TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Sebastian Oeste, Michael Kluge, Ronny Tschüter and Wolfgang E. Nagel
Center for Information Services and High Performance Computing (ZIH)

Analyzing Parallel Applications for Unnecessary I/O Semantics That Inhibit File System Performance

ISC-IODC 2023

ZIH
Center for Information Services &
High Performance Computing

Motivation

- POSIX IO is a long known bottleneck for performance of distributed networking file systems
- POSIX IO has no support for parallel IO (e.g. no way to open a file exclusively or collectively)
- Most high-level libraries come down to POSIX IO
- In 2005 a Working group tried to extend POSIX API for parallel computing without success
- Several distributed file systems already relaxing some POSIX semantics
  - NFS provide close-to-open consistency
  - PVFS leaves coordination of conflicting accesses to the user
  - GekkoFS has eventual consistency for some metadata operations
- Other provide special features to provide performance while being POSIX compliant

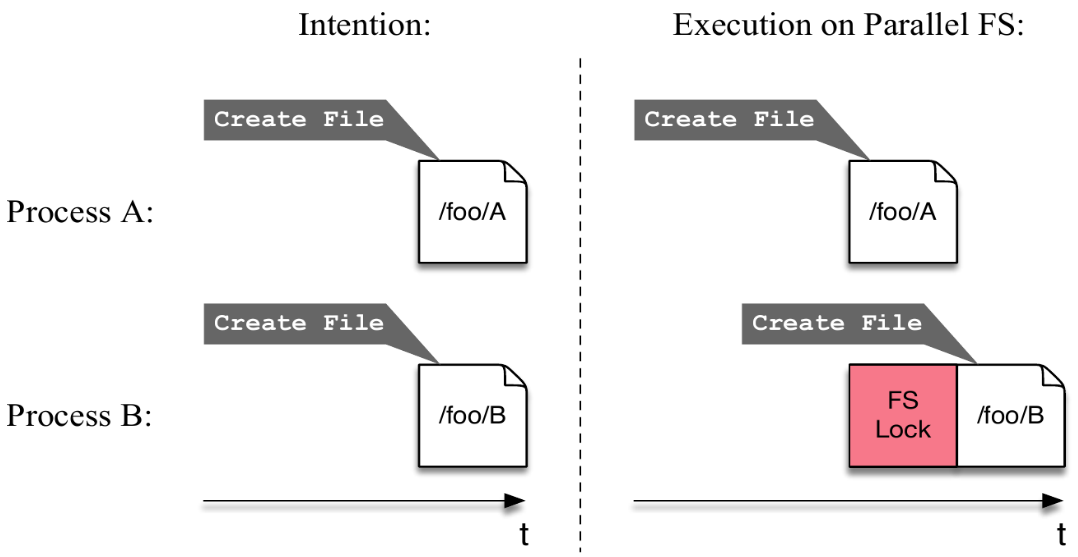
The Idea of rabbitxx

- HPC center may provide multiple file systems
- How does the user know which file system to choose?
- How does the user know which file system features are useful for its job?
- What kind of consistency requirements applications need?
- What kind of pattern are critical regarding to POSIX IO semantics?
- **Needs a view of I/O operations that can occur in parallel!**

Exemplary performance critical patterns

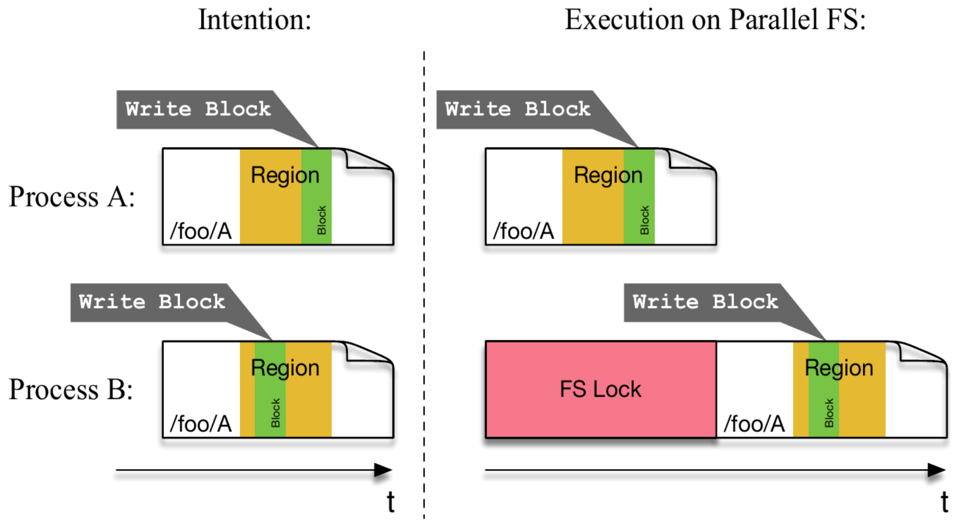
Concurrent creates in the same directory

- may involve locking on the MDS
- problems may just occur at scale



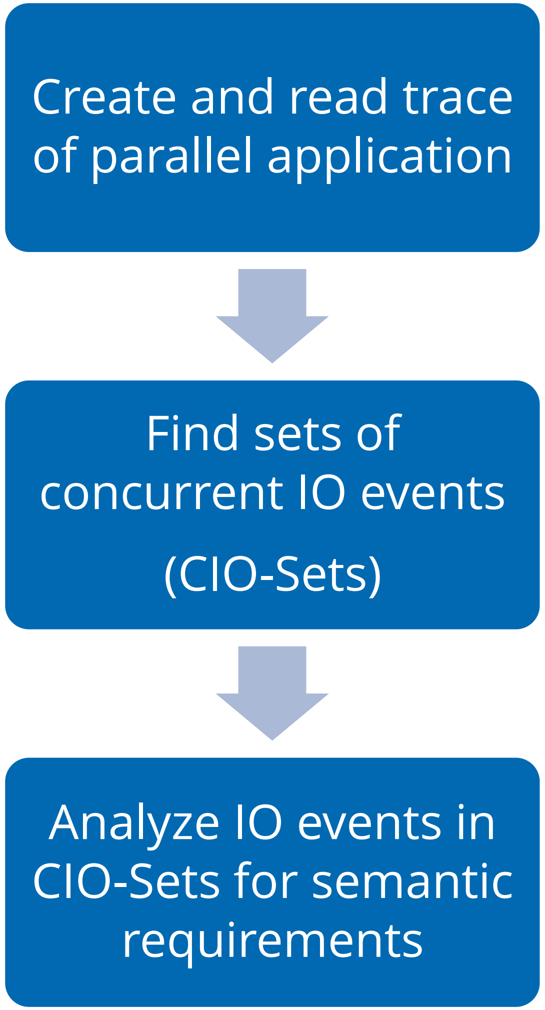
Overlapping writes to the same region of a file

- may result in locking or undefined behaviour



Methodology

- rabbitxx
- Trace-based post-mortem tool
- Group IO operations that can occur in parallel
- Analyze groups against pre-defined patterns
  - Concurrent creates in the same directory
  - Overlapping write access
  - Read-after-write

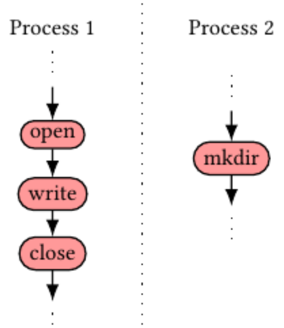


Graph representation

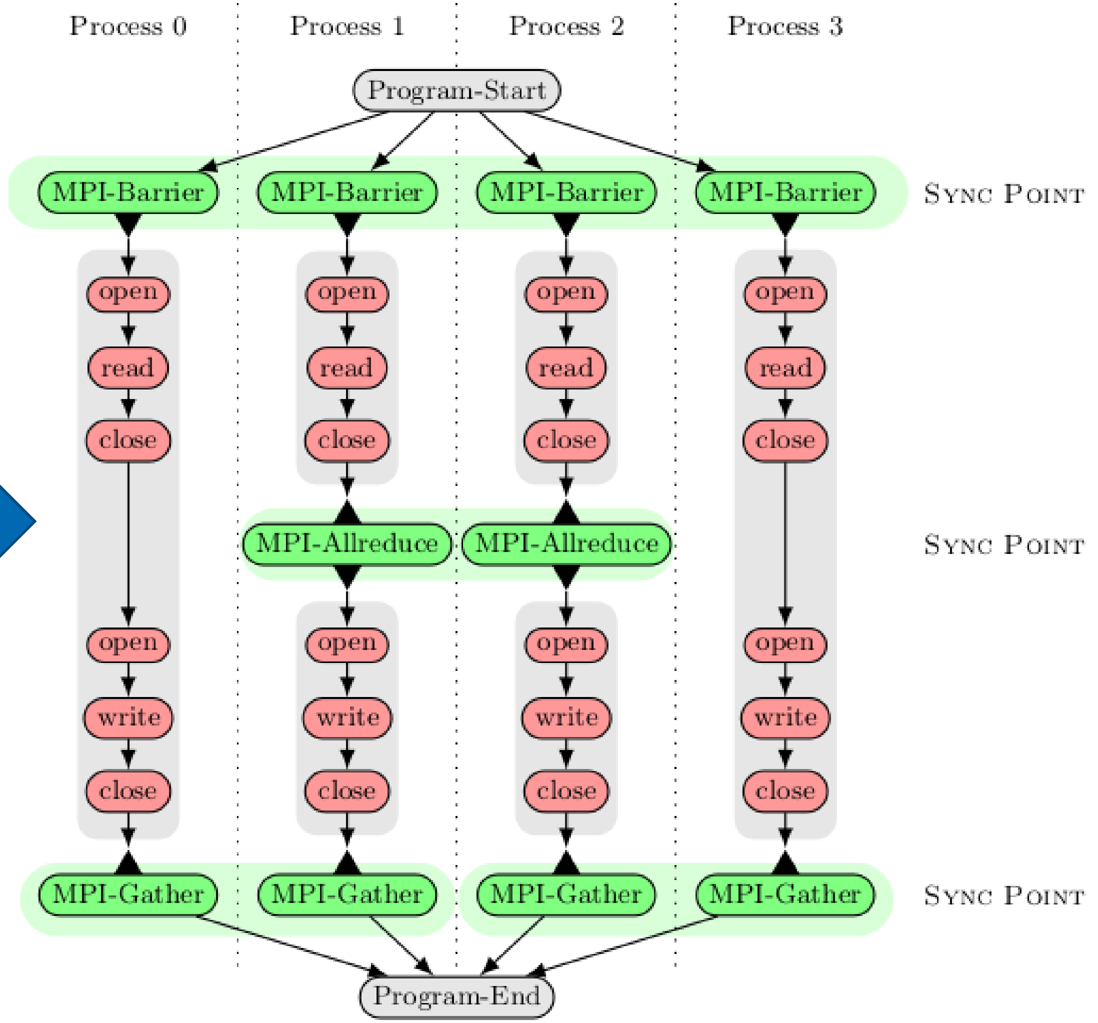
- Read event steam into graph
- Merge consecutive IO events per process
- IO sets bounded by synchronization events
- Synthetic node for program start and end

```
Time  Process   Operation
...
21    1         open(f1)
42    2         mkdir(d2)
55    1         write(f1)
66    1         close(f1)
...
```

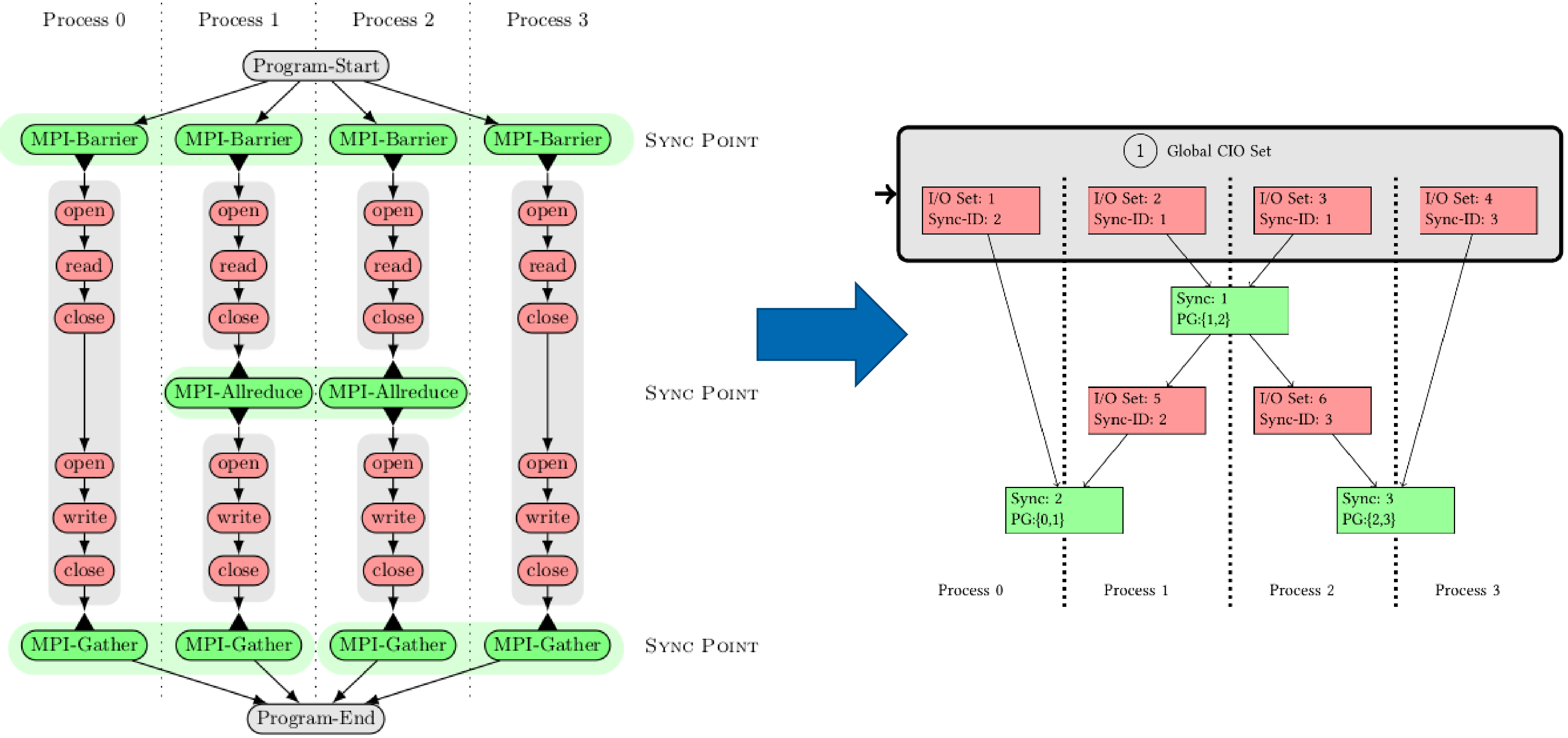
(a) Application trace



(b) Graph representation

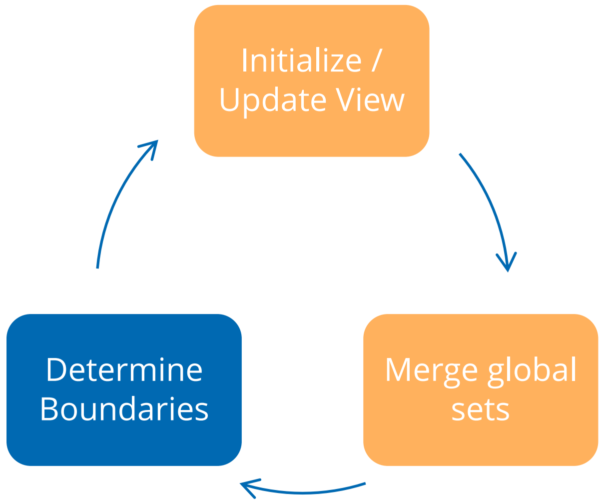


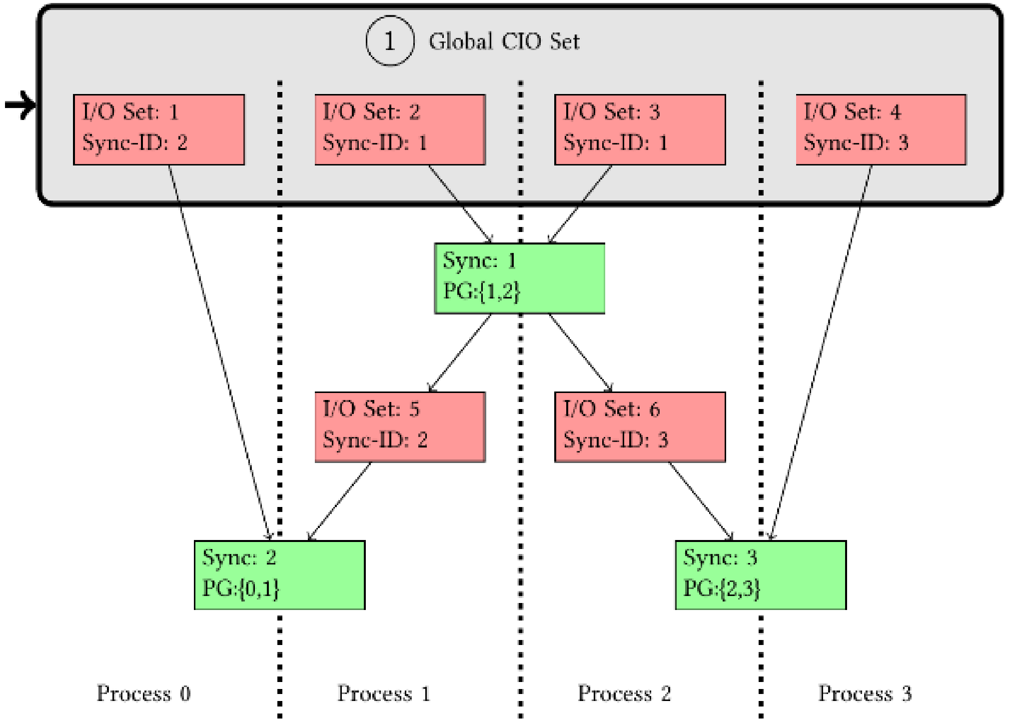
Group IO-Sets per process



CIO-Set Algorithm (1)

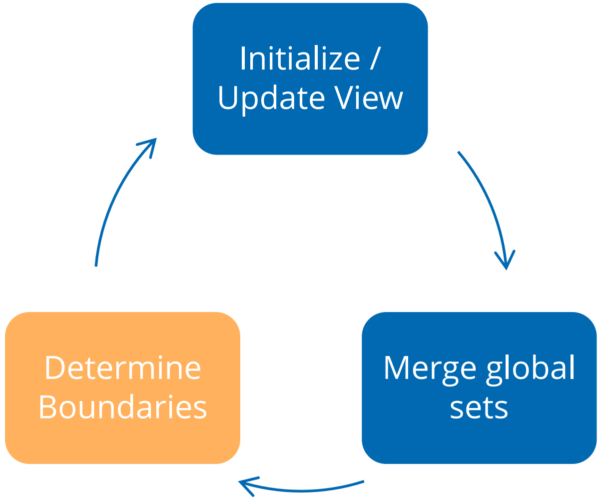
- Initialize view with an IO set from each process
- Merge IO set of each process to global CIO-Set

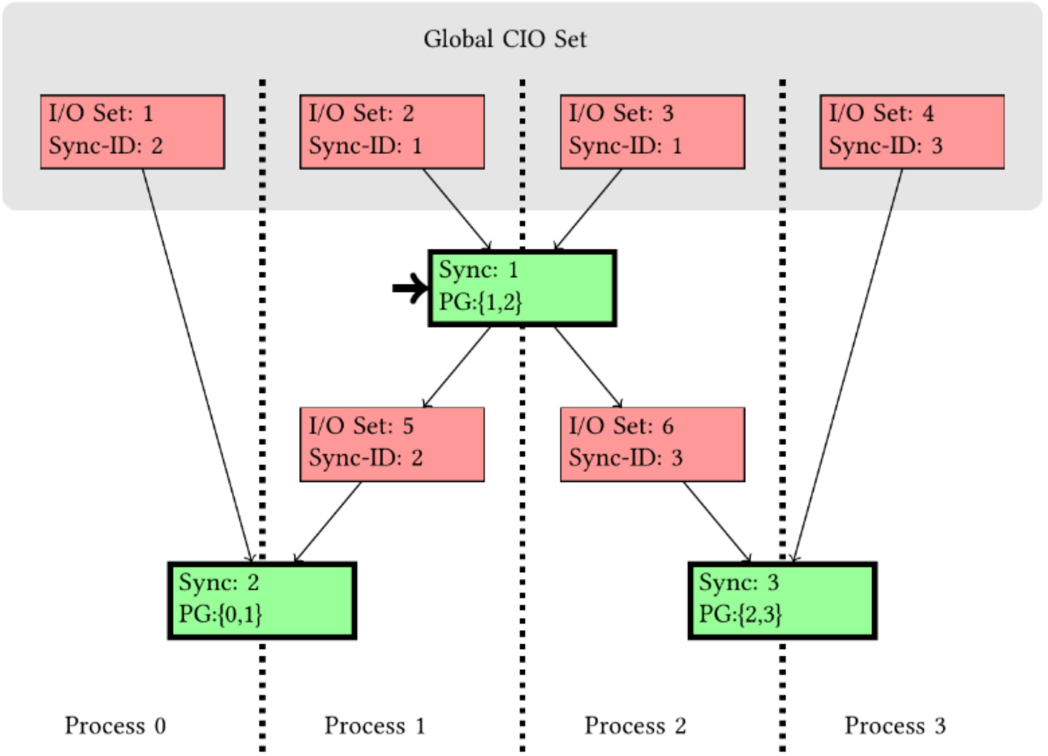




CIO-Set algorithm - (2) determine boundaries

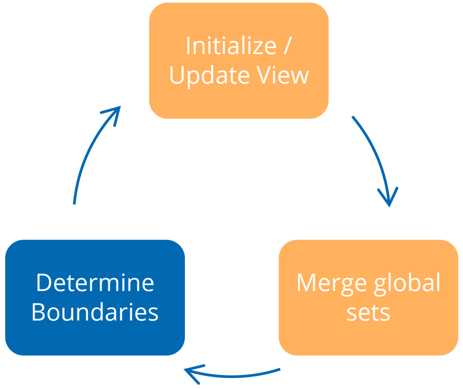
- First synchronization event borders the IO set
- Increment view for processes involved in the synchronization

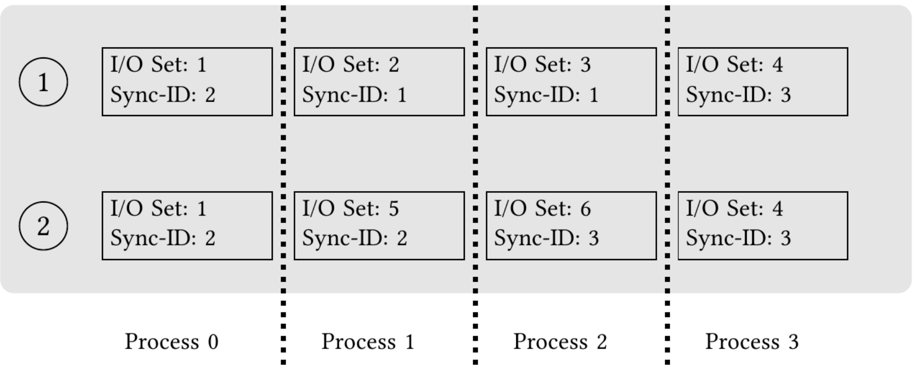


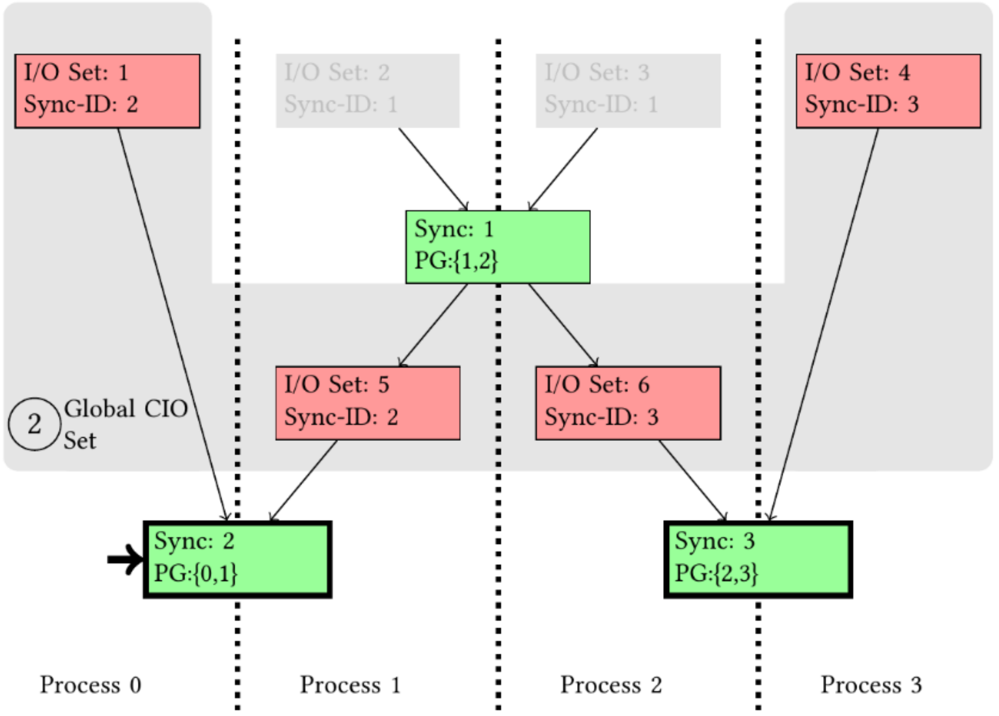


CIO-Set algorithm – (3) update view

- Create new CIO-Set with updated view
- Start from beginning







Evaluation

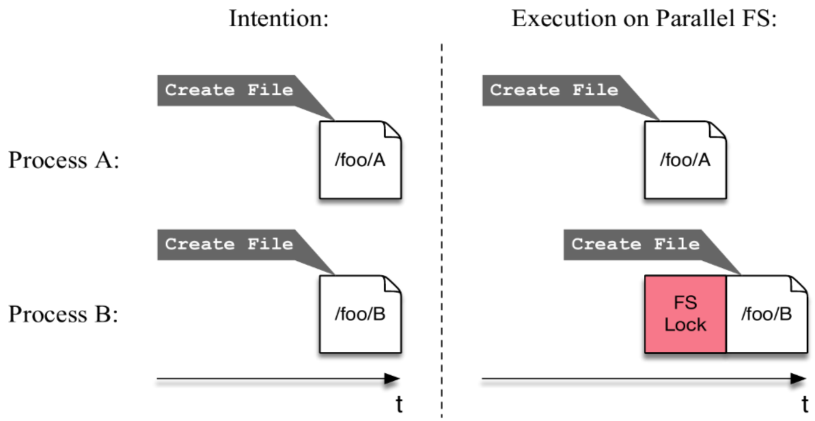
- Analyzed two Benchmarks
  - Madbench2
  - HACCIO
- Both perform non-overlapping and non-conflicting access
- Both create files in concurrently in the same directory on a per process basis

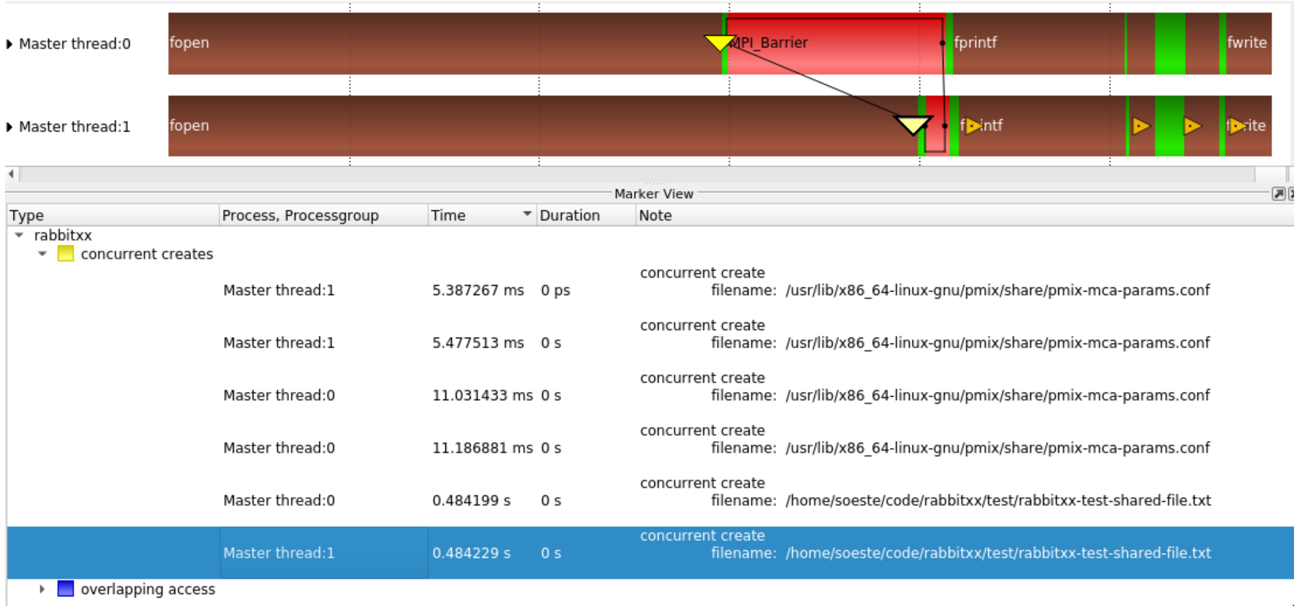
Table 2: MADbench2: Concurrent accesses per file.

CIO-Set	**File**	**#processes**
2	/lustre/scratch2/.../MADbench2/files/data	64
3	/lustre/scratch2/.../MADbench2/files/data	64
4	/lustre/scratch2/.../MADbench2/files/data	64
5	/lustre/scratch2/.../MADbench2/files/data	64

Concurrent creates in the same directory

- Highlight where the patterns are found
- Write OTF2-Markers into the original trace-file
- Vampir can display that





Related Work

Session: Data and I/O HPDC '21, June 22–25, 2021, Virtual Event, Sweden.

File System Semantics Requirements of HPC Applications

Chen Wang
University of Illinois at Urbana-Champaign
Champaign, US
chenw5@illinois.edu

Kathryn Mohror
Lawrence Livermore National Laboratory
Livermore, US
kathryn@llnl.gov

Marc Snir
University of Illinois at Urbana-Champaign
Champaign, US
snir@illinois.edu

- Wang et al. determine the consistency semantics requirements of HPC applications
- develop a method for detecting I/O accesses that can cause conflicts under weaker consistency models.
- just consider data operations (read or write)
- 16 of 17 applications can utilize PFSs with weaker semantics

Conclusion and Outlook

- **Observations**
  - Data operations seem to be coordinated and non-conflicting
  - Metadata operations can be an issue when strict consistency is used
- **Suggestions for the User**
  - User can be guided to incorporate additional levels of subdirectories
  - Using a file system without a hierarchical directory structure
- **Next**
  - More investigations on the semantic of metadata operations

Thank you

Contact:
Sebastian Oeste: sebastian.oeste@tu-dresden.de
rabbitxx: https://github.com/blastmaster/rabbitxx