CHENHAN XU

KUN WANG

MINGYI GUO

Intelligent Resource Management in Blockchain-Based Cloud Datacenters

Chenhan Xu, Nanjing University of Posts and Telecommunications
Kun Wang, Nanjing University of Posts and Telecommunications, Shanghai Jiao Tong University
Mingyi Guo, Shanghai Jiao Tong University

Nowadays, more and more companies migrate business from their own servers to the cloud. With the influx of computational requests, datacenters consume tremendous energy every day, attracting great attention in the energy efficiency dilemma. In this paper, we investigate the energy-aware resource

 2325-6095/17/$33.00

management problem in cloud datacenters, where green energy with unpredictable capacity is considered. Via proposing a robust blockchain-based decentralized resource management framework, we save the energy consumed by the request scheduler. Moreover, we propose a reinforcement learning method embedded in a smart contract to further minimize the energy cost. Because the reinforcement learning method is informed from the historical knowledge, it relies on no request arrival and energy supply. Experimental results on Google cluster traces and real-world electricity price show that our approach is able to reduce the datacenters' cost significantly compared with other benchmark algorithms.

Cloud computing is permeating more and more into aspects of our life. Besides traditional Web services such as Web mail, searching, and online education, billions of devices in the Internet of Things (IoT) also upload their data to the cloud. The data volume is undergoing extremely rapid growth. Companies, organizations, personal developers, and even individuals could make use of cloud computing with the realization of platform as a service and infrastructure as a service. Big companies, such as Google, Amazon, and Microsoft, deploy their datacenters (DCs) not only for supporting company business but also as a kind of purchasable service. However, running those enormous DCs consumes tremendous amounts of energy, which compounds the energy crisis and brings severe economical burden. A report[1] from the US Department of Energy National Laboratory shows that the DCs in the US consumed nearly 1.8 percent of the total US electricity consumption, which achieves 70 billion kWh.

There are two main granularity levels that researchers focus on to reduce the energy cost in DCs: server level and DC level. On the server level, Gu et al.[2] proposed a dynamic voltage frequency-scaling (DVFS)–based operational expenditure minimization method. This work considered the minimization problem as mixed integer nonlinear programming (MINLP) and solved the MINLP issue by an iterative searching algorithm. On the DC level, Zhang et al.[3] proposed a network architecture for DCs called Exchanged Cube-Connected Cycles (ExCCC) to achieve a lower cost and higher network throughput. Recently, Chen et al.[4] developed a framework to evaluate the DCs' cost comprehensively. Considering green energy, the framework underpins their energy and maintenance cost joint optimization. However, how to reduce the energy cost of DCs is still an open issue.

In this paper, we study an intelligent energy-aware resource management problem in cloud DCs. Specifically, we consider a series of requests, each of which requires an amount of computational resources to run its virtual machines (VMs). These requests and corresponding VMs are accommodated by several cloud DCs connected to both power grid and fluctuating green energy generated by wind, solar, and tide. By their very nature, cloud computing service providers prefer to run VMs on the DCs that could use more green energy. Google builds DCs near green energy providers and buys green energy other than grid power at a cheap price.[5] Nevertheless, migrating requests and VMs to those DCs might be a high cost when the migration process suffers network congestion. Our aim is to minimize the total energy cost using both grid power and green energy without any further

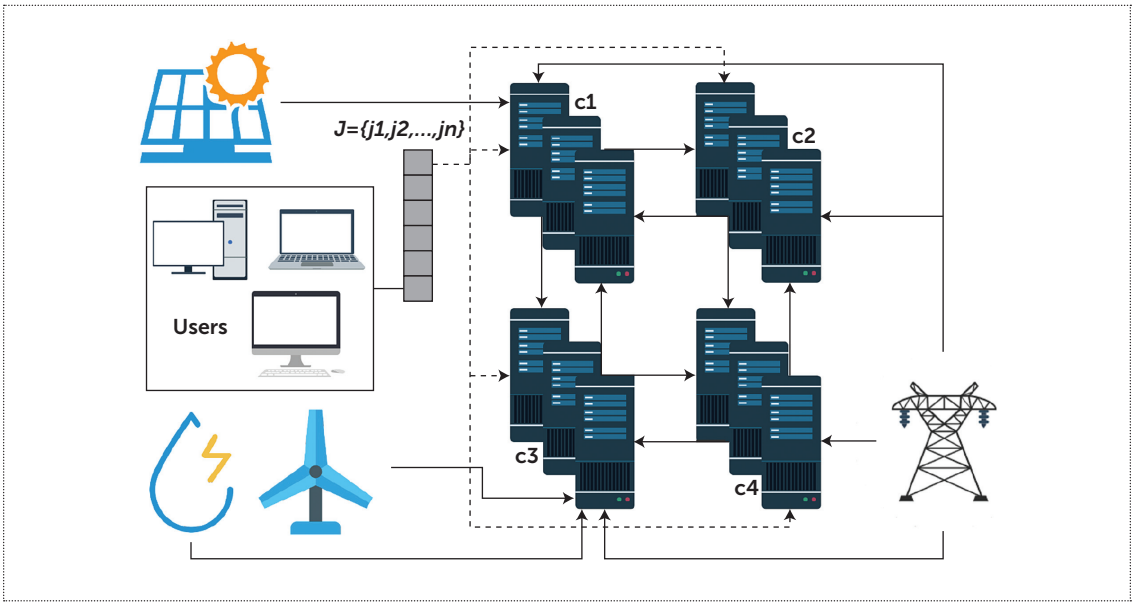

FIGURE 1. System model.

knowledge about future green energy generation. To address this issue, we first propose a resource management framework based on blockchain, a kind of distributed data structure that records all the activities in transactions. Different from other frameworks, with the help of blockchain, our proposal does not require any scheduler, bringing extra energy cost and decreasing the robustness of DCs. Then, a reinforcement learning (RL)–based request migration method in a smart contract is discussed. A smart contract is a script stored and running on our blockchain-based framework and triggered by transactions in our proposed framework. The RL method, which learns from historical knowledge generated from interacting with DCs, does not need any prior knowledge. To this end, the main contributions of the paper are summarized as follows:

1. We consider the request scheduling cost in the energy cost minimization problem of the DCs. Our objective is to minimize the total cost of energy consumption from request scheduling and request migration among DCs.
2. We propose a blockchain-based decentralized resource management framework to save the energy cost by scheduler that exists in most traditional models. The malfunctioning of one DC will not affect the continued resource management, which brings superior robustness to the framework.
3. To minimize the total cost of request migration among DCs, we implement RL-based request migration method by a smart contract in our framework. It is the first attempt for RL to be embedded into the smart contract for solving the energy cost minimization problem.
4. We conduct simulations to evaluate the performance of our proposed algorithm using real-world data traces of renewable power, grid price, and workload. The numerical results show that our proposal has better performance than other benchmark approaches.

System Model and Problem Formulation

As shown in Figure 1, we consider the model that includes cloud DCs, users and their requests, green energy, and grid. DCs distributed in different areas are denoted by a set $C=\{c_1,c_2,\cdots,c_m\}$. Different from typical architectures,[4] users could submit their requests to different DCs. In our proposed blockchain-based framework, these requests can be scheduled by DCs themselves, which removes the dependency on a scheduler in cloud DCs. Particularly, we consider a set of users' requests $R=\{r_0,r_1,\cdots,r_n\}$, each of which asks for more or less computational resources to run VM. Because our proposed framework is decentralized, requests are privileged to be submitted to each DC. In Figure 1, we illustrate the possible requests submissions with dashed paths. Additionally, our model considers a discrete time series, which is expressed as $T=\{t_0,t_1,\cdots,t_n\}$. The data processing rate of requests r_k is denoted by $d_k^j(j\in[1,n])$ that would fluctuate over time t_j. When a large amount of data arrives, the increased computational resource use will consume more energy.

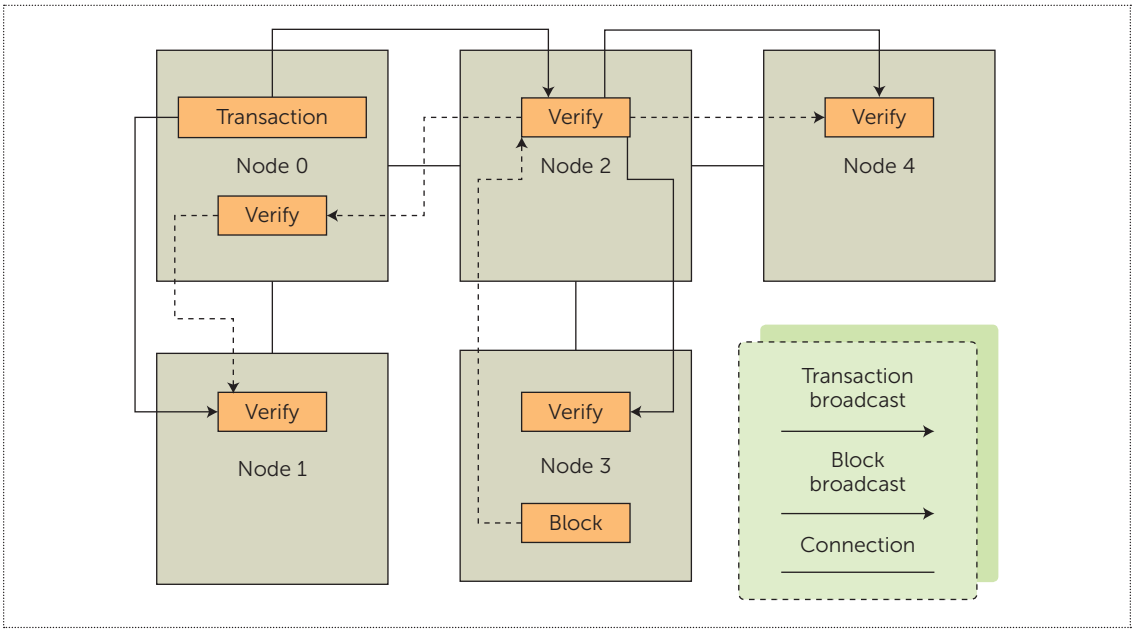

FIGURE 2. The blockchain structure.

On the basis of the denotation previously mentioned, the energy consumption of DC c_i in time slot t_{j-1}, t_j is given as

$$U_i^j = \sum_{r_k \in R_i^j} \Theta_i\left(d_j^k\right), \tag{1}$$

where $\Theta_i\left(d_k^j\right)$ denotes the energy consumed by VM running on DC c_i when it processes data at the rate d_k^j. The $\Theta_i(\cdot)$ mostly depends on the hardware of DC c_i. The requests accommodated by DC c_i in time slot t_{j-1}, t_j are defined by R_i^j in Li et al.[5] We denote the green energy generated by DC c_i in time slot t_{j-1}, t_j with g_i^j. It is assumed that the DCs use green energy with higher priority such as Google,[5] and the insufficient parts are powered by the traditional grid. The green energy and grid electricity price are denoted with ${}^G p_i^j$ and ${}^U p_i^j$, respectively, ${}^U p_i^j$ is usually given by regional transmission organization, and ${}^G p_i^j$ is set to 0 for those DCs that have built-in wind and solar farms. Hence, the total energy cost in time slot t_{j-1}, t_j is given as

$$P_E^j = \sum_{c_i \in C} [{}^U p_i^j \times \max\left(u_i^j - g_i^j, 0\right) + {}^G p_i^j \times g_i^j]. \tag{2}$$

The management needs to migrate requests and its VMs among DCs and to redirect the dataflow by updating the routing rules at gateways. Let $A(r_k, j)$ denote the DC accommodating request r_k in time slot t_{j-1}, t_j, and we express the migration cost as

$$C_M^j = \sum_{r_k \in R_i^j, i \in [0,m]} q_k^j(A(r_k, j), A(r_k, j-1)), \tag{3}$$

where $q_k^j(i, i')$ is the cost of migrating requests r_k from c_i to c_i' in time slot t_{j-1}, t_j, depending on bandwidth costs and the extended time penalty.

Finally, the total cost over the whole time slots on energy consumption and request migration can be calculated by

$$E = \sum_{t_j \in T} (P_E^j + C_M^j). \tag{4}$$

Additionally, we define the panoramic view of requests among DCs at time t_j as

$$V_j = \{R_1^j, R_2^j, \cdots, R_m^j\}. \tag{5}$$

Our proposed framework aims to manage users' requests among cloud DCs to lower the total energy cost. Hence, we are going to minimize the total cost E by planning the view V and executing migration at the beginning of each time slot. These two operations are performed without the knowledge of future incoming green energy g_i^j and request migration cost function $q_k^j(i, i')$.

Blockchain-Based Resource Management

In this section, we are going to discuss how the blockchain is used in managing the requests and VMs among cloud DCs.

Preliminary

Blockchain is a linked data structure that is kept by every participant of a blockchain network. It was proposed by S. Nakamoto to solve the consensus problem of the Bitcoin network.[6] As shown in Figure 2,

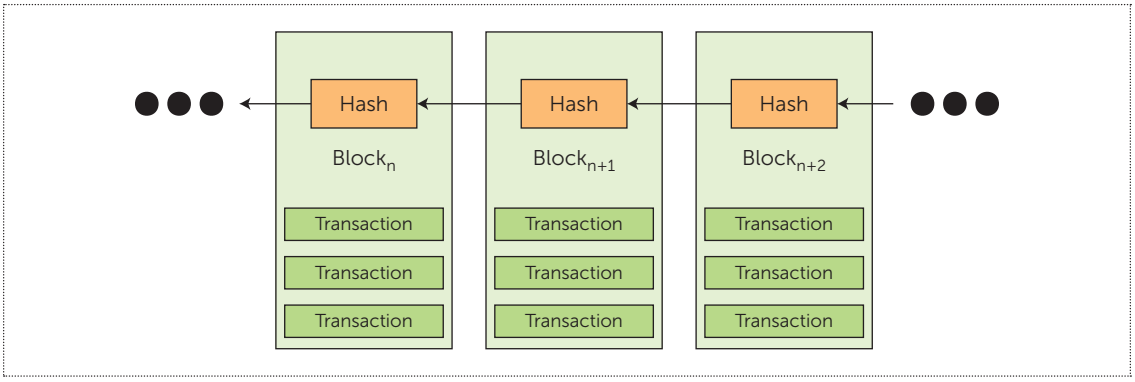

FIGURE 3. A blockchain network.

blockchain is organized as a single list, where every block contains the hash of the previous block except the first block (genesis block). The former block is always generated earlier than the latter, and every block carries some transactions, which are records of blockchain activities, i.e., assets transferring. We further detail the blockchain generation mechanism in Figure 3. The figure depicts that

1. A user uses his private key to sign a transaction while interacting with $Node_0$. Hence, the transaction could be traced via the user's public key, and the digital signature also strengthens security and data integrity. Then, the transaction is broadcast to one-hop neighbor of $Node_0$ (i.e., $Node_1$ and $Node_2$).
2. The neighboring nodes (i.e., $Node_1$ and $Node_2$) verify the broadcast transaction obeying the transaction protocol and broadcast it to neighbors ($Node_3$ and $Node_4$), or the transaction will be dropped.

Note that the transaction protocol is determined at the time when the blockchain is being designed, each network should make it clear to every participant. The main motivation of transaction protocol is preventing chaos in the blockchain network.

By repeating the procedures mentioned previously, this transaction finally spreads across the whole blockchain network. All transactions generated by the network during an agreed time interval by all participants are packaged into one block by a mining node (miner; see $Node_3$ in Figure 3), then

1. The miner (i.e., $Node_3$) broadcasts the block to blockchain network. Like the transaction, block is also broadcast peer to peer. Hence, $Node_2$ receives the block.
2. The receiver (i.e., $Node_2$) verifies
   a. all the transactions contained by the block obey the transaction protocol; and
   b. the block has a correct hash of the previous block on the blockchain.

If the block passes the verification, the receiver adds it to the blockchain and extracts the transactions that the block contains to update the receiver's transactions, which are also called "view of the world."[6] Otherwise, the block is discarded.

Note that the choice of miner is leveraged by the consensus mechanism of the blockchain network. In Bitcoin, the node first finding the random value is entitled to propose the next block of the blockchain.[6] This value should be able to generate a number less than a given threshold after performing double SHA-256 hash operations to the concatenation of the previous block content and the value.[6] The finding process, also called mining, is a kind of consensus mechanism.

Because the blockchain is kept by every participant of the blockchain network and a network would end up with any divergence among participants, the consensus mechanism is extremely important. The blockchain that anyone can participate in (public network) generally uses proof of work (PoW) or proof of stake (PoS)[6] as its consensus mechanism. In the meantime, there are various consensus mechanisms, e.g., practical Byzantine fault tolerance and Tangaroa, for the network that only open to white list members (private network).[6]

Blockchain-Based Resource Management Framework

The core mechanisms of our proposed framework are transaction, mining, and a smart contract. Transaction is used to execute migration, mining brings superior robustness, and a smart contract supports various user-designed algorithms.

TABLE 1. Datacenter-adapted transactions.

ID	Source	Destination	Resource CPU	RAM	Disk	I/O	GPU
0	$Node_0$	Update	−32	−128	-2^{16}	−5	−8
1	Alice	$Node_0$	$-\alpha$	$-\beta$	$-\gamma$	$-\delta$	$-\epsilon$
2	$Node_0$	$Node_1$	$-\alpha$	$-\beta$	$-\gamma$	$-\delta$	$-\epsilon$
3	Finish	$Node_1$	α	β	γ	δ	ϵ

Transaction. Within the cloud DCs context, the transaction serves request migration by recording the resource allocation of request VMs. To illustrate the mechanism of applying transaction in request migration, we introduce our proposed DC-adapted transaction protocol. The content of this protocol includes the request ID, the request migration source and destination, and resource allocation, which are presented in Table 1. As shown in the table, a transaction whose ID is 1 represents that Alice submits a request that asks for α CPU core, β GB RAM, γ GB disk storage, δ Mbps I/O, and ò GPU cards. Note that the number of required resources is negative, which represents that the request *requires* resources. The transaction whose ID is 2 shows that $Node_0$ migrates a VM created by Alice to $Node_1$. Specially, we use a transaction, whose source is *nodename* and destination is *update*, to update the available resource of the node. The typical situation to use this transaction is introducing a new node, e.g., transaction whose ID is 0 that introduces $Node_0$ in Table 1 to the DCs network. When a *nodename* is in the *destination* row, it should add the resource listed in the transaction and vice versa.

Assuming that $Node_0$ has no resource at the beginning and transactions in Table 1 happen according to the order, the CPU resource left can be calculated as $0-(-32)+(-a)-(-a)=32$. We now go over how $Node_0$ migrates the request to $Node_1$ by transaction, i.e., what happens to these two nodes in the migration process. For the first step, $Node_0$ creates the transaction, declares the resource that the request needs, gives the transaction an ID, and sets $Node_1$ as destination. If we consider the blockchain network as a database, $Node_0$ is creating a record that shows $Node_0$ has no resource allocated (the request is migrating to $Node_1$) and deleting the old record, where a part of the resource is allocated to the request. On the $Node_1$ side, the inverse operations are being performed. The creation and deletion are to modify the record of resource. (Records are privileged to be deleted and created but not directly modified, which is done to prevent conflicts among nodes.[6]) For the second step, the transaction is broadcast and verified by nodes in the network. The verification confirms that

1. the transaction does not address the record that has been deleted, because the resource should not be allocated by both sides; and
2. the amount of resource to be allocated is correct. For example, if a transaction transfers 16 CPU cores from $Node_0$ to $Node_1$, but there are 8 cores left in $Node_0$, the upcoming transfer would fail.

Mining. Our proposed framework allows every DC to perform mining. Because all the participants of the framework are DCs, the framework does not require mechanisms, such as PoW and PoS, resulting in tremendous energy expenditure. In the proposal, the DCs mine the next block as follows:

1. Sort the list of DCs by load in decreasing order;
2. Eliminate DCs that used to mine the previous λ blocks in the list; and
3. The first DC in the list mines the next block.

The other DCs will not attempt to mine a block to save the energy consumption. In the procedures previously mentioned, λ is the parameter used to control mining process. The larger λ is, the higher possibility the heavy-load DC will be chosen to mine. We assume each DC has a probability τ of malfunction, the failure probability of mining is given by

$$P_f = P(Bin(m,1-\tau) \le \lambda), \tag{6}$$

where m is the number of DCs and $Bin(\cdot)$ represents the binomial distribution. If $m=20$, $\tau=2.5\%$, the setting $\lambda=14$ keeps P_f below 0.001 percent, which brings superior robustness to the DCs.

Smart Contract. A smart contract is a script stored in our blockchain-based framework, which is triggered by transactions sent to it. The DCs migrate

requests and VMs by executing the smart contract. Consider a simple DCs network that has participants $\{DC_1, DC_2, \cdots, DC_l\}$. The network always migrates requests and VMs to the DC that has the lowest load. The smart contracts stored in each DC can be designed as Algorithm 1.

Algorithm 1 indicates the logic of the smart contract, and it needs no centralized controller. When a block is accepted by a DC, the smart contract running on this DC checks every transaction it contains. When the smart contract find a transaction representing request migration to the DC where it runs, it migrates the migrated requests to the DC that has the lowest load, except itself running on the DC with the lowest load.

Case Study: RL-Based Energy Cost Minimization

In this section, we investigate the RL-based energy cost minimization. The RL-based method is presented first, and the experimental results show the performance of our method.

RL Mechanism

RL is an approach learning what to do in different situations to maximize profit. The key elements of RL are state, action, reward, and agent. The learning process of an agent includes a series of actions and the corresponding reward. In each state, agent evaluates the expected profit of various possible actions by value function (i.e., value function is a function mapping state and action to reward). Then, the agent, according to a certain policy, selects an action to take, and the state thereafter is changed. The reward associated with the former state and the taken action is used to update the value function. Generally, the profit, also called return, is the accumulated reward that measures the benefit of a taken action in a certain state. The RL is an ideal solution to reduce the energy cost among complex DCs, because it does not need any prior knowledge.

In our proposal, the idea is to migrate requests and VMs among DCs according to the historical migrating decisions' energy cost. We implement the idea by a smart contract, which is triggered by every incoming request. In other words, once a request submission or resource allocation happens, our cost minimization algorithm implemented by a smart contract will be triggered. In each learning iteration led by a request, DCs first select an action (migrates requests and their VMs to DCs). Then, the migration is performed. Finally, the new state and reward (DCs' load and energy cost) is obtained for learning. Specifically, actions contain all the possible migrations of requests and VMs among DCs, and states are the load situations that the DCs could achieve. At the start of each time slot, the next mining DC obtains the panoramic view of requests $s_j = V_j = \{R_1^j, R_2^j, \cdots, R_m^j\}$ and determines the location of requests and VMs after migration $s_{j+1} = V_{j+1} = \{R_1^{j+1}, R_2^{j+1}, \cdots, R_m^{j+1}\}$. Then, DCs perform migration a_j by comparing V_j with $V_j + 1$. After that, the reward r_j could be evaluated by Equation 4. This process can be expressed as

$$s_j + a_j \rightarrow r_j + s_{j+1}, \tag{7}$$

which means action a_j is taken at state s_j and the state changes to s_{j+1} with reward r_j. We model the migrations as a Markov decision process (MDP):

$$s_0 + a_0 \rightarrow r_0 + s_1 + a_1 \rightarrow \cdots \rightarrow r_n + s_{n+1}. \tag{8}$$

Based on this MDP model, we formulate the value function $Q(s_j, a_j)$ as

$$\begin{aligned} Q(s_j, a_j) &= E\left[r_{j+1} + \gamma r_{j+2} + \gamma^2 r_{j+3} + \cdots + \gamma^{n-j-1} r_n\right] \\ &= E\left[r_{j+1} + \gamma\left(r_{j+2} + \gamma r_{j+3} + \cdots + \gamma^{n-j-2} r_n\right)\right] \\ &= E\left[r_{j+1} + \gamma Q\left(s_{j+1}, a_{j+1}\right)\right], \end{aligned} \tag{9}$$

where $\gamma \in (0,1]$ is a discount factor that makes the nearer reward more important. Based on Equation 9, the value function could perform incremental learning (Q-learning[7]) by

$$\begin{aligned} Q(s_{j+1}, a_{j+1}) &= (1-\mu) Q(s_{j+1}, a_{j+1}) \\ &+ \mu[r_{j+1} + \gamma \min Q(s_j, a_j)], \end{aligned} \tag{10}$$

where μ represents the learning rate.

We present our RL-based migration method in Algorithm 2. We use the policy called π^{φ}, which chooses a random action with a probability of φ, or use policy π to choose an action. The policy π is a greedy strategy given as

$$\pi(s) = arg \min_a Q(s, a), \tag{11}$$

which means choosing the action that has the lowest value at state s. Because every DC could take part in the requests and VMs migration, the smart contract running Algorithm 2 should be distributed to every DC. To reduce the energy cost, as presented in line 9, Algorithm 2, the DC that performs learning, i.e., the agent, will broadcast the current learned parameters so that other DCs do not need to learn

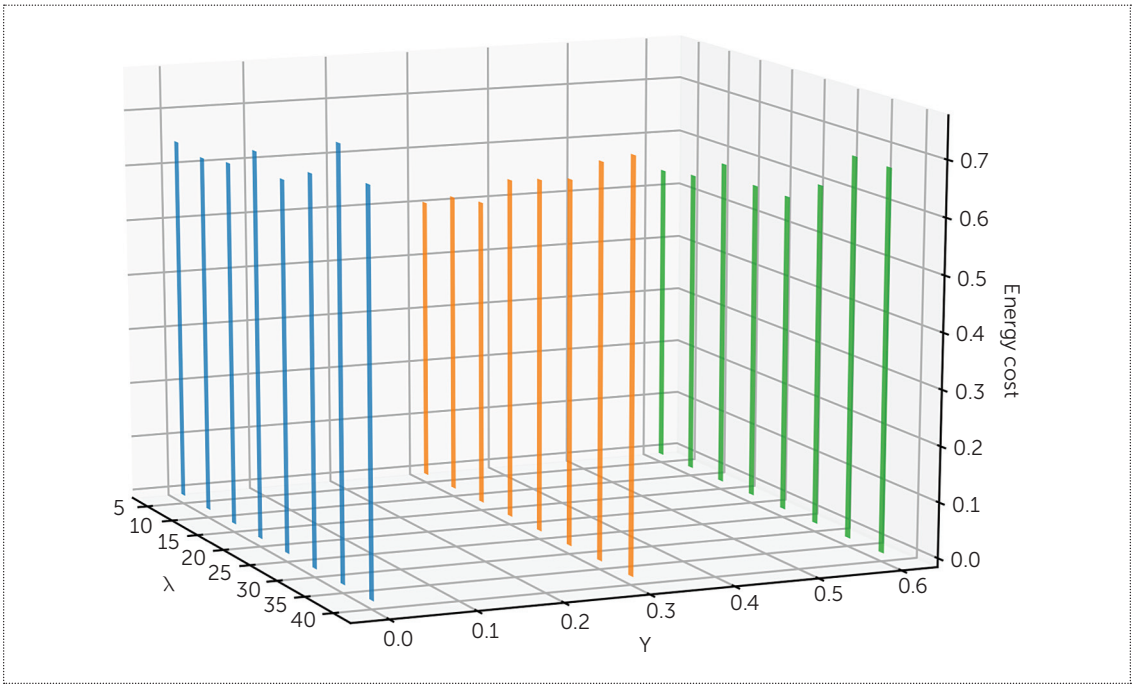

FIGURE 4. The energy cost changes over different λ and γ.

them repeatedly. This broadcast operation is similar to those transactions listed in Table 1. The parameters to be broadcast are treated as some resources recorded by transactions.

Simulation Results

In our experiments, we use the request trace from 1-mo Google cluster workload traces collected in May 2011.[8] The traces contain information such as request arrival time and CPU usage of each request, and we randomly generate the lacked information about GPU usage. The average request arrival rate is about 200 requests per minute. This experiment is conducted in three DCs, which are Prewitt in New Mexico, Phoenix in Arizona (AZ), and Los Angeles in California (CA), and renewable energy is estimated according to the weather conditions published by National Renewable Energy Laboratory. The electricity price is provided by Energy Information Administration. The learning rate of our algorithm is set to 0.8.[7] We normalize the cost to the result of Round-robin (RR), where DCs take turns to accommodate requests.

Figure 4 shows the influences of parameters λ and γ to the whole energy cost. As illustrated in the figure, under different values of γ, a larger λ makes energy cost higher. On the contrary, the larger γ helps to reduce the energy cost. On average, the cost is saved about 60 percent more than RR.

The comparison among RR, MinBrown (MB), and our method is depicted in Figure 5. Our

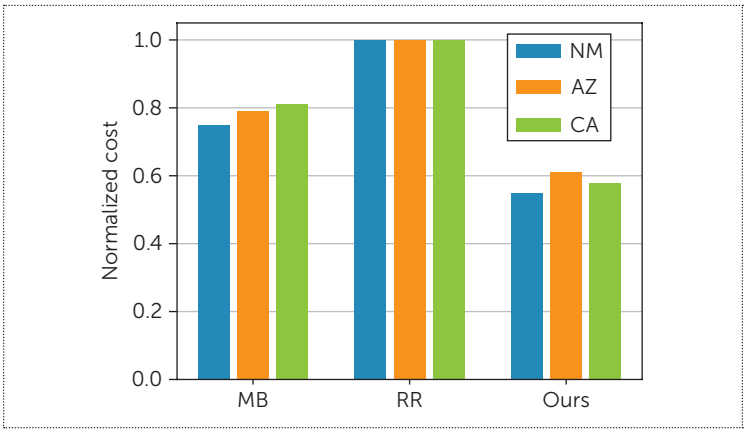

FIGURE 5. Comparisons among RR, MB, and our method on energy cost.

RL-based energy cost minimization method considers jointly green and migration energy; it is obvious that the proposal could save 50 percent more energy cost than RR, which is about 20 percent more than MB. Because MB just considers maximizing the green energy usage but not migration cost, the energy cost saving is limited.

Related Works

Green DCs

Cloud DCs consume tremendous energy;[4,9] thus, green DCs are in urgent demand. Gu et al.[2] proposed a DVFS-based operational expenditure

minimization method and developed the corresponding resolution search algorithm. Zhang et al.[3] proposed ExCCC, a new type of interconnection network that can be employed by DCs, providing low energy consumption and high scalability with the help of its network architecture. Chen et al.[4] introduced a comprehensive framework to evaluate server power, cooling, hardware maintenance cost. Based on this framework, they performed server and inlet cooling water joint cost optimization and gained a considerable cost saving. Recently, some works have introduced RL into the DC field. Zhou et al.[7] proposed a single-agent VM selection method. Zhu et al.[10] combined game theory and RL to assign resources to users more fairly.

Our work for the first time takes the cost of a request scheduler into account. We remove the scheduler in our proposed framework so that DCs could interact peer-to-peer. In this case, the availability of scheduler does not affect the whole network.

Blockchain and Smart Contract

Blockchain recently has attracted the interest of researchers in various fields, including finance, e-health, and distributed systems. A comprehensive survey about blockchain was given by K. Christidis and M. Devetsikiotis.[6]

Saito and Yamada et al.[11] used a probabilistic state machine to model blockchain and gave it an enhanced formal representation. The authors proved that the consensus cannot be reached under the circumstance in which the number of blockchain participants is uncertain.

Kishigami et al.[12] proposed a blockchain-based digital content distribution system. The proposal can easily support the content owner's rights control operation, and the security is also guaranteed.

Ethereum[13] is a blockchain-based decentralized application platform supporting a Turing-complete smart contract, which was first proposed by Szabo in 1994[6] and, in fact, is a program stored in blockchain. Yuan and Wang[14] proposed a blockchain based intelligent transportation system (ITS) framework, which is an elegant solution to the security problem and performance limitation in ITS. The main algorithms are implemented in smart contract layer. Huh et al.[15] discussed the synchronization issue in IoT with the current server-client model and proposed a seven-layer blockchain platform–based IoT device management system. The platform was built on Ethereum with a smart contract.

This research highlights the importance of a smart contract in the blockchain network. Inspired by such a concept, our proposal further investigates the application of a smart contract and combines RL with a smart contract to perform cost minimization among cloud DCs.

Conclusion

With significant improvements in cloud computing technologies and applications in IoT, we have witnessed an explosion of data. Massive amounts of data are generated in DCs, which not only evoke various promising data-driven services but also consume tremendous energy every day. In this paper, we are concerned with the cost minimization issues in cloud DCs and study how to reduce the total cost of energy consumption from the traditional power grid, request scheduling cost, and request migration in DCs. To this end, we develop a blockchain-based decentralized resource management framework, where requests can be scheduled by DCs themselves without depending on the scheduler in cloud DCs. Furthermore, we propose the RL-based request migration method with an embedded smart contract in our framework for cost saving. Finally, simulations are operated based on Google cluster traces and the real-world electricity price, demonstrating the superior performance in energy cost saving compared with other benchmark algorithms in DCs.

Acknowledgments

This work is supported by the National China 973 Project (2015CB352401) and China Postdoctoral Science Foundation (2017T100297 and 2017M610252).

References

1. A. Shehabi et al., *United States Data Center Energy Usage Report,* Lawrence Berkeley National Laboratory, Berkeley, CA, tech. report 2016.
2. L. Gu et al., "Optimal Task Placement with QoS Constraints in Geo-Distributed Data Centers Using DVFS," *IEEE Trans. Computers,* vol. 64, no. 7, 2015, pp. 2049–2059.
3. Z. Zhang et al., "ExCCC-DCN: A Highly Scalable, Cost-Effective and Energy-Efficient Data Center Structure," *IEEE Trans. Parallel Distribution Systems,* vol. 28, no. 4, 2017, pp. 1046–1060.
4. S. Chen, S. Irving, and L. Peng, "Operational Cost Optimization for Cloud Computing Data Centers Using Renewable Energy," *IEEE Systems J.,* vol. 10, no. 4, 2016, pp. 1447–1458.
5. P. Li et al., "Traffic-Aware Geo-Distributed Big Data Analytics with Predictable Job Completion Time," *IEEE Trans. Parallel Distribution Systems,* vol. 28, no. 6, 2017, pp. 1785–1796.

6. K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access,* vol. 4, 2016, pp. 2292–2303.
7. X. Zhou et al., "Reinforcement Learning-Based Adaptive Resource Management of Differentiated Services in Geo-Distributed Data Centers," *Proc. 2017 IEEE/ACM 25th Int'l Symp. Quality of Service* (IWQoS), 2017, pp. 1–6.
8. B. Liu, Y. Lin, and Y. Chen, "Quantitative Workload Analysis and Prediction Using Google Cluster Traces," *Proc. 2016 IEEE Conf. on Computer Communications Workshops* (INFOCOM WKSHPS), 2016, pp. 935–940.
9. K. Wang et al., "Green Industrial Internet of Things Architecture: An Energy-Efficient Perspective," *IEEE Comm. Magazine,* vol. 54, no. 12, 2016, pp. 48–54.
10. Q. Zhu and J.C. Oh, "Learning Fairness Under Constraints: A decentralized Resource Allocation Game," *Proc. 2016 15th IEEE Int'l Conf. on Machine Learning and Applications* (ICMLA), 2016, pp. 214–221.
11. K. Saito and H. Yamada, "What's So Different About Blockchain?—Blockchain Is a Probabilistic State Machine," *Proc. 2016 IEEE 36th Int'l Conf. on Distributed Computing Systems Workshops* (ICDCSW), 2016, pp. 168–175.
12. J. Kishigami et al., "The Blockchain-Based Digital Content Distribution System," *Proc. 2015 IEEE Fifth Int'l Conf. on Big Data and Cloud Computing,* 2015, pp. 187–190.
13. Ethereum Foundation. *Ethereum,* white paper, Sept. 18, 2017; https://github.com/ethereum/wiki/wiki/White-Paper.
14. Y. Yuan and F.Y. Wang, "Towards Blockchain-Based Intelligent Transportation Systems," *Proc. 2016 IEEE 19th Int'l Conf. on Intelligent Transportation Systems* (ITSC), 2016, pp. 2663–2668.
15. S. Huh, S. Cho, and S. Kim, "Managing IoT Devices Using Blockchain Platform," *Proc. 2017 19th Int'l Conf. on Advanced Communication Technology* (ICACT), 2017, pp. 464–467.

CHENHAN XU *is an undergraduate student in the School of Internet of Things, Nanjing University of Posts and Telecommunications, China. His current research interests include big data, cloud computing, and machine learning. Contact him at xchank@outlook.com.*

KUN WANG *received BEng and PhD degrees at the School of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004 and 2009, respectively. From 2013 to 2015, he was a Postdoctoral Fellow in the electrical engineering department, University of California, Los Angeles. In 2016, he was a Research Fellow in the School of Computer Science and Engineering, the University of Aizu, Aizu-Wakamatsu City, Fukushima, Japan. He is currently a Research Fellow in the department of computing, the Hong Kong Polytechnic University, Hong Kong, China, and also a professor at the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China. His current research interests are mainly in the area of big data, wireless communications and networking, smart grid, energy Internet, and information security technologies. He is a Senior Member of IEEE and a Member of ACM. Contact him at kwang@njupt.edu.cn.*

MINYI GUO *received a PhD degree in computer science from the University of Tsukuba, Tsukuba, Japan. He is currently a Zhiyuan Chair Professor in Shanghai Jiao Tong University, Shanghai, China. His research interests include pervasive computing, parallel and distributed processing, and parallelizing compilers. In 2007, he received the Recruitment Program of Global Experts and Distinguished Young Scholars Award from the National Natural Science Foundation of China. He is on the editorial board of IEEE Transactions on Parallel and Distributed Systems. Contact him at guo-my@cs.sjtu.edu.cn.*