An Optimal Task-Scheduling Strategy for Large-Scale Astronomical Workloads using In-transit Computation Model

Xiaoli Wang [1] , Bharadwaj Veeravalli [2] , Omer F. Rana [3]

[1] *School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China, 710071.*

E-mail: wangxiaoli@mail.xidian.edu.cn

[2] *Department of Electrical and ComputerEngineering, The National University of Singapore, 4 Engineering Drive 3, Singapore 117576.*

E-mail: elebv@nus.edu.sg

[3] *School of Computer Science and Informatics, Cardiff University, Queen's Buildings, Newport Road, Cardiff CF24 3AA, UK.*

E-mail: ranaof@cardiff.ac.uk

Received 30 April 2017

Accepted 5 January 2018

Abstract

The Sloan Digital Sky Survey (SDSS) has been one of the most successful sky surveys in the history of astronomy. To map the universe, SDSS uses their telescopes to take pictures of the sky over the whole survey area. Now the total SDSS data volume is larger than 125 TB since every night telescopes produce about 200 GB of data. To improve the processing efficiency of such large-scale astronomical data, we develop an optimal task-scheduling strategy by using in-transit computation model under fog computing. Within the proposed strategy, we design a global optimization technique to derive an optimal load distribution among heterogeneously computational resources. Finally, we conduct various experiments to illustrate the correctness and effectiveness of the proposed strategy. Experimental results show that it can significantly decrease the processing time of large-scale workloads.

Keywords: Task Scheduling, In-transit Computation, Load Distribution, Fog Computing, Genetic Algorithm.

1. Introduction

For millennia, twinkling stars in the night sky have always inspired our curiosity about the universe. Astronomers have launched various scientific sky surveys in the last century attempting to map the universe, over ever-larger areas, to ever-greater depths, and over an ever-increasing range of wavelengths. Among these surveys, the Sloan Digital Sky Survey (SDSS)[1] has created the most detailed three-dimensional maps of the universe ever made, with deep multi-color images of more than one third of the entire night sky, and spectra for more than three million astronomical objects.

SDSS has progressed through several phases. In its first five years of operations, SDSS-I (2000-2005) carried out deep multicolor imaging over 8000 square degrees and measured spectra of more than 700,000 objects. With an ever-growing collaboration, SDSS-II (2005-2008) completed the goals of imaging half the northern sky and mapping the 3-dimensional clustering of one million galaxies and 100,000 quasars. SDSS-III (2008-2014) undertook a major upgrade of the venerable spectrographs[2]. In July 2014, SDSS-IV was launched. It is an extensive imaging and spectroscopic survey of the Northern and Southern sky, using a dedicated 2.5-meter telescope located at southeast New Mexico and the du Pont Telescope at northern Chile[2]. Each telescope is fixed to point directly up at the sky and images a "stripe" of the sky over the course of night. As the Earth rotates, more of the sky becomes visible above the telescopes. Every night the telescopes produce about 200 GB of data. Now the total SDSS data volume is larger than 125 TB[2].

Each image taken by telescopes is composed of myriad pixels, each pixel of which captures the brightness of every tiny point in the sky. But the sky is not made of pixels. Data managers for SDSS requires to extract digitized data from images and process the extracted data to produce information they can use to identify and measure properties of stars and galaxies. It is worth noting that scientists must handle the astronomical workloads as quickly as possible because SDSS astronomers need the information to configure their telescope to work most efficiently during the next dark phase of the moon. If too much time goes by, we might miss the immediate next season of the target objects.

Such large-scale astronomical data could not be processed efficiently without network-based computing systems. One of the key issues in networked computation is obtaining an optimal scheduling strategy, including partition and distribution of workload among computational resources, to achieve shortest processing time. An optimal scheduling strategy depends mainly on the network architecture as well as the number of computational resources and their computing capabilities. Mani and Ghose[3] studied the distribution of divisible workload in a homogeneous linear network and derived recursive equations for obtaining an optimal load partition. Later, asymptotic solutions for homogeneous bus networks were obtained [4]. For heterogeneous star networks, Bharadwaj et al.[5] derived a closed-form expression for an optimal load partition to achieve shortest processing time. The task-scheduling problem turns out to be more difficult when practical issues like the computation and communication start-up overheads are considered. Carroll[6] and Ghanbari[7] studied optimal scheduling strategies for bus and tree networks with arbitrary start-up overheads, respectively. Later on, similar studies have been made on a variety of distributed networks, such as Gaussian, mesh, torus networks[8], complete b-Ary tree networks[9], heterogeneous clusters[10], and cloud computing systems[11].

It should be noted that even the most advanced cloud computing architecture still faces challenge to handle a large amount of astronomical data. Fog Computing is becoming widely known as being the one that extends cloud computing to edge devices and processes directly on the edge devices, thus minimizing the amount of data that is transferred to the cloud[12]. One ubiquitous edge device is network data center. Compared to data centers hosted by cloud providers, network data centers are managed and operated by network providers, which constitute an important part of the current Internet infrastructure. Fog computing can exploit network data centers along the path when workloads are in transit from the user side to cloud data center, so that the spare compute capacities of network data centers could be utilized more efficiently and the processing time of workloads shall be decreased simultaneously. With this idea in mind, Zou et al.[13,14] proposed an in-transit computation infrastructure composed of an ensemble of computational resources, inclusive of a cloud data center and a certain amount of network data centers connecting the source (user) and destination (cloud data center). We employ this in-transit computation model in this work to improve the processing efficiency of astronomical workloads. Our main objective is deriving an optimal task-scheduling strategy for in-transit computation under fog computing so that the processing time of large-scale workloads could be minimized.

The remaining of this paper is organized as follows. Section 2 establishes a novel task-scheduling model for in-transit computation. To solve this model, we accordingly design an effective genetic algorithm in Section 3, which will be evaluated through experiments in Section 4. In the last section, conclusions are obtainable.

2. In-transit Computation under Fog Computing

In this section, we shall first formally define the task-scheduling problem we address and introduce all the notations and definitions used throughout this paper. Then we propose a novel task-scheduling model for in-transit computation of large-scale workloads under for computing.

2.1. *Problem Description*

Suppose that an astronomer needs to compute a workload W_{total}, for example searching among astronomical database or analyzing astronomical data for a certain purpose. The location where workload stores is defined as source s, while the remote cloud data center is defined as destination d. Workload W_{total} transfers from source s to destination d through a network path composed of n in-transit network data centers $\{f_1, f_2, \cdots, f_n\}$ as illustrated in Fig.1. Now the problem lies in how to take full advantage of in-transit computation by deriving an optimal load distribution strategy among $(n+1)$ fog nodes, including n in-transit network data centers and the cloud data center, also denoted as f_{n+1}.

Note that astronomical data, although large in size, are generally partitionable, meaning that they can be partitioned into any number of fractions, or at least fine-grained fractions, and that there are no precedence relationships among these fractions so that they can be independently processed on distributed compute platforms. Hence, workload W_{total} will be partitioned into $(n+1)$ fractions and processed by $(n+1)$ fog nodes independently. Note that source s does not participate in workload computation. We can observe from Fig.1 that after receiving the whole workload from resource s, fog node f_1, an in-transit network data center, keeps a fraction α_1 of W_{total} for itself and transmits the remaining $(W_{total} - \alpha_1)$ to its right immediate neighbor f_2. Similarly, fog node f_i keeps a fraction α_i of W_{total} for itself and transmits the remaining $(W_{total} - \sum_{j=1}^{i} \alpha_j)$ to fog node f_{i+1}. The last node f_{n+1}, also known as the destination cloud data center, upon receiving its load fraction α_{n+1}, does only computation. We have $\sum_{i=1}^{n+1} \alpha_i = W_{total}$ and $0 < \alpha_i < W_{total}$. The total processing time T is the time at which the entire workload W_{total} has been processed. It is given by the maximum of the finish time of all fog nodes. Thus when all fog nodes stop computing at the same time instant, the total processing time T gets minimized.

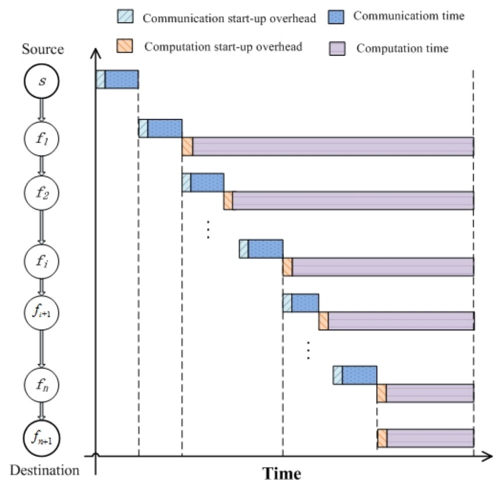

Fig. 1. Timing diagram for in-transit computation

Source s is assumed to start distributing the whole package of workload W_{total} to fog node f_1 at time $t = 0$. It takes node f_i time $\{o_i + z_i \times (W_{total} - \sum_{j=1}^{i} \alpha_j)\}$ to transmit load fraction $(W_{total} - \sum_{j=1}^{i} \alpha_j)$ to node f_{i+1}, and then it cost f_i time $(c_i + w_i \alpha_i)$ to finish computing its assigned load fraction α_i. Here o_i refers to communication start-up overhead of link l_i and c_i represents computation start-up overhead of fog node f_i, while z_i indicates the ratio of the time taken by link l_i to transmit a given workload to that by a standard link and w_i represents the ratio of the time taken by node f_i to compute a given workload to that by a standard compute resource. It may be noted that the computation speed of the last fog node f_{n+1} (i.e., cloud data center) is much faster than that of in-transit nodes. Hence, $w_{n+1} < min\{w_1, w_2, \cdots, w_n\}$.

Let P_i denote the time when node f_i finishes transmitting load fractions to node f_{i+1}. We have

$$\begin{aligned} P_1 &= o_0 + z_0 W_{total} + o_1 + z_1 (W_{total} - \alpha_1), \\ P_i &= P_{i-1} + o_i + z_i \left(W_{total} - \sum_{j=1}^{i} \alpha_j \right), \end{aligned} \quad (1)$$

where $i = 1, \cdots, n$ and $o_0 + z_0 W_{total}$ stands for the transmission time for the total workload distributed from source s to fog node f_1. For the last node f_{n+1}, we have $P_{n+1} = P_n$. Each fog node starts computing only after it finishes transmitting its remaining workload to its immediate neighbor. The finish time of node f_i can be written as,

$$T_i = P_i + c_i + w_i \alpha_i, \quad i = 1, 2, \cdots, n+1. \tag{2}$$

Finally, we can obtain the processing time T of the total workload as $T = \max\{T_1, T_2, \cdots, T_{n+1}\}$.

2.2. *In-transit Computation Model*

Here we formulate a new in-transit computation model under fog computing.

$$\min T(A) \; = \; \min_A \{\max\{T_1, T_2, \cdots, T_{n+1}\}\}.$$

where

(1) $A = \{\alpha_1, \cdots, \alpha_{n+1}\}$;
(2) $T_i = P_i + c_i + w_i \alpha_i$ with $i = 1, 2, \cdots, n+1$;
(3) $P_1 = o_0 + z_0 W_{total} + o_1 + z_1 (W_{total} - \alpha_1)$;
(4) $P_i = P_{i-1} + o_i + z_i \; (W_{total} - \sum_{j=1}^{i} \alpha_j)$ with $i = 1, 2, \cdots, n$;
(5) $P_{n+1} = P_n$.

subject to:

(i) $\alpha_i \in A, \; 0 < \alpha_i < W_{total}, \; i = 1, 2, \cdots, n+1$;
(ii) $\sum_{i=1}^{n+1} \alpha_i = W_{total}$.

There are $(n+1)$ variables involved in this model. Constraints (i) and (ii) indicate that load fractions assigned on fog nodes should be nonnegative and not larger than the entire workload, and that the sum of all load fractions equals the entire workload.

3. Optimal Task-Scheduling Strategy

In this section, we shall design a Genetic Algorithm (GA) searching for an optimal load partition $A = \{\alpha_1, \alpha_2, \cdots, \alpha_{n+1}\}$ for the proposed in-transit computation model. We select GAs to solve our model because GAs have been proven to be a promising technique for combinatorial optimization problems, especially for task-scheduling problems.

3.1. *Encoding and Genetic Operators*

The key point of finding an optimal solution by using GAs is to develop an encoding scheme that can represent the problem to be solved directly while satisfying the problem constraints easily. In this paper, an individual is real coded directly as $I = (\alpha_1, \alpha_2, \cdots, \alpha_{n+1})$. For a given individual I, if $\exists i, \alpha_i \leqslant 0$ or $\sum_{i=1}^{n+1} \alpha_i > W_{total}$, then this individual I violates the constraints of the proposed model and it is considered to be an invalid individual.

As a simple example, assume that there are $n = 6$ fog nodes in the system, inclusive of 5 in-transit network data centers along the path from source to destination (cloud data center). The size of the entire workload is 1000 units. A possible encoding scheme is given as follows:

$$\begin{aligned} I &= (\alpha_1, \; \alpha_2, \; \alpha_3, \; \alpha_4, \; \alpha_5, \; \alpha_6) \\ &= (94, \; 78, \; 60, \; 68, \; 50, \; 650). \end{aligned}$$

We observe that $\forall i, \alpha_i > 0$ and $\sum_{i=1}^{n+1} \alpha_i = W_{total} = 1000$, thus individual I is a valid individual as it satisfies all constraints in our model. It is worth noting that the last fog node f_6 is assigned with the largest load fraction $\alpha_6 = 650 > \max\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$. This is because the last fog node represents the cloud data center with much higher compute capability than other fog nodes (network data centers). This is also validated from our experimental results as shown in Section 4.

According to our proposed in-transit computation model, we have a special constraint as $\sum_{i=1}^{n+1} \alpha_i = W_{total}$. Therefore, if we adopt two-point crossover, it may produce invalid offsprings. Hence, we should normalize the newly generated individuals to ensure that the total value of all genes equals the entire workload W_{total}.

We adopt two-point mutation on offsprings generated by crossover according to a user-definable mutation probability. This probability should be set low; otherwise, the search will turn into a primitive random search. In detail, we randomly generate two integers p and q satisfying that $1 \leqslant p < q \leqslant (n+1)$, then exchanging genes α_p and α_q of individual I. It can be expected that offsprings generated by this mutation operator satisfy all of the constraints in our proposed model by default.

3.2. Local Search

To accelerate the convergence of the proposed GA, we introduce a local search operator in this paper. The main idea is to transfer proper size of load from the fog node with the longest processing time $T_{\max}$ to the one with the shortest processing time $T_{\min}$, so that all of the fog nodes will eventually stop computing at the same time instant. The process of the local search operator is given as follows.

Step 1 For a given individual $I = (\alpha_1, \alpha_2, \cdots, \alpha_{n+1})$, let $P_0 = o_0 + z_0\, W_{total}$.

Step 2 **FOR** $(i = 1, 2, \cdots, n)$
Let $P_i = P_{i-1} + o_i + z_i\, (W_{total} - \sum_{j=1}^{i} \alpha_j)$ and $T_i = P_i + c_i + w_i \alpha_i$.
ENDFOR

Step 3 Let $T_{n+1} = P_n + c_{n+1} + w_{n+1}\alpha_{n+1}$.

Step 4 Among all fog nodes $\{f_1, f_2, \cdots, f_{n+1}\}$, find node $f_{\max}$ with the longest processing time $T_{\max}$ and fog node $f_{\min}$ with the shortest processing time $T_{\min}$. Calculate their time difference by $\Delta = T_{\max} - T_{\min}$.

Step 5 Let $\beta = \Delta / \max\{z_{\max}, z_{\min}\}$. Update individual I by $\alpha_{\max} = \alpha_{\max} - \beta$ and $\alpha_{\min} = \alpha_{\min} + \beta$.

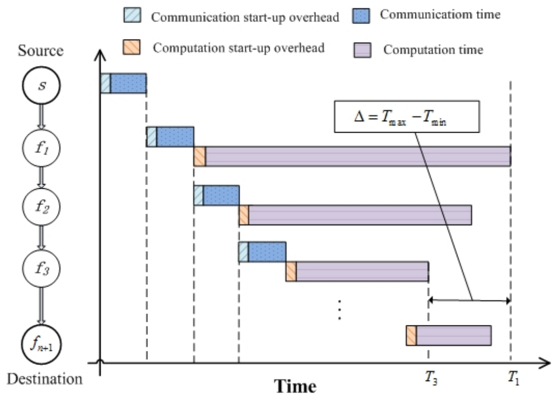

Fig. 2. Timing diagram before applying local search

Figure 2 shows a timing diagram that corresponds to an individual before applying local search. As illustrated in Fig. 2, node f_1 has the longest processing time T_1 and f_3 has the shortest processing time T_3. Thus $f_{\max} = f_1$ and $f_{\min} = f_3$. After load balancing between f_1 and f_3 by local search operator, a possible timing diagram is shown in Fig. 3. It can be observed that the time difference between T_1 and T_3 illustrated in Fig. 3 becomes much smaller than that in Fig.2. Hence, the total processing time of the entire workload would be decreased.

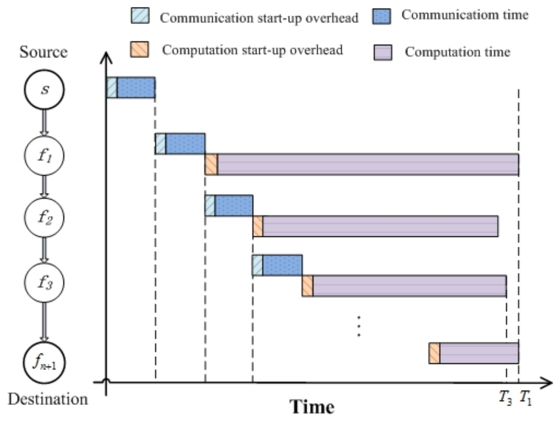

Fig. 3. Timing diagram after applying local search

3.3. Framework of the Proposed Algorithm

Once encoding scheme is defined, a GA initializes a population of individuals and then improves them through repetitive applications of genetic operators, including crossover, mutation, local search, and selection. Given workload W_{total}, population size *Popsize*, crossover probability p_{cros}, mutation probability p_{mut}, elitist number $E = 5$, and stop criterion, the framework of our proposed GA is given as follows.

Step 1 (**Initialization**) Randomly generate *Popsize* individuals as initial population $Pop(0)$ according to the encoding scheme. For each $I \in Pop(0)$, compute processing time T of workload W_{total} and take $1/T$ as the fitness value of I. Let generation number $t = 0$.

Step 2 (**Crossover**) Select *Popsize* individuals into the crossover pool from $Pop(t)$ by roulette wheel selection. Apply two-point crossover on each pair of parents selected from the crossover pool according to p_{cros} and then normalize the newly generated offsprings to ensure that $\sum_{i=1}^{n+1} \alpha_i = W_{total}$. All offsprings constitute a set denoted by $O_1(t)$.

Step 3 (**Mutation**) Apply two-point mutation on each of the selected individuals from $O_1(t)$ according to p_{mut}. All newly generated offsprings constitute a set denoted by $O_2(t)$.

Step 4 (**Local Search**) Apply local search operator on each individual in set $O_1(t) \cup O_2(t)$.

Step 5 (**Selection**) Select the best E individuals for the next population $Pop(t+1)$ from set $Pop(t) \cup O_1(t) \cup O_2(t)$. Select the remaining $Popsize - E$ individuals for $Pop(t+1)$ by roulette wheel selection also from set $Pop(t) \cup O_1(t) \cup O_2(t)$. Let $t = t+1$.

Step 6 (**Stopping Criteria**) If a fixed number of generations reached, then stop and return the best individual I in the current population; otherwise, go to Step 2.

4. Experimental Results and Analysis

As we mentioned earlier, every night the telescopes of SDSS, including the primary 2.5m telescope, 0.5m photometric telescope, and 10 micron all sky scanner, produce about 200 GB of raw imaging data. In our simulation, we have considered this actual data size and normalized it into 10000 units. Then a series of operators are required to process these large-scale telescope imaging data under fog computing, ultimately producing a variety of products including images with instrumental signatures removed, a photometric solution for the night, and a catalog of objects found in the data. The computation speed of each fog node processing every unit of astronomical data is recorded in Table 1.

In each run of our proposed GA, the following parameters are set: $Popsize = 100$, crossover probability $p_{cros} = 0.8$, mutation probability $p_{mut} = 0.02$, elitist number $E = 5$, and stop criterion $t = 2500$.

4.1. Correctness Evaluation

We conduct two experiments to validate the correctness of our proposed GA. In each experiment, we employ a fog computing system with 20 fog nodes, including 19 in-transit network data centers and one cloud data center. In the first experiment, we fix system parameters as given in Table 1 and vary the workload size from 500 to 2500 units. Figs. 4 and 5 collect the experimental resutls. In the second experiment, we fix workload size as $W_{total} = 1000$ unites and vary the network scenarios where the compute capability of cloud data center is q times more powerful than that of in-transit fog nodes, where $q \in \{5, 10, 15, 20, 25\}$. Figs. 6 and 7 record the results.

We observe from Figs.5 and 7 that all fog nodes stop computing at the same time for every test. Hence, the proposed algorithm can obtain an optimal task-scheduling strategy that achieves minimum processing time. As expected, we can see from Figs.4 and 6 that the load fraction assigned to the last node is much larger than that assigned to other fog nodes because the last node represents a cloud data center with high-performance capability, while the other nodes are in-transit network data centers with relatively low-performance capabilities.

4.2. Performance Evaluation

To evaluate the effectiveness of the proposed algorithm, we make two comparisons between our algorithm, labeled as "In-transit computation" in the experiment results, and the task-scheduling strategy with only cloud data center performing computation, labeled as "No In-transit computation." Figure 8 records the comparison results obtained for different workloads ranging from 1000 to 10000 units, while Fig. 9 collects the experimental results obtained under different network scenarios with network size varying from 10 to 30.

It can be observed from Figs.8 and 9 that the processing time obtained by "In-transit computation" strategy is much less than that by "No in-transit computation" strategy for each test, and that the time difference between them grows with increasing workload size and network size. As shown in Fig.8, when workload size is as large as 10000 units in our experiment, the processing time obtained by the "In-transit computation" strategy shows a gain of 65.4% compared to "No In-transit computation" strategy. Also, it can be seen from Fig.9 that when there are 30 fog nodes in the network system, the "In-transit computation" strategy reduces the processing time by over 55.3% compared to "No In-transit computation" strategy. Therefore, it is clear that our proposed algorithm for in-transit computation can derive an optimal task-scheduling strategy that significantly decreases the processing time of large-scale workloads. This holds even in cases where in-transit fog nodes are not very powerful in computation compared to the cloud data center.

Table 1. Parameters for fog nodes

p_i	o_i	c_i	z_i	w_i	p_i	o_i	c_i	z_i	w_i	p_i	o_i	c_i	z_i	w_i
p_1	5.40	9.87	0.14	29.71	p_8	5.03	2.21	0.14	37.26	p_{15}	7.00	6.48	0.20	32.62
p_2	6.97	1.40	0.19	30.52	p_9	3.06	4.18	0.15	24.33	p_{16}	9.40	6.84	0.12	21.28
p_3	7.22	6.96	0.19	39.53	p_{10}	4.03	6.49	0.19	36.73	p_{17}	2.07	7.30	0.20	35.44
p_4	6.50	2.50	0.17	20.51	p_{11}	1.63	3.39	0.11	27.39	p_{18}	8.80	4.40	0.18	33.42
p_5	2.34	3.38	0.13	30.63	p_{12}	9.30	4.34	0.12	30.37	p_{19}	7.93	8.12	0.17	34.67
p_6	4.23	9.35	0.12	33.70	p_{13}	3.59	9.05	0.14	39.52	p_{20}	2.46	9.28	0.16	0.30
p_7	9.71	3.14	0.16	36.62	p_{14}	1.39	7.68	0.15	30.99	—	—	—	—	—

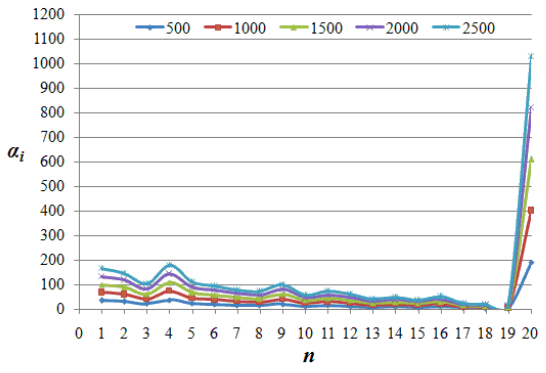

Fig. 4. Optimal load partitions for different workloads

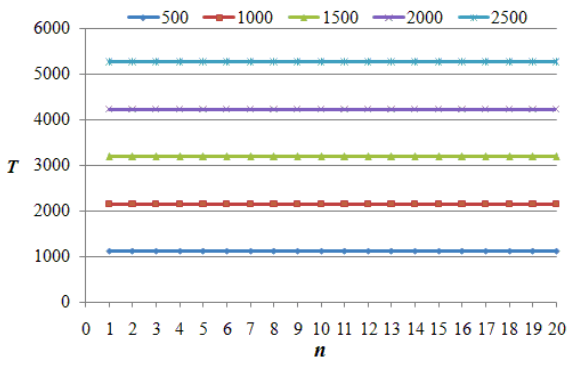

Fig. 5. Finish times of fog nodes for different workloads

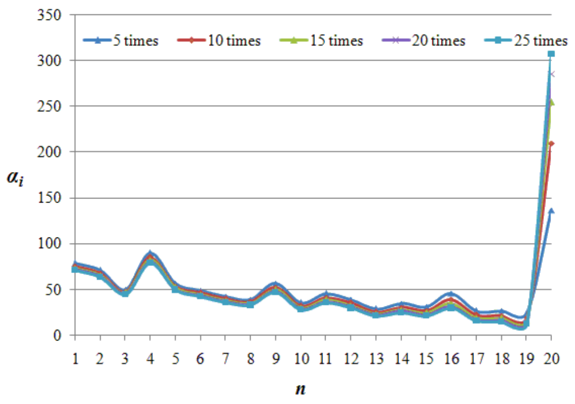

Fig. 6. Optimal load partitions under different networks

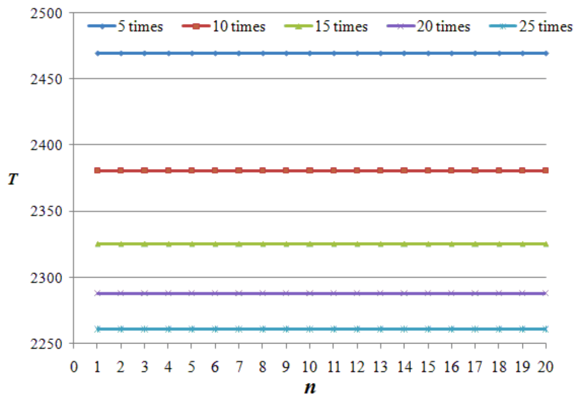

Fig. 7. Finish times of fog nodes under different networks

5. Conclusions

In this paper, we have addressed the task-scheduling problem for in-transit computation of large-scale astronomical workloads under fog computing. We built a novel task-scheduling model and proposed a genetic algorithm to derive an optimal load distribution strategy. We have explicitly considered the astronomical imaging data taken by telescopes of SDSS as our reference size of data volume in our extensive experiments. We demonstrated that the proposed algorithm could significantly decrease the processing time of large-scale workloads by in-transit computation. An important and immediate useful extension to the study posed in this paper is considering complex networks with more than one

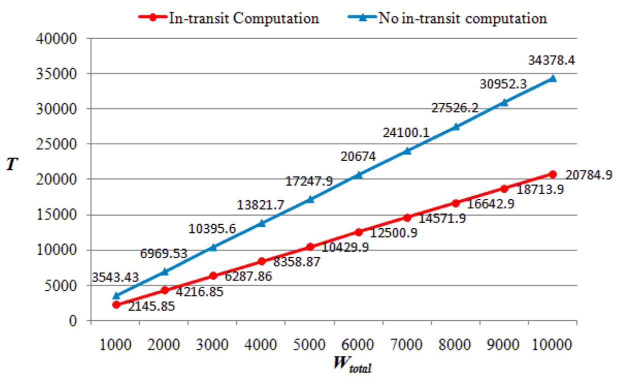

Fig. 8. Comparison between in-transit computation and no in-transit computation for different workloads

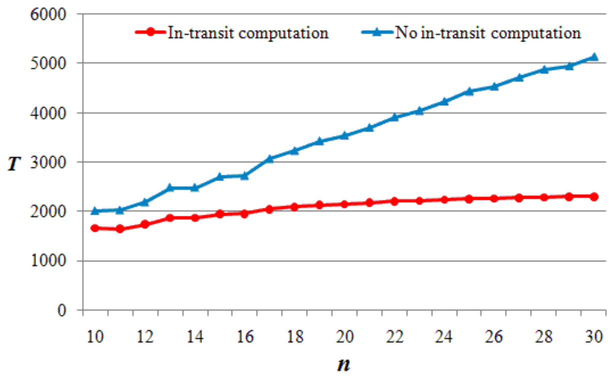

Fig. 9. Comparison between in-transit computation and no in-transit computation under different network scenarios

network path available from source to destination, and deriving an optimal task-scheduling strategy, including selection of in-transit nodes along the path and distribution of workloads among them.

Acknowledgments

This work was initiated and conducted jointly when the second author visited Department of CS, Cardiff University, in September 2016, and supported by MOE Tier-1 grant No. R-263-000-C14-112. This study on data distribution aspects serves as a preliminary study for possible extensions to take care of security aspects while designing IOT based platforms to handle large-scale data processing for intelligent transportation system in Singapore. The first author would like to acknowledge the funding supported by National Natural Science Foundation of China (No.61402350, No.61472297, and No.61572391).

References

1. http://www.sdss.org/
2. S. Collaboration, F. D. Albareti, C. A. Prieto, et al. The Thirteenth Data Release of the Sloan Digital Sky Survey: First Spectroscopic Data from the SDSS-IV Survey MApping Nearby Galaxies at Apache Point Observatory. (2016).
3. V. Mani, D. Ghose, Distributed computation in linear networks: Closed-form solutions, *IEEE Trans. Aero. Elec. Sys.* **30** (1994) 471–483
4. D. Ghose and V. Mani, Distributed Computation with Communication Delays: Asymptotic Performance Analysis, *J. Parallel and Distrib. Computing.* **23** (1994) 293–305.
5. V. Bharadwaj, D. Ghose, and V. Mani, Optimal Sequencing and Arrangement in Distributed Single-Level Networks with Communication Delays, *IEEE Trans. Parallel Distrib. Syst.* **5** (1994) 968–976.
6. T. E. Carroll, D. Grosu, Strategyproof mechanisms for scheduling divisible loads in bus-networked distributed systems, *IEEE Trans. Parallel Distrib. Syst.* **19** (2008) 1124–1135
7. S. Ghanbari, M. Othman, M. R. A. Bakar, W. J. Leong, Multi-objective method for divisible load scheduling in multi-level tree network, *Future Gener. Comput. Sys.* **54** (2016) 132–143
8. Z. Zhang, T. G. Robertazzi, Scheduling Divisible Loads in Gaussian, Mesh and Torus Network of Processors, *IEEE Trans. Comput.* **64** (2015) 3249–3264
9. C. Y. Chen, C. P. Chu, Novel Methods for Divisible Load Distribution with Start-Up Costs on a Complete b-Ary Tree, *IEEE Trans. Parallel Distrib. Syst.* **26** (2015) 2836–2848
10. K. Li, X. Tang, B. Veeravalli, et al, Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems, *IEEE Trans. Comput.* **64** (1) (2015) 191–204.
11. W. Lin, S. Xu, L. He, et al, Multi-resource scheduling and power simulation for cloud computing, *Information Sciences.* **397** (2017) 168–186.
12. B. P. Rimal, M. Maier, Workflow Scheduling in Multi-Tenant Cloud Computing Environments, *IEEE Trans. Parallel Distrib. Syst.* **28** (1) (2017) 290–304.
13. M. Zou, A. R. Zamani, J. Diaz-Montes, et al. Leveraging In-Transit Computational Capabilities in Federated Ecosystems, in *IEEE Symposium on Service-Oriented System Engineering*, (2016), pp. 81–90.
14. A. R. Zamani, M. Zou, J. Diaz-Montes, et al. A computational model to support in-network data analysis in federated ecosystems, in *Future Generation Computer Systems*, (2017). In press.