memory usage of athena and athenaMP-2

Rolf Seuster (TRIUMF)
ATLAS Software Co-Coordinator
Facilities Jamboree
3. December 2014

TOC

- will summarize studies done over the last 3 years, various linux versions (SLC5/6), various athena reconstruction jobs, various athena releases, various (empty) workstations
  - basic behaviour of athena should not change but study should be redone to account for modern HW
  - all of them aim at performance measurements in relation to memory, some athena some other athenaMP
  - Note: they are not ordered in time
- tool to get real memory consumption of athenaMP jobs

Problems of OS tools with AthenaMP

- by OS tools I mean: ps, top, etc.
- memory sharing of processes in OS usually for libraries, not commonly used for data
  - therefore sharing is low, usually few percent
- most tools report what they reported the last 30 years: memory consumption of this process alone, not taking sharing into account, worked well in the past and today, until athenaMP comes along ...

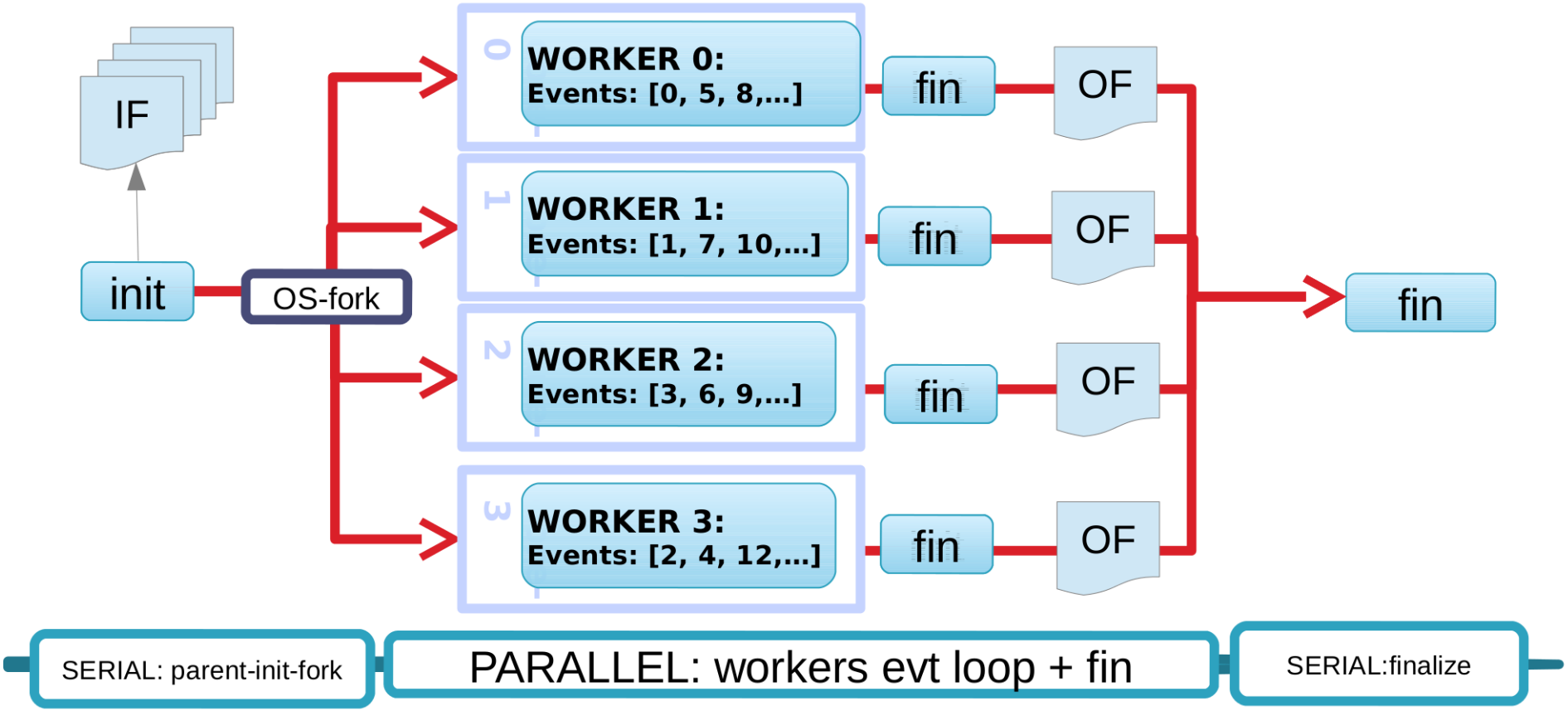
athenaMP

- ATLAS relies with athenaMP on COW to save huge amounts of memory, now also including data: geometry, conditions, etc
  - COW: copy-on-write, forked processes have own memory space, but linux kernel keeps them together until one process writes to a page (usually 4kB), this process than works with its own, private copy of this page, other processes continue to use initial, shared page
- with modern MMUs (memory management units) this becomes almost trivial to implement

AthenaMP in 5 seconds

Athena MP - Multi-Process

https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/AthenaMP



First Study,
own “smaps” + athenaMP

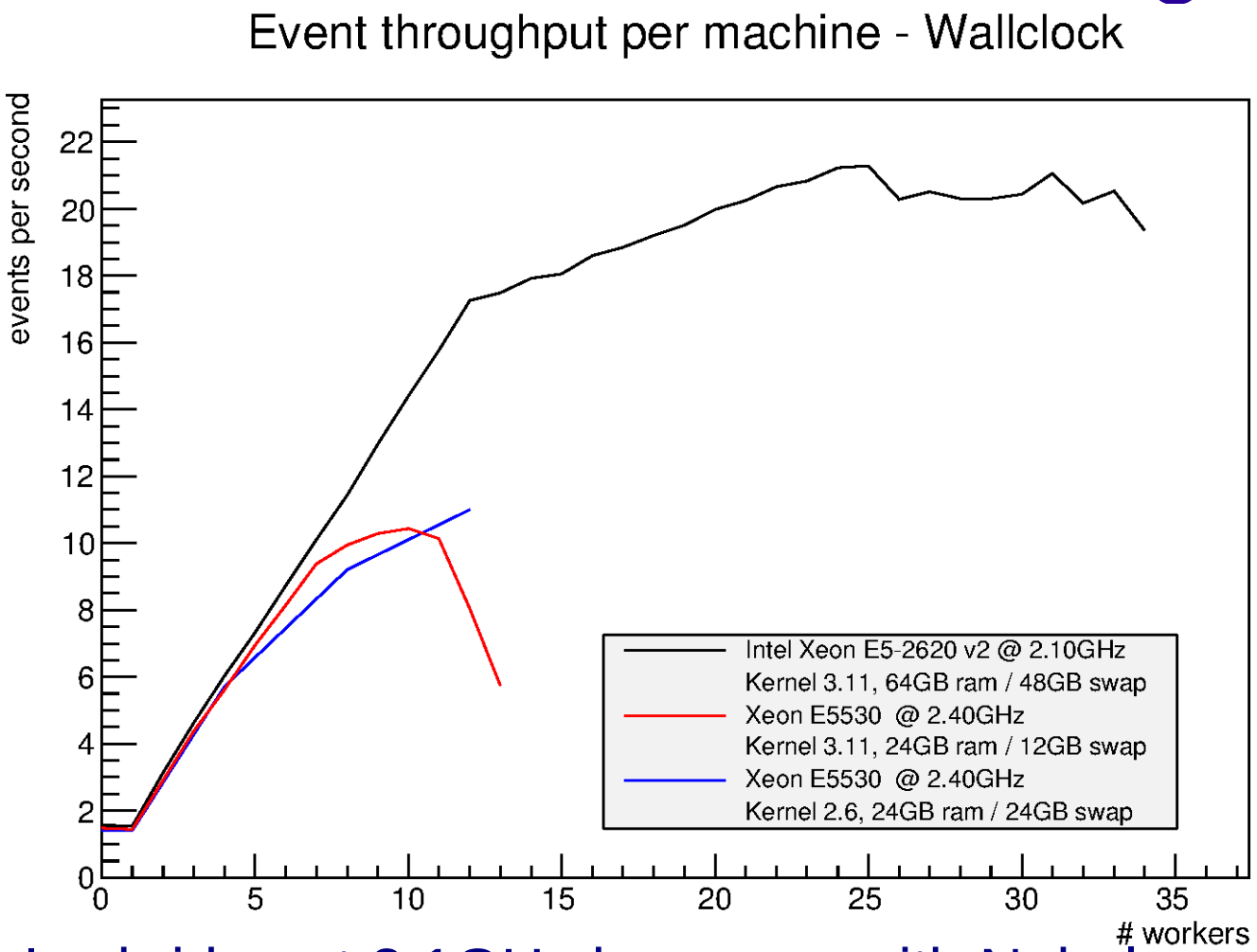
Introduction

- various athenaMP jobs in recent release (18.9.50), input: monte carlo with fixed mu of 40
- in parallel to athena ran
  - program 'free' – however not carefully checked yet
    - Vakho, main developer of athenaMP uses often this
  - own program to estimate shared/private/swap of athenaMP mother and its children (details in backup)
    - tries to get similar info as SMAPS (perl) but from c++. It looks only at specific list of PIDs
      - much improved version now exists
  - postprocessing scripts to make plots

TestBed

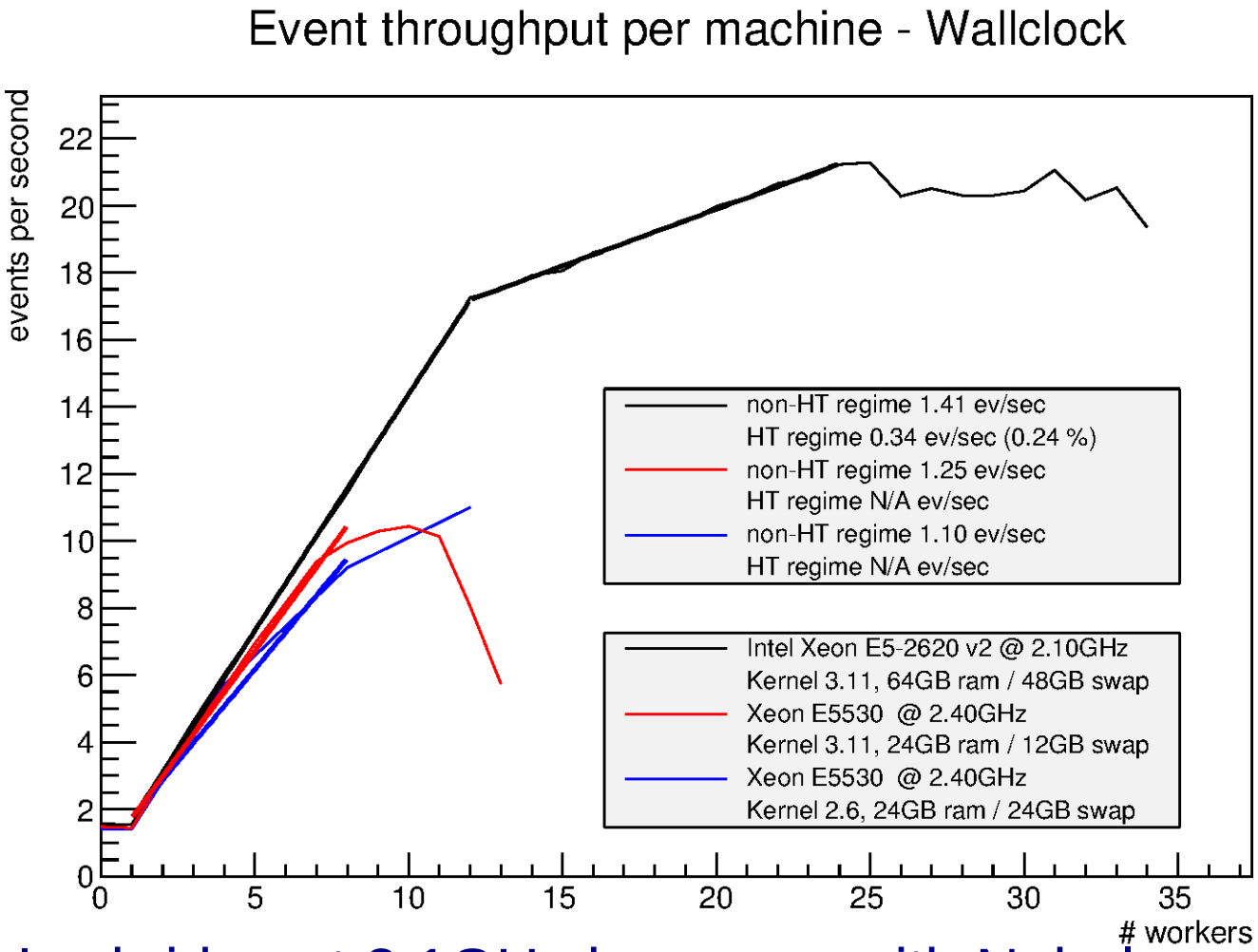
- jobs ran on 3 different machines
  - ATMIC001: 12 core Intel Xeon E5-2620 v2 @ 2.10GHz; with HT 24 core; 64GB memory in total
  - two older machines PCMPP001/2
    - same HW, both SLC6,
      - Nehalem E5530, 2.4GHz, 24GB ram
      - 001 with kernel 2.6, 24GB swap, 12 GB in /tmp
      - 002 with kernel 3.11 from openlab, 12GB swap
- input files stored in /tmp, fixed number of events per worker: N*100 for N workers
- ran with different NPROCs: ATMIC001 0-23 and 39, PCMPP001 0,1,4,8,12 and PCMPP002 0-12
  - “0” means single process athena, “1” means athenaMP with one mother and one worker

AthenaMP-2 – event throughput



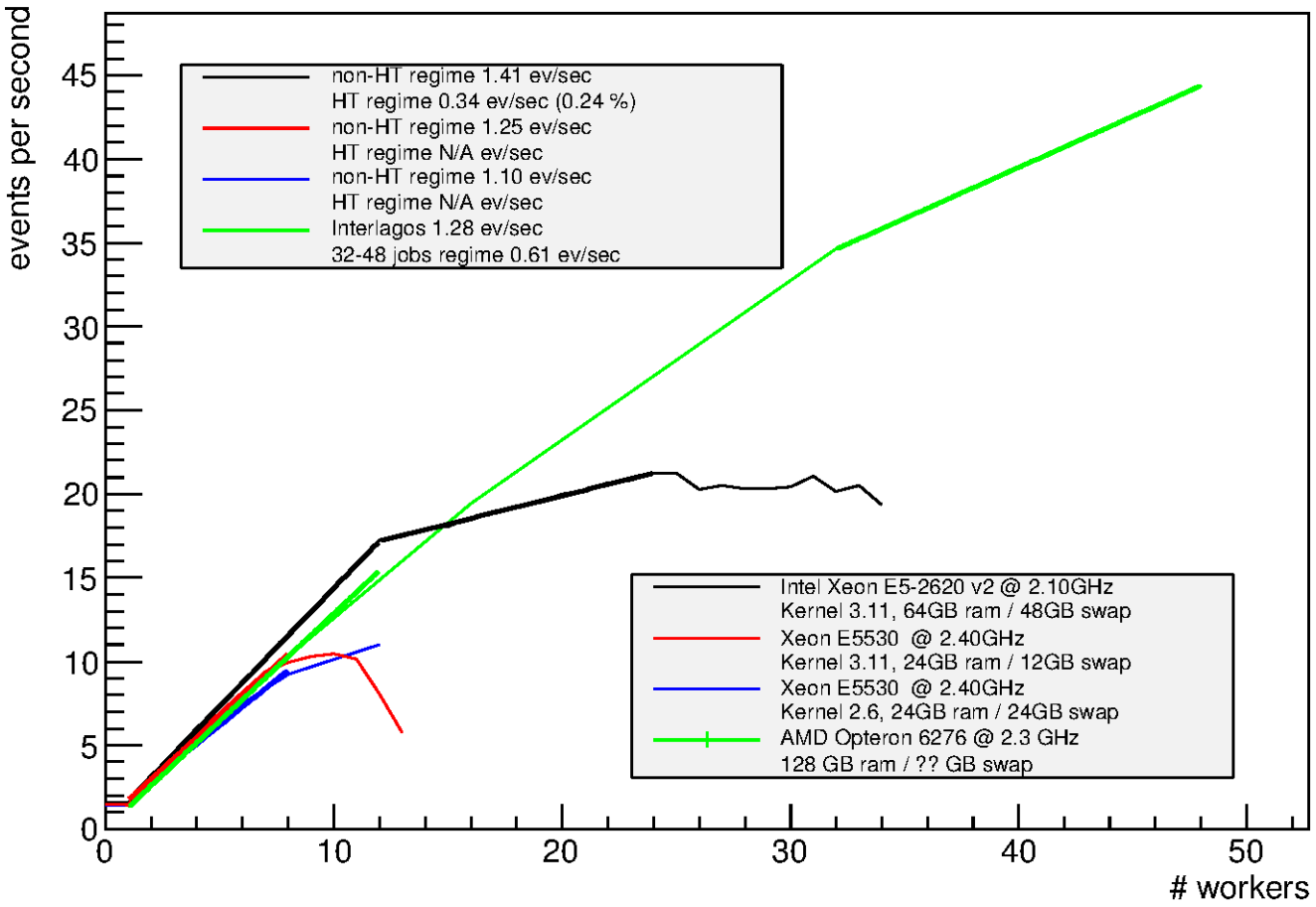
- new Ivy bridge at 2.1GHz keeps up with Nehalem at 2.4GHz
- swap space pays off – or: too little hurts you !

AthenaMP-2 – event throughput



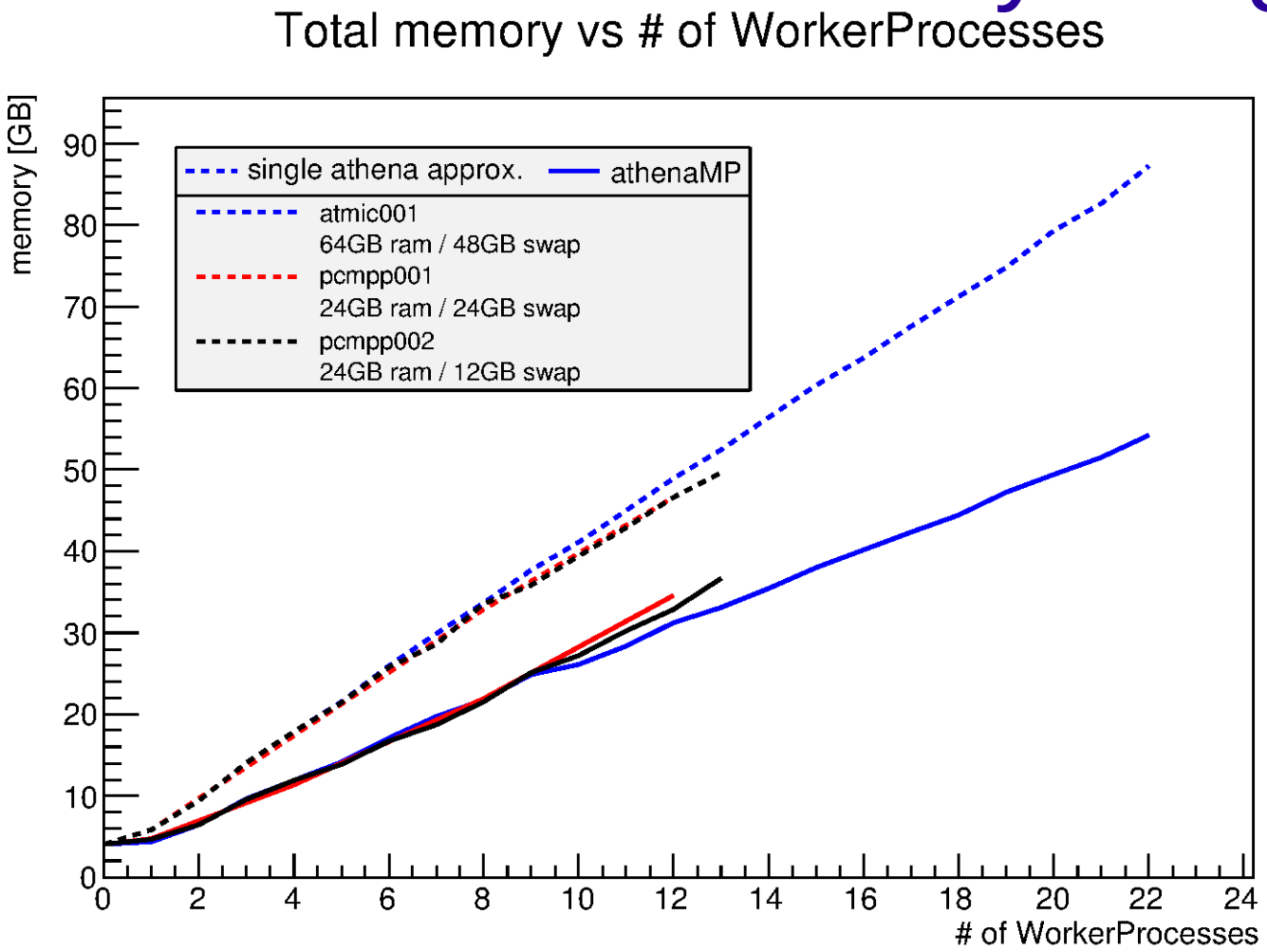
- new Ivy bridge at 2.1GHz keeps up with Nehalem at 2.4GHz
- newer kernel **might** perform better by few % but not really conclusive – reminder : additional swap was in /tmp (SSD)

Interlagos



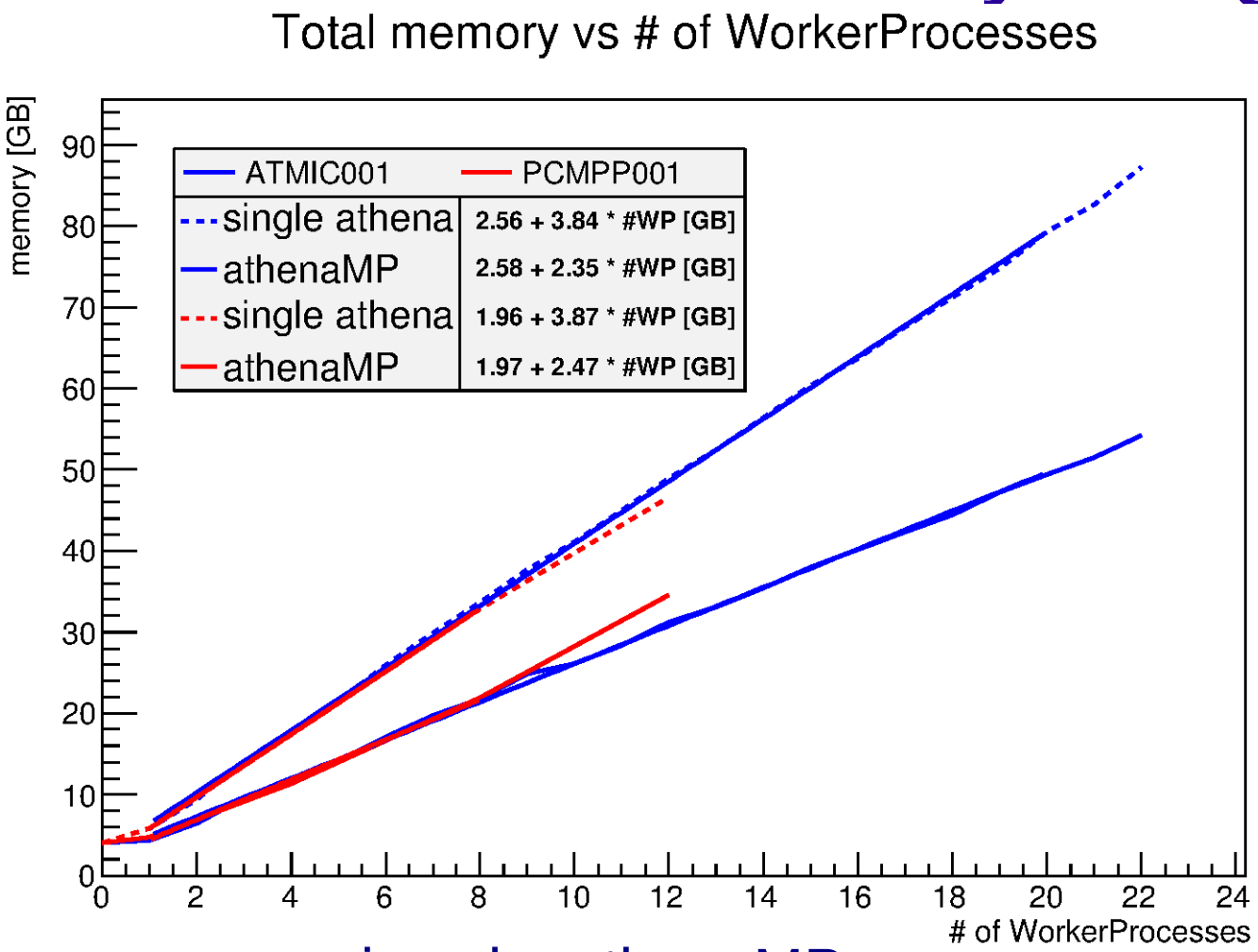
- thanks to Graeme can compare to AMD box with 64 cores
- overall performance well, but significant slowdown for jobs on 32-48 cores w.r.t. 1-12 cores – slowdown visible even before

AthenaMP-2 – memory usage



- clear memory savings by athenaMP by roughtly 30%
- still at 2.4-2.5 GB per core for MC at <mu>=40 !

AthenaMP-2 – memory usage

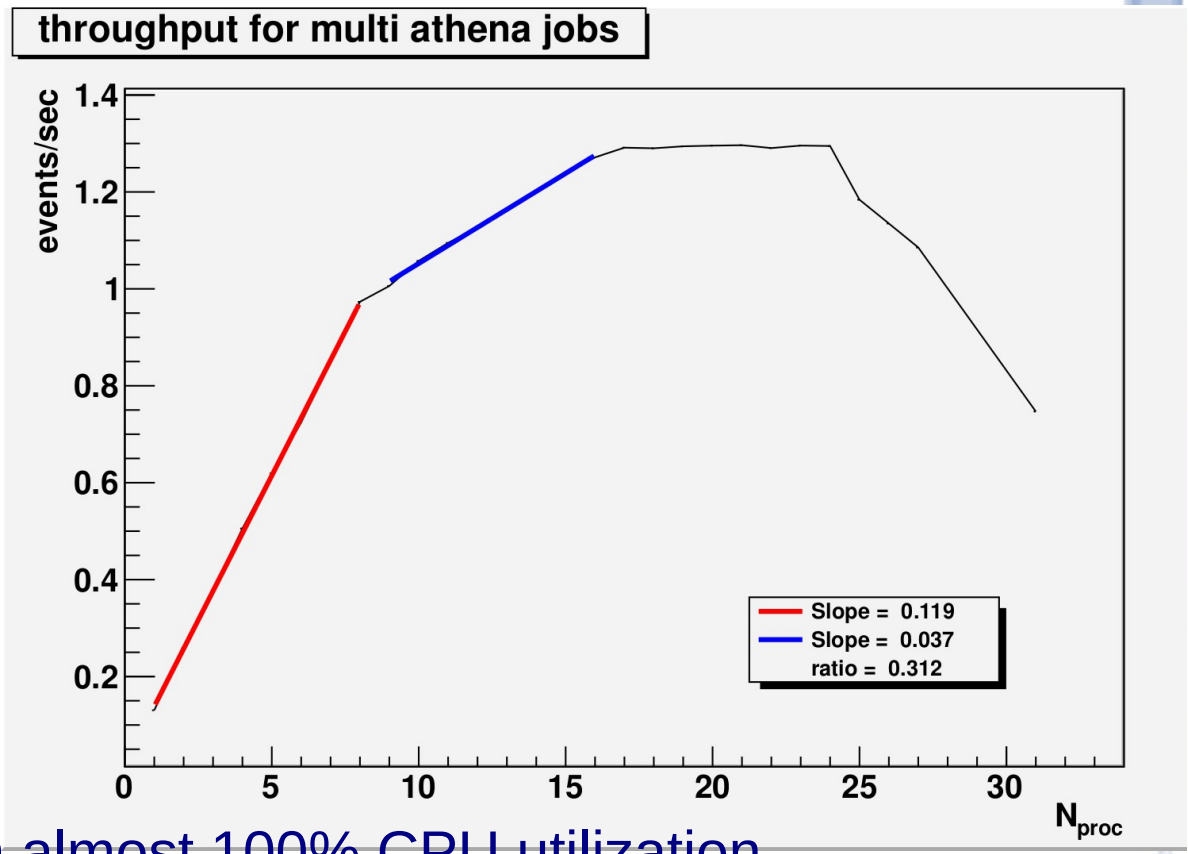


- clear memory savings by athenaMP
  single athena approximation doesn't work well for low number of workers : always mother process with ~2GB, can also be seen in offset in above fits

Second Study,
overloading with single process athena

Overloading a machine

- Test machine:
  - 8 core with hyperthreading
  - 24 GB of memory
  - RAW to ESD low pileup data from 2011
  - study from ~ 3/2012



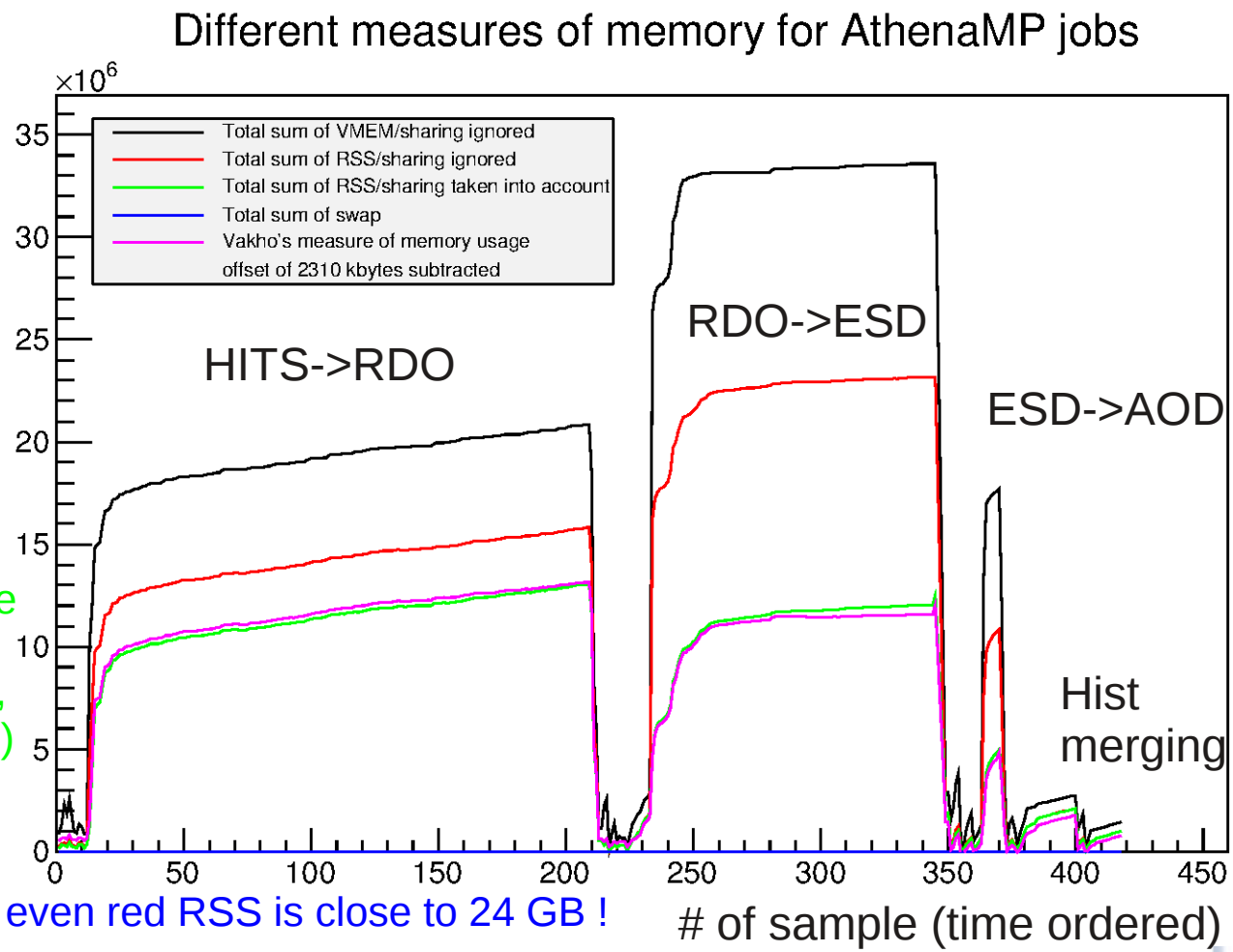
- up to 24 single process athena, i.e. non-athenaMP jobs can run in parallel with almost 100% CPU utilization,
  → 1GB per core real memory, note – very little sharing as these are single process athena jobs !
  → non-active code swapped to disk
  with more than 24 jobs machine start swapping out active code and event throughput on that machine goes down overall

Third Study,
“pss” + athenaMP

PSS actually means ”rss with sharing”

Memory usage in athenaMP in DC14 – random DC14 job

- DigiMReco with 8 workers incl. Pileup (mu = ?)
- Vmem: virtual memory, → irrelevant (some memory profilers use >16 TB of Vmem)
- RSS, sharing ignored: this is approximately how much memory N single athena jobs would use …
- RSS, accounting for sharing: the ratio of the red and the green is the gain we get from athenaMP, THIS IS USING smaps (backup)
- amount of swapping in this job; in this job no swapping, machine has 24GB of real ram, even red RSS is close to 24 GB !



- Vakho's measure of memory usage utilising 'free' to determine used, total memory of workernode – depends on what else goes on on machine so needs dedicated single user machine, might work for whole node VMs

getSharedMemory

- recent releases have executable “getSharedMemory”
  - run as 'getSharedMemory <pid>' with <pid> PID of the mother athenaMP job
- returns 4 numbers:
  - PSS (=RSS, sharing taken into account)
  - swap out pages to disk (recoverable pages not accounted)
  - RSS (=RSS,no sharing)
  - VMem (virtual memory usage, for comparisons only ! – vastly wrong estimate !)
- suggestion: pilot should run this at specific time intervals to collect memory profile, max value, etc.
- plan is to have same code used inside athena to report memory usage via the logfiles

New Brainpower

- Natalie Rauschmayr (CERN-IT) is now starting to look at ATLAS' memory usage
  - her background: she did extensive studies in LHCb and is probably the most experienced person at CERN about memory usage (except her supervisor, maybe!)
  - I picked (randomly) one out of many topics she studied, this is IMHO interesting, not studied in ATLAS (recently at least)
  - she will be looking at the other experiments as well: ATLAS can learn also from the others!

Natalie's Study

Non intrusive Optimization

- Memory Deduplication
- Memory Compression
- x32-ABI
- Auto-Vectorization
- TCMalloc, jemalloc ...

... and there's yet another block of studies about intrusive optimizations e.g. via GaudiMP in LHCb

What is it ?

Memory deduplication

- Detect equal memory pages and share them
- Transparent usage
- SLC 6: enable KSM thread
- Run jobs with malloc hook

run time behaviour

Memory deduplication

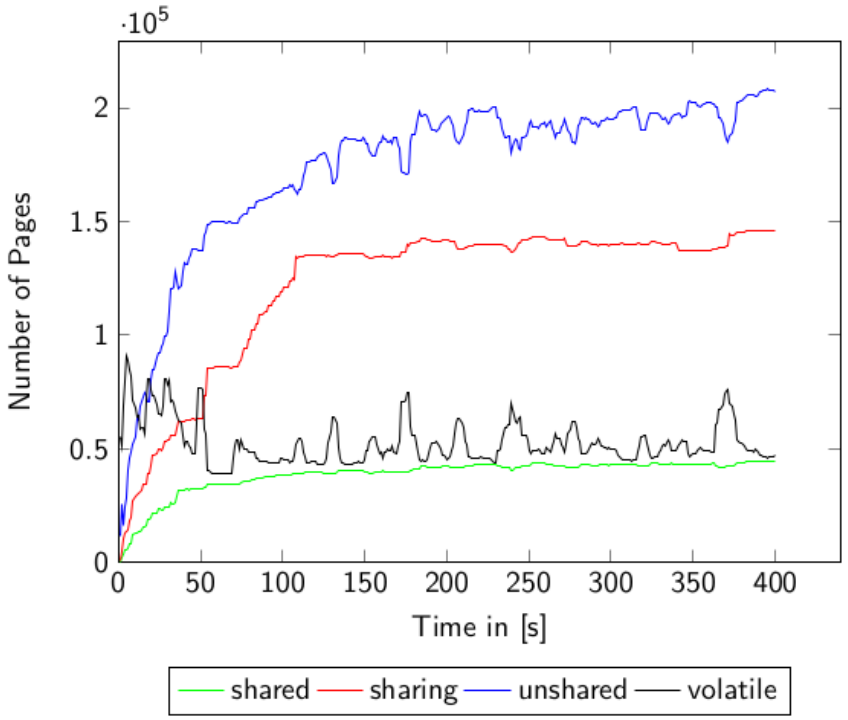


Figure: Results for simulation job running with 2 workers

Results

Memory deduplication

	serial mode	2 workers	8 workers
Simulation	183 MB (22 %)	623 MB (33 %)	2659 MB (48 %)
Reconstruction	100 MB (8 %)	448 MB (21%)	2297 MB (33 %)
Stripping	165 MB (13 %)	890 MB (26 %)	3864 MB (32 %)

Table: Memory reduction reached by memory deduplication in the different applications

Summary

- athenaMP does save lots of memory, and reporting by various OS tools can be vastly wrong !

Memory reported by	VMem	RSS, sharing ignored	RSS, with sharing	Vakho's free
HITS to RDO	19.8GB	15.1GB	12.4GB	12.5GB
RDO to ESD	32.0GB	22.1GB	11.9GB	11.5GB
ESD to AOD	16.9GB	10.3GB	4.7GB	4.6GB

- Depending on type of job, 2-3x overshot from e.g. VMem w.r.t. PSS (=RSS, with sharing taking into account)
- Further studies to come, thanks to Natalie joining !
- About athenaMP, see also Vakho's and Sami's talks in Core session of parallel Software WS happening in Curie