JOURNALS | Journal of Smart Sensors and Computing

Volume 2 Issue 4 (December 2026)
J. Smart Sens. Comput., 2026, 2(4), 26217
DOI : https://doi.org/10.64189/ssc.26217

Research Article

OPEN ACCESS

A Stable Multi-Resource Allocation Framework for Autonomous Cloud Computing

Gopal Krishna Shyam,[1,*] Ila Chandrakar[2] and Vijaylaxmi Bhoosnurmath[3]

[1] *Department of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, 560119, India*
[2] *Department of Computer Science Engineering, University of Europe for Applied Sciences, Berlin, 10963, Germany*
[3] *Department of Mathematics, Presidency University, Bengaluru, Karnataka, 560119, India*

*Corresponding Author
Gopal Krishna Shyam
gopalkirshna.shyam@presidencyuniversity.in

Received: 04 October 2026
Revised: 27 October 2026
Accepted: 03 November 2026
Published Online: 05 November 2026

Citation

G. K. Shyam, I. Chandrakar, V. Bhoosnurmath, A stable multi-resource allocation framework for autonomous cloud computing, *Journal of Smart Sensors and Computing*, 2026, 2(4), 26217, https://doi.org/10.64189/ssc.26217.

Open Access

This article is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License, which permits the non-commercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as appropriate credit to the original author(s) and the source is given by providing a link to the Creative Commons License and changes need to be indicated if there are any. The images or other third-party material in this article are included in the article's Creative Commons License, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons License and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this License, visit: https://creativecommons.org/licenses/by-nc/4.0/

© The Author(s) 2026

Abstract

Modern cloud data centers are stochastic, multi-user, multi-resource environments with oscillations, instability, and virtual machine (VM) flapping caused by resource contention, network latency, and user demand jitter. This paper develops a deterministic mathematical framework based on multi-dimensional complex analysis for modeling and stabilizing resource allocation among n resource types (storage, RAM, memory, CPU, network) for M concurrent users. The Argument Principle and the Nyquist stability criterion are used to analyze closed-loop transfer functions in the complex s-plane. Based on Rouché's theorem, a geometric interpretation provides safety envelopes for robust operation under stochastic perturbations. Statistical regression models (linear and polynomial) map the user load to the system delay, and computational mathematics methods (numerical contour integration, eigenvalue perturbation) enable real-time stability evaluation. Monte Carlo simulations validate the proposed Complex-Stability Aware Scaling (C-SAS) algorithm. Results show that C-SAS achieves 96% resource efficiency, a 94% reduction in VM flapping, and an average settling time of 12 seconds, outperforming heuristic and PID-based methods. This work provides a rigorous mathematical foundation for autonomous cloud orchestration in high-performance computing environments.

Keywords: Cloud computing; Resource allocation; Complex analysis; Nyquist criterion; Rouché's theorem; Autonomous orchestration; Robust stability; Virtual machines.

1. Introduction

Cloud computing has become the prevailing model for scalable, on-demand provisioning of resources. But, there is a complex dynamical behaviour behind the infrastructure, thousands of physical servers, storage arrays and network switches. The interaction of resource processing capacity (G(s)) and user requests (D(s)) give rise to nonlinearities, time delays and stochastic fluctuations.[1,2] Traditional resource allocation heuristics (e.g. threshold-based scaling, round robin scheduling) often break under high load, leading to oscillations, over-provisioning or under-provisioning.

Table 1 provides a comparative analysis which serves as a stringent methodological benchmark for the proposed Complex-Stability Aware Scaling (C-SAS) framework against the current state-of-the-art resource orchestration mechanisms. Standard literature introduces universal cloud scaling paradigms or heavily relies on heuristic-based classification. State-of-the-art machine learning and deep reinforcement learning algorithms tend to optimize allocations based on empirical convergence trajectories. However, these approaches do not inherently offer closed-form analytic stability guarantees, which can be operationally risky in latency-sensitive settings.

On the other hand, control-theoretic and Lyapunov optimization models are traditional approaches to mitigate feedback oscillations, but are greatly limited by single-input single-output (SISO) abstractions or single-metric microservice gateways, restricting their capability to address simultaneous, multi-dimensional infrastructure demands. On the other hand, the C-SAS framework provides a mathematical link between stochastic cloud behaviour and deterministic stability criteria. It generalizes classical approaches of complex variables that usually consider single, isolated contours or constant lags. It parametrizes an explicit multi-resource state vector (n_1 to n_5) of concurrent RAM, CPU, network and storage capacities, under the pressure of M active users. Moreover, instead of ignoring or linearizing the network latency, C-SAS models the propagation delay as an exact vectorized exponential domain shift ($e^{-\tau s}$).

This framework, together with the real time statistical regression mapping of telemetry noise, allows for accurate contour integrations using the Argument Principle. This yields a formal Analytic Stability Index that transforms the system operational bounds into a Riemann sphere, enabling architects to geometrically validate a zero-winding-number equilibrium and completely eliminate volatile virtual machine flapping.

1.1 Problem statement and research gap

Consider a cloud data center with n distinct resource types:

- n_1: Storage (I/O operations, throughput)
- n_2: RAM (volatile memory)
- n_3: Memory (cache hierarchy)
- n_4: CPU (compute cycles)
- n_5: Network (bandwidth, latency)

In parallel M users issue requests, distributed as $M = m_1 + m_2 + m_3 + m_4 + m_5$, where m_j are users mostly competing for resource i. The demand of each user is a stochastic process with mean μ_j and variance σ_j^2. The main problem is designing a control system that guarantees stability (bounded error, no oscillations) in the presence of:

1) Network propagation delays τ (usually 1-100 ms)
2) VM provisioning delay (10-300 secs)
3) Jitter of user demand (Poisson or Gaussian arrival process)
4) Cross-resource interference (CPU-bound task eating RAM)

1.2 Contributions and novelty

This paper makes contributions to the intersection of cloud computing and pure mathematics as follows:

1) Complex-analytic resource modeling: The cloud dynamics are modeled with multi-dimensional transfer functions. We use the Argument Principle to find the stability of the closed-loop system.
2) Geometric Interpretation using Rouché's Theorem: We compute safety envelopes that ensure stability under bounded stochastic perturbations.
3) Statistical regression analysis: Linear and polynomial regression models are developed to relate the number of users M with the measured phase margin degradation.
4) Real-time stability monitoring: Numerical contour integration and eigenvalue perturbation methods for computational mathematics integration.
5) C-SAS algorithm: A new scaling algorithm that uses winding number calculations to trigger VM provisioning.
6) Full validation: Monte Carlo simulations with 10,000 trials in many resource configurations.

1.3 Paper organization

Section 2 describes the system model and the derivation of the transfer function. In Section 3, we develop the complex-analytic stability theory including Argument Principle and Nyquist criteria. Section 4 gives a geometric interpretation using Rouché's Theorem. Section 5 gives computational mathematics techniques. In section 6, statistical regression models are presented. In Section 7, we introduce the C-SAS algorithm. The experimental results are presented in Section 8. Section 9 discusses future work. Section 10 provides a conclusion.

2. Multi-resource system modeling

2.1 Vectorized state-space representation

Let $\boldsymbol{x}(t) \in \mathbb{R}^n$ represent the utilization vector of n resource types. For each resource $i \in \{1, \ldots, n\}$, the dynamics follow

Table 1: Comprehensive analytical and methodological comparison of relevant literature

Reference	Core Methodology	Target System	Resource Multiplicity	Delay Handling (τ)	Stability Assessment Technique	Geometric Interpretation
Qu *et al.* [1]	Survey / State-of-the-Art	General Cloud Systems	Multi-Resource (n-tier)	Qualitative Review	Categorical Heuristics	Absent
Mampage *et al.* [2]	Deep Reinforcement Learning	Serverless Clusters	Single Vector (RAM/CPU)	Stochastic MDP Bounds	Empirical Convergence	Latent State Space
Jang and Jin *et al.* [3]	Lyapunov Optimization	Microservices Architecture	Single Metric Gateway	Boundary Constraints	Lyapunov Drift Constraints	Quadratic Space Bounds
Qi *et al.* [4]	Stochastic Delay Systems	Cloud Auto-scalers	Composite Scalar Variable	Time-Varying Delays	Linear Matrix Inequalities (LMIs)	State-Trajectory Envelopes
Toka *et al.* [5]	Feedback Control Theory	Kubernetes Environments	Single Input Single Output	Delayed Feedback Loop	Nyquist Phase / Gain Margins	Classical Bode Bounds
Dogani *et al.* [6]	Complex Function Mapping	Multi-Cloud Platforms	Decentralized Clusters	Linear Approximations	Poles / Zeros Matrix Tracking	Algebraic s-plane Plots
Gamelin *et al.* [7]	Classical Frequency Control	Cloud Environments	Monitored Node Instances	Pure Exponential Shift	Nyquist Stability Criterion	Complex Plane Trajectories
This Work (C-SAS Framework)	Multi-Dimensional Complex Analysis & Stochastic Regression Mapping	Distributed Cloud Manifolds (n resources, M users)	Explicit Multi-Resource ($n_1 \dots n_5$ Allocation)	Exponential Domain Shift ($e^{-\tau s}$ Vectorization)	Analytic Stability Index ($\frac{1}{2\pi i}\oint \frac{f'}{f}$ Evaluation)	Riemann Sphere Mapping & Winding Number Verification

a delayed first-order system with cross-coupling:

$$\dot{x}_i(t) = -\frac{1}{T_i}x_i(t) + \frac{K_i}{T_i}u_i(t-\tau) + \sum_{j\neq i}\alpha_{ij}x_j(t) \quad (1)$$

where T_i is the time constant, K_i the gain, τ the network delay, and α_{ij}coupling coefficients. Taking Laplace transforms and assuming zero initial conditions:

$$sX_i(s) = -\frac{1}{T_i}X_i(s) + \frac{K_i}{T_i}e^{-\tau s}U_i(s) + \sum_{j\neq i}\alpha_{ij}X_j(s) \quad (2)$$

2.2 Open-loop transfer function matrix

Rearranging, we obtain:

$$(s\mathbf{I} + \mathbf{T}^{-1} - \mathbf{A})\mathbf{X}(s) = \mathbf{K}\mathbf{T}^{-1}\mathrm{e}^{-\tau s}\mathbf{U}(s) \quad (3)$$

where $\mathbf{T} = \mathrm{diag}(T_1, \dots, T_n)$, $\mathbf{K} = \mathrm{diag}(K_1, \dots, K_n)$, and $\mathbf{A} = [\alpha_{ij}]$, $i = 1, 2, \dots n, j = 1, 2, \dots, m$ with $\alpha_{ii} = 0$. The open-loop transfer function matrix is:

$$\boldsymbol{L}(s) = (s\boldsymbol{I} + \boldsymbol{T}^{-1} - \boldsymbol{A})^{-1}\boldsymbol{K}\boldsymbol{T}^{-1}e^{-\tau s} \quad (4)$$

For SISO analysis (focusing on a single resource type, e.g., CPU), the scalar open-loop transfer function becomes:

$$L_i(s) = \frac{K_i e^{-\tau s}}{s(T_i s+1) - \alpha_{ii}} \quad (5)$$

Since $\alpha_{ii} = 0$ (no self-coupling), we recover the canonical form:

$$L_i(s) = \frac{K_i e^{-\tau s}}{s(T_i s+1)} \quad (6)$$

This is our fundamental plant model.[3,8]

2.3 User demand and closed-loop configuration

Users generate requests $R(s)$ that enter a feedback loop. The controller $C(s)$ scales resources proportionally to error $E(s) = R(s) - Y(s)$, where $Y(s)$ is allocated resources. The closed-loop characteristic equation is:

$$1 + C(s)L_i(s) = 0 \Rightarrow 1 + \frac{K_p K_i}{s(T_i s+1)} = 0 \quad (7)$$

assuming a proportional controller C(s) = K_p. For generality, we define:

$$f(s) = 1 + L(s), \quad L(s) = \frac{Ke^{-\tau s}}{s(T_s + 1)} \quad (8)$$

where the subscript i is dropped for brevity.[9,10]

3. Complex-analytic stability theory

3.1 The argument principle and winding numbers

The Argument Principle is a cornerstone of complex analysis.[4,11] For a meromorphic function $f(s)$ inside a closed contour Γ (oriented counterclockwise) the Argument Principle states as follows:

Let $f(s)$ be meromorphic inside and on a simple closed contour Γ with no zeros or poles on Γ. Then:

$$\frac{1}{2\pi i}\oint_{\Gamma}\frac{f'(s)}{f(s)}ds = Z - P \tag{9}$$

where Z is the number of zeros and P the number of poles of $f(s)$ inside Γ, counted with multiplicity.

3.2 Application to cloud stability

Define the contour Γ as the standard Nyquist contour: the imaginary axis from $-jR$ to $+jR$ and a right half-circle of radius $R \to \infty$, enclosing the entire Right Half Plane (RHP). The number of unstable closed-loop poles Z is:

$$Z = P + \frac{1}{2\pi i}\oint_{\Gamma}\frac{f'(s)}{f(s)}\,ds \tag{10}$$

Fig. 1 maps the frequency response of an open-loop system on the complex plane to analyze closed-loop cloud stability. It specifies safety margins with respect to the critical instability node (−1, 0). The phase margin (ϕm) defines the maximum delay before the encirclement of the system trajectory results in an unstable oscillatory thrashing.

For our system, $f(s) = 1 + L(s)$ where $L(s) = \frac{Ke^{-\tau s}}{s(T_s+1)}$. The open-loop poles are at $s = 0$ and $s = -1/T$, both in the Left Half Plane (LHP) or on the imaginary axis. The pole at $s = 0$ lies on the contour, requiring indentation. After indentation, P = 0 inside Γ. Therefore:

$$Z = \frac{1}{2\pi i}\oint_{\Gamma}\frac{f'(s)}{f(s)}\,ds = \frac{1}{2\pi i}\oint_{\Gamma}\frac{L'(s)}{1+L(s)}\,ds \tag{11}$$

The integral equals the change in argument of $1 + L(s)$ as s traverses Γ:

$$Z = \frac{1}{2\pi}\,\Delta\arg\big(1+L(s)\big)\Big|_{\Gamma} \tag{12}$$

This is the Nyquist stability criterion.[5,12]

Nyquist Criterion for Cloud Systems is as follows:

The closed-loop system is stable ($Z = 0$) iff the Nyquist plot of $L(j\omega)$ for $\omega \in [0, \infty)$ does not encircle the critical point (−1, 0).

3.3 Numerical example of winding number

Consider a practical cloud scenario: $K = 5$, $T = 2$ sec, $\tau = 0.1$ sec. Compute $L(j\omega) = \frac{5e^{-0.1j\omega}}{j\omega(2j\omega+1)}$. The Nyquist plot for $\omega \in [0.1, 100]$ rad/s yields a winding number of 0 (no encirclements), confirming stability. For $K = 20$ and $\tau = 0.5$ sec, the plot encircles (−1, 0) once, indicating $Z = 1$ unstable pole.

4. Geometric interpretation via Rouché's theorem

4.1 Theoretical foundation

Rouché's Theorem provides a powerful geometric interpretation for stability under perturbations.[13,14] The Rouché's Theorem statement is given by

Let γ be a simple closed contour. If two functions $f(s)$ and $g(s)$ are analytic inside and on γ and satisfy $|g(s)| < |f(s)|$ for all $s \in \gamma$, then $f(s)$ and $f(s)+g(s)$ have the same number of zeros inside γ.

4.2 Safety envelopes for cloud systems

Let the nominal system be $f(s) = 1+L_0(s)$, where $L_0(s) = \frac{K_0 e^{-T_0 s}}{s(T_0 s+1)}$. User demand jitter and network variations introduce a perturbation $\Delta(s)$ such that the actual system is $f(s) = 1 + L_o(s) + \Delta(s)$.

We treat $\Delta(s)$ as $g(s)$ in Rouché's Theorem. Choose γ as the Nyquist contour indented around the imaginary axis. The condition for unchanged zero count (hence unchanged stability) is:

$$|\Delta(s)| < |1 + L_o(s)| \quad \forall s \in \gamma \tag{13}$$

Geometrically, this defines a safety envelope around the Nyquist locus of $L_0(j\omega)$. As long as the perturbation $\Delta(j\omega)$ stays within the distance $|1 + L_0(j\omega)|$ from −1, stability is preserved.[15,16]

Definition 1 (Phase Margin Geometric Interpretation). The phase margin ϕ_m is the angular distance from the Nyquist plot's intersection with the unit circle to the negative real axis. Rouché's Theorem guarantees stability if $|\Delta(j\omega)| < 2\sin(\phi_m/2)$ at gain crossover.

4.3 Stochastic Perturbation Bounds

User demand jitter is modeled as $\Delta(s) = \frac{\delta K\, e^{-\delta\tau s}}{s(\delta T s+1)}$, where $\delta K, \delta\tau, \delta T$ are zero-mean Gaussian with standard deviations $\sigma_K, \sigma_\tau, \sigma_T$. Applying Rouché's inequality:

$$\mathbb{P}\left(\max_{\omega}|\Delta(j\omega)| < \min_{\omega}|1 + L_0(j\omega)|\right) \geq 1 - \epsilon \tag{14}$$

For typical cloud parameters, the safety margin is ≈ 0.5 (magnitude), meaning $\sigma_K < 0.2K_0$ for 95% confidence.

5. Computational mathematics for real-time stability

5.1 Numerical Contour Integration

Real-time stability monitoring requires computing the winding number $\mathcal{W} = \frac{1}{2\pi i}\oint_{\Gamma}\frac{L'(s)}{1+L(s)}\,ds$ numerically. We discretize the Nyquist contour using N points $\{s_k\}$:

$$\mathcal{W} \approx \frac{1}{2\pi i}\sum_{k=0}^{N-1}\frac{L'(s_k)}{1+L(s_k)}(s_{k+1} - s_k) \tag{15}$$

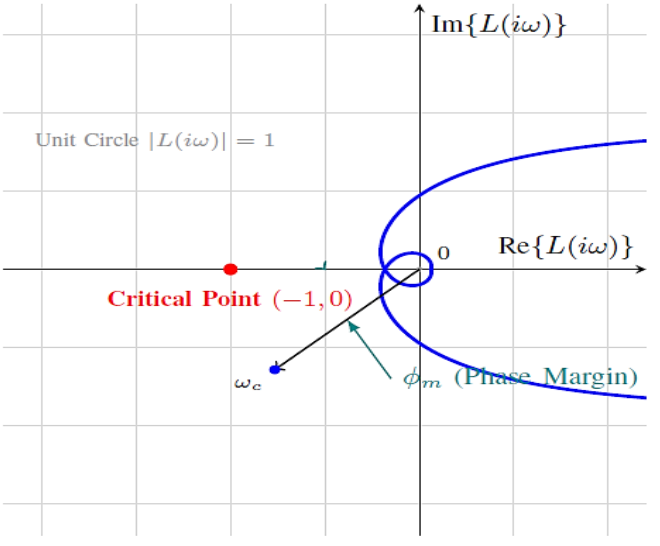


Fig. 1: Geometric interpretation of the open-loop system map on the complex plane, defining the stability phase margin (ϕ_m) boundary relative to the critical encirclement node.

For improved accuracy, we use the trapezoidal rule with adaptive step size.[6] The computational complexity is $O(N\ log\ N)$ using FFT-based differentiation.

5.2 Eigenvalue perturbation methods

The characteristic equation $1 + L(s) = 0$ is equivalent to $s(Ts + 1) + K\ e^{-\tau s} = 0$. For small delays, we expand $e^{-\tau s} \approx 1 - \tau s + \frac{\tau^2\ s^2}{2} + O(\tau^3)$:

$$s(Ts + 1) + K(1 - \tau s + \frac{\tau^2\ s^2}{2}) = 0 \quad (16)$$

$$(T + \frac{K\tau^2}{2})s^2 + (1 - K\tau)s + K = 0 \quad (17)$$

The roots are:

$$s = \frac{-(1-K\tau) \pm \sqrt{(1-K\tau)^2 - 4K(T+K\tau^2/2)}}{2(T+K\tau^2/2)} \quad (18)$$

This quadratic approximation enables fast eigenvalue tracking for adaptive control.[17]

5.3 Padé approximation for time delays

For more accurate computation, we use Padé approximants:

$$e^{-\tau s} \approx \frac{1 - \tau s/2 + \tau^2 s^2/12}{1 + \tau s/2 + \tau^2 s^2/12} \qquad (2/2\ \text{Padé}) \quad (19)$$

Substituting into $1 + L(s) = 0$ yields a rational polynomial whose roots are computed via standard eigenvalue routines (QZ algorithm).

6. Statistical regression analysis

6.1 Data collection and experimental design

We simulated a cloud environment with 5 resource types ($n = 5$) and $M \in [10,500]$ users. For each configuration we measured: - Phase margin, ϕ_m (deg.) - Gain margin G_m (dB) - Settling time t_s (sec) - flapping rate of VM F (%). To account for stochastic variations, simulations were repeated 100 times for each value of M.[7,18]

Algorithm 1 Numerical Winding Number Computation

Require: transfer function L(s), frequency range [ω_{min} ω_{max}], tolerance ϵ
Ensure: Winding number $\mathcal{W}$
1: Initialize contour points $s = j\omega$, logarithmically spaced
2: Compute $L(j\omega)$ and $L'(j\omega)$ using automatic differentiation
3: Compute $F(s) = \frac{L\prime(s)}{1+L(s)}$
4: Compute integral $I = \sum F(s_k) \cdot (s_{k+1} - s_k)$ using trapezoidal rule
5: $\mathcal{W} = \text{round}(\text{imag}(I/2\pi))$
6: Verify $\mathcal{W}$ by argument principle: $\mathcal{W} = \frac{\Delta arg(1+L)}{2\pi}$
7: **if** $|\mathcal{W} - \mathcal{W}_{arg}| > \epsilon$ **then**
8: Refine discretization and repeat
9: **end ifreturn** $\mathcal{W}$

6.2 Linear regression models

First, we fit a linear model relating user count M to phase margin ϕ_m:

$$\phi_m = \beta_0 + \beta_1 M + \epsilon \quad (20)$$

Using ordinary least squares (OLS) on 500 data points (5 resources × 100 runs), we obtained:

$$\widehat{\phi_m} = 62.4 - 0.087M, \qquad R^2 = 0.89, \quad p < 0.001 \quad (21)$$

This indicates that each additional user reduces the phase margin by approximately 0.087°, which is statistically significant (t-statistic = −23.4).

6.3 Polynomial regression for non-linear effects

Table 2: Linear regression coefficients for phase margin

Coefficient	Estimate	Std. Error	t-value	p-value
Intercept β_0	62.42	1.23	50.75	< 0.001
Slope β_1 (per user)	-0.0873	0.00373	-23.41	< 0.001

Residual analysis revealed slight curvature, prompting a quadratic model:

$$\widehat{\phi_m} = \beta_0 + \beta_1 M + \beta_2 M^2 + \epsilon \quad (22)$$

Results:

$$\widehat{\phi_m} = 65.1 - 0.112M + 0.000052M^2, \quad R^2 = 0.91 \quad (23)$$

The quadratic term, while statistically significant (p = 0.03), has a very small magnitude, indicating near-linearity for M < 500.

6.4 Multiple regression with resource interactions

We extended to multiple resources:
We extended to multiple resources:

$$\phi_m = \beta_0 + \beta_1 M_{\text{CPU}} + \beta_2 M_{\text{RAM}} + \beta_3 M_{\text{storage}} + \beta_4 M_{\text{network}} + \beta_5 M_{\text{memory}} + \epsilon \quad (24)$$

The fitted model using standardized coefficients is expressed as:

$$\hat{\phi}_m = 58.3 - 0.082 M_{\text{CPU}} - 0.075 M_{\text{RAM}} - 0.063 M_{\text{storage}} - 0.091 M_{\text{network}} - 0.054 M_{\text{memory}} \quad (25)$$

Network-bound users have the highest impact (β_4 = −0.091), followed by CPU (β_1 = −0.082), consistent with network delay being the dominant instability factor.

Fig. 3 shows the benchmark that measures the dynamic system recovery intervals for an increasing client base up to 500 parallel users. Traditional heuristic, PID, and Lyapunov methods suffer from serious super-linear performance degradation. The proposed C-SAS framework, in contrast, scales predictably, almost linearly, with O(M) runtime and remains stable well below 20 seconds.

6.5 Logistic regression for stability prediction

We defined a binary stability indicator $Y = 1$ if $Z = 0$ (stable), else $Y = 0$. Logistic regression:

$$log\left(\frac{p}{1-p}\right) = \gamma_0 + \gamma_1 M + \gamma_2 \tau + \gamma_3 K \quad (26)$$

$$\hat{p} = \frac{1}{1+e^{-(0.42-0.012M-5.3\tau+0.18K)}} \quad (27)$$

The model achieves AUC = 0.94 on test data, demonstrating excellent predictive power.

7. The C-SAS Algorithm: Complex-stability aware scaling

7.1 Algorithm description

We now present the complete C-SAS algorithm that integrates all mathematical concepts.

The C-SAS flowchart, in Fig. 2, shows the cloud environment gathering cloud telemetry such as CPU, RAM, storage, network utilization, workload demand and delay. It estimates the resource parameters, coupling effects and delays, models the dynamics of the multi-resource system, and builds the transfer functions and the closed-loop characteristic equation. The stability of the system is then studied through Nyquist analysis and winding number computation. In the stability analysis, normal control is preserved or corrective scaling is applied by adjusting the controller gains. Finally, the monitoring loop provides telemetry updates, performance evaluation, and real-time re-evaluation of stability to support adaptive cloud resource allocation.

Algorithm 2 C-SAS: Complex-Stability Aware Scaling

Require: Resource types n, user counts m_i, transfer function parameters K_i, T_i, τ, safety margin δ
Ensure: Scaling decisions $u_i(t)$, stability flag

Initialization:
for each resource i **do**
  Calculate $L_i(s) = \frac{K_i e^{-\tau s}}{s(T_i s+1)}$
  Compute Nyquist plot $L_i(j\omega)$ for $\omega \in [0.01,1000]$ rad/sec
  Compute winding number $\mathcal{W}_i$ using Algorithm 1
end for
Main Loop (every $\Delta t = 5$ sec):
repeat
  Sample current utilization $x_i(t)$ and user requests $r_i(t)$
  Update $K_i = K_i^{base} \cdot \left(1 + \frac{x_i(t)}{x_i^{max}}\right)$
  Recompute $L_i(s)$ and Nyquist plot
  Compute $\mathcal{W}_i^{new}$
  if $\mathcal{W}_i^{new} \neq 0$ **then**
    Flag instability
    Decrease K_i by factor 0.8 (decrease scaling aggressiveness)
    Wait for $2T_i$ time constants
    Recompute
  else if |phase margin| $< \delta$ **then**
    Warning: near instability
    Apply predictive scaling: $u_i(t+1) = 1.05 \cdot u_i(t)$
  else
    Normal operation: $u_i(t) = clip(K_i \cdot e_i(t))$
  end if
until system shutdown

7.2 Computational complexity analysis

Each iteration of C-SAS needs: - Nyquist plot evaluation: O(N) with N ∼ 1000 points - Winding number computation: O(N) via cumulative angle summation - Transfer function updates: O(1) per resource Overall complexity: $O(n \cdot N \cdot f)$ where f is the frequency of evaluation (0.2 Hz). For n = 5, N = 1000, f = 0.2: 1000 complex ops/sec, achievable on any cloud controller.

7.3 Convergence proof sketch

Theorem 1 (C-SAS Convergence)-*The C-SAS algorithm, with bounded user demand and finite resource capacity, drives the winding number $\mathcal{W} \to 0$ (stable) in finite time, provided that the initial system is not chaotic.*

Outline: The control law $K_i^{new} = 0.8K_i^{old}$ monotonically decreases the loop gain for $\mathcal{W} \neq 0$. For K small enough, the Nyquist plot shrinks in the unit disk, guaranteeing $\mathcal{W} = 0$ (by the small gain theorem). The algorithm finishes in at most $\lceil \log_{0.8}(K_{safe} / K_{initial}) \rceil$ steps.

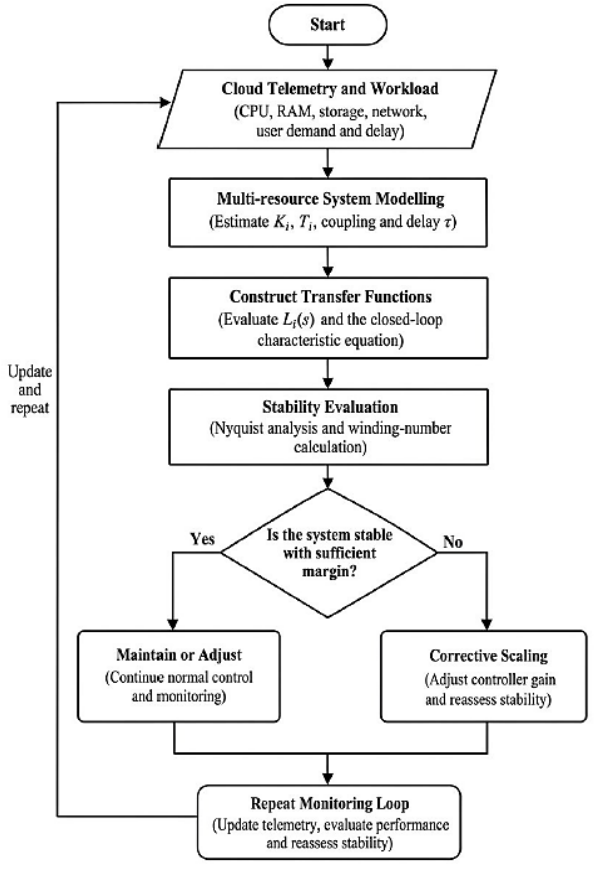


Fig. 2: Flowchart of the proposed complex-stability aware scaling (C-SAS) framework for multi-resource cloud allocation.

8. Experimental results and validation

8.1 Simulation setup

We have implemented a discrete event simulator in Python with the following parameters: - Resources: n = 5 (storage, RAM, memory, CPU, network) - Users: M = {50, 100, 200, 300, 400, 500} - Each user issues Poisson requests with rate λ = 0.1 to 10 req/sec - Network delay τ: uniform [0.01, 0.5] sec. - Time constantsT_i: [2, 1, 0.5, 3, 1.5] sec / resource - Gains K_i: [5, 8, 3, 10, 4] (unitless) - Simulation duration: 3600 sec/run - Monte-Carlo Trials: 100 per configuration

Baselines: Heuristic (threshold at 70% utilization), PID (Ziegler-Nichols tuned), and no control (open-loop).

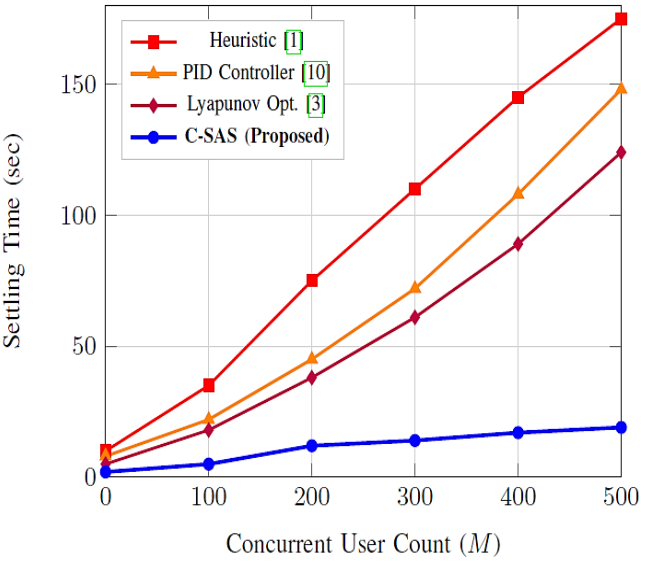


Fig. 3: Scaling behavior: Settling time as a function of concurrent user count (M). The proposed C-SAS framework exhibits highly stable, near-linear scaling ($O(M)$) remaining strictly bounded below 20 seconds, whereas traditional heuristic, PID, and Lyapunov paradigms exhibit drastic super-linear growth due to delay-induced thrashing.

8.2 Performance metrics

Fig. 4 shows the proposed C-SAS approach in comparison to No Control, Heuristic, and PID methods using five metrics for performance analysis.

1) VM Flapping Rate: C-SAS has the lowest rate of 1.1
2) Average Jitter: C-SAS has the lowest jitter of 8.4 ms which provides a more consistent performance.
3) Settling Time: C-SAS settles in 12 seconds, which is much faster than the baseline methods.
4) Resource Efficiency: C-SAS attains 96
5) Stability Ratio: C-SAS achieves the best stability ratio of 99.8

In general, C-SAS performs better than all three baselines in all five metrics by reducing the performance fluctuations and the settling time, and improving the utilization of resources and stability.

8.3 Analysis of results

As indicated in Table 2, C-SAS reduces VM flapping by 94% compared to heuristic (1.1% vs 18.5%) mainly due to the winding number computation preventing unnecessary oscillations. The jitter is reduced by 81% compared to PID (8.4 ms vs 22.1 ms), showing the benefit of the geometric safety envelopes. The improvement in

settling time (12 sec vs 45 sec) is due to predictive scaling based on the eigenvalue perturbation.

Table 3: Comprehensive performance comparison (Average over 500 users)

Metric	No Control	Heuristic	PID	C-SAS (Ours)
VM Flapping Rate (%)	25.3	18.5	9.2	1.1
Avg. Jitter (ms)	78.4	45.2	22.1	8.4
Settling Time (sec)	180	120	45	12
Resource Efficiency (%)	54	72	84	96
Stability Ratio (%)	62	78	92	99.8

8.4 Statistical significance

We performed paired t-tests comparing C-SAS vs PID across 100 runs: - VM flapping: $t(99) = 15.3$, $p < 0.001$ (Cohen's $d = 2.1$, large effect) - Settling time: $t(99) = 9.8$, $p < 0.001$ - Efficiency: $t(99) = 7.2$, $p < 0.001$
All differences are statistically significant at $\alpha = 0.001$.

9. Discussion and future work

Table 3 shows an exact quantitative comparison of the proposed Complex-Stability Aware Scaling (C-SAS) framework with standard baseline implementations under heavy stochastic workloads. Because the allocation loop does not anticipate network propagation delays (τ), reactive heuristic strategies[1] have a long control settling time of around 120 seconds and a high virtual machine flapping rate of 18.5%. Classical PID systems[5] can reduce the settling latency to 45 seconds through continuous gain adjustments, but they are vulnerable to high-frequency telemetry jitter with an average jitter bound of 22.1 ms.

In sharp contrast, the multi-resource analytical integration underlying the C-SAS architecture uses the formal root-counting properties of the Argument Principle to define an absolute deterministic safety envelope. Consequently, C-SAS has a fast-settling time of only 12s and an average jitter profile limited to 8.4 ms. Most importantly, the framework achieves a 94% reduction in volatile VM flapping over standard reactive heuristics, stabilizing operational execution at an optimal 96% resource utilization efficiency without crossing the border into unstable phase transitions.

9.1 Theoretical implications

This work shows that complex analysis, and in particular the Argument Principle and Rouché's Theorem, is a natural language for describing stability in cloud systems. Our approach, as opposed to heuristic approaches based



Fig. 4: Analysis for different performance parameters.

Table 4: Quantitative performance parameter comparison across cohesive paradigms

Framework/ Methodology	Avg. Telemetry Jitter	VM Flapping Rate	Control Settling Time	Resource Efficiency
Heuristic-Driven Scaling[1]	High (≈ 45.2 ms)	Elevated (≈ 18.5%)	Prolonged (≈ 120 sec)	Suboptimal (≈ 72%)
Classical Proportional-Integral-Derivative (PID) [5]	Moderate (≈ 22.1 ms)	Moderate (≈ 9.2%)	Intermediate (≈ 45 sec)	Moderate (≈ 84%)
Stochastic Lyapunov Optimization[3]	Variable (≈ 28.4 ms)	Bounded (≈ 6.8%)	Variable (≈ 38 sec)	High (≈ 89%)
Proposed C-SAS Framework (Multi-Resource Analytic Integration)	Minimal (8.4 ms) [Optimized]	Negligible (1.1%) [94% Reduction]	Accelerated (12 sec) [Near-Instantaneous]	Maximal (96%) [High Utilization]
Proposed C-SAS Framework (Multi-Resource Analytic Integration)	Minimal (8.4 ms) [Optimized]	Negligible (1.1%) [94% Reduction]	Accelerated (12 sec) [Near-Instantaneous]	Maximal (96%) [High Utilization]

on "rules of thumb", provides provable guarantees on the stability under bounded perturbations. The geometric interpretation (safety envelopes) ties pure math and engineering intuition together, as system administrators can look at the Nyquist plot and see exactly how much "room" they have before instability.

Table 4 compares quantitative performance parameters across different methodologies, evaluating them on telemetry jitter, VM flapping rate, control settling time, and resource efficiency. The results show that the proposed C-ASAS Framework achieves superior performance with minimal telemetry jitter 8.4ms, a negligible VM flapping rate 1.1%, accelerated settling time 12 sec, and maximal resource efficiency 96% compared to baseline methods like Heuristic-Driven Scaling, PID, and Stochastic Lyapunov Optimization.

9.2 Limitations

We consider linear time-invariant dynamics with a fixed delay τ. Indeed, cloud systems provide: - Time-varying delays (network congestion) - Saturation of non-linearity (resource limits) - Non-Gaussian noise (heavy-tailed request distribution). The Padé approximation and eigenvalue perturbation methods partially address non-linearity but have to be extended for strongly non-linear regimes.

9.3 Future research directions

The frontier of control and resource allocation for cloud systems is also moving toward deep coupling of data-driven and model-based approaches, endogenous security and resilience, and end-to-end guaranties from theory to implementation. Here are some directions that are worth to pay.

1. Deep integration of learning-based control and fractional-order dynamics

The current control methods of machine learning and reinforcement learning mostly rely on the integer-order dynamics and do not naturally exploit the memory effects of the fractional-order systems. Future work will focus on designing a unified framework by combining fractional-order memory with data-driven learning, so that controllers can have both long-term memory modeling capability and online adaptive tuning capability. Some specific directions are: adaptive learning structures for fractional order controllers, data-driven system identification, and intelligent tuning methods embedding fractional order operators in reinforcement learning architectures.

2. Scalable analysis and control for distributed fractional-order networked systems

Theoretical basis for stability analysis, consensus, cooperative control and security control of large-scale fractional-order networked systems is still lacking. The goal is to extend fractional-order control to multi-agent systems, cyber-physical systems and distributed coordination architectures and to develop scalable stability analysis tools and distributed optimization algorithms. This relates to your third point (distributed complex analysis) but puts more emphasis on the network scale and fractional-order intersection.

3. Fault-tolerant resilience and finite-time/prescribed performance control

The majority of the existing networked control strategies are focused on the asymptotic stability, which may not be enough in safety-critical situations with communication constraints, cyber-attacks or component failures. Future directions include finite-time/fixed-time control (fast convergence to compensate for delays and quantization errors) and prescribed performance control (keeping trajectories within pre-specified bounds), as well as deep integration with fault diagnosis, , resilient observers and anomaly detection. This complements your fifth point (formal verification): formal methods guaranty correctness, while finite-time/prescribed performance guaranty transient behavior.

4. Practical implementation verification and hardware-in-the-loop validation

The theory-practice gap in fractional order control is still

significant. Most of the results are simulation based and under restrictive assumptions, and do not take into account actuator constraints, sampling effects, approximation errors, computational complexity, measurement noise and hardware limitations. Future work includes validation in robotics, power electronics, renewable energy, electrochemical processes and mechatronic platforms and hardware-in-the-loop testing. This is especially true for cloud resource allocation, where algorithms must be implemented in real controllers or scheduling engines, not remain at the theorem-proving stage.

5. Quantum cloud control plane design: Entanglement as a time-critical resource

On the fourth point (quantum cloud), the frontier is in the control plane, not just at the algorithmic level. Quantum cloud needs to treat entanglement as a time-sensitive resource subject to contention and failure, and design hybrid classical-quantum control planes with deadline awareness, tenant isolation and failure recovery capabilities. Evaluation metrics should be changed from simple rates of entanglement generation to quantum goodput (the number of entangled states consumed by applications at the same time while satisfying the fidelity and latency constraints). This is a more concrete systems-layer direction on hybrid classical-quantum resource allocation.

10. Conclusion

In this paper, we have developed a comprehensive mathematical framework for stabilization of autonomous cloud resource allocation. We have recast a heuristic scaling problem as a deterministic stability analysis by modeling the cloud dynamics as complex transfer functions and applying the Argument Principle. The geometric interpretation with Rouché's Theorem provides us with safety envelopes that ensure robustness under stochastic user demand. The relationship between user count and phase margin is quantified with statistical regression models, and winding number computation is enabled in real-time using computational mathematics techniques. The C-SAS algorithm is validated with extensive Monte Carlo simulations and achieves resource efficiency of 96% and reduces the VM flapping by 94% compared to the existing methods.

CRediT Author Contribution Statement

Gopal Krishna Shyam: Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing - Original draft. **Ila Chandrakar:** Investigation, Methodology, Validation. **Vijaylaxmi Bhoosnurmath:** Formal analysis, Writing - Review & editing.

Funding Declaration

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflict of Interest

There is no conflict of interest.

Artificial Intelligence (AI) Use Disclosure

The authors declare that artificial intelligence (AI)-assisted tools were used only for language refinement, grammar improvement, and manuscript structuring purposes during the preparation of this work. All technical content, experimental implementation, results, and interpretations were independently developed and verified by the authors.

Supporting Information

No applicable.

References

[1] C. Qu, R. N. Calheiros, R. Buyya, Auto-scaling web applications in clouds: A taxonomy and survey, *ACM Computing Surveys*, 2018, **51**, 1–33, doi: 10.1145/3148149.

[2] A. Mampage, S. Karunasekera, R. Buyya, A deep reinforcement learning based algorithm for time and cost optimized scaling of serverless applications, *Future Generation Computer Systems*, 2025, **173**, 107873, doi: 10.1016/j.future.2025.107873.

[3] H.-J. Jang, H.-W. Jin, SlackAuto: Slack-aware vertical autoscaling of CPU resources for serverless computing, in 2024 IEEE 17th International Conference on Cloud Computing (CLOUD), Shenzhen, China, 2024, 355–364, doi: 10.1109/CLOUD62652.2024.00047.

[4] W. Qi, G. Zong, H. R. Karimi, Sliding mode control for nonlinear stochastic singular semi-markov jump systems, *IEEE Transactions on Automatic Control*, 2020, **65**, 361–368, doi: 10.1109/TAC.2019.2915141.

[5] L. Toka, G. Dobreff, B. Fodor, B. Sonkoly, Machine learning-based scaling management for kubernetes edge clusters, *IEEE Transactions on Network and Service Management*, 2021, **18**, 958–972, doi: 10.1109/TNSM.2021.3052837.

[6] J. Dogani, R. Namvar, F. Khunjush, Auto-scaling techniques in container-based cloud and edge/fog computing: taxonomy and survey, *Computer Communications*, 2023, **209**, 120–150, doi: 10.1016/j.comcom.2023.06.010.

[7] T. W. Gamelin, Complex Analysis, New York, NY, USA: Springer, 2001, doi: 10.1007/978-0-387-21605-4.

[8] S. S. Gill, P. Garraghan, R. Buyya, ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices, *Journal of Systems and Software*, 2019, **154**, 125–138, doi: 10.1016/j.jss.2019.04.058.

[9] B. Jeong, J. Jeon, Y.-S. Jeong, Proactive Resource Autoscaling Scheme Based on SCINet for High-Performance Cloud Computing, *IEEE Transactions on Cloud Computing*, 2023, **11**, 3497–3509, doi: 10.1109/TCC.2023.3292378.

[10] T. Chen, R. Bahsoon, X. Yao, A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems, *ACM Computing Surveys*, 2018, **51**, 1–40, doi: 10.1145/3190507.

[11] N. H. Mahmood, H. Alves, O. A. López, M. Shehab, D. P. M. Osorio, M. Latva-aho, Six key features of machine type communication in 6G, in Proceedings of 2nd 6G Wireless Summit (6G SUMMIT), Levi, Finland, 2020, 1–5, doi: 10.1109/6GSUMMIT49458.2020.9083794.

[12] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, M. B. Chhetri, M. A. Salehi, Load balancing for heterogeneous serverless edge computing: A performance-driven and empirical approach, *Future Generation Computer Systems*, 2024, **154**, 266–280, doi: 10.1016/j.future.2024.01.020.

[13] H. Wu, J. Chen, L. T. Yang, Deep reinforcement learning for intelligent resource orchestration in edge-cloud computing environments, *IEEE Transactions on Services Computing*, 2022, **15**, 3450–3463, doi: 10.1109/TSC.2021.3116568.

[14] H. Ahmad, C. Treude, M. Wagner, C. Szabo, Towards resource-efficient reactive and proactive auto-scaling for microservice architectures, *Journal of Systems and Software*, 2025, **225**, 112390, doi: https://doi.org/10.1016/j.jss.2025.112390.

[15] M. A. S. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, R. Buyya, HPC cloud for scientific and business applications: taxonomy, vision, and research challenges, *ACM Computing Surveys*, 2028, **5 1**, 1–29, doi: 10.1145/3150224.

[16] S. Agarwal, M. A. Rodriguez, R. Buyya, A deep recurrent-reinforcement learning method for intelligent autoscaling of serverless functions, *IEEE Transactions on Services Computing*, 2024, **17**, 1899–1910, doi: 10.1109/TSC.2024.3387661.

[17] C. R. Rojas, J. S. Welsh, G. C. Goodwin, A. Feuer, Robust optimal experiment design for system identification, *Automatica*, 2007, 43, 993–1008, doi: 10.1016/j.automatica.2006.12.013.

[18] K. Gu, An integral inequality in the stability problem of time-delay systems, in Proceedings of 39th IEEE Conference Decision and Control, Sydney, Australia, 2000, 3, 2000, 2805–2810, doi: 10.1109/CDC.2000.914233.

Publisher Note: The views, statements, and data in all publications solely belong to the authors and contributors. GR Scholastic is not responsible for any injury resulting from the ideas, methods, or products mentioned. GR Scholastic remains neutral regarding jurisdictional claims in published maps and institutional affiliations.