Detecting Data Races on Storage Systems Using Recorder

Chen Wang and Kathryn Mohror – LLNL
Marc Snir – UIUC

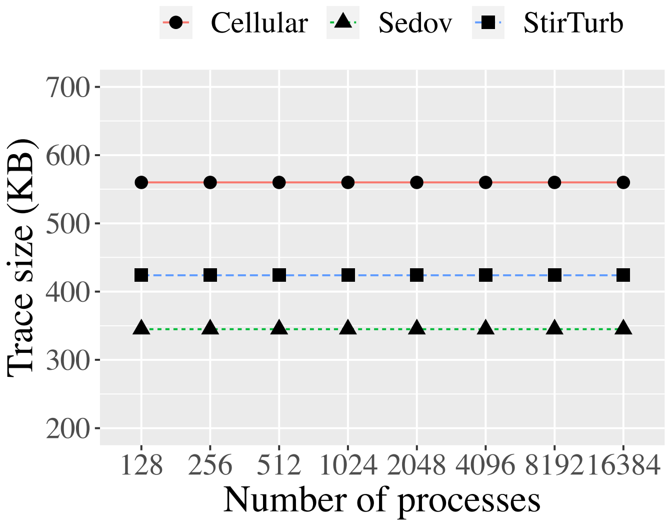
Nov 17, 2022

Lawrence Livermore National Laboratory

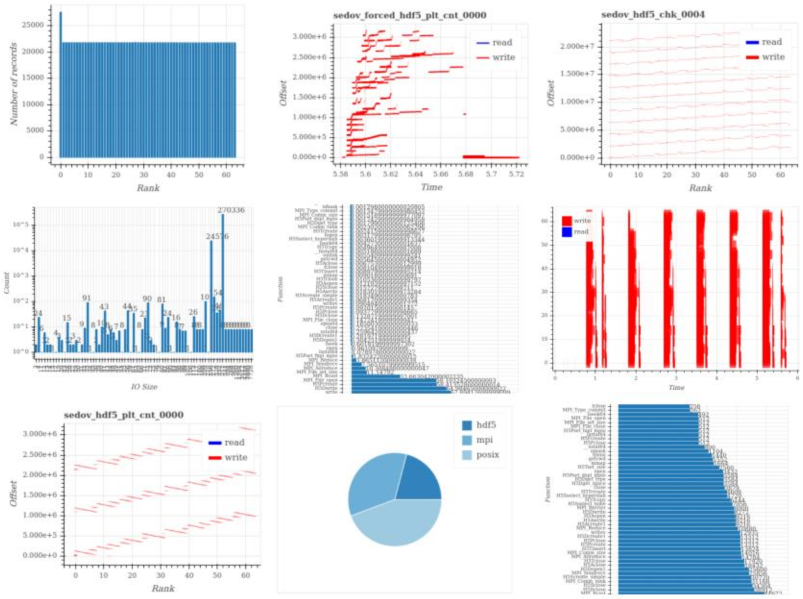
Recorder

- A holistic tracing tool that traces MPI, MPI-IO, POSIX, and HDF5 calls.
  - Stores all function parameters
- https://github.com/uiuc-hpc/Recorder

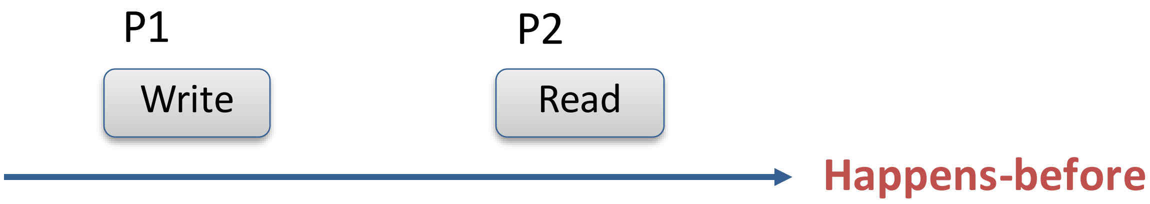
Potentially constant trace size



Post-processing tools and visualizations

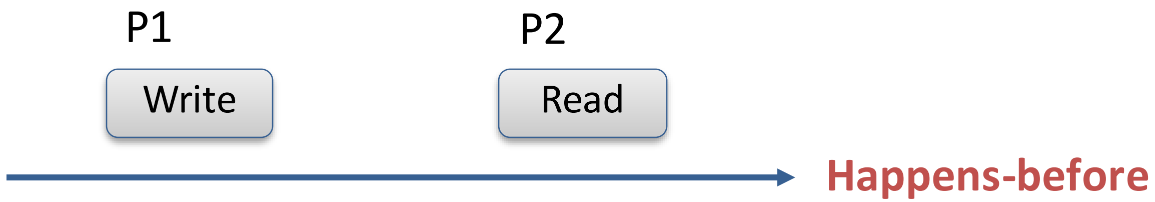


Data Races?



Is P2's read guaranteed to return the data written by P1?
(Do they form a data race?)

Data Races?



P1

P2

Write

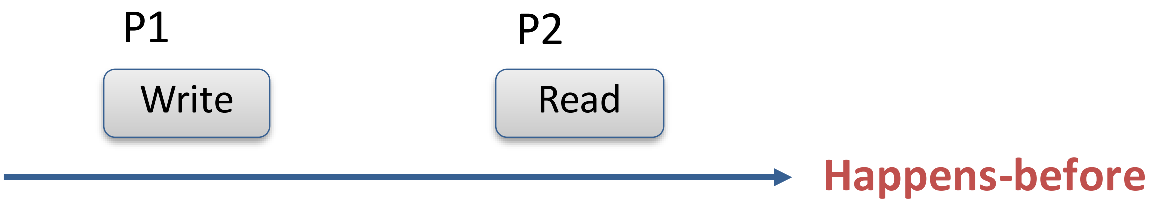
Read

Happens-before

Is P2’s read guaranteed to return the data written by P1?
(Do they form a data race?)

We can’t answer this question because we haven’t defined the *consistency model*.

Sequential Consistency (POSIX)



P1

Write

P2

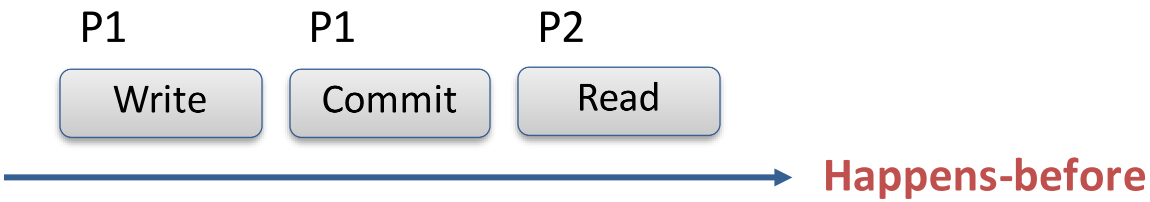
Read

Happens-before

POSIX requires that a write should become immediately visible to all subsequent reads.

Examples of POSIX systems: Lustre, GPFS, BeeGFS, etc.

Commit Consistency



P1 P1 P2

Write Commit Read

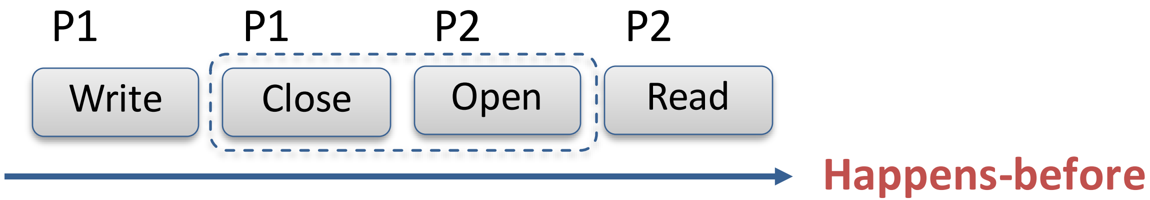
Happens-before

Commit consistency requires an explicit ”commit” operation to make the update visible.

Examples of Commit systems: UnifyFS, BurstFS, BSCFS, etc.

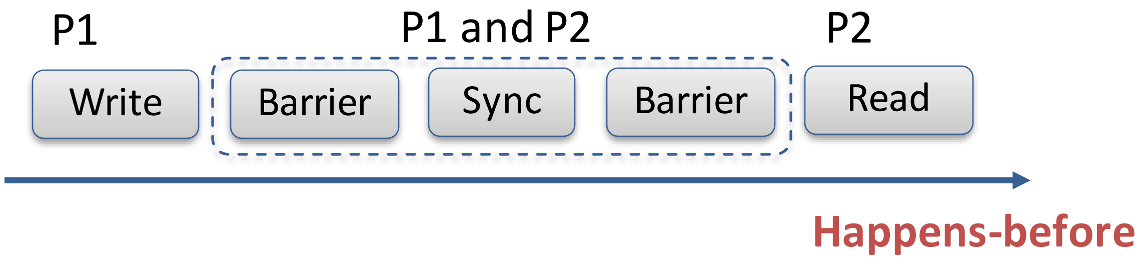
Other Models

Session Consistency:



Examples: NFS, Gfram/BB, etc.

MPI-IO Consistency:



Data Race → Potentially Wrong Result

An application that runs correctly on one model may not run correctly run on a different model.

How to check?

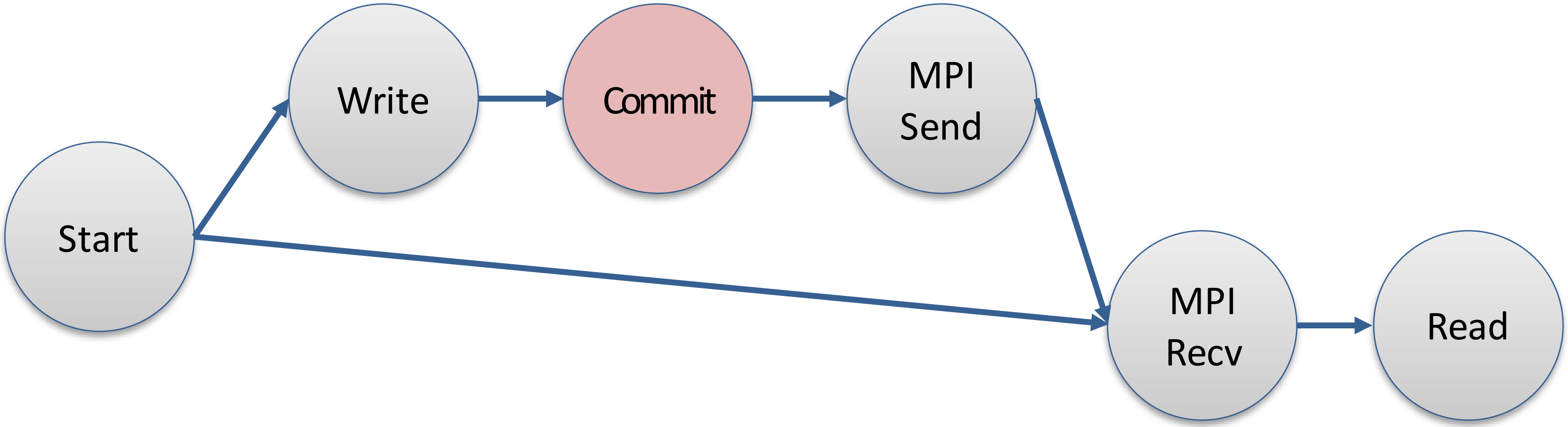
- A trace-driven approach!
- Idea: Check if all conflicting accesses are properly synchronized.

Algorithm for Detecting Data Races

Step 1: Build a happens-before graph from the traces

Step 2: Identify all conflicting accesses

Step 3: Check if all conflicting accesses are properly synchronized



Properly synchronized under POSIX and Commit Consistency
(but not for Session Consistency)

What Do We Need?

1. I/O calls and their parameters
2. Communication calls and their parameters
3. Program order
4. Synchronizations

Recorder captures all the information needed.

Code included in Recorder.

Results and Remarks

We tested 17 HPC applications. 7 show conflicting accesses.

- No data race under Sequential/Commit Consistency
- 1 has data races under Session Consistency.

Most HPC applications should be able to take advantage of storage systems with relaxed consistency models.

[1] Chen Wang, Kathryn Mohror, and Marc Snir. "File System Semantics Requirements of HPC Applications", HPDC, 2021
[2] Sushma Yellapragada, Chen Wang, and Marc Snir. "Verifying IO Synchronization from MPI Traces", PDSW, 2021

Questions?

Contact:
Chen Wang – wang116@llnl.gov

Lawrence Livermore
National Laboratory

Disclaimer
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

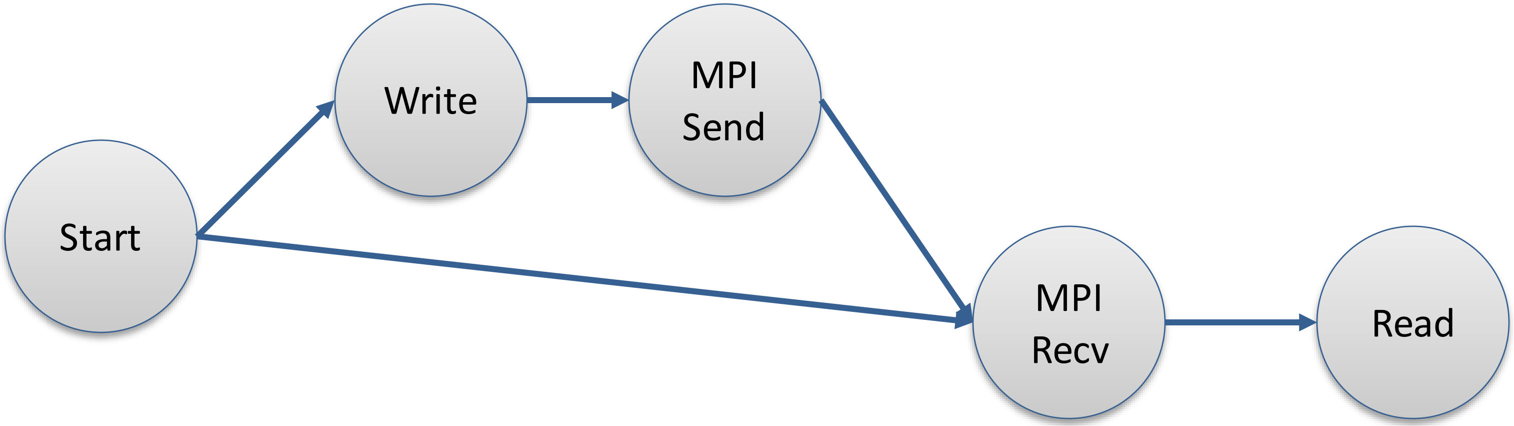
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Backup Slides

Algorithm for Detecting Data Races

Step 1: Build a happens-before graph from Recorder traces.

1. I/O
2. Communication - matching MPI calls.
3. Program order



Algorithm for Detecting Data Races

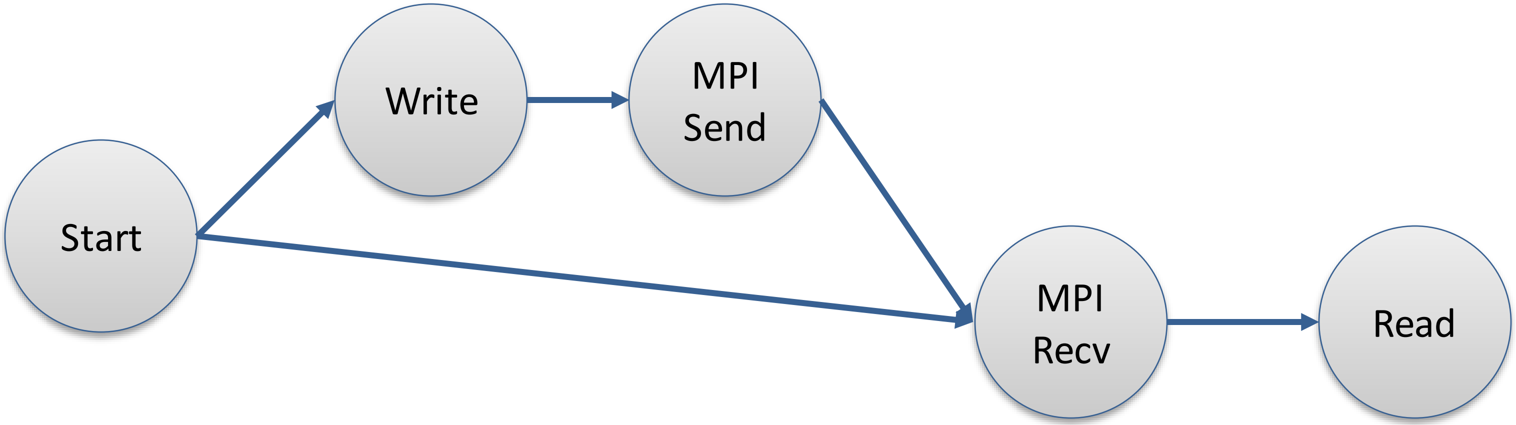
Step 2: Identify all conflicting accesses.

1. Need to examine every I/O operations (data and metadata).
2. Compare their access ranges.
   - pwrite() with explicit offset
   - fwrite() without explicit offset.
   - Nested open/close?

Algorithm for Detecting Data Races

Step 3: Check if all conflicting accesses are properly synchronized.

- A reachability problem (can be done quickly for DAG)



Properly synchronized under POSIX but not Commit Consistency

The 17 Apps

Application	I/O library	WAW		RAW	
		S	D	S	D
FLASH	HDF5	✓	✓		
ENZO	HDF5			✓	
NWChem	POSIX	✓		✓	
pF3D-IO	POSIX			✓	
MACSio	Silo	✓			
GAMESS	POSIX	✓			
LAMMPS	ADIOS	✓			
	NetCDF	✓			
	HDF5				
	MPI-IO				
	POSIX				
MILC-QCD	POSIX				
ParaDiS	HDF5				
	POSIX				
VASP	POSIX				
LBANN	POSIX				
QMCPAC	HDF5				
Nek500	POSIX				
GTC	POSIX				
Chombo	HDF5				
HACC-IO	MPI-IO				
	POSIX				
VPIC-IO	HDF5				