Scaling Data Tokenization for AI Systems

Rajni Pawar Luke Logan Jamie Cernuda Garcia Xian-He Sun
Anthony Kougkas
rpawar4@hawk.iit.edu, llogan@hawk.iit.edu,
jcernudagarcia@hawk.iit.edu, sun@iit.edu, akougkas@iit.edu

Gnosis Research Center

StoreHub

Introduction

1. AI is becoming the primary user of raw data – either for training or inference (e.g., RAG)

2. Data for AI models are tokenized before they can be used.

3. Tokenization is an expensive process requiring reading and parsing large datasets.

4. This data-intensive operation causes performance bottlenecks, especially for massive datasets that don't fit in memory.

5. Current I/O libraries are not optimized for AI workloads.

Proposed Solution

Labios, an active storage system that will:

1. Apply operations to data while it is being transferred.

2. Transparently tokenize data during I/O operations.

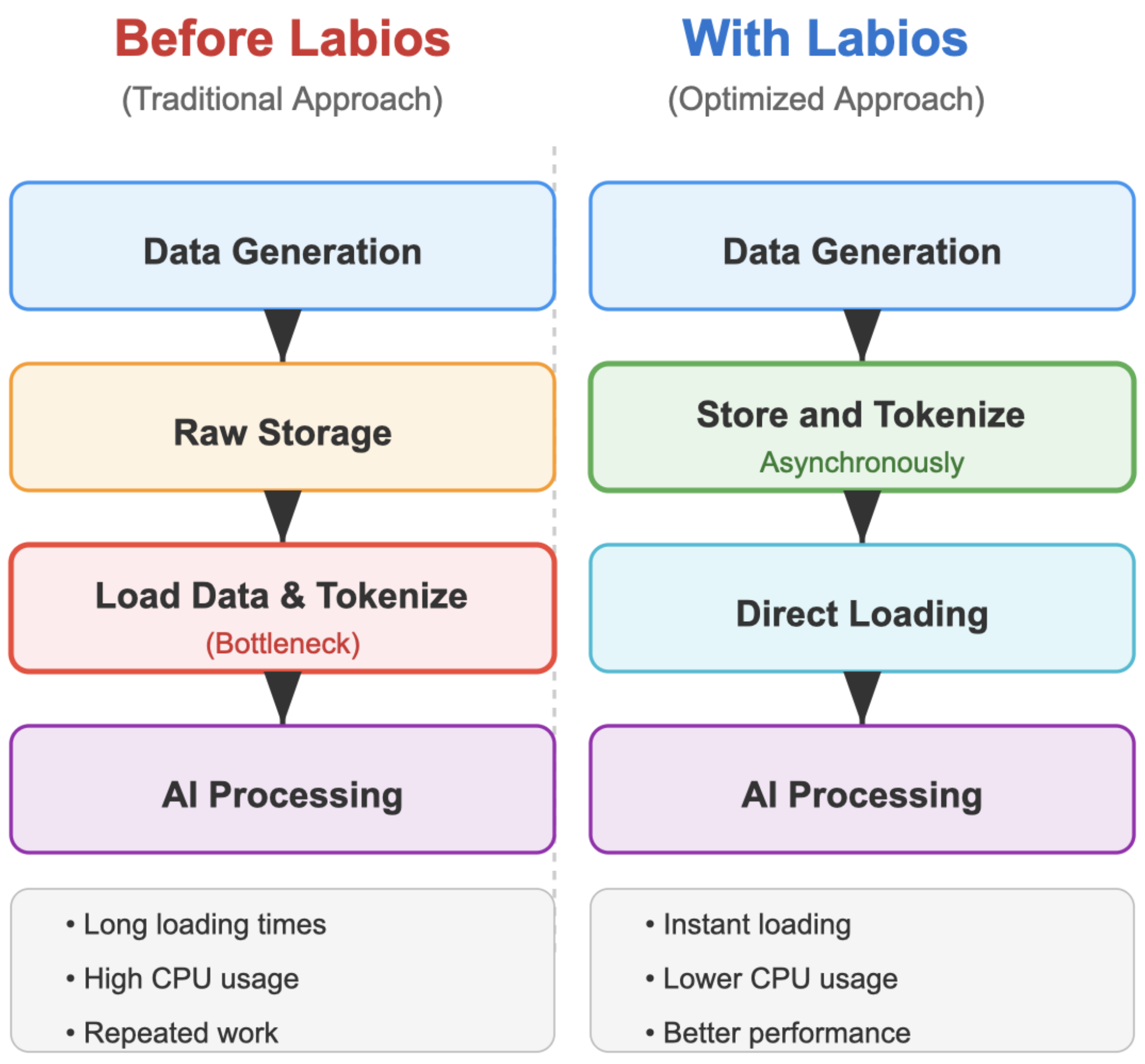
3. Serve as an AI-ready storage system with custom tokenization operators.

4. Built on the IoWarp framework for optimized I/O operations.

5. Eliminate redundant tokenization during model training and inference.

Making I/O Stacks Ready For AI

- **Traditional Approach:** Tokenization performed on every data load. CPU-intensive at runtime. Longer wait times for AI model responses.
- **Optimised Approach:** Tokenization performed only once. No processing overhead at runtime. Immediate data availability for AI Models



Before Labios
(Traditional Approach)

Data Generation

Raw Storage

Load Data & Tokenize
(Bottleneck)

AI Processing

- Long loading times
- High CPU usage
- Repeated work

With Labios
(Optimized Approach)

Data Generation

Store and Tokenize
Asynchronously

Direct Loading

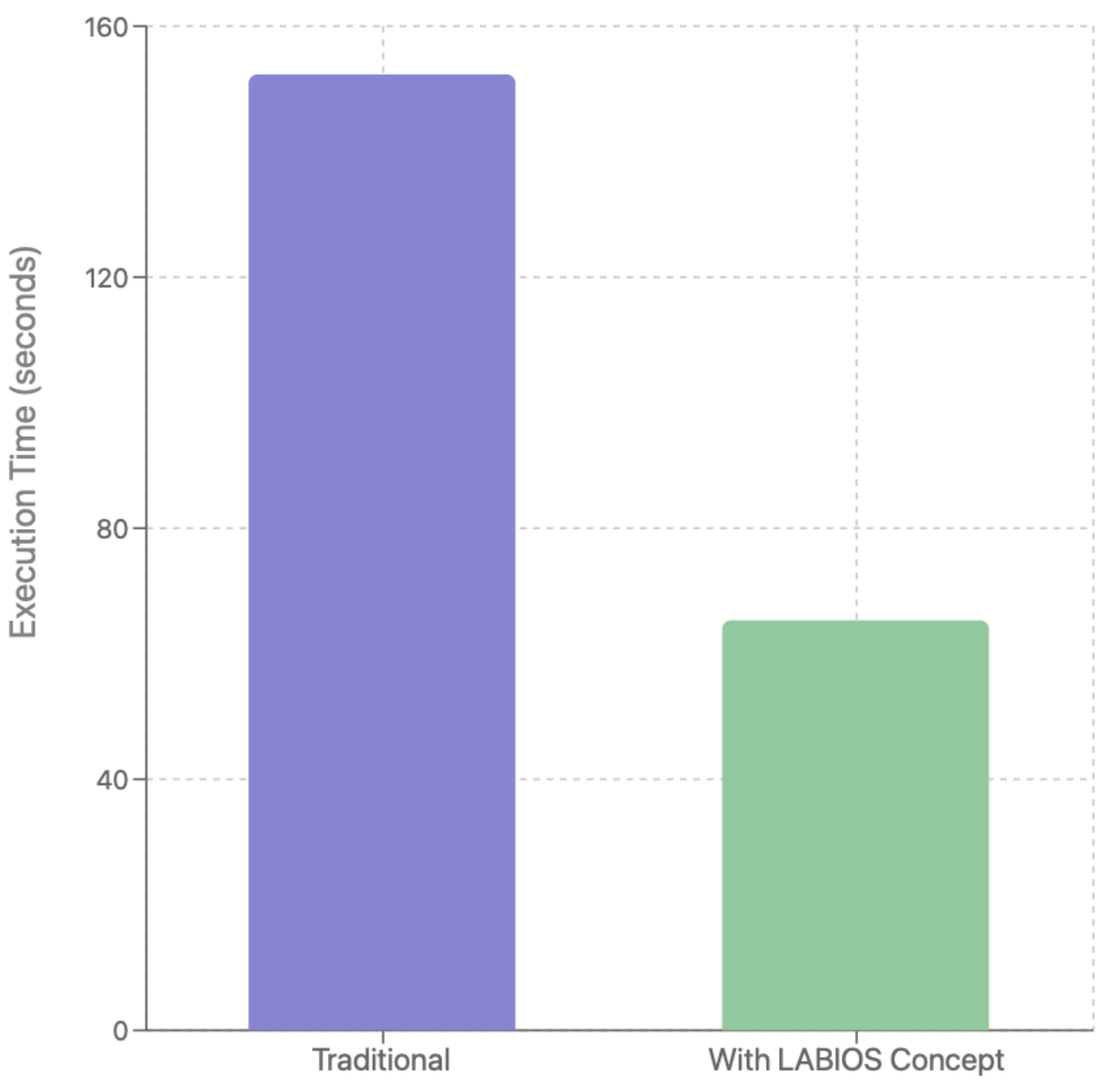
AI Processing

- Instant loading
- Lower CPU usage
- Better performance

Results

Cost of running LangChain on 1GB data with synchronous tokenization vs Labios

- Synchronous tokenization before inference is **152.31 seconds**
- Labios asynchronously tokenizes and stores data before-hand.
- Labios reduces runtime to **65.33 seconds**

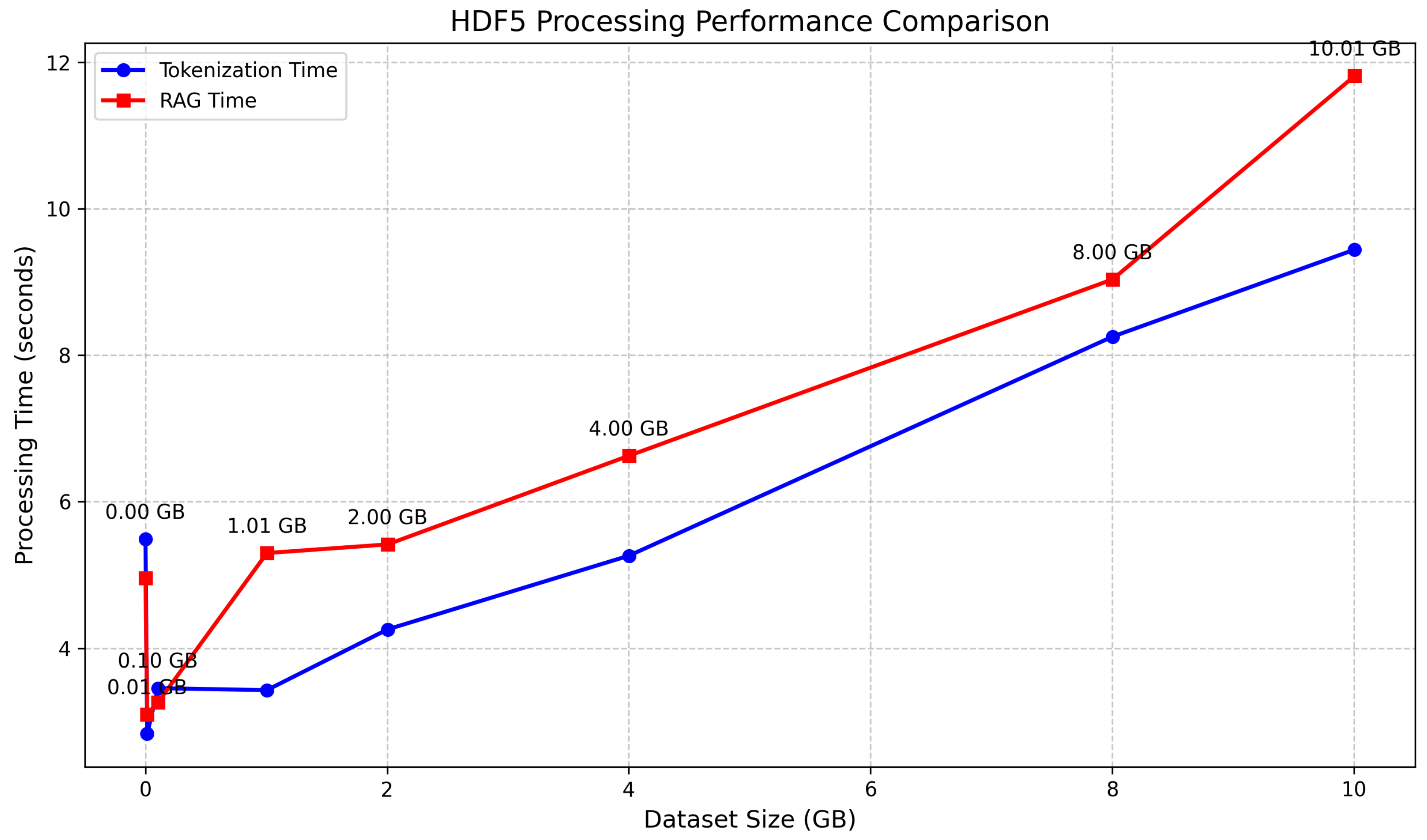


The Cost of Live Tokenization

Standard I/O patterns involve multiple copies of data across pipeline. As AI datasets grow in size, tokenization becomes an increasing bottleneck.

Tokenization time increases linearly with dataset sizes. Hence, RAG processing time increases significantly with larger datasets.

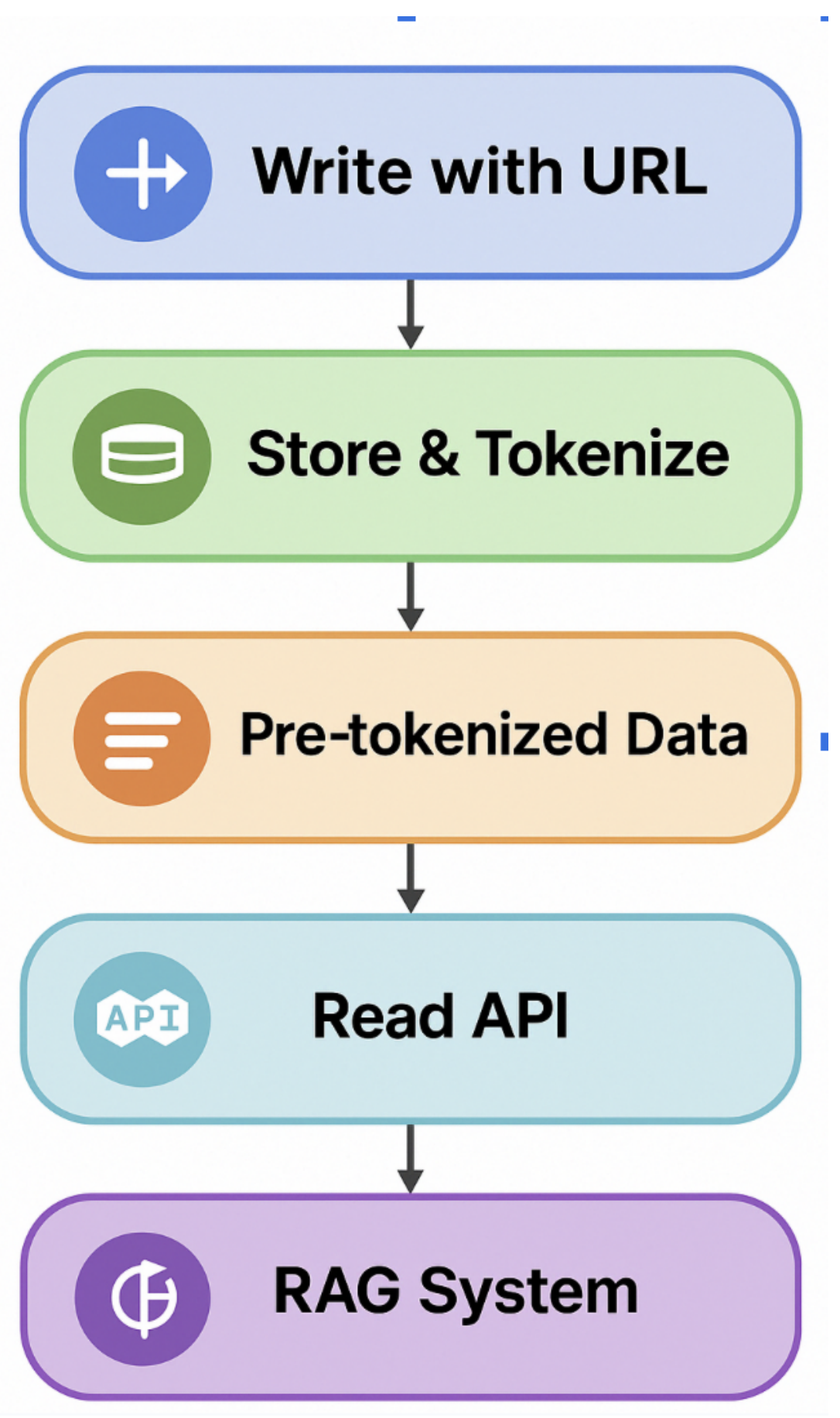
Preprocessing tokenization during write operations can greatly reduce inference and training times.



Labios Workflow

```
//Write Operation
// C++ example using Labios API
labios::File file;
file.write("/path/to/sample.txt?sentencepiece",
 buffer, size);
// Asynchronously tokenizes with SentencePiece
```

```
// Read Operation
# Python example using Labios API
from labios import Labios
lb = Labios()
tokens = lb.labio_read("/path/to/sample.txt")
```



Write with URL

Store & Tokenize

Pre-tokenized Data

Read API

RAG System

Conclusion

Benefits with LABIOS:

1. Reduced latency for model training/inference by eliminating tokenization, resulting in **performance improvement by 57.1%.**

2. Lower resources required during training/inference pipelines.

3. Improved throughput for RAG-based applications.

4. Ability to handle datasets larger than available memory.

5. Parallel tokenization during I/O operations

6. Reduced computational overhead on AI frameworks

Acknowledgments

This material is based upon work supported in part by the National Science Foundation (NSF), Division of Computer and Network Systems (CISE/CNS), under Grant CCRI-CISE 2346504, and the Department of Energy (DOE), Office of Advanced Scientific Computing Research (ASCR), under Grant DE-SC0024593.