Portable Application-level Check-pointing

Contact: Rohit Fernandes, Dan Marques

One big advantage of taking application-level check-points (ALC's) is the possibility of saving state in a portable manner. This allows a program running on one platform to be restarted on a different platform after failure.

What would it take to provide portable ALC for existing C programs? Ideally, one would build a system that allows the original code to run unmodified or with a few modifications, without slowing the program down.

This can be quite tricky in a language like C. For example, consider the "union" construct in C. The problem with unions is that the runtime system cannot safely check-point a union value without knowing the type of the current value. One proposed technique is to develop a code transformation that keeps runtime information to identify the contained type. A naive translation of this approach can be very inefficient, but perhaps this can be optimized.

For the first part of the project, you will do a literature survey to determine what work has been done that is relevant to this question. For example, a number of different systems have looks at different aspects of type-safety in C. Systems like C-Cured and Cyclone try to give the low level power of C but ensure memory-safety by a combination of language extensions, static compiler analysis and runtime checks. These systems achieve performance by trying to minimize the number of runtime checks. Porch is another related system that a different set of design goals.

In the second part of the project, you will try to incorporate some of these ideas into an existing portable C check-pointer called Porch. This would first involve evaluating the current system and understanding its limitations with respect to the C specification. Your next goal will be to identify interesting features that need support but are not handled and provide solutions for them. Two concrete features that are not supported in Porch include unions and type casts involving writes. You will also do a performance evaluation of your solution and make recommendations for optimization.

If you are really ambitious, you can try to tackle this problem for C++.

REFERENCES


