Meeting SLOs, Slashing Hours.

Automated Enterprise LLM Optimization

Nicholas Santavas · eBay Foundation Models

with Kareem Eissa, Patrycja Cieplicka, Piotr Florek, Matteo Nulli, Stefan Vasilev, Seyyed Hadi Hashemi, Antonios Gasteratos, Shahram Khadivi

eBay Foundation Models · Amsterdam, NL
Democritus University of Thrace · Xanthi, Greece

The production reality.

- **LLM feature demand** grows exponentially.
- **GPU supply is fixed** - every inefficiency steals from another team.
- **8B -> 70B+ deployment** requires hand-tuned compression and runtime.
- **80-100 expert hours** - per model, per workload.

Make LLM optimization a job you submit — not a hundred-hour expert engagement.

Three design constraints we held ourselves to:

Same interface for every team.

Submit a job spec, get a deployment-ready model back.
No hand-holding by a quantization expert.

The number you ship is the number you measured.

Throughput and latency certified under your real serving profile, not a research benchmark.

Every trial archived.

Configs, metrics, artifacts — all stored.

The landscape every team already has.

Data & Model Sources

- HDFS / Object Store
- Model Management System
- Experiment Management System

Monitoring & Telemetry

Grafana · Logs · Tracing

Models, data and a GPU cluster.

Pipes between them are the team's problem.

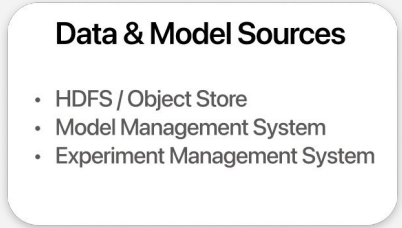
Ray Cluster

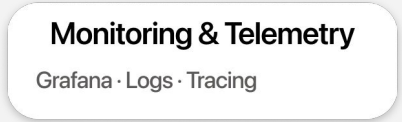
H200 Nodes

H100 Nodes

A100 Nodes

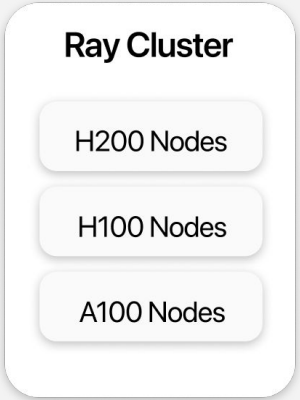
Tooling exists for the pieces.





Each piece has a library.

Wiring them up is still the team's job.

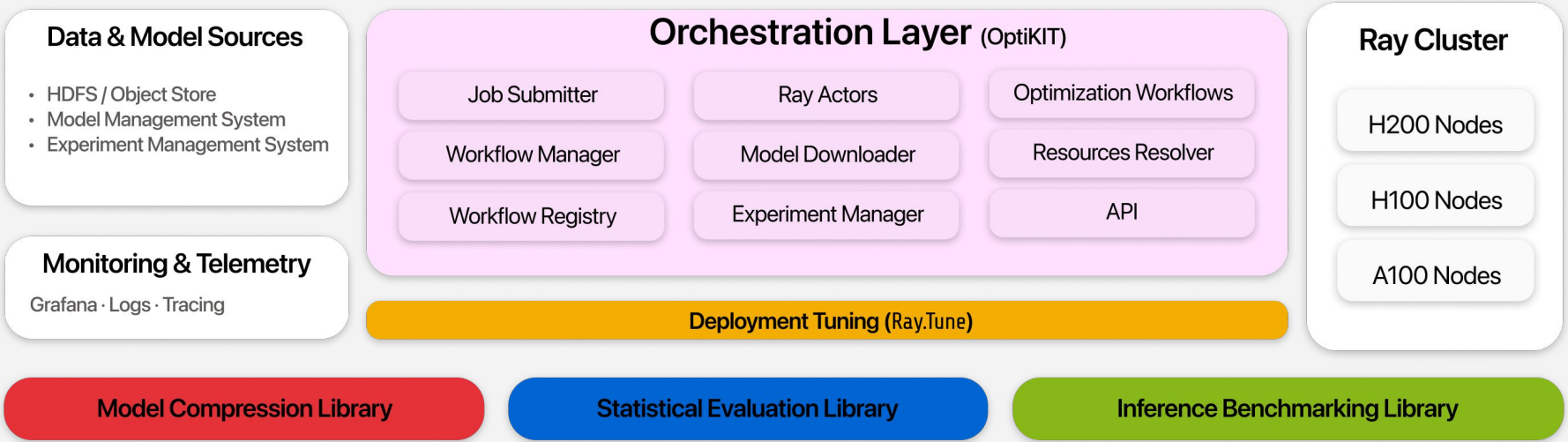


Model Compression Library

Statistical Evaluation Library

Inference Benchmarking Library

OptiKIT is the layer that turns it into one workflow.

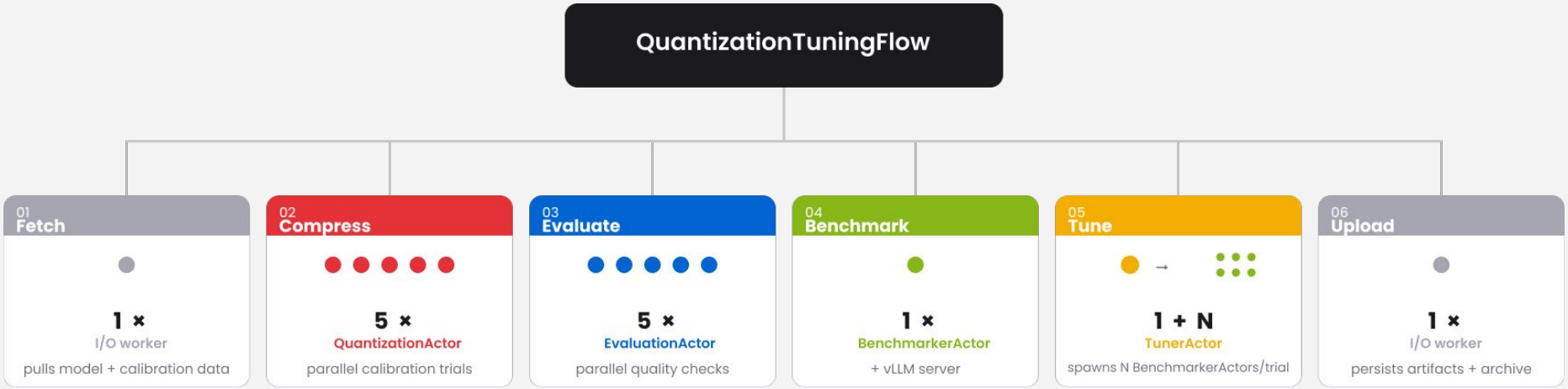


Submit one job spec — done.

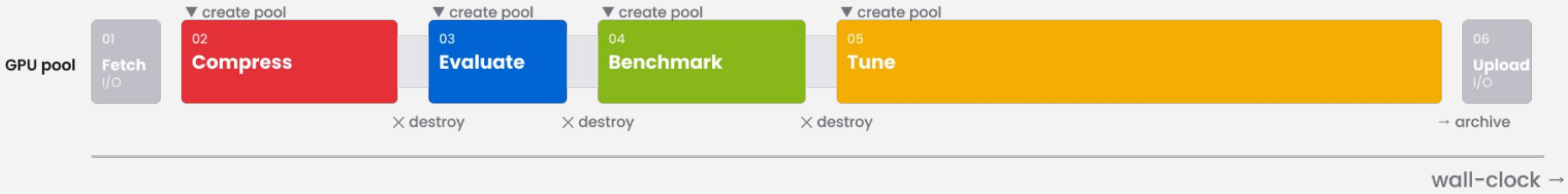
```
job = OptimizationJob(
    flow="quantize_and_tune",
    model=ModelStorage(name="llama-70b", version="v1.0"),
    dataset=DatasetStorage(path="/data/calibration"),
    flow_params={
        "recipe": "int_w8a8",
        "num_trials": 5,
    },
    compute=ComputeConfig(sku=ResourceSKU.H100_8),
)

result = Submitter().submit(job)
```

Six stages, one job.



Stage-isolated actor pools.

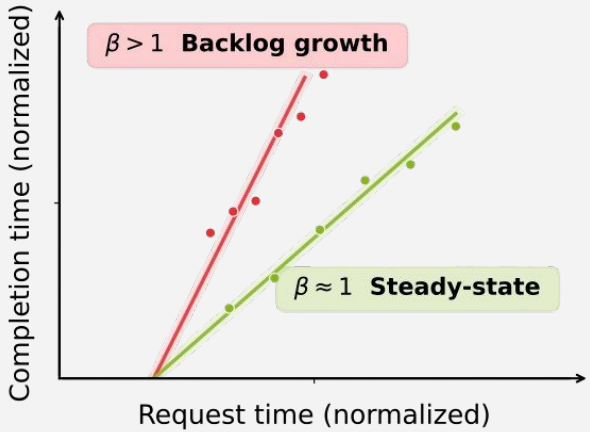


1 Per-stage sizing
different compute & memory footprints get tailored pool sizes

2 Deterministic teardown
destroy between stages → clean GPU state, reproducible runs

3 Fault isolation
one bad trial doesn't sink the job — bounded retries per actor

Certifying the rate you'll actually serve.



$$r_i = \alpha + \beta\, c_i + \varepsilon_i$$

- A linear fit of completion vs. request times, per benchmark trial.
- When β≈1, the system is in steady state; when β>1, backlog grows.

Rate Search

Phase 1 — Closed-loop

One synchronous trial. If latency SLO fails under zero queuing, reject config.

Phase 2 — Open-loop

After each trial:

- Pass → 2r — *doubling*
- Fail → (Lower Bound + r)/2 — *midpoint halving*

Runtime tuning.

$$\text{fitness}(c) = \frac{\text{throughput}(c)}{\text{tensor\_parallel\_size}(c)} + \lambda \cdot \text{slo\_penalty}(c)$$

TPE via Ray Tune+Optuna · 30 trials per workload · each trial spawns a BenchmarkerActor → vLLM server

`tensor_parallel`	{ 1, 2, 4, 8 }
`max_num_seqs`	concurrency ceiling
`max_num_batched_tokens`	per-step batch budget

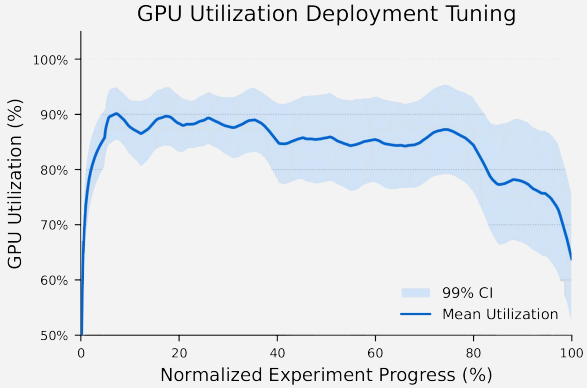
What this bought us.

Model	Input / Output (prefix)	Performance Objective
Llama 3 70B	5000 / 500	Throughput-oriented
Qwen 2.5 7B	1200 / 80	Latency p95 $\leq$ 500 ms
Mistral Small 3 24B	3000 / 200 (2000)	Latency p95 $\leq$ 1500 ms
Mistral Small 3 24B	1500 / 1500 (1000)	TTFT p50 $\leq$ 50 ms, TPOT p50 $\leq$ 10 ms

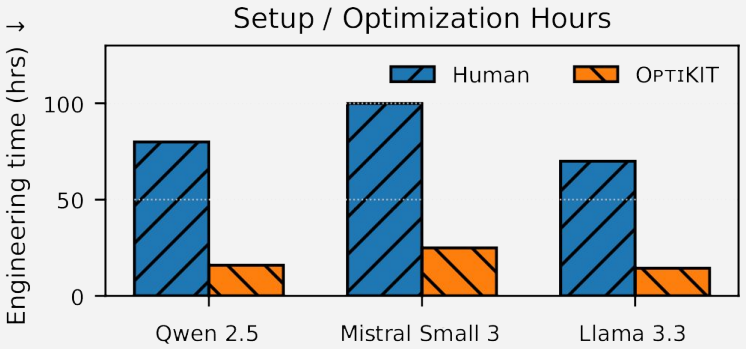
Model	Baseline	Quantization only		Tuning only		Quantization + Tuning	
		Norm. TPS	Gain	Norm. TPS	Gain	Norm. TPS	Gain
Qwen 2.5 7B	3.52*	5.96	1.69×	6.79	1.93×	**7.50**	**2.13**×
Mistral 24B (Latency p95)	0.604*	1.732	2.87×	0.937	1.55×	**1.734**	**2.87**×
Mistral 24B (TTFT & TPOT p50)	0.562*	0.562	1×	0.750	1.33×	**0.875**	**1.55**×
Llama 3 70B	0.468*	**0.593**	**1.26**×	0.468	1×	0.585	1.25×

*SLOs not met without either quantization or tuning at the indicated tensor-parallel levels (TP=1, 2).

What this bought us.



Concurrent trials, ephemeral pools → 88% mean utilization across the tuning stage.



Specialist effort drops 4–5× across model families.

Key takeaways.

01

Throughput is workload-dependent.
Tie every result to a serving profile and an SLO.
Same model, different Π → different RPS, different ceiling.

02

Don't over-curate your calibration data.
~256 random samples held >99% recovery across three base-instruct model families on short-context tasks. Start simple.

03

Tuning helps under tight SLOs
Automation reduces expert time.
>2× on tight-latency workloads, flat on throughput-bound.
The 80h → 20h specialist bottleneck is the bigger win.

Thank you!