Memcached

Когда стоит кэшировать

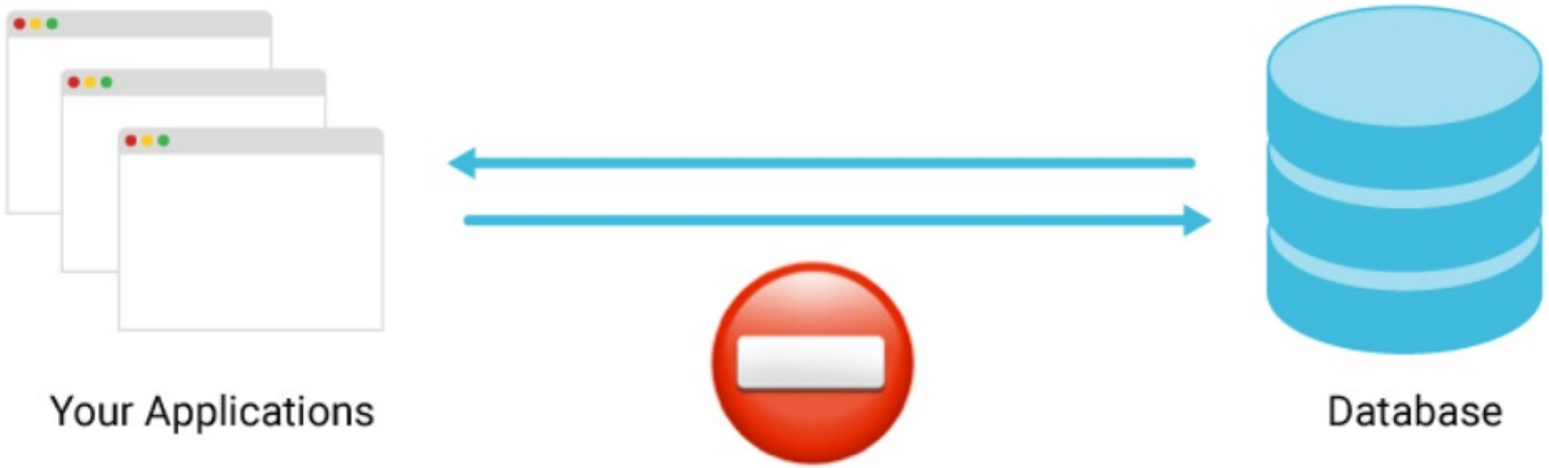
часть 1: чтение

HAZELCAST

redis

Зачем вообще кэшировать:

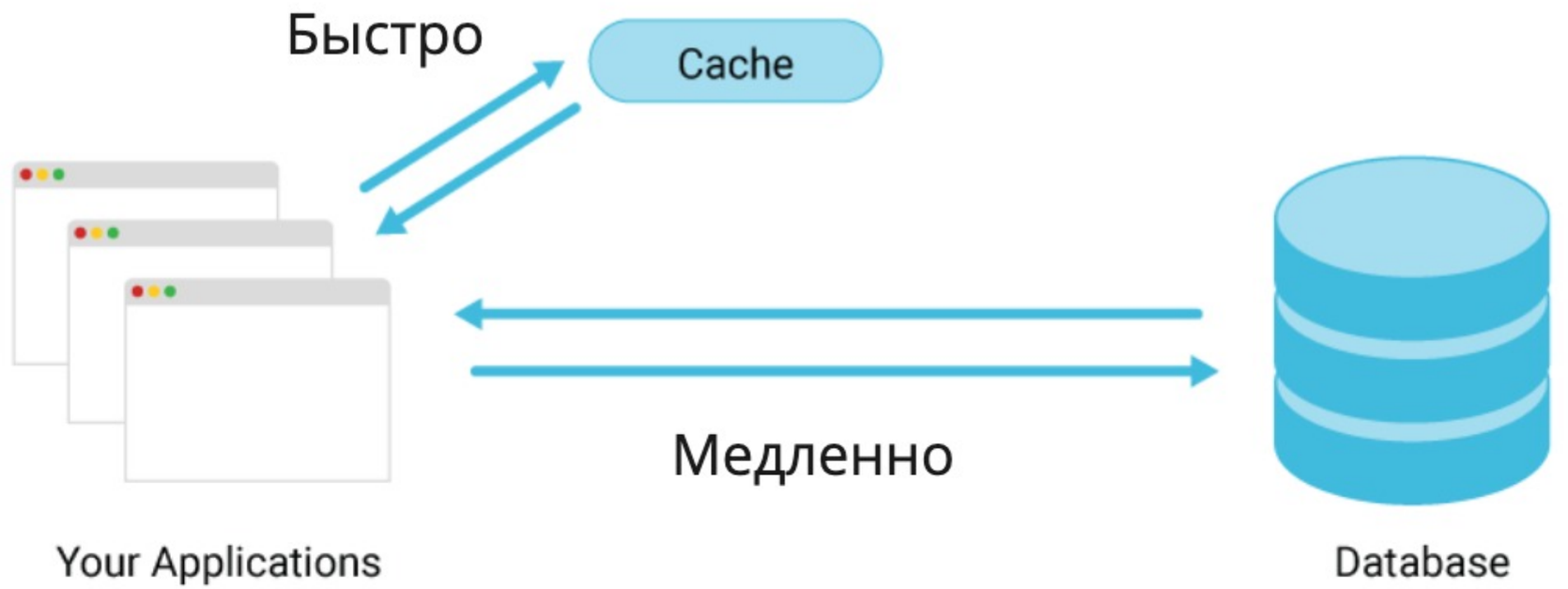
Типичное приложение



Ждем список городов по 2 секунды

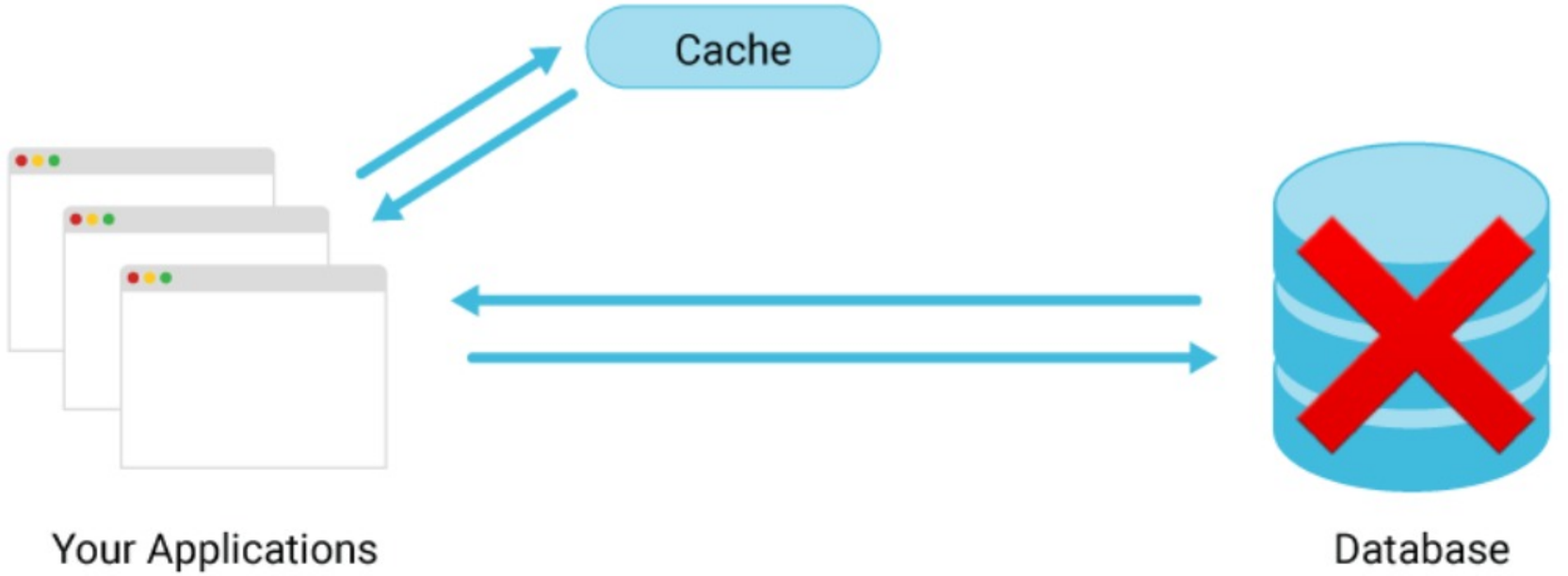
Производительность:

Долгие операции и данные не меняются какое-то время, быстро не теряют свою актуальность



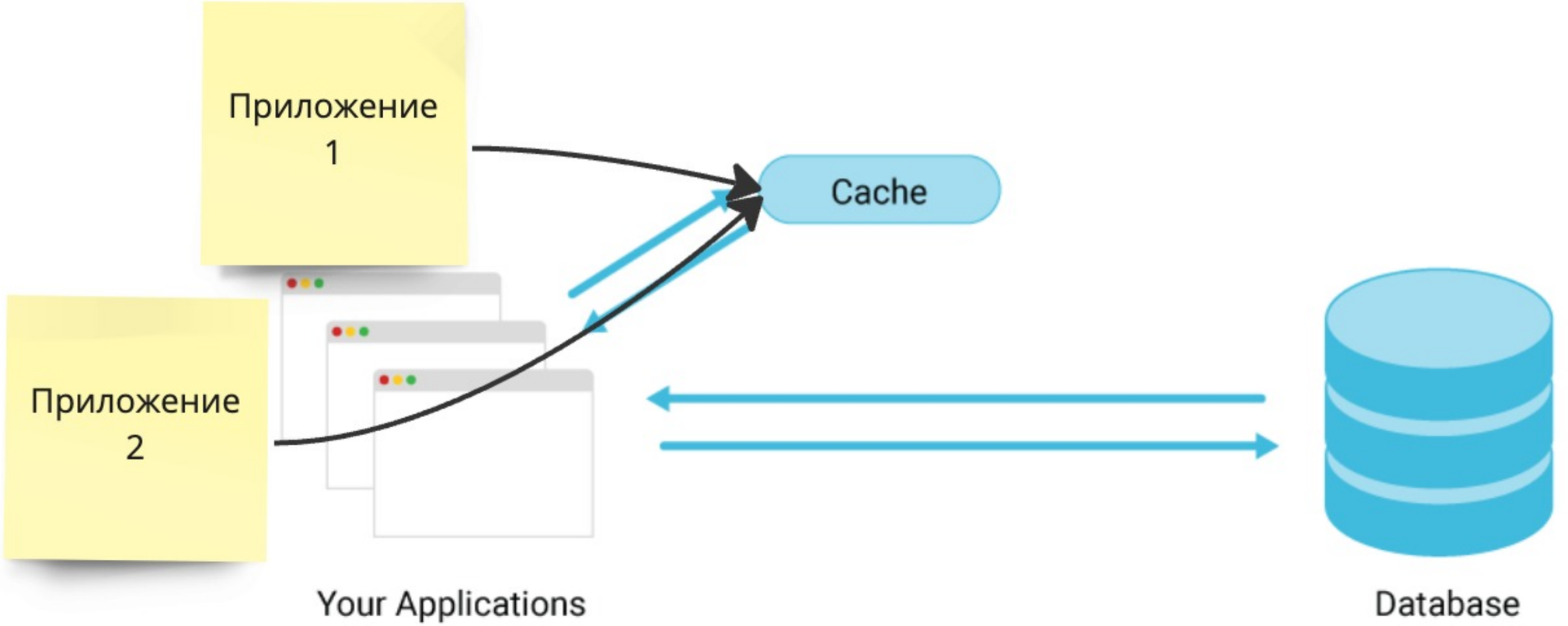
Надежность:

Повысить надежность системы в случае отказов, в зависимости от типа может иногда прикрыть хранилище целиком

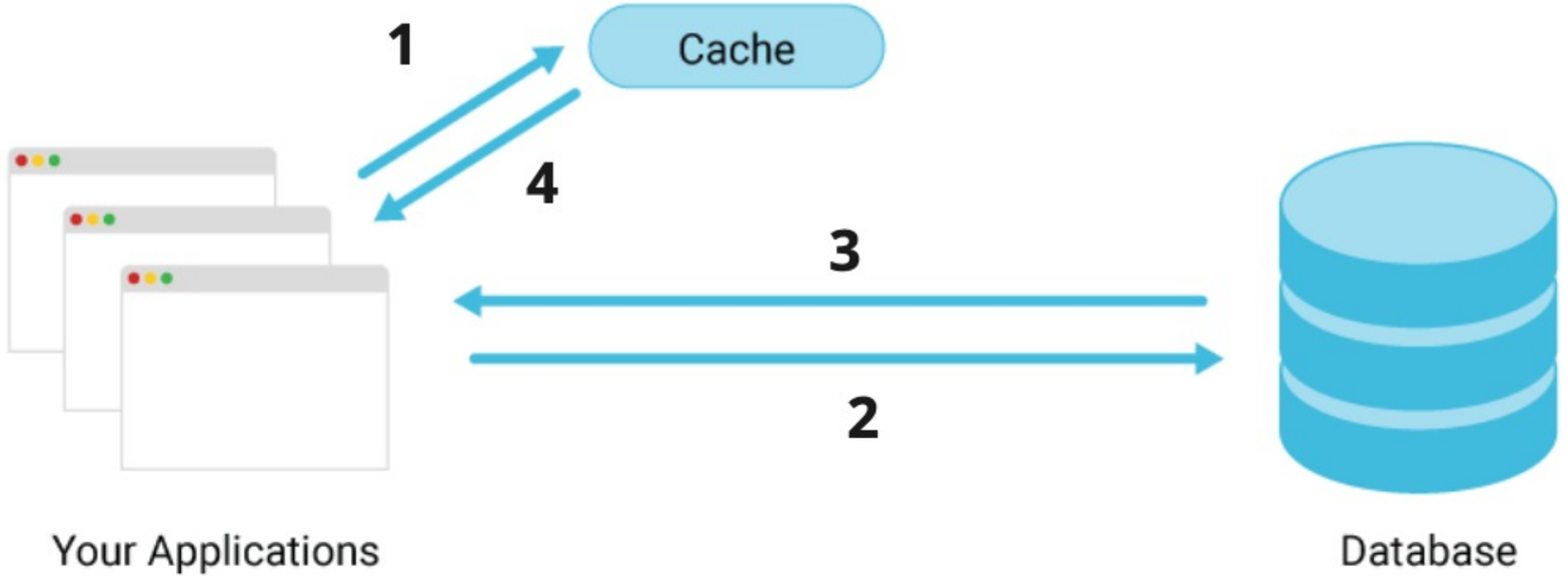


Масштабируемость:

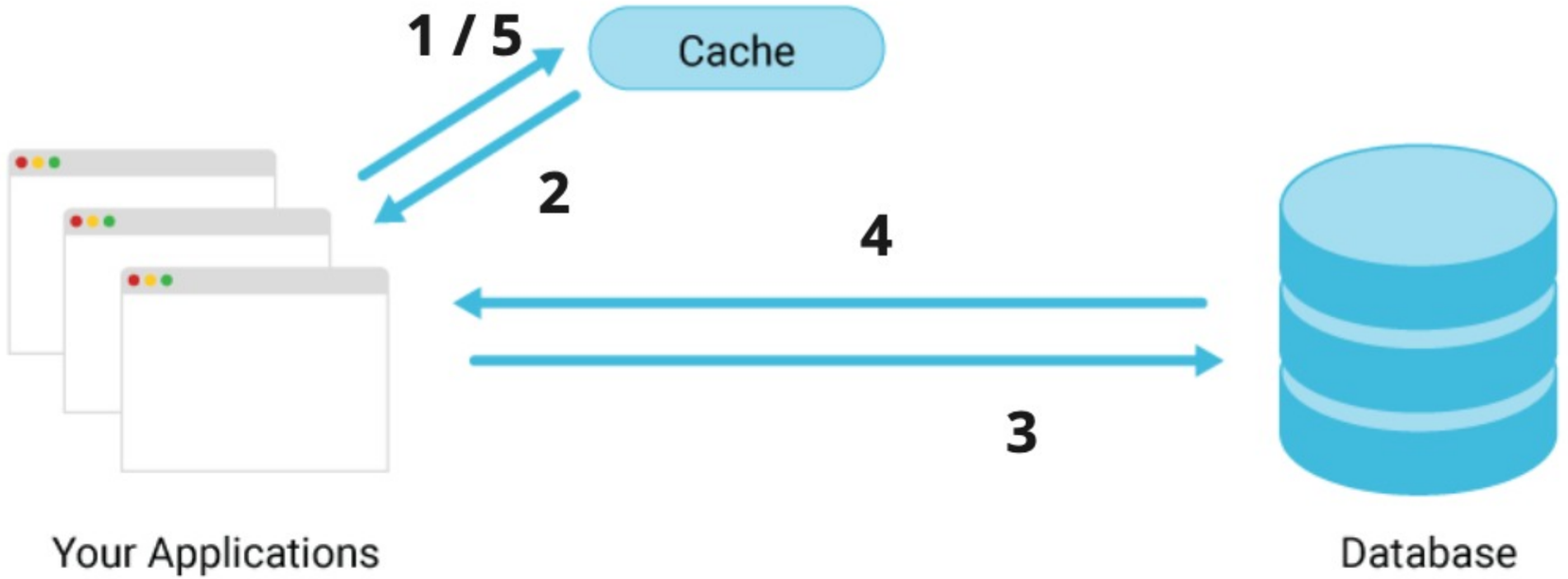
Повысить масштабируемость системы, меньше запросов - больше возможностей, снижается нагрузка и тд



Стандартный способ с кэшом



Обновление кэша в фоне



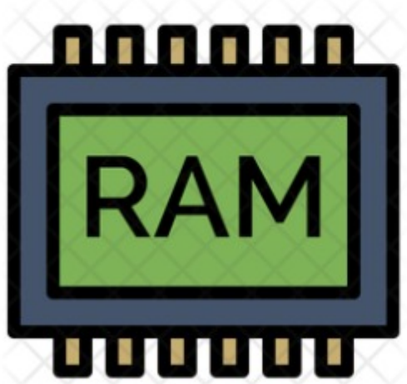
Локальный

Плюсы

1. Легко поддерживать
2. Очень быстрый

Минусы

1. Дубли данных на всех узлах
2. При перезапусках пропадает
3. Все данные в памяти узла,
   утечка памяти ведет к проблемам



Распределенный

Плюсы

1. Устойчив к перезапуску узлов
2. Данные хранятся в одном месте,
   без дублей
3. Отделение хранения данных
   в отдельную систему

Минусы

1. Сложнее поддерживать
2. Синхронизация глобального доступа

redis

Инвалидация кэшей

Стратегии

- Событийно, что-то случилось и пора делать
- По времени (время жизни ключа например 10 минут)
- Принудительно, пытаемся обновить кэш самостоятельно

Когда стоит делать

- В целом понятно, что-то произошло - мы реагируем
- Ключ ставится с таймаутом и сам удаляется через 10 минут
- Отделение кэширующего источника данных, часто используется для нагрузки

Что хранить в кэше

1. Хранение ответов из базы, источники данных
2. Хранение ответов обработчика целиком
3. Хранение сжатых данных - не стоит так делать cpu дороже памяти
4. Бинарные данные и их расшифровка

Когда кэш не нужен

- Каждый раз новые данные
- Количество данных на столько мало, что можно всегда брать из базы
- Данные статические, лучше просто заменить на константы или enum
- Задержки не критичны, надежность достаточная

Что кэш не дает

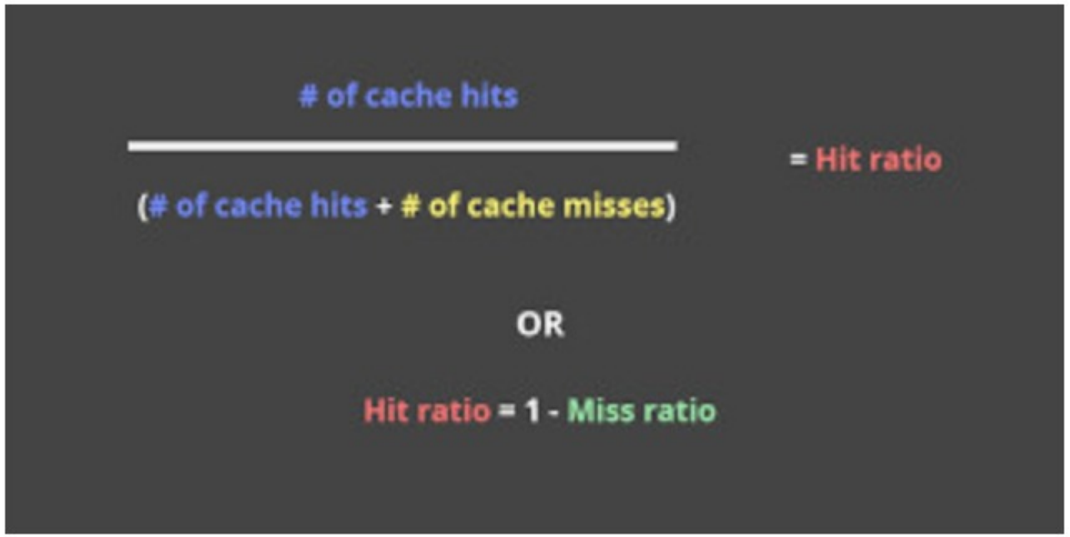
- кэши не влияют на перцентили большие

чем попадания, при промахах все равно идем в базу

Какие метрики собирать

- Попадания и промахи (hit, miss)
- Размер, текущий и максимальный (size, current)
- Количество вытесняемых объектов (evicted objects)
    - По истечению срока (ttl)
    - По превышению размера (size)

В идеале hit>0.95



Причины промахов в кэш и остальные проблемы

- Холодный старт, кэшей может не быть, как победить - в современном мире до readiness пробы загрузить данные
- Переполнение, тут банально - увеличить размер, попробовать другую стратегию
- Протухание единовременное
    - Кэшу ставится время жизни, кэш протух, данных нет
    - Хорошо всегда держать кэш и самим пытаться его просрочить, чтобы если вдруг не получится - данные есть всегда
    - Рандомная задержка между протуханием разных узлов
- Неправильные таймауты. Это целая проблема есть несколько таймаутов к кэшу и все они должны быть настроены валидно, падение кэша или сильная деградация не должны вызывать проблемы.
    - Таймаут подключения
    - Таймаут выполнения чтения
    - Таймаут записи