Yearly Team Meeting

Xavier Besseron

October 2011

About me

PhD in Computer Science

- Laboratoire d'Informatique de Grenoble (LIG) – Grenoble University
- Fault Tolerance and Dynamic Reconfiguration for large scale distributed applications
- Supervised by Thierry Gautier and Denis Trystram

1-year Postdoc

- Network Based Computing Lab – Ohio State University
- MVAPICH2 team
- Supervised by DK Panda

About me

Programmer experience

- Kaapi (Data Flow Graph on distributed platforms): Fault Tolerance, Static-scheduling, ...
- MVAPICH2 (MPI over InfiniBand): Process launcher, Checkpoint/Restart, Process Migration, ...

User experience

- Programming models/Distributed computing middleware: Kaapi/Athapascan, Charm++, ProActive, MVAPICH2, MPICH2, Open MPI
- Libraries: BLCR (Checkpoint library), FTB (Fault Tolerance Backplane), FUSE (Filesystem in UserSpace), FreeIPMI (Hardware monitoring), ZooKeeper (Distributed Coordination Service), ...
- Large scale experiments: Grid'5000 and more

Previous Research

1. Fault Tolerance protocols for Data Flow model
2. Checkpoint/Restart/Migration acceleration
3. Dynamic Reconfiguration
4. Distributed MetaData Service for Parallel Filesystems

Extremely Short Introduction to Fault Tolerance

Distributed computing / High Performance Computing

- Large scale platforms
- Big number of involved components
- Long execution time

$\Rightarrow$ Failure probability is high

Classic approaches

- Replication: replicate computation in time or space
- Message logging: save non-deterministic events of each processes (i.e. communications)
- Checkpoint/Rollback: periodically save a consistent global state of the application

Extremely Short Introduction to Fault Tolerance

Distributed computing / High Performance Computing

- Large scale platforms
- Big number of involved components
- Long execution time

$\Rightarrow$ Failure probability is high

Classic approaches

- Replication: replicate computation in time or space
- Message logging: save non-deterministic events of each processes (i.e. communications)
- Checkpoint/Rollback: periodically save a consistent global state of the application

In HPC, only Message Logging or Checkpoint/Rollback are considered
Practically, only Coordinated Checkpoint/Rollback is used

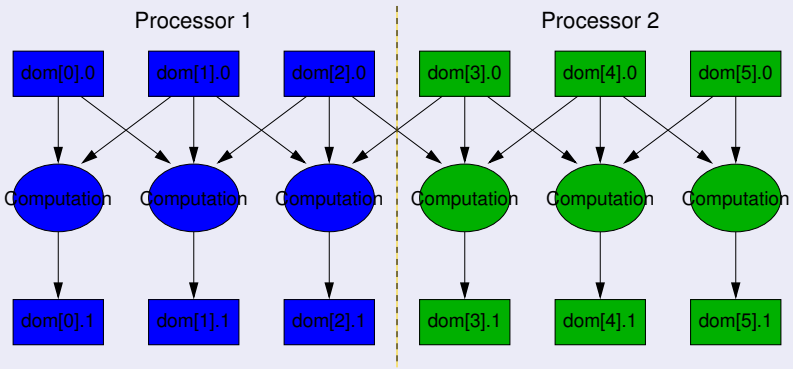
Previous research

1. Fault Tolerance protocols for Data Flow model
2. Checkpoint/Restart/Migration acceleration
3. Dynamic Reconfiguration
4. Distributed MetaData Service for Parallel Filesystems

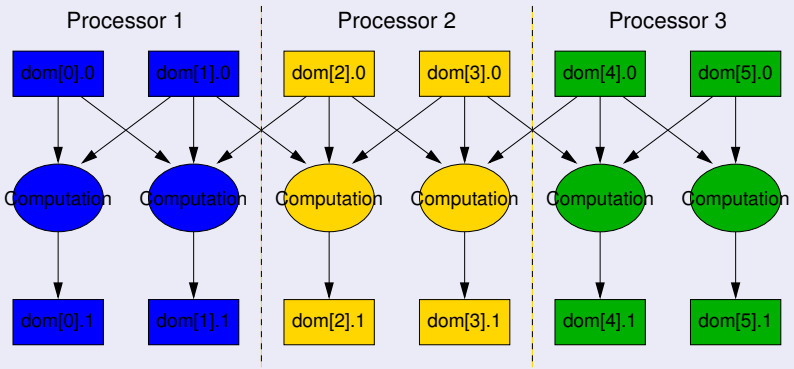
Global Rollback for Data Flow model

Over-Decomposition

⇒ Application is independent of the number of processors



Scheduling on 2 processors



Scheduling on 3 processors

Over-Decomposition for Global Rollback

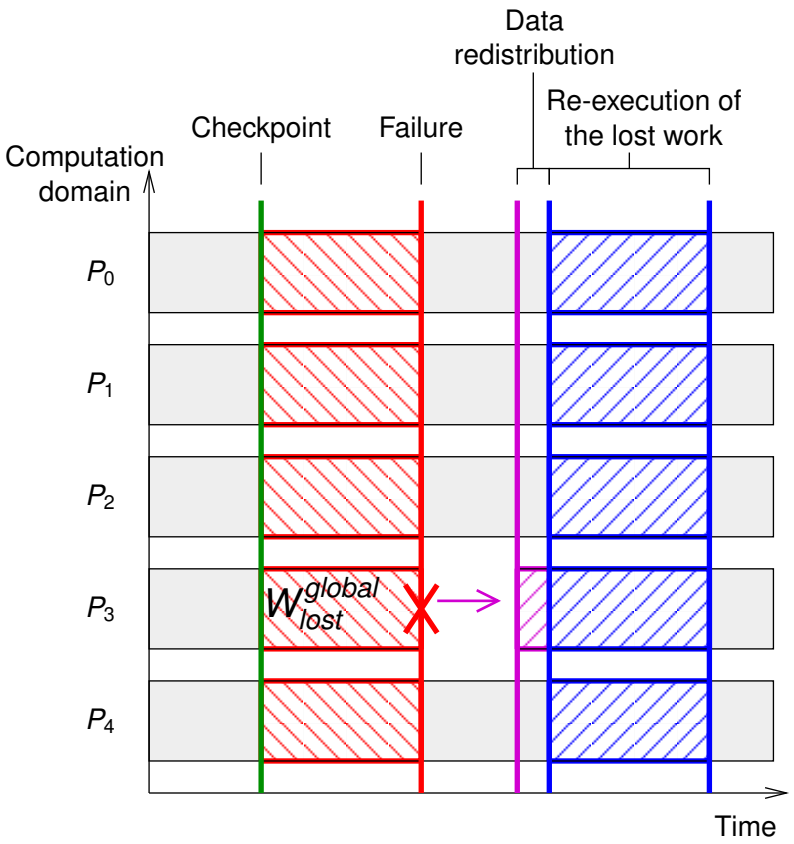
Goal is to flexibility at restart

- Checkpoint = Data Flow Graph
- Restart = Restore checkpoint + Load-balancing

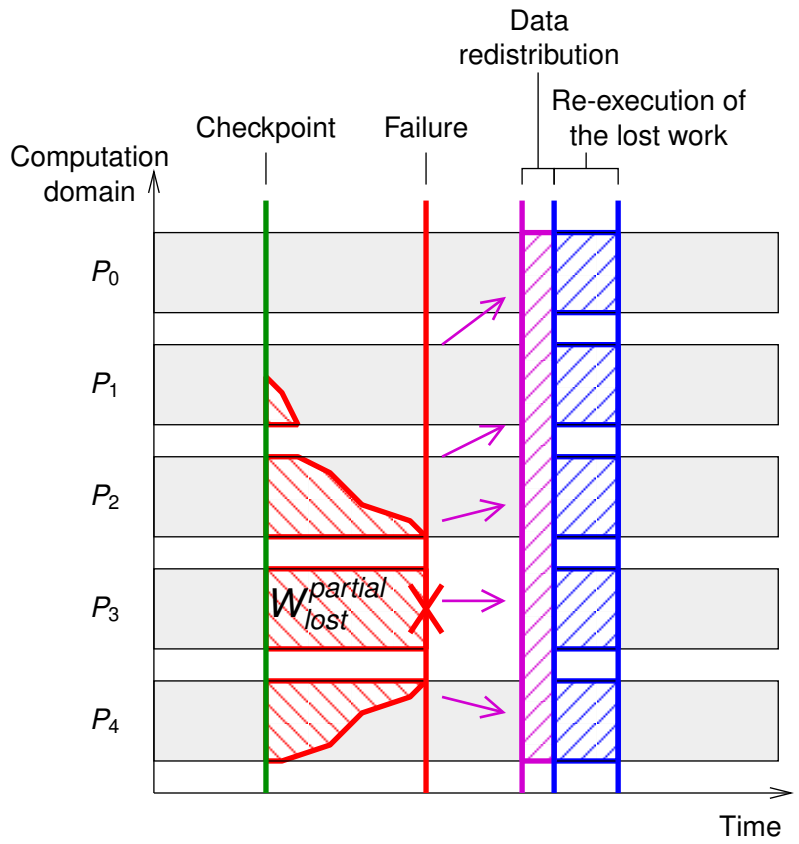
⇒ No need for spare nodes

Partial Rollback for Data Flow model

In case of failure, track computation dependencies to reduce the lost work



Global Restart



Partial Restart

Thanks to over-decomposition, the lost work can be parallelized

Previous research

1. Fault Tolerance protocols for Data Flow model
2. Checkpoint/Restart/Migration acceleration
3. Dynamic Reconfiguration
4. Distributed MetaData Service for Parallel Filesystems

Why Checkpoint Acceleration?

Let's consider a small cluster

- 64 nodes with 8 cores
- 8 GB memory per nodes
- An application with 1 process per core, using 80% of the memory
- One shared filesystem

Checkpointing the application means

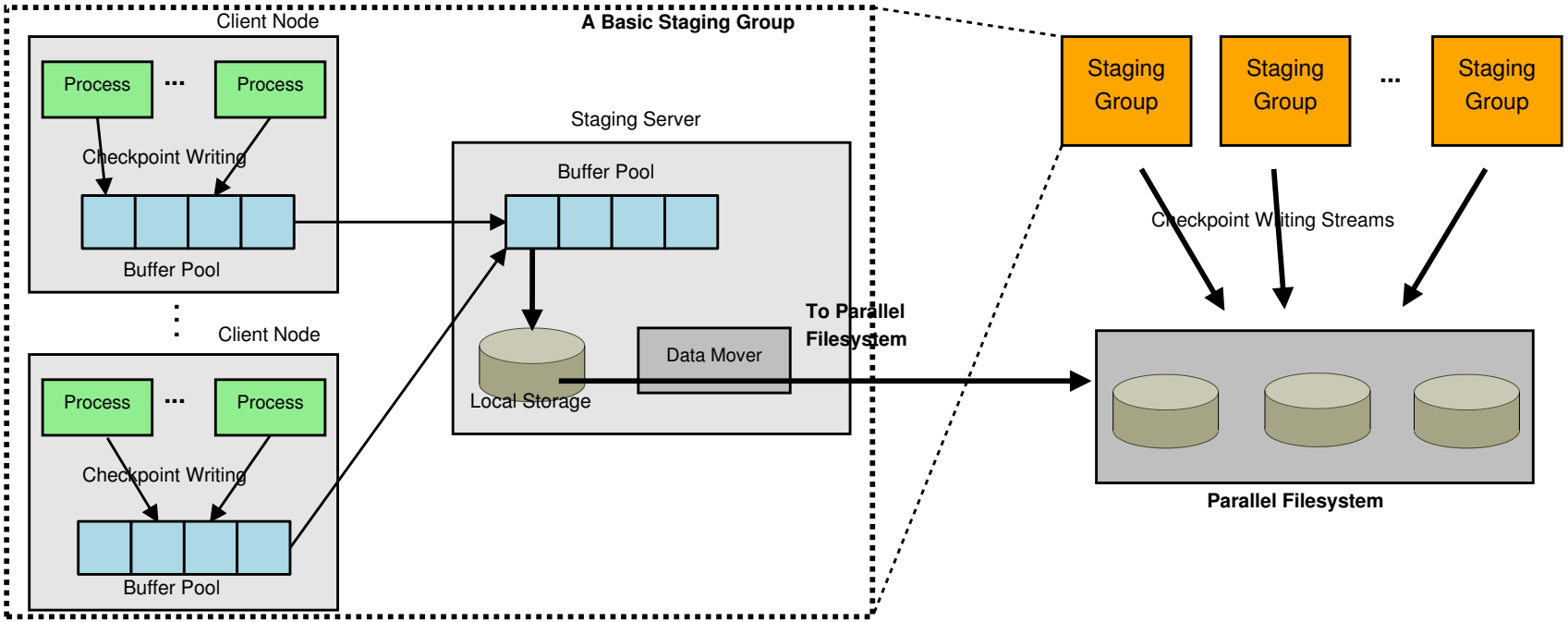
- 512 processes writing 800 MB each on the same shared filesystem
- Total checkpoint size = 400 GB

$\Rightarrow$ Heavy burden on the shared filesystem

$\Rightarrow$ Checkpoint writing represents 99% of the checkpoint time

Checkpoint/Restart/Migration acceleration

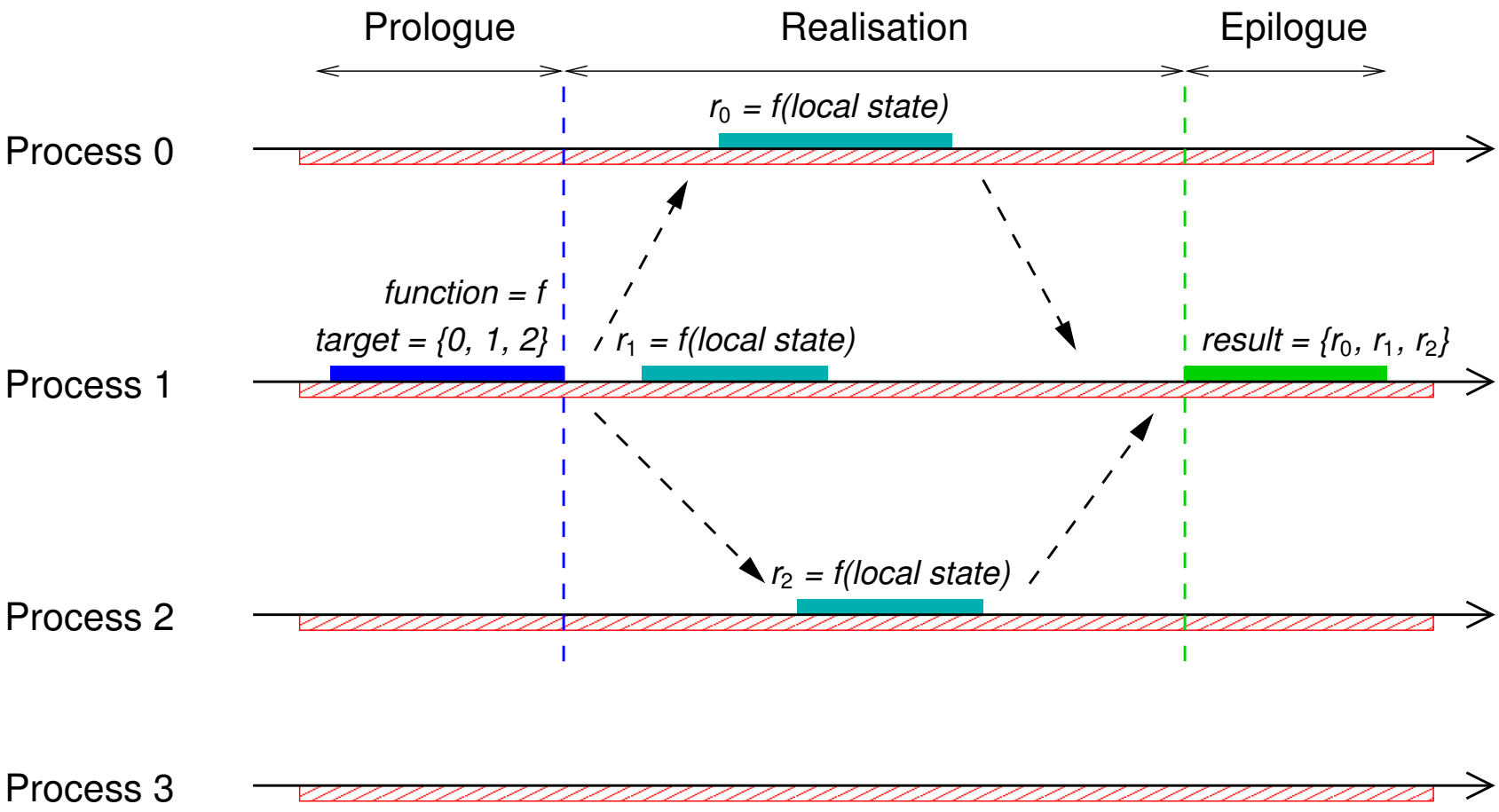
- Hierarchical Data Staging
- Write aggregation at the node level



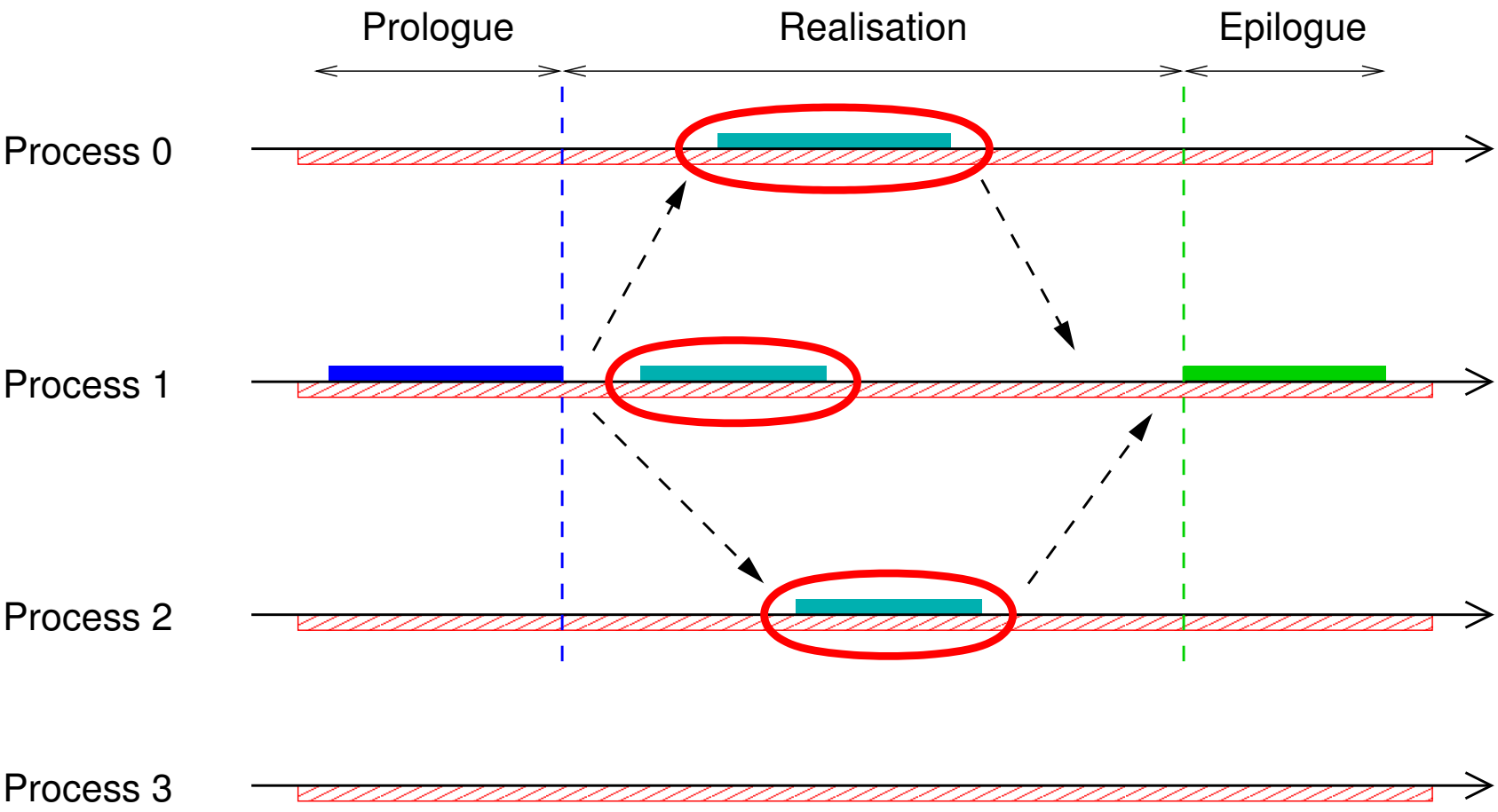
- Filesystem in UserSpace (FUSE)
- InfiniBand RDMA
- Fast storage (SSDs)

Previous research

Model of Dynamic Reconfiguration

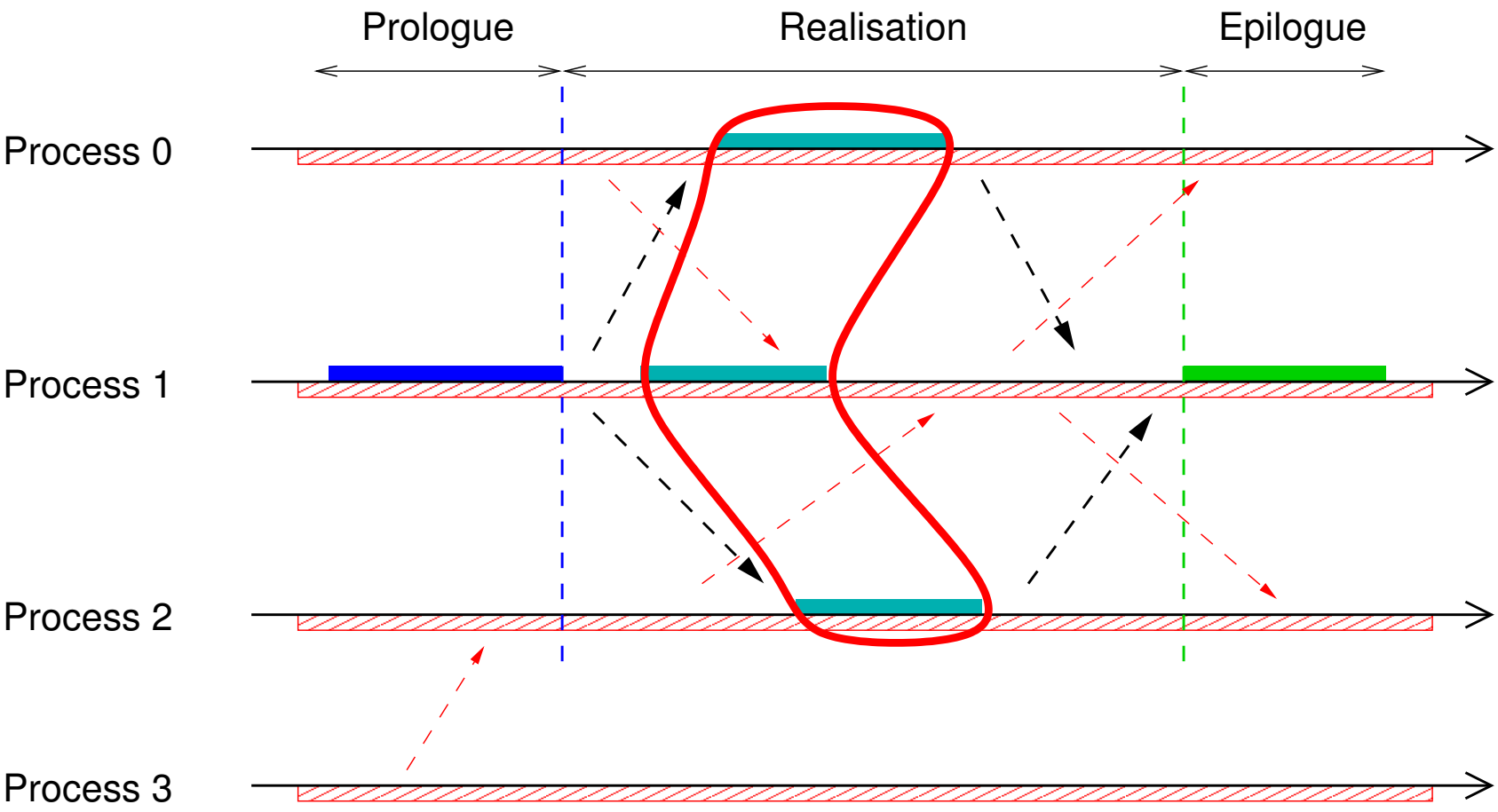


Model of Dynamic Reconfiguration



Concurrent access management inside a process

Model of Dynamic Reconfiguration



Mutual consistency management between processes

Previous research

1. Fault Tolerance protocols for Data Flow model
2. Checkpoint/Restart/Migration acceleration
3. Dynamic Reconfiguration
4. Distributed MetaData Service for Parallel Filesystems