

# Find and fix performance bottlenecks

The trouble with performance bottlenecks is that they can be tough to identify. Is it the CPU? The network? A clumsy bit of code? Often, the most obvious culprit is actually downstream of something larger and more mystifying. And when performance riddles remain unsolved, IT management may find itself faced with a Hobson's choice between admitting ignorance and making up excuses.

Fortunately, as with medical diagnoses or detective work, experience helps. Drawing on our years of sleuthing and experimentation, we've collected 15 of the most likely ailments -- and suggested remedies -- to help your ICT operation track down and crack performance problems.

Some of these bottlenecks are more obvious than others. But by identifying common speed killers across ICT disciplines, we hope to jumpstart your quest to create the highest performing infrastructure your resources will allow.

No. 1: It's probably not the servers

Server upgrades used to make all the difference, which is why the old saw "When all else fails, throw more hardware at it" persists today. That's still true in some cases. But how much of IT is really that compute-intensive? Generally, you can save a lot of time and money by turning your hairy eyeball away from server hardware. The lower end of the server spectrum has more than enough horsepower to handle everyday tasks.

Here's one concrete example. On a network of over 125 users, an elderly Windows domain controller appeared to be ripe for replacement. This server originally ran Windows 2000 Server and was upgraded to Windows Server 2003 some time ago, but the hardware remained unchanged. This HP ML330 with a 1GHz CPU and 128MB of RAM was functioning as an Active Directory domain controller carrying all the AD FSMO roles, running DHCP and DNS services as well as running IAS (Internet Authentication Services).

Molasses, right? In fact, it actually did the job just fine. Its replacement was an HP DL360 G4 with a 3GHz CPU, 1GB of RAM, and mirrored 72GB SCSI drives. Carrying all those services, it runs hardly any load at all -- and the performance difference is unnoticeable.

It's easy to identify applications that will eat all your CPU and memory, but they tend to be pretty specialised. For almost everything else, the humble commodity box will do the trick.

No. 2: Speed up those queries

You can create the niftiest application in the world, but if access to back-end database servers creates a bottleneck, your end users or customers won't be happy. So fine-tune those database queries and maximise performance.

Three basic measures can help you improve query performance. First, most database products include tools (such as DB2 UDB for iSeries' Visual Explain) that can dissect your query during development, providing feedback on syntax and the approximate timing of the various sections of the SQL statements. Using this information, locate the lengthiest portions of the query and break those down further to see how you might shorten the execution time.

Some database products also include performance advice tools, like Oracle's Automatic Database Diagnostic Monitor, that provide recommendations (such as suggesting you create a new index) to speed up queries.

Next, turn on database monitoring tools on a staging server. You might use a third-party monitoring product, such as Fidelia's NetVigil, if your database lacks monitoring support. With the monitors enabled, generate traffic against the

database server using load-testing scripts. Examine the data gathered to see how your queries performed while under load; this information may lead you to some further query tweaking.

If you have enough server resources to mimic your mixed workload production environment fairly closely, you can execute a third round of query tuning using a load testing tool, such as OpenSTA, plus database monitoring to see how your queries perform alongside other applications that hit the database.

As database conditions change -- with volume growth, record deletions, and so on -- keep testing and tuning. It's often well worth the effort.

No. 3: What cost, virus protection?

Virus protection on critical servers is a basic requirement, especially for Windows servers. The impact can be painful, however. Some virus scanners are more obtrusive than others and can reduce server performance significantly.

Try running performance tests with and without your virus scanner running to determine the impact. If you see a marked improvement without the scanner, it's time to look for another vendor. Also check specific features. Disable real-time scans, and quite often you'll bring performance up.

No. 4: Maximising the middle tier

No matter how well written your business logic, when you deploy it to the middle tier, you will need to tune the application server runtime environment to maximise performance.

Like a vintage stereo with oodles of knobs for tweaking sound quality, application servers from vendors such as BEA, IBM and Oracle supply a dizzying set of controls. The trick is to turn the knobs just the right way, depending on the attributes of your application.

For example, if your application is servlet-heavy, you'll want to enable servlet caching. Likewise, if your application uses many SQL statements to support a large user base, you'll want to enable prepared statement caching and set the maximum size of the cache so it's large enough to support the intended workload.

One of the major areas where performance tuning can really help is with the database connection pool. Set the minimum or maximum connections too low and you're certain to create a bottleneck. Set them too high and you'll likely see a slowdown resulting from the added overhead needed to maintain the larger connection pool.

If you know the intended workload, tune the application server runtime by turning on performance monitoring tools such as IBM's Tivoli Performance Viewer for WebSphere on a staging application server. Generate the amount of workload that you expect by using a load-generation tool, then save the monitoring results and play them back to analyse which knobs need adjusting.

When in production, it's a good idea to turn on low-overhead, passive monitoring to keep tabs on the runtime. If your workload changes over time, you'll want to execute a fresh performance review.

No. 5: Optimise network connectivity

Most mid-level enterprise servers now have dual gigabit NICs -- but most of them don't use the second pipe. Moreover, gigabit switch prices have dropped through the floor. With a 120MBps link to your fileserver, a number of 100-megabit clients can achieve wire-rate file access simultaneously.

Even without gigabit switching, NIC bonding should be a staple. At its simplest, bonding two NICs gives you redundancy, but add transmit load-balancing, and you can effectively double the outbound bandwidth. Using switch-assisted teaming will provide the same effect on inbound traffic. Almost every major server vendor offers NIC teaming drivers -- and there are third-party utilities, too. It's a big, cheap bandwidth boost.

#### No. 6: Winding up your Web servers

Is there really that much you can do to tune a Web server and maximise performance? In fact, there is -- mainly by adjusting a handful of critical settings to match the production traffic you expect.

For Web servers already in production, begin by collecting real-time Web server statistics (most major Web servers have that functionality built in). Then move to staging in order to determine which parameters, if any, need adjustment.

Activate the Web server's performance-monitoring tools on the staging server. Execute a load test and inspect relevant parameters, such as response time, bytes sent and received, and the number of requests and responses.

Key parameters you'll want to tune depending on the volume of traffic include caching, threading, and connection settings.

Enable caching for frequently used content; some Web servers allow you to cache files dynamically based on usage, whereas others demand that you specify what will be cached. Make certain your maximum cache size is sufficient for the traffic you expect. And if your Web server supports cache acceleration, enable that, too.

For the threading and connection settings, set the minimums and maximums in accord with expected workload. For connections, you'll also need to define the maximum number of requests per connection and the connection time-out setting. Don't set either of these values too small or too large, or slowdowns may result.

#### No. 7: The woe of the WAN

Think you need to reclaim WAN bandwidth? You can easily spend a bundle on traffic-shaping appliances or caching engines in an attempt to rein in WAN bandwidth utilisation. But what if it's not the pipe?

First things first: before you buy anything, get a solid idea of what traffic is crossing the WAN. Network analysis tools such as Ethereal, ntop, Network Instrument's Observer or WildPacket's EtherPeek NX can give you a fresh look at what's really on the wire.

You may find that replication times for your Active Directory are set far too low and simply configuring longer replication intervals can buy you breathing room during the workday. Are some users in remote locations mapping shares to the wrong servers and pulling large files across the WAN without realising it? Are the vestiges of a long-disabled IPX network still floating around? Some WAN problems boil down to application mis-configuration, where traffic is directed across the WAN when it should have stayed local. Regular reports on WAN traffic patterns will save money and headaches.

#### No. 8: Let's play nice

All too often, applications, Web services and Web sites from multiple departments across the enterprise compete for server resources. Although each of these components may be well-tuned in its own right, an application from another department that is also using the same production clusters may have a poorly tuned query or some other issue, which in turn affects your users or customers.

In the near term, all you can do is work with your system administrators and the department that is having the performance problem to obtain resolution for your users or customers. Longer term, create a community across all of the departments that use the production clusters where your objects are deployed.

Work across teams to ensure that there is adequate funding for a staging environment that is truly representative of the mixed workload production environment. Ultimately, you'll want to develop a series of benchmarks that can be used to validate mixed workload performance in the staging environment.

No. 9: Caching, shaping, limiting, oh my!

If your WAN is truly undersized -- and you can't afford a long-haul frame-relay network -- traffic shaping and caching can help unclog the pipe.

Traffic-shaping configurations are more art than science. Prioritising apps is often more political than technical but may have tremendous effects on perceived network performance.

Caching is a different beast altogether. It requires less work than traffic shaping, but the impact will likely be smaller. Caching engines store and serve up local copies of commonly accessed data to reduce WAN traffic. The downside is that dynamic content can't truly be cached, so e-mail won't enjoy the same performance bump.

No. 10: Predictive patching

You arrive at work on Monday only to learn that a bunch of desktops are hung or that the performance of a critical application has slowed to a crawl. After investigating, you determine that a patch that was applied over the weekend is the cause.

That's why you need tools that support patch rollbacks. Even better, include patch testing as part of your patch-management strategy. First, you must take regular inventory of the applications and technologies in play on desktops and servers. Most systems-management tools, such as Microsoft's SMS, have the capability to take inventory for you automatically.

Next, replicate the applications and technologies into a staging environment. If your operating system and infrastructure software do not include patch testing tools, get a third-party tool such as FLEXnet AdminStudio or Wise Package Studio.

Alternatively, you can write some scripts to functionally exercise the platform or technology with the latest patches in play. You will need to repeat this scenario (and adjust scripts) as new patches arrive and as software changes are made.

The final, crucial step is to execute a round of stress tests with the new patches in play. Quite often, patches prove problematic only under heavy loads.

No. 11: Keeping your cool

It might seem obvious, but environmental problems in server rooms and wiring closets can have a deleterious effect on overall network performance. Hot servers aren't happy servers, and the same goes for network hardware. Make sure your datacentre isn't making the mercury sing, and install environmental sensors to keep it that way.

On a related note, it's always a good idea to leverage the server-management utilities that may have come with your servers. Dell's OpenManage and HP's Insight Manager provide very pertinent information regarding the health and well-being of your server infrastructure, including whether or not it has a fever.

#### No. 12: Reining in mirroring and replication

Slowdowns often plague enterprises that use mirroring or replication for high availability or disaster recovery between locations. If you have many locations or many database tables -- or a lot of transactions or journaling that needs to stay in sync between multiple locations -- watch out, because the performance loss can be dramatic.

If possible, run your mirroring and replication activity across separate WAN "pipes" to keep it isolated from production traffic. Your network design team can assist with a viable topology. At the same time, go over the configuration (star topology, for example) you intend to use to support mirroring and replication. Vendor representatives and the network design team can provide useful input on constructing a configuration that will prevent network saturation.

Configuration aside, mirroring and replication products -- such as XOssoft's WANSyncHA Oracle or High Availability Linux Project's Heartbeat -- usually provide options to control timing and traffic flow between sites. Some products enable you to schedule syncing activity during off hours, whereas others let you activate syncing tasks only if a particular threshold is reached. Use those controls; that's what they're there for.

#### No. 13: 'My cable modem is faster'

A shared T1 was once immeasurably fast to most folks. Now, it's less than half the bandwidth most of us have at home.

Are users complaining that their broadband hookup at home is faster? Well, guess what? A cable connection may be just what you need at work. Buying an asynchronous business-class cable circuit to coexist with a T1 can ease the Internet connection burden substantially. Source-routing at the network core directs regular Internet traffic from within the network to the cable connection while pushing higher-priority traffic through the T1 circuit, giving you the best of both worlds.

Traffic-shaping at the Internet circuit is definitely worthwhile, and many firewalls are integrating this capability, removing the need for more hardware. Blocking or limiting applications deemed "frivolous" at the firewall can help, and deploying a proxy server -- even a transparent proxy -- can also be beneficial, albeit a bit of a chore. Implementation of an AUP enforcement product such as Websense can also control unchecked Internet usage.

If your corporate Web site runs within your network, moving it to a co-location or managed-hosting facility might be a viable option to increase throughput of your Internet circuit. Co-location costs are dropping, and monthly rates on servers suitable to run most sites are quite reasonable. Removing that draw from your link may produce noticeable performance improvements. Of course, gathering metrics on your Web site's used bandwidth is a must before you make any decisions.

#### No. 14: Estimating future speed

One of the trickier tasks is projecting infrastructure requirements in the face of constantly evolving business demands. Most people start with capacity planning tools, such as TeamQuest's View and Modelling products. No tool alone, however, can accurately predict the nature of tomorrow's workload.

A little sleuthing is needed. Put on that detective hat in a sandbox, staging or fail-over environment and do some proof-of-concept work.

Create a virtualised, grid-type environment and replicate a representative sample of workload. From this exercise you'll obtain and use numbers to extrapolate the overall change in infrastructure performance if the topology were to undergo a wholesale change. Execute baseline performance tests that detail today's workload. Then execute load tests that mimic the addition of the expected number of transactions in six months or an increase in the number of users over time.

Even if testing resources are not as plentiful as those found in production, you can still run tests that mimic the type of infrastructure changes required to meet business demands and get reasonably good numbers. With this information you should be able to project when to add resources to keep performance on track.

#### No. 15: Profiling to maximise performance

Be proactive in preventing performance bottlenecks by including a mandatory profiling step in the latter part of development, specifically during the software lifecycle. Profiling your applications and Web services will uncover whether your code is efficient.

A profiling tool analyses and provides reporting on a number of the attributes in your code during its run time. Among other things, it can tell you which sections of code are taking the longest, how effectively you are using processor resources, and how your code is using memory, including reporting that illustrates where you may have a memory leak in your code.

If you use any of the major IDEs, such as Visual Studio or WebSphere Studio, profiling tools are included. If your development tools do not currently include profiling facilities, there are a number of third-party offerings, including Compuware's DevPartner, YourKit, InfraRED, and J-Sprint.

During development, you'll likely want to execute profiling locally to gain a general understanding of how your code is performing. Although profiling tools may differ slightly in their approach, profiling locally entails starting an agent or collection process on your machine. Then, exercise the application as your users would, stop the agent or collection process, and analyse the results.

If your code will be distributed across multiple platforms, consider profiling in a distributed fashion during the staging process.

To accomplish this, you'll need to start agents or collection processes on each of the machines where your code will run. After exercising the code, most profiling tools will combine data from all of the agents or collection processes identified in the profiling session. Analysing the data from all the distributed agents or collection processes will yield an end-to-end illustration of your code's efficiency. Use that map as a basis for your tweaking, and better performance will result.