Approaches to Performance Measurement

There is more than one right way to measure application performance, and much depends on the context, the application's complexity, the type of information required, and the accuracy of the obtained results.

One approach for testing small programs or library methods is *white-box testing*: inspecting source code, analyzing its complexity on the whiteboard, modifying the program's source, and inserting measurement code in it. We will discuss this approach, often called *microbenchmarking*, towards the end of this chapter; it can be very valuable—and often irreplaceable—where precise results and absolute understanding of every CPU instruction is required, but rather time-consuming and inflexible where large applications are concerned. Additionally, if you don't know in advance which small part of the program to measure and reason about, isolating the bottleneck can be extremely difficult without resorting to automatic tools.

For larger programs, the more common approach is *black-box testing*, where a performance metric is identified by a human and then measured automatically by a tool. When using this approach, the developer doesn't have to identify the performance bottleneck in advance, or assume that the culprit is in a certain (and small) part of the program. Throughout this chapter we will consider numerous tools that analyze the application's performance automatically and present quantitative results in an easily digestible form. Among these tools are *performance counters*, Event Tracing for Windows (*ETW)*, and commercial *profilers*.

As you read this chapter, bear in mind that performance measurement tools can adversely affect application performance. Few tools can provide accurate information and at the same time present no overhead when the application is executing. As we move from one tool to the next, always remember that the accuracy of the tools is often at conflict with the overhead they inflict upon your application.