

# Verifying Transactional Consistency of MongoDB

(submitted to VLDB'2022)

Hengfeng Wei

hfwei@nju.edu.cn

December 26, 2021





# MongoDB 的三种经典部署架构

MongoDB 3.0	MongoDB 3.2	MongoDB 3.4	MongoDB 3.6	MongoDB 4.0	MongoDB 4.2
New Storage engine (WiredTiger)	Enhanced replication protocol: stricter consistency & durability	Shard membership awareness	Consistent secondary reads in sharded clusters	Replica Set Transactions	Distributed Transactions
	WiredTiger default storage engine		Logical sessions	Make catalog timestamp-aware	Oplog applier prepare support
	Config server manageability improvements		Retryable writes	Snapshot reads	Distributed commit protocol
	Read concern "majority"		Causal Consistency	Recoverable rollback via WT checkpoints	Global point-in-time reads
			Cluster-wide logical clock	Recover to a timestamp	More extensive WiredTiger repair
			Storage API to changes to use timestamps	Sharded catalog improvements	Transaction manager
			Read concern majority feature always available		
			Collection catalog versioning		
			UUIDs in sharding		
			Fast in-place updates to large documents in WT		

## MongoDB 事务的三阶段发展过程

## *A Fundamental Question:*

*What transactional consistency guarantee do MongoDB transactions in each deployment provide?*

挑战一：MongoDB 官方规约不清楚, SI 有多种变体

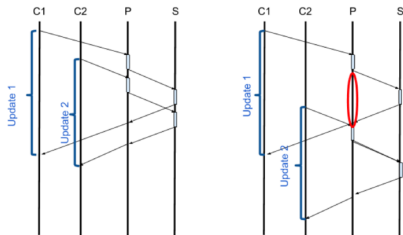
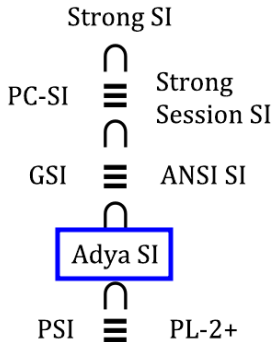


Figure 3: Back-to-Back Transactions with and without **Speculative Snapshot Isolation**



## 挑战二: MongoDB 缺少精简的事务协议描述, 更没有严格证明

mongodb / specifications Public

<> Code Issues 7 Pull requests 10 Actions Projects Security Insights

master specifications / source / transactions / transactions.rst

durran WRITING-6786: Load Balancer Spec (#939) ✓

11 contributors

1433 lines (1118 sloc) | 59.4 KB

### Driver Transactions Specification

Spec Title:	Driver Transactions Specification
-------------	-----------------------------------

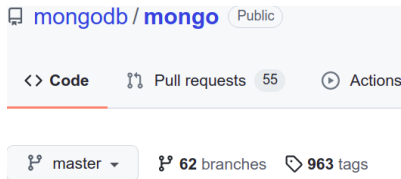
挑战三: SI 检测问题是 NP-complete 问题, 复杂度高

THEOREM 3.2. *For any criterion  $C \in \{\text{PREFIX CONSISTENCY}, \text{SNAPSHOT ISOLATION}, \text{SERIALIZABILITY}\}$  the problem of checking whether a given history satisfies  $C$  is NP-complete.*

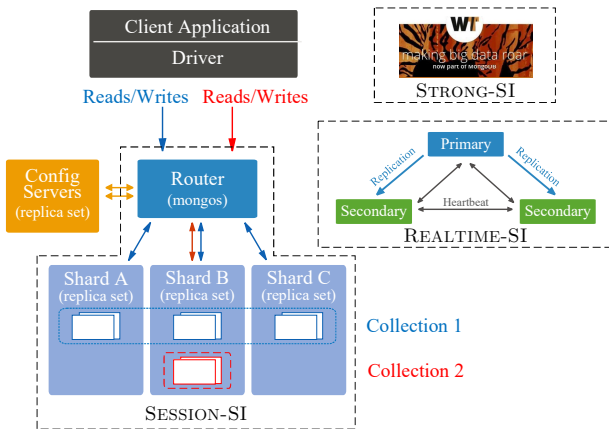


贡献一: 使用 (VIS, AR) 框架, 为多种 SI 变体提供形式化规约

## 贡献二：为 MongoDB 事务一致性协议提供精简的伪代码描述



### 贡献三: 证明 WIREDTIGER、REPLICASET、SHARDEDCLUSTER 事务协议分别满足 STRONGSI、REALTIME-SI、SESSIONSI 变体



贡献四：设计并评估了多项式时间 SI 变体白盒检测算法

# JEPSEN

1. 事务  $T : (E, \text{po})$

- ▶  $\text{po}$  : Program Order
- ▶  $\text{start}(T)$  : 事务开始时间
- ▶  $\text{commit}(T)$  : 事务提交时间

2. 历史  $\mathcal{H} : (\mathbb{T}, \text{so})$

- ▶  $\mathbb{T}$  : 已提交事务集合
- ▶  $\text{so}$  : Session Order

3. 执行  $\mathcal{A} : (\mathcal{H}, \text{vis}, \text{ar})$

- ▶  $\text{vis}$  : 可见性 (Visibility) 偏序关系
- ▶  $\text{ar}$  : 仲裁 (Arbitration) 全序关系
- ▶  $\text{vis} \subseteq \text{ar}$

一个事务一致性模型可定义为一组一致性公理的集合  $\Phi$ 。

历史  $\mathcal{H}$  满足事务一致性模型  $\Phi$ , 如果存在 VIS 与 AR 使得

$$\exists \text{VIS, AR. } (\mathcal{H}, \text{VIS}, \text{AR}) \models \Phi.$$

$\forall (E, \text{po}) \in \mathcal{H}. \forall e \in \text{Event}. \forall \text{key}, \text{val}. (\text{op}(e) = \text{read}(\text{key}, \text{val}) \wedge \{f \mid (\text{op}(f) = \_(\text{key}, \_) \wedge f \xrightarrow{\text{po}} e\} \neq \emptyset) \implies \text{op}(\max_{\text{po}} \{f \mid \text{op}(f) = \_(\text{key}, \_) \wedge f \xrightarrow{\text{po}} e\}) = \_(\text{key}, \text{val})$		(INT)	
$\forall T \in \mathcal{H}. \forall \text{key}, \text{val}. T \vdash \text{read}(\text{key}, \text{val}) \implies \max_{\text{AR}}(\text{VIS}^{-1}(T) \cap \text{WriteTx}_{\text{key}}) \vdash \text{write}(\text{key}, \text{val})$		(EXT)	
$\text{SO} \subseteq \text{VIS}$	(SESSION)	$\text{AR} ; \text{VIS} \subseteq \text{VIS}$	(PREFIX)
$\text{RB} \subseteq \text{VIS}$	(RETURNBEFORE)	$\text{CB} \subseteq \text{AR}$	(COMMITBEFORE)
$\text{VIS} \subseteq \text{RB}$	(REALTIMESNAPSHOT)	$\forall S, T \in \mathcal{H}. S \bowtie T \implies (S \xrightarrow{\text{VIS}} T \vee T \xrightarrow{\text{VIS}} S)$	(NOCONFLICT)

$$SI = INT \wedge EXT \wedge PREFIX \wedge NoCONFLICT$$



$$\text{SESSIONSI} = \text{SI} \wedge \text{SESSION}$$

$$\text{SESSIONSI} = \text{SI} \wedge \text{SESSION}$$

$$\text{REALTIMESI} = \text{SI} \wedge \text{RETURNBEFORE} \wedge \text{COMMITBEFORE}$$

$$\text{SESSIONSI} = \text{SI} \wedge \text{SESSION}$$

$$\text{REALTIMESI} = \text{SI} \wedge \text{RETURNBEFORE} \wedge \text{COMMITBEFORE}$$

$$\text{GSI} = \text{SI} \wedge \text{REALTIMESI} \wedge \text{COMMITBEFORE}$$

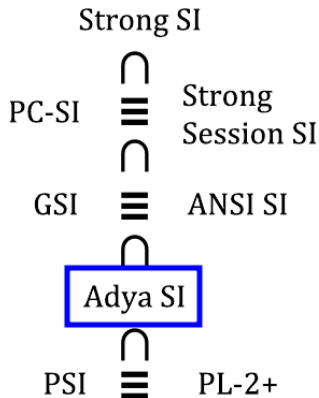
$$\text{SESSIONSI} = \text{SI} \wedge \text{SESSION}$$

$$\text{REALTIMESI} = \text{SI} \wedge \text{RETURNBEFORE} \wedge \text{COMMITBEFORE}$$

$$\text{GSI} = \text{SI} \wedge \text{REALTIMESI} \wedge \text{COMMITBEFORE}$$

$$\text{STRONGSI} = \text{GSI} \wedge \text{RETURNBEFORE}$$

- ▶ ANSI-SI
- ▶ SI
- ▶ GSI
- ▶ STRONGSI
- ▶ STRONGSESSIONSI
- ▶ PSI
- ▶ WRITESI
- ▶ NMSI
- ▶ PCSI





# Conclusion



Hengfeng Wei (hfwei@nju.edu.cn)