Scientific Research Publishing

Atmospheric and Climate Sciences, 2017, 7, 367-373
http://www.scirp.org/journal/acs
ISSN Online: 2160-0422
ISSN Print: 2160-0414

Research on Exception of Meteorological Satellite Ground System Application Based on Resource Bottleneck

Xiangang Zhao, Manyun Lin, Weixia Lin*, Lizi Xie, Cunqun Fan

National Satellite Meteorological Center, BeiJing, China
Email: *linweixia@cma.gov.cn

How to cite this paper: Zhao, X.G., Lin, M.Y., Lin, W.X., Xie, L.Z. and Fan, C.Q. (2017) Research on Exception of Meteorological Satellite Ground System Application Based on Resource Bottleneck. *Atmospheric and Climate Sciences*, **7**, 367-373.
http://dx.doi.org/10.4236/acs.2017.73027

Received: June 26, 2017
Accepted: July 24, 2017
Published: July 27, 2017

Abstract

Meteorological satellite ground application system resources are limited. Abnormal satellite missions often lead to hopple of system resources. In order to analyze the problem, this paper presents an anomaly analysis method for meteorological satellite ground system based on resource bottleneck. Through the CPU, memory and I/O, several types of resources in-depth were analyzed to find the bottleneck caused by the problem, thus providing recommendations for application optimization. Experimental analysis shows that the proposed method can reasonably analyze the resource bottleneck of CPU, memory and I/ O, and draw a good conclusion. To solve the meteorological satellite application system application anomaly caused by the bottleneck of the problem, the application of optimization to a certain extent plays a positive role.

Keywords

Satellite, Ground Application System, Resource Bottlenecks, Application Exception

1. Introduction

Meteorological satellite application system is large and complex. Some applications need a lot of data, which often makes the ground system resources occur bottlenecks. And the emergence of these bottlenecks is often due to the application of the operation caused by abnormal. How to study the anomalies of meteorological satellite ground system through the bottleneck of resources has become an urgent problem to be solved in satellite application system.

Some of the results have been achieved in related research on application anomalies. Biswas S. [1] described Valor, a sound, precise, software-only region conflict detection analysis that achieves high performance by eliminating the costly analysis on each read operation that prior approaches require. Valor instead logs a region's reads and lazily detects conflicts for logged reads when the region ends. Velmourougan S. [2] proposed a set of generic good practices to be observed during each phase of the software development life cycle (SDLC) for establishing the application system with sound exception handling mechanism.

Hassan M. M. [3] proposed a state-of-the-art with respect to issues of importance concerning software testability and an important quality attribute: software robustness.

Sawadpong P. [4] proposed exception-based software metrics that are based on the structural attributes of exception handling call graphs. They empirically validate the proposed metrics through a case study of Hadoop Core using data mined from software repositories and defect reports.

Barbosa E. [5] proposed Exception Handling Policies Language (EPL), a domain-specific language to specify and verify exception handling policies.

Si Y. W. [6] proposed a run-based exception prediction algorithm to predict temporal exceptions in workflows. The proposed algorithm is divided into two phases, design-time and run time. At design-time, all possible runs are generated from a workflow and their estimated execution time and mapping probability are calculated. At run time, temporal exceptions are predicted by analyzing the runs. Abrantes J. [7] proposed a practical approach to preserve the exception policy of a system or a family of systems along with its evolution, based on the definition and automatic checking of exception handling design rules.

This paper addresses the anomalies of meteorological satellite ground system applications due to bottlenecks in resources, optimizing system applications.

2. Exception and Bottleneck Analysis

Exception is the concept of software system sense. It refers to a software system that behaves abnormally or resource occupancy is due to the inherent flaws in code design and implementation. Such as memory leaks, floating-point computational capability exceptions, IO read and write assignments, and so on [8] [9]. Usually an exception in the case of the interaction with the hardware, such as memory allocation, file read and write.

The bottleneck is the concept of the hardware system. It refers to the process of loading an application running in hardware resources affected by high uptime and operational efficiency of operations, due to the high demand for hardware resources beyond the existing hardware level [10]. Which leads to CPU scheduling bottlenecks, parallel computing power, IO performance bottlenecks and so on.

By analyzing and identifying anomalies and bottlenecks, it is possible to visually analyze the risk of downtime in the course of the operation. Through the specific analysis of specific bottlenecks to give the corresponding hardware optimization recommendations and software optimization recommendations.

2.1. CPU Resource Exception and Bottlenecks

1) CPU scheduling bottlenecks

It refers to the application of the thread for a long time in a queue state, or application to start the process or thread too much. CPU scheduling capacity is limited, CPU performance bottlenecks.

Check the strategy: CPU_SoftIRQ + CPU_IRQ (CPU context switch) showed a significant monotonically increasing trend.

2) CPU computing power bottlenecks

Multi-core CPU is long time 100% occupied, and IO is less. This shows that CPU computing power bottlenecks.

Check the strategy: Calculate the 25% time of the CPU_CPI (average CPU average number of clock cycles per instruction) Time concentration C25%. When C25% < 1/3 that CPU has the computing power bottleneck.

3) CPU parallel computing power utilization

The application only takes one CPU and does not extend to other CPUs. The application is only written in a single process mode, even if the other CPU load is low cannot be used.

Check the strategy: for all the CPU core occupancy rate, extract the highest average of the five CPU core for the calculation of variance. When the calculated variance is that the CPU core in most of the time occupancy rate distribution of discrete trends, CPU parallel computing power shows utilization.

2.2. Floating Point Computing Exception and Bottlenecks

1) Floating point operation bottlenecks

CPU floating point computing power is not sufficient to meet the application requirements.

Check the strategy: CPU_All_Flops (CPU floating point calculation occupancy rate) maximum $M \geq 70\%$. CPU floating point calculation occupancy rate is too high. That is, floating-point computing capacity bottlenecks.

2) Floating point calculation is exploited exception

The target job is a floating point intensive computing application. CPU floating point computing power is not fully utilized.

Check strategy: CPU_All_Flops (CPU floating point calculation occupancy rate) Mean $E \leq 5\%$. That is considered to floating point computing power of an exception.

2.3. Memory Resource Exception and Bottlenecks

1) Memory leak exception

Memory occupancy continues to grow, there is no stable line, and there is a potential risk of memory leaks.

Check strategy: Mem_All_MemRatio occupancy rate showed a clear upward trend and an increase of more than 10%. That is, the risk of leakage of memory.

2) Memory allocation bottlenecks

CPU idle time is longer. Memory occupancy rate is still in a high state. CPU waits for memory toggle. And there are memory bottlenecks.

Check the strategy: When the following conditions are met at the same time, that there is a bottleneck in memory allocation.

a) CPU_All_Idle (CPU idle occupancy) showed a clear upward trend and an increase of more than 10%;

b) Mem_All_MemRatio (physical memory occupancy rate) has a longer time (more than 15 seconds) remains above 90%.

2.4. IO Resource Exception and Bottlenecks

1) IO read and write assignments exception

CPU occupancy rate is negatively correlated with IO read and write rate. When the CPU is occupied, the system does not have the file to read and write. While the application is reading and writing files without synchronizing CPU execution. Both do not make full use of resources at the same time

Check the strategy: CPU_All_Sys + CPU_All_User (active CPU usage) and Disk_All_Read + Disk_All_Write (IO read and write total) showed a significant negative correlation.

2) IO resource bottlenecks

The waiting time for reading and writing data requests continues to increase. Multiple processes compete for IO resources at the same time.

Check the strategy: Disk_All_SeqWait = Disk_All_Await − Disk_All_Svctm (IO waiting queue total time - IO average service time) showed a significant increasing trend.

3) IO performance bottlenecks

Disk read and write rates cannot keep up with task requests. Disk read and write tasks queued too long.

Check the strategy: Disk_All_Avgqu (IO queuing length) shows a clear increasing trend.

3. Analysis of Exception and Bottleneck

3.1. CPU Parallel Computing Power Utilization

As shown in Figure 1, the application's five CPU cores (core 0, 4, 12, 20, 28) during the application run, each core CPU active occupancy in three time periods are in a discrete distribution status. That is, the operation of these operations are not evenly shared in the CPU core. Such as in 20:52:10-20:53:02 time period only a CPU core occupancy rate is higher, and other CPU core is idle state. Ideally, several CPU cores with a high primary occupancy rate should maintain a more consistent and averaged CPU activity during application run. So while nearly half of all CPU cores are at 100% occupancy, there is still a risk that the application may have insufficient CPU computing power.

As shown in Figure 2, at least one CPU core is occupied by almost zero at almost every time point before the application runs the entire run time, especially before 20:54:37. And with the change of time, low CPU core is in the ever-changing (such as 20:45:46 before the core of the low occupancy rate of 36, and about 20:50:00 is the core of the core occupancy rate of 8). This indicates that the target application is in the process of frequent CPU core switching.

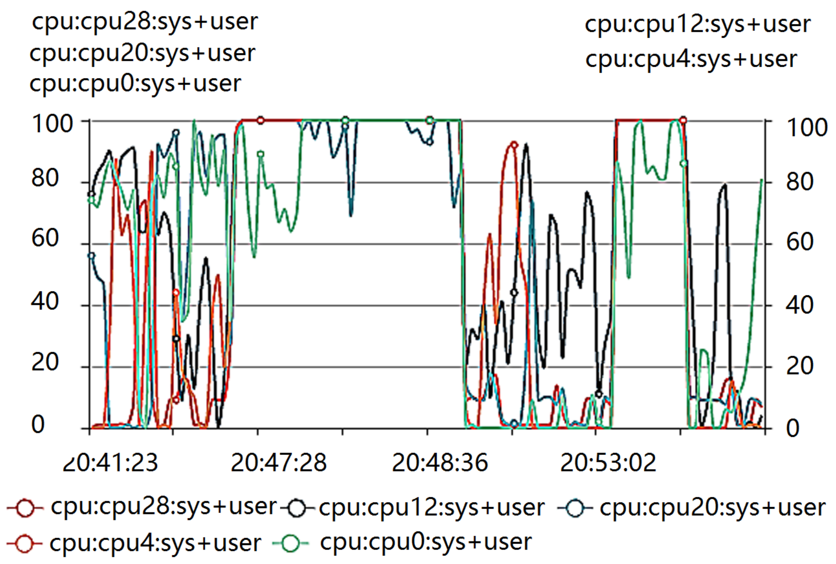

Figure 1. Case 1 of CPU parallel exception.

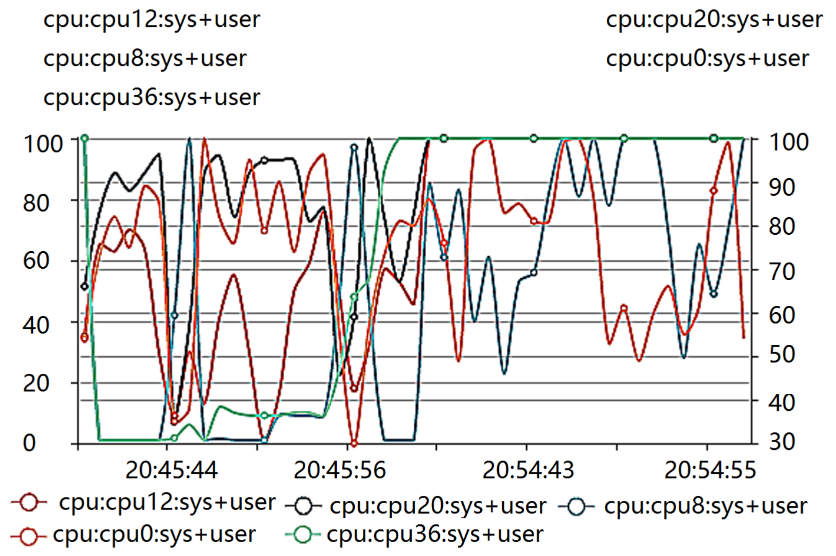

Figure 2. Case 2 of CPU parallel exception.

3.2. IO Resource Bottlenecks

As shown in Figure 3, the blue line represents the IO average service time (Disk_All_Svctm). The red line is the average IO wait time (Disk_All_Await). It can be seen from the figure, in fact, the blue line is in a slow upward trend and a large range of jitter. Although the red line as a whole did not change the magnitude, did not show a clear downward trend. Even in the late IO average service time has a downward trend but the red line still maintained before the level. This shows that IO requests more. Although the performance of the disk itself to ensure that the request queue is no significant growth trend, IO service speed is still unable to effectively ease the IO wait time delay. So it can be considered multiple processes at the same time in competing IO resources, and there is a IO resource bottleneck.

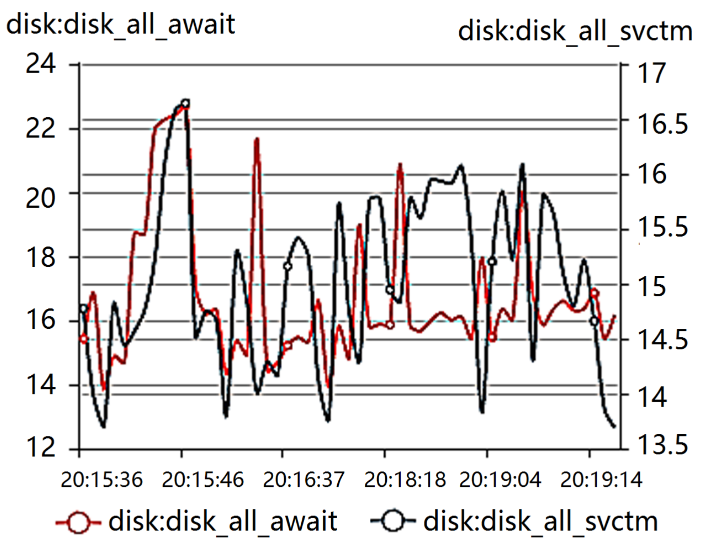

Figure 3. Case of IO resource bottlenecks.

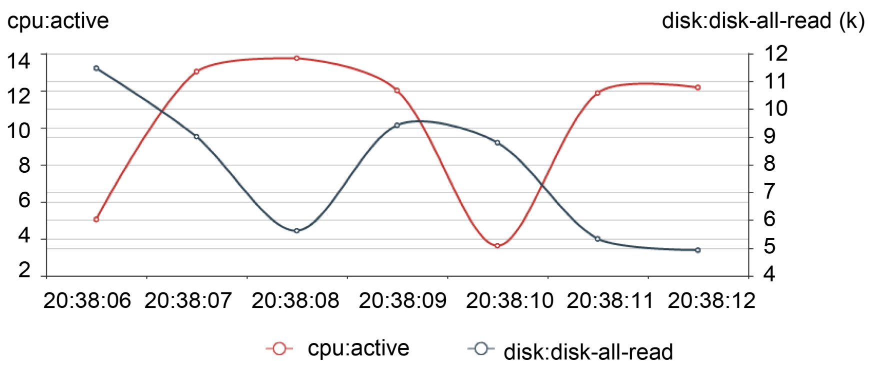

Figure 4. Case of IO read and write assignments exception.

3.3. IO Read and Write Assignments Exception

IO read and write assignments exception example shown in Figure 4. Obviously, during this time the CPU active occupancy (CPU_Active) and IO read rate showed a significant negative correlation. You can think of an IO read and write assignment exception. Considering the short duration of the exception, it should be necessary to refer to the distribution of CPU and IO related parameters before and after the time period.

4. Conclusion

This paper analyzes the resource bottlenecks caused by the abnormal application of meteorological satellite application system. This paper presents an anomaly analysis method for meteorological satellite ground system based on resource bottleneck, analysis of common abnormal state of several kinds of resources from system. And through the CPU parallel computing power and I/O read and write the distribution of experimental analysis, found that the distribution of resources of the two resources, which plays a positive role in some extent on the application of optimization.

Acknowledgments

The work presented in this study is supported by National High-tech R&D Program (2011AA12A104).

References

[1] Biswas, S., Zhang, M., Bond, M.D., *et al.* (2015) Valor: Efficient, Software-Only Region Conflict Exceptions. *ACM SIGPLAN Notices*, **50**, 241-259. https://doi.org/10.1145/2858965.2814292

[2] Velmourougan, S., Ponnurangam, D. and Ramachandran, B. (2015) Software Development Life Cycle Model to Inculcate Exception Handling. *International Journal of Computer Aided Engineering & Technology*, **7**, 401-420. https://doi.org/10.1504/IJCAET.2015.071300

[3] Hassan, M.M., Afzal, W., Blom, M., *et al.* (2015) Testability and Software Robustness: A Systematic Literature Review. 2015 41*st Euromicro Conference on Software Engineering and Advanced Applications*, Funchal, 26-28 August 2015, 341-348. https://doi.org/10.1109/SEAA.2015.47

[4] Sawadpong, P. and Allen, E.B. (2016) Software Defect Prediction Using Exception Handling Call Graphs: A Case Study. *IEEE, International Symposium on High Assurance Systems Engineering, IEEE*, Orlando, FL, 7-9 January 2016, 55-62. https://doi.org/10.1109/HASE.2016.13

[5] Barbosa, E., Garcia, A., Robillard, M., *et al.* (2016) Enforcing Exception Handling Policies with a Domain-Specific Language. *IEEE Transactions on Software Engineering*, **42**, 559-584. https://doi.org/10.1109/TSE.2015.2506164

[6] Si, Y.W., Hoi, K.K., Biuk-Aghai, R.P., *et al.* (2016) Run-Based Exception Prediction for Workflows. *Journal of Systems & Software*, **113**, 59-75. https://doi.org/10.1016/j.jss.2015.11.024

[7] Abrantes, J. and Coelho, R. (2015) Specifying and Dynamically Monitoring the Exception Handling Policy. *The International Conference on Software Engineering and Knowledge Engineering*, Taipei, 24 November 2015, 370-374. https://doi.org/10.18293/SEKE2015-133

[8] Sundaresan, V. and Voldman, A.H. (2016) Prevention of Classloader Memory Leaks in Multitier Enterprise Applications. U.S. Patent 9,229,744.

[9] Hyojin, C., Chulwoo, P., Kang, U., *et al.* (2016) Semiconductor Memory Device with Operation Functions to Be Used during a Modified Read or Write Mode. U.S. Patent 9,292,425.

[10] Chan, T.S.C., Zhu, W., Cho, J., *et al.* (2017) Data Storage Device Increasing Sequence Detector Clock Frequency When Bottleneck Condition Is Detected. U.S. Patent 9,619,379.