Aakash Saini(2018CSM1001)

March 11, 2019

- **Observations:** Paper proposes the TxOS prototype to implement the strong atomicity and isolation for the transactions while maintaining the good performance. It provides ACID properties by implementing the code in appropriate system calls:sys_xbegin(), sy_xabort(), and sys_xend(). TxOS is object-based transnational system that supports checkpoints and rollbacks for kernel data structures. TxOS follows *lazy version management* in which transaction is done on private copies of data structures. For isolating read/write of data, TxOS uses existing OS buffer where it remain in buffer until commit and the application would not send the response in same transaction. Same locking mechanism is followed by all threads in the transaction to maintain the *fairness.*. A non transactional thread is suspended whenever conflict is there between them which in turn prevents the starvation. In the TxOS design, specific file system implementations are responsible only for not writing intermediate transaction results to disk and atomically writing a group of updates at commit. Application state is handled in sevral ways: automatically checkpoint is maintained by OS, using software or hardware transactional memory.

- **Limitations:**TxOS doesn't provide any transactional semantic for handling various classes of OS resources. TxOS doesn't provide the transactional functionality for the network communication protocols. TxOS doesn't support the IPC kernel and different system transactions.

- **Conclusions:** Addition of transactions management functionalities to the Linux API gives strength to the programmers for accessing the system resources in synchronized manner. TxOS solves number of problems system security and performance scalibilty.

- **Future work:**In future more transactional semantics can be provided to improve the performance of concurrent transaction system. Existing work can be extended to make communication possible between the kernel threads and different system transactions. Logging can be implemented in future for the debugging of aborted transactions.