RISA: Round-Robin Intra-Rack Friendly Scheduling Algorithm for Disaggregated Datacenters

Rashadul Kabir
Colorado State University
Fort Collins, CO, USA
rashadul.kabir@colostate.edu

Ryan G. Kim
Intel Labs
Hillsboro, OR, USA
ryan.gary.kim@intel.com

Mahdi Nikdast
Colorado State University
Fort Collins, CO, USA
mahdi.nikdast@colostate.edu

ABSTRACT

Recent trends see a move away from a fixed-resource server-centric datacenter model to a more adaptable "disaggregated" datacenter model. These disaggregated datacenters can then dynamically group resources to the specific requirements of an incoming workload, thereby improving efficiency. To properly utilize these disaggregated datacenters, workload allocation techniques must examine the current state of the datacenter and choose resources that not only optimize the current workload request, but future ones. Since disaggregated datacenters are severely bottlenecked by the available network resources, our work proposes a heuristic-based approach called RISA, which significantly reduces the network usage of workload allocations in disaggregated datacenters. Compared to the state-of-the-art, RISA reduces the power consumption for optical components by 33% and reduces the average CPU-RAM round-trip latency by 50%. Additionally, RISA significantly outperforms the state-of-the-art in terms of execution time.

CCS CONCEPTS

• **Hardware** → **Enterprise level and data centers power issues**; • **Networks** → *Data center networks*; • **Information systems** → Data centers.

KEYWORDS

Disaggregated/composable data centers, Network-aware scheduling, energy-aware scheduling, load balancing

ACM Reference Format:
Rashadul Kabir, Ryan G. Kim, and Mahdi Nikdast. 2023. RISA: Round-Robin Intra-Rack Friendly Scheduling Algorithm for Disaggregated Datacenters. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3624062.3624228

1 INTRODUCTION

Traditionally, datacenters (DCs) have been server-centric, characterized by network-connected homogeneous servers with fixed ratios of compute, memory, and storage. Modern cloud workloads, however, demand diverse resource ratios, leading to inefficiencies and significant amounts of stranded resources. These unused stranded resources not only increase the capital costs but also amplify power consumption, costing up to 85% of total DC expenses [2]. Furthermore, although the life cycle and technological advancements of various server resources differ, this fixed integration paradigm requires any hardware upgrade or resource expansion to be executed at the server level [7]. Figure 1 shows how a disaggregated DC (DDC) is different from a traditional datacenter.

In terms of supporting the implementation of DDC systems, researchers have proposed an alternative to the standard network interface card, known as a switch and interface card (SIC). SIC is able to perform the packet and circuit switching services required in a DDC [19]. Researchers have also investigated operating systems focusing on resource disaggregation [18]. Beyond the academic realm, the industry too has shown a keen interest, with notable contributions including the Intel Rack Scale Architecture [9], dReDBox project [3], and Firebox [1]. Throughout these advancements, both in academia and industry, the central aim has been to maximize resource utilization.

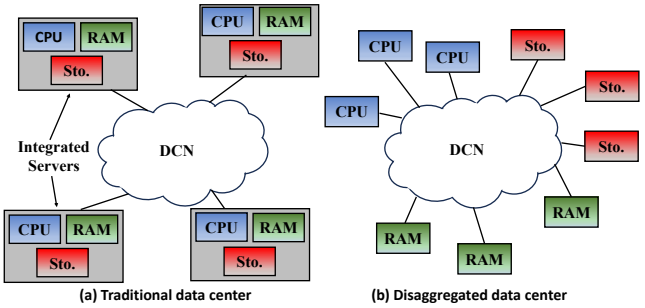


Figure 1: Disaggregated vs. Traditional

Despite the enthusiasm surrounding resource disaggregation, it is challenging to design the prerequisite network infrastructure. In order for disaggregated datacenters (DDCs) to be desirable, the network should achieve similar latency and bandwidths to their traditional direct-attached counterparts while keeping costs and power consumption low. Furthermore, a coordinated orchestration of compute, memory, and network resources is essential to maximize resource utilization and workload performance, while simultaneously keeping both latency and costs low. Thus, compute and network scheduling becomes an integral part of research in resource disaggregation.

One of the seminal efforts in disaggregated resource scheduling was done by Zervas *et al.* [20]. In [20], the authors propose two algorithms for scheduling virtual machines (VMs) onto disaggregated

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11.
https://doi.org/10.1145/3624062.3624228

CPU, RAM, and storage nodes to maximize resource utilization. The first is a network-unaware locality-based (NULB) heuristic-based scheduling algorithm that uses a breadth-first search (BFS) to choose resources. The second algorithm is a network-aware locality-based (NALB) scheduling algorithm that extends NULB to also consider network utilization. NALB chooses to use links with higher available bandwidth; thus, tries to ensure that VMs are not dropped because of unavailable link bandwidth. Later, in Section 4, we will demonstrate that NULB and NALB suboptimally utilize network resources. In particular, they utilize more inter-rack network resources, which motivates the need for an approach that focuses on optimally utilizing both the compute and network resources. Thus, in this paper, we propose a novel heuristic-based algorithm called Round-robin Intra-rack friendly Scheduling Algorithm (RISA), which is able to schedule workloads onto disaggregated CPU, RAM, and storage nodes while attempting to maximize resource utilization and minimize network utilization and CPU-RAM latency.

The rest of the paper is organized as follows. Section 2, discusses some developments in resource disaggregation and workload scheduling in DDC architectures. Section 3 discusses the DDC architecture used in this paper and the optical switch energy model. Section 4 details our proposed approaches RISA and RISA-BF, including deficiencies in prior work. Section 5, presents our simulation results and comparative analysis. Finally, concluding remarks are given in Section 6.

2 RELATED WORK

Over the years, there have been efforts to separate or disaggregate server resources. Significant advancements have been achieved through the use of Storage Area Networks (SANs) and Network-Attached Storage (NAS) systems, both of which offer storage solutions over a network [20]. In 2009, memory disaggregation was introduced to address memory capacity challenges [11]. Following a short period of diminished activity, the field of resource disaggregation experienced a resurgence in 2016. The first work on workload scheduling in DDCs was by Papaioannou *et al.* [16]. They proposed a heuristic for scheduling VMs onto rack-scale DDC, showcasing how resource utilization can be better compared to traditional DC scheduling techniques. They considered only CPU and RAM in their scheduling problem. Another critical difference is the consideration of inter-VM communication, their approach additionally attempted to schedule VMs that communicate with one another closer. However, for DCs that primarily service third-party workloads, VMs may function independently. One such example is the Azure data center traces [5]. Ali *et al.* [14] proposed an MILP-based energy-aware scheduling approach at the DC scale. They considered CPU, RAM, and IO resources. Next, Zervas *et al.* [20] proposed the NULB and NALB algorithms. These algorithms consider CPU, RAM, and storage for the scheduling problem. The primary focus of this paper was to minimize the number of dropped VMs and maximize resource utilization. However, the manner in which compute resource search was prioritized in these algorithms, it encouraged inter-rack VM assignments. One good aspect of [20], was the DDC architecture used in the paper. It was heavily inspired by the DDC architecture developed by IBM, dRedBox [3]. As a follow-up to [20], [17] proposed a reinforcement learning-based algorithm, focusing on reduced network usage. Although [17] significantly outperformed NULB and NALB [20], the problem definition is different from that in our work. First, the datacenter network in [17] uses a three-tier tree network structure while our work uses only two tiers. The three-tier structure scheduling problem requires consideration of intra-rack and inter-rack within the same sub-tree, and inter-rack among different sub-tree resource relationships. Our focus is to schedule resources within a cluster (intra-rack and inter-rack), limiting the maximum number of hops. Second, [17] allows VMs that require more than one box of the same resource type. In our work, the VM resource requirements are always smaller than the capacity of one resource box. Third, storage is not considered in [17]. These last two points significantly change the nature of the VM requests and especially the network considerations needed during scheduling. Additionally, we focus on developing a deterministic heuristic to schedule compute and network resources in a disaggregated datacenter. Since [17] is a non-deterministic machine learning-based approach, it is beyond the scope of this work. Other researchers [4, 8] have approached the DDC workload scheduling problem from different angles. However, none of them considered all three resource types (CPU, RAM, and storage) for the scheduling problem, while utilizing an industry-standard DDC architecture.

3 BACKGROUND

3.1 Disaggregated architecture

The disaggregated datacenter architecture proposed in [20] is used as the case study in this work (see Figure 3). This architecture is structured into racks. Within each rack, there are several boxes, each box has some amount of a single resource type: CPU, RAM, or storage. These boxes are further divided into bricks, with each brick holding a predetermined quantity of its designated resource. Each box is equipped with optical switches, connected to rack-level switches, for communication. Progressing up the hierarchy, rack switches then connect to inter-rack switches. As seen in Figure 2, if a CPU brick within a CPU box in rack 0 intends to interact with a RAM brick within a RAM box in rack 1, the communication journey would entail traversing the box switch of rack 0, its rack switch, the inter-rack switch, the switch of rack 1, and finally, the RAM box's switch before reaching its destination.

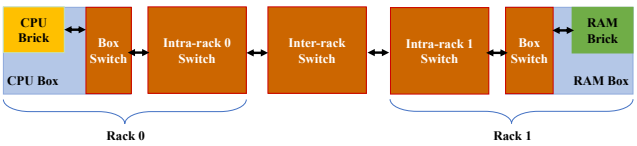


Figure 2: Inter-rack communication in the disaggregated architecture proposed in [20]

Table 1 offers a detailed configuration utilized in our simulation study. Notably, within a brick, all communication is electronic. Once the data leaves the brick, it gets converted from electronic to photonic by a single-mode Luxtera commercial SiP optical module [12] (depicted in Figure 3). This module uses eight spatially

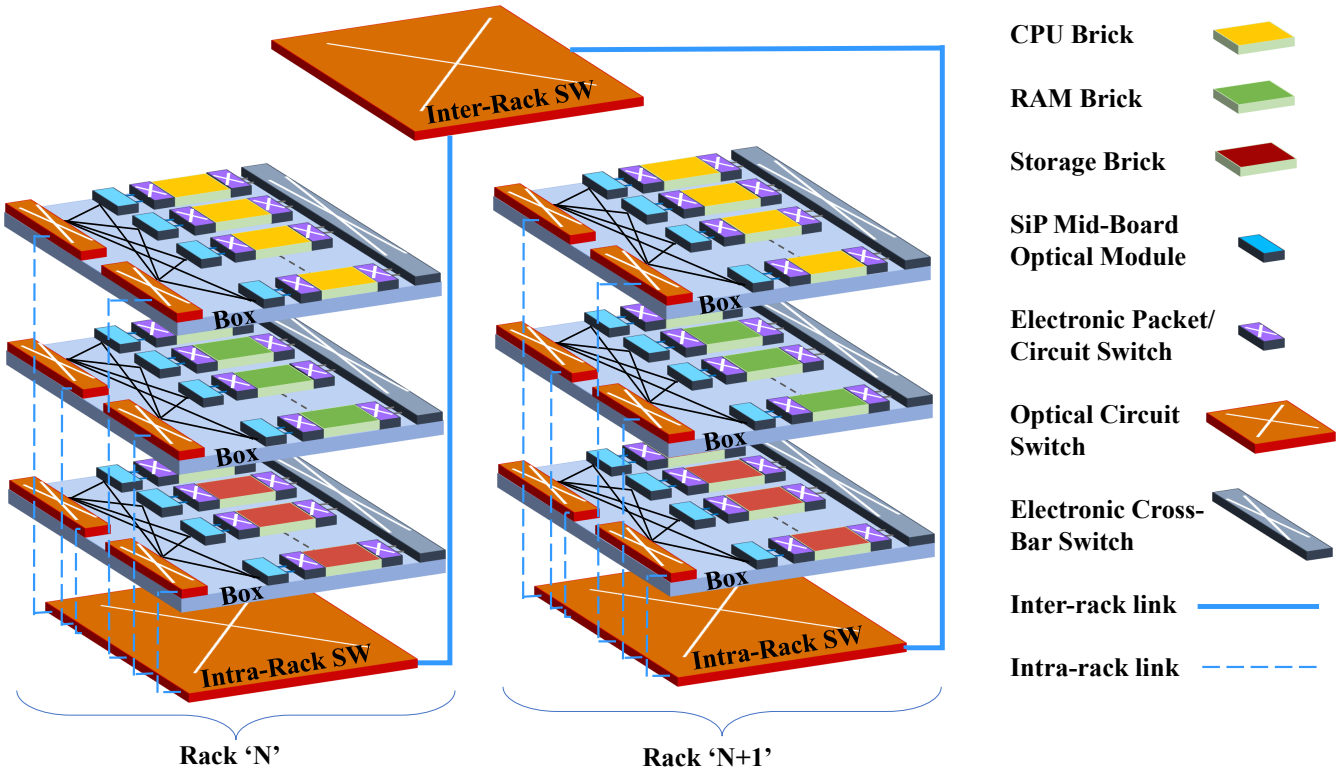


Figure 3: The disaggregated datacenter architecture proposed in [20] used as the case study in this work

multiplexed optical channels, each capable of supporting a 25Gb/s bit rate, for a total of 200Gb/s per link. Utilizing transceivers that operate in single-mode, rather than multimode, enables a more scalable network built on optical circuit switching (OCS). This is because single-mode fibers accommodate optical switches with a greater number of ports [15]. We obtain the power consumption of the transceiver module to be 22.5 pJ/bit from [20]. Using Tables 1 and 2 [20], we can get a sense of network requirements for different sizes of VMs.

Table 1: Disaggregated architecture configuration

DDC Configuration		Brick size	16 units
Cluster size	18 racks	CPU unit	4 cores
Rack size	6 boxes	RAM unit	4 GB
Box size	8 bricks	Storage unit	64 GB

Table 2: Network requirements

CPU-RAM bandwidth	5 Gb/s/unit
RAM-STO bandwidth	1 Gb/s/unit

3.2 Optical switch energy model

To accurately model the optical switch power, we need to consider the states of the cells within a switch. Known for their fast reconfiguration times, we consider microring resonator (MRR)-based switches for our disaggregated setup. For the switch configuration, we selected a commonly used Beneš network configuration.

Within an optical switch, ports are interconnected through a network of cells. Each cell in the switch can be in either "cross" or "bar" state. When a VM requests a path through a switch, some cells must change their states. For this, each of these cells consumes switching power (P_{swcell}). Thus, if there are n number of cells along a switch path, we assume that $n/2$ of the cells will undergo reconfiguration. For the rest of the time, to maintain a cell's state it consumes trimming power ($P_{trimcell}$). Therefore, the optical switch energy consumed for each VM (E_{sw}) is:

$$E_{sw} = (\frac{n}{2} \times P_{swcell} \times lat_{sw}) + (\alpha \times n \times P_{trimcell} \times T) \quad (1)$$

where lat_{sw} is the latency of switching the cells (dependent on the switch size [6]), α is a constant to consider the fact that two VMs can share the same cell, and T is the lifetime of the VM.

Based on the findings in [13], we considered $P_{trimcell} = 22.67mW$ and $P_{swcell} = 13.75mW$. The number of cells in a Beneš-based optical switch is dependent on the number of ports on the switch, as detailed in [10]. Similarly, the latency of switching the cells are also based on the switch size [6]. The switch configurations are detailed

in the experimental results in Section 5. Since two VMs can share the same switch cell α should be between 0.5 (every cell is shared) and 1 (no cell is shared). For the purpose of our simulations, we've chosen the value of α to be 0.9.

Figure 4 shows a generic view of an optical switch, represented by the largest box. The boxes inside the switch with solid outlines represent the cells that are being utilized (dashed are inactive). In the figure, we show two paths, $P1$ and $P2$, corresponding to two VMs that share an MRR cell.

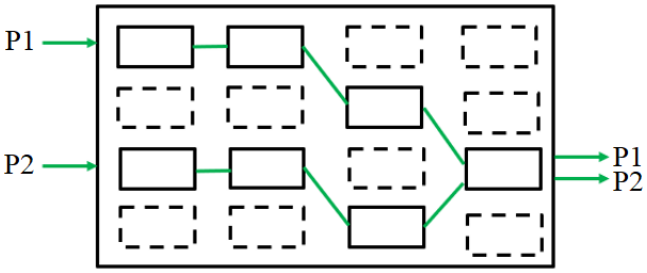


Figure 4: Generic view of an optical switch

4 RISA OVERVIEW

4.1 Discussion of NULB and NALB

Before discussing RISA in detail, it's crucial to first grasp the workings and limitations of NULB and NALB. Both of these algorithms have a compute (CPU/RAM/storage) resource allocation phase and a network resource allocation phase.

During the compute resource allocation phase, [20] prioritizes the most scarce resource by examining the contention ratio (CR) or the amount of a resource required by a VM over the total amount of that available resource. NULB and NALB both first search for a suitable box to satisfy the resource request with the highest CR. Then, NULB searches for the other resources using breadth-first search (BFS), while NALB uses a modified BFS. In modified BFS, NALB reorders neighbors of the scarce resource in descending order of their available bandwidth. In the network resource allocation phase, NULB selects the first available link to establish the connection between each pair of resources. NALB chooses links with the most available bandwidth.

The way the compute resource search is prioritized in NULB or NALB, it encourages inter-rack network utilization. Thus, even though the network allocation phase of NALB is network-aware, it does not truly discourage higher network utilization.

4.2 Discussion of RISA

To better assign VMs to intra-rack resources, RISA keeps track of the boxes with the maximum amount of each resource for each rack. Thus, when a VM requests resources, we can easily determine which racks have boxes that can accommodate the entire VM. These racks that can house the entire VM are placed in a list called *INTRA_RACK_POOL*. To help balance the load between racks, we adopt a round-robin policy for selecting racks from *INTRA_RACK_POOL*. This helps to make the utilization of the racks more uniform. Algorithm 1 details the different steps of RISA. *REQ* indicates the VM compute requirements. *NET* indicates the total network bandwidth available. *AVAIL_INTRA_RACK_NET* indicates the total intra-rack network bandwidth available.

If *INTRA_RACK_POOL* is empty or available intra-rack network resources is insufficient to schedule a VM using *INTRA_RACK_POOL*, we create three lists, known as the *SUPER_RACK* collectively. Within the *SUPER_RACK*, each list contains the racks with boxes with sufficient resources (CPU or RAM or storage) for a VM assignment. In such a scenario, the VM assignment has to be inter-rack. In this case, RISA resorts to NULB [20] (Algorithm 2) to perform the VM allocation. The choice boxes for VM assignment are then limited to the boxes of racks within the SUPER_RACK. Also, VM assignment starts with first finding the most contended resource, then based on BFS, other resources are first searched for in the same rack and then searched for in other racks.

Algorithm 1 RISA

```
while ∃ unscheduled VM do
  Create INTRA_RACK_POOL
  if INTRA_RACK_POOL ≠ ∅ then
    for all rack ∈ INTRA_RACK_POOL (round-robin) do
      if AVAIL_INTRA_RACK_NET ≠ ∅ then
        Compute ← AllocCom(INTRA_RACK_POOL, REQ)
        Network ← AllocNet(AVAIL_INTRA_RACK_NET)

        if (Compute ≠ ∅)and(Network ≠ ∅) then
          Assign VM
        end if
      end if
    end for
  else
    Create SUPER_RACK
    Compute, Network ← NULB(SUPER_RACK, REQ, NET)
    if (Compute ≠ ∅) and (Network ≠ ∅) then
      Assign VM
    end if
  end if
end while
```

Algorithm 2 outlines the different steps of NULB. *res_type* indicates all kinds of resources (e.g., CPU, RAM, and storage). After finding the most scarce resource, res_{max} using CR, NULB first looks for the first box with the available res_{max} requested by a VM. Next, using BFS, NALB first looks for other requested resources by VM in the same rack. If it doesn't find them in the same rack it looks for resources in other racks. Once a set of boxes with available resources is found, this indicates successful completion of the compute allocation phase. Next, NALB checks to see if the available network bandwidth is present in the connecting optical links. If it finds the links with the necessary bandwidth, the VM assignment is successful with the completion of the network allocation phase. If either the compute allocation or network allocation fails, the VM to be assigned is dropped.

Algorithm 2 NULB

```
NULB(RES, REQ, NET)
for all res_type do
  Append CR(res_type) to CR_LIST
end for
res_max ← max(CR_LIST)
if res_max is available on any box then
  Compute ← AllocCompute(BFS(res_max), REQ)
  if Compute ≠ ∅ then
    Network ← AllocNetwork(NET)
    if Network ≠ ∅ then
      Assign VM (when implementing NULB by itself)
    end if
  end if
end if
Return (Compute, Network)
```

In line with traditional VM scheduling algorithms promoting the best-fit packing, we investigate a variant of RISA, RISA-BF, prioritize boxes with lower available resources when INTRA_RACK_POOL is not empty. The main goal for RISA-BF is to better pack resources and reduce resource stranding. The resulting algorithm can be seen in Algorithm 3 and has been shown to have superior results when compared to the RISA.

Algorithm 3 RISA-BF

```
while ∃ unscheduled VM do
  Create INTRA_RACK_POOL
  if INTRA_RACK_POOL ≠ ∅ then
    Sort boxes within each rack in ascending # of resource
    for all rack ∈ INTRA_RACK_POOL (round-robin) do
      if AVAIL_INTRA_RACK_NET ≠ ∅ then
        Compute ← AllocCom(INTRA_RACK_POOL, REQ)
        Network ← AllocNet(AVAIL_INTRA_RACK_NET)

        if (Compute ≠ ∅)and(Network ≠ ∅) then
          Assign VM
        end if
      end if
    end for
  else
    Create SUPER_RACK
    Compute, Network ← NULB(SUPER_RACK, REQ, NET)
    if (Compute ≠ ∅) and (Network ≠ ∅) then
      Assign VM
    end if
  end if
end while
```

4.3 Toy examples for comparative analysis

In this section, we will see scenarios where RISA will outperform NULB and NALB. We will also see how RISA-BF may outperform RISA.

4.3.1 Toy example 1. Firstly, let's consider Scenario 1. Table 3 lists the availability of compute resources. We are considering a typical VM with the following requirements - **8 cores of CPU, 16 GB of RAM, and 128 GB of storage**. Let us assume that there are enough network resources connecting these compute resources.

Table 3: Disaggregated architecture configuration for toy examples

CPU information				
id	rack	box	capacity	avail
0	0	0	64 cores	0 cores
1	0	1	64 cores	0 cores
2	1	0	64 cores	64 cores
3	1	1	64 cores	32 cores
RAM information				
id	rack	box	capacity	avail
0	0	0	64 GB	0 GB
1	0	1	64 GB	16 GB
2	1	0	64 GB	32 GB
3	1	1	64 GB	16 GB
Storage information				
id	rack	box	capacity	avail
0	0	0	512 GB	0 GB
1	0	1	512 GB	0 GB
2	1	0	512 GB	256 GB
3	1	1	512 GB	512 GB

Here, the CR for CPU is 0.08, for RAM is 0.25, and for storage is 0.17. Hence, according to NULB or NALB, the CPU, RAM, and storage ids will be **(2, 1, 2)**. CPU and RAM need to communicate with each other. Also, RAM and storage need to communicate. In both of these cases, this will result in inter-rack network utilization. In comparison, with RISA, INTRA_RACK_POOL will now be equal to [1]. Thus, based on this, the VM will be assigned to CPU, RAM, and storage with ids **(2, 2, 2)**. In this case, there will be no additional inter-rack network utilization. Resulting in lower power utilization corresponding to this VM assignment.

4.3.2 Toy example 2. In the second example, we will once again consider the DDC state in Table 3. We will only consider the CPU requirements for subsequent VMs - **15 cores, 10 cores, 30 cores, 12 cores, 5 cores, 8 cores, 16 cores, and 4 cores**. Assuming, no previous VMs get released, let us see through Table 4 how RISA and RISA-BF will perform in this case. Considering all other compute and network resource requirements are met, based on the first-fit packing, box 0 first, and then box 1, RISA will continue to schedule VMs until id 5. It will drop the VM with id 6, due to inadequate resources and schedule VM with id 7. RISA-BF, however, by performing best-fit packing, alternating between box 0 and box 1, ends up allocating all of the subsequent VMs listed. In an actual scenario, this can either translate to a lower number of dropped VMs or fewer inter-rack VM assignments. Thereby, improving either resource utilization or network utilization.

Table 4: CPU requirement for subsequent VMs

VM id	CPU req.	RISA Rack 1 box	RISA-BF Rack 1 box
0	15	0	1
1	10	0	1
2	30	0	0
3	12	1	0
4	5	1	1
5	8	1	0
6	16	**NA**	0
7	4	1	0

5 SIMULATION RESULTS AND COMPARATIVE ANALYSIS

5.1 Performance evaluation using synthetic workload

We will now see how NULB, NALB, RISA, and RISA-BF schedule a synthetic random workload on our disaggregated architecture discussed in Section 3. We generated a random workload similar to the synthetic random workload in [20]. A VM can have a random amount of CPU cores from 1 to 32 cores and a random amount of RAM from 1 to 32 GB. Storage for every VM is 128 GB. Requests are produced dynamically based on a Poisson distribution with a mean interarrival period of 10 time units. The VM life cycle begins at 6300 time units, with an increment of 360-time units for each set of 100 requests. A total of 2500 VMs were generated.

The average CPU utilization for all algorithms was 64.66%, the average RAM utilization for algorithms was 65.11% and the average storage utilization for algorithms was 31.72%. As seen in Figure 5, despite utilizations well below 100%, both NULB and NALB had 255 inter-rack VM assignments. On the other hand, RISA and RISA-BF had only 7 and 2 inter-rack VM assignments respectively.

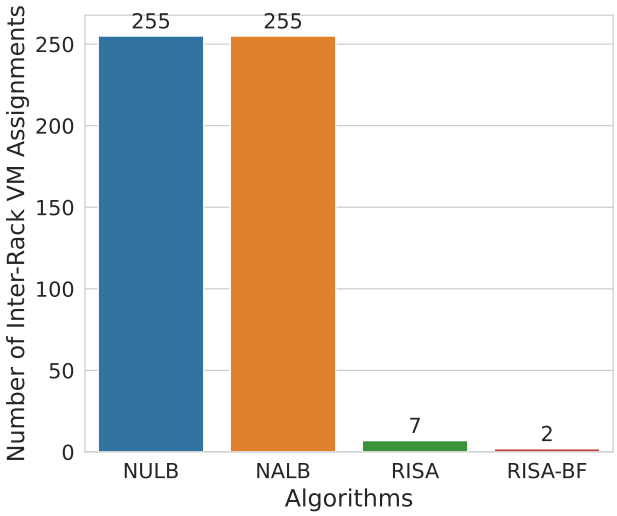


Figure 5: Number of inter-rack VM assignments

In the next subsection, we will see how this difference in inter-rack VM assignment translates into substantially lower power consumption (for optical components)for RISA and RISA-BF, in comparison to NULB and NALB, for a practical workload. Similarly, we will also see how it translates into a lower average CPU-RAM latency for RISA and RISA-BF in comparison to NULB and NALB.

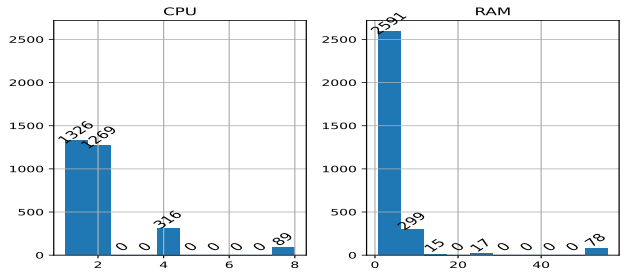
5.2 Performance evaluation using practical workload

We will now compare the performance of the algorithms in the areas of network utilization, energy consumption, and average CPU-RAM latency using the 2017 public release of Microsoft Azure data center traces [5]. To choose workloads of different characteristics, we selected different subsets of the 2017 Azure data center traces - the first 3000 VMs (Azure-3000), the first 5000 VMs (Azure-5000), and the first 7500 VMs (Azure-7500).

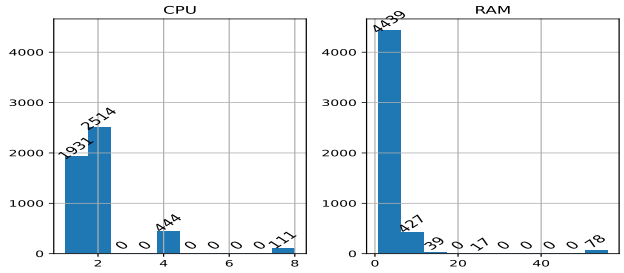
There are many ways in which the Azure data center workload is different from the random workload. Firstly, the CPU requirement is generally low. In comparison to the CPU requirement, the RAM requirement for some VMs is quite high. For this workload, we assume storage to be 128 GB, similar to [20]. Thus, in most cases, storage is the most contended resource. In Figure 6, we see the characterization of the different practical workloads in terms of their CPU and RAM requirements. From Figure 6, it is seen that Azure-3000 has the lowest percentage of small VMs compared to Azure-5000 and Azure-7500. For Azure-5000, the percentage of smaller VMs is more. Azure-7500 has the greatest percentage of small VMs. Thus, the scheduling scenarios for each of these workloads are different. The network requirement for the VMs can be obtained based on Tables 1 and 2. We will see how the RISA and RISA-BF perform in the scheduling task in comparison to NULB and NALB.

Figure 7 shows the percentage of inter-rack VM assignments (out of the total number of VMs) for the three types of workloads discussed in the previous section. Here, we can see that both NULB and NALB have significant amounts of inter-rack assignments, up to 52% and 48% for NULB and NALB, respectively. Notably, RISA and RISA-BF have no inter-rack VM assignments for any of the workloads - showing their ability to exploit intra-rack VM allocations while leaving room for future VM requests.

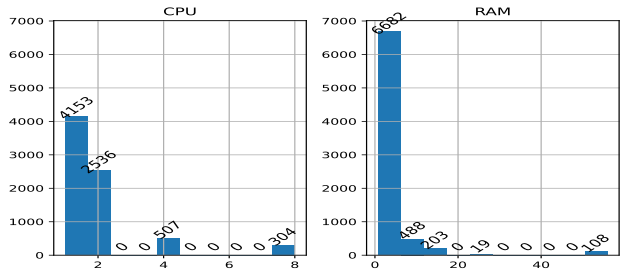
For the DDC, each box only contains one type of resource, all VMs have to use the rack switch to communicate between resources regardless of where the resources reside. Thus, when the same amount of resources is used, an equal amount of intra-rack bandwidth is used to access the rack switches. For the scheduling problems in the discussion, no VMs were dropped during the scheduling process. Thus, accordingly, as seen in Figure 8, the intra-rack network utilization of all the algorithms are the same - 30.4% for Azure-3000, 35.4% for Azure-5000, and 42.6% for Azure 7500. However, the inter-rack network utilization for all the algorithms are not the same. As explained in Section 3, RISA and RISA-BF try to minimize the amount of inter-rack VM assignments. This results in fewer inter-rack resources being used. Subsequently, resulting in lower power consumption from optical components. For the three workloads in the discussion, since there are no inter-rack VM



(a) Azure-3000



(b) Azure-5000



(c) Azure-7500

Figure 6: CPU and RAM distribution of Azure data center traces

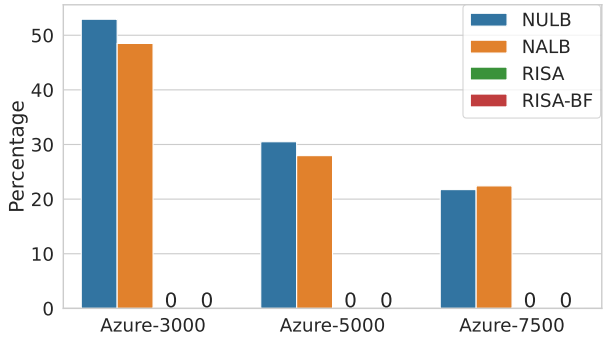


Figure 7: Percentage of inter-rack VM assignments

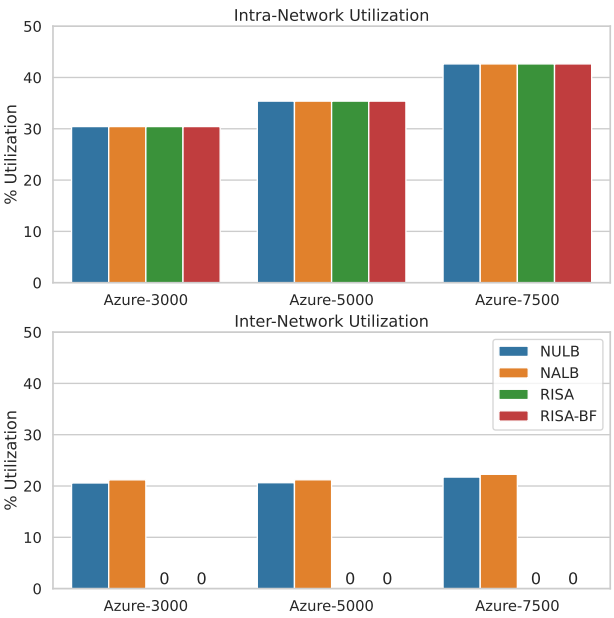


Figure 8: Network utilization

assignments for RISA and RISA and RISA-BF, it can be seen from Figure 8 that the inter-rack network utilization is also zero.

Inter-rack switches usually have much higher port numbers in comparison to rack or box switches. This is because they connect to a large number of racks within a cluster. Thus, if an algorithm requires more inter-network bandwidth, it ends up consuming a larger amount of power from optical components, particularly from the inter-rack switches. For the power consumption calculation for optical components, we considered the transceiver power and total optical switch (box switch, intra-rack switch and inter-rack switch) power. To support the DDC architecture in Section 3, we need to have box switches with 64 ports, intra-rack switches with 256 ports and inter-rack switches with 512 ports. In Figure 9, for Azure-7500, the power consumption due to optical components is seen to be as high as 6.70 kW and 6.72 kW for NULB and NALB respectively. Also, it can be seen that the same power consumption for RISA and RISA-BF can be as low as 3.36 kW for Azure-3000, whereas for NULB and NALB, the values were 5.22 kW and 5.27 kW respectively. This shows that RISA or RISA-BF each have 33% reduction in power consumption from optical components, as compared to NULB or NALB.

Quality of service (QoS) can be an important aspect when it comes to scheduling of VMs within a DC. Since data centers sometimes run third-party workloads, QoS becomes even more of a concern, which cannot change with a change in workload. Thus, the last area of focus in comparing the three algorithms is the average CPU-RAM round-trip latency. From [20], we assume that there is a 110 ns CPU-RAM round-trip latency for communication within a rack, and that across racks is 330 ns. From Figure 10, it can be seen that for Azure-3000, NULB and NALB have an average CPU-RAM round-trip latency of 226 ns and 216 ns respectively. For RISA or RISA-BF, it is only 110ns. This shows that for Azure-3000, RISA or

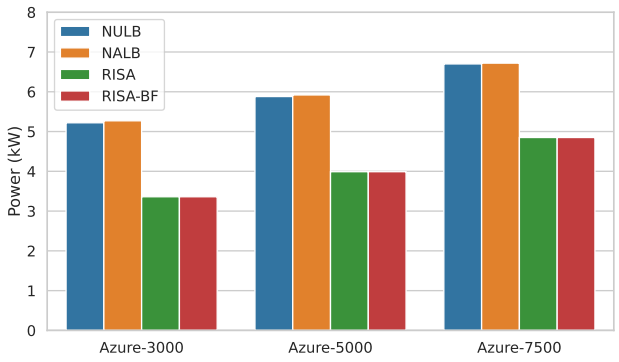


Figure 9: Power consumption for optical components

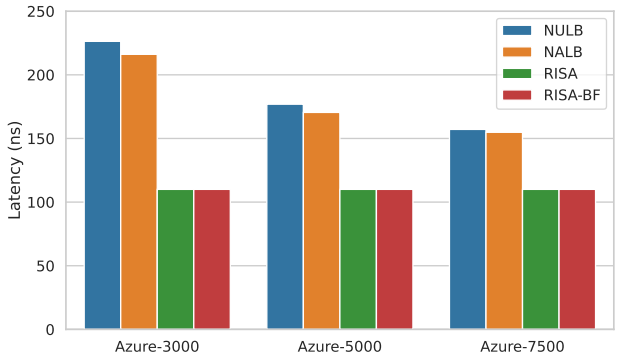


Figure 10: Average CPU-RAM delay

RISA-BF has lower than half of the average CPU-RAM round-trip latency as compared to NULB or NALB. Thus, RISA and RISA-BF are significantly better when QoS is of higher priority. For inter-rack center switches with a larger number of ports, the inter-rack delay may be higher, so the values in Figure 9 represent optimistic values for average CPU-RAM round-trip latency. However, since RISA and RISA-BF both out-perform NULB and NALB in terms of inter-rack VM allocations, we expect RISA and RISA-BF to have even larger improvements in CPU-RAM latency for larger systems.

5.3 Execution times of different algorithms

For RISA or RISA-BF, when *INTRA_RACK_POOL* is empty, it uses NULB for finding required compute and network resources for VM assignment. Thus, the time complexity for NULB, RISA and RISA-BF are the same. NALB performs an additional step of finding the path with the most available network bandwidth. Thus, the time complexity of NALB is higher compared to NULB, RISA, and RISA-BF. However, in practice, *INTRA_RACK_POOL* is not always empty. In fact for the simulation results discussed in preceding subsections, *INTRA_RACK_POOL* was never empty. Thus, in all cases, scheduling under RISA ended first. After RISA, scheduling under RISA-BF ended second, scheduling under NULB ended third, and scheduling under NALB ended last.

Table 5 lists the configuration of the system, which was used to run the simulations. Figure 11 gives a visual representation of the scheduling times of the algorithms for the synthetic workload. It can be seen that the execution time of NALB is the highest, at 865*seconds*, which is followed by NULB, at 233 seconds. RISA takes 111 seconds and RISA-BF takes 112 seconds. For RISA or RISA-BF this translates to greater than 2 × speedup when compared to NULB and close to 8 × speedup when compared to NALB. Figure 12 gives a visual representation of the scheduling times of the algorithms for the practical workload. Here, for Azure-7500, the execution time for NALB was 15929 seconds and that for NULB was 10361 seconds. For RISA and RISA-BF these values were 3679 seconds and 4013 seconds respectively. Thus, for RISA this translates to 2.81 × speedup when compared to the execution time of NULB. For RISA, the speedup when compared to the execution time of NALB is 4.33×.

Component	Specification
Processor	AMD Ryzen 7 2700X, 4.3 GHz (8 cores, 16 threads)
RAM	4×8GB DDR4, 2133 MT/s

Table 5: Simulation System Configuration

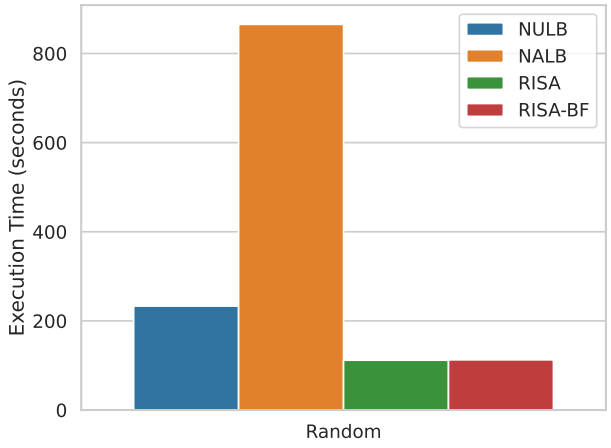


Figure 11: Execution time of synthetic workload

6 CONCLUSION

The goal of this paper was to propose an approach to significantly reduce the network utilization, power consumption, and CPU-RAM round-trip latency. We have been successful to reduce the network utilization significantly, which resulted in up to more than 33% reduction in power consumption of optical components. Compared to the state-of-the-art, our approach achieved up to 50% reduction in CPU-RAM round-trip latency. Additionally, for practical workload, we achieved up to 2.81 × speedup when compared to NULB, and up to 4.33× speedup when compared to NALB.

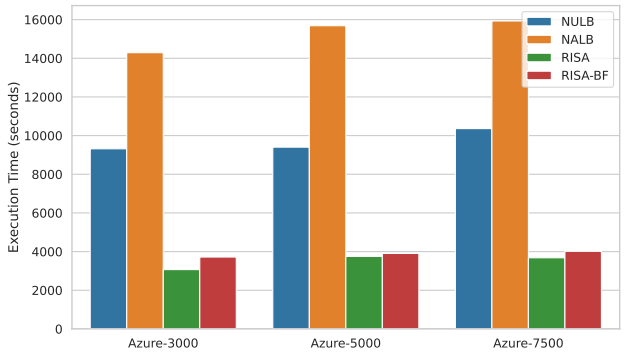


Figure 12: Execution time of practical workload

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF) under grant number CNS-2046226.

REFERENCES

[1] Krste Asanović. 2014. FireBox: A Hardware Building Block for 2020 Warehouse-Scale Computers. USENIX Association, Santa Clara, CA.

[2] Luiz André Barroso and Urs Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (2007), 33–37.

[3] M. Bielski, I. Syrigos, K. Katrinis, D. Syrivelis, A. Reale, D. Theodoropoulos, N. Alachiotis, D. Pnevmatikatos, E.H. Pap, G. Zervas, V. Mishra, A. Saljoghei, A. Rigo, J. Fernando Zazo, S. Lopez-Buedo, M. Torrents, F. Zyulkyarov, M. Enrico, and O. Gonzalez de Dios. 2018. dReDBox: Materializing a full-stack rack-scale system prototype of a next-generation disaggregated datacenter. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1093–1098.

[4] Aaron Call, Jordà Polo, and David Carrera. 2022. Workload-Aware Placement Strategies to Leverage Disaggregated Resources in the Datacenter. *IEEE Systems Journal* 16, 1 (2022), 1697–1708.

[5] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 153–167.

[6] Felipe Göhring De Magalhães, Mahdi Nikdast, Fabiano Hessel, Odile Liboiron-Ladouceur, and Gabriela Nicolescu. 2021. HyCo: A Low-Latency Hybrid Control Plane for Optical Interconnection Networks. In *2021 IEEE International Workshop on Rapid System Prototyping (RSP)*. 50–56. https://doi.org/10.1109/RSP53691.2021.9806198

[7] Chao Guo, Xinyu Wang, Gangxiang Shen, Sanjay Kumar Bose, Jiahe Xu, and Moshe Zukerman. 2023. Exploring the Benefits of Resource Disaggregation for Service Reliability in Data Centers. *IEEE Transactions on Cloud Computing* 11, 2 (2023), 1651–1666.

[8] Chao Guo, Moshe Zukerman, and Tianjiao Wang. 2023. Radar: Reliable Resource Scheduling for Composable/Disaggregated Data Centers. *IEEE Transactions on Industrial Informatics* 19, 8 (2023), 8551–8563. https://doi.org/10.1109/TII.2022.3222348

[9] Intel.com. 2023. Intel rack scale design architecture. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/rack-scale-design-architecture-white-paper.pdf. Accessed: Aug., 2023.

[10] Benjamin G. Lee and Nicolas Dupuis. 2019. Silicon Photonic Switch Fabrics: Technology and Architecture. *Journal of Lightwave Technology* 37, 1 (2019), 6–20.

[11] Kevin Lim, Jichuan Chang, Trevor Mudge, Parthasarathy Ranganathan, Steven K. Reinhardt, and Thomas F. Wenisch. 2009. Disaggregated Memory for Expansion and Sharing in Blade Servers. In *Proceedings of the 36th Annual International Symposium on Computer Architecture* (Austin, TX, USA) *(ISCA '09)*. Association for Computing Machinery, New York, NY, USA, 267–278.

[12] Luxtera. 2023. *Products.* https://www.cisco.com/c/en/us/services/acquisitions/luxtera.html [Online]. Available.

[13] Asif Mirza, Febin Sunny, Peter Walsh, Karim Hassan, Sudeep Pasricha, and Mahdi Nikdast. 2022. Silicon Photonic Microring Resonators: A Comprehensive Design-Space Exploration and Optimization Under Fabrication-Process Variations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 10 (2022), 3359–3372.

[14] Howraa Mehdi Mohammad Ali, Taisir E. H. El-Gorashi, Ahmed Q. Lawey, and Jaafar M. H. Elmirghani. 2017. Future Energy Efficient Data Centers With Disaggregated Servers. *Journal of Lightwave Technology* 35, 24 (2017), 5361–5380.

[15] Kazuya Nagashima, Naoya Nishimura, Atsushi Izawa, Tomofumi Kise, and Hideyuki Nasu. 2016. 28-Gb/s × 24-channel CDR-integrated VCSEL-based transceiver module for high-density optical interconnects. In *2016 Optical Fiber Communications Conference and Exhibition (OFC)*. 1–3.

[16] Antonios D. Papaioannou, Reza Nejabati, and Dimitra Simeonidou. 2016. The Benefits of a Disaggregated Data Centre: A Resource Allocation Approach. In *2016 IEEE Global Communications Conference (GLOBECOM)*. 1–7. https://doi.org/10.1109/GLOCOM.2016.7842314

[17] Zacharaya Shabka and Georgios Zervas. 2023. Network-aware compute and memory allocation in optically composable data centers with deep reinforcement learning and graph neural networks. *J. Opt. Commun. Netw.* 15, 2 (Feb 2023), 133–143.

[18] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. 2018. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 69–87.

[19] Yan Yan, George M. Saridis, Yi Shu, Bijan Rahimzadeh Rofoee, Shuangyi Yan, Murat Arslan, Thomas Bradley, Natalie V. Wheeler, Nicholas Heng-Loong Wong, Francesco Poletti, Marco N. Petrovich, David J. Richardson, Simon Poole, George Zervas, and Dimitra Simeonidou. 2016. All-Optical Programmable Disaggregated Data Centre Network Realized by FPGA-Based Switch and Interface Card. *Journal of Lightwave Technology* 34, 8 (2016), 1925–1932.

[20] Georgios Zervas, Hui Yuan, Arsalan Saljoghei, Qianqiao Chen, and Vaibhawa Mishra. 2018. Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation [Invited]. *Journal of Optical Communications and Networking* 10, 2 (2018), A270–A285. https://doi.org/10.1364/JOCN.10.00A270