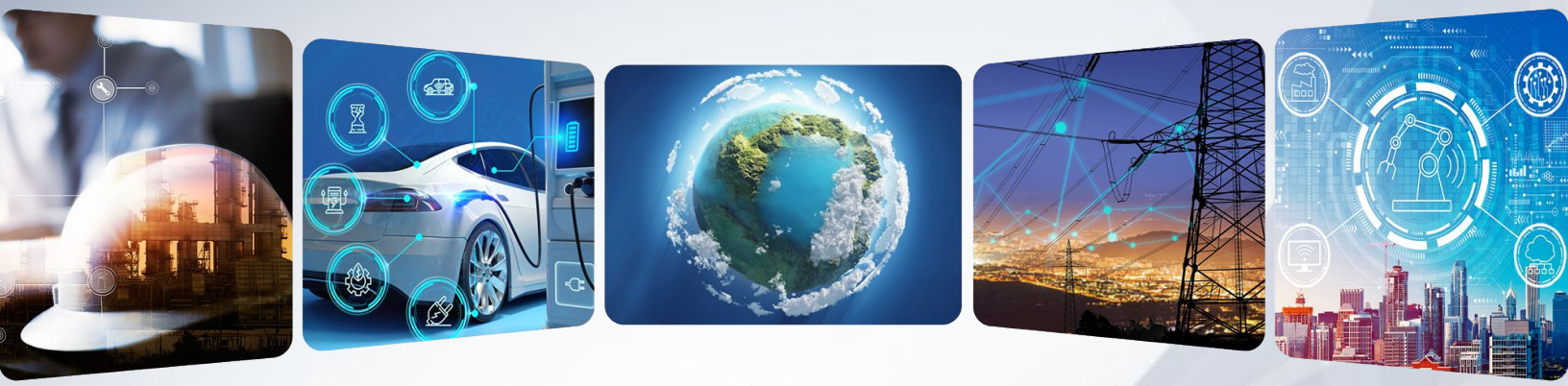


Towards Low-Cost Global Highly Available Large Container-Based Serverless Functions

Jasleen Kaur, Robert Cordingly, Ling-Hong Hung, Wes Lloyd
School of Engineering and Technology
University of Washington Tacoma, Washington USA

IEEE
Advancing Technology for Humanity

OUTLINE

- Introduction
- Problem Summary / Motivation
- Research Questions
- Related Work
- Methodology
- Results
- Conclusion

INTRODUCTION

Container based serverless functions

- **To improve availability and minimize latency, serverless functions can be deployed across multiple regions closer to clients**
- **AWS Lambda supports container-based functions**
  - **Container-based functions are packaged and deployed using container images**
  - **Images must be hosted in regional Amazon Elastic Container Registries (ECR)**

INTRODUCTION - 2

Container based image replication

- **Deploying container-based functions globally across regions features high costs.**
  - **ECR storage costs: 10c per GB per region**
- **To replicate a 10 GB container image across 25 AWS regions and to enable 24/7/365 global high availability for a container-based function costs $300/annually.**

INTRODUCTION - 3

Container Image Replication – ECR CRR

- **For high availability, a container image is required in each region's ECR. Container image replication is expensive and slow across multiple regions.**
- **Amazon launched ECR Cross Region Replication (CRR)**
  - Replicate container images across regions in Amazon Elastic Container Registry (ECR) automatically
  - Only repository content pushed to a repository after CRR is configured is replicated. Pre-existing content in a repository isn't replicated.
  - Filtering based on repository name
    - The replication configuration supports filtering private ECRs by specifying a repository prefix.

PROBLEM SUMMARY

ECR CRR Tag Level Filtering

- **CRR helps with faster replication but there are problems:**
  - All new repository content pushed after ECR CRR is enabled is copied
  - All new repositories after ECR CRR is enabled are also copied.
    - *CRR may replicate more images than required*
  - Repository level filtering but no image level filtering
  - Reported Issue
    - **https://github.com/aws/containers-roadmap/issues/1737**

MOTIVATION

User complains about image-level replication

RESEARCH QUESTIONS

- **RQ1: How should container images be replicated for on-demand use to support multi-region high availability of container-based AWS Lambda functions?**
  - **For viable solutions, we calculated the ensuing storage costs in S3 or ECR, and transfer costs to make the container image available for compute.**
  - **We compared various methods for image replication and found the most cost-effective approach.**

RESEARCH QUESTIONS

- RQ2: How should container-based serverless functions be best created on-demand once a container image is made available locally in the region for execution?
  - We propose using a proxy function approach to initiate deployment of container-based functions in regions lacking the container image
  - This function manages image transfer and function creation, with control transferred to the new function post-deployment.

RESEARCH QUESTIONS

- RQ3: What are the cost implications for container image replication for different image retention policies and image replication methods?
  - Goal: Understand worst-case cost
  - We evaluated cost given different usage scenarios and image retention policies
    - Usage scenarios define different invocation frequencies, (e.g. 3x/hour, 3x/day)
    - Image retention policies define how long an image should be retained in a regional ECR (e.g. one hour, one day, one week)

RELATED WORK

FaaS Deployment tools for pushing updates to different clouds/regions

- **GlobalFlow [1] is a workflow orchestration service**
  - **Operates across multiple regions and effectively coordinates the execution of geographically distributed AWS Lambda functions that are necessary for cloud-based applications.**
- **Uses copy-based approach**
  - **Lambda functions are copied from various regions to a target region to generate and execute a new workflow job in the target region.**
- **Focus on bringing compute to data**

RELATED WORK - 2

Data Availability : On-demand container loading in AWS Lambda [2]

- **Proposes method for container data to be lazily copied**
  - **Only data that is used is copied on-demand**
- **Improves image availability from the backend by just-in-time data copy**
- **Storage and caching system**
  - **Optimize delivering container images on-demand**
  - **Incorporating caching, deduplication, convergent encryption, erasure coding, and block-level demand loading.**

RELATED WORK - 3

Other Cloud providers : Google cloud functions [3]

- **Google cloud functions allows developers to package code as a container image and host them onto Artifact registries for use with the Google Cloud Run (GCR) service**
- **GCR doesn't support automatic container image replication**
- **Cost of storage in Artifact registry is comparable to AWS ECR $0.10/GB/month**

METHODOLOGY

Task 0: Preparing a large workload

- **Created container Image containing SQLite Database**
- **Runs the query**
  - select count(*) from (select * from nyc_speed LIMIT 10000)
- **Database contains 10M rows of NYC RealTimeTraffic Speed Data [4]**
- **9.97 GB container size**

METHODOLOGY - 2

Task 1: Implement ways to replicate data across regions on-demand

- **To address RQ-1, we evaluated the following methods to move container images to local regions on demand for high availability of data:**
  - **By storing supporting files to build docker image in S3, and push the image to ECR** (Build_DockerImage_From_S3_SourceFiles)
  - **Implementing a trigger to replicate container images in private ECR registries (**Transfer_DockerImage_Using_ECR)
  - **Storing the docker image in S3 and triggering replication in ECR** (Transfer_DockerImage_From_S3)
  - **Using 3 services**: AWS CodeBuild, Github Actions, EC2

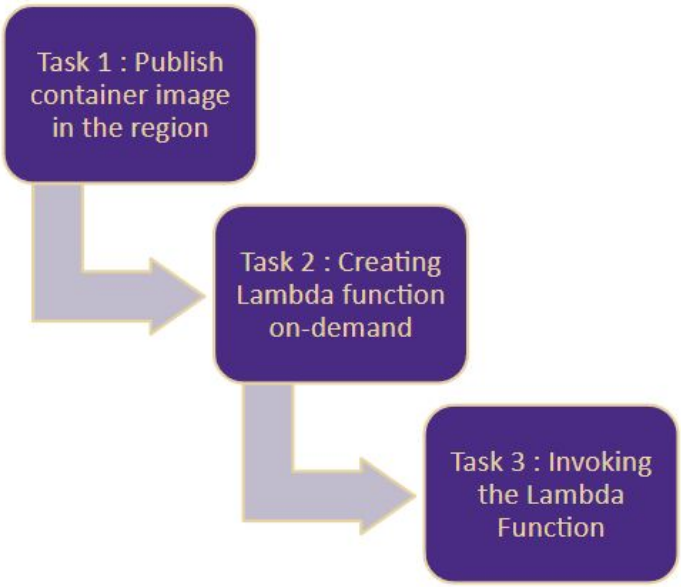
METHODOLOGY - 3

Task 2: Implement triggered Lambda function creation
Task 3: Invoking the newly created Lambda function

- **To address RQ-2, we prototyped mechanisms to transfer control from one Lambda function to another.**
- **Evaluated how much additional time is required:**
  - **To pass-off control of the current function to the newly created function**
  - **After Task 1 triggered the creation of the new container-based function**
  - **To invoke the function**

EXPERIMENT 1

Lifecycle of container based function



METHODOLOGY

Task 0 : Prepare a large workload

- **Created container Image containing SQLite Database**
- **Runs the query : select count(*) from (select * from nyc_speed LIMIT 10000)**
- **Database contains 10M rows of NYC RealTimeTraffic Speed Data [4]**
- **9.97 GB container size**

Task 1 : Publish container image in the region

- **Explored different ways to transfer 10 GB container image to destination ECR and computed costs and transfer time (to address RQ-1):**
  - Build_DockerImage_From_S3_SourceFiles
  - Transfer_DockerImage_Using_ECR
  - Transfer_DockerImage_From_S3
- **Using 3 Different services:** AWS CodeBuild, Github Actions, AWS EC2

Task 2 : Create Lambda function on-demand

- **Prototyped proxy function to transfer control from one Lambda function to another (to address RQ-2)**
- **Evaluated how much additional time is required:**
  - **To pass-off control of the current function to the newly created function**
  - **Triggered the creation of the new container-based function after Task1**

Task 3 : Invoke the Lambda Function

- **Evaluated how much additional time is required:**
  - **To invoke the newly created container based function**

RQ1 - Experiment 1

Tools and Services

- **To address RQ-1, we evaluate the following methods to move container images to local regions on demand for high availability of data:**
  - **By storing supporting files to build docker image in S3, and push the image to ECR** (Build_DockerImage_From_S3_SourceFiles)
  - **Implementing a trigger to replicate container images in private ECR registries** (Transfer_DockerImage_Using_ECR)
  - **Storing the docker image in S3 and triggering replication in ECR** (Transfer_DockerImage_From_S3)
- **Using three different services :**
  - AWS CodeBuild
  - Github Actions
  - AWS EC2

RESULTS

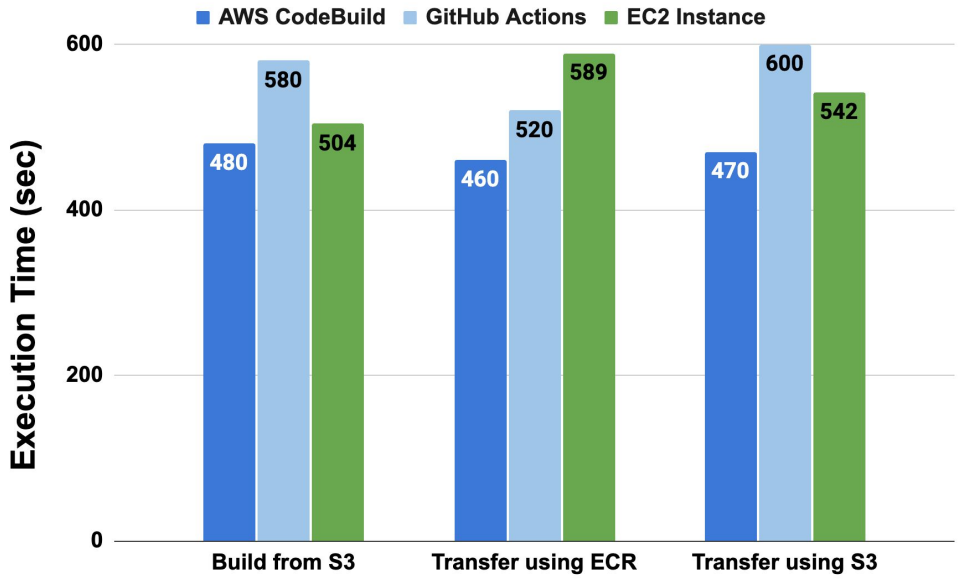
Experiment - 1: Computed associated transfer time and cost incurred

Method	Build_DockerImage_From_S3_SourceFiles		Transfer_DockerImage_Using_ECR		Transfer_DockerImage_From_S3	
	Time (sec)	Cost (cents)	Time (sec)	Cost (cents)	Time (sec)	Cost (cents)
AWS CodeBuild	480	16	460	18	470	15.8
Github Actions	580	22 (14.5)†	520	99 (93)†	600	31 (23.5)†
EC2 Instance M5zn.2xlarge	503.6	9.2	588.49	14.1	541.88	10.1

†- Free tier cost - 2000 Build mins are free with Github Actions

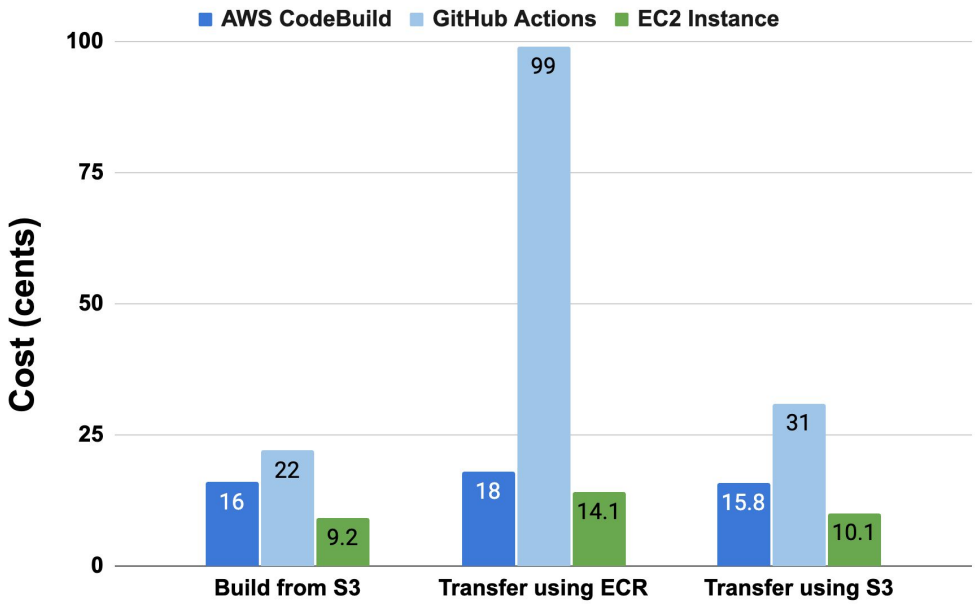
RESULTS - 2

Comparing Execution Times: publish 10GB image



RESULTS - 3

Comparing Execution Cost: publish 10GB image + 1d storage



RQ2 - Results from Experiment 2

Benchmarking function creation + invocation time (Once the 10GB container image is replicated and made available to local ECR)

Scenario	Time (s)
Min time to publish a new container based lambda function with the image on demand	17.02
Invoking the newly created lambda function	0.540
Total turnaround time (reflecting cold start latency for 10 GB container based lambda)	17.56

Defining Retention Policies

Goal: Evaluate worst case costs to provide container images to execute container based functions considering different image retention policies and invocation frequencies

- **Considered the following invocation frequencies:**
  - **3x/day, 3x/week, 3x/month**
- **Considered the following retention policies:**
  - **1 hour, 1 day, 1 week, 1 month**
- ***E.g.: If we invoke the function call three times a day, say once every 8 hours, and retain the image only for 1 hour***

RQ3 - Experiment 3

Usage Scenarios

- We considered various function usage scenarios
  - Frequency of invocation based on a time window
    - X invocations by Y time (e.g. 3 calls/day, 4 calls/week)
- For each scenario we evaluated different image retention policies (e.g. after use, retain image for 1 hr, 4 hrs, 1 day,..)
- We assumed the widest distribution of calls to determine the worst case cost outcome from the perspective of container image storage (caching)

Experiment 3 - cont'd

- To calculate cost, we count the # of image replications required for usage scenarios with different retention policies

- We assumed every function invocation is a 'cache MISS' to evaluate the worst case scenario for COST
  - Costs:
    - image storage in ECR for X time
    - data transfer for Y image replications
    - Z charges for build platform costs
      - EC2 , AWS CodeBuild, Github Actions

Usage Scenarios: Retention Time and Yearly Image Replications

Invocation Rate	Retention time (%) and annual ECR storage time				Calls Per Year	Number of image replications per year (Only replicate images for the misses)			
	1 hr	1 day	1 week	1 month		1 hr	1 day	1 week	1 month
Usage Scenario A 3x / day	12.5% (1.5 mo/yr)	100% (12 mo/yr)	100% (12 mo/yr)	100% (12 mo/yr)	1095	1095	1	1	1
Usage Scenario B 3x / week	1.79% (.21 mo/yr)	42.85 % (5.13 mo/yr)	100% (12 mo/yr)	100% (12 mo/yr)	156	156	156	1	1
Usage Scenario C 3x / month	0.42% (.05 mo/yr)	10% (1.18 mo/yr)	69.90% (8.3 mo/yr)	100% (12 mo/yr)	36	36	36	36	1

RESULTS

Invocation Rate	Yearly cost of replication							
	Using EC2 ($)				Using AWS CodeBuild ($)			
	1 hr	1 day	1 week	1 month	1 hr	1 day	1 week	1 month
Usage Scenario A 3x / day	$119.54	$12.11	$12.11	$12.11	$163.56	$12.15	$12.15	$12.15
Usage Scenario B 3x / week	$17.38	$22.29	$12.11	$12.11	$23.30	$28.22	$12.15	$12.15
Usage Scenario C 3x / month	$4.01	$5.14	$12.26	$12.11	$5.38	$6.51	$13.63	$12.15

RESULTS - 2

Invocation Rate	Yearly cost of replication Using Github Actions ($)			
	1 hr	1 day	1 week	1 month
Usage Scenario A 3x / day	$1,059.38	$12.97	$12.97	$12.97
Usage Scenario B 3x / week	$150.93	$155.84	$12.97	$12.97
Usage Scenario C 3x / month	$34.83	$35.96	$11.67	$12.97

CONCLUSION

- RQ1 : Transferring image to the region
  - > AWS Codebuild was the fastest option with data transfer times for large images:
    - Transfer_DockerImage_Using_ECR as 460 seconds
  - > Using EC2 (m5zn.2xlarge VM) was the cheapest, cost for large image:
    - Build_DockerImage_From_S3_SourceFiles as 9.2 cents
- RQ2: Function cold start latency after image replication
  - > Total turnaround time 17.56 seconds
- RQ3: Costs of retention policies
  - > Cheapest to retain the container image for 1 hour
    - Usage scenario C, invoke 3x/month with EC2 Build methodology ($4.01)

REFERENCES

[1] Zheng, G. and Peng, Y., 2019, July. GlobalFlow: a cross-region orchestration service for serverless computing services. In 2019 IEEE 12th International Conference on Cloud Computing (CLOUD) (pp. 508-510). IEEE.
[2]https://www.usenix.org/system/files/atc23-brooker.pdf
[3]https://cloud.google.com/run/docs/deploying
[4]https://www.kaggle.com/datasets/aadimator/nyc-realtime-traffic-speed-data

THANK YOU