Joint Optimization of Multidimensional Resources Allocation in Cloud Networking

Jialong Li^{1,2,3}, Kangqi Zhu^{1,2}, Nan Hua^{1,2}, Chen Zhao^{1,2}, Yanhe Li^{4}, Xiaoping Zheng^{1,2}, and Bingkun Zhou^{1,2}
^{1}Beijing National Research Center for Information Science and Technology (BNRist), Beijing 100084, China
^{2}Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
^{3}Max Planck Institute for Informatics, Germany
^{4}Tsinghua Unigroup, Tsinghua University, Beijing 100084, China
huan@mail.tsinghua.edu.cn*, xpzheng@mail.tsinghua.edu.cn

***Abstract*—Cloud networking enables flexible service deployments and agile function managements. The convergence of cloud and networking requires the integration of transmission, computation, and storage resources. In this paper, a unified mathematical model on joint optimization of transmission, computation, and storage resources allocation is established, aiming at reducing the job completion time and improving the resource utilization efficiency. To so do, computation resources are first virtualized and then discretized in the time domain. Then we realize the coordination of routing process and transmission, computation resources in cloud networking. Simulation results demonstrate that the propose method could reduce the job completion time by 33%.**

***Index Terms*—Cloud networking, multidimensional resources, joint optimization**

I. Introduction

Networking and cloud computing are responsible for communications and data process/storage respectively, working separately and lack collaborations. The convergence of networking and cloud computing, namely cloud networking [1], enables faster service deployment and higher resource utilization efficiency. Unlike the legacy network, cloud networking requires the allocation of multidimensional resources (transmission, computation, and storage) simultaneously [2], which plays a vital role in the convergence of cloud and networking. Furthermore, the service modelling is different from the traditional network architecture, where data are processed in a centralized way and the networks act as pipes connecting end users and data centers. Benefit from the deployment of computation and storage resources at the network edge, task executions among geographically distributed nodes are possible, which is essential for some applications, such as federated learning (FL) [3], [4] and intelligent video surveillance (IVS) [5]. It is crucial to take into account the availability of the multidimensional resources as well as the distributed tasks' interdependence when assigning distributed tasks. If the task assignment and multidimensional resource allocation could be jointly conducted, the entire job completion time could be reduced greatly.

In terms of the collaborative allocation of multidimensional resources, Ref. [6] proposed a strategy by jointly adjusting the size of allocated computation and transmission resources. In this way, multiple computation results generated by distributed tasks could be transmitted to the next node simultaneously, avoiding the buffer time among these results. However, the buffer time reduction is limited, and its performances are highly dependent on the composition characteristic of the job. Ref. [7] introduced the resource utilization balance factor to allocate multidimensional resources evenly. Unexpected long buffer time caused by the shortage of resources could be avoided. Ref. [8] designed a multi-state adjustment algorithm for multidimensional resource allocation. The task mapping procedures are adjusted according to the availability of the transmission resources as well as the location of the raw data. However, this strategy is based on greedy methods and could not obtain the theoretical optimal value. Furthermore, this work does not take into account the availability of computation resources.

The traditional method based on single-dimensional resource optimization can not realize the collaboration of multidimensional resources, which will cause higher completion time and lower resource utilization efficiency. A more efficient and practical strategy is required in distributed task assignments and allocations of multidimensional resources. In our previous work, we proposed a mathematical model aiming at minimizing the job completion time [9]. However, this optimization model is too complex and requires plenty of time to obtain optimal results. In this work, we focus on minimizing the completion time of jobs with sequential task interdependence. To this end, we design a strategy based on the joint optimization of NEtwork and COmputation resources (NECO). Simulation results validate the effectiveness of NECO compared with other benchmark algorithms.

The rest of the paper is organized as follows. Multidimensional resources and service modelling are shown in Sec. II. In Sec. III, NECO algorithm is presented in detail. We demonstrate the simulation performances in Sec. IV. We conclude this work in Sec. V.

II. Multidimensional Resources and Service modelling

Fig. 1 is a schematic diagram of multidimensional resources in cloud networking. It can be seen from the figure that distributed task executions involve the participation of three types of resources, namely transmission, computation, and

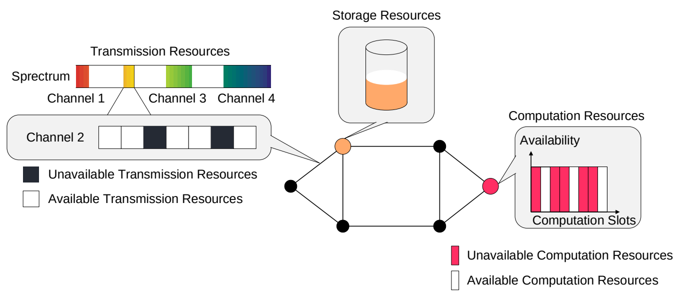


Fig. 1. Multidimensional resources in the network.

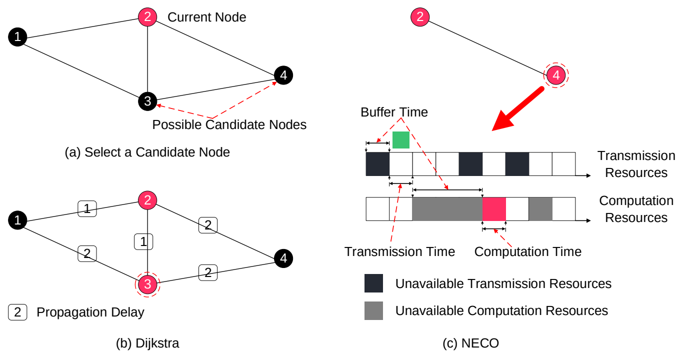


Fig. 2. Illustration of Dijkstra and NECO strategy.

storage resources. Transmission resources are used for transmitting raw data and computation results generated by tasks. Computation and storage resources are responsible for task executions and data storage. In this work, we mainly focus on transmission and computation resources. A unified abstraction for multidimensional resources is critical in cloud networking. In this work, transmission and computation resources are discretized in the time domain. For transmission resources, the optical networks are based on optical time slices switching (OTSS) [10], where optical channels are divided into repetitive frames and each frame is further slicing to multiple slots. Under such circumstances, data transmission could occupy several continuous slots in a selected wavelength. As for computation resources, we assume that a task will occupy all computation resources in the node for execution, while other tasks will be queued until resources are released. In this way, the availability of transmission and computation resources could be measured by time slots, and we could design the NECO strategy based on this assumption.

In terms of the service model, a job is decomposed into several tasks under interdependence constraints. The interdependence among computation tasks could be modelled by a directed acyclic graph (DAG). Three typical interdependence models are mentioned in Ref. [11], namely sequential, parallel, and general interdependence. In this paper, we mainly consider the sequential interdependence, i.e., the DAG is a chain.

III. NECO Strategy

A. Discussion

Fig. 2 presents an example of Dijkstra and NECO algorithm. In Fig. 2(a), Node 2 is the current node, and the next step is to select a candidate node. Node 3 and 4 are both possible candidate nodes. Fig. 2(b) shows how Dijkstra algorithm determines the candidate node. Dijkstra method tries to minize the propagation delay and does not take computation and transmission resources into consideration. It adopts propagation delay as the cost, and Node 3 is chosen as the candidate node since the path $2 \rightarrow 3$ is the shortest.

NECO deals with this problem in another way. It tries to find the optimal completion time by considering the buffer time, transmission/computation time, and the propagation delay. From Fig. 2(c), we could learn that the time required by the task execution and the result transmission is both 1 time slice. The buffer time caused by the shortage of transmission and computation resources are 1 and 3 time slices, respectively. It is a one-step strategy and jointly takes computation and transmission resources into consideration, which will select the candidate node with the minimum completion time.

Algorithm 1 describes the NECO algorithm in detail. NECO consists of three stages, namely Initialization, Update, and Relaxation. In the Initialization stage, the distances from the source node s to other nodes are infinity and all nodes are unvisited. Then the first task is executed in source node s. In the second stage, the distances from sources node s to its neighbour nodes are updated. In the Relaxation stage, if a visiting node could realize a shorter distance than the current distance, update the distance and set the visiting node as a predecessor. We could find the shortest paths and minimum distances from the source node s to other nodes by the three stages.

Then we analyze the time complexity of NECO. The time complexity of Dijkstra algorithm is $O(|N|^2)$, where N is the number of nodes. In NECO algorithm, latency is used to replace the physical length of the edge. The time complexity of calculating the latency for an edge is $O(|W||S|^2)$, where W and S denote the number of wavelengths and the number of minimum time slices in an OTSS frame, respectively. The number of nodes in the DAG is L, so there are $L-1$ edges that requires latency calculation. The overall time complexity of NECO is $O(|W||L||N|^2|S|^2)$.

B. Benchmark Algorithms

Besides Dijkstra algorithm, we further design another two benchmark algorithms, namely Optimal and depth-first search (DFS). In Optimal strategy, all possible mapping cases will be checked and the case with minimum completion time will be selected. In other words, optimal results could be obtained by Optimal strategy.

As for DFS scheme, it could be regarded as a random assignment policy. First, we select a physical node randomly as a root node and adopt DFS to search the following nodes as far as possible. Then tasks are mapped into these nodes

Algorithm 1 : NECO Algorithm

```
Require:
    $\mathcal{N} \leftarrow$ set of nodes;
    $dis \leftarrow$ distance vector; dis[v] $\leftarrow$ distance from source s to node v;
    $pre \leftarrow$ indicate a node is already visited or not;
    $ns(v) \leftarrow$ start time of the task execution in node v;
    $es(s, d) \leftarrow$ start time of the computation result transmission between edge (s, d);
    $pl(s, d) \leftarrow$ propagation latency for edge (s, d);
    $ct(v) \leftarrow$ task execution time required in node v;
    $tt(s, d) \leftarrow$ computation result transmission time required between edge (s, d);
1:  // Initialization
2:  for each node v in $\mathcal{N}$ do
3:      dis[v] = Infinity
4:      pre[v] = Unvisited
5:  end for
6:  calculate ns(s);
7:  dis[s] = ns(s) + ct(s)
8:  pre[s] = Visited
9:  // Update the distance vector
10: for each neighbor v of s do
11:     calculate es(s, v);
12:     calculate ns(v); ns(v) should satisfy: ns(v) $\geq$ es(s, v) + tt(s, v) + pl(s, v);
13:     dis[v] = ns(v) + ct(v)
14: end for
15: // Relaxation operation
16: while not all elements in pre are Visited do
17:     u $\leftarrow$ node with min dis[u] and pre[u] == Unvisited
18:     for each neighbor v of u do
19:         if pre[v] is Visited then
20:             continue
21:         end if
22:         calculate es(u, v);
23:         calculate ns(v); ns(v) should satisfy: ns(v) $\geq$ es(u, v) + tt(u, v) + pl(u, v);
24:         if dis[v] $>$ ns(v) + ct(v) then
25:             dis[v] = ns(v)+ ct(v)
26:         end if
27:     end for
28:     pre[u] = Visited
29: end while
```

in order. In DFS scheme, the task assignment and resource allocations are separated.

IV. Simulation Evaluation

A. Simulation Setup

A 6-node mesh topology is used to evaluate the performances in this paper. The distance between two adjacent nodes is 20 km. One wavelength is deployed in each link. The length of an OTSS frame is 100 ms. Each frame is further divided into 1000 minimum slices, which means the length of a minimum slice is 100 μs. As for the parameters of DAGs, the minimum and maximum length of the DAG are 2 and 4, respectively. The required time for task execution in the physical node ranges from 100 to 200 μs. And the transmission time of the computation results varies from 100 to 300 μs.

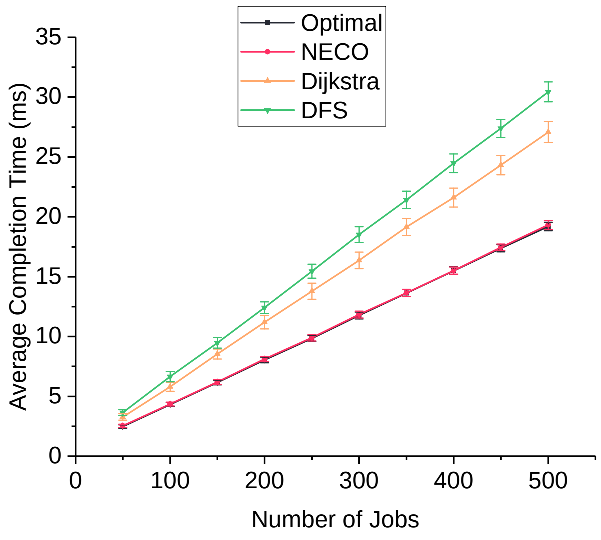


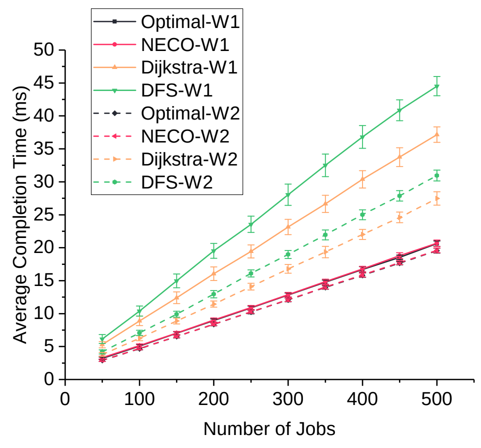
Fig. 3. Average completion time comparison between Optimal, NECO, Dijkstra, and DFS under different number of jobs.

B. Average Completion Time Performance

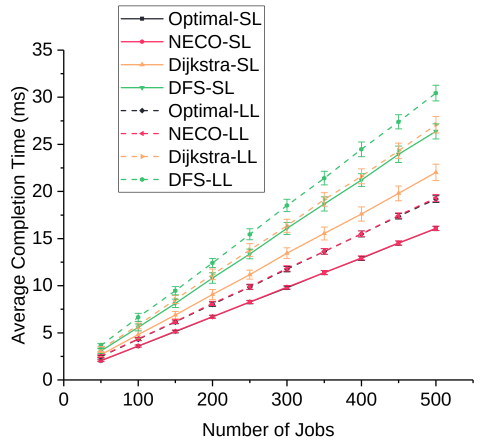
Fig. 3 shows the average completion time comparison between Optimal, NECO, Dijkstra, and DFS under different number of jobs. From this figure, we could learn that average completion time increases with the number of jobs. Among these four algorithms, Optimal algorithm and NECO could realize lower average completion time than Dijkstra and DFS methods. When the number of jobs is 500, the average completion time for DFS and Dijkstra are around 31 ms and 27 ms, respectively. Optimal algorithm and NECO achieve 18 ms average completion time under the same condition, which is 13 ms and 9 ms lower than DFS and Dijkstra, respectively. Furthermore, Optimal and NECO performances are almost identical, and it is hard to tell the differences.

Fig. 4(a) demonstrates average completion time performance under the number of wavelengths of 1 and 2, respectively. W1 and W2 denote the number of wavelengths deployed in each link is 1 and 2, respectively. We could notice that NECO-W2 realizes lower average completion time than NECO-W1. It is the same case with the other three algorithms. The wavelength that provides the lowest buffer time before the transmission will be selected in priority when multiple wavelengths are deployed, which is why W2 could achieve better performances than W1.

Then we study the results under different average lengths of the DAG in Fig. 4(b). SL (short length) denotes the length of the DAG ranges from 2 to 3, while the length ranges from 2 to 4 for LL (long length). In our simulation, the length of the DAG is generated uniformly from the minimum to the maximum value. The average lengths of SL and LL are 2.5 and 3, respectively. The average completion time of LL is higher than that of SL. DAGs with longer average lengths require more physical nodes to execute tasks and traverse more links, leading to higher average completion time.

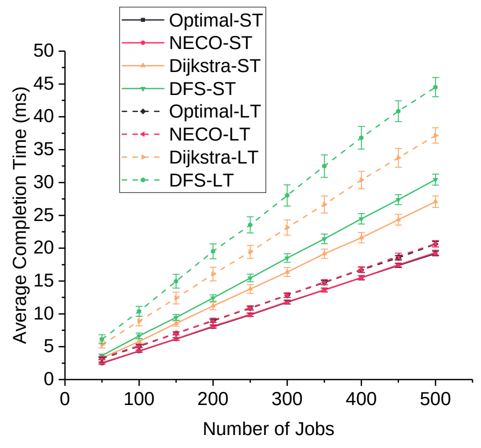


(a)

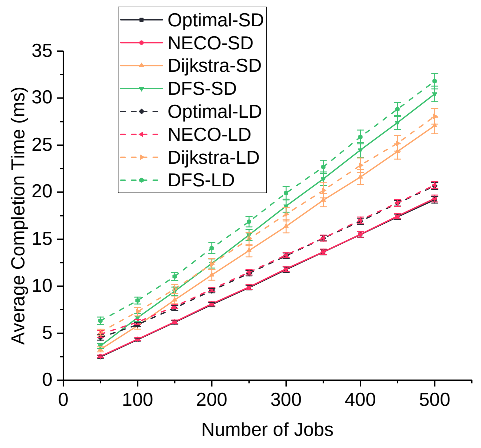


(b)

Fig. 4. (a) Average completion time performance under the number of wavelengths of 1 and 2, respectively. (b) Average completion time performance under different average lengths of the DAG, respectively.



(a)



(b)

Fig. 5. (a) Average completion time performance under the maximum computation result transmission time of 300 and 1000 μs, respectively. (b) Average completion time performance under different propagation delays.

Fig. 5(a) shows the average completion time performances under different computation result transmission time. ST (short transmission time) and LT (long transmission time) represent the computation result transmission time are 300 and 1000 μs, respectively. From the figure, we could learn that the average completion time of LT is higher than that of ST. The longer the transmission time is, the more transmission resources are demanded. The buffer time before the result transmission becomes larger since it is harder to find more available transmission resources, resulting in higher job completion time.

Finally, the performances under different propagation delays are discussed here. SD denotes short distances and LD represents long distances. SD and LD denote the distances between two adjacent nodes are 20 and 200 km, respectively. Optical signals propagates at a speed of 200 km/ms. The propagation delay in SD for each link is 100 μs, while the value is 1000 μs for LD. Fig. 5(b) indicates that the average completion time of LD is larger than that of SD. The reason is that the propagation delay is also considered in the completion time. It is worth noting that the range of the metro network is generally less than 100 km, and the distance between two adjacent nodes is less than this value.

V. Conclusion

In this work, we focus on the integration of transmission and computation resources in cloud networking. To this end, transmission and computation resources are first discretized in the time domain. Based on this, we design a NECO strategy aiming at minimizing the job completion time. Simulation results are demonstrated and validate the effectiveness of our proposed policy.

Acknowledgment

This work was supported in part by the National Key R&D Program (No.2020YFB1805602), and the National Natural Science Foundation of China (No.61871448).

References

[1] Q. Duan, S. Wang, and N. Ansari, "Convergence of networking and cloud/edge computing: Status, challenges, and opportunities," *IEEE Network*, vol. 34, no. 6, pp. 148–155, 2020.

[2] M. Ruffini, "Multidimensional convergence in future 5G networks," *Journal of Lightwave Technology*, vol. 35, no. 3, pp. 535–549, 2016.

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[4] B. Shariati, P. Safari, A. Mitrovska, N. Hashemi, J. Fischer, and R. Freund, "Demonstration of federated learning over edge-computing enabled metro optical networks," in *2020 European Conference on Optical Communications (ECOC)*. IEEE, 2020, pp. 1–4.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] J. Zhang, L. Cui, Z. Liu, and Y. Ji, "Demonstration of geo-distributed data processing and aggregation in mec-empowered metro optical networks," in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*. IEEE, 2020, pp. 1–3.

[7] H. Yang, J. Zhang, Y. Zhao, J. Han, Y. Lin, and Y. Lee, "Cross stratum optimization for software defined multi-domain and multi-layer optical transport networks deploying with data centers," *Optical Switching and Networking*, vol. 26, pp. 14–24, 2017.

[8] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3512–3524, 2018.

[9] J. Li, N. Hua, C. Zhao, Y. Li, X. Zheng, and B. Zhou, "Towards low-latency distributed tasks collaboration by joint optimization of transmission, computation and storage resources allocation in edge computing," in *Asia Communications and Photonics Conference*. Optical Society of America, 2020, pp. M4A–210.

[10] N. Hua and X. Zheng, "Optical time slice switching (OTSS): An all-optical sub-wavelength solution based on time synchronization," in *Asia Communications and Photonics Conference*. Optical Society of America, 2013, pp. AW3H–3.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.