Tsinghua Science and Technology

Volume 21 | Issue 5 Article 1

2016

Efficient Location-Aware Data Placement for Data-Intensive Applications in Geo-distributed Scientific Data Centers

Jinghui Zhang
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China.

Jian Chen
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China.

Junzhou Luo
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China.

Aibo Song
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/11 pp471-481
Volume 21, Number 5, October 2016

Efficient Location-Aware Data Placement for Data-Intensive Applications in Geo-distributed Scientific Data Centers

Jinghui Zhang*, Jian Chen, Junzhou Luo, and Aibo Song

Abstract: Recent developments in cloud computing and big data have spurred the emergence of data-intensive applications for which massive scientific datasets are stored in globally distributed scientific data centers that have a high frequency of data access by scientists worldwide. Multiple associated data items distributed in different scientific data centers may be requested for one data processing task, and data placement decisions must respect the storage capacity limits of the scientific data centers. Therefore, the optimization of data access cost in the placement of data items in globally distributed scientific data centers has become an increasingly important goal. Existing data placement approaches for geo-distributed data items are insufficient because they either cannot cope with the cost incurred by the associated data access, or they overlook storage capacity limitations, which are a very practical constraint of scientific data centers. In this paper, inspired by applications in the field of high energy physics, we propose an integer-programming-based data placement model that addresses the above challenges as a Non-deterministic Polynomial-time (NP)-hard problem. In addition we use a Lagrangian relaxation based heuristics algorithm to obtain ideal data placement solutions. Our simulation results demonstrate that our algorithm is effective and significantly reduces overall data access cost.

Key words: data placement; geo-distributed; data center; Lagrangian relaxation

1 Introduction

In recent years, data intensive applications have gained much attention due to their strong impact in big-data-related scientific areas like high-energy physics, astronomy, medical surgery, and climate modeling[1, 2]. In scientific fields, many international research institutes participate in collaborative data sharing and analysis, and data-intensive scientific applications often involve large amounts of data stored in geographically distributed data centers across regions. The major motivation for the work in this paper is the Alpha Magnetic Spectrometer (AMS-02) experiment, as shown in Fig. 1, with which we have been involved for some years. Installed on the International Space Station (ISS) on May 19, 2011, the AMS-02 is a high-precision particle physics detector that detects cosmic nuclei, elementary charged particles, and gamma rays[1]. To date, the AMS-02 detector has collected more than 70 billion cosmic ray data events. After collection, these data are immediately transferred to ground data centers. Data centers that were built for specific scientific purposes generally have limits in the scale of their storage or computing resources due to policy or budgetary constraints. For instance, in Nanjing, China, Southeast University (SEU)'s regional data center for the AMS experiment, which comprises 500-TB disk array storage, is able to store less than 5% of the entire AMS dataset. Multiple AMS regional data centers, such as those located in the European Organization for Nuclear Research (CERN),

• Jinghui Zhang, Jian Chen, Junzhou Luo, and Aibo Song are with School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: jhzhang@seu.edu.cn; jianchen@seu.edu.cn; jluo@seu.edu.cn; absong@seu.edu.cn.

* To whom correspondence should be addressed.
Manuscript received: 2016-07-26; revised: 2016-08-04; accepted: 2016-08-22

Fig. 1 AMS-02 experiment at the International Space Station.

RWTH Aachen, University of Bologna, and Southeast University, have been built to jointly store the massive AMS science data sets.

Large-scale scientific data processing in globally distributed data centers requires a data management framework that optimizes data access cost by leveraging these distributed data centers. However, the new features characterizing massive amounts of scientific data bring with them many challenges:

- Large Volume: In this big data era, data centers must manage data scales from terabyte (TB) to petabyte (PB). For instance, in the field of high-energy physics, the AMS experiment produces 4 PB of raw, reconstruction, and simulation data every year. For large international collaborations like the AMS experiment, each regional scientific data center can hold only a portion of the total scientific data set due to their storage capacity constraints.
- Remote Data Serving: Each data center generates or stores different massive scientific data sets for different scientific purposes. For instance, in the AMS experiment, CERN stores 80% of the reconstruction data sets while SEU stores most of the simulation data sets and a small percentage of the reconstruction data. As a result, scientists at the SEU data center must request reconstruction data remotely located at CERN to perform their analysis. It is extremely expensive to transfer huge amounts of data between data centers located in different regions, so remote data serving, due to differences in the location between a data processing task and the necessary data, significantly increases the cost of data access.
- Access of Associated Data: A large data set may be comprised of a number of comparatively small but associated data items, and data processing tasks often take multiple associated data items as input. It is not atypical for a data processing task to involve data inputs spanning multiple data centers. For instance, a typical AMS analysis task often requires hundreds of associated data items, collected in the same run (e.g., a continuous, uninterrupted AMS data collection period at the ISS) but stored in multiple data centers, that must be simultaneously accessed and analyzed for validation purposes. Ready access to this associated data contributes to the high expense of data access.

With respect to data intensive scientific applications, we can make the following observations. First, a high frequency of data access requests from scientists located at different data centers can only be met by the data center storing the requested data by transferring these requested data to the data center initiating the request. Since remote data serving is extremely expensive, the data storage location strongly impacts the cost of data access. Second, it is often the case that multiple associated data items may be requested in one data processing transaction. The cost of accessing associated data[3, 4] incurred by the processing overhead at each data center and determined by the number of data centers involved in the storage of these associated data, must also be taken into account when modeling to realize data-access system efficiency. It is for this reason that the efficient placement of massive data items to the distributed data centers within the storage capacity limits of each data center is highly desirable, as is the nature of the data requests that vary across different data centers.

However, it is a challenging task to minimize data access cost when all the related factors are taken into account: remote data serving, associated data access, and each data center's storage capacity constraints. To the best of our knowledge, most existing data placement studies have considered either storing the requested data closer to the users or to the associated data access requests. Very few works have considered the typical storage capacities of data centers for scientific data processing. To address this data placement problem, in this paper we propose a novel data placement algorithm based on Lagrangian relaxation. More specifically, we make the following contributions:

- We formulate the data placement problem using

an integer programming model, wherein each data center's storage capacity is respected, and which optimizes the combined data access cost of remote data serving and of accessing the associated data.
- We propose Lagrangian relaxation based heuristics to efficiently solve this NP-hard data placement problem.
- We evaluate our data placement algorithm in a simulation to compare its average performance with those of its competitors.

This paper is organized as follows: In Section 2, we describe in detail the data placement problem and its integer programming model. In Section 3, we present our data placement algorithm, which we evaluate in Section 4. In Section 5, we discuss related work, and we draw our conclusions in Section 6 with a summary of our findings.

2 Problem Formulation

2.1 Workload and data placement modeling

Let set X of m data items denote the scientific data stored in the geo-distributed data centers, with each data item $x \in X$ occupying a storage space of size_x. At most, scientists may request d different items from the set X for each data processing activity. The request patterns space is represented by P and each actual request pattern $p \in P$. As shown in the example in Fig. 2, the problem inputs contain ten different data items and four request patterns: $(1, 2)$, $(3, 4, 5)$, $(6, 7, 8)$, and $(9, 10)$. Requests to access multiple data items simultaneously for one data processing activity are common. For instance, in the AMS experiment, the analysis results are usually obtained by processing multiple input data items, each of which may have been acquired from a different remote AMS regional data center. With a given mapping solution (in dashed line) between the data items and data centers, as shown in Fig. 2, it is easy to see that request for pattern $(3, 4, 5)$ from DC_2 comprises three data items and their storage locations, including DC_1 and DC_2. The set X of data items is stored in multiple geographically distributed data centers denoted by a set K of $|K|$ data centers. Each data item $x \in X$ is assumed to be stored at a unique data center $k \in K$ and occupies data storage space size_x. The $|K|$ data centers are heterogeneous in the sense that they have storage systems of different storage capacities. Without loss of generality, we assume data center $k \in K$ has a storage

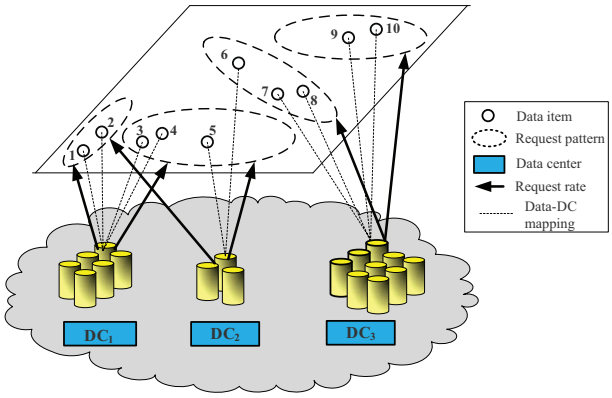


Fig. 2 Problem inputs: (1) data item, (2) request pattern, (3) data centers with storage capacity limit, and (4) request rate.

capacity limit C_k and the total size of all the data items placed in k can not exceed C_k. Therefore, the data-to-DC mapping function can be defined as $M : x \to k$, which designates data item x's storage location k. In this paper, we develop an efficient data placement algorithm that can give a reasonable solution for M.

We assume that a request for data items can be issued either by scientists attached to specific data centers or by data processing tasks being carried out within the data centers. Therefore, the data center is designed to have two simultaneous roles: the source location for data requests and as the destination location for holding requested data. We use R_{pk} to represent the request rate for each pattern $p \in P$ from the requesting data center $k \in K$. Here, we assume request rates to be predictable, which is similar to the practice in most existing studies[4].

We denote the request rate set as R. In the example shown in Fig. 2, the request rate set R is a set of weighted directed edges, where each edge's source is the data center k that initiates the request and each edge's target is the request pattern p. Each directed edge is weighted by the rate R_{pk}, and $R = \{R_{pk} | p \in P, k \in K\}$. For each data item x, its total request rate at each data center k can be calculated as follows:

$$R_{xk} = \sum_{p \in P} R_{pk} \cdot z_{xp} \tag{1}$$

where z_{xp} denotes whether or not pattern p includes the data item x, returning 1 if true and 0 otherwise. For each pattern p, the total request rate is calculated by the following:

$$R_p = \sum_{k \in K} R_{pk} \tag{2}$$

2.2 Data access cost modeling

Data placement can affect data access cost both with respect to accessing associated data and remote data serving.

2.2.1 Cost of accessing associated data

According to the observation made by Ref. [3], the average system cost of a request is influenced not only by the amount of requested information, i.e., the total size of data items to be accessed, but also by the number of data centers involved, due to the processing overhead associated with each data center. We use S_p to denote the number of data items included in request pattern p. As not all the data items of request p may be stored at a single data center k, we use S_{pk} to represent the number of items in p that are stored at k: $\sum_{k\in K} S_{pk} = S_p$. The data access cost of a request p at data center k can thus be modeled by

$$\sum_{x\in p}[\text{size}_x \cdot y_{pk}] + \alpha \cdot y_{pk} \tag{3}$$

where the 0-1 variable y_{pk} indicates whether $S_{pk} \geqslant 1$. In fact, the associated data access cost includes that related to the total volume of data items accessed $\sum_{x\in p}[\text{size}_x \cdot y_{pk}]$ and the overhead associated with handling the request, denoted by $\alpha \cdot y_{pk}$. The constant overhead α is brought by the routine operations performed when processing a request, e.g., establishing the network connection. With request rates of different patterns, the total data access cost to fulfill all the requests is

$$\sum_{k\in K}\sum_{p\in P} R_p[\sum_{x\in p}\text{size}_x \cdot y_{pk} + \alpha \cdot y_{pk}] \tag{4}$$

Placing strongly associated data in the same location minimizes this cost, e.g., placing all the items in a pattern p at the same data center achieves a lower bound cost $R_p(\text{size}_p + \alpha)$. For any given workload, $\sum_{k\in K}\sum_{p\in P} R_p \sum_{x\in p}\text{size}_x \cdot y_{pk} = R_p \cdot \text{size}_p$ is a constant. Therefore, the cost of accessing associated data can be denoted as follows:

$$\text{Cost}_1 = \sum_{k\in K}\sum_{p\in P} R_p \cdot \alpha \cdot y_{pk} \tag{5}$$

2.2.2 Cost of remote data serving

The cost of remote data serving stems from the location difference between the data center that initiates the request and the data center holding the requested data. When the serving data center is different from the requesting data center, requested data items must be transferred from the serving data center, so data traffic is generated accordingly. As the total amount of scientific data can be very large, this traffic may severely deteriorate the data-access system performance. Therefore, we include the total data traffic in the cost modeling of data access, as follows:

$$\text{Cost}_2 = \sum_{x\in X}\sum_{k\in K} R_{xk} \cdot \text{size}_x \cdot [1 - t_{xk}] \tag{6}$$

where R_{xk} represents the total request rate of data item x at data center k and size_x denotes the size of x. t_{xk} denotes if x is stored at data center k.

2.3 Optimization problem modeling

Our objective is to optimize the data access cost of a data placement M, represented by

$$\text{Cost} = \text{Cost}_1 + \beta \cdot \text{Cost}_2 \tag{7}$$

Where we use β as a tradeoff between the two metrics. The formulated problem can be generalized as follows: given the request pattern set, P, request rate set, R, data item set, X, and size of the data items, size, find the optimal data placement solution that minimizes the value of Cost, subject to data center K's storage capacity limit C.

To clearly formulate the data placement problem, we propose an Integer Programming (IP) model as follows:

$$\min\{\sum_{k\in K}\sum_{p\in P} R_p \cdot \alpha \cdot y_{pk} + \beta \sum_{x\in X}\sum_{k\in K} R_{xk} \cdot \text{size}_x \cdot [1 - t_{xk}]\} \tag{8}$$

subject to

$$\sum_{x\in X}\text{size}_x \cdot t_{xk} \leqslant C_k, \forall k \in K \tag{9}$$

$$\frac{\sum_{x\in p} t_{xk} \cdot \text{size}_x}{\text{size}_p} \leqslant y_{pk}, \forall p \in P, \forall k \in K \tag{10}$$

$$\sum_{k\in K} t_{xk} = 1, \forall x \in X \tag{11}$$

$$y_{pk} \in \{0, 1\}, \forall p \in P, \forall k \in K \tag{12}$$

$$t_{xk} \in \{0, 1\}, \forall x \in X, \forall k \in K \tag{13}$$

The 0-1 variable t_{xk} indicates whether x is stored at data center k, while the other 0-1 variable y_{pk} denotes whether any requested data item $x \in p$ is placed at data center k and can thus be fulfilled in k. As soon as a part of pattern p is placed in data center k (even a very small part, e.g., a single data item $x \in p$ or several data items), y_{pk} is equal to 1. Otherwise y_{pk} is equal to 0. So, y_{pk} is a bivalent variable, equal to 0

or 1 (see Constraint (12)) that detects the presence of data item x in data center k. Constraint (10) forces the variable y_{pk} to be equal to 1 as soon as the variable $t_{xk} \cdot \text{size}_x$ is strictly greater than 0. Constraint (9) indicates that the sum of the sizes of the data items placed in data center k must not exceed its storage capacity limit capacity C_k. Constraint (11) indicates that any data item x must be placed into one unique data center. The data placement problem is NP-hard for that bin packing with fragmentable items[5], as a special case of the data placement problem is already NP-hard.

3 Lagrangian Relaxation Based Method for the Data Placement Problem

3.1 Basic concepts of the Lagrangian relaxation based method

Lagrangian relaxation is a computationally efficient method for solving problems that have a set of complicating constraints[6]. In this method, complicating constraints are "moved" into the objective function by Lagrangian multipliers λ and the relaxed problem is usually easier to solve than the original problem. The solution to the relaxed problem is a function of λ, and this solution provides a lower bound on the solution to the original problem. Therefore, a good method of choosing λ is to find λ^* that produces the greatest lower bound. One heuristic algorithm for finding good values for the Lagrangian multipliers and for approaching λ^* is the subgradient optimization algorithm described in the following subsections. If the relaxed solution corresponding to λ^* is not feasible for solving an original problem with complicating constraints, further adjustments must be made to make it feasible.

3.2 Lagrangian relaxation of the data placement problem

As Constraints (9) and (10) in the integer programming model are complex with respect to solving the original data placement problem, we move them to the objective function, Eq. (8), and the Lagrangian relaxation is as follows:

$$\begin{aligned}\min L(y,t,\lambda,\mu) = &\sum_{k\in K}\sum_{p\in P} R_p \cdot \alpha \cdot y_{pk} + \\ &\beta \sum_{x\in X}\sum_{k\in K} R_{xk} \cdot \text{size}_x \cdot [1 - t_{xk}] + \\ &\sum_{k\in K} \lambda_k [(\sum_{x\in X} \text{size}_x \cdot t_{xk}) - C_k] + \\ &\sum_{p\in P}\sum_{k\in K} \mu_{pk} (\sum_{x\in X} z_{xp} \cdot t_{xk} \cdot \text{size}_x - \\ &y_{pk} \cdot \sum_{x\in X} z_{xp} \cdot \text{size}_x) \end{aligned} \tag{14}$$

subject to

$$\sum_{k\in K} t_{xk} = 1, \forall x \in X \tag{15}$$

$$y_{pk} \in \{0,1\}, \forall p \in P, \forall k \in K \tag{16}$$

$$t_{xk} \in \{0,1\}, \forall x \in X, \forall k \in K \tag{17}$$

Together, Eqs. (14)–(17) are called the Lagrangian Lower Bound Problem (LLBP), where $\{\lambda : \lambda_k | k \in K\}$ and $\{\mu : \mu_{pk} | p \in P, k \in K\}$ can be seen to impose a penalty on violated constraints. The optimal objective value of an LLBP(λ, μ) is a lower bound for the optimal objective value of the original data placement problem, and LLBP(λ, μ) is easier to solve than the original problem. We also need to find the vectors of λ^* and μ^* that produces the greatest lower bound, which is called the Lagrangian dual problem and can be solved using the subgradient optimization algorithm.

3.3 Algorithm for solving the data placement problem

To solve the Lagrangian dual problem and obtain feasible solutions for the original data placement problem, we propose an algorithm based on subgradient optimization, which includes three parts: initialization, iteration, and adjustment.

(1) Initialization. As the following subgradient optimization algorithm relies on an upper bound for the optimal objective value of the original data placement problem, we obtain an upper bound by finding a feasible solution with a heuristic for the original problem. The basic idea is to place a data item at the data center that has the largest request size (product of request rate and size) for that specific data item, while respecting the storage capacity limit of each data center. Details of this heuristic are given in Algorithm 1.

(2) Iteration. Subgradient optimization, a method for heuristically solving a Lagrangian dual problem, iteratively adjusts the Lagrangian multipliers to find the values that produce the best or nearly the best lower bound. In additional to an upper bound z_{ub} for the optimal objective value of the original problem, subgradient optimization relies on a solver for the LLBP(λ, μ).

The objective of LLBP(λ, μ) is to find the minimum for the Lagrangian relaxation of the original data

Algorithm 1 Upper Bound Calculation Algorithm

Input: request pattern set P, request rate set R, data items X and size, data centers K and storage capacity limit C, $\forall x, k, t_{xk} = 0$
Compute $R_{xk} \cdot \text{size}_x$ for all data $x \in X$ and data center $k \in K$
Create a list L of R_{xk} in descending order of $R_{xk} \cdot \text{size}_x$
while $X \neq \varnothing$ **do**
  Select R_{xk} at the head of L and record x and k
  if $C_k \geqslant \text{size}_x$ **then**
    Place x in k, $t_{xk} = 1$
    $X = X - x$
    $C_k = C_k - \text{size}_x$
    Remove all R_{xk} indexed by x from L
  else
    if R_{xk} is not the last one indexed by x in list L **then**
      Remove R_{xk} from L
    else
      Place x in a random k' with $C'_k \geqslant \text{size}_x$, $t_{xk} = 1$
      $X = X - x$
      $C'_k = C'_k - \text{size}_x$
      Remove R_{xk} from L
    end if
  end if
end while
Calculate Cost in Eq. (7) with the above t_{xk} as upper bound

placement problem when λ and μ are fixed. To solve LLBP(λ, μ) more easily and efficiently, we reorganize its formulation according to the set of variables $\{y : y_{pk} | \forall p \in P, k \in K\}$ and $\{t : t_{xk} | \forall x \in X, k \in K\}$ as follows:

$$\begin{aligned}\min L(y,t,\lambda,\mu) = &\sum_{p\in P}\sum_{k\in K}[\alpha \cdot R_p - \\ &\mu_{pk}(\sum_{x\in X} z_{xp} \cdot \text{size}_x)]y_{pk} + \\ &\sum_{x\in X}\sum_{k\in K}(\lambda_k \cdot \text{size}_x - \beta \cdot R_{xk} \cdot \text{size}_x + \\ &\sum_{p\in P} \mu_{pk} \cdot z_{xp} \cdot \text{size}_x)t_{xk} + \\ &\beta \sum_{x\in X}\sum_{k\in K} R_{xk} \cdot \text{size}_x - \sum_{k\in K} \lambda_k \cdot C_k \end{aligned} \quad (18)$$

The basic principle for solving LLBP(λ, μ) is that if the coefficients of the specified variable in Eq. (18) is less than 0, we set the bivalent variable (as constrained by Eqs. (16) and (17)) to 1 if true and 0 otherwise, while respecting the constraints of Eqs. (15)–(17). Accordingly, the minimum LLBP(λ, μ) can be determined. Both $\beta \sum_{x\in X}\sum_{k\in K} R_{xk} \cdot \text{size}_x$ and $\sum_{k\in K} \lambda_k \cdot C_k$ at the end of Eq. (18) are constant. The procedure for obtaining a solution for LLBP(λ, μ) is detailed in Algorithm 2.

Subgradient optimization iteratively adjusts the Lagrangian multipliers to find values λ^* and μ^* that produce the best lower bound. In each iteration, the steps of subgradient optimization include: (1) solve LLBP(λ, μ) to optimality using the current Lagrangian multipliers λ and μ and obtain y^*_{pk} and t^*_{xk}; compute the correspondingly objective value z^*_{LLBP} using the current y^*_{pk}, t^*_{xk}, λ, and μ; (2) compute subgradients δ^λ and δ^μ for λ and μ, respectively; (3) compute step sizes Δ^λ and Δ^μ for λ and μ, respectively; (4) update the Lagrangian multiplier for λ and μ using δ^λ, Δ^λ, δ^μ, and Δ^μ, respectively. We perform iterations using the updated Lagrangian multipliers λ and μ which are terminated only when the termination condition is met, e.g., the maximum number of iteration (set to 1000, empirically) is reached. During the iterations, we also check all the feasible solutions for the original data placement problem and the best feasible solution is recorded as $Z^{\min}_{\text{IP}}$. Details of the subgradient optimization for the data placement problem are given in Algorithm 3.

(3) Adjustment. At this point, we have λ^* and μ^* values that produce the greatest lower bound $Z_{\max}$ for the Lagrangian dual problem, and we can calculate y_{pk} and t_{xk} correspondingly. Although $Z_{\max}$ is a near optimal value of Cost for the original data placement

Algorithm 2 LLBP Solver LLBP(λ, μ)

Input: request pattern set P, request rate set R, data items X and size, data centers K and storage capacity limit C, constant overhead α, tradeoff parameter β
Check the coefficients of y_{pk} and t_{xk} in Eq. (18);
for all $p \in P, k \in K$ **do**
  if $[\alpha \cdot R_p - \mu_{pk}(\sum_{x\in X} z_{xp} \cdot \text{size}_x)] < 0$ **then**
    $y_{pk} = 1$
  else
    $y_{pk} = 0$
  end if
end for
for all $x \in X$ **do**
  Find k' with the minimal $(\lambda_k \cdot \text{size}_x - \beta \cdot R_{xk} \cdot \text{size}_x + \sum_{p\in P} \mu_{pk} \cdot z_{xp} \cdot \text{size}_x)$ for all $k \in K$
  $t_{xk'} = 1$
  for all $k \in K$ and $k \neq k'$ **do**
    $t_{xk'} = 0$
  end for
end for
Calculate Eq. (18) with the above setting of y_{pk} and t_{xk} and return the value as LLBP(λ, μ)

Algorithm 3 Subgradient Optimization Algorithm

Input: LLBP(λ, μ)
Input: z_{ub}
$\pi = \pi_{\text{init}}$
λ_k =0
$\mu_{p,k}$ =0
$Z_{\max} = -\infty$
$Z_{\text{IP}}^{\min} = \infty$
while $z_{ub} > z_{\text{LLBP}}^*$ and maximum number of iteration is not reached **do**
  $y_{pk}^* = \text{LLBP}(\lambda, \mu)$
  $t_{xk}^* = \text{LLBP}(\lambda, \mu)$
  $z_{\text{LLBP}}^* = L(y_{pk}^*, t_{xk}^*, \lambda, \mu)$
  if y_{pk}^* and t_{xk}^* is feasible for the original IP problem **then**
    Calculate the value of Cost(y_{pk}^*, t_{xk}^*) as Z_{IP}
    if $Z_{\text{IP}} < Z_{\text{IP}}^{\min}$ **then**
      $Z_{\text{IP}}^{\min} = Z_{\text{IP}}$
    end if
  end if
  $\delta_k^\lambda = (\sum_{x \in X} \text{size}_x \cdot t_{xk}^*) - C_k$
  $\delta_{p,k}^\mu = \sum_{x \in X} z_{xp} \cdot t_{xk}^* \cdot \text{size}_x - y_{pk}^* \cdot \sum_{x \in X} z_{xp} \cdot \text{size}_x$
  $\Delta^\lambda = \dfrac{\pi(z_{ub} - z_{\text{LLBP}}^*)}{\sum_{1 \leqslant k \leqslant |K|} {\delta_k^\lambda}^2 + \sum_{1 \leqslant p \leqslant |P|, 1 \leqslant k \leqslant |K|} {\delta_{p,k}^\mu}^2}$
  $\Delta^\mu = \dfrac{\pi(z_{ub} - z_{\text{LLBP}}^*)}{\sum_{1 \leqslant k \leqslant |K|} {\delta_k^\lambda}^2 + \sum_{1 \leqslant p \leqslant |P|, 1 \leqslant k \leqslant |K|} {\delta_{p,k}^\mu}^2}$
  $\lambda_k = \max(0, \lambda_k + \Delta^\lambda \cdot \delta_k^\lambda)$
  $\mu_{p,k} = \max(0, \mu_{p,k} + \Delta^\mu \cdot \delta_{p,k}^\mu)$
  if $z_{\text{LLBP}}^* > Z_{\max}$ **then**
    $Z_{\max} = z_{\text{LLBP}}^*$
  end if
  if no_improvement **then**
    $\pi = \pi/2$
  end if
end while

problem, y_{pk} and t_{xk} may still not be feasible due to the relaxing constraints. Therefore, we make further adjustments according to the following principle: we remove the data item with the smallest request size (product of request rate and size) from the data center in which the storage capacity constraint is breached, and relocate it to the data center where the data item has the largest request size and the remaining storage capacity is sufficient to store the data item. Adjustments cease when all the storage capacity limits are respected. We can calculate Cost using the adjusted feasible solution and record it as Z'_{IP}. We then compare Z'_{IP} with $Z_{\text{IP}}^{\min}$, and fix the better one as the final solution to the original data placement problem.

4 Performance Evaluation

4.1 Experimental methodology

In our experiment, we considered there were $|K| = 10$ geo-distributed data centers, and a total number of data items of $|X| = 2000$. Each data item is consistent with the uniform size distribution in $[1, 10]$. The number of request patterns was $|P| = 4000$. We generated each request pattern by randomly selecting no more than $d = 30$ data items with a uniform distribution. We uniformly set each data center as the source location of that request pattern. We considered the request rates of different patterns to follow the uniform distribution in [0,10]. The total storage capacity $\sum_{k \in K} C_k$ of the data centers was larger than the total size of data items $\sum_{x \in X} \text{size}_x$. We considered the ratio of the total storage capacity of all $|K| = 10$ data centers to the total size of data items $\dfrac{\sum_{k \in K} C_k}{\sum_{x \in X} \text{size}_x}$ was set as 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, and 1.6, respectively. We considered each data center's storage to be uniform and the sum of all data center's storage did not exceed $\sum_{k \in K} C_k$. The default values of other parameters were $\alpha = 1$ and $\beta = 3$.

With respect to the data placement problem, we compare the results of our proposed Lagrangian Relaxation based method (LR) with other representative approaches: Closest[4], Multiget[3], MostLocalized, and Hash[4].

- Closest: While respecting the storage capacity constraint of each data center, this method places a data item at the data center that has the largest request rate for that specific data item, but overlooks the size of the data item and the cost of associated data access.
- Multiget: While respecting the storage capacity limits of data centers, this method places associated data items in the same data center, but does not consider the cost of remote data serving.
- MostLocalized: This method places a data item at the data center that has the largest request size for that specific data item, but neglects the associated data access. The storage limits of data centers are respected.
- Hash: This method places the data items at data centers based on the hashing results.

In our experiment, we consider the following metrics:

- Cost of associated data access: The average number of data centers involved in fulfilling each request is multiplied by the request rate, and is then divided by the number of data centers. This value is related to Cost_1.
- Cost of remote data serving: The total size of the requested data items that are not locally served, which is related to Cost_2.
- Objective: The weighted sum of associated data access cost and remote data serving cost is the Cost.

4.2 Experiment results

4.2.1 Comparison of associated data access costs

Figure 3 compares the associated data access costs obtained based on the data placement results generated by all the algorithms. The cost value is normalized by the obtained value of the Hash algorithm under the same settings by default. Regarding the associated data access cost, Multiget achieves the best performance, because it focuses on the associated relationship among data items in the placement. The MostLocalized and Closest algorithms only consider the cost of remote data serving and overlook the associated data access cost. Our LR considers the weighted costs of associated data access and remote data serving, and its performance is inferior to Multiget but better than the MostLocalized and Closest algorithms.

4.2.2 Comparison of remote data serving costs

The cost value is also normalized by the value obtained by the Hash algorithm. As shown in Fig. 4, the MostLocalized algorithm achieves the best performance

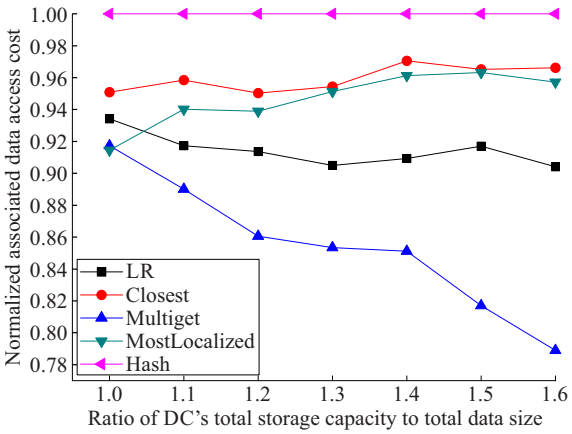


Fig. 3 Comparison of associated data access cost between algorithms.

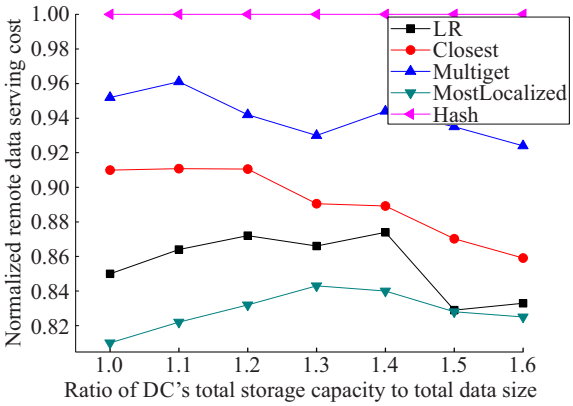


Fig. 4 Comparison of remote data serving cost between algorithms.

in optimizing the remote data serving cost. The reason is that with this algorithm, each data item is always placed at the data center with the largest request size (as the product of the request rate and the data item size) for it. Requests for data, especially those with large request sizes, are thus placed locally and avoid remote data serving to the greatest extent. As the Closest algorithm only considers placing each data item at the data center with the largest request rate, the size of the data items is not taken into account, and thus its performance is inferior to that of the MostLocalized algorithm. The Multiget algorithm merely considers the associated data access cost and totally neglects remote data serving, thus its performance is worse. Our proposed LR considers the weighted cost of both the associated data access and remote data serving, and its performance is better than both Multiget and Closest as expected.

4.2.3 Comparison of the optimization objective

In Fig. 5, we evaluate the performance of LR and compare it with the other algorithms regarding the optimization objective. We normalized the object value by the obtained value of the Hash algorithm under the same settings by default. We can observe that LR performs the best with respect to the objective. The Multiget and MostLocalized algorithms only consider one of the two metrics. By optimizing both metrics, LR achieves the lowest objective value even if its performance is not the best on either individual metric. We also found that with increasing storage capacity, the gap between the performance of LR and the other algorithms is reduced because looser constraints are more beneficial for heuristics other than LR.

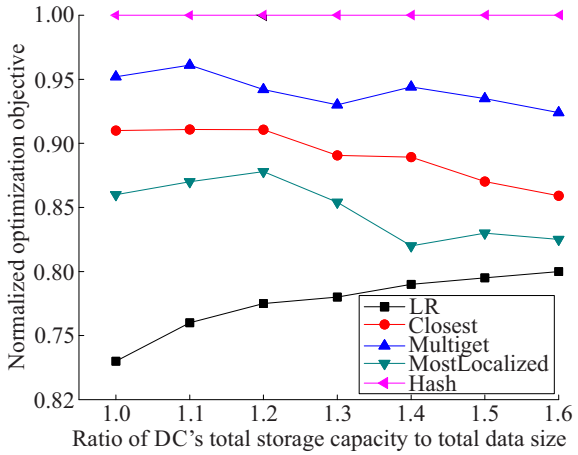


Fig. 5 Comparison of the optimization objective between algorithms.

4.2.4 Effect of metric tradeoff parameter β

The parameter β trades off the importance of the costs brought by associated data access and remote data serving. We set the value of β as $1, 2, 3, 4$, and 5, and obtained the corresponding objective value, respectively. Figure 6 shows that as β increases, the advantage of LR over the MostLocalized algorithm becomes less obvious. This is because the weight of the remote data serving is increased in the objective function and the MostLocalized algorithm gives the best performance on that metric.

5 Related Work

Due to the availability of geo-distributed storage sites, there are certain flexibilities in choosing a data storage location. Agarwal et al.[7] presented the concept of automatic data placement across geo-distributed datacenters, which iteratively move data items closer

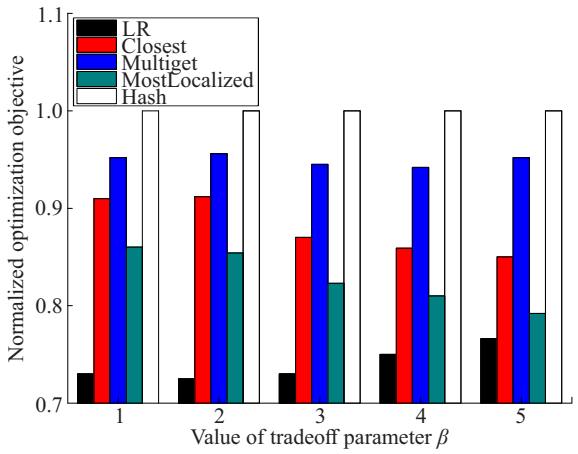


Fig. 6 Effect of metric tradeoff parameter β.

to both clients and other data items to which they are related to. The problem definition of their work is related to that in ours, but they only consider the pairwise relationship between data items. Yu and Pan[4, 8] proposed an associated data placement method, which improves the co-location of associated data and localized data serving while ensuring a balance between nodes. They demonstrated the use of hypergraphs in modeling the data placement problem by partitioning data items and placing them at the data centers. While our proposed model is similar to theirs, we assume each data item may have a different size and that each data center has a storage capacity limit, which is a common understanding of the situation in scientific computing. Moreover, since our problem is more complicated than that in Ref. [4], the hypergraph approach is not applicable and we propose a different method for solving this data placement problem.

Besides the location difference possibly affecting user's data access, other factors also may affect system performance in data access, e.g., the multi-get hole effect[3], in which the use of fewer distributed nodes is preferred to fulfill multiple item requests. The authors recommend that strongly related data items be put together to avoid the associated data access cost. Studies related to data placement have either focused on the distance between the data and users, such as in Ref. [9], or only addressed the co-location of associated data items, such as in Refs. [10, 11]. In Ref. [10], the authors presented a workload-aware framework for solving problems of data placement and replication in cloud data management systems and provided algorithms to minimize the average query span. Other related works have focused on the issues in online social networks scenario[12, 13], which neglect the relationship between data items and associations that involve more than two entities.

Golab et al.[14] presented practical algorithms for minimizing the data communication cost of evaluating data-intensive workloads. The authors reduced the data communication problem to standard graph partitioning and showed how to take into account load balancing and replication. Çatalyürek et al.[15] gave a similar problem a hypergraph partitioning-based formulation, and proposed a heuristic that generates data placement and task assignment schemes simultaneously for scientific workflow. Zhang et al.[16–18] studied data-intensive workflow scheduling in multicluster and distributed data centers, but they neglected the associated data

access cost in workflow scheduling. In Ref. [19], the authors proposed that for data centers with bandwidth limits, it is necessary to solve the graph partitioning problem with capacity constraints, but they proposed no practical solution or algorithm for doing so.

6 Conclusion

In this paper, we first proposed to address the data placement problem by jointly improving the co-location of associated data and localized data serving, which is subject to the storage capacity constraints of the data centers. Then we formulated the problem with an integer programming model and proposed a Lagrangian relaxation based approach for solving this data placement problem. After extensive experimentations, our simulation results show that our Lagrangian relaxation based method significantly reduces the overall data access cost.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61320106007, 61572129, 61502097, and 61370207), the National High-Tech Research and Development (863) Program of China (No. 2013AA013503), International S&T Cooperation Program of China (No. 2015DFA10490), Jiangsu research prospective joint research project (No. BY2013073-01), Jiangsu Provincial Key Laboratory of Network and Information Security (No. BM2003201), Key Laboratory of Computer Network and Information Integration of Ministry of Education of China (No. 93K-9), and partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization and Collaborative Innovation Center of Wireless Communications Technology.

References

[1] AMS02, http://www.ams02.org/, 2016.

[2] LHC, http://home.cern/topics/large-hadron-collider, 2016.

[3] Memcached multiget hole, http://highscalability.com/blog/2009/10/26/facebooks-memcached-multiget-hole-more-machines-more-capacit.html, 2009.

[4] B. Y. Yu and J. P. Pan, Location-aware associated data placement for geo-distributed data-intensive applications, in *Proc. 34th IEEE Conference on Computer Communications*, Kowloon, Hong Kong, China, 2015, pp. 603–611.

[5] B. LeCun, T. Mautor, F. Quessette, and M. A. Weisser, Bin packing with fragmentable items: Presentation and approximations, *Theoretical Computer Science*, vol. 602, pp. 50–59, 2015.

[6] M. L. Fisher, The Lagrangian relaxation method for solving integer programming problems, *Management Science*, vol. 50, no. 12, pp. 1861–1871, 2004.

[7] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, and A. Wolman, Volley: Automated data placement for geo-distributed cloud services, in *Proc. 7th USENIX Symposium on Networked Systems Design and Implementation*, San Jose, CA, USA, 2010, pp. 17–32.

[8] B. Y. Yu and J. P. Pan, Sketch-based data placement among geo-distributed datacenters for cloud storages, in *Proc. 35th IEEE Conference on Computer Communications*, San Francisco, CA, USA, 2016, pp. 1–9.

[9] H. Xu and B. Li, Joint request mapping and response routing for geo-distributed cloud services, in *Proc. 32th IEEE Conference on Computer Communications*, Turin, Italy, 2013, pp. 854–862.

[10] K. A. Kumar, A. Quamar, A. Deshpande, and S. Khuller, SWORD: Workload-aware data placement and replica selection for cloud data management systems, *VLDB Journal*, vol. 23, no. 6, pp. 845–870, 2014.

[11] A. Quamar, K. A. Kumar, and A. Deshpande, SWORD: Scalable workload-aware data placement for transactional workloads, in *Proc. 16th International Conference on Extending Database Technology*, Genoa, Italy, 2013, pp. 430–441.

[12] L. Jiao, J. Li, W. Du, and X. M. Fu. Multi-objective data placement for multi-cloud socially aware services, in *Proc. 33th IEEE Conference on Computer Communications*, Toronto, Canada, 2014, pp. 28–36.

[13] L. Jiao, J. Li, T. Y. Xu, W. Du, and X. M. Fu, Optimizing cost for online social networks on geo-distributed clouds, *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 99–112, 2016.

[14] L. Golab, M. Hadjieleftheriou, H. Karloff, and B. Saha, Distributed data placement to minimize communication costs via graph partitioning, in *Proc. 26th International Conference on Scientific and Statistical Database Management*, Aalborg, Denmark, 2014, pp. 20–28.

[15] Ü. V. Çatalyürek, K. Kaya, and B. Uçar, Integrated data placement and task assignment for scientific workflows in clouds, in *Proc. 4th International Workshop on Data Intensive Distributed Computing*, 2011, pp. 45–54.

[16] J. H. Zhang, J. Z. Luo, and F. Dong, Scheduling of scientific workflow in non-dedicated heterogeneous multicluster platform, *Journal of Systems and Software*, vol. 86, no. 7, pp. 1806–1818, 2013.

[17] J. H. Zhang, J. Z. Luo, and F. Dong, Scientific workflow scheduling in non-dedicated heterogeneous multicluster with advance reservations, *Integrated Computer-Aided Engineering*, vol. 22, no. 3, pp. 261–280, 2015.

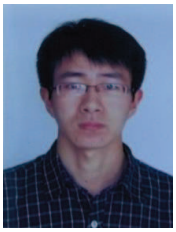
[18] J. H. Zhang, M. J. Wang, J. Z. Luo, F. Dong, and J. X. Zhang, Towards optimized scheduling for data-intensive scientific workflow in multiple datacenter environment, *Concurrency and Computation: Practice and Experience*, vol. 27, no. 18, pp. 5606–5622, 2015.

[19] P. Bodik, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, Surviving failures in bandwidth-constrained datacenters, in *Proc. Annual Conference of the ACM Special Interest Group on Data Communication*, Helsinki, Finland, 2012, pp. 431–442.

Jinghui Zhang received the BS degree from Southeast University in 2005, and the PhD degree in computer science from Southeast University in 2014. He is a lecturer in School of Computer Science and Engineering, Southeast University, China. His current research interests include cloud computing, geo-distributed data management, and distributed computing. He is a member of the ACM and CCF.

Junzhou Luo received the BS degree in applied mathematics and the MS and PhD degrees in computer network all from Southeast University, China, in 1982, 1992, and 2000, respectively. He is a full professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests are next generation network architecture, network security, cloud computing, and wireless LAN. He is a member of the IEEE Computer Society and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design, and he is a member of the ACM and chair of ACM Nanjing Chapter.

Aibo Song received the MS degree from Shandong University of Science and Technology, and the PhD degree from Southeast University, China, in 1996 and 2003, respectively. He is currently an associate professor in the School of Computer Science and Engineering, Southeast University. His current research interests include cloud computing and big data processing.



Jian Chen received the BS degree from Soochow University in 2014. He is currently working toward the master degree in computer science at Southeast University, China. His research interests include cloud computing and geo-distributed data management.