LLMs for the rest of us

Chenggang Wu

AI Camp San Francisco Meetup

4/4/2023

Today's Agenda

1. **Recent advancement in LLMs**: How can businesses derive value?
2. **Challenges and opportunities** with foundation models in 2023.
3. **Demo**: Running a LLaMA pipeline on Aqueduct.
4. **Peeking behind the curtain**: How Aqueduct enables you to run machine learning on any cloud infrastructure.

Recent advancements in LLMs (and other foundation models)

stability.ai

glean

copy.ai

Hugging Face

OpenAI

Jasper

Bard AI

genmo

Rephrase.ai

Character.AI

Meta

But… how can we generate business value with these models?

The critical unlock for businesses to benefit from foundation models is to be able to use them with proprietary data.

But wait! It's not quite that simple. With hosted model APIs, you have to consider...

- Ballooning costs
- IP concerns
- Difficult to debug results
- Vendor lock in
- Data privacy regulations

Open source to the rescue: A recent proliferation of open LLMs

RESEARCH

Introducing LLaMA: A foundational, 65-billio

February 24

As part of M
Model Meta
their work in
research co
democratizi

Hello Dolly: Democratizing the magic of ChatGPT with ope

Share this pos

Alpaca: A Strong, Replicable Instruction-Following Model

Authors: Rohan Taori* and Ishaan Gulrajani* and Tianyi Zhang* and Yann Dubois* and Xuechen Li* and Carlos Guestrin and Percy Liang and Tatsunori B. Hashimoto

We introduce Alpaca 7B, a model fine-tuned from the LL
turn instruction following, Alpaca behaves qualitatively s
(<600$).

Web Demo GitHub

Sta
Al

Prof. Joey Gonzalez
Co-founder & VP, Product

Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality

by the Team with members from UC Berkeley, CMU, Stanford, and UC San Diego

* *According to a fun and non-scientific evaluation with GPT-4. Further rigorous evaluation is needed.*

We introduce Vicuna-13B, an open-source chatbot trained by fine-tuning LLaMA on user-shared conversations collected from ShareGPT. Preliminary evaluation using GPT-4 as a judge shows Vicuna-13B achieves more than 90%* quality of OpenAI ChatGPT and Google Bard while outperforming other models like LLaMA and Stanford Alpaca in more than 90%* of cases. The cost of training Vicuna-13B is around $300. The training and serving code, along with an online demo, are publicly available for non-commercial use.

Vicuna (generated by stable diffusion 2.1)

Open-source LLMs present an **exciting opportunity**

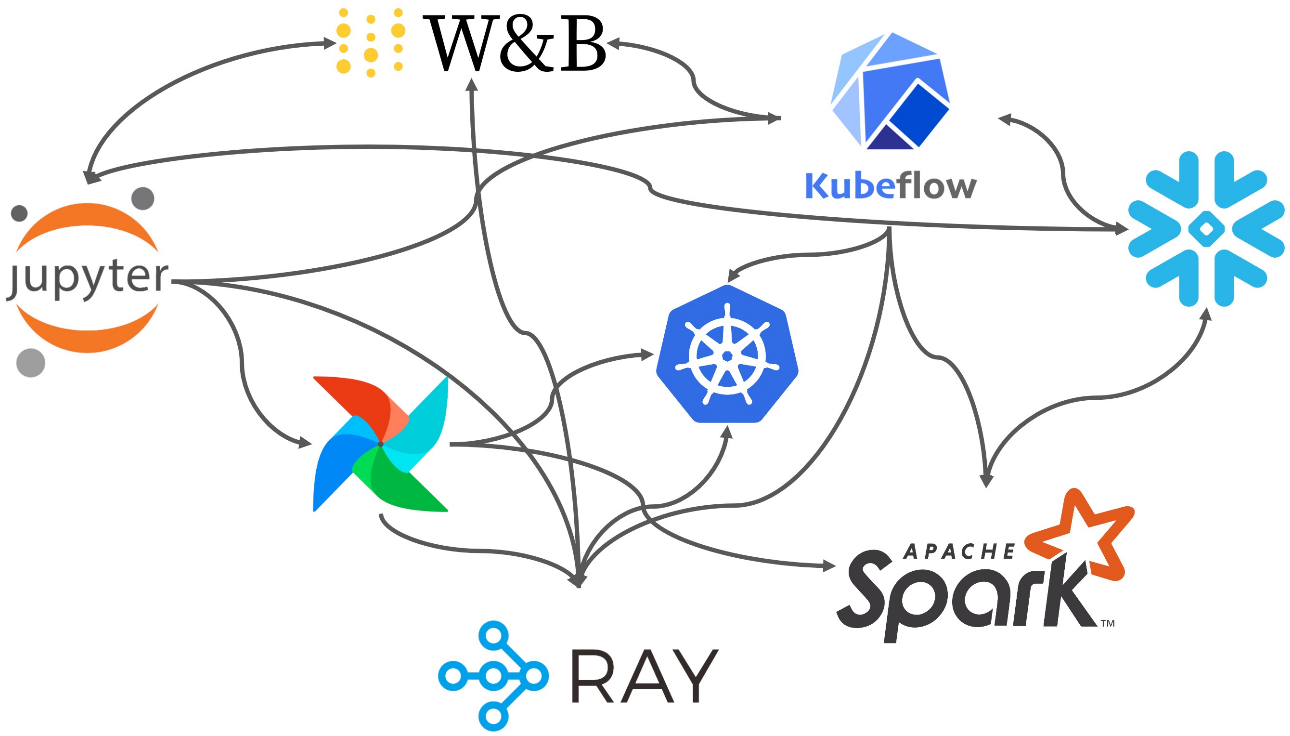
Challenges	Closed-Source LLMs (e.g., GPT, Bard)	Open-Source LLMs (e.g., LLaMa, Vicuna)
Ballooning costs	Expensive inference, combined with vendor markups	Smaller models; many are capable of running on a single Mac
Vendor lock-in	Proprietary APIs tie your applications to a single vendor	Full control over model usage and access to source code
IP Concerns	IP ownership is fraught and ill-defined	Flexible licensing models allow you to freely use, similar to other OSS
Data privacy regulations	Data has to move into the provider's cloud and out of your control	Models are self-hosted, fully in your cloud
Difficult to debug results	Hidden architectures and limited context to avoid leaking weights	Fine-tuning for specific results, with full context available

Open-source LLMs are great! **How do I use them?**

```
pip3 install llm
python3 -c "import llm; llm.generate('How do I increase sales?')"
```

Unfortunately, things aren’t quite this simple in reality.

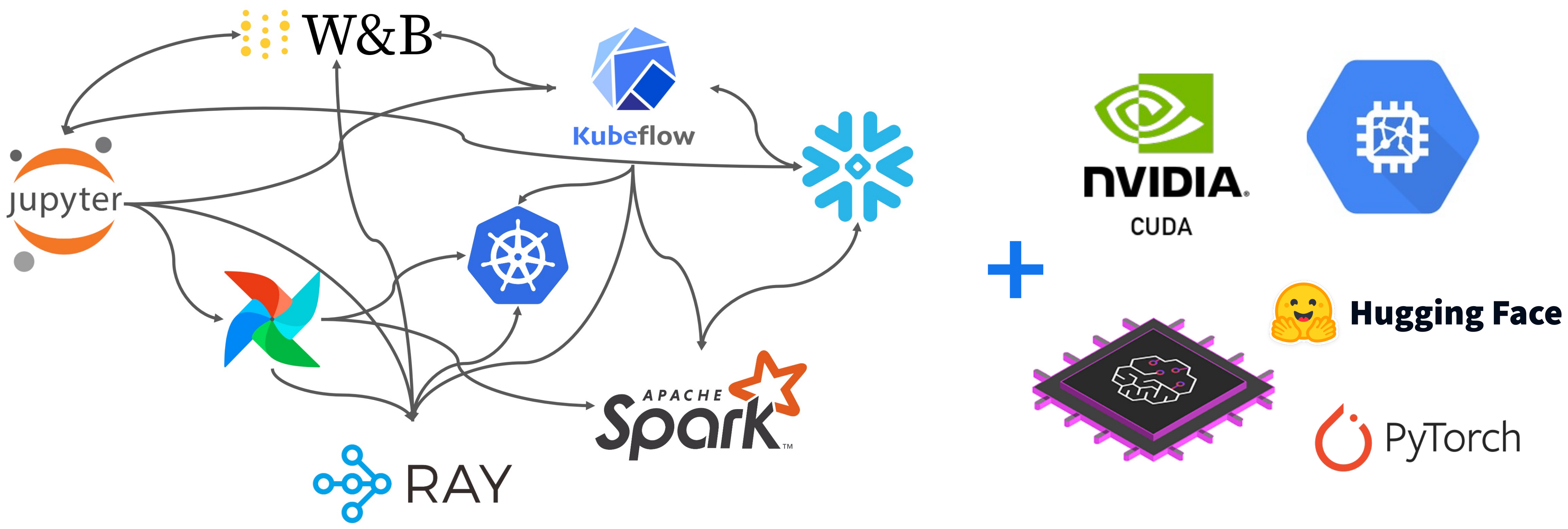
In reality, running machine learning in the cloud is incredibly complex...



Modern ML infrastructure is full of difficult to manage systems that interoperate poorly. We call this [the MLOps Knot](#).

… and LLMs only make things **that much more complicated**.

Because managing LLM environments requires yet more libraries, infrastructure, and specialized hardware.



Not to mention any of a million other challenges

Here’s a few things we’ve heard folks struggle with…

Runaway costs when infrastructure does not autoscale

Lack of resource availability for GPUs on cloud providers

Poor visibility into whether workloads run as expected

A brief history: The rise of Aqueduct

We've been working on scalable infrastructure for ~10 years.

Autoscaling Tiered Cloud Storage in Anna

Chenggang Wu, Vi

{cgwu, vikra

ABSTRACT

In this paper, we describe how we extended a di key-value store called Anna into an autoscaling, r service for the cloud. In its extended form, Anna is to overcome the narrow cost-performance limitati cal of current cloud storage systems. We describe t aspects of Anna's new design: multi-master select cation of hot keys, a vertical tiering of storage lay different cost-performance tradeoffs, and horizonta ity of each tier to add and remove nodes in res load dynamics. Anna's policy engine uses these me to balance service-level objectives around cost, late fault tolerance. Experimental results explore the of Anna's mechanisms and policy, exhibiting order nitude efficiency improvements over both commod KVS services and research systems.

PVLDB Reference Format:
Chenggang Wu, Vikram Sreekanti, and Joseph M. H Autoscaling Tiered Cloud Storage in Anna. *PVLDB*, 1: 638, 2019.
DOI: https://doi.org/10.14778/3311880.3311881

Cloudburst: Stateful Functions-as-a-Service

Vikram Sreekanti, Chenggan
Joseph E. Gonzalez
U
{vikrams, cgwu, charles.li
at

ABSTRACT

Function-as-a-Service (FaaS) platforms and "serverl computing are becoming increasingly popular due to and operational simplicity. Current FaaS offerings a at stateless functions that do minimal I/O and comn We argue that the benefits of serverless computi extended to a broader range of applications and algori maintaining the key benefits of existing FaaS offer present the design and implementation of Cloudburst FaaS platform that provides familiar Python program low-latency mutable state and communication, while n the autoscaling benefits of serverless computing. accomplishes this by leveraging Anna, an autoscaling store, for state sharing and overlay routing combined table caches co-located with function executors for da Performant cache consistency emerges as a key challe architecture. To this end, Cloudburst provides a com *lattice-encapsulated* state and new definitions and pr *distributed session consistency*. Empirical results on b and diverse applications show that Cloudburst mak functions practical, reducing the state-management of current FaaS platforms by orders of magnitude improving the state of the art in serverless consistency

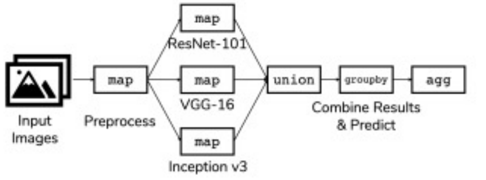
PVLDB Reference Format:
Vikram Sreekanti, Chenggang Wu, Xiayue Charles Lin, Joha Smith, Jose M. Faleiro, Joseph E. Gonzalez, Joseph M. Hellers Tumanov. Cloudburst: Stateful Functions-as-a-Service. *PV* 2438-2452, 2020.
DOI: https://doi.org/10.14778/3407790.3407836

Optimizing Prediction Serving on Low-Latency Serverless Dataflow

Vikram Sreekanti, *UC Berkeley*
Harikaran Subbaraj, *UC Berkeley*
Chenggang Wu, *UC Berkeley*
Joseph E. Gonzalez, *UC Berkeley*
Joseph M. Hellerstein, *UC Berkeley*

Abstract

Prediction serving systems are designed to provide large volumes of low-latency inferences from machine learning models. These systems mix data processing and computationally intensive model inference, and benefit from multiple heterogeneous processors and distributed computing resources. In this paper, we argue that a familiar dataflow API is well-suited to this latency-sensitive task, and amenable to optimization even with unmodified black-box ML models. We present the design of *Cloudflow*, a system that provides such an API and realizes it on an autoscaling serverless back-end. Cloudflow transparently implements performance-critical optimizations including operator fusion and competitive execution. Our evaluation shows that Cloudflow's optimizations yield significant performance improvements on synthetic workloads and that Cloudflow outperforms state-of-the-art prediction serving systems by as much as 2× on real-world prediction pipelines, meeting latency goals of demanding applications like real-time video analysis.




```
fl = cloudflow.Dataflow([('url', str)])
img = fl.map(img_preproc)
p1 = img.map(resnet_101)
p2 = img.map(vgg_16)
p3 = img.map(inception_v3)
fl.output = p1.union(p2,p3).groupby(rowID).agg(max,'conf')
```

Figure 1: An example prediction serving pipeline to classify a set of images using an ensemble of three models, and the Cloudflow code to specify it. The models are run in parallel; when all finish, the result with the highest confidence is output.

intensive; (2) it is a part of an interactive application, meaning

Aqueduct: A Control Center for AI & ML in the cloud

ML Technologies

ML-Centric APIs

Industry-Standard Cloud Infrastructure

Public Cloud Providers

A seamless developer experience to build great ML models.

Native integrations to all your existing cloud infrastructure.

Aqueduct helps you take ML **from your laptop to the cloud** in a few lines of code

```
def train(features):
  return model.train(features)

def validate(model):
    return validation_test(model)

validate(train(features))
```

Aqueduct helps you take ML **from your laptop to the cloud** in a few lines of code

```
@op(
  engine='eks-us-east-2',
  resources={'gpu_resource_name': 'nvidia.com/gpu'}
)
def train(features):
  return model.train(features)

@metric(engine='lambda-us-east-2')
def validate(model):
    return validation_test(model)

validate(train(features))
```

Demo

Running LLaMa on Aqueduct

Recapping the demo: How Aqueduct enables you to use foundation models

Support for accessing LLMs and foundation models in a **few lines of vanilla Python**. (Pre-release – coming soon!)

Seamlessly integrates with (and can manage) **your existing cloud infrastructure**.

Fully open-source and community driven – reach out and let us know what you think!

Peeking under the hood: How Aqueduct works

Compile

- Aqueduct automatically turns your Python code into engine-specific jobs.
- Transpile to Airflow DAGs, manage Python environments on Spark, …
- **Pre-packaged LLM environments** (coming soon)!

Optimize

- Use system-specific techniques to improve performance.
- What's in flight: Operator fusion, locality-aware scheduling, competitive execution, etc.

Orchestrate

- Schedule operators for execution.
- Seamless serialization and data exchange between Python code.
- Automatically persist, snapshot, and version data objects.

Aqueduct is fully open-source – let us know what you think!

aqueducthq/aqueduct

https://aqueducthq.com

@cgwu0530
@AqueductHQ

Chenggang Wu
Aqueduct