

Deadline Performance and Tenant Fairness Under Security Screening in Cloud Workload Scheduling

Kesavan Sundara Mudaliyar¹, S. Satish Kumar²

¹Department of Computer Science and Engineering, AMET University, Chennai, India

²Associate Professor, AMET University, Chennai, India

Article Info

Article history:

Received Aug, 2025

Revised Aug, 2025

Accepted Aug, 2025

Keywords:

Cloud Computing;
Deadline Performance;
Discrete Event Simulation;
Security Screening;
Tenant Fairness;
Workload Scheduling

ABSTRACT

Cloud applications share execution capacity while imposing different deadlines and security screening requirements. A scheduler can therefore face congestion even when aggregate capacity appears sufficient, particularly when a large tenant reaches its concurrency limit. This study investigates interaction through a reproducible discrete event simulation of screened cloud jobs. Four nonpreemptive policies are compared: first in first out, shortest job first, earliest deadline first, and a deadline policy with a bounded penalty for tenants that have already consumed more virtual service. The primary experiment contains 240 runs covering three arrival rates, 20 random seeds, and 1,200 jobs per run. An additional 80 runs remove the restrictive tenant concurrency limit at the highest arrival rate. With eight workers and a four-worker tenant limit, shortest job first has a mean deadline miss rate of 5.26% at 1.5 jobs per simulated second, compared with 44.57% for first in first out and 23.97% for earliest deadline first. The service balance candidate reaches 28.91%, showing that the added fairness penalty does not improve deadline performance in this workload. Increasing the tenant limit from four to eight reduces the earliest deadline miss rate to 1.41%. These findings identify tenant capacity fragmentation and job length variation as stronger influences than the tested fairness adjustment. The results are generated from synthetic workloads and do not measure production cloud performance or resistance to cyberattacks. The contribution is an executable comparative study of scheduling under a mandatory modeled screening stage, with explicit workload assumptions and a tenant capacity sensitivity analysis.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Name: Kesavan Sundara Mudaliyar

Institution: Department of Computer Science and Engineering, AMET University, Chennai, India

Email: kesevan2007@gmail.com

1. INTRODUCTION

A cloud platform may have idle workers while a customer's job continues to wait. That situation arises when the platform has imposed a tenant concurrency limit to

protect other customers or contain the consequences of a compromised workload. A scheduler cannot assign another worker to that tenant, even if no other customer currently needs the available capacity. The resulting delay is a resource allocation

problem with a security related constraint. Its consequences depend on the tenant's demand, the duration of its jobs, and the order in which eligible requests are selected.

Security mechanisms also consume execution time. A submitted task may require content screening or configuration validation before the application stage begins. In this study, screening is represented as mandatory service time that occupies the same worker as the application. This is a deliberately simplified arrangement. It captures the cost of a serial screening stage without claiming to reproduce any vendor product, actual malware scanner, or network intrusion detector. The screening stage is identical across policies so that the comparison evaluates scheduling rather than differences in protection.

The broader cloud data protection setting includes enforcement, monitoring, and governance. [1] provides a relevant architectural context for placing security checks within cloud workflows. The question here is different from whether an encrypted object should be released to an authorized principal. It concerns which eligible job should receive limited execution capacity next, after the platform has accepted the task into its modeled screening pipeline.

The research question is how scheduling policy and tenant concurrency limits affect deadline misses, tail waiting time, and variation in tenant slowdown when cloud tasks require security screening. A second question asks whether adding a bounded service balance penalty to a deadline policy improves those outcomes. These questions are addressed with executed simulations and a sensitivity analysis. No encryption scheme, key broker, access receipt protocol, or decryption authorization model is proposed.

The paper contributes a fully specified synthetic workload, a comparative experiment with matched random seeds, and an analysis of a tenant bottleneck that aggregate utilization alone fails to reveal. A further contribution is the reporting of a negative result for the tested service balance policy. The paper does not claim that the policy is globally novel or optimal. Its value

lies in the reproducible comparison, and the explanation of why a plausible fairness adjustment can perform poorly under a restrictive tenant limit.

2. RESEARCH CONTEXT

2.1 *Security Controls and Operational Capacity*

Zero trust separates the existence of a connection from the right to perform an operation. [2] supply the conceptual setting for explicit access decisions. In the present study, access eligibility is assumed before a job enters the execution queue. Scheduling can change when an accepted task runs, but it cannot override its required screening stage. The modeled protection is therefore held constant while execution order changes.

Application and service identities matter across cloud environments, as described by [3]. A tenant label in this simulation is a trusted scheduling attribute rather than an unverified claim supplied by the caller. A real platform would have to bind that label to its identity and authorization system. That requirement is outside the experiment and is not validated by the simulated concurrency counter.

Key lifecycle management remains an independent cloud responsibility [4]. It supplies an example of operational security work that must be planned alongside computation. This study does not model key rotation, encryption overhead, or cryptographic operations. The screening time parameter is an abstract cost and should not be interpreted as a measured estimate of those other controls.

AI can support security related judgments, but that possibility should be separated from execution scheduling [5]. The simulation assigns two screening durations using a random workload flag. The flag does not come from a trained model and has no evaluated detection

quality. Neither predictive accuracy nor explainability is reported as an outcome.

Transformer and LSTM threat detection research addresses recognition of malicious behavior [6]. This paper uses no architecture. Its connection to that work is operational: a future screening stage could contain a detector whose inference time affects task service duration. Substituting such a detector would require measured runtime and error characteristics, rather than simply adopting its published detection metrics.

Advanced cyber threats motivate limits on how much activity a single tenant can launch [7]. A concurrency ceiling is one possible containment constraint, but it cannot establish that a workload is benign. The experiment examines the delay created by the ceiling. It does not demonstrate that the chosen ceiling of four workers is appropriate against a particular attacker.

2.2 *Analytics and Governance Workloads*

Data governance connects organizational rules with analytical activity [8]. A scheduling platform must likewise distinguish a valid task from a task with a justified completion target. Our simulator assumes that deadlines are assigned correctly and does not optimize their business value. The difference between technical completion and useful organizational work remains important when interpreting a deadline miss.

Cloud based project management creates tasks with varied duration and collaboration needs [9]. This provides practical motivation for both short and bulk jobs. The mixture used here is synthetic; it is not extracted from that study or a project management deployment. Its purpose is to make service heterogeneity explicit rather than to represent a particular organization.

Stakeholder collaboration in cloud information systems provides another setting in which several users compete for shared execution resources [10]. The model represents competition

through tenant arrivals, not through survey responses or collaboration outcomes. Faster execution may be useful for such workflows, but no claim about project success follows directly from the simulation.

DataOps governance concerns the movement of data through automated pipelines [11]. In a deployment, a screening stage would have to preserve those pipelines' required ordering and lineage. Here, each job is independent and has one serial screening and application service interval. Dependencies between tasks are excluded, which limits transfer to complex workflow graphs.

Federated and blockchain supported lakehouse design supplies a distributed analytics context [12]. Its use in this paper is limited to motivating the existence of shared analytical tasks. We do not simulate federation rounds, blockchain consensus, or lakehouse queries. Those activities would produce different duration distributions and should be measured before applying the numerical results to that setting.

2.3 *Distributed Applications and Security Telemetry*

Federated threat intelligence can create distributed computation without centralizing every raw record [13]. Such jobs may place uneven demand on participating cloud tenants. The present workload uses unequal arrival shares for that reason, while keeping the task duration distribution common across tenants. It makes no assertion about the volume or timing of real federated training jobs.

Homomorphic federated models introduce computational requirements that differ from ordinary application tasks [14]. They are a potential future workload class, not an implemented component here. A valid extension would measure their service durations and replace the synthetic bulk distribution. The current short and bulk

labels do not reproduce encrypted computation.

Distributed healthcare analytics is another context in which local workloads and shared computation must be coordinated [15]. Clinical urgency and privacy requirements would require domain specific policies. The deadlines in this simulation are generated algebraically and must not be interpreted as patient care deadlines or evidence of clinical benefit.

Big data based threat detection in management information systems motivates timely processing of telemetry [16]. A delayed analytical job can delay a downstream response, but the present study measures task completion alone. It does not simulate attack progression or infer a reduction in incident response time from a shorter queue.

AI, information systems, and cloud frameworks also appear within broader cyber defense work [17]. That perspective helps explain why execution capacity and security operations need to be considered together. The experiment isolates one operational relationship and avoids combining throughput, fairness, and attack detection into a single undefined resilience score.

3. SYSTEM AND WORKLOAD SPECIFICATION

3.1 Shared Workers and Tenant Limits

The platform contains eight identical workers and four tenants. Each worker executes at most one job at a time. Jobs are nonpreemptive: after dispatch, they occupy a worker until screening and application execution finish. Workers are interchangeable, and service duration is independent of which worker is assigned. Network transfer, cold start, memory pressure, and processor contention are excluded.

In the primary experiment, no tenant may occupy more than four workers simultaneously. A task becomes eligible for dispatch when its tenant has

fewer than four running jobs and a worker is free. Every tested policy uses the same eligibility rule. The sensitivity experiment changes the tenant limit to eight, allowing one tenant to use the whole platform when sufficient workers are free. This removes the restrictive concurrency ceiling; it is not asserted to be a safe production setting.

The queue has no fixed size and no dropped jobs. The simulator continues after the last arrival until every job is completed. Consequently, the experiment describes a finite arrival episode followed by queue drainage. It does not directly estimate stationary performance, prove stability, or represent a service that receives jobs indefinitely. This distinction is particularly important when a tenant's offered work exceeds its permitted concurrency.

3.2 Arrival and Service Generation

Each run contains 1,200 jobs. Interarrival times follow an exponential distribution with rate 0.7, 1.1, or 1.5 jobs per simulated second. Tenant assignment uses probabilities 0.55, 0.25, 0.15, and 0.05. These shares intentionally make tenant zero the largest source of demand. They are analyst specified parameters rather than measurements of cloud customers.

Seventy percent of jobs are drawn from the short class and the remainder from the bulk class. Application durations follow exponential distributions with respective means of two and ten simulated seconds before clipping. Durations are below at 0.2 seconds and above at 30 seconds. Clipping reduces the influence of extremely long tails and must be considered when applying the results to workloads with longer tasks.

An independent flag selects the longer screening stage with probability of 0.25. Screening occupies 1.5 seconds for flagged jobs and 0.4 seconds for the others. Total service duration is the sum of application and screening duration.

The flag represents duration heterogeneity, not an observed probability of malicious activity. All tasks complete their required modeled screening stage before the application interval ends.

The mean total service duration across the 20 generated workloads is 4.962 simulated seconds. Expected screening cost under the specified mixture is 0.675 seconds, or approximately 13.6% of that observed mean total service. This is an accounting property of the synthetic workload. Because no experiment removes screening, it should not be treated as a measured causal estimate of the delay attributable to screening.

3.3 *Deadlines and Available Information*

A short job receives a completion budget of five seconds plus twice its total service duration. A bulk job receives ten seconds plus 2.5 times total service. Its absolute deadline equals arrival time plus this budget. These rules give longer tasks more time and ensure an isolated task can meet its deadline. They also make deadline urgency partly dependent on duration, which affects the comparative advantage of the scheduling policies.

The simulator provides exact total service duration to the policies. Shortest job first therefore operates with oracle information. Earliest deadline first also benefits indirectly because deadlines were generated from exact service. The experiment is a controlled comparison under known durations, not evidence that production schedulers can obtain equally accurate estimates. Duration prediction errors remain an untested source of uncertainty.

3.4 *Capacity Interpretation*

Aggregate offered utilization is calculated as arrival rate times which mean service duration divided by eight workers. The three tested rates yield approximately 43.4%, 68.2%, and 93.0% nominal utilization. The same calculation for tenant zero divides its demand by its

four-worker limit. At the highest rate, 1.5 multiplied by 0.55 multiplied by 4.962 is approximately 4.094 worker equivalents, exceeding its allowed four workers.

This calculation explains why the busiest scenario can produce a growing tenant queue despite aggregate offered demand remaining below eight workers. It is a workload level diagnosis rather than a measured stationary queue growth rate. Finite simulation and sampling variation still matter. A platform report that shows only aggregate utilization could conceal this tenant specific constraint.

4. SCHEDULING POLICIES

4.1 *Reference Policies*

First in first out selects the eligible task with the earliest arrival time. This policy is simple and does not exploit duration or deadline information. In a heterogeneous queue, a long task can be dispatched before a short task that arrived slightly later. Because running tasks cannot be preempted, subsequent arrivals must wait for worker completion even if their completion budgets are much shorter.

Shortest job first selects the eligible task with the smallest total service duration. It uses the exact duration supplied by the simulation. Earliest deadline first selects the task with the earliest absolute completion deadline. Both use the job identifier to resolve ties. The policies remain work conserving within the shared eligibility rule: if a worker is free and an eligible task exists, one is dispatched.

4.2 *Deadline and Service Balance Candidate*

The candidate policy begins with earliest deadline ordering and adds a penalty for tenants that have accumulated more normalized dispatched service. Let S_i be the total service duration already dispatched for tenant i and let w_i denote its configured weight. The weights are 0.4, 0.3, 0.2, and 0.1. They differ from arrival shares

because they represent an intentionally chosen service preference, not measured demand.

At each dispatch, the policy identifies tenants with eligible waiting jobs and obtains the minimum normalized service S_i divided by w_i among those tenants. A candidate's lead is its tenant's normalized service minus that minimum. Its priority score is remaining time to deadline plus the smaller of 20 seconds and 0.15 times the lead. The job with the lowest score runs next. The bound penalty prevents virtual service from completely replacing deadline urgency.

This is a specified experimental heuristic. The coefficient 0.15, the penalty ceiling, and the tenant weights were fixed before examining the reported results and were not tuned to achieve a favorable outcome. They are not claimed to be optimal. The policy also uses cumulative service without decay, which can preserve the influence of earlier activity even after the active queue changes. That design choice helps explain why its fairness objective need not align with the measured slowdown gap.

The score changes execution order only among eligible jobs. It cannot bypass the tenant limit, preempt a running task, or omit screening. Cumulative accounting is updated at dispatch rather than completion so that already committed service affects the next decision. The mechanism seeks service balance, while the outcome metric measures delay balance; these are related but distinct concepts.

5. EXPERIMENTAL METHOD

5.1 *Runs and Matched Workloads*

The primary design crosses three arrival rates with four policies and 20 random seeds, yielding 240 runs and 288,000 job executions. For each seed and rate, all four policies receive identical jobs. Matching workloads permits

differences to be attributed to scheduling choices within the model, rather than differences in random input. Seeds range from zero to 19 and use Python's standard pseudorandom generator.

The same seed also generates the same service and tenant sequence across arrival rates; only interarrival time scaling changes. Thus, rate comparisons use related workloads and are not independent samples of different customers. The sensitivity study adds 80 runs at rate 1.5 with an eight-worker tenant limit, using the same four policies and 20 seeds. The complete study contains 320 simulation runs and 384,000 job executions, representing 72,000 primary generated arrivals replayed across conditions rather than that of many independently observed customers.

5.2 *Event Processing*

The simulator advances to the next arrival or completion. At a time, boundary it removes completed tasks, admits all arrivals up to that time, and dispatches eligible jobs while capacity remains. Completion events are stored in a priority queue. Waiting jobs are selected using the policy score and removed from the pending list. Tenant counters increase at dispatch and decrease at completion.

A deterministic order is used when events that coincide with completions are handled before arrivals, and dispatch follows both. A tolerance of one billionth of a simulated second handles floating point comparison. Although exact event collisions are uncommon under continuous distributions, making the ordering explicit avoids an undocumented source of variation. The implementation checks for a deadlock if jobs remain, but no event can advance the clock.

5.3 *Outcome Definitions*

Deadline miss rate is the percentage of jobs whose completion exceeds their absolute deadline. Waiting time is dispatch time minus arrival time.

The 95th percentile uses the nearest rank convention within each run. Slowdown is total response time divided by total service time; a job that starts immediately has slowdown one. The reported mean slowdown averages jobs within a run and then averages run level values across seeds.

Tenant slowdown gap is the largest tenant means slowdown minus the smallest in each run. It measures disparity in experienced delays, not security, equal CPU allocation, or achievement of the configured service weights. A small gap can coexist with uniformly poor service, so it must be interpreted beside deadline misses and waiting time. All tenant groups contain jobs in the generated workloads.

Observed utilization equals total service across all jobs divided by eight times the elapsed interval from first arrival to last completion. It includes the drainage period. Throughput uses completions divided by the same interval. Neither measure is stationary throughput. Different policies can alter the drainage duration and consequently produce different observed utilization even though the total executed work is identical.

5.4 *Summary and Uncertainty*

Tables report the mean of each run level outcome across 20 seeds. Where dispersion is shown, it is the sample standard deviation of the run level deadline miss percentage. The sensitivity study uses paired differences between the four and eight worker tenant limits. Its approximate 95% confidence interval uses the mean paired difference plus or minus 2.093 times its sample standard deviation divided by the square root of 20.

These intervals quantify sensitivity to the selected random seeds

under the specified model. They do not quantify external validity or uncertainty about the real cloud workload distribution. No significance test is used to declare a universally best scheduler. The interpretation focuses on the numerical pattern, the tenant bottleneck calculation, and the limits of the workload assumptions.

6. RESULTS

6.1 *Performance Across Arrival Rates*

At 0.7, all policies achieve mean deadline miss rates below one percent. First in first out reaches 0.40%, while shortest job first and earliest deadline first reach 0.13% and 0.11%. The service balance policy reaches 0.11% after rounding. In this lightly loaded scenario, there is little deadline benefit to justify the candidate's additional bookkeeping. Tail waiting times range from approximately 1.06 to 1.55 seconds.

At rate 1.1, policy differences become clearer. First in first out misses 5.57% of deadlines on average and has a mean within run 95th percentile waiting time of 9.48 seconds. Shortest job first lowers these outcomes to 0.73% and 5.05 seconds. Earliest deadline first has the smallest deadline miss rate, 0.46%, while its tail waiting time is slightly higher at 5.37 seconds. The candidate reaches 0.60% and 5.68 seconds.

At rate 1.5, shortest job first yields the lowest mean deadline miss rate, 5.26%, and the lowest mean tail waiting time, 17.11 seconds. First in first out reaches 44.57% and 52.38 seconds. Earliest deadline first reaches 23.97% and 43.22 seconds, while the service balance candidate reaches 28.91% and 50.08 seconds. Thus, deadline ordering alone is insufficient to control congestion in the busiest constrained scenario.

Table 1. Mean outcomes across 20 seeds with a four-worker tenant limit

Arrival Rate	Policy	Miss Rate %	95th Percentile Wait Seconds	Mean Slowdown	Tenant Slowdown Gap
0.7	FIFO	0.40	1.55	1.12	0.20
0.7	Shortest job	0.13	1.06	1.07	0.12
0.7	Earliest deadline	0.11	1.12	1.07	0.12
0.7	Service balance	0.11	1.13	1.08	0.13
1.1	FIFO	5.57	9.48	1.86	1.30
1.1	Shortest job	0.73	5.05	1.29	0.41
1.1	Earliest deadline	0.46	5.37	1.33	0.47
1.1	Service balance	0.60	5.68	1.35	0.54
1.5	FIFO	44.57	52.38	9.98	13.25
1.5	Shortest job	5.26	17.11	1.81	1.00
1.5	Earliest deadline	23.97	43.22	4.63	5.81
1.5	Service balance	28.91	50.08	5.66	7.82

Waiting times and slowdowns are summarized within each run before averaging across seeds. Slowdown and its gap are dimensionless. At rate 1.5, deadline miss standard deviations are 11.20, 1.46, 12.41, and 12.11 percentage points for FIFO, shortest job, earliest deadline, and service balance, respectively.

The shortest job result is consistent with the workload's large, short task share and non-preemptive execution. Dispatching shorter tasks frees workers more often and protects many short budgets. It also uses exact durations, which makes the comparison favorable to that policy. The aggregate miss rate does not establish that long jobs receive acceptable service; job class specific tail performance would be needed before recommending the policy for deployment.

6.2 Tenant Delay Disparity

The tenant slowdown gap at rate 1.5 is 13.25 under first in first out, 1.00 under shortest job first, 5.81 under earliest deadline first, and 7.82 under the candidate. The corresponding overall mean slowdowns are 9.98, 1.81, 4.63, and 5.66. Shortest job first therefore improves both aggregate delay and this disparity

measure in the tested configuration. This finding does not establish fairness according to every possible allocation objective.

The candidate underperforms earliest deadline first on the measured gap as well as deadline misses. A likely mechanism is the mismatch between service weights and arrival shares. Tenant zero supplies 55% of arrivals but has a configured weight of 40% and a hard limit of four workers. Penalizing its accumulated service can increase an existing queue bottleneck rather than correct it. This is an interpretation of the specified mechanism, not an independently identified causal effect of each score parameter.

6.3 Tenant Limit Sensitivity

Increasing the tenant limit from four workers to eight reduces every policy's deadline miss rate at the highest arrival rate. First in first out falls from 44.57% to 25.30%. Shortest job first falls from 5.26% to 1.41%, and earliest deadline first falls from 23.97% to 1.41%. The candidate falls from 28.91% to 3.45%. Its remaining disadvantage indicates that removing the capacity constraint does not make the tested service penalty beneficial.

Table 2. Tenant limit sensitivity at 1.5 jobs per simulated second

Policy	Miss Rate with Limit of Four %	Miss Rate with Limit Eight %	Reduction Percentage Points	Approximate 95 Percent Paired Interval
FIFO	44.57	25.30	19.26	12.99 to 25.54
Shortest job	5.26	1.41	3.85	3.06 to 4.64
Earliest deadline	23.97	1.41	22.56	16.93 to 28.20
Service balance	28.91	3.45	25.46	19.93 to 30.99

For earliest deadline first, the mean paired reduction is 22.56 percentage points, with an approximate 95% interval from 16.93 to 28.20 points. Its tenant slowdown gap declines from 5.81 to 0.13. The result is consistent with removal of the identified tenant capacity

bottleneck. It does not show that eliminating tenant limits is operationally desirable; the sensitivity condition omits the containment value that a real operator may assign to the restrictive ceiling.

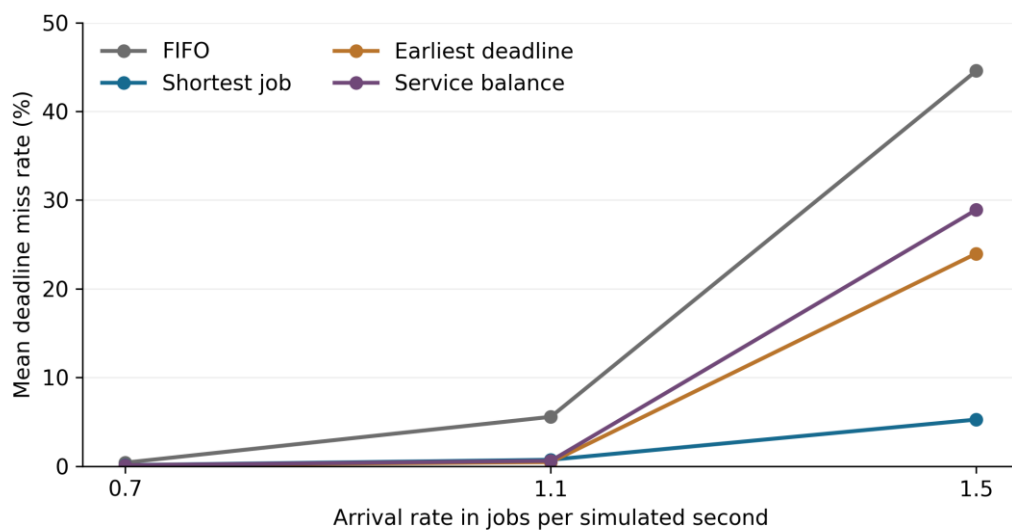


Figure 1. Deadline misses across arrival rates under the four-worker tenant limit

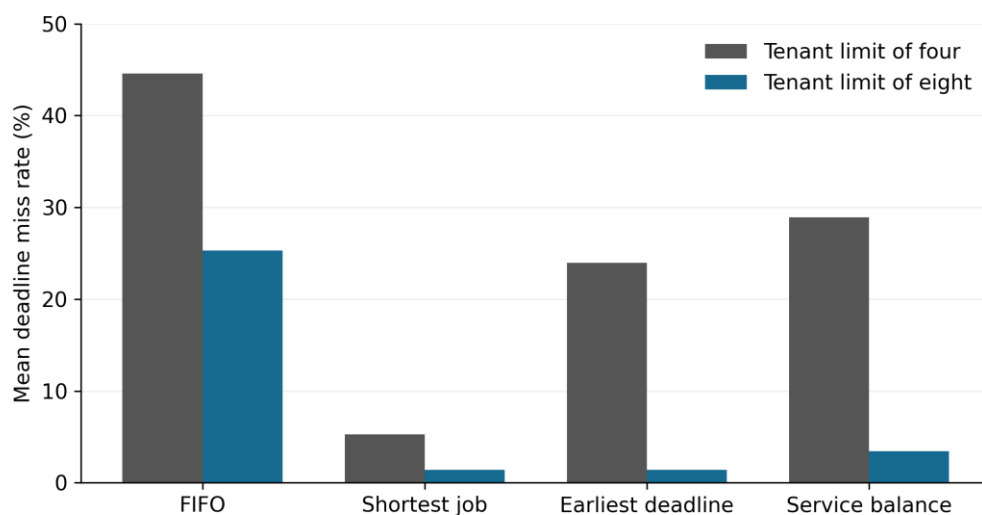


Figure 2. Deadline misses under alternative tenant limits at the highest arrival rate
 Figures show means across 20 seeds and synthetic workload outcomes. No measured cloud deployment performance is represented.

7. Discussion

7.1 Capacity Fragmentation and Security Operations

The central result is that a tenant ceiling can become the binding capacity constraint before the platform reaches its aggregate limit. The highest arrival rate generates approximately 4.094 worker equivalents for the largest tenant, slightly exceeding its ceiling of four. Other tenants cannot absorb that tenant's work. This makes spare platform capacity compatible with substantial waiting and explains why increasing the ceiling changes outcomes more than the candidate's ordering adjustment.

Cybersecurity operations in critical infrastructure combine detection and response requirements [18]. A workload scheduler serving those operations should therefore expose tenant queue pressure alongside platform utilization. Our results provide a reason to measure both, but do not assign criticality to a particular simulated task. An actual operator would need task specific requirements before changing limits.

Supply chain security research connects vulnerabilities with effects outside the originating information system [19]. This creates a reason to examine timeliness of supporting analytics. The simulation does not estimate monetary loss or supply chain disruption. A deadline miss here is a technical outcome under an artificial budget, and its economic significance remains undefined.

Cloud and big data frameworks for infrastructure protection similarly motivate coordination across workloads [20]. The present findings suggest that coordination requires identifying where demand is constrained. An aggregate capacity dashboard cannot by itself distinguish a global shortage from a tenant ceiling. That distinction affects whether the appropriate response is adding workers, revising admission, or

negotiating the tenant's execution budget.

7.2 Screening Cost and Deployment Choices

Real time attack detection and proactive mitigation motivate timely analytical execution [21]. This paper holds the screening requirement constant and varies dispatch. It therefore avoids treating omitted security work as a performance improvement. However, because the simulated screening stage occupies an application worker, a separate screening pool could yield different contention patterns and should be tested as a distinct architecture.

Green and secure data center research makes resource overhead relevant to cloud design [22]. The simulation reports worker occupancy rather than electrical energy. Identical worker seconds need not imply identical power consumption, especially with different idle periods or hardware. No carbon or energy saving is inferred from the reported utilization figures.

IT project security research supplies a setting for maintaining operational controls under changing demand [23]. A deployment of the candidate would require per tenant accounting and policy configuration in addition to deadline data. Since the candidate fails to improve the measured outcomes here, that complexity is not supported by this experiment. It may warrant further study under different weights or workloads, but the present results do not justify a broad adoption recommendation.

7.3 Limits of Transferring Results Across Applications

Cyber threat intelligence within management information systems includes governance as well as detection [24]. In such an environment, the organization must decide whether a completion deadline is advisory or mandatory and who can change it. The simulation assumes fixed deadlines and does not model adversarial claims of urgency. A scheduler that accepts

arbitrary user supplied deadlines could be manipulated in ways absent from this study.

Cybersecurity training and employee behavior also affect operational practice [25]. Tenant administrators could misestimate job duration or misclassify a task when submitting it. The simulator trusts both duration and class. A useful deployment experiment should introduce estimation errors and record whether mistakes systematically disadvantage particular tenants.

Smart building sensor workloads provide an IoT application with potentially different traffic patterns [26]. Device streams can be periodic or correlated, whereas this experiment uses exponential arrivals and independent jobs. Applying the numerical findings to a sensor platform without measuring its arrival and service distributions would exceed the evidence. The present study does not evaluate devices, attacks, or intrusion detection.

National threat intelligence systems provide another setting with distributed analytical demand [27]. They may contain incident driven bursts and tasks with dependencies. Those properties are absent here. The result about tenant ceilings is a testable architectural concern for such systems, while the reported percentages remain specific to the synthetic benchmark.

Blockchain integration in information systems concerns data integrity and decentralized records [28]. Such activity could introduce confirmation delays into a job pipeline, but no ledger is simulated in this study. The receipt and encryption architecture from the earlier manuscript is not part of the scheduling model. Any later integration should measure its actual computational and waiting requirements separately.

7.4 Threats to Validity

The main internal validity concern is not random input imbalance,

because policies share matched jobs. It is the model specification itself. Exact duration knowledge, service dependent deadlines, clipping of long jobs, and nonpreemptive workers favor particular ordering strategies. The proposed penalty has one coefficient and one weight vector; its failure does not establish that every fairness aware scheduler fails.

The fairness metric is another limitation. Mean slowdown gap emphasizes experienced delays and can hide rare extreme waits. The study does not report per class starvation measures, weighted resource entitlement error, or worst case response time. Those outcomes may change the policy ranking. Shortest job first can delay long work in ways an aggregate deadline percentage does not adequately describe.

External validity is restricted to the generated arrival episodes. There is no trace from a real cloud service, no performance measurement on a provider platform, and no verification of scanning effectiveness. The experiment executes an accounting model of mandatory screening; it is not an attack detection experiment. The same 28 references are used for context and interpretation, without importing their numerical findings as benchmark results.

8. CONCLUSION

This study compares cloud workload scheduling under a mandatory modeled security screening stage and tenant concurrency limits. Across 320 executed simulations, the main constrained scenario shows that shortest job first achieves lower aggregate deadline misses and waiting time than the other tested policies. The deadline and service balance candidate does not improve either deadline performance or measured tenant slowdown disparity. A tenant limit sensitivity analysis reveals a stronger influence: the largest tenant exceeds its permitted capacity even when total offered

platform demand remains below eight workers. The practical implication is to evaluate tenant queue pressure and task duration variation before adding an allocation penalty to a deadline scheduler. These

findings are finite synthetic workload results. Deployment decisions require measured task distributions, duration estimation error, workload criticality, and the operational purpose of tenant limits.

REFERENCES

- [1] P. Chakraborty *et al.*, "An Intelligent Cybersecurity Framework for Data Protection in Cloud Computing Environments," *Spectr. Eng. Sci.*, vol. 2, no. 3, pp. 657–677, 2024, doi: <https://thesesjournal.com/index.php/1/article/view/3573>.
- [2] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," *NIST Spec. Publ.*, vol. 800, no. 207, pp. 1–52, 2020, doi: <https://doi.org/10.6028/NIST.SP.800-207>.
- [3] R. Chandramouli and Z. Butcher, "A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments," National Institute of Standards and Technology, 2023. doi: <https://doi.org/10.6028/NIST.SP.800-207Ahttps://doi.org/10.6028/NIST.SP.800-207A>.
- [4] E. Barker, "Recommendation for Key Management: Part 1—General," National Institute of Standards and Technology, 2020. doi: <https://doi.org/10.6028/NIST.SP.800-57pt1r5>.
- [5] S. N. Hasan *et al.*, "The influence of artificial intelligence on data system security," *Int. J. Comput. Exp. Sci. Eng.*, vol. 11, no. 3, 2025, doi: <https://doi.org/10.22399/ijcesen.3476>.
- [6] J. Kaur *et al.*, "Comparative Analysis of Transformer and LSTM Architectures for Cybersecurity Threat Detection Using Machine Learning," *EAI Endorsed Trans. AI Robot.*, vol. 4, no. 1, 2025, doi: <https://doi.org/10.4108/airo.9759>.
- [7] J. Kaur *et al.*, "Advanced Cyber Threats and Cybersecurity Innovation-Strategic Approaches and Emerging Solutions," *J. Comput. Sci. Technol. Stud.*, vol. 5, no. 3, pp. 112–121, 2023, doi: <https://doi.org/10.32996/jcsts.2023.5.3.9>.
- [8] M. A. R. Chy *et al.*, "Understanding the Relationship Between Data Governance and Business Analytics Success: A Case Study of Global Corporations," *J. Inf. Syst. Eng. Manag.*, vol. 9, no. 4s, 2024, doi: <https://doi.org/10.52783/jisem.v9i4s.14807>.
- [9] F. Mahmud *et al.*, "Big data and cloud computing in IT project management: A framework for enhancing performance and decision-making," *Fuel Cells Bull.*, vol. 2023, no. 9, pp. 1–18, 2023, doi: <https://doi.org/10.52710/fcb.166>.
- [10] F. Mahmud, M. A. Goffer, H. Rahman, and G. T. Alam, "The Role of Cloud-Based Management Information Systems in Enhancing IT Project Governance and Stakeholder Collaboration," 2025, doi: <https://doi.org/10.63332/joph.v5i4.974>.
- [11] S. M. Orthi *et al.*, "DataOps-oriented big data governance for automated decision pipelines," in *In 2025 1st International Conference on Advancement in Futuristic Technologies (ICAFT) (pp. 1–8)*. IEEE, 2025, pp. 1–8. doi: <https://doi.org/10.1109/ICAFT66710.2025.11452860>.
- [12] P. Chakraborty *et al.*, "Trustworthy data lakehouse design using federated learning and blockchain," in *2025 1st International Conference on Advancement in Futuristic Technologies (ICAFT)*, 2025, pp. 1–8. doi: <https://doi.org/10.1109/ICAFT66710.2025.11453041>.
- [13] M. K. Tuhin *et al.*, "Federated Machine Learning for Privacy-Preserving Cyber Threat Intelligence in Global Cloud-Integrated MIS Platforms," *Eastasouth J. Inf. Syst. Comput. Sci.*, vol. 2, no. 2, pp. 284–293, 2024, doi: <https://doi.org/10.58812/esiscs.v2i02.1123>.
- [14] N. Das *et al.*, "AI-enhanced privacy preservation using homomorphic federated models," in *In 2025 1st International Conference on Advancement in Futuristic Technologies (ICAFT) (pp. 1–8)*. IEEE, 2025, pp. 1–8. doi: <https://doi.org/10.1109/ICAFT66710.2025.11453096>.
- [15] S. M. Orthi *et al.*, "Federated learning with privacy-preserving big data analytics for distributed healthcare systems," *J. Comput. Sci. Technol. Stud.*, vol. 7, no. 8, pp. 269–281, 2025, doi: <https://doi.org/10.32996/jcsts.2025.7.8.31>.
- [16] S. N. Hasan *et al.*, "Enhancing Cybersecurity Threat Detection and Response Through Big Data Analytics in Management Information Systems," 2023, doi: <https://doi.org/10.52710/fcb.137>.
- [17] M. D. Hossain, M. S. Sikder, M. S. Uddin, R. M. Ahsan, B. Uddin, and T. Hossen, "Cognitive Cyber Defense: AI-MIS Integration through Big Data and Cloud Frameworks for Next-Generation Digital Resilience," *Eastasouth J. Inf. Syst. Comput. Sci.*, vol. 1, no. 2, pp. 140–152, 2023, doi: <https://doi.org/10.58812/esiscs.v1i02.764>.
- [18] M. A. Goffer *et al.*, "AI-Enhanced Cyber Threat Detection and Response Advancing National Security in Critical Infrastructure," *J. Posthumanism*, vol. 5, no. 3 SE-, pp. 1667–1689, Apr. 2025, doi: <https://doi.org/10.63332/joph.v5i3.965>.
- [19] M. A. Goffer *et al.*, "Cybersecurity and Supply Chain Integrity: Evaluating the Economic Consequences of Vulnerabilities in US Infrastructure," *J. Manag. World*, vol. 2, pp. 233–243, 2025, doi: <https://doi.org/10.53935/jomw.v2024i4.907>.
- [20] M. S. Uddin, M. S. Sikder, M. M. Anwar, and F. Hossain, "AI-Driven Cybersecurity and Big Data-Enabled MIS Frameworks: Strengthening Supply Chain Integrity, Energy Resilience, and Critical Infrastructure Protection," *J. Comput. Sci. Technol. Stud.*, vol. 7, no. 9, pp. 223–232, 2025, doi: <https://doi.org/10.32996/jcsts.2025.7.9.26>.
- [21] S. Sultana *et al.*, "AI-Augmented Big Data Analytics for Real-Time Cyber Attack Detection and Proactive Threat

- Mitigation," *Int. J. Comput. Exp. Sci. Eng.*, vol. 11, no. 3 SE-Research Article, Jul. 2025, doi: <https://doi.org/10.22399/ijcesen.3564>.
- [22] M. D. Hossain, M. S. Uddin, M. S. Sikder, T. Hossen, B. Uddin, and R. M. Ahsan, "Green and Secure Data Centers: Balancing Energy Efficiency with Advanced Cybersecurity Measures," *J. Comput. Sci. Technol. Stud.*, vol. 6, no. 5, pp. 300–315, 2024, doi: <https://doi.org/10.32996/jcsts.2024.6.5.24>.
- [23] F. Mahmud *et al.*, "AI-Driven Cybersecurity in IT Project Management: Enhancing Threat Detection and Risk Mitigation," *J. Posthumanism*, vol. 5, no. 4 SE-, pp. 23–44, Apr. 2025, doi: <https://doi.org/10.63332/joph.v5i4.974>.
- [24] S. M. Orthi *et al.*, "AI-Driven Cyber Threat Intelligence as a Management Information System: Integrating Cybersecurity Governance and IT Project Management for Organizational Resilience," *Eastasouth J. Inf. Syst. Comput. Sci.*, vol. 1, no. 2, pp. 194–213, 2023, doi: <https://doi.org/10.58812/esiscs.v1i02.873>.
- [25] A. Shan-A-Alahi, M. Mustafizur, K. M. R. Hossan, A. Al Zaiem, and M. M. Rahman, "Cybersecurity Training and Its Influence on Employee Behavior in Business Environments," *Comput. Fraud Secur.*, vol. 2024, no. 12, 2024, doi: <https://doi.org/10.52710/cfs.689>.
- [26] M. A. Siam *et al.*, "Cybersecure Intelligent Sensor Framework for Smart Buildings: AI-Based Intrusion Detection and Resilience Against IoT Attacks," *Sensors*, vol. 25, no. 24, p. 7680, 2025, doi: <https://doi.org/10.3390/s25247680>.
- [27] M. A. Siam *et al.*, "AI-Driven Cyber Threat Intelligence Systems: A National Framework for Proactive Defense Against Evolving Digital Warfare," *Int. J. Comput. Exp. Sci. Eng.*, vol. 11, no. 3, 2025, doi: <https://doi.org/10.22399/ijcesen.3793>.
- [28] J. Hassan *et al.*, "Blockchain Integration in Management Information Systems: A Decentralized Approach to Strengthening Cybersecurity and Data Integrity," *2025 5th International Conference on Electrical, Computer and Energy Technologies (ICECET)*. IEEE, pp. 1–7, 2025. doi: <https://doi.org/10.1109/ICECET63943.2025.11472020>.