8 PROOF FOR HYBRID EXECUTION CORRECTNESS

In Snapper, the system state is a collection of the state of every named actor, and a transaction is a series of method invocations performed on one or more actors. Each method invocation can perform one or more Read or ReadWrite operations on an actor and invoke methods on some other actors via asynchronous RPCs. A transaction can invoke methods on the same actor multiple times.

We note that Read or ReadWrite operations on an actor always touch the whole of its state, so in the following we make no distinction between the notion of an actor and the data in its state. We use $r_i[x]$ or $w_i[x]$ to denote a Read or a ReadWrite, respectively, operation performed by transaction T_i on actor x. We focus on the concept of conflict serializability [14], so the ReadWrite operation can be regarded as a blind Write operation without being distinguished from a read-modify-write. We say any two read or write operations are conflicting if they are applied on the same actor and one of them is a write.

Besides, in Snapper, PACTs are committed in batches, thus a PACT batch can be considered as one large transaction. We denote an ACT as T_i^a, where i corresponds to the ACT's tid (a non-negative integer, $tid \in \mathbb{Z}^*$), and a batch as T_j^b, where j corresponds to the batch's bid (similarly $bid \in \mathbb{Z}^*$). Snapper guarantees that no two ACTs have the same tid and no two batches have the same bid. When we refer to T_i without further qualification, we denote either an ACT or a batch. We assume, without loss of generality, that the transaction identifiers corresponding to tids or bids be taken from disjoint subsets of Z^*.

Before we prove the correctness of Snapper's hybrid processing, we first define some key concepts similar to the formalism introduced by [14] for the classic transactional model. Definitions 8.1, 8.2, and 8.3 reuse the definitions of transaction, serialization graph, and history introduced in [14] with necessary adaptations to Snapper's context and introduce important notation. We state these definitions here to make this appendix self-contained.

Definition 8.1. In Snapper, a transaction T_i is a partial order with ordering relation $<_i$ where:

1. $T_i \subseteq \{r_i[x], w_i[x] \mid x \text{ is an actor}\} \cup \{a_i, c_i\}$, where a_i and c_i denote abort or commit, respectively;
2. $a_i \in T_i$ iff $c_i \notin T_i$;
3. if t is c_i or a_i (whichever is in T_i), for any other operation $p \in T_i$, $p <_i t$;
4. if $r_i[x], w_i[x] \in T_i$, then either $r_i[x] <_i w_i[x]$ or $w_i[x] <_i r_i[x]$.

To simplify the following discussion, we assume that a transaction T_i does not contain multiple operations of the same type on the same actor as in [14, page 27]. All the following results we get do not depend on this assumption.

Definition 8.2 (from [14]). Let $\mathbb{T} = \{T_1, T_2, ..., T_n\}$ be a set of transactions. A complete history H over $\mathbb{T}$ is a partial order with ordering relation $<_H$ where:

1. $H = \cup_{i=1}^{n} T_i$;
2. $\cup_{i=1}^{n} <_i \subseteq <_H$;
3. for any two conflicting operations $p, q \in H$, either $p <_H q$ or $q <_H p$.

Definition 8.3. The serialization graph SG for H, denoted $SG(H)$, is a directed graph, including a set of nodes $\mathbb{V}$ and edges $\mathbb{E}$:

1. $\mathbb{V} = \{T_i | T_i \subseteq H \text{ is a transaction } \land c_i \in T_i\}$
2. $\mathbb{E} = \{T_i \rightarrow T_j | T_i$ and T_j are different transactions, and there exist $o_i \in T_i, o_j \in T_j$ such that $o_i <_H o_j\}$

In the following, we slightly abuse notation by referring to $T_i \in SG(H)$ as a transaction in the set of nodes $\mathbb{V}$ of $SG(H)$. When the context is clear, we also refer to $T_i \rightarrow T_j$ without further qualification to denote an edge in the set of edges of $SG(H)$. Furthermore, we assume that histories generated by Snapper's hybrid processing always include at least one ACT transaction and one PACT batch.

To check global serializability under hybrid transaction processing, Snapper introduces the concepts of BeforeSet and AfterSet, which contain the scheduling information of each ACT.

Definition 8.4. Given a history H generated by Snapper's hybrid processing and the corresponding serialization graph $SG(H)$, $\forall T_i^a \in SG(H)$, its BeforeSet ($BS_{T_i^a}$) and AfterSet ($AS_{T_i^a}$) are defined as:

1. $BS_{T_i^a} = \{j |$ there exists a path $T_j^b \rightarrow ... \rightarrow T_i^a\}$
2. $AS_{T_i^a} = \{j | T_i^a \rightarrow T_j^b\}$

In addition, $max(BS_{T_i^a})$ and $min(AS_{T_i^a})$ are the maximum and minimum numbers (bids) in $BS_{T_i^a}$ and $AS_{T_i^a}$, respectively. If $BS_{T_i^a} = \emptyset$, $max(BS_{T_i^a}) = -1$. Similarly, if $AS_{T_i^a} = \emptyset$, $min(AS_{T_i^a}) = -1$.

Lemma 8.5. *Given a history H generated by* Snapper*'s hybrid processing and the corresponding serialization graph $SG(H)$, if $T_{i_1}^a \rightarrow T_{i_2}^a$, then $max(BS_{T_{i_1}^a}) \le max(BS_{T_{i_2}^a})$.*

Proof. If $BS_{T_{i_1}^a} = \emptyset$, then $max(BS_{T_{i_1}^a}) = -1 \le max(BS_{T_{i_2}^a})$. Otherwise, according to definition 8.4, $\forall T_j^b \in BS_{T_{i_1}^a}$, there exists a path from T_j^b to $T_{i_1}^a$ which can be extended by adding one more edge $T_{i_1}^a \rightarrow T_{i_2}^a$. Thus there is also a path from T_j^b to $T_{i_2}^a$. In another word $BS_{T_{i_1}^a} \subseteq BS_{T_{i_2}^a}$, so $max(BS_{T_{i_1}^a}) \le max(BS_{T_{i_2}^a})$. □

We propose Theorem 8.7 below to prove that Snapper's hybrid processing preserves conflict serializability for all concurrent transactions. Our proof relies on the serializability theorem (Theorem 8.6), which has been proven in [14].

Theorem 8.6 (from [14]). *A history H is conflict serializable iff $SG(H)$ is acyclic.*

Theorem 8.7. *A history H generated by* Snapper*'s hybrid processing is conflict serializable if:*

(1) $\forall T_{j_1}^b \rightarrow T_{j_2}^b$, $j_1 < j_2$;
(2) the execution of all T_i^a is conflict serializable;
(3) $\forall T_i^a \in SG(H)$, $max(BS_{T_i^a}) < min(AS_{T_i^a})$.

Proof. Here we prove that when the three conditions are met, then $SG(H)$ can be topologically sorted, which means that $SG(H)$ is acyclic and thus H is conflict serializable. More specifically, we first assign a unique rational number for each transaction T_i by applying a function $\mathcal{N}(T_i)$. Then, we take all transactions in ascending order of the assigned numbers to obtain a topological sort of $SG(H)$. To realize this proposed construction, we need to prove that given the three stated conditions, $\forall T_i \rightarrow T_j$, $\mathcal{N}(T_i) < \mathcal{N}(T_j)$.

Before defining $\mathcal{N}$, we introduce new transaction identifiers to all T_i^a corresponding to a valid serialization order. According to condition (2) above and Theorem 8.6, if the execution of all T_i^a is conflict serializable, then the induced sub-graph $SG(H)[\{T_i^a | T_i^a \in SG(H)\}]$ where the vertices consist of all T_i^a is acyclic. Thus, this induced subgraph can be topologically sorted. Suppose we have $m \in Z^+$ ACT transactions T_i^a, and $T_{i_1}^a, T_{i_2}^a, ..., T_{i_m}^a$ is such a topological sort. We can relabel the T_i^a with the identifiers in this topological sort so that now we know that $\forall T_{i_{k_1}} \rightarrow T_{i_{k_2}}, k_1 < k_2$, where $k_1, k_2 \in [1, m]$.

The function $\mathcal{N} : \mathbb{T} \mapsto \mathbb{Q}$ is now defined as follows:

- $\forall T_j^b \in SG(H), \mathcal{N}(T_j^b) = j$
- $\forall T_{i_k}^a \in SG(H), k \in [1, m], \mathcal{N}(T_{i_k}^a) = max(BS_{T_{i_k}^a}) + \frac{k}{m+1}$

Now we prove that $\forall T_i \rightarrow T_j, \mathcal{N}(T_i) < \mathcal{N}(T_j)$. We divide all $\rightarrow$ edges into four cases:

1. $\forall T_{j_1}^b \rightarrow T_{j_2}^b$,
$$\mathcal{N}(T_{j_1}^b) = j_1, \mathcal{N}(T_{j_2}^b) = j_2$$
With condition (1) above, $j_1 < j_2$, so $\mathcal{N}(T_{j_1}^b) < \mathcal{N}(T_{j_2}^b)$.
2. $\forall T_{i_{k_1}}^a \rightarrow T_{i_{k_2}}^a$,
$$\mathcal{N}(T_{i_{k_1}}^a) = max(BS_{T_{i_{k_1}}^a}) + \frac{k_1}{m+1}$$
$$\mathcal{N}(T_{i_{k_2}}^a) = max(BS_{T_{i_{k_2}}^a}) + \frac{k_2}{m+1}$$
In the topological sort of $SG(H)[\{T_i^a | T_i^a \in SG(H)\}]$ discussed above, $k_1 < k_2$, thus $\frac{k_1}{m+1} < \frac{k_2}{m+1}$, and according to Lemma 8.5, $max(BS_{T_{i_{k_1}}^a}) \leq max(BS_{T_{i_{k_2}}^a})$, so
$$\mathcal{N}(T_{i_{k_1}}^a) < \mathcal{N}(T_{i_{k_2}}^a)$$
3. $\forall T_j^b \rightarrow T_{i_k}^a$,
$$\mathcal{N}(T_j^b) = j$$
$$\mathcal{N}(T_{i_k}^a) = max(BS_{T_{i_k}^a}) + \frac{k}{m+1}$$
According to Definition 8.4, $j \in BS_{T_{i_k}^a}$, $j \leq max(BS_{T_{i_k}^a})$, thus
$$\mathcal{N}(T_j^b) < \mathcal{N}(T_{i_k}^a)$$
4. $\forall T_{i_k}^a \rightarrow T_j^b$,
$$\mathcal{N}(T_{i_k}^a) = max(BS_{T_{i_k}^a}) + \frac{k}{m+1}$$
$$\mathcal{N}(T_j^b) = j$$
According to Definition 8.4, $j \in AS_{T_{i_k}^a}$, $min(AS_{T_{i_k}^a}) \leq j$. And according to condition (3) above, $max(BS_{T_{i_k}^a}) < min(AS_{T_{i_k}^a})$, then $max(BS_{T_{i_k}^a}) < j$. And $\frac{k}{m+1} < 1$, so
$$\mathcal{N}(T_{i_k}^a) < \mathcal{N}(T_j^b)$$

$\square$