Sizing Considerations for Large Buffer Caches

The database buffer cache is one of the most important architectural features of an RDBMS. Data that resides in the buffers can be simultaneously accessed by many users without performing disk I/O. Making the buffer cache larger will generally reduce the amount of disk activity needed to retrieve information from the RDBMS. One of the DBA's most important tasks is to determine the best buffer cache size for each RDBMS system. There are two "rules of thumb" for adjusting buffer cache size:

- Increase the cache size until 85 to 95 percent of applications' requests for RDBMS data are satisfied from it.
- Do not allow an instance to use so much memory that it affects the performance of everything on the system.

The first rule of thumb is easy to follow. The *ORACLE RDBMS Performance Tuning Guide* describes an excellent method for estimating the optimal number of buffers for an instance. Alternatively, users can simply increase the db_block_buffers parameter value until it achieves the desired hit ratio while users' applications are running. Keep in mind that increasing the number of buffers may cause the operating system to page so much that average user response time increases to several minutes or hours.

The second rule of thumb requires some knowlege of the operating system. As RDBMS memory usage increases, paging and swapping rates also increase, until the operating system is so busy moving pages of memory that it cannot complete any useful work.

Figure 1 shows an idealization of how the size of the buffer cache can affect the maximum transaction rate an instance can sustain. In general, maximum throughput first increases, then decreases, as the buffer cache size is increased. Throughput increases as the cache hit ratio improves; however, throughput decreases as the RDBMS uses too much memory and the system paging rate goes up. These two trends balance at a point of optimal performance that is different for every system.

Figure 2 presents an example of realistic system behavior. Here both curves "level out" in places where the cache becomes large enough to hold some specific heavily referenced database objects. This curve could only look like Figure 1 if the database consisted of only one large table, or the database were composed of a very large number of small tables. Note that these graphs assume that the database size is much larger than the memory available to the instance, ensuring that the database can never be fully cached.

When an instance uses too much memory, paging rates will increase, affecting system performance. Rather than discussing paging in general terms, this article will consider IBM's MVS operating system.

Paging in MVS

Two types of memory are available on modern IBM mainframes: central storage and expanded storage. Together, they are referred to as processor storage. Central storage can be addressed directly by the processors, and

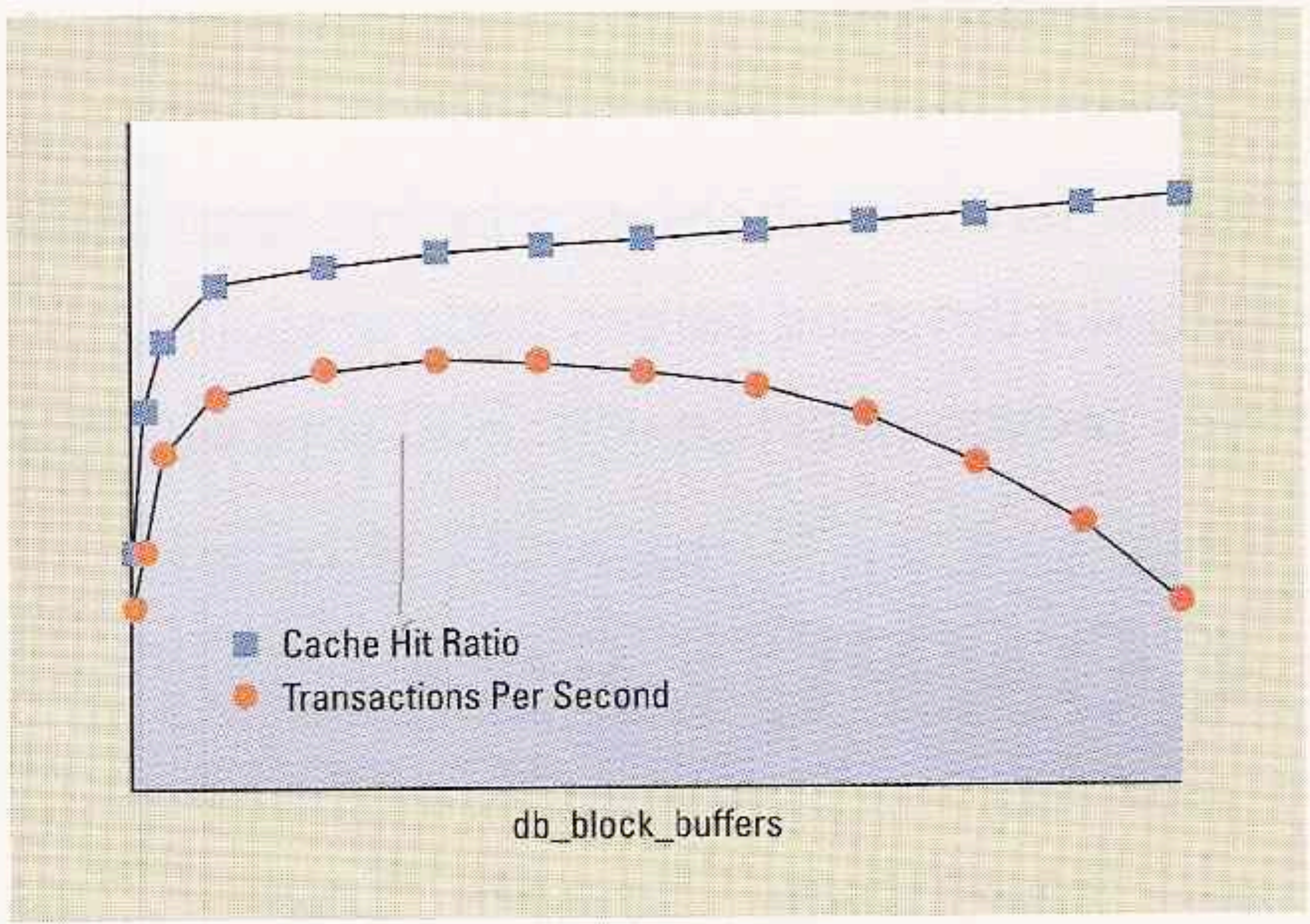

FIGURE 1 EFFECTS OF LOGGING ON TRANSACTION THROUGHPUT

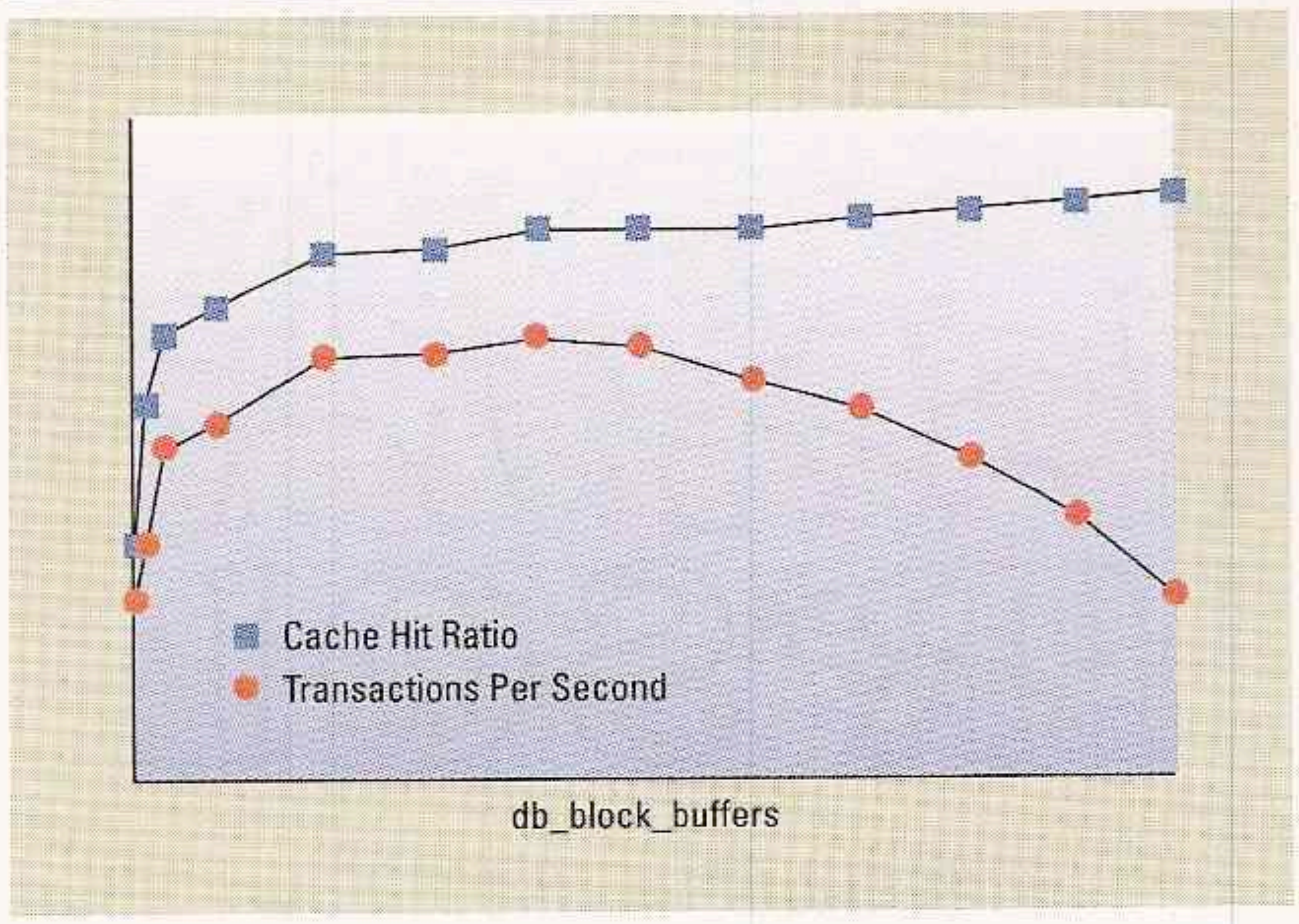

FIGURE 2 ARCHIVING WITH LOG FILES ON DIFFERENT DISKS

is what is normally referred to as memory. Expanded storage is not directly addressable, but a special instruction is supported by the hardware to directly transfer data between central and expanded storage. Since expanded storage is less expensive than central storage, it is used by operating system components and some applications systems to avoid performing I/O operations.

Generally, the biggest user of expanded storage is the paging subsystem, which can synchronously resolve a page fault in less than 100 microseconds by moving pages between central and expanded storage. Current IBM mainframes are typically configured with as much (or more) expanded storage as central storage. With this memory architecture, modern IBM mainframes can operate efficiently at paging rates of over 500 pages per second for each central processing unit in the system.

If the active address spaces in an MVS system reference more memory than can be contained in central and expanded storage, MVS "migrates" pages from expanded storage to disk. A high page migration rate affects performance in the same way that a high paging rate affects a system that does not have expanded storage.

Taking Advantage of Paging

Since the MVS paging subsystem uses expanded storage so efficiently, users can take advantage of expanded storage by allowing the buffer cache to use more than the available amount of central storage. For example, consider a client-server application using a dedicated MVS server system to maintain a 500-gigabyte database for many hundreds of clients. If the IBM mainframe is configured with 64 megabytes of central storage and 64 megabytes of expanded storage, then an 80- to 100-megabyte buffer cache might be allocated without causing any paging problems. Because MVS can use expanded storage almost as efficiently as central storage, a high paging rate can be tolerated.

Page migration, on the other hand, can seriously impact performance. If the client users in the above example allocated a total of 100 megabytes of Process Global Area (PGA) memory on the server while 100 megabytes of memory were allocated to the buffer cache, then the page migration rate would almost certainly be high enough to reduce overall application performance.

Benchmark Measurements

Figure 3 presents measurements taken during a series of benchmark tests. This synthetic benchmark shows a relationship between transaction rate and cache hit ratio on the MVS system. The benchmark database consisted of two tables: a randomly updated, 1.1-gigabyte table with a 192-megabyte index, and one "sequential" table without any indexes. Each transaction updated one row in the large table, and inserted one row to the end of the "sequential" table. Therefore, the first database object to use much cache space was the index for the large table, followed by the most recently referenced blocks in the large table. Only one or two blocks of the "sequential" table were cached, since the transactions only referenced rows at the very end of the table. During each of the tests, 15 batch jobs ("background processes") were run concurrently to generate the transactions. All tests were run stand-alone under MVS/ESA version 3.1.1 on an IBM 3090-180J processor with 128 megabytes of central storage and 256 megabytes of expanded storage.

Notice that the maximum transaction rate in Figure 3 starts to drop off when db_block_buffers is increased to 100,000. Database blocks are four kilobytes in size on MVS, so 100,000 buffers is about 400 megabytes of memory. Since this system is configured with 384 megabytes of processor storage, page migration begins when the memory allocated by all address spaces exceeds 384 megabytes. Maximum transaction rate drops off on the right side of the figure because the page migration rate was about 100 pages per second during the 100,000-buffer test, but was 0 during all other tests.

The transaction rate was highest when 50,000 or 75,000 buffers were allocated by the instance. Allocating 50,000 buffers implies a 200-megabyte buffer cache, which is large enough to hold the index for the randomly updated table. With 50,000 buffers allocated, the cache hit ratio was 90 percent and MVS was moving 140 pages per second to expanded storage. Another 25,000 buffers increased the cache hit ratio to 93 percent and increased the paging rate to 190 per second, without increasing the maximum transaction rate significantly. This data confirms that a 90 percent cache hit ratio is high enough to achieve very good performance.

Most sites will almost always achieve a 90 percent cache hit ratio without allocating 50,000 buffers. This

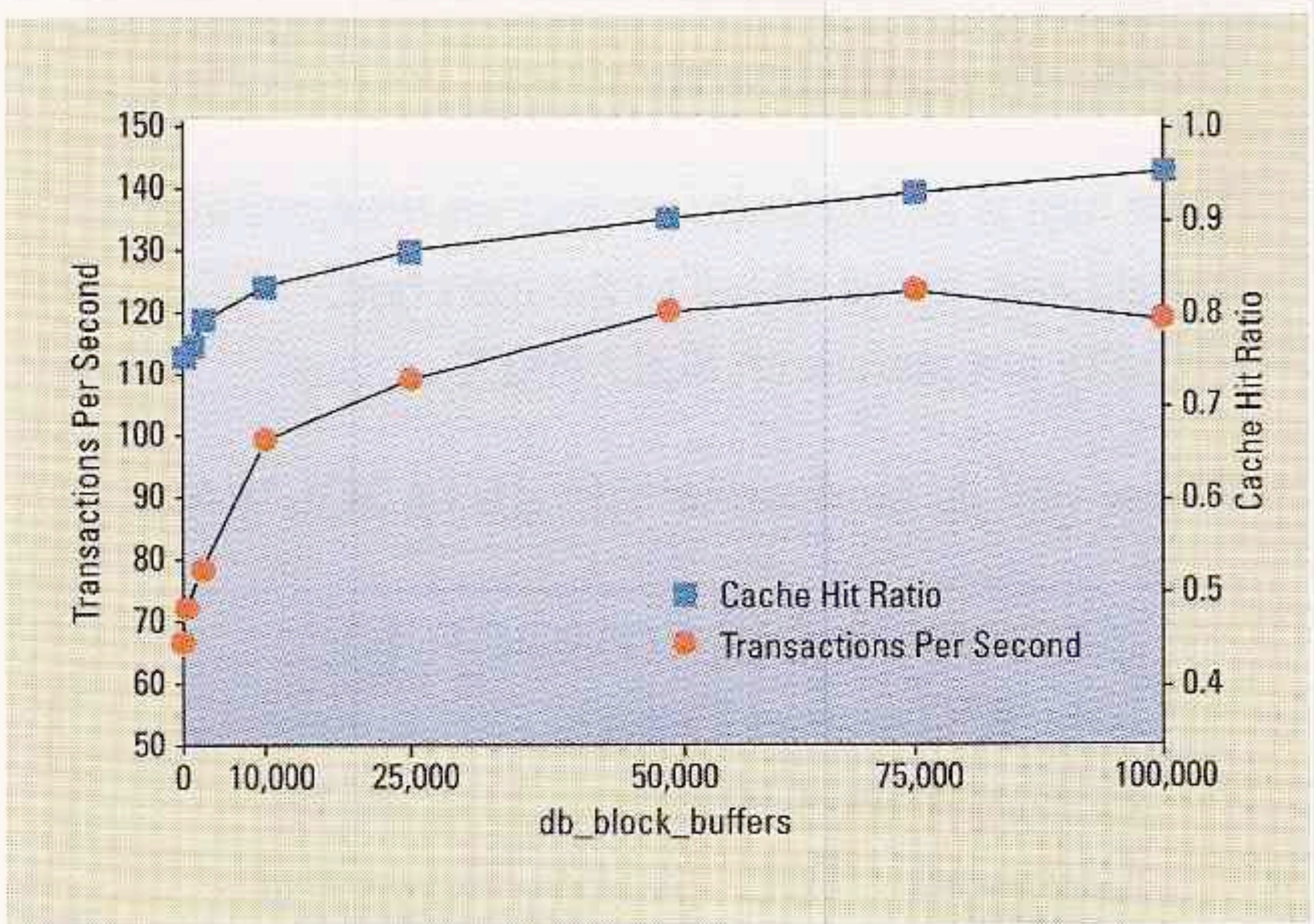

FIGURE 3 THE PROCESS CHANGES WHEN ONE LOG FILE FILLS

benchmark was designed to require a large buffer pool and to clearly show the effects of MVS paging. In reality, most database designs require far less memory. For example, a scale-100 TPC benchmark database on MVS occupies about 1.6 gigabytes of disk space, but only requires 1,000 buffers (a 4 Mb buffer pool) to access with a 90 percent hit ratio. In general, a site will probably need somewhere between 500 and 10,000 buffers on MVS to achieve a satisfactory cache hit ratio.

Not Enough Memory?

These examples and test results assume that enough memory is available to achieve a good cache hit ratio. On some mainframe systems, there may not be enough memory available to allocate an optimal buffer cache. To work effectively on a system with memory constraints, develop a good relationship with the systems staff responsible for system performance. With the systems programmer's help, monitor and tune the ORACLE RDBMS and the MVS paging subsystem carefully, recognizing that ORACLE performance tuning can degrade MVS performance tuning, and vice versa. Users may get the best response time from ORACLE with a cache hit ratio of 80 percent, if very little memory is available.

Limited mainframe systems may ultimately need more memory. Ten years ago, MVS systems were limited to 16 megabytes of memory, necessitating high paging rates. Today, systems are often configured with 8 to 15 megabytes of processor storage for each MIPS of processing power. Many sites still run MVS while migrating hundreds of pages a second, believing they are out of MVS capacity because the processor is 99 percent utilized. In fact, these sites often have processor capacity that cannot be used because they are expending so much CPU in the paging subsystem. Additional central and/or expanded storage can dramatically improve the performance of the entire MVS system and allow ORACLE to run efficiently.

Conclusion

Buffer cache sizing is one of the most important tasks for an ORACLE DBA. Every RDBMS instance has different buffering requirements, requiring careful monitoring of both RDBMS and operating system performance while making adjustments. Remember that new applications often change the buffering requirements for an instance. Also, cultivate a good relationship with the programming staff responsible for the overall performance of the operating system environment. Both DBA and systems programming viewpoints are necessary to achieve the best performance from relational applications.

Tom Childers is a senior technical staff member in Oracle's IBM Mainframe Performance Analysis group.