Precision Cascade:

A novel algorithm for multi-precision extreme compression

J. Gonzalez, J. Lauret (PIs)
G. Van Buren, M. Burtscher,
I.A. Cali, Ph. Canal, R. Nunez, Y. Ying

ACAT 2022, Bari, Italy

U.S. DEPARTMENT OF ENERGY | SBIR/STTR Programs Office

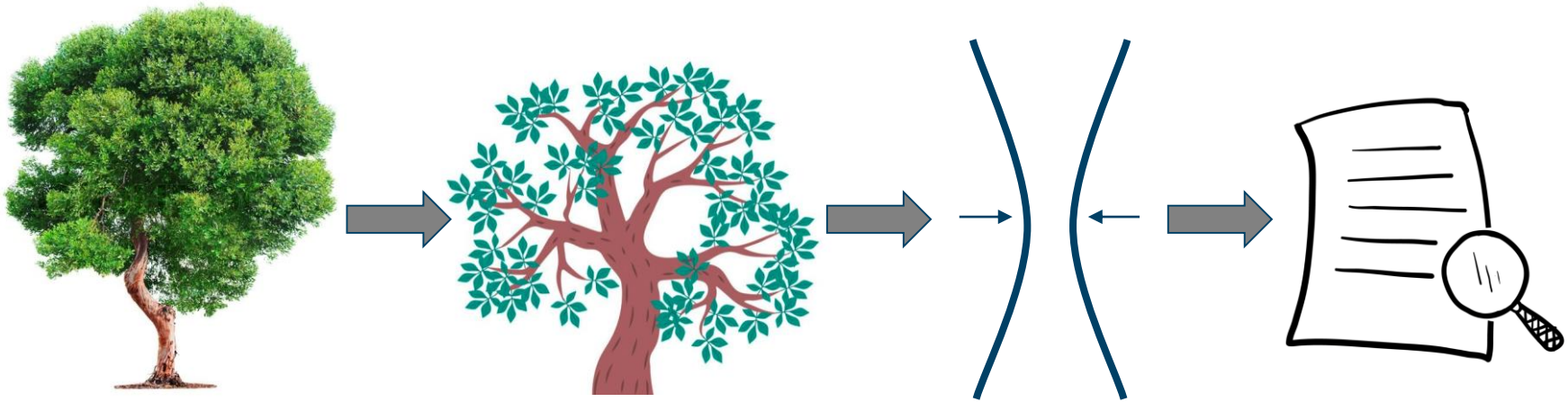
SBIR·STTR America's Seed Fund

Compression Algorithms

- **Lossy compression algorithms** are used in image/sound processing
  - Why not physics?
- Accelogic - theory of **“Compressive Computing”**
  - Offering stunning **lossy** and **lossless algorithms**
- **Concept**: not a true “loss” to remove bits that carry insignificant or zero information

ROOT
Data Analysis Framework

Compression Algorithms



Dataset

Extracted branches

Apply compression (**BLAST**)

Assess results

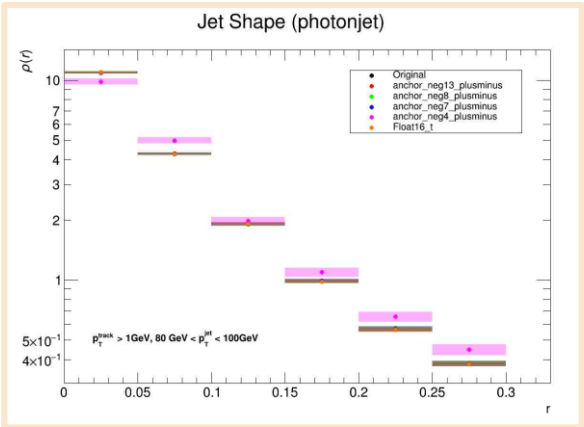
accelogic Compression Results

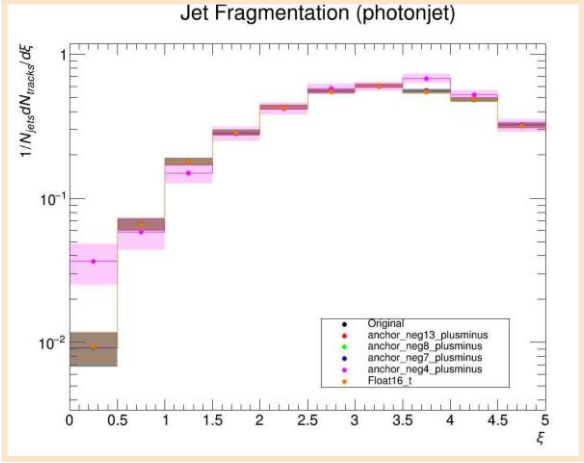
- Previous **outside-of-ROOT** results surpassing current ROOT algorithms
- Prior results **outperform gzip** by *2-4x*
- **Concerns** over too much precision irretrievably lost

Compression Ratios

Comp:	Low (48)	Mid (53)	Mid (54)	High (57)	gzip	float16
pT	4.25	6.24	6.88	9.95	1.97	2
eta	3.75	5.27	5.75	7.86	1.95	2
phi	4.15	6.04	6.65	9.54	2.02	2
mass	14.95	17.25	18.12	22.02	3.69	2
Overall	**6.13**	**8.32**	**8.98**	**11.81**	**2.95**	**2.63**

(Overall includes lossless int compression)

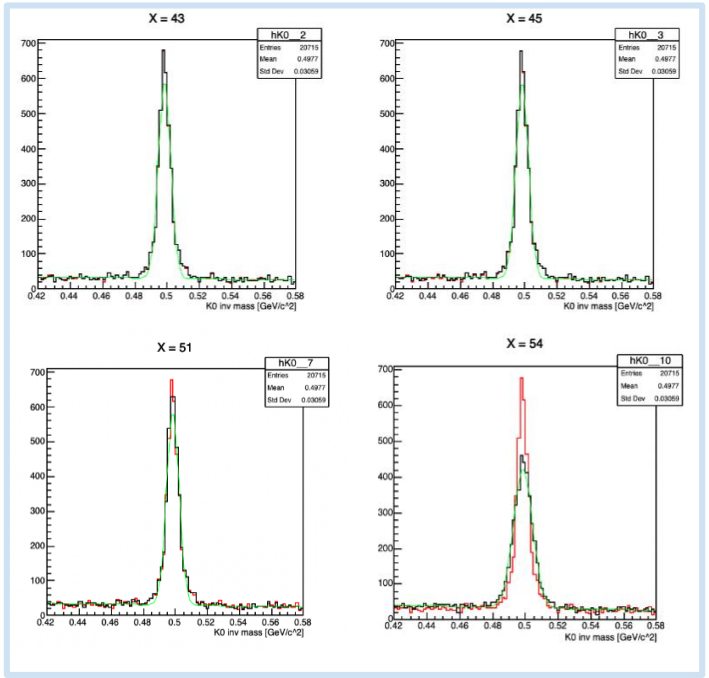




Compression Results

STAR

- Previous **inside-of-ROOT** results surpassing current ROOT algorithms
- Prior results **outperform gzip** by *2-4x*
- **Concerns** over too much precision irretrievably lost

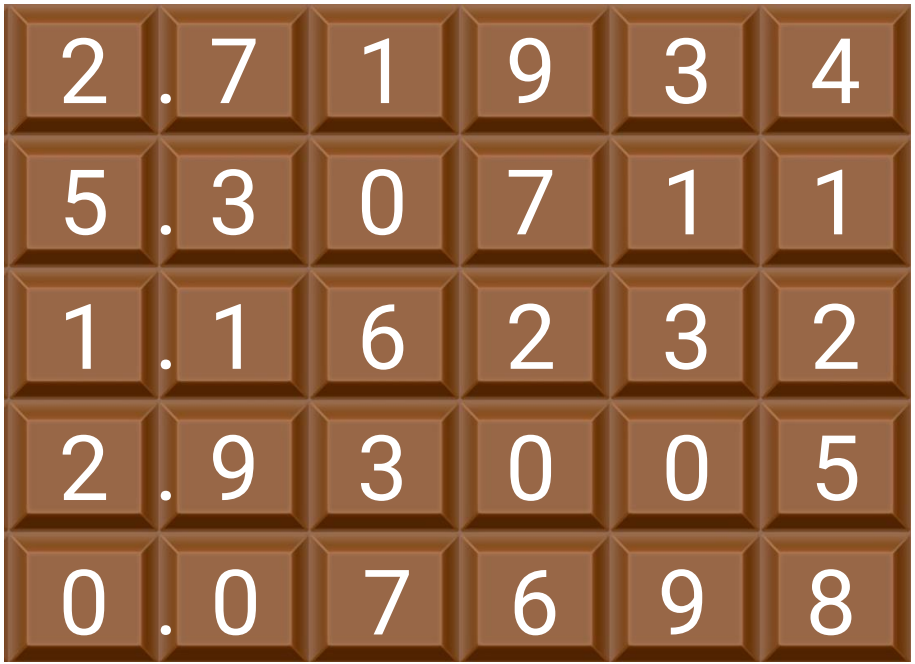


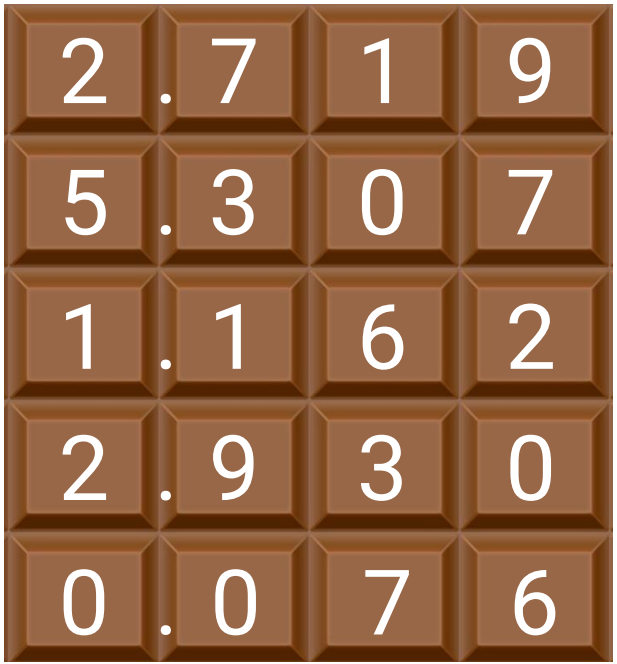
STAR Compression Ratios

Compression:	Low (43)	Mid (51)	ROOT default
px_primary	2.68	5.14	1.83
px_secondary	1.44	2.95	1.07
x	1.12	1.88	1.05
z	1.95	1.60	1.10

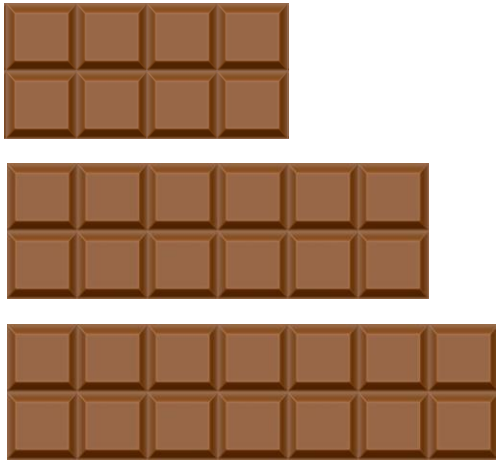
Can we **store tiers of precision** without being repetitive?

...
2.71934
5.30711
1.16232
2.93005
0.07698
...

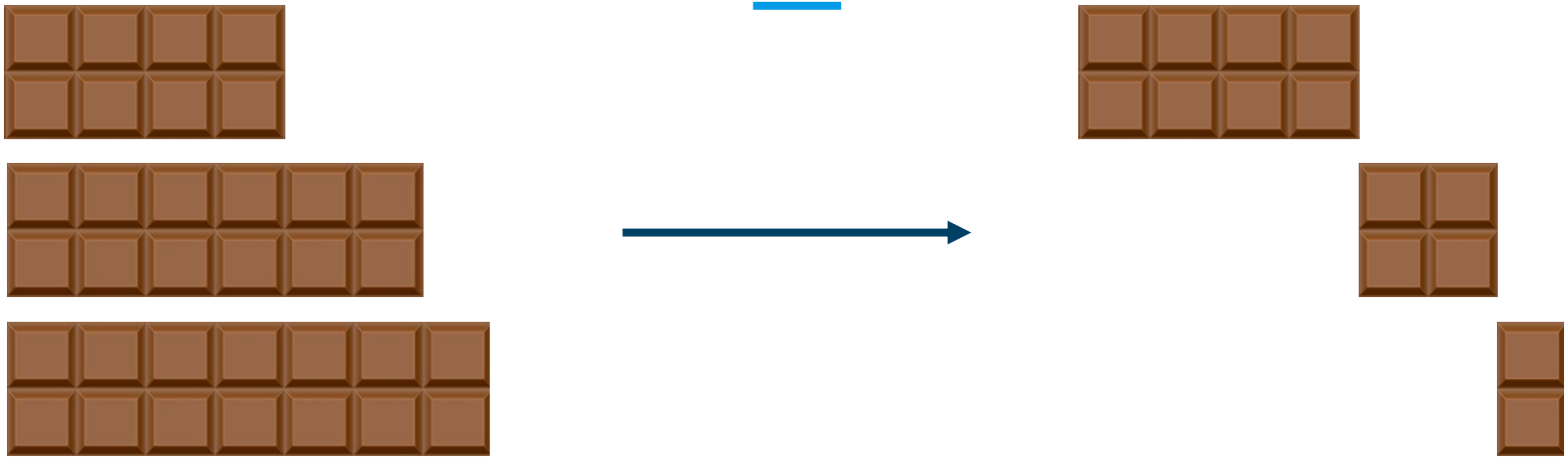

2 . 7 1 9 3 4
5 . 3 0 7 1 1
1 . 1 6 2 3 2
2 . 9 3 0 0 5
0 . 0 7 6 9 8


2 . 7 1 9
5 . 3 0 7
1 . 1 6 2
2 . 9 3 0
0 . 0 7 6

Can we **store tiers of precision** without being repetitive?



Can we **store tiers of precision** without being repetitive?

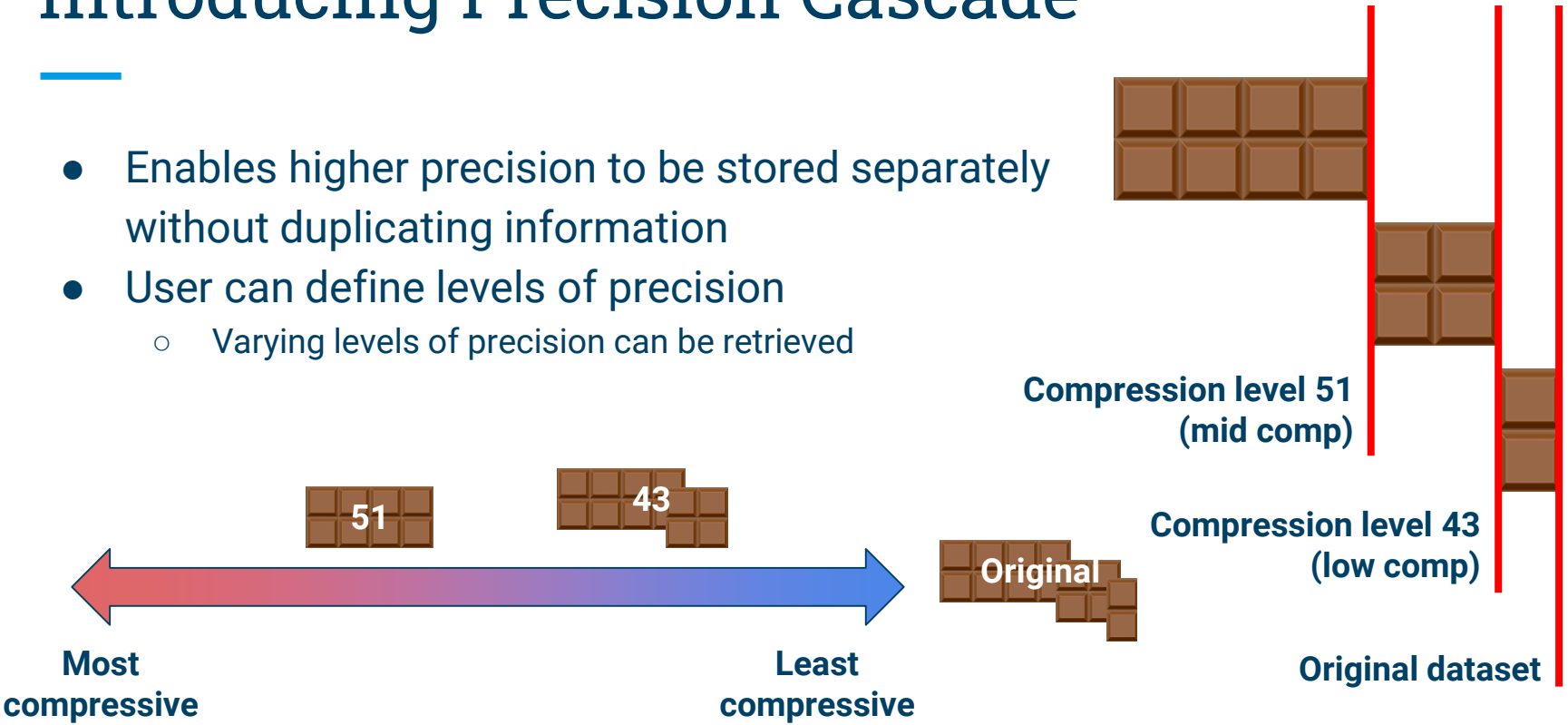


Can we **store tiers of precision** without being repetitive?

YES.

Introducing Precision Cascade

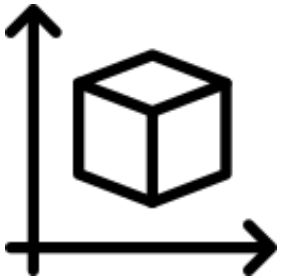
- Enables higher precision to be stored separately without duplicating information
- User can define levels of precision
  - Varying levels of precision can be retrieved



Testing Precision Cascade in ROOT

- Tested inside-of-ROOT on subset of **CMS**-like and **STAR** data
- Compared against inside-of-ROOT BLAST and default ROOT algorithm
- Assessing:

compression ratio

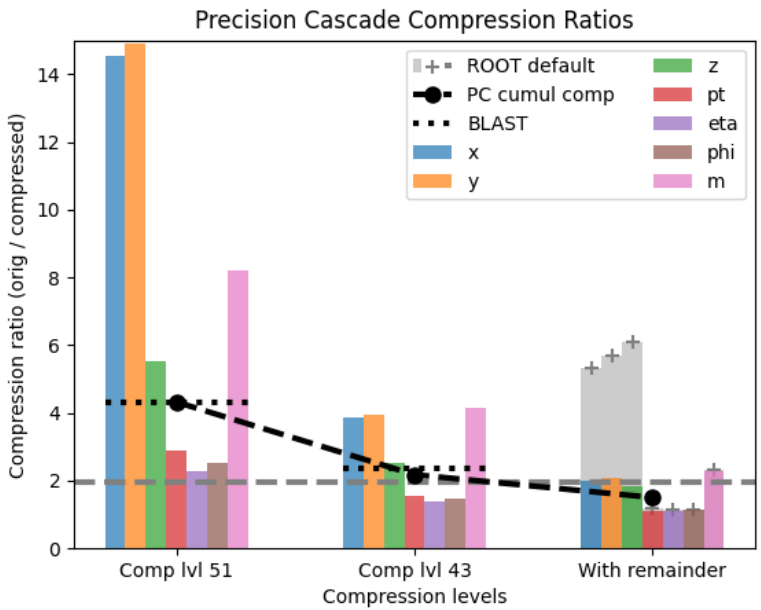


compression speed

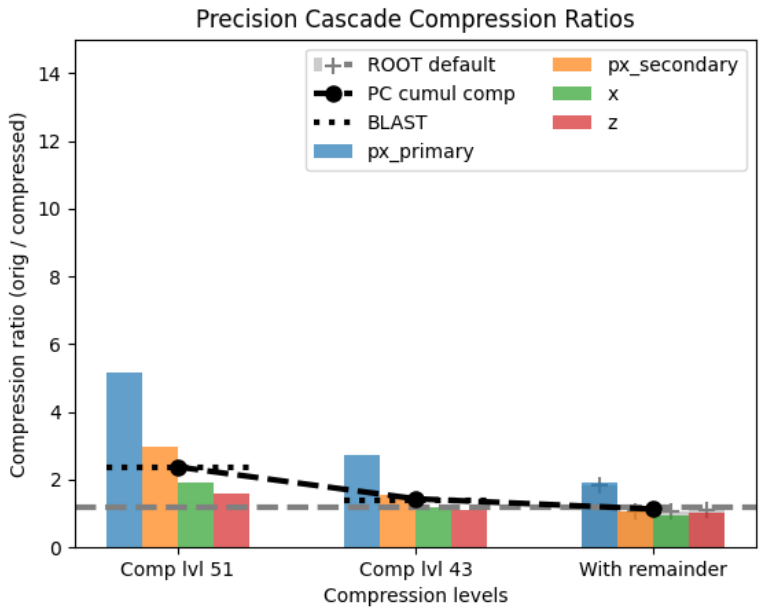
decompression speed

Compression Ratio Results

CMS

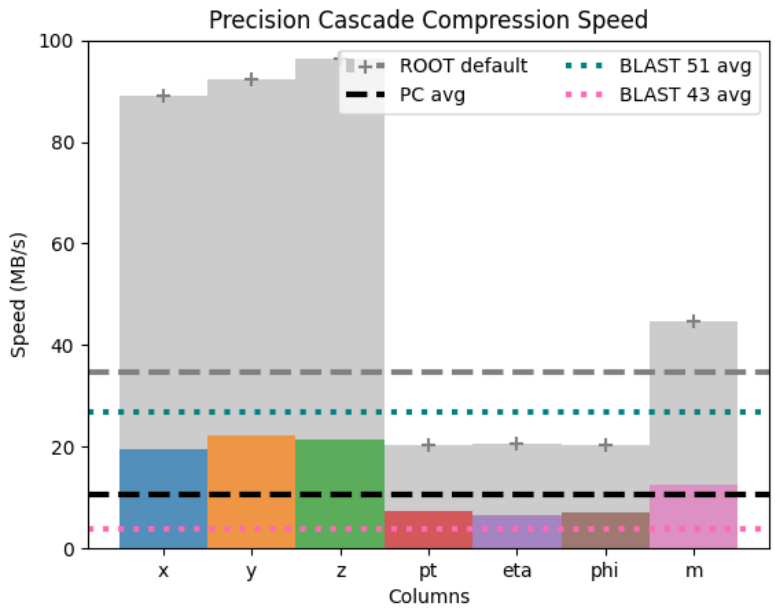


STAR

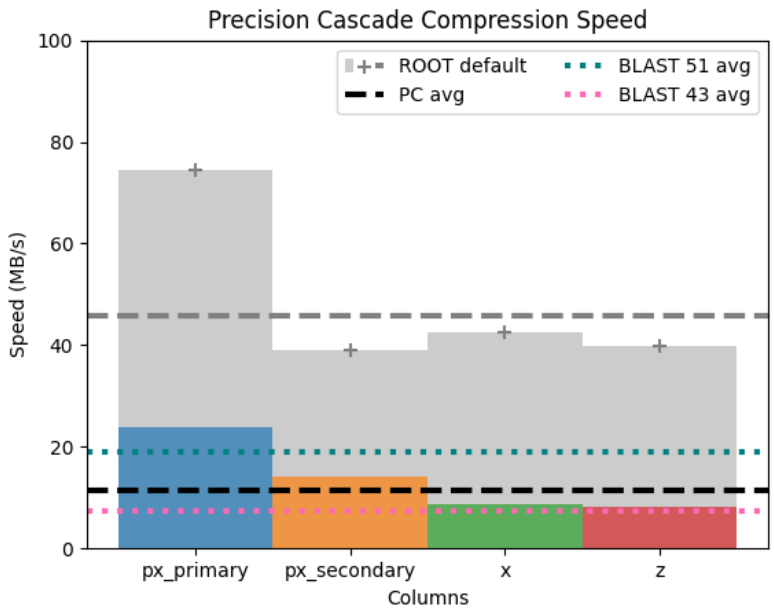


Compression Speed Results

CMS

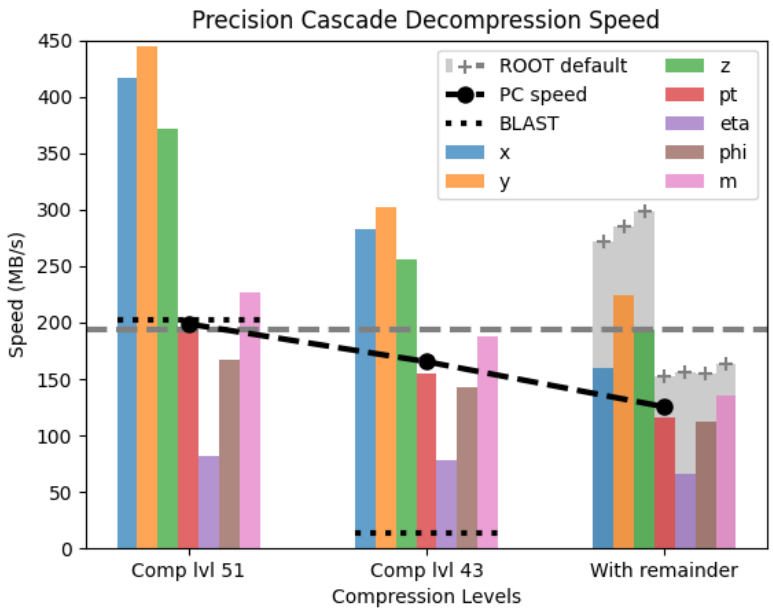


STAR

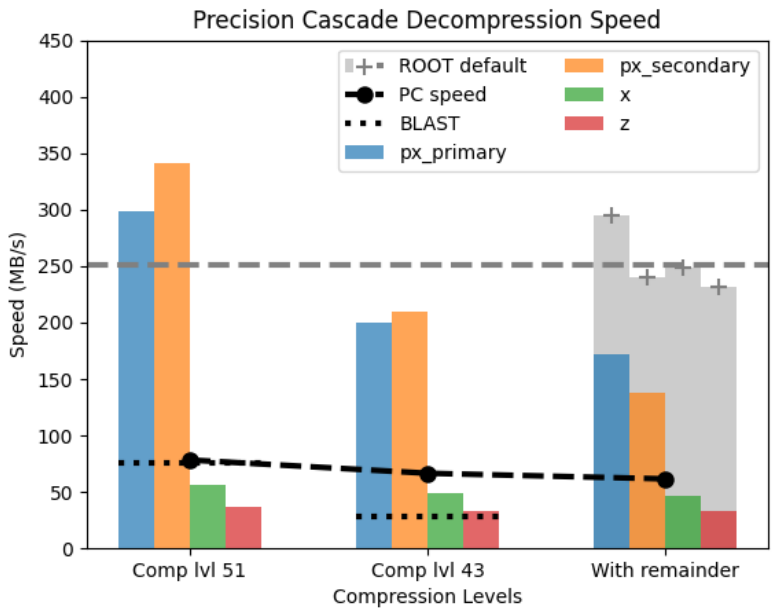


Decompression Speed Results

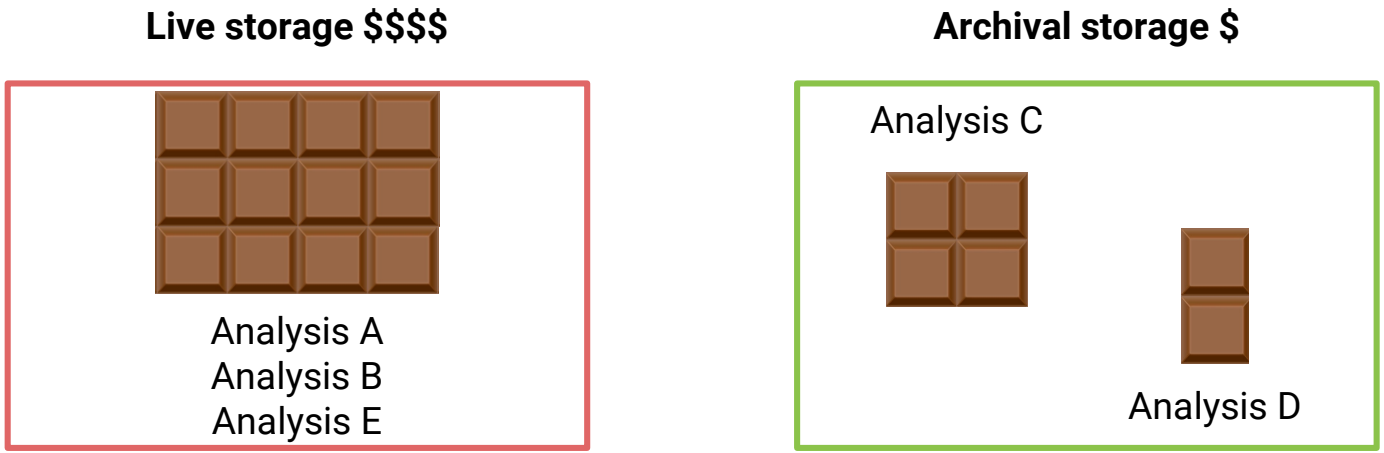
CMS



STAR



Leveraging Precision Cascade



Benefits of Precision Cascade

- Allows storing highly compressed data in **fast access storage**, rest in archival storage
- Certain analysis require **fewer statistics** but **more precision**
- Later on, can **rebuild original dataset** again, if necessary

Integration in ROOT

- BLAST can be applied to all branches containing homogeneous numerical types (e.g. split branches)
  - BLAST is lossy for floating point branches and lossless for integer branches
  - Double32_t and Float16_t which are already lossy are not supported by BLAST
- Precision Cascade is supported only for double and float

```
std::vector<Int_t> levels = { 51, 43 };
ROOT::PrecisionCascadeCompressionConfig targetConfig(
    ROOT::RCompressionSetting::EAlgorithm::kBLAST,
    levels,
    true /* Keep also the residual file */ );
…
lossy_branch->SetCompressionSettings(targetConfig);
```

Integration in ROOT

- Precision Cascade naming scheme
  - The additional files are named with their cascade tier (the suffix is customizable)
    - *${original_filename}*_precisioncascade_[1,2,3,*etc.*].root
- The cascade files are automatically used if present
  - For example immediately after writing, reading back would be done with full precision

Output Folder

original_file.root

original_file.root_precisioncascade_1.root

original_file.root_precisioncascade_2.root

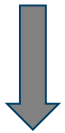
Integration in ROOT

- Precision Cascade naming scheme
  - The additional files are named with their cascade tier (the suffix is customizable)
    - *${original_filename}*_precisioncascade_[1,2,3,*etc.*].root
- The cascade files are automatically used if present
  - For example immediately after writing, reading back would be done with full precision

Output Folder

original_file.root

original_file.root_precisioncascade_1.root



Compression Level 43

Open Issues

- **No valid source-code license yet**
  - License would grant **free unlimited permission** for ROOT to use and integrate the codes for non-profit / academic communities (i.e. redistribute sources as part of ROOT releases)
- In the interim, binary library distribution are available for early adopter

In Conclusion

- BLAST demonstrates outperformance of ROOT compression in many cases, allowing good enough precision for certain analyses
- **Key finding**: Precision Cascade allows user to save "lost" bits in separate file
  - Builds end-user's ease and confidence in using lossy compression algorithms
- We are hoping to release these compression algorithms in ROOT soon to the community, **stay tuned!**

This work was supported by the U.S. Department of Energy, SBIR/STRR Program of the Office of science, Nuclear Physics under Award Number DE-SC0018521

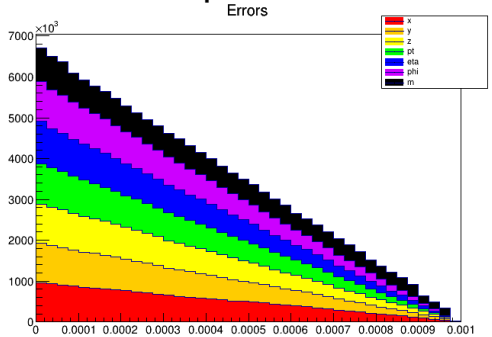
Thank you!

This work was supported by the U.S. Department of Energy, SBIR/STRR Program of the Office of science, Nuclear Physics under Award Number DE-SC0018521

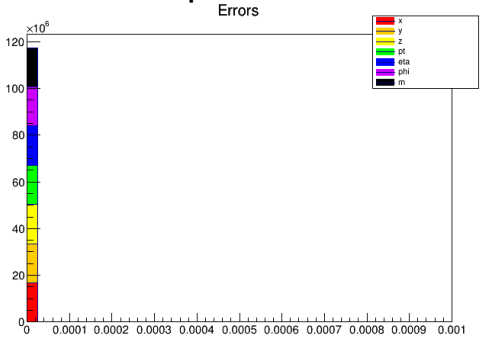
Backup Slides

Error of Compression Levels (Stacked)

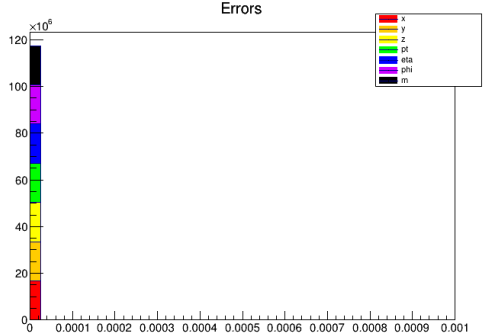
Comp. Level 51



Comp. Level 43

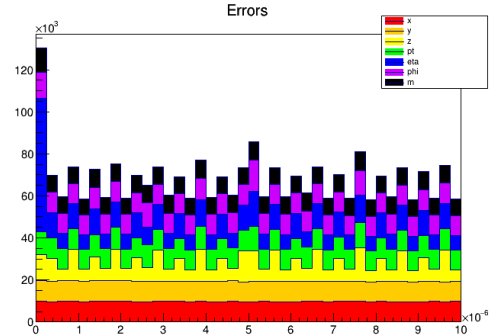


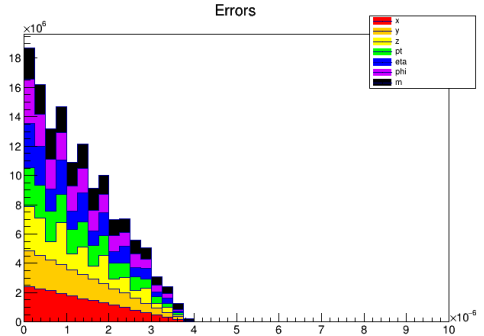
With remainder

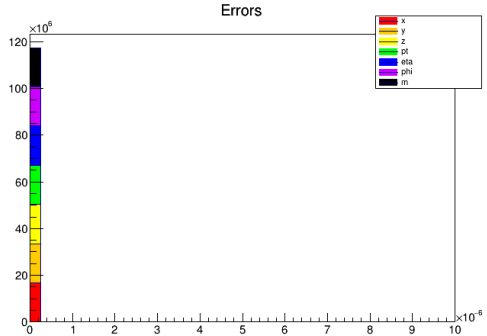


N

X-axis:
[0, 1e-3]







N

X-axis:
[0, 1e-5]

|orig - zig|

|orig - zig|

|orig - zig|

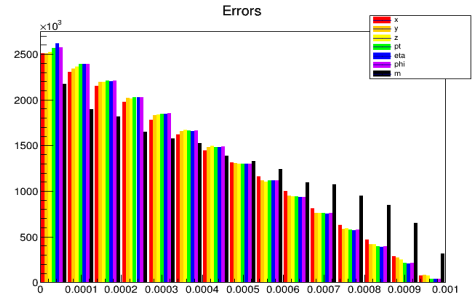
Error of Compression Levels (Unstacked)

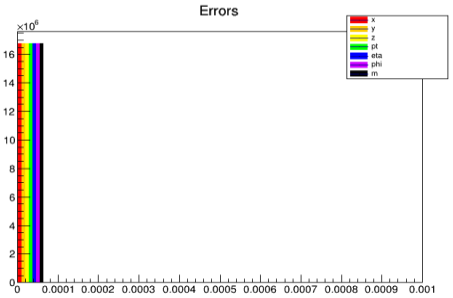
Comp. Level 51

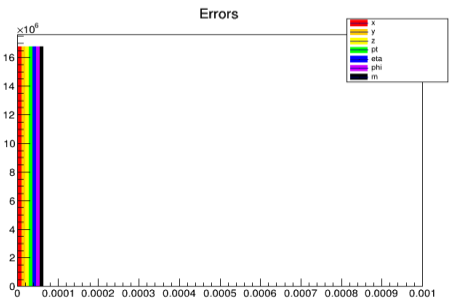
Comp. Level 43

With remainder

N

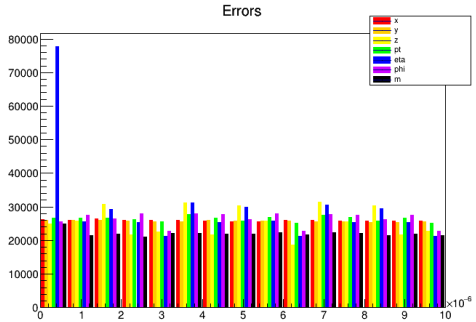


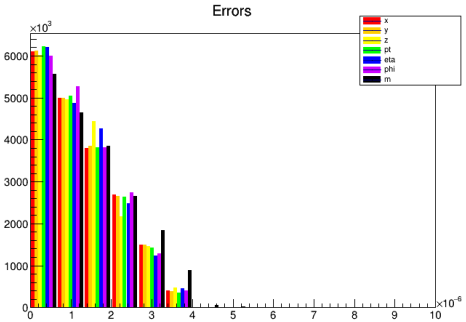


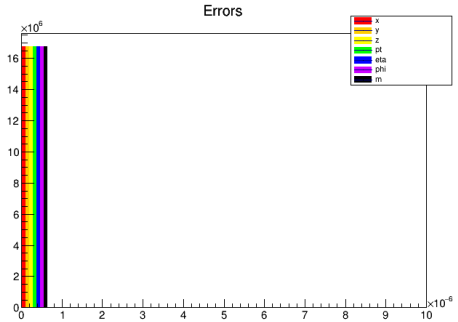


X-axis:
[0, 1e-3]

N







X-axis:
[0, 1e-5]

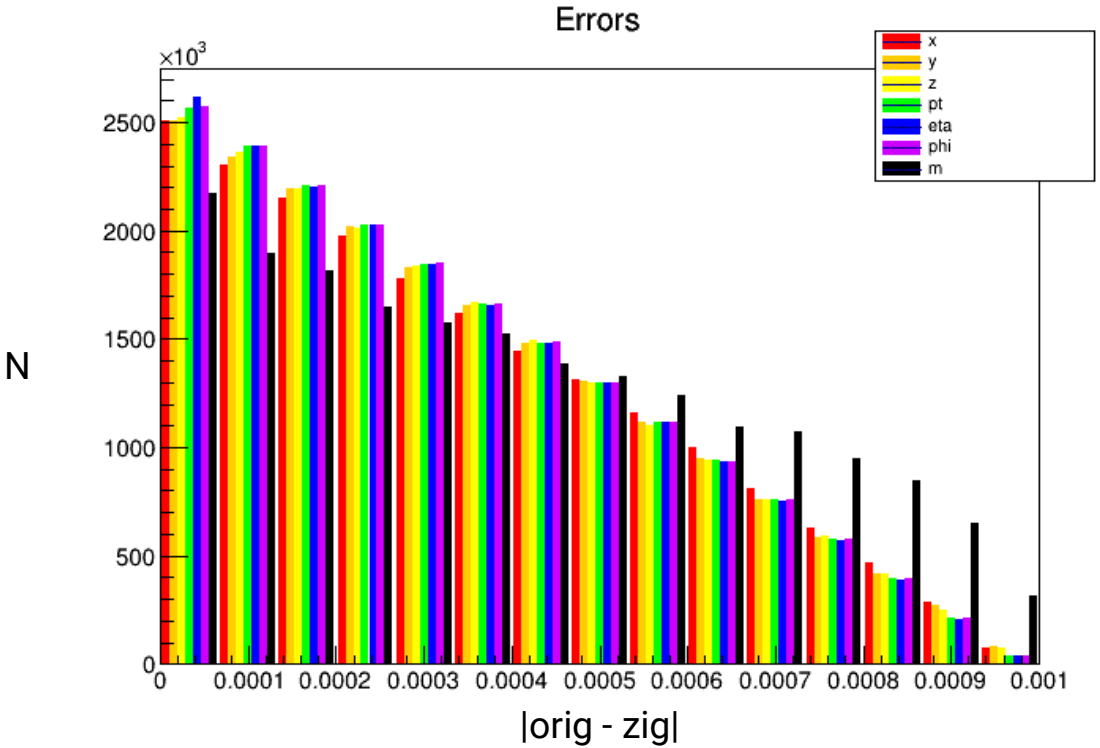
|orig - zig|

|orig - zig|

|orig - zig|

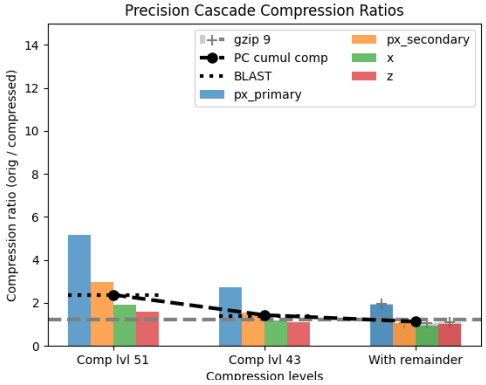
Error of Compression Levels (Unstacked)

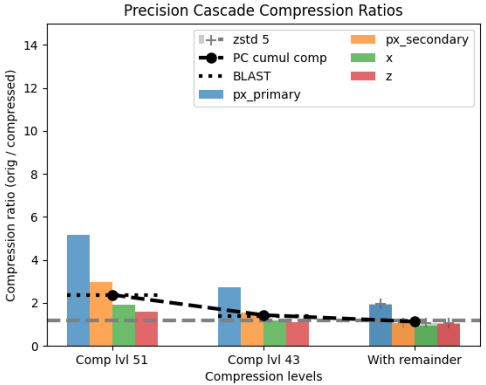
Comp. Level 51

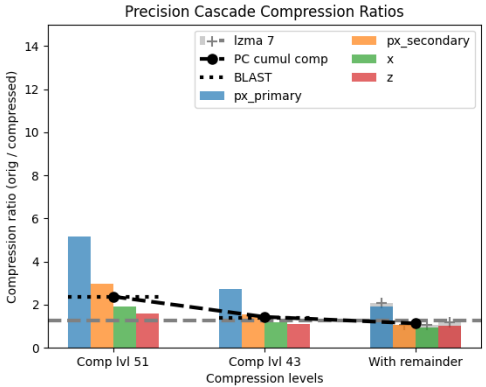


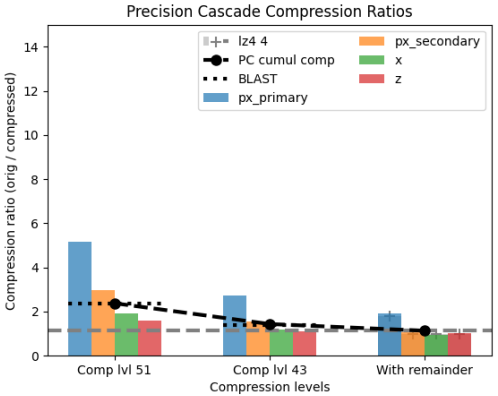
X-axis:
[0, 1e-3]

Compression Ratio - Additional Baselines

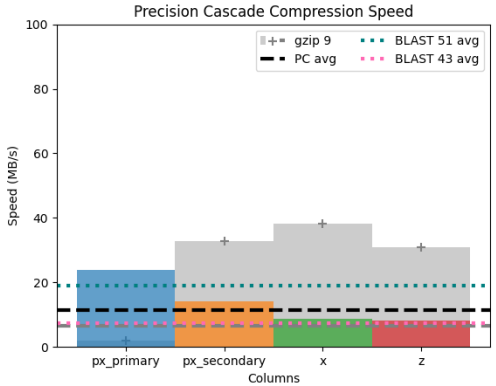


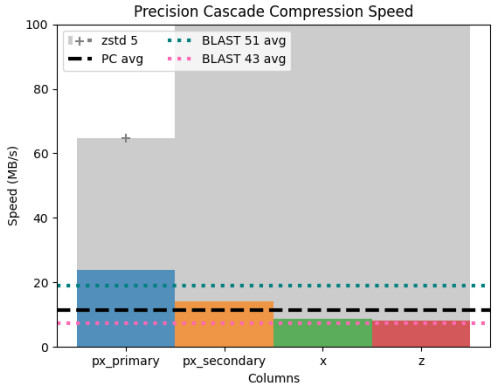


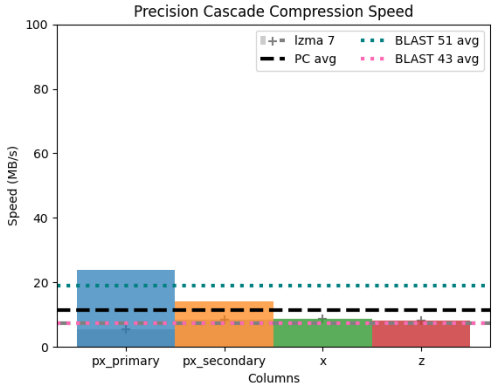


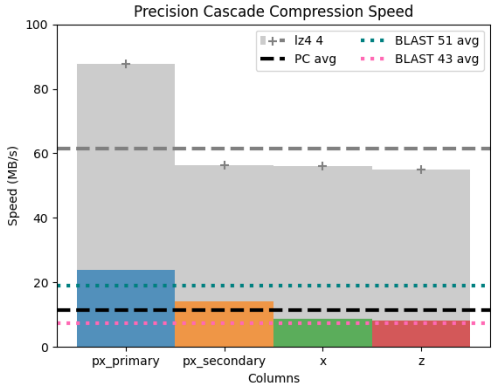


Compression Speed - Additional Baselines

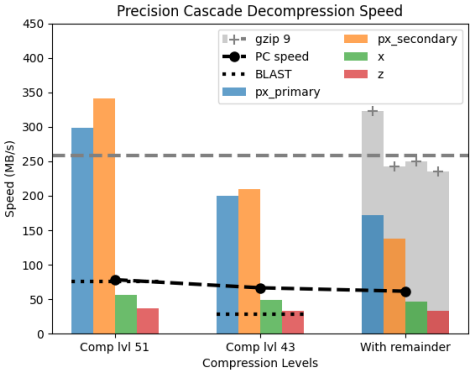


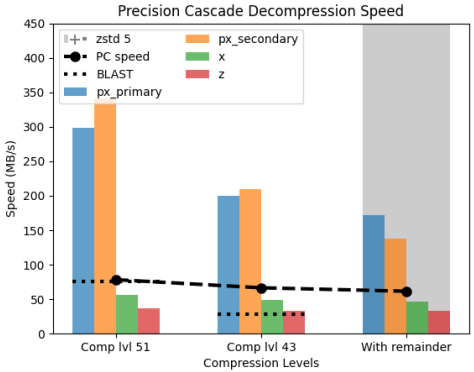


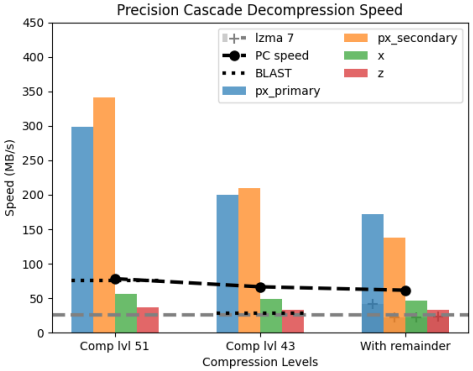


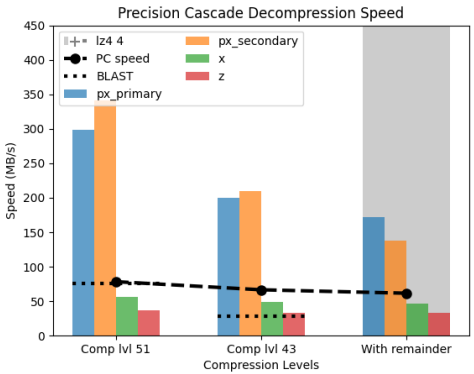


Decompression Speed - Additional Baselines









Precision Cascade Data Types

- Lossy compression
  - Float
  - Double
- Lossless compression
  - Int
  - UInt
  - Long
  - etc

Edge Cases Considered

- We have tested the following edge cases:
  - Applying ZIG to a tree of only integers - should mention that ZIG should not be used for this, but it should still work (and that the auxiliary files are unnecessary for reading the integers)
  - Total size vs. number of tiers
  - Total size vs. buffer sizes (note that ROOT automatically determines buffer sizes unless overridden)