Profit-Based File Replication in Data Intensive Cloud Data Centers

Muhannad Alghamdi[1], Bin Tang[1], and Yutian Chen[2]
[1]Computer Science Department, California State University Dominguez Hills, USA
[2]Economics Department, California State University Long Beach, USA
Email: malghamdi4@toromail.csudh.edu, btang@csudh.edu, Yutian.Chen@csulb.edu

***Abstract*—Many of the applications running in cloud data center are data intensive, processing large amount of data inside the data center. File replication, which brings data files closer to the computing virtual machines (VMs), is an effective strategy that reduces data access latencies and bandwidth consumption, thus saving energy in data centers. In this paper, we formulate and study the file replication problem (FRP) in data center, with the goal of minimizing the total energy consumption of data file access inside data centers. In contrast to all the existing work of data replication in data centers, which are mainly heuristic based, we design a time-efficient approximation algorithm with performance guarantee for energy consumption in file replication. In particular, our file replication algorithm is based on a novel concept called "profit", and optimizes over a submodular function that can be computed efficiently. Our algorithm yields the total profit of file replication at least half of what is achieved by an optimal replication solution. We also design two energy- and time-efficient heuristic file replication algorithms. Via extensive simulations using CloudSim, a popular simulation framework for cloud computing, we compare all the algorithms under different network scenarios. We show that the approximation algorithm outperforms the other two under different network parameters, while all three effectively reducing the total energy consumptions of data access in data centers.**

Keywords – File Replication, Approximation Algorithms, Cloud Data Centers, Energy-Efficiency

I. Introduction

Cloud computing, which provides computing applications, platforms, and infrastructures as services, has emerged as a popular and mainstream technology in today's IT industry. The current cloud data centers, such as Amazon EC2 and Microsoft Azure, support a large number of Internet applications including social networks, video streaming, and search engines. The cloud-based data centers enable individual and business users to easily obtain aforesaid services with pay-as-you-go manner, thus saving the cost of maintaining their own compute infrastructure.

All above applications process extreme large amount of data [17]. When users submit jobs to cloud data centers for processing, virtual machines are allocated to execute corresponding application programs, which process the large amount of data. A virtual machine (VM) running on top of physical machine (PM) is an OS environment with its dedicated resources such as CPU cycles, memory, and bandwidth, and is isolated from other parts of the PM. Such isolation enables multiple OS environments on the same PM, allowing that applications previously running on multiple PMs to be consolidated into a single PM. With virtualization, a cloud data center is able to allocate and utilize its resources more efficiently and provide services to user applications in an effective manner.

The execution of the user applications needs the input data of the application available locally for its allocated VM. Therefore how to efficiently locate and access the data for the VMs becomes very important in data centers. Meanwhile, power consumption is still one of the biggest concerns in any data center [15]. Consequently, in cloud data center with thousands of PMs and switches and hundreds of thousands of network links, data file access could consume large amount of energy power in data center.

Data replication, which brings data files closer to the computing VMs, is an effective strategy that reduces the data access latencies and bandwidth consumption, thus saving energy in data centers. There have been a few research that employ data replication techniques to reduce the energy consumption [2], [3], [9], [5], data access delay [2], [3], [13], as well as achieving fault tolerance [8] in data centers. However, almost all of them design heuristic algorithms that do not offer any performance guarantee. Consequently, it is not clearly how performance improvement can be achieved all the time with those heuristic algorithms. In contrast, we design a time-efficient approximation algorithm with performance guarantee. We prove that our data replication algorithm reduces the total energy consumption of data access in data center by at least half of that achieved by an optimal replication solution. Based on a novel concept called *profit*, our algorithm optimizes over a submodular function that can be computed efficiently. We also design two other energy- and time- efficient heuristic data replication algorithms based on the access patterns of pods and PMs in the data centers. We show that the approximation algorithm outperforms the other two under different network parameters, while all three effectively reducing the total energy consumptions of data access in data centers.

Paper Organization. The rest of the paper is organized as follows. Section II gives an overview of the related literature and introduces the fat-tree data center topology adopted in this paper (however, our algorithms are designed for general graphs, therefore are applicable to any data center topologies). In Section III, we formulate the FRP problem. Section IV presents the different algorithms for FRP, including the approximation data replication algorithm and two heuristic algorithms. In Section V, we compare all the proposed algorithms and discuss the results in details. We conclude the paper and discuss possible future work in Section VI.

II. Background

Related Work. Ping et al. [13] was one of the first that proposed to replicate data across data centers. Their proposed data replica placement algorithm can efficiently achieve near optimal data access delay. The location of replicas for each data object is determined by periodically processing a log of recent data accesses, and by employing a weighted k-means clustering of user locations and deploying replica closer to the centroid of each cluster.

Li et al. [8] proposed a replication-based reliability model, which analyzes data storage failures and data loss probability to determine where to create replica copies. Dong et al. [5] proposed replication strategy to minimize power consumption in the backbone network across multiple data centers. They formulated the problem as linear programming and determined optimal points of replication based on the data center traffic demands and popularity of data objects. Boru et al. [2], [3] proposed a data replication technique for cloud computing data centers for joint optimization of energy consumption and bandwidth capacity of data centers as well as inside each datacenter. Lin et al. [9] proposed a replication placement scheme called eStor, under which data was placed in a constrained layout. Some replicas are placed in a sequential way, while other replicas are placed in a random fashion. eStore allows users to configure the replication level and number of replicas, and turn off some nodes without data loss.

However, almost all above research does not provide any performance guarantee for the energy consumption incurred during the data file replication. In current cloud data centers wherein user data and applications are massive and complex while energy consumption is enormous, it is important that the designed file replication algorithm can provide provable guarantee therefore it calls for new replication algorithms. In this paper, we propose a time-efficient approximation algorithm with provable performance guarantee for energy consumption in file replication. Using a novel concept called *profit*, we prove that our algorithm obtains the profit by at least half of what is achieved by an optimal algorithm. Note that the concept of profit in this work shares the spirit of the concept of profit in economics, which refers to the net gain of taking certain economic activities [10].

Cloud Data Center Topology. We adopt the fat-tree network [1] as the cloud data center topology, as it is widely used in data centers to interconnect commodity Ethernet switches. However, the FRP and its algorithms are applicable to any types of data center topologies. A k-ary fat-tree is shown in Fig. 1 with $k = 4$, where k is the number of ports of each switch. There are three layers of switches: edge switch, aggregation switch and core switch from bottom to top. The lower two layers are separated into k pods. A *pod* is a modular unit of compute, storage, and networking resources that works as a unit in data center. Each pod contains $k/2$ aggregation switches and $k/2$ edge switches, which form a complete bipartite graph in between. Each edge switch is directly connected to $k/2$ PMs; and each of its remaining $k/2$ ports is connected to each of the $k/2$ aggregation switches from the same pod. There are $\frac{k^2}{4}$ k-port core switches, each of which is connected to each of k pods. In general, a k-ary fat-tree data center contains $\frac{k^3}{4}$ PMs.

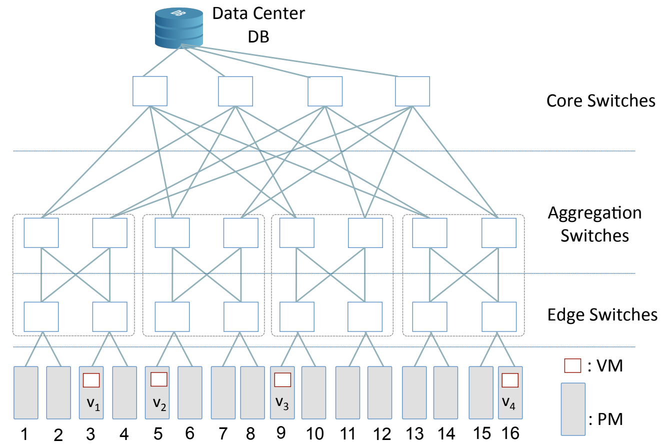

Fig. 1. A k-ary fat tree data center with database (Data Center DB), which stores all the data files in the data center. Here, $k = 4$ and it has 16 PMs. There are four VMs that execute users' submitted jobs.

The data center has its own database called *Data Center DB*, as shown in Fig. 1. The Data Center DB stores all the data files that are needed by the user applications running on this data center. It is connected to all the core switches. This applies to applications such as search engine wherein information is only queried by users, and is in consistent with the data center layout proposed in [2], [3]. However, our problem formulation and solutions work for a more general scenario, wherein the data files are initially produced and placed on any PMs. This applies to applications such as social networking where information is generated by users. Since in both scenarios, the data files are read much more frequently than updated, we assume that data replicas need not be updated.

III. File Replication Problem (FRP) in Data Center

System Model. We model a cloud data center as a graph $G(V, E)$, where $V = V_p \cup V_s$ includes the set of PMs V_p and the set of (edge, aggregate, and core) switches V_s. Each edge in E connects either one switch to another switch or a switch to a PM. We refer to the edges between core switches and aggregation switches as *core network links*, between aggregation switches and edge switches as *aggregation network links*, and between edge switches and PMs as *access network links*. Without loss of generality, let $V_p = \{1, 2, ..., |V_p|\}$, and $V_s = \{|V_p| + 1, |V_p| + 2, ..., |V|\}$. There are l data files $F = \{f_1, f_2, ..., f_l\}$ in the data center, where data file f_j

$(1 \le j \le l)$ is originally produced and stored at its *source PM* $S_j \in V_p$. The size of f_j is s_j. Note that a PM can be the source PM of multiple data files. Let m_i be the storage capacity of PM i.

TABLE I
NOTATION SUMMARY

Notation	Explanation
V_p	The set of physical machines (PMs) in the data center
V_s	The set of switches in the data center
m_i	The storage capacity of PM i
e_{ij}	The energy cost between PM (or switch) i and j
F	The set of l files f_j $(1 \le j \le l)$ in the data center
s_j	The size of file f_j
a_{ij}	The request frequency of PM i towards f_j
F_i	The set of files that is finally stored at PM i
$s(F_i)$	The size of the set of files that is stored at PM i
A_{ij}	PM i's access PM for file f_j
$\mathcal{E}(\mathcal{F})$	The energy consumption of data access for $\mathcal{F} = \{F_1, F_2, ..., F_{\lvert V_p \rvert}\}$
n	The number of jobs submitted to the data center
n_i	The number of jobs allocated to PM i
F_{ik}	The set of files needed by k^{th} job on PM i

There are n user jobs that are submitted to the cloud data center, and the VMs in PMs are allocated to process these jobs. Suppose that PM i is allocated n_i jobs $\{t_{i1}, t_{i2}, ..., t_{in_i}\}$, wherein job t_{ik} $(1 \le k \le n_i)$ requires some of the data files $F_{ik} \subseteq F$ as input files for execution. $\sum_{i=1}^{|V_p|} n_i = n$. Let a_{ij} be the number of times that PM i needs to access data file f_j in order to execute all its n_i jobs. That is, $a_{ij} = \sum_{k=1}^{n_i} x_k$, where $x_k = 1$ if $f_j \in F_{ik}$ and $x_k = 0$ otherwise. a_{ij} is also referred to as the *request frequency* of PM i to file f_j. A file with a larger request frequency therefore needs to be brought closer to the PMs that need them the most, which can be achieved by our data replication algorithm proposed in Section IV.

Energy Model. We measure the power consumption of one time access of data file f_j from PM i as the minimum number of switches existing between PM i and S_j, the source PM of f_j. This is in accordance to the finding made by Meng et al. [11], which observes that the energy consumption of communication inside data center is proportional to the number of switches the communication traverses. However, our problem and algorithm can be easily adjusted to accommodate the scenario that different switches consumes different amount of energy (for example, high-end core switches consume more power than aggregation and edge switches.).

Let e_{ij} denote the energy consumption between any two nodes (switches or PMs) $i \in V$ and $j \in V$. First, we calculate the total energy consumption in the data center to execute all the jobs without any data replication, which is the sum of energy consumption of each PM accessing each data file from its source PM. Denote it as $\mathcal{E}_{init}$, we have

$$\mathcal{E}_{init} = \sum_{i=1}^{|V_p|} \sum_{j=1}^{l} a_{ij} \cdot e_{iS_j}. \tag{1}$$

Problem Formulation. The objective of the FRP is to minimize the total energy consumption of data access in the data center by replicating data files into different PMs while satisfying the storage capacity of each PM. Let's give the following definitions and notations.

File Sets and Set of File Sets. Define *file set* of a PM as the set of data files that this PM stores (including the initial files it stores as a source PM). For PM i, let $F_i \subseteq F$ denote its file set, and let $s(F_i) = \sum_{f_j \in F_i} s(f_j)$ denote the total size of data files in F_i. Let $\mathcal{F} = \{F_1, F_2, ..., F_{|V_p|}\}$ denote the set of file sets. Initially, F_i is the set of files that have PM i as source PMs. That is, $F_i = \bigcup_{1 \le j \le l} x_i$, $1 \le i \le |V_p|$, where

$$x_i = \begin{cases} \{f_j\} & \text{if } (i == S_j), \\ \phi \text{ (empty set)} & \text{otherwise.} \end{cases}$$

We denote the above initial file set of each PM and the set of file sets as F_i^{init} $(1 \le i \le |V_p|)$ and $\mathcal{F}^{init} = \{F_1^{init}, F_2^{init}, ..., F_{|V_p|}^{init}\}$, respectively. Table I shows all the notations used in the paper.

Energy Consumption of Data Access in Data Center. With replication, multiple copies of the same data file can exist in the data center. For the purpose of energy saving, each PM accesses the copy that incurs the smallest amount of energy. Given any $\mathcal{F}$ and any PM i, we refer to the PM that stores a copy of f_j that i can access f_j with smallest amount of energy as i's *access PM* for f_j, and denote it as $A_{ij}(\mathcal{F})$. That is,

$$A_{ij}(\mathcal{F}) = \arg\min_k (e_{ik}, \text{where } j \in F_k).$$

Given any $\mathcal{F} = \{F_1, F_2, ..., F_{|V_p|}\}$, the minimum energy consumption of data access in data center is therefore

$$\mathcal{E}(\mathcal{F}) = \sum_{i=1}^{|V_p|} \sum_{j=1}^{l} a_{ij} \cdot e_{iA_{ij}(\mathcal{F})}. \tag{2}$$

$\mathcal{E}_{init}$ in Equation 1 can then be rewritten as $\mathcal{E}(\mathcal{F}^{init})$.

Objective of FRP. The objective of FRP is to select a set of $|V_p|$ file sets $\mathcal{F} = \{F_1, F_2, ..., F_{|V_p|}\}$, such that the *minimum total energy consumption of data access in data center*

$$\mathcal{E}_{min} = \min_{\mathcal{F}} \mathcal{E}(\mathcal{F}) \tag{3}$$

can be achieved under the storage capacity constraint that

$$s(F_i) = \sum_{f_j \in F_i} s_j \le m_i, \forall i \in V_p.$$

The FRP is NP-hard [7], [14]. Below we design time-efficient approximation algorithm as well as heuristic algorithms to solve it.

IV. Algorithms for FRP

A. An Approximation Algorithm.

Definition 1: (Profit of Replicating file f_j at PM i under $\mathcal{F}$, $\Delta\mathcal{E}(\mathcal{F}, f_j, i)$) The profit of replicating file f_j at PM i under $\mathcal{F} = \{F_1, F_2, ..., F_{|V_p|}\}$, denoted as $\Delta\mathcal{E}(\mathcal{F}, f_j, i)$, is the reduction of total energy cost in the data center when placing a copy of f_j at PM i divided by the size of f_j $s(f_j)$, given that the current set of file sets is $\mathcal{F}$. Let $\mathcal{F}' = \{F_1, F_2, ..., F_{i-1}, F_i \cup \{f_j\}, F_{i+1}, ..., F_{|V_p|}\}$. Then, $\Delta\mathcal{E}(\mathcal{F}, f_j, i) = \big(\mathcal{E}(\mathcal{F}) - \mathcal{E}(\mathcal{F}')\big)/s(f_j)$. □

Obviously, in above definition, if $f_j \in F_i$, i.e., a copy of f_j is already located at PM i, then $\Delta\mathcal{E}(\mathcal{F}, f_j, i) = 0$. The intuition behind the "profit" is that replicating a file into a PM is more profitable if this reduces more energy consumption of file access in the data center as well as the file has a smaller size (so that less storage space of a PM it occupies). We therefore should choose a file-PM pair for replication that achieves the maximum reduction of energy consumption while costing least amount of storage space for the replicated file.

Algorithm 1 below is our "profit"-based greedy algorithm, which takes place in rounds. In each round, it decides that by replicating which file at which PM, it can reduce the total energy of data access the most (Line 5-15). Here we refer to such a file and PM in that round as *target file* and *target PM*, respectively. This continues until either there is no storage space available at any PMs for file replication, or it can no longer reduce the total energy energy by replication (Line 1). Let's denote the set of file sets produced by Algorithm 1 as $\mathcal{F}^g = \{F_1^g, F_2^g, ..., F_{|V_p|}^g\}$.

Algorithm 1: Data Replication Algorithm.
Input: A data center $G(V, E)$ with l data files and n jobs.
Output: $\mathcal{F}^g = \{F_1^g, F_2^g, ..., F_{|V_p|}^g\}$ and $\mathcal{E}(\mathcal{F}^g)$.
Notations:
  p: the target PM in each round
  f: the target file in each round
  $profit$: the profit of placing f at p in each round
0. **Calculate initial energy consumption and sets of file sets before replication**:
  Find F_i^{init}, $1 \le i \le |V_p|$;
  $\mathcal{F}^{init} = \{F_1^{init}, F_2^{init}, ..., F_{|V_p|}^{init}\}$;
  Calculate $\mathcal{E}(\mathcal{F}^{init})$;
  $F_i^g = F_i^{init}$, $1 \le i \le |V_p|$;
  $\mathcal{F}^g = \{F_1^g, F_2^g, ..., F_{|V_p|}^g\}$;
  $s_{min} = \min_{1 \le j \le l} s(f_j)$;
1. **while** $\big(\exists k$, $1 \le k \le |V_p|$, s.t. $m_k - s(F_k^g) \ge s_{min}\big)$
2. $p = -1$;
3. $f = -1$;
4. $profit = 0$;
5.   **for** ($i = 1$ to $|V_p|$)
6.   (**for** $j = 1$ to l)
7.   **if** ($f_j \notin F_i^g$ and $s(F_i^g) + s(f_j) \le m_i$)
8.   **if** ($\Delta\mathcal{E}(\mathcal{F}^g, f_j, i) > profit$)
9. $profit = \Delta\mathcal{E}(\mathcal{F}^g, f_j, i)$;
10. $p = i$;
11. $f = f_j$;
12.   **end if;**
13.   **end if;**
14.   **end for;**
15.   **end for;**
16.   **if** ($p == -1$)
17.   break;
18.   **end if;**
19. $F_p^g = F_p^g \cup \{f\}$; /* Update PM p's file set*/
20. $\mathcal{E}(\mathcal{F}^g) = \mathcal{E}(\mathcal{F}^g) - profit$; /* Update energy */
21. **end while;**
22. **RETURN** $\mathcal{F}^g = \{F_1^g, F_2^g, ..., F_{|V_p|}^g\}$ and $\mathcal{E}(\mathcal{F}^g)$.

Time Complexity of Algorithm 1. The initialization stage (Line 0) takes $O(|V_p|^3 + |V_p| \cdot l)$, as finding minimum energy consumption between any two PMs takes $O(|V_p|^3)$, and calculating the total energy consumption without replication (Equation 1) takes $|V_p| \cdot l$. The while loop (Line 1) takes about $\frac{\sum_{1 \le i \le |V_p|} m_i}{\sum_{1 \le j \le l} s(f_j)/l}$ rounds, which can be upper-bounded by $|V_p| \cdot \bar{m}$ with $\bar{m}$ being the average storage capacity of a PM. Each round takes at most $(|V_p|^2 \cdot l)$, since it iterates over all PM-file pairs to decide which file is replicated into which PM, and it takes $O(|V_p|)$ to calculate $\Delta\mathcal{E}$. Therefore, the time complexity of Algorithm 1 is $O(|V_p|^3 + |V_p| \cdot l + |V_p| \cdot \bar{m} \cdot |V_p|^2 \cdot l)$ = $O(|V_p|^3 \cdot \bar{m} \cdot l)$.

Submodularity. A set function $\phi : 2^U \to N$ is called *submodular* if for every $A \subseteq B \subseteq U$ and $e \in U - B$ it holds that

$$\phi(A \cup \{e\}) - \phi(A) \ge \phi(B \cup \{e\}) - \phi(B).$$

Next we prove that $\mathcal{E}(\mathcal{F})$ is submodular when all the files have the same unit size.

Theorem 1: $\mathcal{E}(\mathcal{F})$ is submodular when $s_{j_1} = s_{j_2} = 1$ $\forall$ $1 \le j_1 \ne j_2 \le l$.

Proof: In each round of Algorithm 1, it selects a data file f_j and places a copy of it into the storage of PM i. It is equivalent to say that a variable D_{ijk} is selected, where $1 \le i \le |V_p|$, $1 \le j \le l$, and $1 \le k \le m_i$, which indicates that f_j is placed in the k^{th} storage slot of PM i. Therefore, Algorithm 1 essentially selects a sequence of such variables. Then we can rewrite $\mathcal{E}(\mathcal{F})$ as $\mathcal{E}(A)$, where A is the set of variables selected so far. Next we prove that $\mathcal{E}(A)$ is submodular.

Let U be the entire set of variables selected after the algorithm, and let $A \subseteq B \subseteq U$. Let $D_{ijk} \in U - B$. Since $\mathcal{E}(A)$ is a minimization function, we need to show that $\mathcal{E}(A) - \mathcal{E}(A \cup \{D_{ijk}\}) \ge \mathcal{E}(B) - \mathcal{E}(B \cup \{D_{ijk}\})$.

Let $\mathcal{E}_j(A)$ denote the total energy consumption accessing f_j after A is selected. Since D_{ijk} can only possibly affect the energy consumption accessing f_j, we only need to show that $\mathcal{E}_j(A) - \mathcal{E}_j(A \cup \{D_{ijk}\}) \ge \mathcal{E}_j(B) - \mathcal{E}_j(B \cup \{D_{ijk}\})$. This is indeed true since in each round of Algorithm 1, it finds the PM-file pair that reduces the energy consumption the most. ■

Next we show that Algorithm 1 delivers a solution whose total "profit" is at least one half of the optimal "profit". The proof technique used below is similar to that used in [12] for a closely related problem of data replication in data grid scientific applications.

Theorem 2: Given any instance of FRP, let $\mathcal{E}_{init}$ be the total energy consumption of data access without replication, $\mathcal{E}_{min}$ be the optimal total energy consumption of data access with replication, and $\mathcal{E}_g$ be the total energy consumption of data access given by Algorithm 1. We have

$$\frac{\mathcal{E}_{init} - \mathcal{E}_g}{\mathcal{E}_{init} - \mathcal{E}_{min}} > \frac{1}{2}$$

when all the files have the same unit size.

Proof: Let L be the total number of rounds in Algorithm 1. And let the sequence of selections in Algorithm 1 be $\{n^g_1 f^g_1, n^g_2 f^g_2, ..., n^g_L f^g_L\}$, with $n^g_i f^g_i$ indicating that at round i, data file f^g_i is replicated at PM n^g_i. Let the optimal sequence of selections be $\{n^o_1 f^o_1, n^o_2 f^o_2, ..., n^o_L f^o_L\}$, with $n^o_i f^o_i$ indicating that at round i, data file f^o_i is replicated at site n^o_i. Let $O = \mathcal{E}_{init} - \mathcal{E}_{min}$ and $C = \mathcal{E}_{init} - \mathcal{E}_g$ be the profit from optimal algorithm and Algorithm 1 respectively.

Consider a new data center graph G', where the storage capacity of each PM i is changed from m_i to $2m_i$. For each PM i, let its first m_i storage slots store the data files obtained in Algorithm 1, and its second m_i storage slots store the data files selected in optimal algorithm. Now we calculate the profit O' for G'. $O' \geq O$, because each site in G' stores extra data files beyond the data files stored in the same PM in G.

Let the sequence of selections in G' be $\{n^g_1 f^g_1, n^g_2 f^g_2, ..., n^g_L f^g_L, n^o_1 f^o_1, n^o_2 f^o_2, ..., n^o_L f^o_L\}$. The profit after the first L selections is C. For the second L selections, we need to calculate the profit when adding $n^o_i f^o_i$ $(1 \leq i \leq L)$ on $\{n^g_1 f^g_1, n^g_2 f^g_2, ..., n^g_L f^g_L, n^o_1 f^o_1, n^o_2 f^o_2, ..., n^o_{i-1} f^o_{i-1}\}$. From Theorem 1, it is less than the profit when adding $n^o_i f^o_i$ on $\{n^g_1 f^g_1, n^g_2 f^g_2, ..., n^g_{i-1} f^g_{i-1}\}$. According to Algorithm 1, the latter is less than the profit due to the addition of $n^g_i f^g_i$ based on the same sequence of $\{n^g_1 f^g_1, n^g_2 f^g_2, ..., n^g_{i-1} f^g_{i-1}\}$. Thus the sum of the profits due to selection of $\{n^o_1 f^o_1, n^o_2 f^o_2, ..., n^o_L f^o_L\}$ is less than or equal to C too. Therefore O is less than or equal to O', which is less than or equal to 2 times of C. ∎

B. Heuristic Algorithms.

We further propose two other time-efficient heuristic file replication algorithms, and compare them with the approximation algorithm via simulations.

Local Greedy Algorithm. In Local Greedy, it replicates each PM's most frequently requested data files in its local storage. That is, for PM i with m_i storage capacity, it places the m_i data files (out of the l files) that have the highest request frequencies by PM i. Using a heap, finding the top m_i files from l files take $O(l + m_i \cdot \log l)$. Therefore it takes $O(|V_p| \cdot (l + \bar{m} \cdot \log l))$ for all the $|V_p|$ PMs, where $\bar{m}$ is the average storage capacity of a PM. After this replication, calculating the total energy cost is $|V_p| \cdot l \cdot |V_p| \cdot \bar{m}$, since for each PM-file pair, finding a copy of this file that is closest to the PM takes $|V_p| \cdot \bar{m}$ time. Therefore the time complexity for the Local Greedy is $O(|V_p|^2 \cdot \bar{m} \cdot l)$.

Pod-Based Greedy Algorithm. In this algorithm, it first finds the aggregate request frequency of each file in each pod, which is the sum of the request frequencies of all the PMs in this pod for that data file. Then in each pod, it replicates the data files with the highest aggregate request frequency that are allowed by the total storage capacity of that pod. Specifically, we start with the file with the highest aggregate frequency, and place a copy of it to the PM that has the highest request frequency to it. If this PM is full, it tries the PM with the second highest request frequency to this file, etc. This finishes until all those data files are placed into the pod. Finding the aggregate request frequencies takes $O(|V_p| \cdot l)$, placing replica copies of data files into all the pods takes $O(|V_p| \cdot l)$, and calculating the total energy cost is $|V_p| \cdot l \cdot |V_p| \cdot \bar{m}$. Therefore the time complexity for the Pod-Based Greedy is $O(|V_p|^2 \cdot \bar{m} \cdot l)$.

V. Performance Evaluation

Simulation Setting. In this section, we compare the performances of the three file replication algorithms. We refer to our approximation algorithm as **Profit**, the pod-based greedy algorithm as **Pod**, and the local greedy algorithm as **Local**. We generate fat-tree data centers of different sizes: $k = 8$, a small data center with 128 PMs; and $k = 16$, a large data center with 1024 PMs. The size of each data file and its replica copies is 2 GB. The storage capacity of each PM is varied from 100GB to 500GB. There are 1000 data files that are either located in the central database of the cloud data center (referred to as *Central DB*), or are randomly placed on the PMs (referred to as *Random Placement*). Each data point is the average of five simulation runs.

Energy Consumption Models. We use r_e, r_a, and r_c to denote the power consumption of transmitting one data file copy on the edge, aggregate, and core switches respectively. We consider two energy consumption models that are currently adopted in cloud data center research:

- In *uniform energy model*, the energy consumption of data access is measured as number of switches the data traverses [11]. In this model, $r_e = r_a = r_c = 1$.
- In *skewed energy model*, the core switches handle huge amount of traffic across the entire data center, therefore consuming more energy power than aggregate switches, which consume more energy power than edge switches [2]. Core links usually have higher bandwidth than aggregation links, which have higher bandwidth than access links. In this model, we set $r_e = 1$, $r_a = 5$, and $r_c = 10$.

Data File Access Pattern. We adopt two data file access patterns to characterize the request frequencies of data files.

- In *Zipf distribution*, the request frequency to access the j^{th} $(1 \leq j \leq l)$ popular data file is represented by $P_j = \frac{1}{j^\theta \sum_{k=1}^{l} 1/k^\theta}$. We choose θ to be 0.6 based on the real trace studies collected at Facebook data center [6], [16].
- In *random access*, the request frequency of each file by each PM is a random number between 0 and 100.

Performance Comparison Under Uniform and Skewed Energy Models. Fig. 2 and Fig. 3 show the total energy consumption of the three algorithms by varying the storage capacity of each PM, under uniform and skewed energy models, respectively. It shows that all three replication algorithms effectively reduce the total energy consumption of file access

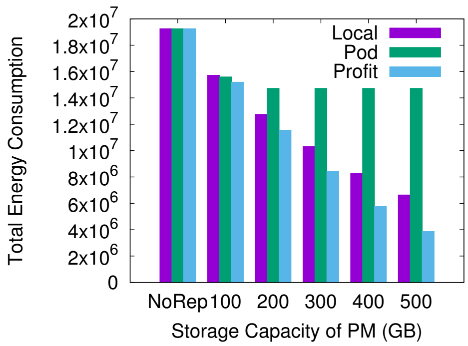

Fig. 2. Performance comparison for data center with 128 PMs under uniform energy model. "NoRep" indicates the total energy consunmption without any replication. Here, we adopt random access pattern and central DB.

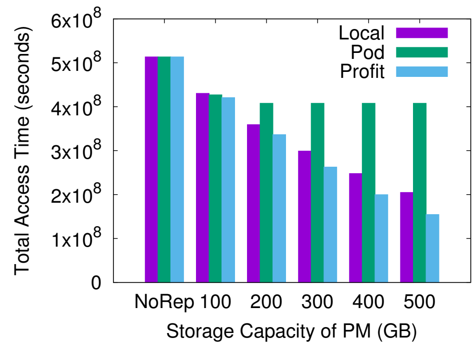

Fig. 4. Performance comparison for data center with 128 PMs with CloudSim. Here, we adopt random access pattern and central DB. The bandwidth of each link is 100MB/s.

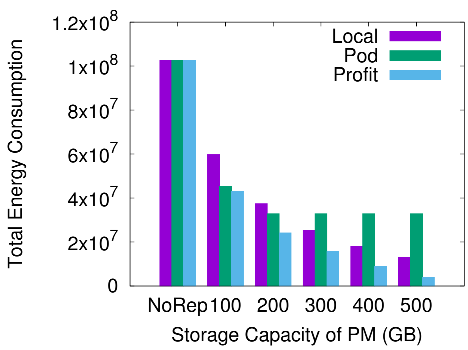

Fig. 3. Performance comparison for data center with 128 PMs under skewed energy model. Here, we adopt random access pattern and central DB.

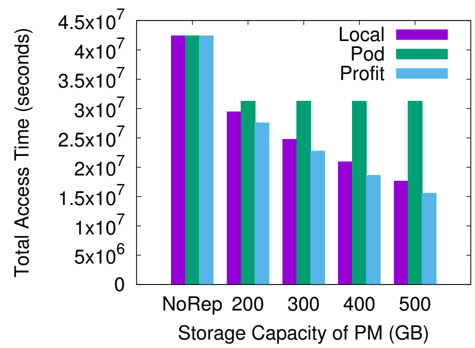

Fig. 5. Performance comparison for data center with 128 PMs, with random access pattern and random initial placement. The bandwidth of each access link is 1GB, each aggregation link is 2GB, and each core link is 5GB.

in the data center. Profit outperforms Local and Pod in the entire parameter range under both energy models. We also observe that all three algorithms perform better under skewed energy model than under uniform energy model by reducing more energy consumptions. This is because in skewed energy model, core switches cost more energy than aggregation and edge switches. By storing the replica copies at local PMs, access traffic does not go through core switches often, therefore reducing energy consumption more in skewed energy model than in uniform energy model. Finally, we observe that under each energy model, the energy consumption by all three algorithms decrease with the increase of storage capacity in most cases, except for Pod when storage capacity exceeds 200 GB. Under Pod, each pod continues storing only one copy of each data file with the increasing of storage, therefore keeping the energy consumption the same.

Performance Comparison Using CloudSim [4]. For the rest of simulations, we use CloudSim, one of the most popular open source cloud simulators in the research and academia. We set the link bandwidth as 100MB/s in CloudSim, and measure the total access time of data files yielded by the three algorithms, as shown in Fig 4. We observe that Profit performs better than Local, which outperforms Pod. In particular, when the storage is large (500 GB), Profit can reduce the total access time of the data center by roughly half via replication. Note that under CloudSim, both uniform and skewed energy models perform the same, since the access time only depends on file sizes and link bandwidth.

Random Placement of Data Files. All simulations so far assume the availability of a central DB, where all the data files are initially stored. Next we study the effects of the random initial placement of data files. We set the bandwidth of the access links as 1GB, the aggregation links as 2GB, and core links as 5GB. Fig. 5 shows that the total access cost of different algorithms. The performance comparison of the three algorithms stays the same as in the central DB. However, the costs are much smaller than those in Fig. 4, since the links have much higher bandwidth.

Study of Scalability. We study the performances of the three algorithms in larger data center of 1024 PMs, in order to understand their scalability, as shown in Fig. 6. All the set up is the same as in Fig. 5, except for the size of the data center. We only show the total energy consumptions of Profit and Local in Table II, since both outperform Pod. We set the storage capacity of each PM as 300GB, the medium in the storage parameter range. The last column, *Improvement Percentage*, is calculated as the energy consumption difference between Profit and Local divided by energy consumption of Local. It shows that in small data center, Profit improves upon Greedy

TABLE II
STUDY OF SCALABILITY. STORAGE CAPACITY = 300GB.

Data Center Size	Local	Profit	Improvement Percentage (%)
128	24740267.60	22747692.00	8.05
1024	201923359.5	174080000	13.78

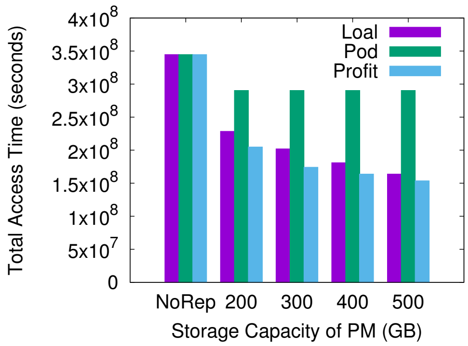

Fig. 6. Performance comparison for data center with 1024 PMs, with random access pattern and random initial placement. The bandwidth of each access link is 1GB, each aggregation link is 2GB, and each core link is 5GB.

by 8.05% while in large data center, it is 13.78% improvement. This shows that our approximation algorithm performs better than Local in large data centers, therefore is more scalable.

Zipf Distribution. Finally we study the performance of the proposed algorithms under Zipf distribution access pattern. Fig. 7 shows that under Zipf distribution, the performance difference between Profit and Local is even larger. This shows that Profit works particularly well for data files with distinct popularity levels. When data files have distinct popularity levels, the popular data is always be replicated to more PMs according to our approximation algorithm, therefore reducing the energy consumption the most.

VI. Conclusion and Future Work

We studied file replication problem in data intensive cloud data centers, and designed a time-efficient approximation algorithm with performance guarantee. It was based on a novel concept called "profit", and optimizes over a submodular function that can be computed efficiently. Our algorithm reduces the total energy consumption of data access by at least half of what is achieved by an optimal replication solution. We also designed two energy- and time-efficient heuristic file replication algorithms. Currently, we assume that the VMs that execute user jobs stay in a particular PM for its entire lifetime. As future work, it would be interesting to investigate how dynamics of VM migration can interplay with the file replication, to better achieve the energy efficiency in cloud data centers.

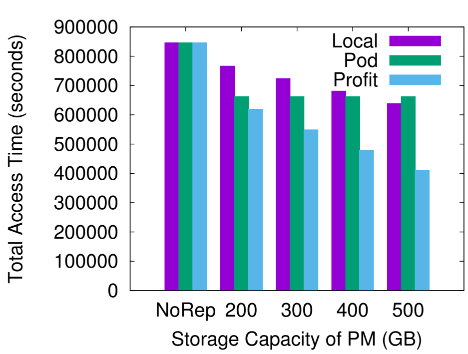

Fig. 7. Performance comparison for data center with 128 PMs with Zipf distribution access pattern. The bandwidth of each access link is 1GB, each aggregation link is 2GB, and each core link is 5GB.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.

[2] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster Computing*, 18(1):385–402, 2015.

[3] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya. Models for efficient data replication in cloud computing datacenters. In *Proc. of the IEEE ICC*, 2015.

[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience (SPE)*, 41(1):23–50, 2011.

[5] X. Dong, T. El-Gorashi, and J. M. H. Elmirghani. Green ip over wdm networks with data centers. *JOURNAL OF LIGHTWAVE TECHNOLOGY*, 29(12):1861–1880, 2011.

[6] L. Durbeck, N. Macias, and J. Tront. Energy efficiency of zipf traffic distributions within facebook's data center fabric architecture. In *Proc. of 25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2015.

[7] M. Karlsson and C. Karamanolis. Choosing replica placement heuristics for wide-area systems. In *Proc. of the IEEE ICDCS*, 2004.

[8] W. Li, Y. Yang, and D. Yuan. A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. In *Proc. of the International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 2011.

[9] B. Lin, S. Li, X. Liao, Q. Wu, and S. Yang. estor: energy efficient and resilient data center storage. In *Proc. of the International Conference on Cloud and Service Computing (CSC)*, 2011.

[10] N. G. Mankiw. *Principle of Economics*. South-Western Cengage Learning, 2011.

[11] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. of IEEE INFOCOM*, 2010.

[12] D. T. Nukarapu, B. Tang, L. Wang, and S. Lu. Data replication in data intensive scientific applications with performance guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 22(8):1299–1306, 2011.

[13] F. Ping, X. Li, C. McConnell, R. Vabbalareddy, and J. H. Hwang. Towards optimal data replication across data centers. In *Proc. of the International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2011.

[14] L. Qiu, V. Padmanabhan, and G. Voelker. On the placement of web server replicas. In *Proc. of the IEEE INFOCOM*, 2001.

[15] V.K. Mohan Raj and R. Shriram. Power management in virtualized datacenter - a survey. *Journal of Network and Computer Applications*, 69:117–133, 2016.

[16] N. Sharma, S. Barker, D. Irwin, and P. Shenoy. Blink: Managing server clusters on intermittent power. *SIGARCH Comput. Archit. News*, 39(1):185–198, 2011.

[17] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos. A survey on data center networking for cloud computing. *Computer Networks*, 91:528–547, 2015.