National College of Ireland

Evaluation of GWO -PSO Algorithm using AWS Lambda

MSc Research Project
Master of Science in Cloud Computing

Ann Mariya Jojo
Student ID: x23241535

School of Computing
National College of Ireland

Supervisor: Shreyas Setlur Arun

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Ann Mariya Jojo
Student ID:	x23241535
Programme:	Master of Science in Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Shreyas Setlur Arun
Submission Due Date:	20/12/2024
Project Title:	Evaluation of GWO -PSO Algorithm using AWS Lambda
Word Count:	9200
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ann Mariya Jojo
Date:	25th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluation of GWO -PSO Algorithm using AWS Lambda

Ann Mariya Jojo
x23241535

Abstract

The novel approach to load balancing in Cloud Environment with the help of integrating the AWS services with optimized hybrid GWO-PSO algorithm.The main objective of the approach is to attain a efficient and scalable traffic managing system with the help of Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO) algorithm. The hybrid algorithm is implemented on AWS Lambda.Lambda is known for it's serverless architecture and scaling capability.For dynamic traffic distribution the Application load balancer is used and EC2 instances acts as the backend server.. The main advantages are resource utilization through efficient workload distribution across the instances.The success criteria considered for this approach are based on CPU utilization,minimal error rates and optimal response time.Comparison is performed between the traditional VM based approach and the hybrid GWO-PSO algorithm on AWS Lambda to analyze the valuable insights for the system performance .The approach guarantees the robust fault tolerance.

1 Introduction

Cloud Computing has made a significant impact in technology by maintaining scalable,cost effective and flexible solution to meet the computational requirement of the individuals and the organization.It helps to use resources dynamically by improving efficiency and also saving the cost. The main challenges now facing by cloud infrastructure is to distribution of the load and managing the resources.For a better optimized performance effective load distribution is crucial for improving resource utilization.
As part of solving the optimization problem the traditional methods and also the hybrid algorithm Gray Wolf Optimizer (GWO) and Particle Swarm Optimization (PSO) have showed the significant changes in load balancing.The GWO shows the hunting behavior of wolves whereas PSO is based on the collective movement of birds and fish.Both the algorithm are suitable for optimization challenges because both algorithm are known for convergence efficiency and also the simplicity.
On other hand of successful implementation there were limitation that were faced by these methods.Mainly handling the complexity and also the nature of cloud environment.The main concern was the computational overhead and in some scenarios dynamic resource allocation and also the quick random response.This study uses the exploration capability of GWO algorithm and and exploitation capability of PSO algorithm. The traditional approach perform well in static and dynamic environment but the hybrid GWO-PSO is designed in such a way that it can withstand on varying high traffic load.

One of the main advantage of AWS Lambda is serverless computing.The algorithm can be optimized further with help of Lambda since it is auto scalable and the computing model is event driven.

1.1 Motivation

The main motivation of the research project is to analyze the limitation of traditional load balancing methods in the cloud environment and to adapt a system where dynamic workloads works seamlessly when there is a spike in traffic pattern and also have efficiency in resource allocation.
The desired scalability and efficiency is not achieved by the algorithm in standalone that is in conventional approaches.The hybrid algorithm overcome these challenges without any limitation with the help of serverless platform AWS Lambda.The hybrid algorithm can be optimized more with the help of this service.Therefore the evaluation between traditional based approach and hybrid algorithm will give valuable insights for practical real scenario optimization.

1.2 Goals

- The first stage is to develop and implement the hybrid GWO-PSO on AWS Lambda and integrating along with Cloud Watch to monitor the live metrics of the instance and Elastic Load balancer for the distribution of traffic
- Compare the scenario of Case 1 and Case 2 using the critical factors such as execution time, throughput, and overall efficiency .The case 1 is traditional VM based approach and the Case 2 is AWS lambda based implementation.
- Evaluation of Scalability and adaptability in Case 2 scenario where hybrid GWO-PSO algorithm is implemented on AWS Lambda for effective load balancing.
- Comparing the cost and resource management in both scenarios

1.3 Research Question

The main research question is :

Does the implementation of the hybrid GWO-PSO algorithm on AWS Lambda provide superior performance, scalability, and cost-efficiency comparing to traditional VM-based architectures for load balancing and task scheduling in cloud environments?

By resolving this question it will be able to answer whether hybrid GWO-PSO algorithm on AWS lambda for effective load balancing during high traffic load effectively considering resource utilization than traditional VM based approach.

2 Related Work

The hybrid algorithm that is GWO(Grey Wolf Optimizer) and FWA (Firecrackers Algorithm is discussed in the research paper Yue et al. (2020).The GWO is considered to

be the best algorithm to find optimized solution due to it's ideal hunting capabilities.On other hand FWA has exploration capabilities which helps in wide range of capabilities but it takes time to settle on a final solution.In this research study GWO and PSO algorithm is used .PSO helps in convergence and GWO helps in finding good solution.

The Al Reshan et al. (2023) discuss mainly about load balancing in the cloud computing.In the load balancing the workloads are distributed across multiple virtual machines for resource usage and to maximize the output.Therefore to have effective optimization the hybrid GWO-PSO algorithm is used for effective load balancing across EC2 instances.The load is dynamically adjusted for the hybrid algorithm based real time live metrics.Therefore it guarantees that none of the virtual machine will be high overloaded since resources are used efficiently.This proposal helps to avoid the bottlenecks and ensure resource utilization by maintaining the overall system performance.Therefore this paper showcases how the hybrid GWO-PSO algorithm can optimize load balancing than traditional VM based approaches.

To improve the system performance task scheduling algorithm plays a major role in the current era Chandrashekar et al. (2023).The factors makespan in the algorithm that is total time to complete the task these factors needs to be optimized.But due to complex search process it reduces efficiency.In this research the author have introduced Hybrid Weighted Ant Colony Optimization (HWACO) algorithm.This new approach helps in the improvement of the ACO algorithm.The main two factors that helps in enhancing are Weighted optimization and Dynamics Pheromones Updates.Based on importance different priority is given to different task in weighted Optimization.The trails of pheromone is adjusted flexibly which helps on adapting quickly to new changes and improve efficiency.Therefore HWACO become more beneficent comparing to other traditional algorithm.

For efficient managing of traffic for Hai et al. (2023) proposed a optimized task scheduling which has reduced execution time and better resource utilization using Heterogeneous Earliest Finish Time (HEFT) algorithm. Mainly this algorithm is used for task scheduling and it has two stages that is Rank generation and other is Processor Selection.In rank generation it calculate the priority of each task through rank and the factor that influence are communication cost and computation.The second stage is processor selection where based on ranks the task is assigned to virtual machines for execution.The main limitation of the algorithm is it select the first empty available slot for scheduling which means it uses average computation compared to other algorithm.There were three modified version they are Average Computation Cost Technique, Maximum Computation Cost Technique and Minimum Computation Cost Technique.Based on computation need and dependency each task is assigned a rank.So the major finding is using average computation cost alone as factor is not optimal.

The Makhija et al. (2022) works on task scheduling using hybrid GWO -PSO algorithm.It uses exploitation and exploration to execute the algorithm .Task allocation is optimized by assigning to virtual machines.The makespan is reduced and the degree of imbalance is also reduced .The algorithm is tested using CloudSim which showed the lifespan of 5.52 percentage and imbalance of 33.22 percentage to have higher convergence.

The Khan (2024) of research paper involve the methodology that involves creation of a dynamic load balancing with the help of cloud computing.The machine learning model that is CNN and RNN is used to compute the load on the virtual machine.A clustering mechanism divides the virtual machine based on the computation load. It also include Hybrid Lyrebird Falcon Optimization (HLFO) algorithm to demonstrate task scheduling

and effective clustering.Task scheduling is optimized with the help of QoS metrics which includes energy consumption,memory utilization and make span.For dynamic distributing of work load the python and cloud sim is used.In the result it is showcased that HLFO shows higher performance than traditional approach like FOA.Lower make span is shown with reduced energy consumption.It also suggested that the methodology contributes to the effective resource utilization and task allocation which is beneficial to load balancing in cloud computing.
The paper proposed Lester et al. (2020) involves a qualitative research focusing in systematic approach. That is thematic analysis helps in analyzing the qualitative data by organizing data and includes coding to find patterns which helps to understand the information in better manner.Same like this in cloud computing it follow a clean process to understand the optimization of the resources and also to manage.For thematic analysis look through each data to find the pattern and in the cloud computing almost same approach is used but related to the demand of the system.First resource usage is analyzed and identify the pattern of the system which means which portion of the system is overused or which is not used to evaluate the performance.In cloud computing there are advanced techniques such optimization and dynamic techniques . Therefore applying thematic analysis into the field of cloud computing helps in understanding the behavior of the system performance.Therefore better strategies can be developed for the optimization of the resources.

The Liu and Wang (2021) discuss about the ceiling effect in data caused mainly due to statical method like -tests and ANOVA.It skew the result when data point comes near to the upper limit which makes it hard for the correct conclusion.In the case of cloud computing also the hybrid GWO -PSO algorithm finds difficult to make decision that is best due to several instances are equally workload.To overcome this scenario it is very important to analyze the ceiling the effects in load balancing .So for that performance metrics should be checked continuously to analyze the limits of each instances . For that metrics may involve CPU utilization.Then the statistical correction discussed in Liu and Wang (2021)would be helpful to adapt the algorithm based on understanding.To prevent the ceiling effects the scaling of the system resource can performed before they reach their maximum limit. Like redistributing the task so the system runs smoothly.

The analysis related to traffic discussed in Muller et al. (2020) mainly discuss about AWS lambda behavior such it's performance.It is beneficial for load balancing with the help of hybrid GWO-PSO algorithm for optimization of traffic management and to improve the efficiency of the system.The main issue that is highlighted in the paper is about the AWS lambda cold latency it happens when Lambda starts initially which causes delay for the response.To avoid this the hybrid GWO-PSO hybrid algorithm can helps in traffic distribution efficiently by transferring to active instances therefore faster response is achieved .It helps in reducing the side effect of the AWS lambda that is cold latency. Other performance that discussed in the research paper is about the automatic scaling behavior in which instance become inactive after low activity.But the algorithm can reduce this negative effective with help of effective distribution during high traffic time.Research paper Muller et al. (2020) also discussed about the virtual machine has limited life span.Therefore the hybrid algorithm should consider the age of the instance during distribution of work load. Considering all the factors of lambda helps in effective traffic distribution with the help of hybrid algorithm.

The Gwo with Simulated Annealing(GWO-SA) is proposed in the research paper Patra et al. (2022) for containerized cloud environment to balance work load.Therefore

Table 1: Comparison of Approaches

Reference	Algorithm	Application/Context	Key Features
Patra et al. (2022)	GWO with Simulated Annealing (GWO-SA)	Load balancing in containerized cloud environments	- Minimized makespan - Ensures tasks meet deadlines - Utilizes containerization for workload distribution
Saif et al. (2023)	MGWO (Multi-objective Grey Wolf Optimizer)	Task scheduling in cloud-fog computing environments	- Adapting load balancing to distributed architectures - Algorithm can be extended to cloud-fog environments - Considers similar multi-objective optimizations
Shao et al. (2023)	PGSAO (Hybrid Genetic Algorithm and PSO)	Scheduling data-intensive tasks in heterogeneous cloud environments	- Potential for incorporate real-time performance data - Aligns with goal for optimizing scheduling using hybrid algorithms
Research Approach	Hybrid GWO-PSO Algorithm on AWS Lambda	Load balancing and task scheduling across EC2 instances	- Mainly Build upon GWO and PSO algorithms - Integration of serverless benefit of AWS Lambda - Considers the characteristics of AWS - Optimization of execution time and response

tasks is distributed equally among the server available to reduce the makespan.And also it ensured that task met the deadline.This approach is almost same to the approach discussed in the research paper.In the research paper hybrid algorithm is used for the optimization of workload across EC2 instances.Both works for the same motive. In Patra et al. (2022) works with containers with algorithm which light weighted.To scale and move application containers makes it easy.

Another related approach is proposed by Pelle et al. (2020) in dynamic layout optimization.That is based on user requirement and the characteristics of the platform the better configuration selected for the application.It uses the AWS Lambda for serverless computing and for edge execution is performed with help of AWS IoT Greengrass which helps in deployment for distributed infrastructure.

The Zhan et al. (2016) discussed the cost of different architecture plan based on whether the application is monolithic or microservices for web application that is in AWS.It is significantly shown in the paper that AWS Lambda can save the cost significantly comparing to other scenarios.But there are negative impacts such as cold start latency which the function will take more time to execute initially.Reducing the cold latency also reduce the cost.The hybrid algorithm used in this research for load balancing have the benefit cost saving.The cold start can be removed by sending the workload to the warm active instance than the new ones.Therefore this cost aware strategy will be beneficial for this research project.
The Yu et al. (2013) papers discuss about the mutli-agent system (MASs) and explored trust management system within it.When the agents provide the resources it will have only limited spaces only which is almost similar to load balancing in cloud computing.When a trusted agent is given the request always it can suffer which affects it's capacity whereas in GWO-PSO algorithm takes the decision based on live metric to distribute the load which mean the trust and reputation factors matters to the algorithm. Congestion Trust Game (CTG) is the model proposed by the author Yu et al. (2013) for understanding the overload is influenced by the trust.
PGSAO algorithm is proposed by Shao et al. (2023) which is hybrid of genetic algorithm and PSO algorithm for distributing data intensive task.The main focus was on completion of task before the deadline and also to attain proper resource utilization which is similar to the approach implemented in this paper.The main connection is the real time performance in hybrid GWO-PSO algorithm is a important feature than PGSAO framework.

3 Methodology

This section mainly discuss about the tool, technology and process used as part of experimentation. The experimentation consist of two section Case 1 and Case 2. It consist of cloud Architecture , Lambda function which facilitates the optimal traffic routing using EC2 instances based on real time data metrics and monitoring mechanism. To achieve efficient and scalable application load balancing every component is important.

3.1 Overview of the Experiment

The main objective is to analyze and implement a load balancing architecture using AWS services along with Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO) algorithm. The tools and technologies that is used is given below:

- **AWS Lambda** - Mainly used to execute custom logic for routing traffic optimally based on metrics from EC2 instances and AWS Lambda is serverless computing architecture.

- **AWS Application Load balancer** - It accommodates the incoming HTTP request and forward it to the targets based on the scenario including Lambda.
- **Amazon Cloud Watch** - The service is mainly used to monitor the metrics of EC2 instances including CPU utilization.
- **AWS EC2 instances** - The instances is used as backend services for hosting applications to this traffic is routed.
- **Boto3** - To ensure seamless integration with AWS services for Python SDK.
- **Python** - The programming language used is python for the implementation of GWO-PSO algorithm using AWS Lambda.

3.2 Tools and Technologies

3.2.1 Amazon Web Services

The experiment mainly used AWS services to maintain reliable , scalable and flexible architecture. The key components of AWS includes:

AWS Lambda

- The custom code can be executed without managing the servers that is serverless execution is possible.
- It also handles HTTP request from the application load balancer and based on real time metrics of EC2 instances the traffic routes to instances.
- For optimal decision making the hybrid GWO-PSO algorithm is implemented.

Application Load balancer

- Based on defined rules the ALB routes traffic to Lambda or backened targets based on the scenario.
- To ensure targets are healthy before routing traffic, health is checked.
- Two path is configured one for health check up /health and other for routing /lambda.

EC2 instances

- For processing the request three EC2 instances acts as backened services.
- Sample Web page for confirming the routing.

Amazon CloudWatch

- The CloudWatch is used for collecting the metrics such as CPU utilization for EC instances.
- For debugging and Optimization it provides detailed monitoring and logging.

3.3 Programming and Tools

Python

- For the implementation of Lambda function python programming language is used.
- For handling HTTP request and integration of AWS , different libraries are used for example requests,boto3.
- Python is used for the implementation of hybrid GWO-PSO algorithm.

Boto3

- For the interaction between CloudWatch, Application Load balancer and EC2 instances , Python SDK for AWS .
- Helps in retrieving of metrics such as CPU utilization and metadata of the instance.

Requests

- It helps in handling the HTTP requests from AWS Lambda to the backend EC2 instances.
- Helps in simulating the traffic request forward.

Shell and Curl Commands

- Mainly used to test the response of the AWS Lambda and endpoints of Application Load balancer.
- To handle proper routing along with health check output.

4 Design Specification

The architecture mainly demonstrates the way of managing traffic across multiple EC2 instances using serverless AWS Lambda and an Application Load Balancer.The architectures demonstrates the interaction between different services to maintain scalability and dynamic traffic routing.

Client / End User

- **Role** - The HTTP request is initiated from the client side for example through an application or browser to interact with the system.
- **Main Features** - Helps in handling the send request to the DNS end point of the application load balancer.
- Handles the processed response from backend through same Application Load balancer.

Application Load Balancer

- **Role** - For all incoming traffics it acts as the entry point.

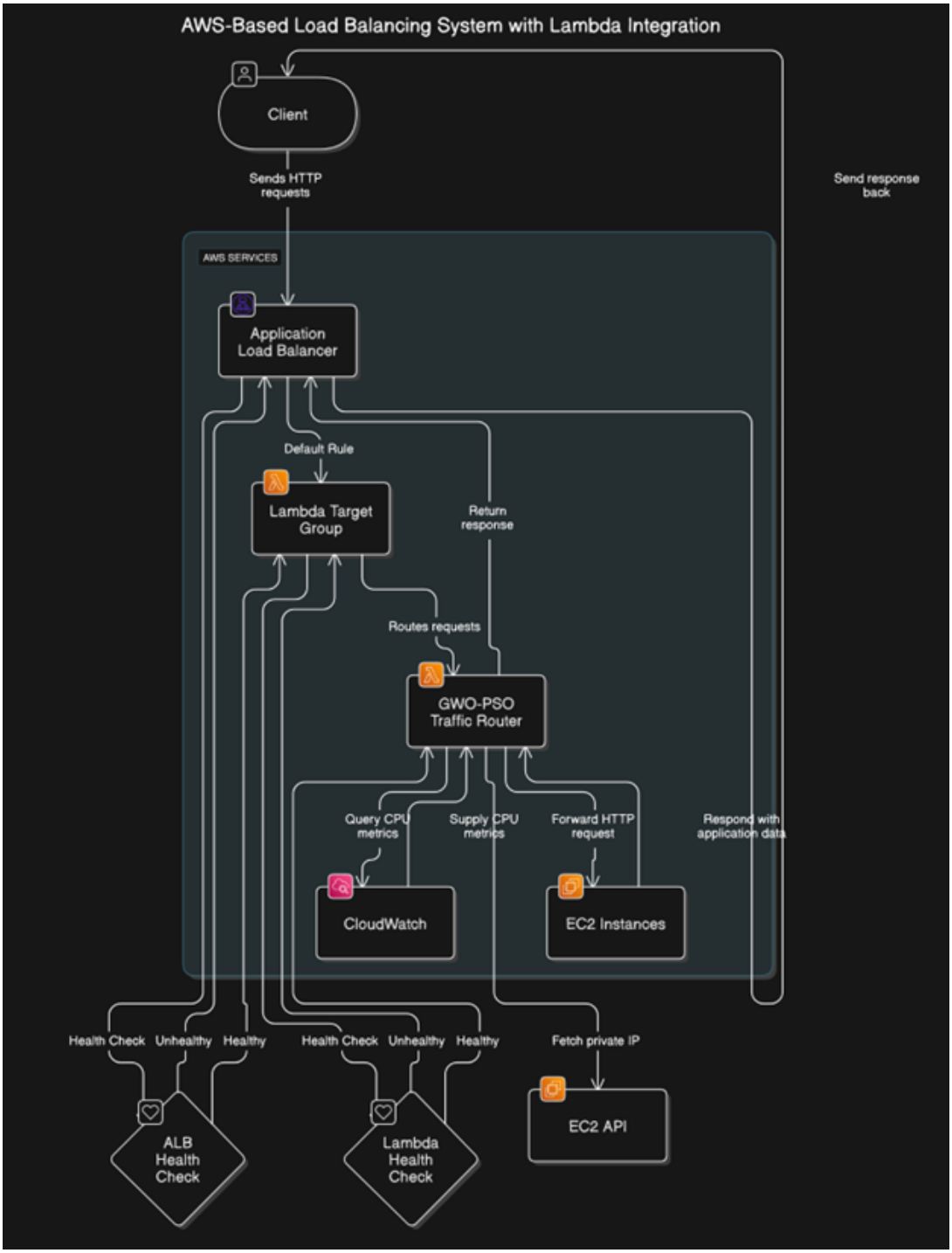

Figure 1: Architecture Diagram

- **Main Features** - It ensures the load is dynamically distributed across targets in this scenario it is Lambda.
- To understand the correct routing path it analyze the listener.
- It also checks the health status of the Lambda for routing traffic to the lambda.
- **Why Application Load Balancer ?** - It handles the workload dynamically
- For HTTP and HTTPS request , it helps in the integration with Lambda function.

Lambda Target Group

- **Role** - Main purpose is to receive the incoming traffic from Application Load balancer.
- **Main Features** - Routing traffic into the specific Lambda for optimization of load balancing.
- To evaluate the health of the Lambda configured the health check up.
- When the lambda function is unhealthy it returns 502 bad gateway.

AWS Lambda Function (GWO-PSO Traffic Router)

- **Role** - The main role is to make decision and move forward .
- **Main Features** - From ALB all the health check message is handled and return a success message.
- Implementation of hybrid Grey Wolf Optimizer-Particle Swarm Optimization (GWO-PSO) algorithm for the optimization of load balancing by selecting EC2 instance which has less CPU utilization.
- To get the private IP address of the selected instance using Amazon EC2 API .
- The private IP address is used to send the request from client side to the optimal EC2 instance.
- **Why Lambda ?** - Main advantage is does not need to manage servers and provision server.
- Varying traffic load can be handled by automatic scaling.

Amazon CloudWatch

- **Role** - The continuous monitoring of metrics for both EC2 instances and Lambda.
- **Main Features** - For EC2 instances it monitors the CPU utilization.
- Make sure based on real metrics the dynamic traffic routing is happening.
- All the logs of AWS Lambda is handled.

Amazon EC2 Instances

- **Role** - Handling the client request along with hosting the application backend.
- **Main features** - The traffic forwarded from Lambda is received in EC2 instances.
- Later route back to the client and also respond with the requested data.
- **Why multiple EC2 instances?** - For the effective load distribution and high availability.
- To handle faults and failures.

Traffic and Response Flow

- **Traffic flow** - The HTTP request from the client is sent to ALB.
- Request is forwarded to AWS Lambda target group from ALB.
- The Lambda function process the request and make decision based on metrics such as CPU utilization for selecting optimal EC2 instance.
- To the selected EC2 instance the request is forwarded.
- **Response Flow** - The response is sent back to Lambda after EC2 instance process the request.
- Then the response is send to ALB from Lambda.
- To the client the response is send from the ALB.

Advantages of the Architecture

- **Scalability** - Based on traffic the Lambda and Application load balancer scale automatically.
- **Cost-Effectiveness** - This architecture eliminates the need of compute resources because Lambda handles traffic routing.
- **Resilience** - In the case of failures the EC2 instances ensures system reliability .
- **Dynamic Traffic Routing** - In this architecture using Hybrid GWO-PSO algorithm optimal EC2 instances is taken based on real metrics for optimal Load balancing.

Therefore the architecture ensures fault tolerance, scalability and optimal resource utilization by maintaining less cost.

5 Implementation

The implementation is performed as 2 cases. That is Case 1 and Case 2.

5.1 Case 1

The main objective is to simulate and measure the performance of a web application under a medium level load that is 20 requests per second with use of artillery and evaluate the behavior.

1. Setup Prerequisites

- Amazon EC2 instance which is t2 micro seconds is launched using Amazon Linux 2.
- The necessary system update is done using the command .

Node.js is Installed

- The artillery requires Node.js and npm. Therefore the commands are used for necessary installation.
- Later verification is done to check if it's installed or not.

Artillery is installed

- Artillery is installed globally on the system to execute load test.

2. Prepare the Test Configuration

Create the Artillery Test File and Run the test

- The YAML configuration file is created that is traffic-test.yml to simulate virtual users which sends request that is 20 requests per second.
- The key parameters are target, phases, arrivalRate and scenarios.
- The file is transferred to EC2 instance with the help of scp.
- Later the test file is executed.
- The live metrics are displayed by artillery during test. It includes HTTP response codes,request rates,latency metrics and Virtual user completion rates.

3. Analyze the Result

When the test is concluded the artillery generated the summary report for the Case 1.The key metrics are :

1. HTTP 200 Responses: 6000.
2. Total Requests Sent: 6000.
3. Request Rate: 20 requests/second.
4. Response Times:
5. Min: 1 ms
6. Max: 42 ms
7. Mean: 2.5 ms
8. Median: 2 ms
9. 95th Percentile: 3 ms
10. 99th Percentile: 21.1 ms
11. No Failures: 0 virtual user (VU) failures.
12. Downloaded Bytes: 327,000 bytes

- The application performed well under the test load that is 20 request per second.
- HTTP 200 response is send by all request.

- The response time is low with minimum average of 2.5 ms to a maximum 42 ms.

4. Insights and Conclusion

- The simulated load is efficiently handled without causing any errors or degradation in the performance.
- For a production grade application the latency metrics was within the limits.
- To demonstrate backend is strong none of the virtual users failed.
- Therefore in case 1 it is understood that with constant performance and reliability the application is able to handle the request which is 20 request per second.
- Hence Case 1 established a baseline border for Case 2 to evaluate the stress handling and scalability.
- In case 2 will increasing the request rate or the duration to evaluate the stress point or bottleneck.

5.2 Case 2

The main objective of the case 2 is to implement the hybrid GWO-PSO algorithm for effective load balancing across three EC2 instances . It also includes error handling, health checkups, caching , optimization of techniques and parallel metrics collection.

1. Infrastructure Setup and Architecture

- Three EC2 instances are launched which is of type t2 micro and Operating system Amazon Linux 2023.
- The system is updated to latest version and Apache Web server is installed.

2. Application Load Balancer

- The type is Application Load Balancer with the scheme internet-facing therefore it receive incoming traffic from user/client through web.
- The standard port web that is HTTP on port 80 is set .Hence it can check for incoming traffic request.

3. Target Groups

- The EC2 instances is attached which acts as backend server for hosting application.
- The health check is also configured to ensure healthy instance receive the traffic.

4. Lambda Function

- To perform the load balancing decision dynamically the Lambda function is created in Python 3.9.
- For easy communication it is configured with same VPC as for EC2 instances
- The Lambda function granted permission for following services below.They are :

1. CloudWatch Metrics Access
2. EC2 Describe Instances
3. CloudWatch Logs.
4. VPC Network Access

5. GWO-PSO Algorithm Implementation - The algorithm parameters are the following:

1. Population size: 3.
2. Maximum iterations: 5.
3. Inertia weight: 0.4.
4. Learning coefficients (C1, C2): 1.5.
5. Alpha coefficient: 1.5.

- The fitness function is implemented to analyse the EC2 instance based on the CPU utlization , response time and throughput.

6. Optimization Logic

- For the population the position and velocities are initialized and also for global best calculated initial fitness value.
- Combined PSO for exploitation along with GWO which is for exploration.Based on performance metrics iteratively refined weight.And also for instance selection ensured normalized weight.

7. Performance Optimization

- Caching is performed where private IP address of EC2 instance is stored temporary and also speeding up the request by avoiding repeated queries.
- From the optimization algorithm the result is cached for 5 seconds.
- For faster performance AWS boto3 client was configured that is to reduce timeouts.
- The metrics is collected from multiple EC2 instances in parallel manner so that time can be saved more so for that thread-based executor is used so data can be gathered in very fast way.

8. Error Handling and Fail over

- A health check point is implemented to verify the health status of the application. If there is any issue or is it running properly.
- The fallback mechanism is also implemented where if one instance fails it moves to another instance to ensure uninterrupted service.

9. Monitoring and Metrics Collection

- The CloudWatch is used to monitor the live metrics of EC2 instances.The metrics include CPU utilization,response time , throughput and execution duration.

10.Testing and Results - To ensure functionality and performance the entire system is tested.To check application is healthy , the endpoint is tested. The /lambda endpoint is also verified to the check load balancer ability to select the best EC2 instance.

6 Evaluation

The comparison of case 1 and case 2 helps to understand how the case 2 scenario performs better than case 1 scenario.The metrics considered for comparison are Execution time, throughput, Response time, Scalability and Error rate. Execution is the time taken to process and route the each request. Throughput is the number of successful request per second. On other hand response time tracks the time from the response is sent to the response is received.Scalability measures how the system handle the sudden spike in the traffic and the last one is error ate which helps to analyze the percentage of the failure based on varying loads.

6.1 Result of Case 1 Experiment

Based on the case 1 scenario when it comes to the execution time it is 15 ms.The execution time received is consistent using the default algorithm that is static round robin mechanism. No additional optimization was involved in the case 1 scenario.The value of steady traffic is 20 request per second and the value of spiked traffic is 12 request per second. In case 1 when there is high spikes of traffic the system showed some degradation while during normal traffic it showed stability.The average response time received for case 1 is 16 ms.During high load scenario it increase but when it was from low to medium the response time was low. The resource bottle neck is shown when there is sudden traffic spike and the system struggles to handle the traffic.In case 1 scenario the incoming traffic is distributed using default algorithm without considering resource utilization which causes ineffective load balancing.The error was up to 8 percentage it showed more error rate when there was spiked traffic incoming.

6.2 Result of Case 2 Experiment

In the case 2 the average execution time is 23 ms.The optimization is obtained with the help of hybrid GWO-PSO algorithm. The execution time higher in this case but he load distribution is effective and improved the performance of the system.The steady traffic value obtained for case 2 is 39 request per second and high incoming traffic that is 40 request/second , the system maintained stable . From this study it can be understood the efficiency of GWO-PSO algorithm.The response time received for case 2 is 16.7 ms which means that it is consistent and it depends on real metric such as CPU utilization. The factor automatic scaling and to accommodate the incoming traffic with AWS Lambda is very important.The hybrid GWO -PSO algorithm helped in the load distribution by considering the resource utilization even under extreme load condition.The error rate in the case 2 is very less than 1 percentage during high incoming traffic because of hybrid GWO-PSO algorithm reroute the traffic to instances based on the metrics.

6.3 Comparative Analysis

On comparing the Case 1 and Case 2 the main metrics considered are execution time, throughput, response time, scalability and error rate as shown in the tabular column.The case 2 performed well when it comes to throughput because of it's performance with the help of hybrid GWO-PSO algorithm even during high traffic incoming. The response time is same in both the cases but in case 2 performed more well when there was load

Metric	Case 1 (VM-Based)	Case 2 (AWS Lambda GWO-PSO)
Execution Time	15ms	23ms
Throughput	20 req/sec (steady)	39 req/sec (steady)
Response Time	16ms	16.7ms
Scalability	Limited	Excellent
Error Rate	8% during spikes	1% during spikes

Table 2: Performance Metrics Comparison

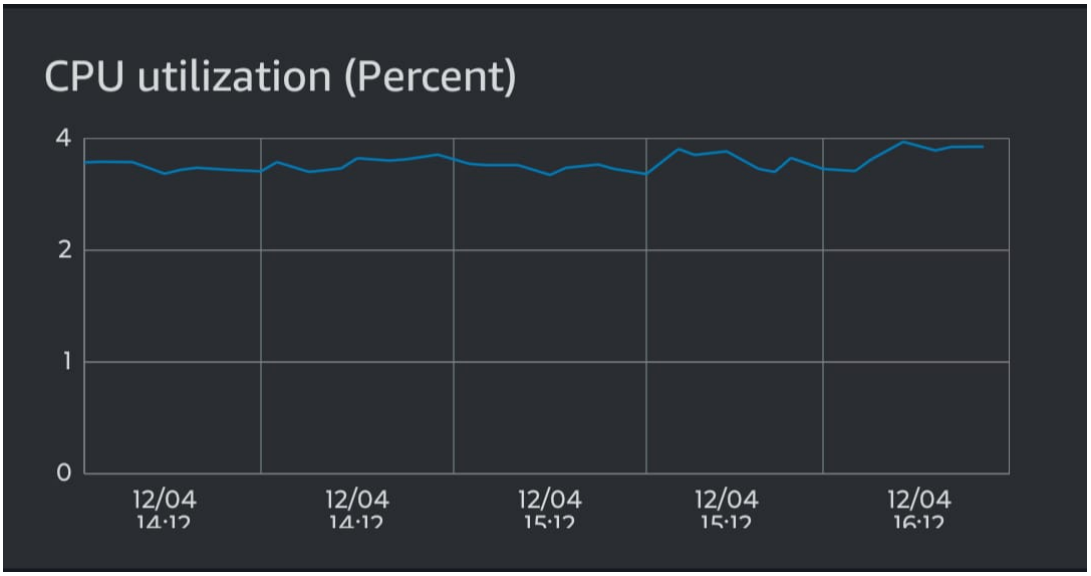

Figure 2: CPU utilization of one instance in Case 1

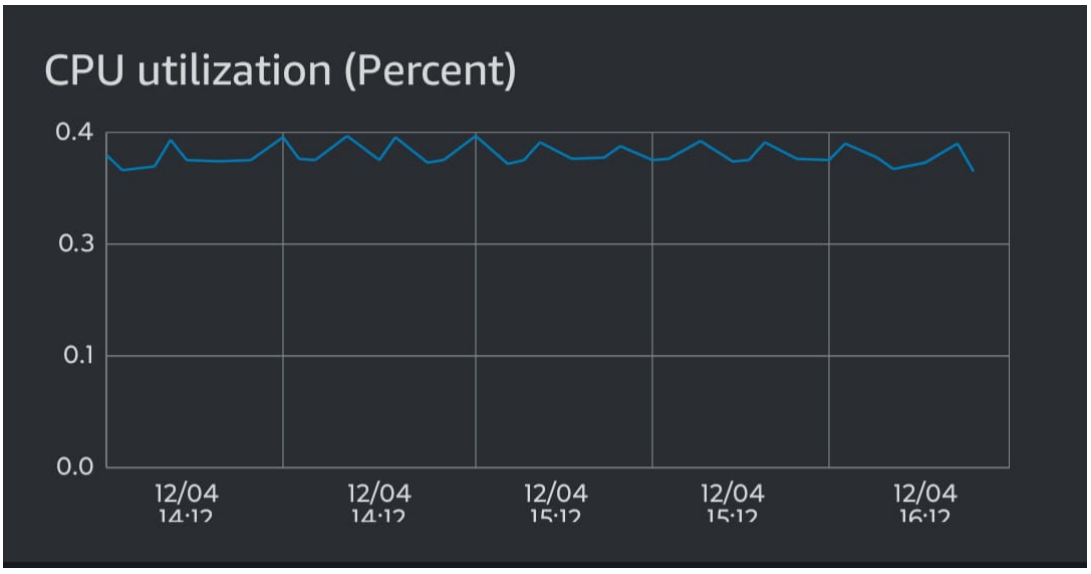

Figure 3: CPU utilization of one instance in Case 2

variation. The default or static algorithm used in case 1 lacks scalability but in case 2 scenario scalability is handled efficiently even during high traffic incoming.The error rate in case 2 is very less because of effective distribution of load across the various instance . But in case 1 the error rate was high with the use of default algorithm.

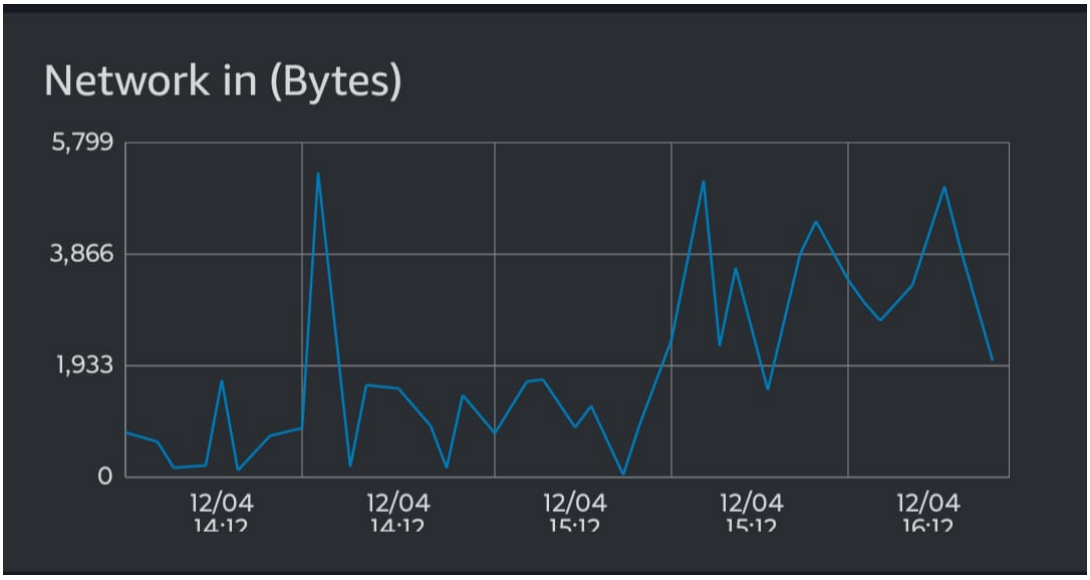

Figure 4: Ntwork in of Case 1

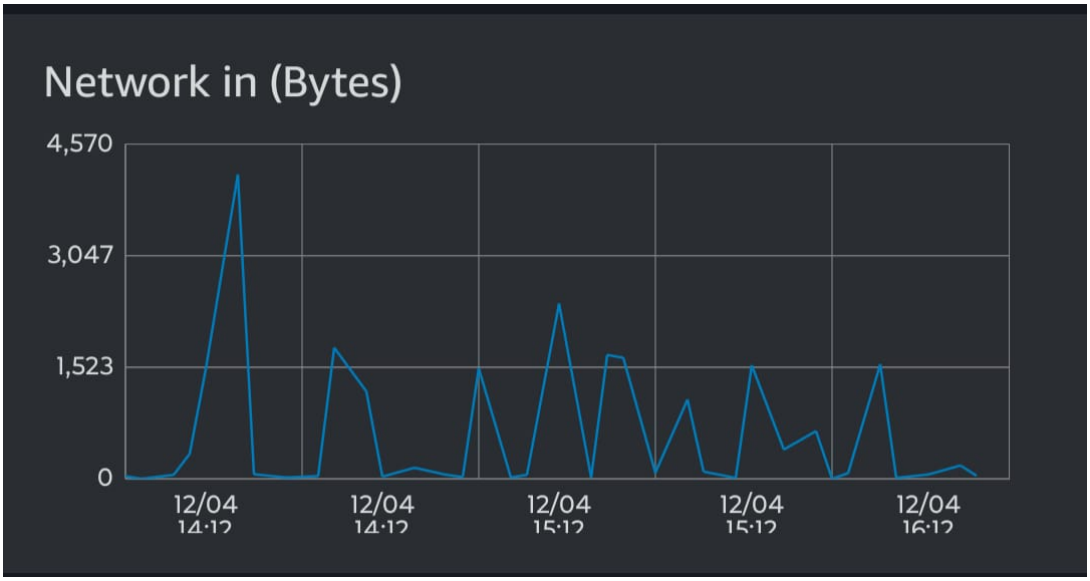

Figure 5: Network in of Case 2

7 Conclusion and Future Work

In the comparative analysis of two case that is traditional VM based approach along with hybrid GWO-PSO algorithm on Lambda various insights have been revealed.In traditional VM based the performance excelled due to it's simplicity and in lower execution time static traffic distribution is performed but it had limitation when it came to adaptability and scalability during high traffic load. On other hand in case 2 when hybrid GWO-PSO algorithm is implemented on AWS lambda it showed scalability,adaptability and high throughput even during high spikes of traffic.
Therefore from Case 2 it is understood that AWS Lambda helped in auto scaling which helped to improve the performance in case 2.For dynamic optimization of resource allocation and also by increased throughput with the help of hybrid GWO-PSO algorithm.There were less error in case 2 compared to case 1 because live metrics is used for resource util-

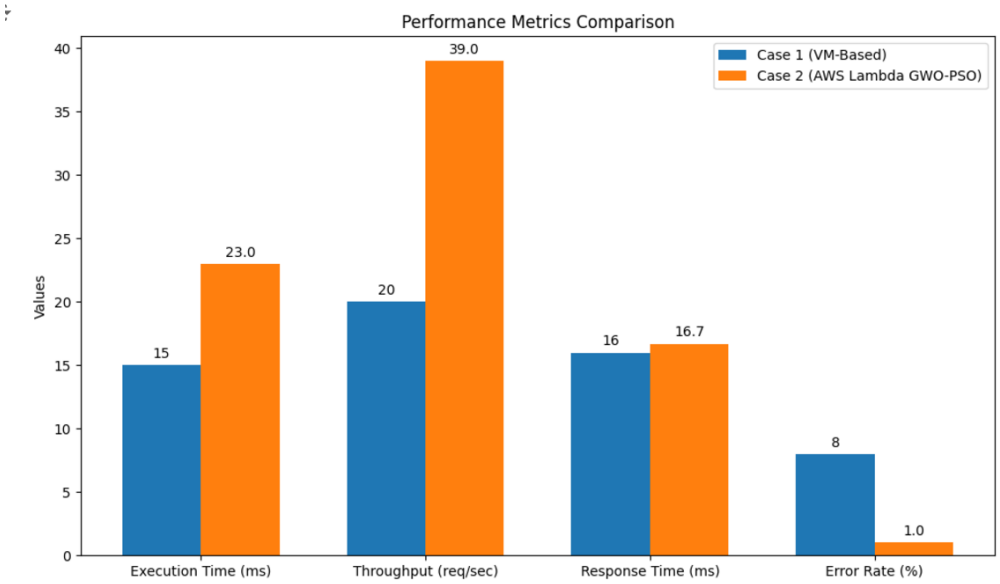

Figure 6: Comparison of two scenarios

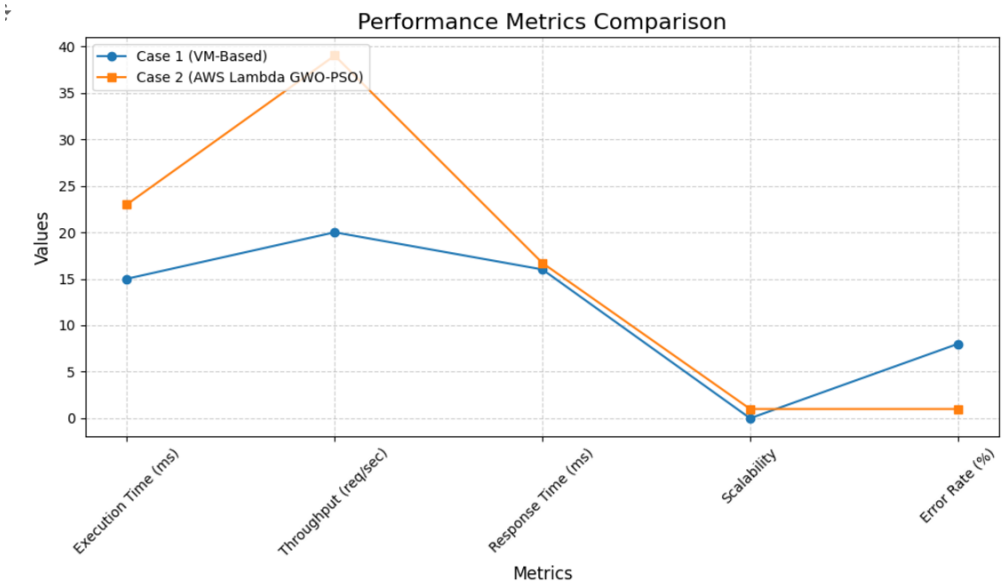

Figure 7: Comparison of metrics

ization in case 2. But drawback is the cost of higher execution time due to computation. The optimization of execution time can be considered as one the future work where the computation overhead need to be reduced by the hybrid algorithm.Another future work is the integration of advanced metrics for evaluation such network bandwidth and memory utilization for effective optimization.Expanding to multi cloud platform such Azure and Google cloud would be another scope for future work.Testing the system in different diverse scenario such as peak sales session is a future scope of work and also detailed cost analysis is also required to analyze financial implication of using AWS Lambda under high load traffic variation.

References

Al Reshan, M. S., Syed, D., Islam, N., Shaikh, A., Hamdi, M., Elmagzoub, M. A., Muhammad, G. and Talpur, K. H. (2023). A fast converging and globally optimized approach for load balancing in cloud computing, *IEEE Access* **11**: 11390–11404.

Chandrashekar, C., Krishnadoss, P., Kedalu Poornachary, V., Ananthakrishnan, B. and Rangasamy, K. (2023). Hwacoa scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing, *Applied Sciences* **13**(6): 3433.

Hai, T., Zhou, J., Jawawi, D., Wang, D., Oduah, U., Biamba, C. and Jain, S. K. (2023). Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes, *Journal of Cloud Computing* **12**(1): 15.

Khan, A. R. (2024). Dynamic load balancing in cloud computing: Optimized rl-based clustering with multi-objective optimized task scheduling, *Processes* **12**(3): 519.

Lester, J. N., Cho, Y. and Lochmiller, C. R. (2020). Learning to do qualitative data analysis: A starting point, *Human resource development review* **19**(1): 94–106.

Liu, Q. and Wang, L. (2021). t-test and anova for data with ceiling and/or floor effects, *Behavior Research Methods* **53**(1): 264–277.

Makhija, D., Sudhakar, C., Reddy, P. B. and Kumari, V. (2022). Workflow scheduling in cloud computing environment by combining particle swarm optimization and grey wolf optimization, *Comput. Sci. Eng. Int. J* **12**: 1–10.

Muller, L., Chrysoulas, C., Pitropakis, N. and Barclay, P. J. (2020). A traffic analysis on serverless computing based on the example of a file upload stream on aws lambda, *Big Data and Cognitive Computing* **4**(4): 38.

Patra, M. K., Misra, S., Sahoo, B. and Turuk, A. K. (2022). Gwo-based simulated annealing approach for load balancing in cloud for hosting container as a service, *Applied Sciences* **12**(21): 11115.

Pelle, I., Czentye, J., Dóka, J., Kern, A., Gerő, B. P. and Sonkoly, B. (2020). Operating latency sensitive applications on public serverless edge cloud platforms, *IEEE Internet of Things Journal* **8**(10): 7954–7972.

Shao, K., Fu, H. and Wang, B. (2023). An efficient combination of genetic algorithm and particle swarm optimization for scheduling data-intensive tasks in heterogeneous cloud computing, *Electronics* **12**(16): 3450.

Yu, H., Shen, Z., Leung, C., Miao, C. and Lesser, V. R. (2013). A survey of multi-agent trust management systems, *IEEE Access* **1**: 35–50.

Yue, Z., Zhang, S. and Xiao, W. (2020). A novel hybrid algorithm based on grey wolf optimizer and fireworks algorithm, *Sensors* **20**(7): 2147.

Zhan, X., Shoaib, M. and Reda, S. (2016). Creating soft heterogeneity in clusters through firmware re-configuration, *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, IEEE, pp. 540–549.