Efficient Method Lookup Customization for Smalltalk

Jan Vrany¹, Jan Kurš¹, and Claus Gittinger²

¹ Software Engineering Group,
Czech Technical University in Prague,
Thákurova 9, Prague, Czech Republic
{jan.vrany,kursjan}@fit.cvut.cz

² eXept Software A.G.
Zeppelinstrasse 4, Bönnigheim, Germany
cg@exept.de

Abstract. Programming languages are still evolving, and programming languages and language features are being designed and implemented every year. Since it is not a trivial task to provide a runtime system for a new language, existing runtime systems such as the Java Virtual Machine or the Common Language Runtime are used to host the new language.

However, most of the high-performance runtime systems were designed for a specific language with a specific semantics. Therefore, if the new language semantics differs from the semantics hard-coded in a runtime system, it has to be emulated on top of features supported by the runtime.

The emulation causes performance overhead.

To overcome the limitations of an emulation, a runtime system may provide a meta-object protocol to alter the runtime semantics. The protocol should fulfill opposing goals: it should be flexible, easy to use, fast and easy to implement at the same time.

We propose a simple meta-object protocol for customization of a method lookup in Smalltalk. A programmer may define his own custom method lookup routine in Smalltalk and let the runtime system to call it when needed. Therefore there is no need to modify the runtime system itself. Our solution provides reasonable performance thanks to low-level support in a runtime system, nevertheless the changes to the runtime system are small and local. At the same time, it provides the flexibility to implement a wide range of features present in modern programming languages.

The presented approach has been implemented and validated on a Smalltalk virtual machine.

1 Introduction

Many new programming, scripting and domain specific languages are created every year. In the past, each of the languages came up with its own runtime system – there is the Java Virtual Machine for Java, the Common Language Runtime for .NET languages and runtime systems for Perl, Python and Ruby.

Unfortunately, the implementation of a high-quality runtime system for a new language is challenging. Modern and high-performance runtime systems are complex machines providing memory management, thread management, performance optimization
and other features. Therefore, the current trend is to reuse an existing runtime system, such as the Java Virtual Machine (JVM) or the Common Language Runtime (CLR) to host the new language. For example, there have been over 300 languages implemented on top of the JVM [12], and many languages have been implemented on top of the CLR (IronPython, IronRuby or IKVM.NET to name some of them) or the Parrot VM.

Unfortunately, most of the high-performance runtime systems were designed with a particular semantics. Part of the hosted language where the semantics differs must be emulated on top of facilities provided by the runtime. A method lookup algorithm is one place where languages differ.

Despite many tricks which can be employed to improve the performance of an emulated method lookup, the performance is worse in comparison with a direct support in the underlying runtime system.

Microsoft provides a Dynamic Language Runtime (DLR) framework [4] built on top of the CLR. The DLR is an universal and flexible framework that does not require any changes in the underlying CLR. As discussed later, a disadvantage of the DLR is that it does not provide sufficient performance. In the JVM, there is a new instruction called invokedynamic in version 7 implemented within a scope of a JSR 292 [8]. Although the invokedynamic provides an excellent performance, the use of the new instruction means that an existing code cannot be customized without recompilation.

In this paper we present a simple meta-object protocol [9] for customization of a method lookup in the Smalltalk environment. A user-provided method lookup routine implemented in Smalltalk can be set for an arbitrary class. This user method lookup routine is then called by the Smalltalk runtime system. We validated our solution on Smalltalk/X, yet we are not aware of any obstacles preventing implementation in an arbitrary Smalltalk runtime system. The proposed meta-object protocol (MOP) allows the extension of an existing Smalltalk language or the implementation of a new language with a wide range of different lookup algorithms.

To get a better performance, our solution relies on a support in the underlying runtime system. However, the changes to the runtime system are kept as small as possible to facilitate the implementation of the proposed solution.

The contributions of this paper are (i) a design and an implementation of a simple, flexible and fast meta-object protocol for Smalltalk runtime systems and (ii) a validation of the presented approach on a language extension (selector namespaces) and a language implementation (Ruby) in Smalltalk/X.

The paper is organized as follows: Section 2 provides motivation and describes the problem in more detail. Section 3 describes our solution – a meta-object protocol (MOP) for a method lookup. Section 4 presents examples of using the MOP and validates its design. Section 6 discusses implementation issues, section 5 presents results of performance benchmarks. Section 7 discusses related work, and finally section 8 concludes and discusses directions of future research.