SOSP '97 poster abstract

Coign: Automated Distributed Partitioning of Component Applications

Galen C. Hunt
Microsoft Research
galenh@microsoft.com

Michael L. Scott
Department of Computer Science
University of Rochester
scott@cs.rochester.edu

Creating distributed applications is difficult because programmers must reason about both decomposition (what are the functional components of the application?) and distribution (where are those components placed within the network?). Distribution forces the programmer to make choices based on tradeoffs that have only a limited connection to application semantics, and that may change from time to time, or from network to network. For instance, an application distribution that works wonderfully on a local Ethernet may not work well at all when some of the components are moved to a handheld device connected through a radio modem.

We claim that automated tools can and should assume most of the burden of mapping components to machines within a network. To support this claim we are developing Coign, a software system that automatically distributes Component Object Model (COM) applications. Given an application binary, Coign instruments all of the interfaces between objects (components), regardless of whether those components lie in the same or different address spaces. The instrumentation tracks the communication across each interface, using semantic information gleaned automatically from interface descriptors to infer the cost of that communication under different distribution strategies. The inference task is complicated by the fact that components may pass parameters by reference within an address space, but must perform deep copies when placed in separate address spaces.

To distribute an application, Coign constructs a graph with one node for each component and weighted edges representing the cost of communication across each interface *if that interface were to cross a boundary between machines*. The programmer is given the opportunity to indicate components that must run at a particular location, e.g. to access some physical device. Using graph cutting heuristics, Coign then partitions the application to minimize communication costs.

We have finished an initial implementation of Coign on the Windows NT platform. Our system requires no access whatsoever to application source code. It includes two sets of runtime instrumentation. The first, with application overhead of 15-80%, is used to gather detailed data for partitioning. The second, with overhead as low as 3%, remains in the application after analysis to enforce the desired distribution and to notify the user when further detailed analysis may be desirable.

Our results demonstrate that applications built from binary standard components can be partitioned automatically. Applications can even be partitioned by end users as part of the installation process (with no knowledge of the application's internal structure or source code), to optimize the distribution to a particular local network or pattern of use of the application's feature set. We are currently investigating alternative graph-cutting heuristics. We are also conducting experiments with a wider set of COM applications to validate our initial results.