Scheduler Aware Algorithm for Operating System Support for Java Virtual Machine

Shyamapriya Chowdhury[1], Sudip Chatterjee[2]

[1]Department of Information Technology, Narula Institute of Technology, India
[2]Department of Computer Science, Ideal Institute of Technology, India

ABSTRACT

Java Virtual Machine is the interpreter of byte code. When a programmer writes a java program having .java extension it is called source code. After compilation of source code byte code is generated which has .class extension. This byte code is then interpreted by java virtual machine(JVM) to produce machine code which is actually responsible for generating output. In this paper we have discussed about the internal architecture of JVM. This actually describes the procedure of java objects going into the java heap. JAVA security is also discussed in this paper and process of JAVA program execution is also discussed in this paper.

Keywords : JVM, JIT, JRE, JAVA heap, sandbox security, garbage collection, Virtual Memory Manager

1. INTRODUCTION

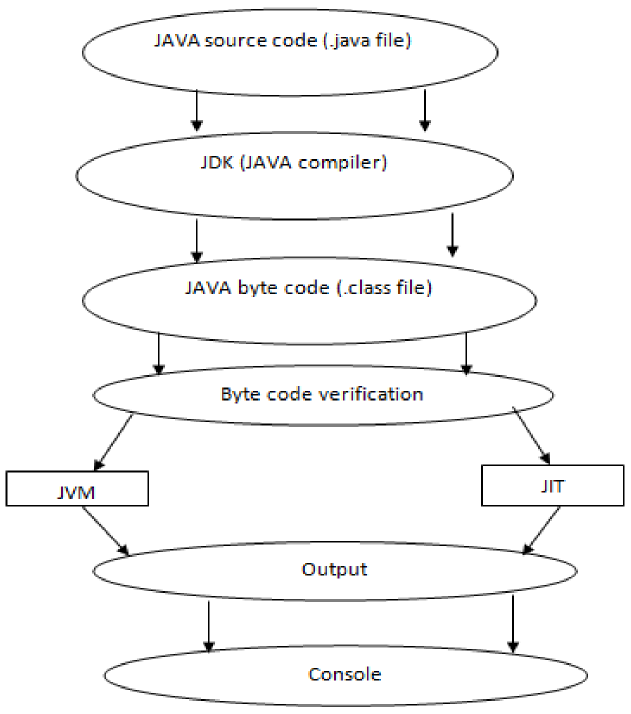
JVM stands for Java Virtual Machine. JVM is platform dependent; it can be available for different hardware and software platforms. Java virtual machine actually provides a virtual execution environment for JAVA applications. JVM employs some major runtime subsystems such as garbage collection, just-in-time compilation to maintain safety and efficiency of the execution program.JAVA is very secure due to its sandbox security model. The local code is directly interacting to JVM but remote code first interacts to the sandbox security. Then this sandbox security checks whether this code is reliable or not. If the code is found reliable then only it gets chances to interact with java virtual machine otherwise the code gets restricted.

JVM mainly performs four tasks:

Loading of codes
Verification of codes
Execution of codes
Virtual runtime environment

To reduce overhead of JAVA runtime systems, some methods have been applied inside Java Virtual Machines to tune performance through utilizing runtime information. To reduce the frequency of garbage collections heap usage information can be reused dynamically. To support highly scalable and efficient java application servers a new technology trend has been introduced. Function of this technology is to make interconnection between java virtual machine and operating systems. Improving efficiency of runtime system with hardware supports is also a goal of this new technology.

1.1 Hierarchy of JAVA Programs

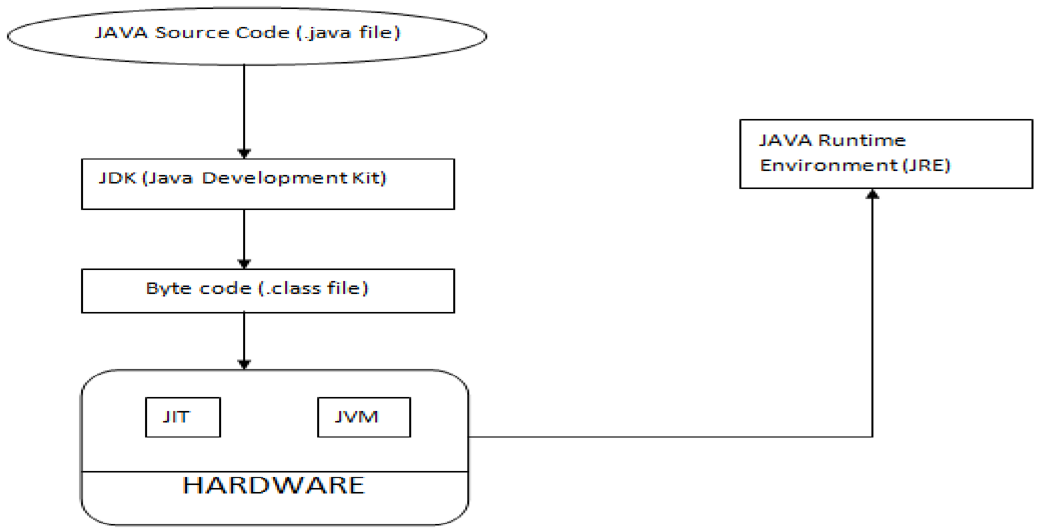


2. PROCESS

In case of java programming language, all source codes are first written in plain text files or in DOS with .java extensions. These .java files are converted into .class files with the help of java compiler. A .class file does not contain the code that is native to your processor; it instead contains byte code which is the machine language of java virtual machine because of the java virtual machine available on many different operating systems. After the successful compilation of source code byte code is generated this is platform independent. Finally the byte code is interpreted through java interpreter (JVM) line by line which is very slow.

Sun Microsystems has introduced a new compiler JIT(Just in Time) to overcome this problem. This compiler executes whole instruction in one goes. JRE is java runtime environment which actually implements JVM. JVM contains byte code checker which checks byte code. After successful verification of byte code by the checker, it executes the right code through JRE.

2.1 Process Diagram of JDK and JRE


JAVA Source Code (.java file)
JDK (Java Development Kit)
Byte code (.class file)
JIT
JVM
HARDWARE
JAVA Runtime Environment (JRE)

3. INTERACTION WITH VIRTUAL MEMORY MANAGER

In case of garbage collection heap size is very important. If it is too large then it could acquire paging overhead. Again if it is too small then garbage collection will take place very frequently which causes long pause time. For the above reason, to improve garbage collection performance some researchers extended Virtual Memory Manager to interact with Java Virtual Machine. Page information can be collected from Virtual Memory Machine. So recent garbage collector systems introduce methods to better coordinate with virtual memory manager to guide heal resizing or garbage collector triggering decisions. There are some representative works which actually extends virtual memory manager of operating system for garbage collector during execution time. These are Bookmarking Collector, Cooperative Robust Automatic Memory Management, and Yield Predictor.

The bookmarking collector is a garbage collector approach which is tightly coupled with virtual memory manager of the operating system. Its main function is to minimize the paging overhead when memory pressure is very high. The Cooperative Robust Automatic Memory Management is a framework which dynamically adjusts heap size through interaction between virtual memory manager and java virtual machine.

The yield predictor provides a simple solution to improve garbage collector efficiency by estimating the number of dead states. It uses the information of hardware page reference bits used by operating system virtual memory manager.

4. OS SCHEDULER AWARE OF JVM ACTIVITIES

Modern OS schedulers are designed to maintain effectiveness and equality .For this reason they can’t distinguish whether a thread is executing application logic or performing java virtual machine functions. So they treat them equally. Some researchers have built some specific operating system scheduler framework which is aware of JVM activities.

Allocation phase aware scheduler
Content aware scheduler
JIT aware scheduler

Allocation phase aware scheduler uses object allotment information to direct thread scheduling decisions. Time based round robin is replaced with memory based round robin in this aware scheduler. Memory based round robin is nothing but a policy that regulates the amount of memory allocated by each thread during its turn on the CPU.

Content aware scheduler develops object synchronization information to reduce the occurrences of lock convoys in large application servers executing in any multiprocessor system. In JVM, locks are maintained through the monitor mechanism.
JIT aware scheduler is a modified scheduler to give higher priority and a longer time slice to compiler thread so that it can have much time to complete the compilation avoiding extensive execution delay. The problem is round robin scheduling affects compilation performance significantly.

CONCLUSION

Java virtual machine provides a virtual execution environment for java applications. In this paper we have understood the execution procedure of JVM. We have also understood the java security through sandbox security model. JVM needs to employ runtime systems like garbage collections, Just in time compilation and lock management to maintain safety and efficiency.

REFERENCES

[1]. Azul Virtual Machine. http://www.azulsystems.com/technology/avm
[2]. M. Hertz, S.M. Blackburn, J. E. B. Moss, K. S. Mckliney, D. Stefanovi’c. “Generatingobject lifetime traces with Merlin”. Transactions on Programming Languages And Systems (TOPLAS) 28, 3(May), 476–516.
[3]. M. Hertz. Quantifying and Improving the Performance of Garbage Collection. VDM Verlag, Saarbr¨ucken, Germany.
[4]. M. Hertz, E. D. Berger. “The performance of automatic vs. explicit memory management”. In Proceedings of the 2005 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA 2005), Volume 40(11) of ACM SIGPLAN Notices (San Diego, CA, Oct. 2005), pp.313–326.
[5]. M. Hertz, Y. Feng, E. D. Berger. “Garbage collection without paging”. In Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2005), Volume 40(7) of ACM SIGPLAN Notices (Chicago, IL, June 2005), pp. 143–153.

[6]. M. Hertz, N. Immerman, J. E. B. Moss. “Framework for analyzing garbage collection”. In R. BAEZA-YATES, U. MONTANARI, AND N. SANTORO Eds., Foundations of Information Technology in the Era of Network and Mobile Computing: IFIP 17th World Computer Congress - TC1 Stream (TCS 2002), Volume 223 of IFIP Conference Proceedings (Montreal, Canada, 2002), pp. 230–241.

[7]. M. Hertz, S.M. Blackburn, J. E. B. Moss, K. S. Mckliney, D. Stefanovi’c. “Error free garbage collection traces: How to cheat and not get caught”. In Proceedings of the International Conference on Measurement and Modeling of Computer Systems, Volume 30(1) of ACM SIGMETRICS Performance Evaluation Review (Marina Del Rey, CA, June 2002), pp. 140–151.

[8]. M. Hertz, J. Bard, S. Kane., E. Keudel, T. Bai, X. Gu, C. Ding.” Waste not, want not: Resource-based garbage collection in a shared environment”. Technical report TR–951, University of Rochester. 2010.

[9]. M. Hertz. Quantifying and Improving the Performance of Garbage Collection. Ph.D Dissertation. University of Massachusetts Amherst.

[10]. F. Xian, W. Srisa-an, H. Jiang. “Allocation-phase aware thread scheduling policies to improve garbage collection performance”. In Proceedings of the 6th international Symposium on Memory Management (Montreal, Quebec, Canada, October 21 - 22, 2007). ACM, New York, NY, 79-90.

[11]. F. Xian, W. Srisa-an, H. Jiang. “Garbage collection: Java application servers' Achilles heel”. Sci. Comput. Program. 70, 2-3 (Feb. 2008), 89-110.

[12]. F. Xian, W. Srisa-an, H. Jiang. “Contention-aware scheduler: unlocking execution parallelism in multithreaded java programs”. In Proceedings of the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications(Nashville, TN, USA, October 19 - 23, 2008). ACM, New York, NY,163-180.

[13]. M. Wegiel, C. Krintz. “Dynamic prediction of collection yield for managed runtimes”. In Proceeding of the 14th international Conference on Architectural Support For Programming Languages and Operating Systems (Washington, DC, USA, March 07 - 11, 2009).ACM, New York, NY, 289-300.

[14]. M. Wegiel and C. Krintz, “X Mem: Type-Safe, Transparent, Shared Memory for Cross-Runtime Communication and Coordination”, ACM Conference Programming Language Design and Implementation (PLDI), Jun, 2008 (PLDI), Mar, 2008.

[15]. M. Wegiel and C. Krintz, “The Mapping Collector: Virtual Memory Support for Generational, Parallel, and Concurrent Compaction”, ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Mar, 2008.

[16]. M. Weigel and C. Krintz, Cross-Language, Type-Safe, and Transparent Object Sharing For Co-Located Managed Runtimes ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), 2010.

[17]. P. Kulkarni, M. Arnold, M. Hind. Dynamic compilation: the benefits of early investing. In Proceedings of the 3rd international Conference on Virtual Execution Environments (San Diego, California, USA, June 13 - 15, 2007). ACM, New York, NY, 94-104.

[18]. L. Zhang, C. Krintz, and P. Nagpurkar, Supporting Exception Handling for Futures in Java, ACM International Conference on the Principles and Practice on Programming in Java (PPPJ), Sep, 2007.

[19]. L. Zhang, C. Krintz, and P. Nagpurkar, Language and Virtual Machine Support for Efficient Fine-Grained Futures in Java, The International Conference on Parallel Architectures and Compilation Techniques, (PACT) Sep, 2007.

[20]. T. Yang, T. Liu, E. D. Berger, S. F. Kaplan, and J. E. B. Moss."Redline: First Class Support of Interactivity in Commodity Operating Systems". In Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation.

[21]. T. Yang, E. D. Berger, S. F. Kaplan, J. E. B. Moss. “CRAMM: virtual memory support for garbage-collected applications”. In Proceedings of the 7th Symposium on Operating Systems Design and Implementation (Seattle, Washington, November 06-08, 2006). USENIX Association, Berkeley, CA, 103-116.

[22]. T. Yang, M. Hertz, E. D. Berger, S. F. Kaplan, J. E. B. Moss: Automatic heap sizing: taking real memory into account. In Proceedings of the 4th international Symposium on Memory Management. Vancouver, BC, Canada, Oct 24-25, 2004. 61-72

[23]. F. Xian, W. Srisa-an, H. Jiang. “Micro Phase: Proactively Invoking Garbage Collection for Improved Performance”. In Proceedings of ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'07), Montreal, Canada. Oct 21-25.

[24]. F. Xian, W. Srisa-an, C. Jia, H. Jiang. "AS-GC: An Efficient Generational Garbage Collector for Java Application Servers". In Proceedings of 21st European Conference on Object-Oriented Programming (ECOOP'07), Berlin, Germany. July 30-Aug 03, 2007.

[25]. F. Xian, W. Srisa-an, H. Jiang. "Investigating Throughput Degradation Behavior of Java Application Servers: A View from Inside a Virtual Machine". In Proceedings of ACM International Conference on Principles and Practices of Programming In Java (PPPJ'06), Mannheim, Germany, Aug 30-Sep 1, 2006, Page 40-49.

[26]. F. Xian, W. Srisa-an, H. Jiang. "Fortune Teller: Improving Garbage Collection Performance in Server Environment using Live Objects Prediction". In Proceedings of ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications, San Diego, CA. Oct 17, 2005. ACM Press, Page 246-247, 2005