SLA-DRIVEN LOAD SCHEDULING IN MULTI-TIER CLOUD COMPUTING: FINANCIAL IMPACT CONSIDERATIONS

Husam Suleiman and Otman Basir
Department of Electrical and Computer Engineering,
University of Waterloo, Canada

ABSTRACT

A cloud service provider strives to effectively provide a high Quality of Service (QoS) to client jobs. Such jobs vary in computational and Service-Level-Agreement (SLA) obligations, as well as differ with respect to tolerating delays and SLA violations. The job scheduling plays a critical role in servicing cloud demands by allocating appropriate resources to execute client jobs. The response to such jobs is optimized by the cloud service provider on a multi-tier cloud computing environment. Typically, the complex and dynamic nature of multi-tier environments incurs difficulties in meeting such demands, because tiers are dependent on each others which in turn makes bottlenecks of a tier shift to escalate in subsequent tiers. However, the optimization process of existing approaches produces single-tier-driven schedules that do not employ the differential impact of SLA violations in executing client jobs. Furthermore, the impact of schedules optimized at the tier level on the performance of schedules formulated in subsequent tiers tends to be ignored, resulting in a less than optimal performance when measured at the multi-tier level. Thus, failing in committing job obligations incurs SLA penalties that often take the form of either financial compensations, or losing future interests and motivations of unsatisfied clients in the service provided. Therefore, tolerating the risk of such delays on the operational performance of a cloud service provider is vital to meet SLA expectations and mitigate their associated commercial penalties. Such situations demand the cloud service provider to employ scalable service mechanisms that efficiently manage the execution of resource loads in accordance to their financial influence on the system performance, so as to ensure system reliability and cost reduction. In this paper, a scheduling and allocation approach is proposed to formulate schedules that account for differential impacts of SLA violation penalties and, thus, produce schedules that are optimal in financial performance. A queue virtualization scheme is designed to facilitate the formulation of optimal schedules at the tier and multi-tier levels of the cloud environment. Because the scheduling problem is NP-hard, a biologically inspired approach is proposed to mitigate the complexity of finding optimal schedules. The reported results in this paper demonstrate the efficacy of the proposed approach in formulating cost-optimal schedules that reduce SLA penalties of jobs at various architectural granularities of the multi-tier cloud environment.

KEYWORDS
Cloud Computing, Task Scheduling and Balancing, QoS Optimization, Resource Allocation, Genetic Algorithms

1. INTRODUCTION

In a cloud computing environment, client jobs have different service demands and QoS obligations that should be met by the cloud service provider. The arrival of such jobs tends to be random in nature. Cloud resources should deliver services to fulfill different client demands, yet such resources might be limited. Arrival rates of jobs dynamically vary at run-time, which in turn cause bottlenecks and execution difficulties on cloud resources. It is typical that an SLA is employed to govern the QoS obligations of the cloud computing service provider to the client. A service provider conundrum revolves around the desire to maintain a balance between two conflicting objectives: the limited resources available for computing and the

DOI: 10.5121/ijccsa.2020.10201

high QoS expectations of varying random computing demands. Any imbalance in managing these conflicting objectives may result in either dissatisfied clients and potentially significant commercial penalties, or an over-sourced cloud computing environment with large assets of computational resources that can be significantly costly to acquire and operate.

Various scheduling approaches are presented in the literature to address the problem so that QoS expectations of client jobs are obtained. Such approaches often focus on optimizing system-level metrics at the resource level of the cloud computing environment, and hence aim at minimizing the response times of client jobs by allocating adequate resources. The response time of a job entails two components: the job's waiting time at the queue level and the job's service time at the resource level. The bottleneck of jobs in the queues has a direct impact on the waiting times of client jobs and, thus, their response times.

A major limitation in schedulers of existing approaches is that they often optimize performance of schedules at the individual resource level. As such, they fail to take advantage of any available capacities of the other resources within the tier. Furthermore, single-resource-driven scheduling is blind to the impact of the resultant schedules on other tiers. Due to complications of the bottleneck shifting and dependencies between tiers of the multi-tier cloud environment, SLA violations of client jobs in a tier would escalate when such jobs progress through subsequent tiers of the cloud environment. Also, such schedules are blind to penalties incurred by the cloud service provider due to SLA violations.

It is typical that a cloud service provider strives to maintain the highest QoS provided to clients, so as to maintain client satisfaction [1–3]. The more satisfied the clients, the higher the likelihood they will choose the cloud service provider to execute their demands. However, cloud jobs often differ with respect to delay tolerance, resource computational demand, QoS expectations, and financial value. Furthermore, certain jobs are time-critical and hence cannot tolerate execution delays, as well as are financially delay-sensitive and tightly coupled with the client experience. Any delays in responding to SLA obligations of such jobs would likely cause financial losses and negative reputation consequences, which thus negatively affects client loyalties of choosing the cloud service provider.

Take, for example, the first notice of a loss application. Once a vehicle gets into an accident, an on-board system detects and sends the accident data to the cloud service provider to process and determine accident location severity, and as a result, notify the appropriate police department. Any delay in processing these data leads to catastrophic consequences. Thus, the SLA that governs this application produces severe penalties reflective of these consequences.

Therefore, the cloud service provider must: (1) ensure resource availability for such jobs under all circumstances, which has to be a function of SLA impacts associated with the jobs; (2) formulate cost-optimal schedules that account for the differential impact of delays in executing client jobs to minimize potential penalties due to such delays; (3) develop a model that computes SLA violation penalties of client jobs and supports the commitment of the cloud service provider in delivering better service and client experience; (4) mitigate the computational complexity of scheduling the excessive client demands on resource queues, as well as facilitate the exploration and exploitation through the search space of schedules to find an optimal scheduling solution.

In this paper, a differentiated impact scheduling approach is proposed to formulate penalty-aware QoS-driven schedules that are optimal in financial performance. The scheduling approach extends the previous work published in [4, 5], and accounts for the followings:

- The utilization of resources within a tier is leveraged so as to influence tier-driven schedules that account for the mutual performance impact of tier resources on the system performance.
- The effect of tier dependencies on the system performance is leveraged so as to produce multi-tier-driven schedules that contemplate the impact of schedules optimized in a tier on the performance of schedules formulated in subsequent tiers.
- A penalty model that allows for differential treatments of jobs is employed so as to ensure financially optimal job schedules.
- A genetic-based approach and a queue virtualization scheme are designed to formulate schedules at the tier and multi-tier levels of the cloud environment, as well as to alleviate and simplify the complexity of finding optimal schedules.

2. Background and Related Work

The performance of a cloud service provider is highly influenced by the availability of resources, to ensure reliable and efficient executions of the varying client demands. Improving the cloud performance through efficient service models is a driving factor to properly tackle the differentiated QoS penalties of client jobs, so as to reduce negative commercial consequences on the cloud service provider and the client [6–8]. Such models should maintain high client satisfactions and business continuity through reliable scheduling and balancing strategies that support efficient utilization of cloud resources. The strategies should provide services to client demands in a timely manner and thus mitigate the negative impact on the QoS delivered to clients [9].

Existing approaches in the literature typically address the scheduling of client jobs that entail identical SLA penalties on a single-tier environment [10–12]. Jia *et al.* [13] propose a multi-resource, load balancing greedy algorithm for distributed cloud cache systems. The algorithm seeks a locally optimal schedule of stored data among cache resources so as to minimize the imbalance degree. Resources are allocated priorities/weights according to the system load-distribution, where higher priorities are given to under-utilized resources. Furthermore, a market-based load balancing algorithm is proposed by Yang *et al.* [14] to distribute workloads between resources. The cost of a job is directly related to the resource loads, as well as resources continuously exchange state load-information to decide on the redistribution and allocation of jobs. Heavily utilized resources are assigned higher cost, and thus are not allocated client jobs. The former algorithms minimize the response times of jobs and load imbalance degrees, however they only compute single-tier-driven and penalty-unaware schedules.

The effect of different levels in computational demands and SLA soft deadlines on the system performance of a single-tier environment is investigated by Stavrinides *et al.* [15]. A tardiness bound relative to the job's service deadline is employed to represent SLA violations. Moon *et al.* [16] describe the SLA as a function of response time, where a client's job does not incur an SLA penalty on the cloud service provider if the job completes the execution within pre-defined service bounds. Also, Chen *et al.* [17] present a client-priority-aware load balancing algorithm to produce schedules that increase the utilization of resources and reduce the makespan of client jobs. Nayak *et al.* [18] propose a scheduling mechanism to enhance the acceptance-rate ratio of deadline-sensitive tasks and maximize resource utilization. However, optimization strategies of the former approaches fail to contemplate differentiated QoS penalties for client jobs when the varying levels of SLA violations and tardiness bounds are translated into quantifiable penalties on the cloud service provider.

Moreover, several scheduling approaches are proposed to improve the latency of client jobs. For instance, the redundancy-based scheduling is a promising reliability approach that makes duplicate copies of a job on multiple resources as presented in Lee *et al.* [19], Birke *et al.* [20], and Gardner *et al.* [21]. Nevertheless, the redundancy approach generally devises scheduling treatment regimes that formulate schedules for client jobs whose QoS penalties are identical, while the various SLA commitments of jobs with their associated differential penalties are not employed when such schedules are produced.

Mailach *et al.* [22] schedule jobs based on their estimated service times. Okopa *et al.* [23] present a fixed-priority scheduling to address the execution of client jobs with variant execution demands. The proposed scheduling policy primarily delivers high service performance to high priority jobs by reducing their average response times, nevertheless, it negatively penalizes the performance of low priority jobs. However, such schedules are only single-tier-driven formulated on single-server and multi-server systems, while differential service penalties of jobs are not applied.

Furthermore, pair-based scheduling mechanisms are proposed to minimize the execution time of tasks on resources of multiple clouds [24, 25]. Panda *et al.* [26] formalize job schedules on multiple clouds, as well as present scheduling algorithms that enhance the makespan of tasks and average utilization of the clouds. One presented scheduling algorithm employs the task minimum completion time as a performance indicator to schedule tasks on the cloud that completes their execution at the earliest time. Another scheduling algorithm computes the median of tasks over all clouds, so as to assign the maximum-median task to the cloud that completes the execution at the earliest opportunity.

Similarly, Moschakis *et al.* [27] present a multi-cloud scheduling model, and propose a scheduling strategy to dispatch tasks into the least loaded cloud using an inter-cloud dispatcher. In each cloud, a private cloud dispatcher is employed to distribute incoming tasks with the goal of minimizing the total makespan and maximizing the utilization of resources. However, the former multi-cloud-based scheduling algorithms identically penalize client jobs regardless of the performance effect of their service tardiness and demands on such clouds. Such approaches would in turn fail to formulate optimal schedules when the multiple clouds employ multi-tier environments, primarily when client jobs entail various differentiated SLA penalties to represent their service performance.

Moreover, meta-heuristic approaches are employed to provide near-optimal schedules in a reasonable time [28–31]. Such approaches are typically used because the various characteristics and SLA obligations of clients make tackling the scheduling problem a complex task that often cannot be effectively addressed in a polynomial time. The honey-bee meta-heuristic algorithm has been employed by Babu *et al.* [32, 33] to distribute workloads between resources and minimize job response times. Jobs removed from overloaded queues are treated as honey bees, while underloaded queues are treated as food sources. Although the honey-bee scheduling approach improves the satisfaction for a specific job, it does not account for the satisfaction status of other client jobs and their effect on the system performance. Thus, other jobs waiting in the queues of the tier would not necessarily be satisfied and benefit from the scheduling decision.

In addition, Gautam *et al.* [34] use a genetic-based approach to formulate QoS-based optimal schedules that reduce the delay cost of client jobs, however, in a single-tier environment. In a similar environment, a resource scheduling genetic-based approach is proposed by Wang *et al.* [35] to allocate independent tasks of known service demands to a set of resources, to minimize the response time and energy consumption cost. Likewise, Boloor *et al.* [36] present a heuristic-based scheduling approach to tackle the execution of client requests on resources of multiple data centers such that the percentile of requests' response times is less than a pre-defined value. Zhan *et al.* [37] present a load-balance-aware, genetic-based scheduling method to minimize the makespan of client jobs. Nevertheless, the former meta-heuristic approaches do not address the differentiated penalties of client jobs at the system-metric level of delay cost and response time of client jobs. Also, such schedules produce non-optimal performance when measured at the multi-tier level.

Furthermore, a combination of Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) algorithms is presented by Cho *et al.* [38] to jointly compose an ACOPS algorithm that balances the load between resources. The PSO operator is used to speedup the convergence procedure of the ACO scheduling. However, the ACOPS employs a pre-reject operator to reject tasks that demand for a memory larger than the remaining memory in the system. Although the pre-reject operator reduces the scheduling solution space and time produced by the ACO algorithm, the various SLA commitments of tasks make such rejection strategies increase QoS penalties and the likelihood of dissatisfied clients. The performance of schedules formulated through such meta-heuristic approaches are not optimized at the tier and multi-tier levels of the cloud computing environment.

Reig *et al.* [39] rely on prediction models to identify the resource requirements that the client is entitled to consume, so as to avoid inefficient allocation and utilization of resources. However, such models do not distribute the load among resources by means of optimal schedules to guarantee the explicit SLA obligations of clients. Nevertheless, maximizing resource utilization would often incur potential SLA violations, while focusing on satisfying SLA requirements of clients may imply poor resource utilization [40].

Generally speaking, existing approaches adopt identical SLA penalties for jobs that demand for optimal QoS-aware schedules formulated through either single-tier-driven or resource-driven strategies. Because client jobs often tend to have different tolerances and sensitivities to SLA violations, such approaches in their optimization strategies would formulate schedules that do not account for the performance impact of the differential SLA penalties of client jobs at the multi-tier level of the cloud computing environment. An optimal balance should be maintained between meeting QoS obligations specified in the SLA and mitigating commercial penalties associated with potential SLA violations.

As such, a differential penalty is a viable performance metric that should be devised to reflect on the QoS provided to clients. This paper presents an SLA-based management approach that mitigates the effect of a penalty in multi-tier cloud computing environments through differential penalty-driven scheduling. Optimal schedules are formulated with respect to SLA commitments of clients, in the context of various QoS and in compliance with the risk operations of the cloud service provider.

3. Penalty-Oriented Multi-Tier SLA Centric Scheduling of Cloud Jobs

A multi-tier cloud computing environment consisting of N sequential tiers is considered:

$$T = \{T_1, T_2, T_3, ..., T_N\} \tag{1}$$

Each tier T_j employs a set of identical computing resources R_j:

$$R_j = \{R_{j,1}, R_{j,2}, R_{j,3}, ..., R_{j,M}\} \tag{2}$$

Each resource $R_{j,k}$ employs a queue $Q_{j,k}$ that holds jobs waiting for execution by the resource. Jobs with different resource computational requirements and QoS obligations are submitted to the environment. It is assumed that these jobs are submitted by different clients and hence are governed by various SLA's. Jobs arrive at the environment in streams. A stream S is a set of jobs:

$$S = \{J_1, J_2, J_3, ..., J_l\} \tag{3}$$

The index of each job J_i signifies its arrival ordering at the environment. For example, job J_1 arrives at the environment before job J_2. Jobs submitted to tier T_j are queued for execution based on an ordering β_j. As shown in Figure 1, each tier T_j of the environment consists of a set of resources R_j. Each resource $R_{j,k}$ has a queue $Q_{j,k}$ to hold jobs assigned to it. For instance, resource $R_{j,1}$ of tier T_j is associated with queue $Q_{j,1}$, which consists of 3 jobs waiting for execution.

$$\beta_j = \bigcup_{k=1}^{M_k} \mathrm{I}(Q_{j,k}), \quad \forall j \in [1, N] \tag{4}$$

where $\mathrm{I}(Q_{j,k})$ represents indices of jobs in $Q_{j,k}$. For instance, $\mathrm{I}(Q_{1,2}) = \{3, 5, 2, 7\}$ signifies that jobs J_3, J_5, J_2, and J_7 are queued in $Q_{1,2}$ such that job J_3 precedes job J_5, which in turn precedes job J_2, and so on.

Jobs arrive in random manner. A job dispatcher JD_j is employed to buffer incoming client jobs to tier T_j. Job J_i arrives at tier T_j at time $A_{i,j}$ via the queue of the job dispatcher JD_j of the tier. It has a prescribed execution time $\mathcal{E}_{i,j}$ at each tier. Each job has a service deadline $\mathcal{DL}_i$, which in turn stipulates a target completion time $\mathcal{C}_i^{(t)}$ for the job J_i in the multi-tier environment.

$$J_i = \left\{A_{i,j}, \mathcal{E}_{i,j}, \mathcal{C}_i^{(t)}\right\}, \quad \forall\, T_j \in T \tag{5}$$

The total execution time $\mathcal{ET}_i$ of each job J_i is as follows:

$$\mathcal{ET}_i = \sum_{j=1}^{N} \mathcal{E}_{i,j} \tag{6}$$

The job dispatcher JD_j queues these jobs to the resource queues R_j of the tier. Job J_i waits $\omega_{i,j}^{\beta_j}$ time units in tier T_j according to an ordering β_j of the jobs waiting for execution at resources R_j. Job J_i gets its turn of execution by resource $R_{j,k}$, and afterward, leaves tier T_j at time $D_{i,j}$ to be queued by the dispatcher JD_{j+1} of tier T_{j+1}. When leaving the cloud environment from tier N, job J_i has a response time $\mathcal{RT}_i^{\beta}$ and end-to-end waiting time $\omega\mathcal{T}_i^{\beta}$ computed according to the overall ordering β of jobs at the N tiers.

$$\beta = \bigcup_{j=1}^{N} \beta_j \tag{7}$$

The waiting time $\omega_{i,j}^{\beta_j}$ of each job J_i at tier T_j is defined as the difference between the time it starts execution by one of the resources and its arrival time $A_{i,j}$. The end-to-end waiting time $\omega\mathcal{T}_i^{\beta}$ of job J_i according to

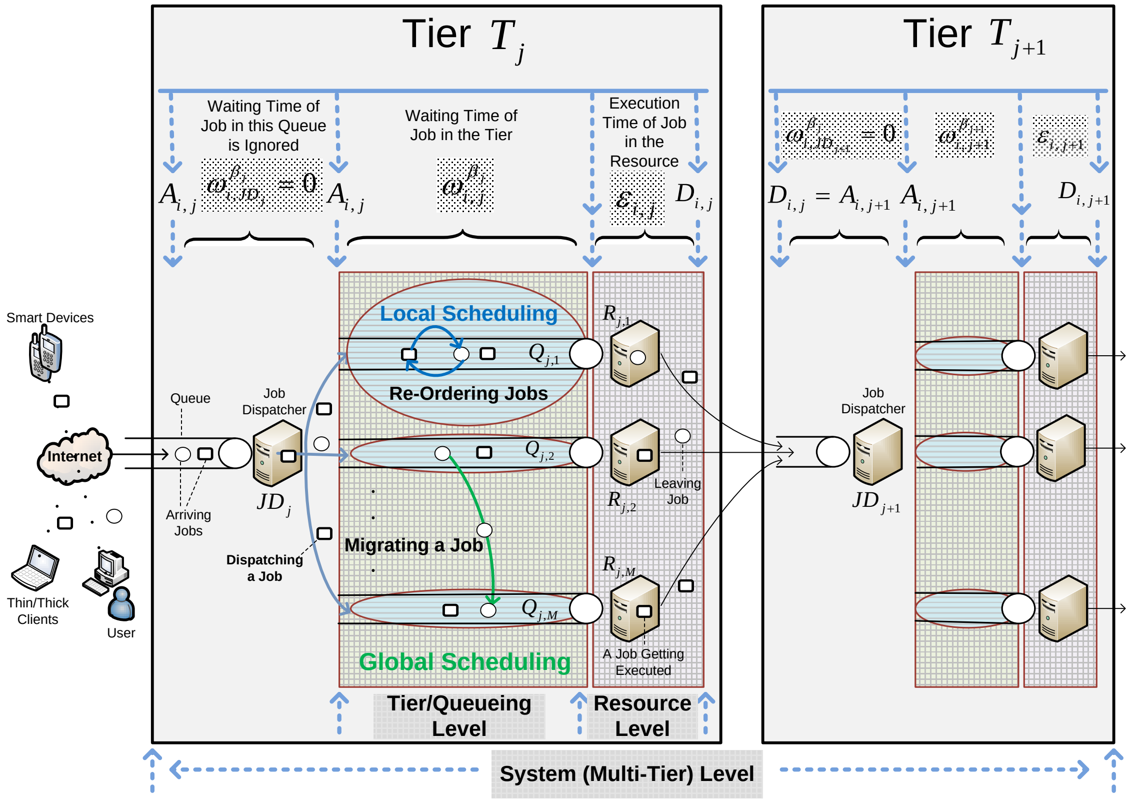


Figure 1. Modeling Parameters and Operators of 2 Consecutive Tiers of the Multi-Tier Cloud Environment

the overall ordering β across all tiers in the multi-tier cloud environment is defined as the summation of the job's waiting time $\omega^{\beta_j}_{i,j}$ in all tiers. The response time $\mathcal{RT}^{\beta}_i$ of job J_i in the multi-tier cloud environment is defined as the difference between the departure time $D_{i,N}$ of job J_i from the last tier T_N and the arrival time $A_{i,1}$ of job J_i to the first tier T_1. The response time $\mathcal{RT}^{\beta}_i$ of job J_i can also be viewed as the summation of waiting times $\omega^{\beta_j}_{i,j}$ and execution times $\mathcal{E}_{i,j}$. The performance parameters $\omega^{\beta_j}_{i,j}$, $\omega\mathcal{T}^{\beta}_i$, and $\mathcal{RT}^{\beta}_i$ for each job J_i are computed as follows:

$$\omega^{\beta_j}_{i,j} = D_{i,j} - \mathcal{E}_{i,j} - A_{i,j} \tag{8}$$

$$\omega\mathcal{T}^{\beta}_i = \sum_{j=1}^{N} \omega^{\beta_j}_{i,j} \tag{9}$$

$$\mathcal{RT}^{\beta}_i = D_{i,N} - A_{i,1} = \sum_{j=1}^{N} (\omega^{\beta_j}_{i,j} + \mathcal{E}_{i,j}) = \omega\mathcal{T}^{\beta}_i + \mathcal{ET}_i \tag{10}$$

The excessive volume of client demands and the potential lack of adequate resource availability are critical situations for the cloud service providers. Priorities are, therefore, given to jobs according to the impact of potential delays in their execution. Such priorities must be reflected in the scheduling strategy in a way that ensures the financial viability of the cloud service provider and, at the same time, high client satisfaction. The scheduling strategy should leverage the available delay tolerance of client jobs so as to satisfy the critical demands of delay intolerant jobs.

3.1. Differentiated Cost of Time-Based Scheduling

The execution time $\mathcal{E}_{i,j}$ of job J_i at tier T_j is pre-defined in advance. Therefore, the resource capabilities of each tier T_j are not considered and, thus, the total execution time $\mathcal{ET}_i$ of job J_i is constant. Instead, the primary concern is on the queueing-level of the environment represented by the total waiting time $\omega\mathcal{T}_i^\beta$ of job J_i at all tiers T according to the ordering β.

A unit of waiting time $\omega\mathcal{T}_i$ of job J_i would incur a differentiated financial service cost ψ_i. Such situations demand the cloud service provider emphasize the notion of financial penalty in the scheduling of client jobs so that schedules are computed based on economic considerations. The service penalty cost ψ_i is assumed to follow a normal distribution with a mean μ and variance σ.

$$\psi_i = N(\mu, \sigma) \tag{11}$$

The service time of job J_i is subject to an SLA that stipulates an exponential differentiated financial penalty curve η_i as follows:

$$\eta_i = \chi * (1 - \mathrm{e}^{-\nu\ \psi_i\ \sum_{j=1}^{N} \omega_{i,j}^{\beta j}}) \tag{12}$$

As such, the total differentiated financial performance penalty cost of the job stream l across all tiers is given by ϑ as follows:

$$\vartheta = \sum_{i=1}^{l} \eta_i \tag{13}$$

The objective is to find job orderings $\beta = (\beta_1, \beta_2, \beta_3, \ldots, \beta_N)$ such that the stream's total differentiated financial penalty cost ϑ is minimal:

$$\underset{\beta}{\text{minimize}}\ (\vartheta) \equiv \underset{\beta}{\text{minimize}} \sum_{i=1}^{l} \sum_{j=1}^{N} (\ \psi_i\ \omega_{i,j}^{\beta_j}\ ) \tag{14}$$

3.2. Differentiated Cost of Time-Based Scheduling: Multi-Tier Considerations

The target completion time $\mathcal{C}_i^{(t)}$ of job J_i represents an explicit QoS obligation on the service provider to complete the execution of the job. Thus, the $\mathcal{C}_i^{(t)}$ incurs a service deadline $\mathcal{DL}_i$ for the job in the environment. The service deadline $\mathcal{DL}_i$ is higher than the total prescribed execution time $\mathcal{ET}_i$ and incurs a total waiting time allowance $\omega\mathcal{AL}_i$ for job J_i in the environment.

$$\begin{aligned} \mathcal{DL}_i &= \mathcal{C}_i^{(t)} - A_{i,j} \\ &= \mathcal{ET}_i + \omega\mathcal{AL}_i \end{aligned} \tag{15}$$

As such, the time difference between the response time $\mathcal{RT}_i^\beta$ and the service deadline $\mathcal{DL}_i$ represents the service-level violation time α_i^β of job J_i, according to the ordering β of jobs in tiers T of the environment.

$$(\mathcal{RT}_i^\beta - \mathcal{DL}_i) = \begin{cases} \alpha_i^\beta > 0, & \text{The client is not satisfied} \\ \alpha_i^\beta \leq 0, & \text{The client is satisfied} \end{cases} \tag{16}$$

A unit of SLA violation time α_i^β of the job J_i at the multi-tier level of the environment incurs a differentiated financial SLA violation cost ζ_i. The cost ζ_i of SLA violation at the multi-tier level is assumed to follow a normal distribution with a mean μ and variance σ.

$$\zeta_i = N(\mu, \sigma) \tag{17}$$

The service-level violation time α_i^β is subject to an SLA that stipulates an exponential differentiated financial penalty curve η_i as follows:

$$\begin{aligned} \eta_i &= \chi * (1 - \mathrm{e}^{-\nu\ \zeta_i\ (\mathcal{RT}_i^\beta - \mathcal{DL}_i)}) \\ &= \chi * (1 - \mathrm{e}^{-\nu\ \zeta_i\ (\omega\mathcal{T}_i^\beta - \omega\mathcal{AL}_i)}) \\ &= \chi * (1 - \mathrm{e}^{-\nu\ \zeta_i\ \alpha_i^\beta}) \end{aligned} \tag{18}$$

where χ is a monetary cost factor and ν is an arbitrary scaling factor. The total performance penalty cost ϑ of the stream l across all tiers is given by Equation 13 and, accordingly, the financial performance of job schedules is optimized such that the differentiated SLA violation penalty is minimized at the multi-tier level.

- *Differentiated $\omega\mathcal{AL}_i$ Based Minimum Penalty Formulation*

The performance of job schedules is formulated with respect to the multi-tier waiting time allowance $\omega\mathcal{AL}_i$ of each job J_i. Accordingly, the SLA violation penalty is evaluated at the multi-tier level of the environment. The objective is to seek job schedules in tiers of the environment such that the total SLA violation penalty of jobs would be minimized *globally* at the multi-tier level of the environment.

The total waiting time $\omega\mathcal{T}_i^\beta$ of job J_i currently waiting in tier T_p, where $p < N$, is not totally known because the job has not yet completely finished execution from the multi-tier environment. Therefore, the job's $\omega\mathcal{T}_i^\beta$ at tier T_p is estimated and, thus, represented by $\omega\mathcal{CX}_{i,p}^\beta$ according to the scheduling order β of jobs. As such, the job's service-level violation time α_i^β at tier T_p would be represented by the expected waiting time $\omega\mathcal{CX}_{i,p}^\beta$ of job J_i in the current tier T_p and the waiting time allowance $\omega\mathcal{AL}_i$ incurred from the job's service deadline $\mathcal{DL}_i$ at the multi-tier level of the environment.

$$\alpha_i^\beta = \omega\mathcal{CX}_{i,p}^\beta - \omega\mathcal{AL}_i \tag{19}$$

where the expected waiting time $\omega\mathcal{CX}_{i,p}^\beta$ of job J_i at tier T_p incurs the total waiting time $\omega\mathcal{T}_i^\beta$ of job J_i at the multi-tier level.

$$\omega\mathcal{CX}_{i,p}^\beta = \sum_{j=1}^{(p-1)} (\omega_{i,j}^{\beta_j}) + \omega E\mathcal{L}_{i,p} + \omega\mathcal{RM}_{i,p}^{\beta_p} \tag{20}$$

where $\omega_{i,j}^{\beta_j}(\forall j \leq (p-1))$ represents the waiting time of job J_i in each tier T_j in which the job has completed execution, $\omega E\mathcal{L}_{i,p}$ represents the elapsed waiting time of job J_i in the tier T_p where the job currently resides, and $\omega\mathcal{RM}_{i,p}^{\beta_p}$ represents the remaining waiting time of job J_i according to the scheduling order β_p of jobs in the current holding tier T_p.

$$\omega\mathcal{RM}_{i,j}^{\beta_j} = \sum_{h \in \mathrm{I}(Q_{j,k}),\ h \text{ precedes job } J_i}^{\forall} \mathcal{E}_{h,j}, \quad \forall j \in [1, N] \tag{21}$$

where $\mathrm{I}(Q_{j,k})$ represents indices of jobs in $Q_{j,k}$. For instance, $\mathrm{I}(Q_{1,2}) = \{3, 5, 2, 7\}$ signifies that jobs J_3, J_5, J_2, and J_7 are queued in $Q_{1,2}$ such that job J_3 precedes job J_5, which in turn precedes job J_2, and so on. However, the elapsed waiting time $\omega E\mathcal{L}_{i,j}$ affects the execution priority of the job. The higher the time of $\omega E\mathcal{L}_{i,j}$ of job J_i in the tier T_j, the lower the remaining allowed time of $\omega\mathcal{AL}_i$ of job J_i at the multi-tier level, thus, the higher the execution priority of job J_i in the resource.

The objective is to find scheduling orders $\beta = (\beta_1, \beta_2, \beta_3, \ldots, \beta_N)$ for jobs of each tier T_j such that the stream's total differentiated penalty ϑ is minimal, and thus the SLA violation penalty is minimal. The financially optimal performance scheduling with respect to $\omega\mathcal{AL}_i$ is formulated as:

$$\underset{\beta}{\text{minimize}}\ (\vartheta) \equiv \underset{\beta}{\text{minimize}} \sum_{i=1}^{l} \sum_{p=1}^{N} \zeta_i\ (\omega\mathcal{CX}_{i,p}^\beta - \omega\mathcal{AL}_i) \tag{22}$$

- *Differentiated $\omega\mathcal{PT}_{i,j}$ Based Minimum Penalty Formulation*

The performance of job schedules is formulated with respect to a differentiated waiting time $\omega\mathcal{PT}_{i,j}$ of the job J_i at each tier T_j. The $\omega\mathcal{PT}_{i,j}$ is derived from the multi-tier waiting time allowance $\omega\mathcal{AL}_i$ of job J_i, with respect to the execution time $\mathcal{E}_{i,j}$ of the job J_i at the tier level relative to the job's total execution time $\mathcal{ET}_i$ at the multi-tier level of the environment.

$$\omega\mathcal{PT}_{i,j} = \omega\mathcal{AL}_i * \frac{\mathcal{E}_{i,j}}{\mathcal{ET}_i} \tag{23}$$

In this case, the higher the execution time $\mathcal{E}_{i,j}$ of job J_i in tier T_j, the higher the job's differentiated waiting time allowance $\omega\mathcal{PT}_{i,j}$ in the tier T_j. Accordingly, the SLA violation penalty is evaluated at the multi-tier level with respect to the $\omega\mathcal{PT}_{i,j}$ of each job J_i.

The waiting time $\omega_{i,j}^{\beta_j}$ of job J_i at tier T_j would not be totally known until the job completely finishes execution from the tier, however, it can be estimated by $\omega\mathcal{PX}_{i,j}^{\beta_j}$ according to the current scheduling order β_j of jobs in the tier T_j. As such, the service-level violation time $\alpha\mathcal{T}_{i,j}^{\beta_j}$ of job J_i in the tier T_j according to the scheduling order β_j of jobs would be represented by the expected waiting time $\omega\mathcal{PX}_{i,j}^{\beta_j}$ and the differentiated waiting time allowance $\omega\mathcal{PT}_{i,j}$, of the job in the tier T_j.

$$\alpha\mathcal{T}_{i,j}^{\beta_j} = \omega\mathcal{PX}_{i,j}^{\beta_j} - \omega\mathcal{PT}_{i,j} \tag{24}$$

$$\alpha_i^{\beta} = \sum_{j=1}^{N} \alpha\mathcal{T}_{i,j}^{\beta_j} \tag{25}$$

where α_i^{β} is the total service-level violation time of the job J_i at all tiers of the environment according to the scheduling order β. The expected waiting time $\omega\mathcal{PX}_{i,j}^{\beta_j}$ incurs the actual waiting time $\omega_{i,j}^{\beta_j}$ of job J_i in tier T_j, and thus depends on the elapsed waiting time $\omega\mathcal{EL}_{i,j}$ and the remaining waiting time $\omega\mathcal{RM}_{i,j}^{\beta_j}$ of the job J_i according to the scheduling order β_j of jobs in the current holding tier T_j.

$$\omega\mathcal{PX}_{i,j}^{\beta_j} = \omega\mathcal{EL}_{i,j} + \omega\mathcal{RM}_{i,j}^{\beta_j} \tag{26}$$

The elapsed waiting time parameter $\omega\mathcal{EL}_{i,j}$ of job J_i in tier T_j affects the job's execution priority in the resource. The higher the time of $\omega\mathcal{EL}_{i,j}$, the lower the remaining time of the differentiated waiting allowance $\omega\mathcal{PT}_{i,j}$ of job J_i in the tier T_j, therefore, the higher the execution priority of the job J_i in the resource, so as to reduce the service-level violation time $\alpha\mathcal{T}_{i,j}^{\beta_j}$ of the job in the tier T_j.

The objective is to find scheduling orders $\beta = (\beta_1, \beta_2, \beta_3, \ldots, \beta_N)$ for jobs of each tier T_j such that the stream's total differentiated penalty ϑ is minimal, and thus the SLA violation penalty is minimal. The financially optimal performance scheduling with respect to $\omega\mathcal{PT}_{i,j}$ is formulated as:

$$\underset{\beta}{\text{minimize}}\ (\vartheta) \equiv \underset{\beta}{\text{minimize}} \sum_{i=1}^{l} \sum_{j=1}^{N} \zeta_i \left(\omega\mathcal{PX}_{i,j}^{\beta_j} - \omega\mathcal{PT}_{i,j}\right) \tag{27}$$

4. Minimum Penalty Job Scheduling: A Genetic Algorithm Formulation

During scheduling of client jobs for execution, a job is first submitted to tier-1 by one of the resources of the tier. Jobs should be scheduled in such a way that minimizes total waiting time and SLA-violation penalties. Finding a job scheduling that yields minimum penalty is an NP problem. Given the expected volume of jobs to be scheduled and the computational complexity of the job scheduling problem, it is prohibitive to seek optimal solution for the job scheduling problem using exhaustive search techniques. Thus, a metaheuristic search strategy, such as Permutation Genetic Algorithms (PGA), is a viable option for exploring and exploiting the large space of scheduling permutations [41]. Genetic algorithms have been successfully adopted in various problem domains [42], and have undisputed success in yielding near optimal solutions for large scale problems, in reasonable time [43].

Scheduling client jobs entails two steps: (1) allocating/distributing the jobs among the different tier resources. Jobs that are allocated to a given resource are queued in the queue of that resource; (2) ordering the jobs in the queue of the resource such that their penalty is minimal. What makes the problem increasingly hard is the fact that jobs continue to arrive, while the prior jobs are waiting in their respective queues for execution. Thus, the scheduling process needs to respond to the job arrival dynamics to ensure that job execution at all tiers is penalty optimal. To achieve this, job ordering in each queue should be treated as a continuous process. Furthermore, jobs should be migrated from one resource to another so as to ensure balanced job allocation and maximum resource utilization. Thus, two operators are employed for constructing optimal job schedules:

- The *reorder* operator is used to change the ordering of jobs in a given queue so as to find an order that minimizes the total penalty of all jobs in the queue.
- The *migrate* operator, in contrast, is used to exploit the benefits of moving jobs between the different resources of the tier so as to reduce the total penalty. This process is adopted at each tier of the environment.

However, implementing the *reorder/migrate* operators in a PGA search strategy is not a trivial task. This implementation complexity can be relaxed by virtualizing the queues of each tier into one virtual queue. The virtual queue is simply a cascade of the queues of the resources. In this way, the two operators are converged into simply a reorder operator. Furthermore, this simplifies the PGA solution formulation. A consequence of this abstraction is the length of the permutation chromosome and the associated computational cost. This virtual queue will serve as the chromosome of the solution. An index of a job in this queue represents a gene. The ordering of jobs in a virtual queue signifies the order at which the jobs in this queue are to be executed by the resource associated with that queue. Solution populations are created by permuting the entries of the virtual queue, using the *order* and *migrate* operators.

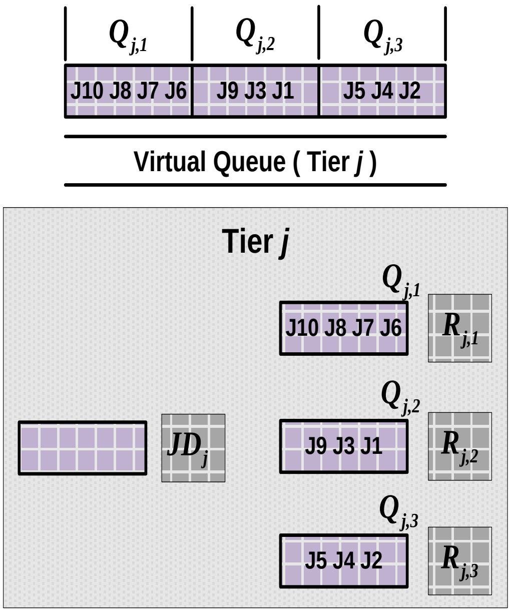


Figure 2. The Virtual Queue of a Tier j

4.1. Tier-Based Virtual Queue

To produce tier-driven optimal performance, a tier-based virtual queue is proposed. In this case, a virtual queue is a cascade of resource queues of the tier. Figures 2 and 3 of the j^{th} tier show the construct of one virtual queue represented as a cascade of the three queues ($Q_{j,1}$, $Q_{j,2}$, and $Q_{j,3}$) of the tier. The schedule's performance is optimized at this virtual-queue level.

4.1.1. Evaluation of Schedules

A fitness evaluation function is used to assess the quality of each virtual-queue realization (chromosome). The fitness value of the chromosome captures the cost of a potential schedule. The fitness value $f_{r,G}$ of a chromosome r in generation G is represented by the differentiated financial waiting penalty of the job schedule in the virtual queue, according to the scheduling order β_j of jobs in each tier T_j.

$$f_{r,G} = \sum_{i=1}^{l} (\psi_i \, \omega_{i,j}^{\beta_j}) \tag{28}$$

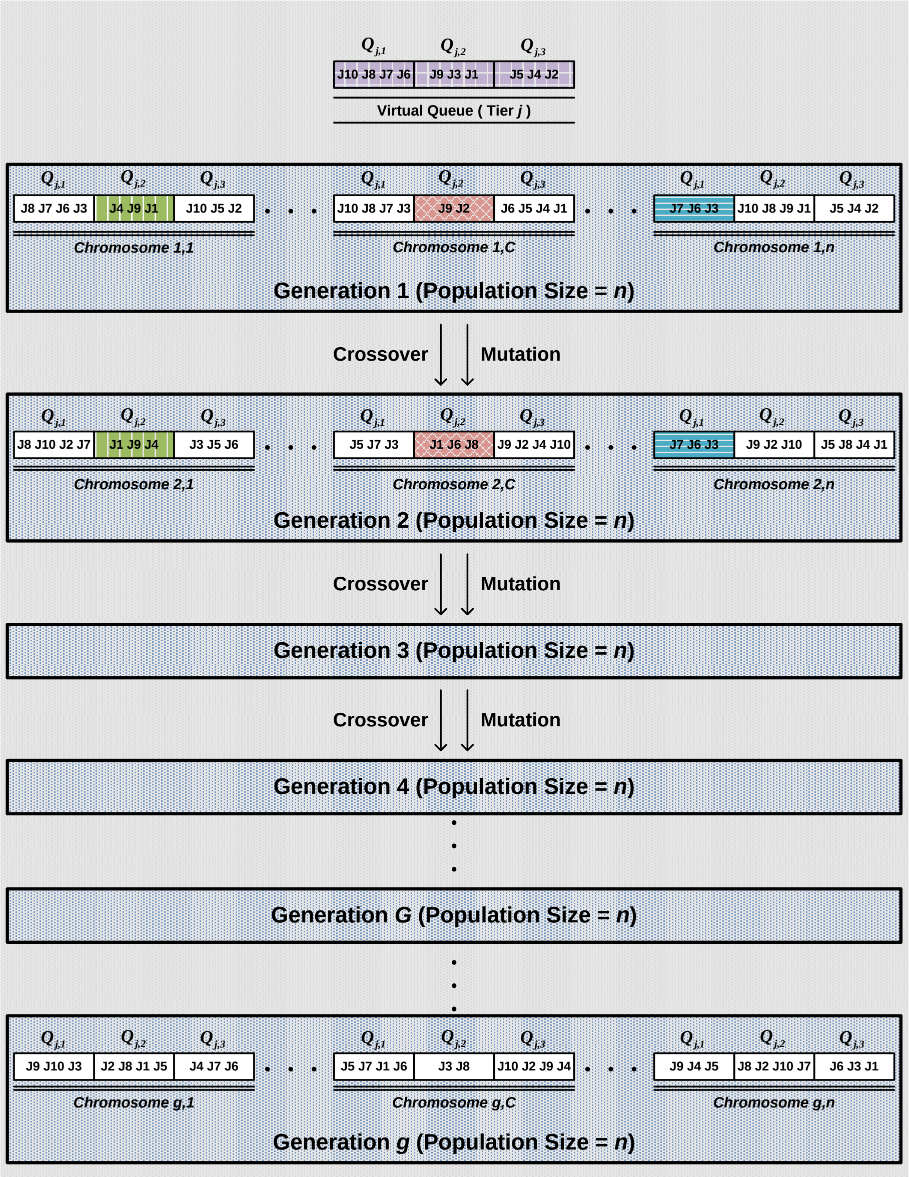

$Q_{j,1}$ $Q_{j,2}$ $Q_{j,3}$
J10 J8 J7 J6 | J9 J3 J1 | J5 J4 J2
Virtual Queue (Tier *j*)
J8 J7 J6 J3 | J4 J9 J1 | J10 J5 J2
Chromosome 1,1
J10 J8 J7 J3 | J9 J2 | J6 J5 J4 J1
Chromosome 1,C
J7 J6 J3 | J10 J8 J9 J1 | J5 J4 J2
Chromosome 1,n
Generation 1 (Population Size = *n*)
Crossover
Mutation
J8 J10 J2 J7 | J1 J9 J4 | J3 J5 J6
Chromosome 2,1
J5 J7 J3 | J1 J6 J8 | J9 J2 J4 J10
Chromosome 2,C
J7 J6 J3 | J9 J2 J10 | J5 J8 J4 J1
Chromosome 2,n
Generation 2 (Population Size = *n*)
Crossover
Mutation
Generation 3 (Population Size = *n*)
Crossover
Mutation
Generation 4 (Population Size = *n*)
Generation G (Population Size = *n*)
J9 J10 J3 | J2 J8 J1 J5 | J4 J7 J6
Chromosome g,1
J5 J7 J1 J6 | J3 J8 | J10 J2 J9 J4
Chromosome g,C
J9 J4 J5 | J8 J2 J10 J7 | J6 J3 J1
Chromosome g,n
Generation *g* (Population Size = *n*)

Figure 3. A Tier-based Genetic Approach on the Virtual Queue

The waiting time $\omega_{i,j}^{\beta_j}$ of the i^{th} job in the virtual queue of the j^{th} tier should be calculated based on its order in the queue, as per the ordering β_j. The normalized fitness value F_r of each schedule candidate is computed as follows:

$$F_r = \frac{f_{r,G}}{\sum_{C=1}^{n}(f_{C,G})} \;, \quad r \in C \tag{29}$$

Based on the normalized fitness values of the candidates, Russian Roulette is used to select a set of schedule candidates to produce the next generation population, using the combination and mutation operators.

4.1.2. Evolving the Scheduling Process

To evolve a new population that holds new scheduling options for jobs in resource queues of the tier, the crossover and mutation genetic operators are both applied on randomly selected schedules (virtual queues) of the current generation. The crossover operator produces a new generation of virtual queues from the current generation. The mutation operator applies random changes on a selected set of virtual queues of the

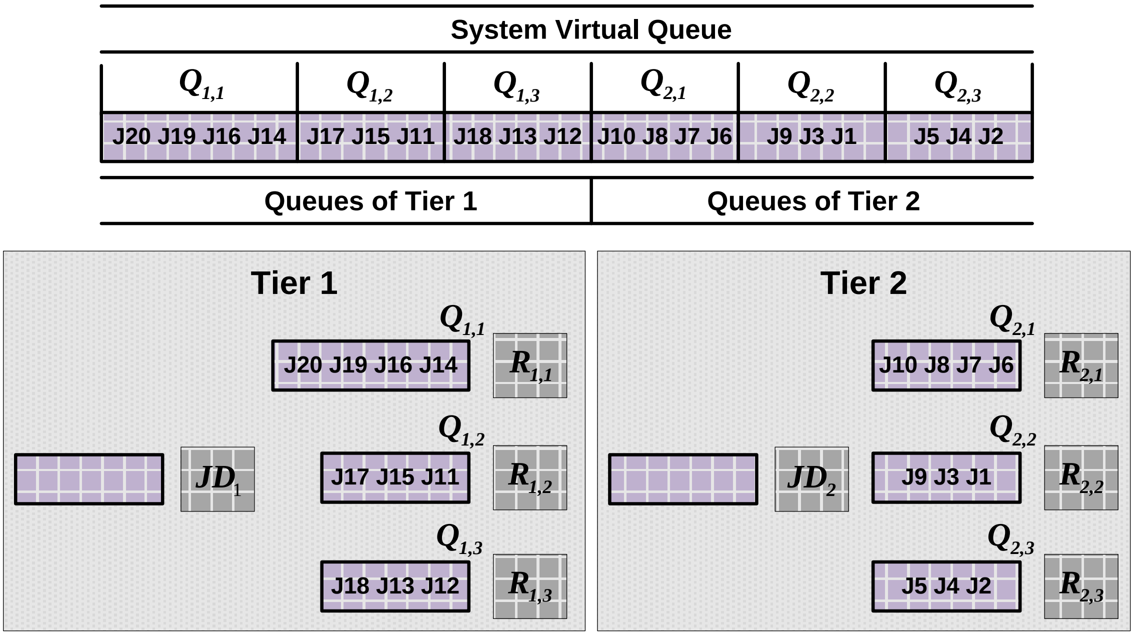


Figure 4. The System Virtual Queue

new generation to produce altered virtual queues. These operators diversify the search direction into new search spaces to avoid getting stuck in a locally optimum solution. Overall, the *Single-Point* crossover and *Insert* mutation genetic operators are used. Rates of crossover and mutation operators are both set to 0.1 of the population size in each generation.

Figure 3 explains how each virtual queue in a given generation is evolved to create a new virtual queue of the next generation, using the crossover and mutation operators. Each chromosome (virtual queue) represents a new scheduling of jobs. The jobs and their order of execution on the resource will be reflected by the segment of the virtual queue corresponding to the actual queue associated with the resource. As a result of the evolution process, each segment of the virtual queue corresponding to an actual queue will be in one of the following states:

- Maintain the same set and order of jobs held in the previous generation;
- Get a new ordering for the same set of jobs held in the previous generation;
- Get a different set of jobs and a new ordering.

For instance, queue $Q_{j,1}$ of *Chromosome* $(1{,}n)$ in the first generation maintains exactly the same set and order of jobs in the second generation shown in queue $Q_{j,1}$ of *Chromosome* $(2{,}n)$. In contrast, queue $Q_{j,2}$ of *Chromosome* (1,1) in the first generation maintains the same set of jobs in the second generation, yet has got a new order of jobs as shown in queue $Q_{j,2}$ of *Chromosome* (2,1). Finally, queue $Q_{j,2}$ of a random *Chromosome* $(1{,}C)$ in the first generation has neither maintained the same set nor the same order of jobs in the second generation shown in queue $Q_{j,2}$ of *Chromosome* $(2{,}C)$, which in turn would yield a new scheduling of jobs in the queue of resource $R_{j,2}$ if *Chromosome* $(2{,}C)$ is later selected as the best chromosome of the tier-based genetic solution.

4.2. Multi-Tier-Based Virtual Queue

The goal is to formulate optimal schedules such that SLA violation penalties of jobs are reduced at the multi-tier level. However, it is complicated to apply the allocation and ordering operators at the multi-tier level. As such, the operator complexities are mitigated by virtualizing resource queues of the multi-tier environment into a single system virtual queue that represents the chromosome of the scheduling solution,

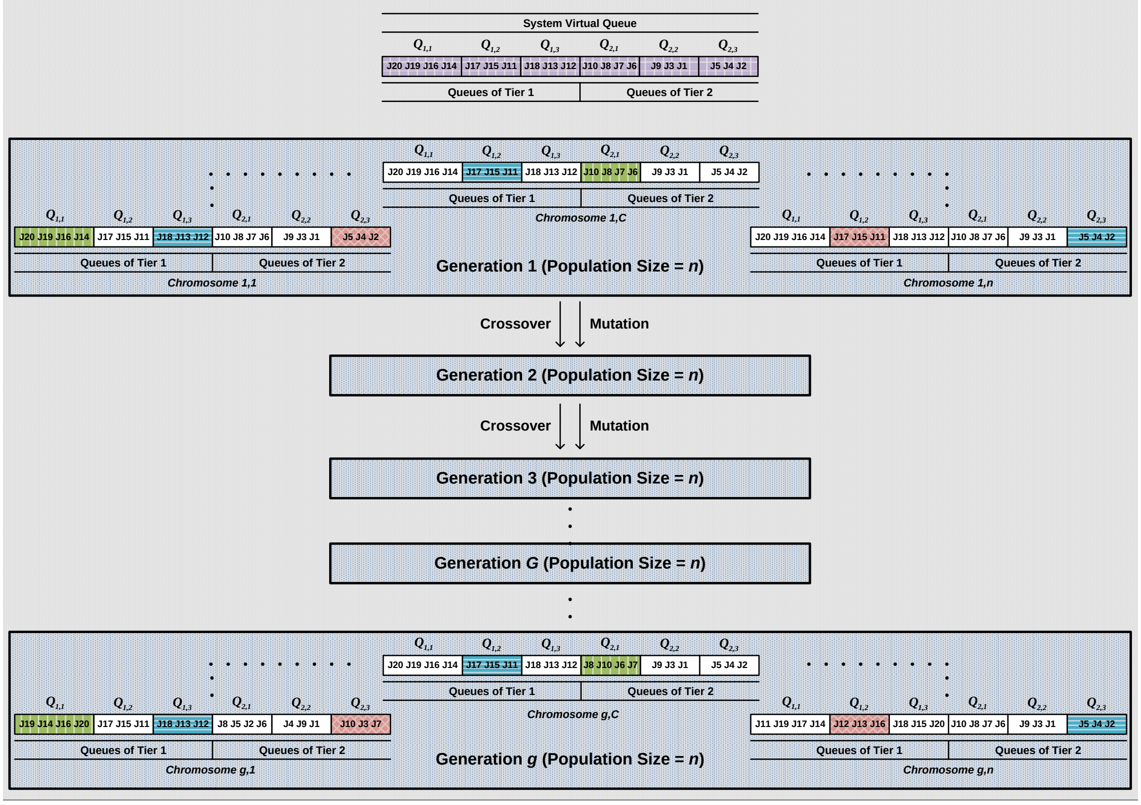

System Virtual Queue
$Q_{1,1}$ $Q_{1,2}$ $Q_{1,3}$ $Q_{2,1}$ $Q_{2,2}$ $Q_{2,3}$
J20 J19 J16 J14 | J17 J15 J11 | J18 J13 J12 | J10 J8 J7 J6 | J9 J3 J1 | J5 J4 J2
Queues of Tier 1
Queues of Tier 2
J20 J19 J16 J14 | J17 J15 J11 | J18 J13 J12 | J10 J8 J7 J6 | J9 J3 J1 | J5 J4 J2
Chromosome 1,1
J20 J19 J16 J14 | J17 J15 J11 | J18 J13 J12 | J10 J8 J7 J6 | J9 J3 J1 | J5 J4 J2
Chromosome 1,C
J20 J19 J16 J14 | J17 J15 J11 | J18 J13 J12 | J10 J8 J7 J6 | J9 J3 J1 | J5 J4 J2
Chromosome 1,n
Generation 1 (Population Size = n)
Crossover
Mutation
Generation 2 (Population Size = n)
Crossover
Mutation
Generation 3 (Population Size = n)
Generation G (Population Size = n)
J19 J14 J16 J20 | J17 J15 J11 | J18 J13 J12 | J8 J5 J2 J6 | J4 J9 J1 | J10 J3 J7
Chromosome g,1
J20 J19 J16 J14 | J17 J15 J11 | J18 J13 J12 | J8 J10 J6 J7 | J9 J3 J1 | J5 J4 J2
Chromosome g,C
J11 J19 J17 J14 | J12 J13 J16 | J18 J15 J20 | J10 J8 J7 J6 | J9 J3 J1 | J5 J4 J2
Chromosome g,n
Generation g (Population Size = n)

Figure 5. A System Virtualized Queue Genetic Approach

as shown in Figure 4. This system-level abstraction converges the operators into simply a reorder operator running at the multi-tier level.

4.2.1. Evaluation of Schedules

The quality of a job schedule in a system virtual queue realization (chromosome) is assessed by a fitness evaluation function. For a chromosome r in generation G, the fitness value $f_{r,G}$ is represented by the SLA violation cost of the schedule in the system virtual queue computed at the multi-tier level. Two different fitness evaluation functions are adopted in two different solutions:

$$f_{r,G} = \begin{cases} \sum_{i=1}^{l} \zeta_i \left(\omega\mathcal{CX}_{i,p}^{\beta} - \omega\mathcal{AL}_i\right), \\ \quad \text{Differentiated Penalty } \omega\mathcal{AL}_i \text{ based Scheduling} \\ \sum_{i=1}^{l} \zeta_i \left(\omega\mathcal{PX}_{i,j}^{\beta_j} - \omega\mathcal{PT}_{i,j}\right), \\ \quad \text{Differentiated Penalty } \omega\mathcal{PT}_{i,j} \text{ based Scheduling} \end{cases} \tag{30}$$

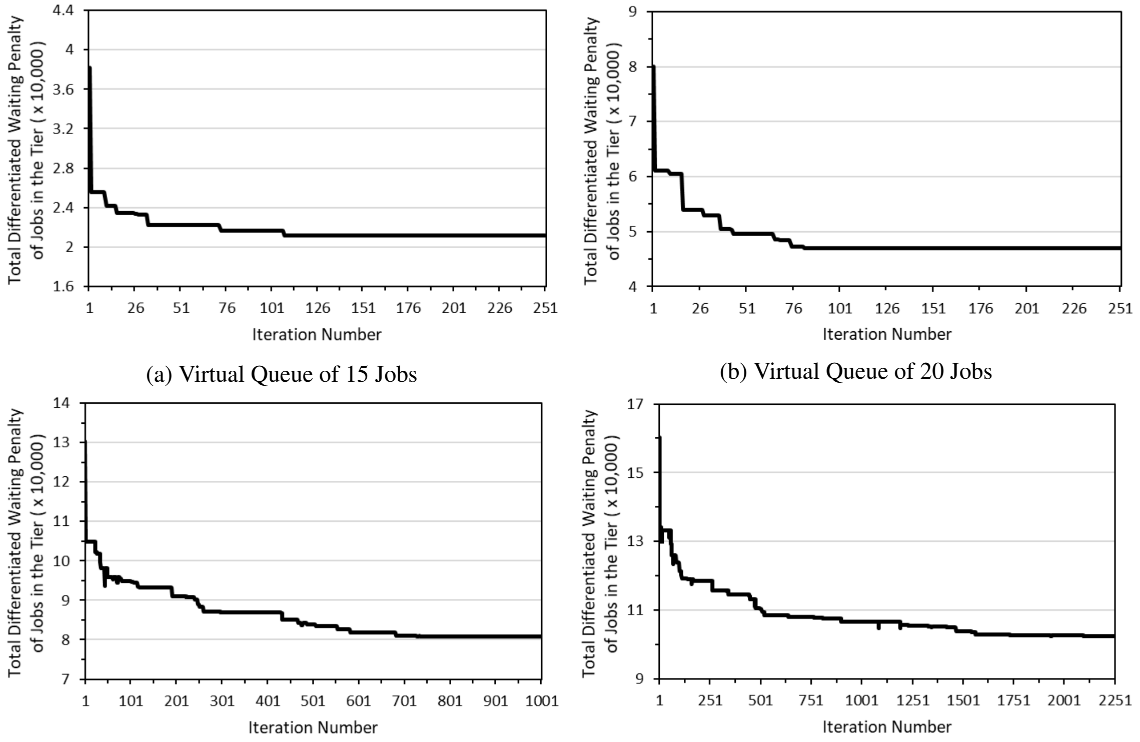
In both scenarios, the SLA violation cost of job J_i is represented by the job's waiting time (either $\omega\mathcal{CX}_{i,p}^{\beta}$ or $\omega\mathcal{PX}_{i,j}^{\beta_j}$) according to its scheduling order β in the system virtual queue and the job's waiting allowance (either $\omega\mathcal{AL}_i$ or $\omega\mathcal{PT}_{i,j}$) incurred from its service deadline $\mathcal{DL}_i$ at the multi-tier level. The normalized fitness value F_r of each schedule candidate is computed as in Equation 29. Based on the normalized fitness values of the candidates, Russian Roulette is used to select a set of schedule candidates that produce the next generation population, using the combination and mutation operators.

4.2.2. Evolving the Scheduling Process

The schedule of the system virtual queue is evolved to produce a population of multiple system virtual queues, each of which represents a chromosome that holds a new scheduling order of jobs at the multi-tier level. To produce a new population, the *Single-Point* crossover and *Insert* mutation genetic operators are applied on randomly selected system virtual queues from the current population. Rates of these operators in each generation are set to be 0.1 of the population size. The evolution process of schedules of the system virtual queues along with the genetic operators are explained in Figure 5. Each segment in the system virtual queue corresponds to an actual queue associated with a resource in the tier. In each generation, each segment is subject to the states examined in Section 4.1.2.

5. Experimental Work and Discussions on Results

The tier-based and multi-tier-based differentiated SLA-driven penalty scheduling are applied on the multi-tier environment. The differentiated service penalty cost ψ_i and SLA violation cost ζ_i for each job are generated using a mean μ of $1{,}000$ cost units and a variance σ of 25. The penalty parameter ν is set to $\nu = \frac{0.01}{1000}$.



(a) Virtual Queue of 15 Jobs

(b) Virtual Queue of 20 Jobs

(c) Virtual Queue of 25 Jobs

(d) Virtual Queue of 30 Jobs

Figure 6. Differentiated Waiting Penalty Tier-Based Scheduling

Table 1. Differentiated Waiting Penalty Tier-Based Scheduling

	Virtual-Queue[1]	**Initial**[2]		**Enhanced**[3]		**Improvement**	
	Length	**Waiting**	**Penalty**	**Waiting**	**Penalty**	**Waiting %**	**Penalty %**
Figure 6a	15	38203	0.318	21168	0.191	44.59%	39.92%
Figure 6b	20	80039	0.551	46190	0.370	42.29%	32.85%
Figure 6c	25	130253	0.728	80532	0.553	38.17%	24.05%
Figure 6d	30	160271	0.799	102137	0.640	36.27%	19.88%

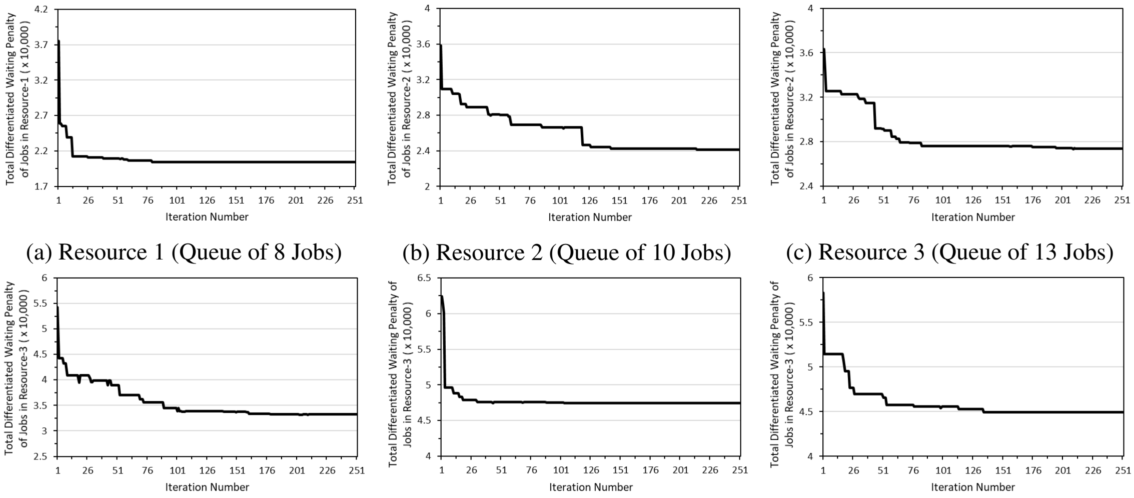
[1] **Virtual-Queue Length** represents the total number of jobs in queues of the tier. For instance, the first entry of the table means that the 3 queues of the tier altogether are allocated 15 jobs.

[2] **Initial Waiting** represents the total waiting penalty of jobs in the virtual queue according to the their initial scheduling before using the tier-based genetic solution.

[3] **Enhanced Waiting** represents the total waiting penalty of jobs in the virtual queue according to the their final/enhanced scheduling found after using the tier-based genetic solution.

5.1. Experimental Evaluation: Performance Penalty

The optimal schedule is the one with a minimum differentiated penalty cost. The penalty cost performance of the proposed scheduling algorithm is mitigated. The effectiveness of penalty cost-driven schedules that produce optimal enhancement and consider the performance of the scheduling algorithm at the single-tier level is evaluated. The virtualized queue and segmented queue genetic scheduling are employed, as well as the service penalty function $f_{r,G}$ in Equation 4.1 is used.



(a) Resource 1 (Queue of 8 Jobs) (b) Resource 2 (Queue of 10 Jobs) (c) Resource 3 (Queue of 13 Jobs)

(d) Resource 1 (Queue of 12 Jobs) (e) Resource 2 (Queue of 8 Jobs) (f) Resource 3 (Queue of 9 Jobs)

Figure 7. Differentiated Waiting Penalty Queue-Based Scheduling

Table 2. Differentiated Waiting Penalty Queue-Based Scheduling

	Queue[1]	Initial[2]		Enhanced[3]		Improvement	
	Length	Waiting	Penalty	Waiting	Penalty	Waiting %	Penalty %
Resource 1 Figure 7a	8	37541	0.313	20431	0.185	45.58%	40.96%
Resource 2 Figure 7b	10	35853	0.301	24126	0.214	32.71%	28.85%
Resource 3 Figure 7c	13	36344	0.305	27162	0.238	25.26%	21.94%
Resource 1 Figure 7d	12	54202	0.418	33130	0.282	38.88%	32.60%
Resource 2 Figure 7e	8	62432	0.464	47481	0.378	23.95%	18.60%
Resource 3 Figure 7f	9	58319	0.442	44934	0.362	22.95%	18.09%

[1] **Queue Length** represents the number of jobs in the queue of a resource.

[2] **Initial Waiting** represents the total waiting penalty of jobs in the queue according to their initial scheduling before using the segmented queue genetic solution.

[3] **Enhanced Waiting** represents the total waiting penalty of jobs in the queue according to their final/enhanced scheduling found after using the segmented queue genetic solution.

The results reported in Table 1 and Figure 6 demonstrate the effectiveness of the differentiated penalty-based scheduling in reducing total service penalty cost, at the virtualized queue level. For instance, the penalty of the initial scheduling shown in Figure 6a has a cost of $38{,}203$ time units. The differentiated penalty scheduling algorithm produces schedules that reduce this cost by 44.59%, to $21{,}168$ units. Consequently, the SLA penalty payable by the cloud service provider has also been improved by 39.92%, a reduction from 0.381 for the initial scheduling to 0.191 for the enhanced penalty-based scheduling.

In addition, the differentiated penalty-based scheduling demonstrates its effectiveness in optimizing financial performance by formulating cost-optimal schedules at the individual-queue level, as shown in Table 2 and Figure 7. For example, resource-3 (presented in Figure 7c) demonstrates the efficacy of the penalty-based scheduling in improving the penalty cost of the job schedule by 25%, a reduction in cost from $36{,}344$ to 27,126 time units. As a result, the performance of the differentiated penalty cost of the queue-state is optimized by 21.94%, reduced from 0.305 due to the initial scheduling order to reach 0.238 due to the improved differentiated penalty-based schedule.

Thus, the virtualized queue and segmented queue genetic solutions have efficiently explored a large solution search space using a small number of genetic iterations to achieve the enhancements. Figure 6c shows that the virtualized queue required a total of only $1{,}000$ genetic iterations to efficiently seek an optimal schedule of jobs in tier T_1, each iteration employs 10 chromosomes to evolve the optimal schedule. As such, 10×10^3 scheduling orders are constructed and genetically manipulated throughout the search space, as opposed to 25! (approximately 1.55×10^{25}) scheduling orders if a brute-force search strategy is employed to seek the optimal scheduling of jobs. Similar observations are in order with respect to the results reported on the segmented queue genetic solution.

Table 3. Total Differentiated Waiting Penalty

Differentiated Penalty Virtualized Queue	Differentiated Penalty Segmented Queue	Virtualized Queue	Segmented Queue	WLC	WRR
2423344	2709716	2976390	3004961	3652770	3899232

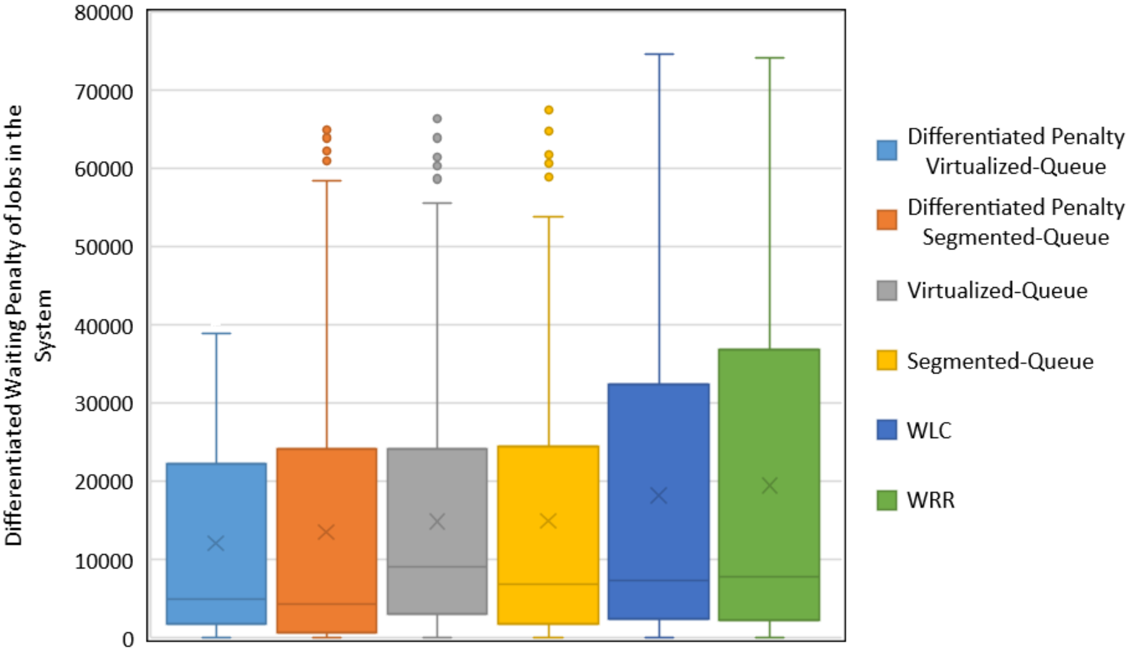
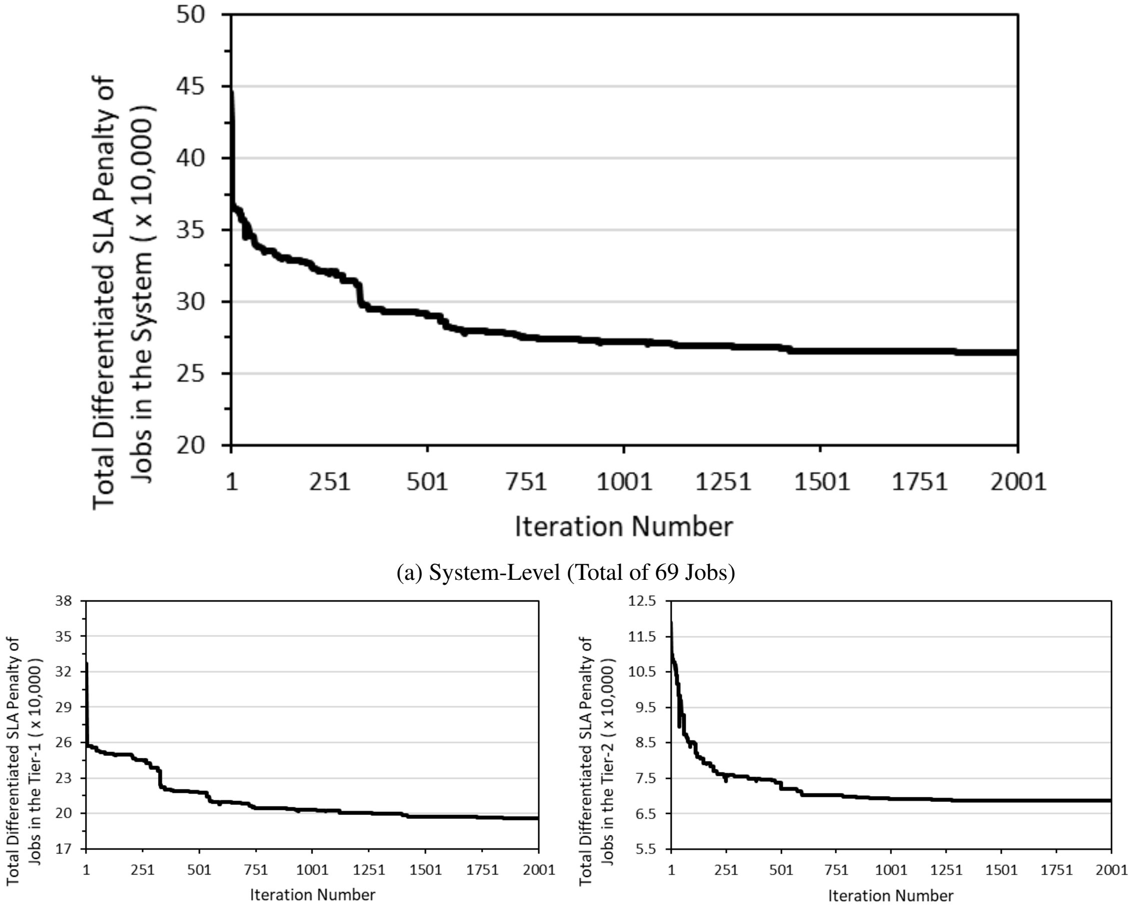


Figure 8. Maximum Differentiated Waiting Penalty Performance Comparison

To contrast the financial performance of the scheduling strategies, Table 3 and Figure 8 evaluate the differentiated service penalty cost. The WLC and WRR entail a cost of 3.65×10^6 and 3.9×10^6 time units, respectively. However, the virtualized queue and segmented queue scheduling approaches (without the service cost ψ_i) show superior performance compared with WLC and WRR, yet show inferior performance in improving the service penalty cost compared with the differentiated penalty-based scheduling approaches.

In fact, the differentiated penalty-based virtualized and segmented queue scheduling approaches produce schedules that improve service penalty cost. The differentiated penalty-based scheduling of the segmented queue genetic approach reduces the service penalty to a cost of 2.7×10^6 time units, demonstrating a superior performance compared with WLC and WRR. In contrast, the differentiated penalty-based scheduling of the virtualized queue genetic approach optimizes financial performance by reducing service penalty cost to 2.4×10^6, demonstrating the best financial performance compared with the other scheduling strategies.

Overall, single-tier-driven differentiated penalty scheduling produces schedules that enhance financial performance. The virtualized queue and segmented queue genetic approaches employed in the scheduling process demonstrate their effectiveness in efficiently facilitating the search for financially performance-optimal schedules at the tier level and individual queue level of the tier, respectively.



(a) System-Level (Total of 69 Jobs)

(b) Tier-1 (40 Jobs)

(c) Tier-2 (29 Jobs)

Figure 9. Differentiated SLA Penalty Multi-Tier $\omega\mathcal{AL}_i$ Based System Virtualized Queue Scheduling

5.2. Evaluation of Differentiated Scheduling: Multi-Tier Considerations

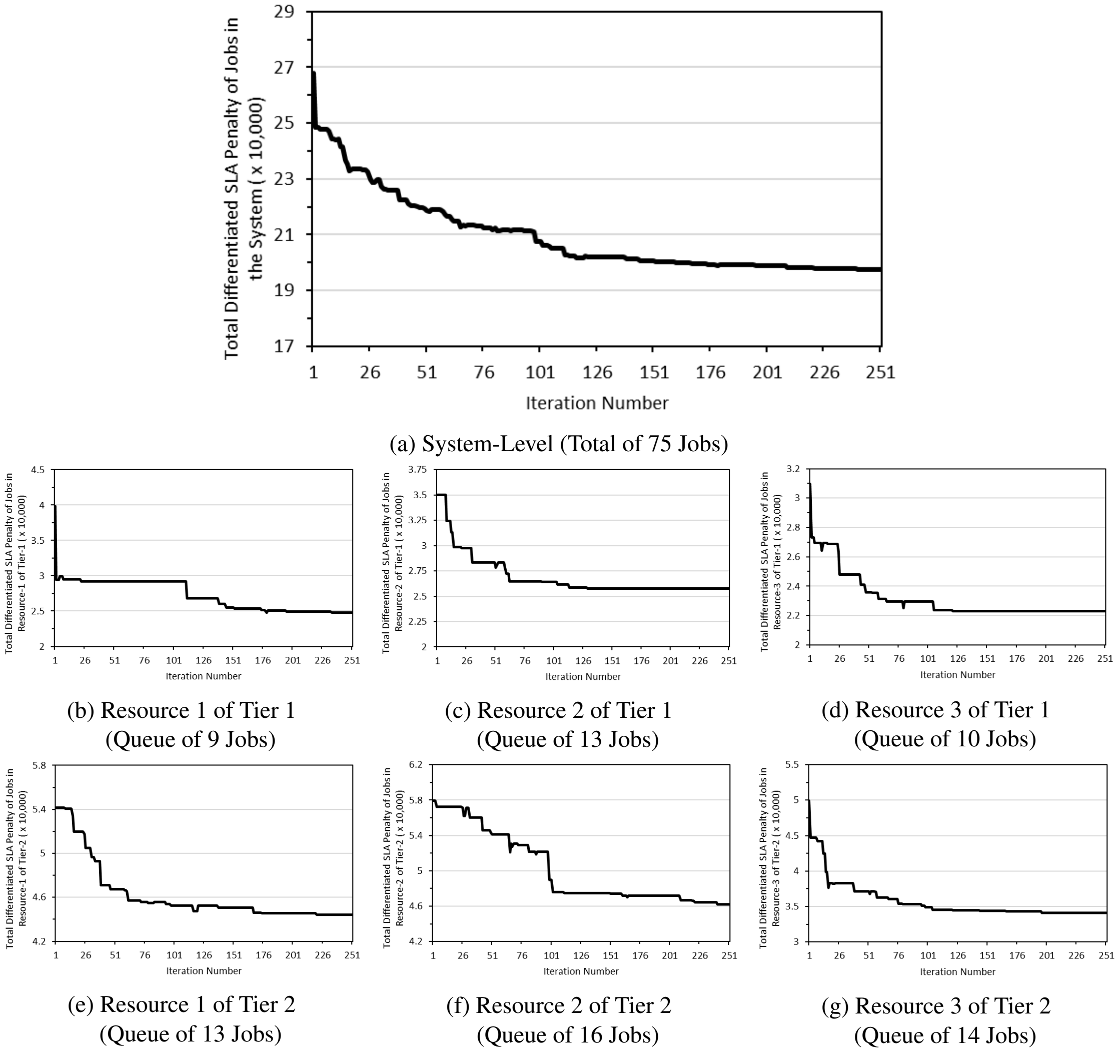
This is concerned with formulating performance-optimal schedules that produce a minimum differentiated SLA penalty at the multi-tier level. The experiments are conducted using the system virtualized queue and segmented queue genetic scheduling, explained in section 4.2. The QoS penalty function $f_{r,G}$ of the multi-tier genetic scheduling in Equation 30 is used. Thus, the penalty function evaluates the effectiveness of schedules to reach an optimal financial performance by minimizing the differentiated multi-tier SLA penalty.

Table 4. Differentiated SLA Penalty Multi-Tier $\omega\mathcal{AL}_i$ Based System Virtualized Queue Scheduling

	Number[1] of Jobs	Initial[2]		Enhanced[3]		Improvement	
		Violation	Penalty	Violation	Penalty	Violation %	Penalty %
System-Level, Figure 9a	69	446183	1.66	262387	1.35	41.19%	18.38%
Tier-1, Figure 9b	40	327232	0.96	193614	0.86	40.83%	11.05%
Tier-2, Figure 9c	29	118951	0.70	68773	0.50	42.18%	28.51%

[1] **Number of Jobs** represents the total number of jobs in queues of the tier/environment. The multi-tier environment contains 69 jobs in total. The 3 queues of tier-1 and tier-2 are allocated 40 and 29 jobs, respectively.
[2] **Initial Violation** represents the total SLA violation time of jobs according to their initial scheduling before using the system virtualized queue genetic solution.
[3] **Enhanced Violation** represents the total SLA violation time of jobs according to their final/enhanced scheduling found after using the system virtualized queue genetic solution.

The results shown in Table 4 and Figure 9 represent a system-state of a multi-tier environment that is allocated 69 jobs; 40 jobs are allocated to tier T_1 and 29 jobs are allocated to tier T_2. The differentiated multi-tier penalty $\omega\mathcal{AL}_i$ based scheduling of the system virtualized queue genetic approach has gradually



(a) System-Level (Total of 75 Jobs)

(b) Resource 1 of Tier 1 (Queue of 9 Jobs)

(c) Resource 2 of Tier 1 (Queue of 13 Jobs)

(d) Resource 3 of Tier 1 (Queue of 10 Jobs)

(e) Resource 1 of Tier 2 (Queue of 13 Jobs)

(f) Resource 2 of Tier 2 (Queue of 16 Jobs)

(g) Resource 3 of Tier 2 (Queue of 14 Jobs)

Figure 10. Differentiated SLA Penalty Multi-Tier $\omega\mathcal{AL}_i$ Based Segmented Queue Scheduling

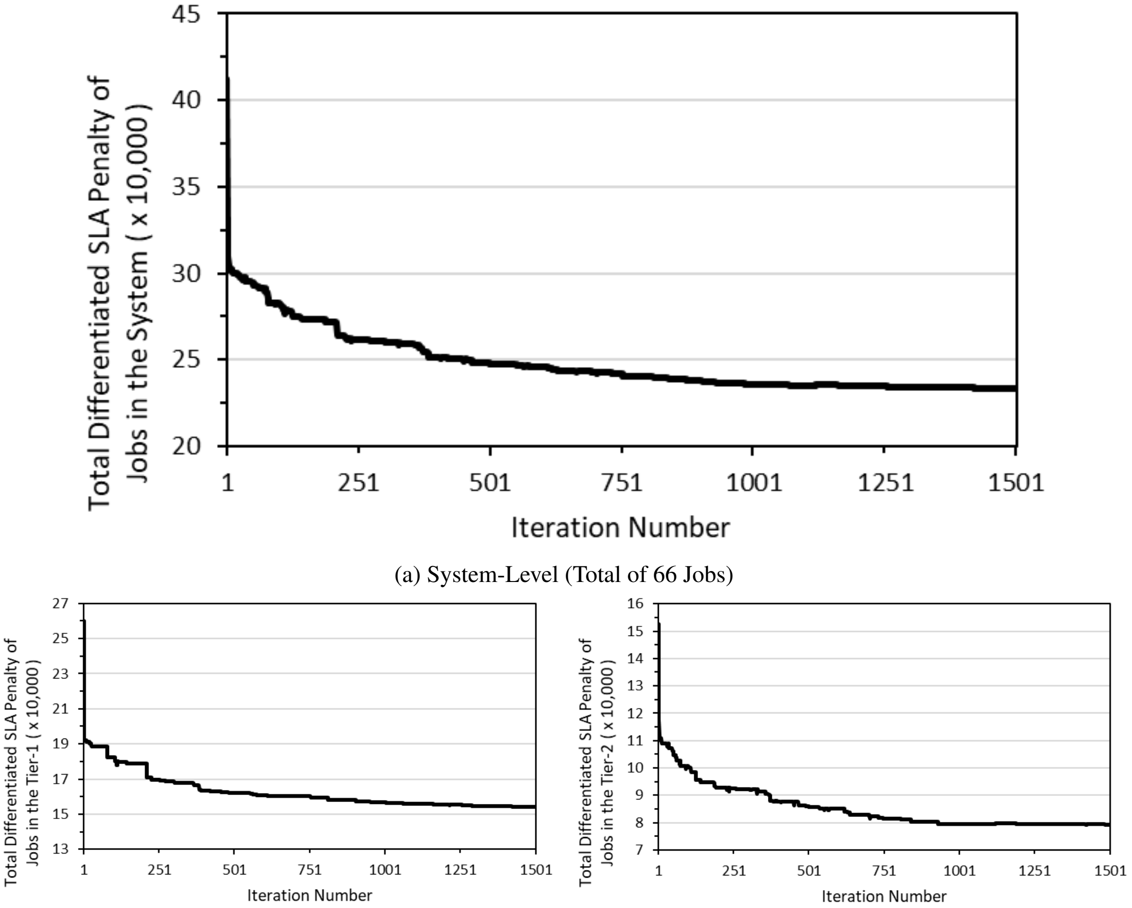
Table 5. Differentiated SLA Penalty Multi-Tier $\omega\mathcal{AL}_i$ Based Segmented Queue Scheduling

	Number	**Initial**[1]		**Enhanced**[2]		**Improvement**	
	of Jobs	**Violation**	**Penalty**	**Violation**	**Penalty**	**Violation %**	**Penalty %**
System-Level, Figure 10a	75	267775	2.14	196484	1.66	26.62%	22.60%
Resource-1 Tier-1, Figure 10b	9	39837	0.33	24775	0.22	37.81%	33.22%
Resource-2 Tier-1, Figure 10c	13	34988	0.30	25724	0.23	26.48%	23.17%
Resource-3 Tier-1, Figure 10d	10	30976	0.27	22281	0.20	28.07%	25.02%
Resource-1 Tier-2, Figure 10e	13	54131	0.42	44182	0.36	18.38%	14.56%
Resource-2 Tier-2, Figure 10f	16	57945	0.44	45633	0.37	21.25%	16.69%
Resource-3 Tier-2, Figure 10g	14	49899	0.39	33890	0.29	32.08%	26.83%

[1] **Initial Violation** represents the total SLA violation time of jobs according to their initial scheduling before using the segmented queue genetic solution.

[2] **Enhanced Violation** represents the total SLA violation time of jobs according to their final/enhanced scheduling found after using the segmented queue genetic solution.

reduced the SLA penalty cost. The differentiated $\omega\mathcal{AL}_i$ based scheduling genetic evaluation function in Equation 30 is employed. The financial performance of the system-state is optimized by 41.19%, through formulating an enhanced cost-optimal schedule that reduces the SLA penalty from a cost of $446{,}183$ time units for the initial schedule to a cost of $262{,}387$ time units for the improved schedule computed at the multi-tier level. As such, the differentiated SLA penalty cost payable by the cloud service provider has been improved by 18.38%, a reduction in the penalty from 1.66 for the initial schedule to 1.35 for the improved cost-optimal schedule of the system-state.



(a) System-Level (Total of 66 Jobs)

(b) Tier-1 (35 Jobs)

(c) Tier-2 (31 Jobs)

Figure 11. Differentiated SLA Penalty $\omega\mathcal{PT}_{i,j}$ Based System Virtualized Queue Scheduling

Table 6. Differentiated SLA Penalty $\omega\mathcal{PT}_{i,j}$ Based System Virtualized Queue Scheduling

	Number[1]	**Initial**[2]		**Enhanced**[3]		**Improvement**	
	of Jobs	**Violation**	**Penalty**	**Violation**	**Penalty**	**Violation %**	**Penalty %**
System-Level, Figure 11a	66	412442	1.71	232573	1.33	43.61%	22.05%
Tier-1, Figure 11b	35	259880	0.93	153300	0.78	41.01%	15.29%
Tier-2, Figure 11c	31	152562	0.78	79273	0.55	48.04%	30.05%

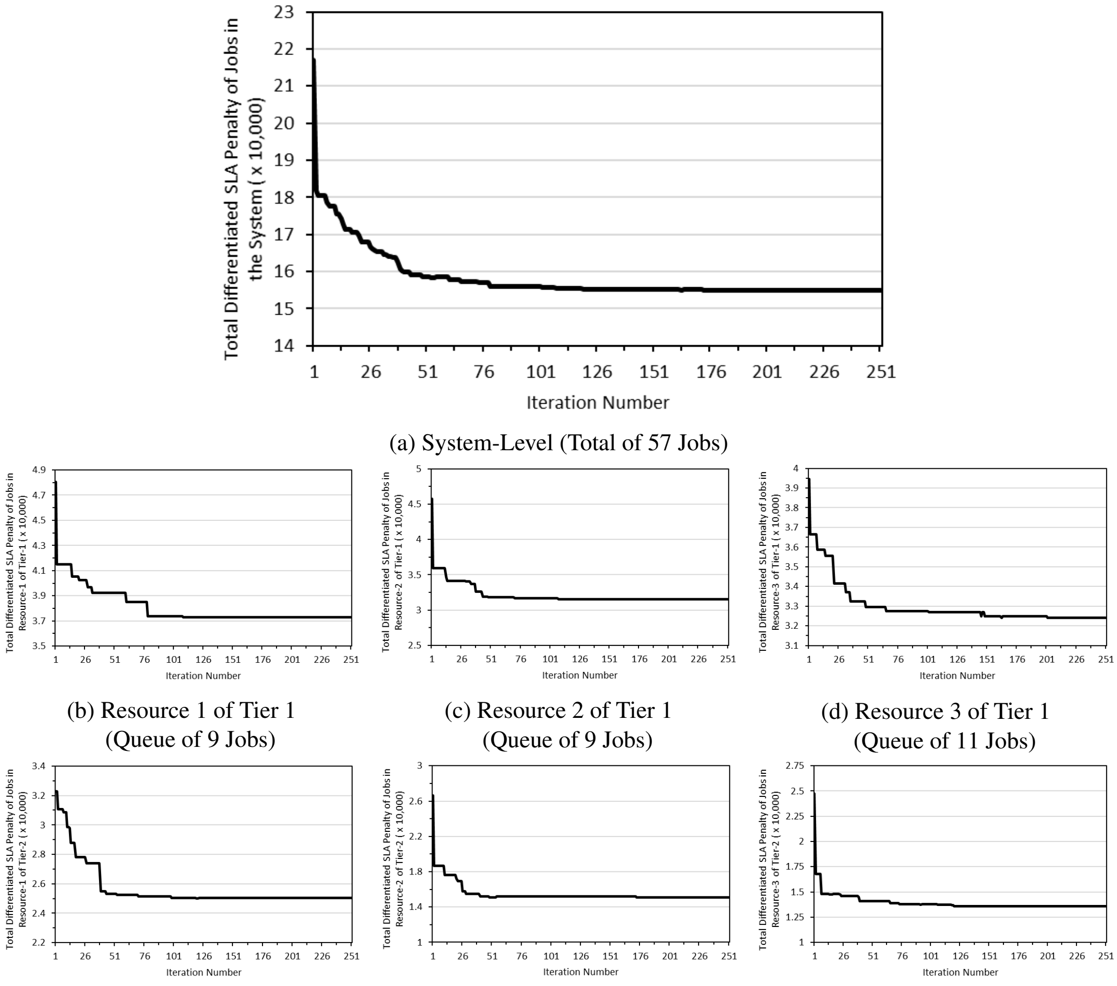
[1] **Number of Jobs** represents the total number of jobs in queues of the tier/environment. The multi-tier environment is allocated 66 jobs in total. The 3 queues of tier-1 and tier-2 are allocated 35 and 31 jobs, respectively.

[2] **Initial Violation** represents the total SLA violation time of jobs according to their initial scheduling before using the system virtualized queue genetic solution.

[3] **Enhanced Violation** represents the total SLA violation time of jobs according to their final/enhanced scheduling found after using the system virtualized queue genetic solution.

Similarly, the differentiated multi-tier penalty $\omega\mathcal{AL}_i$ based scheduling of the segmented queue genetic approach shows an improved financial performance on the system-state. Cost-optimal schedules are formulated in each individual queue to efficiently reduce the differentiated SLA penalty cost at the multi-tier level, as shown in Table 5 and Figure 10. In a multi-tier environment allocated 75 jobs, the differentiated SLA penalty improves by 22.6% at the multi-tier level. The SLA penalty cost of the system-state has been reduced from 2.14 for the initial schedule to reach 1.66 for the cost-optimal schedule.

In the same way, the financial performance of the differentiated multi-tier penalty $\omega\mathcal{PT}_{i,j}$ based scheduling of the system virtualized queue genetic approach corroborates the financial performance of the former differentiated penalty $\omega\mathcal{AL}_i$ based scheduling. Cost-optimal schedules at the multi-tier level are also produced by the differentiated multi-tier penalty $\omega\mathcal{PT}_{i,j}$ based scheduling of the segmented queue genetic approach, which as well corroborates the financial performance of the differentiated multi-tier penalty $\omega\mathcal{AL}_i$ based scheduling of the segmented queue genetic approach.



(a) System-Level (Total of 57 Jobs)

(b) Resource 1 of Tier 1 (Queue of 9 Jobs)

(c) Resource 2 of Tier 1 (Queue of 9 Jobs)

(d) Resource 3 of Tier 1 (Queue of 11 Jobs)

(e) Resource 1 of Tier 2 (Queue of 10 Jobs)

(f) Resource 2 of Tier 2 (Queue of 8 Jobs)

(g) Resource 3 of Tier 2 (Queue of 10 Jobs)

Figure 12. Differentiated SLA Penalty $\omega\mathcal{PT}_{i,j}$ Based Segmented Queue Scheduling

Table 7. Differentiated SLA Penalty $\omega\mathcal{PT}_{i,j}$ Based Segmented Queue Scheduling

	Number of Jobs	**Initial[1]**		**Enhanced[2]**		**Improvement**	
		Violation	**Penalty**	**Violation**	**Penalty**	**Violation %**	**Penalty %**
System-Level, Figure 12a	57	216897	1.80	154844	1.35	28.61%	25.35%
Resource-1 Tier-1, Figure 12b	9	48050	0.38	37272	0.31	22.43%	18.45%
Resource-2 Tier-1, Figure 12c	9	45753	0.37	31513	0.27	31.12%	26.38%
Resource-3 Tier-1, Figure 12d	11	39447	0.33	32400	0.28	17.87%	15.10%
Resource-1 Tier-2, Figure 12e	10	32291	0.28	24992	0.22	22.60%	19.87%
Resource-2 Tier-2, Figure 12f	8	26630	0.23	15065	0.14	43.43%	40.18%
Resource-3 Tier-2, Figure 12g	10	24726	0.22	13601	0.13	44.99%	41.95%

[1] **Initial Violation** represents the total SLA violation time of jobs according to their initial scheduling before using the segmented queue genetic solution.

[2] **Enhanced Violation** represents the total SLA violation time of jobs according to their final/enhanced scheduling found after using the segmented queue genetic solution.

For instance, the SLA penalty of the system-state shown in Table 6 and Figure 11 is optimized at the multi-tier level by 22.05%, a reduction in the SLA penalty cost from 1.71 for the initial schedule to reach 1.33 for the improved schedule efficiently computed by the differentiated multi-tier penalty $\omega\mathcal{PT}_{i,j}$ based scheduling of the system virtualized queue genetic approach. In addition, the differentiated multi-tier penalty $\omega\mathcal{PT}_{i,j}$ based scheduling of the segmented queue genetic approach improves the financial performance of the SLA penalty by 25.35% at the multi-tier level, which reduces the SLA penalty cost of the system-state from 1.8 for the initial schedule to 1.35 for the enhanced schedule shown in Table 7 and Figure 12.

A comparison of the financial performance of the differentiated penalty-based scheduling strategies in optimizing the differentiated SLA penalty cost at the multi-tier level is presented in Table 8 and Figure 13. The differentiated multi-tier penalty $\omega\mathcal{AL}_i$ based and $\omega\mathcal{PT}_{i,j}$ based scheduling efficiently produce optimal schedules that reduce the SLA penalty cost, using the system virtualized queue and segmented queue genetic scheduling solutions. However, compared with the differentiated service penalty scheduling approaches, the multi-tier $\omega\mathcal{AL}_i$ based and $\omega\mathcal{PT}_{i,j}$ based scheduling approaches demonstrate a superior performance in reducing the SLA penalty cost.

Table 8. Total Differentiated SLA Penalty

Differentiated Penalty Multi-Tier $\omega\mathcal{PT}_{i,j}$ Based Scheduling		**Differentiated Penalty Multi-Tier $\omega\mathcal{AL}_i$ Based Scheduling**		**Multi-Tier $\omega\mathcal{PT}_{i,j}$ Based Scheduling**		**Multi-Tier $\omega\mathcal{AL}_i$ Based Scheduling**		**WLC**	**WRR**
System Virtualized Queue	**Segmented Queue**	**System Virtualized Queue**	**Segmented Queue**	**System Virtualized Queue**	**Segmented Queue**	**System Virtualized Queue**	**Segmented Queue**		
1431984	1800853	1589481	1897843	2074843	2521244	2228040	2692282	3559464	3805631

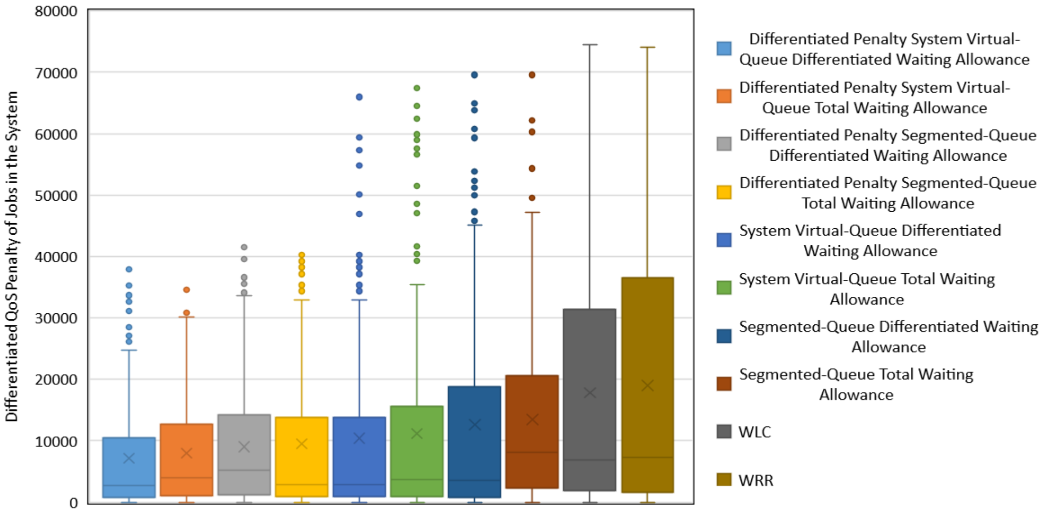


Figure 13. Comparison of the Approaches

Differentiated multi-tier penalty $\omega\mathcal{AL}_i$ based scheduling of the segmented queue genetic approach reduces the SLA penalty by approximately 47% compared with WLC and 50% compared with WRR; however, it shows an inferior financial performance compared with the differentiated multi-tier penalty $\omega\mathcal{PT}_{i,j}$ based scheduling of the segmented queue genetic approach. In contrast, differentiated multi-tier penalty $\omega\mathcal{AL}_i$ based scheduling of the system virtualized queue genetic approach produces schedules that entail a cost of 1.59×10^6 time units of the SLA penalty at the multi-tier level, a reduction of 55% and 58% compared with WLC and WRR strategies, respectively. Superior financial performance is demonstrated in the differentiated multi-tier penalty $\omega\mathcal{PT}_{i,j}$ based scheduling of the system virtualized queue genetic approach, which produces schedules that reduce the SLA penalty to around a cost of 1.43×10^6 time units.

6. Conclusion

This paper presents a QoS-driven scheduling approach to address the differentiated penalty of delay-intolerant jobs in a multi-tier cloud computing environment. The approach emphasizes the notion of financial penalty in scheduling client jobs so that schedules are effectively produced based on economic considerations. Job treatment regimes are devised in a differentiated QoS penalty model, so as the cloud service provider computes schedules that capture the financial impact of SLA violation penalty on the QoS provided. Optimal performance is delivered to clients who cannot afford the cost of SLA violations and delays.

The proposed queue virtualization design schemes facilitate the formulation of schedules at the tier and multi-tier levels of the cloud computing environment. The design schemes leverage the utilization of resources within a tier to derive tier-driven schedules with optimal performance, as well as employ dependencies and bottleneck shifting between tiers to formulate multi-tier-driven schedules with optimal performance. The proposed meta-heuristic approaches, represented by the differentiated penalty virtualized-queue and segmented-queue genetic solutions, reduce the complexity of optimal scheduling of jobs on resource queues of the tiers.

The formulated cost-optimal schedules reduce the cost of SLA penalty for client jobs, which accordingly maximizes client satisfactions and thus loyalties to the cloud service provider. The produced schedules maintain a balance between delivering the highest QoS provided to clients while ensuring an efficient system performance with a reduced operational cost, and thus fulfilling the different QoS expectations and mitigating their associated commercial penalties. It is shown that the financial performance has been improved by reducing the QoS penalty under different SLA commitments of client jobs in a multi-tier cloud computing environment.

7. Future Work

A cloud service provider employs multiple resources that typically demand a huge amount of energy to execute various client demands. Due to its impact on system performance, energy saving has recently become of paramount importance in cloud computing. However, a major challenge on a cloud service provider is maintaining a maximum energy efficiency (minimum consumption) while ensuring high system performance that fulfills the different QoS expectations in executing client jobs of varying computational demands. Any imbalance in managing these conflicting objectives may result in failing to meet SLA obligations of clients and, thus, financial penalties on the cloud service provider. Accordingly, it is imperative to devise scheduling approaches that produce energy-efficient optimal schedules with minimal SLA penalties at the multi-tier level. A sustainable cloud computing environment would help reduce the energy cost required to execute client demands.

References

[1] A. Shawish and M. Salama, *Cloud Computing: Paradigms and Technologies*. Springer, 2014, pp. 39–67.

[2] I. Chana and S. Singh, “Quality of service and service level agreements for cloud environments: Issues and challenges,” in *Cloud Computing: Challenges, Limitations and R&D Solutions*. Springer, 2014, pp. 51–72.

[3] J. Vuong, “Disaster recovery planning,” in *Proceedings of the Information Security Curriculum Development Conference*, October 2015, pp. 1–3.

[4] H. Suleiman and O. Basir, “Service level driven job scheduling in multi-tier cloud computing: A biologically inspired approach,” in *Proceedings of the International Conference on Cloud Computing: Services and Architecture*, July 2019, pp. 99–118.

[5] ——, “QoS-driven job scheduling: Multi-tier dependency considerations,” in *Proceedings of the International Conference on Cloud Computing: Services and Architecture*, July 2019, pp. 133–155.

[6] O. Rana, M. Warnier, T. Quillinan, F. Brazier, and D. Cojocarasu, *Managing Violations in Service Level Agreements*. Springer, 2008, pp. 349–358.

[7] M. Cochran and P. Witman, “Governance and service level agreement issues in a cloud computing environment,” *Journal of Information Technology Management*, vol. 22, no. 2, pp. 41–55, 2011.

[8] J.-H. Morin, J. Aubert, and B. Gateau, “Towards cloud computing SLA risk management: Issues and challenges,” in *Proceedings of the Hawaii International Conference on System Sciences*, January 2012, pp. 5509–5514.

[9] G. Singh and S. Prakash, “A review on quality of service in cloud computing,” in *Big Data Analytics*, October 2018, pp. 739–748.

[10] A. Thakur and M. Goraya, “A taxonomic survey on load balancing in cloud,” *Journal of Network and Computer Applications*, vol. 98, no. 11, pp. 43–57, 2017.

[11] S. Shaw and A. Singh, “A survey on scheduling and load balancing techniques in cloud computing environment,” in *Proceedings of the International Conference on Computer and Communication Technology*, September 2014, pp. 87–95.

[12] A. Abdelmaboud, D. Jawawi, I. Ghani, A. Elsafi, and B. Kitchenham, “Quality of service approaches in cloud computing: A systematic mapping study,” *Journal of Systems and Software*, vol. 101, no. 3, pp. 159–179, 2015.

[13] Y. Jia, I. Brondino, R. J. Peris, M. P. Martinez, and D. Ma, “A multi-resource load balancing algorithm for cloud cache systems,” in *Proceedings of the Annual ACM Symposium on Applied Computing*, March 2013, pp. 463–470.

[14] C.-C. Yang, K.-T. Chen, C. Chen, and J.-Y. Chen, “Market-based load balancing for distributed heterogeneous multi-resource servers,” in *Proceedings of the International Conference on Parallel and Distributed Systems*, December 2009, pp. 158–165.

[15] G. Stavrinides and H. Karatza, “The effect of workload computational demand variability on the performance of a SaaS cloud with a multi-tier SLA,” in *Proceedings of the IEEE International Conference on Future Internet of Things and Cloud*, August 2017, pp. 10–17.

[16] H. Moon, Y. Chi, and H. Hacigumus, “Performance evaluation of scheduling algorithms for database services with soft and hard SLAs,” in *Proceedings of the Second International Workshop on Data Intensive Computing in the Clouds*, November 2011, pp. 81–90.

[17] H. Chen, F. Wang, N. Helian, and G. Akanmu, “User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing,” in *Proceedings of the National Conference on Parallel Computing Technologies*, February 2013, pp. 1–8.

[18] S. Nayak, S. Parida, C. Tripathy, and P. Pattnaik, “An enhanced deadline constraint based task scheduling mechanism for cloud environment,” *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, 2018.

[19] K. Lee, R. Pedarsani, and K. Ramchandran, “On scheduling redundant requests with cancellation overheads,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1279–1290, 2017.

[20] R. Birke, J. Perez, Z. Qiu, M. Bjorkqvist, and L. Chen, “Power of redundancy: Designing partial replication for multi-tier applications,” in *Proceedings of the IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[21] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, E. Hyytia, and A. Scheller-Wolf, “Queueing with redundant requests: Exact analysis,” *Queueing Systems: Theory and Applications*, vol. 83, no. 3-4, pp. 227–259, 2016.

[22] R. Mailach and D. Down, “Scheduling jobs with estimation errors for multi-server systems,” in *Proceedings of the International Teletraffic Congress*, September 2017, pp. 10–18.

[23] M. Okopa and H. Okii, “Fixed priority SWAP scheduling policy with differentiated services under varying job size distributions,” in *Proceedings of the Second International Conference on Digital Information Processing and Communications*, July 2012, pp. 168–173.

[24] S. Panda, S. Pande, and S. Das, “Task partitioning scheduling algorithms for heterogeneous multi-cloud environment,” *Arabian Journal for Science and Engineering*, vol. 43, no. 2, pp. 913–933, 2018.

[25] S. Panda, S. Nanda, and S. Bhoi, “A pair-based task scheduling algorithm for cloud computing environment,” *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, 2018.

[26] S. Panda and P. Jana, “Efficient task scheduling algorithms for heterogeneous multi-cloud environment,” *The Journal of Supercomputing*, vol. 71, no. 4, pp. 1505–1533, April 2015.

[27] I. Moschakis and H. Karatza, “Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing,” *Journal of Systems and Software*, vol. 101, pp. 1–14, 2015.

[28] D. Chaudhary and B. Kumar, “Analytical study of load scheduling algorithms in cloud computing,” in *Proceedings of the International Conference on Parallel, Distributed and Grid Computing*, December 2014, pp. 7–12.

[29] M. Rana, S. Bilgaiyan, and U. Kar, “A study on load balancing in cloud computing environment using evolutionary and swarm based algorithms,” in *Proceedings of the International Conference on Control, Instrumentation, Communication and Computational Technologies*, July 2014, pp. 245–250.

[30] P. Singh, M. Dutta, and N. Aggarwal, “A review of task scheduling based on meta-heuristics approach in cloud computing,” *Knowledge and Information Systems*, vol. 52, no. 1, pp. 1–51, 2017.

[31] S. Mishra, B. Sahoo, and P. Parida, “Load balancing in cloud computing: A big picture,” *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 1, 2018.

[32] R. Babu, A. Joy, and P. Samuel, “Load balancing of tasks in cloud computing environment based on bee colony algorithm,” in *Proceedings of the International Conference on Advances in Computing and Communications*, September 2015, pp. 89–93.

[33] R. Babu and P. Samuel, “Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud,” in *Proceedings of the International Conference on Innovations in Bio-Inspired Computing and Applications*, July 2016, pp. 67–78.

[34] R. Gautam and S. Arora, “Cost-based multi-QoS job scheduling algorithm using genetic approach in cloud computing environment,” *International Journal of Advanced Science and Research*, vol. 3, no. 3, pp. 110–115, 2018.

[35] Y. Wang, J. Wang, C. Wang, and X. Song, “Resource scheduling of cloud with QoS constraints,” in *Proceedings of the International symposium on Neural Networks*, July 2013, pp. 351–358.

[36] K. Boloor, R. Chirkova, T. Salo, and Y. Viniotis, “Heuristic-based request scheduling subject to a percentile response time SLA in a distributed cloud,” in *Proceedings of the IEEE Global Telecommunications Conference*, December 2010, pp. 1–6.

[37] Z.-H. Zhan, G.-Y. Zhang, Y. Lin, Y.-J. Gong, and J. Zhang, “Load balance aware genetic algorithm for task scheduling in cloud computing,” in *Proceedings of the International Asian-Pacific Simulated Evolution and Learning*, December 2014, pp. 644–655.

[38] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, “A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing,” *Neural Computing and Applications*, vol. 26, no. 6, pp. 1297–1309, 2015.

[39] G. Reig, J. Alonso, and J. Guitart, “Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud,” in *Proceedings of the IEEE International Symposium on Network Computing and Applications*, July 2010, pp. 162–167.

[40] I. Menache, S. Perez-Salazar, M. Singh, and A. Toriello, “Dynamic resource allocation in the cloud with near-optimal efficiency,” *arXiv preprint arXiv:1809.02688*, vol. abs/1809.02688, 2018.

[41] Y. Xiaomei, Z. Jianchao, L. Jiye, and L. Jiahua, “A genetic algorithm for job shop scheduling problem using co-evolution and competition mechanism,” in *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence*, October 2010, pp. 133–136.

[42] X. Li and L. Gao, “An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem,” *International Journal of Production Economics*, vol. 174, no. 4, pp. 93–110, 2016.

[43] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. Ammari, “An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem,” *Journal of Intelligent Manufacturing*, vol. 29, no. 3, pp. 603–615, 2018.