Cloud Research Paper Presentation

Cypress : Input size–Sensitive Container Provisioning and Request Scheduling for Serverless Platforms

Authors:
Vivek M. Bhasi (The Pennsylvania State University)
Jashwant Raj Gunasekaran (Adobe Research)
Aakash Sharma (The Pennsylvania State University)
Mahmut Taylan Kandemir(The Pennsylvania State University)
Chita Das(The Pennsylvania State University)

Presented by Group 5: Yafei Li , Sue Yang

UNIVERSITY of WASHINGTON

Contents

Intro – What is the Problem

- The majority of apps on serverless platforms process user-provided input to produce the desired results
- Many such apps have input size–dependent execution times

➢ Nevertheless, existing serverless resource management frameworks are agnostic to the input size–sensitive nature of these apps

Intro – Results the problem leads

It can be demonstrated that this problem can potentially lead to

- Container over-provisioning
- End-to-end Service Level Objective (SLO) violations.

Background & Related Work

Existing work in the area of input-sensitive profiling has shown that many such apps, which will be called Input size–Sensitive apps (IS apps), have execution times that depend heavily on the provided input size.

Challenges:

- ➔ Spawning an inappropriate number of containers
- ➔ Existing serverless Resource Management frameworks do not account for ‘buffer time’

Background & Related Work

1.Serverless Computing

Cold starts can take up a significant proportion of a function’s response time (up to tens of seconds) and can, thus, lead to SLO violations.

Ways to solve it,

- Mitigating cold start overheads
- Hiding the effects of cold starts via proactive container provisioning
- Some of these provisioning policies minimize containers by batching multiple requests onto fewer containers, as opposed to spawning one for each request.

Background & Related Work

Use Kraken[27]

- Employ a Proactive Weighted Scaler (PWS)
- Employ a Reactive Scaler (RS)

Result

- ❖ Kraken spawns up to 76% fewer containers on average, thereby, improving container utilization and cluster-wide energy savings by up to 4× and 48%
- ❖ Kraken guarantees SLO requirements for up to 99.97% of the requests

Weakness

When the execution time is input size–dependent, request batching using average execution times proves to be inaccurate, as there is considerable variation in request execution times.

This can lead to inappropriate container provisioning.

Background & Related Work

2.Input size–Sensitive Functions

Use Sensitive Profiling
(Input size–Sensitive Profiling)

(i) Performing multiple profiling runs of the function/routine with workloads spanning several magnitudes of input size

(ii) Observing the performance

(iii) Fitting these observations to a statistical model that predicts metrics (such as execution time) as a mathematical function of workload size.

IS App	Composite Functions
Sentiment Analysis	*Sentiment Analysis*
QR Code	*QR Code*
Image Compression	*Image Compression*
Email Categorization	*Text Summary → Text Classify*
Audio Translation	*Audio Transcribe → Text Translate → Send Message*

Table 2: Examples of IS apps from the AWS serverless app repository and OpenFaaS function store. The last two apps consist of multiple functions.

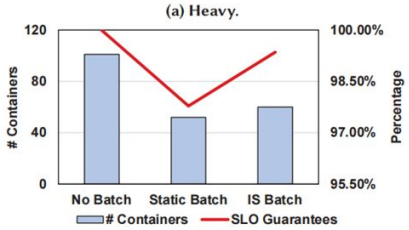
Motivation – 1

Challenge 1: Input size–Sensitive Container allocation

Solution 1: Incorporate an Input size–Sensitive container provisioning policy in the Resource Management framework

- No Batch (which spawns a container per incoming request)
- Static Batch (which uses a static batch size, based on average execution time, to batch requests)
- IS Batch (refers to batching multiple requests onto fewer containers by taking their input size–dependent execution times and respective Service Level Objectives into account.)

(a) Heavy.



(b) Light.

Figure 1: Number of containers spawned vs. SLOs satisfied for the *Sentiment Analysis* app under the heavy and light distributions for a Poisson trace (mean = 100 rps).

Motivation – 2

Challenge 2: Input size–Sensitive Request Scheduling

Solution 2:

- IS Reordering reorders requests in the incoming request queue such that requests with higher potential execution times (typically, higher input sizes) are executed first

- Input size–Sensitive Request Reordering (IS Reordering) on top of the IS Batch scheme

→ Experiment Result
IS Batch+Input size–Sensitive Request Reordering policy (**IS (Batch+RR)**)
vs. IS Batch
vs. No Batch

- Under the light distribution

  IS (Batch+RR) has improved SLO compliance compared to IS Batch (99.98% vs. 99.35%) and nearly matches No Batch's SLO compliance (99.99%)

  Static Batch provisions 13% fewer containers than IS (Batch+RR), IS (Batch+RR) greatly surpasses it in terms of SLO compliance (99.98% vs. 97.78%).

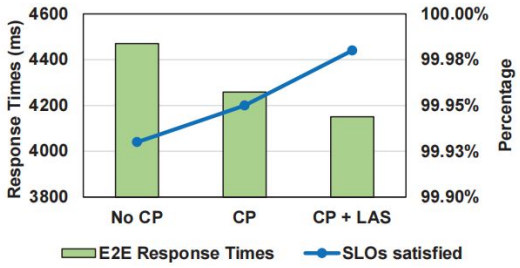
Motivation – 3

Challenge 3: Multi-Function Applications

Solution 3: Multi-function apps require additional policies that are cognizant of some aspects of their function chains to be incorporated into the Resource Management frameworks to effectively provision the requisite containers and maximize SLO compliance.

➔ Chained Prediction

➔ Experiment Result
(No Chained Prediction vs. Chained Prediction vs. Chained Prediction + Look-Ahead Scaling)



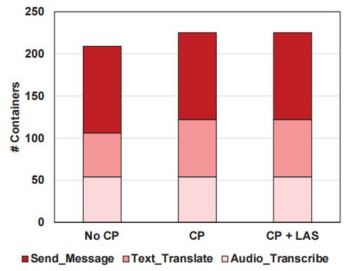


Figure 3: Breakdown of containers spawned for the *Audio Translation* app for a Poisson trace (mean = 100 rps) under the uniform input size distribution.

Input size–Sensitive Profiling of Real-World Serverless Applications

1.Mapping Input Size to Execution Time

- ***Linear Models***

$$\hat{y}_i \overset{\text{def}}{=} \hat{y}(x_i) = a + bx_i,$$

$$\sum_{i=1}^{k} r_i^2 = \sum_{i=1}^{k} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{k} (y_i - (a + bx_i))^2$$

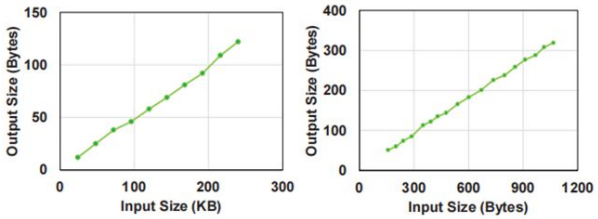
- ***Powerlaw Models:***

$$\sum_{i=1}^{k} (\log y_i - (\log a + b \log x_i))^2 = \sum_{i=1}^{k} \left(\log \frac{y_i}{a x_i^b} \right)^2$$

$$R^2 \overset{\text{def}}{=} \frac{\sum_{i=1}^{k} (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{k} (y_i - \bar{y})^2} = \frac{\left(\sum_{i=1}^{k} (x_i - \bar{x})(y_i - \bar{y}) \right)^2}{\left(\sum_{i=1}^{k} (x_i - \bar{x})^2 \right) \left(\sum_{i=1}^{k} (y_i - \bar{y})^2 \right)}$$

2.Mapping Input Size to Output Size

- ***Chained Prediction***



(a) Audio Transcribe. (b) Text Summary.

Figure 6: Input size vs. output size profiles of the IS functions that appear earlier in the multi-function apps.

Overall Design of Cypress

1. Proactive Scaler (PS)
2. Input size–Sensitive Request Batching (IS Batching)
3. Reactive Scaler (RS)
4. Look-Ahead Scaler (LAS)

Key contributions

1. **Proposed and Implemented Cypress**
- Input size–Sensitive Request Batching (IS Batching)
- Input size–Sensitive Request Reordering (IS Reordering)
- Scaling services of Cypress(the Proactive Scaler, the Reactive Scaler, the Look-Ahead Scaler)
- Chained Prediction

2. **Results**
- ❖ Cypress spawns up to 66% fewer containers, improves container utilization and cluster-wide energy savings by up to 2.95× and 23%
- ❖ Cypress guarantees the SLO requirements for up to 99.99% of requests

Experimental Evaluation

Implementation

- **Evaluation Methodology**: 6 node Kubernetes. 48 cores(Intel Cascade Lake), 192 GB RAM, 1TB Storage
- **Request Traces**:

  A synthetic trace: Poisson (The average rate μ = 250 rps)

  Real-world traces: Twitter(peak-to-mean ratio 5450:3139); Wiki(331:302)
- **Size**: Heavy(50-55%), Medial, Light(~33%, descending)
- **Applications:** 5 apps (3 are single-function, 2 are multi-function)
- **Container Provisioning Schemes Baselines:** Atoll, Kraken, Fifer
- **Large Scale Simulation:** Build a **simulator** in Python using container cold start latencies and function execution times profiled from real system counterpart. ~6.4K core cluster, ~5500 requests

Experimental Evaluation

Real System Results

Containers Spawned vs. SLOs Satisfied

- Single-function apps—-Cypress spawns much fewer containers than Atoll (**66% fewer containers in light distribution**)
- Multi-function apps—-Cypress spawns fewer containers than 3 other schemes

End-to-End Response Times and Latency Distribution

- Cypress is 4% slower than Atoll, but 15%, 10%, 6% faster than Fifer, Kraken and IS Batch.

Container Utilization

- **Cypress has 52% and 71% more container utilization** compared to Atoll for the heavy and light distributions, respectively.

Energy Efficiency

- Cypress consumes 19% and 22% less energy than Atoll for the heavy and light distributions, respectively

Resilience to Erratic Traces

- Cypress outperforms than others

Experimental Evaluation

Simulator Results

Heavy Distribution

- Cypress spawns fewer containers than all other schemes (up to 43% fewer)

Light and Medial Distribution

- Cypress spawns 65% fewer containers than Atoll
- Performances vary from different apps when compared to Kraken/Fifer

Cypress spawns 15% more containers than Kraken/Fifer for Image Compression for the light distribution for the stable trace

Author's Conclusions

The Performance of Cypress

- 66% fewer containers spawned
- Improving container utilization by up to 2.95x
- Cluster-wide energy savings by up to 23%

Critique: Strengths

Big improvements

- 66% fewer containers
- Improving container utilization by up to 2.95x
- Cluster-wide energy savings by up to 23%

Using a simulator in a large scale to cross-verify the results

Critique: Weaknesses

- In conclusion part, the improvement of container utilization by up to 2.95x is not mentioned in any experiment results.
- In simulator, compared to Kraken and Fifer, the improvements on containers spawned of Cypress are slight.

Critique: Evaluation

- In large scale simulation, it does not mention how they build the simulator based on the real system.
- The input sizes are randomly generated. Why can they be defined as heavy, medial or light?

GAPS

- To predict input size distribution, the least-squares linear regression is not compared to other ways, eg.Gradient descent(If the distribution is not linear).
- For light and medial distributions, how to improve Cypress's performances on containers spawned when compared to Kraken/Fifer

  Cypress spawns 9% more containers than Kraken/Fifer for Image Compression for the medial distribution

Thanks for listening

Questions?