ATLAS Whole-node Plans

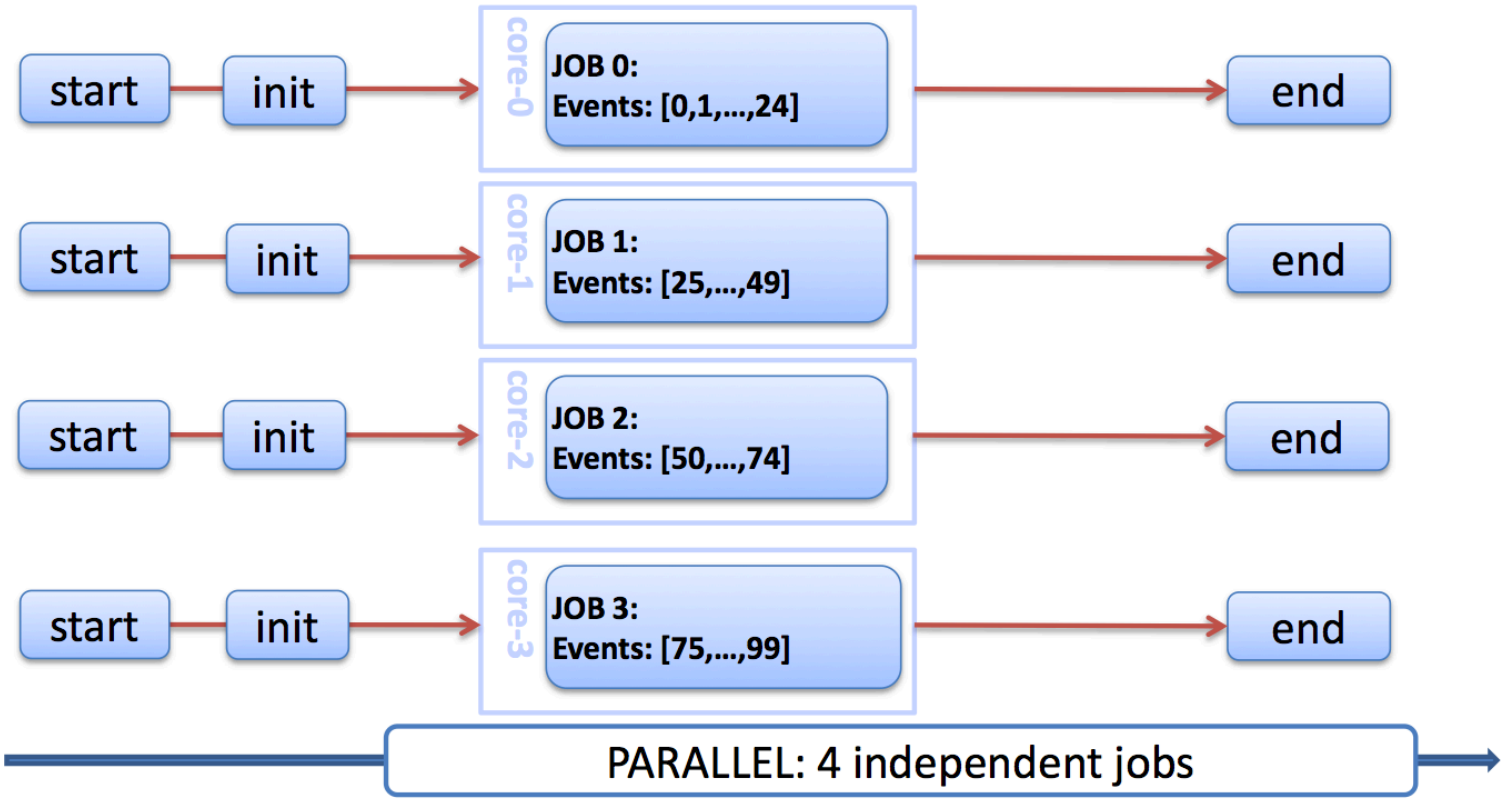
Andrew Washbrook

Future Computing Workshop, 16th June 2011, University of Edinburgh

(On behalf of Douglas Smith (CERN))

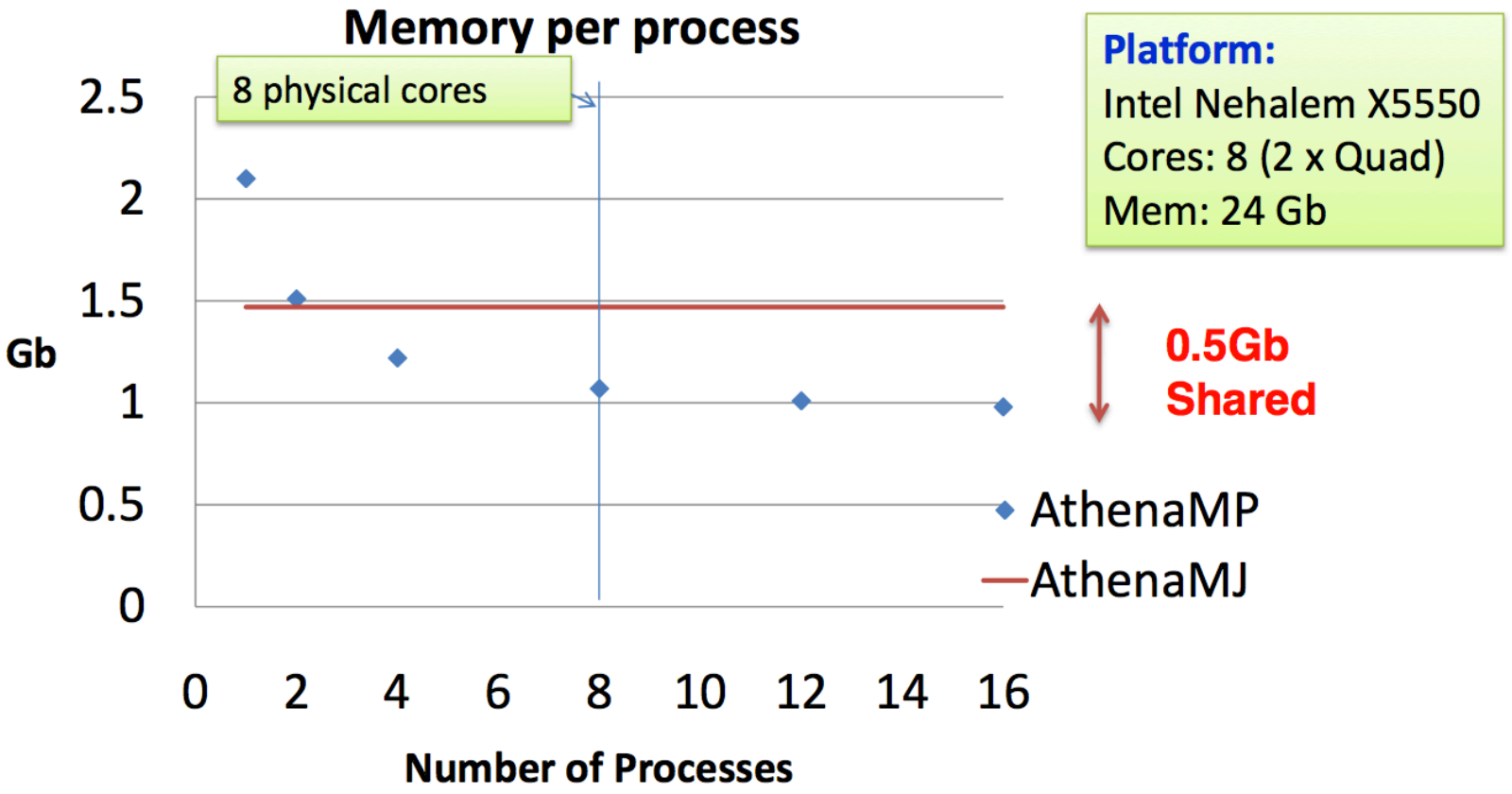
AthenaMP Motivation

- Do we need a many-core solution for Athena?
  - Inefficient to run one instance of Athena on modern servers.
  - So why not run "AthenaMJ"?



Memory Utilisation

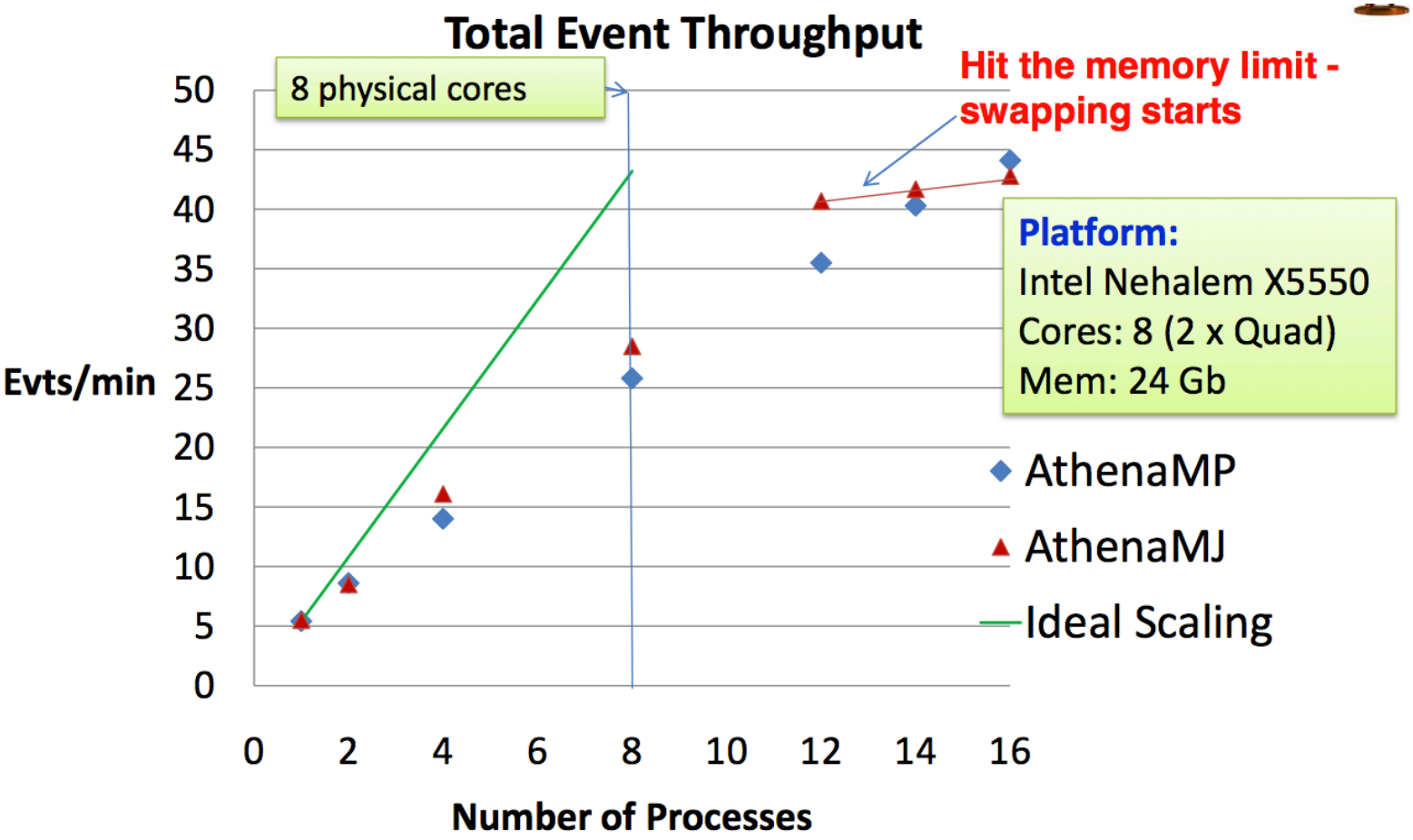
- Is the host memory being used efficiently?



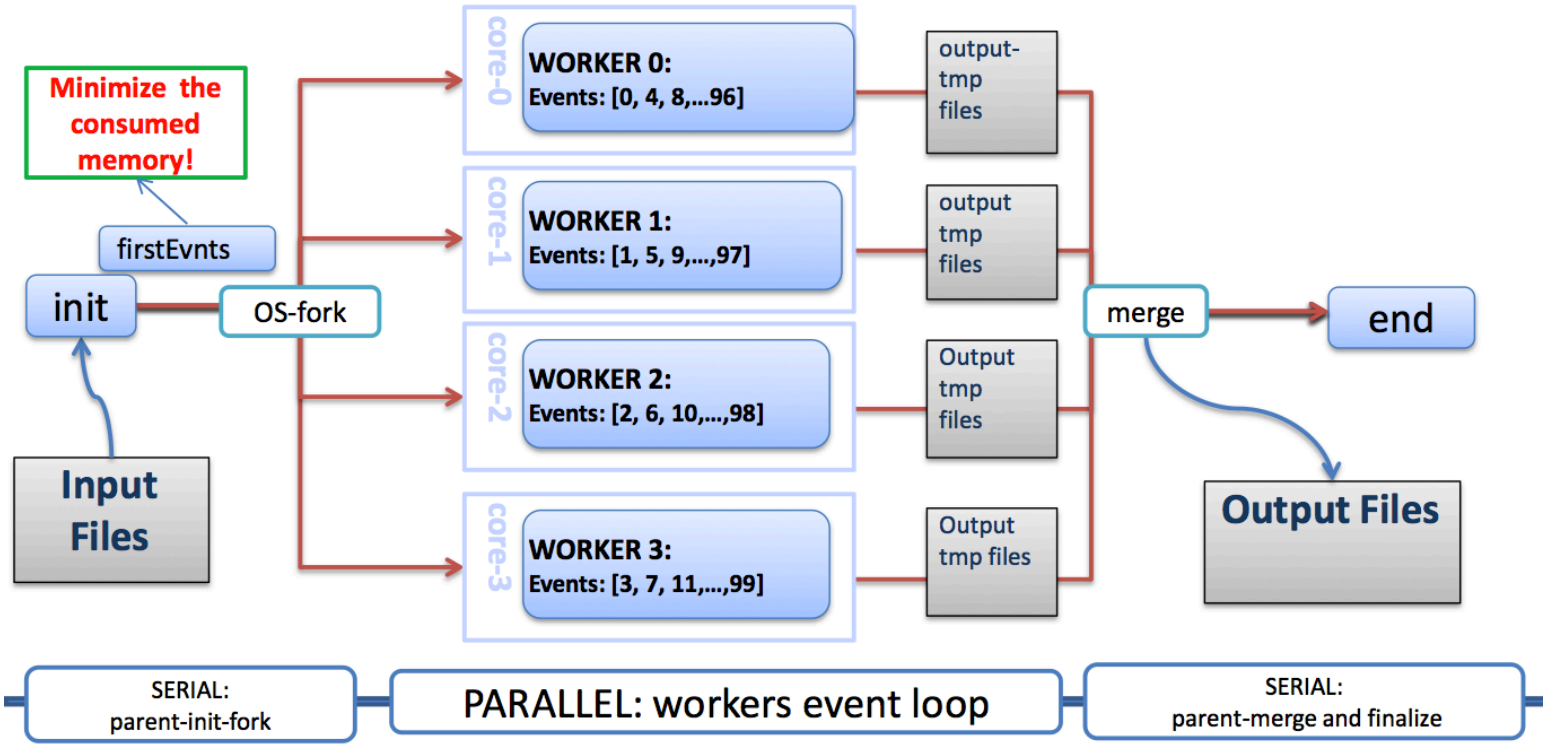
0.5 GB physical memory shared per process

Memory Utilisation

- Performance effect as number of processes exceeds cores?

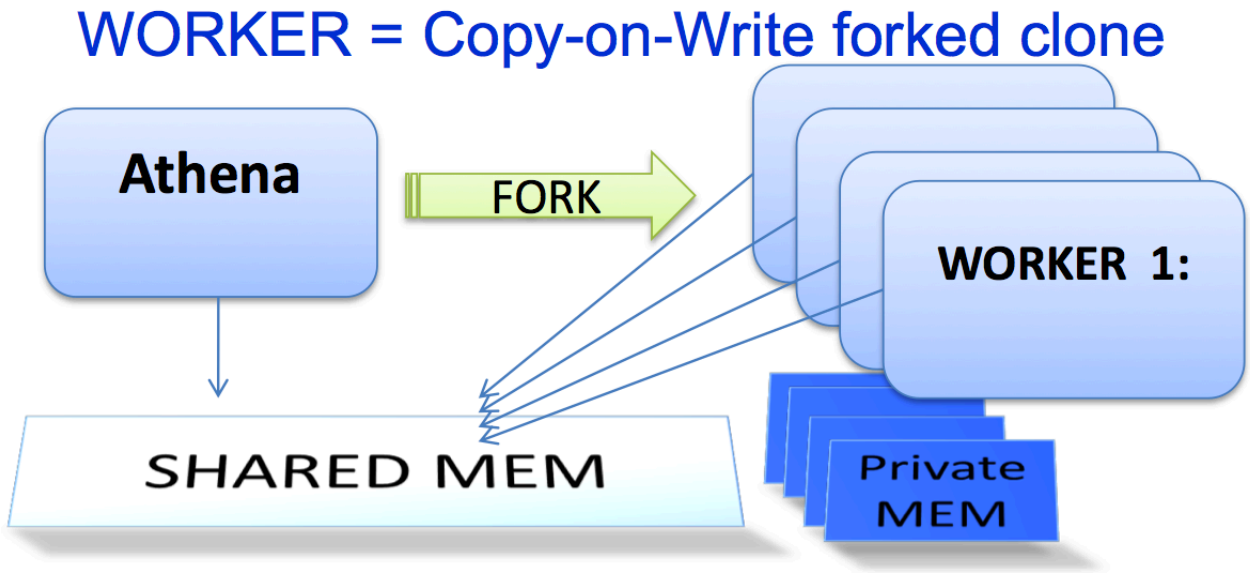


AthenaMP Design



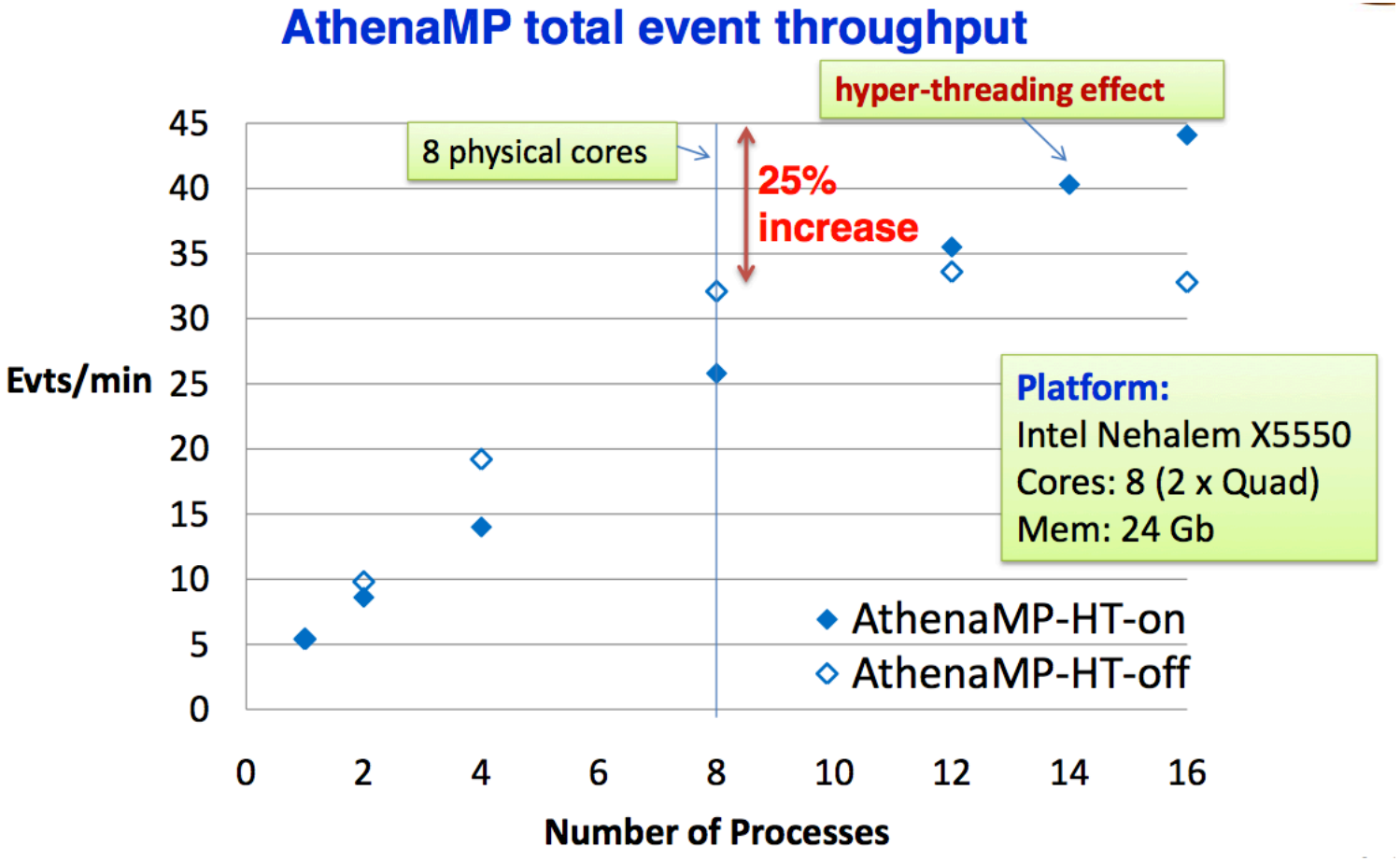
- Worker is forked clone of serial process
  - Master process memory is used by workers
  - Processes see the same physical memory while reading

AthenaMP Design: Copy on Write



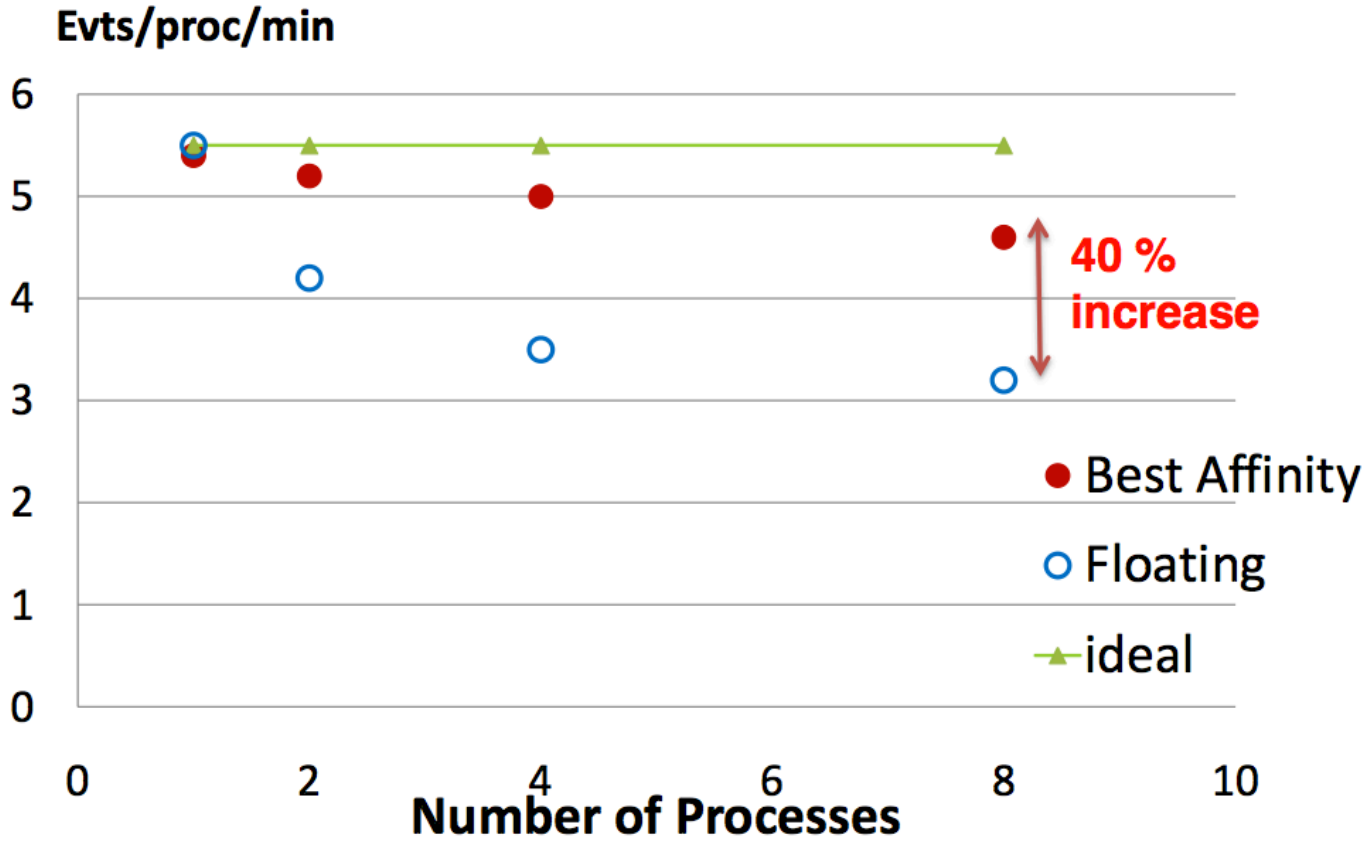
1. Initialize serial version of Athena
2. Create pool of event processing workers
   - Fork workers using python multiprocessing package
   - Processes see the same initial physical memory
   - New allocations and touched pages created in private memory space.
3. Merge output

Hyperthreading



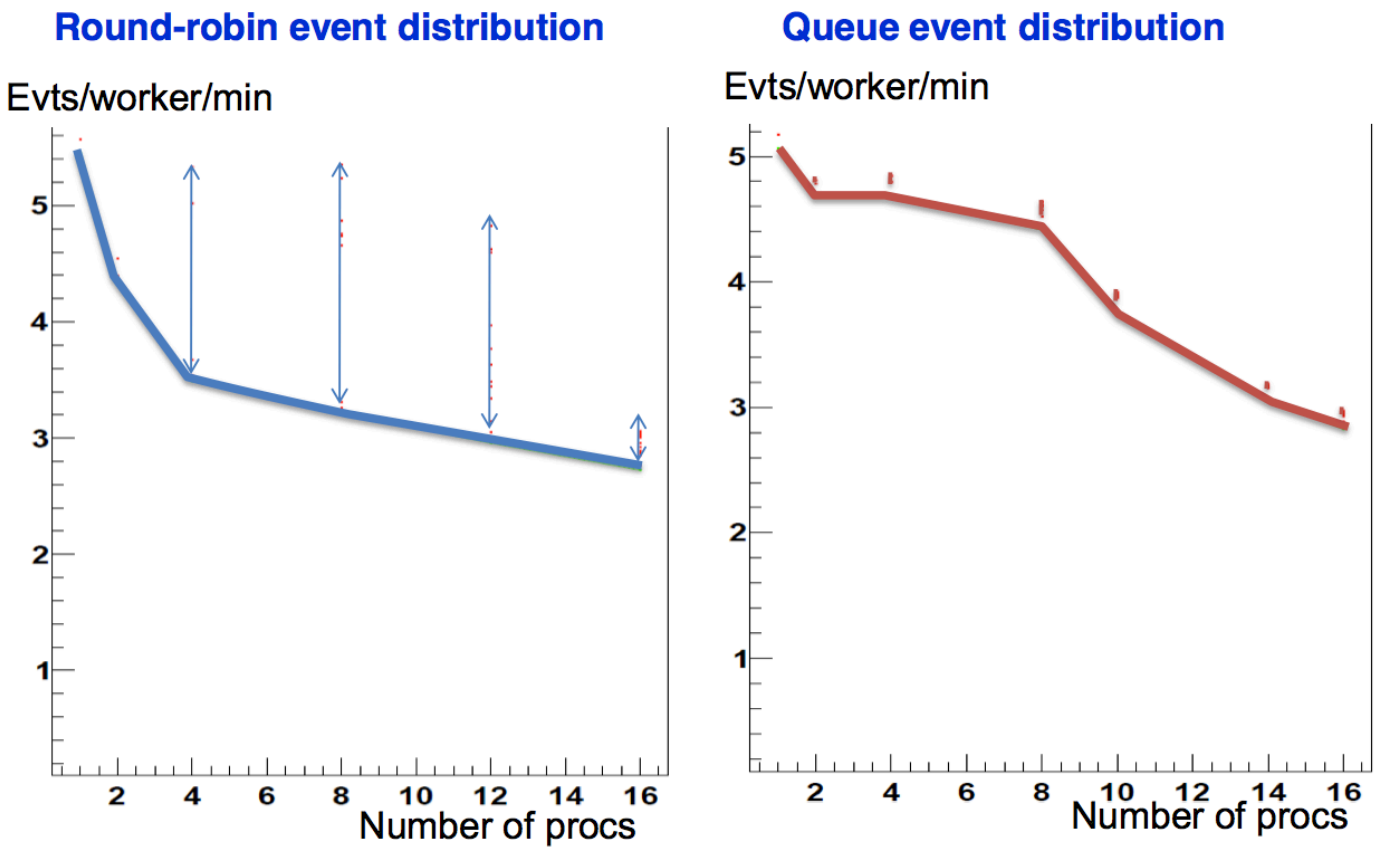
HT increases event throughput by 25%

Core Affinity



- **Affinity**: pinning each processes to a separate CPU-core.
- **Floating**: each process scheduled by OS; frequent task switching.

Multi-core Motivation



- Queue event vs Round-robin
  - Balance the arrival times of workers
  - Slower worker does not get left behind

ATLAS Whole-node Task Force

- Aim to have AthenaMP in standard production on the Grid in the next 3-6 months
- Production jobs assumed to either be single-process or whole-node
- Need to define multi-core queues and job arguments
- Participating sites need to modify batch system queues and (possibly) adjust middleware configuration
- Similar Whole-node Task Force on CMS (Peter Elmer)

Initial Phase

- Started Production tests with Athena release 16.5.0.3
- First testing queue setup at CERN (CERN_8CORE)
- Defined as whole node machines, dedicated machines scheduling only these multi-process jobs
- External sites volunteered to run multi-core jobs (see later)

PanDA Changes for Multi-Core

- 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 before job start ("ATHENA_PROC_NUMBER"). An Athena job uses this variable to set the number of processes.
- If the release doesn't support multi-process running, the job ignores the variable, still runs with one process
- Queue can then run any job, and number of processes for Athena are defined at run time
- Changes made to view to see the mutli-process logs. (Each process has a separate log in a sub-directory)

Current Testing Status

- Tests based on Athena release 16.6.6.1
  - Release incorporates a number of small bug fixes for production running.
  - Works for production jobs, and passes end of job validations. Can produce all data types (ESD, AOD, HIST, ...)
- At the moment: New batch of reconstruction tasks are trying to be redone for multi-core
- Next up: Focus on simulation tasks
- AthenaMP supports G4 Simulation, performance to be evaluated (*Manoj Kumar Jha*)

Whole-node Grid Sites (I)

OSCER (US)

- Queue name: OU_OSCER_ATLAS_MPI
- Batch system: LSF
- Reserved dedicated nodes for whole-node testing to reduce job latency
- Configured small T3 cluster (OUHEP_ITB) to run whole-node jobs with Condor (7.6.0)
- Some persistent crashing of Athena test jobs observed
- Input file placement issue at T3 cluster being resolved

RAL (UK)

- Queue name: RAL-LCG2_MCORE
- Batch system: Torque/Maui
- Provided a new wholenode queue for general use
- Dedicated 4 hosts to queue
- Some initial maui allocation issues observed

Whole-node Grid Sites (II)

ECDF (UK)
- Queue name: UKI-SCOTGRID-ECDF_8CORE
- Batch system: SGE
- ECDF is a shared cluster facility with one main job queue.
- New pseudo-queue defined for wholenode jobs.
- Similar queues for GPU and "highmem" jobs are also available

Glasgow (UK)
- Queue name: UKI-SCOTGRID-GLASGOW_MCORE
- Batch system: Torque/Maui
- Generic whole-node queue available
- Same Maui allocation setup for multi-core as RAL

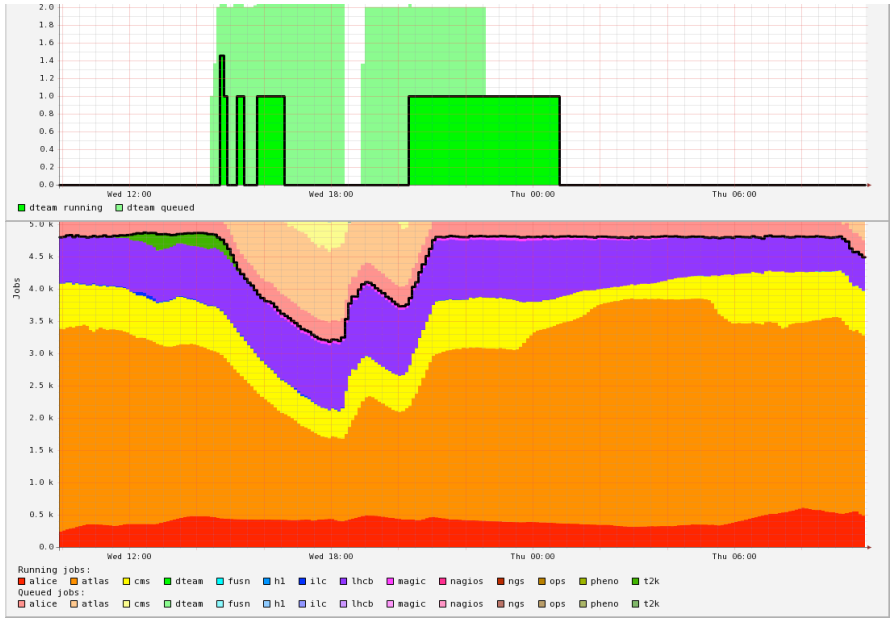
INFN-T1 (IT)
- Queue name: INFN_4CORE
- Batch system: LSF
- Dedicated multi-core queue available
- Extra queue at Napoli will be setup based on INFN-T1 work

+ interest from Imperial College (UK)

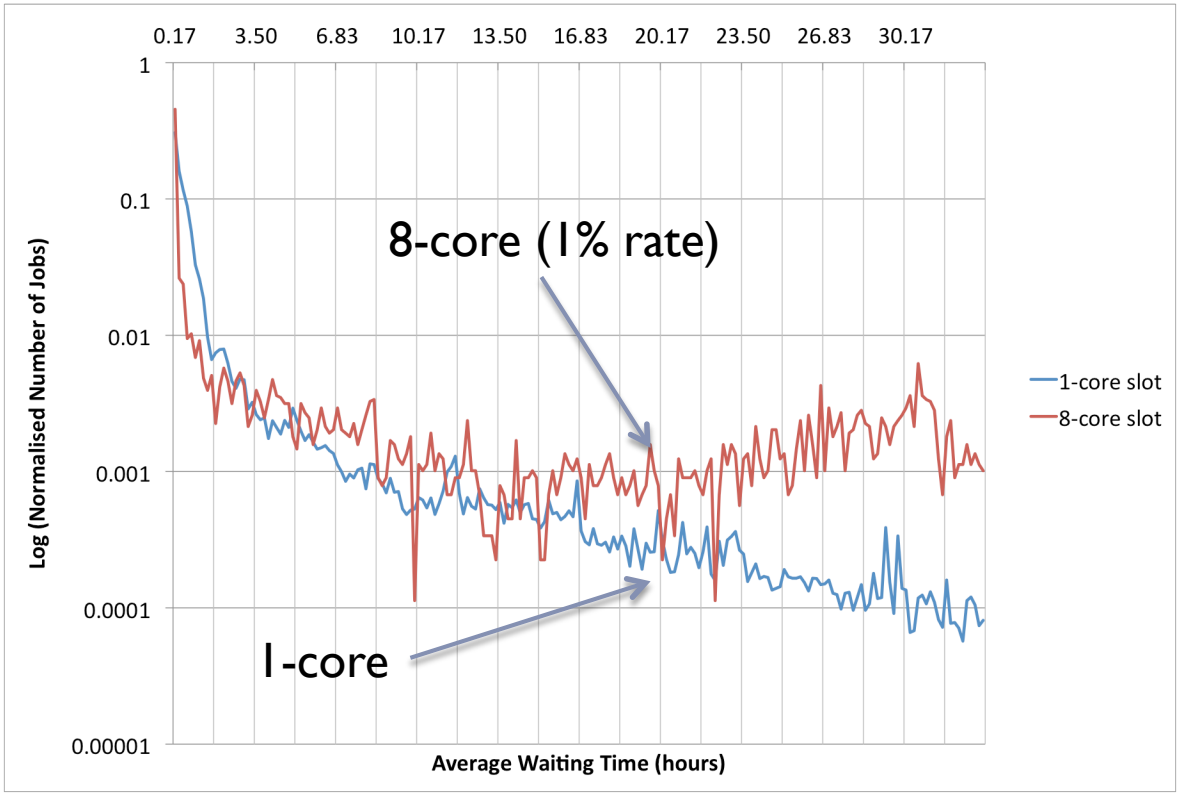
Whole-node Job Latency and Backfilling

RAL March 2011

- Lack of job lifetime estimate causes job priorities to be purely calculated from fairshares
- Fine for single-core jobs but whole-node jobs may not get scheduled immediately and block lower priority single-core slots
- Queue blocking starts to drain resources
- Single-core jobs could backfill multi-core job if job lifetime is more conservative



Whole-node Job Latency and Backfilling



ECDF jobs in May 2011 (Total 806,141)

- Study over last calendar month for all jobs suggest our 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?

Open Questions and Thoughts

Should we recommend a dedicated multicore/wholenode queue for all sites?

- Is there an idealised (and coordinated) setup strategy for LCG whole-node?
- Or do we let sites decide on the best strategy?
- A separate multi-core queue AND a separate sub-cluster?

How many cores should an AthenaMP job ask for?

- What constitutes a whole-node anyway?
- Panda implementation is fairly flexible in defining core count.

Should the core count request be adaptive to local queue conditions?

- Notoriously difficult to estimate average job wait time
- Make use of advance reservation techniques? (e.g. SGE qrsub -a 07171800 -d 8:0:0)
- Multi-core pilot coordination?

Next Steps

- Show "real" production jobs work, and on all currently defined queues
- More sites, more queues: get experience with use of whole-node and shared scheduling queues
- Production tasks setup with jobs that use 1, 2, 4, 8, 16, 24 processes
- Testing for the use of resources vs. number of processes
- Create tables of memory use, time per event, and so on, for each queue
- Define some data to be used for analysis, and validate this production