Youmu: Efficient Columnar Data Pipeline for LLM Training

Tianle Zhong [1] Jiechen Zhao [2] Qiang Su [3] Geoffrey Fox [1]

[1]University of Virginia [2]University of Toronto [3]The Chinese University of Hong Kong

What Format Is Your LLM Dataset?

Large language model (LLM) training datasets are web-scale.

- PiB-level raw data; TiB-level cleaned data.
- Massive data preprocessing.
- Needs to reside on costly high performance storage.

Many training frameworks and datasets are using **JSONL**, but we know that it is not an ideal choice for managing such amount of data.

Data Preparation in Parquet; Why Not Training?

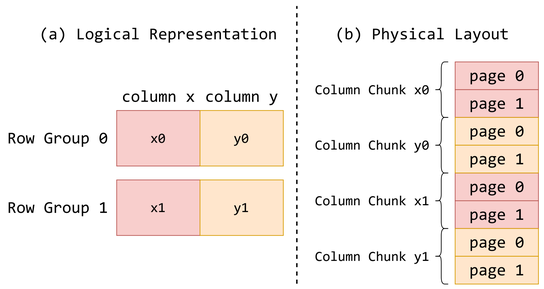


Figure 1. Parquet logical and physical layouts.

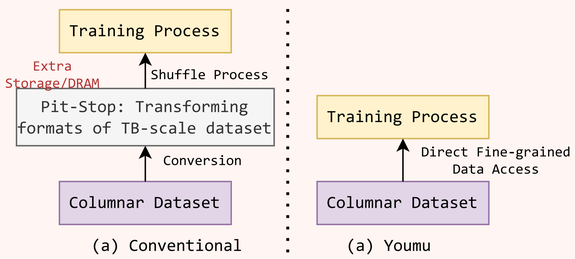
A lot of data cleaning & preparation tools use Parquet, which is designed for efficiently processing *web-scale datasets*.

With Parquet, data preparation reuses existing data infrastructure for analytical workload.

- Designed for scan-based operations.
- Optimized for block storage.
- High compression rate to reduce storage cost.

However, Parquet is inefficient with random access, a key procedure for data shuffling to ensure model accuracy.

Costly Pit-stops between Preparation and Training



The current practice is to convert & pre-shuffle columnar data into other formats like `JSONL`.

- 3× more storage capacity needed on costly performant storage.
- Break single truth of data.
- Additional human efforts to keep up with ever evolving datasets with new data and preprocessing refinement.

Issues with Existing Solutions and Our Goals

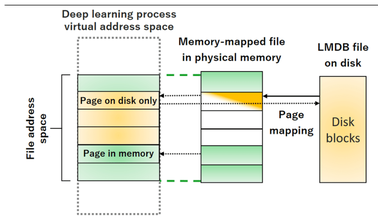


Figure 2. Mmap file I/O

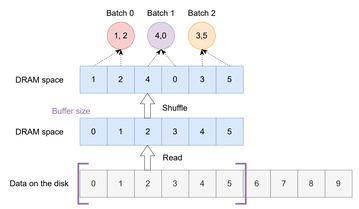


Figure 3. Streaming shuffle.

- Distributed memory like Ray and Spark
  - High *memory footprint and complexity*.
  - Memory contention with other key functionalities like model checkpointing.
- Disk-backed memory mapping (HuggingFace Datasets).
  - Unsatisfactory *throughput* due to thrashing (page faults).
- Streaming I/O-based local shuffle.
  - Obvious model accuracy loss due to limited *shuffle quality*.

Our Goals

We aim to stay with one consistent columnar format for training data pipeline, and achieves

- Controlled DRAM footprint (against distributed memory).
  - provide memory resilience for other critical functionalities like tensor offloading and model checkpointing.
- Sufficient throughput (against MMap I/O) to avoid data starvation for GPU utilization.
- High shuffle quality (against local shuffle) to ensure model accuracy.

Our Observations

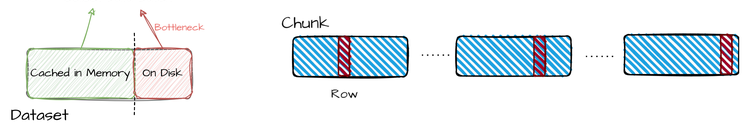


Figure 4. Disk I/O remains as bottleneck

Figure 5. A lot of I/O bandwidth is wasted due to the granularity gap.

- Marginal benefits of caching unless the dataset fully cached, considering that access to the same row is exactly once per epoch.
- Significant granularity gap: Fine-grained shuffling vs. Coarse-grained columnar chunk I/O
  - Consume rows but have to read chunks, resulting in extremely low effective bandwidth.
  - In other words, a lot of data is fed into CPU but never used by GPU.

Youmu Overview

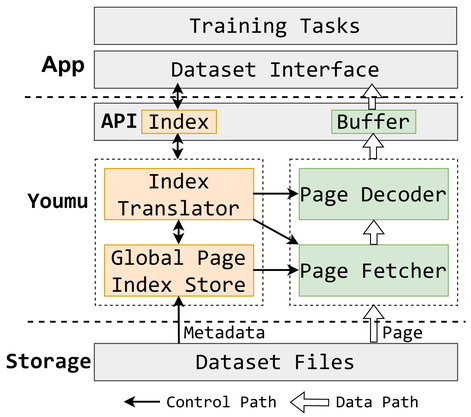


Figure 6. Youmu system overview.

Design highlights:

- No cache, only buffer.
  - From Observation 1.
- Fine-grained page-level I/O.
  - From Observation 2.
- Practical compatibility.
  - Directly works with widely adopted Parquet.

Youmu Design

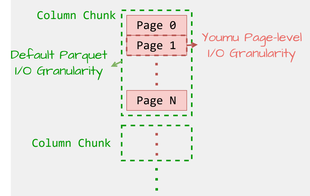


Figure 7. Page-level I/O

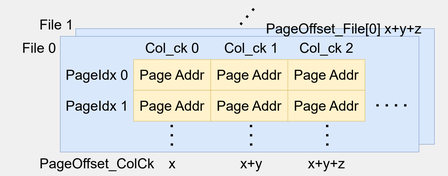


Figure 8. A lot of I/O bandwidth is wasted due to the granularity gap.

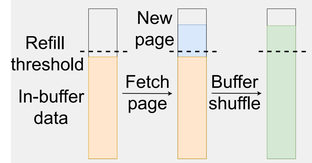


Figure 9. Aggressive buffer shuffle.

Aggressive Buffer Shuffle

- Based on the fact: page size < buffer size.
- Shuffle rows in buffer upon the arrival of each new page.
- Better shuffle quality than standard buffer shuffle at initialization.

Page-level I/O Granularity

- Friendly I/O size for SSD.
- Many pages to shuffle.
- Improved I/O goodput.

Global Page Index

- A 3-dim index across file, column chunk and page.
- Lightweight initialization since only metadata needed.
- Layered binary search-based index translation.

Implementation

- Rust runtime: Based on official Rust implementation of Apache Parquet and Arrow.
- Python APIs to be directly integrated with PyTorch Dataset interface.
  - Zero-copy conversion to various dataframe and tensor formats enabled by Arrow memory model ecosystem.
- Support full shuffle by extracting a random row from retrieved page.

Evaluation

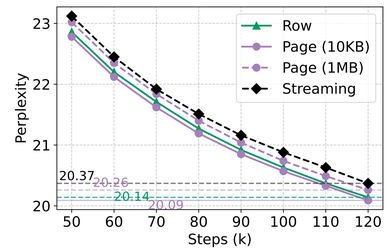


Figure 10. Perplexity against training steps

Key Take-away

- A small page (10KB) achieves competitive quality to row shuffling.
- A big page (1MB) still outperforms streaming shuffle.

TL;DR

Youmu can achieve sufficiently low latency to saturate GPU while keeping very low memory overheads

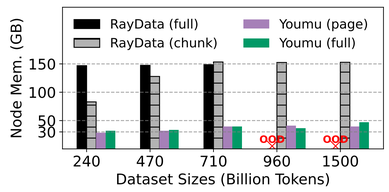


Figure 11. Memory footprint per node against dataset sizes on 16 nodes.

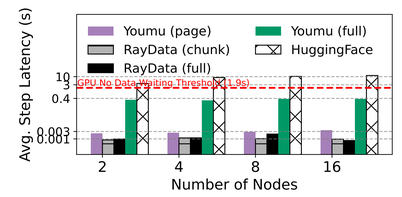


Figure 12. Iteration batch latency against a 200B dataset.