AthenaMP

Sharing memory between processes in ATLAS software using Linux COW

Vakho Tsulaia

LBNL

Annual Concurrency Forum Meeting
FNAL, Feb-5, 2013

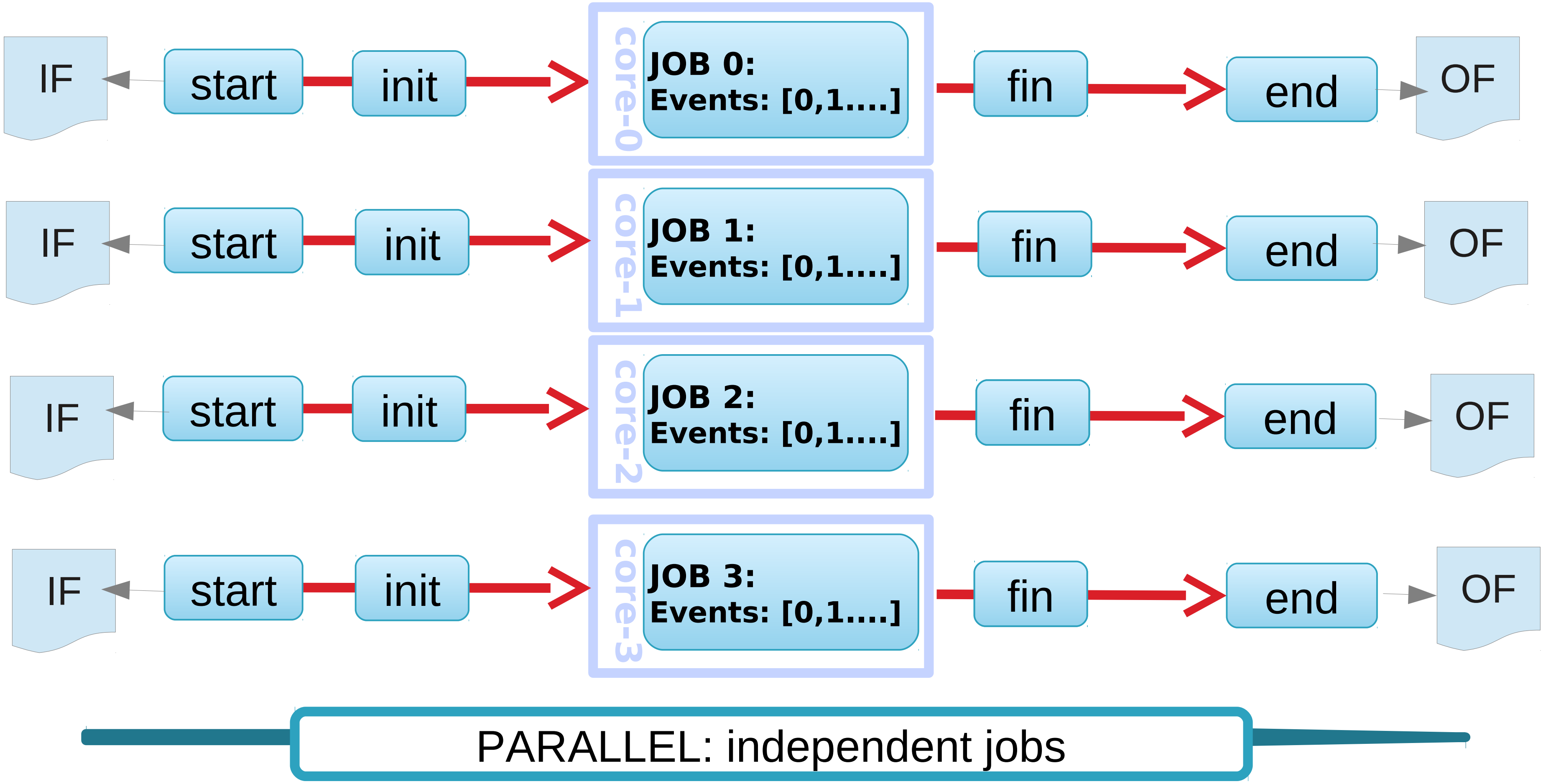
Contents

- Goals
- AthenaMP
  - The concept and the main components
- News since previous Concurrency Forum Meeting
  - Transition to the new implementation: *AthenaMP-2*
- Performance figures
- Future developments

Goals

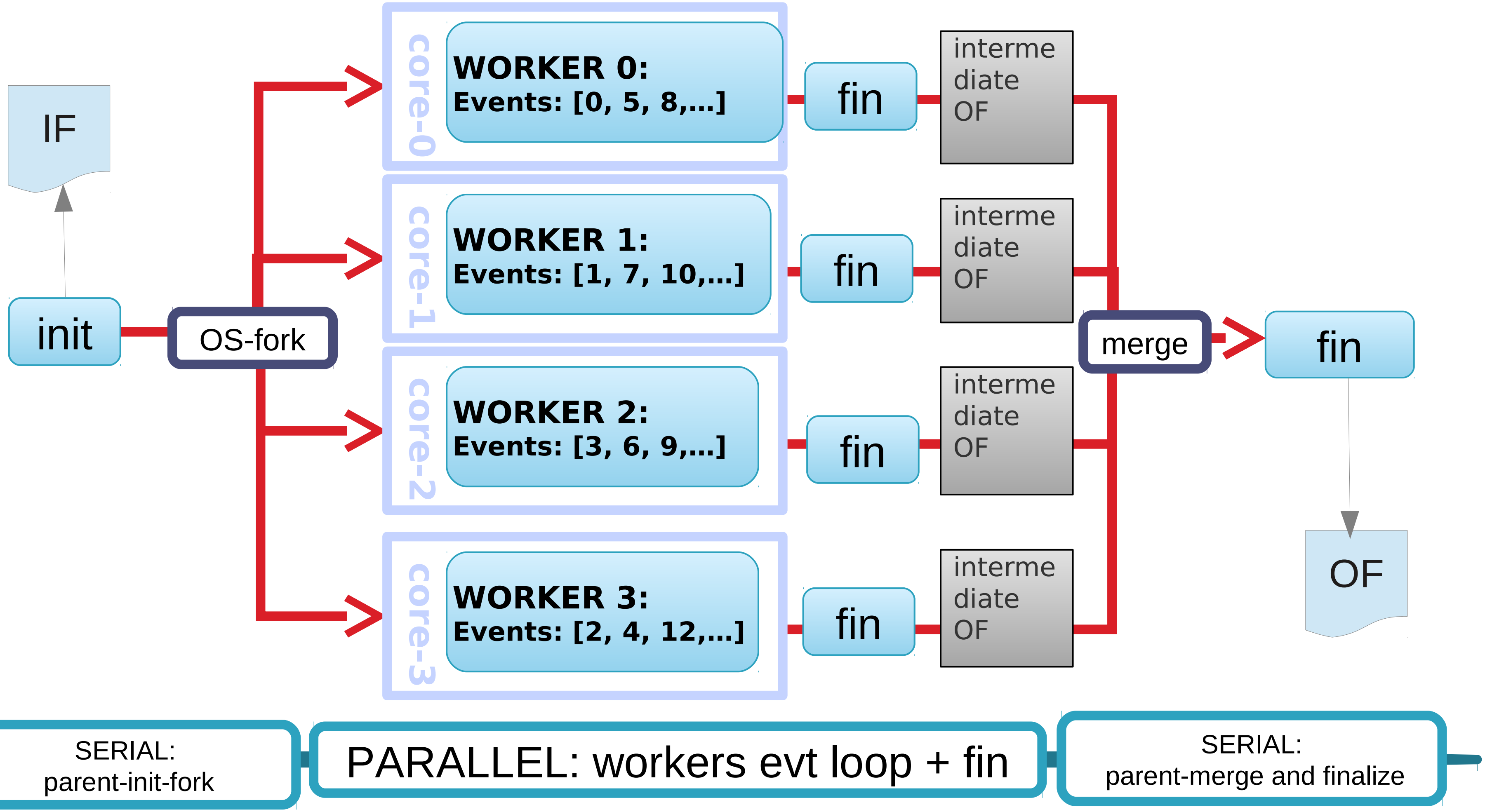
- Effective usage of modern CPU cores
  - **Reconstruction**: Come up with a parallel solution, which would improve event processing throughput of the production nodes wrt current mode of operation (*running many serial reconstructions simultaneously*)
    - **ATLAS Reconstruction is memory-hungry. The parallel solution must allow memory sharing between event processors**
  - **Analysis**: Speedup interactive analysis jobs by processing different input files in parallel instead of going over them serially, one at a time
- We want to achieve this goal with minimal changes to the existing code
  - No changes at all in the user code

Process-based parallelism



Our present mode of operation

Athena MP-1



SERIAL:
parent-init-fork

PARALLEL: workers evt loop + fin

SERIAL:
parent-merge and finalize

Details presented on the first Concurrency Forum Meeting in November 2011

Need of a new implementation

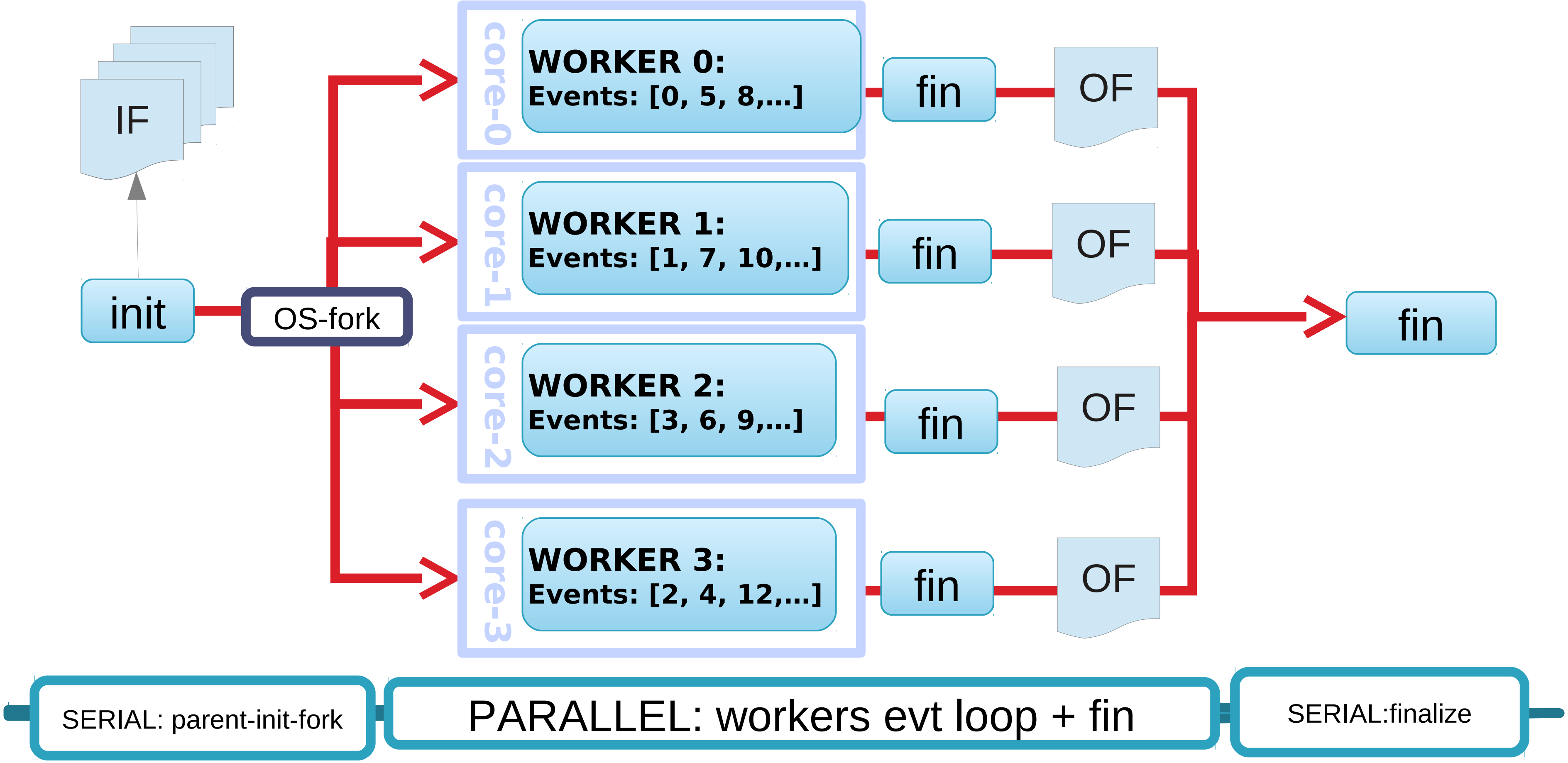
- Original implementation of the AthenaMP lacks design in general, which makes it hard to add new features
- **Output file merging**, which was an **inseparable part of every AthenaMP job**, makes it rather inefficient
  - Short jobs: substantial fraction of the overall wall time spent in merging
  - Long jobs: by merging N full size outputs we make one huge resulting file – difficult for the Production System to digest

AthenaMP-2

- **Features**
  - New infrastructure written completely in **C++**
  - Inter-process communications and process management is handled by a custom library developed in ATLAS
    - Uses **boost interprocessing**: shared queues, shared memory segments
  - Uses components from **GaudiMP:** *IoComponentMgr*
  - **Follows Gaudi component model**: various event scheduling strategies in AthenaMP workers are implemented by specialized components (AlgTools)
    - Should make it easier to plug in new functionalities
  - Output file merging no longer considered the responsibility of the core AthenaMP
    - Now it's up to the clients of AthenaMP to decide how to deal with the outputs made by AthenaMP workers processors

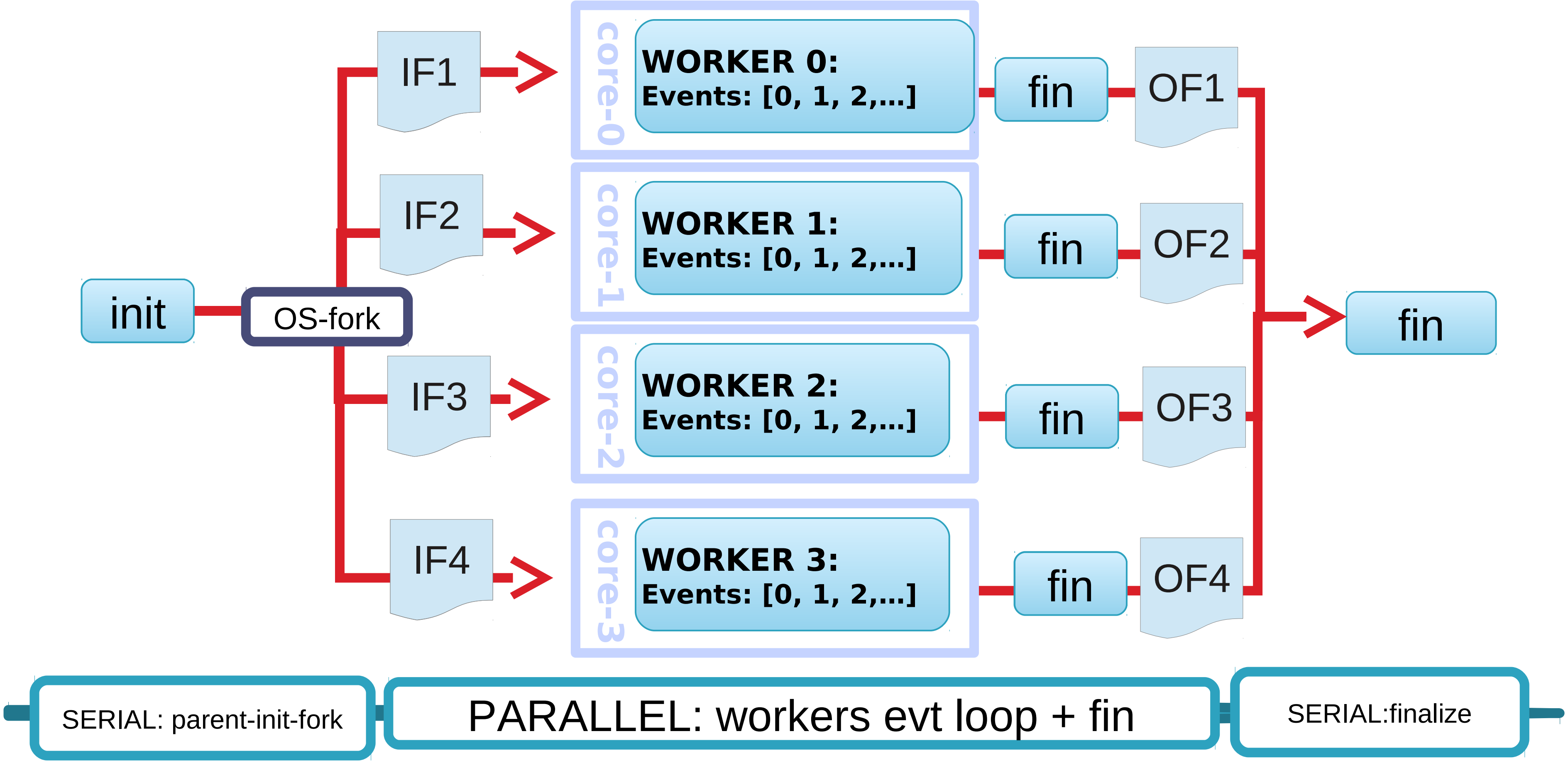
AthenaMP-2

Event scheduling strategies: **Shared Event Queue**

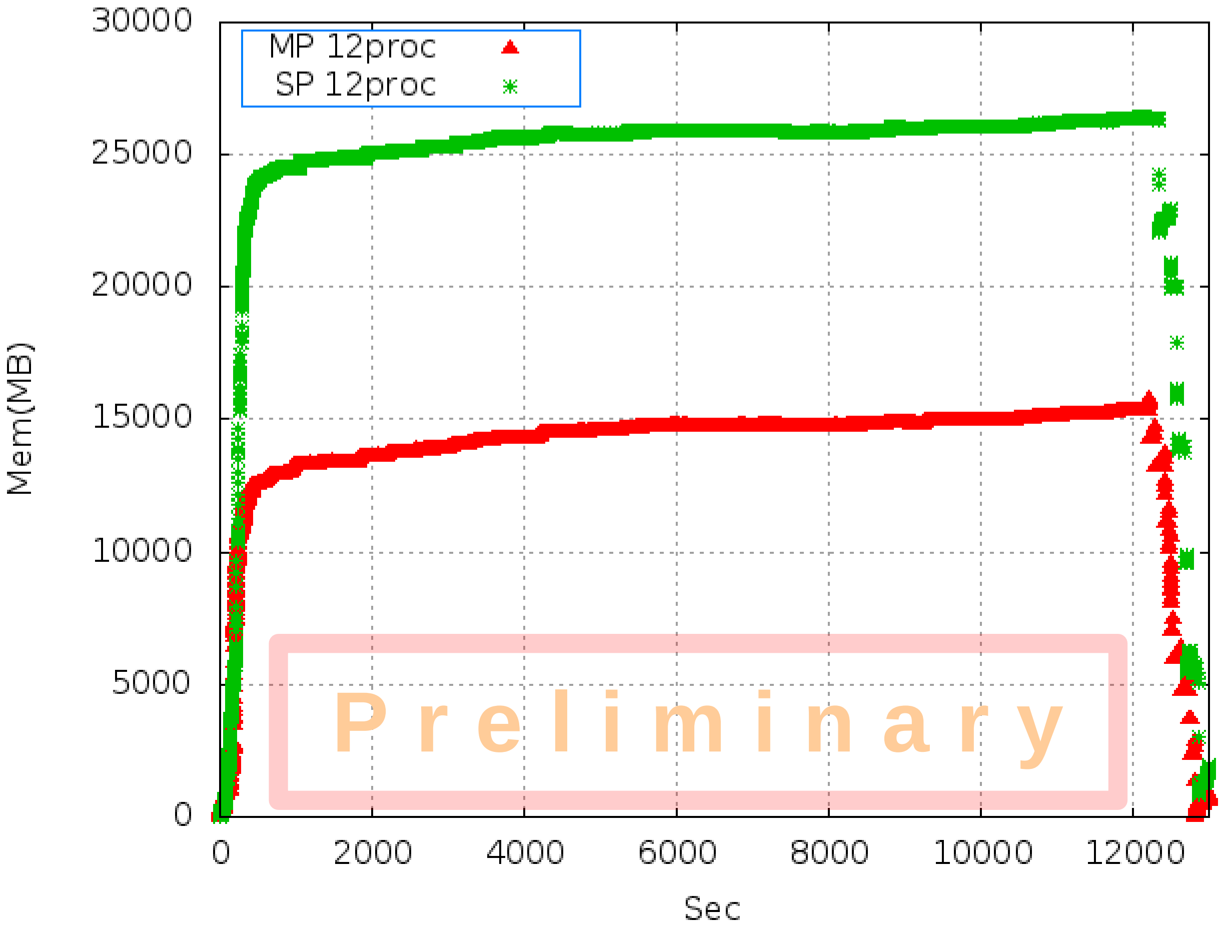


AthenaMP-2

Event scheduling strategies: **Input File Per Worker**

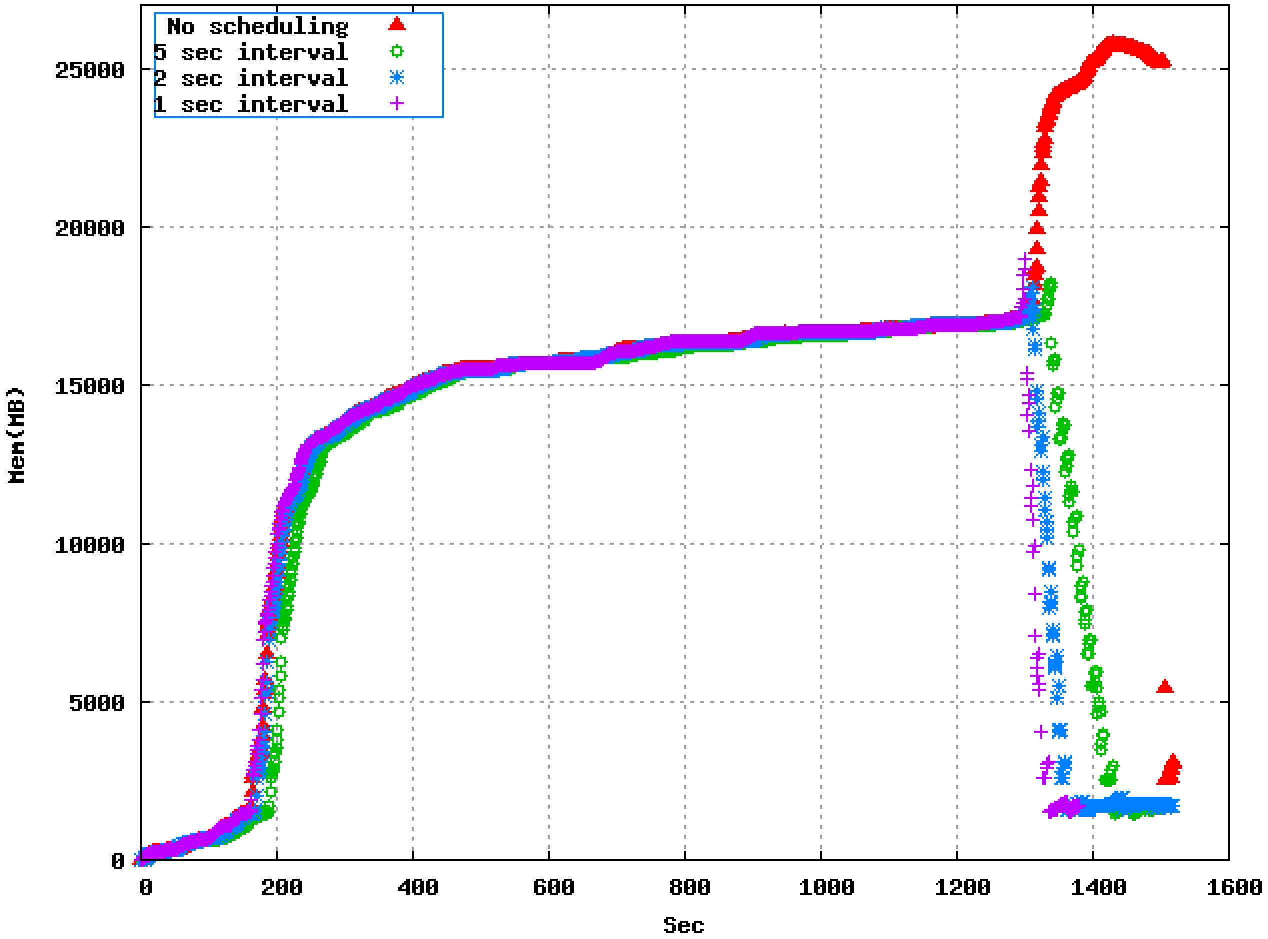


Sharing memory between processes



- Athena reconstruction of real data (RAWtoESD), 64bit, 500evt/job
- Profiling done with '`free -m -s 1`'
- ~45% memory shared between worker processes in AthenaMP

Memory spikes



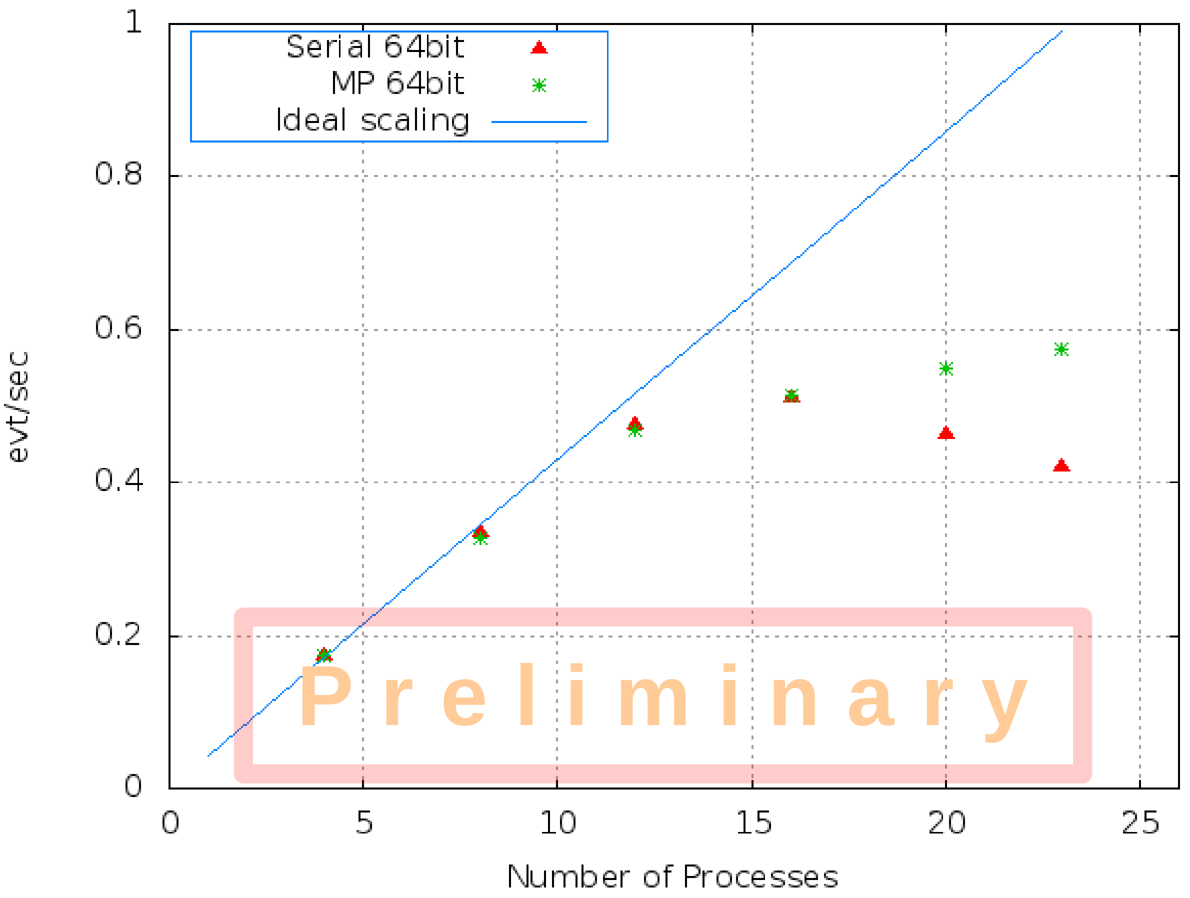
- AthenaMP-1 reconstruction of real data (RAWtoESD), 64bit, 50evt/worker
- Profiling done with '`free -m -s 1`'
- Spikes can be cured by serializing workers' finalization without sacrificing the overall job performance

Event throughput

- Two series of recent tests with 64 bit AthenaMP-2 on the same hardware
  - 12 CPU Core Westmere, Hyper-threading, 48GB memory
  - In order to simulate 3GB/core a special “memory eater” utility was running on the machine bringing available memory limit down to 36GB
- Test #1
  - Real data reconstruction (RAWtoESD), 500evts/job
  - **“Llightweight”** (data quality monitoring algorithms disabled). **~2.2GB/job** of physical memory
- Test #2
  - MC reconstruction (RDOtoESD), 250evts/job
  - **“Heavyweight”** configuration. **~3.3GB/job** of physical memory

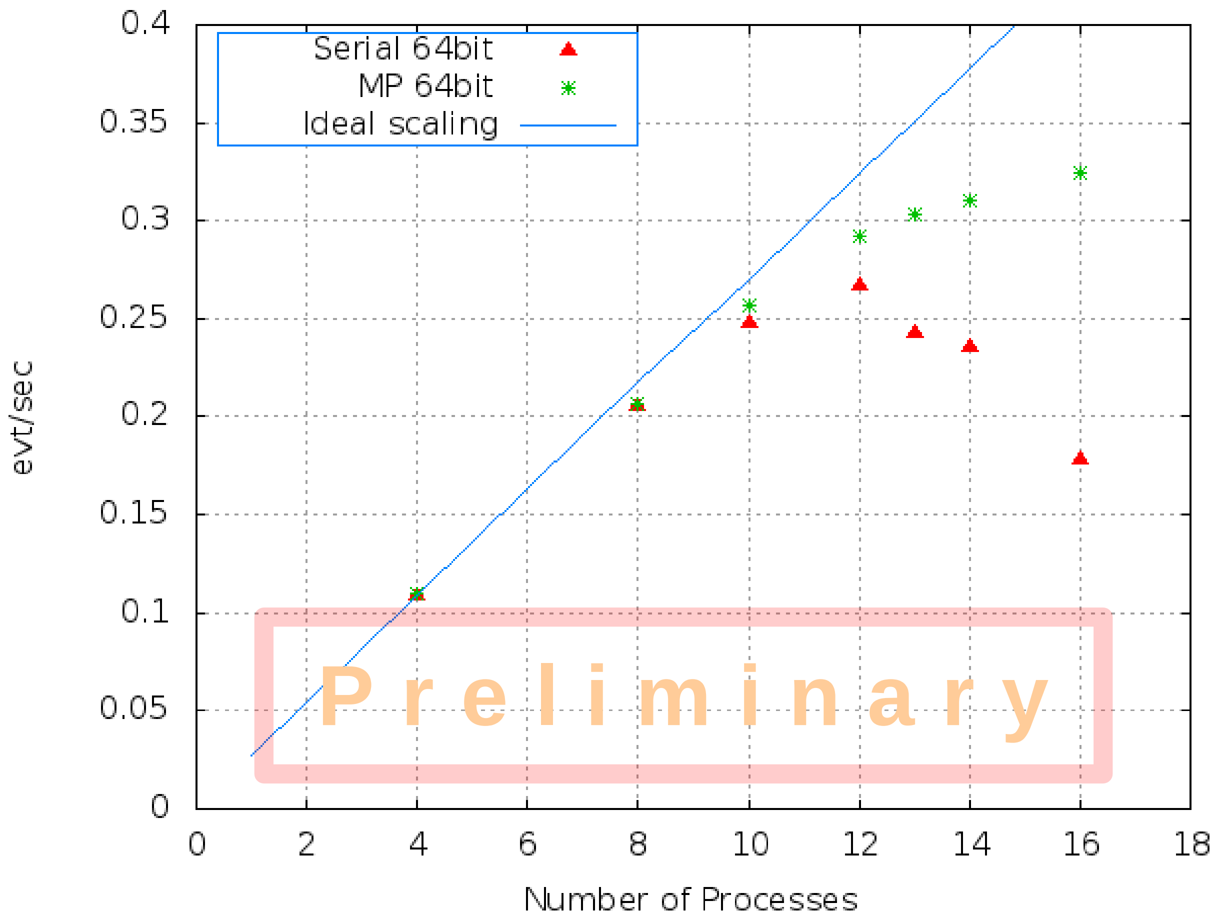
Results are preliminary!

Event throughput. Test #1



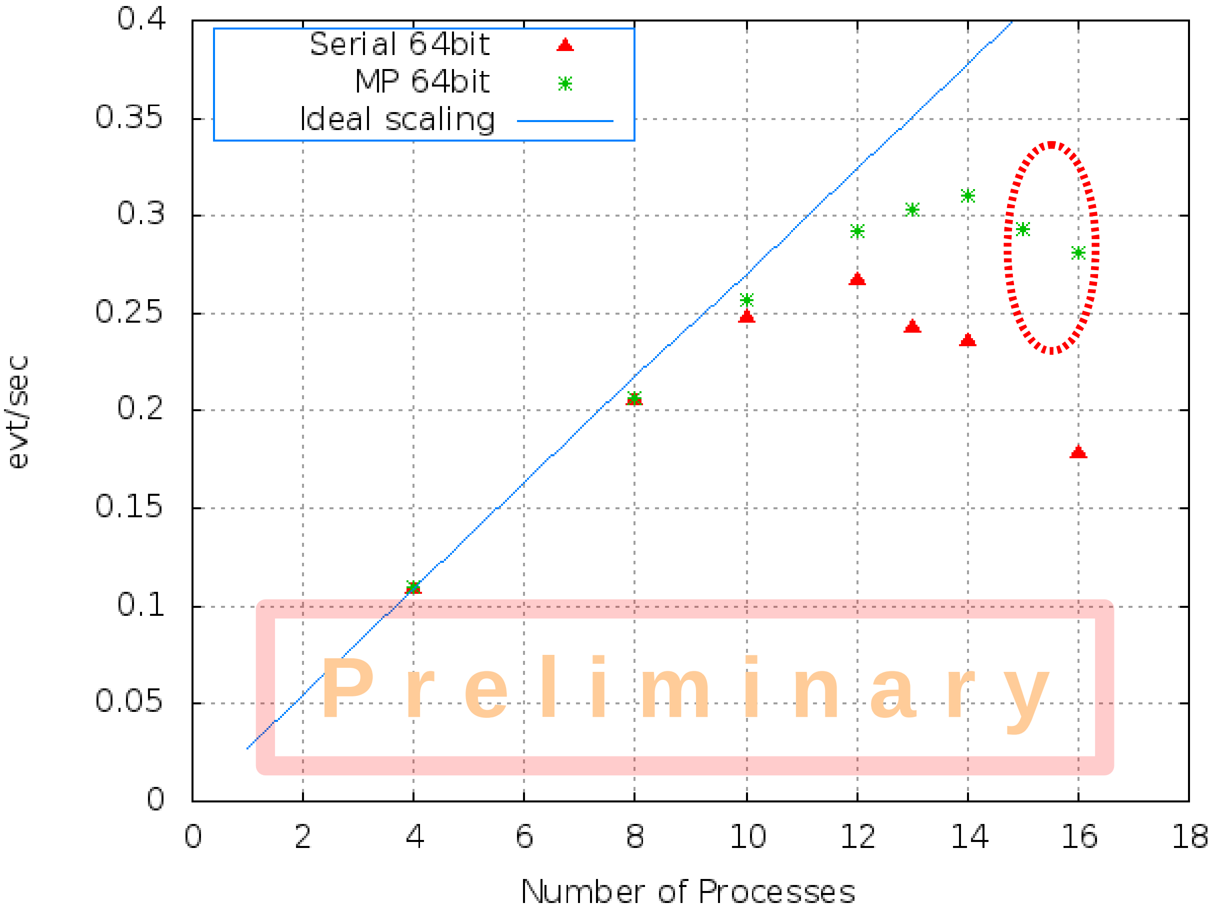
- ~10% gain in event throughput by using AthenaMP

Event throughput. Test #2



- ~20% gain in event throughput by using AthenaMP
- Workers' finalization was serialized in AthenaMP

Event throughput. Test #2



- Workers' finalization was **not** serialized in AthenaMP
- Memory spikes can have a visible effect on overall performance

Summary

- By leveraging Linux fork and COW we achieve a **significant optimization of the overall memory footprint** of multiple Athena reconstruction jobs running on the same machine.

- This optimization comes with **no CPU overhead**.

- It allows us to increase the number of parallel reconstruction jobs and by this way increase the overall event throughput.

- **The exact performance gains depend on concrete job configuration and hardware resources**
  - **The example** included in this talks shows that the event throughput can be increased by **at least 20%** for the **heavyweight reconstruction job on the 3GB/core** machine

Future developments

- Various strategies for scheduling events to worker processes
  - Single event (shared queue). Already exists
  - Event chunks/clusters.
  - Entire file. Prototype exists
- Output file sequencing and its usage in AthenaMP
  - Cut output file when number of events reaches some predefined maximum
  - Or group events by time-dependent conditions (luminosity blocks)
- **Specialized I/O worker processes**
  - Shared reader for RAW data files. Already exists
  - DataHeader/Token scatter for shared POOL reader.
  - Shared writer.
- The **last item** is very important for further developments towards the **event-level I/O:** *replacing files with events as work distribution unit*, which is how ATLAS is considering to follow the **Opportunistic Computing** paradigm