Optimizing Data Storage: A Parquet Benchmarking Study

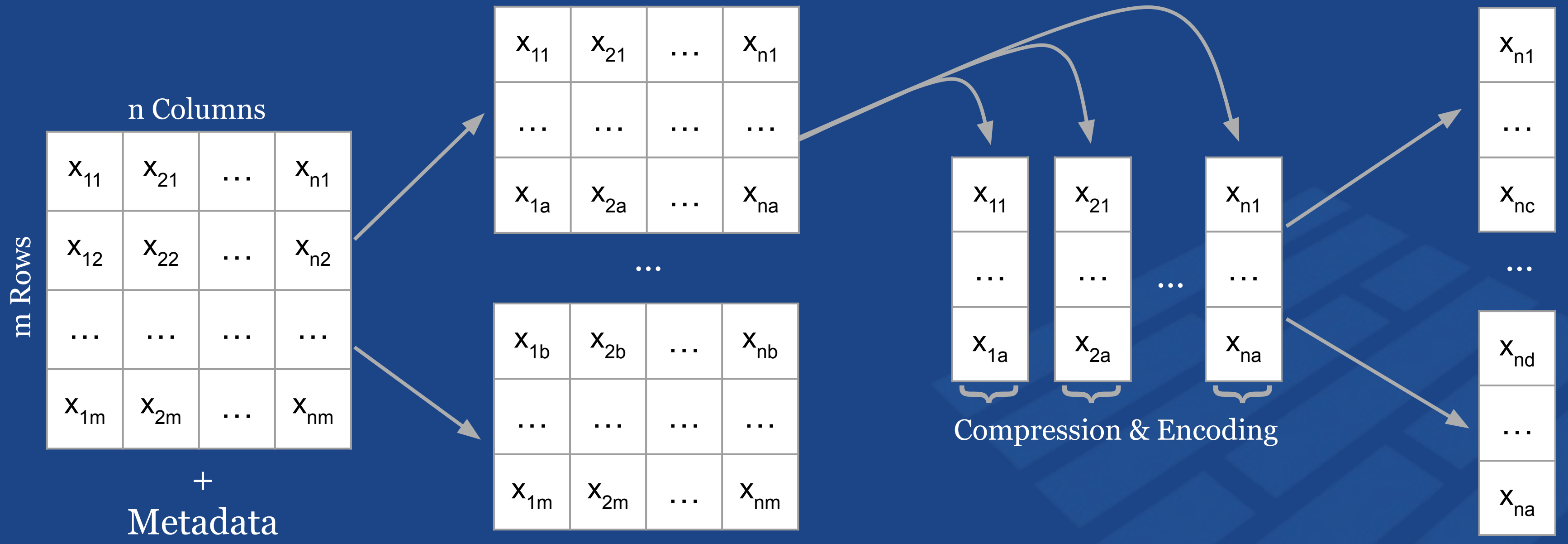
Evaluating Parquet Runtime and Output Size Across Diverse Compression and Encoding Formats

Josh Lim

August 27th, 2024 LA-UR-24-31733

Parquet Structure

Columnar Storage Hierarchy



n Columns
m Rows
x_{11} x_{21} … x_{n1}
x_{12} x_{22} … x_{n2}
… … … …
x_{1m} x_{2m} … x_{nm}
+
Metadata
x_{11} x_{21} … x_{n1}
… … … …
x_{1a} x_{2a} … x_{na}
…
x_{1b} x_{2b} … x_{nb}
… … … …
x_{1m} x_{2m} … x_{nm}
x_{11} … x_{1a}
x_{21} … x_{2a}
…
x_{n1} … x_{na}
Compression & Encoding
x_{n1} … x_{nc}
…
x_{nd} … x_{na}

Parquet File → One or more → Row Group(s) → One per column (n) → Column Chunks → One or more → Data Page(s)

Parquet File
Size: table size + overhead

Row Group(s)
Size: min(table size, 1024^2 rows size) (Default)

Column Chunks
Size: column type * row group rows

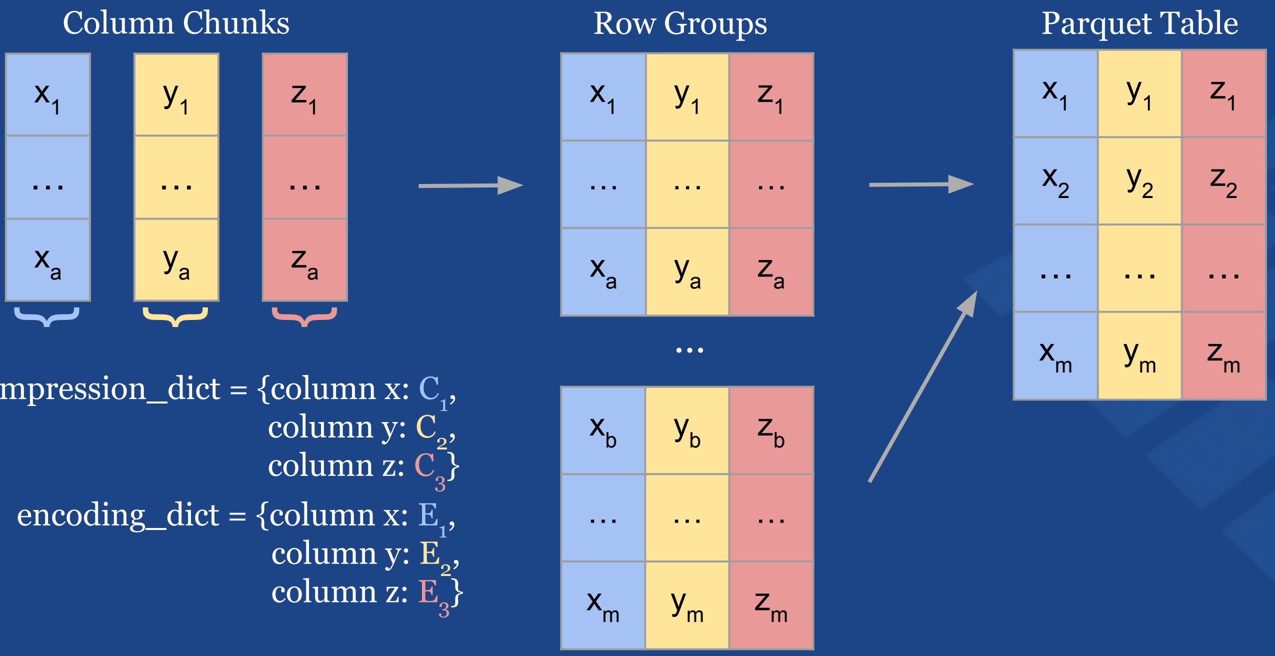
Data Page(s)
Size: 1 MiB (Default)

Column Based Implementation

Columnar Storage Benefits

```
parquet.write_table(data, compression=compression_dict, col_encoding=encoding_dict)
```

|— Schema: x, y, z = various data types



compression_dict = {column x: C_1,
column y: C_2,
column z: C_3}

encoding_dict = {column x: E_1,
column y: E_2,
column z: E_3}

- Assign compression and encoding formats per column
  - Applied at column chunk level - across all data page(s)

Supported Formats

Columnar Storage Benefits

Compressions

- None
- Snappy
- Brotli
- Gzip
- ZSTD
- LZ4

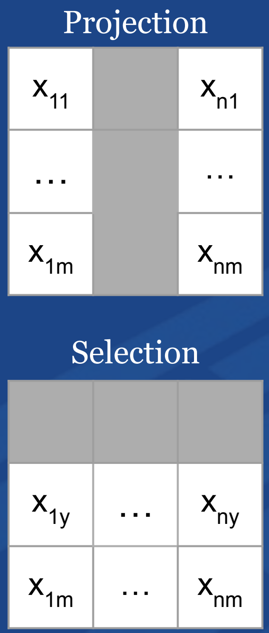
Encodings

- Plain
- Byte Stream Split
- RLE/Bit-Packing
- Plain Dictionary
- RLE Dictionary
- Delta Byte Array
- Delta Binary Packed
- Delta Length Byte Array

Efficient Query and Retrieval

Columnar Storage Benefits

```
Parquet File
|— Header (magic number “PAR1”)
|— Row Group 1
|     |— Column Chunk: Column 1
          …
|     |__ Column Chunk: Column n
…
|— Row Group r
|     |— Column Chunk: Column 1
          …
|     |__ Column Chunk: Column n
|__ Footer
      |— Metadata (version, schema, row group offsets, etc.)
      |— Row Group Metadata
      |     |— Number of rows
      |     |— Total byte size
      |     |— Column Chunk Metadata
      |     |     |— Statistics (min/max values, null count, etc.)
      |     |     |__ Format (codec, encoding, etc.)
      |__ Offset to Metadata
```



Projection

x_{11}		x_{n1}
…		…
x_{1m}		x_{nm}

- Access subset of columns.
- Reads select column chunks across all row groups from disk.

Selection

x_{1y}	…	x_{ny}
x_{1m}	…	x_{nm}

- Filter rows by comparing predicate to row group metadata statistics.
- Reads select row groups from disk.

VPIC Data

LANL C2-VPIC Sample Dataset

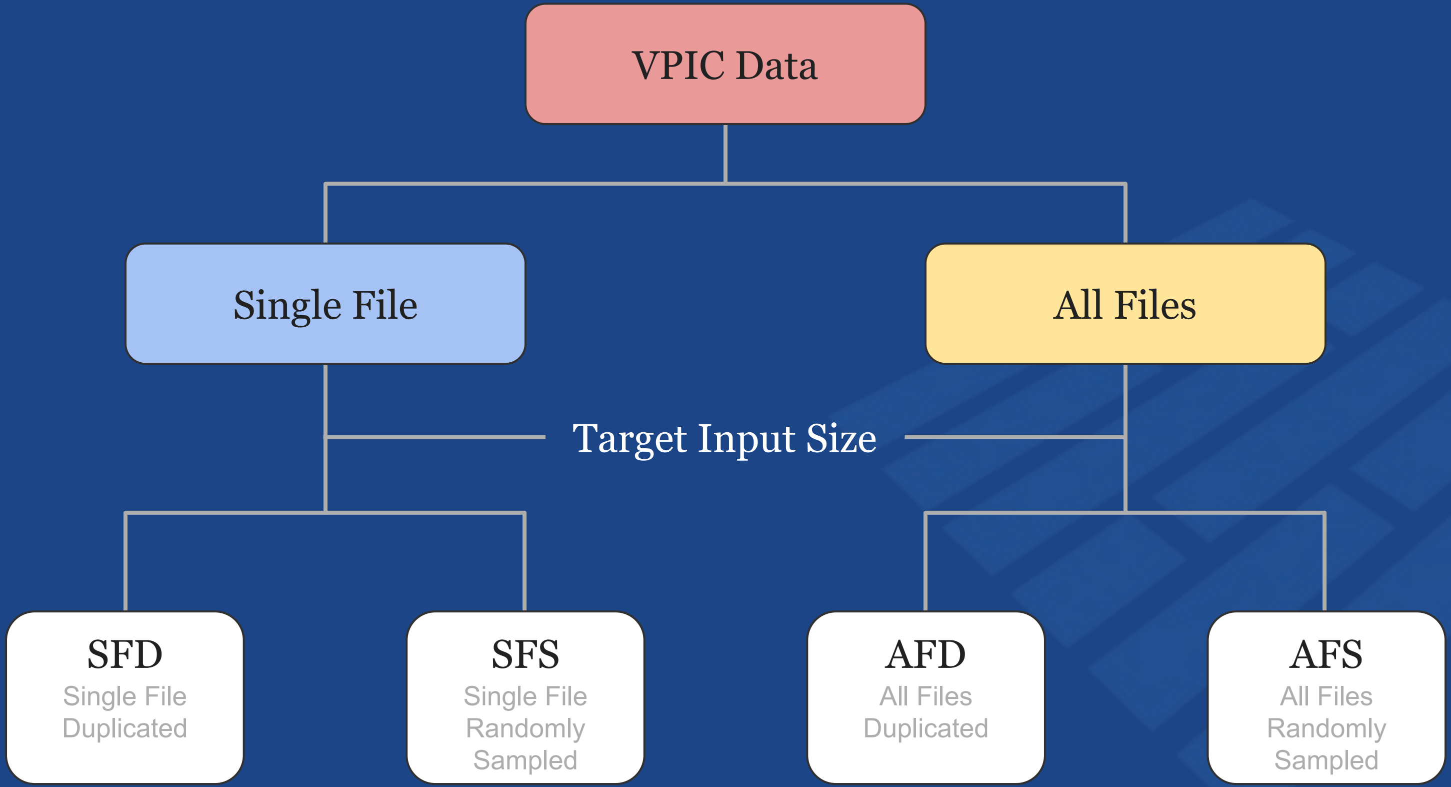
Particle Schema

8 bytes	uint64_t	**ID**	unique ID of a particle
8 bytes	uint64_t	**padding**	
4 bytes	float	**x**	location of particle in X direction
4 bytes	float	**y**	location of particle in Y direction
4 bytes	float	**z**	location of particle in Z direction
4 bytes	float	**i**	index of the cell that had the particle
4 bytes	float	**ux**	momentum of particle in X direction
4 bytes	float	**uy**	momentum of particle in Y direction
4 bytes	float	**uz**	momentum of particle in Z direction
4 bytes	float	**ke**	kinetic energy of particle

- 48 bytes per particle
- 128 * 1024 particles per file
- 6 MiB per file
- 42 total files
- 252 total MiB

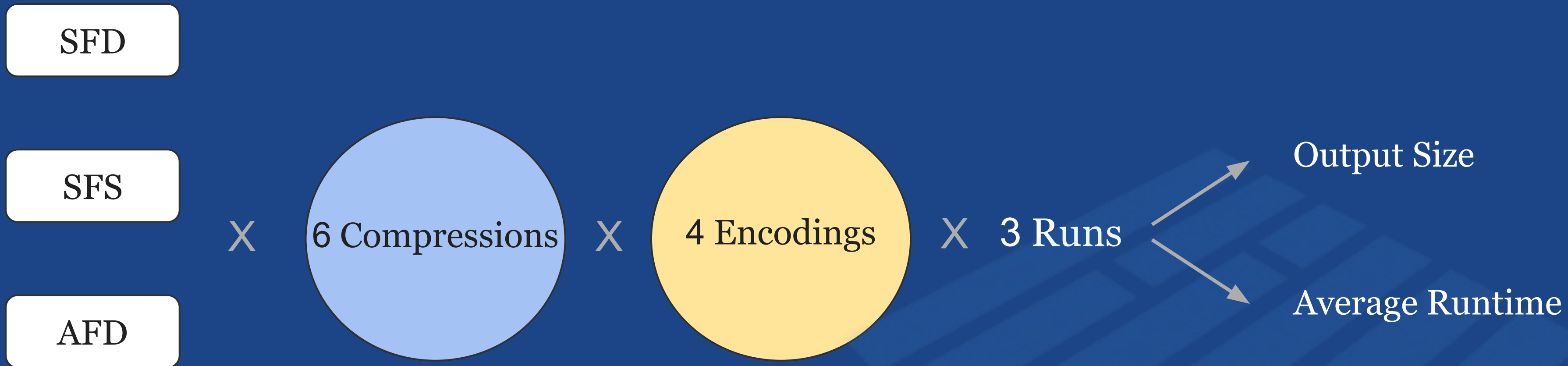
VPIC Input Data Format

Benchmark Set Up



VPIC Benchmark Framework

Writing Input Data Into Parquet



SFD

SFS

AFD

AFS

X 6 Compressions X 4 Encodings X 3 Runs

Output Size

Average Runtime

- Compressions: None, Snappy, Gzip, Brotli, LZ4, ZSTD
- Encodings: Plain, Plain Dictionary, RLE Dictionary, Byte Stream Split

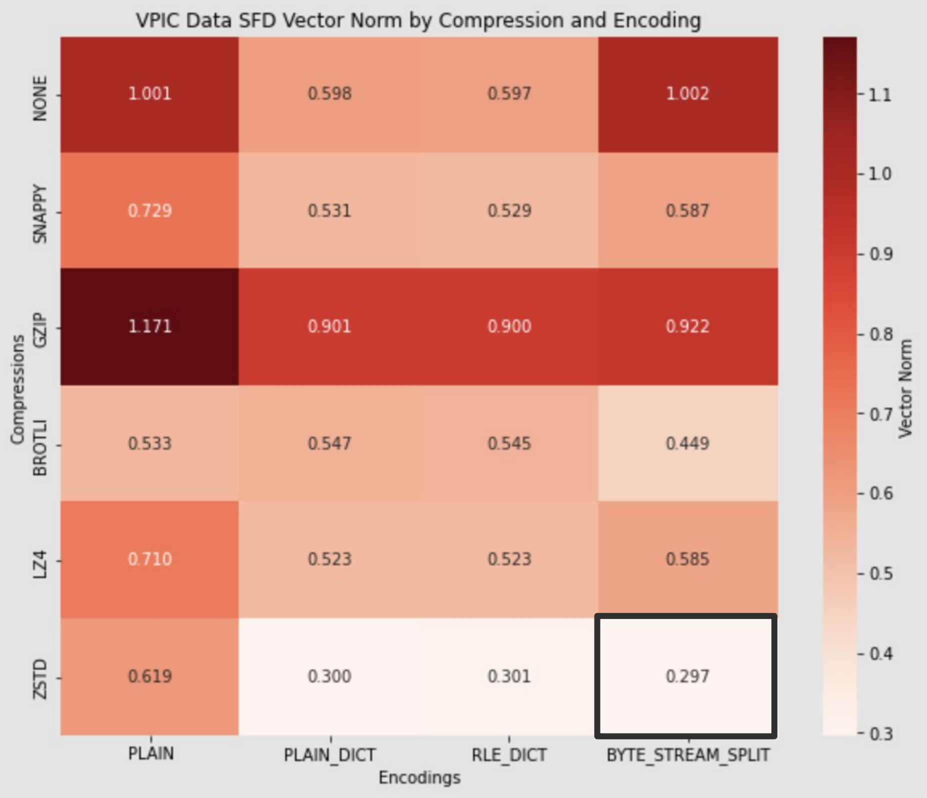
Vector Norm

Analyzing Benchmark Results

$$\sqrt{\left(\frac{\text{Output Size}_i}{\max(\text{Output Size})}\right)^2 + \left(\frac{\text{Average Runtime}_i}{\max(\text{Average Runtime})}\right)^2}$$

Single File Duplicated Results

Benchmark Analysis With VPIC Data

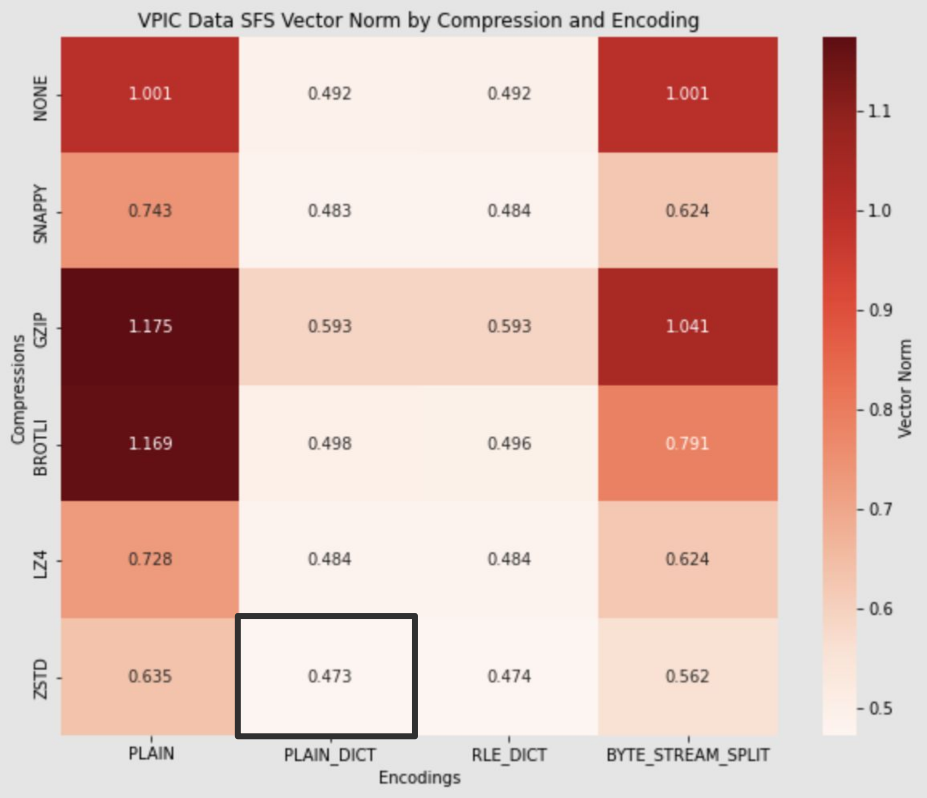


VPIC Data SFD Vector Norm by Compression and Encoding

Compressions	PLAIN	PLAIN_DICT	RLE_DICT	BYTE_STREAM_SPLIT
NONE	1.001	0.598	0.597	1.002
SNAPPY	0.729	0.531	0.529	0.587
GZIP	1.171	0.901	0.900	0.922
BROTLI	0.533	0.547	0.545	0.449
LZ4	0.710	0.523	0.523	0.585
ZSTD	0.619	0.300	0.301	0.297

Single File Sampled Results

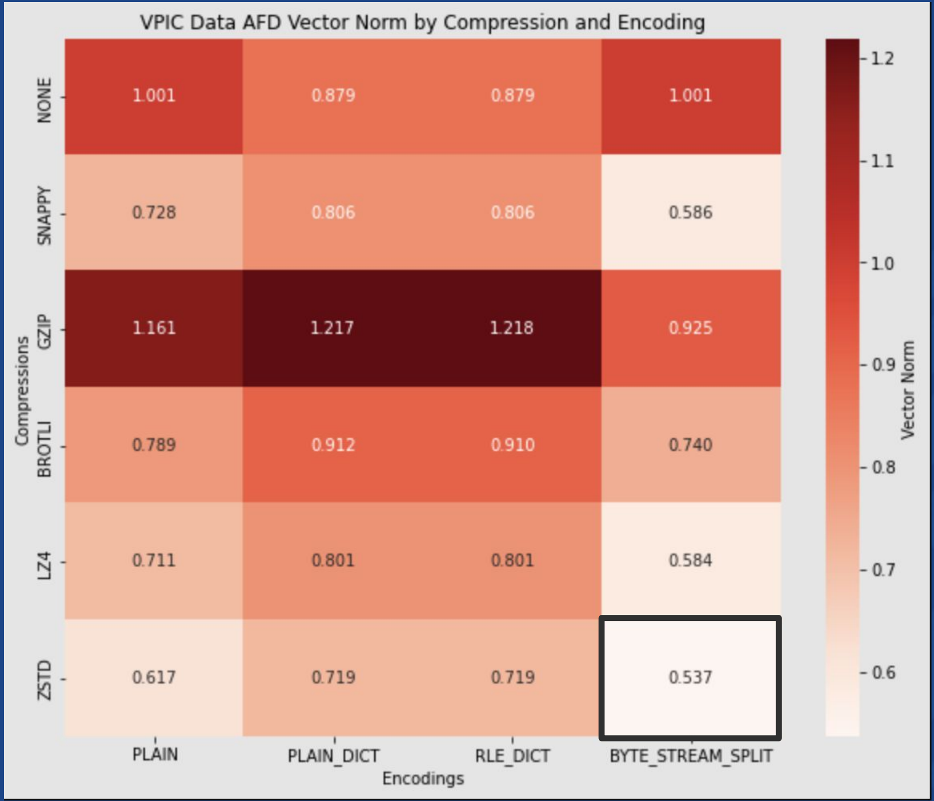
Benchmark Analysis With VPIC Data



	PLAIN	PLAIN_DICT	RLE_DICT	BYTE_STREAM_SPLIT
NONE	1.001	0.492	0.492	1.001
SNAPPY	0.743	0.483	0.484	0.624
GZIP	1.175	0.593	0.593	1.041
BROTLI	1.169	0.498	0.496	0.791
LZ4	0.728	0.484	0.484	0.624
ZSTD	0.635	0.473	0.474	0.562

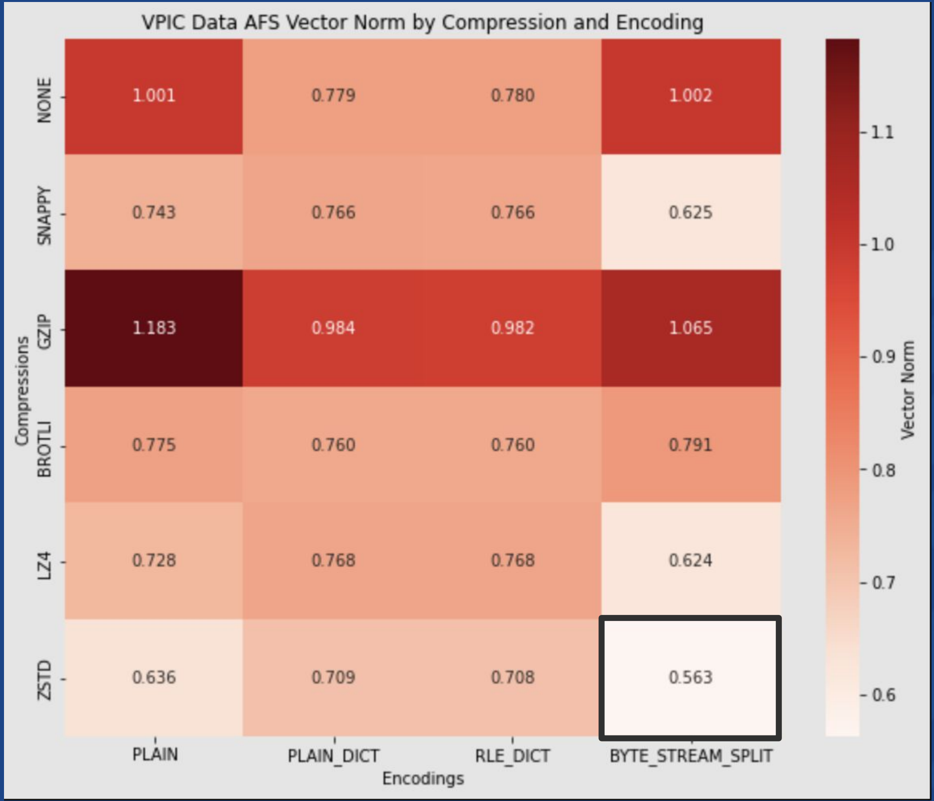
All Files Duplicated Results

Benchmark Analysis With VPIC Data



All Files Sampled Results

Benchmark Analysis With VPIC Data



VPIC Findings

Parquet Benchmark Analysis

	Lowest Vector Norm	
Input Format	Compression	Encoding
SFD	ZSTD	Byte Stream Split
SFS	ZSTD	Plain Dictionary
AFD	ZSTD	Byte Stream Split
AFS	ZSTD	Byte Stream Split

Laghos Data

LANL OCS Laghos Sample Dataset

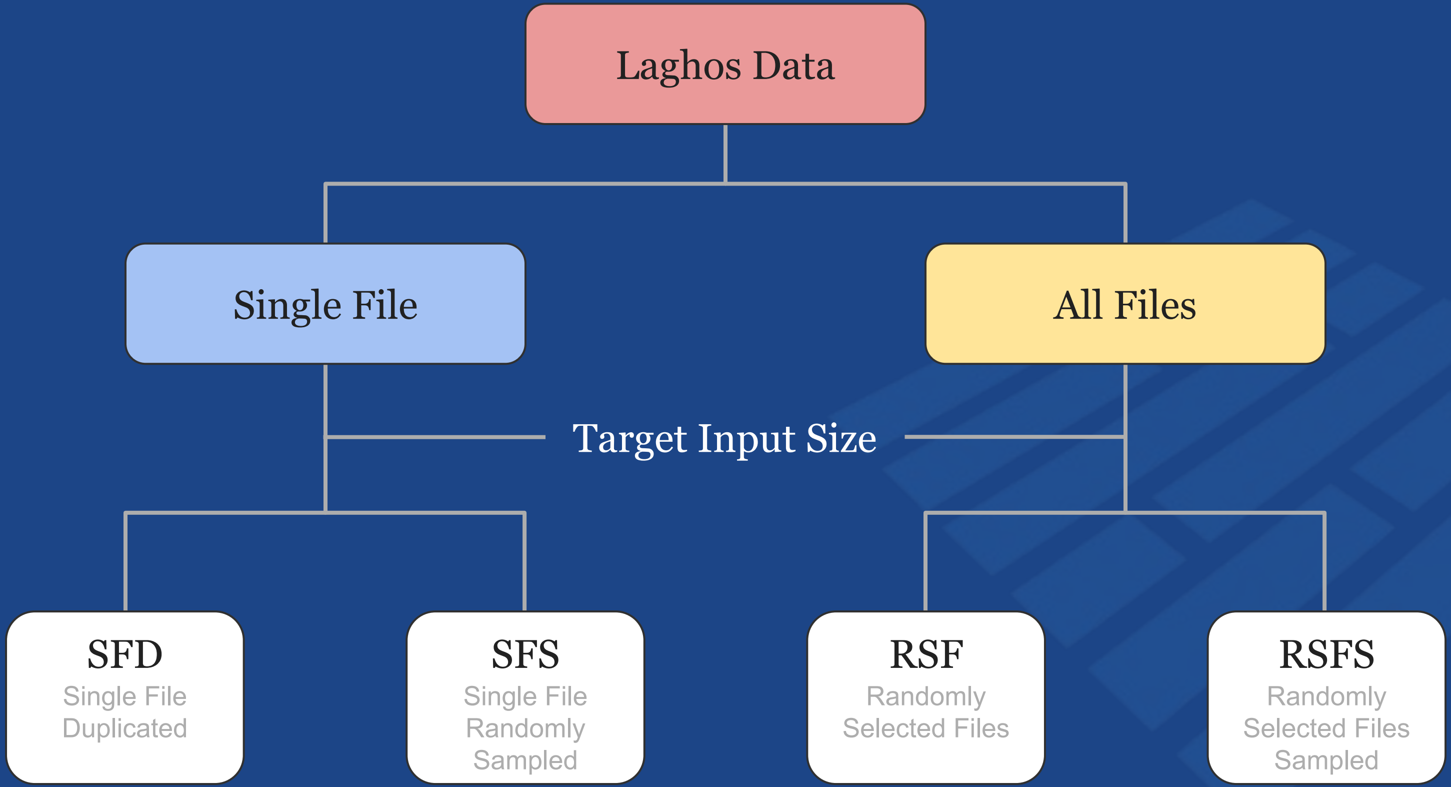
Nodal Schema

4 bytes	int32	**element_id**	unique ID of an element
4 bytes	int32	**vertex_id**	unique ID of a vertex
8 bytes	double	**v_x**	velocity of node in X direction
8 bytes	double	**v_y**	velocity of node in Y direction
8 bytes	double	**v_z**	velocity of node in Z direction
8 bytes	double	**rho**	density of node
8 bytes	double	**e**	energy of node
8 bytes	double	**x**	location of node in X direction
8 bytes	double	**y**	location of node in Y direction
8 bytes	double	**z**	location of node in Z direction

- 72 bytes per element
- 2048^2 elements per file
- 302 MiB per file
- 256 total files
- 75.5 GiB total size

Laghos Input Data Format

Benchmark Set Up



Laghos Data

Single File

All Files

Target Input Size

SFD
Single File Duplicated

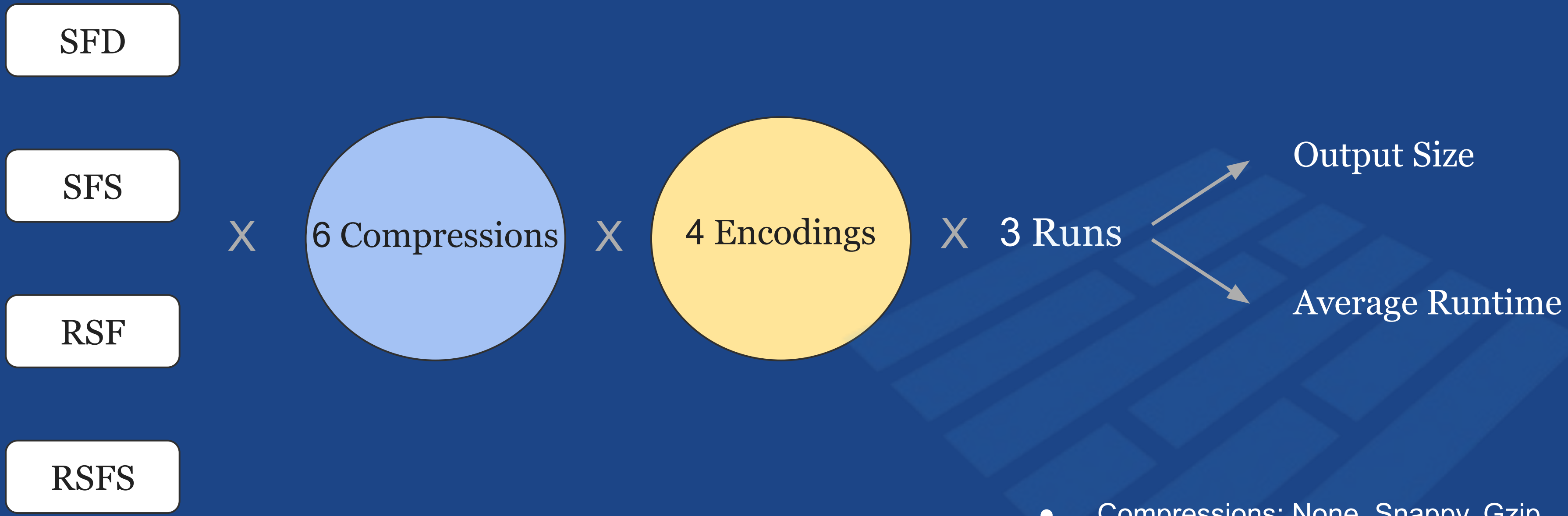
SFS
Single File Randomly Sampled

RSF
Randomly Selected Files

RSFS
Randomly Selected Files Sampled

Laghos Benchmark Framework

Writing Input Data Into Parquet



SFD

SFS

RSF

RSFS

X 6 Compressions X 4 Encodings X 3 Runs

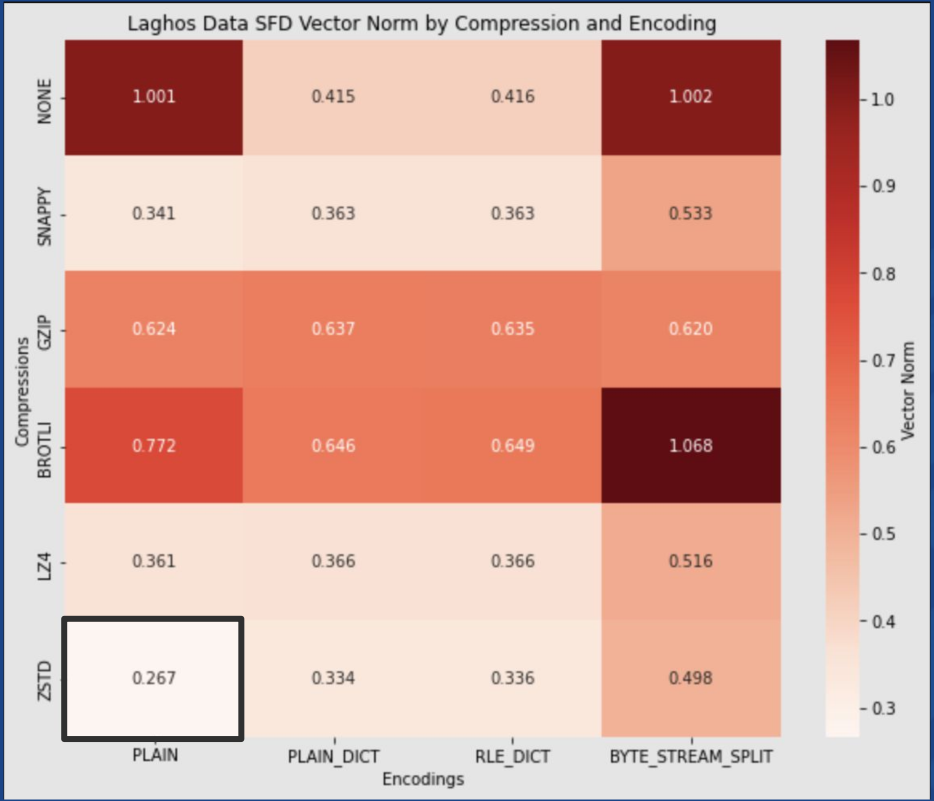
Output Size

Average Runtime

- Compressions: None, Snappy, Gzip, Brotli, LZ4, ZSTD
- Encodings: Plain, Plain Dictionary, RLE Dictionary, Byte Stream Split

Single File Duplicated Results

Benchmark Analysis With Laghos Data

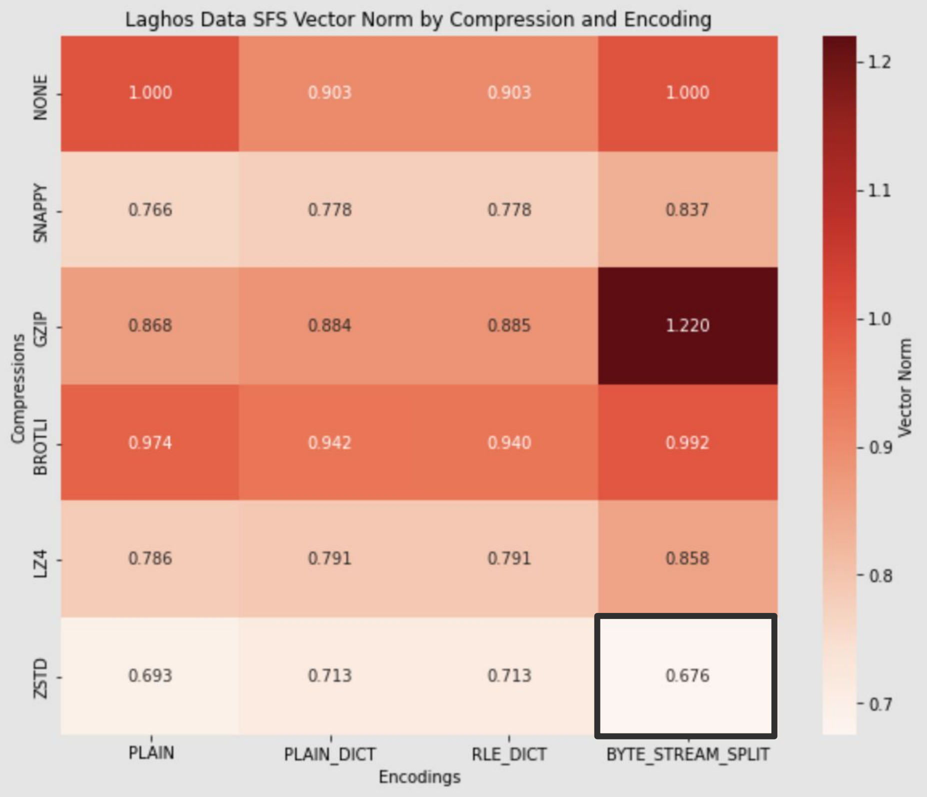


Laghos Data SFD Vector Norm by Compression and Encoding

Compressions	PLAIN	PLAIN_DICT	RLE_DICT	BYTE_STREAM_SPLIT
NONE	1.001	0.415	0.416	1.002
SNAPPY	0.341	0.363	0.363	0.533
GZIP	0.624	0.637	0.635	0.620
BROTLI	0.772	0.646	0.649	1.068
LZ4	0.361	0.366	0.366	0.516
ZSTD	0.267	0.334	0.336	0.498

Single File Sampled Results

Benchmark Analysis With Laghos Data

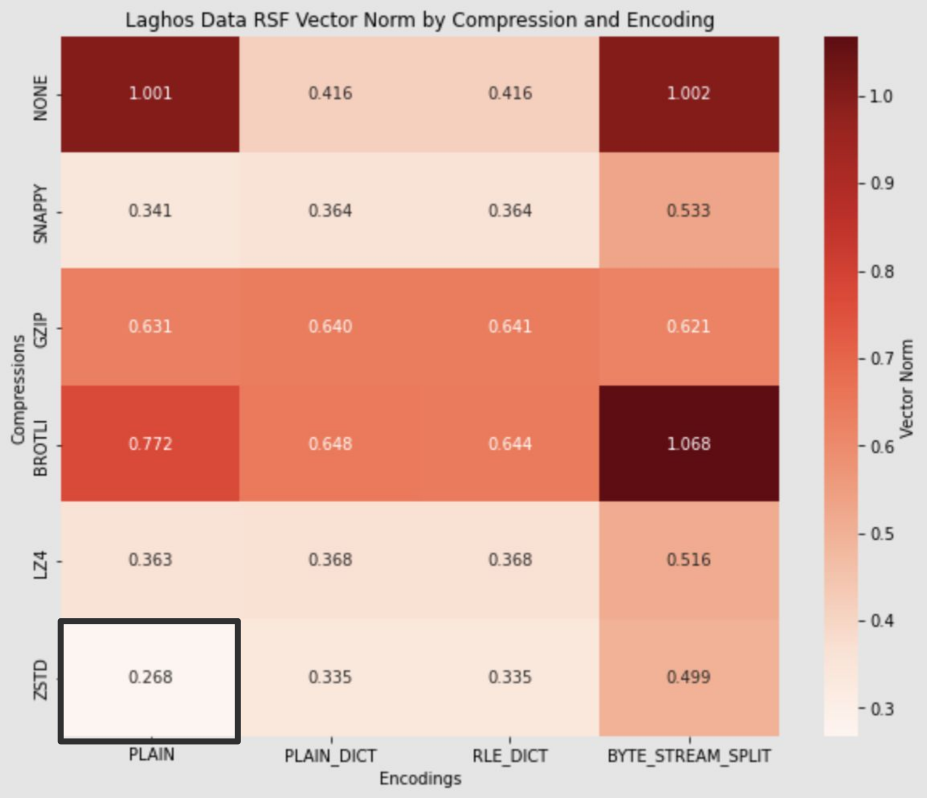


Laghos Data SFS Vector Norm by Compression and Encoding

Compressions	PLAIN	PLAIN_DICT	RLE_DICT	BYTE_STREAM_SPLIT
NONE	1.000	0.903	0.903	1.000
SNAPPY	0.766	0.778	0.778	0.837
GZIP	0.868	0.884	0.885	1.220
BROTLI	0.974	0.942	0.940	0.992
LZ4	0.786	0.791	0.791	0.858
ZSTD	0.693	0.713	0.713	0.676

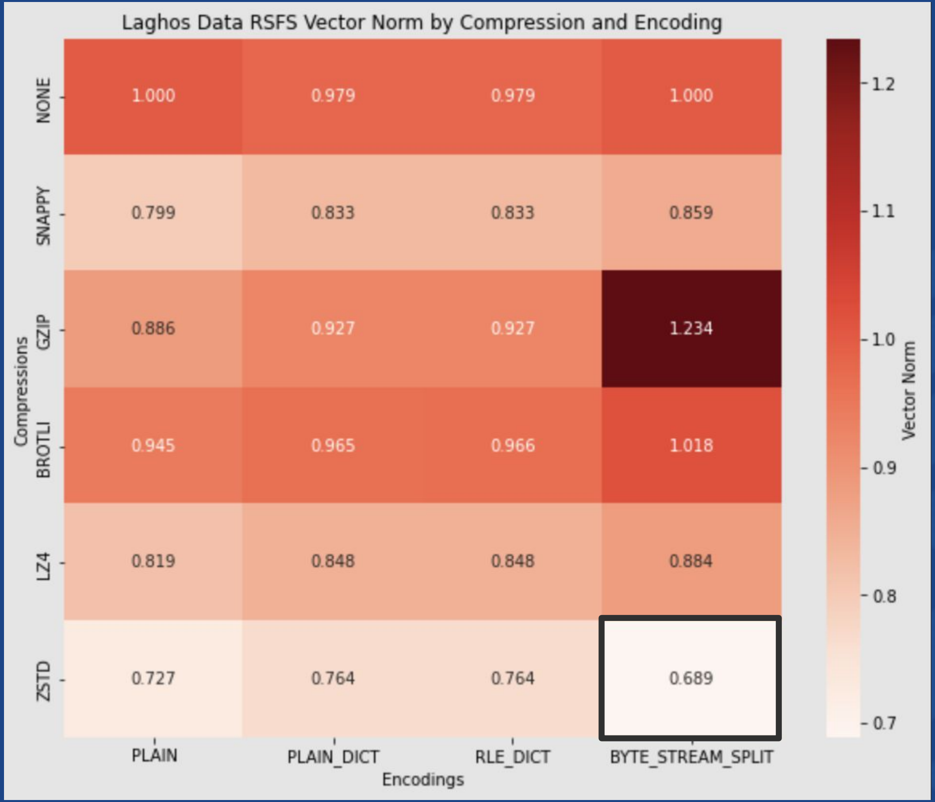
Randomly Selected Files Results

Benchmark Analysis With Laghos Data



Randomly Selected Files Sampled Results

Benchmark Analysis With Laghos Data



	PLAIN	PLAIN_DICT	RLE_DICT	BYTE_STREAM_SPLIT
NONE	1.000	0.979	0.979	1.000
SNAPPY	0.799	0.833	0.833	0.859
GZIP	0.886	0.927	0.927	1.234
BROTLI	0.945	0.965	0.966	1.018
LZ4	0.819	0.848	0.848	0.884
ZSTD	0.727	0.764	0.764	0.689

Laghos Findings

Parquet Benchmark Analysis

	Lowest Vector Norm	
Input Format	Compression	Encoding
SFD	ZSTD	Plain
SFS	ZSTD	Byte Stream Split
RSF	ZSTD	Plain
RSFS	ZSTD	Byte Stream Split

Final Findings

Parquet Benchmark Analysis

	Lowest Vector Norm (Laghos)		Lowest Vector Norm (VPIC)	
Input Format	Compression	Encoding	Compression	Encoding
SFD	ZSTD	Plain	ZSTD	Byte Stream Split
SFS	ZSTD	Byte Stream Split	ZSTD	Plain Dictionary
RSF	ZSTD	Plain		
RSFS	ZSTD	Byte Stream Split		
AFD			ZSTD	Byte Stream Split
AFS			ZSTD	Byte Stream Split

Future Works

Parquet Benchmark Analysis

- Investigate Parquet’s fallback mechanism from dictionary to plain encoding when the dictionary grows too large.
  - Analyze the performance and storage impact of switching encodings based on dictionary size or distinct value count.
  - Explore optimization opportunities by isolating conditions that trigger the fallback, aiming to enhance encoding strategies for diverse datasets.