

TensorDex

A Compact, Tensor-Centric Storage System for Modern AI Models

Tingfeng Lan, Zirui Wang, Yunjia Zheng, Zhaoyuan Su, Juncheng Yang, Yue Cheng

University of Virginia · Harvard University
SOSP 2026

Presenter: Jigao Luo

Background: Model Weights FLOAT32 Are Hard to Compress

Name	Model Type	N° of Params.	Gor.	Ch.	Ch. 128	Patas	ALP _{rd}	Zstd
Dino-Vitb16 [11]	Vision Transformer	86,389,248	34.1	33.4	33.4	45.8	28.3	29.7
GPT2 [42]	Text Generation	124,439,808	34.1	33.5	33.5	45.6	27.7	29.7
Grammarly-lg [43]	Text2Text	783,092,736	34.1	33.4	33.4	45.5	27.7	29.6
W2V Tweets	Word2Vec	3,000	34.1	33.3	33.3	45.5	28.8	29.8
AVG.			34.1	33.4	33.4	45.6	28.1	29.7

**Bits per weight of lossless compressors on FP32 model weights
(original: 32 bits; lower is better).**

Azim Afroozeh, Leonardo Kuffó, Peter Boncz.
ALP: Adaptive Lossless Floating-Point Compression. SIGMOD 2024. Table 7, p. 12.

Background: Model Weights FLOAT32 Are Hard to Compress

Name	Model Type	N° of Params.	Gor.	Ch.	Ch. 128	Patas	ALP _{rd}	Zstd
Dino-Vitb16 [11]	Vision Transformer	86,389,248	34.1	33.4	33.4	45.8	28.3	29.7
GPT2 [42]	Text Generation	124,439,808	34.1	33.5	33.5	45.6	27.7	29.7
Grammarly-lg [43]	Text2Text	783,092,736	34.1	33.4	33.4	45.5	27.7	29.6
W2V Tweets	Word2Vec	3,000	34.1	33.3	33.3	45.5	28.8	29.8
		AVG.	34.1	33.4	33.4	45.6	28.1	29.7

Bits per weight of lossless compressors on FP32 model weights
(original: 32 bits; lower is better).

Background: Model Weights FLOAT32 Are Hard to Compress

Name	Model Type	N° of Params.	Gor.	Ch.	Ch. 128	Patas	ALP _{rd}	Zstd
Dino-Vitb16 [11]	Vision Transformer	86,389,248	34.1	33.4	33.4	45.8	28.3	29.7
GPT2 [42]	Text Generation	124,439,808	34.1	33.5	33.5	45.6	27.7	29.7
Grammarly-lg [43]	Text2Text	783,092,736	34.1	33.4	33.4	45.5	27.7	29.6
W2V Tweets	Word2Vec	3,000	34.1	33.3	33.3	45.5	28.8	29.8
		AVG.	34.1	33.4	33.4	45.6	28.1	29.7

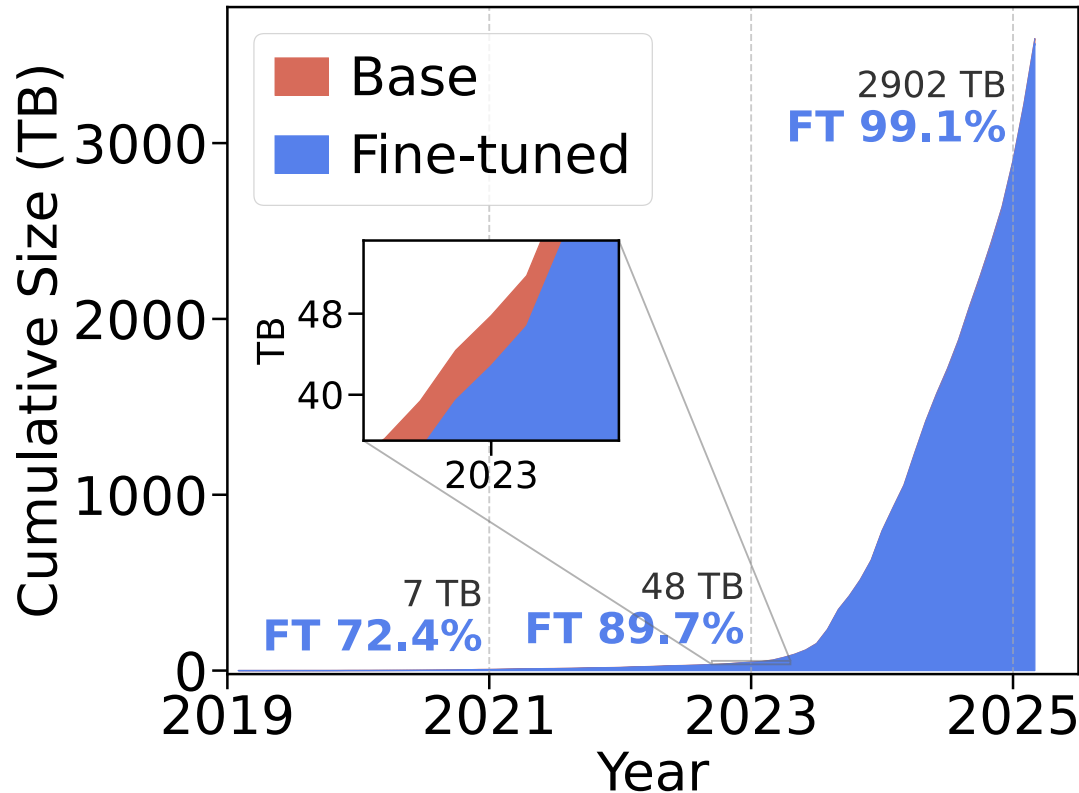
Bits per weight of lossless compressors on FP32 model weights
(original: 32 bits; lower is better).

32 → ~28–30
Even the best method saves only ≈12% on average.

Azim Afroozeh, Leonardo Kuffó, Peter Boncz.

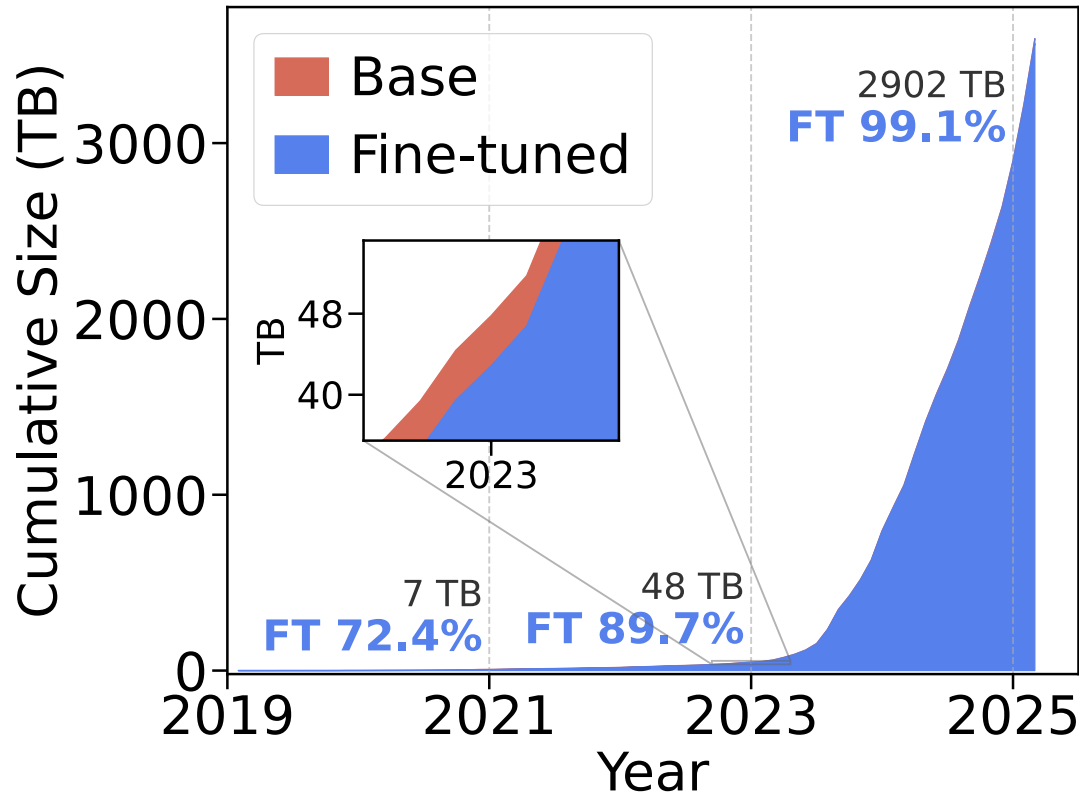
ALP: Adaptive Lossless Floating-Point Compression. SIGMOD 2024. Table 7, p. 12.

Background: Fine-Tuned Models take $\approx 99\%$ Hub Storage



**Cumulative storage size on Hugging Face, 2019–2025.
Fine-tuned models (blue) dominate the storage.**

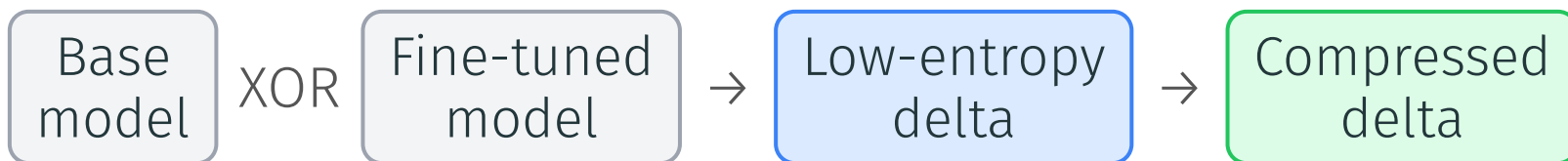
Background: Fine-Tuned Models take $\approx 99\%$ Hub Storage



Substantial cross-model redundancy in fine-tuned derivatives.

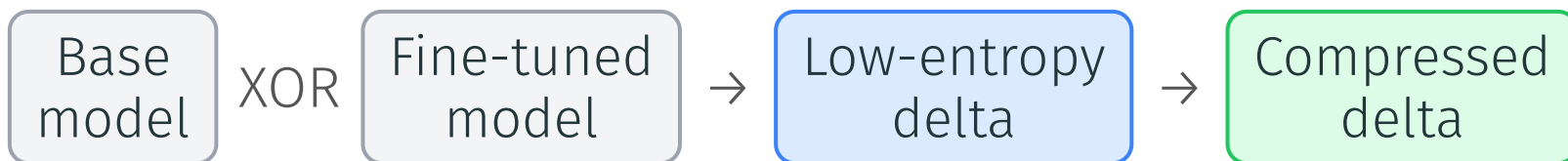
Delta Compression for Lossless

Store only the difference to a **similar base**



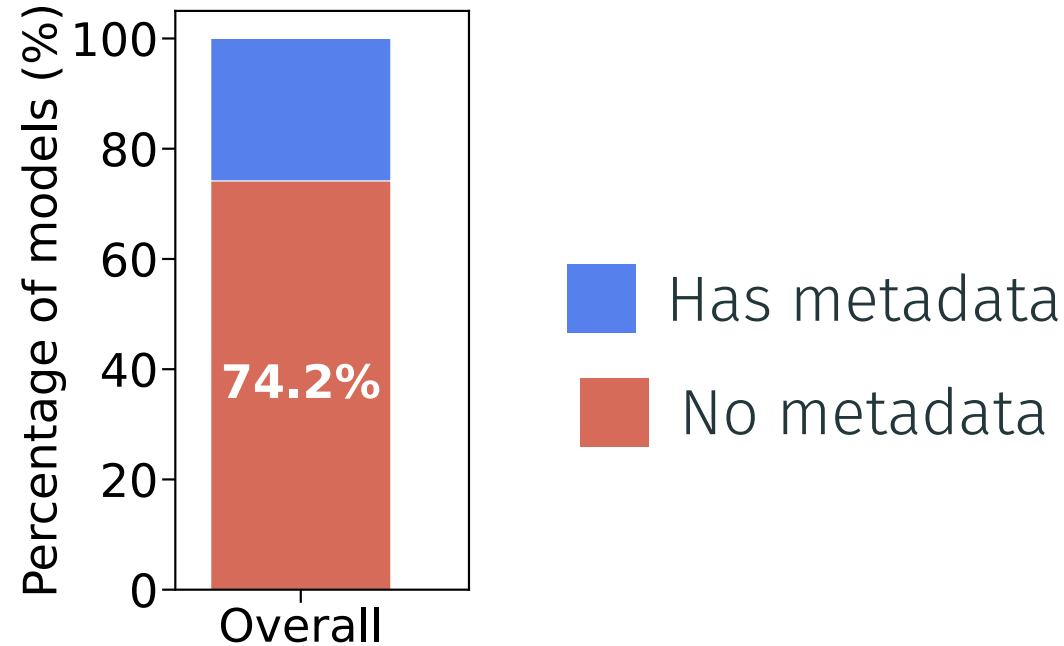
Delta Compression for Lossless

Store only the difference to a **similar base**



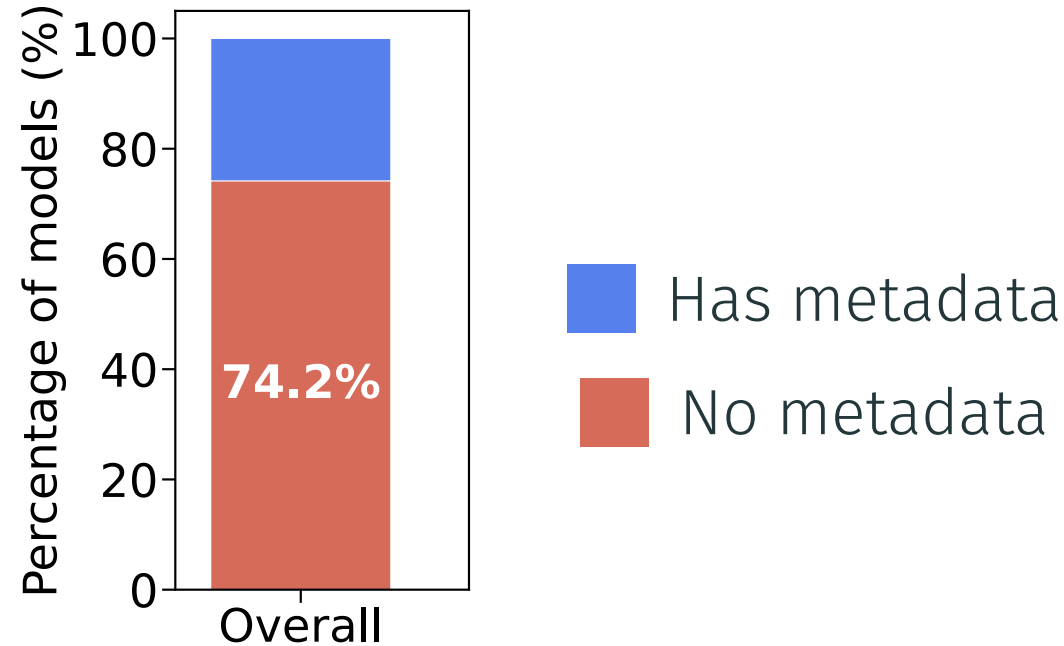
Still nontrivial with three challenges & observations

Observation 1: Missing Model Lineage Metadata



Share of 2.75M Hugging Face models with and without lineage metadata.

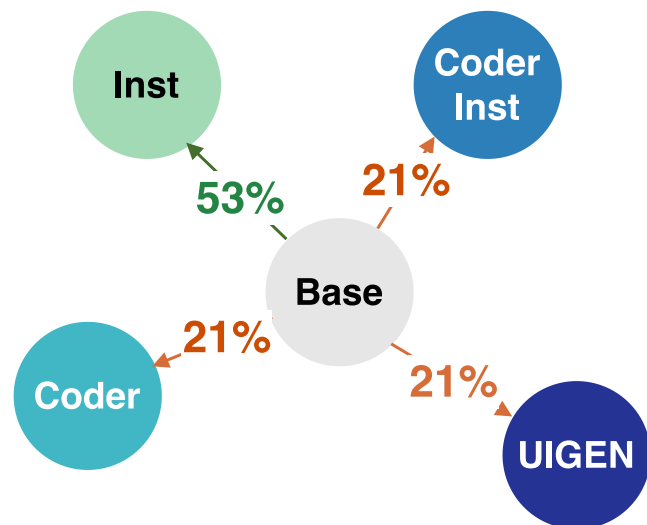
Observation 1: Missing Model Lineage Metadata



**Lineage metadata is often missing or unreliable.
We cannot rely on it to pick delta bases.**

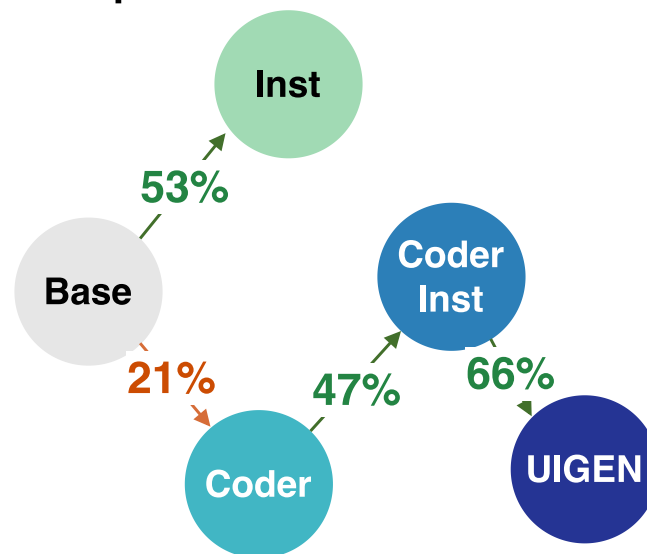
Observation 2: Model Pair Sensitivity

Fixed-Base Compression (ZipLLM)



Overall Reduction Ratio: 29%

Empirical Best-Base Selection

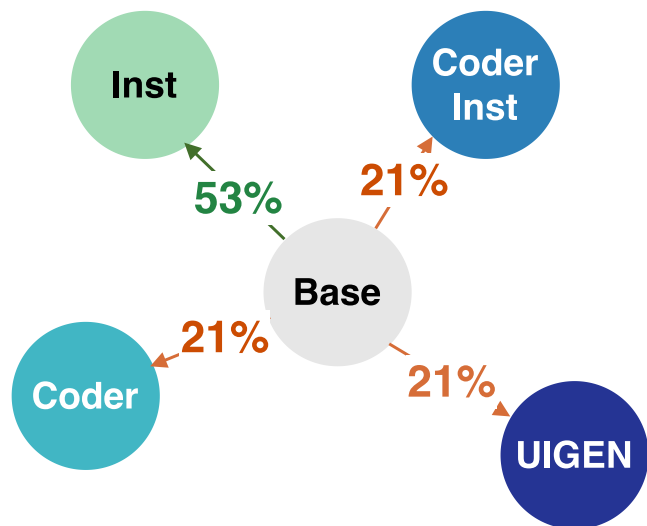


Overall Reduction Ratio: 47%

Same five Qwen2.5-3B models: always using the pretrained base (left) vs. picking the best base for each model (right).

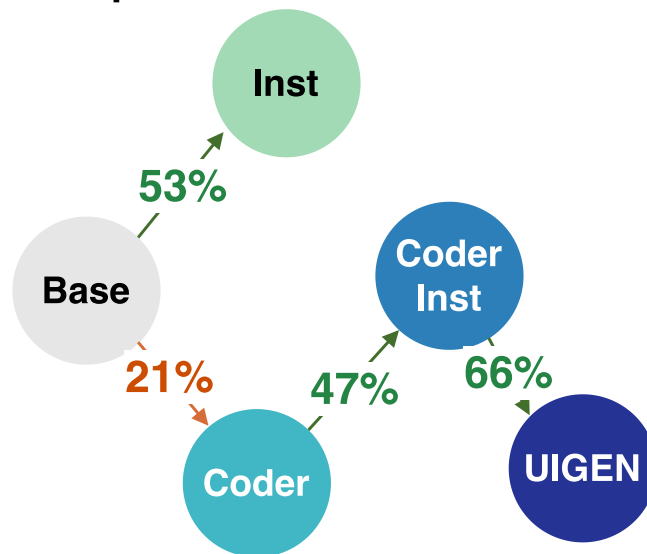
Observation 2: Model Pair Sensitivity

Fixed-Base Compression (ZipLLM)



Overall Reduction Ratio: 29%

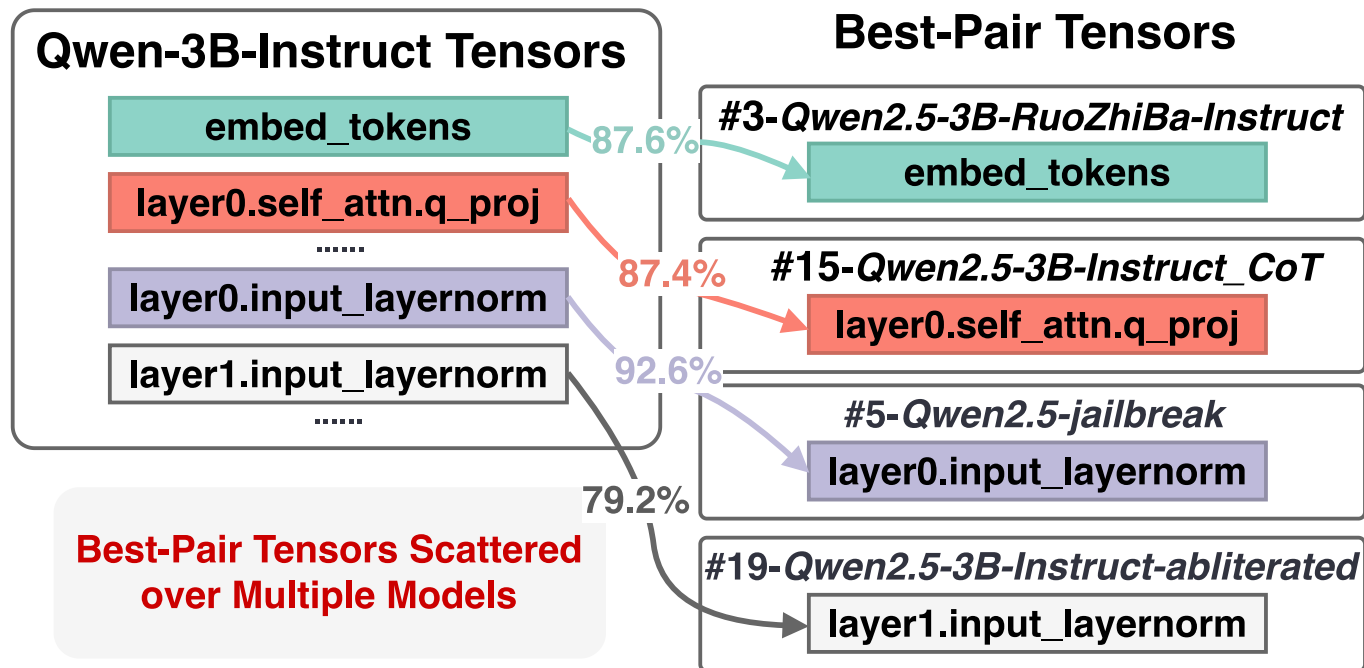
Empirical Best-Base Selection



Overall Reduction Ratio: 47%

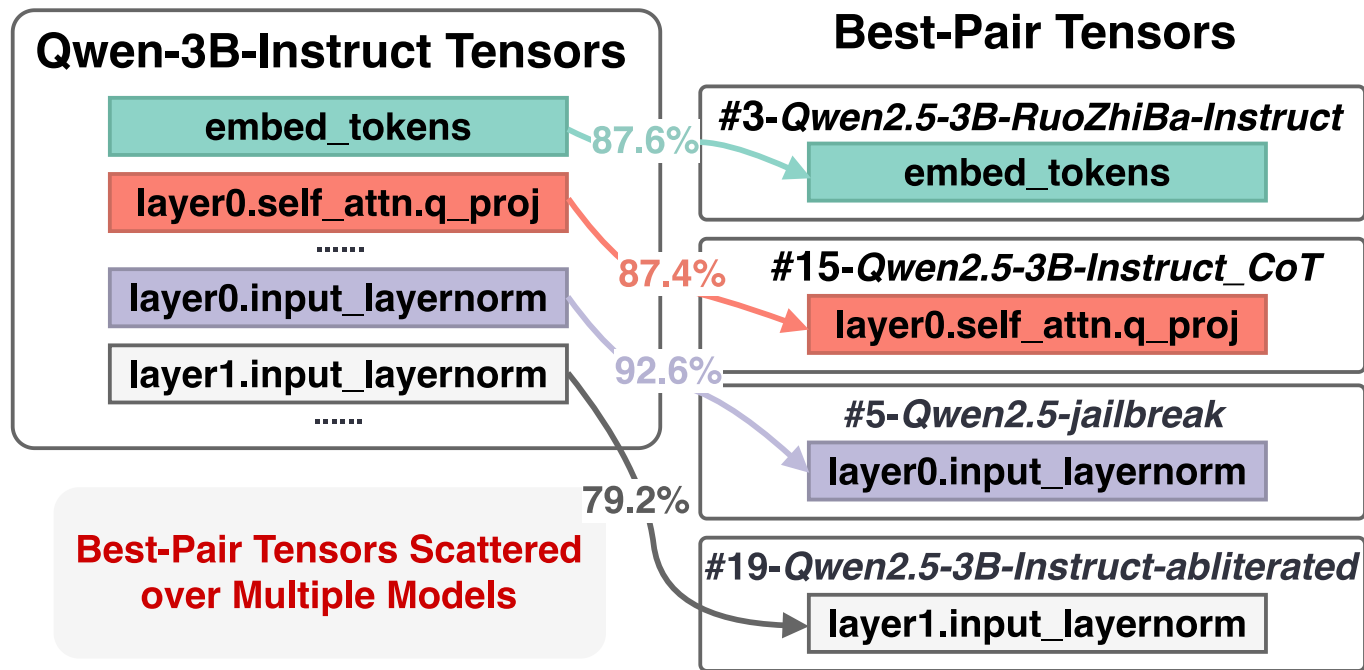
**A pretrained base is not enough.
Pair models by true weight similarity.**

Observation 3: Compression Granularity



Each tensor of Qwen-3B-Instruct has its best base in a different model (percentages = size reduction).

Observation 3: Compression Granularity



**Delta compress at the tensor level, not the model level.
Per-tensor pairing improves data reduction.**

Problem Formulation

The diagram shows the equation
$$\min_{\text{base selection}} \left(\sum \text{cost}_{base} + \sum \text{cost}_{delta} \right).$$
 with several annotations. A green line labeled "pairing decisions" points to the "min base selection" box. A red line labeled "base storage cost" points to the cost_{base} term. A blue line labeled "comp. delta storage cost" points to the cost_{delta} term.

pairing decisions

$\min_{\text{base selection}} \left(\sum \text{cost}_{base} + \sum \text{cost}_{delta} \right).$

base storage cost

comp. delta storage cost

Jointly choose which tensors act as **bases**
and which are stored as **compressed deltas**.

Problem Formulation

The diagram shows the equation
$$\min_{\text{base selection}} \left(\sum \text{cost}_{\text{base}} + \sum \text{cost}_{\text{delta}} \right).$$
 with several annotations. A green line labeled "pairing decisions" points to the "min" part of the equation. A red line labeled "base storage cost" points to the $\text{cost}_{\text{base}}$ term. A blue line labeled "comp. delta storage cost" points to the $\text{cost}_{\text{delta}}$ term.

pairing decisions

$$\min_{\text{base selection}} \left(\sum \text{cost}_{\text{base}} + \sum \text{cost}_{\text{delta}} \right).$$

base storage cost

comp. delta storage cost

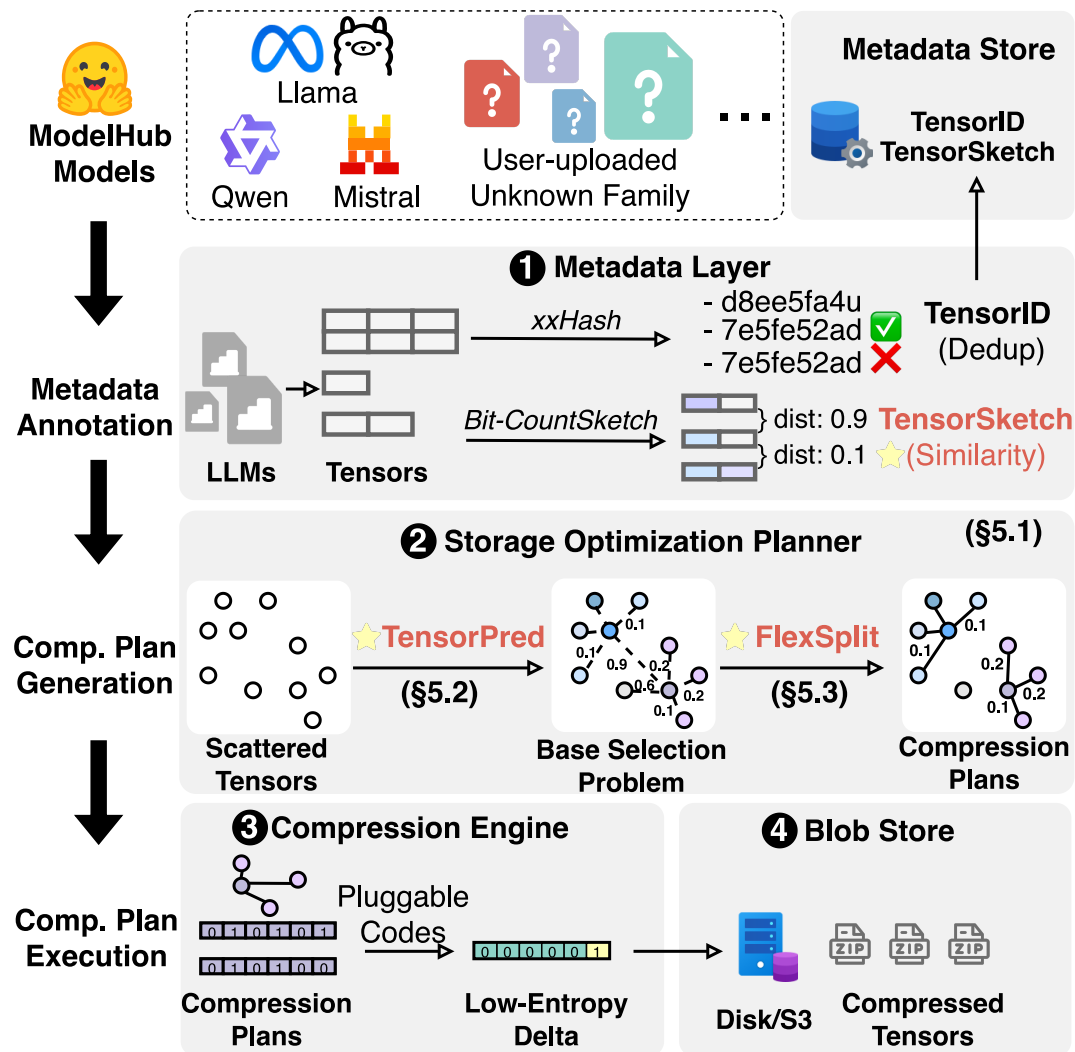
Jointly choose which tensors act as **bases**
and which are stored as **compressed deltas**.

NP-hard and needs integer linear programming (ILP)

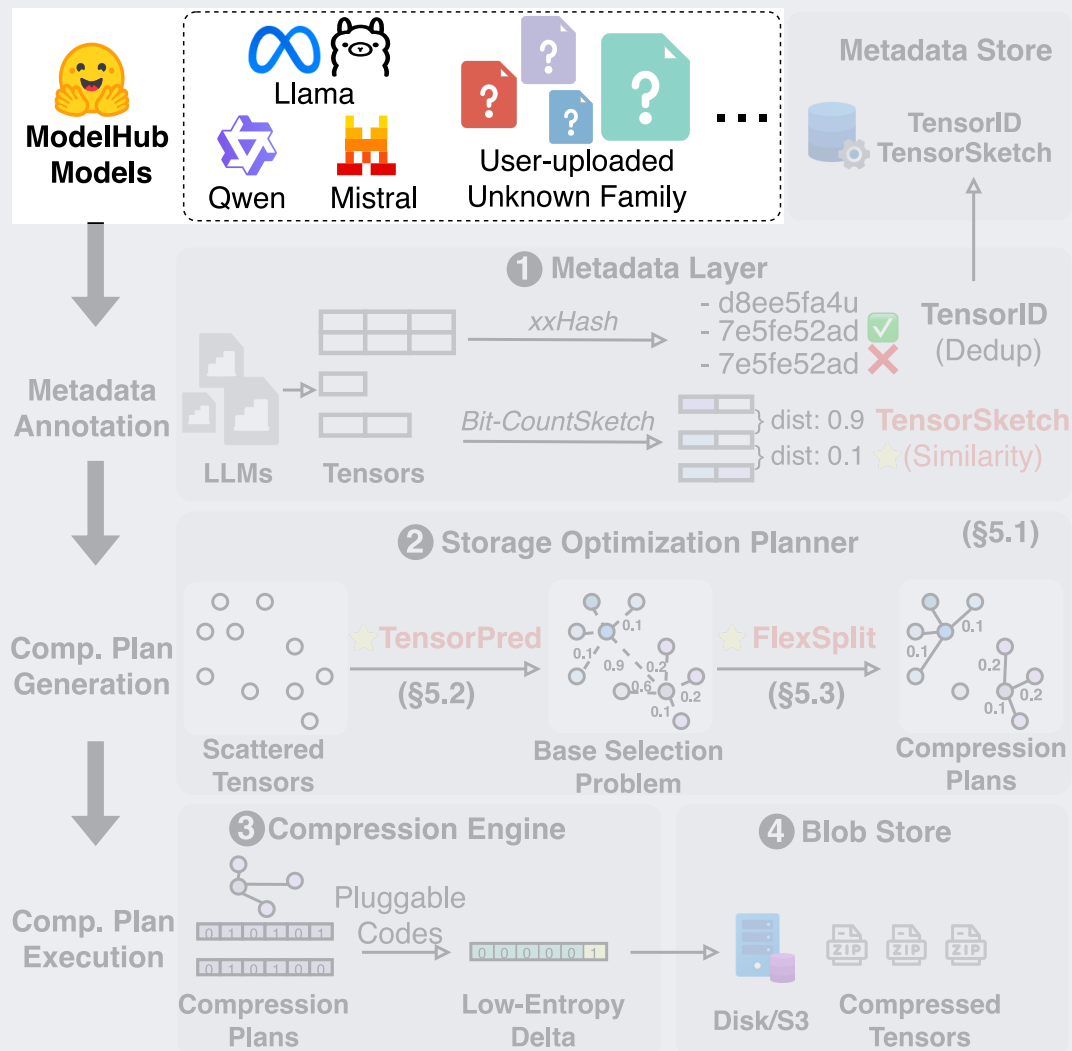
Optimization Challenges

- C1: Estimating pairwise tensor **similarity** is expensive.
- C2: Similarity does not imply **compressibility**.
- C3: **Selecting bases** is combinatorial and dynamic.

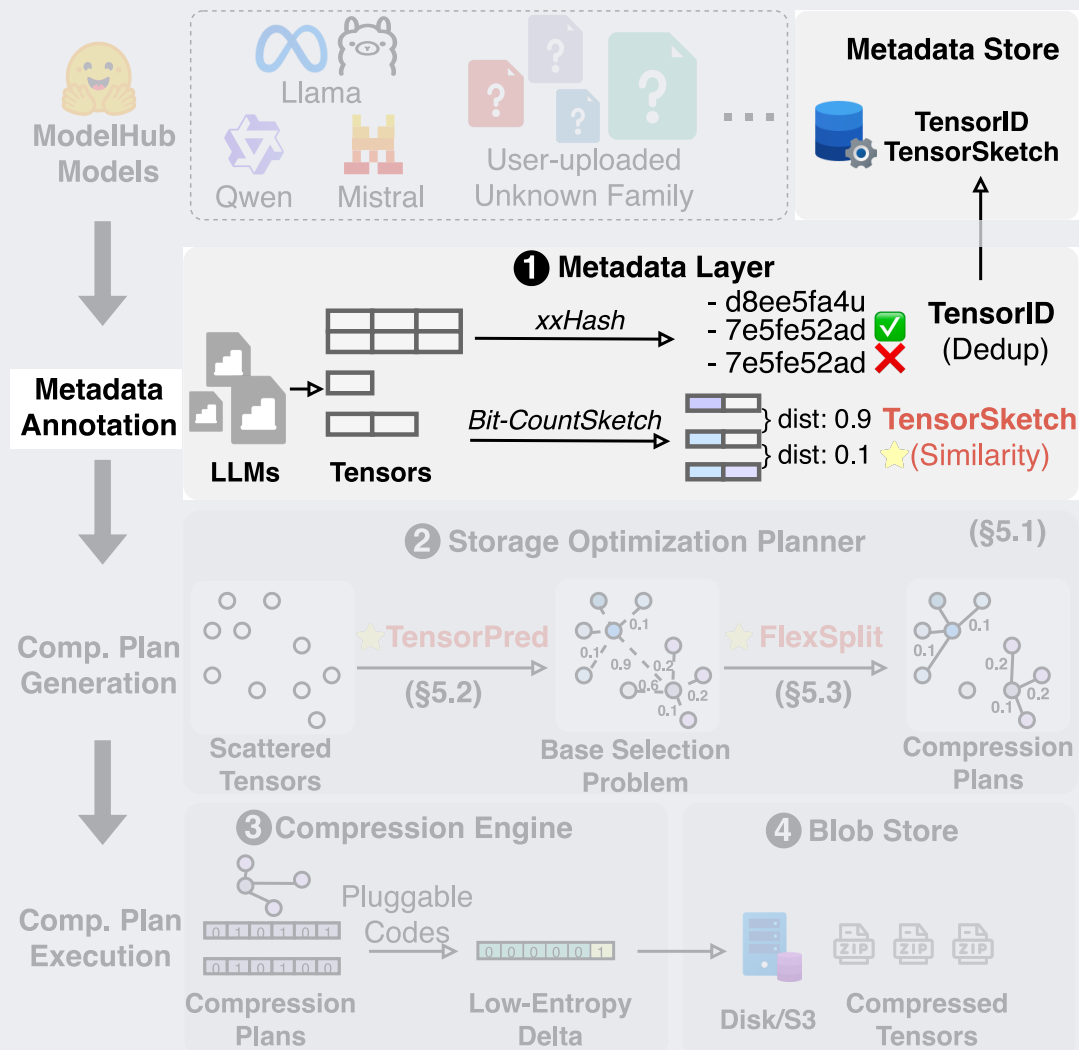
System: TensorDex



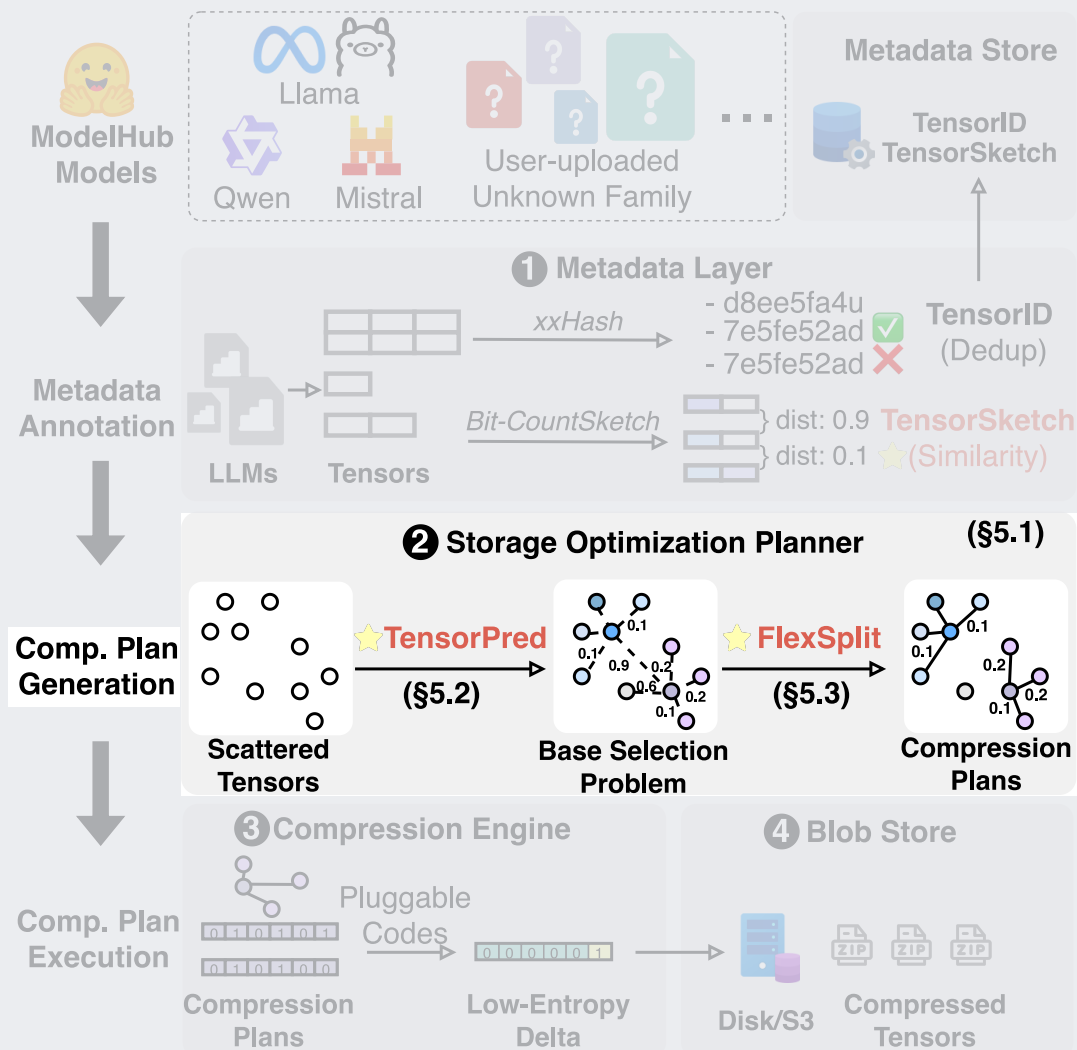
System: TensorDex



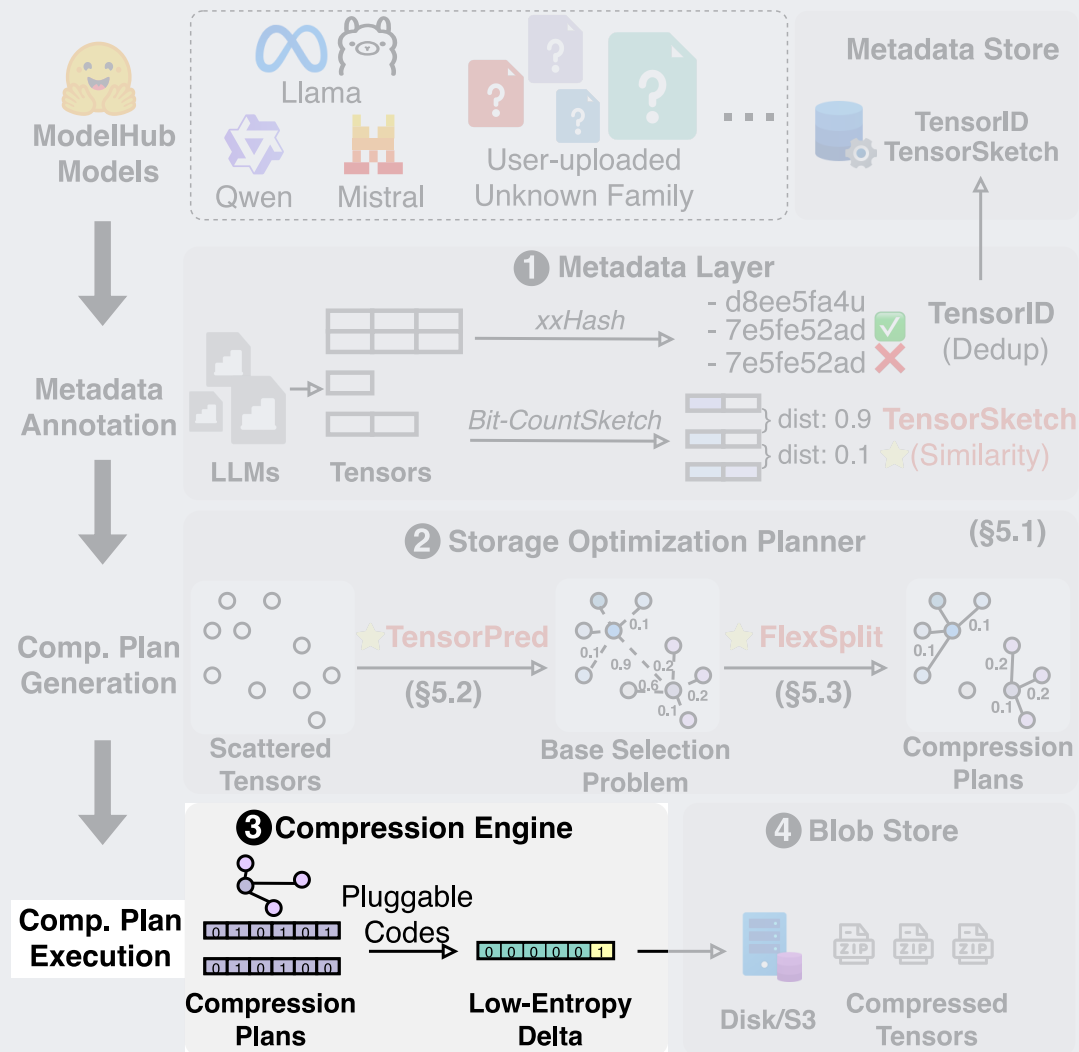
System: TensorDex



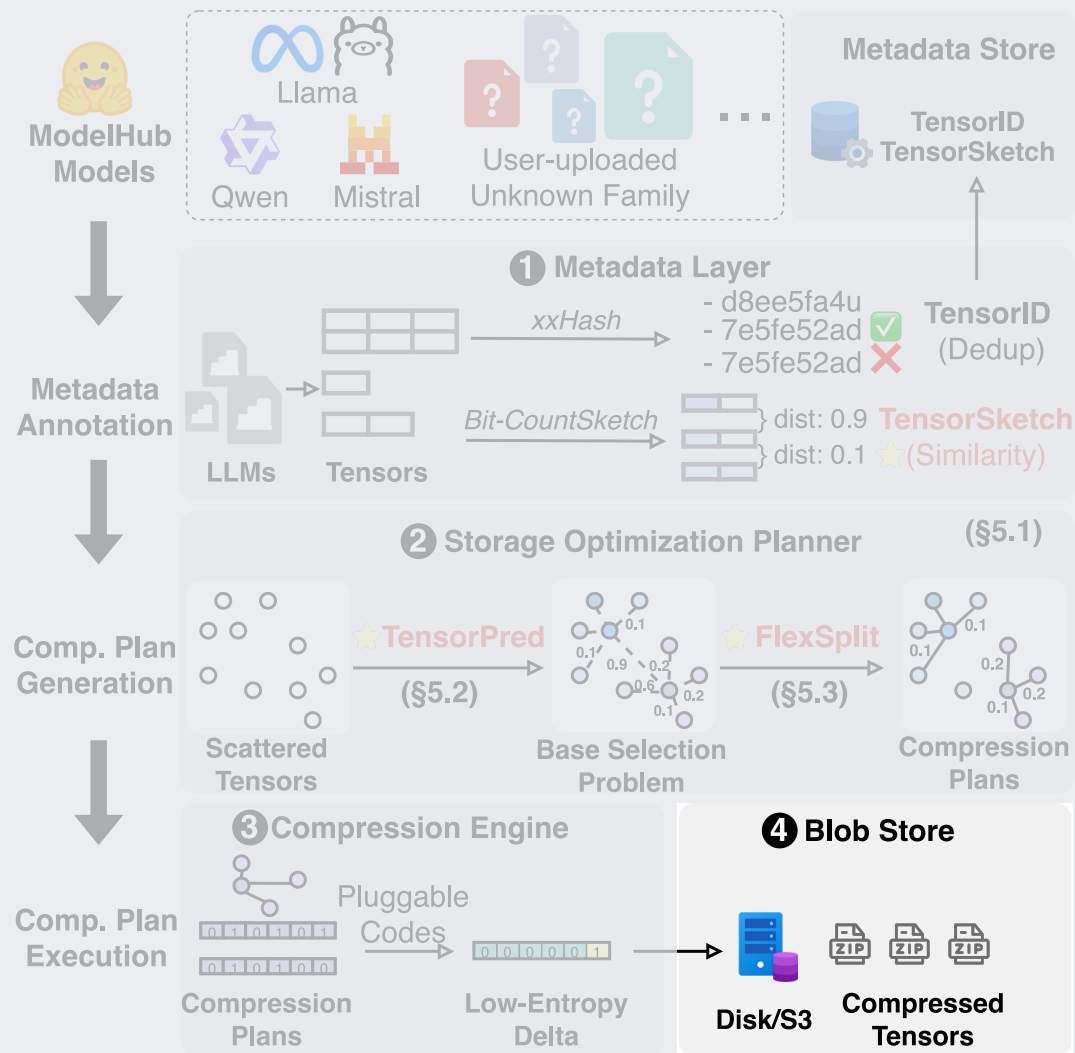
System: TensorDex



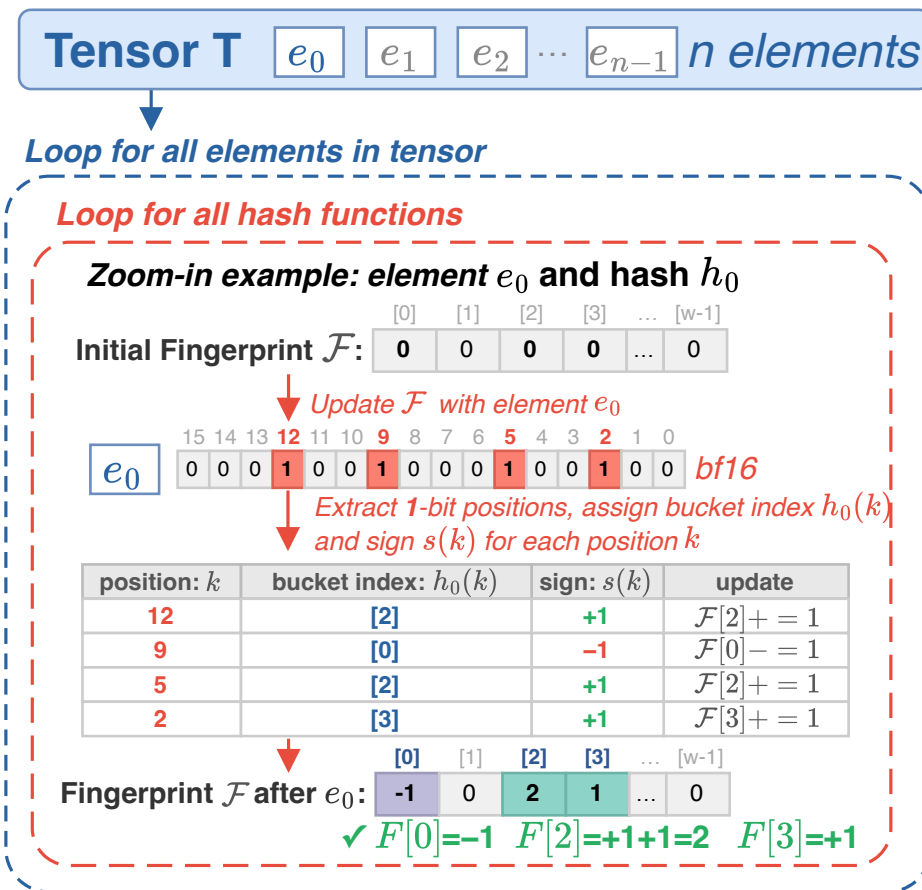
System: TensorDex



System: TensorDex

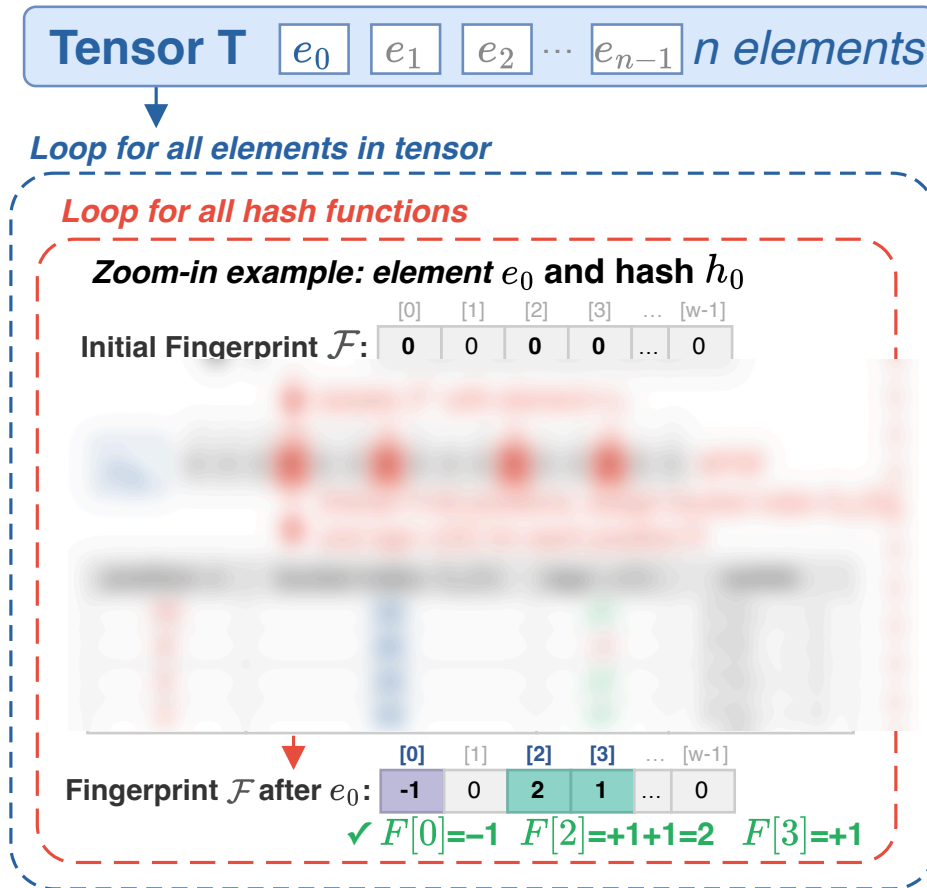


1. TensorSketch: HUGE Tensor $\rightarrow$ tiny Embedding



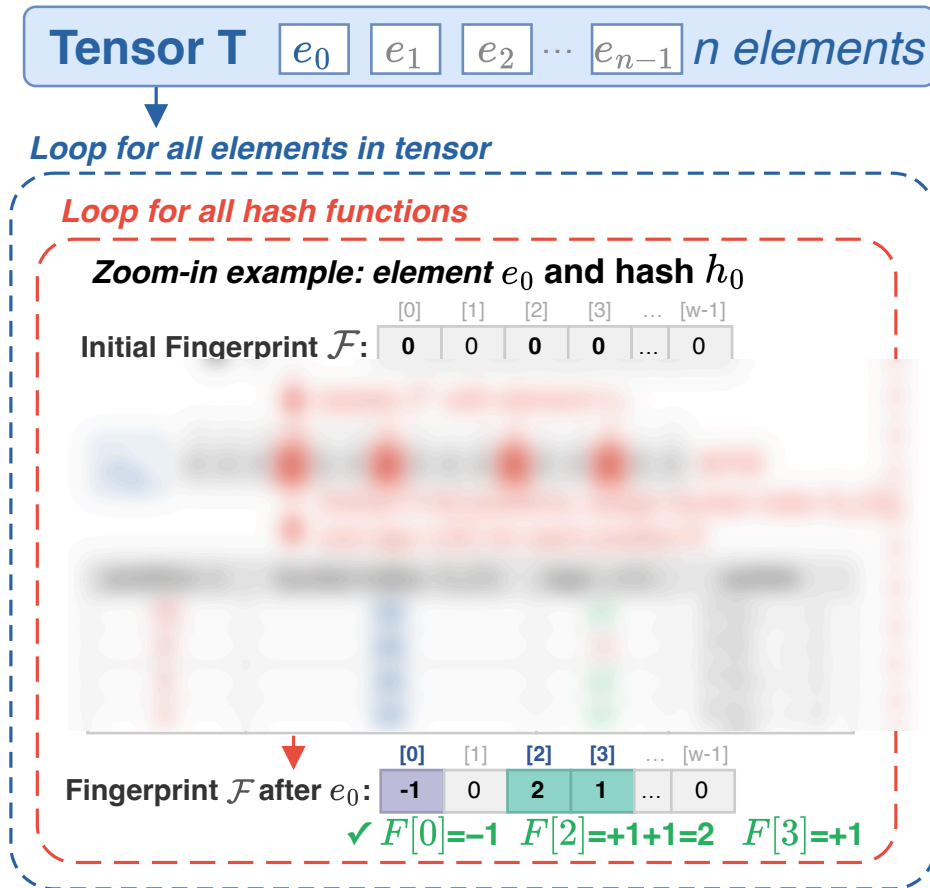
TensorSketch turns a tensor into a small fixed-size embedding (or *fingerprint*).

1. TensorSketch: HUGE Tensor $\rightarrow$ tiny Embedding



TensorSketch turns a tensor into a small fixed-size embedding (or *fingerprint*).

1. TensorSketch: HUGE Tensor $\rightarrow$ tiny Embedding



**Vector Similarity Search
with HNSW**

TensorSketch turns a tensor into a small fixed-size embedding (or *fingerprint*).

3&4. Storage: Dedup. Chance

Models	Before dedup.		After dedup.	
	Tensors	Size	Tensors	Size
2,890	961,509	44.86 TB	741,689	40.11 TB

Table 1. Evaluation dataset: models from the ZipLLM trace (Qwen, Llama, Mistral, Gemma).

3&4. Storage: Encoding Codec

- **TensorX**: throughput-oriented, the fast codec
 - Splits tensors into chunks, compressed in parallel
 - Groups bytes by position (e.g., **first bytes of BF16: exponent**)
 - Deltas are mostly near zero, so each group compresses well
- **FM++**: reduction-oriented, the smaller-output codec
 - Lossless, extended from FM-Delta (another paper)
 - Keeps a higher reduction ratio than TensorX

3&4. Storage: Codec's Decoding

Method	BW (GB/s)
ZipNN	9.6
ZipLLM	8.0
OpenZL	19.1
TensorDex-FM++	8.5
TensorDex-TX	28.4

**Table 3. Decoding throughput with 192 threads,
with all data in memory.**

3&4. Storage: Paper Figure1

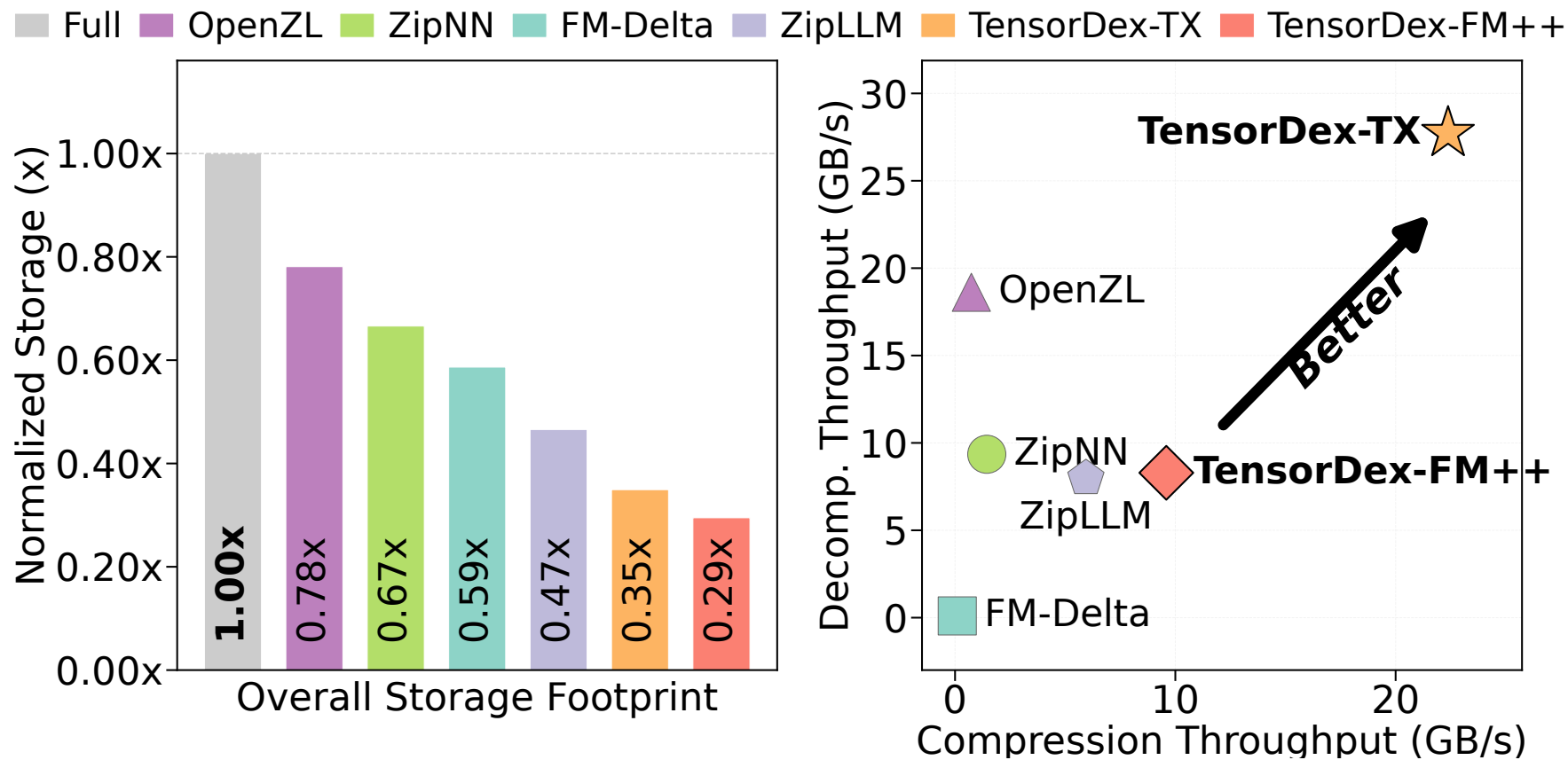


Figure 1. TensorDex cuts storage by 3.39× (left), with high compression and decoding throughput (right).

3&4. Storage: Cloud Storage

Backend	Configuration	GB/s
Codec kernel only (DRAM)		28.6
Full decode path (DRAM)		7.9–8.8
Object store (S3; 64 in flight)	uncompressed pull	3.43
	cold, single-hop dedup	2.83
	warm bases	4.10
Block device (EBS gp3, 2000 MiB/s; 16 in flight)	cold, uncompressed pull	1.70
	cold, single-hop dedup	1.96
	cold, bases resident	2.92
	page-cache warm	4.26

**Table 4. End-to-end retrieval from real storage
(S3 object store and EBS block device).**