Literature Survey: Cloud-Based Integrated Development Environment Using Docker, Machine Learning, and Reinforcement Learning

1. Evolution of Cloud-Based Integrated Development Environments (IDEs)

Cloud IDEs represent a shift from traditional desktop-based development tools to a more scalable, accessible, and collaborative environment. These environments utilize cloud computing resources, enabling developers to access their development environment from anywhere, reducing the need for high-end local machines.

Key Findings:

- Initial Research: Early research on cloud-based development environments, such as in Buse and Zimmermann (2012), emphasized cloud IDEs' ability to address the challenges of resource constraints and collaboration in distributed teams.

- Architecture Design: Research such as 'Cloud-based Development Environments: Benefits, Risks, and Challenges' (IEEE Access, 2018) analyzed architectural models, focusing on how cloud resources are allocated to optimize developer experiences, including IDEs using virtualized environments (VMs).

- Scalability: Later works, such as 'Cloud-based Integrated Development Environments: Enhancing Productivity and Scalability' (IEEE Transactions on Cloud Computing, 2019), shift focus towards lightweight environments using containerization technologies like Docker.

Challenges Identified:

- Latency: Early systems faced latency in real-time collaboration and compilation, an issue that continues to drive research in network optimization.

- Security: Since developers work in shared cloud environments, data integrity, and privacy in cloud IDEs became a focal point for subsequent research.

Research Gaps:

- Development Workflow Optimization: There is still a need to explore how intelligent agents could optimize cloud IDE workflows, reducing bottlenecks such as compilation wait times or resource contention.

- Low-code/No-code Platforms: The trend of low-code platforms as part of cloud IDEs has not been extensively explored in the context of high-performance computing development.

2. Containerization with Docker in Cloud IDEs

Docker, with its ability to create isolated, reproducible, and lightweight containers, has become a cornerstone of modern cloud IDEs. The transition from VMs to containers was a key innovation in enhancing the performance and scalability of cloud IDEs.

Deep Dive into Docker's Contribution:

- 'Docker: Lightweight Linux Containers for Consistent Development and Deployment' (IEEE Cloud Computing, 2017) introduces Docker as a tool that allows developers to create environments that remain consistent across various cloud instances, enabling IDEs to deliver a reproducible experience across users.

- Isolation and Efficiency: Research like 'Improving Software Development Productivity with Containers' (IEEE Transactions on Cloud Computing, 2019) points out that Docker reduces the overhead typically associated with VMs while enhancing process isolation, which is crucial for environments where multiple developers share resources.

Advanced Container Management:

- Networking and Scalability: Using container orchestration tools such as Kubernetes has been a major step forward. 'A Kubernetes-based Cloud Development Environment: Architecture and Performance' (IEEE Cloud Computing, 2020) explores how Kubernetes automates container management across clusters, addressing auto-scaling, resource distribution, and load balancing, thus enhancing IDE performance in large teams.

Research Gaps:

- Container Security: While Docker containers are more lightweight than VMs, they pose unique security risks due to shared kernel architecture. There is a growing need for more sophisticated security mechanisms in containerized cloud IDEs, as discussed in 'Container Security in Multi-tenant Cloud Environments' (IEEE Access, 2020).

- Storage Optimization: How cloud-based IDEs manage persistent data storage in containers remains an open research challenge, particularly for large-scale or stateful applications.

3. Machine Learning Integration in Cloud IDEs

Machine Learning (ML) integration in cloud environments, especially cloud IDEs, opens avenues for intelligent features like automated error detection, smart resource allocation, predictive scaling, and enhanced developer assistance tools.

Intelligent Resource Allocation:

- 'Machine Learning for Cloud Resource Management: A Comprehensive Survey' (IEEE Transactions on Cloud Computing, 2021) provides a detailed overview of ML models that manage resources in cloud infrastructures. Key models like Support Vector Machines (SVMs), Decision Trees, and Neural Networks are discussed for predicting resource usage and adjusting allocation.

- Application to Cloud IDEs: In 'Optimizing Cloud Resource Allocation using Machine Learning' (IEEE Access, 2020), researchers propose ML algorithms for dynamically scaling development environments, which could optimize cost-efficiency and performance in multi-tenant cloud IDE infrastructures.

Error Detection and Workflow Automation:

- Intelligent IDEs like AWS Cloud9 or Google Cloud Shell have started to integrate ML-driven error detection and code recommendation features, a growing trend emphasized in 'AI-assisted Cloud-Based IDEs: Towards Smarter Development Environments' (IEEE Software, 2020).

- Auto-completion and Code Refactoring: ML models that can understand and predict developer code-writing patterns are covered extensively in 'Machine Learning in Software Development: Automating Developer Tasks' (IEEE Software, 2021). This area is seeing rapid advancements, especially with models like GPT-3 and Codex.

Research Gaps:

- Developer Autonomy: As ML models take on more roles in cloud IDEs, maintaining a balance between automation and developer control remains an open issue.

- Bias and Fairness: There is limited research on ensuring that ML-driven code recommendations do not propagate bias, particularly when using models trained on large, unfiltered code datasets.

4. Reinforcement Learning (RL) for Smart Resource Allocation and Auto-scaling

Reinforcement learning (RL) algorithms have demonstrated great potential in managing cloud resources, particularly through intelligent scaling and dynamic allocation.

RL-based Auto-Scaling:

- 'Reinforcement Learning for Dynamic Resource Management in Cloud Computing: A Review' (IEEE Access, 2019) highlights how RL, especially policy-gradient methods and Q-learning, is used to make real-time resource management decisions. RL has outperformed traditional rule-based scaling mechanisms by learning optimal actions over time.

- Auto-scaling in IDEs: The application of RL in auto-scaling containerized environments is further developed in 'Auto-Scaling in Cloud Computing Environments: A Reinforcement Learning Approach' (IEEE Cloud Computing, 2022). This work shows how RL models, when integrated with orchestration platforms like Kubernetes, can autonomously scale developer environments based on real-time demand.

Upper Confidence Bound (UCB) Algorithms:

- UCB algorithms are popular in RL because of their ability to balance exploration (trying new actions) and exploitation (leveraging known actions). In 'Multi-armed Bandit Algorithms and Reinforcement Learning for Cloud Resource Management' (IEEE Transactions on Cloud Computing, 2021), UCB-based methods are applied to efficiently allocate resources across multiple cloud IDE tenants.

- Dynamic Resource Allocation: 'Application of Upper Confidence Bound Algorithms in Resource Allocation for Cloud-based Services' (IEEE Access, 2020) presents how UCB can minimize resource contention and optimize performance across shared environments.

Research Gaps:

- Convergence Speed: While RL models can effectively manage cloud resources, their convergence speed in dynamic environments is often slow. Faster learning algorithms or hybrid approaches combining RL and traditional optimization methods are an emerging area of study.

- Cost Efficiency: Maximizing cost efficiency in RL-based resource management remains a challenge, particularly for small cloud IDE deployments where over-provisioning can be financially unsustainable.

5. Open Challenges and Future Directions

Security:

- 'Security Challenges in Cloud-based IDEs' (IEEE Access, 2020) stresses that multi-tenancy in cloud IDEs raises significant security concerns, particularly related to container security, privilege escalation, and data leakage.

Interoperability:

- Ensuring interoperability between different cloud platforms remains a challenge. Research such as 'Interoperability in Cloud Computing: Challenges and Solutions' (IEEE Cloud Computing, 2019) suggests the need for more standardized APIs and frameworks, particularly for cloud-based IDEs where development environments need to be portable across platforms.

TABLE 1. Autonomic Provisioning - A Comparative Survey

Reference	Objective Parameters	Evaluation Tool	Advantages	Limitations	Dataset Used
Ghobaei et al. [23]	CPU utilization, Response time	CloudSim	Decreased cost, Improves resource utilization	Limited to SaaS only	Clarknet and NASA
Tushar et al. [25]	Response Time, Finishing time, Average VM load, Rejection percentage	CloudSim	Effective Resource utilization, Proactive approach	Finite number of sources for scaling	Clarknet
Xu et al. [33]	CPU utilization,	Apache Hadoop	Reduction in resource cost	Limited workload	Testbed

	Resource usage, Cost and SLA constraint	(YARN)	through optimization, SLA confirmation	scenarios	
Zhang et al. [34]	CPU Utilization, Response time	Kubernetes	SLA violation reduction	Inaccuracy due to cold starts	NASA and FIFA
Orhean et al. [10]	Minimize the time required to execute tasks while ensuring SLAs	Work flowSim	Improves response time	Limited scalability	TestBed
Agarwal et al. [27]	Request arrival rate, SLA	Kubernetes	Reduced cold-start overhead	Efficiency of the training model	Azure traces (HTTP)
Rossi et al. [35]	CPU utilization, Response Time	Docker Swarm	Reduction of resource consumption, Improved training speed, Penalty rate	Simplicity of application models	Amazon EC2
Wang et al. [26]	To decrease service delays and lower migration costs.	MATLAB	Optimized performance	Lack of consideration on load balancing	Shanghai Telecom's dataset and Shanghai Taxi Track
Dezhabad et al. [29]	CPU Utilization, Response time, SLA	MATLAB	Reduces response time, Better resource utilization	Dimensionality problem	Calgary, ClarkNet, NASA and Saskatchewan
Arian et al. [30]	CPU utilization,	Kubernetes	Improved response	Dimensionality problem,	NodeCellar web

	SLA		time	Threshold-based approach	workload
Hu et al. [36]	Execution time, cost, Rental-Time	Simulation Engine	Effective resource utilization, Reduction in Cost, Based on deadline analysis	Not an autonomic predictive based approach	Alibaba Cloud
Ghobaei-Arani et al. [37]	Response time, Number of VMs, Cost	Cloudsim	Stronger prediction model, Reduces the cost for rental resources, Better response Time	Limited amount of resources	Synthetic, RuneScape workload
Etemadi et al. [38]	Cost, CPU utilization, network usage, Delay violation	iFogSim	Reduction in cost, delay violation, network usage, Increases CPU utilization	Resource placement problem	Synthetic, Real workload
Nazeri et al. [39]	Energy consumption, SLA, Response Time	CloudSim	Decreases energy consumption, response time	Availability and reliability not considered	NASA and Clarknet
Fan et al. [20]	Energy consumption, Delay	MATLAB	Task allocation effectively, extensive simulation performed	Decreases task offloading	Synthetic workload

Veira et al. [40]	Cost, SLA, Execution time	Cloudsigma	Increased scheduling cost	Trust and processing considered	Synthetic workload
Proposed Work	Avg VM Load, Response time, Rejection percentage, No of VMS	CloudSim	Reduced cost for VMs, Improved Response time	Resources are limited for a small interval of time	Clarknet, Google traces