Dynamic Program Monitoring and Transformation
Using the OMOS Object Server

Douglas B. Orr, Robert W. Mecklenburg, Peter J. Hoogenboom, and Jay Lepreau

Department of Computer Science
University of Utah
Salt Lake City, UT 84112
E-mail: {dbo,mecklen,hoogen,lepreau}@cs.utah.edu

Abstract

In traditional monolithic operating systems the constraints of working within the kernel have limited the sophistication of the schemes used to manage executable program images. By implementing an executable image loader as a persistent user-space program, we can extend system program loading capabilities. In this paper we present OMOS, an Object/Meta-Object Server which provides program loading facilities as a special case of generic object instantiation. We discuss the architecture of OMOS, the extensible nature of that architecture, and its application to the problem of dynamic program monitoring and optimization. We present several optimization strategies and the results of applying these strategies.\footnote{This research was sponsored by Hewlett-Packard's Research Grants Program and by the Defense Advanced Research Projects Agency (DOD), monitored by the Department of the Navy, Office of the Chief of Naval Research, under Grant number N00014-91-J-1046. The opinions and conclusions contained in this document are those of the authors and should not be interpreted as representing official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the U.S. Government, or Hewlett-Packard.}

1 Introduction

Traditional program loading facilities, such as those found in Unix\cite{11}, have simple semantics, often because they are implemented within the framework of a monolithic kernel where resources tend to be constrained. Similarly they tend to use simple external structures --- executable files, libraries, etc. --- to reduce kernel complexity. One consequence of this simplicity of implementation is that as programs grow in size and complexity, the simple linking and loading algorithms used may produce poor locality of reference characteristics within the resulting programs. Program loading and execution facilities tend to be separate from compilation facilities, making it inconvenient to perform optimizations based on information derived at run-time.

In this paper we investigate the use of OMOS, an Object/Meta-Object Server, to improve locality of instruction reference by dynamically monitoring and transforming executable images. We begin by discussing typical linker technology and the particular problems of maintaining locality of reference within large programs. We next provide an overview of OMOS, its general organization, and its object loading facilities. Subsequently, we describe the use of OMOS' extensible nature to transparently monitor and transform executables to improve locality of reference. Finally, we discuss the results of our efforts, related work, and potential future work.

2 OMOS and Linker Technology

Separate compilation of program sources typically results in the generation of multiple object files which contain the generated program code and data. A linker is the program responsible for combining the object files and resolving inter-object file references. The linker manages large-grain code placement within an executable image. The decisions the linker makes with respect to code placement, in conjunction with the granularity of its data, determine whether a procedure is likely to be placed on the same page as the procedures it references. As program sizes increase, linker placement policies have an increasing effect on working set size and virtual memory utilization. In this paper, we are particularly concerned with the Unix