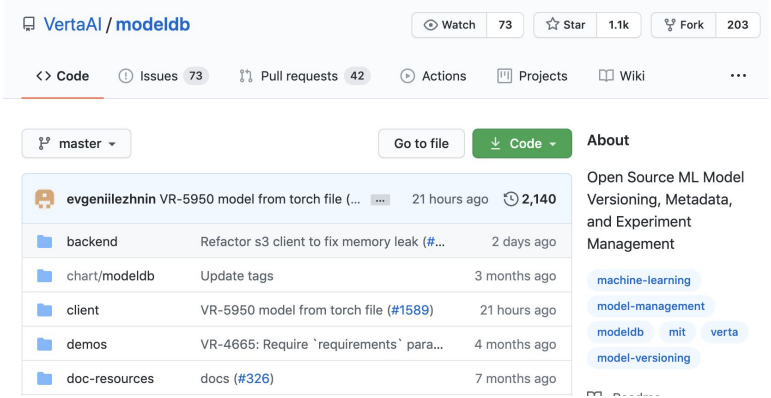
Serverless for ML-Serving on Kubernetes: Genius or Folly?

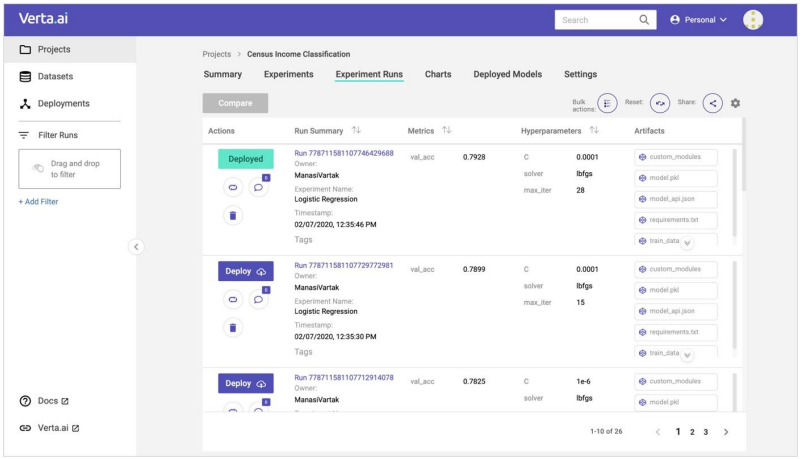
Manasi Vartak, Ph.D.
Founder and CEO, Verta

@DataCereal | manasi@verta.ai | www.verta.ai

Verta

About





https://github.com/VertaAI/modeldb

Open-source ML model management & versioning

Ph.D. thesis at MIT CSAIL

https://www.verta.ai/product

End-to-end MLOps platform for ML model delivery, operations and management

Kubernetes-based, operations stack for ML

Outline

- What is serverless and why is it interesting?
- Unique considerations for ML Serving
- Benchmark
- Key takeaways
- Check out if serverless is appropriate for you

Serverless: what is it and why is it interesting?

Variety of ways to run your code

Higher level
Of Abstraction

Serverless

Kubernetes/Containers

Virtual Machines

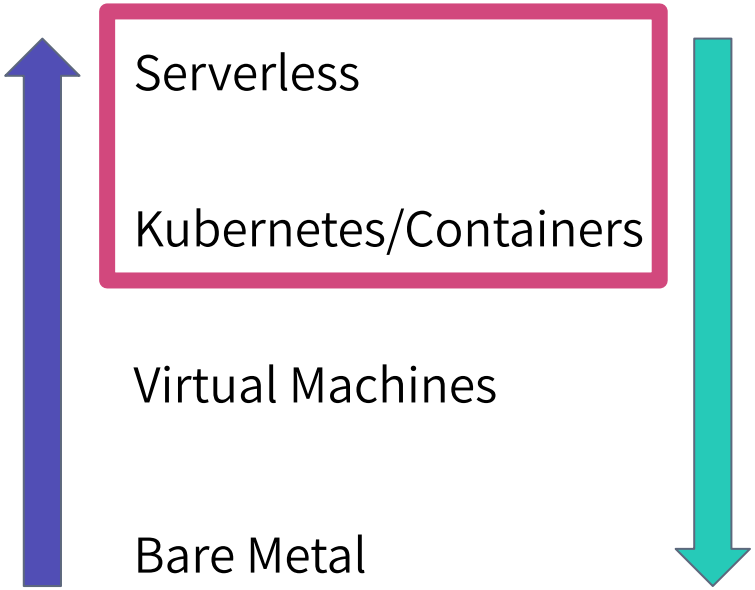
Bare Metal

Less
Control

More
Control

Lower level of
Abstraction

Variety of ways to run your code



Serverless 101

Serverless is, at its most simple, an outsourcing solution.

-- Martin Fowler, https://martinfowler.com/articles/serverless.html

Serverless 101

- Requires no provisioning or management of servers and does not involve a long-running server component
- Developer only writes the code or business logic without having to figure out how to deploy and run it
- Serverless platform takes care of deploying and scaling the application
  - Scale-up and scale down of resources happens on demand
  - Can scale to zero when there is no load
- Also known as function-as-a-service

Popular Serverless systems: AWS Lambda, GCP Cloud Run, etc.

Why serverless?

- Simple to use (developers focus on business logic)
- Simple to scale (more copies of the serverless instance created automatically)
- Low infrastructure maintenance overhead (don’t need to manage nodes, perform upgrades, tune resources requirements)
- Potentially cost-effective depending on workload (don’t need to provision resources that will not be used)

Popular applications: Async message processing, IOT workloads, stream data processing

Why not serverless?

- Serverless applications are stateless
- Implementation restrictions: limits on execution duration, resources available (e.g., memory, CPU, disk, concurrency)
- Cannot choose/control hardware
- Large latency on cold-start

When does serverless make sense?

- Application is stateless
- Resource requirements are modest
- Performance requirements / SLAs are not stringent
- Cold-start latency is not an issue
- Query workload is not steady
- Infrastructure maintenance is a large burden

ML Serving

Unique considerations for ML serving

- ML serving: making predictions against a trained model
- ML models can be large
  - E.g., DistilBERT model is 256 MB (compare to few MB of a Python-based non-ML function)
- ML libraries can be large and are varied
  - Libraries often have optimizations that can be enabled based on hardware
- Some ML models may need to be served via GPUs or TPUs

Benchmarks

Benchmark Specification

- Goal: Identify when it makes sense to use serverless for ML Serving
- Systems
  - Serverless SOTA (AWS Lambda)
  - Serverless on Kubernetes with knative (Google Cloud Run)
  - Container-based platform on Kubernetes (Verta)

Systems: Serverless SOTA - AWS Lambda

Managed Serverless Platform

How it works:

- Upload packaged code to S3
- Upload other dependencies to S3
- Trigger lambda via an event or HTTP request
- Platform manages all resources, scaling, and endpoints

Systems: Serverless on k8s - Google Cloud Run

Managed service running containers in serverless fashion on k8s (knative based)

How it works:

- Upload a **Docker container** to GCR
- Trigger Cloud Run via an event or HTTP request
- Platform manages all resources, scaling, and endpoints

Systems: Containers on k8s - Verta

Platform to run models as containers on k8s

How it works:

- Upload model and metadata to Verta platform
- Optionally specify resource and hardware requirements
- Deploy model on Verta
- Verta manages endpoints and scaling of models

Benchmark Specification

- Metrics
  - Prediction latency (warm-start)
  - Time to first prediction (cold-start)
  - Time to scale (autoscaling)
  - Usability concerns
- Workloads
  - Variety of models including state-of-the-art NLP, CV, and traditional ML models
  - Varying QPS

Results

Caveats!

- Serverless for k8s is still evolving. These numbers are based on the software and capabilities available today
- For managed services, there are knobs that cannot be controlled by the end user and optimizations performed under the hood
- We use off-the-shelf settings in this benchmark
- **This talk covers a subset of the benchmark results, for the full set of results visit: verta.ai/serverless-inference-benchmark**

Results: Usability Concerns (Serverless)

- Serverless platforms have hard restrictions on resources available

	AWS Lambda	**Google Cloud Run**
Memory	3 GB	4 GB
Disk	250 + 500 MB	4 GB
CPU	Proportional to memory	4 vCPUs
GPUs	n/a	n/a

Results: Usability Concerns (Serverless)

- If your model or ML library(s) doesn't fit in these constraints, you cannot use the serverless platform
  - Ex.1.
    - Torch + transformer library (HuggingFace) for DistilBERT > 500 MB
    - Need to surgically remove pieces of the libraries or get creative with model loading to even use lambdas
    - Significant wrangling required
  - Ex.2.
    - Embedding + nearest neighbor lookup model > 20 GB
    - Doesn't fit constraints for any of the serverless platforms

Results: Usability Concerns (Serverless)

Configuration options

- ML libraries and lower-level linear algebra libraries have optimizations that can be tuned via environment variables
- However, settings are tied to underlying hardware and serverless platforms are not transparent wrt hardware used
- Additionally, serverless platforms do not allow hardware to be customized

Results: Warm-start Prediction Latency

Model: DistilBERT

	P50 (s)	P95 (s)	P99 (s)
AWS Lambda	0.4885	0.5341	0.5738
Google Cloud Run	0.3848	0.4574	0.4971
Verta	0.2605	0.2809	0.2935

Configuration:

AWS Lambda: Memory=3 GB
Google Cloud Run: 2 CPUs, 3 GB
Verta: CPU=1.6, Memory=3 GB
1 worker/query

Observations:

Verta (container-based system) has lower latency by 2X. This can be attributed potentially to more control on environment (e.g., Intel vs. AMD processors). Without implementation details of AWS and GCP systems, hard to identify root cause.

Results: Cold-Start Prediction Latency

Model: DistilBERT

	Time to first request (s)
AWS Lambda	41
Google Cloud Run	8.2
Verta	0.7

Configuration:

100 qps, 1 worker/query
Steady state: receives 100 responses/sec
Time to 1st response: time to first successful response
Resource configs same as before

Observations:

Verta (container-based system) always has >=1 model replica running, so time to first request is the lowest.

Results: Scaling Latency

Model: DistilBERT

	Time to reach steady state (s)
AWS Lambda	79
Google Cloud Run	33
Verta	105 (pods only) 315 (pods+nodes)

Configuration:

100 qps, 1 worker/query
Steady state: receives 100 responses/sec
Resource configs same as before

Observations:

Auto-scaling is faster on serverless systems.

Results: Varying model size

	DistilBERT - P95 Latency	**BERT -** P95 Latency	**DistilBERT -** Time to 1st req	**BERT -** Time to 1st req
AWS Lambda	0.5341	0.9878	41.0138	41.1018
Google Cloud Run	0.4574	0.7616	8.2	15.8
Verta	0.2809	0.4887	0.3043	0.8179

Models:

DistillBERT: 254MB
BERT: 416 MB

Observations:

Running a larger model increases latency across the systems. Cold-start metrics show some degradation.

Note on Cost

- Pure infra costs may be comparable or even higher than a non-serverless system
- Tied to exact workload and resources used (e.g., bursty, seasonal, continuous and steady).
- However, TCO is usually lower due to lower development and maintenance costs

Takeaways

Takeaways

- Serverless solutions have hard limits on resources. If your model doesn't fit into those constraints, serverless is not a good fit
- Ability to configure hardware can lead to better performance for non-serverless systems
- Scaling with serverless platforms is faster than vanilla autoscaling in k8s
- Query pattern and workload will affect costs of running ML-inference in serverless fashion vs. w/servers

Check out verta.ai/serverless-inference-benchmark to learn how to run the benchmark!

Thank you.

Is serverless right for your ML workload? Check out verta.ai/serverless-inference-benchmark

Reach out at manasi@verta.ai or @DataCereal with questions.

Verta