ARM with RootCause

Oliver Cole
President and Founder
OC Systems, Inc.

Table of Contents

Introduction

RootCause, a software instrumentation tool from OC Systems, offers a new way to add Application Response Measurement (ARM) instrumentation to software applications. This white paper:

- Addresses the issues related to using RootCause to make ARM calls.
- Outlines the savings that can be realized by using RootCause.
- Explains why the existence of technology like RootCause's can speed the software industry's adoption of ARM.

ARM everything with RootCause

The Application Response Measurement (ARM) standard defines an open, vendor-independent API for measuring end-to-end application response time. The ARM API allows application and tool vendors to create management-ready applications and lets end users (i.e., the customers of vendors) better measure and control the total performance of their business-critical distributed applications.

A number of application vendors have incorporated these calls into their products. Most major applications, however, do not contain ARM API calls, or do not have them in the right places. Some vendors are planning to add calls in a future upgrade, while others plan no support for ARM.

Similarly, only some vendors of network management tools have implemented ARM-compliant agents and tools.

The slow speed of ARM adoption is frustrating to many. Part of the delay is due to the current need to wait for application vendors to incorporate the ARM API calls in their applications. This is a "chicken and egg" problem: application vendors won't insert ARM calls into their applications until the end users need it, and the end users can't use ARM tools until a critical mass of application vendors implement it.

But what if it were not necessary to change the application to insert ARM calls? Suppose end users or third parties could add ARM calls to any application without modifying the application, with or without the cooperation of the vendor?

If this were possible, ARM could be adopted at a much-accelerated rate. Any end users who wanted to take advantage of ARM could start using the calls right away, without having to upgrade their software.

RootCause provides exactly this capability. ARM calls can be added to an application without any access to the source. In fact, they can be added without changing the application at all. There is no need to wait for the vendor to insert ARM calls; end users or third parties (such as system integrators or ARM tool vendors) can insert the necessary instrumentation themselves.

The rest of this white paper is divided into two sections. The first section discusses the business and cost advantages of using RootCause to insert ARM calls into an application without rebuilding the application. The second section is a technical discussion of how RootCause can implement ARM.

RootCause and ARM: The business case

The benefits of using RootCause for ARM are best considered from three different points of view: the application vendor, the middleware vendor, and the end user. (For the purposes of this white paper, "vendors" sell software products that have ARM calls in them to "end users," who use ARM to monitor transaction response times.)

Application vendor's point of view

If a vendor has not yet enabled its product to use the ARM API, RootCause allows a quick and easy way to satisfy customer demand for an ARM-compliant version of the product. Using RootCause to implement ARM can be a stopgap measure in response to an urgent market need. Or it can be long-term, with the vendor perhaps selling the ARM capability as an add-on product.

Vendors can also benefit from the use of RootCause in their own testing and debugging process. These calls may be inserted at the customer's site, perhaps to help resolve a customer response time concern, or may be done at the vendor's site as part of on-going product development. Using RootCause in this manner may be quite cost-effective. For example, helping a customer narrow down a response time problem quickly and easily with temporary ARM calls can save a lot of time and effort, and result in a very satisfied customer. Additionally, vendors usually sell services as well as products. RootCause can help make these services very attractive, for example, by finding customer bottlenecks quickly.

Best of all, the RootCause and ARM technique can be used for already deployed versions of a vendor's software. It is very attractive for end users to be able to add ARM capabilities to their already working system, rather than have to perform system upgrades just to use ARM.

Middleware vendor's point of view

Perhaps the greatest value that RootCause brings to the ARM market is to middleware vendors.

These companies are currently caught in a "chicken or egg" situation: customers cannot commit to ARM and take full advantage of middleware that uses ARM instrumentation to monitor transaction response time until enough applications are ARM-enabled, and application vendors will not ARM-enable their applications unless customers demand it.

With RootCause, applications may be easily ARM-enabled with or without the cooperation of the application vendors. One company can therefore use RootCause to implement an end-to-end ARM solution for an end user without waiting for all of the application components to be modified by other vendors. This opens new markets for middleware vendors who embrace ARM.

Note that the middleware vendor does not need to buy or use RootCause to obtain this benefit. The vendor can simply point customers and potential customers to RootCause. RootCause lowers both the real and perceived risk for ARM users.

End user's point of view

With RootCause, the end user (or systems integrator, solution provider, or other third party) can insert ARM calls into any application, even commercial off-the-shelf (COTS) software, without needing access to the source code. From the end user's point of view, knowing that RootCause can "ARM Everything" means that the end user can commit to ARM wholeheartedly, knowing that all applications can be properly instrumented with ARM calls.

Also, from the end user's point of view, the ability to temporarily insert ARM calls to investigate bottlenecks or prototype systems can save a great deal of money. The temporary ARM calls will send their data to the standard ARM measurement collection software that is part of the end-user infrastructure, so the prototyping and bottleneck determination is done in an integrated fashion with the existing system.

RootCause and ARM: a technical discussion

This section is a technical discussion of how RootCause can add ARM calls to an existing application without changing it.

Overview of how RootCause works

RootCause gives developers and testers the ability to write probes that instrument a program's executable, collecting and analyzing data without modifying the source code. In fact, the probes work on the executable while it is in memory; no changes are made to the executable file. There is no need to re-link or re-compile the code, so testing and debugging can proceed quickly. Since RootCause does not require access to source code, it can be used on third-party code, including COTS software, as well as on software developed in-house.

The probes RootCause creates are non-intrusive and can be left in production systems, collecting data on an ongoing basis. Once a problem has been identified, RootCause can be used to test a fix. If the fix works, the probe can be left in until the fix has been integrated into the next build, allowing testing to continue.

RootCause can do much more than just add ARM calls. In fact, RootCause can add any ANSI C or Java code anywhere in an application. It does this using source-level identifiers, so one need not be aware of the machine-code idiosyncrasies at all.

This paper considers only scenarios that are directly applicable to ARM usage. We will discuss simple examples, but since RootCause allows any C or Java code to be added to an application, you have complete control of the conditions under which the application makes the ARM calls.

The examples given below show probes written in C. Probes can also be written in Java, which is fully supported.

For more information about how RootCause works, view other white papers and documentation at www.ocsystems.com.

ARM when the application vendor is cooperative

Application vendors and end users who have good rapport with their vendors will find it easy to discover where the ARM calls should be placed. Someone with access to the source can tell them to "Put a call to `arm_start` at line 32 of the function `get_message`," for example.

The user then writes the following in a file:

```
#include <arm.h>
probe thread
{
  probe "get_message"
  {
    on_line(32)
    {
      arm_start( 3, 0, $data_item, 200);
    }
  }
}
```

RootCause patches the application in memory based on what is in the file. The "$" in front of a name means that the item comes from the application itself, rather than from the file that is

being added. RootCause can ascertain where the data items are in the application. The user references them by putting a "$" in front of their name.

For RootCause to fully access all the items defined in the application, the application needs to be compiled with debug information. Most application vendors do not ship the debug information with the application, but have a version that they keep internally that has the debug information. In this case, the user can compile the above code using the internal vendor version and then apply it to the shipped version.

RootCause still works without debug information: the user can reference the parameters by position, or write ARM calls that do not need to reference data actually declared in the application. See the next section for a worst-case scenario.

Note that additional logic can also be used to be more selective about the ARM calls. For example, in the following example, only every 10th transaction results in starting an ARM measurement:

```
#include <arm.h>
probe thread
{
  int i = 0;
  probe "get_message"
  {
    on_line(32)
    {
      i++;
      if (i == 10)
        {
          arm_start( 3, 0, $data_item, 200);
          i = 0;
        }
    }
  }
}
```

ARM when the application vendor is not cooperative

In some cases, the application vendor may not be interested in ARMing its applications. Analyzing these applications to identify the correct ARM points requires a certain amount of effort. However, RootCause can be used to discover how the application functions at a low level, then it can be used to insert the ARM calls.

We discuss the worst case here: a vendor that has purposely stripped its application of all useful information. In this scenario, RootCause cannot actually add code to the application. But RootCause can add code to any of the shared libraries (or DLLs) used by that application. In other words, all of the interactions that the application has with the outside world have to go through those shared libraries, which are fully accessible to RootCause. In the general case, the user can use RootCause to follow what the application does with those shared libraries and then write a probe to trigger on the particular shared library calls.

For example, an ARM transaction can be started:

- After reading from a password file.
- After reading from a specific socket.
- After writing to a specific socket.
- After a specific key, or sequence of keys, is pressed.

The following RootCause code is used to add an `arm_start` call to Microsoft Word (MS Word). The `arm_start` call is performed each time a character is typed at the keyboard. RootCause intercepts the calls to `GetMessage()` and `PeekMessage()` that MS Word makes. When MS Word calls either of these routines, RootCause gets control in the `on_entry` section and calls `arm_start`, then RootCause returns to the normal execution of MS Word.

```
#include <windows.h>
#include <winuser.h>
#include <arm.h>
probe thread
{
  probe "GetMessageW" in "user32"
  {
    LPMSG MsgPtr;
    on_entry
    {
      MsgPtr = (LPMSG) $1;
    }
    on_exit
    {
      if( MsgPtr->message == WM_CHAR )
        arm_start( 33, 0, &(MsgPtr->wParam), 1);
    }
  }
  probe "PeekMessageW" in "user32"
  {
    LPMSG MsgPtr;
    DWORD RemoveIt;
    on_entry
    {
      MsgPtr = (LPMSG) $1;
      RemoveIt = $5;
    }
    on_exit
    {
      if( ( RemoveIt == PM_REMOVE ) &&
          ( MsgPtr->message == WM_CHAR ) )
          arm_start( 33, 0, &(MsgPtr->wParam), 1);
    }
  }
}
```

Adding ARM calls with RootCause is a simple and straightforward process. Although it is easier to accomplish when more information about the source code is available, RootCause can be used even when no information is available at all.

Conclusion

ARM is an extremely useful standard that offers significant benefits to organizations that use it to measure and improve performance. However, it has been handicapped by the slow response of application vendors, who have delayed including ARM calls in their software.

With RootCause, end users, ARM vendors, system integrators, VARs and other third parties no longer have to wait for vendors to release ARM-compliant versions of their software. They can use RootCause to insert ARM calls in any software application, whether or not they have access to the source code.

This benefits end users, who are able to commit to ARM without waiting for it to be fully accepted by vendors. It also benefits vendors of ARM tools, as well as application vendors that have already incorporated ARM calls in their products.

How to find out more

To learn more about RootCause, go to www.ocsystems.com, where you will find white papers, downloadable demos, and full documentation for RootCause. You can also arrange a personalized online demo over the Internet.

Or you can contact me, Oliver Cole, directly at +1 (703) 359-8160, or oec@ocsystems.com.