Cloud-SEnergy: A Bin-Packing Based Multi-Cloud Service Broker for Energy Efficient Composition and Execution of Data-intensive Applications

Thar Baker[1], Bandar Aldawsari[1], Muhammad Asim[2], Hissam Tawfik[3]
Zakaria Maamar[4] and Rajkumar Buyya[5]
[1]Department of Computer Science, Liverpool John Moores University, UK
[2]Department of Computer Science, National University of Computer and Emerging Sciences, Pakistan
[3]School of Computing, and Creative Technologies, Leeds Beckett University, UK
[4]College of Technological Innovation, Zayed University, UAE
[5]School of Computing and Information Systems, The University of Melbourne, Australia
t.baker@ljmu.ac.uk; B.M.Aldawsari@2012.ljmu.ac.uk;
muhammad.asim@nu.edu.pk; H.Tawfik@leedsbeckett.ac.uk;
zakaria.maamar@zu.ac.ae; rbuyya@unimelb.edu.au

Abstract

The over-reliance of today's world on Information and Communication Technologies (ICT) has led to an exponential increase in data production, network traffic, and energy consumption. To mitigate the ecological impact of this increase on the environment, a major challenge that this paper tackles is how to best select the most energy efficient services from cross-continental competing cloud-based datacenters. This selection is addressed by our Cloud-SEnergy, a system that uses a bin-packing technique to generate the most efficient service composition plans. Experiments were conducted to compare Cloud-SEnergy's efficiency with 5 established techniques in multi-cloud environments (All clouds, Base cloud, Smart cloud, COM2, and DC-Cloud). The results gained from the experiments demonstrate a superior performance of Cloud-SEnergy which ranged from an average energy consumption reduction of 4.3% when compared to Based Cloud technique, to an average reduction of 43.3% when compared to All Clouds technique. Furthermore, the percentage reduction in the number of examined services achieved by Cloud-SEnergy ranged from 50% when compared to Smart Cloud and average of 82.4% when compared to Base Cloud. In term of run-time, Cloud-SEnergy resulted in average reduction which ranged from 8.5% when compared to DC-Cloud, to 28.2% run-time reduction when compared to All Clouds.

Keywords: Multi-cloud; Bin-packing; Service composition; Energy efficiency; Data-intensive application

1. Introduction

It is largely accepted in the ICT community that cloud computing is one of the technologies of choice when deploying Web services business-applications. In addition to elasticity and pay-per-use appealing features to ICT practitioners, cloud computing revolves around a simple provisioning model: providers offer services[1] to users, users search for the services they need, and high-speed networks ensure the connection between users and providers when invoking services. This provision model is straightforward when a single provider/cloud has all the necessary Web services (services for short) that business applications require. However, this is not always the case because of the diversity and complexity of today's applications; *one size fits all* does not hold. Thus, service composition becomes crucial to alleviate these diversity and complexity. Composition refers to decomposing a user's request according to the clouds hosting the necessary services, invoking concurrently and/or sequentially these services, and finally collecting the partial results prior to their combination and presentation to the user.

This scenario increases in complexity as the number of providers increases. This is typically manifested in service composition built upon a broker-based cloud service model [1] that necessitates the collaboration among a number of cloud service providers, which formulates what is so-called multi-cloud environment, whether explicitly or implicitly to yield the service outcomes and end results to the user. We assume that Web services from separate businesses coordinate their activities in such a way that any conflict with service clouds, such as sharable resources, order dependencies, or communication delays, is avoided [1–4]. The resulting service may be directly used by a service user or be recursively incorporated in further service compositions [5, 6]. Today's service composition approaches in a multi-cloud environment overlook the impact of cloud locations (i.e., datacenters) and resource consumption (e.g., bandwidth and CPU) when identifying the necessary services. In fact, they prioritize non-functional requirements (aka Quality-of-Service (QoS)) such as service cost, response time, and availability regardless of how much carbon footprint will be generated due to increase in data traffic and processing power [2].

The growing demand of Web services and the increase in service providers and datacenters to offer and host these Web services across every geographical region have led to significant increases in network traffic and the associated energy consumption of the extensive infrastructure (e.g., extra servers and switches) required to respond quickly and effectively to user requests. Moreover, transferring data between datacenters and between datacenters and users can consume even larger amounts of energy compared to just process-

[1] In this paper, we focus on Web services as other forms of services exist in clouds, namely infrastructure and platform

ing and storing the data on the datacenters themselves [7–9]. In addition, higher bandwidth and network speed required to cope with the cloud network traffic and to speed up data transformation process generate higher carbon footprint [2]. This has become a major concern to different bodies (e.g., governments and environmental NGOs) in term of meeting environmental requirements such as those published by the 2011 report of PBL Netherlands Environmental Assessment Agency and JRC European Commission [10] and also reducing the energy consumption [11].

Effective ways of reducing cloud computing energy-consumption are relatively under-explored and require further research and development to be fully achieved. Several approaches exist in the literature aim at selecting appropriate service components to optimize the overall quality of the composition according to a set of pre-defined QoS metrics. Energy efficiency, however, tends to get relatively less attention when it comes to the optimization priorities for service composition aiming to achieve the most efficient service components. There exist several approaches in the literature [12–14] consolidated the multi-dimensional bin-packing problem for allocating and minimizing migrating workloads to achieve energy optimal operations. This paper presents and evaluates a novel bin-packing based energy efficient service broker, named Cloud-SEnergy, using Integer Linear Programming (ILP) [15]. The main novelty of our approach lies in that Cloud-SEnergy searches for and integrates the least possible number of most energy efficient services, from the least possible number of service providers. The proposed service composition is based on the multidimensional bin-packing problem [16]; where items of possibly different capacities' need to fit into bins in a way that the total number of bins used, is minimized. An efficient green ICT strategy should address 3 concerns:

1. Amount of energy consumed by the datacenters' equipment, such as Computer Room Air Conditioning (CRAC) unit.
2. Amount of energy consumed when transporting data between users and datacenters.
3. Amount of energy consumed when processing service composition over datacenters.

In contrast to existing approaches that address concerns 1 and 2, as is detailed in Section 3, this paper addresses concern 3 by presenting a novel service composition approach/broker in a multi-cloud environment. The new broker acts as an intermediary bridge between the user and subscribed datacenters. It finds the most energy efficient services that match the user needs, from the least possible number of cloud services' providers in a multi-cloud environment; while meeting the user's needs. The main objective is to pack items of different capacities (i.e., finding energy efficient services) into a minimum number of bins (i.e., from the effective number of datacenters) characterised by their total power consumption. To accomplish this

aim, we first formalise the datacenter-broker communication model, which shows the kind of data that each datacenter is required to send to the broker to facilitate how the broker algorithm finds the most energy efficient composition at a later step, using ILP. An improved service composition algorithm that exhibits better performance compared to 5 competing algorithms is presented and evaluated.

The rest of the paper is organised as follows. The next section presents the problem addressed. Related work is summarised in Section 3 and the proposed broker model is discussed in Section 4. A detailed discussion on how the broker works and its time complexity is presented in Section 5; and the evaluation is detailed in Section 6. Finally, the paper draws some conclusions and paves the way for future work in Section 7.

2. Problem statement

In a traditional multi-cloud environment scenario, a user submits a request to a service broker stating the specifications of the required services. The broker should then find the appropriate service providers that satisfy the request. Currently, locating the best-fit service that matches the user needs and broker aims is considered to be the most challenging task for a multi-cloud broker for the following reasons:

1. Energy efficient cloud services/resources searching and allocation involves identifying and assigning resources to each incoming user request in such a way, that the user needs are met with the least possible number of resources allocated from the least possible number of cloud service providers per request. There have already been vast amount of research work in the area of cloud service discovery and composition, along with techniques and tools that are powerful enough for cloud service consumers to rely on. However, the attention was paid to the Infrastructure-as-a-Service (IaaS) layer and virtualisation types [17–20], as opposed to energy efficient service searching, allocation, and provision.
2. A range of heuristic solutions for IaaS were proposed [21–23] but there is still a lack of powerful algorithms that would ensure energy efficient resource allocation. New hybrid cloud solutions that combine IaaS and Platform-as-a-Service (PaaS) (e.g., openstack Heat), from a single cloud, are evolving over time and being considered more attractive since they enable the joint deployment of infrastructure and platforms. However, these solutions tend to overlook the need for energy efficient resource allocation and little attention is paid to address the problem at this level.
3. Finding the most energy efficient service that satisfies a user needs is currently under investigated,

including how a broker can assist in locating multiple services to serve the request when there is no one service that can match the request.

4. There is a need for efficient algorithms for comparing the energy required to compute and execute the various services that can fulfil the user request on different clouds, in order to choose the most energy efficient one, and creating a composition plan from the least possible number of clouds to ensure that the energy efficiency target is met.

As the number of cloud providers and services increase, the composition of many services from different providers becomes more complicated in a real multi-cloud environment. This would require a massive amount of data interchange among all service participants and will consequently lead to high levels of energy consumption [2]. Brokers and service providers tend to priorities QoS metrics, such as service security [24], availability, response time, as these factors attract clients. The communication cost, and sending and receiving data among the composite Web services from different cloud providers can be expensive and time and energy consuming. Finding the required services from the least possible number of cloud service providers is as important as finding the services themselves. However, what continues to be a challenging and an under-investigated issue is to find the most energy efficient service composition plan, which should have the least possible number of composite services from an effective number of cloud service providers, that fulfils the user request.

3. Related Work

Energy efficiency has been an important research topic that attracted significant interest well before the emergence of cloud systems where the focus was on saving energy in computing appliances by for instance, extending their battery lifetimes [25, 26], and developing energy efficient hardware. Many of these energy saving techniques were initially adopted for cloud systems. However, what makes cloud more challenging is the huge number of servers that reside in datacenters and services that need to be managed, and the fact that multi-cloud datacenters need to support clients' on-demand requests for services at any time and anywhere. Several strategies have been developed to effectively reduce server power consumption for cloud systems [27–29]; however, energy efficiency of service composition is still in its infancy and requires further attention from all stakeholders.

Wajid *et al.* in [30] extend the approach of Lecue and Mehandjiev [31] and analyse its performance for service composition optimization and its application in cloud computing to streamline resource usage

that in turn contributes towards energy efficiency. The composition is optimized based on functional and non-functional criteria to determine a set of cloud services representing energy efficient deployment configuration. The work of Bartalos and Blake in [32] discusses the impact of power consumption of Web services on creating a green cloud computing infrastructure. However, these approaches [30–32] focus on the power consumption of a Web service when it is migrated from a physical server to another to avoid situations that lead to server overload or service under-utiuselisation [33]. Bartalos *et al.* [34] introduce a decision support procedure to provide a deterministic understanding of power consumption of modular software assets or services that reside in the hardware devices/servers. The procedure relies on power estimation models that predict power consumption of a software service considering the type of server upon which it resides. Guo *et al.* [35] proposed a joint inter- and intra-datacenter workload management scheme, Joint ElectriciTy price-aware and cooling efficiency-aware load balancing (JET), to cut and reduce the electricity cost of geographically distributed datacenters. Chen *et al.* [36] present StressCloud, a tool for profiling the performance and energy consumption of cloud systems. They profile the energy consumption of cloud-based application under various task workloads and resource allocation strategies. The procedures of [34, 36] are proposed for a single cloud environment; hence do not tap into opportunities of multiple cloud collaboration [37]. In contrast, we aim at examining energy efficient service composition in multi-cloud environment.

Five different algorithms currently exist in the literature (All Cloud [38], Base Cloud [38], Smart Cloud [38], COM2 [39], DC-Cloud [37]), which examine how service composition can be created by efficiently utilizing multiple clouds. Section 6 presents a comparative evaluation of our proposed algorithm with these algorithms. The All Clouds considers all clouds as inputs for the composition and determines all possible solutions. The algorithm locates a service composition sequence with a least execution time, but does not minimize the number of clouds nor the energy consumption in the final composition. The Base Cloud algorithm recursively enumerates all cloud combination possibilities in increasing order until an optimal solution is identified. It begins by analyzing all singleton sets of clouds and stop searching if the required combination can be found utilizing a single cloud. Otherwise, it extends its search to cloud sets of size 2, then 3 until the required combination is found. It generates an optimal composition solution with a small number of clouds, despite of the energy consumed by the selected clouds. The Smart Cloud algorithm locates a near optimal composition plan based on approximating a multiple cloud environment as a tree and then identifies a minimum demand set from searching the tree. The Smart Cloud locates a sub-optimal solution at a reduced cost while using a reduced cloud set. Heba kurdi *et al.* [39] propose a novel combinatorial optimization algorithm (COM2) that considers multiple clouds and performs service composition with a short execution

time and minimal number of clouds, thereby reducing communication energy and costs. DC-Cloud algorithm is proposed as a theoretical model for service composition in the multi-cloud environment with focus on minimizing service composition overhead. The overhead is measured through two fundamental metrics: (i) average number of clouds involved in service composition and (ii) average number of service files examined.

Luo *et el.* [40] propose a technique to select a composite service by using the path with the best QoS and lowest cost. The technique is based on the Dijkstra search, which assumes that QoS attributes, such as duration and throughput, are additive. However, Hang in [41] suggests that the additive attributes depend on the nature of the composite service. For example, when constituent services of a composite service are invoked in parallel then the overall duration is not the addition of the durations of the constituent services.

The construction of an optimal QoS-aware service composition often leads to inefficient compositions with redundant services. According to Rodriguez-Mier *et al.* in [42], the main drawbacks of the above-mentioned service composition approaches are their low performance and the lack of emphasis on reducing data sharing overhead. When minimizing the number of composite services in a multi-cloud environment is not a priority, the number of selected services and the input/output interaction among them can become high. The number of services involved in a composition has a direct impact on a number of QoA measures. For example, the work of Rodriguez-Mier *et al.* in [12] suggest that minimising the number of services in a composition will help in minimising the total response time and maximising the throughput. Wang *et al.* [13] propose a greedy algorithm to minimise the number of required service composition during a persistent query's life-time such that the routing update cost and transmission cost are minimised.

Several works in the literature observe the similarity between Virtual Machines(VM) placement problem in a single datacenter and the well-known bin-packing problem, in which items of given capacity must be packed into a minimum number of bins. The survey of Wolke *et al.* [14] discusses on the usefulness of bin packing for dynamic resource allocation in cloud datacenters. Song *et al.* [43] propose a practical bin packing resource allocation algorithm that uses virtualization technology to allocate datacenter resources dynamically and support green computing by optimizing the number of servers actively used. Cloud management tools such as OpenStack [44] and Eucalyptus [45] are commonly used in many IaaS cloud environments for resource allocation, and utilize bin packing heuristics for placing incoming VMs on servers [14]. In [46], Liu and Baskiyar develop a heuristics approach, which integrates the classical bin packing algorithm to address the problem of scheduling independent tasks in computational grid with different priorities and deadline constraints.

To conclude, multiple works exist in the literature consolidating the multi-dimensional bin-packing problem for allocating and minimizing migrating workloads to achieve energy optimal operations [21, 43, 47]. However, there currently exist no previous work, which uses the bin packing approach to optimize the resulting composition in a multi-cloud environment. Given the large number of cloud resources available from multiple clouds, we prioritize energy efficiency by adopting a bin packing approach for searching and integrating the least possible number of services, from the least possible number of service providers.

4. The System Model

In order to formulate the problem and proposed solution in this paper, we need to identify the main stakeholders reported in Figure 1: user, broker and cloud service providers. The next subsections formalise the interrelationship among those stakeholders and show how the new broker works.

4.1. Formal datacenter-broker model

In a multi-cloud environment, the requested services may come from different commercial cloud providers. These services can be integrated and used together via mutual communication protocols to satisfy a complex service request. The Multiple Cloud service Providers (MCP) is a set of cloud providers, such that $MCP = \{CP_i, CP_{i+1}, ..., CP_h\}$ where $(1 \leq i \leq h)$ represents a CP identifier. Since energy required for service computation is a significant factor in our proposed model, services' providers are required to send to the broker the Total Energy Consumption (TEC) of all atomic services available at their datacenters. As such, each service provider will be described by the proposed broker using a 2-tuple format $\langle CP_{h,TEC} \rangle$. For illustration purposes, $\langle CP_{3,174} \rangle$ denotes the total power consumption of 174KW by all services available at cloud provider 3. In addition, we use $\pi_j(CP_{h,TEC_j})$ to denote the pre-defined composition plan (j), that is created by the cloud provider‘(h) with a total energy of TEC.

The proposed broker is based on the notion of Bin-Packing including a valid condition in the form of a constraint. The role of the broker is to pack items (finding services in our case) into a least set of bins (from minimum number datacenters) characterized by their total power consumption. Alternatively, one can think about it as to pack the user request of services into an effective number of datacenters. To this end, we define CP_i as a key decision variable for each cloud provider i that is set to 1 if cloud provider i is selected to provide a service, or 0 otherwise. The objective function used to find all requested services from a least

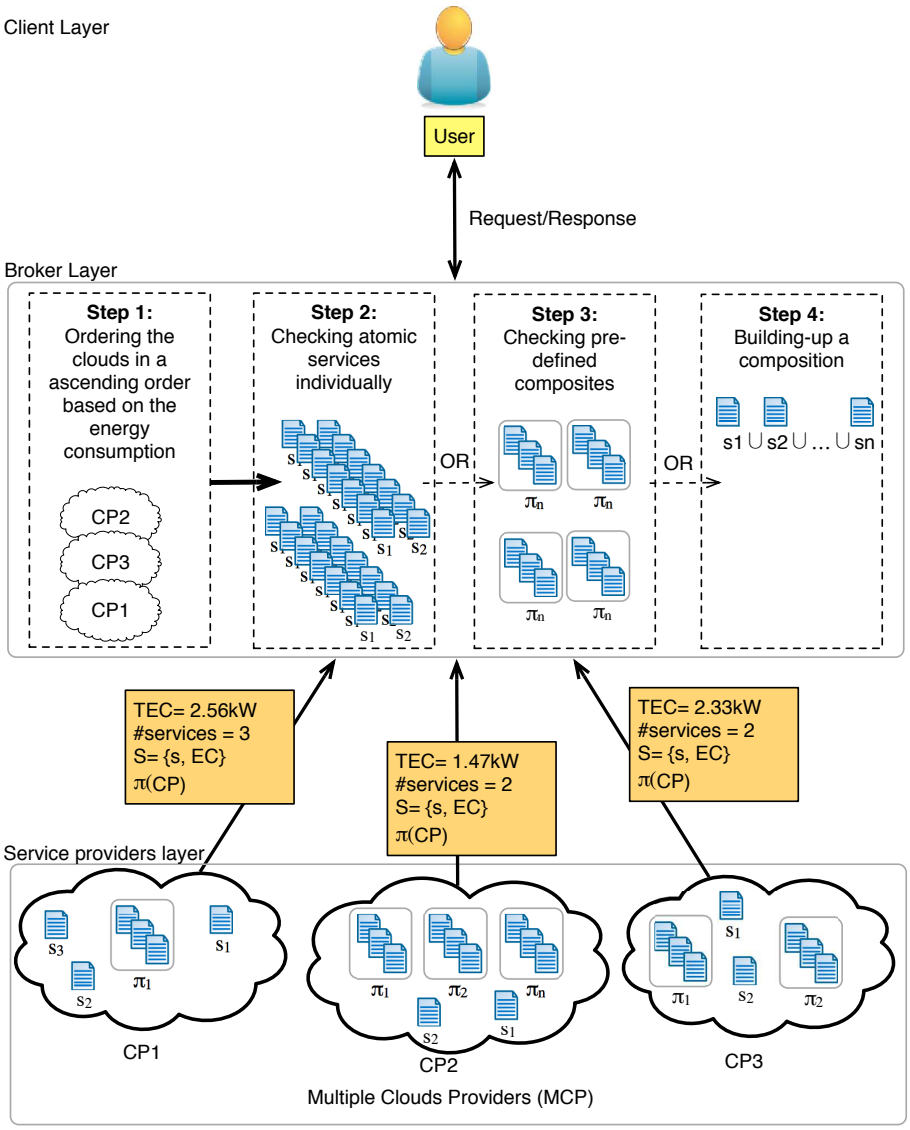

Figure 1: Conceptual representation of the proposed approach

possible number of cloud providers is expressed in (equation 1) as follows:

$$\min \mathrm{I} = \sum_{i=1}^{h} CP_i \tag{1}$$

where

$$CP_i = \begin{cases} 1, \text{ if the cloud provider i is used;} \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

Each cloud service provider offers a set of services S, where $S(CP_i) = \{s_k, s_{k+1}, s_m\}$ where $(1 \leq k \leq m)$; k is the identifier of each of the m atomic services of CP_i. In this sense, in order to satisfy the user request with a least possible number of selected services (equation 2), we use another decision variable s_k as in (equation 3), which is set to 1 when the service has been selected, otherwise 0 as in (equation 4).

$$\min \text{K} = \sum_{k=1}^{m} s_k \, |\{\forall s : s \in S \in CP_i \in MCP\} \tag{3}$$

where

$$s_k = \begin{cases} 1, \text{ if the service k is used;} \\ 0, \text{ otherwise.} \end{cases} \tag{4}$$

To reach the objectives of this work, the following assumptions are made:

1. Each service provider provides information regarding the total amount of energy consumed through the computation of all atomic services' computation at the designated datacenter (TEC), the number of atomic services, the actual pre-defined composition plans $\pi(CP)$, and a list of atomic services in the form of $\langle s_i, EC \rangle$, where EC is the energy consumption of service s_i.
2. In Step 1, the broker lists the cloud providers in an ascending order based on the least total energy consumed ($lTEC$) by all atomic services at the cloud service provider, such that the cloud provider that consumes less energy will be examined first.
3. In Step 2, the broker examins the atomic services of all ascending-ordered providers first to search for a one that matches the user request, from one provider.
4. Still in Step 2, if many atomic single services are found that match the user request, the broker will compare their energy consumption in order to choose the atomic service that consumes less power.
5. In Step 3, if none is found to match the user request, the broker will examine the pre-defined composition plans of all ascending-ordered providers to search for a one that matches the user request from one provider.
6. Otherwise, in Step 4, the broker will create a composition plan such that
   $\pi_B = \{\langle s_i, EC(s_i), CP_p \rangle \cup \langle s_j, EC(s_j), CP_q \rangle \cup, ..., \cup \langle s_k, EC(s_k), CP_r \rangle\}$ is a set of services from either

the same provider or different providers, subject to the following (equation 5):

$$\text{minCons}|\sum_{i=1}^{k}\{EC(s_i)\}\ |\{\forall s : s \in S\ \in CP_r \in MCP\} \tag{5}$$

Figure 1 depicts how the 4 steps feature the Broker Layer interact with each other. It can be seen that the first step starts to sort the subscribed-to-broker cloud providers in an ascending order based on the total energy consumption. To bin-pack the coming request to a single cloud service provider, the broker will start checking the atomic services suitability, within each cloud service provider individually, and then responds back to the user if a match is found. If not, the broker moves to Step 3 to check the pre-defined compositions by each provider separately to search for a "*ready*" composition from a single provider. If none of the previous steps have found a matching service or composition, the broker moves to Step 4 to build up an effective composition plan.

4.2. Service composition plan

As per Section 4.1, the best service, or a composition plan, must be one that satisfies the three decision variables listed in Equations 1, 2, and 3. Hence, the best one will be either an atomic service, a pre-defined composition plan by one of the subscribed cloud providers, or a set of atomic services from the same or different cloud providers that guarantee the least possible number of cloud providers involved with the least amount of energy required to compute each service selected.

To further illustrate this, consider a multi-cloud environment where a broker deals with 4 cloud providers $\{CP_1, CP_2, CP_3, CP_4\}$. Each of these providers provides a set of atomic services, which is a subset of the services $\{a, b, c, d, e\}$, and a set of $\pi(CP)$, as shown in Table 1.

Table 1: Multiple-cloud providers and services

Cloud providers	$\langle \text{CP}_{4,2.601} \rangle$	$\langle \text{CP}_{1,2.35} \rangle$	$\langle \text{CP}_{2,1.04} \rangle$	$\langle \text{CP}_{3,1.65} \rangle$
Atomic services	a, b, c, e	a, b, c	c, d, e	c, d
EC (kW)	0.52, 0.8, 0.721, 0.56	0.65, 0.5, 1.2	0.72, 0.32	1.2, 0.45
TEC (kW)	2.601	2.35	1.04	1.65
π**(CP)**	$\{a, e\}$, $\{b, c, e\}$,$\{c, e\}$, $\{b, e\}$	$\{a, b\}$, $\{a, c\}$, $\{b, c\}$	$\{d, e\}$	$\{c, d\}$

Upon receiving the USer Request (USR), the broker starts re-listing the cloud providers in an ascending order using Algorithm 1 to produce Table 2. It then examines all atomic services listed in row 2 in a bid

to find the most energy efficient one, from all providers, that matches the user request. It starts first with the service provider that would consume the least total energy, which is $CP_{2,1.04}$ in the case of Table 2. For example, if the user requests service c, then service c from CP_2 will be chosen given that it consumes less energy than any service c from any other service provider.

Table 2: Multiple-cloud providers and services sorted by energy consumption

Cloud providers	$\langle \text{CP}_{2,1.04} \rangle$	$\langle \text{CP}_{3,1.65} \rangle$	$\langle \text{CP}_{1,2.35} \rangle$	$\langle \text{CP}_{4,2.601} \rangle$
Atomic services	c, d, e	c, d	a, b, c	a, b, c, e
EC (kW)	0.72, 0.32	1.2, 0.45	0.65, 0.5, 1.2	0.52, 0.8, 0.721, 0.56
TEC (kW)	1.04	1.65	2.35	2.601
π**(CP)**	$\{d, e\}$	$\{c, d\}$	$\{a, b\}$, $\{a, c\}$, $\{b, c\}$	$\{a, e\}$, $\{b, c, e\}$,$\{c, e\}$, $\{b, e\}$

In the case where no match is found, the predefined composition plans of the 4 subscribed service providers will be tested next. For example, if the user asks for services $\{b, c, e\}$, then the providers will be checked in the following order: CP_2, CP_3, CP_1, CP_4. The broker will start checking all compositions of the cloud providers to find out if there is a pre-defined composition plan that satisfies USR, and where more than one possible match is found, it selects the least energy consuming one. Hence, given that CP_4 has the requested composition plan already defined, this arrives at the least possible number of providers involved in the composition, in combination with the least energy consumption.

5. Algorithmic Design of Cloud-SEnergy

The new broker deals with the user request via 4 main algorithms; which are based on the multi-cloud environment, subscribed providers, available services, and the pre-defined composition plans, as follows:

(i) Step 1: in this step, Algorithm 1 orders the cloud service providers in an ascending order according to their total energy consumption.

(ii) Step 2: in which Algorithm 2 checks the individual atomic services and energy consumption of each to find the best possible match.

(iii) Step 3: if Algorithm 2 is unable to find a suitable service(s), Algorithm 3 checks the pre-defined composition plans in each cloud provider and the energy consumption of each composition plan to find the best possible match.

(iv) Step 4: otherwise, Algorithm 4 creates an effective composition plan using the most energy efficient services from the least possible number of providers.

5.1. Time complexity

Although the proposed Cloud-SEnergy algorithm can find the most energy efficient service composition, its time complexity is exponential in the number of cloud providers, atomic services, and predefined composition plans as it may need to enumerate all possible cloud providers and their services in the worst case. Therefore, there are three significant dimensions involved in determining the performance of Cloud-SEnergy: (i) the number of available cloud providers, (ii) the number of available atomic services, and (iii) the number of the predefined service composition plans. In other words, the computational complexity increases exponentially with the number of providers and their services, as follows:

(i) The time complexity of Algorithm 1 is $O(n^2)$ time; this sort all cloud providers in an ascending order, where n is the number of cloud providers (nCP in Algorithm 1). Hence, the more cloud providers, the more time requires to sort them ascendingly.

(ii) Algorithm 2 takes $O(n.m^{'})$ time to search for all available $m^{'}$ atomic services in all n cloud providers, such that n=nCP and $m^{'}$ is the number of atomic services (j in Algorithm 2).

(iii) Algorithm 3 consumes $O(n.m^{''})$ time to search all available $m^{''}$ predefined composition plans in all n cloud providers, where $m^{''}$ has been represented as π_B in Algorithm 3.

(iv) It represents the worst case scenario as it necessitates exhausting the previous steps before it starts. Algorithm 4 requires $O(n.m^{'})$ time to search for all available atomic services in all cloud providers in order to build up a new effective service composition.

As a result, the performance of Cloud-SEnergy is dominated by Algorithm 4.

Algorithm 1: Sorting Clouds in an Ascending Order

input : number of cloud providers (nCP), Total Energy Consumption {TEC(CP_i) | $\forall i : 0 < i \leq$nCP }
output : an ascending ordered list of cloud providers

```
1  Get (nCP, TEC(CP_i) | ∀i : 0 < i ≤ nCP)
2  foreach i = 1 to i ≤ nCP − 1 step 1 do
3      lTEC ← i
4      foreach j = i + 1 to j ≤ nCP do
5          if EC(j) < EC(i) then
6              lTEC ← j
7          end
8      end
9      Temp ← EC(I)
10     EC(I) ← EC(lTEC)
11     EC(lTEC) ← Temp
12 end
13
```

Algorithm 2: Check Atomic Services

input : user service request (USR), number of Cloud Providers (nCP)
output : most energy efficient service from most "possible" energy efficient data centre ($Ser_k(CP_{i,lTEC})$), actual energy consumption of the selected service (minCons)

```
1  Get (USR, nCP)
2  foreach i = 1 to i ≤ nCP step 1 do
3      Select (CP_{i,lTEC})
4      Get #(Ser(CP_{i,lTEC}))
5      j ← #(Ser(CP_{i,lTEC}))
6      foreach k = 1 to k ≤ j step 1 do
7          if ((Ser_k(CP_{i,lTEC}) ∩ USR) == Ø) then
8              go to 26
9          else
10             if (k==1) then
11                 minCons ← EC(Ser_k(CP_{i,lTEC}))
12                 return Ser_k(CP_{i,lTEC}), minCons
13                 go to 6
14             else
15                 if (EC(Ser_k(CP_{i,lTEC})) < minCons) then
16                     minCons ← EC(Ser_k(CP_{i,lTEC}))
17                     return Ser_k(CP_{i,lTEC}), minCons
18                     go to 26
19                 else
20                     go to 26
21                 end
22             end
23             return Ser_k(CP_{i,lTEC}), minCons
24             go to 6
25         end
26         if (k==j) then
27             go to 2
28         else
29             go to 6
30         end
31     end
32     if (i==nCP) then
33         go to 37
34     else
35         go to 2
36     end
37 end
```

Algorithm 3: Check Predefined Composition Plans

```
   input  : user service request USR, multiple cloud providers MCP, largest number of
            composition plan m
   output : Effective Composition Plan π_B
 1 USR ← Ø, π_B ← NULL, minCons ← NULL, m ← largest number of composition plan;
 2 Get (USR, nCP)
 3 Select (CP_m)
 4 i ← m
 5 if (i is True) then
 6     foreach j = 1 to j ≤ i step 1 do
 7         if ((π_j(CP_i) ∩ USR) == Ø) then
 8             if (j = i) then
 9                 go to 35
10             else
11                 go to 6
12             end
13         else
14             if (j = 1) then
15                 minCons ← EC(π_j(CP_i))
16             else
17                 if (EC(π_j(CP_i)) < minCons) then
18                     minCons ← EC(π_j(CP_i))
19                 else
20                     go to 6
21                 end
22             end
23         end
24     end
25     return minCons
26     if (i = m) then
27         π_B ← minCons
28     else
29         if (minCons < π_B) then
30             π_B ← minCons
31         end
32     end
33     return π_B                                   ▷ Optimal composition plan
34 end
35 i=i-1
36 if (i ≥ 1) then
37     Select (CP_i)
38     go to 5
39 else
40     Invoke Algorithm 4
41 end
42
```

Algorithm 4: Creating an Effective Services Composition from Multiple Providers

input : user service request (USR), number of multiple cloud providers (nCP)
output : most energy efficient service from most "possible" energy efficient datacenter ($Ser_k(CP_{i,lTEC})$), actual energy consumption of the selected service (minCons)

```
1  serList← Ø; cldList ← Ø; minCons ← Ø; totalCons ← Ø
2  Get (USR, nCP)
3  foreach i = 1 to i ≤ nCP step 1 do
4      Select (CP_{i,lTEC})
5      Get #(Ser(CP_{i,lTEC}))
6      j ← #(Ser(CP_{i,lTEC}))
7      foreach k = 1 to k ≤ j step 1 do
8          if ((Ser_k(CP_{i,lTEC}) ∩ USR) − serList == Ø) then
9              if ((Ser_k(CP_{i,lTEC}) ∩ USR) ∈ serList == true) then
10                 if (EC((Ser_k(CP_{i,lTEC}) ∩ USR) − serList) < EC(Ser_k(CP_{i,lTEC}) ∈ serList)) then
11                     Swap
12                     minCons ← EC((Ser_k(CP_{i,lTEC}) ∩ USR) − serList)
13                     totalCons ← totalCons + minCons
14                     go to 7
15                 else
16                     go to 7
17                 end
18             else
19                 go to 7
20             end
21         else
22             minCons ← EC((Ser_k(CP_{i,lTEC}) ∩ USR) − serList)
23             serList ← serList ∪ ((Ser_k(CP_{i,lTEC}) ∩ USR) − serList)
24             totalCons ← totalCons + minCons
25             go to 7
26         end
27     end
28     if (i < nCP) then
29         i = i + 1
30         go to 4
31     else
32         go to end
33     end
34 end
```

6. Performance Evaluation

This section discusses the settings of the experiments we performed and then analyses the results.

6.1. Experimental settings

To evaluate the performance and potential efficiency gains of our broker model, it is important to benchmark the results against well established models and quantify the percentage of consumption that can be expected when running Cloud-SEnergy algorithm. Five different algorithms for selecting the cloud services combination were adopted for our comparative evaluation purposes: (All Clouds [38], Base Cloud [38], Smart Cloud [38], COM2 [39], and DC-Cloud [37]). We use identical simulation parameters of the 5 algorithms in order to enable a systematic and consistent evaluation. The experimental data were based on the default Web service test-set provided in the OWL-S XPlan package [48]. A dedicated simulator is developed to conduct the performance assessments and the comparison, using Java EE 8 as the programming language to implement the proposed algorithm on IBM ILOG CPLEX Optimization Linear Solver [49] as the simulator environment. The experiments were run on an Apple iMac (Retina 5K display, 3.2GHz Intel Core i5, and 8GB 1867MHz DDR3). To evaluate the effectiveness of Cloud-SEnergy, we simulated 4 cloud providers $\{CP_1, CP_2, CP_3, CP_4\}$. Each of these providers provides a set of pre-defined composition plans, which are subsets of $\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$, and are based on the MCPs environment as shown in Table 3.

Table 3: Cloud providers composition set per MCP

MCPs	**CP**$_1$	**CP**$_2$	**CP**$_3$	**CP**$_4$
MCP$_1$	π_1, π_2, π_3	π_4, π_5	π_3, π_4	$\pi_1, \pi_2, \pi_3, \pi_5$
MCP$_2$	π_1, π_2	π_3	π_2, π_5	π_1, π_4, π_5
MCP$_3$	π_1, π_3, π_5	π_5	π_1, π_2	π_3, π_4
MCP$_4$	π_2, π_3, π_5	π_3, π_4	π_1, π_2, π_3	π_4, π_5
MCP$_5$	π_1, π_2	π_2, π_3	π_3	π_1, π_4, π_5

In addition, $\{2, 3, 8, 3, 3\}$ in Table 4, represents the number of services involved in each of the aforementioned composition plans respectively.

Table 4: Number of services per compositions

Composition plan	π_1	π_2	π_3	π_4	π_5
Number of services	2	3	8	3	3

The cloud providers are listed in an ascending order based on the Total Energy Consumption (TEC) by the available set of services, as per Algorithm 1 (Table 5.a). As such, this order will be different in each MCP, for the same provider as shown in Table 5.b. For example, CP_4 comes first in MCP1 as the most energy efficient one, and last in MCP4 as the least energy efficient one.

Table 5: CPs and energy consumption per MCPs

(a) Before sorting the CPs ascendingly

MCPs	$\mathbf{CP}_1$	$\mathbf{CP}_2$	$\mathbf{CP}_3$	$\mathbf{CP}_4$
$\mathbf{MCP}_1$	$\langle CP_{1,2.7} \rangle$	$\langle CP_{2,3.1} \rangle$	$\langle CP_{3,3.5} \rangle$	$\langle CP_{4,2.2} \rangle$
$\mathbf{MCP}_2$	$\langle CP_{1,1.9} \rangle$	$\langle CP_{2,2.9} \rangle$	$\langle CP_{3,2.6} \rangle$	$\langle CP_{4,1.7} \rangle$
$\mathbf{MCP}_3$	$\langle CP_{1,2.2} \rangle$	$\langle CP_{2,2.8} \rangle$	$\langle CP_{3,2.4} \rangle$	$\langle CP_{4,2.5} \rangle$
$\mathbf{MCP}_4$	$\langle CP_{1,2.5} \rangle$	$\langle CP_{2,2.8} \rangle$	$\langle CP_{3,2.7} \rangle$	$\langle CP_{4,3.7} \rangle$
$\mathbf{MCP}_5$	$\langle CP_{1,2.4} \rangle$	$\langle CP_{2,2.8} \rangle$	$\langle CP_{3,3.5} \rangle$	$\langle CP_{4,2.2} \rangle$

(b) After sorting the CPs ascendingly

MCPs	Sorting order of CPs			
$\mathbf{MCP}_1$	$\langle CP_{4,2.2} \rangle$	$\langle CP_{1,2.7} \rangle$	$\langle CP_{2,3.1} \rangle$	$\langle CP_{3,3.5} \rangle$
$\mathbf{MCP}_2$	$\langle CP_{4,1.7} \rangle$	$\langle CP_{1,1.9} \rangle$	$\langle CP_{3,2.6} \rangle$	$\langle CP_{2,2.9} \rangle$
$\mathbf{MCP}_3$	$\langle CP_{1,2.2} \rangle$	$\langle CP_{3,2.4} \rangle$	$\langle CP_{4,2.5} \rangle$	$\langle CP_{2,2.8} \rangle$
$\mathbf{MCP}_4$	$\langle CP_{1,2.5} \rangle$	$\langle CP_{3,2.7} \rangle$	$\langle CP_{2,2.8} \rangle$	$\langle CP_{4,3.7} \rangle$
$\mathbf{MCP}_5$	$\langle CP_{4,2.2} \rangle$	$\langle CP_{1,2.4} \rangle$	$\langle CP_{2,2.8} \rangle$	$\langle CP_{3,3.5} \rangle$

6.2. Experimental results

The results for the 5 benchmark algorithms, the All Clouds in Table 6.a, the Base Cloud in Table 6.b, the Smart Cloud in Table 6.c, COM2 in Table 6.d and DC-Cloud in Table 6.e are all consistent with previously published results in [37–39]. We list the results of evaluating the new broker in Table 6.f such that it can be compared against the aforementioned approaches.

In this paper, we follow the same evaluation methodology and adopt the same cloud simulation environment. In the first experiment, we evaluate two performance measures, which are:

- The number of cloud providers that are involved in the final composition $|CP|$, and
- The number of services checked before reaching into the final composition $|S|$.

Table 6 indicates that Cloud-SEnergy algorithm produced performance improvement compared to other algorithms in maintaining a low number of examined services and composite clouds. The number of services examined $|S|$ did not exceed 38, and the number of combined clouds was as low as 2 clouds and never

exceeded 3 in the worst in MCP_2. In addition, the total number of services and clouds examined by our broker were the smallest among all other approaches, 152 and 11 respectively, which has a direct impact on the time spent to find the final composition.

Table 6: CPs and number of (π) composition plans per MCPs

(a) All Clouds algorithm

Performance	**CP involved**	**|CP|**	**|S|**
MCP_1	CP_1CP_2 CP_4	3	46
MCP_2	CP_1CP_2 CP_3CP_4	4	27
MCP_3	CP_1CP_3 CP_4	3	32
MCP_4	CP_1CP_2 CP_3CP_4	4	44
MCP_5	CP_1CP_2 CP_3CP_4	4	32
Total		18	181

(b) Based Cloud algorithm

Performance	**CP involved**	**|CP|**	**|S|**
MCP_1	CP_1CP_2	2	65
MCP_2	CP_1CP_2 CP_4	3	148
MCP_3	CP_3CP_4	2	128
MCP_4	CP_2CP_3	2	68
MCP_5	CP_2CP_4	2	112
Total		11	521

(c) Smart Cloud algorithm

Performance	**CP involved**	**|CP|**	**|S|**
MCP_1	CP_1CP_3	2	70
MCP_2	CP_1CP_2 CP_4	3	48
MCP_3	CP_3CP_4	2	48
MCP_4	CP_2CP_3	2	140
MCP_5	CP_1CP_2 CP_4	3	56
Total		12	362

(d) COM2 algorithm

Performance	**CP involved**	**|CP|**	**|S|**
MCP_1	CP_4CP_2	2	35
MCP_2	CP_4CP_2 CP_3	3	45
MCP_3	CP_1CP_4 CP_3	3	50
MCP_4	CP_1CP_3 CP_2	3	49
MCP_5	CP_2CP_4	2	30
Total		14	209

(e) DC-CLoud algorithm

Performance	**CP involved**	**|CP|**	**|S|**
MCP_1	CP_4CP_2	2	46
MCP_2	CP_4CP_2 CP_3	3	27
MCP_3	CP_1CP_4	2	29
MCP_4	CP_1CP_4	2	44
MCP_5	CP_2CP_3 CP_4	3	32
Total		12	178

(f) Cloud-SEnergy algorithm

Performance	**CP involved**	**|CP|**	**|S|**
MCP_1	CP_4CP_2	2	35
MCP_2	CP_4CP_1 CP_2	3	26
MCP_3	CP_1CP_3	2	29
MCP_4	CP_1CP_3	2	38
MCP_5	CP_1CP_3	2	24
Total		11	152

Figure 2 shows a comparison analysis of the % reduction in the number of examined atomic services (relative to the baseline case for each MCP) by Cloud-SEnergy, All Clouds, Base Cloud, Smart Cloud, and COM2. The baseline for each MCP refers to the case of the maximum number of atomic services examined for that particular MCP, which is then used as the reference point for calculating the percentage reduction (relative improvement). Figure 3 shows a comparison of the average number of examined atomic services in Cloud-SEnergy across all MCPs, compared to the average number of examined services of all other algorithms. This reduction in number of examined services (Cloud-SEnergy is lowest at 30.4) impacts positively on the time needed and energy consumed to find the suitable service(s), as will be further discussed

in the following experiment.

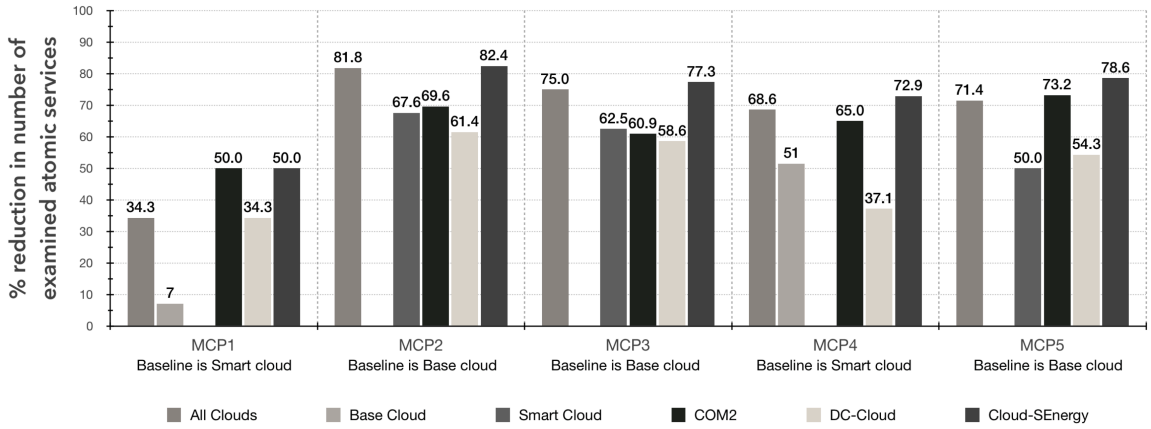

Figure 2: % reduction in the number of examined atomic services.

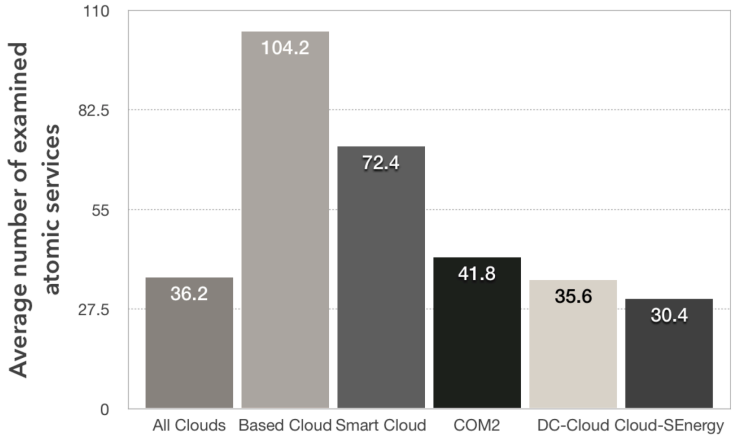

Figure 3: The average number of examined services.

To further validate Cloud-SEnergy algorithm's time and efficiency gains, the second experiment measured the running time and energy consumption of the 5 algorithms to find the requested composition, and to compare the results gained in terms of time and energy consumption when Cloud-SEnergy is used to find the same composition. Figure 4.a shows the performance of Cloud-SEnergy algorithm compared to all other algorithms in term of the actual running time, measured in seconds, for running the algorithm until an appropriate composition is reached. It can be seen that Cloud-SEnergy requires less running time to find

the requested services, across all *MCPs*, compared to the 5 benchmark algorithms. This is due to the fact that Cloud-SEnergy only checks services that match the user request and ignores the others as per lines 7, 7, and 8 of algorithms 2, 3 and 4, respectively. Hence, the number of tested services (Table 6.f) and run time (Figure 4.a) to find the appropriate ones are lower. The average run time improvement of Cloud-SEnergy compared to the 5 benchmarks run time, for all *MCPs*, are listed in Table 7. For example, Cloud-SEnergy spends 28.2% less time to find the requested services when compared to All Clouds algorithm. Figure 4.b also depicts that Cloud-SEnergy algorithm consumes less energy to find the requested services compared to the rest. These results are obtained based on a pre-sorted list of clouds in an ascending order of total energy consumption, as per Step 1/Algorithm 1. The average energy consumption improvement of Cloud-SEnergy over other algorithms are listed in Table 7. For instance, there is 43.3% average reduction in energy consumption when finding requested services, across the 5 used *MCPs*, using Cloud-SEnergy than All Clouds, and so on so forth. Table 7 demonstrates the significant efficiency gains of Cloud-SEnergy compared to the other 5 approaches in terms of both runtime and energy efficiency.

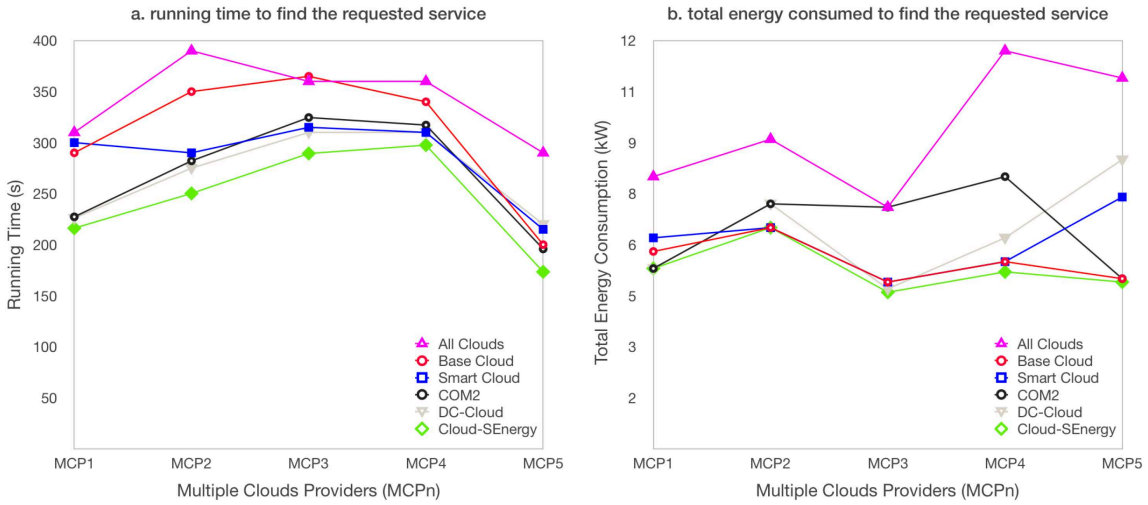

Figure 4: Running time and energy consumption for finding the requested service

Table 7: Percentage performance improvement of Cloud-SEnergy over other algorithms

	% Run-time reduction	% Energy consumption reduction
All Clouds	28.2	43.3
Based Cloud	20.6	4.3
Smart Cloud	14.2	13.1
COM2	8.9	18.7
DC-Cloud	8.5	16.9

7. Conclusion and Future Work

A novel multi-cloud service computing approach (Cloud-SEnergy) is presented, focusing on the selection of energy-efficient services and service composition plans that meet user requirements. Our Bin-packing-based service composition approach determines the least possible number of composite services based on an effective combination of cloud services' providers that satisfy the user request. Our approach addresses the increasing need for optimising energy consumption associated with the rise in complex real-world cloud-based service and user request scenarios, which are characterised by a large number of cloud providers and services. Our findings were evaluated against 5 established service computing algorithms for multiple cloud environments and the simulation results demonstrated that Cloud-SEnergy produces significant relative performance improvements in terms of both running time and energy consumption.

Future extensions to this work include the potential incorporation of other energy efficiency relevant factors, such as the power consumption of switches and links in datacenters, in our objective optimisation function. Other future research avenues include evaluating our approach using other complex service selection scenarios and assessing its applicability and performance in key application domains such as smart cities, mobile commerce, and smart government.

References

[1] B. Aldawsari, T. Baker, D. England, Towards a holistic brokerage system for multi-cloud environment, in: 10th IEEE International Conference on Internet Technology and Secured Transactions (ICITST), IEEE, 2015, pp. 249–255.

[2] T. Baker, B. Al-Dawsari, H. Tawfik, Y. Ngoko, Greedi: An energy efficient routing algorithm for big data on cloud, Ad Hoc Networks 35 (2015) 83–96.

[3] Z. Maamar, N. Nanjangud, P. Thiran, Towards a coordination model for web services, in: Technologies for Collaborative Business Process Management, 2006.

[4] B. Benatallah, Q. Sheng, M. Dumas, The self-serv environment for web services composition, IEEE Internet Computing 7 (2003) 40–48.

[5] D. L.-J. Muhammad Asim, B. Lempereur, B. Zhou, Q. Shi, M. Merabti, Event driven monitoring of composite services, in: In the International Conference on Social Computing (SocialCom), IEEE, 2013, pp. 550–557.

[6] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, X. Xu, Web services composition: A decade's overview., in: Information Sciences, 2014, pp. 218–238.

[7] F. Owusu, C. Pattinson., The current state of understanding of the energy efficiency of cloud computing., in: In IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 2012, pp. 1948–1953.

[8] S. K. Garg, C. S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers., Journal of Parallel and Distributed Computing 71 (6) (2011) 732–749.

[9] Y. Sverdlik. Hong kong, china's data center gateway to the world. [online, cited 16 March 2018].

[10] J. Olivier, G. J. Maenhou, M. Muntean, J. Peters. Trend in global co2 emissions: Pbl netherlands environmental assessment agency and jrc european commission report [online] (2016).

[11] P. Baer. Exploring the 2020 global emissions mitigation gap [online] (2008) [cited 16 March 2018].

[12] P. Rodrguez-Mier, M. Mucientes, J. C. Vidal, M. Lama, An optimal and complete algorithm for automatic web service composition, International Journal of Web Service Research 9 (2) (2012) 1–20.

[13] X. Wang, J. Wang, Z. Zheng, Y. Xu, M. Yang, Service composition in service-oriented wireless sensor networks with persistent queries, in: 2009 6th IEEE Consumer Communications and Networking Conference, IEEE, 2009, pp. 1–5.

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

[15] J. de Souza, M. J. Rider, J. R. S. Mantovani, Planning of distribution systems using mixed-integer linear programming models considering network reliability, Journal of Control, Automation and Electrical Systems 26 (2) (2015) 170–179.

[16] Christensen, Henrik, Arindam, Pokutta, Approximation and online algorithms for multidimensional bin packing: A survey, Computer Science Review 24 (2017) 63–79.

[17] S. Kuma, R. Buyya, Green Cloud Computing and Environmental Sustainability, John Wiley and Sons, Ltd, 2012, pp. 315–339. doi:10.1002/9781118305393.ch16.
URL http://dx.doi.org/10.1002/9781118305393.ch16

[18] A. L. Lemos, F. Daniel, B. Benatallah, Web service composition: a survey of techniques and tools., ACM Computing Surveys (CSUR) 48 (3) (2016) 33.

[19] W. Lin, C. Zhu, J. Li, B. Liu, H. Lian, Novel algorithms and equivalence optimization for resource allocation in cloud computing, International Journal of Web and Grid Services 11 (2) (2015) 193–210.

[20] W. Lin, S. Xu, L. He, J. Li, Multi-resource scheduling and power simulation for cloud computing, Information Sciences 397-398 (2017) 168–186.

[21] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems 28 (5) (2012) 755–768.

[22] J. Wang, C. Huang, K. He, X. Wang, X. Chen, K. Qin, An energy-aware resource allocation heuristics for vm scheduling in cloud, in: 10th IEEE International Conference on High Performance Computing and Communications IEEE International Conference on Embedded and Ubiquitous Computing (HPCCEUC),, IEEE, 2013, pp. 1–7.

[23] D. Kumar, B. Sahoo, B. Mondal, T. Mandal, A genetic algorithmic approach for energy efficient task consolidation in cloud computing, International Journal of Computer Applications 118 (2) (2015) 1–6.

[24] B. Gupta, D. P. Agrawal, S. Yamaguchi, Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security, Springer, 2016.

[25] B. Hamdaoui, T. Alshammari, M. Guizani, Exploiting 4g mobile user cooperation for energy conservation: challenges and opportunities, IEEE Wireless Communications 20 (5) (2013) 62–67.

[26] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, T. Magedanz, Ease: Epc as a service to ease mobile core network deployment over cloud, IEEE Network 29 (2) (2015) 78–88.

[27] Y. C. Lee, A. Y. Zomaya, Energy efficient utilization of resources in cloud computing systems, The Journal of Supercomputing 60 (2) (2012) 268–280.

[28] L. Luo, W. Wu, D. Di, F. Zhang, Y. Yan, Y. Mao, A resource scheduling algorithm of cloud computing based on energy efficient optimization methods, in: Proceedings of the 2012 International Green Computing Conference (IGCC '12), IEEE Computer Society, pp. 1–6.

[29] A. Uchechukwu, K. Li, Y. Shen, Improving cloud computing energy efficiency, in proceedings of the cloud, in: Proceedings of the Cloud Computing Congress (APCloudCC), IEEE Asia, 2012, pp. 53–58.

[30] U. Wajid, C. A. Marín, A. Karageorgos., Optimizing energy efficiency in the cloud using service composition and runtime adaptation techniques., in: In IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2013, pp. 115–120.

[31] L. F, M. N., Seeking quality of web service composition in a semantic dimension, IEEE Transactions on Knowledge and Data Engineering (TKDE) 23 (6) (2010) 942–959.

[32] P. Bartalos, M. B. Blake, Engineering energy-aware web services toward dynamically-green computing, in: Proceedings of the Service-Oriented Computing-ICSOC 2011 Workshops, Springer, 2012, pp. 87–96.

[33] E. Park, H. Shin, Reconfigurable service composition and categorization for power-aware mobile computing, Parallel and Distributed Systems 19 (11) (2008) 1553–1564.

[34] P. Bartalos, Y. Wei, M. B. Blake, H. Damgacioglu, I. Saleh, N. Celik, Modeling energy-aware web services and application., Journal of Network and Computer Applications 67 (2016) 86–98.

[35] Z. Guo, Z. Duan, J. Xu, Electricity cost-aware dynamic workload management in geographically distributed datacenters, Computer Communications 50 (2014) 162–174.

[36] F. Chen, J. Grundy, J.-G. Schneider, Y. Yang, Q. He, Automated analysis of performance and energy consumption for cloud applications., in: In Proceedings of the 5th ACM/SPEC international conference on Performance engineering, ACM, 2014, pp. 39–50.

[37] J. Lu, Y. Hao, L. Wang, M. Zheng, Towards efficient service composition in multi-cloud environment, in: IEEE International Conference on Computational Science and Computational Intelligence, IEEE, 2015, pp. 1–6.

[38] G. Zou, Y. Chen, Y. Xiang, R. Huang, Y. Xu, Ai planning and combinatorial optimization for web service composition in cloud computing, in: Annual International Conference on Cloud Computing and Virtualization (CCV 2010), 2010, pp. 1–8.

[39] H. Kurdi, A. Al-Anazi, C. Campbell, A. A. Faries, A combinatorial optimization algorithm for multiple cloud service composition, Computers and Electrical Engineering 42 (C) (2015) 107–113.

[40] J.-Z. Luo, J.-Y. Zhou, Z.-A. Wu, An adaptive algorithm for qos-aware service composition in grid environments., Service Oriented Computing and Applications 3 (3) (2009) 217–226.

[41] C.-W. Hang, A. K. Kalia, M. P. Singh, Behind the curtain: Service selection via trust in composite services, in: In IEEE 19th International Conference on Web Services (ICWS), IEEE, 2012, pp. 9–16.

[42] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, An integrated semantic web service discovery and composition framework, IEEE Transaction on Services Computing 9 (4) (2016) 537–550.

[43] W. Song, Z. Xiao, Q. Chen, H. Luo, Adaptive resource provisioning for the cloud using online bin packing., IEEE Transactions on Computers 63 (11) (2014) 2647–2660.

[44] OpenStack. [link].
URL https://www.openstack.org/

[45] Eucalyptus. [link].
URL https://www.eucalyptus.com

[46] C. Liu, S. Baskiyar, Scheduling mixed tasks with deadlines in grids using bin packing, in: 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS'08), 2008, pp. 229–236.

[47] S. Srikantaiah, A. Kansal, F. Zhao., Energy aware consolidation for cloud computing., in: In Proceedings of the 2008 conference on Power aware computing and systems, 2008, pp. 1–5.

[48] R. Karunamurthy, F. Khendek, R. H. Glitho, A novel architecture for web service composition, Journal of Network and Computer Applications 35 (2011) 787–802.

[49] IBM. Optimization model development toolkit for mathematical and constraint programming [online] (2017) [cited 20 March 2018].