Пишем собственный менеджер блокировок

Сергей Егоров

Parallels™

Profit from the Cloud

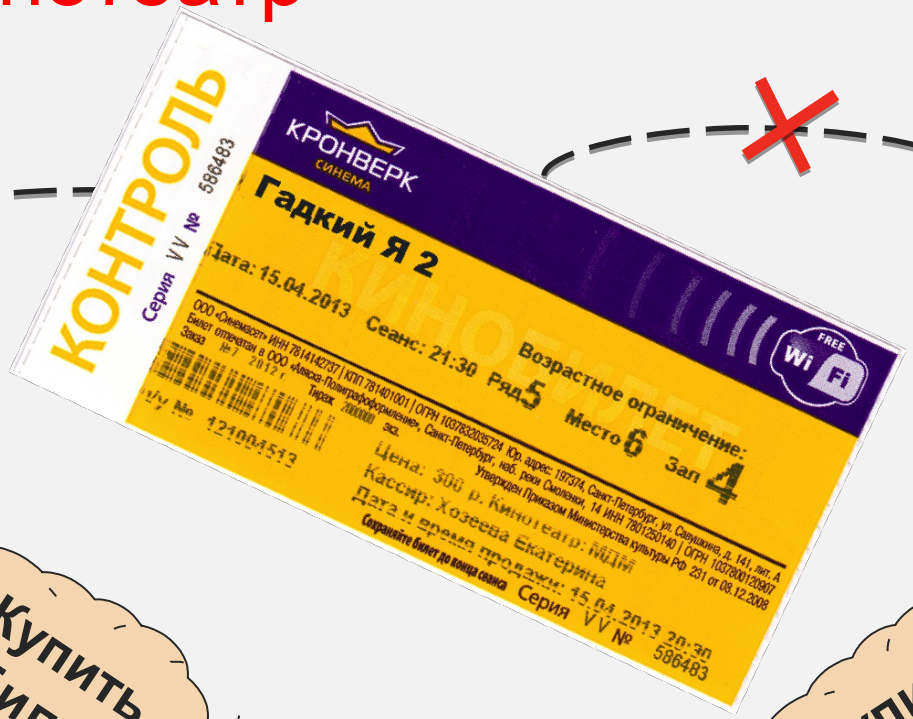
24 October 2013

Задача:

1 000 000

. . .

Покупаем билет в кинотеатр



Купить Билет

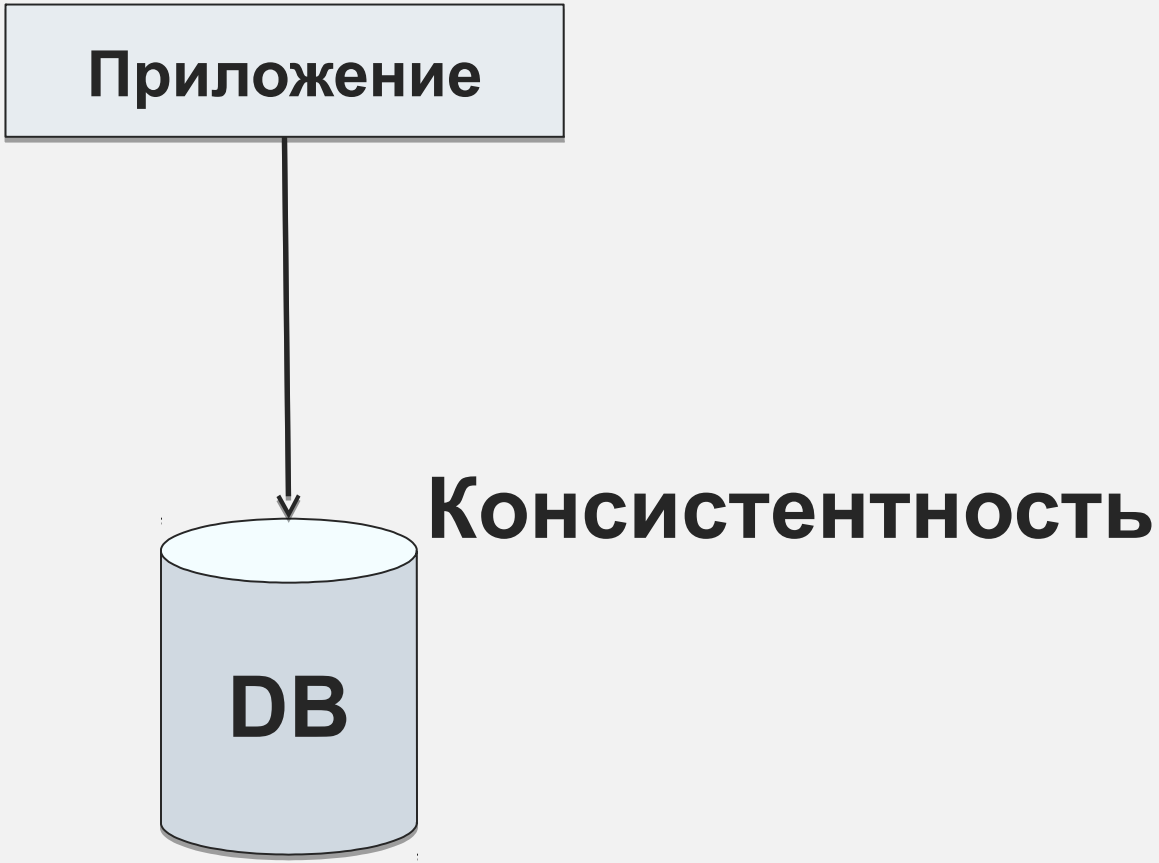
Купить Билет

Продан

Уже продан

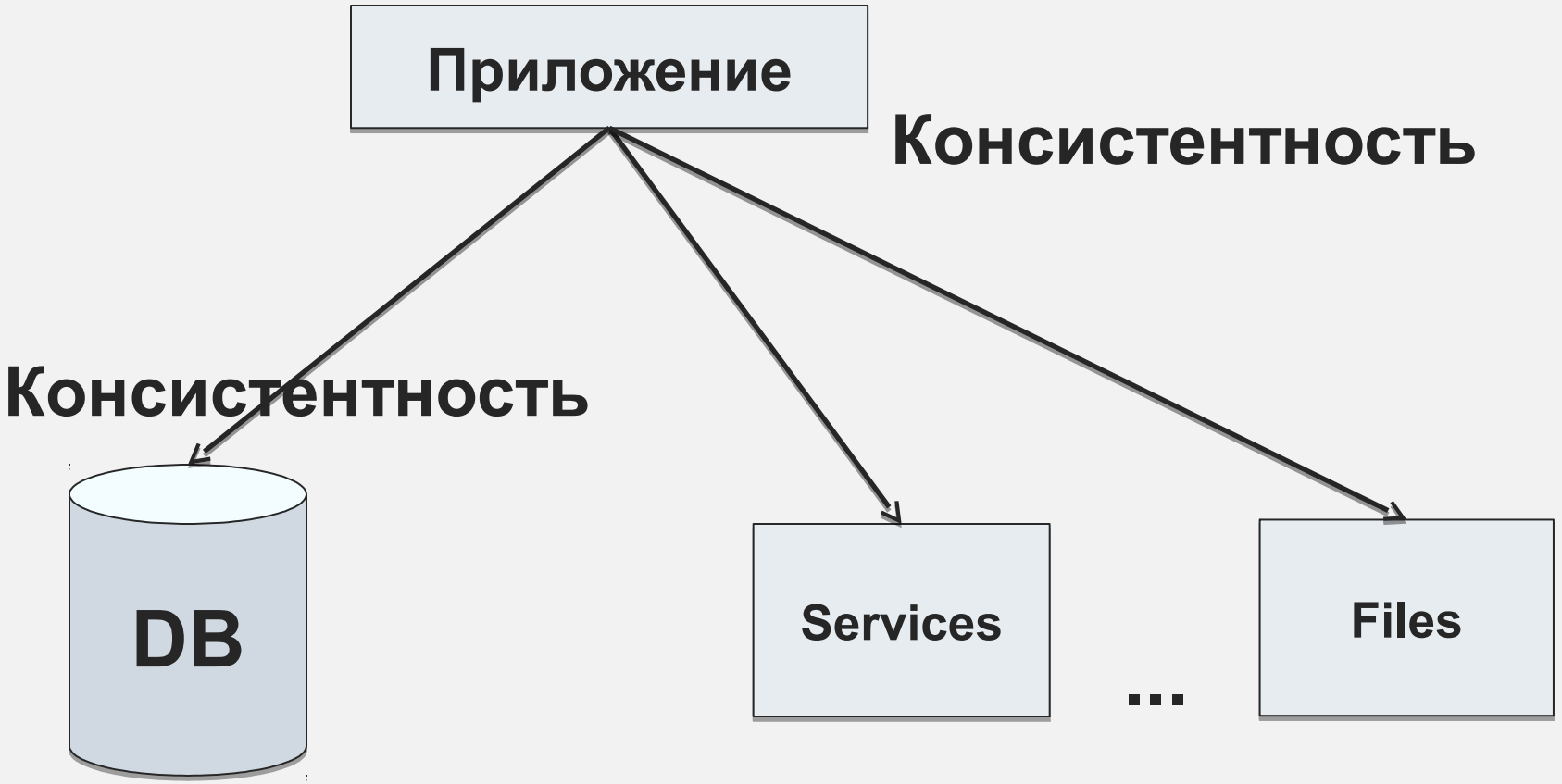
База Билетов

Консистентность



Идем смотреть кино …

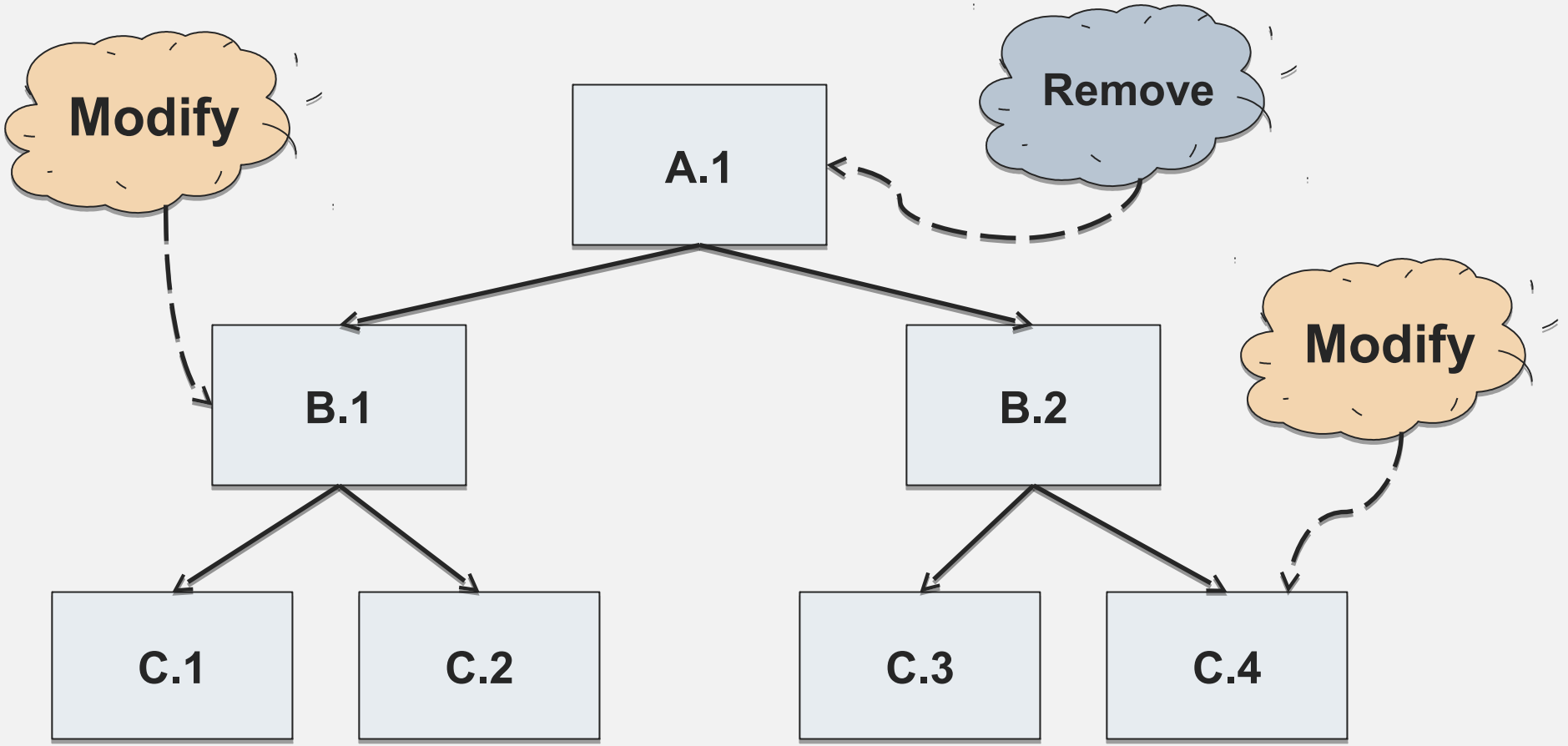
Все таже консистентность



Блокировки (системные)

Блокировки прикладного уровня

Какие сценарии должны работать?



Какой должен быть Lock Manager?

- **Не требовательный по памяти**
  (Необходимо для работы в рамках одного сервера)
- **Работающий в распределенной среде**
- **Надежный**
- **Быстрый**

В MySQL тоже есть Advisory Locks …

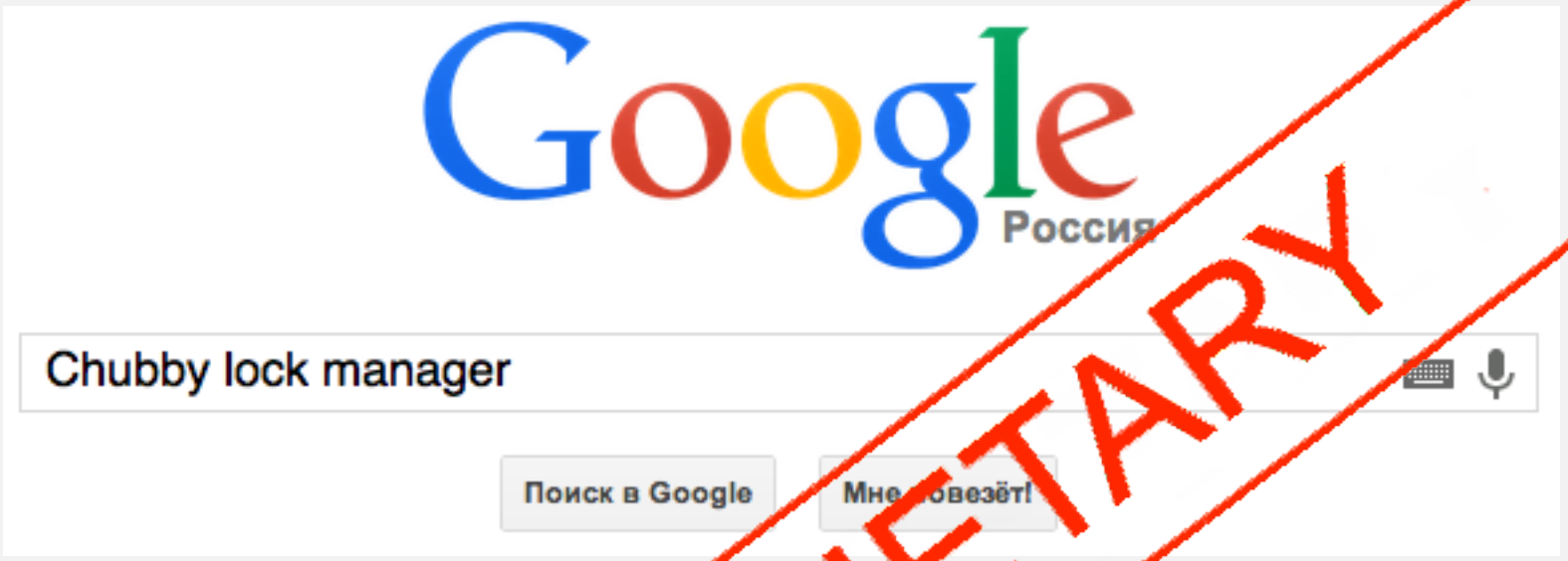
```
#1) mysql> SELECT GET_LOCK('lock1',10);
       -> 1
#2) mysql> SELECT IS_FREE_LOCK('lock2');
       -> 1
#3) mysql> SELECT GET_LOCK('lock2',10);
       -> 1
#4) mysql> SELECT RELEASE_LOCK('lock2');
       -> 1 (not NULL)
#5) mysql> SELECT RELEASE_LOCK('lock1');
       -> NULL
```

MySQL не позволяет вложенных advisory locks

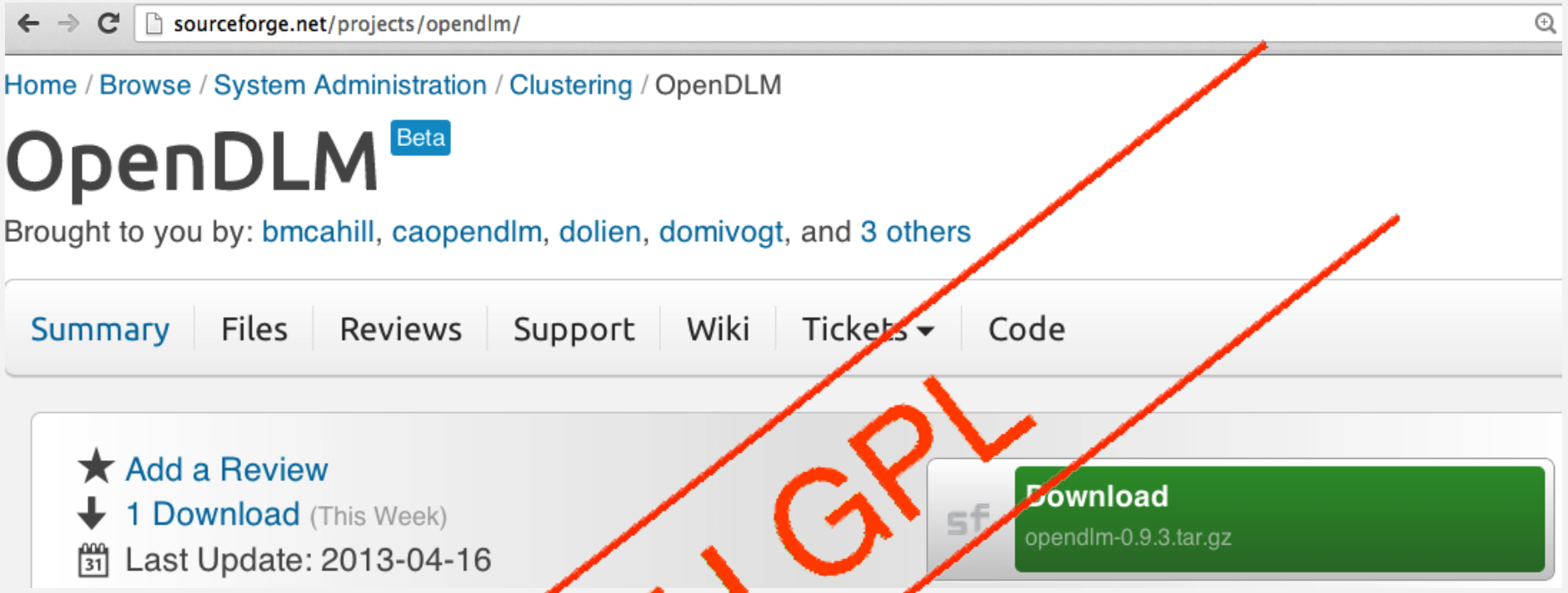
Apache ZooKeeper™

- Распределенный, с выделенным лидером
- Гарантированный порядок блокировок
- Ориентирован более быстрое чтение (10:1)

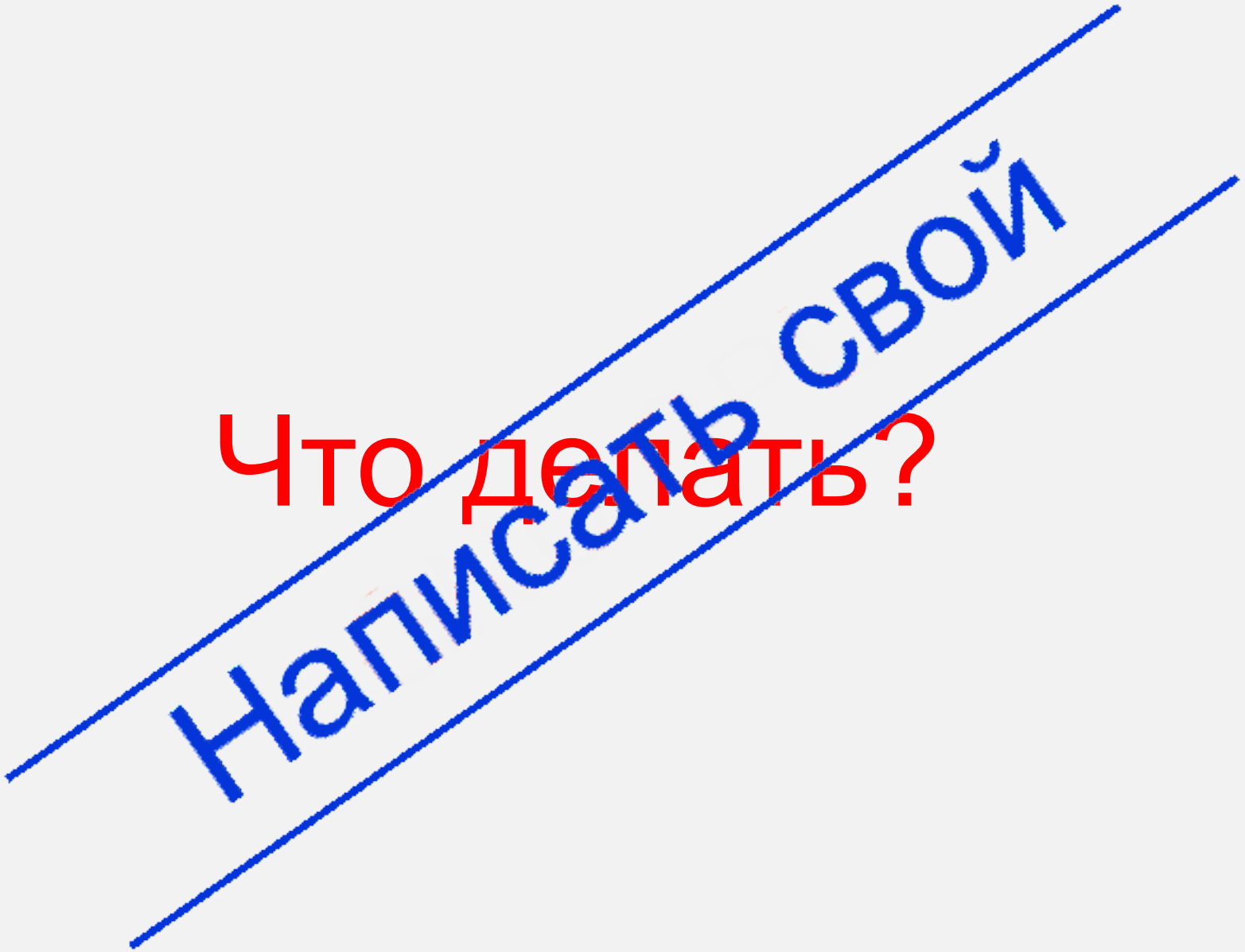
JAVA



- Распределенный
- Ориентирована на слабо-связанные распределенные системы
- Фокус на надежный (более надежный чем быстрый)
- C++

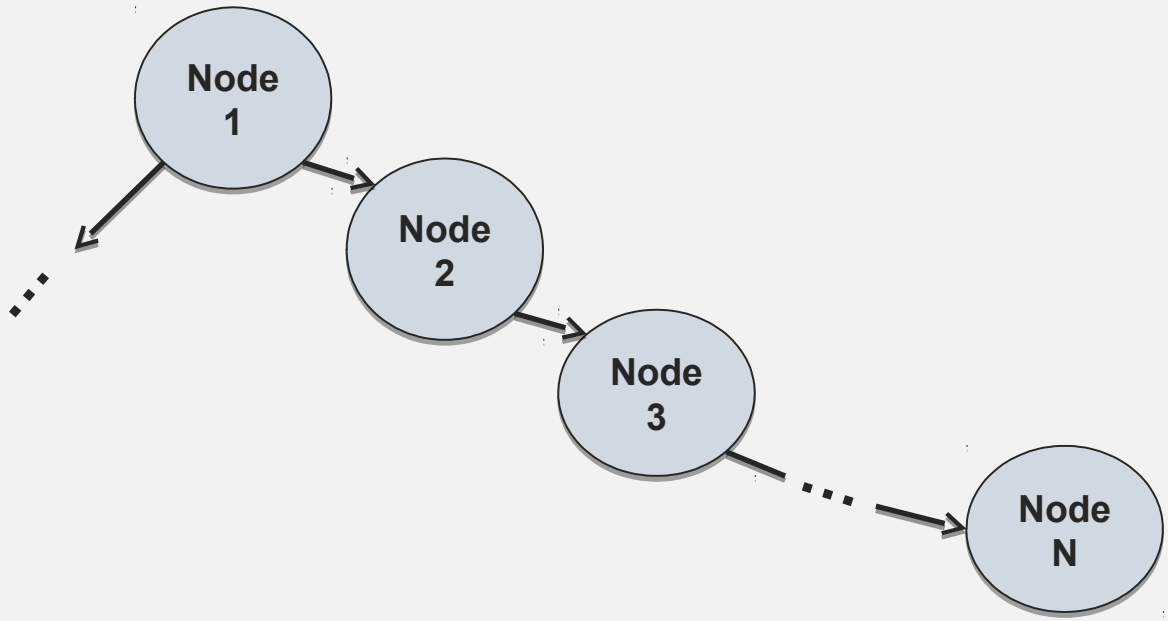


- **Распределенный**
- **Нет выделенной центральной ноды**
- **Эффективное взаимодействие между нодами**
- **Не гарантированный порядок**
- **C++**


Что делать?
Написать свой

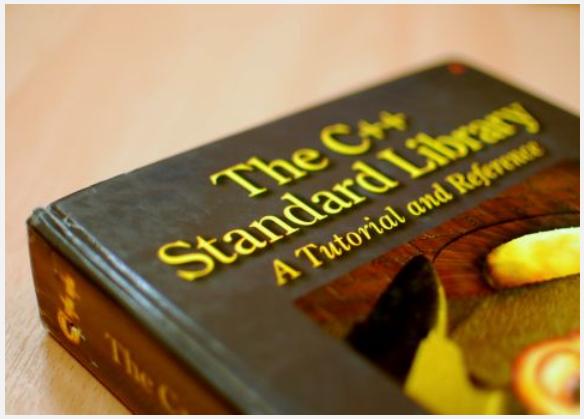
Как это выглядит в коде?

- Lock(WRITE, “/node1/node2/node3/…/nodeN”);
- Unlock(“/node1/node2/node3/…/nodeN”);



Как может использоваться?

- Как библиотека



- Как сервис

- Еще пример как php extension

Конфигурации использования

- В рамках одного сервера

- В распределенная сетевая среда

Что под капотом?

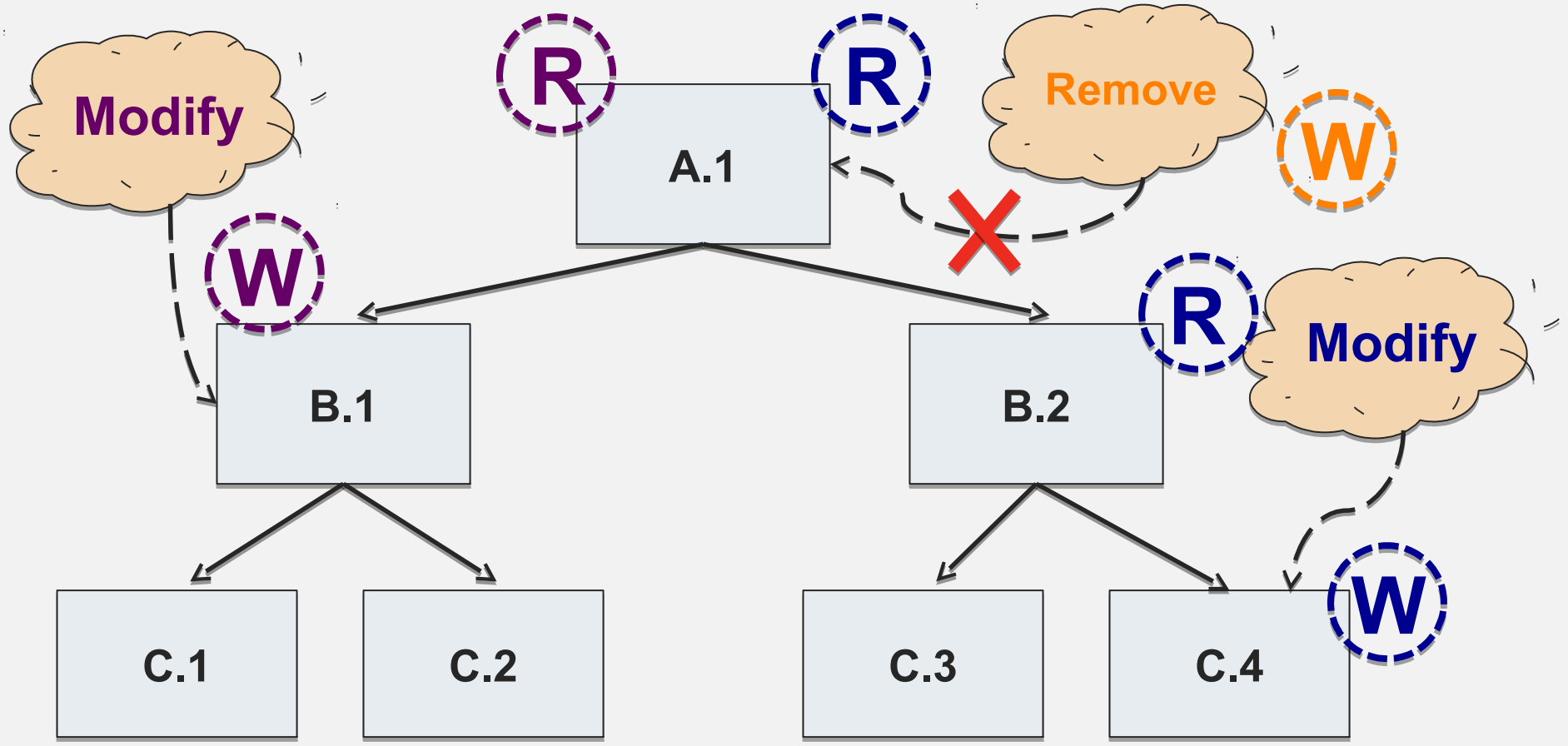
- C++
- Авторизационный модуль
- Shared memory в качестве хранилища блокировок
- Препроцессор (гарантированный порядок блокировок)
- Режимы блокировок: Shared Read, Exclusive Write, Concurrent Read, Concurrent Write

Как работает снаружи?

lock(WRITE, “/A.1/B.2/C.4”);

lock(WRITE, “/A.1”);

lock(WRITE, “/A.1/B.1”);



Как работает под капотом?

Уровень приложения:

lock(WRITE, “/A.1/B.2/C.4”);

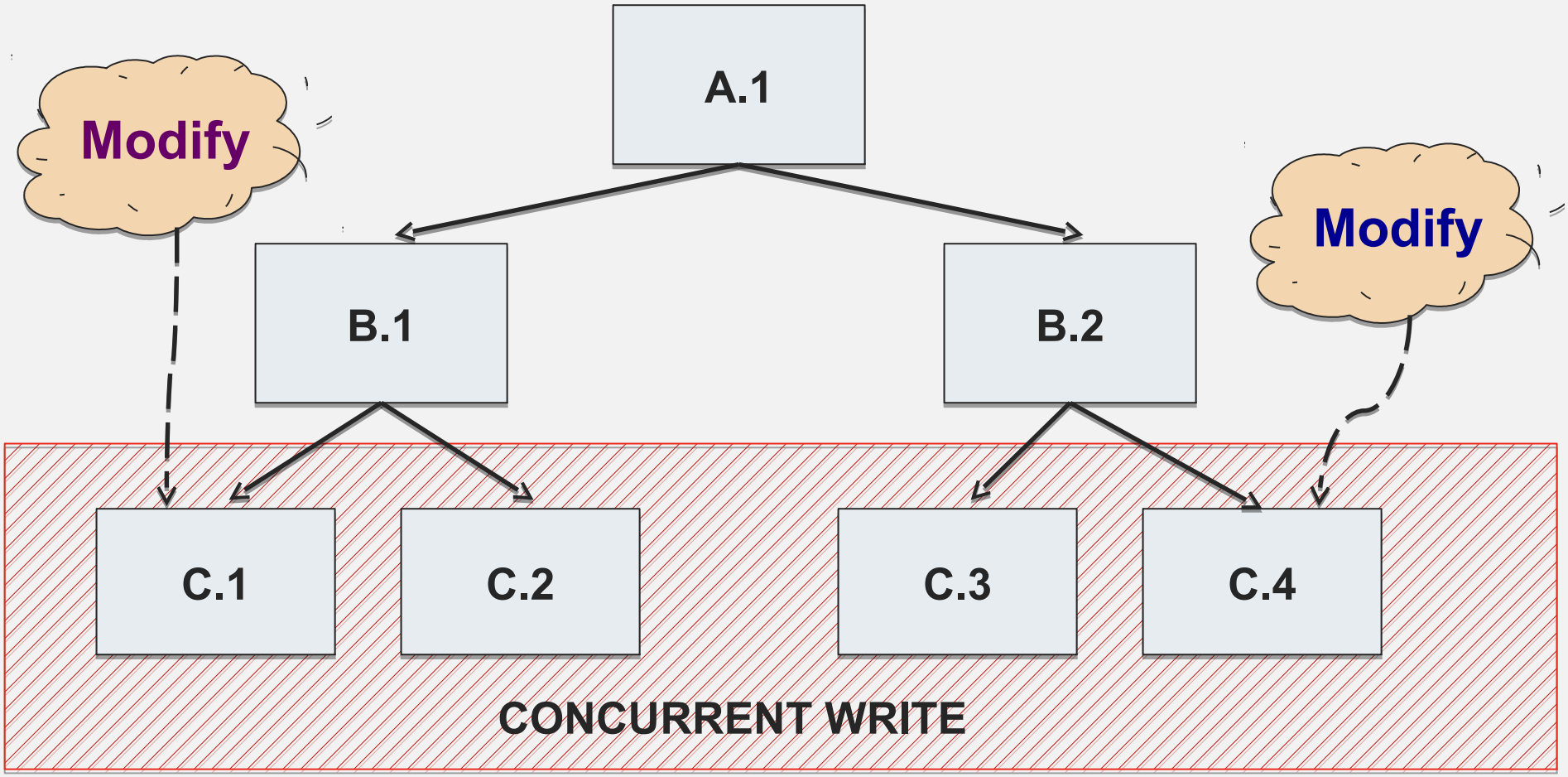
Выставляемые блокировки в Lock Manager:

- **Shared Read: /A.1**
- **Shared Read: /A.1/B.2**
- **Exclusive Write: /A.1/B.2/C.4**
- **Concurrent Write: /A.*/B.*/C.***
- **Concurrent Write: /A.1/B.*/C.***
- **Concurrent Write: /A.1/B.2/C.***

Блокировки по маске

lock(WRITE, “/A.1/B.2/C.4”);

lock(WRITE, “/A.1/B.2/C.1”);

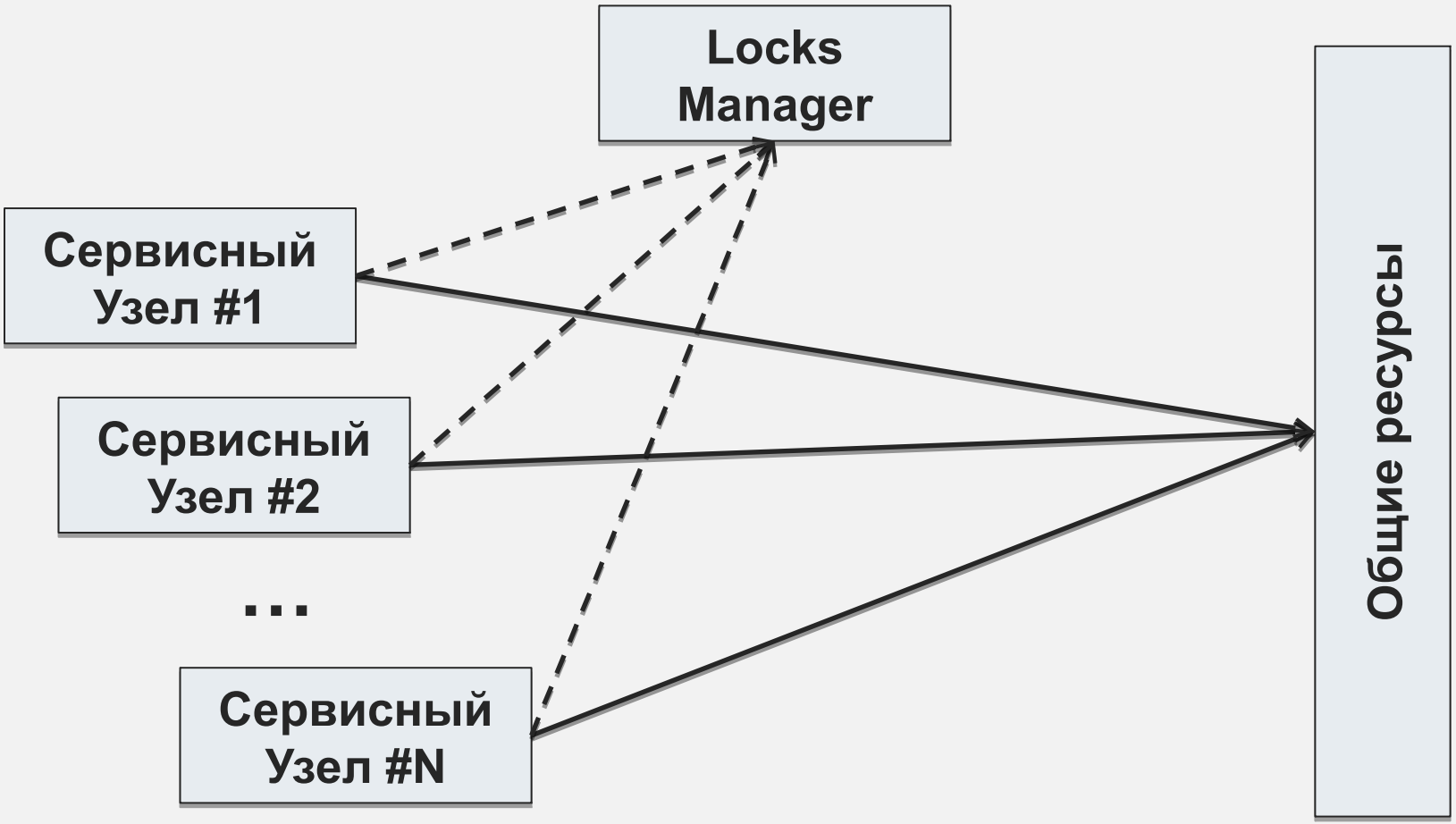


Задача

1 000 000

сайтов

Распределенная среда



Предполагаемая нагрузка

- 1,000,00 сайтов
- 100 сервисных узлов
- 10,000 сайтов на 1 узел
- ~27 одновременных пользовательских сессий на 1 узел
- ~2700 одновременных запросов на блокировки

Какие проблемы пришлось решить?

- Уменьшение вероятности deadlocks
- Диагностика deadlocks
- Обработка ситуации когда “умирает клиент”
- Обработка ситуации когда “умирает сервер”

Итого:

- Не требователен к ресурсам в одно-серверной конфигурации
- Обеспечивает синхронизацию в много-серверной среде
- Инвариантен относительно связей объектов, использующей его программы, но при этом позволяет ей оперировать объектами

EOF

Вопросы ?