摘要:

2017年7月，我参与某互联公司自主研发全国运营的网约车出行平台。该平台主要是为了解决客户在线预约网约车。包含用户叫车、自动派单、司机接单、行程结算、车辆管理等功能模块。我在该项目中担任系统架构师设计师一职，主要负责整个系统的架构设计。本文以该项目为例，主要论述NoSQL数据库技术在该项目的具体应用。通过使用键值数据库Redis做内容缓存解决了项目的性能问题达到系统性能提升的目的；通过键值数据库做分布式锁解决接口多次请求重复提交问题达到幂等性保障的目的；通过键值数据库做高并发数据存储解决车辆位置查询需求达到了克服业务数据库性能的目的。事实证明，使用这些技术手段使得项目整体能够克服业务传统数据库瓶颈和应对大规模高并发场景。最终项目得以顺利完成，取得预期目标，获得用户好评。

正文:

2017年7月，我参与某互联公司自主研发全国运营的网约车出行平台。我公司致力于网络出行市场服务是首批获得当地“网约车出行牌照”, 并且先后获得四川、河南、海南、山东等各地的合法运营牌照背景下。以构建一个合法、合规安全的网约车出行环境为使命。以解决用户打车难、提升用户的出行幸福感，拉动了产业链的发展，构建了一个完整的出行生态链的为目标，构建一个全国性的网约车出行平台。 网约车出行平台包括乘客端、司机端、后台管理系统三部分组成。乘客端供乘客查询车辆、发布订单、支付车费、评论司机；司机端供司机车辆信息认证、出车接单、乘客接送、车费提现等；管理系统主要是提供系统报表查询、规则配置、乘客管理、司机管理、分公司管理、账务管理等。本平台提供了专车、快车、出租车业务，乘客可以通过自身的需求来选择不同类型业务出行，司机需要上传自身拥有的运营车辆等证照信息到平台审核，只有当平台审核通过后才能正常的再平台上合法运营。我在该项目中担任系统架构师设计师一职，主要负责整个系统的架构设计。

出行平台定位是一个全国性的互联网出行平台，如果用传统的关系型数据库会有很多难以克服的问题，于是我们决定采用NoSQL来解决大规模数据集合以及多种数据类型的挑战。NoSQL数据库的4大分类如下：1、键值（Key-Value）存储数据库，通用作内容缓存、处理高访问负载和日志系统。Key-Value的键值对数据模型底层使用hash table实现。具有查询速度快的优点。如：Redis、Oracle BDB。2、列存储数据库，通常用作分布式文件系统，以列簇形式存储。具有查找速度快、可拓展性强，容易分布式拓展的优点。如：HBase, Cassandra。3、文档型数据库，通常使用在Web应用中采用版本化文档数据模型以特定的半结构化文档格式（JSON）存储。如：MongoDb, CouchDB。4、图形数据库（Graph）通常使用在社交网络、推荐系统等采用图结构。如：Neo4J, InfoGrid。通过以上的分析和项目组内的讨论我们决定采用键值存储数据库（Redis）来作为内容缓存数据库，对频繁查询的数据内容做内容缓存提升查询性能。在使用过程中可以降低架构的复杂度、提升程序吞吐量、并且具有高水平拓展能力和低端硬件集群能力等优点，下面就具体论述其实施过程。

1、使用Redis做内容缓存提升系统性能。

在系统中我们使用Redis将查询结果作为分布式内容缓存，来提升程序心性能。在平台中我们主要缓存基础配置信息、乘客司机登录信息、订单信息。缓存基础信息，我们是在程序初始化的时候去判断基础信息是否已经加载，如果没有加载我们会去加载到Redis中，在后台更新的时候我们会同步更新缓存，保持缓存的结果和业务数据库的信息保持一致性，在乘客司机登录信息会在司机和乘客登录过程完成后返回一个Token字符串给APP端，然后token通过加密算法可以关联查询到用户信息，缓存的用户信息包含用户的，姓名，头像，年龄，性别等基础信息和当前执行的订单，当前的预约订单，当待付款订单等。可以方便在网关判断用户登录状态，在订单业务创建订单和派订单的过程中快速检索司机乘客的有效信息状态信息。订单信息主要是缓存订单ID，司机ID，乘客ID，订单状态等订单结果信息缓存起来，在订单执行过程中可以快速的查询，和推送。分布式缓存的运用能够使得系统的性能得以提升，提高了系统的吞吐量。

2、使用Redis做分布式锁解决程序幂等性问题。

在系统中我们通过Redis来做分布式锁来保证幂等性，客户在操作过程中可能由于网络抖动造成提交成功返回的结果是失败，或者一个操作客户多次重复提交的，或者有时候一比订单需要被多次扣款的情况。在系统中我们通过分析决定采用Redis分布式锁来保证接口的的幂等性。例如在派单的过程中，首先乘客创建订单后我们需要对订单状态进行锁定为派单状态，然后如果选定司机后在锁定司机，最终当派单结果完成业务数据入库提交后在解锁。将订单派单的结果推送到司机和乘客完成派单。在支付过程中我们同样是先锁定订单，然后请求支付系统当得到支付系统的返回后在返回给客户，如果在支付请求的过程中并发多次请求。只有获得锁的那个次请求能够正常的执行，其他由于没有或者锁就我们就会给他一个请求支付失败的友好提示。通过Redis分部锁的运用能够使得系统接口的幂等性得到了保障，提升了系统的稳定性降低了程序出错率。

3、使用Redis高并发读写做实时位置上报数据存储。

在系统中我们使用Redis做位置信息读写存储，因为Redis是一个内存数据库支持高并发读写而且支持数据持久化。我们就可以利用它的这些特征来做实时位置信息的存储。位置上报系统是按照某监管部们的要求，需要对运营车辆和做实时位置监控，保障乘车人安全。我们在车辆位置上报系统中每个终端需要每间隔3s向服务器上报一次车辆实时经纬度和车辆状态信息上报，我们采用的是JT808协议报文通过Socket的链接方式让APP和位置服务器做一个长连接进行实时的信息传递，当服务器端收到位置信息后将每个经纬度信息进行栅格化处理定位存储到与之对应的一个栅格中，我们将这个栅格的编号作为Redis的Key然后将具体的详细数据存储到对应的Value中，系统设定每个栅格的大小为1平方公里。这些实时位置信息在乘客端打开APP的时候就会去查询周围3km的车辆信息然后在首页的地图上展示。通过Redis的高效写入和查询能够力使得位置上报系统能够稳定高效的运行。

经过全体成员的不懈努力。在2018年2月，先后在四川、重庆、河南、贵州、海南等城市全国开展内测。2018年6月，全国正式发布运营。上线1年多程序一直稳定可靠运行。无较大线上生产事故、查询缓存命中率在90%以上大大的减少了业务数据库的压力提升了系统的整体性能。在这段时间内NoSQL也进行过2次拓容而且性能都能够的能很好的提升。上线以来得到了多个地区交通部门的点名表扬和和上万用户的好评。也为我们后续的开发、迭代、运维奠定了一个良好的基础。为NoSQL数据库及其运用和性能优化积累了丰富的经验。

项目上线至今运行1年多进入产品优化迭代阶段一直运行稳定运行，无较大生产事故。但是有一次运维同事通过分析日志发现当Redis存储的内容超过100kb 过后会出现查询时间相对较长的情况。经过分析，这个是Redis本身的一个设计缺陷在这样的场景下会出现一个性能拐点。后面我们通过拆解存储设计来改善了这个问题。实践证明，项目能够顺利上线，并运行稳定，性能良好，与系统接入NoSQL数据库设计密不可分。我们对NoSQL数据库技术在项目的应用是一个持续的过程，我们接下来，还会继续不断完善NoSQL数据库技术在项目的应用，使整个出行平台更加稳定可靠。