Update on: Estimates of **potential** Efficiency/Cost Gains

Markus Schulz

Overview

- What has been done already in the past
  - Quick reminder
- What we learned after the last workshop

Background

- In collaboration with experiments and sites **we** have been looking at the efficiency of:
  - Infrastructure, Software, Experiment workloads , Failure rates
  - Scheduling, Operational effort based on the WLCG site survey, Monitoring data….
- Looked at non HEP scientific code bases
  - Instructions per cycle, level of vectorisation
- Work closely with the Benchmarking Working Group
- Work closely with the DOMA working groups

Background

- Found several potential improvements
  - Compiler based
    - Vectorisation , building for specific architectures, FDO (feedback directed optimisation)
  - Linker/loader strategies
  - Analysis based on hardware counters
  - Code profiles + instrumentation
- Tried to estimate the quantitative impact of changes to storage organisation
  - Centralisation
  - Reduced redundancy within storage systems
  - Reduced number of replication between sites
- Tried to classify the effort needed to implement improvements
  - Quantitatively very difficult
    - Some experience is gained by LHCb and ALICE preparing for Run3

Summary of what we had last year:

Change	Effort Sites	Effort Users	Gain
managed storage on 15 sites + caches	Some on large sites/gain on small sites	little	40% decrease in operations effort for storage
Reduced data redundancy	Some large sites	Frameworks some	30-50% disk costs
Reduced data replication and cold data	little	Frameworks some	15% disk costs
Scheduling and site inefficiencies	Some	Some	10-20% gain CPU
Reduced job failure rates	Little	Some-Massive	5-10% CPU
Compiler and build improvements	None	Little-Some	15-20% CPU
Improved memory access/management	None	Realistic	10%-15% CPU
Exploiting modern CPU architectures	None	Massive	100% CPU
Paradigm shift algorithms	Some	Massive-Infinite	Factor 2-100 CPU
Paradigm shift online/offline data	Little	Massive-Infinite	2-10 CPU 10-20 Storage

What has changed

- More studies on caches and remote access
  - By us and by DOMA
- More studies on data popularity
  - Several, by several teams
- Better measurements of the reference workloads
- Better understanding of the TCO cost of 1HS06 and 1 TByte disk/a
- Studies on data loss and re-population
- DomaAccess work on a Strawman for HL-LHC analysis
- Work by the experiments towards more compact and orthogonal analysis data formats
  - CMS, ATLAS recently
- Activities inside experiments started to try some of the ideas
  - Compiler based…..

What Changed

- Better understanding of the estimates
  - Most are still OK, but better understood
- Cache, latency hiding and data access frequency activities indicate that estimates can be more aggressive
  - See DOMA work
    - If we take some of the measurements seriously WLCG could work with 20% of the storage at T2s (with some adjustments …)
  - Especially replacing local replica with remote replica to compensate for data loss
    - Spreadsheet to explore different scenarios
    - Since bulk of disk data is used not very frequently the impact of losing data and replicating it back is minimal (>> 0.01% of jobs are affected)
- Compact data formats for analysis have a massive impact

Summary of how this evolved:

Change	Effort Sites	Effort Users	Gain
managed storage on 15 sites + caches elsewhere	Some on large sites/gain on small sites	little	40% decrease in operations effort for storage
Caches at most sites (dataLake strawman)	Some everywhere	Frameworks some	15% of storage
Reduced data redundancy	Some large sites	Frameworks some	30-50% disk costs
Reduced data replication and cold data	little	Frameworks some	30% disk costs
Compact data formats for analysis	none	Some	>15% disk costs
Scheduling and site inefficiencies	Some	Some	10-20% gain CPU
Reduced job failure rates	Little	Some-Massive	5-10% CPU
Compiler and build improvements	None	Little-Some	15-20% CPU
Improved memory access/management	None	Realistic	10%-15% CPU
Exploiting modern CPU architectures	None	Massive	100% CPU
Paradigm shift algorithms	Some	Massive-Infinite	Factor 2-100 CPU
Paradigm shift online/offline data	Little	Massive-Infinite	2-10 CPU 10-20 Storage

Summary:

- More, deeper, work is going on
- Savings on the “how we run things” level become more concrete

- Even with reduced estimates for future technology gains we moved somewhat closer to the goal
- However: Software+Paradigm changes are still the place where we can gain most
  - Generators, Detector MC, Reconstruction…….
- Detailed update soon in the Working Group Meeting

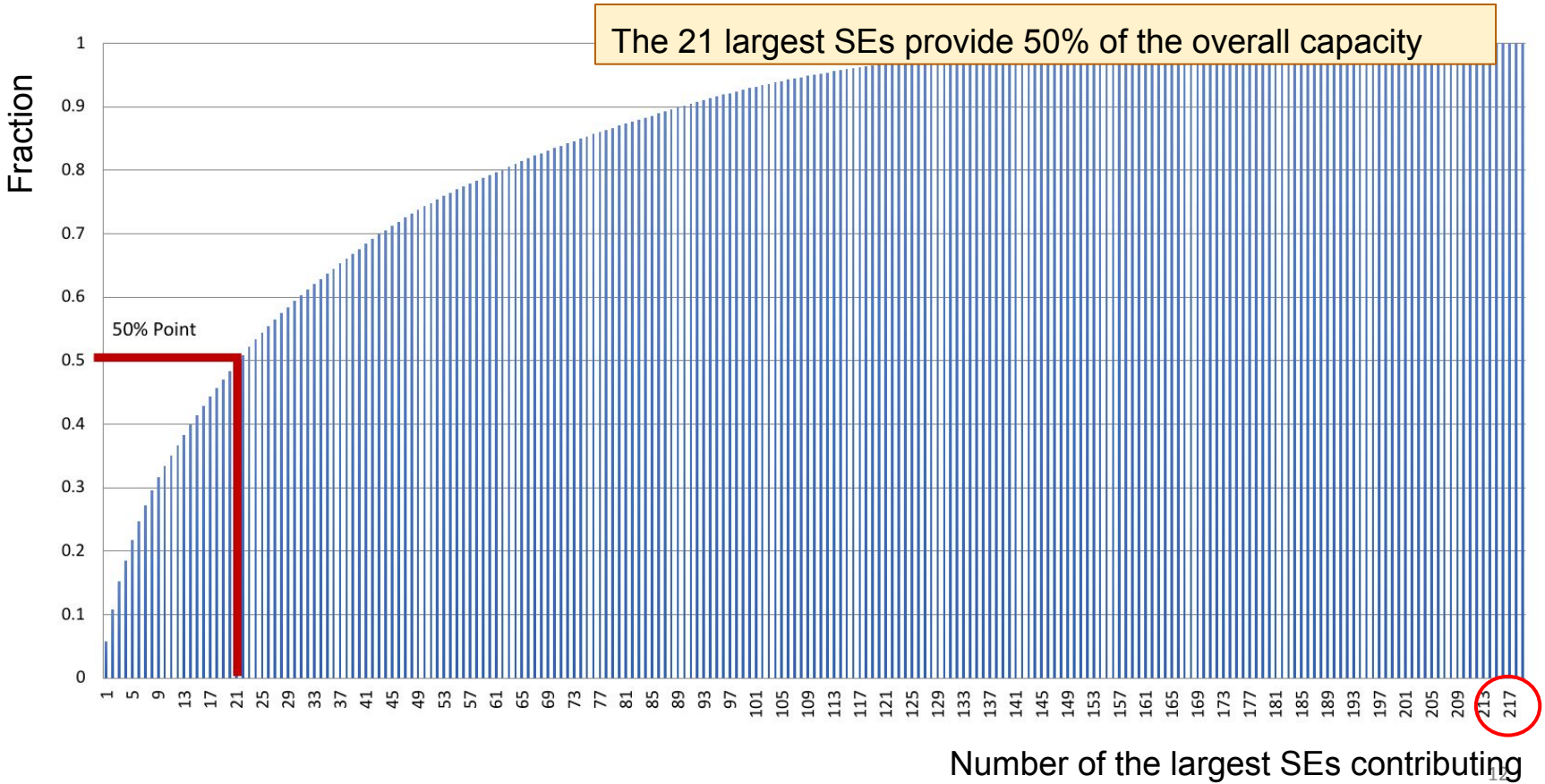
Reference Slides!!

From last year

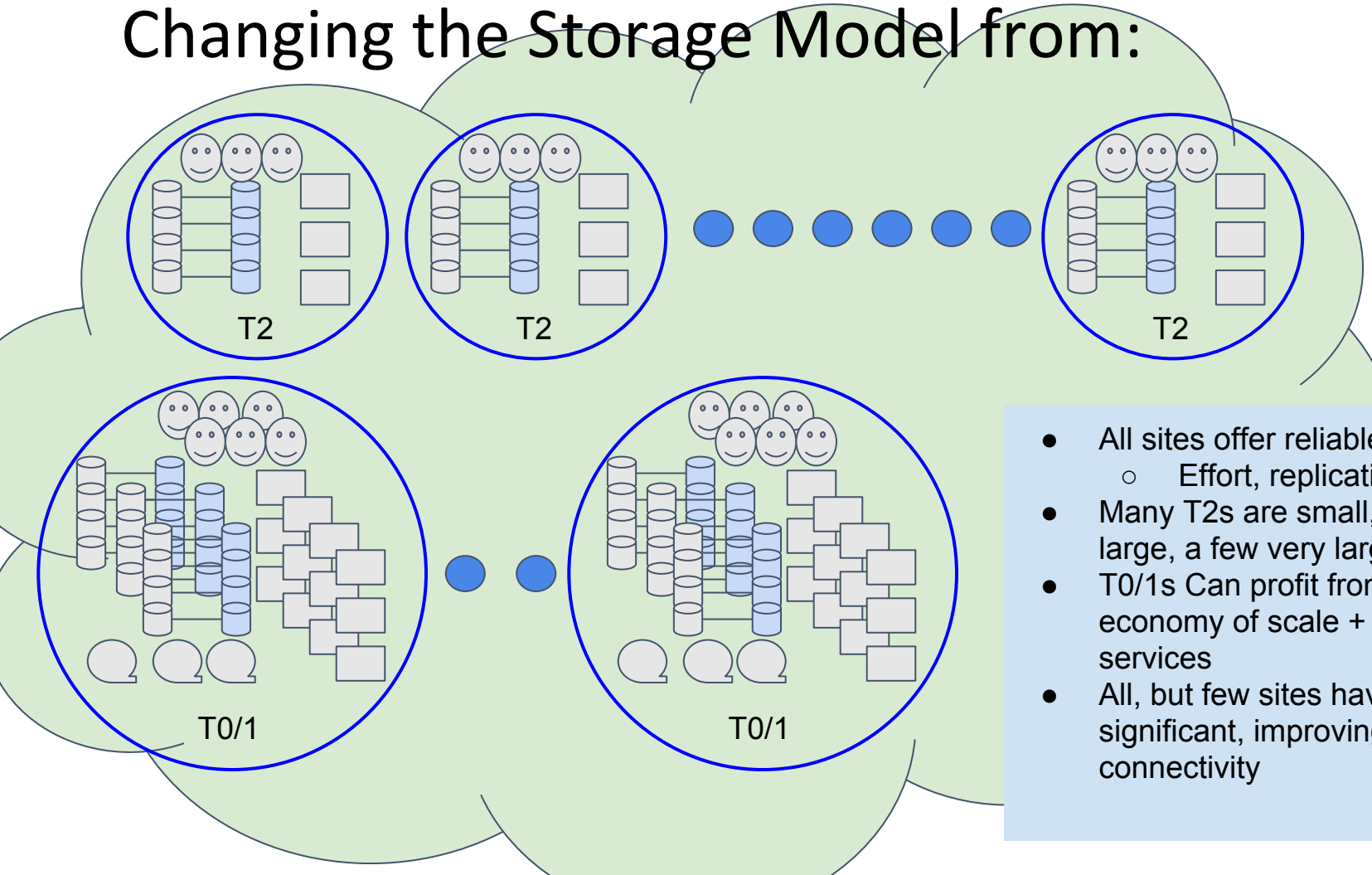
Storage Impact

- Reduction of number of sites with disk storage
  - Replacing them with caches
- Reduction of data redundancy within storage systems
  - Replacing them with nothing (virtual data or tape based backups)
- Reduction of replicas
  - And removing cold data

2016 contribution of SEs to total storage

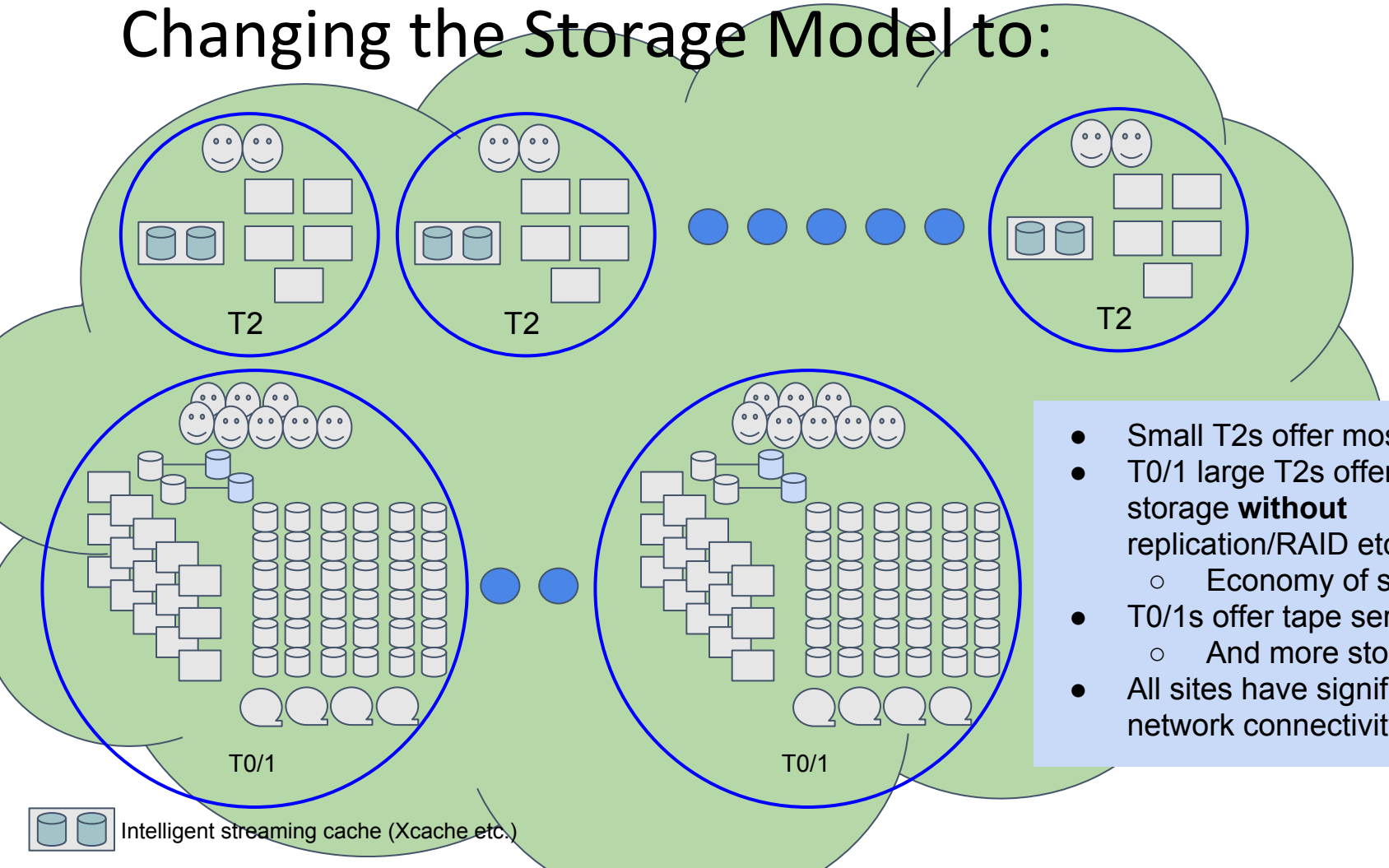


Changing the Storage Model from:



- All sites offer reliable storage
  - Effort, replication cost
- Many T2s are small, some are large, a few very large
- T0/1s Can profit from economy of scale + offer tape services
- All, but few sites have significant, improving network connectivity

Changing the Storage Model to:



- Small T2s offer mostly Caches
- T0/1 large T2s offer most storage **without** replication/RAID etc.
  - Economy of scale…
- T0/1s offer tape services
  - And more storage
- All sites have significant network connectivity

Intelligent streaming cache (Xcache etc.)

Questions:

- What can we save?
- What is the impact?
- Could it work at all?

- Attempt to answer these questions based on current knowledge
- Well aware that this will create more questions…..

Reduction of sites with disk storage

- 2015 site survey indicates that an average T1 requires 2.5 FTEs for (disk) storage operation a T2 0.75 FTEs (hopefully improved to 0.5)
  - In the survey the correlation between size and effort is weak
  - Doubling the size of an existing, large installation requires about 15% more effort
- **≈170 sites (with more than 200 SEs) to O(10) sites**
  - ≈21 largest SEs contribute ≈50%
  - T1(style) sites remain, rest operates stateless buffer/cache systems
    - Effort at the remaining sites increases to about 3 FTEs
  - Buffer/caches require less effort (no stateful services to be maintained)
    - 4 hours a week for swapping broken disks and general system maintenance
- 13 * 2.5 + 157 * 0.5 = 111 15 * 3 + 155 * 0.1 = 60.5 roughly 40% decrease in effort
- Some decrease in effort for the teams managing the experiments' production
  - Fewer stateful sites, simpler services have less complex error patterns
- Started a study on the required sizes of caches for T2s (based on data popularity data)

Reduction of data redundancy within SEs

- Currently most (all) storage systems use some form of redundancy to improve data availability and reliability
  - RAID (depending on RAID level)
  - Duplication/Replication (2->n times)
  - Erasure Encoding (depending on desired behaviour 1.5 – X)
    - But linking multiple servers
  - http://iopscience.iop.org/article/10.1088/1742-6596/513/4/042017/pdf
- Tape costs are about ¼ of disk costs (CERN based figures)
  - Taking into account disk+server costs

Reduction of data redundancy within SEs

- Most of the data can be reproduced
  - Raw data has two archival copies anyway
  - Reproducing data is associated with costs
- **Error rates of disks at the T0** are about 0.9%/a (0.86%/a)<-- EOS[CERN]
  - For a 70k disk system the average daily failure rate is 1.73 disks
  - Data popularity studies can be used to calculate job failure rates
    - And estimate recreation costs/staging costs (lower due to cold data)
- With no internal replication 30%-50% of the disk costs could be saved
  - Requires modifications to experiments' workflow management systems
  - Additional error rate is low compared to current observed job failure rates

Required work

- Storage systems can produce lists of lost files
- Understanding when to recreate
  - Data popularity can give probability for future access
- Understanding cost of recreation for different data
  - Can be complex chains that have to be re processed (avalanche)
  - Cost can be calculated and appropriate level of replication chosen
- Understanding the complexity of automating the complete process
  - A bit like “make” for files
- Understanding which file types would need to be recreated
  - Since disks fail bias free, some losses are identical to a small reduction in Luminosity

How much compute can I get for storage?

- Virtual data only pays off when the amount of compute power to recreate data is less expensive than the storage
- From CERN's (non disclosure) cost figures one can get rough conversion factors between tape/HS06/disk
  - Total cost per year (Building, people, electricity, network,etc)
  - HT on
- **1 TB of raw disk is equivalent to 4 HEPSpec years**
  - erasure encoding X1.36, mirrored X2
  - For 1 TB of disk you can buy 4TB of tape, but here the access frequency changes the costs quickly → for later studies

Example calculation

- If you replace a 100 PB mirrored storage system by a 100PB non mirrored system
- Your additional computer budget is roughly: 4 10^5 HS06 years to compensate for the losses (close to 1000 modern compute servers)
- With the 0.9%/a failure rate you have to recreate 900 TB of data per year
- 1TB of data re-creation is allowed to cost up to: 444 HS06 * y
  - Differs for workloads: AOD 1MB / 30 sec on a 10 HS06 core
  - 1 TB costs roughly 9.5 HS06 years
- For the 4 10^5 HS06 years you can (re)create 42 PB AODs

Rough Estimate

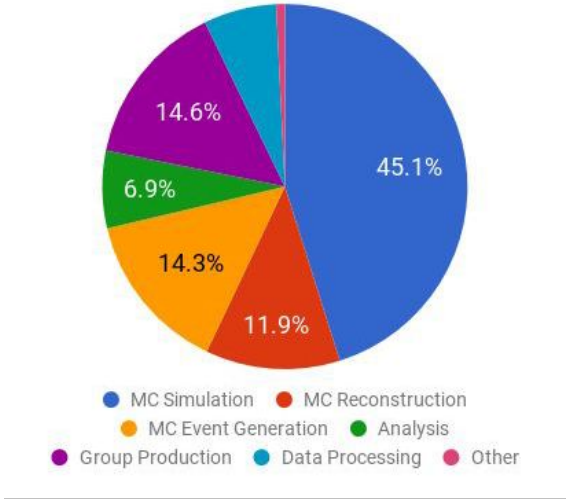
- Take all disk space and all compute of an experiment over a year
- Assume you lose 0.9% of the data randomly
  - The cost should correspond roughly to 0.9% of the computing budget
    - On average better, because some data is raw and a tape copy exists anyway
    - Disk data is hopefully used several times
    - Some data is replicated between sites
    - → **upper bound**
- Example based on CMS 2017 (approx REBUS)
  - 125 PB disk and 1700 kHS06 (T0/1/2)
  - 0.9%/a → 1.125 PB/a corresponding to 15.3 kHS06
  - 125 PB of raw disks buys 500 kHS06 (≈ 30 times what is needed)

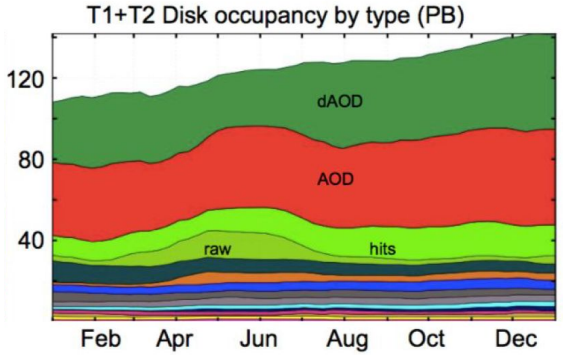
Next Steps

- Take the distribution of different data types and computing spent on them and do more precise estimates
- Take the data popularity records and check what would happen if data is only re-generated on demand (before access)
  - Hint: some data is not accessed before deletion
- Compare the resources needed to compensate for the lack of storage system level replication to other losses
  - Failing jobs
  - “Catastrophic” single events

More realistic approach?

- Looking at ATLAS (similar for CMS)
- Most time is spend on MC Sim and Event Generation, most data is AOD/dAOD
- Replicas for MC Event Generation and MC Sim would for a small fraction protect 60% of the computing investment





Can we replace storage on T2s by caches?

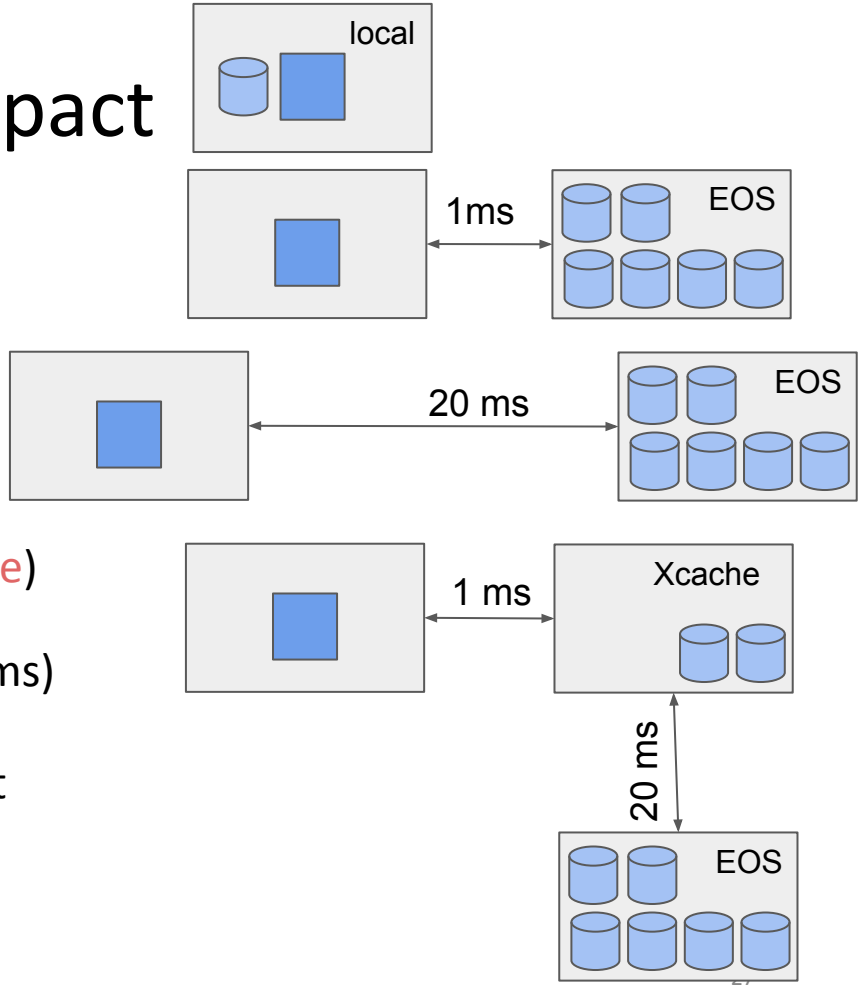
- This could work if the re-used storage space is smaller than the the storage provided
- AND latency can be hidden efficiently by the Cache
- AND the overall bandwidth is sufficient
- Data Popularity records and measurements can answer these questions.
- ***Preliminary*** results of ongoing studies in the next slides
  - Based on work with students during this summer

Credits

- Lucrece Laura AKIRA
  - with David, Andrea and Servesh
  - Comparing workloads run locally, remote and through an xcache
- Irvin Umana Chacon
  - with David and Servesh
  - Using ATLAS DataPopularity logs to study the utilisation of a cache at a T2
- Corentin Bugnot
  - with David
  - Building a tool to run workloads with resource limitations to measure the sensitivity
    - Memory
    - Latency
    - Bandwidth

Data Caching: Xcache impact

- Jobs:
  - ATLAS digi-reco 28GB input
  - ATLAS derivation Job 45GB input
- Setup
  - Data on WN (local)
  - Data on the same site (remote close)
  - Data at Wigner (remote far)
  - Xcache server at the same site (<1ms)
- Goal
  - Measure the impact on throughput



Measurements (Preliminary!!!)

Job type	Run's conditions	Run time	Relative Run time
Atlas-mcdigi-reco	Local Data	240m19s	1.00
	Remote Far	480m28s	**1.9**
	Empty Cache	261m39s	1.08
	Populated Cache	249m43s	1.04
Atlas-derivation	Local Data	147m19s	1.00
	Remote Far	1217m14s	**8.26**
	Remote Close	151m24s	**1.02**
	Empty Cache	155m17s	**1.05**
	Populated Cache	152m44s	1.03

Lesson learned: xcache can, for the tested workloads, hide latency efficiently

Cache Simulation using ATLAS data popularity logs

- Cache with least recently used replacement (Xcache)
  - Increase cache size until hit rate becomes stable
  - Hit rate = hits / (hits + misses) (not ideal :-\ )
- “Average” T2 Prague
- Data Popularity Data for 28 days
  - Limited by storage on our local nodes

PRAGUELCG2

Tier 2 Site (Rebus)

Total Online Storage: 6.3 PB

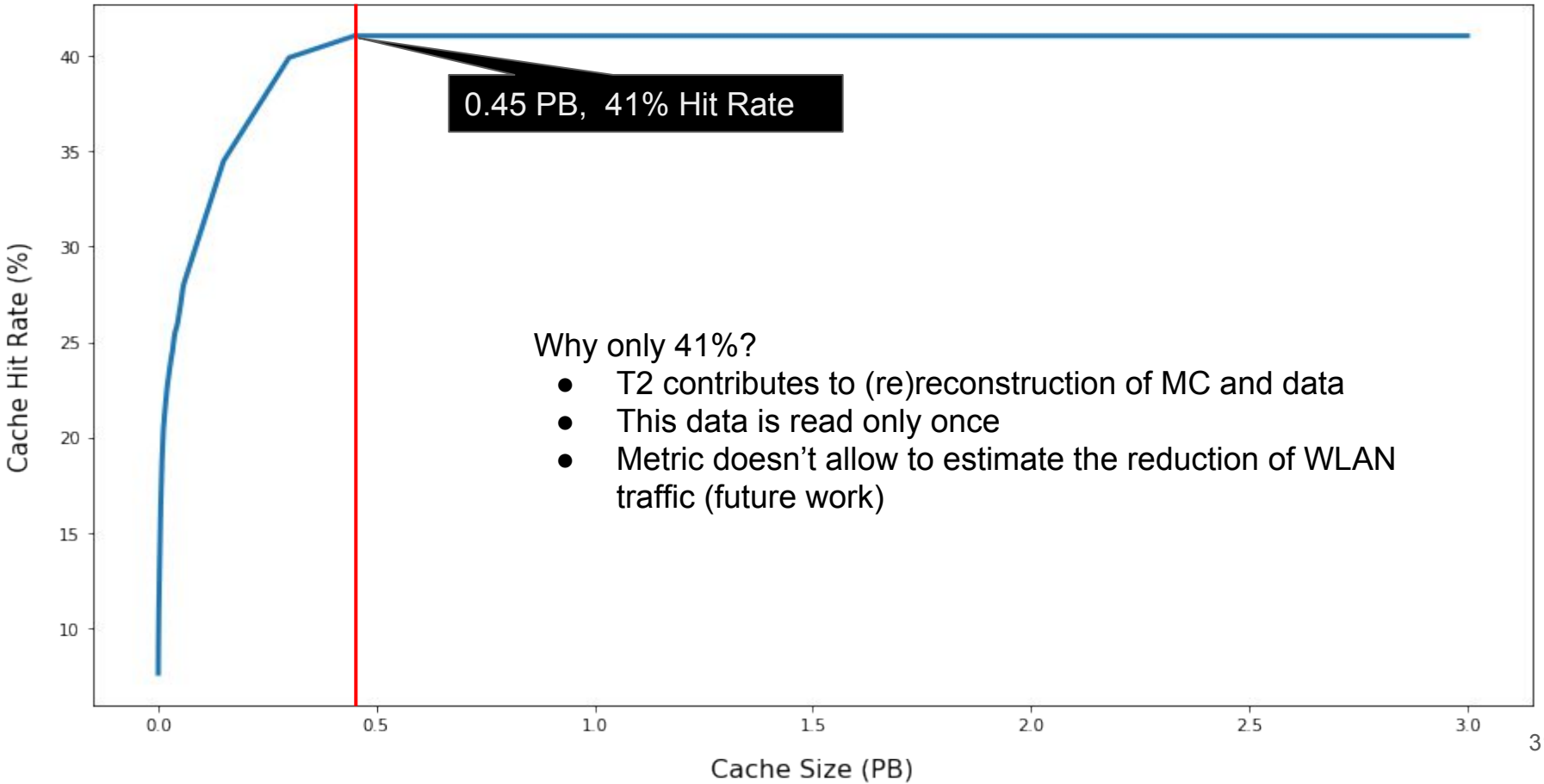
Physical CPUs: 7,028

Logical CPUs: 31,992

Number of operations: **1.4 M (0.58 OP/s)**

Total data accessed: **1.5 PB (620MB/s)**

Hit Rate vs Cache Size for PRAGUELCG2

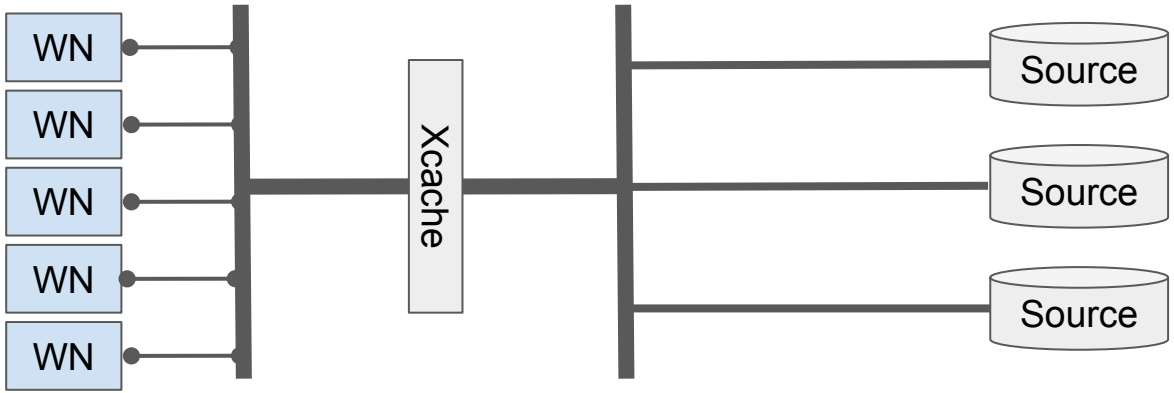


Why only 41%?

- T2 contributes to (re)reconstruction of MC and data
- This data is read only once
- Metric doesn’t allow to estimate the reduction of WLAN traffic (future work)

Additional work to be done

- Repeat the cache measurements to gain some statistics
- Repeat the measurements with workloads from other experiments
  - CMS, LHCb
- How many clients can a single instance serve before it chokes?



Additional Work Planned

- Define a better metric that reflects the amount of data re-used
- Parametrise the cache hit-rate curve by two parameters
  - HitRate at saturation
  - Size needed to reach 80 % of saturation, relative to data throughput at the site
- Check for patterns
  - Rerun the analysis for different sites
  - Rerun the analysis for different time windows
- Introduce a compact binary format for the input data
  - To interface to other experiments
  - To run over larger intervall locally
- Extend the study to other experiments
- ……
- Student(s) and Andrea Sciaba will continue to work on this

https://docs.google.com/document/d/1gzCNIdtCF1IrIUSt-yIMZ18aq0zskljbuFo3Mb_iMss/edit?usp=sharing

Conclusions (preliminary)

- Even for workflows that expect data to be local an intelligent cache can mask latency
- Reduced bandwidth isn’t an issue, especially when the cache averages out peak requests (see additional tests by Corentin)
- T2s storage could be replaced by caches that are significant smaller than the storage on the site (example 10%)

- Many more studies needed!!!
  - Good projects for students (Bachelor/Summer etc.)

What else is needed?

- Lots!!!
- More/different Workloads
- Combining Latency and Bandwidth limiting with cache test
  - Impact of distance between cache and data, slow networks
- Cache tests throughput limitations
- Different Cache implementations
- Data Popularity based analysis
  - More experiments
  - Different time frames
  - Better Metric
  - Throughput Studies
- ……..

Conclusion

- First building blocks to understand the impact of caches are taking shape
- With a bit more work we can make cost and performance statements on different proposed architectures for HL-LHC era

Many thanks to the students and supervisors for their work :-)

Reduction of experiment replicas

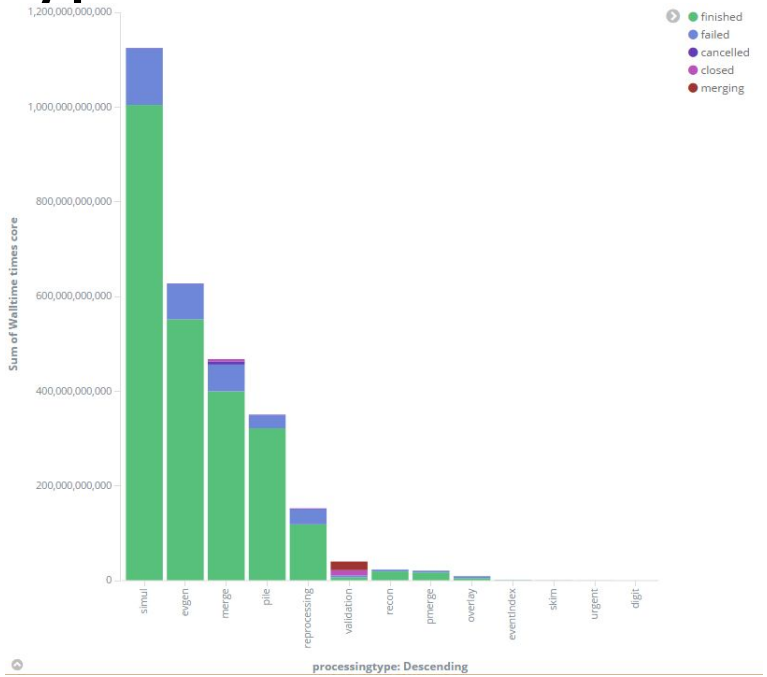
- Reported replication rates are in the order of 1.3-1.5
  - Plus multiple versions of analysis data
  - Plus overlaps in derived AODs
- Using the published data popularity summaries it can be estimated that 20% of the data could be moved to a cheaper layer
  - Less than 6 months since last reading
  - More detailed studies needed (currently underway)
  - Adding complexity and need for more archive facilities
  - → 15% cost reduction

Scheduling and Site inefficiencies

- Problem of schedulers, tessellation, not used cores on storage services, multi core workflows with periods of single core usage,....
- Observation on (T0, T1s)
  - Estimated losses 20-30% (conservative optimistic)
  - Excellent compared to HPC where 40% utilisation is already good
  - Backfill, asynchronous I/O, batch on storage allow to use about 90%
    - With extra complexity
  - → 10-20 % possible gain
- Loss due to intermittent job failures
  - Based on Andrea's log studies 10 – 15% of the wallclock time is lost
  - These could (theoretically) be reduced to >5% → 5-10% gain

Inefficiency by processing type

Processingtype	Wallclock inefficiency (%)	WC contribution (%)
evgen	12	23
merge	15	17
simul	9.5	37
reprocessing	25	5
pile	7.1	9
recon	16	1



Most jobs fail early wasting little wall clock time
But number of failed jobs is significant adding to scheduling costs

carefully looking at what our jobs do (example: A LHC experiment)

- The reported problem was bad efficiency when more threads were used:
  - 4-16 threads the efficiency went from 86% to 35%
  - Some non thread safe code was identified
- As a side effect it was realised that a lot of time was spent in ROOT compression
  - The default parameter was set to 9 (best compression)
  - Setting this to 0 (no compression) brought the wall-time from 805 to 344 sec (but the file grew from 1.7 to 5.4GB)
  - With compression set to 1: 805 to 402sec with an increase from 1.7 to 1.8 GB

Measurement table:

Compression Level (Default - 9)	Output file size (step2.root)	Threads	CPU Util	Real	User	Sys	Efficiency (User time / (Real time X No of threads))	Speedup (Normalized to 4th default)	Speedup (Normalized to 4th in each category)
9	1.7G	4	363%	999.13	3430.53	203.79	85.84%	1	1
9	1.7G	8	568%	805.72	4188.29	392.16	64.98%	1.24	1.24
9	1.7G	16	633%	810.05	4605.24	522.72	35.53%	1.23	1.23
0	5.4G	4	358%	544.45	1869.86	82.31	85.86%	1.84	1
0	5.4G	8	639%	344.03	2087.33	111.57	75.84%	2.9	1.58
0	5.4G	16	992%	250.14	2320.45	161.92	57.98%	3.99	2.18
1	1.8G	8	684%	402.01	2589.94	161.79	80.53%	2.49	

Compiler/Code/Build based improvements

- Studies over the last 3 years by Nathalie R., David Smith, Servesh M.
  - Building code for specific platforms
    - Currently none of the advanced features are used
  - Vectorisation
  - Intel Compiler
  - Changes to the build/link approach
    - Reducing shared library overheads by building “large” libs instead of loading many small libs
    - Patch to the dynamic linker
  - Feedback guided optimisation (Intel and gcc based AutoFDO)
  - Looked at GEANT4, Reco-Code, NLO Generators
  - Several 15-20% CPU improvements (not all can be combined)
    - HEP code is very sensitive to compiler switches, significant work on Physics validation is necessary

Profiling Code and Hardware Counter

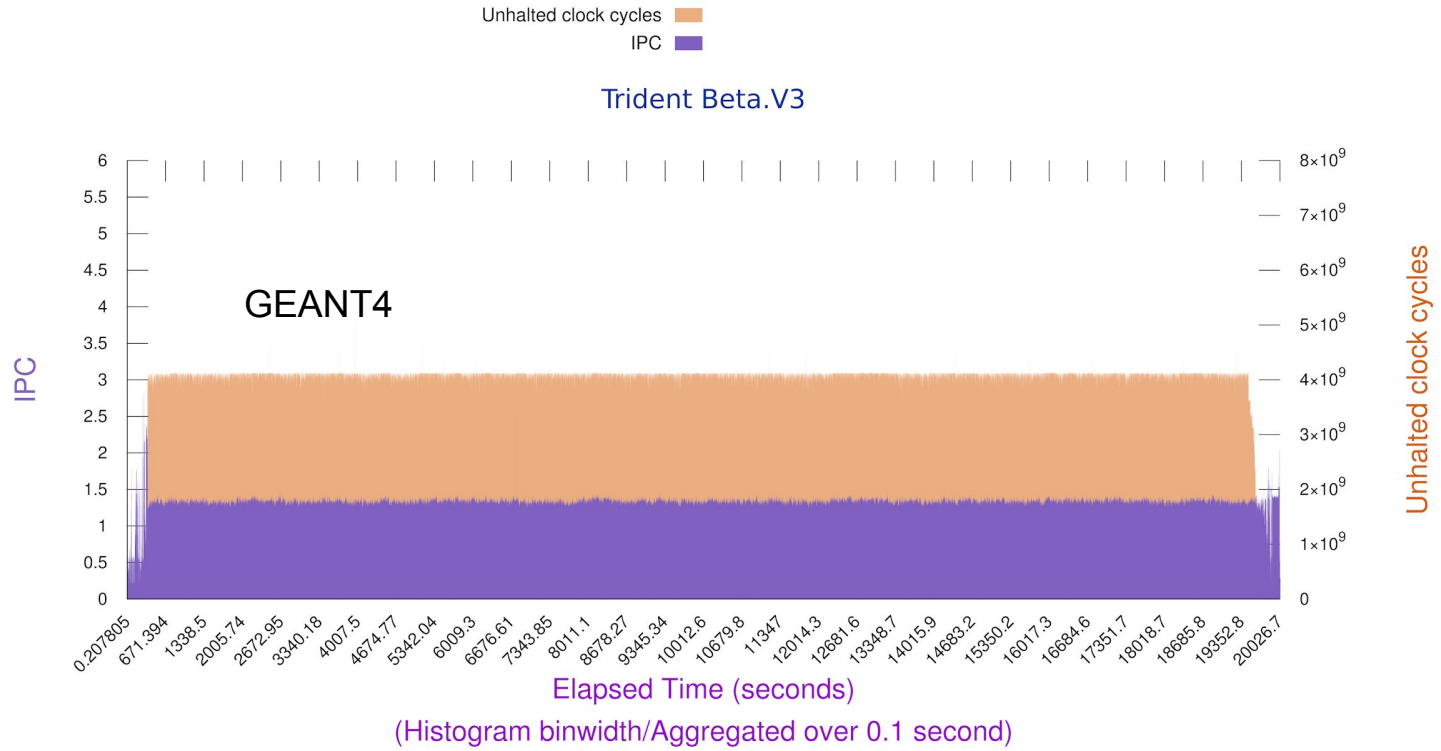
- Profiling code shows up to 25% time spend in memory management
  - Allocation and deallocation of small, extremely short lived objects (FOM tools)
    - Using standard techniques 10% can be saved (some refactorization)
    - Replacing standard template container classes (LHCb works on this for Run3)
- Hardware Counter based analysis
  - HEP code 0.8-1.5 Operations per cycle
  - HPC code up to 4, some being vector instructions delivering up to 8 floating point ops (AVX-2)
  - 100% improvement should be feasible, but will require significant changes in the code
    - While keeping the same algorithms
    - Requires significant skills
    - ALICE HLT tracking code, designed for GPUs produced also huge gains on CPUs with vector units
      - David Rohr & ALICE, several talks and papers on cellular automaton

Gallery of Measurements

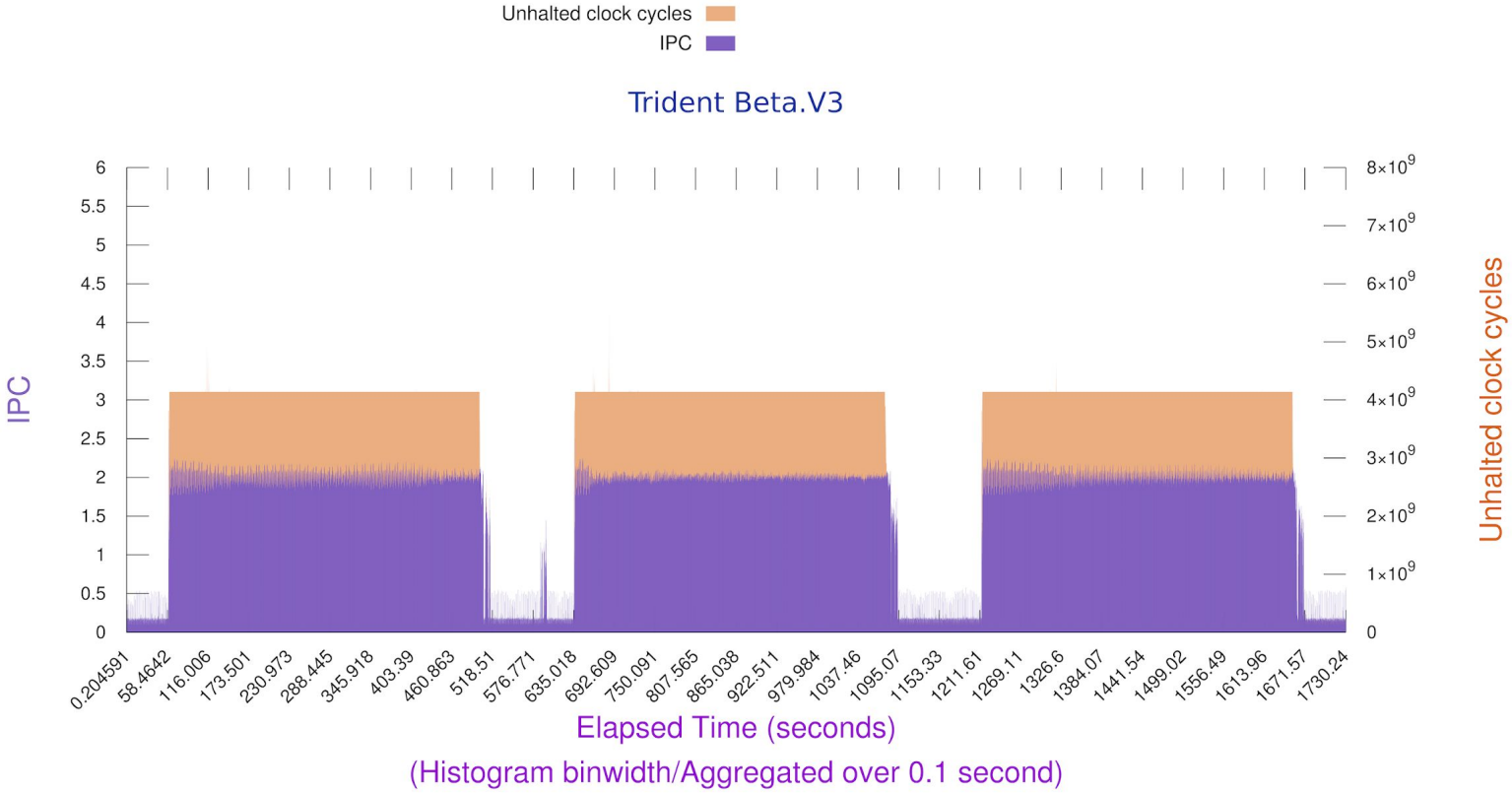
- Still quite complex
- Servesh is improving the readability of the graphs
- Core message: **We use only a fraction of the capability of our hardware**

- Some of the benchmarks in HS06 do much better
  - Which is bad for a benchmark ;-)

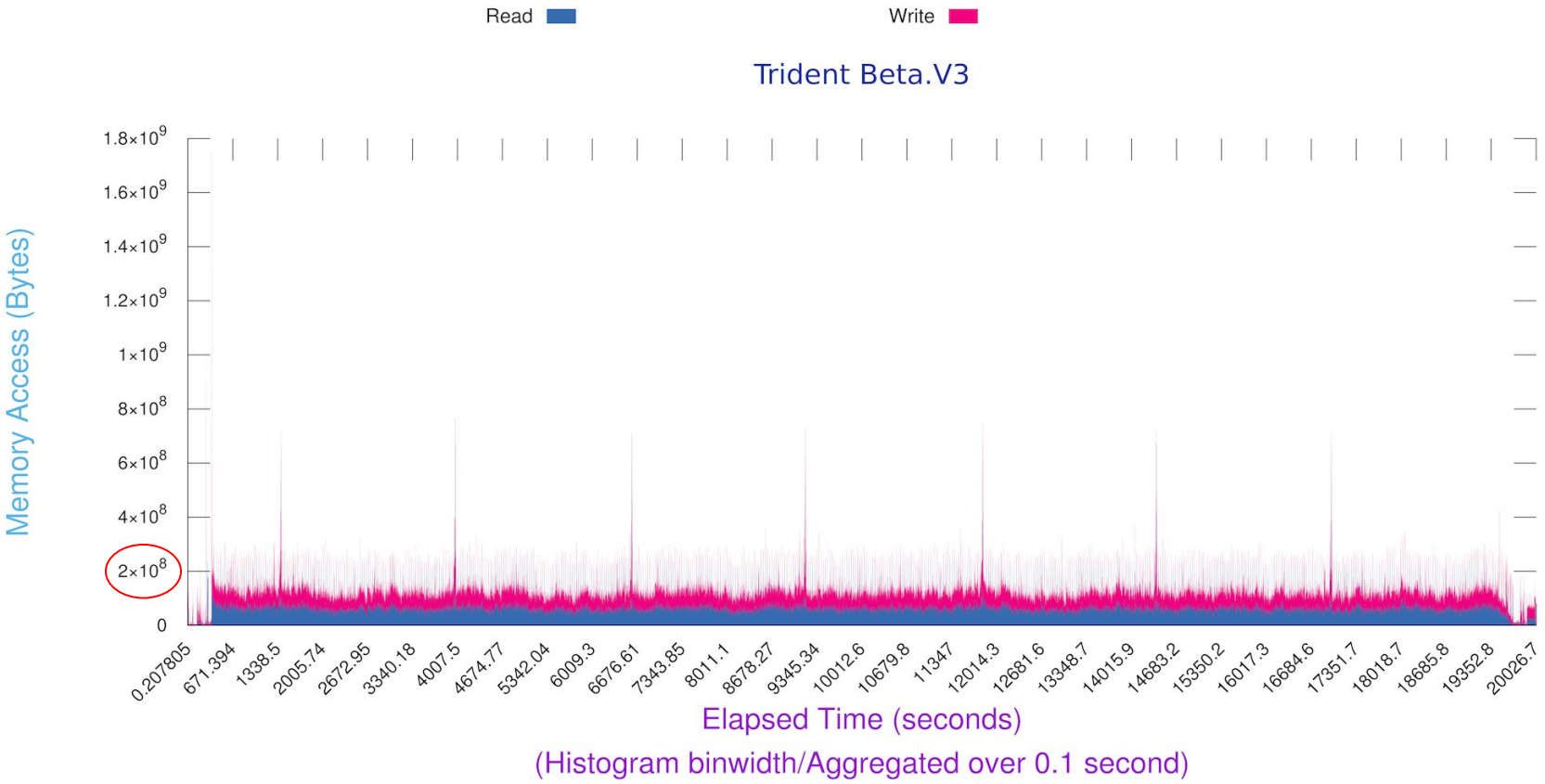
Some examples:



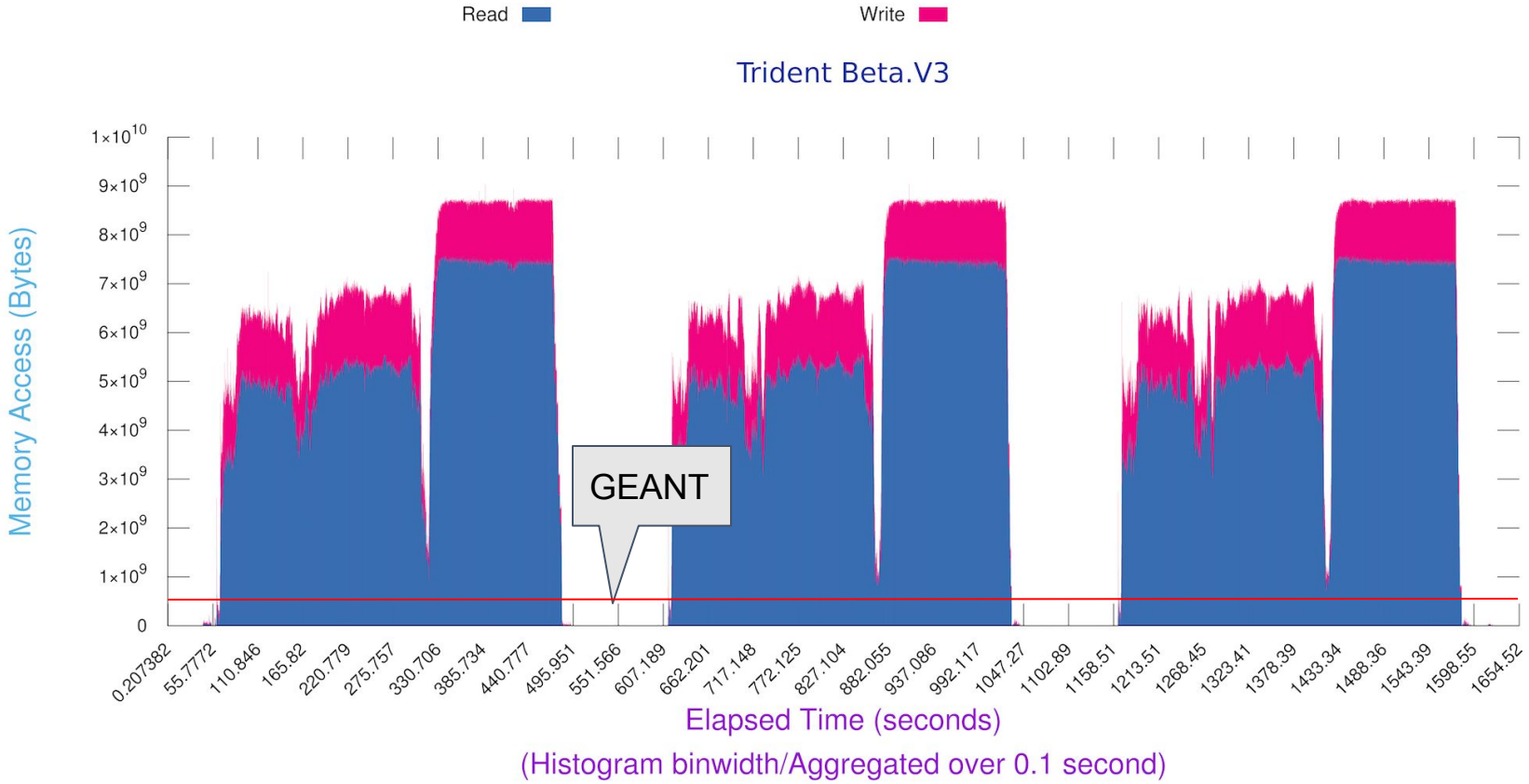
HEPspec 444


Unhalted clock cycles
IPC
Trident Beta.V3
IPC
Unhalted clock cycles
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
8×10^9
7×10^9
6×10^9
5×10^9
4×10^9
3×10^9
2×10^9
1×10^9
0
0.204591
58.4642
116.006
173.501
230.973
288.445
345.918
403.39
460.863
518.51
576.771
635.018
692.609
750.091
807.565
865.038
922.511
979.984
1037.46
1095.07
1153.33
1211.61
1269.11
1326.6
1384.07
1441.54
1499.02
1556.49
1613.96
1671.57
1730.24
Elapsed Time (seconds)
(Histogram binwidth/Aggregated over 0.1 second)

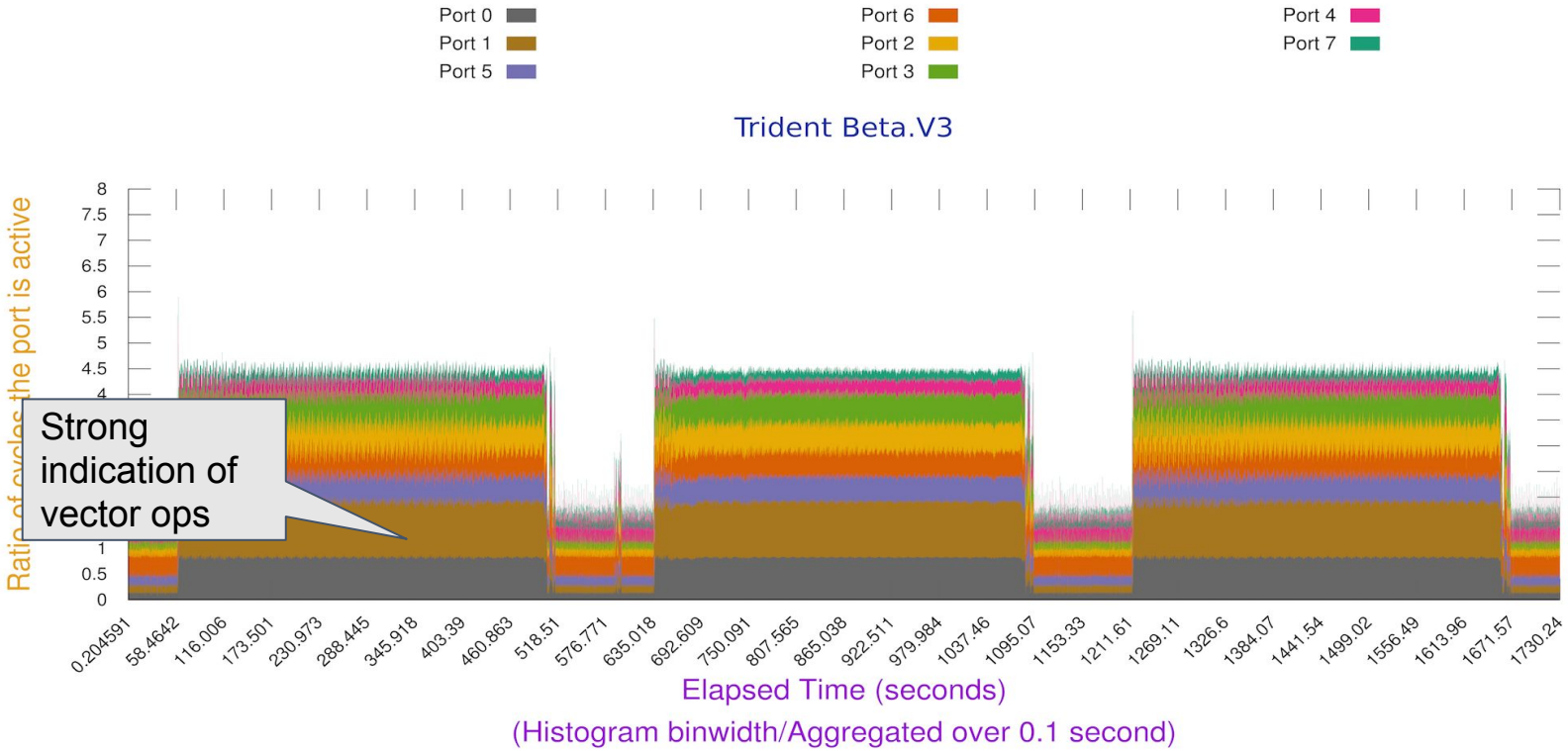
Geant Memory



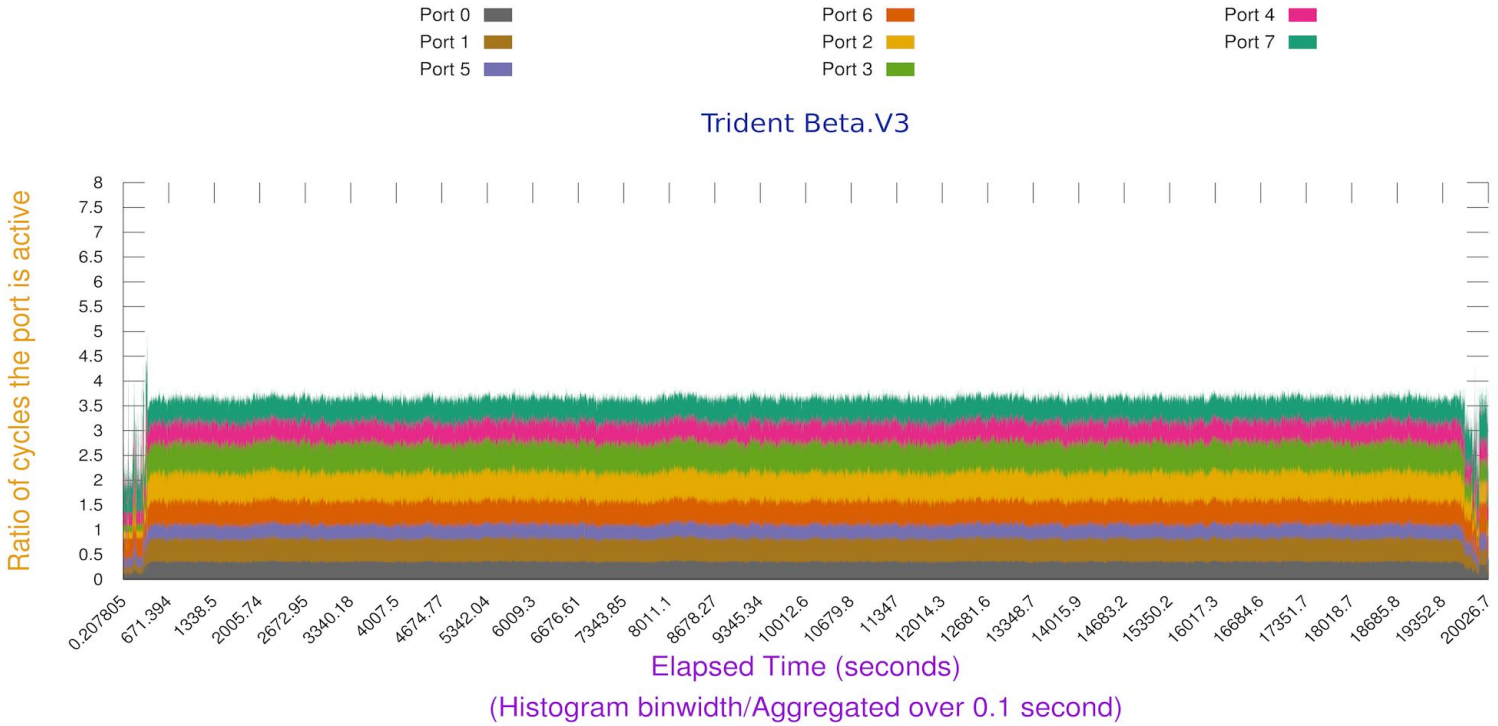
HEPSpec 450


Read
Write
Trident Beta.V3
Memory Access (Bytes)
1×10^10
9×10^9
8×10^9
7×10^9
6×10^9
5×10^9
4×10^9
3×10^9
2×10^9
1×10^9
0
GEANT
0.207382
55.7772
110.846
165.82
220.779
275.757
330.706
385.734
440.777
495.951
551.566
607.189
662.201
717.148
772.125
827.104
882.055
937.086
992.117
1047.27
1102.89
1158.51
1213.51
1268.45
1323.41
1378.39
1433.34
1488.36
1543.39
1598.55
1654.52
Elapsed Time (seconds)
(Histogram binwidth/Aggregated over 0.1 second)

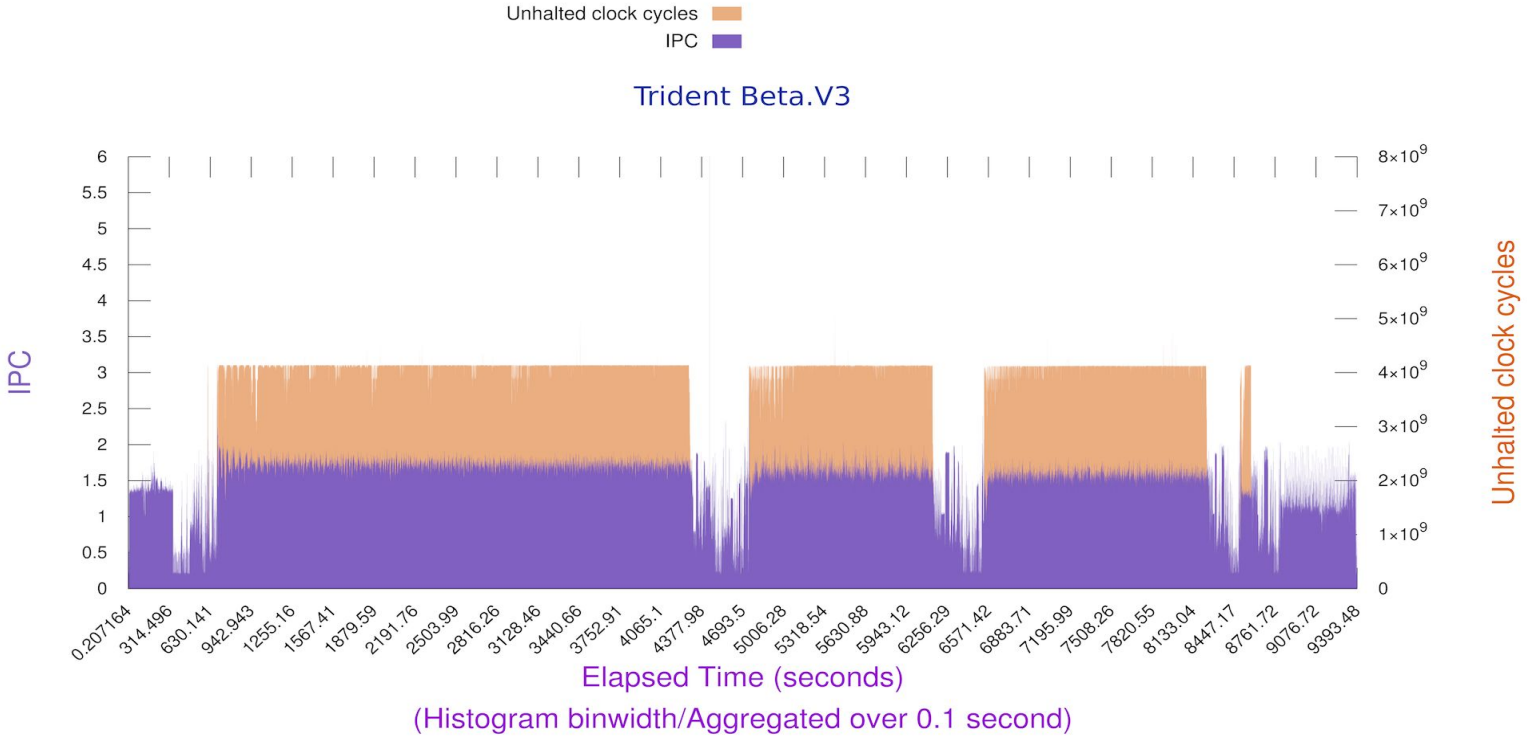
444 port usage (0,1,5,6 compute 0&1 AVX2)



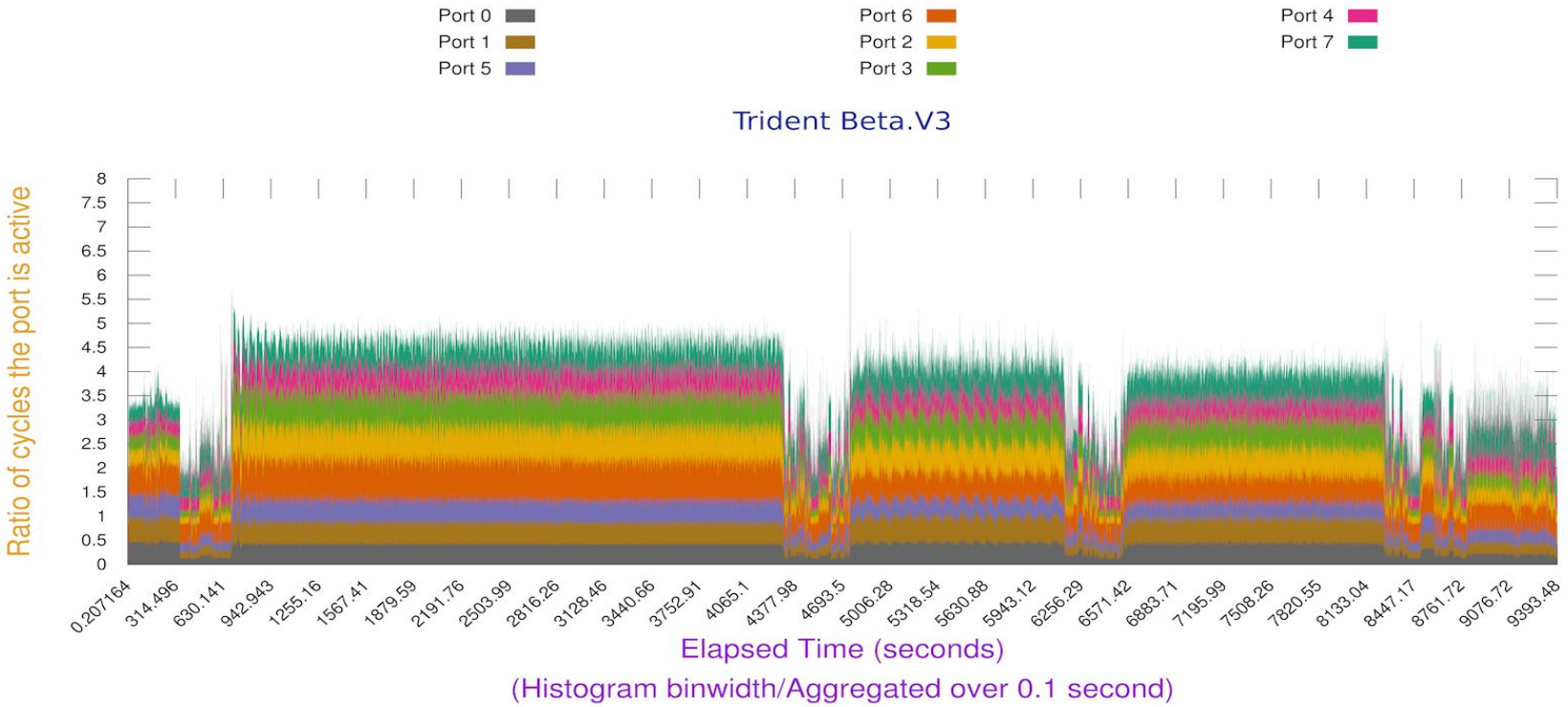
GEANT Port usage



Pileup/digi/reco



pile/digi/reco

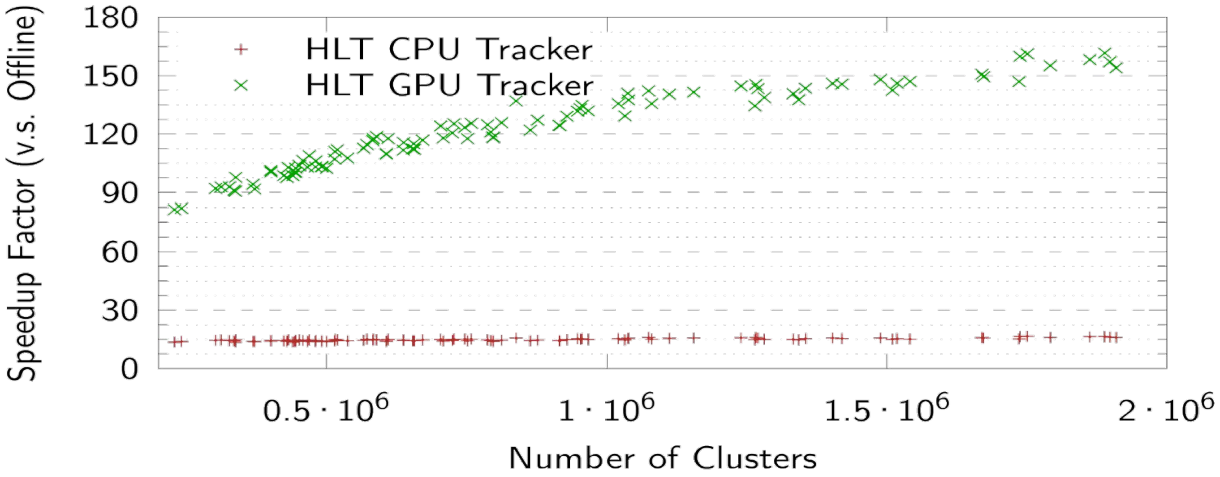


I/O

- Some studies comparing file sizes and block read operations (disk)
- Full machines (less buffer spaces) 2.8 times increase in data read
  - Blocking readahead reduced the overhead to a factor of 2
- Some studies on overhead for object packing, unpacking, compression (ROOT)
  - Significant compared to overall time spend on I/O
  - But in absolute terms irrelevant
- Lower priority since most of the workloads are not I/O limited
  - 80/20 rule ….

Gains from new paradigms: ALICE HLT

- Alice implemented new tracking code for the HLT
  - Based on Cellular Automatons (David Rohr)
  - Written for GPUs from the start
- HLT tracking is 15 times faster than the offline code
  - Additional factor of 10 from GPUs



Summary

Change	Effort Sites	Effort Users	Gain
managed storage on 15 sites + caches	Some on large sites/gain on small sites	little	40% decrease in operations effort for storage
Reduced data redundancy	Some large sites	Frameworks some	30-50% disk costs
Reduced data replication and cold data	little	Frameworks some	15% disk costs
Scheduling and site inefficiencies	Some	Some	10-20% gain CPU
Reduced job failure rates	Little	Some-Massive	5-10% CPU
Compiler and build improvements	None	Little-Some	15-20% CPU
Improved memory access/management	None	Realistic	10%-15% CPU
Exploiting modern CPU architectures	None	Massive	100% CPU
Paradigm shift algorithms	Some	Massive-Infinite	Factor 2-100 CPU
Paradigm shift online/offline data	Little	Massive-Infinite	2-10 CPU 10-20 Storage

Current understanding

- Without change in paradigms and algorithms
  - But with added complexity and significant code work
  - Storage costs could be reduced by up to 45%
  - Effort in site storage operation up to 40%
  - CPU all combined up to a 200% (50% at moderate cost)
- This has to be compared to gains by:
  - New analysis formats (CMS Mini/NanoAODs)
  - Fast MC
  - New algorithms (ALICE tracking code acceleration O(1000))
  - LHCb turbo stream (online calibration/no raw data).....

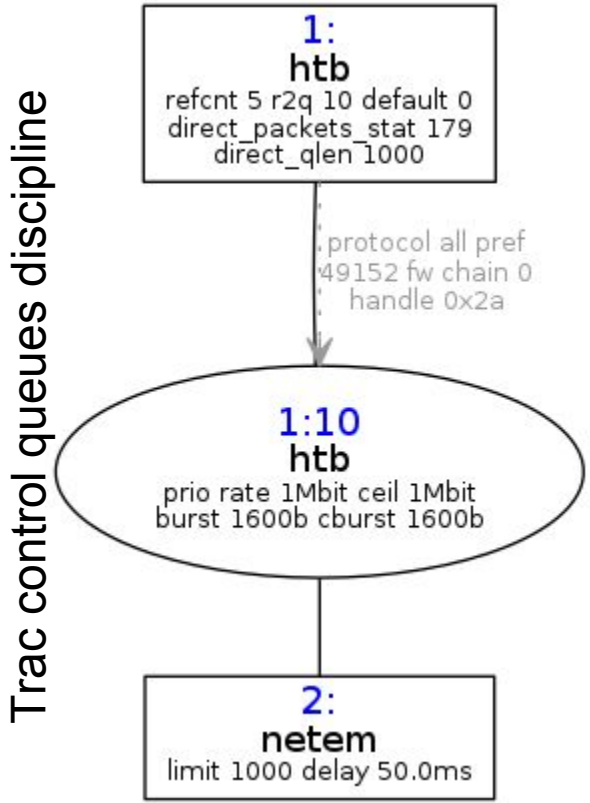
The dilemma

- Should we invest in gradual improvements?
  - Easy to motivate
  - No change in how we do physics
  - Skills largely available
  - Return of investment within short time
  - Results somewhat limited
- Should we focus work on more fundamental changes?
  - Outcome often not clear at the start (**R**&D)
  - Sometimes going against our culture (no raw data stored)
  - Requires sometimes discourse on how we do physics
  - Training investments needed
  - Results potentially massive
- Doing both with the low intensity?

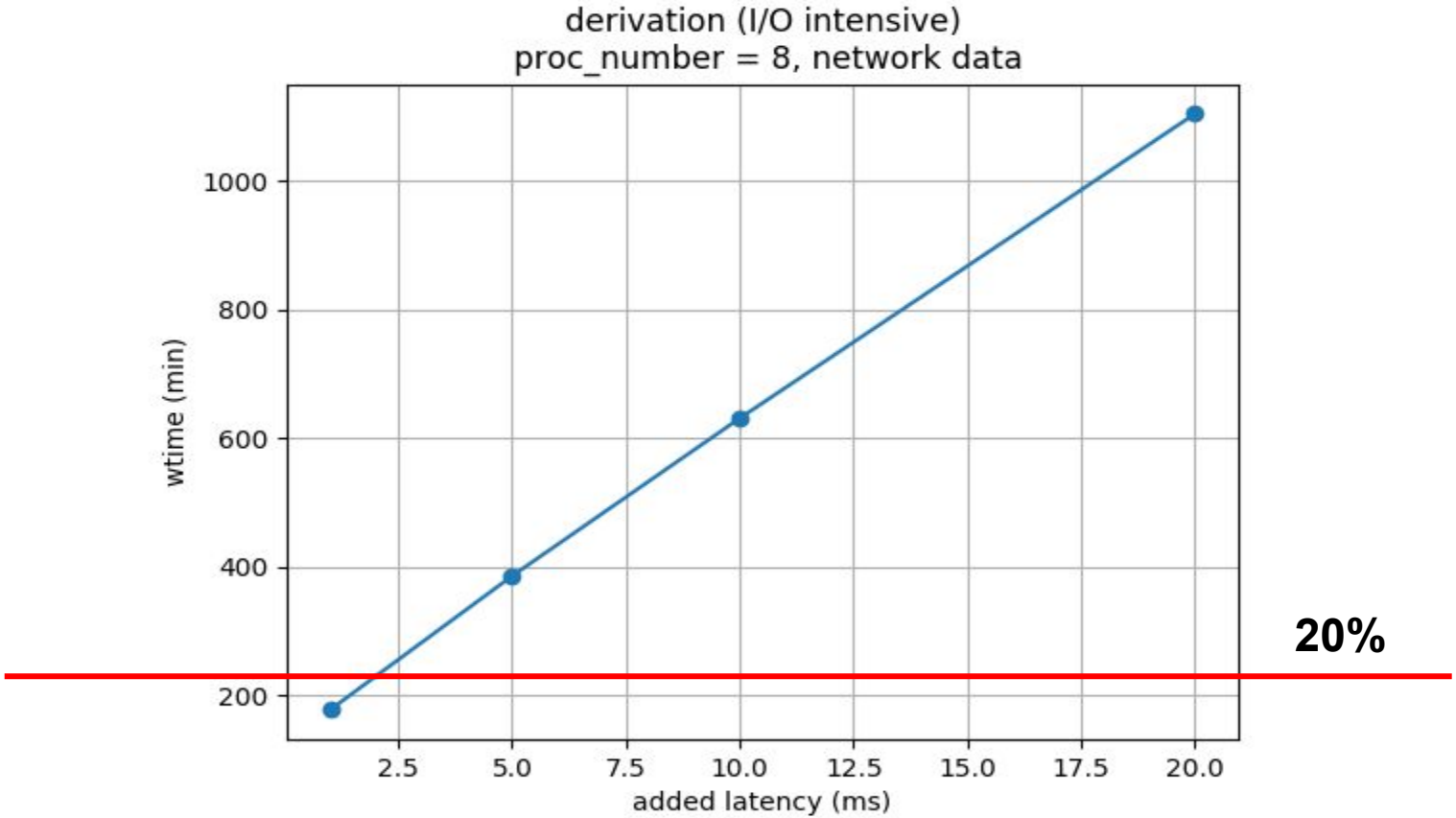
Additional Reference Material

Adding Latency/Bandwidth Limitations

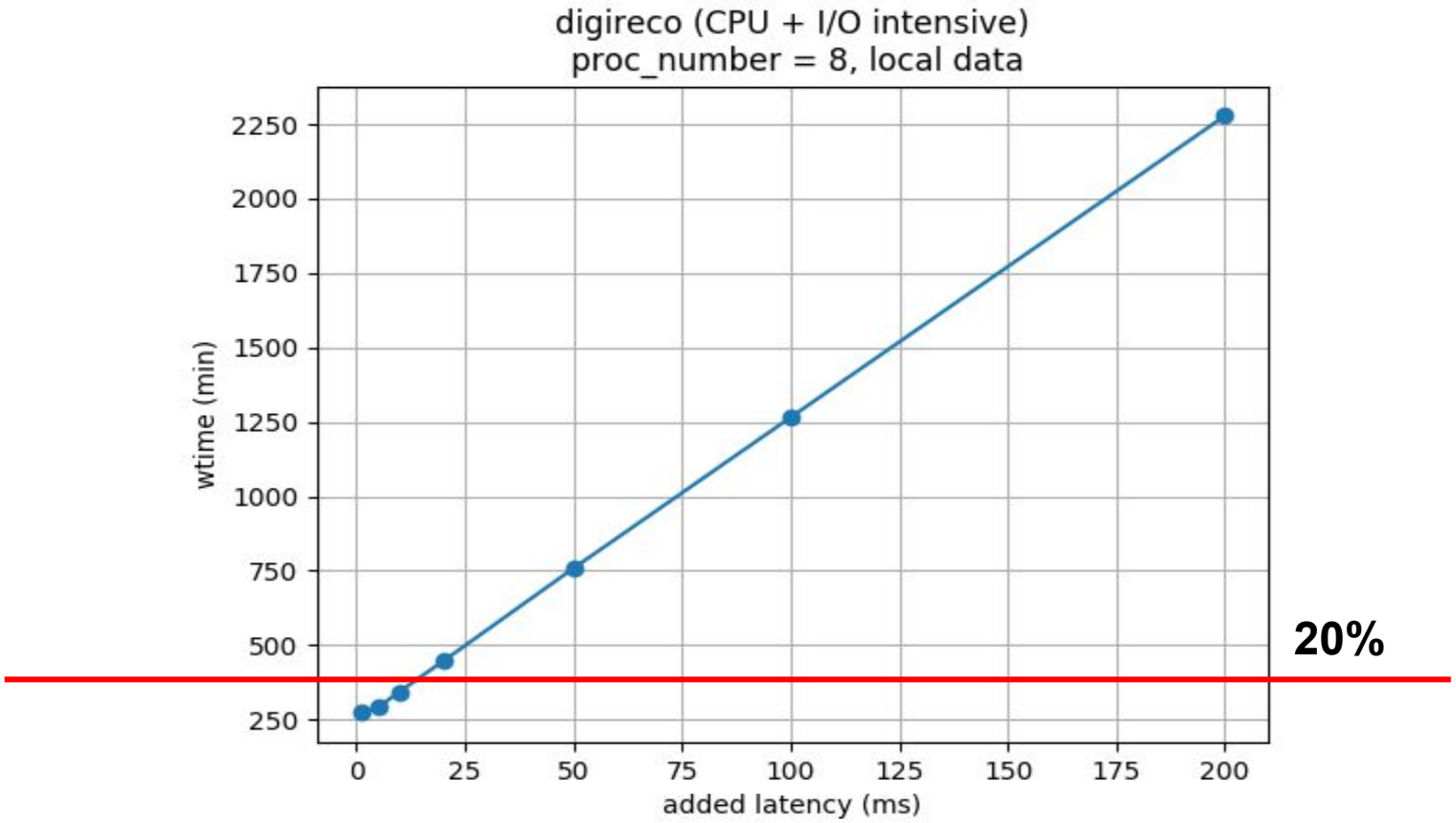
- Cgroup: mark egress and ingress packets from a group of processes
- Egress trac : Trac Control to add rate limit and/or latency
- Ingress trac: iptables module HASHLIMIT to drop packets above a given rate
- Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz 32 cores



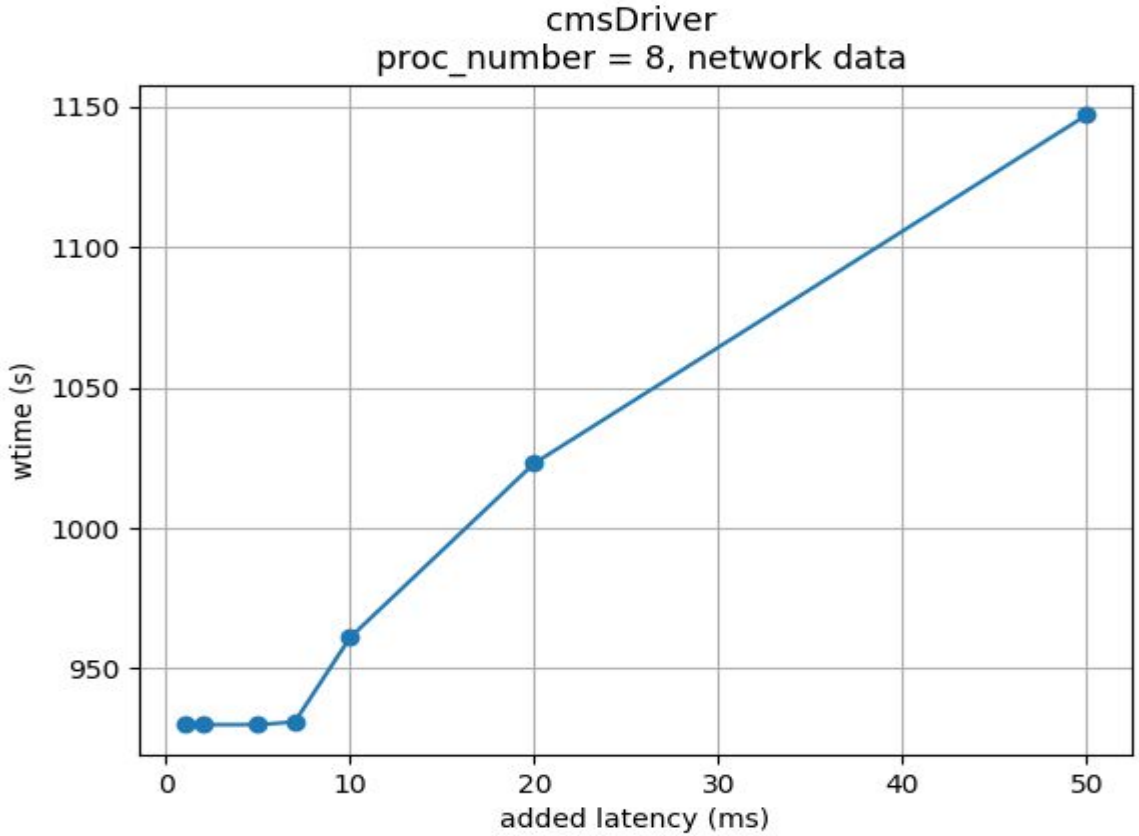
Preliminary results: Latency 1



Preliminary Results: Latency 2

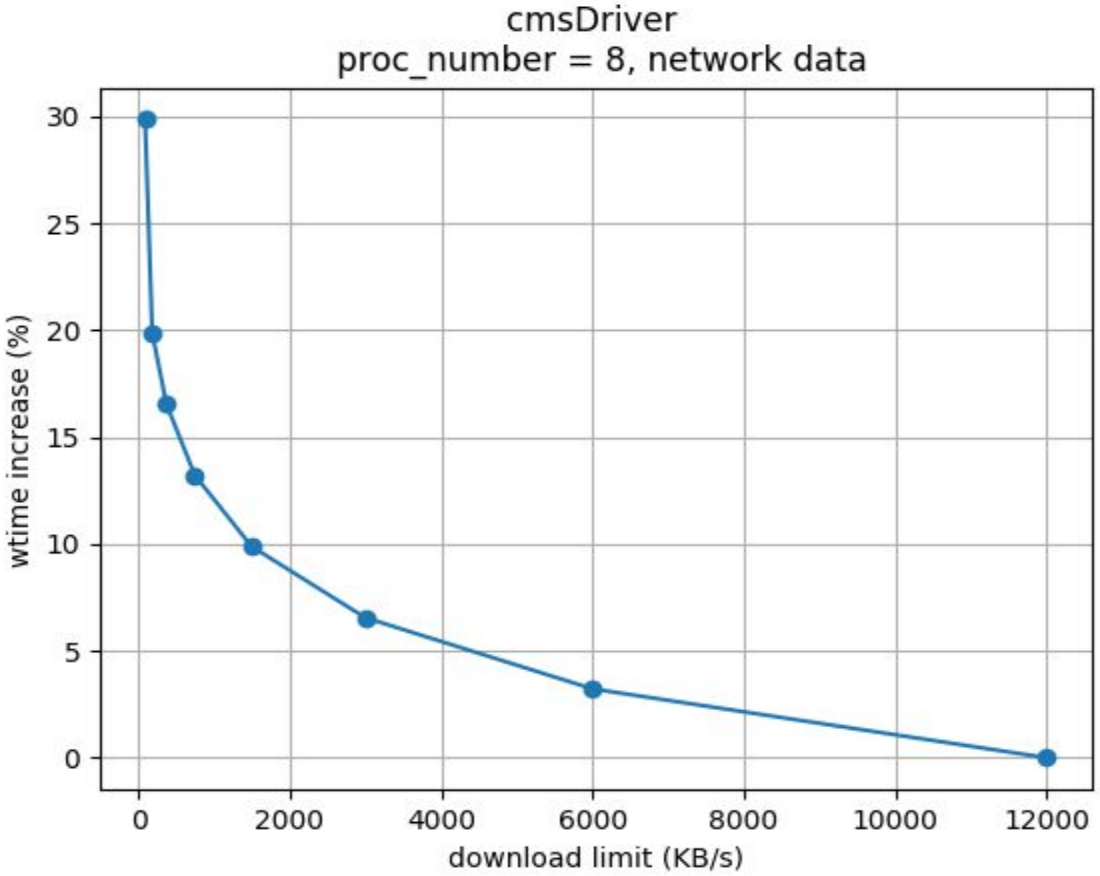


Preliminary Results: Latency 3



20%

Preliminary Results: Bandwidth



Links

- Preliminary work done by our student on cache simulation:
- Presentation:https://indico.cern.ch/event/747825/contributions/3093115/attachments/1699081/2735624/PresentationIrvinUmana.pdf
- Simulation Work: https://cernbox.cern.ch/index.php/s/fxKHTrwrRicWpNU
- Information about the work files: https://docs.google.com/document/d/1dxBrNxLwLwrbwY2NTJfYwStq7VK_bU1P0aLE1o-KLYI
- Preliminary report: https://docs.google.com/document/d/12nHbD7fpdZTFAejCpdZbQgRKFxfFT-D3nueS8A6grg8/edit?usp=sharing
- Presentation, measuring ATLAS digi-reco (panda #3800125271) and derivation (panda #3800239296) workloads under different input file access conditions: https://cernbox.cern.ch/index.php/s/etgN1EBN4X98FjN
- Report: https://cernbox.cern.ch/index.php/s/qlZlp8046B8BEBR
- Resource access limitations: Presentation workloads-limitations(2).pdf