消息队列与限流器
Message Queue & Rate Limiter

授课老师：令狐冲

本次课程大纲

九章算法
www.jiuzhang.com

1. Message Queue 简介及原理

2. 项目中概念介绍

3. Newsfeed 的 Fanout 过程

4. 别样的数据返回机制：Lintcode 评测架构

5. Rate Limiter 简介及原理（面试常考）

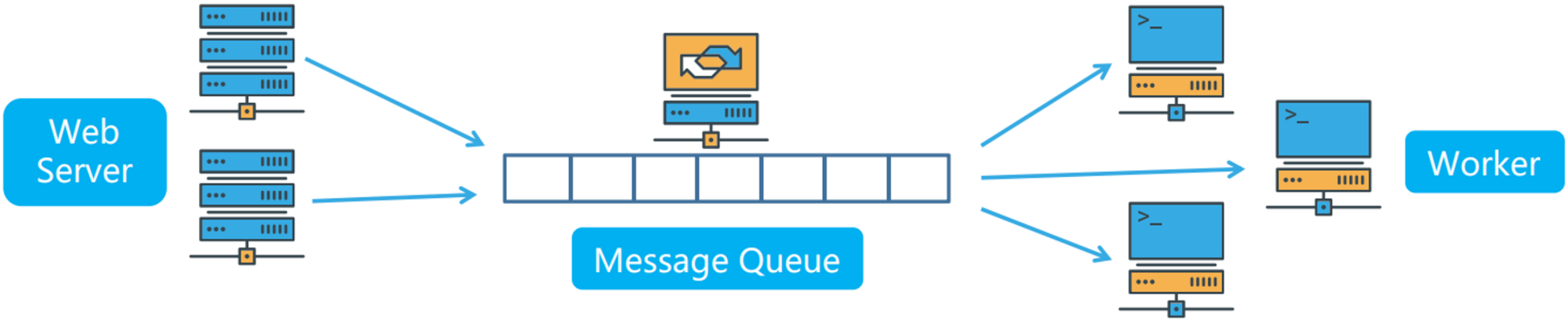
什么是 Message Queue 消息队列

- 用于任务分发的中间件

- 三大作用：
  - 处理一些耗时需要很长的操作
  - 处理一些需要重试机制的操作
  - 需要延迟或者定时处理的操作

- 使用场景介绍：
  - Web Server 在处理一个 Request 的时候，将一些操作放到异步任务的代码中，将任务所需的参数，丢给 MessageQueue，专门处理异步任务的 worker 机器从 MessageQueue 中监听到新的任务后，获得任务参数并执行任务。

Message Queue 的执行原理流程图



项目中相关概念介绍

九章算法
www.jiuzhang.com

- Broker：

  Message Queue 供应商，哪个牌子的 Message Queue，如：Redis, RabbitMQ 等

- Celey：

  一个负责与 Message Queue Broker 沟通的 Python 库（不依赖 Django）

- Task：

  任务，也可以叫异步任务(Async Task)，一个任务类似于一个函数，有名字和参数

项目中相关概念介绍

- Worker：

  执行者，即执行异步任务的机器。和 Web Server 并非同一台机器。通常是单独的集群，单独管理。通常 Web Server 和 Worker Server 的比例在 20: 1 左右，根据实际的需求不同可进行调整。

- Queue：

  队列，特指任务队列（Task Queue），一个 Broker 里可以同时存在多个队列，每个任务需要定义自己将被扔进哪个队列里，每个 Worker 的机器可以自由定制自己要监听的队列有哪些。

Message Queue 的应用场景

- 请判断以下哪些场景需要使用 Message Queue

A. 根据用户提供的参数，创建一条 Comment
B. Newsfeed 的 Fanout 过程
C. 用户注册之后，发送一封提示激活邮箱的邮件
D. 用户报名九章的课程之后，调用 Zoom API 将信息注册到 Zoom 的会议中
E. LintCode 每个月自动发送刷题分析报告给每个用户
F. 发送 10w 封邮件
G. 在 LintCode 上提交代码并评测
H. 当代码中出现 N+1 Queries 而影响 API 执行效率的时候
I. 微信发红包，1天之后未收款自动退回

Question 1：Message Queue 中存放哪些信息?

例子：通过异步任务，给 user1 这个用户发送一封邮件，提醒他激活邮箱

A. user1.id

B. user1.email

C. user 的 ORM object

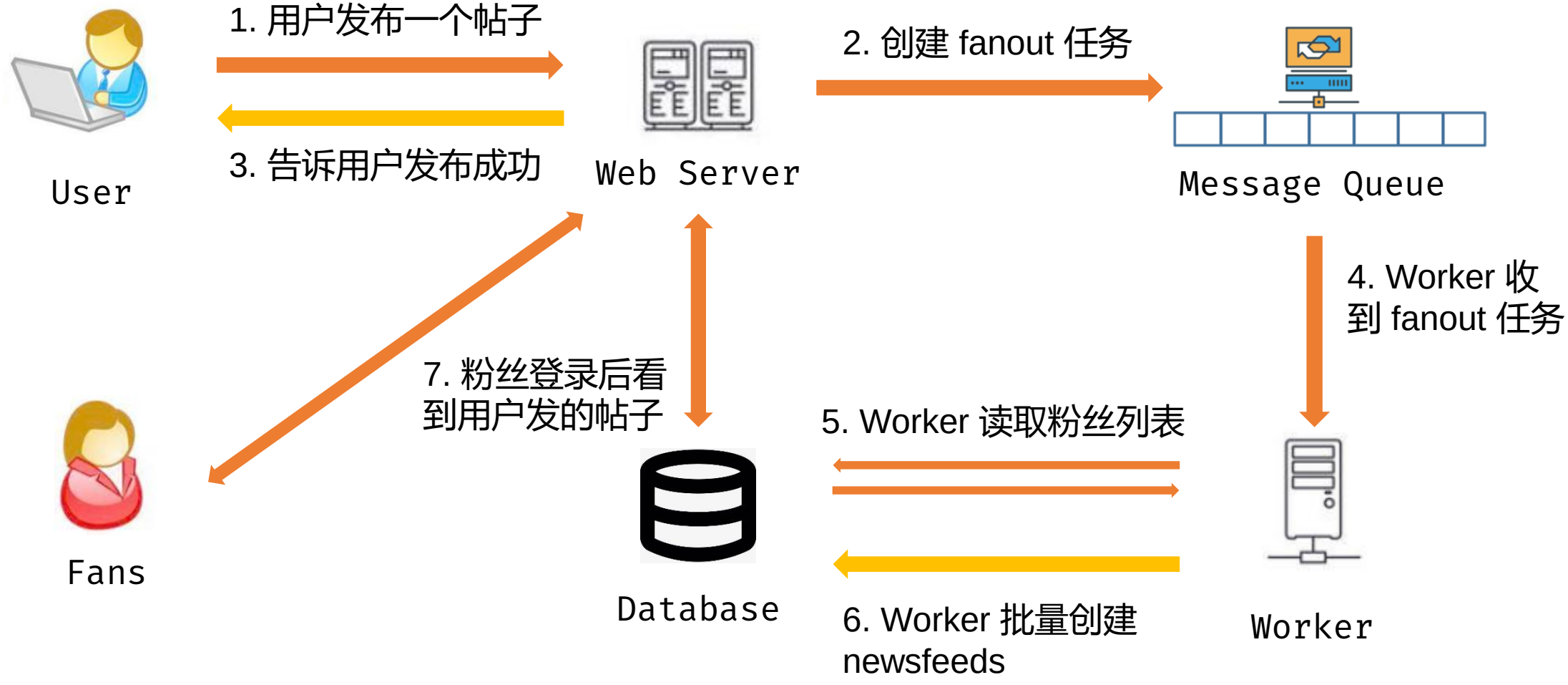
Question 2：Worker 的处理结果该如何返回?

A. 通过写入数据库或其他存储介质实现结果的“返回”

B. 通过 HTTP Request 进行 Callback 返回

C. 直接在 Worker 处理的函数中 return 要返回的数据，由 MessageQueue 负责返回结果

Newsfeed 的 Fanout 过程



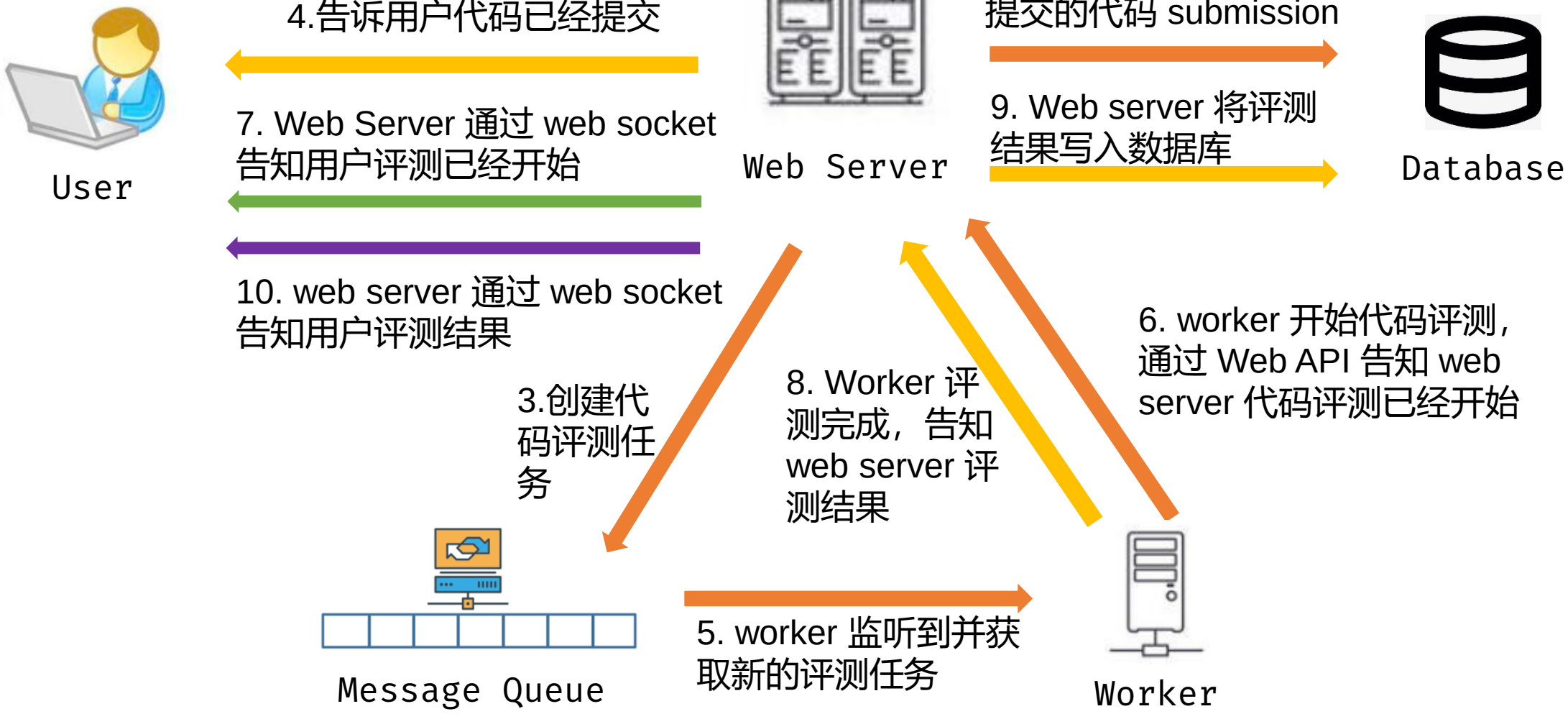
进一步利用 Message Queue 优化 Fanout

- Main Task（参数：user_id, tweet_id, tweet_timestamp）：
  - 根据发帖用户获取粉丝列表
  - 按照 1000 个为一组进行拆分，拆成若干组
  - 针对每一组创建一个 Sub Task 用于执行实际的 Newsfeed 创建工作

- Sub Task（参数：tweet_id，follower_ids，tweet_timestamp）：
  - 根据该批次的粉丝列表和 tweet_id, tweet_timestamp 创建对应的 newsfeed

- 优点：
  - 对于粉丝较多的明星用户，可以更加高效的利用上多台 worker 机器的分布式处理能力
  - 同一个 task 的执行时间不会太久，提高了 task 的完成率，避免因意外情况导致的需要整个任务的重试

别样的数据返回机制：LintCode 评测架构



什么是 Rate Limiter 限流器

- 针对每个 API 进行访问次数的限制

- 如一秒钟不超过 1 次，一分钟不超过 2 次，一小时不超过 5 次，一天不超过 10 次

项目中使用 Rate Limiter 的方法

九章算法
www.jiuzhang.com

- 直接使用 Django Rest Framework 提供的 Throttle 机制
  https://www.django-rest-framework.org/api-guide/throttling/#throttling

- 使用第三方库 django-ratelimiter处理一些耗时需要很长的操作
  https://github.com/jsocol/django-ratelimit

- 两者没有太大区别，都可以用

Rate Limiter的原理讲解（面试常考）

- 方法 1：令牌法
- 方法 2：区间求和法
- 限制要求举例：同一个账号的登录行为每分钟不超过3次
- 特征选取：使用事件名称 login + 用户的 username 作为特征，不同的特征分开限制

令牌法

- 使用 cache 存储某个特征还剩下多少次机会。key = "ratelimit:login:myusername", value = 还剩下多少次登陆机会

- 当用户第一次尝试登录时，访问 cache 发现没有这个 key，将key的 value 设置为 2。并允许此次登录尝试。将 key 的超时时间（timeout），或者叫存活时间（ time to live）设置为 1 分钟。

- 当用户1分钟内第二次尝试登录时，从 cache 中得到这个值并减去1，因为值依然 > 0，所以允许此次登录。

令牌法

- 当用户1分钟内第三次尝试登录时，从 cache 中得到这个值并减去1，因为值依然 > 0，所以允许此次登录。
- 当用户1分钟内第四次尝试登录时，从 cache 中得到这个值并减去1，因为值已经 =0，所以此次登录被拒绝。
- 如果用户在第一次登录之后的一分钟以后再尝试登陆，因为 cache 中这个 key 已经失效了，所以这次登陆会被重新记录为一次新的登录，并被允许。
- 这个方法的缺点?

区间求和法

- 使用 cache 存储某个特征在某个时刻发生了多少次。key = "ratelimit:login:myusername:2021-11-11-11:11:11", value = 在这个时刻指定用户的登录行为发生了多少次

- 用户每次尝试登录，都使用特征+时间戳为 key，在 cache 中对应的 key 的数值上 +1

- 用户每次尝试登录，都计算以当前时间戳为截止时间的过去 60s 加上特征所构成的 60 个 keys 分别是什么，累加 cache 中这些区间内的数值之和。如果超过给定的限制次数 3，则拒绝该次登录尝试。

- 这个方法的缺点?

两种方法的优缺点对比

令牌法

- 优点
  1. 效率高
  2. 规则自由
     - 可以限制 31 秒内不超过 4 次这类比较自由的规则
- 缺点
  1. 准确性低
     - false-positive 多
     - false-negative 多

两种方法的优缺点对比

区间求和法

- 优点
  1. 准确性高
     - false-positive 少
     - false-negative 少
- 缺点
  1. 效率低
  2. 规则不自由
     - 只能限制每秒钟，每分钟，每小时，每天这样的时间跨度