

9. Репликация данных и согласованность

Сухорослов Олег Викторович

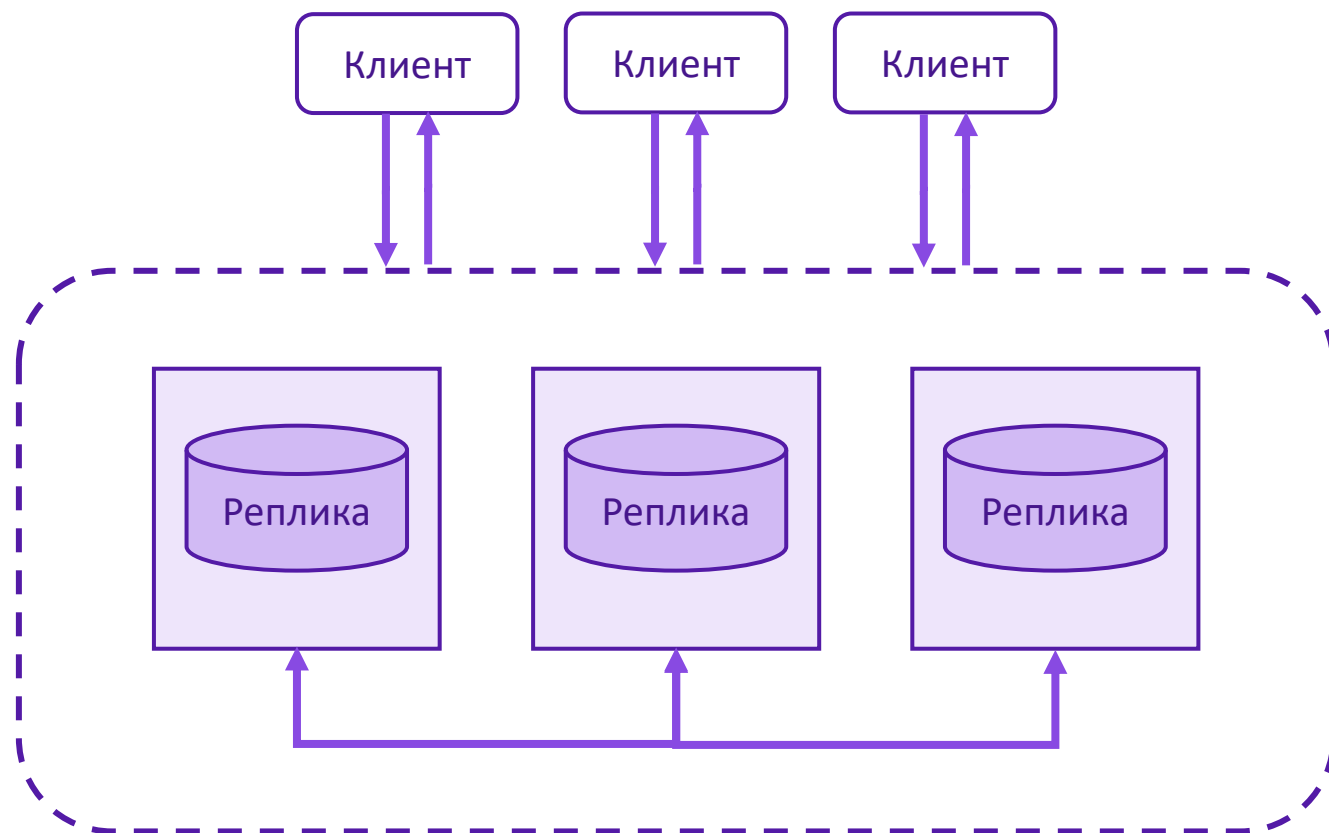
06.11.2023

План лекции

- Репликация данных, известные подходы к реализации
- Согласованность, связь с репликацией, известные модели

Репликация данных

Хранение физических копий (реplik) данных на нескольких узлах



Мотивация

- Обеспечение долговечности данных
- Повышение доступности
- Уменьшение задержки
- Увеличение производительности

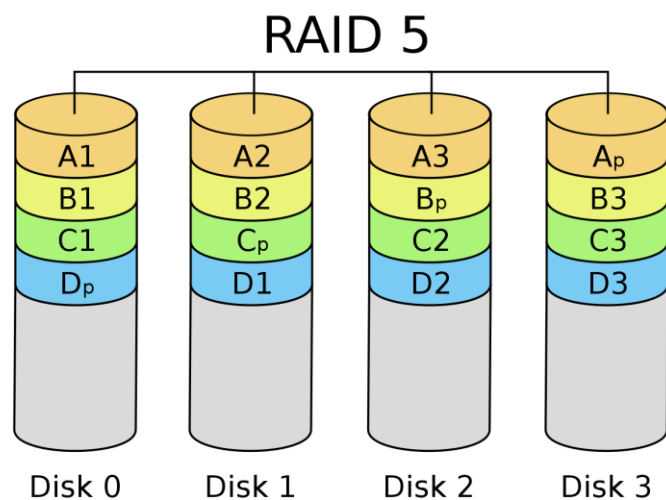
Число реплик	Вероятность отказа		
	≥ 1 реплики	$\geq (N+1)/2$ реплик	всех реплик
1	0,01	0,01	0,01
3	0,03	$3 \cdot 10^{-4}$	10^{-6}
5	0,049	10^{-5}	10^{-10}
100	0,63	$6 \cdot 10^{-74}$	10^{-200}

Недостатки репликации

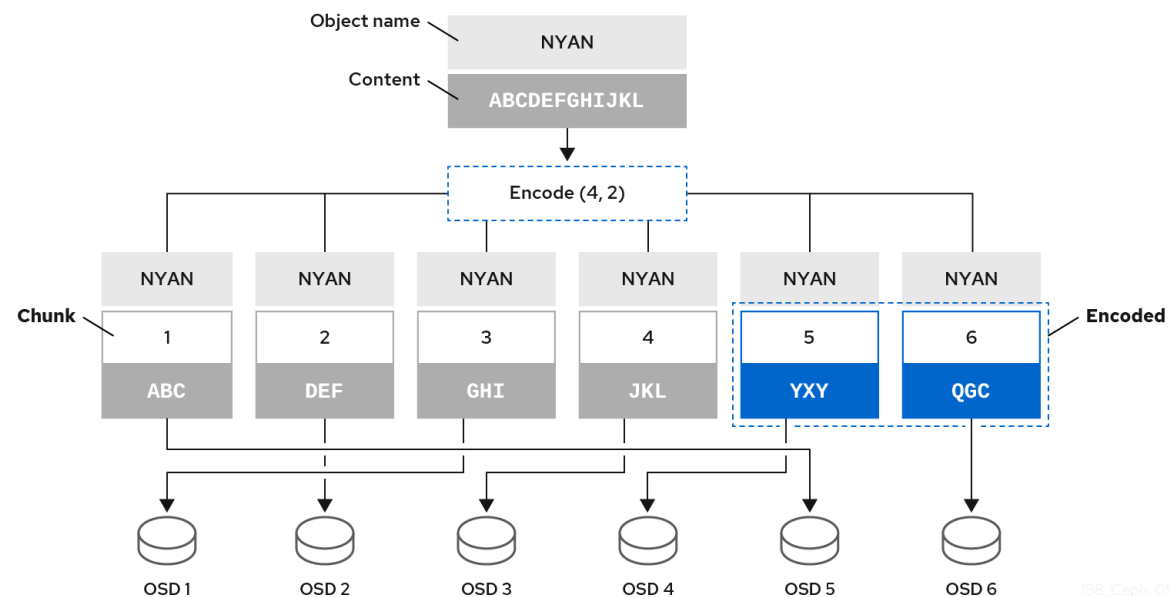
- Требуется в R раз больший объем ресурсов для хранения данных
- Необходимость обеспечения согласованности

Родственные подходы

RAID (Redundant Array of Independent Disks)



Erasure Coding (Стирающие коды)



158_Ceph_0521

Характер данных и операции

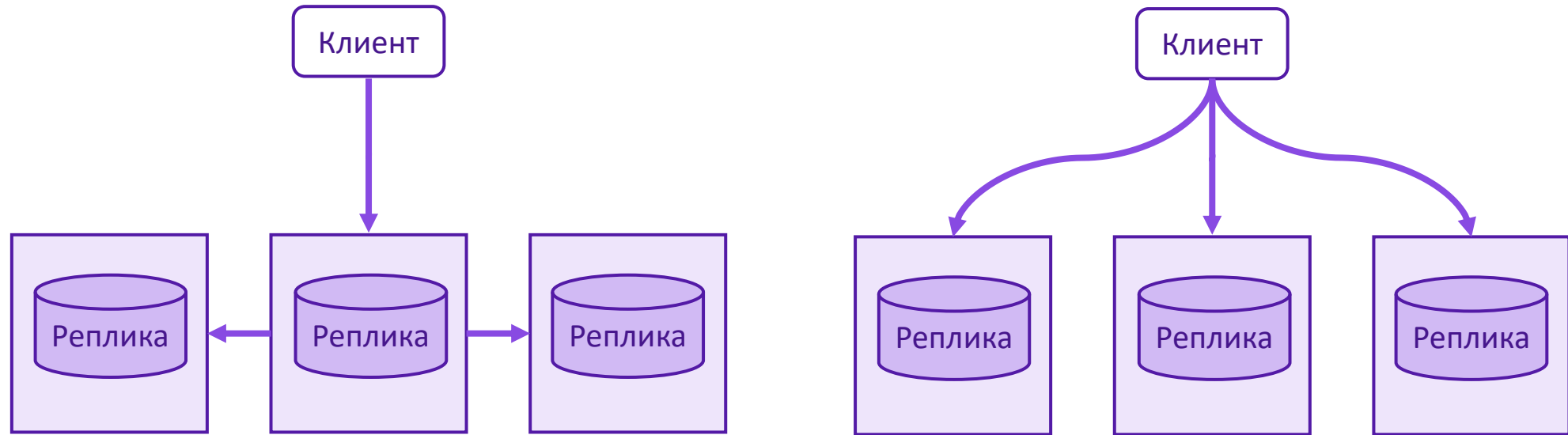
- Неизменяемые данные
 - После записи данные не модифицируются (write once, read many)
 - Пример: хранение логов
 - Репликация реализуется легко
- Изменяемые данные
 - Операции write и read к любым элементам данных в любое время (в т.ч. одновременно)
 - Примеры: база данных, key-value хранилище
 - Проблемы: синхронизация реплик, обеспечение согласованности

Согласованность (consistency)

- **Модель согласованности** определяет гарантии, которые система предоставляет клиентам на операции с хранимыми данными
 - Пример: каждое чтение возвращает последнее записанное значение
- **Сильная согласованность:** с точки зрения клиентов работа с системой не отличима от работы с единственной копией данных
 - Не наблюдаются "аномалии", нарушающие привычные гарантии

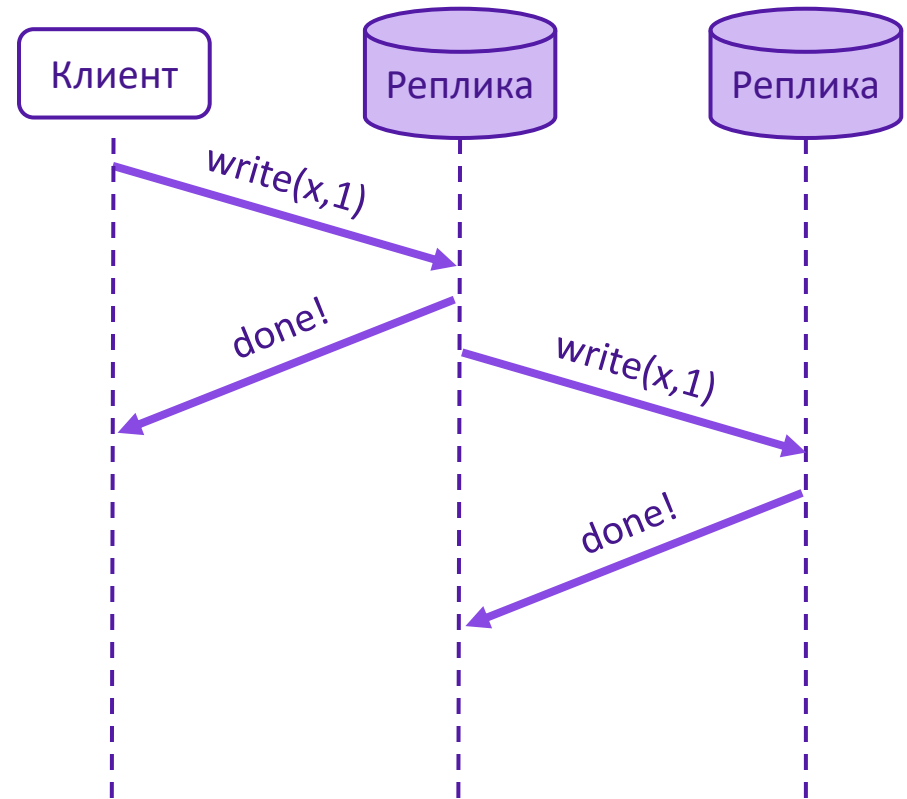
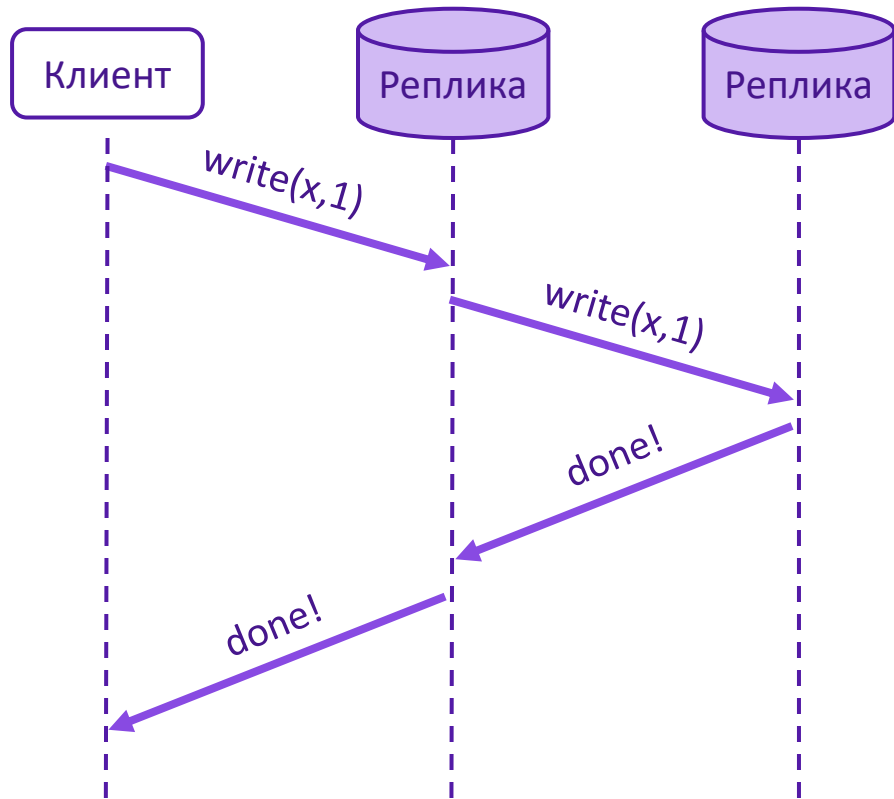


Пассивная и активная репликация



- **Пассивная репликация:** один выделенный сервер, обрабатывающий запросы клиентов и передающий изменения остальным
- **Активная репликация:** серверы равноправны, запросы клиентов обрабатываются каждым сервером

Синхронная и асинхронная репликация

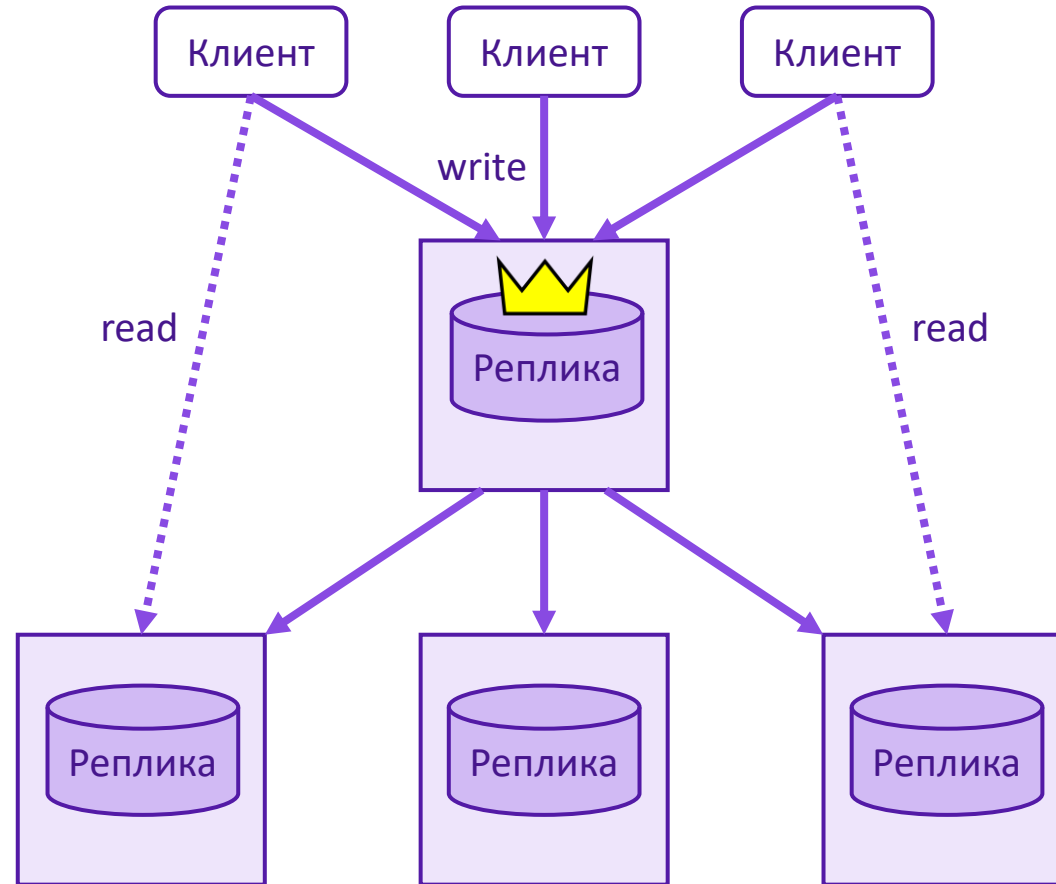


- Какие здесь возникают компромиссы?
- Существует ли промежуточная стратегия?

Основные подходы к репликации

- **Репликация с одним лидером** (active/passive, master-slave)
 - Запись данных ведется только через один узел
 - PostgreSQL, MySQL, Oracle, MongoDB, HBase, Kafka
- **Репликация с несколькими лидерами** (active/active, multi-master)
 - Клиент производит запись через одного из нескольких лидеров
 - WANdisco, CouchDB, Google Docs
- **Репликация без лидеров** (leaderless, quorum)
 - Клиент производит чтение и запись, взаимодействуя с несколькими узлами
 - Dynamo, Riak, Cassandra, Voldemort

Репликация с одним лидером



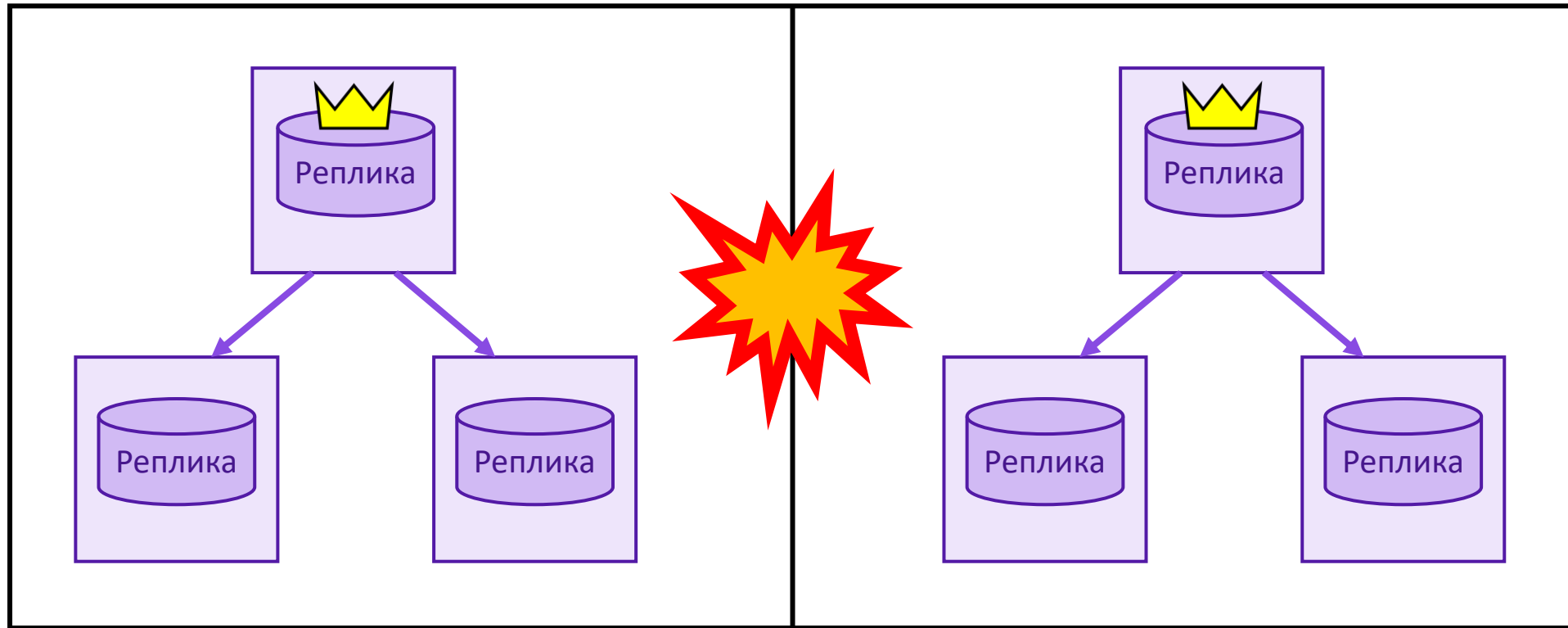
Репликация с одним лидером

- Пассивная
- Синхронная или асинхронная
 - Компромиссы (задержка, согласованность, надежность, доступность)
 - Полусинхронная репликация
- Что передается между лидером и подчиненными?
 - Операции, запросы (statement-based replication)
 - Результаты обработки операций (log shipping, row-based replication)
- Подключение новой реплики
 - Копирование данных с лидера на некоторый момент времени (snapshot)
 - Применение последующих изменений (catch up)

Репликация с одним лидером

- Обработка отказа подчиненного узла (catch-up recovery)
 - Перезапуск и получение актуального состояния
- Обработка отказа лидера (failover)
 - Выбор нового лидера и реконфигурация системы
 - Ручная или автоматическая процедура
 - Перенаправление клиентов на нового лидера
 - Возвращение старого лидера, разделение сети (split brain)

Разделение сети (split brain)

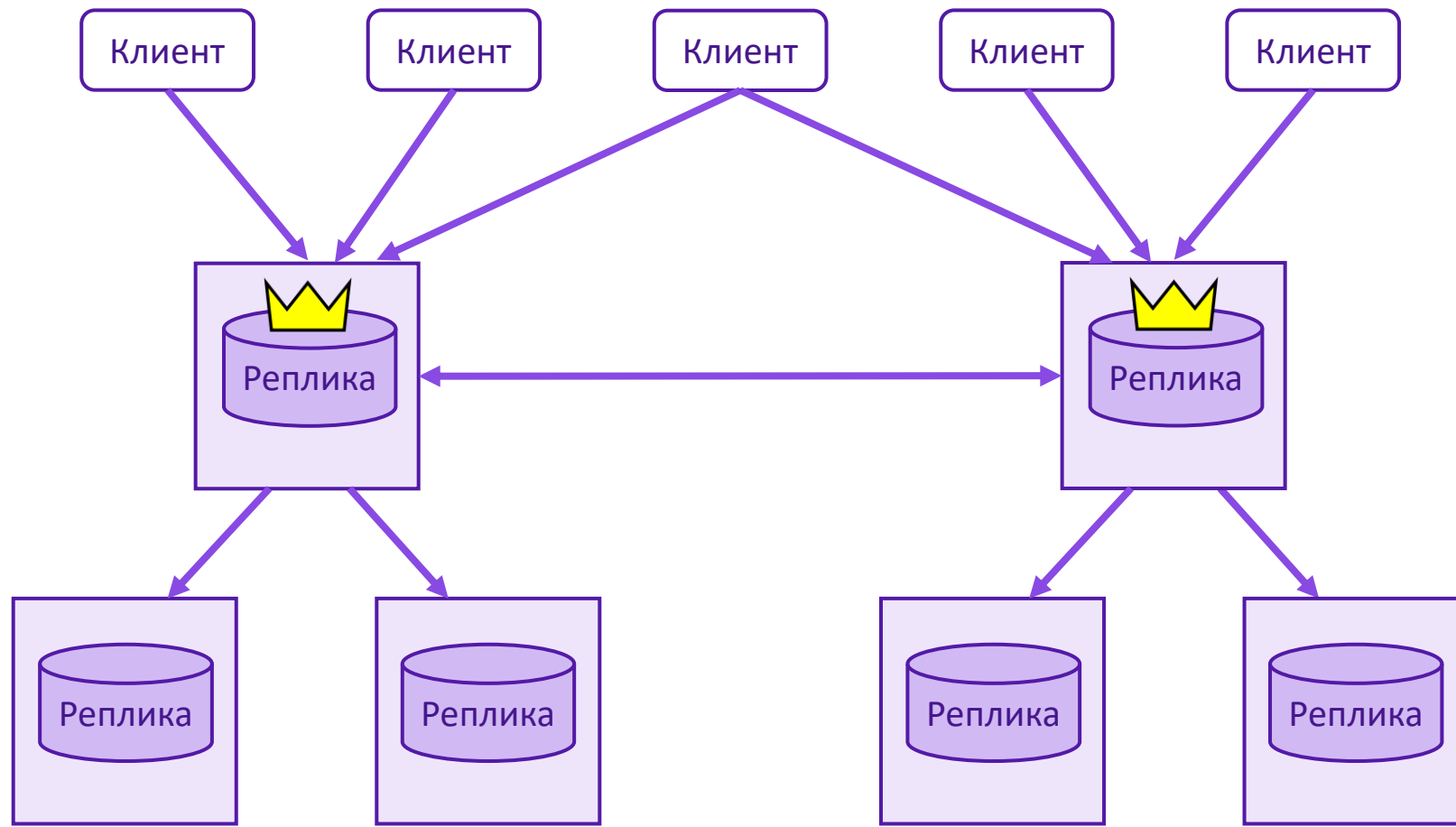


[The Network is Reliable: An informal survey of real-world communications failures](https://github.com/aphyr/partitions-post) (2014)
<https://github.com/aphyr/partitions-post>

Репликация с одним лидером

- Преимущества
 - Отсутствие конфликтов, т.к. лидер упорядочивает операции записи
 - Простота реализации
- Недостатки
 - Ограниченная масштабируемость (в т.ч. гео-)
 - Необходимость обработки отказов (выборов) лидера

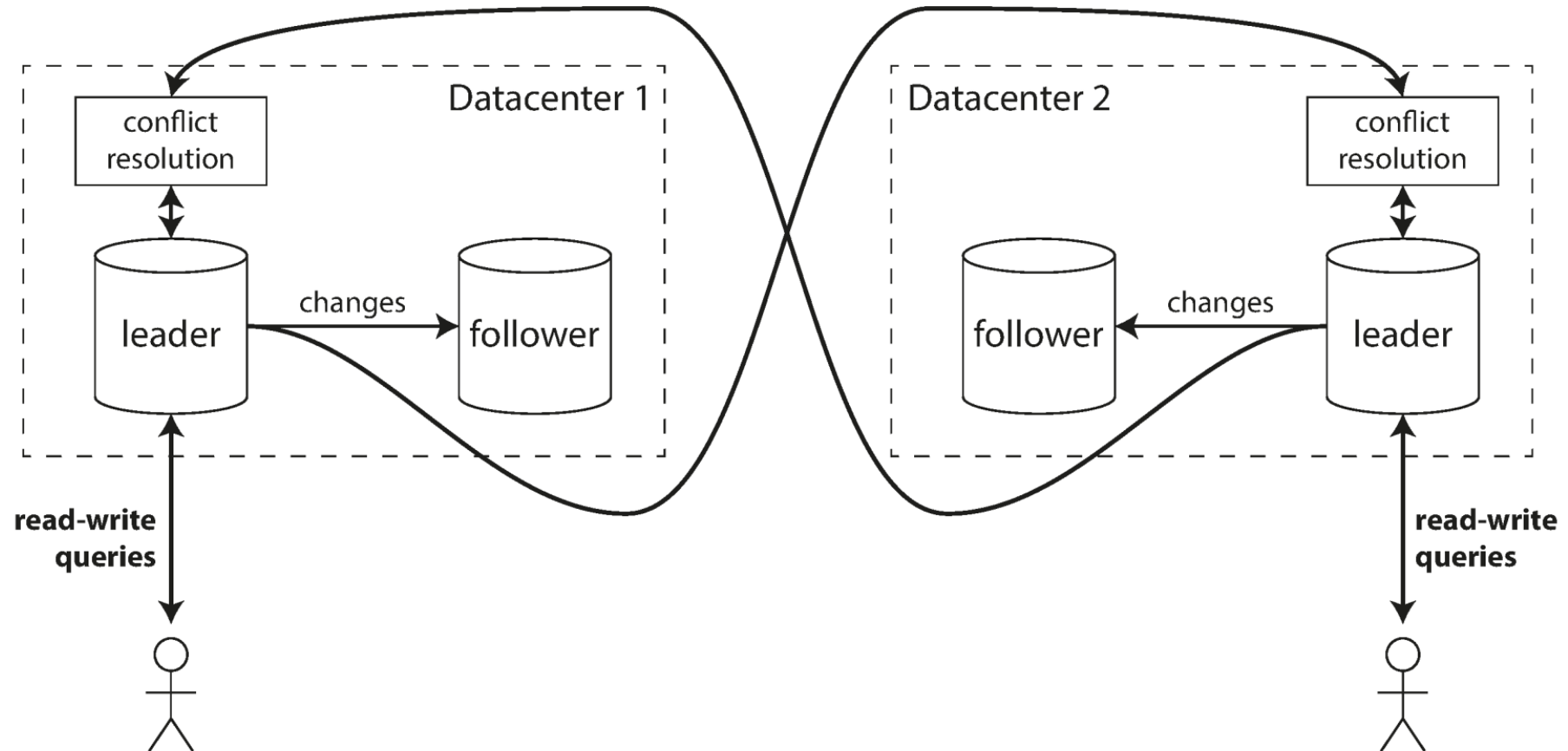
Репликация с несколькими лидерами



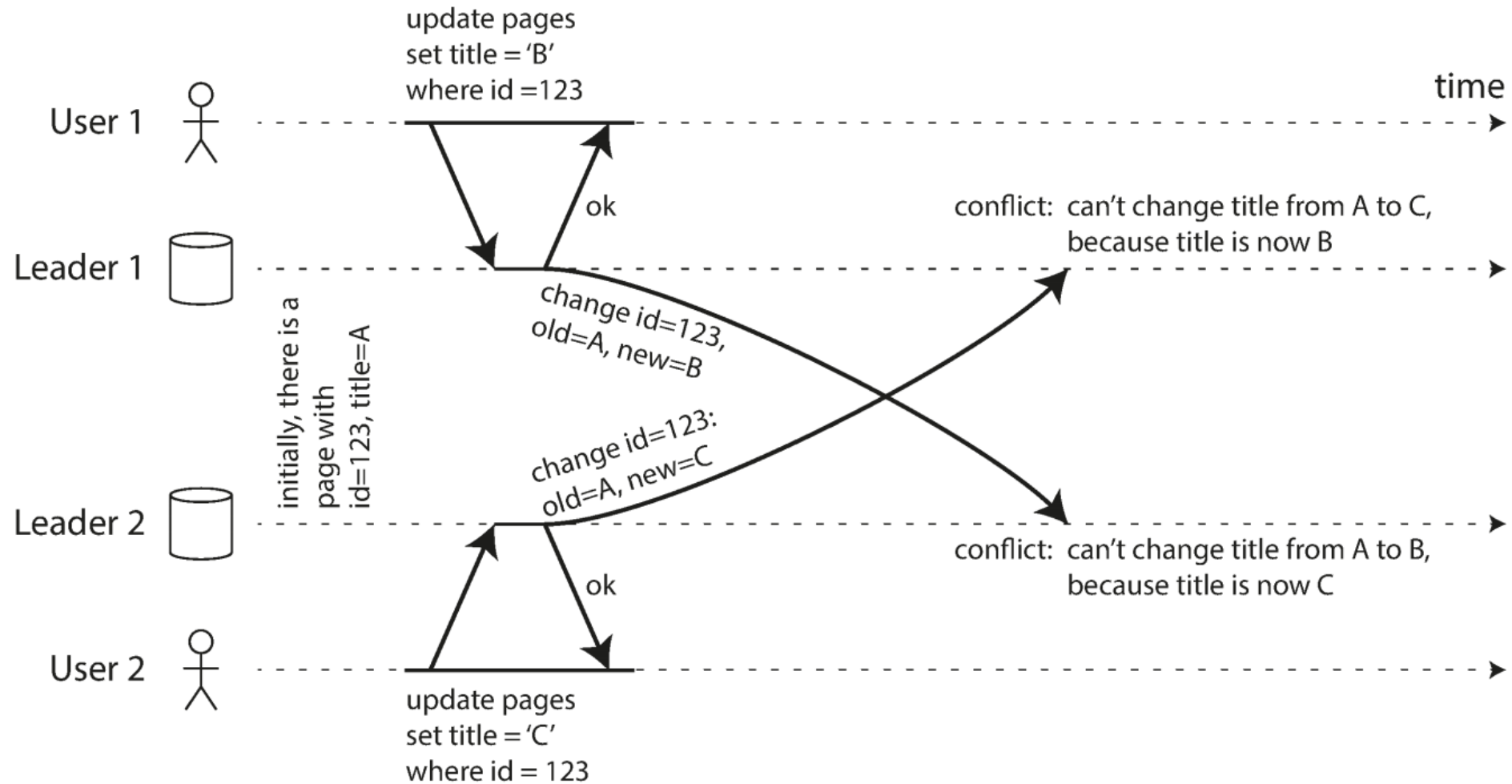
Репликация с несколькими лидерами

- Операции записи обрабатывают несколько узлов-лидеров
 - Клиент взаимодействует с одним лидером
 - Лидер также играет роль подчиненного относительно других лидеров
- Мотивация
 - Репликация данных между датацентрами (задержка, доступность, WANdisco)
 - Функционирование в офлайн-режиме (календарь, Dropbox, CouchDB)
 - Онлайн-сервисы совместного редактирования (Google Docs)

Асинхронный режим



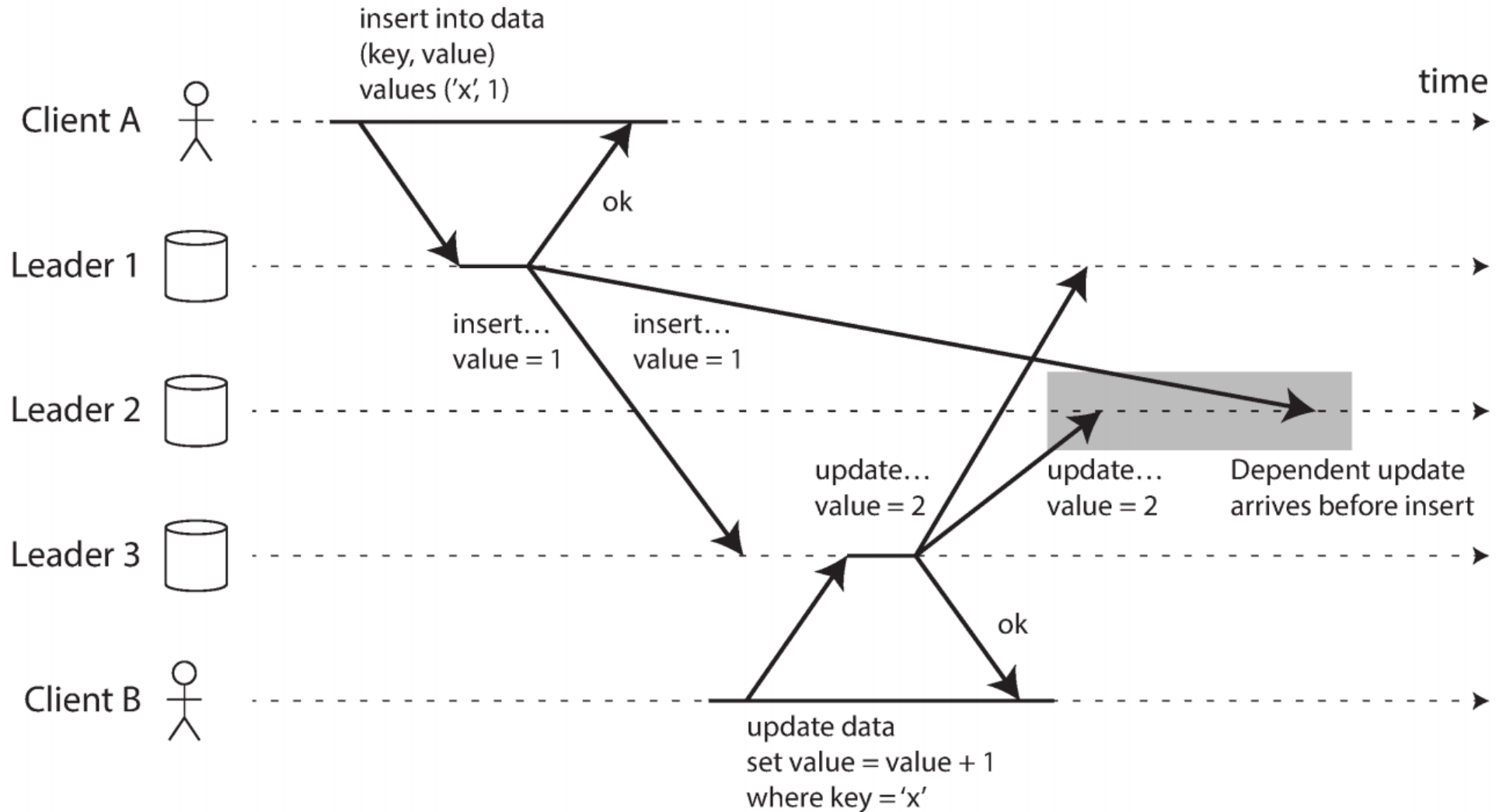
Конфликт при записи данных



Разрешение конфликтов

- Требуется обеспечить свойство сходимости (convergence)
 - Все реплики должны прийти к общему конечному значению после того, как все изменения достигнут всех узлов
- Возможные подходы
 - Операция с наибольшим ID "выигрывает" (last write wins)
 - Слияние конфликтующих значений ("В/С")
 - Специфическая процедура на уровне приложения, в т.ч. с участием пользователя
 - Conflict-free replicated data types (CRDT)
- Когда происходит разрешение конфликта?
 - Во время записи или чтения

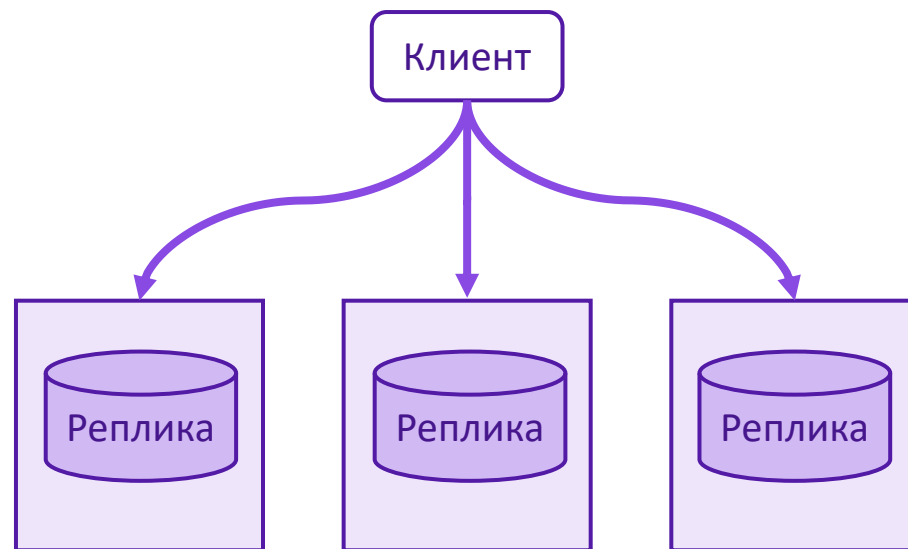
Нарушение порядка операций записи



Репликация с несколькими лидерами

- Пассивная
- В основном асинхронная
- Преимущества
 - Распределение нагрузки между несколькими лидерами
 - Снижение задержки для географически распределенных клиентов
 - Поддержка офлайн-клиентов
- Недостатки
 - Требуется координация между лидерами (синхронный режим)
 - Возможны конфликты при записи (асинхронный режим)
 - Возможно нарушение порядка операций записи

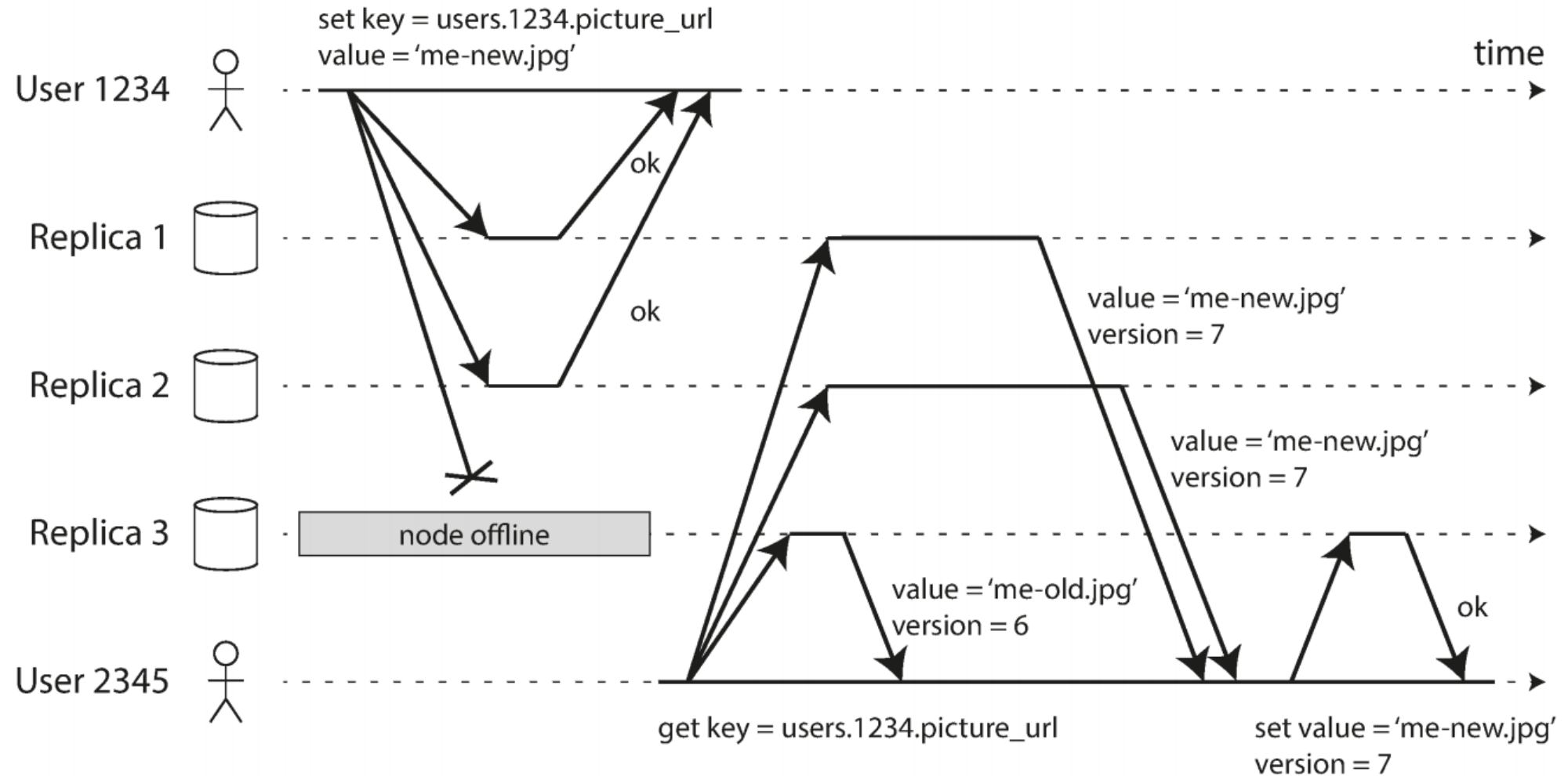
Репликация без лидера



Репликация без лидера

- Клиент взаимодействует не с одним узлом (лидером), а с несколькими
 - Узлы не копируют активно данные между друг другом
 - Клиент сам отвечает за копирование данных
- Операции чтения и записи требуют взаимодействия клиента с **R** и **W** узлами соответственно (запись и чтение с кворумом)
 - [Weighted Voting for Replicated Data](#) (1979)
- Примеры: Dynamo, Riak, Cassandra, Voldemort

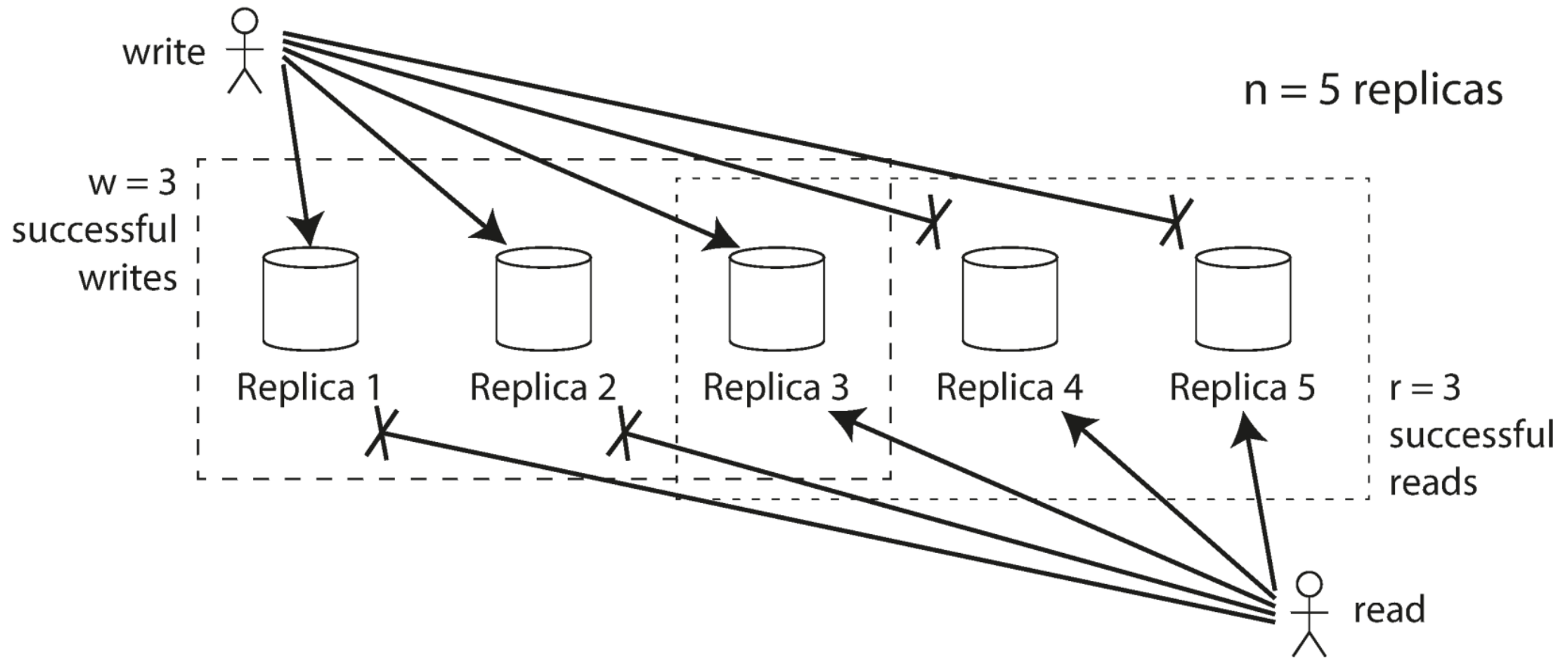
Запись и чтение с кворумом



Размеры кворумов

- Для чтения последних записанных данных необходимо, чтобы $W + R > N$
- Во многих системах параметры W, R, N можно настраивать
 - N — нечетное
 - $W = R = (N + 1)/2$ — сбалансированная конфигурация
 - $W = N, R = 1$ — максимальная оптимизация под чтение
- Устойчивость к отказам
 - $W < N, R < N$
 - $N = 3, W = 2, R = 2$ — допустим отказ 1 узла
 - $N = 4, W = 3, R = 2$ — допустим отказ 1 узла
 - $N = 5, W = 3, R = 3$ — допустимы отказы 2 узлов

Пример: $N=5$ $W=3$ $R=3$



Дополнительные механизмы

- Read repair
 - Обновление устаревших данных и разрешение конфликтов во время чтения
- Anti-entropy
 - Фоновый процесс, постоянно осуществляющий обмен информацией и сравнение хранимых данных между репликами
- Sloppy quorum, hinted handoff
 - Временный перенос функций реплик с отказавших на другие узлы
- См. [Dynamo: Amazon's Highly Available Key-value Store](#) (2007)

Репликация без лидера

- Активная
- Полусинхронная
- Преимущества
 - Упрощается обработка отказов (не надо выбирать лидера)
 - Потенциально высокая доступность
 - Проще архитектура?
- Недостатки
 - Сложнее обеспечивать согласованность данных
 - Возможно возникновение конфликтов
 - Требуются дополнительные механизмы для восстановления согласованности и разрешения конфликтов

Репликация и рассылка

- Можно ли использовать известные нам варианты рассылки для репликации?
- Какие требования должны предъявляться к рассылаемым операциям?

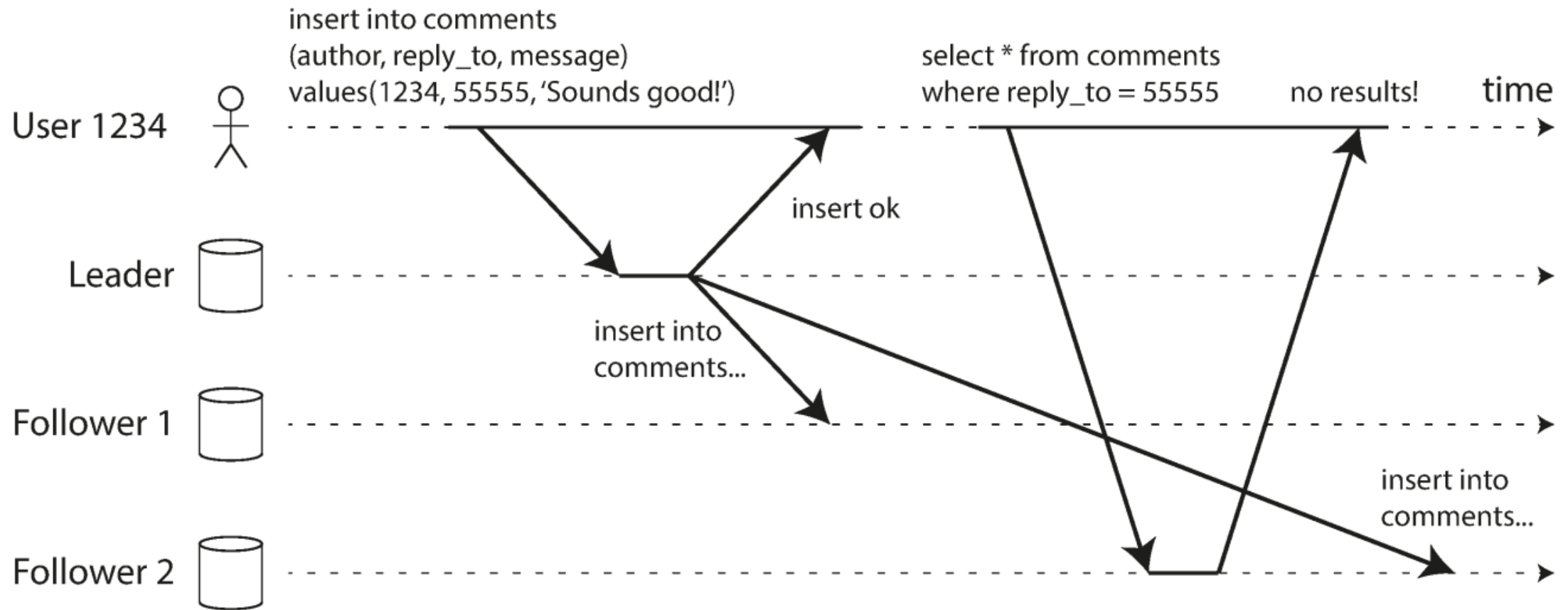
broadcast	assumptions about state update function
total order	deterministic (SMR)
causal	deterministic, concurrent updates commute
reliable	deterministic, all updates commute
best-effort	deterministic, commutative, idempotent, tolerates message loss

Согласованность (consistency)

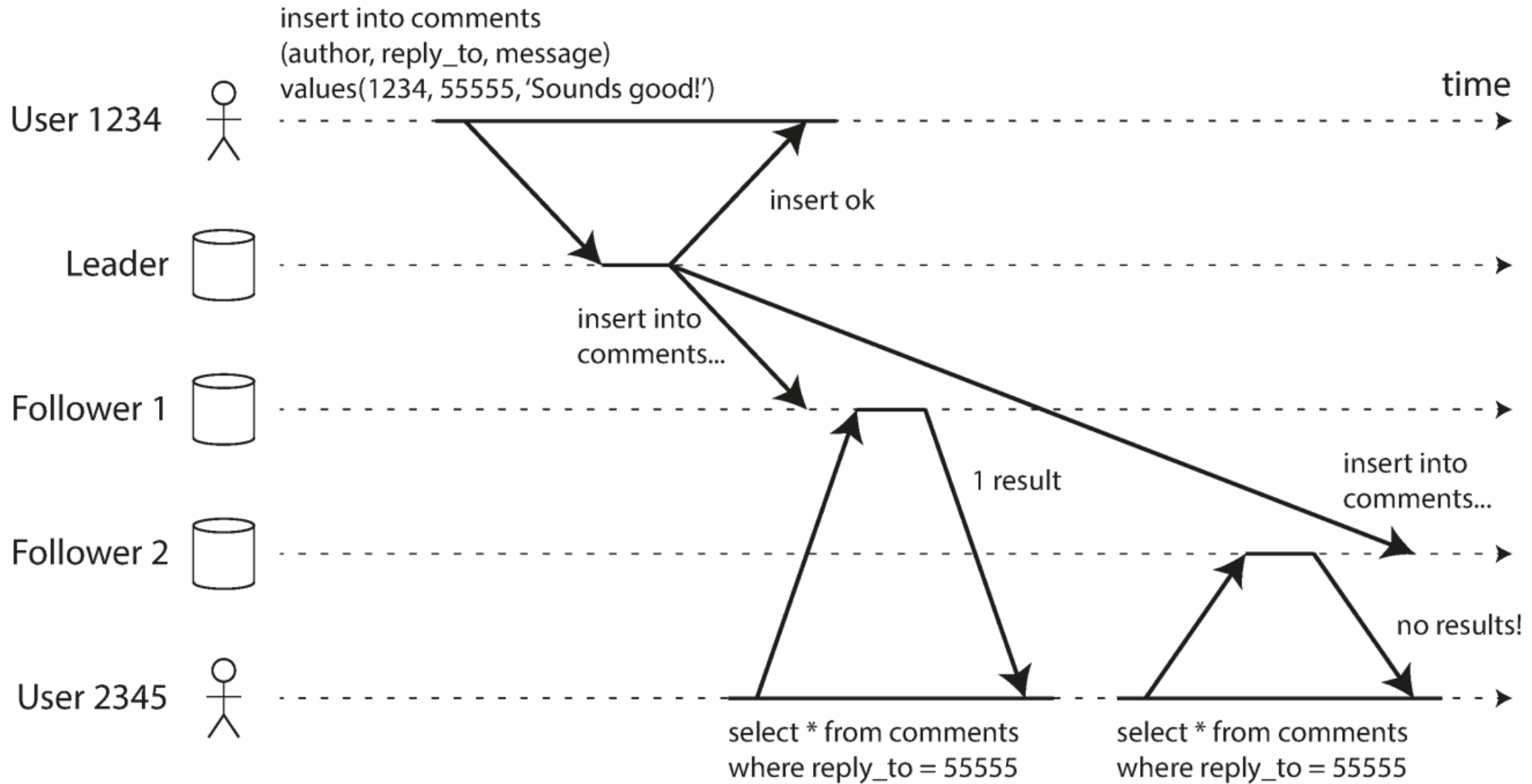
- Какие **гарантии** требуются от системы с несколькими репликами данных и одновременно работающими клиентами?
 - В любой момент времени все реплики хранят одинаковое значение
 - Каждое чтение возвращает последнее записанное значение
 - Некоторые гарантии на порядок и видимость результатов операций
- **Модель согласованности** определяется набором предоставляемых гарантий
 - Клиенты системы не должны наблюдать "аномалии", в которых эти гарантии нарушаются

[The many faces of consistency](#) (2016)

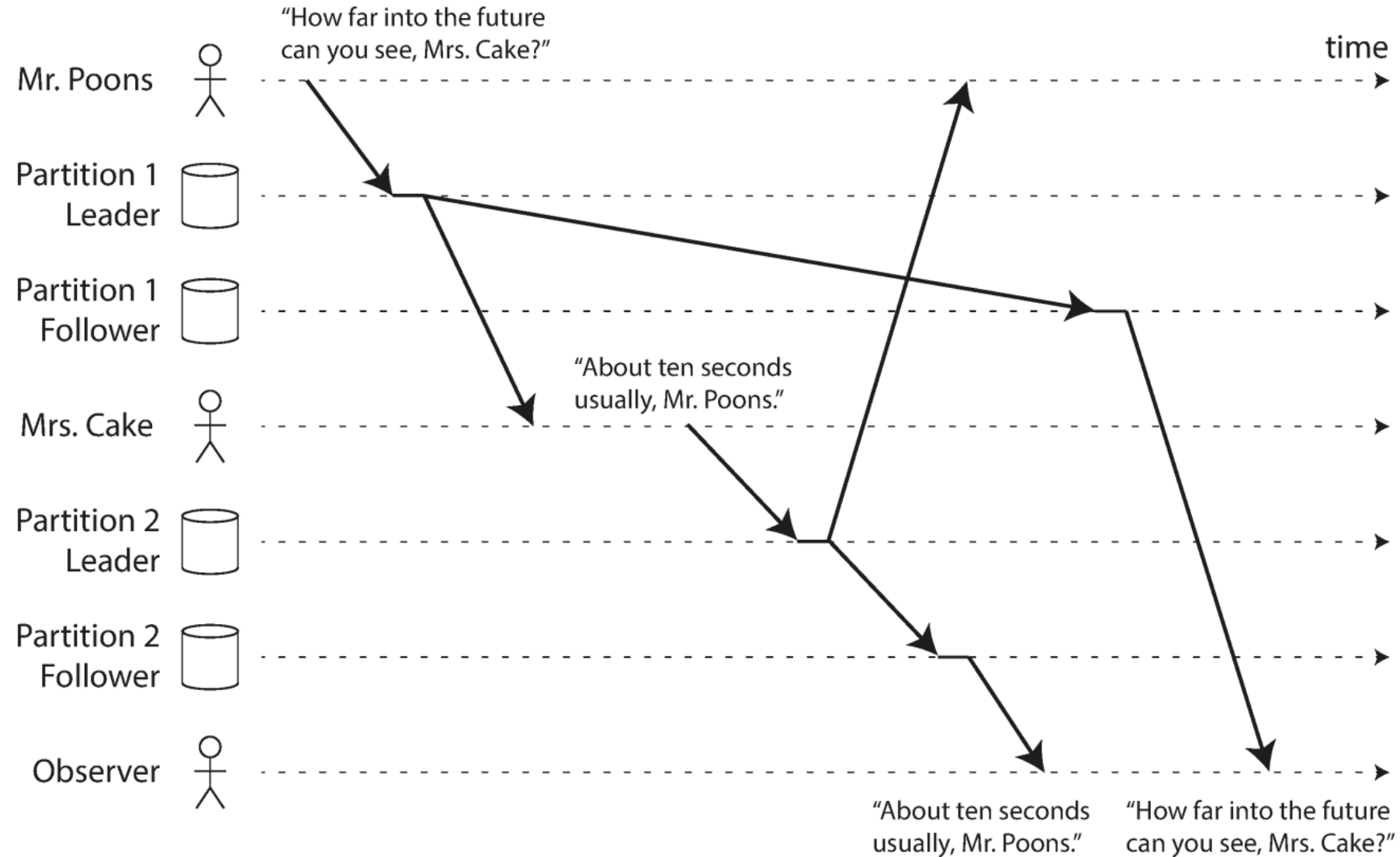
Аномалия 1



Аномалия 2



Аномалия 3



Модель согласованности

- Результаты каких операций записи "видны" клиенту?
- Всех предыдущих операций в соответствии с некоторым единым порядком
 - Строгая согласованность, линеаризуемость, последовательная согласованность
- Любого подмножества операций в произвольном порядке
 - Согласованность в конечном счёте
- Промежуточные модели между этими двумя крайностями
 - Компромисс между строгостью гарантий, производительностью и доступностью

Строгая согласованность

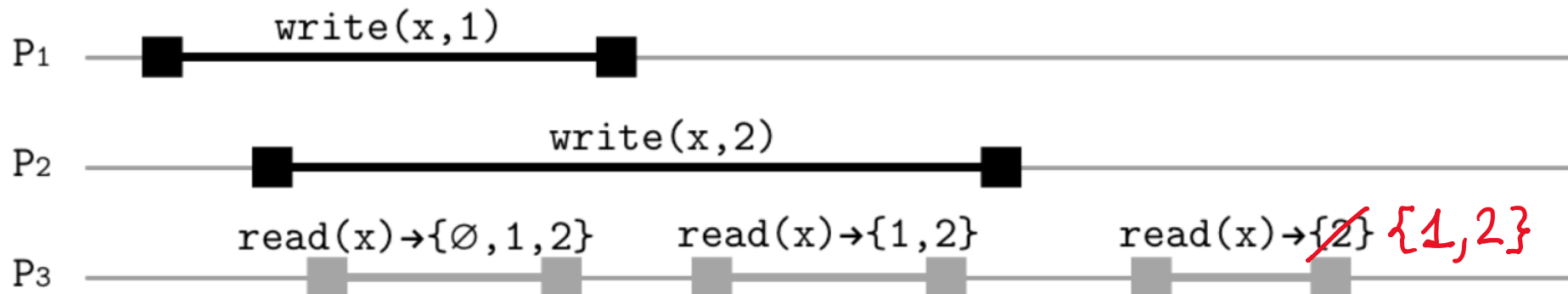
- Чтение всегда возвращает результат самой последней записи
- Результат записи становится мгновенно доступен всем клиентам
- Не реализуема на практике в силу законов физики

Линеаризуемость

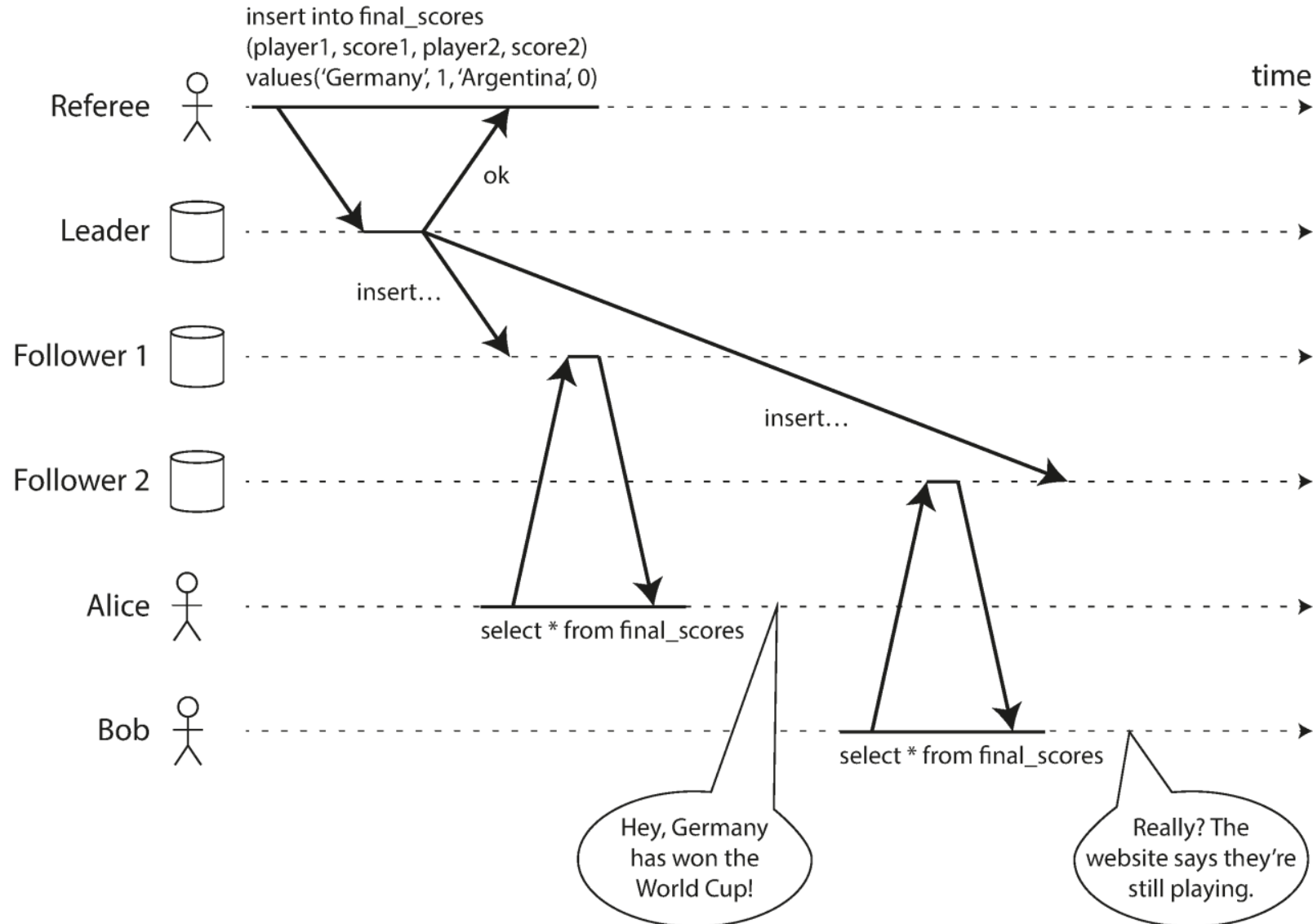
- [Linearizability: A correctness condition for concurrent objects](#) (1990)
- Наиболее строгая модель из реализуемых на практике
- Любое выполнение системы может быть представлено в виде некоторой упорядоченной истории операций, такой что
 - она эквивалентна корректному последовательному выполнению операций над одной копией данных (чтение возвращает последнее записанное значение)
 - порядок операций согласуется с временами выполнения операций

Линеаризуемость (менее формально)

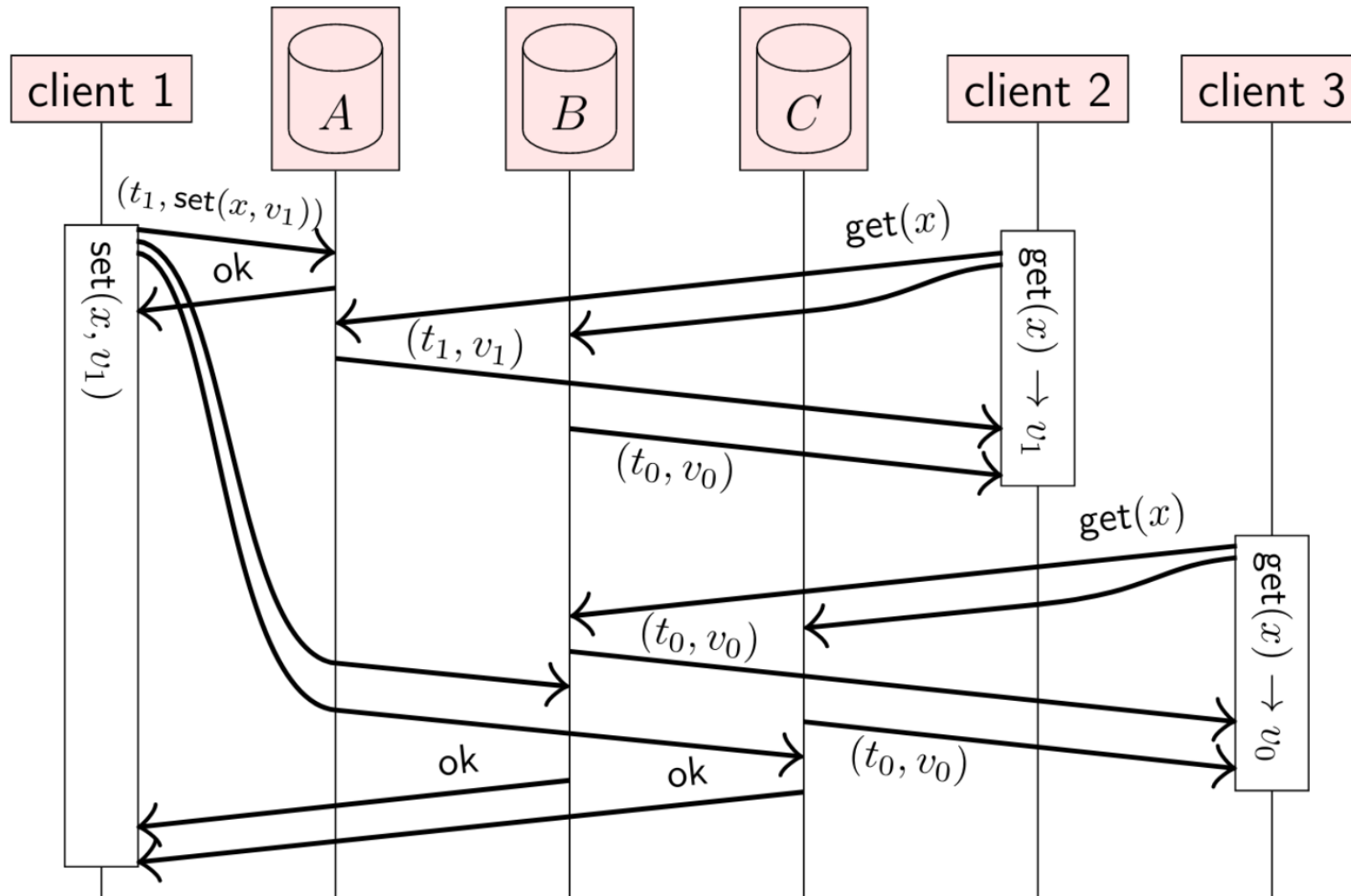
- С точки зрения клиентов система работает так, как будто
 - есть только одна копия данных
 - операции атомарны (операция выполняется мгновенно в некоторый момент времени между ее вызовом и завершением)
- Все клиенты видят результаты операций записи в некотором едином (глобальном) порядке
- Чтение видит результат последней выполненной операции записи



Нарушение линейризуемости (лидер)

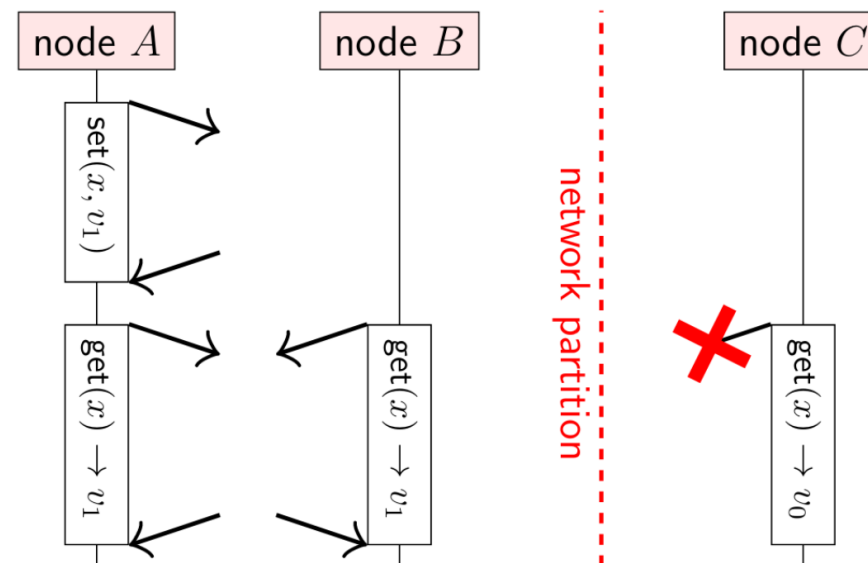


Нарушение линейризуемости (кворум)



Линеаризуемость

- Преимущества
 - Система ведет себя как нераспределенная (одна копия данных)
 - Упрощает написание клиентских приложений
- Недостатки
 - Снижение производительности (много сообщений, ожидание ответов)
 - Ограниченная масштабируемость (лидер - узкое место)
 - Проблемы с доступностью (требуется доступность большинства машин)



[A Critique of the CAP Theorem](#) (2015)

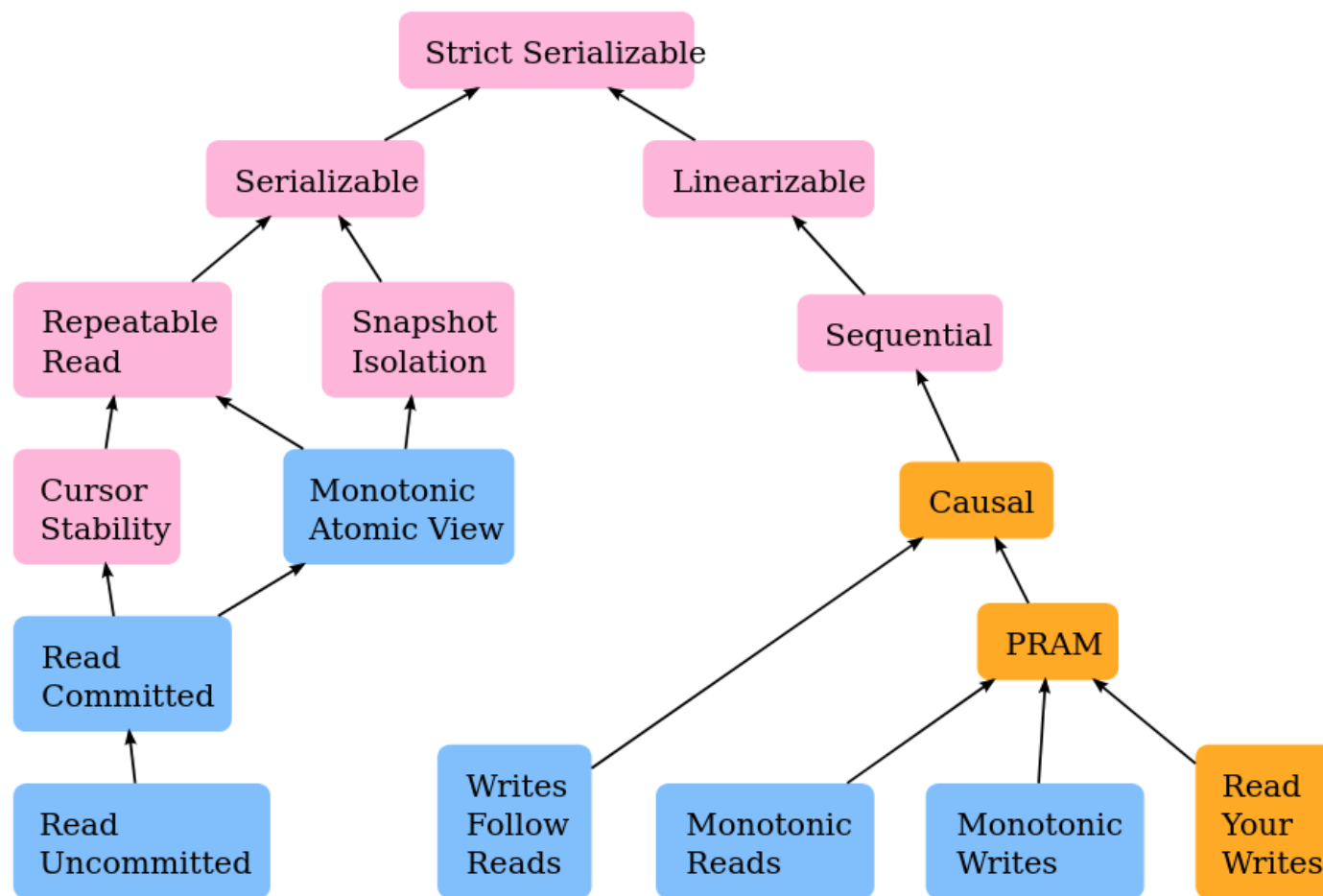
Согласованность в конечном счёте

- [Eventually Consistent](#) (2009)
- "if no new updates are made to an object, eventually all reads will return the last updated value"
- Наиболее слабая из используемых на практике моделей (см. DNS)
- Реплики могут обрабатывать запросы на основе локального состояния
- Ничего не говорит о времени сходимости реплик
- **Strong eventual consistency** гарантирует сходимость реплик
 - каждое изменение рано или поздно достигнет все реплики
 - две реплики, применившие одинаковый набор изменений, придут в одинаковое состояние

Согласованность в конечном счёте

- Преимущества
 - См. недостатки линеаризуемости
 - Лучше производительность, масштабируемость и доступность
- Недостатки
 - Отсутствие привычных гарантий
 - Необходимость разрешения конфликтов
 - Усложнение написания клиентских приложений

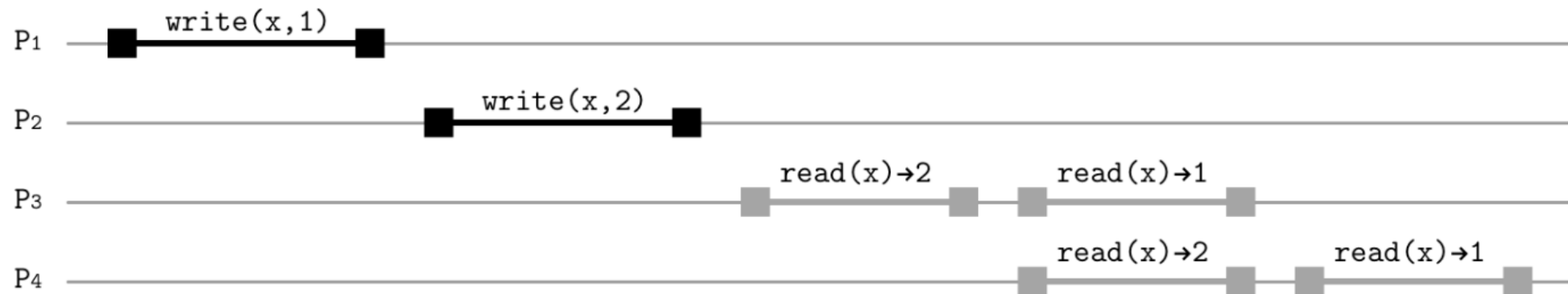
Другие модели согласованности



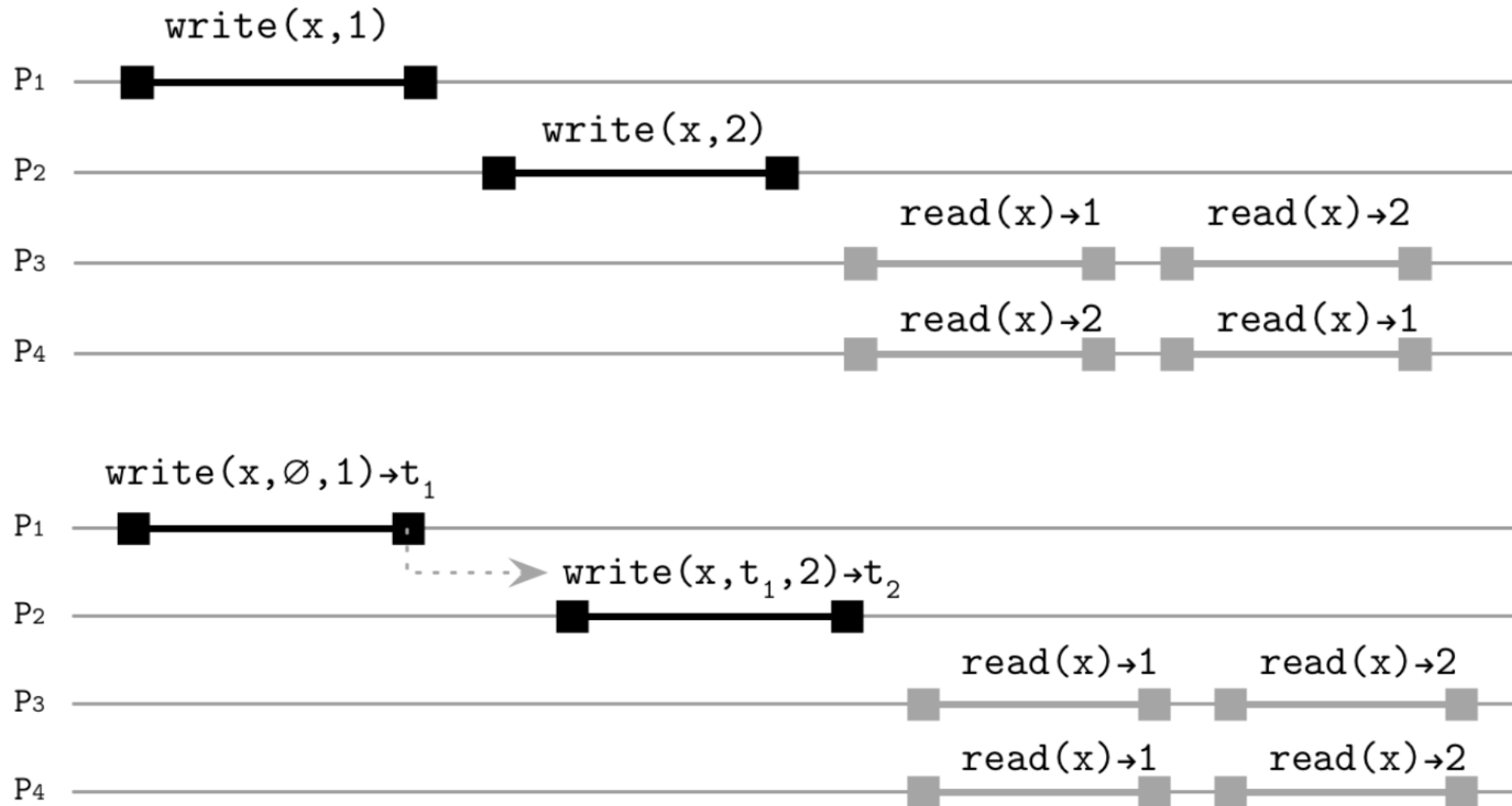
<http://jepsen.io/consistency>

Последовательная согласованность

- [How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs](#) (1979)
- Любое выполнение системы может быть представлено в виде некоторой упорядоченной истории операций, такой что
 - она эквивалентна корректному последовательному выполнению операций над одной копией данных
 - порядок операций согласуется с **порядком выполнения операций** в каждом из процессов



Причинная согласованность



Другие модели и гарантии

- Consistent Prefix
 - некоторая часть операций записи, без пропусков (префикс истории)
- Bounded Staleness
 - все записи, выполненные достаточно давно
- Monotonic Reads
 - увеличивающееся подмножество операций (гарантия на сессию)
- Read After Write (Read Your Writes)
 - все записи, произведенные клиентом (гарантия на сессию)

[Session Guarantees for Weakly Consistent Replicated Data](#) (1994)

Согласованность и задержка

Минимально возможная задержка операции при различных моделях согласованности в зависимости от сетевой задержки d :

Модель согласованности	Задержка при записи	Задержка при чтении
Линеаризуемость	$O(d)$	$O(d)$
Последовательная согласованность	$O(d)$ или $O(1)$	$O(1)$ или $O(d)$
Причинная согласованность (и более слабые модели)	$O(1)$	$O(1)$

См. [A Critique of the CAP Theorem](#) (2015)

Материалы

- [Designing Data-Intensive Applications](#) (глава 5)
- [Distributed Systems Course](#) (главы 5 и 7)
- [Database Internals](#) (глава 11)
- [A Primer on Database Replication](#)
- [Consistency Models](#)
- [A Critique of the CAP Theorem](#)
- [CockroachDB's Consistency Model](#)
- [Jepsen: MongoDB stale reads](#)
- [Jepsen Analyses](#)