Analysis of Java Performance Optimization

Guoqin Li, Haiming Li
School of Computer and Information Engineering,Shanghai University of Electric Power,Shanghai 201300, China
zjxulhm@163.com

Abstract—The performance of the software is an important attribute index, and performance is particularly important for Java programs. Among the Java program, the larger the project, the more important performance optimization of the code is. This requires the program to complete the task with less use of resources as far as possible in the limited memory, CPU time, reduce code size, and improve the operating efficiency of the code. This paper presents five common factors of performance optimization, and puts forward some practical methods on code optimization to improving performance. These methods have also been applied in some actual projects. The paper also mentioned that some of the evaluation criteria of performance. For the different application, normally, the optimization method is appropriately chosen according to request.

***Keywords-** java; performance; optimization; code efficiency; web development*

I. Introduction

Java is a programming language of Sun's, launched in May 1995. It is no doubt that over the past decade, Java was the most widely used programming language. One interpretation of the Sun is that Java programming language is a simple, object-oriented, distributed, interpretative, robust, security system-independent, portable, high-performance, multi-threaded and dynamic language. Java is divided into three systems, JavaSE (Java2 Platform, Standard Edition), JavaEE (Java2 Platform, Enterprise Edition), and JavaME (Java 2 Platform, Micro Edition).

Java technology became the focus of people as soon as it born, and its scope of application in virtually every field. Because of its characteristic, compile once and run anywhere, largely reducing the workload of the software upgrading and servicing, reducing the development costs of the system. HotJava browser with Java implementation shows the charm of Java: cross-platform, dynamic Web, Internet computing. Java also thus been widely accepted and promoted the rapid development of the Web.

Yet, everything has two sides: simple and cross-platform do reduce a lot of work to programmers, but a well-designed Java programs, their performance is often not as good as a completion of the same functions in C or C++ program. This almost becomes the biggest weaknesses of Java. In Java program, the performance of most of the problems are not in the Java language while in the program, the programmer not mastered the methods to optimize the Java language.

Java is different from the general compile-execute computer languages and interpret-execute computer languages, it is first to compile the source code into a binary bytecode and then rely the variety different platforms virtual machine to interpret and execute the bytecode. In order to achieve a cross-platform features: compile once, execute everywhere. However, each execution of compiled bytecode need to consume a certain time, it also to some extent reduces the operating efficiency of Java program.

II. Evaluation Standard of Performance

Program performance is that the required for memory size and time to run a program. General we define following five aspects as evaluated performance standards.

1) Computing performance, which an algorithmic execution of the best performance.

2) Memory allocation, how much memory the program needs to allocate to running efficiency and highest performance.

3) Start time, how much time the program starting need.

4) The scalability, how to the program’s performance in the case the user overloading.

5) Recognizing of performance, users how to recognize program’s performance.

For different applications, requirements for performance are also different.

Generally, we can determine the performance of a program by two methods, analysis methods and experimental method. During analyze performance, using the analysis method, and during the measure performance, by the experimental method.

In order to improve the performance of Java, usually consider several factors the follows:

(1)The selection of the virtual machine

Different implementations of the Java Virtual Machine (JVM) will lead to a different running characteristics, development environments of various implementations or focus on providing convenience are lack of considering other aspects, or only consider certain factors that affect performance, resulting in overall performance may be very different. The high-performance Java virtual machine can improve the performance of Java applications through the full use of the actual hardware platforms and operating systems.

(2)Good framework design

The framework is the basis of the application, if the framework is unreasonably designed, it will produce a great obstacle to the subsequent optimization. Framework should be simple and clear, while taking into account a certain degree of scalability. This aspect can be to get through the learning of software engineering course, will not go into.

(3) The implementation of Java applications

When discussing the performance of the application, most programmers would consider the program's code, which of course it is, but more important is to find bottlenecks in code affect program performance. In order to find these bottleneck codes, we generally use some auxiliary tools, such as Jprobe, Optimizit, Vtune, as well as some analysis tools such as TowerJ Performance. These auxiliary tools can track the time consumed by each function or method in execute the program, so as to improve the performance of the program.

(4) Performance optimizing based on program test

All programs there are performance bottleneck, the bottleneck of the program in order to improve the performance of the program, it is necessary to reduce as much as possible. It can be found in program's deficiencies and bottlenecks through the program testing, and hence to optimize.

(5) Code optimization

The resources, like memory, CPU time, network bandwidth, etc., available to the program use of is limited, and the purpose of performance optimization is to let the program use as few resources as possible to complete the scheduled tasks, which requires the code to be enough efficient, this aspect of optimization is usually includes two elements: reduce the size of the code and improve the operating efficiency of the code. In this section mainly from angle of improve the operating efficiency of code to talk about performance optimization.

This paper mainly discussed from the perspective of code optimization.

III. Improving Code Efficiency

In Java program, the majority of performance problems do not lie in the Java language, but the program itself. It is very important to develop good code writing habit, such as right and clever use of the String class; it can significantly improve the performance of the program. Below specific analysis of this problem from several commonly used methods.

A. Selecting basic data types

In important loops, greater use of basic data types, instead of complex data types, and int type data in the calculation of the data is usually faster than the long and double type. Wrapper classes for basic data types, such as Boolean, Integer, etc., are used when passing the method's parameter must be a reference to an object, rather than a basic data type. Modified all constant expressions with 'static final' so constant could be called easier, because the compiler will be pre-calculated constant expression.

The following examples:

```
final int FLAGE=9;
            public int getTransform()
{
    int alter= FLAGE + 8;
    //...
}
```

To improving the efficiency of the program, and it should be changed as follows:

```
static final int FLAGE=9;
public int getTransform(){
    int alter= FLAGE + 8;
    //...
}
```

It should be noted also avoid free to use of static variables, when defined as static variables refer to an object, the garbage collection (GC) mechanism does not collect the occupied memory of this object.

B. Using StringBuffer

In Java, String is immutable object, once created then can't modify its value. Modification of the existing String object is to re-created a new object, and then save the new value. StringBuffer is a mutable object, modify it does not re-create objects like String, it is only through the constructor to build, after the object is created, memory space allocated to it in memory, the initial value is null, and through its append method to StringBuffer assignment.

```
String str = new String("welcome to ");
            str += "here";
```

The actual implementation steps of the above two lines of code are as follows:

```
            StringBuffer sb=new StringBuffer();
            sb.append("welcome to ");
            String str=sb.toString();
//implement
     String str = new String("welcome to ");
    sb=new StringBuffer(str);
    sb.append("here");
    str=sb.toString();
     //implement str+= "here";
```

As can be seen from the above connection operation of the String object is added to the operation of the StringBuffer object some operations, this will obviously affect the performance, which is a String object is immutable object, each operation the String will re-create a new object to save the new value, so that the original object is useless, and collected which certainly affect performance. This is due to that the String object is an immutable object, each operation String object will re-create a new object to save the new value, so that the original object is useless and is collected as the garbage which certainly affect performance.

C. Using shift operations of multiplication and division

Computer uses binary to record. Binary number left shift one bits indicates this number is multiplied by 2, and right shift divided by 2. The computer calculates the multiplication and division generally by shift operation of two's-implement and true form. Obviously, on the digital direct shift operations are much faster than the computer's conversion and shift operations. If multiplication and division can be substituted with shift operation, you should try to use the shift operation, but it is best to add comments, because the shift operation is not intuitive, and is difficult to understand.

```
for (val = 0; val < 1000; val +=4)
{
      flage = val * 8;
      result = val * 2;
}
```

The calculation of the above alternative multiplication and division using shift operation as follows:

```
for (val = 0; val < 1000; val += 4)
{
      flage = val << 3;
      result = val >> 1;
}
```

D. Making good use of variable

(1) Do not repeated initialize variables

By default, when calling the constructor of class in Java will be variable initialized to a value determined: all objects are set to null, integer variables(byte、short、int、long) are set to 0, the decimal variables(float、double) are set to 0.0, the logical values(boolean) are set to false, then we call the class do not need to go to initialize the various variables. Especially when a class that derives another class should pay attention to it, because when you create an object using the keyword 'new', all of the parent class constructor will be called automatically.

(2) Make use of local variables

In general, the parameters passed when calling the method and temporary variables created in the call are stored in the stack, and therefore faster; other variables, such as static variables, instance variables and so on are in the heap to create, slower. In addition, it is dependent on the specific compiler JVM, local variables may also be further optimized.

E. Using exceptions

In order to facilitate the user to capture exceptions and exception handling Java language provides the try/catch. But if used improperly, can also affect the performance of Java programs. Therefore, we should pay attention to the following two points.

(1) Avoid using try/catch, if you can use the logical statement instead, like if, while, then as much as possible without a try / catch statement in the program logic

(2) Reuse exception. If exception handling must be carried out, to the extent possible reuse of the exception object that already exists. Generate an exception object in the exception handling to consume a lot of time.

Exceptions can only be used for error handling, and should not be used to control program flow.

In order to make the compiler running optimization and code reuse rate should be raised, several methods can be invoked in a try / catch block, instead of a try / catch block for each method call.

```
try{
            method1();
}catch(method1Exception e){
            handle exception 1;
}
try{
      method2();
}catch(method2Exception e){
      handle exception 2;}
try{
      method3();
}catch(method3Exception e)
{
      handle exception 3;
}
```

The above should be written as follows:

```
Try
{
      method1();
      method2();
      method3();
}catch(method1Exception e)
{
      handle exception 1;
}catch(method2Exception e)
{
      handle exception 2;
}catch(method3Exception e)
{
      handle exception 3;
}
```

IV. CONCLUSION

The optimization method had applied in the project Software Simulation of Lifts，lift energy-saving method generally includes the design of the production processes and the use of mature control technology. In the project we compared the effect of a variety of control techniques to achieve the best energy-saving mode. For the optimization of the program, we use Jprobe for evaluation to improving program performance.

The performance of the software is an important attribute index, and performance is particularly important for Java programs. Among the Java program, the larger the project, the more important performance optimization of the code is. This requires the program to complete the task with less use of resources as far as possible in the limited memory, CPU time, reduce code size, and improve the operating efficiency of the code. The above article describes the evaluation of the performance of the general standard is five requirements. In order to increase the performance of Java usually comprehensive consideration of the following aspects: the selection of the virtual machine, good framework design, the implementation of Java applications, performance optimizing based on program test and code optimization. And spread out from the aspects of code optimization discussed several commonly used optimization method. For the different application, usually the requirements on performance are also different, therefore, the optimization method is appropriately chosen according to request.

V. REFERENCES

[1] Bruce Eckel. Thinking in Java (Fourth Edition). China Machine Press.2007.

[2] Gian-Paolo, D.Musumeci and Mike Loukides. System performance tuning. China Electric Power Press.2002.

[3] Azevedo A, Nicolau A, Sharp O. Java annotation-aware just-in-time (AJIT) compilation system [C] . Proceedings of ACM'99 Java Grande Conference. US. 1999.

[4] Li Y, Luo L, Lei H, et al., Performance optimization technology Java virtual machine for pervasive computing terminal s[J]. Application Research of Computers, 2005, 20(3): pp. 55-57.

[5] A.Adl-Tabatabai, et al., “Fast Effective Code Generation in a Just-in-Time Java Compiler,” Proceedings of the ACMSIGPLAN’98 conference on Programming Language Design and Implementation, 1998.

[6] R. Dimpsey, et al., “Java Server Performance: A case of building effcient, scalable JVMs,” IBM Systems Journal, vol. 39, no. 1, 2000, pp. 151-174.

[7] M.Schoeberl, “JOP: A Java Optimized Processor”, In Workshop on Java Technologies for Real-Time and Embedded Systems, JTRES 2003, LNCS 2889, Catania, Italy, November 2003.

[8] Loinig J. , Steger, C. , Weiss, R. and Haselsteiner, E., “Java Card Performance Optimization of Secure Transaction Atomicity Based on Increasing the Class Field Locality”, Secure Software Integration and Reliability Improvement, 2009, pp. 342 – 347.

[9] Xiaofang Zhang , Guohui Li and Xiaoling Lan, “Research on WebGIS Performance Optimization”, Wireless Communications, Networking and Mobile Computing (WiCOM), 2011, pp. 1 – 4.

[10] Lesnik, M. , Boskovic, B. and Brest, J. “Performance tuning of Java EE application servers with multi-objective differential evolution”, Differential Evolution (SDE), 2013, pp. 69 – 76.