Evaluation of Network File System as a Shared Data Storage in Serverless Computing

Jaeghang Choi* and Kyungyong Lee
Department of Computer Science
Kookmin University, South Korea

Sixth International Workshop on Serverless Computing (WoSC6) 2020
https://www.serverlesscomputing.org/wosc6/#p5 // Dec 8, 2020

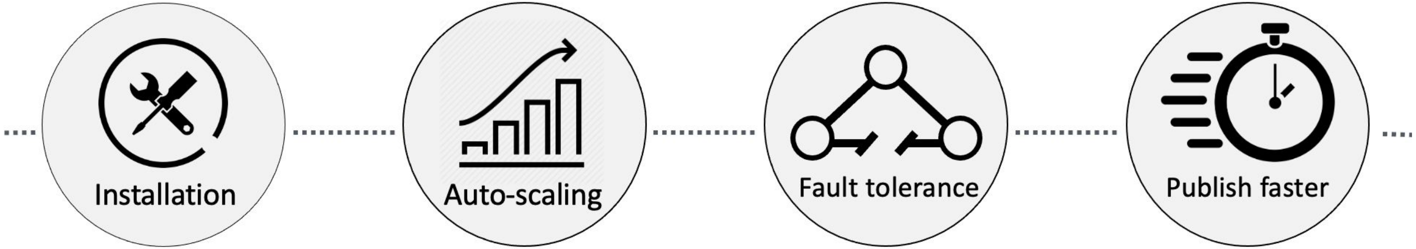
bigDATA Lab.
KOOKMIN UNIVERSITY

Serverless Computing

A software architecture model:
Cloud vendors operate servers and **dynamically manage computing resource allocations**

Role

- Developers: Just implement their workload as a function run-time.
- Cloud Vendor: Managing server considering scalability and reliability.



Function-as-a-Service (FaaS)

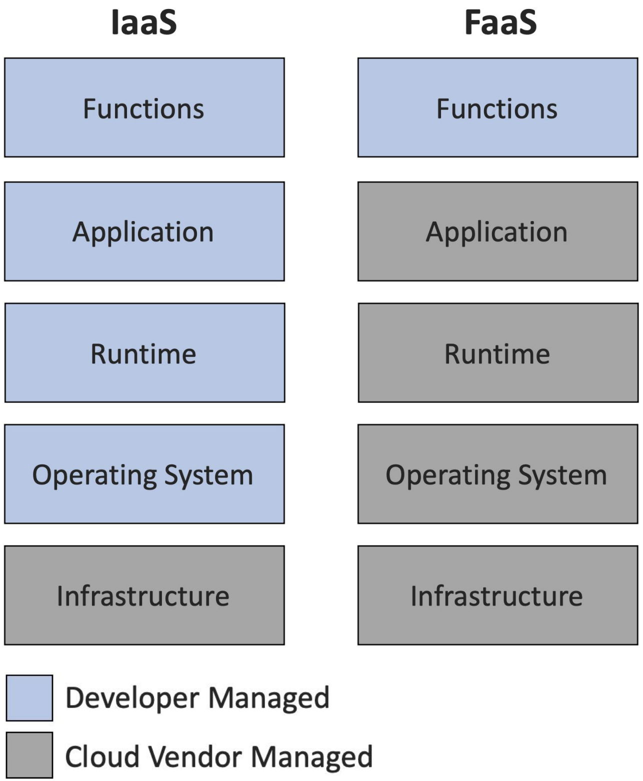
The core component of serverless computing

Developers just register function in cloud

- Python, Go, C, Javascript, Java

Characteristics

- Application composed multiple functions
- Triggered by event
- Pay-as-You-go
- Fully-managed

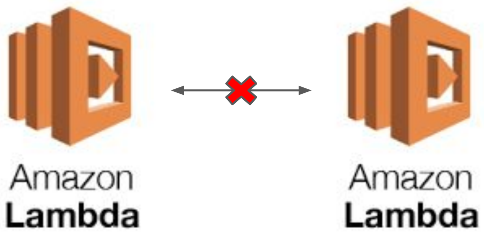


Challenges of Serverless Computing

Disadvantage of Serverless Computing

- Cold start issue causing performance variation
- No specialized hardware support

- No support **Peer-to-peer**(P2P) communication between function run-time



To overcome the limitation of P2P communication

- easy way is using **storage**

bigDATA Lab.
KOOKMIN UNIVERSITY

Major Contributions

Qualitative comparison of various cloud storage services
as intermediate FaaS storage

Quantitative evaluation of NFS as a FaaS ephemeral storage service

Types of Storages for Serverless Computing

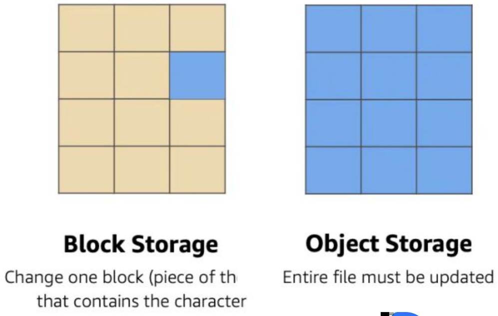
Object storage

- Cheap & slow
  disk-based object storage services (AWS S3).
- Expensive & fast
  RAM-based caching services (AWS Redis).
- Demerit: only **entire file** should be uploaded or downloaded.

Block storage

- AWS EFS, EBS
- **No support** for P2P communication
  in serverless computing
- Merit: **byte-level file access** (data-intensive applications)

Storage Type	Speed	Price
Object (S3)	Slow	$
Caching (Redis)	Fast	$$$



Using EFS for AWS Lambda

Amazon Elastic File System: Network file system(NFS) supported by AWS

AWS Compute Blog

Using Amazon EFS for AWS Lambda in your serverless applications

by James Beswick | on 18 JUN 2020 | in Amazon EC2, Amazon Elastic File System (EFS), Amazon VPC, AWS Cloud9, AWS Lambda, AWS Serverless Application Model, Serverless, Technical How-To | Permalink | Comments | Share

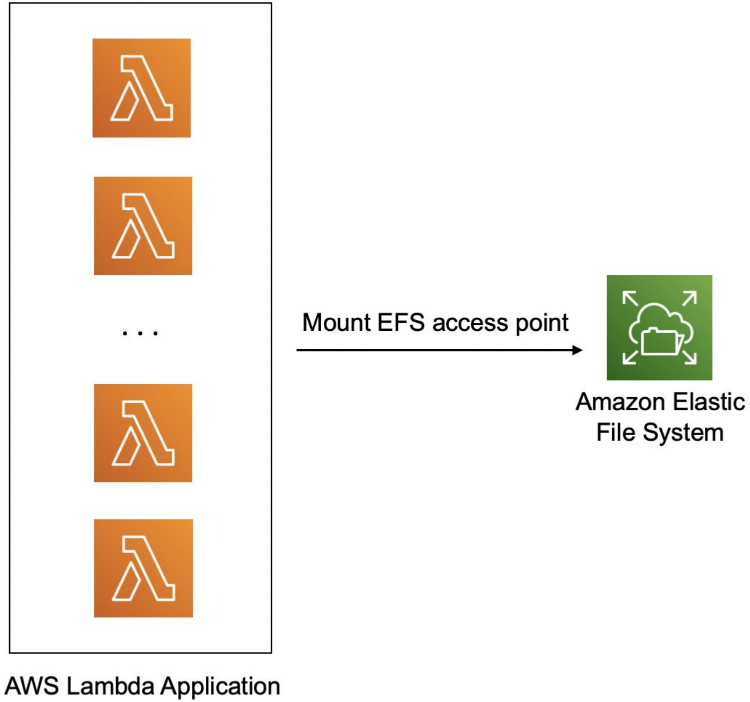
Serverless applications are event-driven, using ephemeral compute functions to integrate services and transform data. While AWS Lambda includes a 512-MB temporary file system for your code, this is an ephemeral scratch resource not intended for durable storage.

Amazon Elastic File System(EFS)

Characteristics of EFS

- Fully-managed
- **POSIX file system APIs**
  (similar local storage disk)

- Two throughput modes
  - Bursting
    : scales with consumed storage size
  - Provisioned
    : purchase additional throughput



Characteristics of NFS as a cloud storage

Network File System(NFS)

- Faster than Object Storage
- Cheaper than Object Caching Storage
- Dataset stored permanently
- Access files in byte-level

Storage Type	Speed	Price	Access	Persistence
Object (S3)	Slow	$	Object	Permanent
Caching (Redis)	Fast	$$$	Object	Permanent
Local disk	Fast	0	Block	Temporary
NFS (EFS)	**Fast**	**$$**	**Block**	**Permanent**

Experiment with EFS

Goal: Understand performance characteristics of NFS as a FaaS storage service

Environment

Function

- AWS Lambda: Python 3.6, boto3

Storage

- Lambda Layer(maximally available size: 250MB)
- AWS S3
- Amazon EFS: provisioned mode - throughput: 100MB/S

Dataset

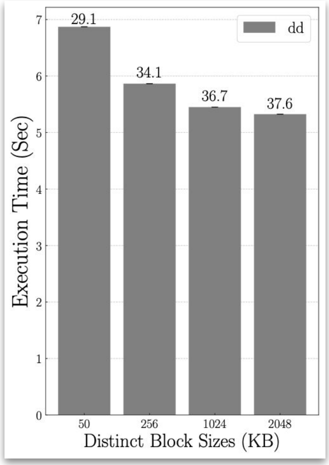
- Target File: dummy data - size: 200MB

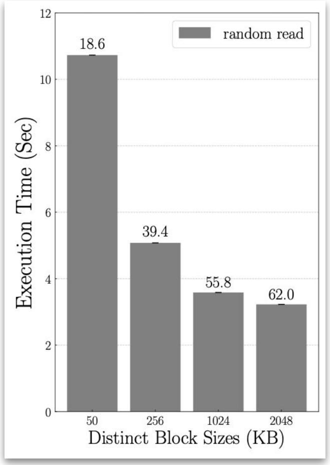
Impact of Block Sizes for EFS

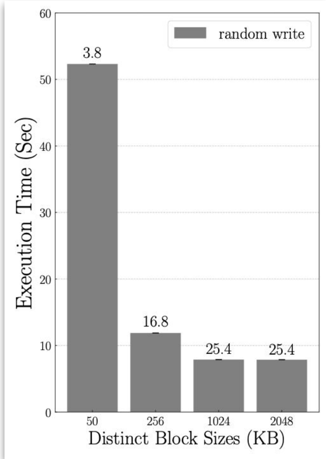
Block sizes under evaluation: 50KB, 256KB, 1MB, 2MB

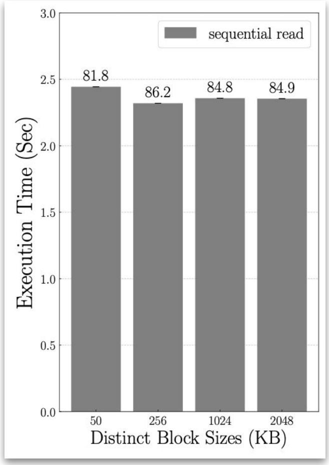
Single function run-time does not fully utilize available bandwidth (100MB/S)

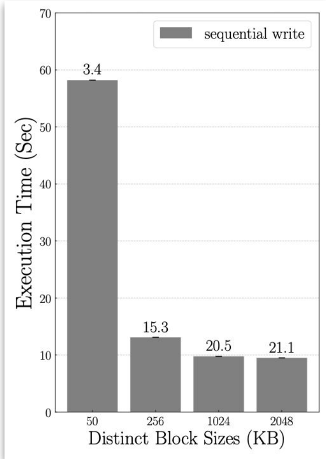
- dd (37.6 MB/S), random read (62MB/S), random write (25.4 MB/S), sequential read (84.9 MB/S), Sequential write (21.1 MB/S)







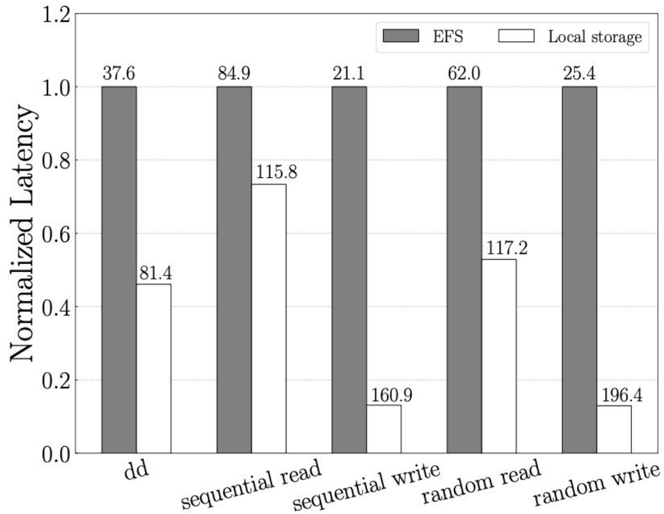




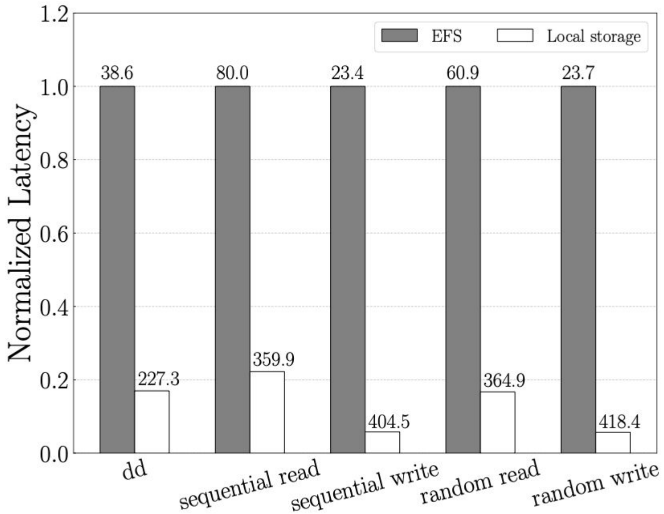
Comparing NFS with a Local Storage

Performance impact from different Lambda memory (512MB → 2GB)

- Local storage : 2~3 times more throughput achieved
- **EFS: no difference**



Lambda RAM 512MB



Lambda RAM 2G

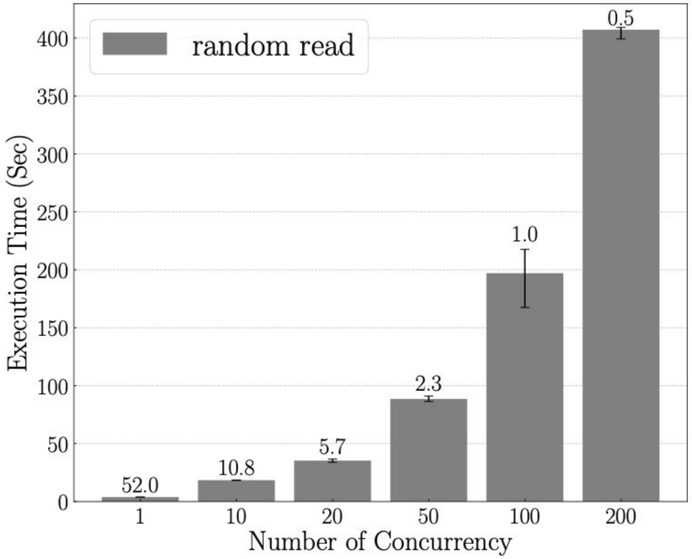
bigDATA Lab.

Evaluation of scalability of EFS

Multiple function request simultaneously

- Representation of workload: random read
- Lambda: 512MB, Block Size: 2MB

- Inverse proportion increasing requests and function bandwidth
- The latency increases linearly with respect to the concurrent access



Total Bandwidth(100MB/S) = Number of Concurrent * Each Function Bandwidth

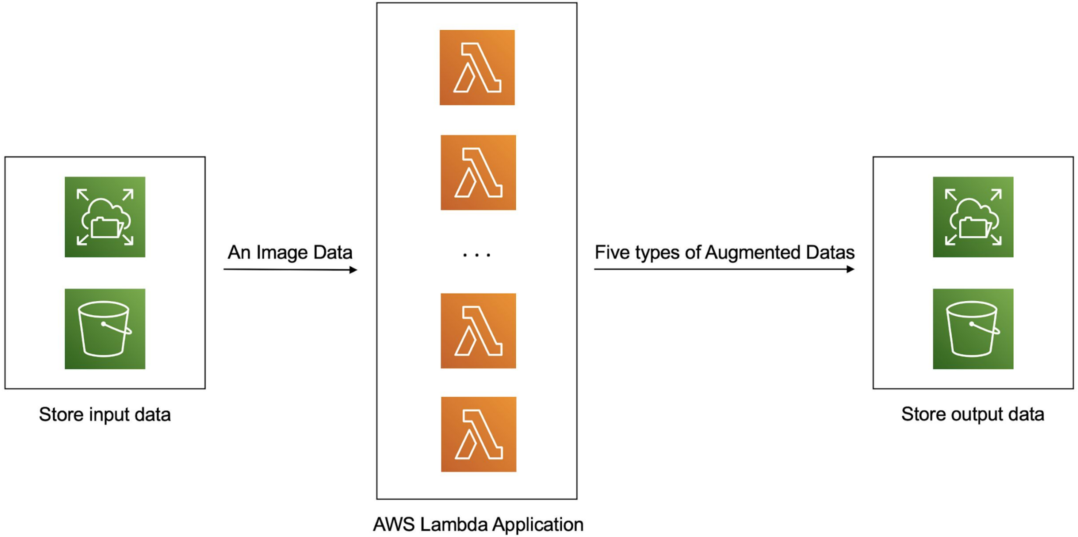
bigDATA Lab. KOOKMIN UNIVERSITY

Comparing NFS with an Object Storage

Performed Image augmentation tasks in python pillow library

5 types of Image Augmentation

- flip left-right
- flip top-bottom
- grayscale
- rotate 90 degrees
- rotate 180 degrees

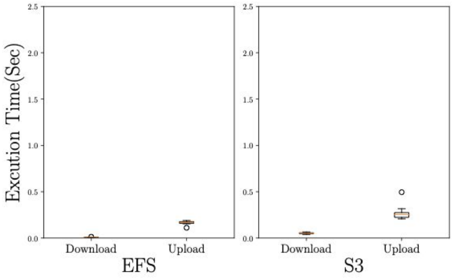


Number of Upload tasks = Number of Download tasks * **5**

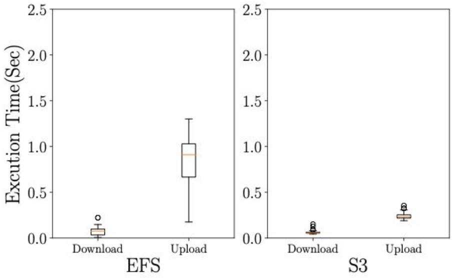
Comparing NFS with an Object Storage

Comparing image upload and download latency of EFS and S3 with a different number of parallel executions

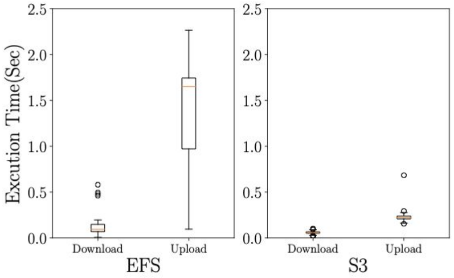
- EFS show severe degradation as parallel requests increase



10 input images



50 input images



100 input images

The Coefficient of variation(CoV)

- the ratio of the standard deviation to the average.

Number Of Images	EFS	S3
10	0.13	0.31
50	0.29	0.14
100	0.41	0.22

Cov of upload task

bigDATA Lab.
KOOKMIN UNIVERSITY

Conclusion and future work

Quantitative comparison of various cloud storage services for ephemeral storage for FaaS

Qualitative evaluation of EFS under various scenarios

- **Limited single function bandwidths** comparing to a local storage
- Bandwidths sharing among **parallel function invocations**
- Noticeable performance degradation when **multiple function run-times access**

Future work

- Discover new application scenarios using EFS with Lambda, such as MapReduce
- Understanding consistency performance of EFS with Lambda

Q&A

contact: chl8273@kookmin.ac.kr