Leveraging Cloud for IoT solutions: An Analysis of AWS cloud Services in Smart Pet Guardian Project

Ashiqur Rahman Habeeb Rahuman
Future Networked Systems
School of Computer Science – Trinity College Dublin
Dublin
habeebra@tcd.ie

***Abstract*— The Smart Pet Guardian project addresses the necessity of constant pet tracking via a cloud-hosted Internet of Things (IoT) approach. This paper explores the use of AWS Lambda and Dynamo DB to provide efficient data processing from sensors, effectively addressing the difficulties associated with managing real-time data. Our cloud architecture facilitates complex procedures, ensuring the safety and effectiveness of pet tracking. This approach showcases the crucial role of cloud computing in enhancing IoT applications for pet care.**

Keywords*— IoT (Internet of Things), Cloud Computing in IoT, AWS Lambda, Data Collection and Management, Amazon DynamoDB, Smart Pet Monitoring System, Real-time Data Processing, AWS Cloud Services, IoT Device Integration, Pet Tracking Technology component,

I. Introduction

The emergence of the Internet of Things (IoT) has the potential to completely change the domestic environment, especially within the area of pet care. The Smart Pet Guardian (SPG) system has been established in response to the developing demand for automated solutions that ensure the protection and monitoring of pets. To improve pet care and supervision, this system combines innovative cloud computing, complex sensor technologies, and cutting-edge communication protocols to deliver a full IoT ecosystem.

II. Current Architecture

The Smart Pet Guardian (SPG) system showcases an advanced IoT architecture designed to facilitate pet monitoring and interaction through a suite of interconnected devices and cloud services. This section outlines the project's architecture, detailing each component's role within the system.

A. Embedded Systems

Two embedded systems, each driven by the ESP32 microcontroller, renowned for its adaptability and Wi-Fi capabilities, lie at the heart of the SPG design. The systems are as follows:

1. Smart Pet Collar

The Smart Pet Collar functions as the central component of the SPG system, offering a wide range of features to track and assess the pet's movements thoroughly. The collar consists of the following parts:

- GPS Tracking: A Global Positioning System GPS unit has been easily incorporated into the collar, allowing for real-time tracking of the pet. This feature offers pet owners precise and real-time updates on their pet's location.
- Gyroscope: The addition of a gyroscope strengthens the collar's ability to track and understand the pet's positioning and alternated movements. This characteristic helps provide a comprehensive insight into the actions of the pet, from playful antics to subtle behaviors.
- Accelerometer: An advanced accelerometer embedded in the collar measures the pet's acceleration and deceleration, providing valuable information about its activity levels. This information adds to a thorough evaluation of the pet's overall health, helping parents of pets remain aware of their well-being and energy levels.

2. Smart Pet Door

The Smart Pet Door is a crucial element of the Smart Pet Guardian system, created to provide more than just a way for pets to enter but also to function as a data collection point that contributes to the broader cloud-based monitoring system. It serves as a connection between the pet's physical actions and the digital data from the system.

- *Access Control:* The door is equipped with RFID technology to allow only authorized pets to enter or exit the premises. The system detects the RFID tags on the pet's collar, leading the door to unlock and ensuring a secure environment.

- *Data Collection:* Each encounter at the entrance is recorded as a data event. Current Radio-Frequency Identification (RFID) technology emphasizes the SPG system's dedication to ensuring secure access management. Each pet is given a distinct RFID tag for a personalized and safe access system.

- *Cloud architecture:* For efficient real-time monitoring, control, and advanced data analysis, the integration, supported by Wi-Fi connectivity, guarantees smooth data transmission to the cloud, at which pet access patterns are carefully recorded, analysed, and saved. This feature allows the system to provide immediate notifications and remote access control, all while ensuring the necessary scalability and dependability for future growth.

- *Visual Monitor:* The Smart Pet Door contains an advanced camera module that is capable of providing live streaming and recording videos when specific events occur. The collection of visual data is of the highest priority, as it not only serves safety standards but also offers a reliable way for establishing the identification and actions of the pet while entering the door.

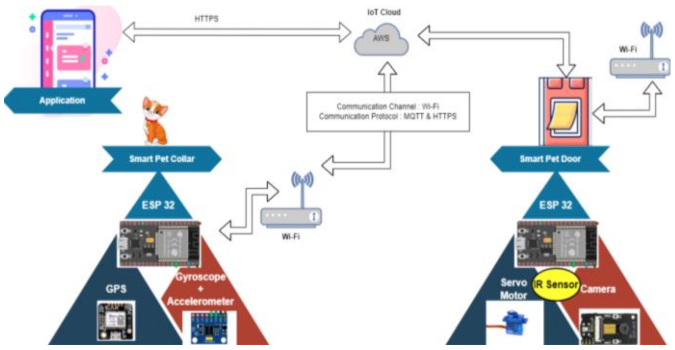

Figure 1: System Architecture of the SPG module

III. Discussion

This section will be a panel of discussion where the selection of specific Amazon Web Services (AWS) in the development of the project's cloud infrastructure, alongside an evaluation of the data collection strategies employed.

A. Cloud Services Selection and Configuration

- *Amazon EC2:*
  The SPG user interface was hosted on Amazon EC2 because of its robust computational capability, which can be easily changed to handle the unpredictable workload usually encountered in IoT applications. The project was able to achieve optimal performance by effectively balancing compute, memory, and networking resources, according to its broad spectrum of instance types. The eligibility for the free tier greatly reduced the initial expenditures, offering a cost-effective alternative for project development. In addition, the EC2 instances' capacity to expand effortlessly was crucial to handling potential rises in user access, ensuring an uninterrupted and responsive user experience.

- *Amazon API Gateway :*
  The SPG project's APIs have been handled using Amazon API Gateway, which was chosen for its completely controlled nature, making API implementation and distribution simpler. The platform included crucial functionalities such as rate limiting, real-time tracking, and the management of API versions, which are all vital in a constantly changing IoT ecosystem.

- *AWS Lambda:*
  The SPG project's real-time data processing demands were well met by AWS Lambda's without servers execute approach, which provided a simplified and event-driven computing solution. Throughout the process of abstracting server administration, Lambda has allowed the team to focus their efforts on developing application logic that is specifically customized to the project's requirements. This method not only decreased the amount of work needed to operate but also matched the project's potential to grow, making sure that the core processing could dynamically adjust to both the frequency and quantity of information from sensor inputs without any operator involvement.

- *Amazon DynamoDB:*
  Amazon DynamoDB was carefully selected because of its independently managed NoSQL database capabilities, which provided superior reliability in terms of minimal latency data retrieval and high quantities. It was crucial for storing and retrieving significant amounts of data generated by IoT devices, such as period sensor readings from pet collars and doors. The scalability characteristics of DynamoDB, like as automated splitting and indexing, enabled effective data organization and retrieval, hence facilitating the execution of complicated searches and analytics that were necessary for the project.

According to Table 2, a general overview of the process of function of the AWS lambda function as follows:

- *Data Retrieval:* The Lambda function fetches the pet's present position out of the 'Collar' table and the location of the geofence variables from the 'Geofence' table.
- *Distance Calculation:* Utilising the Haversine formula, it computes the distance that lies that separates the pet's location and the geofence center.
- *State Determination:* The function then checks whether the pet is within or outside the geofence by considering the computed distance and geofence radius.
- *Alert System:* In the event of a modification in the pet's geofence status (entering or exiting), the system promptly sends an alert to the pet owner.
- *State Update:* At long last, the function modifies the geofence status in the 'Geofence' table.

Table Name	Primary Key	Attributes	Purpose
User	UserID	UserID, Username, Email, Password	Manages user authentication and profiles.
Pet	PetID	PetID, Name, Species, Breed, OwnerID, PetRFID, CollarID, DoorID	Tracks individual pets and their details.
Collar	CollarID	CollarID, PetID, Latitude, Longitude, LastUpdateTime	Stores real-time location data from pet collars.
Door	DoorID	DoorID, CameraID, AccessControlEnabled, LastAccessTime, AccessLog	Controls and monitors smart door access.
Geofence	UserID	UserID, Latitude, Longitude, Radius, Timestamp, State	Allows users to set and manage geofences for pets.

Table 1: *DynamoDB Table Structure and Purpose in the Smart Pet Guardian Project*

Amazon DynamoDB is an essential component of our SPG project as it provides crucial support for the Geofencing logic. This logic is essential for tracking the location of pets in safe zones established by their owners.

From Table 1, the DynamoDB 'Geofence' table stores important data points including user-defined geofence geographic coordinates, radius, and an active/inactive state. By utilizing this data, in conjunction with real-time location information from the 'Collar' table, the system can accurately determine if a pet is within the designated geofence.

- *Amazon Identity and Access Management(IAM):*
  Using the allocation of distinct IAM roles to services such as AWS Lambda, we effectively managed access to essential assets like DynamoDB, guaranteeing secure and authorized operations. These IAM policies prioritize security and improve tracking by adhering to the notion of least privilege. The adaptability of IAM roles played a crucial role in expanding the project, enabling the modification of permissions to match the changing requirements of the project. This not only enhanced security but also improved operational efficiency.

- *Mosquitto MQTT Integration:*
  The Mosquitto MQTT broker is a crucial element in our cloud communication architecture. This messaging protocol is well-suited for IoT ecosystems, providing efficient solutions for low-bandwidth and high-latency scenarios. Mosquitto facilitates the communication of messages from IoT devices such as collars and doors, ensuring prompt and efficient delivery of real-time data to our AWS environment for processing. The implementation was vital for ensuring the accuracy and effectiveness of our pet tracking and geofencing features. It highlights the importance of having strong and consistent protocols for communicating in IoT applications.

B. Data Collection

- *Event-oriented Processing of Data:*
  A. Efficiency and Scalability factor: Through employing Lambda functions, data can potentially be promptly handled upon reception, minimizing the delay often associated with conventional server-based systems. The server-free design of Lambda guarantees automatic scalability in response to an increasing amount of data, which is essential for regulating the unpredictable characteristics of IoT conditions.

- *AWS Lambda for Real-Time Processing:*

  A. *Data Storage:* By employing Lambda functions, data could be swiftly processed upon their arrival eliminating the delays commonly found in conventional server-based systems. The automatic scaling capability of Lambda was essential for effectively managing the unpredictable dynamics of IoT environments, as it allowed the system to seamlessly handle varying volumes of incoming data.

IV.COMPARISSION

For the development of a cloud platform for the Smart Pet Guardian project, our team conducted a thorough evaluation of various cloud service providers and their offerings. This comparison was crucial in determining the most suitable platform, considering factors such as compatibility with IoT, ease of integration, scalability, cost-effectiveness, and the team's familiarity.

A. Google Cloud Platform: Google Cloud Functions and Firestore

1. Google Cloud Functions:

Pros: Google Cloud Functions is highly regarded in the field of server-less computing for its robust support for various programming languages and effortless integration with a wide array of Google Cloud services. IoT applications that demand scalable and responsive solutions will find this platform especially useful as it excels at managing event-driven architecture. Because of its interaction with Big Query for data analytics and Dataflow for batch and stream data processing, it provides a comprehensive platform for intricate IoT networks.

Cons: The main reason for choosing Google Cloud Functions is the team's deep experience and well-established AWS procedures. Making the transition to Google Cloud would entail a significant dedication to understanding and adjusting to a different platform. Recognising Google's programming architecture and operational subtleties is just one aspect of this adjustment; another is connecting it with pre-existing AWS-based components. This transition has the potential to cause delays in the project and longer development timelines, which raises concerns about effectively meeting project milestones.

2. Google Firestore:

Pros: Firestore, a part of the Google Cloud Platform, is known for having the capacity to seamlessly synchronize knowledge through client programs at the moment. This is a powerful NoSQL database service that offers real-time capabilities. The server-less architecture is highly compatible with dynamic mobile and web applications, allowing for quick data updates that are crucial in IoT scenarios. Firestore's scalability is a significant advantage, effortlessly handling changing data demands without adding excessive operational burden.

Cons: Despite the numerous advantages, Firestore was subsequently not selected for the SPG project based on several crucial factors.

1. *Integration with AWS Services:*
   The seamless integration of DynamoDB with various AWS services, particularly AWS Lambda, played a major part. The smooth integration of systems makes it easier to handle IoT data, particularly whilst dealing with the continuous flow of information from sensors in real-time.

2. *Cost-Effectiveness:*
   The availability of DynamoDB within the AWS free tier offered a more cost-effective choice for managing our project's large amount of data. Choosing Firestore would have resulted in giving up these cost advantages, resulting in higher operational costs.

3. *Familiarity and Efficiency:*
   Our team's prior experience with DynamoDB played a pivotal role. We have extensive knowledge of DynamoDB's data modeling and operational management, which allows for fast and efficient development. Choosing Firestore would have necessitated a significant commitment to training and adjusting to new methodologies, which could have potentially hindered the advancement of our project.

B. Microsoft Azure: Azure Functions and Azure Cosmos DB

1. *Azure Functions:*

Pros: Azure Functions provides a server-less computing platform that can scale dynamically based on the application's needs. It also has seamless integration with the Azure ecosystem.

Cons: When considering the transition to Azure Functions, we came across substantial challenges. The transition from our project's existing AWS-based infrastructure to Azure would necessitate a thorough revamp of our system and significant study. In addition, the potential drawbacks of dealing with difficulties with integration and service interruptions made Azure Functions a less viable option.

2. *Azure Cosmos DB:*

Pros: The Azure Cosmos DB can handle different data models and provides wide global distribution, making it a great choice for applications with diverse database needs and a requirement for widespread data availability.

Cons: The decision to choose DynamoDB was mainly influenced by its seamless incorporation within the AWS framework, which plays a crucial role in our project's infrastructure. The deep knowledge and proficiency of our team in DynamoDB's data modeling techniques and functional nuances have greatly enhanced our development process. In addition, the practical difficulties of transitioning to an alternative database platform, such as the challenges of transferring data and the requirement for training on an entirely different platform, were seen as preventing our project's progress and effectiveness.

Component	AWS Service Used	Purpose in Project	Alternate Idea	Reason for AWS Choice
User Interface	Amazon EC2	Hosting the UI with scalable computing capacity	Google Compute Engine	Scalability, Free Tier, AWS familiarity
API Management	Amazon API Gateway	Managing secure API calls for IoT device communicat ion	Google Cloud Endpoints	Robust security, seamless AWS integration
Data Processing	AWS Lambda	Server-less backend code execution for sensor data	Azure Functions	Pay-as-you-use model, seamless AWS integration
Data Storage	Amazon DynamoD B	Storing and retrieving IoT-generated data	Google Fire store / Azure Cosmos DB	Scalability, low-latency performanc e

***Table 2:** Tabular representation over Comparison of Cloud Components and Alternatives*

V. SUMMARY

This paper provides an extensive review of the AWS cloud services, focusing on AWS Lambda and Amazon DynamoDB, within the framework of the Smart Pet Guardian project. We presented the effectiveness of serverless architectures enabled by AWS Lambda in managing real-time data processing needs. The NoSQL database service provided by Amazon DynamoDB played a crucial role in efficiently handling the large amount of data generated by IoT devices. It offered fast performance, which was essential for the real-time tracking abilities of our system. Through our investigation, we discovered that the smooth integration and ability to scale offered by AWS services play a crucial role in advancing IoT solutions. The Smart Pet Guardian project demonstrates the power of cloud computing in improving IoT ecosystems, offering scalable, efficient, and cost-effective solutions that are crucial for the advancement of smart, connected systems.

References

[1] S. Ray, "ESP32 Based Web Server for Accessing Sensor Data over the Internet," in International Journal of Computer Applications, vol. 975, no. 8887, 2019.

[2] M. Villamizar, O. Garces, L. Ochoa, H. Castro, H. Verastegui, and A. Salamanca, “Evaluating the Performance and Scalability of Serverless Computing Functions for IoT,” in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5309-5317, 2019.

[3] M. S. S. Khan, K. Salah, and K. Damaj, "IoT security: Review, blockchain solutions, and open challenges," in Future Generation Computer Systems, vol. 82, pp. 395-411, 2018.

[4] C. -M. Own, C. -Y. Teng, J. -R. Zhang, W. -Y. Yuan and S. -C. Tsai, "Intelligent pet monitor system with the internet of things," 2011 International Conference on Machine Learning and Cybernetics, Guilin, China, 2011, pp. 471-476, doi: 10.1109/ICMLC.2011.6016785

[5] R. V. Bapat, "Amazon DynamoDB: A seamlessly scalable non-relational database service," in AWS Whitepapers, 2013.

[6] L. Botta, W. de Donato, V. Persico, and A. Pescapè, “Integration of Cloud computing and Internet of Things: A survey,” in Future Generation Computer Systems, vol. 56, pp. 684-700, 2016.M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.