Performance Evaluation 79 (2014) 104–119

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Reducing the expenses of geo-distributed data centers with portable containerized modules☆

Marco Brocanelli *, Wenli Zheng, Xiaorui Wang

Power-Aware Computer Systems (PACS) Laboratory, Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, 43210, USA

ARTICLE INFO

Article history:
Available online 10 July 2014

Keywords:
Containerized data centers
Traditional data centers
Container transportation
Workload dispatch
OpEx
CapEx

ABSTRACT

In order to maximize their profits, big IT companies need to reduce the operating expenses (OpEx) and capital expenses (CapEx) of their geo-distributed data centers. To reduce OpEx, recent studies have proposed algorithms that dynamically dispatch workload among different data center sites, to exploit the variability in local electricity price and green energy availability. However, the amount of OpEx savings achieved by workload dispatching is often limited by the local computing capacity. To reduce CapEx, data centers try to minimize the needed increases in size based on estimated workload growth. However, workload overestimation would result in a high CapEx, while underestimation would hurt business increases.

In this paper, we propose ExContainer, a novel strategy that leverages portable containerized modules to minimize both OpEx and CapEx for geo-distributed data centers. With ExContainer, containers are allocated monthly to different sites and workload is dispatched hourly, to minimize total OpEx. To cut CapEx, ExContainer finds the best trade-off between the degree of capacity expansion and the number of needed new containers to handle the workload peak. We evaluate ExContainer with real-world traces. Our experimental results show that compared to state-of-the-art solutions, ExContainer allows significant reduction in both OpEx and CapEx.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

In the era of cloud computing, big IT companies are rapidly expanding their data centers in order to meet the fast-increasing demands for computing. In order to achieve the maximized profits, those companies need to minimize the operating expenses (OpEx) and capital expenses (CapEx) of their geo-distributed data centers.

Dynamic workload dispatching has been recently proposed as an effective way to reduce data center OpEx. For example, it is well-known that different geographical locations may have different and time-varying electricity prices, so Internet requests can be dynamically distributed to different data center sites to take advantage of the lower prices in certain locations [1,2]. Likewise, some data centers may have on-site renewable energy generators (e.g., solar or wind farms) that can provide low-cost green energy. As a result, requests can be dynamically dispatched based on green energy availability as well for reduced OpEx [3,4]. Similarly, requests can also be dispatched based on the ambient temperatures of different data centers, because the ambient temperature can significantly impact data center costs if advanced cooling techniques such as

☆ This work was supported, in part, by the US NSF under grants CNS-1143607 (CAREER Award) and CCF-1331712.
* Corresponding author.
E-mail addresses: brocanelli.1@osu.edu (M. Brocanelli), zheng.691@osu.edu (W. Zheng), wang.3596@osu.edu (X. Wang).

http://dx.doi.org/10.1016/j.peva.2014.07.007
0166-5316/© 2014 Elsevier B.V. All rights reserved.

free cooling [5] are used. Since cooling power accounts for 30%–40% of the total power consumption of a data center [6], the OpEx savings on cooling can also be significant.

However, the amount of OpEx savings that can be achieved by workload dispatching is often limited by the size of each data center site. This is because a data center cannot handle more workload if it already reaches the peak computing capacity (i.e., all its servers are 100% utilized), regardless of its low electricity price, abundant green energy, or cold weather. Simply increasing the servers hosted in every data center may lead to a significantly increased CapEx as we discuss later. In addition, different data centers are commonly configured to provide different services (e.g., search, email, video streaming). Hence, workload can be dispatched only to some data centers that provide the required services and have available capacities, which further limits the benefits of workload dispatching. Since different data centers rarely have their workload peaks at the same time, portable containerized modules [7–9] can be exploited to flexibly move servers among different sites monthly based on electricity price, green energy availability, and ambient temperatures, such that hourly workload dispatching can achieve a further reduced OpEx.

In order to reduce CapEx, data centers try to minimize the needed increases in computing, cooling, and power delivery capacities based on estimated workload growth over the lifetime of the data center infrastructure (about 15 years [6,10]). However, it is well-known that workload growth can be highly unpredictable with significant fluctuations. As a result, workload overestimation would result in over-provisioning of data center capacities and thus a high CapEx [10], while underestimation would hurt business increases due to insufficient capacities. Therefore, if the size of a data center can be increased incrementally with container modules to handle short-term workload peaks, the data center can wait for the demand to grow to a certain level that best justifies an upgrade of the data center power and cooling capacities. Hence, expensive upgrades can be deferred, resulting in reduced CapEx.

Recently, big IT companies such as Microsoft [11], eBay [12], and Google [13] have started to adopt portable containerized modules, which are designed mainly to provide additional computing capacity for disaster recovery, sport events, or military applications. In sharp contrast to traditional (room-based) data centers, containerized modules can be quickly built (typically within 1–6 months), shipped with trucks, and deployed with power and network in a few hours [9]. IT equipment and cooling system can be placed within one module (all-in-one, e.g., HP POD 240a, Huawei IDS1000) or within two different modules (e.g., IBM PMDC). Moreover, the cooling system is usually built with an economizer mode (i.e., free cooling), which uses outdoor air (when the temperature is lower than a certain threshold) to cool down servers. As a result, containers can generally achieve a lower Power Usage Effectiveness (*PUE*, as low as 1.05 [8]).

In this paper, we propose ExContainer, a novel strategy that leverages portable containerized modules to minimize both OpEx[1] and CapEx for geo-distributed data centers. Although containerized modules have already been used in industry, to our best knowledge, they have not yet been used to fully exploit the diversity and variability in electricity price, green energy availability, and ambient temperature across different geographical locations for minimized OpEx. Neither have they been used to optimize data center CapEx by handling short-term workload spikes and deferring the expensive upgrades of data center infrastructure. ExContainer is composed of an OpEx optimizer and a CapEx optimizer. The OpEx optimizer features a two-level approach, which allocates containers to different sites on a coarse time scale (e.g., one month) and conducts dynamic workload dispatching and server turning on/off on a fine time scale (e.g., every hour), for the minimization of electricity and network delay (from dispatcher to data center) cost. The CapEx optimizer uses containers to handle short-term workload peaks that would exceed the capacity of a data center. When the demand grows to a level that justifies an upgrade of the data center capacities, the CapEx optimizer finds the best trade-off between the degree of capacity expansion and the number of needed new containers, to minimize the total CapEx. Specifically, this paper makes the following major contributions:

- While existing work focuses mainly on workload dispatching for reduced OpEx, we identify its two limitations: limited capacity of each site and service heterogeneity. Accordingly, we propose to leverage portable containerized modules for further reduced OpEx.
- While current practice commonly uses containers in a stationery way as modules and only moves containers for special events, we propose to move them periodically for minimized OpEx, based on the electricity prices, green energy availability, and ambient temperatures.
- We propose a new CapEx minimization approach to defer the expensive upgrades of data center infrastructure by finding the best trade-off between the degree of capacity expansion and the number of needed new containers.
- We formulate three constrained optimization problems to minimize OpEx by (1) allocating containers monthly to different locations, (2) conducting hourly dynamic workload dispatching and server turning on/off, and to minimize CapEx. Our results show that compared to state-of-the-art solutions [1,2,14,6,10], ExContainer allows significant reduction in both OpEx and CapEx.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the overview of ExContainer. Sections 4 and 5 describe the detailed designs of the OpEx and CapEx optimizers, respectively. Section 6 evaluates ExContainer with real-world traces. Section 7 discusses the implications of ExContanainer and Section 8 concludes the paper.

[1] We focus on electricity bill in this paper, though OpEx can include maintenance, software license, salaries, etc.

2. Related work

Numerous approaches for workload dispatching and servers turning on/off have been explored in literature. [1,14–16], focus on developing algorithms to maximize the usage of green energy or to balance the load to achieve energy savings for distributed data centers. [2,4,17–19] minimize the electricity bill by exploiting the hourly fluctuations of electricity price or capping a monthly cost budget. Server room temperature is considered in [20] while the impact of outside temperature to a single data center is studied in [3]. [21] considers the variability of temperature across different sites to calculate the total costs of building and managing a single data center. Different from these studies, we exploit containers to move resources to existing data center locations where workload can be processed at a lower cost. We exploit local ambient temperatures of distributed data centers to achieve lower OpEx and we address a more real and general case, i.e., requests for a certain service can be dispatched only to data centers that provide such a service.

A few research studies attempt to minimize CapEx due to data center expansion. [21,22] provide two different algorithms to localize the best site to build a data center and also to provide a consolidation plan to transform the enterprise IT infrastructure. [6,23,10,24] provide a CapEx model for traditional data centers based on workload estimation and minimize CapEx by deferring the installation of servers according to workload variation. [25,26] study how to minimize CapEx for the usage of local green energy or by using UPS batteries, respectively. Different from previous studies, we use containerized modules to defer the upgrade of non-scalable elements such as power plant, cooling system and server room for minimized CapEx.

Research studies have recently been conducted about containerized data centers. [27,7,28] study the design of power and network architectures of container data centers. [29,30] compare the costs of traditional and containerized data centers. [31,32] study the cooling system performance of containers and provide mathematical models. Different from previous studies, we propose to exploit containers to lower hourly workload processing costs and to defer the expansion of traditional data centers. To the best of our knowledge, our paper is the first one considering at same time traditional and containerized data centers for minimized OpEx and CapEx.

3. Overview of ExContainer

In this work, we consider a big IT company that has a network of data centers located in different places. For example, Google owns seven data centers in the US and 13 in total globally. Depending on the IT company, each data center site may exploit green energy by (1) having on-site renewable (e.g., wind or solar) power generators that can provide low-cost renewable energy, or (2) purchasing renewable energy from off-site. In this work, we assume the first case even though ExContainer can be easily extended to adapt to the second case, by including a cost for the purchase of renewable energy in the cost model. Each data center can also do free air cooling [5], i.e., letting outside cold air in to cool down servers when the ambient temperature is lower than a certain threshold. Different data centers may host the same or different services (e.g., search, email, streaming) as explained in [34]. Depending on the actual data center configuration, each service can share servers with other services (using virtualization) or have a dedicated set of servers. For example search service usually has dedicated servers given the high demand of computational needs, while other services for batch workload or streaming may share the same servers. In this work, we assume that different services do not share servers, even though our cost model can be extended to consider a shared environment. A centralized request dispatcher dynamically receives user requests and dispatches each request to a certain data center for processing.

The objective of ExContainer is to utilize portable container modules for reduced CapEx and OpEx for this big IT company. ExContainer features a three-layer strategy, with the first layer for CapEx reduction every three years and the other two layers (monthly and hourly, respectively) for OpEx reduction. In the first layer, in order to minimize CapEx, ExContainer determines the number of containers needed to handle short-term demand peaks every three years (i.e., lifetime of a container [30]) until the cost of upgrading the traditional data center is justified by the amount of expected workload. After the number of containers is determined by the CapEx optimizer, the OpEx optimizer moves the containers among different data center sites in a monthly manner to take better advantage of their different and time-varying power prices, renewable availability, and ambient temperatures. As a result, in the third layer, the hourly dynamic request dispatching can lead to further reduced OpEx.

3.1. OpEx optimizer overview

The OpEx optimizer exploits the portability of containers to process more requests in the site with the minimum workload-processing costs, which depend on four factors: power prices, renewable availability, ambient temperatures, and dispatching delay. As shown in Fig. 2, the electricity prices in New York City (NYC), Atlanta (ATL), and Chicago (CHI) [33] vary significantly in a year. For example, NYC has the lowest price in April, but the highest in December. Therefore, more workload should be processed in NYC in April and less in December. However, as discussed before, the amount of requests that can be dispatched to a data center site is limited by the computing capacity of the data center.

Fig. 1 illustrates the design of the OpEx optimizer, which has a *local information collector*, a *container allocator*, and a *workload dispatcher*. In every month, depending on the workload variation of each service, the container allocator moves every available container to a selected data center. As a result, the computing capacities of those data centers with containers

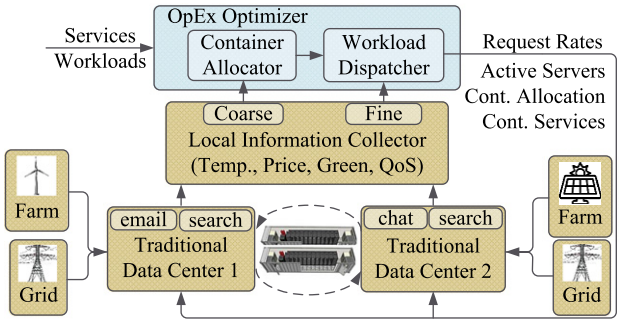

Fig. 1. The OpEx optimizer works at different time-scales to exploit containers and save OpEx.

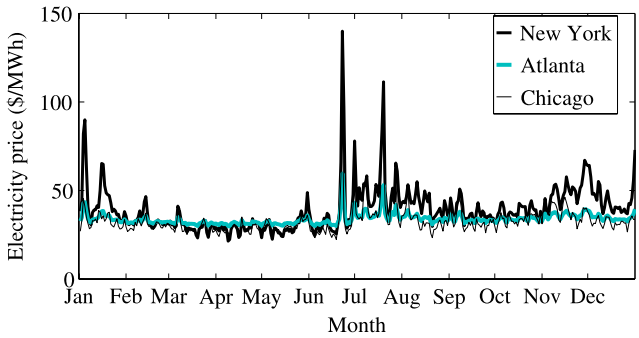

Fig. 2. Real-world variation of daily average electricity price in three cities of USA [33].

moved to are increased. For example, if we move containers to NYC in April, a lower OpEx can be achieved for the IT company by having more servers to take advantage of NYC's lower power price. As a result, the IT company can overcome the two limitations of workload dispatching: limited capacity of each site and service heterogeneity. The one-month period is chosen for the consideration of container transportation time (several days) and deployment time (a few hours [9]). While the container allocator is the main novelty of this paper, the workload dispatcher differs also from existing dispatching solutions (e.g., [2,15,16]) for two reasons. First, while previous work considers only power prices or renewable availability, we consider a new important factor, ambient temperature, which significantly affects the cooling costs that account for 30%–40% of the OpEx [6] (note that temperature is considered in [3,21], but not for workload dispatching among distributed data centers). Second, while previous work assumes that requests can be dispatched to any data center site, we address a more real and general case, i.e., requests for a certain service can be dispatched only to data centers that provide such a service.

3.2. CapEx optimizer overview

The CapEx optimizer exploits containers to defer the expensive upgrades of the data center server room, power, and cooling infrastructures. Traditional (room-based) data centers commonly take about two years to expand such infrastructures by building new server rooms, power facilities and cooling systems [24], when their computing demand is projected to exceed the capacity. Such expansion has to be planned for data center lifetime and thus can lead to a low capacity utilization in the first several years after the expansion and so a high CapEx. In contrast, a newer-generation container-based data center commonly expands by adding new containers (e.g., Microsoft modular data center [11]), but containers commonly cost more than room-based data centers due to their shorter lifetime (3 years vs. 15 years). The novelty of the proposed CapEx optimizer lies in a hybrid solution that determines the optimal time and scale for a data center to conduct infrastructure expansion depending on the workload prediction. Before that time point, ExContainer relies on containers to handle short-term workload peaks that would exceed the capacity. Adding containers to an existing data center is simplified by the fact that often, dedicated power substations are over-sized, thus able to supply some extra power.

Fig. 3 illustrates how the amount of Internet requests grows based on the real-world data (from passive network monitor of San Jose) from 2011 to 2013 [35] and the estimated 18% annual growth [36]. In the figure, we show an example to explain how containers can help a traditional data center on adapting to growing but fluctuating demand. The maximum computing capacity becomes inadequate in 2016. If the data center is expanded immediately, the expansion should be planned for the lifetime of the infrastructures (about 15 years [6]). Thus, a large portion of power and cooling capacity would be wasted in the first several years after 2016. With ExContainer, the CapEx optimizer first deploys containers to the data center to process the short-term peak demand, such that the expansion can be postponed to 2018. At this time point, the data center infrastructure is expanded because the CapEx can be minimized by upgrading the traditional data center to handle the

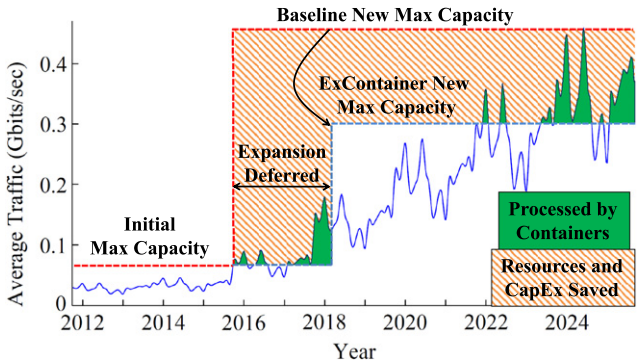

Fig. 3. Containers allow to defer and reduce the upgrade of traditional data centers to save CapEx.

amount of expected workload. Note that just increasing the number of containers after this time point would lead to higher CapEx than our solution, because of the higher amortization of costs of a fully utilized traditional data center. Once the traditional data center is expanded, a new cycle starts and new containers can be used to defer the next upgrade after 2022. As a result, the amount of wasted capacity and the degree of the upgrade are reduced over the lifetime of the data center.

3.3. ExContainer assumptions

For both the OpEx and CapEx optimizers, we assume some coarse-grained knowledge of future workload. This is reasonable because for the container allocator, we only need to know the *monthly averages* of electricity prices, renewable availability and ambient temperatures. While it is true that fine-grained prediction can commonly be difficult, such coarse-grained estimates (often provided directly from government's agencies such as US energy information administration [37]) can be accurate enough for the optimization purpose, as shown in previous studies such as [38]. Moreover, the usage of average values over a month allows the container allocator to have some degree of robustness over unexpected short-lasting spikes (e.g., an unexpected spike in electricity price of two hours or one day has negligible influence on the average price over the same month). For the workload dispatcher, we only need to know the *hourly averages* of those factors, which can also be accurate enough. For example, in an hour-ahead power market, the electricity prices are known one hour ago. Wind speed and temperatures also normally would not deviate significantly from their prediction within one hour. For the CapEx optimizer, a lot of prior work has shown that the *annual average amount* of data center workload can be predicted with acceptable accuracy [39,40]. For this reason we do not focus on implementing new prediction techniques at different time scales. We assume only all-in-one containers (IT equipment and cooling system in the same container) in the rest of the paper.

4. ExContainer OpEx optimizer

In this section we describe the power models and the optimization problems for the two components of the OpEx optimizer: the container allocator and the workload dispatcher.

4.1. Power models

Table 1 summarizes the notations used in the OpEx optimizer. The average server utilization ($u_{i,s}$ or $u_{j,s}$) is defined as the ratio of the request arrival rate over the average processing rate (assumed to be predictable). For simplicity, we model the power demands of a traditional (i.e., room-based) Data Center (DC) site and a container as functions of server utilization [4] and *PUE*, for which we adopt a model similar to that in [3]:

$$P_{DC_i}(t) = \sum_{s=1}^{S} (a_1 + b_1 u_{i,s}(t)) m_{i,s} PUE_i(t) \tag{1}$$

$$P_{C_j}(t) = \sum_{s=1}^{S} \left(a_2 + b_2 u_{j,s}(t)\right) m_{j,s}(t) PUE_j(t) \tag{2}$$

where $t \in \{t_m, t_h\}$ indicates the time-scale (monthly for the container allocator, hourly for the workload dispatcher). a_1, a_2, b_1, and b_2 are parameters that characterize the server power consumption and can be estimated from offline server profiling. *PUE* indicates the cooling efficiency. For example when the outside air is cool enough, the *PUE* can be reduced from 1.35 to 1.05 for an HP POD container, and from 1.83 to 1.56 for a traditional DC [41]. To be more general, we assume that a traditional DC may be setup to handle only a subset of all the services S, as the case in Yahoo! data centers [34]. For example, in Fig. 1, Data Center 1 cannot handle chat service while Data Center 2 cannot handle email service (due to lack of required software or database). In addition, we assume that the number of servers assigned to each service (i.e., $m_{i,s}^{\max}$, including active

Table 1
ExContainer OpEx optimizer variables.

N, M	# of traditional Data Centers (DCs), and containers
S	# of different services offered by IT company
P_{DC_i}, P_{C_j}	Power demand of ith DC and jth container
$P_{sub_i}^{\max}$	Max power of the ith local power substation
P_{G_i}	Local green power of the ith DC
C_{e_i}	Local electricity cost of the ith DC (\$/h)
C_{T_j}	Transportation cost of the jth container (\$)
$m_{i,s}, m_{j,s}$	Service s active servers in ith DC and jth container
$L_{i,s}, L_{j,s}$	Service s requests to ith DC and jth container
L_s^{tot}	Total # requests for the sth service
$u_{i,s}, u_{j,s}$	Service s server utilization of ith DC and jth container
δ_{c_j}	Location of the jth container ($N \times 1$ binary vector)

and inactive ones) in the traditional DC is fixed. However, our scheme can be easily extended to consider that each server can change configuration to serve various services. On the other hand, because containers may change location over time (i.e., each time the container is configured it needs to change workload processed depending on the location), the number of containerized servers dedicated to each local service (i.e., $m_{j,s}^{\max}$) is optimized monthly with the container allocator.

4.2. Container allocator

The objective cost function that calculates the total OpEx of the DCs and containers is the sum of electricity and container transportation costs (see Appendix A). The optimization problem that minimizes OpEx each month can be expressed as follows:

$$\min_{\delta_{c_j}, u_{i,s}, u_{j,s}, m_{j,s}^{\max}} \sum_{i=1}^{N} \sum_{j=1}^{M} \left\{ C_{e_i}(t_m)\tau + C_{T_j}(t_m) \right\} \tag{3}$$

where τ is the number of hours in a month and $\forall i \in \{1, N\}, j \in \{1, M\}, s \in \{1, S\}$, subject to the following constraints:

1. The workload of each service is completely dispatched:
   $\sum_{i=1}^{N} \sum_{j=1}^{M} L_{i,s}(t_m) + L_{j,s}(t_m) = L_s^{tot}(t_m)$
   where $L_{i,s}$ and $L_{j,s}$ depend on manipulated variables;
2. The total power demand of each data center site is lower than the substation power capacity:
   $P_{DC_i}(t_m) + \sum_{j=1}^{M} \delta_{c_{i,j}}(t_m) P_{C_j}(t_m) \le P_{sub_i}^{\max}$
3. The workload of service s is dispatched to a site (either to traditional DC and containers) only if the local traditional DC hosts s:
   $0 \le u_{i,s}(t_m) \le \begin{cases} 1 & \text{if the } i\text{th traditional DC hosts } s \\ 0 & \text{otherwise} \end{cases}$
   $0 \le u_{j,s}(t_m) \le f_{i,j}^1(t_m)$
   where $f_{i,j}^1$ is zero if the ith traditional DC does not host service s, or the fraction of time that the jth container is not being transported over a month (i.e., ≤ 1);
4. Every containerized server is configured to host a local service, with no one left unassigned:
   $0 \le m_{j,s}^{\max}(t_m) \le f_{i,j}^2(t_m) m_j^{\max}$
   $\sum_{s=1}^{S} m_{j,s}^{\max}(t_m) = m_j^{\max}$
   where $f_{i,j}^2(t_m)$ is zero if the ith traditional DC does not host service s, or 1 otherwise.

Based on the trade-off between the container transportation cost and the electricity cost, the container allocator decides whether to leave a container at the current data center or move it to another site. Note that because of constraint 1 the average workload demand of each service influences the optimal allocation of each container (indeed, the amount of workload influences the expected power demand of each data center). Therefore, if the workload peaks for some services are expected to cause some power peaks that exceed the capacity of a traditional DC (e.g., sport event, launch of new software), the container allocator ensures the processing of the demand by allocating more containers where that service can be processed. Locality of service in terms of data availability moreover, can be ensured by allowing the container to update its data once deployed in the new location according to the optimal number of servers $m_{j,s}^{\max}$ (dynamically calculated by the optimizer) of the jth container dedicated to each local service s.

4.3. Workload dispatcher

Once the container allocator has deployed the containers and allocated their servers to desired services, in every hour the workload dispatcher calculates how much workload to dispatch to traditional DCs and containers, and how many servers

to turn on/off. The optimization problem is formulated as follows:

$$\min_{u_{i,s},m_{i,s},u_{j,s},m_{j,s}} \sum_{i=1}^{N} C_{e_i}(t_h) + D_i(t_h) \tag{4}$$

where $D_i(t_h)$ is the network delay cost, a function of the distance between the ith traditional DC and the dispatcher, as assumed in [1]. The constraints are similar to those explained for the container allocator, and thus not shown due to space limitations. An additional constraint is that the maximum allowed response time of each service s should not be exceeded at any time t_h:

$$RT_s(u_{i,s}, m_{i,s}, u_{j,s}, m_{j,s}) \leq RT_s^{\max} \tag{5}$$

where the response time RT_s varies with the manipulated variables and is estimated using Allen–Cullen approximation for G/G/N queuing models [42].

Note that the workload dispatcher is also influenced by (1) ambient temperatures (variable PUE), (2) electricity prices (R_{e_i} see Appendix A), and (3) green energy availability (P_{G_i} see Appendix A). Based on these factors, the workload dispatcher exploits the additional computing capacities provided by the containers to process more workload at lower costs. As we show in Section 6, the consideration of temperature and cooling efficiency across geo-distributed data centers (not considered in previous literature) helps our scheme to outperform the state-of-the-art solutions.

We solve the two optimization problems using the non-linear programming solver of Matlab. We measure the execution time of running the OpEx optimizer on a server with an Intel 3.30 GHz i5 processor for 10 data centers, considering Google owns 13 data centers, 7 of which are in the US, and Facebook owns 3 data centers in total. Our results show that the workload dispatcher takes no longer than 1 min to get the optimization results, which is much shorter than the 1 h period. The time bound is not violated also for a larger number of data centers (about 120). However very few (if not none) IT companies own more than tens of data centers. Moreover, the actual trend of IT companies is to keep low the number of geo-distributed data centers to include at same time a disaster recovery plan and a consolidation plan of services [22]. Therefore, a centralized solution should be sufficient to handle the communications between dispatcher and data centers of an IT company. For these reasons we leave as future work the development of near-optimal algorithms to further decrease computational needs.

5. ExContainer CapEx optimizer

In this section, we first formulate the CapEx optimization problem. We then describe the optimization algorithm, and third we define the cost models in detail.

The CapEx of building a data center includes the expenses of servers (C_S), cooling (C_{Cool}) and power system (C_P), server room (C_B), and eventually the cost for buying containers (C_{Cont}) according to a well established CapEx depreciation cost model explained in [6]:

$$CapEx_i = C_S + C_{Cool} + C_P + C_B + C_{Cont}. \tag{6}$$

The model to derive the CapEx for containers is presented in this section, while all the others are presented in Appendix B. Since we look for the best time to upgrade a traditional DC with a time scale of 3 years (the lifetime of a container), the CapEx needs to be recalculated as time goes on (i.e., $CapEx_i(t)$), based on the number of servers added to the traditional DC ($N_i^{add,dc}(t)$) and added through containers ($N_i^{add,cont}(t)$). Hence the problem of minimizing the total CapEx of geo-distributed data centers is formulated as follows:

$$\min_{N_i^{add,dc}(t),N_i^{add,cont}(t)} \sum_{i=1}^{N} CapEx_i(t) \tag{7}$$

such that the total power demand keeps below the power substation capacity. The power substation is commonly over-sized compared to the power demand of the traditional DC. For example, the RaiginWire 17 MW data center in Sacramento has a 24 MW power substation, and the 5 MW Austin Data Center (ADC) is provided with a dedicated 400 MW power substation. Therefore, the substation can power some containers in addition to the traditional DC. We regard the power substation as a backup to the green farm, which is sized to support a certain percentage of the total energy consumption. This configuration can be found in many data centers of big IT companies such as Apple (solar farm) in North Carolina, eBay (Bloom Energy) in Utah, or Yahoo! (hydroelectric) in New York.

Because an expansion of the traditional DC has to comply with its lifetime, every three years the solution of the CapEx optimization is divided in two sub-problems (both using cost function (7)). In the first problem, the optimization calculates the optimal trade-off between the maximum capacity of the server room $N_{DC_i}^{\max}(t)$ and the maximum number of containerized servers to match the DC lifetime (e.g., 15 years) workload peak, assuming that the expansion is done in the current time interval. This optimization allows decreasing the degree of the upgrade. Note that it is out of scope of this paper to find an algorithm that consolidates workload to fewer servers to avoid over-provisioning. On the other hand, we calculate the number of servers needed to process a certain amount of workload by assuming the IT company already makes usage of consolidation algorithms to increase the utilization of server to near 100%. The second optimization uses the result of the

first optimization to calculate whether we actually should expand the traditional DCs (sizing server room, power, and cooling to host a max of $N_{DC_i}^{\max}(t)$ servers) to install $N_i^{add,dc}(t)$ servers, rather than employing more containers and install $N_i^{add,cont}(t)$ servers through containers (in either case the installed servers must match the peak of next 3 years). A traditional DC is expanded if (1) it is more economical than buying more containers, or (2) the power capacity of the substation is reached. Because this optimization is offline, we solve it through exhaustive search as in [26].

The depreciation cost of the containers deployed at the ith traditional DC can be calculated as

$$C_{Cont}(t) = R_{Cont} P_{Cont_i}^{tot}(t) \tag{8}$$

where R_{Cont} is the depreciated price of containers ($/Watt/year); $P_{Cont_i}^{tot}(t)$ is the total power capacity of all the local containers calculated as $N_i^{add,cont}(t) P_S^{\max} PUE$ (Eq. (2)), where $P_S^{\max}$ is the maximum server power. The depreciation cost models of the traditional DC are included in Appendix B, as they are derived similarly to the model of containers. In this paper, some minor capital expenses of data centers such as building grid connections, water ducts, and land, are omitted but may be easily included in the optimization problem.

6. Simulation results

In this section, we evaluate ExContainer using real-world traces of the workload and the influential factors, such as electricity price, green energy and ambient temperature.

6.1. Simulation setup

We assume each server consumes at most 500 W, and define the maximum power demand of all the servers in a data center as critical power. The simulated IT company owns 3 data centers located in New York City (NYC, 40,000 servers, 20 MW critical power), Atlanta (ATL, 30,000 servers, 15 MW critical power), and Chicago (Chi, 20,000 servers, 10 MW critical power). Each data center has an on-site wind farm to generate green energy, and a power substation that is over-sized with respect to the real peak power demand. This headroom capacity is usually reserved for future needs [43,44], but can also be used to power containers. Each container (40-foot) can host up to 3000 servers. We use the real distances between each pair of cities to calculate transportation time and costs, assuming a speed of 60 miles/hour and a transportation rate of $2.5/mile [45]. We randomly position the workload dispatcher in NYC, which affects the network delay cost as in [1]. We use a 2-year real-world workload trace from an IP-traffic monitor located in San Jose [35]. Although other data center workload traces can also be used in our evaluation, we choose this trace because it is long enough (2 years) to test the CapEx optimizer. We use real-world traces of electricity prices [33], wind speeds [46], and temperatures [46] from January 2012 to December 2012 in the 3 cities. The total workload demand is about 7% higher than the total capacity of the 3 data centers and can be processed with two additional containers. To show the impact of both electricity price and renewable energy availability, we size each on-site green farm to support half of the maximum power demand of the local data center. The cooling system of containers works in the economizer mode when the ambient temperature is below 33 °C [8] (below 2 °C for traditional DC [5]).

6.2. OpEx minimization

To evaluate the OpEx optimizer, we first test a simple case in which the 3 data centers provide the same service. To show how the container allocator and the workload dispatcher influence the OpEx, and how their coordination leads to further minimized OpEx, we compare our scheme with five baselines:

- Workload Dispatcher (WD): our OpEx optimizer without container allocator;
- Geographical Load Balancing (GLB) [1]: a state-of-the-art solution that dispatches workload to balance green energy usage and electricity cost (without the consideration of cooling cost or container allocation);
- Lowest Price (LP) [2]: a state-of-the-art solution that dispatches workload to data centers with lower energy prices (without the consideration of container allocation);
- MaxGreen [14]: a state-of-the-art solution that dispatches workload to data centers with more green energy (without the consideration of container allocation);
- MinTemp: a heuristic solution that dispatches workload to data centers with lower ambient temperature (without the consideration of container allocation).

6.2.1. Monthly container positioning

Fig. 4 shows the OpEx saved by the container allocator by comparing ExContainer to WD with the containers always in a certain location. WD can achieve a little less OpEx in some months when this city happens to be the optimal site to process workload (i.e., zero transportation costs). In general moving the containers can significantly save OpEx up to $60K/month, by exploiting the variability of three influential factors.

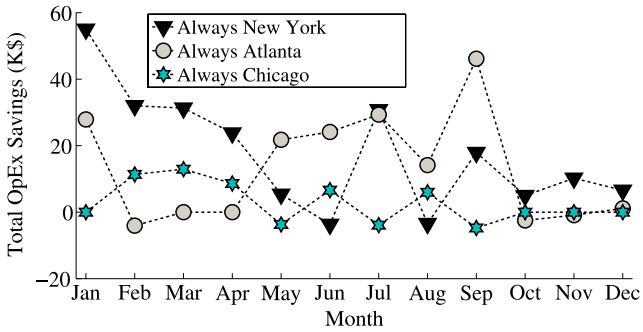

Fig. 4. OpEx savings achieved by moving two containers among three data centers relative to always leaving them in a certain position.

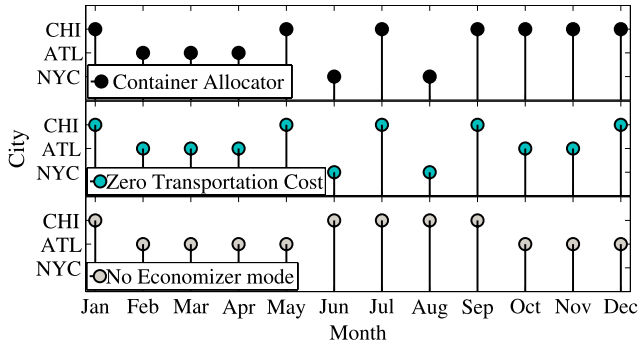

Fig. 5. Optimal positions of the two containers in different cases, determined by the container allocator.

The upper curve of Fig. 5 shows the positions of the two containers during the year. The ambient temperature, which increases from winter to summer, is important for selecting the optimal position of containers. In Atlanta the temperature exceeds the economizer threshold of containers in May, while in Chicago it stays below this threshold until June. Thus the containers are moved from Atlanta to Chicago in May. In June and August containers are positioned in New York City since it is the only one with temperature lower than the containers threshold (in July also NYC exceeds). During the other 8 months, the low temperature in the 3 cities allows the containers to work with the economizer mode everywhere, and the optimal position is mainly determined by electricity price, green energy availability, and container transportation costs. To exclude the influence of temperature and study the other two factors, we disable the economizer mode to generate the lower curve in Fig. 5. In this scenario, the containers are moved between Atlanta and Chicago to exploit the better combination of low electricity price and green energy availability. New York City is ignored as possible position for containers because of their higher power demands for the not consideration of the economizer mode. Indeed, NYC has higher electricity price but lower temperature compared to the other cities during June and August. In the middle curve of Fig. 5 we show how the transportation costs influence the allocation, by assuming zero transportation costs. The containers are moved to Atlanta in October and November instead of staying at Chicago because of the higher savings achievable (Fig. 4). However, the savings achievable by moving containers in Atlanta are not worthwhile compared to the transportation cost. Therefore, when all the costs are considered, the containers remain in Chicago. This example highlights the importance of considering transportation costs in the cost function of the container allocator. In Section 7 we discuss other possible extra costs that may be included in ExContainer for our future work.

These experiments prove that moving containers among distributed data centers according to variations on electricity price, green energy, and ambient temperature leads to OpEx savings over static positioning solutions.

6.2.2. Hourly workload dispatching

We use the first week in January 2012 of our traces to evaluate the workload dispatcher (containers not moving). Fig. 6 compares the workload dispatcher of ExContainer with the baselines on the total OpEx and the container average server utilization. For ExContainer, the modules are placed in Chicago because it is the optimal position in January (Fig. 5), while for the only WD the containers are located in non-optimal sites (NYC and ATL). The baselines LP, MaxGreen, MinTemp and GLB have the containers in Chicago as well for fairness.

Our workload dispatcher (WD) takes the electricity price, the renewable energy availability and the ambient temperature into consideration, while each baseline addresses only one or two of the three factors: LP dispatches more workload to Chicago as its electricity price is usually lower; MaxGreen aims at the green energy and hence prefers New York City;

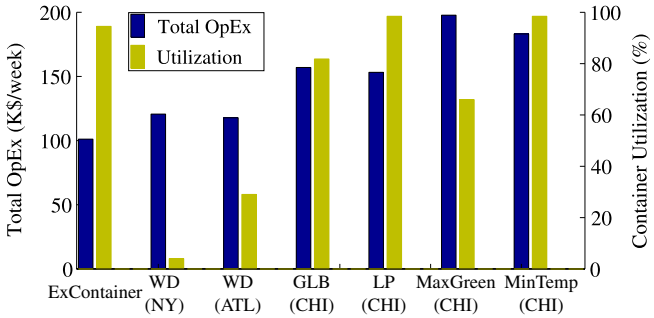

Fig. 6. Comparison between ExContainer and the baselines in terms of OpEx and container utilization.

MinTemp focuses on the ambient temperature and thus dispatches little workload to Atlanta; GLB considers both the electricity price and the green energy but neglects the influence of temperature on cooling. WD instead, finds the best trade-off among the three factors to determine how to dispatch workload. As a result, it saves between $30 K/week to $80 K/week over the baselines. The diversity of the workload dispatching policies also leads to different utilization of the containers. LP and MinTemp give higher priority to Chicago and therefore utilize the containers more. MaxGreen and GLB prefer to dispatch workload to New York City and thus the containers in Chicago are less utilized. WD considers all these factors and balances the workload dispatching among the 3 cities to minimize OpEx. In this case, WD dispatches as much workload as possible to Chicago (the optimal location), which is fully utilized, while containers are utilized just enough to process the rest of the demand in non-optimal cities. As we discuss in next paragraph, ExContainer allows to increase the OpEx saved by allocating more resources to the optimal site.

These results prove that our workload dispatcher outperforms existing solutions by lowering OpEx even when containers are in a non-optimal location.

6.2.3. *The advantages of ExContainer*

Here we study how the integration of container allocator and workload dispatcher leads to additional OpEx savings. Fig. 6 shows the comparison on total OpEx between ExContainer and WD (our workload dispatcher). The former positions the containers in Chicago since it is the optimal location in January, and the latter has the containers in New York or Atlanta to make the difference. The workload dispatcher prefers to distribute more workload to the optimal position (CHI) that costs the least for processing. However, as seen in the previous results, the traditional DC in Chicago is already fully utilized and thus cannot process more requests. On the other hand, ExContainer allocates the two containers at this site such that more workload can be processed at lower costs. Indeed, the container utilization is close to 100%, while for WD it is only 5% in NYC and 30% in ATL. As a result, ExContainer leads to $16 K/week and $20 K/week more OpEx savings compared to the WD (ATL) and WD (NY), respectively.

These results show that when container allocator and workload dispatcher cooperate, the amount of requests processed at lower costs can be increased such that more OpEx are saved.

6.3. *Distributed services and ExContainer*

While an IT company runs distributed data centers to provide multiple services, each data center may only offer a subset of them, thus workload cannot be dispatched everywhere. In contrast to existing solutions, ExContainer is able to handle the constraints given in this scenario, yet it minimizes OpEx. We evaluate this case with traces of video (service 1), email (service 2) and HTTP (service 3) from [35] and simulate a new scenario as follows:

- The data center in New York City offers services 1 with 30% of its servers and service 3 with 70% of its servers.
- The data center in Atlanta offers services 2 with 70% of its servers and service 3 with 30% of its servers.
- The data center in Chicago offers only service 2.

Note that only the data center in New York City provides service 1. In order to clearly show the advantages of ExContainer, we simulate a workload peak of service 1 that is 30% higher than the maximum processing capacity from February to April. Hence, the optimal location of the containers is determined also by how different services are distributed.

Fig. 7 shows the positions of the two containers during the year arranged by ExContainer. The primary principle of the container positioning is to satisfy the workload demand of each service, and then minimize the OpEx. For example, between February and April they are positioned in New York City to provide additional servers to process the workload peak of service 1; in August, Container 1 stays at Chicago to process more service 2 at lower costs (Fig. 9), and Container 2 is moved to New York City to process service 1 and exploit the economizer mode. Fig. 8 compares the amount of service 1 requests that are processed by using ExContainer or the two WD schemes whose containers are fixed in Atlanta or Chicago. Only ExContainer successfully has all the requests processed, while the two WD schemes fail to process part of the service 1 requests, because the number of servers dedicated to this service is inadequate to handle the workload peak.

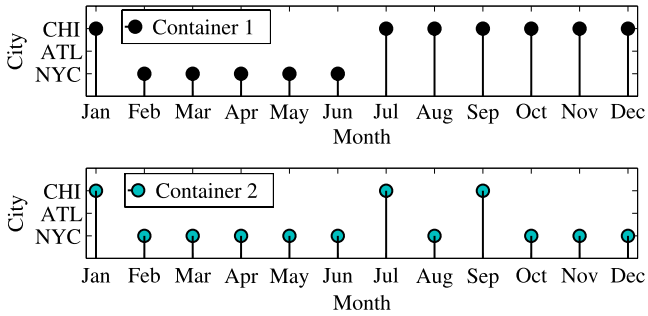

Fig. 7. Simulated peak of workload for service 1. The positioning of containers can be affected by the amount of service requests.

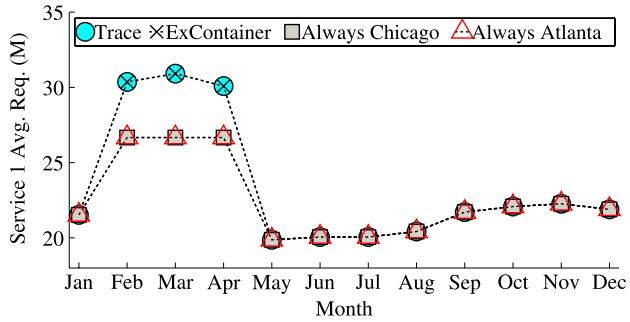

Fig. 8. ExContainer ensures that all the requests can be processed even if there is a workload peak of a certain service.

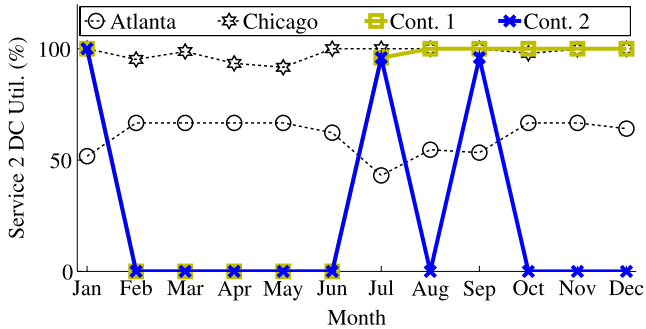

Fig. 9. Data center average utilization to process service 2 requests. Containers are highly utilized to minimize OpEx for different services.

Fig. 9 shows the utilization of servers in the traditional DC and the containers for processing the service 2 requests. Comparing it to Fig. 7, we can observe that a container is fully utilized for service 2 as long as it is positioned in Chicago; otherwise it does not process service 2 because of being positioned in New York City, where service 2 is not offered. Therefore, the containers are never under-utilized when they are deployed at the optimal positions, because processing the workload in a container is more energy-efficient than in a traditional DC due to the different *PUE*s. The utilization of a container can be a little less than 100% in a month if it is transported from one city to another (e.g., the two containers are moved from New York City to Chicago in July), during which the container cannot be utilized.

These results show that ExContainer outperforms existing workload dispatching solutions, by moving containers among data centers to provide more computational resources when the workload demand of a service exceeds the capacity.

6.4. CapEx minimization

In this section, we evaluate ExContainer in terms of CapEx savings. The real-world parameters used in the simulation are summarized in Table 2. We compare ExContainer with two baseline strategies that modern IT companies commonly adopt to increase the computing capacity of a data center to process increasing workload demand:

- Containers-Only [50]: employing more containers to add servers, without expanding the traditional DC;
- Expansion-Only [6,10]: expanding the traditional DC to allow adding servers, without employing containers.

Table 2
Parameters used in the testing of the CapEx optimizer.

Server cost & lifetime	$2000 & 5 years
Container *PUE*	1.35 [8]
Container cost & lifetime	$4.5/W & 3 years [47,30]
Wind farm cost & lifetime	$2.5/W & 20 years [48]
Wind turbine capacity factor	40% [49]
Cooling system cost & lifetime	$2.5/W & 15 years [5]
Structure expansion cost & lifetime	$4.4/W & 15 years [6]

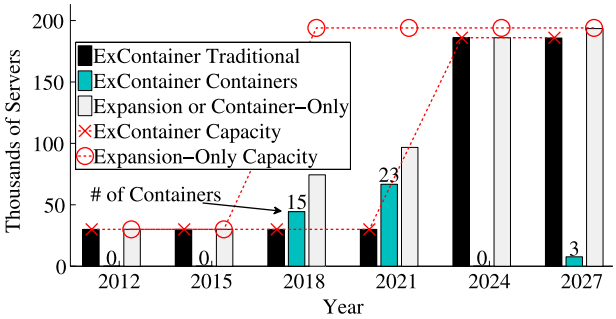

Fig. 10. Number of servers added in the traditional DC and the containers.

Although the CapEx optimizer can handle all the 3 simulated data centers, here we show the results for the data center in Atlanta (hosting 30,000 servers). This case is to simulate that the IT company wants to handle exceeding requests by upgrading only this data center. However, the results for multiple data centers are similar but are not shown due to space limitations. Either ExContainer or the baselines requires upgrading the power substation after a certain number of servers have been added. We assume that at the beginning the capacity of the power substation is 70 MW, which is 2.3 times as the maximum power demand of the data center (i.e., the oversize factor p_{ov} is 2.3, see Appendix B). The initial workload demand is assumed to be as much as utilizing 40% of the servers on average; we set the workload demand to increase 18% per year according to a recent study on traffic growth [36]. During each 3 years, the peak workload demand determines the number of needed servers, including those in the traditional DC and the local containers.

Fig. 10 shows (1) how servers are distributed in the traditional DC and the containers, and (2) the capacity of the traditional DC in terms of the maximum number of servers that can be accommodated. The initial capacity of the traditional DC allows adding servers until 2018. In 2018 and 2021 the flexibility of containerized data centers allows to save CapEx over the immediate expansion of the traditional data center. Expansion-Only indeed expands the traditional DC to accommodate more servers, and the power substation is upgraded also by considering the oversize factor. We can observe that Expansion-Only causes a significant waste of the capacity of the traditional DC by increasing the size of the server room, power and cooling infrastructure to host 194,000 servers because the new capacity is determined by the peak workload demand in the next 15 years, while less than 100,000 servers are actually needed until 2024. In contrast, Container-Only and ExContainer start to add containers in 2018. Fig. 11 shows the total CapEx needed for the upgrade of computing capacity, comparing ExContainer with the baselines. In 2018 and 2021, 15 containers and 23 containers are employed, respectively, by ExContainer or Container-Only, and the avoided upgrade allows the IT company to save $30M/year to $40M/year.

In 2024, the expected workload demand becomes high enough to cancel the advantage of employing more containers to add servers, which becomes less economical than expanding the traditional DC to add servers. This is caused by two reasons: (1) the expanded traditional DC can be filled of servers now as it has much more workload to process in 2024 than in 2018, and thus the CapEx are better amortized over all the servers; (2) the shorter lifetime of containers becomes the major obstacle for CapEx savings. ExContainer determines to upgrade the traditional DC to host 183,000 servers, which is just enough for the 3 years from 2024. This choice makes ExContainer to save about $5M/year compared to Container-Only. In 2027, ExContainer needs to add new servers by employing 3 containers, which defers the expansion of the traditional DC again.

These results show how ExContainer proposes a middle-way strategy between two existing solutions for saving CapEx over data centers expansion. First (as in 2018 and 2021), ExContainer takes advantage of containers' flexibility by handling short-term peaks of workload. Second (as in 2024), it takes the advantage of the fully utilized traditional data center on better amortizing costs, by determining the right time for an expansion when the average workload demand is high enough. Moreover, while the CapEx optimizer adds containers to the data centers before 2024 to handle extra demand, the OpEx optimizer can schedule the positions of those containers every month across geo-distributed data center sites. We repeat the experiment for ExContainer as an integration of the CapEx optimizer and the OpEx optimizer, and the resulted CapEx and OpEx are shown in Fig. 12.

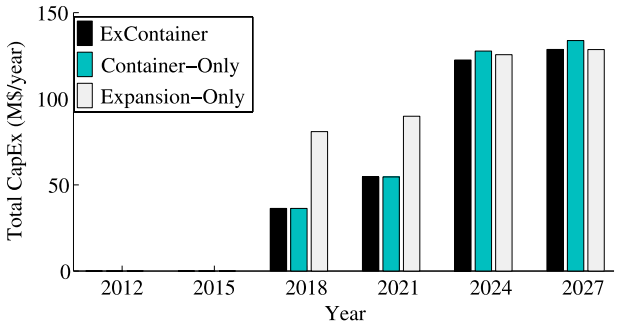

Fig. 11. Comparison on total CapEx. ExContainer costs the same or less CapEx.

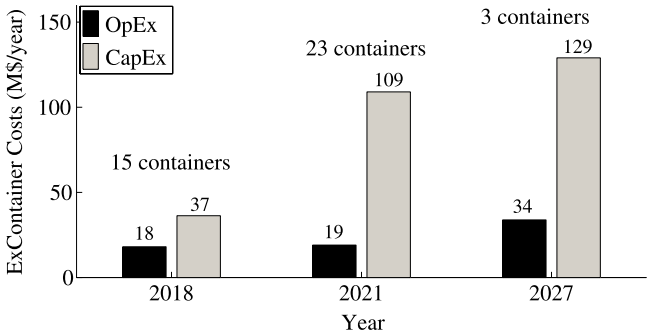

Fig. 12. The annual CapEx and OpEx achieved by ExContainer as the workload increases.

7. Discussion

In this section, we discuss implications of our study for the dynamic allocation of containerized data centers. First, we discuss transportation-related concerns and data management of containers. Second we discuss the possible extra costs that may be associated with the transportation of the containerized modules.

Typical transportation-related concerns include the risks associated with deploying modules, the environmental impact, and the effect of delays on costs. Containerized data centers are optimized and built specifically to ensure portability and equipment security. In particular, some containerized data center models include an anti-earthquake and anti-vibration system to avoid damage to servers and other equipment [51]. Moreover, the procedure for the installation of a container mainly consists of three standard steps: offload the container from the truck, plug power and network cables, and configuration. Anti-vibration system and the usage of standard installation procedures highly reduce the risks associated with relatively frequent transportation across geo-distributed data centers. On the other hand, big IT companies such as Google, Microsoft, and Facebook often own data centers located in different countries or continents, such that truck transportation is not possible. In this case the transportation of containerized data centers may be done through cargo ships. However we limit ExContainer by considering only truck transportation because (1) it requires shorter loading and unloading time of the module itself and (2) it comes with less risks than cargo ship transportation. We leave as future work the inclusion in the optimization framework of other types of transportation. Another concern related to the transportation of containerized modules is the impact on the environment in terms of CO_2 emissions. Even though this paper does not aim to move modules for reduced CO_2 emissions, we believe that the usage of modern technologies such as hybrid and electric shipping trucks can help on reducing the environmental impact of ExContainer. The last concern related with transportation is the effect of traffic and possible delays on costs. Our framework can account for transportation delays by changing the average transportation speed (Appendix A). Simulation results conducted by reducing the average transportation speed from 60mph to 20mph, show an increase in OpEx by less than 1%, thus proving the robustness of ExContainer on transportation delays.

Another related concern is data consistency after transportation. While ExContainer can be applied to any type workload (batch or interactive), we assume that data consistency is ensured by updating data on containerized servers when the container is being configured. Moreover, our framework can be easily extended to account for the need of replicas for a specific service, simply by increasing the average service workload in the first constraint of the container allocator optimization problem (Section 4.2).

Extra costs associated with the transportation of containerized data centers are related mainly to (1) downtime costs, due to unavailability of servers while containers are being transported, and (2) insurance, installation, and de-installation costs. Downtime is the cost associated with planned or unplanned unavailability of equipment. Planned downtime often occurs for maintenance such as database maintenance, operations such as backups, or periodic events such as hardware upgrades,

and accounts for at least 80% of all the downtime. The remaining 20% is unplanned downtime due to application errors or hardware failures [52]. The downtime can be carefully planned to minimize the costs, e.g., by scheduling backups overnight when workload is low. Moving a container across data center sites can be considered as part of the planned downtime. Indeed, the large difference of workload demand between day and night, weekdays and weekends, and among weeks of the month, allows to have some time periods in which part of the servers are not needed and can be shut down to save power, as proposed by many research studies such as [1,14–16]. Therefore, moving containers can be carefully planned overnight or during weekends to avoid service disrupts. In this paper we do not include the problem of finding the best period to move the containers as it would increase the complexity of the optimization problem. On the other hand, we guarantee the respect of service level agreements in the current implementation of the container allocator by allocating containers to data center sites that require extra servers to process a peak of workload demand (as shown in Section 6.3). Therefore, the probability of an unplanned downtime due to peak demand while moving the containers is reduced. It is part of our future work to make the moving time of containers flexible and dynamically chosen by the algorithm itself to minimize downtime costs. Other possible extra costs may come from an insurance policy and from the installation of the modules every time they are transported to another data center. Despite these extra components may increase the total costs, we believe that the transportation costs account for the bigger part of the total. On the other hand, big IT companies may build, install, and transport their own containerized modules, leading to reduced transportation costs. We plan to consider more general cost models for the container transportation in our future work.

While containerized modules have already been increasingly used by IT companies, though not yet for the purpose of reducing OpEx and CapEx as we propose, we believe that this study makes the first step forward to motivate a novel strategy of allocating containerized modules and associating the allocation with dynamic workload dispatching.

8. Conclusions

A major limitation of dynamic workload dispatching is the processing capacity of each data center site, which can be addressed by moving containers around for increased capacities. In addition, the expensive upgrades of the power and cooling infrastructures of a data center can also be delayed with container deployment. In this paper, we have presented ExContainer, a novel strategy that leverages portable containerized modules to minimize both OpEx and CapEx for geo-distributed data centers. With ExContainer, containers are moved monthly among different data center sites based on electricity prices, green energy availability, and ambient temperatures. As a result, hourly workload dispatching can achieve a significantly lowered OpEx. To cut CapEx, ExContainer handles short-term workload peaks with containers by taking advantage of the fast module deployment, such that the expensive upgrades of the power and cooling facilities can be deferred for a minimized CapEx. We have evaluated ExContainer with real-world traces of data center workload, electricity price, green energy availability, and ambient temperature. Our experimental results show that compared to state-of-the-art solutions, ExContainer achieves an OpEx savings up to \$80 K/week, a CapEx savings up to \$40M/year for our case study of three geo-distributed data centers with 90,000 servers.

Appendix A. OpEx optimizer transportation and cost models

We indicate an element of the vector δ_{c_j} (Table 1) as $\delta_{c_{i,j}}$. d is an $N \times N$ matrix with elements indicating the distance between two data centers. T_0 is an $N \times N$ matrix, which indicates the time to plug or unplug containers. v is the average transportation speed. Because only one element of $\delta_{c_j}(t)$ can have value 1, the average time $T_{T_j}(t)$ to ship the jth container is calculated as follows:

$$T_{T_j}(t) = \frac{\delta_{c_j}^T(t-1)d\delta_{c_j}(t)}{v} + 2\delta_{c_j}^T(t-1)T_0\delta_{c_j}(t). \tag{A.1}$$

We adopt the cost model presented in [53] to calculate the transportation cost as the product of the distance and an average price rate r (\$/mile):

$$C_{T_j}(t) = r\delta_{c_j}^T(t-1)d\delta_{c_j}(t) \tag{A.2}$$

Indicating with $R_{e_i}(t)$ the local electricity price in \$/(W hour), the electricity cost $C_{e_i}(t)$ is calculated with the power gap to cover by using the grid, when the green power is lower than the total demand of servers and cooling:

$$C_{e_i} = \left(P_{DC_i}(t) + \sum_{j=1}^{M} \delta_{c_{i,j}}(t)P_{C_j}(t) - P_{G_i}(t)\right)^+ R_{e_i}(t). \tag{A.3}$$

The electricity cost is zero when the green power generated is higher than the demand (indicated with the exponent $+$). Note that when $t = t_m$ (i.e., monthly for container allocator) $R_{e_i}(t_m)$ represents the average price (in \$/(W hour)) of the electricity during the month t_m.

Table 3
ExContainer CapEx optimizer variables.

$P_S^{\max}$	Max power of single server (W)
$P_{Cool}^{\max}$	Max cooling power
$P_{Sub}^{\max}$	Max power substation
$P_F^{\max}$	Max power green farm
R_S	Depreciated price of servers ($/serv/y.)
R_{Cool}	Depreciated price cooling system ($/W/y.)
R_{Sub}	Depreciated price of power substation
R_F	Depreciated price of green farm
R_B	Depreciated price of structure

Appendix B. CapEx optimizer traditional data center cost models

The costs for the addition of $N_i^{add,dc}(t)$ containers in a traditional DC are composed of server cost, cooling cost, power infrastructure cost, and structure cost. The yearly depreciation costs of servers added in the traditional structure can be calculated as (see Table 3 for notation)

$$\begin{aligned}
&C_S(t) = R_S N_i^{add,dc}(t)\\
&C_{Cool}(t) = R_{cool}\left(P_{Cool_i}^{\max}(t) - P_{Cool_i}^{\max}(t_0)\right)\\
&C_P(t) = R_{Sub}\left(P_{Sub_i}^{\max}(t) - P_{Sub_i}^{\max}(t_0)\right)^+ p_{ov} + R_F\left(\frac{p_g P_{Sub_i}^{\max}(t)}{c_f} - P_{F_i}^{\max}(t_0)\right)^+\\
&C_B(t) = R_B\left(N_{DC_i}^{\max}(t) - N_{DC_i}^{\max}(t_0)\right)P_S^{\max}
\end{aligned} \tag{B.1}$$

where c_f is the coefficient factor of the green farm (which measure the efficiency of the green generators), $N_{DC_i}^{\max}(t)$ is the maximum server capacity of the data center, p_{ov} is the oversize factor of the power substation (commonly found in modern designs), and p_g is the percentage of the total power demand that is supplied by a local green farm. The cooling system size is based on the maximum power demand $P_{cool_i}^{\max}(t)$ calculated as $N_{DC_i}^{\max}(t)P_S^{\max}/COP_i$. Note that we calculate all the costs based on the maximum number of servers initially hosted. Thus if an expansion is not needed, cooling, power system, and infrastructure costs are zero.

References

[1] Z. Liu, M. Lin, A. Wierman, S.H. Low, L.L.H. Andrew, Greening geographical load balancing, in: SIGMETRICS', 2011.
[2] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, B. Maggs, Cutting the electricity bill for internet-scale systems, in: Sigcomm, 2009.
[3] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, C. Hyser, Renewable and cooling aware workload management for sustainable data center, in: SIGMETRICS, 2012.
[4] Y. Zhang, Y. Wang, X. Wang, Electricity bill capping for cloud-scale data centers that impact the power markets, in: ICPP, 2012.
[5] J. Niemann, J. Bean, V. Avelar, Economizer modes of data center cooling systems, 2012.
[6] L.A. Barroso, U. Hlzle, The datacenter as a computer, An Introduction to the Design of Warehouse-Scale Machines, Mark D. Hill, University of Wisconsin, Madison, 2009.
[7] IBM, Portable modular data centers—add capacity where and when you need, White Paper, 2010.
[8] T.P. Morgan, Hp welds data center containers into EcoPODs, 2011. http://www.theregister.co.uk.
[9] M. Industria, Mdc plug & play POP, http://www.mdcindustria.com.br/.
[10] N. Rasmussen, Avoiding costs from oversizing data center and network room infrastructure, White Paper, 2012.
[11] I. Fried, Inside one of the world's largest data centers, 2009. http://news.cnet.com.
[12] R. Miller, Ebay containers push frontiers of efficiency, 2013. http://www.datacenterknowledge.com.
[13] R. Miller, Googles infrastructure boom continues expansion ahead in oregon, 2013. http://www.datacenterknowledge.com.
[14] Y. Zhang, Y. Wang, X. Wang, Greenware: greening cloud-scale data centers to maximize the use of renewable energy, Middleware, 2011.
[15] Y. Gao, Z. Zeng, X. Liu, P.R. Kumar, The answer is blowing in the wind: analysis of powering internet data centers with wind energy, in: INFOCOM, 2013.
[16] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, F. Zhao, Energy-aware server provisioning and load dispatching for connection-intensive internet services, in: USENIX, 2008.
[17] W. Fang, Q. Zhou, Y. An, Y. Li, H. Zhang, Cost-aware workload dispatching and server provisioning for distributed cloud data centers, in: IJGDC, 2013.
[18] J. Luo, L. Rao, X. Liu, Temporal load balancing with service delay guarantees for data center energy cost optimization, in: TPDS, 2013.
[19] T. Sakamoto, H. Yamada, H. Horie, K. Kono, Energy price driven request dispatching for cloud data centers, in: CLOUD, 2012.
[20] Z. Abbasi, G. Varsamopoulos, S.K.S. Gupta, Thermal aware server provisioning and workload distribution for internet data centers, in: HPDC, 2010.
[21] I. Goiri, K. Le, J. Guitart, J. Torres, R. Bianchini, Intelligent placement of datacenters for internet services, in: ICDCS, 2011.
[22] R. Singh, P. Shenoy, K.K. Ramakrishnan, R. Kelkar, H. Vin, eTransform: transforming enterprise data centers by automated consolidation, in: ICDCS, 2012.
[23] J. Koomey, K. Brill, P. Turner, J. Stanley, B. Taylor, A simple model for determining true total cost of ownership for data centers, Uptime Institute White Paper, 2007.
[24] N. Rasmussen, S. Niles, Data center projects: growth model, Schneider Electric White Paper, 2011.
[25] N. Rasmussen, S. Niles, Towards the design and operation of net-zero energy data centers, iTherm, 2012.
[26] S. Govindan, A. Sivasubramaniam, B. Urgaonkar, Benefits and limitations of tapping into stored energy for datacenters, in: ISCA, 2011.
[27] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, BCube: a high performance, server-centric network architecture for modular data centers, in: SIGCOMM, 2009.
[28] J. Hamilton, An architecture for modular data centers, in: CIDR, 2007.
[29] W. Torell, TCO analysis of a traditional data center vs. a scalable, containerized data center, Schneider Electric, 2012.
[30] K. Venkatesh, A. greenberg, D.A. Reed, Modular data centers: how to design them? in: LSAP, 2009.

[31] A. Qouneh, C. Li, T. Li, A quantitative analysis of cooling power in container-based data centers, in: IISWC, 2011.
[32] D. Bouley, W. Torell, Conteinerized power and cooling modules for data centers, Schneider Electric White Paper, 2012.
[33] NYISO archives. http://www.ferc.gov.
[34] Y. Chen, S. Jain, V.K. Adhikari, Z. Zhang, K. Xu, A first look at inter data center traffic characteristics via Yahoo! datasets, in: Infocom, 2011.
[35] Passive network monitors. http://www.caida.org/.
[36] Cisco global cloud index: forecast and methodology, Cisco Systems White Paper, 2012.
[37] EIA, Short-term energy outlook, 2013. http://www.eia.gov.
[38] S. Marcacci, Hyref technology revolutionizes renewable energy forecasting, 2013. http://cleantechnica.com/.
[39] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, F. Zhao, Energy-aware server provisioning and load dispatching for connection-intensive internet services. in: NSDI, 2008.
[40] K. Le, R. Bianchini, M. Martonosi, T. Nguyen, Cost and energy-aware load distribution across data centers, in: HOTPOWER, 2009.
[41] J. Kaiser, J. Bean, T. Harvey, M. Patterson, J. Winiecki, Survey results: data center economizer use, White Paper, 2011.
[42] A. Allen, Probability, statistics, and queueing theory with computer science applications, Academic Pr, 1990.
[43] NECA-IBEW, The silicon valley wire, White Paper, 2012.
[44] T. Mason, T. Curry, D. Wilson, Capital costs for transmission and substations. recommendations for WECC transmission expansion planning, Western Electricity Coordinating Council, 2012.
[45] G. Raballand, P. Macchi, Transport prices and costs: the need to revisit donors policies in transport in Africa, Bureau for Research and Economic Analysis of Development, 2008.
[46] Historical Weather. http://www.wunderground.com.
[47] D. Wells, How much should a modular data center cost, 2012. http://blog.schneider-electric.com.
[48] How much do wind turbines cost. http://www.windustry.org.
[49] U. o. M. A. Wind Energy Center, Wind power: capacity factor, intermittency, and what happens when the wind doesn't blow?.
[50] R. Miller, Microsofts $1 Billion Data Center, 2013. http://www.datacenterknowledge.com/.
[51] Huawei, All-in-one container data center solution, Data-sheet, 2012.
[52] V. Solutions, Understanding downtime, White Paper, 2006.
[53] T. Hanssen, T. Mathisen, F. Jrgensen, Generalized transport costs in intermodal freight transport, in: EWGT, 2012.

Marco Brocanelli is a Ph.D. student with the Department of Electrical and Computer Engineering at the Ohio State University, Columbus. He received his M.S. degree in Control Engineering at University of Rome Tor Vergata, Italy, in 2011. He received his B.S. degree in Control Engineering at University of Rome Tor Vergata, Italy, in 2008. He has been a J1 visiting scholar at Ohio State University between October 2010 and April 2011–October 2011 and May 2012, working on Hypersonic Vehicle non-linear Control.

Wenli Zheng is a Ph.D. student with the Department of Electrical and Computer Engineering at the Ohio State University, Columbus. He received his M.S. degree in Computer Engineering from University of Southern California, Los Angeles, California, in 2012, and his B.S. degree in Electrical and Computer Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010.

Xiaorui Wang is an Associate Professor in the Department of Electrical and Computer Engineering at The Ohio State University. He is the recipient of the Office of Naval Research (ONR) Young Investigator (YIP) Award in 2011, the NSF CAREER Award in 2009, the Power-Aware Computing Award from Microsoft Research in 2008, and the IBM Real-Time Innovation Award in 2007. He also received the Best Paper Award from the 29th IEEE Real-Time Systems Symposium (RTSS) in 2008. Prior to joining Ohio State, he was an Assistant Professor at the University of Tennessee, Knoxville, where he received the EECS Early Career Development Award, the Chancellor's Award for Professional Promise, and the College of Engineering Research Fellow Award in 2008, 2009, and 2010, respectively, as well as one-year early promotion with tenure. In 2005, he worked at the IBM Austin Research Laboratory, designing power control algorithms for high-density computer servers. From 1998 to 2001, he was a senior software engineer and then a project manager at Huawei Technologies Co. Ltd., China, developing distributed management systems for optical networks. He received his doctoral degree from Washington University in St. Louis. His research interests include computer systems, computer architecture, data center power management, embedded and real-time systems, and cyber–physical systems. He is an author or coauthor of more than 80 refereed journal and conference publications.