

Introduction into ARM

Stefan Ruppert

31. January 2005

Abstract

Introduction into the ARM (Application Response Measurement) Standard defined by The Open Group. First a brief overview of the history of ARM is presented followed by an overview and a description of the main concepts of ARM.

1 Applications and distributed systems

ARM (Application Response Measurement) is a standard of The Open Group defining interfaces for measuring the performance of business applications broken down into transactions and their possible sub-transactions. In this introduction, the current implementation of the ARM 4.0 standard is described briefly. Current application design tends to be more complex and distributed over networks. This leads to new challenges in today's development and monitoring tools to provide application developers, system- and application administrators with the information they need. For this purpose the CMG's (Computer Measurement Group) ARM working group defined the ARM specification which has been adopted by The Open Group as a technical standard starting with "ARM 3.0 for Java" [ARM3]. Now the ARM working group within The Open Group has moved on to the ARM 4.0 standard which unifies the "ARM 2.0 for C" [ARM2] and the "ARM 3.0 for Java" interface standards and also adds new functionality to better support generic instrumentation, targeting middleware applications and other complex scenarios. ARM 4.0 defines language bindings for C [ARM4C] and Java [ARM4J].

2 Overview

2.1 Issues

Within distributed applications it is not easy to estimate if the application performs well. The following issues help in the evaluation of distributed applications:

1. Are business transactions succeeding and, if not, what is the cause of failure?
2. What is the response time of a transaction?
3. Where are the bottlenecks, which sub-transaction could cause a bottleneck?
4. Which and how many transactions are executed in an application?
5. How to tune an application or its environment to perform better?

By providing answers to these questions, ARM can help the developer to implement an application as well as a system- or application administrator who has to maintain the running system. To achieve this, ARM measures the performance and the availability of defined transactions, where a transaction can be a high-level business transaction as well as a technical transaction such as the response time in a client/server relationship.

2.2 Approach

The main approach of ARM is to define transactions and insert calls to the ARM interface into the application ("instrument the application") to measure these defined transactions. Therefore availability of source code and of developers who instrument the application using standard ARM API calls is required. The instrumented application can be deployed as usual, except that now ARM transaction measurements are made and delivered to the appropriate ARM implementation's analysis or storage part.

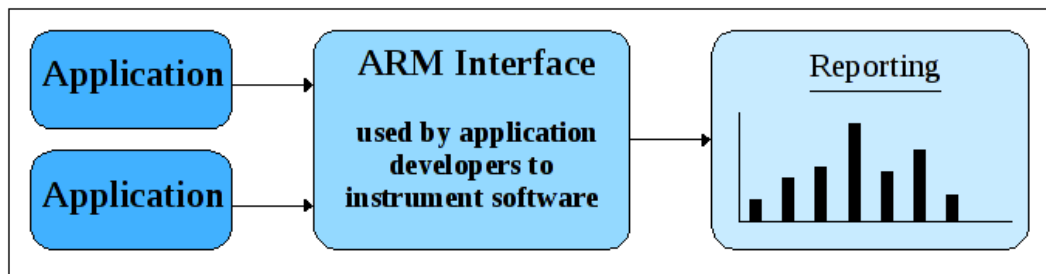


Figure 1: Conceptual ARM usage

2.3 Instrumentation

The following list briefly shows what to do in order to instrument an application using ARM.

1. Determine the business and technical transactions of interest.
2. Instrument the application with code using the ARM interface.
3. Deploy the instrumented application to gather ARM transaction measurements.
4. View, process or use the measured results as input of statistical operations to get an impression of how the application performs.

3 Main ARM concepts

In order to instrument an application, the following concepts are used within ARM.

3.1 ARM Transactions

Transactions are the main entity within the ARM system. Generally, they consist of a unique ID, a status of completion, a response time and the stop time when the transaction has finished. The status of completion signals whether the transaction was executed successfully (*good*), was aborted during the call (*abort*) or failed entirely (*failed*).

In the ARM 3.0 and also in ARM 4.0 version of the specification there exist basically two different concepts for measuring an ARM transaction:

1. Have the ARM implementation perform the actual measurement, including getting time stamps at the start and end points of the transaction.
2. Have the application itself measure the response time of the transaction, providing only an interface to report the measurement result to the ARM system.

The first approach is compatible with older versions of ARM, the latter was introduced with the version 3.0 of ARM, leaving more control to the application developer when and how to measure and/or report timings to the ARM system.

3.2 ARM Applications

Complex distributed applications usually consist of many different single applications (processes). In order to be able to understand the relationship between all single applications the concept of an ARM application

is introduced with version 4.0 of the ARM standard. Each ARM transaction is executed exactly within one ARM application.

3.3 ARM Systems

The nature of distributed applications implies that they are possibly operating on many different hosts simultaneously. Therefore, the concept of an ARM system defines the host an ARM application is executing on. The main information used to identify a system is the network address of a host. With this specification, it is possible to track the execution flow of a compound transaction even when it spans multiple hosts.

3.4 ARM Correlators

ARM correlators are used to express a correlation between two ARM transactions. This is a "parent/child" relationship. Commonly, a parent transaction triggers a child transaction and only continues its execution when the child transaction has finished. Using correlators, it is possible to split a complex transaction into several nested child transactions, where each child transaction can have child transactions of its own. This results in a tree of transactions with the topmost parent transaction being the root of the tree.

Put differently, ARM correlators can help isolating the part where a complex transaction failed, was aborted or has a bottleneck.

4 Refining ARM concepts

In many situations in the context of distributed applications deployment it could be useful to perceive a finer granularity of whats going on. For this, ARM defines the following concepts.

4.1 ARM Properties

Properties are a set of so-called name/value pair strings which qualifies an ARM transaction or an ARM application beyond the basic definition of these entities. There are two kinds of properties for both ARM applications and ARM transactions:

1. Identity properties are used to model more complex definitions of application or transaction types
2. Context properties are used to pass context information along with each application or transaction instance.

4.2 ARM Metrics

ARM Metrics can be used to get more information about the execution of a transaction. ARM defines a set of metric types for different purposes such as a counter, a gauge or just a numeric value.

For example, the response time of a transmission of a web-page from the server to the client depends, among other parameters, on the size of the web-page. To get an impression of how a transmission performs, an obvious approach would then be to normalise the response time by the size of a web-page.

References

- [ARM2] THE OPEN GROUP: *Technical Standard: Systems Management: Application Response Measurement (ARM) API*, Berkshire, United Kingdom, 1998, ISBN: 1-85912-211-6 [2](#)
- [ARM3] THE OPEN GROUP: *Technical Standard: Application Response Measurement, Issue 3.0 - Java Binding*, Berkshire, United Kingdom, 2001, UK ISBN: 185912 232 9, US ISBN: 1931624 04 6 [2](#)
- [ARM4C] THE OPEN GROUP: *Technical Standard: Application Response Measurement (ARM) Issue 4.0, V2 - C Binding*, Berkshire, United Kingdom, 2004, ISBN: 1931624380 [2](#)
- [ARM4J] THE OPEN GROUP: *Technical Standard: Application Response Measurement (ARM) Issue 4.0, V2 - Java Binding*, Berkshire, United Kingdom, 2004, ISBN: 1931624399 [2](#)
- [MYARMMGR] MYARM GMBH: *MyARM Manager Guide*, Gelnhausen, Germany, 2010