Managed Observability Services for NOC/SOC

Application Modernization Solution Accelerators

API rich, Microservices Container Native Service Development

Data Lakehouse Engineering to support Customers GEN AI Initiatives

**Custom Platform Engineering Consulting and Advisory Services**

SLA driven Projects Delivery

**Austin** | Bangalore | Trivandrum

UNOVIE

# EDGEAI GW200


100 TOPS Intelligent Gateway

LTE 4G+


llama3 Gemma2 Mistral

TPU Mesh Orchestrator

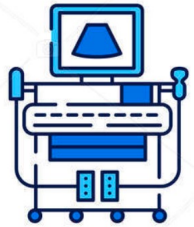
FE100

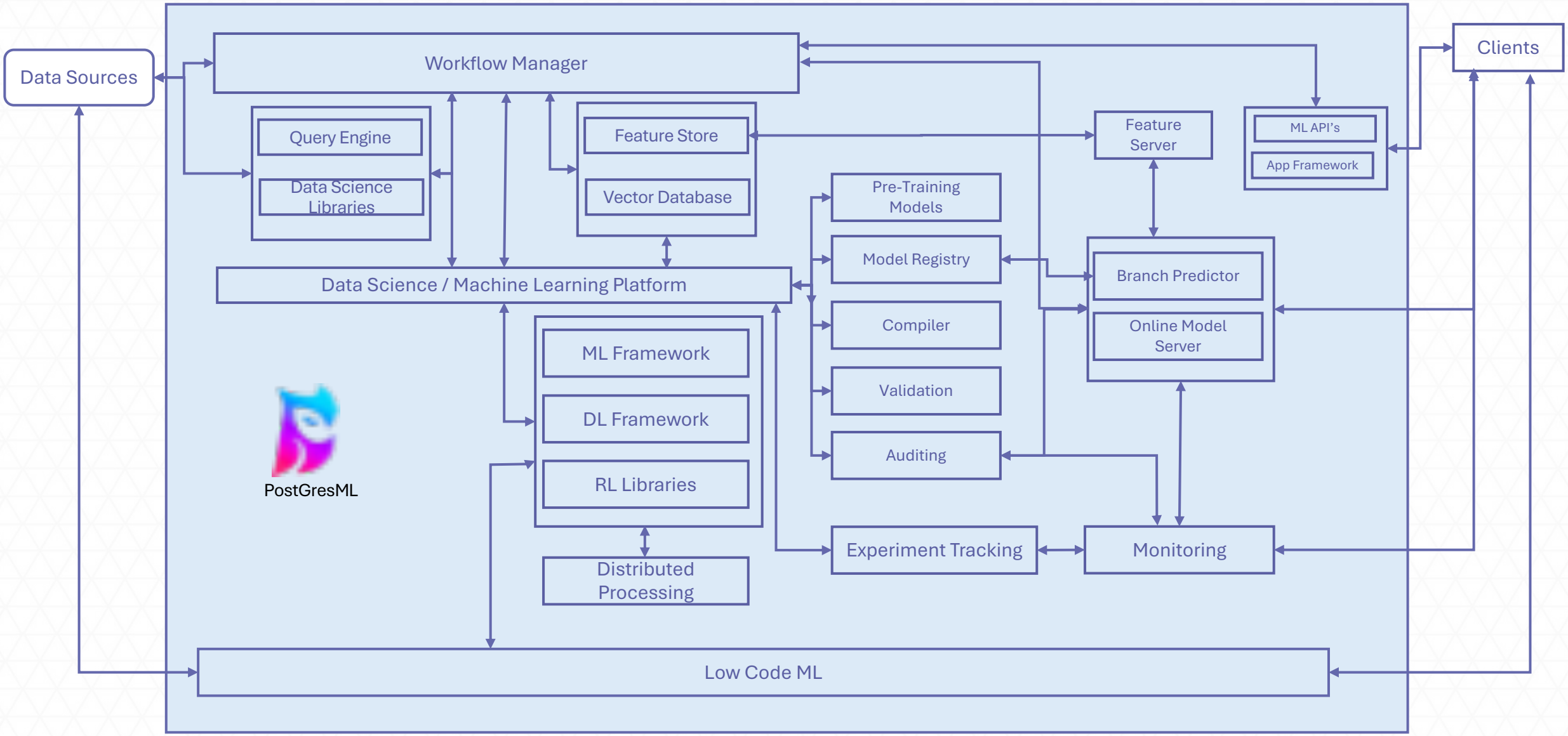
TPU Mesh

FE100

TPU Mesh

FE100





UNOVIE

# EDGEAI DC300

## Stackable

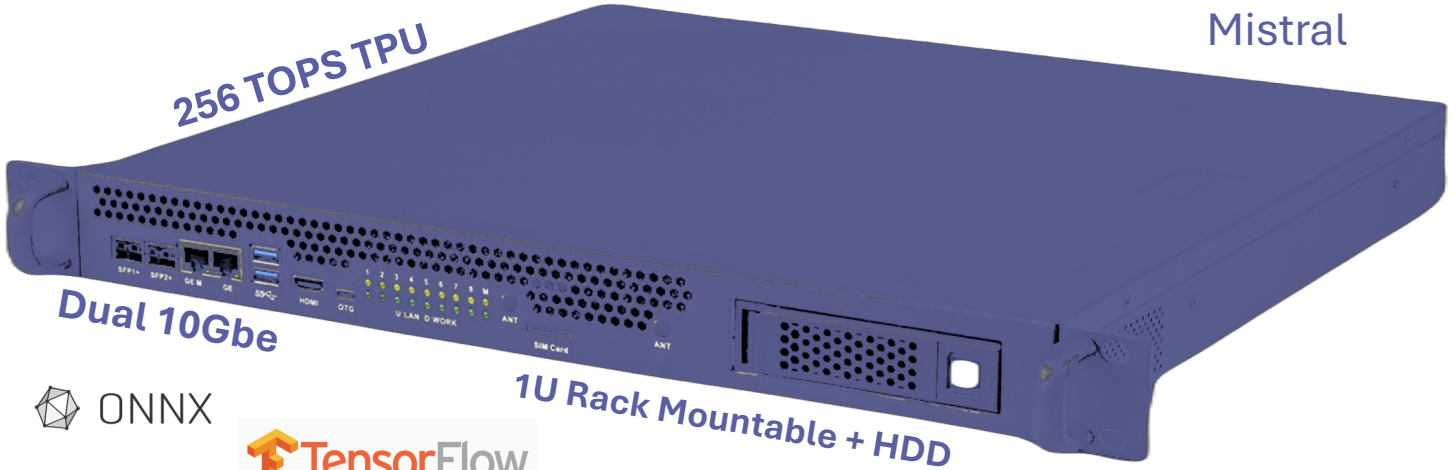
### Rack Cluster

llama3

Gemma2

Mistral

256 TOPS TPU



Dual 10Gbe

1U Rack Mountable + HDD

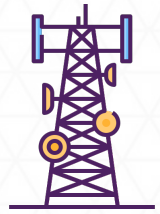
ONNX

TensorFlow

PYTORCH

UNOVIE

# EDGEAI DC300

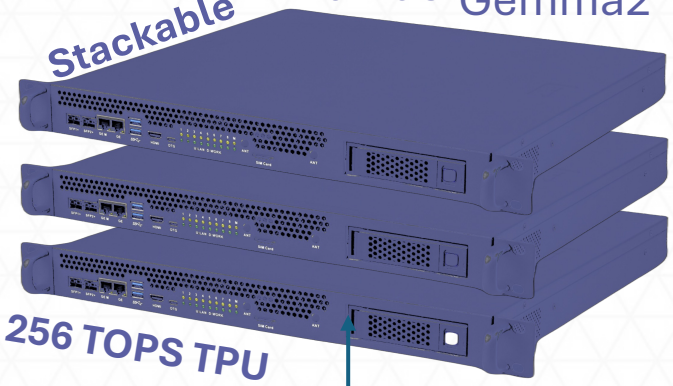


llama3 Gemma2 Mistral

Stackable

Enterprise Systems

10G



256 TOPS TPU

llama3 Gemma2 Mistral

LTE 4G+

GW200

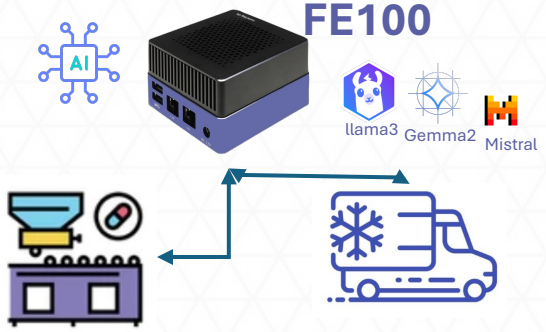
100 TOPS

Intelligent Gateway
Mesh Orchestrator

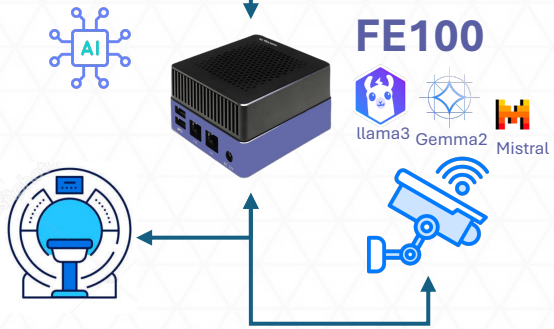
ZERO TRUST

Tensor Mesh

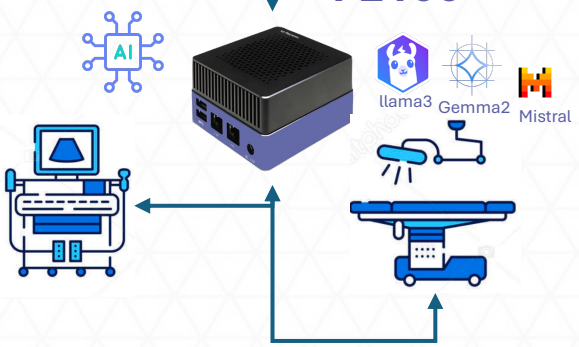
FE100

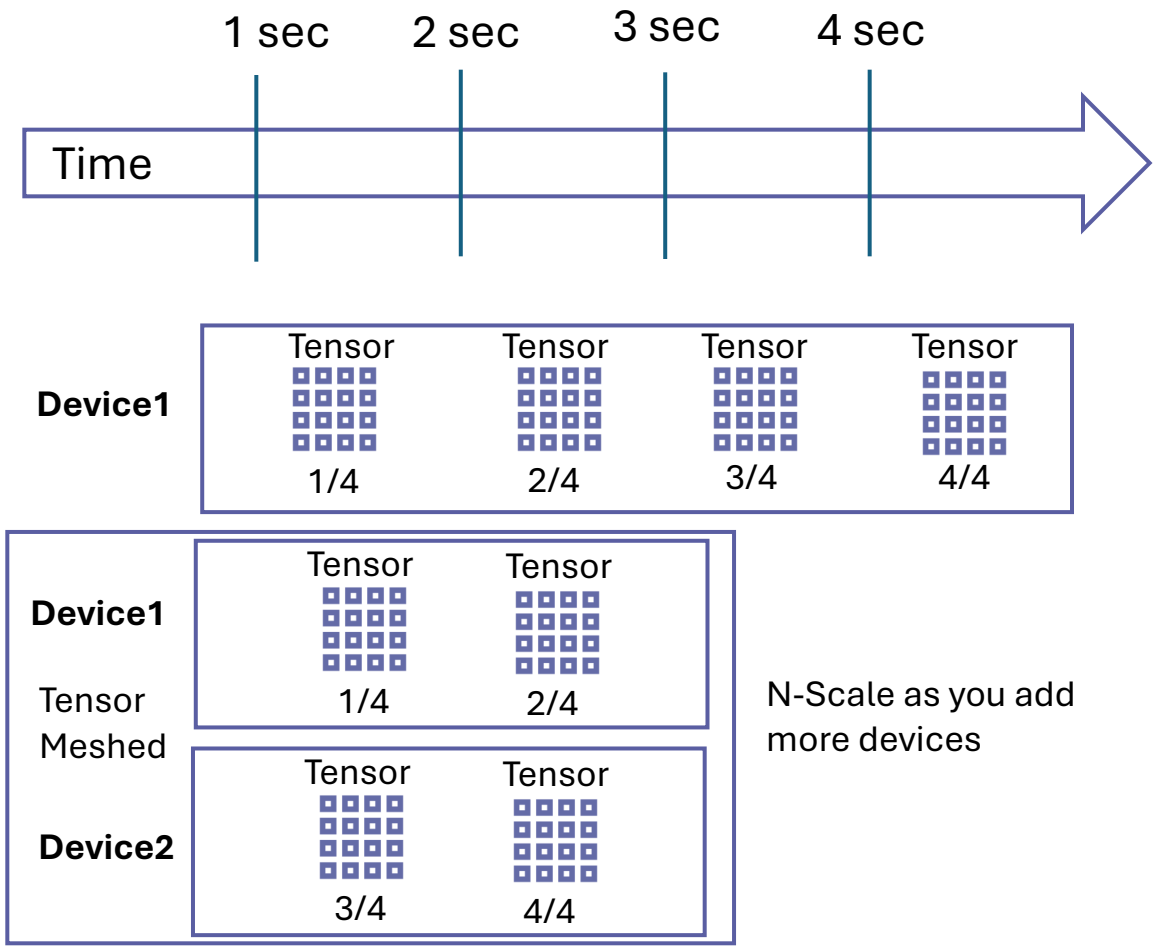


FE100

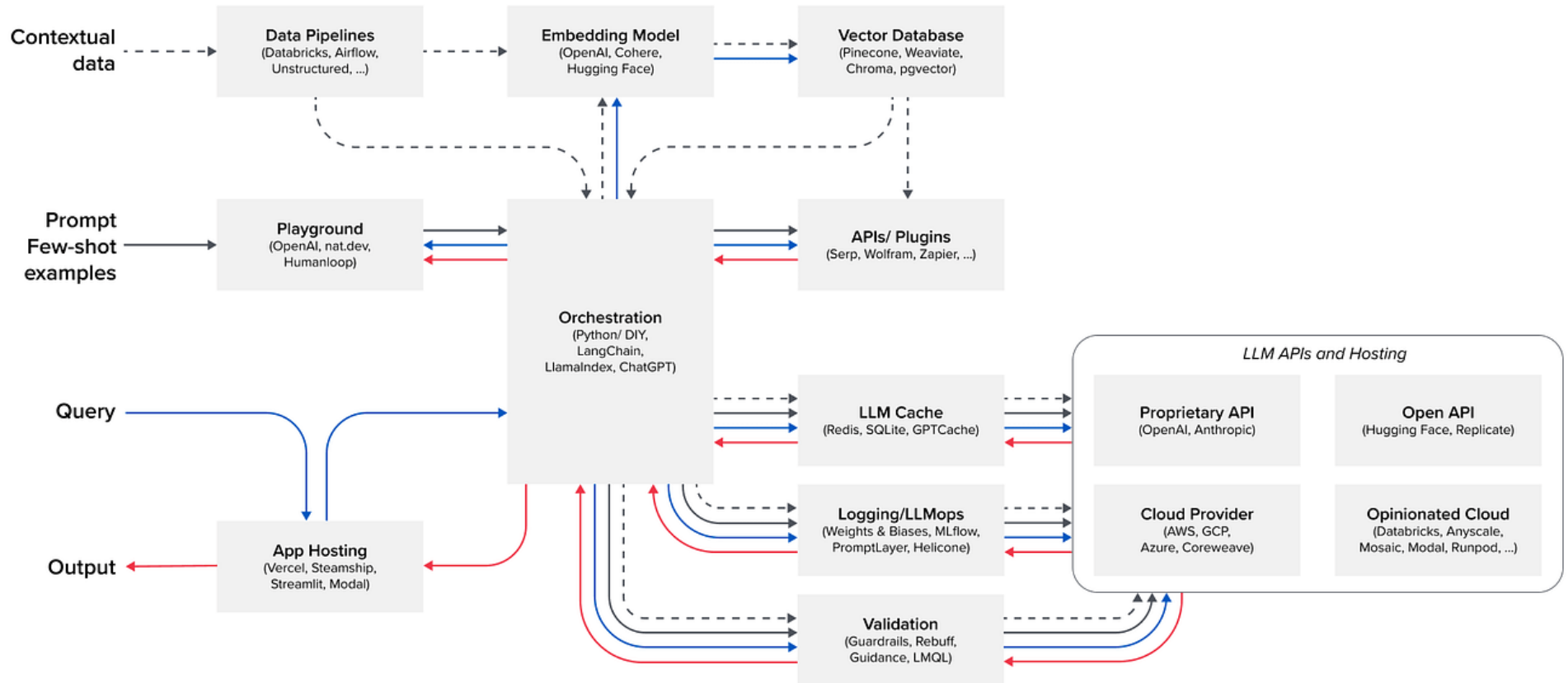


FE100




1 sec
2 sec
3 sec
4 sec
Time
Device1
Tensor
Tensor
Tensor
Tensor
1/4
2/4
3/4
4/4
Device1
Tensor
Meshed
Device2
Tensor
Tensor
1/4
2/4
Tensor
Tensor
3/4
4/4
N-Scale as you add more devices

# Emerging LLM App Stack




LEGEND

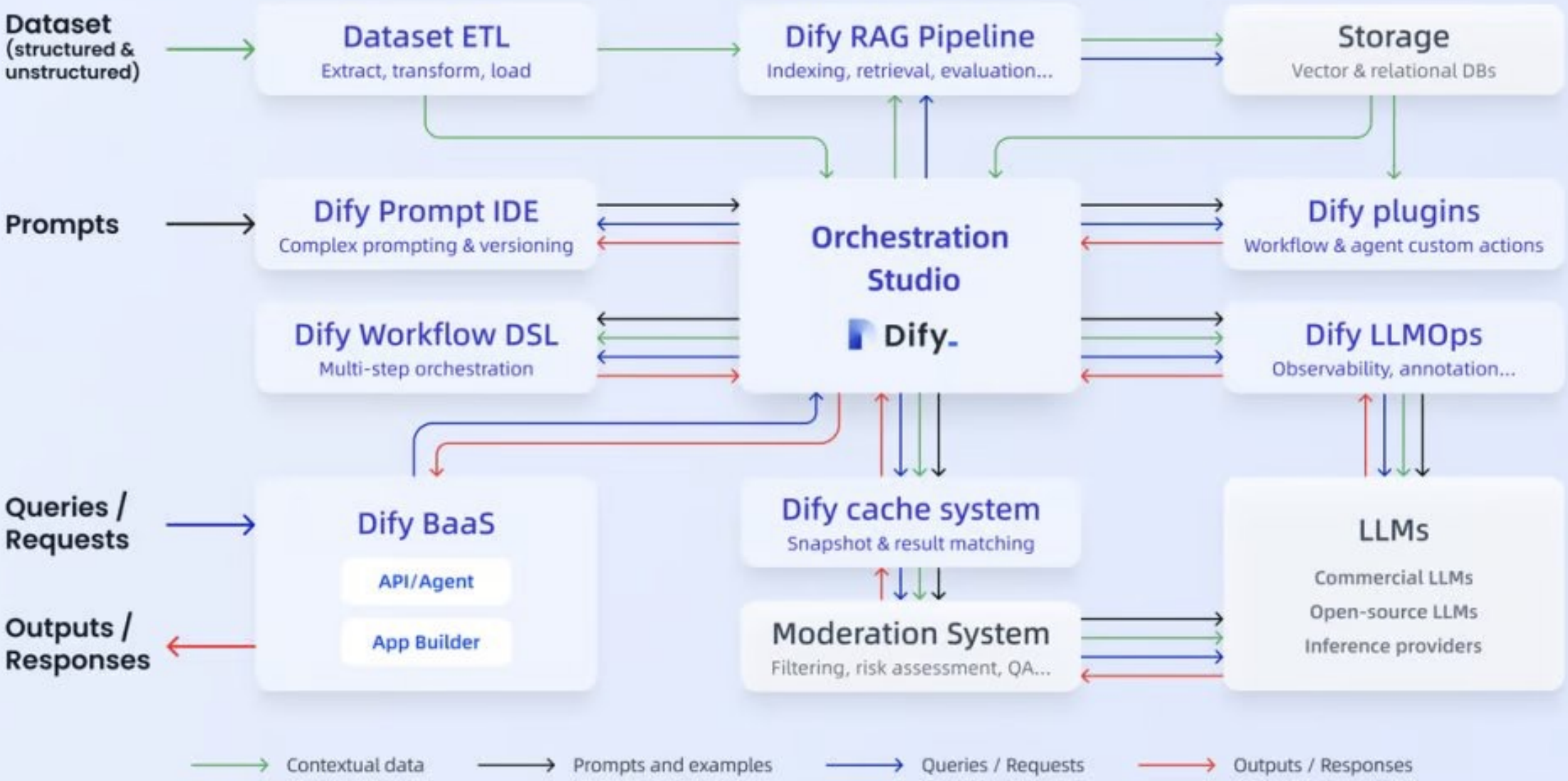
Gray boxes show key components of the stack, with leading tools/systems listed

Arrows show the flow of data through the stack

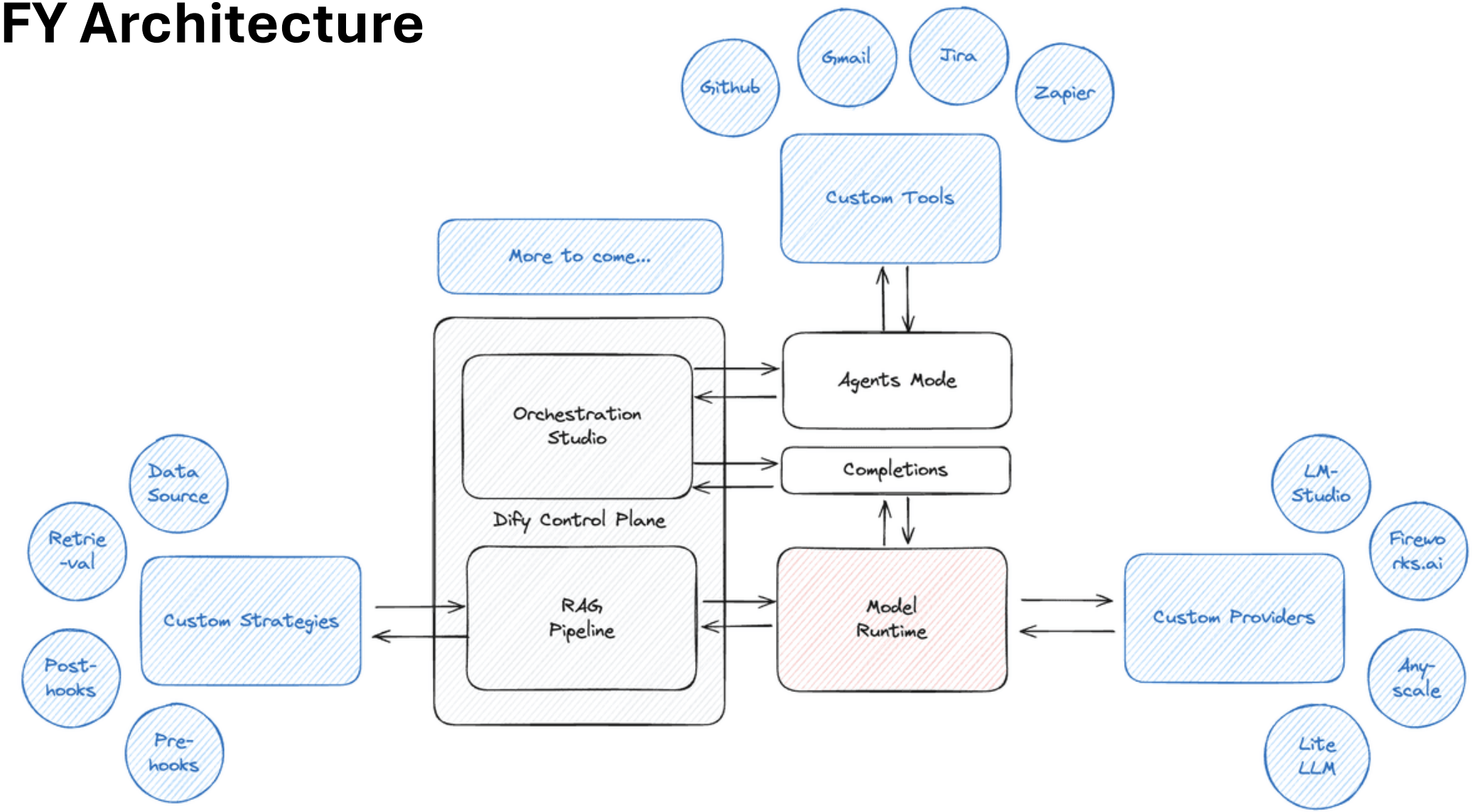
- Contextual data provided by app developers to condition LLM outputs
- Prompts and few-shot examples that are sent to the LLM
- Queries submitted by users
- Output returned to users

a16z Enterprise

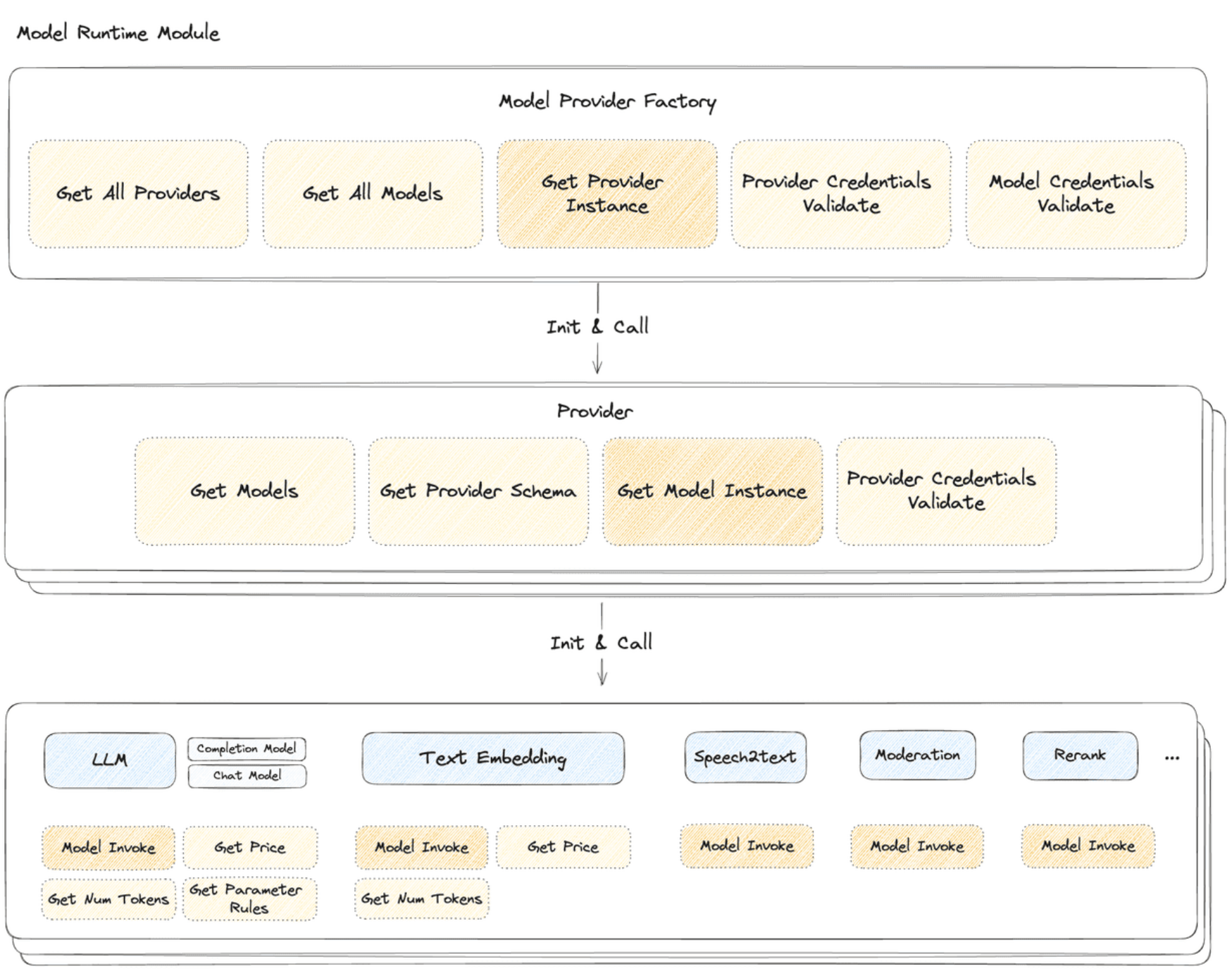
# Dify Architecture



# DIFY Architecture



# DIFY Architecture



Haystack
by deepset

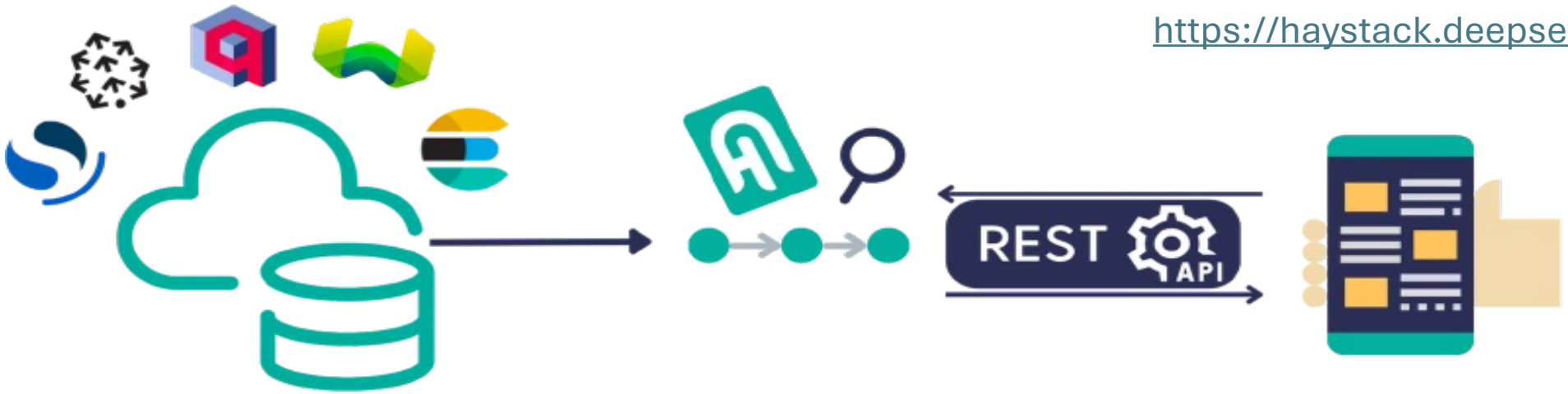
Haystack is an open source framework for building production-ready *LLM applications*, *retrieval-augmented generative pipelines* and *state-of-the-art search systems* that work intelligently over large document collections.

## Powerful Pipelines

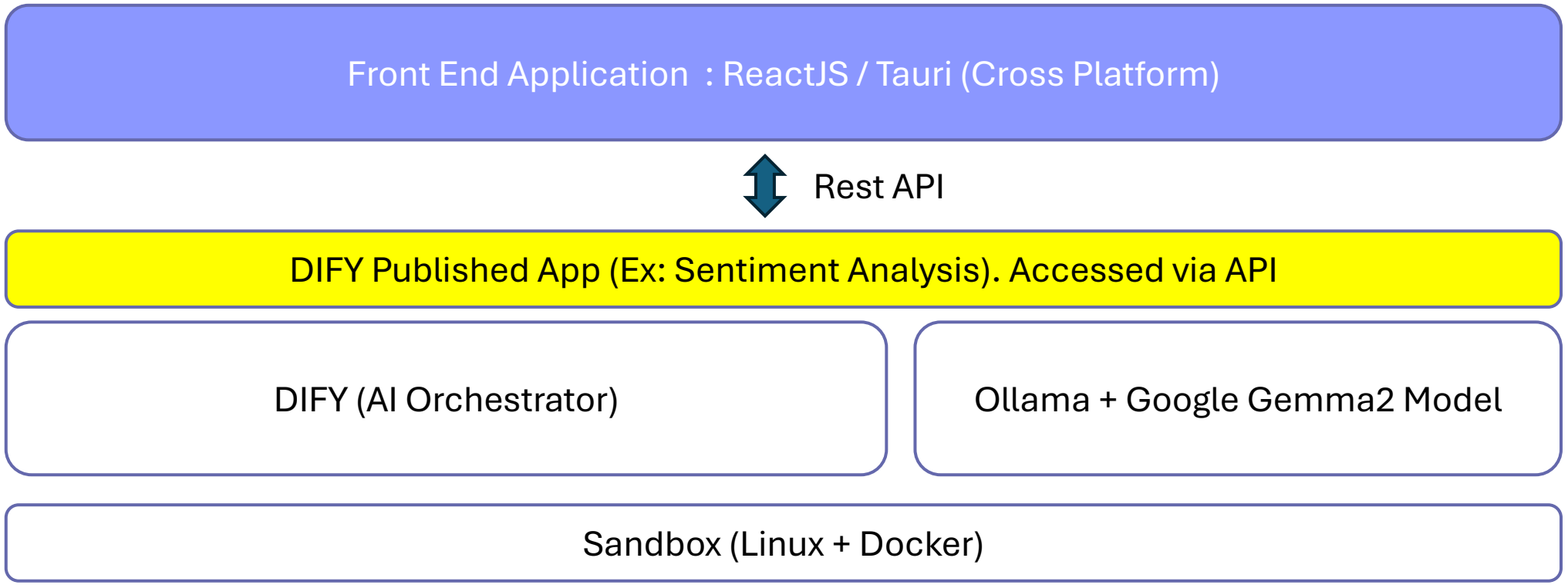
In Haystack 2.0, pipelines are dynamic computation graphs that support:

- 🚦 **Control flow:** Need to run different components based on the output of another? Not a problem with 2.0.
- 👓 **Loops:** Implement complex behavior such as self-correcting flows by executing parts of the graph repeatedly.
- 🎛️ **Data flow**: Consume it only where you need it. Haystack 2.0 only exposes data to components which need it - benefiting speed and transparency.
- ✅ **Validation and type-checking:** Ensures all components in your pipeline are compatible even before running it.
- 💾 **Serialization:** Save and restore your pipelines from different formats.

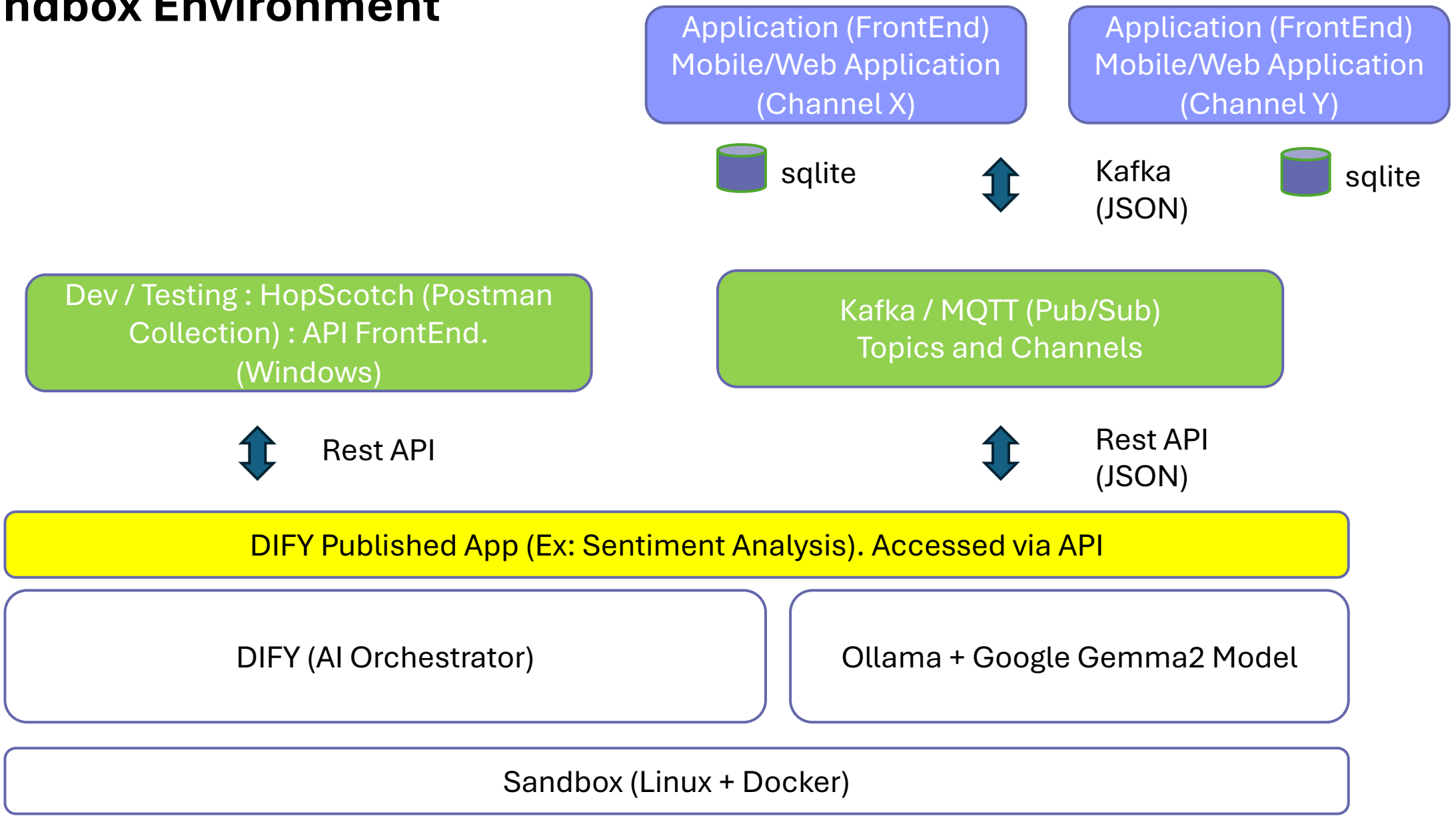
https://haystack.deepset.ai/overview/intro



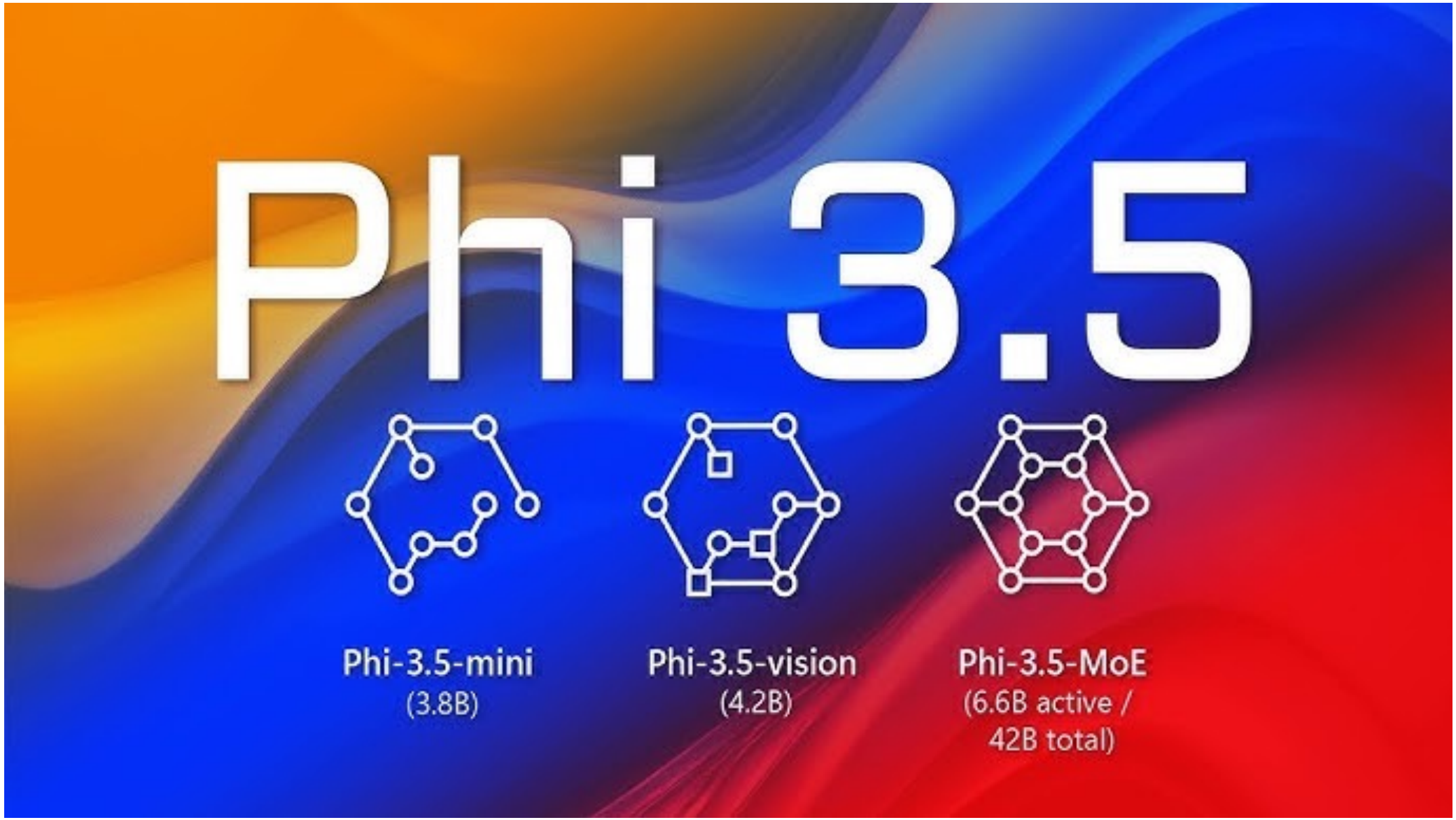
# Sandbox Environment



# Sandbox Environment

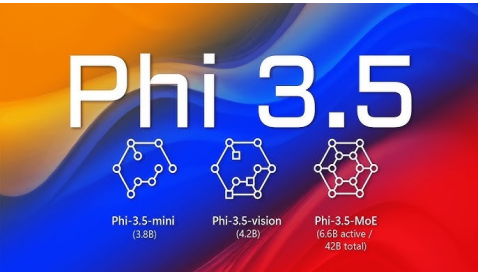


# Phi-3.5 Small Language Model (SLM)




Aug 21st 2024

# Phi-3.5 Small Language Model (SLM)



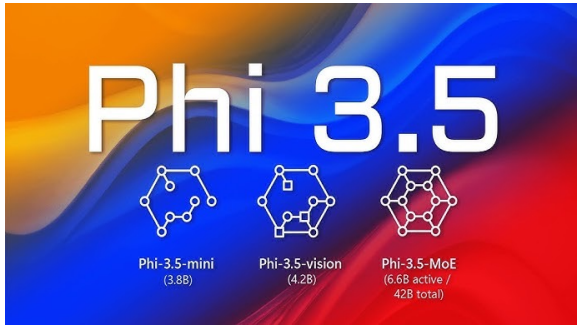

The **Phi-3.5 Mini Instruct** is a compact AI model with 3.8 billion parameters. It's designed for efficient instruction processing and supports a 128k token context length. It excels in environments with limited memory or computational resources. This makes it ideal for code generation, solving math problems, and logical reasoning.

Despite its smaller size, this model achieves impressive results in multilingual and multi-turn conversations, showcasing notable advancements over its predecessors. It performs exceptionally well on various benchmarks, often surpassing other models of similar size, like Llama-3.1-8B-instruct and Mistral-7B-instruct, particularly in the RepoQA benchmark, which evaluates long-context code understanding.

Aug 21st 2024

infinite

# Phi-3.5 Small Language Model (SLM)

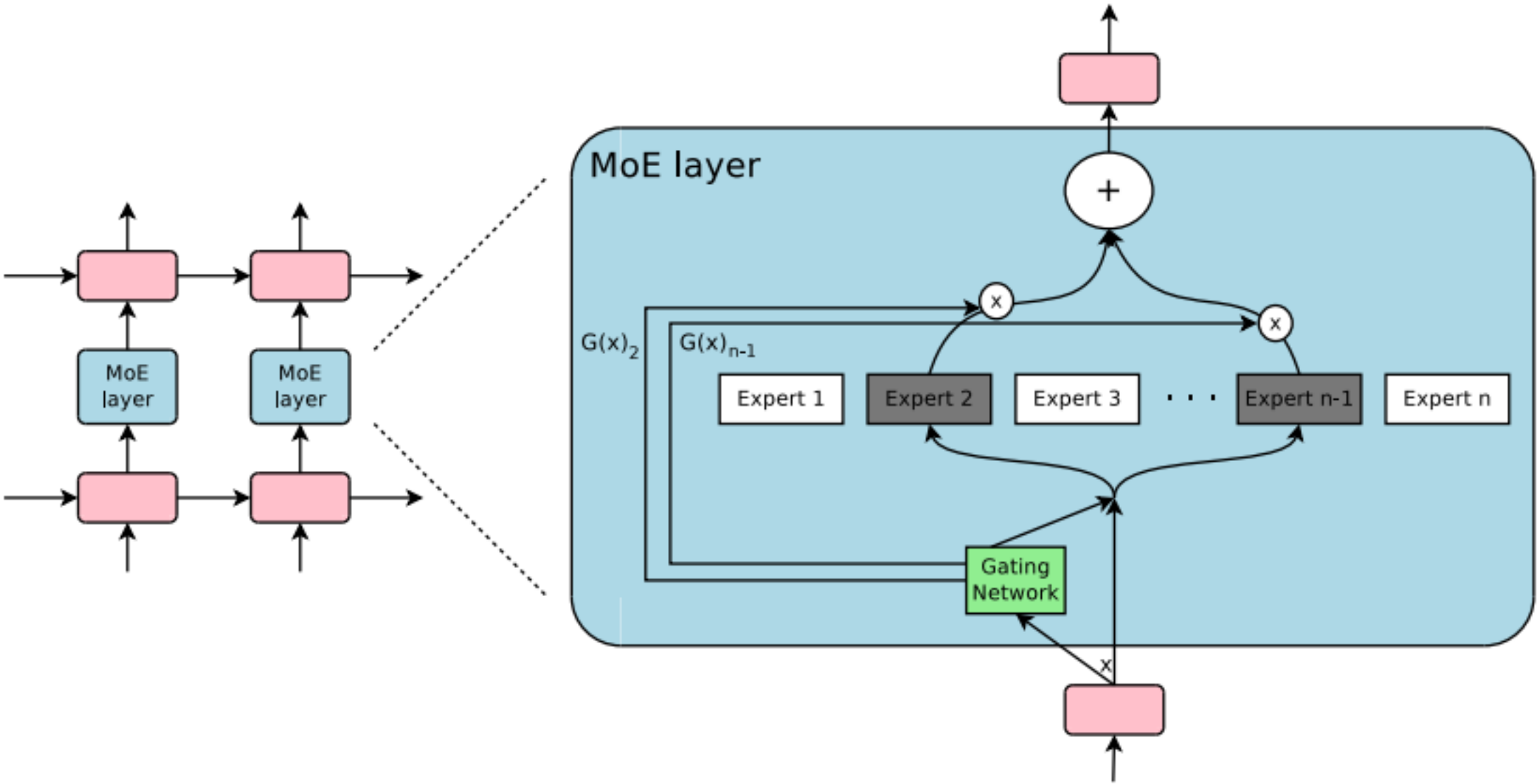



Aug 21st 2024

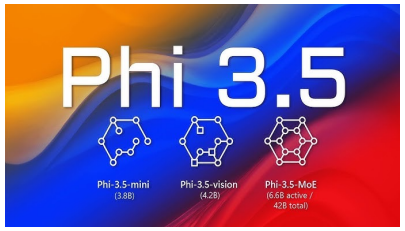
The **Phi-3.5 MoE (Mixture of Experts)** model marks a novel approach from the company, integrating multiple specialized models into a single framework. With an architecture supporting 42 billion parameters and a 128k token context length, it offers scalable performance for complex tasks.

However, it actively utilizes only 6.6 billion parameters. This model is tailored for advanced reasoning tasks, including code generation, mathematical problem solving, and multilingual comprehension. It frequently outperforms larger models in targeted benchmarks, such as RepoQA, demonstrating its efficiency in specific areas.

The model has been trained on selective set of languages listed here: Arabic, Chinese, Czech, Danish, Dutch, English, Finnish, French, German, Hebrew, Hungarian, Italian, Japanese, Korean, Norwegian, Polish, Portuguese, Russian, Spanish, Swedish, Thai, Turkish and Ukrainia


MoE layer
MoE layer
MoE layer
+
x
x
$G(x)_2$
$G(x)_{n-1}$
Expert 1
Expert 2
Expert 3
Expert n-1
Expert n
Gating Network
x

# Phi-3.5 Small Language Model (SLM)



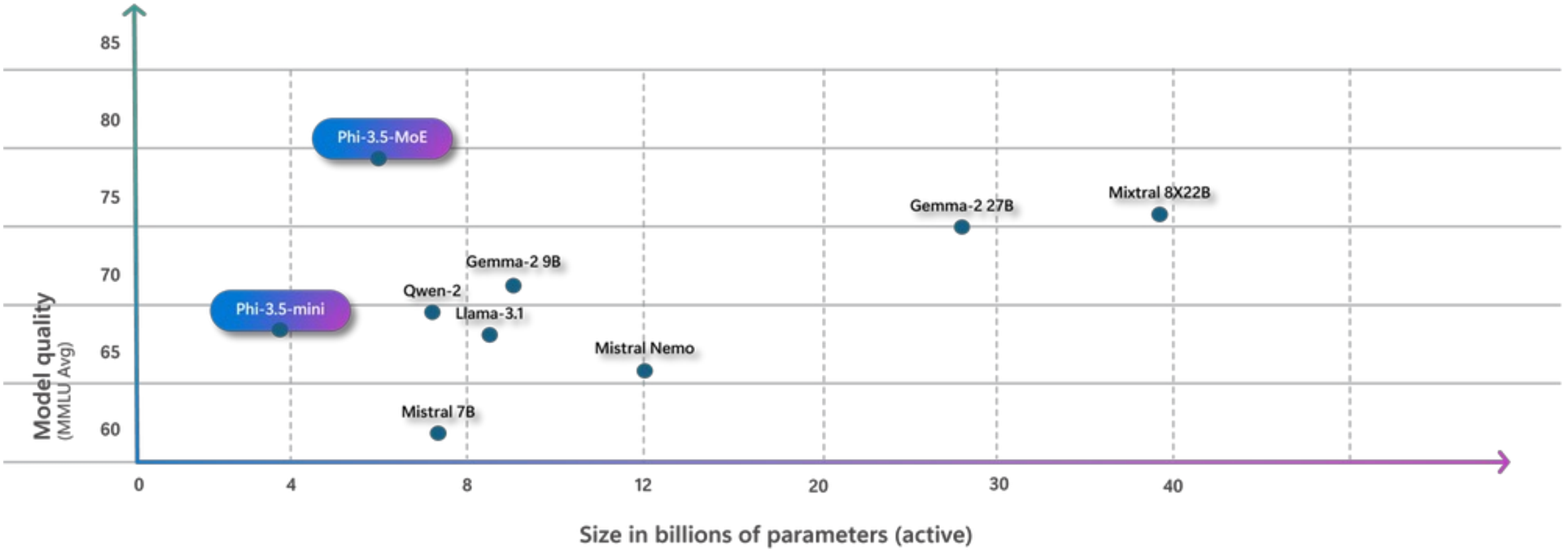

Aug 21st 2024

**Phi-3.5 Vision Instruct** rounds out the Phi-3.5 series with its ability to process both text and images. This multimodal model excels in a variety of tasks. Tasks like image interpretation, optical character recognition, understanding charts and tables, and summarizing videos. It shares the 128k token context length of its Phi-3.5 counterparts, allowing it to handle intricate, multi-frame visual tasks efficiently.

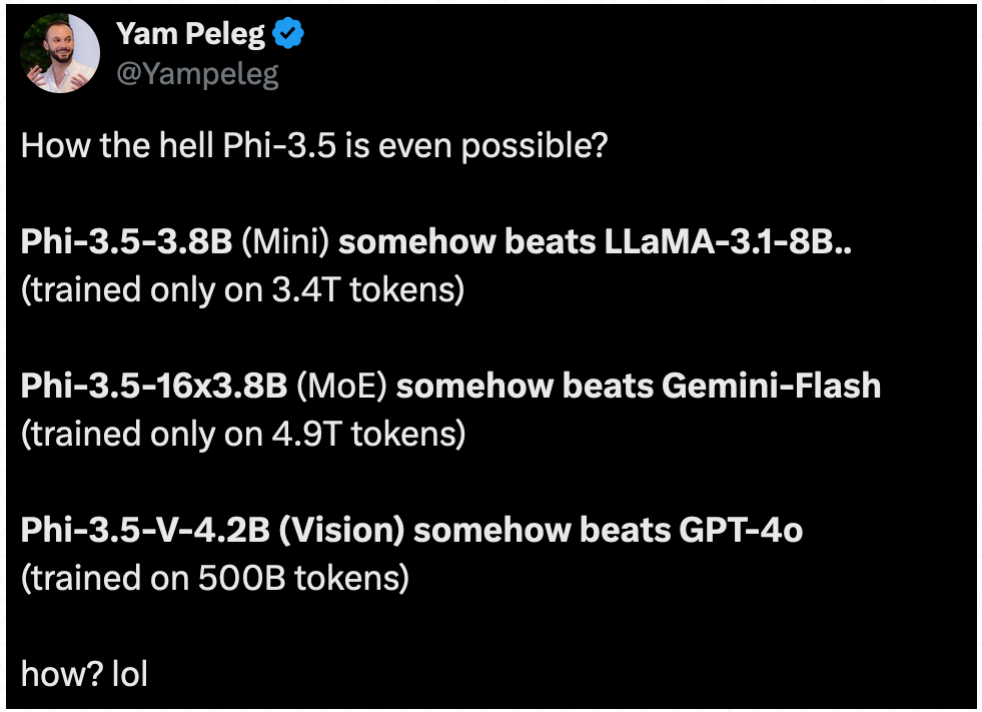
Microsoft trained the Vision Instruct model on a mix of synthetic and carefully curated publicly available datasets, emphasizing high-quality, data-rich reasoning capabilities.

infinite

# Phi-3.5 Quality vs Size in SLM



# Phi-3.5 Small Language Model (SLM)

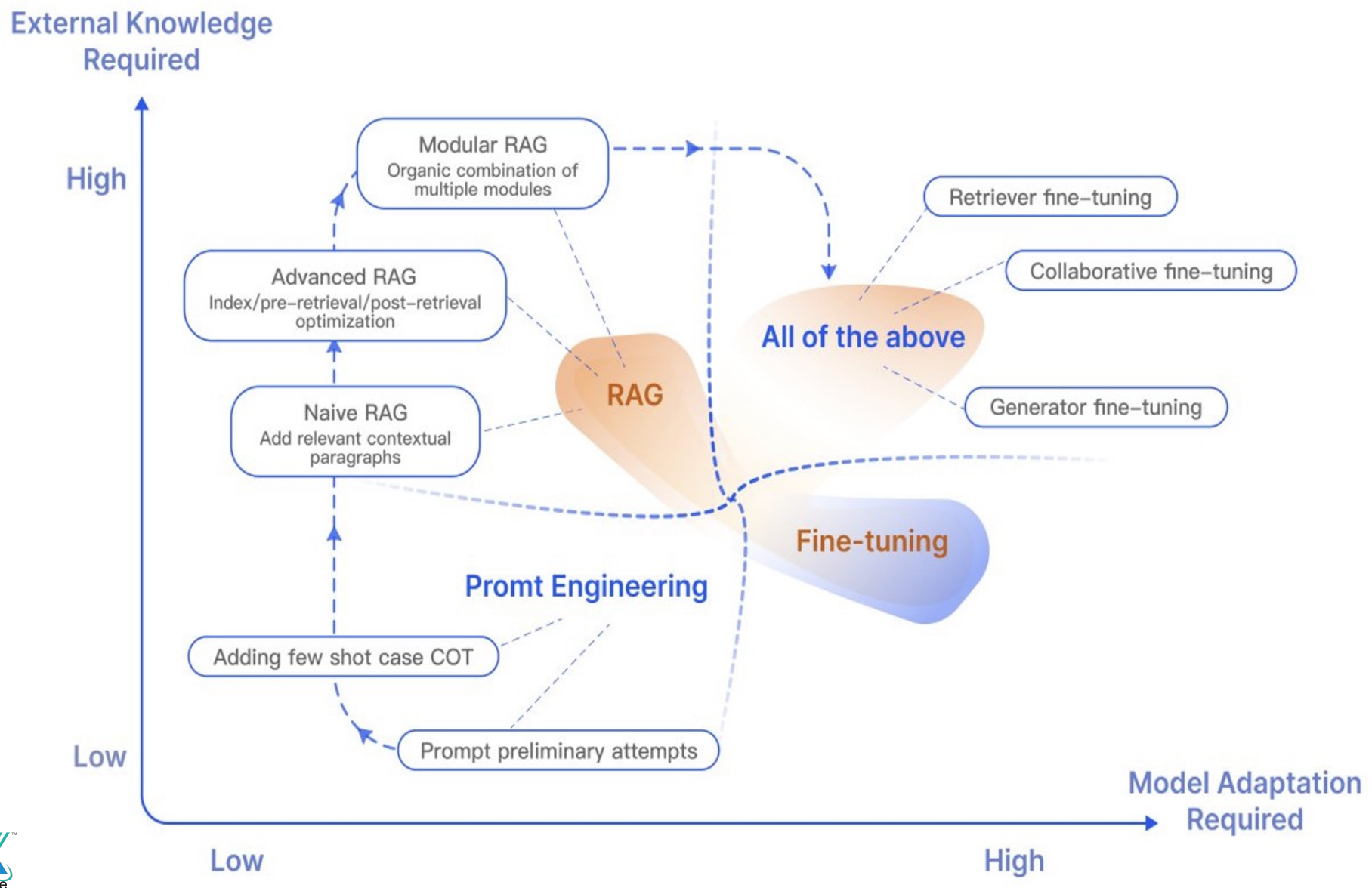



| Category | Benchmark | Phi-3.5-MoE-instruct | Mistral-Nemo-12B-instruct-2407 | Llama-3.1-8B-instruct | Gemma-2-9b-It | Gemini-1.5-Flash | GPT-4o-mini-2024-07-18 (Chat) |
|---|---|---|---|---|---|---|---|
| Popular aggregated benchmark | Arena Hard | 37.9 | 39.4 | 25.7 | 42.0 | 55.2 | 75.0 |
| | BigBench Hard CoT (0-shot) | 79.1 | 60.2 | 63.4 | 63.5 | 66.7 | 80.4 |
| | MMLU (5-shot) | 78.9 | 67.2 | 68.1 | 71.3 | 78.7 | 77.2 |
| | MMLU-Pro (0-shot, CoT) | 54.3 | 40.7 | 44.0 | 50.1 | 57.2 | 62.8 |
| Reasoning | ARC Challenge (10-shot) | 91.0 | 84.8 | 83.1 | 89.8 | 92.8 | 93.5 |
| | BoolQ (2-shot) | 84.6 | 82.5 | 82.8 | 85.7 | 85.8 | 88.7 |
| | GPQA (0-shot, CoT) | 36.8 | 28.6 | 26.3 | 29.2 | 37.5 | 41.1 |
| | HellaSwag (5-shot) | 83.8 | 76.7 | 73.5 | 80.9 | 67.5 | 87.1 |
| | OpenBookQA (10-shot) | 89.6 | 84.4 | 84.8 | 89.6 | 89.0 | 90.0 |

https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/discover-the-new-multi-lingual-high-quality-phi-3-5-slms/ba-p/4225280

# RAG or Fine-Tuning ?

There is a lot of confusion about when to apply which method.




**RAG** makes sense when you have a custom knowledge base and want a standard ChatGPT-like interface on top of it. RAG has multiple components to it and can be tricky to get right. However, it's definitely **easier** to implement than fine-tuning.

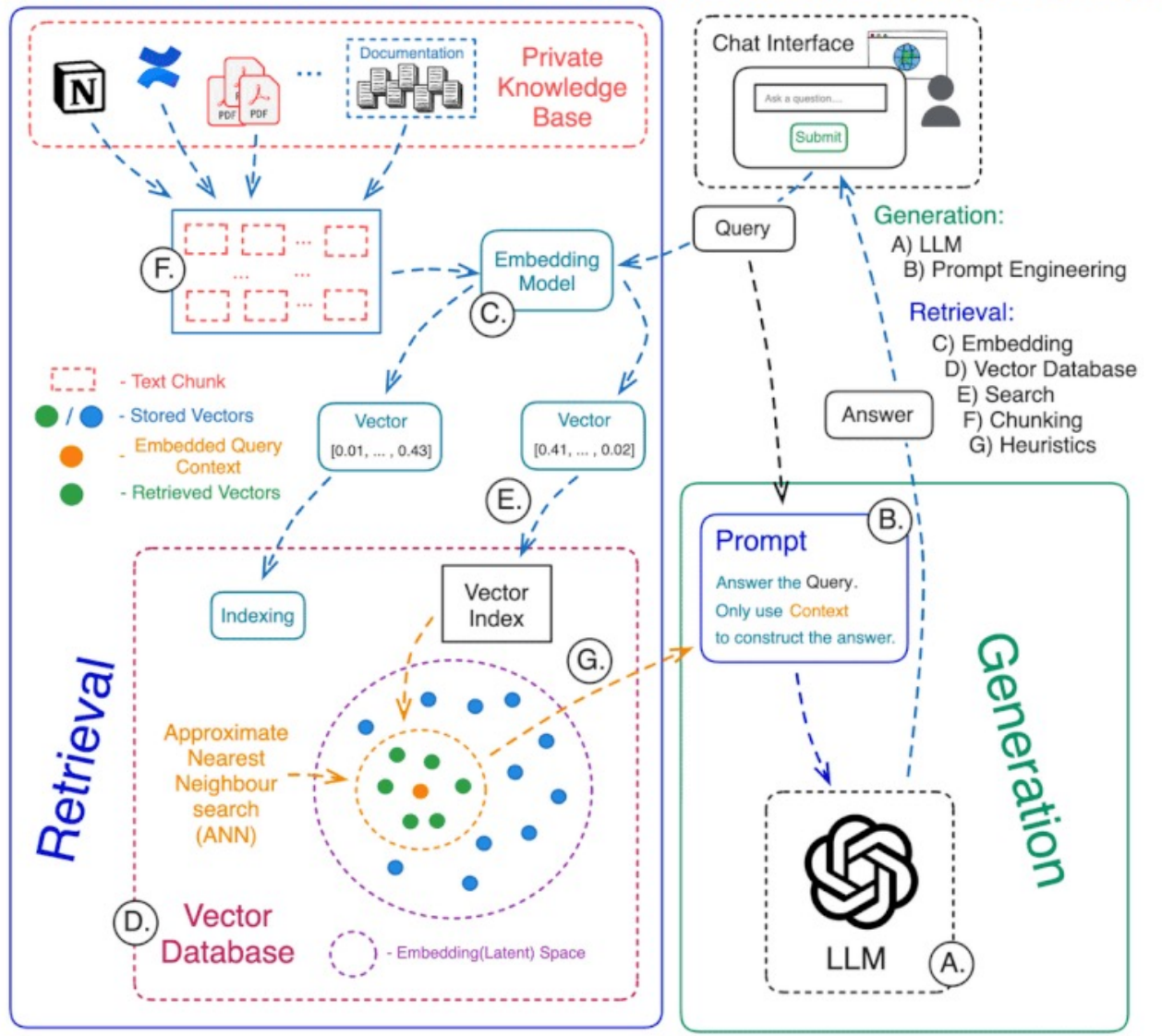
**Fine-tuning** makes sense when you have several supervised examples of request responses and are looking for a particular format for your responses. That is if you want the **model to adapt to** a particular type of response. For example, you can fine-tune a model to be good at a specific type of SQL code generation.

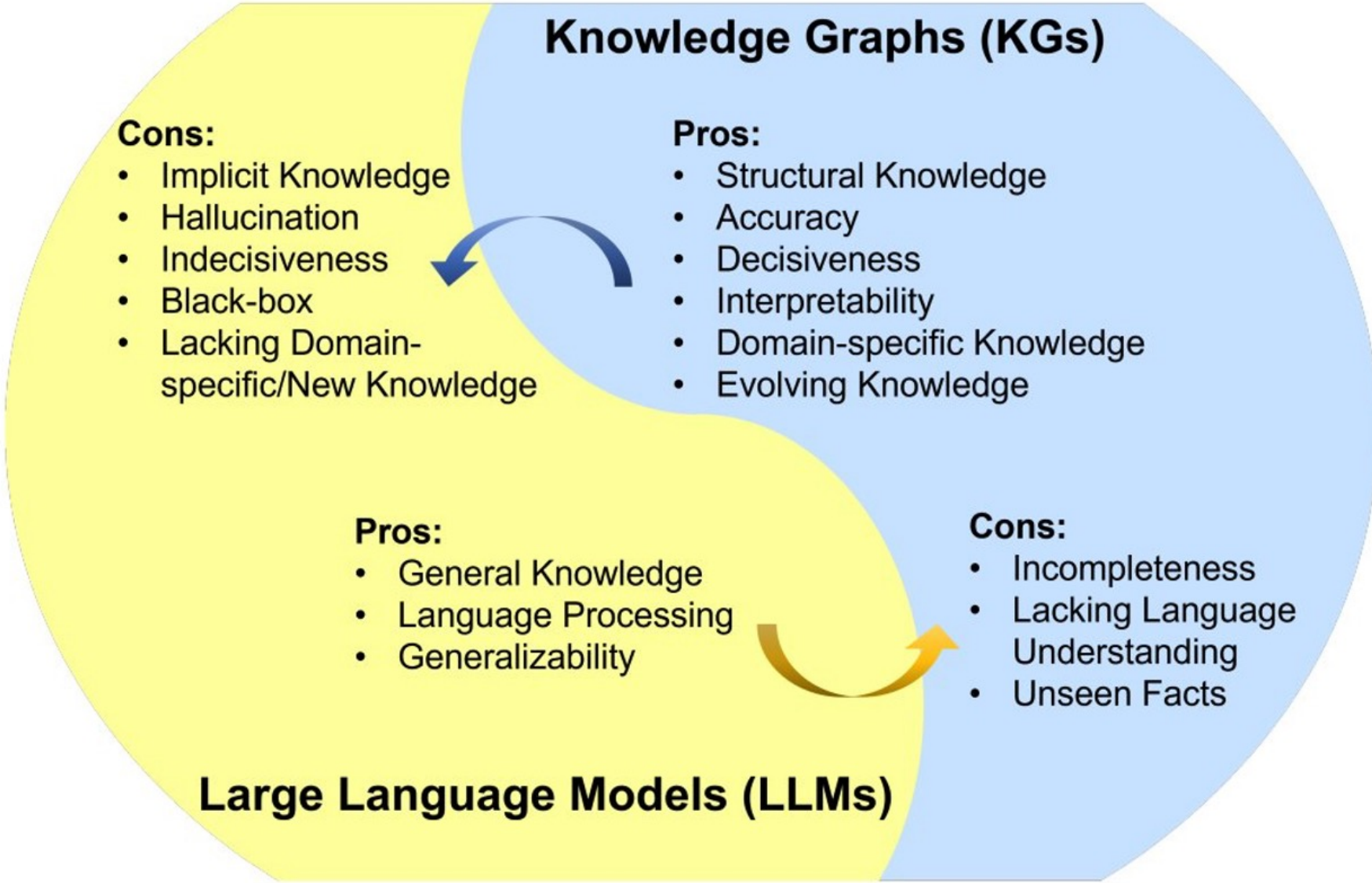
infinite

linkedin.com/in/aurimas-griciunas
@Aurimas_Gr
www.newsletter.swirlai.com
www.swirlai.com

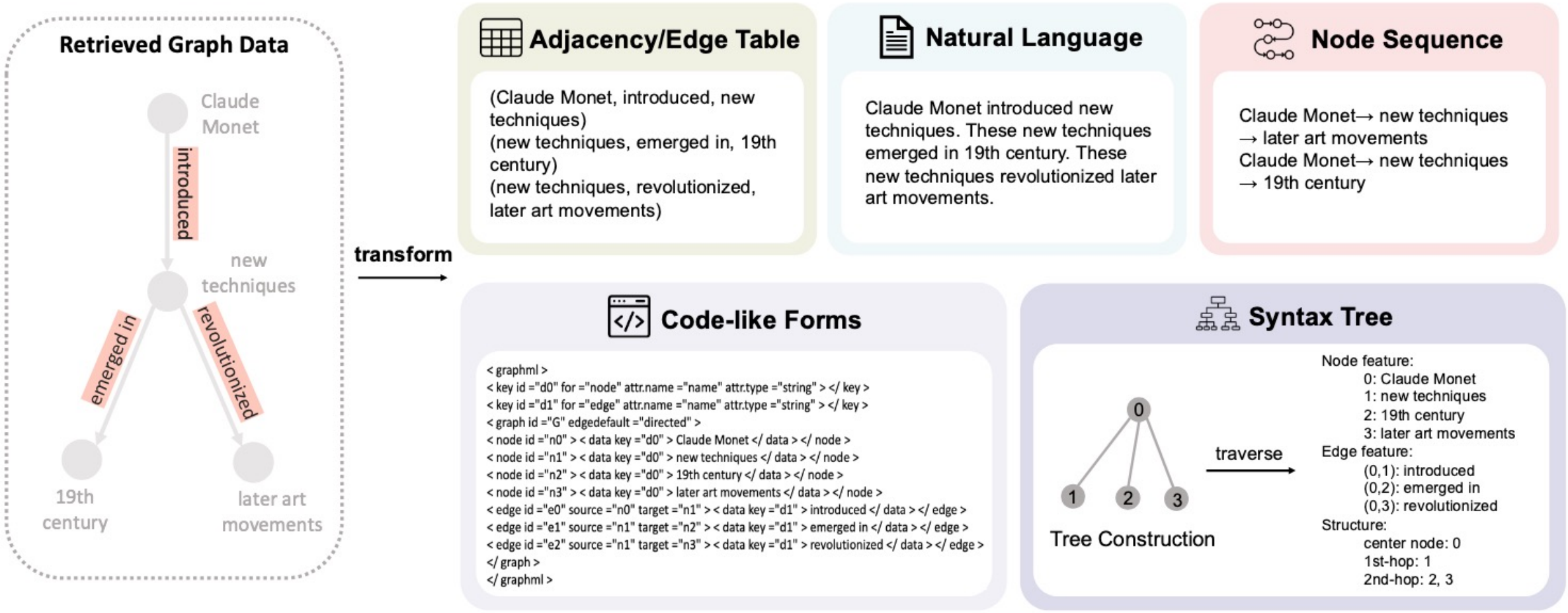
# Retrieval Augmented Generation (RAG)

## The Moving Parts

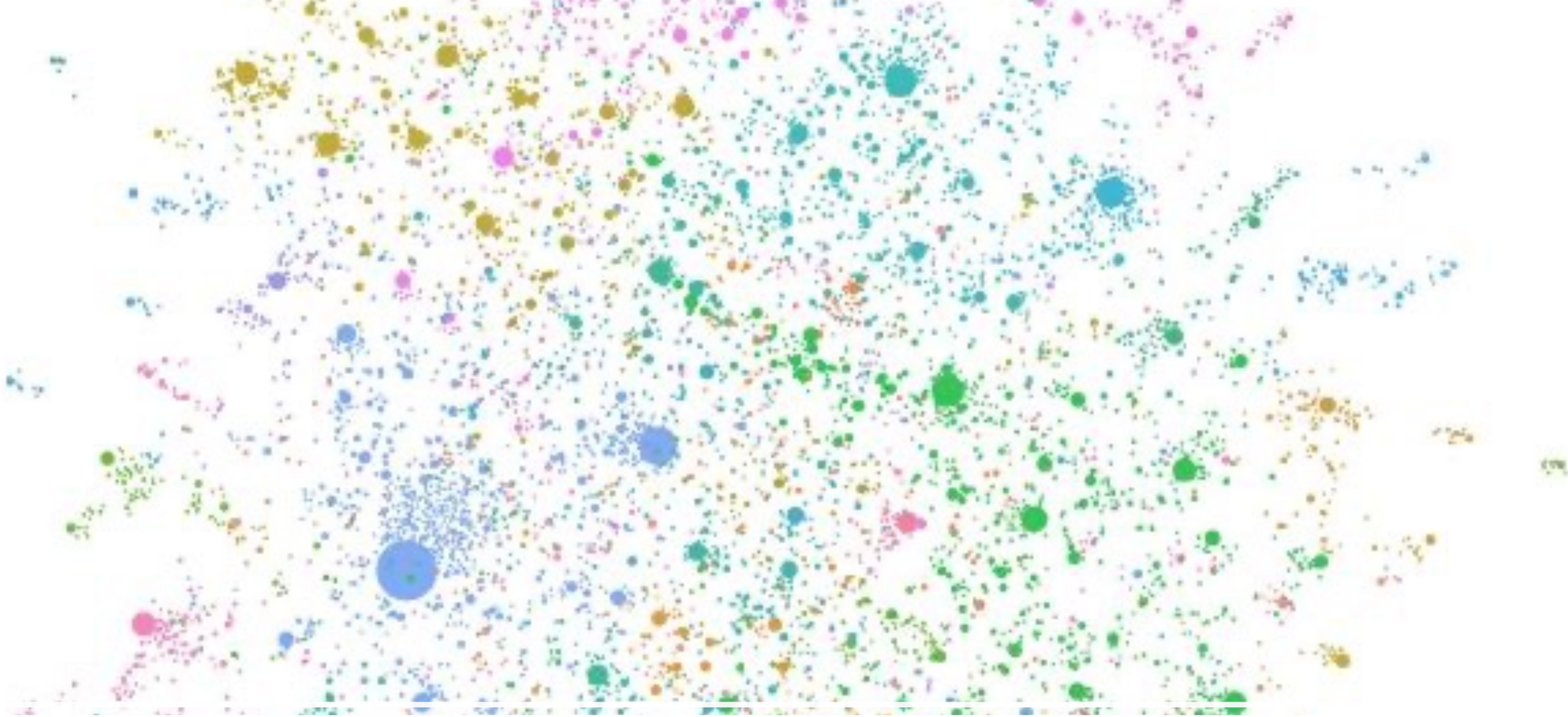



Knowledge Graphs (KGs)
Cons:
• Implicit Knowledge
• Hallucination
• Indecisiveness
• Black-box
• Lacking Domain-specific/New Knowledge
Pros:
• Structural Knowledge
• Accuracy
• Decisiveness
• Interpretability
• Domain-specific Knowledge
• Evolving Knowledge
Pros:
• General Knowledge
• Language Processing
• Generalizability
Cons:
• Incompleteness
• Lacking Language Understanding
• Unseen Facts
Large Language Models (LLMs)

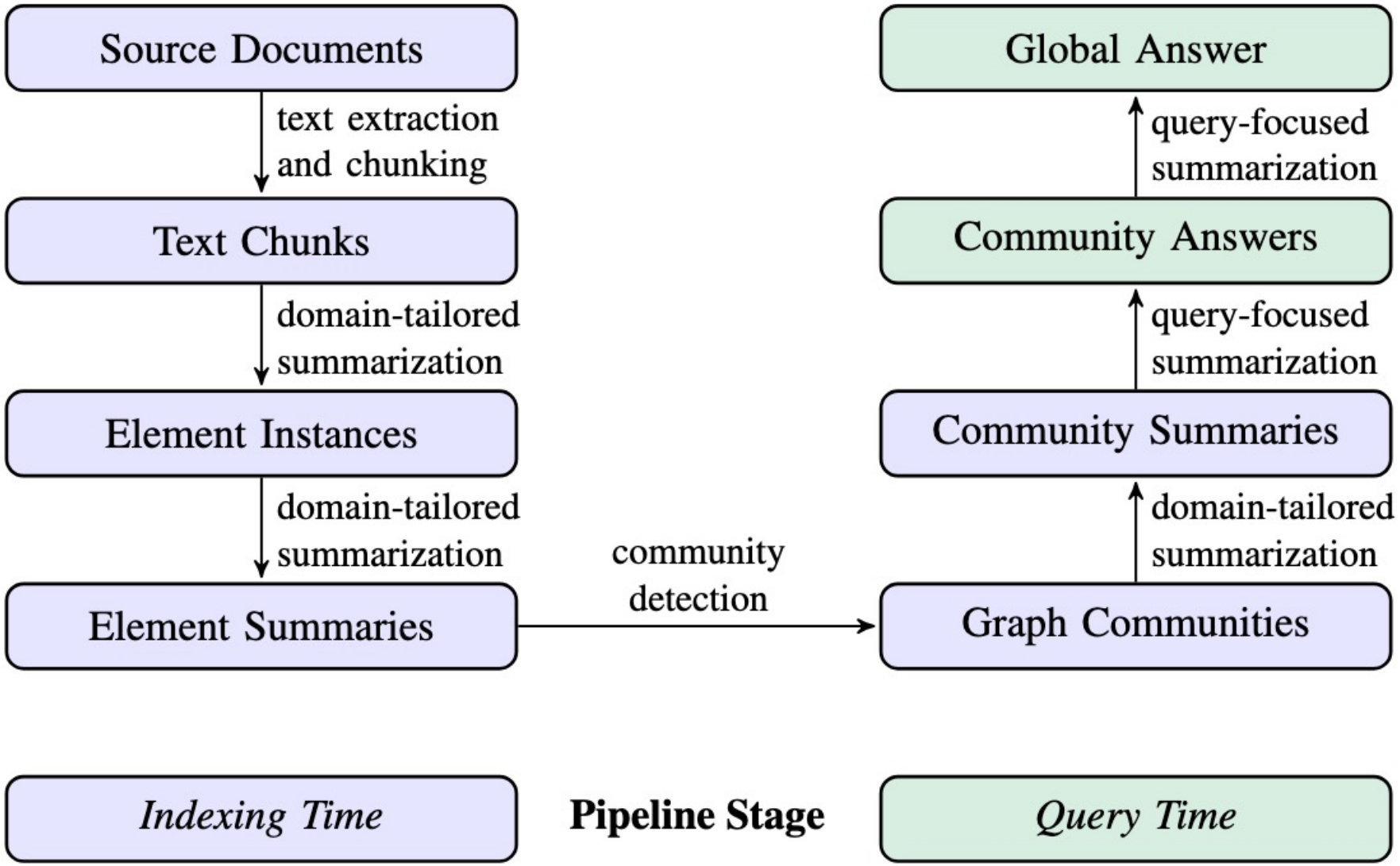
# Graph Language




Source : Paper : https://arxiv.org/pdf/2408.08921

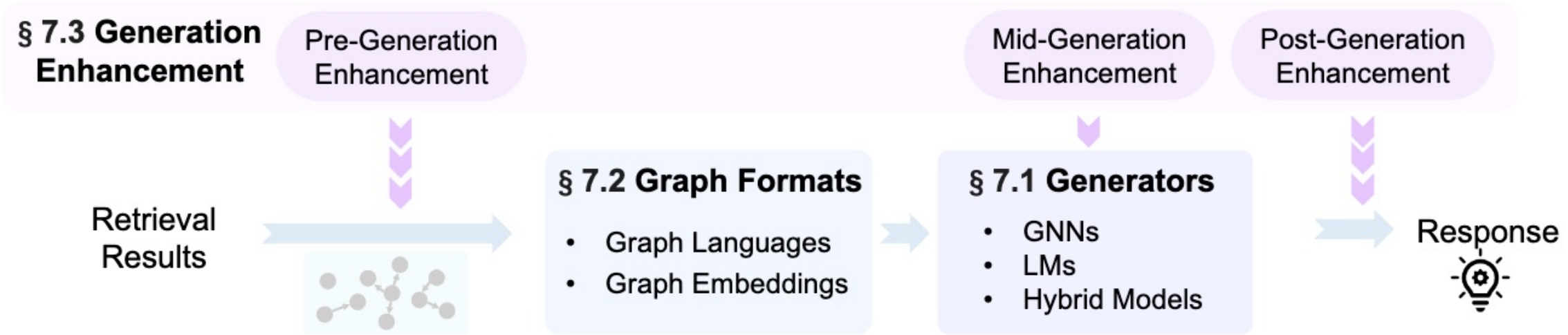


https://microsoft.github.io/graphrag/

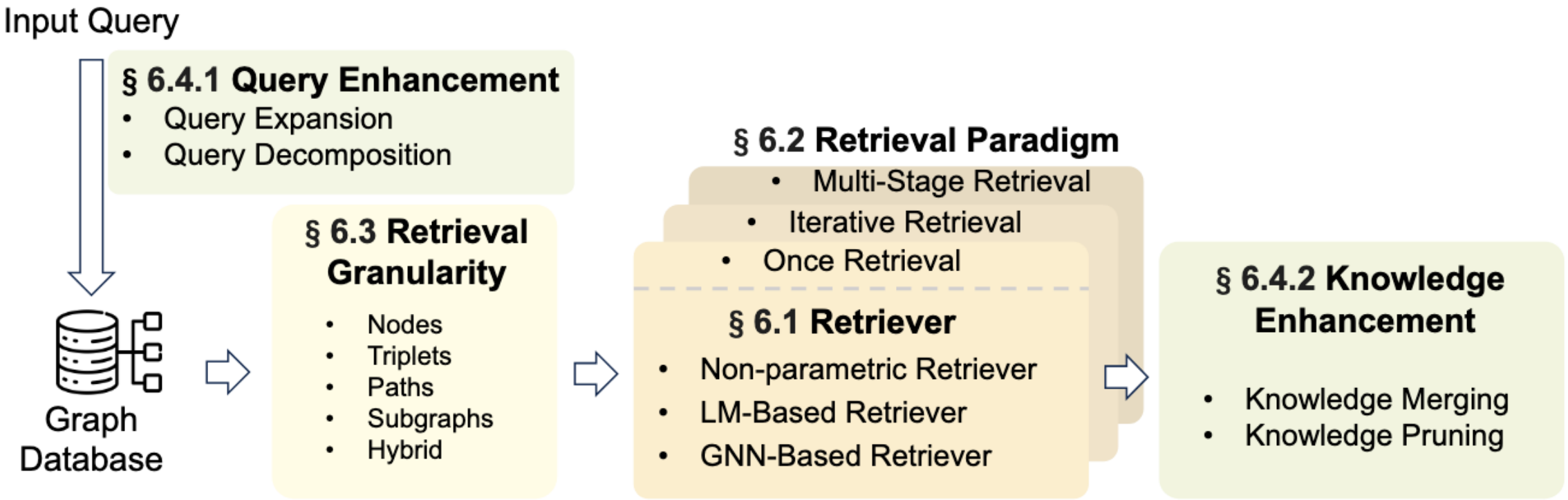


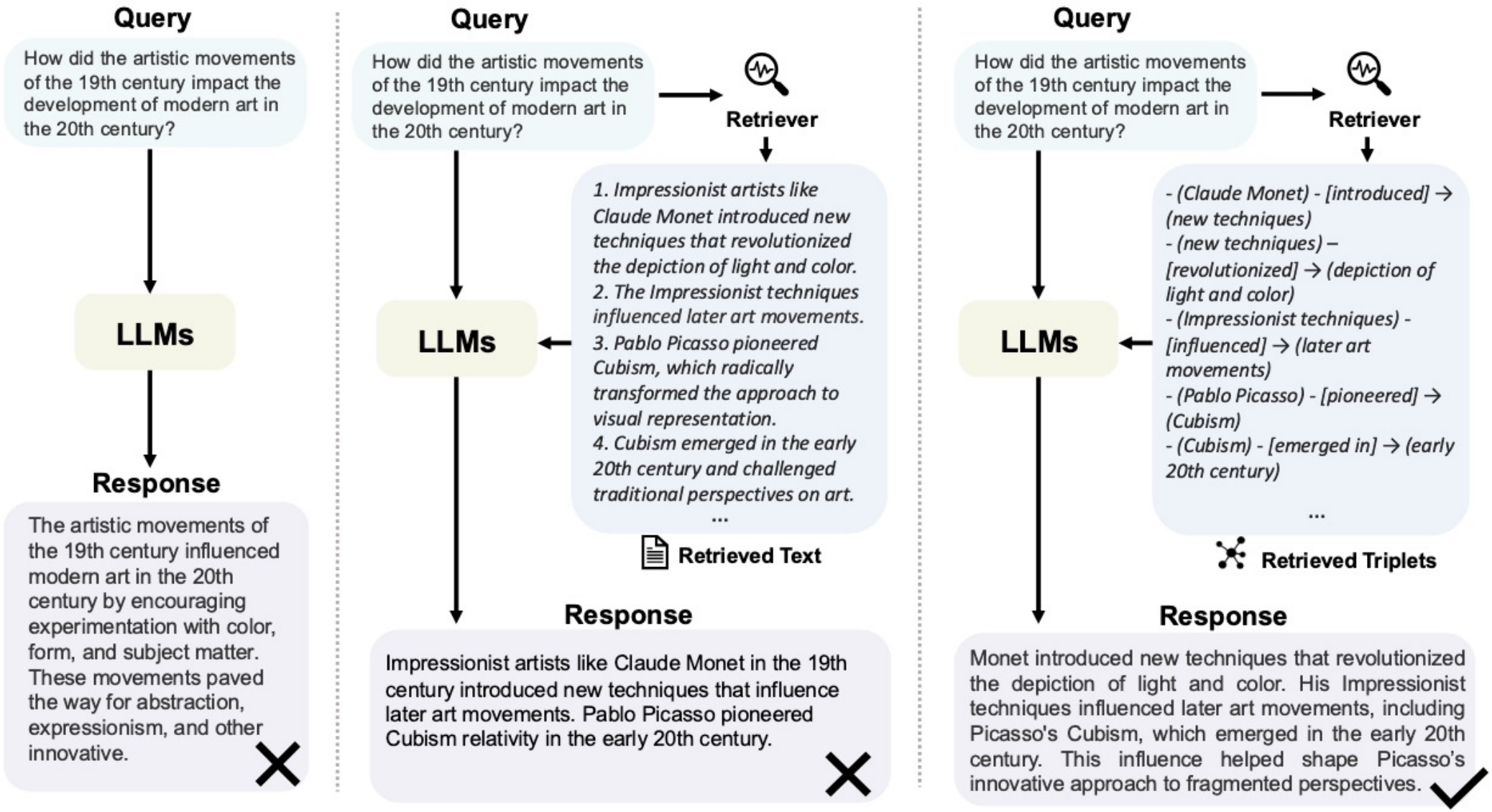

Graph RAG pipeline using an LLM-derived graph index of source document text

# Graph based Generation

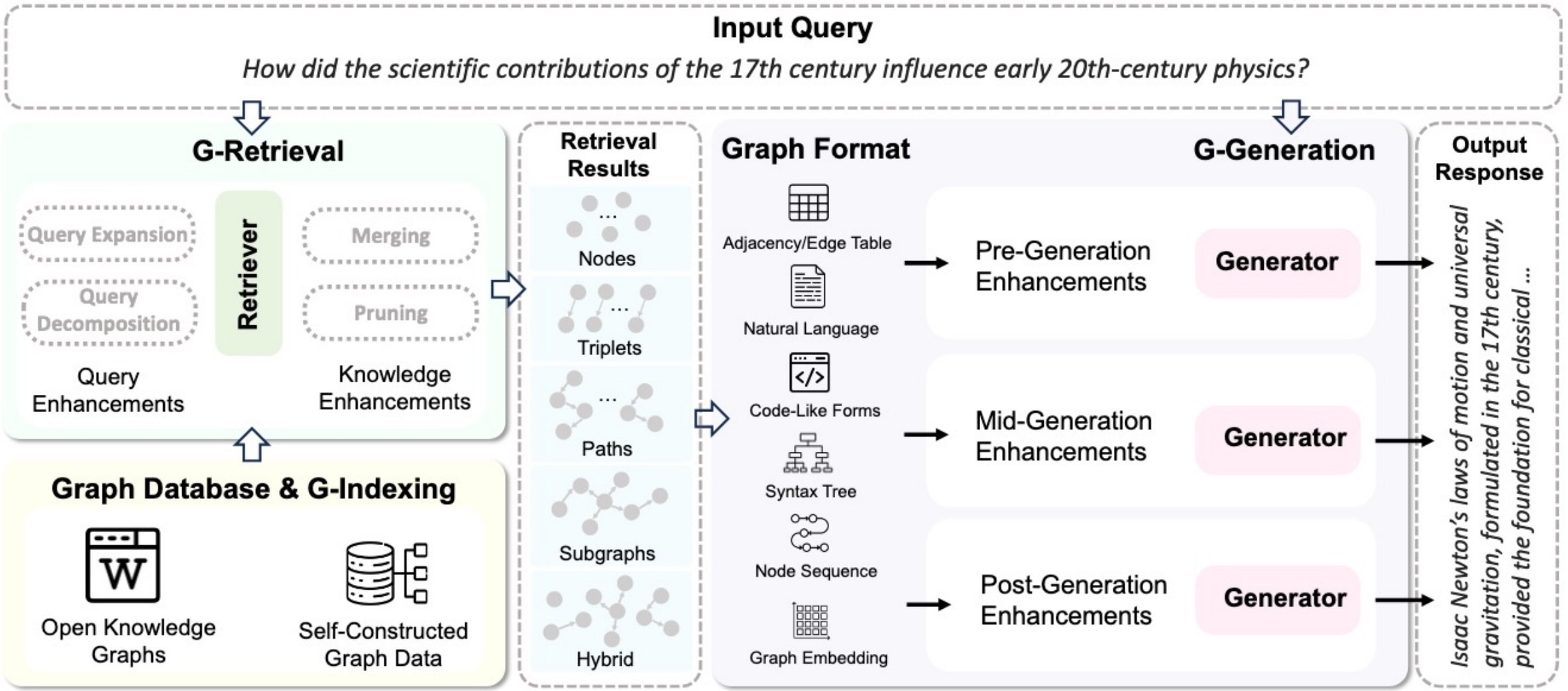


# Graph based Retrieval

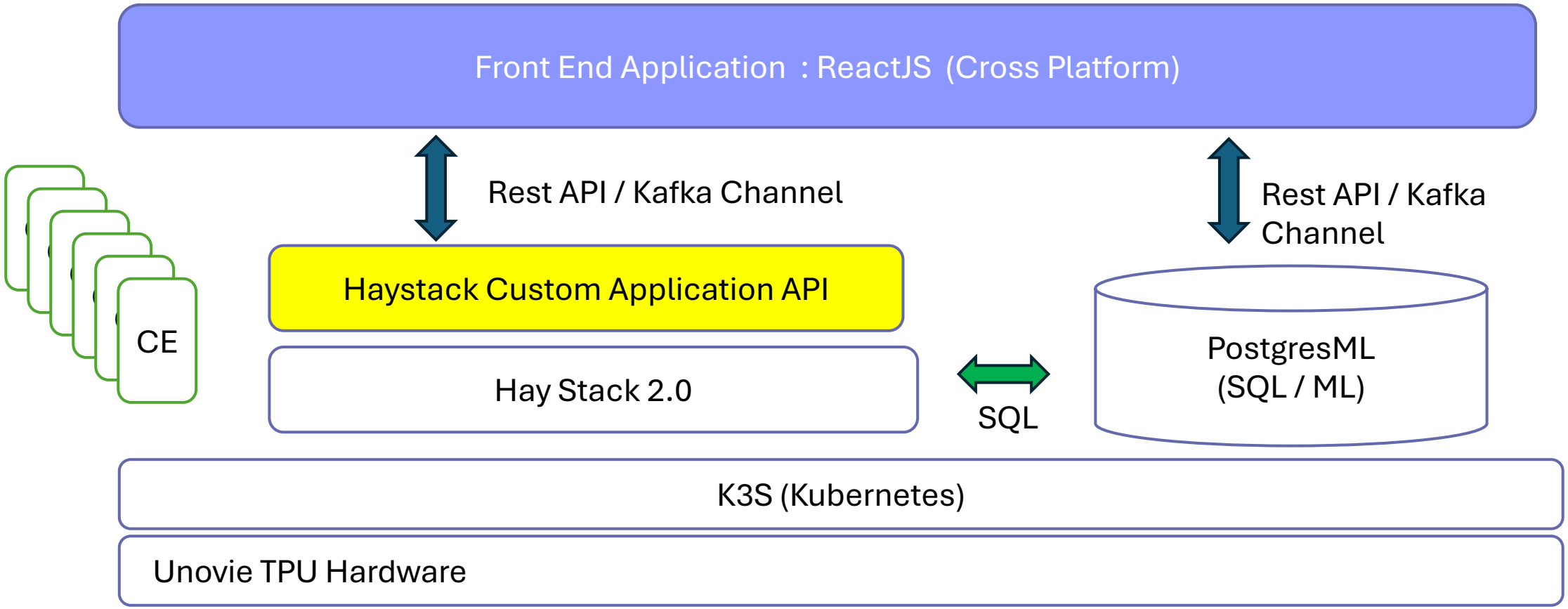



Direct LLM
RAG
GraphRAG
Query
How did the artistic movements of the 19th century impact the development of modern art in the 20th century?
LLMs
Response
The artistic movements of the 19th century influenced modern art in the 20th century by encouraging experimentation with color, form, and subject matter. These movements paved the way for abstraction, expressionism, and other innovative.
Query
How did the artistic movements of the 19th century impact the development of modern art in the 20th century?
Retriever
1. Impressionist artists like Claude Monet introduced new techniques that revolutionized the depiction of light and color.
2. The Impressionist techniques influenced later art movements.
3. Pablo Picasso pioneered Cubism, which radically transformed the approach to visual representation.
4. Cubism emerged in the early 20th century and challenged traditional perspectives on art.
...
Retrieved Text
LLMs
Response
Impressionist artists like Claude Monet in the 19th century introduced new techniques that influence later art movements. Pablo Picasso pioneered Cubism relativity in the early 20th century.
Query
How did the artistic movements of the 19th century impact the development of modern art in the 20th century?
Retriever
- (Claude Monet) - [introduced] → (new techniques)
- (new techniques) – [revolutionized] → (depiction of light and color)
- (Impressionist techniques) - [influenced] → (later art movements)
- (Pablo Picasso) - [pioneered] → (Cubism)
- (Cubism) - [emerged in] → (early 20th century)
...
Retrieved Triplets
LLMs
Response
Monet introduced new techniques that revolutionized the depiction of light and color. His Impressionist techniques influenced later art movements, including Picasso's Cubism, which emerged in the early 20th century. This influence helped shape Picasso's innovative approach to fragmented perspectives.

# GraphRAG

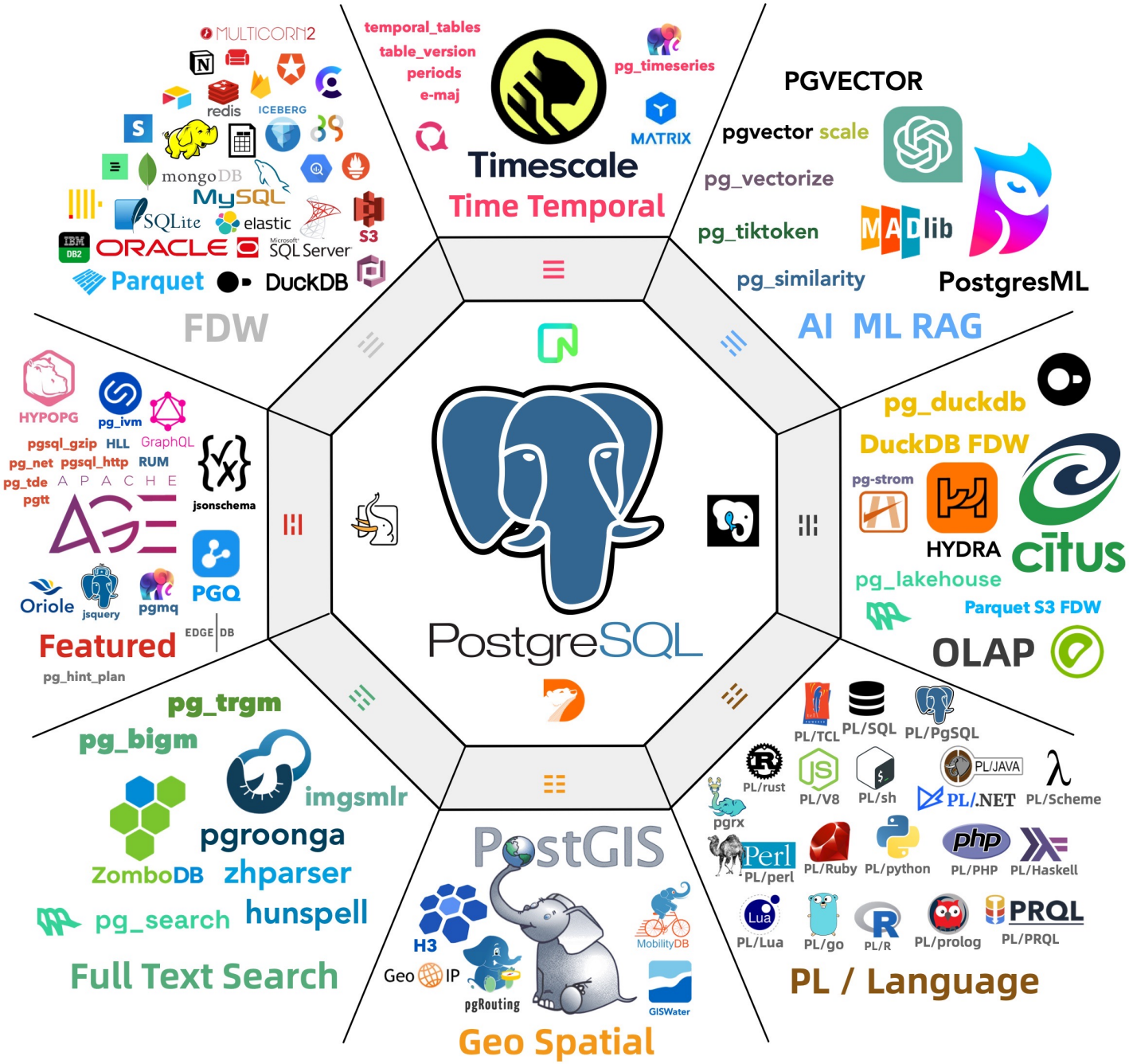

Input Query
How did the scientific contributions of the 17th century influence early 20th-century physics?
G-Retrieval
Query Expansion
Query Decomposition
Retriever
Merging
Pruning
Query Enhancements
Knowledge Enhancements
Graph Database & G-Indexing
Open Knowledge Graphs
Self-Constructed Graph Data
Retrieval Results
Nodes
Triplets
Paths
Subgraphs
Hybrid
Graph Format
Adjacency/Edge Table
Natural Language
Code-Like Forms
Syntax Tree
Node Sequence
Graph Embedding
G-Generation
Pre-Generation Enhancements
Mid-Generation Enhancements
Post-Generation Enhancements
Generator
Generator
Generator
Output Response
Isaac Newton's laws of motion and universal gravitation, formulated in the 17th century, provided the foundation for classical ...

# Production Environment

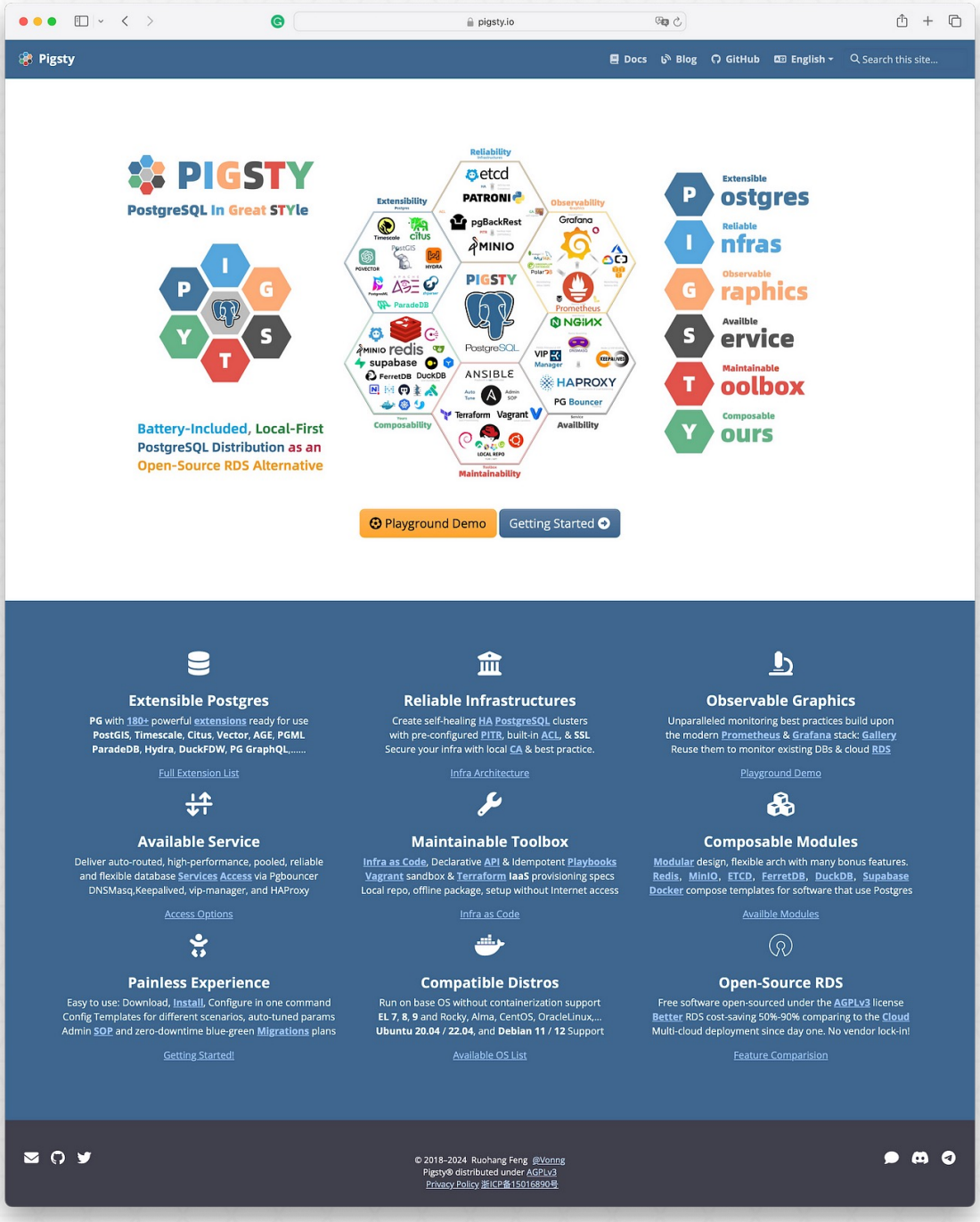


**Postgres is eating the database world**

https://medium.com/@fengruohang/postgres-is-eating-the-database-world-157c204dcfc4

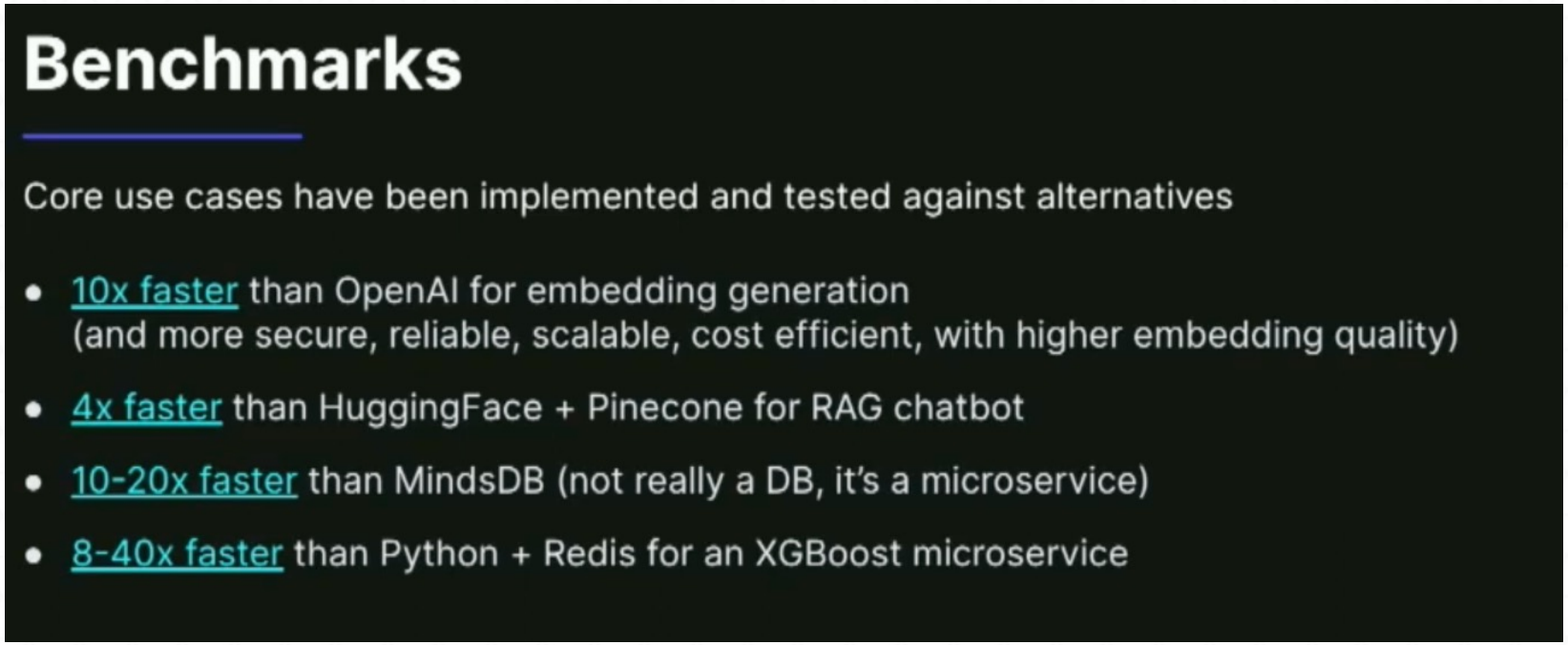



**Battery-Included, Local-First**
**PostgreSQL Distribution as an**
**Free & Better RDS Alternative**



# PostGresML

**Open-source Python Library for Training and Deploying ML Models in PostgreSQL via SQL Queries**




https://www.youtube.com/watch?v=kK3W2qpVa8E&t=3s. <<<< Watch This Video

https://medium.com/pythoneers/postgresml-open-source-python-library-for-training-and-deploying-ml-models-in-postgresql-via-sql-6fea87081ab5

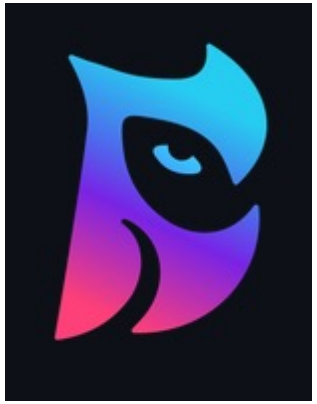
https://www.datacamp.com/tutorial/postgresml-tutorial-machine-learning-with-sql

https://www.infoq.com/presentations/ml-postgresml/

# Database requirements for interactive applications

The data layer ends up being the hard part of interactive ML & AI infrastructure

- Near real-time access, rules out data warehouses
  - High client concurrency, i.e. horizontal scalability, i.e. sharding
  - Millisecond read times
  - Session level streaming update latency (seconds)
- Machine Learning features
  - Key/Value access
  - Text, JSON, and vector indexes
  - ML models: classical, embeddings, LLMs
- Proven
  - Documented deployments at scale
  - Mature open source communities
  - Multiple hosted vendors



# PostGresML

**Open-source Python Library for Training and Deploying ML Models in PostgreSQL via SQL Queries**

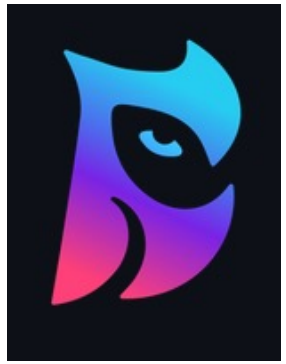
# First Principles Applied to ML

Machine Learning models are the generalized algorithm that recreate the data they are trained on, as closely as possible.

Useful models are:

1) Small relative to the data
2) Updated infrequently relative to the data

∴ Moving the algorithm to the data before inference is more efficient than moving the data to the algorithm during inference.



# PostGresML

**Open-source Python Library for Training and Deploying ML Models in PostgreSQL via SQL Queries**

# Transformers

Text Generation, Summarization, Translation, Embeddings and other "AI" stuff

## pgml.transform()

- Download pre-trained models from HuggingFace
- Serve with the same Postgres infrastructure
- GPU acceleration provided by libtorch/tensorflow.
- GGML & GPTQ for model quantization

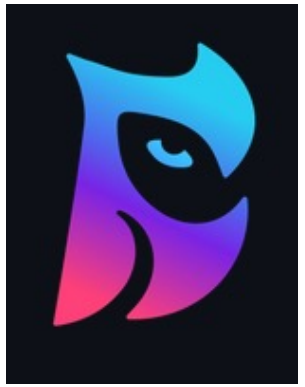
## pgml.embed()

- Higher quality open source embeddings than OpenAI
- Faster end-to-end vector database operations than OpenAI + Pinecone
- HNSW and vector operations from pgvector

## pgml.tune()

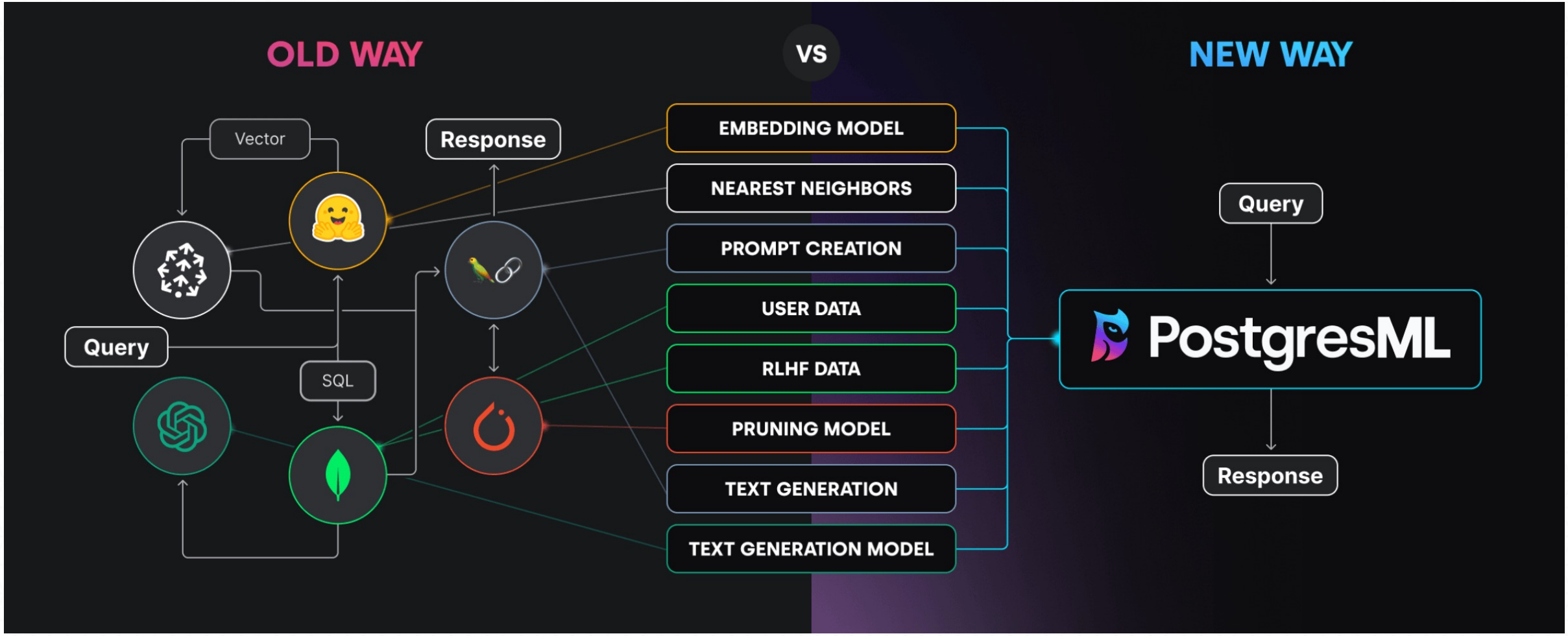
- Fine tune pre-trained models using Reinforcement Learning from Human Feedback (RLHF)
- Same workflow as pgml.train()
- LORA coming soon





# PostGresML

**Open-source Python Library for Training and Deploying ML Models in PostgreSQL via SQL Queries**




**4x Faster**

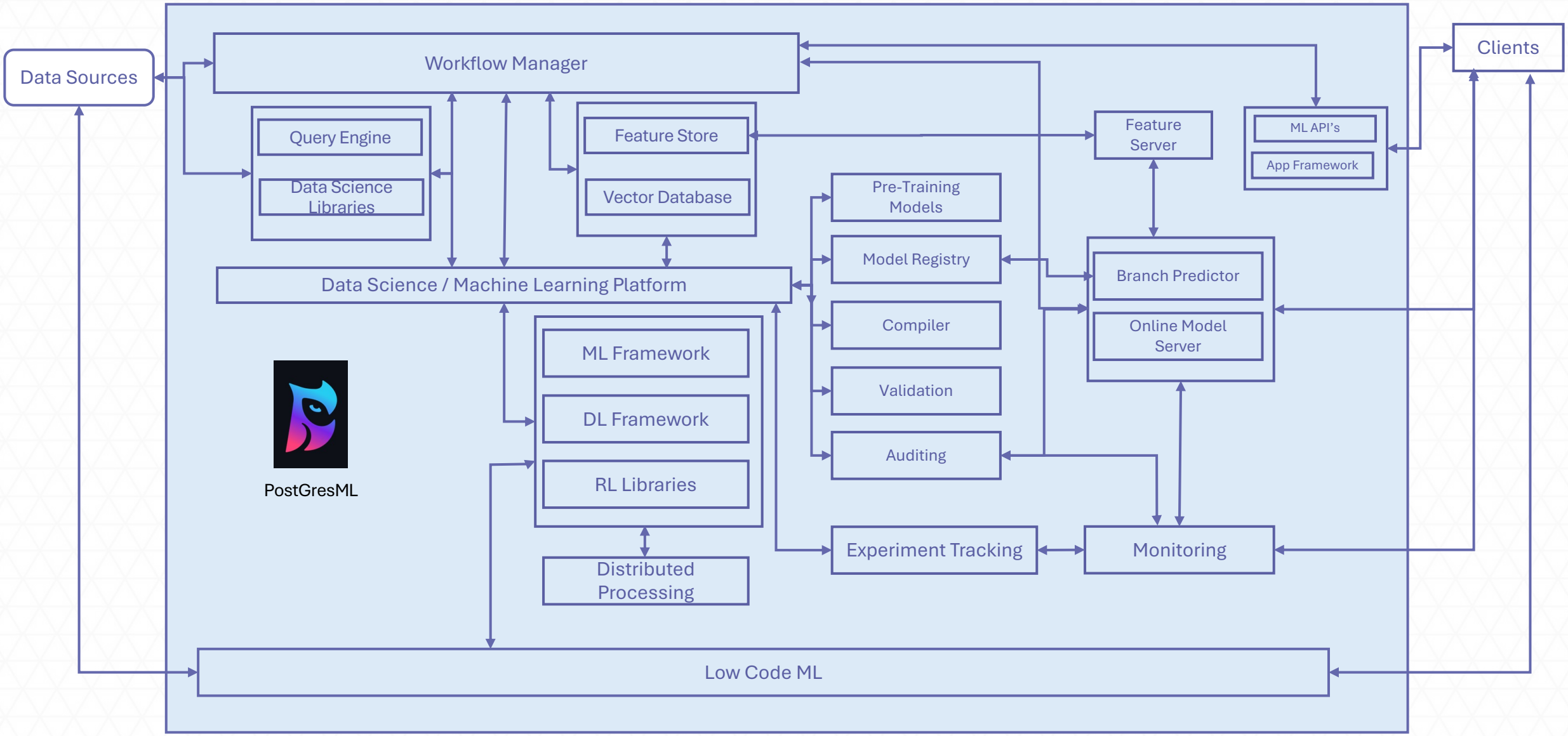
than HuggingFace + Pinecone for a RAG chatbot

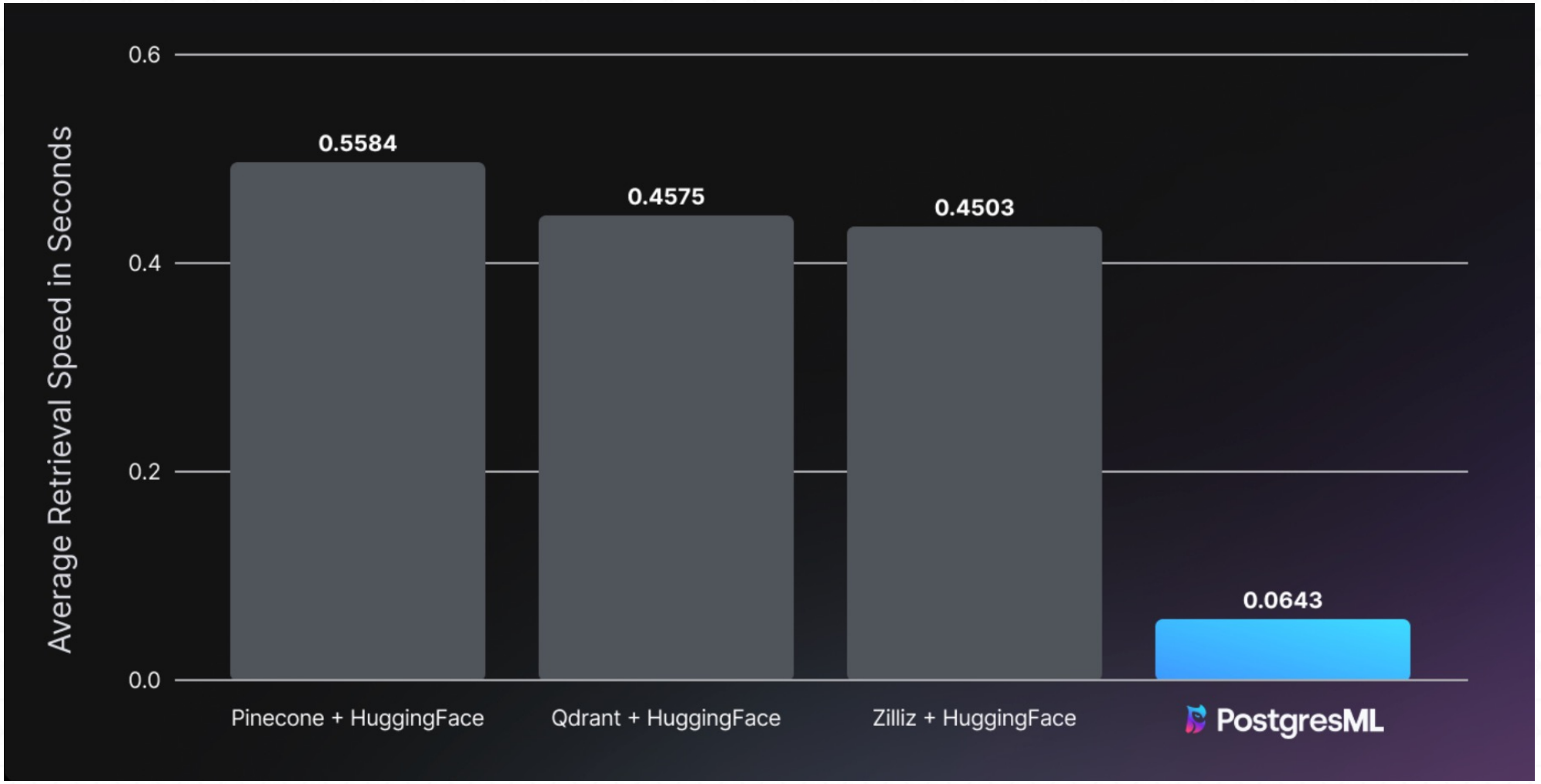
**10x faster**

than OpenAI for embedding generation

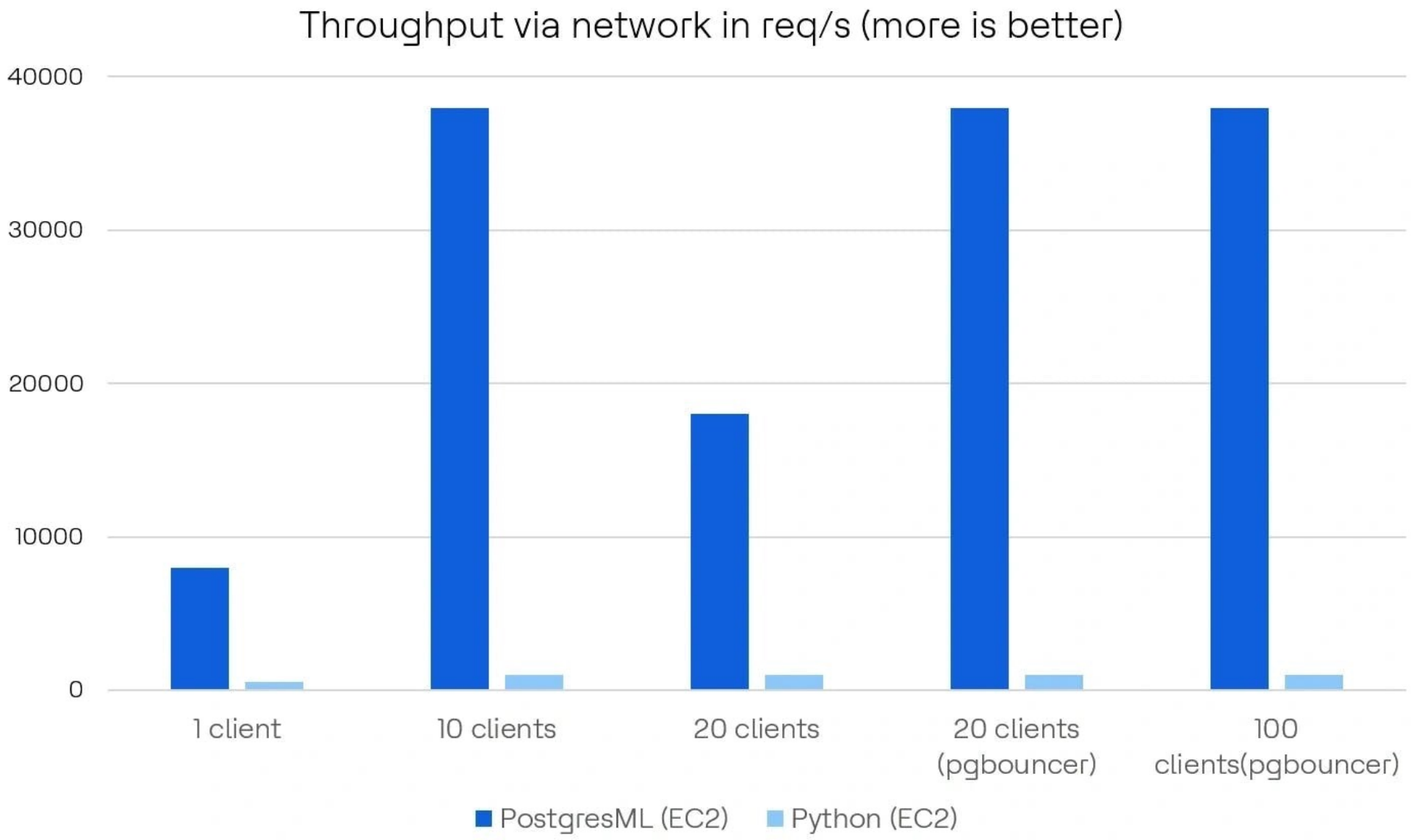
**Save 42%**

On vector database cost compared to Pinecone



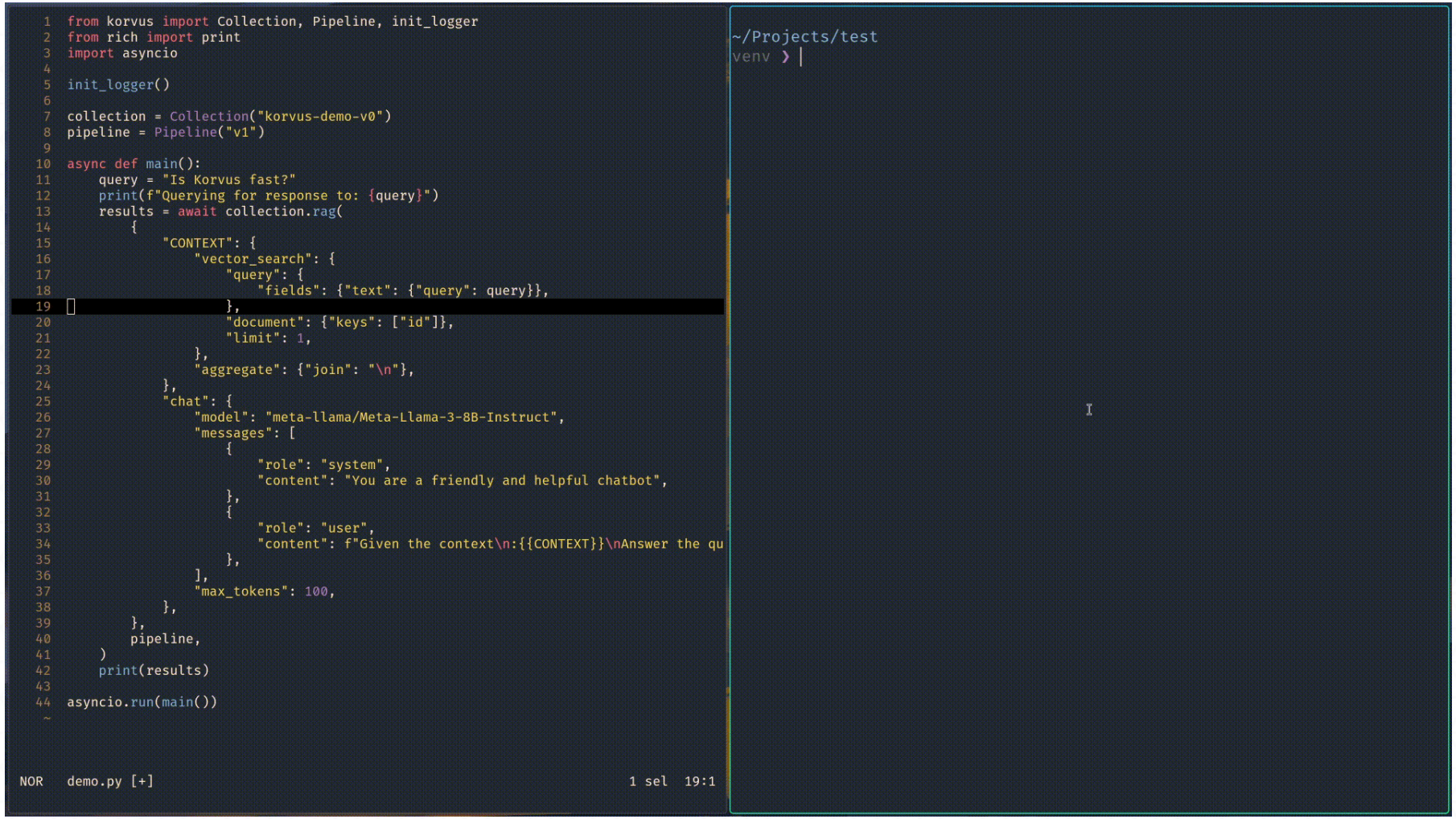



The average retrieval speed for RAG in seconds



# Korvus

Korvus is an all-in-one, open-source RAG (Retrieval-Augmented Generation) pipeline built for Postgres. It combines LLMs, vector memory, embedding generation, reranking, summarization and custom models into a single query, maximizing performance and simplifying your search architecture.

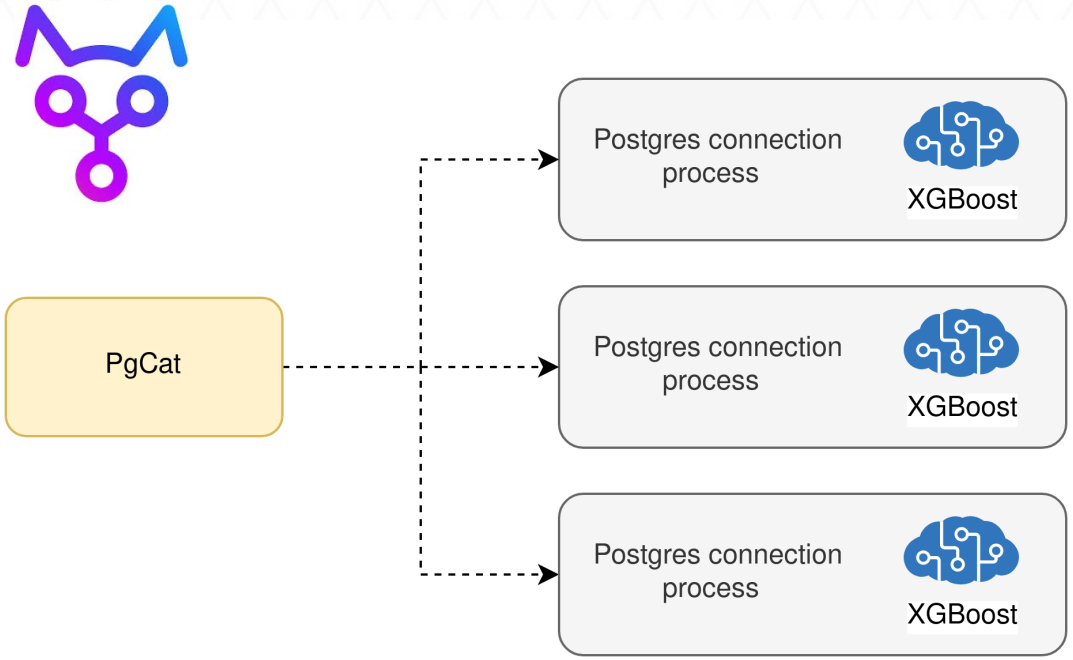


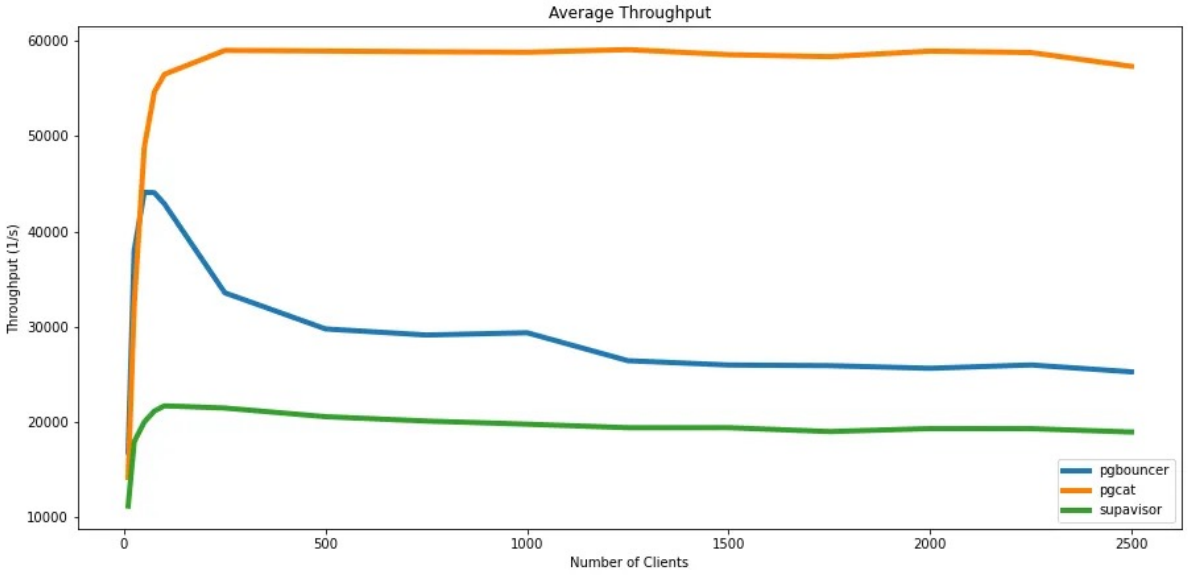
Korvus stands out by harnessing the full power of Postgres for RAG operations:

- **Postgres-Native RAG**: Korvus leverages Postgres' robust capabilities, allowing you to perform complex RAG operations directly within your database. This approach eliminates the need for external services and API calls, significantly reducing latency and complexity many times over.
- **Single Query Efficiency**: With Korvus, your entire RAG pipeline - from embedding generation to text generation - is executed in a single SQL query. This "one query to rule them all" approach simplifies your architecture and boosts performance.
- **Scalability and Performance**: By building on Postgres, Korvus inherits its excellent scalability and performance characteristics. As your data grows, Korvus grows with it, maintaining high performance even with large datasets.

## ⚡ Key Features

- **Simplified Architecture**: Replace complex service oriented architectures with a single, powerful query.
- **High Performance**: Eliminates API calls and data movement for faster processing and greater reliability.
- **Open Source**: Improve your developer experience with open source software and models that run locally in Docker too.
- **Multi-Language Support**: Use Korvus with Python, JavaScript and Rust. Open an issue to vote for other language support.
- **Unified Pipeline**: Combine embedding generation, vector search, reranking, and text generation in one query.
- **Postgres-Powered**: Under the hood, Korvus operations are powered by efficient SQL queries on a time-tested database platform.




https://tembo.io/blog/postgres-connection-poolers

**Scaling PostgresML to 1 Million Requests per Second**

Addressing horizontal scalability concerns, we've benchmarked PostgresML and ended up with an incredible 1 million requests per second using commodity hardware.

| | PgBouncer | PgCat | Supavisor |
|---|---|---|---|
| Max Concurrent Clients Tested | 2500 | 2500 | 2500 |
| Max Throughput | 44,096 tps @ 50 clients | 59,051 tps @ 1,250 clients | 21,708 tps @ 100 clients |
| Latency @ 1,250 concurrent clients | 47.2 ms | 21.1 ms | 64.37 ms |

https://postgresml.org/blog/scaling-postgresml-to-1-million-requests-per-second

{guidance}

https://github.com/guidance-ai/guidance

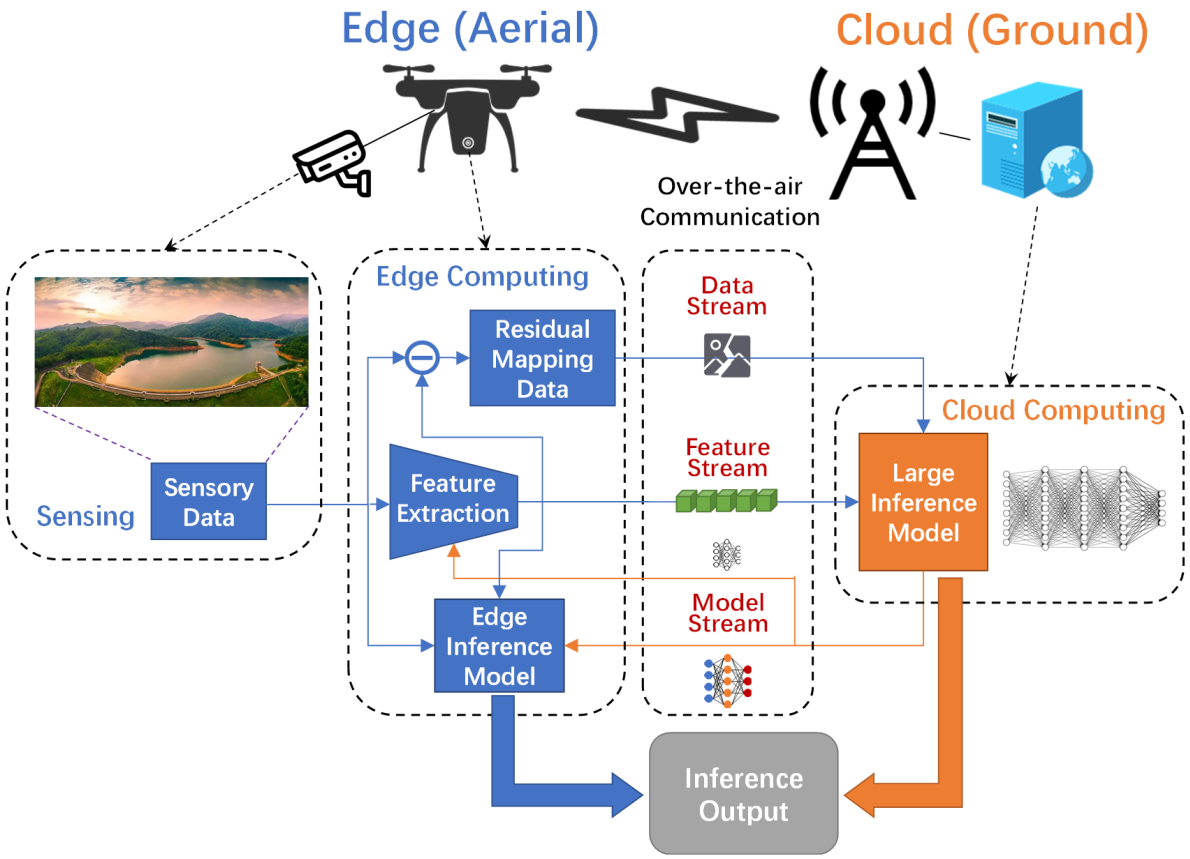
**Guidance is an efficient programming paradigm for steering language models.**

Guidance is a proven open-source Python library for controlling outputs of any language model (LM). With one API call, you can express (in Python) the precise programmatic constraint(s) that the model must follow and generate the structured output in JSON, Python, HTML, SQL, or any structure that the use case requires.

Guidance differs from conventional prompting techniques. It enforces constraints by steering the model token by token in the inference layer, producing higher quality outputs and reducing cost and latency by as much as 30–50% when utilizing for highly structured scenarios.

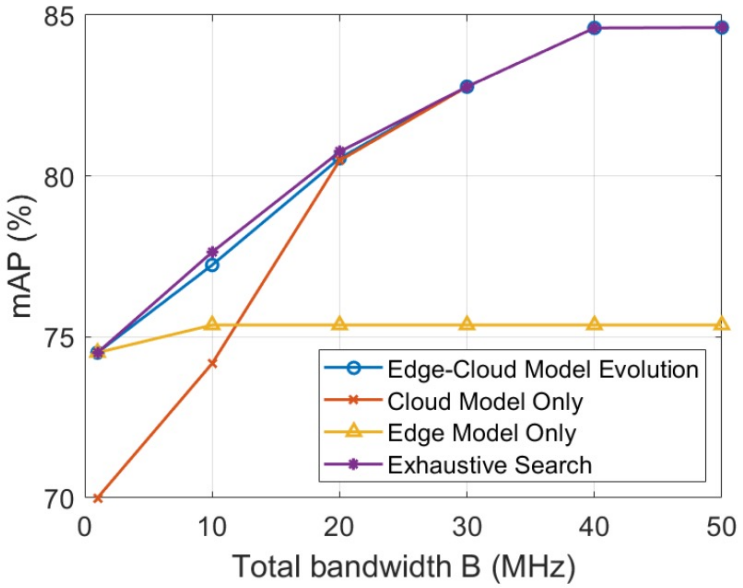
https://github.com/microsoft/Phi-3CookBook/blob/main/code/01.Introduce/guidance.ipynb

# Large Models for Aerial Edges: An Edge-Cloud Model Evolution and Communication Paradigm (paper : Aug 2024)




## Simulation Parameters

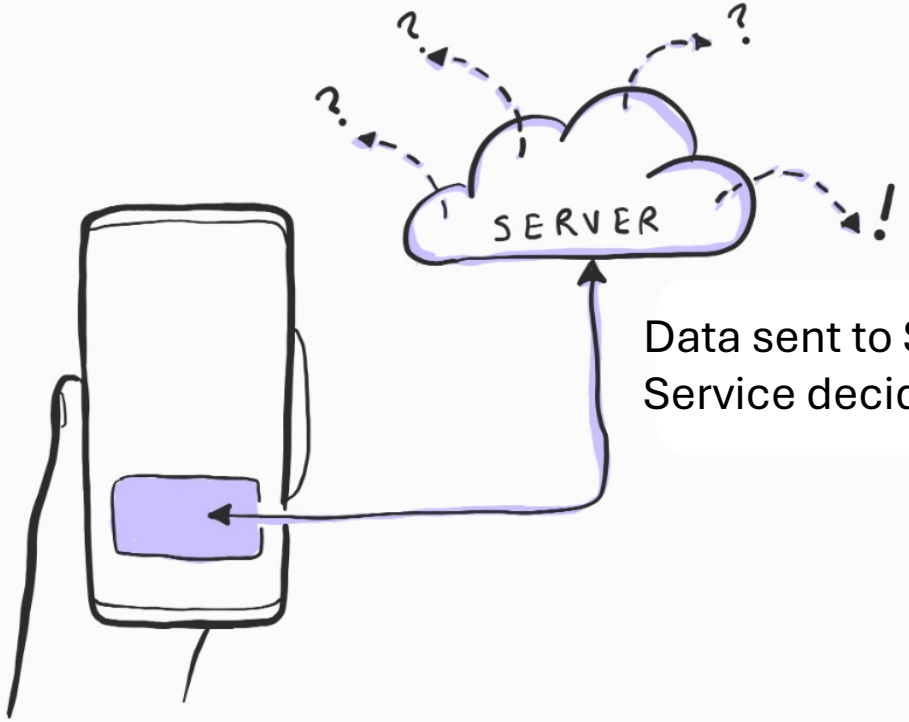
| Parameter | Value |
|---|---|
| Number of sensing frames generated per second $N$ | 10 |
| Number of pixels per frame $x$ | $10^7$ |
| Average data size of the extracted feature $\bar{F}$ | 0.86kbps |
| OTA bandwidth $B$ | 10 MHz |
| Uplink spectrum efficiency $S_u$ | 2.55 bit/s/Hz |
| Downlink spectrum efficiency $S_d$ | 5 bit/s/Hz |
| Maximum model update overhead $M_{max}$ | 230 kbps |
| Minimum model update overhead $M_{min}$ | 23 Mbps |



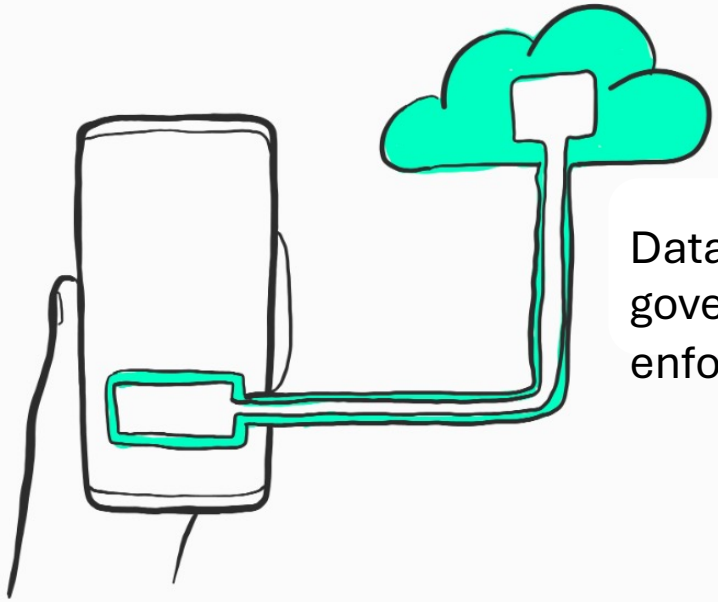

1) The performance gain of the proposed framework stems from dynamically adjusting the mAP of the edge model and the cloud model via model evolution and data uploading, so as to maximize the overall mAP of the framework.
2) The edge model handles the majority of tasks with small communication bandwidth and large data size, where most of the bandwidth is allocated to small model updating.
3) The cloud model handles the majority of tasks with large communication bandwidth and small data size, with most of the bandwidth allocated to residual mapping data uploading.

mAP : Mean Average Precision

# Google’s Project OAK




Data sent to Server.
Service decides on use



Data Stays within Enclave, governed by hardware enforced policy

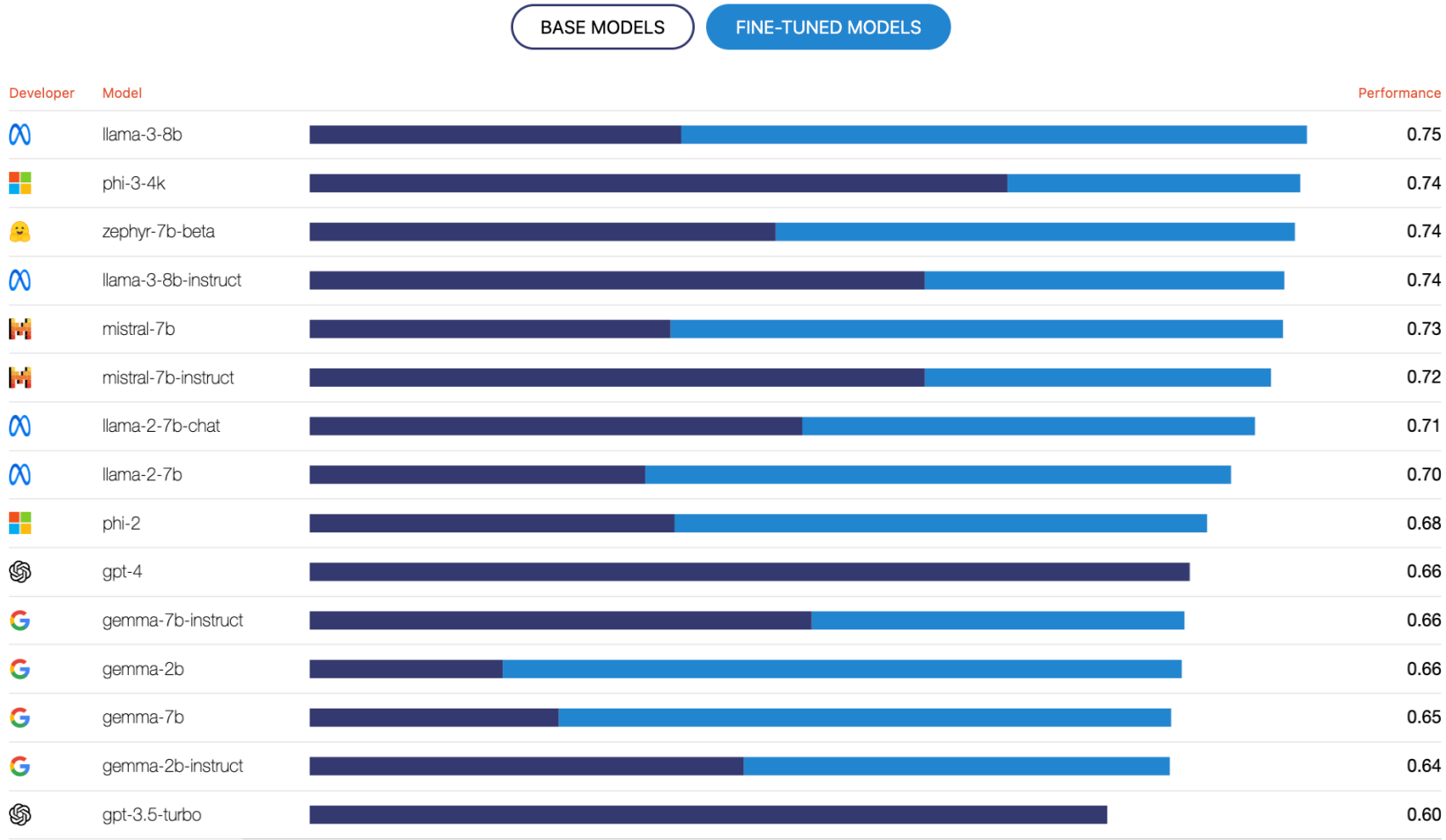
**Now**

Once personal data leaves your device, you lose control of what happens to it. Leaving you exposed to potential misuse

**With Enclaves**

Personal Data remain in Enclaves obeying the rules that you control, resetting the user: service value exchange.

The Fine-tuning Leaderboard shows the performance of each model aggregated across 31 distinct tasks. You can evaluate the performance pre and post fine-tuning by selecting the base or fine-tuned model button at the top. Remarkably, most of the fine-tuned open-source models surpass GPT-4 with Llama-3, Phi-3 and Zephyr demonstrating the strongest performance.

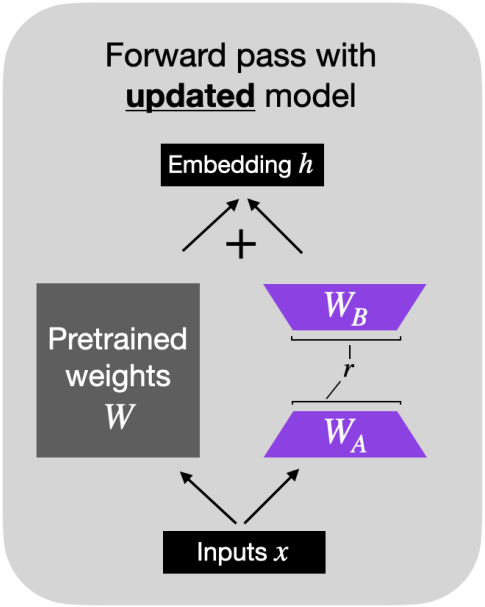



https://predibase.com/fine-tuning-index

PEFT (parameter-efficient fine-tuning) is a method for training large language models (LLMs) by updating only a small number of parameters during training.

LoRA (Low-rank Adaptation) is a popular PEFT technique that uses low-rank decomposition to **reduce the number of trainable parameters in LLMs**. This reduction in parameters makes fine-tuning more efficient and practical, with lower memory consumption and reduced computational and storage costs

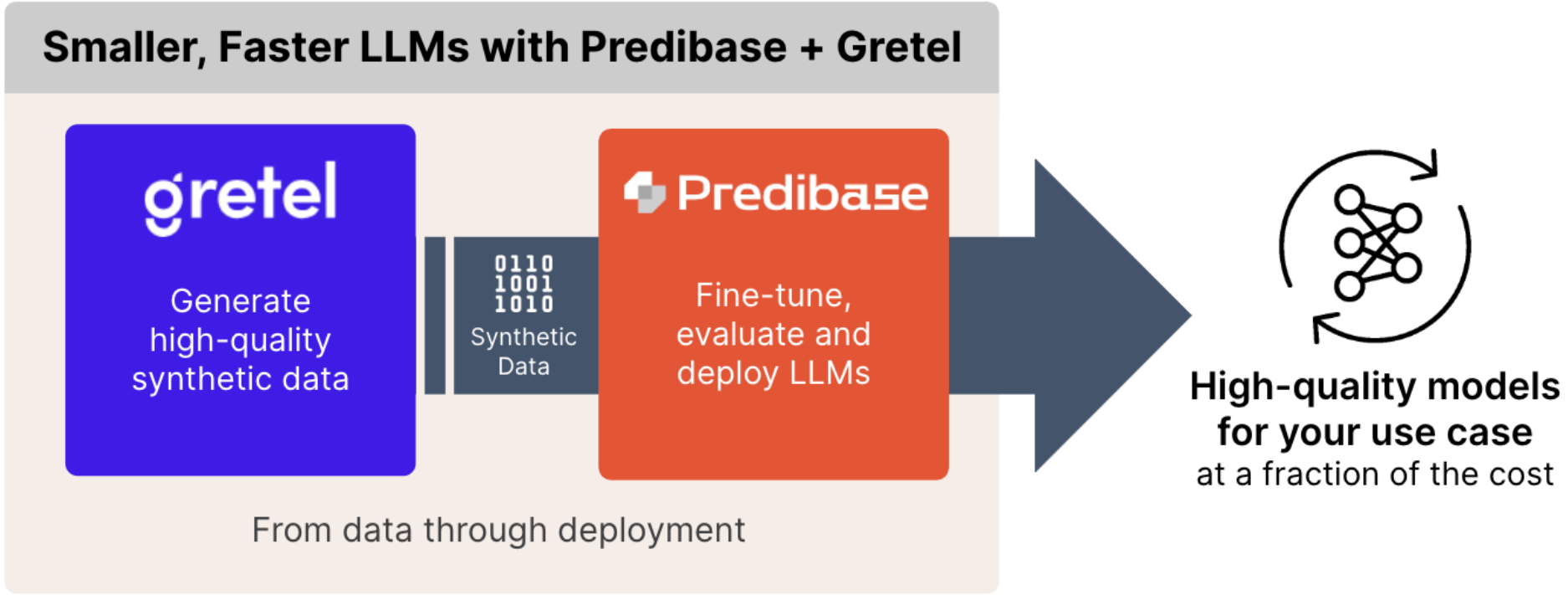
LoRA weights, $W_A$ and $W_B$, represent $\Delta W$




LoRA's approach is to freeze the original weights and introduce new parameters into the model to train through. It does this by decomposing weight matrices into two smaller matrices, which can be trained to adapt to new data while keeping the overall number of changes low.

For example, a weight updation matrix of 200 x 3 and 3 x 500 can be decomposed into 2100 trainable parameters, which is only 2.1% of the total number of parameters.

LoRA can also be applied to specific layers only, further reducing the number of parameters. As a result, the files can be as small as 8MB, making it much easier and faster to load, apply, and transfer the learned models.




https://predibase.com/blog/how-to-create-an-sql-copilot-by-fine-tuning-llms-with-synthetic-data

https://gretel.ai/videos/how-to-generate-synthetic-data-with-gretel

https://github.com/mlabonne/llm-datasets

*High-quality datasets, tools, and concepts for LLM fine-tuning.*

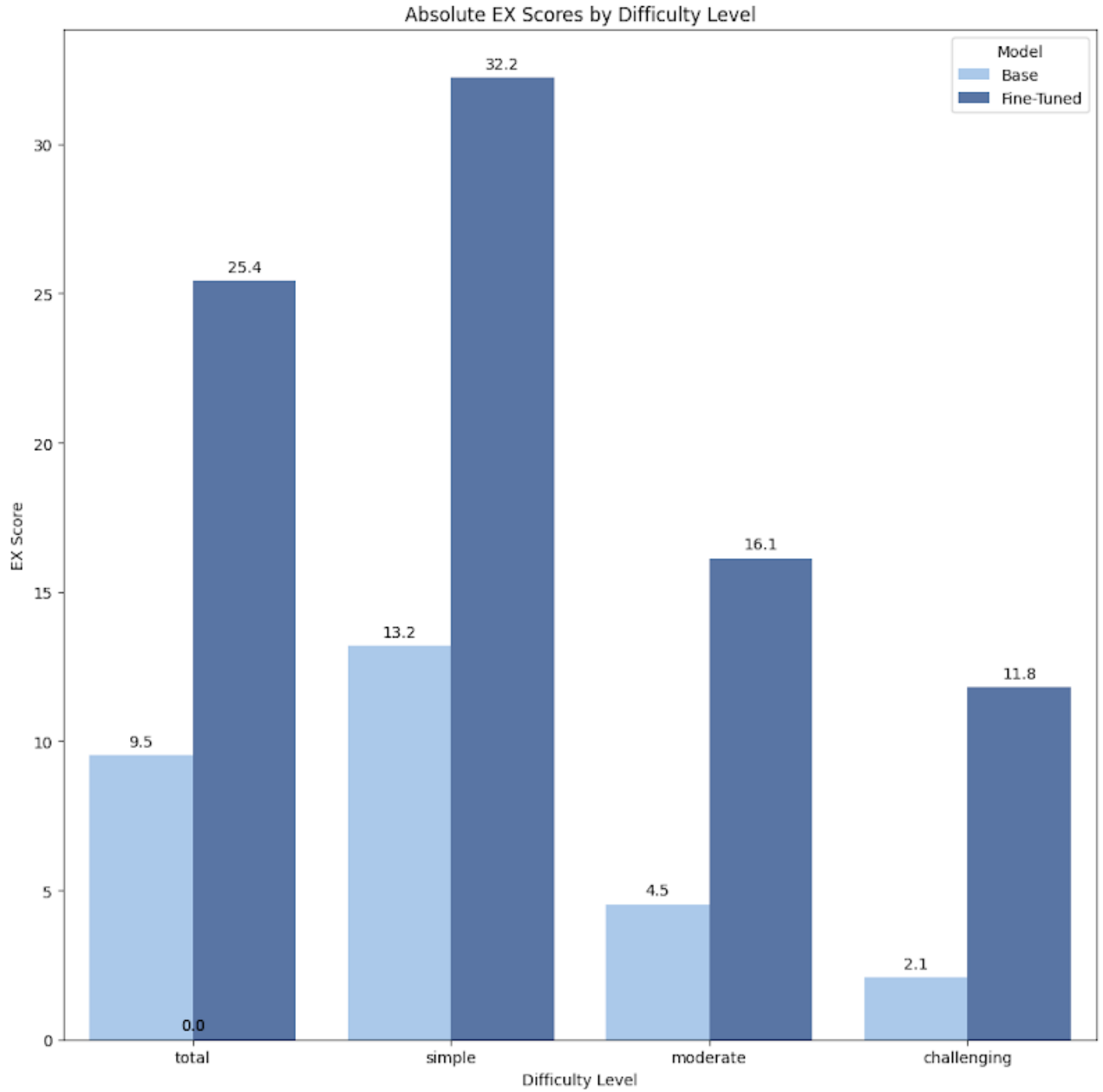
👍 **What is a good dataset?**

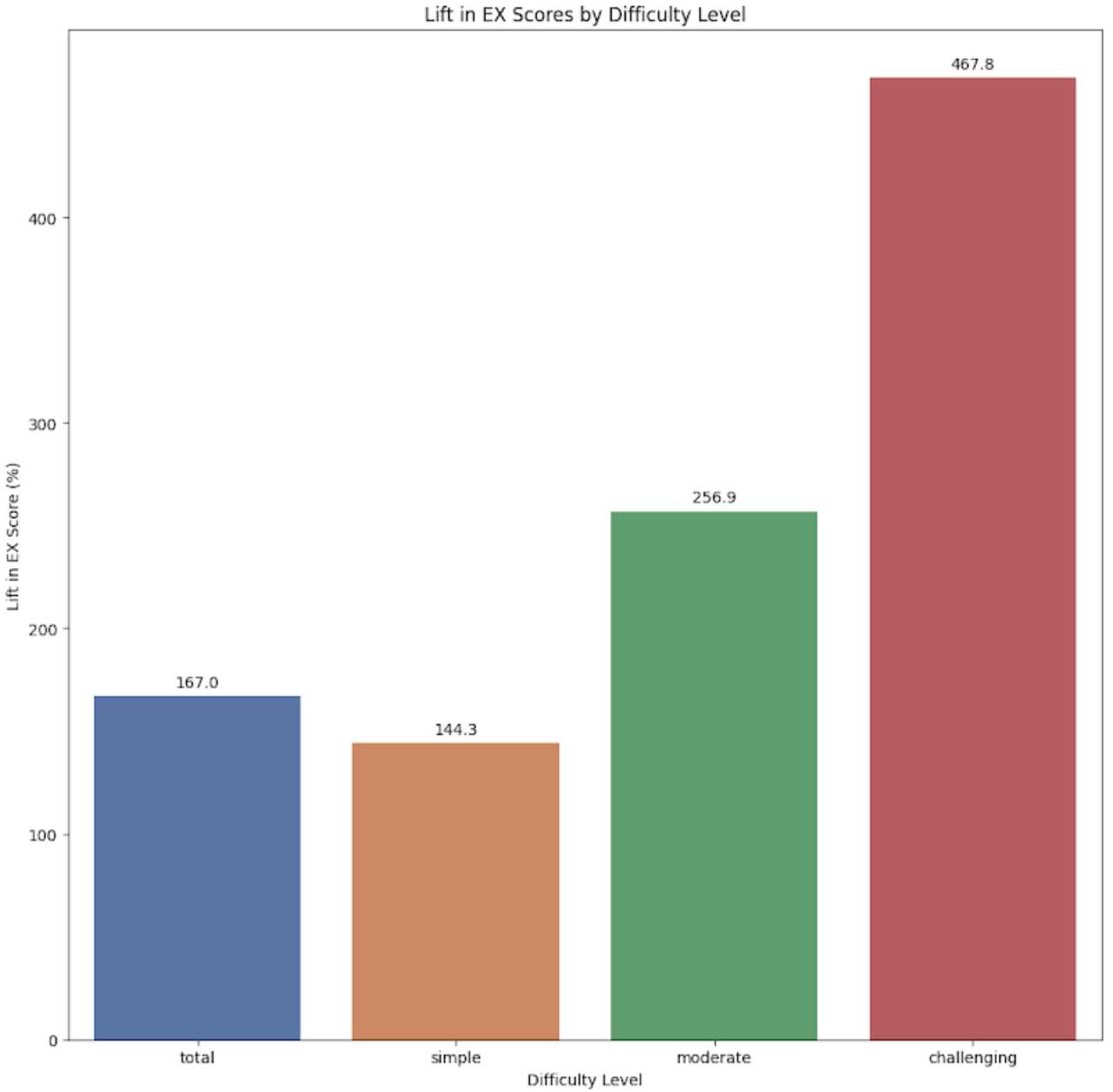
Data is the most valuable asset in LLM development. While datasets can't be directly evaluated like models, high-quality datasets have the following characteristics:

- **Accuracy**: Samples should be factually correct, helpful to users, and well-written. Answers should also be relevant to their corresponding instructions.
- **Diversity**: You want to cover as many use cases as possible to ensure proper instruction-following and relevant answers. This requires a wide range of topics, contexts, lengths, writing styles, etc. sampled in a representative way.
- **Complexity**: Answers should be nontrivial and a/ representative of tasks you expect the model to handle or b/ include complex tasks involving multi-step reasoning, planning, etc.

Measuring accuracy can be easy in the case of mathematical problems using a Python interpreter, or near-impossible with open-ended, subjective questions. On the other hand, clustering datasets by topic is a good way of measuring diversity. Finally, complexity can be assessed using other LLMs acting like judges.

A Llama-3-8B model fine-tuned on a subset of the synthetic text-to-SQL data outperforms the base model by a wide margin on the BIRD-SQL benchmark, delivering an overall lift of 167% and a lift of more than 467% on challenging questions.



LoRAX (LoRA eXchange) is a framework that allows users to serve thousands of fine-tuned models on a single GPU, dramatically reducing the cost of serving without compromising on throughput or latency.

- 🚅 **Dynamic Adapter Loading:** include any fine-tuned LoRA adapter from HuggingFace, Predibase, or any filesystem in your request, it will be loaded just-in-time without blocking concurrent requests. Merge adapters per request to instantly create powerful ensembles.
- 🤹 **Heterogeneous Continuous Batching:** packs requests for different adapters together into the same batch, keeping latency and throughput nearly constant with the number of concurrent adapters.
- 🧁 **Adapter Exchange Scheduling:** asynchronously prefetches and offloads adapters between GPU and CPU memory, schedules request batching to optimize the aggregate throughput of the system.
- 🧑‍🤝‍🧑 **Optimized Inference:** high throughput and low latency optimizations including tensor parallelism, pre-compiled CUDA kernels (flash-attention, paged attention, SGMV), quantization, token streaming.
- 🚢 **Ready for Production** prebuilt Docker images, Helm charts for Kubernetes, Prometheus metrics, and distributed tracing with Open Telemetry. OpenAI compatible API supporting multi-turn chat conversations. Private adapters through per-request tenant isolation. Structured Output (JSON mode).
- 🤯 **Free for Commercial Use:** Apache 2.0 License. Enough said

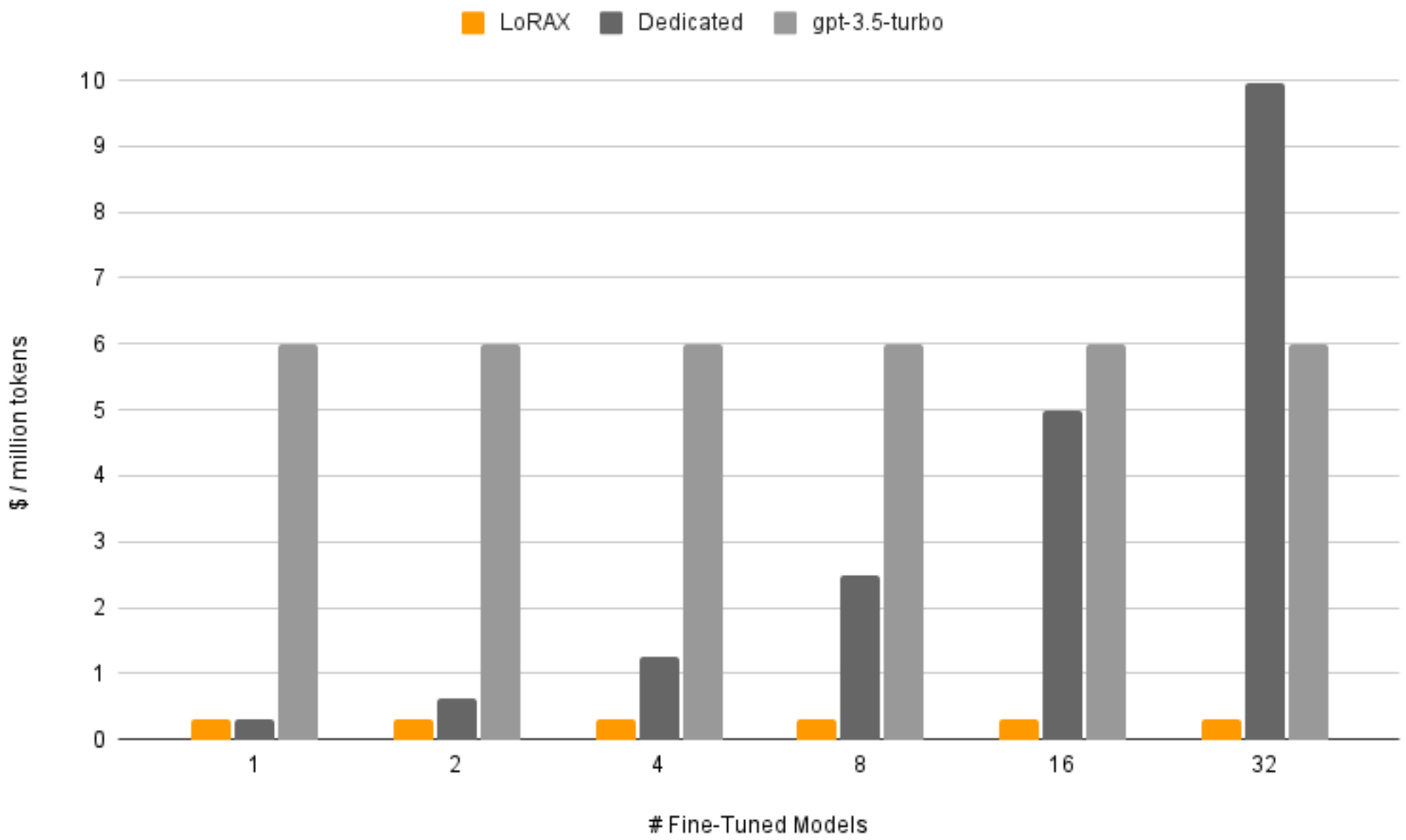
Serving a fine-tuned model with LoRAX consists of two components:

- Base Model: pretrained large model shared across all adapters.
- Adapter: task-specific adapter weights dynamically loaded per request.

LoRAX supports a number of Large Language Models as the base model including Llama (including CodeLlama), Mistral (including Zephyr), and Qwen. See Supported Architectures for a complete list of supported base models.

Base models can be loaded in fp16 or quantized with bitsandbytes, GPT-Q, or AWQ.

Supported adapters include LoRA adapters trained using the PEFT and Ludwig libraries. Any of the linear layers in the model can be adapted via LoRA and loaded in LoRAX.



pip install lorax-client

```
from lorax import Client

client = Client("http://127.0.0.1:8080")

# Prompt the base LLM
prompt = "[INST] Natalia sold clips to 48 of her friends in April, and then she sold half as many clips
in May. How many clips did Natalia sell altogether in April and May? [/INST]"
print(client.generate(prompt, max_new_tokens=64).generated_text)

# Prompt a LoRA adapter
adapter_id = "vineetsharma/qlora-adapter-Mistral-7B-Instruct-v0.1-gsm8k"
print(client.generate(prompt, max_new_tokens=64, adapter_id=adapter_id).generated_text)
```

https://loraexchange.ai/

# SGLang: LMSys New Framework for Super Fast LLM Inference

SGLang is a fast serving framework for large language models and vision language models. It makes your interaction with models faster and more controllable by co-designing the backend runtime and frontend language.

The core features include:

- **Fast Backend Runtime**: Efficient serving with RadixAttention for prefix caching, jump-forward constrained decoding, continuous batching, token attention (paged attention), tensor parallelism, flashinfer kernels, and quantization (AWQ/FP8/GPTQ/Marlin).

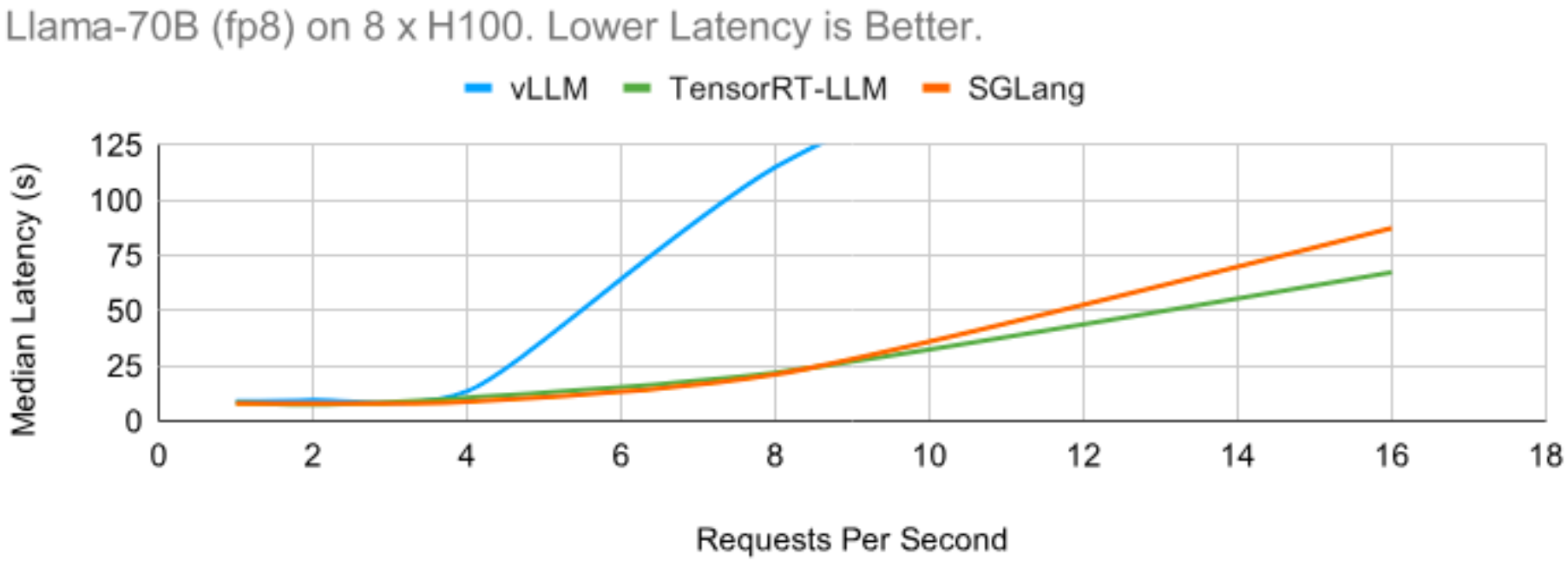
- **Flexible Frontend Language**: Enables easy programming of LLM applications with chained generation calls, advanced prompting, control flow, multiple modalities, parallelism, and external interactions.

SGLang tackles a set of known challenges in LLM applications with a fresh approach. When we think about the elements There are several areas where LLM programming can be improved:

- **Caching:** In LLM programs, caching the computed KV cache from previous tokens can minimize repeated calculations when multiple text segments and generation calls are involved.
- **Batching:** Since LLMs are primarily memory-bound, increasing batch sizes can significantly boost throughput. Employing contiguous batching techniques is also beneficial.
- **Sharing:** LLM programs often need to generate multiple outputs from a single prompt or branch out to a new prompt. Developing more sophisticated sharing patterns can enhance efficiency.
- **Parallelism:** By creating a dependency graph for generation calls within an LLM program, independent calls can be executed simultaneously, enhancing parallelism within the program.
- **Compilation:** Full programs can be compiled into an optimized intermediate representation for more efficient execution. Aggressive optimizations, such as adjusting prompts based on test cases, can further enhance performance.

## Achieving Faster Open-Source Llama3 Serving with SGLang Runtime (vs. TensorRT-LLM, vLLM)

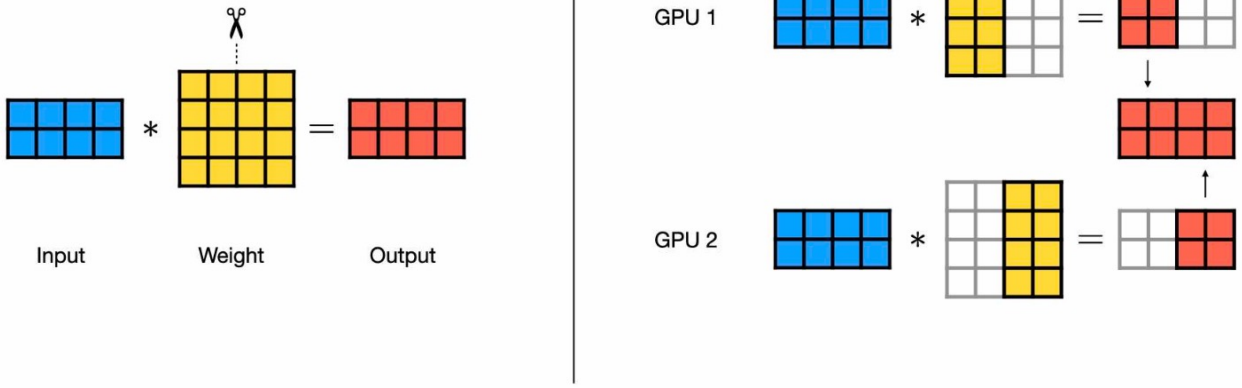
The SGLang Team, Jul 25, 2024




| | SGLang | TensorRT-LLM | vLLM |
|---|---|---|---|
| Performance | Excellent | Excellent | Fair |
| Usability | Good | Poor | Good |
| Customizability | High | Low | Medium |
| Source Code Availability | Fully Open | Partially Open | Fully Open |
| Programming Language | Python | C++ | Python |

https://aflah02.substack.com/p/how-sglang-saved-me-days-of-time

# Tensor Parallelism



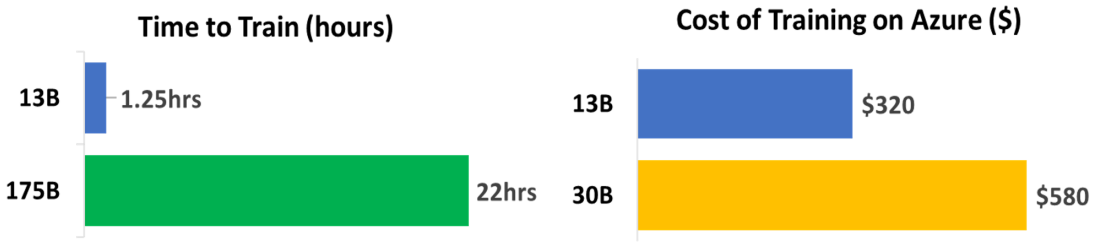

- Add --tp 2 to enable tensor parallelism. If it indicates peer access is not supported between these two devices, add --enable-p2p-check option.
  - python -m sglang.launch_server --model-path meta-llama/Meta-Llama-3-8B-Instruct --port 30000 --tp 2
- Add --dp 2 to enable data parallelism. It can also be used together with tp. Data parallelism is better for throughput if there is enough memory.
  - python -m sglang.launch_server --model-path meta-llama/Meta-Llama-3-8B-Instruct --port 30000 --dp 2 --tp 2

Documentation : https://huggingface.co/docs/transformers/v4.15.0/en/parallelism

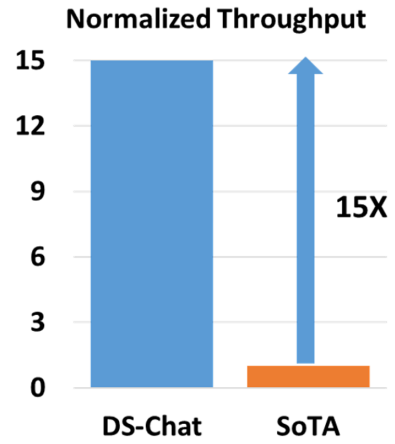
SGLANG : https://github.com/sgl-project/sglang/tree/main?tab=readme-ov-file#supported-models

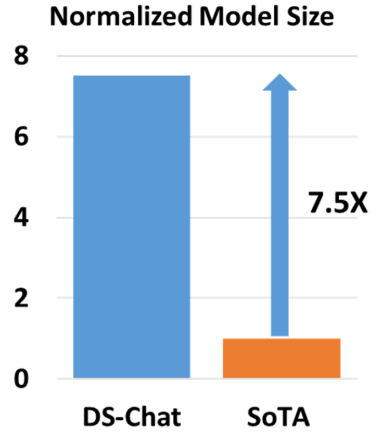
# DEEPSPEED CHAT

## Fast Training with Affordable Cost




## Train 15X Faster and Scale to 5x Bigger Models than SOTA RLHFs




**Easy-Breezy Training**

A complete end-to-end RLHF training experience with a single click

**High Performance System**

Hybrid Engine achieves 15X training speedup over SOTA RLHF systems with unprecedented cost reduction at all scales

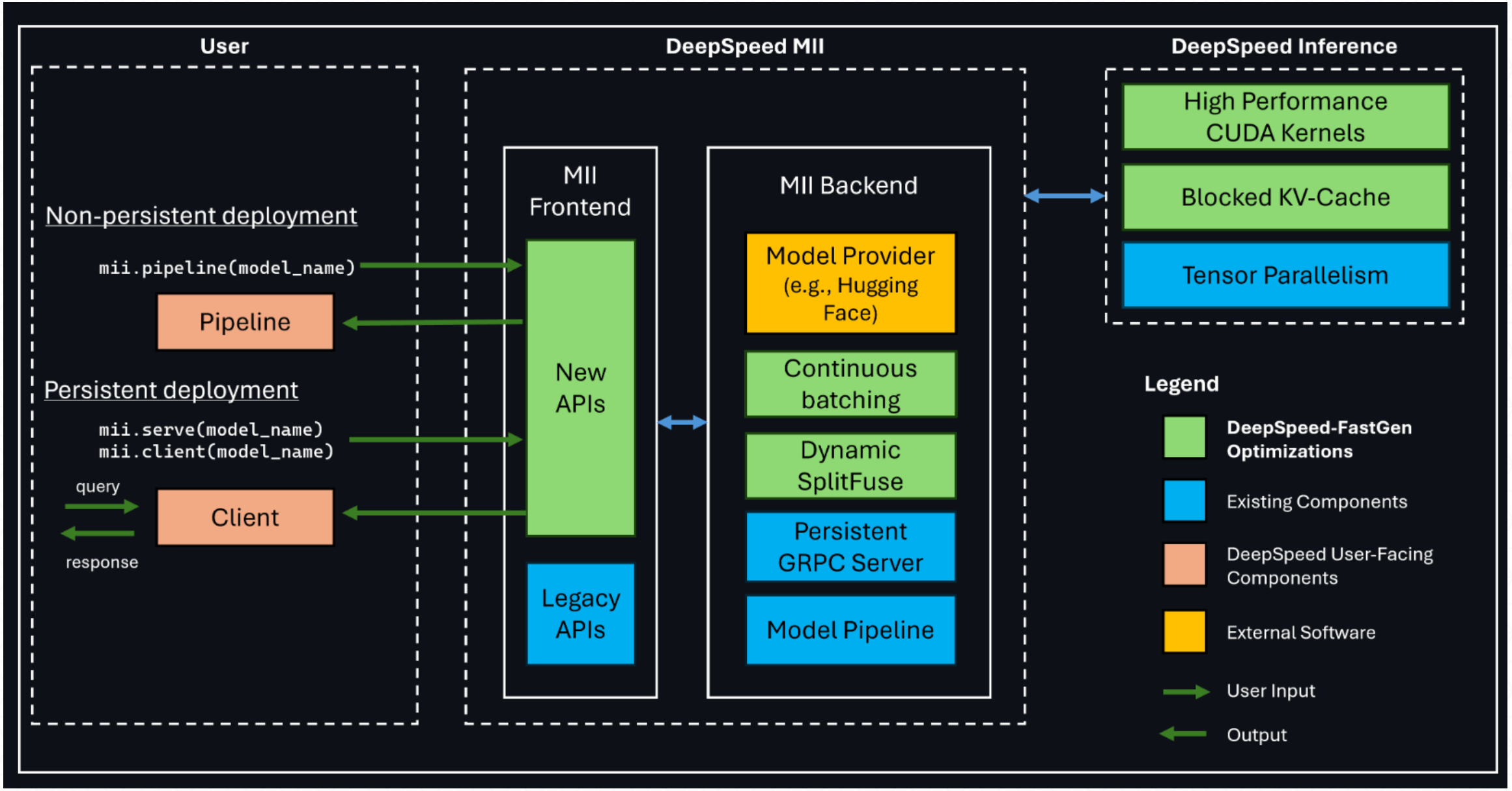
**Accessible Large Model Support**

Training ChatGPT-Style models with tens to hundreds of billions parameters on a single or multi-GPUs through ZeRO and LoRA

**A Universal Acceleration Backend for RLHF**

Support InstructGPT pipeline and large-model finetuning for various models and scenarios

https://www.deepspeed.ai/



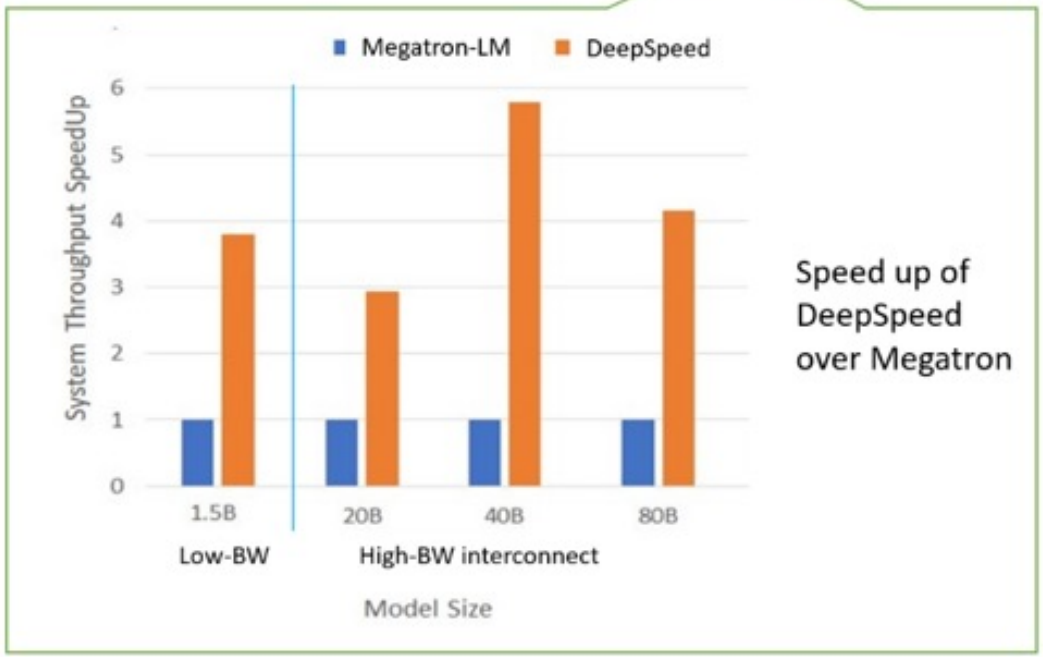

https://github.com/DefTruth/Awesome-LLM-Inference

## Scale
- 100B parameter
- 10X bigger

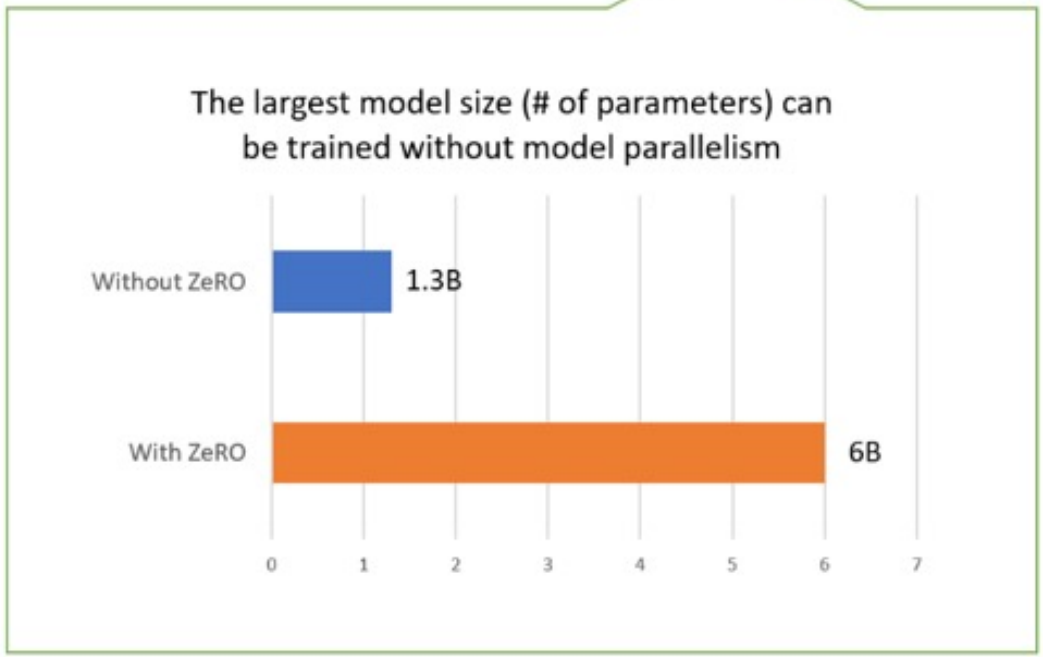
## Speed
- Up to 5X faster

## Cost
- Up to 5X cheaper

## Usability
- Minimal code change



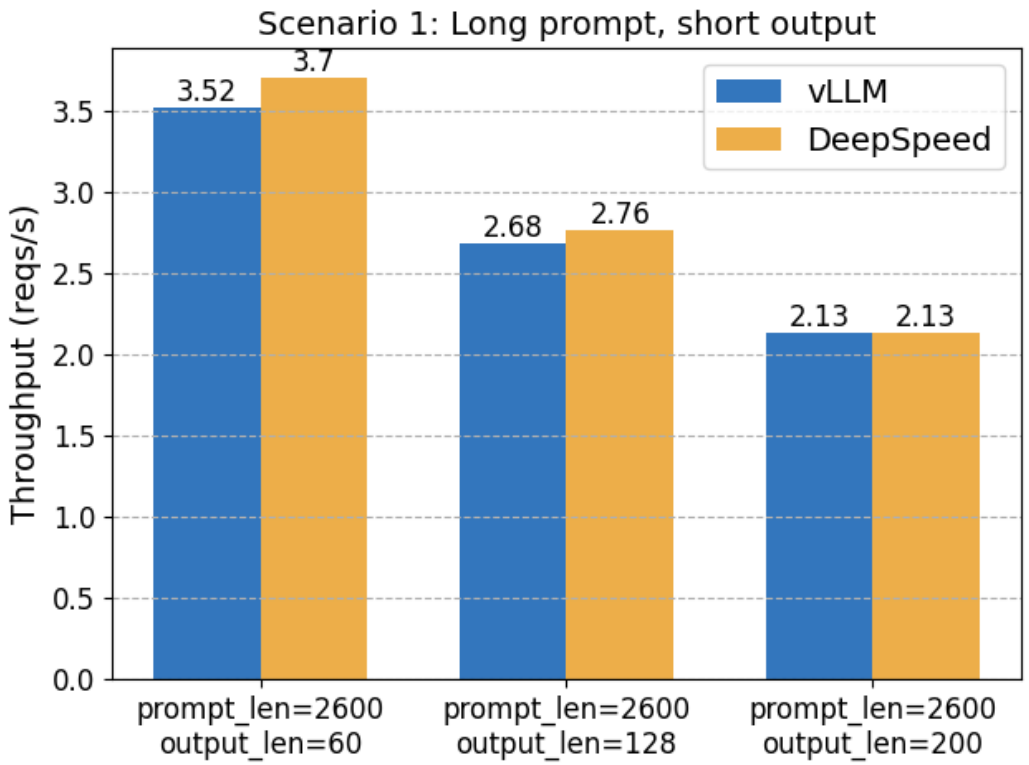

Speed up of DeepSpeed over Megatron

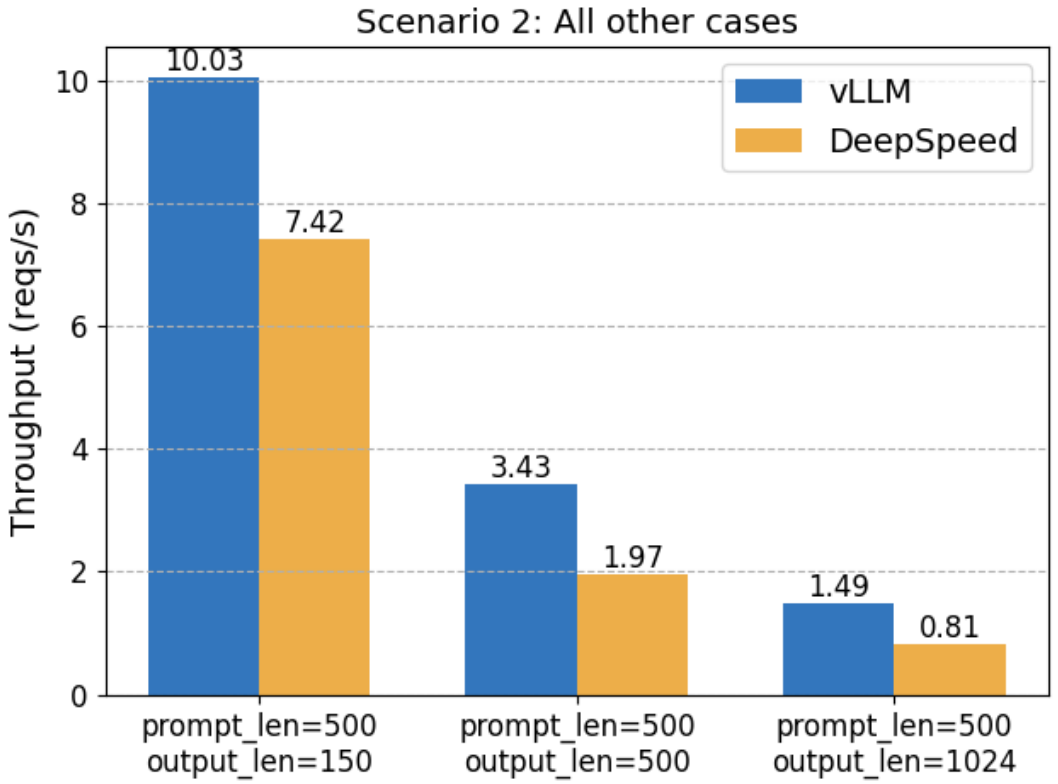



**DeepSpeed empowers ChatGPT-like model training with a single click, offering 15x speedup over SOTA RLHF systems with unprecedented cost reduction at all scales**

https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

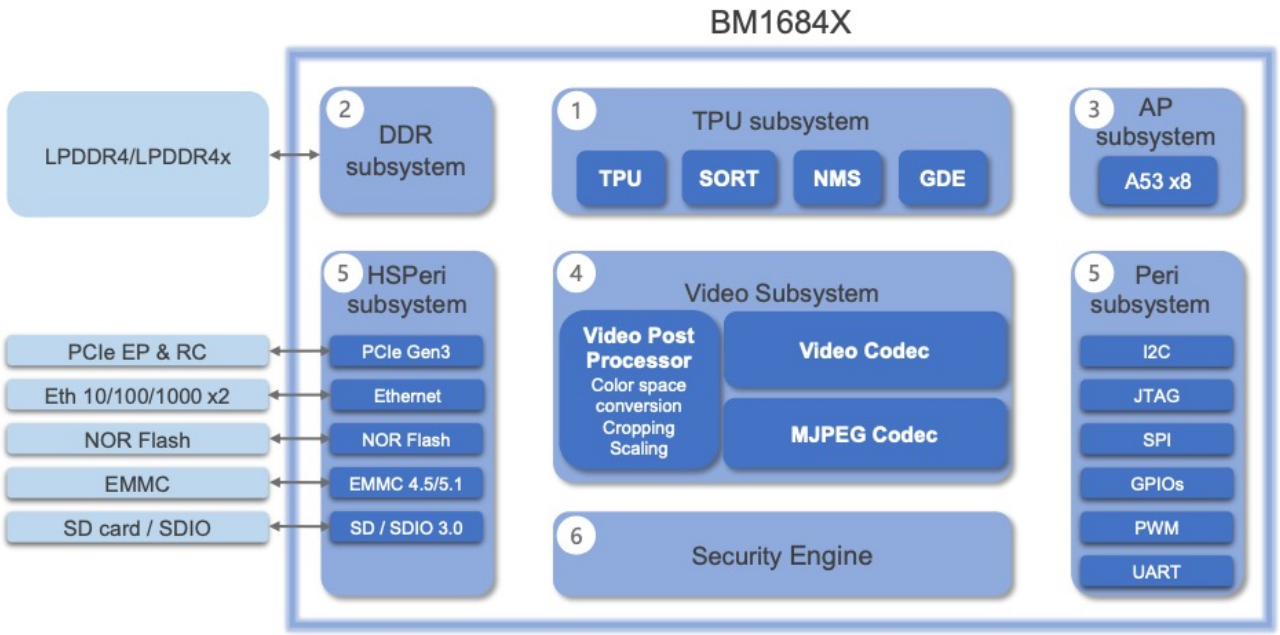
# vLLM's Future: A True Community Project




Coming out of UC Berkeley Sky Computing Lab, we are building vLLM truly in open source with the Apache 2.0 license.

| | v4 | v5e | v5p |
|---|---|---|---|
| Chips per pod | 4096 | 256 | 8,960 |
| Chip Bf16 TFLOPs | 275 | 197 | 459 |
| Chip Int8 TOPs | N/A | 394 | 918 |
| HBM (GB) | 32 | 16 | 95 |
| HBM BW (GB/s) | 1228 | 820 | 2,765 |
| ICI BW per chip (Gb/s) | 2,400 | 1,600 | 4,800 |




The emerging Utility network uses the BM1684x TPU chip from Sophon, suitable for various AI applications. It excels in natural language processing, facial recognition, image generation, video structuring, and audio recognition, focusing on AI inference for cloud and edge with high computational performance.

The BM1684x chip, featuring 64 NPUs with a total of 1024 Execution Units, supports mainstream machine learning frameworks like PyTorch, ONNX, and TensorFlow, emphasizing high performance and energy efficiency.

Source link