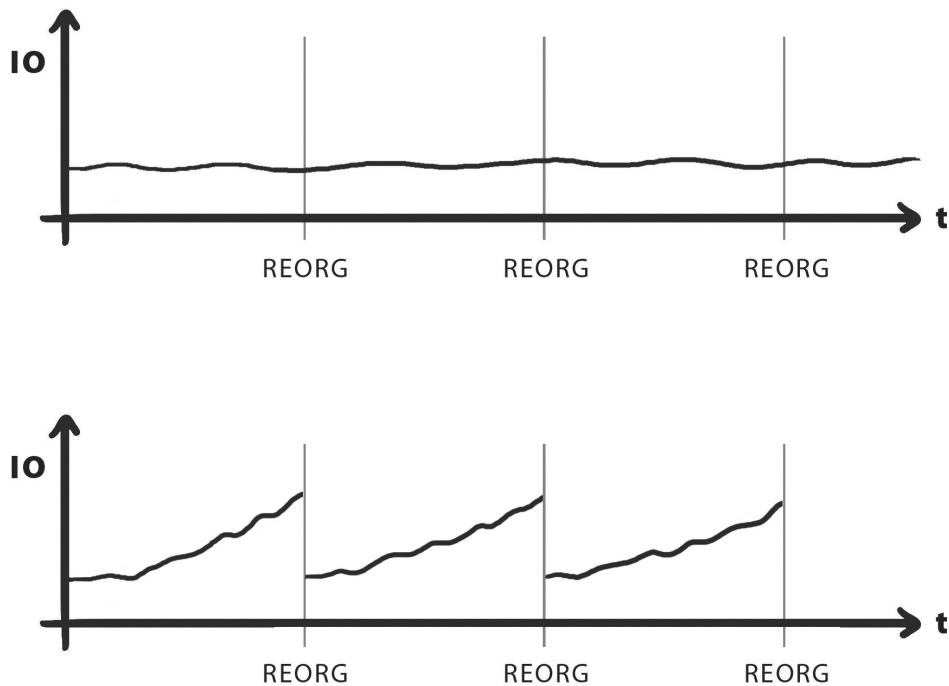




## REORG Avoidance

Performance depends on access path and may be severely affected if the need for REORG is ignored. However, in some cases, REORG does not improve the access performance and there is no point in REORG unless there is a space issue. To avoid unnecessary REORG, it is important to monitor the access performance and identify if there is any increase in the number of synchronous IO requests that is usually associated with performance degradation. The idea is to REORG objects only when there is an impact on performance.

TUC collects statistics trace records and is analyzing the access performance to justify REORG. An object is automatically excluded from REORG if it was identified as a candidate for REORG but does not suffer from increased IO. In addition, the analysis can be used to see if REORG indeed improved access performance and was justified.



In the diagram you can see 2 cases. In the first case, access performance remained the same despite the frequent reorgs. In this case, REORG is useless. In the second case, access performance improved dramatically after each reorg. The difficulty is to identify the optimal REORG point before performance is affected.

TUC collects IO statistics using DB2 Statistics Trace Class 8. This type of trace has a very low impact on your subsystem performance. DB2 writes very few statistics trace SMF records once in a statistics interval. The statistics trace can be also started automatically at DB2 startup. TUC analyses the access performance statistics and accumulates the counters for an accurate overlook of access patterns, before and after REORG.

Avoiding unnecessary REORG allows minimizing the cost of reorg and focus on critical objects that can benefit from REORG. TUC allows you to trigger REORG when there is a performance decrease and save the resources used by useless REORGs.

