Next Generation Energy-Aware In-house WiFi Router based Computing Strategy for handling Large-Scale workloads

Abstract

The growth of data volume and the popularity of mobile devices have spawned a large number of low-latency, high-computing applications. At the same time, under the wave of global new energy, the energy efficiency utilization of electronic equipment has also received more and more attention. Due to a large delay and occupying the large bandwidth of cloud computing, and the limited computing power of end devices, mobile edge computing is considered to be the best way to achieve low latency of complex tasks. Wifi routers can provide high bandwidth and can work as intermediate nodes for forwarding tasks. In this report, we propose to use dynamic programming based on wifi routers to dynamically schedule computing tasks to edge computing nodes to achieve maximum bandwidth utilization, under the energy consumption constraint of the user. The experiment result shows that the DP method compared the with other three traditional algorithms can get higher bandwidth by using less Energy cost and fewer numbers workers. Besides, we can effectively search for the last phase Table to save much time by using Computation Reusability.

1 Introduction

Scheduling computing strategy, as an effective method to reduce delay and improve devices utilization, has attracted more and more attention. The computing strategy can transfer the user's computing task to an edge node with a lower load or a higher bandwidth, minimizing the task execution time and providing the user a low-latency experience. This provides two advantages: 1) an edge node that is close to a user can act as a cloud service provider, decreasing the latency of the task requirement. 2) It can also improve the network transmission of the centralized server [1].

In recent years, artificial intelligence has been proved to solve some complex problems using a black-box method. Therefore, many scholars choose to use deep learning methods to implement computing strategies for handling Large-Scale workloads by building up a general platform to support self-learning procedures to make decisions with less historical data compared with traditional methods. DeepRM [3] translates packing tasks with multiple resource demands into a learning problem. It visualizes the resource profile to represent state-input with a backlog to buffer the arriving jobs and uses a reinforcement learning algorithm to minimize average slowdown. DL2 [4] is a DL-driven scheduler for DL clusters, targeting global training job expedition by dynamically resizing resources allocated to job using supervised learning and reinforcement learning. Harmony [5] presents a deep learning-driven ML cluster scheduler that places training jobs in a manner that minimizes interfaces and maximizes performance by using an auxiliary reward prediction model, which is trained using historical samples and used for producing reward for unseen placement and a reinforcement learning model to make a job placement decision.

However, the deep learning method is not suitable for In-house WiFi Router based Computing Strategy, because they require a long period of training to come to convergence, which is not good at making a quick and good decision.

From another perspective, in a cloud computing environment, through resource allocation and dynamic virtual machine configuration, bandwidth can be fully utilized to complete task migration. VScaler [6] implements autonomic resource allocation using a new method to reinforcement learning based on a parallel learning strategy in order to get an optimal convergence speed. VCONF [7] employs model-based RL algorithms to automate the VM configuration process in terms of both adaptability and scalability. But most of the time, cloud computing strategy has too much latency to handle user's requests on time, resulting in a bad user experience.

In-house WiFi Router-based Computing Strategy to handle Large-Scale workloads needs to pay more attention to network and bandwidth, which is important to transfer user' tasks to local edge computing nodes. As suggested in LVRM [8], demanded network bandwidth among VMs should be given higher priority while allocating users' tasks data to computing workers as insufficient network bandwidth may detain the task execution.

In this report, we propose an In-house computing system using wifi-routers as agents to assist low-spec nodes to offload tasks to some of the most compliant nodes. By maintaining a DP table, wifi-router can quickly assign tasks to nodes with lower load or higher network rates, in order to maximize bandwidth utilization under certain energy consumption constraints.

2 Related Work

2.1 Mobile Edge Computing

Mobile edge computing has been widely used in In-house or local Wifi router based computing strategies to handle large-scale workloads from a detailed survey [10]. EMM [9] propose to use energy-aware mobile management(EMM) strategy, in order to decrease the delay because of computation and radio access, under the energy constraint.

For a single-user an energy-optimal binary offloading policy is from [11] with a stochastic wireless channel to converge energy for mobile devices, executing applications in the mobile device or offloading to the cloud, while Markov decision used as a search method to optimize task scheduling strategy with random task arrivals is used in [12]. For multi-user MEC, there are two points of view to be considered centralized [13] and decentralized [14], both of which take computation resource management into account to optimize performance. However, except for single MEC, most of them ignore user mobility.

As for peer-to-peer offloading methods, [15] considered stochastic computation offload in terms of the stochastic property of CPU states and wireless channel. And this paper [16] proposes to use the heterogeneous computation resources at the network edge to get an optimal energy efficiency within end devices while satisfying delay requirements.

In this report, we propose an In-house Wifi routers system-based computing strategy for handling large-scale workloads. WiFi routers are designed to provide high data throughput with considerable computing power, which also acts as a network boundary. Especially, LAN network is of high speed and low latency, which is good at providing a fast network to handle large-scale workloads from user's end devices.

2.2 Dynamic Programming

Dynamic programming emerged in the 1950s from R. Bellman. The object of dynamic programming research is a multi-stage decision-making problem, which can be transformed into a series of interrelated single-stage optimization problems. Decisions need to be made at each stage. After the decision of each stage is determined, a decision sequence is obtained, which is called a strategy. The problem of multi-stage decision-making is to seek a strategy to optimize the overall goal of each stage. The idea of dynamic programming is to divide the solution of a problem into n stages, and the strategy adopted from the current stage to the final step to realize the goal is the optimal

strategy.

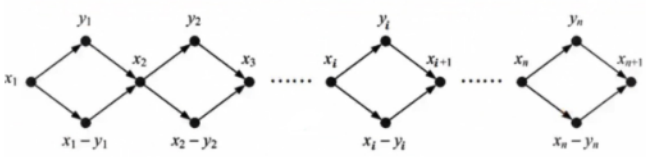
For example, suppose the total number of resources in the kth stage is X_k, and the number of resources invested in department A is Y_k. Then the amount invested in department B is X_k-Y_k. Then you can get the income g(Y_k)+h(X_k-Y_k), and recover aX_k+b*(X_k-Y_k). Therefore, the problem will be get Y_1,Y_2,...,Y_n, to maximize

$$\sum_{1<=k<=n} g(y_k) + h(x_k - y_k) \tag{1}$$

under the constraints of

$$\begin{aligned} & X_1 = X \\ & X_2 = a * Y_1 + b * (X_1 - Y - 1) \\ & ..... \\ & X_n = a * Y_{n-1} + b * (X_{n-1} - Y_{n-1}) \\ & Y_k >= 0, X_k >= 0, k = 1...n-1 \end{aligned} \tag{2}$$

The above problem can be solved by using backward optimization: f_k is



the current number of resources as X, and then after k stages to complete the system goal, the maximum income obtained is:

$$\begin{cases} f_k(x) = max\{g(y) + h(x-y) + f_{k-1}(ay + b(x-y))\}, k >= 2 \\ f_1(x) = max\{g(y) + h(x-y)\}, k = 1 \end{cases} \tag{3}$$

For multi-stage decision-making problems, a system can be divided into several stages. At any stage k, the state of the system can be represented by x_k. Each state x_k has a decision set $Q_k(x_k)$, from which a decision q_k is selected, the state x_k is transferred to the next state x_{k+1}=$T_k(x_k, q_k)$, and the gain $R_k(x_k, q_k)$. The target of the system is to select a decision in its decision set at each stage to maximize the total benefit of all stages.

3 System Model

3.1 System Architecture

The whole system works to connect to the WAN through one or more controllable networks and devices, usually in the places like homes or public areas, or company environment. In each of the local WAN systems, there may be one or more wireless routers and tremendous end devices that connect to Wifi to transform or get data such as phones, computers and IoT devices. Therefore, a well-designed computing strategy and system is the key to maximizing network bandwidth and optimizing energy consumption.

3.1.1 Switch layer

The Switch layer acts as a management role in connecting all the wifi router clusters. It is a logical abstraction of wifi routers that enables all alive routers to interconnect with each other. In a large-scale WAN network, it is merged by the upstream ISP.

3.1.2 WiFi router layer

The wifi routers layer is the most important part of the system connecting all the user end devices. It can receive offloaded tasks from users, calculate to match the optimal assignment strategy using DP method, and schedule tasks to target worker nodes. Besides, Wifi routers can interconnect with other clusters routers to share information in a centralized or decentralized way.

3.1.3 User Equipment Layer

User equipment layer is the devices that have limited computation power, such as IoT devices, phones, computers. Due to limited battery energy or computation power, these user end devices need to offload tasks to wifi routers using in a wireless way. Then the wifi routers will calculate the optimal computing strategy by searching DP-table and offloading the tasks to specific workers. Specifically, the wifi router itself can also be user equipment because if the wifi router-workers cluster can not handle these tasks given certain conditions(e.g. energy or bandwidth constraints), it will also transfer the tasks to other wifi router clusters.

3.1.4 Workers layer

Worker acts as edge computing nodes in the system, similar to Base Station in the MEC system. Workers need to register themselves to the wifi router in one cluster by heart beating. Also, workers send Ping periodically to wifi routers with information, including bandwidth(e.g. the link between the worker and the wifi router) and energy cost of that specific device. In our system, the workers can be servers, laptops, or any other computing nodes.

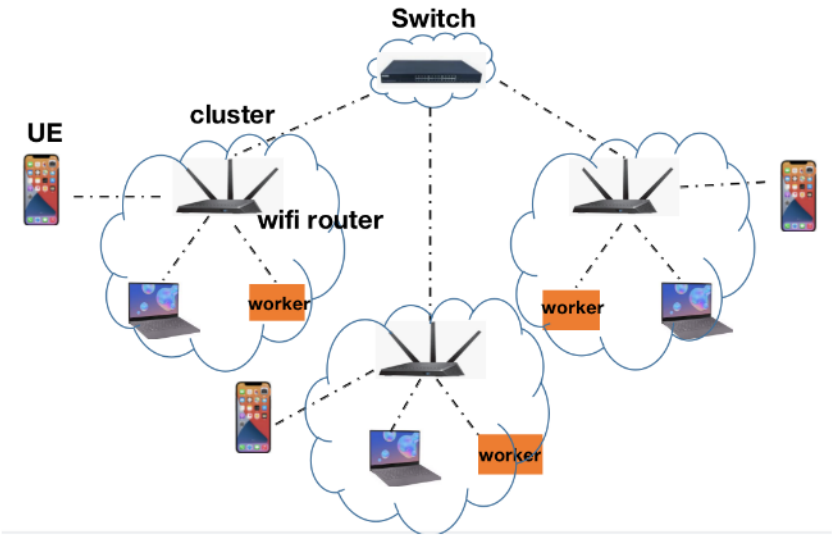


Figure 1: system architecture

4 Mathematical Model

In our model, a user request occurs inside a cluster. This request should first be processed inside the cluster under certain energy constraints. If the computing resources inside the cluster are insufficient, or the energy constraint exceeds the minimum limit, then it is necessary to consider transferring tasks across clusters. According to the optimization principle of dynamic programming, a multi-stage decision-making problem, assuming that the first stage of its optimal decision-making function is q_1, and the system shifts to a new state x_2, then all the following strategies are sub-problems based on the initial state x_2 and must constitute an optimal strategy. The sub-problems of our model in each cluster are the same problems as the original one, but the scale of the problem has been reduced, which means that our model can be solved by dynamic programming.

From an algorithm perspective, another factor that (discrete variable) dynamic programming(DP) relies on is the Overlapping Principle of sub-problems: the solution of a problem can be divided into the solution of several sub-problems, and the calculations to deal with these sub-problems are partially overlapped. The dynamic programming algorithm uses the Overlapping Principle of the sub-problems of the problem to design the algorithm, which can save a lot of calculations. For some problems that seem unlikely to be solved quickly, DP works like polynomial-time algorithms, which is called an efficient algorithm.

Based on the system model above, we have t tasks in the task set $T = \{T_1, T_2, T_3.....T_t\}$, and in order to simplify our model, we use bytes to represent our task size. For example, T1=20 GBytes. In the WAN environment, suppose we have n clusters(wifi routers), which can be represented as $C = \{C_1, C_2, C_3.....C_n\}$. In each of the clusters, we have W workers written in $W = \{w_1, w_2, w_3.....w_n\}$ and U users equipment devices written as $U = \{U_1, U_2, U_3.....U_n\}$ respectively. We assume that all the tasks can not be divided, and all the assigned workers of a specific task execute exactly the same size of the task in order to maximize available bandwidth with the energy constraint.

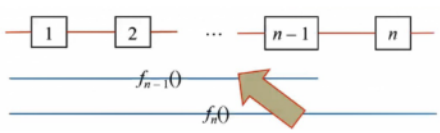


Figure 2: forward optimization

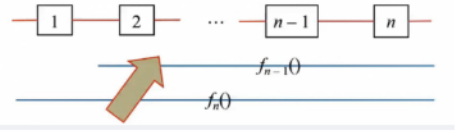


Figure 3: backward optimization

Constructing dynamic programming recursive equations, there are forward optimization Fig.2 and backward optimization Fig.3. For example, the problem has n stages, define $f_k()$ as the optimal solution value of the first k stages (from the first stage to the k-th stage) of the problem, and then recursively $f_k()$ to $f_{k-1}()$. Finally, we can get termination condition $f_n()$ expression. If we define $f_k()$ as last k stages (from n-k+1 stage to n-th stage), it can be deduced to $f_1()$.

- Phase: k = 1,2,3.....n , k represent n phase, which means in each phase we assign workers in specific cluster
- State Variable E_k: means available energy units consumption(less than the maximum Energy-constraint units) in each phase k
- Decision Variable X_k: means assigning how much energy units consumption in the phase k, and $0 \leq X_k \leq E_k$

- State transfer equation: $E_{k+1} = E_k$ - X_k
- Dynamic basic equation:
  $f_k(E_k) = \max[g(E_k,X_k) + f_{k+1}(E_{k+1})]$

  $f_n(E_n) = 0$

$f_k(E_k)$ means in k-th phase, maximum available bandwidth given the available energy units consumption E_k. g(E_k,X_k) means given available energy units consumption E_k, assign X_k in the k phase and get immediate bandwidth. $f_{k+1}(E_{k+1})$ means in k+1-th phase, maximum available bandwidth given the available energy units consumption E_{k+1}.

5 Numerical Example of DP

Table 1: Bandwidth and Energy distribution in cluster

DP-table	**cluster1**	**cluster1**	**cluster2**	**cluster2**	**cluster3**	**cluster3**
allocation	BW	Energy	BW	Energy	BW	Energy
0	0	0	0	0	0	0
1	0.5	100	0.65	80	0.6	95
2	0.6	110	0.7	85	0.7	100
3	0.75	125	0.8	90	0.8	110

For a better understanding of the proposed DP algorithm of computing strategy, we have taken an example as depicted in Table.1. represents Bandwidth and Energy cost when allocating different numbers of workers in each wifi router cluster. Allocation a certain number of workers from a specific cluster for computational purposes means how many workers in a specific cluster can be assigned and how much overall bandwidth can be provided (using that many workers) with an equivalent Energy(cost) units consumption. And we will use backward optimization to solve this problem.

For example, there is a task size of 20 GBytes under energy constraints of 300 units. Let's use the proposed model in the last chapter and draw several DP tables in every phase as follows.

From the Table.2. The minimum energy cost is 0 because we can choose not to assign any worker in this cluster. Because phase 3 is our first step,

Table 2: k=3, DP-table

Energy cost Units	x_3	$f_3^*(S_3)$	X_3^*
0-94	0	0	0
95-99	1	0.6	1
100-109	2	0.7	2
110-300	3	0.8	3

under the energy consumption constraint(300 units), our maximum energy cost will be 110 units. Therefore, our maximum available bandwidth is 0.8 MBytes/sec and 3 workers in cluster 3. For example, in phase 3, if we only have 99 energy units cost, we can get the optimal value of 1 worker and get 0.6 MBytes/seconds bandwidth in this cluster. $f_3(S_3)$ = max g(S_3,X_3), 0 $\leq$$X_3$ $\leq$$S_3$ = 4. $f_3^*(S_3)$ = maxX_3 = 0.8. X_3 = 3. As a result, given 300 energy units constraint, our optimal strategy is to assign 3 workers from cluster3 and get maximum 0.8 MBytes/seconds bandwidth. The worst case time complexity for this first step will be O(n), where n is the number of workers in this cluster.

Table 3: k=2, DP-table

Energy cost Units	x_2 **0**	x_2 **1**	x_2 **2**	x_2 **3**	$f_2^*(S_2)$	X_2^*
0-79	0	-	-	-	0	0
80-84	0	0.65	-	-	0.65	1
85-89	0	0.65	0.7	-	0.7	2
90-94	0	0.65	0.7	0.8	0.8	3
95-99	0.6	0.65	0.7	0.8	0.8	3
100-109	0.7	0.65	0.7	0.8	0.8	3
110-174	0.8	0.65	0.7	0.8	0.8	0
175-179	0.8	1.25	0.7	0.8	1.25	1
180-184	0.8	1.35	1.3	0.8	1.35	1
185-189	0.8	1.35	1.4	1.4	1.4	2
190-194	0.8	1.45	1.4	1.5	1.5	3
195-199	0.8	1.45	1.5	1.5	1.5	2
200-300	0.8	1.45	1.5	1.6	1.6	3

The Table.3 can be obtained based on Table.2. When there is no worker

assigned from both cluster2 and cluster3, there will be no Energy cost and bandwidth in the table. If we have 1 worker from cluster2 and no worker from cluster3, we will get 0.65 MBytes/seconds bandwidth by using 80 Energy units Cost. And if we have 0 workers from cluster2 and 1 worker from cluster3, from the Table.2, we can get 0.7 MBytes/seconds bandwidth by using 95 Energy units cost. Given Energy constraint, we can assign workers in cluster2 and use the rest available energy cost to search for Table.2 to fill in the Table.3.

In Table.3, the maximum energy cost will be 300 units and get a bandwidth of 1.6 MBytes/seconds (3 workers from cluster2, 3 workers from cluster3). For example, if we have an energy constraint of 178 units, from Table.3, we can choose 1 worker of cluster 2 the energy will be 80 units, and the bandwidth will be 0.65 MBytes/seconds, then there will be 178-80=98 energy units available. Then we search Table.2 for maximum bandwidth under the limited energy cost of 98 units. We can assign a maximum 1 worker in cluster 3 without exceeding the energy constraint and the total maximum bandwidth is 0.6+0.65=1.25 MBytes/seconds as the optimal value shown in Table.3. Besides, using the same Energy constraint of 178 units, if we choose X_2=0, the maximum bandwidth will be 0.8 MBytes/seconds by assigning 3 workers all from cluster3 under the available energy consumption of 178 units. And if we choose X_2=2 or X_2=3, available energy will be 178-85=93 units or 178-90=88 units, all of which lower than one worker energy cost in cluster3, so no workers will be assigned in k=3 phase and get maximum bandwidth 0.7 or 0.8 MBytes/seconds respectively. As a result, as X_2 from 0 to 3, we get {0.8, 1.25, 0.7, 0.8} respectively and the optimal computing strategy should be X_2=1 and $X_3 = 1$ with the energy constraint of 178 units. Finally, in the phase 2: $f_2(S_2) = \max\ [g(S_2,X_2)+f_3(S_3)]$, $0 \leq X_2 \leq S_2 = 4$. $f_2^*(S_2) = 1.6$ MBytes/seconds. $X_3 = 3$. As a result, the optimal strategy of this second step(with 300 units Energy Constraint) is to assign 3 workers from cluster2 and cluster3 respectively and get 1.6 bandwidth. Considering the worst case time complexity of this second step, it will be $O(n^2)$.

From Table.4, under the energy constraint of 300 units, we will use one worker from cluster1 to get 0.5 MBytes/seconds, and there are 300-100=200 energy units available. Then after search Table.3, we use 3 workers from cluster2, get bandwidth of 0.5+0.8=1.3 MBytes/seconds and there are 200-90=110 energy units available. Finally we will use 3 workers from cluster3. The maximum of bandwidth is 1.3+0.8=2.1 MBytes/seconds, so that the minimum task transmission time will be 20 GBytes / 2.1(MBytes/seconds) = 9.752 * 10^3 seconds. $f_1(S_1) = \max\ [g(S_1,X_1)+f_1(S_1)]$, $0 \leq X_1 \leq S_1 = 4$. $f_1^*(S_1) = 2.1$ MBytes/seconds. $X_1 = 1$. And the optimal assignment strategy

will be as following:

- X_1=1, X_2=3, X_3=3;

And for the final step, the worst-case time complexity should be $O(n^3)$, where n is the number of workers in each cluster.

For the final optimal strategy, we get a sum of 2.1 MBytes/seconds bandwidth from all the clusters. If the user request(task) is divisible, we could achieve a maximum bandwidth from all the clusters transferring together under certain energy constraints. Although the bandwidth might change a little bit when computation prevails, the DP algorithm can help get an optimal strategy in a transient state or when the fluctuation of bandwidth is small.

6 Performance Evaluation

In order to evaluate the performance of the proposed DP algorithm, a self-developed discrete event python-based simulator PYEng-Sim is used. This allows us to set the nodes numbers, network bandwidth, the Energy cost to generate forward and backward Dynamic Programming tables. We compare the DP algorithm against three normal and widely used traditional scheduling algorithms: Random, RoundRobin and Local-first in Fig.4. Random is that I collect all the edge nodes together no matter which wifi router they belong to, and randomly choose certain number of devices. This method can guarantee all the edge nodes share exactly the same probability to be chosen. Local-first is to use the edge computing nodes in the local cluster, which might result in overload in the specific cluster because of the Hot-Spot-problem. RoundRobin is an algorithm that firstly the number of wanted edge nodes divided by wifi router(cluster) number, and then when assigning a specific number of wanted nodes into a certain cluster, a random strategy will be used to select nodes within the cluster. Therefore, the RoundRobin algorithm can not guarantee equal probability among clusters, but it can ensure computing nodes within a cluster can be equally chosen.

6.1 Simulation Setup

The proposed DP algorithm is evaluated and is compared with Random, RoundRobin and Local-first in our simulation. We will use 6 wifi router clusters in the system. And in each of the clusters, we have 15 edge nodes respectively. Besides, in each of the clusters, we will use the Random Library of Numpy to increasingly set the bandwidth and Energy when selecting more nodes together. For 1-nodes, 1.2 MBytes/seconds, 200 Energy

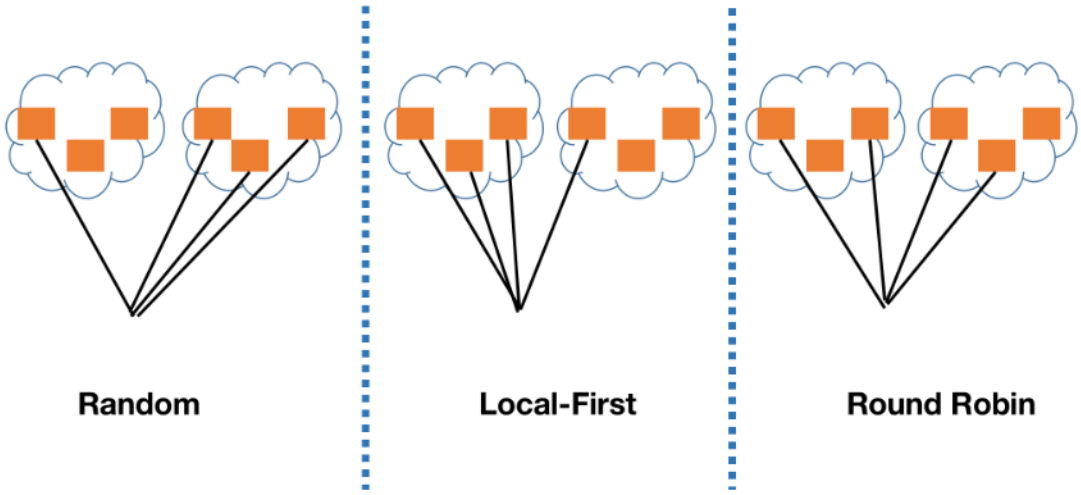


Figure 4: Three traditional scheduling Graph

units cost; 2-nodes, 1.5 MBytes/seconds, 300 Energy units cost etc. Finally, we have a task 20 GBytes with a different Energy Constraint set EC={50,100,150,...1100,1150,1200} units, which are from 50 to 1200 units with s 50-step units size. Taking the above-mentioned performance matrix, the following simulation results are derived.

6.2 Simulation Results

The proposed algorithm is evaluated by comparing its performance with Random, Local-first, RoundRobin algorithms. In the experiment, we set fixed bandwidth to see how many workers and how much Energy cost need among these four algorithms. Besides, another test is that given fixed Energy constraints(which is adhere to our topic in the report), we will see how many workers need and how much bandwidth we can get from the above four algorithms.

From the Figure.5 and Figure.6, we can see that given a certain bandwidth, Dynamic Programming always needs the least number of workers compared with another three algorithms, which means DP is better at maximizing bandwidth with fewer workers installed. And with the fixed bandwidth, DP needs less Energy cost compared with others, which is important and suitable for our report's topic and be better at saving energy.

From the Figure.7 and Figure.8, we can see that with the given Energy constraint, DP needs similar numbers of workers with RoundRobin and

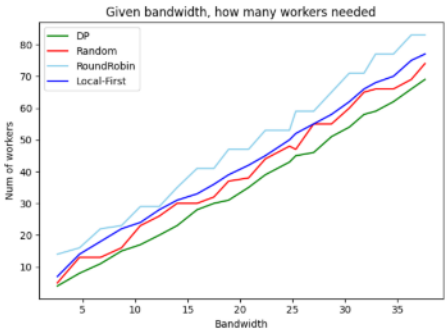


Figure 5: With similar bandwidth, how many workers needed

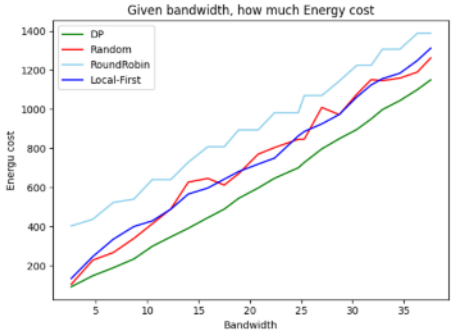


Figure 6: With similar bandwidth, how much Energy cost

Local-first algorithms. However, if considered bandwidth, DP can always get the highest bandwidth than others by giving a specific Energy constraint, which can also prove that our proposed DP method performs better than traditional computing strategy algorithms.

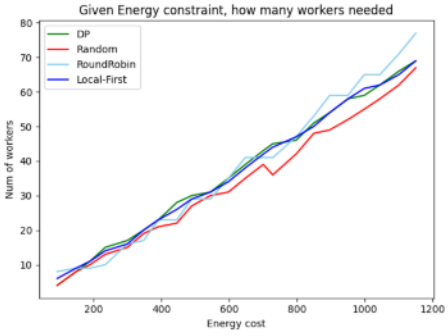


Figure 7: With similar Energy constraint, how many workers needed

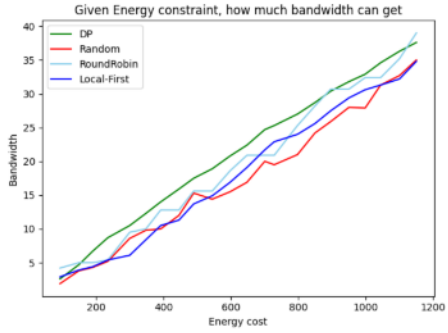


Figure 8: With similar Energy constraint, how much bandwidth can get

This Table.5 can show more direct cooperation among the algorithms. We use DP algorithm results as standard to compare all the others methods. With the very similar Energy constraint and bandwidth, among which DP can get higher bandwidth using less Energy cost, DP methods nearly always use the least number of workers. The table can prove that DP has a better prospect to research.

6.3 Time complexity and Reusability

In this part, we set there are six wifi router clusters as mentioned above, but a different number of workers in each cluster, from 10 to 30 workers. Using backward optimization as the solution of DP algorithm, in the phase k(the kth wifi router), if we do not save the last phase's table the always calculate the table every time, the time complexity of phase k will be T(k) = $(N^k + N^{k-1} + ... + N)$*$N^{k-1}$*N, in which N is the number of workers in kth wifi router. The time can be seen in Fig.9. However, because of the Calculation reusability of DP, we can save the table of every phase and when we need to calculate table-K, we just easily search for table-K-1, which has been saved in advance. Therefore the time complexity by using Calculation reusability is N^k*N^{k-1}*N. The result is shown in Fig.10, lots of time have been saved.

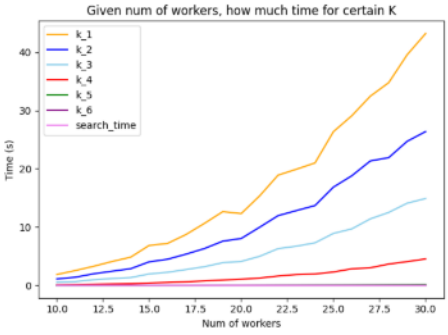


Figure 9: Given num of workers, how much time for certain K

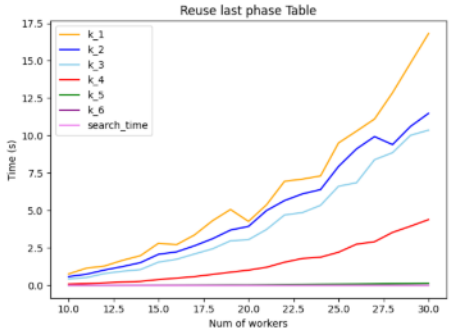


Figure 10: Reuse last phase Table

6.4 Task transmission simulation

In this part, we fix the energy budget and search for resources to complete the load within that budget and hence has an implication of time. We use the same experiments settings with the first part, using a data size of 50GBytes, and the energy constraints are also from 100 to 1200 units. The Figure.11 shows that given a certain workload if energy constraint increases, there will be less time spent. And among all the algorithms, DP always achieves less time spent performance, especially in a lower Energy Constraint, proving that the DP algorithm works well in Energy-aware computing strategy.

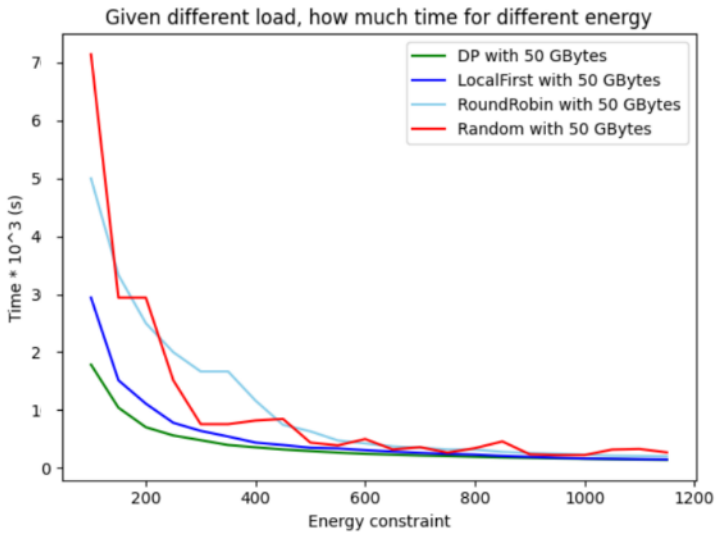


Figure 11: Given different load, how much time for different energy

7 Conclusion

In this report, we deal with an In-house wifi router-based computing strategy for large-scale workloads and have proposed an efficient DP method to maximize bandwidth under the Energy Constraint. We collect bandwidth and Energy cost from every cluster into a table. In every phase, we assign an energy quota and get the current best choice to fill in the DP table. Our method is generic and can replace bandwidth with other factors such as CPU load, Memory size ,etc with certain constraints. Our proposed DP method is simulated extensively to compare our result with similar algorithms solving the same problems and the superiority of our method over others is clearly demonstrated. Besides, by simultaneously using forward and backward optimization, we can achieve Calculation reusability and save much time-consuming. And giving specific workload, the DP method uses less time to translate tasks than the other three algorithms. However, with the purpose of improving the method and making it more robust, we will strive to use the DP mechanism in real-world In-house wifi router environment, which will be the future of this work.

Acknowledgement

I thank Mr. Gokul MC for his collaborative work conducted in the Chapter3 System Model and his comments on report reading and literature survey

works. I also thank Prof.Bharadwaj Veeravalli for his suggestions and guidance on algorithm design and simulation part.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[2] Rashmi Sharma, Nitin, "Performance Evaluation of New Joint EDF-RM Scheduling Algorithm for Real Time Distributed System", Journal of Engineering, vol. 2014, Article ID 485361, 13 pages, 2014.

[3] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource Management with Deep Reinforcement Learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets '16). Association for Computing Machinery, New York, NY, USA, 50–56.

[4] Peng, Yanghua & Bao, Yixin & Chen, Yangrui & wu, Chuan & Meng, Chen & Lin, Wei. (2019). DL2: A Deep Learning-driven Scheduler for Deep Learning Clusters.

[5] Y. Bao, Y. Peng and C. Wu, "Deep Learning-based Job Placement in Distributed Machine Learning Clusters," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, 2019, pp. 505-513

[6] L. Yazdanov and C. Fetzer, "VScaler: Autonomic Virtual Machine Scaling," 2013 IEEE Sixth International Conference on Cloud Computing, 2013, pp. 212-219

[7] Rao, Jia & Bu, Xiangping & Xu, Cheng-Zhong & Wang, Le & Yin, Gang. (2009). VCONF: A reinforcement learning approach to virtual machines auto-configuration. Proceedings of the 6th International Conference on Autonomic Computing, ICAC'09. 137-146.

[8] P. K. Sahoo, C. K. Dehury and B. Veeravalli, "LVRM: On the Design of Efficient Link Based Virtual Resource Management Algorithm for Cloud Platforms," in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 4, pp. 887-900, 1 April 2018

[9] Y. Sun, S. Zhou and J. Xu, "EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks," in IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2637-2646, Nov. 2017

[10] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322-2358, Fourthquarter 2017

[11] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo and D. O. Wu, "Energy-Optimal Mobile Cloud Computing under Stochastic Wireless Channel," in IEEE Transactions on Wireless Communications, vol. 12, no. 9, pp. 4569-4581, September 2013

[12] J. Liu, Y. Mao, J. Zhang and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," 2016 IEEE International Symposium on Information Theory (ISIT), 2016, pp. 1451-1455

[13] C. You, K. Huang, H. Chae and B. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," in IEEE Transactions on Wireless Communications, vol. 16, no. 3, pp. 1397-1411, March 2017

[14] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," in IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 2795-2808, October 2016

[15] Y. Tao, C. You, P. Zhang and K. Huang, "Stochastic Control of Computation Offloading to a Helper With a Dynamically Loaded CPU," in IEEE Transactions on Wireless Communications, vol. 18, no. 2, pp. 1247-1262, Feb. 2019

[16] Li, Shilin Tao, Yunzheng Qin, Xiaoqi Liu, Long Zhang, Zhi Zhang, Ping. (2019). Energy-Aware Mobile Edge Computation Offloading for IoT Over Heterogenous Networks

Table 4: k=1, DP-table

Energy cost Units	x_1 **0**	x_1 **1**	x_1 **2**	x_1 **3**	$f_1^*(S_1)$	X_1^*
0-79	0	-	-	-	0	0
80-84	0.65	-	-	-	0.65	0
85-89	0.7	-	-	-	0.7	0
90-94	0.8	-	-	-	0.8	0
95-99	0.8	-	-	-	0.8	0
100-109	0.8	0.5	-	-	0.8	0
110-124	0.8	0.5	0.6	-	0.8	0
125-174	0.8	0.5	0.6	0.75	0.8	0
175-179	1.25	0.5	0.6	0.75	1.25	0
180-184	1.35	1.15	0.6	0.75	1.35	0
185-189	1.4	1.2	0.6	0.75	1.4	0
190-194	1.5	1.3	1.25	0.75	1.5	0
195-199	1.5	1.3	1.3	0.75	1.5	0
200-204	1.6	1.3	1.4	0.75	1.6	0
205-209	1.6	1.3	1.4	1.4	1.6	0
210-214	1.6	1.3	1.4	1.45	1.6	0
215-219	1.6	1.3	1.4	1.55	1.6	0
220-274	1.6	1.3	1.4	1.55	1.6	0
275-279	1.6	1.75	1.4	1.55	1.75	1
280-284	1.6	1.85	1.4	1.55	1.6	1
285-289	1.6	1.9	1.85	1.55	1.9	1
290-294	1.6	2	1.95	1.55	2	1
295-299	1.6	2	2	1.55	2	1
300	1.6	2.1	2.1	2	2.1	1

Table 5: With similar Energy constraint and bandwidth, how many workers needed

Energy cost	**Bandwidth**	**DP**	**Random**	**RoundRobin**	**Local-first**
95	3.0	5	6	7	5
142	5.4	9	9	13	9
191	6.6	12	12	14	14
248	9.3	15	17	19	18
297	10.5	18	18	21	20
337	12.2	19	20	24	23
389	14.3	23	24	27	28
449	15.7	25	26	29	31
495	18.2	29	31	33	33
549	19.6	32	33	35	35
598	21.1	35	36	38	38
648	22.3	38	40	39	42
700	24.3	42	43	45	45
749	25.7	45	46	47	47
800	27.1	48	49	50	49
850	28.6	51	50	51	53
899	29.9	54	55	53	56
943	31.5	55	56	57	59
999	33.3	59	60	58	62
1049	34.7	60	62	62	63
1098	36.0	64	65	64	66
1147	37.6	67	67	66	69