Summary

Performance measurement is no simple task, and one of the reasons is the large variety of metrics and tools, and the effect of the tools on the measurement accuracy and the application's behavior. We've seen a large number of tools in this chapter, and you might be feeling a little dizzy if asked to recite precisely which situations warrant the use of which tool. Table 2-3 summarizes the important characteristics of all the tools demonstrated in this chapter.

Table 2-3*. The Performance Measurement Tools Used in This Chapter*

Tool	Performance Metrics	Overhead	Special Pros/Cons
Visual Studio Sampling Profiler	CPU usage, cache misses, page faults, system calls	Low	–
Visual Studio Instrumentation Profiler	Running time	Medium	Can't attach to a running process
Visual Studio Allocation Profiler	Memory allocations	Medium	–
Visual Studio Concurrency Visualizer	Thread visualization, resource contention	Low	Visual thread progress information, contention details, unblocking stack
CLR Profiler	Memory allocations, GC statistics, object references	High	Visual heap graphs, allocation graphs, GC timeline visualization
Performance Monitor	Numeric performance metrics at the process or system level	None	Only numeric information, not method-level
BCL PerfMonitor	Running time, GC information, JIT information	Very low	Simple, almost no-overhead runtime profiling
PerfView	Running time, heap information, GC information, JIT information	Very low	Adds free heap analysis capabilities to PerfMonitor
Windows Performance Toolkit	ETW events from system- and application-level providers	Very low	–
Process Monitor	File, registry, and network I/O operations	Low	–
Entity Framework Profiler	Data access through the Entity Framework classes	Medium	–
ANTS Memory Profiler	Memory usage and heap information	Medium	Powerful filters and great visualization capabilities
.NET Memory Profiler	Memory usage and heap information	Medium	Can open memory dump files

Armed with these tools and general understanding of what performance metrics to expect from managed applications, we are ready now to dive into the internals of the CLR, and see what practical steps can be taken to improve the performance of managed applications.