# Parameterized and Runtime-tunable Snapshot Isolation in Distributed Transactional Key-value Stores

**Hengfeng Wei**, Yu Huang, Jian Lu

Nanjing University, China

September 17, 2017

RVSI: Relaxed Version Snapshot Isolation

1. Definition of RVSI

RVSI: Relaxed Version Snapshot Isolation

1. Definition of RVSI

Transaction $T_i$:

- begins with a `start` operation $s_i$
- contains a sequence of `read` or `write` operations
- ends with a `commit` operation $c_i$ or an `abort` operation $a_i$

Transaction $T_i$:

- begins with a `start` operation $s_i$
- contains a sequence of `read` or `write` operations
- ends with a `commit` operation $c_i$ or an `abort` operation $a_i$

History: modelling an execution of a transactional key-value store

- $w_i(x_i)$: transaction $T_i$ writing version $i$ of data item $x$
- $r_i(x_j)$: transaction $T_i$ reading version $j$ of data item $x$ written by $T_j$
- *time-precedes partial order* $\prec_h$ over operations
- $T_i$ and $T_j$ are concurrent:

$$s_i \prec_h c_j \wedge s_j \prec_h c_i$$

Snapshot isolation requires that:

Snapshot Read: Each transaction read data from the "lastest" snapshot as of the time the transaction started.

Snapshot Write: No write-conflicting concurrent transactions

A history $h$ is in snapshot isolation if and only if it satisfies [Adya@Thesis'99]

Snapshot Read: All reads of transaction $T_i$ occur at $T_i$'s start time.

$$\forall r_i(x_{j \neq i}), w_{k \neq j}(x_k), c_k \in h :$$
$$(c_j \in h \land c_j \prec_h s_i) \land (s_i \prec_h c_k \lor c_k \prec_h c_j).$$

Snapshot Write: No concurrent committed transactions may write the same data item.

$$\forall w_i(x_i), w_{j \neq i}(x_j) \in h \implies (c_i \prec_h s_j \lor c_j \prec_h s_i).$$

# Principle of RVSI

- Parameters $(k_1, k_2, k_3)$ to control the severity of the anomalies w.r.t SI

[1] RC: Read Committed

# Principle of RVSI

- Parameters $(k_1, k_2, k_3)$ to control the severity of the anomalies w.r.t SI
- RC [1] $\supset$ RVSI$(k_1, k_2, k_3) \supset$ SI
- RVSI$(\infty, \infty, \infty) =$ RC    RVSI$(1, 0, *) =$ SI

[1] RC: Read Committed

# Principle of RVSI

...

– *"Snapshot Read" property of SI*

1. "stale" data versions

# Principle of RVSI

. . .

– *"Snapshot Read" property of SI*

1. "stale" data versions
2. "concurrent" data versions

# Principle of RVSI

...

– *"Snapshot Read" property of SI*

1. "stale" data versions
2. "concurrent" data versions
3. "non-snapshot" data versions

# Principle of RVSI

. . .

– *"Snapshot Read" property of SI*

1. "stale" data versions — bounded staleness
2. "concurrent" data versions
3. "non-snapshot" data versions

# Principle of RVSI

. . .

– *"Snapshot Read" property of SI*

1. "stale" data versions — bounded staleness
2. "concurrent" data versions — bounded concurrency level
3. "non-snapshot" data versions

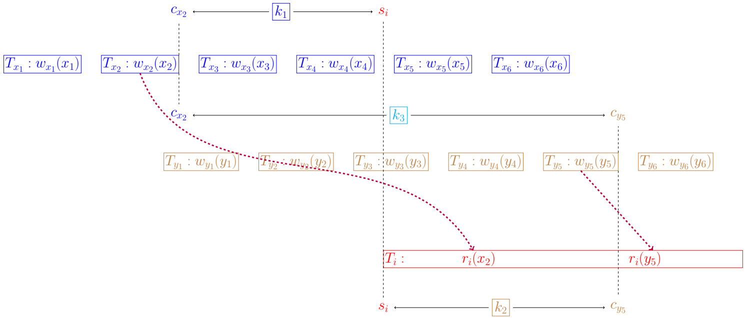
# Principle of RVSI

...

– *"Snapshot Read" property of SI*

1. "stale" data versions — bounded staleness
2. "concurrent" data versions — bounded concurrency level
3. "non-snapshot" data versions — bounded distance

# Illustration of RVSI



# Definition of RVSI

($k_1$-BV)

$$\forall r_i(x_j), w_k(x_k), c_k \in h : \left( c_j \in h \wedge \bigwedge_{k=1}^{m} (c_j \prec_h c_k \prec_h s_i) \right) \Rightarrow m < k_1,$$

($k_2$-FV)

$$\forall r_i(x_j), w_k(x_k), c_k \in h : \left( c_j \in h \wedge \bigwedge_{k=1}^{m} (s_i \prec_h c_k \prec_h c_j) \right) \Rightarrow m \leq k_2,$$

($k_3$-SV)

$$\forall r_i(x_j), r_i(y_l), w_k(x_k), c_k \in h : \left( \bigwedge_{k=1}^{m} (c_j \prec_h c_k \prec_h c_l) \right) \Rightarrow m \leq k_3.$$

# Definition of RVSI

$$h \in \mathsf{RVSI} \iff h \in k_1\text{-}\mathsf{BV} \cap k_2\text{-}\mathsf{FV} \cap k_3\text{-}\mathsf{SV} \cap \mathsf{WCF}.$$

**Theorem**

$$\mathsf{RVSI}(1, 0, *) = \mathsf{SI}.$$