Monitoring System Practices and Explorations in Heterogeneous AI Platforms

Jinhua Wang[1], Ruiyi Liao[2*], Jianfang Luo[3]

[1] Guangzhou AIdynamic Technology Co., Ltd.,510000, China

[2] Guangdong Urban-rural Planning and Design Research Institute Technology GROUP CO. LTD., 510000, China

[3] Guangzhou AIdynamic Technology Co., Ltd.,510000, China

Corresponding Email: liaory@139.com

https://doi.org/10.70695/shuysw13

Abstract

With the growth of AI's business scale in the vertical field, there is a concomitant increase in the number of servers and related services. This escalation leads to a substantial surge in the volume of monitoring data. In response to this scenario, this paper proposes modular architecture design for monitoring services. This design enables comprehensive monitoring of devices, software, and networks, encompassing device status monitoring, software operation quality control, and real-time monitoring of network links and bandwidth. The objective is to meet the exigencies of handling the vast and diverse monitoring data. The system is capable of collecting and displaying data in real-time, promptly detecting the health, performance, and error conditions of systems or applications. It also supports user-defined rules for monitoring alerts and provides a real-time view of trends. When the system or application experiences or is about to experience a failure, the monitoring system must respond expeditiously and issue alerts, thereby facilitating rapid problem resolution or preemptive prevention. By integrating AI task types, platform resource utilization, and network quality analysis, global fine-grained scheduling and resource elastic expansion can be achieved. These operations significantly improve resource utilization of heterogeneous AI computing platforms, optimize task execution efficiency, and reduce operational costs.

Keywords Monitoring System; Heterogeneous AI Computing Power; Data Collection; Resource Utilization; Fine-grained Scheduling

1 Introduction

In modern technological landscapes, monitoring systems are of paramount importance. They provide real-time surveillance of system operational status and performance metrics, enabling the early detection of potential faults and anomalies. This, in turn, facilitates system performance optimization, ensures system security, and underpins data-driven decision-making. Supporting AI computing power includes different types of computing devices, such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), each with unique strengths and applicable scenarios. However, how to effectively manage and schedule these heterogeneous computing power resources to maximize computing performance and optimize service quality has become an important challenge. As a bridge connecting resource status and scheduling strategy, monitoring system can monitor the usage of computing resources in real-time and provide data support for fine scheduling. Heterogeneous AI computing power contains a variety of different types of server resources, different computing devices need different drivers and programming frameworks, and the allocation and scheduling of computing power tasks also need to take into account the availability of computing resources, the priority of tasks and other factors. Consequently, the scheduling of heterogeneous AI computing power demands real-time adaptability and the ability to be dynamically adjusted based on actual tasks and resource occupation.

2 Related Work3D Monitoring of Heterogeneous AI Platforms

The monitoring system is responsible for the 7x24-hour surveillance of devices, networks, and services within the computing resource pool. It conducts dynamic adjustments of computing resources and task scheduling and, in conjunction with a multi-level monitoring and alarm response system, achieves platform-wide computing perception and monitoring. This ensures service stability and continuous high performance. Based on the contents of references [1] -[5], a monitoring system can collect data on system performance, such as response time, throughput, and resource utilization. Analyzing these data is helpful for finding the performance bottlenecks of the platform and optimizing them to provide a better user experience. Additionally, the monitoring system can detect and record potential security vulnerabilities, attack behaviors, and abnormal activities. This aids in early detection and response to security threats, and it is also helpful for ensuring that the system complies with regulatory and compliance requirements. Serving as the foundational base and engine of heterogeneous computing platforms, the monitoring system monitors the health and operations of the entire computing platform 24/7 (see Fig 1). By continuously monitoring the states and anomalies of various devices, nodes, links, and services, it leverages intelligent decision-making algorithms to support computing perception and fault tolerance.

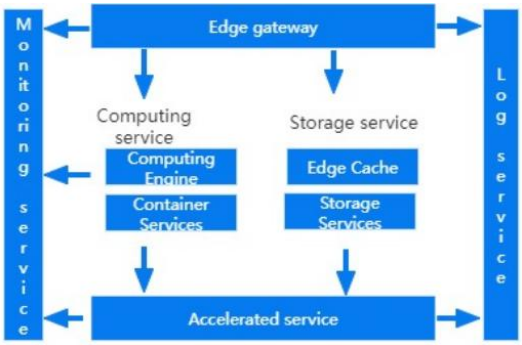

Fig. 1. Monitoring service in AI platform

2.1 Device Monitoring

Equipment is the cornerstone of computing services. The primary objective of monitoring equipment is to ensure its continuous normal operation, thereby providing adequate storage capacity and computing power for computing services:

1.Monitor the status of hardware components such as network cards, network cables, hard disks, and memory;

2.Monitor the running status of servers and switches, including process status, load, cpu usage, memory usage, bandwidth, and packet loss rate.

2.2 Software Monitoring

Software serves as the direct carrier of the computing server. Monitoring software on the computing platform goes beyond checking process vitality, also real-time monitoring software operational quality:

1.Software vitality monitoring to ensure software is operating normally;

2.Software log-level monitoring to uncover impending risks;

3.Service capability monitoring to anticipate insufficient capabilities and dynamically allocate resources.

2.3 Network Monitoring

Networks are the lifeblood of computing platforms and a key focus of monitoring. This includes:

1 .Real-time monitoring of node-level network egress bandwidth for dynamic resource allocation;

2 .Monitoring node-level egress link conditions to promptly replace under performing nodes;

3 .Interface-level network quality monitoring to promptly identify and replace faulty links.

3 Design and Implementation of Monitoring System

3.1 Modular Service Architecture Design

The monitoring system predicated on the Micro-service architecture typically comprises multiple Micro-services, including data collection service, data processing service, data storage service, monitoring indicator calculation service, alarm service, and visual display service. These Micro-services interact through a lightweight communication mechanism to fulfill the monitoring task of the heterogeneous computing platform.

Service function splitting

Monitoring system service functions can be split as follows:

(1) Edge Collection Terminals:The edge acquisition terminal is responsible for monitoring indicator collection and alarm event judgment, and reports the original data and alarm events to the indicator receiving gateway.

(2) Data Receiving Gateway: The data receiving gateway is responsible for receiving the indicators and events reported by the edge, temporarily storing and forwarding them to the upstream central component. Raw data is temporarily stored in a memory queue and periodically pulled by the metric storage gateway. Alarm events are directly forwarded to the event center, while configuration information is obtained from the configuration center and distributed to edge collection terminals.

(3) Event Center: The event center receives alarm events, filters, inhibits, or merges them, selects the events that need to be alerted, submits them to the database for consumption by the alarm center, and obtains the configuration information about the rules from the configuration center[6].

(4) Alerting Center: The alarm center consumes the data in the database queue, sends alarm information according to the rules, and stores alarm events into the database for the monitoring platform to query, and obtains alarm information from the configuration center.

(5) Metric Query Gateway: Index query gateway provides HTTP interface for users to query index data, process external query requests and return results, and obtain the corresponding relationship between the original data and the database from the configuration center.

Deployment mode and advantages

(1) The monitoring system employs multi-active data center deployment, distributing monitoring data traffic and alert workloads across multiple data centers to reduce the risk of service interruption.

(2) When monitoring traffic spikes occur, rapid expansion of the data receiving gateway can be achieved through containerization, enhancing the system's ability to cope with emergencies[7].

(3) The Micro-service modular design facilitates efficient system maintenance, with each service unit facilitating bug fixes, new feature deployments, testing, reducing single-point risks, and improving reliability, availability, and flexibility. After service decomposition, loose coupling is achieved.

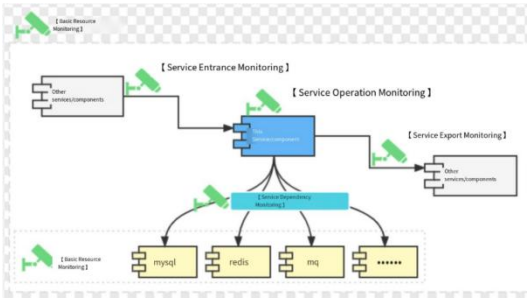

Fig. 2. Monitoring Platform Deployment Architecture Diagram

3.2 Modular Service Architecture Design

For the collection of server indicators and business indicators, the monitoring system includes HTTP mode, code plug-in script mode (where data is collected based on the periodic output of script results) and read fixed directory and fixed data format file collection. As illustrated in the following figure:

Fig. 3. Monitoring Metric Data Reporting and Collection Methods

Users can customize the alarm conditions and acquisition strategy according to the actual needs. After the configuration of the monitoring platform, the monitoring bottom will update the alarm strategy to the edge acquisition terminal regularly. When edge components receive reported metrics, they judge whether they meet the alert strategy, generate alert events if true, notify users, and send corresponding notifications for recovered strategies.

3.3 Network Quality Analysis of the Monitoring System

Full-link tracking

The monitoring system has developed full-link tracing, embedding points in the call chain between each Micro-service to track the overall network link. A trace is added to each request at the start, generating call chain segments through context propagation and submitted upon service completion. Trace data is periodically reported to the message queue, consumed by corresponding components, and stored in the database for query.

Fig. 4. Example of Full-Link Tracing

This diagram easily identifies system issues, such as an API request taking 2s, with HTTP and gRPC calls contributing 500ms and 200ms respectively.

Basic Network Metrics Monitoring

The monitoring system monitors the buried points of network links based on the network performance analysis tool (ping, curl, trace path), turning basic network data into monitoring metrics. These metrics, including inbound/outbound bandwidth, packet transmission counts, packet loss, TCP transmission ratios, and network connection counts, are reported real-time via scripts and configured for alerts.

Service Monitoring

Business-level monitoring: With the help of the powerful computing power of big data cluster, the monitoring system monitors and analyzes the access logs of AI computing power business in real time, and mines the abnormal fluctuations of the business in real-time from the perspectives of status code, request number, access behavior, etc.

Content monitoring: According to the business scenario of computing power demand, real-time monitoring of computing power needs content, and dynamic adjustment of computing power pool resources to ensure sustainable services.

User monitoring: Through deep mining and learning, real-time mining of the service quality of each channel, gradually screen out the characteristics of poor quality groups, and targeted to improve the quality of service.

Service quality monitoring can effectively reduce the probability of failure and ensure that user access will not be affected to the greatest extent. A public cloud platform analyzes the problems occurring in the third and fourth quarters of 2023.

With a perfect node monitoring mechanism, through the internal edge node network, equipment, and link monitoring system, it can find system faults in the first time and automatically switch nodes to reduce the probability of failure.

Fig. 5. Monitoring and Early Warning Effectively Reduces the Probability of Failure

4 Grained Scheduling Strategies for Platform

The design and implementation of an intelligent traffic scheduling system within a multi-IDC, CDN, and cloud host environment, featuring a low-latency scheduling server and an intelligent core computing module, is achieved by integrating with the service monitoring system. Through data monitoring and analysis, edge device scheduling, computing resource load scheduling, load balancing, and dynamic adjustment, it realizes the efficient utilization of computing resources, power consumption reduction, and task execution acceleration.

4.1 Platform Scheduling Objectives

Improve resource utilization

Fully utilize the advantages of heterogeneous computing resources,including the computing power of edge devices and central servers, to avoid idle resources. Reasonably allocate various computing power resources (such as CPUs, GPUs, FPGAs, etc.) in different tasks to maximize overall resource utilization.

Improve task execution efficiency

Reduce the waiting time and execution time of tasks, and ensure that tasks with high real-time requirements can be processed in time. Allocate resources rationally according to task priorities and needs to improve the speed and quality of task execution.

Implement load balancing scheduling

Avoid overloading certain devices or nodes while others remain idle. Achieve balanced load distribution between the edge AI computing power server and the central server to improve the stability and reliability of the entire system.

Real-time scheduling for all platforms

Users submit custom models and deployment configurations to the console, which can monitor and schedule globally according to real-time, and quickly respond to dynamic changes of tasks in the system platform, changes in resource availability, and changes in network conditions. Flexible scheduling capability to adapt to different workloads and environmental conditions.

4.2 Fine-Grained Scheduling Strategy Formulation

Computing Resource Load Scheduling

Select appropriate computing resources for scheduling, and prioritize assigning tasks to computing resources with lower loads according to task types (such as deep learning, image processing, big data analysis, etc.) and computing needs. When a new task arrives, query current platform resource loads and allocate tasks to the least loaded resources. Balance task allocation based on task urgency and resource load. When resources are abundant, allocate multiple tasks compactly to a few nodes to increase resource utilization. In resource-constrained situations, distribute tasks across multiple nodes to avoid overloading single nodes. Implement task queuing and priority management, ensuring high-priority tasks receive resources first.

Load balancing and dynamic scheduling

Continuously monitor resource load during task execution. For example, if the edge device's network connection deteriorates or the load increases, migrate the executing tasks back to the central node or other ed If a resource is overloaded while others are lightly loaded, migrate some tasks from the overloaded resource to the lightly loaded ones. Simultaneously, implement cross-layer scheduling between edge devices and the central cloud, flexibly migrating tasks between the edge layer and the cloud based on task requirements and network status. For edge devices, adjust task assignment in real-time according to their dynamically changing load and connection status.ge devices. For example, if the edge device's network connection deteriorates or the load increases, the tasks being performed can be migrated back to the central node or other edge devices. Dynamically adjust the scale of computing resources based on changes in the workload, monitoring system performance metrics such as adding or removing servers and adjusting VM configurations to handle sudden computing demands.

Edge device scheduling and elastic expansion

Prioritize processing simple or high real-time tasks on edge devices to reduce data transmission delays and bandwidth consumption. Reserve computing resources for critical or high real-time tasks, ensuring timely processing. Use these reserved resources for other tasks when not needed, but quickly reclaim them when required.Expand or contract compute resources dynamically depending on the load of the platform and task requirements. During peak periods, increase computing power by launching more VMs or containers. During off-peak periods, release excess resources to save costs. Adjust edge device computing resource allocations based on load to ensure efficient task execution.

5 Use Cases

5.1 E-commerce Promotion Scenarios

During large-scale promotions on e-commerce platforms, the system needs to handle a large number of concurrent requests and complex computing tasks. Using the monitoring system based on micro-services architecture, the status of heterogeneous AI computing power resources can be monitored in realtime, and resource allocation strategies can be dynamically adjusted according to the load. For instance, during peak hours, the system can automatically augment the proportion of GPU resources allocated to expedite the execution of tasks such as image processing. Conversely, during off-peak hours, resources can be released for other services to enhance resource utilization.

5.2 Automatic Driving Scenarios

Autonomous driving systems have dynamically changing computing demands, especially in complex driving scenarios. By monitoring the real-time data of the system, the platform can find potential fault nodes in time, and give corresponding processing suggestions or automatically isolate the fault server. Low latency is one of the key requirements of Autopilot. The platform must respond quickly to changes in the surrounding environment, which requires the platform to have efficient data transmission capabilities and low latency characteristics, and to process and make decisions quickly and locally. Therefore, the platform needs to be able to support data exchange and communication between different computing tasks. As well as collaboration between different service modules to ensure the overall performance of the Autopilot system. Using heterogeneous AI computing power monitoring and scheduling, a large amount of sensor data collected by autonomous vehicles is processed and analyzed in realtime to achieve accurate driving perception and decision-making capabilities.

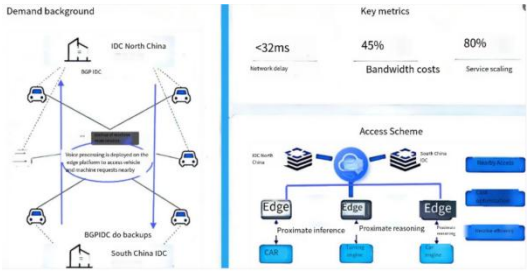

Fig. 6. Cost Reduction and User Experience Enhancement After Migrating an Automotive Enterprise's Autonomous Driving Capabilities to an Edge Computing Power Platform

6 Conclusion and Prospects

By real-time monitoring the operational status of heterogeneous AI computing power, we can provide data support and decision-making grounds for meticulous scheduling, This, in turn, augments the utilization efficiency of heterogeneous AI computing power, optimizes task scheduling strategies, and enhances system reliability and stability. In the future, with the continuous development of technology, the monitoring system is set to become even more intelligent and automated,proffering more accurate and efficient solutions for the scheduling of heterogeneous AI computing power resources. Additionally, we will persist in delving into the optimization of scheduling algorithms and strategies to further elevate the utilization efficiency of heterogeneous AI computing power resources and the overall service quality.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Virtaitech. Software Defined AI Computing power Cloud Scheduling. Solution [J].CSDN. 2024,8-06
2. Zhengke Zhu. Prometheus Cloud Native Monitoring: Operation and development practice.[M]. Beijing: China Machine Press,2020.
3. James Turnbull. The Art of Surveillance: A surveillance Framework in the Cloud Native Age.[M]. Beijing: Posts and Telecommunications Press,2020.
4. Mike Julian. Monitoring operation and maintenance practice: Principles and strategies.[M]. Beijing: Posts and Telecommunications Press,2020.
5. VAN BON, JAN.ITIL Foundation: ITIL management practices.[M].VAN
6. HAREN PUBLISHING,2019.
7. Kai Zhu. Principle Analysis and Application Practice of ClickHouse.[M]. Beijing: China Machine Press,2020.
8. Guangke Wu. Actual Operation and Maintenance of Linux Enterprise [M].Beijing: Tsinghua University Press,2018.