ISSN : 0974 - 7435 Volume 10 Issue 19

2014

BioTechnology

An Indian Journal

FULL PAPER

BTAIJ, 10(19), 2014 [11419-11425]

Research on the application of Java virtual machine in the distributed thread migration

Wang Chao[1], Li Qiang[1], Lian Yingjie[2]

[1]College of Information Science & Technology, Agricultural University of Hebei, Baoding 0710012, (CHINA)

[2]Computer Center of Affiliated Hospital of Hebei University, Baoding, 071000, (CHINA)

ABSTRACT

With the rapid development of electronic information science and technology, the Java virtual machine technology is becoming more and more important, its role is becoming more and greater at the same time, the role of the distributed algorithm is becoming more and greater, too. this study analyzes and studies the Java language and distributed architecture, on the basis of which it conducts the combined research of the Java virtual machine and the distributed computing,, and, in turn, designs the distributed computing architecture of Java virtual machine based on thread migration. By this way the prototype of the Java virtual machine is realized, the distributed thread migration mechanism of Java virtual machine is also realized; finally, the distributed applications of Java virtual machine migration online are summarized and prospected. This study not only summarizes the developing status of distributed system and Java virtual machine, but also proposes a new idea that Java virtual machine realizes the distributed computing in a thread migration, on the basis of which this study puts forward distributed Java virtual machine controlled by single control node where control node takes charge of the management of virtual machine, while the general node is responsible for the execution of the thread. It allows multiple threads' concurrent execution, as well as the intercommunication between various nodes. Object module heap can carry out object access of physical node whether remote or not, and has a unified interface. Threading module is mainly responsible for creation and migration of thread, and is also responsible for balance in the allocation of task at the same time.

KEYWORDS

The Java virtual machine; Online migration; Distributed computing, Object and thread.

INTRODUCTION

With the continuous development of network technology, the application of the Java virtual machine has been developed considerably. Java virtual machine has the characteristic for the users to shield the implementation details of the underlying computer, which is very favorable for developing distributed systems. But underlying heterogeneity is in the best interest to the distributed system based on Java virtual machine. The underlying heterogeneity has the calculating device, different storage structure, different operating systems and other features capable of adapting to diverse systems[1][2]. In the process of implementation, the upper layer of Java virtual machine builds a unified programming interface, so the program doesn't need to care for the underlying implementation details at runtime. The technology of Java virtual machine can better explain the logical relationship between the parts, which can achieve the function only by a unified interface without carrying out some special systematic designs for different businesses, which can be achieved only by a unified interface.

Concurrent with this, distributed computing is also becoming more and more important. Distributed computing plays an indispensable role in high-performance computing, mass data processing, large-scale server-side technologies, and cloud computing and other fields. It can play a bigger role by shielding implementation details on the bottom layer based on Java virtual machine's cross platform and can be comprehensively utilized.

JAVA VIRTUAL MACHINE

A good deal convenience brought about by Java virtual machine makes itself more and more important, and become a universal computing platform. up to now, the Java virtual machine can run more and more programming and scripting languages, which run well because of the excellent characteristics of the Java virtual machine[3][4]. Usually, this kind of operation will have two ways, one is for implementation using the interpreter, and the other one is specialized compiler, which can compile the language into the Java code to be thus carried out. These languages are as shown in TABLE 1.

TABLE 1 : Programming and scripting languages running on Java virtual machine

Implementation Mode	Language	Application Area
Implementation interpreter using Java language	Clojure	General Functional Programming
	JRuby	Can performe Ruby in Java program.
	Jython	Canperform Python in Java program.
	Rhino	Javascript execution engine of Java implementation
	Fantom	Aims at creating an abstract layer which can be implemented on both JRE and CLR.
specialized compiler to be used for compiling Java byte code	Processing	Visual effect programming language for the Java-like syntax
	GROOVY	Object -Oriented dynamic scripting language
	Scala	Combined with functional programming model and Java object - oriented model, being designed to try to improve some of the bad designs of Java
	Gousu	A generic JVM language, which be used in a Web application framework Ronin, building tools Vark and other open source projects, and also be used in the software of insurance industry in Guidewire Software company.
	JavaFX Script	A kind of script language created for Rich Internet applications

The design of the Java virtual machine is very flexible without fixed format and process. Different Java virtual machines can be designed adapting to underlying changes according to the different scenarios, by which the code's seamless migration can be achieved generally needless of varying the upscale application interface once again. there is no fixed format and processes dedicated to the design of the Java virtual machine, with a strong sense of flexibility, can be different according to the different scenarios designed to adapt to the changes of the underlying Java virtual machine, and to the upper layer application interface generally don't need to make other changes to achieve seamless migration of the Code.

DISTRIBUTED COMPUTING

The so-called distributed system means the system made up of several autonomous computers, where the communication between the nodes can be achieved through a network. These nodes will operate for a common task. the distributed computing[5][6] has two common scenarios : The first one is that nodes of more than one computer are required to involve due to the certain features of the application itself, to share the data across the network. The second is that the combination of multiple computers will have greater advantage in completing some tasks than single computer.

Common distributed architecture mainly includes the following modes: the first one is c/s model, which is widely used, such as the common distributed Web services. The customers mainly use it to obtain the data from the server. The second is closely coupled system, which means multiple subtasks are operating, and finally the operating results are combined. The third is unified addressing, which means that a virtual and unified address space is created so that the data can be copied each other. The fourth was a 3- story architecture, which means that an interlayer is created at both client side and the server side again to complete some corresponding service so as to simplify the deployment of the client side and streamline the client side. The fifth is P2P system, which does not have a special node to manage the network and computers on the network where tasks are evenly distributed in each node.

DESIGN CONCEPTS

According to the characteristics of the java virtual machine and distributed computing, this study is expected to design a new type of distributed virtual machine, which can achieve the migration of the concurrent task and concurrent execution. Fully transparent distributed facilities[7] are required to provide to make it become distributed computing platform by design thinking. it should not only enable the programmers to write, release and implement exactly like in a normal Java virtual machine, but also makes the unmodified legacy code operate directly. Code dynamic deployment mechanism is thereafter designed.

This system requires to completely keep the Java programming model, for which a distributed Java heap is needed so that all nodes can access the distributed environment of the local Java heap, after which they can access overall object, realize the synchronization of data reading and writing, to ensure consistency and integrity of the object the system should also be simple, using only a control node, and control node is responsible only for cluster management not involved in the running of Java code.

This system belongs to the above-mentioned closely coupled system in the distributed architectures in which the communication between the nodes of the architecture is very frequent, requiring to enhance the throughput capacity between nodes, and introduce asynchronous response mechanism for the requests unable to respond. The final model is shown in Figure 1 below.

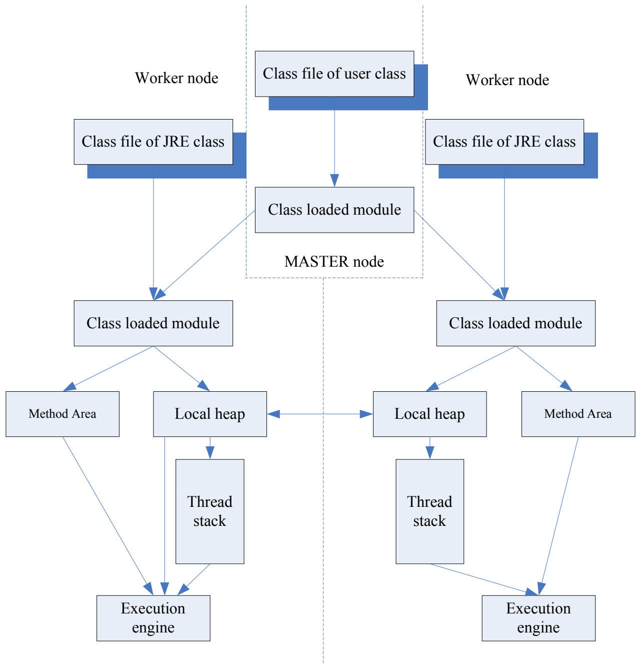

Figure 1 : Architecture of distributed Java virtual machine

COMMUNICATIONS

As a distributed system, communication is particularly important. Communication module is mainly responsible for data communication between nodes, masking off the implementation details of network communication, provides network communications services for upper tier object module and threading module. The implementation of this module realizes the point-to-point communication mainly using the socket of the Linux system according to TCP, and realizes the communication between groups using UDP protocol.

Communication mode mainly utilizes the modes of request and answer. The request can be sent between nodes while a multicast can implement more than one answer. Here we set up a control node (master) which corresponds to the listening port of 10000, while other nodes can independently choose the listening port, reporting to the control node only when registering. After completing registration of the node, the node must use a fixed port, prior to which the port is random when the node sendsrequest, see the following Figure 2 for specific procedures:

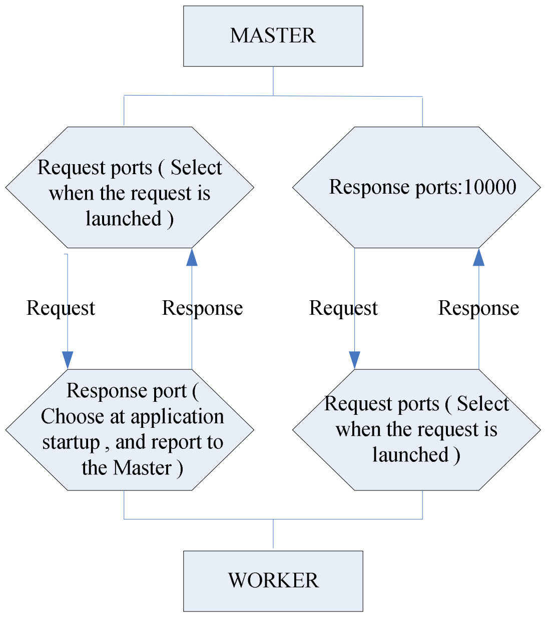
MASTER
Request ports (Select when the request is launched)
Response ports:10000
Request
Response
Request
Response
Response port (Choose at application startup , and report to the Master)
Request ports (Select when the request is launched)
WORKER

Figure 2 : Communication procedures between nodes

For each node, including control node and general nodes, there will be a listening thread monitoring the nodes all the time. The listening thread ignores other requests, and is responsible only for monitoring ports and processing of the distributed requests. Upon arrival of a request, it can seek the corresponding function based on the types of data packet, and then transmit the request source address, network data, data length, and the descriptor connected with the corresponding network to the request handling function. This process is as shown in Figure 3 below.

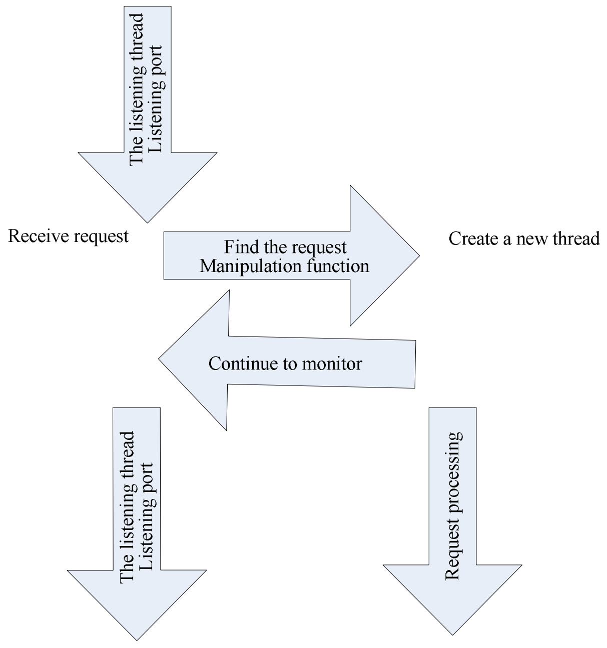

Figure 3 : Request processing procedure

In the system, immediate response for quick processing sometimes occurs when the thread will stay connected until request is responded to, meanwhile other requests will be blocked. If the request persists for a long time, delay will occur. Ifthe processing time cannot be measured, greater impact can be caused. In order to eliminate this effect, we introduced the asynchronous response mechanism. Request segment first produces an ID, and send this ID to the client side, which will wait for the appropriate time, during which time it will not accept the dispatch of thread. The server will send the response in the mode of request to the client side after completing the request. The client side will wake up the corresponding thread after acceptance of request. By this way, the unnecessary delay can be effectively solved.

OBJECTS AND THREADS

Object module is also known as heap module, which is responsible for providing the transparent access to the execution engine, and providing unified interface for access, during this process it doesn't matter whether the object is a remote physical node.

The system uses a 32-bit identifier assigned by the control node. The same reference indicates the same class or the same object on all the nodes within the system. Access across the nodes of the object field can be achieved by the overall reference of the object. References to the classes are distributed by control nodes. When general node requires referring to a class, it needs to apply to the control node at first. After the application is approved, all the nodes can access the class.

Java provides the support for multi-thread spontaneously, so the memory model of Java virtual machine is designed according to visual angle of the thread. Threads in Java can access the data directly. Java heap is the main memory of the Java virtual machine. Private memory of every thread is just its working memory.

The role of threading module is responsible for creation and migration of thread. When Java program is executed, Java virtual machine creates a new thread at first. The tasks of the virtual machine are distributed with thread as a unit among multiple nodes for concurrent execution. At the outset, more content will be added into the control nodes, which are included when general nodes execute thread, and the ID of the main thread is set to 0X00000001. Whenever general nodes create a new thread, start function of the classes will be called for, when the general node will send requests to a control node, and thread object will be included in the request. After acceptance of the request, the control node allocates the ID to the thread, and selects a node as a running node. Then 1 is added to the number of the threads, the control node send the request to the selected node. after receiving the request, the node obtains the object copy from objects, gets the ClassBloc from classes, begins to executes the Start() function, so much for the whole migration of a thread. The entire process is shown in Figure 4.

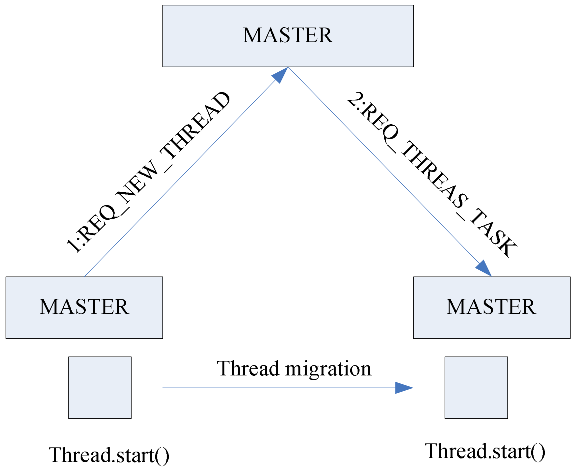

Figure 4 : The migration of Java thread

SYNCHRONIZED MODULE

The Java language naturally supports the multithreaded programming; therefore Java has a strong thread synchronization mutual exclusion mechanism. Life cycle of Java thread has 6 stages, namely: NEW and TERMINATED, and RUNNABLE, and BLOCKED, and WAITING, TIMED_WAITING, which have the meanings shown in TABLE 2.

TABLE 2 : The meaning of the six stages in the life cycle of Java thread

Condition	
NEW	Thread object has been created, but startO method has not yet called.
RUNNABLE	The thread has called start ( ) method, and entered “in progress" state, Java virtual machine may schedule thread execution at any time.
BLOCKED	Thread is waiting to acquire a Monitor from certain object/statics.
WAITING	The current thread is waiting for other threads to perform a specific operation.
TIMED— WAITING	The current thread is waiting for other threads to perform a specific operation , or The timer overrides when it enters the condition.
TERMINATED	The running of thread Mn ( ) method is finished, while the thread object has not yet been in garbage collection.

The transformational relation between these six states is as shown in Figure 5.

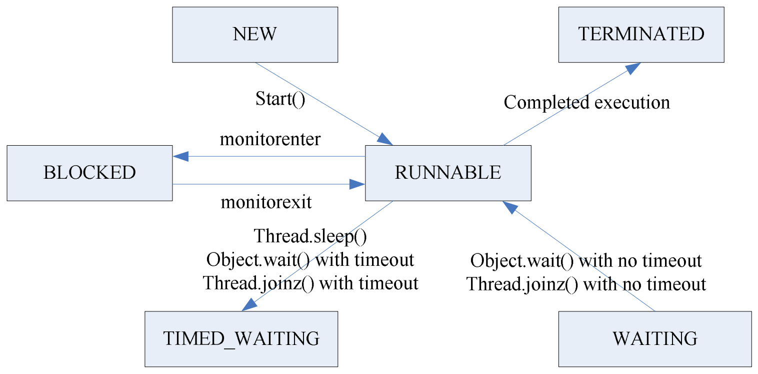
NEW
TERMINATED
Start()
Completed execution
monitorenter
BLOCKED
RUNNABLE
monitorexit
Thread.sleep()
Object.wait() with timeout
Thread.joinz() with timeout
Object.wait() with no timeout
Thread.joinz() with no timeout
TIMED_WAITING
WAITING

Figure 5 : Conversion of java thread states

MEMORY MANAGEMENT

Memory management is responsible for space allocation and garbage collection. At the start nodes will request to the system to allocate memory space, this memory space is aligned according to 8 bytes. Every time the allocation of space needs to be unlocked, and then conduct the space allocating operation. After operation, the space continues to be locked.

Java Virtual Machine is also responsible for the automatic recovery of useless space. This process is divided into two kinds, the first is the user execution, and that is, the user calls the mechanism to compel the Java virtual machine to recover space. The second means collecting the garbage automatically when space allocation failed. Garbage recovery is divided into local recovery and overall recovery, of which local recovery means only recovering the garbage at a node, is triggered by the space distribution failed; overall recovery is triggered by control node. It is caused by general node sending request to control node. The overall recovery is to carry out garbage recovery from all the nodes in overall scope; another situation will also trigger overall recovery: when the allocation need of the heap space can't yet be met after local recovery, the overall recovery will be done. The specific process is shown in Figure 6 below.

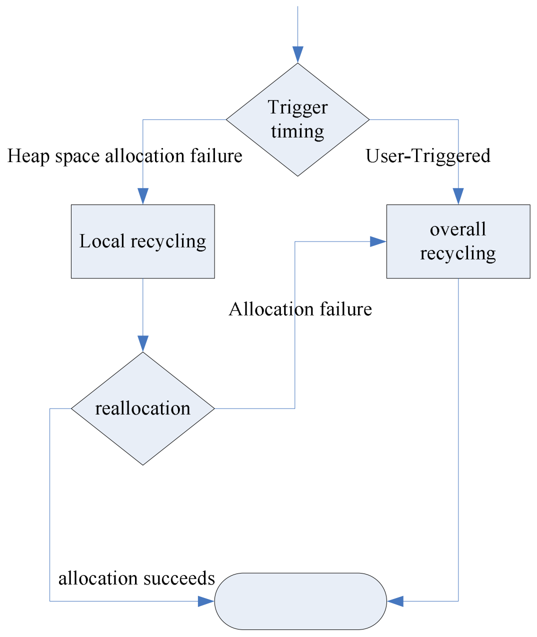

Figure 6 : Two kinds of garbage collection and their trigger timing

CONCLUSION

At present, distributed computing of the thread migration of the Java virtual machine is still comparatively out of practice, and the study on this aspect is relatively less. There are still some flaws with some ideas raised by this study, for example, erroneous node may lead to the potential threat of the downtime of the whole system. in the future work, special attention should be focused on how to standby mechanism of the general nodes, so as to make the general work node where the error occurred automatically exit, then replace them with the alternate nodes to continues to execute tasks quickly, and carry out the dynamic management of system topology on the basis of this system, therefore improving the reliability and MTTR of the whole system

REFERENCES

[1] Shen Yuanqiang; User-level multi-thread design and implementation of Java virtual machine [J], Wireless Internet Technology, **15(1)**, 101-102 **(2013)**.

[2] Xiang Lingling, Ren Guoxi; Some speculation on technical study and practice of Java virtual machine [J], Network Security Technologies and Applications, **15(4)**, 27-28 **(2014)**.

[3] He Yunbin; Adaptive dynamic optimization of Java virtual machine [J], Technological Innovation and Application, **08(13)**, 70-71 **(2014)**.

[4] Hao shuai; Discussion of related technologies in Java virtual machine [J], Success, **15(8)**, 169-170 **(2008)**.

[5] Ma Yanhong; Java thread migration and implementation in distributed Java virtual machine [J], China Science and Technology Information, **15(20)**, 142-146 **(2006)**.

[6] Chen Shi, Liu Rong; Research on memory management of real-time Java virtual machine [J], China Computer & Communication, **15(6)**, 100-102 **(2010)**.

[7] Zhang Yang, Zhang Jianbing, Jin Wenbiao; The research on Java thread migration mechanisms [J], Computer Engineering and Design, **16(5)**, 116-120 **(2006)**.