Intelligent Software Systems Lab
KYUNGPOOK NATIONAL UNIVERSITY

FuncMem: Reducing Cold Start Latency in Serverless Computing Through Memory Prediction and Adaptive Task Execution

Manish Pandey, Ph. D. student

Professor : Young-Woo Kwon

Introduction

- **Cold start** and higher **initialization time** are severe issues in serverless computing.
- Existing techniques extend **keep-alive time** and **prewarm containers** to alleviate performance issues; however, these techniques introduce **overhead** to the overall architecture.
- We proposed ***FuncMem*** *that*,
  - predicts memory usage and reduces the over memory requirements of functions.
  - reschedules the function in the invoker, creating an adaptive task executor queue at runtime for **non-blocking requests**.

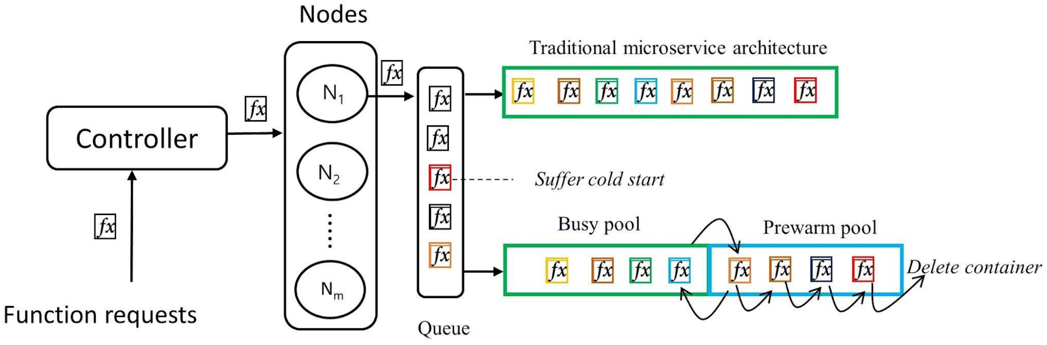


Fig: Current serverless approach

Approach

- **Scheduler**
  - Queue non-blocking requests
  - Reschedule functions based on function **deadline**, and **prewarm containers**.
- **Memory estimator**
  - Simulate invoked function
  - Predict **memory usage** and **execution time**

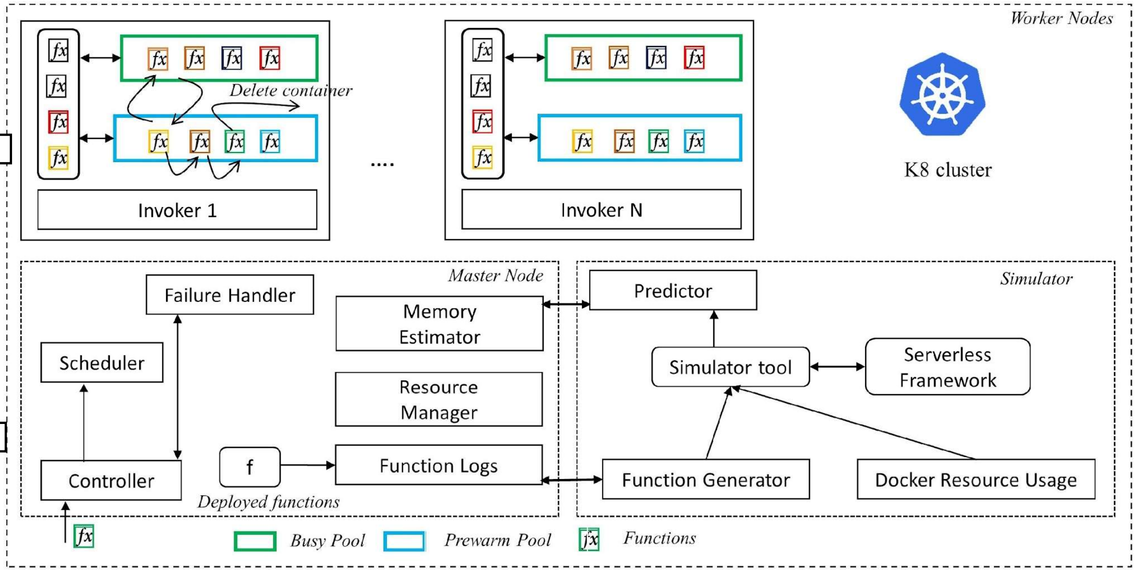


Fig: Proposed approach

System Implementation

- Framework: Openwhisk
- Total number of functions: 30
- Implemented langauge: Scala and Python
- Total invocations: 200
- Average Invocation Time: 170 seconds
- Applications: FaaS application
- Execution method: via Bash Script

Intelligent Software Systems Lab
KYUNGPOOK NATIONAL UNIVERSITY

Evaluation and Experiment results

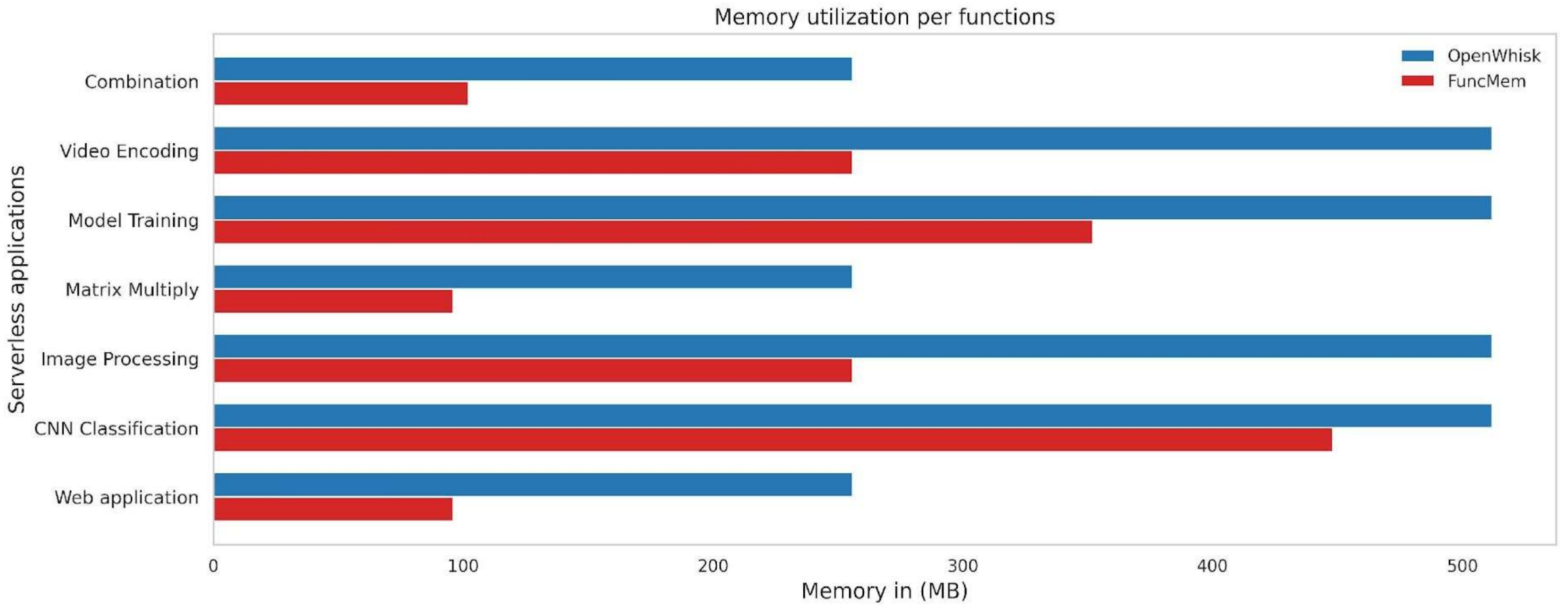
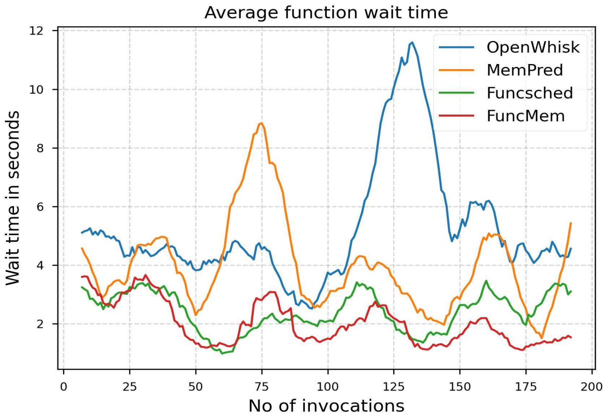
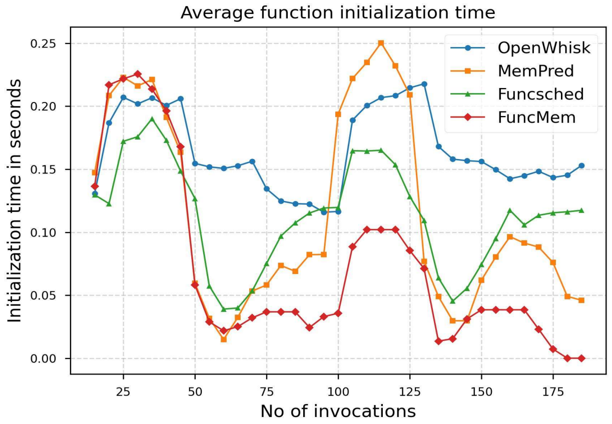
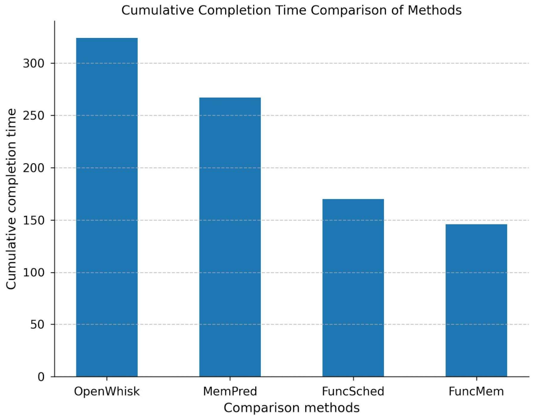


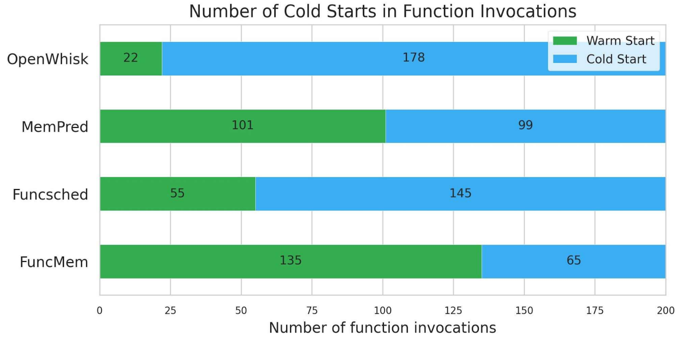
Fig: Memory utilization of FaaS applications

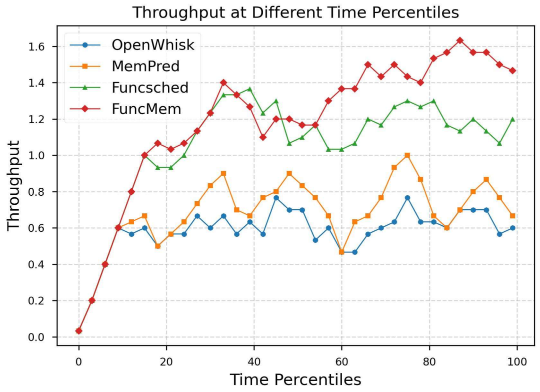
Evaluation and Experiment results


Average function wait time


Average function initialization time


Cumulative Completion Time Comparison of Methods


Number of Cold Starts in Function Invocations


Throughput at Different Time Percentiles

Conclusion and Future work

- Our approach effectively increases system throughput by minimizing memory requirements, function wait time, initialization time, overall execution time, and the number of cold starts.
- We will extend FuncMem to incorporate node selection and caching capabilities.

KNU KYUNGPOOK NATIONAL UNIVERSITY

소프트웨어재난연구센터

FuncMem: Reducing Cold Start Latency in Serverless Computing Through Memory Prediction and Adaptive Task Execution

Manish Pandey, Young-Woo Kwon
Intelligent Software Systems Lab, Kyungpook National University

Background and Approach

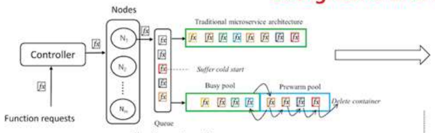


Fig: Current workflow

- **Cold start** and higher **initialization time** are severe issues in serverless computing.
- Existing techniques extend **keep-alive time** and **prewarm containers** to alleviate performance issues; however, these techniques introduce **overhead** to the overall architecture.
- We proposed ***FuncMem*** *that,*
  - predicts memory usage and reduces the over memory requirements of functions.
  - reschedules the function in the invoker, creating an adaptive task executor queue at runtime for **non-blocking requests**.

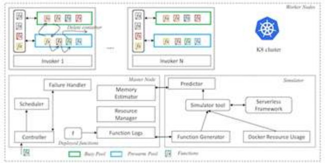
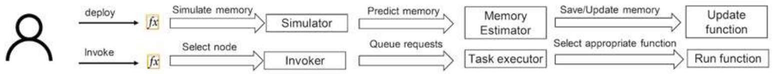


Fig: Proposed approach

- Memory estimation
  - **Simulate** invoked function
  - Predict **memory usage**, and **execution time**
- Scheduler
  - Queue non-blocking requests
  - Reschedule functions based on **function deadline**, and **prewarm containers**.

Workflow



Evaluation and Experimental Result

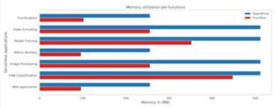


Fig: Memory utilization of FaaS applications

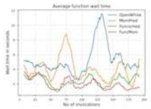


Fig: Average function wait time

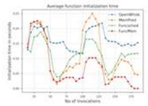


Fig: Average function initialization time

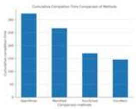


Fig: Cumulative completion time

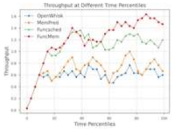


Fig: System throughput

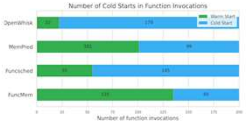


Fig: Number of cold and warm start

- Total number of functions: 30
- Implemented language: Scala and Python
- Total number of Invocations: 200
- Average Invocation Time: 170 seconds
- Framework: Openwhisk
- Application Execution: via Bash Script

Discussion and Applicability

- The performance enhancements provided by our approach make it an attractive option for IoT data processing within the serverless framework.
- To achieve widespread adoption, serverless platforms should prioritize non-blocking requests as a key area of focus.

Conclusion and Future Work

- Our approach effectively increases system throughput by minimizing memory requirements, function wait time, initialization time, overall execution time, and the number of cold starts.
- We will extend *FuncMem* to incorporate node selection and caching capabilities.

KNU KYUNGPOOK NATIONAL UNIVERSITY

Intelligent Software Systems Lab

Thank you