Academic Education Publishing House

Industry Science and Engineering Vol. 3 No. 3, 2026

Research on Software System Fault Monitoring in Domestic Real-Time Operating Systems

Liang Deng

The 29th Research Institute, China Electronics Technology Group Corporation, Chengdu, Sichuan, China

Abstract: In recent years, domestic real-time operating systems (RTOS) have been increasingly applied in mission-critical fields such as aerospace, energy and power systems, and industrial control. In these fields, the completeness, real-time performance, and traceability of fault information logging have become key technical requirements to ensure system reliability. In this study, we systematically sorted out the current development status of fault logging technologies in domestic RTOS, focused on analyzing the fault recording mechanisms of several mainstream systems including AcoreOS, Kylin, ReWorks, and RT-Thread, and conducted in-depth exploration on core technologies such as in-memory logging, eBPF-based dynamic tracing, dynamic fault code registration, and multi-dimensional diagnostic tool matrices. Through actual experimental tests, we found that the in-memory logging service can reduce the record latency to 0.5% of the file system latency, while achieving a write throughput of 270 MB/s, which is 6 times faster than that of SSDs; the eBPF technology can realize panoramic kernel observation with a performance overhead of less than 1%, and can automatically capture abnormal phenomena such as sudden spikes in CPU syscall usage and accumulation of D-state processes; the dynamic fault code registration mechanism effectively improves the portability and scalability of the system. Based on comprehensive experimental data and technical comparison analysis, this study provides systematic reference opinions for promoting the improvement of fault diagnosis capabilities of domestic real-time operating systems.

Keywords: Domestic Real-Time Operating System; Acoreos; Kylin; Rework; In-Memory Logging; Ebpf; Dynamic Fault Code Regist-Ration; Multi-Dimensional Diagnostics

1. Introduction

Real-time operating systems (RTOS) play an irreplaceable role in critical fields such as aerospace, industrial control, energy, and rail transit. With the continuous advancement of China's independent research and development of operating systems, domestic RTOS such as AcoreOS, Kylin, ReWorks, RT-Thread, SylixOS, and Linux have been widely used in a large number of equipment related to the national economy and people's livelihood. This widespread application has put forward higher requirements for the reliability and robustness of software systems. When a software system fails, comprehensive and accurate recording of fault information is not only helpful for quickly locating the root cause of the fault, but also provides an important basis for subsequent system safety analysis and design improvement.

Traditional fault recording mechanisms mostly rely on file system logs, but in real-time systems, this method faces two prominent problems: on the one hand, the I/O operation of writing logs to the file system takes a long time, which may affect the determinism of real-time tasks; on the other hand, after the system crashes or resets, the log information that has not been written to the disk is easily lost. In recent years, with the rapid development of observability technology, new fault recording technologies such as in-memory logs, eBPF dynamic tracing, and intelligent diagnostic tool matrices have gradually become mature, which provides new ideas and methods for improving the fault diagnosis capabilities of domestic real-time operating systems [1].

The main purpose of this paper is to sort out the current status of fault information recording technology based on domestic real-time operating systems, analyze the design concepts and implementation mechanisms of different technical schemes through the comparison of experimental data, and provide useful references

for the development of fault monitoring and diagnosis technologies for real-time software systems.

2. Overview of Fault Recording Technology in Domestic Real-Time Operating Systems

2.1 Basic Requirements for Fault Records

For real-time operating systems, the fault recording mechanism needs to meet five core requirements in practical application, which are specifically explained as follows:

Real-time capability: The process of recording fault information should not have a significant impact on the real-time response of the system, and the recording delay must be controlled within an acceptable range. According to international relevant standards, the maximum interrupt latency of industrial-grade RTOS should not exceed 50 microseconds.

Non-volatility: After the system is reset or powered off, critical fault information should still be recoverable to facilitate post-event fault analysis and investigation.

Completeness: The recorded fault information should include the context information at the time of the fault, such as task status, register values, and call stacks, so as to comprehensively reflect the fault situation.

Scalability: It should support the dynamic registration of different types of fault codes, so as to adapt to different hardware platforms and application scenarios and meet the diverse needs of fault recording.

Observability: It should be able to continuously monitor the internal state of the system, and automatically capture the on-site information when an abnormality occurs, laying a foundation for fault diagnosis.

2.2 The Development Trajectory of Fault Recording Technology in Chinese-Developed Operating Systems

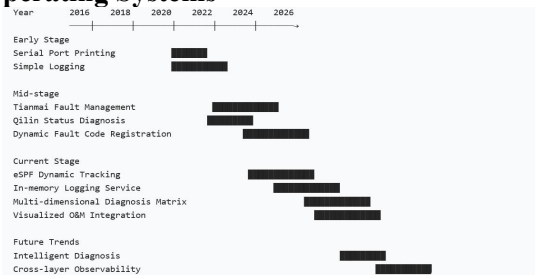

Figure 1. Gantt Chart of the Development Process of Fault Recording Technology in Domestic Real-Time Operating Systems

Table 1. Characteristics of Fault Information Recording Technology in Leading Domestic Real-Time Operating Systems

Operating System	Fault Recording Features	Application Fields	Key Technical Methods	Real-Time Index (Max Interrupt Latency)
AcoreOS	Dynamic registration of fault codes, fault playback	Airborne embedded equipment, military industry	ARINC653 standard, watchdog interrupt, abnormal context capture	Microsecond level
Kylin V11 (Galaxy Kylin)	Multi-dimensional fault diagnosis matrix, full-path analysis	Servers, cloud, desktops	eBPF, iodiag, netmaster, exmonitor	62.98μs after tuning (hard real-time edition)
ReWorks	Memory logging service, persistent hot restart	Embedded real-time systems	Memory file system, non-blocking writing, sleep-wake mechanism	Unpublished data
RT-Thread Smart	POSIX user mode, real-time performance visualization	IoT, industrial control, AIoT	ftrace, performance profiling tools, SMP scheduling optimization	2.4μs (no load) – 8μs (high load)
SylixOS	Low-latency scheduling, aviation-level certification	Rail transit, smart grid, aerospace	Real-time kernel, functional safety certification	Passed aviation-level certification
Wanghuo Real-Time Linux	Hard real-time, functional safety certification	Industrial control, automotive electronics	Kernel scheduling optimization, ISO 26262 ASIL B certification	Deterministic microsecond-level response
UnionTech UOS Server Edition	Dynamic data collection, visual traceback	Server operation & maintenance	Integration of hardware/software/log information, Youwo supervision platform	Overall leading performance in UnixBench tests

Through the investigation and analysis of the development process of domestic real-time operating systems, we found that the fault recording technology of domestic RTOS has generally gone through three development stages. In the early stage, it mainly relied on serial port printing and simple log files. This method had limited functions, and the fault information was easily lost, which had great limitations. In the middle stage, with the continuous improvement of technology, a structured fault management framework was gradually formed. For example, the AcoreOS operating system introduced mechanisms such as dynamic registration of fault codes and fault replay, which significantly improved the fault recording effect. At the current stage, the fault recording technology has integrated the concept of observability, and applied new technologies such as eBPF to realize low-overhead and panoramic fault information collection, which has greatly improved the fault diagnosis level.

To more intuitively show the development process of fault recording technology and the time distribution of key technologies in each stage, we have made the following Figure 1 (Gantt chart) and Table 1.

3. Analysis of Key Technologies and Implementation Mechanisms

3.1 Memory Log Service Technology

In embedded real-time systems, traditional file system logs have obvious performance bottlenecks, which cannot meet the real-time requirements of fault recording. To solve this problem, the ReWorks operating system has developed an in-memory log service. This service records exception information in a specially allocated memory area, so that even after the system is warm-rebooted, the log information before the reset can be retrieved, which effectively avoids the loss of fault information.

The in-memory log service of ReWorks adopts a three-layer design structure, and each layer undertakes different functions. At the memory management layer, the system pre-allocates a fixed-size log buffer and uses a circular overwrite strategy to ensure that the latest fault information can be retained. At the write mechanism layer, the log write operation bypasses the file system layer and directly operates the memory-mapped area, which avoids the context switching overhead caused by the I/O stack and greatly improves the write speed. At the persistence layer, relying on the sleep-wake mechanism of ReWorks, the key index information of the in-memory log is saved to the retention storage area before the system reset, ensuring the non-volatility of the key fault information [2].

We conducted a comparative test on the FT2000A/2 hardware platform (4-core CPU, 1.5 GHz). The test results show that compared with file logging, the in-memory logging service takes only about 0.5% of the time to record a complete error message. The write performance of the in-memory file system reaches 270 MB/s, which is six times that of the SSD-based file system. This obvious performance advantage is very important for real-time systems that require millisecond-level response. The specific test data are shown in Table 2.

Table 2. Comparison of Performance between Memory Logs and Traditional File Systems

Performance Indicator	In-Memory Logging Service	File System Logging (EXT4)	Performance Improvement Multiple
Single Log Write Latency	3.2μs	640μs	200x (0.5%)
Batch Write Throughput	270 MB/s	45 MB/s	6x
System Reset Retention Rate	100%	60%-95%	--
CPU Utilization	<1%	3%-5%	3-5x optimization

3.2 Dynamic Fault Capture Based on eBPF

The introduction of eBPF technology has brought a major breakthrough to the fault recording of operating systems, changing the traditional fault capture mode. The open-source HUATUO project developed by Didi, based on eBPF technology, has built a low-overhead and non-intrusive multi-dimensional kernel observation system, which provides a new way for fault diagnosis.

In addition, the fault diagnosis tool matrix of Kylin V11 also makes extensive use of eBPF technology, which can realize functions such as network packet loss tracing and file system metadata monitoring [3].

The working mechanism of eBPF-based dynamic fault capture can be divided into three levels, which are closely connected and form a complete fault capture system. At the kernel probe layer, dynamic tracing methods such as kprobe, tracepoint, and ftrace are used to set observation points on key kernel paths, so as to monitor the running state of the kernel in real time. At the data collection layer, eBPF programs run safely in the kernel space, only filter and record the context information related to anomalies, and do not need to copy all data to the user space, which reduces the performance overhead. At the trigger strategy layer, the system automatically judges whether to start detailed diagnosis according to the preset threshold. For example, when the CPU system-state usage suddenly spikes, D-state processes accumulate, or I/O delays are abnormal, the system will trigger the diagnosis mechanism.

The HUATUO project regularly collects nearly 300 operating system-level statistical indicators, covering core subsystems such as CPU, memory, network, and I/O. The built-in AutoTracing automatic trigger conditions are shown in Table 3.

Table 3. HUATUO AutoTracing Automatic Trigger Conditions

Trigger Type	Monitoring Index	Trigger Condition	Collected Content
CPUSYS	System-mode CPU Time	Usage > Threshold A or Unit-time Growth > Threshold B	System call stack flame graph, interrupt context
CPUIDLE	CPU Idle Time	Sudden CPU usage spike, abnormal Sys/User growth	Process call stack, container information
DLOAD	D-state Process	D-state process load exceeds threshold	Container operation status, D-state process information
MemBurst	Memory Allocation	Massive memory allocation in a short period	Memory allocation site, OOM-related information
IOTracing	I/O Bandwidth	High disk load, abnormal I/O latency	File name, path, device number, container name

According to the practical application experience of the HUATUO project, the performance overhead of regular indicator collection can be controlled within 1%. When an abnormal event is detected, the system will automatically capture on-site data such as call stacks, register states, and container information, and generate flame graphs for visual analysis. This design concept of "routine monitoring + anomaly triggering" well balances the performance overhead and information completeness, and has strong practical value.

3.3 Fault Code Dynamic Registration and Hierarchical Recording

The dynamic registration mechanism of fault codes allows applications and driver modules to register custom fault codes and their corresponding callback functions during the initialization phase. This design has two obvious advantages: on the one hand, the system does not need to predefine all possible fault types, which greatly improves the scalability of the system; on the other hand, different modules can independently maintain their own fault code tables, which reduces the coupling between modules and facilitates system maintenance and upgrade.

In terms of the hierarchical design of fault recording, AcoreOS 2 divides the recording into three levels according to the importance of fault information. The first level is the fault event summary, which records basic information such as fault type, timestamp, and occurrence location. The second level is the context snapshot, which includes the ID, priority, program counter, and register values of the currently running task. The third level is optional application-specific data, which is provided by the fault source through the registered callback function. Finally, the fault records are stored in a dedicated non-volatile storage area, and can be retrieved through the fault replay interface according to the timestamp or fault type [4].

In the practice of exception management based on AcoreOS 1 and the domestically developed 603e processor, we found that the system can fully capture the processor's runtime exception information and task execution information, and can support the testing and verification of exception management. This mechanism is of great significance for high-reliability application scenarios such as airborne embedded devices [5].

3.4 A Multi-Dimensional Fault Diagnosis Tool Matrix

The Kylin Advanced Server Operating System V11 has built a multi-dimensional fault diagnosis tool matrix, which covers the fault diagnosis of file systems, networks, and application performance. The design concept of this tool set is to change from single-point fault recording to end-to-end observability, so as to comprehensively improve the fault diagnosis efficiency.

For file system faults, kylin-iodiag-tools can capture system calls, file system, and block layer operations, and non-destructively record the entire life cycle of specified files. When a file anomaly occurs, it can trace back the entire operation history, which is convenient for locating the fault cause. Kylin-fs-safe focuses on metadata monitoring, which can comprehensively monitor potentially destructive file system behaviors without affecting business performance.

In terms of solving network faults, kynetobser uses eBPF technology to conduct full-path latency analysis, and uniformly associates the information of each layer of the protocol stack between the receiving and sending nodes.

Netmaster can track the entire process of packets from entering the network card to exiting the kernel, and quickly find the cause of packet loss.
For the correlation analysis between applications and systems, exmonitor can automatically record abnormal information when system indicators trigger alarms. It correlates application performance data with system data, making up for the "application-system" troubleshooting chain and improving the efficiency of fault location.
The evaluation and tuning tools of the Kylin Thor operating system go further. They identify real-time system anomalies by collecting abnormal process stack traces and detailed process information, and play an important role in finding problems affecting real-time indicators, such as scheduling delays, interrupt delays, maximum interrupt disable time, and maximum preemption disable time [6].

3.5 Fault Information Visualization and Operation and Maintenance Integration

The ultimate goal of fault recording is to help operation and maintenance personnel quickly locate and solve problems. Inspur Information KOS integrates the KSysOM visual operation and maintenance platform, which is developed based on the open-source project SysOM of Anolis. This platform integrates host management, monitoring and alarm, anomaly diagnosis and other functions into one, realizing the unified management of fault information.
In addition, the server operation and maintenance supervision platform also provides a "dynamic data collection" function. When a server anomaly occurs, it can immediately capture hardware, software, and log information, and present it in a visual form, which is convenient for operation and maintenance personnel to view and analyze.
The "dynamic data collection" function includes four core modules, which are specifically introduced as follows:
• Web interface display: It formats and displays abnormal information and power-on/power-off records, providing a clear and intuitive interface for operation and maintenance personnel to review.
• Hardware information collection: It monitors CPU usage, disk space, read/write performance, memory usage, etc., and records the hardware category and status when an abnormal event occurs.
• Software information collection: It monitors process ID, executed commands, process status, stack information, etc., and summarizes the feedback information when an anomaly occurs.
• Log information collection: It monitors key log content, especially focusing on abnormal information in newly added logs, so as to comprehensively grasp the fault situation.
The core value of visualization integration is to reduce the technical threshold of fault analysis. Traditional fault records are scattered in different log files, which requires experienced experts to conduct comprehensive analysis. The visualization platform unifies the presentation of multi-dimensional fault information, and displays the abnormal context through intuitive forms such as flame graphs and topology charts, enabling operation and maintenance personnel to quickly grasp the overall situation of the fault and improve the efficiency of fault handling.

4. Real-Time Performance Evaluation and Comparative Analysis

4.1 The Real-Time Performance between RT-Thread Smart and Zephyr

We conducted a comprehensive real-time performance comparison test between RT-Thread Smart and Zephyr real-time operating system (RTOS) on the Rockchip RK3566 development board (quad-core Cortex-A55, 1.8 GHz). In the test, we used the same test case code as the test tool to measure the comprehensive real-time performance indicators of the two RTOS under different load conditions. The measured indicators include the sum of three key indicators: interrupt response latency, asynchronous notification latency, and real-time task scheduling latency.
The test is divided into four load scenarios: no-load, CPU load (Dhrystone integer operations), I/O/interrupt load (IPI cross-core interrupts), and memory load (MBW memory bandwidth stress). The specific test results are shown in Table 4.
From the test data, we can see that RT-Thread Smart has an average latency of only 2.4μs in the no-load scenario, and the maximum latency is 8.2μs under high I/O/interrupt load, showing excellent real-time stability. In contrast, Zephyr has an extreme latency of up to 525μs under I/O/interrupt load, which indicates that Zephyr has a real-time performance bottleneck in high-frequency interrupt scenarios.

Table 4. Real-Time Performance Comparison between RT-Thread Smart and Zephyr (Unit: μs)

Test Scenario	RT-Thread Smart Min Latency	RT-Thread Smart Max Latency	Zephyr Min Latency	Zephyr Max Latency
No Load	2.4	3.2	2.8	4.5
CPU Compute Load	2.8	5.6	3.5	15.2
I/O/Interrupt Load	3.1	8.2	4.2	525.0
Memory Load	2.9	7.8	3.8	18.6

4.2 Comparison of Galaxy Thor Linux before and After Optimization

The Galaxy Thor OS team conducted systematic evaluation and optimization on the original chip Linux, Galaxy Thor Linux Server Edition, Embedded Edition, and High-Realtime Edition on the LubanCat-3 platform. The test environment was built under high load conditions: CPU utilization 75%-85%, memory utilization 75%, disk I/O utilization 95%, and network bandwidth utilization 98%.

The cyclictest test data before and after optimization are compared in Table 5.

After optimization, the maximum latency of the Galaxy Thor Linux High-Realtime Edition decreased from 109μs to 62.98μs, and the interrupt latency was optimized from 391μs to 366μs. Using the Galaxy Thor Evaluation and Optimization Tool, we identified the main interference factors, including the preemption of real-time tasks by other processes and resource competition. The tool can capture the interference process information when the scheduling delay exceeds 100μs, and generate flame graphs for visual analysis [7,8].

Table 5. Galaxy Thor Linux Cyclictest Test Data Comparison (Unit: ns)

Kernel Version	Tuning Stage	Min	Avg	Max	Interrupt Latency
rk3576-standard	Before Tuning	4552	11692	2912958	7641
	After Tuning	4552	11692	2912958	7641
Galaxy.server	Before Tuning	4550	11697	3810225	7054
	After Tuning	4376	12051	2001757	769
Galaxy.preempt	Before Tuning	5030	10016	267088	10718
	After Tuning	5105	9431	262264	671
Galaxy.rt	Before Tuning	5367	7976	109072	391
	After Tuning	5276	7831	62980	366

4.3 Comparison of RK Series Platform PREEMPT_RT and Xenoma

Huajian Intelligence conducted a 240-hour continuous test on two real-time solutions, PREEMPT_RT and Xenomai, on the Rockchip RK35XX series development boards. The test covered five platforms: RK3506, RK3562, RK3568, RK3576, and RK3588. The test was carried out under three scenarios: CPU idle, full load, and full load + core isolation.

The test results show that in terms of real-time performance, Xenomai is generally better than PREEMPT_RT. Taking RK3576 as an example, the maximum delay of PREEMPT_RT under full load is about 85μs, while Xenomai can control the maximum delay within 25μs. It is worth noting that RK3506 (Linux 6.1 kernel) has better real-time delay performance than the more powerful RK3568 (Linux 5.10 kernel). Through analysis, we believe that this is mainly due to the optimizations of the Linux 6.1 kernel in the scheduler, interrupt threading, and lock mechanisms [9].

5. Intelligent Fault Diagnosis Technology

5.1 Smart Diagnosis Based on Multi-layer Anomaly Analysis Model

Aiming at the problem that it is difficult to capture and identify occasional anomalies in the Linux kernel, we proposed a smart diagnosis method based on a multi-layer anomaly analysis and processing model. The core architecture of this method includes two parts: an anomaly knowledge rule library and a multi-layer anomaly analysis and processing model.

The anomaly knowledge rule library is used to collect and sort out common Linux kernel anomaly information, and provide possible causes and repair suggestions for each anomaly, which lays a foundation for rapid fault diagnosis. The multi-layer anomaly analysis and processing model extracts logs from kernel logs in a single round, identifies anomalies in each single-round log, marks the specific location of the anomaly,

and at the same time obtains the cause of the anomaly and repair suggestions from the anomaly knowledge rule library [10,11].

The innovation of this method lies in its ability to handle large-scale log data. When the number of log rounds exceeds a critical value, the system will split the logs and call multi-threaded parallel analysis, which can significantly reduce the time required for anomaly identification. In addition, the system supports extracting key data (such as the start and end time of each round of logs, stage duration, image allocation data). If the key data shows a monotonically increasing trend, it will be marked as an optimizable anomaly.

For the time-consuming data of the device, the system adopts a dual monitoring mechanism of boxplots and normal distribution: it calculates the upper limit of boxplot anomaly monitoring and the upper limit of normal distribution for the device. If the time-consuming data exceeds both upper limits, it is judged as an anomaly and marked. This statistical-based anomaly detection method can effectively identify the performance degradation trend of the device.

5.2 Continuous Performance Profiling and Full-stack Observability

The continuous profiling technology introduced by the HUATUO project can realize comprehensive performance analysis of the operating system kernel and applications. The profiling objects include multiple dimensions such as CPU, memory distribution, memory allocation, and lock contention, supporting multi-language and multi-perspective (thread, process, container, CPU, kernel subsystem) observations.

The core value of continuous performance profiling is mainly reflected in four aspects:

• Replayability: It can replay the performance status of the system at historical moments, which is convenient for retrospective analysis of faults.

• Comprehensive Coverage: It covers various system resources such as CPU, memory, I/O, and locks, realizing all-round performance monitoring.

• Low Overhead: It adopts a non-intrusive method to realize routine operation, which will not have a significant impact on the system performance.

• Language-agnostic: A unified storage structure is adopted to support multi-language applications, which has strong versatility.

In practical application scenarios such as link stress testing, fire drills, holiday protection, and performance profiling, continuous performance profiling technology has played a key role. For example, during high-traffic scenarios on holidays, the system can automatically capture anomalies such as application delays, I/O spikes, and CPU drops, and generate flame graphs to help technical personnel conduct root cause analysis.

6. Performance Analysis and Discussion

6.1 Comparison of Key Technology Performance

Based on the experimental data summarized above, we sorted out the performance indicators of different key technologies for fault recording, as shown in Table 6.

Table 6. Performance Comparison of Key Technologies for Fault Recording

Technology Type	Representative Product	Key Performance Indicators	Advantage Scenarios	Limitations
In-Memory Logging	ReWorks	Latency 3.2μs (0.5% of file system), throughput 270 MB/s	System crash recovery, real-time logging	Limited memory capacity
eBPF Dynamic Tracing	HUATUO	Performance overhead <1%, nearly 300 collected metrics	Full kernel visibility, automatic anomaly capture	Thresholds require empirical tuning
Real-Time Performance Evaluation	Galaxy Thor Tool	Identifies interference with scheduling latency >100μs	Real-time system tuning, performance bottleneck analysis	Requires expert experience
Intelligent Log Analysis	Multi-layer Anomaly Analysis Model	Parallel processing of massive logs, statistical anomaly detection	Occasional anomaly identification, performance degradation warning	Rule library relies on historical data
Visualized Operation & Maintenance	UnionTech Youwo	Integrated visualization of hardware, software and log information	Improved O&M efficiency, rapid fault location	Requires deployment of O&M platform

6.2 Trend Analysis and Future Directions

Through the above technical analysis, we can see that the fault information recording technology of domestic real-time operating systems is showing four obvious development trends:

From passive recording to active prediction. Traditional fault recording can only passively record information after a fault occurs, which is

not conducive to fault prevention. The continuous performance profiling based on eBPF and the anomaly detection based on statistics enable the system to issue early warnings during the performance degradation stage, realizing predictive maintenance and reducing the occurrence of faults.

From single-point observation to full-stack observability. The HUATUO project has realized the connection between the kernel and containers, and between applications and systems. The Galaxy Thor tool has realized the precise measurement of scheduling delays and interrupt delays, and the full-stack observability capability is gradually forming [12].

From manual analysis to intelligent diagnosis. The multi-layer anomaly analysis model integrating statistical anomaly detection and parallel processing has greatly improved the efficiency of large-scale log analysis through intelligent diagnosis technology, reducing the dependence on manual analysis.

From technical indicators to business value. The integration of the visual operation and maintenance platform has transformed fault information recording from a simple technical tool into a business support capability, lowering the technical threshold for operation and maintenance personnel and improving the efficiency of business support [13].

7. The Future Trends and Challenges

Looking forward to the future, the fault information recording technology of domestic real-time operating systems will face both new development trends and many challenges, which are specifically analyzed as follows:

Deepening of intelligent diagnostics: By combining machine learning algorithms, we can analyze historical fault data to learn abnormal patterns, so as to realize predictive fault analysis. At present, the rule-based anomaly detection still relies on expert experience, and the introduction of deep learning technology is expected to improve the accuracy and coverage of anomaly identification.

Enhancement of cross-layer correlation: It is necessary to connect the observation data of hardware, operating system, middleware, and applications to build an all-stack observable system. The HUATUO project has realized the correlation between kernel and container information, but it still needs to be further expanded to the full chain of distributed systems.

Promotion of standardization: It is necessary to promote the standardization of fault recording interfaces for domestic operating systems to reduce the cost of application adaptation. At present, the middleware adaptation rate of domestic RTOS is only 63% of that of international mainstream systems, and standardization will significantly improve the compatibility of the ecosystem.

Continuous evolution of low-overhead techniques: It is necessary to further optimize the performance overhead of dynamic tracing technologies such as eBPF, so as to realize universal deployment on resource-constrained devices. At present, eBPF applications in embedded scenarios still face challenges such as kernel version dependence and resource consumption.

Building a talent ecosystem: At present, there is a shortage of 120,000 embedded development engineers in China. Strengthening talent training and community development is crucial to the development of the domestic RTOS ecosystem.

8. Conclusion

This study systematically investigates the fault information recording technology based on domestic real-time operating systems. By analyzing the fault recording mechanisms of mainstream domestic operating systems such as AcoreOS, Kylin, ReWorks, RT-Thread, and SylixOS, we summarize the implementation principles and performance characteristics of key technologies such as memory logs, eBPF dynamic tracing, dynamic fault code registration, multi-dimensional diagnostic tool matrices, and integrated visualized operation and maintenance.

The research results show that domestic operating systems have formed a variety of technical routes in the field of fault information recording. The memory log technology has significantly improved the real-time performance of recording, with a single log write delay as low as 3.2 μs. The eBPF technology can realize panoramic observation with a performance loss of less than 1%, and automatically capture abnormal conditions such as CPU kernel state spikes, D-state process accumulation, and I/O delay anomalies. The dynamic fault code registration mechanism improves the portability of the system, while the layered recording design ensures the completeness of fault information. The multi-dimensional diagnostic tool matrix realizes the

transformation from single-point fault recording to full-chain observation. The integrated visualized operation and maintenance reduces the technical threshold of fault analysis.

References

[1] Hassan H E, Ibrahiem K H, Madian A H. Optimizing multiprocessor performance in real-time systems using an innovative genetic algorithm approach. Alexandria Engineering Journal, 2024, 83: 4567-4578.

[2] Zhang X H, Sun G X. Research on health monitoring technology in real-time operating systems. Aeronautical Computing Technique, 2005, 35(4): 66-67.

[3] Liu J. Design of State Diagnosis System Based on Domestic Kylin Operating System (Kylin OS). Electronic Enthusiast, 2018(06).

[4] Li N, Wang D D, Wang T. Design and implementation of fault recording scheme based on ACoreOS 3 multi-partition. Electronic Design Engineering, 2024, 32(15): 112-115.

[5] Wang Q. Design and Implementation of Exception Management for Domestic 603e Processor Based on AcoreOS-1. Computer Programming Skills & Maintenance, 2022(04).

[6] Singh J P. Comparative Analysis of Scheduling Algorithms for Latency Optimization in Real-Time Applications. International Journal of Science and Research, 2024, 13(2): 1179-1185.

[7] Li G, Wang W. Auxiliary Analysis Method and System for Occasional Abnormalities of Linux Kernel Based on Domestic Operating System. China Patent, 2024.

[8] Gao Y Q, Luo Z Y, Wang Y C, et al. Evaluation and Optimization of Operating System for Domestic Supercomputers. Computer Science, 2025, 52(5): 11-24.

[9] Hu Y, Fu L Y, Han H Y, et al. Optimization of Vehicle Operating System Scheduling Strategy Based on Improved LLF. Journal of Xihua University (Natural Science Edition), 2026, 45(1): 1-15.

[10] Ren H. Research on Performance Optimization of Linux Kernel Scheduler for Real-Time Scheduling. Information& Computer, 2026, 4:159-161.

[11] Huang C C J, Yang C F. An Empirical Approach to Minimize Latency of Real-Time Multiprocessor Linux Kernel//2020 International Computer Symposium (ICS). IEEE, 2020: 1-6.

[12] Han K, Kang Q, WANG K Y. Design and Implementation of Real-time Fault Diagnosis Function for Onboard Information System. Aeronautical Computing Technique, 2024, 54(2):88-91.

[13] Luo G, Mao H, Zhu Y S, et al. Real-Time Hybrid Task Scheduling Algorithm for Embedded Multicore Systems. Electronic Science and Technology, 2024, 37(8): 84-91.