CASE STUDY | FINANCIAL

# ANTS Performance Profiler

## How it saved my hide

"ANTS Performance Profiler saved the day as well as my hide, and my team came out looking like heroes."

**Graham Russon**
**IT manager and senior developer**

redgate
ingeniously simple

91% of Fortune 100 companies use Redgate's software

Graham Russon
IT manager and senior developer
Industry sector: Financial
Place: Cape Town

## Contents

## Summary

Graham Russon, IT manager and senior developer at a financial institution in Cape Town, turned to Redgate's ANTS Performance Profiler to resolve an urgent crisis with their payment processes. As a result:

- Russon resolved the crisis in minutes, having spent days searching manually.
- He discovered a bottleneck in the code that had appeared to be in the database.
- The team could profile .NET code and SQL queries efficiently, side-by-side.

## Reliant on software

Although Russon’s development team is small, the company relies heavily on their technology and especially on their ability to make rapid changes. Any problems fall at Russon’s feet: "The buck stops with me when there are code problems."

The company has several SQL Server databases within their environment. They have 3rd party software that manages their debtors’ book, accounts, and collection strategy; on top of that, there is an in-house ASP.NET application that their 80+ users work with on a daily basis. The application performs all manner of tasks, from CRM, to account maintenance, journal management, and accounting. Each month, peak operation is around 100,000 database transactions a day.

## Single point of failure

Their most mission-critical application, integral to their debt collection process, is in daily use. "If it ever goes wrong, heads will roll," says Russon. It’s a console application written eight years ago in .NET Framework 2. Several people have worked on the code over time, and quick fixes have had to be made to address changing business needs.

Every morning at 10am, the application uploads a collection file to a bureau that deducts instalment amounts from debtors’ bank accounts, and pays over to the company’s own account. All of this must happen before the 10:30 cut off. If payment doesn’t get taken from an account, the chance of getting that money from the debtor decreases by 25% for each day not collected. Failure to make timely collections would mean severe cash flow problems and increased pressure for the Collections team.

## Critical error takes its toll

One morning, the application collapsed with an obscure error indicating a database timeout. And then it happened again. For weeks, the app would periodically crash, and the development team had no answers. "As it talks to three databases, we weren't even sure where the timeout was occurring," Russon recalls.

The crashes took their toll on the team: in order to make the tight deadline, they had to upload the files by email, and then laboriously enter the data into the database by hand.

## Needle in a haystack

Russon felt powerless. "As an IT manager who's been with the company over 20 years, it is a terrible thing to sit in an emergency management meeting, in front of my leaders and peers, and have no choice but to say those words I detest most in the world: 'I don't know.'"

Worse still for Russon, the company's collectors work on commission, and the malfunctioning app could have a serious impact on their lives. "To think of those people not getting paid properly, because I did not know, weighed heavily on my shoulders. I needed to find an answer."

But where in the application was the database timing out? The development team went through the code until their eyes watered: "We analysed every single stored procedure and SQL statement but could not identify where the bottleneck was. The database loop was working fine." Russon downloaded some profiling tools. Most were too complex to set up and some needed code changes.

## Straight to the source of the problem

One tool seemed to offer him a bit more hope – ANTS Performance Profiler.

He installed it, ran the project, and reviewed the analysis. Russon was impressed: "The interface was awesome, and had a whole bunch of information that for once made sense." It also presented him with performance data for both his .NET code and SQL Server queries, in a single profiling session – critical for the problem at hand.

But what Russon cared about most was finding his answer. "What gave me a jolt of excitement was seeing something I hadn't seen anywhere else: a spike in the wall-clock CPU time, one minute into the process."

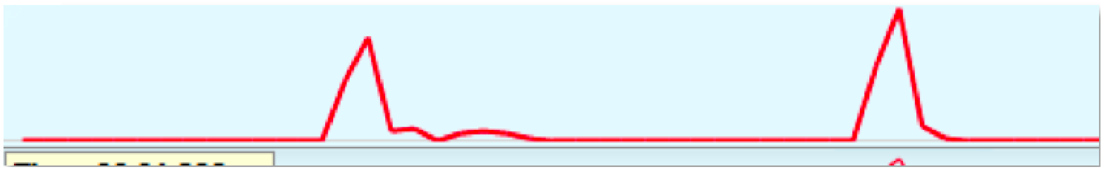
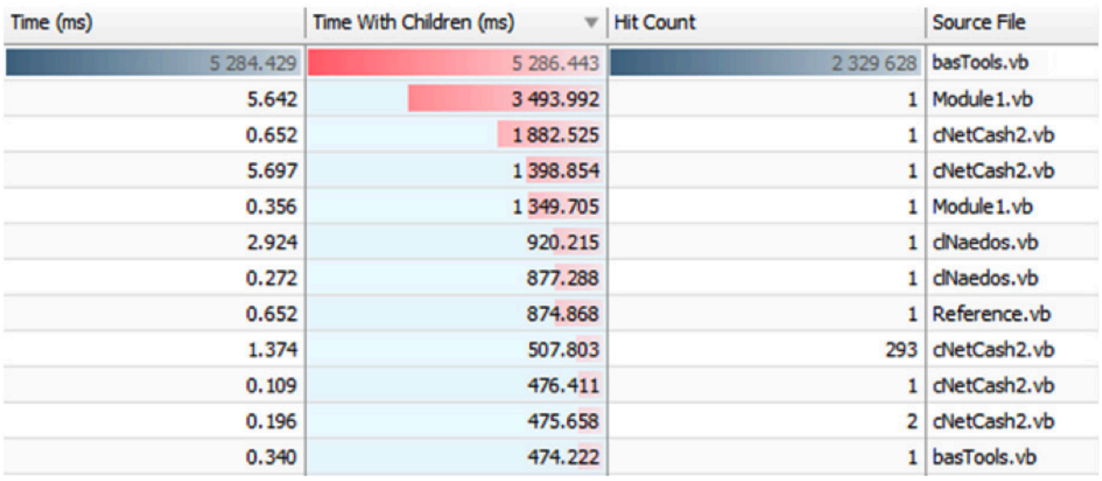


**Fig 1: ANTS Performance Profiler showing a spike in the wall-clock CPU time**

The high hit counts and time outs led him straight to the real culprit: basTools.



| Time (ms) | Time With Children (ms) | Hit Count | Source File |
|---|---|---|---|
| 5 284.429 | 5 286.443 | 2 329 628 | basTools.vb |
| 5.642 | 3 493.992 | 1 | Module1.vb |
| 0.652 | 1 882.525 | 1 | cNetCash2.vb |
| 5.697 | 1 398.854 | 1 | cNetCash2.vb |
| 0.356 | 1 349.705 | 1 | Module1.vb |
| 2.924 | 920.215 | 1 | clNaedos.vb |
| 0.272 | 877.288 | 1 | clNaedos.vb |
| 0.652 | 874.868 | 1 | Reference.vb |
| 1.374 | 507.803 | 293 | cNetCash2.vb |
| 0.109 | 476.411 | 1 | cNetCash2.vb |
| 0.196 | 475.658 | 2 | cNetCash2.vb |
| 0.340 | 474.222 | 1 | basTools.vb |

**Fig 1: The performance profiler pointing to the basTools.vb module as the root cause of the problem.**

Inside the basTools module were some standard basic tools the team used in every project – the last thing the team suspected might cause a timeout problem in SQL Server.

## Simple to fix when you know where to look

"The problem took all of two seconds to fix, and to this day I sometimes bang my head against the wall, re-living the moment," remarks Russon. "When an account has an alert on it, an automatic email gets sent to an account manager. One of the product types had been discontinued, and we'd recently re-instated it. This triggered an alert that called an old "discontinued" mail routine, containing a hard-coded IP address for a proxy server that no longer existed – hence the timeout. We pointed the IP address to a new server (this time, storing the setting in the config file) and ran the project successfully. To date, touch wood, we haven't had a single timeout."
Russon was delighted. "ANTS Performance Profiler saved the day as well as my hide, and my team came out looking like heroes."

The error message looked as though the timeout was coming from the database, so the team hadn't suspected faulty code. The .NET profiler pinpointed the culprit in just a few minutes, and restored the application to full performance, saving the company thousands in uncollected debt, and lifting the pressure from their collection and development teams.

Russon's recommendation is heartfelt: "ANTS Performance Profiler is indeed an ingeniously simple tool. A tool that does what it promises to do, and does it well."