ATLAS I/O Performance Measurements

Martin Errenst

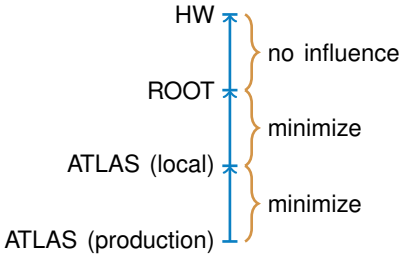
ROOT I/O Workshop

2018-06-20

BERGISCHE
UNIVERSITÄT
WUPPERTAL

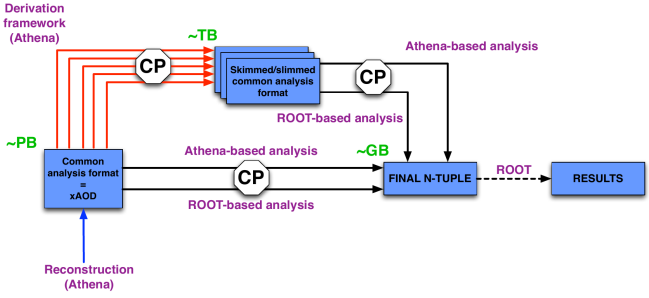
ATLAS I/O PERFORMANCE STUDIES

- Hardware & ROOT define ceiling for ATLAS I/O performance
- Local performance is estimate for ceiling in production
- Minimizing gaps by tuning implementation & workflows
- Dimensions of optimization:
  - Read / write speed
  - File size
  - Size in memory
- Weighting depends on workflow



Focus here on ROOT configuration for reading & compressing ATLAS xAODs

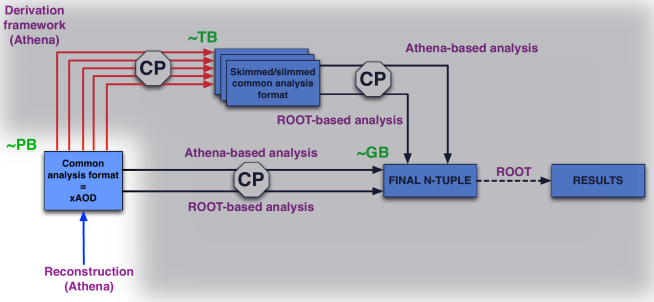
ATLAS PRODUCTION STEPS



Production workflows

1. Reconstruction
2. Derivation
3. Analysis

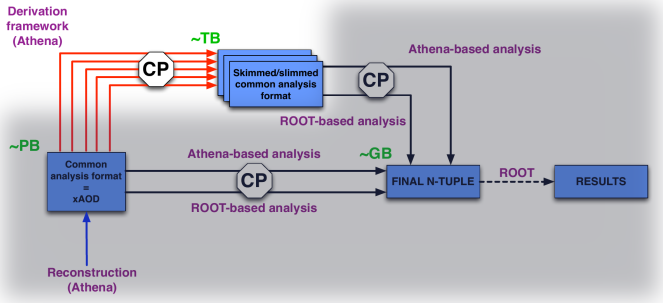
ATLAS PRODUCTION STEPS



Reconstruction:

- Input: Raw
- Output: xAOD
- File size most important
- $\Rightarrow$ LZMA
- Not I/O limited:
  - example writing: $\frac{10.4}{3414}\,\mathrm{s} \approx 0.3\%$

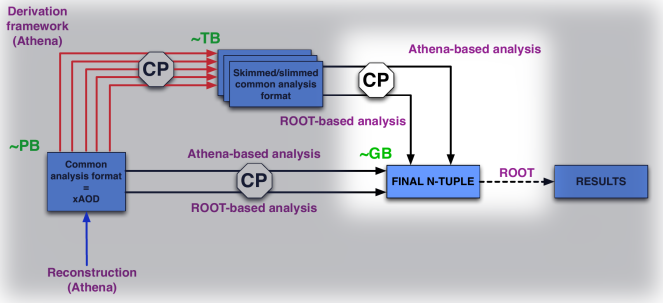
ATLAS PRODUCTION STEPS



Derivation

- Input: xAOD
- Output: DxAODs (multiple formats)
- File size still relevant
- Some I/O dependent
- Average not I/O limited
- $\Rightarrow$ LZMA, LZ4 or keep zlib?

ATLAS PRODUCTION STEPS



Analysis

- Input: DxAODs
- Read often and by many people
- $\Rightarrow$ reading speed is important

SWITCHING TO LZMA FOR XAODS

- Previous default zlib, level 5, compression ratio of 3 - 3.5
- LZMA, level 1 investigated since March, in production about now
- Example dataset: 7.791 TB instead of 8.606 TB $\Rightarrow \approx$ 10% reduction

Compression	**File size** (TB)	**Reduction** (%)	**walltime fraction** (%)
zlib (level 5)	8.606	100	$\approx$ 0.3%
LZMA (level 1)	7.791	90.53	$\approx$ 1.6%

Effect on derivation

- $\approx \frac{1}{10}$ higher walltime for larger derivation
  - Lots of CPU processing (closer to the average derivation)
- $\approx \times 2$ walltime for simple derivation
  - Not much processing done $\Rightarrow$ very I/O dependent
  - Found inefficiency => reduced slowdown to a few %

CONFIGURATION DIMENSIONS

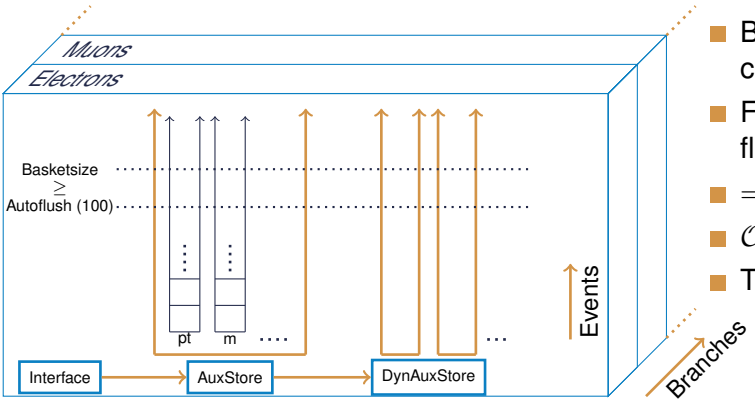
File creation:

Variable	Options
Compression algorithm	zLib (1), lzma (2), lz4 (4)
Compression level	1, 3, **5**, 7, 9 (lz4&zlib), **1**, 2, 3, 5 (lzma)
Flushsize	. . . , 100 (default), . . .
Splitlevel	0 (default), $\geq$1 (only Aux)
Basket size	16000, 32000 (default), 64000

File reading:

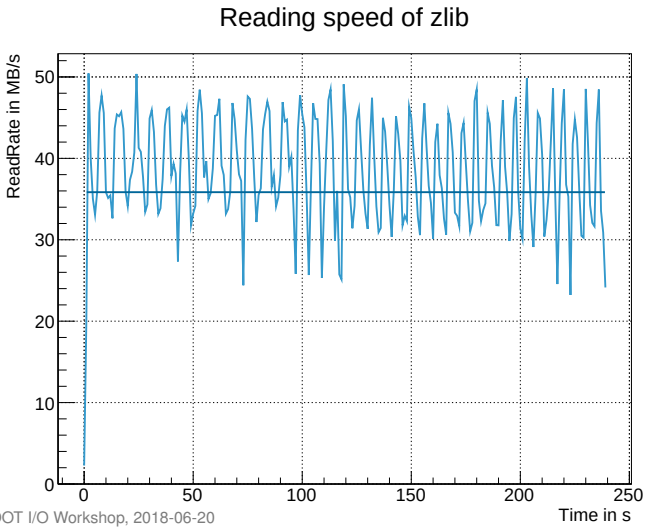
Variable	Options
(Random) Branch skipping	0.0 - 1.0
(Random) Event skipping	0.0 - 1.0
TTreeCache	. . . , 30MB (default), . . .
Read mode	Different modes of variable reading & caching

xAOD IMPLEMENTATION IN ROOT



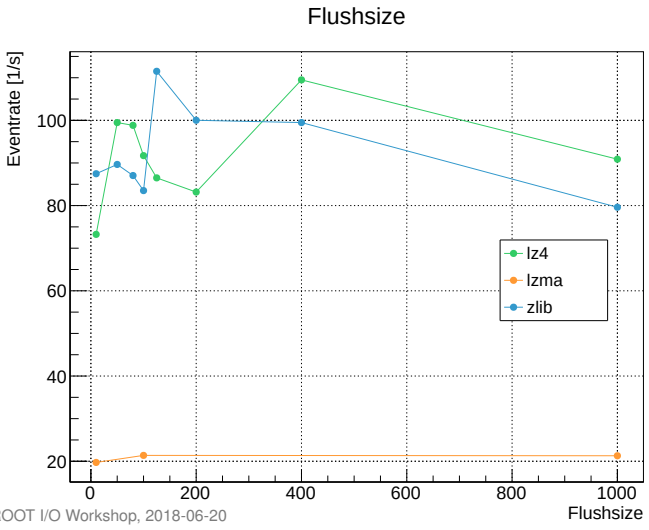
- Basketsize gives compression unit
- Flushed to file, whenever flushsize is met
- $\Rightarrow$ Basketsize $\geq$ Flushsize
- $\mathcal{O}(1000)$ branches & leaves
- TTreeCache (30 MB)

xAOD reading rate



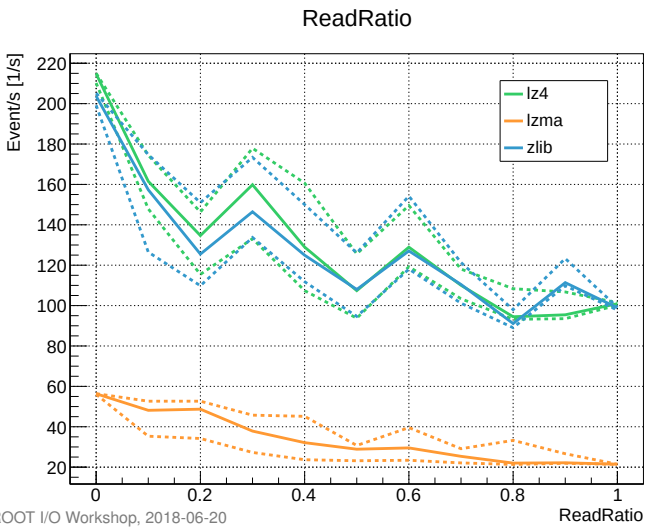
- Accessing 20k Events
- Read all variables of given (large) containers
- Reading rate measured with dstat
- Average reading rate reported by xAOD::PerfStats
- lz4 & lzma in ▸ backup

Flushsize



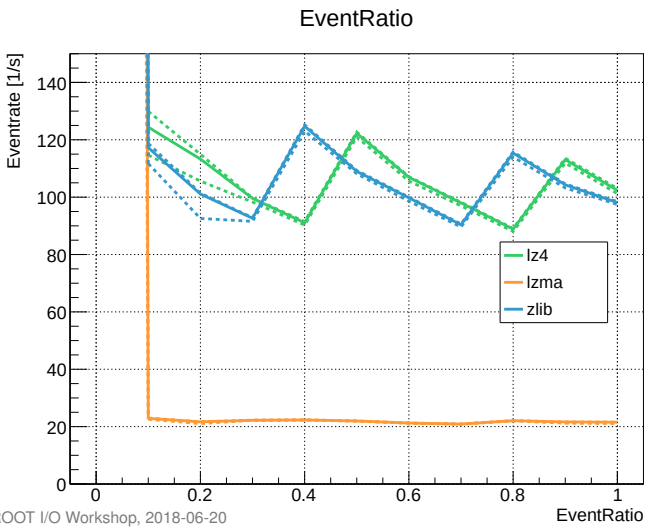
- Event rate for different flushsizes
- High flushsizes could suffer from too small TTreeCache
- Fixed:
  - 20k Events
  - Basketsize 32000
  - Splitlevel 0
  - TTreeCache 30MB
  - ReadRatio 1.0
  - EventRatio 1.0
  - List of read containers

Branch skipping



- Ratio of skipped branches
- max, min & median of 10 runs
- Same branches for all events of that run
- Different set of variables each run
- Fixed:
  - 20k Events
  - Flushsize 100
  - Basketsize 32000
  - Splitlevel 0
  - TTreeCache 30MB
  - EventRatio 1.0
  - List of read containers

Event skipping



- Randomly skipping events
- max, min & median of 10 runs
- Fixed:
  - Same as above
  - ReadRatio 1.0
  - Flushsize 100
- Issues:
  - E.g. EventRatio $= 0.1$, reading every $\frac{1}{10}$th event
  - At flushsize 100 $\Rightarrow$ every basket is uncompressed
  - Plot only meaningful for $\mathit{EventRatio} \leq \frac{1}{\mathit{flushsize}}$
- Pattern for lz4 & zlib still weird

OUTLOOK

- Missing studies:
  - Different configurations for DxAODs
  - Network reading vs. spinning disk & SSD
  - Different splitlevel
  - Comparison of reading modes (Class- vs. AthenaAccess)
- Want to invesitgate ROOTs parallel Branch decompression
- If I forgot any relevant ROOT feature/aspect, please let me know!

BACKUP

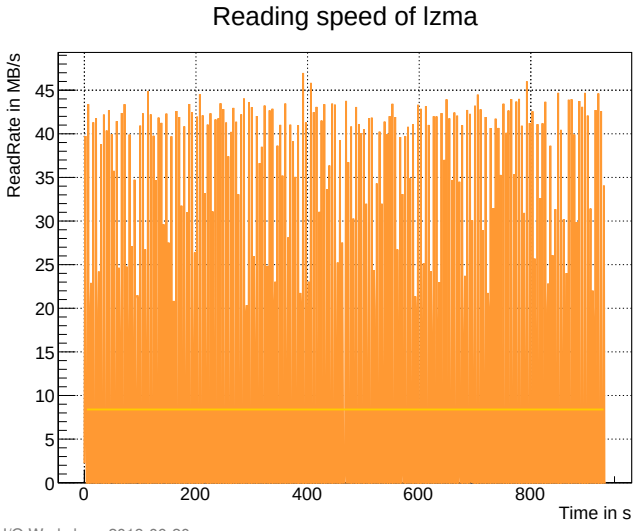
RECONSTRUCTION WALLTIME

Over 100 ttbar events

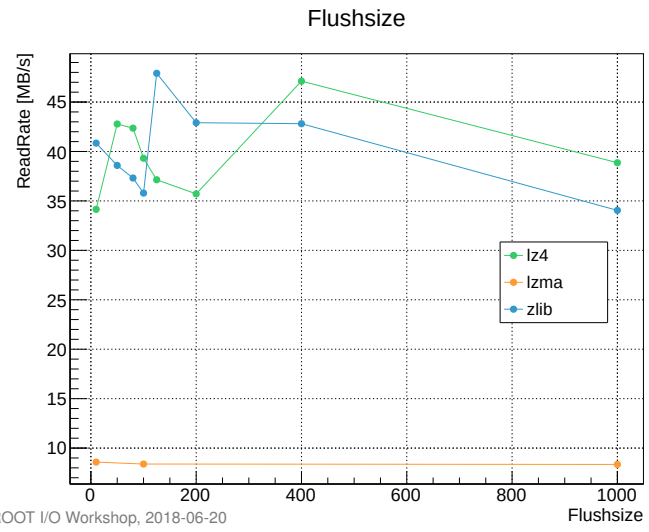
Nevt	Step	Total Read (w/ ROOT and P->T)		ROOT Read fraction	Total Write (w/o compression)		ROOT Write fraction (serialization)	ROOT compression		Total CPU evt-loop time
100	EVNTtoHITS	0.006	0.01%	0.004	0.017	0.02%	0.001	0.027	0.03%	91.986
100	HITtoRDO	1.978	5.30%	0.834	0.046	0.12%	0.001	0.288	0.77%	37.311
100	RDOtoRDO Trigger	0.125	1.23%	0.061	0.153	1.51%	0.014	0.328	3.23%	10.149
100	RAWtoESD	0.166	1.88%	0.081	0.252	2.85%	0.020	0.444	5.02%	8.838
100	ESDtoAOD	0.072	23.15%	0.037	0.147	47.26%	0.013	0.049	15.79%	0.311

READINGRATE OVER TIME

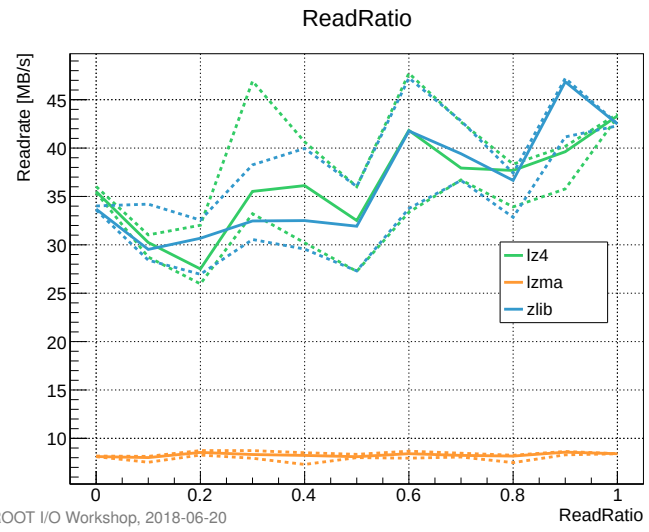
READINGRATE OVER TIME



FLUSHSIZE


Flushsize
ReadRate [MB/s]
45
40
35
30
25
20
15
10
0
200
400
600
800
1000
Flushsize
lz4
lzma
zlib

BRANCH SKIPPING



EVENT SKIPPING