Group 15: Gursharan Ahir,
Mohit Shah, Soham Pachpande

CSE 291, Fall '22
October 17, 2022

Project Proposal

1 Problem Statement

The traditional cloud computing model requires users to put in a lot of effort for setting-up the compute environment for their application. Users themselves need to manage issues like VM initialization, auto-scaling, fault-tolerance, etc. Serverless Computing resolves this issue by managing cloud resources. In this project, we aim to port applications to serverless, optimize the splitting, and draw performance based comparisons.

2 Importance

Serverless computing helps simplify deployment of applications enabling users to focus on developing their application rather than worrying about the infrastructure, and thus saving significant development time. At the same time, it helps reduce the associated cloud expenditure and thus, making it very important to port applications to serverless computing.

3 Challenges

Despite the benefits of serverless computing, there are several limitations like fixed resource allocation, CPU-memory ratios and execution time limit. Thus, it is crucial to split applications in an optimal method such that the serverless deployment leads to performance improvement. Not all applications are ideal for the serverless model, and the first challenge is identifying which applications are ideal for serverless framework. Secondly, we need to understand the associated code, and then find optimal split points.

4 Approach

4.1 Getting started with Serverless

In order to port an application to serverless, we need to first develop an understanding of serverless computing, and it's working along with the tools that enable running applications in the serverless model. We plan to use AWS Lambda which is a Function-as-a-Service tool, and build a simple webserver. This will familiarize us with the required tools, and the process of splitting of a basic application into multiple stages.

4.2 Application and Manual Splitting

The application we plan to work on is a Video Processing tool for ML applications involding the use of open-source packages like ffmpeg[1], moviepy[2] etc. We will first split the application into multiple sequential stages based on functionality. So this approach will have Lambda functions corresponding to downloading, scaling, cropping and other transformations respectively. This will be a rudimentary split based on the functionality of each stage, and will help us evaluate the benefits we can get from transitioning a video processing application to serverless.

4.3 Profiling and Finding Optimal Splits

It is important to find splits within our application which will lead to optimal resource utilization for each function. We undertake profiling experiments to better understand each application stage. We will profile the CPU and memory usage of our application in an end-to-end fashion on a single machine, and later as separate Lambda Functions.

4.3.1 End to End Profiling on a Single Machine

For end to end profiling, we will use native profilers such as *cProfile*[3] to find traces and CPU utilization on a VM or our local machines. This information will help us understand how long each stage within our code takes, and how we can aggregate shorter operations or dis-aggregate longer operations. Of more importance is the memory utilization, and related tools such as *memory_profiler*[4] which run the entire script offline, and gives a line-by-line memory utilization for a given python script. Similarly, *memray*[5] gives the function-level memory utilization. Finally, *austin*[6] reads interpreter virtual memory space to get information about both memory and CPU utilization. We aim to use the a combination of above tools to profile the end-to-end running of our application.

4.3.2 Monitoring Lambda Functions

Once we split and deploy our application as a Lambda functions, we will also monitor the resource utilization of each Lambda function. This will help us understand how our application behaves in the serverless framework. We will use Amazon's cloudwatch service[7] to track metrics. The majority of Lambda metrics focus on the duration of execution and concurrency. We will use the duration metrics to understand CPU utilization. Lambda Logs Insights extracts maxMemoryUsed, memorySize from logs[8] which we will use to understand memory utilization.

4.3.3 Resource Centric Splitting of Applications

The statistics related to CPU and memory usage would lend us insight about key breakpoints in the program which we could use to split our application. We anticipate to find Lambda functions which are underutilized and can be accordingly aggregated. Conversely, certain lambda functions can be limited by compute and will require further investigation with respect to disaggregation. These activities are equivalent to adding nodes and combining nodes in our DAG of functions.

4.4 Benchmarking

We plan to use memory utilisation, execution time, network utilization, and total cost as benchmarks to compare our various serverless deployments.

4.5 Stretch Goals

4.5.1 Using SCAD

We can use SCAD compiler[9] to find optimal splitting points for our application and benchmark the performance. One of our concerns is that the video pipeline might not necessarily be in native python which can make using SCAD non-trivial.

4.5.2 Parallelization

After optimizing memory and CPU utilization, we can improve the efficiency of our pipeline by performing the video pipeline in parallel. We aim to split the source video into batch frames, individually processing each batch using functions, and stitch together the results.

Appendix

Tools

We will utilize provided AWS credits to use AWS Lambda as our choice for serverless computing platform, AWS S3 for data storage, and Cloudwatch for monitoring and profiling. All other tools we plan to use are described in the approach, and are open-source. Accordingly, we do not anticipate any licensing or specialised hardware.

Timelines

Week 4 - Converting a simple Web Server to serverless, and explore the required AWS services
Week 5 - Define the splits, and convert the video processing application to serverless
Week 6 - Profiling the end-to-end application, and the manually split Lambda functions
Week 7 - Defining the resource-based splits, and again convert the video processing application to serverless
Week 8 - Profiling and Benchmarking various serverless deployments of our video processing pipeline
Week 9 - More experiments, analysis and documenting final findings

References

[1] "Ffmpeg." https://ffmpeg.org/.

[2] "User guide — moviepy 1.0.2 documentation." https://zulko.github.io/moviepy/.

[3] "The python profilers — python 3.10.8 documentation." https://docs.python.org/3/library/profile.html.

[4] "memory-profiler · pypi." https://pypi.org/project/memory-profiler/.

[5] "bloomberg/memray: Memray is a memory profiler for python." https://github.com/bloomberg/memray.

[6] "P403n1x87/austin: Python frame stack sampler for cpython." https://github.com/P403n1x87/austin.

[7] "Working with lambda function metrics - aws lambda." https://docs.aws.amazon.com/lambda/latest/dg/monitoring-metrics.html.

[8] "Supported logs and discovered fields - amazon cloudwatch logs." https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_AnalyzeLogData-discoverable-fields.html.

[9] Z. Guo, Z. Blanco, M. Shahrad, Z. Wei, B. Dong, J. Li, I. Pota, H. Xu, and Y. Zhang, "Resource-centric serverless computing," 2022.