Running ATLAS workloads within massively parallel distributed applications using Athena Multi-Process framework (AthenaMP)

Paolo Calafiura, Charles Leggett, Rolf Seuster

Vakho Tsulaia, Peter Van Gemmeren

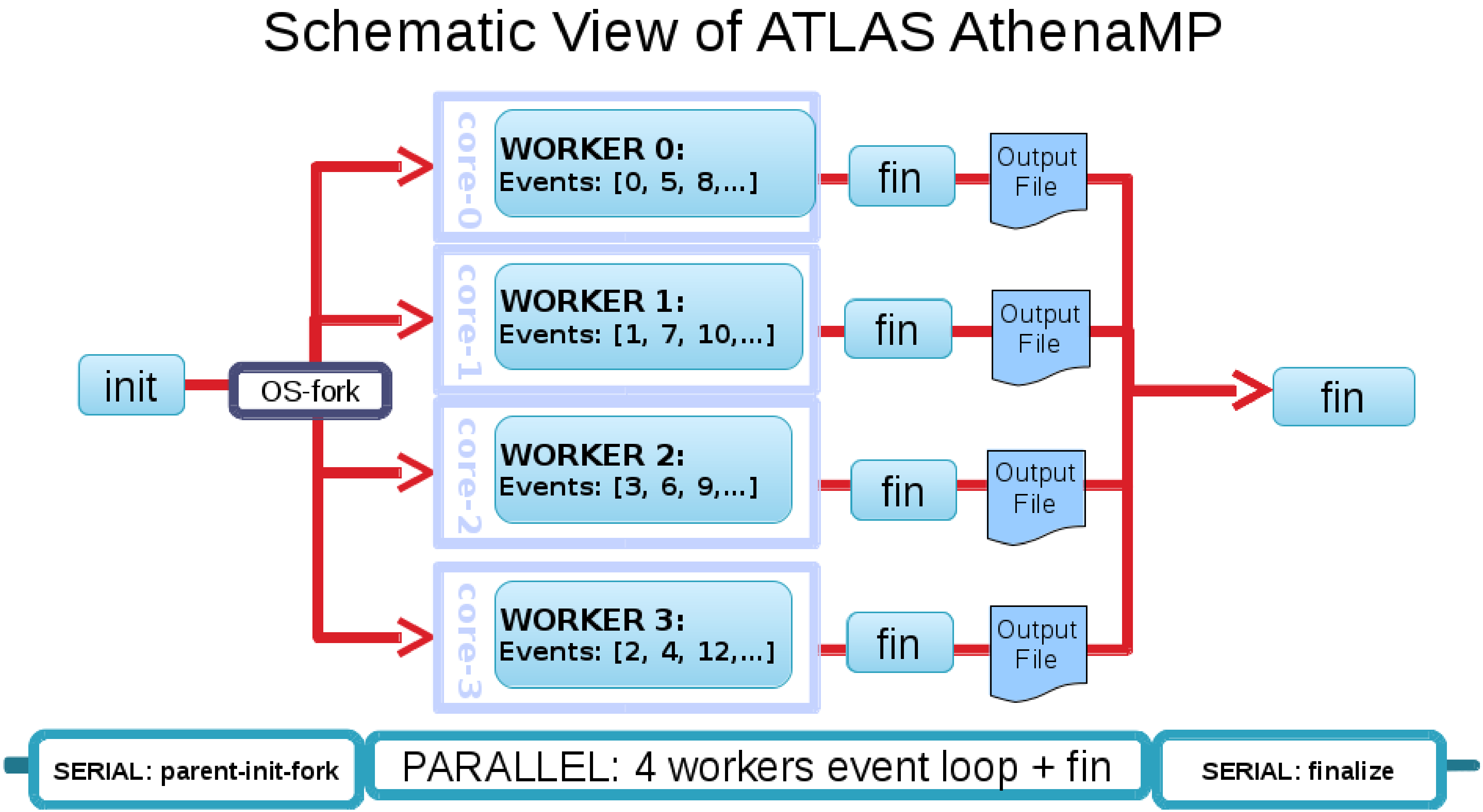
For the ATLAS Collaboration

CHEP 2015, Okinawa, Japan
April 14, 2015

History of AthenaMP

- **Motivation**
  - ATLAS reconstruction is memory-hungry
  - We needed to have a mechanism for optimizing memory footprint without touching the algorithmic code-base
- AthenaMP leverages **Linux Copy-On-Write** for sharing memory pages between processes, which were forked from the same master process
  - Thus the memory sharing comes “for free”
- **Originally implemented as Python layer**
  - Presented at CHEP 2009:
    “Harnessing multicores: strategies and implementations in ATLAS”, S.Binet et al.
- Later on completely **rewritten in C++** to follow Gaudi Component Model
- Currently being actively used for running **ATLAS production jobs on multi-core resources on the Grid**

Workflow


Schematic View of ATLAS AthenaMP
init
OS-fork
core-0
WORKER 0:
Events: [0, 5, 8,...]
core-1
WORKER 1:
Events: [1, 7, 10,...]
core-2
WORKER 2:
Events: [3, 6, 9,...]
core-3
WORKER 3:
Events: [2, 4, 12,...]
fin
Output File
fin
Output File
fin
Output File
fin
Output File
fin
SERIAL: parent-init-fork
PARALLEL: 4 workers event loop + fin
SERIAL: finalize

Workflow (contd.)

- The **master process** goes through the initialization phase and then **forks N sub-processes (workers)**
- By **delaying fork as much as possible**, we increase the amount of memory shared between the workers
- AthenaMP implements **different strategies for scheduling workload to the worker processes.** Each of these strategies is implemented by a specialized **Gaudi AlgTool**
- The workers **retrieve event data independently from each other**. They process events and write their own output files to the disc
- After all events assigned to the given job have been processed, the **master usually proceeds with merging workers' outputs**

Assigning workloads to the worker processes

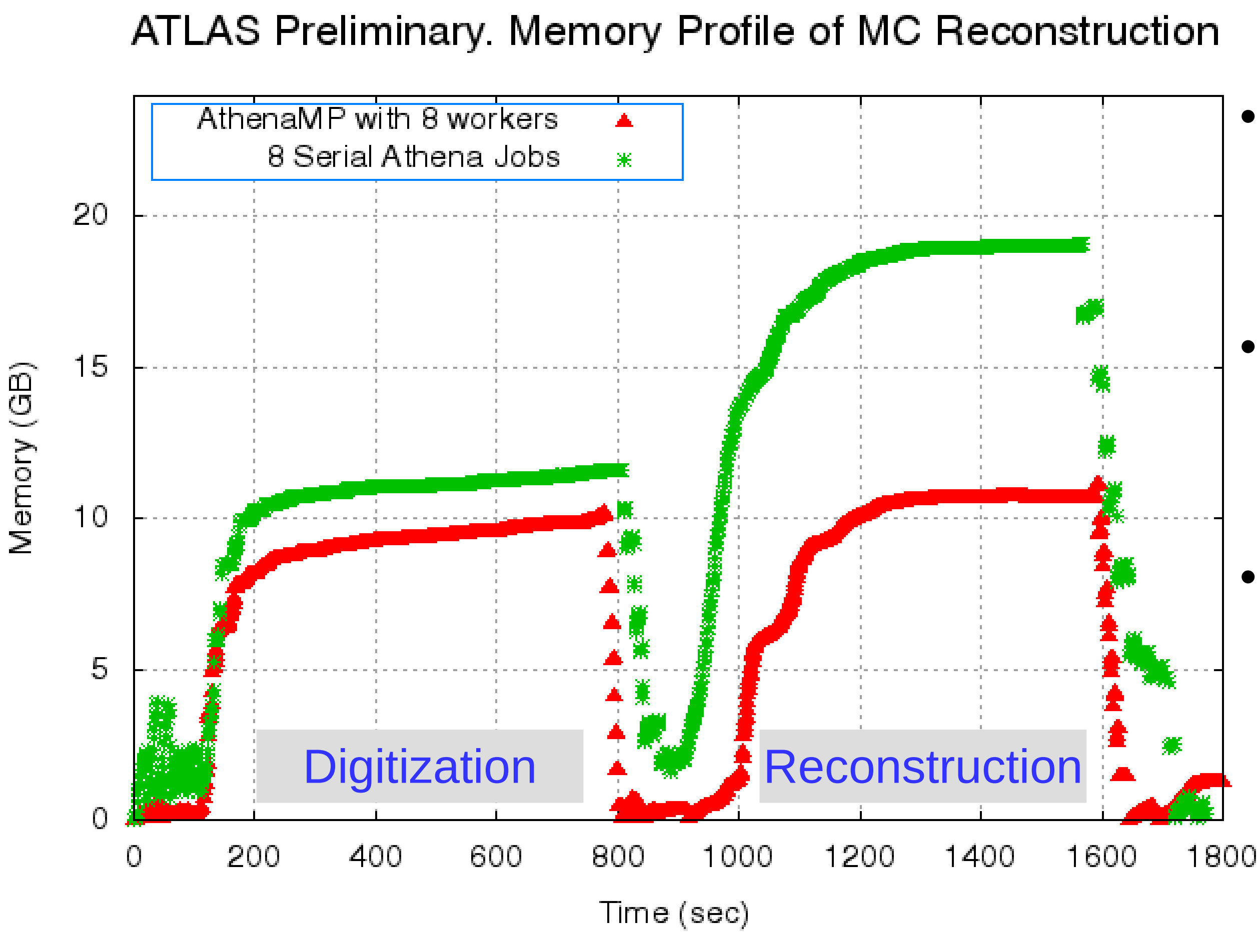
- The complexity of ATLAS Event Data Model does not allow us to directly exchange event data objects between processes
  - ✓ **The only exception is RAW data reconstruction**, where we can pass around event data in the form of `void*` memory buffer
  - ✓ We intend to work on the ATLAS persistency infrastructure in order to address this issue (see "Future Developments" section later in this talk …)

- For the time being, AthenaMP delivers workload to the worker processes in the form of **event identifiers** – either **integers** (event position in the input file) or **strings** (unique event token)

- For each of the above scenarios AthenaMP uses a **special auxiliary process**, which distributes event identifiers/data between worker processes

- The sub-processes of AthenaMP communicate to each other using IPC mechanisms (**shared memory, shared queue**)

Saving memory



- RSS of 8 serial jobs compared to RSS of one AthenaMP job with 8 workers
- Actual memory savings depend on the job type and configuration
- In this example: **AthenaMP reduces overall memory footprint by 45% at the reconstruction step**

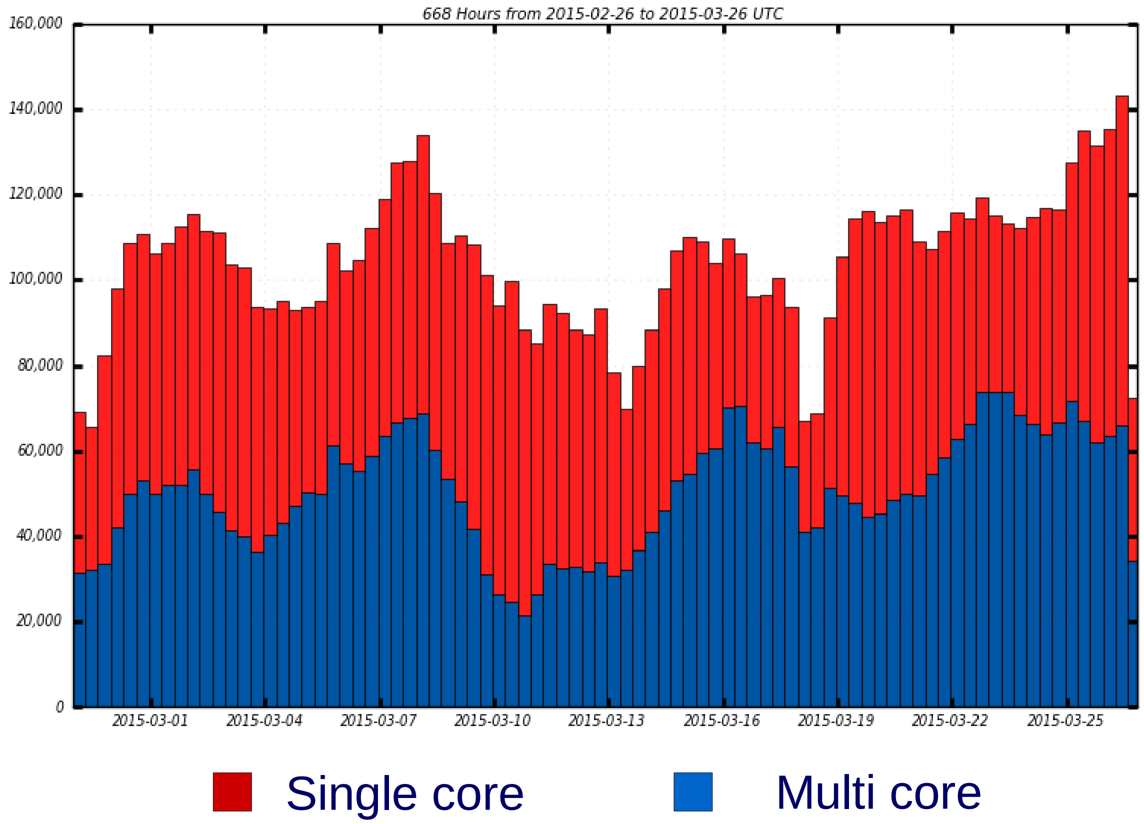
- This plot has been obtained by running test jobs on otherwise empty machine and profiling system memory with `free`

Distributing event numbers. Shared Queue

- AthenaMP uses **shared event queue** for distributing event numbers – either individually or in chunks – between worker processes
- The events are assigned to workers on the “first come first served” basis
  - A worker pulls new event from the queue after it finished processing the previous one
- Such dynamic distribution of the workload guarantees load balancing on the workers
- Shared event queue is **the default strategy for running AthenaMP jobs on the Grid**

AthenaMP on the Grid

Number of CPU cores used by ATLAS production jobs



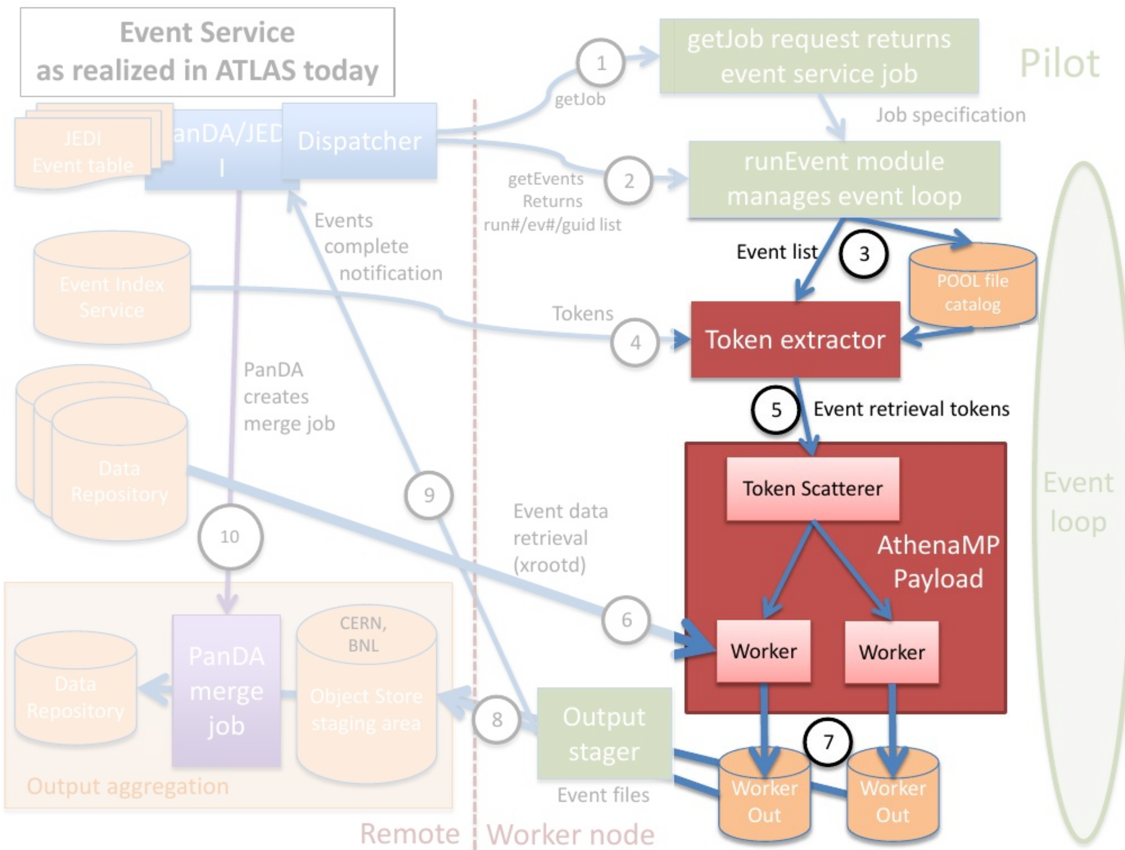
- Today ATLAS runs a substantial fraction of its production workloads on the Grid using AthenaMP
- The workloads include (but are not limited to) Geant4 simulation and simulated data reconstruction

- The plot shows the number of CPU-cores used by ATLAS production jobs – serial and MP – on the Grid in March 2015

Fine grained Event Service

- **Event Service** – a new approach to event processing in ATLAS
  - ✓ Job granularity changes from files to individual events
  - ✓ Deliver only those events to a compute node, which will be processed there by the payload application, don't stage in entire input files
- Event Service is agile and efficient in exploring **diverse, distributed, potentially short-lived (opportunistic) resources:** "conventional resources" (Grid), supercomputers, spot market clouds, volunteer computing
- For more details about Event Service see the presentation by Torre Wenaus at CHEP2015: *"The ATLAS Event Service: A new approach to event processing"* (Contribution #183)
- Event Service uses AthenaMP as **payload application for event processing**

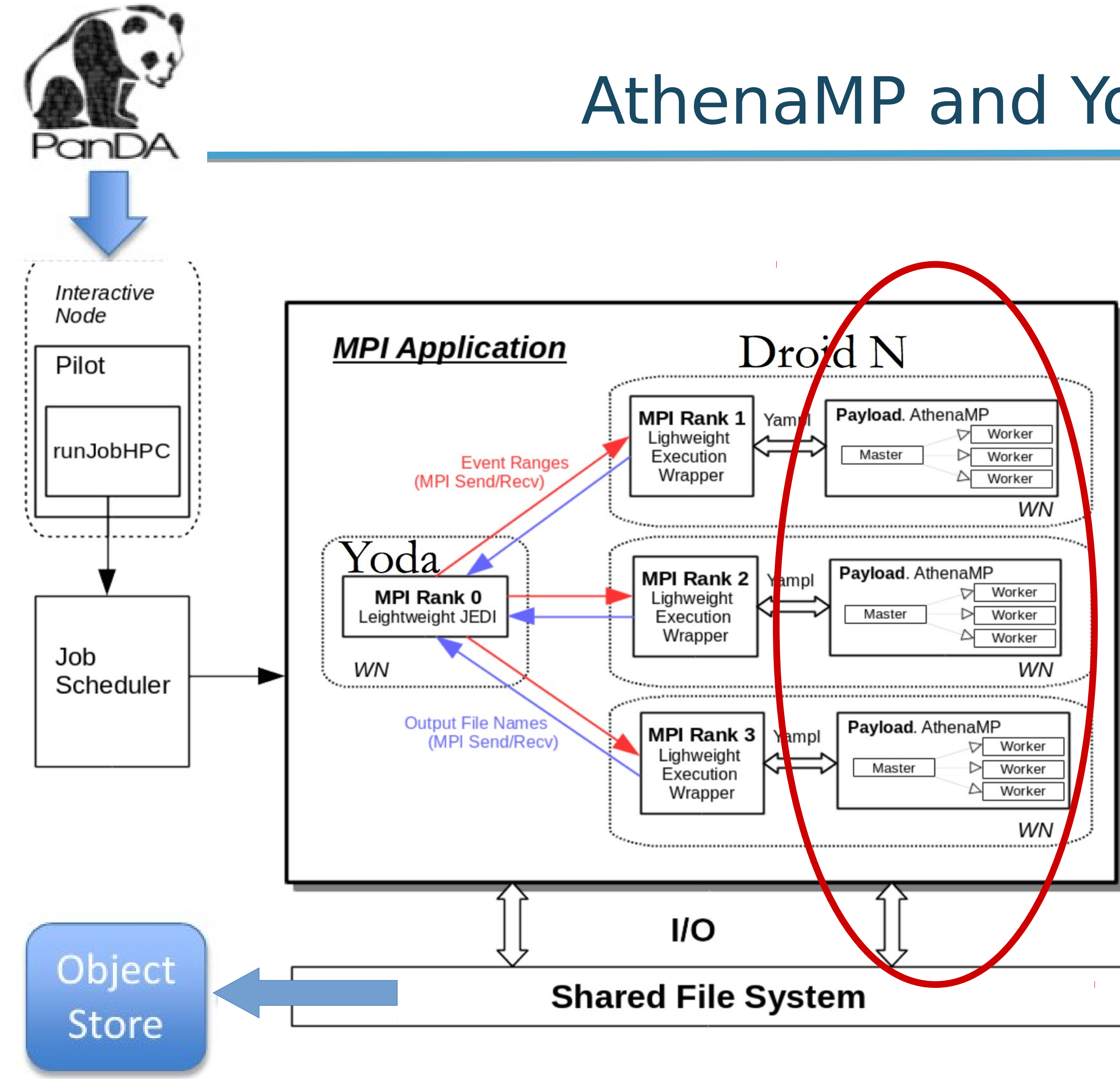
AthenaMP and Event Service



- For the Event Service AthenaMP uses the strategy of **distributing event tokens** to the worker processes
- The tokens are retrieved from external source by a specialized AthenaMP sub-process **Token Scatterer**
- Workers retrieve event data using the token. **Data may be local or remote**
- AthenaMP is configured to **write new output file for each processed event range**

- The latter functionality is implemented by the **Output File Sequencer** – a new mechanism developed in ATLAS, which is currently being used only by AthenaMP within Event Service

AthenaMP and Yoda



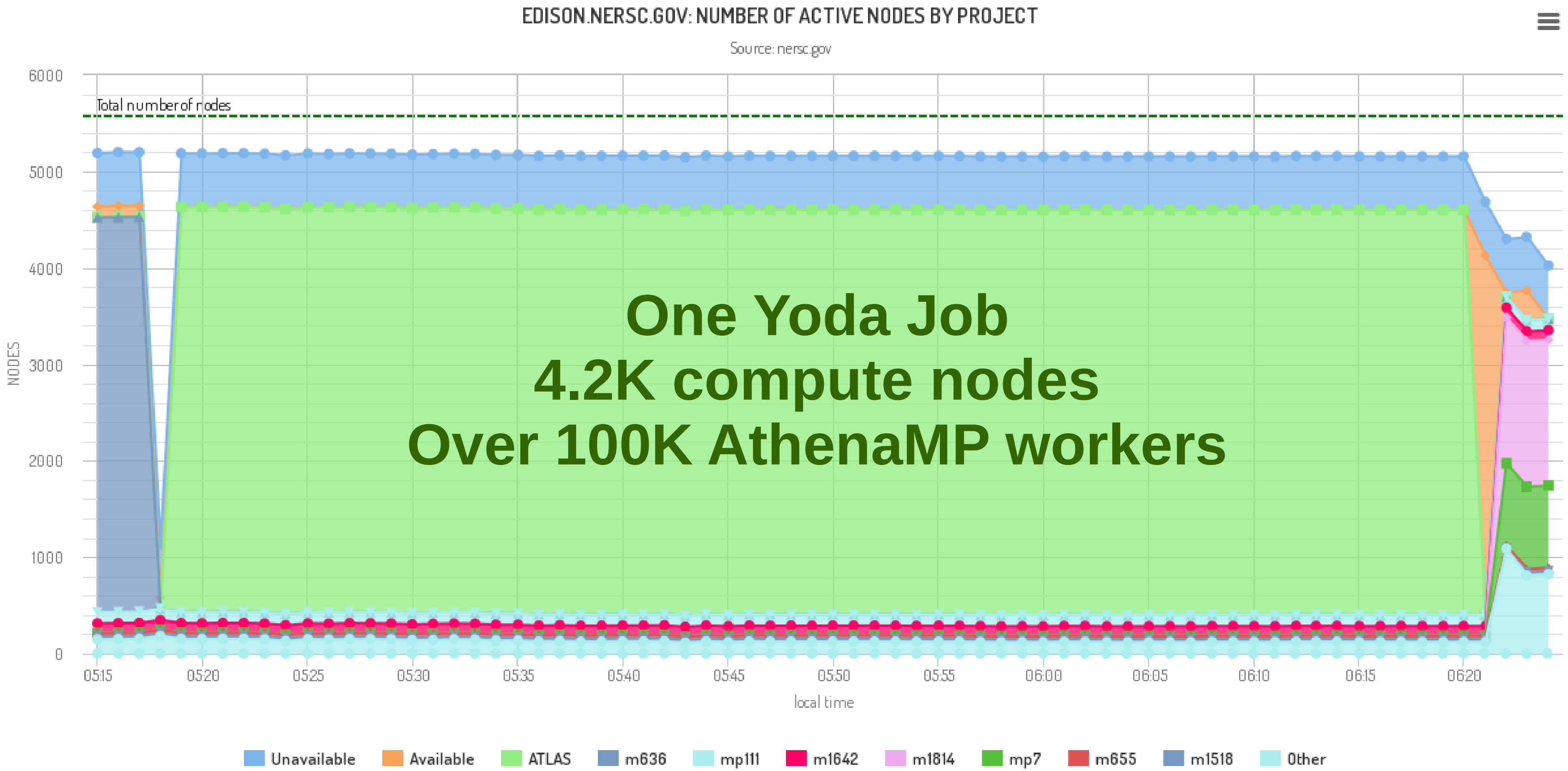
- **Yoda –** MPI-based implementation of the Event Service, designed to run on supercomputer systems with no internet connection from the compute nodes

- For more information about Yoda see the presentation by V. Tsulaia at CHEP2015: *"Fine grained event processing on HPCs with the ATLAS Yoda system"* (Contribution #140)

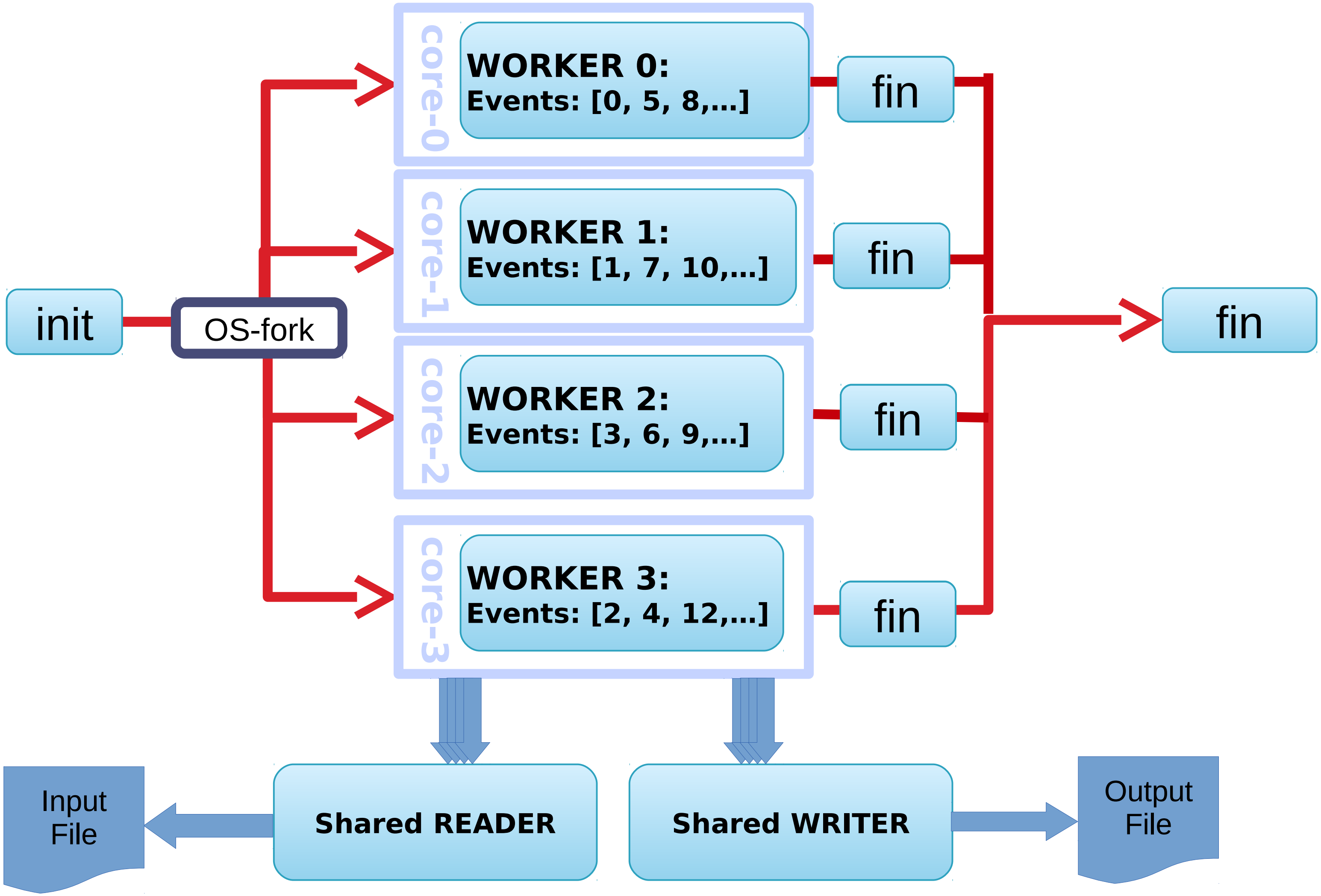
- **Payload components (AthenaMP) of the conventional event service and Yoda are absolutely identical**

V.Tsulaia, ATLAS, CHEP 2015

Running at large scale



Future developments: shared I/O workers



Summary

- AthenaMP has successfully passed the physics validation procedure and now is being widely used for running ATLAS workloads on the Grid
- Originally developed for optimizing memory footprint of reconstruction jobs, AthenaMP later became an efficient mechanism for processing fine grained workloads
- Recent tests on supercomputers demonstrated, that AthenaMP can run efficiently within distributed applications at large scale