Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Scalable Cloud Architecture for Real-Time Genomic Pipeline Processing

Ishana Bagaitkar[1], Vedika Kadam[2], Bhagyashree Kharat[3], Prof. Iffat Kazi[4]

[1,2,3,4]Department of Computer Engineering Usha Mittal Institute of Technology
SNDT Women's University, Mumbai, India

Abstract
With the rapid growth of next-generation sequenc- ing (NGS) technologies, the volume of raw genomic data has in- creased significantly, creating a need for scalable and automated computational systems for efficient analysis. Traditional high- performance computing (HPC) systems often face limitations in scalability, flexibility, and real-time processing.

This project presents a cloud-based, event-driven architecture for automated genomic data processing using Amazon Web Services (AWS). The system enables users to securely upload FASTQ files through a web-based interface, which are then stored in Amazon S3. An event-triggered AWS Lambda function initiates a containerized bioinformatics pipeline executed on AWS ECS Fargate.

The pipeline integrates widely used tools such as FastQC and MultiQC for quality control, along with Ensembl Variant Effect Predictor (VEP) for genomic variant annotation. Processed results are stored securely in cloud storage and made accessible to users via pre-signed URLs. The architecture emphasizes scalability, cost efficiency, fault tolerance, and reproducibility.

By leveraging serverless computing, containerization, and event-driven workflows, the proposed system provides an effi- cient and production-ready solution for real-time genomic data processing in a cloud-native environment.

Index Terms—Genomic Data Processing, Cloud Computing, Event-Driven Architecture, FASTQ, Bioinformatics Pipeline, AWS Batch, Scalability

PROBLEM STATEMENT

The rapid advancement of next-generation sequencing (NGS) technologies has resulted in a significant increase in the volume of raw genomic data, commonly generated in FASTQ format. Processing and analyzing such large-scale data requires substantial computational resources, efficient storage mechanisms, and automated workflows. Traditional high-performance computing (HPC) systems often face chal- lenges related to scalability, resource utilization, and real-time processing.

Bioinformatics tools such as FastQC and MultiQC are widely used for quality control, while variant analysis tools like Ensembl Variant Effect Predictor (VEP) require well- orchestrated pipelines and

consistent execution environments.

Managing these tools manually introduces complexity, de- pendency issues, and inefficiencies, especially when handling multiple datasets.

The objective of this project is to design and implement a cloud-based, scalable, and event-driven system capable of securely handling large genomic datasets. The system auto- mates bioinformatics workflows by leveraging cloud services for storage, serverless orchestration, and containerized execu- tion. It dynamically allocates computing resources based on workload and provides users with seamless access to analysis results through a web-based interface.

The proposed solution aims to be cost-effective, reliable, se- cure, and reproducible, enabling efficient real-time processing of genomic data in a cloud-native environment.

1. Introduction

Modern biological and medical research has been sig- nificantly transformed by the rapid advancement of next- generation sequencing (NGS) technologies. These technolo- gies enable the generation of massive amounts of genomic data quickly and efficiently. Sequencing platforms today can produce datasets ranging from gigabytes to terabytes, typically stored in formats such as FASTQ, which contain millions to billions of short DNA sequence reads. Although the cost of sequencing has decreased, the complexity and volume of data processing, validation, and analysis have increased substan- tially, creating a need for efficient and scalable computational systems.

Before meaningful insights can be derived from raw se- quencing data, several preprocessing and quality control steps are required. Tools such as FastQC are widely used to evaluate data quality by analyzing metrics such as base sequence quality, GC content, and sequence duplication levels. MultiQC further enhances this process by aggregating results from multiple analyses into a single, comprehensive report. Addi- tionally, tools like Ensembl Variant Effect Predictor (VEP) are used to annotate genomic variants, providing biological context to the data. However, these tools often require sig- nificant computational resources, dependency management, and expertise in command-line environments, making them challenging to use in large-scale workflows.

Traditional approaches for genomic data processing rely on high-performance computing (HPC) clusters or local systems. While these systems are capable of handling large workloads, they often lack flexibility, scalability, and ease of use. Resource allocation in such environments is typically static, leading to underutilization during low demand and insufficient capac- ity during peak workloads. Furthermore, managing software dependencies and ensuring reproducibility across different environments can be difficult, especially for users without access to advanced infrastructure.

intervention. Containeriza- tion ensures consistency across environments by packaging applications along with their dependencies. In addition, web- based interfaces simplify user interaction, allowing seamless data upload and result retrieval without requiring deep techni- cal knowledge.

Despite the availability of cloud services, designing a fully integrated genomic data processing system that combines secure data ingestion, automated workflow execution, scalable computing, and user-friendly access remains a complex task. Challenges include handling large data volumes securely, ensuring fault tolerance, optimizing costs, tracking processing status, and supporting parallel execution of multiple jobs.

This project aims to develop a scalable, event-driven cloud- based system for real-time genomic data processing. The system enables secure upload of raw sequencing data to cloud storage, automatically triggers processing workflows us- ing AWS Lambda, and executes containerized bioinformatics pipelines on AWS ECS Fargate. The pipeline incorporates tools such as FastQC, MultiQC, and Ensembl VEP to perform quality control and analysis. Processed results are securely stored and made accessible through a web-based interface using pre-signed URLs.

By integrating cloud infrastructure with established bioin- formatics tools, the proposed system bridges the gap between large-scale data generation and efficient data analysis. The so- lution emphasizes scalability, automation, reliability, security, and cost-effectiveness, making it suitable for modern genomic research and real-time data processing environments.

2. Technologies Used

The proposed system integrates cloud computing services, containerization technologies, bioinformatics software, and secure web technologies to enable scalable and automated genomic data processing.

Cloud Infrastructure

The cloud backbone of the system is built on Amazon Web Services (AWS). Amazon S3 is used as a secure and scalable storage solution for storing FASTQ input files and processed analysis results. AWS Lambda enables event-driven workflow initiation by automatically triggering processing tasks when new files are uploaded. AWS ECS Fargate is used to execute containerized workloads without managing servers, providing scalable and on-demand compute resources. Identity and Access Management (IAM) ensures secure access control by defining user roles and permissions.

A. Containerization

Docker is used to package bioinformatics tools along with their dependencies into portable containers. This ensures consistent execution across different environments. Container images are stored and version-controlled using Amazon Elastic Container Registry (ECR).

B. Workflow Execution

The genomic processing workflow is executed through containerized pipelines triggered by AWS Lambda events. Each uploaded dataset initiates a processing task that runs independently on ECS Fargate, enabling parallel execution and efficient resource utilization.

C. Bioinformatics Tools

FastQC is used to assess the quality of raw FASTQ sequenc- ing data. MultiQC aggregates quality control reports from multiple runs into a single comprehensive summary. Ensembl

Variant Effect Predictor (VEP) is used for genomic variant annotation, providing biological insights into the processed data.

D. Frontend and Backend

The web interface is developed using React.js, enabling users to upload genomic datasets and monitor processing results. RESTful APIs facilitate communication between the frontend and backend services, ensuring seamless data flow and user interaction.

E. Security and Networking

Secure communication is maintained using the HTTPS protocol. Pre-signed URLs allow users to upload large FASTQ files directly to cloud storage without exposing sensitive credentials. Access control mechanisms ensure that only au- thorized users can access data and results.

3. Event-Driven Cloud Storage and Data Ingestion

Event-driven cloud storage and data ingestion form the foundation of the proposed genomic pipeline. This setup enables secure, scalable, and automated handling of large sequencing data.

The pipeline is designed to process raw .fastq files directly into cloud object storage using Amazon S3. S3 provides durable, scalable, and cost-effective storage for large genomic datasets. Next-generation sequencing files can range from hun- dreds of megabytes to several gigabytes, requiring the storage system to support high throughput, reliability, and seamless scalability without manual intervention. Object storage meets these requirements by automatically adapting to varying data volumes while maintaining data durability and integrity.

To prevent performance bottlenecks and server overload, the system uses secure direct uploads through pre-signed URLs. Instead of routing large files through the application server, the frontend requests a temporary upload token from the backend. Once received, the file is uploaded directly to the storage bucket. This approach reduces server load, improves upload speed, and enhances security by restricting access through time-limited credentials. Data is encrypted both in transit and at rest to ensure privacy and compliance with data protection standards.

The event-driven mechanism is implemented using storage- level notifications that trigger automatically when a new .fastq file is uploaded. Upon successful upload, the storage service generates an event containing metadata such as the bucket name, object key, and upload timestamp. This event is pro- cessed by AWS Lambda, which acts as the orchestration trig- ger. The Lambda function extracts the required details, gener- ates a unique job ID, and initiates the containerized processing workflow on AWS ECS Fargate. Since the orchestration layer is serverless, it scales automatically with incoming events, enabling the system to handle multiple uploads concurrently without manual scaling.

In addition to triggering computation, the ingestion layer stores structured metadata. Information such as file identifiers, upload timestamps, user associations, and processing status is stored in a persistent database for tracking and monitoring. This metadata-driven approach allows the frontend to display real- time job status updates, ensures reproducibility of analy- sis, and supports efficient error handling.
The decoupled architecture also enhances fault tolerance. Even if processing fails at later stages, the uploaded data remains securely stored, allowing reprocessing without data loss.

In summary, the event-driven storage and ingestion layer transforms the genomic pipeline into an automated and reac- tive system. By integrating cloud object storage, secure upload mechanisms, and serverless event triggers, the system ensures real-time workflow initiation and efficient processing. This architecture supports high-throughput genomic data analysis in a scalable and reliable cloud-native environment.

4. Containerization and Bioinformatics Tool Integration

Containerization and the integration of bioinformatics tools form a core component of the proposed cloud-native genomic pipeline. These tools consist of multiple software applications with specific dependencies, which can be difficult to manage in traditional environments. Dependency conflicts and version mismatches often lead to inconsistent results and reduced reproducibility. To address these challenges, the proposed system utilizes a containerized environment that packages all required tools and dependencies into a single, portable unit.

The workflow incorporates widely used bioinformatics tools such as FastQC for assessing the quality of raw sequenc- ing data, MultiQC for aggregating and summarizing quality metrics across multiple datasets, and Ensembl Variant Effect Predictor (VEP) for genomic variant annotation. These tools are bundled into a Docker container image that includes the operating system, required libraries, and application dependen- cies.

A Dockerfile is used to define the runtime environment, ensuring that the same configuration can be recreated con- sistently across different platforms. This approach guarantees that the execution environment remains independent of the un- derlying infrastructure, eliminating conflicts between system- level and application-level dependencies. Such consistency is critical in genomic workflows, where specific versions of software packages, programming libraries, and tools are required for accurate results.

Within the container, a processing script defines the se- quence of operations in the pipeline. Starting with a .fastq file as input, the pipeline first executes FastQC to generate quality control metrics such as base sequence quality, GC content, and duplication levels. The outputs from FastQC are then processed by MultiQC, which consolidates the results into a single, user-friendly HTML report. Following quality control, Ensembl VEP is used to perform variant annotation, providing meaningful biological insights. The final outputs, including the multiqc.html report and annotation results, are stored in an output directory and uploaded to cloud storage.

The container image is stored in a managed container registry, enabling version control and consistent deployment across computing environments. Version tagging supports con- trolled updates and rollback mechanisms, which are essential for maintaining system stability. During execution, AWS ECS Fargate pulls the container image from the registry and runs it in an isolated runtime environment, ensuring consistent performance regardless of the underlying infrastructure.

This containerized approach supports horizontal scalability by allowing multiple genomic datasets to be processed si- multaneously in independent container instances. Each task runs in isolation, preventing interference between jobs and enabling efficient parallel execution. The system dynamically allocates compute resources based on workload, ensuring optimal performance and cost efficiency.

Additionally, containerization simplifies maintenance and system upgrades. New tools or updates can be incorporated by modifying the container definition and rebuilding the image, without affecting the orchestration layer.

In conclusion, containerization provides a unified, repro- ducible, and scalable framework for integrating bioinformatics tools. By using isolated container environments for FastQC, MultiQC, and Ensembl VEP, the system ensures consistent ex- ecution, improved reliability, and efficient large-scale genomic data processing in a cloud-native architecture.

5. Scalable Compute Orchestration Using AWS Batch

The ability to manage scalable compute resources is essen- tial for processing genomic data in real time. Genomic data, typically stored in .fastq files, can be extremely large, espe- cially in high-throughput sequencing scenarios. Processing this data involves multiple steps such as quality assessment and variant annotation, which require significant computational resources, including CPU and memory. A fixed infrastructure can lead to inefficient resource utilization, either underutilizing resources during low demand or failing to handle peak work- loads effectively. To address these challenges, the proposed system uses AWS ECS Fargate for scalable and reliable execution of the genomic workflow.

AWS ECS Fargate is a serverless container orchestration service that allows applications to run without managing the underlying infrastructure. In the proposed system, uploading genomic data triggers an event-driven workflow, where AWS Lambda initiates a containerized processing task on ECS Far- gate. Each task is configured with specific CPU and memory requirements, environment variables, and container definitions, ensuring consistent execution across all workloads.

The orchestration process begins when an event-triggered Lambda function processes the uploaded file metadata and initiates a container task. ECS Fargate manages task schedul- ing and execution without requiring manual provisioning of virtual machines. Each processing job runs independently in an isolated container environment, ensuring reliability and consistency regardless of scale.

A key advantage of ECS Fargate is its ability to scale dynamically. When multiple genomic files are uploaded simul- taneously, the system can launch multiple container instances in parallel, allowing concurrent processing of datasets. Since Fargate operates on a serverless model, compute resources are allocated only when tasks are running, eliminating idle infrastructure and reducing operational costs.

Fault tolerance is inherently supported through managed orchestration. If a task fails due to temporary issues, it can be retried based on predefined configurations. Logs generated during execution are centrally collected using monitoring ser- vices, enabling efficient debugging and performance tracking. Resource limits ensure that individual tasks do not consume excessive system capacity, maintaining overall system stability. The containerized nature of the pipeline integrates seam- lessly with ECS Fargate. When a task is triggered, the required container image is pulled from the container

registry and exe- cuted in a secure, isolated runtime environment. This ensures that all processing tasks use the same software configuration, improving reproducibility and reliability. Since each dataset is processed independently, the system supports parallel execution and high throughput.

From a cost optimization perspective, ECS Fargate provides a pay-as-you-use model, where users are billed only for the compute resources consumed during task execution. This eliminates the need for maintaining dedicated infrastructure and allows efficient scaling based on workload demands. Security is enforced through identity and access manage- ment policies that follow the principle of least privilege. Each task is granted only the permissions required to access storage and related services. Network configurations further enhance security by isolating compute tasks within controlled environments.

In conclusion, the use of AWS ECS Fargate for scalable compute orchestration provides a flexible, efficient, and re- liable infrastructure for the genomic pipeline. The system achieves elasticity through dynamic resource allocation, sup- ports parallel processing through containerized execution, and ensures fault tolerance and security in a cloud-native environment.

6. Frontend Integration and Automated Result Delivery

Frontend integration and automated result delivery are es- sential components of the proposed genomic pipeline, acting as the interface between complex cloud-based processing and end users. While the backend handles data storage, workflow or- chestration, and large-scale computation, the frontend provides a simple and user-friendly interface for interaction. This design ensures that users can operate the system without requiring knowledge of command-line tools or cloud infrastructure.

The frontend is developed as a web application using React, which communicates with backend services through RESTful APIs to manage file uploads, job execution, and result retrieval. A key design objective is the secure and efficient transfer of large .fastq files. Instead of routing uploads through the application server, the frontend requests temporary upload permissions from the backend. The backend generates pre- signed URLs, allowing users to upload files directly to cloud storage. This approach improves performance, reduces server load, and enhances security by using time-limited access credentials.

Once a file is uploaded, the system transitions to an asyn- chronous processing model. The frontend receives a unique job ID generated during the orchestration process. This job ID is used to track the status of the processing task through periodic API calls. The backend retrieves status information from the orchestration layer and metadata storage, enabling the frontend to display real-time updates such as submitted, running, completed, or failed.

Automated result delivery is tightly integrated with the stor- age and processing layers. After the containerized workflow is executed on AWS ECS Fargate, output files such as the multiqc.html report and variant annotation results are stored in a designated cloud storage location. The backend gener- ates secure pre-signed URLs for downloading these results, ensuring that only authorized users can access them. These temporary links allow users to retrieve results directly without exposing sensitive data.

A key architectural feature is the separation between user interaction and backend execution. Since genomic processing tasks can take significant time depending on dataset size, the frontend is designed to handle asynchronous workflows efficiently. Users can leave the application and return later to check job progress using the job ID, without losing context.

Security and data privacy are critical considerations in the frontend layer. All communication between the client and backend services is secured using HTTPS. Access to uploaded data and generated results is controlled through authentication mechanisms and least privilege access policies, ensuring that users can only access their own data.

In addition to usability, the frontend abstracts the complexity of the underlying system. Users are not required to understand container execution, cloud orchestration, or storage manage- ment. Instead, they interact with a streamlined interface that supports file upload, status monitoring, and result download.

From a system design perspective, frontend integration com- pletes the end-to-end workflow. The ingestion layer captures and stores data, the orchestration layer processes it at scale, and the frontend delivers results in a secure and accessible manner. This integration enables efficient, automated, and user-friendly genomic data processing in a cloud-native en- vironment.

The system is built as a cloud-based and event-driven setup to automatically and efficiently manage large amounts of genomic data. It combines storage, orchestration, scalable computing, metadata handling, and user interaction into one seamless workflow. The goal is to quickly process big .fastq files without requiring users to set up their own infrastructure.

The process starts with cloud storage, where raw genomic data is securely uploaded. This storage layer ensures reliability, durability, and the ability to handle large files, from gigabytes to terabytes. Instead of sending all data through a central server, users can upload files directly using temporary access keys. This approach reduces server load and speeds up the process.

Once the data is uploaded, an automatic event is triggered. This event starts a serverless function that gathers file details, assigns a unique job ID, stores the data in a metadata system, and sends a job request to the computing part.

The computing section uses managed batch services. Each dataset is processed in a container with tools like FastQC, MultiQC, and analysis modules. Containers ensure consistent execution and reliable results.

Metadata is used to track the progress of each job, showing whether it is started, running, completed, or failed. After processing is done, results like quality reports and summaries are securely stored and users can download them via safe temporary links.

7. SYSTEM ARCHITECTURE

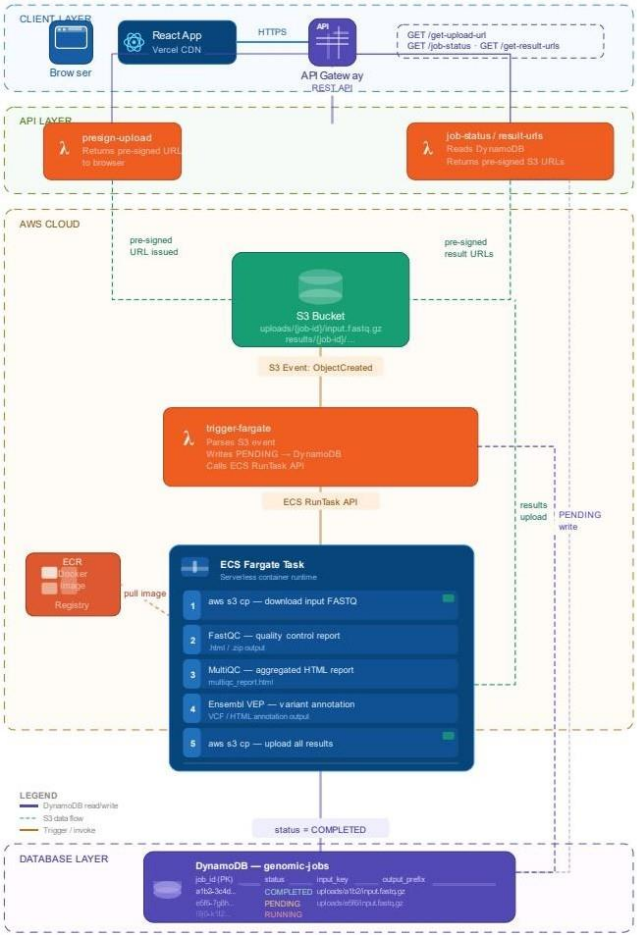

Fig. 1: High-level cloud-native architecture for real-time ge- nomic pipeline processing.

The proposed system follows a cloud-native, event-driven architecture designed to handle large-scale genomic data pro- cessing efficiently and automatically. The architecture inte- grates multiple cloud services and technologies, including object storage, serverless computing, container orchestration, metadata management, and a web-based user interface.

At the core of the architecture is the cloud storage layer, where raw genomic data in .fastq format is securely uploaded. The system uses pre-signed URLs to enable direct uploads from the frontend to cloud storage, eliminating the need for intermediate servers. This approach improves performance, reduces latency, and enhances security by using time-limited access credentials.

Once a file is uploaded, an event notification is generated within the storage system. This event triggers a serverless function, which acts as the orchestration entry point. The func- tion extracts file metadata, generates a unique job identifier, and stores relevant information in a metadata database for tracking and monitoring purposes.

The processing layer is implemented using AWS ECS Fargate, which runs containerized bioinformatics workflows. Each uploaded dataset is processed independently within an isolated container environment. The container includes tools such as FastQC for quality assessment, MultiQC for report aggregation, and Ensembl Variant Effect Predictor (VEP) for variant annotation. This ensures consistency, reproducibility, and efficient execution across all tasks.

The system supports horizontal scalability by allowing mul- tiple container instances to run in parallel. Since ECS Fargate is serverless, compute resources are allocated dynamically based on workload demand, ensuring optimal performance without maintaining dedicated infrastructure.

A metadata management layer is used to track the lifecycle of each job. Information such as job ID, upload time, pro- cessing status, and output references is stored and updated throughout the workflow. This enables real-time monitoring and seamless integration with the frontend interface.

After processing is completed, output files such as quality reports and annotated results are stored in cloud storage. Secure pre-signed URLs are generated to allow users to download results safely without exposing sensitive data.

The frontend layer provides a user-friendly interface for uploading files, tracking job progress, and accessing results. It communicates with backend services through APIs and supports asynchronous workflows, allowing users to interact with the system without waiting for processing to complete.

Overall, the system architecture ensures automation, scal- ability, security, and reliability. By combining event-driven workflows, serverless orchestration, and containerized exe- cution, the architecture efficiently supports high-throughput genomic data processing in a modern cloud environment.

8. Workflow Overview

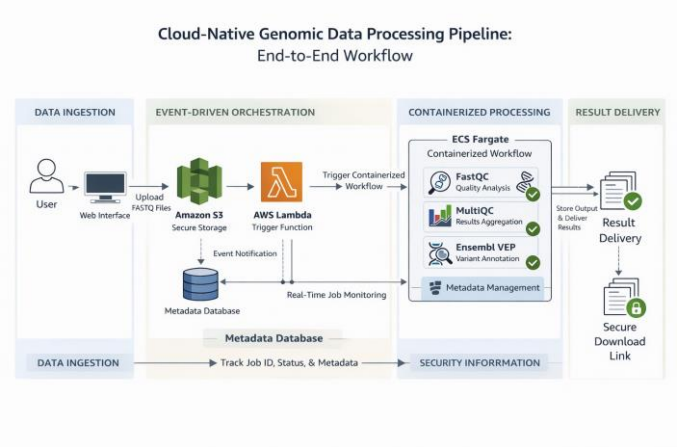

Fig. 2: End-to-end workflow of the cloud-native genomic pipeline, showing data ingestion, event-driven orchestration, containerized processing, and result delivery.

This diagram represents the complete workflow of the cloud-native genomic data processing pipeline, illustrating each stage from data ingestion to result delivery.

The process begins with the user uploading raw genomic data in FASTQ format through a web-based interface. To ensure efficient and secure data transfer, files are uploaded directly to cloud storage using pre-signed URLs, eliminating the need for routing data through an application server.

Once the file is successfully stored, an event notification is automatically triggered. This event activates a serverless function, which extracts relevant metadata, generates a unique job identifier, and initiates the processing workflow.

The processing stage is executed using AWS ECS Fargate, where containerized bioinformatics pipelines are run in iso- lated environments. Each container processes the input data using tools such as FastQC for quality analysis, MultiQC for report aggregation, and Ensembl Variant Effect Predictor (VEP) for variant annotation. This ensures consistency, repro- ducibility, and efficient handling of computational tasks.

Throughout the execution, the system maintains metadata to track the progress of each job. Status updates such as initiated, running, completed, or failed are recorded and made accessible to the frontend, enabling real-time monitoring.

After processing is completed, the generated outputs, in- cluding quality control reports and annotated results, are stored securely in cloud storage. Users can then access and download these results through secure pre-signed URLs.

Overall, the workflow is designed to be modular, scalable, and fully automated. By integrating event-driven orchestra- tion, serverless components, and containerized execution, the system ensures reliable and efficient processing of large-scale genomic data in a cloud-native environment.

9. Conclusion and Future Work

In this project, we designed and implemented a scalable cloud-native pipeline for genomic data processing using mod- ern cloud technologies. The system leverages an event-driven architecture to automate the ingestion, processing, and delivery of genomic datasets. By integrating cloud storage, serverless orchestration, and containerized execution using AWS ECS Fargate, the pipeline ensures efficient handling of large-scale.fastq files.

The use of containerization enabled consistent and repro- ducible execution of bioinformatics workflows, incorporating tools such as FastQC for quality analysis, MultiQC for report aggregation, and Ensembl Variant Effect Predictor (VEP) for variant annotation. The architecture supports dynamic scaling, allowing multiple datasets to be processed in parallel without manual resource management.

The implementation demonstrates improved performance, scalability, and cost efficiency compared to traditional com- puting approaches. Features such as secure data transfer using pre-signed URLs, metadata tracking, and asynchronous pro- cessing enhance both usability and system reliability.Overall, the proposed system provides a robust, automated, and user-friendly framework for real-time genomic data anal- ysis in a cloud-native environment.

Looking ahead, there are several opportunities to further en- hance the system. Integrating Kubernetes

(AWS EKS) can pro-vide more advanced orchestration and resource management capabilities. Enhancing metadata management using scalable databases such as MongoDB or PostgreSQL can improve tracking, reproducibility, and data organization. Implementing cost monitoring and optimization strategies can further reduce operational expenses.

Additionally, extending support for more bioinformatics tools will broaden the analytical capabilities of the system. Modularizing the workflow can make the pipeline more flex- ible and easier to maintain. Finally, developing an advanced real-time monitoring dashboard can improve user experience by providing better visibility into job execution and system performance.

REFERENCES

1. M. A. Quail, M. Smith, P. Coupland, et al., "A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers," BMC Genomics, vol. 13, no. 1, pp. 1–13, 2012.
2. D. C. Koboldt, K. Chen, T. Wylie, et al., "VarScan: variant detection in massively parallel sequencing of individual and pooled samples," Bioinformatics, vol. 25, no. 17, pp. 2283–2285, 2009.
3. P. Di Tommaso, M. Chatzou, E. W. Floden, et al., "Nextflow enables reproducible computational workflows," Nature Biotechnology, vol. 35, no. 4, pp. 316–319, 2017.
4. C. P. Beaulieu-Jones and J. H. Greene, "Reproducibility of compu- tational workflows is automated using continuous analysis," Nature Biotechnology, vol. 35, no. 4, pp. 342–346, 2017.
5. D. Merkel, "Docker: lightweight Linux containers for consistent devel- opment and deployment," Linux Journal, vol. 2014, no. 239, pp. 1–6, 2014.
6. J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
7. M. Schatz, "Cloud computing and the DNA data race," Nature Biotech- nology, vol. 28, no. 7, pp. 691–693, 2010.
8. Langmead and S. Salzberg, "Fast gapped-read alignment with Bowtie 2," Nature Methods, vol. 9, no. 4, pp. 357–359, 2012.
9. Van der Auwera, M. Carneiro, C. Hartl, et al., "From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline," Current Protocols in Bioinformatics, vol. 43, no. 1, pp. 11–33, 2013.
10. Amazon Web Services, "AWS cloud computing overview," AWS White Paper, 2023.
11. Y. Wang et al., "GT-WGS: An efficient and economic tool for large-scale WGS analyses based on the AWS cloud service," 2018.
12. M. Bourgey et al., "GenPipes: an open-source framework for distributed and scalable genomic analyses," 2019.
13. Z. Huang, J. Yu, F. Yu, "Cloud processing of 1000 genomes sequencing data using Amazon Web Service," 2013.
14. N. Kumar and S. K. Sharma, "A Cost-Effective and Scalable Processing of Heavy Workload with AWS Batch," 2022.
15. M. Frampton and R. Houlston, "Generation of Artificial FASTQ Files to Evaluate the Performance of Next-Generation Sequencing Pipelines," 2012.