Realtime publishers

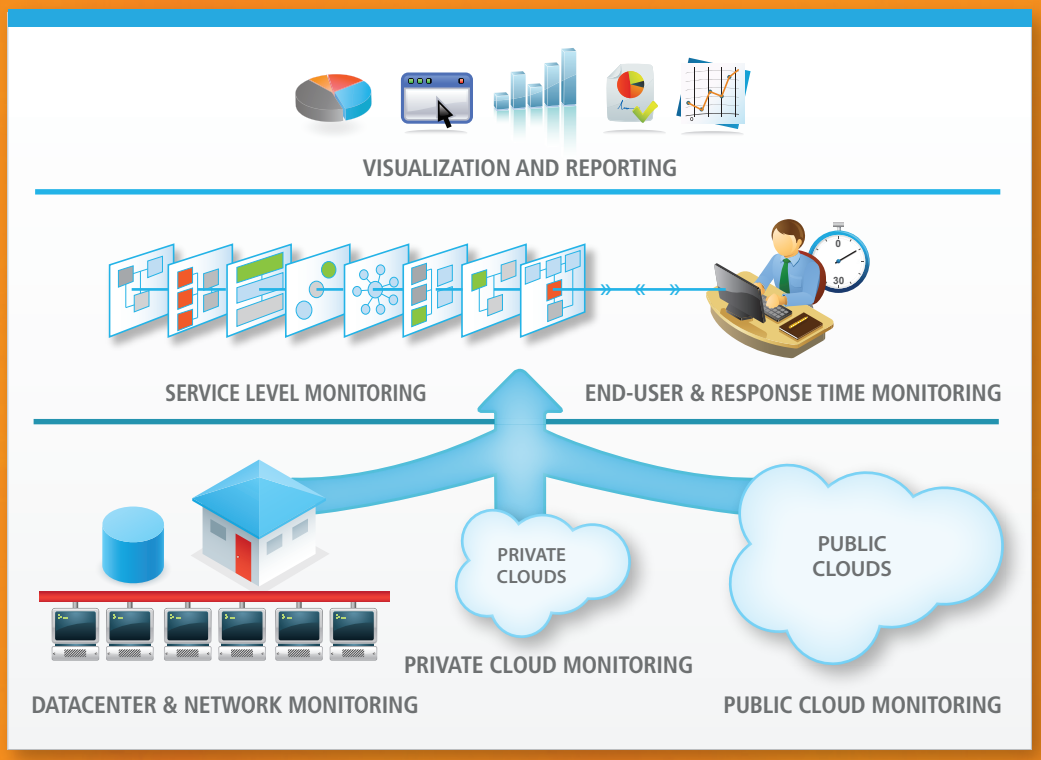
The Definitive Guide™ To

Monitoring the Data Center, Virtual Environments, and the Cloud

sponsored by

Don Jones

The Nimsoft Monitoring Solution



- **Ensures business service delivery regardless of IT platform**
- **Enables rapid adoption of new computer infrastructure such as private and public cloud**
- **Monitors the datacenter to the cloud, including SaaS, hosted, and virtualized environments**
- **Lowers TCO by 80% and delivers proven value in weeks**

Unified Monitoring

nimsoft
From the Datacenter to the Cloud

www.nimsoft.com

Copyright Statement

© 2010 Realtime Publishers. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtime Publishers (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtime Publishers or its web site sponsors. In no event shall Realtime Publishers or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtime Publishers and the Realtime Publishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtime Publishers, please contact us via email at info@realtimepublishers.com.

[**Editor's Note:** This book was downloaded from Realtime Nexus—The Digital Library for IT Professionals. All leading technology books from Realtime Publishers can be found at http://nexus.realtimepublishers.com.]

Chapter 3: The Customer Is King: Monitoring the End User Experience

I've already presented the End User Experience (EUE) as the ultimate metric for an application's overall performance, whether that application lives entirely in your data center, entirely in the cloud, or in some combination of the two. In this chapter, I'll explore the EUE in greater detail: How do you actually measure it? What, specifically, are you looking at? What contributes to a good—or poor—EUE in an application? If you see your EUE metric starting to head toward "poor," what can you do about it? I'll also examine some of the reasons companies traditionally haven't measured the EUE—and why doing so can still be tremendously difficult, especially in newer, highly-distributed applications that involve elements of cloud computing.

Why the EUE Matters

First, however, let's really pin down why the EUE is so important. As I outlined in the previous chapter, businesses have typically measured application performance solely from technical measurements—database response times, for example. What does the EUE really offer that the more traditional measurements don't?

Business-Level Metric

Take a look at Figure 3.1. It's a common-enough chart in a business technology environment, displaying a variety of performance metrics for a database. The bottom graph shows, in orange, physical reads from disk. I guess those look normal. The middle graph shows, also in orange, logons to the database. Had a little peak around 6pm and 5:30. Guess that's okay. The top graph shows a variety of statistics, primarily user input/output in blue and CPU utilization in green. That's a lot of I/O, I guess. CPU looks okay. At least, it's lower than the top of the graph.

Figure 3.1: Database performance.

So what does all of this mean *to the business?* That's harder to gauge. Are we making money or losing money? Do the users of our application believe it's performing well? Do we have phone agents sitting on the phone telling customers, "I'm sorry, the computer is slow today," or is everything popping up pretty quickly for them? It's impossible to tell from this chart.

What about Figure 3.2? This is a VMware vSphere performance graph showing CPU utilization for a week. Got a little spike late Monday, but there's no way to tell whether that impacted our business operations. In fact, it might have been late at night, so possibly it was related to a maintenance task or anything. It's impossible to tell if any users were impacted, though.

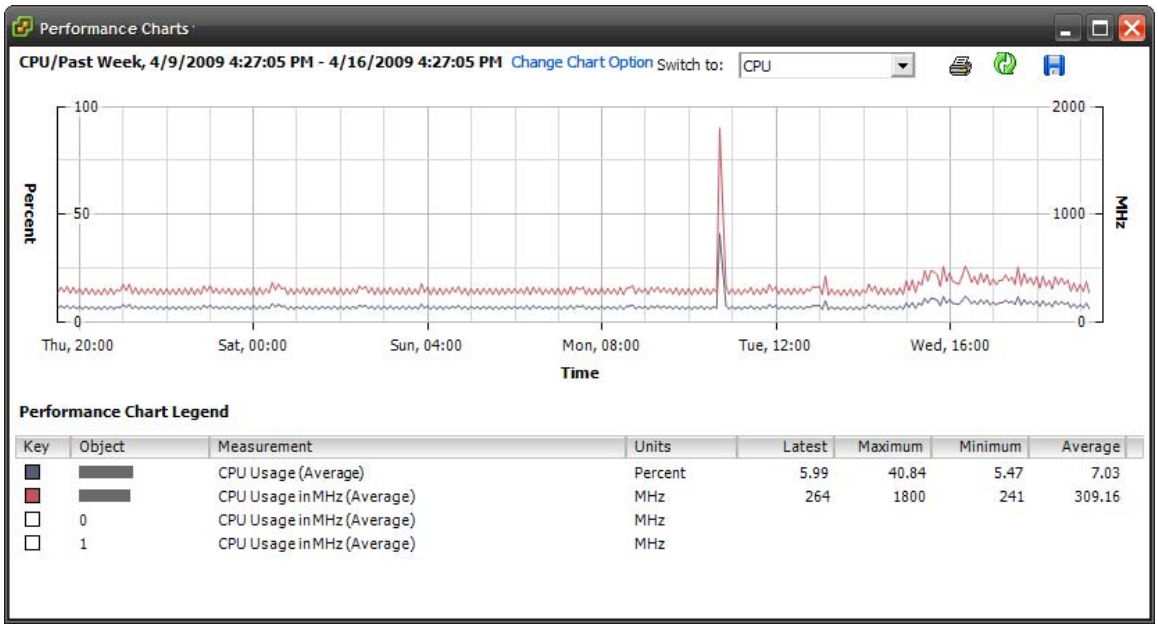


Figure 3.2: Virtualization host CPU performance.

These two examples precisely illustrate the problem with traditional performance measurement: There's no tie to *anything that matters to the business.* Memory, I/O, CPU, disk—none of these things tell us whether the application is performing well *for the business.* Sure, we could add some thresholds to those charts, maybe generating an alert when CPU utilization spiked above 80%, as it did in Figure 3.2. But that still doesn't tie directly to what matters to the business.

Here's how businesses have traditionally tried to make these technical measurements relate to business concerns:

1. We observe technical performance metrics.
2. When we start getting end user calls about application performance, we make a note of the performance metrics at that time and draw a threshold line.
3. From then on, if performance is on one side of the threshold, we assume that equates to good end user performance. On the other side, we assume it means poor end-user performance.

The problem with this approach is that we can't continuously measure *every* technical element that contributes to good or bad end user performance. We continue to use technical performance elements—CPU, disk, memory, and so forth—as our only real metrics, even though we can't *directly* relate them to anything the business cares about.

> **Note**
> It's even worse when your end users don't work for your company because in that case, you might never know that there's a perceived problem. For example, if your customers feel that your online shopping cart is too slow, they won't call you—they'll simply give up and shop elsewhere. Only measuring technical elements can make it easy to miss the fact that you're losing sales and customers.

Tied to User Perceptions

As I discussed in the first chapter, end users don't usually care about CPU utilization and disk throughput: They care about whether the application seems slow. They base their judgment on how long it takes them to complete common transactions, such as looking up a customer order, completing a checkout, and so on. Ultimately, your end users are the ones who can tell you whether your application is performing well and doing its job—unfortunately, end users have some significant problems:

- They aren't consistent—One user might feel the application is performing well, while another feels it's too slow
- They aren't accurate—An application with identical performance might be perceived as slightly slow one day and just fine on another
- They don't always report—Even users within your own organization tend to not report poor performance because they usually get a brush off; external users—such as customers—will generally just take their business elsewhere

That's why *monitoring* the EUE is so important. You get that "end user perspective," which is ultimately the only thing that matters to the business. But you get it more accurately, more consistently, and without the need for users to actually report in on their own.

This is really where the term *application performance management (APM),* comes from. In fact, let's create a formal definition:

> APM can be defined as the process and use of related IT tools to detect, diagnose, remedy, and report application's performance to ensure that it meets or exceeds end users' and businesses' expectations. Application performance relates to how fast transactions are completed on behalf of or information is delivered to the end user by the application via a particular network, application, and/or Web services infrastructure.

Challenges as You Evolve to Hybrid IT

Hybrid IT, as I described in the previous chapter, is the evolution of business technology services that incorporate a variety of highly-distributed technologies; *super-distributed* is another term that refers to this evolution. In the previous chapter, I outlined how our applications have evolved from monolithic code that ran on a single computer to today's applications that rely on resources in the cloud, from your own data center, and from service providers. Figure 3.3 illustrates this kind of application.

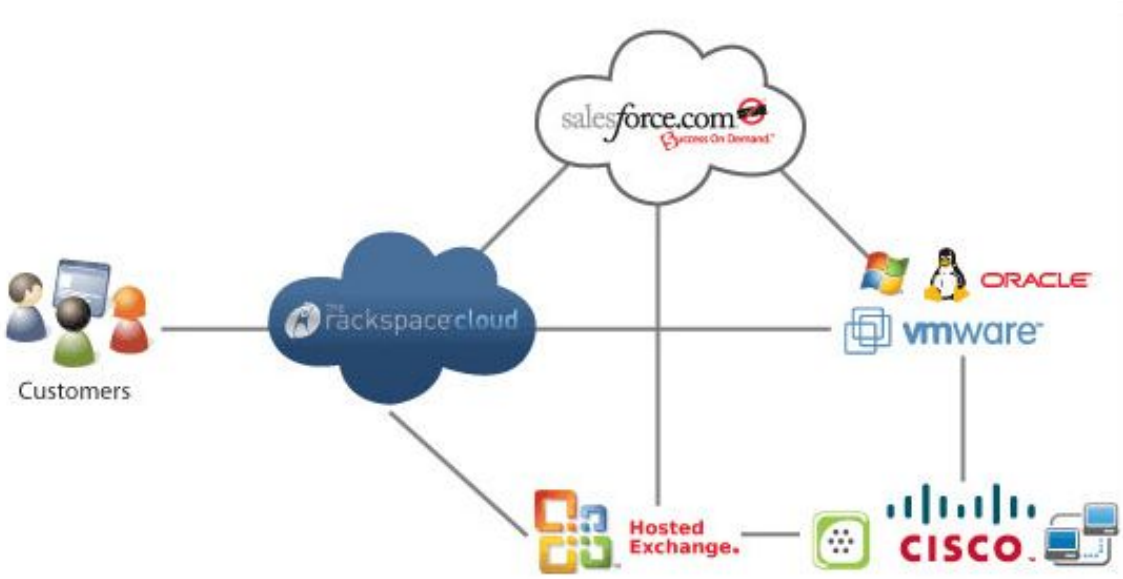


Figure 3.3: Modern "hybrid IT" application.

With this kind of application, customers interact with a Web site that is hosted by a Web hosting company—Rackspace Cloud, in this example. That application is not self-contained, however. It depends on a Software as a Service (SaaS) offering from SalesForce.com to track customer information, and sends email messages through a hosted Microsoft Exchange service. On the backend, additional services—virtualization hosts, Windows servers, Linux servers, Oracle databases, and network elements—provide data and support to the application. This application depends upon multiple data centers, numerous disparate software elements, and more. Monitoring this by using traditional technology-centric techniques is impractical if not impossible, meaning the only way to make sure our customers are happy is to measure the EUE directly. What specific challenges come along with this kind of super-distributed application?

Geographic Distribution

One significant problem is the geographic distribution of today's more complex applications. For example, suppose you've decided to host a Web service or Web site in the Windows Azure cloud. *You have no idea where your application is physically located.* The whole point of the cloud, in fact, is to make your application available more globally so that users all across the world can access it more or less equally.

For example, consider an application that's hosted in a more traditional fashion, in a shared hosting environment, or on a dedicated server that's located in a hosting company's data center. One server, or one group of collocated servers, hosts the application. Monitoring the EUE is straightforward because you've only got one thing to manage. Figure 3.4 illustrates.

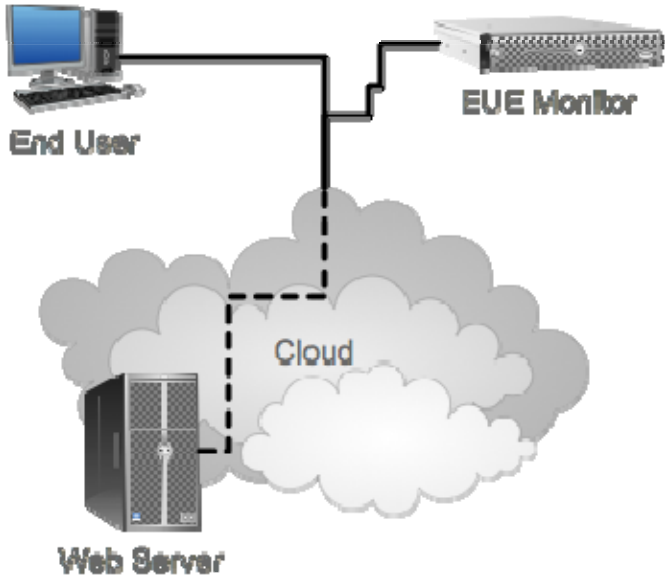


Figure 3.4: Monitoring the EUE in a single-location application.

Move into the cloud, however, and your application is inherently able to be hosted—transparently—on multiple servers *that are geographically distributed.* Figure 3.5 illustrates the difficulty in monitoring the EUE: An EUE monitor in a given location will *probably* be connecting, *most* of the time, to a geographically-close server; end users in other locations, however, may be connecting to entirely different servers across entirely different infrastructure. Your EUE monitoring won't be getting an accurate picture of the entire application's performance because it's only seeing part of it.

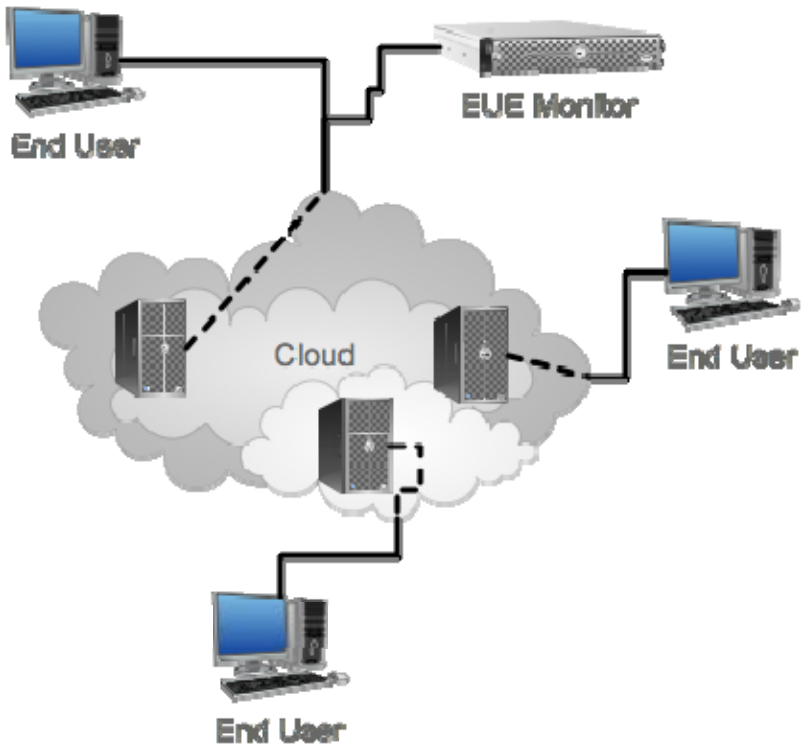


Figure 3.5: Geo-dispersed cloud applications.

You could, of course, start deploying distributed EUE monitors as well. But that means you have to start building a giant, globally-distributed monitoring infrastructure—and I'll go out on a limb and guess that doing so isn't part of your business' overall goals.

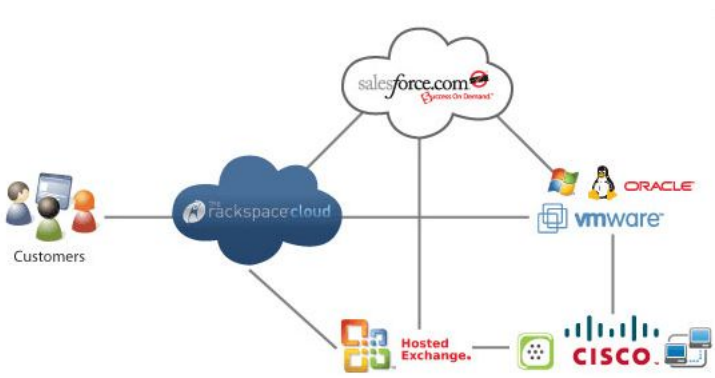
Another option is to work with service providers—such as cloud hosting companies—that *provide* you with distributed monitoring capabilities, meaning they've deployed those capabilities within their own infrastructures and make data related to your application available to you. If you *are* a service provider, that's a very competitive feature to offer to your customers.

The last option is to find a monitoring solution that *comes with* a globally-distributed monitoring service. In other words, rather than just buying yet another monitoring console, you're buying a console that integrates with an in-place ability to monitor distributed applications hosted in commonly-used services, such as specific cloud computing providers. As Figure 3.6 shows, that can give you access to performance information on these cloud providers *without* requiring you to deploy your own extensive monitoring infrastructure.

Figure 3.6: Monitoring cloud provider performance.

Deep, Distributed Application Stacks

Figure 3.3 showed how complicated today's applications can really become, relying on numerous hosted services, different SaaS offerings, and on your own backend infrastructure—which may be running on a variety of operating systems (OSs), virtualization hosts, and so forth. Every single element in this stack can contribute to performance problems, and even if you're monitoring the EUE directly, you'll still want insight into individual component performance for troubleshooting purposes. Considering just the example in Figure 3.3 (which I'll repeat here for your convenience), you're looking at a huge variety of things to monitor:



- SalesForce.com's overall responsiveness
- Performance of your Web site hosted in the Rackspace Cloud—which may involve a quantity of servers that expands and shrinks transparently in respond to demand, and which likely involves servers that are globally distributed
- The performance of the hosted Exchange service
- The performance of your own Windows and Linux OSs, including their memory, CPU, disk throughput, and so forth
- The responsiveness of your backend Oracle database

- The performance of the VMware virtualization infrastructure that hosts some or all of your servers
- The performance of related services such as a Cisco VoIP solution
- The performance of the various Internet connections that let all of these components communicate

That's a lot. Although measuring the EUE doesn't necessarily require deep insight into this highly-distributed application stack, you *will* need that deep insight in order to tune performance and troubleshoot problems. We'll cover that in more detail in the next chapter.

Techniques for Monitoring the EUE

So how do you go about actually monitoring the EUE? What tools are needed, and what skills are involved? How do those tools actually gather EUE information, store it, and present it to you? Because today's applications are themselves constructed from numerous different components, we'll have to adopt a number of techniques and technologies for monitoring the EUE.

But let's first talk about the unachievable ideal way to monitor the EUE: A little monitoring agent installed on all your users' computers. That might be *possible* if your end users are entirely within your organization, but it's outright impractical to distribute that many agents and collect information from them. It's *impossible* if your end users are external users, such as customers; such agents are often referred to as *spyware* no matter how beneficial they may seem. But an EUE-based agent would be the perfect monitor: It would "see" exactly what the end user sees and be able to monitor specific transactions and report back with real-time, real-world performance information.

Given the impossibility of using such an agent, however, we have to look at other techniques. In fact, because we can't directly and empirically monitor the EUE, we may have to use *several* techniques to monitor different aspects of the EUE, depending upon the exact elements of our application. Again, we want to make sure we're monitoring things that *map directly to the end user's actual experience;* we can't just fall back to monitoring CPU utilization and other technology-centric metrics.

Platform-Level APIs

One approach is to use application programming interfaces (APIs) for specific platforms. For example, being able to pull performance information from a Windows server or an Oracle database requires specific knowledge of how those platforms are built and how to pull performance information from them.

Data from Providers

In some cases, we might be able to draw performance information directly from our service providers. Some managed service providers (MSPs) can offer us detailed performance information not only at the technology component level but also at the EUE level, helping us to see the time it takes to complete a specific transaction, for example.

Distributed Monitoring Agents

One excellent tool for monitoring the EUE is, as I've already described, a network of distributed monitoring agents. Properly deployed, these can help us measure elements of the EUE from all around the globe—which is especially useful when we're dealing with a highly-distributed application.

Click-to-Click Monitoring

Click-to-click monitoring is really the ultimate in EUE monitoring because it operates at the EUE level. It involves measuring the exact amount of time that it takes to complete a sample transaction or even discrete steps within a transaction. Literally, "click to click" means measuring the time between an end user's specific actions—clicking "Next" in a shopping cart to clicking "Submit Order," for example.

The reason this is such a good EUE monitoring technique is that it encompasses the entire underlying application, including whatever components are included in it. If a shopping cart submission requires the coordination of twelve backend components, we'll capture all of that delay in the click-to-click time; by using the techniques I've described earlier—platform APIs, provider data, and monitoring agents—we can even break down the amount of time each element contributes to the final EUE.

For example, let's revisit a figure from the previous chapter. Figure 3.7 shows a distributed, multi-component application. At the EUE level, labeled "End-user Response View," we get the "rollup" of the time it takes to complete specific transactions. This is a Web application, so we see the TCP/IP response time, the HTTP connect time, and the time it takes to get a response from the URL. Those are the "wait times" that an end user would experience, and they're our top-level EUE metrics.

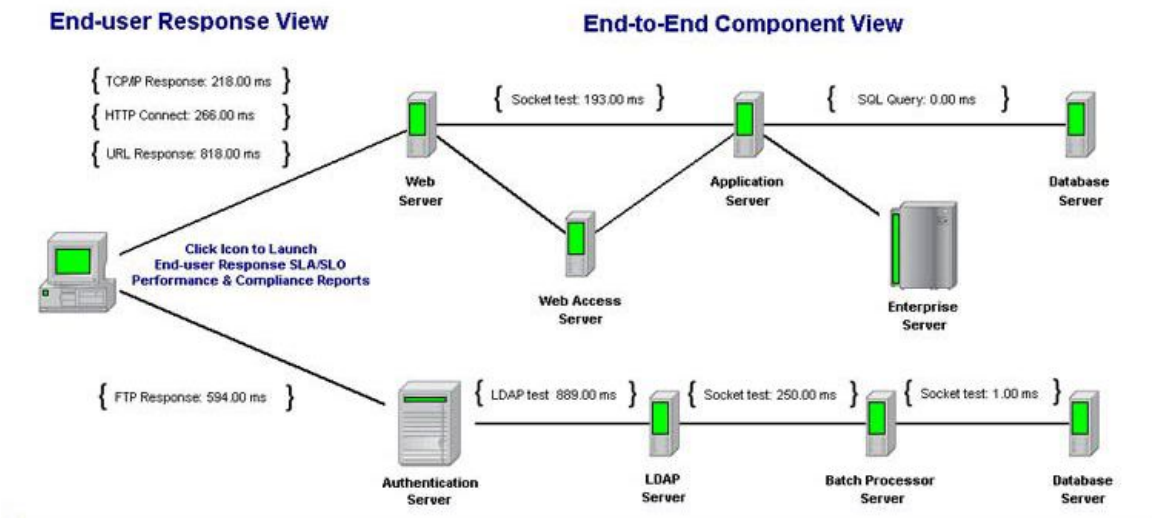


Figure 3.7: Click-to-click monitoring.

Literally, those times are what the user waits on between clicking their mouse button and seeing a final result on the screen, and being able to click their mouse button again to take their next action. Those are the times we manage to, and those are the times we should create SLAs around.

But when those EUE times don't look good, we immediately need to find out why, and that's where the "End-to-End Component View" comes into play (it'll be the subject of a much more detailed discussion in the next chapter). The idea here is to use platform-specific APIs, monitoring agents, and so forth to find out how that EUE time breaks down. Let's look at a detail of that portion, in Figure 3.8.

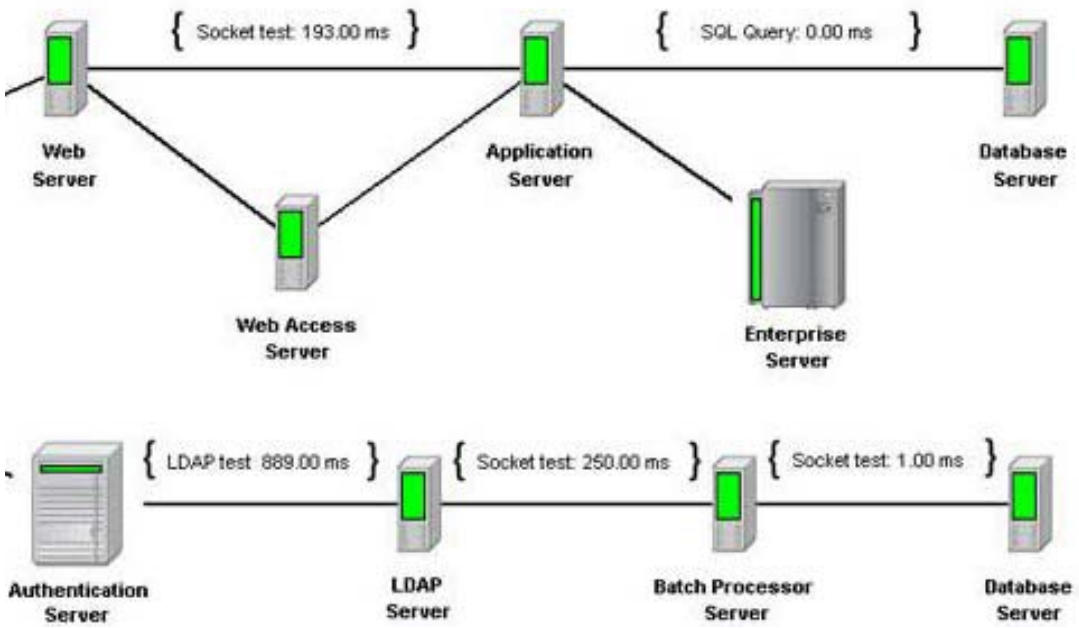


Figure 3.8: End-to-end component view.

We can see that Web server to Application Server communications, in the top left, used 193ms of time; the LDAP communications from our authentication server—on the lower left—took 889ms.

Here's the trick, though: *These times can be summed to provide us with the EUE time.* In other words, sometimes it might not be possible or practical to empirically measure the EUE. Although we cannot derive the EUE by monitoring technical elements like CPU utilization, we *can* derive the EUE by monitoring *response times* between specific elements of the application architecture. That's the secret of the EUE:

> The EUE is simply a measurement of response time. It does not take into consideration resource utilization, such as CPU, memory, or disk; it is entirely the sum of the response times of individual application elements. Although we cannot derive the EUE by monitoring individual elements' consumption of resources, we *can* derive the EUE by monitoring individual elements' *response times.*

That's the big paradigm shift in IT monitoring for APM. We're shifting from monitoring resources against predefined thresholds to monitoring actual response times. End users' sole metric for their perception of application performance is speed, which equates to response time, and so that's what we measure. "Monitoring the EUE" is really just a fancy way of saying "measure how long it takes."

Why We Often Don't Monitoring EUE Today

So if the EUE is such an amazing thing to monitor, why don't we do more of it? There are a number of reasons, one of which is sheer inertia: The IT industry hasn't ever really capitalized on EUE monitoring until recently, and the IT industry—for all that it is a force of change in many companies—doesn't like change. But there are also practical reasons.

Complexity

Measuring response times—which, again, is what the EUE is all about—can be difficult. For one, it almost always requires external instrumentation. You can't always ask a Windows server, for example, to measure its response time for looking up a piece of data because that measurement will be affected by the server's performance. If the server isn't performing well, the measurement won't be accurate. That means our traditional technical monitoring—which just draws on performance information directly from whatever we're monitoring—can't always deliver accurate response time information. We're then faced with the complexity of creating new monitoring schemes and implementing entirely-new tools. That can be complicated.

Lack of Tools

Speaking of tools, until the last half-decade or so, EUE wasn't on anyone's minds. That means our technology components—servers, databases, networks, and so on—weren't built to deliver response time information. It's only in the past 5 years or so that APM has really become a major thing, inviting innovative third parties to create a marketplace to fill the business need. In other words, until fairly recently, there's simply been a lack of tools that could monitor the EUE.

Even now, the industry is still figuring out what works. The very first vendors in the space adopted approaches that were suitable for applications of that time, and are often still applying those same approaches. Today's newer applications, however—and this has just been happening in the past couple of years—are built so differently and along such distributed models that the original EUE monitoring tools often can't keep up. That means we've been waiting on an entirely new set of tools and vendors who can specifically address EUE monitoring for the highly-distributed applications of the "hybrid IT" age.

For example, consider Figure 3.9. This is a portion of a fairly traditional APM console. You can see that it is indeed focused on response times, which contribute to the EUE, and it is generating alarms and alerts on response times that are exceeding predefined SLAs—in other words, it's helping to alert us to a problem in the EUE and helping us find the root cause. That's great.

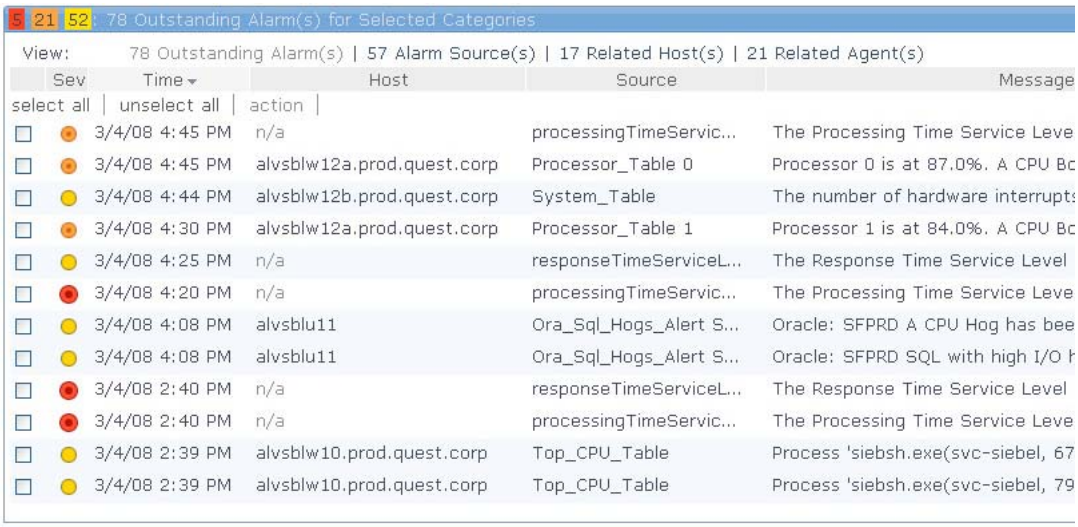


Figure 3.9: Traditional EUE monitoring.

The problem is that much of this intelligence is gleaned from a series of agents installed directly on servers, and even perhaps from network probe appliances connected to the network. After all, you have to collect the response times somehow, and the probe- and agent-based approaches are very common and effective—in traditional applications. Take a look at Figure 3.10, which shows the application stack from a more physical perspective, and see if you can spot the problem with this approach.

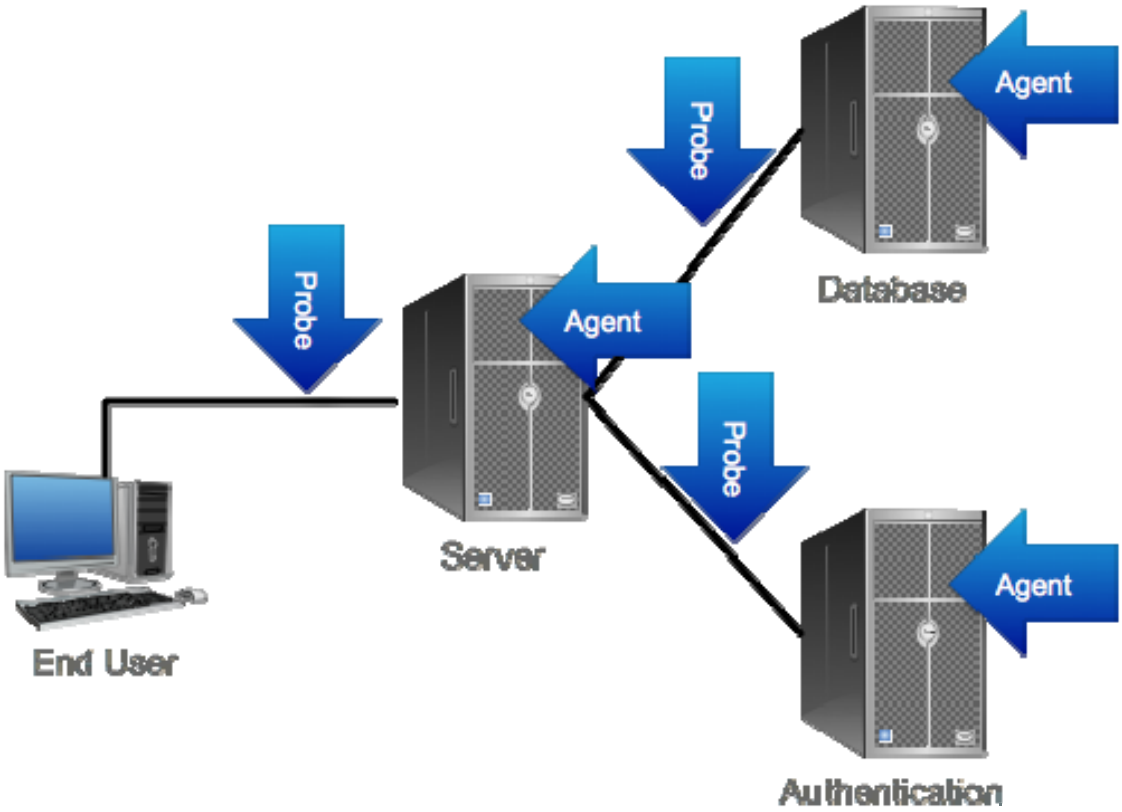


Figure 3.10: Physical application model.

The problem is that these tools *assume we own the entire application infrastructure.* In order to install agents and probes, we need to have all of these components in our own data centers, under our own control.

How do you propose to install monitoring agents on SalesForce.com's servers? How do you think you would monitor response times from the Windows Azure cloud's globally-distributed data centers? It's impractical, and that's why the approaches taken by traditional APM tools aren't always workable with highly-distributed, partially-outsourced "hybrid IT" applications. But there *is* a new breed of tools, both from innovative new vendors as well as from the traditional monitoring vendors, all of whom recognize the special challenges created by hybrid IT.

Cost

On the face of it, EUE monitoring for hybrid IT applications seems very expensive to build yourself. Simply deploying worldwide agents to monitor response times from a cloud computing provider would be prohibitive for most companies. Working with MSPs to gain instrumentation into their infrastructure, for the purposes of monitoring *your* use of that infrastructure, would have to be incredibly expensive for every single business to do—and that's why most businesses haven't done these things.

The trick is to leverage some economies of scale. Rather than each company building their own global monitoring network, companies need to look to application performance vendors who can *provide* that network, thus helping to subsidize the cost of that network across *many* monitoring customers, making that network affordable for all of them to use.

Component-Level Monitoring Can Be "Close Enough"

And do you know the number one reason so many companies have been content to ignore the EUE for so long? Because, in old-style applications, monitoring technical elements has always been close enough. "Look," you might say, "we know that users are happy so long as CPU stays under 70% or so, and so long as disk queues don't get longer than 5 or 6. So we monitor those things, and if they start to go awry, we know we have a problem that needs to be fixed."

And I can't argue with that logic—for *old school applications.* That is, for applications that involve one, two, or maybe even three tiers. Where every element of the application lives right within your own data center. Where you have full control of everything. Where you can *see* the end users, get phone calls from them, and talk to them about how the application is performing. Plus, most IT shops are *really good* at this kind of monitoring. They've learned how to do it over decades, and refined the process down to a true science. The problem is, applications don't look like that anymore.

Why We *Must* Monitor EUE Going Forward

Applications are simply growing too complex. Outsource a *single component* of your application—take a dependency on an SaaS provider, for example—and almost all your old-school monitoring techniques go out the window. Sure, you *could* just build applications that don't use outsources elements—but that's letting your technical limitations drive the business rather than letting what's right for the business drive your technical decisions. The fact is, to support modern business requirements, we're going to be seeing much more distributed, hybridized applications—and we need to figure out how to monitor them effectively. And the bottom line there is that monitoring the EUE is a *more* effective metric for *any* application—whether it's entirely in-house or mostly delivered from the cloud.

Vastly More Complex Environments

The old performance-and-threshold world really didn't offer fantastic performance management; it was simply "good enough" and it was readily achievable within the relatively simplistic application environments of the past. And that's the past.

We are very rightly building ever-more-complicated application environments today:

- Cloud computing offers the potential for near-infinite application scalability, without massive infrastructure investments. Even for internal line-of-business applications in geographically distributed companies, businesses are crazy not to at least investigate and consider putting a portion—if not all—of their applications into a cloud environment. It makes an incredible amount of sense in many cases.
- Anyone who's suffered through agonizingly-long implementations for applications like Customer Relationship Management (CRM) solutions can appreciate the ease, convenience, and lowered overhead of SaaS offerings. Very few businesses are *in* the business of supporting massive software installations, and SaaS can deliver the capabilities businesses need without the overhead and distractions.
- Like SaaS, managed services help businesses lower the cost and overhead—not to mention the distraction factor—of critical business services that *aren't* central to the business' competencies. Say you're a retailer. You obviously need email, but you're not *in the business* of providing email services. Why not let an MSP worry about it for you?

The arguments for this kind of piecemeal IT outsourcing are compelling, and thousands of businesses are benefitting from these new models. But the fact remains that we still need to build our *own* applications that depend upon these outsourced services. An online retailer might not want to deal with their email or CRM systems, but they *do* want to be responsible for their e-commerce systems—which unfortunately need to interface with the email system and the CRM system. The "old school" of monitoring would say, "well, we can't outsource email and CRM because they're critical to the e-commerce app, and the only way we can monitor them is if they're in-house." No more: We *have* to focus on EUE monitoring because it lets us outsource key pieces of our infrastructure while still maintaining visibility into what matters the most to our business.

Business- and Perception-Level Focus

Measuring technical elements like CPU, memory, and disk may have been 'good enough" to spot impending performance problems, but it did nothing for helping drive a business focus within the IT group. Let's face it, "The CPU is running at a steady 80% utilization" isn't quite as compelling, from a business perspective, as, "The Web site is running slowly and we're losing 10% of our shopping carts—that's money out the door!"

Although I do feel that purely-internal applications—ones completely under your control—*can* be effectively monitored even if you're ignoring the EUE, I don't think *any* application's top-level performance metric should be anything *except* the EUE. The speed of the application as perceived by the end user and by the business is the only thing that matters. Knowing your EUE can help make a ton of other business decisions much easier:

1. **You:** "Boss, the server's running 80% all the time. Can we add a processor?"

   Boss: "Um, no."

   versus:

   You: "Boss, the server is only able to maintain a 2-second response time for shopping cart checkouts, and we're losing about $12,000 in carts per day. Can we buy a new processor for $200?"

   Boss: "Um, yes. Immediately, please."

2. **You:** "Hi, Mr. Cloud Provider? Yeah, we have a guy in the office who feels that our Web site is taking a really long time to load. Can you do something about it?"

   Cloud Provider: (laughter)

   versus:

   You: "Hi, Mr. Cloud Provider? We're seeing 800ms response times from our Web application in your cloud. We agreed that 500ms was the limit, and we've narrowed the problem down to a 200ms extra delay in your database layer's response time. Fix it."

   Cloud Provider: "Wow—okay. We'll get on it."

3. **Boss:** "Sales for Asia are down. We're blaming the response times of the Web site. You need to get on it."

   You: "Sure, just let me update my resume, first."

   versus:

   You: "Boss, we're noticing 30% slower response times for our users based in Asia."

   Boss: "That's a million dollars in business a year! We'll get the provider on the phone and dig into this immediately."

The point is that moving to a business focus helps make a number of technology decisions easier because it helps put a business-colored spotlight on everything.

Too Much Is Out of Your Control

The last reason to move to EUE monitoring is that, quite simply, it's the only metric you can accurately and consistently obtain when much of the rest of the application infrastructure is completely out of your control. Yes, you might need *help* in obtaining accurate EUE numbers—especially with globally-distributed application components and end users—but the EUE is the only thing you can point to, with confidence, that tells you that "the business is doing okay." The less of your application infrastructure you can touch, the *more* the EUE is going to mean to you.

The Provider Perspective: You *Want* Your Customers Measuring the EUE

If you're an MSP, the preceding discussion should tell you what your customers are going to be looking for from you. In fact, there are some excellent reasons for you to begin providing performance metrics to your customers immediately.

The Provider Isn't 100% Responsible for Performance

When customers rely on your services as a part of their application, it's all too easy for them to point to *you* as the weak link when things aren't looking perfect. By providing your customers with accurate metrics—preferably in a way that can be consumed by the customers' own performance monitoring tools—you can help not only dispute claims that *your* performance is at fault but also *avoid* those claims entirely.

When customers can "see into" your network to some degree, you're giving them a number of benefits:

- You're proving that you don't have anything to hide.
- You're making yourself a partner in their business, not just a vendor.
- You're helping them eliminate you as a potential "weak link" because they can see the performance they're getting from you.

Everyone benefits. You're able to define more granular and accurate SLAs, and your customers are able to more easily verify that you're meeting them. When you and your customer have the same set of performance data in front of you, you're both able to have a stronger business relationship.

You Gain a Competitive Advantage

There's an enormous competitive advantage in being able to provide your customers with performance metrics. For one, doing so proves that you're a mature, competent, confident service provider. You're not a "fly by night" company who promises amazing performance for a too-good-to-be-true price, then doesn't deliver. By providing metrics—and challenging customers to evaluate your competition's ability to do so—you're making yourself accountable, and providing customers with a transparency that they'll appreciate.

You're also making yourself much more a *partner* in your customers' business. Look, the bottom line is that all service providers want to gain and *retain* customers; customers, for the most part, *want* to stick with their providers—switching is an enormous hassle with little value-add. If you can help to integrate your infrastructure with your customers', you'll help them manage their businesses and IT investment more easily and accurately. They're a lot less likely to *want* to switch—you'll have gone a long way toward making a customer for life.

Coming Up Next…

We're hopefully agreed that the EUE is the way to go for managing application performance at the top level. If you see an EUE problem, though, you're going to have to be able to dig deeper to find the root cause. That's what the next chapter will focus on: Monitoring at the component level. This isn't about ensuring good application performance; it's about fixing performance problems that you've noticed in your EUE measurements. I'll look at the traditional monitoring stack and some of the challenges that come into play with today's multi-discipline applications. I'll also look at newer monitoring techniques, and provide insight for service providers who want to offer their customers deeper insight into their service offerings.

Download Additional Books from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this book to be informative, we encourage you to download more of our industry-leading technology books and video guides at Realtime Nexus. Please visit http://nexus.realtimepublishers.com.