
Яндекс

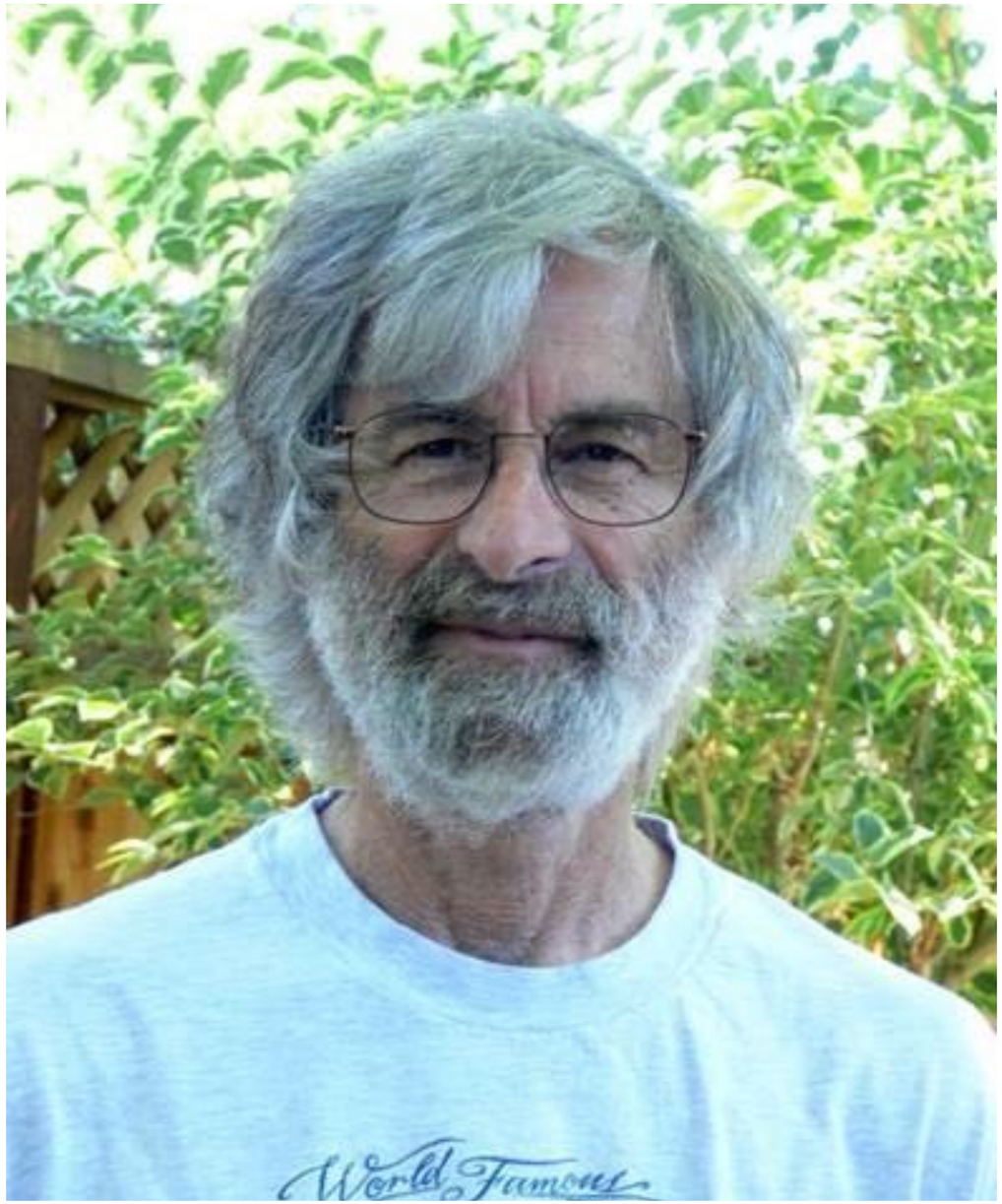
Яндекс

Тестирование распределенных систем

Андрей Сатарин

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable

Leslie Lamport



Распределенные системы в мире

- MongoDB
- Apache Cassandra
- Apache Hadoop/MapReduce
- Apache ZooKeeper
- Apache Kafka
- ElasticSearch

Распределенные системы в Яндексе

- YT — платформа вычислений в парадигме Map/Reduce
  https://habrahabr.ru/company/yandex/blog/311104/
- Yandex Query Language — декларативный язык запросов к системе обработки данных
  https://habrahabr.ru/company/yandex/blog/312430/
- Media Storage —распределенная система хранения данных
  https://habrahabr.ru/company/yandex/blog/311806/
- ClickHouse — открытая колоночная распределенная база данных
  https://clickhouse.yandex/
  https://habrahabr.ru/company/yandex/blog/303282/

Подробнее тут https://events.yandex.ru/events/meetings/15-oct-2016/

Зачем нужны распределенные системы?

- Производительность — много машин могут сделать больше работы в единицу времени, чем одна
- Отказоустойчивость — сбой одной или нескольких машин не обязательно приводит к остановке системы

Производительность

Sort benchmark (http://sortbenchmark.org/)
Мировой рекорд 2016 года:

- 512 машин
- Сортировка 100 TB данных за 134 секунды

Больше не буду говорить про производительность

Отказоустойчивость

- Вероятность сбоя одного узла низкая (железо надежно)
- Больше узлов — больше сбоев каждый день

Пример: сбой одной машины раз в год, в кластере
500 машин — больше одного сбоя в день

Отказоустойчивость: сеть

The Network is Reliable
http://queue.acm.org/detail.cfm?id=2655736

«They found an average failure rate of 5.2 devices per day and 40.8 links per day, with a **median time to repair of approximately five minutes** (and a maximum of one week)»

«Median incident duration of 2 hours and 45 minutes for the highest-priority tickets and a **median duration of 4 hours** and 18 minutes for all tickets»

Отказоустойчивость: диски

One Billion Drive Hours and Counting: Q1 2016 Hard Drive Stats

https://www.backblaze.com/blog/hard-drive-reliability-stats-q1-2016/

«The overall **Annual Failure Rate of 1.84%** is the lowest quarterly number we've ever seen.»

Loud Sound Just Shut Down a Bank's Data Center for 10 Hours

http://www.techworm.net/2016/09/banks-data-center-shut-10-hours-loud-sound.html

«The HDD cases started to vibrate, and the vibration was transmitted to the read/write heads, **causing them to go off the data tracks**.»

Отказоустойчивость: силы природы

Google data center loses data following four lightning strikes
http://www.extremetech.com/computing/212586-google-data-center-loses-data-following-four-lightning-strikes

«However, **four successive lightning strikes** on the electrical systems of its data center pushed the buffering and backups to their limits.»

Mathias Verraes
@mathiasverraes

Follow

There are only two hard problems in distributed systems: 2. Exactly-once delivery 1. Guaranteed order of messages 2. Exactly-once delivery

RETWEETS 6,281 LIKES 4,147

9:40 PM - 14 Aug 2015

6.3K 4.1K

https://twitter.com/mathiasverraes/status/632260618599403520

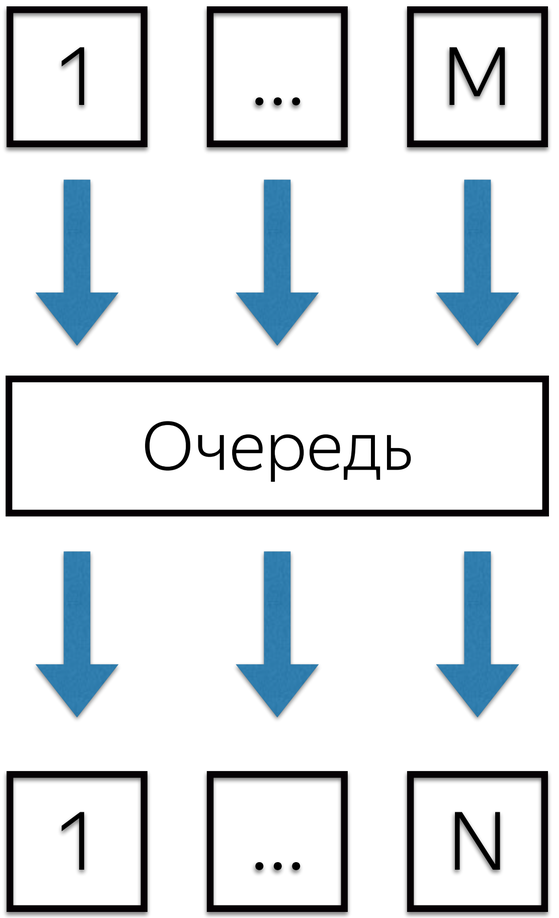
Распределенная персистентная очередь

Зачем нужна очередь?

Очередь — M + N интеграций

Нет очереди — M x N интеграций

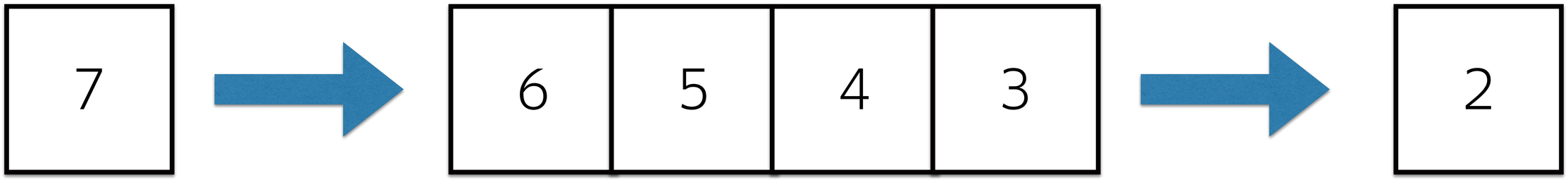
M + N << M x N



Системы источники данных

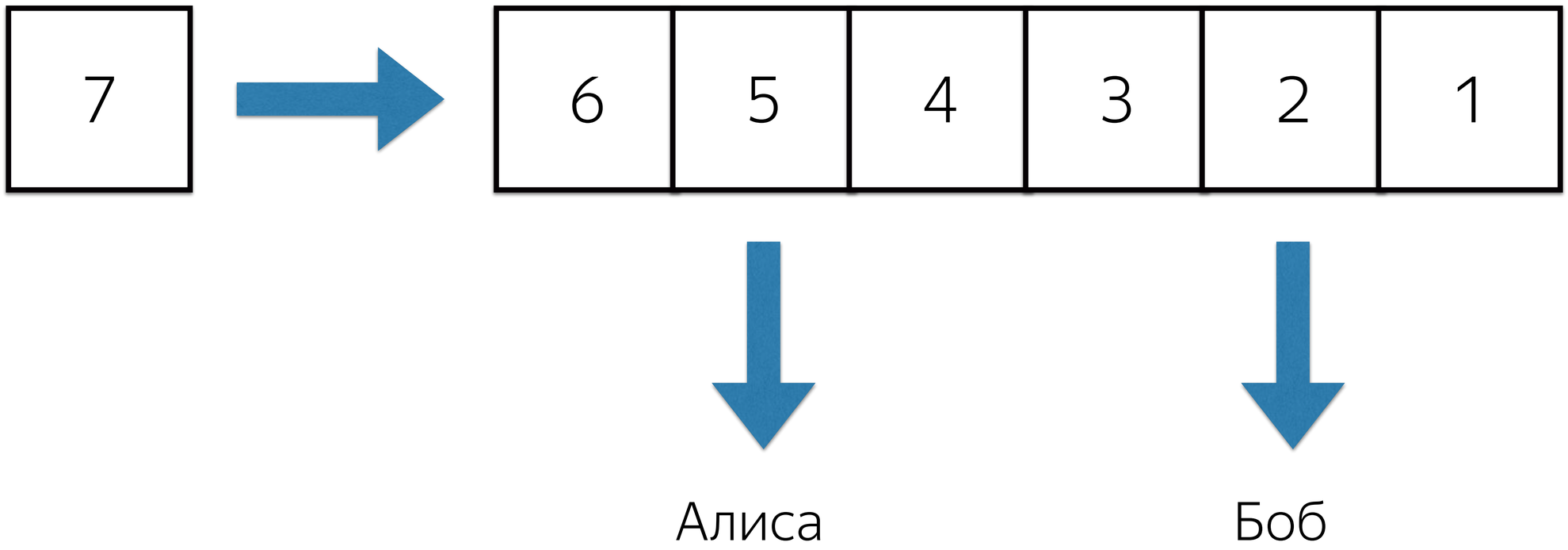
Системы потребители данных

Очередь

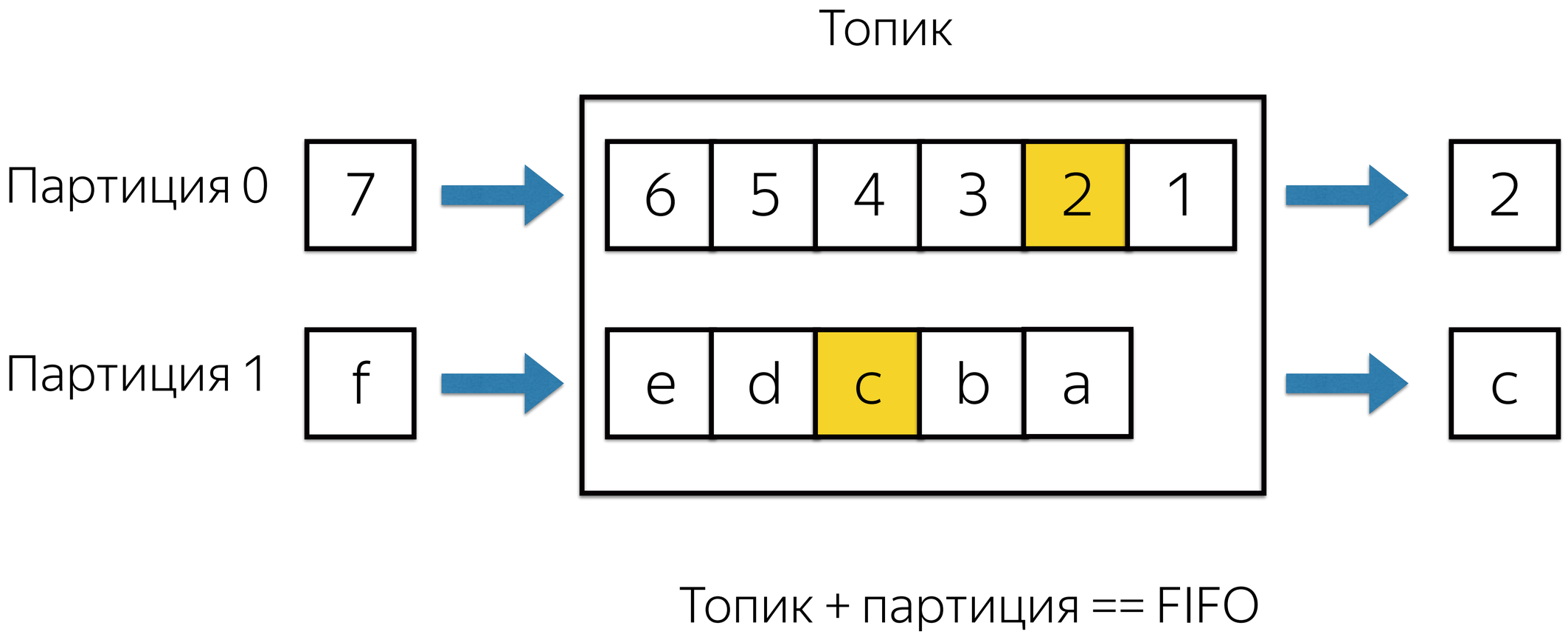


First In First Out (FIFO)

Персистентная очередь

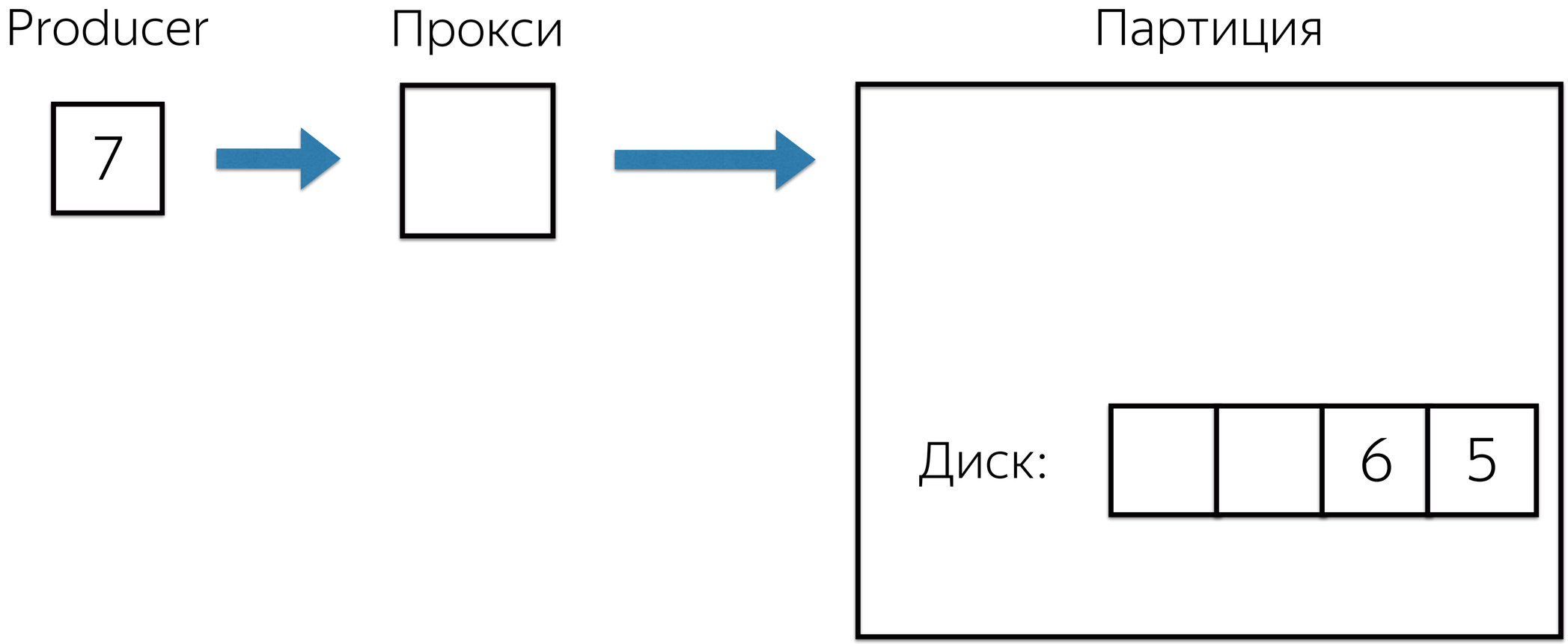


Распределенная персистентная очередь

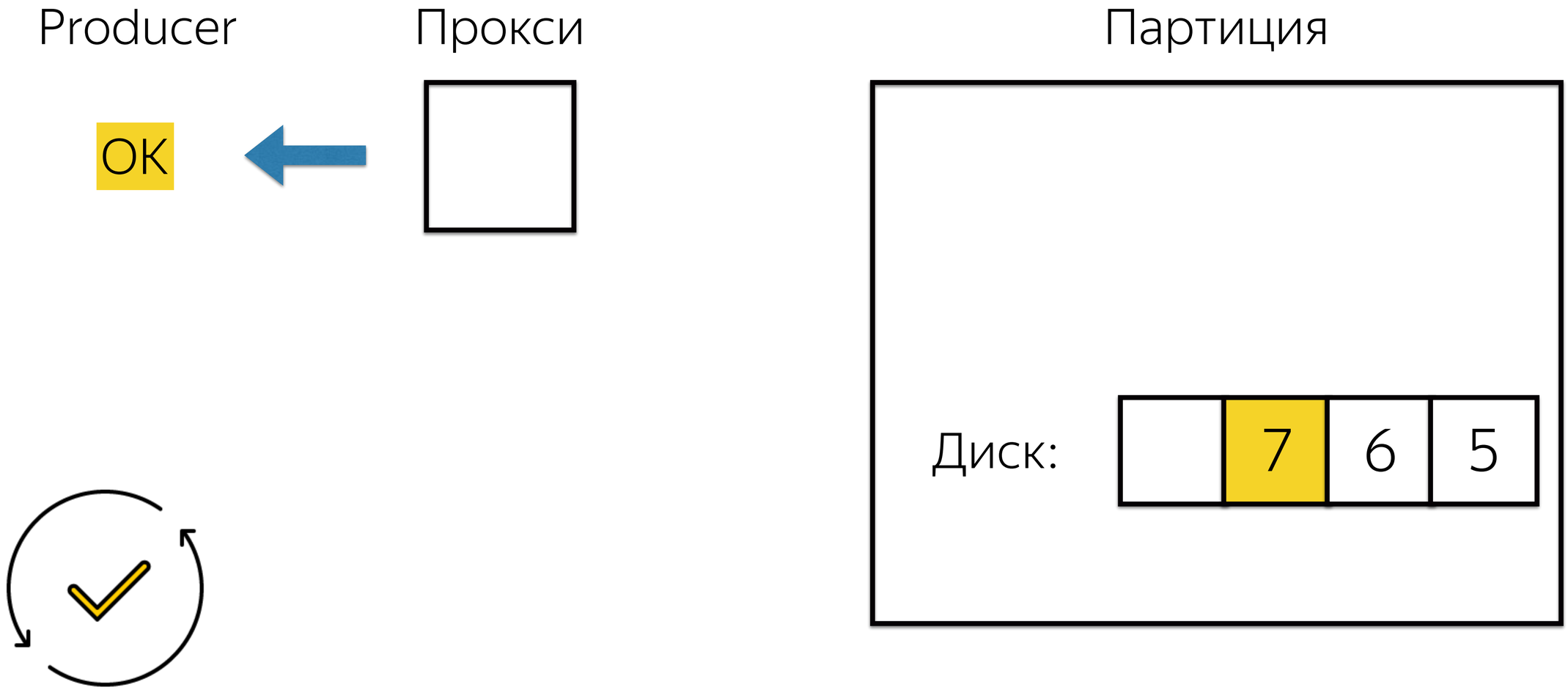


Топик + партиция == FIFO

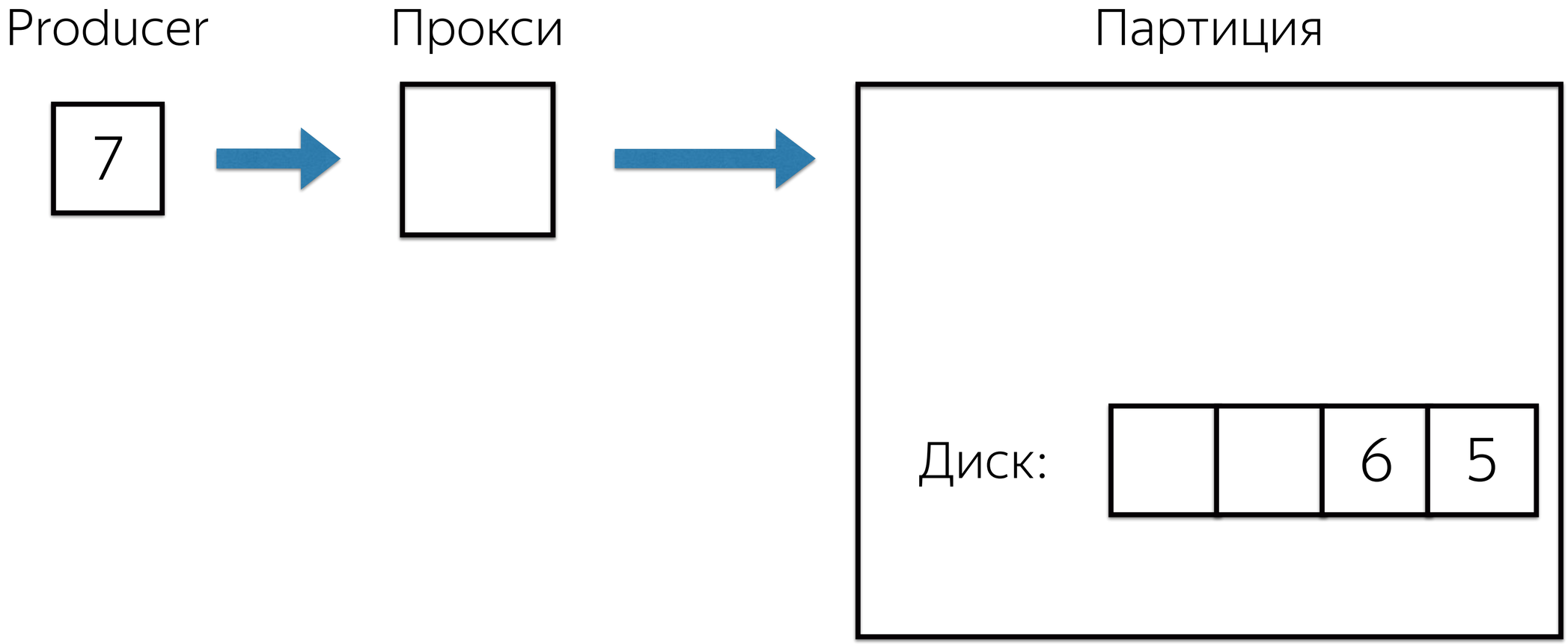
Распределенная персистентная очередь: запись



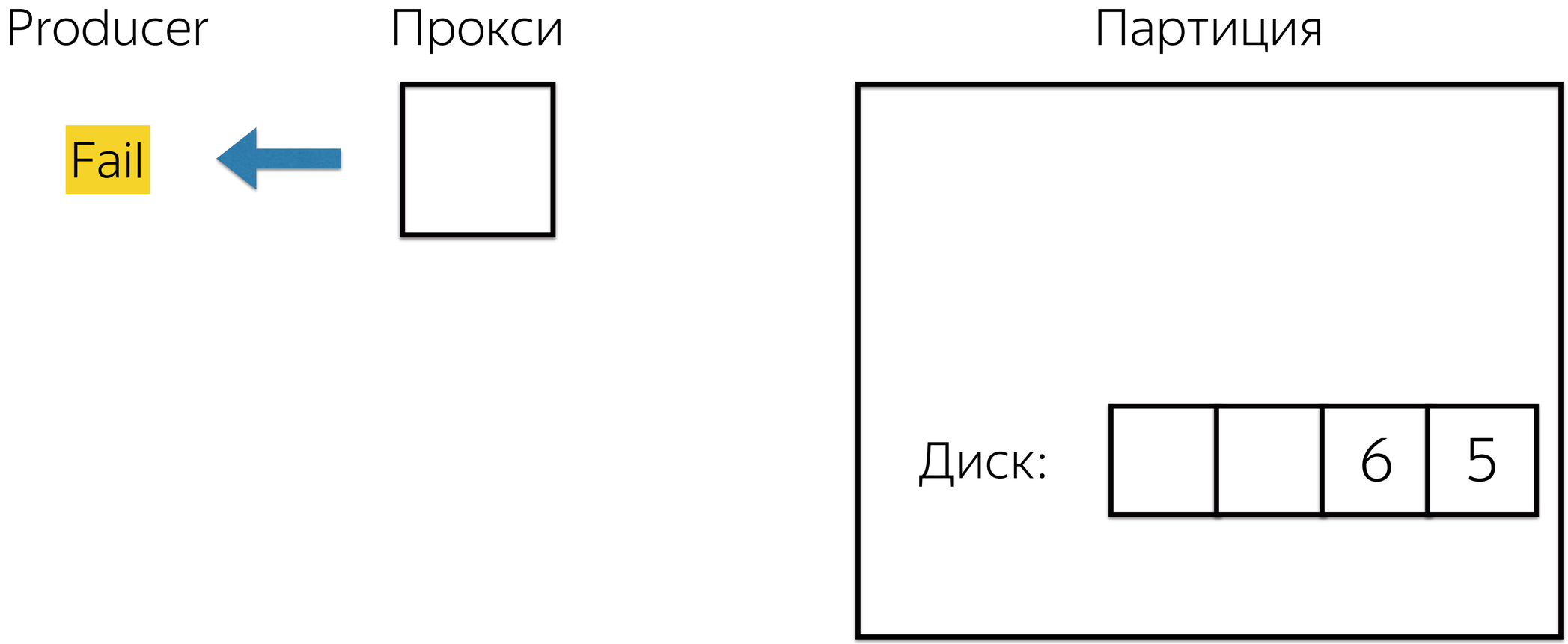
Распределенная персистентная очередь: запись



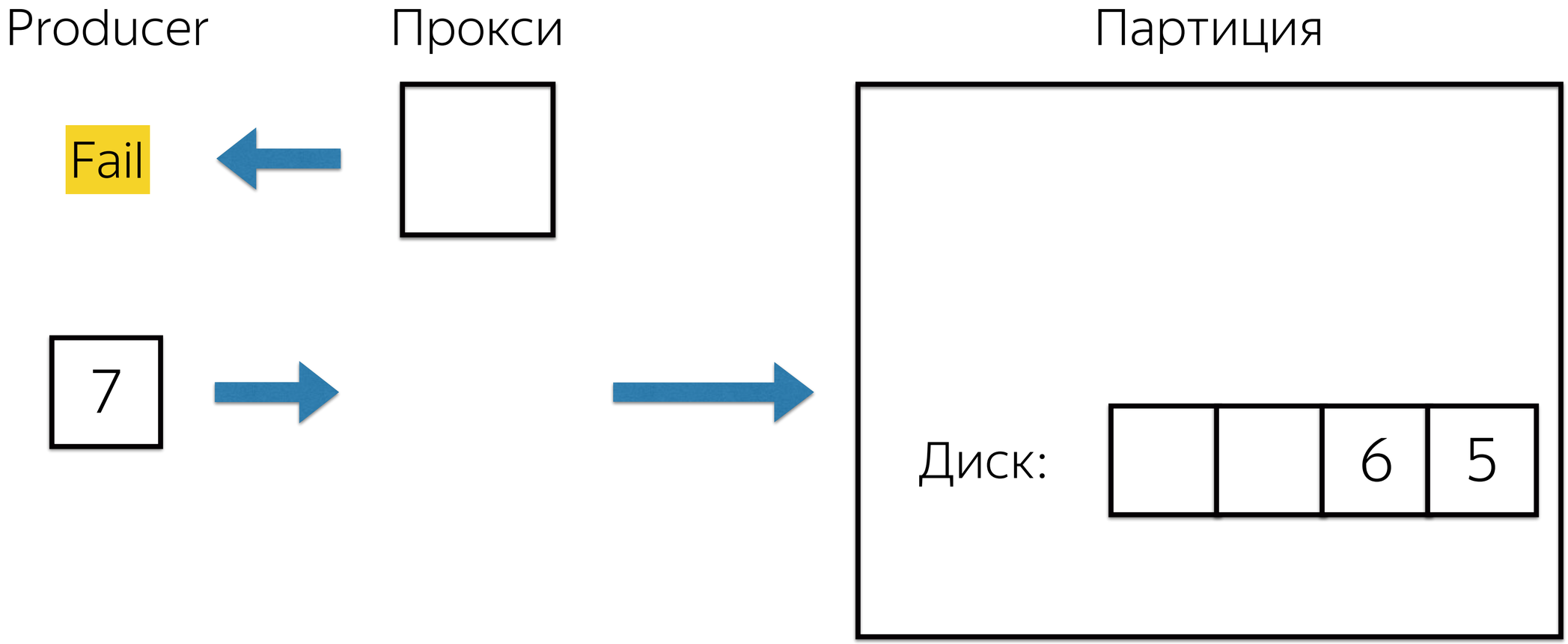
Распределенная персистентная очередь: запись



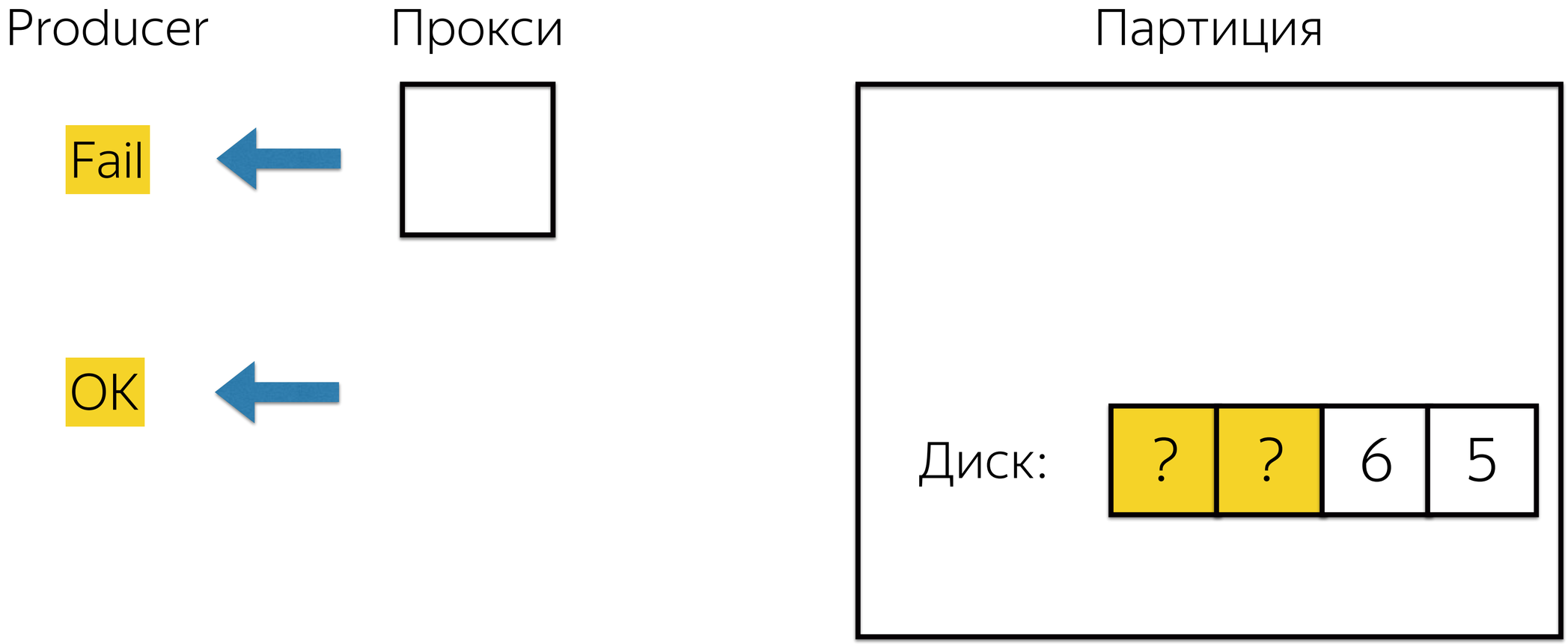
Распределенная персистентная очередь: запись



Распределенная персистентная очередь: запись



Распределенная персистентная очередь: запись



Семантика распределенных очередей

- At most once —можем терять данные, нет дублей (/dev/null)
- **Exactly once** — нет потерь, нет дублей (наша очередь)
- At least once — не теряем данные, возможны дубли (Apache Kafka)

Dan Allen
@mojavelinux

Follow

I'm beginning to believe that writing well-designed tests actually requires more technical skill than the code it tests.

RETWEETS 168 LIKES 161

4:56 AM - 9 Jul 2016

168 161

https://twitter.com/mojavelinux/status/751595888435294209

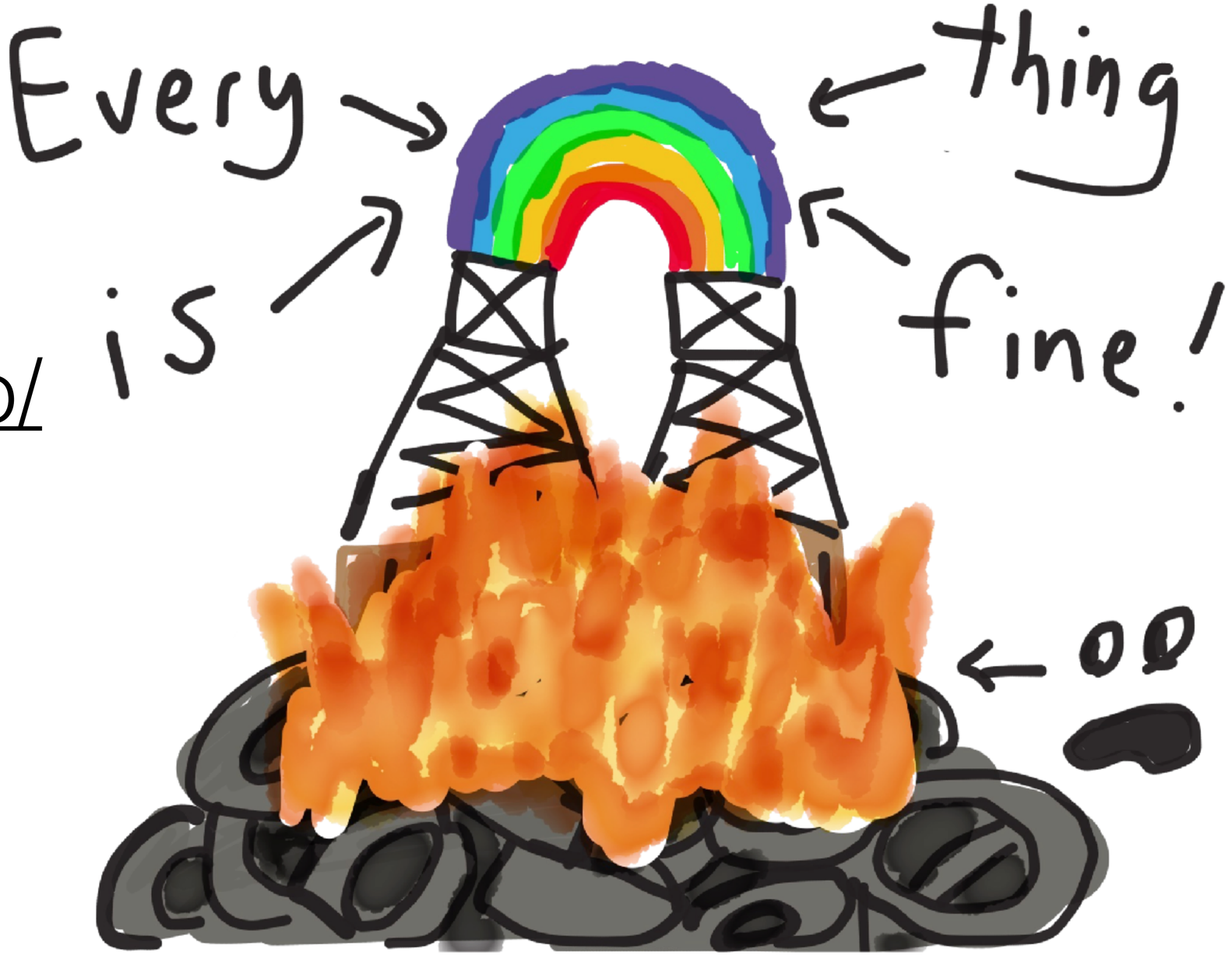
Подходы к тестированию (что уже сделано до нас)

Netflix Chaos Monkey

- Давно развивается компанией Netflix
- Работает на Amazon Web Services
- Запускается в продуктивном окружении
- Про нее многие знают

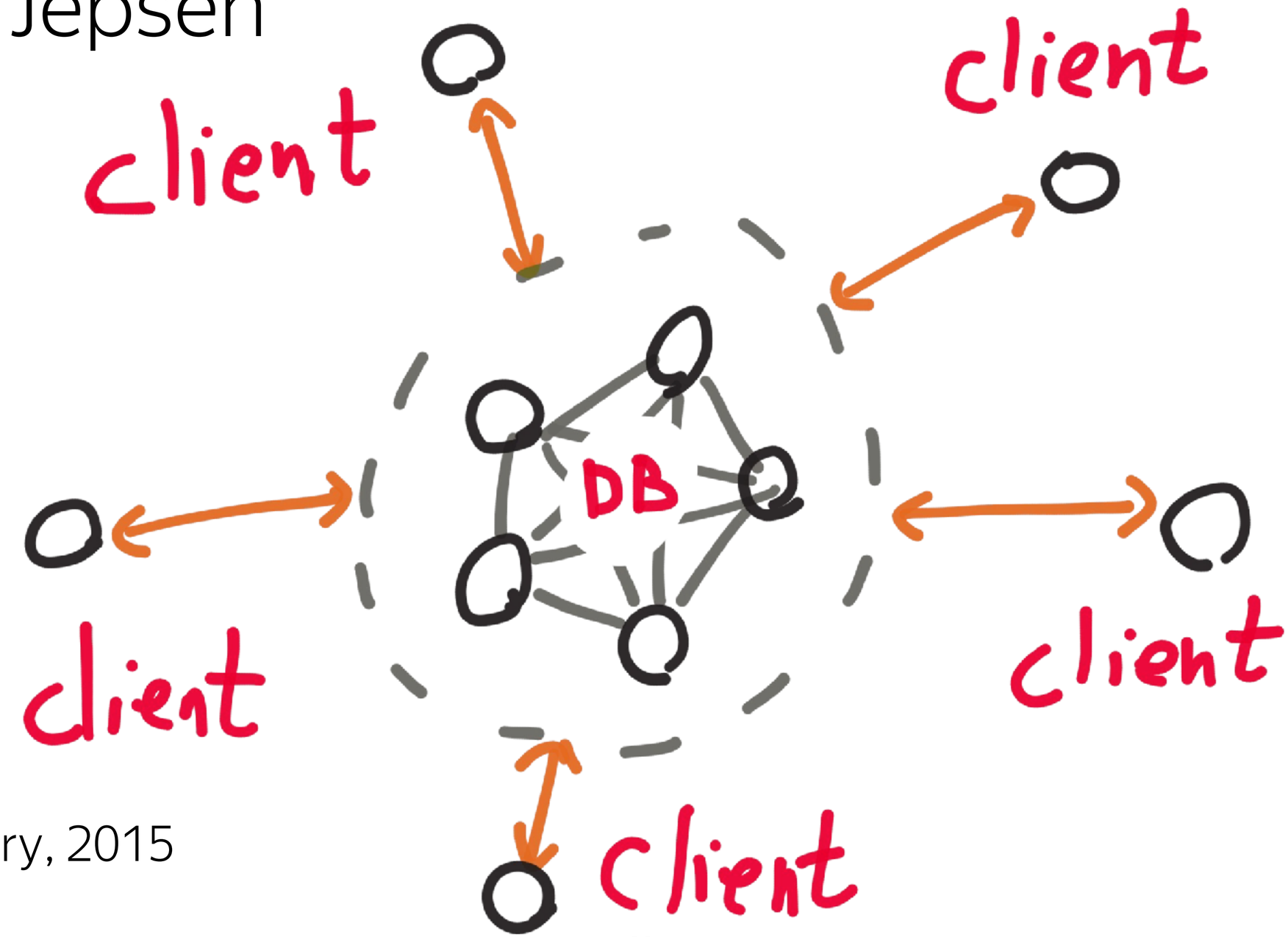
http://techblog.netflix.com/2011/07/netflix-simian-army.html
http://techblog.netflix.com/2016/10/netflix-chaos-monkey-upgraded.html

Jepsen
http://jepsen.io/



Kingsbury, 2015

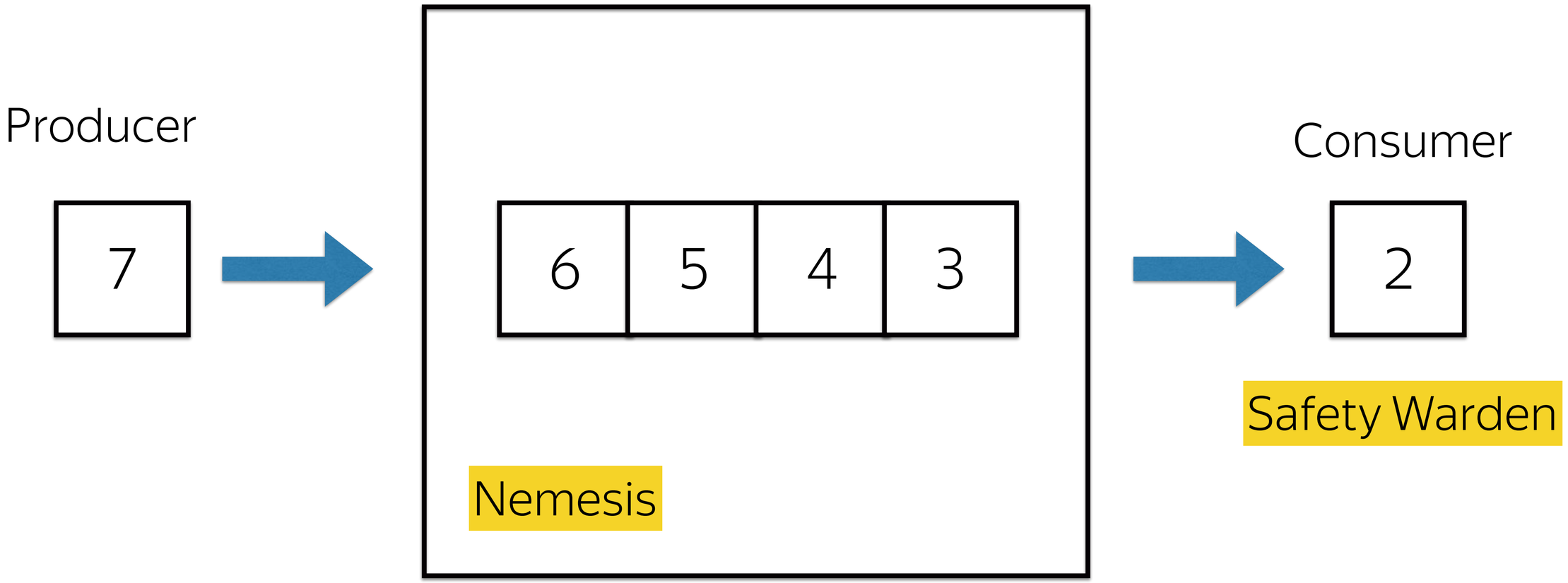
Jepsen



Kingsbury, 2015

Наш подход к тестированию

Наш подход



Producer

- Пишет заранее известный поток данных
- Соблюдает протокол взаимодействия с системой
- Соблюдает single writer principle

Consumer

- Читает данные и проверяет их корректность
- Соблюдает протокол взаимодействия с системой
- Несколько потребителей на одну пару топик + партиция

Nemesis

- Немезида — в древнегреческой мифологии богиня возмездия против тех, кто высокомерен перед богами
- У нас — инструмент для внесения разнообразных сбоев в систему (fault injection)
- Сбои могут быть внешние (black box) и внутренние (white box)

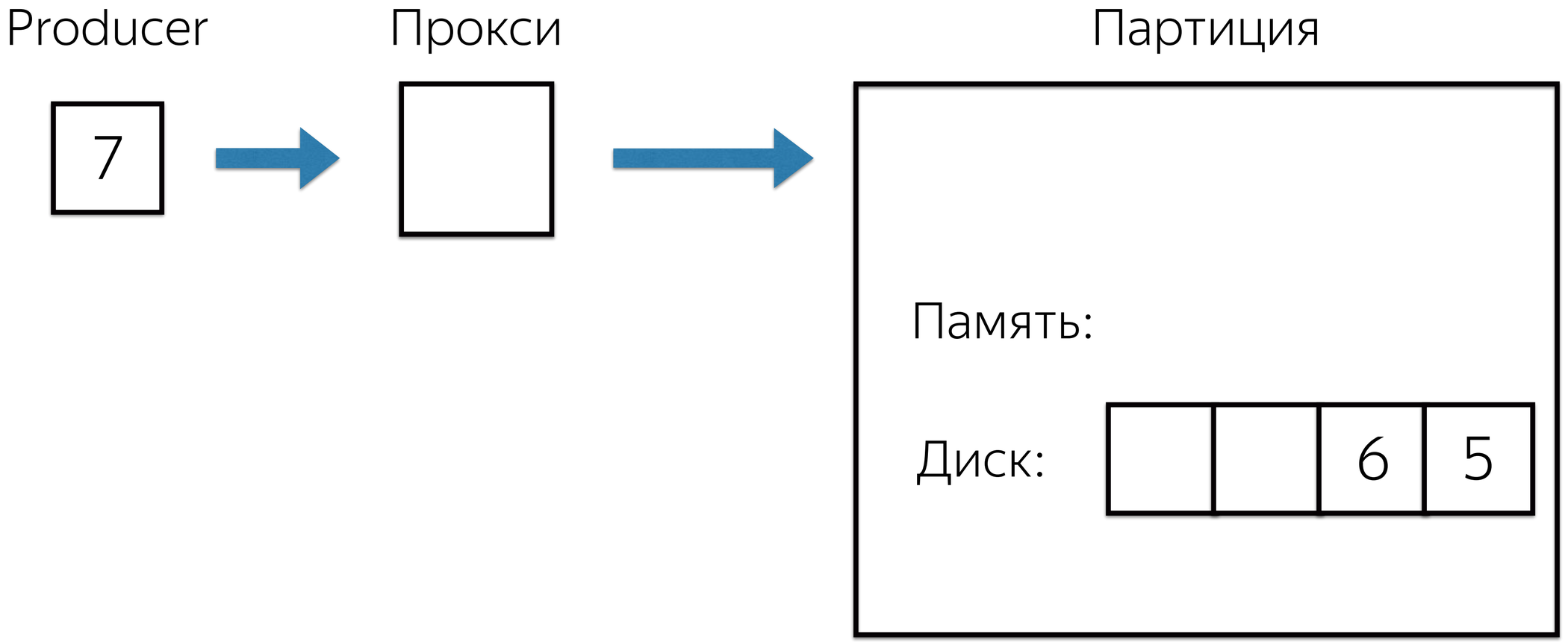
Safety Warden

Проверяет, что ничего плохого не произошло:

- Очередь соблюдает внешние инварианты и exactly once семантику
- Процесс не падает в корку
- Нет out-of-memory ошибок
- Нет критичных сообщений в логах

Баги и выводы

Потеря дублей



Потеря дублей

Producer

Прокси

Партиция

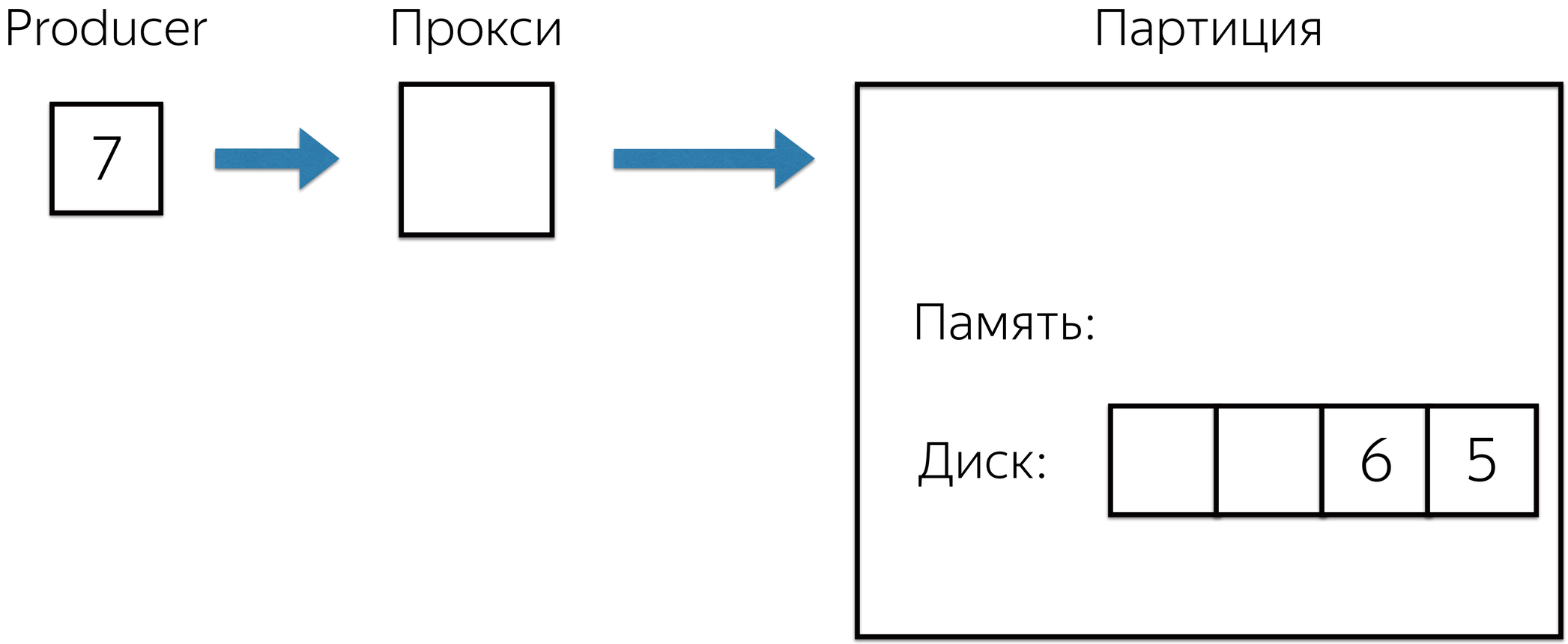
OK

Память:

Диск:

	7	6	5

Потеря дублей

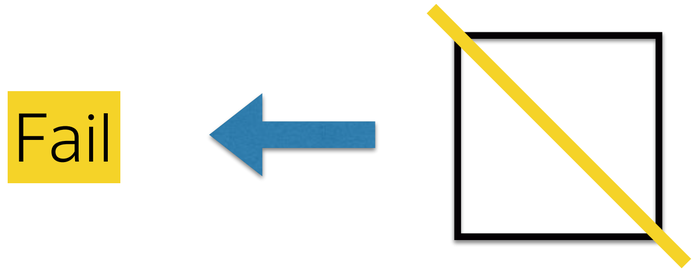


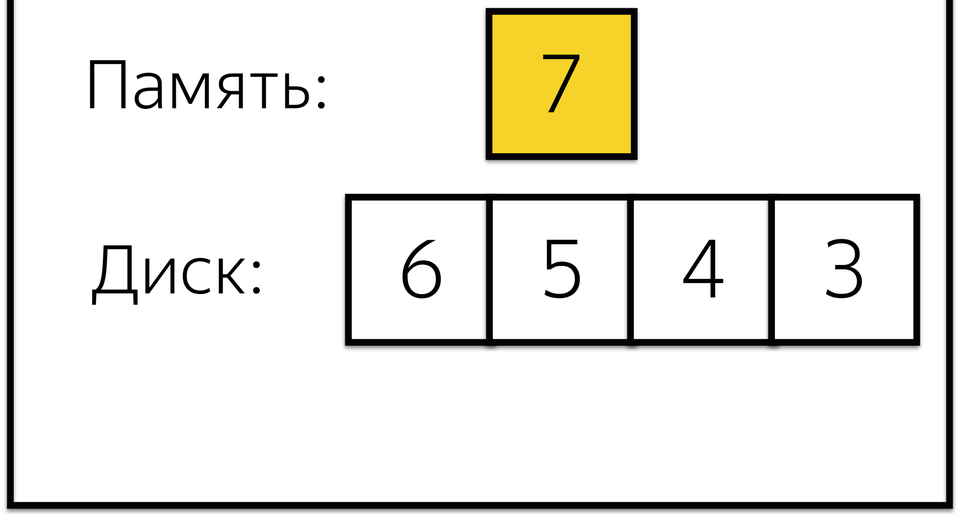
Потеря дублей

Producer

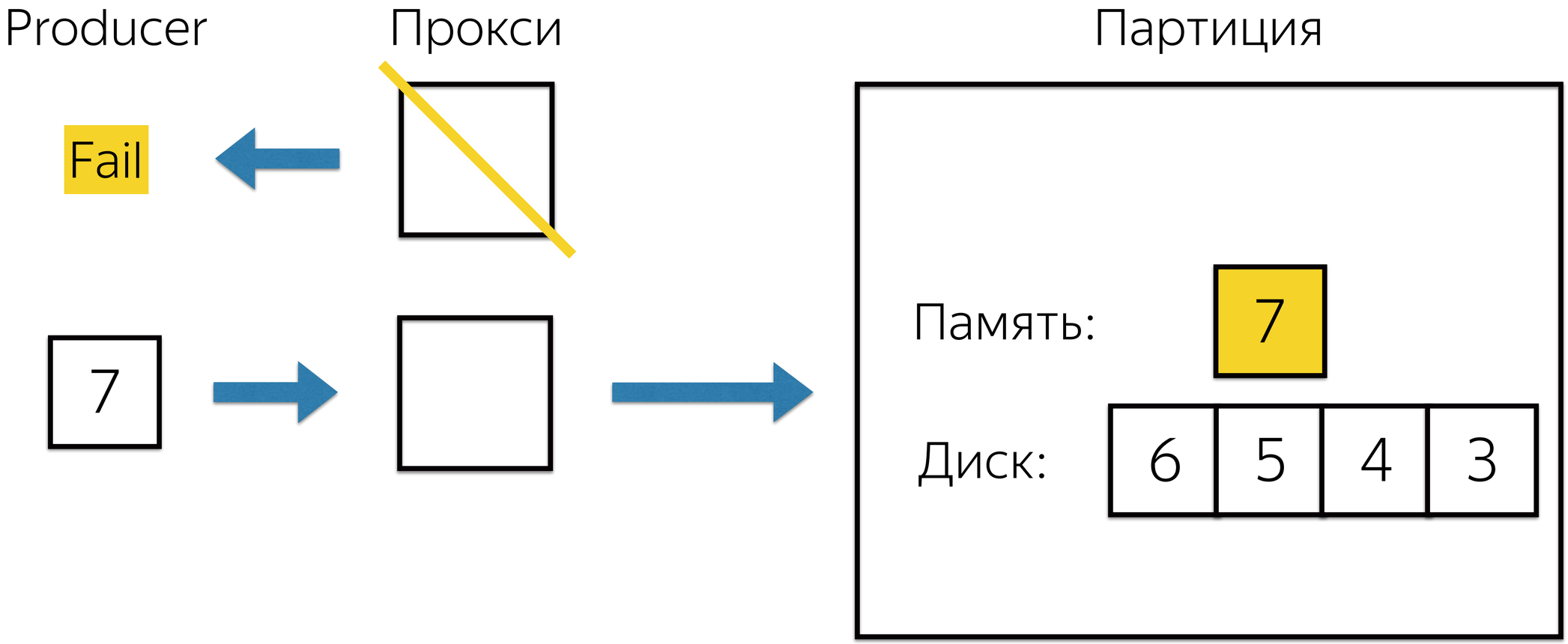
Прокси

Партиция

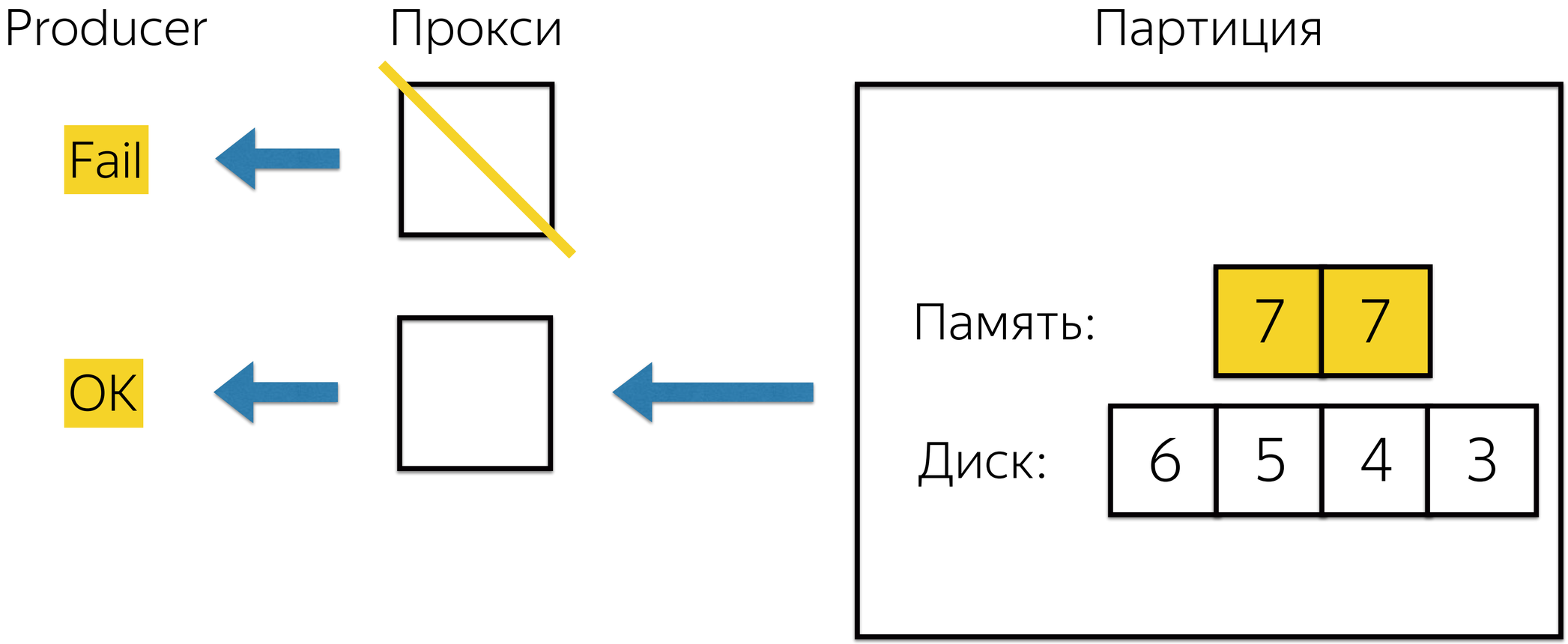




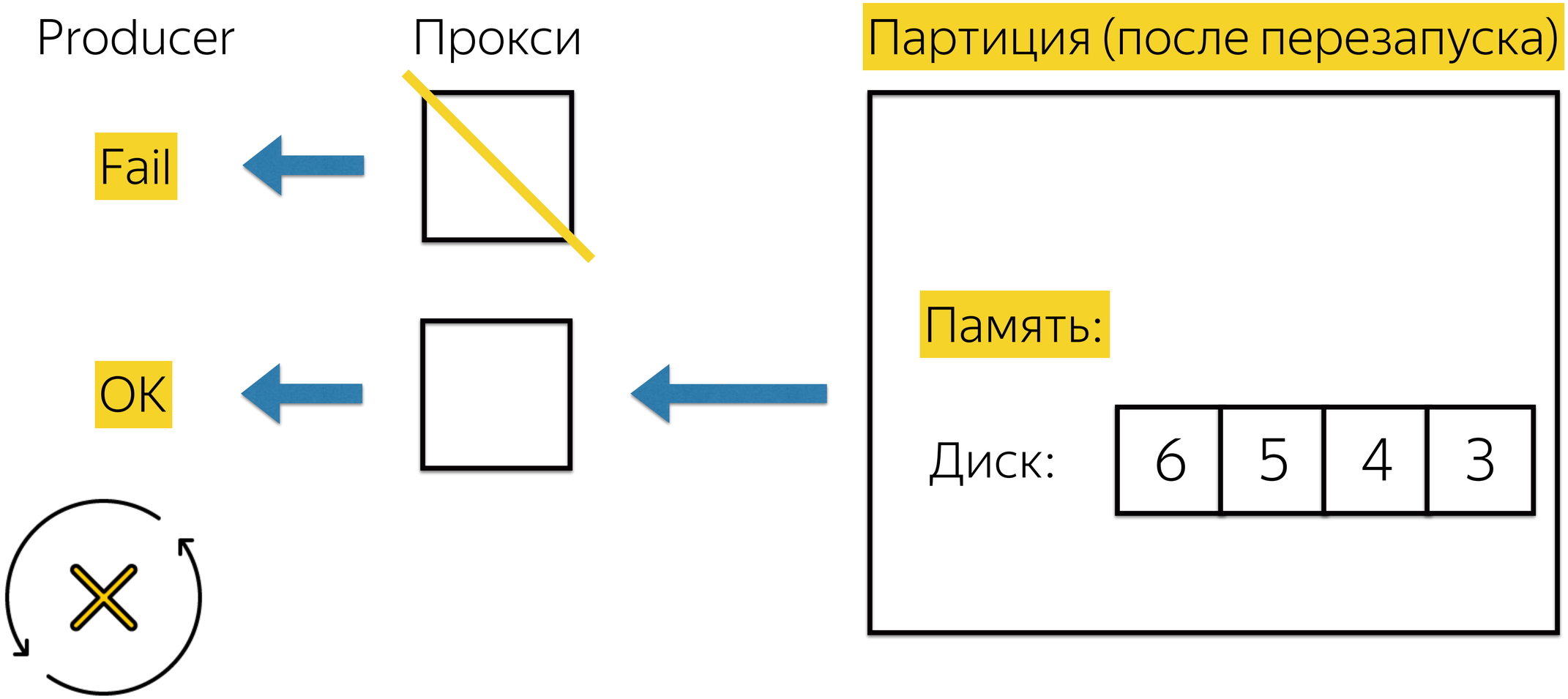
Потеря дублей



Потеря дублей



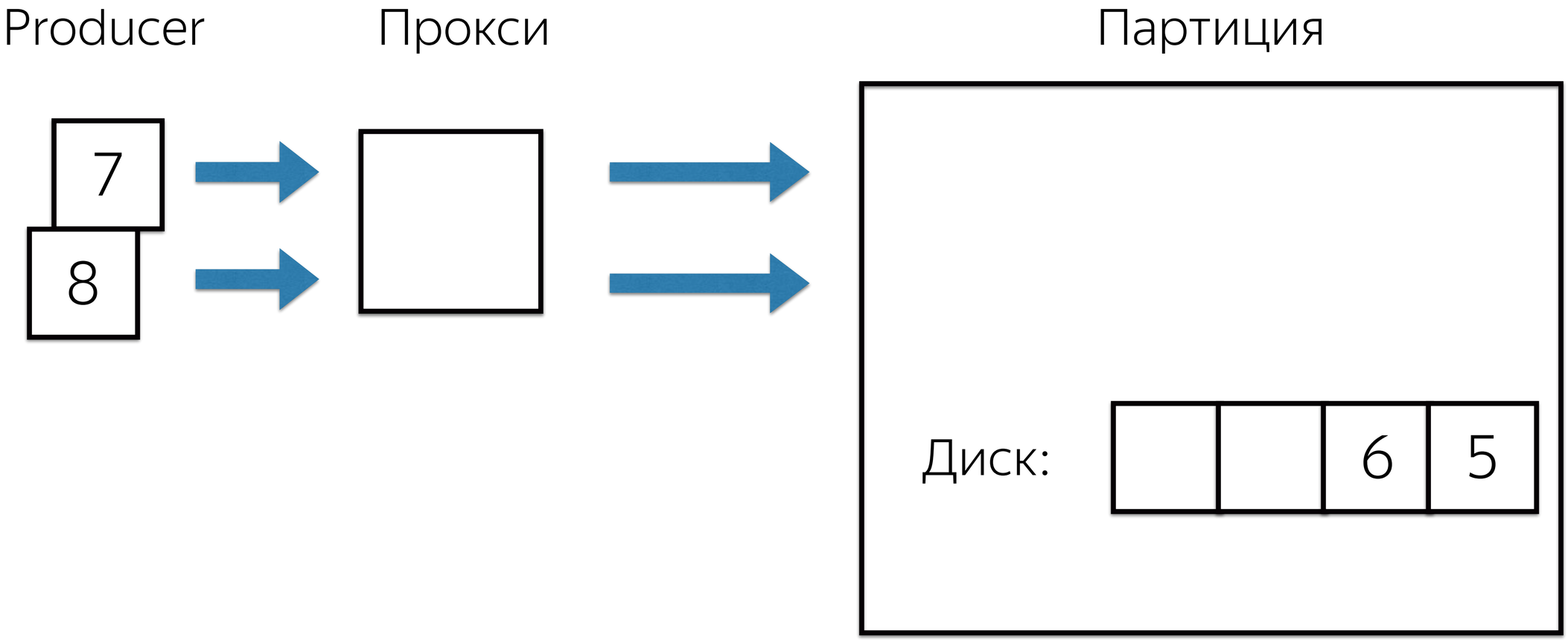
Потеря дублей



Потеря дублей: выводы

- Мы научились находить дефекты консистентности
- Данные должны попасть на диск — только тогда запись прошла успешно
- Потеря данных требует несколько скоординированных сбоев

Переупорядочивание данных

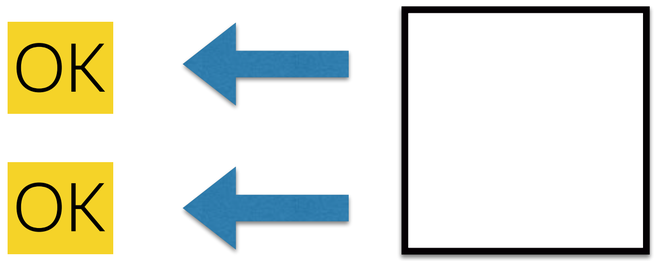


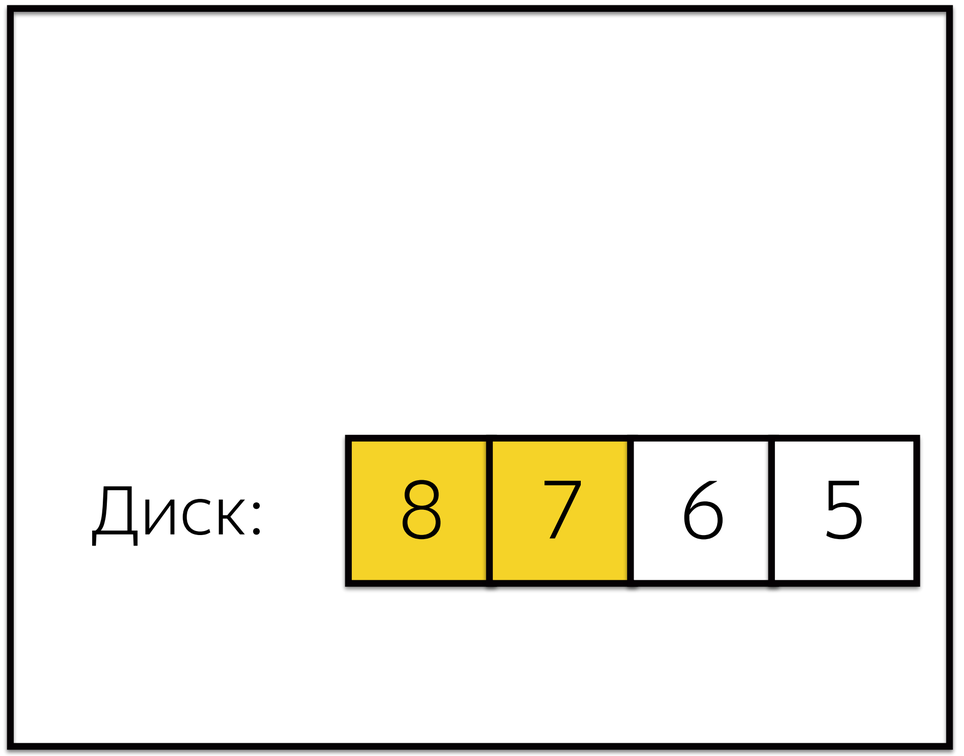
Переупорядочивание данных

Producer

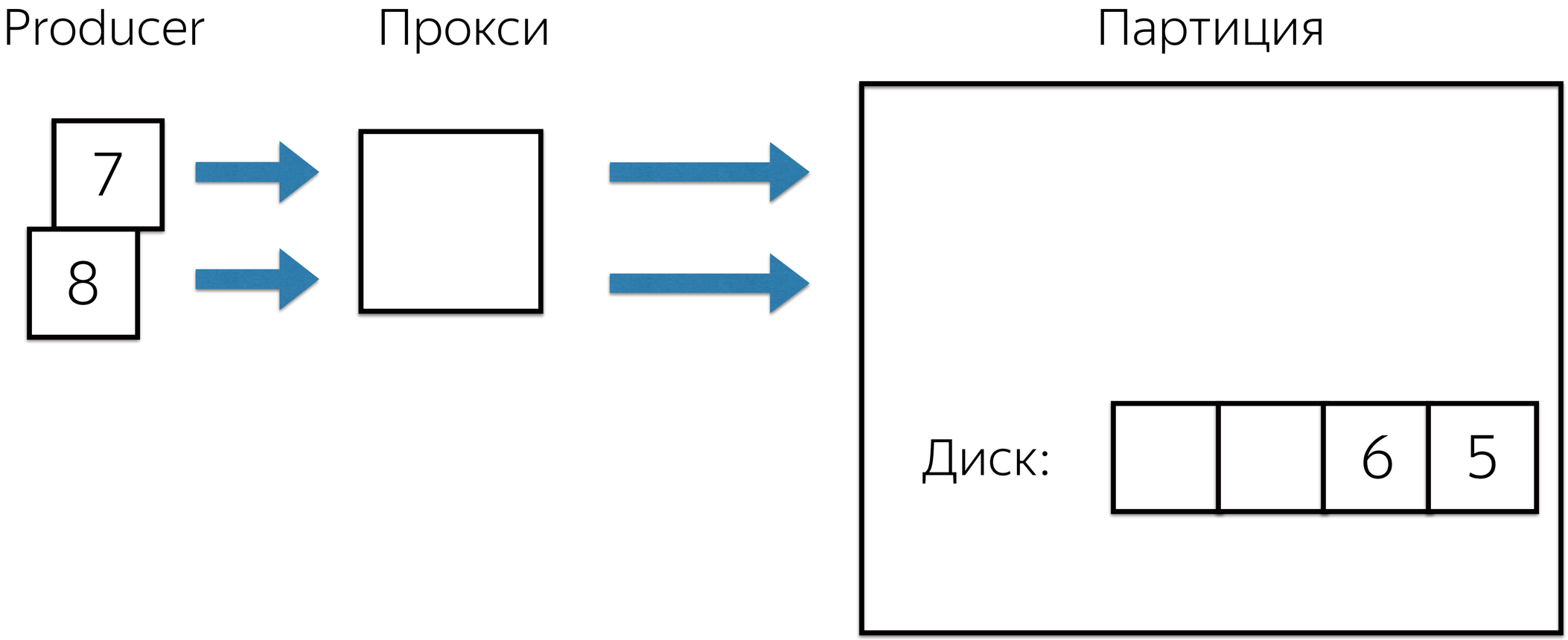
Прокси

Партиция





Переупорядочивание данных

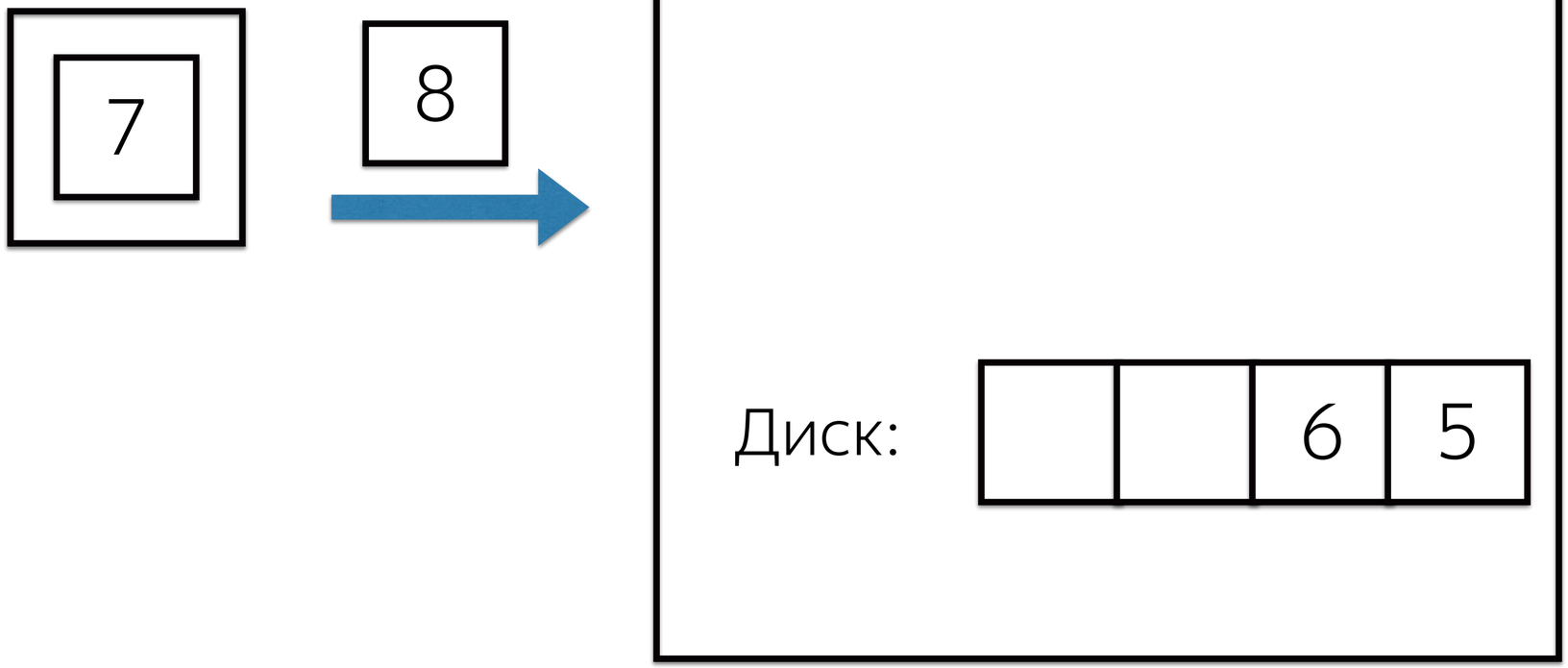


Переупорядочивание данных

Producer

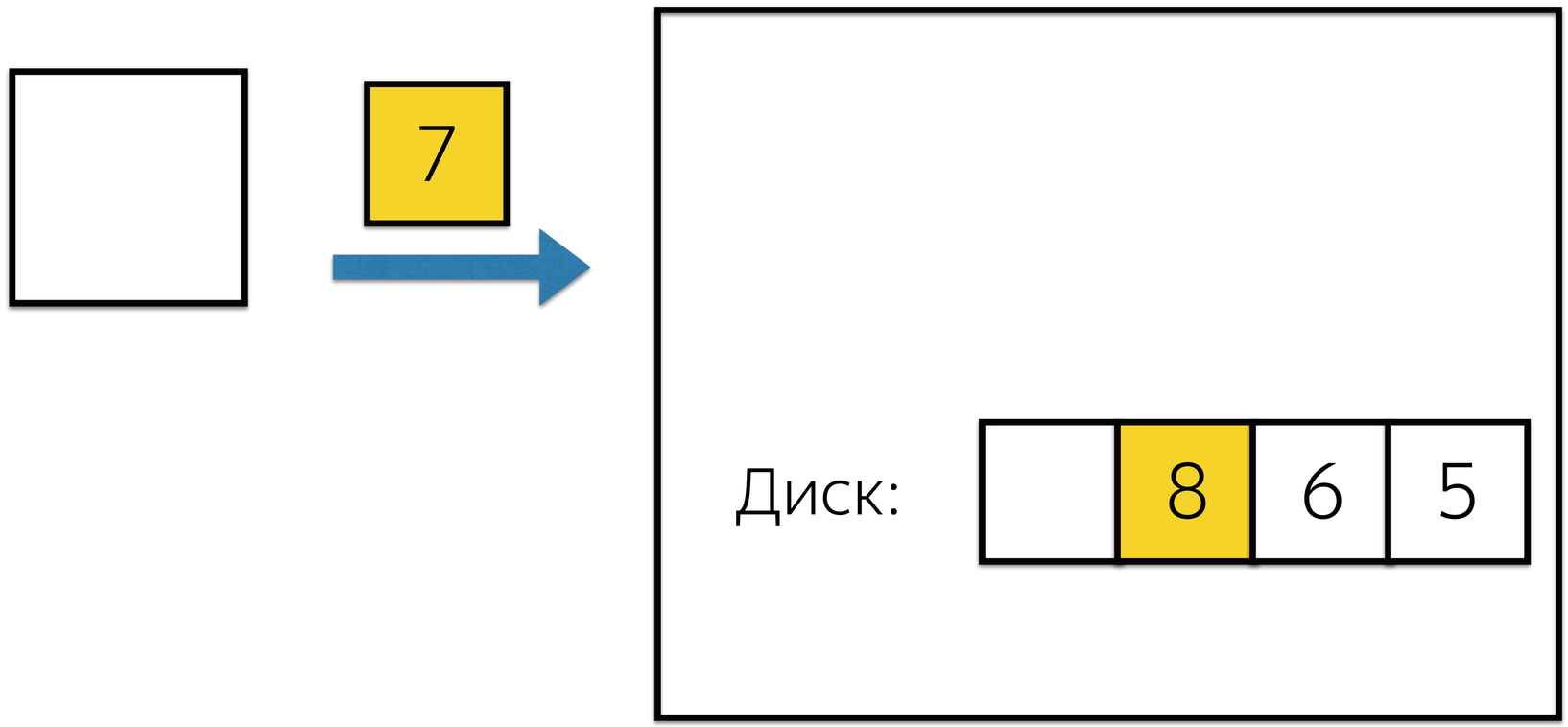
Прокси

Партиция



Переупорядочивание данных

Producer Прокси Партиция

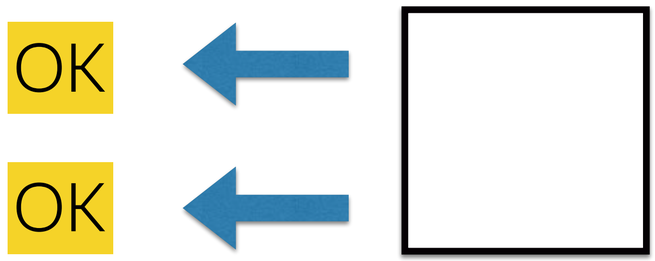


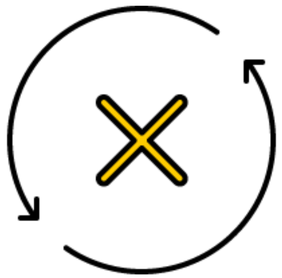
Переупорядочивание данных

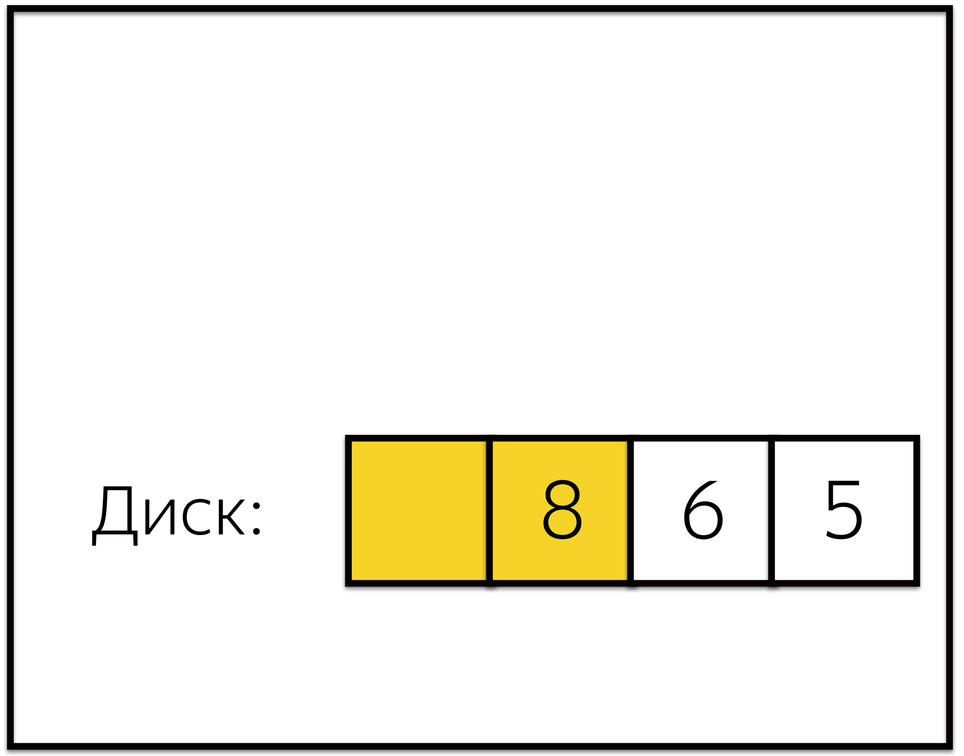
Producer

Прокси

Партиция



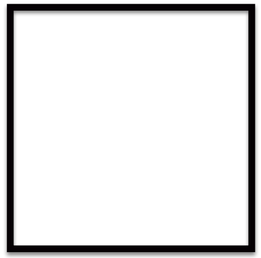




Переупорядочивание данных: выводы

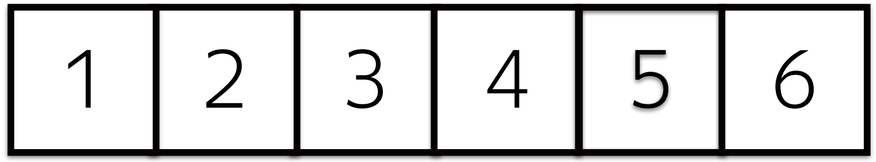
- Проблема в клиентском протоколе — он недостаточно жесткий
- «Хороший» клиент, никогда не получит эту багу
- Мы поменяли протокол — добавили упорядоченный внутри сессии номер записи

История о потерянном логе



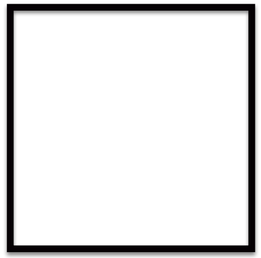
Партиция

Распределенное хранилище



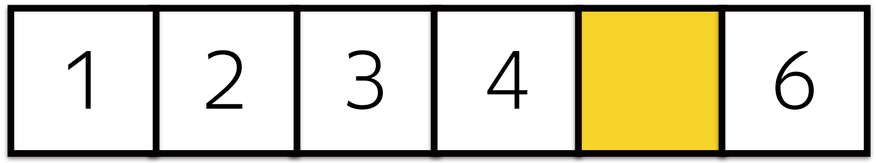
Лог транзакций

История о потерянном логе



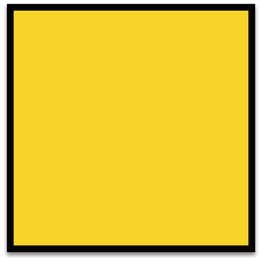
Партиция

Распределенное хранилище



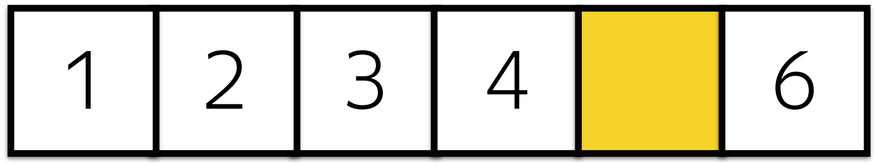
Лог транзакций

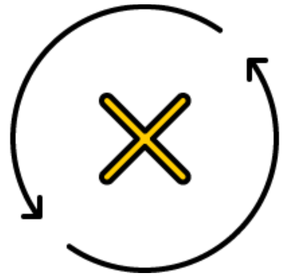
История о потерянном логе



Партиция
Статус — запускается

Распределенное хранилище





Лог транзакций

История о потерянном логе: выводы

- Есть класс дефектов, которые проявляются как **потеря доступности**
- Дефекты доступности невозможно обнаружить через нарушение инвариантов
- Доступность системы — важная характеристика для реальных систем

Safety и Liveness

Safety — ничего плохого не происходит
Liveness — в конце концов произойдет что-то хорошее

Все свойства системы можно описать как комбинацию safety + liveness свойств

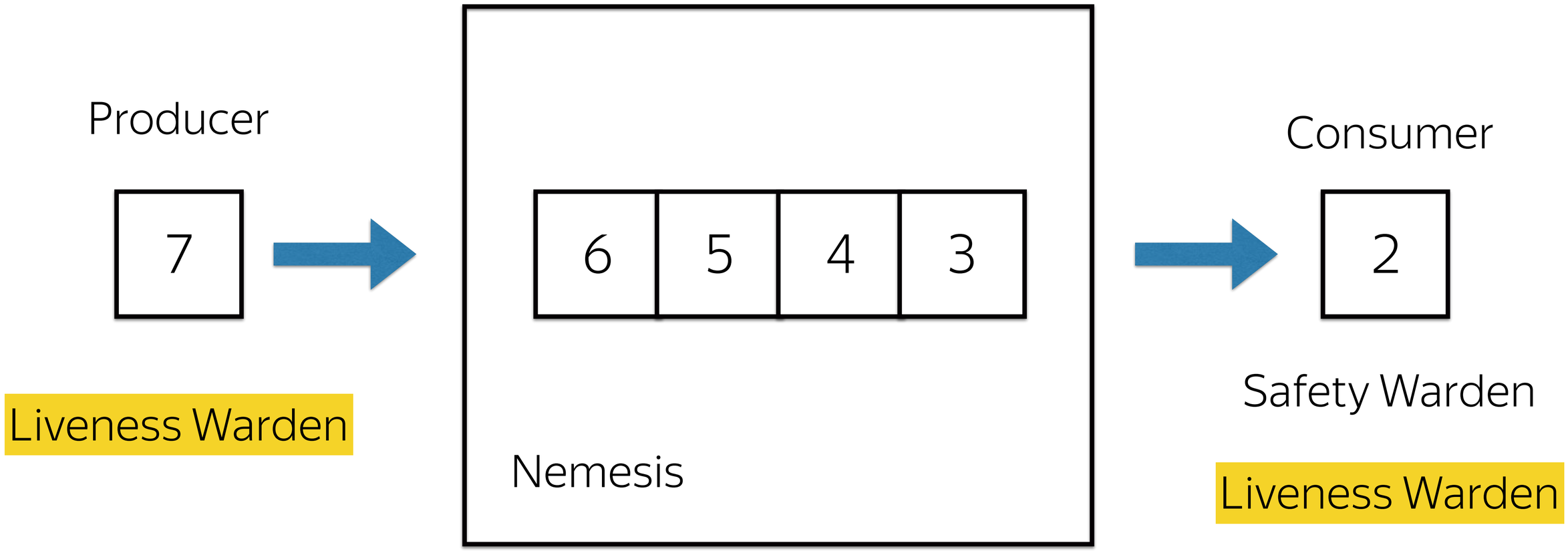
Почему сложно проверять Liveness

- Свойство liveness проявляется только на бесконечной истории событий в системе (в конце концов произойдет что-то хорошее)
- «Impossibility of Distributed Consensus with **One Faulty Process**» Fisher, Lynch, Paterson (1985) aka «FLP result»
  http://the-paper-trail.org/blog/a-brief-tour-of-flp-impossibility/
- «FLP proves that any fault-tolerant algorithm solving consensus has runs that never terminate»
  http://www.cs.cornell.edu/courses/CS5412/2016sp/slides/XII%20-%20Consensus%20and%20FLP.pdf

Как находить liveness дефекты системы?

- Какими liveness свойствами должна обладать система?
- Как на практике описать свойства liveness для нашей очереди?
- Какими способами можно обнаружить нарушение этих свойств?

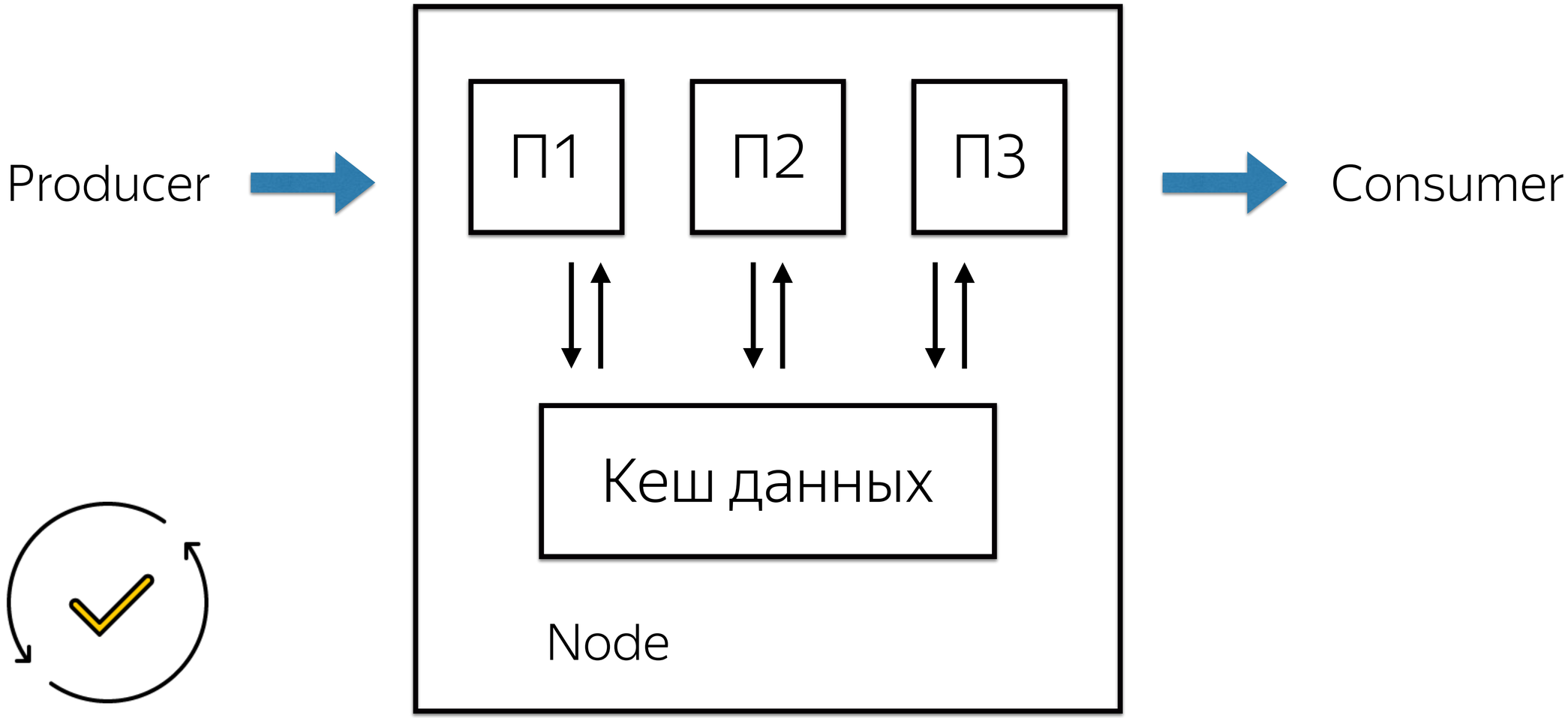
Наш подход + Liveness Warden



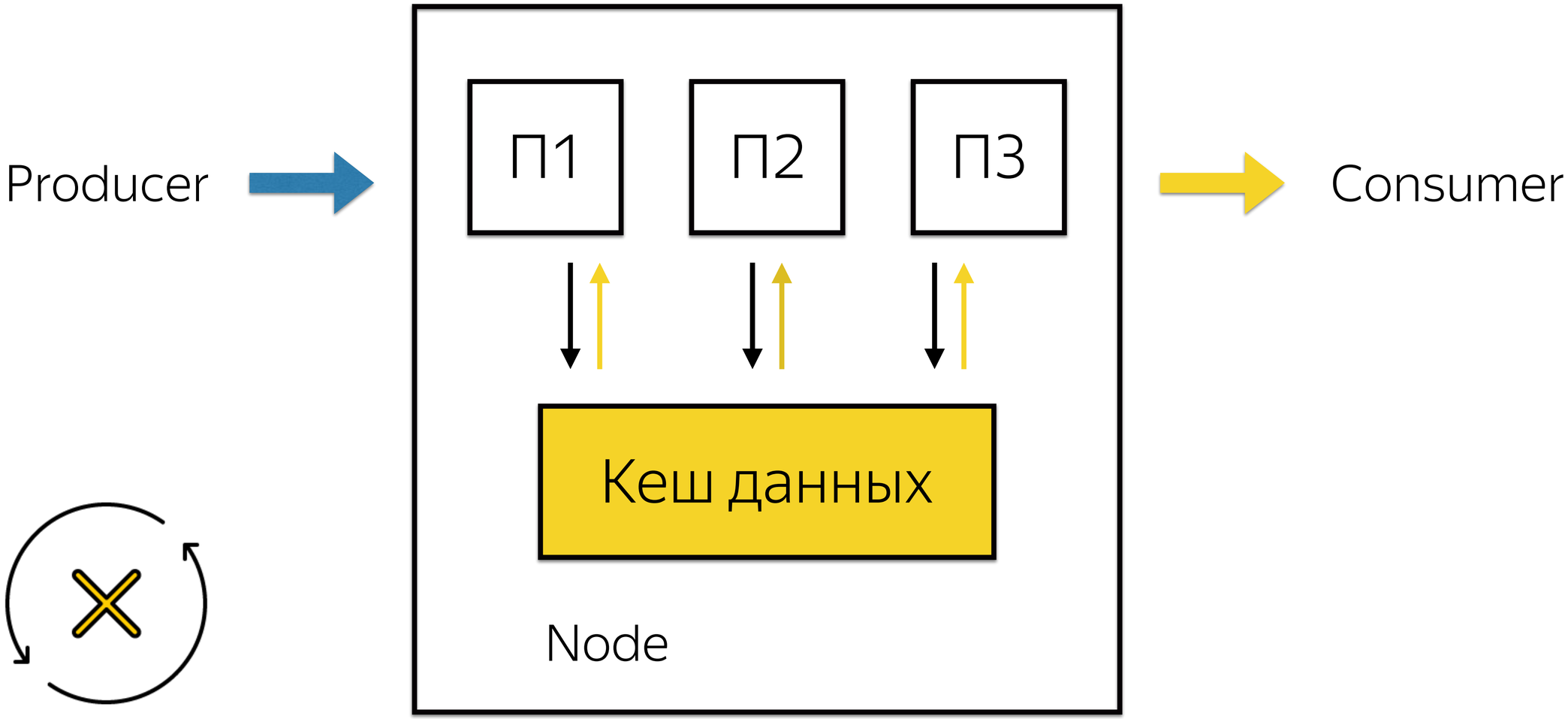
Liveness Warden

- Сложно проверять safety и liveness одновременно
- Поэтому мы останавливаем Nemesis, на время проверки liveness
- Проверяем идет ли **прогресс** записей/чтений (Producer/Consumer)
- Если прогресса нет — liveness ошибка

О залипшем кеше



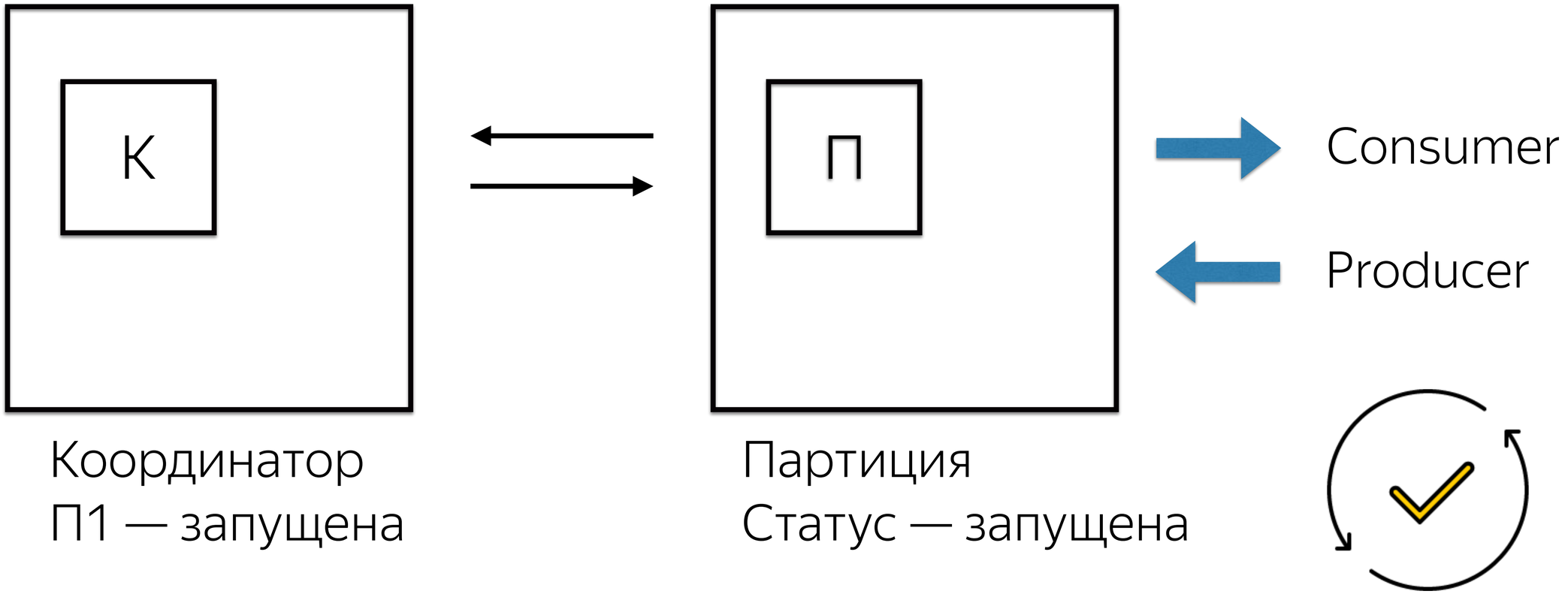
О залипшем кеше



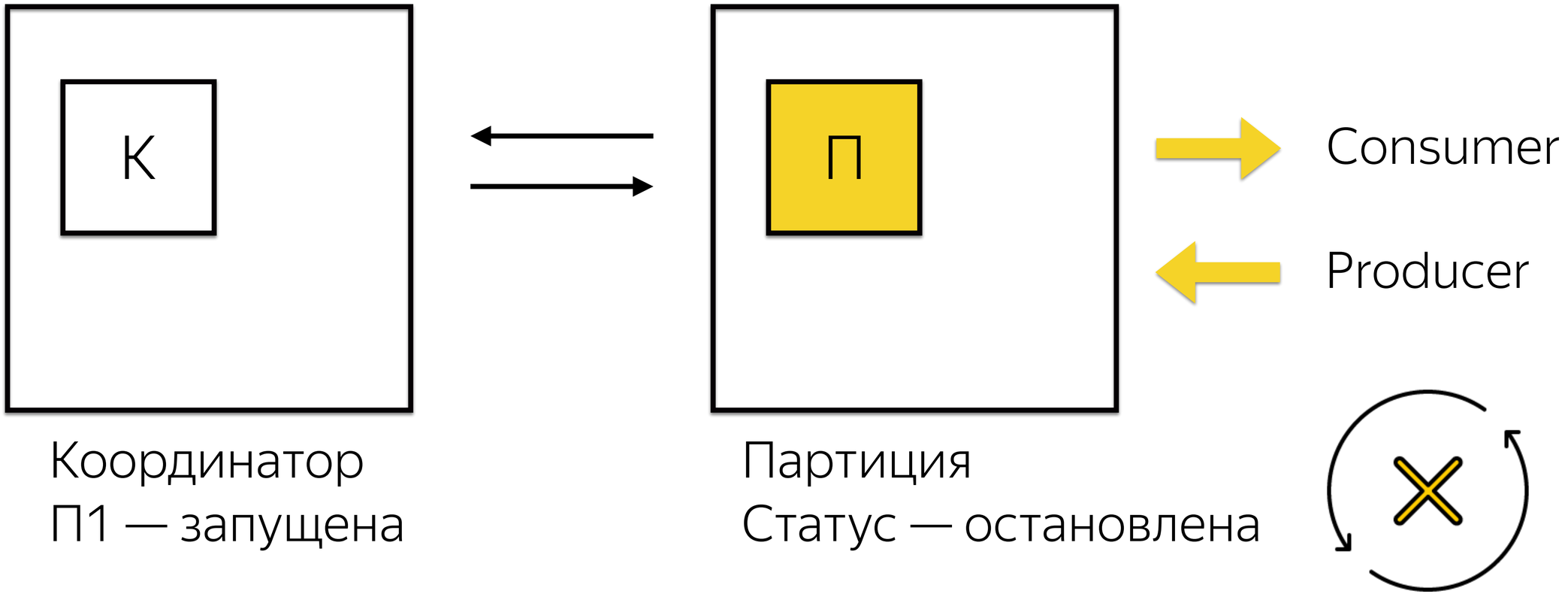
О залипшем кеше: выводы

- При определенных сбоях на ноде залипал кеш и чтения всегда возвращали ошибку. Записи при этом успешно проходили
- Оптимизации производительности это хорошо, если они не нарушают гарантий системы
- Новый компонент — новые баги
- Возможна частичная потеря доступности

Забывчивый координатор



Забывчивый координатор



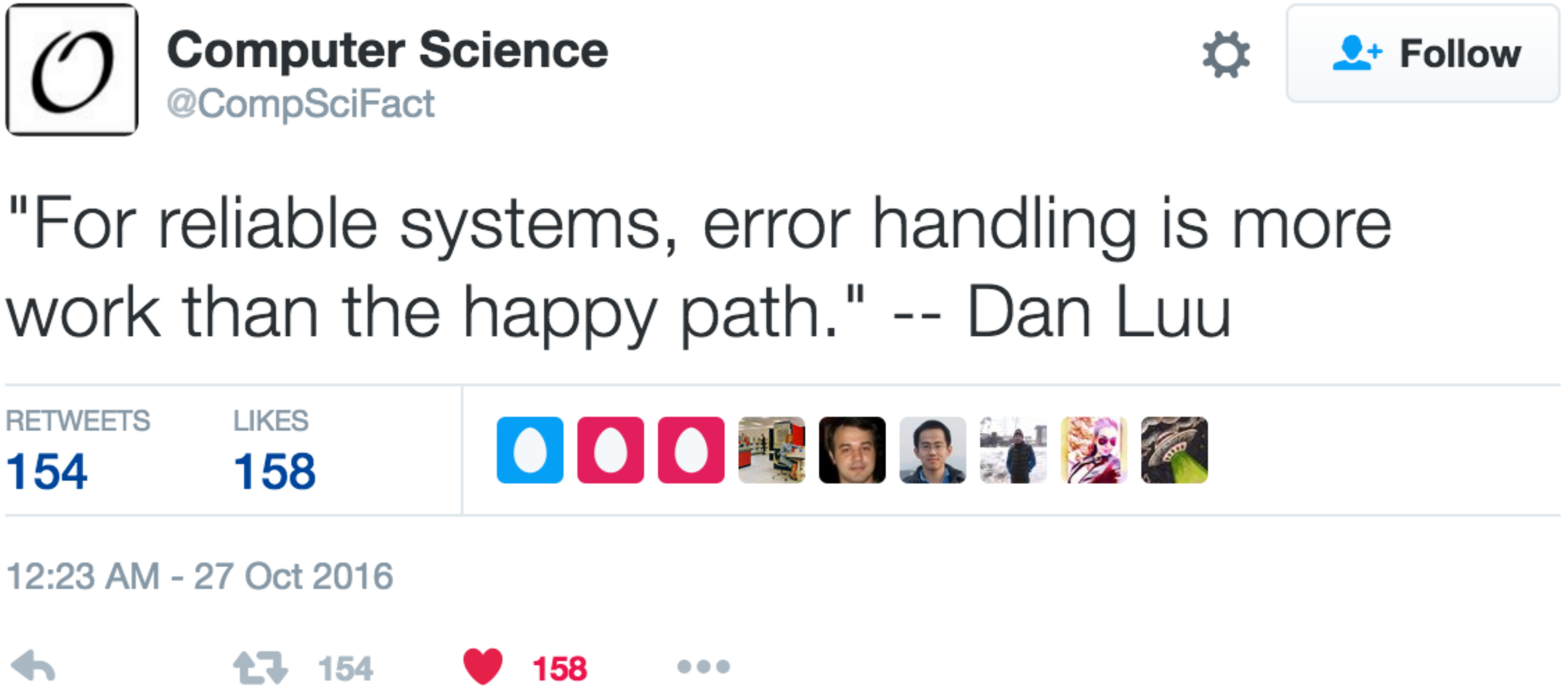
Координатор
П1 — запущена

Партиция
Статус — остановлена

Забывчивый координатор: выводы

- При определенной комбинации сбоев координатор запуска партиций считал, что партиция запущена, но она была остановлена
- Сложно обнаружить проблему, потому что перезапуск партиции или ноды устранял «залипание»
- Главная проблема — полная недоступность партиции

Заключение



https://twitter.com/CompSciFact/status/791389830420762624

Выводы

- Будь готов к сбоям — они неизбежно будут происходить
- Изучай теорию — это помогает на практике
- Знай свои инварианты — они описывают систему
- Помни про liveness и доступность — эти свойства делают систему полезной

Контакты:

Андрей Сатарин

Ведущий инженер по автоматизации тестирования

 asatarin@yandex-team.ru

 https://twitter.com/asatarin

Ссылки

- Testing Distributed Systems
  https://asatarin.github.io/testing-distributed-systems/

- Simple Testing Can Prevent Most Critical Failures
  https://www.usenix.org/conference/osdi14/technical-sessions/presentation/yuan

- Яндекс изнутри: инфраструктура хранения и обработки данных
  https://events.yandex.ru/events/meetings/15-oct-2016/

- Презентации Kyle Kingsbury
  http://jepsen.io/talks.html