Efficiency and cost - examples

Johannes Elmsheuser

20 June 2017
WLCG Workshop 2017, Manchester

Introduction

- The current experiment grid production systems are not completely perfect - known bottlenecks of memory usage, input file access and input and output file transfers, single vs. multi-core job scheduling etc.
- The executed payloads like event generation, simulation, reconstruction have potential inherent bottlenecks or how these are used in the grid production system

- There are basically no low hanging fruits - discussing here briefly examples of CERN IT, CMS and ATLAS where we know there might be "efficiency" gains but it takes/took a while to put them into production due to many reasons

M.Schulz

Batch on EOS project (BEER): use under-utilised EOS nodes for CPU payload processing

- Log analysis late autumn 2015
- Demonstrator build by end 2016
- Plan is putting it into production until December 2017
- Intense work by a person and all others involved can only work part time on it
- Current services are the priority for experts, but also only people who have deep knowledge to efficiently introduce new concepts.

Other example: LHC@home usage by more experiments could also utilise more resources - as heard yesterday

ATLAS Computing workflow performance understanding meetings and Analysis

- Since 1 year every ∼3 weeks a ∼1 hour meeting of ATLAS computing, software and CERN IT persons
- Discussions about grid workflows, detailed performance studies, improvement suggestions, monitoring implementations

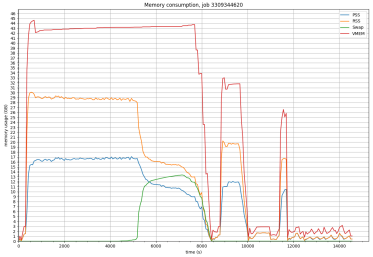
- Greatly benefits from ATLAS analytics integration of all PanDA grid jobs into elastiscsearch/kibana cluster in Chicago/MWT2 (I. Vukotic)
- Every PanDA grid job is instrumented with MemoryMonitor (N. Rauschmayr, A.Limosani) for memory and I/O measurements

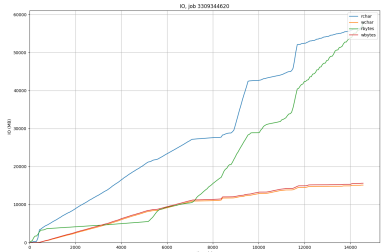
- Examples of possible improvements:
  Geant4 static linking, SharedWriter/Reader AthenaMP, Panda queue length and error codes, disk I/O in digitisation/pile-up simulation jobs

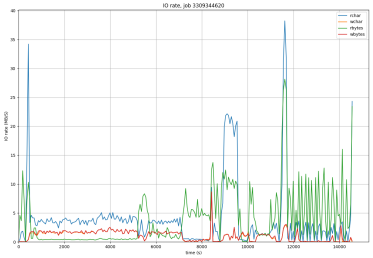
MemoryMonitor & Kibana/elastisearch

Extensive performance analysis with Chicago/MWT2 elasticsearch:

MemoryMonitor integrated into ATLAS grid jobs:







Significant (part-time) effort by a few experts to implement and make the system(s) robust

ATLAS simulation: Geant4 optimization I

- Simulation step is currently the slowest part of our production workflow, closely followed by digitisation.
- Simulation code is fairly self-contained, so it provides a good place to try out techniques which could have wider applicability
  - Increase use of Fast Simulation
  - Alternative approaches within Geant4:
    - the existing 'Frozen Showers' library
    - Build/linking optimisation
    - Multi-threading

- David Smith, CERN IT, Geant4 optimisation reports:
  Oct 20th, 2016
  Nov 10th, 2016,
  March 16th, 2017

Simulation: Geant4 optimization III

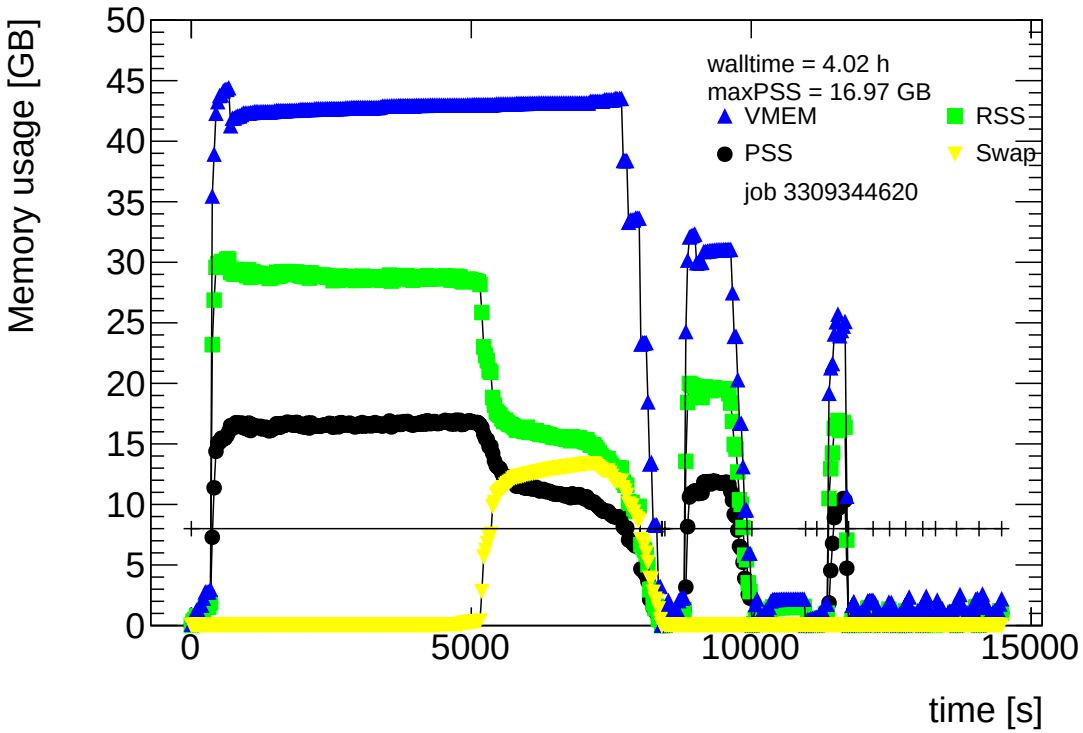
D.Smith

Summary

- Showed 2 approaches where my test showed increase in performance, by
  - Avoiding PIC mode (but using the x86-64 large code model)
    - ~10% Ivy Bridge; ~2% on Haswell
  - Avoiding PLT+GOT overhead on many G4 function calls by keeping many symbols not exported (by way of a large library)
    - ~13% Ivy Bridge; ~3-4% on Haswell

- The ATLAS Geant4 expert is in the process of integrating and testing the proposed changes in new AthSimulation/CMake releases
- Expect this to last until autumn
- Although this has been investigated in the past, it will take $\approx$ one year after "re-discovered" by external expert and potentially put into production

ATLAS reprocessing - job memory profile I



- Recent data reprocessing of 2082 events of run 310863, $< \mu >$=37.6
- RAWtoESD, ESDtoAOD, ESDtoDPD on 8-cores, output file merging sequentially at the end
- Memory swapping at end of RAWtoESD
- Additional separate job to merge outputs to O(10 GB)

ATLAS reprocessing - job memory profile II

- Improvements in memory sharing is paid by subsequent heavier disk usage which might be a potential bottleneck
- Merging in reco job could happen in parallel, but can lead to local disk overload

  → Develop a SharedWriter for output from all subprocesses

  Technically challenging since directly coupled to experiment SW framework memory and IO management
- Still merging step currently needed to have "large enough" files for transfer and storage
- Similar output merging also happens during xAOD group derivation production

→ Very challenging to optimise jobs lengths and data distribution since production works (but could be done more optimal)

Event "Pre-Mixing" for Digitization

- "Classical" pileup mixing
  - Read one event per PU and simulated bunch-crossing (BX)
  - Need typically 30x16 minimum bias events (PU30 and 16 simulated Bxs)
- per hard-scatter event
- Pre-Mixing
  - Compose one large library of mixed minimum bias events
  - Need to read one event from the library per hard-scatter
- Require one library per pileup scenario
- Two main MC DIGI-RECO campaigns in 2016, big pre-mixed one (1PB) and a small classical (50TB)
- Could gain a factor ~2 in CPU needs, as expected before

CPU efficiency varies from ~70% (both inputs local) to ~55% (both inputs remote)

	# Events	Sum CoreHrs	Avg. Time/Evt
premixed	8.0B	66.8M	~30s
classical	0.42B	6.55M	~56s

2

→ ATLAS is currently investigating/prototyping the same pile-up premixing workflow, but it will also take a while until this will be ready to use in production

C.Wissing

Multi-threaded Applications

- All main CMS applications became multi-threaded in 2016
  - RECO: Multi-threaded in production since 2015 already
  - Digitization "DIGI": Multi-threaded in prodcution since 2016
  - Simulation "SIM": Moved to multi-thread-endabled Geant4 v10.x for 2016 CMSSW releases
- Applications are usually executed with 4 threads
- Analysis jobs are single-threaded by default
- Advantages
  - Reduced memory/core requirements
  - Less jobs needed to achieve same amount of events
    - Reduce scaling issues of production system: Less jobs to track
- Challenges
  - Scheduling of multi-threaded and single-threaded jobs
  - Maintaining CPU efficiency

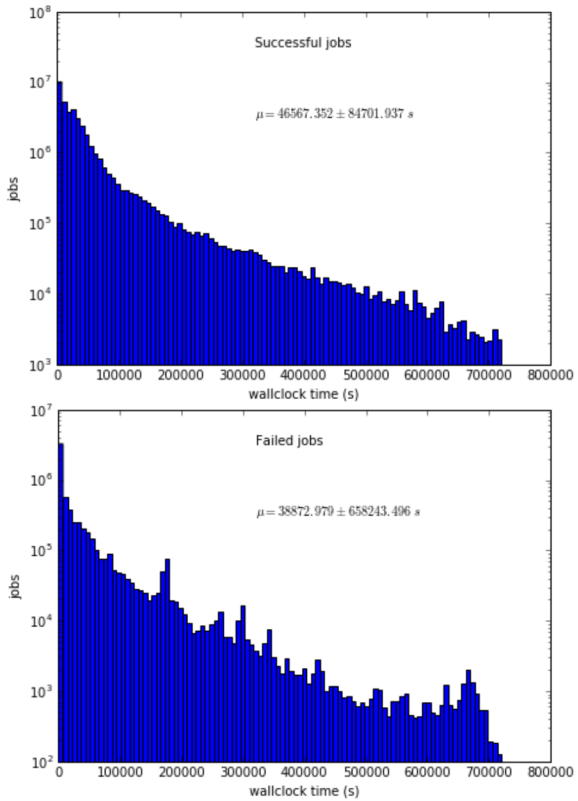
3

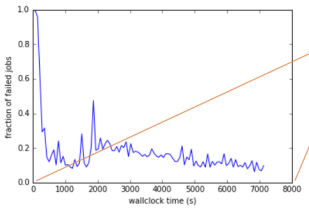
→ similarly for ATLAS Run3 multithreaded framework scheduling challenges ahead

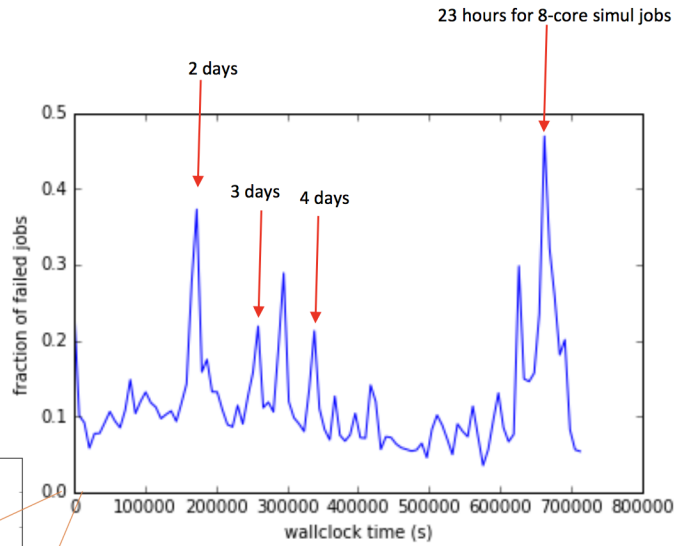
ATLAS Panda jobs error code analysis

- Andrea Sciaba made a detailed study of Panda job error codes, 26 Jan 2017
- All production payloads show similar wallclock inefficiencies

Wallclock distributions







Visible structure in fraction of failed jobs vs. wallclock with peaks at zero and specific times, some easy to interpret, other less
All jobs of O(100s) are failed (of course)

Summary and Conclusions

- There are several examples in the experiments grid systems or experiments payloads that could improve the efficiencies by a few percents each
- But it take sometimes quite a long time to put them into production either due to physics constraints/priorities or since experts have to carefully maintain current systems