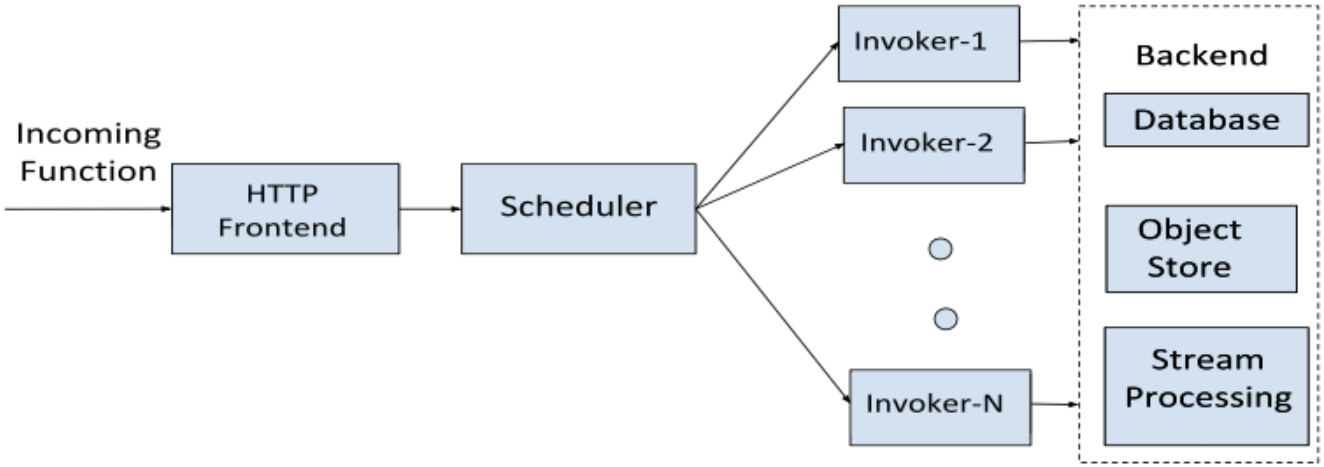
CASSIA: Towards Unified, Autonomous Scheduling for Highly Variable Serverless Workloads

Pranav Digamber Talegaonkar

Guided by: Dr. T Veni

Introduction



- Serverless computing eliminates infrastructure management, enabling automatic scaling and cost-efficiency.
- Challenges include cold starts, resource contention, and inefficient scheduling.

Motivation

- Serverless computing faces performance degradation due to workload variability.
- Efficient scheduling is essential to optimize resource allocation and reduce costs.
- This review explores recent advancements to address these challenges.

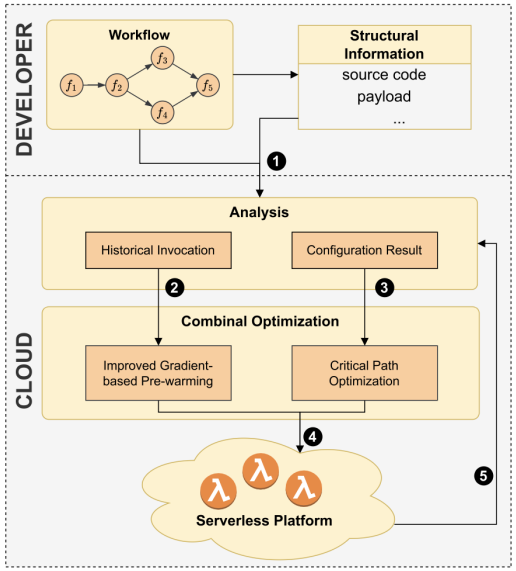
Problem Definition

Design a hybrid scheduler using a self-supervised context encoder and REINFORCE policy to assign function invocations to nodes while minimizing latency and cold starts.

Literature Review - ENSURE(Efficient scheduling and autonomous resource management in serverless environments)

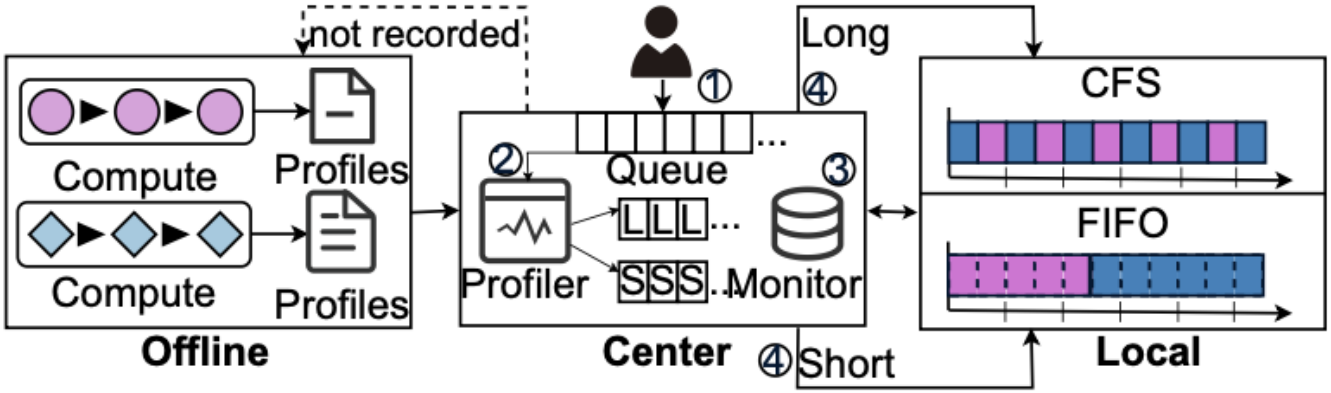
- Classifies functions into Edge-Triggered(short-lived) and Massively Parallel(long-running).
- Regulates resource usage to prevent contention.
- Uses greedy packing to optimize invoker utilization.
- Results: 52% resource efficiency improvement, 60% cold start reduction.

Literature Review - SSC (Sustainable Serverless Computing)



- Gradient-based pre-warming optimization reduces cold start hit rate based on invocation rate.
- Critical path selection optimizes the most impactful function execution.
- Results: 50% cold start reduction, 30% cost savings.

Literature Review - Synergy



- Hybrid scheduling with central and local schedulers for improved coordination.
- Central scheduler for global resource view; local schedulers for immediate decisions.
- Results: 63% average execution time reduction, 30% faster task completion.

Proposed Solution: CASSIA

The **CASSIA** (Context-Aware Self-Supervised Intelligent Autoscheduler) framework uses a three-step learning process:

- **Phase 1: Pre-train** (Offline Self-Supervised Learning)
  - Learns robust "smart profiles" (embeddings) of functions from unlabeled data.
  - Identifies inherent function characteristics and relationships.
- **Phase 2: Schedule** (Real-time Adaptive Decision-Making)
  - Uses pre-trained profiles to make fast, intelligent scheduling decisions for new invocations.
  - Optimizes for latency and cold start avoidance.
- **Phase 3: Fine-Tune** (Online Reinforcement Learning)
  - Continuously improves the scheduling policy based on real-world feedback (latency, cold starts).
  - Adapts to changing workload patterns and environment conditions.

CASSIA Phase 1: Pre-train (Offline Profiling)

- **Self-Supervised Learning:** The model learns by masking a feature (x_m) and trying to predict it ($\hat{x}_m$).
- **Result:** Creates a compact and context-aware vector (**h**) that intelligently represents the function's characteristics.
- **Benefit:** Faster and more accurate scheduling decisions in the next phase.

CASSIA Phase 2: Schedule (Adaptive Live Decision)

- **Inference:** The pre-trained encoder $\mathcal{E}$ generates the function profile (**h**).
- **Decision:** The Policy Head π uses **h** to calculate the best node (n^*), prioritizing warm containers (func_id $\in W_{n^*}$) to avoid cold starts ($C = 1$).
- **Output:** The function is executed, and its actual performance ($\mathcal{L}$) is recorded for the next phase.

CASSIA Phase 3: Fine-Tune (Online Reinforcement Learning)

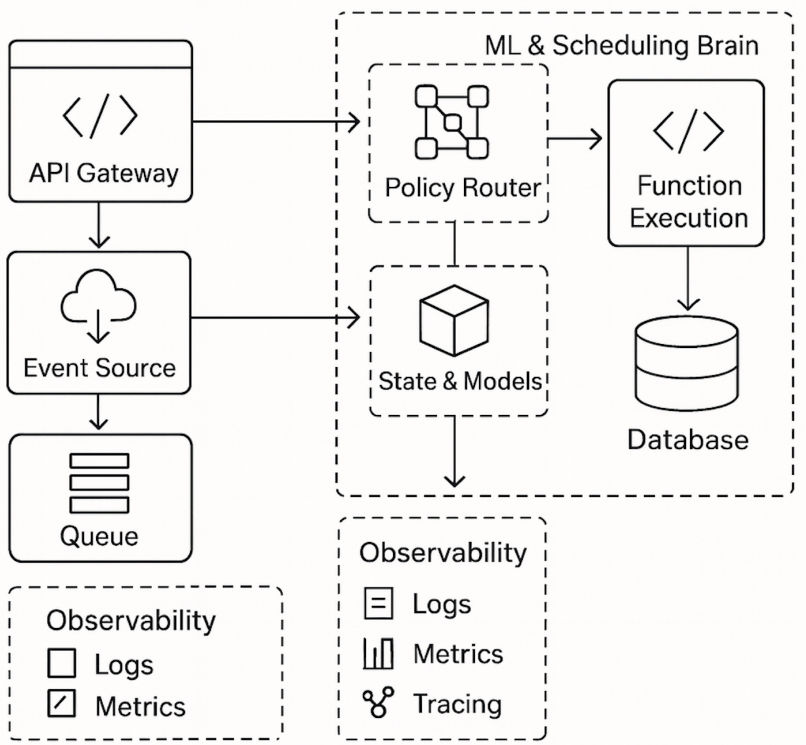
Goal: Continuously adjust the Policy Head (π) using real-time feedback (latency and cold starts) to maximize the overall system reward.

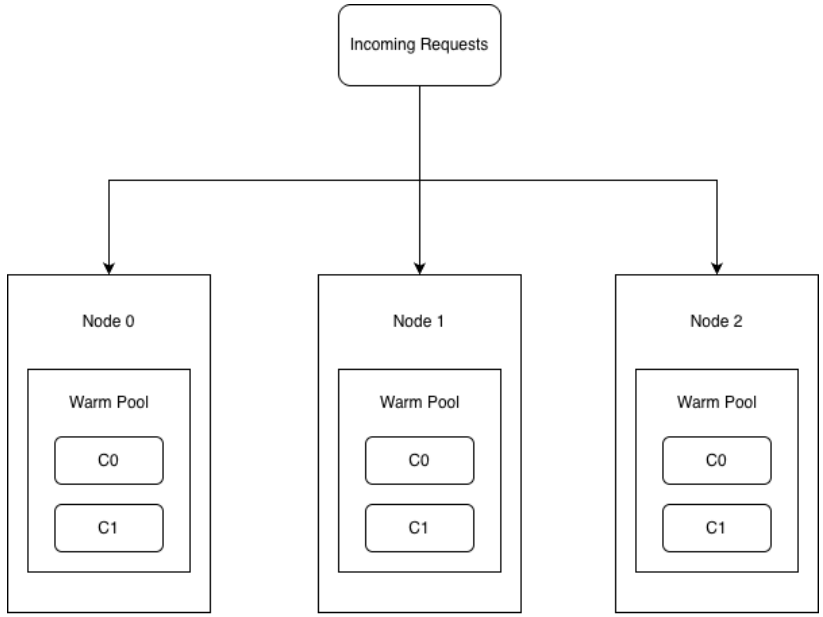
- **Reward Function (R):** Designed to penalize high latency ($\mathcal{L}$), cold starts (C), and load imbalance, encouraging efficient scheduling.
- **RL Update:** The policy (π) is updated based on the calculated reward R (e.g., using PPO), ensuring CASSIA adapts automatically to changing workloads.
- **Adaptation:** The system learns from its own execution history to become increasingly intelligent.

PARL in CASSIA + REINFORCE

Stage	RL Meaning	Scheduler Details
Perception	Observed state	invocation features + live node load (active requests) + implicit warm pool latency feedback
Action	Executed decision	pick a node, start/reuse a container, and invoke function.
Reasoning	Policy computation	Encoder (pretrained) $\rightarrow$ Policy net $\rightarrow$ Load-aware adjustment $\rightarrow$ Node decision
Learning	Policy update	Reward $r = -$(lat+load-pen)

System Architecture

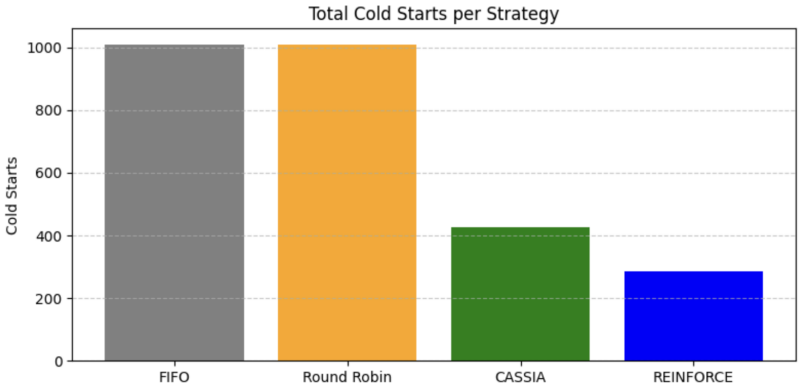




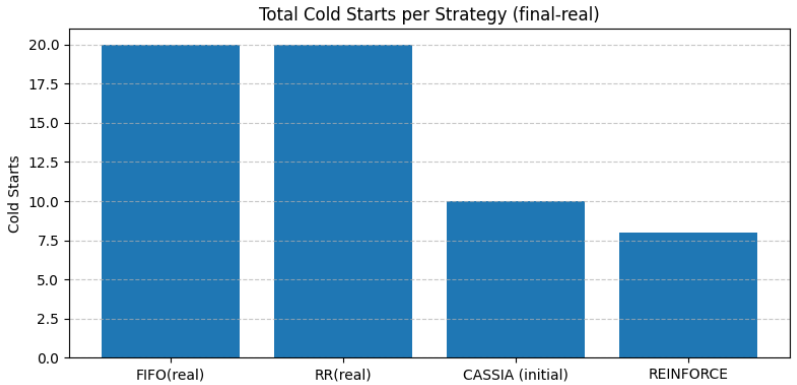
Evaluation Metrics

- **Average Latency:** Mean response time for function invocations across all nodes.
- **Cold Starts:** Total count of cold start events, indicating efficiency in reusing warm containers.
- **Baseline Comparison:** Performance evaluated against traditional **FIFO Scheduler** and **Round Robin** strategies.

Total Cold Starts per Strategy

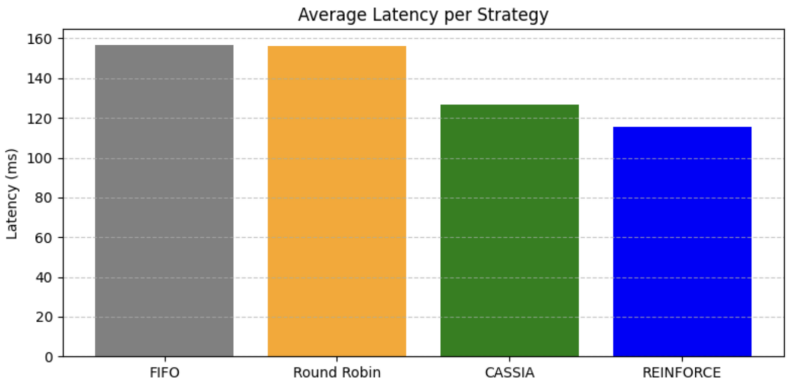


(a) Simulation / synthetic

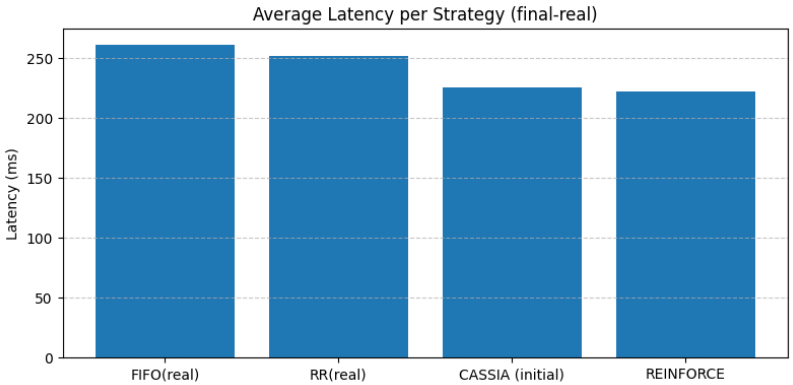


(b) Real containers

Average Latency per Strategy



(a) Simulation / synthetic



(b) Real containers

Evaluation Summary

Strategy	p50(ms)	p95(ms)	latency	Cold starts	Req. served
FIFO (real)	149.09	746.71	261.42	20	160
RR (real)	147.18	651.39	252.05	20	160
CASSIA	140.53	663.91	225.21	10	160
REINFORCE	133.12	751.77	222.09	8	160

Conclusion

Frameworks such as **ENSURE**, **SSC**, **Synergy**, and my proposed **CASSIA** have demonstrated measurable improvements in Cold Start reduction and Latency reduction.

Designed fully autonomous, adaptive, and self-optimizing schedulers that can handle workload variability.

References

[1] Amoghavarsha Suresh, Gagan Somashekar, Anandh Varadarajan, Veerendra Ramesh Kakarla, Hima Upadhyay, and Anshul Gandhi. Ensure: Efficient scheduling and autonomous resource management in serverless environments. Proceedings of the ACM Symposium on Cloud Computing, 2020.

[2] Shanxing Pan, Hongyu Zhao, Zinuo Cai, Dongmei Li, Ruhui Ma, and Haibing Guan. Sustainable serverless computing with cold-start optimization and automatic workflow resource scheduling. IEEE Transactions on Cloud Computing, 2024.

[3] Tanaya Biswas and Prashant Kumar. Optimizing resource management in serverless computing: A dynamic adaptive scaling approach. Proceedings of the IEEE International Conference on Cloud Computing, 2024.

References

[4] Hanmei Chen, Jianing You, Laiping Zhao, Yanan Yang, and Keqiu Li. Synergy: Collaborating centralized and local scheduling for serverless functions. Proceedings of the IEEE International Conference on Distributed Computing Systems, 2024.

[5] Mina Morcos and Ibrahim Matta. Inverse response time ratio scheduler: Optimizing throughput and response time for serverless computing. Boston University Computer Science Research Reports, 2023.

[6] Bartlomiej Przybylski, Pawel Zuk, and Krzysztof Rzadca. Data-driven scheduling in serverless computing to reduce response time. In 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pages 256–266. IEEE, 2021.

References

[7] Hanfei Yu, Athirai A. Irissappane, Hao Wang, and Wes J. Lloyd. Faasrank: Learning to schedule functions in serverless platforms. In Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), pages 31–40. IEEE, 2021.

[8] Nima Mahmoudi and Hamzeh Khazaei. Performance modeling of serverless computing platforms. IEEE Transactions on Cloud Computing, 10(4):2291–2305, 2022.