Postprint: Research on Cross-Domain Big Data Job Scheduling Techniques in Cloud-Edge Systems

Authors: Xu Chao, WU Bo, Jiang Lili, Jin Yibo, Zhang Sheng

Date: 2019-01-03T00:00:00+00:00

Abstract

To improve user service quality, various edge clusters are deployed in proximity to users. While serving as an important supplement to cloud data centers, these clusters generate massive user data through continuous interactions. To reduce job completion latency incurred by processing such cross-domain big data, we first propose an online randomized scheduling algorithm, ranTA, which optimizes for minimizing the average completion time of a sequence of cross-domain jobs. ranTA computes preferences online for scheduling each computational task to different locations based on the heterogeneity of cross-domain resources, and probabilistically schedules each computational task according to these preferences. Furthermore, to prevent performance bottlenecks caused by accumulating "hot" data at edge clusters, we propose a piggybacked data redistribution mechanism, ranTA-data, built upon ranTA, which retains portions of data at the cloud data center during task execution. ranTA-data not only optimizes the completion time of current jobs, but can also be proven to concentrate the average completion time of job sequences near the optimal solution with high probability. Large-scale simulation experiments demonstrate that the proposed online randomized algorithm and data redeployment mechanism reduce the average completion time of job sequences by nearly 30% compared to traditional methods.

Full Text

Preamble

Task Scheduling for Geo-Distributed Data Analytics in Cloud-Edge System

Xu Chao[1], Wu Bo[1], Jiang Lili[2], Jin Yibo[3], Zhang Sheng[3]
(1. Research Institute of State Grid Jiangsu Electric Power Co., Ltd., Nanjing

210008, China;
2. Jiangsu Frontier Electric Technology Co., Ltd., Nanjing 211102, China;
3. Department of Computer Science & Technology, Nanjing University, Nanjing 210023, China)

Abstract: To improve quality of service, various edge clusters have been deployed near end users. While serving as an important supplement to cloud data centers, these edge clusters continuously generate massive user data through interactions. To reduce the job completion latency caused by processing such geo-distributed big data, this paper first proposes an online randomized scheduling algorithm called ranTA that minimizes the average completion time of a series of geo-distributed jobs. ranTA online computes preferences for scheduling each computing task to different locations based on the heterogeneity of cross-domain resources, and uses these preferences as probabilities to schedule tasks. Furthermore, to avoid performance bottlenecks caused by accumulating "hot" data at edge clusters, we propose a piggyback data redistribution mechanism called ranTA-data based on ranTA, which retains some data at the cloud data center during task execution. ranTA-data not only optimizes the completion time of current jobs but can also be proven to concentrate the average completion time of job series near the optimal solution with high probability. Large-scale simulation experiments demonstrate that the proposed online randomized algorithm and data redistribution mechanism reduce the average job completion time by nearly 30% compared with traditional methods.

Keywords: geo-distributed data analytics; cloud-edge system; task scheduling

0 Introduction

Major enterprises and organizations such as Google and Alibaba have deployed multiple data centers and numerous geo-distributed edge clusters worldwide [1]. Leveraging the powerful processing capabilities of data centers and the low-latency advantages of edge clusters, such cloud-edge systems provide high-quality services to users while accumulating substantial user data at various edges [2]. Many business decisions and data analytics tasks require real-time comprehensive processing of these geo-distributed datasets [3], making low-latency geo-distributed big data processing in cloud-edge systems a critical research problem.

Due to limitations in wide-area network data transmission, the approach of first aggregating massive edge data to cloud data centers for processing not only consumes bandwidth but also introduces significant latency. Numerous studies have considered executing tasks locally as much as possible to reduce cross-domain data transmission. Vulimiri et al. [4] investigated how to perform minimal data transfers and fast task execution in geo-distributed environments. Pu et al. [5] discovered that utilizing scarce bandwidth for large-scale data transfers leads to variable cross-domain transmission times, and thus minimized cross-domain data transfer volume through appropriate task scheduling. Refer-

ences [6,7] also considered data transfer and bandwidth usage when selecting execution modes for big data processing jobs to identify optimal data transfer strategies. However, due to the computational heterogeneity of edge clusters, task scheduling that purely optimizes data transfer can lead to load imbalance, causing task accumulation at "hot" edge clusters. To address this, references [8,9] proposed batch task scheduling that utilizes idle resources and bandwidth in cross-domain environments to reduce the overall completion time of batch tasks. However, directly scheduling tasks to remote cloud data centers when local computing resources are occupied places an enormous burden on cross-domain link bandwidth. Since some tasks can obtain idle computing resources through appropriate local queuing, Jin et al. [10] designed a batch task scheduling scheme supporting local queuing to further reduce the overall completion time of batch tasks.

Nevertheless, all these studies focus only on the currently submitted job and reduce its completion time through task scheduling. In fact, in such heterogeneous distributed cloud-edge systems, data distribution is key to job execution. If "hot" data can be transferred to powerful data centers as much as possible, subsequent related jobs can be completed efficiently. Existing work has optimized the completion time of current tasks but has not considered the average completion time across multiple jobs, i.e., they have not systematically studied the benefits that current task scheduling-induced data redistribution brings to subsequent jobs. Therefore, this paper aims to optimize the average completion time of job series and deeply investigates the task assignment problem for geo-distributed big data jobs. We propose an online randomized task assignment algorithm called ranTA and a piggyback data redistribution strategy called ranTA-data. These approaches not only optimize the completion time of current jobs but also prove that the average completion time of job series concentrates near the optimal solution with high probability. Extensive simulation experiments also demonstrate that the online randomized task scheduling algorithm and piggyback data redistribution strategy exhibit excellent performance, reducing the average job completion time by nearly 30% compared with traditional methods.

1 Geo-Distributed Big Data Processing in Cloud-Edge Systems

1.1 Cloud-Edge System and Big Data Processing

A cloud-edge system comprises a series of geo-distributed edge clusters deployed across various locations and a powerful cloud data center. The computing capacity of edge clusters and cloud data centers is generally characterized by the number of computing units (slots) [2]. The cloud data center is typically assumed to have abundant computing units, while edge clusters have relatively weaker processing capabilities with limited computing units that may become heavily loaded. All edge clusters have network connections to the cloud data

center, enabling data transmission and information exchange.

Generally, a data analytics job can be defined as a Directed Acyclic Graph (DAG) [5], where nodes represent stage functions and edges represent dependencies between stages. During execution, mainstream big data processing platforms such as Hadoop and Spark generate a set of parallel executable tasks for each stage of a DAG job and execute these tasks in batches according to the current job stage. Since all tasks in a stage of a DAG job must complete before proceeding to the next stage, the job completion time depends on the last completed task. For a computing unit s, let ϕ_s denote its pending workload, i.e., the time required to process all waiting tasks on computing unit s. Each data analytics task processes a data block of size d stored in HDFS [11] as input data [12], so data must be transferred to the computing unit running the task. The limited number of computing units in edge clusters may cause task queuing and extend task execution time; data centers have abundant computing resources for fast task execution but often face time consumption when extracting data from edges. The objective of job scheduling in cloud-edge systems is to decide, before a big data processing platform begins executing a batch of tasks in a stage, whether these tasks should execute at the edge cluster where the relevant data resides or be transferred to the data center, thereby minimizing job completion time.

Since some data analytics tasks remain in local clusters while others are processed at remote data centers, we need to evaluate the load generated by batch data analytics tasks locally and at the data center after task scheduling to optimize job completion time. For a data analytics job j and its batch of tasks, let D_{ji} denote the dataset accessed by these tasks on edge cluster i. The optimization objective for a single job is to minimize the completion time of the batch tasks. The load generated by batch tasks includes two aspects: (a) computational load produced in local edge clusters, and (b) load transferred from edge cluster i to the cloud data center due to task scheduling. Let U_{js} denote the total load on computing unit s after scheduling job j, and let V_{ji} denote the load transferred from edge cluster i to the cloud data center after scheduling job j. Let B_{ji} denote the available bandwidth from edge cluster i to the cloud data center for job j. For any computing unit s, these two types of load can be expressed as:

$$U_{js} = \sum_{d \in D_{ji}} I_{jds} \cdot \phi_s \cdot e_d \cdot \gamma$$

where I_{jds} is a scheduling indicator variable indicating whether the data analytics task in job j that uses data block d as input is scheduled to computing unit s. The term $\phi_s \cdot e_d \cdot \gamma$ in equation (1) represents the processing latency for data analytics tasks using data block d as input. Since edge clusters have relatively weaker processing capabilities compared to cloud data centers, the processing latency at edge cluster i in equation (2) is $\phi_s \cdot e_d \cdot \gamma$, where ϕ_s indicates that

computing unit s is located in edge cluster i, and γ is the processing speed ratio of edge cluster i relative to the cloud data center.

For the number of data analytics tasks, they share bandwidth B_{ji}, so the total transmission latency is the single task transmission latency multiplied by the number of tasks. Finally, among these data analytics tasks transferred to the data center, the last one to complete is the task with the longest execution time. Therefore, the longest execution time among these tasks is added to the transmission load.

1.2 Cross-Domain Big Data Processing Job Scheduling Problem

For a data analytics job j, the objective is to minimize the job's completion time, which depends on the last completed task. Since related tasks are distributed across various edge clusters or the cloud data center, the task scheduling objective for a single job can be transformed into minimizing the maximum load across clusters where tasks reside:

$$\min_{\phi} \max\{U_{js}, V_{ji}\}$$

For a series of jobs, optimizing the completion time of each individual job locally does not necessarily minimize the average completion time across jobs. Therefore, this paper defines the cross-domain big data processing task assignment problem for optimizing the average completion time of job series.

Definition 1 (Cross-Domain Big Data Analytics Task Assignment Problem, Geo-TA). For a batch of pending tasks from a DAG-based big data processing job, assign these tasks to relevant data clusters or transfer them to the data center to minimize the average completion time of all jobs. The completion time of each job is defined by equation (3).

Theorem 1 The cross-domain big data processing task assignment problem Geo-TA is NP-hard.

Proof. The known multiprocessor scheduling decision problem is NP-hard [13], defined as: given n processors and m jobs with processing times p_i, determine whether there exists a schedule ψ such that the completion time is less than or equal to a given parameter k, i.e., $\max_{s\in[1,n]} \sum_{i:\psi(i)=s} p_i \leq k$. For any instance of the multiprocessor scheduling decision problem, we can reduce it in polynomial time to an instance of the Geo-TA decision problem, and the two decision problems produce consistent outputs under any scheduling strategy. The Geo-TA decision problem is defined as: given parameter k, whether the overall completion latency of Geo-TA, i.e., equation (4), is less than or equal to k.

First, in the Geo-TA decision problem for cross-domain resource deployment, construct an edge cluster with n computing units and available bandwidth B

as an infinitesimal value. Construct a data analytics job with m data analytics tasks, where the execution time of each task using data block i as input corresponds to p_i. This reduction from any multiprocessor scheduling decision problem instance to a Geo-TA decision problem instance can be completed in $O(n+m)$ time, making the parameter k in the multiprocessor scheduling decision problem the same as parameter k in the Geo-TA decision problem. This reduction process is polynomial.

Next, for any multiprocessor scheduling strategy that schedules job i to processor or machine s, in Geo-TA we schedule the i-th data analytics task to computing unit s (since bandwidth is too low, tasks will not be scheduled to the cloud data center). In this way, the overall completion time in the multiprocessor scheduling decision problem is $\max_{s\in[1,n]} \sum_{i:\psi(i)=s} p_i$, which is consistent with the overall completion time in Geo-TA. Therefore, under the same parameter k, the two decision problems produce the same output.

Thus, since the Geo-TA decision problem is NP-complete, the Geo-TA problem is NP-hard. Otherwise, setting k to any value smaller than its optimal solution would allow the Geo-TA decision problem to be decided in polynomial time, contradicting the fact that the decision problem is NP-complete.

2 Online Randomized Task Scheduling Algorithm and Piggyback Data Redistribution Strategy

Since the above task scheduling problem is NP-hard and jobs arrive continuously, this paper proposes an online randomized resource deployment algorithm that attempts to minimize the completion time of each job. Based on this, we design a piggyback data redistribution strategy and theoretically prove that under this task scheduling and data redistribution mechanism, the average completion time of job series can concentrate near the optimal solution with high probability.

2.1 Online Randomized Task Scheduling Algorithm

The task scheduling problem of minimizing the completion time of the current job, i.e., optimizing equation (3), is a special case of the Geo-TA problem and is also NP-hard. However, this problem can be relaxed into a linear programming problem called lpGeo-TA (linear programming Geo-TA). Although the solution obtained by linear programming cannot be directly applied to the original cross-domain resource deployment problem, it reflects the algorithm' s preferences for current resource deployment. Therefore, this paper utilizes the theoretical optimal solution from this linear programming and uses it as a probability to schedule data analytics tasks. For a series of data analytics jobs, we can also prove that the results concentrate near the optimal solution with high probability.

Algorithm 1. Online Randomized Task Scheduling Algorithm ranTA

1. Solve lpGeo-TA $\rightarrow \{p_{jds}\}$
2. for each task do
3. Probabilistically round I_{jds} to $\{0, 1\}$
4. Schedule the task to computing unit s
5. end for

Algorithm 1 first relaxes the original cross-domain resource scheduling problem into a linear programming problem lpGeo-TA (line 1). Based on equation (5), with a single job j as the optimization objective, the variable I_{jds} is relaxed to a real number in $[0, 1]$:

$$\begin{aligned} \min_{\phi} \quad & \max\{U_{js}, V_{ji}\} \\ \text{s.t.} \quad & \sum_{s} p_{jds} = 1; \quad \forall j, d \\ & p_{jds} \in [0, 1] \end{aligned}$$

Since the variables are real numbers in $[0, 1]$, this problem can be efficiently solved using linear programming. For each task, the obtained $\{p_{jds}\}$ is rounded probabilistically (lines 2-5). Specifically, for each data analytics task of job j that uses data block d as input, a random number $r \in (0, 1]$ is selected. If r falls in the interval $(\sum_{k=1}^{s-1} p_{jdk}, \sum_{k=1}^{s} p_{jdk}]$, then I_{jds} is 1; otherwise, it is 0. This randomized rounding strategy ensures that for any data analytics task, exactly one computing unit can serve the task. Moreover, since the probability that I_{jds} equals 1 is exactly the probability that r falls in the interval $(\sum_{k=1}^{s-1} p_{jdk}, \sum_{k=1}^{s} p_{jdk}]$, the probability that I_{jds} equals 1 is precisely p_{jds}.

Finally, when scheduling real data analytics tasks, we can perform two pseudo-scheduling deployments in advance and select the scheme with lower load as the actual deployment strategy. This is because after making a choice between two options, the lower-load strategy can further reduce the probability of high load occurrence. The problem defined by lpGeo-TA in line 1 of the algorithm can be efficiently solved using linear programming techniques, while the remaining part of the algorithm (lines 2-5) has complexity of only $O(\xi)$, where ξ is the number of computing tasks contained in a job.

2.2 Piggyback Data Redistribution Strategy

After scheduling tasks to the cloud data center, we can immediately choose to retain the data there (piggyback data redistribution). This is because task execution itself requires data access. After transferring tasks to the data center, we can directly utilize the already transmitted data for retention. In this way,

as long as subsequent data analytics jobs need this data block again, they can execute directly at the data center without transferring data again from the edge computing cluster.

Meanwhile, piggyback data redistribution can also reduce the burden on edge computing clusters because subsequent data analytics tasks will be computed in the powerful cloud data center, preventing edge computing clusters from becoming hotspots and bottlenecks. This is particularly important in heterogeneous environments where some edge computing clusters have relatively weak capabilities, and retaining large numbers of data analytics tasks locally would create enormous load burdens on these clusters.

Algorithm 2. Piggyback Data Redistribution Strategy ranTA-data

1. ranTA
2. for each task do
3. if the task is scheduled to the cloud data center then
4. `Retain data at the data center`
5. end if
6. end for
7. for each edge cluster i do
8. Perform data redistribution
9. end for

This algorithm differs from the online task scheduling algorithm in that it uses the output of ranTA for data redistribution, i.e., it directly retains associated data at the cloud data center after task scheduling. Because this data redistribution strategy requires no additional cost (lines 3-7), it is called a piggyback data redistribution mechanism.

Furthermore, during piggyback data redistribution, we can optimize using the varying task completion times across different edge clusters (lines 8-10). The completion time Ω_j of job j may differ across edge clusters due to factors including edge cluster computing capacity, bandwidth, pending load, and data distribution. Therefore, we can leverage such load differences to allow data redistribution before the slowest edge cluster completes, i.e., perform data redistribution within the following constraint:

$$\Omega_j - \max_i\{V_{ji}\} \geq 0$$

Since job completion depends on the slowest task, which must exist in some slowest edge cluster, the job cannot finish before that edge cluster completes its tasks. Utilizing this time interval for data redistribution achieves better results.

2.3 Theoretical Analysis

This section first demonstrates that the data analytics job completion time obtained using only the ranTA algorithm can concentrate near its optimal solution with high probability (Theorem 2). Furthermore, using the data redistribution strategy ranTA-data can achieve a better theoretical bound (Theorem 3).

Theorem 2 The data analytics job completion time obtained by Algorithm 1 can concentrate near its optimal solution with high probability, i.e.:

$$\Pr\left[\max_s\{U_{js}\} \leq \max_s\{\mathbb{E}[U_{js}]\} + \Delta\right] \geq 1 - \delta$$

First, consider a single job j. After randomized scheduling, we obtain the real scheduling result U_{js}. Construct the random variable:

$$\Delta_{js} = U_{js} - \mathbb{E}[U_{js}]$$

where the left side of the inequality represents the actual scheduling load, the first term on the right is the maximum expected load across computing units (i.e., the theoretical optimum using $\{p_{jds}\}$ for scheduling), and the last term represents the distance between the actual scheduling and the optimal local scheduling. This distance increases exponentially as the probability $(1-\delta)$ decreases. Let $F(\delta)$ denote the maximum value of the rightmost term in equation (17) across all jobs. Then for all jobs j, the following inequality holds with probability at least $(1-\delta)$:

$$\max_s\{U_{js}\} \leq \max_s\{\mathbb{E}[U_{js}]\} + F(\delta)$$

Next, consider a series of jobs. For each job, the probability that equation (18) does not hold is at most δ. Using the Union Bound [15], for a series of n jobs, the probability that at least one violates the inequality is at most $n\delta$. Therefore, the following inequality holds for the job series with probability at least $(1-n\delta)$:

$$\frac{1}{n}\sum_{j=1}^{n}\max_s\{U_{js}\} \leq \frac{1}{n}\sum_{j=1}^{n}\max_s\{\mathbb{E}[U_{js}]\} + F(\delta)$$

If job j uses data blocks where the successor data block index of d is greater than d, then $\{\Delta_{js}\}$ forms a martingale difference sequence. Applying Azuma' s inequality [14] to this martingale difference sequence yields:

$$\Pr\left[|\Delta_{js}| \geq t\right] \leq 2\exp\left(-\frac{t^2}{2\sum_d \phi_s^2 e_d^2 \gamma^2}\right)$$

where the inequality indicates that after randomized scheduling, the load on computing unit s will not be far from its expectation. This distance increases exponentially as probability $(1-\delta)$ decreases. Since $\max_s\{\mathbb{E}[U_{js}]\}$ is the theoretical optimal solution obtained using $\{p_{jds}\}$, this implies that after randomized scheduling, all computing units concentrate near their optimal solutions with high probability, i.e., the following inequality holds with probability at least $(1-\delta)$:

$$\max_s\{U_{js}\} \leq \max_s\{\mathbb{E}[U_{js}]\} + \sqrt{2\ln(1/\delta)} \cdot \max_d\{\phi_s e_d \gamma\}$$

Similarly, for V_{ji}, we can obtain:

$$\max_i\{V_{ji}\} \leq \max_i\{\mathbb{E}[V_{ji}]\} + \sqrt{2\ln(1/\delta)} \cdot \max_d\{\tau_{ji} \cdot B_{ji}\}$$

Since the local theoretical optimal solution is always better than any integer scheduling result, $\max_s\{\mathbb{E}[U_{js}]\}$ and $\max_i\{\mathbb{E}[V_{ji}]\}$ are lower bounds for any integer scheduling result of job j. For any integer scheduling, its upper bound is to first transfer all data in edge clusters to the data center and then execute. Therefore, the following inequality holds with probability at least $(1-n\delta)$:

$$\frac{1}{n}\sum_{j=1}^{n} \max\{\max_s\{U_{js}\}, \max_i\{V_{ji}\}\} \leq \mathrm{Opt} + O\left(\frac{m}{n}\right) + F(\delta)$$

Finally, since the global optimal solution cannot be better than having all data already in the cloud data center from the beginning, and the total uploaded data volume is at most the total data access volume m, equation (20) can be transformed into the following inequality that holds with probability at least $(1-n\delta)$:

$$\frac{1}{n}\sum_{j=1}^{n} C_j(\phi) \leq \mathrm{Opt} + O\left(\frac{m}{n}\right) + F(\delta)$$

Thus, using only algorithm ranTA, the cross-domain resource deployment result concentrates near its optimal solution with high probability.

Theorem 2 provides the theoretical upper bound for the average completion time of job series using the ranTA algorithm. In traditional execution, even if data is transferred to the cloud data center, it is not retained after task completion. In our proposed piggyback data redistribution, data transferred to the cloud data center is directly retained after task execution, and more tasks will be scheduled to the cloud data center based on job characteristics (i.e., ranTA-data) to optimize benefits for subsequent jobs accessing the same data. Theorem 3 provides the upper bound for the average completion time of job series using ranTA-data.

Theorem 3 Using the piggyback data redistribution strategy ranTA-data, the data analytics job completion time can concentrate near its optimal solution with high probability, i.e. (where m' is the number of distinct data blocks among all data accesses):

$$\frac{1}{n}\sum_{j=1}^{n} C_j(\phi) \leq \mathrm{Opt} + O\left(\frac{m'}{n}\right) + F(\delta)$$

For each scheduling, assume that after scheduling job j, the optimal scheduling transfers z_{ji} tasks from edge cluster i to the cloud data center. Then equation (20) can be rewritten as:

$$\frac{1}{n}\sum_{j=1}^{n} \max\{\max_{s}\{U_{js}\}, \max_{i}\{V_{ji} - z_{ji}\}\} \leq \mathrm{Opt} + O\left(\frac{m'}{n}\right) + F(\delta)$$

Since $\max_i\{\mathbb{E}[V_{ji}]\}$ is the theoretical optimal solution obtained using $\{p_{jds}\}$, this implies that after randomized scheduling, $\max_i\{V_{ji}\}$ concentrates near its optimal solution with high probability across all edge clusters, i.e., the following inequality holds with probability at least $(1-\delta)$:

$$\max_{i}\{V_{ji}\} \leq \max_{i}\{\mathbb{E}[V_{ji}]\} + \sqrt{2\ln(1/\delta)} \cdot \max_{d}\{\tau_{ji} \cdot B_{ji}\}$$

Therefore, across all edge clusters, the following inequality holds with probability $(1-\delta)$:

$$\frac{1}{n}\sum_{j=1}^{n} \max\{\max_{s}\{U_{js}\}, \max_{i}\{V_{ji}\}\} \leq \mathrm{Opt} + O\left(\frac{m'}{n}\right) + F(\delta)$$

This is because if local computing load remains unchanged, transferring one more task to the cloud data center would make the overall latency of the optimal scheduling longer (since $\max_i\{\mathbb{E}[V_{ji}]\}$ is a lower bound for any integer scheduling result of job j). If piggyback data redistribution is performed based on optimal scheduling, for duplicate data, data is uploaded at most once, so $m' \leq m$. Thus, equation (23) becomes:

$$\frac{1}{n}\sum_{j=1}^{n} C_j(\phi) \leq \mathrm{Opt} + O\left(\frac{m'}{n}\right) + F(\delta)$$

The forms of Theorem 2 and Theorem 3 are similar, with the difference being m and m', where m is the number of all data blocks related to tasks, and m' is the number of distinct data blocks related to tasks. m contains repeated data accesses, so $m' < m$. Intuitively, thanks to piggyback data redistribution that retains data at the cloud data center for subsequent job series, the theoretical bound of the ranTA-data strategy is superior to that of ranTA.

3 Performance Evaluation

3.1 Evaluation Methodology and Design

The simulation experiment evaluates algorithms based on the same optimization objective: the average completion time of job series. This paper compares the performance of four algorithms: (a) Local Execution, which directly deploys tasks to the edge cluster where data resides; (b) Aggregation [16], which first aggregates all data to the cloud data center before execution; (c) ranTA, which schedules based on local optimal solution but does not retain data transferred from edges at the cloud data center after task execution; (d) ranTA-data, which performs online scheduling and retains data transferred from edges at the cloud data center.

The simulation experiment simulates common parameter settings in cloud-edge scenarios:

a) **Cloud-Edge Environment Setup**: 800 edge clusters and one central cloud data center; each edge cluster has 10-150 computing units; the processing speed ratio of edge clusters relative to the cloud data center ranges from 1 to 5 times, and the bandwidth of each edge cluster varies from 100Mbps to 1Gbps [5].

b) **Data Analytics Jobs**: 100 data analytics jobs, each containing 50-750 data analytics tasks [17]; each task processes a data block of size 64MB.

c) **Data Distribution**: 30,000 data blocks are deployed across different edge clusters following a Zipf distribution [9] with default parameter 0.85; to reflect data reuse characteristics [18], 30% of data is accessed with probability 0.8.

3.2 Experimental Results Analysis

[Figure 1: see original paper]~[Figure 3: see original paper] show how job completion time changes as the cloud-edge environment settings vary. ranTA-data performs the best, achieving average improvements of 33.9%, 31.6%, and 24.3% compared with traditional data locality and aggregation strategies when the

number of computing units, bandwidth, and number of edge clusters change, respectively. Compared with ranTA without data redistribution, it achieves average improvements of 14%, 14.3%, and 16% under the same respective changes.

[Figure 1: see original paper] shows the completion time varying with the number of computing units. [Figure 2: see original paper] shows the completion time varying with bandwidth. [Figure 3: see original paper] shows the completion time varying with the number of edge clusters.

[Figure 1: see original paper] and [Figure 2: see original paper] indicate that ranTA achieves greater improvements over the other two algorithms when edge cluster resources are relatively scarce (number of computing units or bandwidth). This is because ranTA combines the current system state when each job arrives, scheduling some tasks to the cloud data center to alleviate high load locally or on network bandwidth, avoiding long queuing waits for idle computing resources locally or sharing scarce network bandwidth with massive transfer tasks. However, the improvement is still limited compared with ranTA-data because ranTA-data retains all data scheduled to the cloud data center through piggyback data redistribution, enabling direct benefits for subsequent jobs that reuse the same data.

[Figure 3: see original paper] shows that when the number of edge clusters is small, data distribution becomes concentrated, causing large amounts of data to accumulate in a few edge clusters and increasing the burden on corresponding edge clusters. Consequently, both aggregation and data locality strategies maintain high completion times. Although ranTA' s completion time is also relatively high in this scenario, it has made efforts to 疏散 load. ranTA-data performs the best because, in addition to transferring load, the data retained at the cloud data center effectively improves subsequent jobs.

[Figure 4: see original paper] and [Figure 5: see original paper] show how job completion time changes as data analytics job settings vary. ranTA-data continues to excel, achieving average improvements of 31.6% and 34.1% compared with traditional data locality and aggregation strategies when the number of jobs and average number of tasks change, respectively. Compared with ranTA without data redistribution, it achieves average improvements of 18% and 16% under the same respective changes. As the number of jobs increases or the number of data analytics tasks per job grows, the load on edge clusters increases continuously, causing the completion times of aggregation and data locality strategies to increase significantly. However, ranTA-data increases slowly because it continuously performs load balancing of data analytics tasks at runtime, and data transfers also serve subsequent jobs.

[Figure 4: see original paper] shows the completion time varying with the number of jobs. [Figure 5: see original paper] shows the completion time varying with the average number of tasks within a job.

[Figure 6: see original paper] shows the average job completion time varying with data distribution. ranTA-data achieves an average improvement of 27.2% com-

pared with traditional data locality and aggregation strategies, and at least 13% improvement compared with ranTA with only online data distribution. A larger Zipf parameter in the data distribution means more uneven data concentration in fewer edge clusters. This high load on some edge clusters causes extremely high latency for aggregation and data locality strategies. When data distribution is more uniform, ranTA-data may occasionally perform slightly worse than uploading all data due to the huge number of edge clusters. This is because both ranTA and ranTA-data are based on randomized scheduling strategies; when the number of data analytics tasks is small, randomized rounding may result in worse completion time than scheduling all tasks to the cloud data center. Finally, data locality performs worse than aggregation when data distribution is extremely uniform because edge clusters have a processing speed ratio compared to the cloud data center, making local execution of the same tasks more costly than uploading all to the cloud.

4 Conclusion

This paper addresses the bottleneck problem that occurs in geo-distributed environments when processing big data across heterogeneous edge clusters, where clusters with weak computing capabilities or scarce bandwidth connections become performance bottlenecks. We propose an online randomized task scheduling algorithm called ranTA and a piggyback data redistribution strategy called ranTA-data. The algorithm schedules each computing task probabilistically based on system preferences, and we prove that under this scheduling mechanism with piggyback data redistribution, the average completion time of job series concentrates near the optimal solution with high probability. This approach reduces the average completion time of job series and holds significant theoretical and practical importance.

References

[1] Calder M, Fan Xun, Hu Zi, et al. Mapping the expansion of Google' s serving infrastructure [C]// Proc of Conference on Internet Measurement. New York: ACM Press, 2013: 313-326.

[2] Jalaparti V, Bodik P, Menache I, et al. Network-aware scheduling for data-parallel jobs: plan when you can[C]// Proc of ACM Conference on Special Interest Group on Data Communication. New York: ACM Press, 2015: 407-420.

[3] 卢慧, 高弘博, 张丰满, 等. Hadoop 云平台下基于资源预估的作业调度算法 [J]. 计算机应用研究,2016,33(8): 2311-2314. (Lu Hui, Gao Hongbo, Zhang Fengman,et al. Job scheduling algorithm based on data-aware in Hadoop[J]. Application Research of Computers, 2016, 33(8):2311-2314.)

[4] Vulimiri A, Curino C, Godfrey P B, et al. Global analytics in the face of bandwidth and regulatory constraints[C]//Proc of the 12th USENIX Sympo-

sium on Networked System Design and Implementation. Berkeley, CA: Usenix Accociation, 2015: 323-336.

[5] Pu Qifan, Ananthanrayanan G, Bodik P, et al. Low latency Geo-distributed data analytics [C]// Proc of ACM Conference on Special Interest Group on Data Communication. New York: ACM Press, 2015: 421-434.

[6] Viswanathan R, Ananthanarayanan G, Akella A. CLARINET: WAN-aware optimization for analytics queries[C]// Proc of the 12th USENIX Symposium on Operating Systems Design Implementation. Berkeley, CA: Usenix Accociation, 2016: 435-450.

[7] Yu Boyang, Pan Jianping. Location-aware associated data placement for geo-distributed data-intensive applications[C]//Proc INFOCOM. Piscataway, NJ: IEEE Press, 2015: 603-611.

[8] Hu Zhiming, Li Baochun, Luo Jun. Flutter: scheduling tasks closer to data across geo-distributed datacenters [C]//Proc of IEEE INFOCOM. Piscataway, NJ: IEEE Press, 2016: 1-9.

[9] Hung C C, Golubchik L, Yu Minlan. Scheduling jobs across geo-distributed datacenters [C]//Proc of the 6th ACM Symposium on Cloud Computing. New York: ACM Press, 2015: 111-124.

[10] Jin Yibo, Qian Zhuzhong, Guo Song, et al. ran-GJS: orchestrating data analytics for heterogeneous geo-distributed edges [C]// Proc of the 47th International Conference on Parallel Processing. New York: ACM Press, 2018: 29: 1-29: 10.

[11] Ghemawat S, Gobioff H and Leung S. The Google file system [C]// Proc of the 19th ACM Symposium on Operating Systems Principles. New York: ACM Press, 2003: 29-43.

[12] 曹书豪, 张昌宏, 麻旻. 一种改进的 Hadoop 多用户作业调度方法 [J]. 计算机应用研究,2015,32(5):1395-1398. (Cao Shuhao, Zhang Changhong, Ma Min. Improved method in solving Hadoop multi-user scheduling[J]. Application Research of Computers,2015,32(5): 1395-1398.)

[13] Chen Jianer, Lee C Y. General multiprocessor task scheduling[J]. Naval Research Logistics, 1999, 46 (1): 57-74.

[14] Azuma K. Weighted sums of certain dependent random variables[J]. Tohoku Mathematical Journal, 1967, 19 (3): 357-367.

[15] Galamnos J, Simonelli I. Bonferroni-Type inequalities with applications, probability and its applications[M]. New York: Spring-Verlag, 1996.

[16] Rabkin A, Arye M, Sen S, et al. Aggregation and degradation in jetstream: streaming analytics in the wide area[C]//Proc of the 11th USENIX Symposium on Networked System Design and Implementation. Berkeley, CA: USENIX Accociation, 2014: 275-288.

[17] Ananthanarayanan G, Hung Chienchun, Ren Xiaoqi, et al. GRASS: trimming stragglers in approximation analytics[C]//Proc of the 11th USENIX Symposium on Networked System Design and Implementation. Berkeley, CA: USENIX Accociation, 2014: 289-302.

[18] Chen Yanpei, Archana G, Rean G, et al. The Case for evaluation MapReduce performance using workloads suites[C]// Proc of the 19th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication System. Washington DC: IEEE Computer Society, 2011: 390-399.

Note: Figure translations are in progress. See original paper for figures.