Research on Man-Machine Service Reliability of New Generation Power System

Xia Yin¹²*, Lulu Li¹², Zhimeng LV¹², Zhen Yan¹², Hua Xu¹², Kun Zhao¹² and Kun Huang¹²³

¹NARI Group Corporation/State Grid Electric Power Research Institute, Nanjing, 211106, China
²NARI Technology Co. Ltd., Nanjing, 211106, China
³State Key Laboratory of Smart Grid Protection and Control, Nanjing, 211106, China
*Corresponding author’s e-mail: yinxia@sgepri.sgcc.com.cn

Abstract. With the rapid growth of smart grid dispatching business scale, the traditional fat client mode power system has a large degree of business coupling, which limits the power grid dispatching business. But the lightweight man-machine cloud terminal and man-machine service framework have greatly reduced the coupling degree of the grid dispatching system business. The appearance of man-machine service reduces the development cost of client, and makes the unified browsing of multi system and multi terminal equipment easier. However, how to ensure the reliability and stability of man-machine interaction service is a question worthy of discussion. How to adapt to the interaction of a lot of clients and to adapt to the requests of different clients put forward higher requirements for man-machine service. This article describes the man-machine service architecture and key technologies to ensure reliability, introduces technologies such as load balancing and thread pools that adapt to the characteristics of power grid human-computer interaction, introduces in detail the distributed job scheduling management mechanism within the man-machine service, and describes the standardized request and return result of man-machine service. Finally, through sample testing and on-site operation, demonstrate the reliability of the man-machine service.

1. Introduction
With the rapid development of Internet technology, the computing and processing capabilities of the Internet are advancing by leaps and bounds, communication costs have fallen, and bandwidth has increased. Science and technology such as cloud services, microservices, and the internet of things have emerged. Internet technology has penetrated into all walks of life, power grid dispatching has also entered the internet plus era, and smart grid[1] came into being. With the proposal of a new generation of dispatching technical support system "physical distribution, logical unity" network-wide distributed system architecture[2], the traditional fat client mode dispatching technical support system has been unable to follow the pace of current grid technology development, and the traditional wide-area browsing mode also has certain limitations in supporting local and remote undifferentiated dispatching. At the same time, the front-end business of the traditional fat client model is complicated, which is not conducive to the upgrading of the existing scheduling system. The traditional fat client system has constraints in many aspects, such as undifferentiated remote dispatching of dispatchers, product updates, mobile terminal deployment, unified functions of C/S and B/S systems, and access by
third-party front-end display manufacturers. Based on this, a lightweight human-computer interaction architecture is proposed in the new generation of scheduling technical support system[3][4], which splits the traditional human-computer client into cloud terminal and man-machine service.

In the new lightweight man-machine architecture, the man-machine business of the dispatching system is separated. The front-end dispatching system is only responsible for display and related operation response, and the logic of frequent interaction with the data platform is moved to the server to form a man-machine service. The introduction of the new architecture has brought new developments and new challenges to power grid dispatching. How to meet the interaction of a large number of clients, how to adapt to the access of different languages, different terminals, and clients of different manufacturers, and to ensure the efficient and stable operation of the dispatching technical support system are the key points that man-machine service need to consider. Summarizing these problems, man-machine service mainly faces the following challenges.

1) In the traditional fat client mode, the business of each terminal is relatively independent, and the business volume of a single workstation is not large. In the lightweight man-machine architecture, the business of several clients is all concentrated in the man-machine service. How to balance the load between each man-machine server and prevent a single server from being overloaded is one of the problems that need to be solved.

2) The separation of front-end and back-end businesses allows front-end workstations to focus only on display and operation-related businesses, the display methods are more diversified, and product iteration is more convenient. However, how to design the internal framework of human machine service to meet a large number of business requests and how to schedule them can provide powerful functions for the front-end, ensure rapid response to a large number of front-end requests, and avoid accumulation of front-end requests, which is another challenge that man-machine service needs to accept.

3) The power grid dispatching system is a large-scale integrated system that integrates multi-manufacturers and multi-language products. The emergence of man-machine service makes it unnecessary for terminals of different manufacturers and languages to implement their own set of refreshing procedures for data and coloring. Terminals of various manufacturers and languages only need to follow certain protocols and rules to easily ensure the consistency of the displayed data. How to define a rule that adapts to various terminals is also one of the mechanisms that need to be considered to ensure the reliability of man-machine service.

In response to the above problems, this article briefly introduces the man-machine service framework design, and focuses on some key technologies to ensure the reliability of man-machine services. Finally, a large number of sample tests verify the stability and reliability of man-machine service. In addition, the man-machine service framework has been running at multiple sites, supporting multiple presentations and demonstrations, and has supported the refreshing display of the station diagrams of the third-party manufacturer NARI Relays, which fully reflects the reliability and openness of man-machine service.

2. Man-machine Service Framework Design

2.1. Overall Framework

The man-machine service framework adopts a layered design, which is mainly divided into client, access layer and business layer. The access layer handles terminal protocol differences and realizes network load balancing, while the business layer performs specific business processing. As shown below.
In the lightweight human-computer interaction architecture, the real-time or historical data processing logic, equipment coloring logic, automatic generation of electrical bay map and other data and content processing concentrated on the front-end client are extracted, and then moved to the server to form a general high concurrency man-machine service. The client chooses the TCP or WebSocket protocol according to the needs. For example, you can choose the TCP long chain in the I area, and the WebSocket long chain in the III area, communicate with the man-machine service proxy server, and send the screen refresh request through the reverse proxy and load balancing module to the man-machine server. Among them, the reverse proxy shields the server, combined with the firewall, improves the security of the man-machine server, and accelerates the access of the server. And load balancing balances the load between multiple man-machine services to prevent impact on a single man-machine service.

2.2. Internal Framework of Man-machine Service

In the design of lightweight man-machine architecture, by classifying the functions of the graphics editor and browser, it can be divided into non-real-time refresh tasks and real-time refresh tasks. Non-real-time refresh tasks include relatively independent and immediate return operations such as G file download, version comparison, scene acquisition, editor database attribute box and device dynamic association. Real-time refresh tasks include screen data refreshing, equipment dynamic coloring, curve/table refreshing, power sign drawing, automatic generation of electrical bay map, and other services that require real-time refreshing. Non-real-time refresh tasks are usually relatively single, without complicated processing logic, and such operations are usually triggered by the front-end according to the business, and will not run continuously. The real-time refresh task has high requirements for timeliness, and the image will keep refreshing until the image is closed. In addition, the real-time refresh task involves many processes, from the acquisition and analysis of G files, the refresh of telemetry and telesignal data, the acquisition of equipment status, voltage, topology and other information, to the calculation of equipment coloring information, and the hiding of invalid equipment, which is a very complicated process. The processing time of a single task is much longer than that of a non-real-time task.

Aiming at the division of two types of business, man-machine service provides an extensible plug-in framework based on thread pool. Non-real-time refresh tasks can be directly processed in the plug-in. The thread pool can also avoid business accumulation and improve the throughput of man-machine services. For the business characteristics of real-time refreshing tasks that are complex, time-consuming, and require continuous refreshing, the man-machine service has designed a real-time task
processing architecture composed of scheduler, FIFO, cache and sender, and task processing process. Real-time tasks are processed by independent processes, which are subsequently called "jobs". As shown below.

![Man-machine service task scheduling diagram](image)

Figure 2. Man-machine service task scheduling diagram.

When the man-machine service receives various real-time refresh requests from the client, it assigns it to a job according to the request parameters, and maintains a complete cache and FIFO for the job, where the FIFO is used to store real-time changing data. When the same request is received again, the man-machine service directly returns the complete data in the cache to the client, and adds the client to the FIFO listening queue to obtain subsequent change data. The man-machine service framework itself does not perform business processing, but sends tasks to specific jobs through the scheduler, and these jobs perform business processing. The man-machine service is responsible for returning the task processing results to the requesting client.

3. Key Technology

3.1. Load Balancing

In order to avoid business accumulation and single-machine failures from affecting the scheduling operation, the man-machine service framework is designed to ensure that the service can maintain normal system performance even under high load, and it can operate normally even in single-machine failures. After detailed research and analysis, the man-machine service solves the problem of high load of man-machine service through load balancing at the access layer.

Comprehensive analysis of multiple load balancing schemes, the man-machine service uses Nginx[5] +Keepalived[6] to provide load balancing and failover at the access layer. The Nginx load balancer is responsible for adopting a certain strategy, calculating the optimal node based on the collected load information, and returning the result to the load balancer. The load balancer forwards the client's request to the optimal node, so that all man-machine servers can share responsibility for business processing under normal conditions and achieve load balancing. Load balancing can locate man-machine servers with low latency, light load, and good performance according to the balancing strategy and server status, so as to make full use of system resources, avoiding some machines from being idle and some machines from being in a high load state.

Power grid dispatching is a system with high real-time requirements. When a man-machine server fails, or a man-machine server cannot handle massive requests, if the dispatcher fails to switch in time, unpredictable consequences will occur. Keepalived[7] can effectively detect the status of the server. If a man-machine server fails, keepalived can detect and remove it from the system in time, and transfer its tasks to other man-machine services. After detecting that the server returns to normal, Keepalived can automatically add the server to the system. Keepalived can effectively shield the various situations of each server node, and achieve transparent access from the client layer to the business layer.

3.2. Thread Pool Technology

For non-real-time tasks, man-Machine Service provides an extensible plug-in framework based on thread pools. The thread pool scheduling technology[8] can effectively improve the service response
speed, and multiple requests can be processed concurrently without waiting for the end of the previous request. At the same time, the thread pool can create a certain number of worker threads in advance and limit their number to control the memory consumption of thread objects. In addition, the thread pool can be reused, which reduces the cost of thread creation and destruction.

However, due to the particularity of the grid screen refreshing service, there is a certain sequence between each task. For example, it is necessary to compare the versions first, then obtain the G file, and then refresh the data. The scheduling of threads is uncontrollable, and there is no guarantee that requests can be executed in sequence and returned in sequence, which easily affects client services. In response to this problem, the man-machine service has transformed the traditional thread pool. When assigning tasks, it will locate according to the client ID, and locate the requests sent by the same client to the same thread, which realizes the concurrent processing of each terminal, prevents some clients from failing to respond for a long time, ensures that the requests of each terminal can be responded to in sequence, and returns the results in order to ensure the continuity of client operations.

3.3. Job Scheduling Management
The power grid dispatching system is a large-scale monitoring and control system with second refresh, which requires extremely high timeliness of real-time refresh tasks. And the refresh of power grid picture involves many business logics, such as file analysis, equipment coloring information calculation, and power sign drawing. According to the characteristics of real-time refresh business, the man-machine service transfers the real-time refresh tasks into independent processes for processing. The man-machine service framework itself does not process real-time business, but integrates a distributed job scheduling management module[9], which is used for scheduling and transferring message of these processes. The job management module mainly has the following functions.

3.3.1. Task merging and forwarding
In order to relieve the performance pressure of a single service and avoid the performance bottleneck of a single service due to the complexity of the business, the man-machine service has split the service to realize the miniaturization and fragmentation of the man-machine service. By analyzing the types of grid services, real-time refresh tasks can be divided into these categories: content refresh, data refresh, color refresh, etc. The content refresh result is mainly G format text, which is suitable for power sign service and automatic generation of electrical bay map service; data refresh is mainly dynamic data information, generally numeric and string; color refresh mainly includes line color, fill color, flashing color, and other information. Secondly, through the analysis of the screen refresh content, the power system image is not only identified by the file name, but also based on the scene, sub-scene and other information to identify the content. Finally, analysis the grid control system from the time dimension. Because the file will be constantly modified, the man-machine service needs to realize the real-time refresh of multiple versions of the same file at the same time.

Through the analysis of type, content and time, the man-machine service is divided into several services such as data service, coloring service, power sign service, electrical bay map service and so on. The man-machine service analyzes the parameters requested by the client, and merges the requests of the same file, version, scene, sub-scene and other information into the same request. Only when the request is received for the first time, will one be assigned to the job process, and create a cache and change message monitoring queue in memory. The subsequent requests will get data from the cache and join the change message monitoring queue.

3.3.2. Process management
Considering that the single machine operation will be limited by the performance of a single server, and can not make full use of the resources of other nodes in the system, and the stand-alone mode will have certain limitations in ensuring the efficiency and reliability of the task, the man-machine service job scheduling management module carries out distributed task management for each job. The man-machine service submits the job to the distributed job management module. The distributed job
management module comprehensively considers the memory, CPU and other status of each node, and dynamically starts the job on each node to improve the operating efficiency of the job.

Due to the complexity of the business, the man-machine service starts the jobs after receiving the task request, which will affect the data return speed. When the job scheduling management module is started, it will preload a part of the job process. These jobs will cache some general data during initialization, and they can be processed immediately when the task is received. A process that processes only one task will cause a waste of system resources, and too many tasks processed by a process will affect the refresh performance of the terminal. The job management module will monitor the number of tasks within each job in real time and dynamically adjust the number of job processes. The job scheduling module can flexibly adjust the number of jobs of each type according to the number of preloaded jobs and upper limit tasks configured for each type. When all the jobs of the same type have reached the task limit, the job scheduling module can dynamically expand new jobs. When the business peak passes and there are too many idle jobs, the man-machine service can automatically reclaim the redundant jobs to achieve flexible expansion of processing capacity.

3.3.3. Status monitoring and fault recovery
In order to avoid the impact of task assignment failure on client operations, the job management module manages the status of each client request, records the allocation of each request, and reallocates the failed tasks in time. At the same time, the job management module also monitors the communication with each job. Once the communication is abnormal, all tasks under the job will be reassigned to ensure the stable operation of each task. In addition, the distributed job management module can automatically monitor the status of each job, and divide each job into initialization, start, running, failure, offline, exit, termination, and waiting states. When the man-machine service job management detects that the job is in an abnormal state such as failure, offline, and termination, it will reallocate the tasks in these jobs in time to ensure the reliability and stability of the tasks.

The following figure shows the running situation of some jobs in the actual operating system. From the screenshots, you can see that the jobs are distributed on three different nodes, and the service can monitor the status of the jobs in real time. And adjust according to the status to ensure the reliable operation of the tasks.

![Figure 3. Distributed job operation monitoring screenshot.](image)

3.4. Standardized Request and Return Result
The grid dispatching system is a large-scale integrated system that integrates multi-manufacturers and multi-language products. In order to ensure the openness of the service, and provide undifferentiated data refresh, content generation, equipment coloring and other services for C/S clients in production areas, web browsers in non-production control areas, mobile terminals or third-party front-end display clients at the same time, the man-machine service has designed a standard request format and return structure.

The man-machine service requests adopt a restful-like command style, and all requests are divided
into two parts, separated by a question mark "?". The left side of the question mark is used for plug-in and function positioning, and the right side is the request parameter. Each request parameter must contain the callbackid field, and the result will be returned as it is. Parameters are connected by "&". The specific format is as follows:

Table 1. Man-machine service request format.

| Left | Separator | Right |
|------|-----------|-------|
| keep information/plugin name/request command | ? | parameters |

Take the function of client downloading G file as an example, the specific request can be described as the following command:

Platform/FileModule/GetFile?callbackid=xx&filename=xx (1)

In this request, FileModule is the plug-in name, which is used for plug-in positioning, GetFile is the request command, which is used to locate the requested function, callbackid is the front-end request identifier, and the following are detailed request parameters. Based on this format, we can describe all service requests.

The man-machine service return structure adopts the JSON[10] standard, which is a lightweight data exchange format, a storage and data representation format that has nothing to do with programming language. It is efficient in reading comprehension, machine analysis and generation, and network transmission. The information returned by the man-machine service is assembled into the following structure:

{"callbackid":"xx","retcode":"xx","message":"xx","length":"xx"}0content (2)

Among them, callbackid is the client request identifier, which is convenient for the client to locate the request result, and length is the length of the content after the terminator, which is filled in by the service as needed.

At present, in addition to the C/S man-machine cloud terminal, the man-machine service is also applied in the mobile display project in Shandong. At the same time, it is applied in the new generation man-machine cloud terminal of NARI Relays. These applications verify the openness and high availability of man-machine services.

4. Testing

After a large number of sample simulation tests and on-site operation statistics, non-real-time refresh tasks such as G file acquisition and scene acquisition can all be returned in tens to hundreds of milliseconds. The real-time refresh tasks such as data refreshing, color refreshing, and power sign refreshing are slightly longer than the fat client at the first request, but when the same file is requested again, the data will be directly obtained from the cache, which is significantly faster. In addition, the man-machine service has been running steadily in Jiangsu, Ningxia, Shanghai, Zhejiang and so on, which fully witnessing the reliability of the man-machine service.

The following data tests the average processing time of 500 different requests received by the man-machine service at the same time. The sample is the G file in the Jiangsu new-generation demonstration environment. The test environment is the Linx-Tech 4.2 64-bit operating system, the memory is 32G, and the CPU is 8 cores.

Table 2. Test result

| Service name          | First request | Second request |
|-----------------------|---------------|----------------|
| 1 Data refresh        | 1465ms        | 634ms          |
| 2 Color refresh       | 1534ms        | 658ms          |
| 3 Power sign refresh  | 1081ms        | 415ms          |
5. Conclusion
With the gradual application of lightweight man-machine cloud terminal and man-machine service, new developments have been brought to power grid dispatching. The application of man-machine service enables consistent access between C/S terminals, B/S terminals, mobile terminals and man-machine terminals of third-party manufacturers, and there will be no inconsistencies in information such as image data and color between different manufacturers. However, a large number of requests from different clients put forward higher requirements on man-machine services. How to ensure the reliability of man-machine services is a very important topic. Aiming at this problem, this paper describes the architecture design of man-machine service, and introduces the key technologies used in it. The man-machine service meets the requests of different terminals through a variety of interactive protocols, standardized requests and return results. Through load balancing technology, to ensure the stability of multiple man-machine servers to prevent excessive load. Through the thread pool technology suitable for power grid browsing business, it prevents the accumulation of tasks and guarantees the order of single client requests and responses. Through the distributed job scheduling management technology, the jobs are distributed on each node to run stably, the tasks and jobs are monitored in the scheduling management module, and the cache is used to ensure the efficient and stable operation of each task.

References
[1] ZHANG Y., WANG A.H., ZHANG H., (2021) Overview of smart grid development in China. Power System Protection and Control., 49: 180-187.
[2] HAN M.H., YAO J.G., YANG S.C., et al, (2016) Study on Architecture of New Generation of Dispatching Technical Supporting System for Smart Grid. Southern Power System Technology., 10: 1-7.
[3] ZHAO K., YAN Z., YIN X., et al, (2018) Design and implementation of high concurrency Man-machine service framework for power grid control system. In: 2018 joint annual meeting of power system Professional Committee and new energy grid connection and operation professional committee of China Electrical Engineering Society. WuHan. pp. 367-372.
[4] HUANG K., ZHAO K., YANG L.B., et al, (2019) Architecture and Key Technologies of Lightweight Human-Machine Interaction System for Power Grid Dispatching and Control System. Automation of Electric Power Systems., 43: 159-165.
[5] LIU J.Y., CUI J.M., ZHI C., (2020) Dynamic load balancing strategy based on Nginx server. Journal of Guilin University of Technology., 40: 403-408.
[6] WANG P., XU Y., (2007) A new effective Keep Alive arithmetic used in embedded application. Journal of Computer Applications., 27: 38-39.
[7] ZHANG K., (2020) Experimental simulation of Web cluster system based on LVS+Keepalived. Journal of Shaanxi University of Technology: Natural Science Edition..36: 44-48.
[8] PENG H., (2019) Effect of Thread Pool Scheduling on Server Performance. Communications Technology., 52: 2172-2176.
[9] GU W.J., GAO Y., PENG H., et al., (2017) Architecture and Key Technologies of Distributed Task Management for Power Grid Dispatching and Control Systems. Automation of Electric Power Systems., 41: 109-115.
[10] Tim B., (2013) The JSON Data Interchange Format. Standard ECMA-404.