HAL
archives-ouvertes.fr

# Eventual Consistency

Marc Shapiro, Bettina Kemme

▶ **To cite this version:**

Marc Shapiro, Bettina Kemme. Eventual Consistency. Özsu, M. Tamer and Liu, Ling. Encyclopedia of Database Systems, springer, 2009. <inria-00444791>

**HAL Id: inria-00444791**
**https://hal.inria.fr/inria-00444791**

Submitted on 7 Jan 2010

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Eventual Consistency

Marc Shapiro and Bettina Kemme

None

In a replicated database, the consistency level defines whether and how the values of the replicas of a logical object may diverge in the presence of updates. Eventual consistency is the weakest consistency level that guarantees convergence. Informally, it requires that all replicas of an object will eventually reach the same, correct, final value, assuming that no new updates are submitted to the object.

Eventual consistency is an important correctness criteria in systems with a lazy, update-anywhere strategy, also referred to as *optimistic replication* (q.v.). Update operations can be submitted and executed on any node, and the propagation of updates occurs lazily after commit. Conflict resolution and reconciliation must ensure that all replicas (copies) of a logical object eventually converge to the same value. Different objects are considered independent.

In a system where updates are continuously submitted, eventual consistency can be defined by a weak form of schedule equivalence [1]. A schedule $S_n^x$ describes the sequence of update operations node $n$ performs on its replica of object $x$. An element of $S_n^x$ represents the execution of an operation $w_i$ that operates on $x$, submitted by some user. $S_n^x$ contains $\overline{w_i}$ for an operation, if $w_i$ was received by $n$, but either not executed, or aborted due to conflict resolution.

Typically, two schedules are defined equivalent by restricting how the order of operations in the two schedules may differ. However, for eventual consistency only the convergence of object values matters. Thus, equivalence is defined by comparing the final state of the replicas. Two schedules are said *state equivalent* when, starting from the same initial state, they produce the same final state. For instance: (i) schedules $S = w_1 w_2$ and $S' = w_2 w_1$ are state-equivalent if $w_1$ and $w_2$ commute; (ii) schedules $S = w_1 w_2$ and $S' = w_2$ are state-equivalent if $w_2$ sets the state of the object to a completely new value (e.g., $x := 2$).

Eventual consistency of a replicated object $x$ is defined by the following conditions, which must hold at all times, at any node $n$ with a replica of $x$ [1]. It is assumed that all replicas have the same initial state.

- There is a prefix of the schedule $S_n^x$ that is state-equivalent to a prefix of the schedule $S_{n'}^x$ of any other node $n'$ holding a replica of $x$. Such a prefix is called a *committed prefix* of $S_n^x$.
- The committed prefix of $S_n^x$ grows monotonically over time, i.e., the set of operations and their relative order remains unchanged.

- For every operation $w_i$ submitted by a user, either $w_i$ or $\overline{w_i}$ eventually appears in the committed prefix of $S_n^x$ (but not both, and not more than once).
- An operation $w_i$ in the committed prefix satisfies all its preconditions (e.g., the state of the object immediately before the execution of the operation fulfills certain conditions).

As an example, assume operation $w_1$ sets $x$ to 2, and $w_2$ sets it to 5. Operation $w_1$ is submitted and executed at $n_1$ while $w_2$ is first executed at $n_2$.At this time the local schedules are $S_1 = w_1$ and $S_2 = w_2$ and the committed prefix at both nodes is the empty schedule. Now $w_1$ is propagated to $n_2$ and $w_2$ is propagated to $n_1$. When $n_1$ receives $w_2$ it must detect that $w_1$ and $w_2$ are concurrent and conflict. Conflict reconciliation could prioritize one of the operations, e.g., $w_1$. Then, $w_2$ is simply not executed and the new schedule is $S_1^x = w_1\overline{w_2}$. At $n_2$, when $w_1$ arrives, the conflict is also detected, $w_2$ is undone, $w_1$ is executed and the final schedule is $S_2 = \overline{w_2}w_1$. At this time, $S_1$ and $S_2$ are themselves the committed prefixes. Note that further concurrent operations on $x$ might move the schedules further, but the extensions would still be tentative and only become committed once they are reconciled at all replicas.

Optimistic Replication and Resolution, WAN Replication, Consistency Models for Replicated Data

# References

[1] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Computer Surveys*, 37(1): 42-81, 2005.