Mobile Code Adaptation using Dynamic Code Generation

Daisuke Kawasaki

School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 1999

Keywords: Mobile Code, Dynamic Code Generation, Template, Java, Adaptation.

The paper is about a method for constructing efficient adaptive mobile code. The key technology for our goal is dynamic code generation (DCG), which is a kind of on-the-fly specialization technique of executable codes.

Mobile code technology — e.g., TeleScript and Java — has brought a new paradigm to design and construction of network applications. So far, the most network application have been built upon the Remote Procedure Call (RPC) paradigm. In the RPC paradigm, which was invented in 1970's, the communication between two processes is represented as an action in which one process calls a procedure of the other.

A message carried through the network is the request to the procedure or a response to the result of what the procedure has done. A request includes a procedure and data in the form of argument. A response includes data of the result carried out, and the procedure itself is in the computer in which it is carried out. One of the special features of remote procedure call is that an interaction of the computers requires two communications. First, we request the server to execute the procedure. Then the server sends us the result of process. During this transaction of remote procedure call, computers must be kept connected.

On the other hand, in Remote Programming (RP), in computer transaction, we can not only call the procedure from one computer to the other, but also transfer the procedure itself that is to be carried out. One of the striking characteristic of remote programming is that once users send a mobile code from their computers to the server through the network, the transaction is kept proceeding even if they shut down the network afterwards. In other words, computers don't have to be connected during the transaction. This is the most important advantage that has never been realized despite much of effort.

The mobile code has enabled us to execute the application in various computer environments. Indeed we come to be able to do in them, but the mobile code lacks the

consideration that it is suitable for each of them. There are numerous environments of the destination according to the purpose of the computers. (OS, CPU, network, special device such as mpeg encoder, and so on) There are cases in which we can get the improvement of execution efficiency and easier operation by doing the execution fit for the environments of the destination. In these cases, it is effective that the application adapts to change in the environments. But, it is difficult to make the application that can cope with a change in the environment. In order to adapt to numerous changes, we should imagine as many environments as possible and we can use the hardware that can do such instructions rapidly in some environments.

- An possible way is that we do drastically dispatch during the execution so that we can do a drawing instruction chosen among many. This method has an advantage that we can write the program easily while it has the disadvantage that the process is slow.

- Another way us that we should write the whole method that has no support of the hardware and one that has the support so that we start a method according to dispatch. This way has an advantage that the process is fast while it has a disadvantage that we are forced to make lengthy description.

Here, let us pay attention to changes in the environment. They can be divided into two categories: "environments that have the possibility of constant change ", and "environments which is characteristic of a machine" We can cope with the changes of each kind by dynamically adapting to changes in the environment, doing dispatch and call a method suitable for this environment. But, it is useless in the environments characteristic of a machine to do dispatch every time. The examples of such environments are follow as OS, CPU, network, special device such as mpeg encoder, and so on. Recent Operating Systems are equipped with the mechanism that enables us to dynamically change the size of the screen or the numbers of colors, but in fact we rarely do change them dynamically. Therefore we can decide almost statically on the environment information that is to be adapted to in the destination at the time when the mobile code is transferred in the environment in which it is carried out. So we can get rid of the overhead that dose the dispatch by making a new code fit for the environment in which it is carried out before executing a mobile code.

Let us apply this fact to the examples above. We can remove those disadvantages mentioned above while keeping those advantages by making a code fit for the environment, by using the information statically decided when it is carried out.

The dynamic code generation is often used as the means we can make a code when it is carried out. The dynamic code generation enables us to make the code which carry out very quickly compared with the re-compile of the program. There are following examples of dynamic code generation, which have their own characteristics.

- Method of generating native code by using dynamic code generation from the intermediate code.

- The way of optimization by using partial evaluation.
- The way of code generation by using templates and code bits which fill the templates.

This paper proposes methods for improvement of the processing speed while securing the ability of description by making a code fit for the executive environment at the time of the execution based on the information statically decided.

By using Java, I have made, as a prototype, the process that enables us to replace a command with equivalent but more efficient one and to get rid of unnecessary codes. This prototype doesn't damage Java's advantages, because it has never altered compiler of Java and Java Virtual Machine (JavaVM), Just In Time Compiler (JIT), and so on at all. In dynamic code generation, we have adopted the technique of optimization such as constant propagation, loop unrolling, the removal of the unnecessary codes by partial evaluation. In the prototype, we have made this time we can get the improvement of execution speed though we have done few sophisticated optimizations during dynamic code generation. Therefore, we can expect much more improvement of execution speed by applying sophisticated optimizations used in dynamic code generation to this prototype.