Database Monitoring and Tuning

This chapter describes the data base administrator’s tasks in the area of monitoring and tuning. It covers the following topics:

- Monitoring Resource Use
- Reporting on Resource Use
- Monitoring Database Controls
- Performance Management, Statistics, and Tuning
- Adabas Session Statistics
- Command Logging

Monitoring Resource Use

The DBA is responsible for monitoring the database environment on a continuing basis to ensure that an efficient level of service is provided while maintaining database integrity.

The DBA should implement a set of procedures designed to foresee degradation before the event and to adjust the operation or design of the database in an orderly and controlled way. This set of procedures includes

- identifying potential sources of degradation;
- establishing tools for monitoring database performance; and
- controlling the implementation of adjustments.

Reporting on Resource Use

The DBA should report regularly on database use and performance to both data processing and user management. The reports should be factual, but should also include recommendations for tuning the database environment. It should be remembered that tuning, while benefitting the organization as a whole, may adversely affect the service received by one or more users. Any decision on tuning should, therefore, be made by all affected users.

Monitoring Database Controls

The DBA should establish appropriate controls and monitor them to ensure the integrity of the database.

Computer-generated control totals can be checked and cross-footed between computer processing runs or generated reports. Batch responses (or inquiries) may include such information as the exact run time, search parameters, time of last update of data, and the primary parameter controls. This increases the confidence level and helps to ensure the integrity of the database.

The problem of control totals takes different forms at different installations. Although hard and fast rules are not possible in this area, some general guidelines can be given.

The DBA needs to ensure that proper consideration is given to the following areas in the design of each application system that will use the database:

- What controls can be checked on every batch update run? For example, record counts, additions, deletions, updates.
- What controls require a full file pass to check them? For example, value field hash totals.
- What input transactions, Adabas logs, etc., should be retained in order to be able to recover when control totals are found to be wrong at the end of a given period?
- Are localized control totals (that is, by branch, product group) of any use in identifying the areas affected by a file control total error?

Performance Management, Statistics, and Tuning

The following table illustrates some of the monitoring statistics that may be used and what adjustments to (or tuning of) the database environment may result.

Changes in....	**May require tuning of ....**				
	database structure	**access method used**	**hardware or software configuration**	**processing priority**	**disk storage allocation**
terminal and line traffic		Y	Y	Y	Y
response times (application performance)	Y	Y	Y	Y	Y
access totals by user and descriptor	Y	Y			Y
database size	Y	Y	Y		Y
database growth rate	Y	Y	Y		Y

When any alteration is made to a production database, care must be taken to ensure a continued high level of reliability and integrity. Whatever the change, the DBA must make sure that the decision is the right one and that it is properly and accurately implemented. He should retain absolute control over the tuning process and ensure that it follows the formal acceptance procedures.

The DBA must be careful not to overreact to changes in the items listed in the table. A sudden change in line traffic, response times, etc., may only be temporary. It is important to determine whether the change represents a permanent trend or a temporary disturbance to the normal way of operating.

The table can be used to determine what tuning may be necessary when a new project will cause a significant change in terminal and line traffic, response times, etc. The DBA can then act in advance to minimize these effects before the new application system is implemented.

Adabas Session Statistics

The statistics printed at the end of each Adabas session may be used to monitor Adabas performance. Specifically, the session statistics comprise

- input/output (I/O) statistics;
- command statistics; and
- pool/queue usage statistics.

Input/Output Statistics

The following I/O statistics are provided:

I/O Counts (Including Initialization)

	Reads	**Writes**
ASSO	50	21
DATA	2388	2184
WORK	9	1385
PLOG	9	1603
CLOG	0	0
TOTAL:	2456	5193
LOG. READS	33899	
BUFFER EFF.	13.9	

The input/output (I/O) counts represent the number of physical I/Os executed during the session to the Associator (ASSO), Data Storage (DATA), Work (WORK), the data protection log (PLOG), and the command log (CLOG).

Also provided are the number of logical reads issued for the buffer pool (LOG. READS) and the buffer efficiency (BUFFER EFF.) which is the number of logical reads divided by the number of Associator and Data Storage reads. The higher the value for buffer efficiency, the more efficient is buffer pool usage. If the value is less than 10, the DBA may wish to increase the size of the Adabas buffer pool (see the Adabas Operations documentation, the ADARUN LBP parameter description).

Distribution of ASSO/DATA I/Os by VOL-SER Number (Excluding Initialization)

VOL-SER	HIGH RABN	COUNT
ADA003	(ASSO: 894)	38
ADA003	(ASSO: 2544)	6
ADA003	(DATA: 894)	0
ADA003	(DATA: 1344)	4572
TOTAL:		4616

The distribution of I/Os for the Associator and Data Storage per physical volume is also provided. The data provided are the highest RABN accessed/updated (HIGH RABN) and the number of I/Os (COUNT). The DBA can use this data to determine if any adjustments are necessary to the buffer pool parameters and/or to the physical allocation of the database.

Command Statistics

In the following example, command statistics are provided for a session in which Adabas executed 12,687 calls in five threads.

Distribution of Commands by Source

The following table shows the source of commands for the session: either from the same environment (local) or from a remote environment across a network:

Source	Number
REMOTE LOGICAL	0
REMOTE PHYSICAL	0
LOCAL LOGICAL	0
LOCAL PHYSICAL	12,686

Distribution of Commands by Thread

The following table shows the thread activity for the session:

Thread	Number
1	7,328
2	2,728
3	1,240
4	814
5	541
TOTAL:	12,651

If the thread with the highest number has an activity count greater than zero it can be assumed that the Adabas nucleus would be able to process a larger number of commands if the number of threads were increased. Increasing the number of threads would prevent commands from waiting in the command queue for selection.

Distribution of Commands by File

The following table shows the distribution of commands by file:

File	Number
0	4,247
1	8,404
TOTAL:	12,651

Commands that are not file-related (e.g. BT, ET) are counted against file 0.

Distribution of Commands by Type

The following table shows the distribution of commands by command type:

Command Type	Number
A1/4	4,198
ET	4,191
L1/4	4,242
OP	56
TOTAL:	12,687

The command type UC indicates privileged call issued by Adabas utilities.

Note:
The command type REST indicates commands such as C1, C5, RI and HI.

Additional Session Statistics

```
THERE WERE      56 USERS PARTICIPATING
MOST CALLS (    57) INITIATED BY USER user ID
MOST I/O-S    (     14) INITIATED BY USER user ID
MOST THR.-TIME (04:16:32) WAS USED BY USER user ID
```

28	Formats had to be translated
0	Formats had to be overwritten
0	Autorestarts were done
20	Throw-backs due to ISN problem
16	Throw-backs due to space problem
186	Buffer-flushes were done

Formats Translated/Overwritten

Adabas read and update commands require a Format Buffer that specifies the fields to be read or updated. This Format Buffer is interpreted and converted into an internal Format Buffer by Adabas, which enters each resulting internal Format Buffer into the internal Format Buffer pool. Each internal Format Buffer is identified by a combination of user and command IDs.

For each new read/update command, Adabas looks to see if a user ID/command ID entry is already present in the format buffer pool. If not, Adabas translates the command's new format buffer and enters it into the pool. Once the format buffer pool becomes full, an existing entry must be overwritten to accommodate a new entry.

The format translation process is CPU intensive. Therefore, the DBA should ensure that an excessive number of format overwrites are not occurring by doing the following:

1. Ensure that user programs are making correct use of command IDs; that is, using non-blank command IDs when appropriate and releasing command IDs when no longer needed. For further information on command ID use, refer to the Adabas Command Reference documentation.

2. Consider increasing the size of the internal format buffer pool (with the ADARUN LFP parameter, described in the Adabas Operations documentation).

The Adabas nucleus produces statistics on format translations and format overwrites at the conclusion of each session. The Adabas operator command DSTAT may also be used to obtain this information.

Autorestarts

The number of Autorestarts performed during the session.

Command Throwbacks

The number of times a command could not be executed because the Adabas nucleus was waiting for

- an available ISN; or

- Adabas work pool space.

In such an event, the command is thrown back into the command queue for processing at a later point in time.

If either of these numbers is greater than zero:

1. adjust the ratio between the ADARUN LWP (work pool size) and LS (sort work area) parameters;

2. increase the size of the Adabas work pool (ADARUN LWP parameter);

3. evaluate ADARUN TT (transaction time limit) parameter;

4. check application program hold logic;

5. increase the Adabas hold queue size (ADARUN NH parameter); and

6. use superdescriptors to reduce complexity of search commands.

The ADARUN parameters are described in the Adabas Operations documentation.

Buffer Flushes

The number of buffer flushes performed during the session.

The Adabas buffer pool represents a virtual database that is shared by all active users. It contains the most frequently used Associator and Data Storage blocks, and its purpose is to minimize physical I/O activity.

The size of the buffer pool is determined by the ADARUN LBP parameter. LBP should be set as large as possible with the restriction that setting too large a value may cause excessive paging by the operating system.

Buffer and Queue Statistics

Session statistics include the maximum buffer and queue use during the session. These statistics are presented for all buffers and queues (except the buffer pool) for which high-water marks can be computed. The following table shows high-water marks for a sample session:

Pool Area	ADARUN Parameter	High-Water Mark	%
AB	NAB = 10	12032	29
CQ	NC = 20	3648	95
DUQ	LDEUQP = 5000	500	10
FI	LFP= 12000	1760	14
HQ	NH = 100	552	23
SC	LCP= 10000	0	0
TBI	LI = 10000	0	0
TBS	LQ = 10000	0	0
UQ	NU = 20	4880	86
UQF	NU = 20		
WORK	LWP = 14000	70464	50
XID	XID = 0	0	0

Note:
The UQF is the user queue extension that holds the file list. The size of its pool is computed using the UQ pool size.

The high-water marks are provided together with the applicable ADARUN parameter setting that was in effect for the session.

The DBA should monitor each high-water mark and, if necessary, make adjustments to the appropriate ADARUN parameters.

Command Logging

Adabas command logging may be used to generate information on all the commands issued by users to Adabas. Some of the information provided is

- user identification;
- time of day;
- the command used;
- the file accessed;
- the record accessed;
- the Adabas response code received;
- the time required for the command to perform.

Command logging is controlled by the ADARUN parameter LOGGING.