3-2023

Green data analytics of supercomputing from massive sensor networks: Does workload distribution matter?

Zhiling GUO
Singapore Management University, ZHILINGGUO@smu.edu.sg

Jin LI
Xi'an Jiaotong University

Ram RAMESH
State University of New York College at Buffalo - Buffalo State College

Citation

GUO, Zhiling; LI, Jin; and RAMESH, Ram. Green data analytics of supercomputing from massive sensor networks: Does workload distribution matter?. (2023). *Information Systems Research*. 34, (4), 1664-1685.
Available at: https://ink.library.smu.edu.sg/sis_research/7813

Green data analytics of supercomputing from massive sensor networks: Does workload distribution matter?

Zhiling Guo,[a] Jin Li,[b],* Ram Ramesh[c]

[a] School of Computing and Information Systems, Singapore Management University, Singapore 178902, Singapore;
[b] School of Management, Xi'an Jiaotong University, Xi'an 710049, China;
[c] Department of Management Science and Systems, State University of New York, Buffalo, New York 14260
*Corresponding author
Contact: zhilingguo@smu.edu.sg, https://orcid.org/0000-0002-9058-4710 (ZG); jinlimis@xjtu.edu.cn, https://orcid.org/0000-0003-3340-1516 (JL); rramesh@buffalo.edu (RR)

Published in Information Systems Research (2023 March) DOI: 10.1287/isre.2023.1208

Abstract. Energy costs represent a significant share of the total cost of ownership in high- performance computing (HPC) systems. Using a unique data set collected by massive sensor net-works in a petascale national supercomputing center, we first present an explanatory model to identify key factors that affect energy consumption in supercomputing. Our analytic results show that, not only does computing node utilization significantly affect energy consumption, workload distribution among the nodes also has significant effects and could effectively be lever-aged to improve energy efficiency. Next, we establish the high model performance using in-sample and out-of-sample analyses. We then develop prescriptive models for energy-optimal runtime workload management and extend the models to consider energy consumption and job performance tradeoffs. Specifically, we present four dynamic resource management methodologies (packing, load balancing, threshold-based switching, and energy optimization), model their application at two levels (purely within-rack and jointly cross-rack resource allocation), and explore runtime resource redistribution policies for jobs under the emergent principle of computational steering and comparatively evaluate strategies that use computational steering with those that do not. Our experimental studies show that packing is preferred when the total workload of a rack is higher than a threshold and load balancing is preferred when it is lower. These results lead to a threshold strategy that yields near-optimal energy efficiency under all workload conditions. We further calibrate the energy-optimal resource allocations over the full range of workloads and present a bicriteria evaluation to consider energy consumption and job performance tradeoffs. We demonstrate significant energy savings of our proposed strategies under various workload conditions. We conclude with implementation guidelines and policy insights into energy- efficient computing resource management in large supercomputing data centers.

Keywords: high-performance computing, data center, energy-efficient operation, data analytics, autoregressive model, dynamic panel data, optimization

Funding: The authors thank the National Supercomputing Center of Singapore for enabling and sup-porting this research. J. Li received support from the National Natural Science Foundation of China [Grants 71901169, 72032006, and T2293774].

1. Introduction

High-performance computing (HPC) involves a network of supercomputing nodes that are primarily intended for solving computational problems that are either too large or too complex for traditional computing platforms. Some of the widely used HPC applications include parallel com-puting for scientific research and solving data-intensive industrial problems. A recent report from Markets and Markets expects the global HPC market to grow from USD 32.11 billion in 2017 to USD 44.98 billion by 2022.1This growth rate also indicates a concomitant increase in the energy consumption by data centers worldwide. To illustrate, there were approximately 7.2 million data centers worldwide in 2021.2Data centers currently con-sume about 2% of the total global electricity, which is expected to reach 8% by 2030 (Drozdiak 2020). The global data center industry has been one of the fastest-growing energy-consuming sectors of the economy and a significant source of carbon emissions (World Economic Forum 2020). The rise in energy costs and the urgent need to reduce the carbon footprint together create an imperative for new approaches to achieve energy efficiency in data centers.In both traditional and HPC data centers, information technology (IT) devices are the major energy consumers (Shuja et al. 2012). About 80% of electricity use in a data center is attributable to servers, among which 40%–50% comes from the energy needed to extract heat from and cool its racks of servers (Info-Tech Research Group 2007, Emerson Network Power 2009). Notably, *server load* is a key factor in energy consumption. Compared with traditional data center architectures, an HPC data center requires denser banks of computing nodes to minimize communication latency and increase capacity. A typical HPC data center hosts a series of server racks where each rack commonly hosts a large number of physical servers. These closely located servers not only consume large amounts of electrical power by themselves but also inevitably generate significant amounts of heat that require advanced rack-wide cooling systems (Chetsa et al. 2014). These observations together suggest that as much as server load is a major contributor to energy costs, *load distribution* among the servers could be another significant factor in heat dissipation and total energy consumption. The principal motivation of this research is therefore to investigate the impact of load and distribution factors on the overall energy consumption in HPC data centers and develop energy-efficient workload management strategies.

The system architecture and the service framework of an HPC data center are different from those of other computing contexts. In the HPC context, each server has a fixed number of *cores*, where a core can be viewed as an independent hardware computational unit that is fully self-contained with its own memory and interfaces. In the cloud and cluster computing contexts, a virtual machine (VM) is a fully software-defined analogue of a core. Although VMs can be created and terminated as needed in a traditional physical server, the number of cores available in an HPC server is fixed, and a core in itself could host several VMs for an application. Despite the structural differences, resource allocation problems in these data centers are conceptually similar. An HPC application would request a fixed number of cores rather than VMs for its execution. The job scheduling software allocates cores across a set of HPC servers to requesting applications based on some nontransparent algorithms. In this study, we develop energy-efficient core allocation strategies to support green data center operations. We focus on the following research questions. (i) What are the key workload-related factors that significantly affect an HPC data center's overall energy consumption? (ii) How does one estimate with reliability and validity the overall energy use from these factors? (iii) How does one use these estimates to optimally allocate cores to jobs such that the overall energy consumption is minimized? (iv) How does one evaluate the impact of workload distribution on job performance and consider the energy-performance tradeoffs in core allocations?

Our research originates from and is based on the operations of the National Supercomputing Center (NSCC) of Singapore. Using a large network of system sensors, NSCC collects fine-grained IT load data and the corresponding facility data almost on a continuous basis. Using these data, we first carry out a dynamic panel data analysis to investigate how the server utilization and workload distribution on a rack contribute to the total energy consumption of the data center. We then establish the model performance using in-sample and out-of-sample analyses and further develop effective strategies to achieve energy-optimal runtime core allocations. We find that not only does server utilization significantly affect data center energy consumption, distribution of workload also has a significant effect on energy efficiency. When the total load of a rack is high, the load factor essentially dominates the load distribution factor in impacting energy consumption. The *packing strategy* that consolidates jobs to as few servers as possible would not generate significant disparity in the core allocations among servers and is thus the preferred allocation strategy. However, under lighter load conditions, the load distribution factor dominates the load factor. The *load-balancing strategy* that distributes the workload over the servers more evenly would bring significant savings in total energy consumption and thus is preferred. Based on these insights, we develop a *threshold-based strategy* to switch between packing and load balancing as the total load varies over time. We theoretically establish the conditions for the existence of a threshold and its bounds. We further propose an *energy-optimization strategy* that achieves the optimal runtime core allocations to minimize energy consumption. Finally, we extend the model to consider energy consumption and job performance tradeoffs.

In a dynamic environment characterized by significant workload variations, data centers should adopt flexible strategies for resource allocation. For this purpose, our research question (i) leads to an explanatory data analysis of the workload factors that impact energy consumption; question (ii) leads to a predictive model to measure the effects of changes in the workload and distribution variables on energy consumption; and using this model, questions (iii) and (iv) lead to the development of prescriptive resource allocation methods for runtime energy optimization and for the bicriteria consideration of energy consumption and job performance tradeoffs. We implement our proposed models by considering two strategic management options for an HPC data center, computational steering (CS) and no computational steering (NCS), and validate our model performance in the presence or absence of the job completion time effects under different core allocation strategies. Our extensive experimental studies based on both simulated data and NSCC data sets show consistent superior performance of the proposed prescriptive models. In particular, our results demonstrate 3.8% energy savings over the state-of-the-art commercial scheduler and as much as 3.3%

and 87.7% energy savings over the load-balancing and packing strategies, respectively.

In summary, although total computational load has been widely known as a driver of energy consumption, it has rarely been recognized in both industry and research that load distribution among servers can be a significant factor of energy use. Our important theoretical contribution is to not only uncover causes of energy consumption but also use them in energy-efficient scheduling of computing resources. Practically, we develop a comprehensive algorithmic framework for energy-efficient real-time load management and runtime assignment of cores to jobs in HPC data centers using dynamically generated big data from large-scale system-wide sensor networks. We further provide guidelines for data center administrators to consider the tradeoffs between energy consumption and job performance in resource allocations. Such an integrated approach to energy-efficient resource management does not exist in practice. The proposed data analytics framework and methodologies can be automated and embedded into the existing data center's resource management scheme to support green supercomputing operations. Online Appendix A summarizes our key research contributions to both theory and practice.

The organization of this paper is as follows. Section 2 reviews related literature. Section 3 introduces our data analytics framework and provides data description. Section 4 presents an explanatory model to identify key factors that affect energy consumption and further validates our predictive model performance. Section 5 develops prescriptive models for energy-efficient resource management and the extension to a bicriteria optimization of energy consumption and job performance. Section 6 conducts extensive computational experiments focusing on energy-efficient operations. Section 7 further provides bicriteria evaluation of efficient strategies and presents a decision tree guideline that leads to critical policy insights and practical implementation approaches. Section 8 concludes with directions for future research.

2. Related Literature

Our research is related to three broad topical areas: server workload management, energy-efficient data centers, and big data and green information systems (IS). We briefly review the related literature in each category and position our current research in their milieu as follows.

2.1. Server Workload Management

Because servers are the highest energy consuming components in data centers, virtual server consolidation is considered a key solution to effective power management, especially in the contexts of cloud, cluster, and grid computing environments (Chernicoff 2009, Buyya et al. 2010, Varasteh and Goudarzi 2015, Bermejo et al. 2019). Server virtualization enables the traditional data centers to run applications on various VMs, and server consolidation increases the average use of the physical machines (PMs) that host VMs. Normally VM consolidation problems can be formulated as a bin-packing problem, where job requirements of different VMs must be packed into a finite number of PMs to minimize the total number of PMs used (Korte and Vygen 2006). Speitkamp and Bichler (2010) propose a mathematical programming model for server consolidation with the consideration of quality-of-service levels. Cohen et al. (2019) model the resource allocation in the cloud as bin packing with chance constraints to guide job scheduling decisions. In a dynamic environment, VMs could be migrated to other PMs in response to workload variations (Beloglazov and Buyya 2010, Dabbagh et al. 2015) or energy considerations (Qiu et al. 2019). The consolidation algorithm can be triggered when a host PM's utilization reaches a threshold value (Gmach et al. 2009, Deng et al. 2014) or hotspots are identified (Ilager et al. 2019). We refer to Delorme et al. (2016) for a recent survey of the models and solution approaches.

Practically, workload management is supported by various online approximation algorithms, including next-fit, first-fit, best-fit, and best-reply algorithms (Coffman et al. 1996), as well as shortest job first (SJF), Tetris, and random policy (Hovestadt et al. 2003, Grandl et al. 2014). For example, SJF sorts jobs according to their execution time and schedules jobs with the shortest execution time first, and Tetris schedules job by a combined score of preferences for the short jobs and resource packing. As far as workload distribution is concerned, load balancing is often used to improve the data flows and workload distribution across multiple computing resources (Tang et al. 2018, Aghdashi and Mirtaheri 2019). Simple algorithms include random choice, round robin, or least connections that can be used to select which servers to allocate the jobs (Kushwaha and Gupta 2015). In this research, we examine both packing and load-balancing strategies in HPC data center workload management. When allocating cores to jobs, we adopt the first-fit and round-robin algorithms for its simplicity, flexibility, and performance.

2.2. Energy-Efficient Data Centers

Data center energy efficiency is now a chief concern for data center administrators (Dayarathna et al. 2016). There are two ways to improve the energy efficiency of HPC centers. The first involves hardware innovations on supercomputing infrastructure, and the second, which is the focus of this work, is software optimization to make more energy-efficient use of available resources (Schöne et al. 2014, Pitkin 2018). Notable industry examples of energy-efficient supercomputing include the European Union's Horizon 2020 project READEX,[3] which develops automated tool suites for dynamically tuning parameters in runtime (e.g., changing CPU core frequency) to achieve energy-efficient computing of HPC applications. Modern high-performance cluster nodes are equipped with dynamic voltage and frequency scaling technology, which enables automatic adjustment of CPU frequency based on hardware monitoring counters (e.g., applications and workloads execution phases, runtime statistics) to save energy (Chetsa et al. 2014). Different from these approaches, we focus on energy optimization through workload distribution to dynamically adjust HPC core allocation in our research.

Recent studies have proposed several power profiling methods to improve and optimize the energy consumption of servers in the HPC context. Ei-Moursy et al. (2019) build multiple linear regression models to estimate host utilization. Shoukourian et al. (2014) develop a model to calibrate the average power consumption metrics with respect to a given number of compute nodes. In contrast to these prior works that address server-level power profiling and chip-level energy efficiency, we focus on rack-level energy effects. As the servers are powered through racks, the total energy consumption of a rack depends on the aggregate power profile of the rack, which takes into account both the server utilization and distribution of cores across the computing nodes. We propose a dynamic panel regression model to perform energy profiling of the racks that enable us to focus on rack-level workload distribution for energy optimization.

Realizing that job assignment has thermal consequences, a few studies focus on controlling the temperature of cooling system in data centers to achieve the overall energy savings (Pakbaznia and Pedram 2009, Van Damme et al. 2019). Gupta et al. (2021) develop a workload and cooling management framework to set optimal temperatures for both chilled water and cold airflows. Akbar and Li (2022) consider thermal-aware computing components and devise a Shapley value-based workload to schedule tasks for minimizing the cooling cost. Chen et al. (2012) take into account data center cooling dynamics and design adaptable workload scheduling algorithm to minimize energy consumption across geographically distributed data centers. Different from this stream of research to manage temperature and cooling costs in an engineering system, we focus on policy-level core allocation decisions to guide dynamic workload management toward conserving energy in HPC data center operations.

Energy efficiency is often achieved via different strategies which may or may not support job migration. Generally, if job migration is allowed under the job scheduling policy, it is termed as computational steering (CS), a notion that introduces interactive steering of resources within the scope of real-time performance monitoring and adaptive control (Van Liere et al. 1997, Vetter and Reed 2000). In cloud data centers, CS is made possible by transferring a running VM from a PM to another PM without considerable service downtime (Huang et al. 2011). If service interruption is a concern, then noncomputational steering (NCS) policy is followed. Wolke et al. (2015) find that CS-enabled resource reallocation strategies achieve high levels of energy efficiency in different workload environments. In the HPC context, Atanasov et al. (2010) and Danani and D'Amora (2015) discuss implementations of CS solutions that direct or redirect the progress of an HPC application at runtime by modifying application-defined control parameters using a steering client application. We develop both the CS and NCS versions of our prescriptive core allocation strategies and discuss their performance implications in our HPC context.

2.3. Big Data Analytics and Green IS

The wireless sensor networks (WSNs) have evolved into the backbone of big data gathering and made it possible for large-scale data analytics (Rani et al. 2017). The distributed WSNs can track various environmental variables and generate a huge volume of streaming data in real time (Takaishi et al. 2014). In addition, the new Internet of Things applications have gained traction in the IS research community (Ketter et al. 2016). In modern data centers, the real-time data collected by network sensors shows great promise to optimize the scheduling solutions (Chatterjee et al. 2019). In this research, we pool the fine-grained, large-scale sensor data from multiple sources to obtain the real-time rack-level total energy consumption measures and workload distributional statistics, which are essential to support our data-driven analytics and runtime optimization of performance.

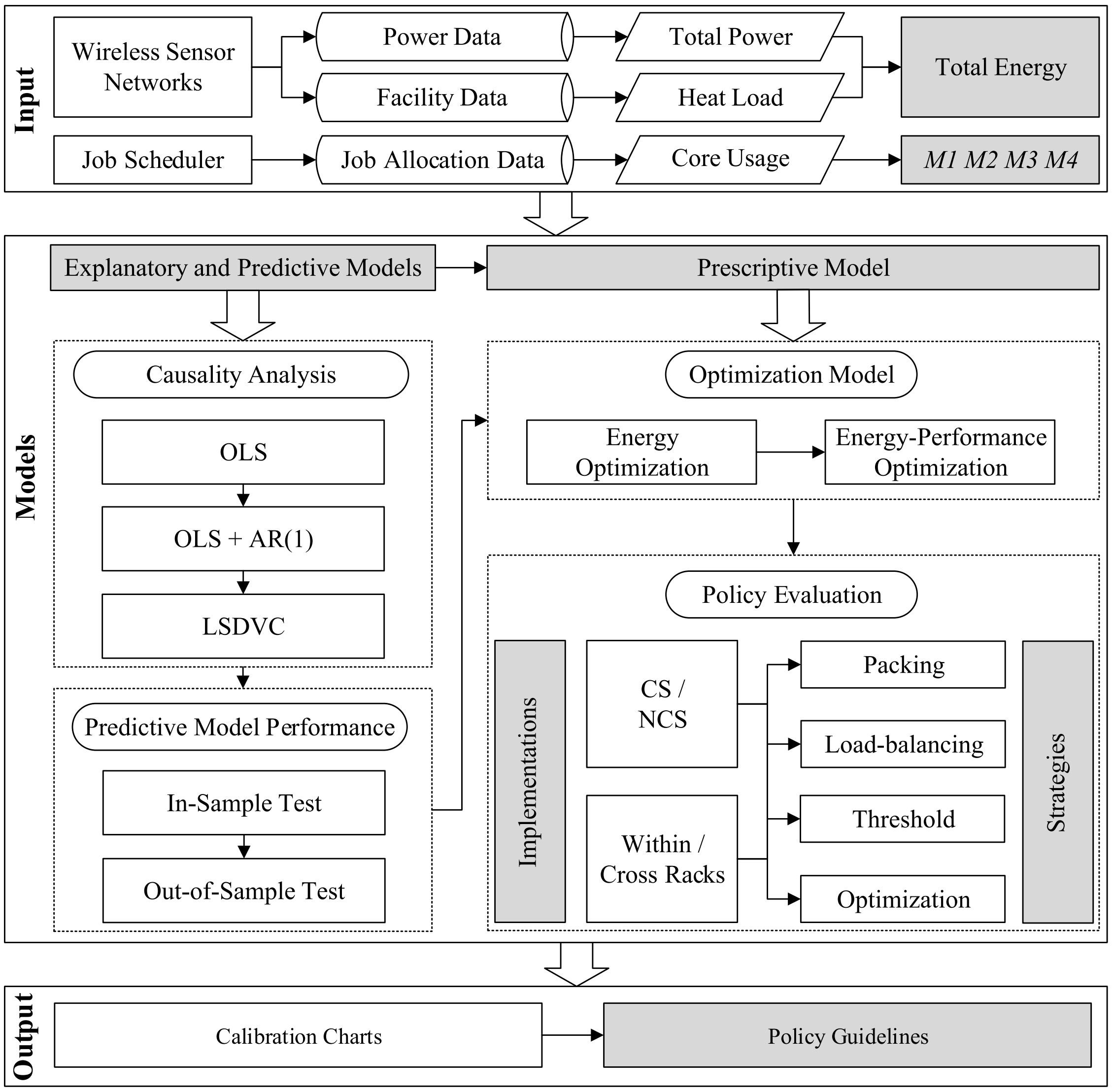
Methodological advances in data analytics and optimization have allowed large-scale data to be used for complex decision making in a variety of business research fields. The "predict-then-optimize" paradigm is one such new approach that first builds a predictive model using data and then embeds the model into the objective function of an optimization problem (Mišić and Perakis 2020). However, good out-of-sample prediction does not necessarily yield good out-of-sample decisions. We thus focus on econometrics modeling for understanding the causal impact of both load and load distribution factors on energy consumption and then embed this causal model within a data-driven prescriptive analytics framework to optimize energy-efficient resource allocation and job scheduling decisions. Watson et al. (2010) have proposed energy informatics as a core subfield of IS to reduce IT-related energy consumption and create a sustainable society. This research contributes to the growing field of green IS research that focus on the roles of information technologies and data analytics in tackling environmental sustainability problems (Melville 2010, Loock et al. 2013, Malhotra et al. 2013, vom Brocke et al. 2013, Loeser et al. 2017).

3. Analytics Framework and Data Description

Figure 1 presents our data analytics framework for energy-efficient supercomputing. We first collect data from the supercomputing center. The proposed analytics framework consists of (1) an explanatory model to establish causality, which empirically identifies the key factors that affect energy consumption; (2) a predictive model to validate model performance using in-sample and out-of-sample tests; and (3) a prescriptive model for policy evaluations, which help achieve energy-efficient data center operations. Based on a series of policy experiments on core allocation strategies, we produce calibration charts and policy guidelines to support the practical implementation of energy-efficient strategies in data centers.

The NSCC data center has 1,160 base compute nodes (CPU servers) and 128 accelerated nodes (GPU servers). Each node has 24 cores. These compute nodes are deployed over 20 standard equipment racks, consisting of 16 CPU racks, 3 GPU racks, and 1 mixed rack. A CPU rack chassis holds 72 identical CPU servers, and a GPU rack chassis holds 36 identical GPU servers. For our research purpose, we focus on the data on server loads and energy consumption. We thus used three big data sets collected from NSCC on power use in the server racks, their associated environmental sensor measurements, and the job schedules. This unique database is comprised of data logs over a one-month period from July 1 to July 31, 2018, at minute-level data granularity. It has a total of 892,800 observations in the one-month study period. Of these, 604,800 observations occur in the first three weeks and are used for the main model analyses. The remaining data set is used for out-of-sample validations and robustness checks of the proposed models.

Figure 1. Proposed Data Analytics Framework



The first data set *Power* captures the streamed power sensor data at one-minute intervals from the rack power distribution units (RPDUs). This data set consists of about 70 million records (13 sensor types × 6 RPDUs × 20 racks × 44,640 minutes). For the purpose of this study, we first extracted the power consumption rate recorded by each RPDU at each time unit. We then aggregated the power consumption rate (in watts) by the six RPDUs of a rack into a rack-level measure $TotalPower_{it}$, where $i = 1, \ldots, 20$ and $t = 1, \ldots, 44{,}640$.

The second data set *Facility* provides various facility sensor data such as heat load, facility pressure, facility water flow rate, exchanger water temperatures (inlet and outlet), and many other facility metrics, all measured at one-minute intervals. This data set consists of about 9.8 million records (11 sensor types × 20 racks × 44,640 minutes). The heat load is a comprehensive measure of the energy consumption rate in cooling the servers and accounts for nearly 40% of the total energy costs of the data center. Accordingly, we extracted the heat load (in watts) recorded at the rack level, leading to the measure $HeatLoad_{it}$, where $i = 1, \ldots, 20$ and $t = 1, \ldots, 44{,}640$. Because high server power consumption is associated with high heat load, which requires high amounts of energy to cool down the servers, we observed high correlation between the server power use data and the heat load data (the Pearson correlation coefficient is 0.91). We thus combined these two major sources of energy consumption to obtain an aggregate measure $TotalEnergy_{it} = TotalPower_{it} + HeatLoad_{it}$, which serves as the dependent variable in our model.[4]

The third data set *Jobs* logs the job arrivals, job starting and completion times, and core allocations among the execution hosts at one-minute intervals. The descriptive statistics of jobs are given in Table A5 in the online appendix. Jobs could be assigned to cores over multiple servers, either within or cross racks. These assignments are recorded as multiple records pertaining to the same job, where each record corresponds to the core assignment to the job in a unique server. In total, there were 252,016 such records for 162,799 jobs during the time period of this study. We obtained the total number of cores used in each unique server in every minute by aggregating the number of cores assigned to all the jobs running on the server concurrently. We then modeled the workload and load distribution factors in terms of the first four moments (as proxy of mean, variance, skewness, and kurtosis) of the load distribution over the set of servers in a rack. Let S denote the number of servers in a rack. Define *core distribution vector* $\boldsymbol{x}_{it} = \{x_{i1t}, \ldots, x_{iSt}\}$ as an S-dimensional vector of total number of cores assigned in each server of rack i at time t. The four moments of the core distribution vector can be calculated as $M1_{it} = \sum_{s=1}^{S} x_{ist}/S$, $M2_{it} = \sum_{s=1}^{S}(x_{ist} - M1_{it})^2/S$, $M3_{it} = \sum_{s=1}^{S}(x_{ist} - M1_{it})^3/S$, and $M4_{it} = \sum_{s=1}^{S}(x_{ist} - M1_{it})^4/S$. These moments measure the distribution of the number of cores used in the servers in a rack at any given time, which form the basis of our explanatory and prescriptive analyses.

Table A1 in Online Appendix B presents the descriptive statistics for the key variables based on minute-level data. The mean value for $M1$ is 18.38. Because the maximum number of cores per server is 24, this implies an average of 77% utilization of the servers at any time of the day. The mean values of $M3$ and $M4$ are -314.3 and 12,870.9, respectively, which suggest that the distribution of cores on average has relatively long left tails and is heavily tailed. This is consistent with our observation that the server utilization in the data center is mostly high, and many times the servers have been operating with a large number of running cores. Online Appendix B further provides a detailed description of the NSCC architecture, user service management, energy system configuration, and the database schema, data extraction, and aggregation.

4. Impact of Load and Distribution

In this analysis, we first examine the causal influences of the moments of the core distribution vector on total energy consumption. We present a dynamic panel data model to identify these effects and demonstrate the robustness of the estimation results. We then validate the model's predictive performance using in-sample and out-of-sample tests.

4.1. Model Specification

The unit of analysis in our model is a rack. The workload defined by the moments of the core distribution vector of each rack follows a time series measured at one-minute intervals over the one-month period of this study. Based on the first three weeks of the workload data, Table A2 in Online Appendix B shows the pairwise correlations among the four moments of the core distribution vector. We observed high correlation between $M1$ and $M2$, as well as $M2$ and $M4$. Furthermore, the variance inflation factors (VIFs) based on a linear regression of total energy consumed on the four moments indicate that the VIF for $M2$ is greater than five, whereas the VIFs of $M1$, $M3$ and $M4$ are less than five. Based on these considerations, we removed $M2$ from our model specification. Next, we observed intertemporal correlations between energy consumptions in adjacent periods; we thus include an autoregressive term with one lag into the model. Indexing rack by i and time by t and denoting the total energy consumed ($TotalEnergy_{it}$) as Y_{it}, we have the following model specification:

$$Y_{it} = \beta_0 + \beta_1 Y_{i,t-1} + \beta_2 M1_{it} + \beta_3 M3_{it} + \beta_4 M4_{it} + \beta_5 controls_{it} + u_{it}. \tag{1}$$

Prior studies find that power consumption of the physical server is approximately linear in server utilization

(Chen et al. 2008, Tang et al. 2008, Huang et al. 2011). We used the first moment of workload distribution to represent server utilization, and the third and fourth moments to represent the shape of distribution. We also incorporated several appropriate controls in the model. We used rack type dummies to distinguish between CPU, GPU, and mixed racks. In our panel data, we observed that more jobs are completed on specific days than others, and in particular, could differentiate weekdays from weekends. We thus included a weekend dummy for this purpose. We have also checked other variables related to job characteristics (e.g., total number of server instances involved in inter-server communication of a job) and environmental control variables (e.g., rack cooling distribution unit pressure, facility pressure, facility water flow rate, exchanger water temperature difference). These variables are either highly correlated with $M1$ or are statistically insignificant and do not contribute to model explanatory power. Hence, we did not include them in our final empirical model specification. The main reason, as we believe, is that server is the energy consumer in a rack. Therefore, server characteristics, including the well-documented workload-related measure $M1$ and our proposed distributional measures $M3$ and $M4$, are key determinants of energy consumption.

4.2. Model Estimation

Using the minute-level data as our base model analysis, Table 1 compares various model specifications with different controls at different levels of data aggregation.

Column (1) is the basic ordinary least squares (OLS) model that only uses the three moment variables and rack-type for estimation. Columns (2)–(7) present the basic OLS model plus autoregressive term with one lag, with different specifications on rack fixed effects and different levels of data aggregation from one minute, half hour, one hour, and two hours. We observe that all these different model specifications and data aggregation levels produce consistent estimation of results. The moments $M1$, $M3$, and $M4$ have positive coefficients, and the effects are all significant, which demonstrate that (1) as the average server workload in a rack increases, energy consumption in a rack increases; and (2) as the third and fourth moments of the workload distribution among servers in a rack increases, energy consumption in a rack increases.

The lagged dependent variable (the first-order autoregressive term) could lead to biased OLS model coefficient estimates in dynamic models. To establish the causal impact of $M1$, $M3$, and $M4$ on energy consumption, we need to address the endogeneity concern. Nickell (1981) shows that the least square dummy variable

Table 1. Model Estimation Results

	Basic OLS	OLS+AR(1)						LSDVC
		One minute		Half hour	One hour	Two hours		
Core allocation strategies	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Intercept	−3132***	−59.12***	311.7***	−400.9***	−843.5***	−1650***	2154***	
	(50.17)	(10.03)	(11.42)	(99.48)	(179.1)	(315.6)	(353.2)	
M1	1169***	22.36***	23.37***	86.41***	173.2***	341.6***	365.6***	359.4***
	(1.778)	(0.464)	(0.476)	(4.717)	(8.571)	(15.08)	(15.34)	(13.80)
M3	2.767***	0.057***	0.057***	0.275***	0.549***	0.920***	0.959***	0.945***
	(0.018)	(0.004)	(0.004)	(0.037)	(0.068)	(0.122)	(0.122)	(0.126)
M4	0.113***	0.002***	0.002***	0.013***	0.022***	0.034***	0.024 **	0.022 **
	(0.001)	(0.000)	(0.000)	(0.002)	(0.004)	(0.008)	(0.008)	(0.008)
AR(1)		0.981***	0.977***	0.931***	0.872***	0.759***	0.713***	0.716***
		(0.000)	(0.000)	(0.003)	(0.005)	(0.008)	(0.009)	(0.007)
CPU Rack	11317***	224.2***		796.8***	1394***	2403***		
	(32.86)	(7.094)		(69.58)	(123.9)	(214.5)		
GPU Rack	3179***	62.50***		225.2 **	404.2 **	723.4 **		
	(34.86)	(6.995)		(68.24)	(121.2)	(209.2)		
Weekend Dummy	636.2***	15.06***	19.37***	104.6***	176.4 **	299.4 **	331.4***	327.2 **
	(15.00)	(2.995)	(2.995)	(29.22)	(51.93)	(89.76)	(88.55)	(100.3)
Rack fixed effects	No	No	Yes	No	No	No	Yes	Yes
Observations	604,800	604,780	604,780	20,140	10,060	5,020	5,020	5,020
R^2	0.7173	0.9886	0.9887	0.9634	0.9418	0.9122	0.9150	—
Adjusted R^2	0.7173	0.9886	0.9887	0.9634	0.9418	0.9121	0.9146	—
Root MSE	5,245.0	1,045.2	1,044.2	1,860.1	2,334.1	2,845.5	2,804.3	2,797.3

Notes. Standard errors in parentheses. Bias correction is initialized by Anderson and Hsiao estimator and corrected up to order $O(N^{-1}T^{-2})$. Blank entries to indicate specific variables are not included.

$^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

(LSDV) estimator is inconsistent for finite T in autoregressive panel data models. The widely used instrumental variable (IV) and generalized method of moments (GMM) consistent estimators such as the Arellano-Bond estimator also suffer from a small sample bias due to weak instruments. Bruno (2005) shows that bias-corrected LSDV (denoted as LSDVC) estimators are preferred over the original LSDV, IV, and GMM methods. We ran the fixed effects LSDVC dynamic panel data regression to take into account unobserved individual heterogeneity and to obtain unbiased model estimates.

Column (8) of Table 1 presents the LSDVC model results. Compared with column (7), we see that the coefficient estimates for $M1$, $M3$, $M4$, and $AR(1)$ terms are very close to each other. This shows that the potential biases from the OLS estimation do not seem to be significant in our data. The main reason, as we believe, is that we have relatively large T (e.g., $T = 252$ in two-hour aggregation), which helps alleviate the concern of biased estimates. Furthermore, the OLS and LSDVC models yield similar root mean squared error (MSE; 2,804.3 and 2,797.3, respectively), which implies that the two models yield similar in-sample performance.

4.3. Model Performance

The various empirical model analyses in Table 1 demonstrate robustness of our model insights. The reliability of the proposed model in estimating energy consumption is further assessed using an in-sample and out-of-sample data analysis as follows. The first three weeks of data comprising of 30,240 minute-level observations for each rack constitutes the in-sample panels. The data from the fourth week comprising of the following 10,080 minute-level observations for each rack constitute the out-of-sample panels.[5] Using the model estimates from column (2) in Table 1, the model accuracies with the respective in-sample and out-of-sample data sets have been determined. For each rack $i = 1, \ldots, 20$, the estimation of total energy consumption is as follows.

$$\hat{Y}_{it} = \beta_0 + \beta_1 Y_{i,t-1} + \beta_2 M1_{it} + \beta_3 M3_{it} + \beta_4 M4_{it} + \beta_5 controls_{it} \tag{2}$$

In Equation (2), $Y_{i,t-1}$ is the observed energy consumption in period $(t-1)$, and the other variables are as defined before. It intuitively reveals two broad workload-related effects on energy consumption—The total energy consumed by a rack is the sum of two parts: the baseline energy consumption by all its idle cores and the additional energy consumption by its running cores. We denote these effects as the *baseline consumption effect* and the *running cores effect*, respectively. The baseline consumption effect is accounted in Equation (2) by the sum of the constant β_0, the autoregressive consumption component $\beta_1 Y_{i,t-1}$, and the rack- and time-fixed effects $\beta_5 controls_{it}$. The running cores effect is accounted by the sum of the total workload $\beta_2 M1_{it}$ and its distribution given by $(\beta_3 M3_{it} + \beta_4 M4_{it})$. The baseline consumption and the running cores effects together yield a complete model for the total energy consumption in a rack.

Under the two-hour, one-hour, half-hour, and one-minute data aggregation, we first performed in-sample and out-of-sample estimation of energy consumption for each rack. We then summed up the estimation over all 20 racks at each time point to obtain the total energy consumption at any given time. Consistent with Table 1, and also visually shown in Figures A2 and A3 in Online Appendix C, under these four levels of aggregation, the in-sample estimations achieve the root mean squared error (RMSE) of 2,845.5, 2,334.1, 1,860.1, and 1,045.2 and the mean absolute percentage error (MAPE) of 7.82%, 5.68%, 3.94%, and 1.77%, respectively. The out-of-sample estimations yield the RMSE of 2,499.7, 2,320.7, 1,959.6, and 1,083.5 and MAPE of 7.58%, 4.95%, 3.91%, and 1.96%, respectively.[6] This shows that the model accuracy significantly improves with decreasing levels of data aggregation.

Furthermore, different core allocation strategies would result in different core distribution vectors, and their moment measures at each point of observation are the energy-estimating variables in our model. Accordingly, we can envision a wide range of allocation strategies from complete packing to full load balancing of cores among the physical servers in a rack. Among the universal set of all core allocation strategies, complete packing and full load balancing are two bounding strategies that define two extreme core distributions. Figure A4 in Online Appendix D presents visualizations of the moments of the core distribution vectors with observed workload data in our data sets and under the two bounding strategies. These visualizations show that the sample data covers a wide range of vectors between the two bounding strategies and hence would be a good sample representation of core distribution vectors resulting from the universal set of allocation strategies. All together, the previous analyses empirically establish the internal and external validity of the estimation model and provide strong evidence that the estimation model is robust to different levels of data aggregation and can produce reliable prediction of energy consumption under a wide range of core allocation strategies.

5. Prescriptive Analytics for Energy Efficiency

In the following, we first discuss a class of representative core allocation strategies. We then develop overarching energy-optimization models that yield optimal core distribution vectors for racks to minimize total energy consumption in the data center. Subsequently, we develop a bicriteria optimization model to derive efficient solutions that capture tradeoffs between energy conservation and job performance optimization.

5.1. Core Allocation Strategies

As discussed in the previous section, the total enregy consumption of a rack consists of the baseline consumption effect and the running cores effect. For a given workload in a rack, the baseline consumption is a constant, and the running cores effect depends on the core allocation strategy. Specifically, energy consumption depends on two key factors: total workload measured by $M1$ and workload distribution among servers measured by $M3$ and $M4$. $M3$ and $M4$ are different measures of the imbalance in the workload distribution among the servers with respect to a standard even load distribution. As $M3$ becomes either highly positive or highly negative, workload tends to get concentrated in a few servers. When $M3$ is close to zero, workload is more evenly distributed. Similarly, when $M4$ becomes very high, workload is concentrated in a few servers, and when it gets close to zero, it is more evenly distributed. We introduce two baseline bounding core allocation strategies, packing and load balancing, to represent the least and most balanced workload distributions, respectively. The packing strategy aims to improve server utilization by minimizing the total number of servers used to execute a given set of jobs, leading to greater imbalance of workload distribution. In contrast, the load-balancing strategy aims to better distribute the workload by minimizing the difference between the allocated cores to each server. The range of values of $M3$ and $M4$ under packing is far greater than those under load balancing (see Figure A4 in the online appendix). Hence, the two strategies represent extreme levels of workload distributions.

We propose a simple threshold approach to switch between packing and load balancing to dynamically achieve energy-efficient resource allocations. Its rationale is as follows. If the total workload in a rack is higher than a threshold value, then most servers would be working at high core utilization levels; in this case, it will clearly be advantageous to pack jobs because the effect of total load would essentially dominate the effect of load distribution on total energy consumption, whereas at the same time extracting the full performance benefits of packing. However, if the total workload is lower than the threshold value, then the effect of load distribution dominates that of the total load on energy consumption, and load balancing could yield significant energy savings. Motivated by this, we develop a threshold strategy where packing is used when the total workload exceeds a threshold and load balancing otherwise. Online Appendix E provides empirical support for the threshold strategy from our supercomputing data. We next theoretically explore how the threshold—a cross-over point in the total workload that signals this switch—depends on the general data center configuration.

Consider a rack configuration consisting of S servers, each having a maximum of N cores. Let $K = SN$ denote the maximum core capacity for the rack. We denote the total load of a rack as $X \in [0, K]$. Let $M3_q(X)$ and $M4_q(X)$, $q \in \{P, L\}$, represent the respective moment parameters under packing (P) and load-balancing (L) strategies, respectively. We define $E_q(X) = \beta_3 M3_q(X) + \beta_4 M4_q(X)$, $q \in \{P, L\}$, where β_3 and β_4 are the positive coefficients estimated from Equation (1). The following proposition establishes the condition for the existence of the cross-over point and shows how it can be identified if it exists. If it does not exist, then load balancing will be superior to packing under all total load conditions. The proof of the proposition is given in Online Appendix F.

Proposition 1. *Define $X_1 = K/2$ and $X_2 = argminM3_P(X)$. If and only if $E_L(X_2) > E_P(X_2)$, there exists a cross-over point $\tau \in [X_1, X_2]$ that defines the threshold strategy, where τ can be determined by $\frac{M3_L(\tau) - M3_P(\tau)}{M4_L(\tau) - M4_P(\tau)} = -\frac{\beta_4}{\beta_3}$.*

Proposition 1 reveals several important insights that can be used to guide the data center administrators to identify the threshold. First, there is a critical evaluation point $X_2 = argminM3_P(X)$, which defines the least imbalanced load that can be achieved under the packing strategy. Thus, $E_P(X_2)$ is the lowest energy consumption that can be achieved under the packing strategy. If the energy consumption under the packing strategy at X_2 cannot outperform the load-balancing strategy, that is, $E_L(X_2) \leq E_P(X_2)$, then packing strategy can never outperform load-balancing strategy at any load level. Second, we show that the threshold τ should be bounded if it exists. The cross-over point should occur after the half load of a rack (X_1) but before the lowest energy consumption load under the packing strategy (X_2). Finally, assuming τ is continuous, the cross-over point can be identified by solving $E_L(\tau) = E_P(\tau)$, which leads to $\frac{M3_L(\tau) - M3_P(\tau)}{M4_L(\tau) - M4_P(\tau)} = -\frac{\beta_4}{\beta_3}$. However, because the load takes integer values in practice, the cross-over point is identified in the neighborhood of this solution.

Next, we propose an *energy-optimization* strategy that yields the distribution of workload among servers at any time that minimizes the expected total energy consumption of a rack. Using the optimal workload distribution, jobs can then be assigned to cores within the optimal use levels of the servers. The threshold strategy is easy to implement and incurs low computational load, whereas the energy-optimization strategy would achieve the highest energy savings. Both the threshold values under the threshold strategy and the optimal solutions under the energy-optimization strategy are derived from the calibrated empirical models developed in Section 4. Without loss of generality, we develop the optimization models for a homogeneous set of racks, either CPU or GPU type. Our proposed methods are general and can be applied to any rack type, including the mixed rack containing both types of servers.

5.2. Energy-Optimization Model

For notational simplicity, let $i = 1, \ldots, I$ denote the sequence of racks in the data center and $s = 1, \ldots, S$ denote the sth server in a rack. Define c_{it} as rack i's load at time t, which is the total number of cores required for all jobs allocated to rack i at time t. Let x_{ist} be an integer variable representing the number of cores allocated to rack i, server s, at time t, which should be no larger than the server's maximum core capacity N. Given any rack i, our objective is to determine the load distribution in that rack that minimizes its total expected energy consumption. For a given c_{it}, $M1_{it} = \frac{c_{it}}{S}$ is fixed, regardless of the way the cores are allocated to jobs over its physical servers. Furthermore, because the autoregressive component and the other controls have fixed values in Equation (2), it suffices to minimize $(\beta_3 M3_{it} + \beta_4 M4_{it})$ to obtain the energy-optimal load distribution at time t. The within-rack energy-optimization problem for rack i at time t under the CS policy is as follows:

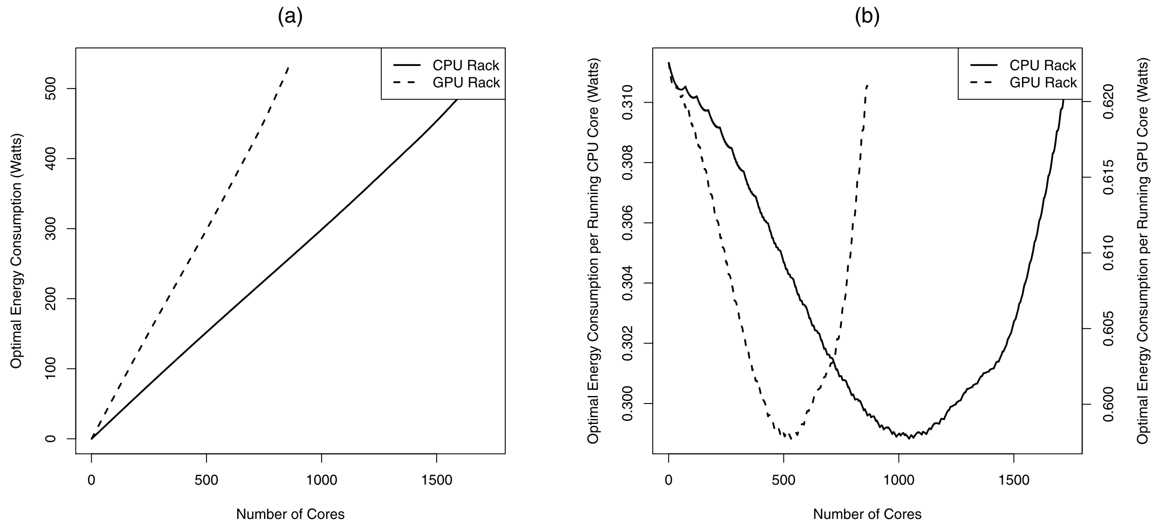
Model [WO].

$$\begin{aligned} \min_{x_{ist}} \quad & \beta_3 M3_{it} + \beta_4 M4_{it} \\ s.t. \quad & \sum_{s=1}^{S} x_{ist} = c_{it} \qquad (3) \\ 0 \le x_{ist} \le N,\ & \text{for } s = 1, \ldots, S \\ & x_{ist} \text{ integer.} \qquad (4) \end{aligned}$$

The coefficients β_3 and β_4 are estimated from Equation (2). In our empirical investigations, we have used the estimates using minute-level data aggregation given in column (2) of Table 1. Constraint (3) ensures that the total demand for cores at time t is exactly satisfied in the core allocations over the servers, and Constraint (4) enforces the core capacity limit of each server. Substituting $M3_{it}$ and $M4_{it}$ into the objective function yields $\sum_{s=1}^{S}\left[\frac{\beta_3}{S}\left(x_{ist} - \frac{c_{it}}{S}\right)^3 + \frac{\beta_4}{S}\left(x_{ist} - \frac{c_{it}}{S}\right)^4\right]$. This objective function is nonlinear and solving the optimization problem is strongly NP-hard. Therefore, following Croxton et al. (2003), we use a piecewise linear approximation of this nonlinear function and solve the resulting optimization problem. The problem reformulation and solution are presented in Online Appendix G.

In the NSCC context, $I = 16$, $S = 72$, and $n = 24$ for the CPU rack configuration, and $I = 3$, $S = 36$, and $n = 24$ for the GPU rack configuration. Let K denote the maximum core capacity for any rack type. Then we have $K = SN = 1{,}728$ possible values of the demand for cores in a CPU rack at any time t, and for a GPU rack, we have $K = 864$ values of cores demand; that is, $c_{it} \in \{1, \ldots, K\}$. The model [WO] can be presolved for each value of c_{it}, and the resulting energy-optimal solutions x^*_{ist}, $s = 1, \ldots, S$, can be saved in a $(K \times S)$ *core distribution matrix* $\mathbf{Z}$, where each row corresponds to a c_{it} value and represents its optimal core distribution vector. This vector provides optimal use bounds on the total number of cores to be allocated to jobs in each server under energy-optimal conditions. The core distribution matrix would instantaneously yield the energy-optimal allocation to an automated job scheduler or even serve as a ready-reckoner for a human administrator. Furthermore, alongside the core distribution matrix, the optimal energy consumption due to the running cores effect can be computed from the optimal solution of model [WO] as $W^*(c_{it}) = \beta_2 M1_{it} + \beta_3 M3_{it} + \beta_4 M4_{it} = \beta_2 \frac{c_{it}}{S} + \sum_{s=1}^{S}\left[\frac{\beta_3}{S}(x^*_{ist} - \frac{c_{it}}{S})^3 + \frac{\beta_4}{S}(x^*_{ist} - \frac{c_{it}}{S})^4\right]$ and can be stored in a $(K \times 1)$ *running cores energy consumption vector* $\mathbf{W}$. Figure 2(a) presents plots of $W^*(c_{it})$ over the entire range of c_{it} values for a CPU and a

Figure 2. Energy Consumption Calibrations



GPU rack, respectively. Similarly, Figure 2(b) plots $(W^*(c_{it})/c_{it})$ over this full range for each rack type. These plots serve as calibration charts for optimal energy consumption under every possible total load conditions for a rack.

Not surprisingly, Figure 2(a) shows that $W^*(c_{it})$ increases in the total number of running cores c_{it} on a rack. Furthermore, Figure 2(b) shows that the energy consumption per running core is a nonlinear "U-shaped" function of c_{it}, and this is attributed to the cubic and quartic terms of $(W^*(c_{it})/c_{it})$. We observe that the energy consumption per running CPU (GPU) core is the lowest when the total load of a CPU (GPU) rack is around 1,045 (532) cores. This also suggests that the most energy-efficient allocation of the total workload to a rack should be in the neighborhood of these values. This insight leads to the idea of cross-rack optimization where the workload is partitioned among the racks in a way to yield optimally energy-efficient workload allocation to each rack so that the total energy cost of all racks can be minimized.

Specifically, the cross-rack energy optimization uses a two-stage core allocation strategy. In the first stage, we solve a cross-rack energy-optimization problem [CO] using the already calibrated running cores energy consumption vector $\boldsymbol{W}$ to determine the optimal number of cores to be allocated in each rack. Therefore, different from model [WO], where c_{it} is a constant, it is a decision variable in model [CO]. In the second stage, given the optimal c_{it} from the first-stage model, we directly use the calibrated core distribution matrix $\boldsymbol{Z}$ to determine the optimal number of cores to be assigned to each server within each rack. We present the cross-rack energy-optimizatation model [CO] in Online Appendix H. Online Appendix I further discusses the computational performance of the [WO] and [CO] models.

5.3. Energy-Performance Tradeoffs

The distribution of cores across servers to a job will affect its performance. Because the cores in the same server share the internal RAM using a connecting bus, the communication overhead among cores residing within the same server is minimum. As a job is spread across multiple servers, the interserver core-to-core communication overhead may affect job execution and performance (Meng et al. 2015). Furthermore, each HPC application is different, and their computational and intercore communication loads are nontransparent to the data center managers. Moreover, each job is typically run only once, and there are no empirical data on how different core distributions on physical servers could impact a given job's completion time. Hence, we used a worst-case model of the communication overhead as a measure of performance in our analysis.

Denote J_{it} as a set of jobs running in rack i at time t. Assume that job $j \in J_{it}$ requires n_j cores. Denote z_{jst} as the number of cores allocated to server s for job j in period t. The total number of core-to-core communication pairs is $\binom{n_j}{2} = \frac{n_j!}{2!(n_j-2)!}$. The number of intraserver core-to-core communication pairs on server s is $\binom{z_{jst}}{2}$. Therefore, the total number of interserver core-to-core communication pairs associated with job j at time t is computed as $ISC_{jt} = \binom{n_j}{2} - \sum_{s=1}^{S}\binom{z_{jst}}{2}$. This model assumes every core communicates with every other core and is clearly the worst-case communication scenario.

We propose a bicriteria model that involves simultaneously minimizing the energy cost and the communication overhead described previously. Without loss of generality and for the sake of simplicity, we consider within-rack optimization. The energy cost is $f_1(\beta_3 M3_{it} + \beta_4 M4_{it})$, where the function f_1 follows the energy estimation Equation (2). Similarly, the performance cost is modeled as $f_2\left(\sum_{j\in J_{it}}\left[\binom{n_j}{2} - \sum_{s=1}^{S}\binom{z_{jst}}{2}\right]\right)$, where f_2 is a function of the communication overhead. Again, without loss of generality, because f_1 and f_2 are nondecreasing functions of their respective arguments, we use these functions in our empirical studies as simply $(\beta_3 M3_{it} + \beta_4 M4_{it})$ and $\left(\sum_{j\in J_{it}}\left[\binom{n_j}{2} - \sum_{s=1}^{S}\binom{z_{jst}}{2}\right]\right)$, respectively. Furthermore, because the two criteria are in different units of measurement, they are normalized first. The normalization procedure is detailed in Online Appendix J. Using a convex combination of the two criteria with a weight $\lambda \in [0,1]$ on the energy cost and $(1-\lambda)$ on the performance cost, we formulate the bicriteria energy-performance tradeoff optimization [EPT] model under the CS policy as follows:

Model [EPT].

$$\min_{z_{jst}\geq 0} \quad \lambda f_1(\beta_3 M3_{it} + \beta_4 M4_{it}) + (1-\lambda) f_2 \left(\sum_{j\in J_{it}}\left[\binom{n_j}{2} - \sum_{s=1}^{S}\binom{z_{jst}}{2}\right]\right) \tag{5}$$

$$\text{s.t.} \quad \sum_{s=1}^{S} z_{jst} = n_j, \text{ for } j \in J_{it}$$

$$\sum_{j\in J_{it}} z_{jst} \leq N, \text{ for } s = 1, \ldots, S \tag{6}$$

The first set of demand constraints allocate cores to jobs while satisfying each job requirement in set J_{it}. The second set of supply constraints ensure the total number of cores assigned to jobs in server s do not exceed the maximum number of cores in the server. Note that $c_{it} = \sum_{j\in J_{it}} n_j$. When $\lambda = 1$, the objective function of [EPT] purely focuses on energy consumption and this general model reduces to the (normalized) within-rack optimization model. Although model [WO] optimizes capacity allocation at aggregate server level, whereas model [EPT] optimizes the core assignments at individual job level, the two models give the same optimal solution when $\lambda = 1$. When $\lambda = 0$, the objective function of model [EPT] purely focuses on job performance, and the decision problem minimizes the total interserver core-communication costs of all jobs. When $\lambda \in (0, 1)$, the general model [EPT] takes into account the tradeoffs between energy consumption and job performance. Solutions to problem [EPT] with values of $\lambda \in [0, 1]$ yield the efficient frontier of solutions to the bicriteria problem.

6. Computational Experiments: Energy Optimization

We carried out extensive computational studies to evaluate the energy savings from our proposed methodologies. We evaluate four core allocation strategies: packing, load balancing, threshold, and energy optimization. We further consider two types of job allocation policies to be used within any of these strategies: CS and NCS. The implementation of these strategies and policies can be in either a single rack or across all the racks in the data center. These result in a total of $4 \times 2 \times 2 = 16$ combinations of experimental conditions that we have used to evaluate the proposed prescriptive models. Online Appendix K presents the implementation details of the core allocation strategies, and Online Appendix L presents the complete set of algorithms. All the optimization models were solved using Gurobi 9.0 (Gurobi Optimization LLC 2020).

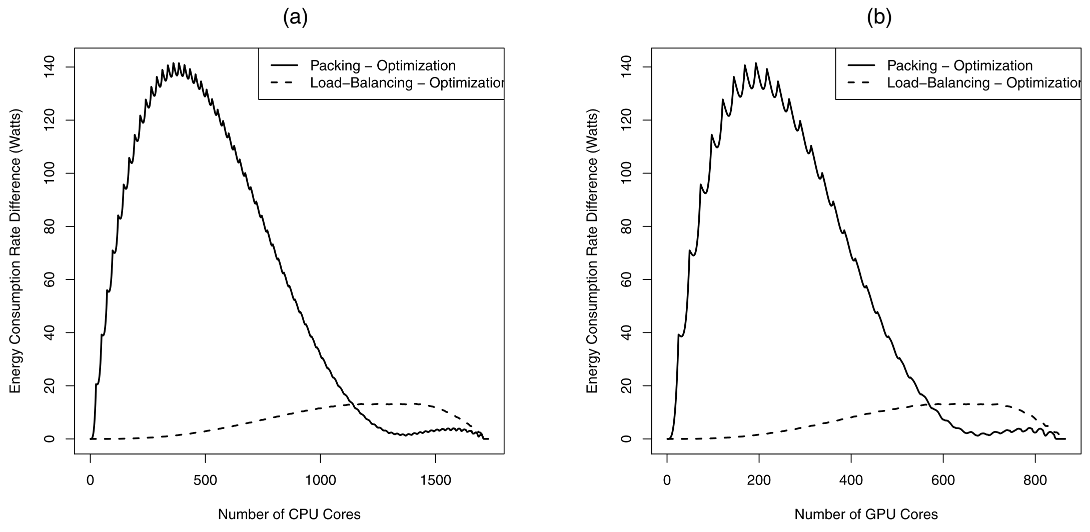
Using the empirical model for energy consumption in Equation (2), we first provided comprehensive calibrations of energy consumptions with packing and load balancing vis-à-vis the optimal energy consumption levels under the full range of workload conditions. These calibrations provided a basis for the threshold methodology that yields near-optimal energy consumption levels in the entire workload range. We then analyzed the performance of the energy-optimization model under both CS and NCS policies based on both simulated data sets with different job requirements distributions and the real data set provided by NSCC. We further empirically identified key factors affecting job completion times and demonstrated that the main insights hold in the presence or absence of the job completion time effects. We present these rsults in the following discussion.

6.1. Calibration of Energy Consumption

We have the maximum capacity of 1,728 and 864 cores in a CPU and GPU rack, respectively. Figure 3 shows the difference in the predicted energy consumption (using Equation (2)) between the packing/load-balancing strategies and the energy-optimization strategy (based on model [WO]) for each rack type and under each workload level.

We observe that, when the total workload of a rack (CPU or GPU) is relatively low, the energy-optimization strategy leads to significant energy savings compared with the packing strategy, whereas load balancing yields near-optimal energy consumption. This is because the distribution effect dominates the workload effect under low workload conditions. Packing leads to highly skewed distribution that brings significantly negative effect on energy consumption. When the total workload of a rack is comparatively high, the energy-optimization strategy achieves higher energy savings over the load-balancing strategy, whereas packing yields near-optimal energy

Figure 3. Total Energy Savings Between Energy-Optimization and Packing/Load-Balancing Strategies



consumption. This is because the workload effect dominates the distribution effect under high workload conditions. In such situations, energy optimization would lead to both high average server utilization and less skewed distribution of workload. The workload distribution under these two strategies becomes closer to what can be achieved under energy optimization. Therefore, the energy consumption rate differences among the three strategies reduce.

We further observe from Figure 3 that the calibration curves for load balancing and packing intersect when the total load is 1,141 (569) cores for CPU (GPU) racks at 66.0% (65.9%) core utilization. We denote these cross-over points as τ_{CPU} and τ_{GPU}, respectively.[7] These cross-over points define the threshold strategy: When the total load of a rack is less than the threshold, load balancing should be followed; otherwise, packing should be adopted. Online Appendix M provides more details for identifying the cross-over points within and across racks.

To empirically validate the threshold strategy and insights revealed by the calibration chart in Figure 3, we randomly choose a rack and construct subsamples with low load and high load. For each observed load, we obtain the theoretical core allocation vectors under pure packing and pure load balancing. We then extract the actual (observed) core allocation vector and calculate the pairwise Euclidean distances between packing (P), load balancing (L), and observed (O) strategies, denoted as D_{PL}, D_{PO}, and D_{LO}, respectively. We focus on observations with $D_{PL} > \eta$ to ensure packing and load balancing strategies can be sufficiently distinguished. We then classify an observed core allocation strategy as being proximal to packing if $\frac{D_{LO}-D_{PO}}{D_{PL}} > \epsilon$; that is, when the observed core allocation is sufficiently closer to packing than load balancing relative to its maximum distance to the two strategies (i.e., the relative difference is positive and larger than a threshold). Similarly, we classify the observed core allocation strategy as being proximal to load balancing if $\frac{D_{PO}-D_{LO}}{D_{PL}} > \epsilon$. Otherwise, we deem the allocation strategy to be a mixed strategy. Online Appendix E provides the detailed procedure for sampling, classification, and empirical model estimation. Table A7 provides strong evidence that load balancing outperforms packing when the load is low, and packing outperforms load balancing when the load is high, justifying the theoretical insights revealed by the threshold strategy.

The threshold strategy is intuitively appealing because, when the total resource demand of a rack is relatively low, the number of jobs is significantly smaller than the number of servers, and packing would yield extremely unbalanced workload distribution. Accordingly, energy savings outweigh the performance consideration, and the load-balancing strategy is used. On the contrary, when the total resource demand of a rack is high, almost all servers are occupied with high utilization. Packing strategy well aligns performance with energy saving considerations, and hence, the packing strategy is adopted. Clearly, the threshold strategy combines the best of the packing and load-balancing strategies by appropriately trading off job performance against energy consumption.

6.2. Simulation Experiments

The workloads in data centers exhibit dynamic patterns. Although our HPC data center operates at an average of 77% utilization in our study period, past studies have shown that utilization rates in many data centers are quite low, resulting in poor use of resources (Pawlish et al. 2012). To assess the energy efficiency of the core allocation methodologies over a period of time as workloads vary, we carried out a set of simulation experiments as follows. We simulated rack operations for a three-week period comprising of 30,240 minutes of service. The core loading for each minute was randomly generated from an underlying Uniform, Normal, or Beta distribution. With the Uniform distribution, random loadings have been generated from U[0, 1728] and U[0, 864] for CPU and GPU racks, respectively. Under the Normal distribution, we experimented with mean values of 360, 864, and 1,269 for CPU racks and 192, 432, and 624 for GPU racks to reflect light (22% utilization), normal (50% utilization), and heavy (78% utilization) load conditions, respectively. We set the standard deviation at high or low levels as 500, 200, 250, and 100 for CPU and GPU racks, respectively. For Beta distribution, we first simulated data by choosing B(0.5,2), B(2,2), and B(2,0.5) to represent 20%, 50%, and 80% utilization load conditions, respectively. We further fit the Beta distribution with our real sample data from different racks over the three-week period and chose three representative workload patterns with B(1.75, 1.96), B(0.43, 0.22), and B(1.87, 0.22) at average utilization of 47%, 66%, and 90%, respectively.

In every minute of the data center operation simulated, we computed the total energy costs under each core allocation strategy for a rack: packing (P), load balancing (L), threshold (T), and energy optimization (O). We aggregated the energy costs under each strategy over the three-week period, denoted as $\hat{TE}_q$, $q \in \{P, L, T, O\}$. As in the calibration studies, we determined the percentage of energy savings of the optimization strategy over the other given strategies, given as $\Delta_q = (\hat{TE}_q - \hat{TE}_O)/\hat{TE}_O$, $q \in \{P, L, T\}$. The mean percentage savings over the set of simulations under Uniform, Normal, and Beta distributions are summarized in Tables 2 and 3, respectively.

We observe that, under all distributions, the expected energy savings of the energy-optimization strategy over the packing strategy are significantly larger than those

Table 2. Expected Energy Savings of Energy-Optimization Model Under Uniform/Normal Distributions (High and Low Variances)

	CPU rack				GPU rack			
	Uniform	Normal			Uniform	Normal		
Distribution	$U[0,1728]$	$N(360,\frac{500}{200})$	$N(864,\frac{500}{200})$	$N(1269,\frac{500}{200})$	$U[0,864]$	$N(192,\frac{250}{100})$	$N(432,\frac{250}{100})$	$N(624,\frac{250}{100})$
Core utilization	50.0%	20.8%	50.0%	73.4%	50.0%	22.2%	50.0%	72.2%
Packing	13.12%	25.83%	14.17%	6.72%	20.70%	47.52%	22.16%	10.23%
		40.76%	14.03%	2.02%		87.69%	20.30%	3.17%
Load balancing	1.60%	1.34%	1.80%	1.88%	2.54%	2.57%	2.87%	2.77%
		0.69%	2.09%	2.20%		1.69%	3.34%	3.16%
Threshold	1.01%	1.17%	1.34%	1.14%	1.67%	2.25%	2.13%	1.71%
		0.69%	1.96%	1.11%		1.69%	3.14%	1.66%

with the load-balancing strategy, which in turn, are larger than those with the threshold strategy. This clearly shows that the threshold strategy performs very well even under dynamically varying load conditions. In the presence of workload dynamics, the packing strategy performs better when the load is heavy (70%–90% utilization for 1.26%–10.23% energy savings), and the load-balancing strategy performs better when the load is light (around 20% utilization for 0.69%–2.57% energy savings). In contrast, the threshold strategy yields near-optimal energy consumption performance under all dynamic workload conditions. Only 0.33%–3.14% energy savings can be gained via within-rack optimization over the threshold strategy. In addition, within-rack optimization can achieve higher energy savings for GPU racks than CPU racks under all three types of distributions. The insights gained under dynamic workload distributions are consistent with the theoretical energy calibrations presented in Figure 3.

6.3. Experiments with NSCC Data Sets

In this section, we present our experiments with NSCC data sets. First, assuming core allocations will not affect the job completion times, we evaluated the model performance using the full set of 16 experimental settings in Section 6.3.1. However, there is a concern that different core allocations may affect a job's completion time due to the changes in its interserver core-to-core communication requirements. This concern cannot be fully addressed unless we rerun all the jobs again under each core allocation strategy. This is not feasible in a data center environment. Furthermore, the communication requirements of an HPC application depend entirely on its developers, and the data center administrators have neither the knowledge nor the control over them. Therefore, we first developed a model to fit the observational job stream data to analyze how the job-related characteristics could affect the job completion times, based on which we then estimated the energy consumption under different core allocation strategies in Section 6.3.2. We present these two sets of experiments in the following discussion.

6.3.1. Core Allocations Not Affecting Job Completion Times.

We first extracted the total energy consumed by each rack and aggregated all racks to obtain the total energy consumption in the data center over the three-week period, denoted as $TE = 282,042.7$ kWh. This serves as a baseline comparison. Based on the real job streams observed in the data center over the three weeks from the NSCC data set *Jobs*, we allocated the cores to jobs at each scheduling time. We estimated the energy consumption under each of the 16 methodologies in each minute using Equation (2). We

Table 3. Expected Energy Savings of Energy-Optimization Model Under Beta Distributions: Simulation and Parameter Estimates from Sample Data

	CPU rack			GPU rack		
Simulated distributions	$B(0.5,2)$	$B(2,2)$	$B(2,0.5)$	$B(0.5,2)$	$B(2,2)$	$B(2,0.5)$
Core utilization	20.0%	50.0%	80.0%	20.0%	50.0%	80.0%
Packing	25.26%	14.29%	2.91%	58.22%	22.61%	3.93%
Load balancing	0.93%	1.85%	1.23%	2.23%	2.95%	1.69%
Threshold	0.79%	1.36%	0.63%	1.89%	2.17%	0.83%
Data-estimated distributions	$B(1.75,1.96)$	$B(0.43,0.22)$	$B(1.87,0.22)$	$B(1.75,1.96)$	$B(0.43,0.22)$	$B(1.87,0.22)$
Core utilization	47.0%	66.0%	90.0%	47.0%	66.0%	90.0%
Packing	16.04%	5.55%	1.26%	25.34%	7.71%	1.63%
Load balancing	1.78%	0.83%	0.68%	2.90%	1.15%	0.89%
Threshold	1.33%	0.47%	0.33%	2.17%	0.64%	0.40%

then aggregated the estimated energy consumption over the three-week period to get the total estimated energy consumption under each methodology, denoted as $\hat{TE}$ in general, for the sake of brevity. The relative energy saving percentage is computed as $\Delta = (TE - \hat{TE})/TE$ for each methodology. A positive value of Δ implies energy savings, and larger its value, the higher the total energy savings. Table 4 presents the energy saving percentages obtained from this study.

There are several interesting findings and policy implications. First, not surprisingly, other things being equal, the energy savings under the NCS policy are lower than those under the CS policy. This is because the CS policy implementation assumes full control of resources, whereas the NCS policy limits this resource adjustment flexibility and cores can only be released when jobs complete.

Second, we observe that the packing strategy performs the worst. The marginal costs/savings under packing is almost negligible in all cases. This intrinsically reveals that the NSCC data center's automatic Altair PBS Pro scheduler, which is the industry-leading HPC workload manager and job scheduler, mostly focuses on job performance rather than energy optimization. The small negative savings reveal the slight improvements over the pure packing strategy that were achieved by the data center managers' manual interventions in run-time job scheduling. In our interviews with the data center administrators, they acknowledged that sometimes human interventions that override automatic scheduler allocations have been carried out to avoid overloading some servers. This is corroborated by our experimental findings that showed less than 1% improvement in energy consumption. These results also reveal that such interventions are infrequent and have been carried out only when deemed necessary.

Third, in contrast, the load-balancing strategy and the threshold strategy could achieve 1.86% and 2.35% energy savings, respectively. The threshold strategy marginally outperforms load-balancing strategy in most cases. As expected, the threshold strategy has an advantage over the pure load-balancing strategy because it can switch between packing and load balancing to help reduce energy consumption when warranted. However, in the case of cross-rack allocation under the NCS policy, the threshold strategy does not outperform the load-balancing strategy in energy consumption. The main reason is because the NCS policy requires fixed core allocation for a job from start to finish. More opportunities of core redistribution across racks are restricted at the instant the switch is warranted.

Fourth, the energy-optimization models perform the best in energy savings under all scenarios. In addition, because cross-rack optimization provides more flexibility for core redistribution, it can provide higher energy savings of 3.81% (CS) and 2.74% (NCS) compared with only within-rack optimization (2.95% and 2.45%, respectively). The level of energy saving is justified by our theoretical calibration. Because the mean number of cores allocated per server is 18.38 and on average the servers have been running at a high utilization level of about 77% during our study period, Figure 3 shows that the packing strategy mostly employed by the commercial PBS Pro scheduler would yield near-optimal energy consumptions at such high utilization levels. This explains why we do not observe very high improvements in energy consumption by optimization over that actually consumed during the period of this study.

Assuming that all the servers are run at the average utilization of 77% throughout the year, the estimated 3.81% energy saving from optimization would result in a minimum annual savings of 186,772.7 kWh of energy consumption. However, server utilization levels are seasonal and fluctuate continuously between low and high throughout the year. As we show in Table 3, the estimated percentage of energy savings would be much higher under lower levels of core utilization. Hence, with varying levels of core utilization in the data center throughout the year, we expect annual savings to accrue at significantly higher levels of magnitude through energy optimization.

6.3.2. Core Allocations Affecting Job Completion Times.

We first conduct an explanatory analysis to understand the key factors that influence the job completion times, based on which we estimate the new job completion times resulting from different core allocation strategies. Consequently, the total workload of a rack at time t could be affected, which subsequently could affect the estimated energy consumption of a rack. We

Table 4. Estimated Energy Savings Under Different Methodologies

	Computational steering (CS)		Noncomputational steering (NCS)	
Core allocation strategies	Within-rack	Cross-rack	Within-rack	Cross-rack
Packing	−0.56%	0.82%	−0.03%	−0.91%
Load balancing	1.86%	1.85%	1.85%	1.15%
Threshold	2.35%	1.89%	2.02%	0.99%
Energy optimization	2.95%	3.81%	2.45%	2.74%

demonstrate that our main insights are robust even if different core allocation strategies would cause variations of the job completion times.

6.3.2.1. Explanatory Analysis of Job Completion Times. We estimated a regression model that specifies the impact of core scheduling and job characteristics on job completion times using the available sample data. The estimation is as follows. We extracted job-related characteristics and rack-level and job-level communication overhead measures from 162,799 jobs that we collected in our NSCC sample. The first set of variables is related to job-specific requirements, including the total number of cores required (*NumCore*), the duration of the job, the resource type, and job arrival time. Because NSCC has used different queues to manage jobs with different length and core requirements, in the absence of queue information and to account for unobserved factors influencing job allocation, we used the four quartiles of the job duration data as cutoff points (i.e., 16, 81, and 188 minutes) to classify all jobs into four groups indicated by group dummies (*Group*1–*Group*4). In terms of resource type and job arrival time, we included dummies for whether the job demands CPU (*TaskCPU*), GPU (*TaskGPU*), or large memory (*TaskLMN*) resources and whether the job arrives during weekends (*Weekend*). The second set of variables is related to communication overhead. If the cores required for a job are allocated across multiple racks, it may incur some cross-rack communication overhead. We thus measured the number of racks a job is allocated to (*NumRack*). In addition, the number of concurrent running jobs and the total workload at rack level may affect the communication of cores within servers in a rack because all jobs share the same network switches within the rack. For each job over its duration on a specific rack, we measured the average number of concurrent running jobs in the rack per minute (*AvgNumJob*) and the average number of running cores per allotted server in the rack per minute (*AvgLoad*), respectively. Furthermore, the job-level communication overhead is measured by the logarithm of interserver core-to-core communication pair (*ISC*) defined in Section 5.3.

Using the logarithm of job completion time in minutes (*JobTime*) as the dependent variable, we estimate different OLS model specifications in columns (1)–(3) by incorporating the various sets of variables. Column (4) is our full model with the interaction term. The estimated results are shown in Table 5.

Based on the adjusted R^2 values, column (4) has the best goodness-of-fit performance among all model specifications. The results indicate that the effect of the number of cores required by a job on its completion time is significant, but its effect is very small. Compared with jobs in the first group as a baseline, jobs belonging to other groups require longer running time with a monotonic increasing trend. Jobs demanding GPU or large memory resources run a relatively longer time than jobs that only need CPU cores. Jobs arriving over the weekend and allocated to multiple racks require longer execution time. In addition, the estimated coefficients for *ISC* and the interaction terms indicate that job groups moderate the heterogeneous effects of *ISC* on job completion times. The interserver core-to-core communication has a small negative effect on job completion times for jobs in the first three groups but a large positive effect in *Group*4. Consistent with our empirical data analysis, it is reasonable to expect that jobs that run for longer durations would be affected more by

Table 5. Explanatory Analysis of Job Completion Times

	(1)	(2)	(3)	(4)
Intercept	1.6680(0.0032)***	1.6339(0.0087)***	1.6404(0.0088)***	1.6988 (0.0087)***
NumCore	0.0001(0.0000)***	0.0001(0.0000)**	−0.0000(0.0000)	0.0002(0.0000)***
Group2	2.0618(0.0042)***	2.0634(0.0043)***	2.0623(0.0043)***	2.0405(0.0043)***
Group3	3.0969(0.0042)***	3.1000(0.0044)***	3.0987(0.0044)***	3.0649(0.0044)***
Group4	4.3603(0.0042)***	4.3620(0.0043)***	4.3595(0.0043)***	4.2821(0.0044)***
TaskGPU	0.0794(0.0081)***	0.0654(0.0084)***	0.0622(0.0084)***	0.0800(0.0083)***
TaskLMN	0.2884(0.0239)***	0.2858(0.0243)***	0.2863(0.0243)***	0.3082(0.0239)***
Weekend	0.1148(0.0033)***	0.1141(0.0033)***	0.1154(0.0033)***	0.1075(0.0033)***
NumRack		0.0513(0.0065)***	0.0437(0.0065)***	0.0160(0.0065)*
AvgNumJob		−0.0001(0.0000)**	−0.0000(0.0000)	0.0000(0.0000)*
AvgLoad		−0.0006(0.0003)*	−0.0008(0.0003)**	−0.0013(0.0003)***
ISC			0.0084(0.0011)***	−0.0521(0.0016)***
ISC × *Group2*				0.0233(0.0023)***
ISC × *Group3*				0.0447(0.0026)***
ISC × *Group4*				0.1174(0.0018)***
Observations	162,799	162,799	162,799	162,799
R^2	0.8767	0.8767	0.8768	0.8801
Adjusted R^2	0.8767	0.8767	0.8767	0.8801

Notes. Standard errors in parentheses. Blank entries to indicate specific variables are not included.
*$p < 0.05$; **$p < 0.01$; ***$p < 0.001$.

the communication overhead, especially the interserver core-to-core communication measured by *ISC* in our model.

6.3.2.2. Energy Savings Estimation. When considering the effect of core allocation on job completion times, we can only focus on NCS policy implementation because core allocation to a job is not fixed over time under the CS policy. To avoid the confounding effect brought by cross-rack core allocation, we focus on within-rack implementation to estimate the energy savings.

Based on the same job streaming data from July 1 to 21, 2018, we implemented the four core allocation strategies with varying job completion times. If a job's *ISC* resulting from a core allocation strategy changes, then the estimated job completion time also changes. Therefore, the departure epoch of each job under each core scheduling strategy could be affected, which subsequently affect core availability and core allocation for newly arrived jobs. Hence, given the arrival stream of jobs in the data set, we estimated their completion times under each core allocation strategy by taking into account each job's *ISC* using the model specification in column (4) of Table 5. To illustrate in this context, when the threshold strategy is used to schedule arriving jobs, about 0.1% of all jobs remain at the same job completion times as with the observed values, while 44.3% of the jobs have increased durations by 87.0 minutes on average, and 55.6% of the jobs have decreased durations 108.5 minutes on average. Putting all jobs together, the overall core distributions over the servers at each time interval was determined. Using this, the total energy consumption under each core scheduling strategy was finally determined using our energy estimation Equation (2). We then compute the percentage of energy savings of the optimization strategy over packing, load-balancing, and threshold strategies as in Section 6.2. Table 6 presents a comparison of the estimated energy savings when core allocation affects the job completion times and when it does not.

Clearly, the optimization strategy performs the best. Table 6 shows that threshold strategy performs the next, followed by load balancing and packing. This performance ordering of the core allocation strategies is consistent in the presence or absence of the job completion time effects. However, we should caution the readers that what matters is the relative ordering of the strategies. The estimated energy savings when considering the effect of core allocation on job completion times is subject to the empirical estimation of this effect, which is data dependent. Online Appendix N provides additional experiments based on dynamic workloads simulated from distributions fitted by the NSCC data. The estimated energy savings under different core allocation strategies show robust insights as in our main simulation experiments.

Table 6. Estimated Energy Savings of the Energy-Optimization Strategy

Core allocation strategies	When core allocation does not affect job completion times	When core allocation affects job completion times
Packing	2.54%	4.16%
Load balancing	0.61%	1.18%
Threshold	0.44%	0.75%

7. Computational Experiments: Energy-Performance Tradeoffs

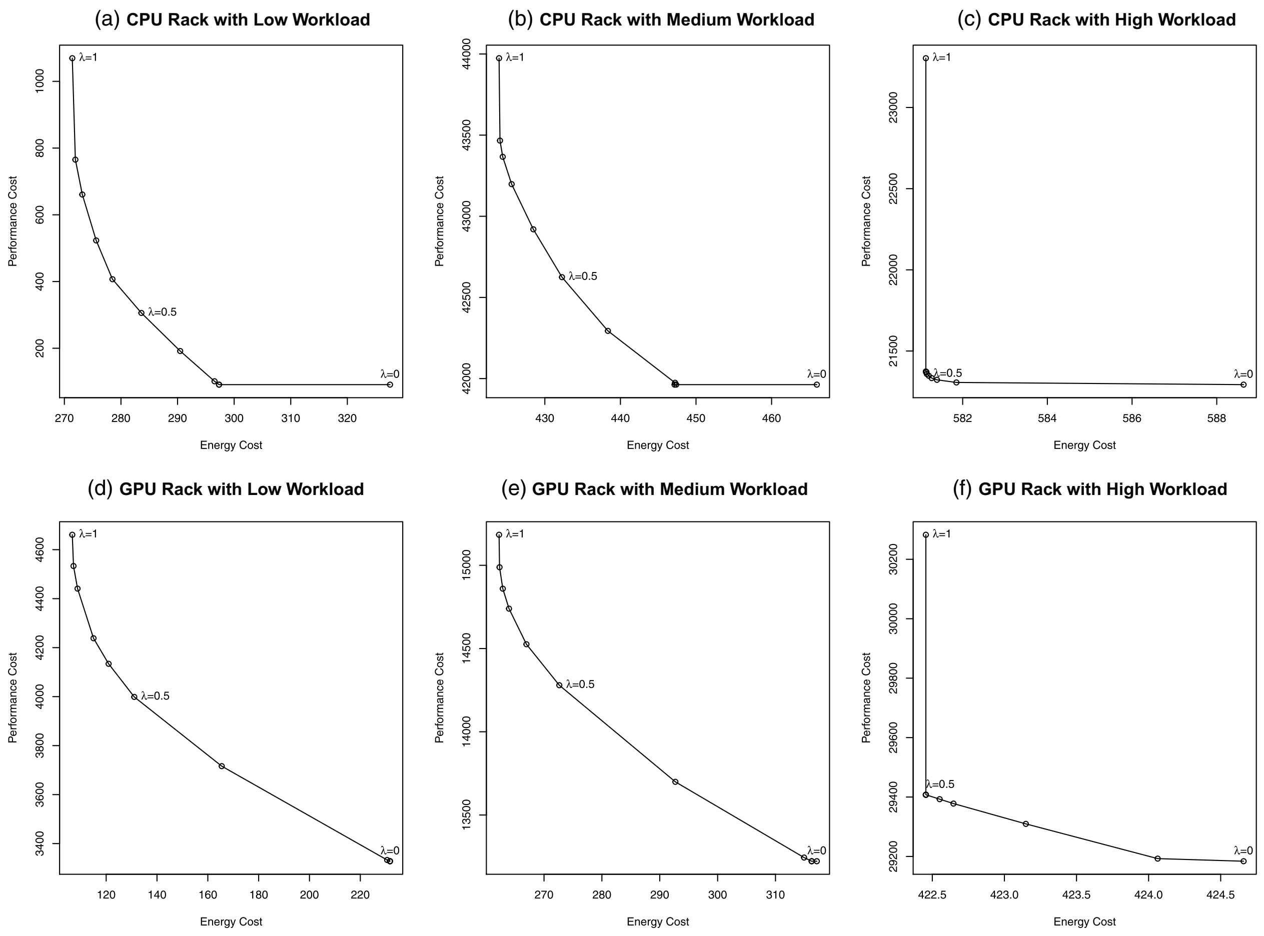
In this section, we extend our evaluation to simultaneously consider both energy consumption and job performance. We first demonstrate the general energy-performance tradeoffs under different total workloads. We then perform a bicriteria evaluation of the core allocation strategies. Finally, we develop a decision tree guideline for strategic and operational choices in data center resource management for green computing while considering job performance.

7.1. Tradeoffs Under Different Workloads

The efficient frontier is a set of Pareto optimal solutions to problem [EPT] as the decision weight of energy cost λ varies. For this study, we extracted sample data from the NSCC data set by choosing all observations with 346, 864, and 1,382 CPU cores and 168, 432, and 696 GPU cores, representing low (19%–20% utilization), medium (50% utilization), and high (80%–81% utilization) load conditions, respectively. This resulted in a total of 141 observations. For each observation, based on the raw data of workload and associated job requirements, we solved model [EPT] by varying the weight $\lambda \in [0, 1]$ with a step size of 0.1. The optimal solution to each instance of model [EPT] solved yielded its corresponding normalized energy cost and performance cost measures. For a given workload c_{it} and λ value, the energy and performance costs for each instance solved in that category would be different because their job requirements, and hence their core allocations would be different. Therefore, we averaged the energy costs and performance costs under each workload and λ value category. Figure 4 presents the efficient frontier of energy-performance tradeoffs under low, medium, and high workloads for both CPU and GPU racks. As λ decreases, the energy cost increases and the performance cost decreases, demonstrating the tradeoffs. The data center administrator can choose λ according to the desired level of tradeoff between the two objectives in allocating cores to jobs under different workload conditions.

The total workload, core allocation strategy and the tradeoff between energy and performance costs constitute a trifecta of considerations in allocating cores to

Figure 4. Energy-Performance Efficient Frontier Under Low, Medium, and High Workloads



jobs. However, the choice of an appropriate λ requires an exploration of the preference structure of the data center administrator. This could impose significant cognitive load and may not even be viable. In practice, data center administrators prefer to adopt easy-to-implement, simple, and effective strategies for implementation. We next evaluate the performance impacts of the energy-optimization model and the packing and load-balancing strategies.

First, we solved model [EPT] using $\lambda = 1$. This is the pure *energy-optimization* strategy that yields the minimum total energy cost of a rack. Next, we solved model [EPT] using $\lambda = 0$. This is the pure *performance-optimization* strategy that yields the minimum total performance cost of a rack. Figure A11 in Online Appendix O compares the two strategies under energy and performance costs. Considering the energy-performance tradeoff, it is preferrable to use the energy-optimization strategy in low workload conditions and the performance-optimization strategy in high workload conditions.

The energy costs under packing and load-balancing strategies demonstrate the same pattern as in Figure 3. Figure A12 in Online Appendix O compares the performance costs of the two strategies. We observe that packing always outperforms load balancing under all workload conditions. Therefore, when the total workload in a given rack is higher than the threshold, packing is preferred over load balancing because there is a natural alignment of both energy and performance optimization under packing. However, when the total workload in a given rack is lower than the threshold, load balancing performs better in energy conservation but packing performs better in performance, so there is a clear tradeoff between the two strategies. Overall, these experiments provide additional support to our main insights regarding the choices of different core allocation strategies in data center operations.

7.2. Implementation Guidelines

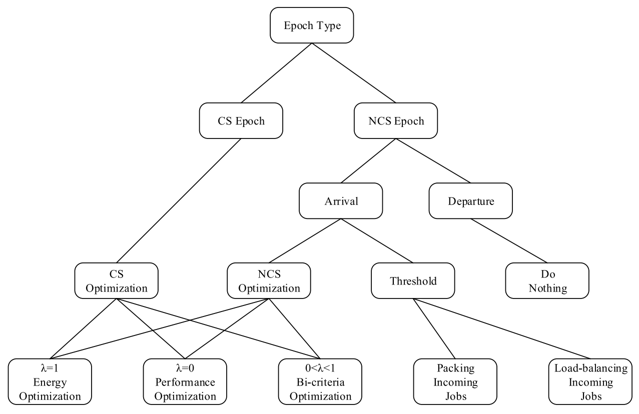
Although our experimental results show that the CS policy achieved higher energy savings than the NCS policy, data center managers need to be cognizant of its potential negative impact on job interruption. To this end, we propose an integrated approach to implementing the CS and NCS policies as follows. The arrival or completion of a job signals a change to the current core distribution. This can be operationally accomplished using the virtually noninterruptive NCS policy. We term the events triggering this action as *NCS Epochs*. Conversely, independent of job arrivals or departures, CS can be performed at specific times to redistribute cores for all running jobs with a view to achieve the required balance between energy and performance costs. The CS interventions can be performed periodically at either fixed intervals or flexible intervals of time. We term the times that signal such CS interventions as *CS Epochs*. Hence, there will be a series of NCS epochs between successive CS epochs over time. Figure 5 presents guidelines structured as a decision tree for implementing these strategic and operational choices.

Under the CS epoch, CS optimization can focus on energy costs ($\lambda = 1$), performance costs ($\lambda = 0$), or some combination ($0 < \lambda < 1$). In the latter case, the system administrator needs to determine the preferred weight λ by examining the energy-performance efficient frontier and evaluating the different options it presents. Under the NCS epoch, there are two cases depending on the type of events. If it is a job departure event, then no action is needed. If it is a job arrival event, then the system administrator needs to decide whether to perform NCS optimization or use a simple threshold-based heuristic approach to schedule incoming job(s).

8. Discussion and Conclusion

HPC that drives large-scale research and operation-intensive applications has developed into a substantially large sector in the data center industry. This research is motivated by the increasing need for energy-efficient operations in HPC data centers worldwide and is developed from and validated by NSCC's large-scale databases on job requirements, computing resource management, and energy consumption. We first present an explanatory model to identify key factors that affect an HPC data center's energy consumption and then demonstrate our model's high predictive performance. We further

Figure 5. Implementation Guidelines



develop prescriptive models for energy-efficient and performance-efficient operations.

The major contributions of this research are fourfold. First, using a dynamic panel data analysis we establish the causal effects of the level of workload and its distribution over the servers of a rack on the rack's total energy consumption. We extend the existing literature by showing that not only server utilization, but also the variability in workload distribution matter in energy consumption. Understanding the effects of workload distribution on energy consumption has enabled the development of new strategies to increase the data center's overall energy efficiency.

Second, we propose prescriptive models for energy-efficient workload management. In our prescriptive analytics framework, we focus on four core allocation strategies (packing, load balancing, threshold-based switching, energy optimization), two implementation policies (with and without computational steering), and job scheduling at two levels (within-rack and cross-rack). Based on our empirical estimation of the energy consumption under different strategies, we calibrate the energy savings under all total workload conditions. The calibrations can be either used as a ready-reckoner by a human data center administrator or be embedded within an automated scheduling software to make optimal decisions on workload allocation.

Third, we find evidence that the packing strategy is preferred when the total workload on a rack is higher than a threshold and a load-balancing strategy is preferred when it is lower. Our experimental studies reveal that a strategy of switching between packing and load balancing according to this threshold on total workload would yield near-optimal energy consumption under all total workload conditions. These findings are robust when jobs' completion times can be affected by different core allocation methodologies.

Fourth, we consider the trifecta of factors—the total workload, core allocation strategy, and the tradeoff between energy and performance costs in allocating cores to jobs. We develop a bicriteria optimization model to evaluate the potential tradeoffs between energy consumption and job performance. An analysis of this model has led to the development of implementation guidelines framed as a decision tree for strategic and operational choices in data center resource management for green computing while concomitantly considering job performance. The insights and results from this research significantly contribute to energy-efficient data center operations and would guide the design and development of commercial scheduling systems for HPC data centers.

The stability and reliability of the empirical models are shown through extensive in-sample and out-of-sample testing. Given that the system under study is an engineering system, its behavior under similar conditions are predictably repeatable. In this regard, we contrast this engineering system from other social science applications involving human behaviors that are both difficult to predict and repeat. The online product recommendation systems is a classic example. In such social applications, knowledge on human behavior is continuously revealed through data collected over time, and it is important to correspondingly update the parameters of a model fitting the data as human behaviors change and evolve. Maintaining this currency is important in applying the model to runtime instances such as making correct recommendations to a user based on his past actions and preferences. It also requires maintaining a balance between costs of knowledge obsoleteness and model recalibrations over time (Fang et al. 2013). Unlike these systems, the engineering systems such as data centers yield stable, reliable, and repeatable model parameters over time.

Some of the important directions for future research are as follows. First, we primarily focus on green computing and energy-efficient data center operation in this research. In the future, based on the predicted demand patterns (e.g., relatively stable or fluctuating), the scheduling algorithm may switch between energy-optimal mode and performance-optimal mode, where the energy-mode aims to achieve energy efficiency, whereas the performance-mode prioritizes applications based on their execution requirements. Second, there is an emerging trend on leveraging human-AI intelligence to improve organizational decision making. Future work may develop sophisticated machine learning models, combining deep learning or reinforcement learning methods with human insights to better understand the relationships among job characteristics, performance and energy impacts, and further embed the learned knowledge into the core allocation and job scheduling strategies. Finally, along with the strong demand for data center services, there are potential improvements in the efficiency of servers, storage devices, network switches and data center infrastructure. A comprehensive study of energy-efficient data center configurations and operations are interesting topics for future research.

Acknowledgments

Authors are listed in alphabetical order. All authors contributed equally to this paper.

Endnotes

[1] See https://www.marketsandmarkets.com/Market-Reports/Quantum-High-Performance-Computing-Market-631.html.

[2] See https://www.statista.com/statistics/500458/worldwide-datacenter-and-it-sites/.

[3] See https://www.readex.eu/.

[4] Energy is the total amount of work performed by a rack over a period of time, whereas power is the rate at which the work is performed by the rack. If measured by unit times, the value of energy and power becomes equal. In this paper, we use the rate measure (watts) in energy consumption for ease of comparison.

[5] In addition to this 75%–25% split into in-sample and out-of-sample data, we also carried out other data partitioning consisting of the first 14, 18, and 24 days of the data in-sample and the rest out-of-sample, respectively. These represent 50%, 64%, and 86% splits of the data into training and test sets, respectively. Results in Online Appendix C show that the performance of our model is robust against the different data partitions we implemented and tested in our experiments.

[6] As shown in Tables A3 and A4 in the online appendix, other data partitioning also achieved similar model performance.

[7] Because $S = 72$ (or 36) and $N = 24$ for CPU (or GPU) racks, by Proposition 1, $X_1 = 864$ (or 432) and $X_2 = 1368$ (or 672) for CPU (or GPU) racks. We see that $864 < 1{,}141 < 1{,}368$ and $432 < 569 < 672$, so τ_{CPU} and τ_{GPU} fall within their respective intervals prescribed by our theoretical proposition.

References

Aghdashi A, Mirtaheri SL (2019) A survey on load balancing in cloud systems for big data applications. Grandinetti L, Mirtaheri S, Shahbazian R, eds. *Proc. Internat. Congress on High-Performance Comput. and Big Data Analysis* (Springer, Berlin), 156–173.

Akbar S, Li R (2022) A Shapley value-based thermal-efficient workload distribution in heterogeneous data centers. *J. Supercomput.* 78:14419–14447.

Atanasov A, Bungartz HJ, Frisch J, Mehl M, Mundani RP, Rank E, van Treeck C (2010) Computational steering of complex flow simulations. Wagner S, Steinmetz M, Bode A, Müller M, eds. *High Performance Computing in Science and Engineering* (Springer, Berlin), 63–74.

Beloglazov A, Buyya R (2010) Energy efficient resource management in virtualized cloud data centers. Parashar M, Buyya R, eds. *Proc. 10th IEEE/ACM Internat. Conf. on Cluster, Cloud and Grid Comput.* (IEEE, New York), 826–831.

Bermejo B, Juiz C, Guerrero C (2019) Virtualization and consolidation: A systematic review of the past 10 years of research on energy and performance. *J. Supercomput.* 75:808–836.

Bruno GSF (2005) Estimation and inference in dynamic unbalanced panel-data models with a small number of individuals. *Stata J.* 5(4):473–500.

Buyya R, Beloglazov A, Abawajy J (2010) Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. Preprint, submitted June 2, https://arxiv.org/abs/1006.0308.pdf.

Chatterjee S, Misra S, Khan SU (2019) Optimal data center scheduling for quality of service management in sensor-cloud. *IEEE Trans. Cloud Comput.* 7(1):89–101.

Chen C, He B, Tang X (2012) Green-aware workload scheduling in geographically distributed data centers. *Proc. IEEE 4th Internat. Conf. on Cloud Comput. Tech. and Sci.* (IEEE, New York), 82–89.

Chen G, He W, Liu J, Nath S, Rigas L, Xiao L, Zhao F (2008) Energy-aware server provisioning and load dispatching for connection-intensive Internet services. *Proc. 5th USENIX Sympos. on Networked Systems Design and Implementation* (USENIX Association, Berkeley, CA), 337–350.

Chernicoff D (2009) *The Shortcut Guide to Data Center Energy Efficiency* (Realtime Publisher, New York).

Chetsa GLT, Lefèvre L, Pierson JM, Stolf P, Costa GD (2014) Exploiting performance counters to predict and improve energy performance of HPC system. *Future Generation Comput. Systems* 36:287–298.

Coffman EG Jr, Garey MR, Johnson DS (1996) Approximation algorithms for bin packing: A survey. Hochbaum DS, ed. *Approximation Algorithms for NP-Hard Problems* (PWS Publishing Co., Boston), 46–93.

Cohen MC, Keller PW, Mirrokni V, Zadimoghaddam M (2019) Overcommitment in cloud services: Bin packing with chance constraints. *Management Sci.* 65(7):3255–3271.

Croxton KL, Gendron B, Magnanti TL (2003) A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Sci.* 49(9):1268–1273.

Dabbagh M, Hamdaoui B, Guizani M, Rayes A (2015) Energy-efficient resource allocation and provisioning framework for cloud data centers. *IEEE eTrans. Network Service Management* 12(3):377–391.

Danani BK, D'Amora BD (2015) Computational steering for high performance computing: applications on Blue Gene/Q system. Watson LT, Weinbub J, Sosonkina M, Thacker WI, Rupp K, eds. *Proc. Sympos. on High Performance Comput.* (Society for Computer Simulation International, San Diego), 202–209.

Dayarathna M, Wen Y, Fan R (2016) Data center energy consumption modeling: A survey. *IEEE Comm. Survey Tutorial* 18(1):732–794.

Delorme M, Iori M, Martello S (2016) Bin packing and cutting stock problems: Mathematical models and exact algorithms. *Eur. J. Oper. Res.* 255(1):1–20.

Deng W, Liu F, Jin H, Liao X, Liu H (2014) Reliability-aware server consolidation for balancing energy-lifetime tradeoff in virtualized cloud datacenters. *Internat. J. Comm. Systems* 27:623–642.

Drozdiak N (2020) EU eyes carbon-neutral data centers by 2030 in green-tech switch. *Energy* (February 6). Accessed October 27, 2022, https://www.datacenterknowledge.com/energy/eu-eyes-carbon-neutral-data-centers-2030-green-tech-switch.

Ei-Moursy AA, Abdelsamea A, Kamran R, Saad M (2019) Multi-dimensional regression host utilization algorithm (MDRHU) for host overload detection in cloud computing. *J. Cloud Comput. Adv. Systems Appl.* 8(8):1–17.

Emerson Network Power (2009) Energy logic: Reducing data center energy consumption by creating savings that cascade across systems. Accessed October 27, 2022, https://www.varinsights.com/doc/energy-logic-reducing-data-center-energy-0008.

Fang X, Sheng OR, Goes P (2013) When is the right time to refresh knowledge discovered from data? *Oper. Res.* 61(1):32–44.

Gmach D, Rolia J, Cherkasova L, Kemper A (2009) Resource pool management: Reactive vs. proactive or let's be friends. *Comput. Networks* 53:2905–2922.

Grandl R, Ananthanarayanan G, Kandula S, Rao S, Akella A (2014) Multi-resource packing for cluster schedulers. *Comput. Comm. Rev.* 44(4):455–466.

Gupta R, Asgari S, Moazamigoodarzi H, Down DG, Puri IK (2021) Energy, exergy and computing efficiency-based data center workload and cooling management. *Appl. Energy* 299:117050.

Gurobi Optimization LLC (2020) Gurobi optimizer reference manual. Accessed August 3, 2021, https://www.gurobi.com/documentation/9.0/refman/index.html.

Hovestadt M, Kao O, Keller A, Streit A (2003) Scheduling in HPC resource management systems: Queuing vs. planning. Feitelson D, Rudolph L, Schwiegelshohn U, eds. *Proc. Workshop on Job Scheduling Strategies for Parallel Processing* (Springer, Berlin), 1–20.

Huang Q, Gao F, Wang R, Qi Z (2011) Power consumption of virtual machine live migration in clouds. Yuan D, Cao M, Wang CX, Huang H, eds. *Proc. 3rd Internat. Conf. on Comm. and Mobile Comput.* (IEEE, New York), 122–125.

Ilager S, Ramamohanarao K, Buyya R (2019) Etas: Energy and thermal aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation. *Concurrent Comput.* 31(17):e5221.

Info-Tech Research Group (2007) Top 10 energy-saving tips for a greener data center. Accessed October 27, 2022, http://static.infotech.com/downloads/samples/070411_premium_oo_greendc_top_10.pdf.

Ketter W, Peters M, Collins J, Gupta A (2016) Competitive benchmarking: An IS research approach to address wicked problems with big data and analytics. *Management Inform. Systems Quart.* 40(4):1057–1080.

Korte B, Vygen J (2006) Bin-packing. *Combinatorial Optimization: Theory and Algorithms* (Springer, Berlin), 425–442.

Kushwaha M, Gupta S (2015) Various schemes of load balancing in distributed systems: A review. *Internat. J. Scientific Res. Engrg. Tech.* 4(7):741–748.

Loeser F, Recker J, Brocke JV, Molla A, Zarnekow R (2017) How IT executives create organizational benefits by translating environmental strategies into Green IS initiatives. *Inform. Systems J.* 27(4):503–553.

Loock C, Staake T, Thiesse F (2013) Motivating energy-efficient behavior with green IS: An investigation of goal setting and the role of defaults. *Management Inform. Systems Quart.* 37(4):1313–1332.

Malhotra A, Melville NP, Watson RT (2013) Spurring impactful research on information systems for environmental sustainability. *Management Inform. Systems Quart.* 37(4):1265–1274.

Melville NP (2010) Information systems innovation for environmental sustainability. *Management Inform. Systems Quart.* 34(1):1–21.

Meng J, McCauley S, Kaplan F, Leung VJ, Coskun AK (2015) Simulation and optimization of HPC job allocation for jointly reducing communication and cooling costs. *Sustainable Comput. Inform. Systems* 6:48–57.

Mišić VV, Perakis G (2020) Data analytics in operations management: A review. *Manufacturing Service Oper. Management* 22(1):158–169.

Nickell SJ (1981) Biases in dynamic models with fixed effects. *Econometrica* 49:1417–1426.

Pakbaznia E, Pedram M (2009) Minimizing data center cooling and server power costs. *Proc. Internat. Sympos. on Low Power Electronics and Design* (IEEE, New York), 145–150.

Pawlish M, Varde AS, Robila SA (2012) Analyzing utilization rates in data centers for optimizing energy management. *Proc. Internat. Green Comput. Conf.* (IEEE, New York), 1–6.

Pitkin E (2018) Slashing HPC energy costs with automated, dynamic optimization. *The Next Platform* (August 24). Accessed October 27, 2022, https://www.nextplatform.com/2018/08/24/slashing-hpc-energy-costs-with-automated-dynamic-optimization/.

Qiu Y, Jiang C, Wang Y, Ou D, Li Y, Wan J (2019) Energy aware virtual machine scheduling in data centers. *Energies* 12(4):1–21.

Rani S, Ahmed SH, Talwar R, Malhotra J (2017) Can sensors collect big data? An energy-efficient big data gathering algorithm for a WSN. *IEEE Trans. Industrial Inform.* 13(4):1961–1968.

Schöne R, Treibig J, Dolz MF, Guillen C, Navarrete C, Knobloch M, Rountree B (2014) Tools and methods for measuring and tuning the energy efficiency of HPC systems. *Sci. Programming* 22(4):273–283.

Shoukourian H, Wilde T, Auweter A, Bode A (2014) Predicting the energy and power consumption of strong and weak scaling HPC applications. *Supercomput. Frontiers Innovations* 1(2):20–41.

Shuja J, Madani SA, Bilal K, Hayat K, Khan SU, Sarwar S (2012) Energy-efficient data centers. *Computing* 94(12):973–994.

Speitkamp B, Bichler M (2010) A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Trans. Service Comput.* 3:266–278.

Takaishi D, Nishiyama H, Kato N, Miura R (2014) Toward energy efficient big data gathering in densely distributed sensor networks. *IEEE Trans. Emerging Top Comput.* 2(3):388–397.

Tang F, Yang LT, Tang C, Li J, Guo M (2018) A dynamical and load-balanced flow scheduling approach for big data centers in clouds. *IEEE Trans. Cloud Comput.* 6(4):915–928.

Tang Q, Gupta SKS, Varsamopoulos G (2008) Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *IEEE Trans. Parallel Distribution Systems* 19(22):1458–1472.

Van Damme T, De Persis C, Tesi P (2019) Optimized thermal-aware job scheduling and control of data centers. *IEEE Trans. Control Systems Tech.* 27(2):760–771.

Van Liere R, Mulder JD, Van Wijk JJ (1997) Computational steering. *Future Generation Comput. Systems* 12(5):441–450.

Varasteh A, Goudarzi M (2015) Server consolidation techniques in virtualized data centers: A survey. *IEEE Systems J.* 11(2):772–783.

Vetter J, Reed D (2000) Real-time performance monitoring, adaptive control, and interactive steering of computational grids. *Internat. J. High Performance Comput. Appl.* 14:357–366.

vom Brocke J, Watson RT, Dwyer C, Elliot S, Melville N (2013) Green information systems: Directives for the IS. *Comm. Assoc. Inform. Systems* 33(30):509–520.

Watson RT, Boudreau MC, Chen AJ (2010) Information systems and environmentally sustainable development: Energy informatics and new directions for the IS community. *Management Inform. Systems Quart.* 34(1):23–38.

Wolke A, Tsend-Ayush B, Pfeiffer C, Bichler M (2015) More than bin packing: Dynamic resource allocation strategies in cloud data centers. *Inform. Systems* 52:83–95.

World Economic Forum (2020) Global innovations from the energy sector 2010-2020. Accessed October 27, 2022, https://www.weforum.org/whitepapers/global-innovations-from-the-energy-sector.