分布式系统

包括介绍，现状，所需的进一步工作，代码说明等

# 一、介绍

分布式系统的实验教学围绕着购物车的案例，覆盖了分布式系统中的计算、存储、通信三方面的知识，从面向用户的Web接口逐层贯穿到数据存储，涉及分布式系统设计中经常被考虑的因素：功能、容错、并发、事务、一致性等，也蕴含了系统设计中的软件工程概念：耦合、接口、扩展性等。

在此分布式系统的实验教学中，同学们通过设计和实现系统的多个组件，将分布式系统中的经典理论应用在实践中；通过对多个组件的耦合组装，完成整个购物车系统的分布式应用，满足功能、容错、并发、一致性等要求。

分布式系统的实验分为6个组件，分别是HTTP通信组件、购物车用户组件、购物车业务服务组件、分布式事务一致性组件、分布式容错一致性组件和键值对存储组件。购物车用户组件和单一实例的键值对存储组件为已知的组件，剩下的4个组件和多副本的键值对存储组件由同学们在一定的系统约束下完成设计和实现。这六个组件都是用Golang编程语言实现。

HTTP通信组件、购物车业务服务组件、分布式事务一致性组件、分布式容错一致性组件和键值对存储组件，这5个组件的系统约束主要以已有的代码结构和注释来体现，额外地，还有接口、格式等描述文档和单元测试来体现。

通过对5个组件的设计和实现，同学们最终将会使用HTTP通信组件、购物车业务服务组件、分布式事务一致性组件、分布式容错一致性组件和键值对存储组件构建一个具有分布式事务和多副本容错，具备可扩展、高可用、高并发的购物车系统。如图1.

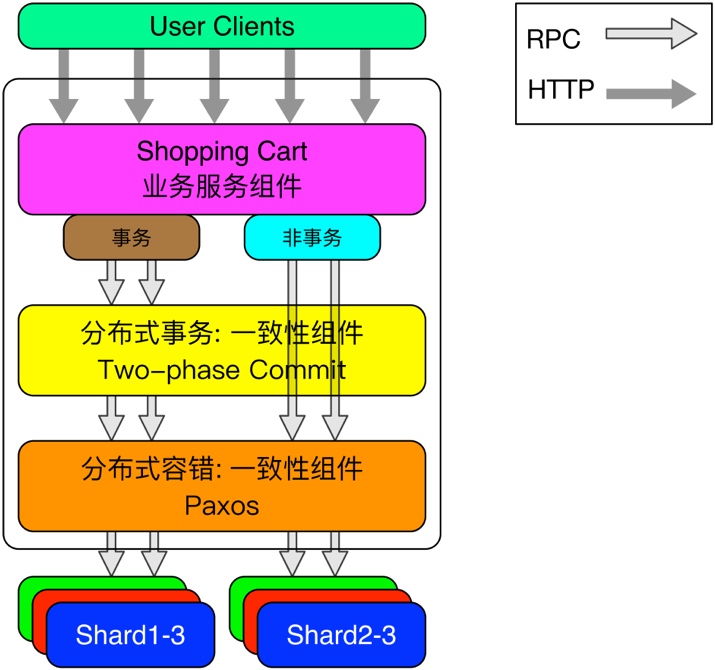


图1. 购物车系统

5个组件的单元测试、购物车用户组件的功能和压力测试的实验，可以引导同学们进行整个系统的正确性和高性能的实验测试和研究。

以下从5个组件要达到的预期目标进行介绍。

## 1. HTTP通信组件

HTTP通信组件是基于HTTP协议的通信库，包含HTTP客户端和HTTP服务端两部分。HTTP客户端提供了对基于HTTP服务端的资源访问，HTTP的服务端提供了基于HTTP协议的网络服务。

该通信组件对RFC2616的HTTP/1.1协议进行了精简，能完成基本的HTTP资源访问的语义。虽然精简，却能达成如下低耦合的设计目标：使用该库的HTTP客户端，能访问标准的HTTP服务端，不限服务端的实现语言和运行时环境；使用该库的HTTP服务端，能够支持绝大多数的标准。

相比于HTTP/1.1，该通信组件保持HTTP报文格式，只支持GET和POST两种HTTP Method，不关心报文中绝大部分的消息头的语义，无缓存机制，更多的约束细节可参考代码注释。额外地，该通信库支持客户端的并发资源访问和服务端多请求处理机制。

该通信库的传输层支持TCP和Unix Socket两种可靠的字节流传输协议，其API参考了Golang的net/http包的接口，进行了精简的重新设计。要注意的是，该通信组件完成的功能类似于Golang标准库中的net/http包，其实现不依赖net/http包，但是可以使用标准库中的net包，net包中包含了TCP和Unix Socket的实现。特别地，对于如何URL如何对应Unix Socket的名字可参考代码注释。

### 单元测试

HTTP服务端实现对于某一个变量资源的访问和加操作，通过对该变量的读写操作来测试客户端和服务端的功能和并发性能，分为四个测试案例：

1. 使用此通信库的HTTP客户端顺序地访问Golang标准库的HTTP服务端。
2. 使用Golang标准库的HTTP客户端顺序地访问此通信库的HTTP服务端。
3. 使用此通信库的HTTP客户端并发地访问Golang标准库的HTTP服务端。
4. 使用Golang标准库的HTTP客户端并发地访问此通信库的HTTP服务端。

## 2. 购物车业务服务组件

购物车业务服务组件是基于HTTP通信的Web服务应用，来源于典型的电子商务平台的抢购场景。即多用户在短时间内对仓库中有限的商品进行购买，每位用户还有一系列的约束条件，例如一个用户最多下一个订单 。

购物车组件以RESTful的接口提供服务，接口的格式、请求和回应的格式，以及接口的语义可参考接口说明文档。购物车组件中数据分为用户数据和商品数据，数据的存储依赖于键值对数据组件。

要注意的是，数据存储可以是单一实例的数据组件，也可以是多实例支持容错的分布式数据存储组件。在实验刚开始的时候，采取单一实例的数据组件（已实现），让设计和实现者关注购物车业务本身，而并非分布式容错的太多技术问题。在后续设计中，再融入分布式事务和多副本容错来支持高扩展和高可用的分布式服务。

一个用户完成一个订单的周期共有6个操作，分别是用户登录、查询所有商品详情、创建购物车、将商品加入购物车、下订单和支付订单。特别地，一个用户可以有多个将商品加入购物车的操作。每个操作都有对应的功能语义，功能语义决定了该业务的一系列的约束和处理逻辑。另外，每个操作在数据的读写上具有不同的一致性要求，其中下订单和支付订单是事务操作，具有SERIALIZABLE隔离性。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 对数据组件的读/写 | 业务一致性 | 事务性 | 事务隔离性 |
| 用户登录 | 读 | N/A | 非事务 | N/A |
| 查询商品详情 | 读 | 弱/最终一致性 |
| 创建购物车 | 写 | 强一致性 |
| 加入购物车 | 写 | 强一致性 |
| 下订单 | 写 | 强一致性 | 事务 | SERIALIZABLE |
| 支付订单 | 写 | 强一致性 |

为了体现该服务组件对大量用户的吞吐的可扩展性，该服务组件理论上应该支持任意多个相同的服务实例的运行，不同的请求会以随机的方式递送到不同的服务实例中进行处理，所以在业务和数据存储设计上都需要注意到如何保证数据一致性的问题。

### 单元测试

对购物车服务组件发起API的请求，通过检测返回的响应是否满足要求，来测试所有操作的功能语义、一致性语义、事务语义。单元测试前，需要运行该服务组件和其及依赖的服务组件，例如数据存储，事务、容错一致性组件等。

### 压力测试

对购物车服务组件发起大量用户并发请求，每个用户按照完成一个订单的生命周期进行顺序的API请求的访问，即用户登录、查询所有商品详情、创建购物车、将商品加入购物车（可有多个操作）、下订单和支付订单，多个用户之间并发发起请求。假设所有的请求都满足功能约束，即非异常的处理逻辑。测试购物车服务组件支持的吞吐量和访问延迟（包括尾延迟）。

## 3. 分布式事务一致性组件

事务一致性组件是基于two-phase commit（2PC）一致性提交协议的程序库，支持固定拓扑的协调者和参与者们的分布式事务提交。

分布式事务首先在协调者端进行初始化，然后再将事务多个部分分配给对应的参与者上进行执行，并记录相关状态，包括执行的结果和回滚操作。2PC的第一个阶段结束后，协调者将会决定下一个阶段是执行提交动作，还是回滚动作。回滚动作将会发送给第一个阶段的参与者们，将改变的状态进行回滚。提交动作将会在协调者和参与者上记录此事务的成功执行。协调者和参与者之间采用RPC协议通信。

在创建新的事务逻辑时，需要注入事务的初始化函数、哈希分区函数、处理函数和对应的回滚函数，这些函数由用户自定义。本实验中支持分布式事务提交的数据存储服务可以基于此程序库实现。

为了简化事务的并发导致的一致性问题，事务隔离性为SERIALIZABLE，所以事务在协调者上采取顺序执行的方式进行，后续事务会因正在执行的事务而阻塞，直到前序事务完成。这样的设计会简化系统的复杂度，但是也会减小资源的利用率，在大量并发事务的下，性能会严重下降。

### 容错模型

协调者无网络和停机故障；参与者们存在网络延迟、网络中断和网络分区的网络故障。参与者们也会出现停机故障，此实验中假设内存是持久化的，通过关闭服务端口来模拟停机故障。

### 单元测试

1个协调者和2个参与者进行分布式事务提交，2个参与者各对应着一个数据分片，事务操作正是分布在两个数据分片上，其操作会出现失败。检查最终事务的完成状态和返回值来测试分布式事务提交协议的safety和liveness。

## 4. 分布式容错一致性组件

容错一致性组件是基于paxos一致性共识机制的程序库，支持固定拓扑的参与者们的共识问题。该组件中，每一个序号对应一个共识问题，该程序库实现参与者们在连续序号的共识问题的一致性，提供实时查询参与者对于某一个共识问题的状态。考虑到保存共识状态的所消耗的系统资源，此组件支持对共识状态的遗忘，以释放资源空间所消耗的资源。

固定拓扑在共识问题初始化时就决定了，所有的参与者们之间的通信采用RPC协议通信。每一个参与者既是Proposer，也是Acceptor。每一个序号对应的共识问题决定着一个paxos实例，每个实例为了达到一致性，需要参与者们进行paxos的一致性协议。Paxos一致性协议的理论可参考Leslie Lamport的论文“Paxos Made Simple”。

该容错一致性组件可实现状态机多副本容错，因此本实验中的多副本容错的键值对存储系统可基于此程序库进行实现。

### 容错模型

参与者之间的网络故障可能有网络延迟、网络中断、网络分区，但是最终总会有超过半数的参与者们都在正常的网络服务分区中。参与者们也会出现停机故障，此实验中假设内存是持久化的，通过关闭服务端口来模拟停机故障。

### 单元测试

三个参与者的paxos系统进行共识过程，在不同的网络故障条件下，检测达到一致的参与者数量和一致的状态值是否满足要求，从而实现paxos共识系统的safety和liveness的目标。

## 5. 键值对存储组件

键值对存储组件提供基本键值对操作的存储服务，是购物车业务的基础服务。

该存储组件提供统一的键值对操作接口，以客户端的形式给使用者，以屏蔽存储的实现细节。包含了Get/Put/Incr/Del的4个基本操作，键和值都为字符串类型，值为整数的字符串表示。

存储组件的的实现可以是单一执行实例的服务，也可以是多副本的分布式存储服务以提供容错机制。单一的执行实例是实验已提供的存储组件，4个基本操作具有线程安全的特性。多副本的分布式存储服务扩展单一实例的数据组件，以一致性的协议来实现数据的容错机制。

多副本的分布式存储服务是基于分布式容错一致性组件实现，借助paxos保证的状态机多副本的一致性来实现可容错的键值对数据库。有序的操作序列（Get/Put/Incr/Del）就是要维护的状态机，每一个操作序列都是paxos的一个实例要达到的共识目标。通过维护有序的操作，键值对的值得以维护。

### 容错模型

多副本的分布式服务具有容错能力。其存储结构不持久化到磁盘上，而是全部存储到内存系统中。在本实验中，假设内存系统的数据是持久化的，存储组件会出现网络故障（网络延迟、网络中断、网络分区），和停机故障。

### 单元测试

基于分布式容错一致性组件构建的内存式的键值对存储服务，在不同的网络故障下，能够保持有序操作的一致性，数据修改后的状态能够立即反应到后续的操作中（即强一致性）。

# 二、现状和将来工作

HTTP通信组件、购物车业务服务组件、分布式事务一致性组件、分布式容错一致性组件这4个组件已实现了上一章介绍的目标，功能测试和单元测试全部通过。

键值对存储组件中还缺少对多副本可容错的完整支持，具体来说，支持Get/Put的操作，但不支持Incr/Del的操作，但是实现起来并不困难。

对于整套购物车系统来说，已经实现了购物车组件的多实例可扩展性，使用了分布式事务一致性组件实现了下订单和支付订单的事务，唯一不足的是数据存储还未结合多副本可容错的存储服务，目前的数据存储使用的单例的存储组件。

当多副本可容错完整支持，满足了键值对的统一用户接口后，即可实现整个购物车系统的完整性。

以下，将会从更多细节介绍当前实验系统的细节。

## 1. 容错模型的容器化

存储组件出现的故障有网络故障和停机故障，目前这些故障都是通过不同的程序异常来模拟。为了模拟网络故障，采取Unix Socket对控制进行模拟，停机故障采取关闭服务端口进行模拟。以下是故障模拟的实现方式：

1. 停机故障通过关闭服务端口。
2. 网络延迟通过Unix Socket代理接管后，延迟发送到目标Unix Socket。
3. 网络中断通过数据丢包的形式实现。
4. 网络分区通过Unix Socket File链接的方式搭起通信通道。

除了上述模拟的方式，更好的方法是在真实的网络隔离的环境中进行故障的触发。将分布式的组件部署在虚拟隔离的环境中，例如使用容器的方式进行部署。每一个组件都提供触发故障和恢复故障的接口，通过故障的调用接口和容器的环境控制，来实现分布式环境下的故障控制，检测系统实际的可用性和性能。

## 2. 强一致性的事务成为性能瓶颈

目前，购物车业务组件的事务在2PC的协调者端采用顺序执行的方式，容错模型中协调者虽然没有单点故障，但是在性能上却会成为瓶颈。

购物车可以采用多实例提供高并发的网络服务，数据分片也可以采用多个实例提高并行度，但是大量的下订单和支付订单的事务性操作将会阻塞到协调者上，事务延迟将会大大加大。这一点的根本原因在于事务隔离级别太高，阻碍了高并发的Web服务对于数据的访问。后面可以考虑使用较低的隔离级别，或者使用类似2PL 更细粒度的分布式并发协议。

额外的，购物车4个具有写数据的操作都具备业务上的强一致性，或者说具有业务的同步性，即操作的响应返回时，操作的任意处理过程都不能滞后。下订单和支付订单的操作也是强一致性的操作，在响应返回时，事务操作必须完成，所以事务的延迟高直接导致用户响应慢，也就是端到端的延迟会受到影响。事务操作的等待采取忙等还是通知的方式也会影响到业务服务系统的高并发性能。目前，实验采取的是忙等的方式观察事务操作是否完成。

## 3. 实验理论和实验方法

类似于MIT6.824，实验都有相应的理论教学，实验提示等。。。