CSE 291 - Virtualization

Video Processing on Serverless

Group 15

Mohit Shah (A59005444)

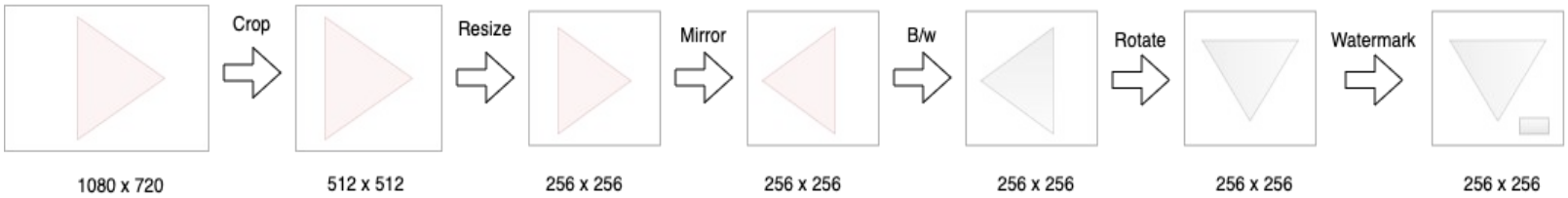
Gursharan Ahir (A59012151)

Soham Pachpande (A59001664)

Video Processing Pipeline

- The pipeline performs operations like crop, rotate, mirror etc. on a given input video in order to synthesise data for ML-related applications

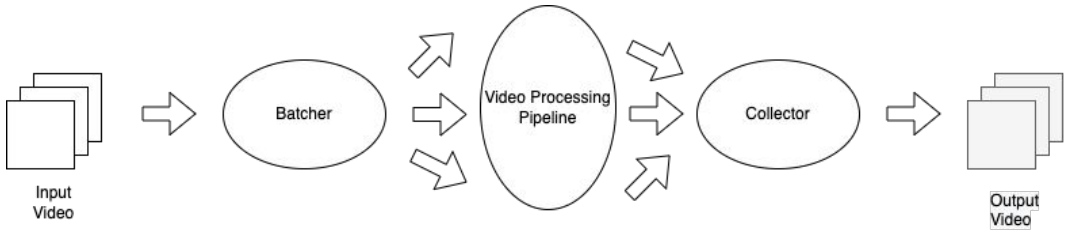
- The pipeline is implemented using ffmpeg-based moviepy[1] python library



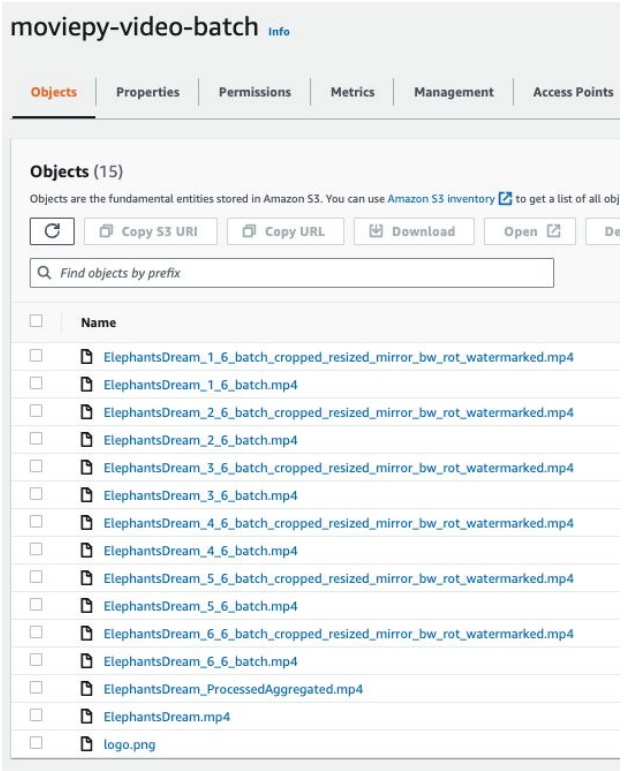
1- https://pypi.org/project/moviepy/

Batching

- Irrespective of the memory-cpu configuration, the serverless video pipeline would timeout for videos greater than a threshold (since lambda has 15 minutes timeout)
- We introduce batching and collection mechanism with variable batch length to support even larger input videos



Video pipeline with batching capability integrated

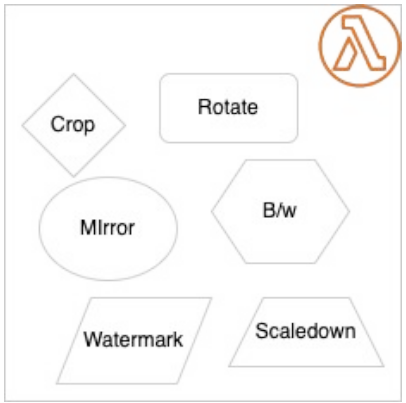


S3 bucket with objects created by our batching capability integrated video-processing pipeline

Approaches

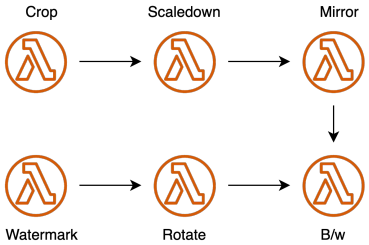
Monolithic

- All stages within a single Lambda function



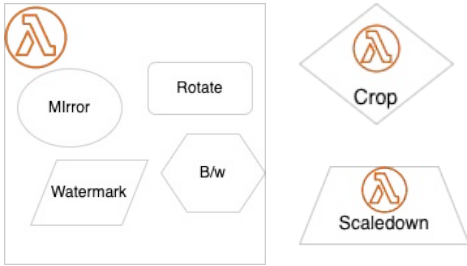
Manual Splitting

- Separate Lambda Functions for each independent step



Strategy-based Splitting

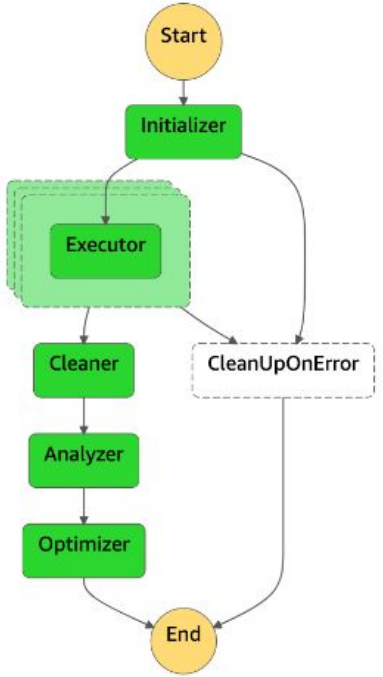
- Aggregating Small Video Processing Steps



AWS Step Functions and Log Insights for Profiling

- AWS Step Functions allows users to orchestrate workflows on AWS Lambda as per defined configuration
  - We used Lambda Power Tuning[1]
- Allowed us to rapidly experiment with various Lambda configurations of our video processing pipeline
- Deployed ~120 Step Function Executions on over 20 Lambda functions

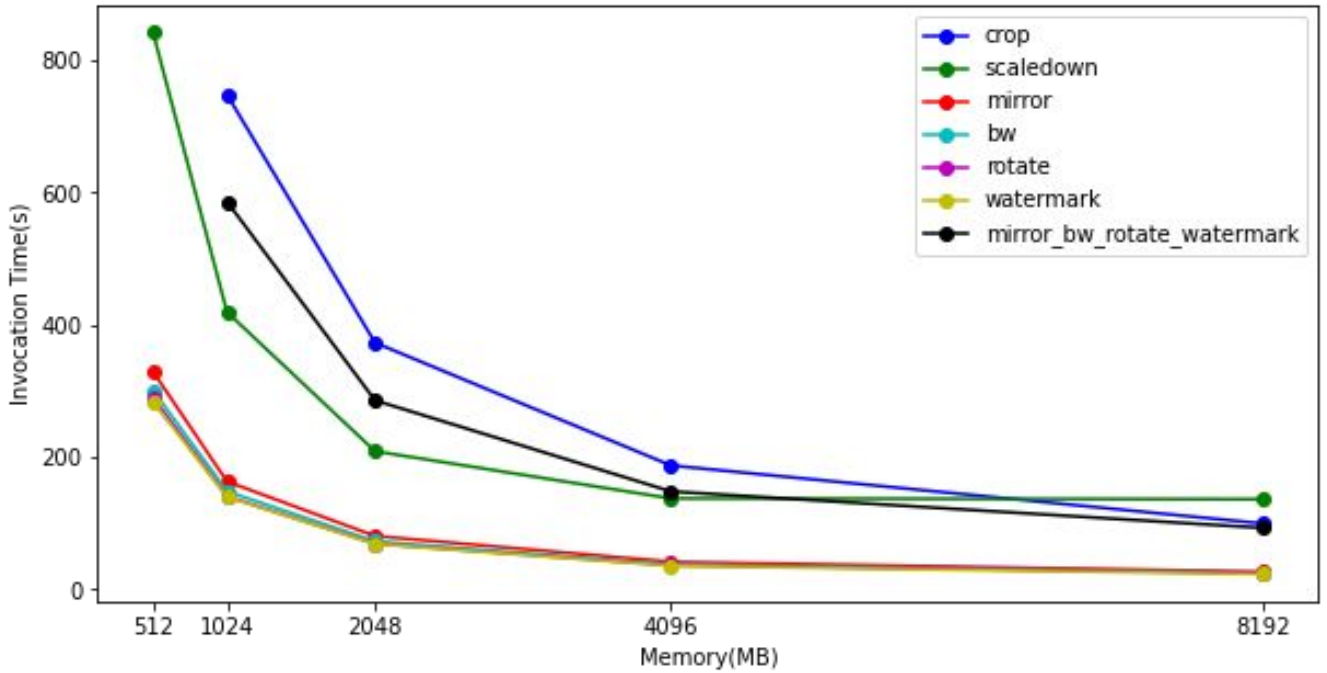
- AWS Cloudwatch Log Insights allowed us to analyze memory utilisation of our Lambda functions



Lambda Power Tuning State Machine

1 - https://docs.aws.amazon.com/lambda/latest/operatorguide/profile-functions.html

CPU Profiling

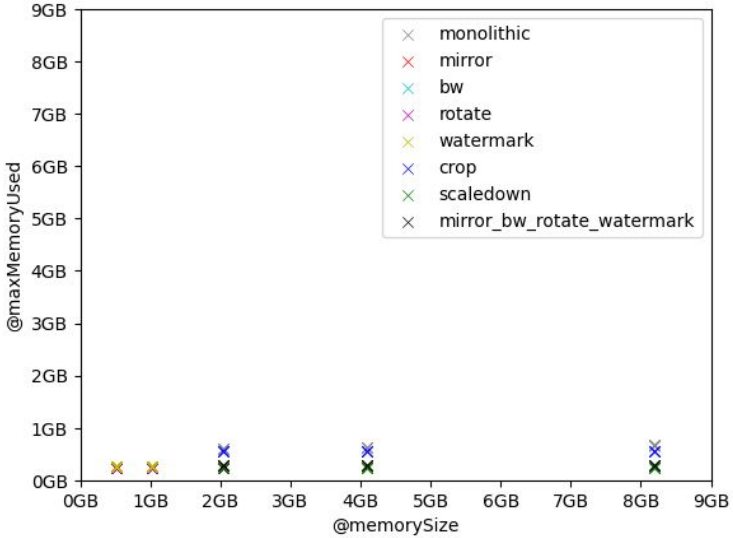


Memory Profiling

All Functions had low memory % utilization…

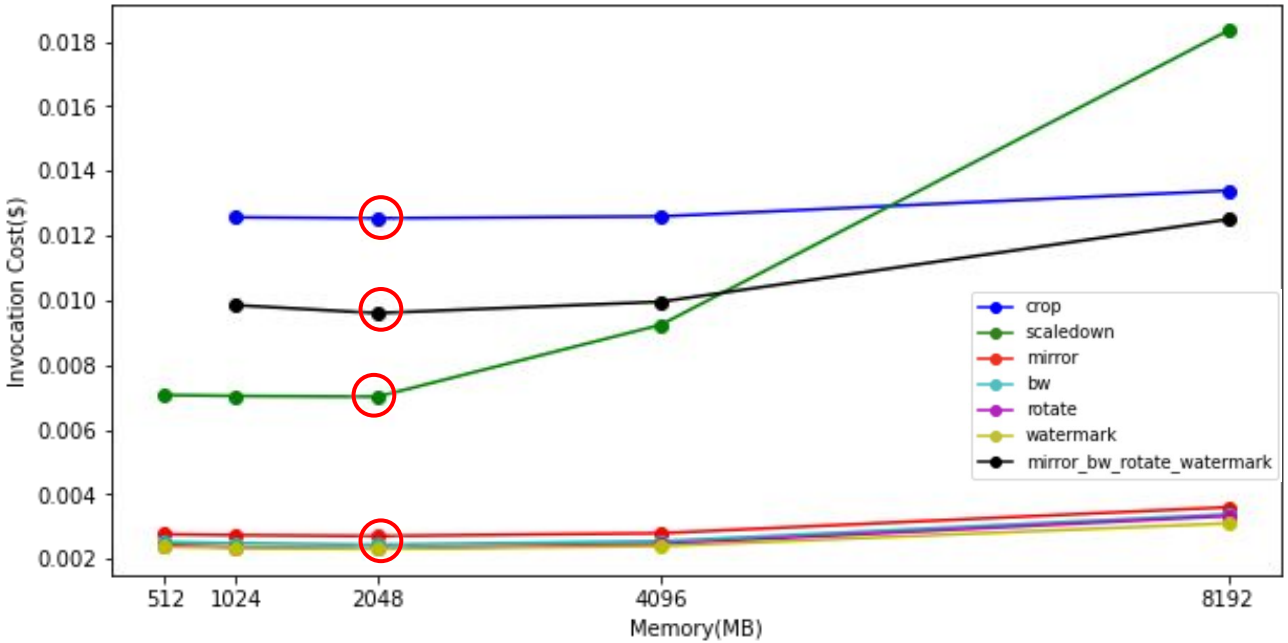
BUT Functions timeout for lower memory configuration…

The infamous fixed CPU-Memory ratio



Memory Consumption of Various Lambda Functions

Cost Profiling

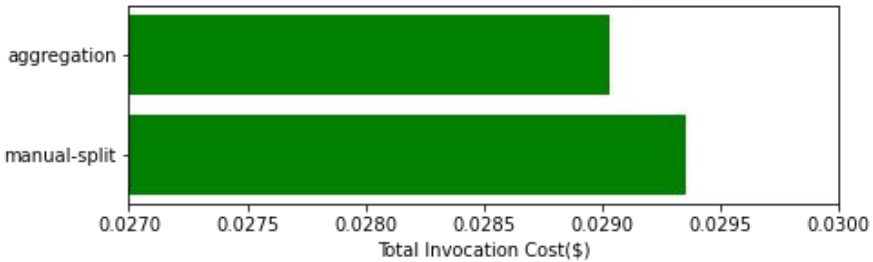


https://aws.amazon.com/lambda/pricing/

Results

- The cost-performance optimal memory allocation was found to be **2048MB**

- The estimated cost benefit of aggregation over manual-splitting for 1 billion invocations would be **$320k**



Conclusions

Monolithic

+ Good for Small Videos and Small Applications
+ No intermediate storage interface required
- Timeout for larger videos
- High cost for less intensive memory processes

Manual Splitting

+ Right-size each stage
+ Handle larger videos
- Intermediate Storage introduces latency and costs
- Underutilized Compute Resources

Strategy-based Splitting

+ Right-size each stage
+ Handle larger videos
+ Better Resource Utilization
- Finding splits is non-trivial

Q/A