AI Inferencing Storage IO Traffic Profiling and Analysis

Xiangyu Tang, Roy Leonard, Craig Lucero, Veera Saripalli, Chaman Saurav

micron

Agenda

1. AI Inferencing Steps and Performance Measurement in PCs
2. AI Inferencing Traffic
   a) Model Loading in AI Benchmarks
   b) Field Usage: Multi-model, Multi-modal, RAG
3. Uniqueness of AI Inferencing Traffic

AI Inferencing for PCs

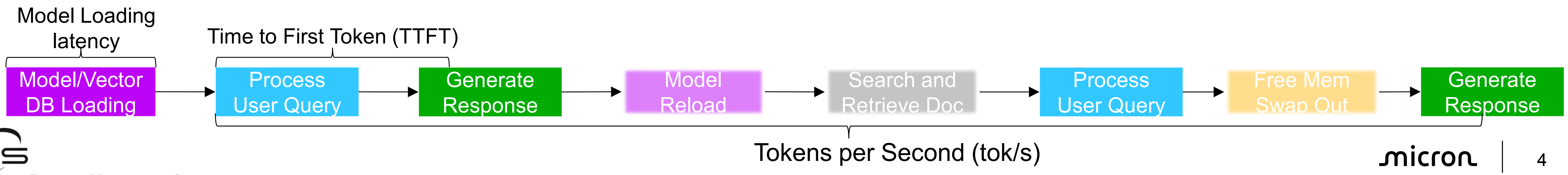
micron

FMS

Inferencing on PCs

- AI Inferencing involves 3 main steps: loading the AI model, process user query, generate response
- Inferencing performance for LLM is measured using 3 main criteria
  1. AI model loading time: loading model weights to GPU
  2. Time to first token (TTFT): processing user query and prefill/initialize kv cache
  3. Tokens per second (Tok/s): measures inferencing performance after the first token
- Model loading time is dependent upon
  - SSD performance
  - AI framework (software stack, processor, etc)
  - Model (size, type, gen, etc)

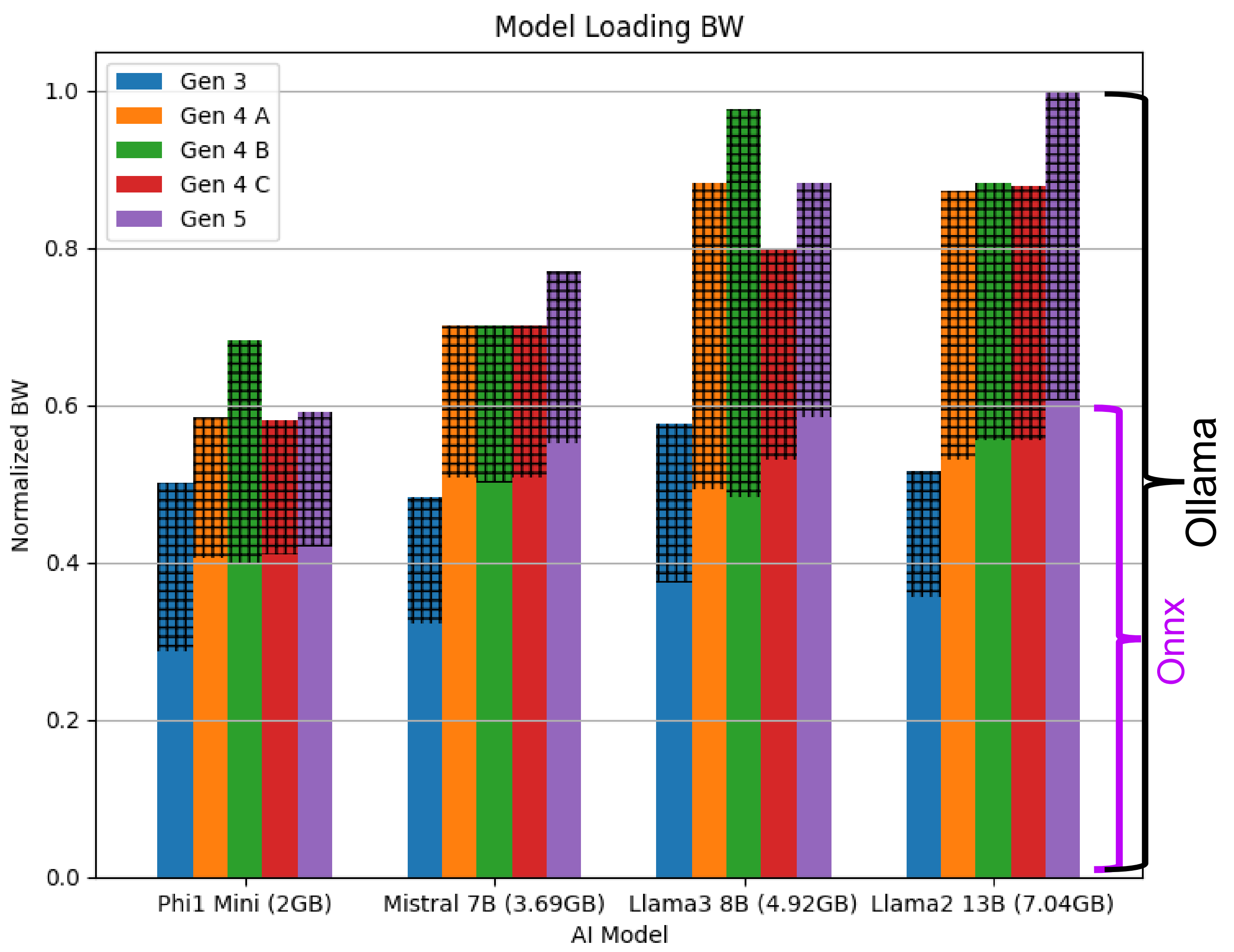
- TTFT is dependent on GPU processing power, since prefilling/initializing the kv matrix is compute intensive
- Tok/s is memory IO bound, dominated by kv cache read
- In many cases TTFT and Tok/s are also SSD performance bound
  - In multi-model scenarios, AI models in background could be discarded from memory to free space, and needs to be reloaded when user query arrives
  - In RAG scenarios, index searching for user query involves loading relevant index from disk to memory; data chunk retrieval needs to read data from disk
  - As KV matrix grows, may need to swap to disk



AI Inferencing Traffic Patterns and IO BW

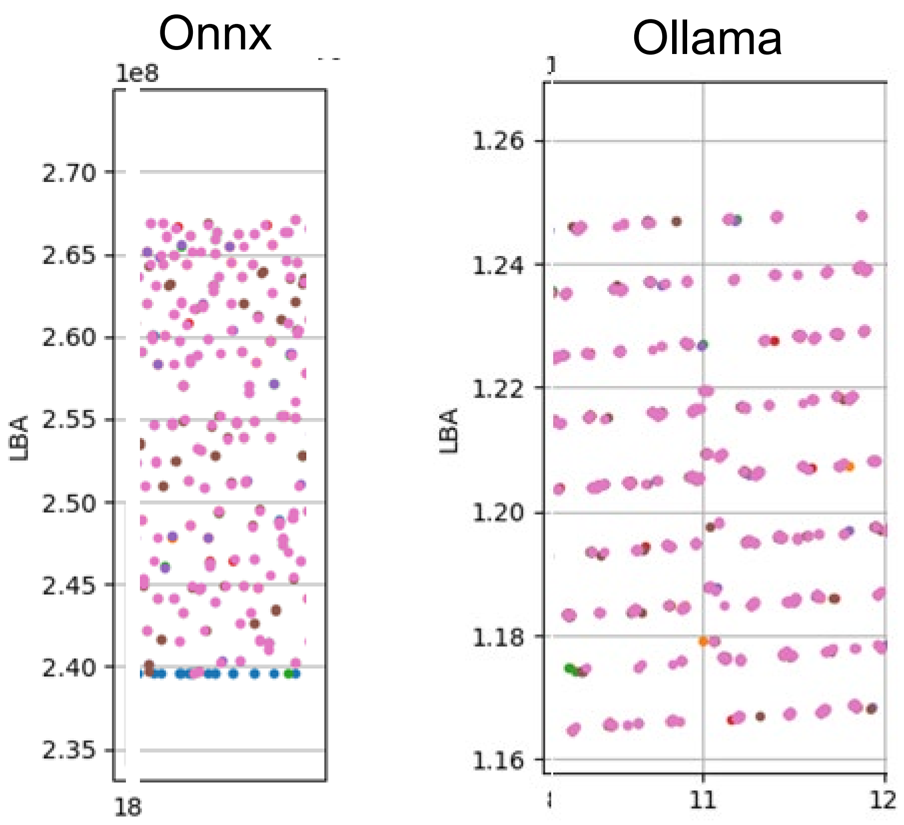
Model load time Disk Read BW

- Model loading BW is affected by
  1. SSD BW and optimizations
  2. AI model
  3. Run time framework



Llama and Ollama Model Loading Patterns

Disk Read Pattern



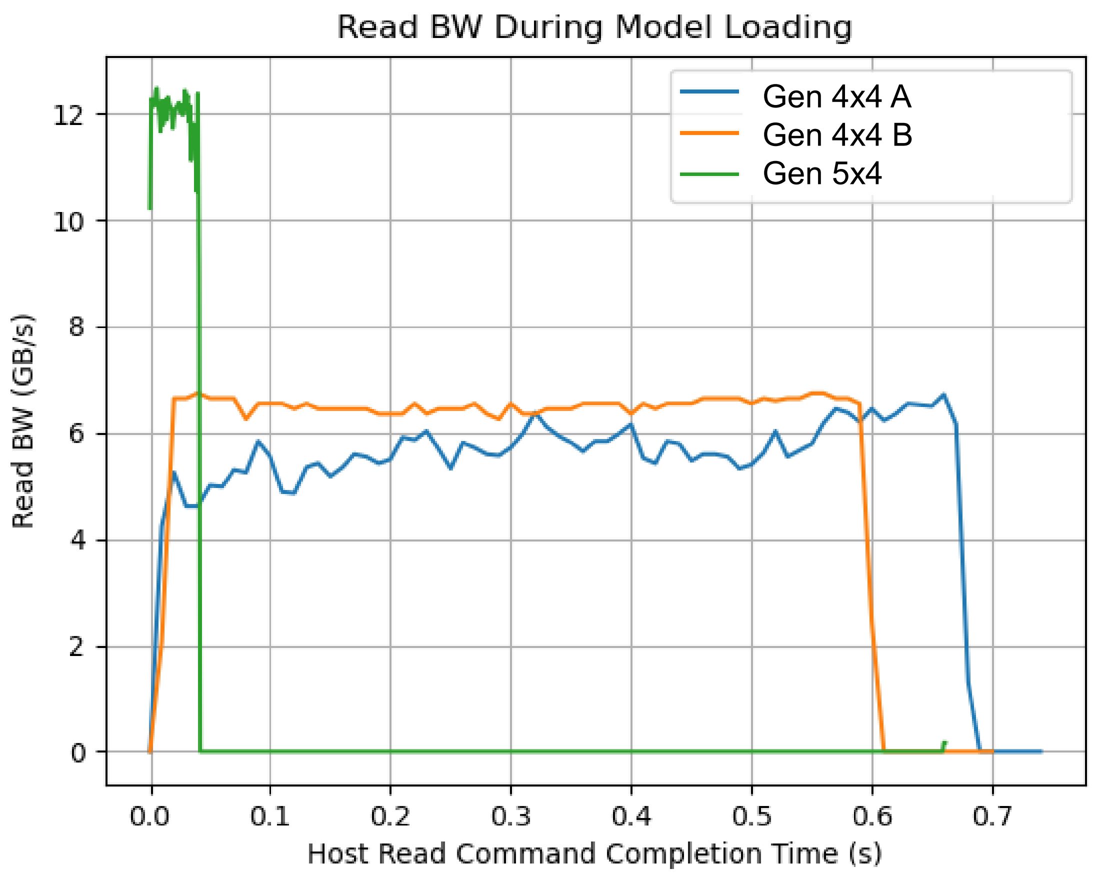
- Onnx run time (ORT) is less optimized than Ollama when loading models
  - Onnx model loading exhibits **random** read patterns, instead of Ollama's **multi-stream sequential** read patterns

– Versions of Ollama can saturate NVMe BW when loading models

- Latest Onnx RT shows improvement in generating sequential read patterns and higher BW during model loading

Mistrallite Model Loading with Ollama

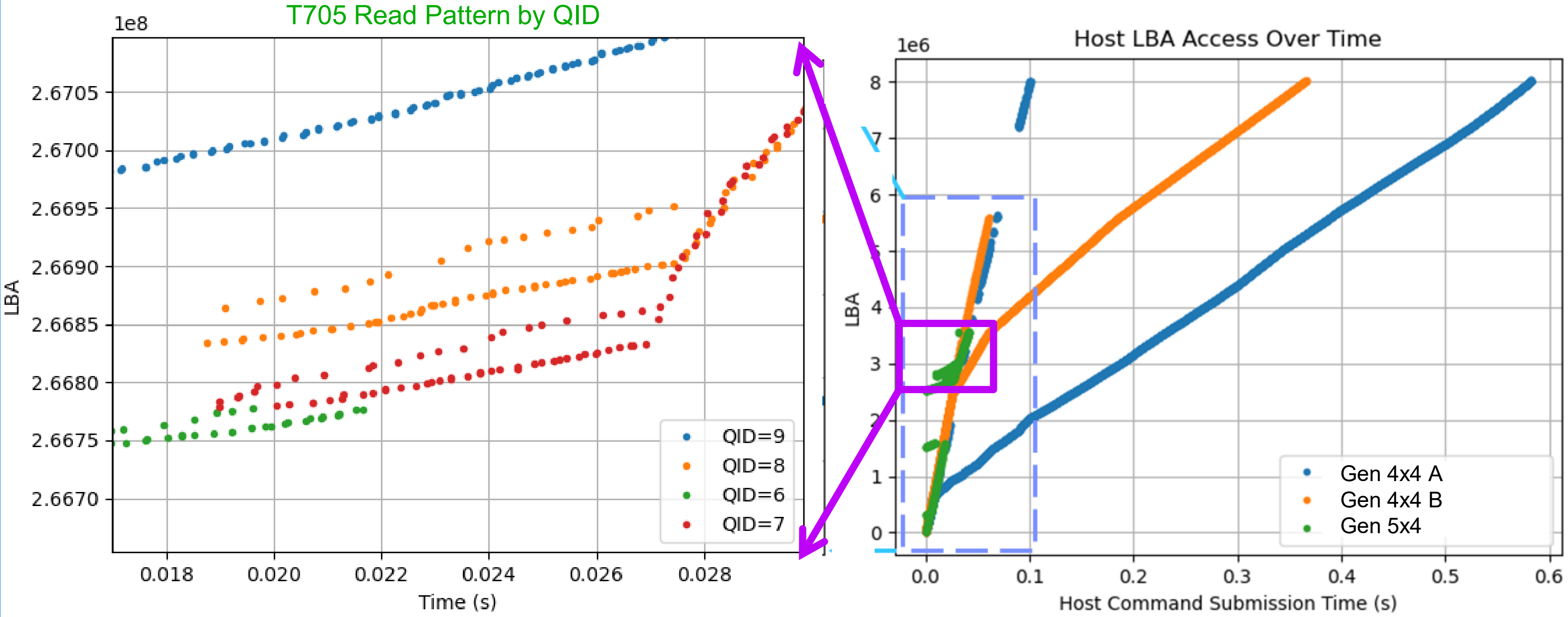
- Mistrallite NN model loaded with Ollama (v0.1.10)
  - About ~3.9GB is read
- Disk read saturates PCIe BW
- QD is high
- Read payload at MDTS
- Multiple stream sequential reads



Mistrallite NN Model

Read Sequence

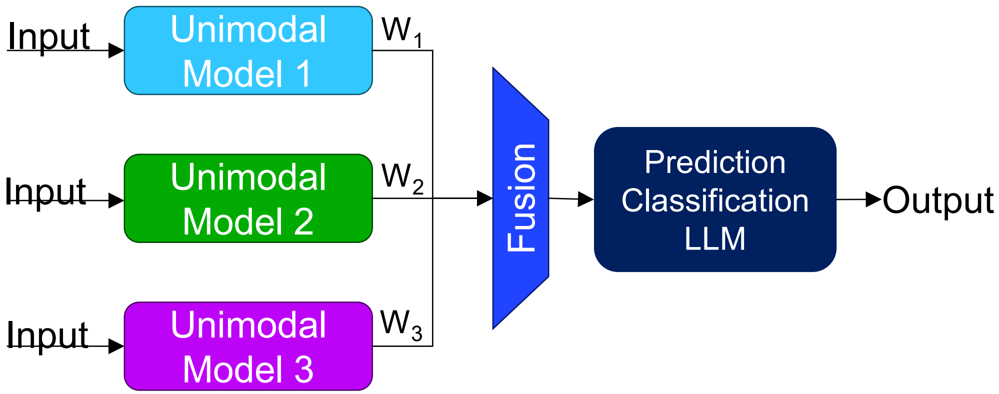
- Between 2 to 5 streams are used to read the entire file in a mostly sequential fashion
- Zooming in on Gen5 drives shows that one queue ID corresponds to one sequential stream



Multi-Modal NN

Model Architecture

- Multi-modal NN can have either multiple types of inputs, or multiple types of outputs, or both
- Tested Llava model which is a multi-modal NN with multiple types of input (image and text) and a text output
  - Combines a vision model (VTi) with Vicuna LLM (13B, 8GB) for general purpose visual and language understanding

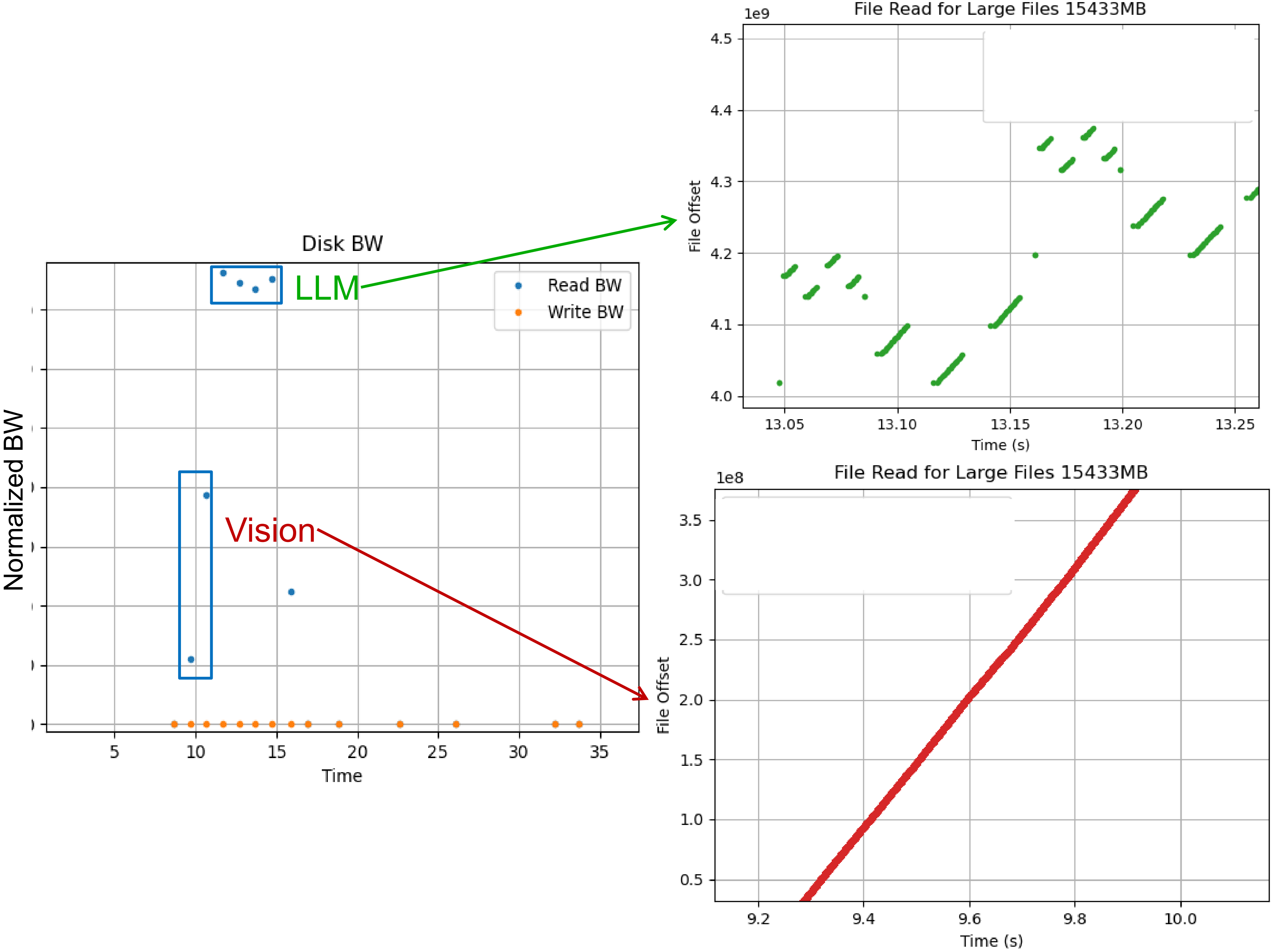


Query: What is the ending of this movie

Multi-Modal NN

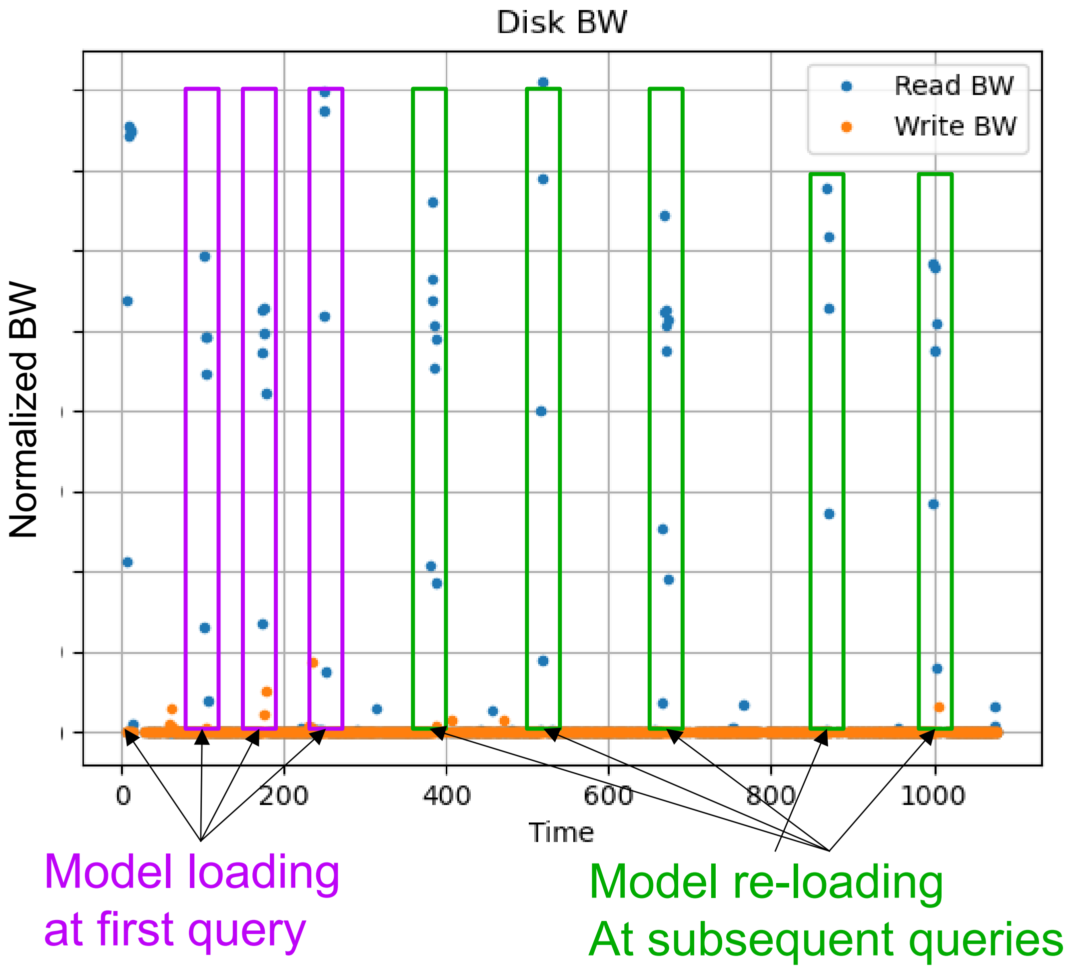
Llava Loading on Ollama

- Loading pattern is model dependent
  - Vision model: single stream sequential read, smaller payload size
  - LLM model: higher BW/QD, multiple stream sequential read, large payload size (MDTS)

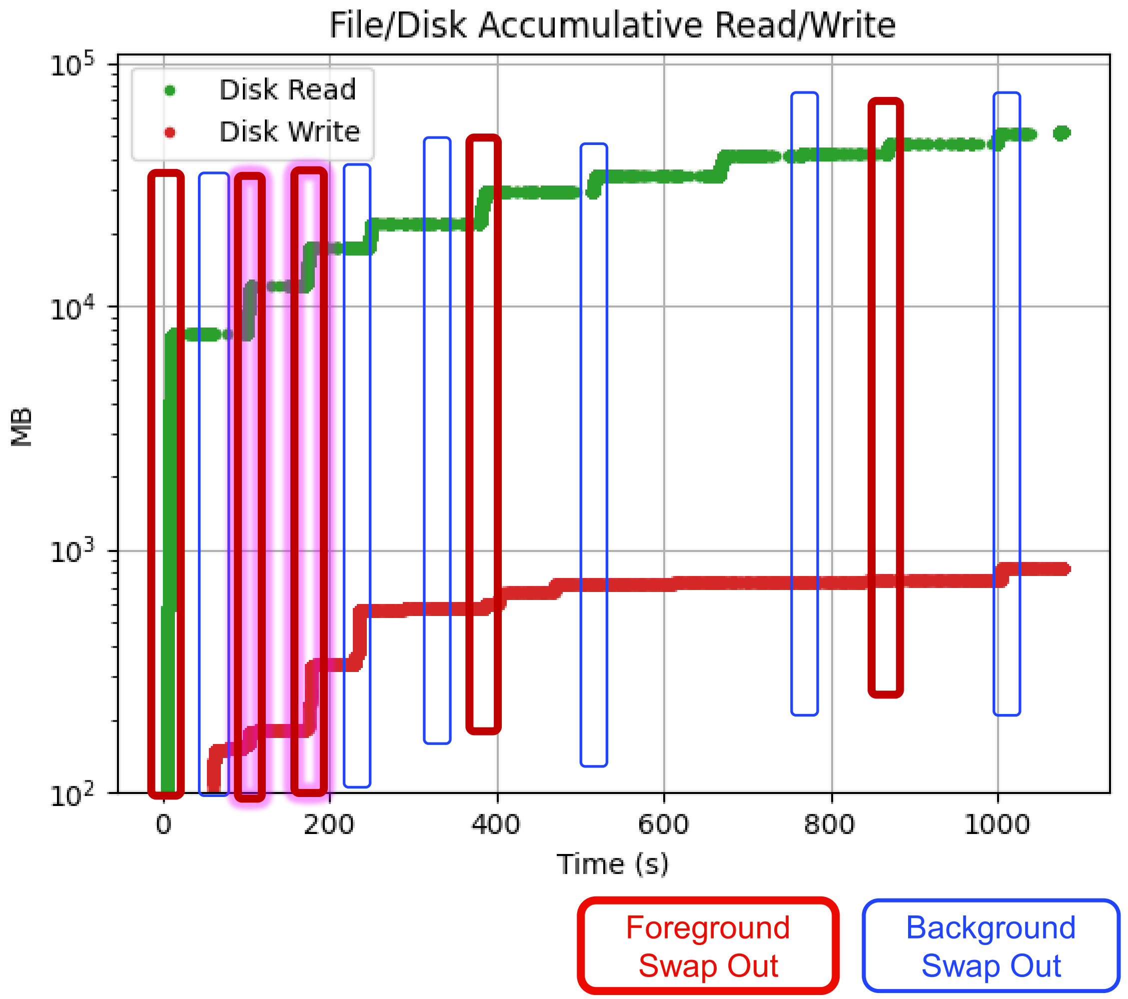


Multi-Model NN

- Multiple models (4) running concurrently in the same Ollama framework
  - Llava 13B(8GB), qwen2.5(4.7GB), gemma2(5.4GB), llama3.1(4.7B)
- Combined model size greater than physical memory capacity
  - Query only a single model at a time, rotate which model is queried
- Model weights and active user data are periodically swapped in/out, making SSD BW a gating factor
  - When memory is full, memory is freed by *discarding* existing models in memory, *not* by swapping out
  - When a model is needed again, it is reloaded from disk
  - Only ~500MB of live user data (not models) are swapped out to disk into virtual memory (pagefile.sys), 50% in foreground

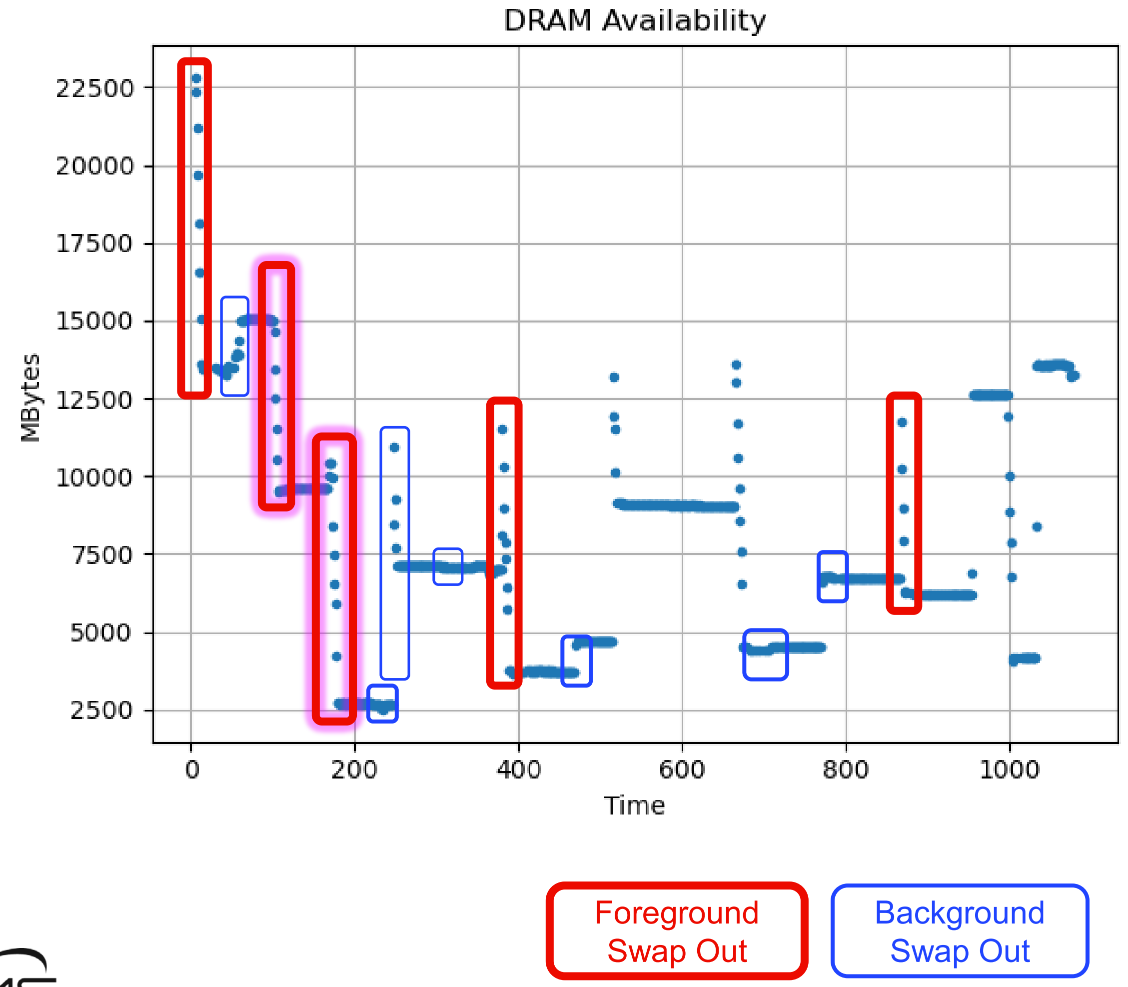


Multi-Model NN



- Models are reloaded from disk during user query in multi-model scenario
  - This gates query response time
- A significant portion of swap-outs occur during model reload (250MB out of 500MB)
  - SSD write BW gates model reload and thus query response time
- Disk swap outs are large sequential writes
- Sequential reads and mixed read/writes can impact Tokens-per-Second

Physical Memory Availability



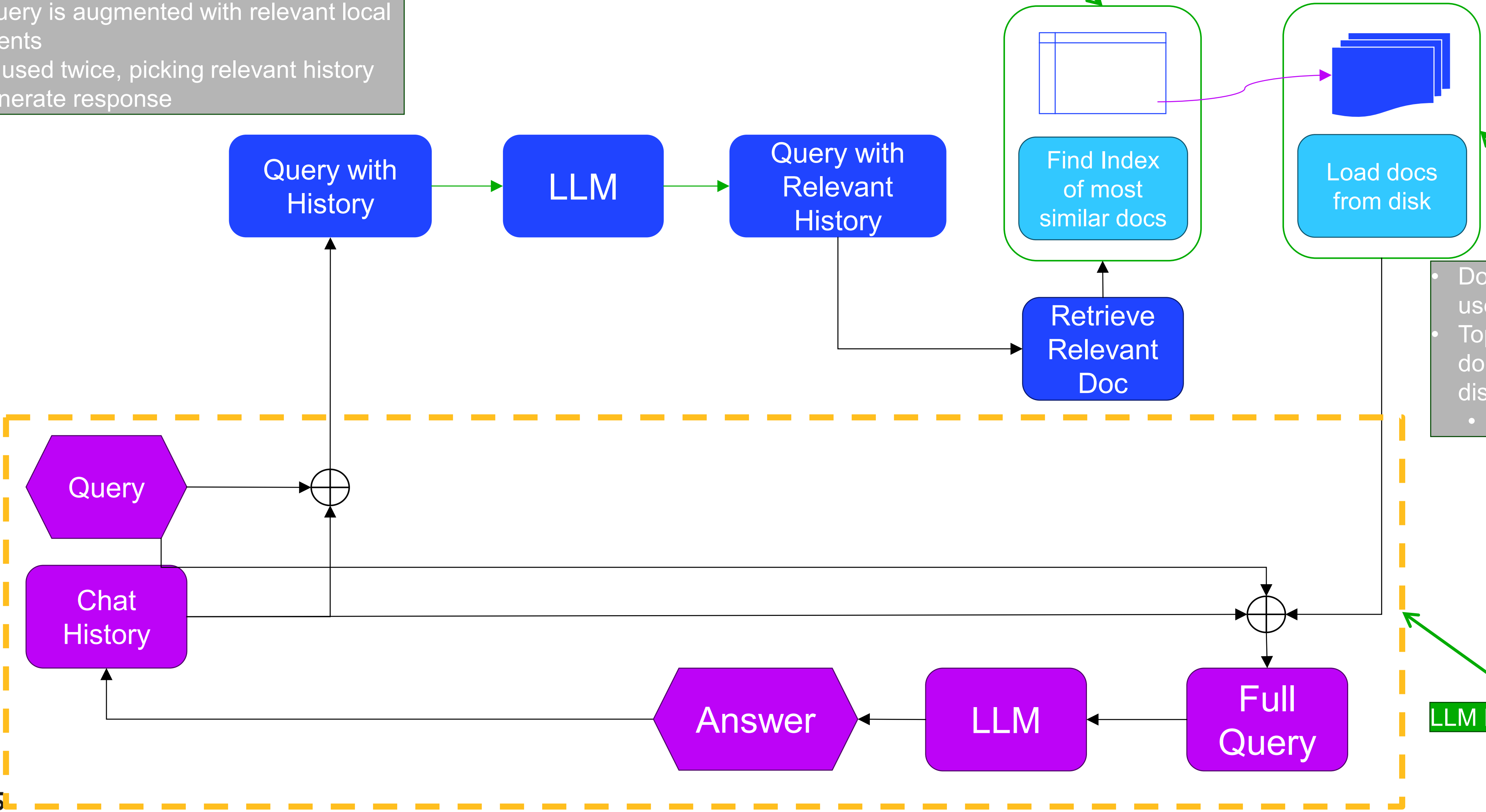
- Majority of memory reclamation comes from discards
- Swap-out writes to virtual memory is an order of magnitude smaller than reading from disk
  - 500MB total
- Conjecture:
  - Live user data from other non-AI apps are swapped out to disk, in order to free memory to load models
  - Models are simply discarded from memory if more space is needed
  - KV cache will eventually be swapped out to disk

Composite

RAG

- RAG used to optimize LLM with private and updated user data
  - Eg, Itinerary, Perfect Shot, Recall
- User query is augmented with relevant local documents
- LLM is used twice, picking relevant history and generate response

- Vector DB (>7x expansion) is multi-level and is too large to be stored entirely in memory, split into level-0 and SQL and stored on disk
  - Level-0 loaded along with LLM from disk into memory (sequential read, low QD)
  - Similarity search loads SQL DB OTF from disk for every “cold” query, gates query latency
    - Sub-512B of index generates 4kB payload



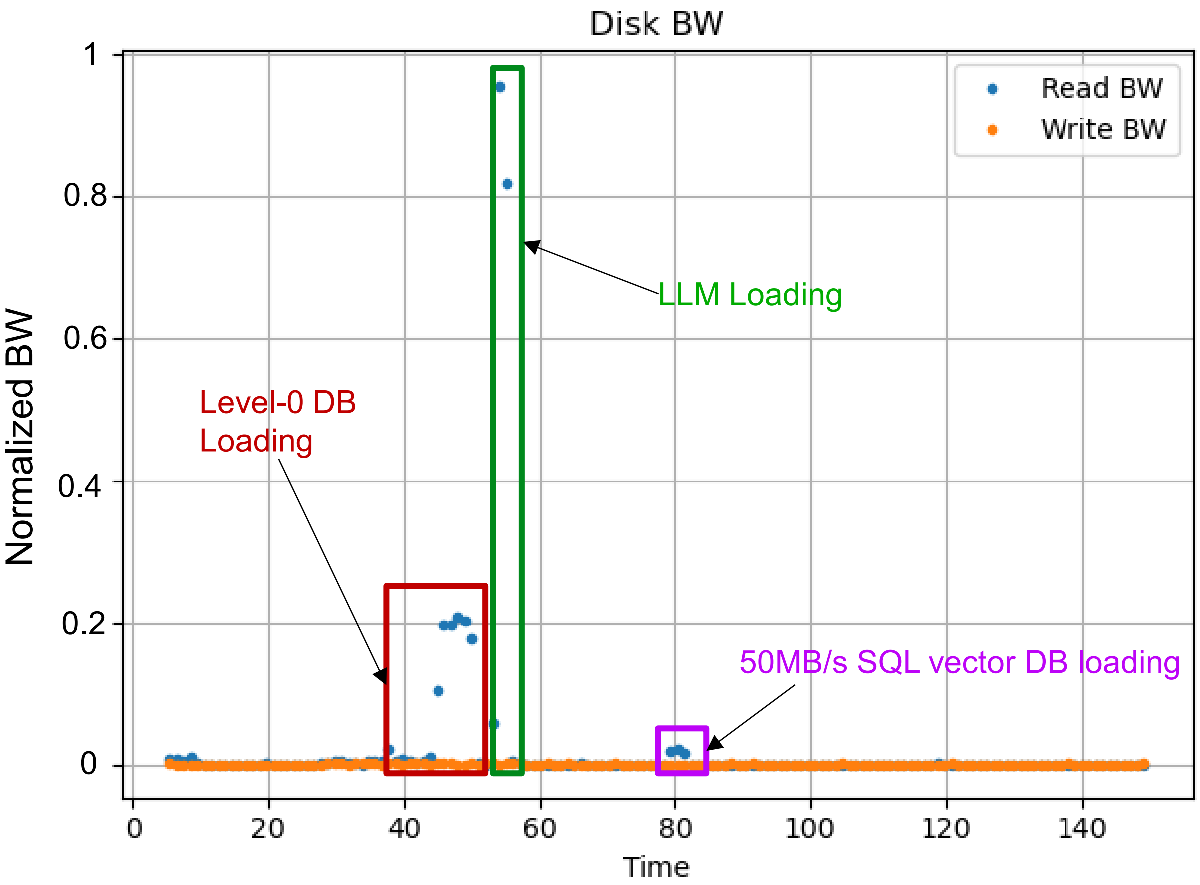
- Docustore is a DB of chunks of user documents (3x expansion)
- Top ~10 most relevant document chunks are read from disk
  - Small 4k random reads

LLM Inferencing

Rag

Model and Database Loading

- Loading sequence:
  1. Level-0 index (2.5GB) loaded at first
     - Small sized single stream sequential read, lower QD, 500MB/s
  2. LLM model (5GB) loaded next
     - Large sized multiple stream sequential read, higher QD, 2.5GB/s
  3. After user query, portion of SQL DB (150MB/6GB) loaded for similarity search
     - 4kB random read, QD 1, 50MB/s
     - 100k 4k reads into ~150MB of LBA range for single query, many cache hits
  4. Document chunk loaded (several 4kB)
     - 4kB random read, low QD
  5. SQL DB loaded for every "***cold***" query



Unique Characteristics of AI Inferencing Traffic

Summary

- Uniqueness of AI traffic affords opportunity to target these traffic for performance improvements
  1. Multi-stream sequential reads
  2. Large volume (GB levels) of continuous read operations
  3. Mixed read/write patterns:
     - SR+SW for memory swapping