WEKA

IO Profiles in Generative AI Pipelines

TECHNICAL BRIEF

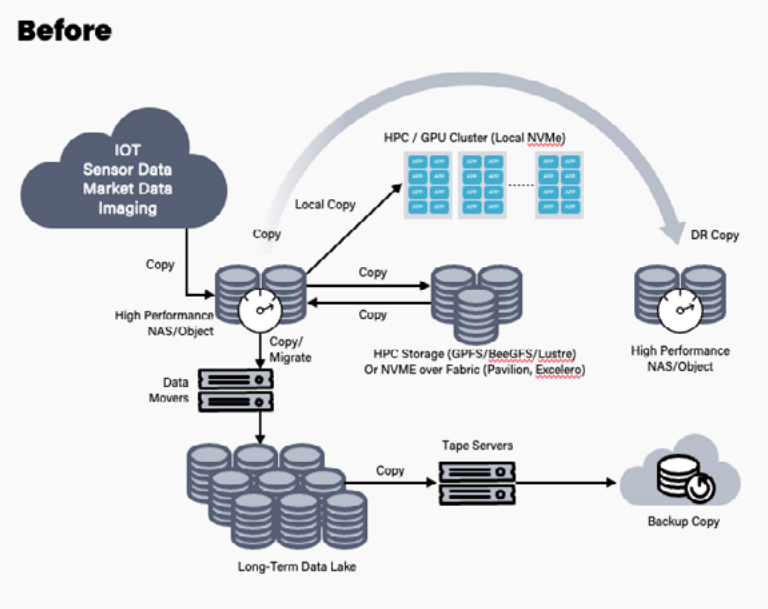
Executive
Summary

Generative AI (GenAI) has a dirty secret that is preventing many users from achieving their goals: IO profiles that can overwhelm traditional storage systems. The key to unlocking GenAI's full potential lies in making efficient use of high-quality, diverse training data with billions of parameters. The large amount of data to train models effectively, as well as acquiring, storing, and processing this data can be a challenge.

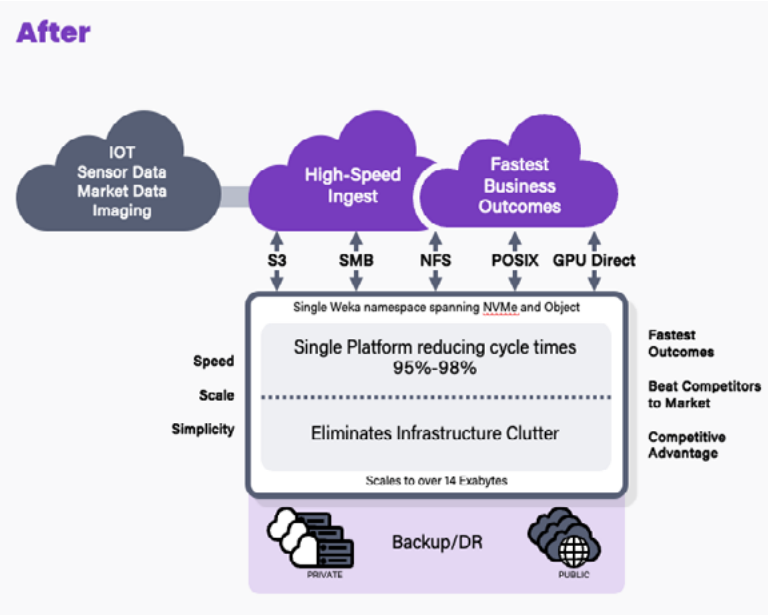
During a GenAI pipeline of operations, including ingestion of data, pre-processing, embedding, retuning, and then extensive validation/backtesting, IO patterns are widely varied and may lead to an IO blender issue, a situation where different workloads consisting of transactional and streaming IO contend with a sub-optimal performance level. The IO blender impacts time-to-results in generative AI by reducing the utilization of the compute GPU and CPU. WEKA has analyzed many customer environments to determine IO patterns and their impact on GenAI data pipelines.

Data Challenges

There are 2 distinct challenges in a GenAI pipeline whether on-prem or in the cloud. First is the concept of 'data stalls.' If we go back 10 years before many modern storage and filesystem solutions were available, customers would segregate their data based on performance capabilities. They would create silos for each type of IO in a workflow to optimize it for a particular step.

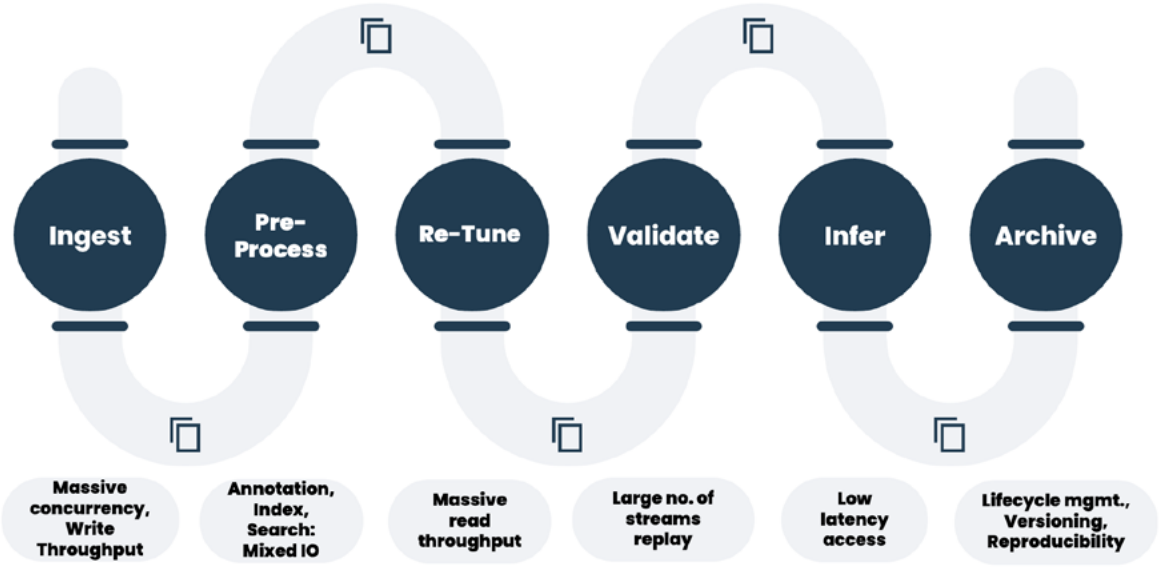


Today, however, customers are consolidating workflows due to resource availability pressures, where a single storage platform handles the entire data pipeline for ease of management, prevention of "silo sprawl", and overall cost-effectiveness. WEKA makes this consolidation a reality via the extremely high-performance capabilities of the WEKA® Data Platform.



Most GenAI pipelines involve processing on a blend of CPU and GPUs. Even if some of the pipeline may be better suited to CPUs, customers don't want to have to copy data back and forth from silo to silo as this creates a data stall problem resulting in low utilization of the GPUs. The preference is to consolidate the data on a single data platform. This creates a dichotomy; consolidate for copy efficiency, but then suffer the second challenge: The "IO blender".

While the link above talks about the IO blender in terms of Virtual Machine consolidation, this same concept applies to GenAI Data Pipelines.



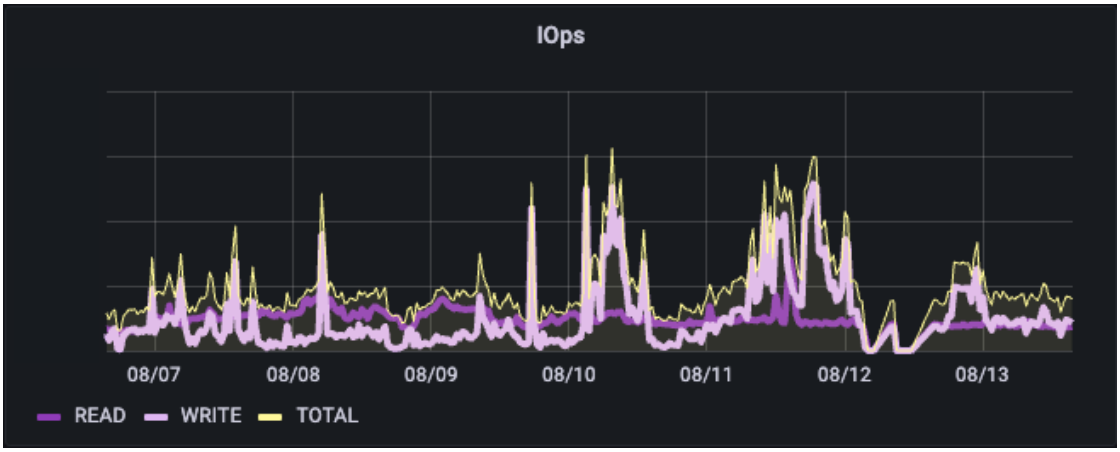
In a GenAI pipeline, you have multiple discrete stages of processing of data, with varying IO profiles. When consolidated, the IO creates contention with differing access types, concurrency, queue depth and latency requirements. In these environments you may have tens, hundreds or even thousands of CPU and GPU clients all operating against the same platform concurrently. This is a challenge that typical data platforms struggle with due to their architecture of focusing on only a few parameters of performance without considering the entire data pipeline. This focus on just a few characteristics leaves a lot of room for portions of the data pipeline to be underserved and slow down the time-to-value of the data.

Data Profiles of Different AI Pipelines

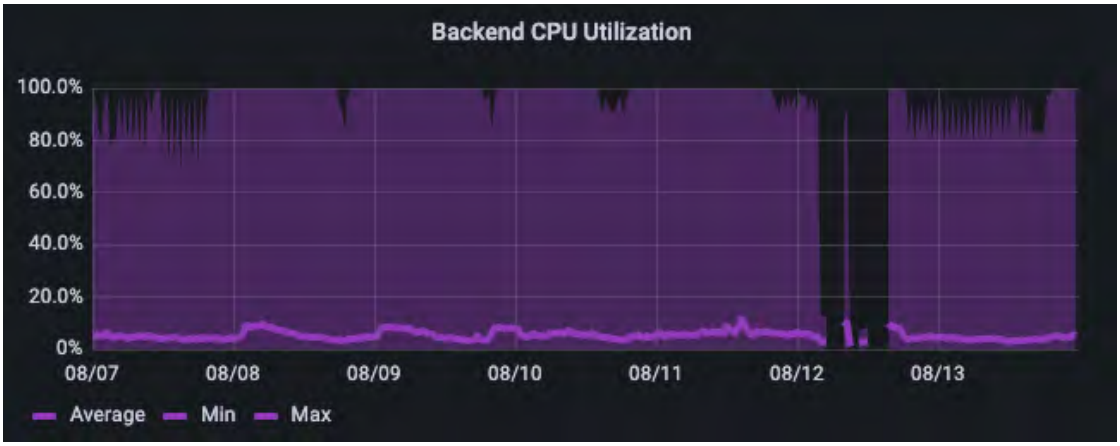
WEKA has done significant work to understand how the IO blender stresses the capabilities of many storage and data platforms. Below, we can see examples of the different IO from various stages in the following customers WEKA Home telemetry information. WEKA Home is a cloud-based phone-home support and analysis tool where customers can manage their WEKA environments.

Natural Language Processing (NLP)

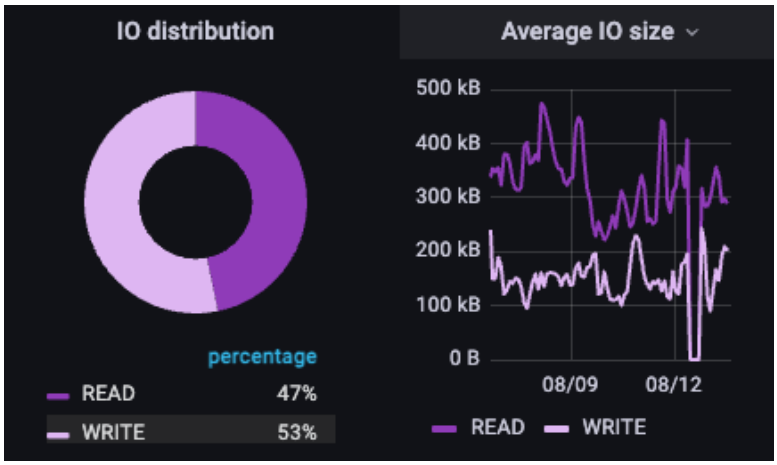
This customer use case is for building NLP models for audio-text and audio-action usage. In this IO profile, they are doing a significant amount of natural language parsing.



In this environment, we see that this is a fully blended mixed IO pipeline consisting of relatively consistent numbers of reads with spikes of IO on writes. This information can be mapped to how heavily loaded the WEKA backend servers are.



The deep purple and light purple as indicators of maximum utilization and average utilization show the intensely bursty nature of IO in AI pipelines. Seeing this level of detail on the pattern is critical. If we only looked at averages, we would think the system is underutilized at 5%. The reality is that it spikes up and down for intense bursts every few minutes, resulting in spikes that take it to 80-100% on a regular basis.

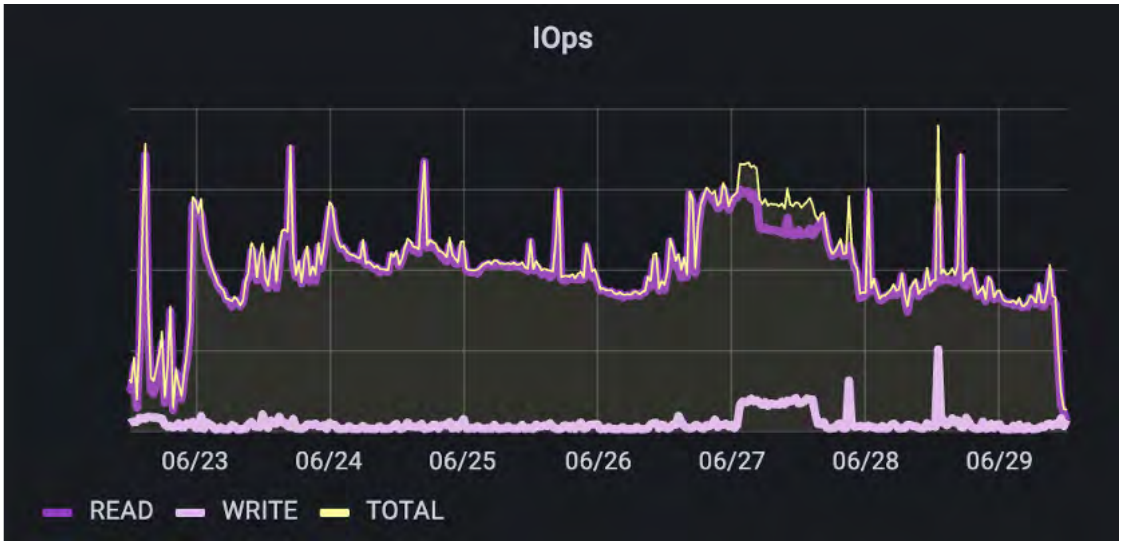


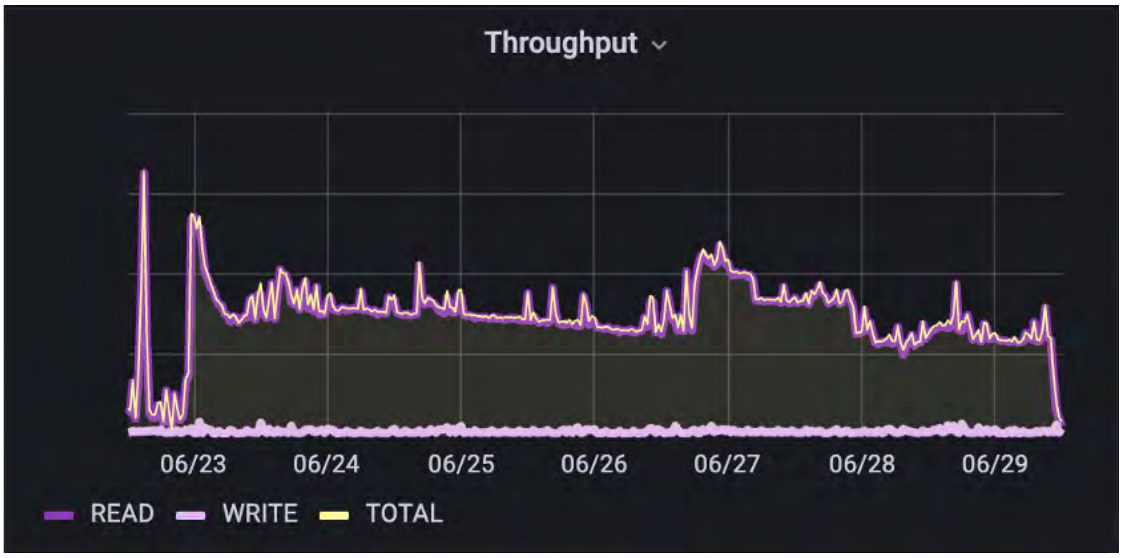
For this NLP pipeline read/write ratio is 47/53, but of note are the IO sizes: Large IOs are read in, but smaller IOs are written out. In this case, the customer has taken small samples that are generally sub-100k in size, concatenated them together into a larger file, and is reading against the file as if it were a stream. The smaller write IO is most likely checkpointing happening within the NLP workflow. This wide variance of data again challenges the data platform to tolerate multi-dimensional performance in this workflow.

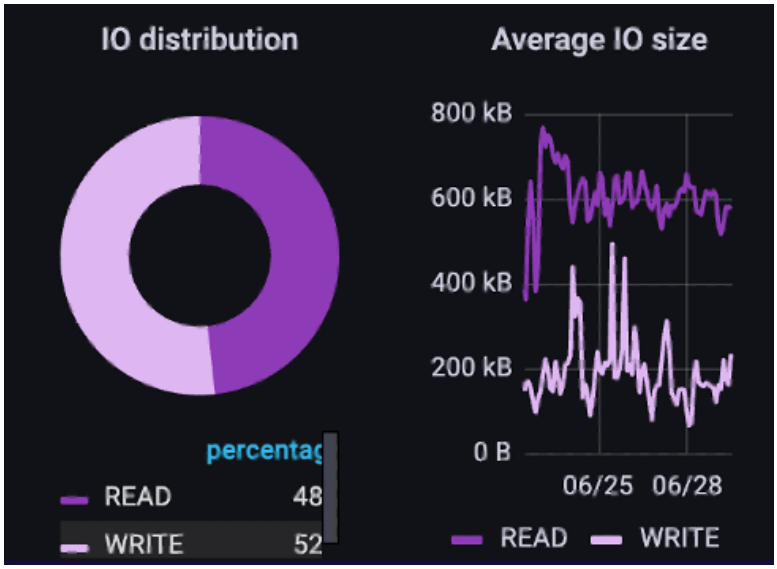
AI/ML Model Development

This customer use case is in building a combination of NLP and ML functions for text-text and text-action in business applications. Their data pipeline is more segregated with less overlap than the NLP use case. They have significant IO changes during their lifecycle of the development process. While still a production workload, they are in the process of ramping up to full-scale in the near future which will increase overlap of each portion of the data pipeline.

As we see here, they have a blended IO profile during ingestion and pre-processing phases. Both IOPs and throughput match up with a 48% read/ 52% write ratio The IO sizes again tend to be large reads with smaller writes.



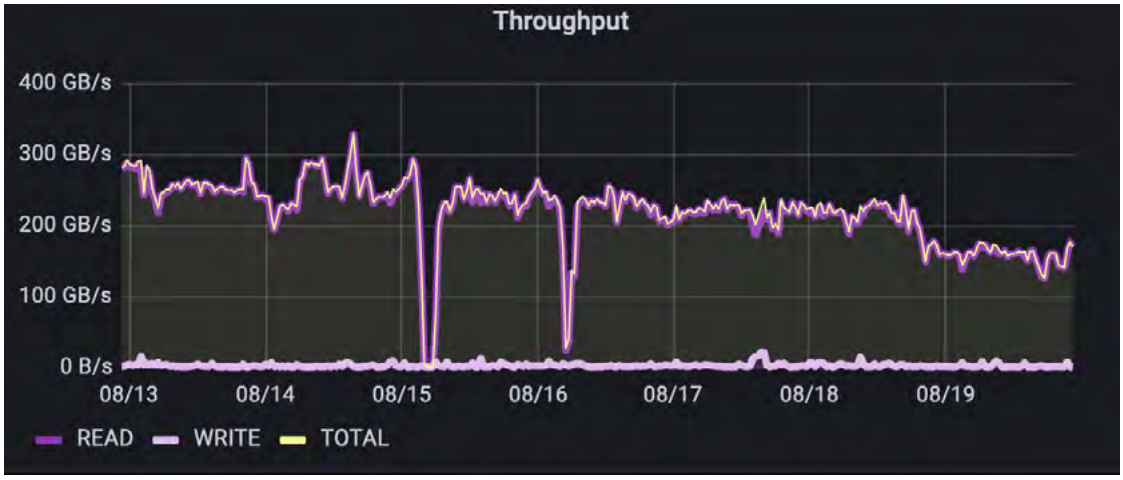


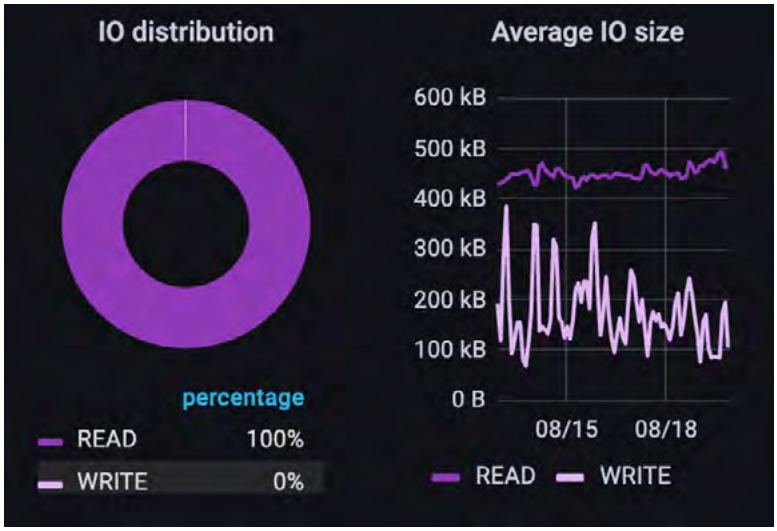


Note that there is less variance in the read IO sizes. The customer is using data that has undergone pre-processing and normalizing as they built their H5 files and tends to be of a more consistent size, thus reflecting the narrower range of read IO sizes.

Image Recognition/Computer Vision Training

The customer use case here is for a subset of the data pipeline. In this case they are doing almost exclusively an automated training cycle on ingested images where most of the pre-processing has already been done. This may be different than the metadata example shown earlier as this customer has not told us which tools (Tensorflow, Pytorch, etc.) they are using.

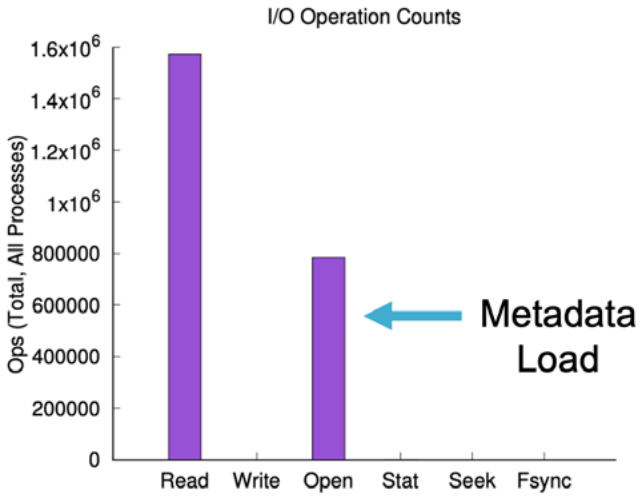




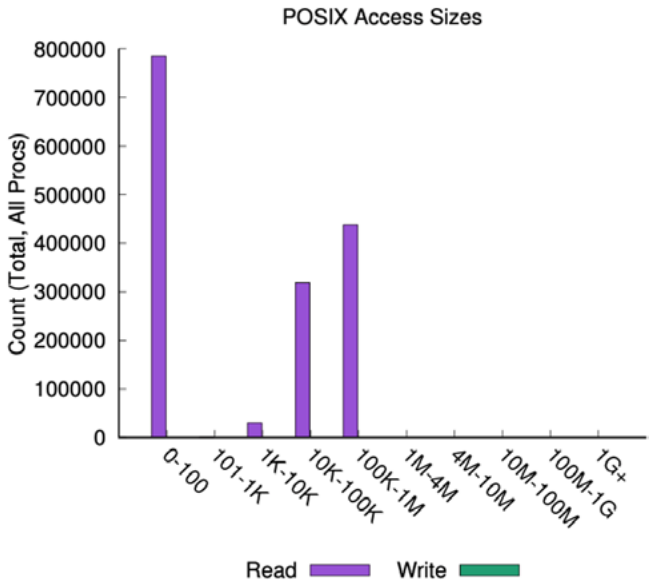
As expected, the overall throughput is dominated by the reads. While the IO size of the reads was very consistent in the 400k-500k range, the writes were widely varied. Most of the writes are either metadata updates to files, writebacks of small process data, and checkpointing. While the measured IO is for actual data transfer, there can be significant metadata operations that don't show up as IO. These Metadata operations can significantly impact the performance of a training model.

Metadata Handling

One area that is typically underestimated is the overall Metadata load when dealing with lots of small files, which is the dominant data style for many AI workloads. As an example of this metadata overhead, WEKA looked at a customer who was doing a deep learning pipeline using TensorFlow over Imagenet with a training database of 14 million images.



During this training, which is almost all reads, over 30% of the operations were file opens. This is a significant load that tests a data platform's ability to scale. WEKA has found that this overhead can range from 25-75% depending on the training data being used and the size of the training files. This type of metadata overhead is indicative of deep learning training and Resnet-50 type of loads with transfer learning. However, this is not exclusive to just training processes. For example, WEKA has seen environments where when looking at all the unstructured source data for ingestion, most of the IO was Stat and Seeks from finding the files needed to pull into pre-processing. This "Hidden IO" can create overhead on a data platform that gets worse as the workload scales up. Picture a 10 million read IOP environment where you need to process an extra 5 million metadata IOPs on top of that.



The size of the files matter as well. In the same TensorFlow pipeline, we see that the data sizes were either extremely small (sub-100 bytes) or mid-large sized at 10KB-1MB in size. Because of the variance in file sizes, the data platform needs to be capable of handling both very small random IO as well as somewhat more sequential IO at the same time.

Conclusions

GenAI and AI data pipelines in general have multiple challenges to overcome to effectively use the latest generation GPU and CPU at high utilization rates. The first challenge is data stall from having to copy data between siloed platforms that are designed and/or tuned to only meet the IO needs of a single portion of the AI data pipeline. The second challenge is being able to handle the IO blender that occurs when data is consolidated from solving the siloed platform problem. Within the IO blender, we see several ongoing trends that exist across all AI use cases:

- **Bursty IO at small time intervals.** As a data pipeline continues to be run with multiple portions of the pipeline overlapping this burstiness increases with varying IO ratios and IO sizes.
- **IO read/write ratios and IO sizes wildly vary.** In individual segments of the data pipeline, they may be relatively consistent, but in others, they may be quite different, including between reads and writes.
- **Latency is the leading indicator of overall performance within the GenAI environment.** Because most operations in the AI environment must be immediately consistent, (i.e., retuning the foundational model or creating and embedding features), having low latency for all operations makes the consistency happen faster, and the AI process can move to the next iteration quicker.
- **Metadata overhead.** Regardless of the IO profile or use case, Metadata lookups and overhead can account for 25-75% of the operations in a system. File opens, stats, seeks, etc. account for this.

All these trends point to the need for a data platform that is capable of handling the multi-dimensional performance requirements of various GenAI and AI environments. Many data platforms struggle with achieving this performance for several reasons such as:

- Trying to create low latency transports by layering RDMA with older file protocols like NFS. While it works, it still cannot achieve the ultra-low latencies that newer technologies such as DPDK with NVMeoF-like transports can achieve.
- Filesystem latency. If the filesystem itself is not designed for massive parallelization of both metadata and data operations, then it winds up serializing access, resulting in delays in responding to IO requests from clients.
- Tuning for individual workload profiles. Many platforms have made design choices resulting in optimizing for singular performance profiles. This results in having to copy data within the platform to specifically tuned sections, which creates data stalls.
- Inability to scale against the Lots Of Small Files (LOSF) problem.. With foundational models and other large learning models ingesting millions and sometimes billions of files of training data, many platforms struggle with handling this unstructured data. The workaround is to concatenate the data into a smaller number of large files, but this creates other access problems and can increase latency.

WEKA for Generative AI

The WEKA® Data Platform provides the fastest, most scalable file system for generative AI, delivering the performance developers expect and the scalability and simplicity the cloud promises - for all stages of model development, whether on-prem or on any cloud. Over 200 leading AI deployments leverage WEKA for enhanced performance and cost efficiency including Stability AI, Midjourney, and Adept.AI.

WEKA solves the issues that other platforms have with handling consolidated GenAI workflows:

- WEKA's use of kernel bypass technologies such as DPDK and SPDK along with fast networking up to 400Gbit. creates a ultra-low latency transport layer similar to NVMeoF for data and metadata.
- By having a completely distributed and parallelized architecture that looks to achieve the lowest queue depths for any operation in the system, WEKA can handle all types of workloads concurrently while maintaining consistent 100-200 *microsecond* latencies.
- This same architecture and focus on low latency allows WEKA to handle high IOP, large streaming, or mixed workflows with aplomb. This allows for consolidation without performance tradeoffs, or having to constantly tune sections of the system to optimize for a single part of a workflow.
- WEKA's architecture was designed upfront for multi-exabyte capacity scaling challenges. This translates into not having to worry about where your data lives in the WEKA platform. Billions of files in a single directory and trillions of files in a single namespace? With WEKA you aren't forced to concatenate files to make up for limits in the filesystem, simplifying your workflow and number of operations to make use of your data. WEKA easily handles the LOSF problem.

Deliver the Performance Your Developers and Researchers Need

Whether you're developing next-generation assistants, virtual reality applications, video games, or advertising campaigns, WEKA can accelerate and simplify the use of training data in a variety of formats, including images, video, text, and audio. WEKA delivers unbeatable file storage performance at any scale for your most demanding model development and training supporting high I/O, low latency, small files, and mixed workloads with zero tuning. The WEKA Data Platform simplifies and accelerate your data infrastructure so you can focus on creating models for more realistic and engaging content.

WKA368-01 08/2023