Multicore in Production: Advantages and Limits of the Multiprocess Approach

ACAT 2011, 08-Sep-2011

Sébastien Binet, Paolo Calafiura, Wim Lavrijsen, Charles Leggett, David Lesny, Manoj Kumar Jha, Horst Severini, Douglas Smith, Scott Snyder, Mous Tatarkhanov, Vakhtang Tsulaia, Peter van Gemmeren, Andrew Washbrook

Outline

- **Introduction: the concept of Athena MP**
  - **Athena** – the common ATLAS reconstruction / analysis framework
- **Pros and cons of the multiprocess approach**
- **Validation work**
- **Large scale tests of the Athena MP**
  - CERN Computer Center (“Tier 0” of the LHC Computing Grid)
  - LHC Computing Grid
- **Performance figures**
- **Plans for the future**

Motivation

- **Inefficient to run single instance of sequential event Reconstruction or Simulation program on modern servers**
  - Uses only one out of N available cores
  - Need to have a parallel solution in order to exploit CPU resources efficiently
- **Embarrassingly parallel code**
  - Solving many small, but independent tasks simultaneously with little to no need for coordination between them
- **Boundary conditions**
  - ATLAS has large code base mostly written and designed in “pre-multi-core” era
  - Reconstruction code mainly process-based and single-threaded
  - Modifications have to be as transparent as possible

Possible Parallel Solutions

- **Several individual jobs in parallel - Athena Multi-Job (MJ)**
  - Current default
  - Simplest: no code rewriting
  - **No sharing of resources** either
- **Parallel execution of event processing loops by separate processes - Athena Multi-Process (MP)**
  - Minimal changes to the existing code: just a few “core” packages
  - Worker processes share memory acquired at initialization phase
    - **Allows for efficient usage of memory resources**
- **Multithreaded approach**
  - Started by ATLAS HLT in 2002, but never got beyond proof of concept
  - Too much code to port; too many developers to educate

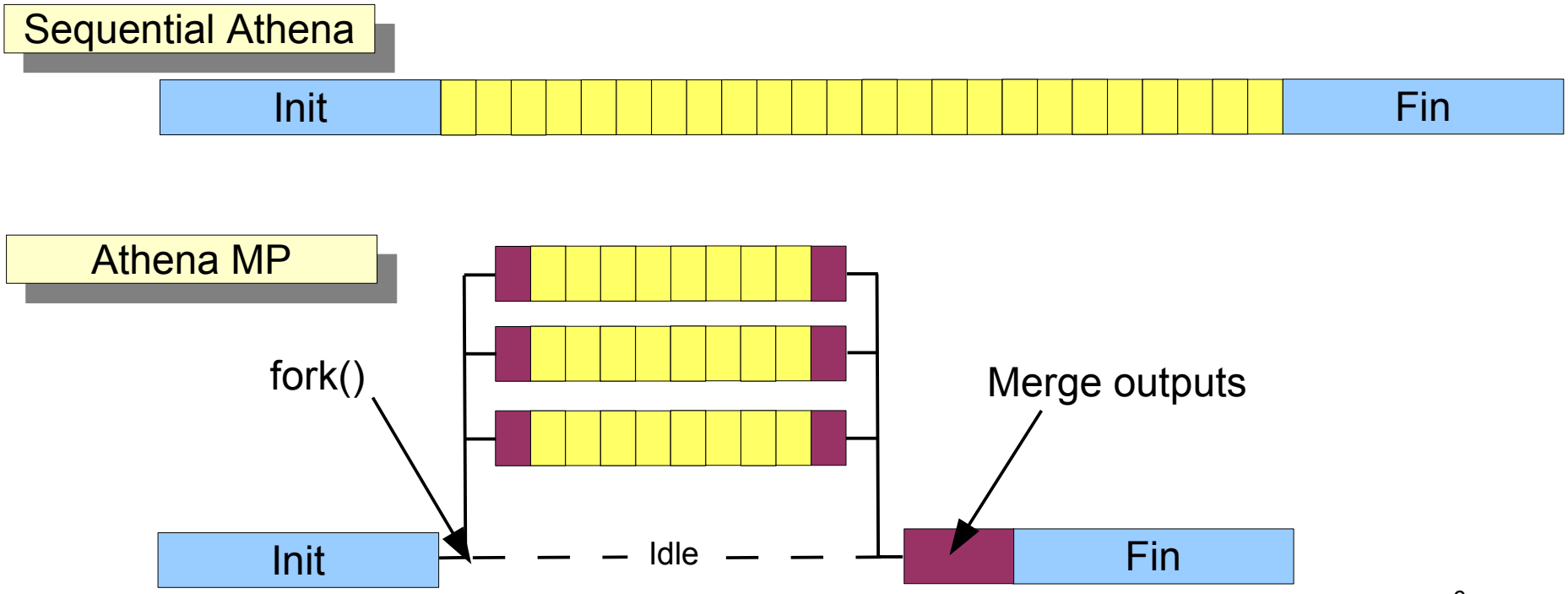
Will focus on Athena MJ and Athena MP in this talk

Athena MP: basic concepts

- **Start standard Athena job as single process**
- **Go through** *load*->*configure*->*initialize* **stages then `fork()` *n* worker processes**
  - `fork()` clones a process using its entire address space
  - Linux uses **Copy On Write (COW)** when forking a process
    - **Memory is shared up to the point a process writes to it**
    - **Memory will be copied and affected changes become unshared**
  - Do `fork()` as late as possible in order to maximize memory sharing between workers
    - Athena MP demonstrates quite remarkable reduction of overall memory footprint if **fork()** is done **after processing the 1st event** *(see later in this presentation)*

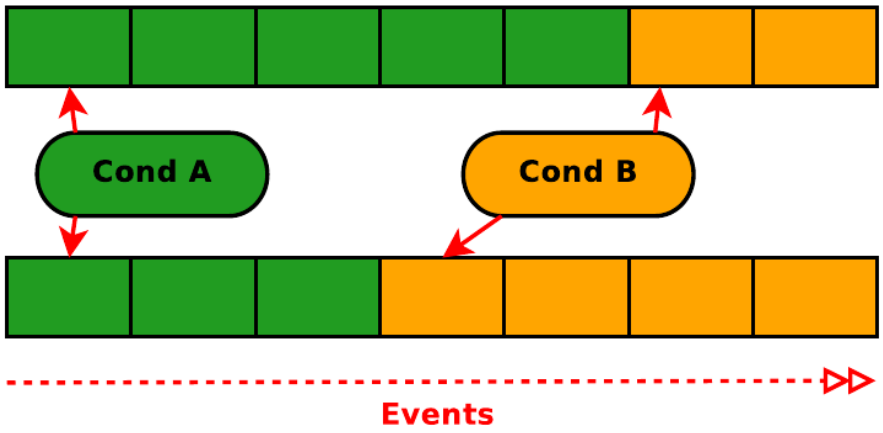
Athena MP: basic concepts (2)

- **Let the workers process all events**
  - Two ways of distributing events between workers:
    - **Queue** (default). Workers get new events on the 'first come first served' basis
    - **Round Robin**. Static schema of distributing events. (*Example: Nworkers=2, worker #1 gets odd events and worker #2 gets even events*)
- **At the end merge all output files produced by workers**

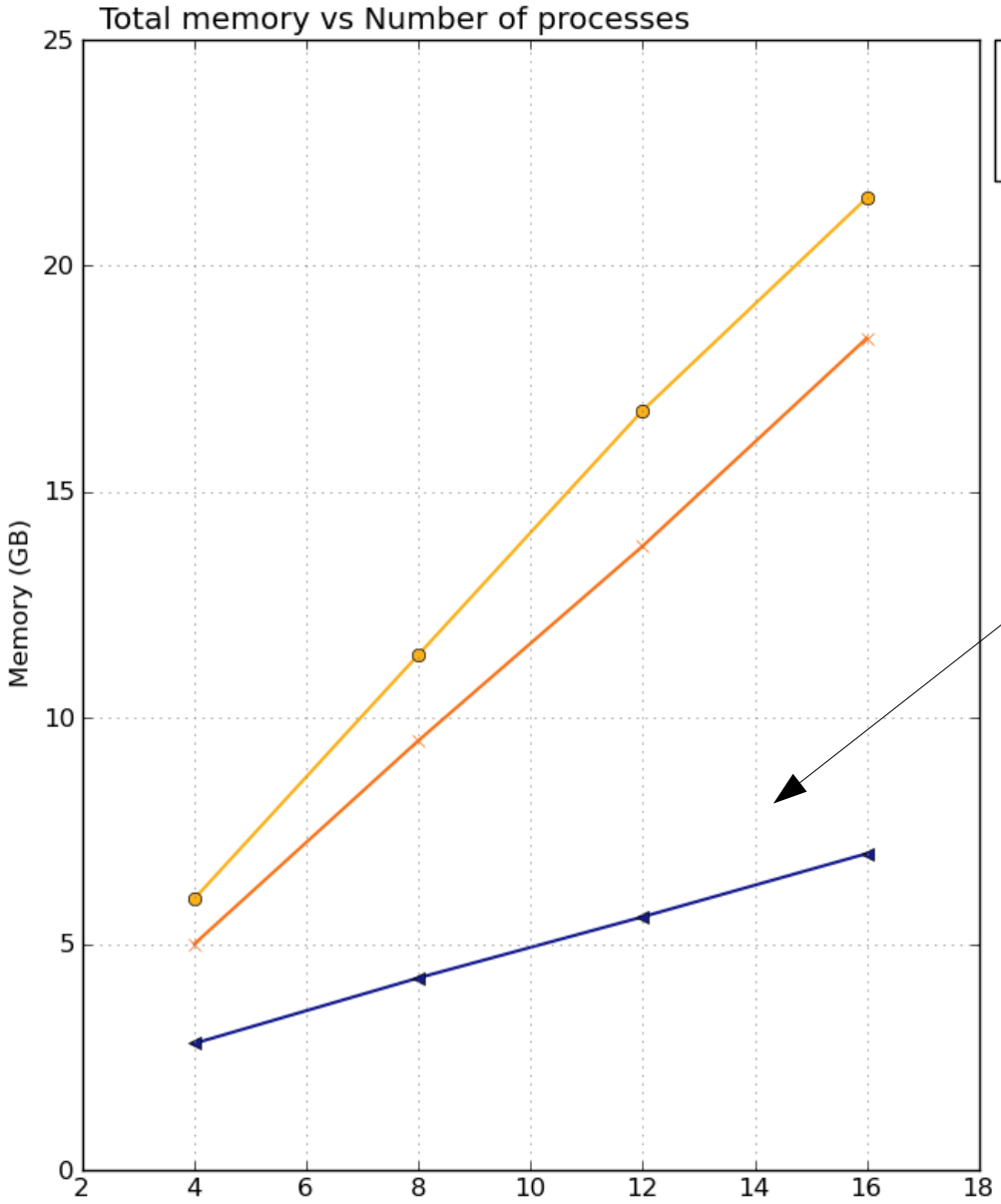


Athena MP and COW

- **Pros**
  - If memory can be shared between processes, it will be shared
  - No need to do anything on our side to achieve that – let the OS do the work
  - No need to worry about memory access synchronization
  - Code changes restricted to a few framework packages
- **Cons**
  - Once unshared the memory cannot be re-shared again
    - **Detector conditions change during the run, need to get new data from the Conditions database**



Sharing memory between workers



Delayed fork()
1.4GB shared

- Standalone test running standard Athena reconstruction with different number of processes
- Platform:
  - Intel Xeon L5520 @ 2.27GHz
  - 8 Cores
  - Memory 24GB
  - Hyper-threading ON

Process management

- **Athena MP uses *python multiprocessing* module**
- **MP semantics hidden inside Athena in order to avoid client changes**
  - Special MP Event Loop Manager
- **When it is time to `fork()` create Pool of worker processes**
  - Initializer function
    - **Change work directory**
    - **Reopen file descriptors**
  - Worker function
    - **Call executeRun of the wrapped Event Loop Manager**
- **Once all workers have returned proceed with output file merging**

I/O

- **Introduced explicit file management in Athena**
  - Core Athena I/O components implement interface where they declare their input and output files
- **Output**
  - Each worker has its own output file(s) which is/are written into worker's run directory
  - When control is passed back to the master process it merges worker's output files
- **Output file merging is a tedious process, which has very negative impact on the overall performance of the Athena MP**
  - Until recently for merging output files we ran separate Athena jobs in serial
  - **Now we switched to a fast merger tool, which reduced merging time by an order of magnitude**

Validation work

- **Several months of focused effort with ultimate goal to make Athena MP production quality**
- **First round of validation & debugging**
  - Make sure different types of jobs can run in MP (Simulation, Reconstruction).
  - Fix various bugs and crashes
  - Check that output files look reasonable and memory/CPU consumption behaves as expected
- **Second round**
  - Validate the output by comparing MP to MJ
  - By design Athena MP reshuffles events in the output file, which makes validation process more difficult

Still not there yet … but at least we are now able to run large scale tests on the Grid and at CERN Tier 0

ATLAS Whole-node Task Force

- **Aim to have in standard production on the Grid in coming months**
- **Need to define multi-core queues and job arguments**
- **Participating sites need to modify batch system queues and (possibly) adjust middleware configuration**
- **Initial phase**
  - First testing queue setup at CERN
  - Defined as whole-node machines, dedicated machines scheduling only these multi-process jobs
  - External sites volunteered to run multi-core jobs:
    - **OU_OSCER_ATLAS (US), RAL, ECDF, Glasgow (UK), INFN-T1 (IT)**

Changes in the Production System (PanDA)

- Extra parameter to queue definition to define the number of processes for Athena to use ("*corecount*")
- Production pilots use this value to set an environment variable, which is used by Athena jobs to set the number of processes
  - If older release does not support multi-process running, the variable is ignored
- Queue can then run any job, and number of processes for Athena are defined at run-time
- Changes made to view the multi-process logs
  - Each worker makes a separate log file in its run directory

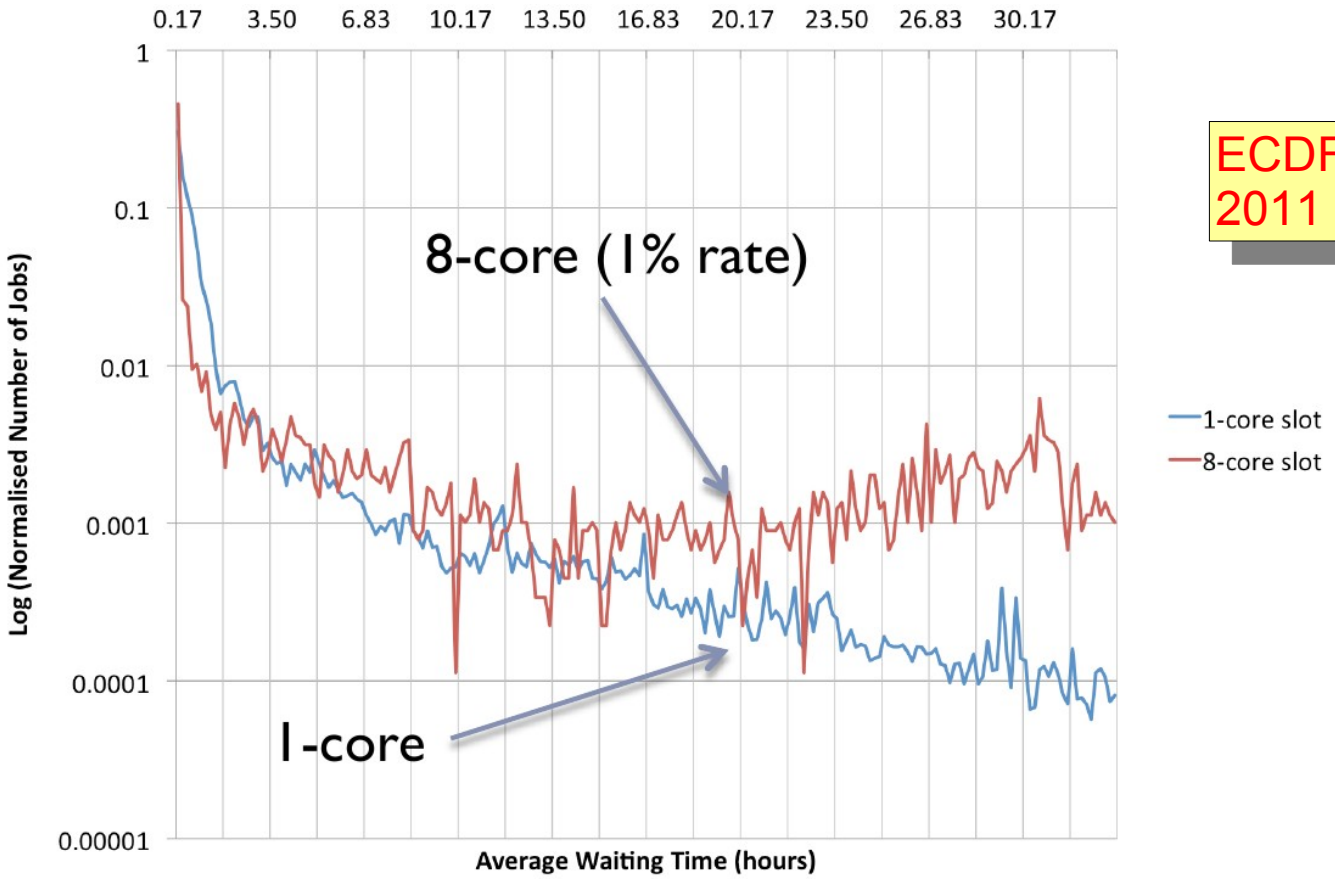
Athena MP on the Grid

- **Job scheduling concerns**
  - Need to have multiple cores free at one time to schedule job
  - Need to have “whole-node” free for jobs of same size to be sequentially scheduled
  - Or need to have virtual machines of same size used for jobs
- **Job efficiency concerns**
  - When we increase Athena MP job size then job failures will result in much larger losses of CPU time

Athena MP on the Grid (2)

- **Concerns of sites**
  - If a queue is setup all nodes on that queue have to be the same
    - Number of CPU cores, memory, hyper-threading etc
  - Sites then see this as “dedicated resources” not shared and we lose overall CPU efficiency if machines are under utilized
  - But worse than that, sites might not like defining all queues needed for heterogeneous sets of resources
- **Possible solution is that the pilot requests jobs of appropriate core-count**
  - Panda will then define a job based on number of input files to match the resources.
  - Then jobs will use up files in a dataset, as the pilot requests them

Whole-node Job Latency



ECDF Jobs in May 2011 (Total 806,141)

- Study over May 2011 for all jobs suggests that scheduler is coping well but some outliers for 8 core jobs observed
- Difficult to anticipate scaling up of whole-node jobs. Significant pile-up foreseen or just an edge effect?

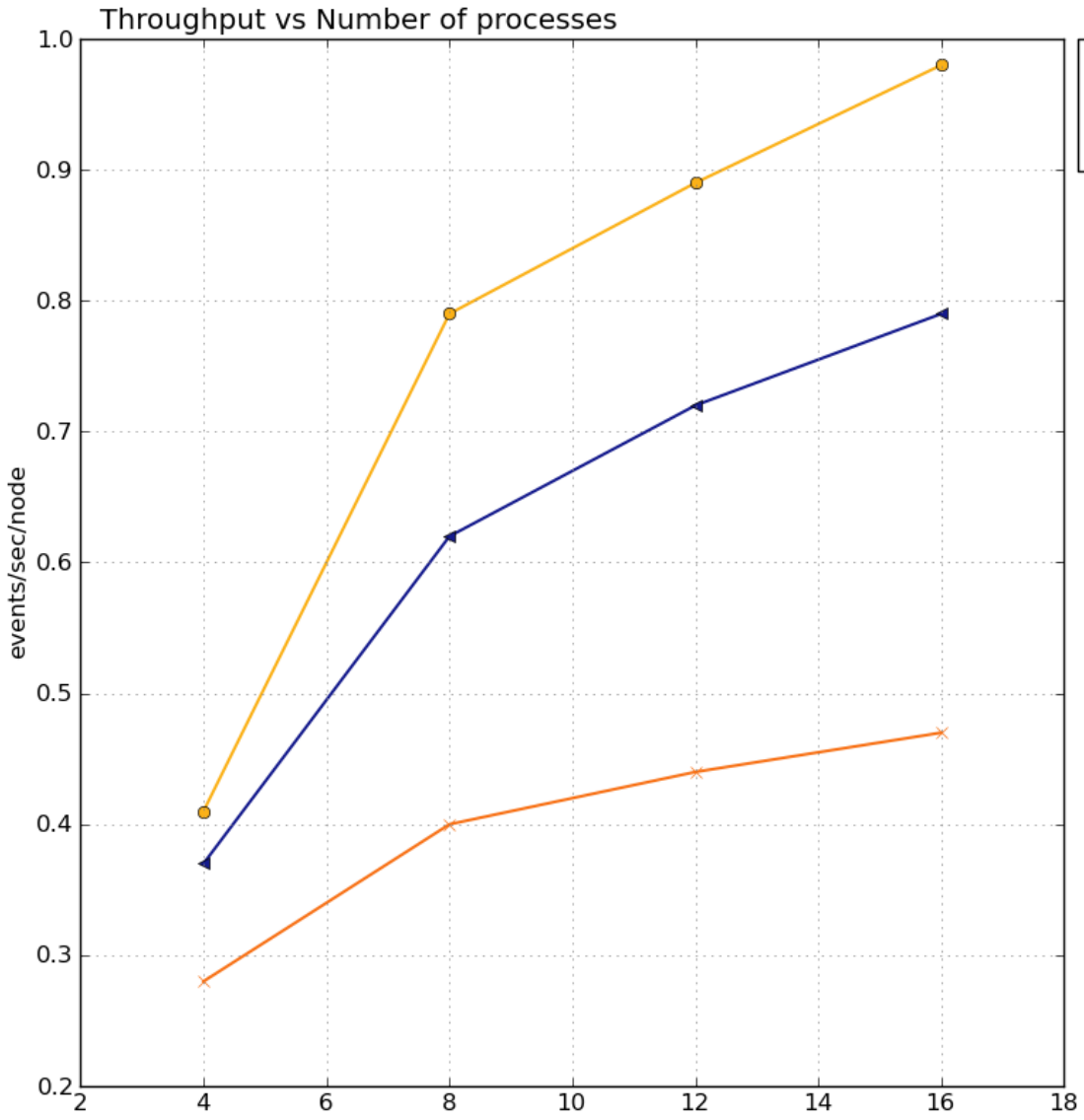
Athena MP at CERN Tier 0

- **Two rounds of tests for the Athena MP at CERN Tier 0 performed over Summer 2011**
- **Same scenario used for both tests**
  - Run standard ATLAS reconstruction over RAW data files in the MP and MJ modes (~2M events processed in total)
  - Monitor performance characteristics
  - Compile list of problems
  - Compare the output
- **In general both of the tests were successful**
  - Very few jobs crashed because of Athena MP related problems (~0.5% of the total number of jobs)
- **The tests confirmed several known problems and revealed a few new ones**
  - Some of them already solved, the others currently being looked at

MP vs MJ: wall time

- **T0 reports for the total wall time for all jobs. MP and MJ jobs of the same size**
  - Round 1 (core-days)
    - MP 360, MJ 173 MP slower by x2.08
  - Round 2 (core-days)
    - MP 106, MJ 72 MP slower by x1.47
- **The speedup of Round 2 was achieved by switching to the fast merger for output files**
- **The remaining difference mostly because of the job size effect**
  - More details later in this presentation

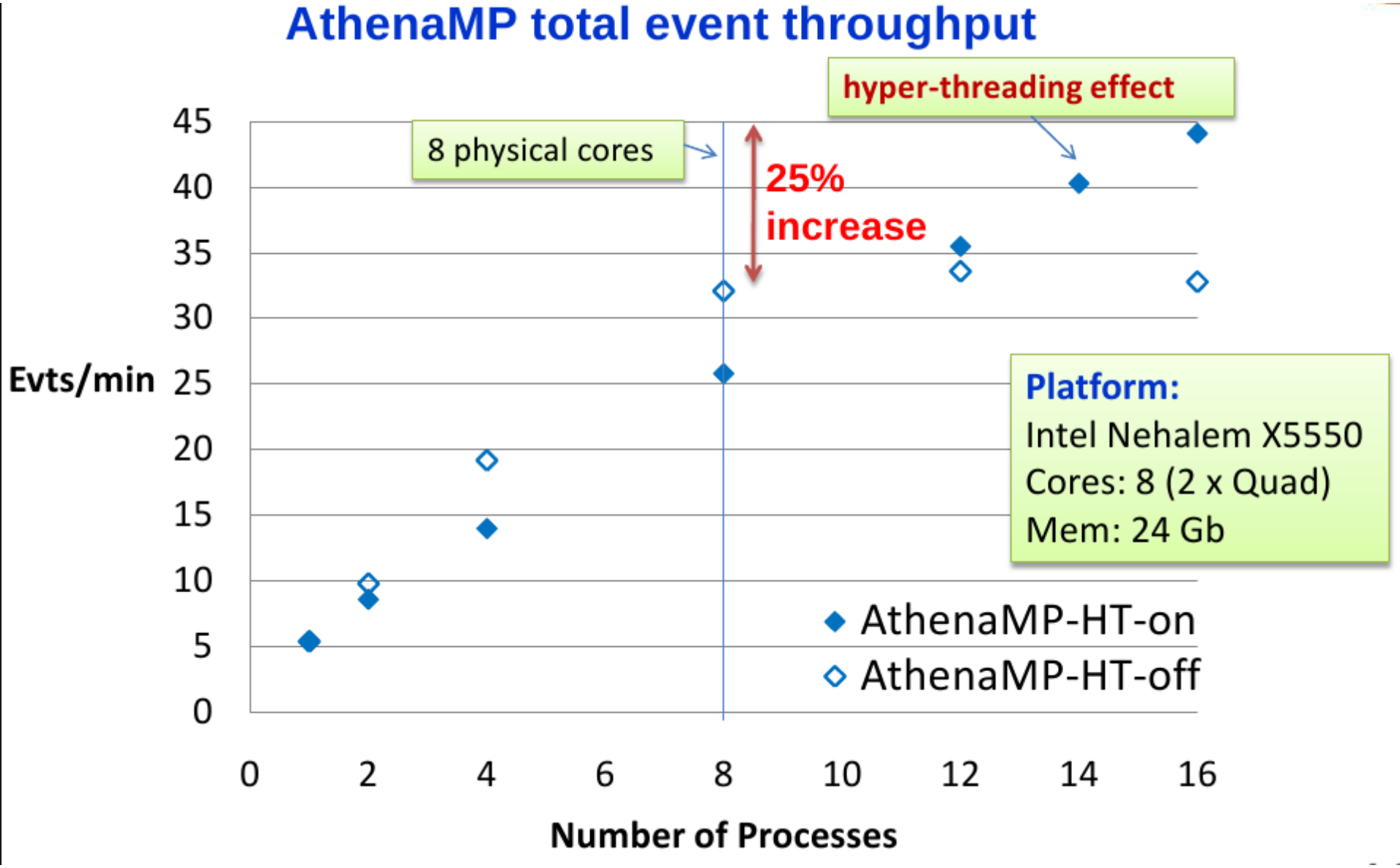
MP vs MJ: Throughput



- ✔ Big improvement coming from the fast merger
- ✔ Remaining difference mainly from the job size effect

- Standalone test running standard Athena reconstruction with different number of processes
- **MP and MJ jobs of the same size**
- Platform:
  - Intel Xeon L5520 @ 2.27GHz
  - 8 Cores
  - Memory 24GB
  - Hyper-threading ON

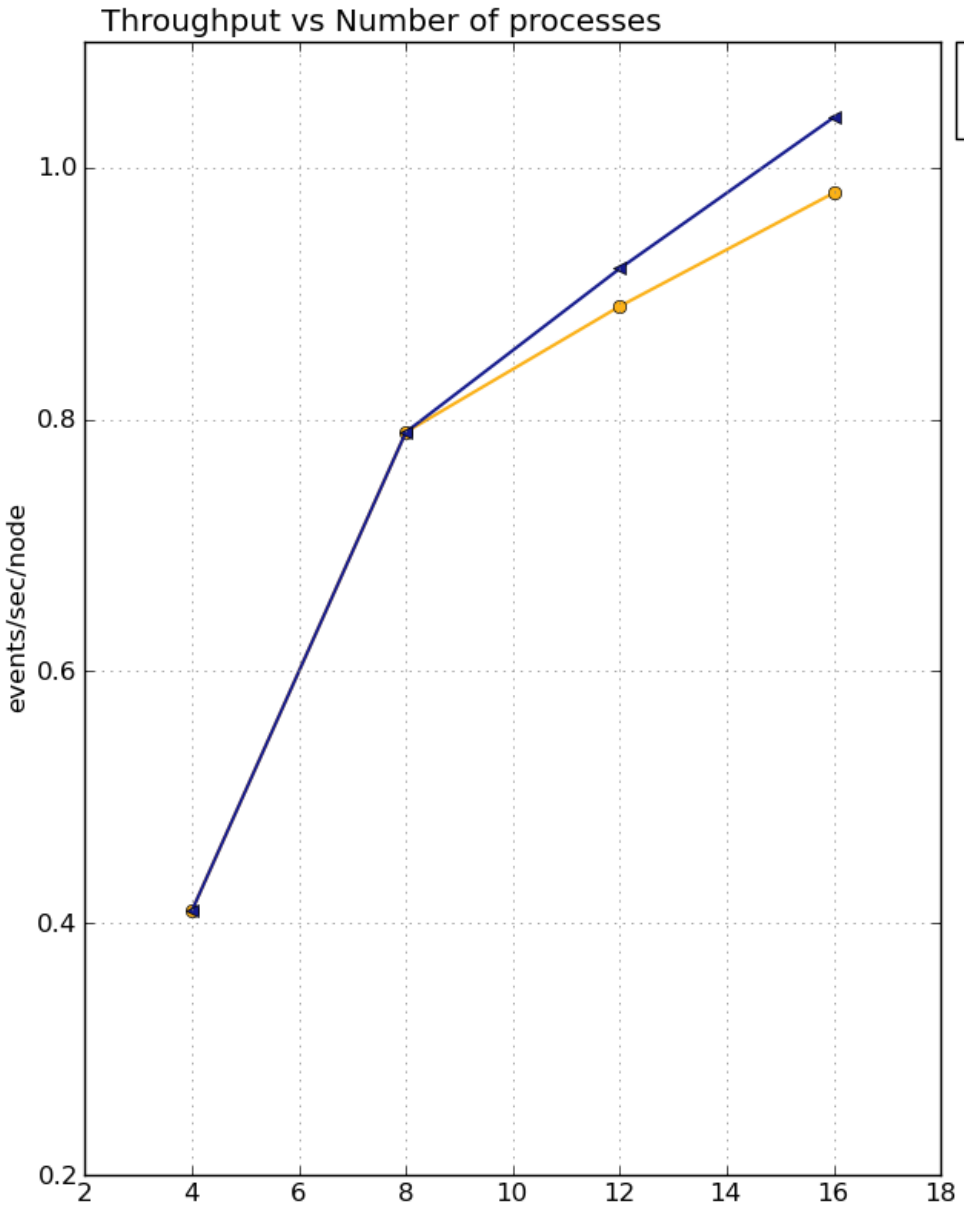
Effect of the Hyper-threading



Tackling job size problem

- **Suppose we have 8 core machine and we need to run reconstruction over 8 files with events**
- **Scenario 1 (current). MP and MJ jobs of the same size**
  - 8 sequential jobs run in parallel
  - 8 MP jobs with 8 workers run in sequence
- **Scenario 2. Large size MP job**
  - 8 sequential jobs run in parallel
  - 1 MP job with 8 workers processing all input files in one go
- **In order to compete in efficiency with MJ jobs we need to increase MP job size**

What effect do we expect?



- ✔ **This is just an estimate!**
- ✔ **MJ** numbers are real
- ✔ **MP** numbers are obtained by extrapolating the throughput of small size jobs

- Standalone test running standard Athena reconstruction with different number of processes
- Platform:
  - Intel Xeon L5520 @ 2.27GHz
  - 8 Cores
  - Memory 24GB
  - Hyper-threading ON

Tackling job size problem (2)

- **Possible solution**
  - Run Athena MP jobs over N input files where N >= number of workers
  - At the beginning assign one input file per worker
    - **No need to merge outputs**
  - Use remaining input files for the worker load balancing
- **Such mechanism would require substantial modifications in the current implementation of Athena MP**
  - The system must be flexible enough to run either over single or over multiple input files
- **The work will start in coming weeks**

Next development: I/O workers

- **In order to further improve performance of the Athena MP we plan to develop specialized I/O worker processes:**
  - **Event source.** Read data centrally from disk, deflate once, do not duplicate buffers
  - **Event sink**. Merge events on the fly, same memory, no merge post processing
- **The event workers will remain as before**
  - Minus all I/O dependencies
  - Will communicate with I/O workers via shared queues
- **… Longer term development, after we deal with job size issues**

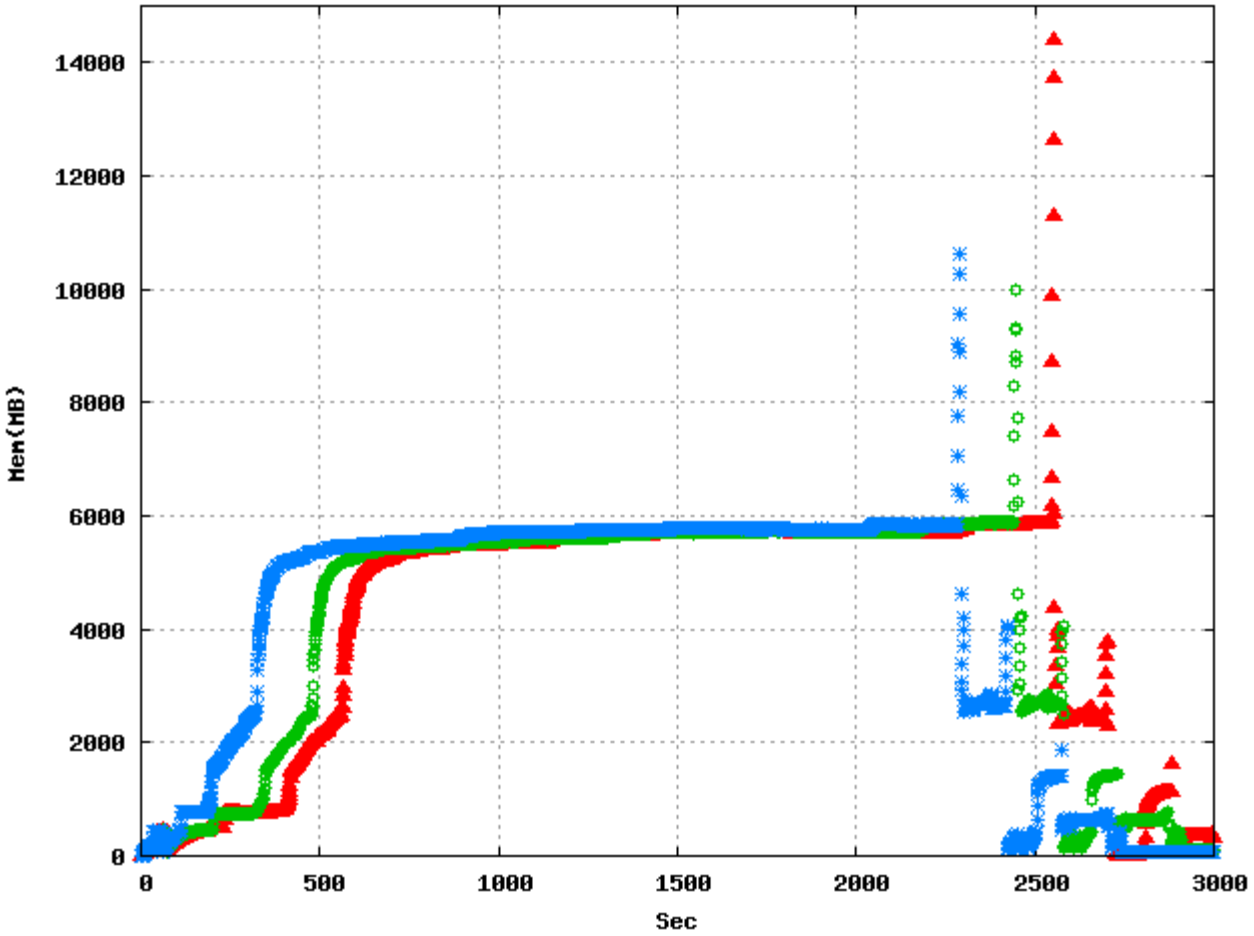
Summary

- **After several years of development and validation effort Athena MP is reaching the production quality**
  - There is a strong interest from ATLAS computing management to start using MP in production (T0 in particular)
- **Late forking leads to more optimal usage of memory resources**
- **Future developments will be focusing on the Athena MP efficiency improvements**
  - Job size adjustment
  - I/O worker processes

Backup

Spikes in memory consumption

Total memory of one Athena MP reconstruction job vs Wall Time



- Same job run 3 times on the same machine
- Spikes at the end of worker processes caused by massive memory unsharing
- Spike sizes non reproducible (race conditions)