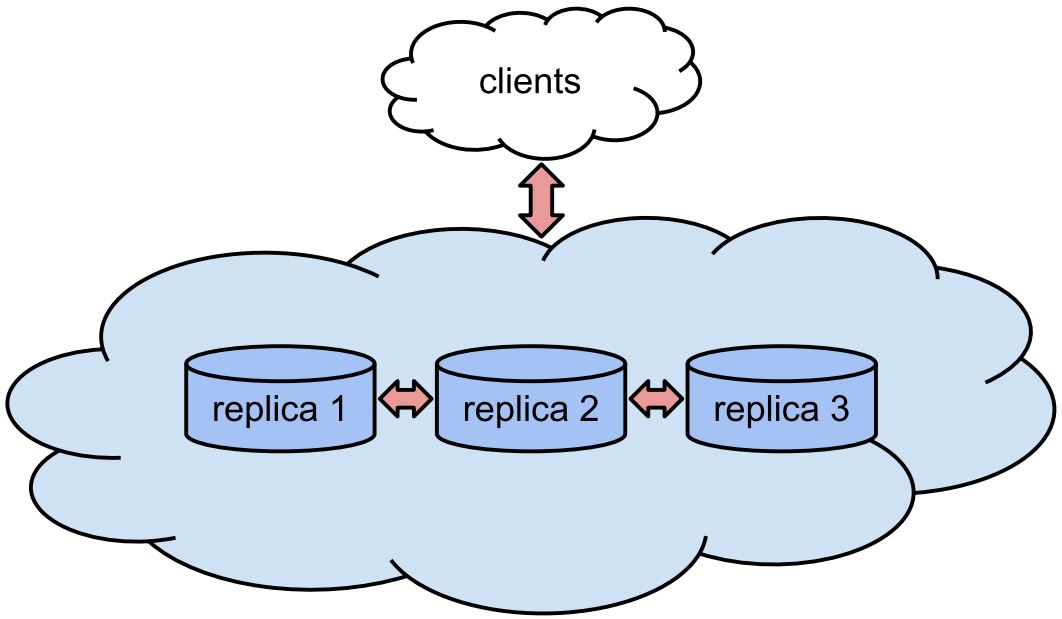
План

- Репликация данных
- Модели согласованности

Репликация данных

Хранение физических копий (реплик) данных на нескольких машинах (файлы, записи, конфигурация, состояние процессов)



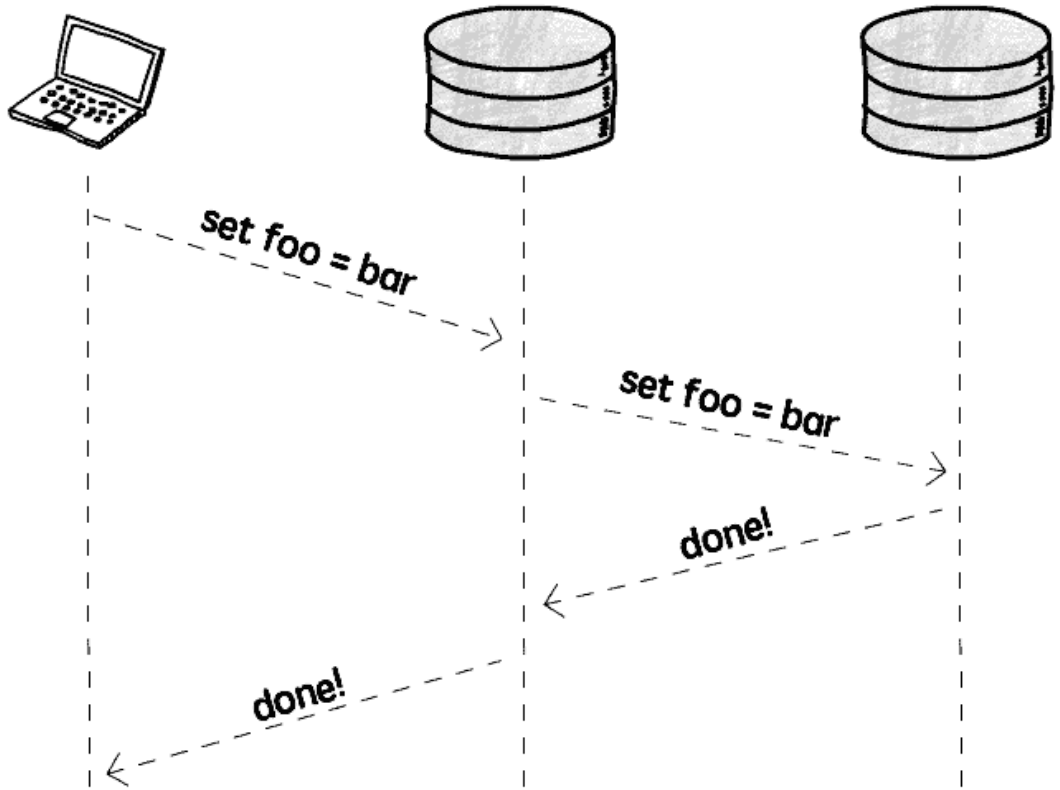
Мотивация

- Повышение доступности
- Уменьшение задержки
- Увеличение производительности

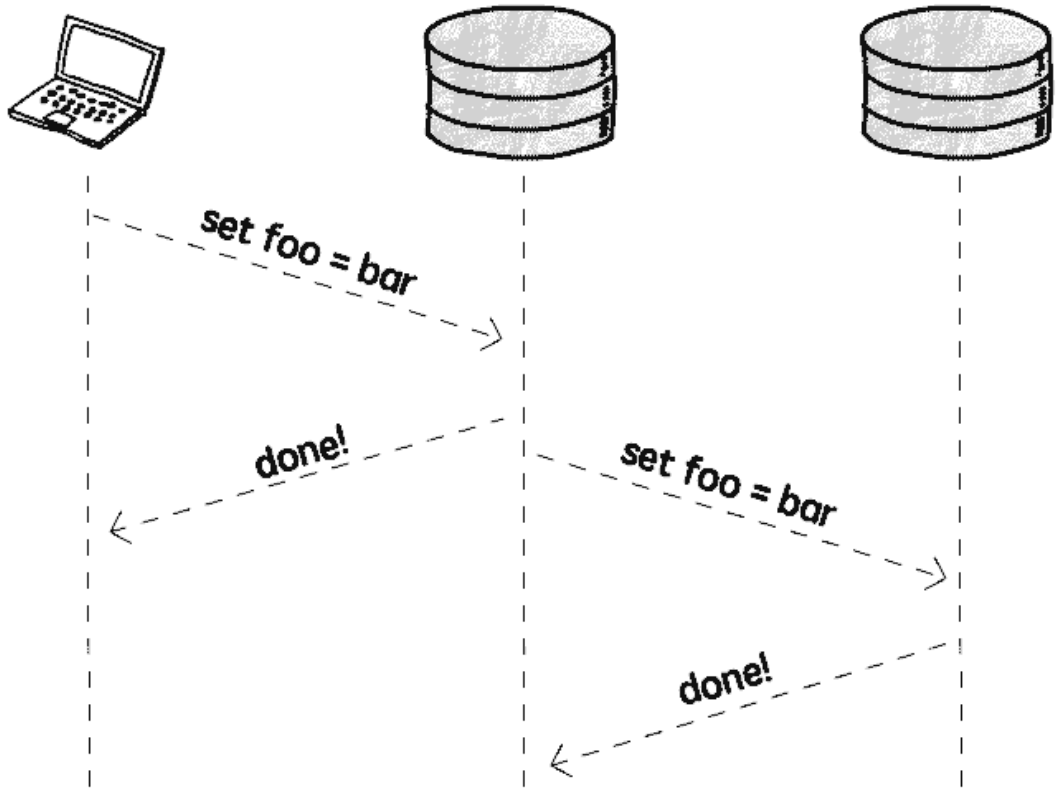
Характер данных и операции

- 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)
  - Пример: реляционная база данных
  - Проблемы - синхронизация реплик, обеспечение согласованности

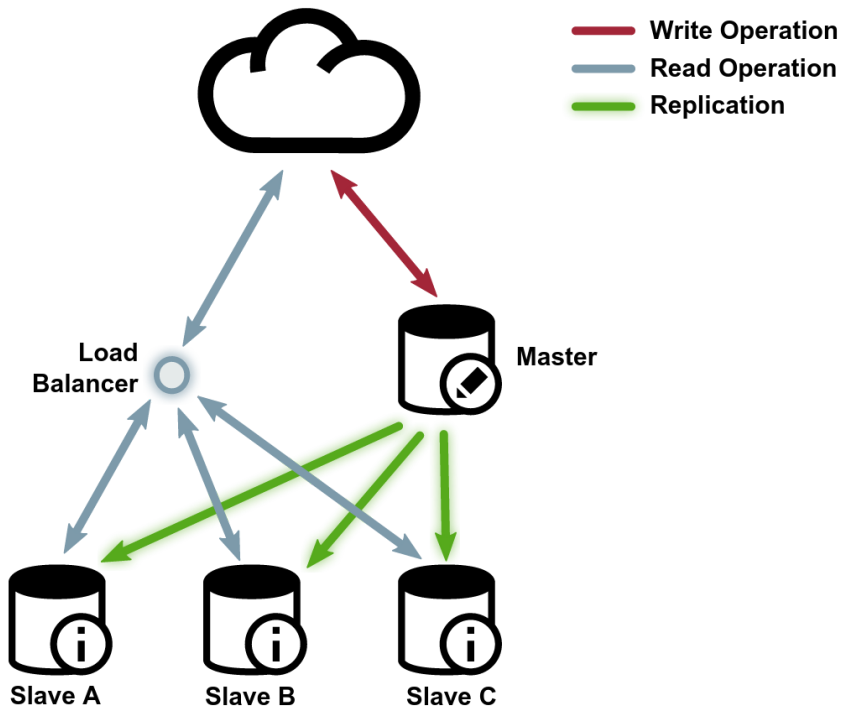
Синхронная репликация



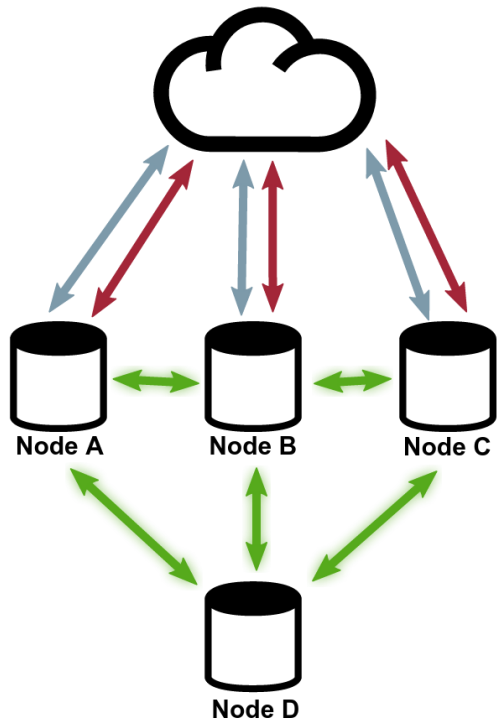
Асинхронная репликация



Возможные архитектуры



Master-Slave Replication

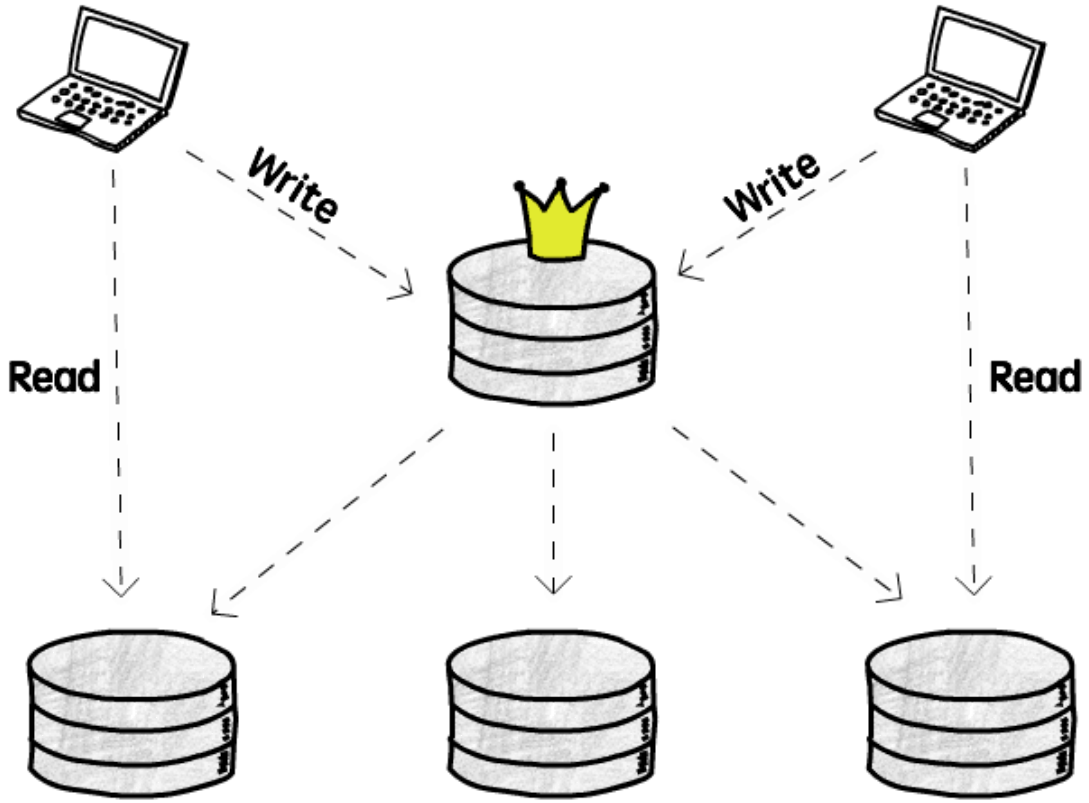


Multi-Master Replication

Существующие подходы

- 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

- Сихронная или асинхронная
  - Компромиссы (задержка, согласованность, надежность, доступность)
- 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)

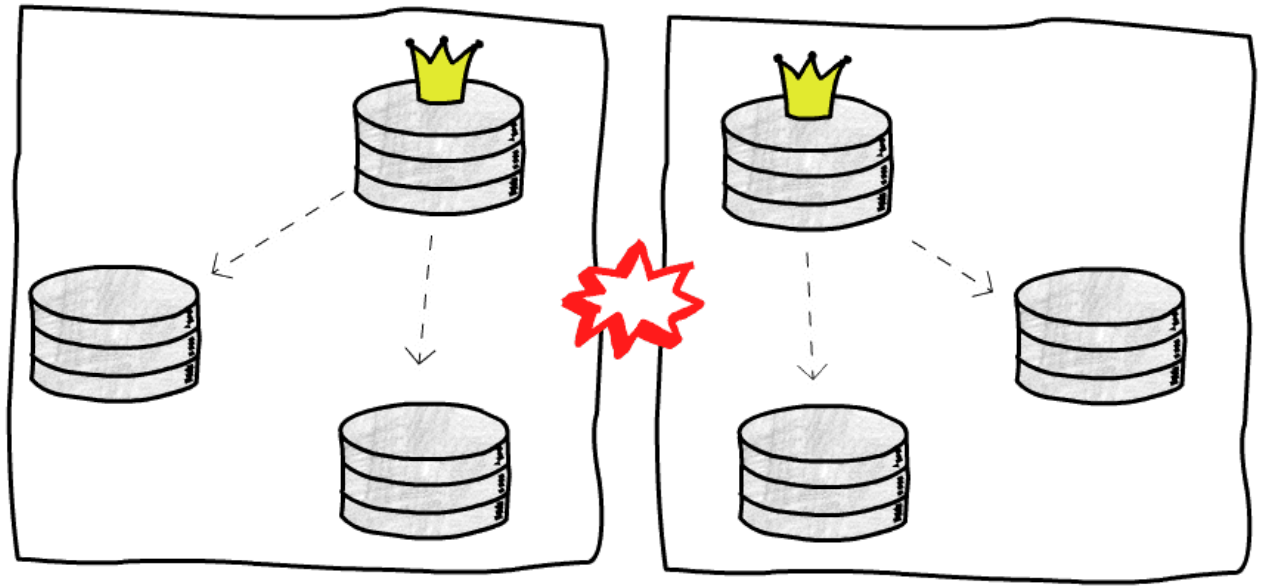
Репликация с одним лидером

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

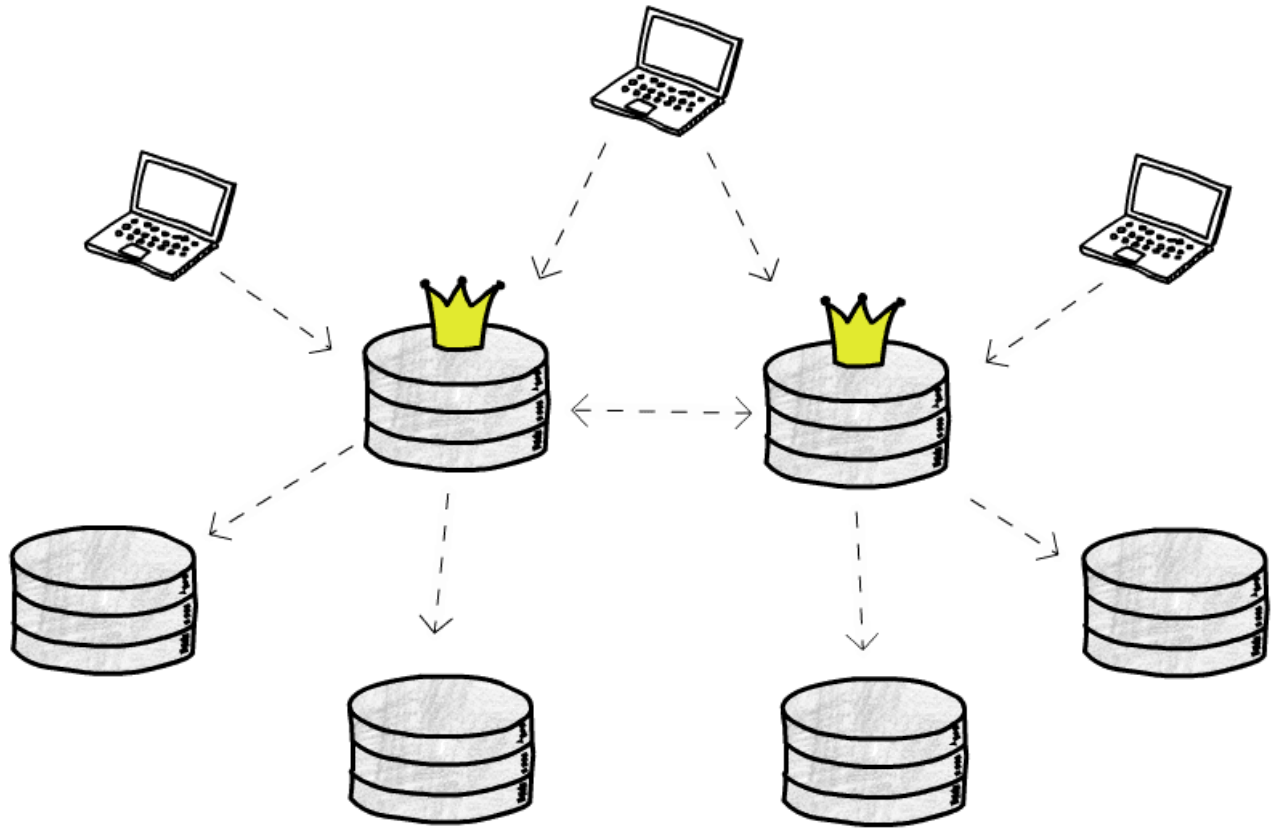
Репликация с одним лидером

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

Разделение сети (network partition)



Репликация с несколькими лидерами



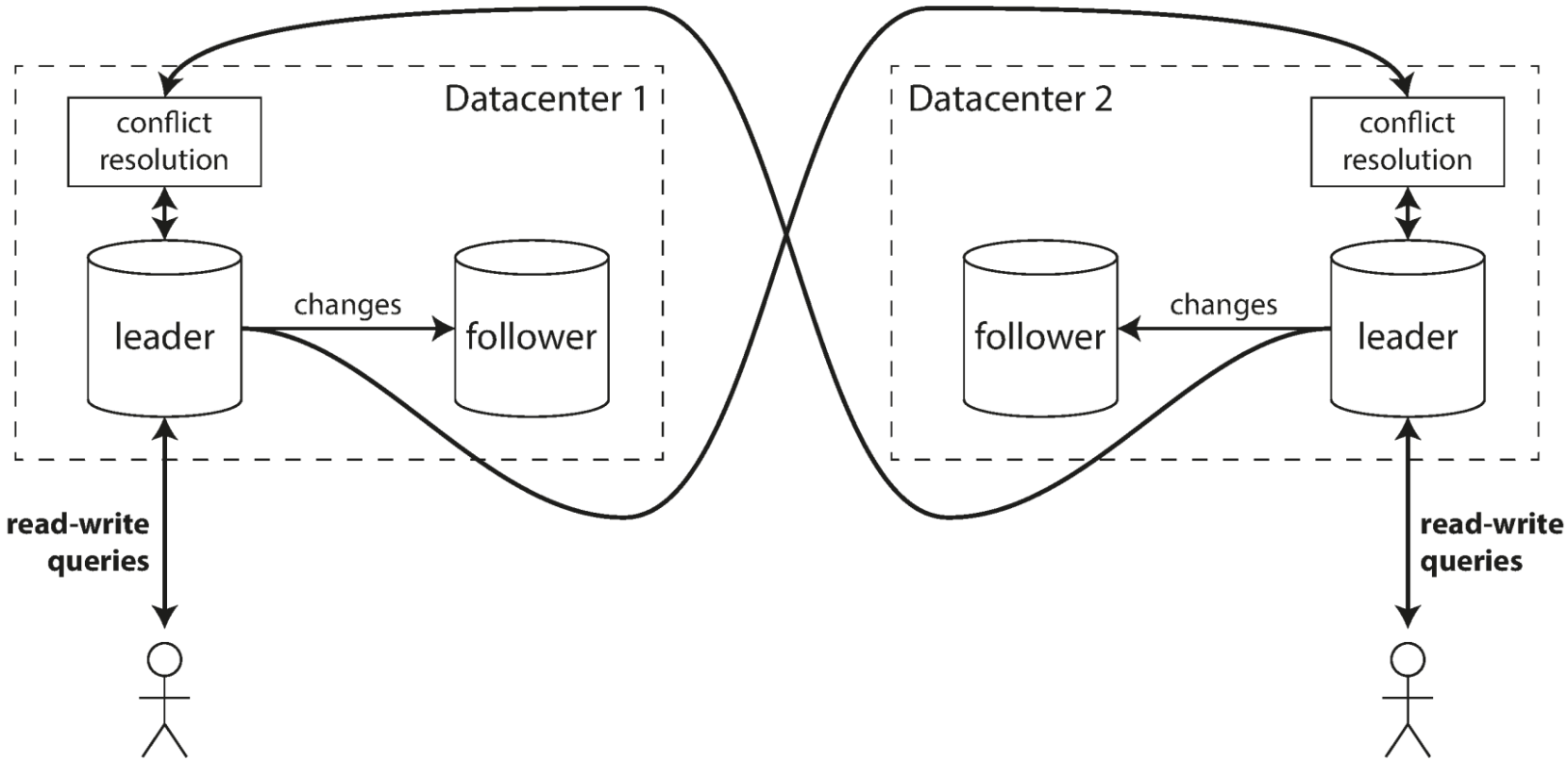
Репликация с несколькими лидерами

- 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)

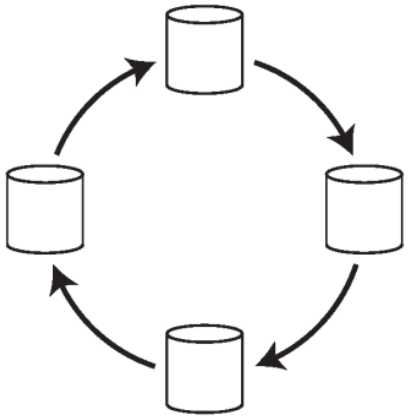
Репликация с несколькими лидерами

- 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
  - Сложнее обновлять схему данных

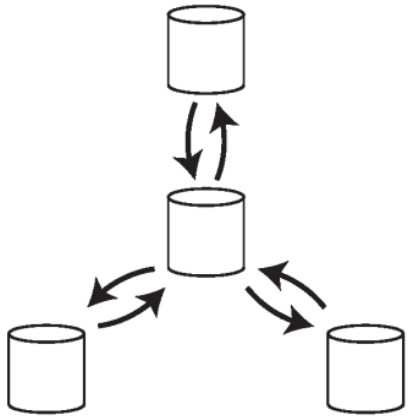
Асинхронный режим



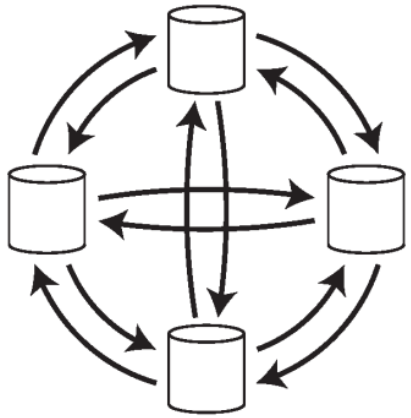
Топологии репликации



(a) Circular topology

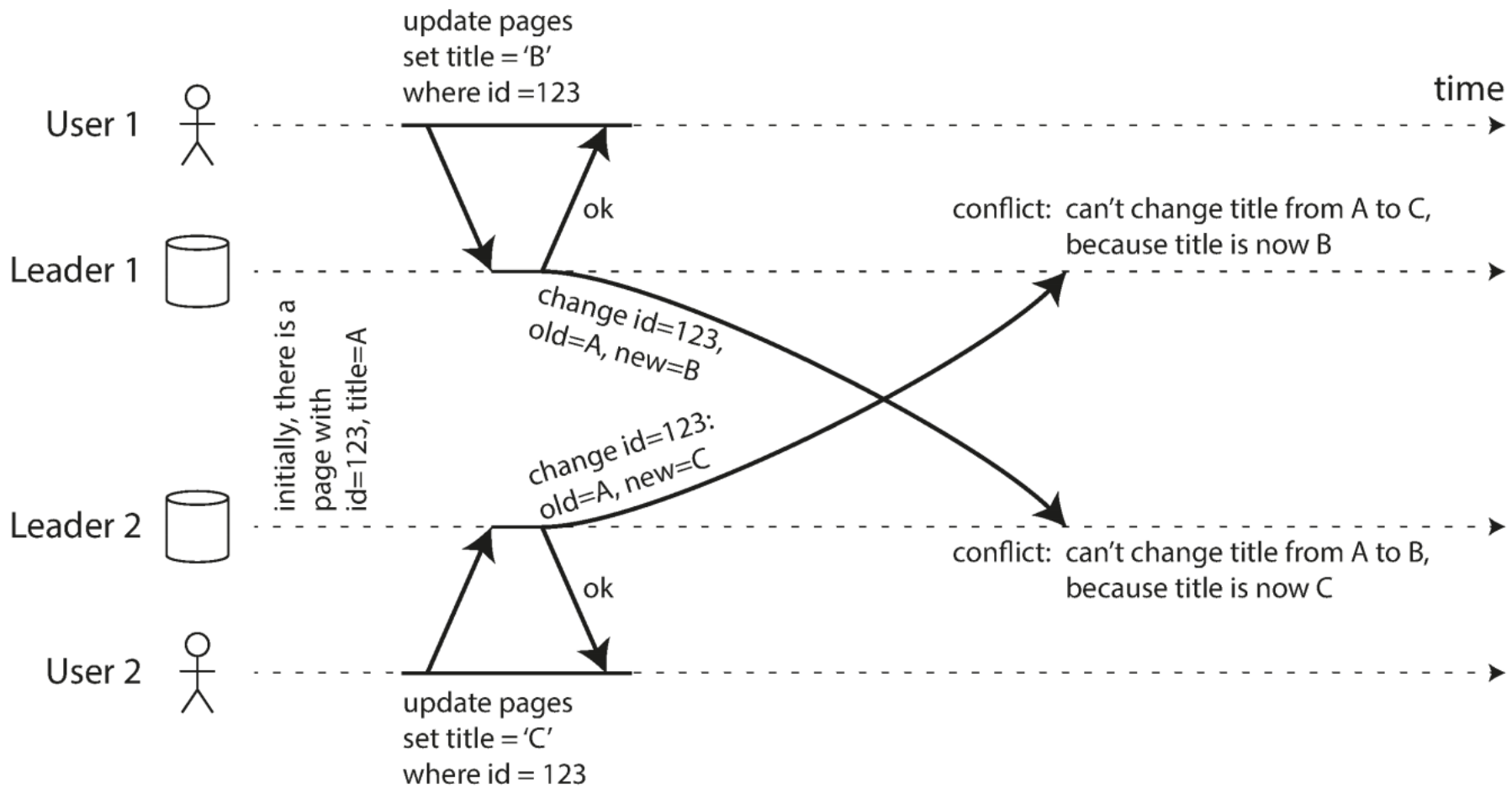


(b) Star topology



(c) All-to-all topology

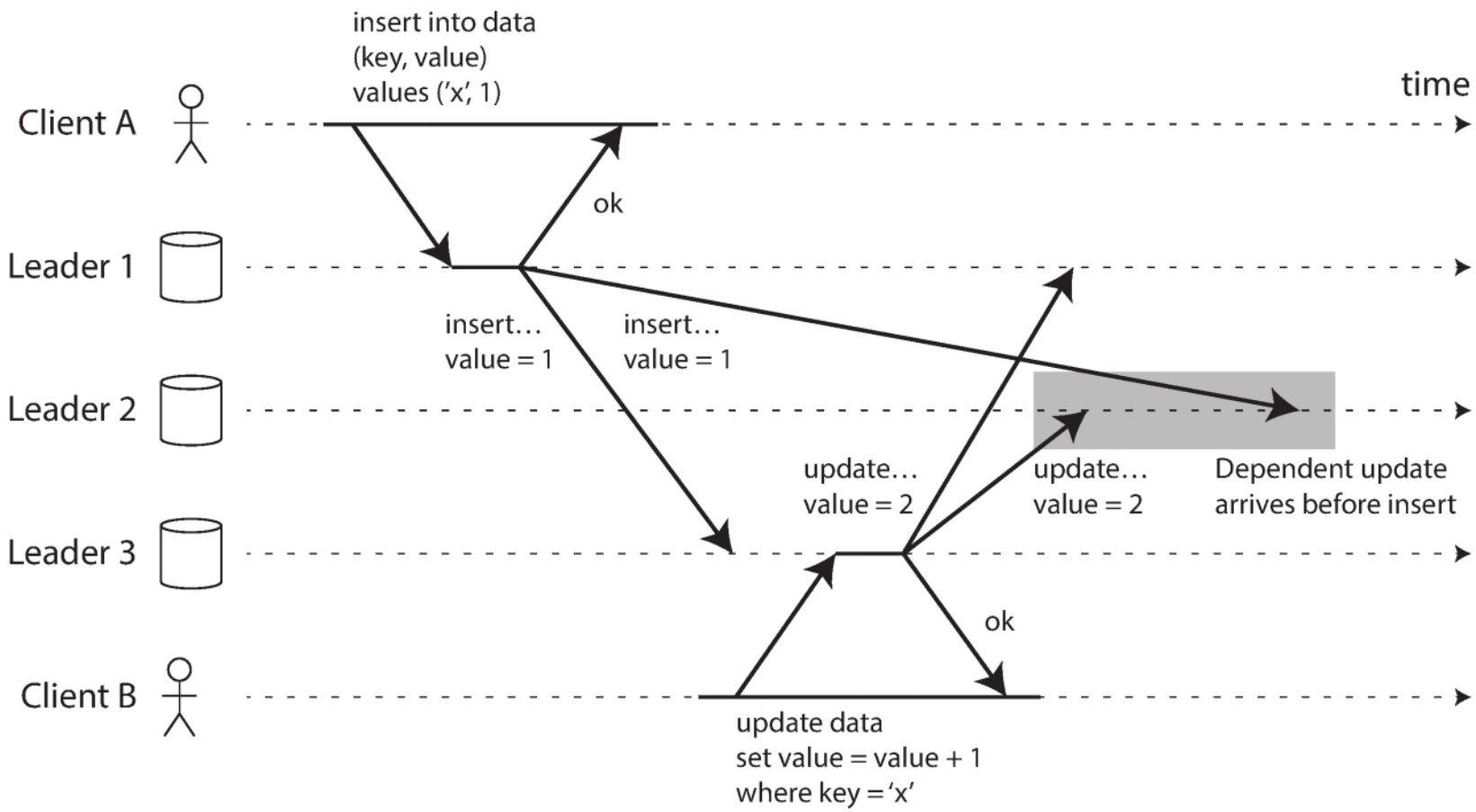
Конфликт при записи данных



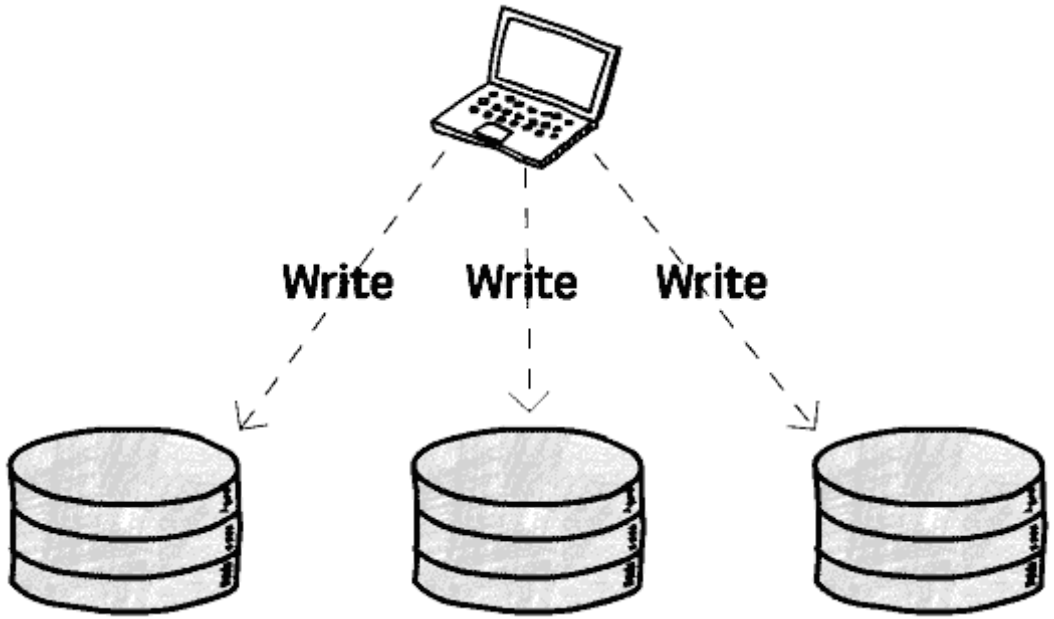
Разрешение конфликтов

- 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 ("B/C")
  - 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 без лидера



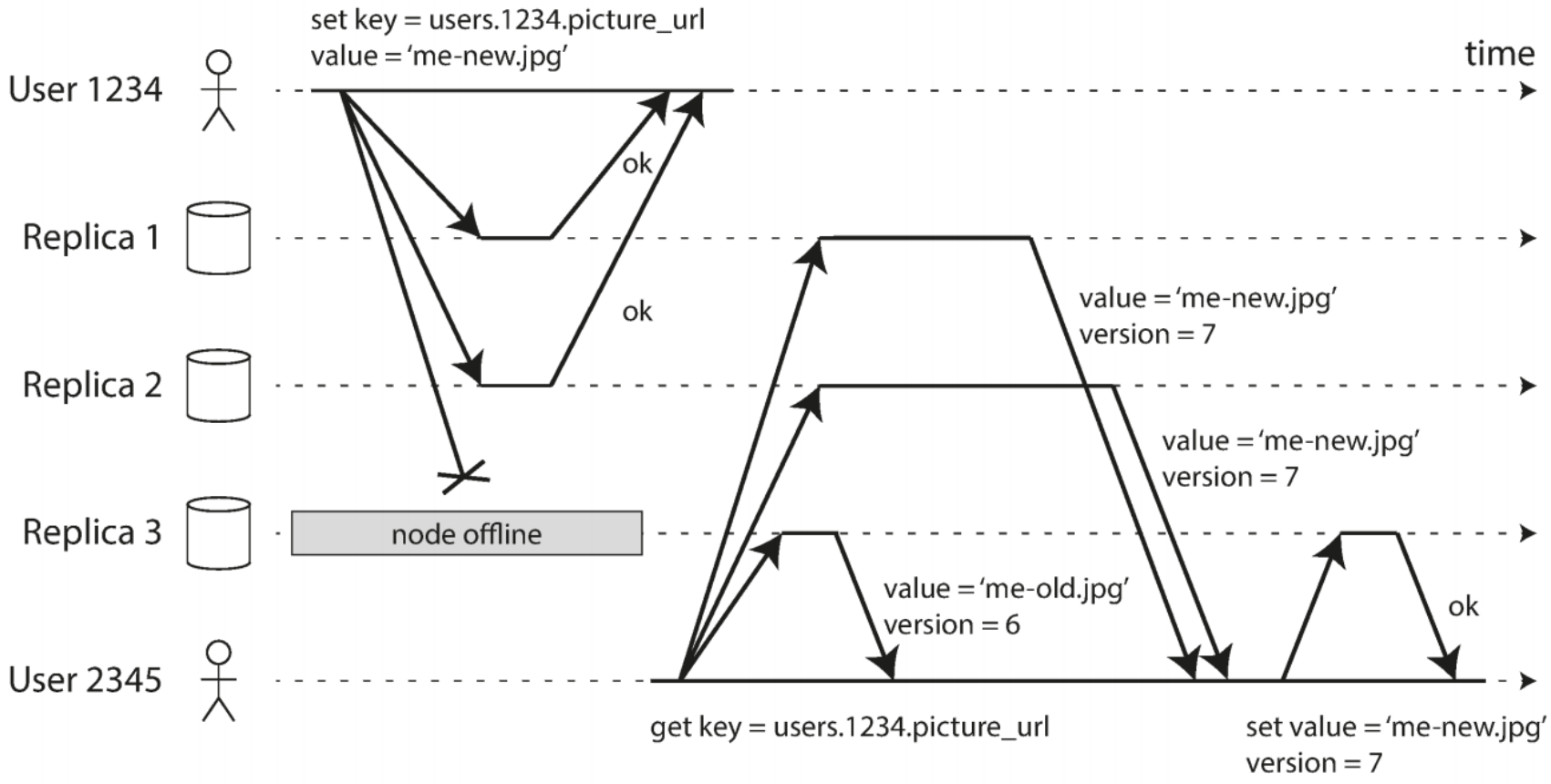
Репликация без лидера

- 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 (запись и чтение с кворумом)
- Gifford D.K. Weighted Voting for Replicated Data (1979)
- Примеры: Dynamo, Riak, Cassandra, Voldemort

Репликация без лидера

- 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

Запись и чтение с кворумом



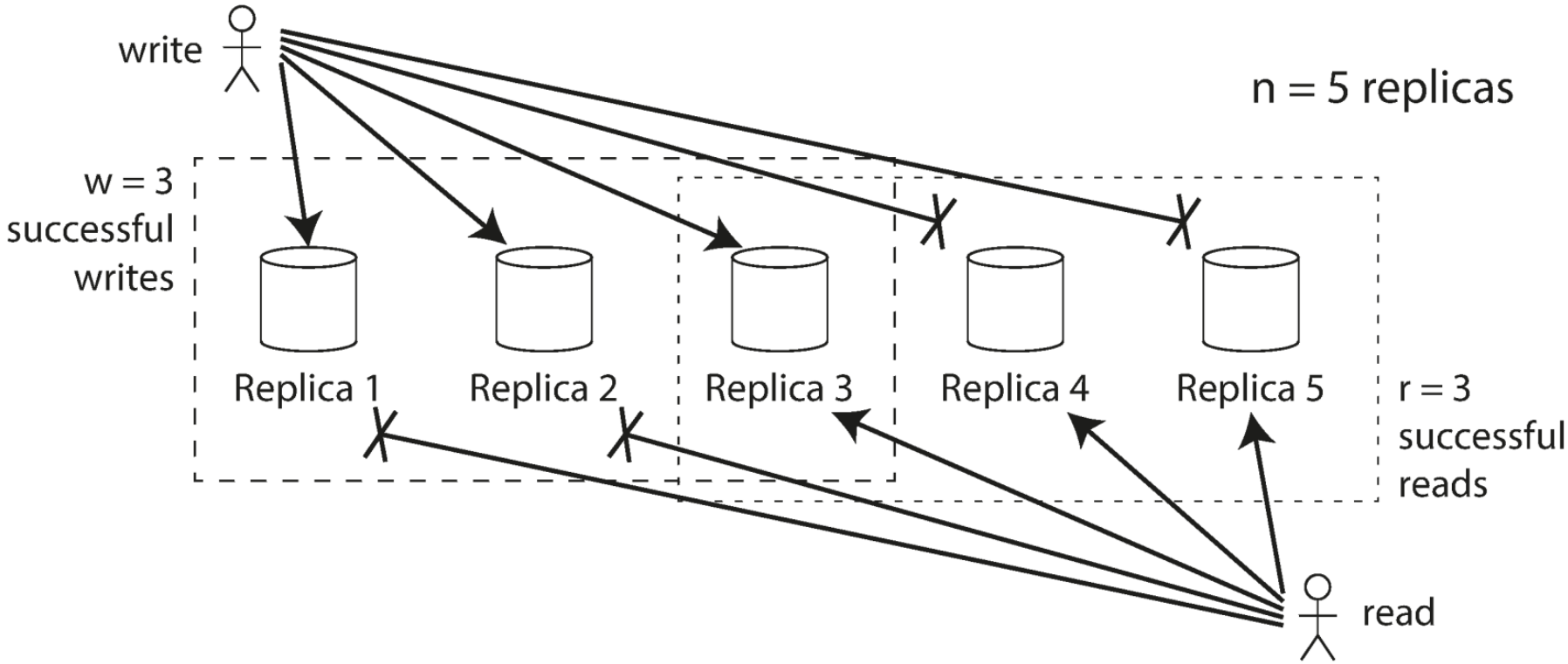
Дополнительные механизмы

- 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
- См. Dynamo: Amazon’s Highly Available Key-value Store (2007)

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
- Усточивость к отказам
  - $W < N,\ R < N$
  - $N = 3,\ W = 2,\ R = 2$ — допустим отказ 1 узла
  - $N = 5,\ W = 3,\ R = 3$ — допустимы отказы 2 узлов

Пример: N=5 W=3 R=3

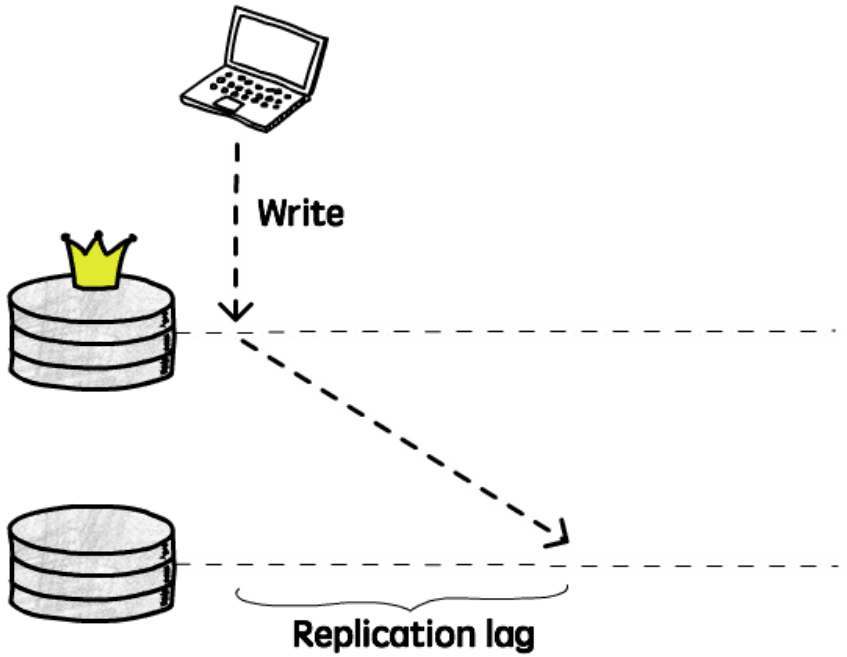


Согласованность (consistency)

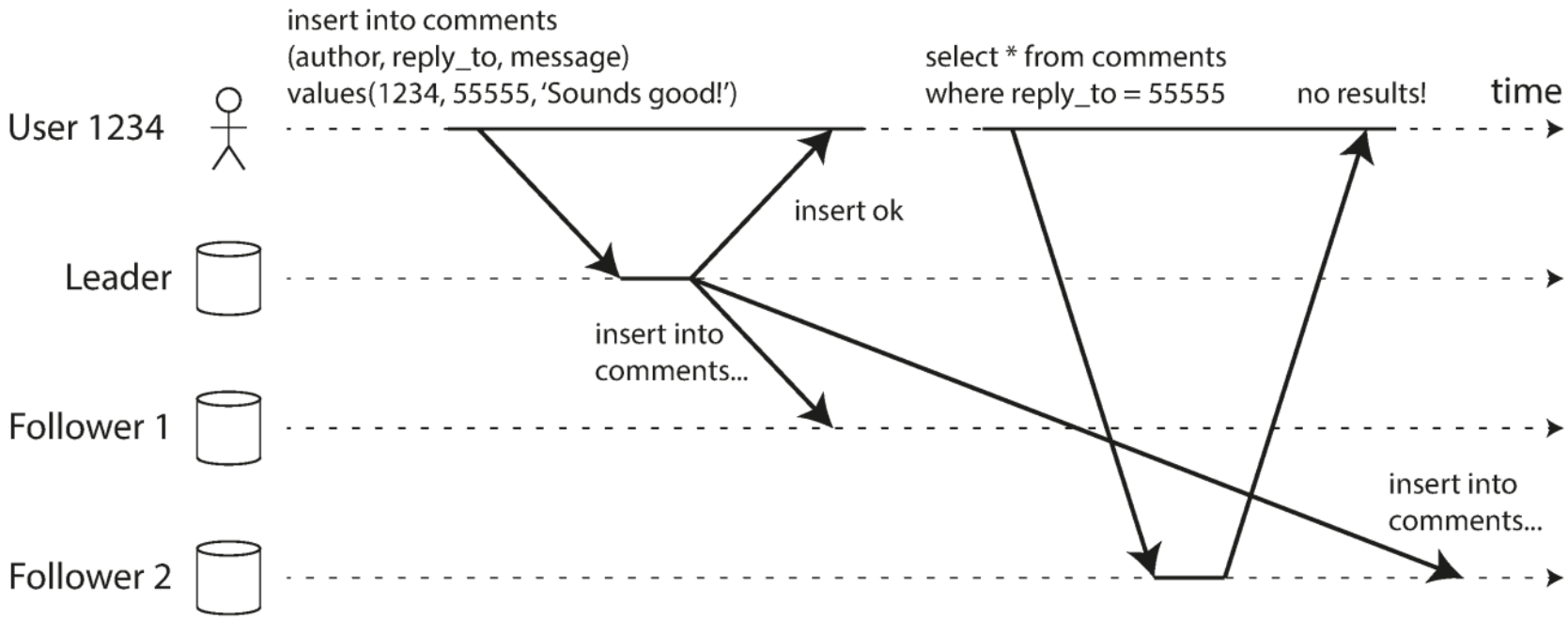
- Какие гарантии требуются от системы с несколькими репликами данных и одновременно работающими клиентами?
  - В любой момент времени все реплики хранят одинаковое значение
  - Каждое чтение возвращает последнее записанное значение
  - Некоторые гарантии на порядок и видимость результатов операций
- Aguilera M., Terry D. The many faces of consistency (2016)

Репликация и согласованность

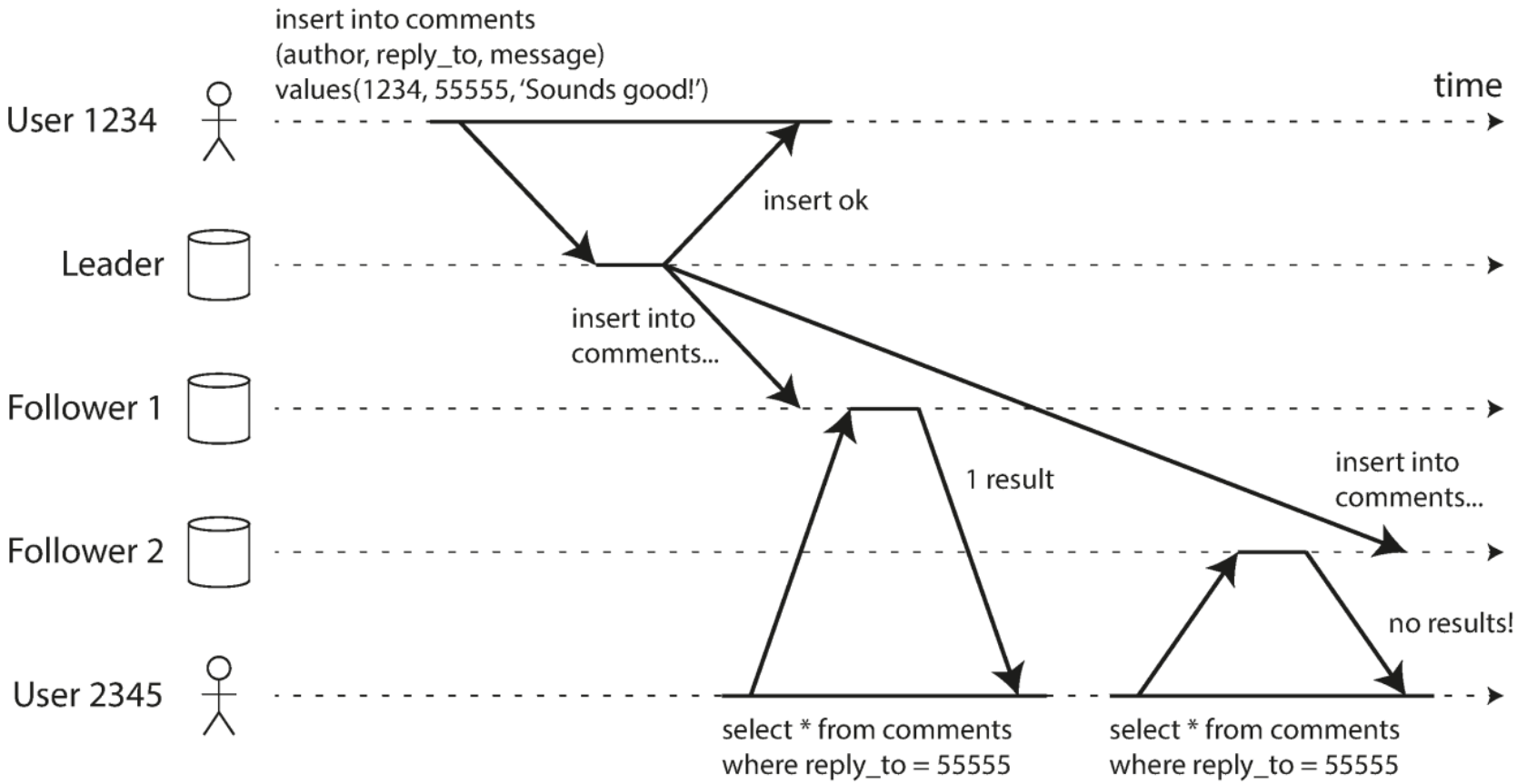
- Асинхронная репликация приводит к отставанию реплик (replication lag)
- Чтение с реплик приводит к нарушению согласованности



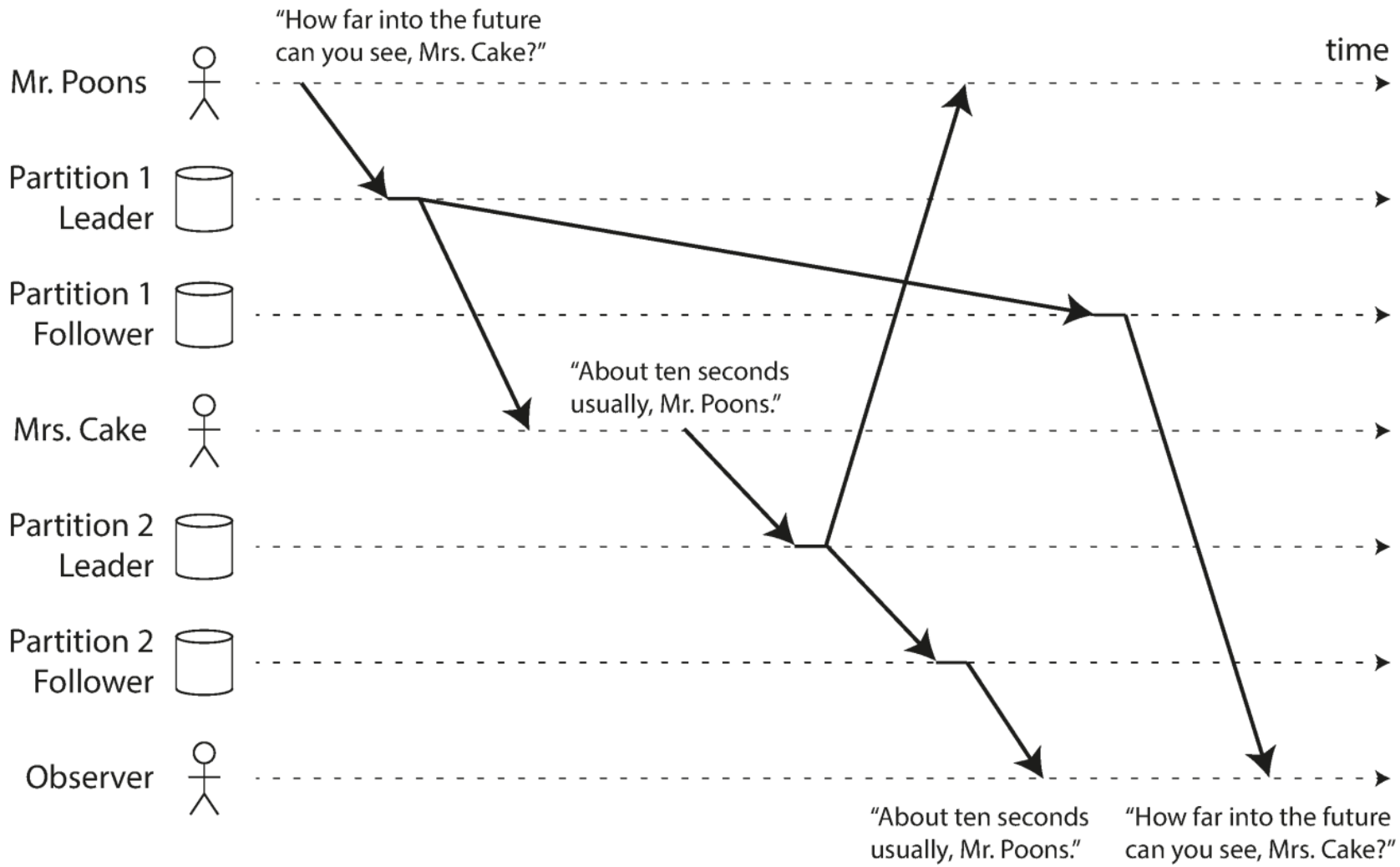
Чтение устаревших данных после записи



Путешествия назад во времени



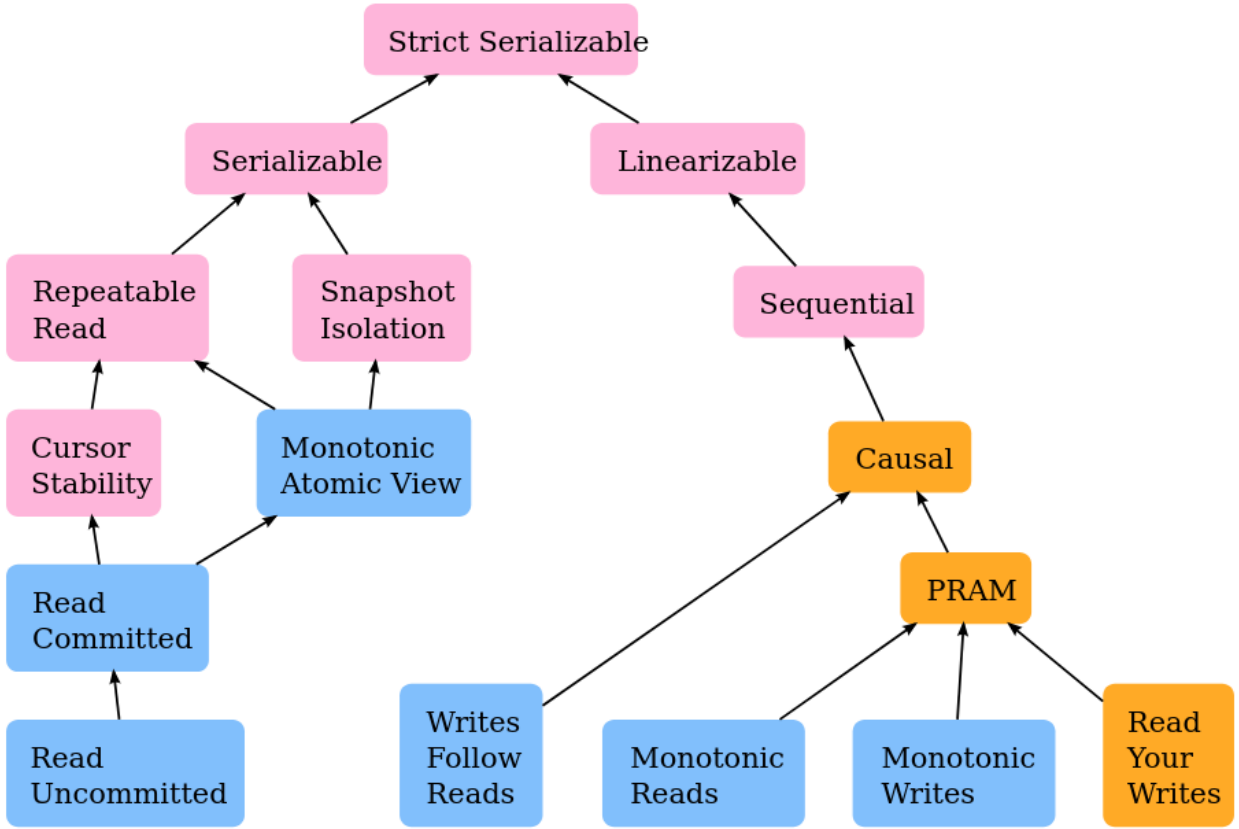
Нарушение причинного порядка



Модели согласованности

- Гарантии на поведение системы в присутствии нескольких реплик и клиентов
- Результаты каких операций записи "видны" клиенту?
  - всех предыдущих операций в соответствии с некоторым единым порядком
  - любого подмножества операций в произвольном порядке
- Возможны различные модели с разной степенью "строгости"

Модели согласованности



http://jepsen.io/consistency

Строгая согласованность

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

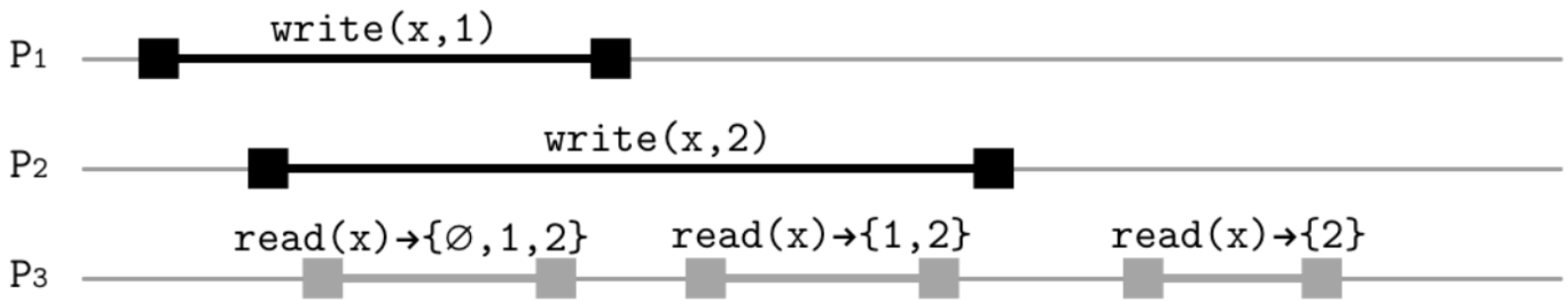
Линеаризуемость

- 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

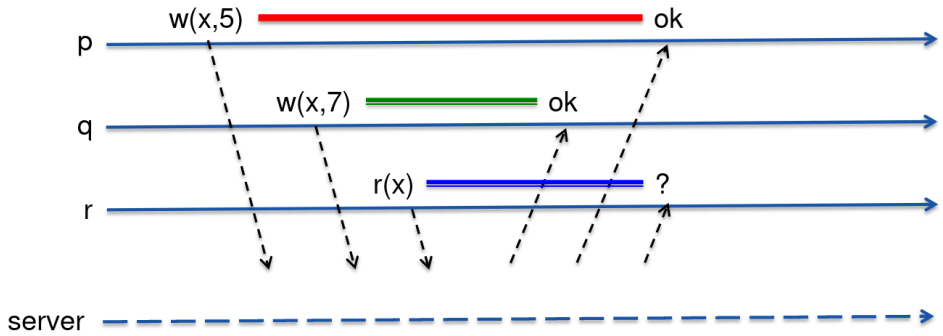
Herlihy M. P., Wing J. M. Linearizability: A correctness condition for concurrent objects (1990)

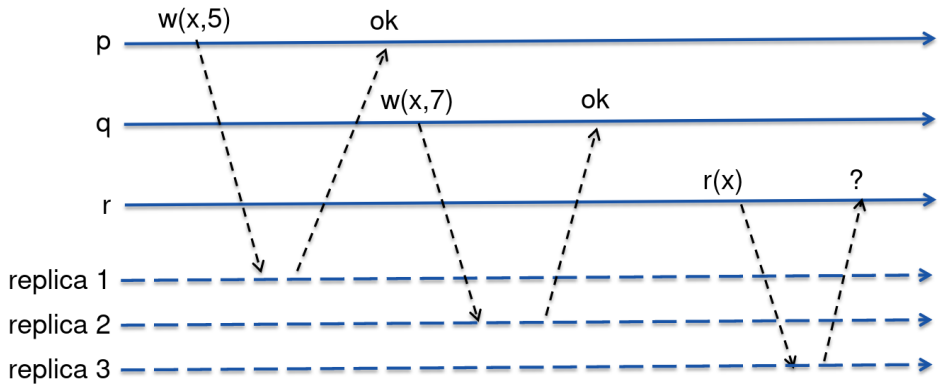
Линеаризуемость (менее формально)

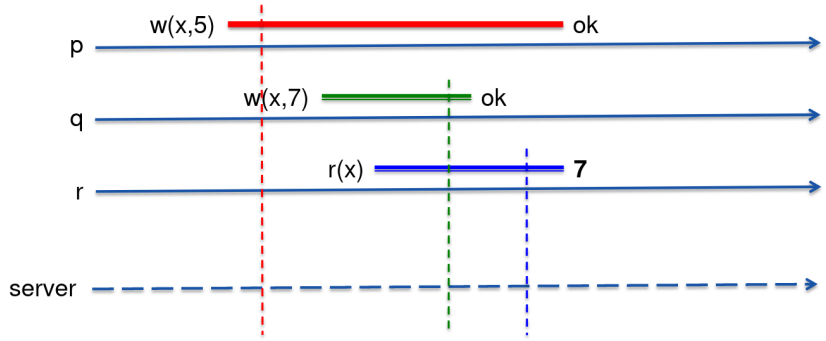
- С точки зрения клиентов система работает так, как будто
  - есть только одна копия данных
  - операции атомарны (операция выполняется мгновенно в некоторой момент времени между ее вызовом и завершением)
- Все клиенты видят операции в одном (глобальном) порядке
- Чтение видит результат последней выполненной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

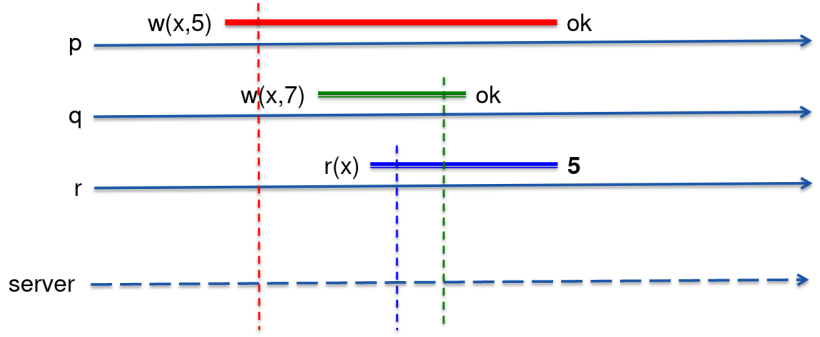


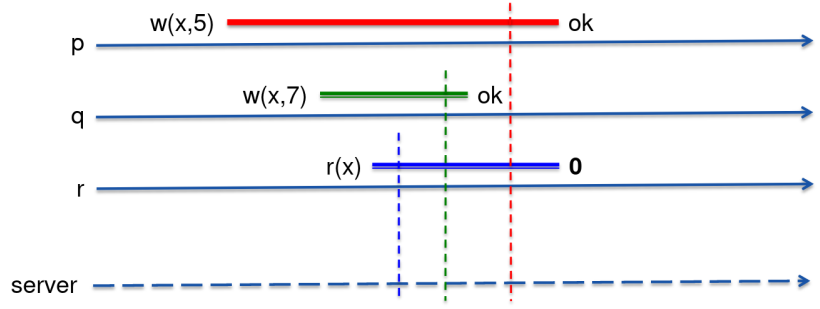
Пример


w(x,5)
ok
p
w(x,7)
ok
q
r(x)
?
r
server

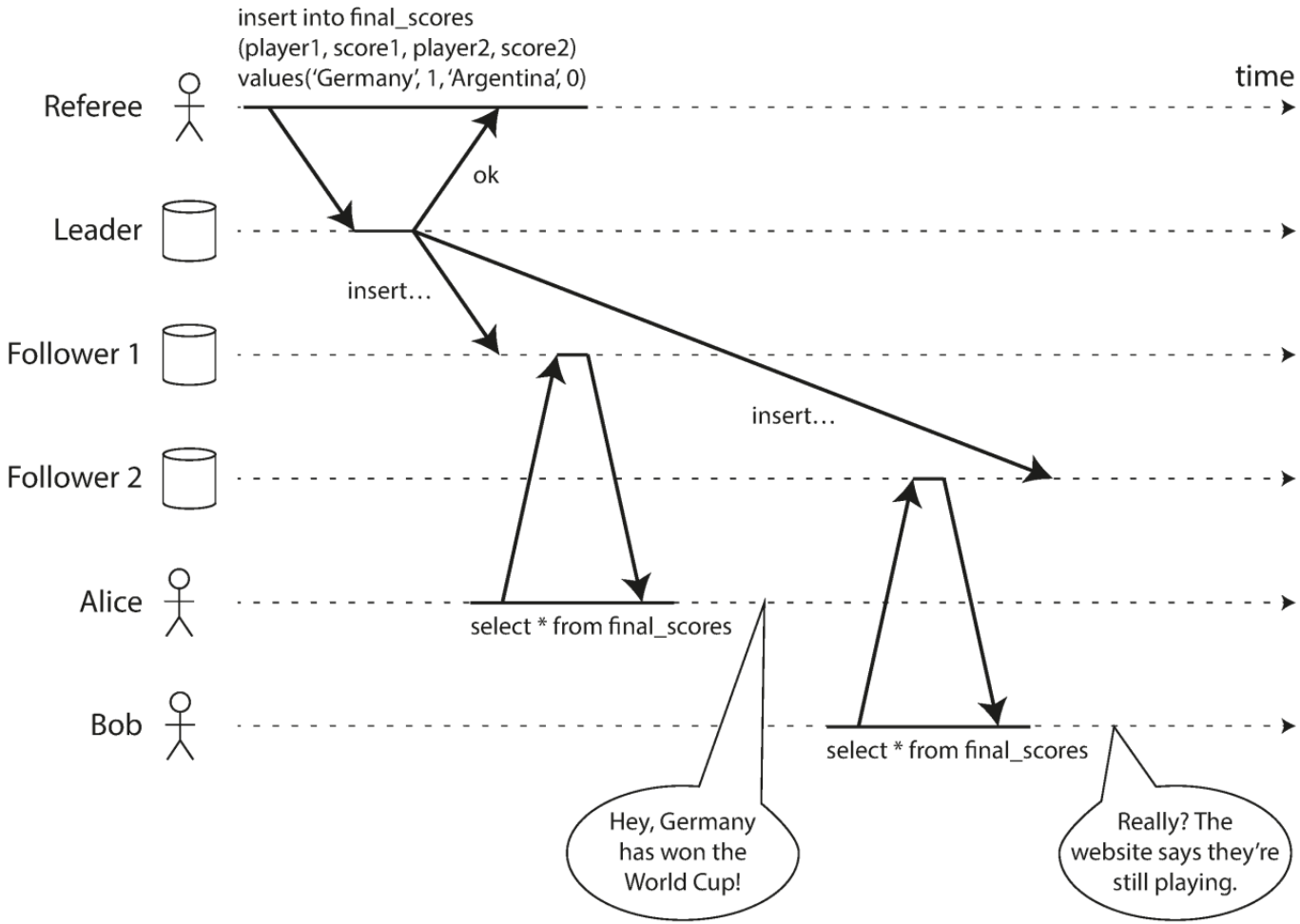

w(x,5)
ok
p
w(x,7)
ok
q
r(x)
?
r
replica 1
replica 2
replica 3


w(x,5)
ok
p
w(x,7)
ok
q
r(x)
7
r
server


w(x,5)
ok
p
w(x,7)
ok
q
r(x)
5
r
server


w(x,5)
ok
p
w(x,7)
ok
q
r(x)
0
r
server

Нарушение линеаризуемости

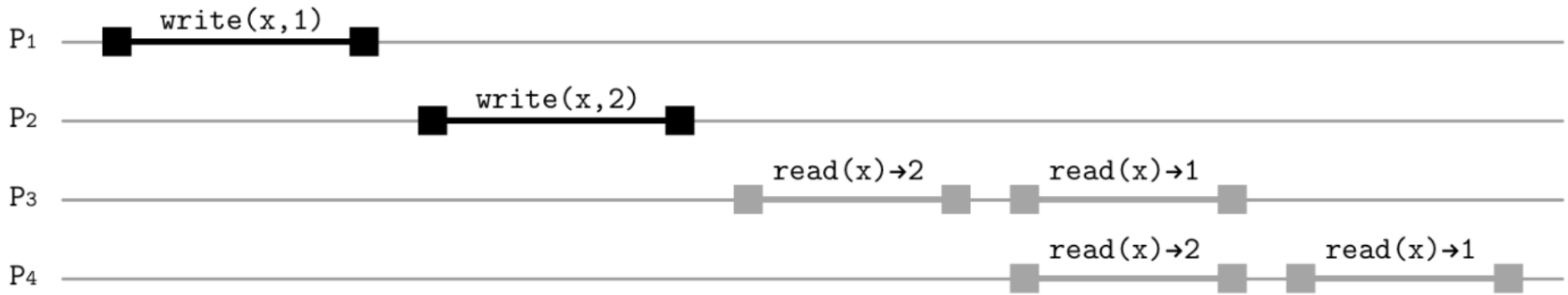


Последовательная согласованность

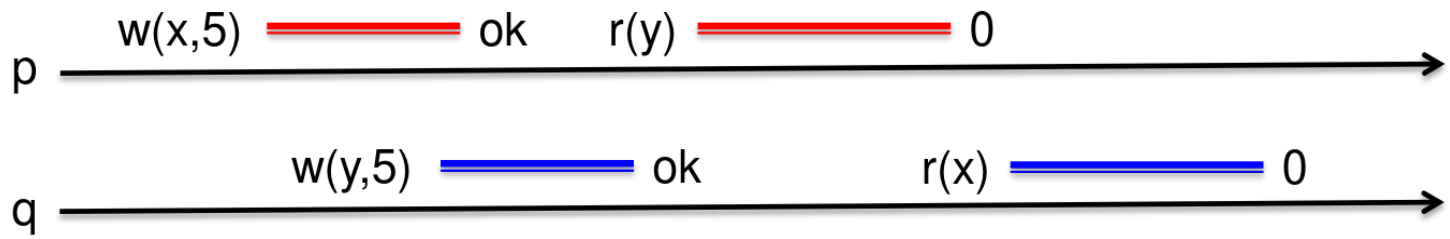
- Менее строгая модель согласованности
- Любое выполнение системы может быть представлено в виде некоторой упорядоченной истории операций, такой что
  - она эквивалентна корректному последовательному выполнению операций над одной копией данных
  - порядок операций согласуется с *порядком выполнения операций* в каждом из процессов
- Аналогично линеаризуемости, все процессы видят операции в одном порядке
  - но не требуется чтобы он соответствовал реальным временам выполнения операций

Lamport L. How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs (1979)

Пример 1

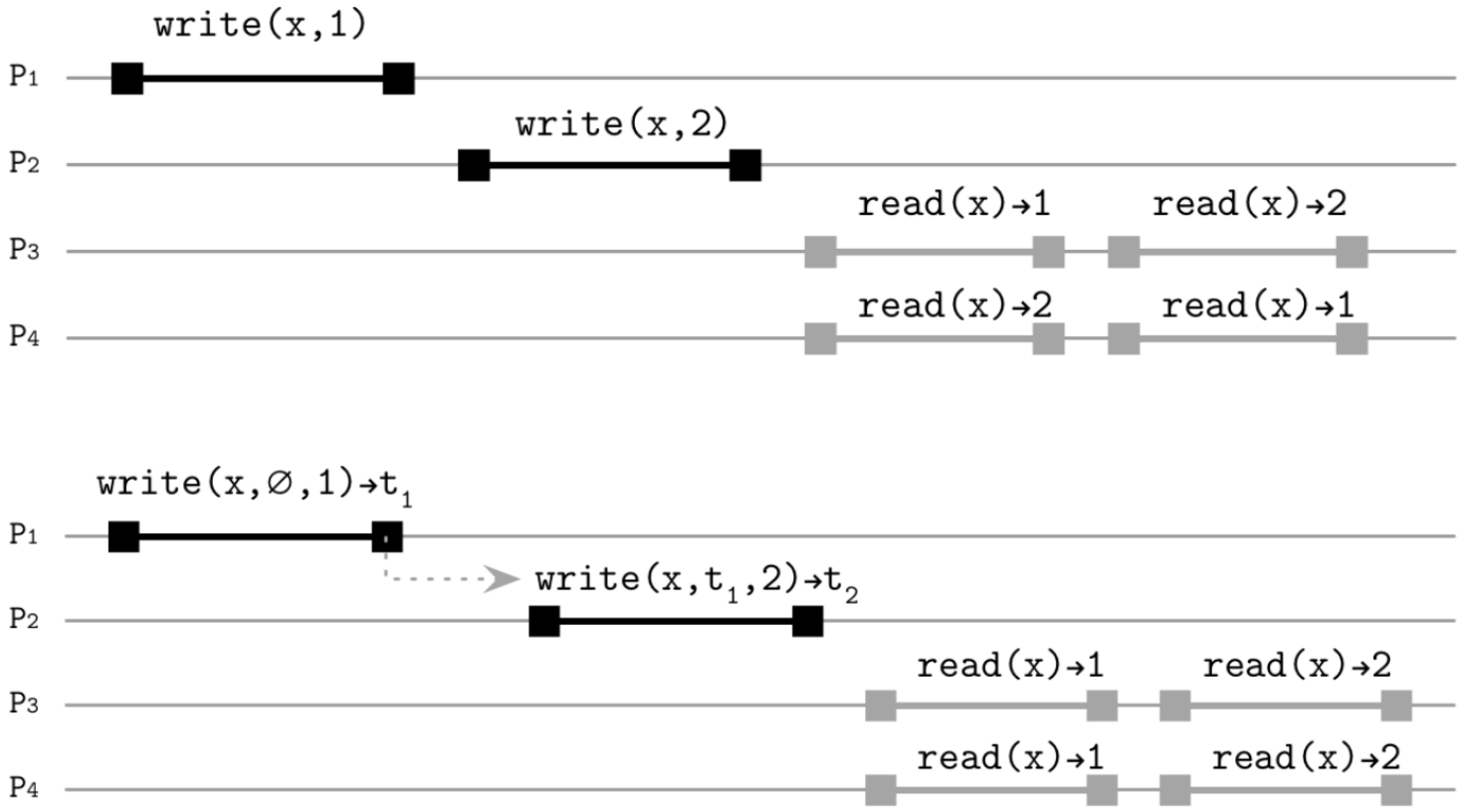


Пример 2



Bornholt J. Memory Consistency Models: A Tutorial

Причинная согласованность



Lloyd W. et al. Don't settle for eventual: scalable causal consistency for wide-area storage with COPS (2011)

Согласованность в конечном счёте

- В отсутствии изменений данных, через какой-то промежуток времени после последней записи («в конечном счёте») все запросы будут возвращать последнее записанное значение
- Наиболее слабая из используемых на практике моделей (см. DNS)
- Ничего не говорит о времени сходимости реплик, возможны конфликты
- Strong eventual consistency гарантирует отстутствие конфликтов

Vogels W. Eventually Consistent (2009)

Другие модели и гарантии

- Consistent Prefix
  - префикс истории операций в соответствии с некоторым глобальным порядком
- Bounded Staleness
  - все записи, выполненные достаточно давно
- Monotonic Reads
  - увеличивающееся подмножество операций (гарантия на сессию)
- Read My Writes
  - все записи, произведенные клиентом (гарантия на сессию)

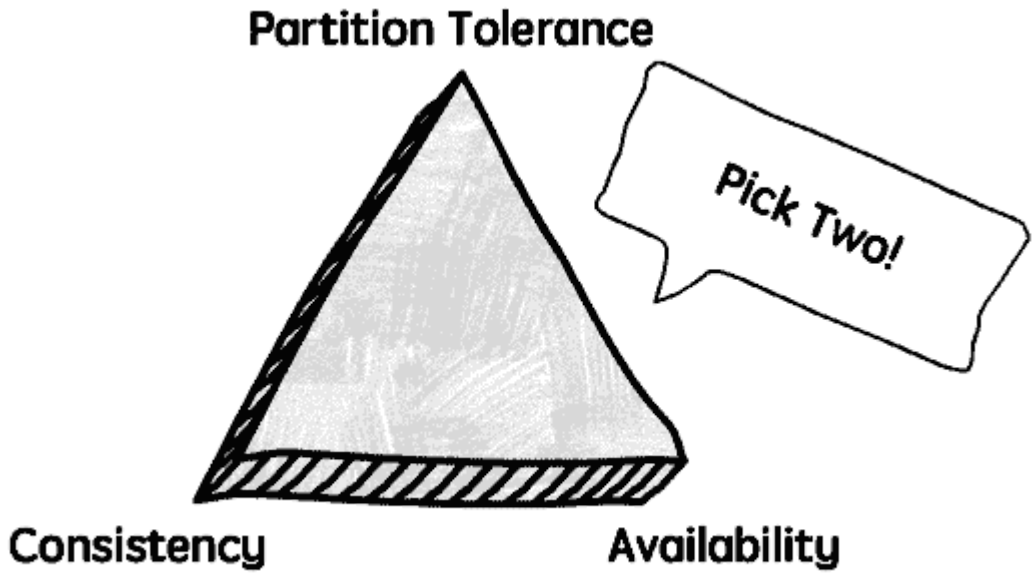
Terry D. et al. Session Guarantees for Weakly Consistent Replicated Data (1994)

Компромиссы

Guarantee	Consistency	Performance	Availability
Strong Consistency	excellent	poor	poor
Eventual Consistency	poor	excellent	excellent
Consistent Prefix	okay	good	excellent
Bounded Staleness	good	okay	poor
Monotonic Reads	okay	good	good
Read My Writes	okay	okay	okay

Terry D. Replicated data consistency explained through baseball (2011)

Теорема CAP



Hale C. You Can’t Sacrifice Partition Tolerance (2010)
Abadi D. Теорема PACELC (2012)
Kleppmann M. A Critique of the CAP Theorem (2015)
Kleppmann M. Please stop calling databases CP or AP (2015)

Литература

- Kleppmann M. Designing Data-Intensive Applications (глава 5)
- Storti B. A Primer on Database Replication
- Petrov A. Database Internals (глава 11)
- Consistency Models

Литература (дополнительно)

- Упомянутые статьи
- Jepsen: MongoDB stale reads
- Jepsen Analyses
- Sewel P. x86-TSO: A Rigorous and Usable Programmer's Model for x86 Multiprocessors (2010)