Operating System Support for Database System

---- Michael Stonebraker.

Summary

The main ideas discussed in this paper are:

- Examining several operating system services such as buffer pool management, the file system, scheduling, process management and interprocess communication, and consistency control and indicating whether they are appropriate for support of database management functions.
- The above services that the operating system provided has too much overhead or too slow to perform an operation for the database. As a result, database applications have to provide their own services instead of using the operating systems' provided services.
- Problems with Operating system services are listed below:

1. **Buffer Pool Management**: The overhead of fetching a block from the buffer pool manager is very high. So many DBMS implement their own buffer pools in user space. LRU is a worst replacement for DBMS. So, OS should be allowed to take advice from applications on replacement strategy. OS cannot implement correct prefetch strategy for DBMS because the block that needs to be accessed next will not necessarily in the next block in logical level. Most DBMS maintain an intentions list and commit flag and process them in an idempotent manner. During recovery from crash, the commit flag is examined and recovery utility process the intentions list correctly and update changes if the flag is set. If not the utility removes the intention list and back out the transaction. Where as in buffer pool manager, the commit flag must be forced to disk to commit the transaction. The service required from OS buffer manager is a selected force out of intentions list and commit flag to disk in proper order. The strategy used by most DBMS is to maintain a separate cache in user space and this buffer pool is managed by DBMS specific algorithm.

2. **The file system**: Two approaches to file systems include character arrays and record management systems. Multilevel directories, hashing and secondary indexes are used to provide this service. Character arrays are usually expanded one block at a time, which frequently results in blocks of a file being scattered across disk. Sequential storage is desired for efficient DBMS storage, so this is exceptionally bad for DBMS. Unix implements two services using trees. The blocks in given file are kept track of in a tree pointed to by i-node. The files in given mounted file system have a user visible hierarchical structure composed of directories. DBMS adds a third tree structure to support keyed access via multilevel directory structure. One tree with all kinds of information is effective than three separately managed trees.

3. **Scheduling, Process Management and Interprocess Communication**: Two ways of organizing a multiuser database system is process-per-user and server-process model. Most OS favors the first approach. In Process-per-user approach, Task switching costs a thousand of instructions causing overhead to buffer pool.

The server process model, forces the server process to do its own scheduling and multitasking, which results in duplication of OS facilities. To avoid this, we can use first-come-first-served server with no internal parallelism or we can have a collection of servers where each send low level requests to group of disk processes to perform the I/O and handle locking. But both process per user model and server model are not attractive because of overhead in OS due to task switches and messages. One solution for this is either providing fast path functions for DBMS or to create a special scheduling class for DBMS where the processes will never be forcibly descheduled but voluntarily relinquish CPU at appropriate intervals.

4. **Consistency Control**: The services provided by OS include locking objects for shared access and support for crash recovery. If DBMS provides buffer management in addition to OS, The transaction management by OS is effected because the buffer manager must maintain its own intentions list and cannot be immune from knowledge of transactions and OS functions are duplicated.

5. **Paged Virtual Memory**: The problems in binding large files into user's paged virtual address space are the I/O operation may produce faults for page containing page table and for data. To avoid these problems, we can wire down a large page table in main memory or to bind chunks of file into one's address space. All the problems with buffering exist even in paged virtual memory.

- **Conclusion**:

  The paper showed the main issues of the operating system support for databases. The engine of a database uses a similar architecture like what the operating system provides.

  All of the issues existed because the services that the operating system provided has too much overhead or too slow to perform an operation for the database. As a result, database applications have to provide their own services instead of using the operating systems' provided services. So, It is important that future OS should be more sensitive towards DBMS.

- **Question**:

  The OS is designed to support many other functionalities and other applications.

  Is it advisable to make changes to OS for supporting DBMS or make changes to DBMS to support all OS?

Submitted By

Lakshman Madhav Kollipara

SID- 932655504

Submitted To

Arash Termehchy

CS540- Database Management Systems