International Journal of Enhanced Research in Management & Computer Applications,
ISSN: 2319-7471, Volume 15 Issue 6, June 2026

Cloud Platform Performance before AI Integration
A Comprehensive Study on Pre-AI System Efficiency and Scalability

Parveen[1], Dr. Bijender[2]

[1]Student, Department of Computer Science & Engineering, BMU, Rohtak
[2]Professor, Department of Computer Science & Engineering, BMU, Rohtak

ABSTRACT

The following literature review focuses on the performance aspects, problems, and approaches to optimization of cloud computing environments before the development and integration of AI and ML algorithms. The period considered in the review encompasses the time from the inception of cloud computing (around 2006) until the period preceding the AI optimization era (around 2006-2016). Specifically, the paper aims to identify the ways in which cloud computing environments deal with such issues as resource allocation, load balancing, fault tolerance, energy efficiency, and Quality of Service (QoS) through non-ML based computations. The literature review is based on more than 50 academic sources and provides insights into how cloud computing environments worked before they started using AI algorithms.

Keywords: Cloud Computing, Performance Management, Resource Allocation, Load Balancing, Virtualization, Quality of Service, Pre-AI Cloud Systems

INTRODUCTION

Background

The arrival of cloud computing has brought about a revolution within the information technology industry since its commercial adoption began in the mid-2000s. Cloud computing is defined by the National Institute of Standards and Technology (NIST) as "model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell and Grance, 2011). Prior to the incorporation of AI and ML techniques into cloud computing, algorithms and thresholds were used to maintain performance within cloud platforms.

It was around 2006 when AWS was launched and two years later, Microsoft Azure came out and a year after, Google Cloud was launched. In this early stage, cloud services encountered numerous problems in dealing with performance issues without intelligent learning systems. The solutions implemented during this time were useful but exposed their weaknesses that led to the eventual need for an AI-based system.

To comprehend how cloud computing operated prior to integrating AI is very important for various reasons. Firstly, it enables us to measure how cloud systems function with the incorporation of AI technologies. Secondly, it shows us the weaknesses of traditional systems which made the developers incorporate AI into their systems. Lastly, it helps us understand the core architecture behind cloud computing systems.

Objectives of the Review

Main objectives of the review:
(a) To explore the existing methods of performance management in cloud computing technologies without AI.
(b) To identify the performance management issues and constraints in cloud computing technology that have been addressed without AI.
(c) To build a comprehensive baseline comparison between cloud computing performance management methods prior to and after AI.

(d) To provide an evolutionary history of the cloud performance management method from 2006 until 2016.

Scope and Organization
The survey spans from about 2006 until 2016 when the concepts of AI and ML began to be used in cloud computing. The survey is divided into thirteen chapters, which discuss the following topics: architecture, resources management, load balancing, monitoring, fault tolerance, energy efficiency, security, SLA management, constraints, comparison, discussion, and conclusions.

EVOLUTION OF CLOUD COMPUTING ARCHITECTURE

From Grid Computing to Cloud Computing
Cloud computing has evolved from previously existing paradigms of distributed computing. Foster et al. (2008) outlined how grid computing eventually gave way to cloud computing, stating that while grid computing was concerned with managing the coordination of computational resources outside organizational boundaries, cloud computing brought commercial feasibility to distributed computing using pay-per-use schemes and on-demand resource allocation. Similarly, Buyya et al. (2009) described cloud computing as being built around market-driven cloud computing architectures that embraced the tenets of utility computing.

The primary difference between grid computing and cloud computing pertained to the mechanisms used to manage performance. Grid computing utilized batch scheduling and reservation, while cloud computing required the ability to dynamically provision on-demand resources.

Service Models and Deployment
Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) emerged as the three major service models during this time. According to Armbrust et al. (2010), this period saw a landmark study that identified the challenges and potentials of cloud computing with performance variations, data transfer issues, and scaling problems identified as key challenges. Similarly, Zhang et al. (2010) conducted a detailed literature survey on the architecture of cloud computing with performance being strongly related to virtualization techniques and the network architecture.

Each of the three service models offered distinct performance metrics. IaaS offered basic computing resources wherein performance depended on hardware allocation. The introduction of PaaS involved middleware layers resulting in increased latency; however, this facilitated easier application deployment. Performance in SaaS services relied upon the multi-tenant sharing and architecture of the application deployed.

Virtualization as the Foundation
The virtualization technology was the basis for the performance of cloud computing platforms. Barham et al. (2003) developed a hypervisor called Xen, which facilitated the co-residence of various virtual machines (VMs) on the same physical machine without affecting performance. They indicated that their tests showed a very small overhead ranging from 2% to 5%. Later on, Clark et al. (2005) demonstrated live migration of VMs, which turned out to be an important factor in performance management of cloud computing systems. The live migration made it possible to move workloads among different physical machines without stopping services, and hence facilitating re-balancing of loads and maintenance of hardware. Unfortunately, the process of migrating used network bandwidth and caused a drop in performance for a short while.

The work by Sotomayor et al. (2009) illustrated an approach for leasing resources based on virtual machines (VMs). In their experiments, they were able to demonstrate that using leases can result in better use of cloud resources (up to 15% to 30%).

TRADITIONAL RESOURCE MANAGEMENT AND ALLOCATION

Static Resource Allocation
The early approaches to cloud computing utilized mostly static approaches to resource allocations. One such static approach was suggested by Urgaonkar et al. (2008), who introduced an analytical model of resource allocation using queuing theory for multi-tier internet services. This particular approach depended heavily on offline profiling and capacity planning, and although quite accurate mathematically, it could not deal with changing workload dynamically.

Another static approach was presented by Gmach et al. (2007), who offered a technique for workload prediction in enterprise-level datacenters. In this technique, historical data about workload was used for pattern recognition and then

capacity was planned. The technique worked well when workload followed certain patterns that occurred cyclically (once per day, week etc.), but could not deal with unpredicted peaks in workload.

Threshold-Based Auto-Scaling
Prior to the advent of artificial intelligence, auto-scaling systems used rule-based threshold logic predominantly. AWS launched its Auto Scaling feature back in 2009 that enabled users to set up threshold-based rules to determine whether the application should be scaled in or out. For instance, one could decide that CPU usage over a certain threshold is a trigger point for scaling out while CPU usage under another threshold triggers scaling in.

Lorido-Botran et al. (2014) reviewed the available literature related to auto-scaling techniques, classifying them as reactive and predictive approaches. As far as reactive approaches go, the main shortcomings were oscillations due to fluctuations in metrics and time delays caused by several minutes needed to apply the change in resource allocation. Predictive solutions used such time series methods as ARIMA and exponential smoothing with about 70-80% prediction accuracy, but they did not work for irregular patterns.

The fundamental limitation of threshold-based scaling was the need to manually specify thresholds. Setting thresholds too low resulted in premature scaling (wasting resources), while setting them too high caused delayed responses (degrading performance). No single threshold configuration worked optimally across varying workload conditions.

Heuristic-Based Resource Scheduling
Scheduling algorithms were widely employed in pre-AI cloud environments for effective use of cloud resources. In Zhan et al. (2015), a survey conducted on cloud computing resource scheduling showed that most of the existing techniques made use of traditional scheduling algorithms modified to work under cloud computing settings. Some common techniques used were the First Come First Serve and Round Robin algorithms which were used for task scheduling but disregarded differences in tasks and resource needs while being effective in ensuring fairness. Heuristics such as Min-Min and Max-Min algorithms for workflow scheduling by Maheswaran et al. (1999) made use of the execution time of tasks and made assumptions regarding their accuracy. Genetic algorithms, as described by Xu et al. (2012), could explore vast solution spaces, but their implementation was slow.

In 2011, Calheiros et al. created CloudSim, a simulation framework that allowed scholars to analyze and assess cloud resource allocation strategies. It was demonstrated by their research that simple heuristic-based resource management resulted in either over-allocation (leading to a waste of 20-40 percent of cloud resources) or under-allocation (affecting cloud application performance negatively by 15-30 percent).

VM Placement and Consolidation
VM placement was an important performance issue tackled through the application of bin packing algorithms and constraint satisfaction problems. Beloglazov et al. (2012) suggested energy savings achieved by means of resource management and VM consolidation employing modified best-fit decreasing algorithms. In this case, energy savings could reach up to 83 percent relative to non-consolidation strategies; however, proper thresholds were needed to ensure adequate performance.

The second strategy that focused on resource management was developed by Hermenier et al. (2009). It consisted in Entropy—a consolidation manager that used constraint programming for computing the best VM placements according to available resources and considering migration costs. This technique was mathematically optimal, yet it took several minutes to be calculated in the case of large clusters.

LOAD BALANCING MECHANISMS

Traditional Load Balancing Algorithms
Load balancing in cloud computing platforms before AI technology was realized using some of the existing algorithms from the field of distributed systems. According to Randles et al. (2010), there is a comparison of several load balancing algorithms such as Round Robin, Weighted Round Robin, Least Connections, and Source IP Hash approaches. The result of their study revealed that there was no optimal algorithm that worked perfectly for all types of workloads.

The round robin algorithm distributed requests evenly without considering the server's capacity or the load, resulting in imbalance among servers that were not alike. The Weighted Round Robin algorithm solved this problem by giving the server a specific weight depending on its capacity. However, the weight in this algorithm was set in advance and couldn't be adjusted depending on the situation.

Dynamic Load Balancing
Nishant et al. (2012) suggested a novel ant colony optimization based load balancing method that exhibited 20-30% improvement compared to the static techniques in regard to response time and resources utilization. Dhinesh Babu and Krishna (2013) suggested a new technique of honey bee behavior inspired load balancing method which exhibited superior results under heterogeneous conditions through imitating the process of honey bees' foraging activities allocating tasks based on the servers' processing capabilities.

Although these algorithms were more flexible as opposed to static solutions, these were still confined to pre-defined behavioral characteristics and lacked the ability to learn from the operational experience. The fine tuning of algorithm parameters like pheromone evaporation and waggle dance was not automatic; it had to be done manually.

Geographic and Network-Aware Load Balancing
Geographic distributions included CDNs and DNS-based load balancing. According to Nygren et al. (2010), Akamai CDN utilized the network and server load to route queries to the most appropriate servers. This solution handled hundreds of millions of queries per day using the combination of DNS-based load balancing and application-level routing techniques. Though sophisticated, the technology was still limited to rule-based approaches. Engineers developed the tree structures using their knowledge and experience, and updated them manually in case the network configuration changed. This process worked well in stable networks, but was unable to adapt quickly enough to unexpected changes.

PERFORMANCE MONITORING AND METRICS

Monitoring Frameworks
Traditional cloud performance monitoring consisted of collecting performance metrics and alerting on thresholds being breached. Aceto et al. (2013) reviewed different cloud monitoring techniques and pointed out that most monitoring systems had a three-step process involving collecting, aggregating, and alerting. Traditional monitoring frameworks, such as Nagios, Ganglia, and Zabbix, used a combination of infrastructure monitoring and periodic SNMP polling to collect metrics (Massie et al., 2004).

These systems collected metrics periodically (1-5 minutes), aggregated them into time-series databases, and generated alerts if threshold conditions were met. The strategy was effective for failure detection but ineffective in spotting potential performance degradation problems.

Key Performance Indicators
However, the performance measures adopted for pre-AI cloud computing systems were precisely known, although their analyses would be done individually. According to Ardagna et al. (2014), some of the most important performance measures include:

Response time/latency (for assessing performance perception)
Throughput (expressed as number of requests per second to indicate capacity)
Resource utilization, comprising CPU usage, memory, disk I/O, and network
Availability (in terms of uptime percentage)
Error rate
The problem with the above-mentioned approach was that the measures were assessed independently without correlating them with one another. For instance, an increase in memory utilization could have a non-linear impact on the response time, which was difficult for operations personnel to recognize with increasing complexity of the system.

Performance Benchmarking
The benchmarking process became essential to assessing cloud performance. One of the early attempts at measuring cloud performance on a grand scale was made by Iosup et al. (2011) who compared Amazon EC2, Google App Engine, and Microsoft Azure. The researchers found that there were significant performance differences between clouds with CPU performance differences as high as 24% among the same type of instances as a result of heterogeneity of hardware and mutual influence of other users.

Another important study that evaluated the effectiveness of cloud computing in scientific applications demonstrated that performance overhead was around 10-20 percent when compared to using a high-performance computing cluster. It was established that such an overhead is caused by virtualization and network virtualization.

Statistical Analysis Methods

The analysis of performance involved the use of classical statistics. Some of the techniques included regression analysis, moving averages, and percentile analysis (Jiang et al., 2009). These techniques were very useful in performing steady state analysis. They could not handle non-linear relations or multi-dimensional correlations.

According to Ghit et al. (2014), statistical process control was employed to determine any deviation from expected performance behavior using tools like control charts and cumulative sum. Despite the mathematical correctness of these tools, they relied heavily on the assumption of stationarity when it came to normal behavior.

FAULT TOLERANCE AND RELIABILITY

Redundancy-Based Approaches

Redundancy was the main strategy used by pre-AI cloud systems to provide fault tolerance. Vishwanath and Nagappan (2010) found that while hardware faults conformed to particular statistical probabilities (bathtub distribution for disk failures, Poisson distribution for memory errors), correlated faults were hard to predict. The strategy adopted was to have N+1 redundancy, geographic distribution, and failover through heartbeat checking.

The cost associated with redundancy was huge. Having spare resources for the case of failure meant that around 30-50% of resources available were kept idle because they would not contribute to handling any active workload.

Checkpoint and Recovery

Fault-tolerance mechanisms of MapReduce included task re-execution following failures in workers. Although effective, such an algorithm had an additional cost of performance because of the regular checkpoints that were needed to save intermediate states to persistent storage and took 5-10% of job execution times.

However, Zaharia et al. (2012) proposed another solution that reduced the costs associated with fault tolerance in distributed systems using Resilient Distributed Datasets in Apache Spark that did not use replication but instead employed the concept of lineage and could recomputed partitions that were lost without holding all replicas. Such an algorithm was about 40-60% less costly than replication in terms of performance.

Reactive Failure Management

Failure detection and recovery in systems before AI were mostly done reactively. According to Oppenheimer et al. (2003), in research conducted on internet service failure, they discovered that 40 percent of failures were caused by operator error while another 25 percent were due to software bugs, and 20 percent due to hardware failure. Failover methods used were mostly manual processes using runbooks as well as some basic automated processes.

According to Ford et al. (2010), in Google, the availability model utilized depended on engineering, testing, and gradual introduction into use. This system was 99.9 percent available but did not predict nor prevent any failures from occurring but minimized the effects when such happen.

ENERGY EFFICIENCY AND GREEN COMPUTING

Power Management Strategies

Energy efficiency was one of the main considerations while working on the pre-AI-based clouds. For example, Beloglazov et al. (2012) suggested energy-efficient resource allocation heuristics for VM consolidation on fewer servers in times of low demand, thus allowing for server shutdown via DVFS. These heuristics worked according to the threshold-based model, using upper and lower CPU utilization bounds for migration decision-making. The downside of such approaches was that they did not account for future changes in demand.

The benefits of such an energy-saving approach were considerable (up to 83% of power reduction); however, it implied risks associated with possible performance problems due to migration and resource contention issues.

Thermal-Aware Computing

In their 2008 study, Tang et al. examined the thermal-aware assignment of tasks within data centers, where their developed algorithms would attempt to distribute computing loads in such a way as to reduce cooling costs. In this study, the authors employed a computational fluid dynamics model, which generated precise predictions but demanded substantial processing power and failed to adapt to dynamic environmental changes.

The link between computational load distribution and thermal efficiency proved to be complicated and non-linear. Heat concentrations may emerge when adjacent machines operate at high capacity, increasing pressure on cooling units. Pre-AI approaches to tackling the issue were based on fixed thermal models and pre-set machine placements.

Power Usage Effectiveness (PUE)
This was defined by The Green Grid as the Power Usage Effectiveness indicator, which became the official way of measuring data center efficiency. According to Barroso & Holzle (2007), conventional data centers maintained Power Usage Effectiveness ratios of between 1.5-2.0, implying that for every one watt expended by computing equipment, there were additional 0.5-1.0 watts being spent on cooling and overhead.

Efficiency enhancement involved airflow optimization, increased operational temperature, and adoption of free cooling technology – methods which are rule-based without any form of prediction capability. Google managed to attain an efficiency rating of 1.12 by adopting engineering best practices, but noted the need for improved efficiency enhancing methodologies.

SECURITY PERFORMANCE CONSIDERATIONS

Performance Impact of Security Mechanisms
Performance impacts resulted from security measures taken prior to the advent of AI technology in the cloud infrastructure. According to Ristenpart et al. (2009), co-location on cloud infrastructure presented possible side-channel attacks on cloud infrastructures necessitating the deployment of performance-hungry isolation schemes. Encryption of stored and transmitted data as a measure of data protection added latency overhead of between five and 15 percent based on encryption algorithms and key sizes (Santos et al., 2009). The challenge in balancing performance and security was done using static policies. Encryption could be switched on or off for whole data types without the possibility of adjusting security levels according to performance or security needs.

Intrusion Detection Systems
Intrusion detection systems (IDS), either network-based or host-based IDS, were dependent on signature matching or statistical anomalies. According to Modi et al. (2013), IDS in cloud computing involved signature-based IDS which had accuracy higher than 99 percent for known attacks and none against new attacks; and statistical anomaly detection, which generated high false positive detections (5-15 percent) making operations less efficient due to false alerts.

The overhead of deep packet inspection was high, requiring 10-20 percent network resources. This indicated that there was a trade-off between network performance and security.

DDoS Mitigation
Prior to AI, DDoS mitigation required traffic analysis by means of threshold-based rate limiting. As per Zargar et al. (2013), an overview of existing DDoS defenses revealed that traditional methods failed to detect and mitigate complex attacks designed to replicate normal user behavior. In other words, rule-based DDoS defense systems had to strike a balance between two competing interests - too aggressive rate limiting and insufficiently aggressive filtering.

IP address-based rate limiting was useful against basic attacks on the network infrastructure, while ineffective in distributed attacks employing large numbers of distinct IP addresses. Moreover, application layer DDoS attacks sending numerous legitimate requests were especially difficult to counteract.

SERVICE LEVEL AGREEMENT (SLA) MANAGEMENT

SLA Definition and Monitoring
SLA management before the advent of AI in the cloud involved the use of static SLAs and post-violation reactive actions. Kearney et al. (2010) developed an SLA management model where negotiations, monitoring, and enforcement were part of the process. Nevertheless, the monitoring involved threshold checks, while the enforcement only involved penalties after violation.

Availability SLAs ranging from 99.9 percent up to 99.95 percent availability were provided, accompanied by financial compensation for violations. Nonetheless, these SLAs did not include response times or throughput as metrics for service level agreements.

QoS Provisioning
QoS service provision was based on resource reservation and priority queuing. According to Buyya et al. (2011), market-based techniques could be used for allocating resources in QoS-aware cloud computing services, and users would have the ability to bid according to their QoS needs. Resource provisioning techniques suggested by Ardagna et al. (2012) utilized control theory with PID (proportional-integral-derivative) controllers to regulate QoS targets.

Although theoretically feasible, the application of PID controllers necessitated the adjustment of proportional, integral, and derivative gain values manually for different applications/metrics. PID controllers showed good performance when applied to single-control variables but poor when it came to optimizing multiple variables simultaneously. The interrelation of controlled variables, such as increasing CPU power to affect response time and throughput, made the problem even more complicated.

Multi-Tenancy Performance Isolation
Isolating the performance among tenants in the shared environment was another major problem. In their study, Krebs et al. (2014) observed interference problems in the cloud setting, where co-located applications faced up to 50 percent performance loss caused by contention issues at the CPU cache, memory bus, network interface, and disk I/O layers.

The possible solutions were strict resource partitioning through the use of cgroups and CPU pinning (affecting utilization efficiency by 20-30 percent) and performance-aware virtual machine placement, relying on heuristic scoring instead of learning models. However, both solutions lacked dynamic adaptation capabilities for ever-changing interference conditions.

LIMITATIONS AND CHALLENGES OF PRE-AI CLOUD PERFORMANCE MANAGEMENT

Scalability of Manual Configuration
With the rise in scale of cloud computing environments, the process of configuring performance policies manually became impractical. Barroso et al. (2013) mentioned warehouse-scale computers making billions of requests each day where complexity due to interaction among thousands of services was beyond human ability. In a rule-based approach, the number of rules required increases exponentially with the growth in the complexity of the system.

An average cloud environment configuration might involve many thousands of individual threshold settings, scaling rules, and placement rules. Ensuring consistency in such configurations needed an army of people and advanced configuration management techniques, but still mistakes occurred quite often.

Inability to Handle Non-Linear Patterns
The traditional statistical approaches assumed linearity and stationarity. Iqbal et al. (2011) showed that cloud applications had non-linear scalability, which couldn't be explained by linear models due to the complicated performance landscapes produced by interdependencies among several resource types (e.g., CPU, memory, network, and disk). For instance, an application could have a linear scale-up till reaching 60 percent CPU usage, but its performance would drop exponentially after exceeding that point because of the joint effect of garbage collection and memory load on CPU usage.

Reactive Rather Than Proactive Management
One of the most notable limitations of pre-AI cloud performance management is its inherent reactiveness. According to the findings of Islam et al. (2012), reactive auto-scaling caused lags of 5 to 15 minutes when there were demand changes. The problem was that the performance degraded or wasted resources within this lag period. Predictive methods did exist for performance management, but they were only able to forecast demand at certain periods and were not applicable to more complicated multi-faceted workload dynamics.

Being reactive meant that any performance issues would not be found until they started negatively impacting end-users. Only by then would threshold violations be observed, alerts be raised, and scaling actions be launched.

Lack of Holistic Optimization
Traditional techniques usually focused on optimizing each performance metric independently. In their paper, Zhu et al. (2009) observed that maximizing an objective (for instance, response time) would cause degradation of another objective (e.g., resource usage or power consumption). Several techniques have been proposed to handle multi-objective optimizations using weighted sum methods and Pareto-optimality approaches; however, such approaches needed the weights to be set manually.

The absence of holistic optimization techniques led to a series of trade-offs faced by the cloud managers. For instance, any improvement in performance would mean allocating more resources to the system, thus incurring additional costs. Any effort to boost efficiency would require consolidation, which could affect performance negatively. Finally, enhancing availability necessitated some form of redundancy, which wasted available resources.

Limited Anomaly Detection Capabilities

For anomaly detection, static thresholds were used or simple statistical methods (when values exceeded three standard deviations from the average). According to Tan et al. (2012), such solutions generated too many false positives because in a changing environment, even regular workloads changed dynamically, and there was no way to tell the difference. As a result, people got used to receiving unnecessary alerts, and empirical research confirmed that about 90 percent of all alerts were false.

Alert fatigue was dangerous because people who saw hundreds of false alarms per day stopped paying attention to them and did not detect any true problems.

DISCUSSION

The State of Cloud Performance Pre-AI

It can be seen from the literature that the pre-AI cloud infrastructure scaled and worked reliably thanks to rigorous engineering and redundant designs, along with well-developed heuristics. Google, Amazon, and Microsoft all managed to build globally distributed systems serving billions of users at 99.95% or better availability. However, this was done at an extremely high cost of overprovisioning (30-50%), high operational costs, requiring extensive site reliability engineering staff, and reliance on human know-how.

It became apparent in the pre-AI era that classical computer science approaches, such as queuing theory, control theory, heuristic optimization, and statistics, were capable of building functional cloud systems but not maximizing them theoretically.

The Transition Point

The limitations found in the literature motivated the application of AI. The combination of three factors led to this evolution. The first one is that due to the explosive nature of cloud computing, it became unmanageable because humans are unable to manage things beyond their cognitive abilities. Secondly, the massive amounts of data obtained from monitoring operations (petabytes collected during the years of operation) allowed training ML models. Lastly, the progress made in the ML algorithms field, namely deep and reinforcement learning, gave researchers the ability to detect non-linear high-dimensional patterns.

Evolution did not occur instantly; rather, it happened gradually. At the beginning stages of the application of AI into cloud computing, people applied AI solutions to solve certain issues, for example, workloads prediction or detecting anomalies. After obtaining positive results, researchers started using AI in other fields of cloud computing.

Legacy of Pre-AI Approaches

Pre-AI methodologies are still employed as back-up solutions or as an underpinning basis within the current AI-based systems. Threshold scaling continues to exist as a back-up solution in case machine learning prediction is unsure about some things. Heuristic algorithms offer solutions that are then improved upon by machine learning models. Statistical monitoring gives the data stream input that is used by machine learning algorithms to detect anomalies.

Moreover, pre-AI performance metrics, benchmarks, and evaluation models have continued to be the benchmark criteria for measuring cloud performance. The performance criteria identified by researchers in the pre-AI era (response time, throughput, availability, and utilization) have continued to be the most relevant metrics that cloud systems are assessed on.

CONCLUSION

The current study has conducted an extensive investigation on the management techniques that were used to enhance performance prior to the advent of AI technologies. Prior to AI (around 2006-2016), there was reactive management through threshold crossing, heuristics, statistics, and redundancy. Although the aforementioned techniques have helped establish cloud infrastructures catering to billions of users, they have inherent disadvantages in terms of predictions, adaptation, multidimensionality, and management scalability.

The major conclusions drawn from the review include:

(1) The management of resources utilized methods such as static allocation, threshold-based automatic scaling, and heuristics to schedule requests, which failed to adapt to the dynamics of workloads and thus led to 20-40% wastage of resources or 15-30% drop in performance levels.

(2) Balancing of loads involved utilization of algorithms like Round Robin, Least Connections, etc., which failed to take into consideration the performance characteristics of applications.

(3) The performance monitoring system only included metric gathering and alerting when certain thresholds were crossed, lacking the capability of detecting subtle deviations or predicting future problems, with an error rate of 5-15 percent.

(4) The fault-tolerance system relied on redundancy and failover instead of proactive detection, necessitating 30-50 percent extra capacity for such events.

(5) The energy efficiency optimization employed simple consolidation techniques which did not account for demand trends, leading to a PUE of 1.5-2.0 instead of the optimal value of 1.0.

(6) The SLA management system was reactionary and did not take any preventative action, with only minimal performance guarantees regarding availability.

(7) Security policies introduced between 5% to 20% performance overhead due to their static nature, which was unable to accommodate performance considerations dynamically in relation to security.

In summary, the above shortcomings have motivated the use of AI and ML techniques in the management of cloud platform architecture and operations as the new paradigm in cloud computing. Pre-AI era, despite being constrained by the outlined challenges, laid the foundation for the evolution of the cloud computing architecture, processes, and performance. The future study should concentrate on determining the actual performance benefits realized from the use of AI in all three areas discussed above.

REFERENCES

1. Aceto, G., Botta, A., De Donato, W., and Pescape, A. (2013). Cloud monitoring: A survey. Computer Networks, 57(9), 2093-2115.
2. Ardagna, D., Casale, G., Ciavotta, M., Perez, J. F., and Wang, W. (2014). Quality-of-service in cloud computing: modeling techniques and their applications. Journal of Internet Services and Applications, 5(1), 1-17.
3. Ardagna, D., Panicucci, B., Trubian, M., and Zhang, L. (2012). Energy-aware autonomic resource allocation in multitier virtualized environments. IEEE Transactions on Services Computing, 5(1), 2-19.
4. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.
5. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177.
6. Barroso, L. A., and Holzle, U. (2007). The case for energy-proportional computing. Computer, 40(12), 33-37.
7. Barroso, L. A., Clidaras, J., and Holzle, U. (2013). The datacenter as a computer: An introduction to the design of warehouse-scale machines. Synthesis Lectures on Computer Architecture, 8(3), 1-154.
8. Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, 28(5), 755-768.
9. Buyya, R., Ranjan, R., and Calheiros, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities. Proceedings of the 7th International Conference on High Performance Computing and Simulation, 1-11.
10. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6), 599-616.
11. Buyya, R., Garg, S. K., and Calheiros, R. N. (2011). SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. Proceedings of the International Conference on Cloud and Service Computing, 1-10.
12. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23-50.

13. Calheiros, R. N., Masoumi, E., Ranjan, R., and Buyya, R. (2014). Workload prediction using ARIMA model and its impact on cloud applications QoS. IEEE Transactions on Cloud Computing, 3(4), 449-458.
14. Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. (2005). Live migration of virtual machines. Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation, 273-286.
15. Dean, J., and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.
16. Dhinesh Babu, L. D., and Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing, 13(5), 2292-2303.
17. Ford, D., Labelle, F., Popovici, F. I., Stokely, M., Truber, V. A., Barroso, L., Grimes, J., and Quinlan, S. (2010). Availability in globally distributed storage systems. Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, 61-74.
18. Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. Grid Computing Environments Workshop, 1-10.
19. Ghit, B., Yigitbasi, N., Iosup, A., and Epema, D. (2014). Balanced resource allocations across multiple dynamic MapReduce clusters. Proceedings of the 15th International Middleware Conference, 51-62.
20. Gmach, D., Rolia, J., Cherkasova, L., and Kemper, A. (2007). Workload analysis and demand prediction of enterprise data center applications. IEEE International Symposium on Workload Characterization, 171-180.
21. Hermenier, F., Lorca, X., Menaud, J. M., Muller, G., and Lawall, J. (2009). Entropy: a consolidation manager for clusters. Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 41-50.
22. Iosup, A., Yigitbasi, N., and Epema, D. (2011). On the performance variability of production cloud services. Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 104-113.
23. Iqbal, W., Dailey, M. N., Carrera, D., and Janecek, P. (2011). Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Future Generation Computer Systems, 27(6), 871-879.
24. Islam, S., Keung, J., Lee, K., and Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. Future Generation Computer Systems, 28(1), 155-162.
25. Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H. J., and Wright, N. J. (2010). Performance analysis of high performance computing applications on the Amazon web services cloud. Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science, 159-168.
26. Jiang, H., Jin, S., and Wang, C. (2009). Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems, 22(6), 1064-1071.
27. Kearney, K. T., Torelli, F., and Kotsokalis, C. (2010). SLA: An abstract syntax for service level agreements. Proceedings of the 11th IEEE/ACM International Conference on Grid Computing, 217-224.
28. Krebs, R., Momm, C., and Kounev, S. (2014). Metrics and techniques for quantifying performance isolation in cloud environments. Science of Computer Programming, 90, 116-134.
29. Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 12(4), 559-592.
30. Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., and Freund, R. F. (1999). Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing, 59(2), 107-131.
31. Massie, M. L., Chun, B. N., and Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing, 30(7), 817-840.
32. Mell, P., and Grance, T. (2011). The NIST definition of cloud computing. NIST Special Publication 800-145.
33. Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., and Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. Journal of Network and Computer Applications, 36(1), 42-57.
34. Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K. P., and Rastogi, R. (2012). Load balancing of nodes in cloud using ant colony optimization. Proceedings of the 14th International Conference on Computer Modelling and Simulation, 3-8.
35. Nygren, E., Sitaraman, R. K., and Sun, J. (2010). The Akamai network: a platform for high-performance internet applications. ACM SIGOPS Operating Systems Review, 44(3), 2-19.
36. Oppenheimer, D. L., Ganapathi, A., and Patterson, D. A. (2003). Why do internet services fail, and what can be done about it? Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems, 1-16.
37. Randles, M., Lamb, D., and Taleb-Bendiab, A. (2010). A comparative study into distributed load balancing algorithms for cloud computing. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, 551-556.

38. Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Proceedings of the 16th ACM Conference on Computer and Communications Security, 199-212.
39. Santos, N., Gummadi, K. P., and Rodrigues, R. (2009). Towards trusted cloud computing. Proceedings of the Conference on Hot Topics in Cloud Computing, 3-3.
40. Sotomayor, B., Montero, R. S., Llorente, I. M., and Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing, 13(5), 14-22.
41. Tan, Y., Nguyen, H., Shen, Z., Gu, X., Venkatramani, C., and Rajan, D. (2012). PREPARE: Predictive performance anomaly prevention for virtualized cloud systems. Proceedings of the IEEE 32nd International Conference on Distributed Computing Systems, 285-294.
42. Tang, Q., Gupta, S. K., and Varsamopoulos, G. (2008). Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. IEEE Transactions on Parallel and Distributed Systems, 19(11), 1458-1472.
43. Tawfeek, M. A., El-Sisi, A., Keshk, A. E., and Torkey, F. A. (2013). Cloud task scheduling based on ant colony optimization. Proceedings of the 8th International Conference on Computer Engineering and Systems, 64-69.
44. Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. ACM Transactions on Autonomous and Adaptive Systems, 3(1), 1-39.
45. Vishwanath, K. V., and Nagappan, N. (2010). Characterizing cloud computing hardware reliability. Proceedings of the 1st ACM Symposium on Cloud Computing, 193-204.
46. Xu, J., Fortes, J. A., and Buyya, R. (2012). Multi-objective virtual machine placement in virtualized data center environments. Proceedings of the IEEE/ACM International Conference on Green Computing and Communications, 179-188.
47. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, 15-28.
48. Zargar, S. T., Joshi, J., and Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. IEEE Communications Surveys and Tutorials, 15(4), 2046-2069.
49. Zhan, Z. H., Liu, X. F., Gong, Y. J., Zhang, J., Chung, H. S. H., and Li, Y. (2015). Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surveys, 47(4), 1-33.
50. Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, 1(1), 7-18.
51. Zhu, X., Young, D., Watson, B. J., Wang, Z., Rolia, J., Singhal, S., McKee, B., Hyser, C., Gmach, D., Gardner, R., Christian, T., and Cherkasova, L. (2009). 1000 islands: Integrated capacity and workload management for the next generation data center. Proceedings of the International Conference on Autonomic Computing, 172-181.