Craig S. Mullins

Return to Home Page

September 1998
Computing : News&Review

Defining Database Performance

By Craig S. Mullins

Most organizations monitor and tune the performance of their IT infrastructure. This infrastructure encompasses servers, networks, applications, desktops, and databases. But the performance management steps taken are almost always reactive. A user calls with a response time problem. A table space runs out of space to expand. The batch window extends into the day. Someone submitted that "query from hell" that just won't stop running. Those of you in the trenches can relate — you've been there; done that.

Handling performance problems is truly an enterprise-wide endeavor. However, the task of enterprise performance management frequently becomes the job of the DBA group. Anyone who has worked as a DBA for any length of time knows that the DBMS is usually "guilty until proven innocent." Every performance problem gets blamed on the database regardless of its true source cause. DBAs need to be able research and decipher the true cause of all performance degradation, if only to prove that it is not caused by a database problem. As such, DBAs must be able to understand at least the basics of the entire IT infrastructure, but also need to have many friends who are experts in other related fields (such as networking, operating systems, communication protocols, etc.). Possessing a sound understanding of the IT infrastructure enables DBAs to respond effectively when performance problems arise. Event-driven tools exist on the market that can make performance management easier by automatically invoking pre-defined actions when specific alerts are triggered. For example, an alert can be set to proactively reorganize a database when it reaches its storage capacity. And other tools exist that can ease the burden of performance management and analysis. But many of the supposedly proactive steps taken against completed applications in production are truly mostly reactive. Let's face it, DBAs are often too busy taking care of the day-to-day tactical database administration tasks to proactively monitor and tune their systems to the degree they wish they could.

All of this discussion is useful, but it begs the question: just what do we mean by the term database performance? You need a firm definition of database performance before you can learn ways to plan for efficiency. Think, for a moment, of database performance using the familiar concepts of supply and demand. Users demand information from the database. The DBMS supplies information to those requesting it. The rate at which the DBMS supplies the demand for information can be termed "database performance."

Five factors influence database performance: workload, throughput, resources, optimization, and contention.

The *workload* that is requested of the DBMS defines the demand. It is a combination of online transactions, batch jobs, ad hoc queries, data warehousing analysis, and system commands directed through the system at any given time. Workload can fluctuate drastically from day to day, hour to hour, and even minute to minute. Sometimes workload can be predicted (such as heavy month-end processing of payroll, or very light access after 5:30 p.m., when most users have left for the day), but at other times it is unpredictable. The overall workload has a major impact on database performance.

Throughput defines the overall capability of the computer to process data. It is a composite of I/O speed, CPU speed, parallel capabilities of the machine, and the efficiency of the operating system and system software. The hardware and software tools at the disposal of the system are known as the *resources* of the system. Examples: database kernel, disk space, cache controllers, and microcode.

The fourth defining element of database performance is *optimization*. All types of systems can be optimized, but relational databases are unique in that query optimization is primarily accomplished internal to the DBMS. However, there are many other factors that need to be optimized (SQL formulation, database parameters, etc.) to enable the database optimizer to create the most efficient access paths.

When the demand (workload) for a particular resource is high, contention can result. *Contention* is the condition in which two or more components of the workload are attempting to use a single resource in a conflicting way (for example, dual updates to the same piece of data). As contention increases, throughput decreases.

Therefore, database performance can be defined as the optimization of resource use to increase throughput and minimize contention, enabling the largest possible workload to be processed. Of course, I do not advocate managing database performance in a vacuum. In addition, applications regularly communicate with other subsystems and components of the IT infrastructure. Each of these must also be factored into the overall performance planning of your organization. But it is wise to place limits on the actual responsibility for tuning outside the scope of this definition. If it is not defined above, it probably requires expertise outside the scope of database administration. Therefore, performance management tasks not covered by the above description should be handled by someone other than the DBA — or at a minimum shared between the DBA and other technicians.

A Basic Database Performance Roadmap

A basic plan needs to be forged to ensure that database performance management and analysis is accomplished at your site. Following the old 80-20 rule, the first step should be to identify the most troublesome areas, but this is not as easy as it might seem! As mentioned in the May edition of this column ("The Most Important Thing is Performance"), inefficient SQL is the single most prevalent cause of poor application performance. Finding the SQL statements that are the most expensive in a large shop is an extremely difficult thing to do. Resource hogging SQL statements might be hiding in one of hundreds or even thousands of programs. Interactive users who produce dynamic, ad hoc SQL statements might physically reside anywhere, and any single one person who is generating ad hoc queries can severely effect overall production performance.

A good approach is to use an SQL monitor that can identify all SQL running anywhere in your environment. Typically, these tools can rank SQL statements based on the amount of resources being consumed and track the statement back to who issued it and from what program it was called. Once you have identified the top resource consuming statements, you can concentrate your tuning efforts on the most costly statements.

However, it is not always obvious how to tune poorly coded SQL statements to make them better. A plan analysis tool can be used to identify how the SQL is currently being satisfied as well as to provide a set of expert tuning recommendations on how to fix each inefficient SQL statement. You should also have additional tools such as the agent-based performance management solution mentioned earlier as well as tools to report on DBMS-specific and automation tools to analyze database objects and system resources.

Summary
Plan on combining a good definition of database performance with a detailed performance plan specific to your shop and suite of proactive DBA tools.

From *Computing News and Review*, September 1998.