# Cavatools

Cavatools runs your application on a virtual machine, presenting a user-mode Linux system call interface to the application binary. The application with extended RISC-V instructions appear to be running in a Linux process with the same ?le system and network environment as the virtual machine. It seems to be in the same current working directory, was invoked with the same shell variables environment, and all the system call functions work as if the application was running natively on a Linux machine with the extended instruction set.

There are di?erent ways to achieve this virtual machine e?ect. Cavatools is a very fast software simulator: on an ordinary laptop the performance is 100-200 million instructions per second (MIPS) when creating a dynamic trace to drive an analysis program, 300-400 MIPS when not tracing (for statistical sampling). Most existing software are 100 to 1000 times slower. FPGA emulators have comparable performance but are more di?cult to create and maintain.
The tradition methodology for architecture evaluation relies on extracting small snippets of code from the application that are believed to dominate total execution time. Detailed simulations of these “performance kernels” are used to represent architectural improvements for the entire application. This approach is very successful for programs conforming to the 90-10 rule: 90% time spent in 10% code. Many mature, commercially valuable applications do not exhibit this behavior because they have been extensively optimized over many years to speed up commonly occurring specialized cases. The execution time of large real applications tends to be distributed across much greater percentage of code.
Cavatools encourages a di?erent architecture evaluation methodology. Instead of studying the behavior of a small number, e.g. few billions, of instruction execution in great detail, the evaluation is decomposed into a number of simpler simulations and studies carried out on trillions of instructions execution in the same amount of time.
The caveat is decomposability—Cavatools is not appropriate for modeling highly complex, tightly coupled behavior. Real computers are of course highly complex, and all the components are indeed tightly coupled to each other. Skill is required to decompose simulation by breaking some of the dependencies and interpreting the necessarily approximate results. But all simulation studies involve simpli?cation and are therefore approximations.

The cavatools system consists of an instruction set emulator caveat, a pipeline simulator pipesim, and a real-time non-intrusive performance viewer called erised. Caveat interprets a RISC-V binary and creates an in-memory instruction execution trace in shared memory, which is consumed on-the-?y by pipesim in a second Linux process running in parallel. Pipesim counts how many times each (static) instruction in the program is executed and total number of simulation cycles associated with that program counter. The frequency and cycle counters are in a shared memory segment which can be read non-intrusively by erised running in yet another Linux process. Erised displays execution frequency of di?erent parts of the program using colors in real time, with zoom and pan capabilities to examine details and a window showing assembly listing with frequency and CPI (cycles per instruction) for each instruction.