Understanding Windows File System Transactions

Christian Allred

Senior Software Development Engineer

Microsoft

Agenda

- Filesystems Today
- Transactional NTFS (TxF)
- TxF Scenarios
- Distributed TxF Transactions
- Challenges Implementing TxF

Filesystems Today

- No user-controllable atomicity
  - Difficult to present changes to multiple files as happening simultaneously
- Error recovery difficult in some scenarios
  - Example: Installer programs need to undo all changes in the event of an error, can be complex
- Can't get durability while taking advantage of cached I/O speed
  - Have to use write-through for user data; metadata changes require disk flush to make durable

Meet Transactional NTFS (TxF)

- What is TxF?
  - A feature of NTFS, introduced in Windows Vista
  - Provides database-like ACID semantics in the NTFS file system to application programs
    - Atomicity
    - Consistency
    - Isolation
    - Durability

What TxF Brings


```
HANDLE Transaction, File1;
BOOL WorkDone = FALSE;
DWORD TransactionOutcome;

Transaction = CreateTransaction( NULL, NULL, 0, 0, 0, 0, NULL );

try {
    File1 = CreateFileTransacted( File1Name,
                                  GENERIC_READ | GENERIC_WRITE, ...,
                                  Transaction, NULL, NULL );

    if (!WriteFile( File1, ... )) { leave; }

    if (!DeleteFileTransacted( File2Name, Transaction )) { leave; }

    WorkDone = TRUE; //  By setting TRUE, we commit rather than roll back in finally

} finally {

    CloseHandle( File1 );

    if (WorkDone) {
        if (!CommitTransaction( Transaction )) {
            GetTransactionInformation( Transaction, &TransactionOutcome, ... );
        }
    }

    CloseHandle( Transaction ); //  If Transaction not committed, it will roll back here
}
```

Scenarios for TxF – Installers

- Installers need to update multiple files and have either all updates persist, or none of them
  - File updates often accompanied by registry updates, in Windows handled by TxR, transaction extensions to the registry
- Windows Update uses TxF/TxR to handle system updates, including service pack installation
  - Almost 2/3 of their code was for error recovery; using TxF/TxR practically eliminated that code

Scenarios for TxF – Website Update

- Website consists of multiple pages (files), many with links referring to other pages of the site
- Don't want to take site down for maintenance
- Don't want to present broken links to visitors
- Update in a transaction allows changes to multiple files while isolating those changes from visitors
- When transaction commits, all pages update simultaneously

Distributed TxF Transactions

- Transactions involving files on two or more volumes on the same machine, or involving files and the registry, are "distributed" from TxF's perspective
  - Subject to two-phase commit protocol
- TxF does not directly support remotely-distributed transactions, i.e. over SMB
  - An application that wants remotely-distributed transactions with TxF uses the Distributed Transaction Coordinator (DTC)

Distributed TxF Transactions


```
hr = DtcGetTransactionManagerEx( NULL, NULL,
         IID_ITransactionDispenser, OLE_TM_FLAG_NONE,
         NULL, (void**) &pITransactionDispenser );

hr = pITransactionDispenser->BeginTransaction(
         NULL, ISOLATIONLEVEL_READCOMMITTED,
         ISOFLAG_RETAIN_BOTH, NULL, &pITransaction );

hr = pITransaction->QueryInterface( IID_IKernelTransaction,
                                    (void**) &pKernelTransaction );

hr = pKernelTransaction->GetHandle( &hTransactionHandle );

hAppend = CreateFileTransacted( TEXT("test.txt"),
                                FILE_APPEND_DATA, FILE_SHARE_READ,
                                NULL, OPEN_ALWAYS,
                                FILE_ATTRIBUTE_NORMAL, NULL,
                                hTransactionHandle, NULL, NULL );

... // work with the file using its file handle "hAppend"

CloseHandle( hAppend );

hr = pITransaction->Commit( FALSE, XACTTC_SYNC_PHASEONE, 0 );
```

Challenges Implementing TxF

- Complexity
  - TxF accounts for ~30% of NTFS driver size on AMD64
  - MSDN lists 19 new Win32 *Transacted() file/registry APIs, 22 file I/O APIs whose behavior is affected by TxF
  - Applications using transactions must be aware that transaction rollback may happen at any time
- Environmental Constraints
  - TxF's crash resiliency on SATA disks is not as good as on SCSI
    - FUA often not supported or enabled on SATA

Challenges Implementing TxF

- We did too much
  - Advanced features
    - Secondary (user-defined) resource managers
    - Support for transacting changes to alternate data streams
    - Savepoints – user-defined points in time to which a transaction may be partially rolled back
    - Miniversions – user-defined points at which a caller may get a frozen-in-time view of a file
  - All to be deprecated

Questions?

Extra Details

- Resource Manager
  - Entity that enables system resources (e.g. files, registry keys) to take part in a transaction
  - Each resource manager maintains metadata and a log associated with transactional work
  - TxF is the resource manager for the filesystem
    - Each volume in a Windows system is a TxF resource manager
  - TxR is the resource manager for the registry
    - Each Windows registry hive is a TxR resource manager

Extra Details

- Kernel Transaction Manager (KTM)
  - The transaction coordinator, implemented as a set of routines in the Windows kernel
  - Provides abort, commit abilities, coordinates multiple resource managers
  - Applications typically communicate with KTM, rather than TxF directly, to affect transaction state.
    - For example, CreateTransaction(), CommitTransaction(), RollbackTransaction() are calls into KTM

Extra Details

- Common Log File System (CLFS)
  - Provides logging facility to KTM, TxF, TxR, etc.
  - Each resource manager and KTM has a CLFS log where it records the actions it intends to take
  - The log also enables transaction rollback and recovery

For More Information

- *Windows Internals, 5th Edition*, pp. 965-974
- MSDN Magazine, "Enhance Your Apps With File System Transactions"
  - http://msdn.microsoft.com/en-us/magazine/cc163388.aspx
- MSDN: Transactional NTFS (TxF)
  - http://msdn.microsoft.com/en-us/library/bb968806(VS.85).aspx