2023-12-08

Energy-Efficient Workload Placement with Bounded Slowdown in Disaggregated Datacenters

Sefati, Amirhossein

UNIVERSITY OF CALGARY

Energy-Efficient Workload Placement with Bounded Slowdown in Disaggregated Datacenters

by

Amirhossein Sefati

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA
DECEMBER, 2023

Abstract

Disaggregated Data Center (DDC) is a modern datacenter architecture that decouples hardware resources from monolithic servers into pools of resources that can be dynamically composed to match diverse workload requirements. While disaggregation improves resource utilization, it could negatively impact workload slowdown due to the latency of accessing disaggregated resources over the datacenter network. To this end, we consider CPU and memory disaggregation and conduct measurements to experimentally profile several popular datacenter workloads in order to characterize the impact of disaggregation on workload execution slowdown. We then develop a workload placement algorithm, called Iterative Rounding-based Placement (IRoP), that given a set of workloads, determines where to place each workload (*i.e.*, on which CPU) and how much local and remote memory is allocated to it. The key insight in designing IRoP is that the impact of remote memory latency on slowdown can be substantially masked by assigning workloads to higher-performing CPUs, albeit at the cost of higher power consumption. As such, IRoP aims to find a workload placement that minimizes the DDC power consumption while respecting a bounded slowdown for each workload. We provide extensive simulation results to demonstrate the flexibility of IRoP in providing a wide range of trade-offs between power consumption and workload slowdown. We also compare IRoP with several existing baselines. Our results indicate that IRoP can reduce power consumption and slowdown in the considered scenarios by up to 8% and 12%, respectively.

Preface

This thesis is an original work by the author. Parts of this research have been published in the 19th International Conference on Network and Service Management (CNSM) in October 2023.

All plots in this thesis were created using the `Matplotlib` library in Python. All other figures are either original works by the author or adaptations or inspired by other works that are cited in the figures' captions.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Professor Majid Ghaderi, for his invaluable guidance and insights throughout this research. His expertise has been instrumental in shaping the outcome of this project. Additionally, I am deeply grateful for the support and encouragement I have received from my friends, colleagues, and family. Their input and motivation have played a significant role in the successful completion of this thesis.

I would also express my gratitude to Dr. Mahdi Dolati, my mentor, for his support and guidance on multiple occasions.

Since we're using LaTeX and the `memoir` package, we should thank Don Knuth, Leslie Lamport, and Peter Wilson.[1]

[1] These are the authors of [26, 28, 56].

To my parents and to my wife, who always supported me.

Table of Contents

List of Figures

List of Tables

List of Symbols, Abbreviations, and Nomenclature

Symbol	Definition
$\mathcal{C}$	Set of all computation-cable modules
F_c	Frequency of CPU cores in module c
N_c	Number of CPU cores in module c
L_c	Amount of local memory in module c
I_c^s	Static power consumption of module c
I_c^d	Dynamic power consumption of module c
M	Total amount of remote memory in DDC
$\mathcal{W}$	Set of all workloads
ν_w	Number of cores requested by workload w
ϕ_w	Processor frequency requested by workload w
μ_w	Total amount of memory requested by workload w
$P_w^f(.)$	Model to show the frequency effect on workload slowdown w
$P_w^m(.)$	Model to show the memory effect on workload slowdown w
Δ_w	Maximum acceptable slowdown of workload w
z_c	Activation of module c
$y_{w,c}$	Assignment of workload w to module c
x_w	Fraction of local to total requested memory of workload w

f_w	Frequency of allocated CPU to workload w
u_c	CPU utilization of module c

Abbreviations	Definition
DC	Data Center
DDC	Disaggregated Data Center
CPU	Central Processing Unit
RAM	Random Access Memory
NIC	Network Interface Card
ToR	Top of Rack
DIMM	Dual In-line Memory Module
QoS	Quality of Service
SLA	Service Level Agreement
IRoP	Iterative Rounding-based Placement
HPCG	High Performance Conjugate Gradients
MILP	Mixed-Integer Linear Problem
SAN	Storage Area Network
NAS	Network-Attached Storage
SoC	System-on-Chip
RDMA	Remore Direct Memory Access
API	Application Programming Interface
OS	Operating System
VM	Virtual Machine
CR	Contention Ratio
BFS	Breadth-First Search
DRAM	Dynamic Random Access Memory
DWLP	Disaggregated Workload Placement

F-DWLP	DWLP with Fixed Memory Ratios

Chapter 1

Introduction

1.1 Motivation

Today's data centers (DCs) are designed based on a server-centric model. The building block of this model is a monolithic server that includes all necessary hardware resources such as CPU, RAM, and NIC to run typical datacenter workloads. One of the main limitations of the server-centric model is the difficulty to achieve full resource utilization due to *resource stranding*, where a server that has used up one type of resource cannot run more workloads even though it may still have large amounts of other resources available. Server virtualization helps improve the utilization of hardware resources, but it cannot eliminate the problem completely as stranded resources on one server cannot be allocated to virtual machines running on other servers.

Indeed, measurements in production DCs show that the average utilization of hardware resources is relatively low. For example, a recent report [31] reveals that 80 percent of the time datacenter clusters utilize $10 - 30\%$ of their CPU capacities and more than half of the time the average memory utilization is around 50%. Also, Google and Alibaba report that the memory utilization of their clusters is around 60% [51, 32]. In addition to equipment costs, this under-utilization results in elevated power consumption, as static power consumption of fixed-ratio servers is significant [3], which has financial and environmental consequences.

For example, data centers consumed around 1.5% of the total power consumed in the U.S. in 2018, amounting to $4.5 billion [9]. Addressing the stranded resource problem has become even more critical as the model of computing is evolving in response to emerging data-centric workloads such as those in Machine Learning/Artificial Intelligence. These workloads require large amounts of processing capacity as well as memory to work efficiently. Simply over-provisioning servers with more hardware resources not only exacerbates the stranded resource problem but faces practical limitations (*i.e.*, limited DIMM slots for memory on the server board).

To overcome the limitations of the server-centric model, a *resource-centric* model is proposed for datacenters based on resource disaggregation. In a disaggregated datacenter (DDC), server hardware resources are physically or logically disaggregated into homogeneous resource pools from which resources can be allocated to workloads on demand. The scale of disaggregation can be within a single rack, a group of racks (*i.e.*, a cluster), or the entire datacenter. One of the most crucial components of a disaggregated architecture at any scale is the network interconnecting resource pools. Recent advances in low-latency networking demonstrate that microsecond-scale host-to-host latency is achievable in datacenters [14], with sub-microsecond latency within the host networking stack [21]. Nevertheless, even such network latencies are still high when accessing remote disaggregated memory, noting that accessing local memory on the same server box takes the order of tens of nanoseconds [13]. As memory access latency increases, execution slows down for those workloads that are memory intensive.

To compensate for the slower memory access, workloads can be assigned to higher-performing CPUs, *e.g.*, CPUs with higher compute capacity or higher clock frequency. However, based on the work of the authors in [47], this strategy increases datacenter power consumption (the higher the frequency, the higher the power consumption), and consequently, infrastructure costs increase. Fig. 1.1 shows how increasing the frequency of a Core i7 CPU module increases its power consumption.

Different workloads have different levels of sensitivity to memory access latency. In

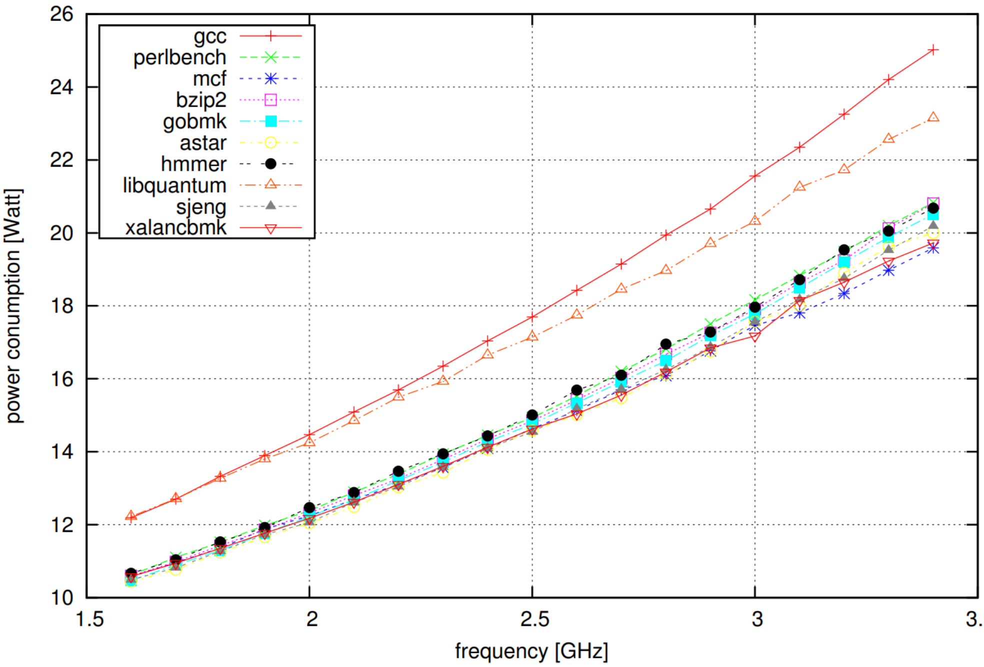


Figure 1.1: Power consumption of SPEC Integer benchmarks on Core i7 SandyBridge. [47]

Fig. 1.2, we have plotted measured *slowdown* for several popular datacenter workloads (see Section 3 for details). The slowdown of a workload is defined as below:

$$Slowdown = \frac{\text{Workload completion time when executed in a disaggregated DC}}{\text{Worklaod completion time when executed in a traditional DC}} \tag{1.1}$$

The figure clearly shows that some workloads such as Kmeans [50] are highly sensitive to memory access latency, while others such as WordCount [42] show negligible sensitivity. This behavior can be exploited to minimize the impact of memory disaggregation on workload slowdown. Recently, a few works have considered optimizing workload slowdown through run-time management [24, 34]. However, run-time management alone is not sufficient as it cannot help during the startup time. For example, workloads with a low ratio of hot to cold pages severely suffer from insufficient low-latency memory at startup [35]. Moreover, when the starting resource configuration of a workload is far from optimal, it suffers from workload slowdown while the run-time management tries to transition the workload to an optimal resource configuration. Such approaches incur substantial run-time management. Admissible placement of such jobs requires information about their sensitivity to different types

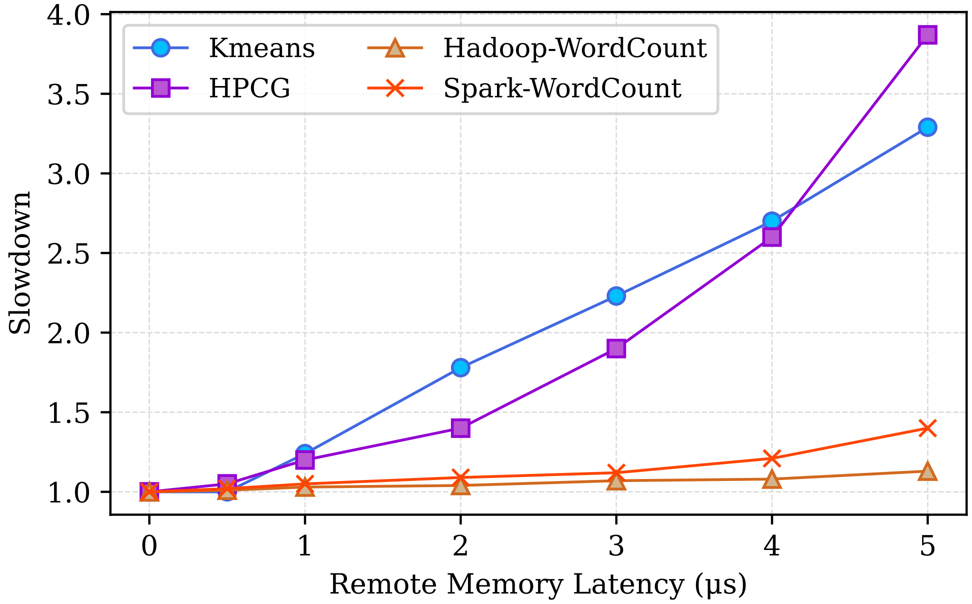


Figure 1.2: Workload slowdown as remote memory access latency increases. The local memory ration is fixed at 25%. The x-axis shows the amount of additional latency when accessing remote memory compared to local memory.

of memory at deployment time. Therefore, methods that follow a run-time management paradigm and do not provide a mechanism to incorporate performance-related information at the deployment time can not guarantee a consistent quality of service (QoS). As such, in addition to run-time management, careful workload placement at the time of deployment is needed to ensure workload slowdown is not unacceptably degraded due to remote memory access latency while minimizing datacenter power consumption. The workload placement determines: i) which CPU should we choose to run the workload on, and ii) how much local and remote memory should we allocate to it. Several works have considered workload placement in DDC. However, these works focus on either optimizing workload slowdown [57, 4, 14] or minimizing datacenter power consumption [38, 37, 40, 43, 3, 45], without considering the trade-off between the two. In this thesis, we aim to study workload placement with bounded slowdown, *i.e.*, guaranteeing slowdown does not exceed a pre-specified target level based on service-level agreements (SLAs), while minimizing DDC power consumption.

1.2 Thesis Objective

The primary objective of this thesis is to explore the impact of resource disaggregation on workload slowdown in disaggregated datacenters. Specifically, we focus on CPU and memory disaggregation and conduct measurements to experimentally profile various data center workloads. This profiling enables us to characterize the impact of disaggregation on workload execution slowdown. Then, we propose a novel workload placement algorithm, called IRoP, which aims to minimize DDC power consumption while ensuring a bounded slowdown for each workload. The algorithm optimizes the placement of workloads on CPUs and determines how much local and remote memory should be allocated to each workload, ultimately helping data center owners reduce their power costs while respecting the service-level agreements (SLAs). We emphasize achieving a balance between power consumption and slowdown to enhance data center efficiency. The specific objectives of this thesis are explained in the following subsections.

1.2.1 Study Workload Slowdown in DDCs

We consider CPU and memory disaggregation in DC, where each CPU is provisioned with a fixed (small) amount of local memory but can access remote memory modules, which are considerably larger, over the datacenter network. As it will introduce significantly higher latency for accessing memory, the first objective is to devise a technique to accurately profile workloads in order to comprehensively model the impact of remote memory latency and CPU processing capacity on the workload slowdown. Specifically, we consider several popular workloads and run each of them in isolation on a modified Linux system which allows us to change the local and remote memory ratios as well as scale the CPU frequency. We then measure the completion time of each workload and approximate the impact of remote memory latency and CPU frequency on workload slowdown using piece-wise linear functions.

1.2.2 Develop a Workload Placement Algorithm

In the pursuit of optimizing datacenter operations, two crucial objectives stand at the forefront: reducing power costs and ensuring timely completion of workloads. Conventional datacenters often suffer from *resource stranding*, where certain resources remain underutilized, leading to increased power consumption. On the other hand, the advent of disaggregated datacenters introduces a new challenge - the *slowdown* of workloads caused by the higher latency associated with accessing remote memory. Given these critical concerns, our thesis aims to address this dilemma by devising a rapid algorithm for power-efficient workload placement in disaggregated datacenters, all while ensuring that each workload operates within a predefined threshold of slowdown. This innovative approach seeks to strike a balance between power consumption and workload slowdown, offering a practical and effective solution for enhancing the efficiency and adaptability of datacenter infrastructures.

1.3 Thesis Contribution

In this section, we will provide a comprehensive overview of the significant contributions made in this research thesis, which addresses the challenges of optimizing resource allocation in Disaggregated Data Centers (DDCs). Our contributions involve the development of a novel Workload Profiling Framework and the proposal of an efficient Workload Placement Algorithm, both aimed at minimizing power consumption while ensuring performance objectives for each workload.

1.3.1 Workload Profiling Framework

The first major contribution of this thesis is the introduction of a robust Workload Profiling Framework. In modern DDC environments, the performance of workloads can be significantly impacted by remote memory access latency and CPU processing capacity. To tackle this challenging issue, we designed a profiling framework that accurately models the sensitivity of various workloads to memory access latency and CPU frequency. We define *workload slowdown* as the ratio of completion time when the workload is executed under a disaggregated environment compared to a traditional data center. The key innovation of our approach lies in the use of piece-wise linear functions to represent the relationship between workload slowdown and these two critical factors.

We embarked on an in-depth exploration to develop the Workload Profiling Framework. Our research involved studying and analyzing a diverse range of workloads representative of real-world applications and scenarios. By carefully capturing the slowdown characteristics of these workloads under varying conditions of remote memory access and CPU frequency, we formulated piece-wise linear models that accurately depict the impact of these factors on workload slowdown.

To validate the effectiveness of our profiling models, we conducted extensive experiments in a controlled environment. For this purpose, we made intricate modifications to the latest version of the Linux Kernel (version 6.1), allowing us to inject artificial delays when accessing the swap area. These delays closely mimic the real-world remote memory latency experienced in a DDC. The experimental results showcased the high accuracy and reliability of our profiling models in characterizing real-world workload slowdown. By enabling us to understand the sensitivity of different workloads to remote memory and CPU frequency variations, this framework lays the foundation of an efficient workload placement algorithm in DDCs.

1.3.2 Workload Placement Algorithm

The second major contribution of this research is the proposal of an innovative Workload Placement Algorithm. Armed with the valuable insights gained from our Workload Profiling Framework, we formulated the workload placement problem in DDCs as a mixed-integer linear problem (MILP). We recognized that while minimizing workload slowdown is crucial, we must also take into account the overarching objective of reducing power consumption in the DDC.

The formulation of the MILP is an essential step in addressing the workload placement problem effectively. To this end, we built upon the comprehensive MILP model proposed in a prior work [3], which accurately represents and models the power consumption of resources within the DDC. However, we acknowledged that directly solving the MILP would be computationally expensive, as it is inherently NP-hard. As a result, we focused on devising a fast polynomial-time algorithm, which we named IRoP, to efficiently solve the workload placement problem while considering both power consumption and workload slowdown.

The design of IRoP is based on transforming the MILP problem into a modified variant of the multi-dimensional bin packing problem [8]. Leveraging the framework of deterministic rounding of relaxed integer programs, we ensured that IRoP provides high-quality approximations while being computationally efficient. By striking an optimal balance between power consumption and workload slowdown, IRoP efficiently allocates resources in the DDC and delivers near-optimal solutions to the workload placement problem.

1.3.3 Evaluation and Results

We recognized the paramount importance of evaluating the performance and effectiveness of our proposed IRoP algorithm. Therefore, we conducted extensive experiments, evaluating its utility across a wide range of realistic scenarios and workloads. Our evaluation metrics revolved around power consumption and workload slowdown, both of which are critical aspects of DDC resource allocation.

The results of our experiments were highly promising, further affirming the strength of our contributions. The IRoP algorithm demonstrated significant reductions in both power consumption and workload slowdown, achieving impressive improvements of up to 8% and 12%, respectively, in the considered scenarios. These results not only highlighted the efficacy of our proposed algorithm but also showcased its practical applicability in real-world DDC environments.

In summary, the main contribution of this thesis can be summed up as follows:

- We present a framework for profiling workload slowdown with respect to remote memory and CPU processing capacity using simple piece-wise linear functions. We empirically show that the proposed functions provide a highly accurate estimation of real-world workload slowdown.

- We formulate workload placement in DDCs as a MILP problem whose objective is to minimize a combination of total power consumption and workload slowdown. We then design an efficient approximation algorithm, called IRoP, to solve the MILP problem based on the deterministic rounding framework and analyze its theoretical performance.
- We conduct extensive experiments to evaluate the performance and utility of IRoP in terms of power consumption and workload slowdown in a variety of realistic scenarios. Our results indicate that IRoP reduces both power consumption and slowdown in the considered scenarios by up to 8% and %12, respectively.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2: Background and Related Works provides a comprehensive overview of networking protocols, data center interfaces, and related research works in the context of physically disaggregated data centers. It highlights the challenges posed by high latency and CPU load in such environments and explores the evolution of networking protocols, including RDMA and InfiniBand, to address these issues. Additionally, it discusses the categorization of interfaces into clean-slate and swap-based designs, emphasizing their impact on workload management. The chapter also delves into the extensive research efforts aimed at optimizing resource allocation in disaggregated data centers, focusing on minimizing workload slowdown, power consumption, and maximizing utilization. This wealth of information forms a foundational understanding of the key concepts and challenges in the field of resource disaggregation within data centers.

Chapter 3: Profiling Datacenter Workloads provides a comprehensive overview of the datacenter model under consideration and focuses on developing workload profiles that capture the impact of resource disaggregation on workload slowdown. It introduces the key elements of the datacenter model, including resource disaggregation, power consumption modeling, and workload characteristics. The chapter then dives into the detailed profiling of popular datacenter workloads, analyzing the effects of remote memory access and CPU frequency on workload performance. This profiling equips datacenter operators with valuable insights for optimizing resource allocation, enhancing datacenter efficiency, and meeting the unique requirements of diverse workloads.

Chapter 4: Workload Placement Problem Formulation introduces a thorough exploration of the fundamental elements and constraints involved in formulating the Disaggregated Workload Placement (DWLP) problem. It introduces essential decision variables, constraints, and the objective function that underpins the optimization framework for allocating workloads to CPU modules in a disaggregated data center. Furthermore, it highlights the critical considerations related to workload slowdown, power consumption modeling, and optimization objectives, emphasizing the trade-off between workload performance and power efficiency. This chapter serves as the foundational framework upon which subsequent chapters will build to address the efficient operation of disaggregated data centers.

Chapter 5: Proposed Algorithm for Workload Placement presents a novel algorithm, named "Iterative Rounding-based Placement" (IRoP), designed to address the complex problem of workload placement in disaggregated data centers. It presents a comprehensive theoretical analysis of IRoP, including its approximation ratio and runtime complexity. The algorithm's theoretical guarantees, such as a bounded approximation ratio and polynomial-time complexity, underscore its effectiveness and efficiency in providing near-optimal solutions for data center resource allocation challenges. This chapter equips data center operators and researchers with a powerful tool to optimize performance, power consumption, and resource utilization in modern data center infrastructures.

Chapter 6: Performance Evaluation provides a comprehensive methodology and analysis framework for assessing the efficacy of different workload allocation strategies within a large-scale disaggregated data center environment. It describes the environment setup, alternative algorithms for comparison, and evaluates performance based on various metrics such as total slowdown, total power consumption, and a weighted sum of both. Additionally, it explores the runtime scalability of the proposed algorithm (IRoP) in comparison to an optimization benchmark (OPT). Overall, this chapter offers a detailed foundation for understanding and optimizing data center operations in a complex and dynamic context.

Chapter 7: Conclusion and Future Works offers a recapitulation of the key findings and contributions of the research, emphasizing the effectiveness of the IRoP algorithm in optimizing workload placement within Disaggregated Data Centers (DDCs). Additionally, it outlines promising avenues for future research, including the expansion of the power consumption model, development of orchestration strategies, real-world implementation, profiling of different workloads, and the integration of AI and machine learning techniques. This chapter serves as both a summary of the accomplishments and a roadmap for continued advancements in the field of DDC resource allocation and management.

Chapter 2

Background and Related Works

This chapter provides the background knowledge required for understanding the proposed solutions in this thesis. First, we provide an overview of disaggregation in the datacenters, networking protocols, and interfaces. Then, we elaborate on the requirements that a resource allocation algorithm should respect when used in a DDC. Finally, related works are studied and investigated in this chapter.

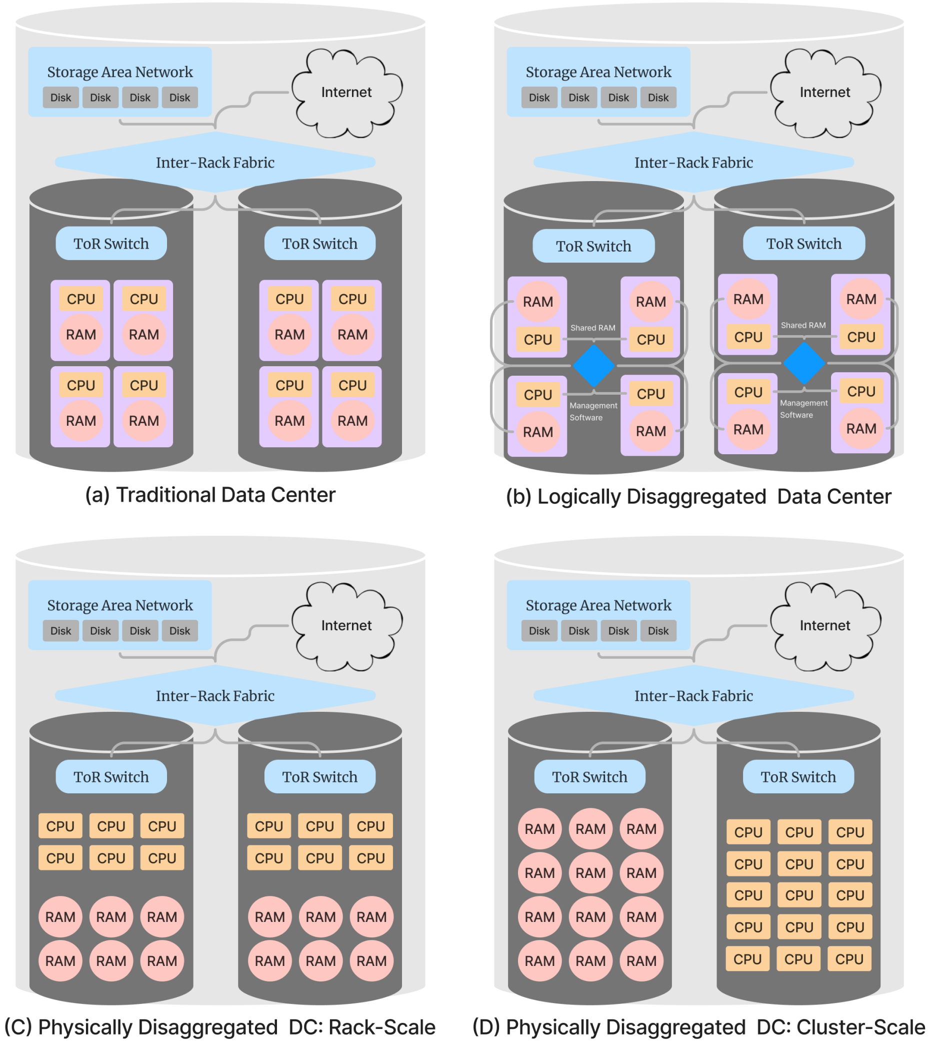
2.1 Datacenter Architectures

In this section, we review the main architectures of datacenters and we bring the benefits of using a disaggregated datacenter over a traditional one.

2.1.1 Traditional Datacenters (TDCs)

TDCs, an abbreviation for traditional data centers, are distinct for their utilization of monolithic servers, where different components such as CPU and memory are densely integrated into a single hardware unit. This architectural paradigm, clearly illustrated in Fig. 2.1a, gives rise to a series of limitations that warrant a deeper examination to uncover the potential for improvement and innovation.

The first noteworthy limitation that arises in this traditional server-centric model is the



(a) Traditional Data Center

(b) Logically Disaggregated Data Center

(C) Physically Disaggregated DC: Rack-Scale

(D) Physically Disaggregated DC: Cluster-Scale

Figure 2.1: Different Architectures of Data Centers.

challenge of low resource utilization, particularly considering the diverse and varying nature of workload requests. To elucidate this concern, envision a scenario where a single server is equipped with an impressive 16 CPU cores and an abundant 64 GB of memory. However, suppose a specific job only necessitates 16 CPU cores and a modest 16 GB of memory to fulfill its requirements. In this case, the remaining 48 GB of memory remains largely untapped and unutilized, representing an inefficiency that translates into increased power consumption and higher maintenance costs [45].

The second limitation lies in the inherent challenges associated with incorporating or updating individual components within this server design. Due to the tightly coupled nature of these monolithic servers, introducing changes or advancements often entails modifying the entire server unit. As a result, embracing new and more effective technologies that may enhance efficiency and optimization becomes cumbersome and cost-prohibitive [11]. The lack of flexibility in the face of technological advancements poses a substantial obstacle for data centers seeking to stay on the cutting edge of innovation.

Additionally, the tightly coupled architecture in traditional data centers can lead to significant consequences if any of the integrated components within a server encounter a malfunction or failure. In such instances, the entire server becomes adversely affected, resulting in downtime and service disruptions [7]. The lack of fault tolerance and redundancy in this setup poses reliability challenges that may hinder seamless operations in critical situations.

Moreover, the limitations of traditional data centers extend beyond their architectural design and resource utilization challenges. Another significant concern is their environmental impact and power consumption. As data centers continue to grow in scale and complexity to accommodate the escalating demand for digital services, their power requirements have surged. The power consumption of TDCs not only contributes to higher operational costs for data center operators but also raises environmental concerns, given the substantial carbon footprint associated with traditional power sources.

In conclusion, as data centers strive to evolve and adapt to meet the ever-growing demands of modern applications and services, especially with the rapid rise of resource-intensive technologies such as artificial intelligence (AI) and machine learning (ML) applications, addressing these limitations becomes imperative. By finding innovative solutions to enhance resource utilization, improve flexibility, and ensure robustness in the face of potential hardware failures, data centers can aim to achieve higher efficiency, cost-effectiveness, and adaptability in their operations. The exploration of alternative data center architectures, such as logically or physically disaggregated models, presents promising avenues to overcome these challenges and embrace a more dynamic and agile approach to meet the diverse needs of modern computing workloads. Fig. 2.2 shows a more detailed traditional datacenter that uses standalone servers in each rack and does not require external communication between CPU and memory modules. In this architecture, networking is only used for external applications and it is generally not a vital component for the servers to execute workloads. Also, as a result of accessing only local memory, there is no slowdown for the workloads running in TDCs.

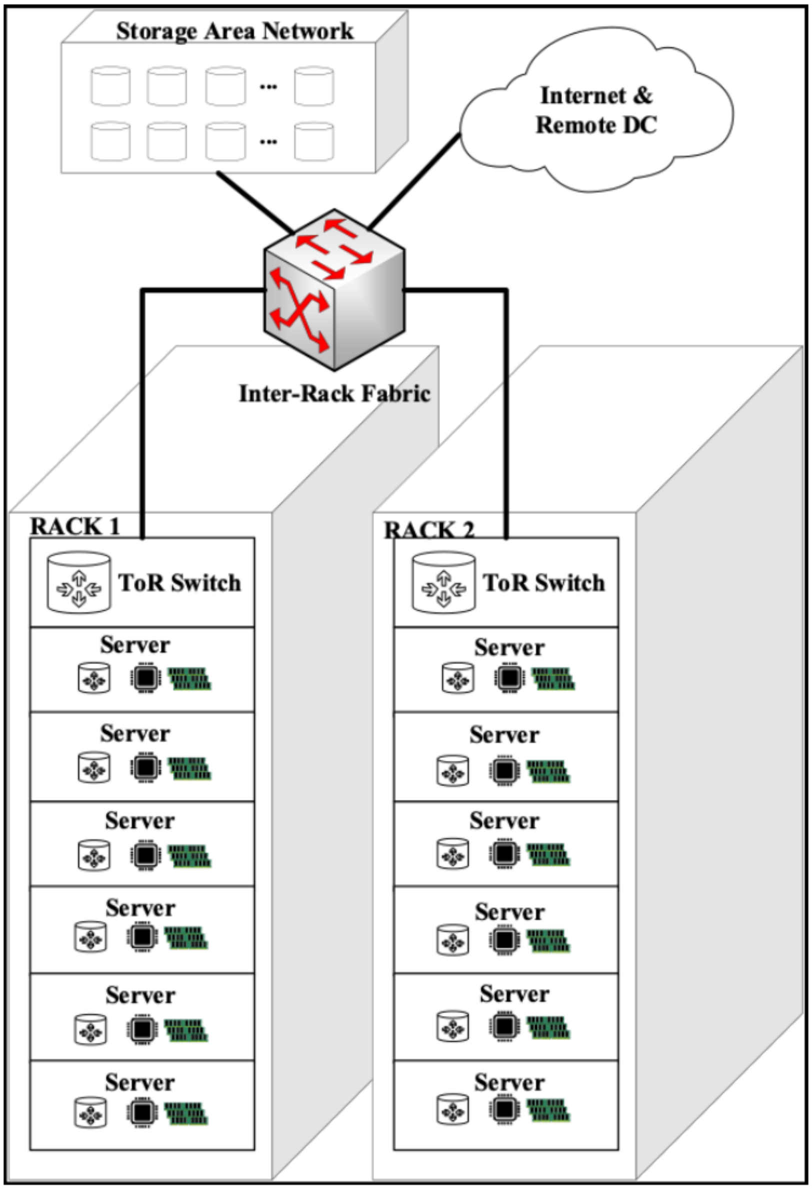


Figure 2.2: Traditional datacenter infrastructure. The figure shows two racks in a traditional DC comprised of servers and a storage area network. [3]

2.1.2 Logically Disaggregated DCs

Logically disaggregated data centers, akin to traditional ones, retain servers as the fundamental unit of computation, interconnected through low-latency network technologies, as depicted in Fig.2.1b. Within this architectural model, one global software plays a vital role in effectively managing access to remote memory, facilitating the utilization of unutilized memory from other servers by various jobs. Consequently, this logical disaggregated design enables the software to intelligently allocate the remaining available memory, which would otherwise go to waste, to cater to other jobs in need of additional memory resources. Additionally, the fault tolerance and redundancy offered by logically disaggregated data centers address the reliability challenges of TDCs. With resources distributed across separate units, a failure in one component does not lead to the collapse of the entire system. Instead, the affected resource unit can be quickly isolated and replaced, ensuring continuous operations and maintaining the availability of critical services. The industry has shown increasing interest in this architecture, owing to its promising advantages [27, 16, 2], with previous studies also having delved into its potential [12].

Despite the benefits it offers, this architecture does not completely eliminate the challenges faced by traditional data centers. The tightly coupled nature of servers introduces certain hurdles related to adoption and reliability. As data centers strive to keep pace with the escalating demands of modern applications and services, tackling these challenges becomes imperative to achieve heightened efficiency, scalability, and fault tolerance in disaggregated data center operations. To this end, exploring innovative solutions that optimize resource utilization, streamline software management, and enhance server interconnectivity will play an important role in shaping the future of logically disaggregated data centers.

In conclusion, logically disaggregated data centers represent a promising evolution in data center architecture, maintaining servers as the core units of computation while enhancing memory allocation through global software management. While this approach addresses some utilization concerns faced by traditional data centers, the journey toward realizing its full potential necessitates overcoming challenges related to server coupling. By strategically exploring and implementing innovative solutions, data centers can aspire to achieve higher levels of efficiency, scalability, and reliability, thereby laying a robust foundation for the seamless execution of modern computing workloads in the digitally transformative landscape.

2.1.3 Physically Disaggregated DCs

Unlike the other two architectures discussed previously, physically disaggregated data centers take the separation of hardware components to a whole new level. This approach marks a notable departure from the traditional monolithic server-centric model and the logically disaggregated architecture. In the case of physically disaggregated data centers, significant advancements have been made historically in disaggregating data storage, leveraging technologies such as storage area networks (SANs) and network-attached storage (NAS) systems. For instance, in a seminal work by the authors of [25], they proposed a flash storage disaggregation method to address complex scenarios where jobs require diverse compute-to-storage resources. Moreover, as far back as 2015, Facebook introduced Yosemite [30], an innovative disaggregated system-on-chip (SoC) that harnessed the power of interconnecting storage components to optimize data handling. Consequently, these architectures, just like the previously mentioned ones, still utilize SANs as an integral part of their design to effectively decouple storage from computation resources, as shown in Fig 2.1.

However, the real challenge lies in the disaggregation of CPU and memory components, which forms the struggle of implementing physically disaggregated data centers. In this architecture, the CPU and memory are physically decoupled while remaining interconnected via a fast, low-latency network. Within this dynamic setup, the memory modules are commonly referred to as *far* [4, 27] or *remote* [13, 54, 7, 3] memory. This arrangement introduces a novel concept where CPU modules operate in tandem with remote memory, which may be situated on separate servers. The notion of remote memory in this context implies that certain portions of the memory are geographically distant from the CPU, requiring the efficient orchestration of data transfer and access to ensure seamless performance. In many studies (*i.e.*, [49, 7]), it is assumed that the CPU modules should retain at least some amount of local memory, acting as a cache to the remote memory, while the remote memory primarily hosts the main workload codes and datasets. Furthermore, to avoid the undesirable impact of high workload slowdown due to the relatively high access latency of remote memory, the system may intelligently assign available local memory to workloads. By doing so, the total number of accesses to remote memory regions can be minimized, contributing to enhanced overall performance and efficient resource utilization.

The concept of physically disaggregated data centers manifests itself in two distinct scopes, commonly known as rack-scale and cluster-scale, both depicted in Fig.2.1c and Fig.2.1d, respectively. In the rack-scale model, homogeneous resources (i.e., different modules) are allocated to single-type resource pools within a rack, and each module remains exclusively available to other modules situated within the same rack. A notable example of a rack-scale disaggregated platform is detailed in [19], wherein Huawei presented a novel architecture tailored for big data applications. This design features single-type pools of each module (e.g., CPU, memory, I/O) interconnected by a high-throughput network, enhancing communication and data exchange between components. Fig. 2.3 shows a more detailed rack-scale disaggregated data center that separates CPU and memory modules in each rack.

While the rack-scale approach undoubtedly offers some advantages, such as localized resource management, it does present certain limitations concerning scalability, which might become more apparent as data center demands continue to grow. As a result, researchers and data center operators have sought alternative solutions to address scalability concerns effectively. Cluster-scale architectures adopt a more encompassing approach, where the disaggregation extends across the entire data center. However, in the past, these cluster-scale architectures faced practical challenges due to the inherent complexity of managing network requirements and facilitating efficient traffic between CPU and memory modules located in different racks. This complexity introduced bottlenecks that hindered the widespread adoption of cluster-scale architectures. Nonetheless, recent advancements in high-throughput low-latency network protocols have proven instrumental in making the cluster-scale architecture more practical and viable. These improved protocols enable robust communication

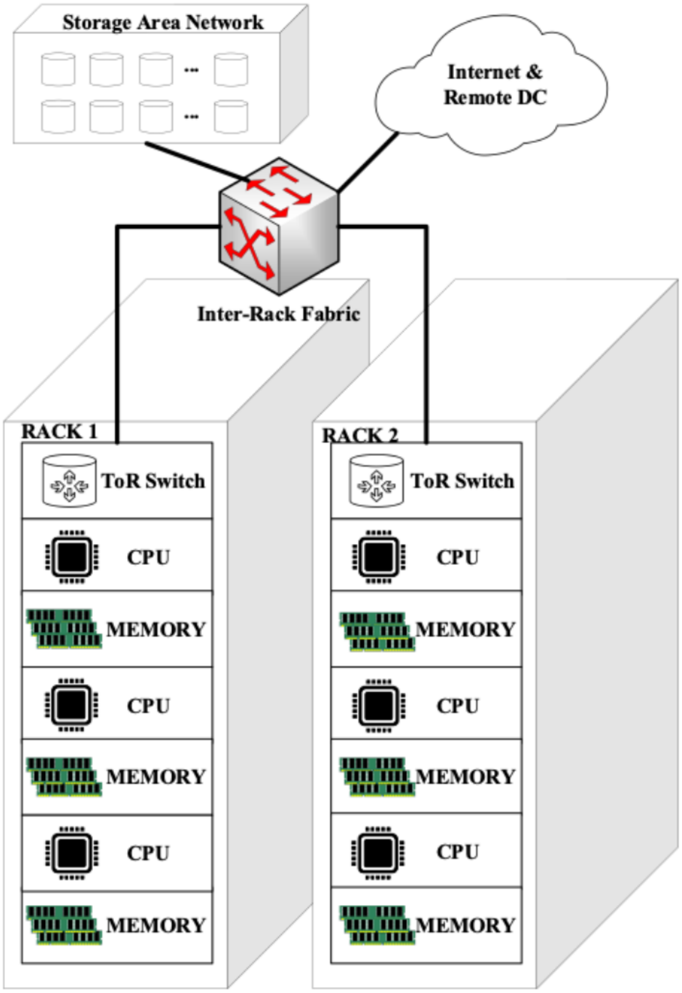


Figure 2.3: Rack-scale Disaggregated DC Infrastructure [3]

and data exchange between various components, mitigating potential bottlenecks and facilitating seamless coordination across the entire data center. Therefore, for the purposes of this thesis and the proposed resource allocation algorithm, we adopt a cluster-scale physically disaggregated model. The cluster-scale approach aligns well with the demands of modern data centers, offering increased flexibility, scalability, and potential for future growth. Fig. 2.4 shows a more detailed cluster-scale disaggregated data center that separates CPU and memory modules cluster-wide.

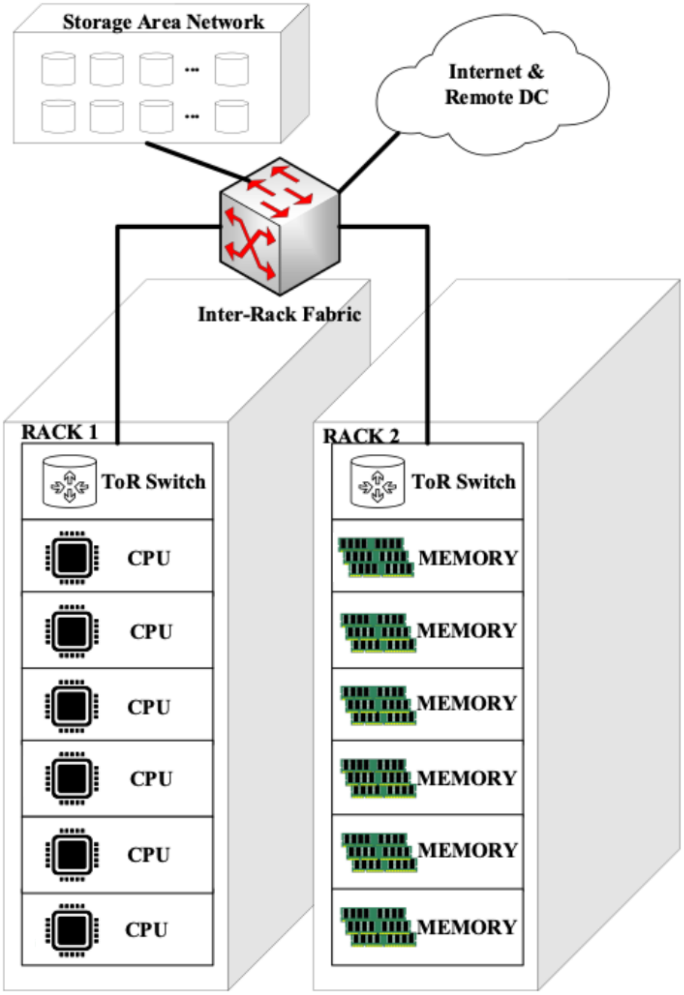


Figure 2.4: Cluster-scale Disaggregated DC Infrastructure [3]

2.2 Networking Protocols in Datacenters

In a physically disaggregated data center, where CPU and memory are decoupled, the completion time of workloads, especially high latency-sensitive memory-intensive workloads, can suffer significantly due to the inherent high latency associated with accessing remote memory [3]. Recognizing the importance of addressing this issue, recent research has extensively explored the evolution of networking protocols to mitigate the impact of high latency and improve overall system performance.

One notable study [20] extensively reviewed the evolution of networking protocols, particularly in traditional data centers where Ethernet technology and TCP/IP are commonly

Table 2.1: Network Requirements in DDCs [13].

Communication Type	Latency (ns)
CPU - CPU	10
CPU - Local Memory	20
CPU - Remote Memory	$5 \cdot 10^3$
CPU - Disk	$10^4 - 10^5$

used for data transmission. TCP/IP, being an OS-based networking protocol, imposes tens of microseconds of delay and leads to high server CPU load, which can adversely affect workload slowdown. In an interesting observation made in [60], it was pointed out that dedicating a core to handle high throughput TCP/IP connections, to enhance the workload slowdown, would render that core unusable as a virtual machine, limiting the overall resource utilization.

To address these latency and CPU load challenges, researchers have explored alternative network protocols such as RDMA (Remote Direct Memory Access) over Converged Ethernet (RoCE). RoCE provides lower access latency and significantly reduces the CPU overhead compared to traditional TCP/IP-based solutions. Furthermore, InfiniBand [6], as a dedicated networking stack, offers lower latency, efficient flow control, and reliable lossless transportation of packets, making it a compelling alternative specifically designed for RDMA applications [10, 4, 13].

Recent works in the field have demonstrated impressive achievements in hardware-based memory disaggregation solutions. For instance, the Clio [17] project reported an end-to-end latency of 2.5 μs at the median and 3.2 μs at the 99th percentile, showcasing its effectiveness in reducing access latencies. Additionally, Google's Aquila [14] proposed an extraordinary 4 μs (median) end-to-end latency for Remote Memory Access (RMA) operations within a high-scale cluster of up to 1152 interconnected hosts. This breakthrough was attributed to Aquila's highly integrated NIC (Network Interface Card) and network, enabling ultra-low latency communication.

In addition to these protocols, there is an important study authored by Peter X. Gao and his team *et al.* [13]. This study outlines specific network requirements crucial for resource disaggregation, and you can find a concise summary of these requirements in Table 2.1. After conducting a thorough review of various network protocols and recent advancements, it is evident that the network prerequisites needed to support resource disaggregation are either already in place or expected to be available in the near future.

For the purposes of this thesis, we are working with the assumption of an end-to-end latency of 5 μs for accessing remote memory. This assumption sets the stage for our evaluation of the resource disaggregation model and its performance in line with these network requirements. By making the most of these cutting-edge networking solutions, our goal is to attain efficient power consumption and optimize workload slowdown within the context of cluster-scale physically disaggregated data centers.

2.3 DDC Interfaces

In a cluster-scale disaggregated data center, the utilization of local and remote memory by workloads relies on the interface provided by the system. Interfaces in a resource disaggregated model can be broadly categorized into two types [53]. First, we have the *clean-slate* designs that offer application programming interfaces (APIs) to developers, enabling them to exert control over local and remote memory access within the cluster. This approach is known for its efficiency, as it bypasses the operating system (OS) and reduces the CPU load. However, it necessitates developers to modify their workload implementations to accommodate the new interface. Notable examples of clean-slate interface providers include Kona [7], AIFM [48], and Remote Regions [2].

On the other hand, *swap-based* systems extend swap techniques to the virtual address of remote memory modules. In this design, the OS takes charge of controlling and managing memory accesses and instructions, such as Load and Store operations. While this approach introduces some latency overhead, the interface remains transparent and readily accepts workloads in their existing form, without requiring extensive modifications. Notably, several research efforts, including LegoOS [49], FastSwap [4], InfiniSwap [16], and Semeru [52], have advocated for the adoption of swap-based interfaces, offering their practicality and cost-effectiveness.

In conclusion, the choice of interface in a cluster-scale disaggregated data center plays a pivotal role in determining how workloads leverage both local and remote memory resources. The clean-slate designs provide developers with direct control and increased efficiency but at the cost of requiring modifications to workloads. On the other hand, swap-based systems offer a more transparent and readily adaptable interface, making them a practical and cost-effective choice. In this thesis, while our proposed models may still remain valid using a clean-slate interface, we have opted for the latter category, the swap-based approach.

2.4 Related Works

Resource allocation in disaggregated data centers presents a unique set of challenges distinct from traditional data centers. While conventional data centers focus mainly on minimizing network power consumption (e.g., Network-Aware Algorithm [36] and MCRVMP [5]), resource allocation in disaggregated data centers necessitates consideration of various additional factors. In this context, efficient resource allocation requires addressing the impact of remote memory latency, CPU processing capacity, and workload characteristics on overall performance and power consumption.

Unlike traditional data centers, where allocating resources to jobs is relatively straightforward, disaggregated data centers introduce complexities due to the physical separation of resources like memory from compute nodes. This separation results in increased latency for remote memory access, which can significantly impact the performance of memory-intensive workloads. Additionally, assigning workloads to CPUs with inadequate processing capacity can lead to slowdowns and resource inefficiencies.

Power efficiency is another crucial concern in modern data centers, given the substantial power costs associated with data center operations. Resource allocation strategies in disaggregated data centers must aim to minimize power consumption while ensuring that performance objectives are met.

To tackle the resource allocation problem in disaggregated data centers, researchers have taken a diverse range of approaches. These efforts can be neatly grouped into three primary categories: i) Minimizing Workload Slowdown, ii) Minimizing Power Consumption, and iii) Maximizing Utilization. Each of these categories explores specific aspects of resource allocation, all with the shared aim of elevating the overall performance and efficiency of disaggregated data centers.

In the following sections, we will delve into these three categories and examine the different efforts made by researchers to achieve efficient resource allocation in disaggregated data centers. Through these efforts, we aim to gain a comprehensive understanding of the state-of-the-art techniques and their respective contributions to the field of resource allocation in this evolving and critical domain.

2.4.1 Power Consumption

Authors in [45] proposed a novel algorithm with the primary goal of minimizing power consumption while simultaneously maximizing resource utilization in a rack-scale physically disaggregated DDC. They consider the presence of optical and electrical interconnects as fast and generic backplanes, respectively, within the DDC architecture. The proposed model for power consumption takes into account the utilization of CPU modules and network resources, attributing 85% of the total power consumption to CPUs and 15% to the network. Despite its efficacy in optimizing power usage, this algorithm neglects to consider the potential slowdown of workloads using the mentioned resource allocation algorithm, which may lead to SLA violations, particularly when allocating more remote memory to latency-sensitive workloads causing high degradation in performance.

In [38], researchers tackle the power consumption optimization challenge using a mixed-integer linear programming (MILP) model. Their study encompasses three types of workloads: memory-intensive, IO-intensive, and processor-intensive. The results demonstrate impressive power savings of 42%, 24%, and 11% for memory, IO, and processor-intensive applications, respectively, compared to traditional data centers. As the problem scales up, EERPVMM-DS [37] comes into the picture as a scalable heuristic solution. Employing a greedy approach, this heuristic tackles the challenge of minimizing power consumption while assuming a time-slotted model for the arrival of virtual machine (VM) workloads. However, a potential drawback of this approach lies in the possibility of multiple migrations for long-life-cycle VMs due to the allocation process in each time slot which causes delays in workload execution. This means a higher slowdown for the workloads that have a longer life-cycle which affects the fairness of the algorithm.

Building upon the work of EERPVMM-DS, EERP-DSCF [40] extends the previous heuristic by dropping the time-slotted assumption while maintaining the same underlying model. The authors aim to address the issue of multiple migrations by optimizing VM allocation across time slots, resulting in a more stable allocation strategy. Furthermore, the work in [3] presents a more sophisticated MILP model that accounts for the complex characteristics of disaggregated architectures, seeking to minimize power consumption more realistically. To complement the MILP model, the authors propose HEEP, a heuristic that embraces a greedy approach. This heuristic involves sorting workloads and CPUs in descending order based on their resource demands and power consumption efficiency, respectively, for optimal allocation.

Adding to the array of research efforts, [45] introduces another variant of the power consumption model for CPUs, this time considering a physically disaggregated DC with optical and electrical interconnects as backplanes. In this model, CPUs undergo an iterative filtering and prioritizing process, incorporating predefined weights associated with power efficiency, utilization, and communication delay. This iterative process aims to strike a delicate balance between minimizing power consumption and maximizing resource utilization. Nonetheless, one common limitation across all these works is the absence of consideration for the potential slowdown of jobs, which could have significant implications on SLA compliance.

As the research in the domain of power consumption in DDCs continues to unfold, the quest for comprehensive and efficient solutions remains ongoing. The identified research endeavors present valuable contributions to understanding and optimizing power usage, but the aspect of workload slowdown and its impact on SLA compliance needs further investigation. As data centers endeavor to meet the ever-increasing demands of modern applications and services, addressing power consumption challenges becomes a critical aspect in ensuring sustainable and cost-effective data center operations. Novel solutions that strike a balance between power efficiency and workload slowdown will undoubtedly pave the way for the next generation of power-efficient and resilient data center architectures.

2.4.2 Workload Slowdown

In the context of slowdown in disaggregated data centers (DDCs), several innovative approaches have been investigated to optimize system performance and minimize the impact of workload slowdown on critical applications. As the demand for higher computational power and memory capacity increases, DDCs have gained popularity due to their scalability and flexibility. However, the separation of resources in DDCs, mainly memory and CPU, can introduce communication delays which results in workload slowdown. Researchers and industry experts have devoted considerable efforts to address this challenge and enhance the overall efficiency of DDCs.

One notable approach proposed by Nvidia in [57] involves the development of an integrated system within the operating system (OS) to efficiently optimize workload slowdown. By minimizing the time spent on data transfers and streamlining memory access, this approach aims to reduce workload slowdown and improve the responsiveness of workloads running in a disaggregated environment. Also, the authors propose four additional optimizations: native support for transparent huge page migration, multi-threaded migration of a page, concurrent migration of multiple pages, and symmetric exchange of pages. Through experimental evaluations using x86, Power, and ARM64 systems, Nvidia demonstrates the potential benefits of their approach, reducing kernel software overheads and improving raw page migration throughput over 15× leading to lower workload slowdown by up to 40%.

Another research effort, highlighted in [4], focuses on examining the impact of the remote memory to local memory ratio in DDCs and its correlation with workload slowdown. First, a faster swapping mechanism that makes it possible to support remote memory at rack-scale is proposed as an RDMA-based swapping system which is called *FastSwap*. Then, the authors also develop a polynomial model and introduce a latency-based workload placement algorithm called CFM. By carefully allocating an appropriate amount of local memory to meet the slowdown requirements (*i.e.*, SLAs) of workloads, CFM aims to proactively address slowdown concerns before they significantly impact critical tasks. They review different memory allocation policies including uniform policy (*i.e.*, using a fixed local memory ratio for all the jobs), variable policy (*i.e.*, using a variable ratio), and memory-time policy (*i.e.*, using a memory-time product to determine the best local memory ratio for each job). They use the last configuration as the main policy to decide how much of local and remote memory should be assigned to each workload. This intelligent resource allocation strategy also improves success rates in meeting future demand, contributing to enhanced system stability and responsiveness.

In addition to software-level optimizations, architectural proposals have also emerged to address slowdown challenges in DDCs. One such proposal is Aquila, presented by Google in [14], an experimental data center network fabric that prioritizes ultra-low latency. Aquila incorporates the GNet protocol and custom ASIC with low-latency Remote Memory Access (RMA), achieving remarkable sub-10 μs execution times for data transfers. By reducing communication delays and optimizing data access, Aquila effectively minimizes workload slowdown, making it an attractive solution for high-performance DDCs.

Following a similar approach to [4], HoPP [29] proposes a revised operating system for DDCs that significantly improves memory management. The novel operating system decouples address capture from page faults by recording all memory access logs in the memory controller. This design optimizes memory access and reduces unnecessary overhead, leading to potential reductions in workload slowdown and improved workload slowdown.

While the aforementioned approaches have demonstrated promising results in minimizing slowdown and enhancing the efficiency of DDCs, it is essential to consider the broader implications of such optimizations. For instance, some of the explored approaches, including those mentioned in [4] and [49], have primarily focused on optimizing performance without considering the implications of power consumption. In DDCs, where resource allocation and usage directly affect power efficiency, overlooking the power aspect can have significant financial and environmental impacts.

In summary, the increasing demand for computational power and memory capacity has led to the widespread adoption of Disaggregated Data Centers (DDCs) due to their scalability and flexibility. However, the separation of resources within DDCs, particularly memory and CPU, can result in communication delays and subsequent workload slowdowns. To mitigate this, innovative approaches have been investigated, such as Nvidia's integrated system within the operating system, aimed at optimizing data transfer and memory access to minimize workload slowdown. Additionally, research efforts have focused on examining the impact of remote memory to local memory ratios, proposing intelligent resource allocation strategies to proactively address slowdown concerns. Architectural proposals like Aquila and HoPP have also emerged, prioritizing ultra-low latency and optimizing memory management to reduce unnecessary overhead and improve workload responsiveness. While these approaches demonstrate promising results in enhancing DDC efficiency, it's crucial to consider broader implications, including power consumption, to ensure sustainable and cost-effective operations.

2.4.3 Utilization

In the quest for optimizing resource utilization and performance in data centers, researchers have explored various algorithms and models to efficiently allocate resources and meet the demands of modern workloads. In [59], the authors propose two algorithms, namely NULB and NALB, aimed at realizing globally optimized IT (i.e., CPU, memory, and storage) resources in disaggregated data centers. NULB introduces the concept of Contention Ratio (CR), which specifies the demand for each resource type. The algorithm then initiates a search for IT resources based on their CR scores and utilizes a breadth-first search (BFS) algorithm to find other compatible IT resources surrounding the allocated one. Building on this foundation, the improved algorithm, called NALB, incorporates a modified version of BFS that also considers available network bandwidth when searching for network resources after allocating IT modules. By considering both IT and network resources, NALB aims to achieve even higher levels of resource utilization and performance optimization. However, it is worth noting that in these algorithms and research, the authors focused on workload rejection rates and resource utilization, while other important aspects such as power consumption of the resources or the workload slowdown (*i.e.*, Service Level Agreements) were not explicitly taken into account.

In [44], another approach to resource allocation is presented, employing a simulated annealing-based algorithm within a rack-scale architecture. The primary objective of this algorithm is to minimize the rejection rate of workloads while maximizing overall resource utilization. By simulating the annealing process, the algorithm explores different resource allocation configurations, allowing it to gradually reach an optimal solution. Through this approach, it aims to strike a balance between accommodating as many workloads as possible while making efficient use of available resources. By considering both workload rejection rates and resource utilization, this method strives to enhance data center efficiency while ensuring a lower rejection rate.

Furthermore, in [1] and [41], a more comprehensive model based on Mixed Integer Linear Programming (MILP) is proposed to address resource allocation challenges in a cluster-scale architecture. The model considers multiple objectives, including the power consumption of the data center (*i.e.*, the total cost of the data center maintenance) and the utilization of individual resources including memory, CPU, and networking modules. By formulating the resource allocation problem as a MILP, the authors can optimize resource allocation across the entire data center infrastructure while considering various constraints and objectives simultaneously. This approach allows for more holistic decision-making, where the trade-offs between power efficiency and resource utilization can be carefully balanced.

In summary, researchers have explored a diverse range of algorithms and models to tackle the resource utilization challenges in data centers. From contention-based algorithms that optimize IT and network resource allocation to simulated annealing techniques that minimize workload rejection rates, each approach contributes to enhancing the efficiency and performance of data centers. Additionally, MILP-based models provide a comprehensive and systematic approach to resource allocation, considering multiple objectives and constraints simultaneously.

Chapter 3

Profiling Datacenter Workloads

In this section, first, we present the datacenter model considered in our work. Then, we focus on developing workload profiles that succinctly capture the impact of resource disaggregation on workload slowdown. Table 6.2 summarizes the main notations used throughout the paper.

3.1 Datacenter Model

Datacenters play a critical role in supporting a wide range of applications and services, from cloud computing to big data analytics. As the demand for data processing and storage continues to grow, datacenter operators are constantly seeking ways to improve efficiency, performance, and resource utilization. One promising approach that has garnered significant attention is resource disaggregation, which involves physically separating computing and memory resources, offering greater flexibility in resource allocation.

In this section, we present the datacenter model considered in our work, with a particular focus on developing a workload model that captures the impact of resource disaggregation on workload slowdown. Understanding the interplay between resource allocation, workload characteristics, and power consumption is essential for optimizing datacenter operations and delivering enhanced performance.

Table 3.1: Important Notations.

Datacenter Notations	
Symbol	Definition
$\mathcal{C}$	Set of all computation-cable modules
F_c	Frequency of CPU cores in module c
N_c	Number of CPU cores in module c
L_c	Amount of local memory in module c
I_c^s	Static power consumption of module c
I_c^d	Dynamic power consumption of module c
M	Total amount of remote memory in DDC

Workload Notations	
Symbol	Definition
$\mathcal{W}$	Set of all workloads
ν_w	Number of cores requested by workload w
ϕ_w	Processor frequency requested by workload w
μ_w	Total amount of memory requested by workload w
$P_w^f(.)$	Model to show the effect of frequency on slowdown of workload w
$P_w^m(.)$	Model to show the effect of memory on slowdown of workload w
Δ_w	Maximum acceptable slowdown of workload w

3.1.1 Disaggregation Model

Generally shown in Fig. 3.1, we consider a physically disaggregated datacenter consisting of two distinct types of modules, each contributing to the overall computing and memory capacity of the system. The first type, denoted by $\mathcal{C}$, represents the computing modules responsible for delivering the processing capacity. Each of these computing modules, denoted by $c \in \mathcal{C}$, is equipped with a specific number of CPU cores, denoted as N_c, and operates at a maximum frequency of F_c. The CPU cores' clock frequency plays a crucial role in determining the processing capacity of these modules which affects the overall performance of our disaggregated datacenter.

Moreover, the computing modules are equipped with a certain amount of local memory, commonly referred to as Dynamic Random-Access Memory (DRAM). This local memory, represented as L_c, is exclusively accessible by the CPU cores residing within the same computing module. The presence of local memory provides a significant advantage in terms of access latency since data can be fetched and stored directly without traversing the datacenter network. This proximity between the CPU cores and their corresponding local memory

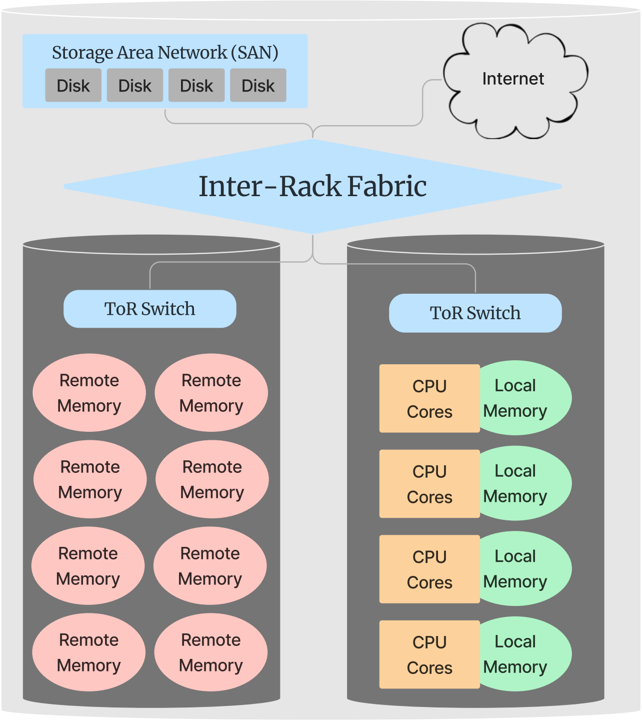


Figure 3.1: Disaggregation model used in this paper.

enhances the performance of memory-intensive workloads.

The second type of module in the datacenter is specialized in providing memory capacity rather than computational power. These memory modules, notated separately, cater to the memory requirements of the computing modules. Unlike the local memory associated with the computing modules, these memory modules can be accessed by any CPU core present within the datacenter through the datacenter network. The total memory capacity offered by these remote memory modules is denoted by M. By efficiently managing the remote memory and its accessibility, the datacenter can effectively balance the overall memory availability and improve memory-intensive workload slowdown.

The remote memory modules, acting as the memory pool for the computing modules, introduce a trade-off in terms of access latency. While local memory enjoys the lowest access latency due to its proximity to the CPU cores, accessing remote memory introduces higher latency as data must traverse the datacenter network. Therefore, the proposed resource allocation algorithm within the datacenter must take into account this higher latency and accurately distribute memory-intensive tasks between local and remote memory to optimize the overall performance. The trade-off happens when the system tries to minimize power consumption by allocating more workloads to the same CPU module (*i.e.*, allocating more remote memory to each of the workloads in order to save local memory for other workloads on the datacenter) which minimizes the power consumption by limiting the number of powered-on CPUs. Fig. 3.2 shows how one CPU module is connected to the local and remote memory modules in our disaggregation model. Based on previous research in the literature [14, 13], the end-to-end latency for CPU to local memory communication is in the order of tens of nanoseconds while remote memory has a 100x higher communication latency which affects the slowdown of workloads.

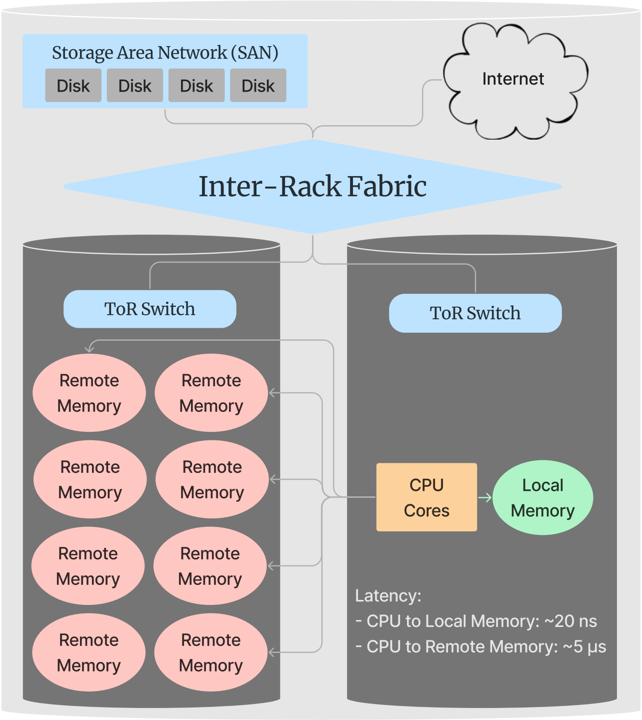


Figure 3.2: Connection between CPU and memory modules.

3.1.2 Power Consumption Model

Previous studies have provided valuable insights into the power consumption patterns within datacenters, revealing that servers are responsible for a significant portion of the overall power usage. Specifically, datacenter servers account for approximately 85% of the total power consumption [45]. Among the various server components, the CPU modules emerge as the primary power consumers, contributing to approximately 90% of the overall server power consumption [45]. In addition, Fig. 3.3 shows that the utilization of networking components such as switches does not significantly impact its power consumption [33]. Therefore, this observation underscores the importance of optimizing the power consumption of CPU modules to achieve significant gains in overall datacenter power efficiency.

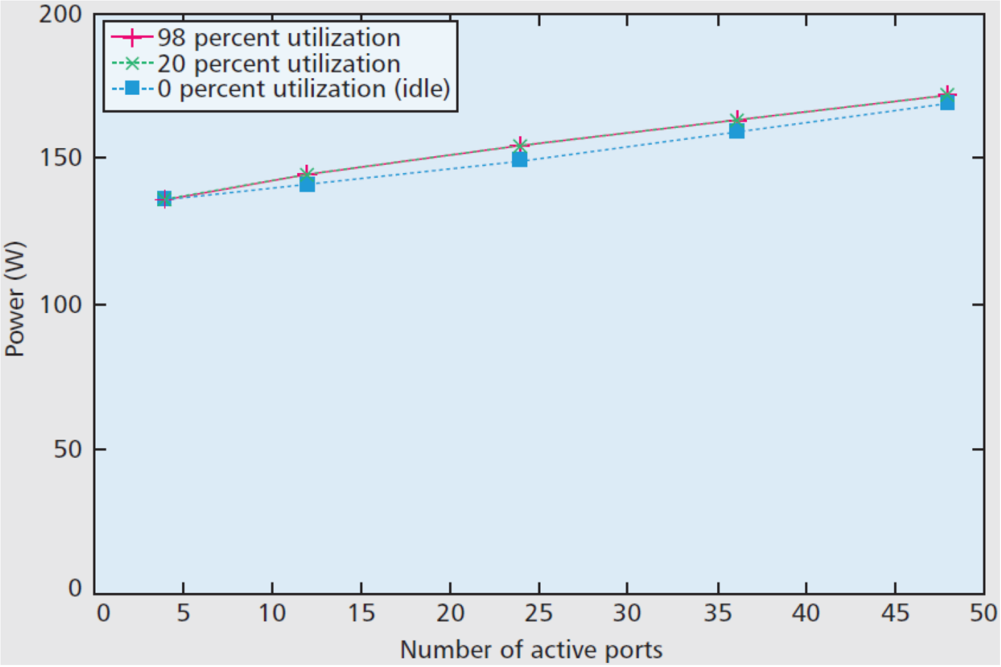


Figure 3.3: Power consumed by a 48-port switch as a function of the load (traffic) through the switch. [33]

Power optimization in datacenters is a multifaceted challenge, primarily due to the interplay of various factors, including resource disaggregation, network utilization, workload characteristics, and more. As mentioned, while addressing power consumption concerns related to network resources is undoubtedly important, the higher potential for power optimization lies within the CPU modules. Thus, in our exploration of resource disaggregation and its impact on datacenter performance, we narrow our focus to the CPU modules as the primary target for power optimization efforts.

To model the power consumption of each CPU module $c \in C$, we take into account both static and dynamic power components. The power consumption, denoted as $E_c(x, y)$, is expressed as a combination of the static power consumption (I_c^s) and the dynamic power consumption (I_c^d) [45], and is formulated by:

$$E_c(x, y) = I_c^s \cdot x + I_c^d \cdot y, \tag{3.1}$$

Here, x is an indicator variable that signifies whether module $c \in C$ is powered off ($x = 0$) or powered on ($x = 1$). The variable y represents the utilization or load of the module, capturing the actual computational activity taking place. Notably, when a CPU module is powered off ($x = 0$), its utilization (y) must also be zero, as no processing occurs in this state and no power is consumed.

The coefficient I_c^s represents the static power consumption when the CPU module is powered on, regardless of its utilization. This component embodies the power consumed by the module even when it is idle or lightly loaded, reflecting the baseline power consumption attributed to the CPU module's operational state. On the other hand, the coefficient I_c^d signifies the maximum dynamic power consumption of the CPU module when operating at full load. At maximum utilization, the CPU module consumes power proportional to I_c^d, on top of the baseline static power consumption.

Considering the maximum power consumption ($E_c(1, 1)$) of a CPU module, which occurs when it operates at full load and is fully powered on, we find that it equals the sum of the static and dynamic power components. Specifically, $E_c(1, 1) = I_c^s + I_c^d$. It is worth noting that the static power consumption (I_c^s) typically constitutes around 75% of the maximum power consumption [45, 3]. This suggests that even during idle or low-load periods, a significant portion of the power is consumed by the CPU module.

By understanding the power consumption model of the CPU modules, we gain valuable insights into their power characteristics, enabling us to develop strategies for optimizing power consumption and enhancing the overall power efficiency of the datacenter. Through thoughtful resource allocation, workload management, and power management policies, we can capitalize on the inherent trade-offs and dynamics of the datacenter environment to achieve substantial power savings while meeting the performance demands of diverse workloads. In the subsequent sections, we delve deeper into the workload characteristics and the interplay of remote memory and CPU frequency on workload slowdown, aiming to optimize the power consumption of the datacenter and ensure efficient resource allocation.

3.1.3 Workload Model

Within our datacenter model, we consider a diverse batch of workloads denoted by $\mathcal{W}$. Each workload $w \in \mathcal{W}$ is characterized by a comprehensive tuple comprising four essential parameters denote by the following:

$$\langle \nu_w, \mu_w, \phi_w, \Delta_w \rangle, \tag{3.2}$$

Understanding these workload characteristics is crucial for effective resource allocation and power optimization within the datacenter environment.

The first parameter, ν_w, denotes the requested number of CPU cores for workload $w \in \mathcal{W}$. Different workloads demand varying amounts of computational cores, and by considering this parameter, we can align the resource allocation to match the specific requirements of each workload. This targeted allocation ensures that workloads receive the necessary CPU resources to execute efficiently while minimizing any potential resource waste.

The second parameter, μ_w, is a critical factor related to the total amount of requested memory by workload $w \in \mathcal{W}$. Workloads have varying memory requirements, and efficiently managing memory resources is essential for optimal datacenter performance. Proper allocation of memory ensures that each workload has access to the necessary memory capacity, avoiding potential performance bottlenecks and contention issues. The ability to properly assign requested memory of each workload according to the available local and remote memory in the datacenter improves the overall workload slowdown in DDCs.

The third parameter, ϕ_w, represents the requested frequency of CPU cores for workload $w \in \mathcal{W}$. CPU frequency significantly impacts workload slowdown, and higher frequencies generally lead to faster execution. However, operating at higher frequencies may also incur higher power consumption. By understanding the frequency preferences of each workload, we can optimize the datacenter power efficiency while maintaining the predefined bounded threshold of each workload slowdown which is detailed next.

Lastly, the parameter Δ_w plays a crucial role in workload slowdown management. This parameter represents the maximum allowable slowdown that workload $w \in \mathcal{W}$ is willing to tolerate as part of its service level agreement (SLA). Workload slowdown occurs when the completion time of a workload is adversely affected due to the resource contention, limited CPU frequency, or remote memory access delays. By allowing workloads to specify their tolerance for slowdowns, the datacenter can tailor resource allocation strategies to meet individual workload requirements.

In our datacenter model, we take into account the diverse nature of workloads found in modern datacenters. We understand that these workloads often come with different priorities and performance expectations. Consequently, our datacenter operator has devised a strategy that offers incentives to workloads capable of accommodating a certain degree of slowdown. For instance, the operator might provide lower pricing or other advantages to these adaptable workloads, encouraging them to be more flexible in terms of their performance requirements. This approach allows the datacenter to strike a balance between optimizing its power consumption and efficiently utilizing the available resources, all while ensuring that the distinctive needs of each workload are catered to.

In summary, by characterizing the workloads with the tuple $\langle \nu_w, \phi_w, \mu_w, \Delta_w \rangle$, we equip ourselves with valuable insights into the resource demands, performance preferences, and flexibility thresholds of each workload. Leveraging this knowledge, we can devise intelligent

Table 3.2: Profiled Workloads.

Workload	Implementation	Dataset Size	Memory Usage
Kmeans	Python-TensorFlow [50]	200MB	2GB
WordCount	Java SE [42]	10GB	8GB
HPCG	C++ [22]	1GB	12GB

resource management strategies and power optimization techniques within the datacenter. In the subsequent sections, we delve deeper into the impact of remote memory and CPU frequency on workload slowdown, aiming to maximize datacenter efficiency and ensure that workloads receive the optimal resource allocation that aligns with their individual slowdown and resource expectations.

3.2 Evaluation of Slowdown Characterizations

In this section, we delve into the comprehensive analysis of workload slowdown within our datacenter model. We assume that workloads are versatile and can effectively function with CPU cores operating at frequencies different from their requested ϕ_w. Additionally, they can adapt to varying amounts of both local and remote memory, as long as the total memory received, matches their memory requirement, denoted by μ_w. The datacenter employs modern operating systems equipped with mechanisms to seamlessly handle heterogeneous memory latency, leveraging the support for Non-Uniform Memory Access (NUMA).

To optimize the datacenter's performance and resource utilization, we aim to characterize the effect of remote memory and CPU frequency on the slowdown experienced by various workloads. To achieve this, we embark on a profiling journey where we meticulously examine the behavior of three widely used and popular datacenter workloads. These workloads include (1) K-means clustering [50], (2) WordCount on a sizable 10 GB dataset obtained from web crawling of Project Gutenberg [42], and (3) the High-Performance Conjugate Gradient (HPCG) benchmark [22]. Each workload has distinct computational requirements and memory usage patterns, making them suitable candidates for our investigation.

In the profiling process, we measure the slowdown characteristics of these workloads by executing them under various configurations. For this purpose, we utilize a computer with a well-defined hardware setup, consisting of 16 GB of DDR4 RAM, an Intel(R) Core(TM) i9-12700 processor running at 2.4 GHz, and operating Linux Ubuntu 22.04 LTS. This well-controlled environment ensures that we can precisely monitor and analyze the impact of different parameters on workload slowdown.

One of the main objectives of our research is to enable the datacenter operator to make informed decisions regarding resource allocation and workload placement. To achieve this, we consider two strategies for workload profiling and slowdown characterization.

The first approach is a proactive profile-building method, where the datacenter operator generates profiles for popular workloads in an offline manner. This proactive strategy is particularly relevant for large-scale workloads that benefit significantly from the advantages of disaggregated memory. By analyzing the behavior of these workloads under various scenarios, the datacenter operator can gain valuable insights into their resource demands, frequency adaptability, and sensitivity to remote memory. Armed with this knowledge, the operator can optimize resource allocation and design effective techniques to enhance overall datacenter performance.

On the other hand, we also mention a passive profile construction strategy, which leverages runtime management for workload characterization. In this scenario, when a new workload arrives at the datacenter without an existing profile, it is initially treated with zero tolerance during the workload placement phase. Subsequently, the runtime management system diligently attempts to identify an appropriate resource configuration for the workload while concurrently constructing a profile during the process. This approach allows the datacenter to continuously learn and adapt to the unique requirements of each workload, ensuring optimal resource allocation and performance enhancement.

In the following sections, we present the results of our slowdown characterization for each of the three popular datacenter workloads: K-means clustering, WordCount, and HPCG benchmark. Through detailed analyses and modeling, we aim to provide valuable insights into the effect of remote memory and CPU frequency on workload slowdown, empowering

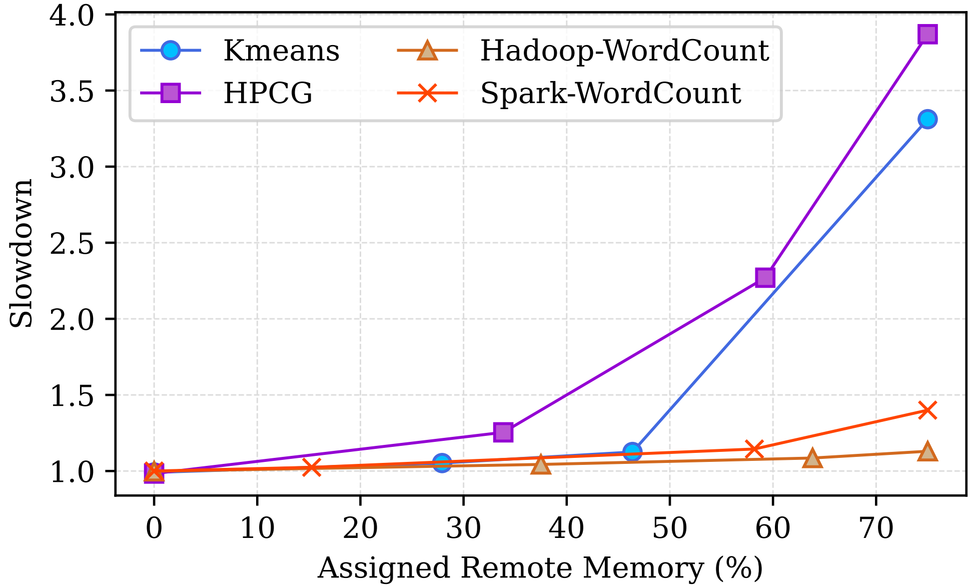


Figure 3.4: Workload slowdown as the assigned remote memory ratio increases. The injected latency for remote memory is fixed at 5 μs.

the datacenter operator to make well-informed decisions and optimize datacenter efficiency.

3.2.1 Effect of Remote Memory

In this subsection, we delve into the profound impact of memory configuration on workload slowdown within our datacenter model. To accurately characterize the influence of remote memory, we perform a series of meticulously designed experiments, systematically varying the ratio of remote to local memory for each workload. To simulate local and remote memories, we employ the powerful *Ramdisk* feature in Linux, allowing us to designate a portion of the local memory as a disk representing remote memory. Additionally, to emulate the network latency experienced when accessing remote memory in DDCs, we implement a carefully crafted artificial delay during major page swaps by modifying the page swap procedure in Linux Kernel version 6.1, which is the latest version of the kernel today.

In each experiment, we introduce an artificial delay of 5 μs as the remote memory latency and meticulously measure the completion time of each workload using the *time* library in the Linux Kernel. To ensure the robustness and reliability of the results, each workload's completion time is measured 10 times, and the averaged outcomes are presented in Fig. 3.4.

As expected, the results clearly demonstrate that as the ratio of remote memory is increased from zero to 75%, the slowdown experienced by the workloads also increases. However, intriguingly, different workloads exhibit varying levels of sensitivity to remote memory. Specifically, K-Means clustering and HPCG benchmark workloads experience slowdowns of approximately 4x and 3.5x, respectively, when confronted with 75% remote memory allocation. Conversely, the variations of WordCount encounter a relatively minor slowdown of less than 1.5x under the same conditions. This observation highlights the importance of workload-specific memory allocation strategies in optimizing datacenter performance.

To effectively model and understand the observed slowdown patterns, we explore different regression approaches. Among them, we find that a piece-wise linear function provides an accurate approximation of the impact of remote memory on workload slowdown. This finding is further validated through the fitting of piece-wise functions with one, two, and three segments to the data points of each workload. The results indicate that a three-piece linear function yields the most precise estimation of the slowdown behavior. Table 3.3 showcases the Mean Squared Errors (MSE) of all three types of linear functions, including the average, minimum, and maximum values, underscoring the superiority of the three-piece linear model.

For the sake of conciseness and clarity, we introduce the function $P_w^m(r)$ as the piece-wise linear approximation of the impact of remote memory on workload slowdown. To mitigate significant workload slowdown and maintain datacenter efficiency, we enforce a minimum of 25% local memory allocation, following similar strategies adopted in previous studies and researches [29, 13]. Leveraging three segments with breakpoints at 25%, 50%, and 75% for the ratio of remote memory, $P_w^m(r)$ is mathematically expressed as follows:

$$P_w^m(r) = \begin{cases} a_1 \cdot r + c_1 & 0 \le r \le 0.25 \\ a_2 \cdot r + c_2 & 0.25 \le r \le 0.5 \\ a_3 \cdot r + c_3 & 0.5 \le r \le 0.75, \end{cases} \tag{3.3}$$

Table 3.3: MSE of Piece-Wise Linear Approximation for the Impact of Remote Memory Model.

Application	Function Type	Error		
		Min	Avg	Max
Kmeans	1-Segmented	8.3	36.6	61.7
	2-Segmented	0.2	1.8	4.3
	3-Segmented	10^{-7}	1.7	4.3
HPCG	1-Segmented	7.6	35	68
	2-Segmented	2.9	7.6	19.0
	3-Segmented	10^{-8}	0.9	3.3
Spark-WordCount	1-Segmented	1.1	4.4	9.6
	2-Segmented	10^{-7}	0.3	0.6
	3-Segmented	10^{-9}	0.09	0.3
Hadoop-WordCount	1-Segmented	0.7	0.8	1.7
	2-Segmented	0.1	0.4	1.3
	3-Segmented	10^{-8}	0.2	0.6

Table 3.4: MSE of Linear Approximation for CPU Frequency Effect.

Application	Error		
	Min	Avg	Max
Kmeans	0.8	1.7	2.4
HPCG	1.0	1.6	2.0
Spark-WordCount	0.9	1.5	2.1
Hadoop-WordCount	0.8	1.6	2.0

where r represents the percentage of total requested memory allocated remotely for the workload, while a_i and c_i are workload-specific coefficients obtained from profiling. Importantly, when no remote memory is assigned to a workload ($r = 0$), we anticipate zero slowdown, allowing us to set c_1 to zero. This detailed piece-wise linear model provides the datacenter operator with invaluable insights into the intricate relationship between memory configuration and workload slowdown, enabling smart decisions for efficient resource allocation.

3.2.2 Effect of CPU Frequency

The CPU frequency serves as a critical factor in determining the performance and execution speed of workloads within our data center model. To gain a deeper understanding of

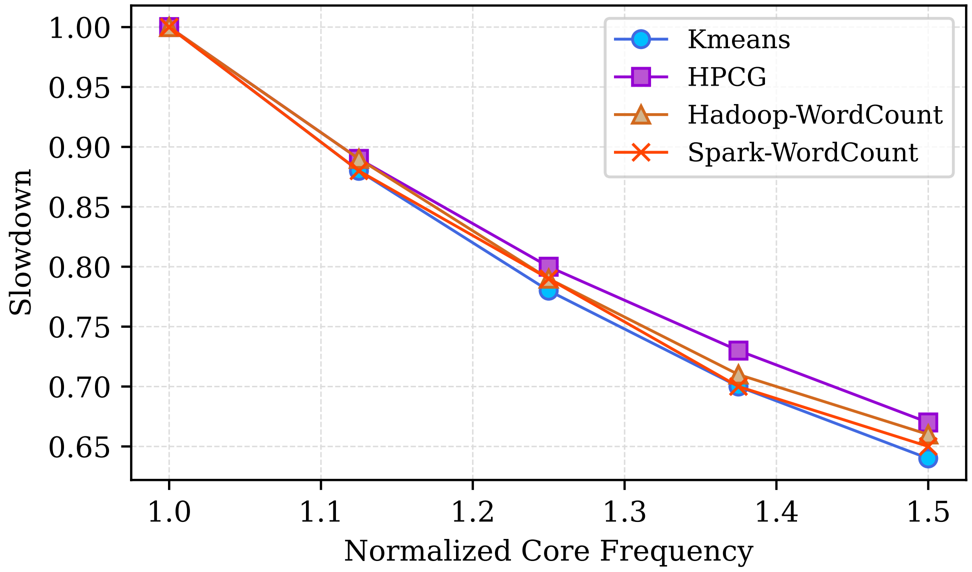


Figure 3.5: Workload slowdown as the assigned CPU frequency increases. The lower slowdown means faster execution.

the impact of CPU frequency on workload slowdown, we conducted a series of experiments. These experiments involved measuring the completion time of various workloads at different CPU frequencies, ranging from the base frequency of 1.6 GHz to the maximum of 2.4 GHz. The frequency adjustments were dynamically made using the powerful Linux Kernel module *cpupower*, enabling precise control over CPU frequencies which is called *CPU scaling*. The results of our comprehensive experiments, as depicted in Fig. 3.5, revealed an intriguing trend: the slowdown of workloads displayed a nearly linear decrease as the CPU frequency was increased from 1.0 to 1.5 times the originally requested frequency. This observation suggests that by appropriately allocating CPU modules with higher frequency rates than initially requested, significant improvements in workload performance and mitigation of slowdown can be achieved.

To gain further insights into the relationship between CPU capacity and workload slowdown, we performed regression analysis on the experimental data. For each workload, we fitted a linear function to the data points obtained from varying CPU frequencies and calculated the Mean Squared Error (MSE) of the linear approximation. The results of the regression analysis, presented in Table 3.4, demonstrate the reasonable level of accuracy achieved by using a linear function to model the impact of CPU frequency on workload slowdown.

Building upon the observations and regression analysis, we propose the function $P_w^f(x)$ to quantitatively measure the effect of CPU frequency on workload w. The function $P_w^f(x)$ is defined as follows:

$$P_w^f(x) = \frac{\phi_w}{x}, \tag{3.4}$$

where ϕ_w represents the originally requested CPU frequency by workload $w \in \mathcal{W}$, and x denotes the received frequency. This function offers a valuable tool for data center operators to accurately assess the impact of varying CPU frequencies on individual workloads, facilitating informed decisions for workload placement and resource allocation.

In summary, by thoroughly analyzing the effects of both remote memory access latency and CPU frequency on workload slowdown, our comprehensive profiling of popular data center workloads empowers data center operators to optimize resource utilization and enhance overall data center performance. The generated profiles provide a powerful basis for proactive decision-making, enabling the data center to offer tailored service-level agreements (SLAs) that align with the unique requirements of diverse workloads.

Chapter 4

Workload Placement Problem Formulation

4.1 Workload Placement

Table 4.1: Decision Variables.

Symbol	Definition
z_c	Activation of module c
$y_{w,c}$	Assignment of workload w to module c
x_w	Fraction of local to total requested memory of workload w
f_w	Frequency of allocated CPU to workload w
u_c	CPU utilization of module c
$\widetilde{N}_c$	Number of allocated cores of module c
$\tilde{L}_c$	Amount of allocated local memory of module c

Let $\mathcal{W}$ denote the set of workloads that are planned for deployment. We introduce the binary decision variable $y_{w,c}$ to indicate whether workload w is assigned to CPU module c. Previous studies such as [54, 4, 45, 39] have demonstrated that current networking technologies cannot fulfill the network requirements for CPU-to-CPU communications. This fact also is mentioned in Table 2.1, where the CPU-CPU communication requirement is measured in the order of less than 10 ns which can not be met using the current networking technologies. Therefore, in DWLP, workloads are assigned to a single CPU module with a sufficient number of cores. Constraint (4.1b) ensures that each workload is assigned to exactly one CPU module. It is worth noting that the mentioned assumption is not limiting the flexibility of our proposed placement algorithm as today's workloads are mainly executed in one single CPU module and they don't need to be assigned to different CPUs in the placement phase.

Let the decision variable $x_w \in [0, 1]$ represent the proportion of the memory requirement of workload w that is allocated locally. Constraints (4.1c)-(4.1f) ensure that the processing and local memory loads of all assigned workloads to a CPU module comply with its capacity. Here, z_c is an indicator variable that shows whether processing module $c \in \mathcal{C}$ is powered on or off. A workload can only be assigned to powered-on modules. Recall that ν_w and μ_w denote the number of CPU cores and the total amount of memory requested by workload w, respectively.

The constraint in (4.1e) ensures that the total number of cores assigned to the workloads in processing module $c \in \mathcal{C}$ does not exceed its total number of cores. Equation (4.1f) serves a similar purpose but for the local memory capacity of processing module $c \in \mathcal{C}$. Constraint (4.1g) ensures that the total remote memory assigned to workloads does not exceed the capacity of remote memory modules.

In the optimization problem DWLP, we aim to minimize the objective function (4.1a). This objective function considers two components: the first part aims to minimize the weighted sum of the fractions of allocated CPU resources for each workload, where θ_w represents the fraction of CPU resources allocated to workload $w \in \mathcal{W}$; the second part aims to minimize the power consumption of the active CPU modules, where η is a parameter that controls the trade-off between workload slowdown and power consumption. The constraint in (4.1b) ensures that each workload is assigned to exactly one CPU module, and constraints (4.1c) and (4.1d) guarantee that the processing and memory loads of each CPU module do not exceed their capacities. Constraints (4.1e) and (4.1f) introduce the binary variable z_c, which indicates whether a CPU module $c \in \mathcal{C}$ is activated or turned off. This ensures that workloads can only be assigned to activated CPU modules. Additionally, constraint (4.1g) limits the total remote memory assigned to workloads based on the capacity

Problem 1: DWLP: Disaggregated Workload Placement.

$$\text{Min. } (1-\eta)\cdot\sum_{w\in\mathcal{W}}\theta_w + \eta\cdot\sum_{c\in\mathcal{C}}E_c(z_c,u_c) \tag{4.1a}$$

$$\text{s.t. } \sum_{c\in\mathcal{C}} y_{w,c} = 1, \qquad \forall w\in\mathcal{W} \tag{4.1b}$$

$$\sum_{w\in\mathcal{W}} y_{w,c}\cdot\nu_w \le \widetilde{N}_c, \qquad \forall c\in\mathcal{C} \tag{4.1c}$$

$$\sum_{w\in\mathcal{W}} y_{w,c}\cdot x_w\cdot\mu_w \le \widetilde{L}_c, \qquad \forall c\in\mathcal{C} \tag{4.1d}$$

$$\widetilde{N}_c \le N_c\cdot z_c, \qquad \forall c\in\mathcal{C} \tag{4.1e}$$

$$\widetilde{L}_c \le L_c\cdot z_c, \qquad \forall c\in\mathcal{C} \tag{4.1f}$$

$$\sum_{w\in\mathcal{W}}(1-x_w)\cdot\mu_w \le M, \tag{4.1g}$$

$$0.25 \le x_w, \qquad \forall w\in\mathcal{W} \tag{4.1h}$$

$$\theta_w = \sum_{c\in\mathcal{C}} y_{w,c}\big(P_w^m(1-x_w)\cdot P_w^f(F_c)\big), \forall w\in\mathcal{W} \tag{4.1i}$$

$$\theta_w \le \Delta_w, \qquad \forall w\in\mathcal{W} \tag{4.1j}$$

$$u_c = \frac{\widetilde{N}_c}{N_c}, \qquad \forall c\in\mathcal{C} \tag{4.1k}$$

$$z_c\in\{0,1\}, y_{w,c}\in\{0,1\}, x_w \le 1\,. \tag{4.1l}$$

of remote memory modules. Constraint (4.1h) restricts the values of x_w to be greater than or equal to 0.25 for all workloads. By using this constraint, we ensure that each workload at least gets some amount of local memory to avoid high and unpredictable workload slowdowns. The variable u_c in constraint (4.1k) represents the CPU utilization of module $c\in\mathcal{C}$.

Overall, the DWLP addresses the placement of workloads on different CPU modules while considering their memory requirements and processing capabilities. The objective is to find an assignment that minimizes both the fraction of allocated CPU resources and the power consumption of the active CPU modules, ensuring that all assigned workloads fit within the capacity of their respective modules.

4.2 Workload Slowdown

Workload slowdown is a critical metric that directly impacts the overall system performance. It is influenced by various factors, including remote memory access latency and the allocated CPU frequency. To optimize the system's performance, it is essential to carefully manage the slowdown experienced by each workload.

In our optimization framework, we adopt a holistic approach to address the workload slowdown problem. First and foremost, we aim to reduce the impact of remote memory access latency on workload slowdown. To achieve this, we introduce Constraint (4.1h), requiring that each workload must receive at least 25% of its requested memory allocation locally. The mentioned constraint is designed to minimize the reliance on remote memory modules, which tend to have higher access latency compared to local memory.

To capture the effect of resource allocation decisions on workload slowdown, we introduce a novel variable, denoted as θ_w, for each workload $w \in \mathcal{W}$. The variable θ_w quantifies the total change in slowdown experienced by workload $w \in \mathcal{W}$ due to the allocation decisions made during the workload placement process. Constraint (4.1i) plays a central role in optimizing the value of θ_w for each workload.

The calculation of θ_w relies on two crucial functions, $P_w^m(\cdot)$ and $P_w^f(\cdot)$, as given in Constraint (4.1i). These functions capture the intricate relationship between memory allocation and CPU frequency on the slowdown of a workload. More precisely, $P_w^m(1 - x_w)$ represents the impact of memory allocation on workload w. When a fraction $(1 - x_w)$ of its total requested memory is allocated from remote memory modules, it can lead to increased access latencies and, consequently, a higher slowdown for the workload.

Similarly, $P_w^f(F_c)$ represents the effect of CPU frequency on the workload slowdown. It captures how the slowdown (or speedup) of a workload is influenced by the CPU processing capacity F_c of the allocated processing module. The significance of this function lies in optimizing the assignment of workloads to processing modules to maximize overall system performance.

While evaluating the slowdown, it is crucial to consider the specific requirements and performance constraints of each workload. To this end, we assume that each workload has a maximum acceptable slowdown, denoted as Δ_w. This value serves as a threshold beyond which the slowdown of the workload is considered unacceptable. Therefore, Constraint (4.1j) ensures that the final slowdown of each workload does not exceed this predefined threshold.

Our optimization problem's primary aim is to minimize the collective slowdown across all workloads while upholding memory allocation constraints and considering the impact of CPU frequency on workload slowdown. By introducing slowdown-related variables and constraints, our approach presents a comprehensive and efficient workload placement strategy. This strategy optimally balances system resources (*i.e.*, power consumption of CPU modules which is detailed next) and workload requirements, ultimately leading to improved overall system efficiency and a better user experience.

4.3 Power Consumption

Managing power consumption is a critical aspect of running data centers efficiently, both in terms of cost-effectiveness and environmental sustainability. In this section, we try to dive into how we model the dynamic power consumption of CPU modules within our disaggregated datacenter (DDC).

To get started, we use the concept of CPU utilization to estimate how much power CPU modules consume. CPU utilization for a processing module $c \in \mathcal{C}$ is defined as the ratio of active cores within that module to the total number of available cores. This is expressed mathematically as u_c and is governed by Constraint (4.1k). Understanding this utilization value provides us with valuable insights into how efficiently each CPU module is using its resources.

With the utilization information, we can now compute the power consumption of each

CPU module. The power consumption model used in our optimization framework is based on the power model as represented by Equation (3.1). The power model characterizes the relationship between the CPU frequency, the number of active cores, and the power consumption.

The total power consumption of the entire disaggregated data center, denoted as E_{DDC}, is determined by summing the power consumed by each individual CPU module. The expression for E_{DDC} is formulated as follows:

$$E_{DDC} = \sum_{c \in \mathcal{C}} (I_c^s + I_c^d \times \frac{\sum_{j \in \mathcal{J}} w_c^j \times N_j}{A_c}) \,. \tag{4.2}$$

In this equation, I_c^s and I_c^d represent the static and dynamic power consumption of processing module $c \in \mathcal{C}$, respectively. The static power consumption refers to the power consumed by a CPU module when it is idle or powered on but no workload is assigned to it. On the other hand, dynamic power consumption corresponds to the power consumed while actively executing tasks. The term $\sum_{j \in \mathcal{J}} w_c^j \times N_j$ in the denominator of the dynamic power component accounts for the total number of active cores across all workloads assigned to module $c \in \mathcal{C}$. The factor $\frac{\sum_{j \in \mathcal{J}} w_c^j \times N_j}{A_c}$ represents the average utilization of module $c \in \mathcal{C}$, which scales the dynamic power component.

In addition to the bounded slowdowns of workloads, it is essential to highlight a key aspect of our model that enables enhanced efficiency and simplicity in workload placement. Our model allows for the possibility of each CPU module simultaneously serving different workloads, effectively leveraging the potential of multitasking within the datacenter. However, we recall the fundamental assumption that each workload can be entirely processed by a single CPU module, thereby neglecting any considerations for CPU-to-CPU communication in this specific scenario.

This assumption brings several advantages to the overall optimization process. By isolating each workload to a single CPU module, we simplify resource allocation and avoid the complexities that would arise from managing and coordinating communication between multiple CPU modules. This streamlined approach significantly reduces the computational overhead, allowing our algorithm IRoP to achieve its efficiency in run-time complexity.

Moreover, the single-CPU-per-job assumption aligns well with literature [3, 4, 13] and many real-world applications and workloads, where tasks are often designed to be independent and can be efficiently executed by a single processor core. This allows datacenter operators to tailor the allocation of resources for individual workloads while optimizing the overall performance and power consumption of the system.

By incorporating the power consumption model into our optimization framework, we can make informed decisions on workload placement and resource allocation to minimize overall power consumption, effectively promoting power efficiency and sustainable data center operations. Properly managing power consumption not only reduces operational costs but also contributes to environmental conservation by optimizing resource usage and reducing carbon footprints.

4.4 Optimization Objective

The optimization objective plays a crucial role in guiding our workload placement algorithm within the disaggregated data center (DDC). It essentially helps us make strategic decisions when it comes to resource allocation.

This objective function, outlined in (4.1a), comprises two distinct components, each addressing vital aspects of data center performance. The first component is all about quantifying the total slowdown experienced by the workloads within the data center. As we have discussed earlier, slowdown is a critical metric that has a direct impact on the quality of service delivered to our users and, consequently, the overall user experience. By minimizing this total slowdown, our data center can ensure the efficient and timely execution of tasks, meeting the performance expectations of our clients.

On the other hand, the second component of the objective function focuses on the total power consumed by the active CPU modules in the DDC. Power consumption is a significant concern in modern data centers, and optimizing power usage is vital for achieving cost-effectiveness and environmental sustainability. By minimizing the total power consumption, the data center can operate more efficiently, leading to reduced operational costs and reduced environmental impact.

It is important to recognize the inherent trade-off between these two components of the objective function. As coefficient $0 \leq \eta \leq 1$ is introduced to the objective function, it allows for adjusting the relative importance of the two components. When η is set to 0, the optimization prioritizes minimizing the total slowdown of workloads, neglecting the impact on power consumption. Conversely, when η is set to 1, the optimization focuses solely on minimizing power consumption, disregarding the impact on workload slowdown. Intermediate values of η allow for finding a balance between these two competing objectives, striking an optimal trade-off between workload slowdown and power consumption efficiency.

While Constraint (4.1j) guarantees that the maximum slowdown of each workload is within the acceptable Service Level Agreement (SLA) requirements ensuring that each workload is maintained within acceptable limits, the inclusion of the total slowdown component in the objective function serves a critical purpose. It provides the workload placement algorithm with the flexibility to further improve the slowdown of workloads, beyond the minimum SLA requirements, when such improvements do not disproportionately impact the power costs of the DDC. For instance, in situations where power consumption is of lesser importance compared to the overall workload slowdown, the algorithm can opt to reduce workload slowdowns even further, optimizing user experience and service quality. Several workloads in today's datacenters need a timely execution and they can not tolerate workload slowdowns [46]. In these scenarios, the administrator of the datacenter can aggressively adjust the parameters in order to optimize the total slowdown of workloads for a specific period of time without considering the power consumption of the DDC until the real-time latency-sensitive workloads are completed executing.

The flexibility afforded by the proposed objective function allows the optimization algorithm to adapt its decision-making process based on varying priorities and requirements.

This adaptability enables the DDC owner to cater to diverse scenarios and operational goals, ensuring that resource allocation decisions align with the data center's specific needs and objectives. Ultimately, the optimization objective empowers the workload placement algorithm to strike an optimal balance between workload slowdown and power efficiency, paving the way for a highly efficient and sustainable disaggregated data center.

Chapter 5

Proposed Algorithm for Workload Placement

The DWLP presents a challenging task involving the joint allocation of memory and CPU frequency, making it inherently difficult to solve. To tackle this complexity, we introduce a key idea that simplifies the problem significantly. Our approach is rooted in the observation that when a workload's achieved slowdown precisely matches its slowdown requirement, denoted as Δ_j, the workload slowdown needs are fully satisfied. Leveraging this insight, we transform DWLP into a more manageable variant, named *DWLP with Fixed Memory Ratios* (F-DWLP). By doing so, we shift our focus to solving F-DWLP as a proxy for the original DWLP problem.

Although F-DWLP is relatively simpler to solve compared to DWLP, it remains a challenging extension of the multi-dimensional bin-packing problem [8], which is a well-known NP-complete problem. Consequently, devising an exact solution for F-DWLP is computationally infeasible, necessitating the development of an efficient approximation algorithm to obtain a feasible solution within polynomial time.

In response to this challenge, we propose a novel approximation algorithm, aptly named *Iterative Rounding-based Placement* (IRoP). The algorithm builds upon the deterministic rounding framework [55], a powerful technique for obtaining approximation solutions with provable guarantees. By strategically applying iterative rounding techniques, IRoP provides a computationally efficient and effective solution to the placement problem, yielding results that are close to the optimal solution.

The key strength of IRoP lies in its ability to strike a balance between computational complexity and solution quality (*i.e.*, results should be near-optimal). While the algorithm may not guarantee an exact optimal solution, its provable approximation guarantees ensure that the solution obtained is of high quality and close to the optimal solution. This feature makes IRoP an excellent choice for large-scale real-world scenarios where finding an exact solution is intractable, and an approximate but high-quality solution is more practical and actionable.

By utilizing IRoP to solve the transformed F-DWLP, we achieve significant progress towards tackling the original DWLP problem. The successful application of the deterministic rounding framework and the innovative iterative rounding techniques employed in IRoP allow us to handle the memory and CPU frequency allocation challenges efficiently, significantly enhancing the data center's performance and resource utilization.

In the way of solving the DWLP problem, the subsequent subsections present a comprehensive and systematic exploration of our approach's key components. In the first two subsections, we delve into the intricacies of the algorithm, with each subsection dedicated to a specific phase that advances us closer to achieving an efficient and effective solution. In the first subsection, *Phase 1: Fixing Memory Ratios*, we elaborate on the process of transforming the original DWLP problem into the simplified variant called F-DWLP. By meticulously fixing the memory ratios for the workloads, we successfully streamline the problem complexity, setting the stage for the subsequent steps of the algorithm. In the second subsection, aptly titled *Phase 2: Workload Placement Phase*, we shift our focus to the crux of the workload placement problem. Here, we harness the power of the approximation algorithm IRoP to conduct an optimized and strategic allocation of workloads to CPU modules. The algorithm adeptly takes into account both the fixed memory ratios and the intricate relationship between CPU frequency and workload slowdown, offering a detailed account of IRoP's iterative rounding approach and its vital role in obtaining a feasible solution for F-DWLP.

In the final subsection, *Algorithm Analysis*, we embark on a comprehensive theoretical journey that illuminates the performance and characteristics of our innovative approach. This critical phase involves an in-depth examination of the algorithm's theoretical underpinnings, including the derivation and analysis of its approximation ratio and run-time complexity. Through theoretical evaluations, we elucidate the algorithm's scalability, solution quality, and computational tractability, thereby gaining a profound understanding of its practicality and effectiveness in real-world scenarios. Furthermore, we compare the results obtained by IRoP with established lower bounds to ascertain the quality of the solutions it produces and the degree to which it approximates the optimal solution. This detailed theoretical analysis provides valuable insights into its strengths, limitations, and potential for broad applicability in addressing the resource allocation challenges faced by modern data centers.

In summary, by dissecting each phase of our approach and complementing it with comprehensive theoretical analysis, we present a holistic and robust methodology for resolving the intricate resource allocation problem in the disaggregated data center. Armed with a deeper understanding of the algorithm's inner workings and theoretical guarantees, we can confidently assert its potential to transform data center operations and optimize resource utilization, ultimately fostering a more intelligent, adaptive, and high-performance data center infrastructure. The knowledge and insights gained from these subsequent subsections pave the way for efficient and sustainable data center management, empowering data center operators to meet the diverse demands of workloads and deliver exceptional performance while minimizing power consumption and adhering to critical performance guarantees. Finally, the proposed algorithm is evaluated in the next chapter to measure the effectiveness of IRoP as well as the accuracy of the theoretical analysis.

5.1 Fixing Memory Ratios

To transform DWLP into F-DWLP, we embark on a critical step that involves computing the minimum local memory ratio for each workload $w \in \mathcal{W}$ and every computing module $c \in \mathcal{C}$. The objective is to determine the local memory allocation strategy that is sufficient to satisfy the slowdown constraint of each workload under any CPU allocation strategy determined by $y_{w,c}$. The computation of this ratio follows a meticulous process, outlined as follows:

$$x_{w,c} \leftarrow \max \left\{ 1 - (P_w^m)^{-1}\Big(\frac{\Delta_w}{P_w^f(F_c)}\Big), 0.25 \right\}, \tag{5.1}$$

Here, the memory slowdown function, denoted as $(P_w^m)^{-1}(\cdot)$, is linear and easily computed. This function is proposed and evaluated in Subsection 3.2.1 and it is a piece-wise approximation for the impact of the remote memory assignment ratio on the workload slowdown. A notable aspect is that we guarantee the allocated local memory ratio to be no less than 25%, as mandated by Constraint (4.1h) in the original DWLP problem formulation. By considering the minimum local memory assignment, we retain a carefully selected subset of placement options where the slowdown of workloads can be better than Δ_w since we can assign workloads to a higher-performing CPU module. This particularly comes into play with powerful CPUs, i.e., modules possessing higher processing capacity, that require only a small amount of local memory to satisfy slowdown constraints. In such scenarios, it becomes possible to place workload $w \in \mathcal{W}$ on a more powerful CPU, yielding a slowdown that is less than what was acceptable by the workload (*i.e.*, Δ_w) in exchange for consuming more power).

To reflect the updated memory allocation strategy, we introduce Constraints (5.2g), (5.2h), and (5.2i) in the transformed version of DWLP, denoted as F-DWLP. These constraints replace the original Constraints (4.1d), (4.1g), and (4.1i), respectively, with the computed value $x_{w,c}$ in place of the variable x_w. Constraint (5.2g) ensures that the sum of local memory usage of workloads assigned to the same module respects the local memory capacity of that module. Similarly, Constraint (5.2h) enforces the capacity constraint for the remote memory in the DDC, utilizing the constant value $x_{w,c}$ and variable $y_{w,c}$ to compute the remote memory of workloads based on their assigned CPU modules. Lastly, Constraint (5.2i) plays a crucial role in computing workload slowdowns, wherein $x_{w,c}$ remains constant, and the only variable is $y_{w,c}$.

Problem 2: F-DWLP: DWLP with Fixed Memory Ratios.

$$\text{Min. } (1-\eta)\cdot\sum_{w\in\mathcal{W}}\theta_w+\eta\cdot\sum_{c\in\mathcal{C}}E_c(z_c,u_c) \tag{5.2a}$$

$$\text{s.t. } \sum_{c\in\mathcal{C}}y_{w,c}=1, \qquad \forall w\in\mathcal{W} \tag{5.2b}$$

$$\sum_{w\in\mathcal{W}}y_{w,c}\cdot\nu_w\le\widetilde{N}_c, \qquad \forall c\in\mathcal{C} \tag{5.2c}$$

$$\widetilde{N}_c\le N_c\cdot z_c, \qquad \forall c\in\mathcal{C} \tag{5.2d}$$

$$\widetilde{L}_c\le L_c\cdot z_c, \qquad \forall c\in\mathcal{C} \tag{5.2e}$$

$$u_c=\frac{\widetilde{N}_c}{N_c}, \qquad \forall c\in\mathcal{C} \tag{5.2f}$$

$$\sum_{w\in\mathcal{W}}y_{w,c}\cdot x_{w,m}\cdot\mu_w\le\widetilde{L}_c, \qquad \forall c\in\mathcal{C} \tag{5.2g}$$

$$\sum_{w\in\mathcal{W}}\sum_{c\in\mathcal{C}}y_{w,c}\cdot(1-x_{w,c})\cdot\mu_w\le M, \tag{5.2h}$$

$$\theta_w=\sum_{c\in\mathcal{C}}y_{w,c}(P^m_w(1-x_{w,c})\cdot P^f_w(F_c)), \forall w\in\mathcal{W} \tag{5.2i}$$

By adopting this innovative approach of fixing memory ratios and formulating F-DWLP, we simplify the resource allocation challenge while preserving crucial performance guarantees. This transformation empowers our approximation algorithm IRoP to effectively navigate the multi-dimensional bin-packing problem, optimizing the allocation of workloads and CPU modules. The careful consideration of memory and CPU frequency allocation, coupled with the iterative rounding techniques of IRoP, sets the stage for an efficient and provably good-quality near-optimal solution for the DWLP problem, enabling data center owners to achieve enhanced performance, reduced power consumption, and adaptable workload management.

5.2 Workload Placement Phase

Algorithm 1 presents the pseudocode of IRoP, which employs the deterministic rounding framework to solve F-DWLP. The algorithm takes an instance of the problem, denoted by $\mathcal{M}$, and starts its procedure by relaxing the integrality constraints in line 1. This relaxation transforms F-DWLP into a linear program (LP), denoted by $\widetilde{\mathcal{M}}$, that can be efficiently solved using interior point methods. After solving $\widetilde{\mathcal{M}}$, IRoP obtains fractional values for $y_{w,c}$ and z_c that respect all constraints except for the integrality constraints (*i.e.*, the decision variables have fractional values rather than achieving binary - 0 or 1 - results). We denote the fractional value of these decision variables with $\widetilde{y}_{w,c}$ and $\widetilde{z}_c$, respectively.

To obtain a feasible solution for $\mathcal{M}$, IRoP rounds the fractional values to either zero or one in a manner that maintains the feasibility of constraints without significantly increasing the objective value. We mainly focus on the decision variables $y_{w,c}$ in the following discussions, as the value of z_c can be directly computed from the value of decision variable $y_{w,c}$. If the value of $y_{w,c}$ is rounded to one for workload w and module c, then the value of z_c for the corresponding module must also be rounded to one.

The algorithm proceeds to round the variables iteratively, considering the workloads one by one. For each workload $w \in \mathcal{W}$, IRoP examines modules in an order based on the value of decision variables $y_{w,c}$. In line with this, it finds the module with the maximum value of $\widetilde{y}_{w,c}$, denoted by c', as follows:

$$c' \leftarrow \arg\max_{c \in \mathcal{C}} \widetilde{y}_{w,c} \,. \tag{5.3}$$

Next, IRoP fixes the value of decision variable $y_{w,c'}$ to one and solves the problem again, starting from the current values of other decision variables. Due to the small change in the value of decision variables, the linear program solver finds the next solution quickly or

Algorithm 1: IRoP: Iterative Rounding-based Placement.

```
   Input  : $\mathcal{M}$: Instance ofF-DWLP
   Output: $\{z_c, y_{w,c}\}$: Activation and Assignment Decisions
 1 $\widetilde{\mathcal{M}} \leftarrow$ relax($\mathcal{M}$)
 2 $\{\widetilde{z}_c, \widetilde{y}_{w,c}\} \leftarrow$ solve($\widetilde{\mathcal{M}}$)
 3 for $w \in \mathcal{W}$ do
 4     $\overline{\mathcal{C}}_w \leftarrow \{\}$
 5     while $\overline{\mathcal{C}}_w \neq \mathcal{C}$ do
 6         $c' \leftarrow \arg\max_{c \in \mathcal{C} - \overline{\mathcal{C}}_w} \widetilde{y}_{w,c}$
 7         $y_{w,c'} \leftarrow 1$
 8         $\{\widetilde{z}_c, \widetilde{y}_{w,c}\} \leftarrow$ solve($\widetilde{\mathcal{M}}$)
 9         if $\widetilde{\mathcal{M}}$ is feasible then
10             break
11         else
12             $y_{w,c'} \leftarrow 0$
13             $\overline{\mathcal{C}}_w$.add($c'$)
14     if $|\overline{\mathcal{C}}_w| = |\mathcal{C}|$ then
15         return FAIL
16 return $\{z_c, y_{w,c}\}$
```

determines that the problem has become infeasible.

If $\widetilde{\mathcal{M}}$ remains feasible, IRoP retains the allocation of module $c' \in \mathcal{C}$ for workload $w \in \mathcal{W}$ and terminates the current iteration to handle the next workload in the following iteration (see lines 9 and 10). However, if the problem becomes infeasible, IRoP concludes that it is impossible to allocate module $c' \in \mathcal{C}$ to workload $w \in \mathcal{W}$. Consequently, IRoP reverts the change to variable $y_{w,c}$ and solves $\widetilde{\mathcal{M}}$ again to select another processing module to assign it to the workload.

To ensure that a processing module is not repeatedly selected, IRoP maintains a set $\overline{\mathcal{C}_w}$, which starts empty at the beginning of each iteration. When $\widetilde{\mathcal{M}}$ becomes infeasible, IRoP fixes the value of decision variable $y_{w,c'}$ to zero and adds $c' \in \mathcal{C}$ to set $\overline{\mathcal{C}_w}$. If the size of $\overline{\mathcal{C}_w}$ becomes equal to the number of available modules $|\mathcal{C}|$, it is impossible to allocate any

CPU module to workload $w \in \mathcal{W}$, and the algorithm fails. In such a situation, lower-priority workloads can be eliminated, and the process can be repeated with a reduced set of workloads.

Once IRoP successfully completes the iterative rounding process, it obtains a feasible solution for F-DWLP, which respects the slowdown constraints of all workloads. The solution may not be optimal, as the rounding process introduces some degree of sub-optimality. However, the approximation ratio of IRoP indicates how close the solution is to the optimal solution. In the next subsection, we will present the theoretical analysis of IRoP and mathematically prove its approximation ratio.

5.3 Algorithm Analysis

In this section, we undertake a thorough theoretical analysis of IRoP, aiming to gain insights into its performance guarantees and computational efficiency. We commence our exploration by focusing on the algorithm's approximation ratio and its implications on solving F-DWLP. One crucial observation is that in our formulated problem, the slowdown of each workload $w \in \mathcal{W}$ is intrinsically bounded by the constant Δ_w, proposed in Constraint (4.1j). This inherent constraint ensures that the essential slowdown requirement of each workload remains satisfied throughout the algorithm's execution, preserving the overall quality of the solution and respecting the service level agreement. Our rounding procedure, which plays a significant role in obtaining feasible solutions, does not impact the slowdowns of the workloads, adding to the stability and reliability of the approach.

5.3.1 Approximation Ratio

In this subsection, we embark on deriving the theoretical approximation ratio of IRoP. The obtained approximation ratio is characterized by a combination of key factors, including the number of cores and demands ratio, denoted as $\widehat{N}$, the maximum ratio between static power coefficients, represented by $\widehat{S}$, and the maximum ratio between dynamic power coefficients among any two modules in the datacenter, denoted as $\widehat{D}$. Theorem 5.1 establishes that the approximation ratio ψ achieved by IRoP is bounded by $\max\{C \cdot \widehat{N} \cdot \widehat{S}, \widehat{D}\}$, providing an essential theoretical guarantee on the performance of the algorithm.

Theorem 5.1. *Algorithm IRoP attains the approximation ratio*

$$\psi = \max\{C \cdot \widehat{N} \cdot \widehat{S}, \widehat{D}\}, \tag{5.4}$$

where $\widehat{N} = \max_{\substack{c\in\mathcal{C}\\ w\in\mathcal{W}}} \{N_c/\nu_w\}$ *is the maximum ratio between the number of cores and demands,* $\widehat{S} = \max_{c,c'\in\mathcal{C}}\{I^s_c/I^s_{c'}\}$ *is the maximum ratio between static power coefficients, and* $\widehat{D} = \max_{c,c'\in\mathcal{C}}\{I^d_c/I^d_{c'}\}$ *is the maximum ratio between dynamic power coefficients among any two modules in the DDC.*

Proof. The power consumption term in the objective is given by:

$$\sum_{c\in\mathcal{C}} I^s_c \cdot z_c + I^d_c \cdot u_c\,. \tag{5.5}$$

In the worst-case scenario, the value of each z_c can increase by a factor of $\frac{C\times N_c}{\widetilde{N}_w}$. This increase occurs because the value of $\widetilde{N}_c$ can increase by a factor of C, and $\widetilde{z}_c$ can be as small as $N_w/\widetilde{N}_c$ to be larger than the left-hand side of the constraint. However, the value of z_c does not necessarily increase to exactly one. Therefore, the first term in (5.5), i.e., the static power consumption of powered-on modules, can increase by a factor of $\frac{C\times N_c}{\widetilde{N}_w}$ compared to its value in the solution of the linear problem. We should note that the static power consumption is not identical for all modules. Consequently, during rounding, a module with the highest static power consumption might be selected among modules with a positive value of z_c. As a result, the static power consumption increases by at most a factor of $\widehat{S}$.

To characterize the increase of the second term, we note that if the load due to each workload in one of the modules (i.e., module c' that was selected in (5.3)) increases by a factor of α, its combined load in other modules (i.e., $c \neq c'$) will decrease by a factor of $\frac{1}{\alpha}$. Therefore, the overall dynamic load characterized by the second term in (5.5) does not change. The only difference stems from the variations between dynamic power consumption coefficients. Similar to the static power consumption case, the dynamic power consumption increases at most by a factor of $\widehat{D}$.

Let $O_{\mathcal{M}}$ and $O_{\widetilde{\mathcal{M}}}$ denote the power consumption of the optimal solution of $\mathcal{M}$ and $\widetilde{\mathcal{M}}$ used in IRoP. Also, define O_{IRoP} to be the power consumption of the solution obtained by IRoP after rounding the decision variables of $\widetilde{\mathcal{M}}$. We have:

$$\begin{aligned} O_{\mathsf{IRoP}} &\leq \sum_{c \in \mathcal{C}} \left\{ \frac{C \times N_c}{\widetilde{N}_c} \cdot \widehat{S} \cdot I_c^s \cdot \widetilde{z}_c + \widehat{D} \cdot I_c^d \cdot u_c \right\} && (5.6) \\ &\leq \max \left\{ C \cdot \widehat{N} \cdot \widehat{S}, \widehat{D} \right\} \times \sum_{c \in \mathcal{C}} \left\{ I_c^s \cdot \widetilde{z}_c + I_c^d \cdot u_c \right\} && (5.7) \\ &= \psi \times O_{\widetilde{\mathcal{M}}} \leq \psi \times O_{\mathcal{M}}, && (5.8) \end{aligned}$$

which, establishes the theorem. □

In conclusion, the derivation of the approximation ratio involves a comprehensive analysis of the power consumption terms, revealing how the values of decision variables impact the overall power consumption of the datacenter. In the worst-case scenario, the rounding procedure may increase the static power consumption of powered-on modules, but it is constrained by the maximum factor of $\frac{C \times N_c}{\widetilde{N}_w}$. A careful examination of dynamic power consumption coefficients further contributes to bounding the approximation ratio, ensuring that the performance of IRoP remains within a provable range. By establishing the theoretical foundation for the algorithm's approximation ratio, we gain valuable insights into its performance and set the stage for further analysis of its computational complexity.

5.3.2 Run-time Complexity

Next, we delve into the analysis of IRoP's runtime complexity, a crucial aspect that sheds light on its computational efficiency. Theorem 5.2 establishes that our algorithm runs in polynomial time, which is a highly desirable property for practical solutions when it comes to resource allocation and workload placement problems.

Theorem 5.2. *IRoP runs in polynomial time.*

Proof. The rounding procedure consists of calling the linear program solver at most $n \times m$ times, where n and m represent the number of workloads and CPU modules, respectively. In [23], the authors prove that the complexity of solving a linear program with $n \times m$ variables using the interior point method is $O((n \times m)^{3.5})$. Therefore, in the worst-case scenario, the complexity of IRoP is $O((n \times m)^{4.5})$, ensuring its polynomial-time complexity. □

In conclusion for run-time analysis, the polynomial-time complexity of IRoP significantly enhances its computational efficiency. As a polynomial-time algorithm, IRoP scales effectively, even when dealing with large-scale instances, a critical trait in the era of massive datacenters and ever-growing workloads. The ability to handle complex optimization problems efficiently sets IRoP apart as a potent and promising algorithm in the field of datacenter management.

As a summary of this subsection, the theoretical results obtained thus far yield invaluable insights into the performance of IRoP. One crucial metric, the approximation ratio ψ, serves as a yardstick for measuring the proximity of IRoP's solutions to the optimal ones which comes from solving the original NP-Hard problem denoted by DWLP. A smaller ψ value indicates a more accurate approximation, underscoring the reliability and effectiveness of our algorithm in providing solutions that are remarkably close to the best possible outcomes. This enhanced approximation capability reinforces the practicality and relevance of IRoP in the context of datacenter management, where achieving near-optimal results is of utmost importance. The theoretical approximation ratio ψ, and the polynomial-time complexity make IRoP an attractive and powerful solution for tackling the multifaceted challenges in datacenter resource allocation. By striking a balance between computational efficiency and solution quality, IRoP emerges as a practical and effective approach for solving F-DWLP and advancing the state-of-the-art in datacenter optimization. In the next chapter, the effectiveness of the proposed workload placement algorithm is extensively evaluated while comparing the results in different metrics with the baselines coming from the recent literature.

Also, the completion time of the IRoPis compared to the optimal solution (*i.e.*, Gurobi solution for DWLP) for different datacenter scenarios, and the theoretical run-time analysis proposed in this section is evaluated.

Chapter 6

Performance Evaluation

In this section, we present a comprehensive evaluation of our proposed workload placement algorithm, IRoP. To provide an in-depth understanding of IRoP's effectiveness and operation, we have structured the evaluation into distinct subsections. Section 6.1 outlines the methodology of our experiments, detailing the setup of the testing environment and the alternative algorithms, referred to as our baselines, against which we will compare IRoP.

In Section 6.2, we assess the performance of our proposed algorithm based on the first metric, total workload slowdown, and compare its results with CFM, which is the most relevant baseline in this regard. Section 6.3 focuses on evaluating the performance of IRoP in terms of DDC power consumption, with a comparison against HEEP, which emphasizes this objective.

Combining the results, Section 6.4 elaborates on the primary objective of IRoP, optimizing the weighted sum of slowdown and power consumption while adhering to service level agreements. This section also provides a comparative analysis of IRoP's results with all other baselines.

Finally, in Section 6.5, we conduct a detailed assessment of the run-time complexity of IRoP, comparing its completion time with that of the most optimized algorithm, OPT, which utilizes the Gurobi solver and solves the NP-Hard MILP model outlined in Problem 1.

Table 6.1: CPU Modules Used in Evaluations.

CPU Modules Used in Evaluations			
Model	# Cores	Processing Capacity	TDP
AMD EPYC 9654	192	2	2.2
AMD EPYC 7763	128	1.5	1.7
Intel Xeon E7-8890	48	1	1

6.1 Methodology

In this section, we delve into the detailed methodology and setup that underpin our simulations, aimed at evaluating the performance and efficiency of different workload allocation algorithms in the context of a large-scale (*i.e.*, cluster-scale) disaggregated data center (DDC) environment. Our approach encompasses the emulation of Google's Aquila architecture, known for its capacity to house up to 1152 servers within a single cluster [14]. By replicating key facets of Google's data center architecture, we ensure the relevance of our analysis is comparable to other baselines detailed in the following subsections.

6.1.1 Environment Setup

This section provides a comprehensive description of the environment and setup employed for conducting the simulations in our study. The simulations were executed within a large-scale disaggregated data center (DDC) environment based on Google's Aquila architecture, which is recognized for its capability to accommodate up to 1152 servers in a single cluster [14].

To establish realistic workload requirements, which delineate the resource prerequisites of diverse tasks executed within the data center, we draw upon the extensive analysis conducted by Glawion et al. [15]. This analysis encompasses a comprehensive range of prevalent workloads that are typically encountered in data center operations, thus imbuing the simulated scenarios with authenticity. The simulated data center is equipped with three distinctive types of processors, each endowed with specific specifications and capabilities. The characteristics of these processor types are outlined in Table 6.1.

The development of the proposed IRoP algorithm and the associated baseline algorithms

Table 6.2: Datacenter and Workload Parameters.

Datacenter Parameters	
Symbol	Absolute Value or Range
$\mathcal{C}$	1200
F_c	Based on the Table 6.1
N_c	Based on the Table 6.1
L_c	64 GB (Latency is considered to be 20 nanoseconds)
I_c^s	75% of TDP (see Table 6.1)
I_c^d	25% of TDP (see Table 6.1)
M	20 TB (Latency is considered to be 5 microseconds)

Workload Parameters	
Symbol	Absolute Value or Range
$\mathcal{W}$	150
ν_w	1 - 48
ϕ_w	1 - 2 (Normalized)
μ_w	4 - 64 GB
$P_w^f(.)$	Based on the results in Section 3.2.2
$P_w^m(.)$	Based on the results in Section 3.2.1
Δ_w	5% - 25%

necessitated the creation of an approximately 1500-line codebase utilizing Python version 3.11. These algorithms were formulated to orchestrate the allocation of workloads to the available processors within the simulated DDC environment. The parameters governing the data center's configuration and the specific parameters of the simulated workloads are summarized in Table 6.2.

For each distinctive experimental configuration, a comprehensive suite of simulations was executed employing diverse algorithms, including the proposed IRoP algorithm and the established baselines. The principal objective of these simulations was to dissect and evaluate the performance implications of different allocation strategies within the system. The outcomes garnered from these simulations encapsulated two distinct categories of data: the prevailing state of the disaggregated data center, encompassing attributes such as available local memory and CPU utilization, and the completion times of all the simulated workloads.

To bolster the credibility and robustness of the results, each configuration underwent rigorous testing across a repertoire of 100 randomized scenarios. The final results were

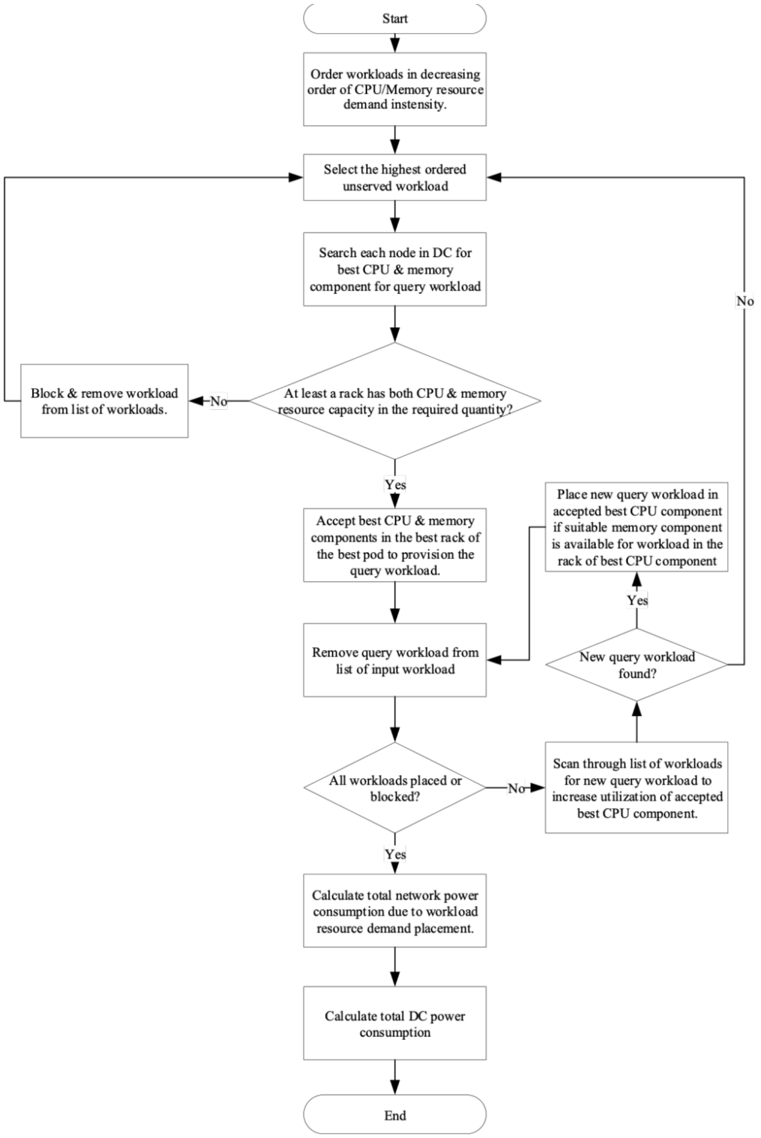

Start
Order workloads in decreasing order of CPU/Memory resource demand instensity.
Select the highest ordered unserved workload
Search each node in DC for best CPU & memory component for query workload
At least a rack has both CPU & memory resource capacity in the required quantity?
No
Block & remove workload from list of workloads.
Yes
Accept best CPU & memory components in the best rack of the best pod to provision the query workload.
Remove query workload from list of input workload
All workloads placed or blocked?
No
Scan through list of workloads for new query workload to increase utilization of accepted best CPU component.
New query workload found?
Yes
Place new query workload in accepted best CPU component if suitable memory component is available for workload in the rack of best CPU component
No
Yes
Calculate total network power consumption due to workload resource demand placement.
Calculate total DC power consumption
End

Figure 6.1: Flowchart of HEEP.

distilled by computing the averages across these diverse scenarios. To facilitate effective comparison and nuanced analysis, the aggregated outcomes were subsequently normalized by dividing each individual value by its corresponding maximum value among all the tested scenarios.

6.1.2 Alternative Algorithms

In addition to the proposed IRoP algorithm, a suite of baseline algorithms were implemented to serve as benchmarks for comparison and evaluation. These alternative algorithms encom-

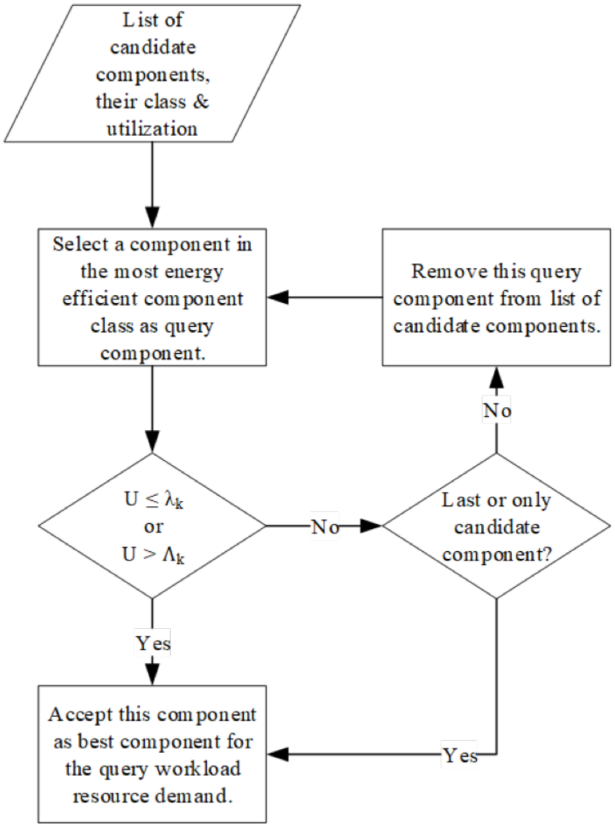

List of candidate components, their class & utilization
Select a component in the most energy efficient component class as query component.
Remove this query component from list of candidate components.
No
$U \leq \lambda_k$ or $U > \Lambda_k$
No
Last or only candidate component?
Yes
Accept this component as best component for the query workload resource demand.
Yes

Figure 6.2: Best component selection in HEEP.

pass a diverse spectrum of strategies for workload allocation within the complex landscape of the DDC environment which are mitigated from recent literature:

- **OPT**: This algorithm calculates the optimal solution for the intricate workload placement problem (DWLP). Its realization hinges on leveraging the capabilities of the Gurobi optimizer [18]. However, it is worth acknowledging that the resolution of the Integer Linear Programming (ILP) problem intrinsic to DDC placement entails a time-intensive problem. Consequently, relaxation and rounding techniques are employed to generate an approximate solution in the proposed algorithm while the OPT is attempting to solve the pure MILP formulations proposed in Section 4.1 and called DWLP using the Gurobi interface in Python.

- **HEEP** [3]: HEEP uses a greedy algorithm engineered to minimize the collective power consumption of different IT resources (*i.e.*, CPU, RAM, and networking resources such

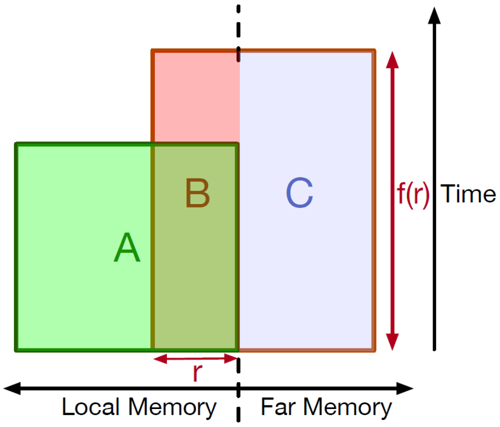


Figure 6.3: How CFM minimizes workload slowdown

as switches) within the physically disaggregated data center. This feat is accomplished through the meticulous sorting of workloads and resources based on their respective power efficiency and performance requirements. Fig. 6.1 shows the flowchart of this baseline while Fig. 6.2 shows how HEEP chooses the best power-efficient CPU module for each request. We recall the fact that the authors proposing HEEP have also formulated the power consumption of different resource types as a MILP model. However, since solving their proposed model is NP-hard, they proposed HEEP that mitigates the optimized solution while executing faster. Additionally, to make HEEP comparable to IRoP, our proposed algorithm, we do not consider the power consumption of other resources such as memory and networks when implementing HEEP and we only focus on the power efficiency of the CPU modules. All the other details of this algorithm are taken into account such as the utilization factors or the power efficiency level for the three types of CPU modules.

- **CFM** [4]: The CFM algorithm pivots its focus on the optimization of workload slowdown by modulating the allocation of local and remote memory. It tries to maximize the memory-time products attributed to each workload, which includes the minimization of remote memory usage. Fig. 6.3 shows how CFM tries to minimize the workload slowdown by reducing the remote memory time product. As shown in Fig. 6.3, A is the original memory-time product when no remote memory is used and workloads are using 100% local memory (*i.e.*, the scenario which happens when assigning workloads to traditional data centers or lightly loaded disaggregated data centers which have lots of available free local memory per CPU module). B+C is the new memory-time product when the workload is assigned to a disaggregated data center that is highly loaded or forced to use remote memory. In this scenario, B is the local portion of memory time, and C is the remote portion of the product. It is also noteworthy to mention that the authors in [4] have profiled some of the popular workloads and their results are similar to our profiling framework detailed in Section 3.2. The result of their work also proves that different workloads have different reactions to the remote memory latency which causes various workload slowdowns. However, they have not provided any evaluations or information on the accuracy of their profiling method and fitting function.

- **First-Fit**: The First-Fit algorithm is characterized by a simplistic strategy, wherein each workload is assigned to the first available CPU module that boasts an adequate reservoir of local memory. In scenarios where local memory is not sufficient, the algorithm seamlessly transitions to allocating remote memory to the workload. Therefore, in this baseline, all the available local memory is consumed at first before remote memory is assigned to any workloads, causing more power consumption and also high workload slowdown in some scenarios. For example, if a batch of less time-sensitive workloads, such as WordCounts, arrive at the datacenter first, and subsequently, a batch of more time-sensitive workloads, such as Kmeans, request to join, there might not be any available local memory to allocate to them while all the local memory is assigned to the first batch of the workloads. However, First-Fit is known for its simplicity and fast execution. Also, it is observable that there is always a sequence of workloads which if found properly, First-Fit will generate the optimum solution. In our experiments and testings, we also faced some scenarios where First-Fit generated the best solutions since the random shuffling of the workloads was accidentally creating the best sequence of workloads requesting resources from the datacenter.

In summary, the cumulative effect of these baseline algorithms, complemented by the proposed IRoP algorithm, engenders a comprehensive array of approaches catering to workload allocation within the the disaggregated data center environment. OPT serves its purpose as the optimal solution of the MILP formulation of the workload placement problem. Its results are comparable in two ways. The optimality of the results and the completion time (*i.e.*, run-time complexity) of IRoP. On the other hand, HEEP and CFM serve as the baselines for the power efficiency and workload slowdown optimality compared to IRoP, respectively. Finally, First-Fit complements the other baselines as the most simple approach for workload placement. In the following subsections, we extensively evaluate IRoPwith the above baselines using different metrics such as power consumption, total workload slowdown, weighted sum of slowdown and power consumption, and run-time complexity of the algorithms.

6.2 Total Slowdown of Workloads

As detailed in Section 3.2, one important disadvantage of physically disaggregated datacenters is the fact that utilizing remote memory introduces higher latency compared to the local memory used in traditional datacenters. In our proposed algorithm, we have taken this issue into account in three steps as follows:

- First, we established an accurate workload profiling methodology which creates an offline profile for each workload regarding how remote memory affects workload slowdown. This profiling framework is detailed in Chapter 3.
- Second, we considered a constraint that forces IRoP to always ensure that the slowdown each workload experiences is less than its service level agreement denoted by Δ_w and presented in Problem 1.
- Lastly, we considered the total slowdown of workloads in the objective of our proposed workload placement problem which is also presented in Problem 1. We should recall that θ_w denotes the final slowdown of each workload. In Chapter 5, this objective is also presented in Problem 2 and then used to optimize the output in the Algorithm 1.

In this section, first, we run IRoP in a DDC environment with different parameters using different scenarios, and then we evaluate the efficiency of IRoP in minimizing the total slowdown of workloads by comparing its results to the related baseline called CFM, detailed in the previous section.

To present a detailed assessment of IRoP's performance on controlling the slowdown of workloads, we embark on a journey through a spectrum of scenarios defined by varying values of Δ_w, specifically drawn from the set $\{0.5, 0.10, 0.15, 0.20, 0.25\}$. The focal point of our investigation lies in understanding the normalized slowdown results achieved by IRoP across this range of Δ_w values. The normalized slowdown for each workload is calculated using the following definition:

$$\text{Normalized Slowdown} = \frac{\text{Slowdown of the workload}}{\text{Maximum slowdown of workloads between all scenarios}} \tag{6.1}$$

dividing the slowdown of each workload by the maximum slowdown between all the tested scenarios. Our evaluation extends beyond this scope as we also utilize η, a critical parameter that governs the equilibrium between power consumption and workload slowdown. We chart the trajectory of normalized slowdown values as η changes from zero to one.

Notably, the interplay between η and the normalized slowdown is a key aspect of this exploration. The relationship becomes apparent as we observe an intuitive trend: an increase in η corresponds to an escalation in the normalized slowdown. This outcome aligns with expectations, as higher η values signify a heightened emphasis on power conservation (*i.e.*, trying to minimize the power consumption of CPU modules by compressing workloads into less number of CPUs and maintaining CPU modules powered-off as much as it can), consequently allowing for larger results for workload slowdown. This dynamic interdependence between η and normalized slowdown provides a versatile framework for datacenter operators to calibrate IRoP's performance in alignment with their specific priorities.

In our pursuit of comparative insights, we extend our gaze to incorporate the perfor-

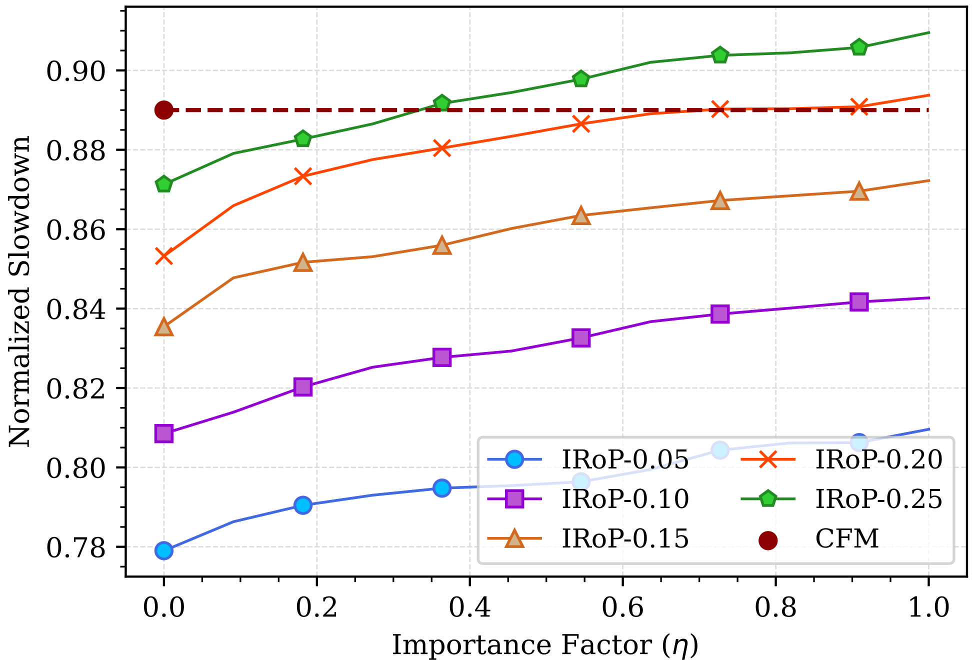


Figure 6.4: Total slowdowns of IRoP for different values of slowdown threshold compared to CFM.

mance of CFM, another algorithm that serves as a benchmark for evaluation. A decisive criterion in this context is the focus on the slowdown. Consequently, our analysis only permits meaningful comparisons at $\eta = 0$, since CFM exclusively prioritizes workload slowdown optimization and ignores the power consumption of the DDC completely. However, to render the comparison more holistic, we depict the result of CFM for all different values of η which is represented by a dashed line. As CFM is not dependent on the value of η, the output of this workload placement is fixed through the range of eta, creating a straight line on the chart. This technique is used only as a help in providing a broader perspective while acknowledging the algorithm's behavior beyond its exclusive focus on the workload slowdown.

In the presented data in Fig. 6.4, we observe a compelling pattern. IRoP consistently outperforms CFM across the range of Δ_w values. This observation underscores IRoP's superior adaptability and efficiency in diverse scenarios. Nonetheless, a critical eye will recognize a few exceptions. Specifically, when $\Delta_w = 0.25$ and $\eta > 0.3$, IRoP's performance exhibits a higher normalized total workload slowdown in contrast to CFM. This divergence in outcomes

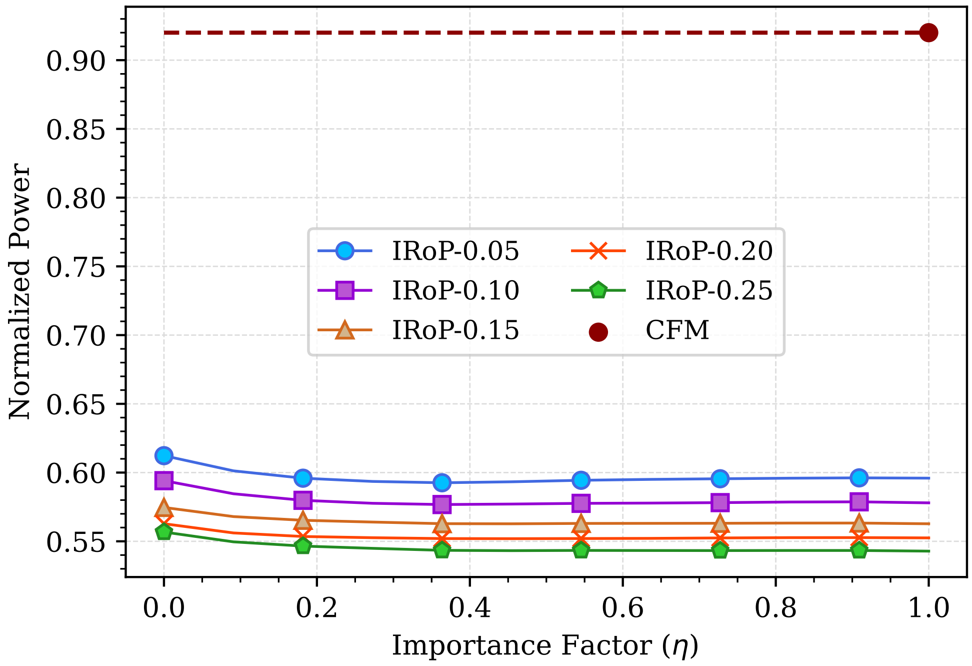


Figure 6.5: Total power consumption of IRoP for different values of slowdown threshold compared to CFM.

can be attributed to the interplay between the loose slowdown bounds imposed by IRoP and the substantial significance assigned to power conservation. In these instances, IRoP demonstrates its willingness to make certain trade-offs, favoring power efficiency at the cost of a slightly higher workload slowdown.

A comprehensive analysis, encompassing over a hundred randomized experiments with $\eta = 0$, yields insightful statistics. Comparing IRoP to CFM, we uncover a slowdown reduction of up to 12% at its maximum, with an average reduction of 5% and a minimum of 2%. Note that any comparison between IRoP and CFM with η set to any values rather than 0 is not reasonable as it causes IRoP to lose the workload slowdown to compensate for the power consumption considerations. These figures testify to the consistent results of IRoP in outperforming its benchmark counterpart, CFM.

In addition, Fig. 6.5 shows the normalized power consumption of the CPU modules when IRoP is utilized compared to the scenario where the CFM is used as the workload placement algorithm. Normalized power consumption is calculated using the following definition:

$$\text{Normalized Power} = \frac{\text{Total power consumption of CPUs}}{\text{Maximum total power consumption between scenarios}} \tag{6.2}$$

the total power consumption of CPUs by the maximum power consumption between tested scenarios. We observe that CFM does not consider the power consumption of the DDC as an important factor that causes large amounts of power consumption, costs, and a highly negative impact on the environment. In the following section, this aspect of IRoP is evaluated while compared to HEEP, the placement algorithm designed to minimize the power consumption of DDC.

In summary, in our study of workload slowdown in physically disaggregated datacenters, conducted by extensive experiments across various scenarios with different values of Δ_w and η, we found that IRoP consistently outperformed the CFM approach, achieving an average slowdown reduction of 5% and a maximum reduction of 12%. However, we observed that IRoP occasionally prioritized power conservation over minimizing workload slowdown, especially when Δ_w was high and η exceeded 0.3. The findings in this section emphasize the adaptability of IRoP compared to CFM, showcasing its capability to minimize total workload slowdowns when serving latency-sensitive workloads, while also accommodating higher slowdowns for cases where workloads are less sensitive to latency and power conservation is more important.

6.3 Total Power Consumption of DDC

In Subsection 3.1.2, we explained the importance of considering the power consumption of DDC resources as part of the objective of our proposed algorithm, IRoP. Throughout this thesis, we include the aspect of the power consumption of the DDC resources into our objective function in the three steps summarized below:

- First, resulting from the literature, we assumed that the main power consumers are the CPU modules and we ignored the consumption of the other components such as memory, networking modules, and cooling system. This assumption is based on the fact that servers are responsible for around 85% of the total power consumption of the DDCs and also, CPU modules are responsible for 90% of the power consumption of the servers. [45] Additionally, authors in [33] claimed that optimizing the power consumption of networking switches may not be effective, which is shown in Fig. 3.3.

- Second, after thoroughly considering the recent state-of-the-art literature [45, 38, 3, 45], we found a complete MILP model and formulation for the power consumption of the CPU modules based on their static (I_c^s) and dynamic power (I_c^d) consumption. The formulation proposed by [45] and [3] is presented in Equation (3.1) and is fully detailed in Subsection 3.1.2.

- Lastly, we used the utilization of CPUs as a factor for dynamic power usage. We assumed that CPUs can be powered off when there are no workloads assigned to them and the more active cores, the higher utilization and power consumption will be. Using the mentioned assumptions, we finally conclude our power consumption model presented in Equation (4.2) which is detailed in Section 4.3.

In this section, we evaluate the effectiveness of minimizing power consumption in IRoP while comparing it with HEEP, which is detailed in Subsection 4.3.

As mentioned in the previous subsection, we run IRoP in a pre-defined DDC environment with the same parameters and scenarios, and we calculate the total power consumption of the powered-on CPU modules based on Equation (4.2). To ensure comparability of the results, we use the same scenarios and environment to calculate the power consumption of the DDC when utilizing HEEP as the workload placement algorithm. To ensure fairness in the results, we conducted the experiments 100 times for each algorithm. Then, we calculate and report the average results for the total power consumption.

There are two parameters that affect the results of IRoP: Δ_w and η. As mentioned in the previous subsection, the value of Δ_w is drawn from the set $\{0.5, 0.10, 0.15, 0.20, 0.25\}$, while

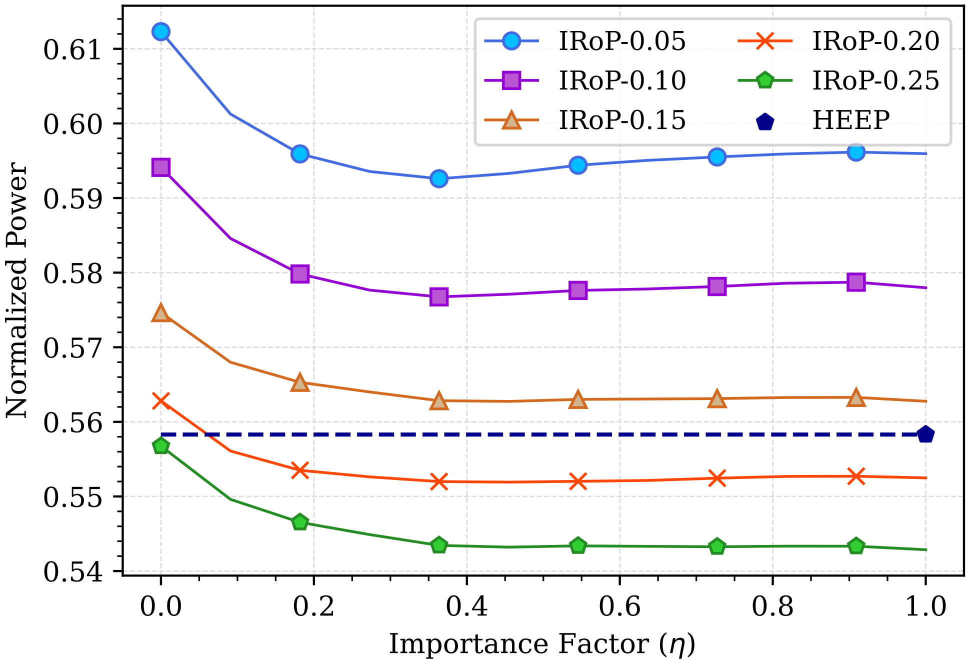


Figure 6.6: Power consumption of IRoP for different values of slowdown threshold compared to HEEP.

the value of η ranges from 0 to 1, determining the importance level of power consumption, from complete disregard to high consideration. These parameters not only influence the total slowdown of the workloads, as observed in the previous subsection, but also impact the overall power consumption of the DDC. Therefore, IRoP provides the datacenter owner with the flexibility to select the desired output based on the type of workloads assigned to the DDC at any given time. Fig. 6.6 not only sheds light on the interplay between normalized power consumption and the parameters Δ_w and η but also unravels deeper insights into the performance characterization of IRoP. This visual representation offers a rich canvas upon which we can paint a detailed narrative of how IRoP responds to varying conditions and priorities.

As observed in Fig. 6.6, an increase in the value of Δ_w, which determines the threshold of the maximum slowdown for each workload, results in a decrease in the total power consumption of the DDC. This phenomenon can be explained by the fact that when using higher slowdown thresholds, IRoP has more flexibility in selecting power-efficient CPUs with lower processing capacities, leading to reduced power consumption despite higher workload slowdown. This trade-off enables IRoP to achieve lower power consumption by keeping high-power consumer CPUs powered off, a feat impossible when dealing with inflexible or latency-sensitive workloads that exhibit a low tolerance for workload slowdown. It is evident that when Δ_w is set to 0.05 or 0.10, the system lacks flexibility in managing power consumption, leading to a significant disparity in total power consumption compared to the values of Δ_w set to 0.15, 0.20, and 0.25.

Furthermore, when varying the value of η, it is evident that completely disregarding power consumption (i.e., $\eta = 0$) leads to significantly higher power consumption, and any increase in this parameter will markedly affect the performance of IRoP in this regard. This observation can be explained by the way the objective of IRoP is defined in Problem 2. When $\eta = 0$, the power consumption-conservative behavior of IRoP is entirely eliminated, allowing the selection of CPU modules that fulfill other terms and constraints of the IRoP objective without considering the power efficiency of these modules. As the value of η increases, IRoP endeavors to utilize more power-efficient modules while adhering to other constraints, resulting in a linear decrease in total power consumption once this aspect is taken into account (i.e., for $\eta > 0$).

Considering that our comparable baseline, HEEP, focuses solely on the power consumption of DDC resources without minimizing workload slowdown, a comparison between HEEP and IRoP is only relevant when $\eta = 1$. It should be noted that even in this specific scenario, there exists a distinction between HEEP and IRoP: the constraint that alters the behavior of IRoP by restricting the maximum allowed slowdown for each workload, denoted as Δ_w. Hence, to accurately capture the differences between these two workload placement algorithms, we utilize the same values for Δ_W as depicted in Fig.6.6. Notably, it is evident that HEEP outperforms our proposed algorithm when $\Delta_w < 0.15$, regardless of the value of η. This outcome can be attributed to the fact that imposing a limitation on workload slowdown compels IRoP to utilize highly powerful CPU modules, thereby sacrificing power consumption efficiency. In contrast, HEEP does not take workload slowdown into consideration and selects

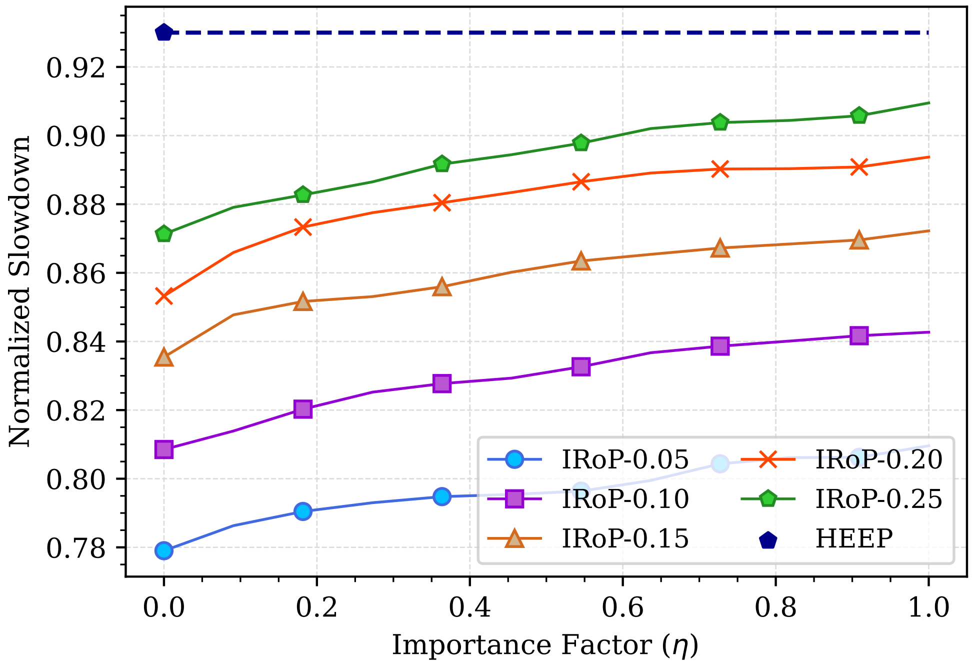


Figure 6.7: Total slowdown of workloads when using IRoP for different values of slowdown threshold compared to HEEP.

the most power-efficient modules in the DDC through an accurate and calculated procedure outlined in Fig 6.1 and Fig. 6.2. However, when utilizing $\Delta_w \geq 0.20$, IRoP outperforms HEEP by selecting a more power-efficient combination of CPU modules that ensures lower power consumption while adhering to the service level agreements. It is noteworthy that since the output of HEEP remains constant when altering Δ_W or η, the fixed output of this algorithm is illustrated by a pentagon in Fig. 6.6. However, to ensure a comprehensive comparison, we represent the result of HEEP for all different values of η with a dashed line, similar to the previous subsection.

Finally, to quantify the comparison between the results generated by both algorithms, we observe the superiority of IRoP over HEEP. Specifically, IRoP achieves a maximum power reduction of up to 8% and an average reduction of 2.5%, while also delivering lower total workload slowdown in all tested scenarios, irrespective of the combination of Δ_w and η. These findings are clearly depicted in Fig. 6.7.

In summary, we conducted over 100 trials to test the performance of IRoP with various values for Δ_W and η, employing different randomized settings as outlined in Table 6.1 and Table 6.2. Subsequently, we computed the total power consumption of CPU modules, based on Equation (4.2), to compare the results with HEEP, which is designed to minimize power consumption in a greedy manner, as elaborated in Fig.6.1 and Fig6.2. The findings reveal the superiority of IRoP over HEEP in terms of power consumption savings when suitable values for Δ_W are used. Furthermore, IRoP outperforms HEEP in terms of workload slowdown across all tested scenarios. The flexibility and superiority of IRoP are also evident in Fig.6.6 and Fig.6.7 with regard to power consumption and workload slowdown, respectively. Finally, we reported a maximum power reduction of 8% and an average reduction of 2.5% compared to HEEP, calculated based on the average result of 100 experiments.

6.4 Weighted Sum of Slowdown and Power Consumption

To this end, we compared the individual objectives of total workload slowdown and the power consumption of DDC in Subsections 6.2 and 6.3, respectively. While the evaluations demonstrated how IRoP outperforms the respective baselines for each objective by exhibiting a lower total workload slowdown compared to CFM and achieving a more optimized power consumption compared to HEEP, the primary objective of IRoP has not been thoroughly explored and explained in the preceding subsections. As indicated in Problem 2, the objective of our proposed algorithm incorporates both the minimization of total workload slowdown and power consumption, with a balance achieved by manipulating a factor known as η. In this subsection, we delve into the combined results when employing IRoP and other baselines (CFM, HEEP, and First-Fit), concluding that IRoP provides a superior balance between the two aspects of the objective in most scenarios.

To maintain consistency with the previous subsections in this chapter, we utilize the same values of Δ_w drawn from the set $0.5, 0.10, 0.15, 0.20, 0.25$, while the value of η ranges from 0 to 1. We also replicated the aforementioned scenarios randomly selected from the parameters and notations outlined in Table 6.1 and Table 6.2. Furthermore, we executed the

First-Fit algorithm under the same settings and variables, incorporating the results of this simple greedy method in our analysis to gain a comprehensive understanding of the differing performances offered by each workload placement algorithm.

It should be noted that, as explained in Subsection 6.1.2, we assume that this algorithm fills the available local memory in a greedy manner and only resorts to remote memory when no local memory is left in any of the CPU modules installed in the DDC environment. Under this assumption, we assert that this behavior is somewhat akin to that of CFM, with the initial aim being to minimize the workload slowdown by allocating all available local memory before resorting to remote memory allocation. Consequently, we consider this algorithm to be more relevant to IRoP when using $\eta = 0$ as an input parameter, effectively disregarding power consumption.

However, in cases where CPU modules or the requesting workloads are not arranged in an optimized manner (which is typically unknown to the algorithm or the datacenter owner), it is highly likely for First-Fit to allocate a substantial amount of local memory to latency-insensitive workloads. This may result in insufficient local memory allocation for latency-sensitive workloads, ultimately leading to a significant increase in the total workload slowdown within the DDC.

Figure 6.8 illustrates how the objective of the weighted sum of workload slowdown and power consumption varies across different values of Δ_w and η for our proposed algorithm as well as other baselines. It is important to note that we omitted the results for IRoP-0.10 and IRoP-0.20 to enhance the readability of the plot, as the general trend remains consistent across different values of Δ_w. Furthermore, as previously mentioned, the results for CFM and First-Fit are represented as solid points at $\eta = 0$, while HEEP is depicted as a solid point at $\eta = 1$. We computed both the workload slowdown and power consumption of each of these mentioned baselines. Considering that the outputs remain the same across different values of Δ_w and η, we used the linear summation of these two terms, resulting in a continuous dashed line for each of these alternative algorithms.

To provide further clarity on Figure 6.8, in line with the objective outlined in Problem 2,

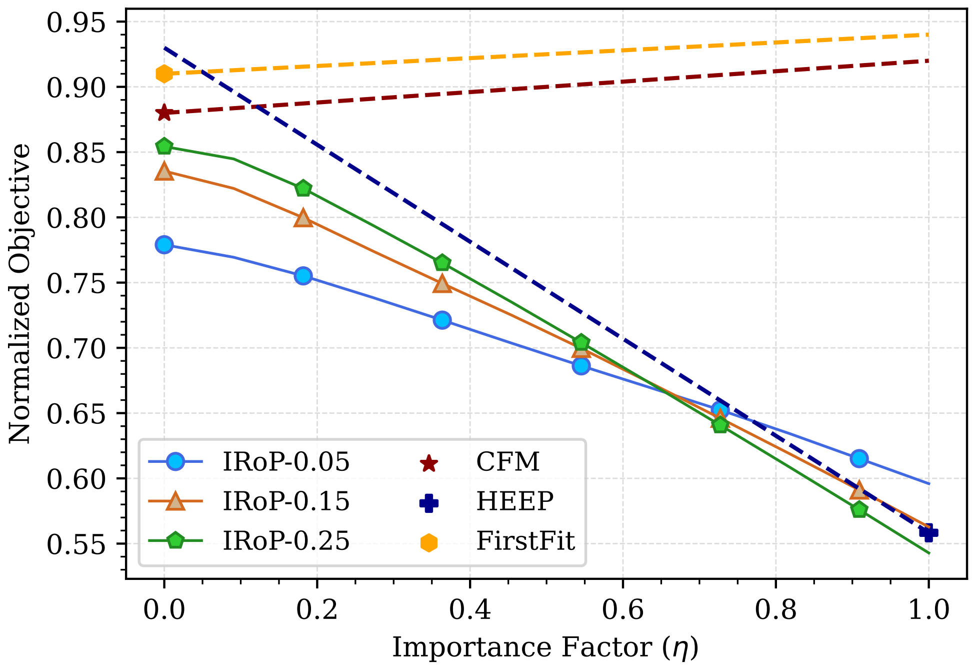


Figure 6.8: Weighted sum of slowdown and power of IRoP for different values of slowdown threshold compared to baselines.

the aim is to minimize the summation of the two terms. Consequently, in our plot, lower values indicate a superior performance of the algorithm. It is evident that IRoP outperforms all the other baselines in this objective, as it consistently delivers lower values across almost all the values of Δ_w and η. However, in scenarios where $\eta \geq 0.7$ and $\Delta_w = 0.05$ (*i.e.*, a situation characterized by stringent slowdown thresholds for IRoP), HEEP exhibits superior performance in terms of the objective value. It is important to note that in such scenarios, the use of a strict threshold for IRoP is not recommended if power consumption takes precedence, and the datacenter should introduce incentives for end-users to encourage them to be more flexible and tolerate higher workload slowdowns. For example, during peak power consumption hours when power costs are at their highest, datacenter owners can implement a cost-effective approach by offering cheaper plans or discounts to users who are willing to accept workload slowdowns. This strategy not only minimizes costs but also reduces the datacenter's carbon footprint, demonstrating a heightened concern for the environment.

In addition to the objective results visualized in Fig. 6.8, Fig. 6.9 also demonstrates how

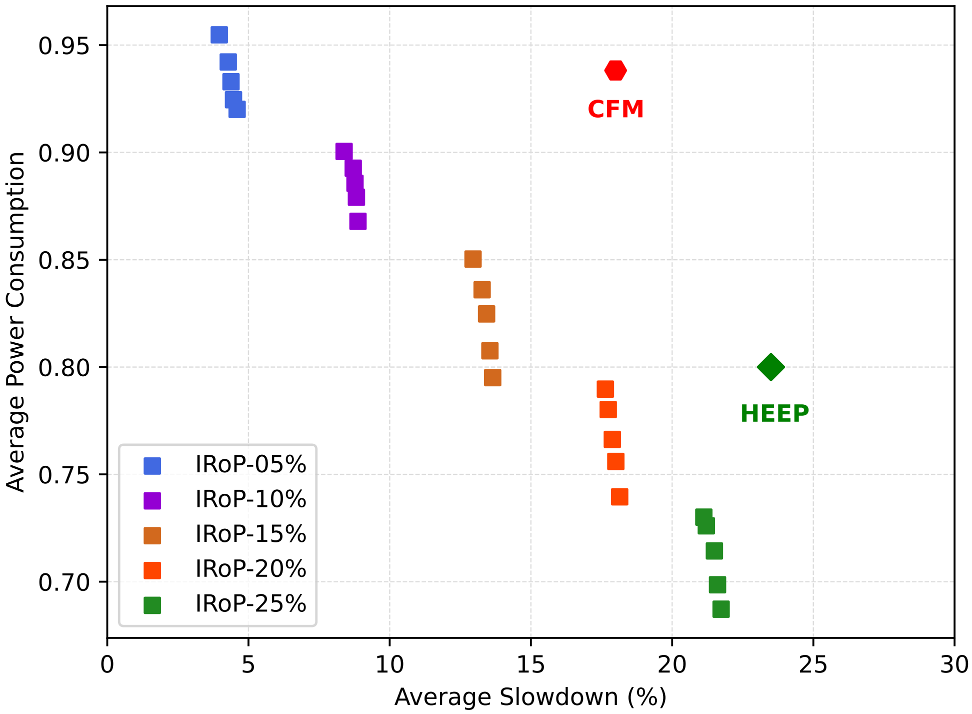


Figure 6.9: Normalized average power consumption versus the average slowdown of each workload when using IRoP compared to CFM and HEEP.

the two objectives of IRoP, namely power consumption and workload slowdown, vary as the datacenter administrator adjusts the system parameters: workload slowdown threshold (Δ_w) and the importance factor (η). While the other two baselines lack flexibility and have fixed results, IRoP is capable of producing different outputs by carefully selecting parameters based on the preferences of the datacenter owner. Notably, in this Pareto front plot, we are illustrating the average slowdown for each workload and the average power consumption for each CPU module when utilizing IRoP, CFM, and HEEP as the datacenter workload placement algorithm. Additionally, it is important to note that the y-axis is normalized by the maximum power consumption and is averaged by dividing the total power consumed by the total number of CPUs, including the CPU modules that are powered off during the simulation.

In summary, the Pareto front plot depicted in Fig. 6.9 indicates that as the slowdown threshold increases, the power consumption decreases. With higher slowdown thresholds, IRoP assigns workloads to CPUs with lower power consumption to mitigate high power costs,

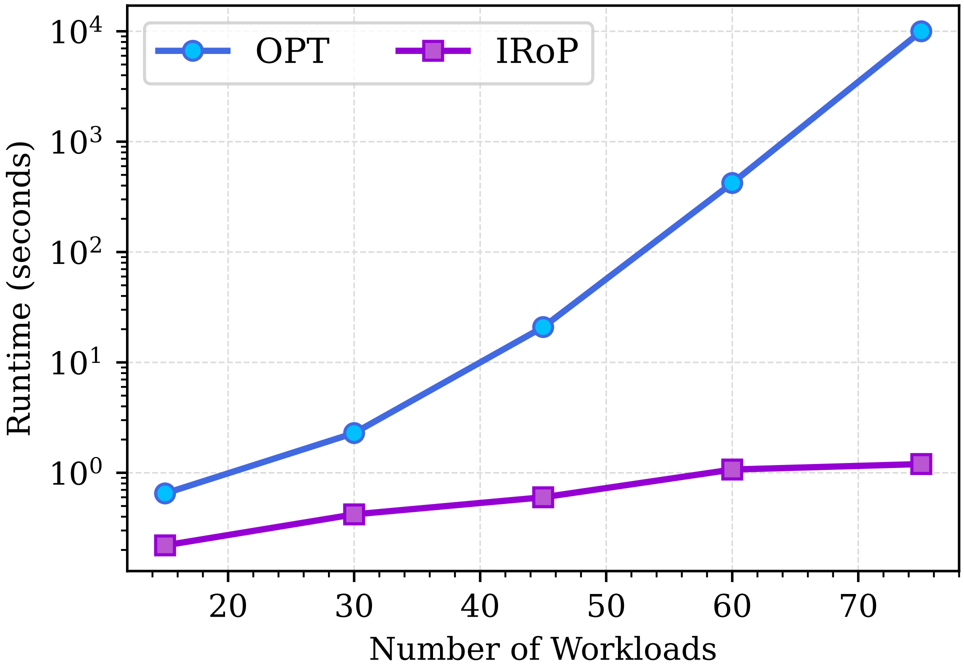


Figure 6.10: Impact of the number of workloads on the run-time of IRoP and OPT. The number of CPU modules is fixed at 30.

while allowing for a more relaxed constraint on workload slowdown. Conversely, when implementing stricter constraints for workload slowdown, CPU modules with higher processing capacity are utilized to compensate for the slowdown caused by the high latency of remote memory, albeit at the expense of the power efficiency of the placement algorithm. Overall, both Fig.6.9 and Fig.6.8 collectively demonstrate how IRoP outperforms other baselines by displaying flexibility and providing the datacenter owner with greater freedom to achieve their objectives, from power conservation to minimizing the total workload slowdown, all while adhering to the service level agreements, the slowdown threshold of each workload.

6.5 Run-time Complexity

In Section 4.1, we introduced a MILP model for workload placement. We combined this model with the power consumption MILP model outlined in Section 4.3. Subsequently, we formulated Problem 1, which, if correctly solved (*e.g.*, using the Gurobi solver), yields the

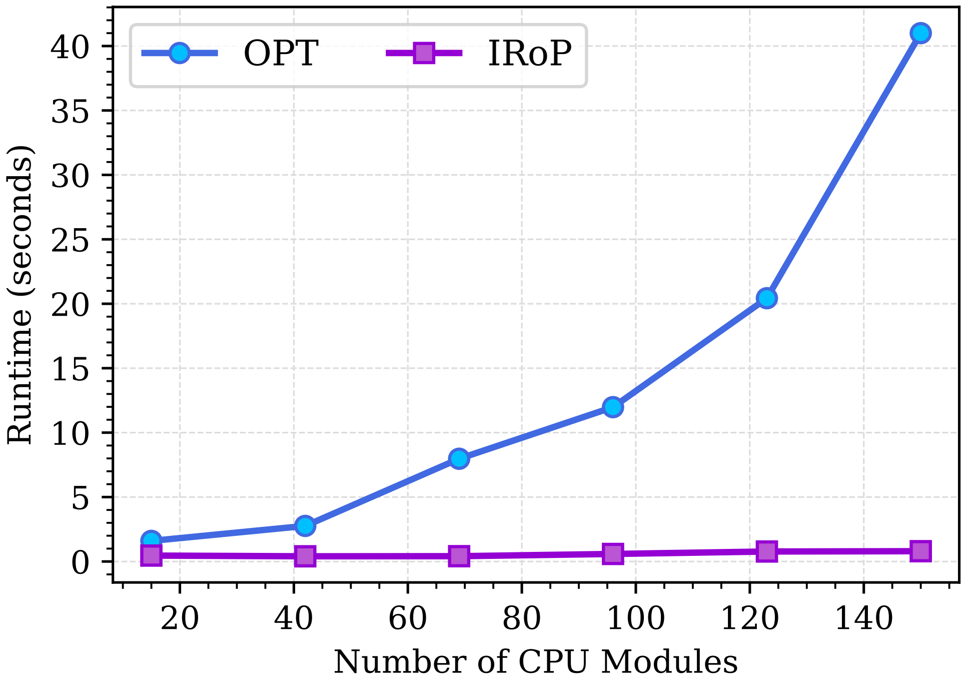


Figure 6.11: Impact of the number of CPU modules on the run-time of IRoP and OPT. The number of workloads is fixed at 30.

most optimized results for our workload placement problem, ensuring bounded slowdown while minimizing the power consumption of the DDC. However, as noted in [58], the one-dimensional bin packing problem is known to be NP-Hard. Consequently, the proposed MILP model is also NP-Hard, given that it is a variant of the multi-dimensional bin packing problem. As a result, the time complexity of the main MILP model would be exponential, making it time-consuming to implement in real-world scenarios involving a substantial number of CPU modules and workloads. To ensure the scalability and applicability of IRoP to large-scale scenarios, we proposed an approximation algorithm that solves the workload placement problem in polynomial time, formulated in Problem 2 and Algorithm 1. In Subsection 5.3.2, we conducted a theoretical analysis of the runtime complexity of IRoP, revealing a polynomial complexity of $O((n \times m)^{3.5})$. In this section, to evaluate the theoretical analysis and performance of IRoP, we thoroughly tested both IRoP and the optimal solution of Problem 1 (referred to as OPT), comparing their respective completion times.

Since we are dealing with a 2-dimensional bin packing problem, we aim to investigate the influence of each dimension (i.e., the number of CPU modules and requesting workloads) on the completion time of IRoP and OPT. To begin, we set the number of CPU modules at a fixed value of 30 and vary the number of workloads from 10 to 75. As depicted in Fig. 6.10, an increase in the number of requesting workloads leads to an exponential escalation in the completion time of OPT. It is important to note that the y-axis of the plot is presented in a logarithmic scale.

Conversely, under the same scenario, in the same plot, it is observable that the completion time of IRoP exhibits a linear growth pattern, highlighting the scalable nature of the proposed algorithm and proving the accuracy of the run-time complexity analysis provided in Subsection 5.3.2. It is crucial to mention that we were unable to continue the experiment with a larger number of workloads (more than 75) as OPT becomes too complex to solve within a reasonable time frame using the Gurobi solver.

To investigate the impact of the number of CPU modules, we conducted tests on both algorithms using a fixed number of 30 workloads while progressively increasing the number of CPU modules from 15 to 150. Subsequently, we computed the completion time of IRoP and CFM for each configuration. The final results are illustrated in Fig. 6.11.

It is evident that while OPT still demonstrates an exponential run-time behavior, the increase in the number of CPU modules does not contribute to the complexity of the problem in the same manner as the increase in the number of workloads. This observation can be attributed to the fact that the CPU modules serve as the bins in a bin packing problem, and a higher number of bins does not significantly impact the problem's complexity; rather, it provides more flexibility for the algorithm. Additionally, the higher the number of workloads, the more challenging it becomes for an algorithm to solve the problem with the same number of CPU modules. Moreover, as demonstrated, an increase in the number of CPUs has a minimal effect on the completion time of IRoP, further validating the scalability of the proposed algorithm.

Furthermore, in these small-scale tests, IRoP introduces approximately a 5% higher total workload slowdown while increasing the power consumption by about 7% compared to OPT.

These values remain significantly below the theoretical bound computed in Subsection 5.3.1. Moreover, for large-scale tests, the run-time of IRoP remains under 30 seconds, a satisfactory performance within the testing environment.

In conclusion, our practical run-time experiments validate the theoretical analysis provided in Subsection 5.3.2. Both the results depicted in Fig. 6.10 and Fig. 6.11 underscore the linear increase in the completion time of IRoP, alongside the exponential behavior observed for OPT under the same settings and scenarios. Notably, increasing the number of workloads exhibits a greater impact on the completion time of both algorithms compared to the number of CPUs. Also, the outcomes produced by IRoP closely resemble those generated by OPT, with the workload slowdown and power consumption staying within 5% and 7%, respectively, proving the accuracy of bounds computed in Subsection 5.3.1. Lastly, the completion time of IRoP remains under 30 seconds when simulating a large-scale DDC environment, a performance that is deemed satisfactory.

Chapter 7

Conclusion and Future Works

In this chapter, we provide a summary of the discoveries and invaluable contributions made throughout the course of this thesis. Then, we discuss potential future directions for further research, including the extension of the power consumption model, orchestration development, and improving the performance of IRoP by utilizing machine learning techniques. By providing a review of the outcomes drawn from this research and illustrating the exciting potential for future work, this chapter serves as a road-map for the continued advancement of resource allocation in disaggregated datacenters and its potential impact on power consumption and carbon emissions.

7.1 Conclusion

Throughout this thesis, the central objective has been to develop a workload placement algorithm that optimizes the overall slowdown of all workloads within the DDC while minimizing the power consumption of DDC resources. To achieve this, we began by examining the slowdown characteristics of workloads, understanding the impact of remote memory latency on workload completion time, as well as the influence of CPU processing capacity (i.e., CPU module frequency) on execution time. Subsequently, by modifying the Linux Kernel swap routine, we introduced an artificial delay to simulate remote memory. This allowed us to evaluate different popular workloads in our simulated DDC environment, assessing the proposed function that approximates the impact of remote memory on workload slowdown. Furthermore, we utilized Linux modules to assess the proposed function for approximating the impact of CPU frequency on workload slowdown. By combining these two slowdown characteristics, we developed a comprehensive function that estimates workload slowdown based on the CPU frequency and assigned remote memory ratio.

In terms of estimating the power consumption of DDC resources, we initially determined that servers account for the majority of power consumption in a datacenter, contributing up to 80% of the total power usage. Our investigation highlighted CPU modules as the primary contributors to this power consumption, representing as much as 90%, as indicated by recent literature. Subsequently, we conducted a thorough review of CPU power consumption models proposed in the literature, eventually selecting a state-of-the-art model that precisely replicates the power consumption patterns of CPU modules within a datacenter. This chosen model was then seamlessly integrated with the CPU utilization formula. Lastly, we selected three distinct real-world CPU modules currently available in the market for utilization within our simulated disaggregated datacenter.

Using the derived slowdown function and the power consumption model, we formulated a complete MILP model (Problem 1) with all the necessary constraints, including the workload slowdown threshold and the trade-off between slowdown and power consumption. As the proposed model was inherently NP-Hard, we introduced an approximation algorithm that effectively emulates the optimized model. In our evaluations, we found that the proposed algorithm outperforms other baselines provided by recent literature across various relevant scenarios and settings. Thus, this thesis successfully presents an algorithm that effectively balances the trade-off between power consumption and total workload slowdown, while adhering to the threshold for the slowdown of each workload.

7.2 Limitations of Our Work

In this subsection, we discuss the limitations and assumptions that were considered to make the defined problem approachable.

- **Offline Profiling:** In this thesis, we profiled various popular workloads in an offline environment. This means that for IRoP to be deployed in a real-world scenario, each workload should first be profiled in a DDC, which is a challenge for larger workloads. Interested researchers can explore online profiling methods or integrate ML/AI methods to tackle this challenge, as mentioned in the following section.

- **Trusting Environment:** We assume that the datacenter owner and clients can completely trust each other. For example, if a client allows a maximum slowdown of 5% (i.e., $\Delta = 5\%$), we assume that the datacenter owner respects this agreement and sets IRoP to follow the exact SLAs.

- **Cache Interference:** We assume that all of the workloads are independent and there is no communication between them. Also, we assume a completely and ideally isolated setup. Therefore, we ignore cache interference between different co-located workloads. In our experiments, we profiled workloads independently and did not consider running them simultaneously to see how this might change the formulations.

- **Isolation of CPU Frequency and Remote Memory:** When characterizing workload slowdown, we assume that the impact of CPU frequency on slowdown is completely independent of the impact of remote memory ratio on slowdown. When scaling CPU frequency or changing the remote memory ratio, we conducted our experiments independently of each other. Therefore, we have not considered any relationship between these two factors in our slowdown model.

7.3 Future Works

The contents of this thesis not only make a significant contribution to the domain of resource allocation within disaggregated datacenters but also lay down a pathway for driving subsequent progress in the realms of data center optimization and management. Future research directions could include:

- **Extending IRoP to support the power consumption of other resources:** In this thesis, we model the power consumption of CPU modules in a Disaggregated Data Center (DDC) environment. While CPUs are responsible for 85% of the total power consumption of DDCs [45], other resources can be included in the future and make IRoP more accurate.

- **Orchestration Development:** Adding an agent to monitor the slowdown of workloads and perform re-allocation is an interesting extension of our work for volatile environments which improves the scalability of IRoP.

- **Investigating IRoP in real-world DDCs:** Our evaluations replicate the environment of a large-scale DDC and the promising results exhibited by IRoP in this context underscore the potential. However, conducting practical implementation and testing of IRoP within a real-world DDC is essential for gaining invaluable insights into its real-world performance and effectiveness. This step would not only allow for the refinement and optimization of IRoP, but also significantly contribute to a deeper understanding of how it influences power consumption and workload slowdowns in real-world DDC scenarios.

- **Profiling other vital workloads:** We evaluated three workloads in this thesis based on the fact that they are all using an iterative approach that requires multiple accesses to the memory. However, we can use the same emulated DDC environment to test and evaluate other workloads under different circumstances. Then, we can revise the workload models to enhance IRoP accuracy and performance.

- **Utilizing AI and machine learning methods:** To effectively handle incoming workload requests, we can leverage the latest advancements in machine learning (ML) and artificial intelligence (AI) to forecast future loads. By integrating machine learning techniques into IRoP, it would be capable of resource allocation not solely based on the present job batch, but also utilizing predicted data regarding upcoming workloads and their associated demands. This enhancement can further optimize the performance of IRoP.

Bibliography

[1] Bulent Abali, Richard J. Eickemeyer, Hubertus Franke, Chung-Sheng Li, and Marc A. Taubenblatt. Disaggregated and optically interconnected memory: when will it be cost effective?, 2015.

[2] Marcos K. Aguilera, Nadav Amit, Irina Calciu, Xavier Deguillard, Jayneel Gandhi, Stanko Novaković, Arun Ramanathan, Pratap Subrahmanyam, Lalith Suresh, Kiran Tati, Rajesh Venkatasubramanian, and Michael Wei. Remote regions: a simple abstraction for remote memory. In *Proc. USENIX ATC*, 2018.

[3] Opeyemi O. Ajibola, Taisir El-Gorashi, and Jaafar Elmirghani. Energy efficient placement of workloads in composable data center networks. *Journal of Lightwave Technology*, 39(10), 2021.

[4] Emmanuel Amaro, Christopher Branner-Augmon, Zhihong Luo, Amy Ousterhout, Marcos K. Aguilera, Aurojit Panda, Sylvia Ratnasamy, and Scott Shenker. Can far memory improve job throughput? In *Proc. EuroSys*, 2020.

[5] Ofer Biran, Antonio Corradi, Mario Fanelli, Luca Foschini, Alexander Nus, Danny Raz, and Ezra Silvera. A stable network-aware vm placement for cloud systems. In *Proc. IEEE/ACM CCGRID*, 2012.

[6] Rajkumar Buyya, Toni Cortes, and Hai Jin. *An Introduction to the InfiniBand Architecture*, pages 616–632. 2002.

[7] Irina Calciu, M. Talha Imran, Ivan Puddu, Sanidhya Kashyap, Hasan Al Maruf, Onur Mutlu, and Aasheesh Kolli. Rethinking software runtimes for disaggregated memory. In *Proc. ACM ASPLOS*, 2021.

[8] Hadrien Cambazard, Deepak Mehta, Barry O'Sullivan, and Helmut Simonis. Bin packing with linear usage costs – an application to energy management in data centres. In *Proc. PPCP*, 2013.

[9] Qixiang Cheng, Meisam Bahadori, Madeleine Glick, Sébastien Rumley, and Keren Bergman. Recent advances in optical technologies for data centers: a review. *Optica*, 5(11), 2018.

[10] Aleksandar Dragojevi, Dushyanth Narayanan, Miguel Castro, and Orion Hodson. FaRM: Fast remote memory. In *Proc. USENIX NSDI*, 2014.

[11] Brice Ekane, Yohan Pipereau, Boris Teabe, Alain Tchana, Gael Thomas, Noel de palma, and Daniel Hagimont. Network in disaggregated datacenters, 2021.

[12] M. J. Feeley, W. E. Morgan, E. P. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing global memory management in a workstation cluster. *SIGOPS Oper. Syst. Rev.*, 29(5), 1995.

[13] Peter X. Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. Network requirements for resource disaggregation. In *Proc. USENIX OSDI*, 2016.

[14] Dan Gibson, Hema Hariharan, Eric Lance, Moray McLaren, Behnam Montazeri, Arjun Singh, Stephen Wang, Hassan M. G. Wassel, Zhehua Wu, Sunghwan Yoo, Raghuraman Balasubramanian, Prashant Chandra, Michael Cutforth, Peter Cuy, David Decotigny, Rakesh Gautam, Alex Iriza, Milo M. K. Martin, Rick Roy, Zuowei Shen, Ming Tan, Ye Tang, Monica Wong-Chan, Joe Zbiciak, and Amin Vahdat. Aquila: A unified, low-latency fabric for datacenter networks. In *Proc. USENIX NSDI*, 2022.

[15] Alex Glawion. How much ram do you need? different workloads explored., 2022.

[16] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G. Shin. Efficient memory disaggregation with infiniswap. In *Proc. USENIX NSDI*, 2017.

[17] Zhiyuan Guo, Yizhou Shan, Xuhao Luo, Yutong Huang, and Yiying Zhang. Clio: A hardware-software co-designed disaggregated memory system. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 417–433, New York, NY, USA, 2022. Association for Computing Machinery.

[18] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[19] Qinfen Hao. Keynote: "high throughput computing data center". In *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 1–1, 2014.

[20] Howard. A quick look: Roce vs infiniband rdma vs tcp/ip., Feb 2023.

[21] Stephen Ibanez, Alex Mallery, Serhat Arslan, Theo Jepsen, Muhammad Shahbaz, Changhoon Kim, and Nick McKeown. The nanopu: A nanosecond network stack for datacenters. In *Proc. USENIX OSDI*, 2021.

[22] Piotr Luszczek Jack Dongarra. Hpcg benchmark, 2016.

[23] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4, 1984.

[24] Jonghyeon Kim, Wonkyo Choe, and Jeongseob Ahn. Exploring the design space of page management for Multi-Tiered memory systems. In *Proc. USENIX ATC*, 2021.

[25] Ana Klimovic, Christos Kozyrakis, Eno Thereska, Binu John, and Sanjeev Kumar. Flash storage disaggregation. In *Proc. EuroSys*, 2016.

[26] Donald E. Knuth. *The TeXbook*. Addison-Wesley, Boston, 1986.

[27] Andres Lagar-Cavilla, Junwhan Ahn, Suleiman Souhlal, Neha Agarwal, Radoslaw Burny, Shakeel Butt, Jichuan Chang, Ashwin Chaugule, Nan Deng, Junaid Shahid, Greg

Thelen, Kamil Adam Yurtsever, Yu Zhao, and Parthasarathy Ranganathan. Software-defined far memory in warehouse-scale computers. In *Proc. ACM ASPLOS*, 2019.

[28] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Mass., 1986.

[29] Haifeng Li, Ke Liu, Ting Liang, Zuojun Li, Tianyue Lu, Hui Yuan, Yinben Xia, Yungang Bao, Mingyu Chen, and Yizhou Shan. Hopp: Hardware-software co-designed page prefetching for disaggregated memory. In *Proc. IEEE HPCA*, 2023.

[30] Hu Li. Introducing ”yosemite”: The first open source modular chassis for high-powered micro-servers, Jun 2015.

[31] Rui Lin, Yuxin Cheng, Marilet De Andrade, Lena Wosinska, and Jiajia Chen. Disaggregated data centers: Challenges and trade-offs. *IEEE Communications Magazine*, 58(2), 2020.

[32] Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, and Tongxin Bai. Imbalance in the cloud: An analysis on alibaba cluster trace. In *Proc. IEEE Big Data*, 2017.

[33] Priya Mahadevan, Sujata Banerjee, Puneet Sharma, Amip Shah, and Parthasarathy Ranganathan. On energy efficiency for enterprise and data center networks. *IEEE Communications Magazine*, 49(8):94–100, 2011.

[34] Hasan Al Maruf and Mosharaf Chowdhury. Effectively prefetching remote memory with leap. In *Proc. USENIX ATC*, 2020.

[35] Dimosthenis Masouros, Christian Pinto, Michele Gazzetti, Sotirios Xydis, and Dimitrios Soudris. Adrias: Interference-aware memory orchestration for disaggregated cloud infrastructures. In *Proc. IEEE HPCA*, 2023.

[36] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. IEEE INFOCOM*, 2010.

[37] Howraa M. Mohammad Ali, Ali M. Al-Salim, Ahmed Q. Lawey, Taisir El-Gorashi, and Jaafar M. H. Elmirghani. Energy efficient resource provisioning with vm migration heuristic for disaggregated server design. In *Proc. ICTON*, 2016.

[38] Howraa M. Mohammad Ali, Ahmed Q. Lawey, Taisir E. H. El-Gorashi, and Jaafar M. H. Elmirghani. Energy efficient disaggregated servers for future data centers. In *Proc. EuroSys NOC*, 2015.

[39] Howraa M. Mohammad Ali, Ahmed Q. Lawey, Taisir E. H. El-Gorashi, and Jaafar M. H. Elmirghani. Energy efficient disaggregated servers for future data centers. In *Proc. EuroSys NOC*, 2015.

[40] Howraa Mehdi Mohammad Ali, Taisir E. H. El-Gorashi, Ahmed Q. Lawey, and Jaafar M. H. Elmirghani. Future energy efficient data centers with disaggregated servers. *Journal of Lightwave Technology*, 35(24), 2017.

[41] Howraa Mehdi Mohammad Ali, Taisir E. H. El-Gorashi, Ahmed Q. Lawey, and Jaafar M. H. Elmirghani. Future energy efficient data centers with disaggregated servers. *Journal of Lightwave Technology*, 35(24):5361–5380, 2017.

[42] Nicolas. Mini-cluster part iv: Word count benchmark, 2015.

[43] Vlad Nitu, Boris Teabe, Alain Tchana, Canturk Isci, and Daniel Hagimont. Welcome to zombieland: Practical and energy-efficient memory disaggregation in a datacenter. In *Proc. EuroSys*, 2018.

[44] Albert Pagès, Jordi Perelló, Fernando Agraz, and Salvatore Spadaro. Optimal vdc service provisioning in optically interconnected disaggregated data centers. *IEEE Communications Letters*, 20(7), 2016.

[45] Antonios D. Papaioannou, Reza Nejabati, and Dimitra Simeonidou. The benefits of a disaggregated data centre: A resource allocation approach. In *Proc. IEEE GLOBECOM*, 2016.

[46] DA Popescu, Noa Zilberman, and Andrew Moore. *Characterizing the impact of network latency on cloud-based applications performance.* 2017.

[47] Thomas Rauber, Gudula Rünger, Michael Schwind, Haibin Xu, and Simon Melzner. Energy measurement, modeling, and prediction for processors with frequency scaling. *The Journal of Supercomputing*, 70:1451–1476, 12 2014.

[48] Zhenyuan Ruan, Malte Schwarzkopf, Marcos K. Aguilera, and Adam Belay. AIFM: High-Performance, Application-Integrated far memory. In *Proc. USENIX OSDI*, 2020.

[49] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. Legoos: A disseminated, distributed OS for hardware resource disaggregation. In *Proc. USENIX OSDI*, 2018.

[50] Tensorflow. Tensorflow benchmarks: A benchmark framework for tensorflow.

[51] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E. Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: The next generation. In *Proc. EuroSys*, 2020.

[52] Chenxi Wang, Haoran Ma, Shi Liu, Yuanqi Li, Zhenyuan Ruan, Khanh Nguyen, Michael D. Bond, Ravi Netravali, Miryung Kim, and Guoqing Harry Xu. Semeru: A Memory-Disaggregated managed runtime. In *Proc. USENIX OSDI*, 2020.

[53] Chenxi Wang, Yifan Qiao, Haoran Ma, Shi Liu, Yiying Zhang, Wenguang Chen, Ravi Netravali, Miryung Kim, and Guoqing Harry Xu. Canvas: Isolated and adaptive swapping for multi-applications on remote memory, 2022.

[54] Qing Wang, Youyou Lu, and Jiwu Shu. Sherman: A write-optimized distributed b+tree index on disaggregated memory, 2021.

[55] David P Williamson and David B Shmoys. *The design of approximation algorithms.* Cambridge university press, 2011.

[56] Peter Wilson. The memoir class for configurable typesetting, 2016.

[57] Zi Yan, Daniel Lustig, David Nellans, and Abhishek Bhattacharjee. Nimble page management for tiered memory systems. In *Proc. ACM ASPLOS*, 2019.

[58] Abdolahad Noori Zehmakan. Bin packing problem: Two approximation algorithms, 2015.

[59] Georgios Zervas, Hui Yuan, Arsalan Saljoghei, Qianqiao Chen, and Vaibhawa Mishra. Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation [invited]. *Journal of Optical Communications and Networking*, 10(2), 2018.

[60] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. 2015.