Task Assignment Optimization in Geographically Distributed Data Centers

Bowu Zhang
Marist College
Poughkeepsie, NY, USA
bowu.zhang@marist.edu

Jinho Hwang
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
jinho@us.ibm.com

***Abstract*—Recent advance in geo-distributed systems has made distributed data processing possible, where tasks are decomposed into subtasks, deployed into multiple data centers and run in parallel. Compared to conventional approaches that process every task in a single datacenter resulting in high latency and large data aggregation, the geo-distributed cloud systems provide a highly available and more economic platform. However, distributed application (task) execution introduces extra cost and latency as data need to be exchanged between data centers. In addition, task dependency and diverse task constraints make it even more challenging to choose an appropriate task assignment strategy. In this paper, we discuss a task assignment problem in geographically distributed cloud systems. In light of growing demand from big data processing and storage, we consider data intensive tasks where a task often requires significant computing resources and its input data typically located in multiple data centers. By taking the distributed input, task dependency, heterogeneous pricing scheme, and resource constraints into account, we aim to optimize the performance when deploying tasks in geo-graphically distributed data centers. A heuristic algorithm is presented to provide an approximate solution to the proposed NP-hard problem. We perform an extensive simulation study to evaluate the performance of our solution under various settings. The simulation results demonstrate that our approach can outperform the state-of-the-art strategies, and achieve significant reduction in cost and latency.**

***Index Terms*—Cloud computing; Geo-distributed deployment; Task assginment**

I. Introduction

The recent years have witnessed the explosive growth of data generated by various applications such as web search, health monitoring, and social networks [1, 2] . To serve the rising demand of big data storage and processing, major cloud service providers now build tens of geographically distributed datacenters. On the other hand, virtualization has evolved into a more flattened and lighter function layer thanks to lightweight virtual machines, containers and library OSes (unikernels), calling for applications with a distributed deployment by segregating the application in different physical boxes, or further different data centers. Compared with conventional cloud systems where all the tasks are executed in a single data center, geo-distributed cloud systems provide a highly available and more economic platform where tasks can be partitioned and executed in multiple data centers in parallel [3, 4]. Taking advantage of these distributed resources, large-scale and computation-intensive applications, can be realized with significantly less resources and time.

However, deploying tasks in distributed data centers introduces extra cost and latency as the data may need to be transferred between data centers, and the transmission cost (due to electricity and bandwidth) and latency between servers in different data centers are far greater than that between servers in a single data center. Thus to implement big data applications with optimized performance, we need to strategically design and follow a rigorous optimization method. Intuitively, we may want to deploy tasks in the closest data center to the user for low transmission cost and latency, while it is possible that the closest data center does not have enough capacity for tasks, or it is more beneficial to place applications at data centers that are less close but more efficient in processing. In addition, in geo-distributed data centers, it has been a growing trend that a user has his data and replicas saved in multiple data centers. The performance of running tasks thus critically depends on how many replicas are maintained, where these replicas are located, and the choice of replica for input data. Moreover, as a task can be splitted and its subtasks can be executed distributedly in parallel, when the task would be finished is determined by its "longest" subtask. Finishing one subtask fast does not necessarily result in finishing the entire task fast.

To address the challenges outlined above, in this paper, we focus on the problem of deploying tasks into distributed data centers with an objective of optimized performance (i.e., minimizing latency/cost). Compared to the traditional all-in-one single data center scenario, the problem of task assignment in distributed cloud systems remains largely unexplored. Our solution to the problem is motivated by the following observations in geo-distributed cloud services. First, geo-distributed data centers vary in the pricing scheme and resource capacity [5, 6]. Thus it is possible that it is more economic to run tasks in data centers cheaper in price/greater in capacity than in the closest data center. Second, most users have a number of replicas stored on different data centers, and these replicas are updated synchronously whenever there is a change in the primary replica [7, 8]. When obtaining data, an application can connect to any replica, not necessarily the primary one. Third, data processing tasks in geo-distributed cloud systems often involve significant workloads. These tasks can be decomposed into subtasks that take input and run in parallel. However,

978-3-901882-89-0 @2017 IFIP

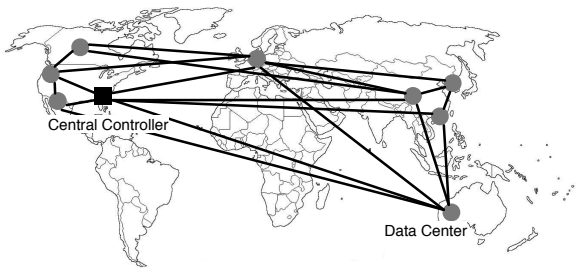

Fig. 1. Geographically Distributed Data Centers and Central Controller.

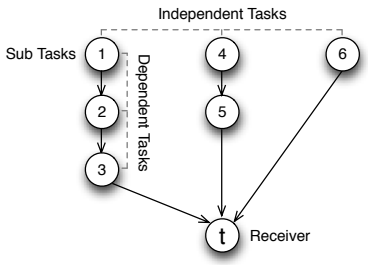

Fig. 3. Task Execution with Independent and Dependent Sub-tasks.

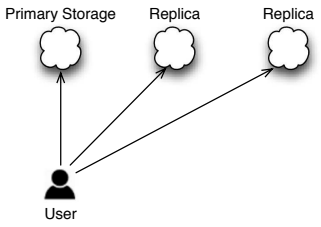

Fig. 2. Storage Replication.

there might exist dependency between subtasks such that the input (or at least part of the input) of one subtask comes from the output of another subtask [9]. As a result, the former task has to wait until the latter finishes. Taking these unique characteristics into account, we seek to design a set of principles to guide task assignment in geo-distributed cloud systems in an economic and effective fashion. In general, our contributions are summarized below:

- We propose a task assignment problem in context of distributed data centers, task dependency, heterogeneous pricing scheme and resource constraints, with an objective of minimizing cost and latency;
- We develope a heuristic algorithm that can deploy tasks into distributed data centers in an economic and timely fashion, to solve the proposed NP-hard problem;
- We perform an extensive simulation study to evaluate the performance of the proposed solution under various settings. Our simulation results demonstrate that our algorithm outperforms the existing approaches.

The rest of the paper is organized as follows. In Section II, we formulate the task assignment problem. A heuristic algorithm is introduced in Section III. Section IV illustrates the results of our simulation works. Finally, we introduce related work in Section V, and conclude the paper in Section VI.

II. Problem Formulation

A. Notations

Consider a geo-distributed cloud system where N data centers are placed at different geographical locations and managed by a central controller. Let $D = \{d_1, d_2, \ldots, d_N\}$ be the set of data centers, where $d_i, i \in \{1, \ldots, N\}$ represents the data center in region i. We evaluate the capacity of a data center d_i by n_i, the number of servers residing in d_i. Assume all servers considered in this paper are homogeneous, but the number of servers that each data center contains (n_i) may vary due to energy and location constraints. Servers in a data center are all connected to a switch, and data centers are connected through switches. Let $G(D, E)$ be the graph that describes the data center topology where nodes stand for data centers, and edges (in edge set E) stand for WAN links between data centers (Fig.1).

We adopt a multi-cloud service paradigm [10, 11] which has been widely used when processing big data in geo-distributed cloud systems. In this paradigm, every user has a primary replica and multiple secondary replicas while each replica is stored in a different data center (Fig.2). When there is a change in the primary replica, all secondary replicas will be updated accordingly. Let $U = \{uid\}$ be the set of users where every user can be identified by an unique number uid. Define $RP_{uid} = [rp_{uid}(1), rp_{uid}(2), \ldots, rp_{uid}(N)]^T$, where $rp_{uid}(i)$ is 1 if there is a replica of user uid located in data center d_i, and 0 otherwise. Let $rpm_{uid} \in D$ be the index of the data center that the primary replica of uid resides in, and k_{uid} indicates the number of replicas that user uid has.

In this paper, we consider big data processing tasks which often involves heavy loads of data processing and computation. Assume that every task that arrives at the central controller can be decomposed into multiple subtasks that can run in parallel. As the input data storage spans multiple data centers, such tasks(sub-tasks) can be deployed on any of input data replicas, not necessarily the primary replica. By leveraging the distributed storage of input data and parallel processing, we can achieve lower latency than that by utilizing servers in a single data center. We assign every task an unique identification number tid. Define $S_{tid} = \{s_{tid}(1), s_{tid}(2), \ldots, s_{tid}(m_{tid})\}$ as the set of subtasks that task tid is composed of, where $s_{tid}(i), i \in \{1, 2, \ldots, m_{tid}\}$ is a positive integer that indicates the number of servers required to process the i_{th} subtask of tid, and m_{tid} indicates the number of subtasks that task tid contains. Let $r(tid) \in U$ represent the receiver that the task results will be directed to. In this paper, we assume that results of a task will be written to receiver's primary replica. Let $o_{tid} \in U$ represent the user/owner of the input data for task tid. Without loss of generosity, we assume that the input data of all the subtasks of tid is from the same user, but our model can be easily extended to the case where a task has multiple

input data sources.

In addition, there might be task dependency between subtasks in S_{tid}. Let DPM_{tid} be a $m_{tid} \times m_{tid}$ matrix, where $DPM_{tid}(i,j) = 1$ if there exits task dependency between $s_{tid}(i)$ and $s_{tid}(j)$, and $DPM_{tid}(i,j) = 0$ otherwise. If $DPM_{tid}(i,j) = 1$, subtask $s_{tid}(j)$ must wait until $s_{tid}(i)$ completes its work, as the input of $s_{tid}(j)$ includes both data from o_{tid} and output of $s_{tid}(i)$. Let $TG_{tid}(VS_{tid}, DP_{tid})$ be a directed task graph where nodes $\in VS_{tid}$ stand for the indices of subtasks in S_{tid}, and edges $\in DP_{tid}$ indicate the task dependency between corresponding tasks. An edge $dp_{tid}(i,j)$ exists when $DPM_{tid}(i,j) = 1$. Define a path in TG_{tid} as a sequence of nodes that are connected by edges. The starting node of a path is the node that no edges coming in, i.e., node $1, 4$ and 6 in Figure 3, while the ending node of a path is the node that no edges are going out, i.e., node 3 and 5 in Figure 3. We add an extra node t to TG_{tid} to represent the receiver of task tid, and draw an edge from every ending node to t as shown in Figure 3. Let $PT_{tid} = \{pt_{tid}(1), \ldots, pt_{tid}(np_{tid})\}$ be the set of paths in TG_{tid}, where np_{tid} represents the number of paths in TG_{tid}, and each path in TG_{tid} is described by a vector $pt_{tid}(i) = [pt_{tid}(i)[0], pt_{tid}(i)[1], \ldots, pt_{tid}(i)[ns_{tid}(i)]]^T$ that nodes along the path are inserted to this vector in order, i.e., $pt_{tid}(i)[0] \in VS_{tid}$ is the starting node, and $pt_{tid}(i)[ns_{tid}(i)] = t$ is the ending node. The number of nodes along path $pt_{tid}(i)$ is denoted by $ns_{tid}(i)$. For example, in Figure 3, there are 3 nodes along the path from node 1 to node 3, and the vector to represent this path would be $[1, 2, 3, t]^T$. Assume that subtasks on different paths are independent of each other, in other words, there would be no two paths in TG sharing the same node (except the ending node t).

Let $ce^i_{tid}(j), i \in \{1, 2, \ldots, N\}, j \in \{1, 2, \ldots, m_{tid}\}$ be the execution cost of running the j_{th} subtask of task tid, $s_{tid}(j)$ on data center d_i. Let $ct(i,k)$ be the unit cost to transfer data from data center d_i to another data center d_k. Let $f(s_{tid}(j))$ represent the size of result for subtask $s_{tid}(j)$ in terms of units. Thus the cost of transferring the output of $s_{tid}(j)$ from d_i to d_k is $f(s_{tid}(j)) \times ct(i,k)$.

Let $le^i_{tid}(j), i \in \{1, 2, \ldots, N\}, j \in \{1, 2, \ldots, m_{tid}\}$ be the execution latency of $s_{tid}(j)$ on data center d_i. Let $lt(i,k,f)$ be the transmission latency to transfer f units of data from data center d_i to another data center d_k. Thus the latency of transferring the output of $s_{tid}(j)$ from d_i to d_k is $lt(i,k,f(s_{tid}(j)))$.

Let X_{tid} be a $m_{tid} \times N$ matrix that represents the data center assignment for task tid, where the element at row i and column j, $x_{tid}(i,j)$ is 1 if subtask $s_{tid}(i)$ is deployed at data center d_j, and 0 otherwise. Assume a subtask can not be splitted further, and can only be deployed to one and only one data center, we have:

$$\sum_j x_{tid}(i,j) = 1, \forall i \in \{1, 2, \ldots, m_{tid}\} \tag{1}$$

On the other hand, the aggregated assignment at datacenter d_j is $\sum_i x_{tid}(i,j)$, and we have:

$$\sum_i x_{tid}(i,j) \leq n'_i, \forall j \in \{1, 2, \ldots, N\} \tag{2}$$

where n'_i represents the capacity of d_j at the time task tid gets processed at the central controller. This ensures that aggregated assignment processed at datacenter d_j does not exceed its current capacity.

Let c_{tid} be the total cost of processing task tid in $G(D,E)$. We have:

$$\begin{aligned} c_{tid} = & \sum_{i=1}^{m_{tid}} \sum_{j=1}^{N} ce^j_{tid}(i) \times x_{tid}(i,j) \\ & + \sum_{i=1}^{m_{tid}} \sum_{j=1}^{m_{tid}} \sum_{k=1}^{N} \sum_{q=1}^{N} f(s_{tid}(i)) \times \\ & ct(k,q) \times x_{tid}(i,k) \times x_{tid}(j,q) \times DPM_{tid}(i,j) \end{aligned} \tag{3}$$

Let l_{tid} be the latency of processing task tid in $G(D,E)$. As subtasks of tid can run in parallel, we define l_{tid} as the the maximal accumulated latency of paths in $TG_{tid}(VS_{tid}, DP_{tid})$.

$$\begin{aligned} l_{tid} = \max_{pt_{tid}(i) \in PT_{tid}} \{ & \sum_{j=1}^{ns_{tid}(i)-1} \sum_{k=1}^{N} le^k_{tid}(pt_{tid}(i)[j]) \\ & \times x_{tid}(pt_{tid}(i)[j], k) + \\ & \sum_{j=1}^{ns_{tid}(i)-2} \sum_{k=1}^{N} \sum_{q=1}^{N} lt(k, q, f(s_{tid}(pt_{tid}(i)[j]))) \\ & \times x_{tid}(pt_{tid}(i)[j], k) \times x_{tid}(pt_{tid}(i)[j+1], q) \\ & + \sum_{k=1}^{N} lt(k, rpm_{r(tid)}, f(s_{tid}(pt_{tid}(i)[ns_{tid}(i)-1]))) \\ & \times x_{tid}(pt_{tid}(i)[ns_{tid}(i)-1]], k) \} \end{aligned} \tag{4}$$

where $rpm_{r(tid)}$ represents the index of the data center that the primary replica of receiver of task tid resides in.

B. Problem Definition

Given a geo-distributed cloud system $G(D,E)$, considering an arrival task tid at the central controller, our goal is to find a task assignment matrix X_{tid} that optimizes the performance (i.e., minimizes the latency/cost) and satisfies constraints from Equations (1), (2). Take the latency minimization problem for example. The problem can be expressed as,

$$\begin{aligned} & \min l_{tid} \\ \text{s.t } & \sum_j x_{tid}(i,j) = 1, \forall i \in \{1, \ldots, m_{tid}\}, \text{ and} \\ & \sum_i x_{tid}(i,j) \leq n'_i, \forall j \in \{1, \ldots, N\} \end{aligned} \tag{5}$$

As elements in X_{tid} are all binary variables, and there are quadratic terms of these binary variables in Equation (5) (more specifically, in c_{tid} and l_{tid}), the proposed task assignment problem Equation (5) falls into Binary Quadratic Programming

(BQP) problems, which are known to be NP-hard [12, 13]. In the following section, we will propose an approximation scheme to solve problem.

III. Task Assignment Algorithm

In this section, we present a heuristic algorithm to provide an approximate solution to the proposed NP-hard problem Equation (5). Given a geo-distributed cloud system $G(D, E)$, when a task tid arrives at the central controller, we first partition it into subtasks, figure out the task dependencies, and create a task graph $TG_{tid}(VS, DP)$. Then we run algorithm 1 to assign subtasks to distributed data centers. In particular, we would look at subtasks on the longest path in PT_{tid} first, then subtasks on the second longest path, and so on, as the latency of long paths are usually longer than that of short paths and we need to ensure that the latency of running tid on $G(D, E)$ is less than the given bound bid. When assigning $s_{tid}(j)$, we would evaluate every applicable data center d_i (d_i is applicable if $rp_{oid}(i) = 1$ and if $n'_i \geq s_{tid}(j)$) by $score^i_{tid} = \alpha c^i_{tid} + \beta l^i_{tid} + \rho lt^i_{tid} + \gamma ct^i_{tid}$, where $\alpha \geq 0$, $\beta \geq 0$, $\rho \geq 0$, and $\gamma \geq 0$ are constants, and

$$c^i_{tid} = \frac{ce^i_{tid}(j) + f(s_{tid}(p)) \times ct(k, i)}{c^{ave}_{tid}} \tag{6}$$

$$l^i_{tid} = \frac{le^i_{tid}(j) + lt(k, i, f(s_{tid}(p)))}{l^{ave}_{tid}} \tag{7}$$

$$lt^i_{tid} = \frac{lt(i, rpm_{oid}, f(s_{tid}(j)))}{lt^{min}_{tid}} \tag{8}$$

$$ct^i_{tid} = \frac{ct(i, rpm_{oid}) \times f(s_{tid}(j))}{ct^{min}_{tid}} \tag{9}$$

In the above equations, d_k represents the data center that the immediate previous subtask $s_{tid}(p)$ is assigned to, $c^{ave}_{tid} = \frac{\sum_{i=1}^{N}(ce^i_{tid}(j)+f(s_{tid}(p))\times ct(k,i))\times rp_{oid}(i)}{N}$ and $l^{ave}_{tid} = \frac{\sum_{i=1}^{N}(le^i_{tid}(j)+lt(k,i,f(s_{tid}(p)))\times rp_{oid}(i)}{N}$ represent the average cost and latency of running $s_{tid}(j)$ on all data centers in $G(D, E)$, lt^{min}_{tid} and ct^{min}_{tid} represent the minimum of them, respectively. The date center with smallest $score^i_{tid}$ would be selected for assignment. In particular, constants α, β, ρ, and γ can be adjusted to fit the application purpose. In Eq.5, since we aim to minimize the total latency, we would let α and γ be 0, and β, ρ be greater than 0. When assigning the last subtask on a path, besides c^i_{tid} and l^i_{tid}, we would also consider the transmission cost/latency between the candidate data center and the receiver's data center $d_{rpm_{oid}}$. As a result, we would evaluate every applicable data center by $rscore^i_{tid} = \alpha(c^i_{tid} + f(s_{tid}(j)) \times ct(i, rpm_{oid})) + \beta(l^i_{tid} + lt(i, rpm_{oid}, f(s_{tid}(j))))$ when we assign the last subtask on a path.

IV. Simulation

In this section we validate the performance of our algorithm over simulations under various conditions.

Algorithm 1 Task Assignment Algorithm

Input:

- $G(D, E)$: Geo-distributed data center graph
- $TG_{tid}(VS, DP)$: Task graph for task tid

Output:

- X_{tid}: The task assignment matrix for task tid

function Task Assignment Algorithm
  Create a matrix X_{tid} and set all elements in X_{tid} to 0
  Create PT_{tid} based on $TG_{tid}(VS, DP)$
  Sort paths in PT_{tid} in decreasing order of length
  for $pt_{tid}(k) \in PT_{tid}$ **do**
    for $q \in [1, ns_{tid}[k] - 1]$ **do**
      for $i \in [1, N]$ **do**
        if $rp_{oid}(i) = 1$ AND $n'_i \geq s_{tid}(pt_{tid}(k)[q])$ **then**
          Calculate $score^i_{tid}$
        end if
      end for
      Select the data center d_{min} that results in the smallest $score^{min}_{tid}$. Let $x_{tid}(pt_{tid}(k)[q], min) = 1$
      Update the capacity n'_{min} of data center d_min: $n'_{min} = n'_{min} - s_{tid}(pt_{tid}(k)[q])$
    end for
  end for
  Output X_{tid}
end function

A. Evaluation Settings

We create a geo-distributed cloud system that consists of 20 data centers. For data center d_i, its capacity is represented by n_i, the number of processing servers within a range of $[1000, 15000]$. We adopt the Amazon EC2 price model [14] for cost and latency. The cost/latency for each data center (pair) is randomly taken from a set of real world prices/latency measurements on Amazon EC2 machines (round trip time (RTT) between instances in the same region for execution latency, RTT between instances in different regions for transmission latency). We create 100 users and every user has a number of replicas across data centers where the number of replicas varies in $[2, 20]$ and data centers which replicas locate in are selected randomly. We consider big data processing tasks in simulations that each task can be decomposed into subtasks which further can be grouped into a set of independent paths. The owner of the input data and the receiver of each task is selected randomly from the user set $\{uid\}$. The number of paths and the number of subtasks along each path are chosen randomly from $[2, 8]$ and $[1, 10]$, respectively. The number of processing servers required by every subtask varies in $[100, 500]$, and output of every subtask is proportional to the number of required processing servers. We set α to 0, β to 1, ρ to 1, and γ to 0(We also tested different values for latency minimization and cost minimization. Results exhibit similar characters, therefore omitted due to space limitation).

We implement two heuristic algorithms for performance comparison: a random assignment algorithm that randomly picks a data center for every subtask, and a greedy algorithm that always selects the data center with the largest number of servers available. They are denoted as $RAND$ and

$GREEDY$, respectively. Every result presented is an average of 50 runs.

B. Evaluation Results

First, we investigate how the number of replicas affects the latency via varing its value from 1 to 18 in Figure 4. It is obvious that compared with the other two algorithms, the proposed algorithm achieves the least latency under different number of replicas. The difference of latency between the proposed algorithm and the other two increases as the number of replicas increases, indicating that the proposed method performs especially well when the number of replicas is big. We can also observe that the latency generated by the proposed algorithm stays steadily around 150ms when the number of replicas is between 1 and 8, and drops gradually as the number of replicas grows beyond 10. This is because when there are more replicas, there are more choices. Since the proposed algorithm 1 ranks data centers by $score^i_{tid}$, we would always assign tasks to data centers bigger in capacity and smaller in latency which in turn results in better performance. On the other hand, the latency of GREEDY and RAND increases as the number of replicas increases when the number of replicas is less than 10, and does not change significantly when the number of replicas is larger than 10. For GREEDY algorithm, as it always picks the data center with the largest capacity, when there are more data centers available, it is more likely to assign neighboring subtasks on a path to different data centers even when there exists a data center able to run both subtasks, leading to a high transmission latency. The same reason applies to the RAND algorithm as it chooses a data center randomly, when there are more replicas in the pool, it is more likely to choose a different data center for every new subtask. When replicas spread across the majority of the data centers (greater than 10), the latency generated by GREEDY and RAND stays about the same, as the difference having one data center more or one less becomes insignificant.

Next in Figure 5, we show the results when the average path length increases from 2 to 10 where the path length refers to the number of subtasks along a path in TG. It can be seen from Figure 5 that the latency is an increasing function of the path length due to the fact that the longer the path, the more the subtasks, which in turn results in bigger execution latency and transmission latency. The latency generated by the proposed algorithm is significantly lower than that of the other two.

Finally Figure 6 investigates the effect of the number of paths existing in TG on latency, where the number is set from 2 to 10. We can observe that the latency grows as an increasing function of the number of paths. This is because when there are more paths, there are more subtasks. As a result, it would be more competitive to obtain resources from data centers, which further leads to a higher latency. The proposed algorithm outperforms the other two algorithms under different number of paths.

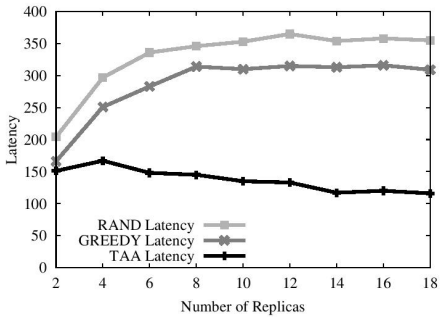

Fig. 4. Latency vs. Number of replicas.

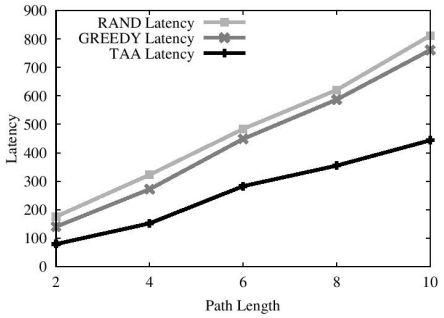

Fig. 5. Latency vs. Path length.

V. Related Work

In this section, we review the related work on big data processing in geo-distributed cloud systems.

Compared to running a large scale data processing task in a single data center, distributed data processing significantly reduces the operation cost and improves the service quality [3, 4]. One of the biggest challenges on distributed data processing is to minimize cost due to its high demand on computation and communication resources. A series of recent work have addressed the cost minimization issue in geo-distributed cloud systems from various perpectives. [10, 13, 15] explored workload control and balancing by taking account of energy consumption and electricity prices in order to reduce the energy expenditure. Chen [13] studied how to reduce electricity cost via optimal virtual machine placement in geo-distributed data centers with heterogeneous pricing scheme and resource constraints. Jiao in [10] investigated the impact of data placement on cost for socially aware services, and proposed a cost-minization data placement algorithm based on graph cuts. Zeng [15] considered task assignment, data placement and data movement jointly to minimize the overall operational cost in geo-distributed data centers.

Another important feature of big data processing in geo-distributed clouds is that data stored in the cloud have multiple replicas located in different data centers [16, 17]. A substantial body of research has been devoted to replica placement to achieve performance optimization [18, 19] from different points of view such as QoS, data correlation and probability trust. In this paper, when processing big data tasks, as there

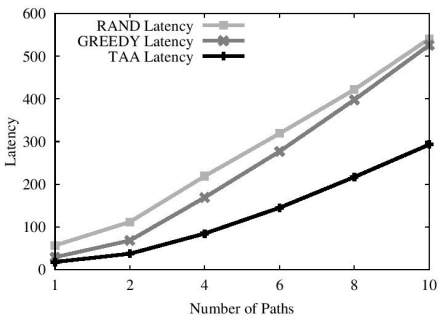

Fig. 6. Latency vs. Number of Paths.

are multiple data replicas available, we can get the input data from any of the replicas, not necessarily the primary one. We assume data replicas are known to users/controllers, thus instead of figuring out where to save data replica, we focus on finding out which replica we should use in order to achieve performance optimization. To the best of our knowledge, this is the first work to investigate the impact of replica selection on data processing in geo-distributed cloud systems.

VI. Conclusion

In this paper, we have investigated the problem of task assignment in geographically distributed cloud systems. We focused on partitioning and deploying data intensive tasks into geographically distributed data centers with an objective of performance optimization. We proposed a heuristic algorithm to solve the proposed NP-hard problem. Extensive simulations have been performend, and the results demonstrate that our method outperforms the random and greedy algorithms significantly. Our future work includes detailed studies on task partition, inter-cloud communications, and deriving variations of the proposed strategy (i.e., load-balanced task assignment in distributed cloud systems), and more constraints such as data regulations that can restrict the storage geographic location.

References

[1] D. Lazer, R. Kennedy, G. King, and A. Vespignani, "The parable of google flu: traps in big data analysis," *Science*, 2014.

[2] Z. Tufekci, "Big questions for social media big data: Representativeness, validity and other methodological pitfalls," *arXiv preprint arXiv:1403.7400*, 2014.

[3] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen, "G-hadoop: Mapreduce across distributed data centers for data-intensive computing," *Future Generation Computer Systems*, 2013.

[4] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers," *Journal of Parallel and Distributed Computing*, 2011.

[5] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, and F. C. Lau, "Dynamic pricing and profit maximization for the cloud with geo-distributed data centers," in *IEEE INFOCOM 2014*. IEEE, 2014.

[6] H. Roh, C. Jung, W. Lee, and D.-Z. Du, "Resource pricing game in geo-distributed clouds," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, 2008.

[8] T. Kraska, G. Pang, M. J. Franklin, S. Madden, and A. Fekete, "Mdcc: Multi-data center consistency," in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013.

[9] Q.-y. Huang and T.-l. Huang, "An optimistic job scheduling strategy based on qos for cloud computing," in *Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on*. IEEE, 2010.

[10] L. Jiao, J. Lit, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," *Proceedings - IEEE INFOCOM*, pp. 28–36, 2014.

[11] J. Baker, C. Bond, J. Corbett, J. J. Furman, A. Khorlin, J. Larson, J. M. Leon, Y. Li, A. Lloyd, and V. Yushprakh, "Megastore: Providing scalable, highly available storage for interactive services." in *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, 2011, pp. 223–234.

[12] K. Katayama and H. Narihisa, "Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem," *European Journal of Operational Research*, 2001.

[13] K. Y. Chen, Y. Xu, K. Xi, and H. J. Chao, "Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems," pp. 3498–3503, 2013.

[14] Amazon-AWS, "https://aws.amazon.com/ec2/pricing," 2016, ONLINE.

[15] D. Zeng, L. Gu, and S. Guo, *Cost Minimization for Big Data Processing in Geo-Distributed Data Centers*. Springer International Publishing, 2015.

[16] D.-W. Sun, G.-R. Chang, S. Gao, L.-Z. Jin, and X.-W. Wang, "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments," *Journal of computer science and technology*, 2012.

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

[18] Z. Ye, S. Li, and X. Zhou, "Gcplace: geo-cloud based correlation aware data replica placement," in *ACM Symposium on Applied Computing*, 2013, pp. 371–376.

[19] N. Kumar and J. Kim, "Probabilistic trust aware data replica placement strategy for online video streaming applications in vehicular delay tolerant networks," *Mathematical and Computer Modelling*, 2013.