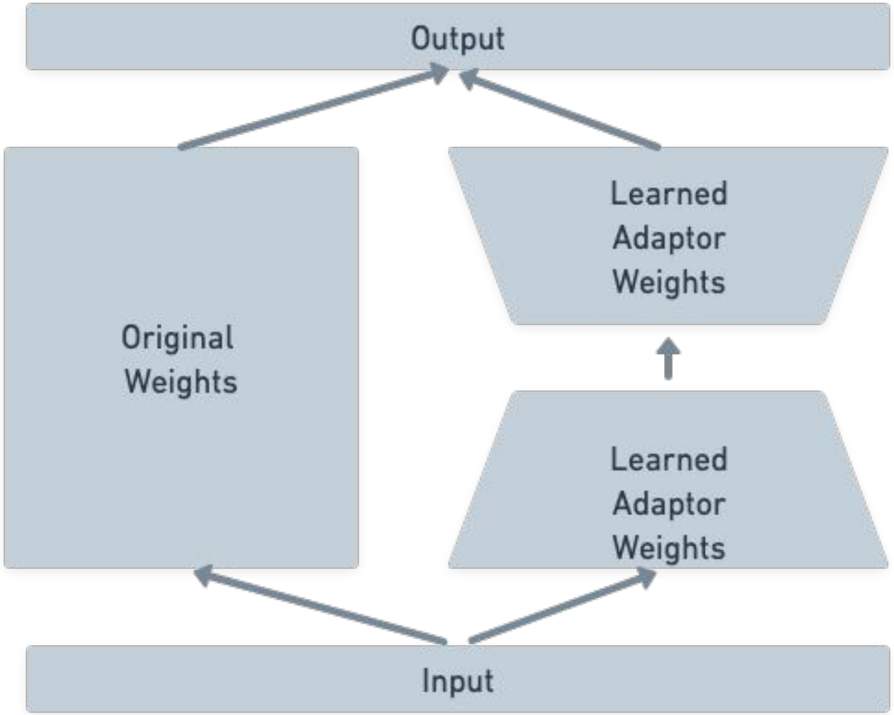
Mastering LLMs

A Conference For
Developers & Data Scientists

Plan For Today

- Serving Overview: Dan
- Deployment Patterns: Hamel
- Nvidia Inference Stack: Joe Hoover
- Lessons from Building A Serverless Platform: Travis Addair
- Batch vs Real Time and Modal: Charles Frye

Recap on LoRAs



Input: 4000 dimensions

Output: 4000 dimensions

Original weights: 16M

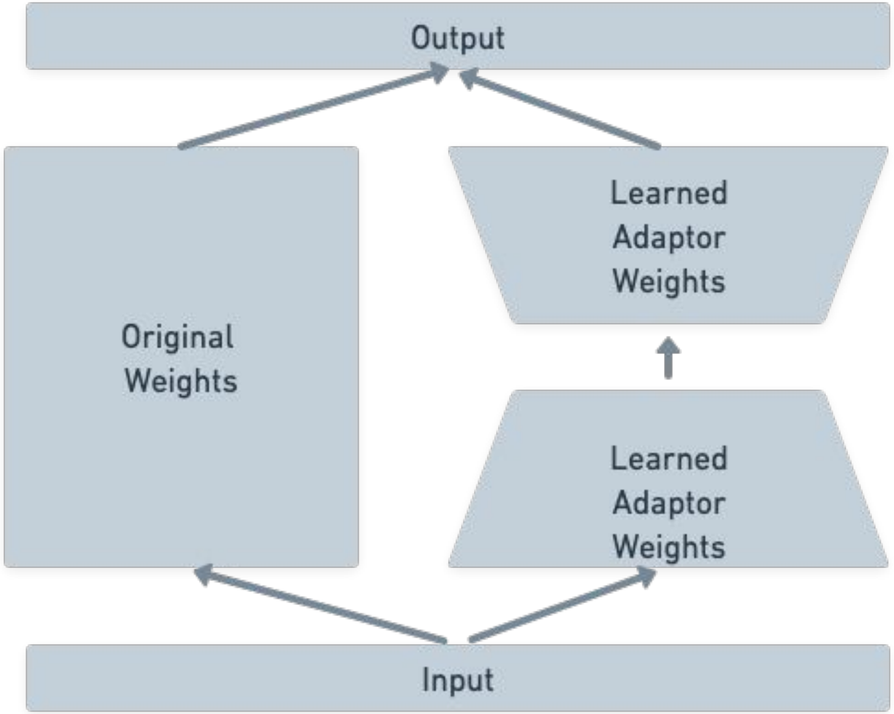
Adaptor rank: 16

LoRA weights: 2 * 16 * 4000

= 128,000

Merging Base Weights and LoRA

Before



Output

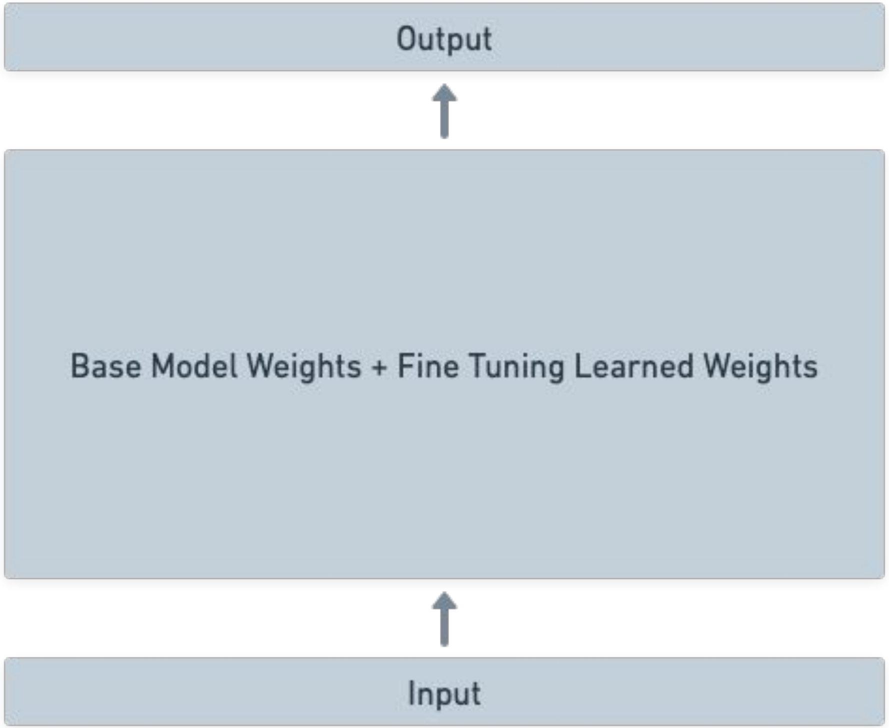
Original Weights

Learned Adaptor Weights

Learned Adaptor Weights

Input

After

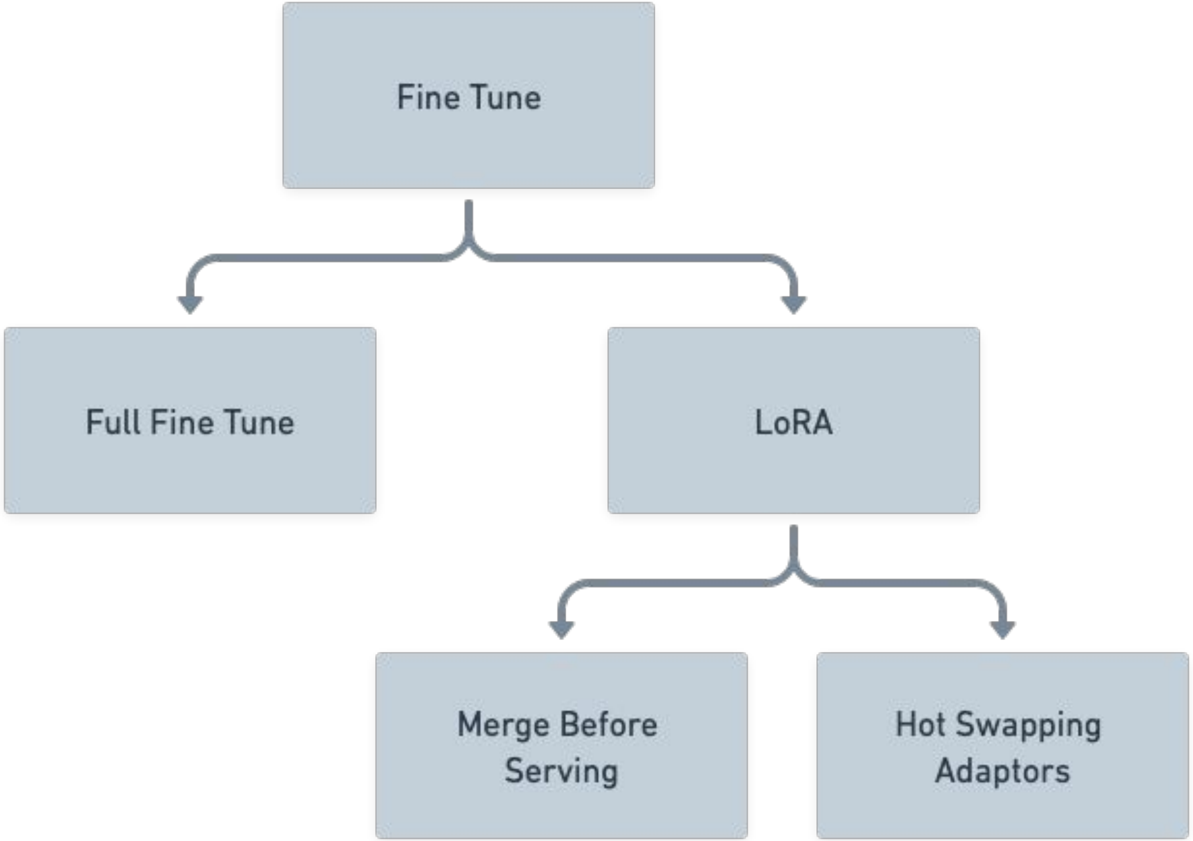


Output

Base Model Weights + Fine Tuning Learned Weights

Input

Recap on LoRAs



Fine Tune

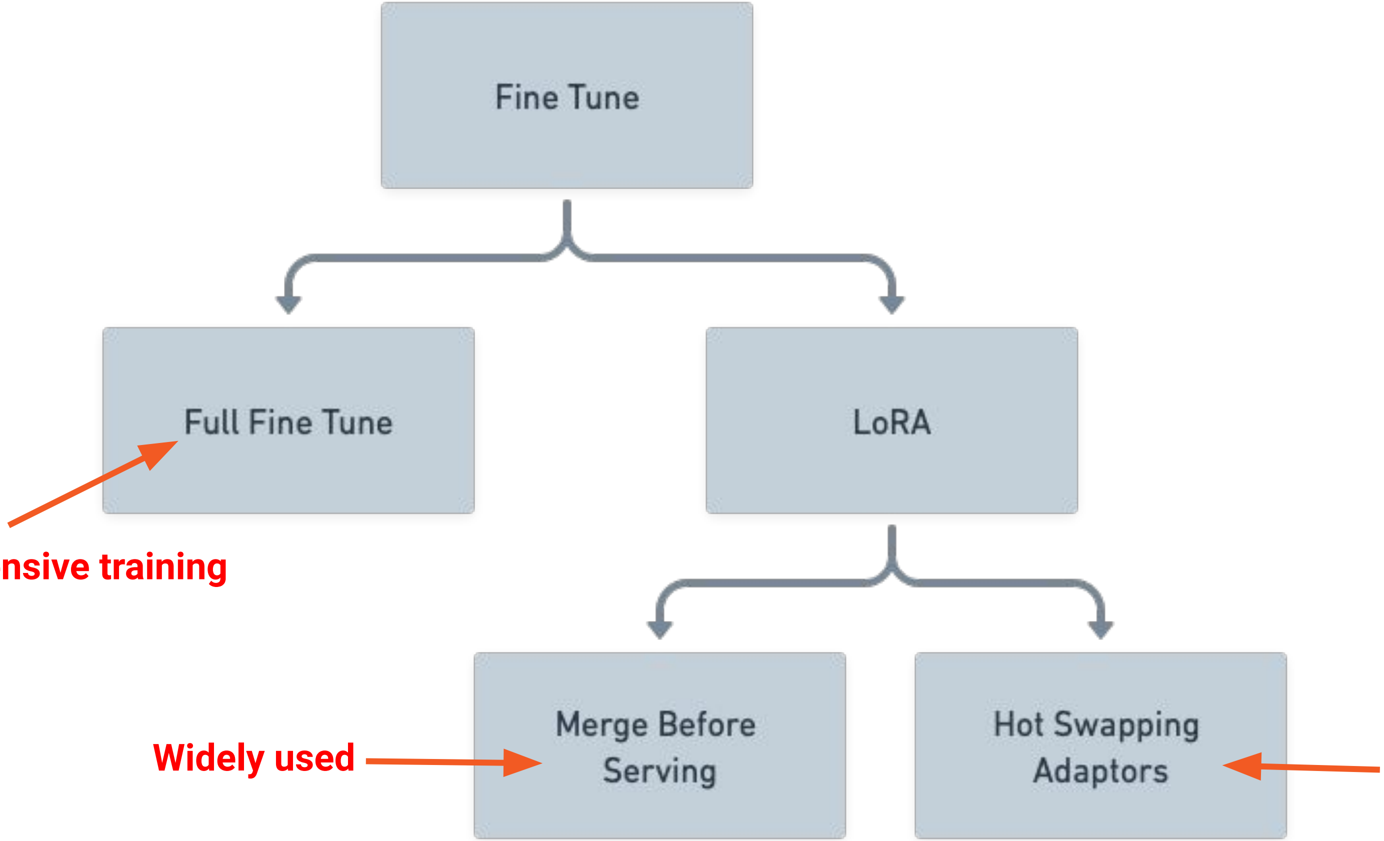
Full Fine Tune

LoRA

Merge Before Serving

Hot Swapping Adaptors

Recap on LoRAs



Fine Tune

Full Fine Tune

LoRA

Merge Before Serving

Hot Swapping Adaptors

Memory intensive training

Widely used

Potentially very efficient

Performance vs Costs

- GPU speed
- Model size
- Engineering efficiency
- **Cold starts vs idle time**
  - Potential win from hot-swapping

Many Projects Aren't Real Time

- Write Alt-Text Descriptions of images
- Extract chemical properties from papers for structured DB
- Edit journal articles to remove stereotypes
- Text-to-SQL analytics tool

Real-Time vs Batch/Offline

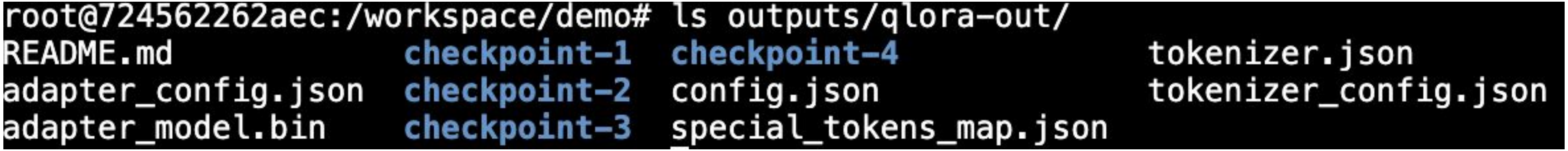
- Write Alt-Text Descriptions of images
- Extract chemical properties from papers to fill structured DB
- Edit journal articles to remove stereotypes

Offline

- Internal only text-to-SQL tool

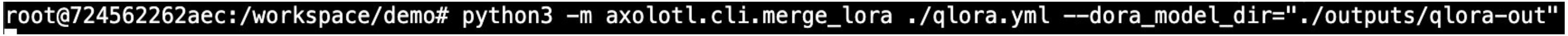
Used OpenAI

Merging LoRA to Base

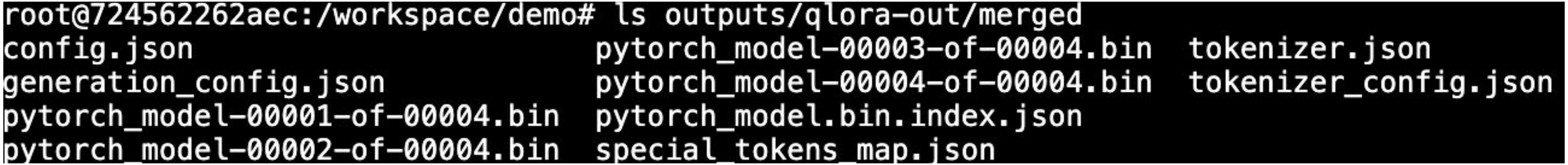


```
root@724562262aec:/workspace/demo# ls outputs/qlora-out/
README.md            checkpoint-1  checkpoint-4             tokenizer.json
adapter_config.json  checkpoint-2  config.json              tokenizer_config.json
adapter_model.bin    checkpoint-3  special_tokens_map.json
```

168MB



```
root@724562262aec:/workspace/demo# python3 -m axolotl.cli.merge_lora ./qlora.yml --dora_model_dir="./outputs/qlora-out"
```



```
root@724562262aec:/workspace/demo# ls outputs/qlora-out/merged
config.json                       pytorch_model-00003-of-00004.bin  tokenizer.json
generation_config.json            pytorch_model-00004-of-00004.bin  tokenizer_config.json
pytorch_model-00001-of-00004.bin  pytorch_model.bin.index.json
pytorch_model-00002-of-00004.bin  special_tokens_map.json
```

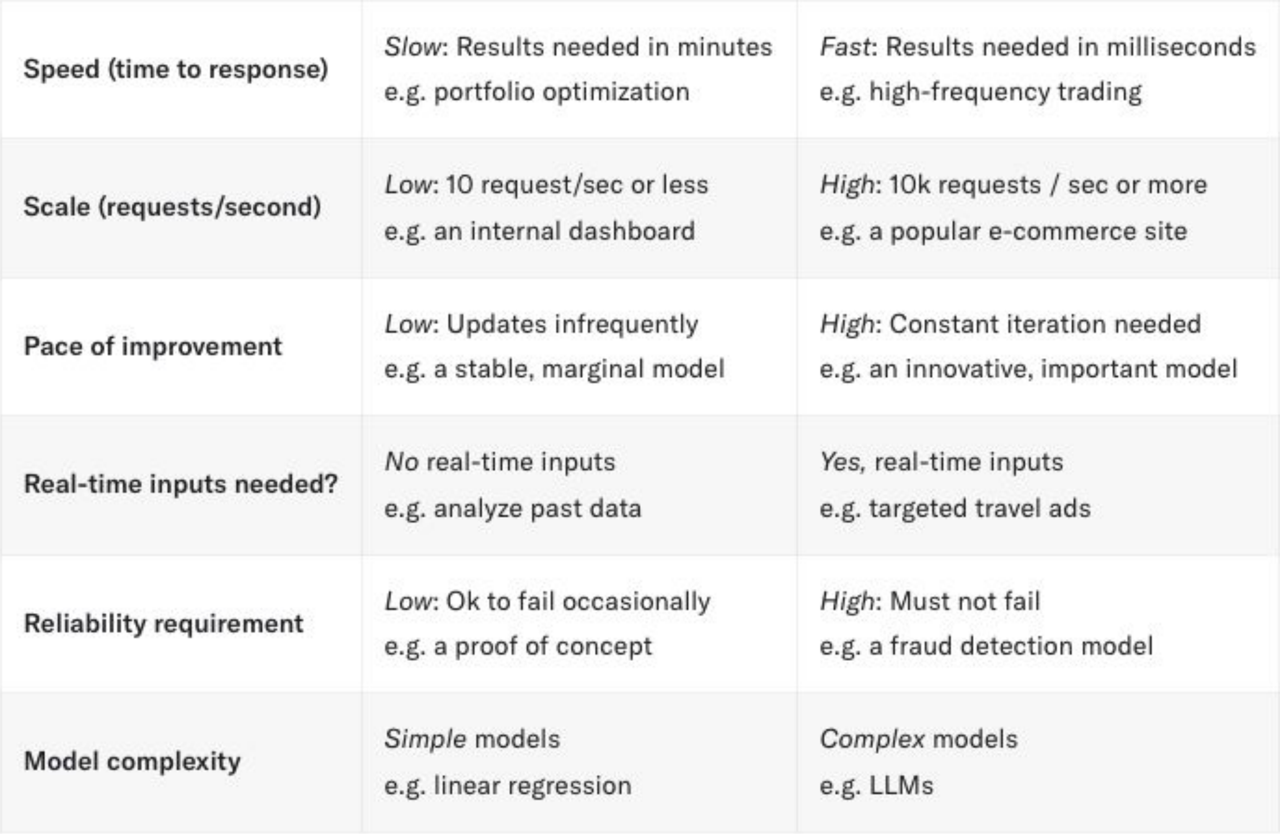
16 GB of weights in .bin files

Push Model Files to HF Hub

```
huggingface-cli repo create conference-demo

cp ./outputs/qlora-out/merged/* conference-demo

git lfs track "*.bin"

git add *

git commit -am "Push merged files"

git push origin main
```

Model Deployment

The Many Faces Of Deployments

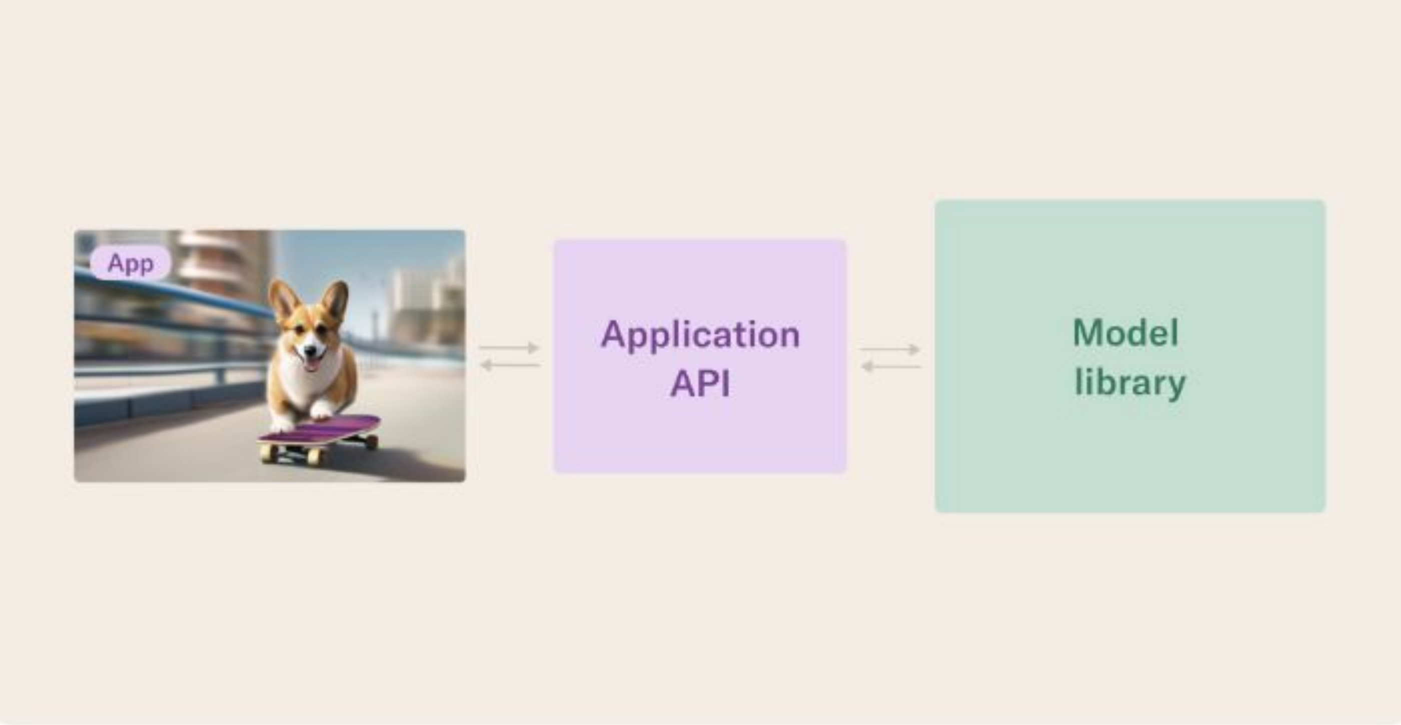


Speed (time to response)	*Slow*: Results needed in minutes e.g. portfolio optimization	*Fast*: Results needed in milliseconds e.g. high-frequency trading
Scale (requests/second)	*Low*: 10 request/sec or less e.g. an internal dashboard	*High*: 10k requests / sec or more e.g. a popular e-commerce site
Pace of improvement	*Low*: Updates infrequently e.g. a stable, marginal model	*High*: Constant iteration needed e.g. an innovative, important model
Real-time inputs needed?	*No* real-time inputs e.g. analyze past data	*Yes*, real-time inputs e.g. targeted travel ads
Reliability requirement	*Low*: Ok to fail occasionally e.g. a proof of concept	*High*: Must not fail e.g. a fraud detection model
Model complexity	*Simple* models e.g. linear regression	*Complex* models e.g. LLMs

Left: simple, lots of tools

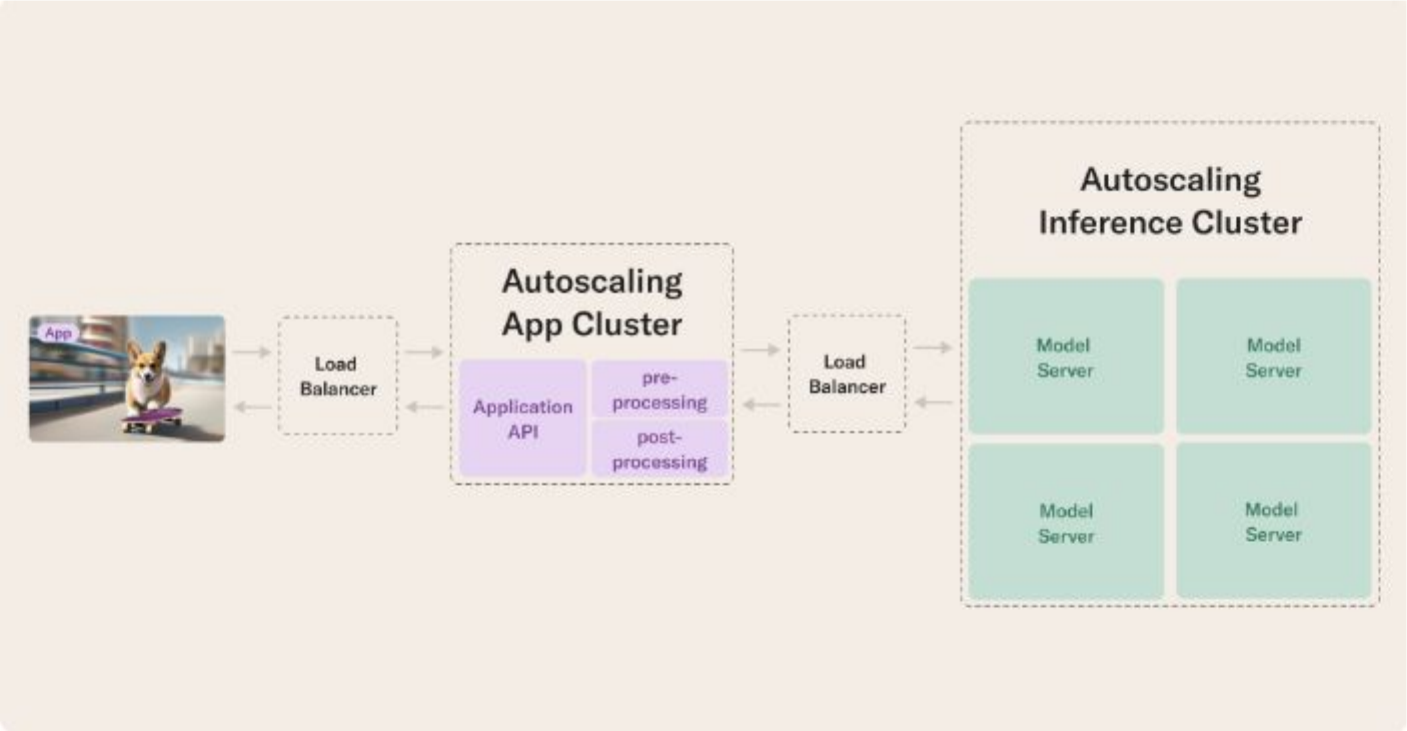
Right: some tools, customization could be needed.

Simple Model Serving



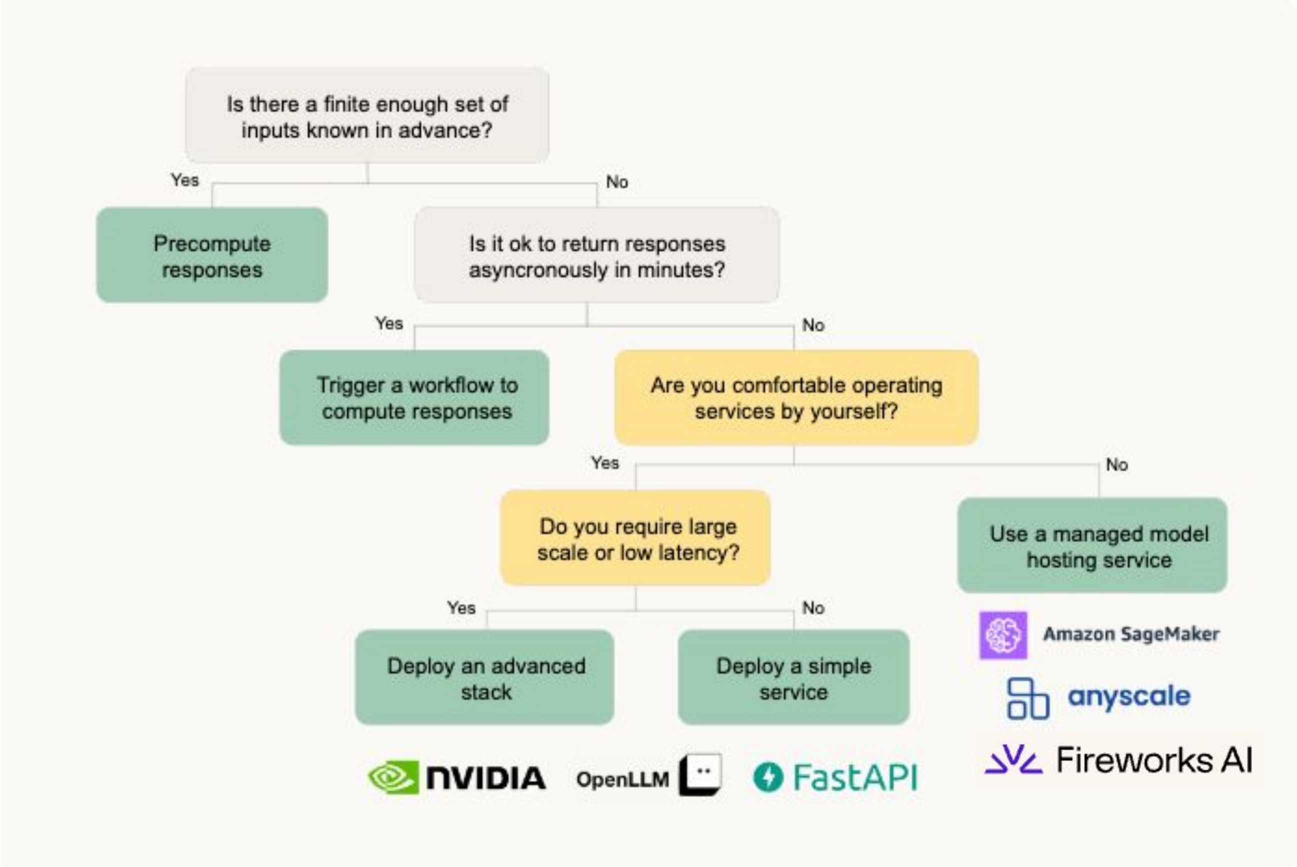
Ex: FastAPI

Advanced Model Serving



Ex: TensorRT-LLM + Triton + K8s

Kinds of Model Serving



Is there a finite enough set of inputs known in advance?

Yes

No

Precompute responses

Is it ok to return responses asyncronously in minutes?

Yes

No

Trigger a workflow to compute responses

Are you comfortable operating services by yourself?

Yes

No

Do you require large scale or low latency?

Use a managed model hosting service

Yes

No

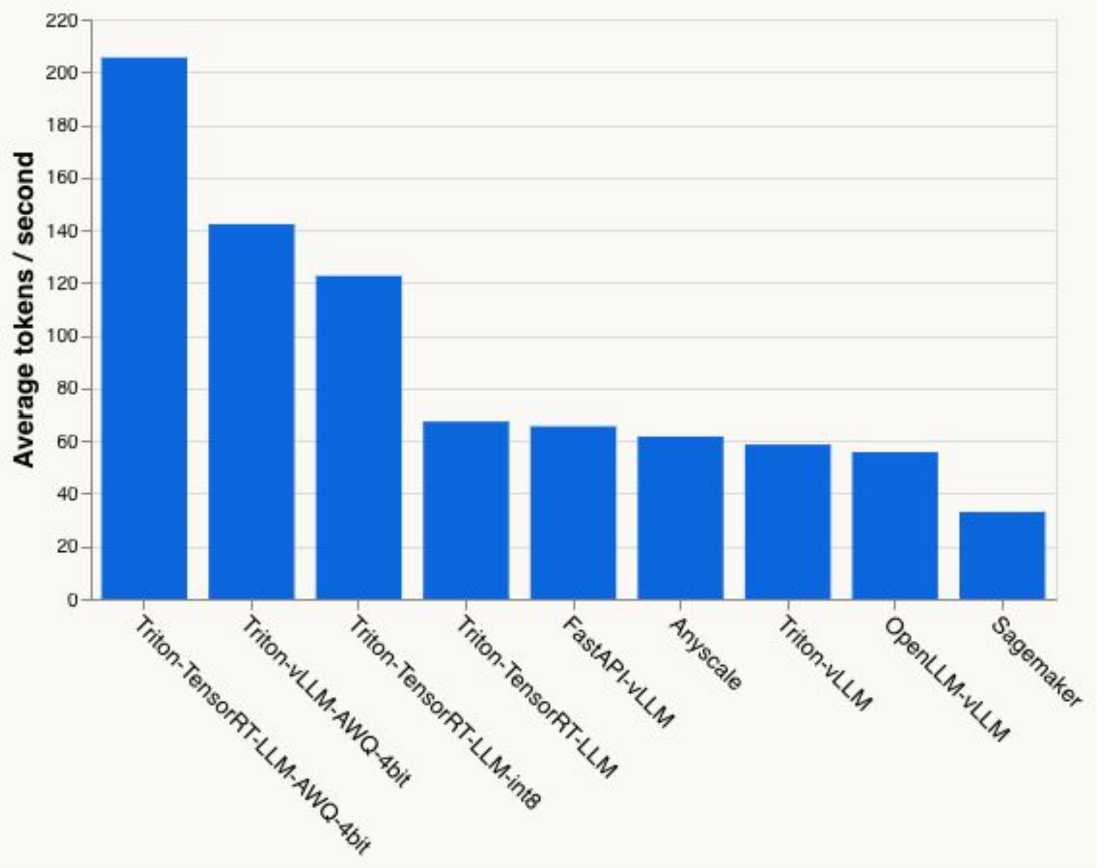
Deploy an advanced stack

Deploy a simple service

Exercise For Reader:

- Replicate
- Modal

GPU Poor Benchmark (Wrong, but useful)



These are most likely outdated. You need to try them.

vLLM is the most ergonomic. I recommend this unless you need the highest perf.

TRT-LLM is hard to use, but performant. (Joe)

Quantization can make things much faster, with caveats (Travis)

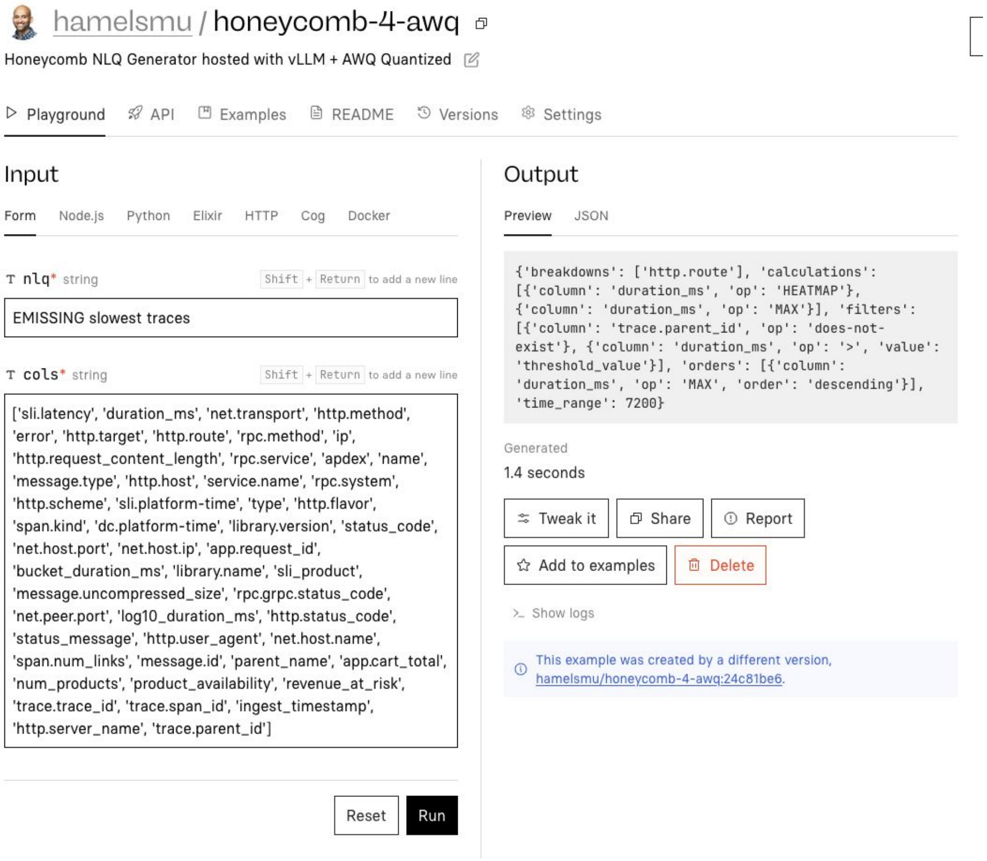
Footnotes:

- Run on a single RTX 6000 Ada GPU card.
- Use meta-llama/Llama-2-7b-hf, with a common set of input prompts.
- Max new output tokens are limited to 200.
- Batch size = 1, with eight different requests, over which we average our results. We always send one request prior to the eight requests to ensure that the inference server is “warmed up”.
- We measure the time it takes to return the input + output tokens, averaged over the eight requests.

Credit: https://outerbounds.com/blog/the-many-ways-to-deploy-a-model/

Honeycomb - Replicate

Why Replicate



Honeycomb NLQ Generator hosted with vLLM + AWQ Quantized

Input

T nlq* string Shift + Return to add a new line

EMISSING slowest traces

T cols* string Shift + Return to add a new line

['sli.latency', 'duration_ms', 'net.transport', 'http.method', 'error', 'http.target', 'http.route', 'rpc.method', 'ip', 'http.request_content_length', 'rpc.service', 'apdex', 'name', 'message.type', 'http.host', 'service.name', 'rpc.system', 'http.scheme', 'sli.platform-time', 'type', 'http.flavor', 'span.kind', 'dc.platform-time', 'library.version', 'status_code', 'net.host.port', 'net.host.ip', 'app.request_id', 'bucket_duration_ms', 'library.name', 'sli_product', 'message.uncompressed_size', 'rpc.grpc.status_code', 'net.peer.port', 'log10_duration_ms', 'http.status_code', 'status_message', 'http.user_agent', 'net.host.name', 'span.num_links', 'message.id', 'parent_name', 'app.cart_total', 'num_products', 'product_availability', 'revenue_at_risk', 'trace.trace_id', 'trace.span_id', 'ingest_timestamp', 'http.server_name', 'trace.parent_id']

Output

```
{'breakdowns': ['http.route'], 'calculations':
[{'column': 'duration_ms', 'op': 'HEATMAP'},
{'column': 'duration_ms', 'op': 'MAX'}], 'filters':
[{'column': 'trace.parent_id', 'op': 'does-not-
exist'}, {'column': 'duration_ms', 'op': '>', 'value':
'threshold_value'}], 'orders': [{'column':
'duration_ms', 'op': 'MAX', 'order': 'descending'}],
'time_range': 7200}
```

Generated

1.4 seconds

This example was created by a different version, hamelsmu/honeycomb-4-awq:24c81be6.

1. UI out of the box to share w/non-technical folks. Can lock down the inputs for specific domains.

2. Permalink for predictions - debugging!

SHOW ME THE CODE

https://github.com/parlance-labs/ftcourse/tree/master/replicate-examples

https://huggingface.co/parlance-labs/hc-mistral-alpaca-merged-awq

Joe

Q & A

Travis

Charles

Q & A