515131 Introduction to Algorithms

Final Project: AI Agents Deployment Optimization

Deadline: 2024/12/31 23:59

1. Description

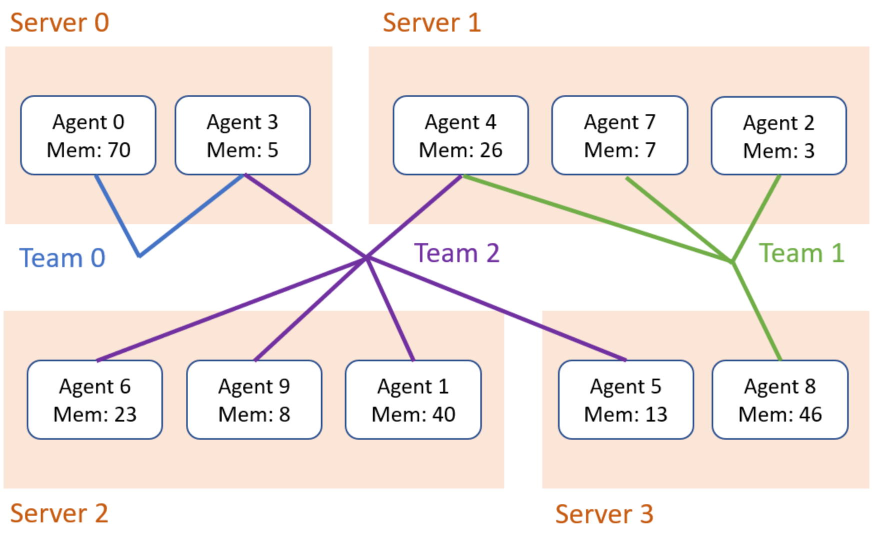
Your task is to deploy multiple AI agents onto a set of servers. Each AI agent requires a certain amount of memory space to store its model and associated data. Additionally, AI agents often work in collaborative teams, requiring frequent data exchanges to perform their tasks. Your goal is to assign AI agents to servers such that:

1. The total memory usage by AI agents on a server does not exceed its disk capacity.
2. The number of servers spanned by a single collaborative team is minimized to reduce inter-server communication costs and improve efficiency.

Given the maximum disk capacity, the memory required by each AI agent, and the connections between agents in the same team, you have to implement algorithms of **2-way (basic)** and **k-way (advanced) partitioning** for model deployment. A 2-way partition divides a given set into two subsets based on certain conditions, whereas a k-way partition separates the set into k subsets, where k ≥ 2. The cost for splitting a collaborative team across multiple servers is calculated as:

$$C(i) = ((\text{the number of servers that team } i \text{ spans}) - 1)^2$$

The total cost is the sum of $C(i)$ for all collaborative teams. Take the following figure for example. Team 0 spans only one server. $C(0) = (1 - 1)^2 = 0$. Team 1 spans two servers. $C(1) = (2 - 1)^2 = 1$. Team 2 spans four servers. $C(2) = (4 - 1)^2 = 9$. Hence, the total cost of this example is $C(0) + C(1) + C(2) = 10$.



2. Input Format

The input begins with a single line indicating the maximum disk capacity of a server. Starting from the second line, the data is divided into two sections, which are marked by the keywords, "**.agent**" and "**.team**". They describe the memory requirements of each AI agent and the network connections between agents respectively.

The line next to the keyword ".agent" specifies the total number of agents, denoted as **n**. For the following n lines, the ith line indicates the memory needed by the agent i. Here, line numbering starts from 0, and agent IDs range from 0 to n-1.

The line next to the keyword ".team" specifies the total number of teams, denoted as **m**. The following 2m lines describe the network. The (2i)th line indicates the number of agents in team i, and the (2i+1)th line lists the IDs of these agents, separated by a space. Here, i ranges from 0 to m-1, and line numbering starts from 0.

```
45          // Maximum disk capacity
.agent
6           // Total number of agents, n = 6
10          // Agent 0 requires 10 units of memory
20          // Agent 1 requires 20 units of memory
15
25
25
10
.team
3           // Total number of teams, m = 3
3           // Team 0 contains 3 agents
0 1 2       // Agents 0, 1, 2 are in Team 0
4           // Team 1 contains 4 agents
2 3 4 5     // Agents 2, 3, 4, 5 are in Team 1
2
4 5
```

3. Output Format

The output file describes the partitioning result:

- The first line represents the **total cost** of your deployment.
- The second line indicates the number of servers you used, denoted as **k**. For basic testcases, k should be 2. For advanced testcases, k can be ≥ 2.

- For the following **n** lines, the ith line indicates the server ID to which agent i belong. The server ID is ranging from 0 to k-1.

```
2       // the total cost = (2-1)² + (2-1)² + (1-1)² = 2
3       // the total number of servers = 3
0       // Agent 0 is deployed on server 0.
0       // Agent 1 is deployed on server 0.
1       // Agent 2 is deployed on server 1.
1
2
2
```

4. Requirements

This assignment should be programmed in C/C++ within c++11 standard. You have to provide a makefile to compile your program. Prepare your codes in the following structure.

```
Algorithm_FP_2024/
├── src/                // Put your source codes here
│   ├── file_1.cpp
│   └── file_2.cpp
├── Makefile
└── ap
```

TA will use the following commands to execute your program

```
>> cd Algorithm_FP_2024
>> make
>> ./ap <input_file> <output_file>
   (E.g., ./ap testcases/basic/sample_in.txt sample_out.txt)
```

Please compress the directory into a single tar file by the following command. The {StudentID} should be your student ID.

```
>> tar zcvf Alg_FP_{StudentID}.tar.gz Algorithm_FP_2024
```

Additionally, please note that your program should decide to run 2-way or to run k-way partitioning by the given memory constraint, rather than the input filename. Also, your program should be single-threaded.

5. Grading

This assignment will be ranked and scored according to the performance of your program. TA will evaluate your results on a Linux system. The grading breakdown is as follows. There will be 10 testcases.

- Basic testcases (50%, 10% x 5)
- Advanced testcases (50%, 10% x 5)

Each testcase is evaluated on two key metrics:

1. Validity (70%): The result produced by your program is correct and meets the requirements. It will also be considered invalid if the runtime exceeds **2 hours**.
2. Performance (30%): Performance grading is based on the total cost of your result. The rankings are divided into seven tiers, each evenly distributed.

Rank Tier	Score
Tier 7	30 pts
Tier 6	25 pts
Tier 5	20 pts
Tier 4	15 pts
Tier 3	10 pts
Tier 2	5 pts
Tier 1	0 pts

6. E3 Submission

1. Submit your files to the E3 system.
   - Alg_FP_{StudentID}.tar.gz
2. Follow the structure illustrated in Section 4, ensuring that unnecessary files, such as test data, are excluded.
3. Remember the submission rules mentioned above, or you will be penalized by **-30 points** on your grade.

7. Policy

1. The upload deadline would be at **23:59 on December 31, 2024.** We DO NOT accept any late submission.
2. **Plagiarism: -100 points.**

8. Problems

1. If you have any problem with this project, please post it on the E3 forum.

9. Reference

1. Graph partition - Wikipedia
2. The ISPD98 circuit benchmark suite