CHAPTER 1

Performance Metrics

Bef ore we begin our journey into the world of .NET perf ormance, we must understand the metrics and goals inv olv ed in perf ormance testing and optimization. In Chapter 2, we explore more than a dozen prof ilers and monitoring tools; howev er, to use these tools, y ou need to know which perf ormance metrics y ou are interested in.

Dif f erent ty pes of applications hav e a multitude of v ary ing perf ormance goals, driv en by business and operational needs. At times, the application's architecture dictates the important perf ormance metrics: f or example, knowing that y our Web serv er has to serv e millions of concurrent users dictates a multi-serv er distributed sy stem with caching and load balancing. At other times, perf ormance measurement results may warrant changes in the application's architecture: we hav e seen countless sy stems redesigned f rom the ground up af ter stress tests were run—or worse, the sy stem f ailed in the production env ironment.

In our experience, knowing the sy stem's perf ormance goals and the limits of its env ironment of ten guides y ou more than half way through the process of improv ing its perf ormance. Here are some examples we hav e been able to diagnose and f ix ov er the last f ew y ears:

- We discov ered a serious perf ormance problem with a powerf ul Web serv er in a hosted data center caused by a shared low-latency 4Mbps link used by the test engineers. Not understanding the critical perf ormance metric, the engineers wasted dozens of day s tweaking the perf ormance of the Web serv er, which was actually f unctioning perf ectly .
- We were able to improv e scrolling perf ormance in a rich UI application by tuning the behav ior of the CLR garbage collector—an apparently unrelated component. Precisely timing allocations and tweaking the GC f lav or remov ed noticeable UI lags that annoy ed users.
- We were able to improv e compilation times ten-f old by mov ing hard disks to SATA ports to work around a bug in the Microsof t SCSI disk driv er.
- We reduced the size of messages exchanged by a WCF serv ice by 90 %, considerably improv ing its scalability and CPU utilization, by tuning WCF's serialization mechanism.
- We reduced startup times f rom 35 seconds to 12 seconds f or a large application with 300 assemblies on outdated hardware by compressing the application's code and caref ully disentangling some of its dependencies so that they were not required at load time.

These examples serv e to illustrate that ev ery kind of sy stem, f rom low-power touch dev ices, high-end consumer workstations with powerf ul graphics, all the way through multi-serv er data centers, exhibits unique perf ormance characteristics as countless subtle f actors interact. In this chapter, we brief ly explore the v ariety of perf ormance metrics and goals in ty pical modern sof tware. In the next chapter, we illustrate how these metrics can be measured accurately ; the remainder of the book shows how they can be improv ed sy stematically .