CHEP 2016

22nd International Conference on Computing in High Energy and Nuclear Physics, Hosted by SLAC and LBNL, Fall 2016

Monitoring of Computing Resource Use of Active Software Releases in ATLAS

Antonio Limosani

On behalf of the ATLAS collaboration

ATLAS

THE UNIVERSITY OF SYDNEY

CERN

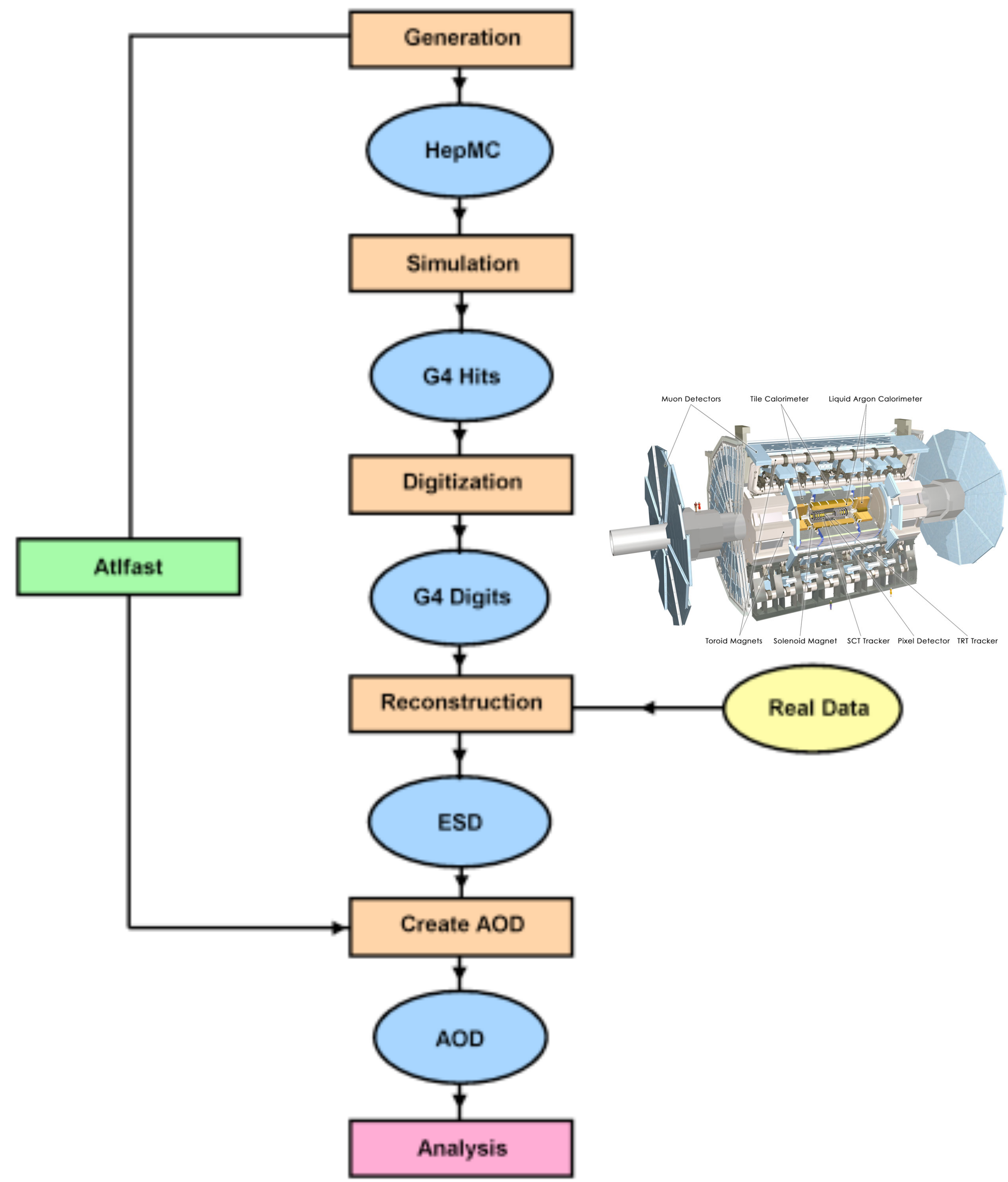
CoEPP

ARC Centre of Excellence for Particle Physics at the Terascale

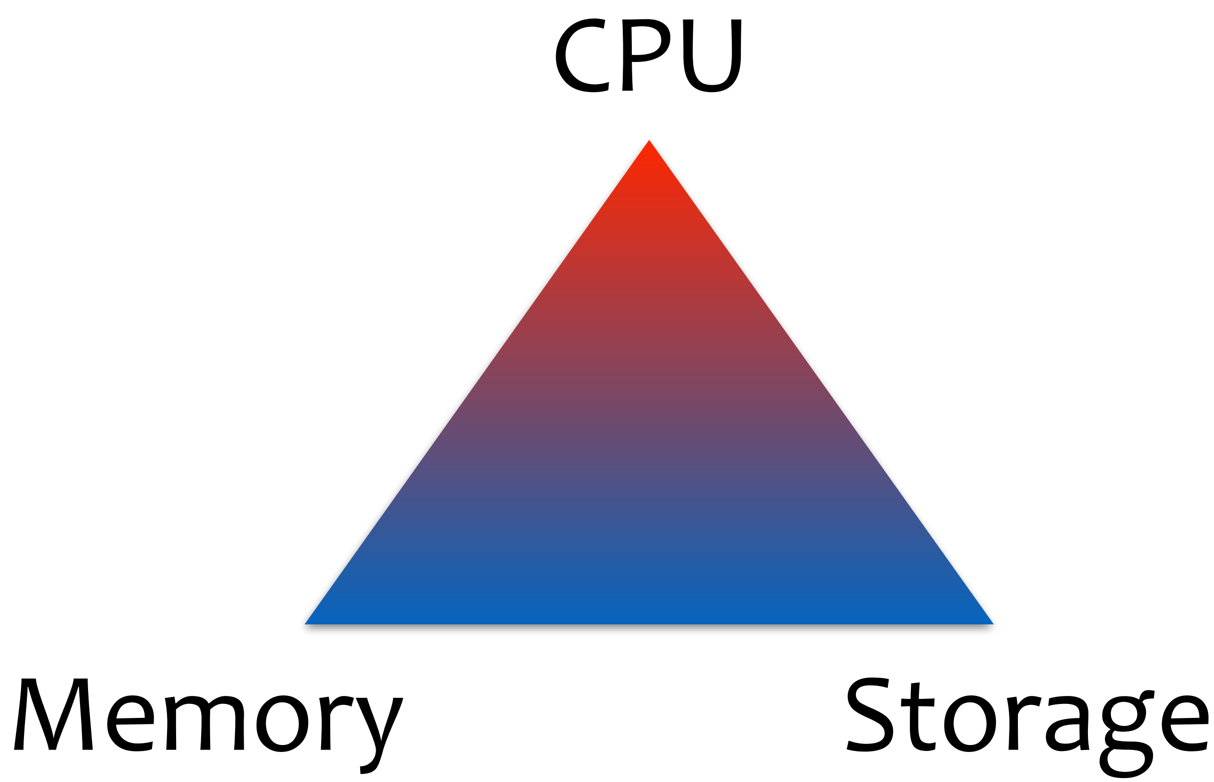
ATLAS Workflows

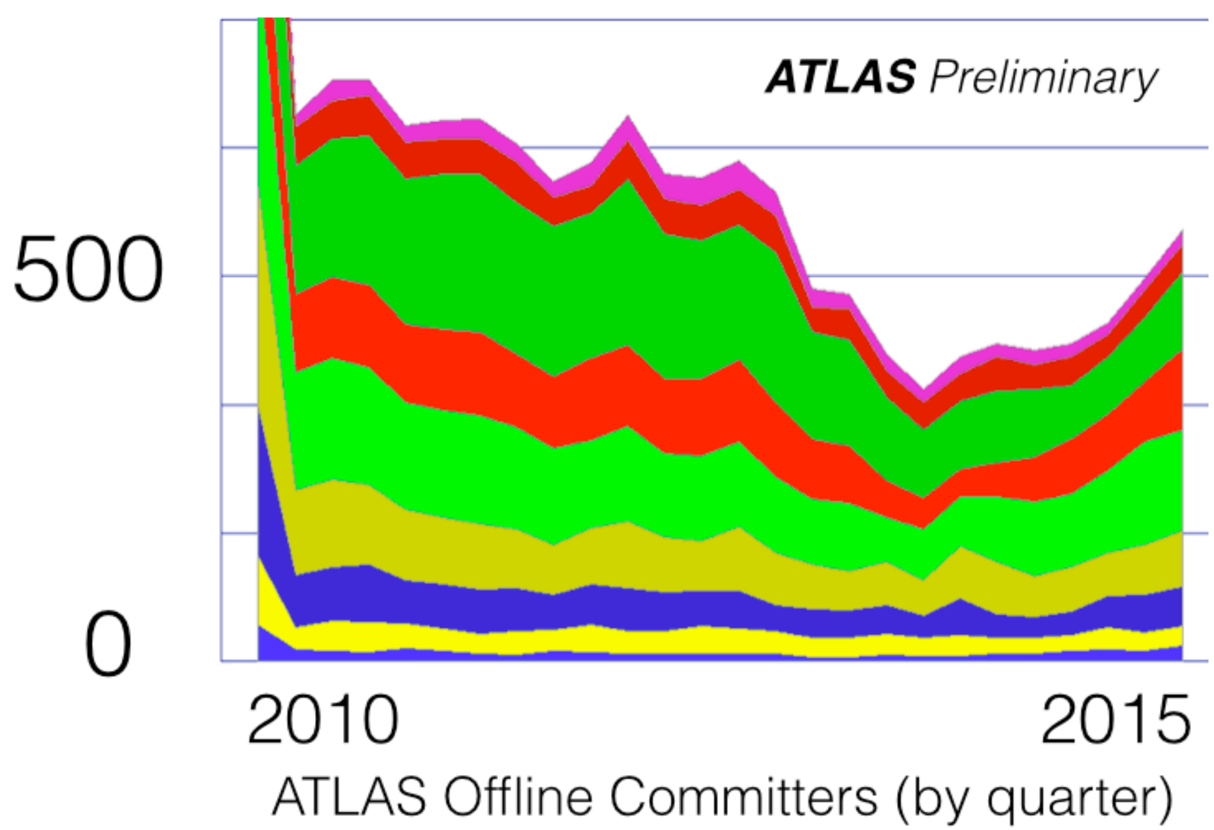
- ATLAS is a diverse and globally distributed software project
- Millions of lines of codes distributed across more than 2500 packages built into shared libraries. Almost one package per every ATLAS author
- Diverse workflows to satisfy the goals of research
- Workflows deployed on heterogeneous computing systems
  - Tier0/1/2/3, WLCG, Cloud, HPC, laptops
- Perform these tasks optimally (cheaply) as possible
  - Achieve maximum throughput of useful "physics events"
- Precision physics requires billions of events

Life of a particle collision at the LHC



Resource monitoring





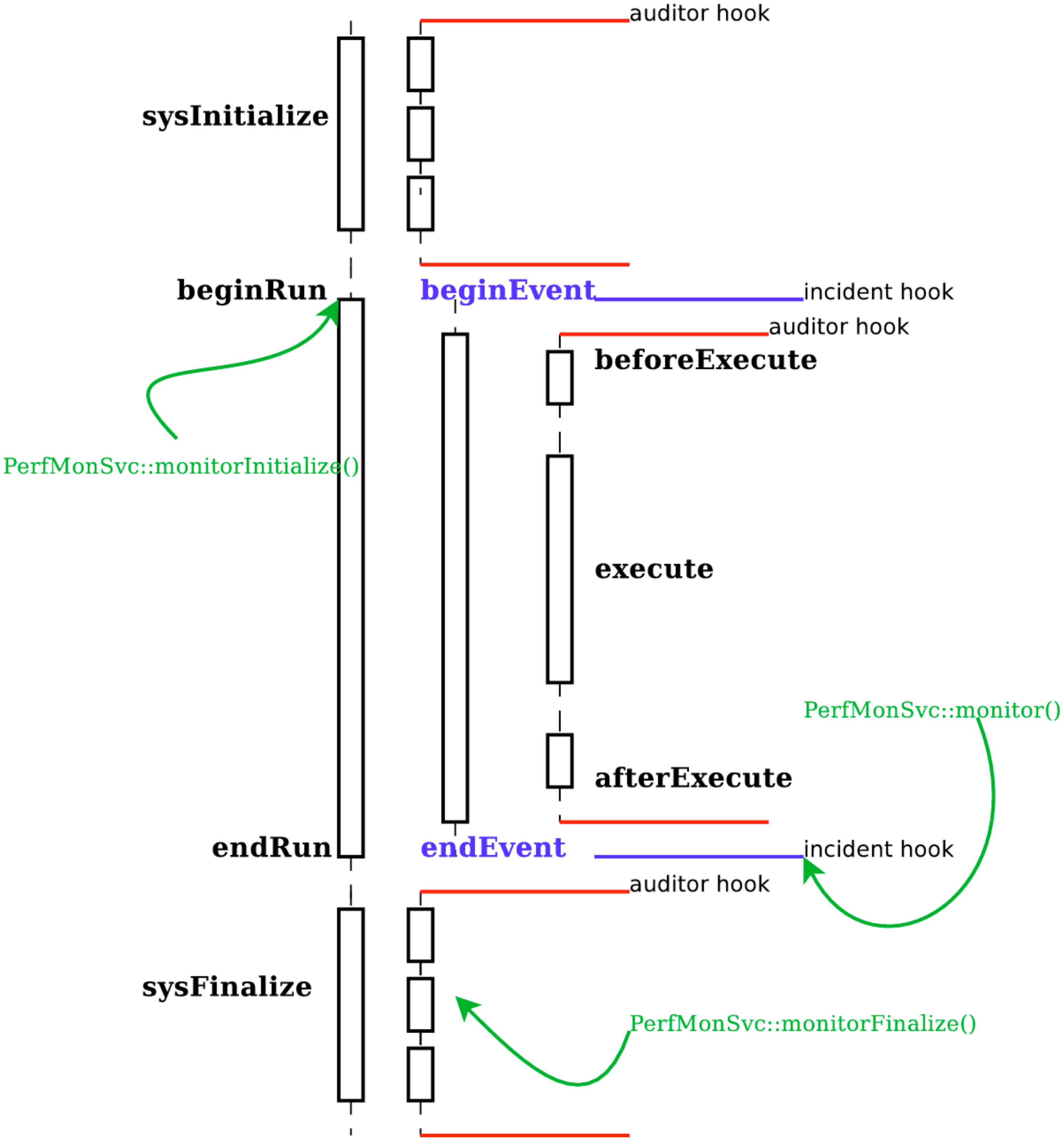
"If you can't measure it, you can't improve it"
Peter Drucker

CPU, Memory and Disk Storage are measured

- To track usage with respect to pre-defined directives and hardware limits
- Prevent bottlenecks entering production workflows
- Reduce incidence of job failures
- Give feedback to developers to remove bottlenecks and optimise their code
- Promote practice and culture of monitoring and optimisation

PerfMon

- PerfMon is a toolkit developed in the context of the Athena framework
- Captures snapshots of the application's state for later analysis
- Provides detailed information of every algorithm executed
  - CPU and wall clock time,
  - Complete memory footprint of every algorithm executed and library loaded
- Uses hooks provided by the Gaudi framework to monitor the different stages of a job
  - configuration
  - initialisation
  - execute
  - finalise
  - Retrieves data from Auditors e.g. ChronoStatAuditor, MemAuditor



PerfMonSD

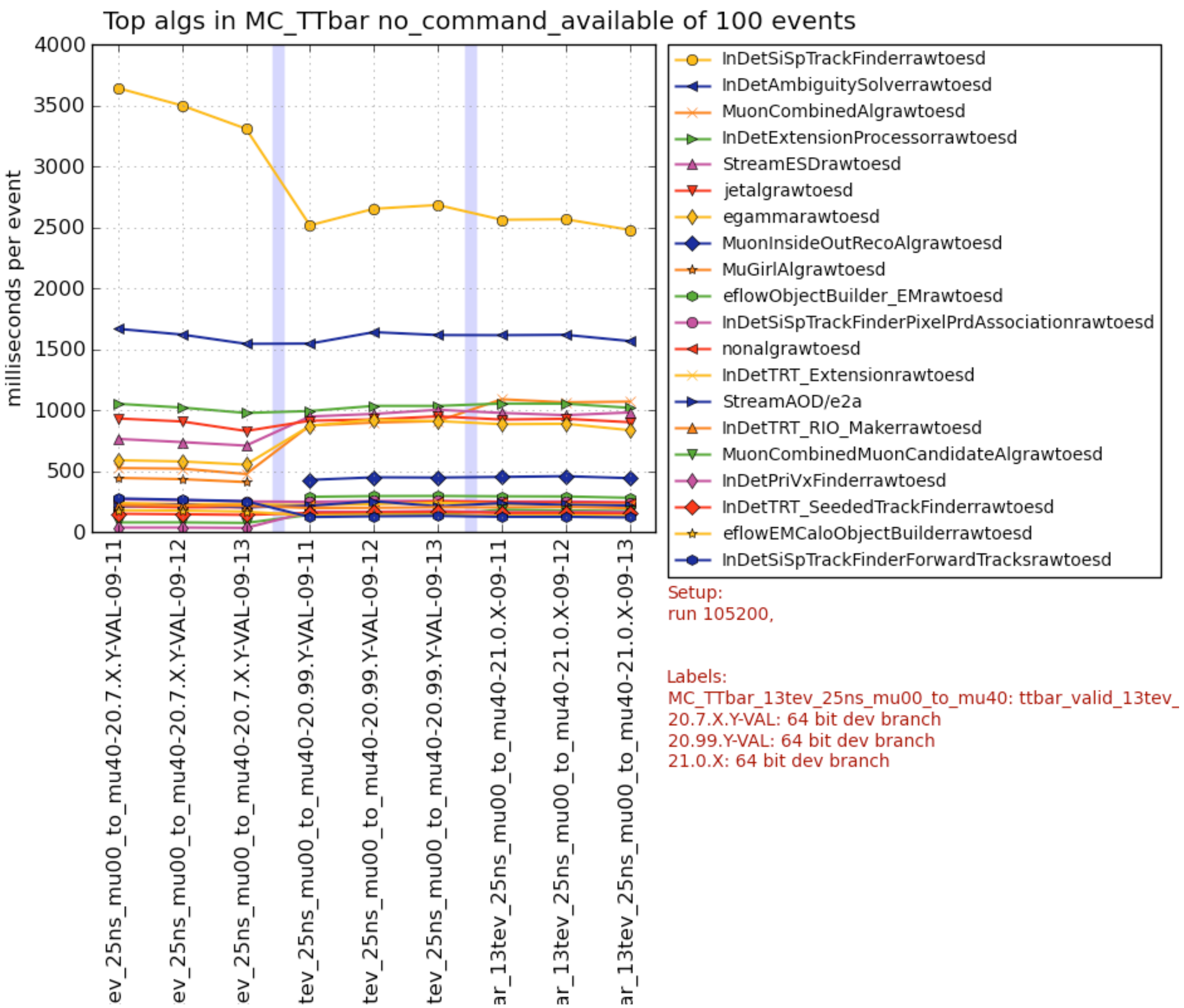
- PerfMonSD = Semi-Detailed mode

  Monitor stages of a job
- "ini" Initialisation
- "1st" the first event
- "cbk" callbacks
- "evt" event loop
- "fin" finalisation
- "dso" cost of loading libraries
- "preLoadProxy" cost of conditions data
- Summary information
- Tests with PerfMonSD enabled are run on development software releases daily
- Little overhead ~ typically adding 1% to overall resources

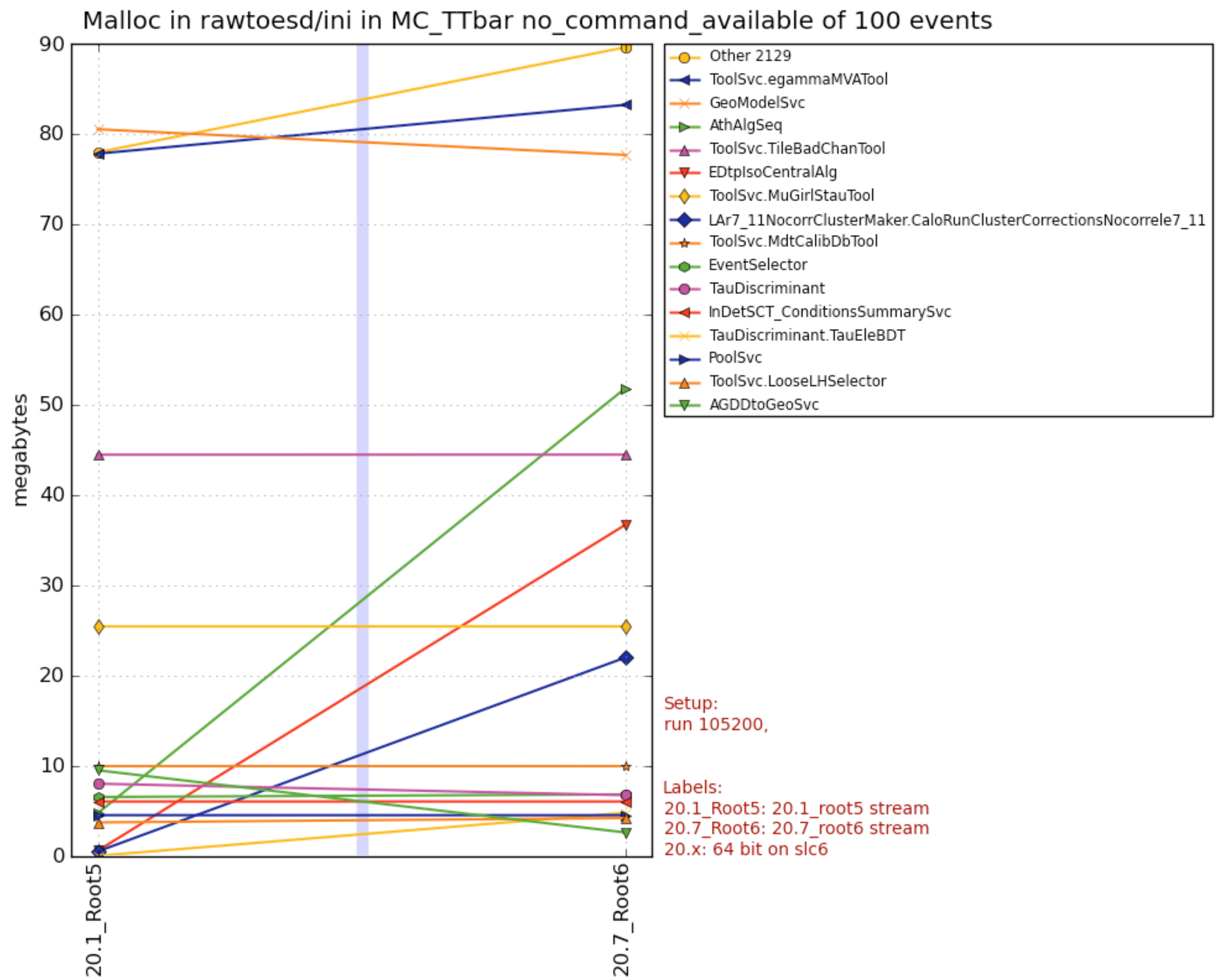
```
PMonSD ================step ini ==================
PMonSD          n    cpu    vmem  malloc component
PMonSD [ini]    1   6390   24704   23623 GeoModelSvc
PMonSD [ini]    1    160   10240   10373 ToolSvc.MdtCalibDbTool

PMonSD ================step 1st ==================
PMonSD          n    cpu    vmem  malloc component
PMonSD [1st]    1     40   16384   17610 MooSegmentMaker
PMonSD [1st]    1     70    1024     779 MboyRec

PMonSD ================ step cbk ==================
PMonSD          n    cpu    vmem  malloc component
PMonSD [cbk]    1   1880   56204   64841 TrackingGeometrySvc[0x14661258]+21
PMonSD [cbk]    1   3280   22168   20806 MboySvc[0x12587000]+e732ae70

PMonSD =================step evt ====================
PMonSD          n    cpu   max@evt    vmem   max@evt  malloc   max@evt  component
PMonSD [evt]  249    153  3850@182       0      0@0      175  2009@182  MboyRec
PMonSD [evt]  249    102    110@1        0      0@0      145   152@182  MuonRpcRawDataProvider

PMonSD ================= step fin ====================
PMonSD          n    cpu    vmem  malloc component
PMonSD [fin]    1     60       0  -49214 DetectorStore
PMonSD [fin]    1     60       0  -36871 ToolSvc.MuonTrackingGeometryBuilder

PMonSD ================= step dso =====================
PMonSD          n    cpu    vmem  malloc component
PMonSD [dso]    1    120   33100       6 liblcg_OracleAccess.so
PMonSD [dso]    1     20   32236       5 libBFieldAth.so

PMonSD ================== step preLoadProxy ================
PMonSD                   n    cpu    vmem  malloc component
PMonSD [preLoadProxy]    1    230   15692   11754 CondAttrListCollection[/MDT/T0]
PMonSD [preLoadProxy]    1    210    8992    5573 CondAttrListCollection[/RPC/CABLING/MAP_SCHEMA]

PMonSD =================== special info =====================
PMonSD          n    cpu    wall    vmem   malloc component
PMonSD [---]  250    379     379       -        - evtloop_time
PMonSD [---]  200      -       -     127      103 leakperevt_evt51plus
PMonSD [---] vmem_peak=676648 vmem_mean=660290 rss_mean=501841
```

Monitoring webpages

CPU time

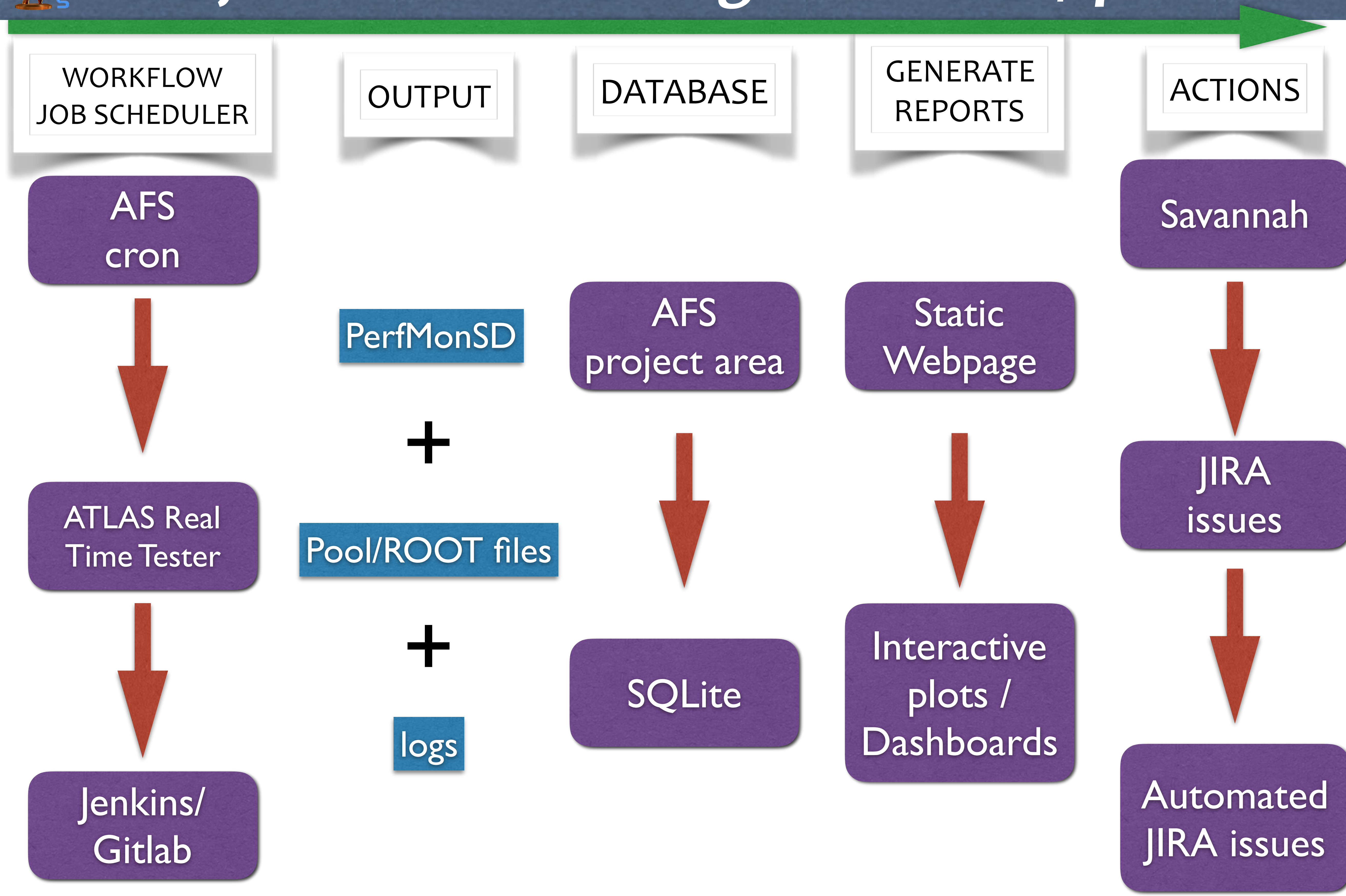


Memory



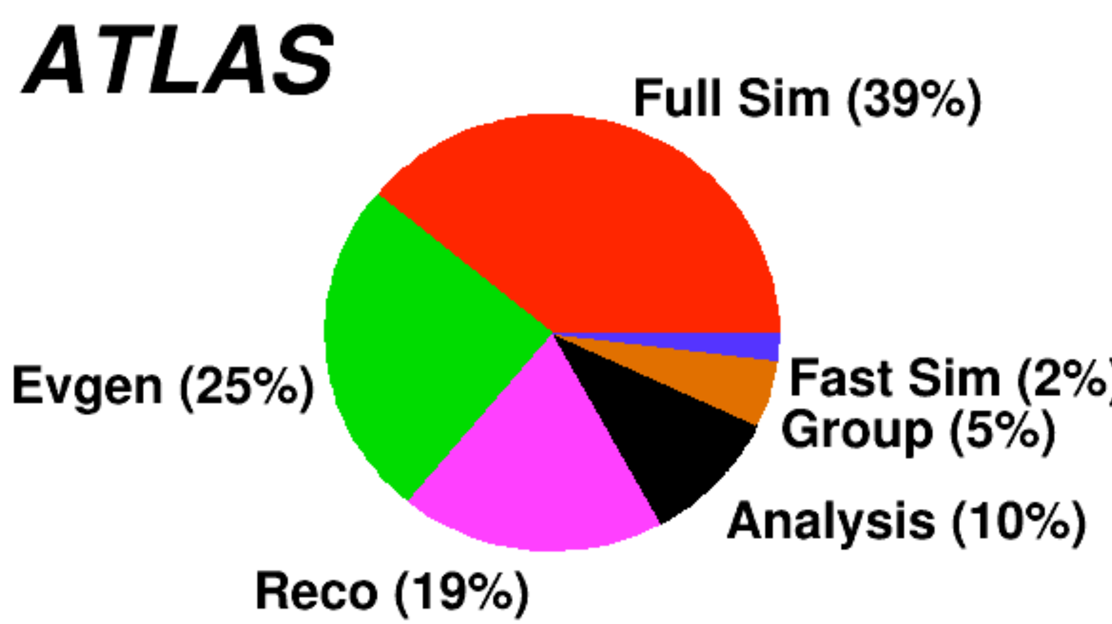
- http://atlas-pmb.web.cern.ch/atlas-pmb/

Software Monitoring evolution / plan



Monitoring of workflows in daily builds

- Generation (of physics events) EVNT
- Simulation (Produce GEANT4 hits) **EVNT to HITS**
- Digitisation (Digitise hits as detector readout) **HITS to RDO (Raw Data Objects)**
- Reconstruction (Reconstruct physics objects components) **RDO to AOD (Analysis Object Data)**
- Analysis Derivations (Slim, Skim, and Thin events for dedicated physics analysis streams) **AOD to DxAOD**
- Final Analysis (Calibrations and plot making leading to physics measurements)



Wall clock time fraction for grid and HPC jobs
July 2015 - July 2016

Step	Time per event (sec)	PSS/RSS(GB)	Release
EVNT to HITS	180	1.7	19.2
HITS to RDO (4-core)	32	6.1/8.4	20.7
RDO to RDOTrigger (4-core)	10	6.7/12.8	20.7
RAW to ESD (4-core)	13	7.2/11.9	20.7
ESD to AOD (4-core)	0.3	5.1/7.9	20.7
AOD to DxAOD MC15ExotJetM	0.4	2.4	20.7
AOD to DxAOD Data15ExotJetM	0.15	1.7	20.7
Physics Main $\langle\mu\rangle \approx 30$ (4-core)			
BS to ESD (4-core)	11	11.0/18.3	20.7
ESD to AOD	0.9	7.7/11.7	20.7

Monitoring of production workflow @ Tier0

- New program : **RunTier0Tests.py** where developer updates are tested for changes in file outputs, memory, CPU time and log file messaging in both simulated and real data workflows.

- Configured to run simultaneous tests on the same machine with and without the proposed code changes

- Around an hour to complete on typical CERN/ lxplus interactive nodes

- Options for speedup, include running all tests simultaneously but need multi-core machine with free slots

- Special options include comparisons against production releases

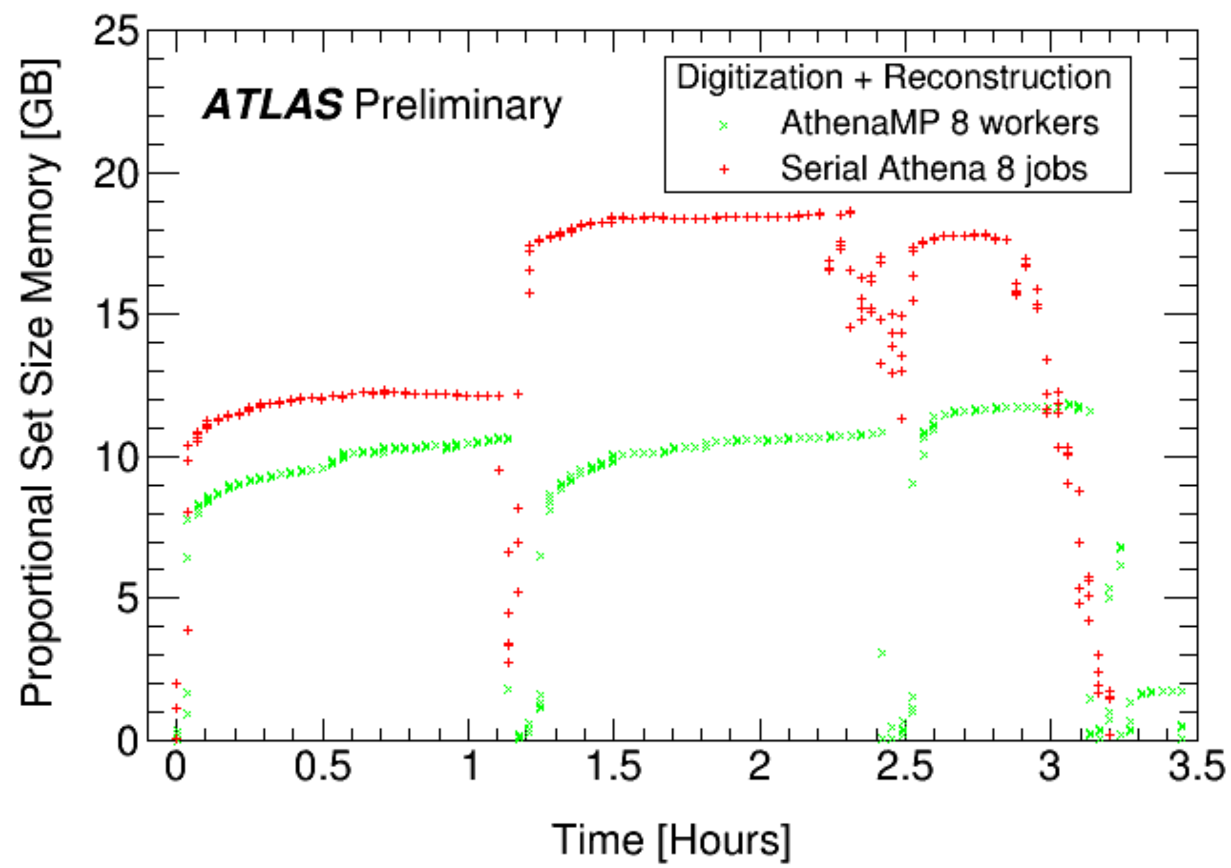
- Has had the effect of preventing disastrous memory leaks and new bottlenecks entering into the code base in imminent production releases

```
INFO
INFO    The head directory for the output of the clean Tier0 q-tests will be /tmp/harkusha
INFO
INFO    Please be aware that you are not testing your tags in the latest available nightly, which is rel_1
INFO    Your tags will be tested in environment 20.7.X.Y-VAL,x86_64,slc6,gcc49,opt,rel_2
INFO    Patch packages in your InstallArea that will be tested are:

INFO        TileMonitoring-00-08-00

INFO        WorkArea-00-00-00

INFO    ----------------- Run Athena q-test jobs---------------
INFO    Running clean in rel 20.7.X.Y-VAL,x86_64,slc6,gcc49,opt,rel_2 "Reco_tf.py --AMI q221 "
INFO    Running patched in rel 20.7.X.Y-VAL,x86_64,slc6,gcc49,opt,rel_2 "Reco_tf.py --AMI q221 "
INFO    Finished patched "Reco_tf.py --AMI q221"
INFO    Finished clean "Reco_tf.py --AMI q221"
INFO    Running clean in rel 20.7.X.Y-VAL,x86_64,slc6,gcc49,opt,rel_2 "Reco_tf.py --AMI q431 "
INFO    Running patched in rel 20.7.X.Y-VAL,x86_64,slc6,gcc49,opt,rel_2 "Reco_tf.py --AMI q431 "
INFO    Finished patched "Reco_tf.py --AMI q431"
INFO    Finished clean "Reco_tf.py --AMI q431"
INFO    ---------------------------------------------------
INFO    ---------- Post-processing of q221 Test ----------
INFO    ---------------------------------------------------
INFO    Did each step of the q221 test complete successfully?
INFO
INFO    HITtoRDO Reference test successful
INFO    HITtoRDO Patched test successful
INFO
INFO    RAWtoESD Reference test successful
INFO    RAWtoESD Patched test successful
INFO
INFO    ESDtoAOD Reference test successful
INFO    ESDtoAOD Patched test successful
INFO
INFO    AODtoTAG Reference test successful
INFO    AODtoTAG Patched test successful
INFO
INFO    All q221 athena steps completed successfully
INFO    ------------------------------------------------------------------------------
INFO    Running q221 Frozen Tier0 Policy Test on RDO for 10 events
INFO    Passed!
INFO    ------------------------------------------------------------------------------
INFO    Running q221 Frozen Tier0 Policy Test on ESD for 10 events
INFO    Passed!
INFO    ------------------------------------------------------------------------------
INFO    Running q221 Frozen Tier0 Policy Test on AOD for 20 events
INFO    Passed!
INFO    ---------------------------------------------------
INFO    Running q221 CPU Time Test
INFO    Passed!
INFO    ---------------------------------------------------
INFO    Running q221 Physical Memory Test
INFO    Passed!
INFO    ---------------------------------------------------
INFO    Running q221 Virtual Memory Test
INFO    Passed!
INFO    ---------------------------------------------------
INFO    Running q221 Memory Leak Test
INFO    Passed!
INFO    ---------------------------------------------------
INFO    Running q221 WARNINGS Test
INFO    Passed!
```

Multicore memory monitoring

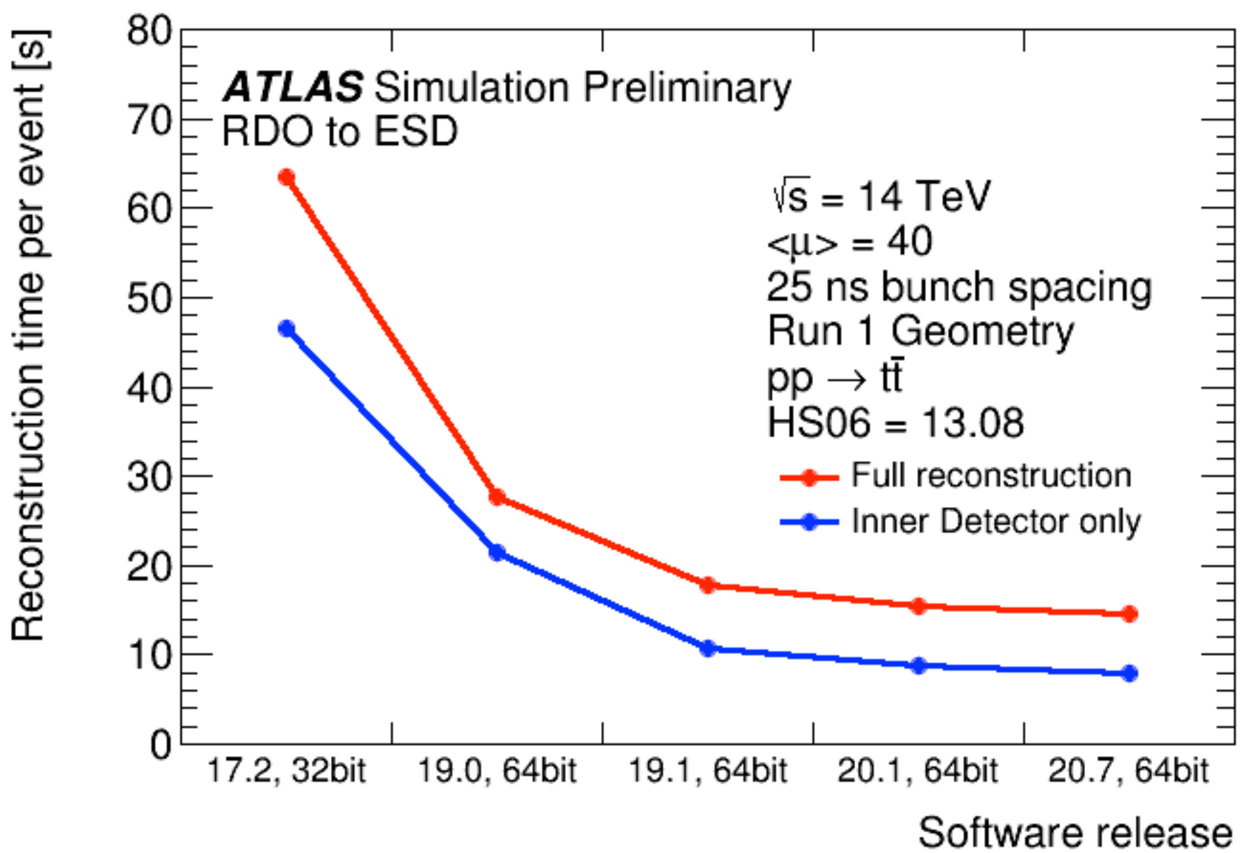
- To save memory we use "Athena" framework in multi-processor mode "AthenaMP"
- Absolutely essential to take into account memory shared across processors
- Use MemoryMonitoringTool (N. Rauschmayr, R. Seuster) and which records evolution of VMEM, Resident Set Size, Proportional Set Size and Swap memory usage for a process and all its child processes. Probes /proc/<process_ID>/smaps files.

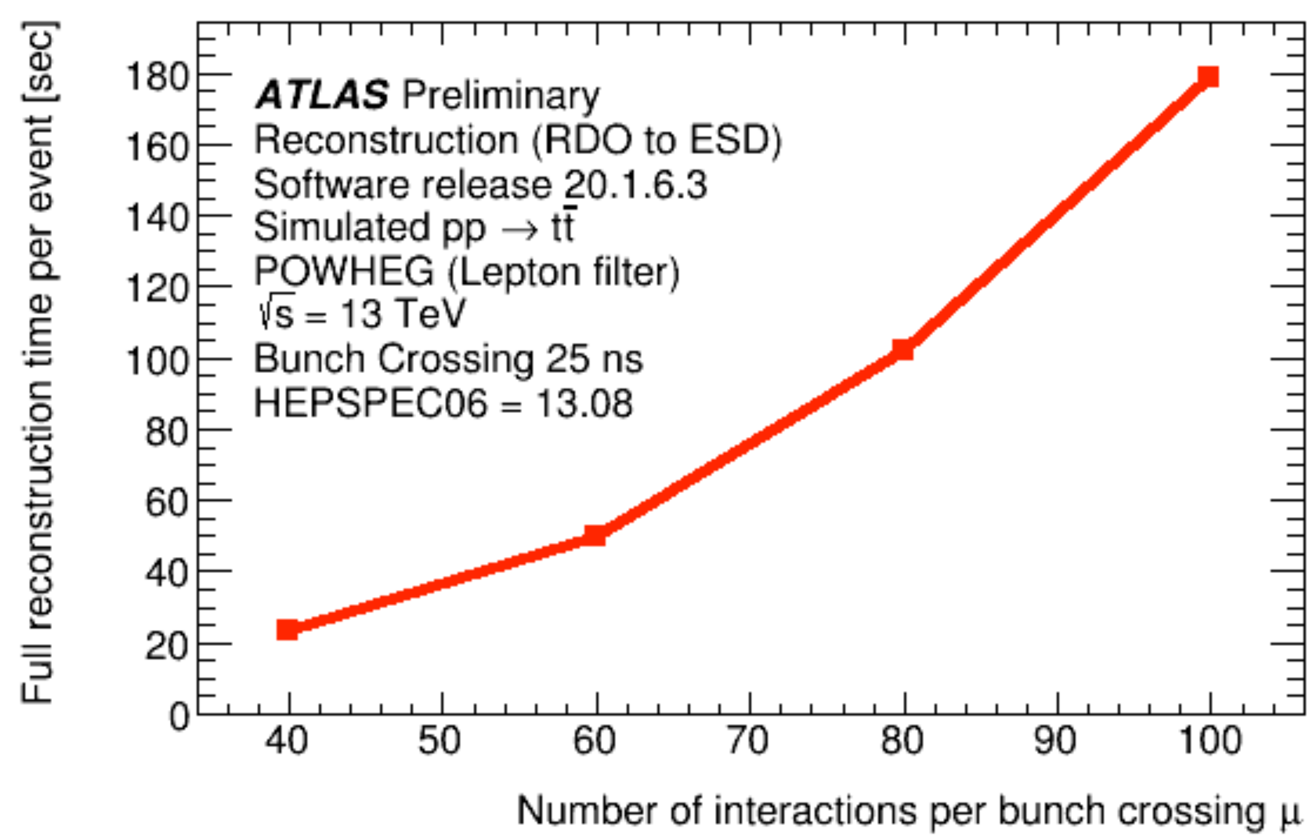
- Needed to monitor and track improvements in memory usage
- Here comparing memory using 1 multicore job using 8 cores versus 8 serial jobs using 1 core each
- Advanced and comprehensive studies of memory usage and patterns. See



Rauschmayr & Kama "Identifying memory allocation patterns in HEP software" @ CHEP 2016
Rauschmayr & Kama Find Obselete Memory

CPU time evolution / perspectives





- Current monitoring and benchmarking framework
  - has allowed developers to optimise code performance
  - anticipate resource usage based on current workflows
  - plan resource requests and define limits for for future workflows
  - react to and measure impact of external changes i.e. ROOT5 to ROOT6, CMT to CMake, CLHEP versions, migration from CLHEP to Eigen

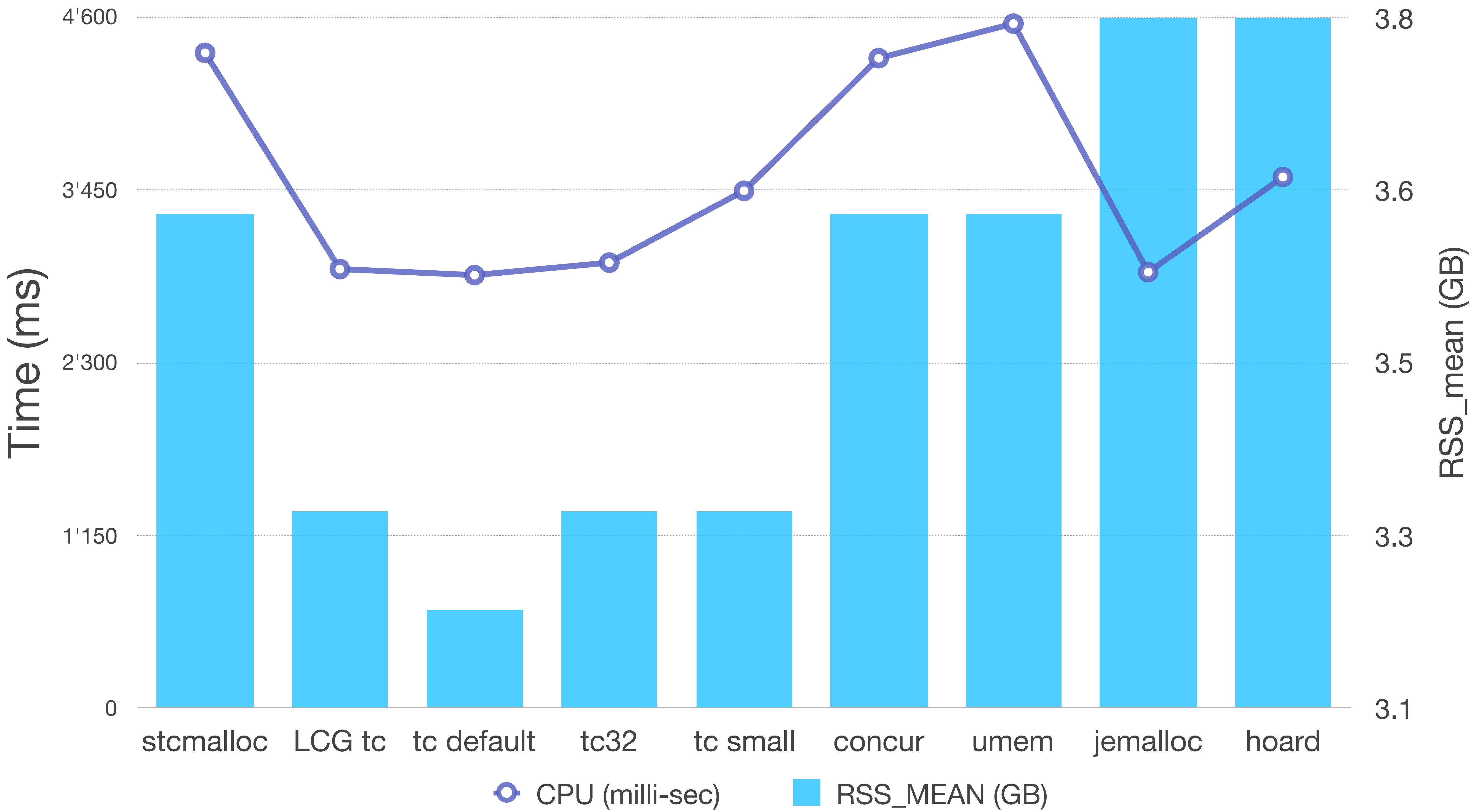
Compiler Optimisation studies

- gcc compiler has multiple optimisation levels
- Benchmark and PerfMon allow for systematic studies across multiple workflows
- gcc4.9.3
- Average CPU time per event in seconds

Workflow	-O2	-Os	-O3
Simulation	156 (1)	171 (1.10)	140 (0.90)
Digitisation	22 (1)	25 (1.14)	20 (0.91)
Trigger	7.9 (1)	11.2 (1.42)	7.6 (0.96)
Reconstruction	10.0 (1)	11.4 (1.14)	9.4 (0.94)

Memory allocator optimisation studies

- Multiple memory allocators are available were investigated for use in the reconstruction workflow (tc: Google tcmalloc)



Conclusions

- Other tools used : AthMemoryAuditor (R. Seuster) quick, lightweight memory leak checker; Valgrind (callgrind, memcheck, massif); Heap Profiles using jemalloc; FOMTool (N. Rauschmayr & S. Kama), gperftools, igprof.org
- Software monitoring is essential to optimisation of ATLAS workflows in a distributed multi-developer environment
- Existing framework is being upgraded to handle current and future workflows
- SQL will provide a powerful means of analysing performance data and will allow greater flexibility in tracking changes.
- Move to multi-threading algorithms will present considerable challenge

G. Stewart "How To Review 4 Million Lines of ATLAS Code" @ CHEP