Joint Laboratory for Petascale Computation

Transparent low-overhead checkpoint for GPU-accelerated clusters

Leonardo BAUTISTA GOMEZ[1,3], Akira NUKADA[1], Naoya MARUYAMA[1], Franck CAPPELLO[3,4], Satoshi MATSUOKA[1,2]

1 Tokyo Institute of Technology,
2 National Institute of Informatic,
3 INRIA, 4 University of Illinois

Outline

- Background and motivations
- Tsubame 2.0
- Transparent GPU Checkpointing
- Checkpointing framework
- Porting to other architectures

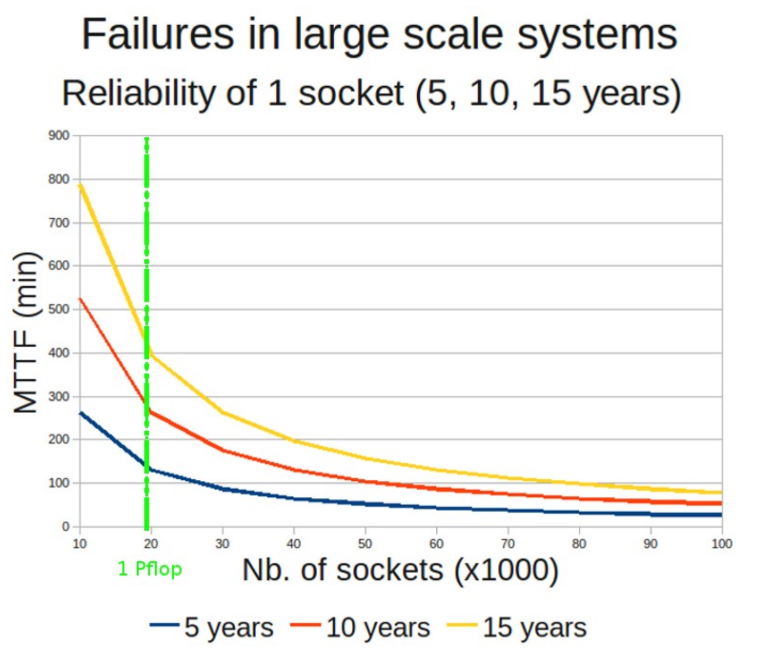
Failures

Number of failures	Type of failure	Ratio
1280	Total	100 %
1217	1 Node	95,07 %
11	2 Nodes	0,85 %
5	3 Nodes	0,39 %
2	4 Nodes	0,15 %
7	4<X<=8 Nodes	0,54 %
10	>8 Nodes	0,78 %
5	iStorage	0,39 %
19	Router	1,48 %
3	Power outage	0,23 %

Background

Jaguar has ~2.2 failures per day

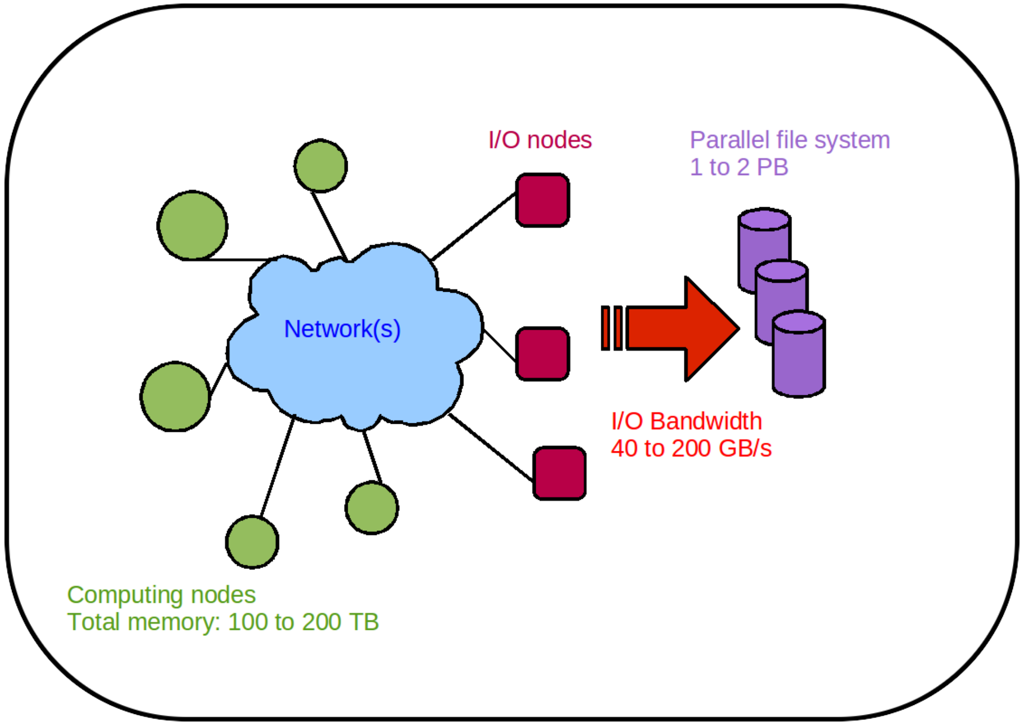
- Number of nodes increasing
- Number of components per node increasing
- Higher density
- Failures more frequent
- The checkpoint frequency must increase



Joint Laboratory for Petascale Computation

Disk-based chkpt.

- **I/O bandwidth** is a bottleneck
- FS has poor perf. for some access patterns
- **Power efficiency**



- High **file system load**
- Disk-based checkpoint will be too expensive

Tsubame 2.0

#4 in the Top 500

Multi-GPU appl.

2006
TSUBAME 1.0
85TFlops/1.1PB

2007
TSUBAME 1.1
100TFlops/1.6PB

2008
TSUBAME 1.2
160TFlops/1.6PB

2010/11/01
TSUBAME 2.0
2.4PFlops/7.1PB
日本初のペタコン、TSUBAME1.0の30倍の性能

NEW DESIGN Hewlett Packard CPU+GPU Thin Node

*** Node: High BW Compute Nodes x 1408**
*** CPU: Intel Westmere-EP 2.93GHz 12Cores/node**
*** Mem: 55.8GB (=52GiB)**
103GB(=96GiB)
*** GPU: NVIDIA M2050 515GFlops, 3GPUs/node**
*** SSD 60GB x 2 120GB ※55.8GB node**
120GB x 2 240GB ※103GB node
*** OS: Suse Linux Enterprise + Windows HPC**

4224 NVIDIA "Fermi" GPUs
Memory Total ： 80.55TB
SSD Total ： 173.88TB

CUDA checkpoint

- The problem: Transparent checkpoint libraries such as **BLCR do not support CUDA applications**
- Solution:
  - Send all data from device to host
  - Destroy all CUDA context
  - Make BLCR checkpoint
  - Reallocate CUDA resources
  - Restore the data on the device

Joint Laboratory
for Petascale Computation

CUDA checkpoint

- The problem: After reallocation, the address or handle may differ from previous one.
- Handles are opaque pointers: can be virtualized
- Addresses must be **visible** for user application and may be **kept in the device** memory space
- The **memory address** of a device memory region **reallocated after checkpointing must be the same as that before checkointing**

Replay strategy

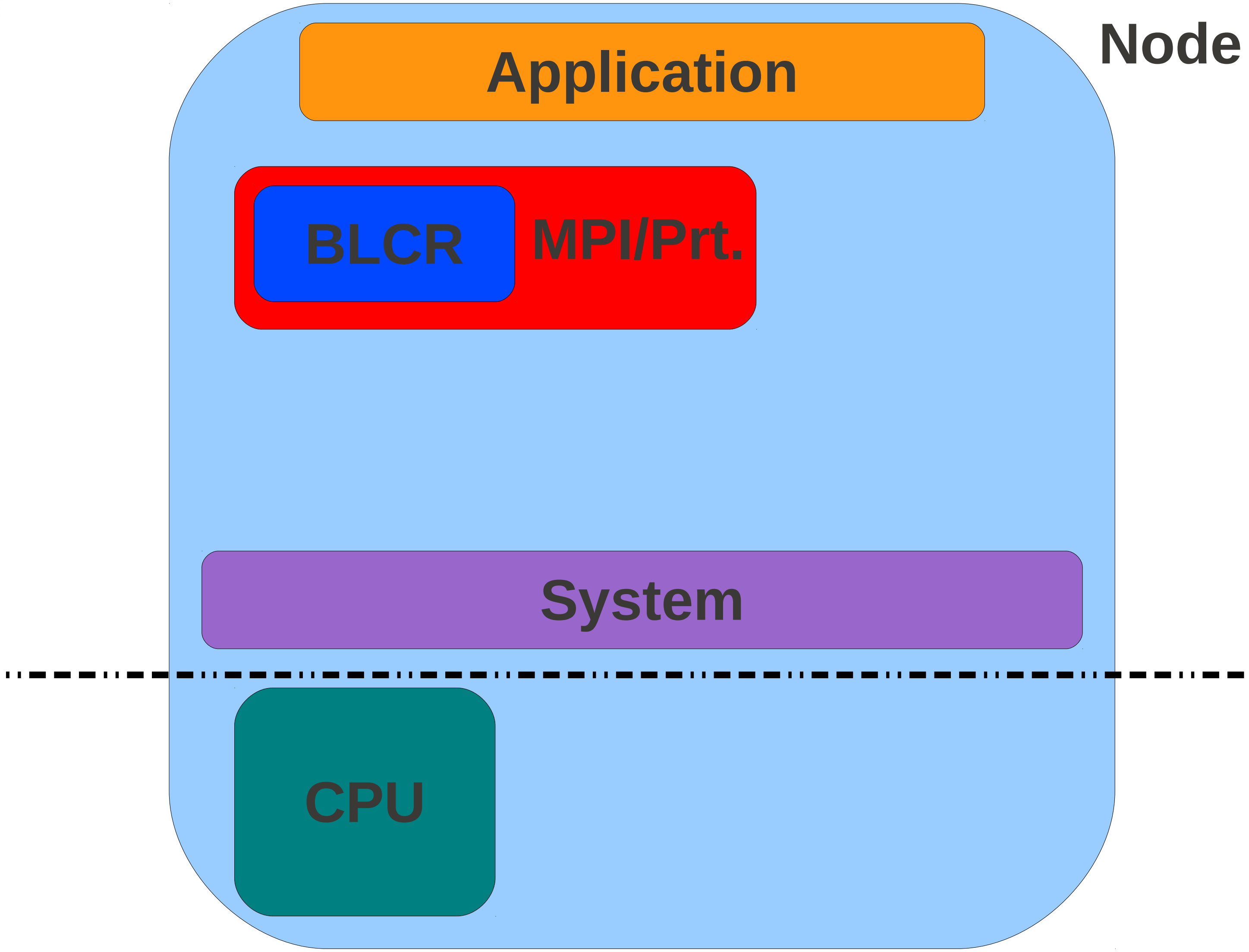
- We **record all API calls** related with the device memory address (such as cuMem*(), cuArray*(), cuModule*()) and **replay them when restoring**
- We **optimize by removing** locally resolved alloc-free **pairs from the record**
- The **time for replying is negligeable** compared with the BLCR checkpointing procedure

Joint Laboratory
for Petascale Computation

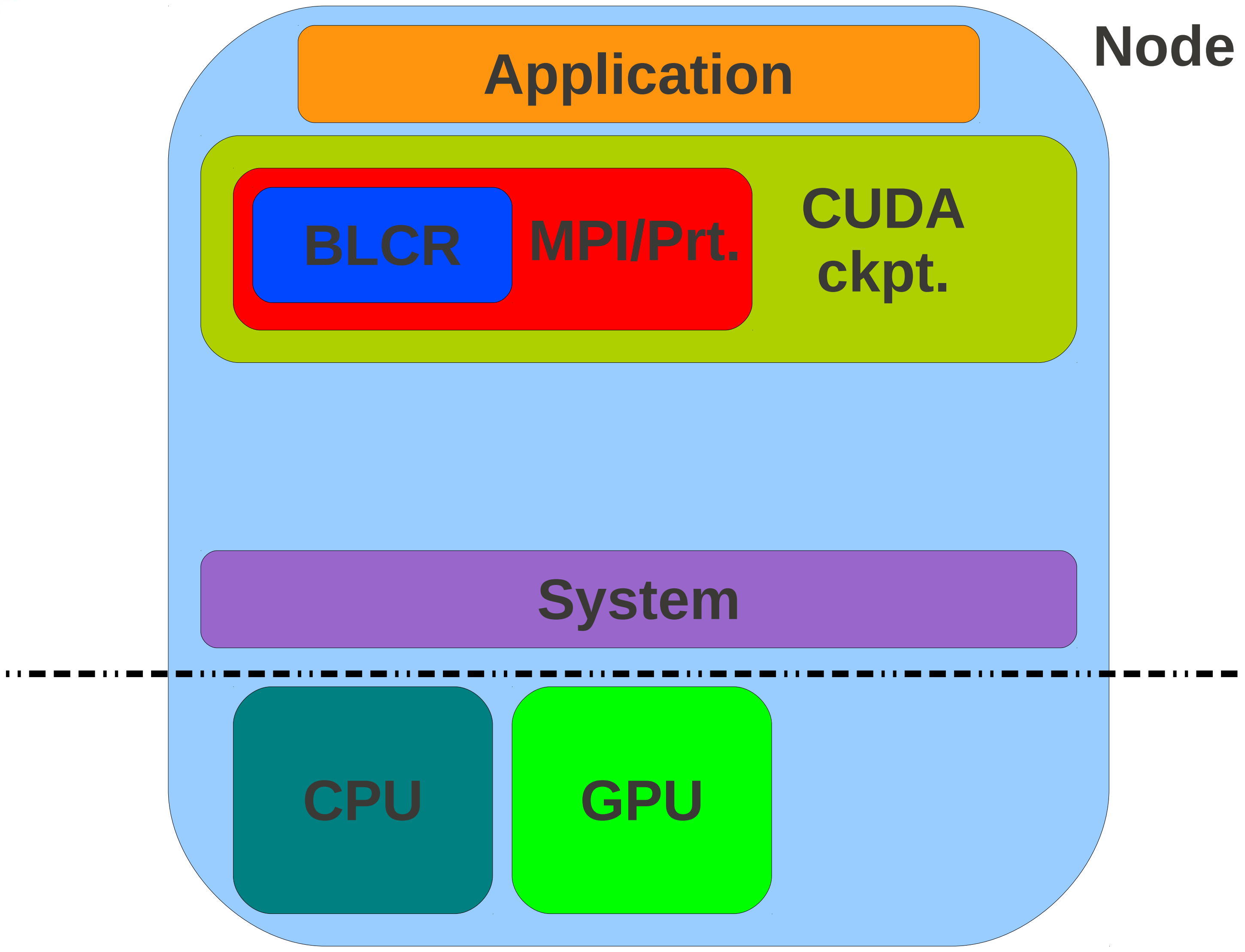
Pinned memory

- **Pinned memory is a memory region on host memory** that improve data transfers by DMA cont.
- We have to reallocate the pinned memory at the **same address as before checkpointing**
- Replay strategy is not possible in this case.
- If we don't use pinned memory we have to **serialize data transfers and kernel executions.**
- malloc()-ed memory to CUDA pinned memory?

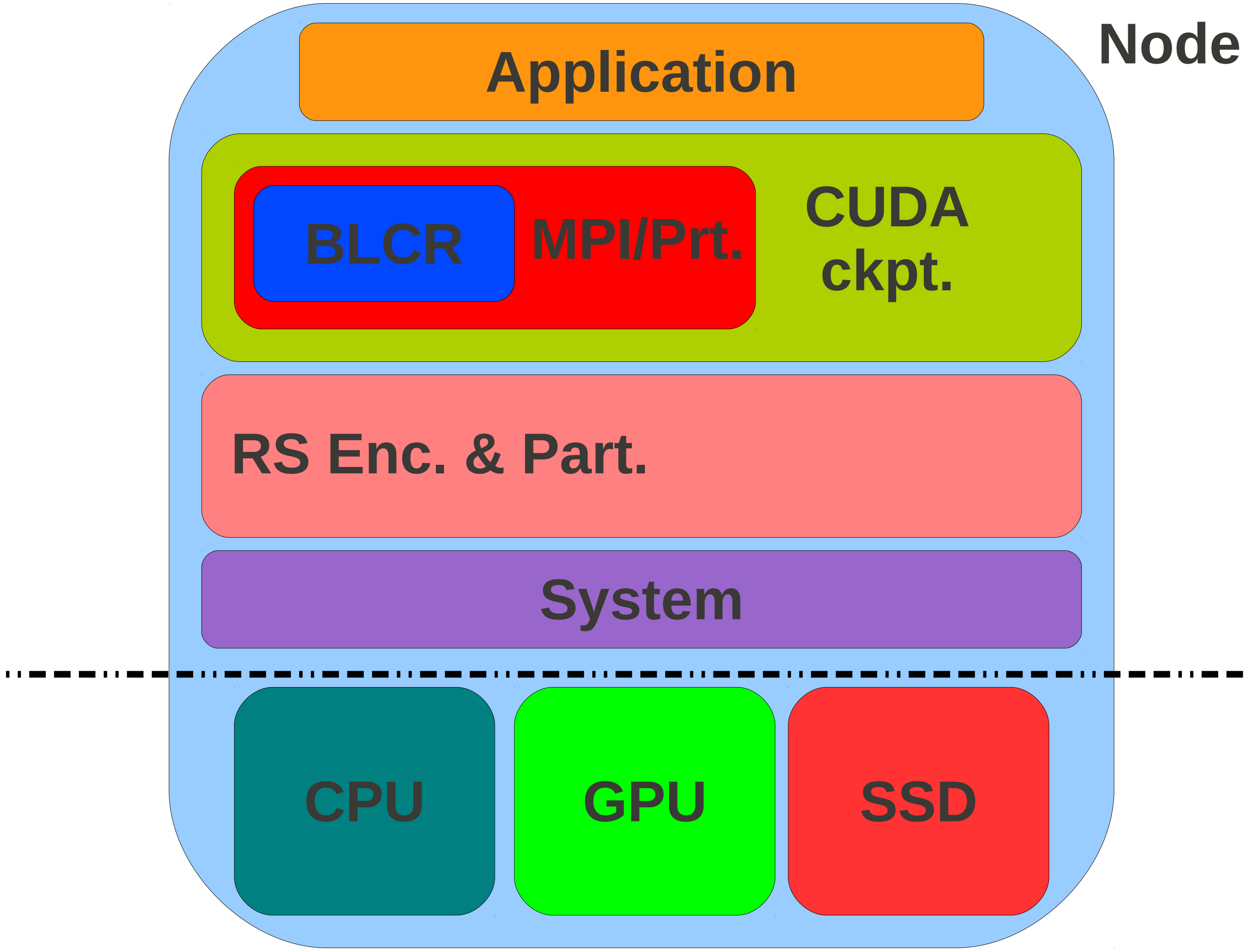
Ckpt. framework



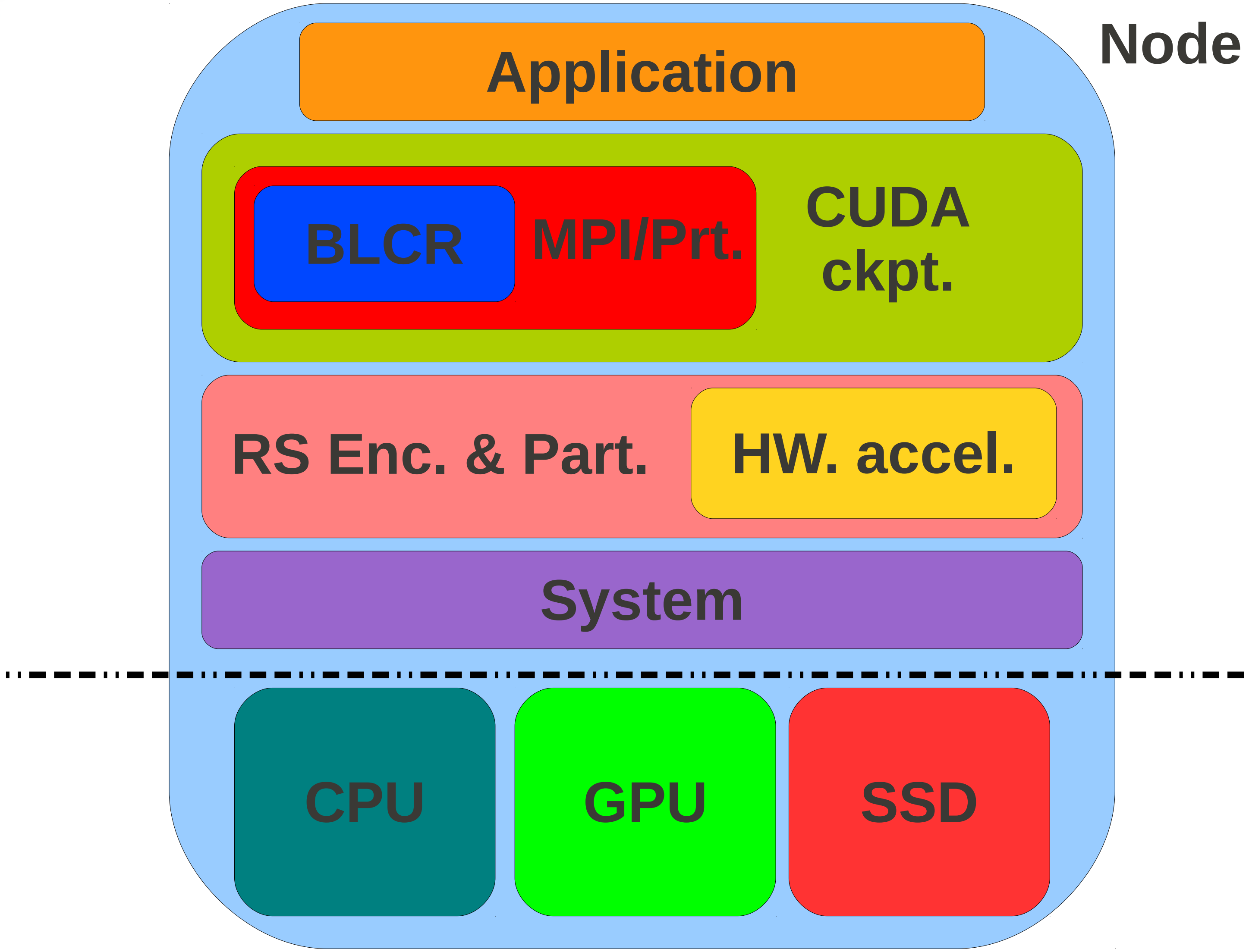
Ckpt. framework



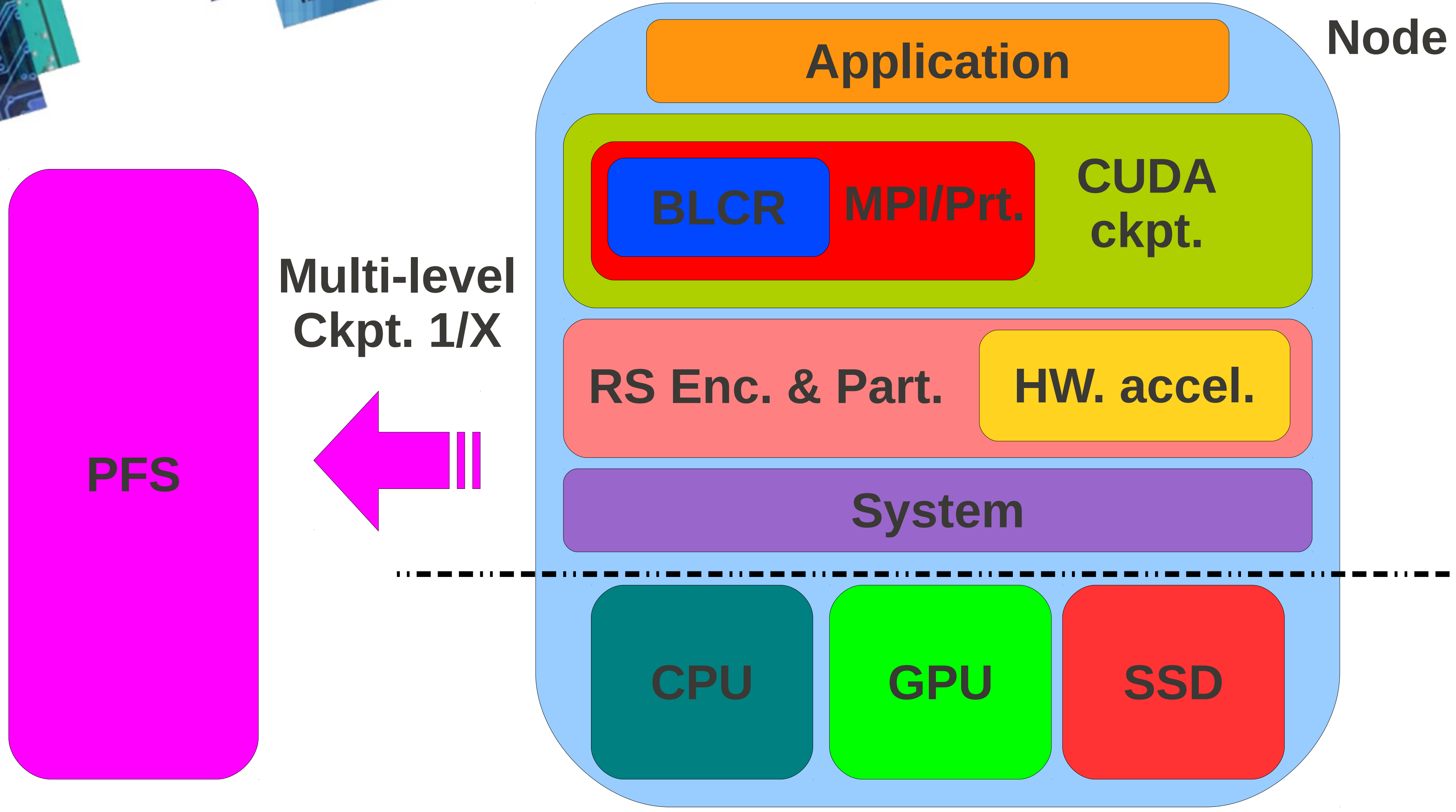
Ckpt. framework



Ckpt. framework



Ckpt. framework



Other architectures

- Faster than Remote-disk checkpoint
- Decrease the I/O load

Joint Laboratory for Petascale Computation

Other architectures

- Faster than Remote-disk checkpoint
- Decrease the I/O load
- Not-heterogeneous systems can take advantage of this checkpoint framework
- Dedicated Fault-tolerance threads

Other architectures

- Faster than Remote-disk checkpoint
- Decrease the I/O load
- Not-heterogeneous systems can take advantage of this checkpoint framework
- Dedicated Fault-tolerance threads
- In-memory checkpoint if not SSD/HDD
- Extra nodes to deal with memory limits

Summary

- We need more information about failures in HPC
- Better understanding of different kind of failures
- Portable fault tolerance techniques

Joint Laboratory for Petascale Computation

INRIA NCSA

Thank you

Questions?