Artificial Intelligence (AI)-Centric Management of Resources in Modern Distributed Computing Systems

Shashikant Ilager¹, Rajeev Muralidhar¹,², and Rajkumar Buyya¹

¹ Cloud Computing and Distributed Systems (CLOUDS) Laboratory
SCIS, The University of Melbourne, Australia
² Amazon Web Services (AWS), Australia

Abstract. Contemporary Distributed Computing Systems (DCS) such as Cloud Data Centers are large scale, complex, heterogeneous, and are distributed across multiple networks and geographical boundaries. Cloud applications have evolved from traditional workloads to microservices/serverless based even as underlying architectures have become more heterogeneous and networks have also transformed to software-defined, large, hierarchical systems. Fueling the pipeline from edge to the cloud, Internet of Things (IoT)-based applications are producing massive amounts of data that require real-time processing and fast response, especially in scenarios such as industrial automation or autonomous systems. Managing these resources efficiently to provide reliable services to end-users or applications is a challenging task. Existing Resource Management Systems (RMS) rely on either static or heuristic solutions that are proving to be inadequate for such composite and dynamic systems, as well for upcoming workloads that demand even higher bandwidth, throughput, and lower latencies. The advent of Artificial Intelligence (AI) and the availability of data have manifested into possibilities of exploring data-driven solutions in RMS tasks that are adaptive, accurate, and efficient. This paper aims to draw motivations and necessities for data-driven solutions in resource management. It identifies the challenges associated with it and outlines the potential future research directions detailing different RMS tasks and the scenarios where data-driven techniques can be applied. Finally, it proposes a conceptual AI-centric RMS model for DCS and presents the two use cases demonstrating the feasibility of AI-centric approaches.

Keywords: Distributed Computing, Resource Management, AI Techniques, Edge Computing, Cloud Computing

1 Introduction

Internet-based Distributed Computing Systems (DCS) have become an essential backbone of the modern digital economy, society, and industrial operations. Cloud services have recently transformed from monolithic applications to microservices, with hundreds or thousands of loosely-coupled microservices comprising an end-to-end application, along with newer computing paradigms like serverless processing and Function-as-a-Service (FaaS). On the other side of the spectrum, the emergence of the Internet of Things (IoT), diverse mobile applications, and cyber-physical systems such as smart grids, smart industries, and smart cities, has resulted in massive amounts of data being generated and has thus relentlessly increased the demand for computing resources [1]. According to Norton [2], it is expected that 21 billion IoT devices will be connected to
the internet by 2025. Computing models such as cloud and edge computing have revolutionized the way services are delivered and consumed by providing flexible on-demand access to services with a pay-as-you-go model. While the newer application and execution models like microservices and serverless computing [3] greatly reduce the complexities in the design and deployment of software components, recent studies have shown that they have a large impact on the efficiency of the underlying system architectures [4]. The underlying computing infrastructures of clouds have also transformed from traditional computer-based servers to more heterogeneous systems composed of CPUs, GPUs, FPGAs, and AI/ML accelerators. Additionally, increased connectivity and newer heterogeneous workloads demand distinct Quality of Service (QoS) levels to satisfy their application requirements[4],[5],[7]. These developments have led to building hyper-scale datacenters and complex multi-tier computing infrastructures that require new innovative approaches in managing resources efficiently and provide reliable services. Deployment of 5G and related infrastructures like dynamic network slicing for high bandwidth, high throughput, and low latency applications is only increasing these challenges.

Resource Management Systems (RMS) in DCSs are middleware platforms that perform different tasks such as resource provisioning, monitoring, workload scheduling, and many others. Building an efficient RMS for future distributed systems is challenging due to several factors. Resource management for the emerging class of applications, networks, and Cyber-Physical-Systems (CPS) is complex as it is hard to manually fine-tune different configuration parameters for optimality. For example, ‘just 10 pieces of equipment, each with 10 settings, would have 10 to the 10th power, or 10 billion, possible configurations — a set of possibilities far beyond the ability of anyone to test for real’ [8]. Emerging network technologies, including 5G and satellite networks, such as Amazon’s Project Kuiper and SpaceX’s StarLink have opened up new dimensions [9] and opportunities for applications that require high bandwidth, high availability and low latency. In addition, the availability of huge data and advancement in computing capabilities has witnessed the resurgence of Artificial intelligence (AI) techniques driving innovation across different domains such as healthcare, autonomous driving, and robotics [10], [11]. Training Machine Learning (ML) models itself need huge resources and is increasing exponentially and doubling every 3.4 months [12] (compared to Moore’s Law’ 2-year doubling period) for the largest AI models. To accommodate these rapid changes across different domains, the required resources are delivered through cloud and edge that are large scale, heterogeneous, and highly distributed. Furthermore, the presence of multi-tenancy in these platforms with different users having a wide variety of workload characteristics and applications add more challenges for RMS. Thus, providing the performance requirements in such a shared environment while increasing resource utilization is a critical problem [13],[14].

The existing RMS techniques from operating systems to large scale DCSs are predominantly designed and built using preset threshold-based rules, or heuristics. These solutions are static and often employ reactive solutions [15], and works reasonably well in the general case but cannot adjust to the dynamic contexts [16]. Moreover, once deployed, they considerably fail to adapt and improve themselves in the runtime. In complex dynamic environments such as cloud and edge, they are incapable of capturing
the infrastructure and workload complexities and hence fall through. Consequently, it is envisioned that AI-centric approaches built on actual data and measurements collected from respective DCS environments are promising, perform better, and can adapt to dynamic contexts. Unlike heuristics, these models are built based on historical data, and accordingly, can employ proactive measures by foreseeing the potential outcome based on current conditions. For instance, a static heuristic solution for scaling resources uses workload and system load parameters to trigger the scaling mechanism, however, this reactive scaling diminishes the QoS for a certain period (due to time required for trigger application and boot-up). An ML model can predict future demand and scale the resources as needed to provide better QoS and user experience. Such techniques are highly valuable for service providers to provide reliable services and retain the business lead in the market. In addition, AI-centric RMS can be continuously improved with techniques like Reinforcement Learning (RL) [17] that uses the monitoring and feedback data in runtime. These adaptive approaches can improve RMS’s decisions and policies responding to the current demand, workload, and underlying system status.

AI-centric RMS in DCS is more feasible now than ever for multiple reasons. Firstly, AI techniques have matured and have proven to be efficient in many critical domains such as computer vision, natural language processing, healthcare applications, and autonomous vehicles. Secondly, most DCS platforms produce enormous amounts of data that are currently pushed as logs for debugging purposes or failure-cause explorations. For example, Cyber-Physical-Systems (CPS) in data centers already have hundreds of onboard CPU and external sensors monitoring workload, energy, temperature, and weather parameters. Data from such systems can be used to build ML models for performance prediction, anomaly detection, etc. Finally, increasing scale in computing infrastructure and complexities require automated systems that can produce the decisions based on the data and key insights from the experience for which AI models are ideal.

In this regard, this paper makes the following key contributions: (1) We present the evolution and the state-of-the-art RMS techniques in DCS and enlist the challenges associated with data-driven RMS methods. (3) We then identify the future research directions and point out the different tasks in which AI-centric methods can apply. (4) Subsequently, we propose a conceptual AI-centric RMS model and demonstrate two real-time use-cases applying data-driven AI methods related to energy-efficient GPU clock configurations and management of resources in data centers.

The rest of the paper is organized as follows. Section 2 gives an overview of DCS evolution and state-of-the-art practices in RMS. Section 3 identifies the challenges associated with AI-centric RMS. Section 4 draws future research directions. A conceptual AI-centric RMS model is proposed in Section 5. Section 6 presents the feasibility of AI-centric methods using two use cases. Finally, the conclusion is drawn in Section 7.

2 Evolution of DCS and the State-of-the-Art

2.1 DCS Evolution

An overview of the evolution of major DCSs is given in Fig. 1. Early cluster and supercomputing systems have been widely used in the scientific domain where applications are composed into parallel tasks (distributed jobs in grid computing) and executed on one or multiple clusters. The development of service-orientated computing
Fig. 1. An overview of contemporary DCS evolution (Timeline shows approximate time of the genesis of the system and became mainstream with some overlapping. The points shown for all dimensions are representative but not exhaustive and only lists the important facets.)

technologies (Web services, REST, SOAP, etc.), virtualization technologies, and demand for utility-oriented services created the current Cloud computing-based data centers.

The next decade of DCSs will be driven by emerging workloads like AI/ML across different end-user applications and IoT-based systems that need to process enormous amounts of data and derive meaningful intelligence from it. These IoT-based applications composed of a large number of sensors and computing nodes distributed across different network layers from Edge to remote Cloud and requires an autonomic sense-actuate model [1], where application tasks are composed, deployed, and executed autonomously. These requisites additional machine-to-machine interactions (compared to the current human-to-machine interactions) compelling rapid resource provisioning, scheduling, and task execution along with managing application’s demand for QoS and low latency.

In parallel to system advancements, application models have continued to evolve and create new software design approaches like microservices and execution models like serverless or Function as Service (FaaS) computing. To that end, managing these modern resources and applications requires intelligent decisions enabled from the data-driven AI-centric solutions. Although AI techniques can be applied in many aspects of these different DCS platforms, in this paper, we mainly focus on the Cloud and Edge and keep our discussions and illustrations around these two important paradigms.
2.2 State-of-the-art

With increased scale and complexities in the next generation DCSs, traditional static or heuristics solutions are becoming inadequate as these methods require careful hand pruning and human intervention to adapt to the dynamic environments [8], [15]. Consequently, AI-centric data-driven solutions are promising and there have been many attempts in recent years to address the resource management problems using data-driven AI solutions [16]. For example, Google has achieved a 40% efficiency in managing its cooling infrastructure using simple ML techniques and learning from historical data [8]. Many other techniques explored problems such as device placement, scheduling, and application scaling using data-driven methods [18]. At the system architecture level, [19] used massive data sets of hardware performance counter profiles collected at the datacenter level to reason about specific patterns that affect the front-end performance of large servers in Google data centers and used this data to mitigate front-end stalls of such warehouse-scale systems.

However, data-driven AI solutions in this domain are in its superficial stage and require meticulous attention to address the challenges they pose and simultaneously identify potential avenues in which these methods can be incorporated. Moreover, it is important to build the general frameworks and standards to adopt AI solutions in resource management that are scalable and manageable.

3 Challenges

In this section, we identify and describe the critical issues associated with the adoption of AI solutions in the resource management of distributed computing systems.

3.1 Availability of Data

The success of machine learning techniques is determined by the quality of the data used to train the models. The features in data should be available in large quantities with proper preprocessing from the expert domain knowledge [20] [21]. Within DCS, multiple challenges exist concerning the availability of such data. First, different resource abstraction platforms collect the data at different granularity. The physical machine-level data from on-board sensors and counters is gathered and accessed by tools like Intelligent Platform Management Interface (IPMI), while at a higher abstraction level, middleware platforms collect data related to workload level, user information, and surrounding environmental conditions (temperature, cooling energy in the data center). Also, network elements such as SDN controllers collect data related to network load, traffic, and routing. Unifying these data together and preprocessing it in a meaningful way is a complex and tedious task as the respective tools gather the data in a different format without common standards between them. Hence, building data-pipelines combining different subsystems’ data is crucial for the flexible adoption of ML solutions. Secondly, current monitoring systems collect data and push them into logging repositories only to be used later for debugging. However, converting this data for ML-ready requires monotonous data-engineering. Hence, future systems should be explicitly designed to gather the data that can be directly fed to the ML models with min-
imal data engineering and preprocessing effort. Lastly, although there are several publicly available datasets representing workload traces, there are hardly any public datasets available that represent different infrastructure including physical resources, energy footprints, and several other important parameters (due to privacy agreements). Therefore, getting access to such data is a challenge and needs collaborative efforts and data management standards from the relevant stakeholders. As described in [22], handling such diverse data openly and collaboratively requires standardized data formats and domain-specific frameworks.

3.2 Managing the Deployment of Models
Managing the life cycle of AI models is a challenging task. It includes deciding how much to train, where to train, and deploy for inference in multi-tier computing architectures like Edge/Fog. As resources have limited capabilities at a lower level and should be allocated to needful applications, if these available resources are predominantly used for training the ML models or running the RL agents, the latency-sensitive applications will be left with a small number of resources. On the other hand, if the models (RL agents) are trained or deployed in resource enriched cloud, the latency to push the inference decisions or the runtime feedback data to edge nodes shoots up, thus creating a delay-bottlenecks in RMS decisions. Furthermore, ML models tend to learn excessively with the expense of massive computational resources. Therefore, innovative solutions are needed to decide how much learning is sufficient based on specific constraints (resource budget, time-budget, etc.) and also rely on adaptive thresholds for accuracy of ML models [23]. To overcome this, techniques like transfer learning and distributed learning can be applied to reduce computational demands [20]. In addition, inferencing can be done on CPUs, GPUs, and domain-specific accelerators like Google TPU, Intel Habana, and/or FPGAs.

3.3 Non-Deterministic Outputs
Unlike statistical models that provide deterministic outputs, ML models are intrinsically exploratory and depend on stochasticity for many of its operations, thus producing non-deterministic results. For example, cognitive neural nets which are basic building blocks for many regression, classification, and Deep Learning (DL) algorithms primarily rely on principles of stochasticity for different operations (stochastic gradient descent, exploration phase in RL). When run multiple times with the same inputs, they tend to approximate the results and produce different outputs [10]. This may pose a serious challenge in DCS where strict Service Level Agreements (SLAs) govern the service delivery requiring deterministic results. For example, if a service provider fixes a price based on certain condition using ML models, consumers expect the price should be similar in all the time under similar settings. However, ML models may have a deviation in pricing due to stochasticity creating transparency issues between users and service providers. Many recent works are focusing on this issue introducing techniques like bringing constraints within neural nets to produce the deterministic outputs [24], yet, stochasticity in ML model is inherent and requires careful control over the output it produces.
3.4 Black Box Decision Making

AI techniques, specifically, the ML model’s decision-making process follows a completely black-box approach and fail to provide satisfactory reasons for its decisions. The inherent probabilistic architectures and complexities within ML models make it hard to avoid the black-box decisions. It becomes more crucial in an environment such as DCS where users expect valid feedback and explanation for any action taken by the service provider that is instrumental to build the trust between service providers and consumers. For instance, in case of a high overload condition, it is usual that service provider shall preempt a few resources from certain users with the expense of some SLA violations; however, choosing which user’s resources to be preempted with fairness and simultaneously providing the valid reasons is crucial in business-driven environments. Many works have been undertaken to build the explanatory ML models (Explainable AI - XAI) to address this issue [25], [26], however, solving this satisfactorily remains a challenging task.

3.5 Lightweight and Meaningful Semantics

DCS environments having heterogeneous resources across the multi-tiers interacting with different services and hosting complex applications create an obstacle to meaningfully represent all entities. Existing semantic models are heavy and inadequate for such complex environments. Therefore, lightweight semantic models are needed to represent the resource, entities, applications, and services without introducing overhead in meaningful data generation and parsing [27].

3.6 Complex Network Architectures, Overlays, and Upcoming Features

Network architectures across DCS and telecom networks are evolving rapidly using software-defined infrastructure, hierarchical overlay networks, Network Function Virtualization (NFV), and Virtual Network Functions (VNF). Commercial clouds like those of Amazon, Google, Microsoft have recently also partnered with telecom operators around the world to deploy ultra-low latency infrastructure (AWS Wavelength and Azure Edge Zone, for example) for emerging 5G networks. These 5G deployments and resource provisioning for high bandwidth, throughput, and low latency response through dynamic network slicing require a complex orchestration of network functions. RMS in future DCS needs to consider these complex network architectures, the overlap between telecom and public/private clouds and service function orchestration to meet end-to-end bandwidth, throughput, and latency requirements. Additionally, there is an explosion of data from such network architectures across the control plane, data plane, and signaling, which can be used to glean meaningful insights about network performance, reliability, and constraints that are important for end-to-end applications. As different types of data generated in different abstraction levels, standardized well-agreed upon data formats and models for each aspect need to be built.

3.7 Performance, Efficiency and Domain Expertise

Many ML algorithms and RL algorithms face performance issues. Specifically, RL algorithms face the cold-start problem as they spend a vast amount of the initial phase in
exploration before reaching its optimal policies creating an inefficient period where the decisions are suboptimal, even completely random or incorrect leading to massive SLA violations [20]. RL-based approaches also face several other challenges in the real world including: (1) need for learning on the real system from limited samples (2) safety constraints that should never or at least rarely be violated, (3) need of reward functions that are unspecified, multi-objective, or risk-sensitive, (4) inference that must happen in real-time at the control frequency of the system [28]. In addition, AI models are designed with a primary focus on accuracy-optimization resulting in a massive amount of energy consumption [14]. Consequently, new approaches are needed to balance the tradeoffs between accuracy, energy, and performance overhead. Furthermore, current ML algorithms are designed to solve the computer vision problems and adapting them to RMS tasks needs some degree of transformation the way input and outputs are interpreted. For example, most of the neural network architectures/libraries currently designed to solve computer vision problems. Many AI-centric RMS algorithms must transform their problem space and further use simple heuristics to interpret the result back and apply to the RMS problems. Such complexities require expertise that is not directly related to the DCS but also from other domains. Thus, newer approaches, algorithms, standardized frameworks, and domain-specific AI techniques are required for energy-efficient AI for RMS.

4 Future Research Directions

Despite the challenges associated, AI solutions provide many opportunities to incorporate these techniques into RMS and benefit from them. In this section, we explore different avenues where these techniques can be applied in the management of distributed computing resources.

4.1 Data-driven Resource Provisioning and Scheduling

Resource provisioning and scheduling are basic elements of an RMS. Usually, resources are virtualized, and specifically, computing resources are delivered as Virtual machine (VM) or lightweight containers. The problems related to provisioning such as estimating the number of resources required for an application, co-locating workloads based on their resource consumption behaviors and several others can be addressed using AI techniques. These techniques can be extended to special case provisions such as spot instances. Utilizing spot instances for application execution needs careful estimation of application run time (to avoid the state-corruption or loss of computation if resources are preempted) and accordingly deciding resource quantity and checkpointing logic. This requires building prediction models based on previous execution performance counters or co-relating with clusters based on existing knowledge-base [29].

In edge computing environments, RMS should utilize resources from multi-tier infrastructure, and selecting nodes from different layers also requires intelligence and adaptation to application demands and infrastructure status [13]. Furthermore, AI solutions can be employed in many scheduling tasks such as finding an efficient node, VM consolidation, migration, etc. The prediction models based on historical data and adaptive RL models can be used to manage the scheduling and resource provisioning.
4.2 Managing Elasticity using Predictive Analytics

Elasticity is an essential feature providing flexibility by scaling up or scaling down the resources based on applications QoS requirements and budget constraints. Currently, elasticity is based on the reactive approaches where resources are scaled according to the system load in terms of the number of users or input requests. However, such reactive measures diminish the SLAs due to boot-up time and burst loads. In contrast, forecasting the future load based on the application’s past behaviors and proactively scaling the resources beforehand vastly improves SLAs and saves costs. Essentially, it needs time-series analysis to predict future load using methods like ARIMA or more advanced RNN techniques such as LSTM networks that are proven to be efficient in capturing the temporal behaviors [30]. Such proactive measures from service providers enable efficient management of demand-response without compromising the SLAs.

4.3 Energy Efficiency and Carbon-footprint Management

One of the major challenges in computing in recent years has been energy consumption. Increasing reliance on computing resources has created a huge economic and environmental impact. It is estimated that by 2025, data centers itself would consume around 20% of global electricity and emit up to 5.5% of the world’s carbon emissions [31]. Energy efficiency can be achieved at different levels from managing hardware circuits to data center level workload management. Recent studies have shown promising results using AI techniques in device frequency management [29], intelligent and energy-efficient workload management (scheduling, consolidation), reducing cooling energy by fine-tuning cooling parameters [8], and executing applications within power budgets [15], etc. In addition, it can also be effectively used in minimizing the carbon-footprints by renewable energy forecasting and accordingly shifting the workloads across clouds. Each of these subproblems can be addressed by using a combination of predictive and RL models based on application scenarios and requirements.

4.4 Security Management

As cyber-systems have become sophisticated and widely interconnected, preserving the privacy of data, and securing resources from external threats is quintessential. There has been widespread use of ML algorithms in many aspects of security management. It includes AI-based intruder detection systems, anomaly detection [32] [33] for identifying deviations in the application/resource behaviors. Techniques including Artificial Neural Networks (ANNs), ensemble learning, Bayesian networks, Association rules, and several classification techniques like SVM can be effectively utilized to address different security problems [34]. They can also be predominantly used in preventing DDoS attacks by analyzing traffic patterns [35]. Such measurements will vastly help to manage the resources securely and thus increasing the reliability of the systems.

4.5 Managing Cloud Economics

Cloud economics is a complex problem and requires vast domain knowledge and expertise to price services adequately. It is also important for consumers to easily under-
stand pricing models and estimate the cost for their deployments. Current pricing models largely depend on subscription types, e.g., reserved, on-demand, or spot instances. The pricing for these subscription models is driven by standard economic principles like auction mechanisms, cost-benefit analysis, profit and revenue maximizations, etc. These pricing problems are solved using operation research (OR) techniques or stochastic game theory approaches [36]. However, such methods are largely inflexible and they either overprice the services or it results in loss of revenues for cloud service providers [36]. ML models can be applied to forecast the resource demand and accordingly excessive resources can be pooled in the open market for consumers. Besides, pricing can be more dynamic based on this forecasted demand-response that benefits both consumers and service providers.

4.6 Generating the Large-scale Data Sets

Machine learning models require large amounts of training data for improved accuracy. However, access to large scale data is limited due to privacy and lack of capabilities to generate a large quantity of data from infrastructure. For this, AI models itself can be used to generate large-scale synthetic datasets that closely depict the real-world datasets. For instance, given a small quantity of data as input, Generative Adversarial Networks (GANs) can be used to produce large-scale data [37]. These methods are highly feasible in generating time-series data of DCS infrastructure. Moreover, these methods can also be leveraged to adequately produce incomplete datasets. Such large-scale data sets are necessary to train efficient predictive models and bootstrap the RL agents to achieve a reasonable efficiency in its policies.

4.7 Future System Architectures

Cloud services have recently undergone a shift from monolithic applications to microservices, with hundreds or thousands of loosely-coupled microservices comprising the end-to-end application. In [4] the authors explore the implications of these microservices on hardware and system architectures, bottlenecks, and lessons for future datacenter server design. Microservices affect the computation to communication ratio, as communication dominates, and the amount of computation per microservice decreases. Similarly, microservices require revisiting whether big or small servers are preferable. In [19], the authors use an always-on, fleet-wide monitoring system to track front-end stalls, I-cache and D-cache miss (as cloud microservices do not lend them amenable to cache locality unlike traditional workloads) across hundreds and thousands of servers across Google’s warehouse-scale computers. The enormous amounts of data generated and analyzed help to provide valuable feedback to the design of the next-generation system architectures. Similarly, deep learning can be used to diagnose unpredictable performance in cloud systems. Data from such systems can thus be invaluable for the hardware and system architectures of future DCS.
4.8 Other Avenues

Along with the aforementioned directions, AI-centric solutions can be applied to several other RMS tasks including optimizing the heuristics itself [21], network optimizations (e.g. TCP window size, SDN routing optimization problems) [21], and storage infrastructure management [20]. Moreover, learning-based systems can be extended across different computing system stacks from lower levels of abstraction including hardware design, compiler optimizations, operating system policies to a higher level interconnected distributed systems [16].

5 Conceptual Model for AI-centric RMS

In the AI-centric RMS system, ML models need to be trained and deployed for the RMS usage for its different tasks. However, integrating data-driven models into DCS platforms in a scalable and generic manner is challenging and is still at a conception stage. In this regard, as shown in Fig. 2, we provide a high-level architectural model for such an AI-centric RMS. The important elements of this system are explained below. It consists of three entities:

**Users/ Applications:** Users requiring computing resources or services interact with the middleware using interfaces and APIs.

**AI-centric RMS Middleware:** This is responsible for performing different tasks related to managing user requests and underlying infrastructure. The RMS tasks continuously interact with the AI models for accurate and efficient decisions. The RMS needs to perform various tasks including provisioning of the resources, scheduling them on appropriate nodes, monitoring in runtime, dynamic optimizations like migrations, and consolidations [15] to avoid the potential SLA violations. Traditionally, these tasks are done using the algorithms implemented within the RMS system that would execute the policies based on the heuristics or threshold-based policies. However, in this AI-centric RMS, the individual RMS operations are aided with inputs from the data-driven models. The AI models are broadly categorized into two types, (1) predictive models, and (2) adaptive RL models. In the former, models are trained offline using supervised or unsupervised ML algorithms utilizing historical data collected from the DCS environment that includes features from resources, entities, and application services. This data is stored in databases and preprocessing, cleaning, normalizing is done to suit the requirements of models. The offline training can be done on remote cloud nodes to benefit from the specialized powerful computing resources. The trained models can be deployed on specialized inference devices like Google Edge TPU and Intel Habana. Choosing the optimal place and deciding where to deploy these ML models depends on where the RMS engine is deployed in the environment and this is itself a challenging research topic that should be addressed as described in Section 3.2. In the latter case, run-time adaptive models such as Reinforcement Learning (RL) that continue to improve their policies based on agent’s interactions and system feedback that requires both initial learning and runtime policy improvement methods that need to be updated after every episode (certain time reaching to terminal state). The RMS operations can interact with both the predictive and RL-based data-driven models using the RESTful APIs in runtime [15] and RL agents can directly interact with the environments.
Fig. 2. Conceptual AI-centric RMS Model

**DCS Infrastructure:** The computing infrastructure comprises heterogeneous resources including gateway servers, cloudlets, edge micro data centers, and remote clouds. Data-collector service in each of these infrastructures should interact with the monitoring component in RMS middleware to provide sufficient data and feedback for AI-centric models. Therefore, adopting the data-driven AI-centric RMS models needs a significant change in the way current RMS systems are designed and implemented, as well as data collection methods, interfaces, and deployment policies that can be easily integrated into existing environments.

### 6 Case Studies

In this section, we present two use cases that have applied AI techniques to two different problems: (1) configuration of device frequencies for energy-efficient workload scheduling in cloud GPUs, (2) management of data center resources.

#### 6.1 Data-Driven GPU Clock Configuration and Deadline-aware Scheduling

Graphics Processing Units (GPUs) have become the de-facto computing platform for advanced compute-intensive. Additionally, ML models itself are massively reliant on the GPUs for training as they provide efficient SIMD architectures that are highly suitable for parallel computations. However, the energy consumption of GPUs is a critical problem. Dynamic Voltage Frequency Scaling (DVFS) is a widely used technique to
reduce the dynamic power of GPUs. Yet, configuring the optimal clock frequency for essential performance requirements is a non-trivial task due to the complex nonlinear relationship between the application’s runtime performance characteristics, energy, and execution time. It becomes even more challenging when different applications behave distinctively with similar clock settings. Simple analytical solutions and standard GPU frequency scaling heuristics fail to capture these intricacies and scale the frequencies appropriately. In this regard, in our recent work [29], we have proposed a data-driven frequency scaling technique by predicting the power and execution time of a given application over different clock settings. Furthermore, using this frequency scaling by prediction models, we present a deadline-aware application scheduling algorithm to reduce energy consumption while simultaneously meeting their deadlines.

The high-level overview of the system is given in Fig. 3. It is broadly classified into two parts, predictive modeling, and data-driven scheduler. In the first part, we collect the training data that consists of three parts, profiling information, energy-time measurements, and respective frequency configurations. We then predict two entities for a given application and frequency configuration, i.e., energy consumption and execution time. Subsequently, in the second part, the new applications arrive with the deadline requirements and minimal profiling data from a default clock frequency execution. The scheduler finds correlated application data using the clustering technique, and this data is used for predicting the energy and execution time over all frequencies. Finally, based on the deadline requirements and energy efficiency, the scheduler scales the frequencies and executes the applications.

We use twelve applications for evaluation from two standard GPU benchmarking suites, Rodinia and Polybench. The training data is generated from profiling the applications using nvprof, a standard profiling tool from NVIDIA. We collected around 120 key features representing key architectural, power, and performance counters. To build the predictive models, we explored several regression-based ML models including Linear Regression (LR), lasso-linear regression (Lasso), and Support Vector Regression (SVR). Also, ensemble-based gradient boosting techniques, extreme Gradient Boosting (XGBoost), and CatBoost. The goal is to build energy and execution time prediction models for each GPU device to assist the frequency configuration.

We conduct extensive experiments on NVIDIA GPUs (TESLA P100) on Grid5000 testbed. The experiment results have shown that our prediction models with CatBoost...
have high accuracy with the average Root Mean Square Error (RMSE) values of 0.38 and 0.05 for energy and time prediction, respectively (Fig. 4). Also, the scheduling algorithm consumes 15.07% less energy (Fig. 5) as compared to the baseline policies (default and max clock) while meeting the application deadlines. It is because our approach can scale the frequencies that have energy-efficient settings (Fig. 6) also able to meet performance requirements. More details on prediction-models, scheduling algorithms, and implementation can be found in [29].

6.2 Industrial Cloud Data Center Management

A cloud data center is a complex Cyber-Physical System (CPS) that consists of numerous elements including thousands of rack-mounted physical servers, networking equipment, sensors monitoring server, and room temperature, a cooling system to maintain acceptable room temperature, and many facility-related subsystems. It is one of the highest power density CPS of up to 20 kW per rack thus dissipating an enormous amount of heat. This poses a serious challenge to manage resources energy efficiently and provide reliable services. Optimizing data center operation requires tuning hundreds of parameters belonging to different subsystems where heuristics or static solutions fail to yield a better result. Even a 1% improvement in data center efficiency leads to savings in millions of dollars over a year and reduce the carbon footprints. Therefore,
optimizing these data centers using potential AI techniques is of great importance. Accordingly, we discuss two real-time AI-centric RMS systems built by researchers at Google and Microsoft Azure Cloud.

ML-centric cloud [15] is an ML-based RMS system at an inception stage from the Microsoft Azure cloud. They built Resource Control (RC)- a general ML and prediction serving system that provides the insights of workload and infrastructure for resource manager of Azure compute fabric. The input data collected from the virtual machine and physical servers. The models are trained using a gradient boosting tree and trained to predict the different outcomes for user’s VMs such as average CPU utilization, deployment size, lifetime, and blackout time. The Azure resource manager interacts with these models in runtime. For instance, the scheduler queries for virtual machine lifetime, and based on the predicted value, the appropriate decision is taken to increase infrastructure efficiency. Applying these models to several other resource management tasks is under consideration including power management inside Azure infrastructure.

Similarly, Google has also applied ML techniques to optimize the efficiency of their data centers. Specifically, they have used ML models to configure the different knobs of the cooling system thus saving a significant amount of energy [8]. The ML models are built using simple neural networks and trained to improve the PUEs (Power Usage Effectiveness), a standard metric to measure the data center efficiency. The input features include total IT workload level, network load, parameters affecting the cooling system like outside temperature, wind speed, number of active chillers, and others. The cooling subsystems are configured according to the predictions and results have shown that 40% of savings are achieved in terms of their energy consumption.

Therefore, the brief use cases presented here strongly attest to the feasibility of AI-centric solutions in different aspects of resource management of distributed systems.

7 Conclusions

Future distributed computing platforms will be complex, large scale, and heterogeneous enabling the development of highly connected resource-intensive business, scientific, and personal applications. Managing resources in such infrastructure require AI-centric approaches to derive key insights from the data, learn from the environments, and take resource management decisions accordingly. In this paper, we investigated challenges in managing distributed computing resources using AI approaches and proposed many research directions. We also provided two use cases demonstrating the feasibility of AI-centric RMS. We envision that advanced and sophisticated AI tools can be widely applied in numerous RMS tasks. Such AI-centric approaches will enable better hardware design, efficient middleware platforms, and reliable application management in future distributed computing platforms.

8 References


