Adaptive Compilation Framework for Heterogeneous Computing Environments

A CAPSTONE PROJECT REPORT

Course Code: CSA1471

Course Faculty: S.Sankar

Slot: A

Date of Submission:17/06/2024

Submitted by

B. Venu Gopal [192210065]

Syed Umar Farooq Basha [192211886]

R. Varun [192210065]

Guide By

S.Sankar

ABSTRACT:

A crucial work scheduling issue that directly affects system performance and customer satisfaction is faced by a technology that offers flexible and scalable computing resources. Finding solutions to the NP-completeness of the task scheduling issue is challenging. Researchers have proposed a hybrid approach that combines the Grey Wolf Optimization approach (GWO) with the Genetic Algorithm (GA) in order to solve this. The hybrid GWO-GA method seeks to minimize costs, energy usage, and makespan when it comes to multi-objective task scheduling in cloud computing. By utilizing the crossover and mutation operators of the genetic Algorithm, the suggested method may be improved(Heinroth and Minker 2011). Moreover, the faster convergence of the GA-based GWO method is advantageous in big scheduling issues.The suggested algorithm's effectiveness in comparison to current techniques is demonstrated through evaluation utilizing the Cloud toolbox(Terzo et al. 2019). We have employed both artificial and real-world datasets. The statistical method of Analysis of Variance (ANOVA) is used to confirm the findings. The outcomes of the experiments demonstrate how well it works to reduce makespan, energy usage, and computational expense. In particular, the suggested algorithm performs better than the conventional GWO, GA, and PSO algorithms in terms of makespan, cost(Bell et al. 2022), and energy usage; in comparison to each technique, it achieves reductions of 19%, 21%, and 15%, respectively. Furthermore, it results in energy savings of 17%, 19%, and 23% in comparison to GWO, GA, and PSO, respectively, and a 13%, 17%, and 22% decrease in overall scheduling expenses(De Bosschere 2012). The results indicate that the suggested approach is effective in resolving the issue of work scheduling inside environments(De Bosschere 2012; Schmorrow and Fidopiastis 2014).

INTRODUCTION

Customers can access IT resources through a unified collection of computer resources through cloud computing, an increasingly common phenomena that uses a consumption-based payment approach [26]. Cloud computing has advanced to a mature state in recent years and has blended in perfectly with contemporary internet technology(Grandinetti 2008). These days, it's more like utility computing, offering platforms, software, and infrastructure online on a pay-per-use basis.

High-performance computing (HPC) resources and other IT equipment are grouped together to construct cloud data centers (CDCs), which are prepared to meet customer needs. Prominent cloud service companies with many geographically dispersed data centers include Amazon and Google.Through the use of virtual technology, a big computer job may be divided into smaller ones over a network and then assigned to a system of several servers using different allocation techniques. The user receives the results after processing [1], [14], and [28].

CDCs are an example of extremely sophisticated, energy-intensive technology. To balance performance, management expenses, and service availability, effective management is essential. When used and set correctly, CDCs provide enormous processing capability at a low cost and energy consumption. Virtualization technology allows CDCs to share HPC services and resources with different users, but it also complicates resource management(Zhou and Chen 2017).

One of the main issues with effective resource management is task scheduling. As seen in Fig. 1, job scheduling in a cloud system can be divided into many parts. It is one of the NP-class problems that entails distributing incoming jobs to available computing resources [36]. Because of this, conventional scheduling techniques don't work well in these kinds of settings and don't offer adequate efficiency [62]. Heuristic and metaheuristic techniques are frequently used to produce scheduling solutions in order to address this difficulty. These approaches optimize a variety of objectives, including makespan and resource consumption.

Depending on the specific needs of each issue, the GWO and GA algorithms may be used to a variety of optimization problems to increase the accuracy and effectiveness of the optimization process. Each approach has its own benefits. Thus, in this research, we construct a method for cloud computing job scheduling by combining both GA and GWO. By including the crossover and mutation processes from GA, the suggested algorithm improves upon the typical GWO. Through this integration, the problem of premature convergence is resolved, and the conventional GWO's solution exploitation is enhanced. This makes the search process more effective in finding the best answers and locating the wolves in new places between iterations(usenix International Middleware et al. 2004).

Furthermore, by accelerating the exploration process, this hybridization helps find the optimum answer in an acceptable amount of time. The primary goal of this research is to apply both GA and GWO to the work scheduling problem in cloud computing settings. Here are the differences between our suggested strategy and earlier scheduling techniques:

Furthermore, by accelerating the exploration process, this hybridization helps find the optimum answer in an acceptable amount of time. The primary goal of this research is to apply both GA and GWO to the work scheduling problem in cloud computing settings. Here are the differences between our suggested strategy and earlier scheduling techniques:

-> In order to improve job scheduling in cloud computing, the proposal presents a unique hybrid approach inspired by nature: the Grey Wolf Optimization algorithm and the Genetic Algorithm.

->The suggested algorithm's application seeks to overcome the shortcomings of the conventional GWO, including its early convergence and insufficient solution exploitation.

->The suggested method combines a modified GWO algorithm with GA. To our knowledge, relatively few attempts have been made to create job scheduling algorithms for environments, and the majority of research has concentrated on traditional metrics like makespan.

->The suggested method distributes jobs to physical machines in a way that maximizes data center efficiency by concurrently accounting for critical factors including makespan, energy usage, and computational cost.

In conclusion, the suggested Grey Wolf Optimization method, which is based on Genetic Algorithms, skillfully improves its exploration and exploitation skills while successfully avoiding local optima. Extensive tests were carried out to verify its effectiveness in solving the task scheduling issue, utilizing actual and synthetic datasets to compare it with current state-of-the-art algorithms. The suggested approach outperforms these cutting-edge algorithms in terms of makespan, energy usage, and cost, or at the very least, performs comparably.

This is how the remainder of the paper is structured: A summary of earlier research on task scheduling in cloud computing is provided in Section 2, and an evaluation of the suggested

approach is given in Section 3. The suggested method for job allocation in physical machines is described in detail in Section 4, and the experimental results and performance assessment are presented in Section 5. Finally, the study's results, next projects, and the projected work's limitations are outlined in Section 6.

Problem description and model description

Several models are included with the suggested algorithm in this part to help with understanding. Here, we offer models for mathematical optimization. In addition, we include a nomenclature table (Table 3) to facilitate readers' comprehension and make the text simpler to read.

Proposed Design:

Requirements Gathering and Analysis: Conduct stakeholder interviews and surveys to learn about the organization's network infrastructure, security objectives, and specific testing requirements

Tool Selection Criteria: Create a comprehensive list of automated network security testing tools using industry research, peer evaluations, and expert suggestions

Scanning and Testing Methodology: Specify target environments, network segments, and test scenarios to imitate real-world conditions and thoroughly evaluate tool effectiveness.

Functionality:

User Authentication and Role-Based Access Control:

Part of the Framework for Adaptive Compilation: User authentication plays a crucial role in ensuring that only authorized users may access and alter compilation settings, code optimizations, and other important components of an adaptive compilation framework designed for heterogeneous computing.

Execution: Multi-factor authentication, biometric authentication, and username-password combinations are examples of common authentication techniques. The decision is based on the system's security requirements.

RBAC, or role-based access control:

Role in Adaptive Compilation Framework: Within the adaptive compilation framework, RBAC is critical for controlling access to various capabilities and resources. It guarantees that users only have access to the configurations and tools necessary for their respective responsibilities.

Application: Roles such as system administrator, compiler developer, or end-user may be specified in an adaptive compilation framework according to their respective duties. Their ability to alter compilation techniques, obtain performance profiling data, or manage hardware resources would depend on the permissions assigned to each role.

Framework for Adaptive Compilation in Heterogeneous Computing Environments:

Security Considerations: It is essential to manage access to the framework because of the important role of the compilation process in heterogeneous computing. Unauthorised access may affect the security and performance of programs operating on heterogeneous architectures by allowing malicious alterations to code optimizations or jeopardising the integrity of the compilation process.

Tool Inventory and Management:

1. Programming Models and Languages:

2. OpenCL: A framework for writing programs that execute across heterogeneous platforms, including CPUs, GPUs, and other accelerators.

CUDA: Developed by NVIDIA, it is a parallel computing platform and programming model for their GPUs.

SYCL: A standard developed by the Khronos Group for single-source programming of OpenCL.

2. Compiler Tools:

LLVM (Low-Level Virtual Machine): An open-source compiler infrastructure that supports multiple programming languages and is often used for building custom compilers.

ROSE Compiler Framework:An open-source infrastructure for building source-to-source program transformation and analysis tools.

3. Performance Analysis and Profiling Tools:

AMD CodeXL, NVIDIA Nsight:Tools for profiling and debugging GPU applications.

Intel VTune Profiler: Provides insights into the performance bottlenecks in the code.

4. Adaptive Compilation Frameworks:

Polly: An open-source polyhedral optimizer for LLVM.

Halide: A domain-specific language for image processing with an optimizing compiler.

5. Resource Management:

Slurm, Torque, and Maui:Job scheduling and resource management systems that can be used in heterogeneous environments.

6. Version Control and Build Systems:

Git, SVN: Version control systems for managing source code.

CMake, Make, and Autoconf:Build systems for configuring and building software projects.

7. Monitoring and Visualization:

Grafana, Kibana:Tools for monitoring and visualizing system performance.

Paraver: A tool for visualizing and analyzing the performance of parallel and distributed applications.

8. Containerization and Orchestration:

Docker, Kubernetes:Tools for containerizing applications and orchestrating their deployment in heterogeneous environments.

Security and compliance control:

Access Control: Use strong authentication procedures and role-based access control (RBAC) to limit access to the framework to only authorized users.

Data Protection: - Encrypt sensitive data while it's in transit and at rest.

Audit Logging: - Enable thorough logging of activities within the framework for auditing purposes.

Compliance with Standards: - Align the framework with pertinent industry standards and regulations, such as ISO 27001, NIST, or other applicable standards in your region.

safe Coding Procedures:To reduce vulnerabilities in the codebase, use secure coding methods.Update and patch the framework often to fix known security flaws.

Security of Networks:To safeguard communication between components in a diverse environment, use network security mechanisms.Install intrusion detection/prevention systems and firewalls as necessary.

Isolation of Resources:Put safeguards in place to keep computing resources isolated and safe from outside intervention or access.

-> Verification and Permission:Make advantage of robust authentication procedures to confirm users' and components' identities.To manage access to various framework features, use fine-grained authorization.

-> Documentation of Compliance:Keep records that show adherence to pertinent requirements and describe the security measures in place.

-> Safe Interaction:Make sure that technologies like TLS (Transport Layer Security) are used to safeguard the communication routes between various components.

Architectural Design:

Application layer:

Interfaces that let developers take advantage of the features of a heterogeneous computing environment are provided by APIs and libraries.

Domain-Specific Language Support: Enables programmers to define calculations in a manner compatible with the hardware specifications of the intended application.

Monitoring and Management Layer: Data on resource usage, execution time, and other performance parameters are gathered through performance monitoring.

Profiling Tools: Help programmers comprehend how their programs behave and make the most of them.

UI Design:

Dashboard:

include building a dashboard that gives users an understandable summary of the system's functionality, compilation procedures, and other pertinent data.

User management:

Confirm the users' identities who are logging into the system.

Based on the user's credentials, assign responsibilities and permissions that are suitable.

Limit who has access to what features and resources within the framework.

Help and Support:

Include tooltips that offer succinct, context-specific information about different UI components.

Make sure that user activities, including lingering over buttons or icons, cause tooltips to appear and offer real-time assistance.

Feasible Element Used:

Dashboard:

Tiles/cards that show summary information about the network security testing tool, such as the number of scans performed, vulnerabilities found, and system condition.

User Management:

Permit users within the compilation framework to alter their preferences and profiles.

Give users the ability to select and store particular compilation parameters.

Help and Support:

Include tooltips that offer succinct, context-specific information about different UI components.

Make sure that user activities, including lingering over buttons or icons, cause tooltips to appear and offer real-time assistance.

Real-time Monitoring:

Include systems for monitoring resource use in real-time across various computing devices (e.g., CPUs, GPUs, accelerators).

To help you decide how best to distribute jobs during compilation, gather information on CPU and GPU use, memory utilization, and other pertinent metrics.

Collaboration Features:

Incorporate real-time collaboration functionalities to facilitate the simultaneous work of several people on a same project.

Make sure that modifications made by one user are immediately reflected for other users to enable smooth collaboration.

Trend Analysis:

Examine patterns in the dynamic placement of components in environments with heterogeneous computing.

Examine how real-time element placement optimization is achieved via adaptive compilation frameworks, which can adjust to changing computer resources. Analyze patterns in the application of machine learning methods to forecast ideal element placements using past performance information.

Project Time Line

Day 1: Research and Understanding

- Study the existing literature on adaptive compilation frameworks and heterogeneous computing environments.
- Understand the requirements of the project, including the need to investigate relations between performance degradation of VMs according to their own loads and their neighbors competing for shared resources on PMs.
- Identify the tools and benchmarks provided for monitoring and testing.

Day 2: Data Collection

- Collect raw data on the performance degradation of VMs and their neighbors.
- Use the provided tools to monitor and collect data on the shared resources on PMs.
- Organize and preprocess the collected data for modeling.

Day 3: Modeling and Analysis

- Develop a model to represent the complex ecology of the heterogeneous computing environment.
- Analyze the collected data using the developed model to identify patterns and relationships between VM performance and resource utilization.
- Investigate the effectiveness of existing technologies for achieving application-level isolation in cloud computing infrastructure providers.

Day 4: Algorithm Development

- Develop new algorithms for energy-aware resource management allocation for large-scale distributed systems.
- Implement the algorithms using a programming language such as Python or C++.
- Test and evaluate the performance of the developed algorithms

Day 5: Implementation and Testing

- Implement the adaptive compilation framework using the developed algorithms.

- Test the framework using the provided benchmarks and tools.
- Evaluate the performance and energy efficiency of the framework.

Conclusion:

A meta-heuristic approach applied to a scheduling problem is typically analogous to deploying a robot in a vast maze. As intelligence impacts an algorithm's efficacy, so too does the robot's capacity for exploration and decision-making determine how quickly it can reach the end of the maze. But occasionally—especially when working with intricate and wide-ranging issues—the ideal answer is not discovered. There must be equilibrium in order to improve the calibre of scheduling solutions.

REFERENCES

Bell, Steven, Jing Pu, James Hegarty, and Mark Horowitz. 2022. *Compiling Algorithms for Heterogeneous Systems*. Springer Nature.

De Bosschere, K. 2012. *Applications, Tools and Techniques on the Road to Exascale Computing*. IOS Press.

Grandinetti, Lucio. 2008. *High Performance Computing and Grids in Action*. IOS Press.

Heinroth, Tobias, and Wolfgang Minker. 2011. *Next Generation Intelligent Environments: Ambient Adaptive Systems*. Springer Science & Business Media.

Schmorrow, Dylan D., and Cali M. Fidopiastis. 2014. *Foundations of Augmented Cognition. Advancing Human Performance and Decision-Making through Adaptive Systems: 8th International Conference, AC 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings*. Springer.

Terzo, Olivier, Karim Djemame, Alberto Scionti, and Clara Pezuela. 2019. *Heterogeneous Computing Architectures: Challenges and Vision*. CRC Press.

usenix International Middleware, Acm, ACM/IFIP/USENIX International Middleware Conference, and ifip. 2004. *Middleware 2004: ACM/IFIP/USENIX International Middleware Conference, Toronto, Canada, October 18-20, 2004, Proceedings*. Springer Science & Business Media.

Zhou, Yu, and Taolue Chen. 2017. *Software Adaptation in an Open Environment: A Software Architecture Perspective*. CRC Press.