

International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com ISSN: 2250-3552

ITMCS: Migration-Aware Node Consolidation for Workflow Scheduling in Virtualized Clouds

Devendra Rewadikar

Research Scholar, Department of Computer Science & Applications, Rabindranath Tagore
University, Bhopal, India

Dr. Pratima Gautam

Professor, Department of Computer Science & Applications, Rabindranath Tagore University,
Bhopal, India

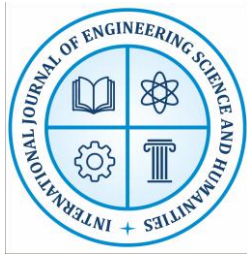
Abstract

Scheduling workflow jobs on a virtualized cloud involves more than handing each job to whichever resource happens to be free. Idle capacity may be spread over several resources in pieces too small to start a waiting job, even though the total number of idle nodes would be enough. This paper presents ITMCS (Improved Task Migration Consolidation Scheduling), an epoch-based and node-aware scheduler. At fixed intervals it retires finished work, collects the nodes that were released, merges fragmented free capacity through controlled migration, and then admits jobs from the queue. The scheduler assumes a space-shared node model, and its migration rule discards any move whose estimated delay could exceed the remaining execution time of the workload being moved. ITMCS is evaluated in CloudSim 3.0.2 with synthetic job batches and with the Montage and CyberShake workflow structures, using FCFS and Minimum Completion Time (MCT) as baselines. The measures are makespan, scheduling execution time, resource utilization, makespan standard deviation, and makespan skewness. In the simulations, ITMCS shortens the schedule, lowers decision overhead, raises utilization, and reduces run-to-run variability. The paper also discusses the limits of a simulation-only study and the need for validation on a physical cloud testbed.

Keywords: Cloud computing, Workflow scheduling, Task migration, Node consolidation, Makespan, Resource utilization, Virtualization, CloudSim

I. INTRODUCTION

Cloud computing delivers computing, storage, and platform resources as on-demand services [1], [2]. Much of this flexibility comes from virtualization, which lets several isolated execution environments share one physical infrastructure [3]. Provisioned capacity, however, is rarely busy all the time. Idle nodes, jobs of unequal length, and varying resource demands tend to leave free capacity scattered across the infrastructure.



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com **ISSN: 2250-3552**

Scheduling therefore shapes completion time, throughput, utilization, responsiveness, energy use, scalability, and cost to a large extent [4]. Workflows sharpen the difficulty because their component jobs can differ widely in duration and resource demand, and a handful of long jobs may fix the final makespan even when all other jobs are spread evenly.

Classic rules such as First-Come-First-Served (FCFS), Min-Min, Max-Min, and Minimum Completion Time (MCT) are popular because each decision is cheap to compute. A placement fixed at assignment time can nevertheless become poor once other jobs complete and free their resources. A scheduler may then see enough idle nodes in total while no single resource holds a large enough block for the next waiting job.

This paper targets that fragmentation problem by letting the scheduler reconsider the placement of running jobs at fixed decision epochs. Instead of adding hardware, ITMCS tries to turn scattered idle nodes into one usable block by migrating selected jobs. Its design rests on three elements: periodic state review, node-aware admission, and migration-based consolidation. The main contributions are: (1) a node-partitioned workflow scheduling model whose objectives are makespan, utilization, and schedule stability; (2) an epoch-driven ITMCS procedure that couples admission with controlled consolidation; and (3) a simulation study based on scientific workflows and several performance indicators.

The rest of the paper covers scheduling background, related work, the problem formulation, the ITMCS design, the experimental setup, results, limitations, and future directions.

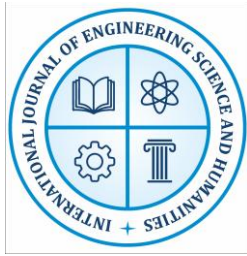
II. BACKGROUND

A. Scheduling in Virtualized Clouds

Resource management in an infrastructure cloud operates at two connected levels. At the host level, virtual machines (VMs) are placed on physical servers. At the task level, jobs are mapped to VMs or resource partitions that have already been provisioned. This study concerns the task level. Jobs may be independent or linked by precedence constraints; a workflow with dependencies is commonly modeled as a directed acyclic graph in which finishing one task releases the next. The completion time of a workflow thus depends on both computation and the communication between dependent tasks.

B. Scheduling Strategy Taxonomy

Scheduling policies can be classified by how and when they decide. A centralized scheduler holds a global view but places all decisions in one controller, whereas decentralized schemes spread that role. Static policies fix placements from information known before execution, while dynamic policies react to conditions observed at run time. Preemptive policies may suspend and later resume work; non-preemptive ones keep a resource until the assigned job ends. Online scheduling responds to each arrival, while batch or periodic scheduling decides over



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com ISSN: 2250-3552

accumulated work. Heuristics apply task- or resource-specific rules, and meta-heuristics search a wider solution space without promising an optimum.

Among the established heuristics, FCFS serves jobs in arrival order, and Shortest-Job-First favors short jobs at the cost of longer waits for long ones. Max-Min gives priority to the largest remaining job and picks the resource on which it finishes earliest [5], [6]. MCT assigns each job to the resource with the smallest estimated completion time. RASA alternates between Min-Min and Max-Min behavior to benefit from the strengths of both [7].

C. Space-Shared and Time-Shared Allocation

Under space sharing, a processing element or node belongs exclusively to one VM or job for the length of its allocation. Under time sharing, several entities advance together by splitting processor time. Either policy can be applied at the VM level and at the task level. ITMCS adopts a space-shared node model and revisits allocations only at defined epochs, so nodes stay exclusively owned during execution while controlled reconfiguration remains possible at epoch boundaries.

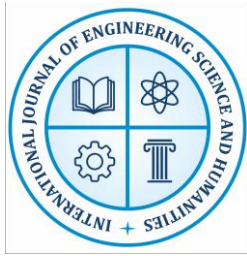
D. Virtualization and Migration

Virtualization presents physical resources as logical execution environments. Type-1 hypervisors run directly on the hardware, whereas Type-2 hypervisors depend on a host operating system. Full virtualization, paravirtualization, and hardware-assisted virtualization trade off compatibility, isolation, and overhead differently. These mechanisms enable server consolidation and workload relocation. Live migration matters most for ITMCS because it allows a running workload to move between hosts with only a brief interruption [8]. Migration still costs time and network bandwidth, so a consolidation policy must free enough capacity to justify each move.

III. RELATED WORK

Cloud load balancing aims to spread demand so that overloaded resources are relieved and idle ones are put to work. Typical techniques include round-robin, random selection, throttling, Min-Min/Max-Min variants, and nature-inspired methods, and some schemes build balancing directly into VM placement or task scheduling. A common weakness is that decisions taken on static or partly observed state cannot exploit capacity that is released later during execution.

Zhao and Huang studied distributed balancing based on live migration and accounted for processor, memory, I/O, and network utilization when redistributing load [9]. Tordsson et al. addressed VM placement across several cloud providers, optimizing cost, performance, and balance together [10]. Rouzaud-Cornabas proposed a peer-to-peer dynamic balancer in which cooperating balancers combine static and dynamic host information [11]. Tian et al. brought several resource dimensions into a dynamic scheduling and load-balancing algorithm [12].



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com ISSN: 2250-3552

Bhadani and Chaudhary evaluated a central balancing policy for VMs in Xen-based environments [13].

IV. PROBLEM FORMULATION

Let $J = \{j_1, \dots, j_m\}$ be the set of workflow jobs and $R = \{r_1, \dots, r_k\}$ the set of resource partitions. Resource r_q offers c_q allocatable nodes. Each job has a node requirement n_j and an execution requirement j . A schedule is feasible if it assigns every job to a resource and a start time such that the nodes occupied at any instant never exceed the capacity of that resource.

Let $CT(j)$ denote the completion time of job j . The primary objective is to minimize the makespan:

$$M = \max \{ CT(j) : j \in J \} \quad (1)$$

A secondary objective is to raise the share of provisioned node-time that does useful work. A third objective is schedule stability, expressed as the spread of the makespan over repeated runs. Scheduling under capacity constraints is computationally hard, which motivates a deterministic heuristic whose work per epoch is bounded.

The condition ITMCS is built around is fragmentation. Suppose that several resources together hold enough idle nodes for a waiting job, but the nodes are divided into smaller blocks. The job cannot start, although total capacity would suffice. If active jobs can be moved at a tolerable migration cost, the separate blocks can be merged into one usable region. This is the central operating principle of ITMCS.

V. ITMCS SCHEDULER

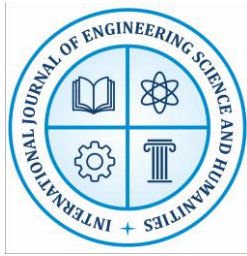
A. Epoch-Based Operating Model

ITMCS splits execution into fixed epochs of length Δ . Jobs run undisturbed within an epoch. At each epoch boundary the scheduler first returns the nodes of completed jobs, then checks whether the active allocation is fragmented, and finally tries to admit waiting jobs. This order is deliberate, so that freshly released capacity is considered before any migration is started.

A job is directly admissible when at least one resource has a contiguous free block of size n_j . If the job at the head of the queue cannot be admitted that way although the total free capacity suffices, ITMCS looks for migration options on the most fragmented resource. A migration set is accepted only if the resulting free block can hold the waiting job and the estimated migration delay stays below the remaining execution requirement of the jobs being moved. This restriction keeps consolidation from costing more time than it saves.

B. Scheduling Procedure

The procedure has six steps. (1) Initialize the job list, resource capacities, epoch length, and a queue in submission order. (2) Retire completed jobs and return their nodes to the free pool. (3) Scan the queue for direct admissions. (4) If the queue head is blocked by fragmentation, select



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com ISSN: 2250-3552

candidate active jobs for migration and re-pack the allocation. (5) Scan the queue again, so that the blocked job and any smaller jobs fitting the remaining capacity can be admitted. (6) Reduce the remaining execution requirement of every active job by one epoch and advance the clock. The procedure ends when both the waiting queue and the active set are empty.

C. Admission and Migration Condition

Let $f_q(t)$ be the number of free nodes on resource r_q at epoch t , and let $F(t) = \sum_q f_q(t)$ be the total free capacity. Direct admission of job j requires $\max_q f_q(t) \geq n_j$. Consolidation is possible when $F(t) \geq n_j$ and there is a migration set S satisfying

$$\delta(S) < \min \{ \rho_s(t) : s \in S \} \quad (2)$$

where $\delta(S)$ is the estimated cumulative migration delay and $\rho_s(t)$ is the remaining execution requirement of the migrated job

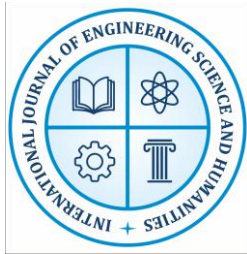
s. The condition is conservative on purpose: migration must not take longer than the work left in the jobs it moves.

D. Complexity and Determinism

In each epoch, reclaiming completed jobs and updating active ones takes time linear in the size of the active set. The queue scan compares jobs with the available resources, which gives an $O(m \cdot k)$ bound for the basic pass. Consolidation considers only candidate jobs on the most fragmented resource instead of searching every possible migration combination over the whole infrastructure. The per-epoch cost therefore stays polynomial and does not depend on the workflow topology. Since admissions, retirements, and migrations happen only at epoch boundaries and the queue order is fixed, identical input and epoch length yield an identical scheduling sequence.

E. System Architecture

Fig. 1 shows the conceptual architecture. Workflow requests arrive at a broker that keeps the waiting queue and receives resource-state information. ITMCS acts as the scheduling controller between the broker and the VM manager, while a resource monitor reports node availability. The controller sends start or migration commands to the VM manager, which carries out placement or relocation on the virtualized infrastructure. After execution states are updated, the control loop moves on to the next epoch.



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com ISSN: 2250-3552

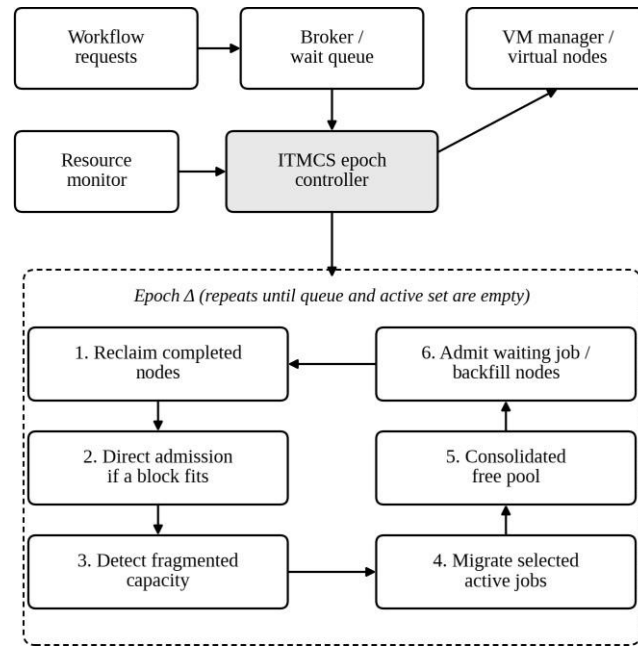


Fig. 1. Conceptual control loop of ITMCS: reclamation, fragmentation detection, migration-based consolidation, and re-admission

VI. EXPERIMENTAL DESIGN

A. Simulation Environment

The experiments run on CloudSim 3.0.2 [14]. The simulation driver creates the scheduler together with the job, machine, and failure components. Hosts are modeled as x86 systems running Linux with Xen as the virtualization layer, storage parameters follow the toolkit defaults, and resource characteristics are taken from Amazon EC2 instance descriptions. This setup allows repeated, controlled runs with consistent workload and resource definitions.

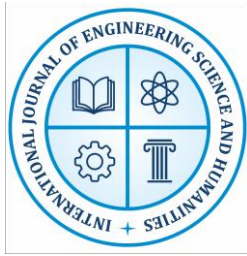
B. Workloads and Baselines

Two kinds of workload are used. Synthetic batches show how the scheduler behaves as the number of jobs grows. In addition, two scientific workflows from the Pegasus collection are included: Montage, which stitches astronomical images into mosaics, and CyberShake, which characterizes seismic hazard [15]. Their different structures and task-duration patterns create different forms of resource fragmentation. FCFS and MCT serve as baselines.

C. Performance Measures

Five indicators are used. Makespan M is the total schedule length. The optimization measure for makespan is

$$OMS = ((MMS - M_{min}) / MMS) \times 100\% \quad (3)$$



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com ISSN: 2250-3552

where MMS is the mean makespan and M_{min} the smallest makespan observed. Mean scheduling execution time (MSET) is the wall-clock time needed to compute scheduling decisions, excluding task execution time.

Resource utilization rate (RUR) is the share of available resource time that is used productively over the observation intervals. Makespan standard deviation (MSD) measures the spread of schedule length over repeated runs, with smaller values meaning more predictable behavior. Makespan skewness (SM) is the adjusted Fisher–Pearson coefficient and describes the asymmetry of the distribution; a positive value indicates a longer right tail and hence occasional unusually long schedules.

VII. RESULTS AND DISCUSSION

A. Schedule Length

At every batch size tested, ITMCS produced shorter makespans than FCFS and MCT in the simulations. The two baselines stay fairly close to each other, and their order changes with workload size and job mix. The advantage of ITMCS grows with the queue length, because a longer waiting queue offers more chances to reorganize fragmented capacity.

B. Scheduling Overhead

ITMCS also shows a lower mean scheduling execution time in the tested configurations. Consolidation adds a search step, but the step is conditional and not continuous: it runs only when the queue head is blocked although aggregate capacity is sufficient. The decision time of the baselines rises more steeply with workload size, whereas ITMCS confines its consolidation search to a limited part of the active allocation.

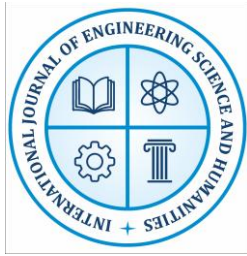
C. Resource Utilization

Resource utilization is higher for ITMCS at every scale tested. Utilization still falls as the workload grows, because the tail of a schedule can be dominated by a few long jobs while other nodes sit idle. ITMCS shortens part of this idle period by using the capacity freed through consolidation and by letting smaller queued jobs fill residual blocks. At the largest batch size the gap between the schedulers narrows, which is consistent with less room to reorganize a resource pool already filled with long-running jobs.

D. Makespan Robustness

For both Montage and CyberShake, MSD and skewness are lower under ITMCS than under either baseline. The smaller dispersion suggests that schedule length depends less on placement variation, and the lower right-tail skewness points to fewer unusually long executions. The effect is stronger for CyberShake, which fits the observation that its heterogeneous task durations create more fragmentation and therefore more consolidation opportunities.

E. Interpretation and Scope



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com ISSN: 2250-3552

The results support the hypothesis that aggregate free capacity is not the same as immediately usable capacity when jobs need blocks of nodes. By revisiting active placements at controlled intervals, ITMCS exposes capacity that a one-shot mapping would leave stranded. The evidence is nevertheless simulation-based and should be read as an evaluation under the stated CloudSim model, not as a measurement of production-cloud behavior.

VIII. THREATS TO VALIDITY AND LIMITATIONS

Several limitations apply. First, migration downtime, network contention, and hypervisor effects are represented by simulation parameters and were not measured on physical servers. Second, the usefulness of the migration rule depends on how accurate the delay estimate in the admission condition is. Third, only FCFS and MCT are compared; adding Min-Min, Max-Min, RASA, and meta-heuristic methods would broaden the evaluation. Finally, only two workflow families are examined, so the findings should not be generalized to all scientific or commercial workflows without further testing.

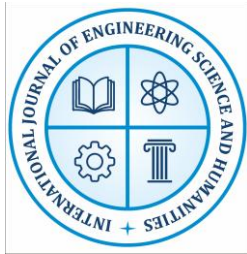
IX. CONCLUSION AND FUTURE WORK

This paper presented ITMCS, an epoch-based scheduler for virtualized cloud workflows that treats fragmented node capacity as a schedulable resource. It combines reclamation of completed work, direct node-aware admission, selective migration, consolidation, and queue re-evaluation. In simulations with synthetic workloads, Montage, and CyberShake, it reduced makespan and scheduling overhead and improved utilization and stability compared with FCFS and MCT.

Future work should extend the consolidation decision with meta-heuristics such as Ant Colony Optimization, Genetic Algorithms, and Particle Swarm Optimization. The model could also include resource reservations and tiered application scaling, in which scaling one application changes the capacity left for others. Most importantly, a physical testbed should be used to measure migration downtime and network cost and to see how they affect the admission condition. Such measurements would give a firmer basis for judging the practical value of migration-aware consolidation.

REFERENCES

- [1] P. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Spec. Publ. 800-145, Sep. 2011.
- [2] M. Armbrust et al., "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, Apr. 2010.
- [3] P. Barham et al., "Xen and the art of virtualization," in Proc. 19th ACM Symp. Oper. Syst. Princ. (SOSP), 2003, pp. 164–177.
- [4] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," J. Grid Comput., vol. 14, no. 2, pp. 217–264, 2016.



International Journal of Engineering, Science and Humanities

An international peer reviewed, refereed, open-access journal
Impact Factor 8.3 www.ijesh.com **ISSN: 2250-3552**

- [5] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," J. ACM, vol. 24, no. 2, pp. 280–289, 1977.
- [6] T. D. Braun et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," J. Parallel Distrib. Comput., vol. 61, no. 6, pp. 810–837, 2001.
- [7] S. Parsa and R. Entezari-Maleki, "RASA: A new task scheduling algorithm in grid environment," World Appl. Sci. J., vol. 7, special issue, pp. 152–160, 2009.
- [8] C. Clark et al., "Live migration of virtual machines," in Proc. 2nd USENIX Symp. Netw. Syst. Design Implement. (NSDI), 2005, pp. 273–286.
- [9] Y. Zhao and W. Huang, "Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud," in Proc. 5th Int. Joint Conf. INC, IMS IDC, 2009, pp. 170–175.
- [10] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," Future Gener. Comput. Syst., vol. 28, no. 2, pp. 358–367, 2012.
- [11] J. Rouzaud-Cornabas, "A distributed and collaborative dynamic load balancer for virtual machine," in Euro-Par 2010 Parallel Processing Workshops, LNCS 6586, 2011, pp. 641–648.
- [12] W. Tian, Y. Zhao, Y. Zhong, M. Xu, and C. Jing, "A dynamic and integrated load-balancing scheduling algorithm for cloud datacenters," in Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst. (CCIS), 2011, pp. 311–315.
- [13] J. Bhadani and S. Chaudhary, "Performance evaluation of web servers using central load balancing policy over virtual machines on cloud," in Proc. 3rd Annu. ACM Bangalore Conf. (COMPUTE), 2010.
- [14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Softw., Pract. Exper., vol. 41, no. 1, pp. 23–50, Jan. 2011.
- [15] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," Future Gener. Comput. Syst., vol. 29, no. 3, pp. 682–692, Mar. 2013.