Update operations

- at source: some initial processing
- downstream: transmit updates asynchronously to all replicas

State-based replication

在state-based replication中，update操作仅仅在source源发生，而后，通过在各replicas之间传输更新后的payload，达到传递updates的目的.

为了传递updates，系统在任意对replicas之间传输状态state，其接收方通过merge操作更新自己的payload.

Op-based objects

在op-based replication中，系统传递的是操作operations. 其update操作分为两个阶段：第一阶段(at source)，在本地计算结果，并返回给caller，and/or 为第二阶段准备参数；第二阶段(downstream)，在其他所有的replicas出异步地更新downstream状态，并不会返回结果.

Causal Hostory of a replica x_i (marked $\mathcal{C}\left(x_i\right)$)

- 初始化: $\mathcal{C}\left(x_i\right)=\emptyset$
- downstream阶段在x_i上执行f 操作（marked $f\left(x_i\right)$）后，$\mathcal{C}\left(f\left(x_i\right)\right)=\mathcal{C}\left(x_i\right)\cup\{f\}$

Eventually Convergence (object x的两份replicas x_i和x_j)

- safety: 对于任意i和j，x_i和x_j的abstract state等价，即$\mathcal{C}\left(x_i\right)=\mathcal{C}\left(x_j\right)$
- liveness：对于任意i和j，$f\in\mathcal{C}\left(x_i\right)\Longrightarrow f\in\mathcal{C}\left(x_j\right)$ eventually

CvRDT

CvRDTs (Convergent Replicated Data Types) are state-based which means that every instance keeps the full history of all changes that have ever happened to the datastructure and is therefore self-contained and fault-tolerant by design. The convergent nature of the CvRDTs means that state updates do not need to obey any ordering requirements, the only requirement is that all state changes eventually reaches all replicas, and the system will converge.

Some definitions

- partial order $\leq_v$
- least upper bound (LUB) $\sqcup_v$,$\sqcup_v$ 满足交换律、幂等律和结合律

- Join Semilattice $(S, \leq_v)$

CvRDT form a *monotomic semilattice*

CmRDT

CmRDT (Commutative Replicated Data Types) are operations-based which means that it is the state changing events, and not the state changes themselves that are stored. These events are persisted in an event/transaction log and a CmRDT is brought up to its current state by simply replaying the event log. This solves the problem with unbound growth of history, as seen in CvRDTs, since the history is kept in a event log, outside the CmRDT. It also opens up for optimisations like sharding.

对于op-based object，不遵循delivery order的操作被称为*concurrent*，如果所有的concurrent operation可交换的话，则对应于delivery order的execution order相互等价，从而所有的replicas聚合于同一state