Don’t Let the IO Blender Destroy your AI model Training

Mike Bloom

Office of CTO

AI Solution Architect

The Modern Enterprise AI environment

Classic HPC Environments are converging with Accelerators & AI workloads

- Sharing the same data sets
- Simulation tests can be closely predicted via AI (https://www.anl.gov/nse/ai-ml/surrogate-models)
- Math is being distributed to benefit from thousands of accelerators cores in parallel
- More…

Result

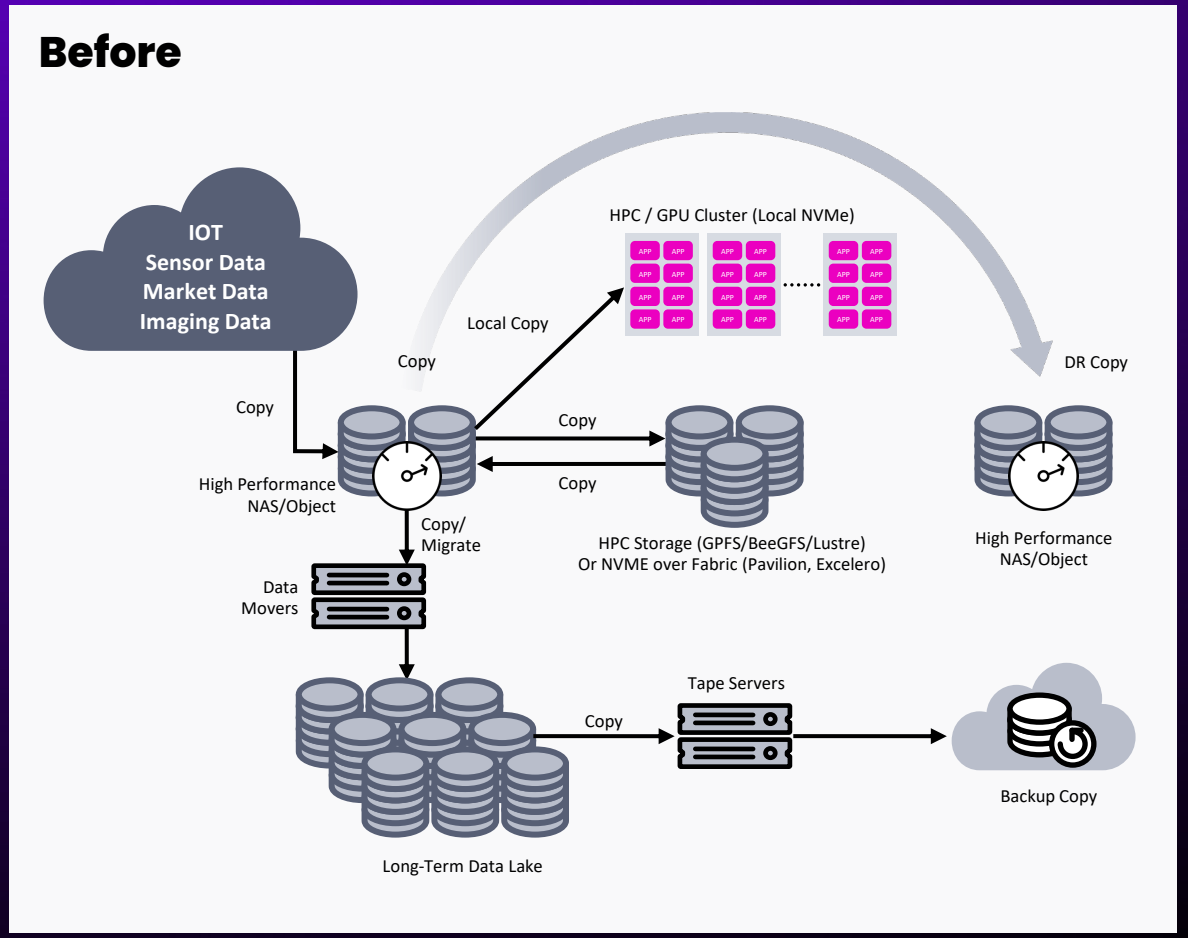
- Billion of files
- Varying IO sizes
- Varying IO access patterns

NET-NET: IO in these environments is radically different from classic ENTERPRISE file access or classic HPC Large files + Large files IO access

How did we get here?

- HPC and Enterprise infrastructure sprawl based on performance and capacity capabilities
  - AI practitioner survey shows 32% of responders say infrastructure is the top challenge to overcome

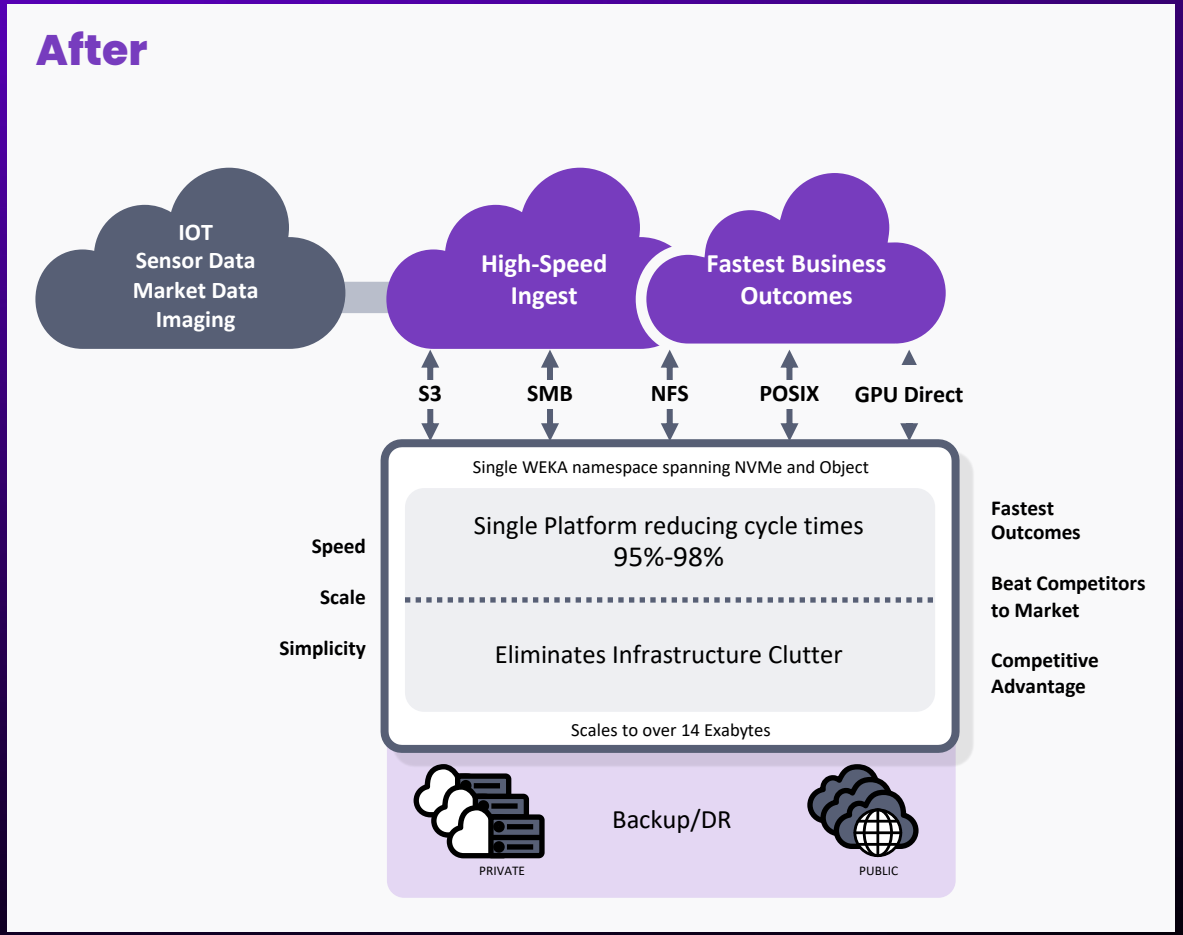
- Data Copying=“Data Stall”
  - Up to 90% of an epoch time used for copying data
  - GPU waiting for data



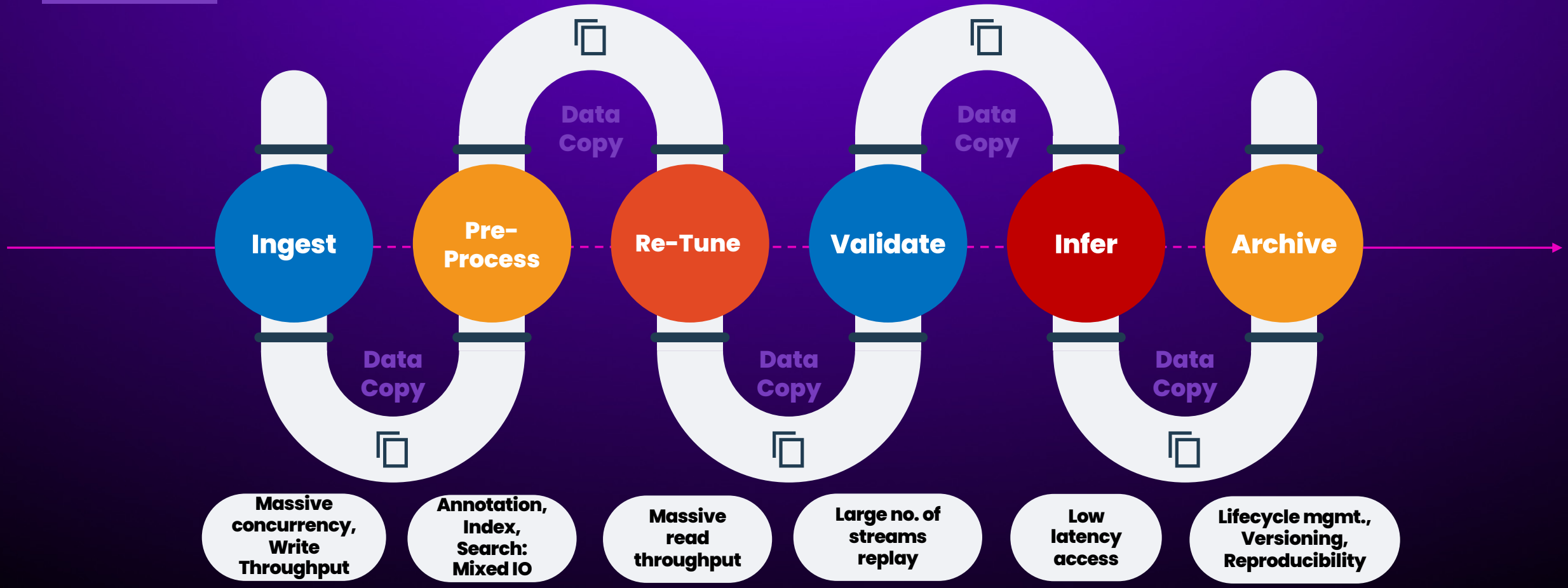
How did we get here?

- Customers consolidate workflows onto a single data platform due to:
  - Resource availability pressures
  - Prevention of “Silo Sprawl”
  - Ease of management
  - Cost effectiveness

- Consolidated data=”IO Blender”
  - Overlapping AI pipeline data creating contention for all available IO
  - WEKA can handle the IO blender, but not everyone else can.



A GenAI pipeline

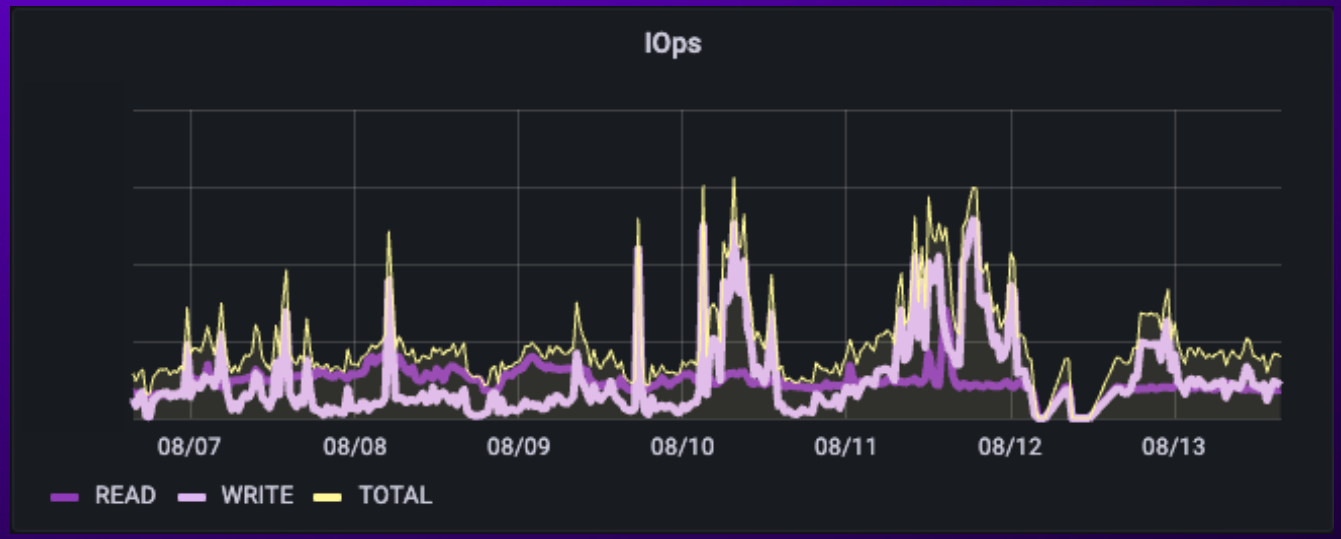


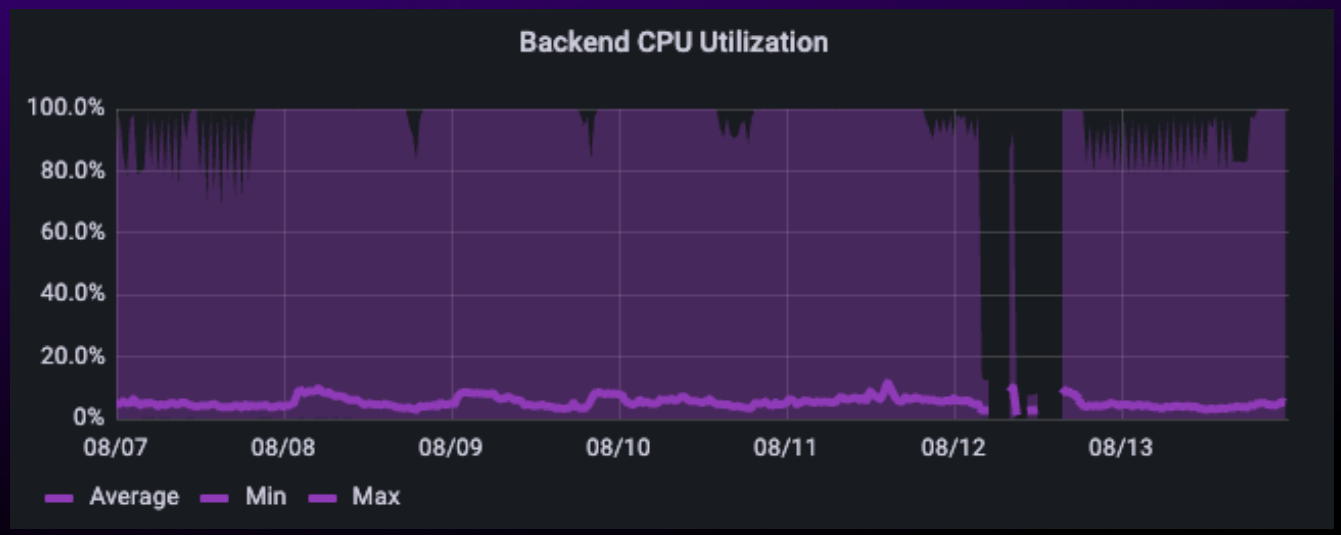
WEKA

Data Profiles of Different AI Pipelines

Natural Language Processing

- Use case: audio-text and audio-action
  - Lots of natural language parsing
- Mixed IO with relatively consistent reads, but significant IO spiking on writes
- Backend utilization has low average, but Max shows IO is bursting up and down constantly.
  - Needs to be able to burst high very quickly with low latency.

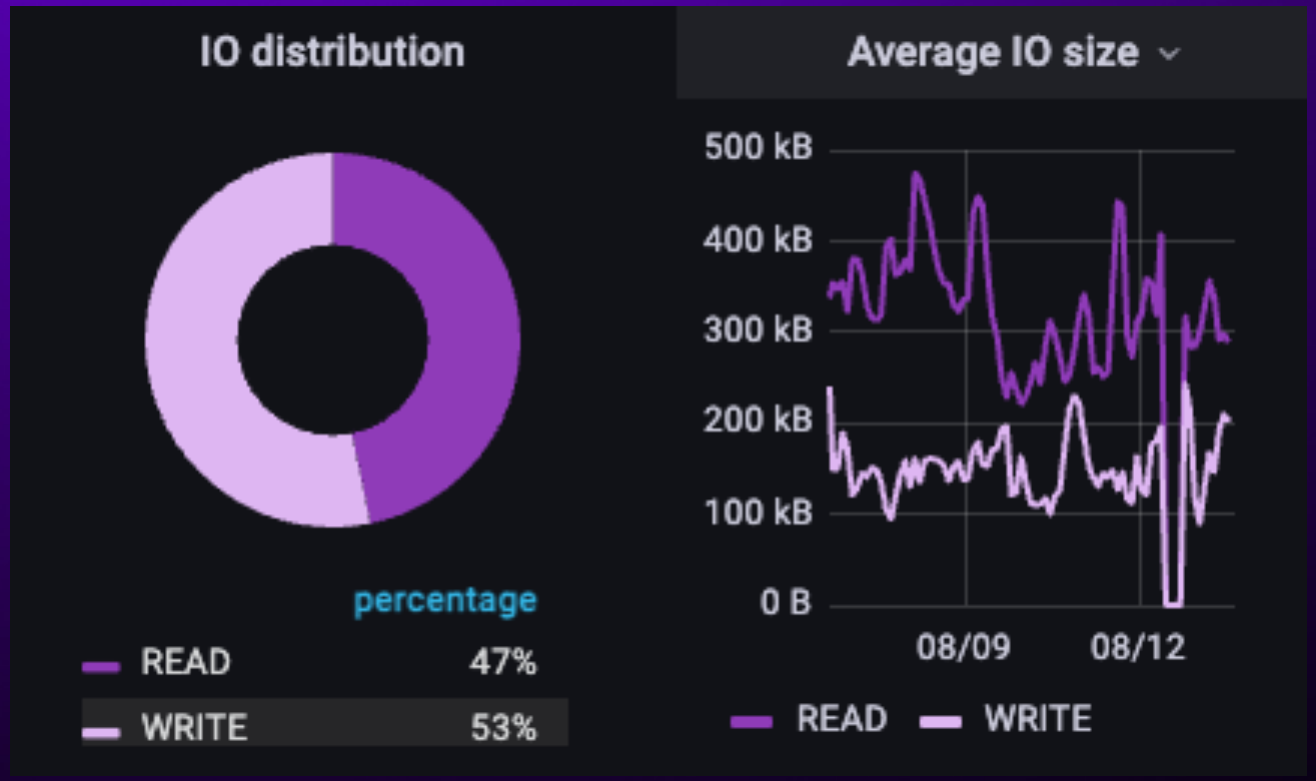




Data Profiles of Different AI Pipelines

Natural Language Processing

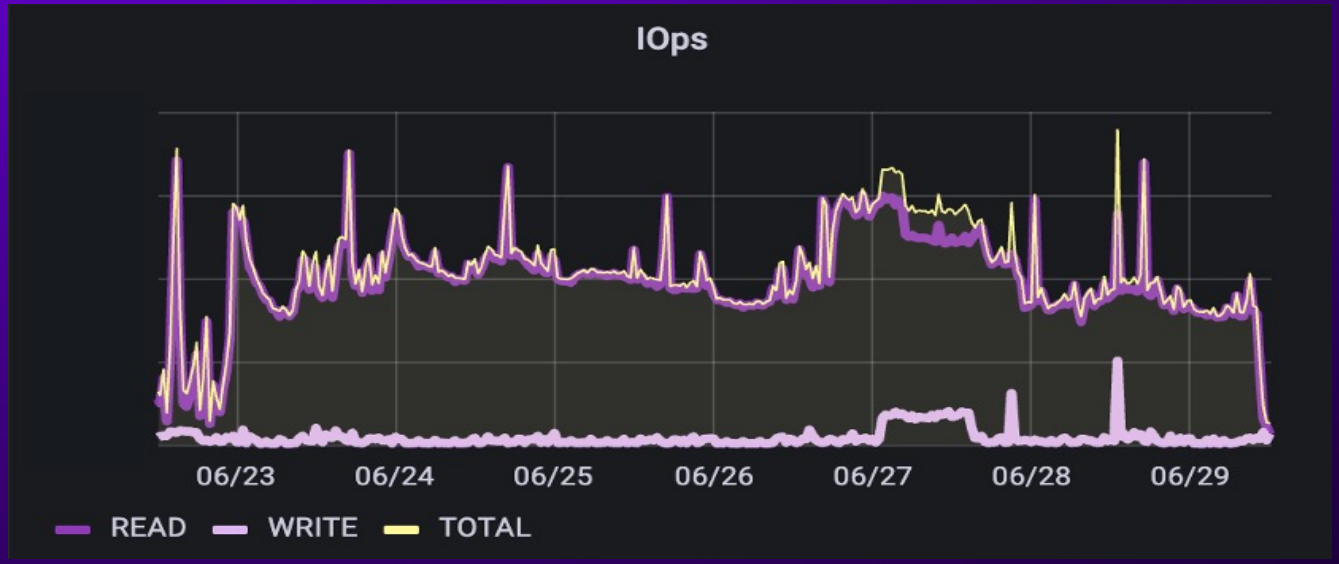
- Use case: audio-text and audio-action
  - Lots of natural language parsing
- Read/Write ratio of 47/53, but notice the IO sizes
- IO difference between reads and writes:
  - Writes are concatenating small samples and checkpointing
  - Reads are mostly streams of the larger concatenated files

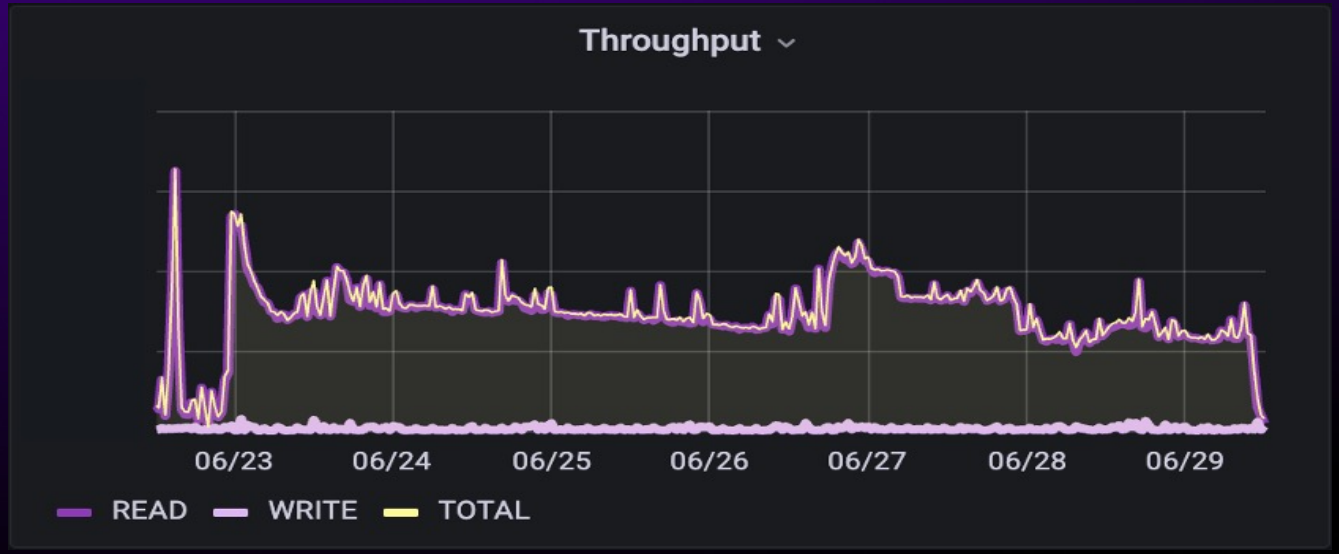


Data Profiles of Different AI Pipelines

AI/ML Model development

- Use case: text-text and text-action for business applications
  - Combination of NLP and ML
- Less overlap in mixed IO than just NLP. The workload is more segregated until the environment scales out further
- Higher reads, with relatively small writes other than some small burstiness.

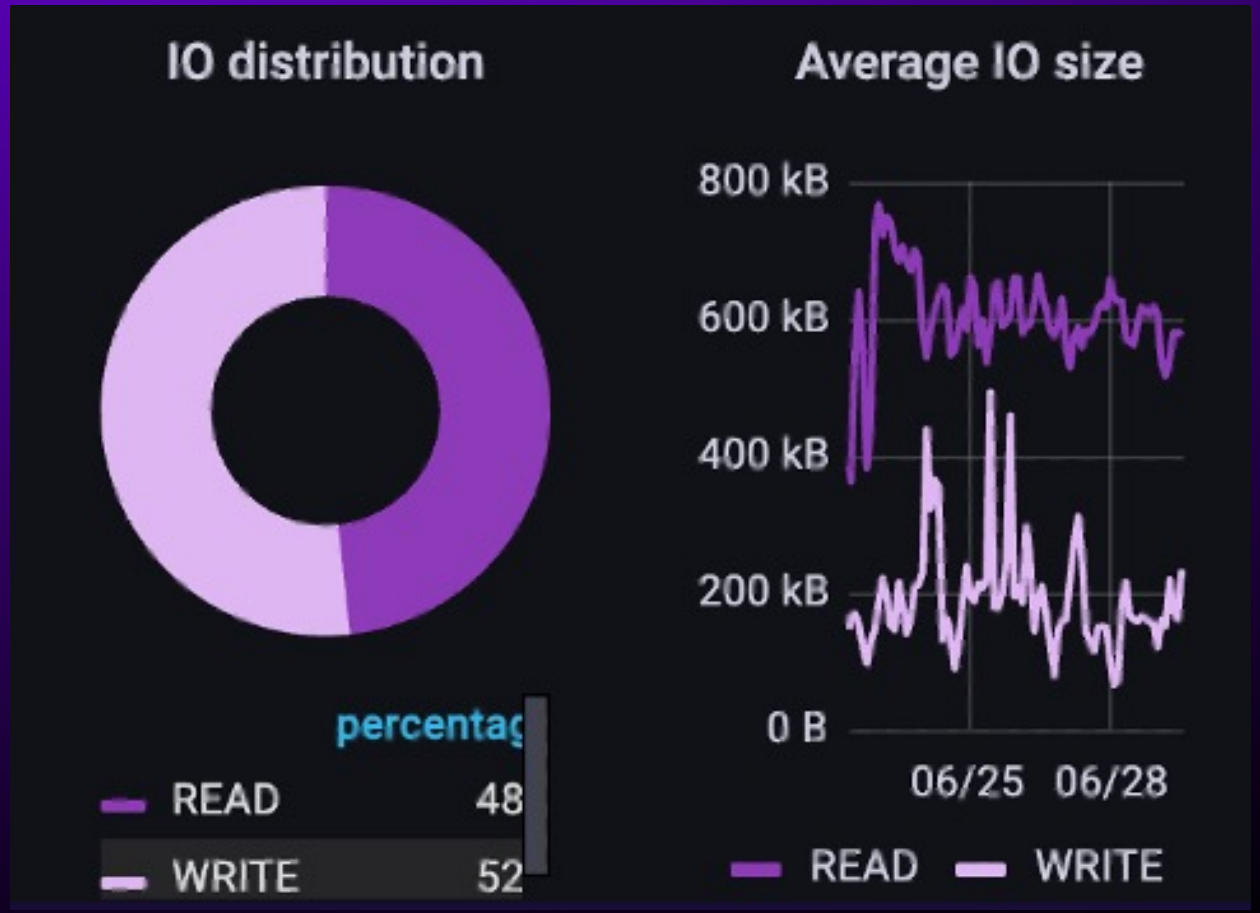




Data Profiles of Different AI Pipelines

AI/ML Model development

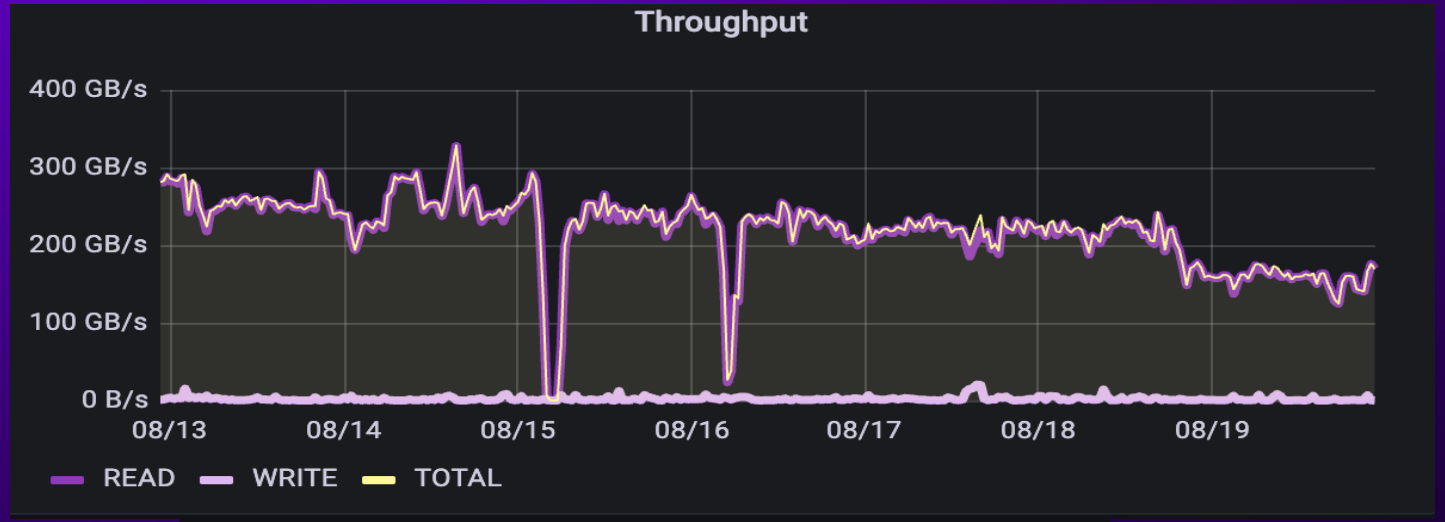
- Use case: text-text and text-action for business applications
  - Combination of NLP and ML
- Read/Write ratio of 48/52, with read sizes again much higher than writes.
- Read sizes in much narrower range:
  - Indicative of more pre-processing to a normalized data size
- Writes are concatenating small samples and checkpointing

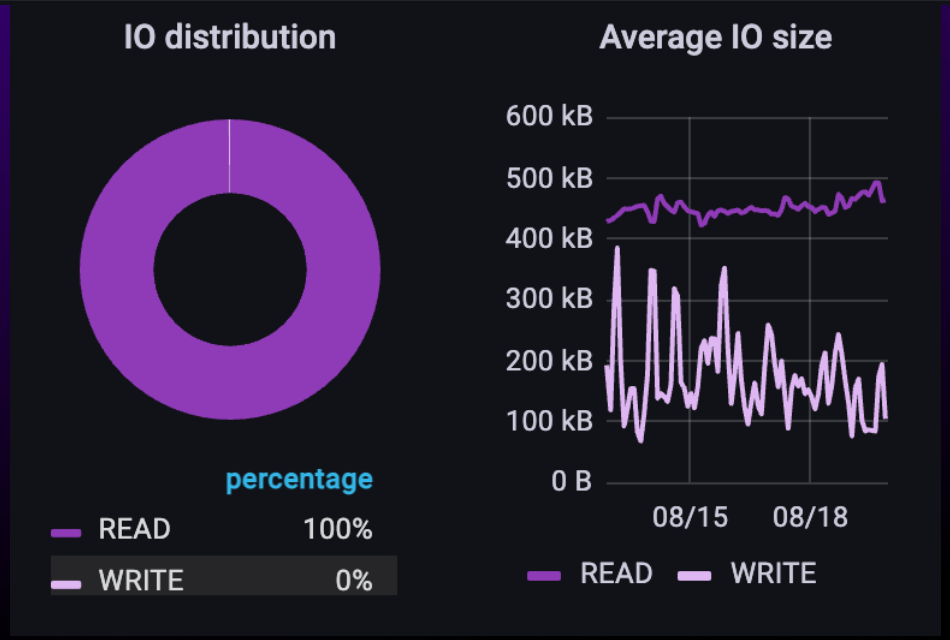


Data Profiles of Different AI Pipelines

Image Recognition/ Computer Vision

- Use case: Subset of an AI pipeline.
  - Automated training cycle of fully processed images
- Read IO in a very narrow, consistent range
  - Indicative of highly standardized individual items in the dataset
- Writes are metadata updates to files, writebacks of small process data and checkpointing.
  - Note that the chart is IO sizes. Metadata ops won't show up as IO for the most part.



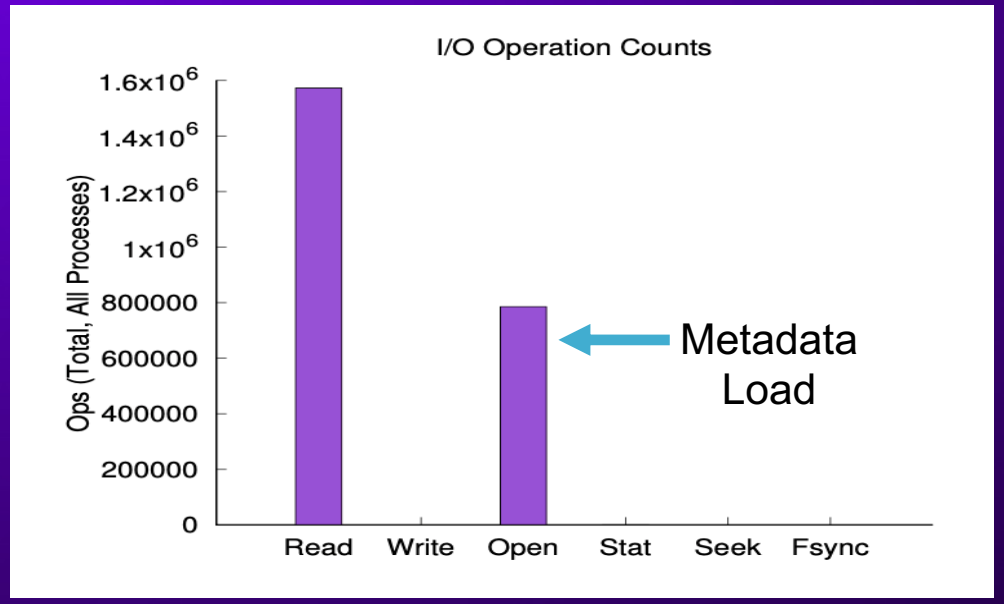


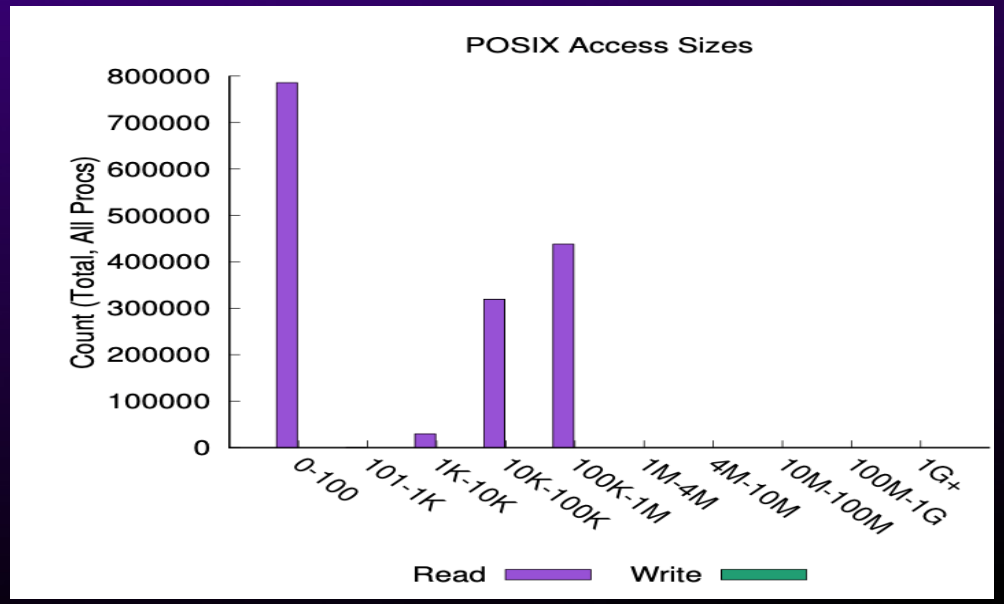
WEKA

Metadata Handling

Deep Learning IO Patterns

- Lots of random reads, lots of small files
- The Deep Learning IO Process:
  - Mini-batch and iterate over random subsets of the entire data set
  - Train on each mini-batch
  - Full epoch processing of the entire dataset, usually in a random order
  - Hyper-parameters control the process (e.g., precision needed, # of epochs, etc.)
- Deep Learning IO Characteristics:
  - Dominated by many small IO requests
  - **Huge metadata overhead**



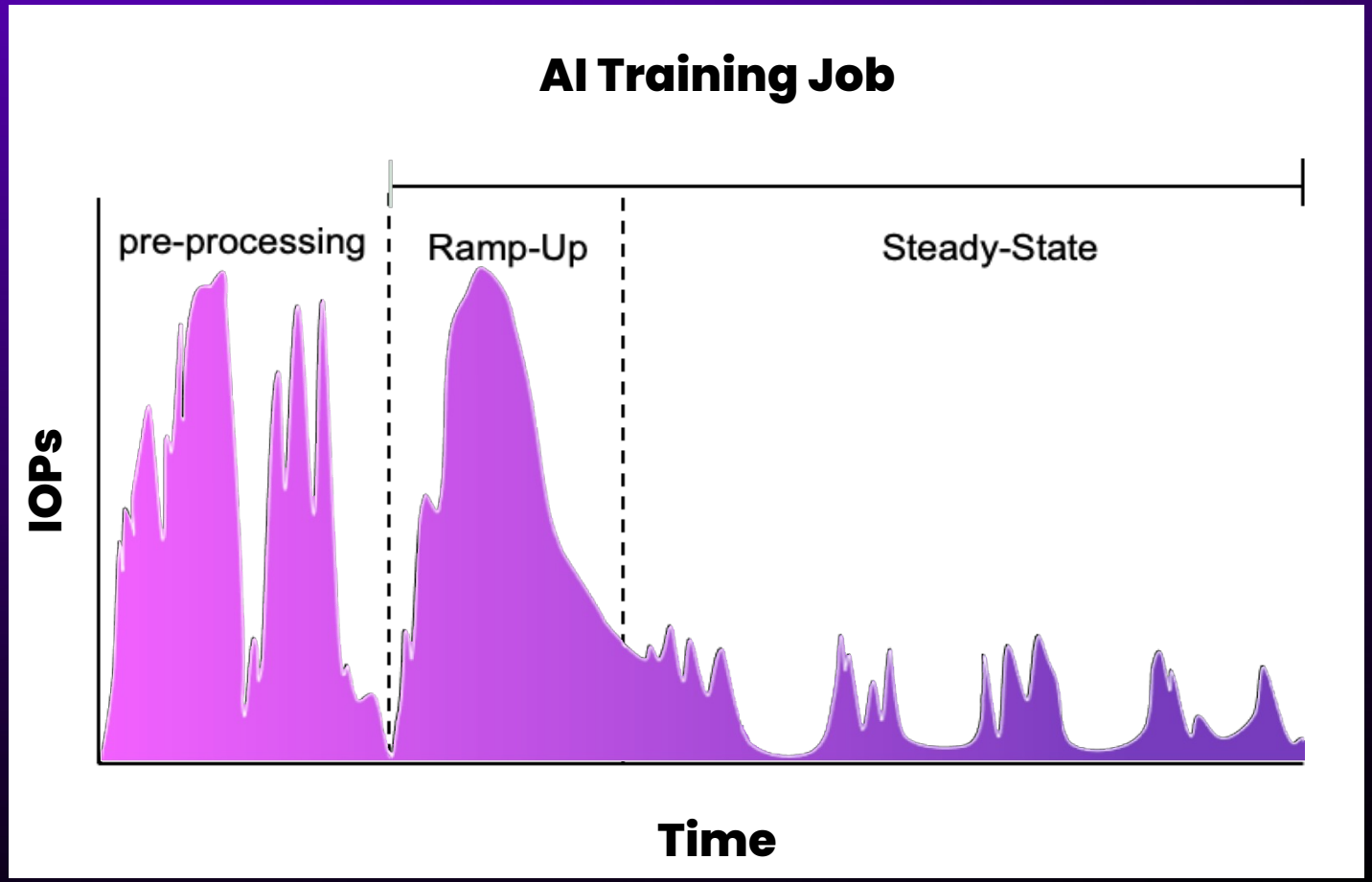


WEKA

Metadata Handling

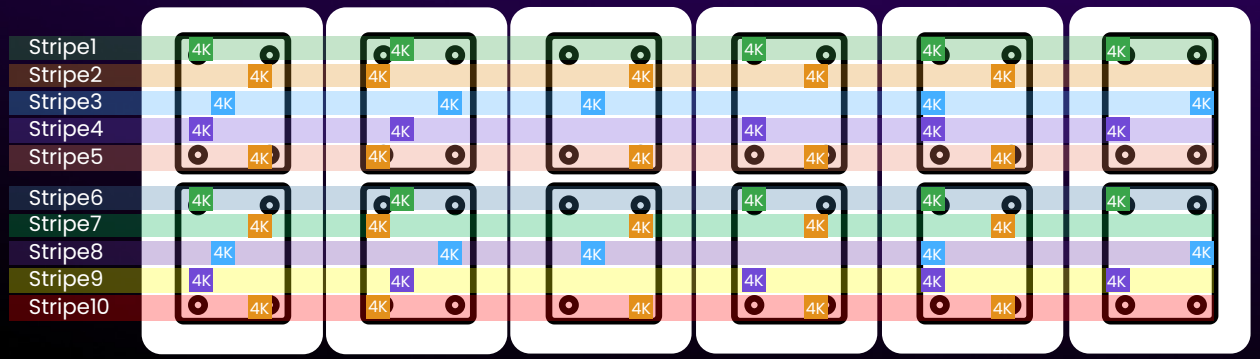
Pre-processing Implications at scale

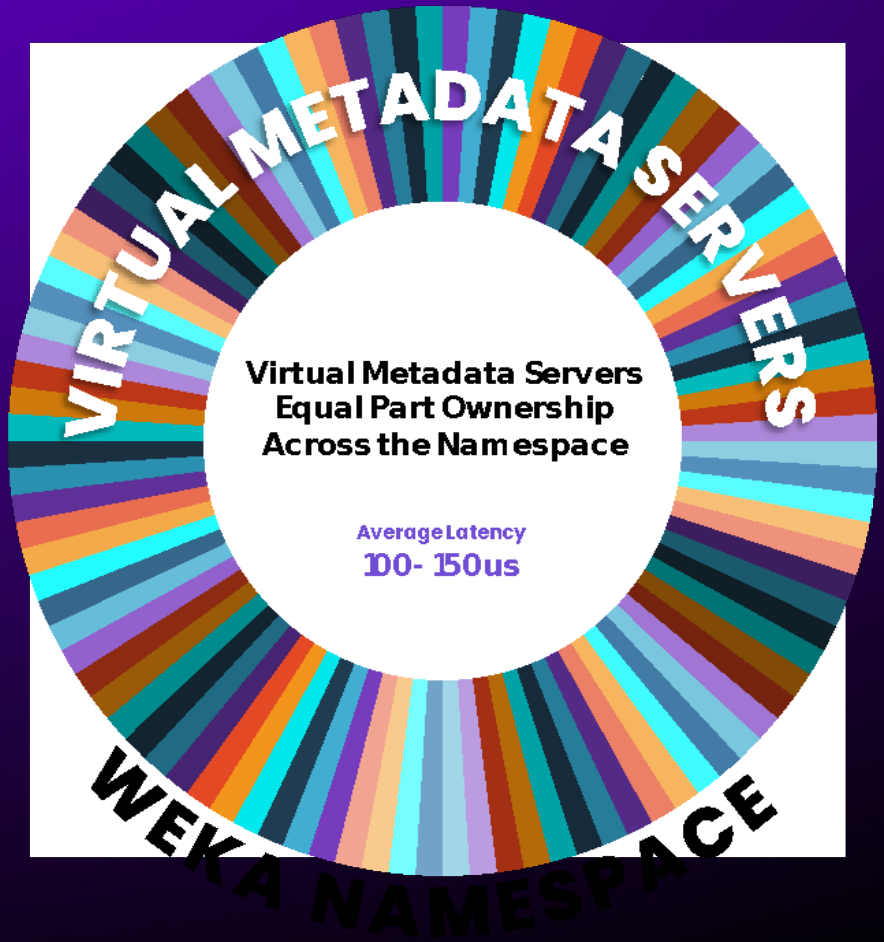
- Pre-Processing is manipulation of the accumulated data to a state that is expected by the AI Model (e.g., image resize, contrast changes, etc.)
- Within groups there are huge variations between the pre-processing steps each group performs - even within a group, every researcher might need to pre-process the data differently
- Pre-Processing can consist to 50% or more of the training epoch time
- **The IO Implications are massive. Millions of file operations, reads and writes with varying changes to raw accumulated data – WRITE CACHE MISS**



WEKA Data Distribution & Metadata Handling

- Every compute node runs some number of buckets - or virtual metadata servers (MDS)
- Each bucket is responsible for an equal shard (1/Nth) of Weka filesystem namespace
- Data fully distributed and written in 4K blocks across pool of NVMe SSDs - no ‘Hot Spots’
- Environment is aware of SSD responsiveness, prioritizing writes to SSDs w/lowest queue depth
- Every 8MB-range of every single file handled by a different virtual metadata server
- Virtual MDS use Weka’s block layout to persist the journaling of the metadata
- Buckets themselves can failover between compute nodes





WEKA

Checkpointing is still a challenge

- **Checkpoint – Periodic memory dump during a long running process in order to recover from potential outages**
  - Performed by thousands of CPU Cores / GPUs periodically, with overlap concurrency
  - Classic N:N and N:1 checkpointing are still a challenge

- **Trained AI Models still needs to be checkpointed !**

- **Checkpoint recovery is a huge issue**
  - Parallelism, Recovery Time Objective (RTO) needs are paramount

Source Text Placeholder

WEKA

Summary – Why WEKA with AI/ML Workloads?

Enterprise AI workloads have new DATA challenges that are unlike previous enterprise workloads

Ease of Use / Manageability

- NO Data engineering is required - Avoid data movement and data silos – Single Namespace
- Accelerate development – Improved Jupyter notebooks experience

Performance - Train /Re-tune/ Fine Tune / Inference many times faster

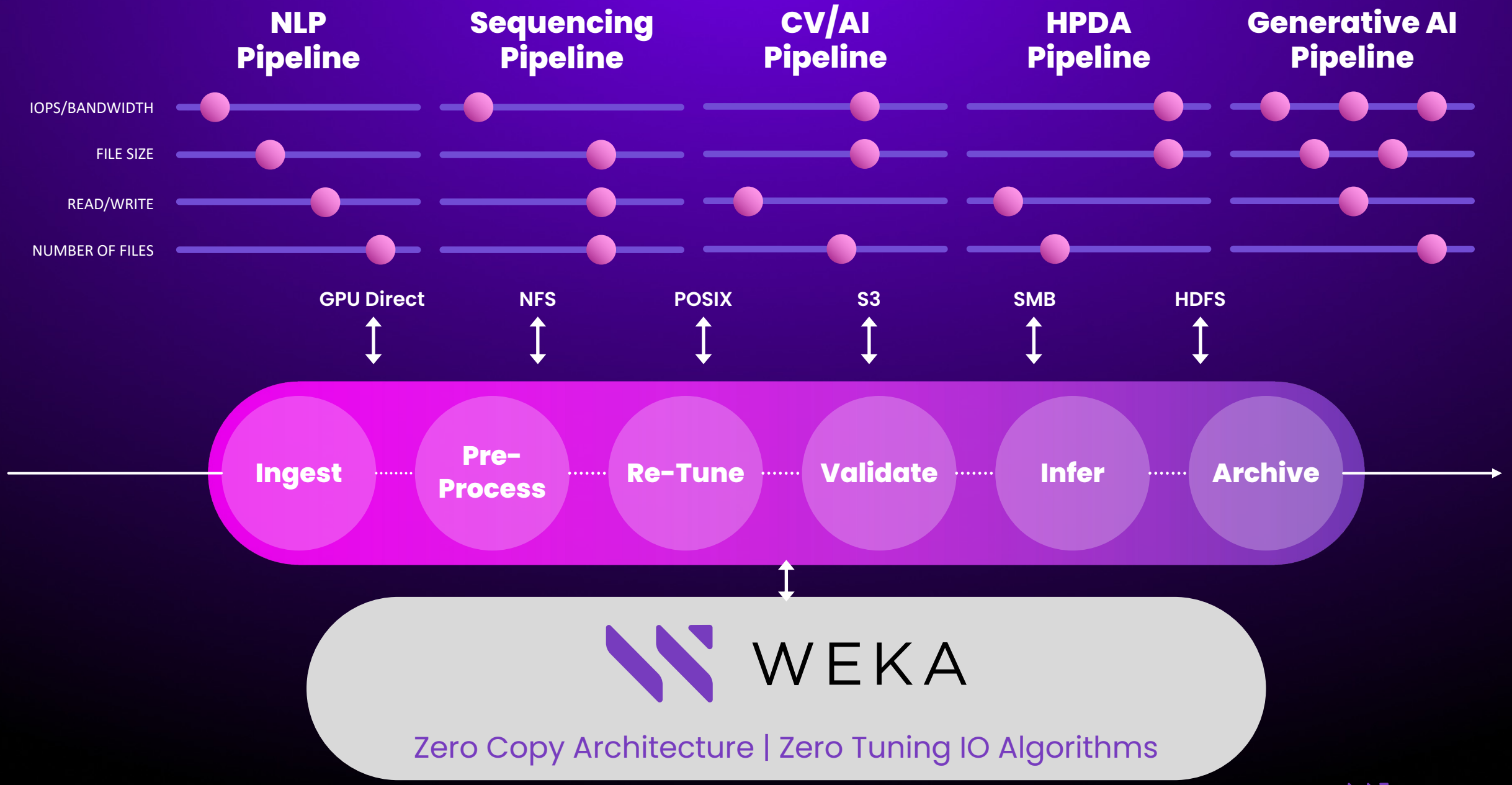
- Use of kernel bypass technologies and fast networking= ultra low latency transport for data AND metadata
- Completely distributed and parallelized architecture delivers consistent 100-200 *microsecond* latencies
- Low latency focus allows WEKA to handle high IOP, large streaming or mixed workflows without having to tune the system.
- Designed for multi-exabyte scale challenges. No worrying about where your data lives in the namespace as you scale.

Data retention and governance

- Save and archive datasets at petabyte/exabyte scale via snapshots and Snap-to-Object
  - Always know what model version was trained on what data set
  - Can audit the model training by showing the actual data set it was trained on: ”Explainable AI”. Helpful for governance

Data Mobility - integrate with MLOps environment

- Avoid data gravity (AT SCALE)
- Burst between Data centers / Clouds



Thank You!

 @wekaio

 /wekaio

 @wekaio

WEKA