Article

Optimal Cloud Orchestration Model of Containerized Task Scheduling Strategy Using Integer Linear Programming: Case Studies of IoTcloudServe@TEIN Project

Nawat Swatthong and Chaodit Aswakul *

Wireless Network and Future Internet Research Unit, Department of Electrical Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand; nawat.sw@alumni.chula.ac.th

* Correspondence: chaodit.a@chula.ac.th

Abstract: As a playground for cloud computing and IoT networking environment, IoTcloudServe@TEIN has been established in the Trans-Eurasia Information Network (TEIN). In the IoTcloudServe@TEIN platform, a cloud orchestration for conducting the flow of IoT task demands is imperative for effectively improving performance. In this paper, we propose the model of optimal containerized task scheduling in cloud orchestration that maximizes the average payoff from completing tasks within the whole cloud system with different levels of cloud hierarchies. Based on integer linear programming, the model can take into account demand requirement and resource availability in terms of storage, computation, network, and splittable task granularity. To show the insights obtainable from the proposed model, the edge-core cluster of IoTcloudServe@TEIN and its peer-to-peer federated cloud scenario with OF@TEIN+ are numerically experimented and herein reported. To evaluate the model's performance, payoff level and task completion time are considered by comparing with a well-known round-robin scheduling algorithm. The proposed ILP model can be a guideline for the cloud orchestration in IoTcloudserve@TEIN because of the lower task completion time and the higher payoff level especially upon the large demand growth, which is the major operation range of concerns in practice. Moreover, the proposed model illustrates mathematically the significance of implementing cloud architecture with refined splittable task granularity via the light-weighted container technology that has been used as the basis for IoTcloudServe@TEIN clustering design.

Keywords: cloud orchestration; integer linear programming; cloud computing; container technology

Citation: Swatthong, N.; Aswakul, C. Optimal Cloud Orchestration Model of Containerized Task Scheduling Strategy Using Integer Linear Programming: Case Studies of IoTcloudServe@TEIN Project. *Energies* **2021**, *14*, 4536. <https://doi.org/10.3390/en14154536>

Academic Editors: Francesco Bellotti and Riccardo Berta

Received: 23 May 2021

Accepted: 21 July 2021

Published: 27 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Nowadays, cloud computing [1] is so commercially widespread, and IoT technology is considered a state-of-the-art future technology that can connect several devices into a communication system. As a research playground for understanding cloud computing and IoT networking in a natural environment, the IoTcloudServe@TEIN project [2] has been established with the emphasis on collaborative efforts in inter-connecting cloud resources distributed and managed by participating countries in the Trans-Eurasia Information Network (TEIN) [3]. IoTcloudServe@TEIN utilizes container technology [4] from Kubernetes [5], which helps managing resources throughout the whole system. Kubernetes consists of two parts called K-master and K-worker. As shown in Figure 1, there are two levels of resource hierarchy in IoTcloudServe@TEIN in Thailand—core and edges. IoTcloudServe@TEIN is also connected to OF@TEIN+ [6], a gigantic WAN community full of usable cloud resources shared by participating countries.

To provide a cost-effective and power-efficient policy, establishing cloud orchestration as well as federation [7] is essential that IoTcloudServe@TEIN needs to implement to cooperate with several interactive cloud clusters throughout Asia. In the past, researchers have put efforts into finding the best solution for allocating tasks and optimizing power consumption depending on different assumptions. The linear programming has been

formulated in [8] to manage cloud resources from green energy and in [9] to solve optimal processes deployment on hierarchical cloud resources. Both works in [8,9], however, lack the demand's granularity concern. In [10], the authors concentrate on predicting the cost of utilization on private cloud networks and finding a linear programming solution to minimize the cost within an interval time. However, they do not concern about the time of task completion, which is one of the crucial quality of services.

Figure 1. Overall conceptual structure of IoTcloudServe@TEIN edge-core cluster and cloud federation with OF@TEIN+.

On the contrary, the work in [11] prioritizes the time for task computation and the energy used for computing each task. Additionally, the work in [12] applies container technology into the mathematical model of task allocation to minimize the total cost of computation. An integer linear programming to minimize energy consumption [13] has been proposed by considering the main three parts of cloud computing—computation, networking, and storage resources. However, in [11–13], there is no consideration on the cloud hierarchy that should be focused on since cloud system is usually interconnected with several types of other cloud systems.

To the best of our knowledge, there was thus no complete work in the past that could be directly applied to all cloud orchestration requirements in the IoTcloudServe@TEIN project. Therefore, our primary motivation for this paper is to develop a cost-effective and power-efficient cloud federation model that can be practically deployed in the IoTcloudServe@TEIN environment. Notably, as a playground, IoTcloudServe@TEIN is considered a private hierarchically structured cloud system that needs high additional cost to forward tasks to other connected public cloud systems. Moreover, since IoTcloudServe@TEIN is also connected to other private cloud clusters in the TEIN community that may provide different patterns of payoff calculation, cloud federation for conducting the flow of demand is imperative for effectively improving performance. Therefore, the proposed cloud federation model will consider the cloud hierarchy that reflects resource prioritization among different types and resource levels of cloud clusters. To evaluate insightful scenarios here, we propose the theoretical model of optimal containerized task scheduling in cloud orchestration that maximizes the average payoff from completing tasks within the whole cloud system with different levels of cloud hierarchies. The main contributions of this paper are as follows:

- Integer Linear Programming (ILP) model formulation for maximizing the average payoff from completing tasks by considering different levels of cloud hierarchies.
- Mimicking scenarios of peer-to-peer federated and edge-core cloud clusters with the analysis of the ILP model being used in each scenario.

- The emphasis on the importance of implementing cloud architecture with refined splittable task granularity.

This paper is organized as follows: Section 2 presents the task scheduling model in cloud orchestration, Section 3 presents the formulation of the proposed ILP model, Section 4 presents the parameter for model experiment and evaluation, Section 5 presents the results and discussion, and Section 6 presents the conclusion.

2. Task Scheduling Model in Cloud Orchestration

As shown in Figure 2, cloud orchestration consists of a K-master, a task scheduler, and several K-workers. K-master receives the characteristic parameters of requesting task i to rearrange the scheduling policy at the task scheduler. Then, task i is scheduled to be split into several sub-tasks, each of which is allocated to K-worker j 's. All K-workers can be divided into C_I and C_O : the set of K-workers at the private cloud and the public cloud. We need to avoid utilizing the latter because the public cloud typically incurs high operational costs.

Figure 2. Task scheduling model in cloud orchestration.

Table 1 summarizes all the mathematical notations used for our formulation. In general, there is no correlation between data size and the number of instructions of each sub-task after the task is split. However, without losing mathematical generality in this article, the assumption regarding the ability to split is conducted in that the storage and computational requirement of each sub-task is split at the same proportion $x_{i,j}$. This means that task i is allocated to K-worker j with input data size $x_{i,j}T_i^S$ and computing instruction $x_{i,j}T_i^C$.

Table 1. Model notations.

Notation	Description
$x_{i,j}$	Proportion of task i which is allocated to the K-worker j
T_i^C	Number of instructions for computing task i on CPU
T_i^S	Input data size of task i before task execution (bits)
T_i^T	Maximum allowable completion time for task i before the task execution is timed out
T_i^β	Ratio between the input and output data size of task i before and after the task completion respectively
P_i	Set of the schedulable proportion of task i , $P_i = \left\{0, \frac{1}{\epsilon}, \frac{2}{\epsilon}, \dots, 1\right\}$ where ϵ is the number of splittable equally sized units for task i
R_j^C	Speed of CPU for computing on K-worker j (million instructions per second, MIPS)
$R_{i,j}^N$	Communication bandwidth from the source of task i to K-worker j where the input data of task i must be delivered
$R_{i,j}'^N$	Communication bandwidth from K-worker j to the destination where the output data of task i must be delivered
R_j^S	Storage capacity of K-worker j to store the input data required for the execution of task i
N	Set of all tasks

For each task i and K-worker j , as a major concern in cloud federation, task completion time t_i consists of the transmission time of each sub-task in the network $t_{i,j}^N$ and the computation time for sub-task on K-worker $t_{i,j}^C$. The task completion time is calculated from the longest completion time among the sub-tasks, $t_i = \max(t_{i,1}^N + t_{i,1}^C, t_{i,2}^N + t_{i,2}^C, \dots)$. The task completion time must not exceed the maximum allowable total task completion time T_i^T , $t_{i,j}^N + t_{i,j}^C \leq T_i^T$, $\forall i \in N$, $\forall j \in C_I \cup C_O$.

The transmission time of each sub-task is the summation of the period of time for transmitting the sub-task from the requesting location to the assigned K-worker and the period of time for forwarding the completed sub-task from the K-worker to the destination of the task, $t_{i,j}^N = \frac{T_i^S}{R_{i,j}^N} x_{i,j} + T_i^B \frac{T_i^S}{R_{i,j}^N} x_{i,j}$, $\forall i \in N$, $\forall j \in C_I \cup C_O$. Moreover, compressibility T_i^B is linearly assigned as the data size of the task may be reduced after the task is completed at K-worker.

Considering the computation time $t_{i,j}^C$, it is crucial to examine the worst-case scenario in order to set the upper bound of the total completion task time. In particular, when sub-task i is allocated to K-worker j , the sub-task needs to wait, in the worst case, until other sub-tasks which are also allocated to K-worker j are completely computed in order to begin computing sub-task i . Therefore, the computation time of sub-task i on K-worker j can be upper-bounded by the summation of computation time of all sub-tasks that are allocated to K-worker j , $t_{i,j}^C \leq t_j^C = \sum_{i \in N} \frac{T_i^C}{R_j^C} x_{i,j}$, $\forall j \in C_I \cup C_O$.

3. Proposed Integer Linear Programming Model

To reduce the cost of resource utilization, the objective function of ILP excludes the tasks computed on the public cloud. Moreover, function $f(T_i^S, T_i^C)$ is formulated to represent payoff obtainable from using storage and computation resources. In our proposed optimization model, a choice of $f(\dots)$ is arbitrary but, in practice, the function should be monotonically non-decreasing with the resource requirements T_i^S and T_i^C . To consider cloud cluster hierarchy, the weighted average payoff is formulated as the objective function by assuming here for national convenience two cloud clusters $C_{I1}, C_{I2} \subset C_I$ in the private cloud. Here, we propose the following optimization:

$$\max_{x_{i,j}} \sum_{j \in C_{I1}} \sum_{i \in N} w f(T_i^S, T_i^C) x_{i,j} + \sum_{j \in C_{I2}} \sum_{i \in N} (1-w) f(T_i^S, T_i^C) x_{i,j} \quad (1)$$

$$\text{s.t. } \sum_{i \in N} T_i^S x_{i,j} \leq R_j^S, \quad \forall j \in C_{I1} \cup C_{I2} \cup C_O \quad (2)$$

$$\sum_{j \in C_{I1} \cup C_{I2} \cup C_O} x_{i,j} = 1, \quad \forall i \in N \quad (3)$$

$$x_{i,j} \in P_i, \quad \forall i \in N, \forall j \in C_{I1} \cup C_{I2} \cup C_O \quad (4)$$

$$\left(\frac{1}{R_{i,j}^N} + \frac{T_i^B}{R_{i,j}^N} \right) T_i^S x_{i,j} + \frac{1}{R_j^C} \sum_{i' \in N} T_{i'}^C x_{i',j} \leq T_i^T, \quad \forall i \in N, \forall j \in C_{I1} \cup C_{I2} \cup C_O \quad (5)$$

Payoff weight w in the objective function can be set to a high value close to 1 (say, 0.99) for prioritizing the scheduling of task into a given private cloud cluster C_{I1} over the other private cloud cluster C_{I2} . As a result, all demands need to fill in the cloud cluster C_{I1} before doing so in the others, meaning that cluster C_{I1} has the higher level of cloud scheduling preference to cluster C_{I2} .

Equation (2) is to ensure that the storage requirement for the input data of all tasks does not exceed the storage capacity R_j^S . Equation (3) is to confirm that all requesting tasks are completely allocated to be computed at K-workers. Equation (4) is referred to the task's granularity reflected on the set P_i . Finally, Equation (5), which is derived earlier in Section 2, is to limit the total task completion time in the whole cloud federation.

4. Experimental Peer-to-Peer Federated Cluster and Edge-Core Cluster Scenarios

Two different scenarios are experimented to reflect practical situations. In scenario 1, named peer-to-peer federated cluster scenario, the cloud clusters in the private cloud are similar in resource parameters to create peer-to-peer networking scenarios focusing on sharing resources between cloud clusters.

In scenario 2, named edge-core cluster scenario, the cloud clusters in the private cloud have a drastic difference in the number of resources. Cluster C_{I1} mimics the characteristic of small edge resource focusing on real-time processing data near the task source, while cluster C_{I2} imitates core resource that has enormous storage and computation resource.

For the resource of public cloud C_O , all parameters are assigned at infinite levels for both test scenarios to ensure that the overwhelmed demand from the private clouds can still be handled by the public cloud as the last resource.

To compare the test scenarios, we set the resource parameters in C_{I1} and C_{I2} in the sense of similarity in overall resources. For the storage resource, the overall capacity in the edge-core cluster scenario is equal to the peer-to-peer federated cluster scenario, while for the part of networking and computational resource, the resources in each scenario are fixed at the same geometric averages as shown in Table 2.

Table 2. Resource parameter settings for test scenarios.

Resource Parameter	Peer-to-Peer Federated Cluster Scenario		Edge-Core Cluster Scenario	
	C_{I1}	C_{I2}	C_{I1}	C_{I2}
R_i^S (GBits)	128	128	4	252
R_i^C (MIPS)	1000	1000	100	10,000
$R_{i,j}^N$ (GBits/s)	1	1	10	0.1
$R_{i,j}^N$ (GBits/s)	1	1	10	0.1

Furthermore, three case studies are reported for each scenario to evaluate the ILP model's performance by considering the effect of task storage requirement, task computation requirement, and task granularity. Only one task parameter is varied to study its effect in each case study, as shown in Table 3, while the other task requirements are kept constants by assigning a value for each parameter based on [8]. The proportion of task $x_{i,j}$, the task completion time t_i and the payoff from private cloud computing $\sum_{i \in N} \sum_{j \in C_I} f(T_i^S, T_i^C)$ are determined as the results. Round-robin (RR) scheduling is used as a comparable baseline to evaluate the proposed ILP model's performance since it is well-known for being a default task scheduler in Kubernetes [12].

Table 3. Task parameter settings for investigated case studies.

Task Parameter	Case Study 1	Case Study 2	Case Study 3
$f(\dots)$	$(T_i^S + T_i^C) \times 10^{-6}$	$(T_i^S + T_i^C) \times 10^{-6}$	$(T_i^S + T_i^C) \times 10^{-6}$
T_i^S (Mbits)	1 to 1,000,000	10	1000
T_i^C ($\times 10^6$ instructions)	150	10 to 10,000,000	100
T_i^T (s)	1	1	1
T_i^B	0.2	0.2	0.2
ϵ	100	100	1 to 1000

5. Results and Discussions

The experimental results, solved by MATLAB R2017b, from the case studies 1–3 in the test scenarios 1 and 2 are presented via line graphs showing the proportion of the task allocated to each cluster, the task completion time, and the amount of payoff from completing tasks. In terms of the time complexity of solving the optimization problem, a standard computer (2.9 GHz Dual-Core Intel Core i5 and 8 GB memory size) computes

each data point using only a few milliseconds, which is equivalent to the amount of time that the task scheduler needs to use for calculation before assigning sub-tasks to each cloud cluster. Therefore, this calculation time is negligible in comparison to the maximum allowable total task completion time. In practice, the effect of the task storage requirement, as shown in Figure 3, reflects the characteristic of big data analysis that becomes the typical tasks nowadays. In peer-to-peer federated cluster scenario as shown in Figure 3a, since the cluster C_{I1} in the ILP model is selected to be the highest level of cloud hierarchy, the task is allocated entirely to the cluster C_{I1} when the input data size of the task is considerably small. As the input data size grows, the task in the ILP model is allocated to the cluster C_{I2} , followed by public cloud C_O ; however, the highest proportion of task in the cluster C_{I2} is only at 0.38 units before dropping at the same number of that in the cluster C_{I1} . This is due to the fact that the two clusters in the private cloud are identical. When the two clusters reach their limitation, the amount of the task working on both of them should be equivalent. Then, at the extremely high level of input data size, the task is completely shifted to the public cloud since the task is too large for the entire private cloud.

Figure 3. Proportion of task in case study 1: (a) peer-to-peer federated cluster scenario; (b) edge-core cluster scenario.

Moving on to the edge-core cluster scenario, as shown in Figure 3b, the edge resource starts at 0.66 units while the core one is at 0.34 units. There is a difference in task proportion in each private cloud cluster due to the contrast of parameter settings. At the low input data size, the task is divided into two cloud clusters because the edge resource is fully used as the first level of hierarchy. However, when the data size increases, the task is significantly moved to the public cloud as same as in the peer-to-peer federated cluster scenario.

Considering the effect of the task computational requirement, varying the number of instructions in the peer-to-peer federated cluster scenario, as shown in Figure 4a, provides a nearly similar result with Figure 3a. The reason is that if the scenario is the peer-to-peer federated cluster, then the situation can be classified into two types. The first one is when the task is so tiny that can be entirely fit in only the private cloud. In this situation, cluster C_{I1} will always be received a higher amount of task than the cluster C_{I2} thanks to the different levels of cloud hierarchy. However, the other type of situation happens when the whole private cloud is insufficient for dealing with the enormous task. Therefore, an equal amount of task is undertaken at each cluster in the private cloud while the public cloud is used for the surplus demand.

Figure 4. Proportion of task in case study 2: (a) peer-to-peer federated cluster scenario; (b) edge-core cluster scenario.

For the edge-core cluster scenario as Figure 4b, the edge resource dominates at the tiny number of instructions of task as same as in Figure 3b. Nevertheless, when the number of instructions is moderate, the core resource becomes the highest instead. The explanation is that tasks can be real-time proceeded at the edge unless they are too complex. Otherwise, the core resource is used instead as the larger scale of resources but at the expense of higher transmission time. Finally, the public cloud is still needed when the task has a gigantic number of instructions.

Considering the ILP and RR model's performance in terms of task completion time as in Figure 5, when the task storage size or computation size is still small that the public cloud C_O is not needed, the ILP model provides a higher completion time than the RR model. Then, the task completion time from the RR model is continuously longer as the task size grows while the time from the ILP model remains just below 1 second, which corresponded to the maximum allowable completion time requirement. Moving on to the payoff as in Figure 6, the graph pattern in the peer-to-peer federated cluster scenario is nearly the same as the graph pattern of task completion time. In other words, the growth of payoff level received from the ILP model has a limit, while that from the RR model continuously grows as the task size increases. However, in the edge-core scenario as in Figure 5b,d and Figure 6b,d, the ILP model provides better performances than the RR model regarding both the task completion time and payoff when the storage size, as in Figures 5b and 6b, is between about 300 Mbits to 2 Gbits and when the computation size, as in Figures 5d and 6d, is between about 300 MIPS to 10,000 MIPS. This illustrates that the proposed ILP model can be a guideline for the cloud orchestration in IoTcloudserve@TEIN because of the lower task completion time and the higher payoff level especially upon the large demand growth, which is the major operation range of concerns in practice.

Figure 5. Task completion time: (a) peer-to-peer federated cluster scenario in case study 1; (b) edge-core cluster scenario in case study 1; (c) peer-to-peer federated cluster scenario in case study 2; (d) edge-core cluster scenario in case study 2.

Figure 6. Payoff obtained from cloud computing: (a) peer-to-peer federated cluster scenario in case study 1; (b) edge-core cluster scenario in case study 1; (c) peer-to-peer federated cluster scenario in case study 2; (d) edge-core cluster scenario in case study 2.

Concentrating on the effect of task granularity, the number of separable equally-sized units has been varied from 1, indivisible, to 1000, almost freely divisible. It is clearly seen in Figure 7a,b that the inseparable tasks are moved to the public cloud in both scenarios. As the task's granularity rises, the proportion of task allocated to the cluster C_{IL} sharply increases because the task can gradually fill in the remaining capacity in the cluster C_{IL} , meaning that the payoff grows productively. The effect of granularity can determine the comparison between VMs and container technology. In particular, VMs require more overall resources for running virtual operation systems while container technology is more lightweight. Thus, container technology is used for dividing tasks better and finer than VMs. Therefore, granularity is crucial for payoff improvement, reflecting the advantage of implementing container technology in cloud architecture.

Figure 7. Proportion of task in case study 3: (a) peer-to-peer federated cluster scenario; (b) edge-core cluster scenario.

6. Conclusions

This report proposes an ILP model that can provide optimal containerized task scheduling for different levels of cloud hierarchy by maximizing the weighted payoff from the private cloud. This model can be applied to different practical scenarios by adjusting both demand requirements and resource availability. Peer-to-peer networking and edge-core resource management are the two examples of scenarios that have been herein studied in the proposed model. The result shows that when the task is considerably small, either in storage or computation size, the cloud cluster with the higher hierarchy receives a higher proportion of task. As the task requirement increases, the two cloud clusters in the private cloud in the case of peer-to-peer federated cluster scenario tend to receive an equal proportion of the task, while the public cloud turns to be the dominant cloud. In the case of the edge-core cluster scenario, the core cluster can be the dominant cloud since there are higher scales of computation resources in the core cluster that can reduce the task completion time to be in the range of acceptable requirements.

Focusing on the model's performance by comparing the proposed model to the RR model, there is a tradeoff between obtaining a higher payoff level and completing the task in a shorter time. In particular, the ILP model provides a higher payoff level when the task is relatively small, while the lower task completion time can be seen when the task is comparatively enormous. However, the ILP model overcomes the RR model both in the payoff and completion time aspects when the computation size of the task is moderate in the case of the edge-core cluster scenario. Therefore, the ILP model can be applied to IoTcloudserve@TEIN as the better cloud orchestration, especially upon large demand growth, which is the major operation range of concerns in practice.

Moreover, the proposed model illustrates the significance of implementing cloud architecture that refined splittable task granularity via, i.e., light-weighted container technology. The improvement in the payoff from completing tasks is evidently found from the cloud platform with the ability to split demand finely since it can maximize the resource utilization by granularly fitting in the resource requirement.

For future research associated with this paper, a worthy direction would be an actual application of the model with different real-world structures of the number of tasks, cloud clusters, and the level of cloud hierarchy. Moreover, future research could focus on the time-series nature of demands instead of a one-snapshot task for generalized construction of the model.

Author Contributions: Conceptualization, N.S. and C.A.; data curation, N.S.; formal analysis, N.S. and C.A.; funding acquisition, C.A.; investigation, N.S. and C.A.; methodology, N.S. and C.A.; project administration, C.A.; resources, C.A.; software, N.S.; supervision, C.A.; validation, N.S. and C.A.; visualization, N.S.; writing—original draft, N.S.; writing—review & editing, C.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Asi@Connect’s Data-Centric IoT-Cloud Service Platform for Smart Communities (IoTcloudServe@TEIN) project with grant contract ACA 2016/376-562.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Buyya, R.; Yeo, C.S.; Venugopal, S. Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications, Dalian, China, 25–27 September 2008; pp. 5–13. [CrossRef]
2. IoTcloudServe@TEIN, November 2019. Available online: <https://www.facebook.com/iotcloudServe> (accessed on 23 May 2021).
3. TEIN3, April 2020. Available online: <https://www.tein3.net/Pages/Home.aspx> (accessed on 23 May 2021).
4. Merkel, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.* **2014**, *2014*, 2.
5. Hightower, K.; Burns, B.; Beda, J. *Kubernetes: Up and Running Dive into the Future of Infrastructure*, 1st ed.; O’Reilly Media, Inc.: Sebastopol, CA, USA, 2017.
6. OF@TEIN+, April 2020. Available online: <https://github.com/OFTEIN-NET/OFTEIN-Plus> (accessed on 23 May 2021).
7. Kurze, T.; Klems, M.; Bermbach, D.; Lenk, A.; Tai, S.; Kunze, M. Cloud Federation. In Proceedings of the Second International Conference on Cloud Computing, GRIDs, and Virtualization, Rome, Italy, 20–30 September 2011; pp. 32–38.
8. Thirasupa, R.; Saivichit, C.; Aswakul, C. Cloud Infrastructure Design Model for Green Smart City: Case Study of Electricity Generating Authority of Thailand. In *Information Science and Applications*; Springer: Singapore, 2020; pp. 135–147. [CrossRef]
9. Rekik, M.; Boukadi, K.; Assy, N.; Gaaloul, W.; Ben-Abdallah, H. A Linear Program for Optimal Configurable Business Processes Deployment into Cloud Federation. In Proceedings of the 2016 IEEE International Conference on Services Computing (SCC), San Francisco, CA, USA, 27 June–2 July 2016; pp. 34–41.
10. Tordsson, J.; Montero, R.S.; Moreno-Vozmediano, R.; Llorente, I.M. Cloud Brokering Mechanisms for Optimized Placement of Virtual Machines across Multiple Providers. *Future Gener. Comput. Syst.* **2012**, *28*, 358–367. [CrossRef]
11. Liu, J.; Mao, Y.; Zhang, J.; Letaief, K.B. Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems. In Proceedings of the IEEE International Symposium on Information Theory, Barcelona, Spain, 10–15 July 2016; pp. 1451–1455.
12. Guan, X.; Wan, X.; Choi, B.-Y.; Song, S.; Zhu, J. Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers. *IEEE Commun. Lett.* **2017**, *21*, 504–507. [CrossRef]
13. Ibrahim, H.; Aburukba, R.; El-Fakih, K. An Integer Linear Programming Model and Adaptive Genetic Algorithm Approach to Minimize Energy Consumption of Cloud Computing Data Centers. *Comput. Electr. Eng.* **2018**, *67*, 551–565. [CrossRef]