Paper Review- INFaaS: Automated Model-less Inference Serving

Summary: This paper introduces INFaaS, an automated, model-less system for distributed machine learning inference serving that simplifies deployment and improves cost efficiency. Traditional systems force developers to manually choose from thousands of model variants that combine different hardware, optimizations, and batch sizes, causing inefficiency and high costs since inference already accounts for more than 90% of ML infrastructure expenses. INFaaS introduces a declarative API where developers specify only high-level goals such as latency, accuracy, or cost, and the system automatically generates, profiles, and selects the best variant and hardware for each query. Internally, INFaaS acts as a smart middle layer with a Controller that routes queries, Worker machines that execute models, a Metadata Store that tracks performance and resource usage, and a Variant Generator and Profiler that create and evaluate model variants, functioning together like a traffic control system for inference. Beyond standard VM-level horizontal autoscaling, INFaaS adds model-level decisions that can replicate an existing variant or switch (vertical scaling) to a faster or cheaper one depending on load and hardware availability. INFaaS uses a smart cost optimization approach, formulating a mathematical integer linear program (ILP) to minimize total cost while meeting performance goals, but since solving it exactly is too slow, it applies a greedy heuristic that makes near-optimal decisions in milliseconds. Implemented in C++ with gRPC and Redis, and using TensorRT and Neuron for variant generation, INFaaS supports TensorFlow, PyTorch, and Caffe2 models. Evaluated on 22 architectures and 175 variants across CPUs, GPUs, and AWS Inferentia chips using real Twitter-like workloads, INFaaS achieved 1.3x higher throughput, 1.6x fewer latency violations, 21% lower cost, and made scheduling decisions 35x faster than brute-force approaches compared to systems like TensorFlow Serving, SageMaker, and Clipper.

Evaluation: The paper identifies an important and timely challenge in simplifying ML inference deployment while managing cost, accuracy, and latency trade-offs. Existing systems overlook the complexity introduced by heterogeneous hardware and numerous model-variants, often forcing developers into static and suboptimal configurations. INFaaS's model-less abstraction and automatic variant selection represent a meaningful step forward, enabling dynamic optimization and improved resource utilization, such as 5.6x higher GPU utilization demonstrated in experiments. The design choices, including model-vertical autoscaling, the state-machine-based scheduler, and the cost-aware greedy heuristic, are technically sound and well-motivated. The evaluation is detailed, covering 22 model architectures and 175 variants across CPUs, GPUs, and AWS Inferentia using realistic Twitter-like workloads. The results strongly support the authors' claims, showing 1.3x higher throughput, 1.6x fewer SLO violations, and 21 percent lower cost compared to strong baselines. Overall, the paper is very clear, well written and well explained with flow diagrams to explain the architecture and concepts.

Main takeaways

1. INFaaS simplifies deployment with an abstract, goal-based API where developers specify latency, cost, or accuracy targets instead of choosing specific models or hardware.
2. INFaaS maximizes efficiency by using fast, state-machine-driven variant selection that turns hardware and model diversity into higher utilization, lower latency, and better throughput.

Strengths

1. INFaaS optimizes intelligently using an ILP-based cost model and a fast greedy heuristic that makes near-optimal decisions in 2-3 ms, about 35x faster than brute-force search.
2. Comprehensive evaluation across frameworks, hardware, and real workloads shows consistent gains over TensorFlow Serving, SageMaker, and Clipper.

Weaknesses

1. INFaaS optimizes inference serving but does not address how model updates or retraining cycles might affect variant management in real-world pipelines.
2. Static profiling may become outdated as hardware and drivers evolve.

Discussion: Could INFaaS adaptively re-profile variants to handle hardware drift over long deployments?