Multiprocessing in Athena

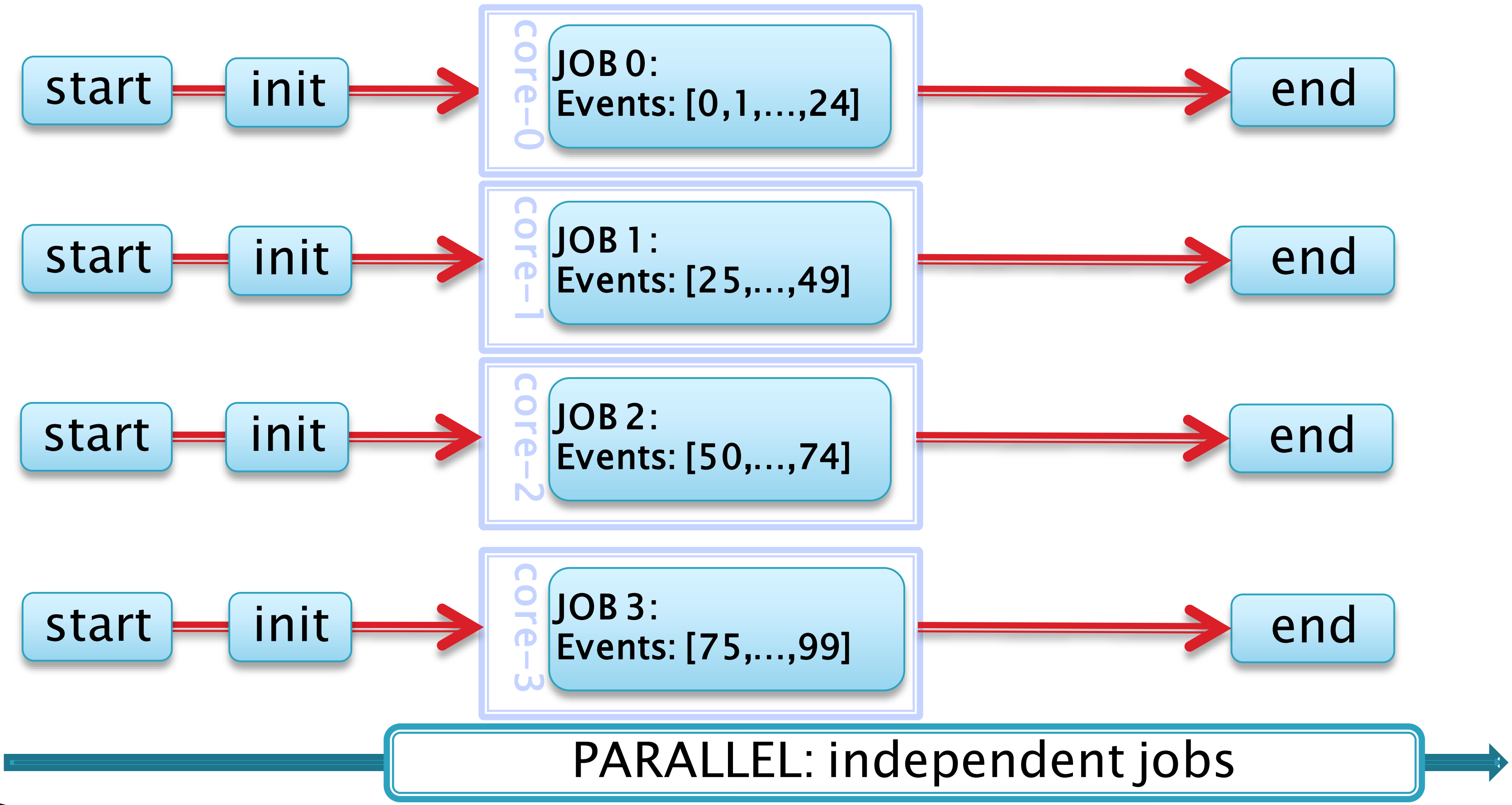
I. Performance study of Athena event and job level parallelism on multi-core systems.

II. Performance optimizations in AthenaMP.

Mous Tatarkhanov
ATLAS Core Software, Lawrence Berkeley National Laboratory

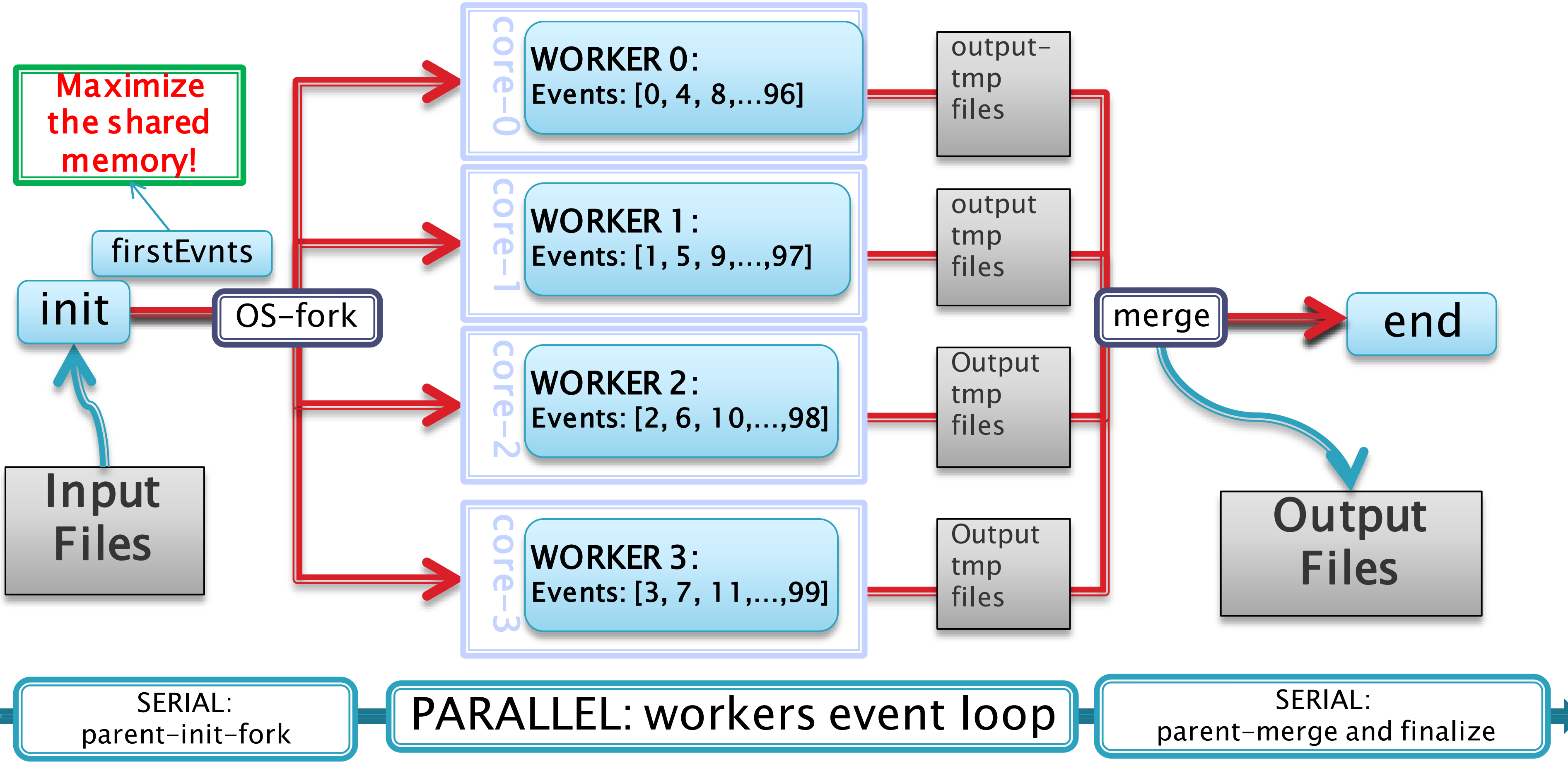
Athena multi jobs
Athena MJ – job level parallelism

```
for i in range(4):
   $> Athena.py -c "EvtMax=25; SkipEvents=$i*25" Jobo.py
```



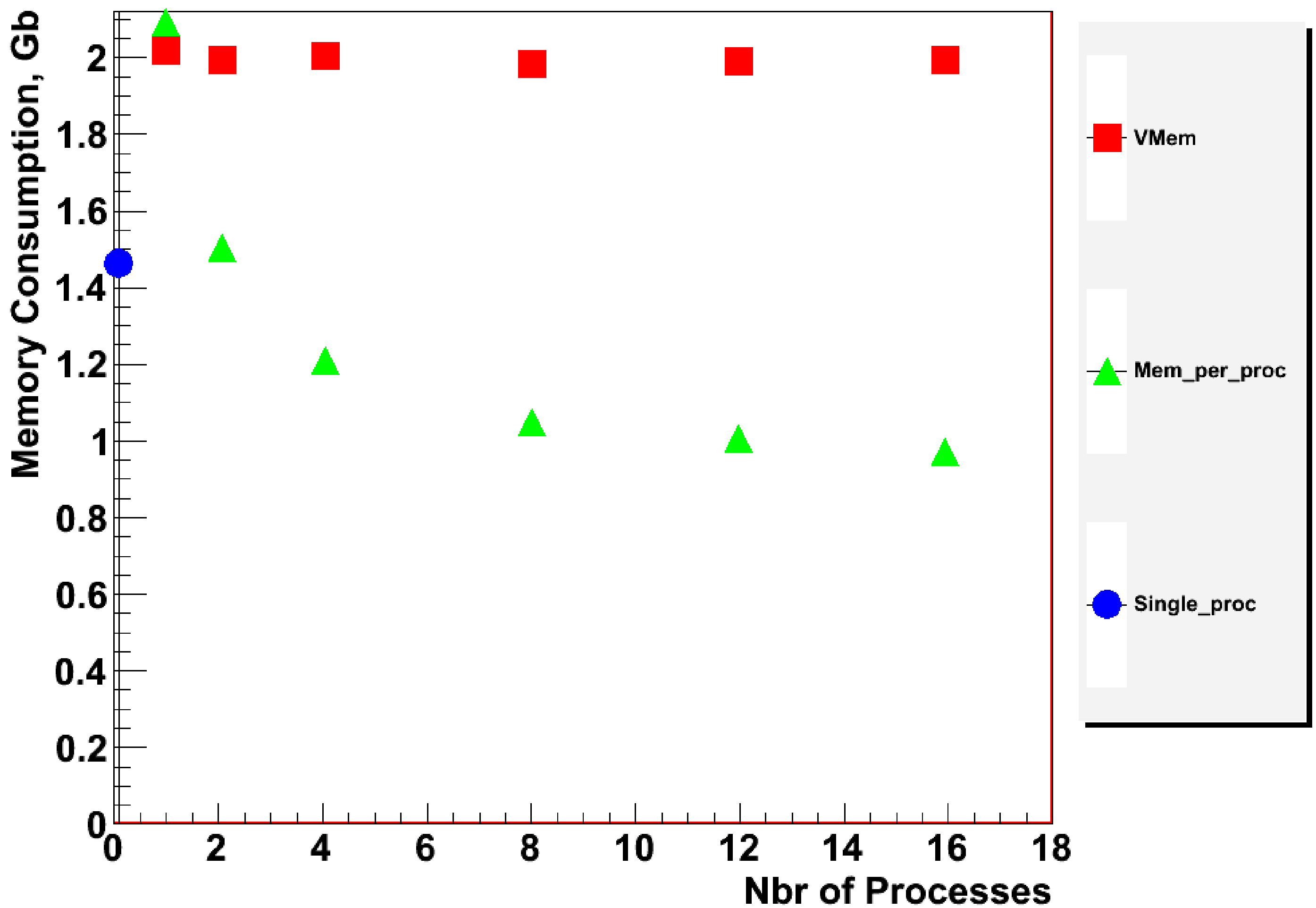
AthenaMP – event level parallelism

$> Athena.py --nprocs=4 -c EvtMax=100 Jobo.py



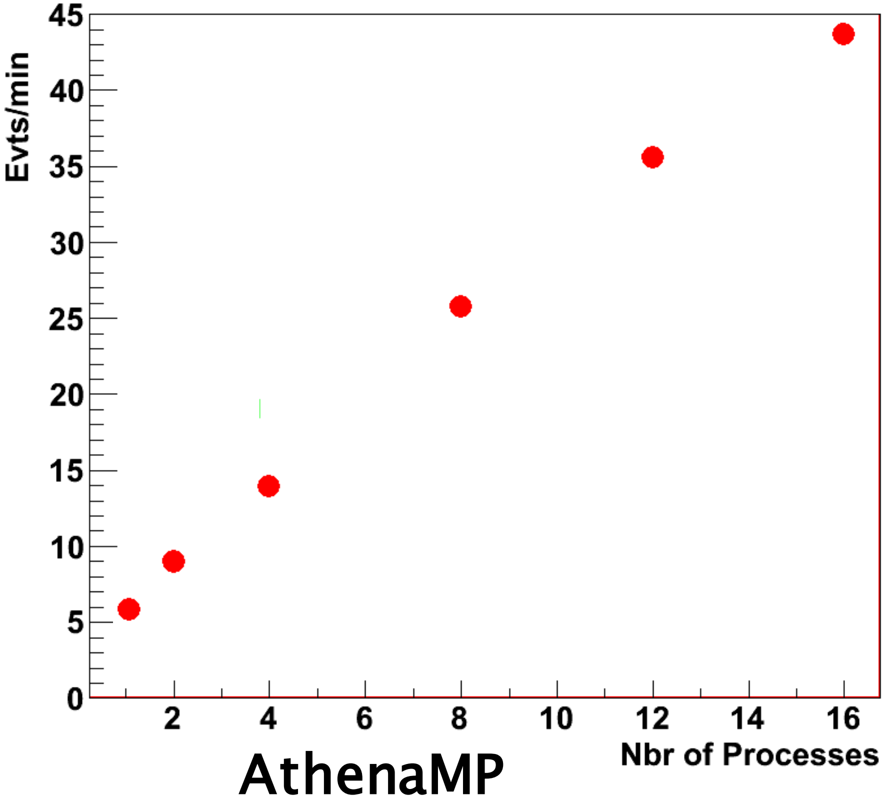
AthenaMP Status by S.Binet – *http://indico.cern.ch/getFile.py/access?contribId=2&resId=0&materialId=slides&confId=92059*

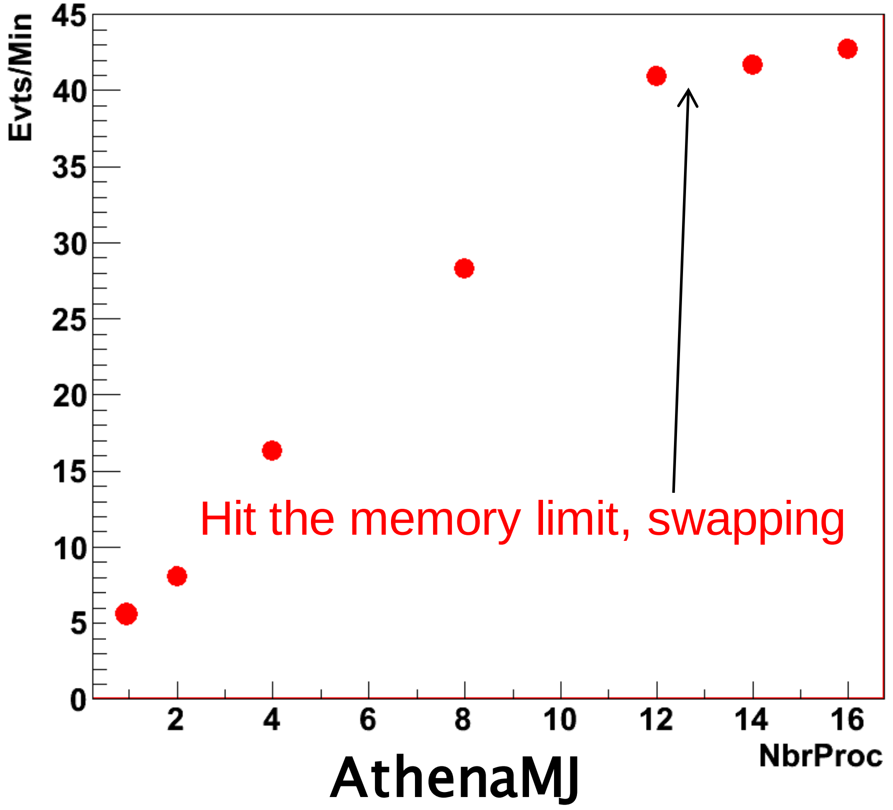
Memory footprint of AthenaMP & AthenaMJ



AthenaMP ~0.5 Gb physical memory saved per process

Event throughput of AthenaMP and AthenaMJ





Performance Boost and Optimizations

1. External Optimizations:

(no touching complex Athena code)

- Hardware Optimizations: HT, QPI, NUMA, Affinity
- OS optimizations: affinity, numactl, io-related, disks, virtual machines, etc.
- Compiler, Malloc, etc.

2. Gains from AthenaMP/Athena design improvements:

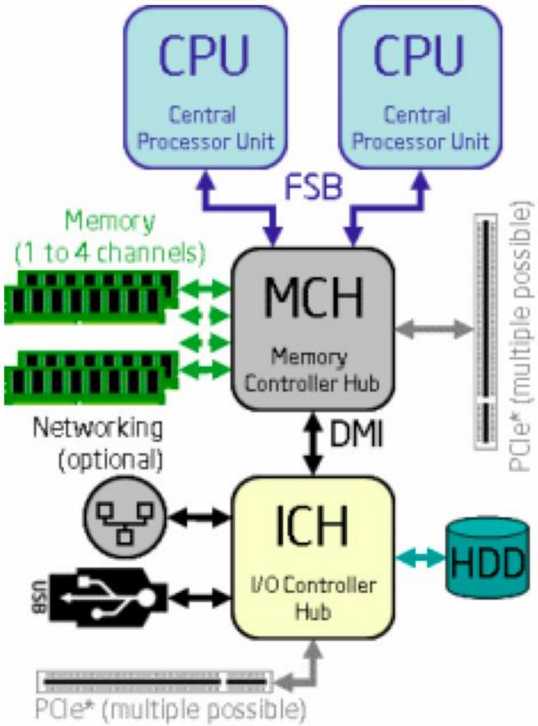
- Shared memory, forking later after init
- Queue event distribution

endless ground for improvements :)

Architecture upgrades

Intel sub-Nehalem

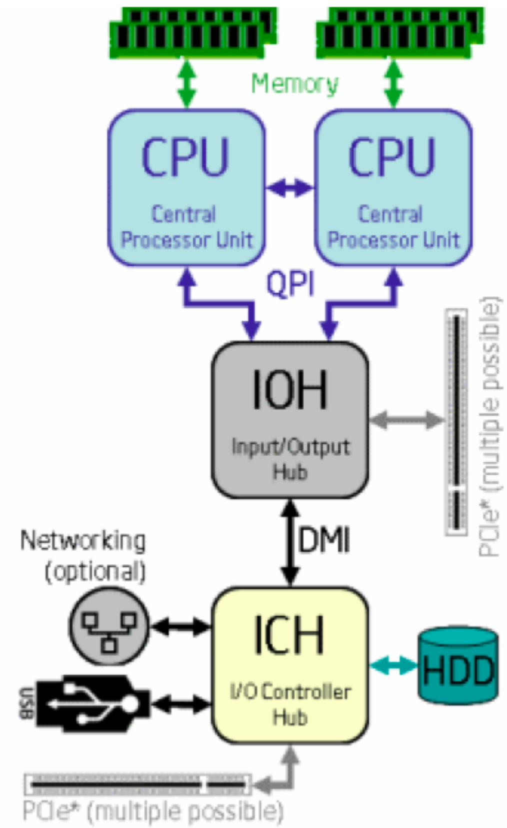
most of LXPLUS machines: Voatlas91,lxplus250,lxplus251



CPU-Memory symmetric access

Intel Nehalem

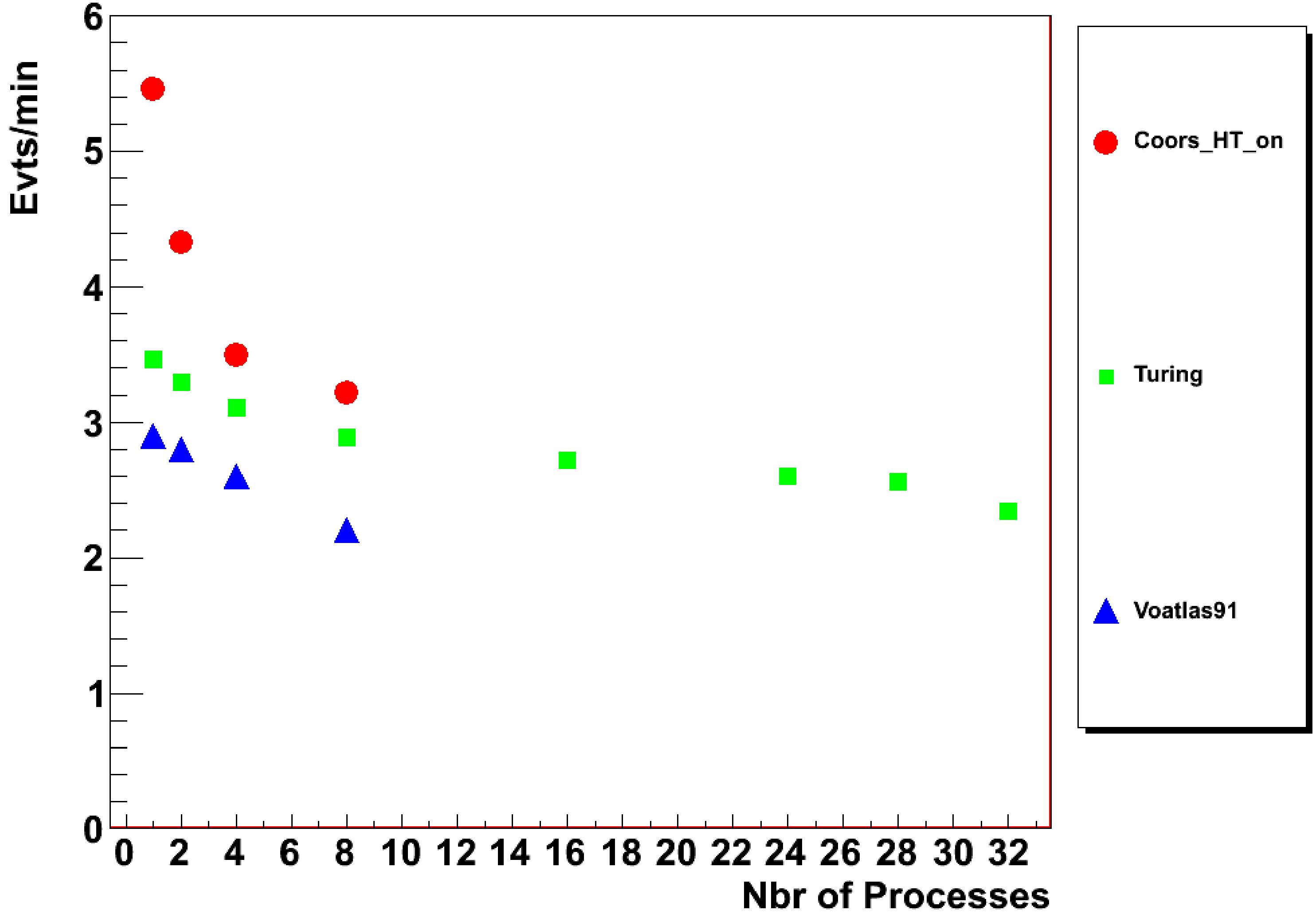
coors.lbl.gov, rainier.lbl.gov



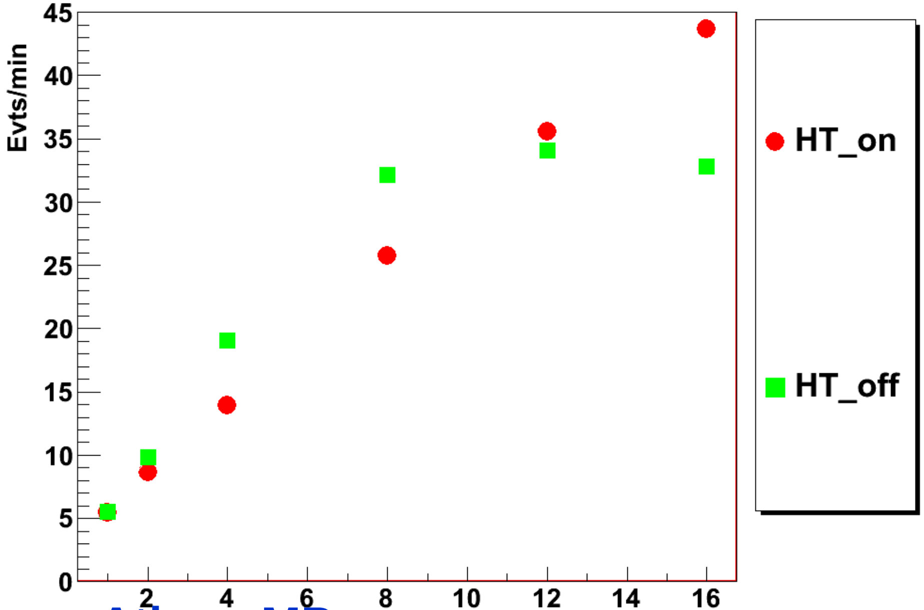
- Hyper Threading ->two logical cores on physical one
- QPI Quick Path from CPU to CPU and CPU-to-Memory
- Turbo Boost -> dynamic change of CPU-frequency

- CPU-Memory non-symmetric access (NUMA)

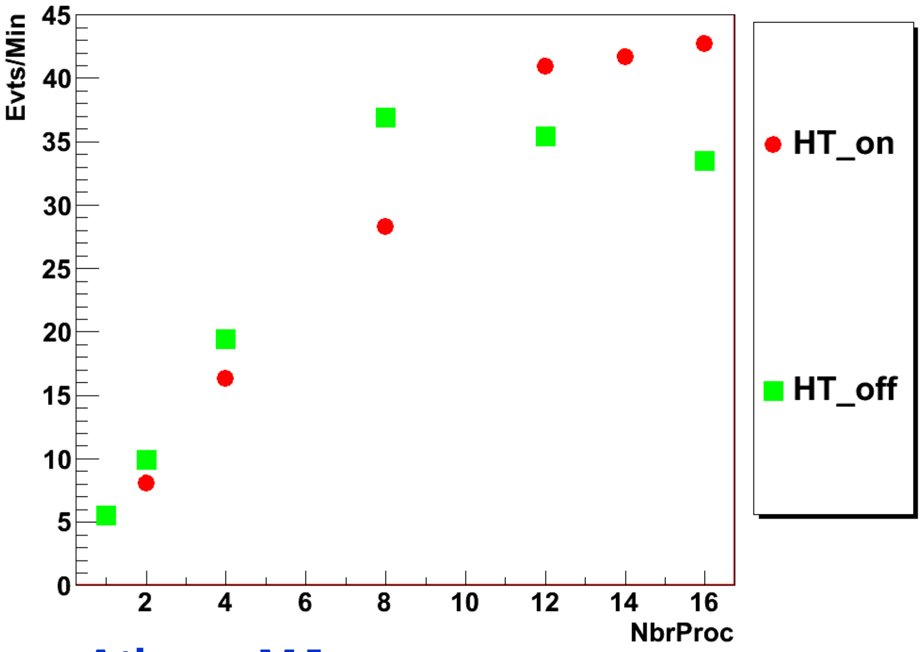
Event Throughput per process for RDO to ESD reco on different machines



Gain from Hyper-Threading



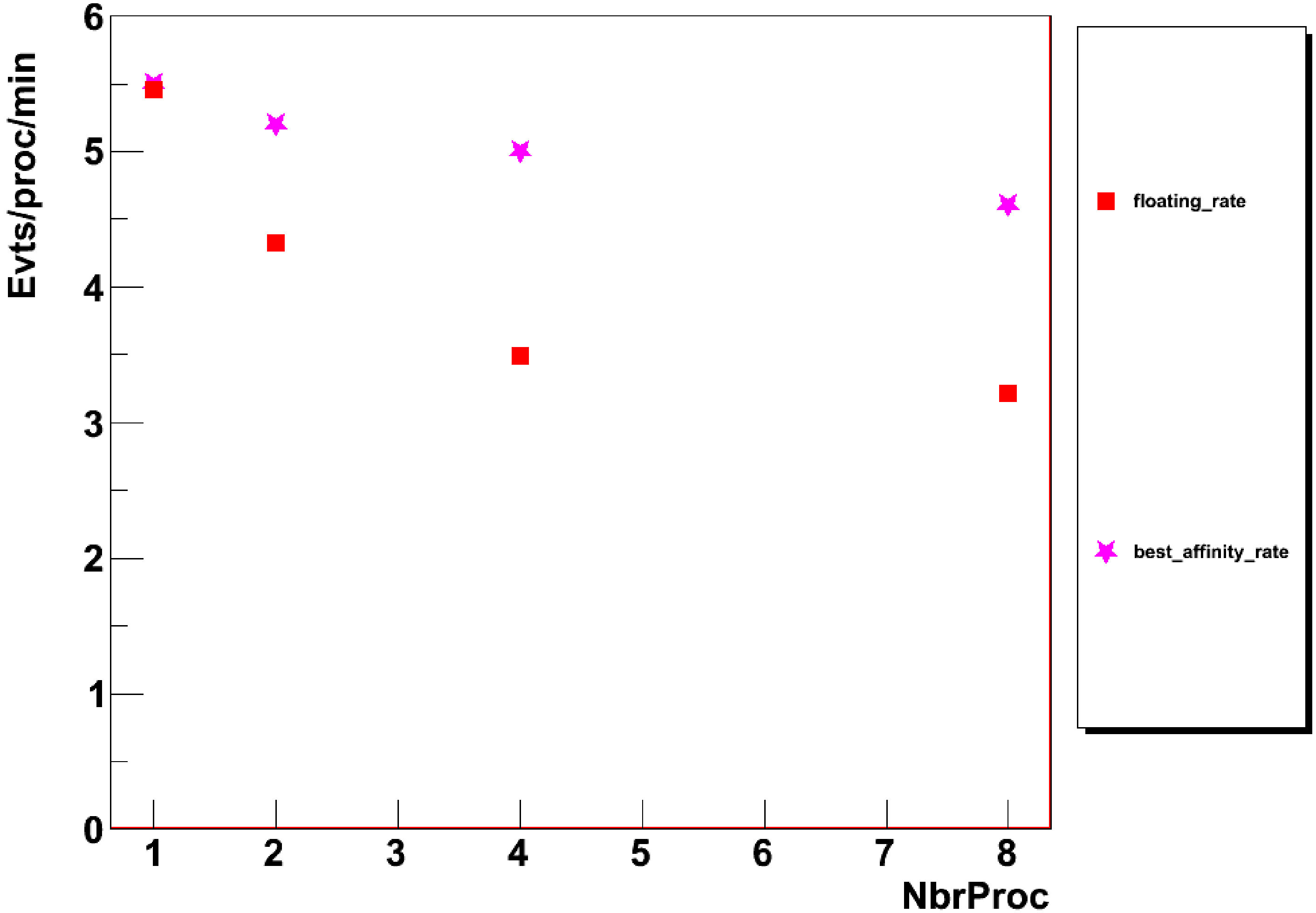
AthenaMP



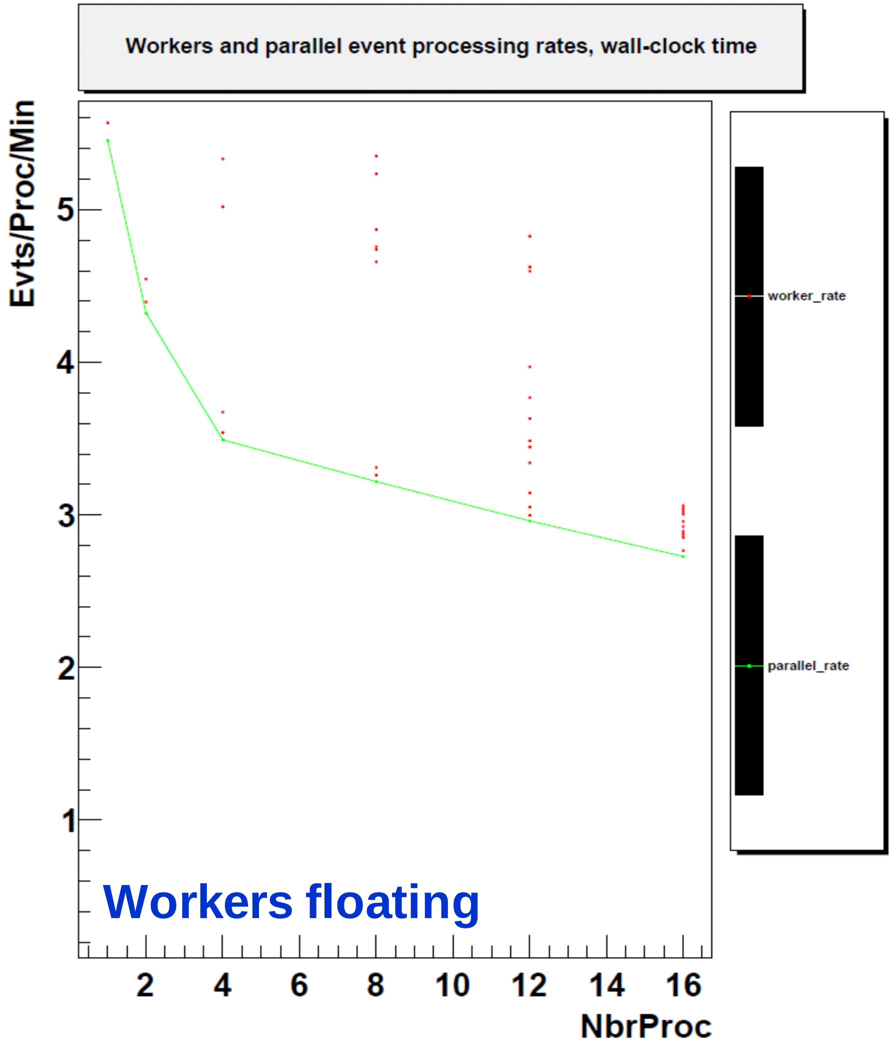
Athena MJ

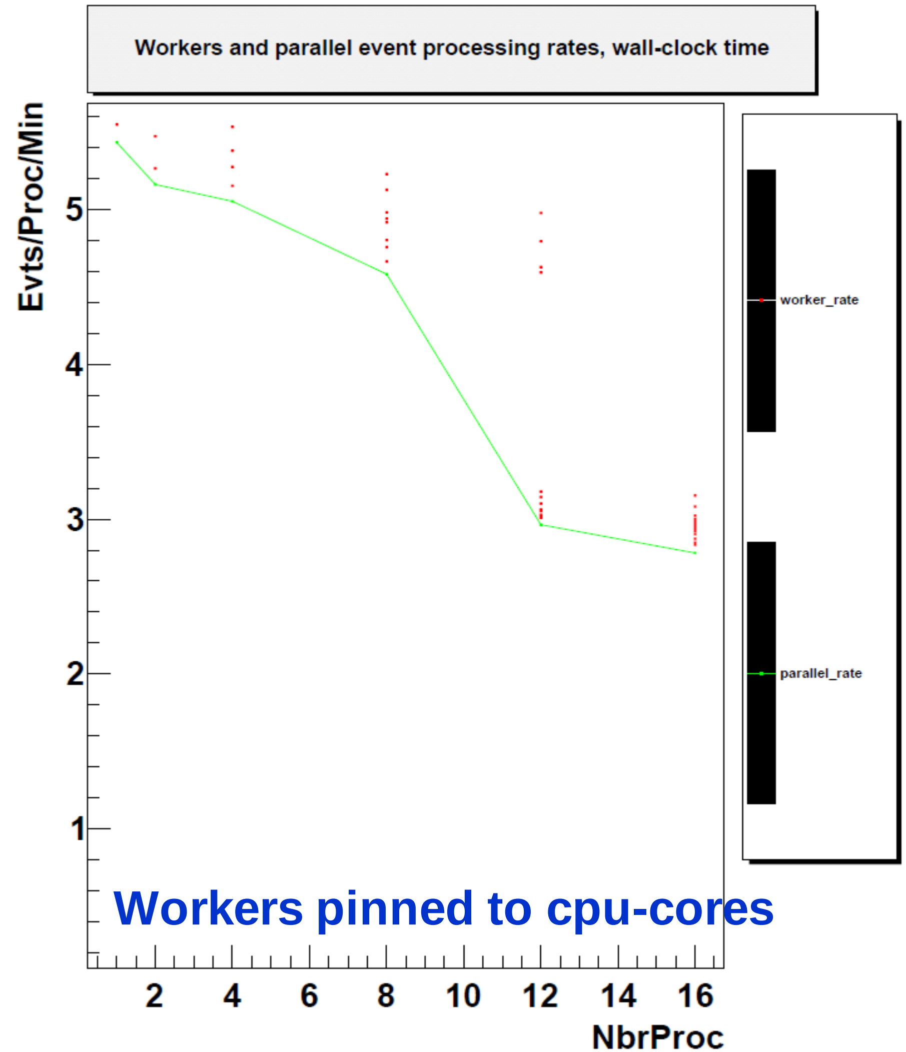
Setting affinity of workers to cpu-cores

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



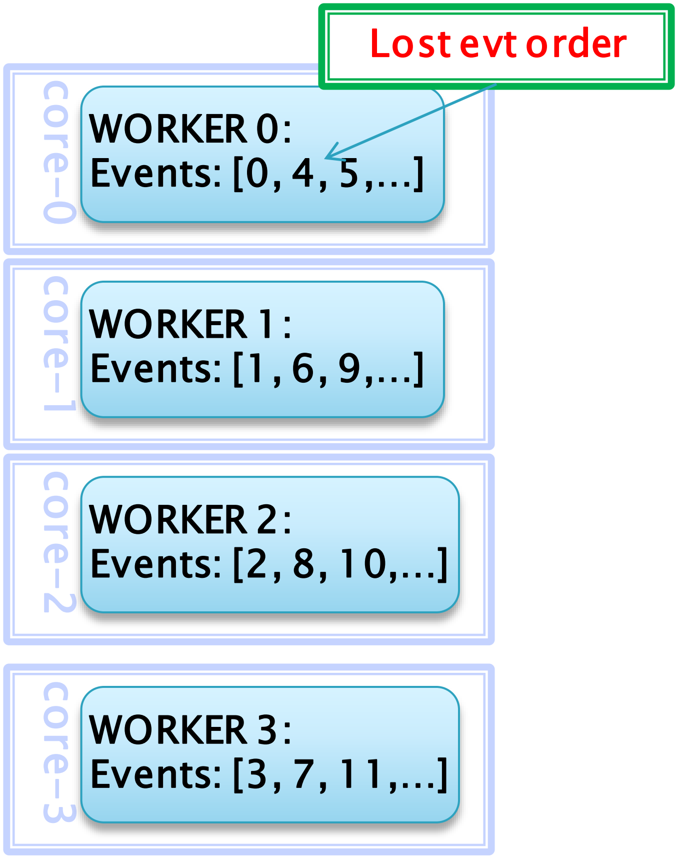
Event workers throughput





Recent Progress: Event distribution using Queue…

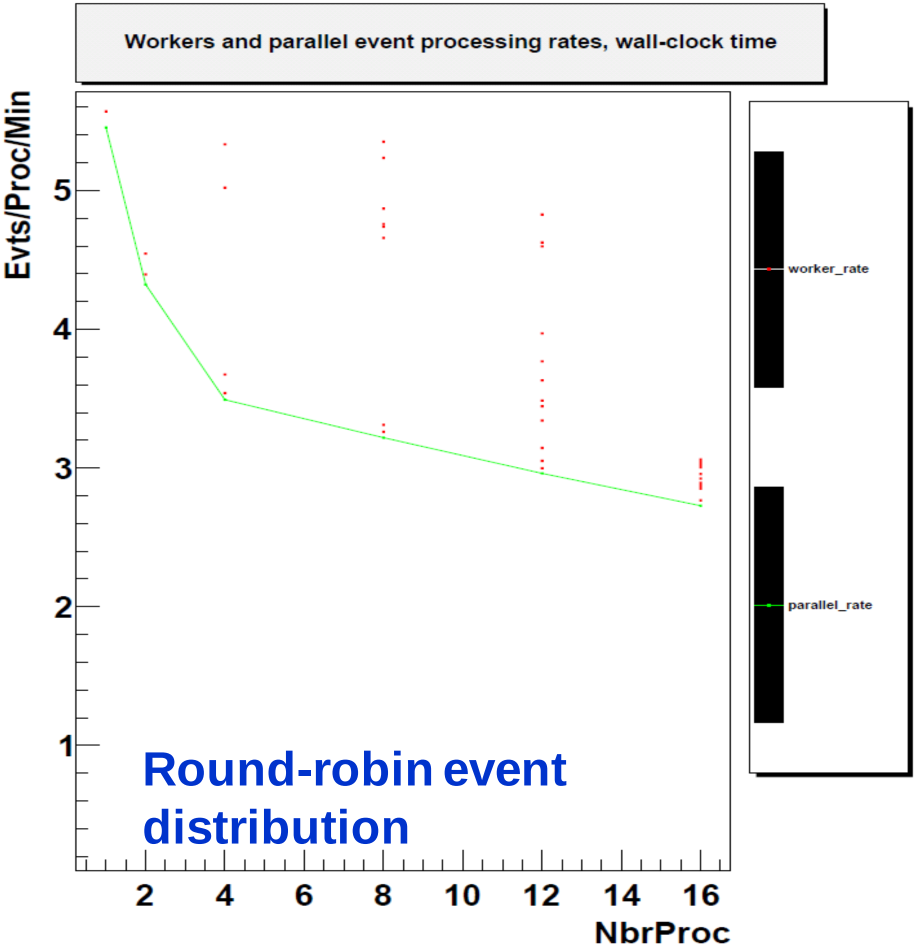
```
events = multiprocesssing.queue(EvtMax+ncpus)
events = [0,1,2,3,4,…,99, None,None,None,None]

…

evt_loop(evt=events.get(); evt != None):
     evt_loop_mgr.seek (evt_nbr)
     evt_loop_mgr.nextEvent ()
```

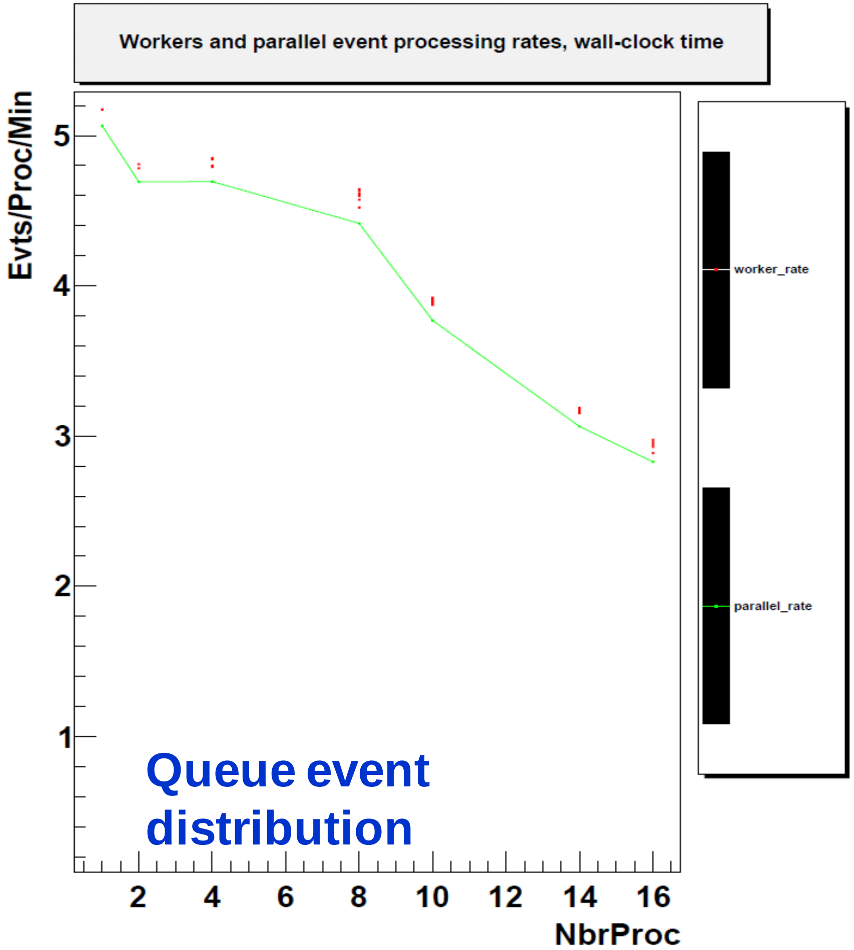


Balance the arrival times of workers!

Slower worker doesn't get left behind

Workers throughput for Queue





Conclusions

- AthenaMP shares memory about ~0.5 Gb of real memory footprint per worker.
- Queue balances workers arrival times thus improving mp-scaling.
- Hyper-Threading can give 25-30% gain on events throughput
- Affinity settings exploit CPUs better than linux cpu scheduling.
- NUMA effects take place on Nehalem CPUs.

Maximizing AthenaMP performance

1. Externally available performance gains (without touching the athena code)

- Architectural gains: HyperThreading, QPI, NUMA etc.
- OS gains: affinity, numactl, io-related, disks, virtual machines, etc.
- Compiler, Malloc, etc.

2. Gains from Athena/AthenaMP design improvements:

- Faster initialization…
- Faster distribution of events to workers...
- Faster merging: merging events processed by workers instantly by one writer on a fly, without waiting for workers to finish…
- Faster finalization…

endless ground for improvements :)

BERKELEY LAB

Acknowledgments

- Paolo Calafiura, Sebastien Binet, Yushu Yao, Charles Leggett, Wim Lavrijsen

- Keith Jackson, David Levinthal

- Ian Hinchliffe and LBL ATLAS Group

- LBNL and DOE for Funding
- CERN for Research