The impossible while you wait, Miracles take slightly longer.

Marco Greco
IBM UK Ltd

Alternate
Monday, April 28, 2008 • 09:00 a.m. – 10:30 a.m.

2008 IIUG Informix Conference

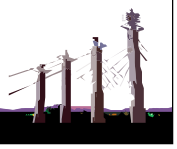
www.iiug.org

About me

- Member of the Advanced Problem Diagnostics L3 team in Europe
- With Informix / IBM since 1998
- Past chairman of the Informix on Linux IIUG committee
- Informix user for two decades

Agenda

- Rollforward or rollback replay errors
- Long Transactions
- Engine not switching to next log
- Conversion issues
- Down Chunks
- Overlapping Chunks
- Overlapping extents

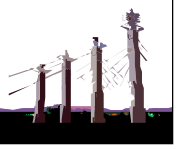


Agenda

- Lost partitions
- Dangling databases
- Corrupted system catalogs
- Oninit –i!
- Inability to restore

Rollforward or rollback replay errors

```
14:36:10 Rollback error 186
14:36:10 Assert Failed: Rollback error 186
14:36:10 IBM Informix Dynamic Server Version 10.00.FC8
14:36:10 Who: Session(17247923, informix@192.168.0.101, 3208,
25dabb168)
Thread(17405972, sqlexec, 2645b04e0, 14)
File: rsextlog.c Line: 1715
14:36:10 Results: Log record (OLDRSAM:CINDEX) in log 59504, offset
0x1a3ac020 was not rolled back
14:36:10 Action: Use 'onlog' to view the transaction and repair
manually.
14:36:10 stack trace for pid 3405 written to
/appl/InformixDump/af.9bfc1235
14:36:11 See Also: /appl/InformixDump/af.9bfc1235
```



Well, the transaction can certainly be seen with onlog, as for repairing it manually…

Rollforward or rollback replay errors

Caused by

- Corrupted pages
- Corrupted indexes
- Logging issues
- Engine defects

Fuzzy checkpoints entail skipping physical logging and buffer flushing for certain classes of records.

On occasion, as a result of these activities being skipped, pages on disk have been known to be at odds with what logical recovery expects.

Rollforward or rollback replay errors

Preventing them

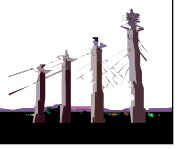
- Disable fuzzy checkpoints

Fuzzy checkpoints have been found to be the cause of recovery issues in a limited number of cases, which is why it may make sense to disable them, however be aware that a substantially larger class of recovery issues falls outside of the dba's control, and recovery failures should still be expected with fuzzy checkpoints turned off.

Rollforward or rollback replay errors

Tech support fixes – index corruption

- Indices can be marked as disabled
- Can also be marked as deferred
- Deferred indices rebuilt automatically when recovery ends
- Mandatory for system catalogs



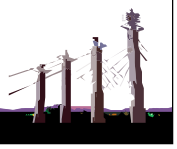
•Disabled indexes will be ignored during recovery. They can be dropped and recreated after recovery terminates.

•System catalog indices cannot be rebuilt through sql.

Rollforward or rollback replay errors

Tech support fixes – data / indices

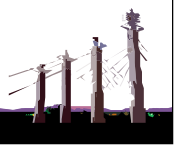
- Failures on log records located after checkpoint may be fixed by terminating fast recovery at failure point through log patches
- May imply rollback of some transactions after checkpoint, but no loss of data integrity



Rollforward or rollback replay errors

Tech support fixes – data / indices

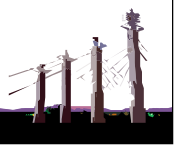
- Alternatively, failing logical log records can be patched as 'null' and skipped
- Mandatory for rollback failures located before checkpoint
- Implies controlled table inconsistencies
- Affected tables known
- These require manual consistency checks / rebuilds



Rollforward or rollback replay errors

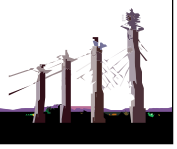
If everything else fails

- Recovery can be truncated entirely
- Except in specific circumstances, leaves substantial amounts of inconsistency
- Only contemplated as data rescue operation (eg lack of proper archive)



Long Transactions

- Use conservative watermarks
- Dynamic logs a workaround, not a solution
- Whatever you do, do not bring the engine down, unless the rollback is proven to be stuck
- When rollback hangs because of lack of space, tech support patch can add new log(s)

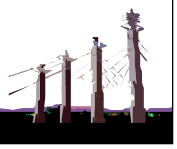


-In dire examples, dynamic logs can fill the whole instance with logs, and never get the transaction to be marked as long – as such they should only be used as a temporary mechanism to allow the rollback to complete.

-Bringing the engine down means that upon startup, during fast recovery, the instance will have to rollback the transaction again from the beginning, adding to the down time.

Engine not switching to next log

- Logs not being created dynamically?!
- Commonly due to LSN pointing to next log to be used
- Prevent issue by not using fuzzy checkpoints
- Tech support patch can add new log



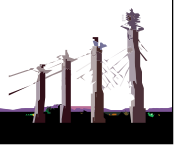
•The LSN points to the first logical log record not physically logged.

•Ideally, it should never be more than one log behind from the current checkpoint.

•In practice, under specific circumstances, certain transactions, while not being long, can still fill the logs to the point of reaching the log containing the LSN.

Conversion issues

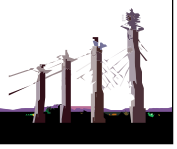
- Migration from one major version to the next done as single operation
- Cannot be restarted after failure
- Restore to previous version only way to guarantee version consistency after conversion failure



Conversion issues

Very little advanced support can do

- Recovery can be stopped after last successfully converted database
- Useful if non converted databases can be discarded or rebuilt
- Dropping non converted databases may fail
- Doesn't work more often than it does



•Generally there is very little that advanced support can do. If all of the non converted databases can be discarded or rebuilt, appropriate disk patching can be used to stop recovery after the last successfully converted database.

•Once the engine is online, dropping non converted databases through standard sql may very well fail. If it does, removing partitions can be done manually, but this entails substantial amounts of work.

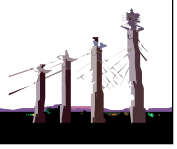
Conversion issues

Preventive actions

- Read the migration guide!
- Get rid of in place alters
- Oncheck everything
- Leave sufficient space in rootdbs and dbspaces containing system tables
- Avoid points of failure: drop unneeded databases
- Take an archive!
- Onmode –***u***ky
- Prevent users from connecting during conversion

Down Chunks

- Hardware issues
- Corruption
- User error

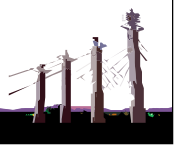


•Chunk added on top of another.

•Deleted chunks or symbolic links to chunk without dropping the chunk or dbspace first.

Down Chunks

- Under certain circumstances, can be marked online without data or consistency loss
- Logical log records not applied if chunk marked offline during fast recovery
- Marking chunk online akin to truncating recovery, described earlier (consider only for data rescue)
- Underlying disk problem must be fixed before chunks can be marked online
- Disk swaps require restore



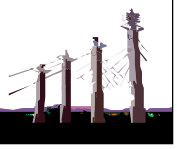
•Once a chunk is marked offline, subsequent transactions in the logical logs will not be able to alter the data it contains, but will apply changes to other chunks.

•Marking the chunk as online at the end of logical recovery will make the chunk inconsistent with itself, and other chunks.

Down Chunks

Deleted chunk files

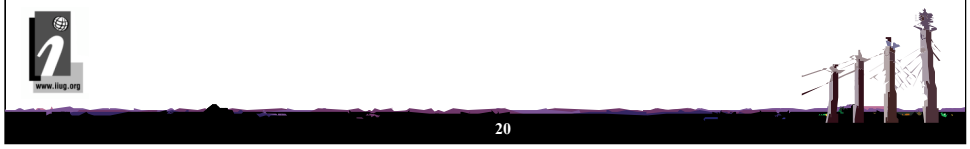
- Can be dropped with onspaces, or reconstructed
- This may leave dangling databases or tables



"I had completed the test and I didn't needed it anymore so I deleted the files"

Down Chunks

- Certain classes of down chunks can be avoided with ONDBSPACEDOWN



If a chunk is unavailable at startup for reasons other than hardware errors, ONDBSPACEDOWN set to 1 or 2 will prevent the next checkpoint from completing, and thus writing the chunk reserved pages to disk with the affected chunks marked as offline.

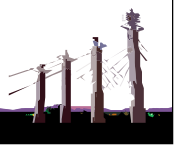
Upon fixing the issue, the instance can be restarted and the chunks will still be marked as online.

Overlapping Chunks

- Rogue instance created on top of existing instance
- New chunk added using symbolic link pointing to existing chunk
- Chunk can be fixed and marked online without data loss if
  - no activity has occurred on newly added chunk, and
  - new chunk not first in dbspace
- Chunk free list rebuild only required, new chunk can be dropped or moved somewhere else
- In all other circumstances, point in time restore to prior to adding new chunk required

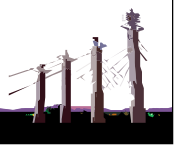
Overlapping extents

- Corruption (eg OS archive with engine in use + OS restore)
- One extent can be patched away without data loss, if located in chunk free list
- All other cases may result in data loss, consistency issues, corruption (patch only for data rescue)



Lost partitions

- Corruption (eg OS archives with engine in use + OS restore)
- User error
- Destructive patching
- Can be marked as temporary and dropped
- If schema is known can be 'reattached' to database

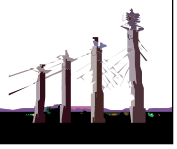


•Result of hacking systables or sysfragments.

•Terminating recovery or marking log records as null may result in lost partitions.

Dangling databases

- Corruption
- User error
- Can be easily patched away
- Feasible only if all tables (including system) are located in one dbspace / chunk…
- …only containing database to be dropped



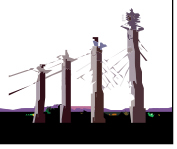
"I had finished testing and didn't need the database anymore, so I deleted the cooked files"

Corrupted system catalogs

- Corrupted indexes can be marked as bad and deferred - will be rebuilt after fast recovery
- Under certain circumstances corrupted partitions can be swapped for empty partitions
- Eg sysprocedures / sysprocplan
- Stored procedures will have to be recreated
- Fragmentation by expression sysfragments corruption can be fixed marking affected tables as fragmented by round robin

Oninit –i!

- We are in the miracles realm
- Only information lost is knowledge of first few chunks and databases residing at beginning of root chunk
- Sysutils will have been overwritten

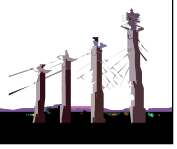


•All other information, for instance root dbspace partition partition additional extents, partition pages, extended reserved pages, is preserved, although not directly accessible.

•Losing sysutils may very well be a moot point, since a search and rescue operation wouldn't be required if an archive was available!

Oninit –i!

- Onstat –d output and database list make things easier
- Strategy is to look for systables partition pages
- Otherwise, need to scan chunks looking for
  - chunk reserved pages
  - chunk free list pages
  - partition bitmap pages
  - partition partition pages
- Heuristic strategy
- Massive amounts of work – rarely worth the effort
- Use oninit shell wrapper to avoid mishaps!

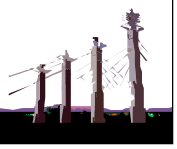


•Not an invitation to try advanced support skills! During the course of my employment at IBM I have seen this activity asked for three times, and done once.

•Unless the instance structure is known, finding any of the above doesn't mean finding a good item. Pages could be leftovers from a previous instance.

Inability to restore

- Corrupted media
- User error
- On occasion, engine defects



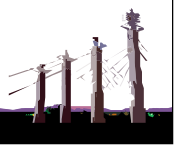
•Critical dbspaces haven't been archived. Ever.

•Parallel archive taken while logs are being discarded.

•External archive taken with engine online and not blocked.

Inability to restore

- In certain circumstances logical logs and checkpoint record can be created, chunks marked online (data rescue only)
- Critical dbspaces (other than rootdbs) that cannot be restored can (in part) be reconstructed, allowing restore to continue



•Up to version 10 it is possible to take parallel archives while logs are being discarded (LTAPEDEV set to a valid value in onconfig, but /dev/null in shared memory). A restore of such an archive would only go as far as physical recovery. This can be rescued by patching chunks online and fabricating logs and checkpoint records.

•After a restore of an OS archive taken without blocking the engine may be unable to find current checkpoint logical log record, depending on how the archive was taken. This too can be fabricated

•Even if the checkpoint record is found, the engine will most likely fail during logical recovery. This can be terminated for data rescue purposes.

•Where the dbspaces containing logical logs or physical log cannot be restored (eg they were never archived!), it is possible to fabricate chunk free list page partition partition bitmap page and partition partition partition page. Further, chunk and dbspace structures have to be altered to indicate that physical recovery has taken place. If such dbspaces contain nothing else, logical recovery will complete successfully.

Alternate

The Impossible While You Wait…

Marco Greco
IBM UK Ltd
marco@uk.ibm.com