Database Availability for Transaction Processing

By Ananth Raghavan and T.K. Rengarajan

Abstract

A transaction processing system relies on its database management system to supply high availability. Digital offers a network-based product, the VAX DBMS system, and a relational data-based product, the VAX Rdb/VMS database system, for its transaction processing systems. These database systems have several strategies to survive failures, disk head crashes, revectored bad blocks, database corruptions, memory corruptions, and memory overwrites by faulty application programs.

They use base hardware technologies and also employ novel software techniques, such as parallel transaction recovery, recovery on surviving nodes of a VAXcluster system, restore and roll-forward operations on areas of the database, on-line backup, verification and repair utilities, and executive

Modern businesses store critical data in database management systems. Much of the daily activity of business includes manipulation of data in the database. As businesses extend their operations worldwide, their databases are shared among office locations in different parts of the world. Consequently, these businesses require transaction processing systems to be available for use at all times. This requirement translates directly to a goal of perfect availability for database management systems.

VAX DBMS and VAX Rdb/VMS database systems are based on network and relational data models, respectively. Both systems use a kernel of code that is largely responsible for providing high availability. This layer of code is maintained by the KODA group. KODA is the physical subsystem for VAX DBMS and VAX Rdb /VMS database systems.

mode protection of trusted database management system code.

Introduction

It is responsible for all I/O, buffer management, concurrency control, transaction consistency, locking, journaling, and access methods.

Database Availability for Transaction Processing

In this paper, we define database availability, and describe downtime situations and how such situations can be resolved. We then discuss the mechanisms that have been implemented to provide minimal loss of availability.

Database Availability

The unit of work in transaction processing systems is a transaction. We therefore define database availability as the ability to execute transactions. One way the database management system provides high availability is by guaranteeing the properties of transactions: atomicity, serializability, and durability.[1] For example, if a transaction that has made updates to the database is aborted, other transactions must not be allowed to see these updates; the updates made by the aborted transaction must be removed from the database before other transactions may use that data. Yet, data that has not been accessed by the aborted transaction must continue to be available to other transactions.

Downtime is the term used to refer to periods when the database is unavailable. of downtime are useful. Unexpected downtime is caused by factors that are beyond the control of the transaction processing system. For example, a disk failure is quite possible at any time during normal processing of transactions. However, scheduled downtime is entirely within the control of the database administrator. High availability demands that we eliminate scheduled downtime and ensure fast system recovery from unexpected failures.

The layers of the software and hardware services which compose a transaction processing system are dependent on one another for high availability. The dependency among these services is illustrated in Figure 1. Each service depends on the availability of the service in the lower layers. Errors and failures can occur in any layer, but may not be detected immediately. For example, in the case of a database management system, the effects of a database corruption may not be apparent until long after the corruption (error) has occurred. Hence it is difficult to deal with such errors. On the other hand, failures are noticed immediately. Failures usually make the

Downtime is caused by either an unexpected failure (unexpected downtime) or scheduled maintenance on the database (scheduled downtime). Such classifications

system unavailable and are the cause of unexpected downtime.

Each layer can provide only as much availability as the immediate lower layer. Hence we can also express the perfect-availability goal of a database management system as the goal of matching the availability of the immediately lower layer, which in our case is the operating system.

At the outset, it is clear that a database management system layered on top of an operating system and hence only as available as the underlying operating system. However, a database management system is in general not as available as the underlying layer because of the need to guarantee the properties of transactions.

Unexpected Downtime

In this section we discuss the causes of unexpected downtime and the techniques that minimize downtime.

A database monitor must be started on a node before a user's process running on that node can access a database. The monitor oversees all database activity on the node. It allows processes to attach to and detach from databases and detects

Application Program Exceptions

Although transaction processing systems are based on the client /server architecture, Digital's database systems are process based. The privileged database management system code is packaged in a shareable library and linked with the application programs. Therefore, bugs in the applications have a good chance of affecting the consistency of the database. Such bugs in applications are one type of failure that can make the database unavailable.

The VAX DBMS and VAX Rdb /VMS systems guard against this class of failure by executing the database management system code in VAX'S executive mode. Since application programs execute in user mode, they do not have access to data structures used by the database management system. When a faulty application program attempts such an access, the VMS operating system detects it and generates an exception. This exception then forces an image rundown of the application program.

In general, when an image rundown is initiated,

failures. On detecting a failure, the monitor starts a process to recover the transactions that did not complete because of the failure. Note that this database monitor is different from the TP monitor. [2]

Digital's database management products use the condition-handling facility of VMS to abort the transaction. Condition handling of image rundown is performed at two levels. Two condition handlers are established, one in

user mode and the other in kernel mode. The user mode exit handler is usually invoked, which rolls back the current transaction and unbinds it from the database. In this case, the rest of the users on the system are not affected at all. The database remains available. The execution of the user mode exit handler is, however, not guaranteed by the VMS operating system. Under some abnormal circumstances, the user mode exit handlers may not be executed at all. In such circumstances, the kernel mode exit handler is invoked by the VMS system. This handler resides in the database monitor. The monitor starts a database recovery (DBR) process. It is the responsibility of the DBR process to roll back the effects of the aborted transaction. To do this, the DBR process first establishes a database freeze. This freeze prevents other processes from acquiring locks that were held by the aborted transaction and hence see and update uncommitted data. (The VMS lock manager releases all locks held by a process when that process dies.) The DBR process then proceeds to roll back the aborted transaction.

Code Corruptions

It is important to prevent is examined at different points in the code. If any inconsistency is found, a bug-check utility is called that dumps the internal database format to a file. The utility then raises an exception that is handled by the monitor, and the DBR process is started as described above.

To deal with corruptions to the database that are undetected with this mechanism, an explicit utility is provided that verifies the structural consistency of the database. This verify utility may be executed on-line, while users are still accessing the database. Such verification may also be executed by a database administrator (DBA) in response to a bug-check dump. Once such a corruption is detected, an on-line utility provides the ability to repair the database.

In general, corruption in databases causes unexpected downtime. Digital provides the means of detecting such corruption on-line and repairing them on-line through recovery utilities.

Process Failure

In the VMS system, a process failure is always preceded by image rundown of the current image running as part of the

coding mistakes within the DBMS from irretrievably corrupting the database. To protect the database management system from coding mistakes, internal data structure consistency

process. Therefore, a process failure is detected by the database monitor, which then starts a DBR process to handle recovery.

Node Failure

Among the many mechanisms Digital provides for availability is node failover within a cluster. When a node fails, another node on the cluster detects the failure and rolls back the lost transactions from the failed node. Thus the failure of one node does not cause transactions on other active nodes of the cluster to come to a halt (except for the time the DBR process enforces a freeze). It is the database monitor that detects node failure and starts a recovery process for every lost transaction on the failed node. The database becomes available as soon as recovery is complete for all the users on the failed node.

Power Failure

Power failure is a hardware failure. As soon as power is restored, the VMS system boots. When a process attaches to the database, a number of messages are passed between the process that is attaching and the monitor. If the database is corrupt (because of power failure), the monitor is so informed by the attaching process, and again the monitor starts recovery processes to return the database to a consistent state. The database becomes available as soon as the recovery. The only differences in the case of process, node, or cluster failure is the mechanism by which the monitor is informed of the failure.

Disk Head Crash

Some failures can result in the loss or corruption of the data on the stable storage device (disk). Digital has a mechanism for bringing the database back to a consistent state in such cases.

A disk head crash is a failure of hardware that is usually characterized by the inability to read from or write to the disk. Hence database storage areas residing on that disk are unavailable and possibly irretrievable. A disk head crash automatically aborts transactions that need to read from or write to that disk. In addition, recovery of these aborted transactions is not possible since the recovery processes need access to the same disk. In this case, the database is shut down and access is denied until the storage areas on the failed disk are brought on-line. Areas are restored from backups and then rolled forward until consistent with the rest of the database. The after image journal (AIJ)

recovery is complete for all such failed users.

As described above, recovery is always accomplished by the monitor process starting DBR processes to do files are used to roll the areas forward. As soon as all the areas on the failed disk have been restored onto a good disk and rolled forward, the database becomes available.

Bad Disk Blocks

Bad blocks are hardware errors that often are not detected when they happen. The bad blocks are revectored, and the next time the disk block is read, an error is reported. Bad blocks simply mean that the contents of a disk block are lost forever. The database administrator detects the problem only when a database application fails to fetch data on the revectored block. Such an error may cause a certain transaction or a set of transactions to fail, no matter how many attempts are made to execute the transactions. This failure constitutes reduced availability; parts of the database are unavailable to transactions. Exactly how much of the database remains available depends on which blocks were revectored.

The mechanism provided to reduce the possible downtime is early detection. Digital's database systems provide a verification utility that can be executed while users are running transactions. The verification utility checks the structural consistency of the database. Once a bad block is detected by such a utility, that area of the database may be restored and rolled forward. These off against the downtime needed to restore and roll forward.

Site Failure

A site failure occurs when neither the computers nor the disks are available. A site failure is usually caused by a natural disaster such as an earthquake. The best recourse for recovery is archival storage. Digital provides mechanisms to back up the database and AIJ files to tape. These tapes must then be stored at a site away from the site at which the database resides. Should a disaster happen, these backup tapes can be used to restore the database. However, the recovery may not be complete. It cannot restore the effects of those committed transactions that were not backed up to tape.

After a disaster, the database can be restored and rolled forward to the state of the completion of the last AIJ that was backed up to tape. Any transactions that committed after the last AIJ was backed up cannot be recovered at the alternate site. Such transaction losses can be minimized by frequently backing up the AIJ files.

Memory Errors

two operations make the whole database temporarily unavailable; however, the bad block is corrected, and future downtime is avoided. The downtime caused by the bad block may be traded

Memory errors are quite infrequent, and when they happen, they usually are not detected. If the error happens to a data record, it may never be detected by any utility, but may

be seen as incorrect data by the user. If the verification utility is run on-line, it may also detect the errors. Again, the database may only be partially available, as in the case of bad blocks. However, it is possible to repair the database while users are still accessing the database. Digital's database management products provide explicit repair facilities for this purpose. The loss of availability during repair is not worse than the loss due to the memory error itself.

As explained previously, the database monitor plays an important part in ensuring database consistency and availability. Most unexpected failure scenarios are detected by the monitor, which then starts recovery processes. In addition, some failures might require the use of backup files to restore the database.

Scheduled Downtime

Most database systems have scheduled maintenance operations that require a database shutdown. Database backup for media recovery and verification to check structural consistency

Digital's database systems allow two types of transactions: update and "snapshot." The ability to back up data on-line depends on the snapshot transaction capability of the database.

Database backup is a standard way of recovering from media failures. Digital's database systems provide the ability to do transaction consistent backups of data on-line while users continue to change the database.

The general scheme for snapshot transactions is as follows. The update transactions of the database preserve the previous versions of the database records in the snapshot file. All versions of a database record are chained. Only the current version of the record is in the database area. The older versions are kept in the snapshot area. The versions of the records are tagged with the transaction numbers (TSNs). When a snapshot transaction (for example, a database backup) needs to read a database record, it traverses the chain for that database record and then uses the appropriate version of the record.

There are two modes

are examples of operations that may require scheduled downtime. In this section we describe ways to perform many of these operations while the database is executing transactions.

Backup

of database operation with respect to snapshot activity. In one mode, all update transactions write snapshot copies of any records they update. In the deferred snapshot mode, the updates cause

snapshot copies to be written only if a snapshot transaction is active and requires old versions of a record. In this mode, a snapshot transaction cannot start until all currently active update transactions (which are not writing snapshot records) have completed; that is, the snapshot transaction must wait for a quiet point in time. If there are either active or pending snapshot transactions when an update transaction starts, the update transaction must write snapshot copies.

Here we see a trade-off between update transactions and snapshot transactions. The database is completely available to snapshot transactions if all update transactions always write snapshot copies. On the other hand, if the deferred snapshot mode is enabled, update transactions need not write snapshot copies if a snapshot transaction in not active. This approach obviously results in some loss of availability to snapshot transactions.

Verification

Database corruption can also result in downtime. Although database corruption is not probable, it is possible. Any database system that database to check the structural consistency of the database. These utilities may also be executed on-line through the use of snapshot transactions.

AIJ Backup

The backup and the AIJ log are the two mechanisms that provide media recovery for Digital's database management products. The AIJ file is continuously written to by all user processes updating the database. We need to provide some ability to back up the AIJ file since it monotonically increases in size and eventually fills up the disk. Digital's database systems offer the ability to back up the AIJ file to tape (or another device) on-line. The only restriction is that a quiet point must be established for a short period during which the backup operation takes place. A quiet point is defined as a point when the database is quiescent, i.e., there are no active transactions.

On-line Schema Changes

Digital's database management systems allow users to change metadata on-line, while users are still accessing the database. Although this may

supports critical data must provide facilities to ensure the consistency of the database. Digital's database management systems provide verification utilities that scan the

be standard for relational database management systems, it is not standard for network databases. The VAX DBMS system provides a utility called the database restructuring utility (DRU)

to allow for on-line schema modifications.

Acknowledgments

Many engineers have contributed to the development of the algorithms described in this paper. We have chosen not to enumerate all such contributions. However, we would like to recognize the contributions of Peter Spiro, Ashok Joshi, Jeff Arnold, and Rick Anderson who, together with the authors, are members of the KODA team.

References

1. P. Bernstein, W. Emberton, and V. Trehan, "DECdta - Digital's Distributed Transaction Processing Architecture," Digital Technical Journal, vol. 3, no. 1 (Winter 1991, this issue): 10-17.
2. T. Speer and M. Storm, "Digital's Transaction Processing Monitors," Digital Technical Journal, vol. 3, no. 1 (Winter 1991, this issue): 18-32.