Realtime publishers

The Definitive Guide™ To

Application Performance Management

sponsored by

Greg Shields

Copyright Statement

Chapter 4: Integrating APM into Your Infrastructure

"You want me to enable this on all the equipment? Switches, routers, everything?"

"Yep. That's the plan. Point everything back to 192.168.0.55. That's the server we've set up to collect all this information."

The network engineer sits back in his chair, patently annoyed with the request, "You realize how much effort this will involve, connecting to every network device all across the network to do this one change. When do you need this done?"

"How about…now…?" responds John Brown, IT manager for TicketsRus.com. Now it's John's turn to get annoyed at how this conversation is going. He'll admit that this is a big request, but that's why he pays his network engineer a salary, to make these sorts of things happen.

John thinks a bit as he sits across the desk from his network engineer. He's been reading more of late about Gartner's concept of IT maturity, most especially since that last incident on Memorial Day. In learning more about the differences between a reactive IT culture and one that proactively solves problems, he's come to realize how "reactive" his organization really is. That Memorial Day incident should have never happened…

"So tell me again about this 'magical' monitoring solution you're dreaming up, John. You know we're already collecting MRTG statistics off all the network equipment. You look at the same graphs I do. We know when we're seeing problems with bandwidth or when our ISP isn't meeting their agreements," continues the network engineer, obviously irritated about this intrusion into his team's traditional boundaries, "I hate to be blunt, but why should my team care about this new technology?"

John fires back, "Because it's this 'new technology' that's going to save this company. You too got called in for the Memorial Day incident. You remember how long it took us to track down the solution."

"It was the developer's fault!"

"True," responds John, mentally noting this conversation for the engineer's performance appraisal coming up later this year, "but the extra 2 hours we spent sitting around the table pointing fingers at each other didn't get us up and running any faster. That's what the APM solution is for, finger-pointing prevention."

The engineer chuckles, "…and emergency Memorial Day quit-the-barbeque-and-drive-in prevention too!"

That breaks the ice, giving them both a laugh out of their situation. From the outside looking in, TicketsRus.com appears like a single entity to its customers, providing a unified storefront for selling tickets. As a technology company, however, the reality of its internal struggles between teams is a constant battle.

John leaves the engineer's office and reflects a bit on the conversation as well as all the other similar conversations he's had with server administrators, developers, and even fellow members of management. This APM installation is more than just a technology insertion. Just getting this solution installed has been a lesson in professional growth as much as clicking Next, Next, Finish. John realizes that the sheer process of fitting his "magical" monitoring solution into TicketsRus.com's culture is in and of itself a maturing activity.

He just can't wait to see what the thing will look like as a finished product.

Implementing APM Isn't Trivial, Nor Is Its Resulting Data

As you can see in this chapter's story, integrating an APM solution into your environment is no trivial task. Although best-in-class APM software comes equipped with predefined templates and automated deployment mechanisms that ease its connection to IT components, its widespread coverage means that the initial setup and configuration are quite a bit more than any "Next, Next, Finish."

That statement isn't written to scare away any business from a potential APM installation. Although a solution's installation will require the development of a project plan and coordination across multiple teams, the benefits gained are tremendous to assuring quality services to customers. Any APM solution requires the involvement of each of IT's traditional silos. Each technology domain—networks, servers, applications, clients, and mainframes—will have some involvement in the project. That involvement can span from installing an APM's agents to servers and clients to configuring SNMP and/or NetFlow settings on network hardware to integrating APM monitoring into off-the-shelf or homegrown applications.

In this chapter's story, the network engineer refers to John's APM solution as "magical," yet the reality of its resulting situational awareness couldn't be any further the opposite. Rather than a source of any subjective mysticism, an APM solution enables a level of objective analysis heretofore unseen in traditional monitoring.

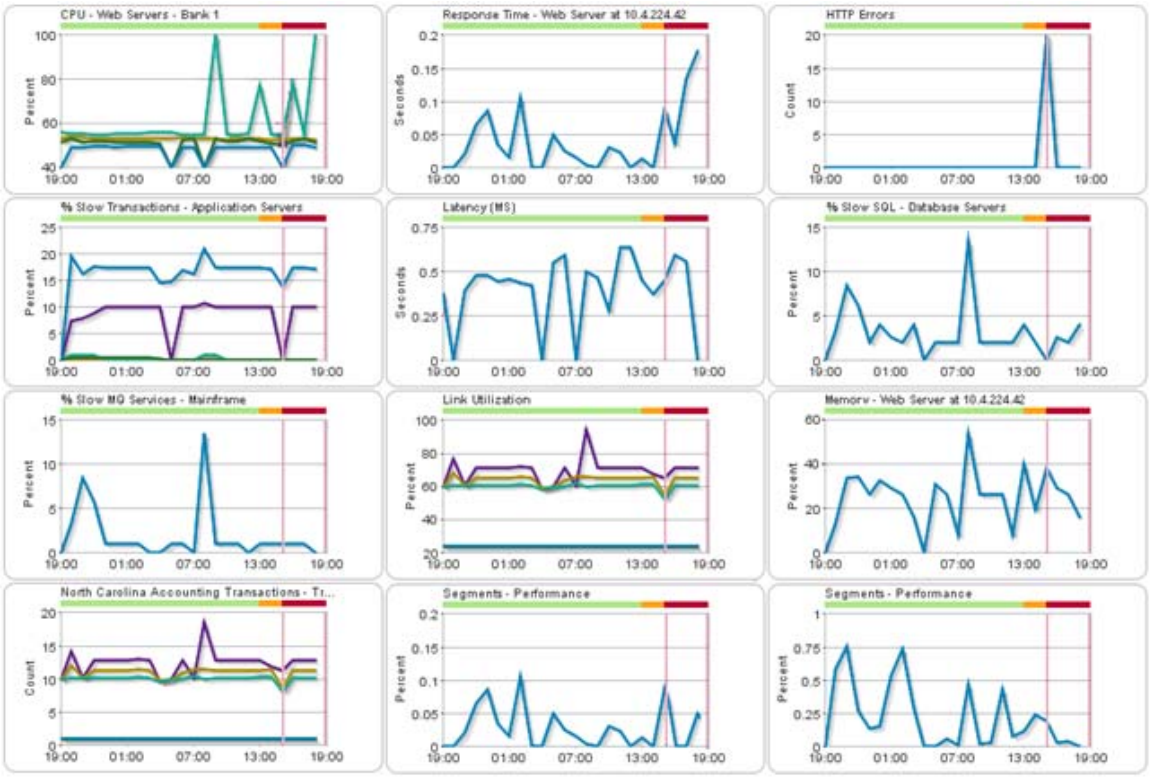

Figure 4.1: APM's integrations enables real-time and historical monitoring across a range of IT components, aggregating their data into a single location for analysis.

The realities of that objective data are best exemplified through APM's mechanisms to chart and plot its data. Figure 4.1 shows a sample of the types of simultaneous reports that are possible when each component of an application infrastructure is consolidated beneath an APM platform. In Figure 4.1, a set of statistics for a monitored application is provided across a range of elements. Take a look at the varied ways in which that application's behaviors can be charted over the same period of time. Measuring performance over the time period from 10:00 AM to 7:00 PM, these charts enable the reconstruction of that application's behaviors across each of its points of monitoring.

Chapter 8 will analyze graphs like this in dramatically more detail, running through a full-use case associated with a problem's resolution. But, for now, let's take a look at some of the information that can be immediately gleaned through the types of visualizations seen in Figure 4.1.

Finding Meaning in Charts and Graphs

Starting in the upper-right graph of Figure 4.1, at approximately 3:00 PM, the application experienced a dramatic increase in its number of HTTP Errors over a baseline of zero. This information in and of itself assists the troubleshooting administrator with recognizing that *a problem has occurred*. However, all by itself, it doesn't necessarily point towards that problem's solution.

With this information in hand, it is possible to cross-reference the problem's timeframe with other metrics that were collected at the same time:

- During and immediately prior to that same period—identified by the red vertical bar in each graphic—you can quickly determine that the problem was preceded by a spike in *Web server processor use*. This spike is found from the data in the upper-left graph.
- During and immediately after the problem, a spike in *Web server response time* was also experienced. This data can be distilled from the information presented in the top-middle graph.
- Perhaps as a result of the problem, or as one of its root causes, simultaneous drops were experienced in *network link utilization* (third row, middle), *percentage of slow transactions* (second row, left), and *accounting transactions* (fourth row, left).

With this information in hand, one can theorize that a natural correlation between these events has occurred. Some situation on the Web server appeared to cause the short spike in the HTTP error rate. That spike in error rate caused a simultaneous drop in processing, one that may have been noticeable by the application's users. The net result is a behavioral change on the part of the distributed application.

At this point, you should be thinking, "This analysis comes to a pretty obvious conclusion about the problem's culprit." Yet that's exactly the point that's important to recognize: *Your initial assumption of a system problem is often not the problem itself but merely a symptom of the problem.*

This example started out by looking at a recognizable spike in the HTTP error rate. Yet without a monitoring system in place to notify on this problem, your actual starting point for a problem like this might instead be with the accounting group and a phone call to the service desk. Perhaps that group noticed a short slowdown in their data processing. Perhaps the network's MRTG statistics showed a slight and unexplained dip in link utilization. Maybe a user called in with a concern that the application "was working slow today."

In all these cases, quickly identifying the correlation between the root cause of a problem and the down-level fallout from that problem is only possible using statistics across the gamut of that application's elements. To access this data, however, requires the involvement of individuals across the IT organization. Implementing APM's integrations into infrastructure components, network devices, and applications requires concerted effort. This chapter will discuss some ways in which those integrations are commonly inserted into your existing technology infrastructure.

The Tiering of Business Applications

Chapter 3 discussed how today's applications are much larger in scope and dramatically more complex than those of yesteryear. Today's business environment requires processing that is at the same time extremely data-driven, highly-available, and interconnected with multiple systems both in and out of control of that application's administrators. This complexity is necessary due to the distributed processing requirements of many customer-facing businesses today.

Consider for a moment the architectures that are required to build such systems today. Massive levels of redundancy are required, which at the same time bring higher levels of availability while adding higher levels of complexity. Tracing down a system problem grows significantly more difficult when that problem could have occurred on any one of many servers in a cluster.

The data needs of applications are similarly more challenging. Customer-facing applications require massive amounts of data along with corresponding levels of analysis during each click of the mouse. Today's applications may require the analysis of user usage patterns in real time, enabling the dynamic presentation of data based on the expected needs of users. This added processing can slow the user's experience if not properly architected.

Customer-facing systems also require real-time updates while leveraging third-party "cloud-based" solutions such as credit card processing. These external connections require additional care to protect the environment while ensuring the right levels of service from external service providers.

To give you an example of how a complex system like this might look, see Figure 4.2. This example represents a large-scale system not unlike what you might expect out of an e-commerce company like TicketsRus.com. This system contains multiple services and interconnections, including some that are out of the direct control of local administrators.

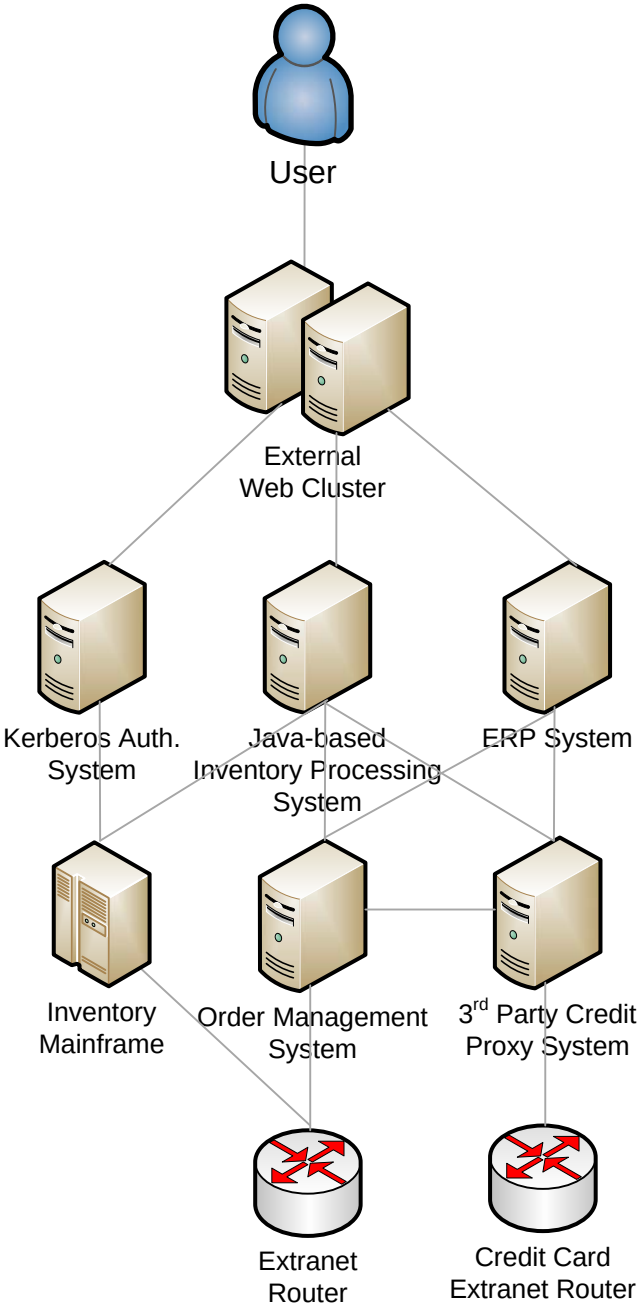

Figure 4.2: An example customer-facing e-commerce system with linkages to inventory mainframes, external credit card processing, and internal state/basket management.

In this system are a number of elements, each with a specific role to fill:

- In the first tier sits an externally-facing Web cluster that provides the front-end servicing of business clients. That cluster handles the presentation load from incoming clients, resting atop a second-tier Kerberos-based authentication system for the processing of user logins and passwords. This cluster is the primary point of entry for users to connect into the environment.
- Servicing the cluster is a set of second-tier systems. User data such as state and shopping basket information is stored within a separate second-tier ERP system. Inventory-processing functions have been offloaded onto a set of Java-based application servers.
- Mainframe and order management systems are in the third-tier. Here, product inventory information is stored as well as the processing of business logic associated with orders. This includes functions such as inventory, management of user shopping baskets, suggestions for alternate and/or complimentary products, and so on.
- A final system in this third tier, the 3rd Party Credit Card Proxy, manages the local processing for credit card orders. The job of this proxy is to work with the external credit card processing service, forwarding credit card information and receiving approvals.
- In the fourth tier is the routing equipment necessary for external connections to suppliers—used for real-time inventory updates and other supplier communication—as well as the third-party credit card processing facility.

This system is obviously highly comprehensive in the services it provides for its users and its business. It can display a list of inventory on a Web page. It can process user mouse clicks and present alternate and complimentary products to users based on their click habits. It can aggregate user-desired products into shopping baskets, and process their credit cards for the completion of purchases. It can even tie into supplier extranets for the automated updating of product information in real time. Whether for a company like TicketsRus.com or any company requiring a customer-facing e-commerce system, this example architecture is designed to provide the necessary kinds of functionality.

This tiering of clients to Web servers, Web servers to application servers, and application servers to databases is a common architecture in many of today's complex systems. Interconnecting these systems are networks, firewalls, and security devices that ensure the secure connectivity of data. That data is stored onto centralized storage in multiple places.

It is these types of systems that make excellent starting points for an APM solution. As a revenue driver, such solutions must remain highly available while at the same time providing their customers with an acceptable level of performance during their interaction. This is critically important in the case of customer-facing solutions, because when your system cannot perform to *your customers' demands,* you may find yourself losing business.

Business Applications and Monitoring Integrations

With the structure of Figure 4.2's business application in mind, consider the points of integration where you might want monitors set into place. You will definitely want to watch for server processing. You'll need to record your network bandwidth utilization and throughput. You need to know transaction rates between mainframes and inventory processing.

All these monitors illuminate different behaviors associated with the greater system at large, and all provide another set of data that fills out the picture you first saw in Figure 4.1's charts and graphs. Now take a look at Figure 4.3, where some of these monitoring integrations have been laid into place.

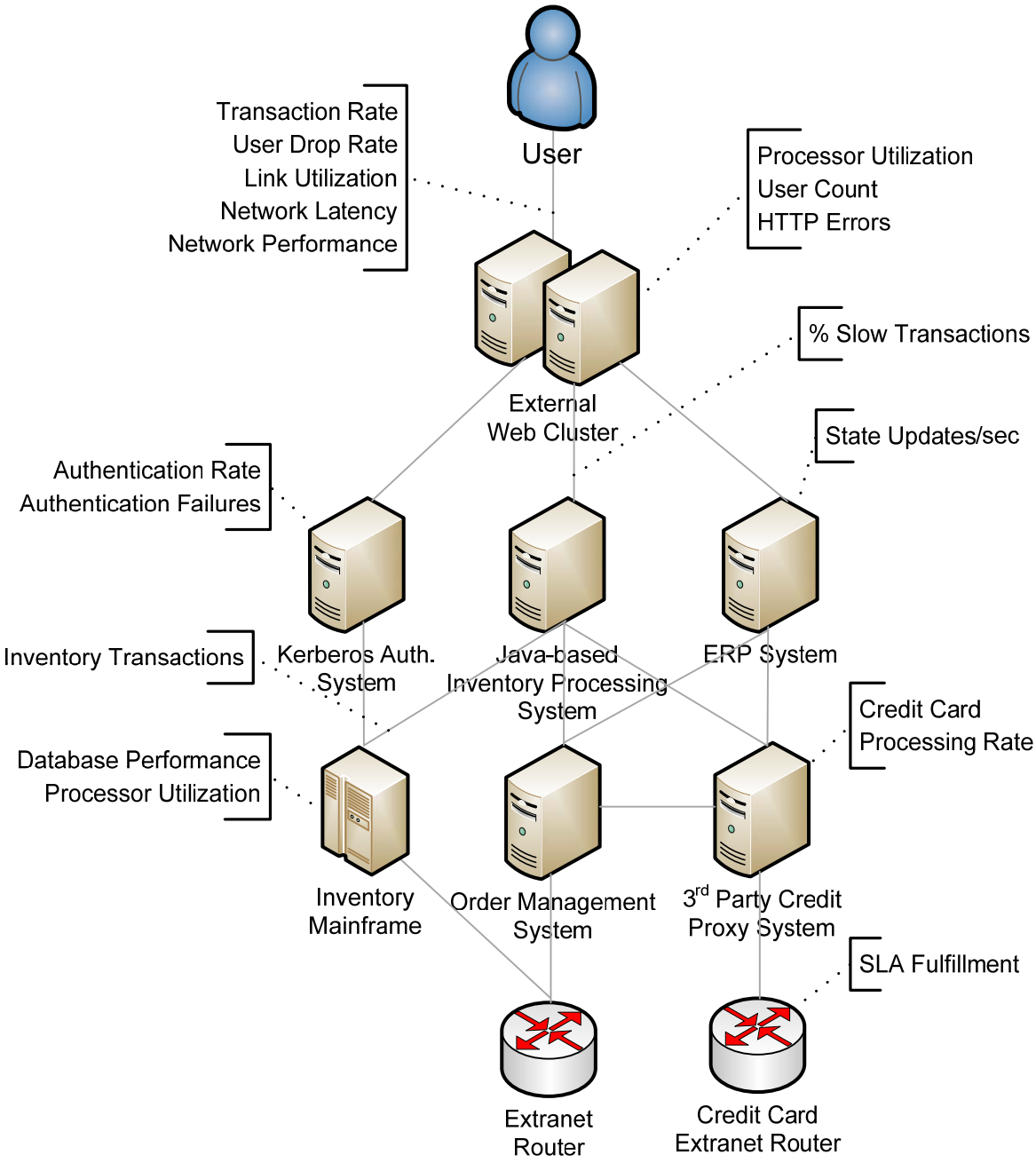

Figure 4.3: Overlaying potential monitoring integrations onto a complex system shows the multiple areas where measurement is necessary.

Installing System Agents

But what exactly makes up that picture of a system's health? How does the system integrate all this data to create a much larger picture of the application's quality of service? That information is first gathered by the individual monitoring integration. Consider first the data gathered through the agent-based approach.

Using agents that are installed directly onto individual servers, it is possible to gather metrics directly off those servers. Each server OS has its own mechanism for gathering and reporting on server-specific performance characteristics: The Microsoft Windows OS uses the Windows Management Instrumentation (WMI) service for gathering such information, storing it in a special area of the Windows registry, and presenting it to external servers through either external WMI queries or its WS-Management Web service. Event log data is stored in proprietary logs and presented through similar interfaces. Linux and UNIX servers leverage a combination of tools—vmstat, iostat, netstat, nfsstst, others—for the gathering and dissemination of data. Event log data can be gathered and distributed through the Syslog daemon.

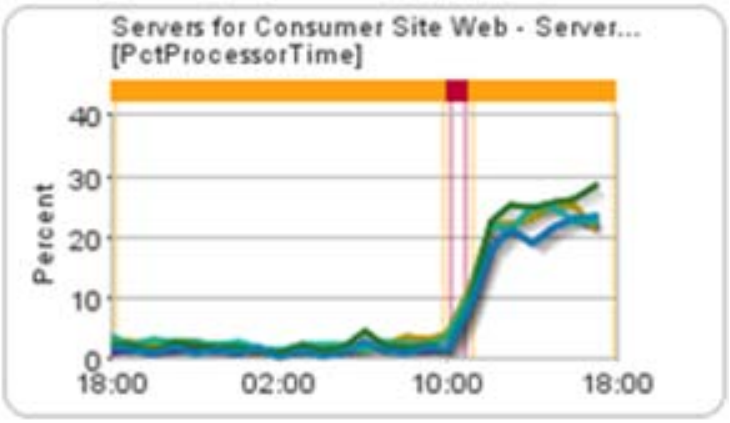

Figure 4.4: % Processor Time for a Web server, as gathered by an agent on the Web server itself.

In all these cases, installed agents on each server are empowered with gathering the right kinds of data and reporting this data back to central APM servers. The net result is the creation of graphs similar to Figure 4.4, where % Processor Time for the Web server is shown over a 24-hour period of time. In that graph, it should be apparent that the Web server's utilization grew dramatically beginning at 10:00 AM during the day of measurement.

The process of actually installing agents to the correct servers in your environment is one important facet of your APM installation and should be included as an action in your project plans. Depending on the type of APM solution selected, that agent may require a manual installation or may be automated through its console. Obviously, each added agent to a system involves slightly more of its resources that are consumed for management functions. To reduce the number of agents on a system, some APM solutions enable the ability to leverage data from other monitoring solutions. Using this "monitor-of-monitors" approach, the APM platform gathers its data instead from another in-place monitoring solution. In the end, your selection of APM solution should consider its abilities to integrate with the existing management platforms that may already be present in your environment.

Beware Resource Use by Agents Themselves

Agents themselves should also be carefully screened to ensure that their processing doesn't negatively affect the overall processing on the server. The actions completed by the agent in collecting performance and log data will require a measure of processing power on the computers where they are installed; however, the level of resource use by any APM agent should be minimal. Effective APM agents should reside in the background and not have a dramatic impact on server performance. Prior to deploying any APM solution, test the functionality of its agents and verify that those agents themselves will not cause a negative impact to your servers' performance.

Augmenting Agents with Application Analytics

Agents on the server needn't gather only system-focused performance metrics. Smart agents are those that have been augmented with the capacity to gather performance and other behavior characteristics from common middleware applications and databases as well.

For example, an environment's Order Management System might run atop an Oracle database. It is entirely possible that server-centric statistics such as % Processor Time cannot discretely capture the internal behaviors of the Oracle database. Perhaps that database is processing a large number of "bad" records that impede its ability to correctly work with the good ones. Maybe the application's individual queries are not correctly optimized. In both of these situations, it is feasible that an Oracle-specific behavior doesn't directly manifest into server-centric metrics. Necessary are deeper integrations, built-in to the installed agent, which can query Oracle's native performance statistics for additional data.

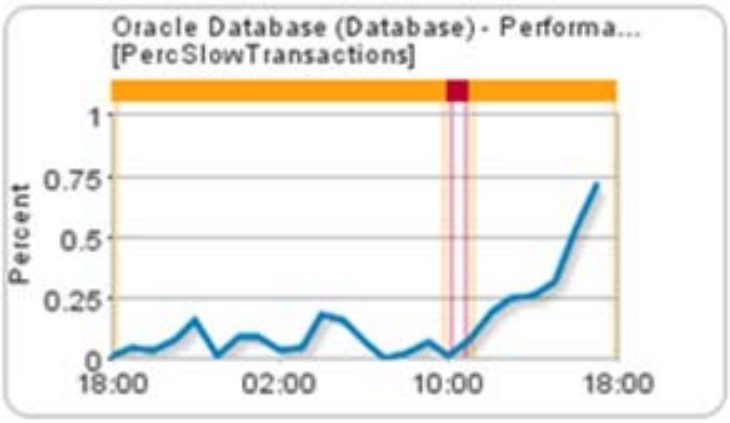

Figure 4.5: Agent integrations into the Oracle database itself display more specific data about the database's behaviors.

Figure 4.5 shows a graph of information gathered from just that Oracle database. Here, the same time period is shown as in Figure 4.4 but the gathered information instead shows the percentage of "slow" transactions as defined by the administrator and experienced within the Oracle database. As with Figure 4.4, the area of concern has been highlighted between the red bars immediately after 10:00 AM.

In this second graph, it should be obvious an increase in the rate of slow transactions is highly correlated with Figure 4.4's increase in processor utilization. The combination of these two graphs illuminates a greater level of detail about how one application's behavior can have an impact on overall system performance. In this case, perhaps there is extra processor use required to deal with slow transactions. Or, the individual transactions being processed by the database are particularly complex or are not properly optimized for performance.

When looking at an APM solution, look for one whose agents are enabled to support your needed middleware applications and databases. By incorporating application- and database-specific integrations directly into agents, it is possible to discover more detailed information about the inner workings of these otherwise-opaque systems.

Although this capability is useful when the number of supporting applications and databases is small, it grows substantially more useful as their count increases in the environment. Figure 4.6 shows a rollup visualization that details the level of performance across a set of applications.

Software service	Usage		Performance	
	Unique users	Total bytes ▼	Application performance	Affected users
Frontline	1.15 k	6.55 GB	96.8 %	108
Salesforce.com HTTP	4	3.87 GB	-	0
Exchange Inter-Site SMTP	232	3.79 GB	69.3 %	31
Changepoint SQL	8	3.15 GB	99.9 %	4
Changepoint	369	2.79 GB	99.7 %	29
Oracle Financials DB	9	2.76 GB	99.9 %	4
Compuweb	184	893 MB	98.8 %	8
Changepoint (Active)	15	702 MB	100 %	2
Exchange - Outlook	637	640 MB	99.9 %	63
Adobe Forms	76	276 MB	94.5 %	19

Figure 4.6: Viewing the performance across a set of applications in a single visualization.

The graphic here details the behaviors across ten different applications that may be involved in a particular infrastructure or business service. Specific to each application is information about the number of users who may use that application, the total amount of data used by the applications, its aggregate performance, as well as the number of users that may be affected by that application should it experience a problem.

The percentages displayed in the column marked "Application Performance" relate to that application's instantaneous performance. To create these percentages, an application administrator or predefined template must identify which performance metrics make sense for that application's measurement. Also needed are the threshold values for identifying when an application is not performing to expected levels. Your APM solution should provide the internal mechanisms for identifying these thresholds.

The net benefit of these calculations arrives during normal operations. When one or more applications' performance levels degrade past acceptable levels, administrators can use the information in the "affected users" column to prioritize their resolution effort to those with the highest impact on users.

Configuring Devices for Network Analytics

As you've already learned, servers and their applications are only one source of a system's overall behaviors. You simply can't get a comprehensive view of the network without additional integrations into the network itself. Those integrations enable a look at conditions such as bandwidth utilization, latency, and the all-important Web site performance statistics, among others.

The actual gathering of these types of network statistics can be enabled through a number of solutions, many of which are likely already available on your network hardware today. Virtually all of today's business-class network hardware natively includes the support for SNMP integration. With this information in hand, a centralized network monitor can pull statistics directly from networking equipment through SNMP polls.

Yet SNMP is only one solution. SNMP alone cannot provide the right kind of information associated with network traffic "flows," meaning the overarching conversations between elements on the network. Flow data monitoring can be considered a superset of that seen by measuring individual packets. It provides a view of network traffic at a higher level than at the individual packet, yet not to the level of inter-server transactions.

To provide this level of data, additional protocols have been developed such, as Cisco's NetFlow, that report on high-level "flows" rather than packet-level inspection. Although still not looking at this data from the level of the server-to-server transaction, flow information gives the troubleshooting administrator a better sense of their network's high-level traffic patterns.

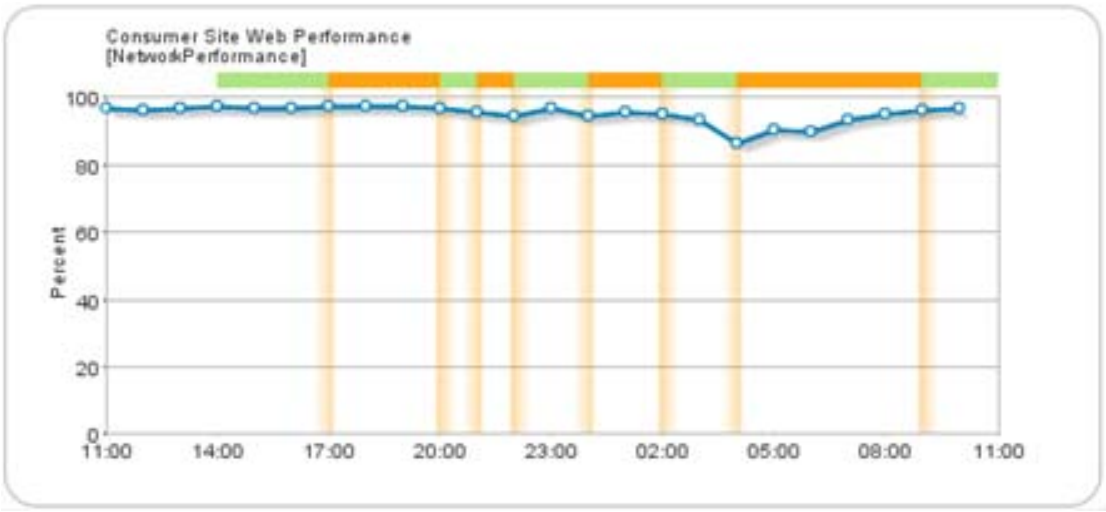

Figure 4.7: Another view of the system from the network's perspective shows that network performance is nominal during the period of measurement.

An example of this kind of data is displayed in Figure 4.7. Here is shown a view of the impact of the network during a slightly offset period of time. In this graph, it is possible to quickly see that the network experienced a slight dip in performance between 2:00 AM and 5:00 AM. That performance returned back to baseline by 10:00 AM, when the database began experiencing problems. This information is critically useful for the troubleshooting administrator, as it quickly shows that high-level network performance doesn't appear to have a direct impact on the system's problem.

As with application monitoring, aggregate views of the network are also possible once network integrations are laid into place. One such view is shown in Figure 4.8. Here, aggregate network statistics across multiple sites are shown in a single view, with associated trending arrows attached to each.

Area	Unique local hosts	Total bytes	Upstream bandwidth usage	Downstream bandwidth usage	Local RTT	Upstream loss rate	Downstream loss rate
Michigan	1.61 k	108 GB	9.82 Mbps	8.39 Mbps	2.5 ms	0.2 %	0.2 %
Brazil	37	530 MB	13.2 kbps	74 kbps	214 ms	0.2 %	0.6 %
Ohio	37	1.65 GB	23.5 kbps	255 kbps	107 ms	0.2 %	0.9 %
Wisconsin	31	432 MB	10.8 kbps	60.3 kbps	88 ms	0.2 %	1 %
Canada	30	375 MB	19.7 kbps	41.9 kbps	86.4 ms	0.2 %	0.3 %
North Carolina	23	1.09 GB	27 kbps	156 kbps	99.8 ms	0.7 %	0.7 %

Figure 4.8: Aggregate network statistics across multiple sites are shown, with trending arrows showing prior behaviors.

Graphs such as this are only possible when network statistics from across the environment are gathered into a single location. By integrating each individual network device's statistics into an APM framework, an administrator can quickly pinpoint network errors and transfer rates, further isolating a problem to particular network segments.

Installing Probes

Chapter 3 introduced the concept of probes and the efficacy of their use in network monitoring. Your decision to use network probes will be first dependant on the capabilities of your networking equipment. Networking equipment that cannot support built-in integrations (which are rare these days) or network security policies that prohibit the passing of monitoring data are both situations where probes may be necessary.

Probes are by nature operationally expensive to use in a production environment due to their in-line installation. A network probe by definition is a separate physical device that watches the traffic that passes by a particular network segment. Figure 4.9 shows a network diagram of how one can be installed between an internal LAN and an external WAN point of demarcation. As such, their installation requires a manual change to the network. Thus, limiting their use to situations where they are specifically required is considered a best practice.

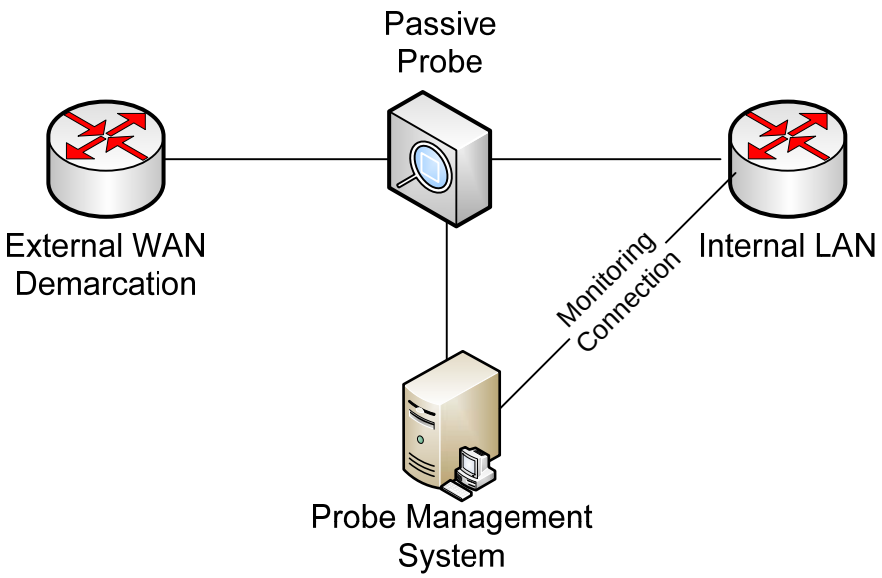

Figure 4.9: A passive network probe is installed between an internal LAN and its connection to an external WAN.

One situation where probes can provide special data not otherwise possible through in-device integrations is in measuring traffic across network links that are not within local IT control. For example, Figure 4.2 showed a connection between a third-party credit card proxy and a router to the extranet that is shared with the credit card processing service. Often, these kinds of connections are not within the direct control of the local IT organization, which makes problematic the installation of on-device monitoring integrations. In these cases, network probes can be physically installed between otherwise-uncontrollable connections to monitor their traffic for inconsistencies.

Figure 4.3 showed a particularly useful example of such an installation. Here, a probe can be installed with the intent of monitoring and alerting for Service Level Agreement (SLA) breeches between the business and the external processing network. This adds a level of due diligence during SLA breech negotiations as well as added protection against the impact of external actions such as outages or performance losses.

Measuring Transactions

Yet another perspective on system behaviors occurs when the monitoring solution is pointed towards the connection between the Web site and its Inventory Processing System. The view enabled through this integration combines otherwise packet-based data to look at the individual transactions between these two servers. Although the packet-based data is useful for recognizing the overall behaviors of the network in relation to that individual connection, transaction measurements provide more details about the specific "conversations" between servers and services on two different hardware components.

Take a look at Figure 4.10 for another chart that details this view of individual transaction rates. Here, you can see that immediately prior to the time in question, a spike in transactions began to occur between the two systems. Perhaps the timing of this transaction spike gives some added details about the original processor utilization spike from Figure 4.4. Perhaps an added spike in use by users was a cause of the problem rather than an internal issue.

Transaction monitoring is critically necessary in distributed systems, most especially those that use the Web for the display of data. Web-based data is highly transaction-oriented, so great levels of detail can be gathered by watching those transactions as they go by on the server. As you'll discover later on, this aggregate transaction monitoring can be expanded even further by looking at the individual conversations as they pass on the wire.

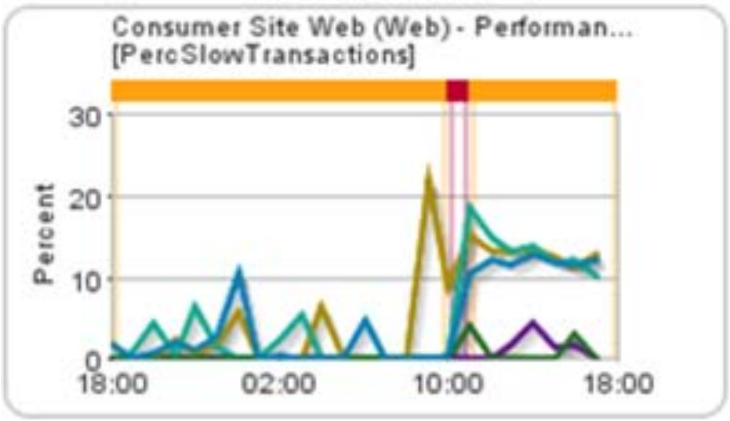

Figure 4.10: Elevating the view from individual network packets shows additional information about the conversations between servers.

When thinking about this concept of transactions, it is important to consider the individual conversations between the different elements on a system. Consider, for example, the types of conversations that could potentially occur between a generic Web server and its Inventory Processing System. Those conversations have both a source and destination IP address, but they also operate over a known set of TCP or UDP ports. Within a single set of ports, individual Web services at both ends can transfer multiple types of information, with the ultimate end consumer of this information being different services on each server.

Your APM solution should include a set of network filters that works with both agent and agentless monitoring to identify these conversations. Only by leveraging the right network filters can a system gather meaning through the combination of individual packets of information. Those filters must understand the protocols being used by both sides of the communication as well as the well-formed data used in their conversations.

Continuing the example from Figure 4.10, let's assume an administrator wishes to see the actual "words" in the transactional communication between the Web cluster and the Inventory Processing System. To do so, they must drill down even further. By correlating the network probe's "external" view of a transaction's performance with the "internal" application analytics perspective from a server-resident agent, it is feasible that a view similar to Figure 4.11 can be created.

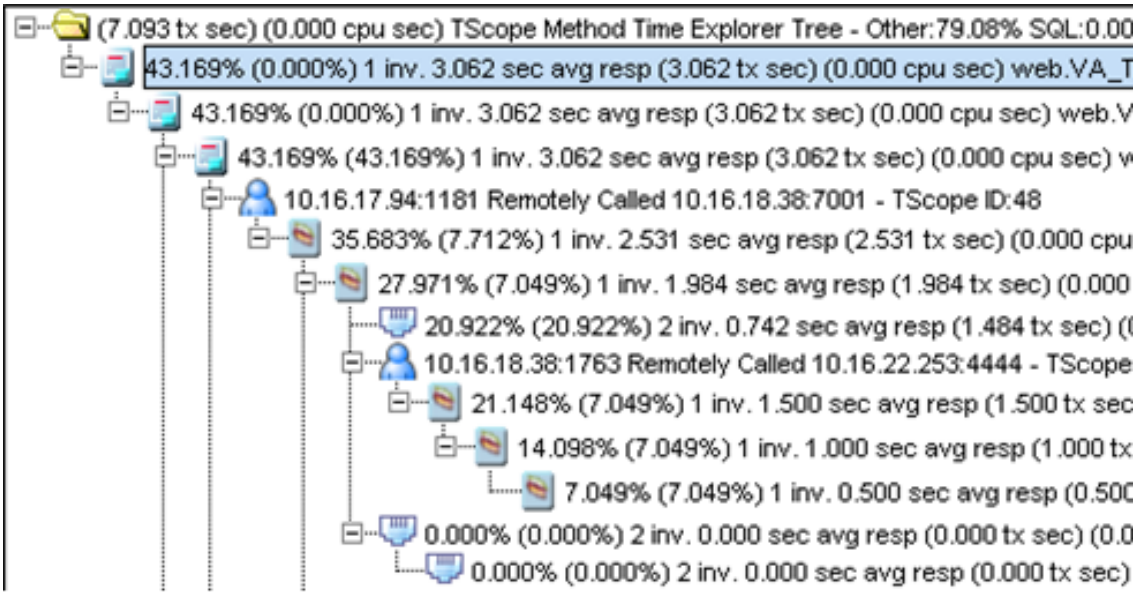

Figure 4.11: Zooming in on a particular time slice of transactions provides detail about the "words" in their communication.

Figure 4.11 shows a truncated view of the details of that conversation. Here, the discussion between the two servers is drawn out in detail. The Web server attempts to contact the Inventory Processing System, with the processing system eventually responding. Statistics associated with delay timing is displayed in addition to the conversational details. The result is a kind of time-oriented log of the conversation within and between the two servers, showing exactly where areas of delay are found.

A result of this level of detail is that conversational elements and areas of unacceptable time lag can be identified within the individual code elements of each server's Web services. For example, if a particular callback from one server to another shows a high rate of delay, while the network itself can be eliminated as a source of lag, it grows very easy to flag the situation to developers. Ultimately, the end goal is to quickly identify the problem and come to a resolution that can be patched into the system.

The actual implementation of transaction measurements happens in parallel with the installation of agents and the incorporation of agentless monitoring across the network. When network monitoring is configured to watch for specific conversations between selected servers, this information can be captured in detail. At the same time, onboard agents have the ability to monitor data as it passes in and out of the network interface cards (NICs) on the servers themselves. In your APM solution, transaction-based monitoring should be a function of both agent collection as well as administrator console configuration. Another task in your implementation will be the identification of the types of traffic to collect as well as the devices between which to collect it.

Overall Service Quality

One end goal of all this monitoring is the ability to create an overall sense of system "health." As should be obvious in this chapter, an APM solution has a far-reaching capability to measure essentially every behavior in your environment. That's a lot of data. A resulting problem with this sheer mass of data is in ultimately finding meaning. Essentially, *you can gather lots of data, but it isn't valuable if you don't use it to improve the management of your systems.*

As a result, APM solutions include a number of mechanisms to roll up this massive quantity of data into something that is useable by a human operator. This process for most APM solutions is relatively automatic, yet requires definition by the IT organization who manages it.

The concept of "service quality" is used to explain the overarching environment health. Its concept is quite simple: Essentially, the "quality" of a service is a single metric—like a stoplight—that tells you how well your system is performing. In effect, if you roll up every system-centric counter, every application metric, every network behavior, and every transaction characteristic into a single number, that number goes far in explaining the highest-level quality of the service's ability to meet the needs of its users.

Figure 4.12: The quality of a set of services is displayed, showing a highest-level approximation of their abilities to serve user needs.

This guide will talk about the concept of service quality in much greater detail over the next chapters, but here it is important to recognize that implementing service quality metrics also requires the involvement of the APM solution's implementation team. Consider the graphic shown in Figure 4.12. Here, a number of services in different locations are displayed, all with a health of "Normal." This single stoplight chart very quickly enables the IT organization to understand when a service is working to demands and when it isn't. The graph also shows the duration the service has operated in the "normal" state, as well as a monthly trend. This single view provides a heads-up display for administrators.

Yet actually getting to a graph like this requires each of the monitoring integrations explained to this point in this chapter. The numerical analysis that goes into identifying a service's "quality" requires inputs from network monitors, on-board agents, transactions, and essentially each of the monitoring types provided by APM.

Also required is the configuration of threshold values by administrators *at each level of monitoring*. Figure 4.12's view of system health comes through the aggregation of individual down-level health monitors, each of which must be implemented and configured by administrators. One implemented, administrators must determine your application's baseline performance values in comparison with those that they consider "acceptable."

For example, if processor use on servers becomes unacceptable when it goes over 80% for a 5-minute period of time, this behavior must be specifically ingested into the APM platform. The same holds true for network behaviors: If network bandwidth utilization over 85% is considered an unacceptable situation, it too must be configured into the system. The aggregation of each of these thresholds ultimately combine to create the service quality metric shown in Figure 4.12.

Although this process can seem extremely time-intensive, effective APM solutions speed the process by incorporating industry-standard templates for common thresholds. The inclusion of these templates assists administrators with a starting point for later customizing the unique characteristics of their environment. More on this concept of service-centric monitoring will be covered in Chapter 6.

APM's "Magic" Is in Its Metrics

This chapter started out with a conversation between TicketsRus.com's IT manager John Brown and his network engineer. In that conversation, the engineer referred to John's APM solution as some form of "magical" monitoring. Yet the engineer couldn't be further from the truth in his representation of APM's efficacy. An APM solution is by nature extremely objective. It enables the centralized gathering of vast quantities of data for numerical analysis. When users call to complain that "the server is slow today," an APM solution enables IT teams to understand why, or in many places, already be working on the problem.

You may have noticed that there is one major omission in this chapter's discussion on implementing APM. End-User Experience (EUE) monitoring is one topic not found in this chapter. Due to its relatively new entrance into the market as well as its potential for wide-sweeping changes in the way services are monitored, this technology gets an entire chapter to its own. The next chapter will discuss EUE in detail, talking about its technology underpinnings, where it fits into the environment, and the types of data that can be gathered as a result of its implementation.