1. 利用消息中间件和缓存实现简单的秒杀系统

a. Redis是一个分布式缓存系统，支持多种数据结构，我们可以利用Redis轻松实现一个强大的秒杀系统。

b. 我们可以采用Redis 最简单的key-value数据结构，用一个原子类型的变量值(AtomicInteger)作为key，把用户id作为value，库存数量便是原子变量的最大值。对于每个用户的秒杀，我们使用 RPUSH key value插入秒杀请求， 当插入的秒杀请求数达到上限时，停止所有后续插入。

c. 然后我们可以在台启动多个工作线程，使用 LPOP key 读取秒杀成功者的用户id，然后再操作数据库做最终的下订单减库存操作。

d. 当然，上面Redis也可以替换成消息中间件如ActiveMQ、RabbitMQ等，也可以将缓存和消息中间件 组合起来，缓存系统负责接收记录用户请求，消息中间件负责将缓存中的请求同步到数据库。

2. 如何实现双11购物限流：

1、限流策略：

- Nginx接入层限流

按照一定的规则如帐号、IP、系统调用逻辑等在Nginx层面做限流

- 业务应用系统限流

通过业务代码控制流量这个流量可以被称为信号量，可以理解成是一种锁，它可以限制一项资源最多能同时被多少进程访问。

2、lua脚本：

```
local key = KEYS[1] --限流KEY（一秒一个）
local limit = tonumber(ARGV[1]) --限流大小
local current = tonumber(redis.call('get', key) or "0")
if current + 1 > limit then --如果超出限流大小
    return 0
else --请求数+1，并设置2秒过期
    redis.call("INCRBY", key,"1")
    redis.call("expire", key,"2")
end
 return 1
```

减少网络开销: 不使用 Lua 的代码需要向 Redis 发送多次请求, 而脚本只需一次即可, 减少网络传输;

原子操作: Redis 将整个脚本作为一个原子执行, 无需担心并发, 也就无需事务;

复用: 脚本会永久保存 Redis 中, 其他客户端可继续使用.

2、ip限流lua脚本：

```
local key = "rate.limit:" .. KEYS[1]
local limit = tonumber(ARGV[1])
local expire_time = ARGV[2]

local is_exists = redis.call("EXISTS", key)
if is_exists == 1 then
    if redis.call("INCR", key) > limit then
        return 0
    else
         return 1
     end
 else
     redis.call("SET", key, 1)
     redis.call("EXPIRE", key, expire_time)
     return 1
 end
```

3、java执行代码：

```
import org.apache.commons.io.FileUtils;
```

```java

import redis.clients.jedis.Jedis;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CountDownLatch;

public class RedisLimitRateWithLUA {

    public static void main(String[] args) {
        final CountDownLatch latch = new CountDownLatch(1);

        for (int i = 0; i < 7; i++) {
            new Thread(new Runnable() {
                public void run() {
                    try {
                        latch.await();
                        System.out.println("请求是否被执行: "+accquire());
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }).start();

        }

        latch.countDown();
    }

    public static boolean accquire() throws IOException, URISyntaxException {
        Jedis jedis = new Jedis("127.0.0.1");
        File luaFile = new File(RedisLimitRateWithLUA.class.getResource("/").toURI().getP
        String luaScript = FileUtils.readFileToString(luaFile);

        String key = "ip:" + System.currentTimeMillis()/1000; // 当前秒
        String limit = "5"; // 最大限制
        List<String> keys = new ArrayList<String>();
        keys.add(key);
        List<String> args = new ArrayList<String>();
        args.add(limit);
        Long result = (Long)(jedis.eval(luaScript, keys, args)); // 执行lua脚本，传入参数
        return result == 1;
    }
}
```

3. 要缓存网站登录的用户信息，你有几种方式?

a. redis和memcached

b. Ehcache

c. ConcurrentHashMap

4. **让你设计一套分布式缓存，如何设计可以同时更新所有服务器的缓存?**

a. rabbitmq?

5.