Proceedings of the 3rd International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2025.30490

Intelligent Scheduling and Observability Optimization Technology for Massive Traffic in Large-Scale Power Dispatch Cloud

Wenchong Fang[1*], Wei Jiang[1], ZhiFeng Zhou[1], Yi Zhu[2], Fei Chen[3]
[1]China Southern Power Dispatching and Control Center, China Southern Power Grid Co., Ltd., Guangzhou, China
[2]Kylinsoft Corporation, China National Software & Service Company Limited, Tianjin, China
[3]Yunnan Power Dispatching and Control Center, Yunnan Power Grid Co., Ltd., Kunming, China
**Corresponding Author. Email: fangwc@csg.cn*

Abstract. In large-scale power dispatching cloud environments, massive traffic dispatching suffers from low efficiency and insufficient observability. This paper proposes a seed-end-cloud collaborative intelligent dispatching and observability optimization scheme. A three-tier pipeline architecture based on a lightweight agent, a stateless processing cluster, and distributed storage is designed. Source traffic shaping is achieved through real-time data aggregation at the agent end and Zstandard compression technology. A server-led dynamic rebalancing mechanism and CNI network plugin traffic shaping strategy are used to address instantaneous traffic surges and uneven load distribution. Storage layer performance is optimized through a self-developed high-availability connection pool and a dual-trigger batch write mechanism. In a simulated 10,000-node power dispatching scenario, the system throughput reaches 503.7 thousand records per second, with end-to-end latency controlled within 115.6 ms, providing feasible support for high-concurrency, high-reliability power services.

Keywords: Power Dispatch Cloud, Intelligent Traffic Dispatch, Observability Optimization, Edge-cloud Collaboration

1. Introduction

With the rapid growth of new energy installed capacity and the advancement of digital transformation of the power system, the number of power grid dispatching targets is increasing exponentially. Against this backdrop, cloud computing technology, with its advantages of elastic scaling and resource pooling, has become the core support platform for building a new generation of large-scale power dispatching systems.

This paper proposes and implements a massive traffic intelligent scheduling and observability optimization technology system for large-scale power dispatching clouds: A three-level pipeline architecture with edge-cloud collaboration is designed, achieving separation of the data plane and control plane, and laying a resilient and observable foundation for the system through a closed-loop feedback mechanism. A full-stack optimization scheme covering the data lifecycle is proposed. At the source side, source traffic shaping is achieved through real-time data aggregation and compression at the agent end and a fixed-frequency reporting mechanism. At the transmission and computing layers, highly available and elastic processing clusters are built using stateless services, elastic scaling, and ring buffer decoupling technologies. At the observability level, panoramic monitoring and intelligent traffic scheduling are achieved through multi-dimensional data acquisition, server-led dynamic rebalancing, and CNI network traffic shaping strategies. At the storage layer, efficient and stable data writing is ensured through a self-developed high-availability connection pool and a dual-trigger batch write mechanism.

2. Related work

With the rapid growth of new energy installed capacity and the digital transformation of power system, the grid dispatch objects are showing an exponential growth. The emergence of Unified Power Flow Controller (UPFC) has changed the pattern of power flow control in power system. Naderi et al. [1] applied evolutionary algorithm to optimize the active power allocation considering the unified power flow controller in a fuzzy framework, used the fuzzy framework to deal with the uncertainty factors in the power system, and searched for the optimal active power allocation scheme through evolutionary algorithm. Ibrahim et al. [2] adopted the alternating optimization method to solve the problem of optimal reactive power allocation in multiple time periods for voltage safety, and gradually optimized the reactive power allocation scheme through alternating iteration. Qi [3] took a certain district-level power system as an example to carry out the design research of remote automated dispatch method. Zhang et al. [4] pointed out that the decoupling of system software platform and application, as well as the adoption of related technologies such as standardized interface definition, data subscription and publishing technology, are conducive to promoting the continuous expansion of the later network-level power dispatch application analysis business. Xu Heyan [5] designed an automated system based on cloud computing platform, and realized the intelligent analysis and dispatch strategy planning of power big data through cloud computing platform.This paper focuses on the research of intelligent dispatch and observability optimization technology for massive flow in large-scale power dispatch cloud. It aims to explore efficient dispatch strategies and optimization methods to meet the complex needs of modern power system, so as to provide support for the safe, stable and economical operation of power system.

3. Method

3.1. Scheme design

This solution constructs a three-tier pipeline architecture for edge-cloud collaboration, achieving full-stack optimization of the data link. (as shown in Figure 1.)

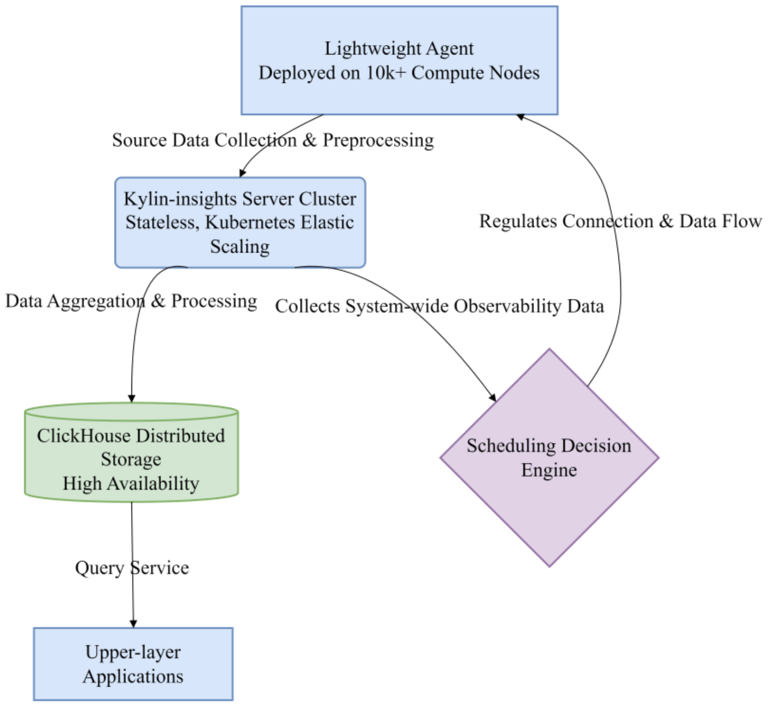
Lightweight Agent
Deployed on 10k+ Compute Nodes
Source Data Collection & Preprocessing
Kylin-insights Server Cluster
Stateless, Kubernetes Elastic Scaling
Regulates Connection & Data Flow
Data Aggregation & Processing
Collects System-wide Observability Data
ClickHouse Distributed Storage
High Availability
Scheduling Decision Engine
Query Service
Upper-layer Applications

Figure 1. End-to-cloud collaborative architecture

The data flow begins with a lightweight Agent deployed on tens of thousands of computing nodes, responsible for collecting and initially processing the source data. The processed data is then aggregated and processed by the stateless kylin-insights Server cluster within the cloud environment. This cluster is built on Kubernetes and has elastic scaling capabilities. Finally, the processed data is persisted to the highly available ClickHouse distributed storage cluster to provide query services for upper-layer applications. The control flow, on the other hand, is dominated by the Server cluster. By collecting observable data across the entire domain, it generates scheduling decisions and controls the connections and data flow of the Agents in reverse.

3.2. Source governance: agent-side data compression and traffic shaping

On the agent side, real-time aggregation of the raw observable data is performed in memory. This operation packages multiple system metrics such as CPU and memory at the same timestamp into composite data units, reducing the number of independent data packets and protocol header overhead. If n independent metrics $d_1, d_2, \ldots, d_n$ are collected at the same time t, the aggregation process is as follows:

$$D_t = \bigcup_{i=1}^{n} d_i(t) \tag{1}$$

D_t represents the composite data unit corresponding to timestamp t. Next, Zstandard is used to compress the aggregated data block. Addressing the high redundancy of indicator data in power monitoring scenarios, a compression ratio approaching 100:1 is achieved by adjusting the dictionary and compression level. The compression process can be modeled as follows:

$$C(D_t) = \mathrm{ZstdCompress}(D_t, L, D) \tag{2}$$

L represents the compression level, and D represents the preset dictionary. Both work together on highly redundant data to achieve the target compression ratio.

The Agent does not send data immediately after it is generated, but adopts a batch reporting mechanism with a fixed time window. All data collected within a time window is aggregated and compressed as described above, and then combined into a large data packet for one-time reporting.

3.3. Optimization of transmission and computing layers: high availability and elastic scaling processing cluster

3.3.1. Stateless service and HPA elastic scaling

Kylin-insights Server is designed as a totally stateless service, all instances are deployed in kubernetes clusters as Pods. The context information that is needed by the business logic processing is not stored in the server memory, but based on the external storage or request itself. HPA's monitoring metrics are not just limited to CPU and memory integration, but also custom Queries-Per-Second and pending data queue length metrics are integrated.

Within each Pod of the Kylin-Insights Server, a high-performance circular buffer is deployed as the core memory queue. After receiving and decompressing the data packets reported by the Agent, the network I/O thread does not immediately perform complex business logic processing. Instead, it writes the formatted data objects to the Ring Buffer in a non-blocking manner.

3.4. Observability optimization: panoramic monitoring and intelligent traffic scheduling

3.4.1. Multi-dimensional data acquisition and aggregation

To achieve awareness of the system's own state, a multi-dimensional data collection system is constructed. Each Agent, in addition to reporting observable business data, also reports its own metadata, including node resource utilization, estimated egress bandwidth (B_{out}), and connection health status (H_{conn}) with the Server. This metadata can be formally represented as $M_{agent}=\{R_{cpu},R_{mem},B_{out},H_{conn}\}$, where R_{cpu} and R_{mem} represent CPU and memory utilization, respectively.

3.4.2. Server-led dynamic rebalancing strategy

Traditional client-side or middleware load balancing strategies struggle to handle fine-grained traffic shifts. This solution proposes a server-side-led dynamic rebalancing mechanism. The controller component in the kylin-insights Server cluster periodically calculates the weighted average load of all Pod instances:

$$\bar{L}=\frac{1}{N}\sum_{i=1}^{N} w_i L_i \tag{3}$$

N represents the total number of Pod instances; L_i refers to the load metric of the i-th instance; and w_i is its corresponding weight coefficient. When the system detects that the current load L_j of a Pod instance continuously exceeds a set multiple β of the global average, i.e., when $L_j>\beta\bar{L}$ is met, the instance is determined to be a hotspot.

3.4.3. CNI network plugin traffic shaping

To defend against instantaneous network surges caused by Agent reconnection or sudden business activity, a Pod-level inbound traffic shaping strategy was implemented at the Kubernetes CNI network layer. This strategy is based on the Linux Traffic Control subsystem and sets a token bucket filter for each kylin-insights Server Pod virtual network interface.

3.5. Storage layer optimization: high availability connections and batch writes

To overcome the shortcomings of the native ClickHouse client in handling long-lived connections, a high-availability connection pool was developed in-house. This connection pool is configured during initialization and establishes persistent TCP connections with all replica nodes in the ClickHouse distributed data storage cluster, thus avoiding the overhead of TCP handshakes and TLS negotiations for each write request. The connection pool integrates a proactive health check mechanism that periodically sends liveness checks to all nodes.

To address the "Too many parts" problem that ClickHouse is prone to in high-frequency, small-batch write scenarios, a concurrency-safe memory buffer and a dual-trigger write mechanism were designed on the kylin-insights Server side. After processing the data, the server does not immediately write it to ClickHouse, but instead temporarily stores it in a memory buffer partitioned according to the target table.

4. Results and discussion

4.1. Performance evaluation and analysis

This paper constructs a large-scale power dispatch cloud experimental environment. Based on a private cloud platform, this environment simulates a typical dispatch scenario with 10,000 computing nodes. To generate simulated observable data streams, a lightweight agent is implemented for each computing node. The system compares the performance of traditional polling scheduling and intelligent scheduling in terms of system throughput, end-to-end latency, and resource utilization. Specific performance comparisons are shown in Figures 2, 3, and 4:

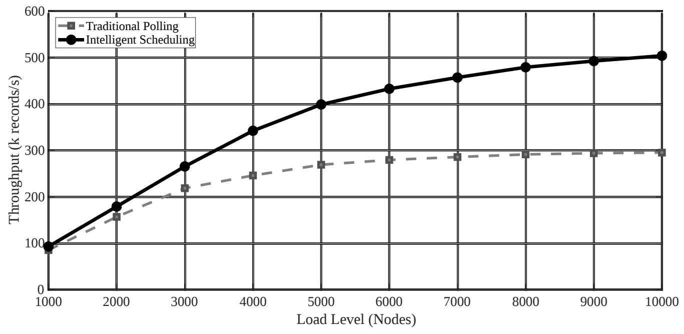

Figure 2. Throughput

At every load, the intelligent scheduling strategy works better than the traditional round-robin scheduling, especially when the number of nodes is more than 4000. Obviously, the traditional strategy has a clear performance bottleneck after the number of nodes is more than 6000, and the throughput does not increase much, which is only from 279,300 messages/second to 295,100 messages/second, the increase rate is less than 5.7%. In contrast, the throughput of the intelligent scheduling under the same conditions increases almost linearly from 432,500 messages/second to 503,700 messages/second, a positive change of 16.5%. The reason is due to the dynamic rebalancing mechanism of the intelligent scheduling which has excellent single-point congestion avoidance capability. In addition, the circular buffer design decouples I/O and computation, and allows the system to fully exploit the parallel computing resource provided by the distributed architecture, so that it exhibits an important advantage in traffic scheduling.

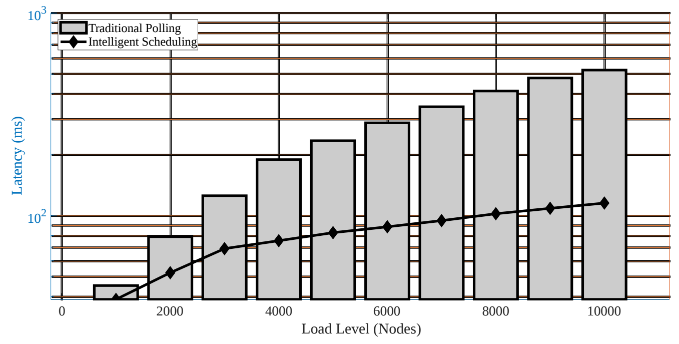

Figure 3. End-to-end delay

Traditional polling latency increases almost exponentially with load, rising sharply from 45.3 milliseconds with 1000 nodes to 523.4 milliseconds with 10000 nodes, an increase of more than 10 times. In contrast, intelligent scheduling latency increases monotonically from 38.7 milliseconds to 115.6 milliseconds, an increase of less than 3 times. Compared to traditional scheduling, intelligent scheduling exhibits significantly lower load instability under high loads. More importantly, in the critical range of 3000-5000 nodes, intelligent scheduling, through a server-side dynamic rebalancing mechanism, can effectively maintain latency below 85 milliseconds, while traditional strategies result in latency exceeding 200 milliseconds.

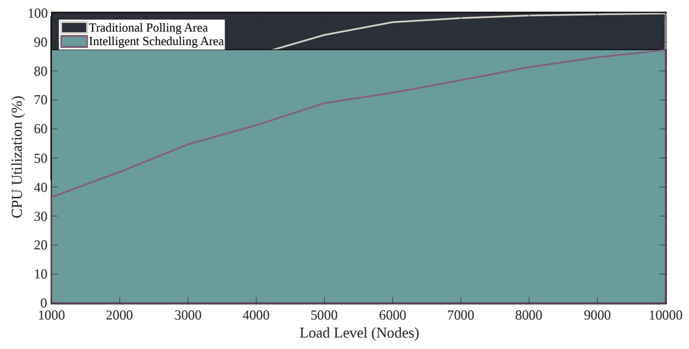

Figure 4. CPU utilization

The stacked area diagram visually shows the resource utilization pattern of the two strategies: traditional round-robin CPU utilization quickly reaches saturation (96.8%) after 6000 nodes, which has the characteristics of resource-constrained characteristics; while intelligent scheduling can save more than 24 percentage points, and consume only 72.5% of CPU resources under the same load. The grey line between the two plots illustrates the fact that the resource savings are greater with load, up to an absolute value of 12.6% at nodes in the range of tens of thousands. The obtained results demonstrate that intelligent scheduling gets considerable resource savings while guaranteeing performance improvement and keeping a redundancy margin for the system to cope with unexpected traffic spikes.

4.2. System resilience and fault recovery testing

Five failure scenarios were simulated sequentially: stopping 10% and 20% of Agent nodes, terminating single and multiple kylin-insights Server Pods, and shutting down the ClickHouse master node. After each failure injection, the Prometheus monitoring system recorded the failure detection time, service recovery time, data loss rate, and performance degradation degree (expressed as a percentage decrease in throughput). The results are shown in Table 1.

Table 1. Fault recovery test results

Fault Scenario	Fault Detection Time (s)	Service Recovery Time (s)	Data Loss Rate (%)	Performance Degradation (%)
Stop 10% Agent Nodes	2.1	10.5	0.1	5.0
Stop 20% Agent Nodes	3.2	15.8	0.2	10.2
Terminate Single Server Pod	1.0	5.2	0.0	2.1
Terminate Multiple Server Pods	2.5	8.7	0.1	5.5
Shutdown ClickHouse Primary Node	5.0	10.3	0.0	3.0

Table 1 indicates that the fault detection time and the service recovery time become longer as the fault size is enlarged. The longest recovery time (15.8 seconds) is the one with 20% of Agent nodes terminated, and the fastest recovery time (5.2 seconds) is with one Server Pod terminated. In general, the fault recovery mechanism is stable, and the data loss can be controlled, which meets the requirements of high availability of the power dispatch cloud.

5. Conclusion

This paper's architecture combines a lightweight agent, a stateless processing cluster, and distributed storage to achieve a comprehensive solution encompassing data compression, dynamic rebalancing, traffic shaping, and batch writing, enabling efficient intelligent traffic scheduling and panoramic scene information processing. However, this research still has limitations in anomaly detection and prediction intelligence, and the experimental environment differs from the production environment. Future research will introduce artificial intelligence technology to achieve fault prediction and self-healing, and investigate its deployment and application in complex heterogeneous environments.

References

[1] Naderi E, Mirzaei L, Pourakbari-Kasmaei M, et al. Optimization of active power dispatch considering unified power flow controller: application of evolutionary algorithms in a fuzzy framework [J]. Evolutionary Intelligence, 2024, 17(3): 1357-1387.
[2] Ibrahim T, De Rubira T T, Del Rosso A, et al. Alternating optimization approach for voltage-secure multi-period optimal reactive power dispatch [J]. IEEE Transactions on Power Systems, 2021, 37(5): 3805-3816.
[3] Qi Sennan. Research on remote automated dispatching of district-level power systems based on cloud computing [J]. Automation Application, 2025, 66(16): 279-281
[4] Zhang Changkai, Liu Bin, Xie Kai, Li Ying, Zhang Zhixue. Application research of urban rail transit network-level power dispatching system [J]. Urban Rail Transit Research, 2025, 28(4): 238-242

[5] Xu Zhihao, Yan Ren. Architecture and optimization of power dispatching automation system based on cloud computing platform [J]. Automation Application, 2024, 65(8): 75-77