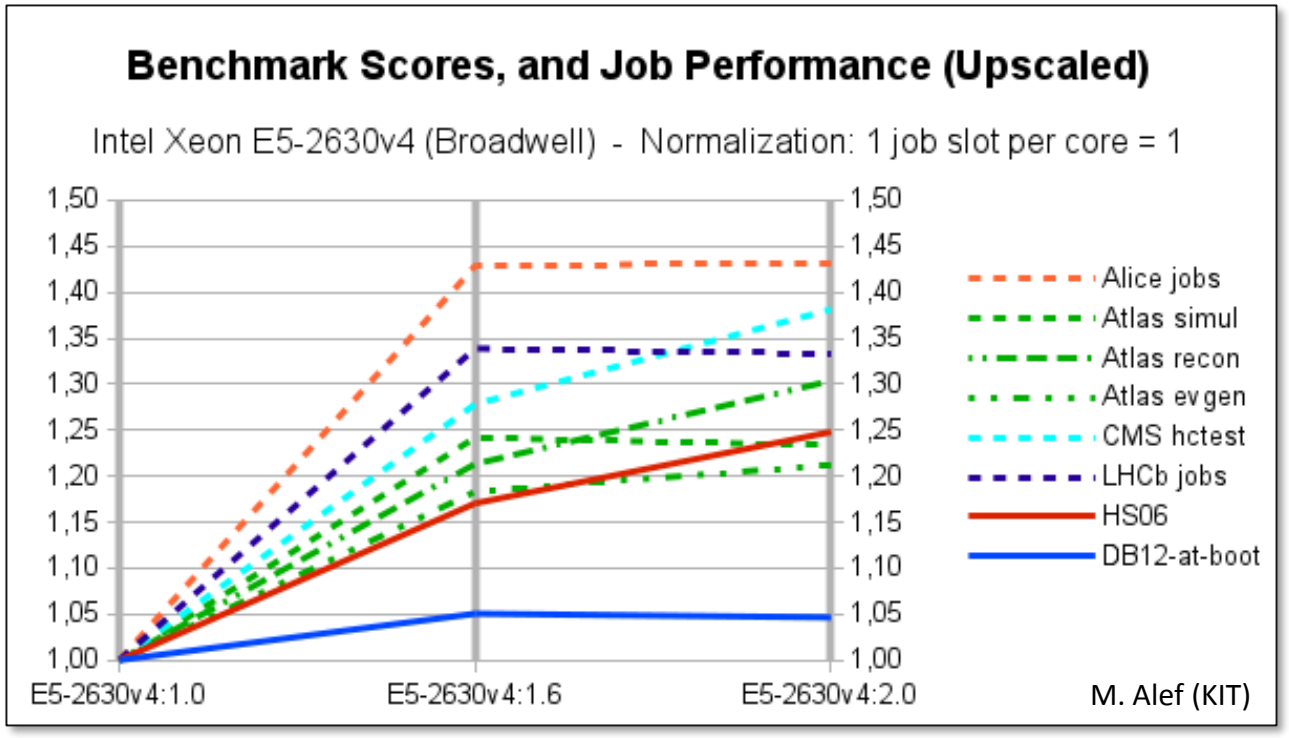
Manfred Alef (KIT)
Domenico Giordano (CERN)
Michele Michelotto (INFN)
On behalf of HEPiX Benchmarking Working Group

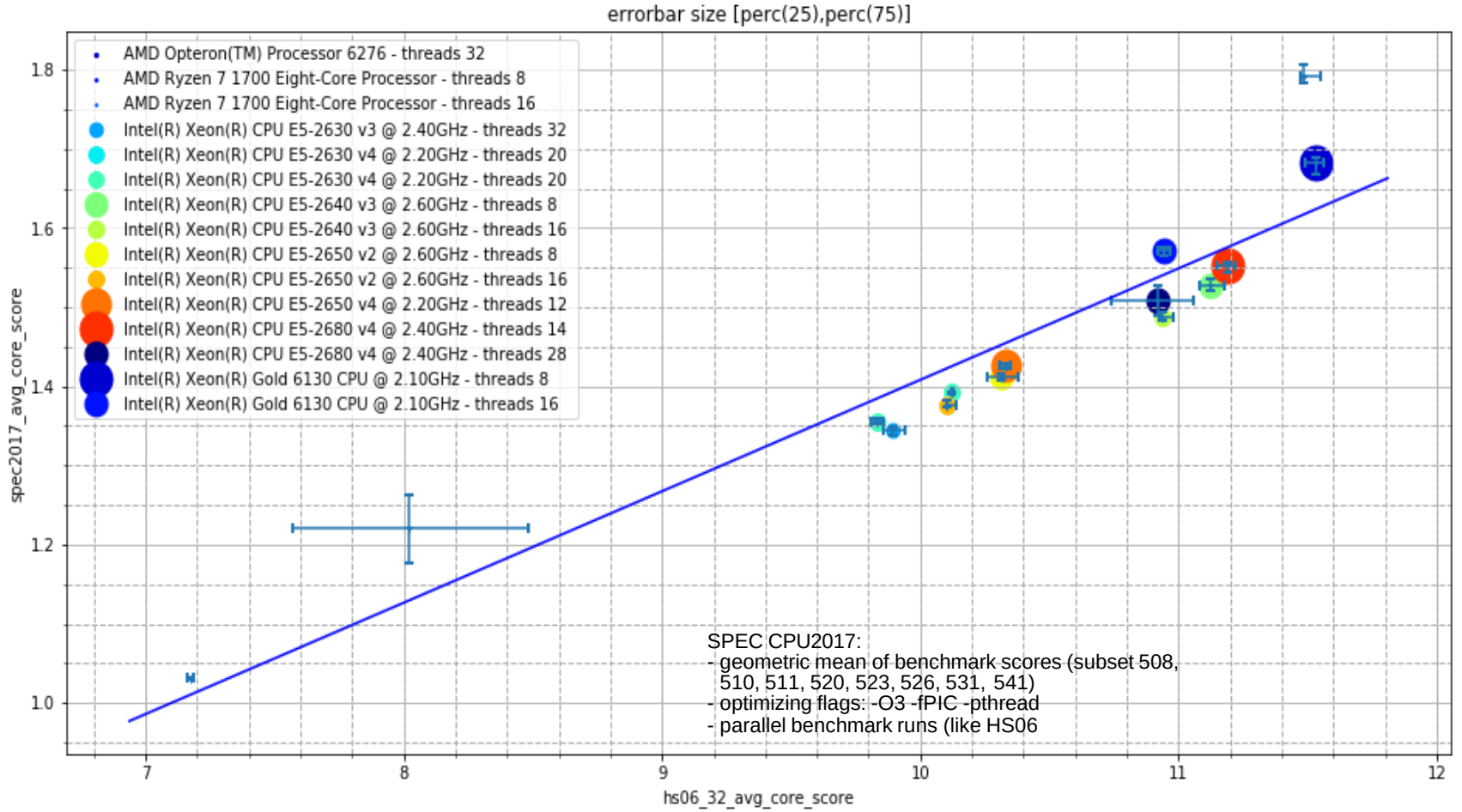
HEPiX Spring 2019

HS06 obsolescence



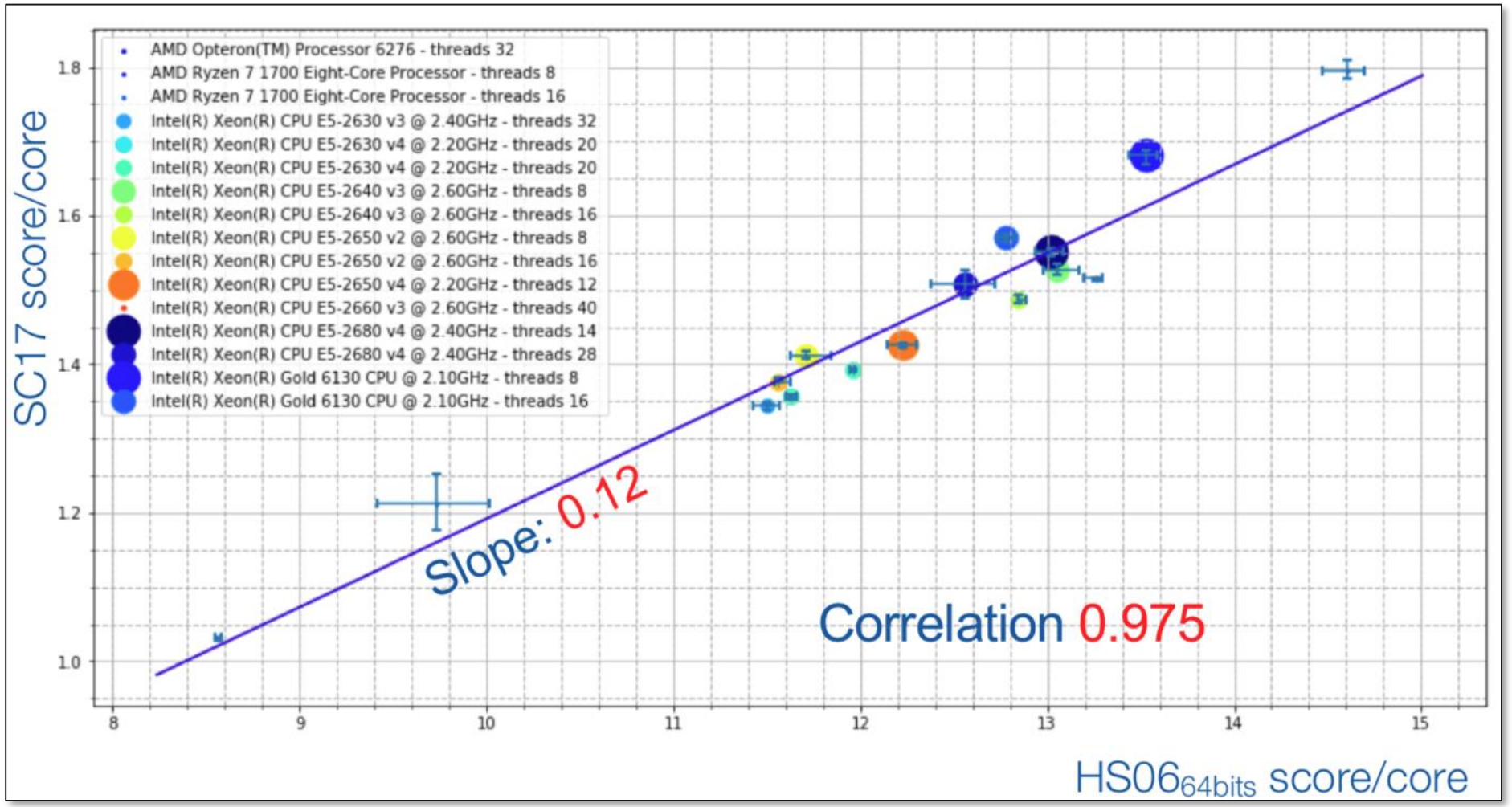
- Still using HS06 32bit since a decade even if scaling with “real” LHC applications is poor
- Experiments run at 64 bit but this a +10-20% effect
- Some experiments have some kind of magic boost on intel cpu after Haswell
- DB12 proposed at some point, but found to be inadequate
  - Dominated by front-end calls and branch prediction units
  - Ok as a rough fast estimate of HS06 at runtime

SPEC CPU2017 vs HS06



- SPEC CPU2017 (64bit) very similar to official HS06 (32bit) with current x86 architectures
- Slope 0.14, correlation 0.935

SPEC CPU2017 vs HS06 (64bit)



- SPEC CPU2017 (64bit) very similar also to HS06 (64bit) with current x86 architectures
- Correlation with HS06 r=0.975

Workloads provided by experiments

- ALICE
  - A p-p simulation job using Geant3
- ATLAS
  - A ttbar simulation job using Geant4
  - A digitization + reconstruction job
  - A DxAOD derivation job
- CMS
  - Generation + simulation of ttbar events
  - Digitization and trigger simulation with premixed pile-up
  - Reconstruction job producing AODSIM and MINIAODSIM
- LHCb
  - Generation + simulation using Geant4 of $D^{*}(\rightarrow\pi(D^{0}\rightarrow K\,\pi))\,\pi\,\pi\,\pi$ events

Thanks to Andrea Sciabà for all the slides on workload

Summary of performance metrics

Job	Events	Processes /threads	Wallclock time (sec)	CPU efficiency	Time per event (sec)	Memory per core (GB)	Read rate per core (MB/sec)	Write rate per core (MB/sec)
ALICE sim	1000	1	10901	100%	10.90	0.96	0.0788	0.1660
ATLAS sim G4	1000	8	33627	100%	269.02	0.44	0.0152	0.0090
ATLAS digireco	2000	8	13981	87%	55.92	1.12	0.3174	0.2412
ATLAS deriv	95741	8	8401	98%	0.70	1.20	0.6849	0.0705
CMS gensim	1000	8	2651	99%	21.21	0.19	0.0473	0.0377
CMS digi	1000	8	737	78%	5.90	0.65	0.2854	0.3004
CMS reco	1000	8	1221	83%	9.77	0.45	0.3073	0.2153
LHCb gensim	10	1	1782	100%	178.20	0.89	0.3115	0.0117

https://cernbox.cern.ch/index.php/apps/files/?dir=/__myprojects/wlcg-cost-model/Workloads/sciaba&

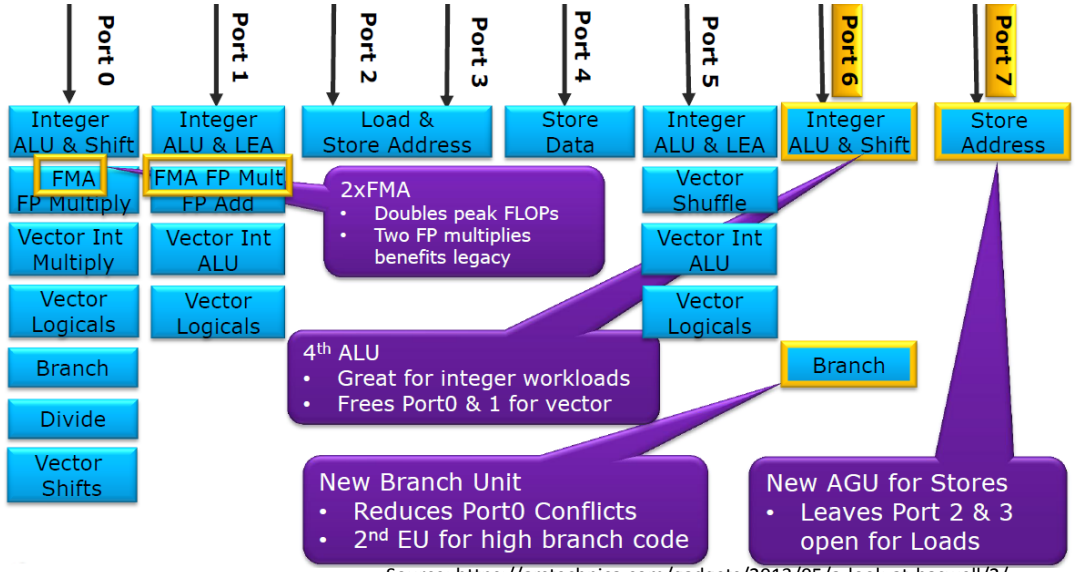
Thanks to Andrea Sciabà for all the slides on workload

Trident - Core Performance Analysis

- Instruction Per Cycle (IPC)
  - Denotes ratio of parallel instructions executed
  - Modern processors like Intel Haswell EP can do 4 IPC
- Top down characterization
  - Identifies the resources dominated by workload
  - Front-End – fetch and decode program code
  - Back-End – monitor and execution of uOP once the dependent data operands availability
  - Retiring – Completion of the uOP
  - Bad speculation – uOPs that are cancelled before retirement due to branch misprediction
- Execution Unit Port Utilization
  - Determines how many cycles the port was busy
  - Identifies broadly the pressure from different types of uOPs
  - INT & FP operations, address calculation, etc.,

Thanks to Servesh Muralidharan for all the slides on Trident

Intel Haswell EP



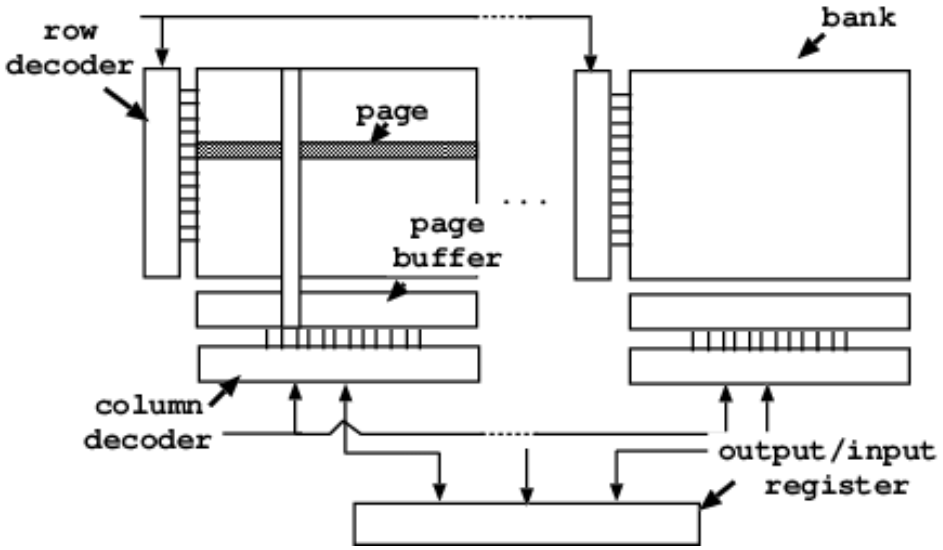
Source: https://arstechnica.com/gadgets/2013/05/a-look-at-haswell/2/

	Expected Range of Pipeline Slots in This Category, for a Hotspot in a Well-Tuned:		
Category	Client/Desktop Application	Server/Database/Distributed application	High Performance Computing (HPC) application
Retiring	20-50%	10-30%	30-70%
Back-End Bound	20-40%	20-60%	20-40%
Front-End Bound	5-10%	10-25%	5-10%
Bad Speculation	5-10%	5-10%	1-5%

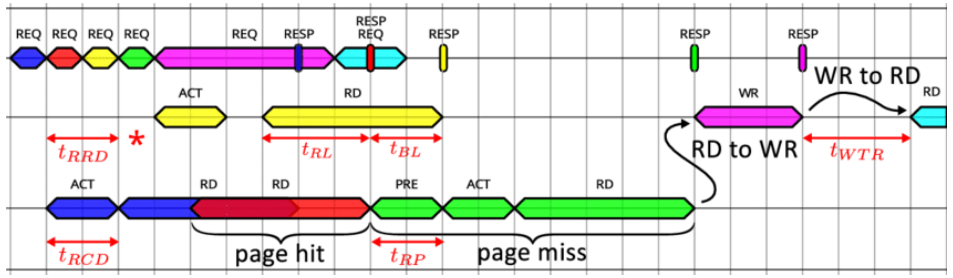
Source: https://software.intel.com/en-us/vtune-amplifier-help-tuning-applications-using-a-top-down-microarchitecture-analysis-method

Trident - Memory Performance Analysis

- Memory bandwidth usage
  - Amount of data read and written to main memory
- Memory transaction classification
  - Memory transaction occurs through pages from memory bank
  - Page-Hit
    - Memory bank in open state
    - Lowest access latency (Open)
    - Sequential memory access usually have high page hits
  - Page-Empty
    - Memory bank is idle and needs to be activated
    - Moderate access latency (Usually 2x of page hit)
    - Random memory access usually have high page empty counts
  - Page-Miss
    - Memory to be accessed requires closing of a page in the same bank
    - Worst access latency (Usually 3x of page hit)



Source: https://www.researchgate.net/figure/Multi-banked-SDRAM-architecture_fig1_221341164

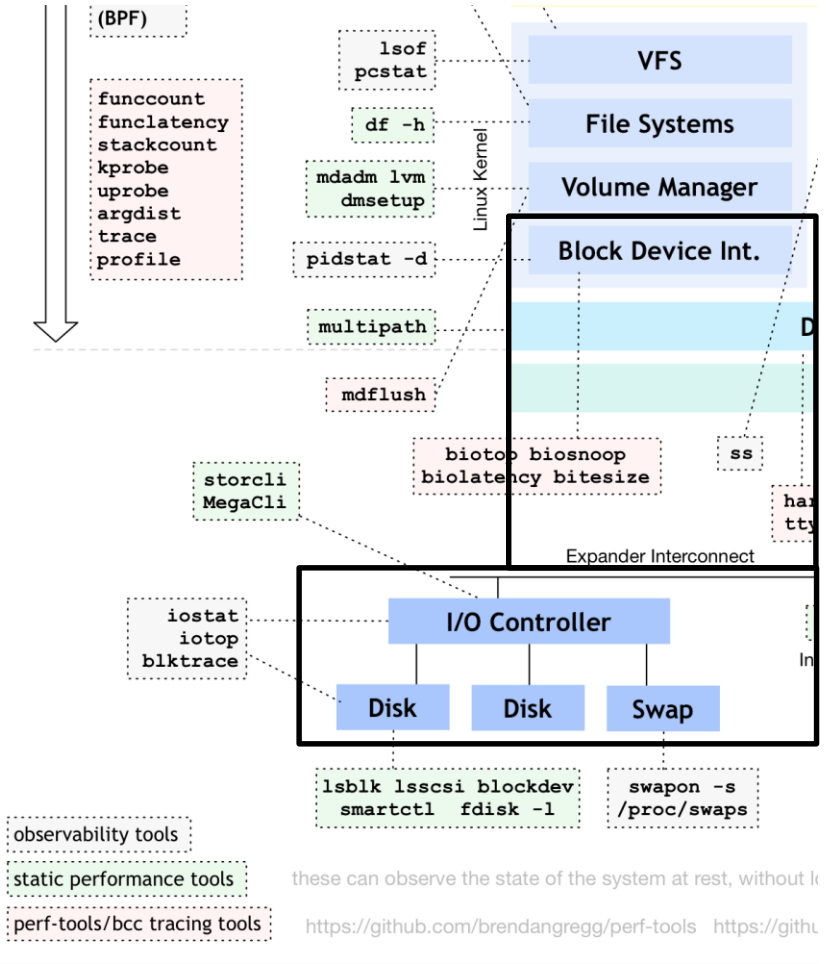


Source: https://www.researchgate.net/figure/Example-of-a-typical-TLM-trace-with-DRAM-related-phases_fig4_282806593

Further info: https://www.anandtech.com/show/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-were-afraid-to-ask/5

Trident - IO Performance Analysis

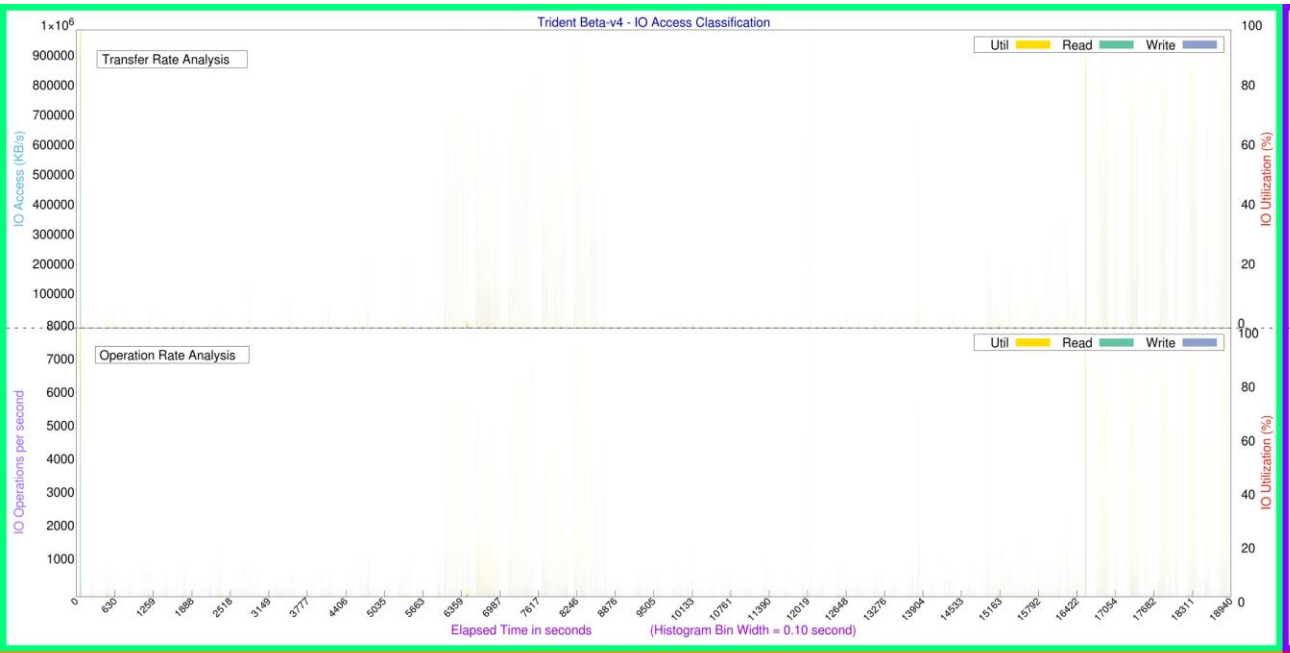
- Data recorded form ProcFS
- Transfer Rate Analysis
  - Amount of data read and written to storage
  - Limited by interface and type of memory
  - Sustained Vs Bursts
- Operation Rate Analysis
  - Amount of operations performed
  - Limited by controller for fast memory
  - Limited by disk head for HDDs
  - Random Vs Sequential accesses

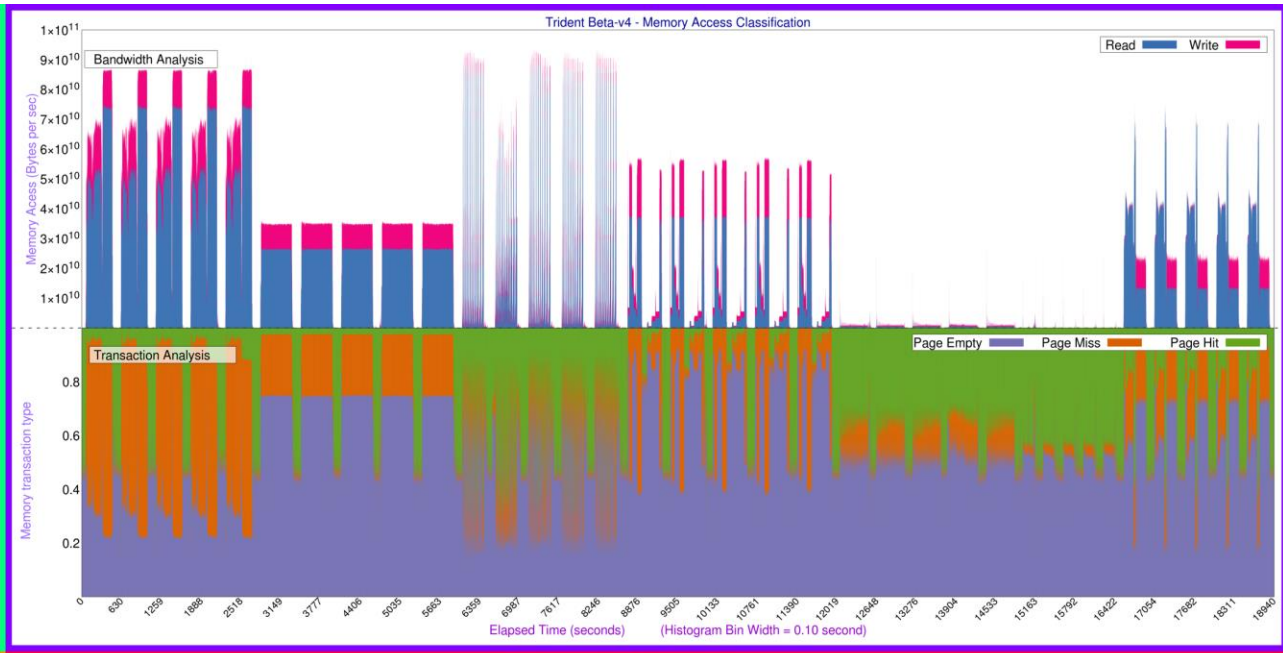


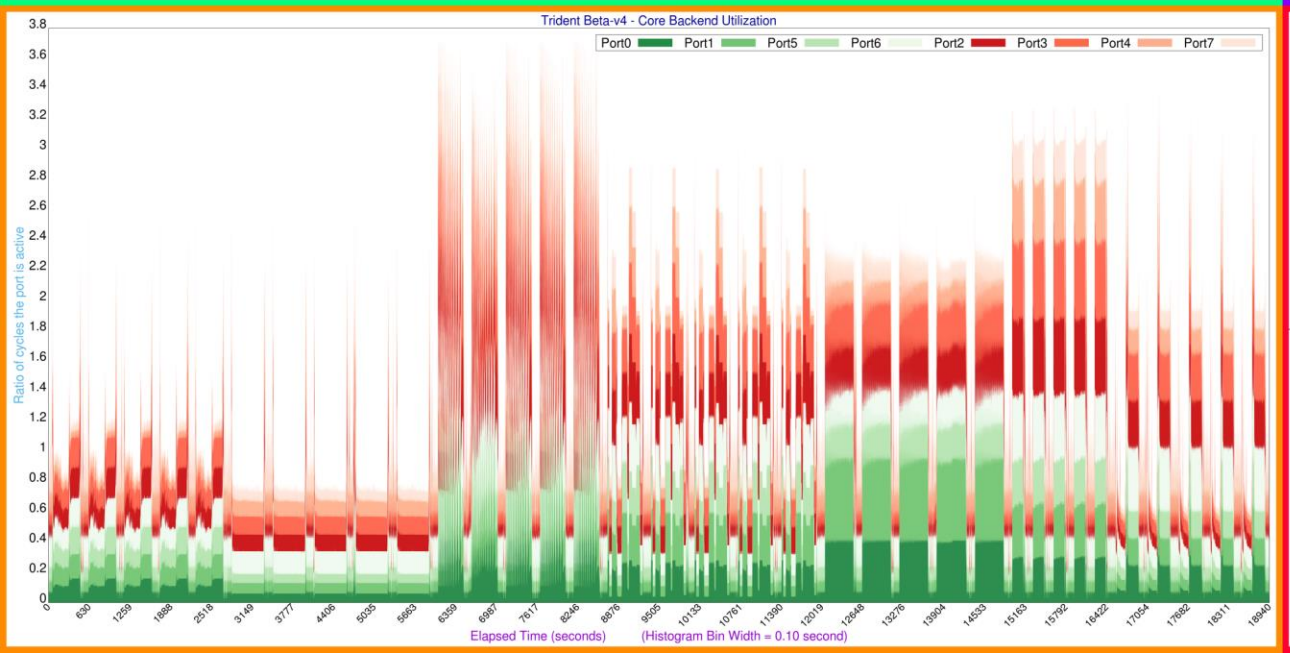
Experiment setup

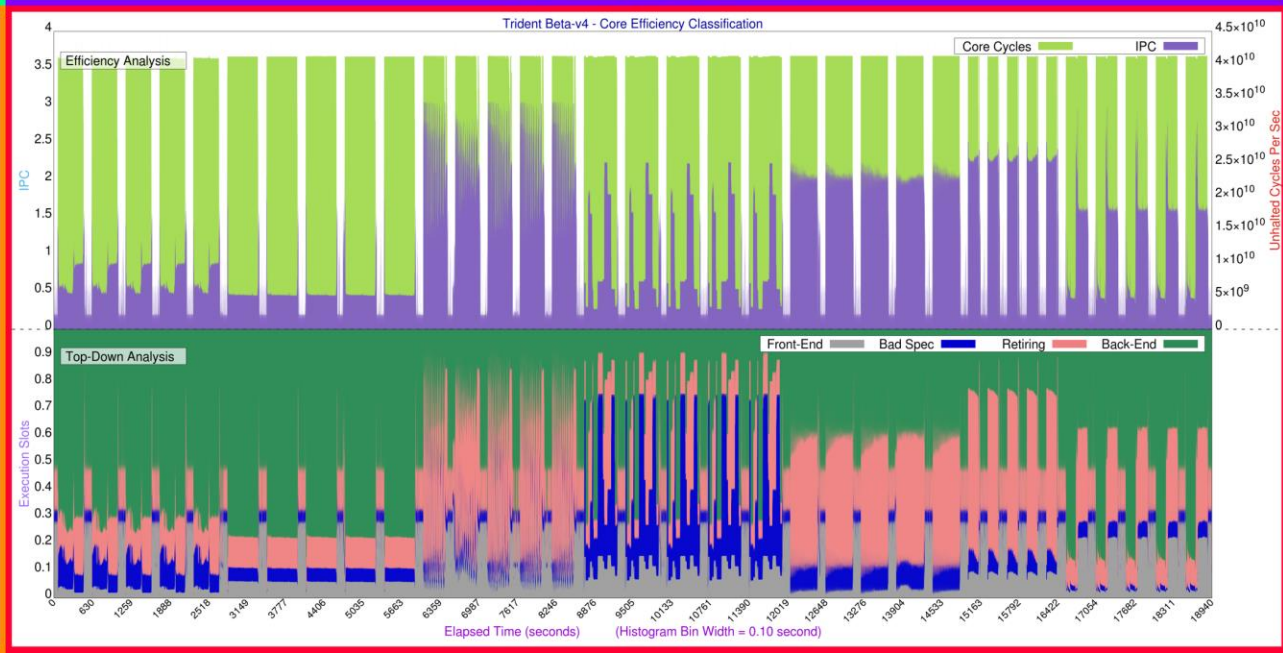
- Test System
  - 2x Intel(R) Xeon(R) E5-2630 v3@2.40GHz /
  - 64GB DDR4-2133 RAM
  - Centos 7 (3.10.0-957.5.1.el7.x86_64)
  - Non-HT, 8 physical cores per socket
- Workloads
  - Standard Benchmarks – Repeated five times
    - Extracted from Domenico Giordano's Cloud Benchmark Suite
    - HEPSPEC06 Benchmark Suite
      - 450.soplex,471.omnetpp,447.dealII,473.astar,444.namd,453.povray,483.xalancbmk
    - SPEC CPU2017 Benchmark Suite
      - 508.namd_r,510.parset_r,511.povray_r,520.omnetpp_r,523.xalancbmk_r,526.blender_r,531.deepsjeng_r,541.leela_r
  - HEP Workload Based Benchmarks - Pre & Post wait time of 60 seconds
    - ATLAS – MC Simulation, MC Digitization & Reco, Derivation production
    - CMS – Gen+Sim, Digi+Trigger+PileupSimulation, Reco+Analysis data creation – Streaming Inputs
    - LHCb – Generation + Simulation (DG Cloud Benchmark Suite)
    - ALICE – Generation + Simulation (DG Cloud Benchmark Suite)

HS06

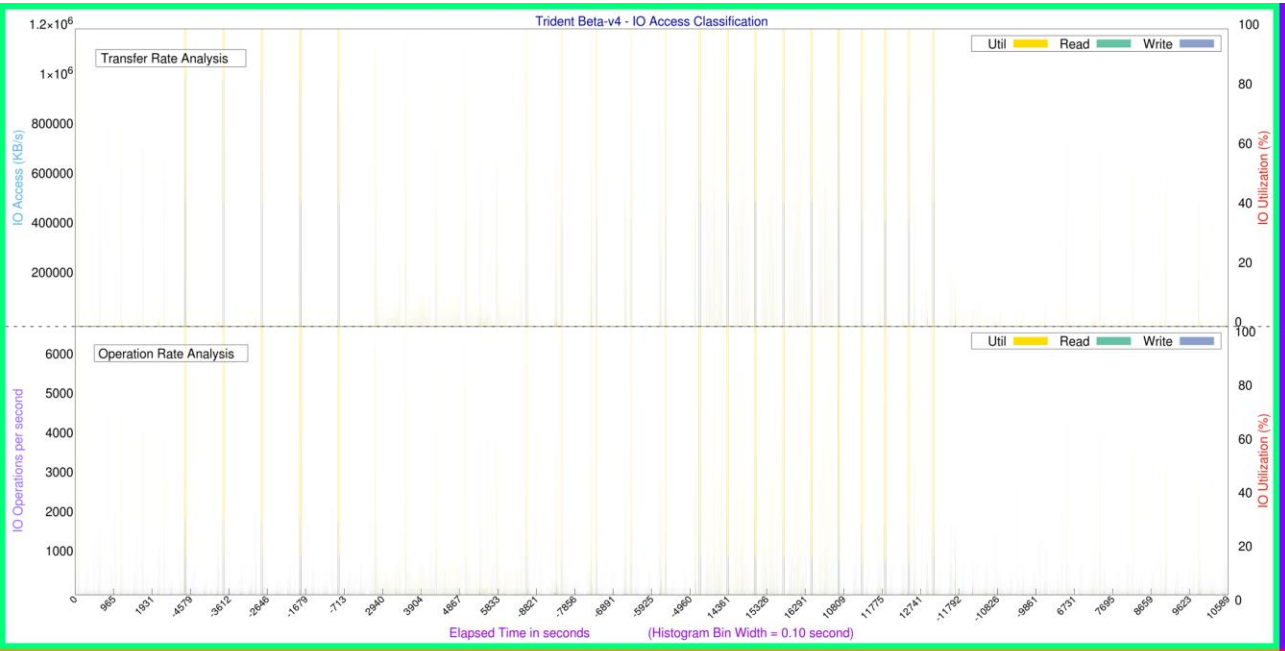

Trident Beta-v4 - IO Access Classification
Transfer Rate Analysis
Util Read Write
IO Access (KB/s)
IO Utilization (%)
Operation Rate Analysis
IO Operations per second
Elapsed Time in seconds
(Histogram Bin Width = 0.10 second)

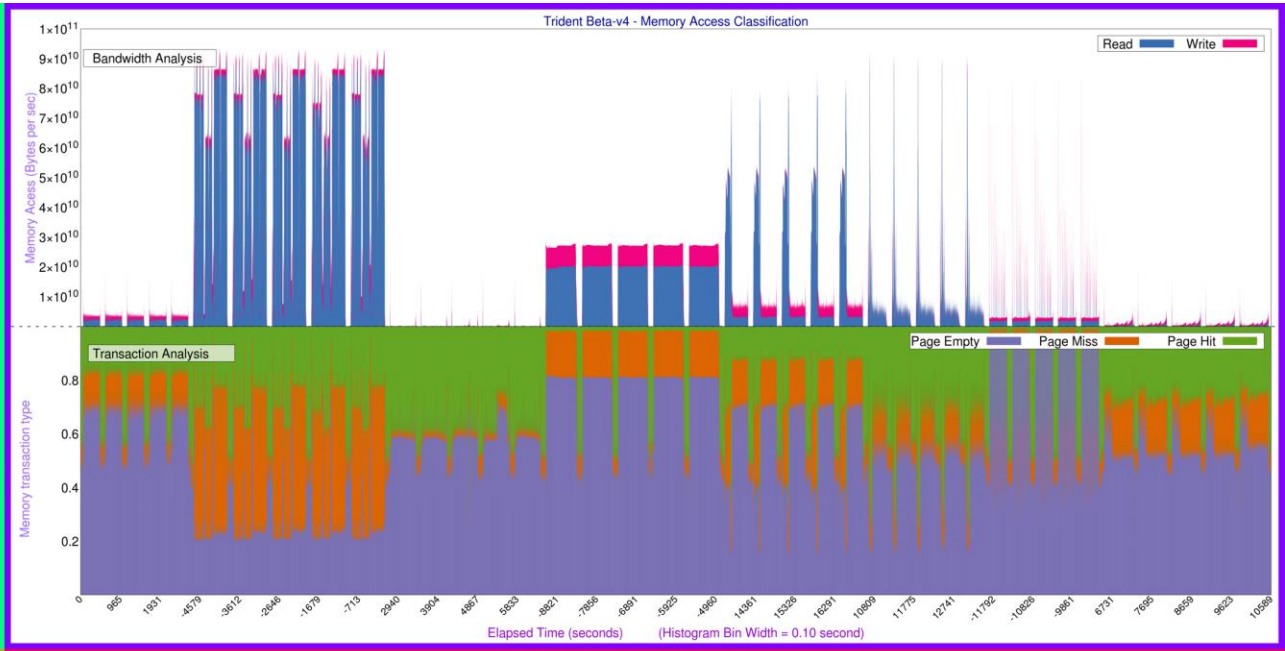

Trident Beta-v4 - Memory Access Classification
Bandwidth Analysis
Read Write
Memory Acess (Bytes per sec)
Transaction Analysis
Page Empty Page Miss Page Hit
Memory transaction type
Elapsed Time (seconds)
(Histogram Bin Width = 0.10 second)

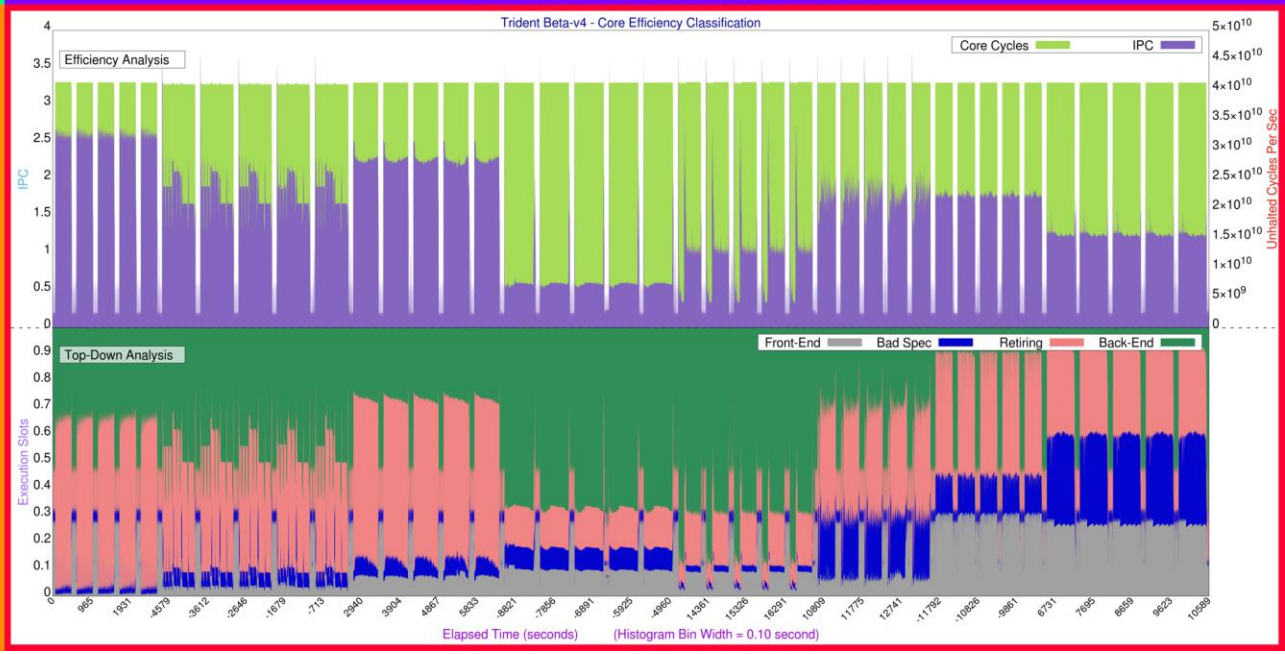

Trident Beta-v4 - Core Backend Utilization
Port0 Port1 Port5 Port6 Port2 Port3 Port4 Port7
Ratio of cycles the port is active
Elapsed Time (seconds)
(Histogram Bin Width = 0.10 second)


Trident Beta-v4 - Core Efficiency Classification
Efficiency Analysis
Core Cycles IPC
IPC
Unhalted Cycles Per Sec
Top-Down Analysis
Front-End Bad Spec Retiring Back-End
Execution Slots
Elapsed Time (seconds)
(Histogram Bin Width = 0.10 second)

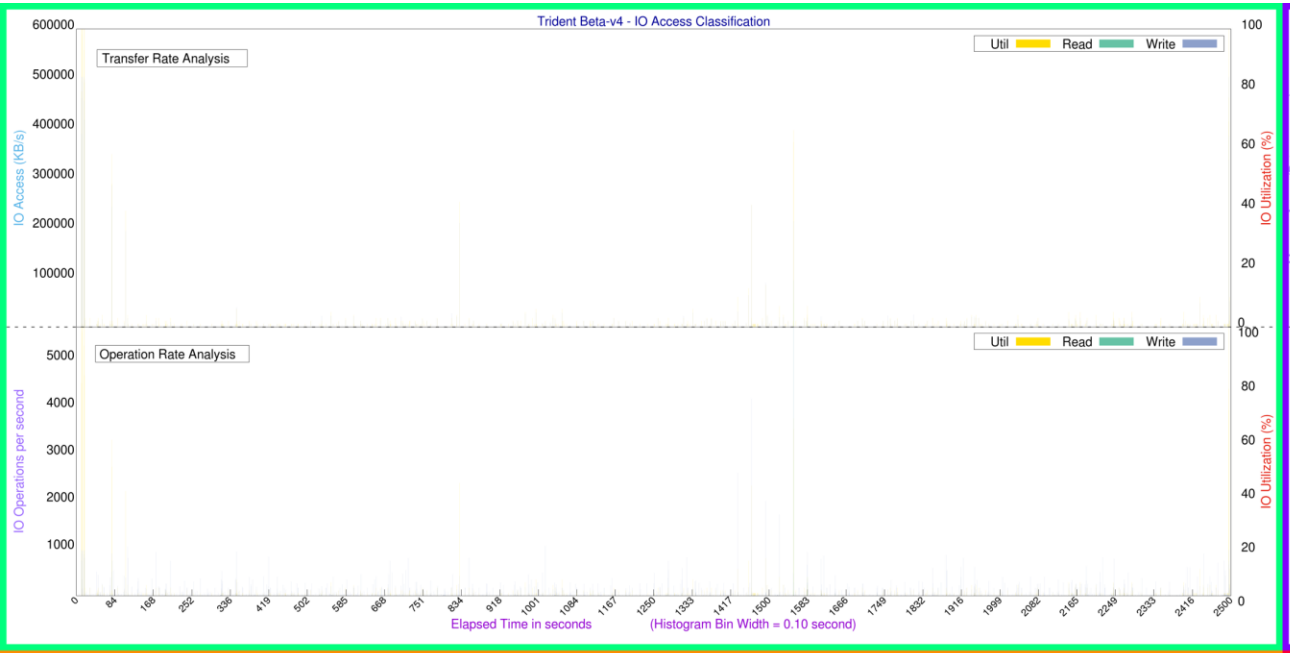
SPEC CPU2017

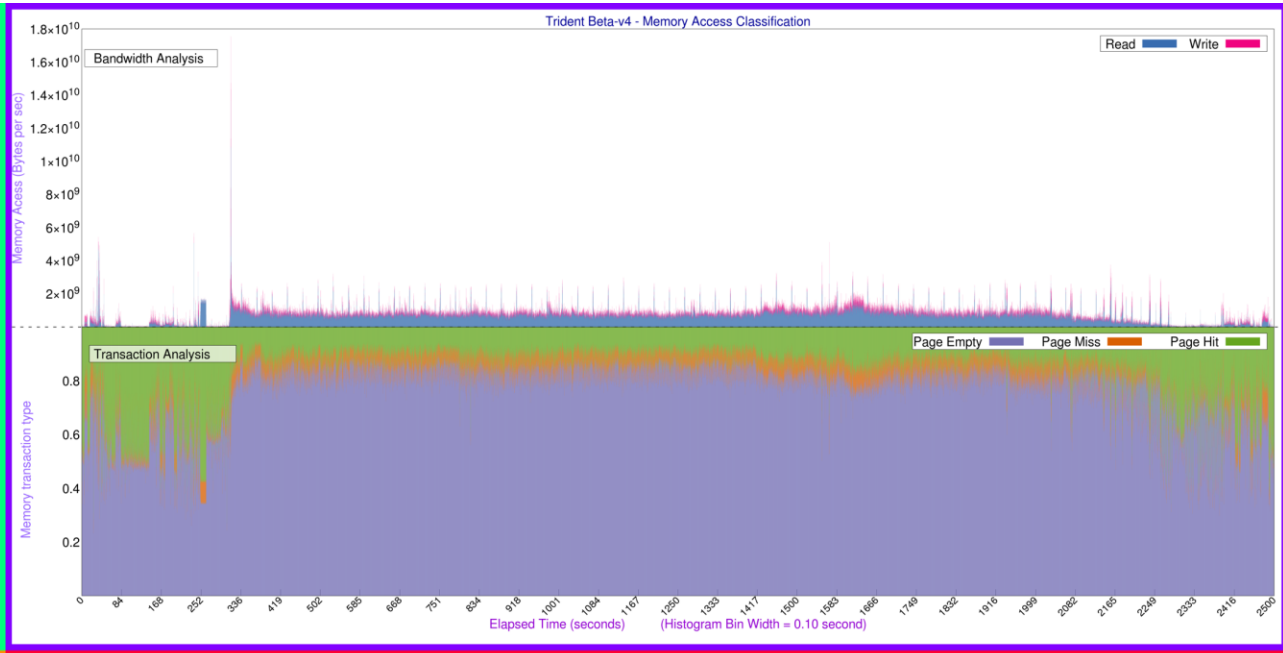


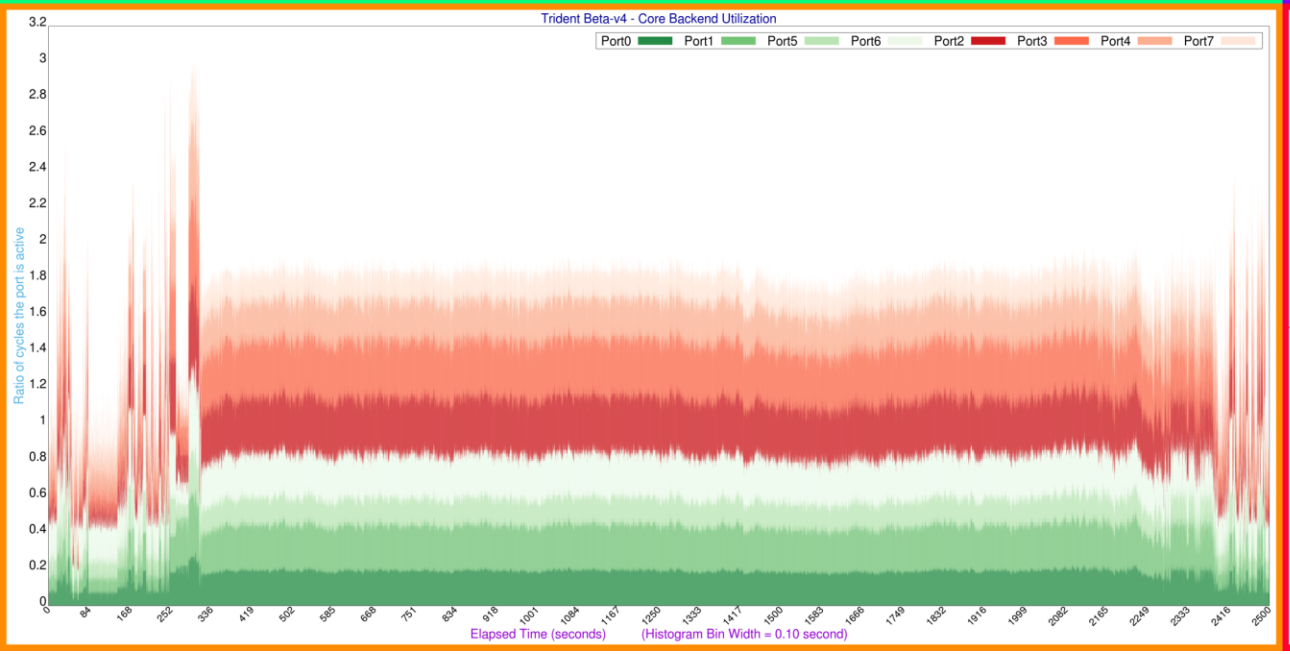


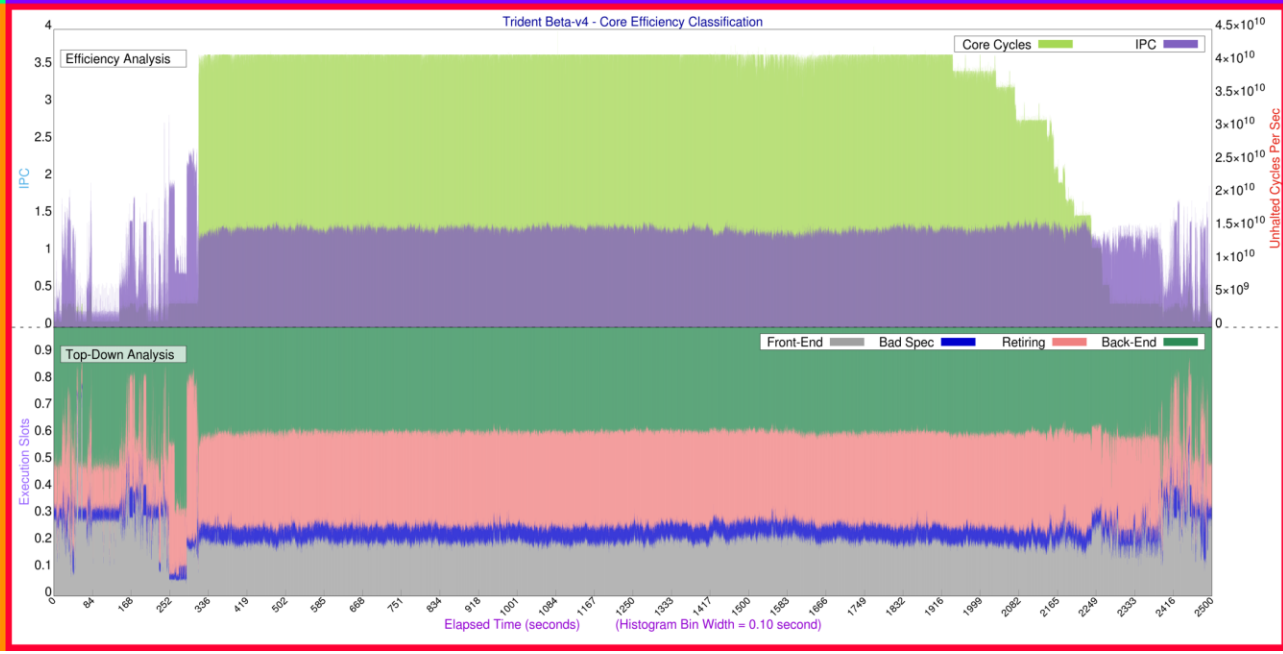


ATLAS – MC Simulation

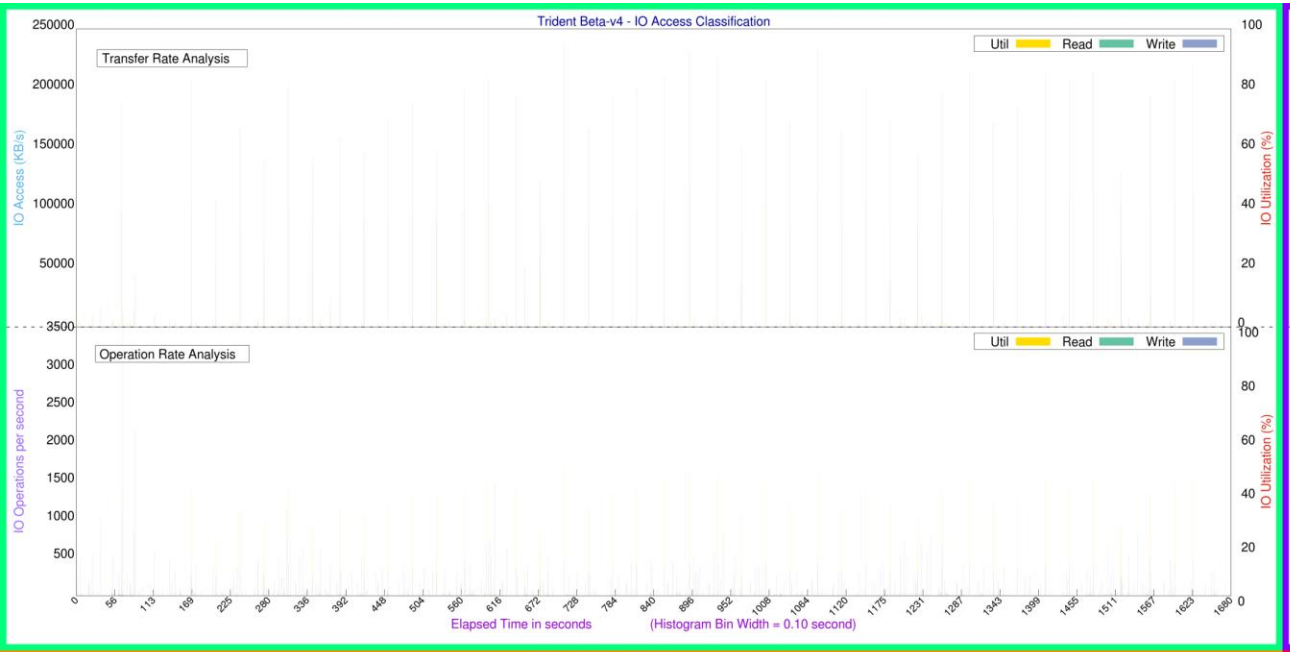


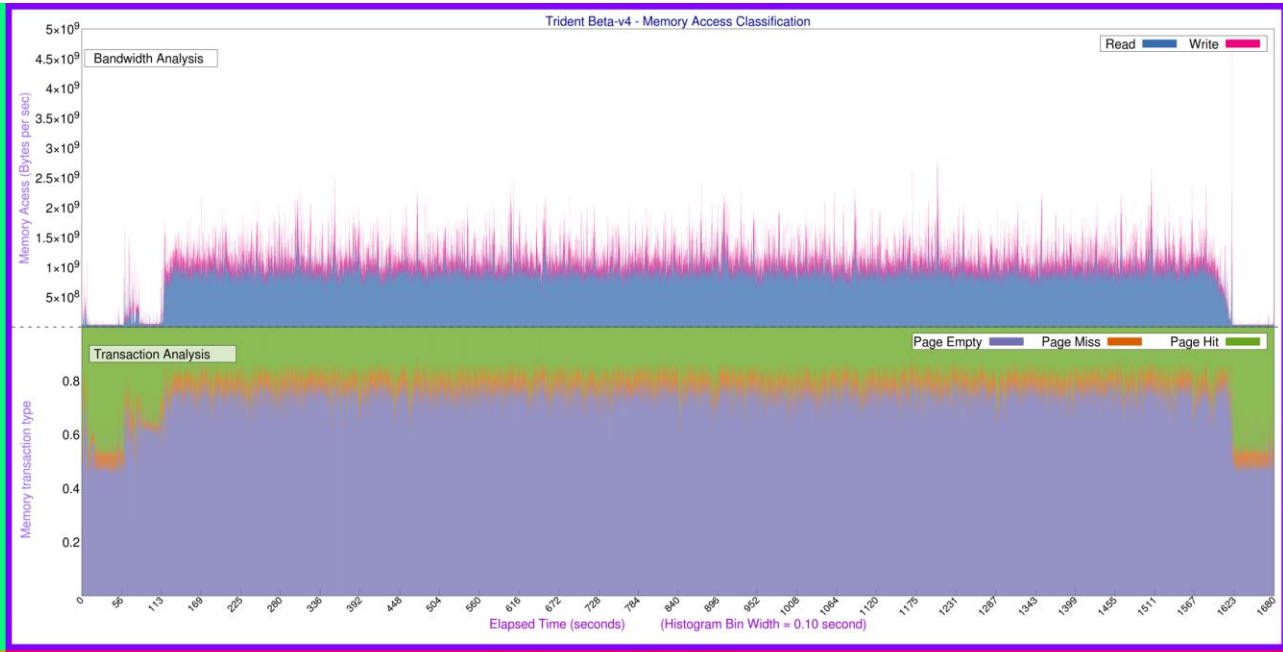


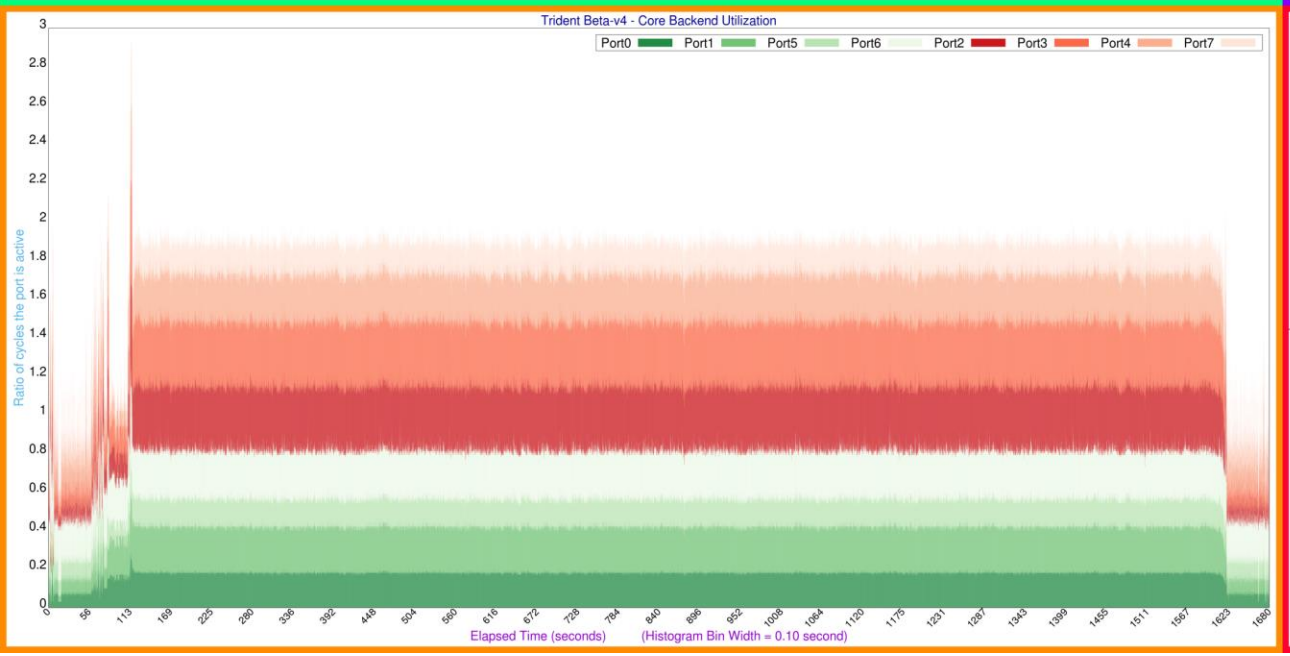


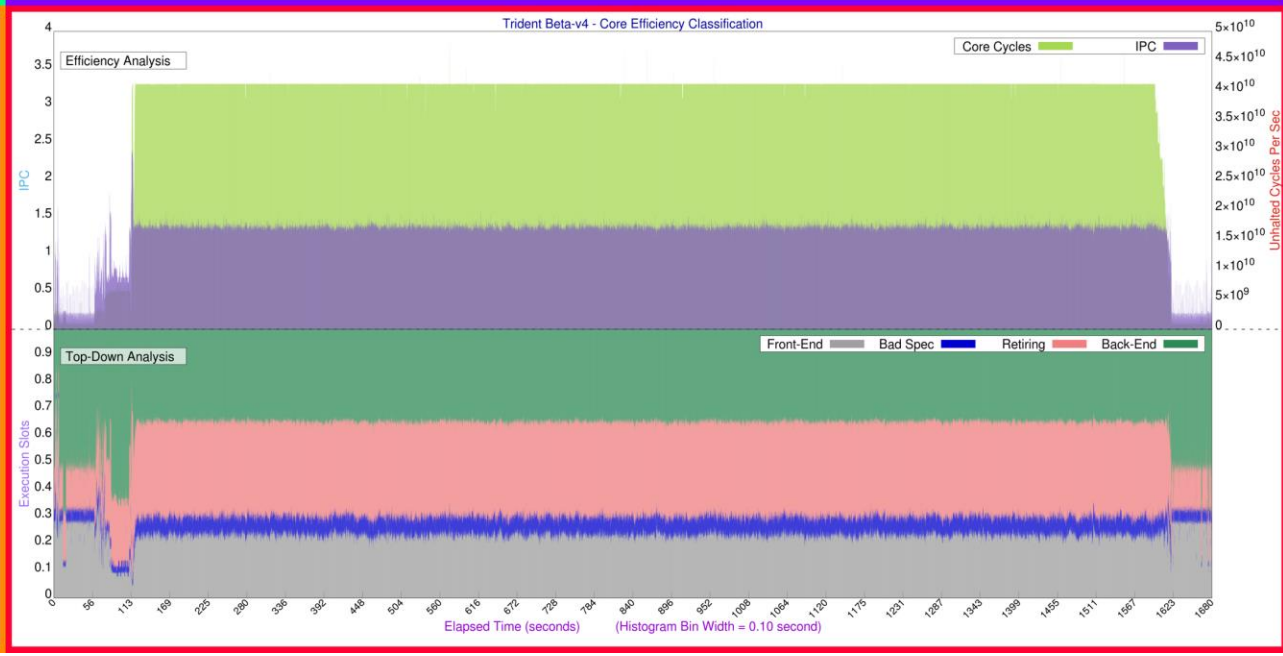


CMS – Generation + Simulation

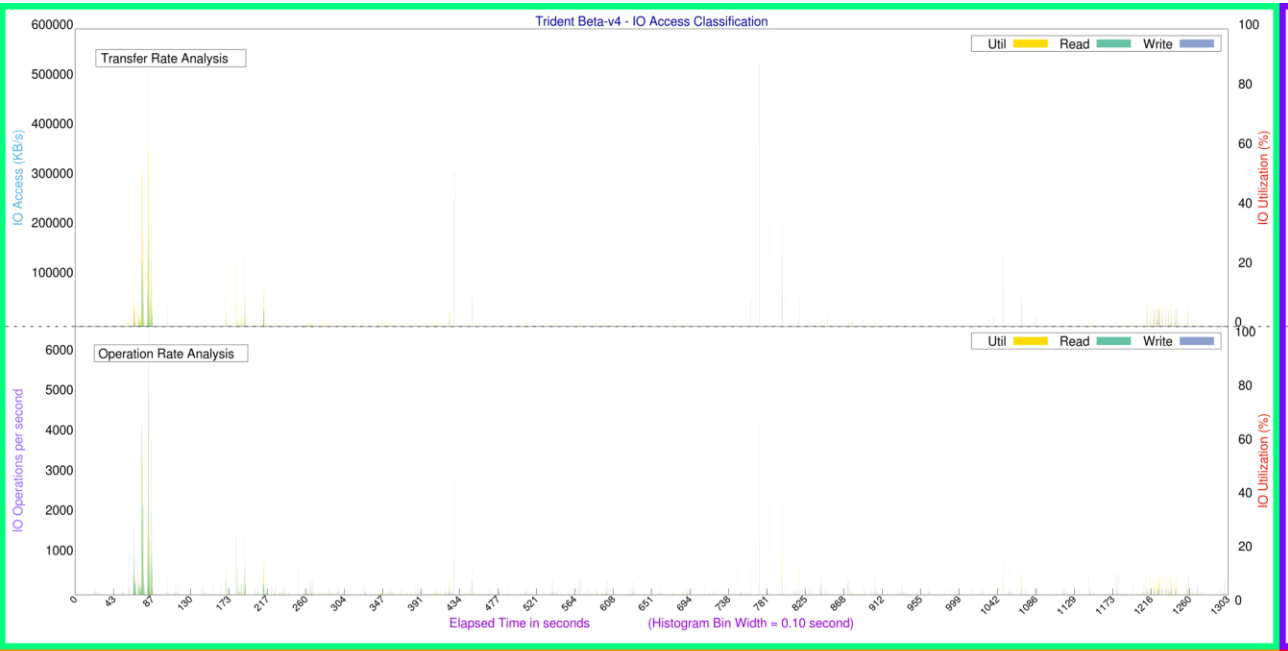


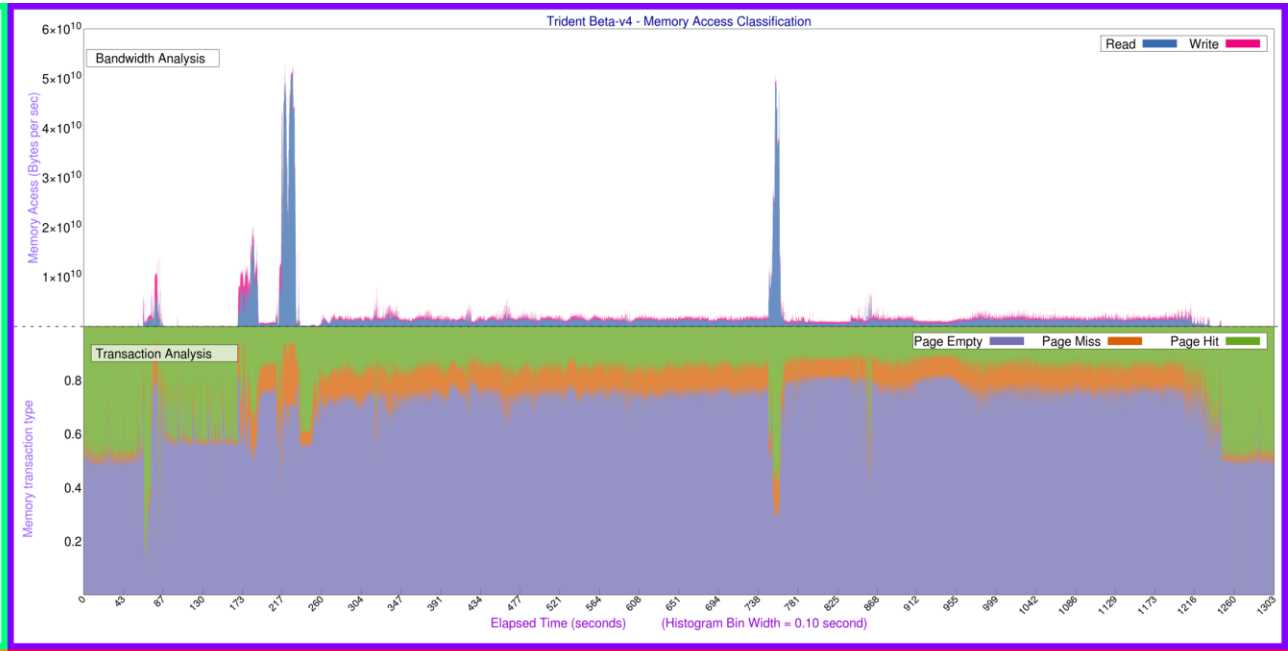




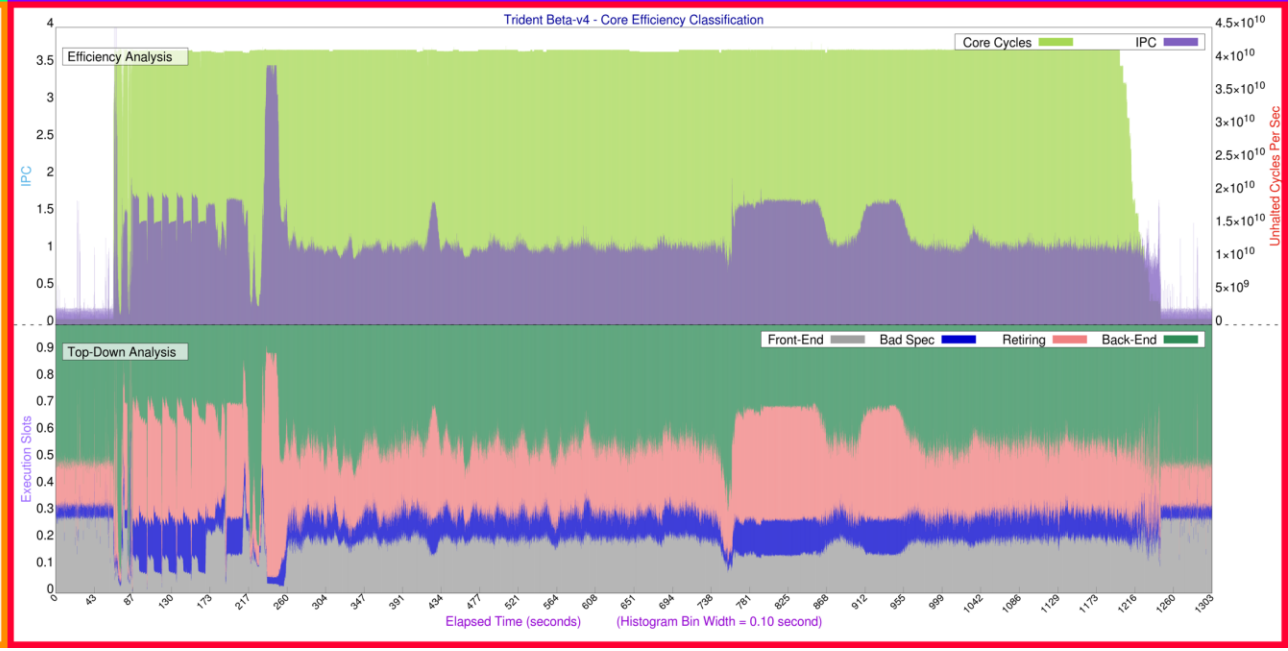


LHCb – Gen+Sim

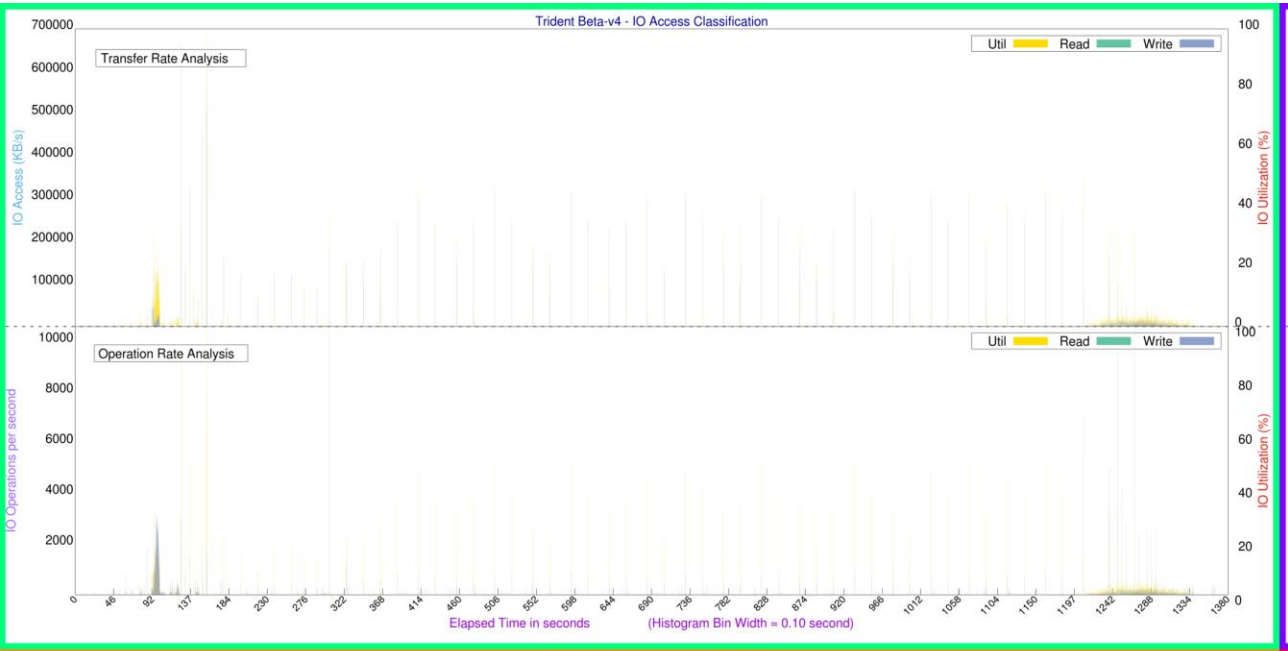

Trident Beta-v4 - IO Access Classification
Transfer Rate Analysis
Util Read Write
IO Access (KB/s)
IO Utilization (%)
Operation Rate Analysis
IO Operations per second
Elapsed Time in seconds
(Histogram Bin Width = 0.10 second)

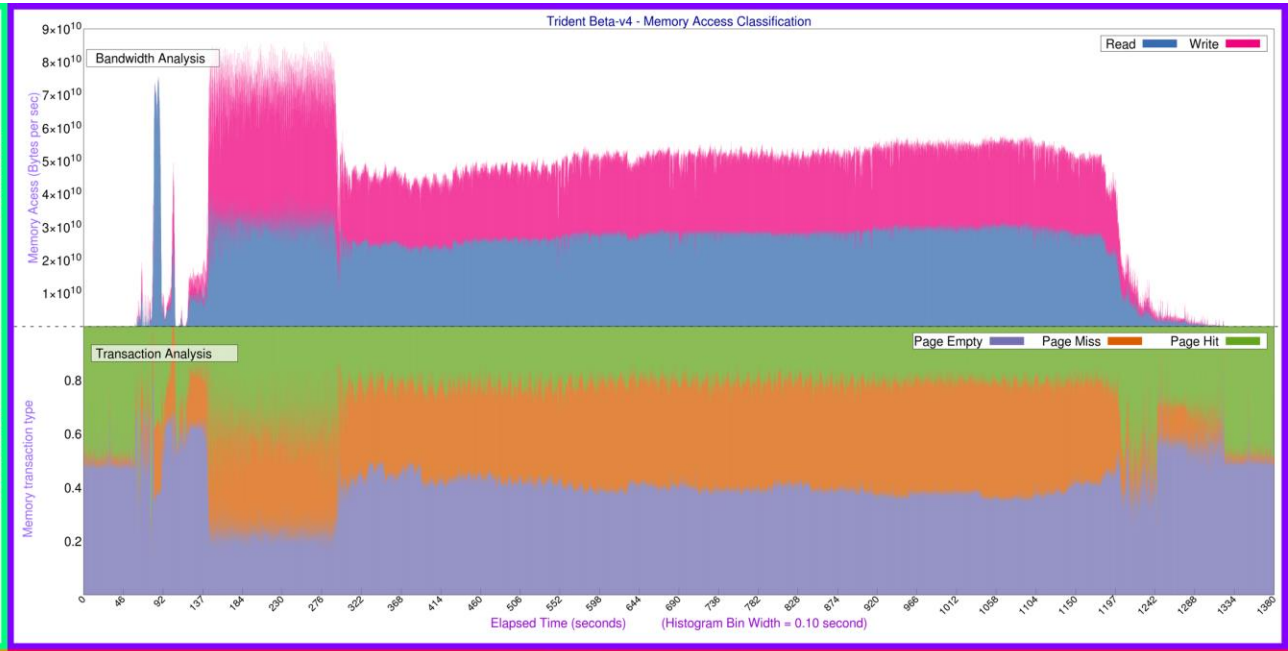

Trident Beta-v4 - Memory Access Classification
Bandwidth Analysis
Read Write
Memory Acess (Bytes per sec)
Transaction Analysis
Page Empty Page Miss Page Hit
Memory transaction type
Elapsed Time (seconds)
(Histogram Bin Width = 0.10 second)

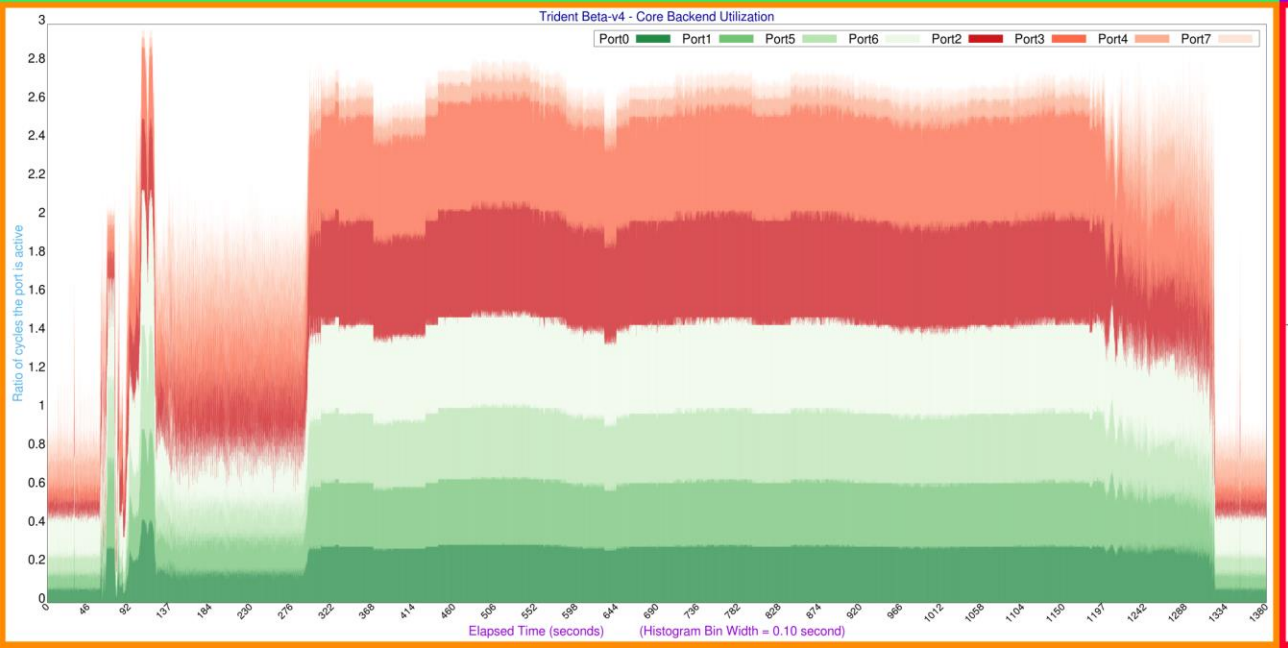

Trident Beta-v4 - Core Backend Utilization
Port0 Port1 Port5 Port6 Port2 Port3 Port4 Port7
Ratio of cycles the port is active
Elapsed Time (seconds)
(Histogram Bin Width = 0.10 second)

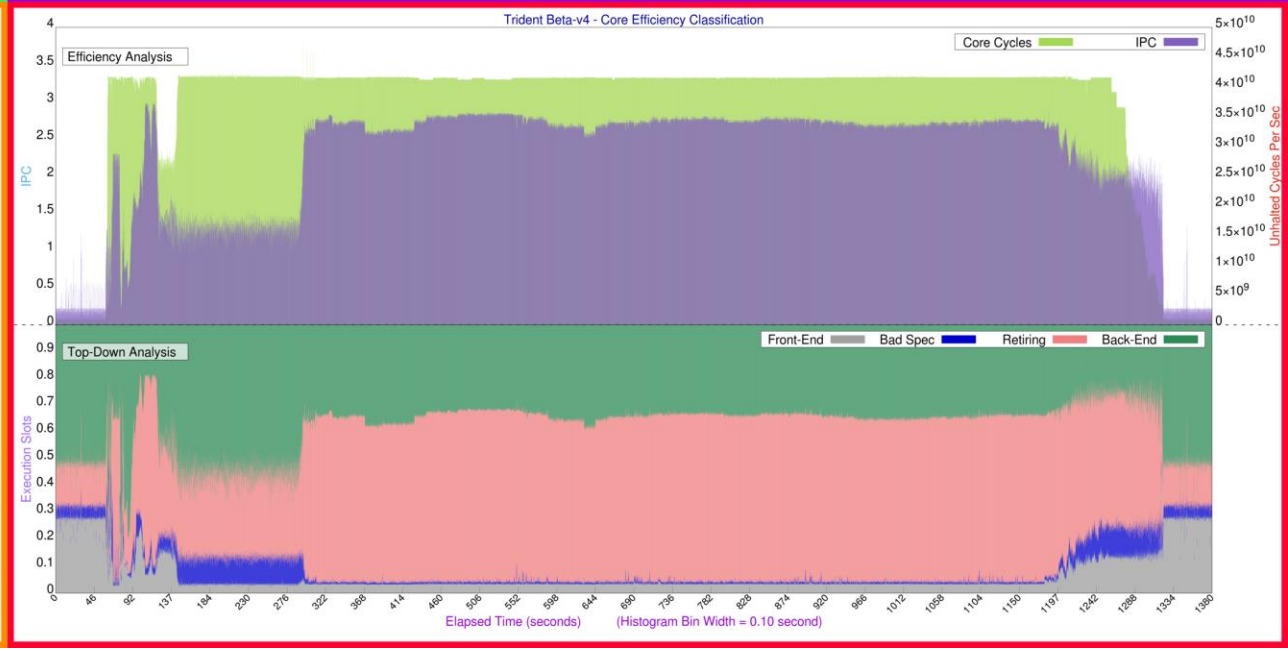

Trident Beta-v4 - Core Efficiency Classification
Efficiency Analysis
Core Cycles IPC
IPC
Unhalted Cycles Per Sec
Top-Down Analysis
Front-End Bad Spec Retiring Back-End
Execution Slots
Elapsed Time (seconds)
(Histogram Bin Width = 0.10 second)

ALICE – Gen+Sim



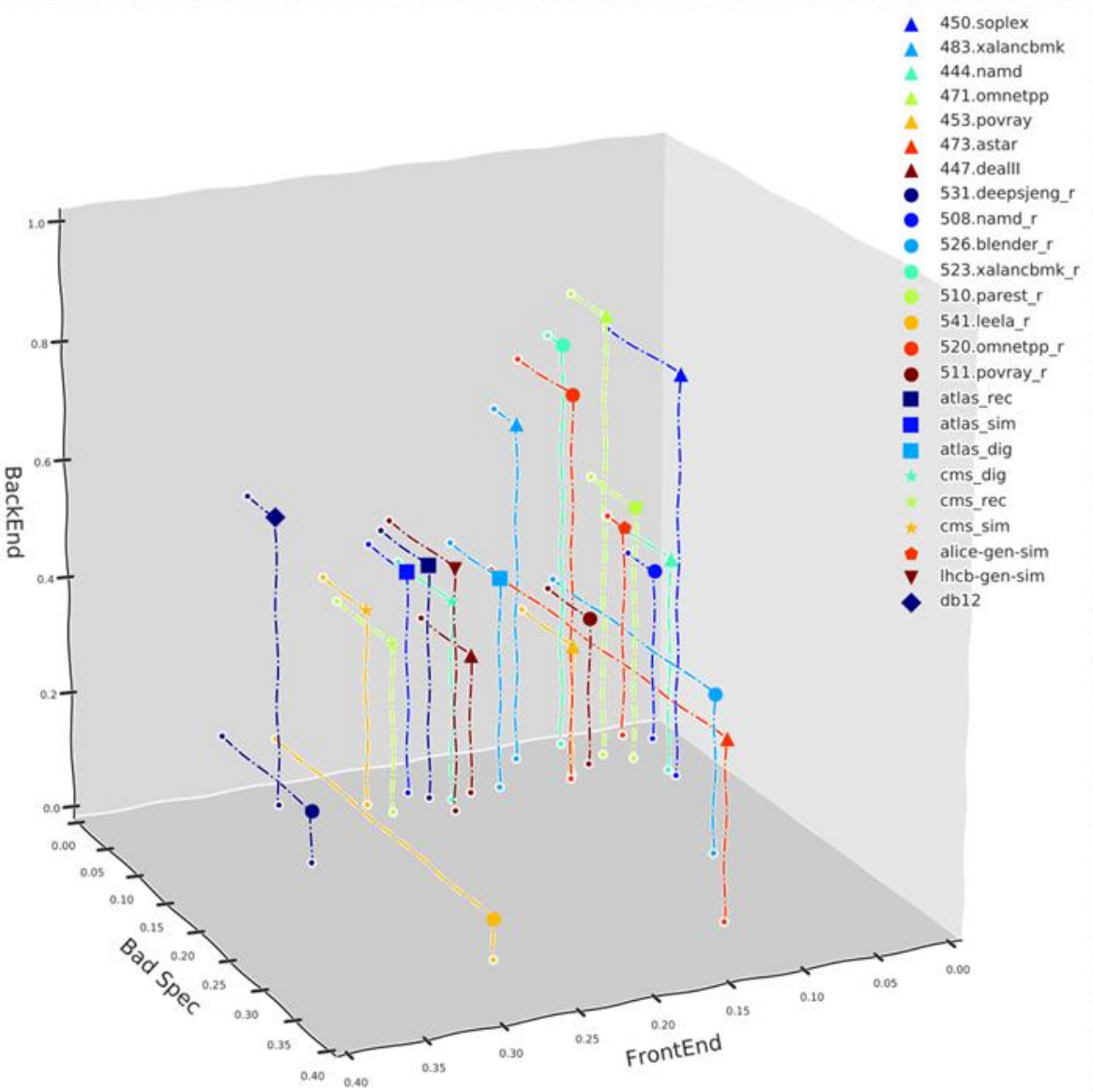






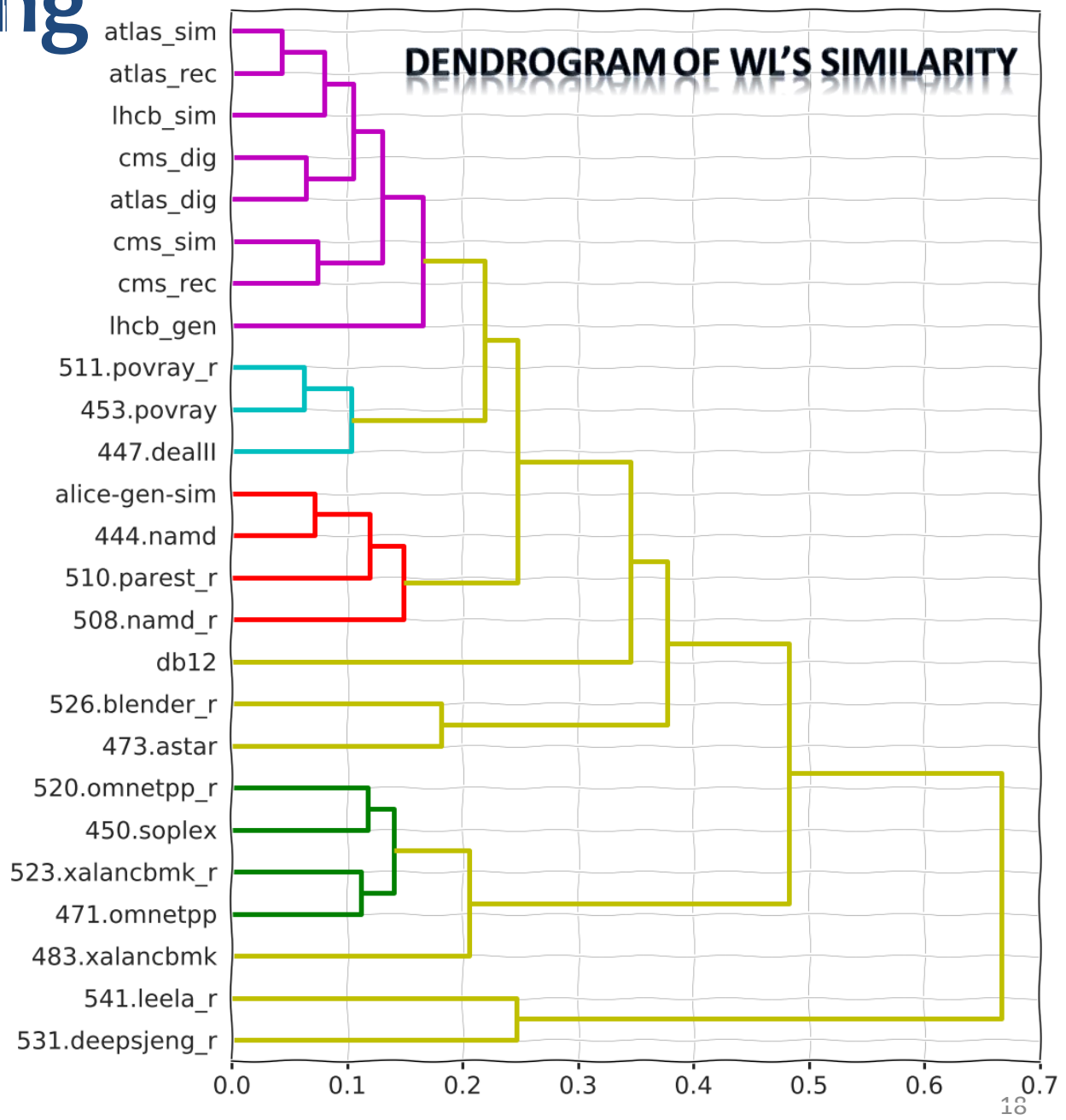
Quantitative comparison

- Used Trident to quantitatively compare how the CPU is used by LHC applications vs. synthetic benchmarks
  - Percentage of time spent in front-end, vs back-end, vs bad speculation
    - Used as coordinates
    - Retired not appearing since it is complement to 1 of the other 3 coordinates
    - Measured “distance” in 3D space

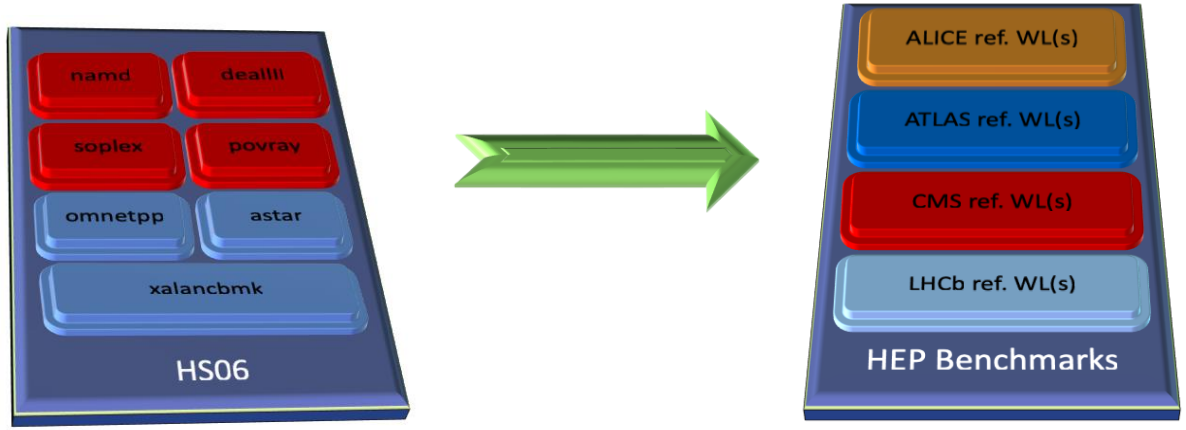


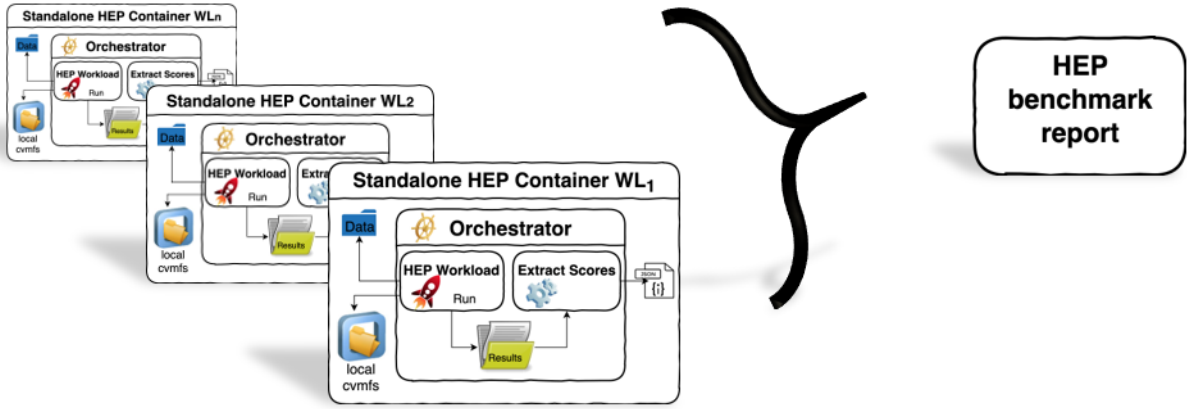
Hierarchical clustering

- All LHC applications clustered together (apart from ALICE gen-sim, due to Geant3 ?)
- Rather far from clusters consisting in HS06 (4xx) or SPEC CPU 2017 (5xx) benchmarks
- Strong argument in favour of building an LHC benchmark using LHC applications



- Requirements
  - No need for network connectivity
  - Not too large package
  - Easy to use
  - Reproducible results
- Ingredients
  - SW repository (CVMFS)
  - Input data (ROOT files and conditions DB dumps)
  - An orchestrator script per workload
    - Sets the environment
    - Runs the application
    - Parses the output
- Packaging
  - Decided to go for Docker containers
    - Also work with Singularity – thanks to Chris Hollowell
    - Very easy to use
  - Containing also the needed SW from CVMFS and input data





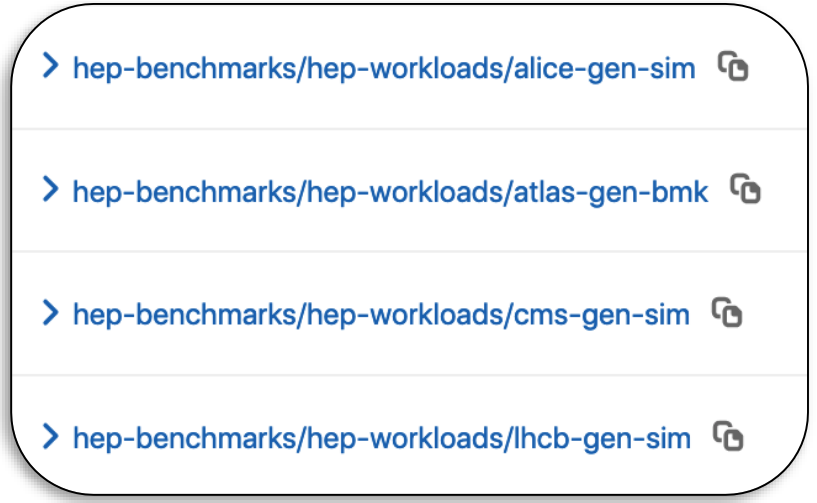
- All (GEN-)SIM workloads available as containers
  - https://gitlab.cern.ch/hep-benchmarks/hep-workloads
  - https://twiki.cern.ch/twiki/bin/view/HEPIX/HEP-Workloads
- Working now on DIGI and RECO
  - Will also include a ROOT analysis job
    - Useful to select hardware for analysis workloads
- Support from experiment experts
  - On containerizing the workloads
  - On how to best extract a score
  - Debugging the workload

Experiment contact person

Exp	name	email
ALICE	Costin Grigoras	grigoras@cern.ch
ATLAS	Lorenzo Rinaldi	Lorenzo.Rinaldi@bo.infn.it
CMS	David Lange	David.Lange@cern.ch
LHCb	Andrea Valassi	andrea.valassi@cern.ch

Advantage of standalone containers

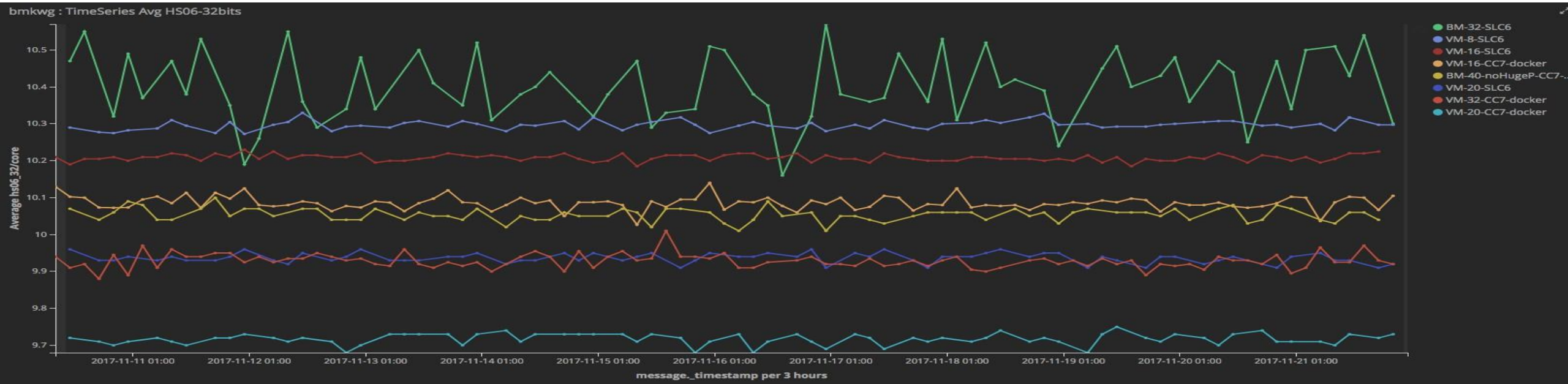
- Usability:
  - Simple instructions: *Insert disk, run shell script, wait, and read and report score*
    - `docker run --rm -v /tmp/results:/results $IMAGE [args]`
    - `singularity run -B /tmp/results:/results docker:://$IMAGE [args]`
- Accessibility:
  - No need for remote data access
    - With containers the benchmark can be distributed as full tarball in a drive
- Free License:
  - Follow the experiment code license
- Long term support
  - Containers are versioned (tagged), reference software is also tagged in cvmfs and/or git



https://gitlab.cern.ch/hep-benchmarks/hep-workloads/container_registry

Debugging and validation

- Benchmark results must be reproducible
  - They should not vary more that 1-2%
  - Performing benchmark validation (repeated ≥ 20 times per workload, 1 task per each logical processor in parallel)
  - Sometimes we notice a spread not related to the number of events
  - Experiments experts and site admins are working on improvements
    - Twiki: https://twiki.cern.ch/twiki/bin/view/HEPIX/HEPWorkloadReproducibility



Status of the activity

Very good progress since the last report

- Improvement in the gitlab-ci infrastructure
  - it takes care of the automation of the container builder
  - it eases the validation of changes via gitlab CI pipelines
  - A. Valassi is heavily contributing here!
- Increased participation of experiment experts
  - Contact persons, field experts, beta testers, …

Plans

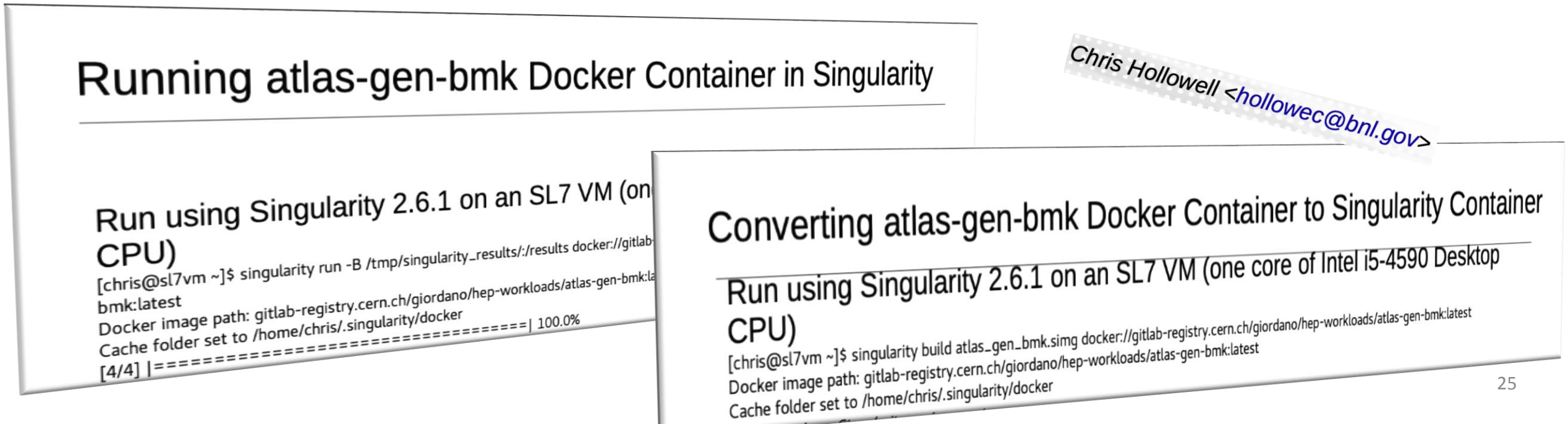
- Finalize the Continuous Integration
  - Deploy a gitlab runner to run the full build procedure
    - Implies running a CPU intensive HEP workload for some time
      - We prefer to run in a separate gitlab runner respect to the CERN gitlab central runners
  - Add validation test
- Integration with the benchmarking suite
  - When a HEP workload container is available, integrate it in the benchmarking suite
  - Start collecting individual results from resources under benchmarking
    - Together with HS06, SPEC CPU 2017, DB12,…
- Increase the participation of experts from the Experiments
  - Keep reporting to the Benchmarking WG and Cost Model WG

Bring the experiments know-how in the WG

- We have asked 1 expert per experiment to work on 2 areas
  - Containerisation of the remaining experiment workloads
  - Definition/validation of the metrics extracted from the experiment job, and used to build a score
- The active participation of experiment experts contributed to the integration of all the workloads and their validation
  - Example: the need to remove Frontier from the jobs of CMS and ATLAS and replace sqlite files.
  - Those problems have been solved recently
  - Now the situation is progressing much better than few months ago

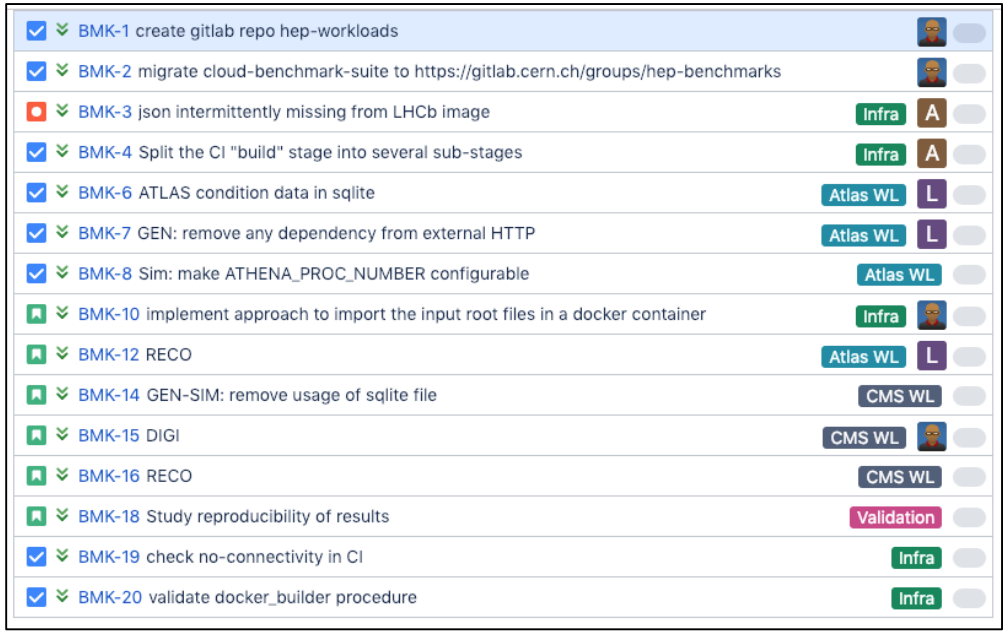
Not only the docker container

- The approach is not bound to a single “container” technology
  - Decouple the benchmark workload from the packaging and distribution
- Successfully tested adoption of Singularity containers
  - Same performance score obtained as with Docker containers
    - When the Singularity image is generated and used score is ~2% higher



Consolidating the project management

- Dedicated Jira project
- New Gitlab group hep-benchmarks
  - Hosting the benchmarking related projects
    - hep-workloads
    - benchmarking suite
      - Toolkit to run all available benchmarks and simply collect, track and share results.
  - Migration from previous repos completed
- HEPiX Bmk WG twiki to summarise the status



#	Description	Status
1	Implement a fully automated procedure to build a standalone container image for each HEP reference workloads	✅
2	Create containers starting from Experiments' recipes	✅: GEN-SIM 🔶: DIGI-RECO
3	Implement Gitlab Continuous Integration approach for long term maintainability (see https://gitlab.cern.ch/giordano/hep-workloads/pipelines)	✅
4	Consolidate the CI approach	🔶
5	Integration in the benchmarking suite	🔶
6	Test migration to singularity containers	✅
7	Migrate gitlab repository to https://gitlab.cern.ch/hep-benchmarks	🔶

Reference

- All LHC experiments provided and maintain instructions to manually submit jobs representative of the main workloads
  - Simulation, digitisation, reconstruction, …
  - But analysis still missing
- The initial goal was to have fixed references to be used for performance studies
  - They became natural candidates for a HEP benchmark
- Instructions are currently in the gitlab repo in the orchestrator scripts such as:
  - https://gitlab.cern.ch/hep-benchmarks/hep-workloads/blob/qa/alice/gen-sim/alice-dpgsim-bmk/alice-dpgsim-bmk.sh
  - https://gitlab.cern.ch/hep-benchmarks/hep-workloads/blob/qa/atlas/sim/atlas-sim/atlas-sim-bmk.sh
  - https://gitlab.cern.ch/hep-benchmarks/hep-workloads/blob/qa/cms/gen-sim/cms-gen-sim/cms-gen-sim.sh
  - https://gitlab.cern.ch/hep-benchmarks/hep-workloads/blob/qa/lhcb/gen-sim/lhcb-bmk/lhcb-bmk.sh

Open Access

- The experiments must provide data files to be inserted in the containers
  - E.g. Snapshot of the conditions database to run GEN and SIM
  - Minimum bias event in the DIGI benchmark
- These files should be OPEN ACCESS if we need to distribute the benchmark outside HEPiX/WLCG
  - E.g. to the vendors that want to participate to a tender for the procurements of future worker nodes
- This is a political issue, not a technical one

Next Steps

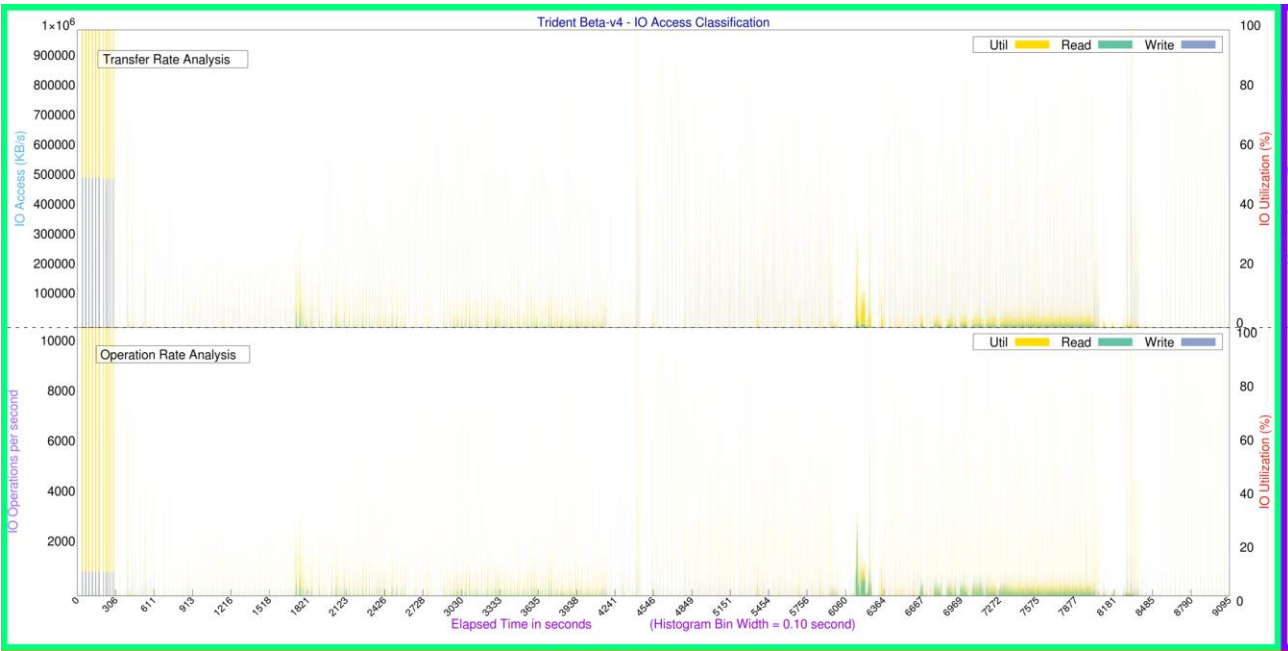
- Finalize debugging and validation
- Add remaining workloads
- Finalize the score calculations
- Start using for real-world benchmarking
- Agree on a (single) total performance score (e.g. a weighted average of the individual workload scores)
- Include GPU workloads (reconstruction algorithms and machine learning algorithms)
  - There are already contacts with experiments
- Integrate all those benchmark in the bmk suite that automate the running and collection

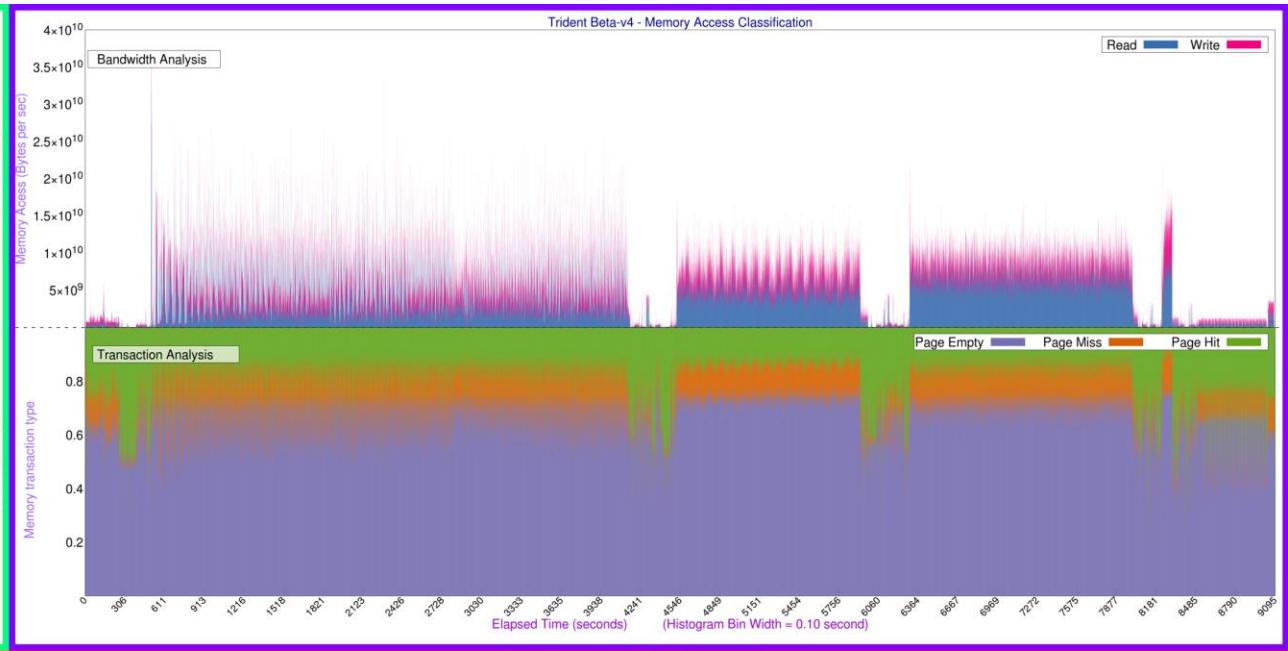
Conclusions

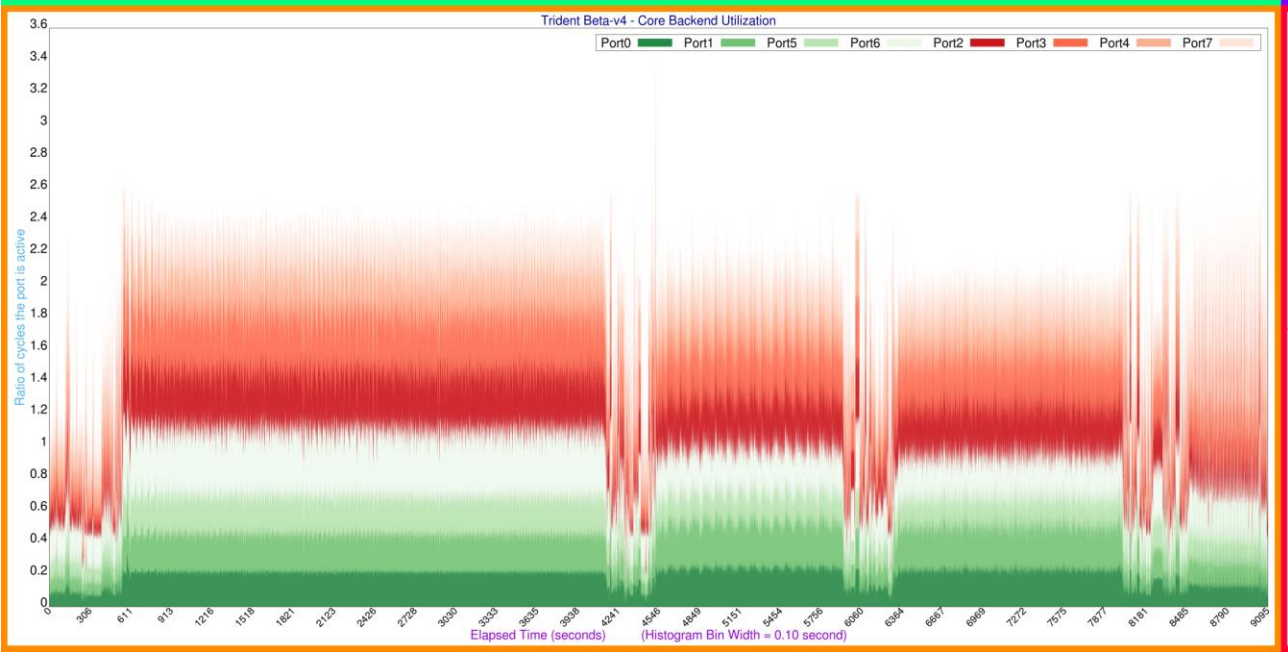
- Common effort between **performance modelling and cost model** working group, and the **HEPiX benchmarking** working group
- From simple performance studies to a working and already usable LHC benchmarking suite
- We welcome new participants with new ideas or new processor to benchmark

BACKUP SLIDES

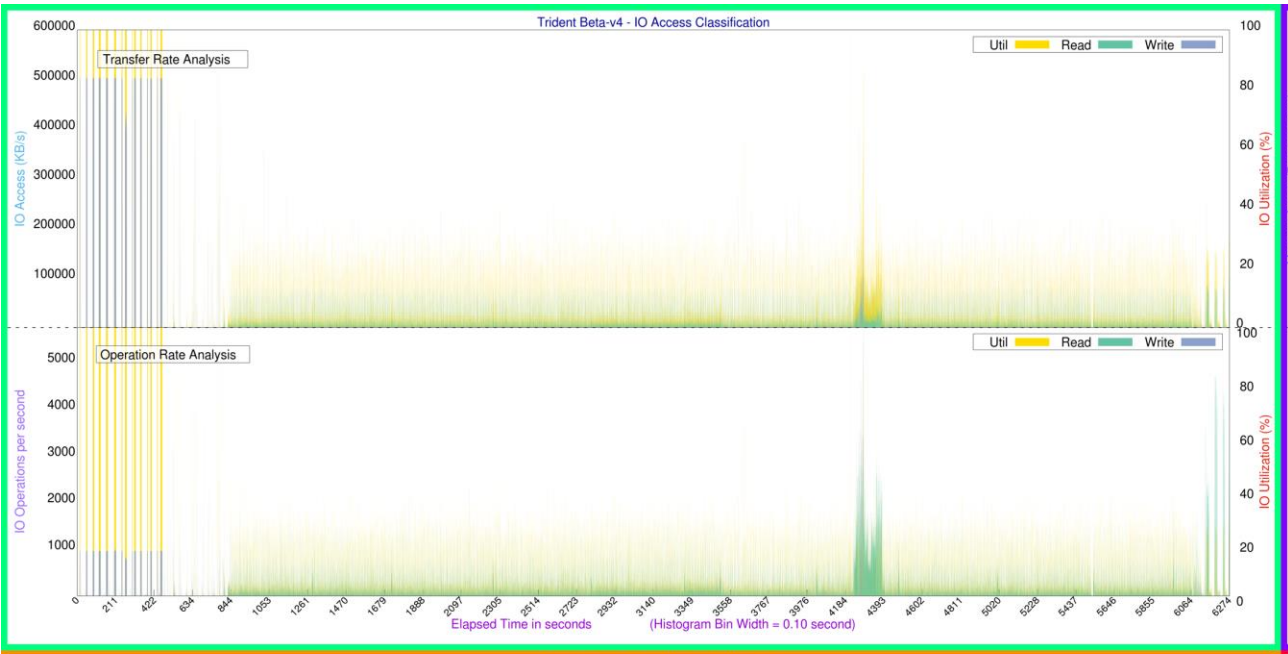
ATLAS – MC Digi + Reco

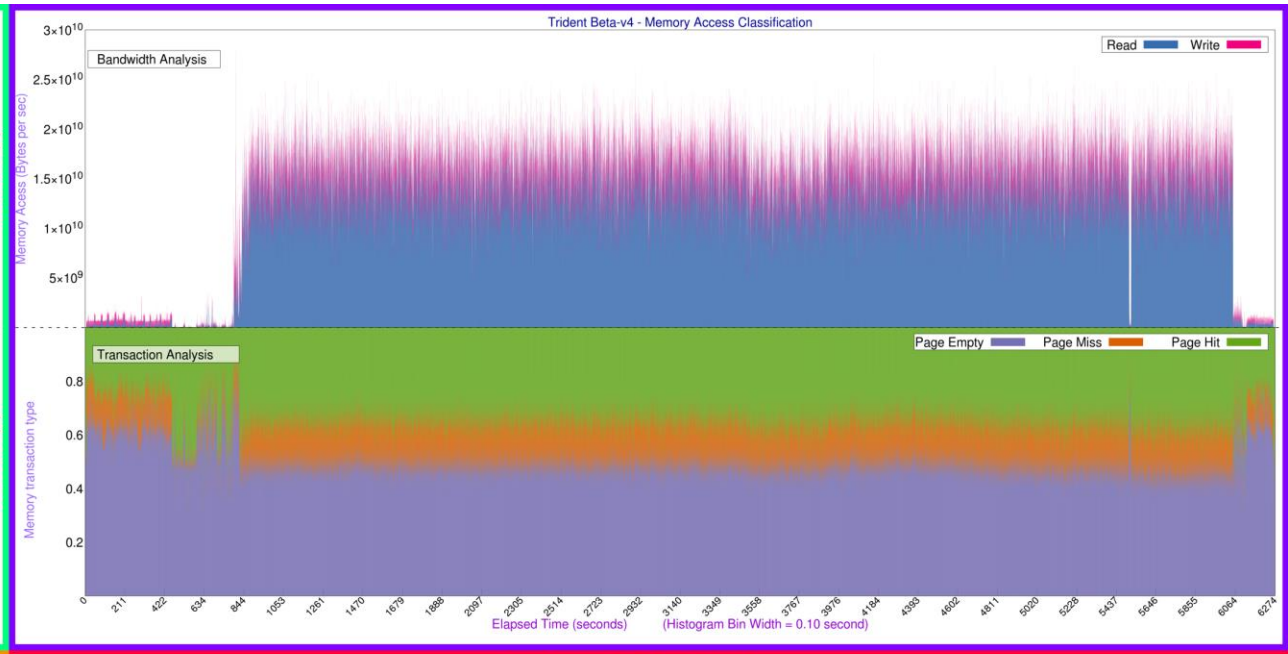


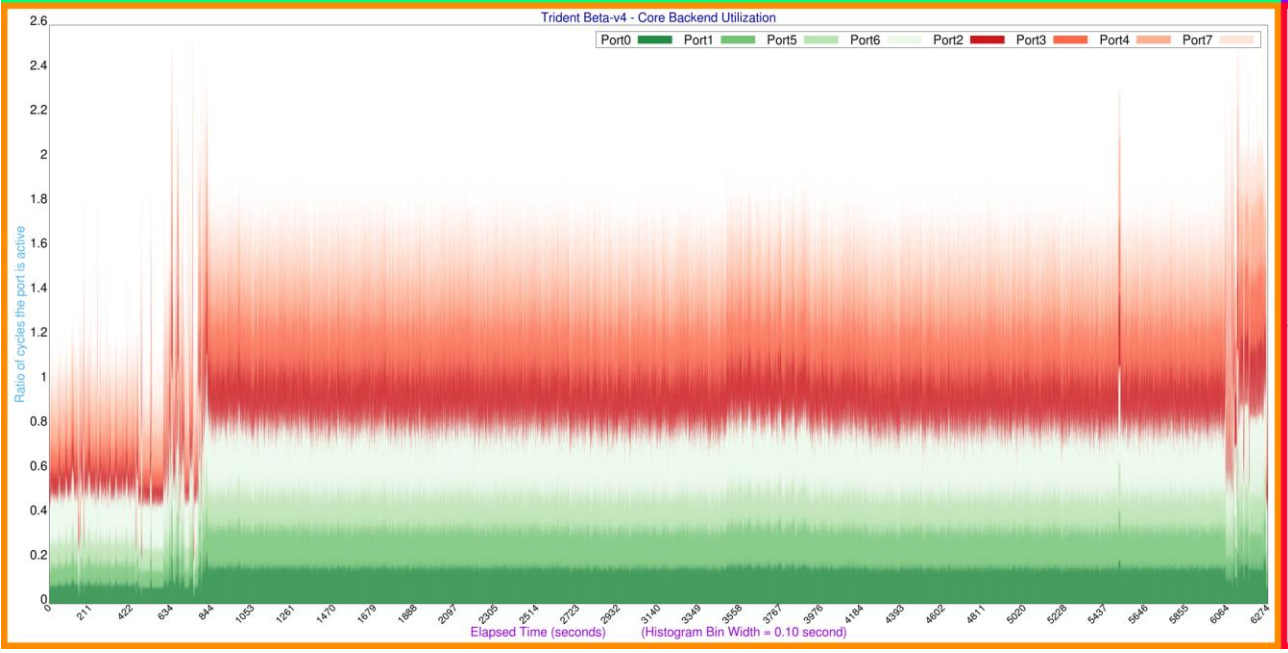


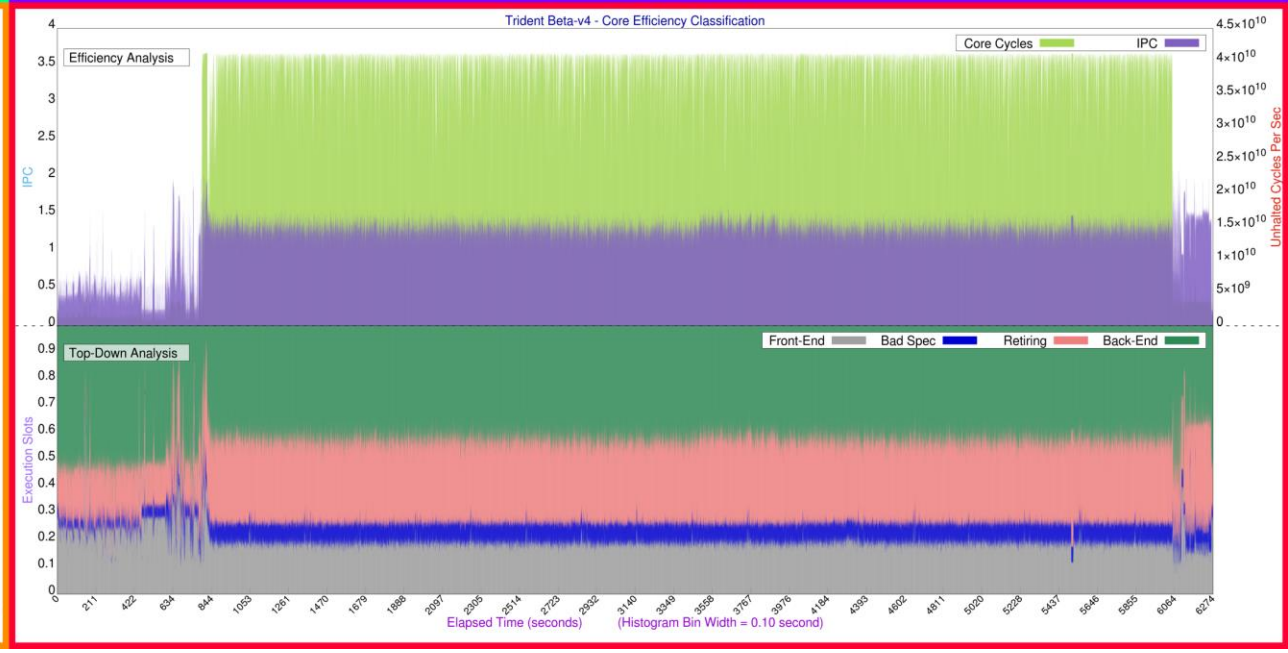


ATLAS – Derivation Production

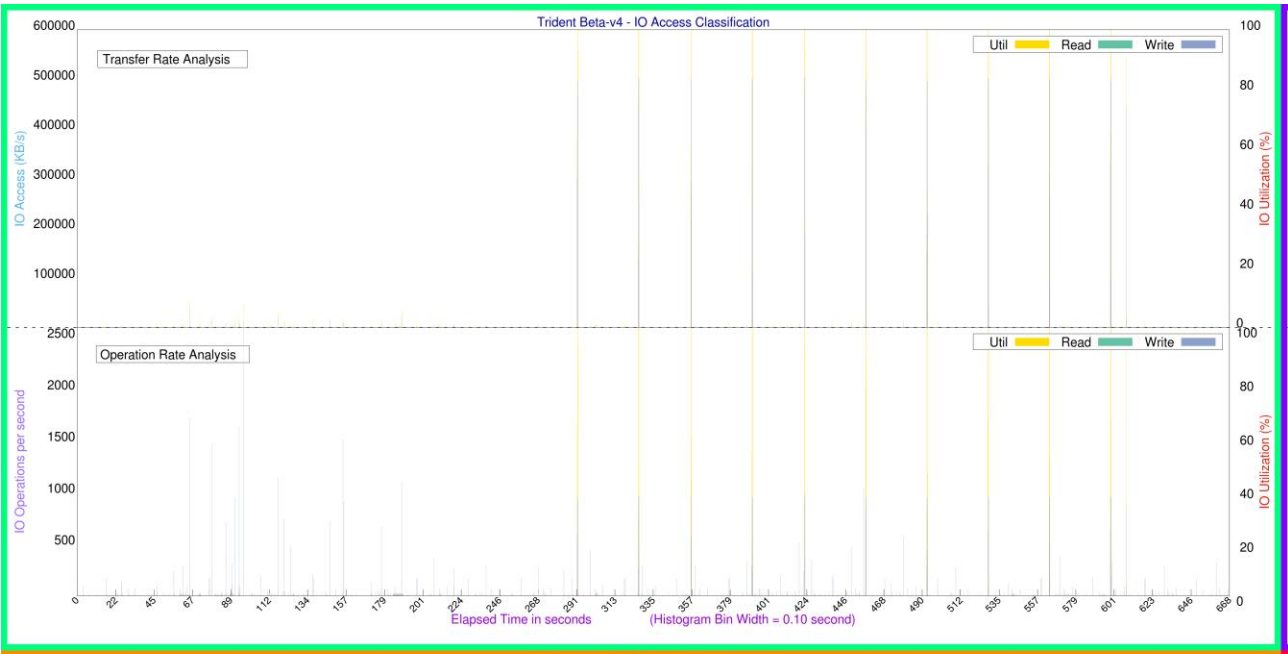


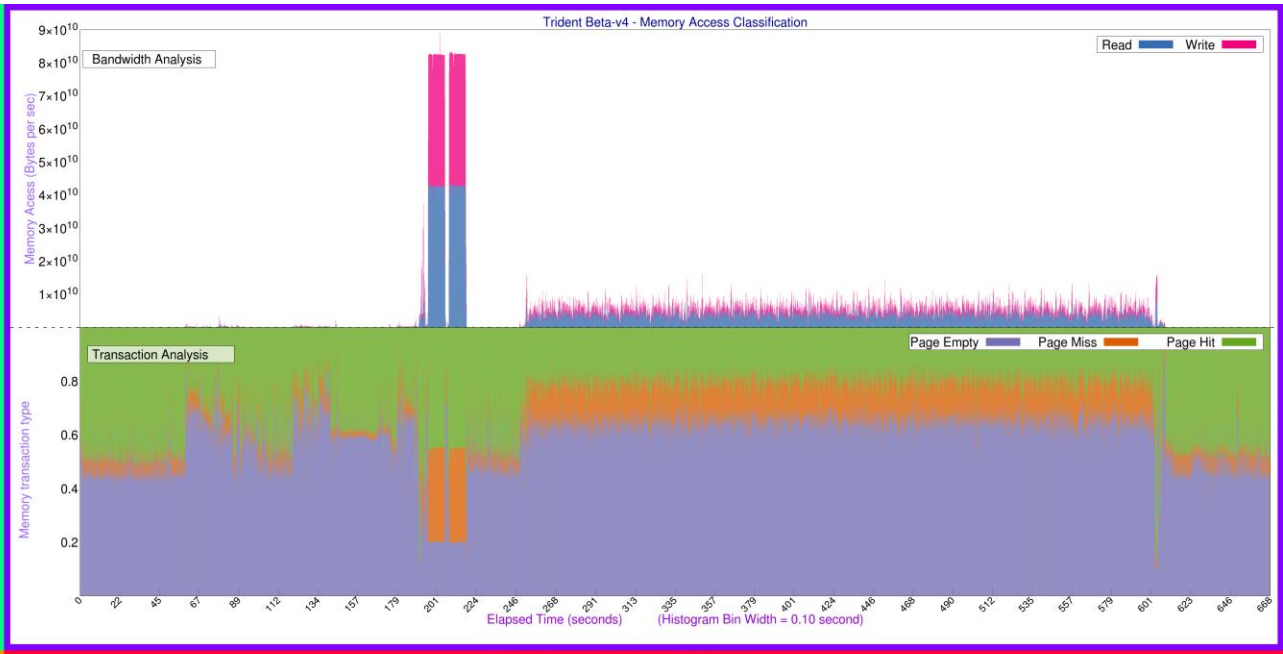


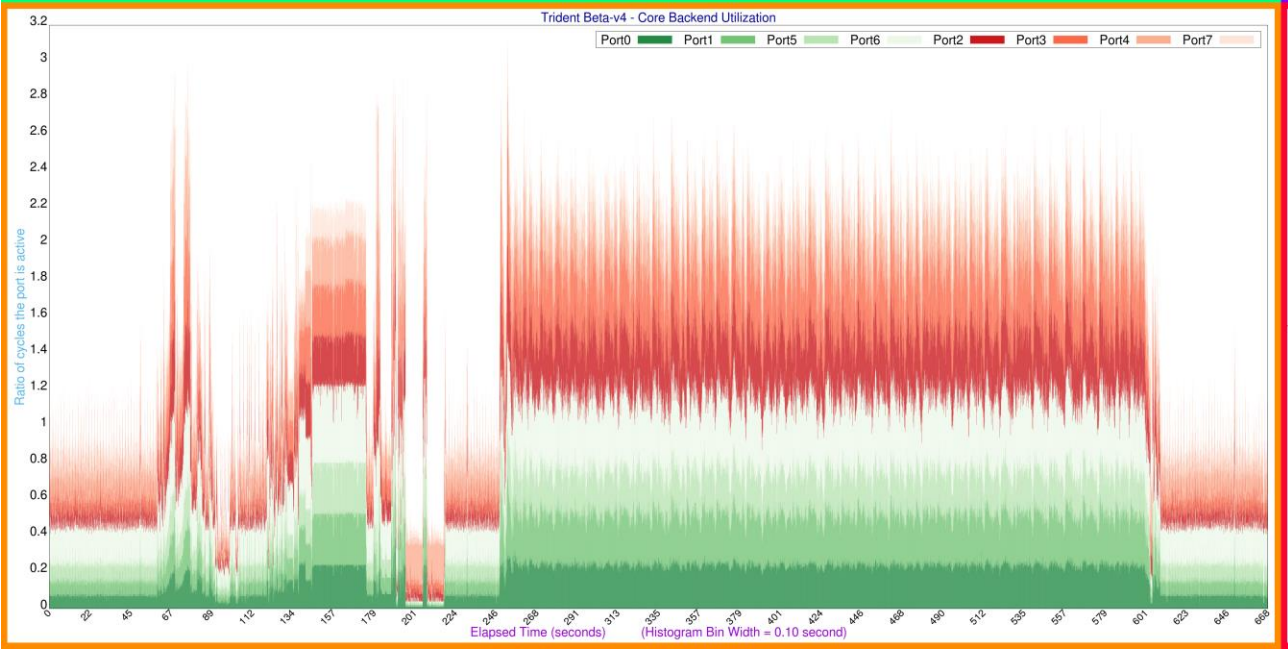


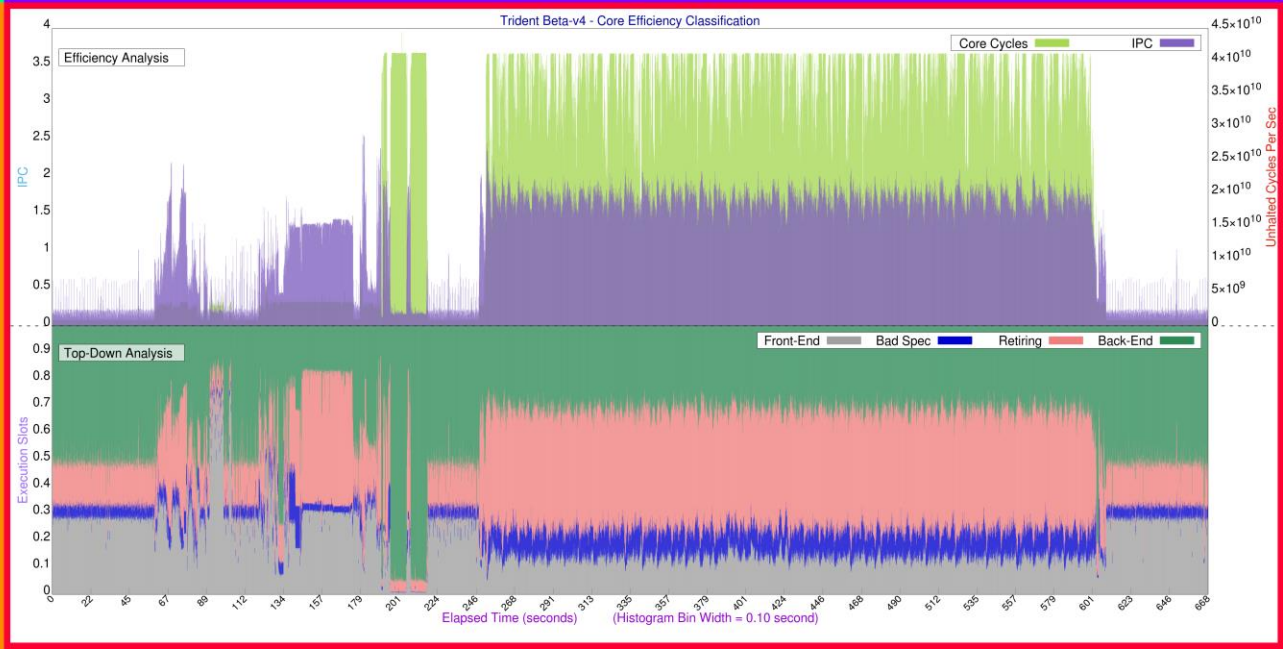


CMS – Digitization + Trigger + Pileup Simulation









CMS – Reconstruction + Analysis Data Creation