Cost-Effective Task Scheduling for Collaborative Cross-Edge Analytics

ZHAO Kongyang[1], GAO Bin[2], ZHOU Zhi[1]

(1. Sun Yat-sen University, Guangzhou 510275, China;
2. National University of Singapore, Singapore 119077, Singapore)

Abstract: Collaborative cross-edge analytics is a new computing paradigm in which Internet of Things (IoT) data analytics is performed across multiple geographically dispersed edge clouds. Existing work on collaborative cross-edge analytics mostly focuses on reducing either analytics response time or wide-area network (WAN) traffic volume. In this work, we empirically demonstrate that reducing either analytics response time or network traffic volume does not necessarily minimize the WAN traffic cost, due to the price heterogeneity of WAN links. To explicitly leverage the price heterogeneity for WAN cost minimization, we propose to schedule analytic tasks based on both price and bandwidth heterogeneities. Unfortunately, the problem of WAN cost minimization underperformance constraint is shown non-deterministic polynomial (NP)-hard and thus computationally intractable for large inputs. To address this challenge, we propose price- and performance-aware geo-distributed analytics (PPGA) , an efficient task scheduling heuristic that improves the cost-efficiency of IoT data analytic jobs across edge datacenters. We implement PPGA based on Apache Spark and conduct extensive experiments on Amazon EC2 to verify the efficacy of PPGA.

Keywords: collaborative cross-edge analytics; Internet of Things; task scheduling

DOI: 10.12142/ZTECOM.202102003

https://kns.cnki.net/kcms/detail/34.1294.TN.20210607.1503.002.html, published online June 9, 2021

Manuscript received: 2021-04-09

Citation (IEEE Format): K. Y. Zhao, B. Gao and Z. Zhou. “Cost-effective task scheduling for collaborative cross-edge analytics,” *ZTE Communications*, vol. 19, no. 2, pp. 11 – 19, Jun. 2021. doi: 10.12142/ZTECOM.202102003.

1 Introduction

With the fast burgeoning as well as accelerating convergence of artificial intelligence (AI) and the Internet of Things (IoT), unprecedented prosperity of AI of Things or AI-empowered IoT applications has emerged recently. This new trend is coined as AIoT, which pushes the frontier of AI from the centralized cloud to the ubiquitous IoT devices, paving the last mile delivery of AI capabilities. Recently, AIoT has gained mounting attention from industrial giants and boosted many intelligent applications as exemplified by intelligent personal assistants, personalized shopping recommendations, video surveillance and smart home appliances, which are significantly changing our daily life. For example, by bringing live video analytics to urban traffic management, Microsoft[1], Baidu, and Alibaba remarkably improve commuting efficiency and safety in Bevellue (a city outside of Seattle), Beijing, and Hangzhou, respectively.

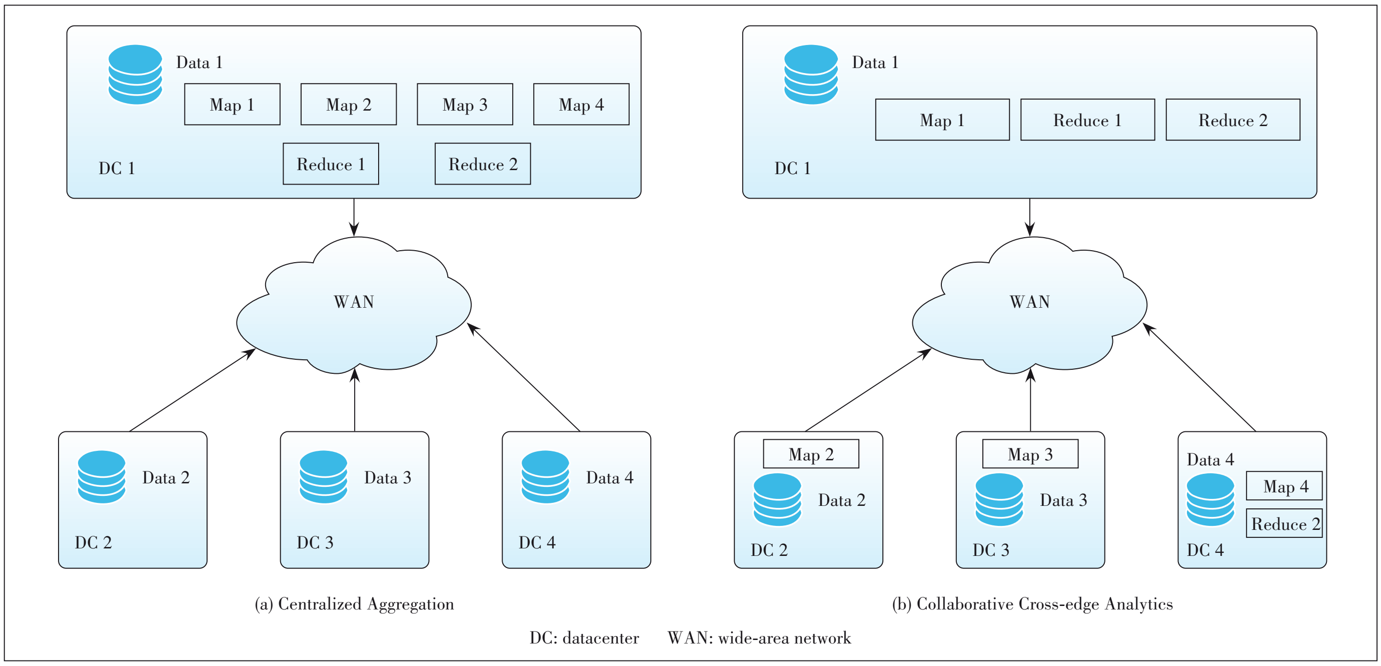
Specifically, for many emerging collaborative AIoT appli-

This work was supported in part by the National Natural Science Foundation of China under Grant No.61802449 and the Guangdong Natural Science Funds under Grant No. 2021A1515011912.

cations spanning across multiple regions and edge datacenters[2–3], the raw data generated at each datacenter can be huge, redundant and unlabeled[4]. For example, for urban traffic control and management, the raw data born at each edge datacenter contains only the ID and timestamp of the vehicles traversing this region, but not the global trajectory of each vehicle, which can be used to learn the vehicle behavior. While for the smart retail application, the raw data of each unmanned store only contains the shopping record of each user and item information of each good, rather than the user preference and item popularity data that can be used to learn the user's shopping behavior. To extract the labeled data (e.g., vehicle trajectory, user preference and item popularity data over multiple stores) that can be used to train an AI model in such collaborative cross-region scenarios, we need to perform preprocessing to the raw data stored at cross-region edge datacenters to obtain some statistical query results (i.e., labeled data, such as the item popularity over multiple stores), employing data-parallel frameworks such as Map-Reduce, Spark and Flink.

In this paper, we propose to empower AIoT by optimizing data preprocessing across multiple edge datacenters, which is referred to as collaborative cross-edge analytics[5–6]. With collaborative cross-edge analytics, we push the computation tasks of both the input stage (e.g., Map) and the output stage (e.g., Reduce) of the analytics framework (e.g., Map-Reduce) to the widespread edge datacenters where the data was born, instead of collecting raw data to the master datacenter as shown in Fig. 1a. Fig. 1b shows an example of collaborative cross-edge analytics, in which both Map and Reduce tasks are moved to the place where the data is stored. Since in the output stage tasks are distributed across edge datacenters, the intermediate data after the input stage still needs to be shuffled across edge datacenters for final analytic results. Unfortunately, due to the aforementioned bandwidth heterogeneity of the cross-edge wide-area network (WAN) links, the duration of different cross-edge shuffle transfers can be greatly diverse. To cut down the finish time of the shuffle phase by mitigating the stragglers (i.e., the slowest ones), a WAN bandwidth-aware task placed for the output stage has been proposed.

Apart from the performance, another primary objective for collaborative cross-edge analytics is the WAN bandwidth cost. Compared with the sufficient intra-edge network bandwidth, cross-edge network bandwidth is far scarcer and more expensive. Motivated by the exacerbating shortage of WAN bandwidth, recent efforts have been made to minimize the amount of data that traverses the expensive WAN. For example, Pixida[7], Geode[8] and Iridium[6] reduce the WAN bandwidth usage by cost-efficient task placement, i.e., placing more tasks at datacenters with more input data. However, for collaborative cross-edge analytics, is the reduction of WAN bandwidth usage really translated into cost savings? The answer is probably "No", unfortunately. The rationale is that the price of WAN bandwidth usage (in US dollar per GB, in terms of the bill for one unit of data transfer) for different WAN links also shows diversity. Currently, Infrastructure-as-a-Service (IaaS) cloud service providers such as Amazon EC2, Microsoft Azure and Google Cloud Engine charge outgoing bulk data transfer from different datacenters with various prices, as illustrated in Table 1. The price diversity clearly indicates that using less WAN bandwidth does not



▲**Figure 1. Illustration of collaborative cross-edge analytics**

▼Table 1. Price heterogeneity (in US dollar per GB) of outgoing WAN bandwidth usage at different regions of Amazon EC2

Location	Price
Frankfurt	0.02
Iceland	0.02
London	0.02
Northern California	0.02
Northern Virginia	0.02
Ontario	0.02
Oregon	0.02
Tokyo	0.02
Mumbai	0.086
Seoul	0.09
Singapore	0.09
Sydney	0.09
Sao Paulo	0.16

WAN: wide-area network

necessarily mean less usage cost. Even worse, our statistical study on the Amazon AWS platform in Section 2 shows that, for datacenters connected to faster WAN links, the WAN bandwidth usage price is usually higher. Intuitively, with such a performance-cost tradeoff, existing performance driven task scheduling for collaborative cross-edge analytics would greatly increase the usage cost of WAN bandwidth.

Motivated by the price heterogeneity as well as the performance-cost tradeoff for the WAN bandwidth, we argue that price-awareness should be brought to task scheduling of collaborative cross-edge analytics. Given the non-coincidence between performance and bandwidth usage cost, we propose to minimize WAN bandwidth usage cost under performance guarantee, i. e., response time constraint. This problem is proven NP-hard, and we design PPGA, an efficient greedy-based scheduler that explicitly leverages both price and bandwidth heterogeneity, to place tasks across geo-distributed edge datacenters. PPGA is implemented based on Apache Spark and deployed across five Amazon EC2 regions. Extensive experiments using realistic benchmarks and workloads have shown that PPGA can reduce the WAN cost of collaborative cross-edge by up to 31.6%.

2 Background and Motivation

2.1 Collaborative Cross-Edge Analytics

We consider a cross-edge infrastructure, in which multiple edge datacenters at different regions are interconnected by a WAN and to host AIoT services locally. With such a setup, data are naturally born at each edge cloud in a distributed fashion, while a dataset (e.g., shopping records for a smart retail service) generally contains multiple data partitions that are originated across different edge datacenters. Traditionally, to analyze such a dataset, the centralized aggregation approach is adopted as shown in Fig.1a, i.e., all the data partitions are transferred to a master datacenter to be processed.

With collaborative cross-edge analytics[6], a job query script is first converted into a direct acyclic graph (DAG) of consecutive stages, each stage consisting of many parallel tasks that can be executed at different DCs. For the example of the Map-Reduce query illustrated in Fig. 1, the corresponding DAG contains two stages: Map and Reduce. Tasks of the input stages (Map in Fig. 1) are pushed to the place where the data partitions are stored, so that data can be processed locally. The output of input stages, called intermediate data, is then shuffled to output stages (e. g., Reduce in Fig. 1) to compute the final query results. Since consecutive stages are linked by data dependency, a stage (e.g., Reduce) can be started only after it has received all the intermediate data from parent stages (e.g., Map). For input stage tasks, the commonly adopted approach is "site-locality", i.e., their locations are the same as the locations of their input data. As a result of data locality and in-memory caching, input stages can be finished extremely quickly.

The scarcity of WAN bandwidth has resulted in the high usage price of WAN bandwidth. Currently, IaaS cloud providers such as Amazon AWS, Microsoft Azure and Google Cloud Engine charge the WAN bandwidth usage while leave the intra-datacenter bandwidth usage for free. Moreover, the price of outgoing WAN bandwidth usage at different datacenters also exhibits moderate diversity, due to various capital expenditure (Cap. Ex) and operational expenditure (Oper. Ex) at different regions. As illustrated in Table 1, for the case of Amazon EC2, the outgoing WAN bandwidth usage price at different datacenters varies from $0.02 per GB to $0.16 per GB.

2.2 Motivation

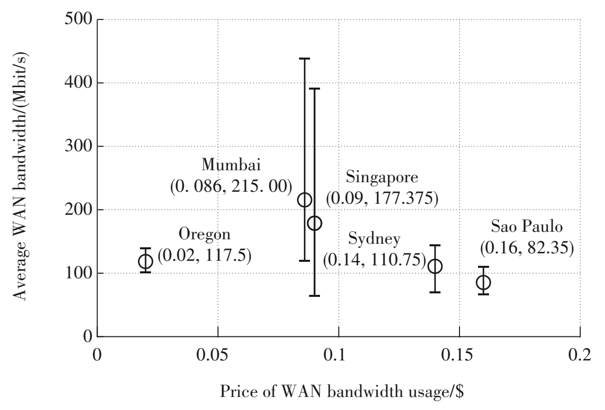
The tradeoff between the performance and the cost is shown in this section. The above example motivates us to take advantage of the heterogeneities of usage price and bandwidth of the WAN, and then further optimize the cost and performance of the shuffle phase. Clearly, to minimize the cost of WAN bandwidth usage, more Reduce tasks should be placed at datacenters with higher usage cost of out-going WAN bandwidth, so that more intermediate data would be transferred out from datacenters with lower usage price of WAN bandwidth. On the other hand, to minimize the finish time of the shuffle phase, more Reduce tasks should be placed at datacenters connected to faster inter-datacenter WAN links. Thus, if it is the idle case that datacenters with higher usage price of WAN bandwidth are connected to faster WAN links, placing more Reduce tasks at these datacenters optimizes the cost and performance simultaneously.

Unfortunately, however, our empirical measurements based on Amazon EC2 show that datacenters with higher usage price of WAN bandwidth are not necessarily connected to faster WAN links. Specially, we first measure the available inter-datacenter WAN bandwidth between 5 regions of Amazon EC2: Oregan (North America), Singapore (Asia-Pacific), Sao Paulo (South America), Sydney (Oceania) and Mumbai (Asia-Pacific). We then plot the usage price of WAN bandwidth, as well as the average bandwidth of the WAN links connected to each region in Fig. 2. Interestingly, we observe that there is an approximate negative correlation between the average WAN bandwidth and the usage price. Specifically, for the 4 locations of Mumbai, Singapore, Sydney and Sao Paulo, there is a clear negative correlation between the average WAN bandwidth and the usage price. This observation convincingly demonstrates the inherent tradeoff between performance and cost of the shuffle phase.

To navigate the cost-performance tradeoff, we propose to minimize the cost of WAN bandwidth usage while enforcing the shuffle phase to be finished within a pre-defined deadline. The rationale of bounding the finish time rather than minimizing it is that, for some realistic analytic jobs, the analytic result is used by a future event or decision making with a time lag. Therefore, finishing the analytic job within the time lag would not compromise future events or decision making. Towards the goal of minimizing WAN bandwidth usage with bounded finish time of the shuffle phase, we propose PPGA, a task scheduler that leverages the heterogeneities of usage price as well as the bandwidth of the cross-edge WAN, in the next section.

3 Model and Optimization for Collaborative Cross-Edge Analytics

In this section, we present the formulation and optimization for WAN price and performance-aware task scheduling of collaborative cross-edge analytics.



WAN: wide-area network

▲Figure 2. Usage price of outgoing WAN bandwidth versus the average outgoing WAN bandwidth of 5 Amazon EC2 regions

3.1 Infrastructure

We consider an AIoT service provider running AIoT services and originate data on a set of N geographically dispersed edge datacenters, denoted by $\mathcal{D} = \{1, 2,..., N\}$. Here each edge datacenter can be a private datacenter, a public multi-tenant datacenter (e.g., Amazon EC2), or a public colocation datacenter (e.g., Equinix). The computational capacity of each datacenter $i \in \mathcal{D}$ is denoted as C_i, in terms of the maximal number of computational tasks that can be executed in a parallel manner.

With collaborative cross-edge analytics, a job query script is first converted into a direct acyclic graph (DAG) of consecutive stages[9]. Then, an intuition globally schedules the stages in the DAG all together. However, such global scheduling requires prior knowledge of the task characteristics of all stages, as well as the long-term bandwidth availability. Though they are predictable, an exact prediction without error is difficult. Therefore, scheduling the DAG is not practical[10]. Instead, we schedule tasks stage by stage in an online fashion, i.e., we choose to schedule the tasks within the same stage to geo-distributed datacenters, rather than considering all the tasks in the DAG. Note that the default task scheduler of Spark also adopts the stage-by-stage method. Though such online scheduling may not be globally optimal, it enables adjustments that can be made on the fly to better cater to the dynamic job progress and network environment.

For a given stage of the job DAG, we use $\mathcal{J} = \{1, 2,..., M\}$ to denote the set of M Reduce tasks which can be scheduled to different edge datacenters and executed in parallel. To denote the placement scheduling of those Reduce tasks, we introduce binary variables $x_{ij} \in \{0, 1\}, \forall i \in \mathcal{D} = \{1, 2,..., N\}, \forall j \in \mathcal{J} = \{1, 2,..., M\}$. Specifically, if the task $j \in \mathcal{J}$ is allocated to edge datacenter $i \in \mathcal{D}$, then $x_{ij} = 1$, otherwise $x_{ij} = 0$. Since each task $j \in \mathcal{J}$ can be scheduled to one and only one edge datacenter, we have the following placement constraint:

$$\sum_{i \in \mathcal{D}} x_{ij} = 1, \forall j \in \mathcal{J}. \tag{1}$$

Besides the above placement constraint, the task scheduling is also constrained by the computational capacity of each edge datacenter. That is, the number of Reduce tasks allocated to each edge datacenter i cannot exceed the computational capacity C_i of datacenter i:

$$\sum_{j \in \mathcal{J}} x_{ij} \leqslant C_i, \forall i \in \mathcal{D}. \tag{2}$$

For Reduce task $j \in \mathcal{J}$, it has to gather intermediate data from other edge datacenters to reduce operation. Here we use S_{ij} to denote the amount of intermediate data stored at datacenter i and to be collected by Reduce task j. For the shuffle phase where intermediate data are transferred across edge datacenters and given the source edge datacenter i and destination edge datacenter k, the amount of intermediate data transferred from edge datacenter i to edge datacenter k can be denoted as $\sum_{j \in \mathcal{D}} S_{ij} x_{kj}$.

3.2 Performance of Shuffle Phase

Given the available pair-wise bandwidth of the WAN link from edge datacenter i to edge datacenter k, B_{ik}, and together with the amount of intermediate data traversing this WAN link, the duration of the corresponding intermediate data transfer can be denoted as $\frac{\sum_{j \in \mathcal{D}} S_{ij} x_{kj}}{B_{ik}}$. Since the performance, in terms of the finish time of the intermediate data shuffle phase, is determined by the slowest intermediate data transfer, it can be formulated as:

$$z = \max_{i \in \mathcal{D}} \max_{k \in \mathcal{D}} \frac{\sum_{j \in \mathcal{D}} S_{ij} x_{kj}}{B_{ik}}. \tag{3}$$

Here if $i = k$, it means that the corresponding intermediate data transfer is an intra-edge transfer rather than a cross-edge WAN transfer. Since the intra-edge network bandwidth typically far more overweighs the cross-edge WAN bandwidth, the former can be finished very soon.

If we enforce the shuffle phase to be finished before the deadline W, each intermediate data transfered from edge datacenter i to edge datacenter k should be finished within this deadline W, that is:

$$\frac{\sum_{j \in \mathcal{D}} S_{ij} x_{kj}}{B_{ik}} \leqslant W, \forall i,k \in \mathcal{D}. \tag{4}$$

3.3 Cost of WAN Bandwidth Usage

Unlike the moderately sufficient intra-edge bandwidth, the cross-edge WAN bandwidth represents a scarce resource that incurs high capital and operational expenditure. For this reason, internet service providers (ISP) and IaaS edge cloud providers typically charge WAN bandwidth usage according to the bytes transferred, i.e., the amount of data transferred across the WAN. Specifically, for IaaS edge cloud providers such as Amazon AWS, Microsoft Azure and Google Cloud Engine (GCE), the price for inter-datacenter WAN transfer is dependent on the source datacenter of the transfer, and different source datacenters usually have various prices. Here we use P_i to denote the WAN bandwidth usage price for traffic going out of datacenter i, and P_i exhibits geographical diversity across datacenters. Given the amount of $\sum_{j \in \mathcal{D}} S_{ij} x_{kj}$ of intermediate data transferred from edge datacenter i to edge datacenter k, the WAN bandwidth usage cost is given by $P_i \sum_{j \in \mathcal{D}} S_{ij} x_{kj}$. Considering all the inter-datacenter WAN transfers, the total cost of WAN bandwidth usage can be computed by:

$$\sum_{i \in \mathcal{D}} \sum_{k \in \mathcal{D}, k \neq i} P_i \sum_{j \in \mathcal{D}} S_{ij} x_{kj}. \tag{5}$$

3.4 Problem Formulation

The objective of optimizing the task scheduling of collaborative cross-edge analytics is twofold: shortening the finish time of the shuffle phase and reducing the cost of WAN bandwidth usage. Here comes the question that whether the above two objectives are coincident. To answer this question, we first look at the performance formulation and cost formulation. Intuitively, to reduce the finish time of the shuffle phase, we should place more Reduce tasks at edge datacenters connected to higher bandwidth (i.e., B_{ik}) WAN links. On the other hand, to minimize the cost of WAN bandwidth usage, we should place more Reduce tasks at datacenters with lower bandwidth usage prices (i.e., P_i). Unfortunately, as we have empirically demonstrated in Section 2, the bandwidth usage price at the datacenter with faster WAN links is typically higher and thus the performance goal is consequently not aligned to the cost goal.

To address the challenge of contradictory performance and cost objectives, we propose to minimize the cost of WAN bandwidth usage while enforcinge the shuffle phase to be finished within a pre-defined deadline. Formally, such a performance-cost tradeoff problem can be formulated as the following integer programming (IP):

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{D}} \sum_{k \in \mathcal{D}, k \neq i} \sum_{j \in \mathcal{D}} P_i S_{ij} x_{kj} \\
\text{s.t.} \quad & \sum_{k \in \mathcal{D}} x_{kj} = 1, \forall j \in \mathcal{J} \\
& \sum_{j \in \mathcal{J}} x_{kj} \leqslant C_k, \forall k \in \mathcal{D} \\
& \frac{\sum_{j \in \mathcal{D}} S_{ij} x_{kj}}{B_{ik}} \leqslant W, \forall i,k \in \mathcal{D} \\
& x_{kj} \in \{0,1\}, \forall k \in \mathcal{D}, j \in \mathcal{J}.
\end{aligned} \tag{6}$$

Remark: A widely adopted alternative approach to navigating the performance-cost tradeoff is to transform the finish time into monetary cost, and then minimize the total cost. It is however nontrivial to precisely map the finish time of the shuffle phase to economic cost. In contrast, our model is more amenable to practical implementation, since based on the stringency of the job query it is ready to set a reasonable deadline W to ensure moderate performance.

Theorem 1: The cost-performance tradeoff problem is NP-hard.

Proof: We construct a polynomial-time reduction from the cost-performance tradeoff problem to the Generalized Assignment Problem (GAP), a classic combinatorial optimization problem which is proven NP-hard:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = 1, \forall j = 1,\dots,n$$

$$\sum_{j=1}^{n} w_{ij} x_{ij} \leqslant t_i, \forall i = 1,\dots,m$$

$$x_{ij} \in \{0,1\}, \forall i = 1,\dots,m, j = 1,\dots,n. \tag{7}$$

Given an instance $A = \left(m,n,c_{ij},w_{ij},t_i\right)$ of the GAP, we map it to an instance of the cost-performance tradeoff problem with $A' = \left(|\mathcal{D}| = m, |\mathcal{J}| = n, c_{ij} = \sum_{k \neq i, k \in \mathcal{D}} P_k S_{kj}, w_{ij} = 1, t_i = C_i, W = +\infty\right)$. Clearly, the above mapping can be done in polynomial time. Then, if there exists an algorithm that solves the cost-performance tradeoff problem A', it solves the corresponding GAP A as well. As a result, the GAP can be treated as a special case of the cost-performance tradeoff problem. Given the NP-hardness of GAP, the cost-performance tradeoff problem must be NP-hard as well.

Theorem 1 reveals that solving the cost-performance tradeoff problem is NP-hard and it is computationally infeasible for large input. Thus, we propose to develop a heuristic that seeks a good approximate solution to the cost-performance tradeoff problem.

3.5 Greedy-Based Heuristic for Task Scheduling

Before deriving the heuristic for task scheduling, we first rewrite the objective function of the cost-performance tradeoff problem as follows:

$$\sum_{i \in \mathcal{D}} \sum_{k \in \mathcal{D}, k \neq i} \sum_{j \in \mathcal{D}} P_i S_{ij} x_{kj} = \sum_{i \in \mathcal{D}} \sum_{j \in \mathcal{J}} P_i S_{ij} \sum_{k \in \mathcal{D}, k \neq i} x_{kj} = \sum_{i \in \mathcal{D}} \sum_{j \in \mathcal{J}} P_i S_{ij}\left(1 - x_{ij}\right). \tag{8}$$

The above equation indicates that, to minimize the cost of WAN bandwidth usage, we need to maximize the term $\sum_{i \in \mathcal{D}} \sum_{j \in \mathcal{J}} P_i S_{ij}\left(1 - x_{ij}\right)$. Intuitively then, task $j \in \mathcal{D}$ should be scheduled to edge datacenters with a larger $P_i S_{ij}$. With this insight, we first list the tasks in $\mathcal{D}$ in a non-increasing order of the value $\sum_{j \in \mathcal{J}} P_i S_{ij}$, and then the next task on the list is scheduled to the available datacenter with the largest $P_i S_{ij}$ without exceeding the computing capacity or compromising the performance goal. The detail is given in Algorithm 1.

Algorithm 1: Greedy-based heuristic for task scheduling

```
Input:
    Edge datacenter capacity, $C_k$;
    WAN bandwidth, $B_{ik}$;
    Price of WAN bandwidth, $P_k$;
    Deadline, $W$;
    Amount of intermediate data, $S_{kj}$;
Output:
    Task placement, $x_{kj}$;
1: list the tasks in $\mathcal{D}$ in a non-increasing order of the value $\sum_{k \in \mathcal{D}} P_k S_{kj}$;
2: Define $u_{kj} = \max_{i \in \mathcal{D}} \frac{S_{ij}}{B_{ik}}, \mathcal{M}_k = \varnothing, \forall k, j$;
3: for $j = 1,2,\dots,M$ do
4:     $k^* = argmin_{k \in \mathcal{D}}\left\{P_k S_{kj} | \sum_{l \in \mathcal{M}_i} x_{kl} \leqslant C_k - 1, \sum_{l \in \mathcal{M}_i} x_{kl} u_{kl} \leqslant W - u_{kj}\right\}$
5:     $x_{k^* j} = 1, \mathcal{M}_{k^*} = \mathcal{M}_{k^*} \cup \{j\}$
6: end for
7: return $x_{kj}$;
```

4 Implementation and Performance Evaluation

4.1 Implementation

The implementation of PPGA is on top of the Apache Spark framework. We override Spark's default scheduler and build our existing solution together with the default scheduler. When a taskset is submitted, we choose the scheduling method according to the dependency type of the taskset. When the taskset has shuffle dependency, we try to use PPGA to optimize the task placement. If PPGA is not suitable for the taskset, we choose default scheduler to finish task scheduling.

4.2 Experimental Setup

1) Experimental platform: Our experiment cluster is deployed on 5 datacenters with 10 instances. The 5 datacenters we choose are Singapore, Mumbai, Sao Paulo, Oregon, and Sydney. All instances used in our experiment are M4.4 XLARGE, which contains 16 vCPUs and 64 GB memory. We choose the instance at the Singapore region as Spark's master

and Hadoop Distributed File System's (HDFS) name node. The pair-wise bandwidth between the different EC2 regions is shown in Table 2. The price of outgoing WAN bandwidth usage across geo-distributed datacenters is shown in Table 3.

2) Software Settings: Our instances are running on centOS 7. The Spark version number of our implementation system is 2.1. We use HDFS from Apache Hadoop 2.7.1 as our distributed file system and start all instances as data nodes and worker nodes. The HDFS's block size that we use is 128 MB. The replication of HDFS is 3. Spark runs in standalone mode and does not require additional resource manager intervention.

3) Workload Specifications: To evaluate the effectiveness of our system to various kinds of computation tasks, we measure the performance metrics of three applications: WordCount, PageRank[11], and TeraSort[12]. Computation tasks can be divided into two categories (i. e., computation intensive and I/O intensive). In our benchmarks, WordCount is computation intensive while PageRank and TeraSort are I/O intensive.

- WordCount: WordCount aims to calculate the number of every single word in passages. WordCount first produces map tasks to calculate the frequency of words in every partition. Then Reduce tasks will collect the results of map tasks to get the final result. This application represents a typical data processing job. We use 11 GB data from Wikipedia as the input file.
- PageRank: PageRank computes the weights of the website using the amount and quality of links. It is a fundamental data processing application. It is used to calculate PageRank for a website. Our experiment uses a dataset that has 1 632 803 nodes and 30 622 564 edges.
- TeraSort: TeraSort is also a benchmark that measures computing ability of the big data framework. It is used to sort the sequences of results in the distributed situation. We generate 1 GB raw data to TeraSort for measurement.

4) Baseline: We compare our scheduler with the situation setting the price factor of each node to 1. According to the objective function of our problem, setting the price to 1 actually minimizes the amount of bandwidth usage. In the case of deadline insensitivity, we will contrast the different performances between minimizing the cost of WAN Bandwidth usage and minimizing WAN bandwidth usage.

4.3 Evaluation Results

1) Cost of WAN bandwidth usage: The primary performance metric is the cost of WAN Bandwidth usage. As we can see in Fig. 3, PPGA reduces the cost of WAN Bandwidth usage of WordCount, PageRank, and TeraSort by 30.26%, 25.82% and 31.60%, respectively. There are two reasons behind the cost reduction. Firstly, PPGA mainly considers minimizing the cost of WAN Bandwidth usage of an application rather than the volume of the WAN bandwidth usage. When the price is set to 1, the target of PPGA transforms to minimize the bandwidth usage. The second reason is that PPGA entirely considers all reduce tasks in a taskset. When a taskset is submitted to the schedule, PPGA makes a scheduling plan by analyzing overall tasks' distribution, data transferred price and bandwidth speed factors. So, we can get a smaller cost to complete the whole taskset. Further, the cost of WAN bandwidth usage of the whole application will be reduced.

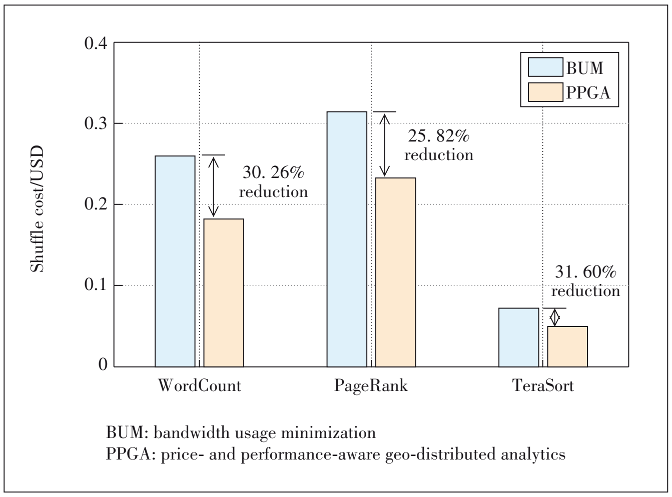
▼Table 2. Pair-wise bandwidth (in Mbit/s) between 5 different EC2 regions

City	Singapore	Oregon	Sydney	Sao Paulo	Mumbai
Singapore	46 028.8	116	140	63.5	390
Oregon	123	46 284.8	137	110	100
Sydney	144	124	46 592	69.0	106
Sao Paulo	66.6	109	74.5	46 899.2	79.3
Mumbai	390	103	106	73.7	46 694.4

▼Table 3. Price (in US dollar per GB) of outgoing WAN bandwidth usage

City	Singapore	Oregon	Sydney	Sao Paulo	Mumbai
Price	0.09	0.02	0.14	0.16	0.086

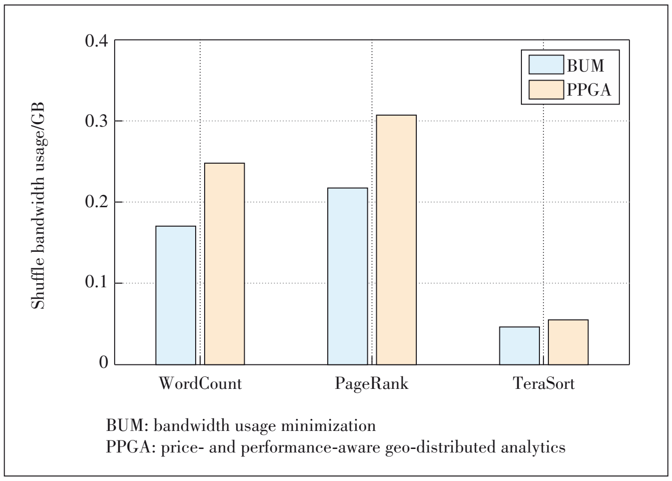
WAN: wide-area network



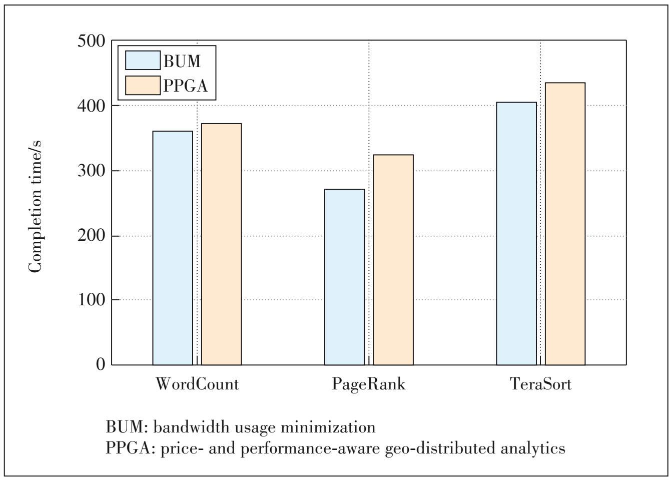
▲Figure 3. Data transfer costs of WordCount, PageRank and TeraSort

2) Volume of data transferred across datacenters: Fig. 4 shows the results of bandwidth usage. It is clear that the bandwidth usage minimization task scheduling incurs less bandwidth usage. This corresponds to our model in which we donot consider the price. At the same time, the results suggest that PPGA has a larger bandwidth usage, but it has a smaller data transfer cost, which also confirms our assumptions in Section 2.2.

3) Application completion times: Fig. 5 demonstrates that all applications' completion time with different prices is increased relative to the price situation (set to 1). This can be seen as a tradeoff between bandwidth usage and the cost of bandwidth usage. The price set to 1 just minimizes the bandwidth usage. When PPGA considers the price and the



▲ Figure 4. WordCount, PageRank and TeraSort's shuffle bandwidth usage across datacenters



▲ Figure 5. Application completion times of WordCount, PageRank and TeraSort

bandwidth, a task may be scheduled to a datacenter with a smaller cost, which needs more data transfer time. That is why applications running with PPGA have longer completion time.

5 Related Work

Collaborative cross-edge analytics, also known as geo-distributed analytics, has received great attention recently, due to the unprecedented increase in data volume. Most of the existing work focuses on optimizing a single objective of either query response time or WAN bandwidth usage. For reducing the usage of expensive WAN bandwidth, Pixida[7] works on dividing the DAG graph of a job into several parts to be processed in a datacenter. JetStream[13], a stream processing for data structured as online analytical processing (OLAP) cubes, relies entirely on aggregation and approximation to reduce bandwidth. However, JetStream does not optimize data and task placement. Geode[8] jointly catches intermediate results and optimizes task placement for Structured Query Language (SQL) queries. It optimizes WAN bandwidth usage but may lead to a poor performance. Flutter[10] carefully orchestrates task placement by exploiting bandwidth heterogeneity of the WAN. Note that none of the above work considers the cost of WAN bandwidth usage. On the other hand, Iridium[6] and the most recent work[5] jointly optimize the task and input data placement to reduce both response time and WAN traffic. Our work is inherently different from them in at least three important aspects. First, we leverage the heterogeneities of both the usage price and the bandwidth of the WAN, while Iridium and the work in Ref. [5] only consider the bandwidth diversity of the WAN. Second, rather than assuming that the network bottleneck exists in the up/down links of edge sites, we assume the cross-edge links as the bottleneck. Third, Iridium places tasks via solving the NP-hard problem with solvers like Gurobi, while we apply a simple heuristic which has better computational efficiency and scalability.

6 Conclusions

In this paper, we study the task scheduling problem for collaborative cross-edge analytics to jointly optimize cost and performance. We first demonstrate that, the commonly adopted approach of WAN bandwidth usage does not necessarily minimize the cost of WAN bandwidth usage, due to the price heterogeneity of WAN bandwidth usage. To fully explicit the price heterogeneity, we propose PPGA, a price and performance-aware task scheduler for collaborative cross-edge analytics. Unfortunately, the problem of WAN cost minimization under performance constraint is shown to be NP-hard, and thus computationally intractable for large inputs. To address this challenge, we propose an efficient greedy-based heuristic to improve the cost-efficiency of collaborative cross-edge analytics. The implementation of PPGA is based on Apache Spark, and extensive experiments across 5 Amazon EC2 regions demonstrate the cost-efficiency of PPGA.

Reference

[1] ANANTHANARAYANAN G, BAHL P, BODÍK P, et al. Real-time video analytics: the killer App for edge computing [J]. Computer, 2017, 50(10): 58–67. DOI: 10.1109/MC.2017.3641638

[2] JAIN S, ZHANG X, ZHOU Y H, et al. Spatula: Efficient cross-camera video analytics on large camera networks [C]//2020 IEEE/ACM Symposium on Edge Computing (SEC). San Jose, USA: IEEE, 2020: 110–124. DOI: 10.1109/SEC50012.2020.00016

[3] JIANG J C, ANANTHANARAYANAN G, BODIK P, et al. Chameleon: scalable adaptation of video analytics [C]//2018 Conference of the ACM Special Interest Group on Data Communication. Budapest, Hungary: ACM, 2018: 253 – 266. DOI: 10.1145/3230543.3230574

[4] ZHOU Z, CHEN X, LI E, et al. Edge intelligence: paving the last mile of artificial intelligence with edge computing [J]. Proceedings of the IEEE, 2019, 107(8): 1738 – 1762. DOI: 10.1109/JPROC.2019.2918951

[5] JIN H, JIA L, ZHOU Z. Boosting edge intelligence with collaborative cross-edge analytics [J]. IEEE Internet of Things journal, 2021, 8(4): 2444 – 2458. DOI: 10.1109/JIOT.2020.3034891

[6] PU Q F, ANANTHANARAYANAN G, BODIK P, et al. Low latency geo-distributed data analytics [C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. London, United Kingdom: ACM, 2015: 421 – 434. DOI: 10.1145/2785956.2787505

[7] KLOUDAS K, MAMEDE M, PREGUIÇA N, et al. Pixida [J]. Proceedings of the VLDB endowment, 2015, 9(2): 72 – 83. DOI: 10.14778/2850578.2850582

[8] VULIMIRI A, CURINO C, GODFREY PB, et al. Global analytics in the face of bandwidth and regulatory constraints [C]//The 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI). Oakland, USA: USENIX, 2015:323 – 336. DOI: 10.1109/TST.2016.7442496

[9] MAO H Z, SCHWARZKOPF M, VENKATAKRISHNAN S B, et al. Learning scheduling algorithms for data processing clusters [C]//The ACM Special Interest Group on Data Communication. Beijing, China: ACM, 2019: 270 – 288. DOI: 10.1145/3341302.3342080

[10] HU Z M, LI B C, LUO J. Flutter: Scheduling tasks closer to data across geo-distributed datacenters [C]//The 35th Annual IEEE International Conference on Computer Communications. San Francisco, USA: IEEE, 2016: 1 – 9. DOI: 10.1109/INFOCOM.2016.7524469

[11] PAGE L, BRIN S, MOTWANI R, et al. The PageRank citation ranking: bringing order to the Web. Technical report [R]. 1919

[12] O'MALLEY O. Terabyte sort on apache Hadoop [EB/OL]. [2021-01-04]. http://sortbenchmark.org/YahooHadoop.pdf

[13] RABKIN A, ARYE M, SEN S, et al. Aggregation and degradation in JetStream: streaming analytics in the wide area [C]//Usenix Conference on Networked Systems Design & Implementation. Seattle, USA: USENIX Association, 2014

Biographies

ZHAO Kongyang received the B.E. degree from the South China University of Technology, China in 2020. He is currently pursuing his master's degree in Sun Yat-sen University, China. His research interests include edge computing, edge intelligence, and serverless computing.

GAO Bin is now a research assistant of School of Computing in National University of Singapore (NUS). Before this, he received the master's degree and bachelor's degree from Huazhong University of Science and Technology (HUST), China in 2017 and 2020, respectively. His research interests include operation system, mobile edge computing, cloud computing, and geo-distributed data analytics.

ZHOU Zhi (zhouzhi9@mail.sysu.edu.cn) received the B.S., M.E., and Ph.D. degrees in 2012, 2014, and 2017, respectively, all from the School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), China. He is currently an associate professor in the School of Computer Science and Engineering at Sun Yat-sen University, China. In 2016, he was a visiting scholar at University of Goettingen, Germany. He was nominated for the 2019 CCF Outstanding Doctoral Dissertation Award, the sole recipient of the 2018 ACM Wuhan & Hubei Computer Society Doctoral Dissertation Award, and a recipient of the Best Paper Award of IEEE UIC 2018. His research interests include edge computing, cloud computing, and distributed systems.