Cloud Computing COMSM0010: Birdception

Bradley Miles (bm15731) Spencer Warren (sw15256)

Abstract— **This report details the design, implementation and testing of Birdception. Birdception is a platform for bird watching enthusiasts which classifies user uploaded images of birds and displays them in an online portfolio. Using current machine learning techniques and a cloud based architecture, we are able to accurately provide classification on an app which scales well with demand. Birdception can be accessed here: `http://spenley-wales.s3-website.eu-west-2.amazonaws.com/`**

I. INTRODUCTION

A. *Cloud computing*

Cloud computing can be described broadly as the delivery of software and computing resources over a network, most often the internet. Cloud computing represents a significant change in the way that computing resources are provisioned and paid for. The NIST definition of Cloud computing places emphasis on its "on-demand self-service", "rapid elasticity" and "resource-pooling" characteristics [1]. The use of resource-pooling has been attributed to a large reduction in costs for providers as resources can be dynamically allocated and shared between multiple customers in accordance with demand [2]. This dynamic allocation also underpins the rapid elasticity of resources as a user can be provisioned any of the available resources for only the amount of time that they need them.

The benefits of Cloud computing have not gone unnoticed. Statistica reported that the public cloud computing market has an estimated worth of $130bn worldwide [3]. A key differentiater is that customers can have access to large-scale computing power without needing large capital expenditure for the required hardware. Amazon's CTO, Dr Werner Vogels describes it as a solution to deal with the large economic uncertainty as you can "turn fixed costs into variable (costs)" [5]. Businesses can effortlessly scale with demand allowing for riskier business ideas.

B. *Birdception*

Inspired by the avid bird watching community, Birdception is an application which allows users to uploaded images of the birds they have spotted and receive classifications of their breeds. They can view this in their online portfolio, which contains a collection of the images they have uploaded and the associated breeds of the birds. Birdception is built using AWS and is serverless in nature. The architecture was designed such that its performance will remain largely consistent regardless of the load.

II. BIRDCEPTION WALKTHROUGH

After reaching the website through the specified link, the user has the option to learn more about Birdception as well as register for the service. After entering registration details a verification code is sent to the user's email address to confirm registration. Once registered, the user can access their portfolio page. The portfolio page allows users to upload images for processing and also view their previously processed images. To log out, the user can click the "sign out" button at any time whilst logged in. Birdception has persistent login to allow for ease of use.

III. ARCHITECTURAL DESIGN

The cloud provider chosen for this project is Amazon Web Services (AWS). Launched publicly in 2006, it is the longest running commercial cloud platform [4]. The decision to use AWS over competitivecool cloud platforms, such as Google Compute Engine or Oracle Compute Cloud, was driven by a variety of factors including functionality, availability and supporting documentation. AWS provides a plethora of scalable and interoperable services, akin to the requirements of Birdception. The global presence of AWS facilitates future growth of the application as it can be deployed in a vast number of regions and availability zones throughout the world with minimal difference in user experience regardless of location. Finally, AWS offers the largest support community and supporting documentation for its services.

A. *Compute*

Birdception is designed as a publicly accessible web-application. Consequently, it was paramount that a solution was found for hosting the client-side content as well as deploying the back-end application programming interface (API) of the website. AWS provides a variety of different methods for resolving this problem.

The first we explored was creating a stateful web server hosted on an AWS EC2 instance. EC2 is Amazon's elastic compute cloud, providing bare metal instances capable of hosting any web server desired by the developer. In the example of Birdception, the web server would be responsible for processing all incoming requests, such as, serving the HTML to client, executing the image classifier and communicating with the chosen storage and database services of the application. With increasing traffic, this would be a lot for a single server to process, as a result we proposed creating an auto-scaling group that manages the instantiation and termination of multiple EC2 instances which sit behind an AWS Elastic Load Balancer. AWS Elastic Beanstalk would then manage the deployment of the code base onto these instances. The main limitation of this solution however is that we desired separation between the web server and the slow image classifier. We had considered using Amazon's Simple Queue Service (SQS), to manage the pending images to be classified and using a separate EC2 instance to perform the classification step only. Conversely, as the classifier only take a couple of seconds to execute, it seemed excessive to add such an architectural overhead to the solution. Furthermore, the asynchronous nature of SQS results in the user not knowing when their bird photograph has been classified without the use of Amazon's Simple Notification Service.

The second solution available is to use stateless computing; AWS Lambda is Amazon's stateless compute service capable of executing back-end code without provisioning or managing servers. In this implementation, AWS Lambda and Amazon's API Gateway would be used in unison to provide a public API for Birdception. API Gateway provides an interface for creating, publishing and maintaining secure APIs capable of accepting and processing up to hundreds of thousands of concurrent API calls. Each endpoint can be configured to trigger and execute a specific Lambda function, thus providing back-end code for the application. The major advantage of stateless computing compared to conventional hosted servers is that you only pay for the requests made, rather than per instance. This is considerably cheaper at all levels of traffic, with the sacrifice of a small increase in latency - the time required to instantiate the lambda function. Finally, to complete this solution, a method of hosting the website is still required, solvable by Amazon's Simple Storage Service (S3). A bucket in S3 can be configured to provide static web hosting, serving the client-side content to the user and beneficially reduce much of the undifferentiated heavy lifting coupled with web hosting. This alongside the ability to decouple the classification process was the resounding reason why we chose the latter of these two solutions to be the final compute implementation for Birdception.

B. Storage

In addition to classification, Birdception's supplementary functionality is to manage a portfolio of photographs for each of its users. Storing these images on the compute server is unsuitable given the quantity of images and the necessity for their persistence. Consequently, the need for the application to have an accessible and scalable object storage service was essential. Amazon offers a variety of different storage services including, S3, Glacier and EFS. Despite this, S3 is the natural choice as it is designed as the object store of Amazon's web services; offering secure and scalable storage with 11 9's of durability. Glacier is a low-cost cloud storage service designed for archiving and long-term backups, consequently not a viable solution as the time for data retrieval would be too slow for a responsive web application. Finally, Amazon's Elastic File System (EFS) is designed to provide massively parallel shared access to thousands of Amazon EC2 instances for big data analytics or media and entertainment workflows. This is unnecessary for Birdception and thus we chose S3 as our final storage service.

C. Database

Once an image has been classified, Birdception is designed to store the metadata of that classification in a persistent database. Amazon provides two real-time database services called RDS and DynamoDb - SQL and NoSQL databases respectively. We decided to use DynamoDb as the output of the image classifier was already formatted in JSON, although either service could have been used to store the metadata appropriately. Both services can handle thousands of concurrent requests and by configuring indexes for the user's id the data can be accessed with millisecond performance.

D. Security, Identity, & Compliance

Birdception is designed as a secure application, with each portfolio only accessible by its owner. As a result, the application requires a user identity manager. One viable solution is storing the user's credentials in a DynamoDB table, however this requires custom management and knowledge of the latest security procedures and protocols. Amazon provides a managed service called Cognito designed to handle website authentication, a common instance of undifferentiated heavy lifting. Cognito provides user pools which allow sign up and sign in resources. Futhermore, to ensure the validity of the users, email and mobile verification methods can be enabled.

IV. IMPLEMENTATION DETAILS

Each service component in Birdception is decoupled which allows for functional decomposition. Functional decomposition enables greater opportunity to allocate resource consumption with demand as only the services which are in demand will be active. An overview of the system's architecture can be seen in Figure 1. The user interacts with the website hosted on S3 which, in turn, interacts with Cognito for user identification and verification. For all user related website functionality API calls are made through the API Gateway. The API Gatway invokes calls to Lambda functions which further interact with S3 and DynamoDB (for persistent storage).

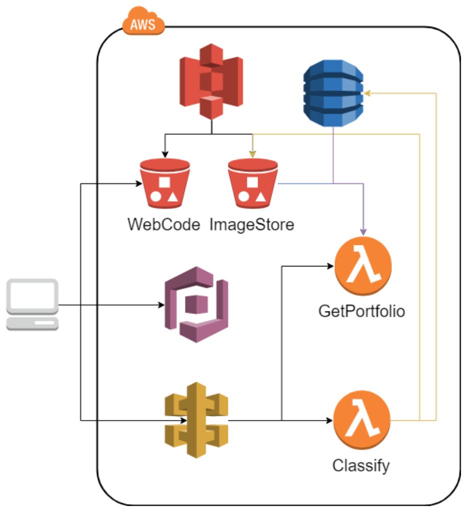

Fig. 1. Birdception's system architecture

A. Front end

The website is produced with HTML and CSS for static content and uses Javascript for serving dynamic content. It is hosted using S3 in the EU (London) region as that is where we believe the majority of our traffic will come from. The basis for the website's structure and authentication method was derived from the AWS tutorial "Build a Serverless Web Application" [7]. The Birdception website has multiple routes: `/index.html`, `/register.html`, `verify.html`, `/signin.html` and `/portfolio.html`.

1) : `/index.html` is the homepage, providing information regarding Birdception as well as links to register and sign in.

2) : `/resgister.html` provides a user form from which to register for a Birdception account. When details have been submitted, the `signUp()` function from the Cognito user pool is invoked. An authentication email is sent to the provided address by Amazon's Simple Notification Service (SNS) and the user needs to follow the link in order to finalise the set up of their account.

3) : `/verify.html` allows the user to enter the verification code obtained from the registration email along with their other account details. This sends a request to Cognito which updates the account status to `CONFIRMED`.

4) : `/signin.html` lets the user enter the protected area once they have successfully registered. AWS Cognito verifies their account and returns an authentication token to be used in authorising requests. Signing in successfully forwards the user onto the protected area `/portfolio.html`.

5) : `/portfolio.html` is the page from which users can upload images and view their portfolio. When the page loads, an authenticated GET request is made to the API gateway using the `/index` path. A list of tuples containing the image path, image url and image classification is return and are presented on the page. To upload and classify an image an authenticated POST request is made to the API gateway using the `/classify` path. The JPEG image is encoded in base64 and is sent in the request body. To confirm the success or failure of the upload, a response with the appropriate status code is returned.

B. API Gateway

To process the requests made by the website, Birdception uses a REST API, served by API Gateway. The integration with Cognito takes care of authorisation for protected requests. Birdception's API has two paths as specified in section IV-A.5. When accessed, each path triggers an AWS Lambda function. `/index` triggers the *GetPortfolio* Lambda and `/classify` triggers the *Classify* Lambda.

C. Lambda

AWS Lambda is a serverless compute platform which automatically provisions computing resources and uses the containerisation paradigm. Lambda functions use no resources until they are triggered and the owner is only billed for the amount of time the function spent executing. Birdception contains two Lambda functions which perform the heavy lifting for the application. Each Lambda function is triggered by API calls as specified in section IV-B. When a Lambda is triggered, the API passes the function the trigger event and a callback function. The trigger event contains the image data and Cognito username and the callback is used to return data to the API. When a Lambda receives a trigger, a container is instantiated. The container consists of a minimal linux kernel required to execute the code and the application code itself. For each concurrent invocation, a new container is made. By default, Lambdas can handle 1000 concurrent invocations. AWS Lambdas take longer to produce a response if they have not been invoked recently. This is known as a Cold start. The converse is called a Hot start which decreases response latency.

The *Classify* Lambda code is written in Python 2.7 and uses the TensorFlow library for image classification. The classification model used is similar to the Inception architecture and is trained on 871 classes of birds. After running the inference function, the prediction labels and probabilities are produced. The Classify function returns the label of the highest probability. The image provided is then stored in an S3 bucket and the classification, image path and relevant user information is stored in an Amazon DynamoDB table.

The *GetPortfolio* Lambda code is written in Javascript for execution in Node 8.10. It obtains the *userId* from Cognito then scans the DynamoDB database for all the user's portfolio image details. Once these image details have been obtained, signed urls are generated and returned to the user through the API.

To provide more security to the Lambdas Identity Access Management (IAM) roles are applied. The roles ensure that the Lambda can only access resources it has specific permission to access. Hence, The *Classify* Lambda only has putObject access for the image storage bucket and only has write access to the DynamoDB table. This mitigates code changes with unintended side effects.

D. S3

Amazon's Simple Storage Solution (S3) is an object storage service which is designed for 11 9's of durability and "stores data for millions of applications for companies all around the world" [6]. S3 is used for multiple purposes in Birdception. The first is hosting the static webpages. A bucket named `spenley-wales` (hence the URL) contains the `html` files and relevant Javascript files. When a user accesses Birdception's webpage, requests are made to the bucket to receive the required files. The second bucket `deeplearning-code-bucket` contains the function code and relevant libraries for the image classification Lambda. As the function code exceeds 100mb it must be stored in an S3 bucket. The final bucket contains directories which store the user's images. Paths are specified as `users_Cognito_username/image.jpg`.

E. DynamoDB

DynamoDB is a NoSQL, serverless, database service. It provides some of the persistent storage for Birdception and stores the classifications along with the paths to the corresponding images. This data is accessed by the *GetPortfolio* Lambda - described in section IV-C.

F. Cognito

AWS Cognito provides user management and authentication which is typical undifferentiated heavy lifting. Cognito is accessed by the website and is integrated with the API in order to authorise requests. The Cognito *userId* also provides a unique method for identifying users.

V. Scalability and Performance

To test the scalability of Birdception, we implemented a test plan configuration for Apache JMeter - an open source transaction per second (TPS) generator. JMeter provides a mechanism for load testing different endpoints by defining thread pool executors against endpoints of your target application. We used the *Throughput Shaping Timer* JMeter plugin to configure a ramp up test, increasing the number of transactions per seconds over time. The ramp up graph of the test can be seen in Figure 2 below, reaching a maximum of 1000 TPS over a four minute period.

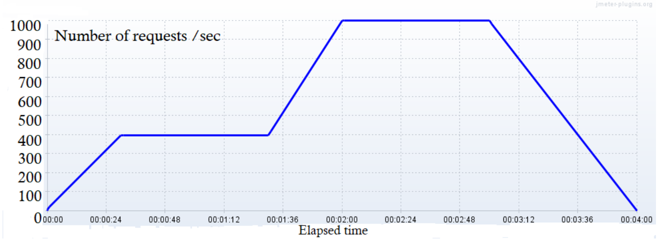

Fig. 2. JMeter target TPS graph.

We hit the `index.html` endpoints using JMeter in order to test the speed of accessing the website as the number of concurrent users increases. We ran the test in the command line as directed by the JMeter documentation and recorded both the throughput and the average response latency per five minutes period.

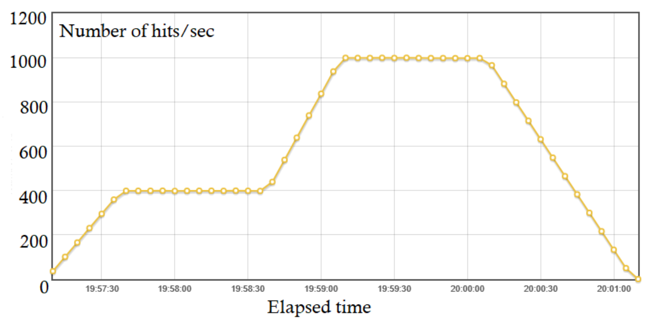

Fig. 3. Throughput of load test.

As expected, the actual throughput was close but not identical to the target TPS, shown in Figure 3. This is due to the computational limitations of TPS generators and the implementation difficulties of timing so many requests accurately across multiple threads.

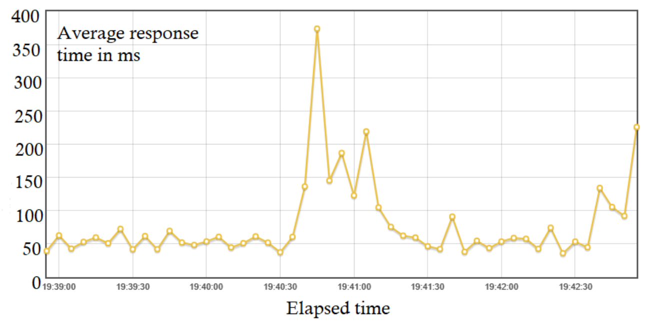

Fig. 4. Latency for webserver requests over time

Our results are incredibly promising as at the maximum 1000tps throughput, Birdception did not throttle and was able to successfully handle all requests with an average latency of 50ms. Two minutes into the test, at 400tps, a latency spike of 350ms is observable, this is not a concern however, and can be attributed to standard varying latency over the web.

VI. Conclusion

Birdception is a great demonstration of the advantages of serverless cloud computing. The site suitably provides a platform for any bird watching enthusiast to upload a portfolio of photographs and to get them automatically classified by a neural network based on the Inception architecture. With scalability and cost driving the design of this functionally decoupled site, Birdception ensures that the owner only pays for the resources utilised within AWS. Each service has been configured with strict access policies and logging configuration to maximise the number of metrics for debugging and performance evaluation. Finally, by utilising a third party TPS Generator we have evaluated and proven from the client-side that the site can scale with increasing load and concurrent connections.

VII. Future Works

Loading the function code and libraries into the *Classify* Lambda takes a significant amount of time during invocation. To lessen this, images could be queued and then classified in groups by a Lambda invocation - spreading the start-up overhead between more classifications. AWS SQS would be the best infrastructure for providing the queue. Once the number of messages on the queue reached a certain threshold, a Lambda function would be invoked to consume multiple messages and classify the images. Adding this infrastructure however imposes asynchronous processing on the application and would require additional development overheads and configuration. Furthermore, it would be essential to validate that the increased communication overheads of publishing and consuming from the queue does not exceed the time required to invoke multiple containers.

References

[1] NIST. July 2009. “The NIST Definition of Cloud Computing”. Accessed December 2018 from https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf

[2] Kevin L. Jackson, Forbes. September 2011 “The Economic Benefit of Cloud Computing”. Accessed December 2018 from https://www.forbes.com/sites/kevinjackson/2011/09/17/the-economic-benefit-of-cloud-computing/#40ecf8fb225c

[3] Statistica. 2019. “Total size of the public cloud computing market from 2008 to 2020 (in billion U.S. dollars)”. Accessed December 2018 from https://www.statista.com/statistics/510350/worldwide-public-cloud-computing/

[4] AWS. “About AWS”. Accessed December 2018 from https://aws.amazon.com/about-aws/

[5] Divina Paredes, CIO. 2013. “Amazon CTO: Stop spending money on ’undifferentiated heavy lifting’”. Accessed December 2018 from https://www.cio.co.nz/article/466635/amazon_cto_stop_spending_money_undifferentiated_heavy_lifting_/

[6] AWS. “Amazon S3”. Accessed December 2018 from https://aws.amazon.com/s3/

[7] AWS. “Build a Serverless Web Application”. Accessed December 2018 from https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/