Introspective computers

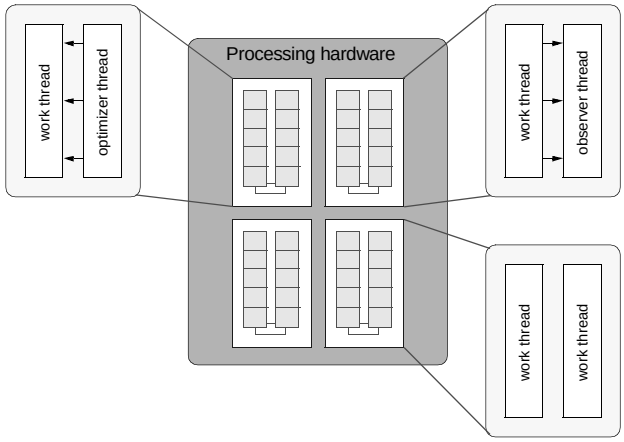
Krisztián Flautner **(manowar@engin.umich.edu)**
Trevor Mudge **(tnm@eecs.umich.edu)**

Abstract

Profile-feedback based analysis is a simple but powerful way of collecting information about a program. The downside of this approach is that it requires a set of “representative” input sets, it is time consuming and is hard to apply to interactive programs. These deficiencies are largely due to the prevailing compilation model, where compilation is an off-line activity. Having a compiler as a run-time resource, along with the ability to observe a program in real-time would increase the applicability of profile feedback. Instead of simulating a program only with input sets that are deemed to be representative, an introspecive computer can **optimize for the specific instance of program execution**. Hardware can only optimize based on a narrow window, while compilers have access to the big picture. These considerations have led us to propose the “introspecive” computer which attempts to combine the strengths of both approaches.

Introspecive processors blur the separation line of the traditional hardware-software interface. The compilers and profilers become part of the run-time environment (see Figure 1). During execution, information is collected by observer threads, which is fed back to a compiler that watches over the program’s execution. The compiler can make the decision to recompile parts or the entire program — taking run-time information into account — and insert the recompiled pieces into the executing program.

FIGURE 1. **High level model of the introspecive computer**



The introspecive computer consists of a hardware component that is able to execute one or more program threads and their corresponding observer threads simultaneously. A fast communications channel exists between these two threads. The software component consists of a compiler and custom threads that observe work-threads’ execution. The aspect of the execution that is observed and how that information is acted on is under compiler and operating system control.

The programs that are executed on the introspecive computer are encoded in a high-level distribution form, which is compiled to the instruction set of the processor by the resident compiler. The distribution form facilitates high quality optimizations, since information does not have to be rediscovered by the compiler and allows the architecture to evolve from one generation to another while providing **forward compatibility for software**. The form can be annotated with information that is collected during runs of the program to provide a richer context to the compiler. The distribution form currently under development is called Mirv.

In order to facilitate the real-time observation and analysis of threads, the hardware must support the **concurrent execution of multiple threads** (i.e. single chip SMP or SMT hardware). The compiler decides what aspects of the execution are observed, and inserts code into the work-thread to accomplish the measurements. The observer is custom generated for the program, and the aspects that are being observed. Moreover the work-threads must be able to pass information to their observers quickly, which implies that there is support in the ISA for inter-thread communications along with a **fast communications channel** between threads. The partial recompilation and insertion of code into a running program requires some level of hardware support as well. The open question that we hope to answer is whether the profile collection overhead can be offset by the benefits of optimizations enabled by it.