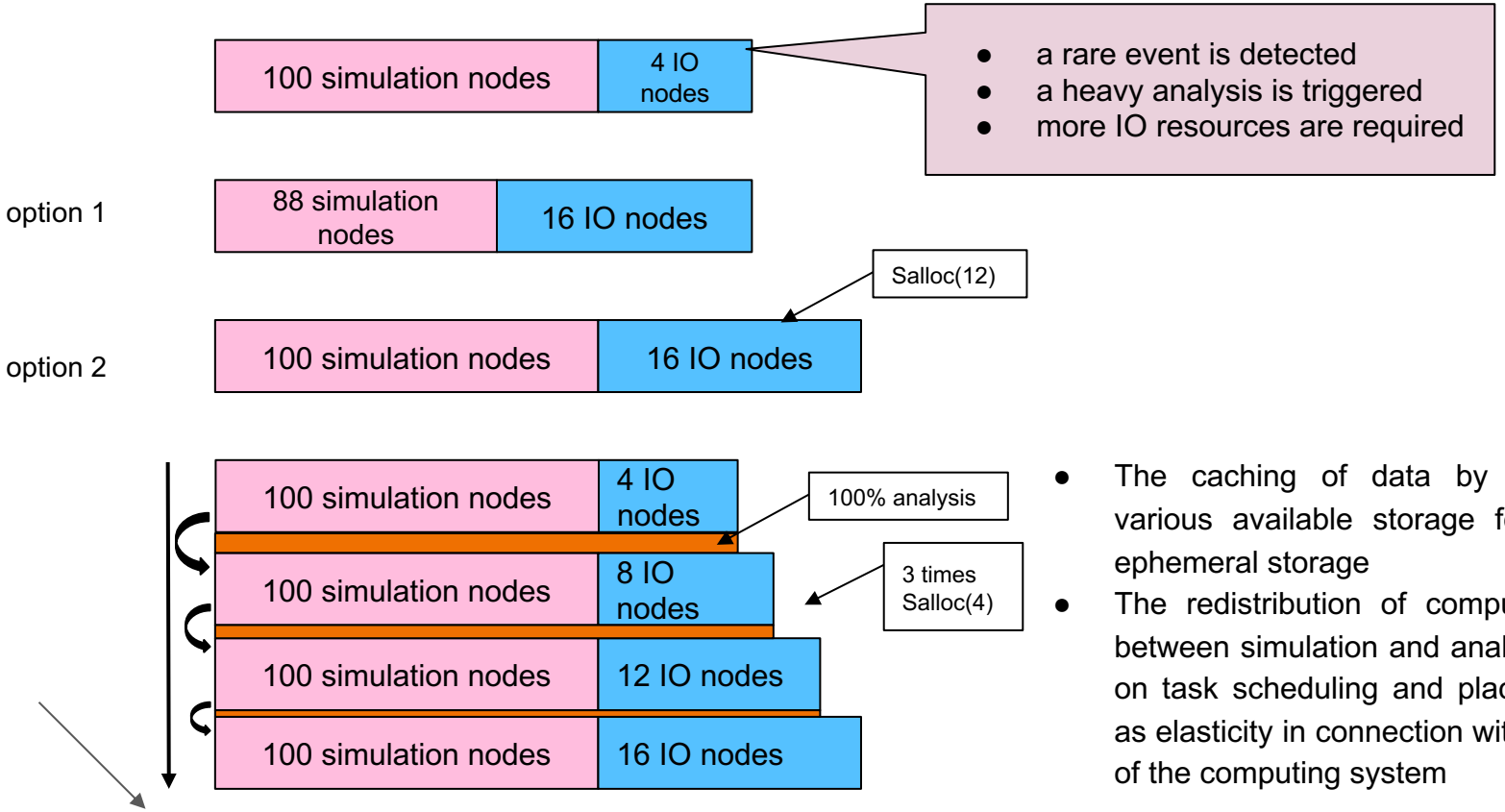
- Add context in PDI terminology
  - We want buffer be exposed for longer period of time
  - a better way to identify buffer (time_stamp, mpi_rank, …)



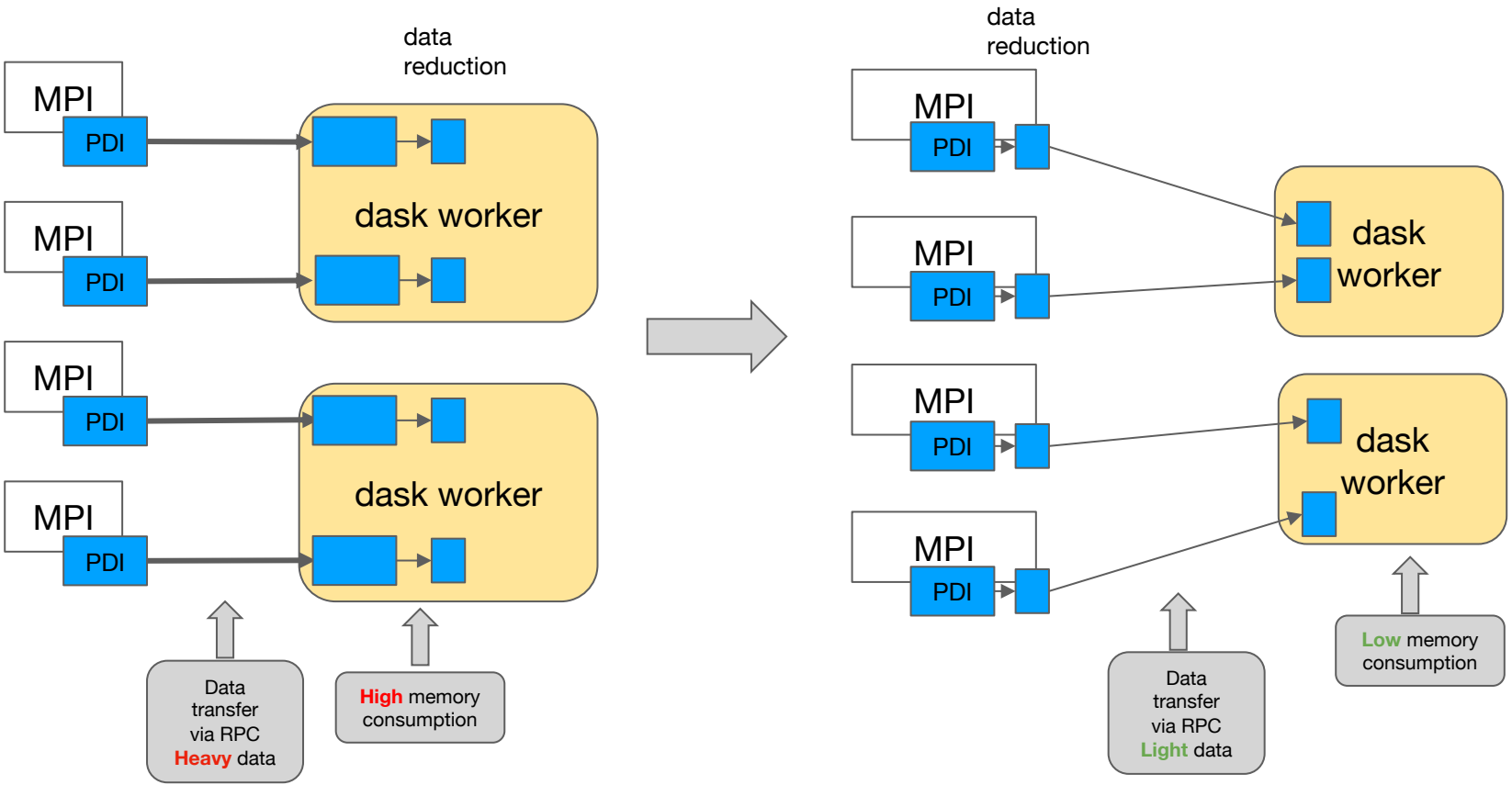
- An analysis can either inform the code that will trigger a new analysis, or an analysis, based on a physical criterion, can trigger a new analysis by itself.
- An analysis may request an increase in the frequency of an analysis call that is already present in the contract.
- What happens if the 'sent and compacted' data needs to be modified with the arrival of analysis 5? A rollback or, if planned, a local iteration via the PDI API.
- Analysis 5 could also be a request for a rollback of the code with a different time step.

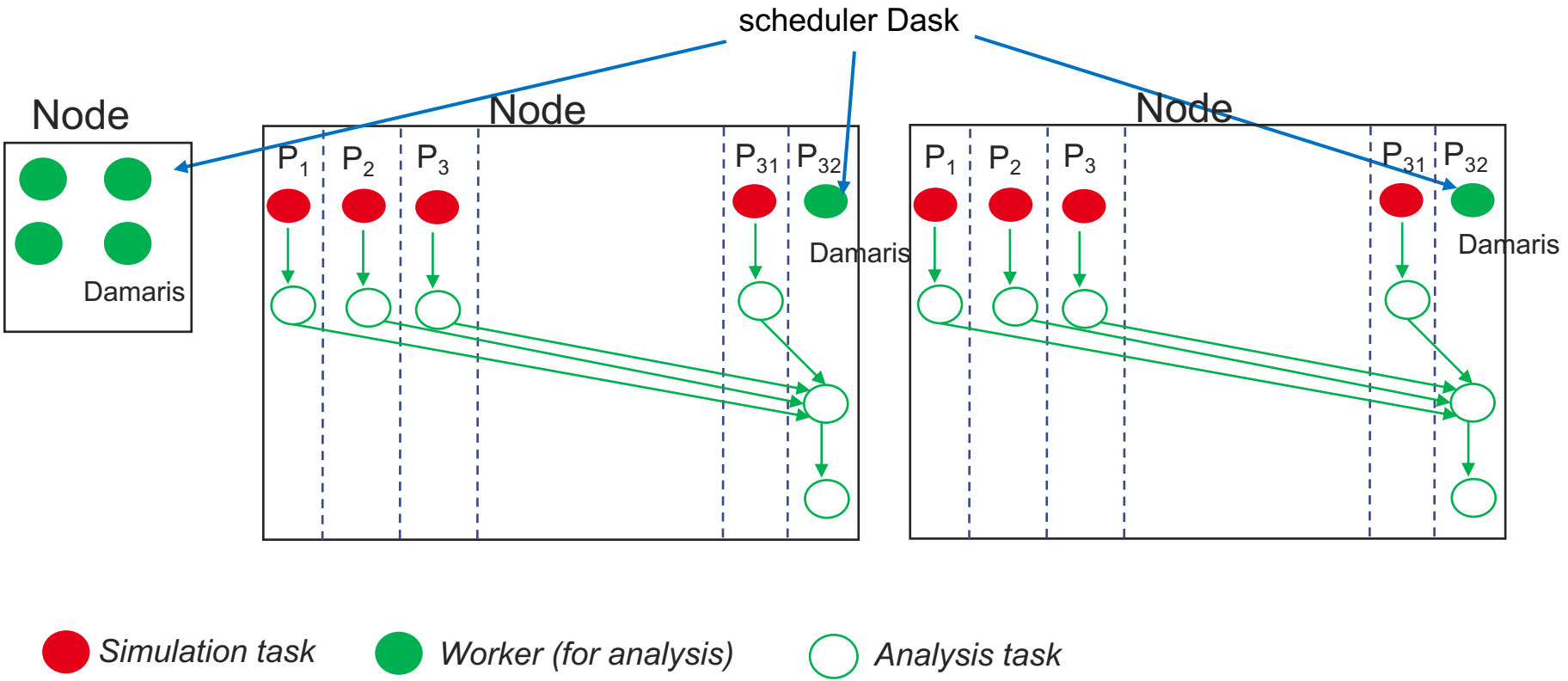
SPOT: Simulation Pattern Observation Tool



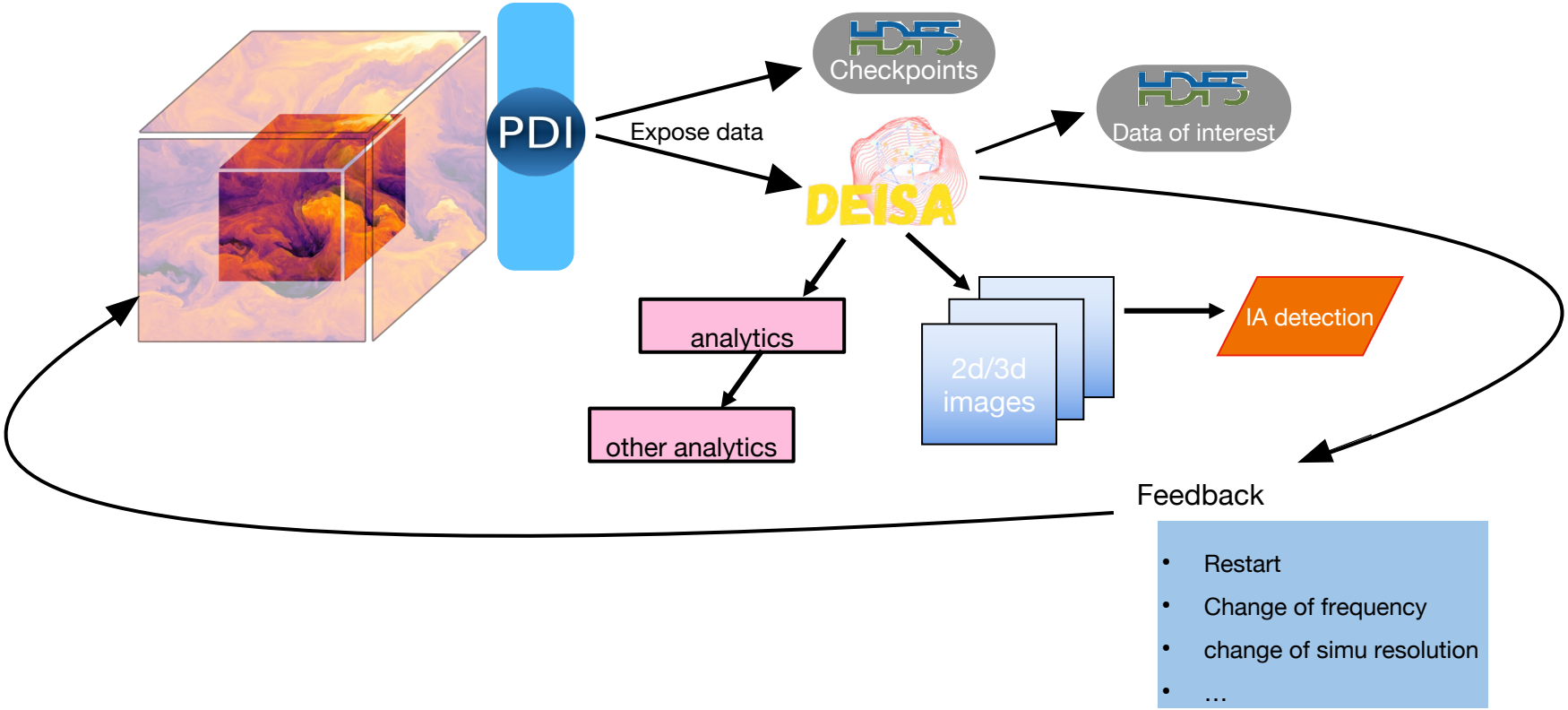


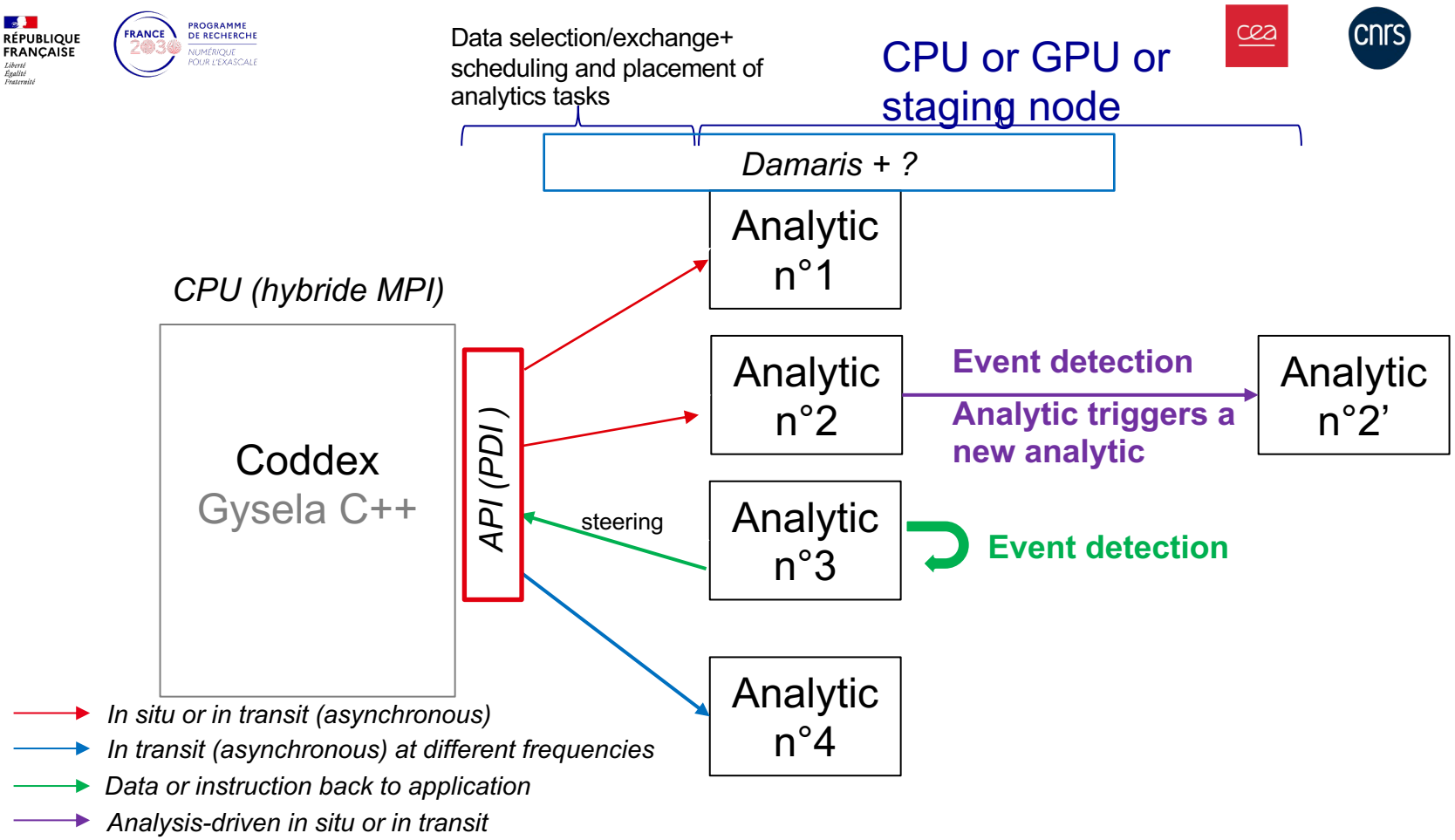
- The caching of data by leveraging the various available storage forms, including ephemeral storage
- The redistribution of computing resources between simulation and analysis by working on task scheduling and placement, as well as elasticity in connection with the scheduler of the computing system


data
reduction
MPI
PDI
MPI
PDI
MPI
PDI
MPI
PDI
dask worker
dask worker
Data
transfer
via RPC
Heavy data
High memory
consumption
data
reduction
MPI
PDI
MPI
PDI
MPI
PDI
MPI
PDI
dask
worker
dask
worker
Data
transfer
via RPC
Light data
Low memory
consumption


scheduler Dask
Node
Damaris
Node
P1
P2
P3
P31
P32
Damaris
Node
P1
P2
P3
P31
P32
Damaris
Simulation task
Worker (for analysis)
Analysis task

SPOT: Simulation Pattern Observation Tool




Data selection/exchange+ scheduling and placement of analytics tasks
CPU or GPU or staging node
Damaris + ?
Analytic n°1
Analytic n°2
Analytic n°2'
Event detection
Analytic triggers a new analytic
CPU (hybride MPI)
Coddex
Gysela C++
API (PDI)
steering
Analytic n°3
Event detection
Analytic n°4
In situ or in transit (asynchronous)
In transit (asynchronous) at different frequencies
Data or instruction back to application
Analysis-driven in situ or in transit

Action items

- Explore use PDI as common interface with the analytics
- Use Damaris, Paraview, and Deisa as plugins
- Explore PDI to Catalyst and VTK connectors with Kitware
    - Damaris plugin for Catalyst would serve as an intermediate interface in the case of dedicated cores/resources
    - PDI would be a good solution as well, very light plugin with Paraview
    - PDI-Damaris plugin also WIP.
- Resort to Damaris for dedicated core (in the future GPU) analytics support
- Explore application of Damaris to overcome IO bottleneck in GYSELA checkpointing (not easy due to extra copy needed, but there is an opportunity with the GPU simulation) → Connection with WP1
- Fault tolerance : use VeloC ? or use FTI (not maintained anymore)? (not really in this wp, but PC4?)