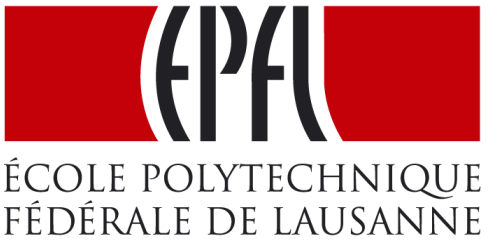
Replication for Large-Scale MPI Applications

Arnaud Lefray, Thomas Ropars, André Schiper

Distributed Systems Laboratory



Failure is a Major Concern at Scale

- One of the main challenges for future Exascale systems
  - MTBF of a few hours (or even tens of minutes)
  - Increased rate of undetected errors
    - Not our concern in this talk
- The strategy based on global checkpoints on a Parallel File System will not work
  - Too much time to save a checkpoint
  - Too much time to recover

If checkpointing is so costly, why not using replication?

- *Replication (Duplication) can make sense if*
  - *More than 50% of the time is spent dealing with failures*

- *"Evaluating the viability of Process Replication Reliability for Exascale Systems" (SC11, Ferreira et al)*
  - Combining global checkpoints and dual replication:
    - Reduces the risk of rollback
    - Reduce checkpoint frequency
  - Replication has advantages at scale but:
    - Compared to **basic** checkpointing strategies

Improved Checkpointing Strategies

- Multi-level checkpointing
  - Use of intermediate storage to save checkpoints
    - RAM, local disks
  - Encoding techniques
- Hierarchical checkpointing protocols
  - Failure containment
  - Combines coordinated checkpoints with message logging
- Use of applications' communication characteristics
  - Send-determinism
  - Uncoordinated checkpoints without domino effect

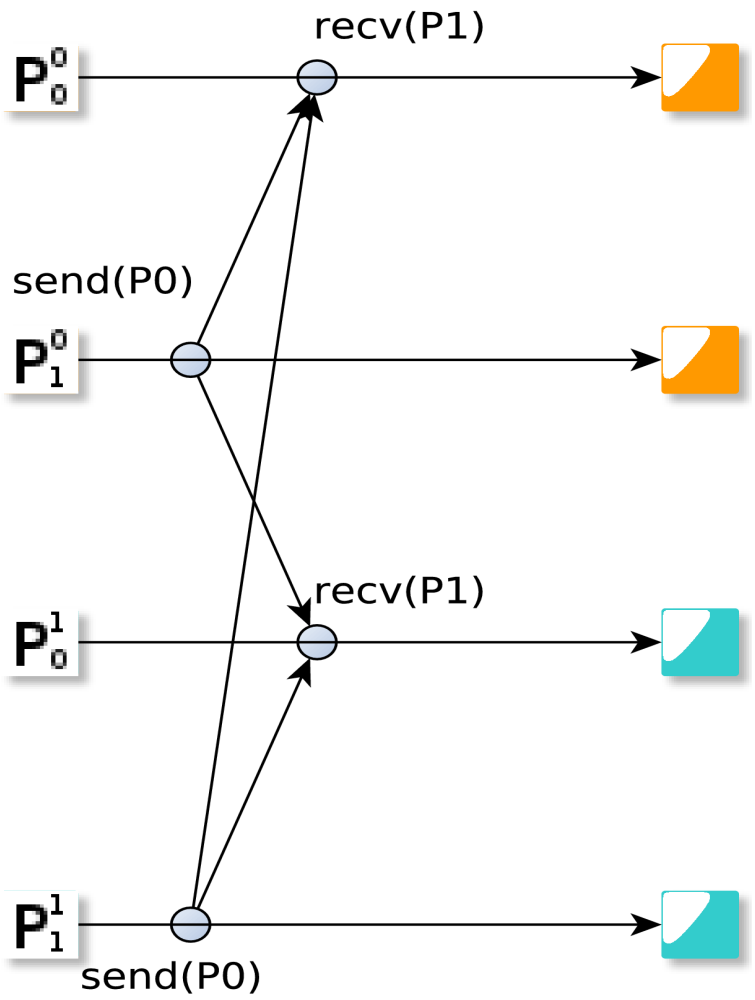
Can we also improve replication techniques?

Contributions

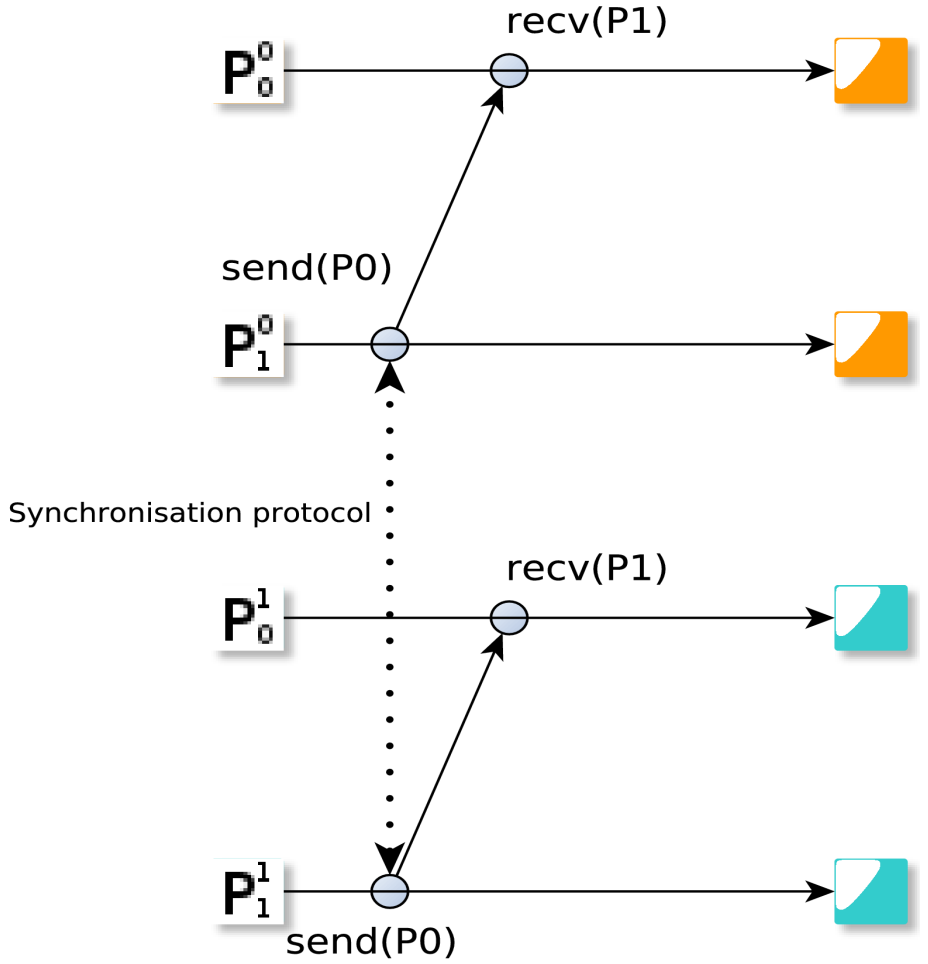
- Improved replication protocol for MPI applications
  - SDR-MPI: send-deterministic-based replication
  - Prototype in Open MPI
  - High performance (less than 5%)

- A protocol to share work between active tasks
  - Intra-MPI: interface to share work
  - Breaks the 50% max efficiency: up to 80% efficiency

Replicated MPI Protocol



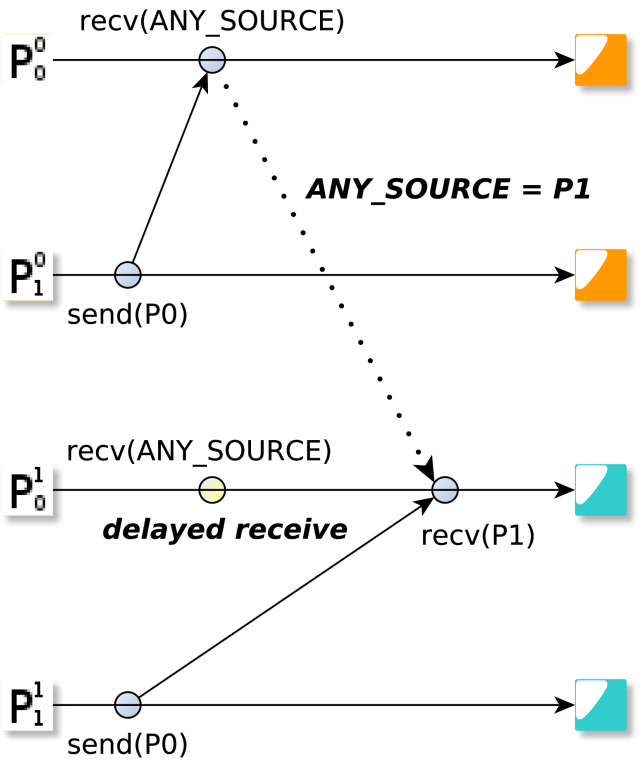
Mirror Protocol



Parallel Protocol

Existing Replication Solutions

- Reliable delivery of messages
  - Parallel or Mirror protocol
  - **Our solution: parallel protocol with acks by the receivers**
- Total order message delivery
  - Leader-based protocol
  - **Our solution: Leveraging send-determinism**
- Implemented at the PMPI level
  - Fully independent of the MPI library
  - Low performance + incomplete MPI support
  - **Our solution: integration in Open MPI**



Leader-based protocol

Send-determinism

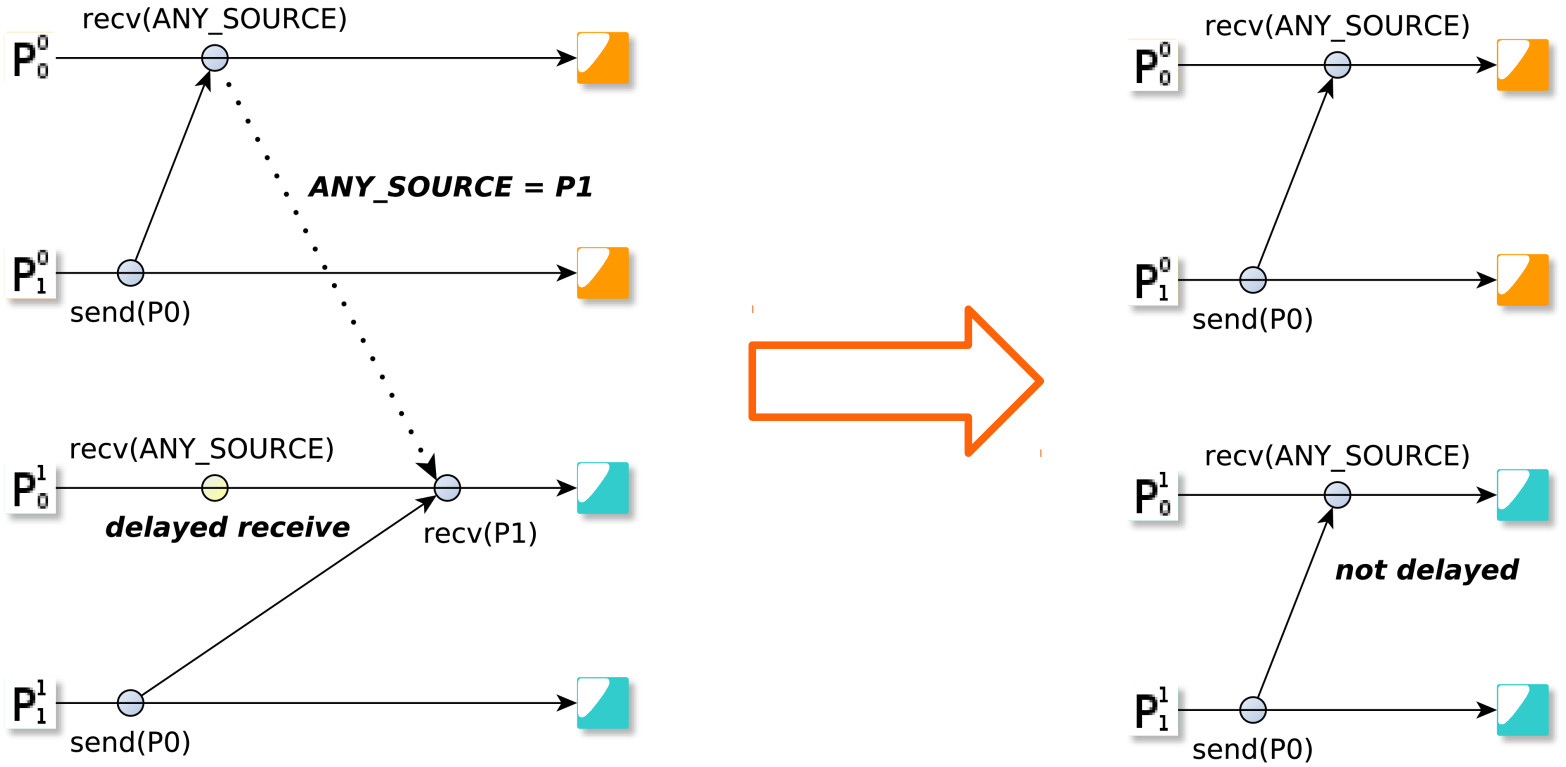
- Definition: *For a given set of input parameters, the sequence of messages sent by a process of a send-deterministic application is the same in any correct execution.*

- Study of a representative set of HPC workloads [Cappello et al, 2010]
  - → Most HPC workloads are send-deterministic

- It implies that:
  - → The order of delivery of concurrent messages has no impact on the execution of the process

SDR-MPI: Replication for send-deterministic applications

- No need for a leader-based protocol
  - ➔ Messages can be delivered in a different order on different replicas

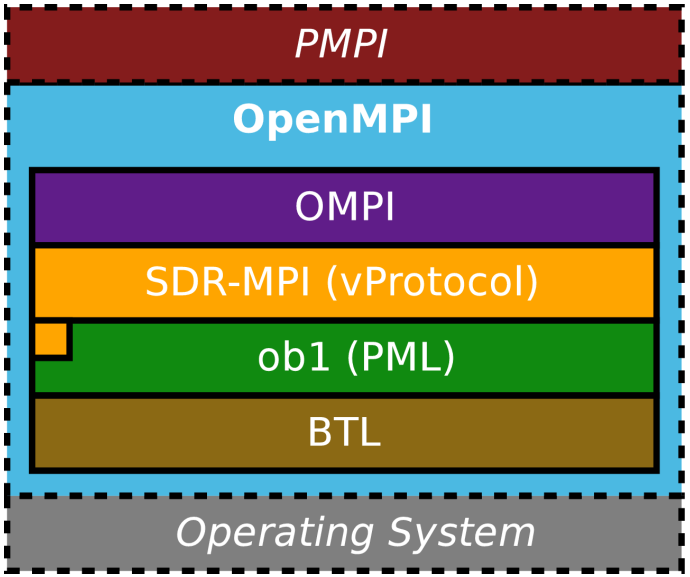


Leader-based protocol

No leader

Implementation in Open MPI

- On top of the the point-to-point management layer (PML)
  - Use of the vProtocol interface
- Slight modification of the PML
  - Interception of the *recv_complete* event
- Characteristics:
  - Fully transparent for the application
  - Supports all collective/group operations



Performance evaluation

- Experimental setup:
  - 64 nodes (8 cores , 16 GB of memory)
  - Infiniband 20Gbps
  - Replicas on different nodes
- Failure free performance:
  - NAS benchmarks
  - HPCCG, CM1 (include any_source)

Performance evaluation

Application	Overhead
BT	1.49%
CG	4.92%
FT	3.04%
MG	2.56%
SP	2.41%
HPCCG	0%
CM1	3.14%

Less than 5% overhead with all applications

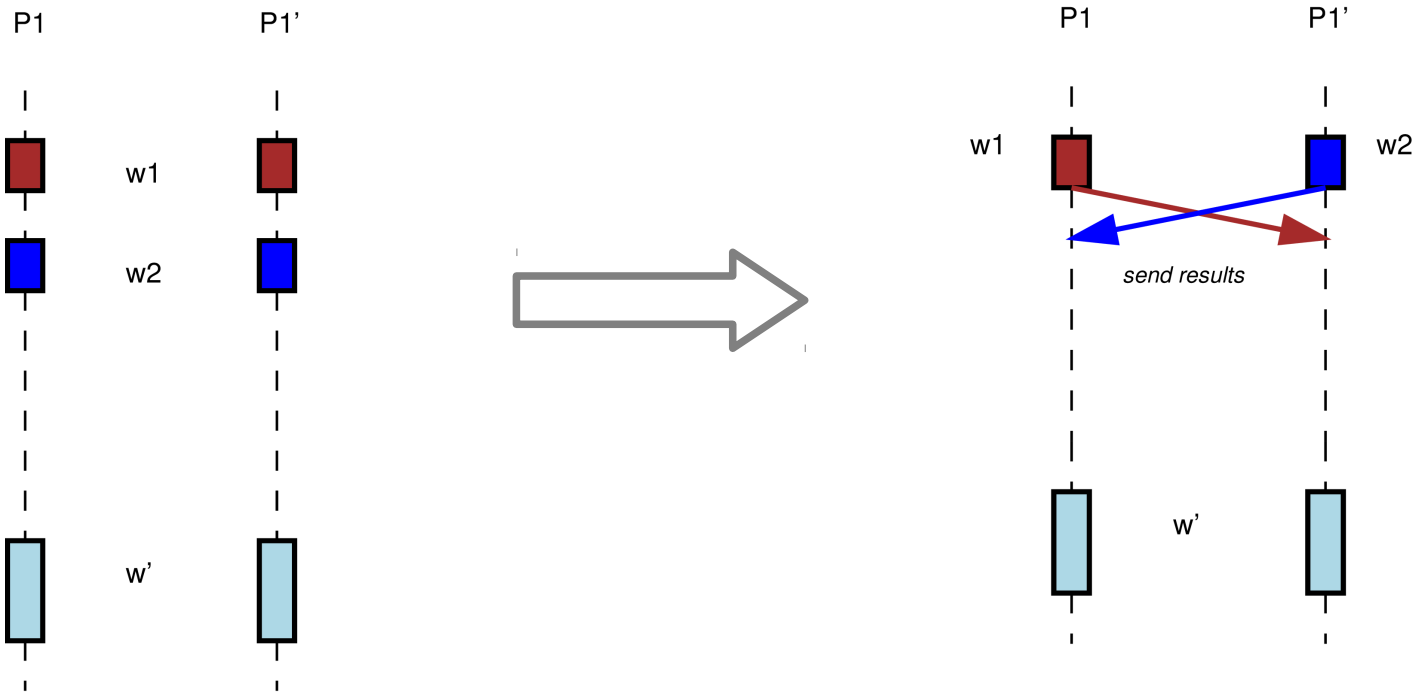
Contributions

- Improved replication protocol for MPI applications
  - SDR-MPI: send-deterministic-based replication
  - Prototype in Open MPI
  - High performance (less than 5%)

- A protocol to share work between active tasks
  - Intra-MPI: interface to share work
  - Breaks the 50% max efficiency: up to 80% efficiency

Intra-parallelization

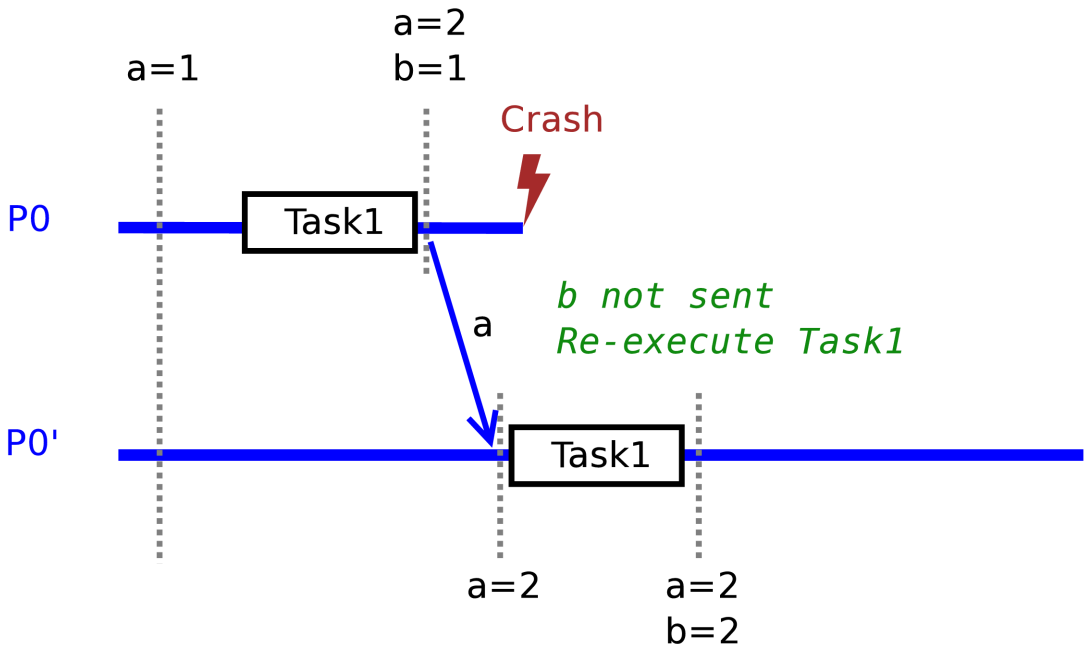
- Goal: Avoid executing two times all computation tasks
  - Run the computation on one replica and share the results with other replicas
  - A task: set of instructions that does not include communication



Handling failures: dealing with dependencies between instructions

- An example of task

```
a=1
section Section1
    task Task1
        b=a
        a=2
```

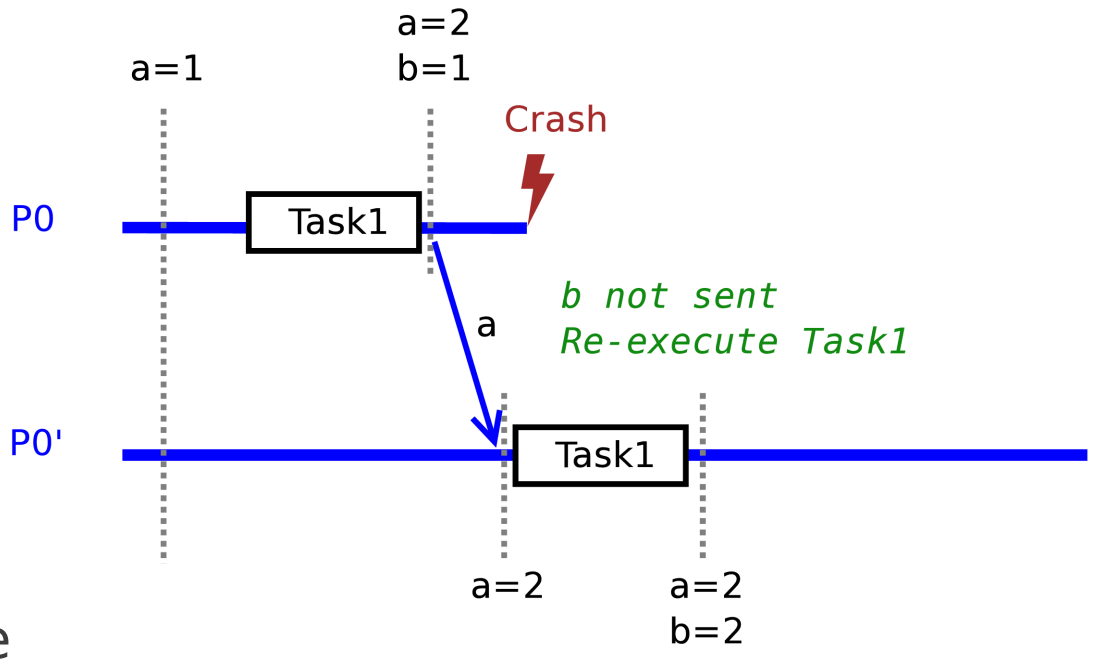


- Variable “a” is read and then modified

Handling failures: dealing with dependencies between instructions

- An example of task

```
a=1
a'=a
section Section1
    task Task1
        a=a'
        b=a
        a=2
```



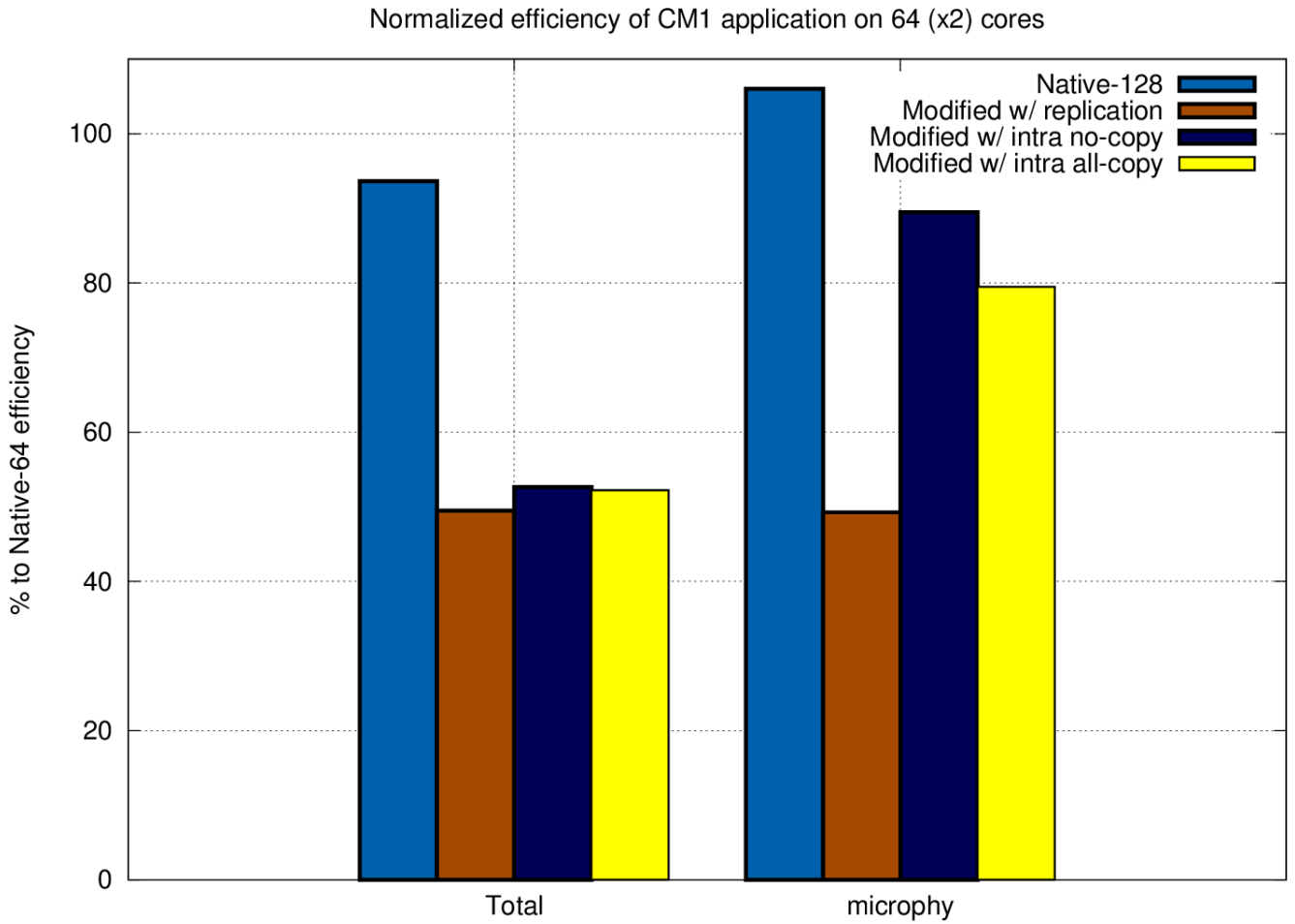
- Variable “a” is restored at the beginning of the task

Current Implementation

- Declare sections and tasks
- Declare variables
  - Outputs of the task
  - That need to be copied
- Algorithm executed by each replica
  - Get next task (deterministic)
  - If task is not completed
    - Make required data copies
    - Execute the task
    - Send results to other replicas

Evaluation with CM1 (microphy)

- 2x64 cores (infiniband)
- **80% efficiency**



Conclusion

- We manage to achieve more than 50% efficiency with replication
  - SDR-MPI: Efficient replication protocol
  - Share work between replicas
- It is different from in-memory checkpointing
  - Data is also replicated in memory
  - But no rollback (and so, no recovery)
- Can it be interesting in some use-cases?
- We would be happy to have some collaborations

Replication for Large-Scale MPI Applications

Arnaud Lefray, Thomas Ropars, André Schiper

Distributed Systems Laboratory