A hierarchical decentralized architecture to enable adaptive scalable virtual machine migration

DOI:
10.1002/cpe.7487

Document Version
Final published version

Received: 11 April 2021 | Revised: 12 September 2022 | Accepted: 13 October 2022

DOI: 10.1002/cpe.7487

RESEARCH ARTICLE

WILEY

A hierarchical decentralized architecture to enable adaptive scalable virtual machine migration

Abdul R. Hummaida | **Norman W. Paton** | **Rizos Sakellariou**

Department of Computer Science, University of Manchester, Manchester, UK

Correspondence
Abdul R. Hummaida, Department of Computer Science, University of Manchester, Kilburn Building, Oxford Road, Manchester M13 9PL, UK.
Email: abdul.hummaida@postgrad.manchester.ac.uk

Abstract

Cloud computing is an established paradigm for end users to access resources. Cloud infrastructure providers seek to maximize accepted requests, meet Service Level Agreements (SLAs), and reduce operational costs by dynamically allocating Virtual Machines (VMs) to physical nodes. Many solutions have been presented to manage cloud infrastructure, however, these tend to be centralized and suffer in their ability to maintain Quality of Service (QOS) and support data centers with thousands of nodes. Decentralized approaches, with no central management, can manage large data centers. However, these tend to reduce the ability to obtain an optimal resource allocation across the data center. To address this, we propose a hybrid hierarchical decentralized architecture that achieves lower SLA violations and lowers network traffic. We used simulation to evaluate our proposal in practice with a variety of existing VM placement policies.

KEYWORDS

data center QOS, data center scalability, decentralized architecture, hierarchical architecture, resource management, VM migration

1 | INTRODUCTION

Cloud computing environments enable end users to access computing resources through a simplified service model, instead of purchasing, configuring and maintaining these resources. Infrastructure Providers (IPs) typically abstract data center resources and present them to customers through a virtualization layer, with a Virtual Machine (VM) as a common form, with each VM being isolated from other VMs. End users of cloud resources typically request these through web APIs, which map the user's requests to virtual resources that reside on physical resources in the data center.[1]

VM Placement involves efficient utilization of available resources for applications to meet performance goals such as SLA. To place VMs, a scheduling process needs information on the resource requirements of the VMs being allocated, such as CPU and number of cores, amount of memory and network bandwidth. Once a resource is provisioned, IPs track the varying demands on each resource, and this varies over time as end users consume and release these resources.

To meet workload demands, IPs may reconfigure the assignment of physical resources to VMs, by increasing or reducing resource allocation to a workload, through Elasticity.[2] Resource reconfiguration is complex and may cause service interruption, hardware wear and tear and system instability.[3] Live VM migration is a common reconfiguration case to achieve Service Level Agreements (SLAs) and reduce energy consumption, and has been applied at scale.[4] Proposals in the literature aim to minimize SLA violations, and some trade this off with another objective, such as reducing energy consumption or maximizing IP revenue, and typically focus on the CPU as the primary resource to manage.[5]

There are also technologies, such as containerisation, which support the fast deployment of cloud applications. A container housing an application is allocated to a VM, and a VM is allocated to a node.[6-9] While container resource management overlaps with our work, we focus on the allocation of VMs to nodes.

Centralized, hierarchical and decentralized architectures have their merits. Centralized architectures have a global view, hierarchical architectures have increased scalability compared to centralized, and decentralized architectures have no central controller and can scale to manage a large number of nodes. We hypothesize that the strengths of these architectures can be combined, and their weaknesses reduced. We propose a Scalable Hierarchical Decentralized Framework (SHDF), which overcomes the weaknesses of hierarchical, decentralized and centralized approaches, by achieving improved QoS metrics. SHDF consists of hierarchical controllers operating at different scopes. On the lowest level, every node in the infrastructure contains a *Node Controller* (NC), which dynamically adjusts resource configurations to satisfy VM demands on each node. Each NC operates in a decentralized architecture and cooperates with other NCs and a *Lead Node* (LN), which is a higher level controller for all the NCs within a given cluster.

In this paper, we extend a preliminary version of this paper,[10] by implementing and evaluating QOS metrics on multiple policies from the literature. Our goal is to demonstrate improved QOS metrics compared to centralized, hierarchical, and decentralized architectures.

The scalability of a system is defined as the ability to meet a larger workload requirement by adding a proportional amount of resources, and maintaining its performance.[11] By achieving improved QOS performance metrics, our proposal should scale better compared to the examined architectures; the extent to which this is the case is explored through empirical evaluation.

We used several of the metrics in the literature[12,13] to evaluate our approach. The objective is chiefly to achieve:

1. Fewer SLA violations compared to centralized, hierarchical and decentralized architectures.
2. Lower network traffic utilization to manage resources, compared to centralized architectures.
3. Improved scalability, as the number of nodes in the data center increases, compared to centralized, hierarchical and decentralized architectures.

The rest of this paper is organized as follows. Section 2 outlines the different resource management architectures and Section 3 describes related work. Section 4 describes the architecture of our proposed scalable framework. Section 5 presents multiple management algorithms and an evaluation of our implementation, comparing it to centralized, hierarchical and decentralized architectures. In Section 6 we conclude and discuss future work.

2 | BACKGROUND

In centralized resource management approaches, a global controller has a view of all the resources being managed, including VMs and consumption of resources such as memory, CPU, storage, and network on each physical node. The whole data center infrastructure may be used as input into a single adaptation cycle. Resource metric collection occurs periodically,[14] from every node in the entire infrastructure, with each managed node sending its metrics to the global controller through the network. Centralized approaches have a single global controller and are the most prevalent in literature proposals.[15] As centralized approaches consider the entire infrastructure, they are subject to scalability, reactivity, and fault-tolerance issues.[16] To address the scalability challenges in centralized approaches, researchers investigated other architectures, including hierarchical approaches.

Hierarchical architectures typically divide the infrastructure into multiple sections, with a decision engine in each section, which has the effect of creating a distributed centralized architecture for each section. In such a section, typically on the cluster level, each node is connected to a cluster manager from which it receives adaptation commands. Within each divided section, metric collection operates in a similar way to centralized architectures. Resource metric collection typically occurs periodically in a similar way to centralized architectures. However, the metrics being sent to a global controller are sent to a controller managing a smaller subset of nodes at the rack or cluster level.[17] As the number of nodes within the cluster increases, the decision-making algorithm faces a similar challenge to traditional centralized approaches. The execution time can result in an inability to react in a timely manner to SLA violations. Hierarchical approaches improve the reliability properties compared to centralized approaches, as they divide the managed infrastructure and increase the number of managing controllers. However, hierarchical approaches suffer similar challenges to centralized, at the single cluster level, and require explicit redundancy strategies.

Decentralized architectures distribute the management of the infrastructure without a centralized controller,[18] and both hierarchical and decentralized architectures are a form of distributed management architectures. Some of the decentralized approaches use an overlay network as a mechanism for nodes to communicate. An overlay is a logical network that runs independently of a physical network and does not change the underlying network. While decentralized architectures can achieve large scale, they have higher complexity in their design due to the close cooperation between nodes to achieve resource management objectives.[19] Each node cooperates with other nodes in its overlay with no ability to migrate a VM outside the overlay, and in an SLA violation case, this could result in a node remaining in a stressed state for an extended period until a suitable migration is found inside the overlay.

3 | RELATED WORK

As the aim of our work is to investigate the architecture of the decision-making process, we categorize related work based on the management architecture used. In particular, we focus on hierarchical and decentralized architectures as these form the constituent parts of our proposal.

3.1 | Hierarchical controllers

Hierarchical approaches, a type of distributed decision-making, have been widely studied in cloud resource management. The approaches distribute the decision-making process, and typically use a multilevel hierarchical approach running at different time intervals. In Reference 20 the lowest level controller runs every hour and performs VM placement, power management and workload profiling and was evaluated on up to 7200 nodes. In Reference 17, the lowest level controllers manage a small number of machines and the applications that are hosted on them. At the next higher level, a controller manages machines owned by multiple lower level controllers. In Reference 21 three levels are used, where the highest level controller managed multiple clusters operating at seconds (Level 1), minutes (Level 2) and days (Level 3) intervals, however, the scalability of the approach was not explored. In Reference 22 the hierarchy is sliced along the operations of the controllers. A Level 1 controller handles VM placement and load balancing, and runs every 30 min. A Level 2 controller handles the resources of a node, and runs every few minutes. In References 23,24 VM placement is performed using two-level hierarchical controllers, and in Reference 24 a local controller guarantees the stability and performance of an application. The global controller models a set of operating points from the local level to address load balancing and achieve a given objective.

Hierarchical proposals typically utilize a controller running in a centralized manner within the scope of a cluster of nodes. As the number of nodes within the cluster increases, the decision-making algorithm faces the same challenges as traditional centralized approaches, with the execution time resulting in an inability to react to SLA violations. Our proposed framework tackles this by reducing the operations performed by the centralized controllers and moving management functionality to lower execution nodes, therefore providing a hybrid hierarchical decentralized framework. In our approach, each NC autonomously manages the execution node, including detection of stress states and violation of SLAs. These NCs can cooperate with other nodes and higher level controllers to perform management of the infrastructure and workloads.

In Reference 25 a hierarchical approach with VM migration escalation is used. The approach performs the initial assignment of VMs to clusters (containers) and periodically, lower controllers decide what to optimize and pass the decision to parent controllers. The approach considers the time complexity of the decision-making algorithm and places an upper bound on optimisations to be performed by the hierarchical controllers.

Our approach has similarities to Reference 26, which proposed a framework for managing a hierarchical cloud management system. Due to the computational requirements of management nodes, the approach proposes every node is either used as a management node or an execution node, never playing both roles. This contrasts with our approach, in which execution nodes can also perform management roles, resulting in fewer nodes dedicated to the management of the infrastructure. Our approach additionally enables the leaf nodes to operate in a decentralized manner. Proposals typically do not build cooperation between controllers, resulting in controllers operating independently. In contrast, References 21,27-29 proposed controller cooperation. In References 21,28, the lower-level controllers propagated workload satisfaction with assigned resources to a higher controller, which is then used to possibly further optimize resource assignment. In Reference 27, lower-level controllers can escalate a request to a higher controller, rather than waiting for action in the next management cycle. However, escalation only occurred from a mid-level controller within the hierarchy and did not escalate from the lowest-level controllers, which our approach does. Scalability was typically not evaluated in these approaches.

Mesos[30] uses a two-level scheduling mechanism, with a master process managing slave daemons running on cluster nodes. User frameworks that run tasks execute on the slave nodes. The master process makes resource offers to User frameworks, which they can accept or reject. Mesos was evaluated to manage 50,000 nodes. However, user frameworks need to be modified to be Mesos aware.

3.2 | Decentralized controllers

In References 31,32, a heuristic peer-to-peer protocol enables nodes to communicate directly without a centralized controller. Two cooperating nodes determine whether to migrate a VM based on the defined objectives. While Reference 31 did not take into account the cost or duration of the conflict before applying the migration,[32] incorporated migration cost into the decision-making. A periodic node discovery service enables nodes to find new neighboring nodes to communicate with. On each round of the protocol, two cooperating nodes determine to migrate a VM based on defined objectives. Using simulation, the approach claims to manage more than 100,000 nodes. A challenge with decentralized approaches is the lack of a global view of the infrastructure, which impacts the ability to reach a globally optimal solution. In contrast, our approach has the scalability characteristic of decentralized systems and a global view of the infrastructure through controllers at each level of the hierarchy.

A distributed probabilistic algorithm was used in Reference 33 that utilized an overlay network and a decentralized load balancing technique. A decentralized approach is proposed in Reference 34 for managing the workload of large, enterprise cloud data, focusing on reducing energy consumption and SLA violations. The approach uses a *hypercube* structured overlay, with a similar cost to our approach, with N nodes reaching status propagation in $O(\log N)$ rounds.

A decentralized approach is proposed in Reference 35, where nodes broadcast resource requests to all nodes during migration. In contrast, our approach has a bigger view of the infrastructure, through the additional hierarchical controllers, and can consolidate nodes across a whole cluster.

A decentralized self-organizing approach is proposed in Reference 36, where nodes cooperate within the overlay. Migration decisions are performed after an evaluation of the whole overlay state. Similar to our approach, nodes collaborate across the overlay, however, the close physical proximity of cooperating nodes in our approach means gossip traffic and migration traffic are localized within the cluster. Similar to us, the authors also concluded that a centralized view can achieve better consolidation results, as it has a global view of the infrastructure.

A decentralized reinforcement learning-based controller is proposed in References 37,38. Each autonomous node is responsible for managing the performance of its own hosted applications, and collaborates with other nodes. Each node has two monitoring components to keep track of the applications and the system resources and a learning-based controller. However, the scalability of managing a large infrastructure was not evaluated.

To the best of our knowledge, SHDF is the first hybrid hierarchical decentralized framework that enables leaf nodes to operate in a decentralized manner, combining this with the hierarchical escalation of migration and consolidation of VMs across large sections of the infrastructure. This utilizes the benefits of scale in decentralized systems and the global view of hierarchical systems.

4 | HYBRID ARCHITECTURE

The management of infrastructure resources and achieving IPs objectives is a complex challenge. We classify the management process into two dimensions, *Management Algorithm (MA)* and *Management Framework (MF)*. The MA is responsible for deciding how workloads are assigned to infrastructure resources, while the MF enables the MA to execute by providing common functionality, such as hierarchy-level management and aggregation of metrics between nodes. The combined functionality results in workloads executing on infrastructure nodes. In the following sections, we briefly describe the MA and detail our proposal, which is in the form of an MF.

4.1 | Management algorithm

In our previous work,[5] we have shown MAs are widely covered in the literature, and drive the decision-making process in cloud systems adaptation. Several analytical techniques are used, including Control Theory, Heuristics, and Machine Learning. Cloud systems adaptation needs to be invoked to evaluate the infrastructure and determine whether resource reconfiguration is required. The approaches used in the literature fall into a reactive and proactive engagement. Reactive approaches invoke adaptation at defined time intervals or when a monitored metric, for example, CPU utilization, reaches a specific threshold. Proactive approaches predict what demands will be placed on the infrastructure and invoke adaptation ahead of the predicted resource contention point. The MA assigns resources in the infrastructure and regularly assesses the satisfaction of such assignments in achieving a given SLA. The frequency of this assessment is influenced by the time complexity of the algorithm; the lower the complexity, the more frequently the algorithm can be executed. Our proposed architecture enables the MA to run more frequently, which opens the opportunity to apply adaptation of the infrastructure more frequently and could lead to a more optimal assignment of resources.

4.2 | Management framework

The MF provides common utilities that enable the MA to execute, including a mechanism to propagate node state, and a decision engine architecture that may be centralized, hierarchical or decentralized. The MF is the focus of our work in this paper.

The architecture of our proposed framework SHDF, as shown in Figure 1, consists of hybrid hierarchical decentralized controllers operating at different scopes. On the lowest level, every node in the infrastructure contains a NC, which dynamically adjusts resource configurations to satisfy VM demands on each node, shown as a node in Figure 1. A collection of NCs form a cluster of nodes. Each NC cooperates with a LN, which is a higher level controller for all the NCs within a given cluster. Unique to our proposal, the NCs within each cluster are divided into logical groups, called overlays, where a NC cooperates with other NCs within the same overlay. Each NC exists in only one overlay and one cluster. The size and method for constructing an overlay are configurable. This hybrid hierarchical decentralized approach enables nodes to cooperate, and thus obtain the benefit of decentralized architectures. Additionally, the hierarchical element of our proposal enables a cluster wide view to consolidating VMs.

Unique to our approach, the LN operates as a normal node within the infrastructure in addition to its management responsibility toward the cluster. At the highest level, the Data Center Controller (DC) manages the controllers one level below it.

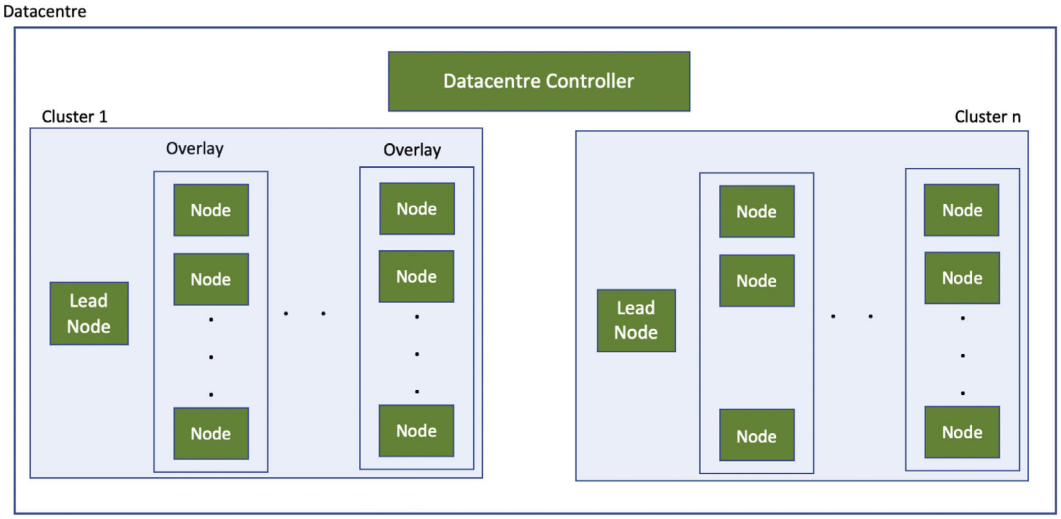

FIGURE 1 Scalable hierarchical decentralized framework escalation architecture

SHDF attempts to service resource requests at the lowest local level possible, to reduce the overhead of servicing the request[39] and to reduce the performance impact of migrating VMs across cluster boundaries.[40] The proposed MF provides mechanisms for migration and additional constraints can be provided by the chosen MA. An example constraint could be to perform a cost benefit analysis and only perform the VM migration when the additional energy consumption is lower than the SLA violation.

The following subsections describe the components of SHDF and the functionality it provides to the MA.

4.2.1 | Overlay management

Execution nodes operate in a decentralized way within the boundary of a cluster and are organized in multiple overlays. An overlay is a fundamental concept to reliable multicast, as it abstracts the details of the underlying physical network by building a virtual network on top of it, which can be seen as a graph that represents the links between nodes. To construct overlays, typically two main approaches exist:[41] structured and unstructured protocols. Structured approaches tend to be efficient in terms of the number of links. However, they are sensitive to node failure, because the overlay structure takes into account metrics such as latency or bandwidth, and upon node failure, the structure needs to be rebuilt. In unstructured approaches, links are established randomly among the nodes. To ensure overlay stability, multiple links are established. However, this can cause nodes to receive multiple copies of a given message through its different neighbors.

SHDF is closer to structured approaches,[42] where the overlay is constructed to span a subset of a cluster, and by definition is layered on a physical structure of nodes. This solves the proximity challenge in structured approaches, without needing to associate latency metrics with each of the overlay links. When a new node is added to the infrastructure, it gets added to a cluster and assigned an overlay. SHDF enables the MA to determine where a new node is added. When a node is added to an overlay, it receives details of neighboring nodes from the same overlay, through the data dissemination mechanism described below.

4.2.2 | Data dissemination

In centralized and hierarchical architectures, a central component receives state updates from all the nodes under its management to enable the MA to adapt the infrastructure. SHDF removes this dependency and enables the nodes to exchange their state within an overlay. Each node holds a local cache of the state of other nodes in the overlay, and to populate this cache, nodes exchange state messages. A node selects another node at random from the overlay and exchanges its view of the overlay. We used a pull and push gossip approach, where node x sends new state updates to node y, and retrieves new states updates at node y. A new node state takes $O(\log N)$ rounds to reach all nodes, where N is the number of nodes in the overlay.[43] A gossip round completes when every node has initiated an exchange exactly once. The local cache within each node allows the MA to perform VM migration to other nodes within the overlay and allows the LN to perform VM consolidation across the cluster.

Similar to Reference 44, a NC regularly exchanges state with other NCs and holds a cache of free capacity of other cooperating controllers in the same overlay, shown in Table 1. The gossiping algorithm is shown in Algorithm 1. We enhance this further and make nodes gossip when there

TABLE 1 Exchanged data between overlay nodes

Field	Description
NodeID	The ID of the node, data belongs to
VMSize[NumberAvailable]	number of predefined VM sizes that can be hosted on this node
TimeStamp	Time stamp for captured metrics

Algorithm 1. stateExchange@Node

```
1:  procedure EXCHANGESTATE(localState)
2:      overlay ← getOverlayMembers - thisNode
3:      randomMember ← random(overlay)
4:      remoteState ← randomMember.gossip(localState)
5:      for i in remoteState do
6:          remote ← remoteState[i].timeStamp
7:          local ← localState[i].timeStamp
8:          if remote>local then
9:              localState[i] ← remoteState[i]
10:         end if
11:     end for
12: end procedure
13: procedure GOSSIP(remoteState)
14:     respond to caller ← getLocalSate
15:     receivedState ← remoteState
16:     for i in receivedState do
17:         recived ← receivedState[i].timeStamp
18:         local ← localState[i].timeStamp
19:         if received>local then
20:             localState[i] ← receivedState[i]
21:         end if
22:     end for
23: end procedure
```

is a state change such as inward migration, or a node is being shut down. Each node will keep the latest metrics seen, and every time exchange of state occurs the timestamp is used to decide if each exchanged entry is later than any existing entries. If it is, the node's internal state is updated, otherwise, the data exchanged is discarded. The MA running on each node can use these metrics to enable nodes to cooperate in resource allocation requests, which will be described later.

As in Figure 2, the LN is an ordinary node and will exchange its state. The LN is a member of all the overlays in the cluster, and will receive state exchanges from all the nodes within the cluster. To restrict the view of a node and reduce redundancy of state management, when a node gossips with the LN, it will only receive state updates from its overlay and not from other overlays the LN knows about.

The LN sends aggregated metrics for all the nodes within its cluster, to other cooperating controllers. The LN participates in an overlay with other LNs, in a similar way to NCs, and it can exchange aggregated metric data. The parent controller to the LN is always included in each LN overlay, resulting in state exchanges between the LN in an overlay and the DC controller, in a similar way to the exchanges between a LN and a NC, as shown in Figure 3, resulting in an aggregated view of the whole infrastructure at the DC controller level. This exchanged state data is then used during adaptation actions, which we describe in detail in the VM Migration section.

For the gossip protocol, the exchange frequency and fan out values influence propagation speed and reliability. The fan out value determines the number of times a new state is exchanged by a node. For example, when a node receives an update and the fan out value is 2, the node will exchange this newly received state in the next 2 cycles of the gossip protocol. Through experimentation, we chose a fanout value of 2 as a tradeoff of bandwidth consumption with propagation speed.

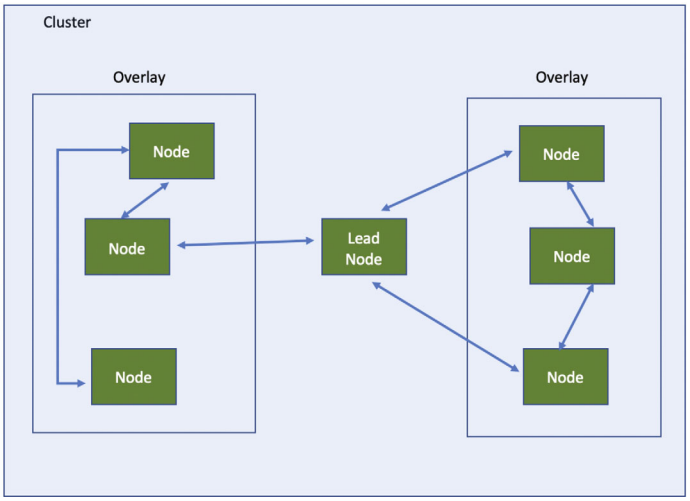

FIGURE 2 A gossip cycle, within a cluster

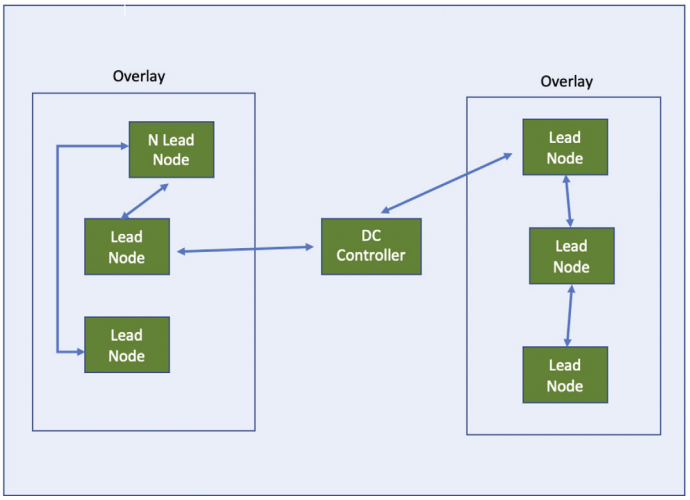

FIGURE 3 Lead node overlays

4.2.3 | Controller functionality–VM migration

When a node cannot satisfy the demands of the VMs it hosts, it starts an escalation process that aims to resolve the request at the lowest possible level. The MA running on the stressed node and the LN cooperate to resolve the escalated VM migration, by using our provided framework mechanisms.

Table 2 shows the available methods to support migration and escalation of a VM. SHDF uses different methods to migrate within the overlay and escalate outside it. As shown in Figure 4 and Algorithm 2, when a NC needs to perform a migration, it attempts to service this through a series of escalation steps. The process starts within the NC's overlay by sending a request to other nodes within the same overlay (step 1), by using the accumulated metrics of other nodes. If a target node is available and accepts the migration request, then the migration process completes for this cycle. If the selected target node does not accept, other nodes within the overlay are attempted until no further options are available within the overlay. If a target is not available within the overlay, the stressed NC escalates the VM migration request to the LN (step 2), and the MA running on the LN can query the cluster records from all the overlays, which has state data from all nodes in the cluster, to find a suitable node to house an escalated VM. If the LN locates a target within the cluster, the migration request is forwarded (Step 3). If this does not succeed, then other nodes within the cluster are attempted, in a similar way to the overlay step. If the LN cannot find a suitable target for the migration within the cluster, it will use its knowledge of other available clusters, through participation in the LNs overlays, to forward the migration request to another LN (Step 4). This target LN will repeat the process performed by the forwarding LN, and attempt nodes within this cluster (step 5). If a suitable target is found

TABLE 2 Controller functionality for Migration of virtual machines

Method	Source	Destination
Migrate VM	Node	Node
Escalate VM	Node	LN
Place escalated VM	LN	Node
Escalate VM	LN	LN
Escalate VM	LN	DC Controller
Place escalated VM	DC controller	LN

Abbreviations: DC, data center controller; LN, lead node; VM, virtual machine.

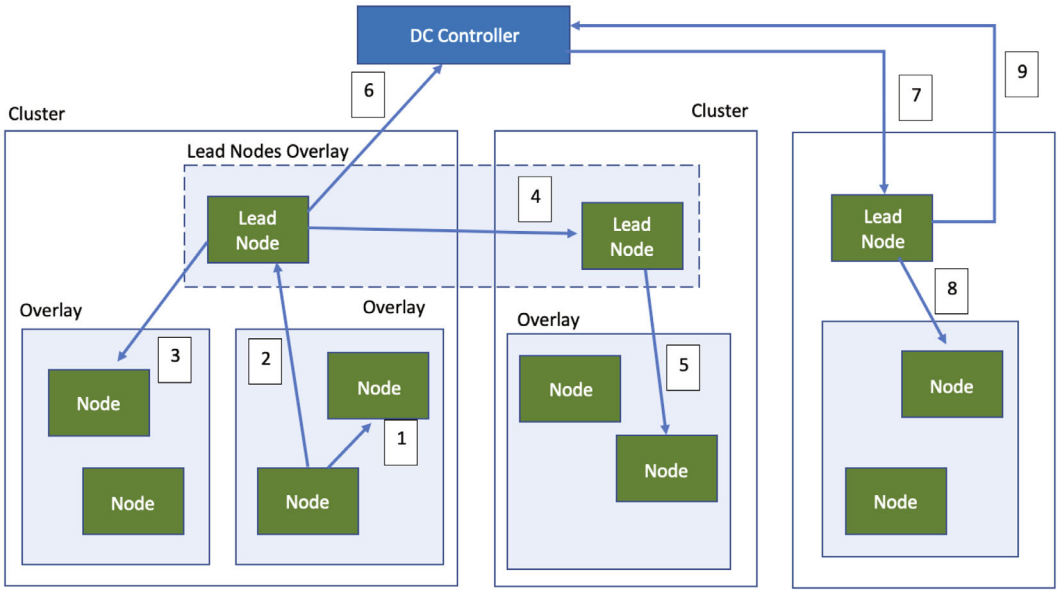

FIGURE 4 VM migration and escalation process

the process competes. If the LN in step 4 cannot find a target, the request is rejected back to the forwarding LN, which will attempt other LNs in its overlay. If a suitable target is found through another LN, the process completes. If this fails to find a suitable target, the request is escalated to the DC Controllers (step 6). The DC has a view of the entire infrastructure and can forward the request to other LNs (step 7), which the original LN does not cooperate with. This recipient LN will repeat the process carried out by other LNs in the escalation chain (step 8), and if a target is found the process completes. If a target is not found, the recent LN will reject the request back to the DC controller (step 9). The DC controller will attempt other LNs, which have not already been tired, until a target is found or all LNs have been attempted. If a target is not found then the infrastructure is highly stressed and the request is rejected back to the original escalating NC. In each of these escalation phases, the MA uses data from the data dissemination to decide on the list of targets to forward a request to.

As requests progress through the escalation process, they are assigned an increasing priority, which can be used by the MA in the decision-making process. For example, the MA may choose to prioritize finding a host for an escalated VM compared to a new VM placement.

4.2.4 | Controller functionality - Consolidation

At configurable time intervals, 1 h by default, each of the management controllers, and LNs can invoke a consolidation process where the MA can examine the state of the infrastructure and for every node under its management, decide to:

- Migrate some VMs from a node,
- Migrate all VMs off a node and switch the node off, or
- Make no change.

Algorithm 2. migrateVM@Node

```
1: procedure MIGRATEVM(localState)
2:     isStressed ← isNodeStressed(localState)
3:     if isStressed then
4:         targets ← validOverlayTargets
5:         vmToMigrate ← MA.orderVMs
6:         targetNode ← MA.findTargetNode(targets, vm)
7:         if targetNode == valid then
8:             invalidateTarget
9:             migrate(vm, targetNode)
10:        else
11:            esclateToLead(vm)
12:        end if
13:    end if
14: end procedure
15: procedure ESCLATETOLEAD(vm)
16:    targetNodes ← validClusterMembers - sendingOverlay
17:    targetNode ← MA.findTargetNode(targetNodes, vm)
18:    if targetNode ≠ null then
19:        invalidateTarget
20:        migrate(vm, targetNode)
21:    else
22:        potentialClusters ← ClustersInOverlay - sendingCluster
23:        targetCluster ← findTargetCluster(potentialClusters, VM)
24:        if targetCluster ≠ null then
25:            targetCluster.esclateToLead(VM)
26:        else
27:            esclateToDataCentreController (vm)
28:        end if
29:    end if
30: end procedure
31: procedure ESCLATETODATACENTRECONTROLLER(vm)
32:    potentialClusters ← allClusters - sendingCluster
33:    targetCluster ← getAvailableCluster(potentialClusters)
34:    targetCluster.esclateToLead(vm)
35: end procedure
```

The advantage of SHDF is it allows the nodes to primarily operate in a decentralized manner for time-sensitive operations such as VM migrations, which could improve SLA violation metrics. For non-time-sensitive operations like VM placement and consolidation, SHDF enables the running of time complex algorithms across parts of the infrastructure or even the whole infrastructure. Compared to typical decentralized approaches,[31,32] this will enable the MA to have a more comprehensive view of the infrastructure during these operations, leading to opportunities for more optimal VM migration.

The consolidation process starts by marking all active nodes that have no running VMs for shutdown. It then excludes nodes that have incoming migrations. The remaining under-utilized nodes become a possible source for migration of VMs, with partially utilized nodes becoming target nodes. The process proceeds to select VMs from under-utilized nodes, and to avoid conflict a source is never used as a target node.

Consolidation of VMs into fewer nodes is a complex process, that aims to conserve energy consumption. However, there are other considerations including ensuring the reliability of the consolidation process. In our work, we assume consolidation can be performed with no reliability constraints, as this is outside the scope of this paper.

5 | EXPERIMENT SETUP AND EVALUATION

In our experiments, we evaluate the Quality of Service (QOS) metrics of our proposed hybrid architecture approach versus centralized, hierarchical and decentralized architectures, with varying workloads and VM configurations. There is an abundance of MAs in the literature that have been implemented using a centralized architecture, and we discuss these in more detail in subsequent sections. We use the same MAs in our comparison of the different approaches, and in doing so we focus the evaluation on the architecture.

We used simulation to facilitate the rapid development of experiments in large data centers. We selected DCSim[45] because of its extensibility and an existing implementation of a centralized architecture, allowing us to create a baseline comparison for our proposed MF. We additionally implemented 8 MAs, detailed later in this section.

We instantiate SHDF with three levels of controllers, running on the root of the data center (DC Controller), the cluster manager (LN), and leaf nodes (NC).

5.1 | Simulator setup

DCSim allows a VM to use more CPU than reserved, up to an amount that does not impact other VMs. Like,[27] we use a CPU utilization threshold of 90% for high, indicating stress level, and we used 60% for low, indicating low utilization.

In DCSim, an application is modeled as an interactive multitiered web application. Each application has a specified client think time, a workload component and a request service time, which is the amount of time required to process each incoming request. The workload defines the current number of clients connected to the application, which can change at discrete points in the simulation based on a *trace* file. The resource requirements are defined in terms of the expected amount of CPU, memory, bandwidth, and storage. DCSim treats bandwidth and storage as fixed requirements, however, CPU requirements can be varied across the simulation based on the VM demands. DCSim applies a cost to VM migration including the time taken for migration, as a function of memory consumed by a VM, and factors in the bandwidth required for the VM migration on the hosting node. Additionally, the boot time of a switched off node has an elapsed time cost. The time taken to switch on a node for migration is reflected in the period the VM is in a stressed state, and therefore the SLA achieved by a VM. Due to the complexities of building accurate power models, we focus our investigation on scalability metrics.

Live VM migration is a complex process and has been researched widely.[46] In our work, we assume VM migration can be carried out reliably and with no interruptions, and we do not cater for scheduling the order of VMs during migrations, as it is outside the scope of this paper.

Each experiment is run to simulate 24 h of elapsed time and each simulated application contains a workload trace from the public traces included in DCSim. The simulation is designed to instantiate a certain number of VMs per hour, where each VM will run one of the public traces in a round-robin approach. For example, when 1000 VMs are created per hour and five traces are used, 200 VMs will use one of the traces. Each trace sets the size of incoming traffic to a VM and typically changes every 5 min. As the round-robin approach is used in the experiments for each MA and MF evaluation, we assume this is a valid method to instantiate the VMs. Each VM runs for 10 h before shutting down.

DCSim tracks the energy consumption of each node in the infrastructure. Each node has an energy model that defines how much energy it consumes given its current CPU utilization. DCSim uses the SPECpower benchmark data[47] to build a table of energy consumption at each CPU utilization level. DCSim uses these values and calculates intermediary values using linear interpolation.

5.2 | Workload, SLA violations and energy model

We run the experiments at a load that requires more than 70% of the CPU resources of active nodes. Each experiment is run to simulate 24 h of elapsed time and each simulated application contains a workload trace based on the number of incoming requests to web servers from publicly available traces. We used the following traces included with DCSim: Google 1, Google 3, EPA, and Clarknet. Figure 5 shows the normalized shape of the workload requests for each of these traces. We create VMs with different cores and RAM configurations, as shown in Table 3. DCSim can track total energy consumption within the data center, by mapping CPU utilization of a node to a defined energy consumption amount, and tracking this accumulatively for all nodes.

5.3 | Compared MAs

To evaluate our proposed architecture, we build on the work in Mann[48] and implemented similar MAs within DCSim.

Common dimensions that determine how these MAs operate are: (1) *Overload detection mechanism*, which is how the MA decides a node is at a utilization point that could impact a metric such as SLA; (2) *VM Selection*, which is the mechanism used to select the VM to be migrated; (3) *Node*

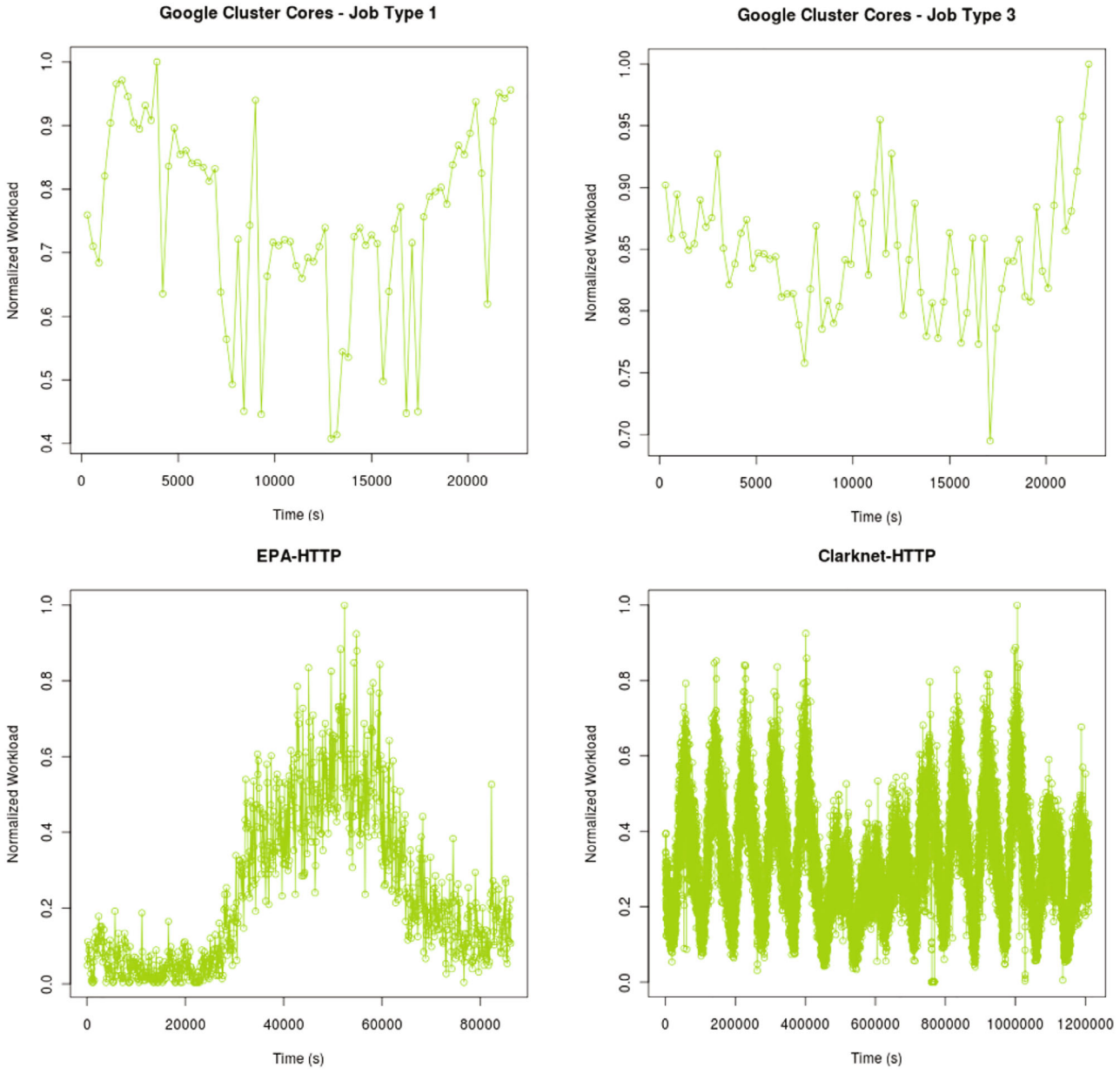

FIGURE 5 Traces used

TABLE 3 Experiment configuration

Config	Config options	Base config
VM Core (MHz)	1000, 1300, 2500	Round Robin (1000, 1300, 2500)
Node capacity (MHz)	3000	3000
Number of cores	1,2	Round Robin (1,2)
VM memory (MBs)	1024,2048	Round Robin (1024, 2048)
Workload	clarknet, EPA, Google 1, Google 3	Round Robin (clarknet, EPA, Google 1, Google 3)
Number of nodes	1000, 2000,3000, 4000, 5000	1000
Node stress check frequency	2 min	2 min
Application service time	0.2 s	0.2 s

TABLE 4 Dimensions of policies

		Dimension		
MA identifier	**Management algorithm**	***Stress detection***	***VM selection***	***Node selection***
P1	Lago[49]	Local Regression	Not specified	Highest energy efficiency (plus further rules for tie-breaking)
P2	Guazzone[50]		Highest CPU load	Highest free CPU capacity (plus further rules)
P3	Demand Risk[51]		VMs of the most loaded PM	Lowest demand satisfaction
P4	Shi Absloute52,[52]		Highest CPU load on the PM with lowest absolute capacity	Highest absolute capacity
P5	Shi Percentage[52]		Highest CPU load on the PM with lowest utilization ratio	Highest utilization ratio
P6	Chowdhury[53]		Highest CPU load	Highest increase in energy consumption
P7	Beloglazov[54]		Highest CPU load	Smallest increase in energy consumption
P8	Tighe[45]		VMs that bring load to below stress	From partially utilized, under utilized, empty nodes

Selection, which is the approach used to select the target node for MV migration. Table 4 shows these dimensions for each of the compared algorithms. The following are the management algorithms used in our experiments.

P1[49] uses a set of rules for selecting the next node for a VM migration. No rules are specified for selecting a VM for migration within the node, and in our implementation, we use the highest CPU utilization for VM migration. P2[50] used a best-fit-decreasing heuristic, and in case of a tie, target nodes are chosen in increasing order of idle energy consumption. P3[51] selects a target node based on historic workload data, and attempts to achieve the smallest risk of demand dissatisfaction using a scoring mechanism named *demand risk*. Shi[52] evaluated multiple algorithms, and similar to Reference 48, we used the AbsoluteCapacity (P4) and PercentageUtil (P5). In both algorithms, source nodes are sorted according to utilization and capacity, and then the algorithms iteratively attempt to free the smallest node by trying to migrate its VMs to the biggest nodes. P6[53] created a Modified Worst Fit Decreasing VM Placement, in which target node selection favors nodes with the highest increase in energy consumption. The authors argue the counterintuitive approach has merits over the best-fit heuristic. P7[54] aims to remove VMs from overloaded nodes and removes VMs from underloaded nodes so that they can be switched off. Stress detection uses Local Regression to decide on loads that are overloaded. Once a node is deemed overloaded a VM is selected using the highest CPU utilization. Targeting node selection uses a Modified Best Fit Decreasing approach to migrate a VM. P8[45] selects VMs that reduce the node stress as candidates for migration, and order candidate target nodes based on partially utilized, under-utilized and empty nodes.

5.4 | Compared MFs

We evaluate our proposed hybrid hierarchical decentralized architecture against centralized, hierarchical and decentralized architectures.

The centralized approach[45] has a single central decision-making component for VM migration. This controller receives state information from all of the nodes periodically, and at regular intervals invokes an adaption process that checks for stressed nodes and attempts to migrate VMs from stressed nodes to other available nodes within the data center.

The hierarchical[27] approach distributes the decision-making to multiple controllers: a central data center controller, a cluster controller and a rack controller. The rack controller receives state information from all of the nodes it houses and at regular intervals invokes an adaption process, which checks for stressed nodes and attempts to migrate VMs from stressed nodes. The migration process has some similarities to our escalation process where the rack controller will first find target nodes within the rack. If this fails, the rack controller will forward the migration request to the cluster controller. If this fails, the Cluster controller will forward the migration request to the DC, which can forward it to other clusters in the data center.

The approach in Reference 35 decentralizes the decision-making to individual nodes, which cooperate with other nodes to perform VM migration. Each node does not hold state information about other nodes in the infrastructure, and when a node becomes stressed, it broadcasts a migration request to the whole infrastructure and waits for replies. At the end of the waiting period, the node processes the offers it received and chooses to accept one or power on a new node.

5.5 | Modeling the impact of centralized decision-making

DCSim[45] applies a migration cost once a VM is selected for migration, by adding additional time to complete the migration based on the amount of memory used by the VM. However, DCSim does not account for the time it takes to execute the decision-making process or the impact of such time. The length of the decision-making process impacts stressed nodes by increasing the amount of time the node stays stressed. In a centralized architecture, all nodes are used as input into the decision-making process. Therefore, the execution of decision-making could get progressively higher, as the number of managed nodes increases.

To capture the cost of the decision-making, we extend DCSim to measure the amount of time during decision-making, and add this time to the VM migration duration. As the decision-making execution time varies based on the MA and the hardware it is running on, we add a configurable scaling factor that can be applied to the measured execution time.

5.6 | Data center

Our experiments use nodes model ed as ProLiant DL380 G5 Quad Core,[55] with 2 dual-core 3GHz CPUs and 16GB of memory. The number of nodes used is specified in each of the experiments, with a minimum of 1000 nodes. We assume that the data center supports live VM migration, as this technique is currently supported by most major hypervisor technologies, such as VMware[56] or Xen.[57]

To minimize the number of variables in our experiments, we chose to keep a homogeneous infrastructure. The various parameters used in our evaluation are outlined in Table 3.

5.7 | Evaluation

Our goal is to demonstrate improved QOS metrics compared to centralized, hierarchical and decentralized architectures, and we evaluate these by examining the following metrics: *SLA violations*, *Energy consumption* and *Total Migration Traffic*.

Simulated applications are modeled as an interactive web server, running inside a VM. *SLA Violations* are the instances when the allocated CPU is less than the requested CPU. *Total Migration traffic* is the amount of additional data that passes through the network due to VM migration.

We evaluate all architectures under varying scenarios to understand the impact on our chosen metrics. Initially, we evaluate a *mixed workload* scenario to represent the varying workloads deployed in data centers. To understand the impact of specific workloads, we evaluate these individually. We additionally examine scalability and how the approaches cope with the dynamic arrival rate of new workload.

5.7.1 | No adaptation

In this experiment, we used a combined workload of all traces in a Round-robin approach, where each created VM uses a different workload trace. We used the centralized architecture with 1000 homogenous nodes and no adaptation is invoked. When a node becomes stressed, it remains stressed for the remainder of the experiment, and Table 5 shows the results.

5.7.2 | Mixed workload assessment

In this experiment, we used a combined workload of all traces in a Round robin approach, where each created VM uses a different workload trace, in a deterministic order. The mixed workload simulates different workloads that could be experienced in a data center. The workload arrival rate is the frequency that new application placement requests arrive at the data center. For this experiment, we use an arrival rate of 90 new applications per hour, with each application running for 10 h before shutting down. The experiment simulates 24 h of elapsed time. We use 1000 nodes and the base configuration in Table 3. We captured metrics for SLA violation, energy consumption and the total migration traffic.

TABLE 5 No adaptation metrics

Number of service level agreement violations	Energy consumption (KWh)
27,053	1745.51

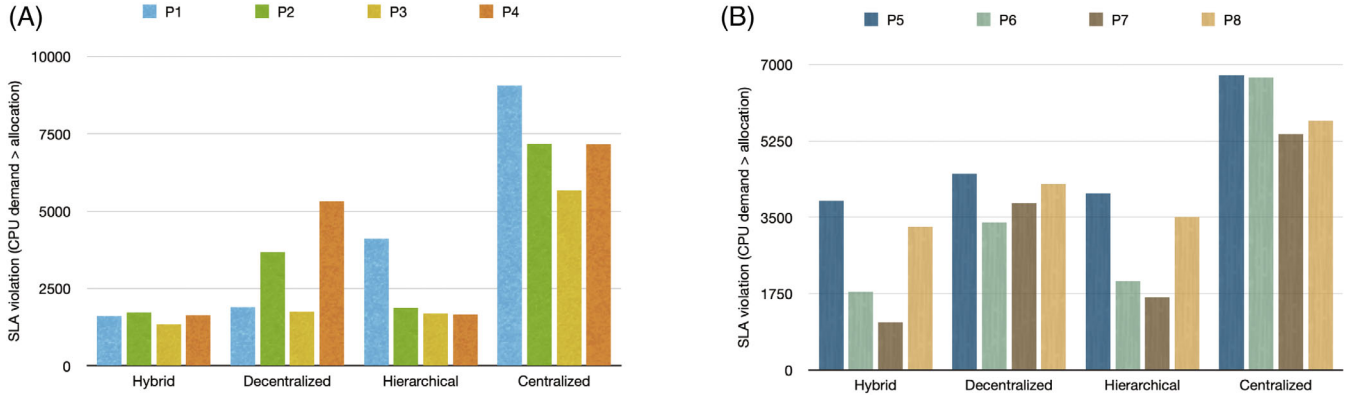

FIGURE 6 Service level agreement (SLA) violation instances when demand > allocation, (A) P1–P4; (B) SLA Violations P5–P8

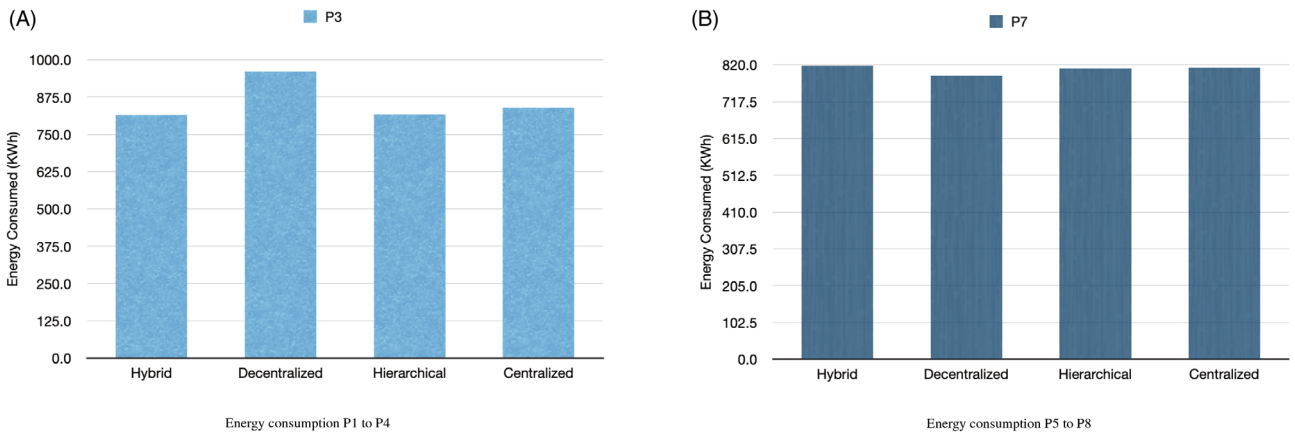

FIGURE 7 Energy consumption. (A) P3; (B) P7

The SLA violation results using the eight MAs are shown on Figure 6, and to aid readability we display MAs (P1 to P4) on Figure 6A and MAs P5 to P8 on Figure 6B .

The hybrid architecture achieved lower SLA violation using all MAs, compared to other architectures, with to 29%, 26%, and 321% against the decentralized, hierarchical, and centralized approaches, respectively, for the P3 MA. Examining the performance of the MAs, P3 and P7 performed best overall across the different approaches. A contributing factor to the lower numbers of SLA violations in the hybrid architecture is each target node autonomously accepts incoming VM migration requests based on the current state of the node. In contrast, the centralized and hierarchical architectures use the last seen state collected at an earlier point in time. This could result in target nodes accepting VMs when in a stressed state. When this occurs, it will trigger an additional subsequent migration, which increases the length of time the VM is not receiving the required CPU demand, leading to higher SLA violations. As the centralized architecture uses a single decision making point, it takes longer to make decisions, and it is unsurprising it has the highest number of SLA violations compared to all the other approaches. The decentralized approach relies on broadcast cooperation for VM migration and thus has a slower decision-making process compared to the hybrid architecture, which keeps a state of cooperating nodes within an overlay.

Compared to the *No adaptation* experiment, all architectures reduced SLA violations. Additionally, adaptation conserved energy even though it can switch on nodes to perform the migration, as the consolidation process can switch nodes off and conserve energy. Adaptation achieved an improvement in energy consumption of up to 128.8%, 16.7%, 28.1%, and 141.9% for the hybrid, decentralized, hierarchical, and centralized approaches.

The decentralized approach does not keep a local state and needs to wait on broadcast replies to make a migration and consolidation decision. Additionally, the decentralized approach keeps nodes switched on, Figure 7, before they are consolidated and switched off, and thus consumes more energy compared to all other approaches. An advantage of this additional energy consumption is the migrations will not wait for nodes to boot up, compared to the other approaches. This aids the decentralized approach in achieving its SLA violation performance. The Hybrid architecture had a strong performance with a mean difference of 47% and 16.1% lower energy consumption compared to the decentralized and hierarchical approaches, respectively. The Hybrid approach had a similar energy consumption profile to the centralized approach.

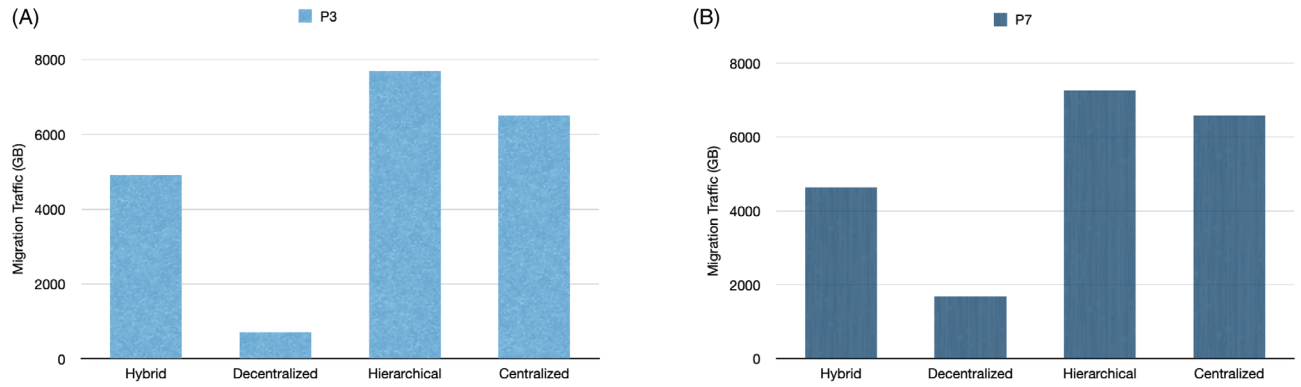

FIGURE 8 Migration traffic (GB). (A) P3; (B) P7

Examining the migration traffic, shown in Figure 8, the decentralized approach performs a lower number of migrations, as it has to wait to receive offers for advertised migrations and thus consumes less migration traffic. In contrast, the hierarchical approach performs more migrations to archive its SLA violation performance and carries out more migrations than the Hybrid and decentralized approaches.

The hybrid architecture, by design, uses an escalating level of migrations, starting within the overlay, then within the cluster and then to other clusters. The hybrid architecture averaged 97.8% of migrations within the same overlay and the hierarchical averaged 85.5% of migrations within the same rack. Therefore, the hybrid approach reduces the migration distance, which reduces the impact of VM migration.[58] In contrast, the centralized and decentralized approaches had no awareness of target node proximity and could choose a target node further away from the source node.

To examine the impact of migration instability,[42,59,60] a VM being migrated several times during its lifetime, we choose a high-performing MA, P3. Figure 9 shows VM migration buckets, representing VMs being migrated exactly once, twice, etc, and the migration count in each bucket.

The P3 MA under the hybrid architecture had 23 occurrences of a VM being migrated exactly four times, compared to 272 times in the centralized architecture and 133 in the hierarchical. The decentralized performed fewer migrations due to the dependency on the broadcasting mechanism. The majority of VMs are migrated once in all architectures. However, the centralized and hierarchical architectures experience higher VM migration instability, with more VMs being migrated twice or more (cumulative sum of bucket 2 and above), and in the centralized and hierarchical architectures the worst affected VM was migrated 15 times, compared to seven times in the hybrid architecture.

Another factor influencing higher SLAs in the centralized architecture is the execution time of the decision making process. When a node becomes stressed in the hybrid architecture, the node searches for a target migration node in a smaller search space, starting within the overlay, compared to the global search space in the centralized architecture, and can identify a target quicker. As the number of nodes increases, the difference in search space becomes more pronounced. The longer it takes to find a target node, the longer a VM is in a stressed state and thus incurs SLA violations. The hybrid architecture will typically search a fixed number of nodes in the overlay and achieve lower SLA violations.

In summary, the hybrid architecture was able to achieve improved metrics compared to the other approaches, with lower SLA violations, comparable or lower energy consumption and lower total migration traffic.

5.7.3 | Workload impact

To investigate the impact of workload further, we use the individual workloads from the previous experiment and evaluate them individually. For this experiment, we use an arrival rate of 90 new applications per hour, with each application running for 10 h before shutting down. The experiment simulated 24 h of elapsed time. We use the base configuration in Table 3.

The results for the Google 1 workload are shown in Figure 10 and show the effect on SLA violations and energy consumption. We chose the best performing MAs from the previous experiment, P3 and P7. This workload has an average of 0.74 normalized load and thus high stress. The hybrid approach outperformed other approaches by 118%, 28.5%, and 428.7% for the decentralized, hierarchical, and centralized, respectively (P3 MA). The hybrid approach performs even better on this workload, compared to the mixed workload, indicating that more stressful workloads benefit from the speed of the decision making process in the hybrid approach.

Examining the energy consumption, the hybrid approach consumes 6% and 1% more energy compared to the hierarchical and centralized approaches, respectively, for the P3 MA, as shown in Figure 10B. This is due to the faster decision making process in the hybrid, and is offset by the significantly better SLA violation performance, in Figure 10A.

The results for the Google 3 workload are shown in Figure 11, and show the effect on SLA violations and energy consumption. This workload has an average of 0.83 normalized load and is the highest workload stress we have evaluated. The hybrid approach outperformed other approaches

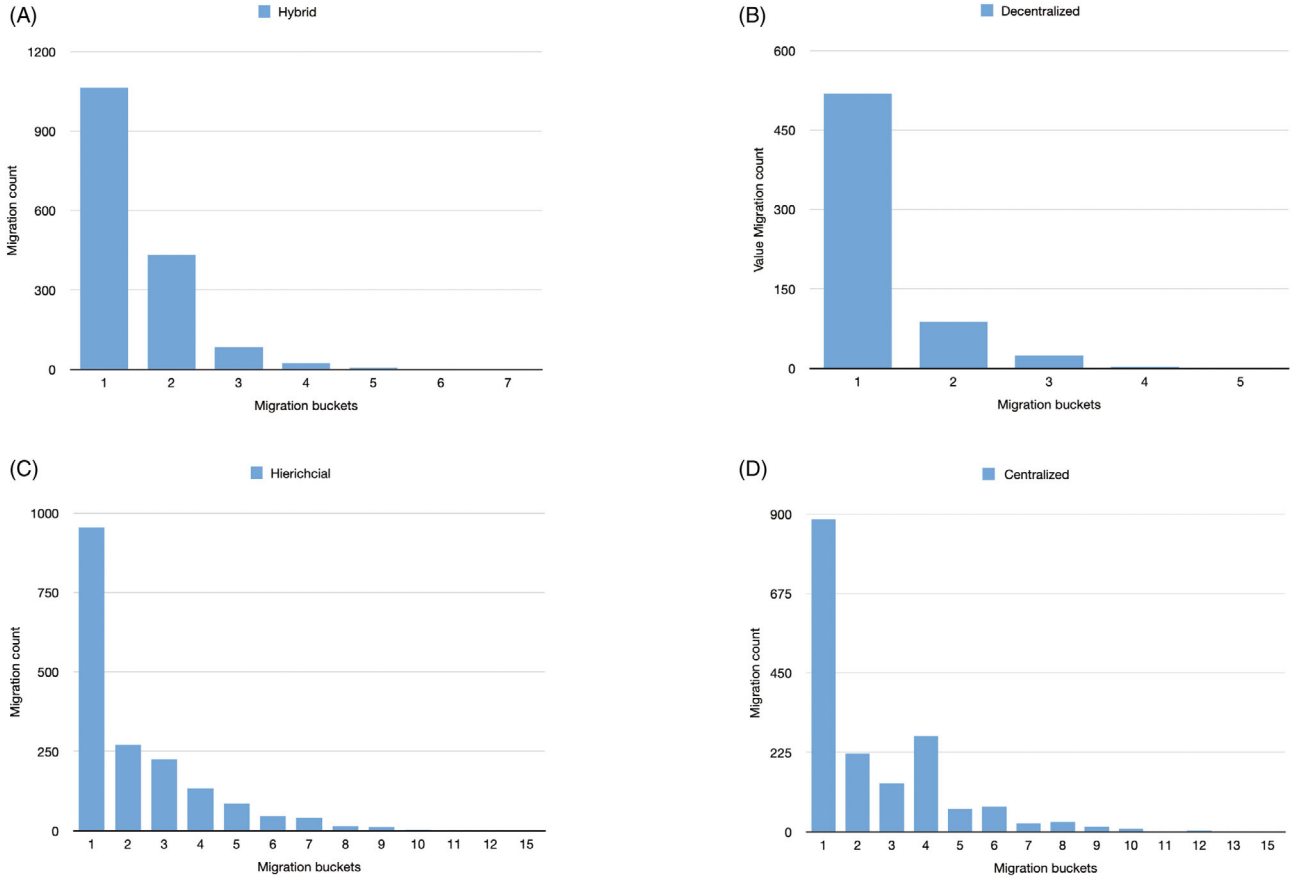

FIGURE 9 Migration buckets for MA P3. (A) Hybrid; (B) decentralized; (C) hierarchical; (D) centralized

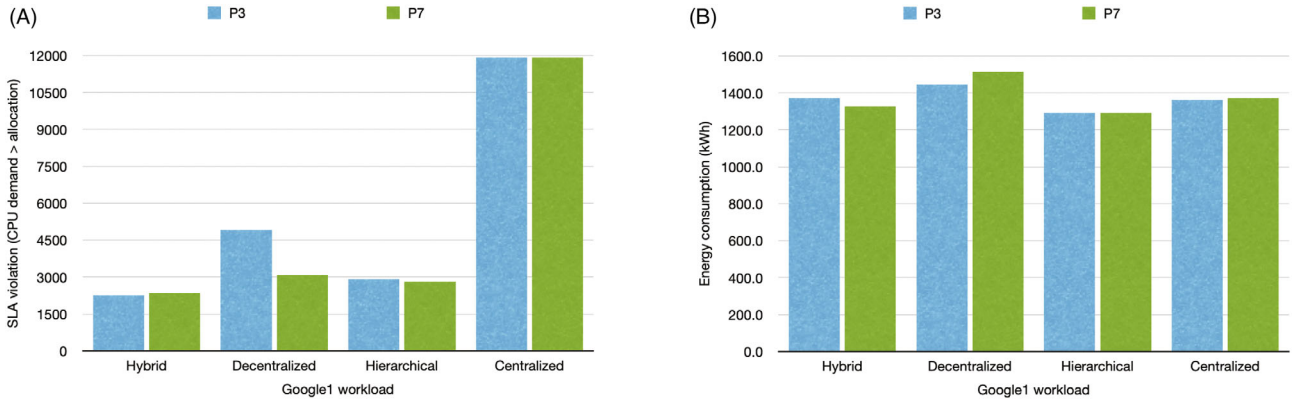

FIGURE 10 Google 1 workload evaluation. (A) Service level agreement violations; (B) Energy consumption

by 434%, 92.%, and 796% for the decentralized, hierarchical, and centralized, respectively (P3 MA). The hybrid approach performs even better on this workload, compared to the mixed workload, indicating that more stressful workloads benefit from the speed of the decision-making process in the hybrid approach.

The Clarknet workload has an average of 0.31 normalized workload and exhibits lower average stress than the Google 1 and Google 3 workloads. The results for SLA violations and Energy consumption are shown in Figure 12. The hybrid approach achieves lower SLA violations compared to the hierarchical and centralized approaches, but not against the decentralized approach, which has nodes switched on by default to satisfy the workload demands. This results in the decentralized consuming 72%, 67%, and 79% more energy compared to the hybrid, centralized, and hierarchical approaches, respectively, for the P7 MA. However, for the P3 MA, the decentralized approach was able to achieve the lowest SLA violations and least energy consumption for this workload.

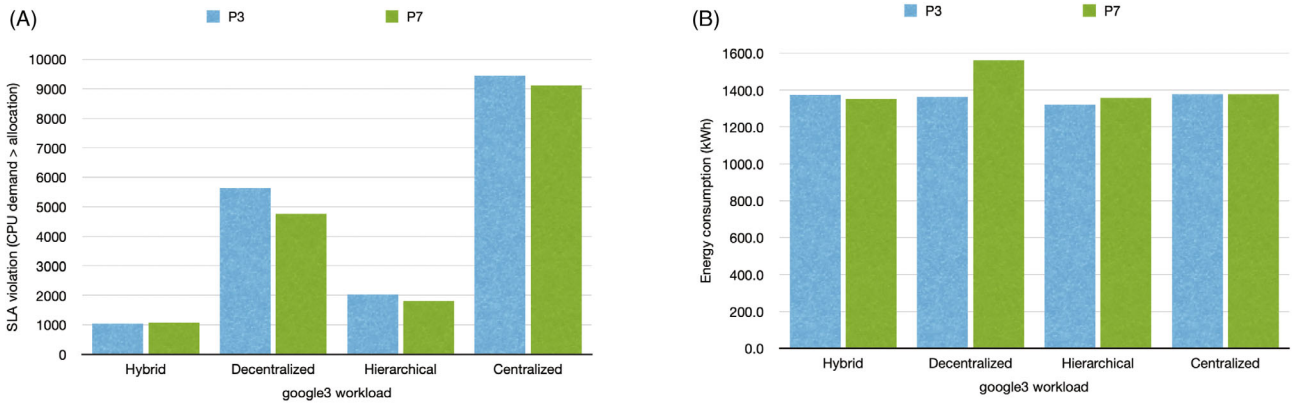

FIGURE 11 Google 3 workload evaluation. (A) Service level agreement violations; (B)energy consumption

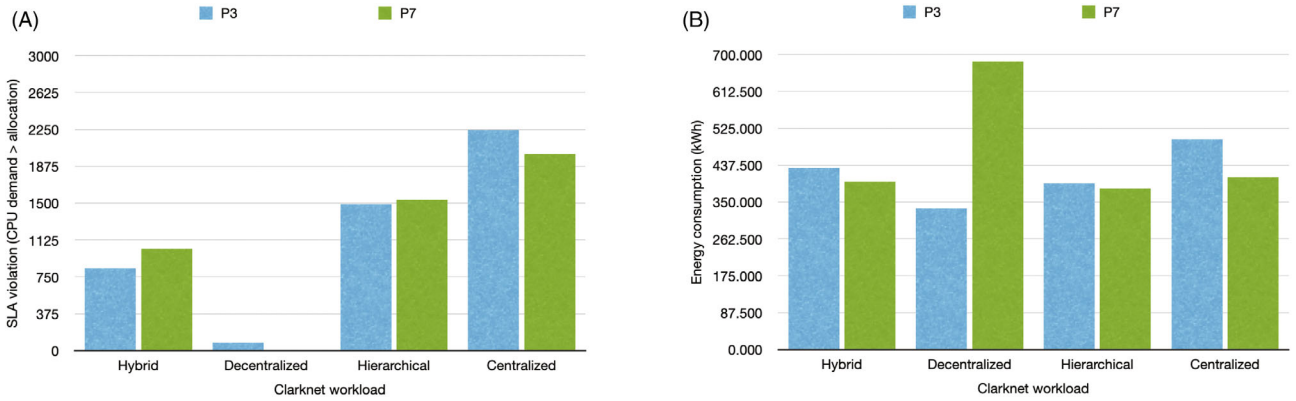

FIGURE 12 Carknet workload: (A) Service level agreement violations; (B) energy consumption

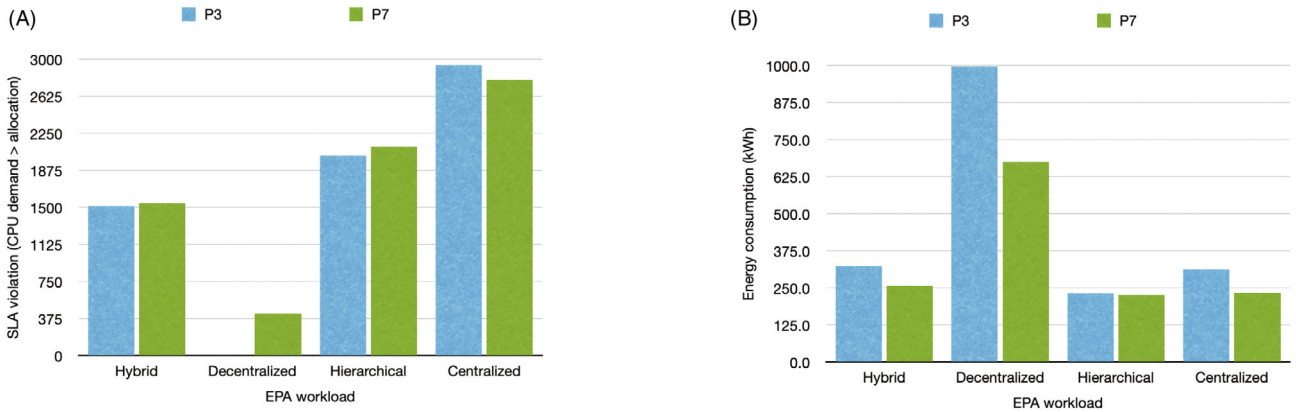

FIGURE 13 EPA workload: (A) Service level agreement violations; (B) energy consumption

The EPA workload has an average of 0.24 normalized workload and exhibits lower average stress compared to the Google 1 and Google 3 workloads. The results for SLA violations and Energy consumption are shown in Figure 13. Similar to the Clarknet workload, the hybrid approach achieves lower SLA violations compared to the hierarchical and centralized approaches, but not against the decentralized approach, which has nodes switched on by default to satisfy the workload demands. In contrast, the decentralized P3 MA has consumed 209%, 219% and 331% more energy compared to the hybrid, centralized and hierarchical approaches respectively. A similar effect occurred on the decentralized P7 MA, which achieved the lowest SLA violations out of all the approaches, but used 163%, 200%, and 189% more energy compared to the hybrid, hierarchical, and centralized, respectively.

Examining the results of the individual workloads, we conclude that on stressful workloads (Google 1/Google 3) hierarchical-based approaches (hybrid/hierarchical) can provide rapid decision-making, which leads to improved SLA violation performance compared to the decentralized and

centralized approaches. In contrast, on low stress workloads (EPA/Clarknet) the decentralized approach with more nodes switched on can achieve lower SLA violation, trading this with significant energy consumption. The hybrid approach can achieve a balanced SLA violation performance and energy consumption tradeoff.

5.7.4 | Dynamic arrival

Up to this point, we have evaluated the approaches using a steady arrival rate. To understand the impact of rapid and sharp arrival rates, we examine two scenarios, as shown in Figure 14. In the first scenario (A), there is a small and steady flow of new VM creation requests, followed by a single sharp increase in the number of VM creation requests that persists for several hours, followed by a return to the previous small steady number of VM creation requests. The second scenario (B), is similar to the first scenario but with two sharp increases in the arrival rate of new VM creations. Each application runs for 10 h before shutting down. We use the P3 MA and Google 3 trace. The experiment simulates 24 h of elapsed time, and we use the base configuration in Table 3.

The high arrival rate of VMs on the infrastructure, as each VM runs the Google 3 workload, increases the VMs that begin to experience a stressed state where they do not deliver the requested CPU demand, and thus enter SLA violation. The results are shown in Figure 15, and show that the distributed approaches (decentralized, hybrid, and hierarchical) outperform the centralized approach. The hybrid approach has a rapid decision-making process and was able to achieve the lowest SLA violations compared to the other approaches. The hierarchical approach and decentralized approach had comparable performance. The disadvantage of the broadcasting mechanism was countered by nodes being switched on in the decentralized approach. In the two spike scenario, the decentralized lagged in the first spike but was able to switch on additional nodes to outperform the hierarchical approach on the second spike. It achieved this at the cost of energy consumption, utilizing 42%, 55%, and 72% more energy compared to the hybrid, hierarchical, and centralized, respectively.

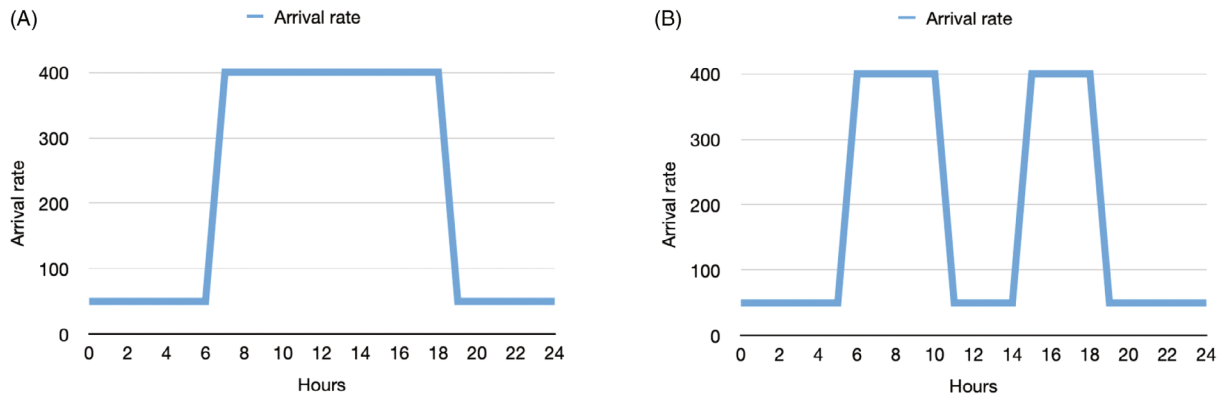

FIGURE 14 Arrival rate patterns

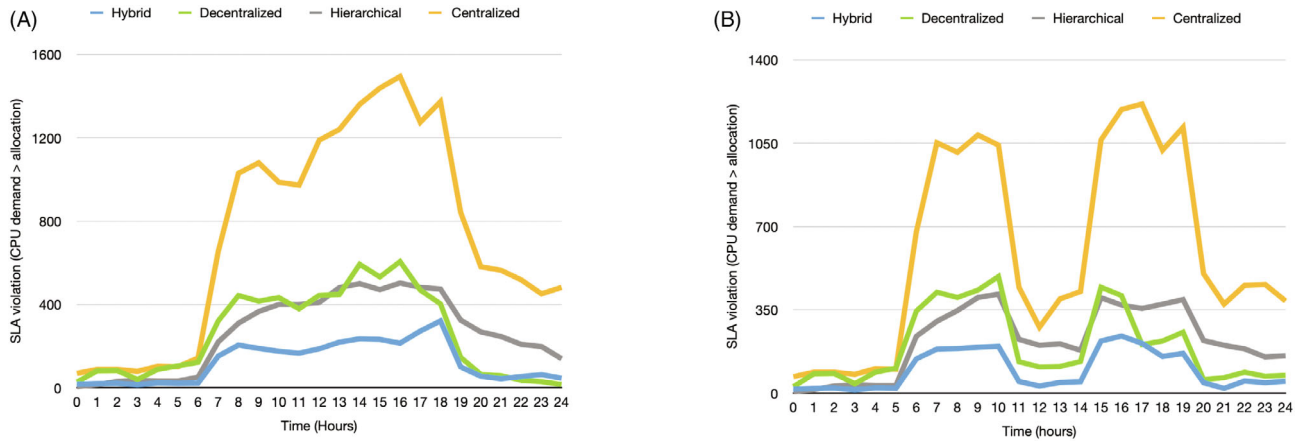

FIGURE 15 High spike in arrival rate: (A) 1 spike; (B) 2 spikes

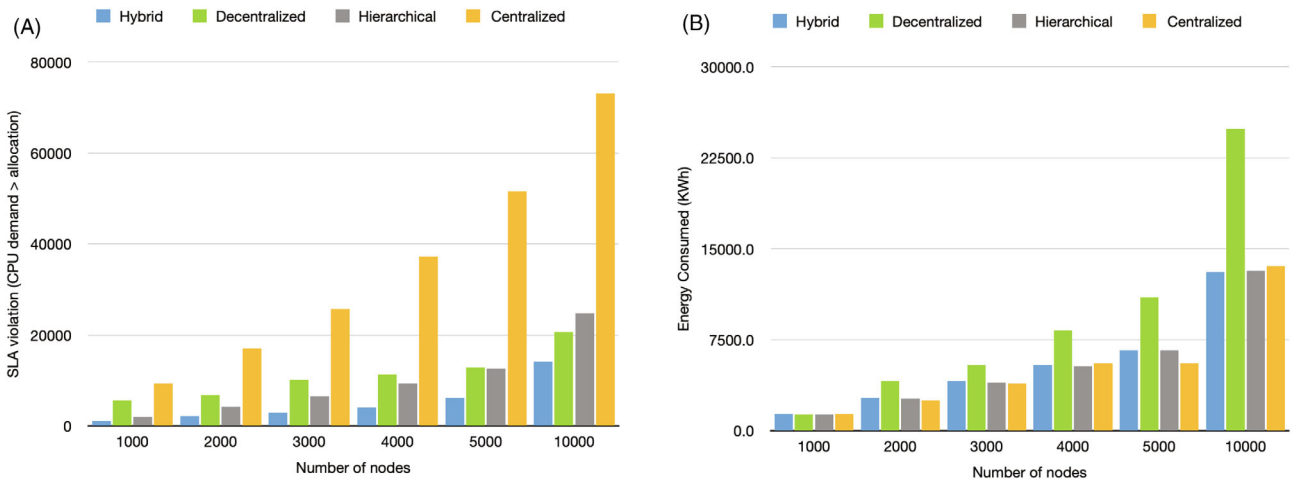

FIGURE 16 Scalability evaluation: (A) Service level agreement violations; (B) energy consumption

5.7.5 | Scalability

To evaluate how the approaches scale, we use the most stressful workload, Google 3, and maintain the stress ratio for each node, by increasing the load and the number of nodes in the data center. Similar to other experiments, we simulate 24 h, and we use the P3 MA; the results are shown in Figure 16.

The hybrid approach maintains its SLA performance as more nodes and VMs are added to the data center, with lower SLA violations than all other approaches. The hierarchical approach initially outperforms the decentralized in SLA violations, but the latter catches up at 5000 nodes. The decentralized trades its SLA violation performance for energy consumption, using 65.1% more energy compared to the hierarchical approach (5000 nodes). As expected, the distributed approaches outperform the centralized approach.

6 | CONCLUSIONS

In this paper, we have presented a scalable architecture for managing cloud resources, that combines the benefits of both the hierarchical and decentralized approaches in a hybrid manner. Our escalation approach attempts to service resource requests at the lowest local level possible, to reduce the overhead of servicing the request. We demonstrated empirically that our proposed architecture achieves significantly improved QOS metrics compared to centralized, hierarchical and decentralized architectures, particularly in high stress workloads.

In the hybrid architecture, nodes are autonomous and decide when to accept migration requests, and are typically less stressed compared to nodes in the evaluated architectures. This results in lower VM migration instability and enables more opportunities for VM consolidations. We have shown these factors lead to lower SLA violations and less migration traffic when combined with a variety of MAs from the literature, and the hybrid architecture retains the benefits of the hierarchical and decentralized approaches.

In the future, we plan to continue investigating resource allocation decision-making, by implementing and incorporating a MA that can take further advantage of the specific feature features of the hybrid architecture.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

DATA AVAILABILITY STATEMENT

The data traces used as input for this paper are part of the DCSim[45] simulator, and are available from Github: https://github.com/digs-uwo/dcsim/tree/master/traces. Most of the data generated by this research is contained in the results section of this paper, and the full datasets generated are available from the corresponding author on request.

ORCID

Abdul R. Hummaida https://orcid.org/0000-0002-3007-2289

REFERENCES

1. Hameed A, Khoshkbarforoushha A, Ranjan R, et al. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*. 2016;98(7):751-774. doi:10.1007/s00607‐014‐0407‐8
2. Herbst NR, Kounev S, Reussner R. Elasticity in cloud computing: what it is, and what it is not. Proceedings of the 10th International Conference on Autonomic Computing; 2013:23-27.
3. Jiao L, Tulino AM, Llorca J, Jin Y, Sala A. Smoothed online resource allocation in multi-tier distributed cloud networks. *IEEE/ACM Trans Netw*. 2017;25(4):2556-2570. doi:10.1109/TNET.2017.2707142
4. Ruprecht A, Jones D, Shiraev D, et al. VM live migration at scale. *SIGPLAN Not*. 2018;53(3):45-56. doi:10.1145/3296975.3186415
5. Hummaida AR, Paton NW, Sakellariou R. Adaptation in cloud resource configuration: a survey. *J Cloud Comput*. 2016;5(1):1-16.
6. Zhang F, Tang X, Li X, Khan SU, Li Z. Quantifying cloud elasticity with container-based autoscaling. *Future Gener Comput Syst*. 2019;98:672-681. doi:10.1016/j.future.2018.09.009
7. Tan B, Ma H, Mei Y. Novel genetic algorithm with dual chromosome representation for resource allocation in container-based clouds. Proceedings of the IEEE 12th International Conference on Cloud Computing (CLOUD); 2019:452-456.
8. Hu Y, Zhou H, de Laat C, Zhao Z. Concurrent container scheduling on heterogeneous clusters with multi-resource constraints. *Future Gener Comput Syst*. 2020;102:562-573. doi:10.1016/j.future.2019.08.025
9. Zhang Y. *Virtualization and Cloud Computingch*. Vol 2. John Wiley & Sons, Ltd; 2018:13-36.
10. Hummaida AR, Paton NW, Sakellariou R. SHDF - a scalable hierarchical distributed framework for data centre management. Proceedings of the 16th International Symposium on Parallel and Distributed Computing (ISPDC); 2017:102-111.
11. Islam S, Lee K, Fekete A, Liu A. How a consumer can measure elasticity for cloud platforms. Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. Association for Computing Machinery ICPE'12; 2012:85-96; New York, NY.
12. Lehrig S, Eikerling H, Becker S. Scalability, elasticity, and efficiency in cloud computing: a systematic literature review of definitions and metrics. Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures. Association for Computing Machinery QoSA'15; 2015:83-92; New York, NY.
13. Ghahramani MH, Zhou M, Hon CT. Toward cloud computing QoS architecture: analysis of cloud systems and cloud services. *IEEE/CAA J Automat Sin*. 2017;4(1):6-18.
14. Hermenier F, Lorca X, Menaud JM, Muller G, Lawall JL. Entropy: a consolidation manager for clusters; 2009:41-50; ACM, Washington, DC.
15. Zolfaghari R, Sahafi A, Rahmani AM, Rezaei R. Application of virtual machine consolidation in cloud computing systems. *Sustain Comput Inform Syst*. 2021;30:100524. doi:10.1016/j.suscom.2021.100524
16. Quesnel F, Lèbre A, Südholt M. Cooperative and reactive scheduling in large-scale virtualized platforms with DVMS. *Concurr Comput Pract Exp*. 2013;25(12):1643-1655. doi:10.1002/cpe.2848
17. Jung G, Hiltunen MA, Joshi KR, Schlichting RD, Pu C. Mistral: dynamically managing power, performance, and adaptation cost in cloud infrastructures. Proceedings of the International Conference on Distributed Computing Systems; 2010:62-73; IEEE, Washington, DC.
18. Rahmanian AA, Horri A, Dastghaibyfard G. Toward a hierarchical and architecture-based virtual machine allocation in cloud data centers. *Int J Commun Syst*. 2018;31(4):e3490. doi:10.1002/dac.3490
19. Costache S, Dib D, Parlavantzas N, Morin C. Resource management in cloud platform as a service systems: analysis and opportunities. *J Syst Softw*. 2017;132:98-118. doi:10.1016/j.jss.2017.05.035
20. Addis B, Ardagna D, Panicucci B, Squillante MS, Zhang L. A hierarchical approach for the resource management of very large cloud platforms. *IEEE Trans Depend Sec Comput*. 2013;10:253-272.
21. Zhu X, Young D, Watson BJ, et al. 1000 Islands: integrated capacity and workload management for the next generation data center. Proceedings of the International Conference on Autonomic Computing; 2008:172-181; IEEE, Washington, DC.
22. Almeida J, Almeida V, Ardagna D, Cunha Í, Francalanci C, Trubian M. Joint admission control and resource allocation in virtualized servers. *J Parallel Distrib Comput*. 2010;70:344-362.
23. Moens H, Famaey J, Latre S, Dhoedt B, Turck FD. Design and evaluation of a hierarchical application placement algorithm in large scale clouds. Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management; 2011:137-144.
24. Leontiou N, Dechouniotis D, Denazis S, Papavassiliou S. A hierarchical control framework of load balancing and resource allocation of cloud computing services. *Comput Electr Eng*. 2018;67:235-251. doi:10.1016/j.compeleceng.2018.03.035
25. Goudarzi H, Pedram M. Hierarchical SLA-driven resource management for peak power-aware and energy-efficient operation of a cloud datacenter. *IEEE Trans Cloud Comput*. 2016;4(2):222-236.
26. Moens H, Turck FD. A scalable approach for structuring large-scale hierarchical cloud management systems. Proceedings of the 9th International Conference on Network and Service Management (CNSM); 2013:1-8.
27. Keller G, Tighe M, Lutfiyya H, Bauer M. A hierarchical, topology-aware approach to dynamic data centre management. Proceedings of the Network Operations and Management Symposium (NOMS); 2014:1-7.
28. Van HN, Tran FD, Menaud JM. SLA-aware virtual resource management for cloud infrastructures. Proceedings of the IEEE International Conference on Computer and Information Technology; 2009:357-362; IEEE, Washington, DC.
29. Mola O, Bauer M. Towards cloud management by autonomic manager collaboration. *Int J Commun Netw Syst Sci*. 2011;4(12A):790-802.
30. Hindman B, Konwinski A, Zaharia M, et al. Mesos: a platform for fine-grained resource sharing in the data center. Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation NSDI'11; 2011:295-308; USENIX Association.
31. Wuhib F, Stadler R, Spreitzer M. Dynamic resource allocation with management objectives: implementation for an OpenStack cloud. *IEEE Trans Netw Serv Manag*. 2012;9(2):213-225.
32. Sedaghat M, Hernández-Rodriguez F, Elmroth E, Girdzijauskas S. Divide the task, multiply the outcome: cooperative VM consolidation. Proceedings of the IEEE International Conference on Cloud Computing Technology and Science; 2014:300-305; IEEE, Washington, DC.
33. Calcavecchia NM, Caprarescu BA, Di Nitto E, Dubois DJ, Petcu D. DEPAS: a decentralized probabilistic algorithm for auto-scaling. *Computing*. 2012;94(8):701-730.
34. Pantazoglou M, Tzortzakis G, Delis A. Decentralized and energy-efficient workload management in enterprise clouds. *IEEE Trans Cloud Comput*. 2016;4(2):196-209.

35. Tighe M, Keller G, Bauer M, Lutfiyya H. A distributed approach to dynamic VM management. Proceedings of the 9th International Conference on Network and Service Management; 2013:166-170.
36. Loreti D, Ciampolini A. A decentralized approach for virtual infrastructure management in cloud. *Int J Adv Intell Syst*. 2014;7(3/4):507-518.
37. Nouri SMR, Li H, Venugopal S, Guo W, He M, Tian W. Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications. *Future Gener Comput Syst*. 2019;94:765-780. doi:10.1016/j.future.2018.11.049
38. Barrett E, Howley E, Duggan J. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurr Comput Pract Exp*. 2013;25(12):1656-1674. doi:10.1002/cpe.2864
39. Maurer M, Brandic I, Sakellariou R. Adaptive resource configuration for cloud infrastructure management. *Future Gener Comput Syst*. 2013;29(2):472-487.
40. Aldhalaan A, Menascé DA. Autonomic allocation of communicating virtual machines in hierarchical cloud data centers. Proceedings of the 2014 International Conference on Cloud and Autonomic Computing (ICCAC); 2014:161-171.
41. Matos M, Sousa A, Pereira J, Oliveira R, Deliot E, Murray P. *CLON: Overlay Networks and Gossip Protocols for Cloud Environments*. Springer; 2009:549-566.
42. Ganesh AJ, Kermarrec AM, Massoulié L. HiScamp: self-organizing hierarchical membership protocol. Proceedings of the 10th Workshop on ACM SIGOPS European Workshop EW 10; 2002:133-139; ACM, New York, NY.
43. Birman K. The promise, and limitations, of gossip protocols. *SIGOPS Oper Syst Rev*. 2007;41(5):8-13. doi:10.1145/1317379.1317382
44. Wuhib F, Yanggratoke R, Stadler R. Allocating compute and network resources under management objectives in large-scale clouds. *J Netw Syst Manag*. 2015;23(1):111-136.
45. Tighe M, Keller G, Bauer M, Lutfiyya H. DCSim: a data centre simulation tool for evaluating dynamic virtualized resource management. Proceedings of the Network and service management (CNSM), 2012 8th International Conference and 2012 Workshop on Systems Virtualization Management (SVM); 2012:385-392.
46. Zhang F, Liu G, Fu X, Yahyapour R. A survey on virtual machine migration: challenges, techniques, and open issues. *IEEE Commun Surv Tutor*. 2018;20(2):1206-1243. doi:10.1109/COMST.2018.2794881
47. Standard performance evaluation corporation; 2022. http://www.spec.org
48. Mann ZÁ, Szabó M. Which is the best algorithm for virtual machine placement optimization? *Concurr Comput Pract Exp*. 2017;29(10):e4083. doi:10.1002/cpe.4083
49. de Lago DG, Madeira ERM, Bittencourt LF. Power-aware virtual machine scheduling on clouds using active cooling control and DVFS. Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science. Association for Computing Machinery MGC'11; 2011; New York, NY.
50. Guazzone M, Anglano C, Canonico M. Exploiting VM migration for the automated power and performance management of green cloud computing systems. Proceedings of the 1st International Workshop on Energy Ecient Data Centers; 2012:81-92.
51. Calcavecchia NM, Biran O, Hadad E, Moatti Y. VM placement strategies for cloud scenarios. Proceedings of the IEEE 5th International Conference on Cloud Computing; 2012:852-859; IEEE.
52. Shi L, Furlong J, Wang R. Empirical evaluation of vector bin packing algorithms for energy efficient data centers. Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC); 2013:000009-000015.
53. Chowdhury MR, Mahmud MR, Rahman RM. Study and performance analysis of various VM placement strategies. Proceedings of the IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD); 2015:1-6.
54. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr Comput Pract Exp*. 2012;24:1397-1420.
55. HPE ProLiant; 2016.
56. VMware; 2016. http://www.vmware.com.
57. Citrix Xen; 2016. http://www.xenserver.org.
58. Zhang W, Han S, He H, Chen H. Network-aware virtual machine migration in an overcommitted cloud. *Future Gener Comput Syst*. 2017;76:428-442. doi:10.1016/j.future.2016.03.009
59. Sniezynski B, Nawrocki P, Wilk M, Jarzab M, Zielinski K. VM reservation plan adaptation using machine learning in cloud computing. *J Grid Comput*. 2019;17(4):797-812. doi:10.1007/s10723‐019‐09487‐x
60. Tso FP, Hamilton G, Oikonomou K, Pezaros DP. Implementing scalable, network-aware virtual machine migration for cloud data centers. Proceedings of the IEEE 6th International Conference on Cloud Computing; 2013:557-564.

How to cite this article: Hummaida AR, Paton NW, Sakellariou R. A hierarchical decentralized architecture to enable adaptive scalable virtual machine migration. *Concurrency Computat Pract Exper*. 2023;35(2):e7487. doi: 10.1002/cpe.7487