An Extensive Exploration of Techniques for Resource and Cost Management in Contemporary Cloud Computing Environments

Raghava Satya SaiKrishna Dittakavi
Independent Researcher, USA

Abstract
Resource and cost optimization techniques in cloud computing environments target minimizing expenditure while ensuring efficient resource utilization. This study categorizes these techniques into three primary groups: Cloud and VM-focused strategies, Workflow techniques, and Resource Utilization and Efficiency techniques. Cloud and VM-focused strategies predominantly concentrate on the allocation, scheduling, and optimization of resources within cloud environments, particularly virtual machines. These strategies aim at a balance between cost reduction and adhering to specified deadlines, while ensuring scalability and adaptability to different cloud models. However, they may introduce complexities due to their dynamic nature and continuous optimization requirements. Workflow techniques emphasize the optimal execution of tasks in distributed systems. They address inconsistencies in Quality of Service (QoS) and seek to enhance the reservation process and task scheduling. By employing models, such as Integer Linear Programming, these techniques offer precision. But they might be computationally demanding, especially for extensive problems. Techniques focusing on Resource Utilization and Efficiency attempts to maximize the use of available resources in an energy-efficient and cost-effective manner. Considering factors like current energy levels and application requirements, these models aim to optimize performance without overshooting budgets. However, a continuous monitoring mechanism might be necessary, which can introduce additional complexities.

Keywords: Cloud Computing, Cost Optimization, Resource Utilization, VM-focused Strategies, Workflow Techniques, Energy Efficiency

---

Introduction
Cloud computing refers to the delivery of various computing services, such as storage, processing, and networking, over the internet [1]. This infrastructure paradigm shift enables organizations and individuals to access and utilize computational resources without the need for owning or managing the underlying hardware. Instead, these resources are provisioned and managed by third-party entities, often referred to as cloud service providers. The offered services can be broadly segmented into several categories, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), each serving distinct operational needs. The significance of cloud computing in today's technological environment is

profound. As digital transformation becomes increasingly important for businesses, the agility, flexibility, and scalability provided by cloud environments have become necessary [2]. Not only does it eliminate the capital expenditure associated with procuring and maintaining physical infrastructures, but it also ensures that businesses can rapidly adapt to changing market demands. Additionally, with the rise of big data, artificial intelligence, and the Internet of Things (IoT), the ability to process vast datasets in real-time necessitates the computational prowess that cloud platforms readily offer [3].

Inadequate resource management in cloud environments brings about multiple complications. Performance issue emerge when resources are not allocated efficiently, causing application delays, downtimes, or in severe cases, crashes. Such interruptions can have detrimental effects on business operations, leading to potential revenue losses and a damaged brand reputation. Conversely, allocating excess resources beyond requirements results in wastage. Even though this approach might prevent immediate operational issues, it causes organizations to incur costs for unused services. The inherent scalability and flexibility of cloud computing, if not managed proficiently, can become problematic. Absent a strategic resource management approach, organizations may find themselves in a cycle of constant adjustments, leading to operational inefficiencies and unpredictable expenditures. Furthermore, uncontrolled expenses in cloud settings can diminish the cost-saving benefits that initially attract organizations to the cloud. In a competitive business environment, maintaining a balance between operational performance and cost efficiency is not merely a technical challenge but a foundational element for sustained success.

Workflows play an integral role in the framework of scientific applications within distributed systems. Conventionally, these workflows are delineated by a Directed Acyclic Graph (DAG) [4]–[6] . In this model, each computational task is symbolized as a node, whereas each data or control dependency that exists between these tasks is denoted by a directed edge connecting the respective nodes. Given the paramount significance of these workflow applications, numerous Grid projects have emerged to address this need. These systems are crafted to not only define but also manage and execute workflows on the Grid.

The success witnessed in these Grid projects has subsequently paved the way for new research directions. One of the prominent areas of exploration is the feasibility and efficiency of running large-scale scientific workflows on Cloud-based systems. The transition to Cloud systems brings about potential advantages, such as scalability, flexibility, and cost-effectiveness. As distributed computing evolves, the Cloud emerges as a vital infrastructure that can potentially support the requirements of scientific workflows, given its inherent attributes.

With this transition towards the Cloud, several recent versions of Grid workflow management systems have incorporated features that facilitate this shift. These modifications and adaptations indicate a trend wherein Cloud systems are increasingly being viewed as viable platforms for executing large-scale scientific workflows, reflecting the evolutionary trajectory of distributed computing systems.

Scientific Workflow Scheduling (SWFS) refers to the process of mapping and scheduling tasks or jobs from a scientific workflow onto distributed computing resources to achieve specific objectives, such as minimizing execution time or cost [7], [8]. Scientific workflows are structured, organized sequences of computational tasks designed to achieve a particular scientific goal. These tasks might be data processing, simulations, or complex analyses that require multiple computational steps. Given the heterogeneous nature of distributed computing resources—such as different types of processors, varying amounts of memory, and diverse network bandwidths—SWFS aims to efficiently allocate these tasks to the available resources. The challenges in SWFS include handling task dependencies [9], [10], optimizing for multiple objectives, dealing with resource failures or uncertainties, and addressing the dynamic nature of resources and tasks. Efficient SWFS can lead to reduced turnaround times and more optimal use of computational resources, both of which are crucial for advancing scientific discoveries in various domains.

Scientific Workflow Scheduling (SWFS) in cloud and grid computing poses intricate challenges, particularly when focusing on cost optimization during workflow execution [11], [12]. One core element of this challenge is addressing the needs of different users who, more often than not, find themselves in competition for limited resources within the cloud or grid computing environments. This competition is driven by the need to satisfy Quality of Service (QoS) constraints. These constraints ensure that the services provided meet certain standards, which, in turn, can influence the overall cost of operations.

Further complicating the issue of cost optimization in SWFS is the presence of inter-dependencies among workflow tasks. These inter-dependencies mandate a coordinated and well-orchestrated execution of tasks to ensure the seamless flow and successful completion of workflows. Any disruptions or inefficiencies can lead to increased costs. An additional challenge arises from the high communication costs associated with these inter-dependencies. Specifically, as tasks rely on one another, there's a necessity for data transfer between different resources. This data transfer, especially if frequent or involving large volumes of data, can lead to substantial communication costs.

Resource and Cost Optimization Techniques in cloud computing environments can be categorized into three principal categories: Cloud and VM-focused strategies, Workflow techniques, and Resource Utilization and Efficiency techniques. The primary objective of the study is to conduct an in-depth analysis and categorization of the various techniques used for resource and cost optimization in cloud computing environments. The research attempts to discuss the methodologies and strategies in use, presented in Table 1, evaluating their advantages, challenges, and specific applications. We believe that dissecting these techniques into distinct groups based on their functionality and focus, we can offer clarity and a structured overview of the domain.

Table 1. Contemporary Techniques for Resource and Cost Management

Technique	Description
SaaS Cloud Partial Critical Paths (SCPCP) [13]	Identifies critical paths within a workflow in the SaaS cloud model, optimizing costs without violating specified deadlines.
Hybrid Cloud Optimized Cost model (HCOC) [14]	Operates in hybrid cloud environments, scheduling workflows to minimize costs while meeting set deadlines.
Scalable Heterogeneous Earliest-Finish-Time algorithm [15]	Prioritizes tasks for early completion in heterogeneous environments, offering scalability and resource adjustment based on demands.
Workflow-aware Preprocessing Provisioning Dynamic Scheduling (WPPDS) [16]	Tailored for scientific workflows, incorporates preprocessing and dynamic provisioning for optimal task scheduling.
Dynamic Provisioning Dynamic Scheduling [17]	Focuses on flexibility, adjusting resource allocation and task scheduling based on current demands and conditions.
Workflow Orchestrator for Distributed Systems Architecture [18]	Addresses inconsistencies in Quality of Service (QoS) features in distributed systems, optimizing batch queues for better performance.
Multi-cost job routing and scheduling [19]	Enhances resource reservation by providing insights into data transmission and task completion timing while considering multiple costs.
Integer Linear Programming (ILP) technique [20]	Utilizes linear equations and inequalities to find optimal solutions for scheduling challenges, known for precision but computationally demanding.
Compatibility of Hybrid Processor Scheduler [21]	Selects resources based on their energy levels and application requirements, ensuring smart choices for optimal resource utilization.
Partitioning-Based Workflow Scheduling (PBWS) [22], [23]	Uses partitioned Directed Acyclic Graph (DAG) to minimize communication overheads, reducing both communication costs and workflow execution time.

Cloud and VM-focused Strategies

SaaS Cloud Partial Critical Paths (SCPCP)

The SaaS Cloud Partial Critical Paths (SCPCP) introduces a distinctive approach to optimizing workflow execution in a cloud environment. Specifically designed for the Software as a Service (SaaS) model, it centers on the concept of Partial Critical Paths (PCP). The primary goal of this algorithm is to strike a balance between cost-efficiency and adherence to a user-defined deadline. By identifying the critical paths within a workflow, SCPCP seeks to streamline costs without violating the set deadlines. To achieve this, the algorithm takes an iterative approach, scheduling the partial critical paths that culminate at tasks scheduled in prior iterations.

A feature of the SCPCP method is its scheduling approach, which is fundamentally retrogressive. Rather than scheduling tasks in a linear or forward direction, this technique works backward. The inherent nature of this reverse scheduling process means that there may be occasions when a task within a partial critical path cannot be scheduled appropriately. In such cases, additional constraints are incorporated into the scheduling process. This forces the algorithm to restart and attempt the scheduling process anew. An observation reveals that the

SCPCP has characteristics in common with the Maximum Dot Product (MDP) method. However, an intrinsic challenge with the SCPCP method is its time complexity. Given the backward scheduling nature and the potential for multiple rescheduling events, the algorithm can demand significant processing time during its execution.

SCPCP algorithm

```
Begin
  Determine available computation services
  Add nodes tentry and texit to G with their dependencies
  **For** each task ti in G **do**
    Compute EST(ti), EFT(ti), and LFT(ti)
  **EndFor**
  AST(tentry) = 0
  AST(texit) = D
  Mark tentry and texit as assigned
  AllocateAncestors(texit)
End
Algorithm AllocateAncestors(t):
Begin
  While t has an unassigned parent do
    PCP = empty list
    ti = t
    While ti has an unassigned parent do
      Add CriticalParent(ti) to the beginning of PCP
      ti = CriticalParent(ti)
    EndWhile
    AssignPath(PCP)
    For each task ti in PCP do
      Update EST and EFT for all successors of ti
      Update LFT for all predecessors of ti
      AllocateAncestors(ti)
    EndFor
  EndWhile
End
```

There is the potential for optimal cost management within a SaaS cloud model. By focusing on critical paths and strategically scheduling them, the algorithm ensures that the workflow execution is cost-effective. Secondly, the SCPCP method is rigorous about meeting deadlines. By its design, it ensures that tasks, especially those on the critical path, are executed within the user-defined timeframes, thereby assuring quality of service. One primary concern is its specificity. Given that it has been tailored predominantly for the SaaS model, its applicability to other cloud models or environments may be limited. This restricts its versatility in diverse cloud settings. Additionally, while the algorithm excels at scheduling and executing critical paths, it might not always deliver the most optimal results for tasks outside these critical paths. The potential for suboptimal execution for non-critical paths can, in some scenarios, affect the overall efficiency of the workflow.

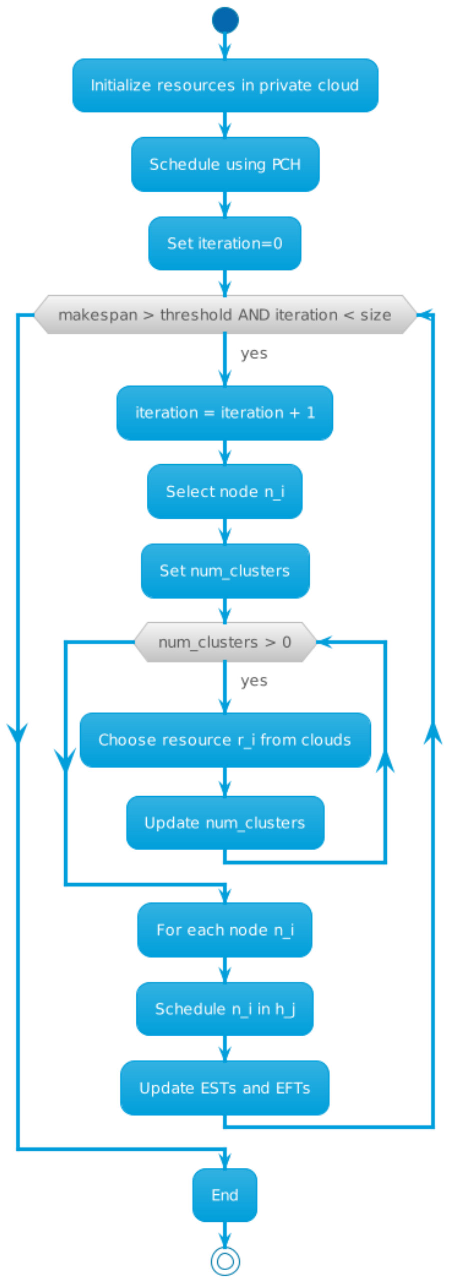
Hybrid Cloud Optimized Cost model (HCOC)

The Hybrid Cloud Optimized Cost (HCOC) scheduling algorithm is designed for hybrid cloud environments. Its primary goal is to accelerate the execution of workflows to meet a specific execution time and to do so at a reduced cost compared to standard greedy algorithms. The HCOC algorithm's effectiveness in a hybrid cloud context comes from its ability to understand and utilize multicore systems. Additionally, it incorporates cost considerations into its scheduling decisions. This combination allows users to influence costs by determining the desired workflow execution time based on their preferences. The algorithm can reduce expenses in the public cloud as the specified workflow execution time increases. Even when the desired execution time is particularly short, HCOC has been shown to produce more efficient schedules than the greedy approach by leveraging multicore resources. An additional feature of the HCOC algorithm is its adaptability; it can be adjusted to prioritize budgets over strict deadlines, making it versatile for varying user needs [24].

The procedure begins by initializing all resources within the private cloud, followed by the execution of a schedule using the PCH (Private Cloud Heuristic) method. A counter, named "iteration", is initiated at zero, serving as a loop control mechanism for the subsequent steps.

Within the primary loop, several actions are executed based on specific conditions. If the makespan, which is a measure of the total time required to complete a set of tasks, exceeds a predetermined threshold and the iteration count remains below a predefined size, the iteration count is incremented. Following this, a specific node, termed "n_i", is selected based on given criteria. The number of node clusters is set, and another loop is entered to select resources from public clouds. Resources are chosen by minimizing a specific ratio involving price and core count. This process continues until there are no more clusters left to assign. Concurrently, for every node "n_i", it's scheduled in a resource such that its Earliest Finish Time (EFT) is minimized. The Earliest Start Times (ESTs) and EFTs are recalculated after each node scheduling. This cycle of actions repeats until the primary loop's conditions are no longer met, culminating in the conclusion of the algorithm.

Figure 1. HCOC algorithm



In hybrid cloud settings, HCOC uses multicore resources to develop its schedules. The intention is to use these resources to produce a schedule that balances cost considerations with the need to meet specific deadlines. The specialties of the HCOC algorithm include its ability to maximize the use of multicore resources, leading to more efficient scheduling and execution. Additionally, it can balance cost considerations with the need to meet specific deadlines, providing users with control over their expenditure while ensuring timely execution.

The primary challenge lies in its hybrid design, which can complicate its implementation. Integrating resources from both public and private clouds while maintaining efficient scheduling can be a complex task. Additionally, there might be a continuous need for optimization to maintain the algorithm's efficiency. This ongoing adjustment can introduce overhead, potentially impacting system performance.

Scalable Heterogeneous Earliest-Finish-Time algorithm

operates in two distinct phases: task prioritizing and resource selection. During the task prioritizing phase, the algorithm's primary function is to establish a hierarchy of tasks based on their urgency. This is achieved through the deployment of a task prioritizing algorithm which meticulously ranks tasks according to their determined priority. The outcome of this phase is a meticulously curated list of tasks, referred to as ListPriority.

Subsequent to the task prioritization, the resource selection phase is initiated. Within this phase, the Scalable-Heterogeneous-Earliest-Finish-Time (SHEFT) algorithm is introduced, specially designed to schedule workflows within a Cloud computing context. SHEFT can be recognized as an evolved version of the HEFT algorithm, specifically adapted to map a workflow application onto a predefined number of processors. The resource allocation procedure begins with an open slate, where any resource might be allocated to any task. The task which stands atop the ListPriority due to its high priority rank is the primary candidate for scheduling. For every task, denoted as Ti, the algorithm determines both the earliest time it can commence (EST) and the earliest time it can conclude (EFT) on each designated resource, Rk. The resource that promises the quickest task conclusion time, or minTFT, is momentarily assigned to a variable known as RS. Following this, if there exists a resource, RS, available for the task Ti before or by its Estimated Ready Time (ERT), then this resource is confirmed for the task, updating its available time to the previously calculated minTFT. If no such resource is available, the algorithm applies predetermined decision-making rules based on calculated EFTs to ensure the task is appropriately scheduled. Throughout this process, the algorithm also accounts for resource idleness. If any resource remains inactive beyond a stipulated threshold, it's released, ensuring that resources are scaled effectively based on demand.

Functioning in heterogeneous environments, the primary aim is the efficient and expedited completion of tasks. It exhibits adaptability, scaling according to shifting requirements.

It is evident that it focuses on quick task completion by always targeting the earliest possible finishing times. Another significant benefit is its scalability, allowing the system to adjust seamlessly according to changes in resource availability and demand. Primarily, its design caters predominantly to heterogeneous environments, which may restrict its applicability in other settings. Additionally, by always seeking the earliest finish times, there's a potential risk of not making full use of available resources, leading to possible underutilization.

Workflow Techniques

Workflow Orchestrator for Distributed Systems Architecture

The Workflow Orchestrator designed for Distributed Systems Architecture is a structured mechanism that unfolds in three sequential phases: workflow preprocessing, elastic resource provisioning, and task scheduling. Beginning with the workflow preprocessing phase, this stage entails the primary analysis and preparation of the workflow. It examines the integral components of the workflow to ensure they are ready for further processing and execution. Following preprocessing, the next stage is elastic resource provisioning. In this phase, the Orchestrator evaluates and adjusts the resources dynamically. Depending on the current demands and the specific requirements of the workflow, resources can be provisioned or de-provisioned to ensure efficient execution. The final stage is task scheduling. This is where the actual placement and execution of tasks take place, ensuring that each task is assigned to an appropriate resource, considering factors like load, availability, and specific task requirements.

One of the primary objectives of this Orchestrator is to address the inconsistencies that frequently arise in the Quality of Service (QoS) across varying distributed systems. By doing so, it places a special emphasis on refining and augmenting batch queues, which are crucial for task execution in distributed systems. The intention is to reduce wait times, ensure tasks are processed promptly, and optimize overall system performance.

It offers a noticeable enhancement in the Quality of Service (QoS). By targeting and rectifying inconsistencies in QoS across distributed systems, it provides a more uniform and improved service quality. Additionally, its approach to resource allocation stands out. By focusing on the nuances of batch queues and ensuring their optimal use, resources are employed efficiently, reducing wastage and improving system throughput.

Predominantly, its design and functioning are primarily aligned with distributed systems. This specialized focus may restrict its adaptability or functionality in other architectures or environments. Moreover, while it strives to address and optimize various QoS parameters, it might not encompass all the variables associated with QoS, potentially leaving some aspects unaddressed.

Multi-cost job routing and scheduling

The Multi-Cost Job Routing and Scheduling mechanism aims to refine the approach to job routing and task allocation in computing environments. One of its central features is the enhancement of the resource reservation process. By diving deeper into the nuances of resource management, this mechanism provides a detailed outlook on the best possible timings for data transmission and task execution. What makes this mechanism distinct is its ability to consider multiple cost factors. Instead of just focusing on one dimension, such as time or financial implications, it holistically assesses various cost elements, providing a comprehensive view that aids in making well-informed scheduling decisions.

This mechanism offers a notable strength is its precision in dictating task initiation times. By considering multifaceted costs and employing advanced scheduling algorithms, it can pinpoint the most suitable moments to kickstart specific tasks, ensuring optimal resource usage and timely execution. Another significant benefit is the redefined resource reservation process. By continuously monitoring and adjusting reservations, the mechanism actively works to prevent potential bottlenecks, ensuring smoother, uninterrupted operations. One primary challenge is the complexity associated with simultaneously handling various costs. The need to balance multiple cost factors can introduce a layer of complication to the scheduling process, potentially demanding more sophisticated algorithms and processes. Additionally, to achieve the precision it promises, the mechanism might necessitate detailed and accurate data. The absence of such data can impact its ability to deliver precise scheduling, meaning that there's a reliance on the availability and accuracy of input information.

Integer Linear Programming (ILP) technique

Integer Linear Programming (ILP) serves as a potent mathematical method that utilizes linear equations to obtain optimal solutions, particularly for complex challenges such as workflow scheduling. ILP is fundamentally structured to ensure the finest results by incorporating various factors and limitations.

One significant application of ILP highlighted in the document pertains to workflow scheduling in SaaS or PaaS cloud settings. The issue is dual-pronged: on one hand, customers desire their tasks to be completed within their expected response time or set deadline; on the other, the SaaS or PaaS cloud service providers aim to enhance their profit margins. Addressing this divergence, the document proposes an ILP-centered approach, specifically configured to address the workflow scheduling dilemma in these cloud scenarios. Two levels of Service Level Agreements (SLA) are considered. The initial level describes the accord between the cloud service provider and its clientele, while the subsequent level characterizes the pledge between the SaaS or PaaS cloud and its IaaS providers. Augmenting the ILP, the document introduces two heuristic methods, BMT and BMEMT, constructed to identify workable integer solutions in situations where the ILP may be modified.

Exploring the ILP structure further, an array of elements, constants, and restrictions are discussed. Within these elements, certain notations indicate whether a node concludes at a specific moment on a particular Virtual Machine (VM), while other notations demonstrate if a VM is in use at a certain moment. The primary aim of this ILP is to curtail the cumulative expenses related to VM utilization. Designated constraints form the essential guidelines of the ILP, ensuring that tasks are allotted correctly, with considerations such as VM accessibility, task interdependencies, and individual VM processing capacities. Additionally, these restrictions confirm that the count of VMs adheres to the parameters set by IaaS providers or SLAs, and that the utilized elements maintain binary characteristics.

The document also presents a systematic method to decipher the ILP. This iterative method begins by triggering a modified ILP solver. Successively, it evaluates unscheduled tasks, and upon identification, selects a node and the corresponding element based on the heuristic. A fresh constraint is added to the ILP, after which the modified solver re-engages with this revised ILP.

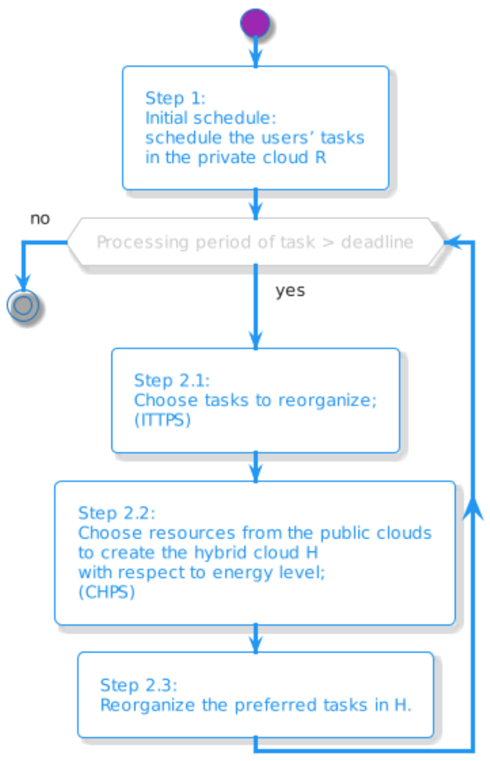
The repetition persists until tasks are accurately allotted, after which the definitive answer is provided.

Resource Utilization and Efficiency

Compatibility of Hybrid Processor Scheduler

For users, a common dilemma is determining the best resources to request from a public cloud, particularly when considering current demands and the overheads linked to these resources. The primary purpose of CHPS is to discern which resources from the public cloud should be incorporated with the private cloud, ensuring an adequate processing capacity to execute workflows within predetermined time frames.

Figure 2. CHPS algorithm


Step 1:
Initial schedule:
schedule the users' tasks
in the private cloud R
no
Processing period of task > deadline
yes
Step 2.1:
Choose tasks to reorganize;
(ITTPS)
Step 2.2:
Choose resources from the public clouds
to create the hybrid cloud H
with respect to energy level;
(CHPS)
Step 2.3:
Reorganize the preferred tasks in H.

The CHPS system's methodology is divided into three distinct phases. Initially, the process revolves around the identification and assignment of processors. This phase is concerned with recognizing the available processors and then delegating them based on specific task schedules and time parameters. Such an approach ensures that there is a seamless alignment between task demands and the processors' capabilities. Subsequently, the system transitions to the second phase, where the spotlight is on optimal resource allocation. At this juncture, the objective is to allocate resources to processors in the most efficient manner, ensuring that the needs of tasks are met without unnecessary resource wastage.

The third phase of the CHPS system is a more intricate, going into the broader compatibility aspects of the Hybrid Processor Scheduler. Here, the focus is on assessing the scheduler's adaptability across a spectrum of processors, ranging from the older, traditional models to the contemporary, high-performance ones. Additionally, this phase evaluates the scheduler's compatibility with various operating systems and diverse application prerequisites, ensuring a versatile performance across multiple platforms. By integrating both low-range and mid-range processors with their high-end counterparts, the system promotes a Green IT framework. This initiative not only ensures optimal performance but also minimizes e-waste, making it a sustainable solution. The integration process is facilitated by the cloud interface, which plays a pivotal role in task scheduling and process allocation. The subsequent phases, while technical, are essential in ensuring that resources are not only allocated efficiently but also utilized based on their capabilities, ensuring that the user tasks, time constraints, and processor abilities are in harmony.

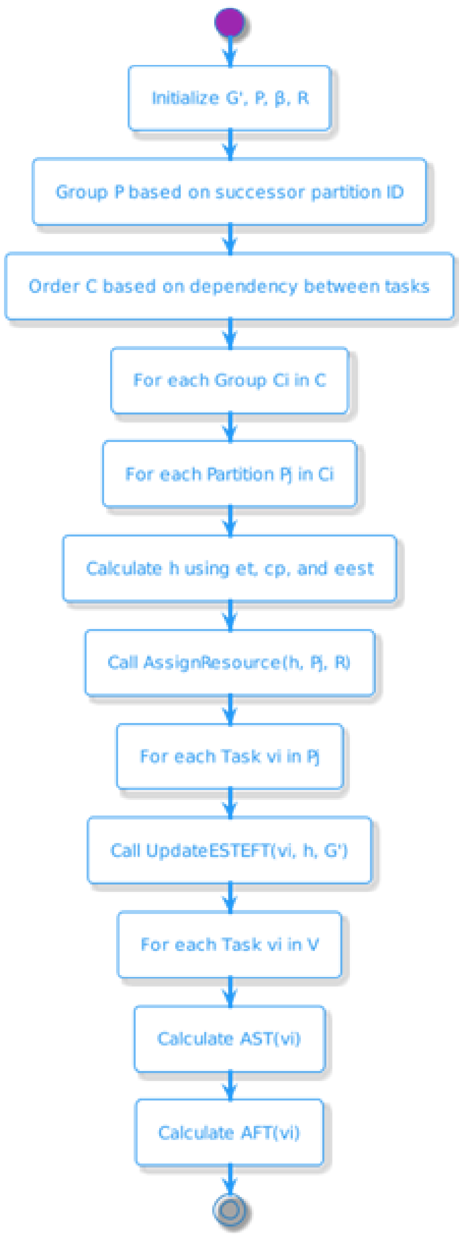
Partitioning-Based Workflow Scheduling

The Partitioning-Based Workflow Scheduling algorithm at its core shifts the focus from users having to define strict boundaries for deadlines and costs. Instead, it introduces a slack parameter. This parameter guides the balance between adhering to deadlines and managing costs. The central tenet of PBWS revolves around partitioning a workflow into multiple task graphs or partitions, with the granularity of these partitions being directly influenced by the slack parameter.

One of the features of PBWS is its adaptability. The algorithm assigns each partition to the cloud resource that delivers the optimal balance between execution time (makespan) and cost. However, PBWS does not adopt a rigid partitioning structure. The partition sizes can be dynamically adjusted by redistributing tasks across different partitions. This flexibility ensures that resource assignments can be continuously optimized based on changing requirements and conditions.

The PBWS approach works in three distinct steps. The first is the partitioning phase. Here, the aim is to divide the workflow such that the resultant partitions maintain their dependencies in a manner that they can still be represented as a DAG. This structured partitioning simplifies subsequent resource assignments. In the second phase, or the partition adjustment step, tasks are strategically rearranged across partitions. The intention behind this step is to reduce the overall makespan further. The resource assignment phase comes into play. This phase is responsible for pairing tasks with the most suitable resources, ensuring that resource idle times are minimized, leading to cost savings.

Figure 3. resource scheduling in PBWS


Initialize G', P, β, R
Group P based on successor partition ID
Order C based on dependency between tasks
For each Group Ci in C
For each Partition Pj in Ci
Calculate h using et, cp, and eest
Call AssignResource(h, Pj, R)
For each Task vi in Pj
Call UpdateESTEFT(vi, h, G')
For each Task vi in V
Calculate AST(vi)
Calculate AFT(vi)

When viewed considering its application, PBWS, by applying a partitioned Directed Acyclic Graph (DAG), seeks to minimize communication overheads and boost execution times. The advantages are clear: communication is streamlined, significantly cutting down on overheads, and the execution of the workflow is time-efficient. However, the approach includes the high computational demands required during the partitioning process. Moreover, there's always an inherent risk of ending up with suboptimal partitions, especially if the graph specifications are particularly complex. This could inadvertently affect the anticipated efficiencies that the algorithm promises.

The resource assignment under this method comprises two primary stages: the identification of the resource set and the allocation of resources. The initial phase, resource set identification, aims to discern which types of resource sets should be designated for partitions in a way that minimizes the wait times caused by data dependencies between partitions. For this, partitions are categorized into distinct groups, with each group led by a specific partition and associated with predecessor partitions linked to the leader. The partitions' resource set types are identified based on various parameters, such as the execution start time and computational time required for tasks. An aspect of this phase is the establishment of resource set types that reduce overall computational time. After establishing the resource set types, the resource allocation phase ensues, wherein tasks are assigned to specific resources.

Resource allocation starts by first determining the start (AST) and finish (AFT) times for each task based on the previously identified resource set types. The guiding principle here is the slack parameter, β, which plays a pivotal role in balancing resource usage and execution time. A value range between 0 and 1 for β helps control the allocation of resources to tasks. For instance, a β value of 1 prioritizes cost-saving by using the fewest resources, while a value of 0 emphasizes reducing execution time even if it necessitates more resources. This phase integrates Upper and Lower Bound calculations for each task's AST, considering factors like available resources and budget constraints. In essence, the algorithm's flexibility in resource allocation is driven by the balance between cost and execution time, ensuring that tasks are executed efficiently while adhering to resource and budgetary constraints.

Conclusion

Effective resource and cost management holds significant importance in cloud computing. As more organizations transition their operations to the cloud, achieving optimal utilization of resources becomes crucial. Proper resource management ensures the peak performance and responsiveness of applications hosted within the cloud. In many cloud service models, particularly the pay-as-you-go structures, the financial implications are directly tied to resource usage. Hence, any wastage or over-provisioning of resources can lead to unnecessary financial outlays. Thus, adept management of resources not only upholds application performance but also enforces financial discipline. Scientific Workflow Systems (SWFS) in cloud and grid computing have risen to prominence due to their capacity to manage and coordinate a multitude of computational tasks. Within this context, one predominant challenge is the optimization of the cost associated with the execution of these workflows. Cost optimization in SWFS for cloud computing is not a straightforward concern. Rather, it includes cost-aware dilemma that mandates a nuanced approach. First and foremost, cloud and grid computing environments are typically populated by various users. These users invariably compete for the limited resources available, aiming to achieve certain Quality of Service (QoS) constraints. Such competition can result in contention for resources, driving up costs if not appropriately managed.

Three principal categories Resource and Cost Optimization Techniques were discussed in this study: Cloud and VM-focused strategies, Workflow techniques, and Resource Utilization and Efficiency techniques. Cloud and Virtual Machine (VM)-focused strategies mainly focus on optimizing resources within a cloud setup, particularly virtual machines, ensuring efficient usage and cost minimization. Various techniques fall under this category, each catering to specific cloud setups and scenarios. The SaaS Cloud Partial Critical Paths technique, specifically designed for Software as a Service (SaaS) cloud models, pinpoints critical paths in workflows and aims to optimize costs while adhering to specified timeframes. Another model, the Hybrid Cloud Optimized Cost model (HCOC), is particularly effective in hybrid cloud setups, leveraging multicore resources. It concentrates on workflow scheduling, balancing between cost-effectiveness and deadline adherence. The Scalable Heterogeneous Earliest-Finish-Time algorithm, as the name suggests, focuses on heterogeneous environments. It ranks tasks based on their completion times, ensuring that they finish at the earliest possible opportunity, while also offering scalability to accommodate varying demands. Additionally, the Workflow-aware Preprocessing Provisioning Dynamic Scheduling is tailored for scientific workflows. It integrates preprocessing steps with dynamic provisioning to ensure efficient scheduling. Dynamic Provisioning Dynamic Scheduling approach prioritizes flexibility, adjusting both resource allocation and task scheduling according to prevailing demands.

Workflow techniques primarily deal with optimizing the execution of tasks in cloud environments. The Workflow Orchestrator for Distributed Systems Architecture is engineered to provide uniformity in the Quality of Service (QoS) offered by various distributed systems. By doing so, it focuses on refining batch queues, ultimately leading to heightened performance and efficiency. The Multi-cost job routing and scheduling algorithm takes a comprehensive view of task management. It provides in-depth insights into resource reservation, specifying the ideal times for data transmissions and task completions while constantly keeping an eye on the associated costs. Another technique, the Integer Linear Programming (ILP), is a mathematical method known for its precision. Utilizing linear equations, it seeks the best possible solutions for scheduling challenges, although it might be computationally intensive, especially for larger, more complex problems.

Techniques under the resource utilization and efficiency category prioritize making the most out of available resources, ensuring they are used in the most cost-effective and energy-efficient manner. The Compatibility of Hybrid Processor Scheduler exemplifies this approach. It selects resources based on their current energy levels and the specific needs of the applications they support, ensuring efficient energy consumption. In scenarios where budgeting is crucial, the Budget-based constraint workflow scheduling become effective. It schedules tasks, ensuring that they align with the user-defined budget, thus emphasizing cost-effectiveness.

References

[1] F. Liu, J. Tong, J. Mao, R. Bohn, and J. Messina, “NIST cloud computing reference architecture,” *NIST Spec. Pub.*, 2011.

[2] W. Voorsluys, J. Broberg, and R. Buyya, “Introduction to cloud computing,” *Cloud computing: Principles*, 2011.

[3] T. C. Chieu, A. Mohindra, and A. A. Karve, “Dynamic scaling of web applications in a virtualized cloud computing environment,” *Conference on e …*, 2009.

[4] M. Fiore and M. Devesas Campos, “The Algebra of Directed Acyclic Graphs,” in *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky: Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, B. Coecke, L. Ong, and P. Panangaden, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 37–51.

[5] I. Shrier and R. W. Platt, “Reducing bias through directed acyclic graphs,” *BMC Med. Res. Methodol.*, vol. 8, p. 70, Oct. 2008.

[6] T. J. VanderWeele and J. M. Robins, “Directed acyclic graphs, sufficient causes, and the properties of conditioning on a common effect,” *Am. J. Epidemiol.*, vol. 166, no. 9, pp. 1096–1104, Nov. 2007.
[7] M. Masdari and M. Zangakani, “Efficient task and workflow scheduling in inter-cloud environments: challenges and opportunities,” *J. Supercomput.*, vol. 76, no. 1, pp. 499–535, Jan. 2020.
[8] C. Wan, C. Wang, and J. Pei, “A QoS-awared scientific workflow scheduling schema in cloud computing,” *Conference on Information Science …*, 2012.
[9] J. Liu, L. Pineda, E. Pacitti, and A. Costan, “Efficient scheduling of scientific workflows using hot metadata in a multisite cloud,” *on Knowledge and …*, 2018.
[10] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, “Scientific Workflow Scheduling with Provenance Support in Multisite Cloud,” in *High Performance Computing for Computational Science – VECPAR 2016*, 2017, pp. 206–219.
[11] M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, “Multi-Objective Task and Workflow Scheduling Approaches in Cloud Computing: a Comprehensive Review,” *Int. J. Grid Util. Comput.*, vol. 18, no. 3, pp. 327–356, Sep. 2020.
[12] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, “Scientific Workflow Scheduling with Provenance Data in a Multisite Cloud,” in *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXIII*, A. Hameurlain, J. Küng, R. Wagner, R. Akbarinia, and E. Pacitti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 80–112.
[13] S. Abrishami and M. Naghibzadeh, “Deadline-constrained workflow scheduling in software as a service Cloud,” *Scientia Iranica*, vol. 19, no. 3, pp. 680–689, Jun. 2012.
[14] L. F. Bittencourt and E. R. M. Madeira, “HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds,” *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, Dec. 2011.
[15] “Scheduling scientific workflows elastically for cloud computing.” [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6008783/.
[16] “A budget and deadline aware scientific workflow resource provisioning and scheduling mechanism for cloud.” [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6846925/.
[17] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, “Cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. InSC’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.” IEEE, 2012.
[18] L. Ramakrishnan, J. S. Chase, D. Gannon, D. Nurmi, and R. Wolski, “Deadline-sensitive workflow orchestration without explicit resource control,” *J. Parallel Distrib. Comput.*, vol. 71, no. 3, pp. 343–353, Mar. 2011.
[19] T. Stevens *et al.*, “Multi-cost job routing and scheduling in Grid networks,” *Future Gener. Comput. Syst.*, vol. 25, no. 8, pp. 912–925, Sep. 2009.
[20] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira, “Workflow Scheduling for SaaS/PaaS Cloud Providers Considering Two SLA Levels,” *scholar.archive.org*, 2012.
[21] K. L.Giridas and A. Shajin Nargunam, “Compatibility of hybrid process scheduler in green IT cloud computing environment,” *Int. J. Comput. Appl.*, vol. 55, no. 5, pp. 27–33, Oct. 2012.

[22] K. Almi'ani and Y. C. Lee, "Partitioning-based workflow scheduling in clouds," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Crans-Montana, Switzerland, 2016, pp. 645–652.
[23] M. Tanaka and O. Tatebe, "Workflow Scheduling to Minimize Data Movement Using Multi-constraint Graph Partitioning," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, infona.pl, 2012.
[24] J. Yu and R. Buyya, "A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms," *scholar.archive.org*, 2006.