Performance Evaluation of Video Streaming Service Chains in Cloud Networks with Task Graph Reduction

Frank Loh, Valentin Burger, Florian Wamser, Phuoc Tran-Gia
University of Würzburg, Institute of Computer Science, Germany
{frank.loh,valentin.burger,wamser,trangia}@informatik.uni-wuerzburg.de
Giovanni Schembra, Corrado Rametta
Dipartimento di Ingegneria Elettrica, Elettronica e Informatica (DIEEI) - University of Catania
{schembra,corrado.rametta}@dieei.unict.it

***Abstract*—With the adaption of cloud computing many benefits for service providers as well as private persons are emerging. Especially scalability of services by segmentation into several components helps cloud providers to monitor and analyze there services. Thus, influence factors for performance changes are examined improving the usage for private persons. This linking of components to one service is called service chaining. For the analysis one approach is the creation of analytic models by means of task graph reduction. In this work, each component is described by a task graph node and a processing time distribution. Out of all nodes, a task graph is created and reduced receiving one probability density function characterizing a performance evaluation of the whole service chain. With the example of cloud based video streaming one use case is given and analyzed with a created tool.**

I. Introduction

Cloud Computing describes the provision of IT infrastructure and IT services such as storage space, computing power, or application software as a service over the Internet. Due to the simplicity of instantiating new services and service components one major benefit of cloud computing is scalability.

The typical service in a cloud environment consists of many functions that are mapped on the available infrastructure of the specific data center. These functions are called service components. The linking of several components is called service chaining. The advantage of this structure within the cloud is encapsulation. Each component can be evaluated, scaled or exchanged independently.

From the cloud service provider point of view, the processing in the cloud need to be monitored and influence factors for changing performance must be examined. One approach is the evaluation of service chains based on each component by means of analytic models. However, there are currently no models or existing tools dealing with these challenges.

In this work, analytic modeling of service chains is done with task graph reduction. Each task graph $G = (V, E)$ created in this work consists of nodes or vertices V and edges E. The nodes are describing each service chain component by a processing time probability density function (PDF) while the edges are connections indicating the workflow of the task graph. With task reduction mechanisms the whole task graph is reduced to one node influenced by the input PDFs of all nodes. The output is a single vertex with a processing time PDF describing the analyzed service chain.

The contribution of this work is a method for performance evaluation in cloud networks based on task graph reduction for service chains. Especially the influence of changing PDFs in one service component and the interaction of several linked components can be observed in detail. As a specific example for an application area of this analyzing technique, cloud based video streaming is presented. Within the INPUT project [1] a video streaming setup is created and measurement values are received. These data are evaluated with task graph reduction by a created tool presented in this work.

The remainder is structured as follows. In Chapter 2, related work is summarized. Afterwards, in Chapter 3 the modeling concept and methodology is presented by introducing the components of the created task graphs and the reduction mechanisms. The implementation of the created tool is presented in Chapter 4 while Chapter 5 presents the evaluation and the results of the work. At the end conclusion is drawn in Chapter 6.

II. Related Work

In this section related work with focus on modeling and analyzing methods for service chains and performance parameters in cloud based architectures is summarized. Additionally, since the developed framework deals with service chain analysis for video streaming related work in this context is discussed.

In [2] an approach for analyzing parallel and serial processing of stochastic workload by multi-processor/multi-core processing resources in DC environments is introduced by task graph reduction. A total execution time of the whole system is received while serial and parallel processing of jobs is analyzed in detail. Better results are obtained with parallel processing in low to medium load ranges while serial processing outperforms parallel processing for high loads. In [3], Petri Nets are used for modeling a correct execution together with a deadlock detecting system by the control of state transitions with places and tokens. The drawback of this solution is having no time conditions. This is corrected later by the introduction of timed Petri Nets and stochastic Petri Nets by [4]. In [5] a cloud computing system is analyzed focusing on the performance using a model based on queuing theory. The user's service changes are described by Poisson arrivals. A batch arrival system model is created based on queuing theory and solved in the steady state. Another approach is the introduction of QoS guarantees regarding the response time in a cloud system in [6]. In [7] and [8] analytical techniques based on an approximate Markov chain model for performance evaluation of a cloud computing center are presented. An accurate estimation of the complete probability distribution of the request response time and other important performance indicators is given. Different from the two works above, in [9] multiple priority classes are introduced to analyze performance of clouds. Especially the general problem of resource provisioning is discussed.

By focusing on related work in the video streaming area, the problem of delay between stream generation and delivery is addressed in [10] by renting additional cloud resources on demand to the overlay increasing the total available bandwidth together with the probability of receiving the video in time. The system created in [10] estimates the available capacity and provisions the required resources from the cloud to guarantee a given level of QoS at low cost. In [11] the goal is to reduce the number of cloud resources together with delivering robust and scalable video streaming. Their result saves around 50 % of the total number of resources. Compared to that solution, [12] is addressing the issue with having users from all over the world by leasing and adjusting cloud server resources adaptively in a fine granularity. In contrast to live streaming, [13] is having a closer look at video on demand streaming. A queuing network based model is created and an algorithm is designed and implemented to effectively configure the cloud services meeting user demands in a multichannel video on demand application. In [14], the challenge on how to provide on-demand services to heterogeneous users with different bandwidth requirements is tackled, while in [15] a resource provisioning framework allowing both VoD and LiveTV channel changes both with a deadline constraint is created. Next to frequently viewed VoID or live broadcasting, [16] is having a closer look at streaming unpopular videos using cloud technologies. Especially the issues with low data health and low data transfer rate is investigated.

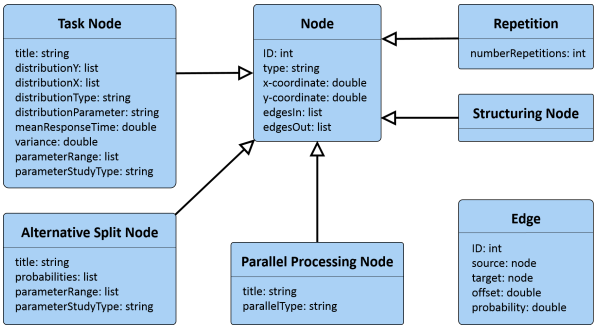


Fig. 1: Task graph components diagram

III. Modeling Concept and Methodology

In this section, the modeling concept and methodology is introduced. To analyze service chains, task graphs are created and evaluated. A task graph is a directed graph $G = (V, E)$ consisting of a finite number of vertices V and edges E. One edge connects two vertices and is weightless pointing from one single vertex to another. All task graph components are presented together with the method of task graph creation and reduction hereafter. The whole process is called task graph reduction in the following.

A. Definition of Task Graph Reduction

In a task graph reduction process, first the graph is created. There, several components are required. Each one contains specific parameters depending on its type. A detailed diagram about all task graph components is depicted in Figure 1 and a brief overview of them is given below.

Task Graph Nodes: Each graph node inherits from the node class, independent on its type. The node class is described by a unique ID, a type as well as an x and y coordinate. The x and y coordinate is required for node visualization. A short introduction to the visualization is presented in the implementation section. Additionally, an array for ingoing and outgoing edges of each node is created to build an adjacency matrix. This matrix is later used for node handling and graph simplification. In the following the attributes of each specific node type are introduced.

Task Nodes: Since task nodes are describing processing units like servers or databases it is the most complex type in this work. Next to the inherited attributes from the node class, it has a title attribute initialized with *node* that can be specified by the user. The distribution type attribute is initialized with *Markov* with 0.2 as distribution parameter. According to the distribution type and the distribution parameter, the PDF can be calculated by the software which is stored in a distribution array. The distribution array contains an x and y vector. Another way to specify the PDF is importing simulation or measurement values to the software. Based on the PDF, for example the mean response time or the variance can be calculated and stored.

The last two attributes, the parameterRange and the parameterStudyType are required for the parameter study introduced in detail at the end of this section.

Alternative Split Nodes: Compared to the task node shown above, the *alternative split* node has four additional parameters. The node's title is *If* by default and a list of probabilities are initialized with zeros. This list indicates the outgoing probabilities from the node towards the successor nodes. Similar to the task node, the *alternative split* has a parameterRange and parameterStudyType attribute for the parameter study.

Parallel Processing Nodes: The parallel task node contains only an additional title attribute that is *Parallel* by default and a parallelType attribute. It can be switched between minimum and maximum indicating different types of parallel nodes.

Repetition Nodes: The *repetition* node has one extra attribute, the numberRepetiton, that is $k = 1$ by default. The k-value defines how often the specific node is passed and evaluated.

Structuring Nodes: The structuring node is dividing different other nodes for better readability and task graph reducibility. It has no parameters next to the inherited ones.

Task Graph Edges: The edge class has five attributes. First, it has a unique ID required to identify the edge. Additionally, each edge has a source and a target node and an optional probability variable, which is only defined if the source node of an edge is an *alternative split* node. In that case, the probability using that edge as outgoing edge is stored there.

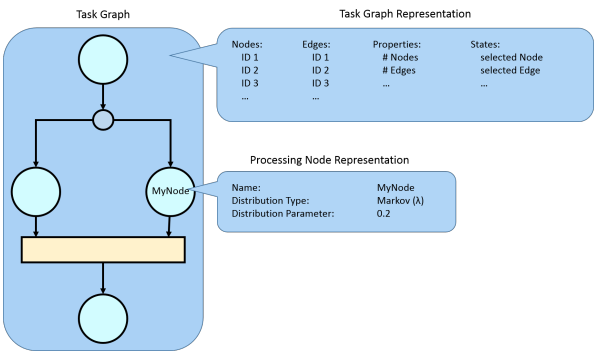
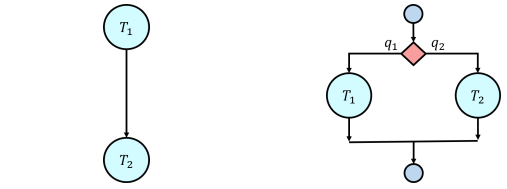


Fig. 2: Task graph representation



(a) Sequential processing (b) Alternative split

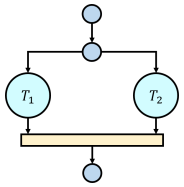
Fig. 3: Sequential processing and alternative split

B. Distribution Representation

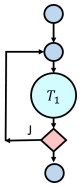
Each task node contains a PDF specified by distribution parameters or imported values. After the initial usage of a node, its distribution is represented by a two dimensional array for the x and y-values. An overview about the task graph display as well as a node and distribution type representation is presented in Figure 2.

C. Task Graph Reduction and Processing Types

To simplify and evaluate a task graph, it can be stepwise reduced. Therefore a part of the graph is selected, pushed to the server for reduction and sent back to the client as a single node. The new node has a calculated PDF dependent on the attributes of the selected graph part. A task graph is reducible if it follows specific requirements: The graph has one start and one end node. Between these nodes each other node must be located on a path between the start and the end node. Last, only specific processing types can be created between these nodes. The calculation of a task graph reduction is based on [2]. In the following the processing types are introduced briefly and the calculation is presented.



(a) Parallel processing



(b) Repetition

Fig. 4: Parallel processing and repetition

Sequential Processing: The first type called sequential processing is the concatenation of several nodes, $T_1, ...T_n$, shown in Figure 3a for two nodes. Let $f_i(t)$ be the PDF for node i. The resulting PDF $f_(t)$ after reduction is calculated by a convolution of each nodes PDF according to

$$f(t) = f_1(t) \otimes f_2(t) \otimes \cdots \otimes f_n(t), n \in \mathbb{N} \quad (1)$$

Alternative Split: The *alternative split* is started with an alternative split node containing two or more outgoing edges, as depicted in Figure 3b. The calculation of the PDF after reduction is described by

$$\begin{aligned} f(t) &= q_1 f_1(t) + q_2 f_2(t) + \cdots + q_m f_n \\ \sum_{i=1}^{m} q_i &= 1, n, m \in \mathbb{N}. \end{aligned} \quad (2)$$

Parallel Processing: The *parallel processing* type is presented in Figure 4a. This type is used when tasks are executed simultaneously. In the minimum case, the processing continues as soon as one task of the parallel part is finished. In the maximum case when all are done. The PDF $f(t)$ of the reduced node for two nodes as an example is obtained by

$$\begin{aligned} X &= max(X_1, X_2) : \\ f(t) &= f_1(t)F_2(t) + f_2(t)F_1(t) \end{aligned} \quad (3)$$

for the minimum case and by

$$\begin{aligned} X &= min(X_1, X_2) : \\ f(t) &= f_1(t)[1 - F_2(t)] + f_2(t)[1 - F_1(t)] \end{aligned} \quad (4)$$

for maximum.

Repetition: The repetition processing type is used to indicate an iteration through a specific node like depicted in Figure 4b. The PDF $f(t)$ of the resulting node after reduction is received by

$$f(t) = f_1(t) \otimes \underbrace{[f_1(t) \otimes \cdots \otimes f_1(t)]}_{\mathrm{J}} \quad (5)$$

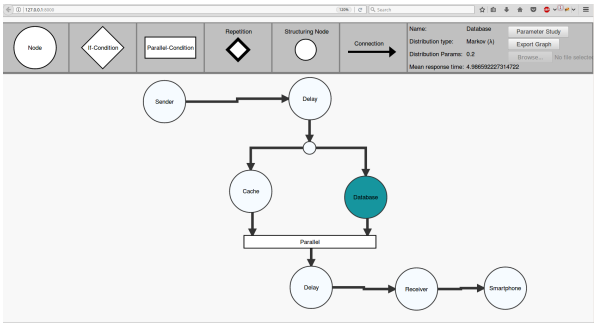


Fig. 5: Task graph reduction tool

D. Parameter Study and Analysis Metrics

To simulate different processing time distributions within one task node, a parameter study can created. There, a single parameter like λ in a Markov distributed PDF is varied within a specific value range. For each parameter setting, the whole task graph is reduced and the PDF is generated. At the end, variations in mean processing time or variance can be detected based on the parameter selection.

In this way, the changing of specific metrics can be simulated and the impact on the whole service chain can be detected. Typical metrics in this work are delay, cache hit rate or mean processing time of a sub-service or the whole service chain.

IV. Implementation

The following chapter presents an overview on the implementation done for this work. An important aspect of the created web application is simplicity. The user is able to create a task graph, specify all task nodes by a PDF and simplify the graph in a graphical user interface (GUI). A screenshot of the web GUI is shown in Figure 5. When creating a task graph, a representation of each graph component is added to the document object model (DOM) of the web page to display it for the user. The platform for running the tool is a common web browser like Firefox or Chrome with activated JavaScript. For temporary data storage as well as visualization, modification and data handling a web framework called Data-Driven Documents (D3) is used. D3 is a JavaScript library to create dynamic and interactive data visualizations on common web browsers. It works with HTML5, CSS, as well as SVG standards for data handling. The D3 code is embedded inside an HTML page creating SVG elements using pre-build JavaScript functions. It is possible to change, add or style each object according to the users intention. Data can be imported or exported as JSON objects or into CSVs.

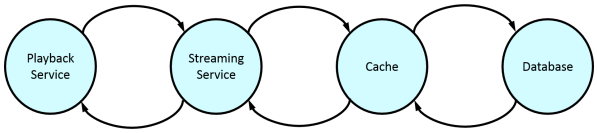


Fig. 6: Cloud-based video streaming service chain

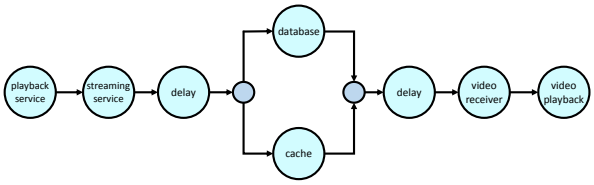


Fig. 7: Streaming task graph

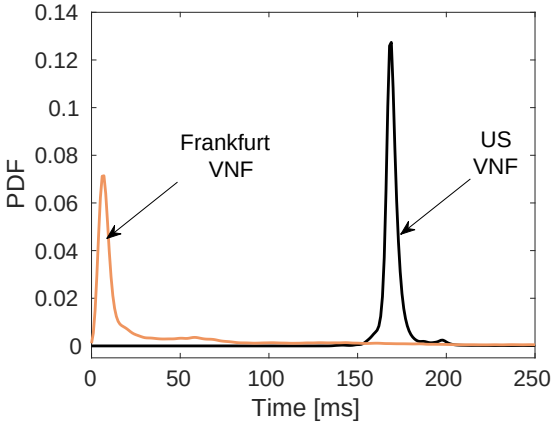


Fig. 8: Comparison of video access times

V. Evaluation of a Video Streaming Service Chain

As an example of service chain evaluation with task graph reduction, in the following a typical video streaming service chain, like presented in Figure 6 is evaluated. As soon as a user requests a video at the playback service, the streaming service decides where to access the video. One possibility is a video request from the cache. If the content is not located there, a database in a bigger data center is accessed. Based on this service chain, a streaming task graph is created depicted in Figure 7. In the following several streaming scenarios are presented and evaluated.

A. Scenario Definition

Based on the task graph shown in Figure 7, it is possible to define several streaming scenarios by changing the database access method and location. The first scenario is the comparison of content access from a regional datacenter or a big datacenter at another continent. Next, either the edge cloud or any big datacenter is accessed. In this scenario the probability for an edge cloud access is varied simulating different probabilities of having the content located in the edge cloud. Additionally, the data request from either the regional or the remote database is compared together with the edge cloud access. For both scenarios, different load conditions at the video receiver are measured and evaluated. Additionally, delay values from a VNF located in Wuerzburg to a VNF in Frankfurt and one in the US are measured while the delay to the cache is assumed to be negligible. Since the authors in [17] present 1.5ms as a meaningful value for the mean database access time, the service time PDF for each database is an exponential distribution with a mean of 1.5ms hereafter. In the following, the evaluation of these scenarios is presented.

B. Scenario Evaluation

First of all, the influence of content location on the access time distribution is evaluated. For that reason, the data request from Wuerzburg to Frankfurt and to the US is compared and evaluated. Figure 8 shows the result of a task graph reduction with three nodes, delay to the database, database and delay back to the user. This experiment compares the access to Frankfurt in orange and the US VNF in black showing the PDF at the y-axis and the time in ms at the x-axis. The figure shows a peak for the access to Frankfurt at 6.5ms and for the USA at 169ms. The probability for receiving data from the US faster than 130ms is 0 and the mean is 178ms. Compared to the low peak at 6.5ms for the access to Frankfurt the mean is 63ms. It is remarkable having a probability mass of 94% below 200ms for the access to the US compared to about 88% to Frankfurt. This is indicating higher disturbances in the network to the Frankfurt VNF. However, in general the content request from the closer VNF in Frankfurt is faster in average than from the US.

VI. Conclusion

Acknowledgment

This work was partly funded in the framework of the EU ICT project INPUT (H2020-20-14-ICT-644672) and the DFG grant QoE-DZ (TR257/41). The authors want to thank Lam Dinh-Xuan for his delay measurement results.

References

[1] [Online]. Available: http://www.input-project.eu/

[2] P. J. Kuehn, "Performance and energy efficiency of parallel processing in data center environments," in *International Workshop on Energy Efficient Data Centers*. Springer, 2014, pp. 17–33.

[3] J. L. Peterson, "Petri net theory and the modeling of systems," 1981.

[4] J. Wang, "Stochastic timed petri nets and stochastic petri nets," in *Timed Petri Nets*. Springer, 1998, pp. 125–153.

[5] Z. Yong-Hua, Z. Zhen, Z. Fan-Zi, and L. Yuan, "The cloud computing center performance analysis model based on queuing theory," *Open Automation and Control Systems Journal*, vol. 7, pp. 2280–2285, 2015.

[6] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 492–507, 2014.

[7] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems," *IEEE Transactions on parallel and distributed systems*, vol. 23, no. 5, pp. 936–943, 2012.

[8] ——, "Modelling of cloud computing centers using m/g/m queues," in *2011 31st International Conference on Distributed Computing Systems Workshops*. IEEE, 2011, pp. 87–92.

[9] W. Ellens, J. Akkerboom, R. Litjens, H. van den Berg *et al.*, "Performance of cloud computing centers with multiple priority classes," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 245–252.

[10] A. H. Payberah, H. Kavalionak, V. Kumaresan, A. Montresor, and S. Haridi, "Clive: Cloud-assisted p2p live streaming," in *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2012, pp. 79–90.

[11] H.-Y. Chang, Y.-Y. Shih, and Y.-W. Lin, "Cloudpp: A novel cloud-based p2p live video streaming platform with svc technology," in *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, vol. 1. IEEE, 2012, pp. 64–68.

[12] F. Wang, J. Liu, and M. Chen, "Calms: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 199–207.

[13] Y. Wu, C. Wu, B. Li, X. Qiu, and F. C. Lau, "Cloudmedia: When cloud on demand meets video on demand," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011, pp. 268–277.

[14] Z. Chen, C. Lin, and X. Wei, "Enabling on-demand internet video streaming services to multi-terminal users in large scale," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 4, pp. 1988–1996, 2009.

[15] V. Aggarwal, X. Chen, V. Gopalakrishnan, R. Jana, K. Ramakrishnan, and V. A. Vaishampayan, "Exploiting virtualization for delivering cloud-based iptv services," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. IEEE, 2011, pp. 637–641.

[16] Y. Huang, Z. Li, G. Liu, and Y. Dai, "Cloud download: using cloud utilities to achieve high-quality content distribution for unpopular videos," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 213–222.

[17] B. Lebiednik, A. Mangal, and N. Tiwari, "A survey and evaluation of data center network topologies," *arXiv preprint arXiv:1605.01701*, 2016.