Design and Implementation of the Enterprise Cluster Manager

Chen Chen, Ming Cheng, Yang Yang
China Automotive Technology and Research Center Co., Ltd., 68, East Xianfeng Road, Dongli District, Tianjin, China
yangyang@adcsoft.cn

Abstract. A practical cluster manager is designed to fulfill the management requirements of server resources allocation to multi-UE processes and optimize the security, storage and retrieval of big data. Based on the application background of cluster manager, it introduces the structural designs, including frame, database, member node, master node and internal workflow. Meanwhile, it expounds the optimizing processes of cluster manager, including load balance algorithms, emergency processing, partition store of big data and others. Finally, the feasibility of whole schedules is verified by the test data of practical application, providing a reference for the subsequent research on cluster management.

1. Introduction
Cluster manager is mainly applied for the intelligent network auto competition training platform system. This project is aiming to investigate the theory of new energy vehicles and intelligent connected vehicles.

Based on this project demands, the system architecture is simply designed to client side and server side. The client side contains a competition website and a practical training client software. The users would answer the questions on competition website and practical training client and expect results about driving simulation processes including lots of continuing image data and scores from server side. Those driving simulation processes is generated from Unreal Engine (UE) tasks, which need enough computing resources to deal with images. Then the system would operate UE process after receiving the request for grading a question, provide environment for users to simulate driving and give related scores based on users’ answer status.

Enough computing resources are essential for UE process, while a single server cannot meet the requirements of whole practical system due to limited processing capacity. A cluster management schedule is designed to apply in server and reasonable distribute multiply independent computing resources, so as to solve huge UE computing requirements. From the aspect of specific application of this project, the cluster manager can be taken as an entirety to work as a server side in the system.

2. Basic concept of cluster manager
2.1. Definition about cluster
Cluster, as the definition of cluster technical expert Gregory F. Pfisher[1], is a parallel or distributed system consisting of connected computer structures. Looked from its appearance, it is just a single system to provide unified service. In other words, several computers are managed uniformly by cluster
manager and taken as a functional model to provide services. Hence, it can improve the computing capacity of model, and increase fault tolerance and high concurrency [2].

2.2. Structural design of cluster manager
In this cluster manager, according to specific functions, the connected computers can be classified to cluster master node, cluster member node and managing database. The functional structures of whole cluster management system are shown in the figure 1.

![System Structure: Client Side and Cluster Manager](image)

**Figure 1.** System Structure: Client Side and Cluster Manager
Cluster managing node, or master node, is consisting of a single computer. It is not responsible for the operation of UE process, but only for those lightweight tasks to act as a go-between for client side and undertaking node. It will obtain all load statuses of current nodes from managing database and equally distribute those scheduling tasks to the idliest undertaking nodes which are compared by weighting algorithm.

Cluster undertaking node, or member node, would be responsible for weighted tasks which are operating UE processes through multiply separated computers. Huge computing tasks would be completed in a shorter scheduling time because of the same services, thus ensuring the high availability of whole system. When troubles happen, the tasks of the undertaking node would be transferred to another node, ensuring the high reliability of whole system.

Managing database is used to store current status of member node and status of task which would be scheduled by cluster manager.

3. Design of database
SQLServer is selected to be the operating environment of management database.

3.1. Database table structure
The cluster manager is mainly responsible for distributing tasks to all undertaking nodes. The information of member node and task status should be recorded and respectively stored in SimNodes table and SimTask table. Among them, SimNodes table would store the information and current status of member node, while SimTask table will be used to store all tasks which would be or already be scheduled by cluster manager. The structures of two tables are designed in Table 1 and Table 2.

**Table 1. SimNodes Table**

| NAME           | DATA TYPE    | Explanation                                      |
|----------------|--------------|--------------------------------------------------|
| Name           | nvarchar(50) | Name of member node                              |
| IP             | nvarchar(50) | IP of member node                                |
| NODE_STATUS    | nvarchar(50) | Status of member node: 1 means start up, 0 means closure |
| UPDATETIME     | datetime     | Update time of node status                       |
| LOCAL_IP       | nvarchar(50) | Local IP of member node                          |
 USAGE_CPU  int  Current CPU utilization of member node
 USAGE_GPU  int  Current GPU utilization of member node
 USAGE_MEMORY  int  Current memory utilization of member node

Table 2. SimTask Table

| NAME              | DATA TYPE     | Explanation                                      |
|-------------------|---------------|--------------------------------------------------|
| ID                | bigint        | Total ids of Task                                |
| CREATETIME        | datetime      | Creation time of Task                            |
| UPDATETIME        | datetime      | Update time of Task status                       |
| NODE_NAME         | nvarchar(50)  | Corresponding name of member node                |
| LOCAL_ID          | int           | Task ID in the member node                       |
| LOCAL_PROCESS_ID  | int           | Local process id of member node                  |
| T_STATUS          | int           | Task Current completing status                   |
| CREATE_PROGESS    | float         | Process creation progress(Percentage)            |
| MEMO              | nvarchar(500) | Note of process status                           |
| XML               | xml           | Contents of XML transaction package              |

3.2. Database partition optimization

As a physical database design technology, database partition is designed to reduce the total amount of data read and write in specific SQL operation, thereby reducing response time. There are common partitions, namely Horizontal Partitioning and Vertical Partitioning. Horizontal Partitioning is to divide table rows according to the value of one attribute column. In results, all defined column were included in data sets without change of original table. Vertical partitioning means to partition the special columns to other parts and thus reduce the width of table.

This cluster manager provides solutions for storage and search problems of massive data by making partition design. In this database design, all tasks distributed by cluster managers are recorded in SimTask table, including completed tasks and uncompleted tasks. However, the search and distribution of uncompleted tasks are involved in the operation of cluster manager. Completed tasks would not be added in distributed process although recorded in database.

Table 3. T_STATUS Value Set

| VALUE | Meanings    |
|-------|-------------|
| 0     | creating    |
| 1     | Running     |
| 2     | Completing  |
| 3     | External ending |
| 4     | False       |

Figure 2. Partitioning based on T_STATUS Value

Current statuses of tasks are recorded by T_STATUS in SimTask table. The possible values and meanings are listed in Table 3. Tasks under creation and operation would be regarded as uncompleted, with T_STATUS value of 0 and 1. Those completed, externally terminated and failed are seen as completed, with corresponding T_STATUS values of 2, 3 and 4. Meanwhile, through horizontal
partitioning schedule, completed and uncompleted tasks can be respectively stored in two files by cluster manager, and thus promoting the retrieval performance of database which is shown in Figure 2.

4. Design of master node
Load balance and node status monitoring are two main functions of cluster master node. The relative procedures of master node should be started and operated for a long time in the computer, so Windows Service emerges as the reality requires. Furthermore, monitoring windows are provided for the node status in the form of exe by cluster manager.

4.1. Load balance algorithm
The master node should distribute tasks equally to different member nodes, avoiding the long-term idle conditions or overload of one member node. This method is named as load balance algorithm[3]. Common load balancing algorithms can be divided into two categories: non-adaptive and adaptive. For non-adaptive load balance algorithm, there is no need to obtain information from member node. Instead, the member node is selected by the meaning of random or round-robin method to handle with current tasks [4]. For adaptive load balance algorithm, the member node is obtained in accordance with its load change status to handle with current task. Common adaptive load balance algorithms include the least load algorithm and the next minimum load algorithm. The minimum load algorithm is used in this design of cluster manager. Its main idea is to query the load status of all nodes and select one with the lowest load values to handle with tasks. Load value records the value to describe the load status of node, and it is calculated by weighting several load factors that affect the performance of member nodes. Memory utilization, CPU utilization and GPU utilization are three factors to be taken into consideration for cluster manager. There are high computing requirements for graphic processing when the member node handles with UE process, so that the weight of GPU utilization should be set as high as 0.5. However, the weights of CPU utilization and memory utilization are respectively set to 0.3 and 0.2. The final load value is the weighted sum. The specific realization of load balance algorithm is achieved as Figure 3.

4.2. Node status monitoring window
The monitoring window is provided in the computer where the master node is operated by the cluster manager, which will be convenient for monitoring and operation of administrator. The administrator can install, start and uninstall master node services in this computer through this window. The administrator also has right to monitor the serving operation status of master node in real-time. Meanwhile, this window can feed back and display the current statuses of all member nodes and tasks, which is updated when the master node periodically queries the data of SimNodes table and SimTask table. It is set to query two seconds every time to fulfill the project requirements.

5. Design of member node
The cluster member node is designed to operate UE process, real-timely give feedback about status and write in database, communicate with client side and return the results, and deal with any emergency status.
Figure 4. Member Node Window Design

5.1. Member node window
The function of member node window is simple. From the window, we can see whether the undertaking is online, and update CPU utilization, memory utilization and GPU utilization in real-time. The window is designed as Figure 4.
Among them, the program of member node can make use of WMI object to achieve the information of CPU, memory and GPU. First of all, it can obtain total sizes of physical memory and GPU memory through WMI. Then, the sizes of CPU, GPU and memory are got. Finally, their percentages are respectively calculated and real-timely fed back to the window for update in the database.

5.2. Design of Member Node Security
For communication security, token is defined for authentication by cluster manager. Token is made up of a string generated by the server as a temporary identity for the client site to request. Server would generate a token and return it to the client side when receiving UE task request from client site. In this designed cluster manager, token is signed by member node and returned to client side by master node. After that, the client side firstly sends data for request to the member node. When identity is successfully verified through token, the member node would return to the relative data after receiving this token.
Token is generated from CBC (Cipher Block Chaining) of AES (Advanced Encryption Standard) symmetric encryption algorithms by the undertaking node. There are two variables of Initialization Vector (IV) and Key. Different member node would own different IV while all member nodes share the same Key. A unique IV value would be set in App.comfig file by the manager when an member node is firstly added in cluster manager. Here, the uniqueness of IV value is manual by the manager. The use of Token can relieve the transmission pressures caused by inquiring database frequently and reduce the corresponding computation for more robust of cluster manager.

5.3. Steps to create a UE process
The main function of member node is to locally create and operate UE process. The steps are designed as follows:
(1) Member node receives creation request of UE Task and adds corresponding Task record to SimTask table in database.
(2) Member node starts UE process in the local and updates the progress of process creation in the database synchronously.
(3) If process creation is successful, member node would update task status in the database to be in operation and add this process to the memory management dictionary.
(4) If it cannot update the database, member node would terminate the corresponding UE process.
(5) If process creation is fail, member node would update Task status in the database to be fail.
5.4. Emergency and corresponding Process

Member node would poll and examine operating system process, local memory status and database in turn, aiming to timely restore local data status and the data status of database. There are some emergencies and corresponding processes as follows:

1. Overtime of start process: If UE process cannot be successfully started more than ten minutes, member node would terminate the relative process, clear local dictionary and set the corresponding Task status in the database to fail.
2. False record of database: If one task can be queried in local dictionary not database, member node would delete the task record from local dictionary and terminate the task.
3. False local process: Under the conditions that tasks exist in local dictionary but there is no corresponding ID process, or tasks don’t exist in local dictionary when database is in operation, task in the database would be set to be fail or terminated.
4. Ending task in scoreboard: The last step of UE process is to provide score. When users obtain scores during UE process, member node should kill this UE process.

6. Internal logic of cluster manager

By giving an example of a general process which cluster manager receive an UE task request and distribute it to one member node, this part shows the internal logic design of cluster manager to link three main functional parts: master node, member node and database. This example would be divided into several steps which are shown as follows:

1. Send UE task requirements to master node from client site, the realization of which is called xml transaction packet.
2. Query the scores of all running member nodes from the database through master node, obtaining the information of optimal node.
3. Send transaction packet which contains task information from master node to the selected member node.
4. When the transaction packet is received by the member node, UE process would be started and a new Task record would be added in database.
5. When UE process is successfully started, update SimTask table and update time in database through member node.
6. Calculate Token, sent it from member node to master node through WCF, and then to client side.
7. Establish the communication between client side and member node depending on token and IP and local IP of member node. Meanwhile, update the status of Task during communicating period.
8. From starting up, member node will poll 3 pieces of self-check UE status. Master node would poll and update three kinds of status of member node.

7. Analysis about performance promotion

By comparing UE processes loading between a single member node and multiple member nodes in cluster status, this part analyzes the optimization function of cluster manager on whole system.

7.1. Performance test of single member node

We test the maximum loading quantity of UE process on a single member node. Without cluster manager, the task distribution can only be processed by a single server. Therefore, the performance test of a single member node can be considered as that of server without cluster manager. During the test process, multiply UE processes would be created in the same server and the relative data is recorded, including CPU utilization, GPU utilization, memory utilization, frame rate and internet speed.

| CPU | Memory | GPU |
|-----|--------|-----|
| i7  | 32G    | 15G+24G |

The test environment is shown as Table 4.
Test results can refer to the Table 5. Analyzing the CPU utilization, up to 7 processes can be loaded by the single member node. However, when there are more than 4 UE processes running on the member node, the obvious frame drop would appear on display screen, which can never meet the normal project needs. Besides, if the total number of processes is exceeded 4, GPU utilization will not increase with the decrease of frame rate and the soar of CPU utilization because of limited image processing quantity. Based on the above analysis, we can consider that there can be 4 UE processes which can run effectively at the same time on a single member node.

Table 5. Test Result on Single Node

| Number of Started Processes | CPU | Memory | GPU | Frame Rate | Internet Speed Mbps |
|-----------------------------|-----|--------|-----|------------|---------------------|
| 0                           | 2   | 20     | 6   | 80         | 0                   |
| 1                           | 30  | 21     | 18  | 70         | 11                  |
| 2                           | 50  | 23     | 31  | 50         | 22                  |
| 3                           | 65  | 24     | 37  | 36         |                     |
| 4                           | 71  | 26     | 37  | 20-40      | 40                  |
| 5                           | 85  | 27     | 37  | 20-40      | 40                  |
| 6                           | 98  | 29     | 37  | 5-20       | 42                  |
| 7                           | 100 | 30     | 37  | 5-20       | 42                  |

7.2. Performance test of multiple member nodes under the cluster manager

Three member nodes and one master node are equipped to test the performance of cluster manager. The specific test method is to send multiple UE requests to client side and examine whether the tasks can be equally distributed to each member node though cluster manager. Test results are shown in Table 6. When tasks are distributed by cluster manager, 12 UE processes can be loaded equally by three member nodes without obvious dropped frame.

Table 6. Test Result when running 11/12/13 Tasks

| Node | Num of Process | CPU | Memory | GPU | Frame Rate | Internet Speed |
|------|----------------|-----|--------|-----|------------|----------------|
| 1    | 11             | 60  | 22     | 33  | 50         | 33             |
| 2    | 58             | 24  | 40     | 20-40| 40         |
| 3    | 66             | 27  | 41     | 20-40| 40         |
| 1    | 12             | 65  | 24     | 37  | 20-40      | 40             |
| 2    | 56             | 22  | 40     | 20-40| 40         |
| 3    | 66             | 37  | 42     | 20-40| 40         |
| 1    | 13             | 65  | 25     | 37  | 20-40      | 40             |
| 2    | 74             | 27  | 40     | 7-20 | 40         |
| 3    | 67             | 33  | 41     | 20-40| 40         |

7.3. Test result analysis

The above test results show that the cluster manager has successfully expanded capacity in horizontal of multiply member nodes, greatly improving the operation performance of the server. The load balance algorithm is adopted for cluster manager to flexibly, equitably and effectively complete the task distribution. Therefore, it can perfectly meet the project needs and be widely applied in other distributed computing scenes.

8. Conclusion and prospect

This thesis introduces a set of complete and feasible cluster manager schedule, including overall architecture, model functions design, load balance algorithm design, interior logic design, security design and store optimization. Furthermore, it tests the performance of single member node and multiple member nodes under cluster management, so that the practical values of this schedule is verified to provide reliable reference examples for the application of cluster manager on other scenes and promote the development of future cluster management technique. In the cluster manager
application mentioned above, there are same share of Task resources to be allocated. In future, the direction of improvement could be to allocate resources properly in accordance with the task occupancy requirements and the status of member nodes, and thus apply in more scenes.

9. References
[1] Pfister G F 1996 IEEE Parallel & Distributed Technology: Systems & Applications, 4(3), p 12-14.
[2] Bo Zhou 2013 Web Server Load Balancing System Design and Implementation Master Thesis University of Electronic Science and Technology of China p 2
[3] Junjie Zhu 2011 Research and Implementation of the Enterprise Cluster Task Management System Master Thesis South China University of Technology p 7
[4] Linjun Chen 2012 The Design and Implementation of Load Balancing Module of Cluster System Master Thesis Harbin Institute of Technology p 4