CMG Computer Measurement Group

The Challenges Monitoring Composite Applications

February, 2010

by David Stephens, Longpela Expertise

About the Author

[an error occurred while processing this directive]

Composite applications, or applications that cross application environments and platforms, pose unique problems for monitoring and tracking. This article discusses composite applications: what they are, monitoring issues, and solutions available.

Transaction monitoring traditionally concentrates on the individual transaction or database manager. It is straightforward to find the CPU usage for a CICS transaction, or response time for an Oracle database call. However this approach has limitations for applications that span transaction or database environments - composite applications. For example, monitoring an application that calls a CICS transaction, that itself calls two J2EE transactions on different servers is not straightforward. This problem is further complicated by middleware and message switching technologies available.

This article will look at composite applications. It will investigate the scope and difficulties inherent with tracking and monitoring these applications. It will also look at some monitoring technologies and solutions available.

Composite Applications

In its simplest form, a basic commercial application can be viewed as a program that accesses a local database as shown in Figure 1. In almost all cases, this database will be managed by a database manager such as SQL Server or Oracle Database.

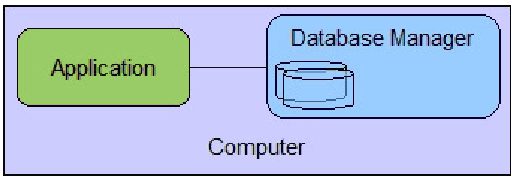

Fig. 1 - A simple application

The definition of a composite application varies widely. However for this article, a composite application is one that calls another application in a different application environment or transaction server; or accesses data on a different server. Examples are shown in figure 2.

It is easy to equate composite applications with SOA applications, however this is not strictly correct. The basis of SOA is to allow applications to communicate without knowledge of each other's platform, language or operating environment. Composite applications have no such limitations, and have been around far longer than SOA. However SOA applications are composite applications.

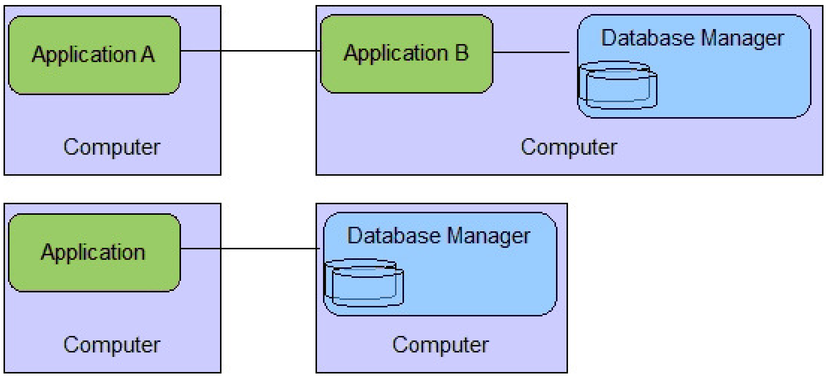

Fig 2 - A composite application

Client-Server: The First Composite Application

In the 1980s, as network connected PCs becoming more accessible, PC based applications accessing data on a central server became popular: two-tiered client-server. This is a composite application in its simplest form: a client program accessing data on a server.

More complicated, three-tiered client-server applications soon followed, where the client PC accessed an intermediate server with business logic and data access routines - an application server as shown in Figure 3.

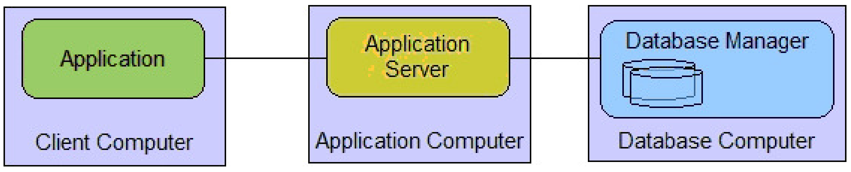

Fig. 3 - A three-tier, client-server application

Such applications become more complex as intermediate servers, database managers and applications are added to create four, five and six tiered applications.

Mainframe Composite Applications

Composite applications have been available for many years on mainframe IMS and CICS subsystems. Both have functionality to route transactions to other regions, and even to call transactions in those remote regions. For example, CICS Multiple Region Operation (MRO) allows CICS transactions and database calls to be routed to other CICS regions. IMS Multiple Systems Coupling (MSC) allows IMS transactions to be similarly routed.

The development of parallel SYSPLEX facilities has expanded this functionality. CICS transactions can be dynamically routed to any region within a CICSPlex using CICSPlex Systems Manager (CPSM). IMS queues can be shared by any IMS region within a sysplex using Common Queue Services (CQS), allowing multiple regions to process transactions in parallel.

Both IMS and CICS have mature features to allow access from J2EE and other non-mainframe applications. CICS Transaction Gateway provides a JCA connector for Java programs. IMS Connect provides a gateway for Websphere Application Server and TCPIP connected non-mainframe programs to call IMS transactions (and issue IMS commands). ISV companies further add mainframe connectivity options with products such as the TIBCO Adapters and Oracle Tuxedo.

In recent years, both CICS and IMS have introduced SOAP facilities that can process, and even generate SOAP message calls. IMS uses the IMS SOAP gateway and IMS Connect; CICS has built-in facilities.

Middleware

The past two decades has seen the release of many middleware technologies and frameworks with the aim or easing the pain of 'un-like' application communication. For example, Oracle has a wide range of Integration Adapters as part of Oracle Fusion Middleware to simplify application access to anything from PeopleSoft to IMS.

Websphere MQ takes this to another level. In fact the release of Websphere MQ (then MQSeries) in the early 1990s can in many ways be seen as the birth of SOA. Websphere MQ takes ownership of the logistics of communicating between applications, including server address, security, message delivery and data conversion. Websphere MQ also allows Queue Managers to route Websphere MQ messages on to other Queue Managers as shown in figure 4.

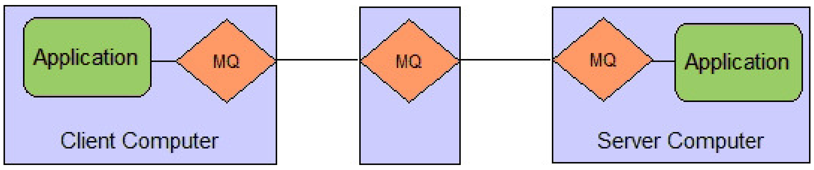

Fig. 4 - An application with WebSphere MQ messages routed through an intermediate queue

This picture gets more complicated with the dynamic routing abilities of Websphere Message Broker, and the use of Queue Sharing Groups. This technology makes it more difficult, if not impossible, to know ahead of time exactly which server and application will process an MQ message, or what route a message may take.

Similar issues exist for the increasingly popular SOAP framework. Using standard HTTP and XML, applications can send and receive SOAP requests irrespective of platform or application. Enterprise Bus software such as TIBCO Enterprise Service Bus and Websphere Enterprise Service Bus can dynamically reroute (or stop) SOAP messages, again resulting in possible uncertainty as to which application or server is actually processing the message, or which route will be taken.

Such dynamic rerouting facilities are not restricted to middleware solutions. Many options are available to dynamically reroute TCPIP traffic from one server to another.

Complications from Database Managers

Figure 1 shows a simple application accessing data from a local database manager. With the wealth of database manager options available, many applications can now access databases from multiple database managers as shown in Figure 5.

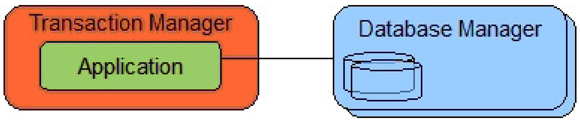

Fig. 5 - An application accessing multiple database managers

Such applications will often require two-phase or three-phase commit facilities from a transaction manager or operating system. Monitoring is more complicated, and usually done separately for the transaction manager and database manager. However some transaction manager monitoring tools can provide information about database calls - which databases manager, how many were processed, and how long they took.

Facilities to allow database managers to route database access requests to other database managers makes this picture more complicated. The DB2 Distributed Data Facility (DDF) allows DB2 subsystems on different platforms to route database requests to each other. The Open Group's Distributed Relational Database Architecture (DRDA) standard goes one step further, allowing different database managers such as Apache Derby and DB2 to route database requests. So Figure 5 could become Figure 6.

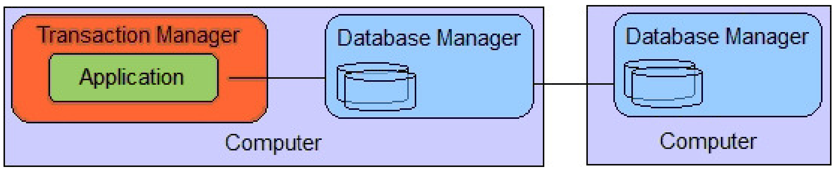

Fig. 6 - An application with database calls routed to another DBMS

IMS databases aren't far behind, with the Open Database feature of IMS 11, IBM IMS Enterprise Suite, and other ISV products such as the iWay Intelligent Adapter for IMS easing access to IMS databases.

Stored procedure facilities in database managers now blur the distinction between a database manager and a transaction manager. Many database managers like DB2 allow stored procedures to be written in high level languages such as C, and even access transactions on other transaction managers such as CICS. So it's not impossible to see an application as shown in figure 7.

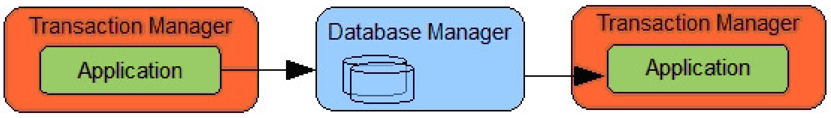

Fig. 7 - An application with a database manager calling another application

Problems When Monitoring Composite Applications

As middleware, transaction and database managers have made it easier for applications and database managers to access each other, it is logical to expect the number and complexity of such composite applications to increase. This is made more difficult by dynamic routing and queue sharing features. In many cases there is no way to know which database manager or application will process a request or message before it occurs.

This is the biggest challenge when monitoring composite applications. Consider the hypothetical composite application in figure 8. A Windows Java application calls a J2EE application running on a UNIX server. This application sends SOAP requests to either Application B running on another UNIX application, or Application C running in CICS on z/OS. An ESB solution dynamically controls which application receives the SOAP request.

Traditionally, monitoring of this application is done at each application server level. However this is only partially effective. There is no way to correlate a transaction running Application A with a transaction running Application B or Application C. In fact, it may not be possible to know for certain which application processes each SOAP request. This problem increases if the technical staff are unsure or unaware of the full path of a composite application, particularly if it is complex or has been developed over a long period.

This raises the real concern that organizations will not know the processing path of some composite applications, making any monitoring, problem determination, capacity planning or change management extremely difficult, if not impossible.

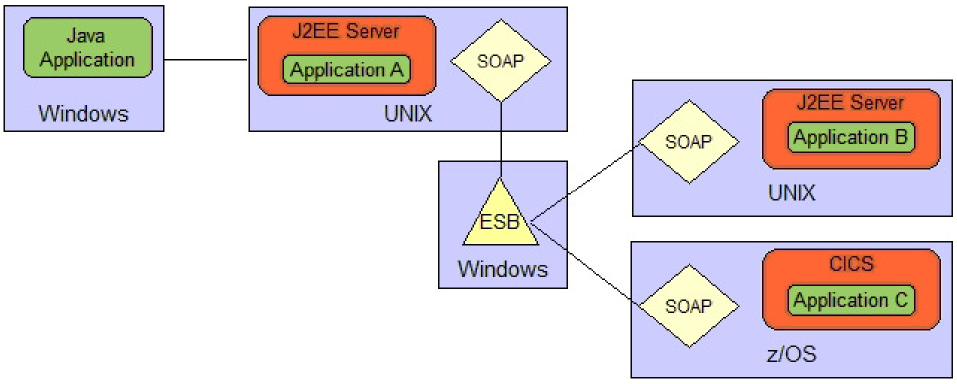

Fig. 8 - An example of a composite application

To monitor any composite application from end to end, a way of matching or correlating each step of the composite application is required. In almost every case, this will require some form of token to be available at each end of a composite application's step as shown in figure 9.

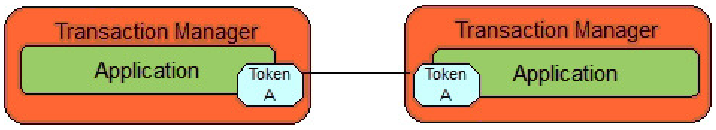

Figure 9 - A composite application step with a common token

This token, or *correlator*, must be identical at both ends of the composite application step. Unless an API call or calculation can produce or generate such a unique correlator, it must be passed from the beginning to the end of each step - either by the application, transaction or database manager, message transmission system or middleware. This is the major obstacle to monitoring composite applications.

ARM - A Possible Solution

Application Response Measurement (ARM) holds a lot of promise for monitoring composite applications. APIs are available for Java and C, and other languages such as SAS. From ARM 2.0, functionality exists to obtain a correlator for linking different composite application steps. Monitoring software can use this correlator to build a topology view of the composite transaction and its performance.

But ARM isn't perfect. It requires someone to call ARM APIs on both ends of a step. It also requires someone to pass this correlator from one step to another. Few companies will accept the expense and work to retrofit all applications with such ARM calls. So that 'someone' must be the application environment, transaction or database manager. However few currently offer this functionality.

The basic problem of passing a correlator from one application to another still remains. Even if a transaction manager were to add ARM calls, it would need to add a correlator to data passed to the remote system. Similarly the destination transaction manager would need to obtain this correlator for its own ARM calls. Even ignoring the risks involved with modifying data passed between them, there is little chance that un-like transaction managers will be enhanced to do this at any time in the future.

What's more, ARM is not available for legacy applications on z/OS, so the CICS transactions in figure 8 cannot be monitored in this way. EWLM was to be the z/OS implementation of ARM, but has been removed from service by IBM.

ARM also needs software to process the ARM related information, so a monitoring solution for every step of the composite application is required.

Commercial Monitoring Software

The scale and magnitude of the task of monitoring composite applications provides an opportunity for software vendors. Major players include CA-Wily Introscope, Tivoli ITCAM, and Oracle Composite Application Monitor. However, only CA (using their recently announced Cross Enterprise APM to connect Wily and SYSVIEW information) and IBM (ITCAM for Transactions) attempt to span the mainframe/non-mainframe divide.

The task for these vendors is enormous. Leaving aside the problems of obtaining and passing a correlator, this software must attempt to support as wide a range of environments as possible. This range is enormous, and will surely increase over time. Achieving 100% coverage of a composite application environment may not be achieved by any single monitoring product, particularly if unorthodox methods or lesser-used software are used for communication. ITCAM for Transactions does provide an API for users to add their own information to that automatically accumulated. However IBM is yet to release the information needed to link user API calls to monitoring data automatically collected by the product.

ASG-becubic takes a different approach for composite applications, scanning application source code as well as monitors, database managers and middleware. Although this does not provide real-time performance monitoring information, it can provide a picture of a composite application's path and relationships.

This area is relatively new, and vendors are still working to produce mature solutions with sufficient scope to address complex customer environments.

Conclusion

Composite applications provide another layer of complexity to the task of monitoring transactions and applications. As middleware, transaction and database managers ease the task of creating composite applications, the number and complexity of such applications is sure to increase. Further, dynamic routing and queue sharing technologies now blur the links and relationships between applications, removing the certainty of exactly which application will communicate with which.

Although ARM provides a basic framework to tackle this problem, the problem of passing a correlator between composite application steps is still not resolved. With such a difficult problem, it makes sense to look toward commercial monitoring solutions. However these solutions are relatively new, and must support every step of each composite application to be monitored. Because of this, it may be difficult to achieve 100% monitoring coverage of a complex composite application environment in the short term.

References

- Three-Tier Architecture, Ariel Ortiz Ramirez, Linux Journal, Jul 2000 (http://www.linuxjournal.com/article/3508)
- DRDA, Craig Mullins, dbazine.com, Feb 2002 (http://www.craigsmullins.com/drda.htm)
- DRDA V4 Technical Standards documentation, volumes 1-3 (http://www.opengroup.org/)
- IBM Redbook: DB2 V9 for z/OS: Distributed Functions (http://www.redbooks.ibm.com/abstracts/SG246952.html)
- iWay Intelligent Adapter for IMS (http://www.iwaysoftware.com/products/adapters/ims.html)
- Extending the Power of IMS, Chong Guzik and Lai, IBM Systems Magazine, Nov 2009 (http://www.ibmsystemsmag.com/mainframe/enewsletterexclusive/27676p2.aspx)
- IBM Redbook: IMS Version 11 Technical Overview, Bruni, Greenshaw, Martinez, Aerschot, Wilkinson, IBM Redbook (http://www.redbooks.ibm.com/abstracts/sg247807.html?Open)
- Enabling z/OS Applications for SOA, Kooijams, Akimoto et al, IBM Redbook (http://www.redbooks.ibm.com/abstracts/sg247669.html?Open)
- TIBCO Enterprise Service Bus (http://www.tibco.com/software/enterprise-service-bus/default.jsp)
- Websphere Enterprise Service Bus (http://www-01.ibm.com/software/integration/wsesb/)
- Open Group ARM website (http://www.opengroup.org/management/arm/)
- Building Application Manageability using ARM 4.0 Standards, Panicker, Parhar, Chadha, CMG Measure IT, Oct 2005 (http://www.cmg.org/measureit/issues/mit26/m_26_3.html)
- Monitoring and Diagnosing Applications with ARM 4.0, Johnson, CMG Measure IT, Mar 2005 (http://www.cmg.org/measureit/issues/mit20/m_20_3.html)
- IBM Software Withdrawal Notice 907-245, Dec 2007 (http://www-01.ibm.com/common/ssi/index.wss?DocURL=http://www-01.ibm.com/common/ssi/rep_ca/5/897/ENUS907-245/index.html&InfoType=AN&InfoSubType=CA&InfoDesc=Announcement+Letters&panelurl=index.wss%3F&paneltext=Announcement%20letter%20search)
- Oracle Integration Adapters (http://www.oracle.com/technology/products/integration/adapters/index.html)
- Websphere MQ Information Center (http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.branding.doc/help_home_wmq.htm)
- IBM Redpaper Websphere MQ Queue Sharing Group in a Parallel Sysplex Environment, Injey Raja, Bhattall, Siddall (http://www.redbooks.ibm.com/abstracts/redp3636.html)
- Websphere Message Broker website (http://www-01.ibm.com/software/integration/wbimessagebroker/).
- CA Wily Introscope webpage (http://www.ca.com/us/application-management.aspx)
- ITCAM for Transactions webpage (http://www-01.ibm.com/software/tivoli/products/composite-application-mgr-transactions/)
- Oracle Composite Application Monitor and Modeler Documentation (http://download.oracle.com/docs/cd/E14209_01/index.htm)
- ASG becubic webpage (http://www.asg.com/products/product_details.asp?code=BSZ)