SDC | StorageAI™

A SNIA Event

April 29, 2026 • Denver, Colorado

Storage for AI 103

An Introduction to Storage for AI Inference

Curtis Ballard
Co-Chair SNIA Technical Council

Speaker

Curtis Ballard

SNIA Technical Council

HPE

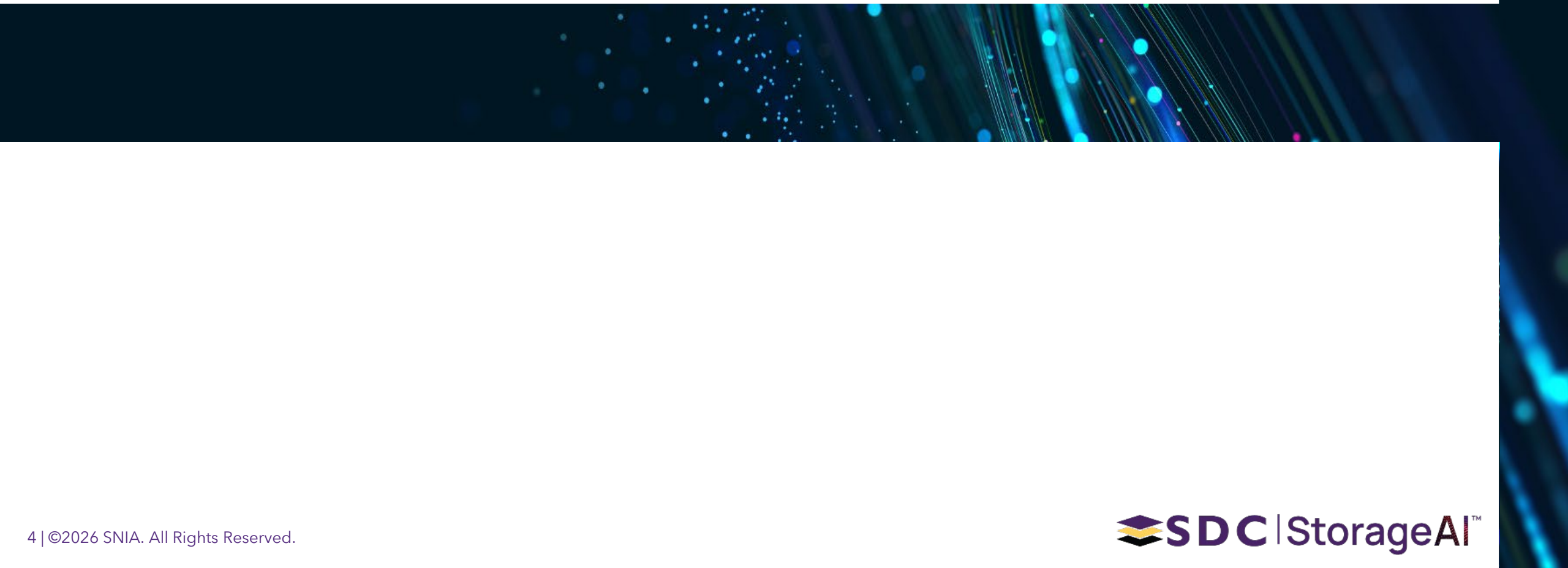
Bio on the SNIA Technical Council Webpage

SDC | StorageAI™

Agenda

- Intro
- Refresh from Storage for AI 102
- Key Concepts
- Using Tokens, Vectors, & Tensors
- Transformer Models
- Some Thoughts on KV Cache Offload
- Recent Related SNIA Presentations

Does AI Inference Need Storage?

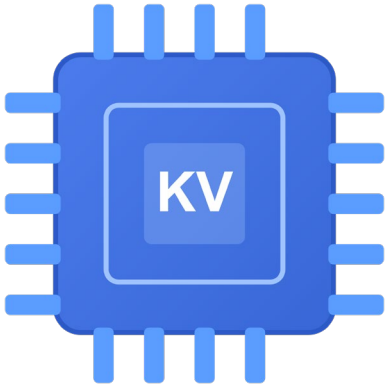


Yes!

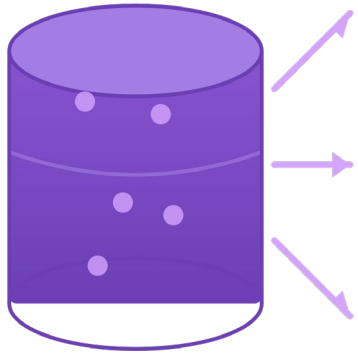
For most AI inference today, storage is a critical component

- Early models, and small models today, can run in GPU memory
- Large models in use today require external storage in addition to GPU memory
- Many models in use today also augment prompts with information from external storage

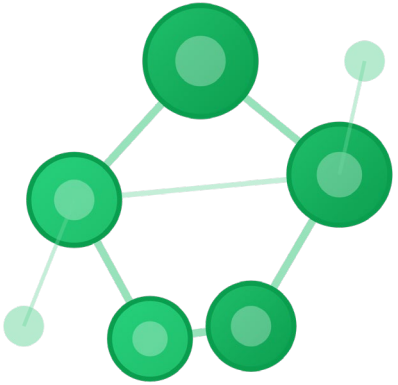
Key Types of Storage for AI Inference



KV Cache or Context Offload



Vector Databases for Data Augmentation



Graph Databases for Knowledge Graphs

Today in Storage for AI 103

- Introductory material on AI Inference and KV cache storage for Inference
  - Foundational background for many presentations to follow
  - What is KV cache
  - WHAT data is being stored when discussing KV cache
  - WHY that data is being stored
  - Later presenters will share viewpoints on HOW
- Presented on behalf of the SNIA Technical Council
  - Primarily based on earlier SNIA presentations
  - Links at the end of this presentation
- Vector Databases & Graph Databases
  - Stay tuned for more Storage for AI 100 series presentations

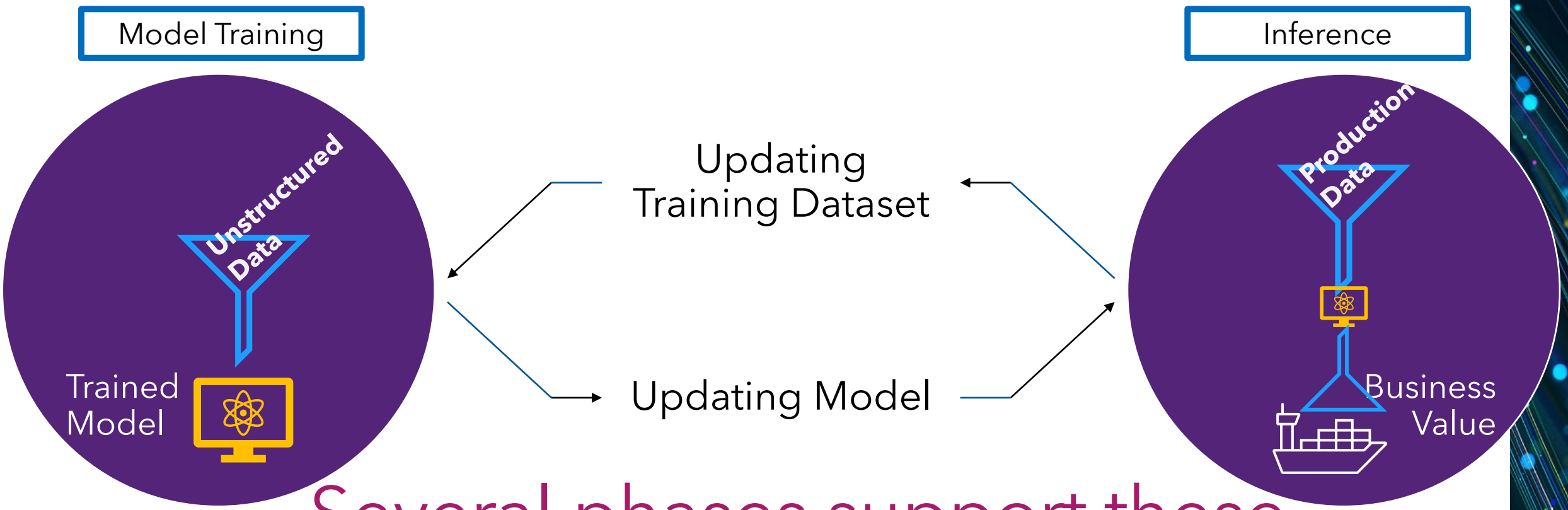
Storage for AI 102 Review

A Few Key Points From:
SDC 2025 - Storage for AI 102*

* https://www.snia.org/sniadeveloper/session/19360

SDC | StorageAI™

The Two Sides of AI: Training and Inference

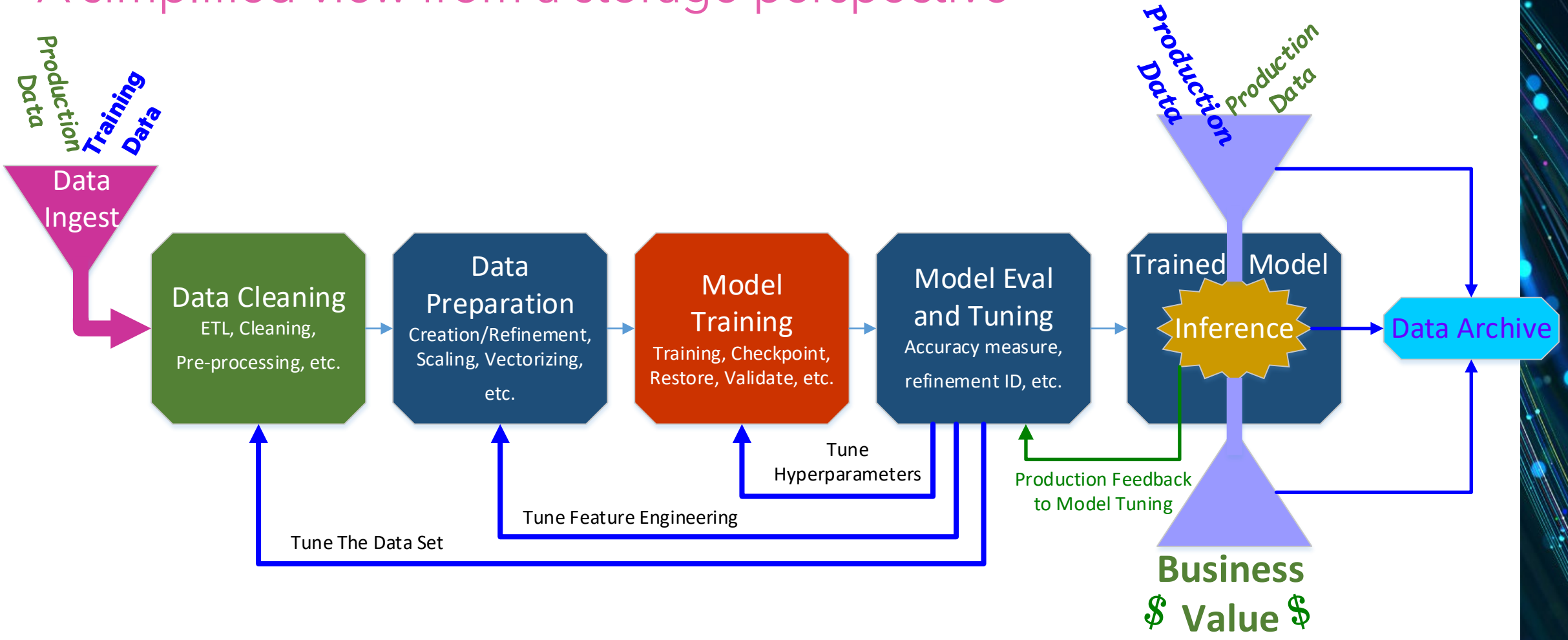


Several phases support these

And Every Phase Uses Storage Differently

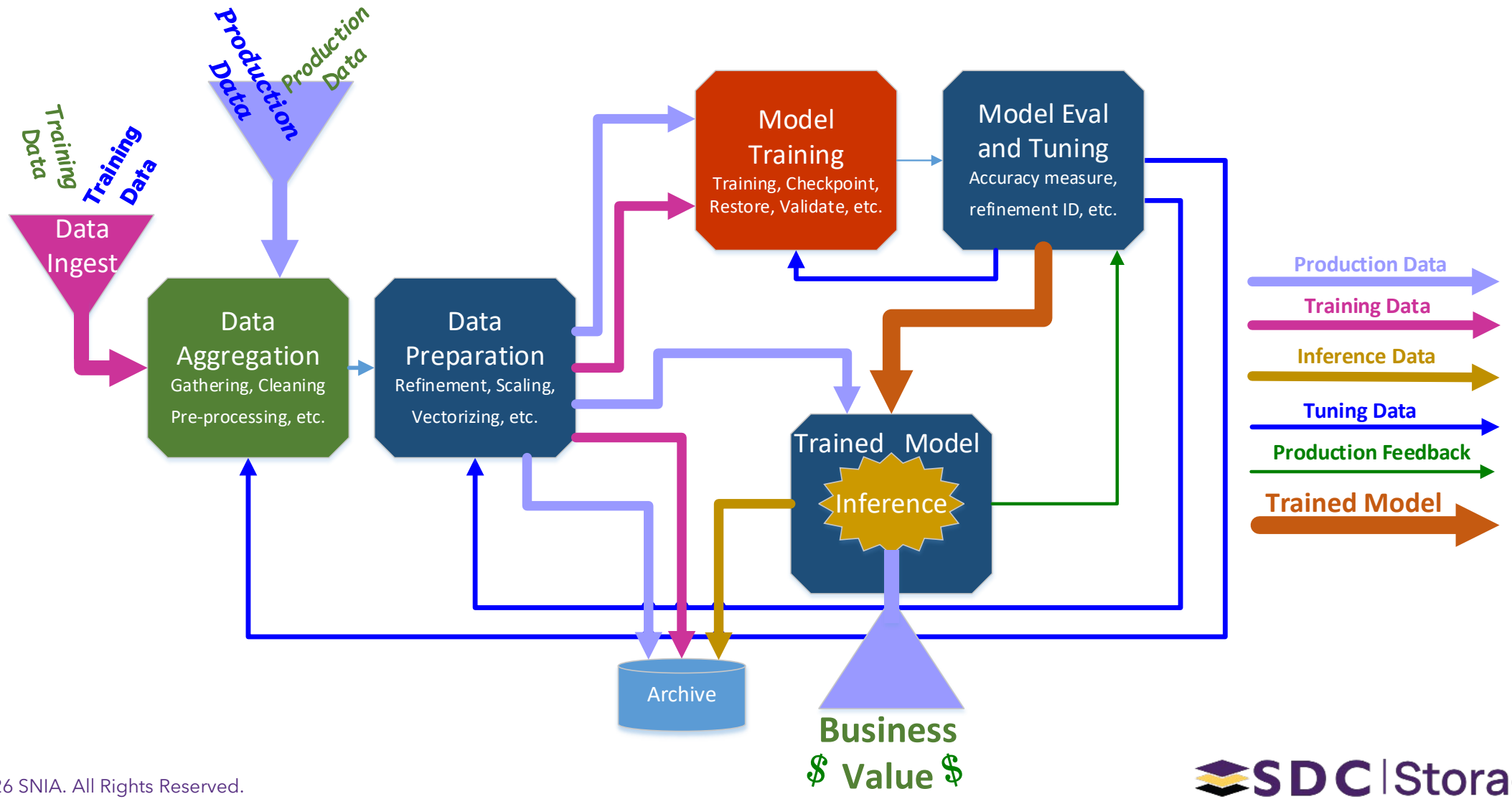
AI Data Pipeline – One Perspective

A simplified view from a storage perspective



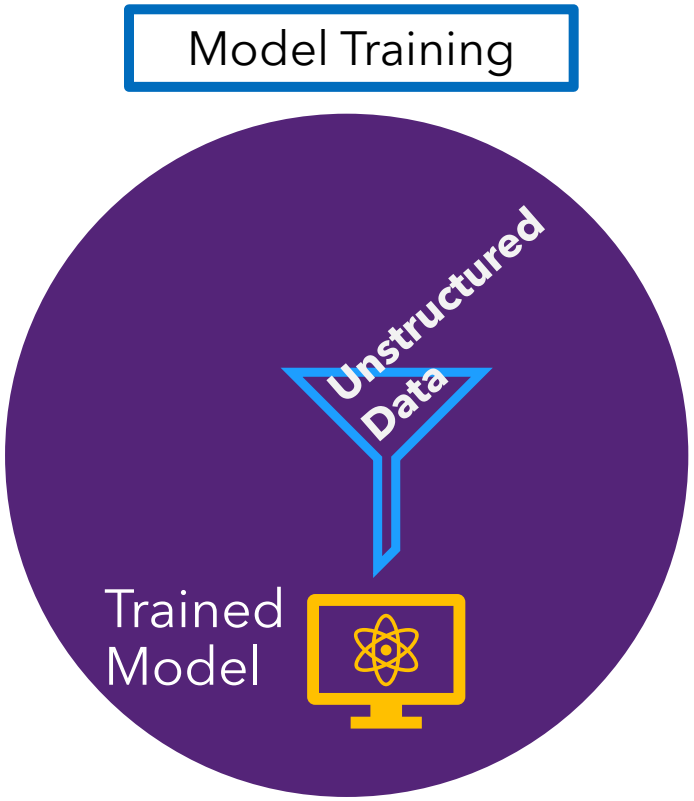
SDC | StorageAI™

AI Data Pipeline – A Storage Data Flow Perspective



SDC | StorageAI™

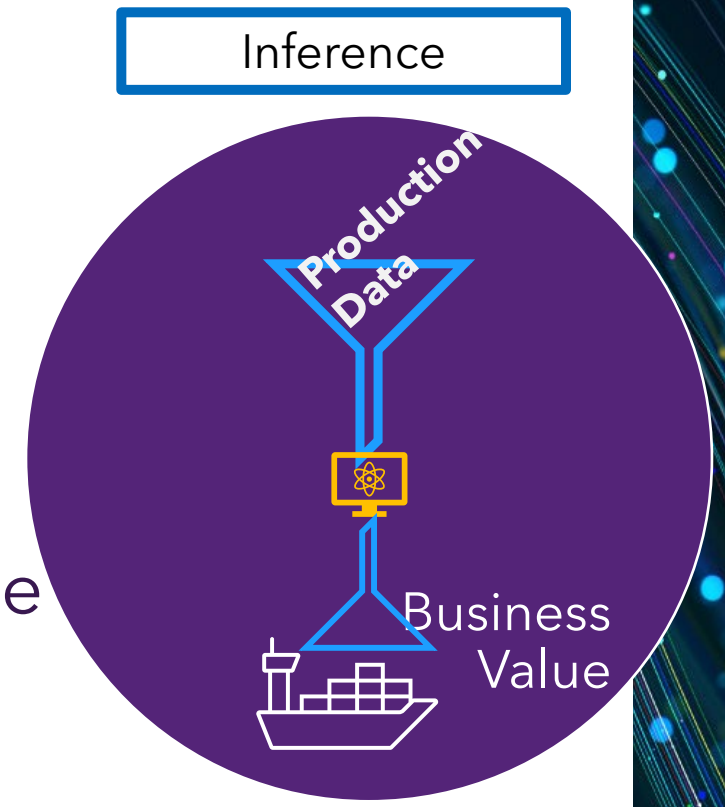
The Two Sides of AI: Training and Inference



A simplified view:
Model training builds the skills that the AI model will use

The Two Sides of AI: Training and Inference

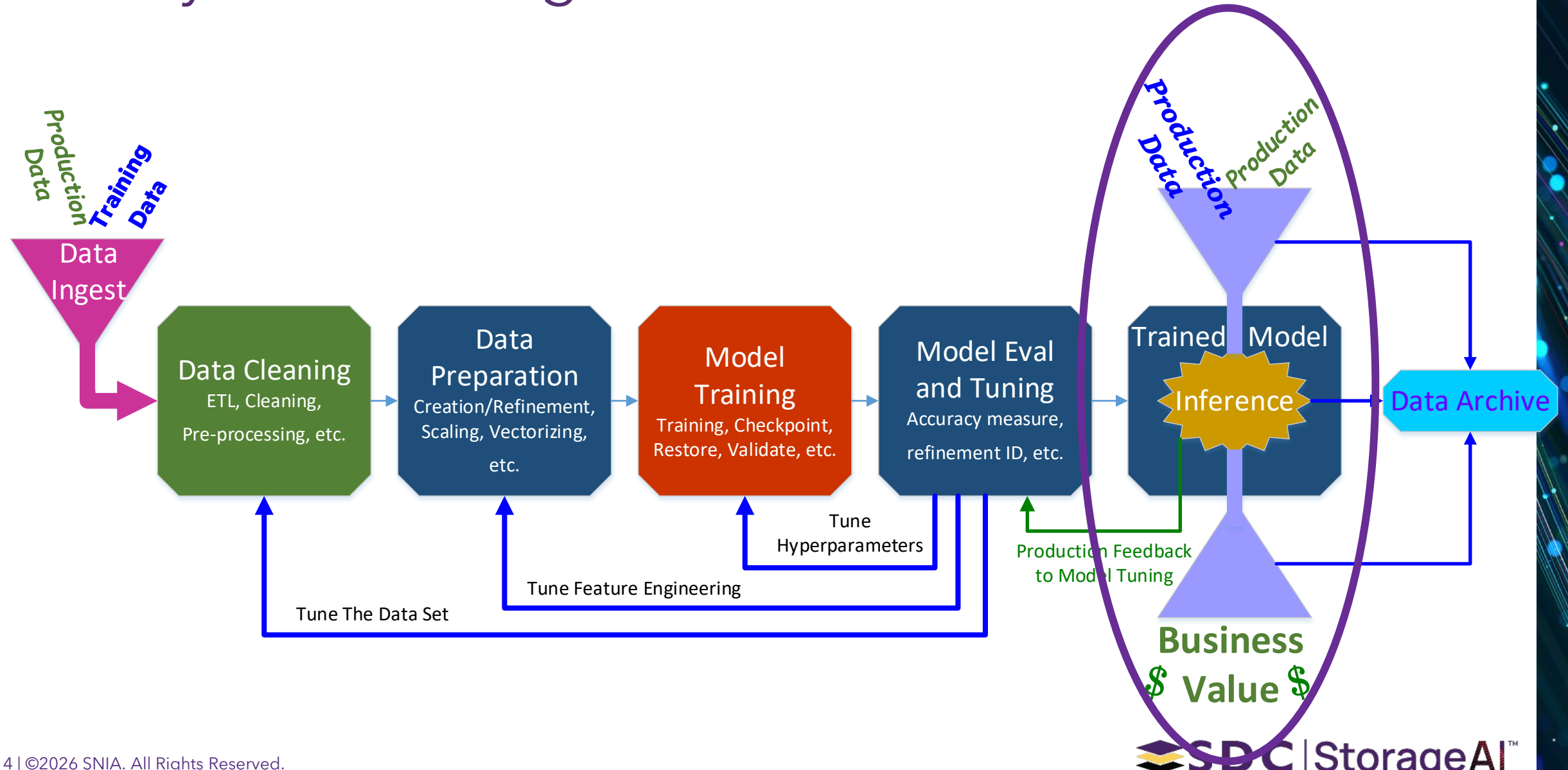
Inference is where most business value is created!



- **Inference:**
  Using an AI model to "Infer" information
  Based on your inputs and the model's training

- Trained models develop pattern recognition techniques which are then applied during inference for tasks like:
  - Making conclusions from the data
  - Using learned patterns/techniques to manipulate the data
  - Generate recommendations

Today We're Going to Focus on Inference

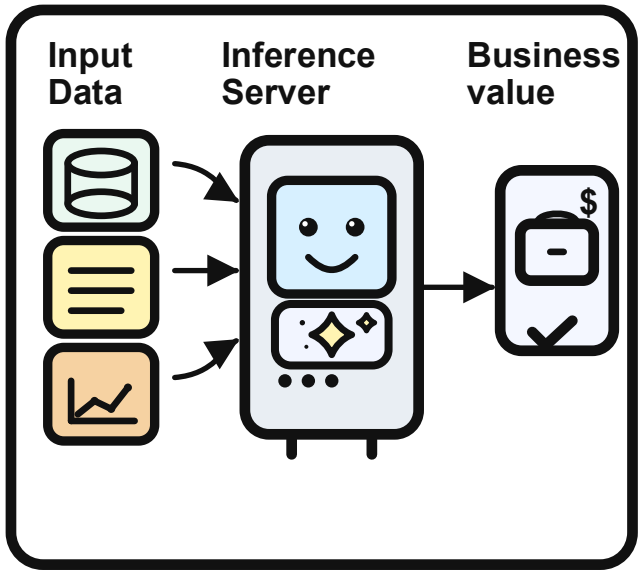


SDC | StorageAI™

Inference Overview

“Inference” – A Broadly Applied Term

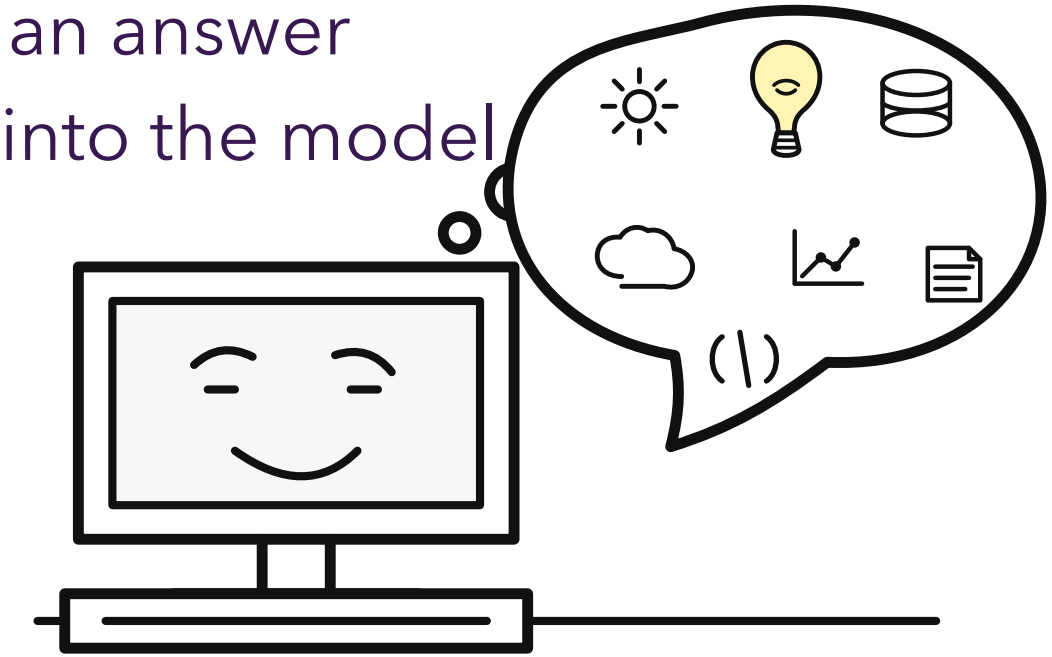
- Inference primarily applies to feeding a prompt into an AI model which processes that prompt to generate an output based on the model’s training



- Inference is often used to refer to all of the steps in using an AI model to generate a result
- In today’s talk we’ll look at several steps for generating results and where those steps need (or can use) storage

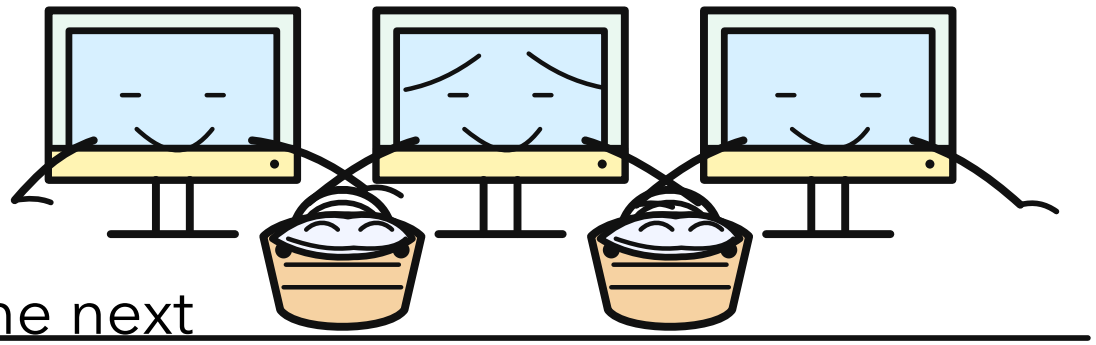
“Early” Forms of Inference

- Most early (or was it yesterday?)
  - Non-Generative
  - Process the input data to create an answer
  - Using existing information built into the model
- Natural Language Processing
- Prediction
- Pattern Recognition



Inference Today

Just a few variations from many



- Multi-turn
  - Information from one step fed into the next
- Information Augmented
  - Search or query adds additional information
  - RAG – Retrieval Augmented Generation
- Information passed from step-to-step
  - Context – a set of information generated by previous steps and available for use in the next steps and which influences the output

Transformer Models – A Brief Introduction

More details to follow

- Most popular AI Models today are Transformer Models
  - Introduced in the famous "Attention is All You Need" paper (Vaswani et al., 2017)*
- Evaluate all parts of a prompt against each other
  - How much does each part relate to, or influence, other parts?
- Build a contextual representation
  - Context is retained across steps and influences the following steps

* https://arxiv.org/abs/1706.03762

Going Deeper

Key Concepts

Tokens

The smallest unit of data processed by an AI model

- Tokens are numeric representations of the smallest data portion
  - For text, a word, or a part of a word
  - Usually: ~4 characters

- Tokenization is converting the data to representative numbers

“Every”	=	3582
“cloud”	=	2439
“has”	=	1384
“a”	=	7786

These numbers are model-specific

Vectors

The representative form used by AI models

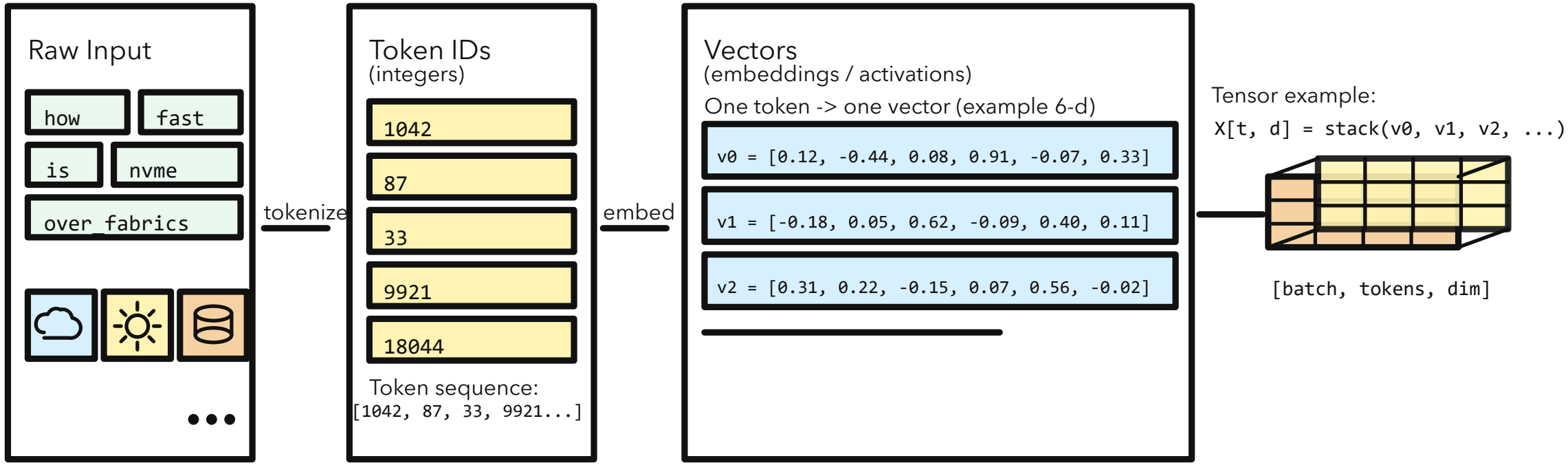
- Vectorization – converting tokens into model-specific vectors
- An AI model has an "embedding matrix" – a table of vectors with model-specific encoded meaning
- Tokens are used to perform a lookup to the matching vector
- The vector is a fixed length array of numeric values with model-specific meaning

- Example: "Buick" → 8472 → [0.64, -0.22, 0.71, 0.09, -0.15, 0.48]

Tensors

The core data unit processed by neural networks

- A stacked set of vectors becomes a tensor
- Usually multi-dimensional stacks of vectors

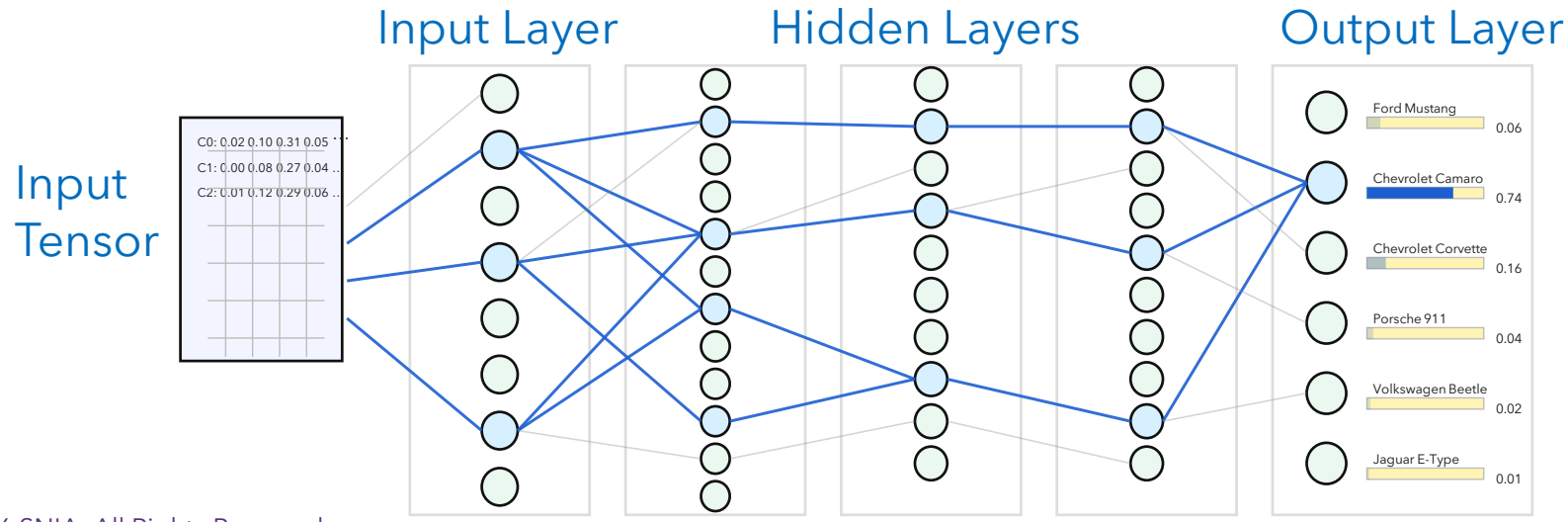


Using Tokens, Vectors, & Tensors

Deep Learning Models

Deep Learning

- Neural networks with many layers
- Visible layers at the edges – Input layer and Output Layer
- Input is passed through many hidden (deep) layers
- Weights learned during training are applied in different layers
- Generates output probabilities



SDC | StorageAI™

Transformer Models

A High-Level Introduction

Transformer Models – a Bit Deeper

What is Attention?

- Now that we've learned about tokens, vectors, and tensors
  - Transformer models are a specific type of deep learning model
  - Also referred to as Attention, or Attention-Based models

1. Tokens are embedded into vectors, with no relational information
2. Attention is applied to identify relationship between tokens
3. Tokens are weighted based on their relationship
4. A new vector is created combining the weights and vector values
5. The new context-aware vectors are combined into new tensors

What is Context?

- The model's internal state capturing how the tokens are related
- Created by applying attention
- Represented as a set of vectors
  - Q – Query
  - K – Key
  - V – Value

- The K & V vectors are updated at every layer
- A KV cache (Key-Value cache) is the set of K&V vectors that can be used to reconstruct the context for new tokens in the same sequence

Key Phases For Transformer Models

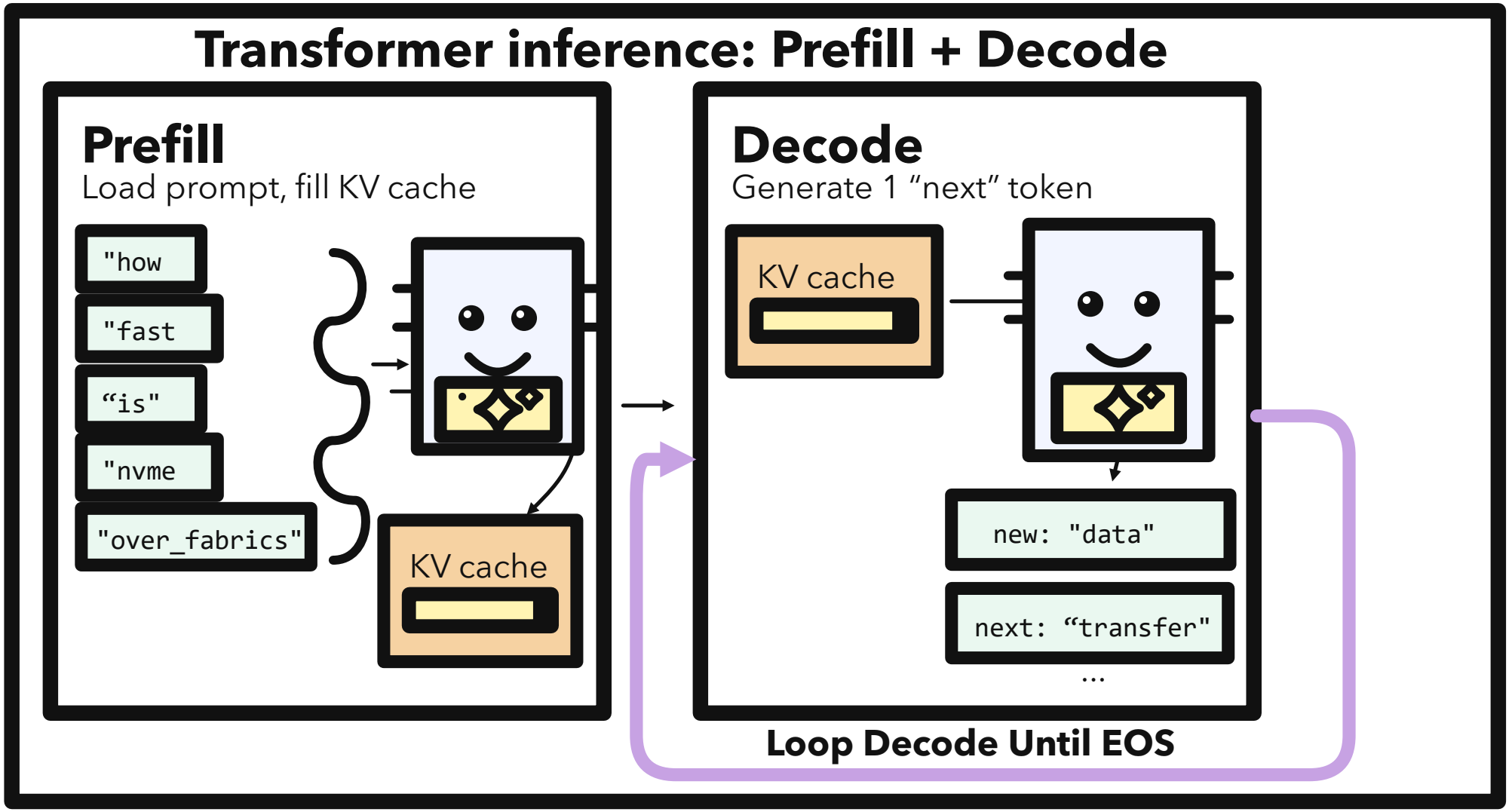
Prefill

- Load the prompt
- Include augmentation data (e.g, RAG)
- Tokenize/Vectorize all input data
- Forward pass through all layers
  - All tokens processed in parallel
- Build the KV cache of attention state
- Predict the first “next” token
- Affects “Time To First Token” (TTFT)

Decode

- Use KV cache from prefill
- Use predicted “next” token
- Forward pass for THAT token
  - One token through all layers
- Compute attention against KV cache
- Generate new KV for KV cache
- Predict the next “next” token
- Repeat until “End of Sequence” (EOS)

Graphically Represented Prefill and Decode

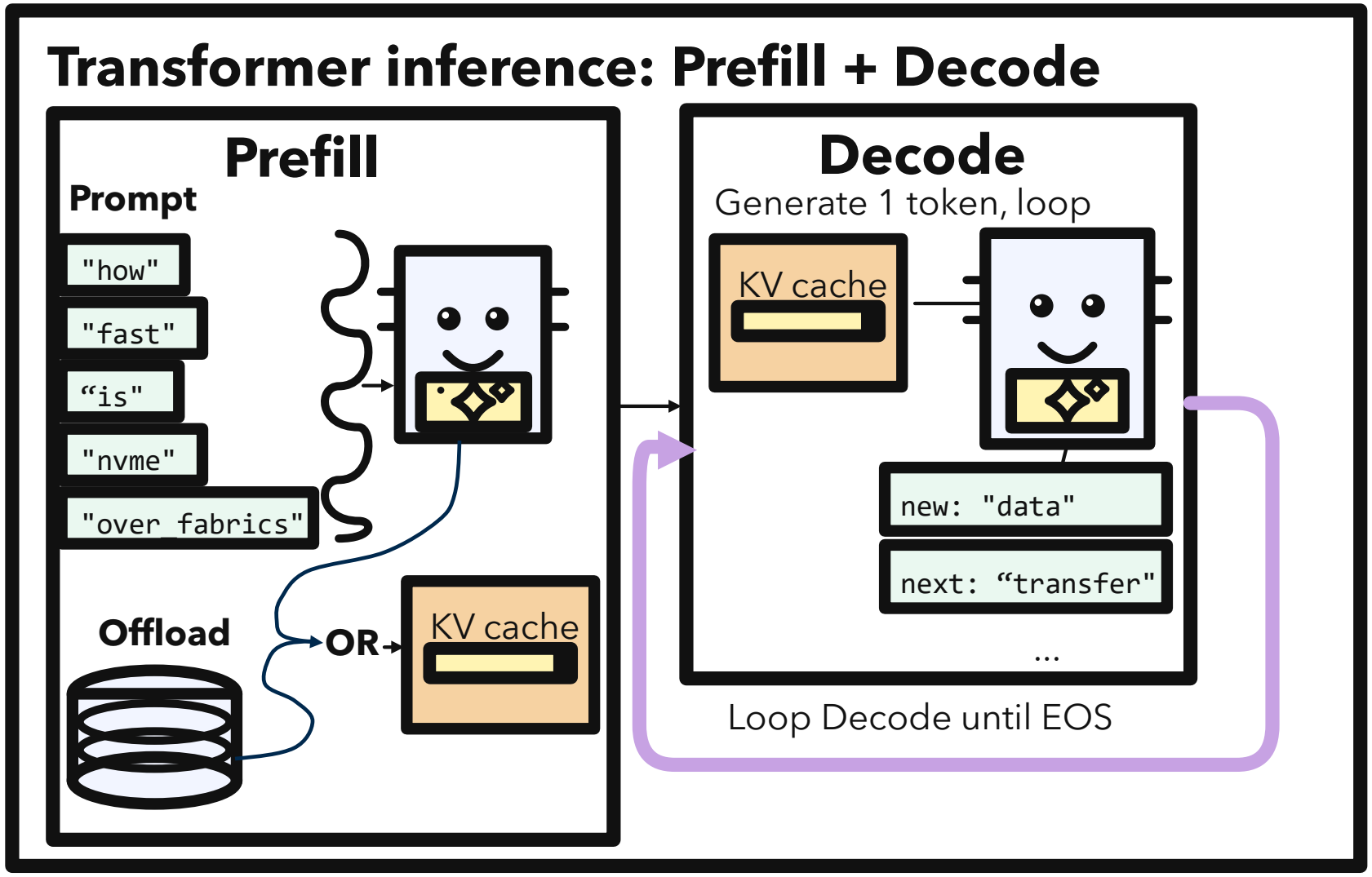


At Last! Some Storage – KV Cache Offload

- The KV cache can be offloaded at a processing boundary
  - For example, a pause in a chatbot conversation
- The KV cache contains the Key and Value (KV) tensors that represent the attention state after processing the input tokens
- The output of the prefill phase can either be created by:
  - Loading the prompt data and calculating the KV cache

  Or
  - Loading a saved copy of the KV cache from the same token sequence

Graphically Represented KV Cache Offload



Some Thoughts on KV Cache Offload

How Big is KV Cache Data

- Determined by the model and the context size
  - Relatively large and mostly growing
  - Simple formula to calculate using model parameters
  - Model-specific and general calculators available
- Parallel sessions multiply the KV cache size
- A few examples*

Context Size	Mistral-7B	Llama-3.1-70B†	DeepSeek-V3‡
4K tokens	0.5GB	1.25GB	0.27GB
32K tokens	4GB	10GB	2.14GB
128K tokens	16GB	40GB	8.58GB

† Llama-3.1-70B uses GQA (Grouped Query Attention), for reducing KV cache size
‡ DeepSeek-V3 uses MLA (Multi-Head Latent Attention), for reducing KV cache size

* https://github.com/Potterluo/KVCache-calculator-Skill

Is KV Cache Offload Worth It?

- The answer is some variation of “It Depends” and “Yes”
- Recomputing KV cache consumes GPU cycles and takes time
- Loading KV cache from storage takes time

- General model:

If the KV cache can be loaded faster than it can be recomputed, then KV cache offload can reduce time to first token (TTFT) (i.e., get results faster)

Other Reasons for KV Cache Offload

It isn't just used for paused sessions

- KV cache can become the largest consumer of GPU memory
  - Offloading inactive KV cache frees up GPU memory
- Additional value in GPU clusters
  - Enable moving session between GPUs
  - Allow pausing/resuming sessions (e.g., multi-tenant time slicing)
  - Support tiering from memory to storage to support longer context
  - Allow graceful degradation, apply Quality of Service rules to limit resource starvation
  - Disaggregate prefill and decode (e.g., to optimize hardware for each)
  - Fault tolerance

Offloading KV Cache Data, A High-Level Overview

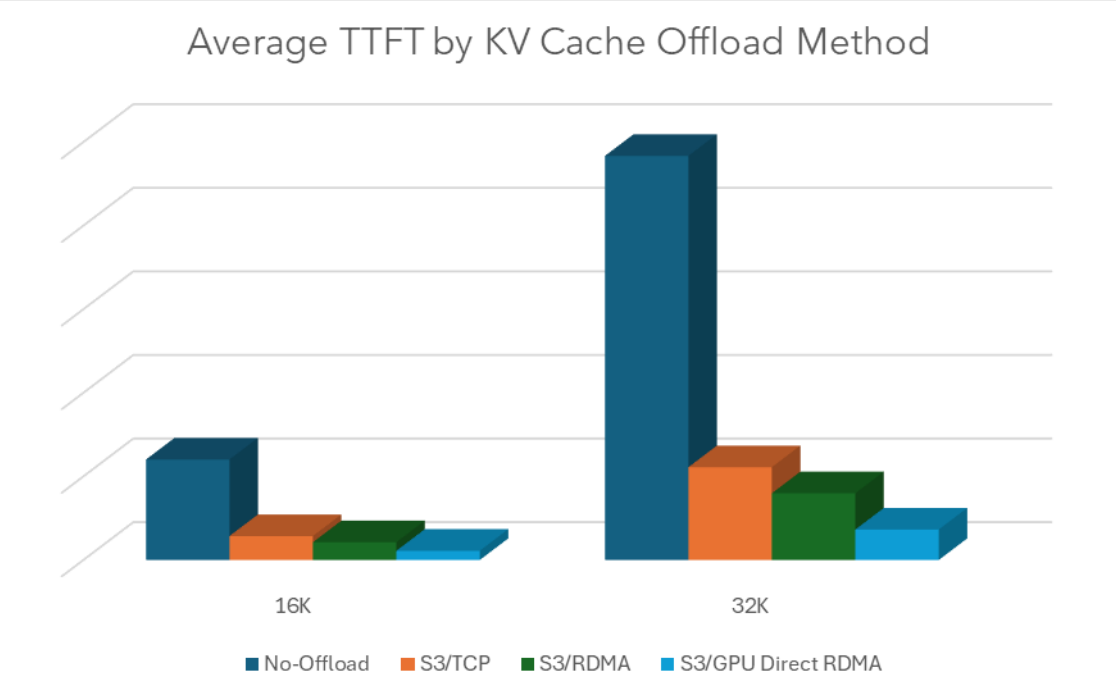
General techniques - there are new ideas almost daily

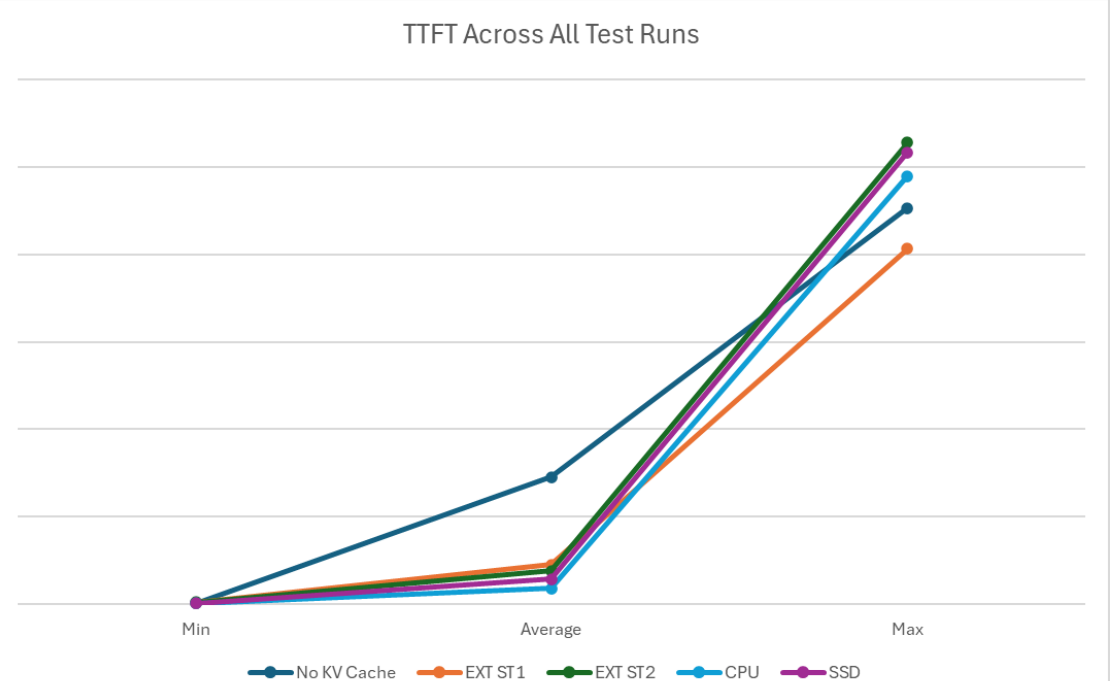
After a decision is made to offload the KV cache

- Snapshot the KV cache metadata (e.g., session, layers, etc.)
- Mark selected KV entries as inactive if partial offload
- Preprocess KV tensors (e.g., layout normalization, compression)
- Segment into blocks of vectors and package for storage
- Write to storage, method may include:
  - Block write to local NVMe (CPU-Initiated or GPU-Initiated)
  - Block to NVMe-oF (usually RDMA)
  - Emerging: RDMA accelerated object storage (SNIA TWG exploring)

A Couple of Views on KV Cache Benefits

Accelerating Time to First Token (TTFT)





SDC | StorageAI™

Prior SNIA Presentations

Other SNIA presentations that cover material related to this presentation*

- Eshcar Hillel (Pliops) Disaggregated KV Storage: A New Tier for Efficient Scalable LLM Inference
- Ugur Kaynar (Dell) KV-Cache Storage Offloading for Efficient Inference in LLMs
- Alessandro Goncalves (Solidigm) Scaling RAG with NVME: DISKANN's Hybrid Approach to Vector Databases Indexing
- CJ Newburn and Wen–Mei Hwu (NVIDIA) Storage Implications for the New Generation of AI Applications
- Wes Vaske (Micron) Discussion and Analysis of the MLPerf Storage Benchmark Suite and AI Storage Workloads
- Jason Goldschmidt (Dell) Accelerating Object Storage for AI/ML with S3 RDMA
- Himabindu Tummala, John Cardente, Fatemeh Azmandian (Dell) and Michael Hoard (SNIA)
  How Agentic AI Transforms the Role of Storage
- Kevin Marks (Dell), Luis Freeman (IBM), Tim Lustig (NVIDIA) AI Model Inferencing and Deployment Options
- Jayanthi Ramakalanjiyam (Celestica), Justin Potuznik (Dell), Tim Lustig (NVIDIA), Erik Smith (Dell)
  Introduction to AI and Machine Learning
- Anil Godbole (Intel), Andy Mills (SMART Modular Tech.), Steve Scargall (MemVerge)
  Accelerating AI with Real-World CXL Platforms

* Presenter names taken from SNIA webpage or from the slides


SDC | StorageAI™
A SNIA Event
Thank You