Shared I/O Developments for Run 3 in the ATLAS Experiment

Alaettin Serhan Mete, Peter Van Gemmeren

Argonne National Laboratory (obo the ATLAS Collaboration)

U.S. DEPARTMENT OF ENERGY Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

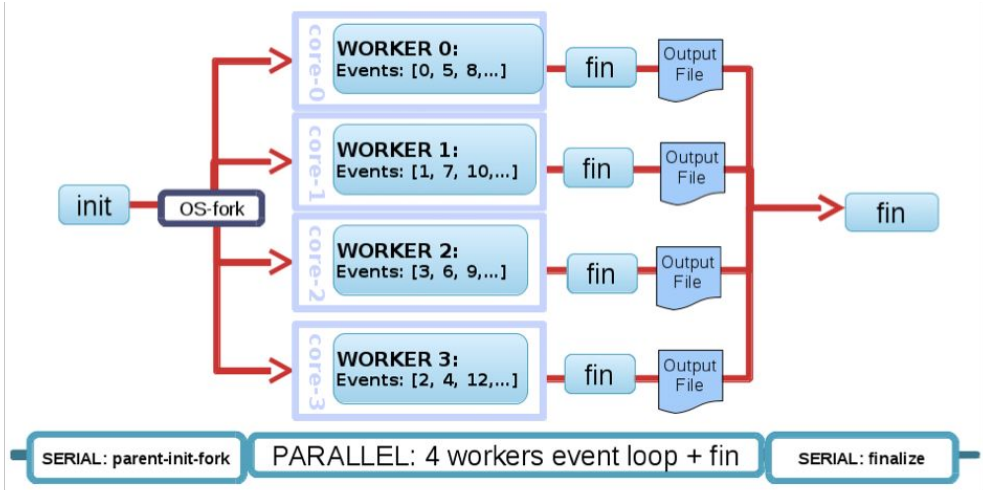
International Conference on High Energy Physics
6-13 July 2022

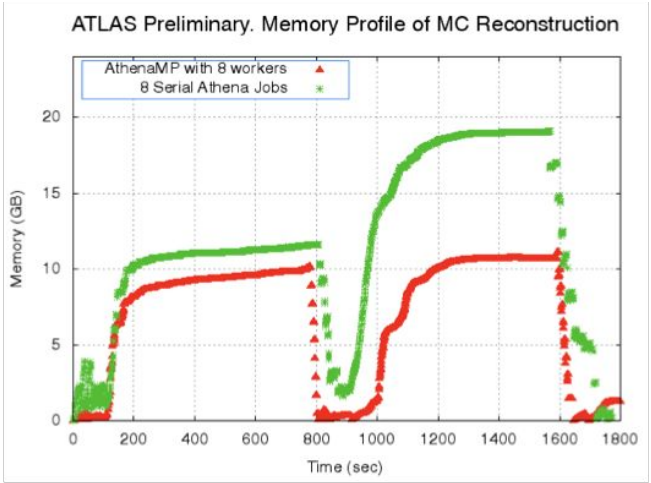
Athena in a Nutshell

- **Athena is the main ATLAS software framework (open-source)**
  - Based on the Gaudi framework, a common LHCb and ATLAS effort (also open-source)
- It consists of about **4 (1.5) million lines of C++ (python)** code
  - CMake is used for *building*, python for *configuration*, and C++ for *algorithms/core framework*
- It has been in use since the early days of the ATLAS experiment
  - Each job consists of 4 main steps: *Configuration*, *Initialization*, *Event-loop*, and *Finalization*
- Today Athena supports **4 different modes of operation**:
  - Serial Athena : All relevant code is executed on a single core
    - Original mode, used throughout Run 1, still used for some workflows (& debugging) today
  - AthenaMP : The event-loop is distributed across many cores via processes
    - Introduced in Run 2 to reduce memory and improve parallelism
  - AthenaMT : The event-loop is distributed across many cores via threads
    - Introduced in Run 3 to further reduce memory and achieve intra-event parallelism
  - AthenaMP/MT : The event-loop is distributed across many cores via processes/threads
    - Currently an experimental mode that targets most optimal throughput/memory scaling

AthenaMP: Multi-process Athena

- Takes advantage of Linux *fork* and *copy-on-write* mechanisms
  - Allows sharing of memory pages between worker processes with little-to-no code change
- Workers process a unique set of events & produce unique outputs
  - These output files need to be merged at a subsequent step introducing sizable overhead

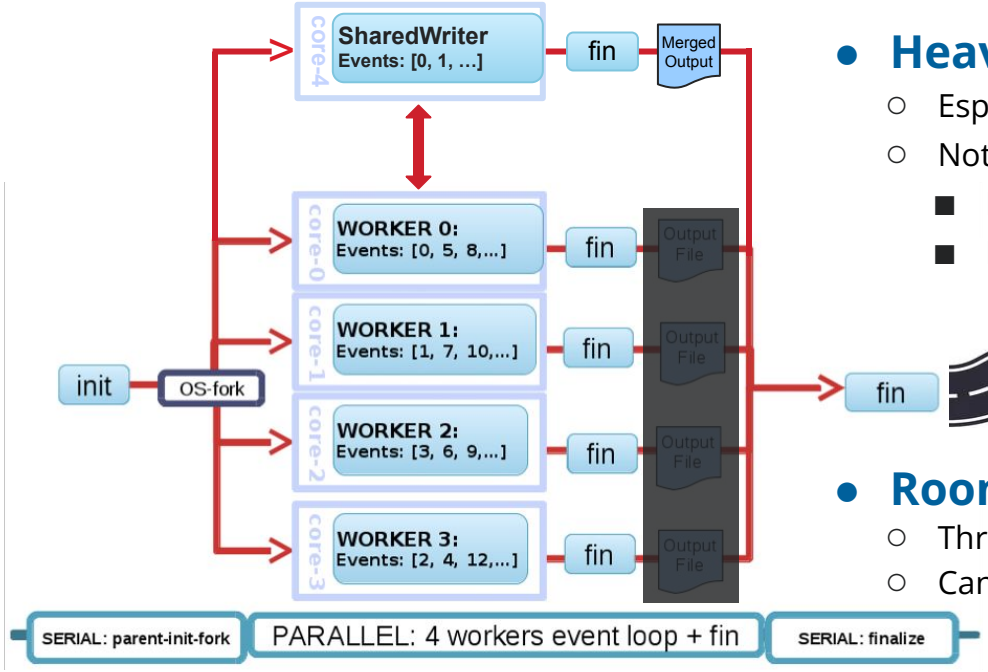




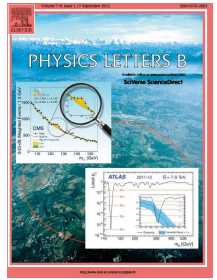
Paolo Calafiura *et al* 2015 *J. Phys.: Conf. Ser.* **664** 072050

Shared I/O: Handling Multi-process I/O

- Data shared through Shared Memory
- Serialization/compression by SharedWriter
- Also summarizes in-file MetaData



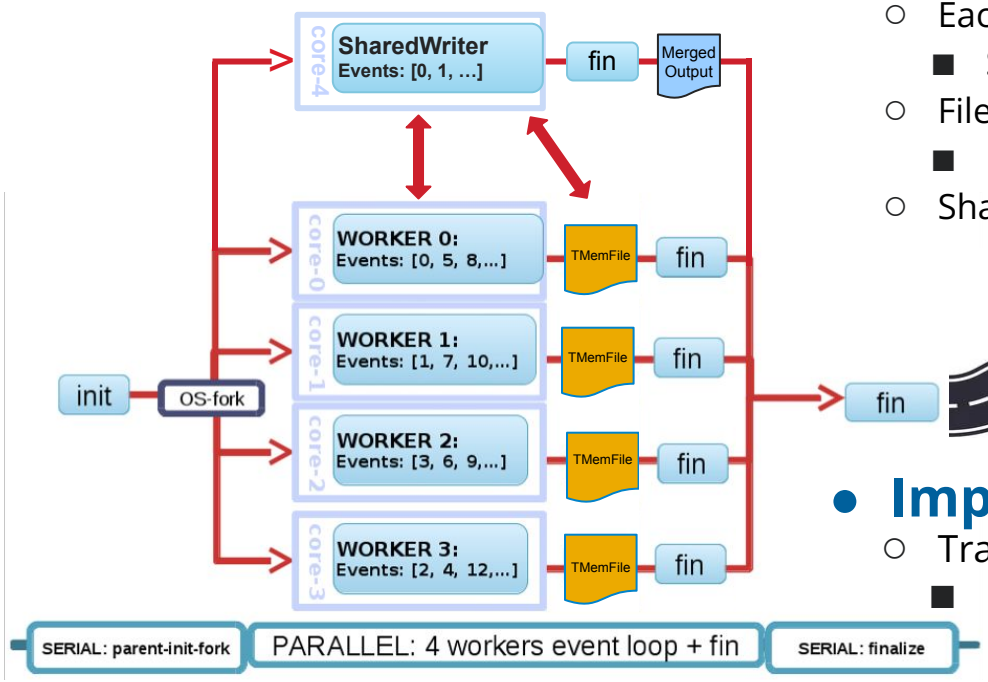
- **Shared I/O was designed for AthenaMP**
  - **ShareWriter** merges output files "*on-the-fly*"
- **Heavily/successfully used in Run 2**
  - Especially in I/O intensive workflows
  - Not only improves throughput but also job success rates
    - **Reduces wall-time by 20-30% in derivation prod.**
    - **No additional (merging) jobs**



- **Room for improvement:**
  - Throughput scaling not optimal beyond 8-10 workers
  - Can also throttle when writing multiple streams, e.g. >2-3

Shared I/O: Handling Multi-process I/O in Run 3

- Data shared through Shared Memory
- Serialization/compression by Worker
- SharedWriter summarizes in-file MetaData



- **Redesigned Shared I/O for Run-3**
  - **SharedWriter** still merges output files "*on-the-fly*"
  - Each worker has in-memory outputs: **TMemFile**
    - Serialization/compression on the worker
  - Files are merged using a custom **ParallelFileMerger**
    - Based on a server/client network protocol
  - SharedWriter de-serializes/summarizes MetaData

- **Improved performance:**
  - Trade-off memory for parallelism
    - Memory is similar to vanilla AthenaMP

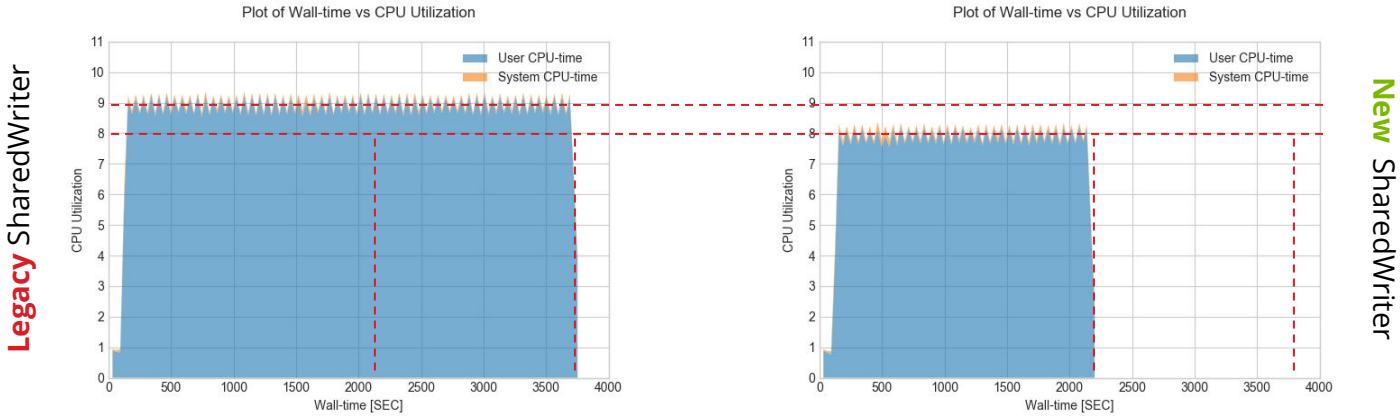
Benchmarks: A Global Overview

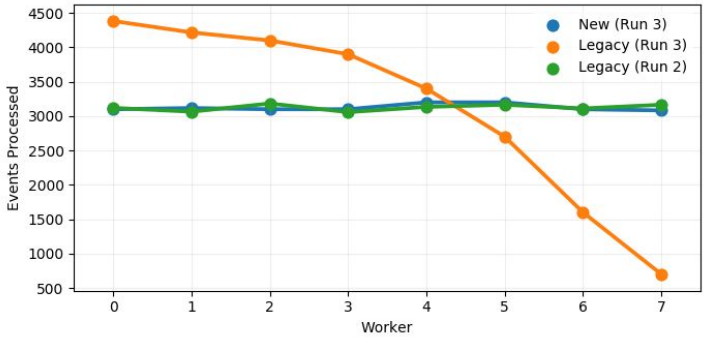
- **Machine and Job:**
  - **AMD EPYC 7302 16-Core Processor @ 3 GHz** w/ **252 GB memory**
  - Producing **DAOD_PHYS + DAOD_PHYSLITE** w/ **25000 reconstructed data18 events**
    - **D**erived**A**nalysis**O**bject**D**ata (**DAOD**) is the data format used by the physics analyses
    - PHYS(LITE) formats include all input events w/ an event-size of O(10 KB/event)

# of Cores	Events/Wall-time [1/s]			Memory/Core [GB]		
	Legacy SharedWriter	New SharedWriter	Difference [%]	Legacy SharedWriter	New SharedWriter	Difference [%]
4	4.9	6.0	+21	1.94	2.10	+9
8	6.7	11.4	+70	1.67	1.88	+13
16	6.7	21.2	+216	1.45	1.81	+25

- **Executive Summary:**
  - New SharedWriter has a **much better throughput scaling** (+70% @ 8 worker processes)
  - The **increased memory usage** still stays **below** the grid resource **limits** (2 GB/core)

Benchmarks: Closer Look at the 8-worker Job





*"Legacy (Run 2)" shows a typical use case in Run 2

- Legacy SharedWriter limitations:
  - In Run 3 jobs are more I/O intensive due to new analysis model
    - Producing multiple inclusive formats (PHYS/LITE) in a single job
  - A single instance can't keep up with large # of workers/formats
  - Effective # of parallel workers are reduced, hurting throughput
- New SharedWriter improvements:
  - Workers are practically independent: Ideal scaling
  - Work is equally distributed, optimizing throughput
  - No resource (CPU) usage overhead, helps the GRID

Ongoing & Future Developments

- Delayed OS-fork of workers to maximize the shared memory
  - Let the main process execute N events (typically 1) before launching the worker processes
  - This is proven to significantly improve the memory profile of the relevant jobs
  - The new SharedWriter support for this mode is currently being validated
- Taking advantage of SharedWriter to help I/O intensive MT jobs
  - The MP/MT hybrid mode can allow us to enjoy the best of both worlds
    - Running N threads in M processes to achieve NxM parallelism
    - This approach has the potential to achieve:
      - Better memory scaling than MP-alone
      - Better throughput scaling than MT-alone
  - In I/O intensive workflows, SharedWriter can improve with the throughput scaling w.r.t cores
- Overall there are a number of such improvement in the pipeline
  - Today is more interesting than yesterday but tomorrow will be even more interesting!

Conclusions

- The **Shared I/O** infrastructure has been **successfully used in Run 2!**
- **Various improvements are made ahead of Run 3:**
  - The new analysis model with common inclusive formats requires improved parallelization
  - Improving the throughput scalability also allows us to take better advantage of HPCs
  - **At 8 processes, events/second is improved by 70%, at 16 processes by >200%!**
  - The legacy SharedWiter is still a valuable asset for less I/O intensive (more memory limited) jobs
- A number of new improvements are already in the pipeline:
  - Supporting late OS-fork etc.
- **Shared I/O** can also help w/ certain MT applications:
  - Especially those workflows that are I/O intensive/limited
  - Primarily taking advantage of Athena's rather unique MP/MT hybrid mode of operation
- ATLAS will be using the **Shared I/O** infrastructure for years to come!
  - **We look forward to all the challenges and fun ahead…**

Thank you for your attention!