www.automatedqa.com

How to Choose a Code Profiling Tool

AQtime WhitePaper

AutomatedQA
test, debug, deliver!

Contents

How to Choose a Code Profiling Tool

This white paper examines what features to look for when choosing a profiling tool to improve application performance.

Application performance is crucial to a software company's success. When code executes quickly and efficiently, customers see an application as responsive and reliable; they view it as a time saver. However, when code goes into unnecessary loops, calls extraneous functions, or trips over itself in some other way, customers have a very different reaction. They may find the application sluggish or unresponsive, and come to see using it as a waste of time. If left unchecked, this ultimately results in frustrated users, a damaged corporate reputation, and countless lost dollars in revenue.

No matter how talented your development team is, very few lines of code run at peak performance when they're first written. Code must be analyzed, debugged, and reviewed to determine the most effective way to make it run faster. How can software developers and quality engineers ensure that their code is quick, efficient and ultimately seen as valuable? The solution lies in using a profiling tool to examine an application's code and locate and eliminate performance bottlenecks. These tools can quickly diagnose how an application performs and enable programmers to zero in on areas of poor performance. The result is a streamlined code base that performs at or exceeds customers' expectations.

Why Should You Own A Profiling Tool?

There are numerous reasons to invest in a profiling tool. Maybe you're one of the 81% of companies surveyed by Digital Focus who are employing (or looking to employ) an agile software development process[1]. One of agile development's hallmark characteristics is short development cycles, which usually range from two to eight weeks in length[2]. These short cycles leave little to no room for code refactoring or performance tweaks. Thus, programmers need to ensure code runs efficiently from the outset of the project.

You might also be trying to keep pace with your end users' rapidly diminishing attention spans. In 1999, Business Week reported that surfers would wait up to 8 seconds for a web site to load[3]. By 2006, Akamai found that surfers would only wait 4 seconds. Akamai also found that more than one-third of online shoppers with a poor experience abandoned the site entirely, while 75% were likely not to shop on that site again[4]. No company can afford that kind of negative attention.

Or perhaps you're experiencing high customer volume, and need to ensure you can handle the increased traffic. Amazon.com's sales, for example, increased 26% between 2005 and 2006[5]. An increase this dramatic shows how much successful online retailers need optimization in their day to day operations. A function or routine may respond quickly when one user executes it, but what happens when hundreds or even thousands of users execute it simultaneously? Code that hasn't been optimized can quickly degenerate into a performance nightmare when placed under stress.

Life Without A Profiling Tool

Optimizing code is challenging; it requires time, thought, and investigation from developers. Without the proper tools, programmers have to fall back on slower, less efficient ways of trying to optimize their applications. Some developers take to "pre-optimizing" code; they guess where performance problems will occur and refactor their code in an attempt to eliminate problems before they appear. This approach is problematic because a developer will often incorrectly diagnose the potential bottleneck. She may look only at her own code, instead of the full code base, thus missing integration issues. She may not have insight into her target users' expected behavior, or she may focus on an area of code that's infrequently used.

While well intended, this approach generally fails to find any true performance bottlenecks. Without a profiling tool to help locate bottlenecks, blind optimization only wastes time, as shown in the graphic below.

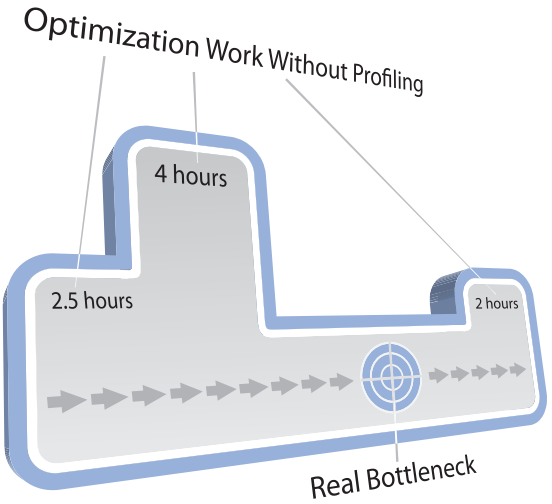


Figure 1: Blind optimization is a time waster

Simply choosing to not optimize code is a bad idea as well. No end user will want to make a slow application part of their daily life. If they become frustrated enough, they'll start looking for an alternative application. Disgruntled users often voice their opinions on blogs and other message boards, steering potential clients away from one company's products to another. That kind of damage to a company's reputation can be irreparable, and can cost the company countless dollars in lost revenue.

Profiler Use Challenges

Despite clear benefits, programmers have been slow to adopt profiling tools. They complain that traditional profilers are invasive and difficult to use. In some cases, programmers have found that certain profilers added so much overhead to application execution that performance data was skewed. Those profilers actually contributed to the problem instead of solving it.

Low Impact, Integrated Profiling Tool

Recently, however, a new generation of profiling tools has begun to emerge. This latest generation of profilers is not subject to many of the limitations of their predecessors, and truly does speed up the optimization process.

The ideal profiling tool integrates smoothly into your development process and allows precise and accurate measurements of your code's performance.

Minimal Impact Profiling

It's important to select a profiler that's as non-invasive as possible. Some profiling tools are highly invasive to an application's code; they may actually require the code be modified so that the profiler can take accurate measurements. Because this could result in hundreds or thousands of changes in the code, it has to be done by an automated source-modification tool. Once this takes place, a developer no longer has control over her code. When that happens, many people tend to question if the profiling tool itself may be injecting problems into the application.

The profiler you choose should not require any changes in source code, and should not introduce so much overhead that the application response time is impacted.

Ease of Use

Developers should select a profiler that allows them to quickly and easily diagnose performance problems in their code. If a user needs a complex set of commands to perform a simple action, he's going to look elsewhere to see if an easier solution exists. Additionally, because of the increased demand to create high performance applications in less time, developers don't have time to configure overly complicated tools. A profiler is meant to be something that speeds up the software lifecycle, not bog it down further. Such a tool should have a minimal learning curve, and allow developers to begin using it immediately.

Multiple Measurements

An ideal profiler would allow a developer to profile his code in several different ways. Having such a versatile profiling tool ensures that a developer can be certain his code performs well with regard to a number of factors; such as memory allocation, operating system resource usage, and overall application performance. It's also essential that switching between these various measurements be fast, easy, and intuitive, to ensure developers have ample time to run them.

Detailed Reporting

All the information generated by a profiler is worthless unless it can report that information in a detailed, easy-to-understand fashion. Whether the profiler outputs HTML pages or Microsoft Word documents, the data needs to be easy to interpret. What's more, the reporting system needs to be robust, providing all the information a developer needs, yet still be flexible enough so that the data can be filtered to show only what's relevant.

AQtime: AutomatedQA's Profiling Optimization Solution

AQtime is AutomatedQA's award-winning performance profiling and memory and resource debugging toolset for Microsoft, Borland, Intel, Compaq and GNU compilers.

AQtime includes dozens of productivity tools that help users easily isolate and eliminate performance, memory and resource issues within their code.

AQtime is built with one key objective - to help users completely understand how their programs perform during execution. Using its integrated set of performance and debugging profilers, AQtime collects crucial performance and memory/resource allocation information at runtime. It then delivers its profilers' results both in summarized and detailed forms. In short, AQtime provides you with all of the tools you need to begin the optimization process, and provides the following advantages:

Minimal Intrusiveness

AQtime never modifies source code. In fact, it always uses the least intrusive method to achieve the requested results. Since one normally expects results to refer to functions or sections of code, most AQtime profilers require only that the application be compiled with debugger information, so that code points can be linked to function or unit names.

Easy to Use Interface

Developers don't need to learn complex shell commands or proprietary programming languages to profile code with AQtime. It sports an intuitive interface that was designed with the busy programmer in mind. With just a few clicks, a developer can isolate performance or resource issues in a fraction of the time that other tools take. What's more, AQtime can function stand-alone or integrate directly into Microsoft Visual Studio or Borland Developer Studio. This feature provides you with the full AQtime functionality without leaving your IDE of choice. You can create and manage AQtime projects, profile your applications and view profiling results directly from the IDE.

AQtime extends the Visual Studio and Borland Developer IDEs in the following ways:

Context Sensitive Profiling

Right click in the Visual Studio editor and select "Profile this method" or "Profile this class" to start a new profiling session. It's easy and natural.

Inline AQtime Panels

All AQtime panels - Setup, Report, Summary, Call Graph, Assistant, and others - become true Visual Studio/Borland Developer panels.

Add a New AQtime Project Type

You can create AQtime projects or add them to an existing Visual Studio solution the same way that you create or add other VS/BDS projects; by using the Create Project

and Add New Project dialogs of Visual Studio, or the New Items dialog of Borland Developer Studio.

AQtime Menu and Toolbar Integration

AQtime adds the Profile menu to the IDE's main menu. This menu contains commands that let you start, pause and resume profiling, as well as select the profiling mode, modify the profiler and panel options, etc.

In Visual Studio, project-specific commands, such as Add Module, Add Output or Add Assembly, are added to the Project menu, and are also available through the context menu of AQtime's project node in the Solution Explorer.

In Borland Developer Studio, a Run with Profiling command is added to the Run menu.

AQtime also adds the AQtime toolbar to both IDEs. This toolbar contains the most frequently used items, such as Run, Select Profiler, etc.

Contributing to the Visual Studio Automation Model

The automation model of Visual Studio provides users with the ability to create custom add-ins, wizards, and work with macros. As a true Visual Studio-integrated product, AQtime provides program interfaces for its internals (for example, for the Projects interface). This allows third-party developers to create Visual Studio add-ins, wizards and macros that use AQtime's object model.

Fully Integrated into IDE Help system

The AQtime help system is fully integrated into the both the VS and BDS Help systems. F1 context-sensitive Help is provided for all AQtime panels, dialogs and project items.

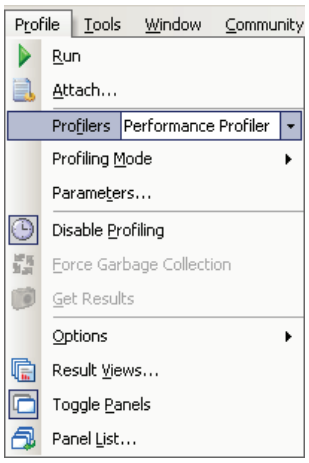


Figure 2: AQTime menus integrated into Visual Studio

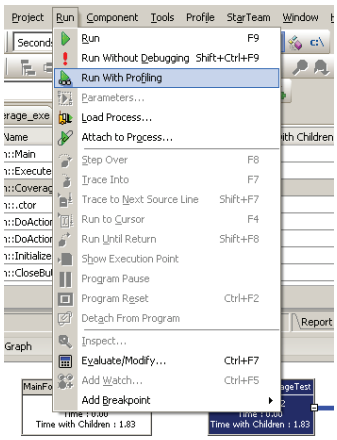


Figure 3: AQtime menus integrated into Borland Developer Studio

Easy to Understand Reports

Once a profiling session is complete, AQtime displays a summary report of its findings. This summary clearly pin-points the sections of code that need to be optimized,

allowing developers to quickly zero in on pain points in their code. The information shown is specific to the profiler that was run, so only relevant data is displayed. A more in-depth report is also available, allowing users to view detailed information for individual methods. Once generated, these reports can be stored as XML, HTML or TXT.

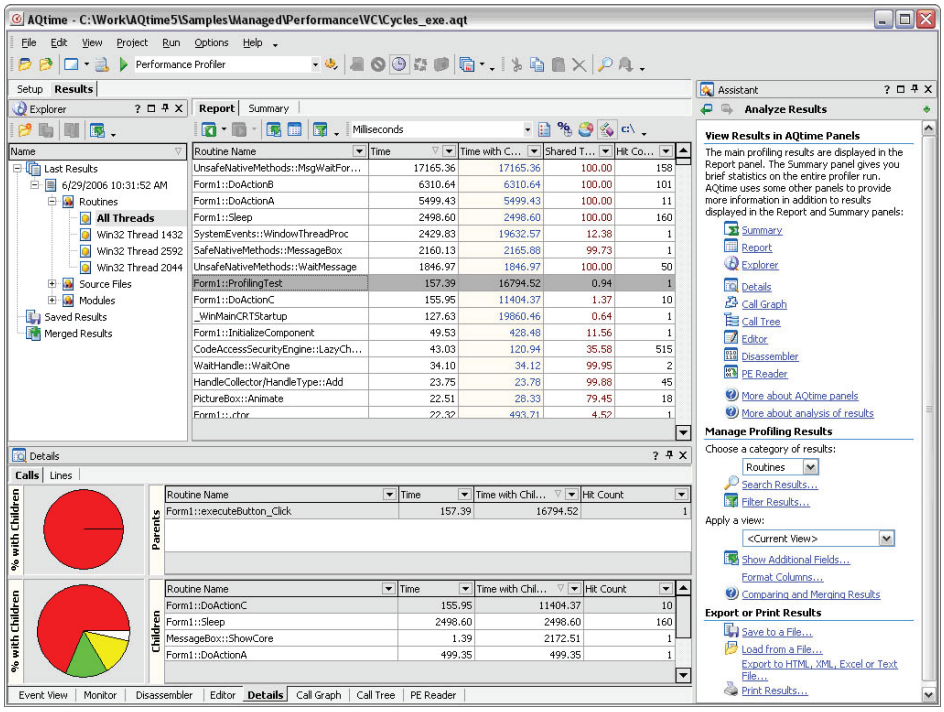


Figure 4: A sample report generated by AQtime

Support for a Multitude of Environments

AQtime supports a wide range of Windows versions including Windows 98, ME, 2000, NT 4.0, XP, Windows 2003 Server and Vista. Additionally, AQtime supports 32 and 64-bit chipsets from both AMD and Intel.

Support for a Wide Array of Languages

AQtime can profile code written in a wide variety of languages, including:

- Microsoft Visual C#
- Microsoft Visual Basic
- Microsoft Visual J#
- Microsoft C++
- Borland C# Builder
- Borland Delphi for .NET
- Borland Delphi 2005
- Intel C++
- GNU Compiler Collection
- Compaq Visual Fortran

A Variety of Metrics

AQtime boasts ten different profiler mechanisms which can be configured in several different ways. The end result is a flexible solution that allows users to easily ascertain those areas of most concern. The following profiler mechanisms are supported:

Performance Profiler - This profiler helps find poor performing functions and aids in their debugging. It monitors all of the method calls in an application, counts the calls, and traces the call hierarchy on a thread-by-thread basis.

Allocation Profiler - Helps you determine if an application is releasing memory properly. It does this by tracing the memory use in 32-bit and 64-bit applications during runtime.

Coverage Profiler - Lets you know how much of your code is actually being executed and tested. It determines whether a routine or a line was executed during the profiler run and how any times it was executed.

Exception Trace Profiler - Lets you confirm that the appropriate error messages are displayed for given circumstances. It monitors the application execution and (if applicable) outputs exception trace information.

Function Trace Profiler - Shows what code is called and when, allowing you to make sure the most efficient code path is being followed. It investigates the route and the order in which the routines are called during the application runtime.

Load Library Tracer Profiler - Loading and unloading dlls multiple times can significantly slow down an application. This profiler determines what dynamic link libraries were loaded and unloaded by the profiled application and how many times they were loaded and unloaded.

Platform Compliance Profiler - Lets you determine whether the profiled application can work on a specific operating system.

Resource Profiler - Checks to see if the application creates Windows resources correctly and releases those resources appropriately. It monitors resource allocations and de-allocations and calls to resource management routines.

Sequence Diagram Link Profiler - Generates a graphical map of how function calls are made, allowing for faster debugging of code. It analyzes the sequence of function calls in your application and then builds a UML-style diagram of function calls. This enables tracing links between methods and functions without running the application.

Static Analysis Profiler - Exposes routines that may not be written for optimal performance by analyzing debugging information or metadata. In doing this, a wealth of information is exposed, such as the number of loops in a routine, the size of the routines in bytes, all the possible code branches in an application, and more.

Fast ROI

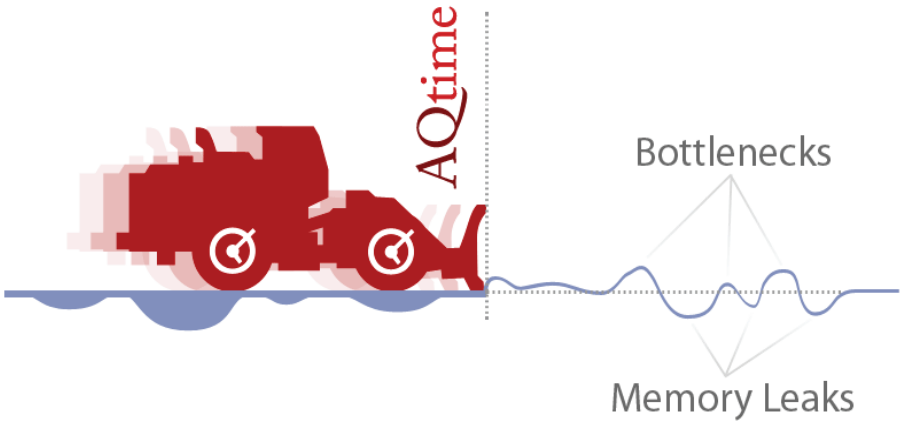
Unlike some tools, which have a steep learning curve and require extensive training, AQtime can be put to work immediately. Many users say that they were able to isolate and correct performance problems within minutes of running AQtime. Thanks to an intuitive interface and easy to understand reports, the optimization process can immediately change from painful to painless. There's no lengthy ramp-up time, no clunky configuration steps to follow, just a clean UI and a series of simple steps to ensure peace of mind for a development team's performance concerns.

Cost Effective

AQtime's pricing model is incredibly cost effective. Some other vendors charge up to five times more for products with fewer features than AQtime. This means an entire team of programmers can be outfitted with AQtime for the same price as one license for a competitor's product. No other profiling application can claim so much functionality for such a small price tag.

Learn More

To take the first step toward a smoother optimization process, visit www.automatedqa.com/downloads/aqtime http://www.automatedqa.com/downloads/aqtime and download a free trial version of AQtime."



About AutomatedQA

AutomatedQA offers software products and services for development and quality assurance projects worldwide. We create innovative, award-winning and affordable products for the entire software development lifecycle including TestComplete for test automation and AQtime, a sophisticated performance and memory profiler. AutomatedQA has an impressive list of customers ranging from huge teams in the world's largest organizations to progressive one-developer shops.

References:

[1] *Digital Focus http://www.infoq.com/news/Digital-Focus-Unveil-Survey-2006*

[2] *STSC http://www.stsc.hill.af.mil/crosstalk/2005/12/0512miller.html*

[3] *Business Week http://www.businessweek.com/1999/99_30/c3639021.htm*

[4] *Akamai http://www.akamai.com/html/about/press/releases/2006/press_110606.html*

[5] *Amazon.com http://phx.corporate-ir.net/phoenix.zhtml?c=97664&p=irol-newsArticle&ID=957221&highlight*