Microservice Based Video Cloud Platform with Performance-aware Service Path Selection

这两个单词之间用不用加短横线，文中有的加有的没有加

Haitao Zhang, Ning Yang, Zhengjun Xu, Bingchang Tang, Huadong Ma
Beijing Key Lab of Intelligent Telecomm. Software and Multimedia,
Beijing University of Posts and Telecomm., Beijing, China
Email: zht@bupt.edu.cn; xny@bupt.edu.cn; hzxuzj@bupt.edu.cn; zyjhtangtang@bupt.edu.cn; mhd@bupt.edu.cn

***Abstract*—Microservice based cloud architectures can provide distributed loosely coupled autonomous services and powerful virtual resources, which becomes a promising solution to deal with the challenges of large-scale intelligent video applications. However, the current service selection methods usually do not consider both the fine-grained online service capability and the features of video tasks, and this will result in the degradation of the overall efficiency of the service composition. In this paper, we firstly design a microservice-based video cloud computing platform which can take advantage of the loosely coupled feature of microservices to accomplish a video application. Secondly, we propose a novel Performance-Aware Service PAth Selection (PSPAS) approach which includes the initial path selection stage and the adaptive service path updating stage. For optimizing the microservice path, we establish a fine-grained time estimation model which synthetically consider the processing capability of microservice instances, the characteristics of video processing tasks, and the data transfer conditions between microservice instances. Based on the proposed performance model, we use the shortest path algorithm to construct an optimal microservice path at the initial service selection stage. And then we use an adaptive and efficient microservice path updating method during the task execution, which can narrow the service path search space prior to online performing the microservice path selection. Finally, our proposed approach is evaluated through the experiments executed in different conditions, and the evaluation results demonstrate the effectiveness of our method.**

***Keywords*-Microservice, service path selection, cloud computing, video processing**

I. Introduction

The cloud platform can provide a flexible stack of powerful virtual resources like CPU, memory, storage and network bandwidth to deal with the challenges of large-scale video processing [1] [2]. Therefore, the various video application systems, which employ the automatic video analysis technologies to provide intelligent functions, have been deployed on the cloud computing platform. On the other hand, the microservice based cloud architectures have been attracted much attention recently, in which an application is decomposed into different fine-grained lightweight cloud services to improve modularity, flexibility, agility and efficiency [3]. Consequently, the microservice based cloud computing platform will become a promising solution for the future supporting platform of large-scale intelligent video application systems.

这句话中的been去掉

The microservice based cloud platform can provide various types of microservices each of which can be called to implement a concrete data processing function such as data collection, data transformation, data characterization, and data classification. And usually many microservice instances with the same type are created to immediately respond to the online service requests of network applications. At run time, the microservice instances executed in sequence are selected from the pool of the candidate cloud microservice instances to constitute a microservice path. However, the different microservice instances have the different resource configurations and runtime loads, and thus provide the different Quality of Service (QoS). Therefore, the automatic service selection strategy in this environment plays a major role in determining the performance of the cloud computing tasks.

Currently, as the number of web services on Internet is increasing, a lot of service selection methods have emerged. Several recent literatures [4] [5] [6] focus on supporting services selection and composition based on QoS. But they just take into account the static QoS attributes such as responsiveness, availability, and throughput, and do not consider the runtime features of the selected service instance. After a subtask of the cloud application has been finished by one of composite services, the QoS of the following service instances will change dynamically over time. So a predefined optimal service composition may be inefficient during the execution of a large-scale cloud computing task.

In addition, some dynamic and adaptive algorithms are proposed to optimize the overall QoS of a service composition [7] [8], and the literatures [11] [12] propose performance-aware service selection and composition for cloud computing platform. These methods select appropriate component services for creating an optimal service composition, and also dynamically adapts to optimal service composition based on the various changes of the service provision platform. But they do not consider the fine-grained features of the video processing tasks, which significantly affect the total execution time of video tasks in addition to the performance of service resources.

In the paper, we focus on the efficient task execution of the microservice based video cloud computing platform. The objective is to improve the task processing efficiency of video applications by optimizing the microservice path selection. The main contributions of this paper are as follows.

Firstly, we present a microservice-based video cloud computing platform which can be integrated with the general video application system seamlessly. Our platform can take advantage of the loosely coupled feature of microservices to accomplish a video application, and supports automatic and adaptive microservice path selection for improving the efficiency of video processing tasks.

Secondly, we propose a novel Performance-aware Service PAth Selection (PSPAS) approach in the microservice-based video cloud computing platform. For optimizing the microservice path, we give a fine-grained performance quantification method of microservice instances, which includes data processing time model and data shuffling time model. We adopt the regression analysis prediction technology to estimate the processing time of the video subtask performed by a microservice instance, and give a common expression of the data shuffling time model. In our solution, we synthetically consider the processing capability of microservice instances, the characteristics of video processing tasks, and the data transfer conditions between microservice instances. Based on the established performance model, our PSPAS approach includes two execution stages: Initial path selection strategy which use the shortest path algorithm to construct an optimal microservice path at the beginning of the video task; Adaptive service path updating strategy which online improves the microservice path based on the path search space pruning principle at the runtime of the video task.

use 改为 uses

Finally, we conduct the extensive experiments to verify the benefits of the microservice-based video cloud platform and the proposed microservice path selection approach. Experimental results demonstrate that our PSPAS approach can achieve approximate performance of the optimal method and outperform other widely used methods.

The rest of the paper is organized as follows. Section II describes the architecture and workflow of the microservice based video cloud platform. We propose a service path selection approach in Section III. Section IV presents the experimental results. Section V outlines the related work. We conclude the paper in Section VI.

II. Microservice Based Video Cloud Platform

In this section, we first present the microservice based architecture of our video cloud platform, and then introduce the basic workflow of the video microservice path selection for video applications.

A. Platform Architecture

Figure 1 illustrates the architecture of the proposed video processing cloud platform. As shown in the figure, the

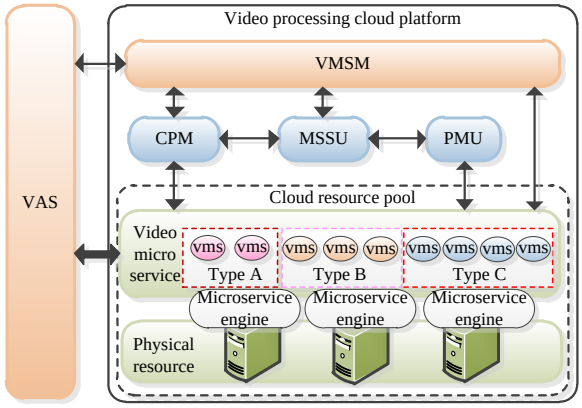

Figure 1. Architecture of microservice based video cloud platform

cloud platform is integrated with the Video Application System (VAS) which is a network system focusing on video application technology such as video surveillance. The cloud platform is built based on the microservice architecture (which is typically the container technology), can provide the intelligent video processing microservices such as image filtering, face detection, characterization of visual objects, and face classification, which can be combined to implement a whole video processing task of the VAS. The cloud platform includes several functionally independent management units and a cloud resource pool consisting of a server cluster.

In the video cloud platform, the main functions of the Video MicroService Manager (VMSM) include: receiving the requests of the video processing service from the VAS, managing the execution of the video processing tasks in the cloud platform, and forwarding the video processing results from the cloud platform to the VAS. The Cloud Platform Manager (CPM) is responsible for managing the cloud resources and video microservice instances, e.g., the allocation and release of the cloud resources, and the creation and removal of microservice instances. By using the following proposed performance-aware service path selection approach, the MicroService Selection Unit (MSSU) can make a suitable decision on the microservice selection for video processing tasks based on the current states of the microservice instances and the features of the video tasks. The Performance Monitoring Unit (PMU) is responsible for collecting and recording the performance information of the microservice instances, which can be used to select efficient video microservice instances by the MSSU. The cloud resource pool is built on a physical server cluster configured with the microservice engine, and can provide various types of Video MicroService (VMS) instances. The VMS instances with the same type can provide the same function of video processing, but have different and time-varying processing capabilities. Therefore, the key problem is to adaptively select the optimal VMS instances from the

cloud resource pool for the efficient processing of the video tasks.

B. Workflow of Microservice Path Selection

The basic workflow of the microservice path selection in the video processing cloud platform is as follows. First of all, we need to create various types of microservice instances in the video processing cloud platform according to the video application requirements, and this process is executed by call the related functional interfaces of the VMSM, which are actually implemented by the CPM. After the set of the microservice instances has be created, the VAS can send a request of the video processing task to the VMSM. The video processing task can be performed through the sequential execution of the subtasks processed by the specific microservice instances. When the VMSM receives the video processing request, it first analyzes the structure of the video processing task, including the type of each subtask, the relation among the subtasks, and input and output information of each subtask. Then, for optimizing the task processing efficiency, the VMSM needs to request the MSSU to provide an optimized microservice path consisting of the sequential microservice instances. The MSSU periodically inquires the PMU about the current states of the microservice instances, and can adaptively select the suitable microservice instances from the cloud resource pool for the online video task. Finally, based on the selection results of the MSSU, the VMSM invokes the available microservice instances to perform the video task, and forwards the video processing results to the VAS.

III. Performance-aware Service Path Selection

In this section, we first define the service path selection problem for the microservice based video cloud platform, and then give our PSPAS approach. Based on the proposed performance model of the video microservice, the PSPAS approach includes two execution stages which are respectively the initial path selection and the adaptive service path updating.

A. Problem Description

In the microservice based cloud platform, the core workflow of the video application can be described as a video processing pipeline which consists of multiple video processing subtasks such as video preprocessing, object segmentation, characterization of the shape and texture features of the segmented objects, and machine learning based object classification. Each video processing subtask can be implemented by a microservice instance selected from a set of microservice instances with the same functionality, and the whole processing pipeline of the video application can be implemented by the sequential composition of the selected microservice instances.

Assume that P is a video processing pipeline of video application, which can be divided into n sequential subtasks $P = \{P_i\}$ $(i = 1, ..., n)$. We also define a microservice class set $S = \{S_i\}$ $(i = 1, ..., n)$. Each microservice class S_i consists of all microservice instances $\{s_{ij}\}$ $(j = 1, ..., j_i$ where j_i is the number of the microservice instances in the microservice class $S_i)$ that deliver the same functionality (e.g. object segmentation), but potentially differ in terms of execution efficiency. Each video subtask P_i can be implemented by one microservice instance $\{s_{i\cdot}\}$ in the microservice class S_i. The execution time T_i of the subtask P_i is defined by

$$T_i = T_i^c + T_i^s \tag{1}$$

where T_i^c is the computation time of the data processed by the microservice instance $\{s_{i\cdot}\}$, and T_i^s is the transmission time of the data shuffled from the upstream microservice instance $\{s_{i-1\cdot}\}$ to the microservice instance $\{s_{i\cdot}\}$.

We define a microservice path SP as a sequential microservice instance set $\{s_{1\cdot}, ..., s_{n\cdot}\}$ where $s_{i\cdot}$ is selected from S_i $(i = 1, ..., n)$, and SP can determine a whole implementation of the video processing pipeline P. Therefore, the total execution time of P is computed as

$$T = \Sigma_{i=1}^{n} T_i. \tag{2}$$

For the video processing pipeline P, our objective is to select the optimal microservice path from all feasible microservice paths, which can minimize the execution time T of P.

B. Performance Modeling for Video Microservice

For optimizing the service path selection, we need to quantify the execution time T_{ij} for each microservice instance s_{ij} when s_{ij} is selected for implementing the subtask P_i. According to the equation (1), T_{ij} is the summation of the data processing time T_{ij}^c and the data shuffling time T_{ij}^s.

Data processing time model. The execution time T_{ij}^c for each microservice instance s_{ij} is influenced by various factors including the task descriptions and the context of the cloud platform. In this paper, we use the online regression method to learn the prediction model of the data processing time in microservice instance, which takes both model construction overhead and online prediction efficiency into account. In addition, our model not only considers the historical and current states of the microservice instance, but also incorporates the video task features that have the impact on the execution time. And this performance model with fine-grained context description can provide an efficient and accurate time prediction result [15].

Assume a subtask P_i running in the microservice instance s_{ij} is represented by a vector $X = (x_1, ..., x_m) \in R^m$ where m is the number of the features of the our prediction model. Specifically, the model features include the video task information which are the video resolution and the size

of the video file, and the descriptions of the microservice resources which are the number of CPU cores, the clock rate of CPU, the occupancy rate of CPU, the occupancy rate of the Memory, the bandwidth of the Memory. In our regression prediction model, each feature is transformed to the log_2-scale form to obtain a good linear fitting model [16]. The regression function is given by

$$f(W, X) = \sum_{l=0}^{m} w_l \log_2 x_l \quad (3)$$

这个公式左边下标，应该是i+1还是i呢？？

where $W = (w_0, w_1, ..., w_m)$ are the parameter vector (to be learned) of the model, and $x_0 = 0$. Obviously, the input of a subtask (except the first one) is the output of the previous subtask in the video pipeline and is not the original video file. However, the original video data naturally has a direct impact on the execution time of every subtask. So this linear regression function can be applied to any video subtask running in microservice instance.

We use squared difference as the loss function to measure the error between the predicted value and the actual value. In order to prevent the overfitting of the model, ℓ_2 norm regularizer is used as a punishment term. This choice is motivated by the availability of highly robust algorithms to fit online linear regression models [17], even in the presence of an adversary scaling of the features. The linear regression problem can be described as:

$$\arg\min_{W} \sum (f(W, X) - \log_2 t^{ac})^2 + \lambda \|W\|_2 \quad (4)$$

where t^{ac} is the actual execution time of subtask, and λ is the regularization parameter. Based on the training data set collected from the cloud platform, this regression model is learned by using the Normalized Adaptive Gradient (NAG) algorithm [17]. Once model parameter vector W has been learned, the log_2 value of the new execution time is predicted through Equation 3. Finally, we can estimate the the execution time T^c_{ij} of every each microservice instance s_{ij} according to the corresponding regression model.

改为equation (3)

Data shuffling time model. For giving a common expression for the data shuffling time model, we define a source subtask P_0 and a destination subtask P_{n+1} which are respectively the starting subtask and the ending subtask of all video processing pipelines. Now, we set the subtask set $P \leftarrow P \cup \{P_0, P_{n+1}\}$. Furthermore, we define a source microservice instance s_{00} and a destination microservice instance $s_{n+1,0}$ which are respectively the starting node and the ending node for all microservice paths. Assume that $s_{i-1,k}$ ($\in S_{i-1}$) is a previous microservice instance of s_{ij} in one microservice path ($i = 1, ..., n+1$). The data shuffling time from $s_{i-1,k}$ to s_{ij} is presented by

$$T^s_{ij}(k) = \frac{Vol_i}{Nr_{ij}(k)} \quad (5)$$

where Vol_i is the amount of the data shuffled from the subtask P_{i-1} to the subtask P_i, and $Nr_{ij}(k)$ is the data transfer rate between $s_{i-1,k}$ and s_{ij}.

Assume Vol_0 is the amount of the raw data to be processed by the video processing pipeline P. For the subtask P_i in P, we define α_i as the ratio between the amount of the output data and the amount of the input data ($i = 1, ..., n+1$, $\alpha_0 = 1$, and $\alpha_{n+1} = 0$). So we can obtain $Vol_i = \alpha_{i-1} \cdot Vol_{i-1}$. α_i, which is correlated with the subtask P_i, can be estimated through the offline experiment evaluation, and can be taken as the average of the ratio values of the experimental results. In the equation (3), the data transfer rate $Nr_{ij}(k)$ can be obtained by the network speed measurement tool.

C. Initial Service Path Selection

Based on the performance model proposed in the last subsection, we can construct a layered weighted directed graph G. In G, each layer represents a service class S_i, and contains multiple nodes each of which represents a microservice instance s_{ij} ($\in S_i$). There is an edge $e_{ij}(k)$ between each node s_{ij} in the layer S_i and each node $s_{i-1,k}$ in the layer S_{i-1}. The weight of the edge $e_{ij}(k)$ is the summation of the data processing time T^c_{ij} and the data shuffling time $T^s_{ij}(k)$. The microservice path weight is the summation of the weight values of all edges in the path. Therefore, our objective is transformed to select the optimal microservice path $\{s_{00}, s_{1\cdot}, ..., s_{n\cdot}, s_{n+1,0}\}$ which has the minimum path weight among all feasible microservice paths. And essentially this is a classic shortest path problem with a single source node, which can be solved by many algorithms such as Dijkstra algorithm, Bellman-Ford algorithm, and Viterbi algorithm. In this paper, we use the Viterbi algorithm to obtain the shortest path SP with the minimum weight as the optimal microservice path.

D. Adaptive Service Path Updating

Given the layered weighted directed graph G, the shortest path algorithm introduced above can be used to determine the optimal microservice path SP. However, after one subtask P_l is finished by $s_{l\cdot}$ ($l = 1, ..., n-1$), the resource operation states or the service capabilities of the subsequent microservice instances $s_{i\cdot}$ ($i = l+1, ..., n$) in SP has changed. It means that the weights of the edges in G change over time in the process of the online running of the video subtasks. Therefore, the pre-determined optimal microservice path SP is not optimal in the actual operation process. To overcome this problem, a straightforward method for finding the online optimal microservice path is to repeatedly call the above shortest path algorithm when each subtask of the pipeline has finished. But this solution can be very expensive in terms of computation time and inappropriate for run-time service selection in the video pipelines with many microservices.

In this subsection, we propose an adaptive service path updating method which can narrow the service path search space prior to online performing the microservice path selection. First, we define a time ratio $\beta = \frac{T_i^c}{T_i^s}$ for each subtask P_i. Then, we give the following two definitions.

Definition 1: Computation-Dominant Subtask (CDS) is a subtask in the video processing pipeline, and its time ratio $\beta \geq \beta_c$ where β_c is a given time ratio threshold of CDS.

Definition 2: Transmission-Dominant Subtask (TDS) is a subtask in the video processing pipeline, and its time ratio $\beta \leq \beta_s$ where β_s is a given time ratio threshold of TDS.

The execution time of the CDS is mainly influenced by the computation capability of the microservice instance, and consequently we can ignore the transmission time of the input data for improving the execution efficiency of the service path updating. For example, the feature computation subtask is a typical CDS. Similarly, we can ignore the computation time of the TDS, e.g., video preprocessing subtask. The time ratio thresholds β_c and β_s are usually the empirical values, and can be set based on the experimental results in the specific cloud platform. Therefore, once the subtask P_i is ready to run in the process of the video pipeline operation, we can prune the path search space for the service path updating according to the following principle ($i = 2, ..., n$).

Principle 1: The Path Search Space Pruning (PSSP) principle:

- If P_j is a CDS, we select the top m microservice instances with the shortest estimated data processing time in S_j ($j = i, ..., n$).
- If P_j is a TDS, we select the top m microservice instances with the shortest average data shuffling time between S_{j-1} and S_j in S_j.
- If P_j is not a CDS or a TDS, we select the top m microservice instances with the shortest average execution time in S_j.

In the PSSP principle, m is defined as the pruning parameter of the microservice path search space, and usually $m \ll$ the number of the microservice instances in S_j. In this paper, we set $m = 3$, and this setting can lead to very high quality results which will be shown in the following experiment section. After the above pruning process of the path search space, we can obtain a very simplified layered directed subgraph G_i. In G_i, the single source node represents the microservice instance which performs P_{i-1}, $s_{n+1,0}$ is also the ending node, and there are m nodes in each layer which represents a service class S_j. Note that the weights of each edge in G_i is also updated based on the current operation state of the platform. Then, we reselect the current optimal microservice path in G_i by using the Viterbi algorithm. Finally, we repeat the above operations of the path search space pruning and the subpath reselection until the video pipeline is finished. Our PSPAS approach is summarized in Algorithm 1.

Algorithm 1 PSPAS Algorithm

```
Construct the layered weighted directed graph G based on the current microservice class set S and the performance of the cloud resource;
Set i = 1;
Select the current optimal path SP_i using Viterbi algorithm;
while The pipeline P is not finished do
    Perform the subtask Pi by call the microservice instance s_i. in SP_i;
    Set i = i + 1;
    Perform the path search space pruning operation for the unfinished subtasks according to the above PSSP principle;
    Reconstruct the current subgraph G_i based on the result of the path search space pruning;
    Reselect the current optimal path SP_i using Viterbi algorithm;
end while
```

IV. Experimental Analysis

In this section, we present the experimental evaluation of the effectiveness of the proposed microservice platform and performance-aware microservice path selection approach for the video processing. We have conducted four groups of experiments comparing with other algorithms, and provide the result analysis.

A. Experiment Setup

Before performing the experiment validation, we introduce the experimental configuration used in the following subsection. The experiment platform of the microservice-based video cloud is implemented according to the architecture presented in Section II. The video cloud platform consists of 10 physical servers, and the configurations of the physical servers are shown in Table II. The servers are respectively deployed in two racks each of which has 5 servers. The servers, the switches within two racks, the switch connecting two racks all support 1000 Mbit/s. One of the physical servers is used as the controller node that supports the core management functions of VMSM, CPM, and MSSU, such as resource scheduling, microservice management, and microservice path selection. Another physical server performs the microservice performance monitoring. The other physical servers are the compute nodes consisting of the cloud resource pool, which are equipped with the Docker container engines for supporting the video microservices.

We implement an object tracking application used for the experimental evaluation. The video processing pipeline is divided into nine subtasks, which are respectively data

Table I
PHYSICAL SERVERS

Type	CPU Type	CPU number	memory	number
1	6 core	2	32GB	4
2	10 core	2	32GB	2
3	8 core	2	32GB	2
4	8 core	2	64GB	2

reading, gray-scale processing, Gaussian blurring, interframe difference processing, contour extraction, SIFT feature extraction, feature matching, tracking window drawing, and data writing. Each subtask can be performed by the corresponding microservice instances in the cloud platform. Note that some subtasks of our pipeline can be merged to be an efficient coupled operation, and we carry out the fine-grained division of subtasks for experimental needs. The experimental video data is collected from a real video surveillance system deployed in Fuzhou, China.

B. Experimental Results

This subsection gives the evaluation results of our approach. Firstly, we verify the accuracy of our data processing time model for video microservice. The data set includes 1000 video files with different resolutions and different amount of data, and the amount of the data set is about 10GB. The data set is divided into two parts. 700 video files are used for model training, and another 300 video files are used for model test. The object tracking pipeline consists of nine subtasks which are respectively implemented the corresponding video microservice instances. In this experiment, we ignore the data processing time of the data reading and the data writing. We predict the processing time of the other subtasks by using the data processing time model proposed in sebsection III.B, and then compare the predicted results with the practical execution values measured by the offline processing. In order to minimize the influence of the unstable platform factors, each subtask is performed 10 times, and we use the average processing time of every subtask for the accuracy comparison. Then, we use the accuracy and Mean-Square Error (MSE) to evaluate the accuracy of our data processing time model. The equation of MSE is as follows.

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] \tag{6}$$

where $\hat{\theta}$ is the prediction time of subtask, θ is the actual execution time, and $E[\cdot]$ is the mean operator. The results are given in the Table II. As shown in the table, average accuracies are above 96% except the accuracy of feature extraction subtask is 93.73%, and the each MSE value is acceptable. Next, we can find that the performance of our data processing time model is good enough to be used for guiding the microservice path selection.

For evaluating our PSPAS approach, we compare the efficiency of our approach with those of the following three methods. The optimal microservice path selection method

Table II
ACCURACY AND MSE

Subtask name	Accuracy	MSE
Gray-scale processing	97.04%	0.0125
Gaussian blurring	96.43%	0.0067
Inter-frame difference processing	96.23%	0.0023
Contour extraction	96.10%	0.0196
SIFT feature extraction	93.73%	0.3211
feature matching	97.16%	0.0460
Tracking window drawing	96.73%	0.0034

(called OPTIMAL method in this paper) is obtained by offline task processing and optimal path selection. The Shortest Path Selection (SPS) method finds the microservice path by using the Viterbi algorithm without considering the dynamic changes in the service capabilities of microservice instances. Dynamic Service Path Selection (DSPS) method can online adjust the optimal microservice path according to the current states of microservice instances, but uses a simple end-to-end time model based on the historical performance records.

In the second group of experiment, we measure the task execution time of the four service path selection methods when the number of microservice instances increases. We use a 1GB video data set as the experimental input. The video object tracking task consists of nine subtasks which are respectively performed by the corresponding nine microservice classes. First, the number of instances of each microservice class is randomly generated from integer interval $[10, 40]$. In this group of experiment, we randomly change the resource states of the microservice instances in the process of task execution, but these changes are slight (e.g., within 10%). Then, we perform the video object tracking task in the cloud platform according to the four service path selection methods. This process is repeated 10 times, and we record the average task execution time. Next, the number of instances of each microservice class is randomly generated from the different integer intervals $[41, 70]$, $[71, 100]$, $[101, 130]$, $[131, 160]$, and $[161, 190]$, and then we repeat the above experiment process for each result of microservice instance generation.

Figure 2 shows the results of the second group of experiment. In this figure, we can find that the efficiency of our proposed PSPAS approach is close to that of the OPTIMAL method. However, the task execution time of the SPS method is higher than that of our PSPAS method. This is because the SPS method selects the initial optimal microservice path which is not the practical efficient selection in the online task execution process. Though the DSPS method can improve microservice path adaptively, the efficiency of the DSPS method is worst among the four methods. And it means that the accurate performance model of the microservice instance significantly influences the optimality of the microservice path selection. As the number of the microservice instances increases, the task execution time of every method decreases

accordingly. So the microservice path selection methods have good scalability.

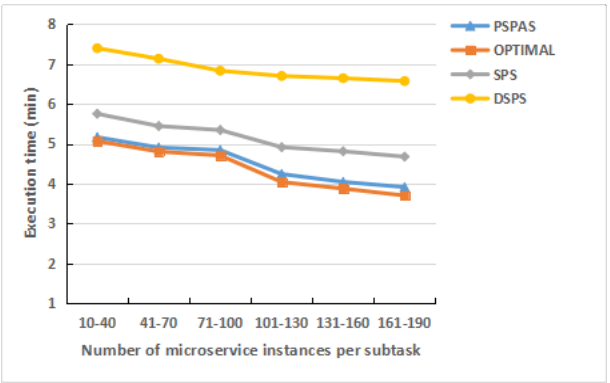

Figure 2. Execution time with slight changes in the resource states of microservice instances

In the third group of experiment, the resource states of the microservice instances change significantly in the process of task execution (e.g., between 30% and 60%). The other configurations and the process of the third group of experiment are the same as the those of the second group of experiment. As shown in Figure 3, compared with Figure 2, there is an obvious gap between the performance curve of our PSPAS method and that of the OPTIMAL method. It is because the dynamic nature of the cloud platform can degrade the effectiveness of our PSPAS method. Similarly, since the static microservice path selection is significantly influenced in this dynamic environment, the performance of the SPS method is reduced. However, our PSPAS method still outperforms the SPS method and the DSPS method.

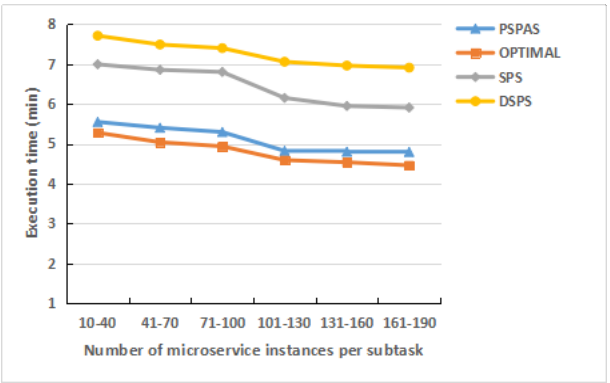

Figure 3. Execution time with significant changes in the resource states of microservice instances

In the fourth group of experiment, we examine the impact of the number of subtasks on the performance of the microservice path selection. In this experiment, we use the above implemented video processing subtasks to construct the different video task combinations with different number of subtasks. The number of the microservice instance per service class is set to 90. Each video subtask combination is performed 10 times in the cloud platform, and we record the average execution time for performance comparison. As shown in Figure 4, the execution time of each microservice path selection method increases as the number of the subtasks increases. It is obvious to see that the efficiency of our PSPAS method is close to that of the OPTIMAL method and outperforms those of the other two methods in the different experiment environment. The gap between the time curve of the PSPAS and those of the SPS and DSPS methods also increases as the number of the subtasks increases.

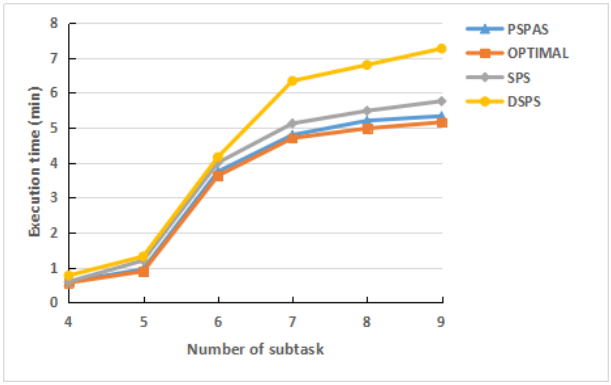

Figure 4. Execution time with different number of subtasks

V. Related Work

With the development of large scale video applications, the cloud computing technology is a promising solution for dealing with the challenges of data-intensive and computation-intensive intelligent video processing tasks. In our previous work [3], we designed a novel video surveillance cloud platform to provide the video surveillance services with the fine-grained cloud resource provisioning. In [2], we proposed a two-stage data distribution approach for optimizing the large scale video processing in the cloud platform. In [13], the authors presented a hybrid cloud model for video surveillance systems with mixed-sensitivity video streams. The literature [14] proposed a video surveillance system based on the cloud computing platform. However, all these previous works take the video tasks as a monolith service in the cloud platform, and there is no microservice division for the video tasks.

Following the service-oriented architecture paradigm, the problem of web service selection and composition has received a lot of attention recently. Alrifai et al. [4] proposed a skyline based service selection approach for QoS-based web service composition. Deng et al. [5] proposed a novel service selection approach for service composition with QoS correlations. In [10], the authors focused on the quality-driven service selection, and a global planning method was used to find the best service components for the web service

composition. Although the objective of these approaches is to get the optimal service composition, they do not consider the runtime states of the service resources and cannot guarantee the efficiency of the online video processing tasks.

To deal with the problem of the static Qos-based service selection, some recent literatures proposed some dynamic and adaptive service selection methods. In [7], the authors proposed an genetic algorithm based approach to optimize the overall QoS of a service composition and leveraged a recovery plan for service adaptation. Wang et al. [8] proposed a method that makes use of multi-agent reinforcement learning to guarantee the adaptation in service composition. They utilized the game theory to make decision on the service optimization direction. Peng et al. [9] proposed an adaptive method rEDA to support re-optimization of dynamic QoS-aware service composition. Although these optimization methods support the runtime service adaptation, these algorithms do not take into consideration the features of the video processing tasks. So they cannot directly be applied to the microservice-based video processing cloud platform.

In this paper, our proposed service path selection approach synthetically takes into account the capabilities of the runtime services and the features of the video processing tasks. In addition, we can optimize the service selection dynamically and adaptively for improving the online execution efficiency of the video tasks.

VI. Conclusions

This paper focuses on the design of the microservice based video cloud computing platform. For improving the execution efficiency of the video processing task in the cloud service platform, we propose a novel performance-aware microservice path selection approach (PSPAS). Our PSPAS approach includes two execution stages which are respectively the initial path selection stage and the adaptive service path updating stage. Firstly, a microservice performance model is presented to guide the appropriate microservice path selection. The performance model not only considers the computing capabilities of microservice instances and the transfer conditions between microservice instances, but also incorporates the characteristics of video tasks. According to the proposed performance model, we choose an initial optimal microservice path by using the shortest path algorithm. Then, we adaptively update the microservice path based on the online running states of microservice instances. Finally, the extensive experimental results show that the proposed PSPAS approach performs better than other widely used methods and can achieve approximate performance of the optimal method which cannot be obtained online.

这个文献引用改成the authors proposed an automated approach based on a genetic algorithm to calculate the Qos-optimal recovery plan for the composite service.

References

[1] X. M. Zhao, H. D. Ma, H. T. Zhang, Y. Tang, and G. P. Fu, "Metadata Extraction and Correction for Large-Scale Traffic Surveillance Videos," *IEEE Big Data 2014*, pp. 412-420, 2014.

[2] Y. Y. Gao, H. T. Zhang, B. C. Tang, Y. P. Zhu, and H. D. Ma, "Two-Stage Data Distribution for Distributed Surveillance Video Processing with Hybrid Storage Architecture," *IEEE Cloud 2017*, pp. 616-623, 2017.

[3] H. T. Zhang, H. D. Ma, G. P. Fu, X. D. Yang, Z. Jiang, and Y. Y. Gao, "Container based Video Surveillance Cloud Service with Fine-Grained Resource Provisioning," *IEEE Cloud 2016*, pp. 758-765, 2016.

[4] M. Alrifai, D. Skoutas, and T. Risse, "Selecting Skyline Services for QoS-based Web Service Composition," *WWW 2010*

Composition with QoS Correlations," *IEEE Trans. on Services Computing*, vol. 9, no. 2, pp. 291-303, 2014.

[6] M. S. Saleem, C. Ding, X. Liu, and C. Chi, "Personalized Decision Making for QoS-based Service Selection," *IEEE ICWS 2014*, pp. 17-24, 2014.

[7] T. H. Tan, M. Chen, É. André, J. Sun, Y. Liu, and J. S. Dong, "Automated Runtime Recovery for QoS-based Service Composition," *WWW 2014*, pp. 563-574, 2014.

[8] H. Wang, Q. Wu, X. Chen, Q. Yu, Z. Zheng, and A. Bouguettaya, "Adaptive and Dynamic Scervice Composition via Multiagent Reinforcement Learning," *IEEE ICWS 2014*, pp. 447-454, 2014.

[9] S. Peng, H. Wang, and Q. Yu, "Estimation of Distribution with Restricted Boltzmann Machine for Adaptive Service Composition," *IEEE ICWS 2017*, pp. 114-121, 2017.

[10] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality Driven Web Services Composition," *WWW 2003*, pp. 411-421, 2003.

[11] H. Matsuba, K. Joshi, M. Hiltunen, and R. Schlichting, "Airfoil: A Topology Aware Distributed Load Balancing Service," *IEEE Cloud 2015*, pp. 325-332, 2015.

[12] K. Lee, H. Yoon, and S. Park, "A Service Path Selection and Adaptation Algorithm in Service-Oriented Network Virtualization Architecture," *IEEE ICPADS 2013*, pp. 516-521, 2013.

[13] C. Zhang, and E. Chang, "Processing of Mixed-Sensitivity Video Surveillance Streams on Hybrid Clouds," *IEEE Cloud 2014*, pp. 9-16, 2014.

[14] D. A. Silva, L. Orellana, and D. Martinez, "Video surveillance based on cloud storage," *IEEE Cloud 2012*, pp. 991-992, 2012.

[15] E. Gaussier, D. Glesser, V. Eeis, and D. Trystram, "Improving Backfilling by Using Machine Learning to Predict Running Times," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis 2015*, 2015.

[16] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. Supinski, and M. Schulz, "A Regression-Based Approach to Scalability Prediction," *2008 ACM International Conference on Supercomputing*, pp. 368-377, 2008.

[17] S. Ross, P. Mineiro, and J. Langford, "Normalized online learning," *Uncertainty in Artificial Intelligence*, 2013.

会议名称最后多了年份，没有页码

没有页码