Which Memory Abstraction for NVRAMs – A Perspective from the Sage2 project

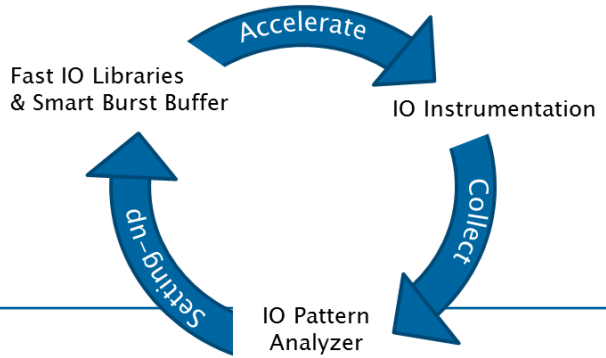
Gregory Vaumourin, Christophe Laferrière, Philippe Couvée, Sebastien Valat

Atos BDS R&D Data Management, France

Bull
atos technologies

Who is this guy ?

- PhD at INRIA Bordeaux (France) in 2016 with Denis Bartou, about memory caches management for embedded architectures
- Post-Doc at University of Uppsala (Sweden) with David Black-Schaffer about NVM for Memory caches
- Now working at ATOS in the R&D SW Data Management Team with Philippe Couvée
  - Tools for profiling/analyzing I/Os
  - I/O SW Accelerators
  - Flash-based Burst Buffers



Outline of the presentation

- I – A bit of context with Sage2
- II – How we plan to use NVDIMMs
  - Overall semantics
  - uMMAP-IO description
- III – One Optimization for Checkpoint optimization

I - One Storage System to rule them all

Extreme Computing

- Changing I/O needs
- HDDs cannot keep up

Big Data Analysis

- Avoid Data Movements
- Manage & Process extremely large data sets

AI/Deep Learning

- Large Memory Requirements
- I/O Requirements not well defined!

I- Sage2 Partners

SEAGATE
UK

Atos Bull
atos technologies
France

CCFE
CULHAM CENTRE FOR FUSION ENERGY
UK

cea
DE LA RECHERCHE À L'INDUSTRIE
France

Kitware
France

JÜLICH
Forschungszentrum
Germany

epcc
UK

arm
UK

KTH
VETENSKAP
OCH KONST
Sweden

"Sage2"
Percipient
Stor**AG**e
for **E**xascale data
Centric computing2

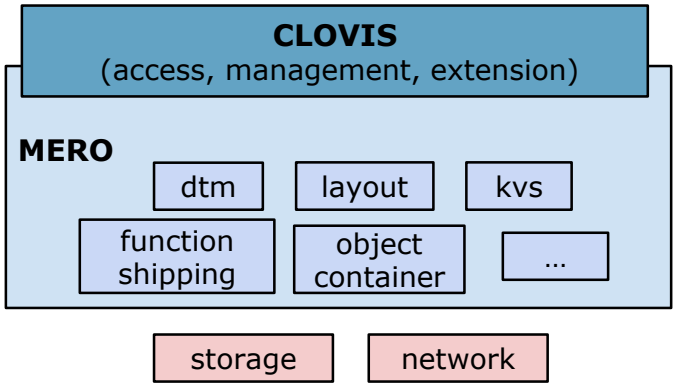
FETHPC Project
~€4M
9 partners

Sept 2018 – Aug 2021

Bull
atos technologies

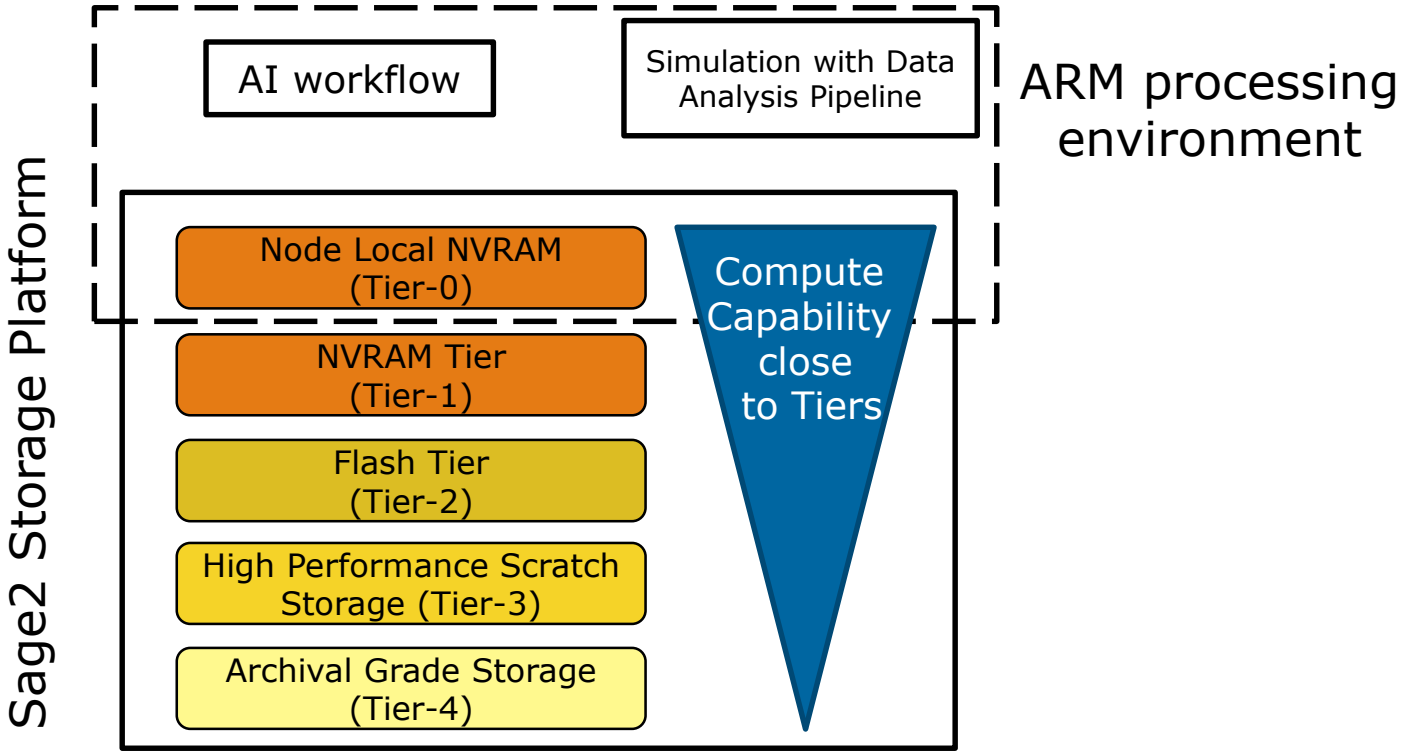
I – MERO Introduction

- Mero: distributed object storage platform developed by Seagate
  - Tackle scalability problems of traditional file system for exascale
  - Handle a pool of nodes that might have non-volatile memory devices
  - Optimizations such as function shipping, object containers, …
  - Will become open-source soon

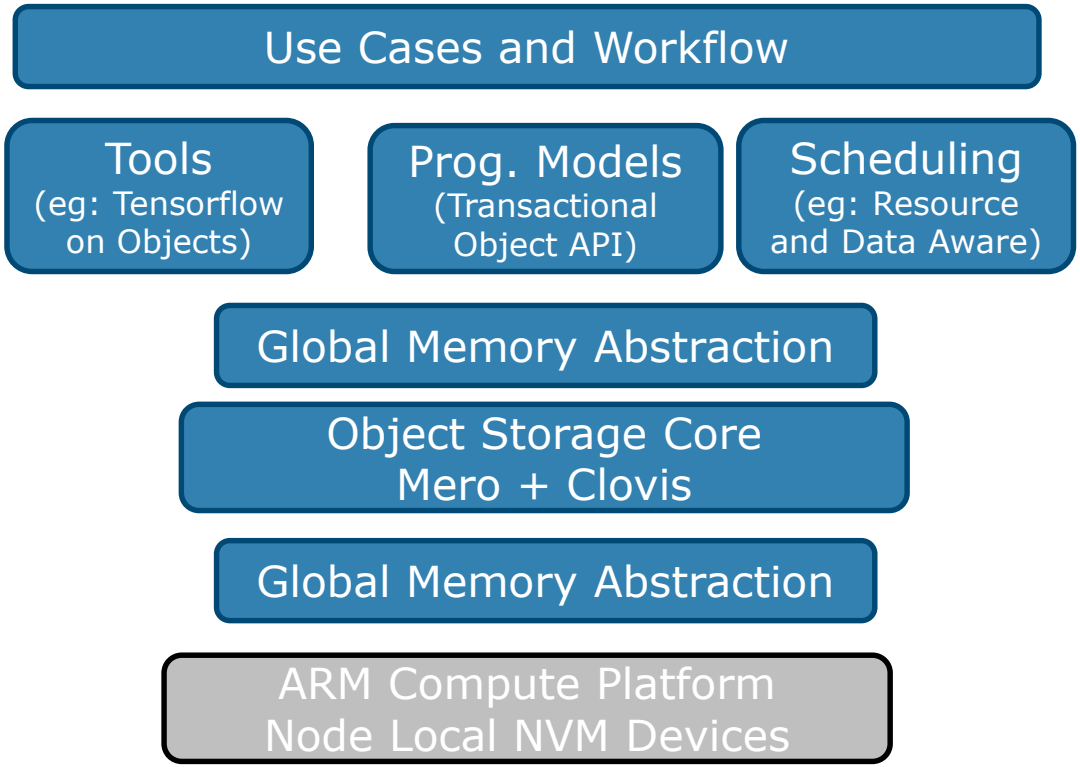


- Clovis: API built on top of Mero:
  - Handle system conf, services management

I – Sage2 Storage Platform



I – Sage2 Software stack



I – Sage2 Objectives

- WP2: Design and implement software infrastructure needed for the Global Memory Abstraction on persistent storage tiers

- "The work package also delivers the necessary studies and support for accommodating and managing data with a new generation of Non Volatile Memories both within the compute node and externally."

- Tasks:
  - First year: Global memory abstraction support design
  - Second year: Global memory abstraction support implementation

Outline

- I – A bit of context with Sage2
- **II – How we plan to use NVDIMMs**
  - **Overall semantics**
  - **uMMAP-IO description**
- III – One Optimization for Checkpoint optimization

II – Mapping of MERO Objects

- « Memory map » Mero objects from lower tiers:
  - Allow access to remote data
  - Mapping directly in the Memory/NVRAM of the local compute
  - Direct load/store interface can be used

II – uMMAP-IO Presentation

- Complexity of interface between all tiers …
- How to have a transparent interface for applications ?

Buffer-based

```
obj = clovis_obj_init(&id, …);
…
clovis_obj_op(&obj, CLOVIS_READ, …);
…
clovis_obj_op(&obj, CLOVIS_WRITE, …);
```

I/O happens on read/write calls

mapped object (load/store)

```
obj = clovis_obj_init(&id, …);
…
base = mmap(&obj, O_WRITE, size, …);
strcpy(base, “hello there”);
…
msync(base,size, 0);
```

I/O happens on:

- Read/Write page faults
- Flush to persistence

II – uMMAP-IO Presentation

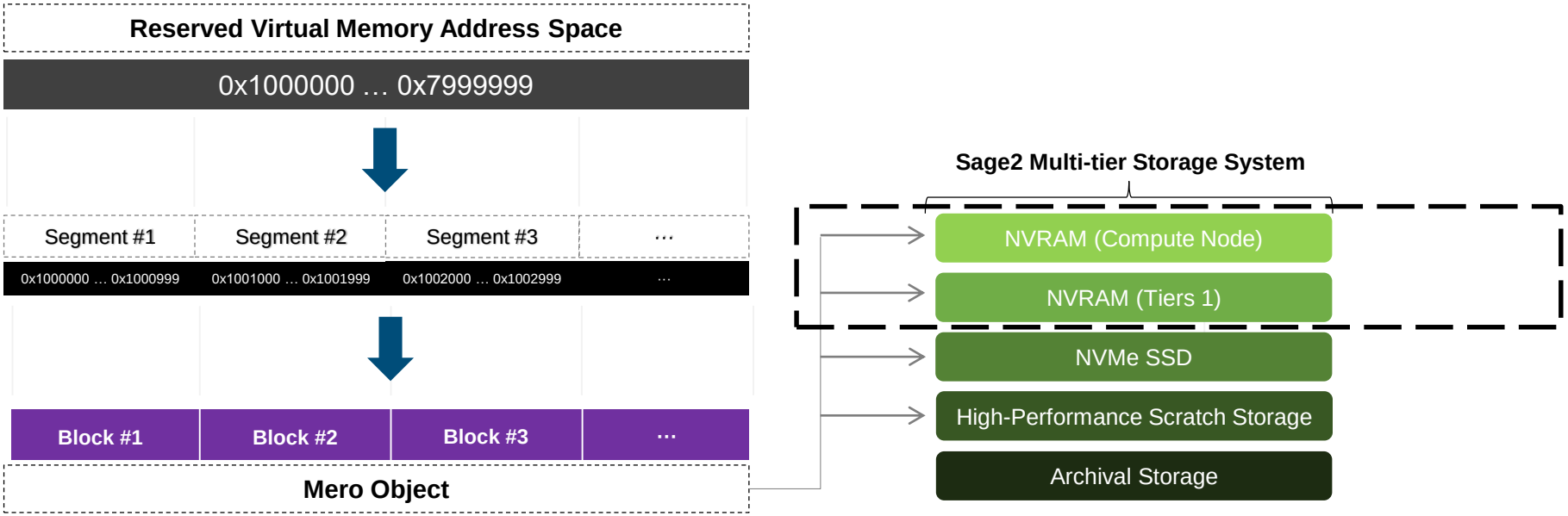
- uMMAP-IO[1]: memory mapping in user-space instead
- Allow parameters configuration:
  - Segment Size
  - Synchronization mechanism with storage
  - Eviction policy of memory segments

Similar interface as standard mmap

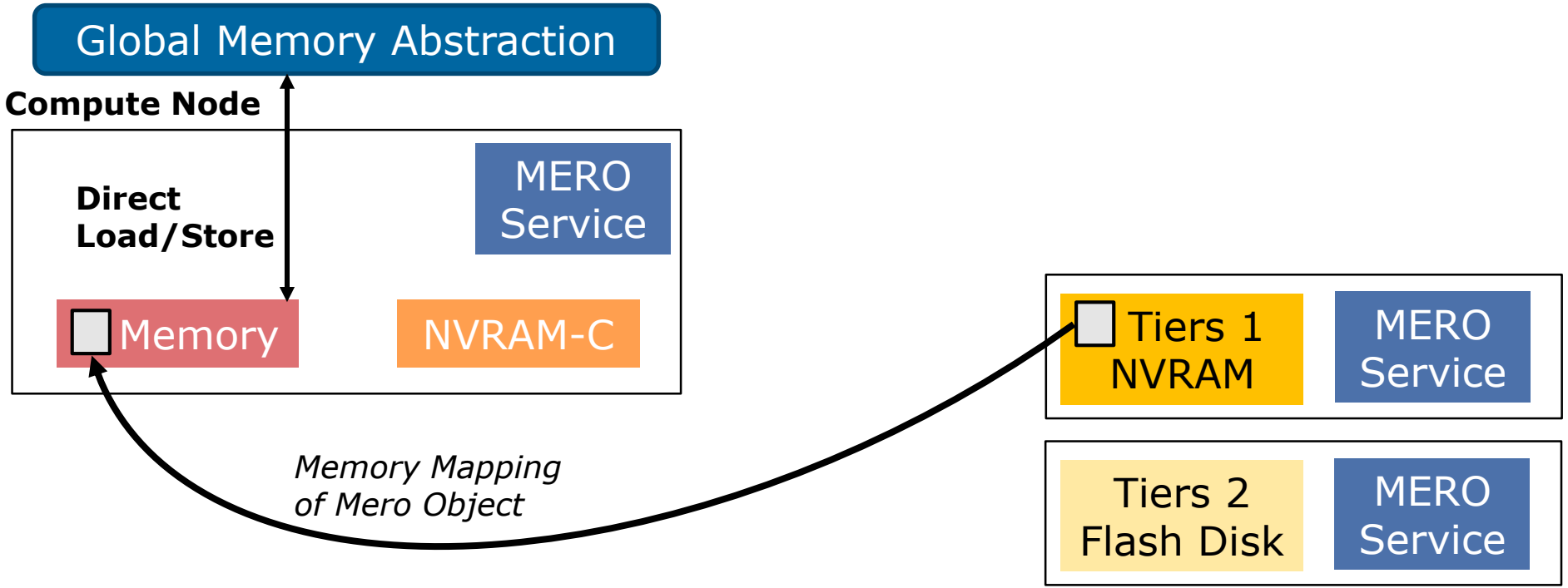
```
int ummap(size_t size, size_t seg_size, int prot, int fd, off_t offset, int flush_interval,
int umsync(void *addr, int evict);
int umremap(void *old_addr, size_t size, int fd, off_t offset, int sync, void **new_addr);
int umunmap(void *addr, int sync);
```

[1] *uMMAP-IO: User-level Memory-mapped I/O for HPC,* S.R. Gomez et Al. **HiPC 2019**
https://github.com/sergiorg-kth/ummap-io

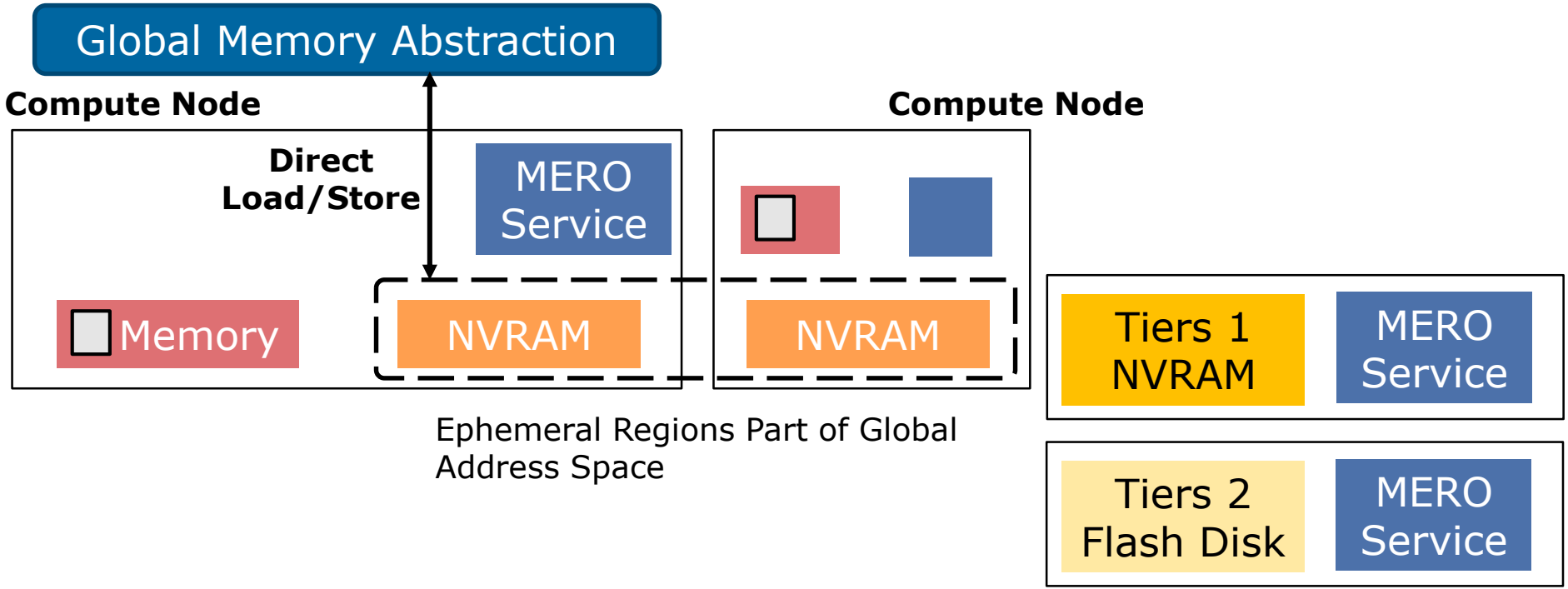
II – uMMAP-IO Presentation



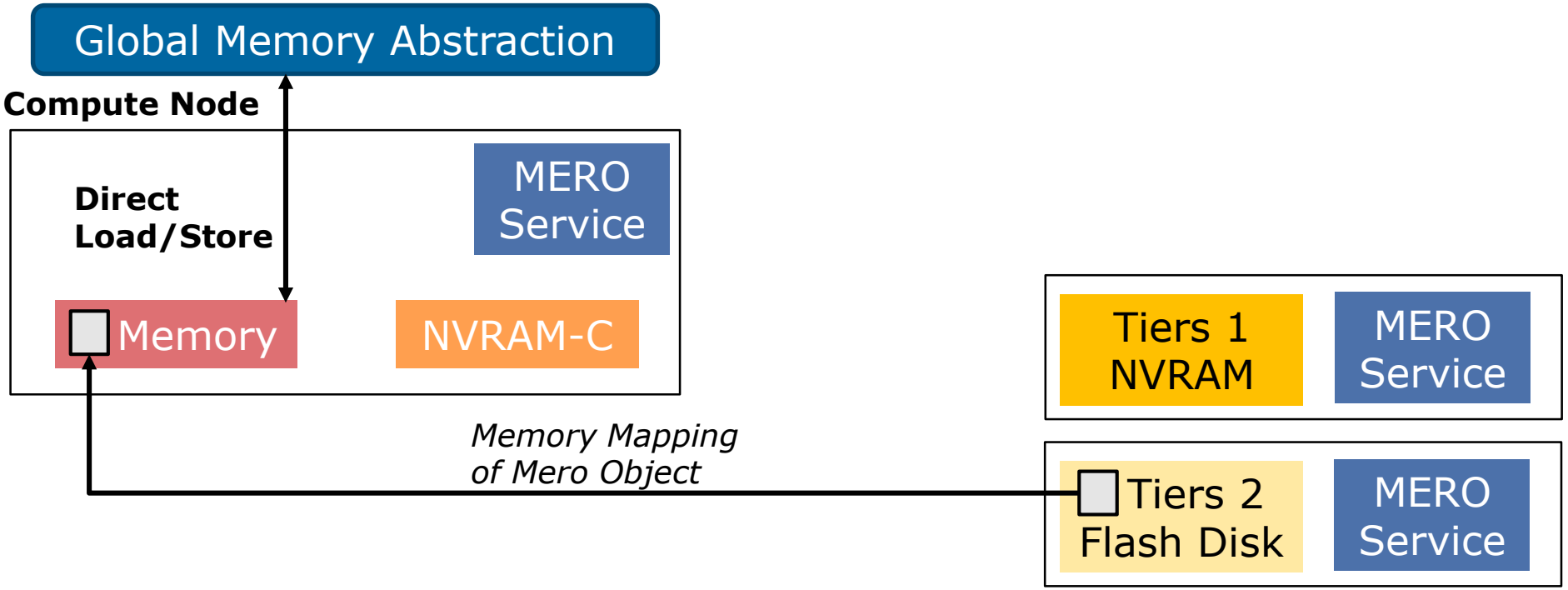
II – 1st Scenario: Mapping NVRAM from Tiers 1



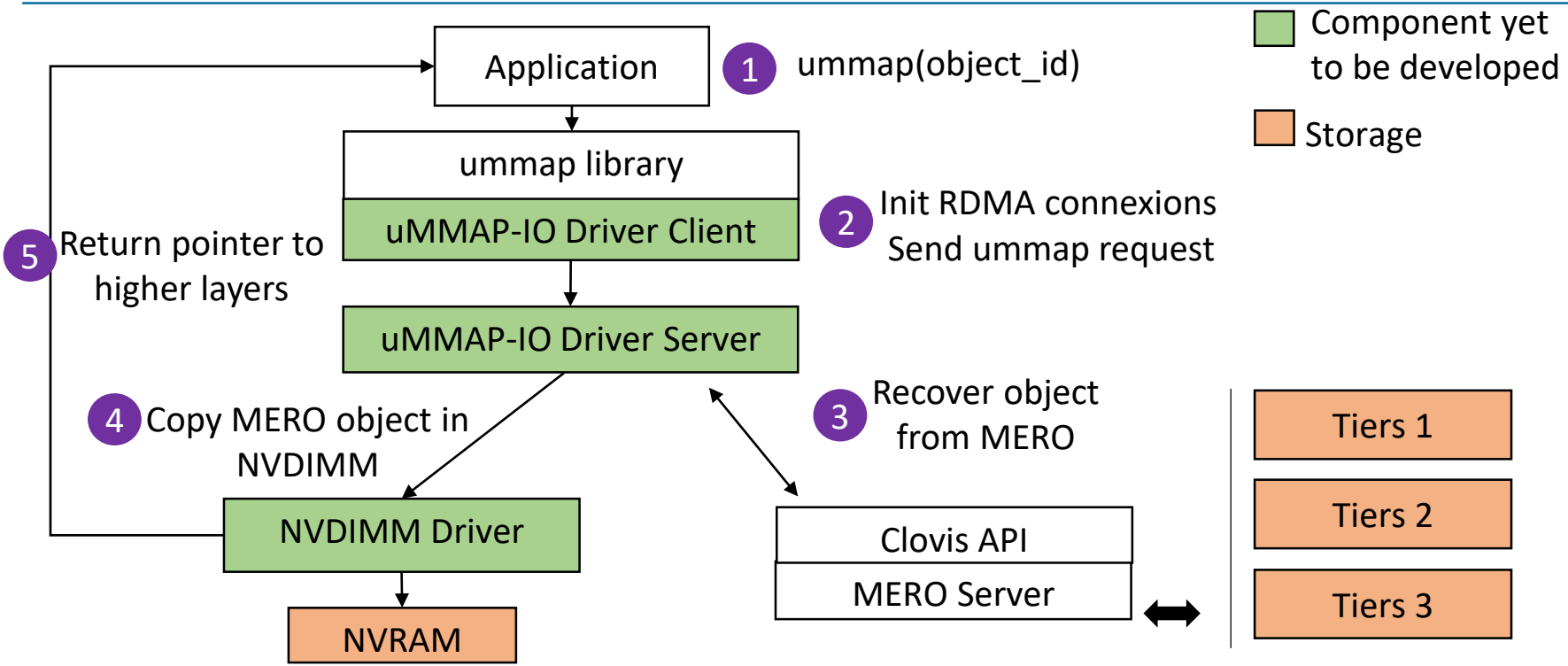
II – 2nd Scenario: Mapping NVRAM from Tiers 0



II – 3rd Scenario: Mapping from Tiers 2

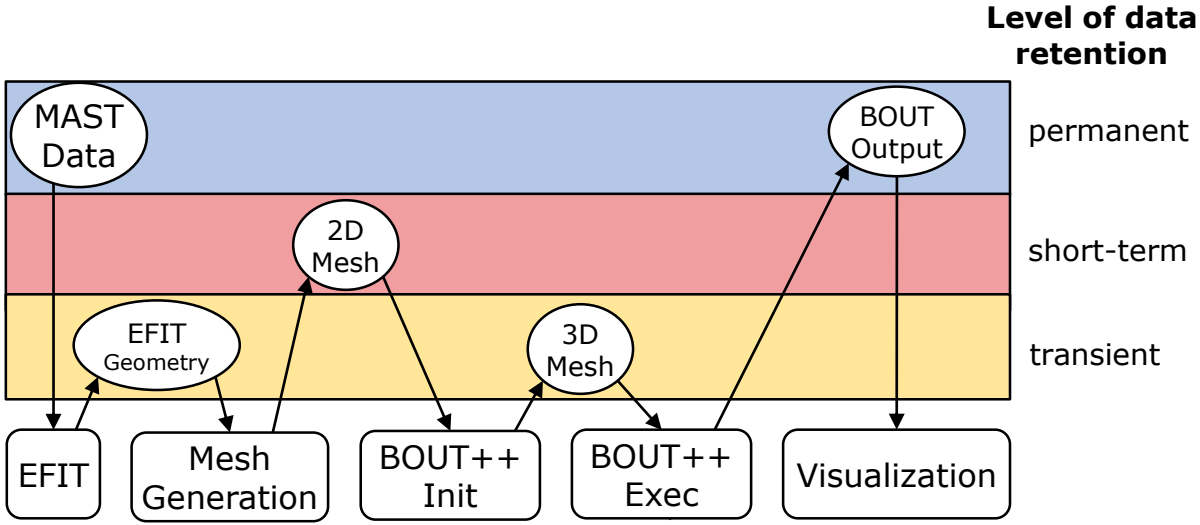


II –NVDRAM Integration in Sage2



II – Typical Scenario

- BOUT++: Simulation of Plasma
  - I/O bound, meshes can be in the order of TBs
  - Update one element at a time (~100bytes)



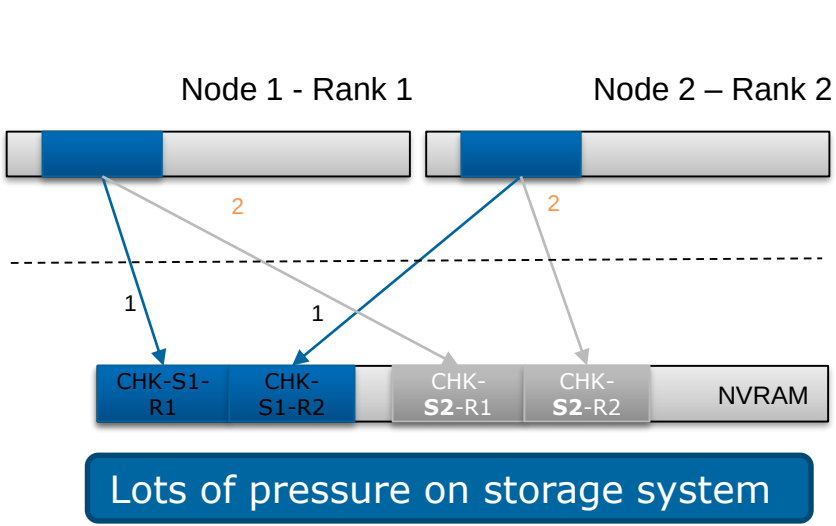
We can directly map the whole mesh in NVRAM and access it seamlessly !

Outline

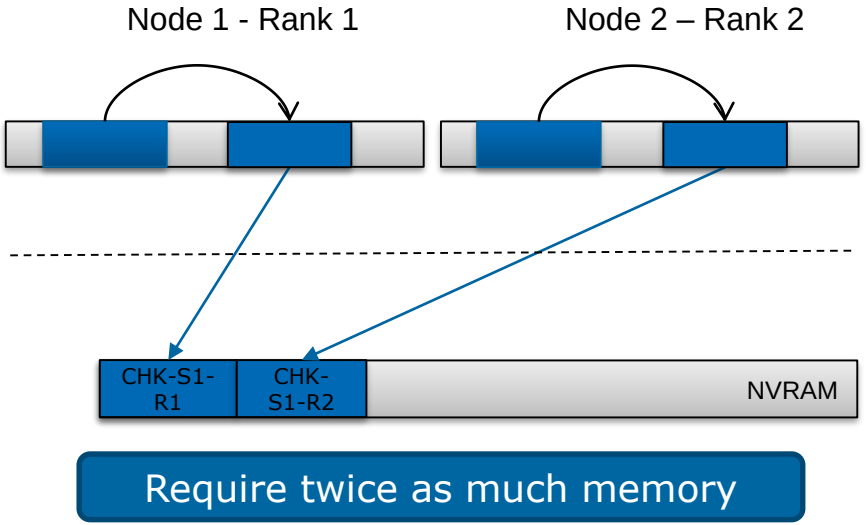
- I – A bit of context with Sage2
- II – How we plan to use NVDIMMs
  - Overall semantics
  - uMMAP-IO description
- **III – One Optimization for Checkpoint optimization**

III – Checkpoint Problem

- **Checkpoint restart Problem:**
  - Tend to dump too much data
  - Complexity of full automation, no unique standard way to do it



Lots of pressure on storage system

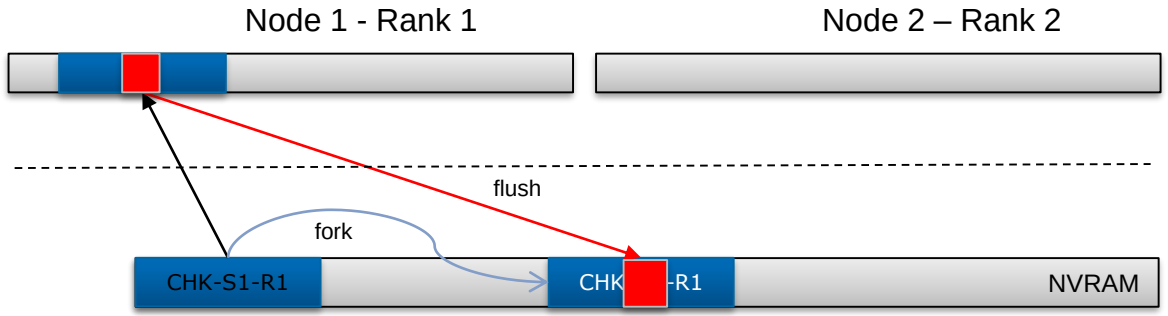


Require twice as much memory

III – Object fork extension

- New semantic: uMMAP-IO_fork()
  - clones the remote object
  - Reattaches the new one through memory mapping
  - Similar to copy on write
- Flush only send the modified segment to lower Tier

```
obj = obj_create()
mesh = obj_uMMAP-IO (obj)
compute()
obj_uMMAP-IO _fork(mesh)
obj_uMMAP-IO _flush(mesh)
compute()
…
```



Conclusion

- Our approach: Mapping memory addresses to Storage devices including NVRAM memory
  - With ummap-IO, we get lots of flexibility
- Opportunity to build new semantic that could help developpers with their applications
- Entering the implementation phase:
  - Which PDMK library to start with ?