APPENDIX

I. Theorem Proof

Theorem 1: Algorithm 1 dispatches traces in monotonically increasing order of before timestamps.

Proof 7: Supposing we have n_{local} local buffers and W is the watermark that is the smallest before timestamp among all traces in local buffers. According to Algorithm 1, the dispatched trace $\mathcal{T}$ satisfies:

$$\mathcal{T}.ts_{bef} \leq \min_{\mathcal{T}_i \in global} \mathcal{T}_i.ts_{bef} \tag{4}$$

$$\mathcal{T}.ts_{bef} \leq W = \min_{0 \leq i \leq n_{local}-1} local_i[0].ts_{bef} \tag{5}$$

Since the traces in each client C_i are generated in an increasing order of before timestamps, then we have:

$$\begin{aligned} local_i[0].ts_{bef} &\leq \min_{\mathcal{T}_j \in local_i} \mathcal{T}_j.ts_{bef} \\ &\leq \min_{\mathcal{T}_j \in C_i} \mathcal{T}_j.ts_{bef}, 0 \leq i \leq n_{local}-1 \end{aligned} \tag{6}$$

From Equations (1)-(3), we can infer that the dispatched trace has the minimum before timestamp among all traces in the global buffer, local buffers and clients. Thus theorem is proven.

Theorem 2: The candidate version set contains a minimum number of versions that are possibly visible to a given read operation.

Proof 8: Suppose a version x^i falls into the candidate version set but is impossibly visible to a given read operation. There exist two cases if x^i must be invisible to the read operation. In the first case, the version x^i appears after the read operation. Then, we can infer that the version installation time interval of x^i must not overlap with the visible snapshot time interval. Otherwise, x^i is possibly visible to the read operation since we could not determine the chronological order of the exact read operation time point and version installation time point. This implies that x^i is a future version*.*

In the second case, the version x^i appears before the read operation but has been overwritten by another version. As discussed above, we could not determine the chronological order of the pivot overlap version*,* pivot version *and* overlap version *since their exact arrive times are not available. That is, any version among them is possibly visible to the read operation. Thus, the version x^i must not be one of the three versions, which implies that x^i is a* garbage version*.*

Since our approach excludes all the future versions *and* garbage versions *from the candidate version set, this is contradicted with the initial assumption. The theorem is proven.*

Theorem 3: Given two transactions t_0 and t_1, for any overlapped time intervals of two conflicting locks, there exists at most one possible order in which a ww dependency can be deduced. Specifically, each of the other possible orders is identified to have incompatible locks.

Proof 9: Suppose there exist two possible orders in which two ww dependencies can be deduced. On the one hand, if t_0 has a ww dependency on t_1, then we can infer that the exact lock acquiring time of t_0 must happen before that of t_1. On the other hand, if t_1 has a ww dependency on t_0, then the exact lock acquiring time of t_1 must happen after the lock releasing time of t_0. To make the two ww dependencies possibly deduced, the lock acquiring time interval of t_0 (i.e., $\mathcal{A}^{\mathcal{T}_{w_{t_0}}}$) must overlap with the lock acquiring and releasing time intervals of t_1 (i.e., $\mathcal{A}^{\mathcal{T}_{w_{t_1}}}$ and $\mathcal{R}^{\mathcal{T}_{c_{t_1}}}$). Similarly, $\mathcal{R}^{\mathcal{T}_{c_{t_0}}}$ must also overlap with $\mathcal{A}^{\mathcal{T}_{w_{t_1}}}$ and $\mathcal{R}^{\mathcal{T}_{c_{t_1}}}$.

From $\mathcal{A}^{\mathcal{T}_{w_{t_0}}}$ overlaps with $\mathcal{A}^{\mathcal{T}_{w_{t_1}}}$ and $\mathcal{R}^{\mathcal{T}_{c_{t_1}}}$, we have $\mathcal{A}^{\mathcal{T}_{w_{t_1}}}.ts_{aft} < \mathcal{A}^{\mathcal{T}_{w_{t_0}}}.ts_{aft}$. Additionally, the lock releasing time interval must happen after the lock acquiring time interval, then we have $\mathcal{A}^{\mathcal{T}_{w_{t_0}}}.ts_{aft} < \mathcal{R}^{\mathcal{T}_{c_{t_0}}}.ts_{bef}$. Taken together, we have $\mathcal{A}^{\mathcal{T}_{w_{t_1}}}.ts_{aft} < \mathcal{R}^{\mathcal{T}_{c_{t_0}}}.ts_{bef}$, which indicates that $\mathcal{R}^{\mathcal{T}_{c_{t_0}}}$ would not overlap with $\mathcal{A}^{\mathcal{T}_{w_{t_1}}}$ and $\mathcal{R}^{\mathcal{T}_{c_{t_1}}}$ simultaneously. This is contradicted with the initial assumption. The theorem is proven.

Theorem 4: Given two committed transactions t_0 and t_1, for any overlapped time intervals of the two transactions, there exists at most one possible order in which a ww dependency can be deduced. Specifically, each of the other possible orders is identified to have concurrent versions.

Proof 10: The proof is similar to that of Theorem.3.

Theorem 5: A *garbage transaction* t is not a part of any future cycle on *DG*.

Proof 11: From C1 *in Definition 4, we can infer that the in-degree of t would be zero unless a future transaction creates a new dependency on t. Let $\mathcal{T}_k$ be the trace of the first operation of any committed transaction in future. From* C2 *in Definition 4, we can deduce that $\mathcal{T}_t.ts_{aft} \leq \mathcal{S}_e \leq \mathcal{S}^{\mathcal{T}_k}.ts_{bef}$. This indicates that any future transaction would not have dependencies on t. Taken together, the in-degree of t will keep as zero. However, the in-degree of* garbage transaction t *must be large than zero if t is contained in a cycle. Thus, t is not a part of any future cycle on* DG*. The theorem is proven.*