应用系统体系结构 2021 期末考试试卷

参考解答

在本学期，同学们设计并开发了一个 Bookstore 的升级版，它面向广大的用户，提供书籍在线销售服务。在系统设计与开发的过程中，大家碰到了很多问题，在解决问题的方法上也产生了很多分歧，请你根据在本课程中学习到的知识，帮助他们分析处理以下问题（每题 5 分，共 100 分）：

请不要二次上传本 PDF 到传承获取积分或者用作其他商业用途~ 非常感谢！答案仅供参考，不保证完全正确，部分本学期未教学内容不予解答。

一、简答题：请你帮助同学们解答在设计和开发服务器端程序时遇到的下列问题：

1. 为了防止单个用户登录系统后长时间不操作导致的安全问题，同学们在后端设计了一个定时器，每当该用户有与网站交互的操作时，就重置定时器。若该用户在定时器时间到达后仍无交互操作，则删除该用户会话信息中的登录状态。请问，这个定时器服务的 Controller 和 Service 类的 Scope 应该如何设置？为什么？

回答：

两个都设置为 session（参考课程作业 1）。具体理由如下：

首先，先考虑服务层 Service，如果设置为 singleton，那么就只有一个计时器的实例，对于不同的用户不能区分，显然这不是我们想要的，如果设置为 prototype，那么就是多例，每次用户发请求来都会创建一个实例，这不仅浪费资源，而且不能记住用户上一次操作的时间，没有意义。而 session 就比较好，用户在同一个浏览器的请求都是一个实例在相应处理，这样计时器就可以记住用户上一次的访问的状态。

然后，考虑控制器 Controller，单例模式不好因为并发性能不太好，多例又比较浪费，每次都要创建一个，反而浪费时间，所以最佳应该也是 session，只要用户是一个浏览器，就是创建一个实例。

2. 由于在线书店越来越受欢迎，所以按照现有硬件配置，网站长期处于超负荷运行状态，系统性能下降导致用户体验下降。为了提高系统性能，同学们将下订单服务改写为基于异步通信的消息机制实现。请问，你觉得这种策略能够改变系统性能不佳的情况吗？为什么？

回答：

不能，题目已经说明，按照现有的硬件配置，网站长期超负荷，系统性能下降。这就是说明网站日常的业务量都非常的大，处理性能不够。下订单服务改写为基于异步通信的消息机制实现，只能缓解比如某个服务器在双十一或者某个特定时间订单量暴增的情况，例如使用卡夫卡消息中间件把订单的信息暂时放入消息队列里面，然后等到服务器压力不是那么大的时刻再来逐渐处理。这样可以应对某时间段的高并发/访问量的应对方法。

但是，如果服务器一直都处于压力状态，如果使用异步消息机制，不仅要消耗一些性能在异步通信方面，而且相当于“本身压力就很大，还在把压力往后面拖延处理”。这样对于解决问题没有作用。你应该考虑适度增加服务器的数量，或者提升单台机器的性能，而不是改用消息中间件。

3. 同学们在设计订单处理服务的 orderProcess()时，在其中调用了三个方法：placeOrder()将订单写入数据库；shipping()生成订单配送信息；log()记录日志。同学们希望 placeOrder()和 shipping()要么都执行，要么都不执行；同时，log()是否记录成功不影响订单处理。请问，要怎样设计才能实现这个目标？

回答：

orderProcess()整体要作为一个事务，在方法的前面加上@Transaction(propagation＝Propagation.REQUIRED),因为整个处理订单的服务是一个事务，要目处理完成，要目处理一半，不能出现处理一半的情况，才能保证数据库的数据的一致性。一旦处理失败整个事务就要回滚。

log()在方法的前面加上@Transaction(propagation＝Propagation.REQUIRED_NEW).因为 log 记日志是否记录成功不影响订单处理，所以要开一个新的事务，把原来的旧的事务挂起。placeOrder()和 shipping()方法的前面加上@Transaction(propagation＝Propagation.REQUIRED)，因为要让他们加入到 orderProcess()的这个事务里面，一旦函数内部出现问题整个订单处理的事务可以回滚。综上所述采用这样的处理方法。

4. 在多线程编程中，我们鼓励使用 Immutable Object，即不可变对象，这样就可以避免多线程改写对象属性时造成的状态不一致问题。请问，使用不可变对象时，如果存在更改对象属性的需求，应该如何实现呢？使用不可变对象有什么缺点？

回答：

如何实现：创建一个新的对象。缺点：每次更新都要创建一个新的对象，因此大多数程序员不太愿意使用不可变对象，认为开销比较大。

5. 在集群部署方案中，同学们使用了 Redis 缓存来存储 HttpSession 对象，请问，即使不使用 Redis，HttpSession 对象也会存储在内存中，那么为什么要使用 Redis 来存储 HttpSession 对象呢？

回答：

使用 Redis 来存储 HttpSession 对象的好处如下：

首先：服务器重启 Session 不丢失，假如后端 Java 服务器突然崩溃了重启了一下，由于 Session 是存储在 Redis 的，所以不会丢失用户的 Session，可靠性增加。

其次，便于集群里面的 Session 维护，假设我们有一个 Nginx 负载均衡，三个 Java 后端服务器，Nginx 采用 RR 轮流的方法进行负载均衡，如果不使用 Redis，可能第一次用户登录是 A 服务器记录了当前用户的 Session，第二次下订单是 B 服务器处理，那么这时候 B 服务器要找到用户的 Session 就需要跟 A 服务器通讯，这样很不好。而使用 Redis，在集群里面部署一个 Redis 服务器，所有的 Java 后端服务器都可以来找 Redis 拿 Session。

最后：Redis 便于水平方向拓展，也可以集群化部署，比如部署多个 Redis，备份和可靠性大大的增加。Redis 的数据也可以落到硬盘里面，即使崩溃了也可能可以恢复相关的数据。

6. 同学们在系统中增加了支付功能，调用了包括支付宝和微信支付在内的第三方支付服务，发现它们都是 RESTful 的 Web 服务，请问，它们为什么要设计成 Web 服务？又为什么要设计成 RESTful Web 服务？

回答：

先解释 Web 服务：Web 指的是 web 协议，例如 Http、FTP、SMTP（web 协议适用领域广，传递纯文本）不能用一些 RPC 远程调用的协议。使用 Web 协议的好处是突破访问的限制，让更广阔的区域的人能够访问得到。（所以，比如一个网络文件系统既可以通过 http 的协议，也可以通过 ftp 的协议访问）服务：独立于具体的实现：Service 解决的是异构的问题（操作系统、编程语言的差异），比如客户端是 C#开发的，服务端是 java 开发的，那为了方便他们之间的交互，我们就要走纯文本的路线大家都能认识的。既然是纯文本，那需要一定的格式，SOAP 传的数据里面包括了一系列的 operation、参数的名称类型，但是数据驱动型的。

然后说为什么设计 RESTful Web 的服务：RESTful 指的是 Representational 表述性，State 状态，Transfer 转移，全名表述性状态转移。优点如下：

1. 所有的数据都是资源(通过设计 URL 然后来表示数据，客户端用来展示数据)，每一个资源都有自己的表示方式，有自己的 URI。通过 URL 设计，Get 方法对应读、POST 代表创建、PUT 代表更新、DELETE 代表删除，四个 HTTP 方法严格对应增删改查。便于开发者开发。
2. 客户端和服务端传递的都是数据，降低带宽压力
3. 服务端只处理数据，数据展示还有处理结果完全依赖客户端，降低服务器压力
4. 幂等的，如果多次发某种请求不会导致数据库出现多个条目数据，带来一致性问题
5. 无状态的，服务器不需要维护客户端的状态，减少压力。

7. 同学们将采用微服务架构重构在线书店，将整个在线书店的功能拆分为多个微服务进行单独部署。但是，前端与后端交互时，应该通过单一访问入口来实现，请问，在微服务架构中，如何才能为前端提供单一访问入口？

回答：

通过 GateWay 转发。GateWay 的用途是：作为一个统一访问的网关。应用程序（前端）不需要知道具体的是哪个服务器处理对应的服务，只需要对 GateWay 发送请求就可以。此外，GateWay 还可以起到负载均衡的作用。

对于具体处理请求的服务器，每次启动的搜后都会到注册中心那里注册，告诉注册服务器他自己的 IP 和端口。GateWay 对于发来的请求会去根据注册中心的 IP 和端口作为转发的依据，确保成功的传达给目标服务器处理。

二、 请你帮助同学们解答在配置和使用数据库，并进行系统优化时遇到的下列问题：

1. 同学们为了优化 MySQL 的配置，考虑调整 innodb_buffer_pool 和 table_open_cache 的尺寸。请问，这两个缓存是同一块缓存吗？它们是否有区别？有什么区别？

回答：

不是。table_open_cache 这个 cache 存储的是打开文件的文件标识符。数值越大，可以同时打开的表格的数量就越多。innodb_buffer_pool 存储的是打开的数据。假如执行三条 SQL 语句修改对于某一个表里面的某一行某一列的数值，这个 buffer 的目的就是避免频繁的读写硬盘带来的磁盘 IO，把三次修改的操作全部都在缓存里面执行，这样就提高了效率。

2. 同学们打算使用第三方工具在 MySQL 上实现快照备份。请问，快照备份是对整个 MySQL 文件系统做物理复制吗？如果是，请说明这样做的好处；如果不是，请说明快照是如何实现的。

回答：

不是。快照是增量式备份的一种，一些文件系统实现允许拍摄“快照”。它们在给定时间点提供文件系统的逻辑拷贝，而不需要整个文件系统的物理拷贝。（例如，实现可以使用 copy-on-write 技术，以便只复制快照时间后修改的文件系统的部分。）

3. 因为订单表数量庞大，同学们打算使用分区方式来存储订单表，具体分区方式为使用 RANGE 分区方法按照 orderid 分区。请问，你觉得这种分区方式是否合适？为什么？

回答：

不合适，利用 orderid 分区效果不好。应该利用经常查询的列作为分区的依据。假如我的 ebook 面向全球的用户，我就可以根据地区来进行分区，比如亚洲区域、欧洲区域。或者用户可能经常要查询他自己的订单，那我可以考虑用用户的名字分区，比如 a-n 开头的一个分区 o-z 开头的一个分区。这样才能更好的发挥性能，提高性能，让分区更有意义。

当然也不是完全没有意义，如果经常 orderid 会被做自然连接，这样分区也能一定程度提高性能！

4. 同学们使用 MongoDB 来存储系统中的一些非结构化数据，例如书的简介、样章和封面图等，对于这些数据，MongoDB 会采用 Sharding 机制做分布式存储。请问，MongoDB 中是如何将一个 Collection 自动 Sharding 后实现分布存储的？

回答：

Sharding 指的是分片，MongoDB 支持自动分片。假如一个 collection 比较大，他就会把 collection 切成几个 block（chunks），把这些 block 分布到多台机器上面存储。此外 MongoDB 会自动保证不同的机器之间，block 的数量差异小于等于 2。分区之后，还有一些配置的元数据需要记录，在 config 配置服务器存储，它相当于一个路由器，针对访问的请求结合元数据，把请求转发到对应的分片存储的数据服务器来获得结果。这样的好处就是存储的压力比较平衡，当然也支持人工的管理，因为有时候数据的冷热有差差别，数据访问的频率有差别，这种情况就会手动分片，把一些经常访问的数据均衡放置，因此这种情况下，可能某一台机器存储的大小比较大，不同机器之间 block 数量的差距可能超过了 2（**这也是为什么有时候也需要人工管理 sharding 分片**）

5. 同学们在系统中增加了书评功能，即用户可以对购买的书籍发帖和对他人的书评进行回复，即支持楼上楼的功能。同学们决定使用图数据库存储书评数据，但是有些同学不理解为什么不能使用 MySQL 来存储这些数据。请你告诉这些同学，为什么这类数据更适合用图数据库存储？

回答：

这一类数据是涉及到比较复杂的关系（楼上楼回复），或者说每个个体数据之间存在很强的关联关系。假如要用 MySQL 存储，那就需要首先存储点评数据、还要存储点评数据之间的关系。那这样的话我再做查询一个数的所有点评我需要做很多次的自然连接才能把所有的查询完。

而图数据库存储的是结点、边。通过一个节点很容易就可以查找到所有跟他相关联的点。因此非常好适合楼上楼的需求。把一个数作为顶节点，然后把回复作为跟书直接相连的节点，然后对于回复的回复就跟回复的节点相连。

当然，如果是在用户只能对书发表评论的情况，Mysql 就比较适合，而在这个题目里面显然不适合。

6. 同学们在研究是否要使用日志结构合并树类型的数据库来存储订单，同学们认为既然这种数据库不存在空间放大问题，那么为什么还存在写放大问题呢？请你回答这个问题，帮助同学们解惑。

说明：

此部分是日志结构合并树内容，非本学期教学内容。

7. 无论是日志结构合并树类型的数据库，还是时序数据库，都包含一个 Write Ahead Log，请问，这个 Log 的作用是什么？

回答：

假如有数据要写入数据库，通常来说我们都是把数据写入到缓存，为什么呢，因为如果频繁直接写入到文件，对于磁盘的 IO 很大。比如三次连续的修改某个值，假如在缓存里面我只需直接修改三次，然后时间比较久后我可能会让数据落硬盘，如果每次修改都直接修改文件需要三次打开读写文件的硬盘操作。

那问题就是假如在缓存里面的数据因为断电或者其他原因崩溃而丢失了怎么办呢？所以就有了 WAL。我们在写入缓存之前，先把数据写到一个 Log 文件里面，这样哪怕崩溃了我还可以根据日志文件恢复出缓冲区里面的数据。

那这个时候就有人会问：我最开始是为了避免写硬盘的，然后我现在又开始写硬盘了这不是自相矛盾？实际上 WAL 这种日志文件的访问效率比那种数据库表的存储文件访问起来更方便，而且读写速度一般很快（一般都是在后面直接追加），所以不矛盾。

三、 请你帮助同学们解答在系统部署和运维，以及系统优化时遇到的下列问题

1. 同学们使用集群方式部署在线书店，并通过 Nginx 实现负载均衡，为了确保维护会话状态，采用了 ip hash 的策略。请问，这种策略的缺点是什么？

回答

缺点：这是三种方法里面负载均衡效果最差的一种，如果 hash 函数选择的不好的话，最终的结果很可能是导致集群里面的经常是一台机器在处理请求，而另外的几个机器可能没有什么请求被分配；当然还有一个缺点就是如果用户改变了 IP，比如上网途中开启了或者关闭了 VPN 或者代理服务器，导致用户 ip 改变，这时候 session 就可能丢失。

2. 同学们决定使用 Docker 来实现在线书店的轻量级虚拟化部署，在使用 Docker 时，同学们认为应该使用 named volume 来持久化在线书店的数据。请你告诉同学们 named volume 的实现原理。

说明：

此部分是容器与虚拟化内容，非本学期教学内容。

3. 同学们打算使用 Hadoop 的 MapReduce 计算范型来执行在线书店的大数据处理功能，例如对订单的统计和查询等。但是有的同学反对，认为它是一种读写硬盘操作密集的范型。请问，你认为 Hadoop 的 MapReduce 是一种频繁读写硬盘的操作吗？为什么？

回答：

是的！频繁的读写硬盘，因为每次计算的中间结果都会落到硬盘上保持，这样换来了可靠性但是丢失了效率。具体解释 MapReduce 的过程如下：

- 输入文件非常大，如下图分成几个 split 分片。假设我们现在要审查这个里面是否有少儿不宜的镜头还有恐怖的镜头。
- 假设我们找三个人来看，如下图所示的三个 worker。由于电影片头片尾可能内容不一样，三个人不会同时看完，也就是说他们的进度会有所差异。
- Master 是 HadoopMR 的核心，负责分配任务，看得快的人看完了 Master 就会再分配任务，如果有机器寄了偷懒不干活了，Master 就会重新把他的任务分配给活着的机器。（Master 不断的发心跳包）
- 三个 worker 会分别把恐怖的镜头放在自己存储的一个地方，少儿不宜的镜头放在另外一个地方。也就是把看的中间结果存储。
- 接下来就是 shuffle 操作，恐怖的镜头都会传给一个 worker 来进行处理，少儿不宜的镜头都会传给另外的一个 worker 来进行处理。
- 最终的两个 worker 会把前面的三个 worker 产生的恐怖的所有镜头合并，产生一个输出，所有的少儿不宜的镜头合并然后产生一个输出。
- Map Reduce 中的 Map 定义是：把输入映射成输出，每个机器不会管别的输入，只会管自己的输入部分，把输入结果产生中间结果。Reduce 负责合并的部分，把所有的中间结果合并，得到最终的输出。

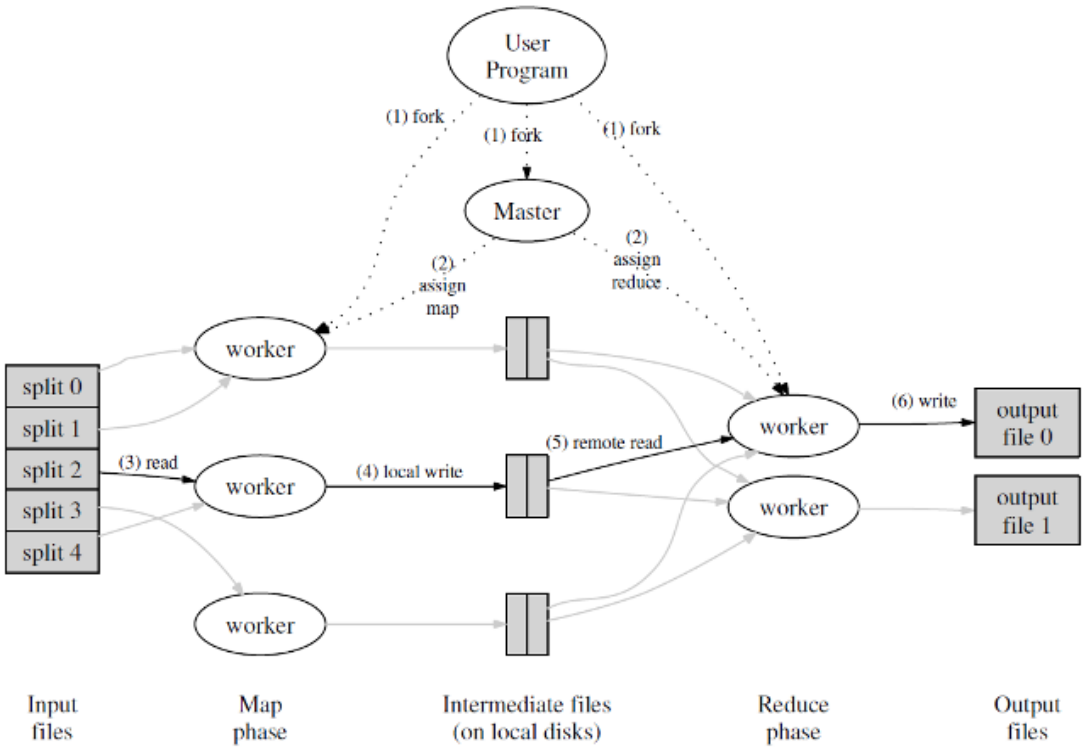

Figure 1: Execution overview

4. 同学们准备使用 Spark 来实现订单统计功能，以提升并行计算的效率。有的同学认为，Spark 的 RDD 不能修改，那么在内存中就会创建出很多 RDD，这样很浪费内存。请问，你觉得这种说法对不对？为什么？

说明：

此部分是 Spark 相关内容，非本学期教学内容。

5. 同学们在使用 HDFS 存储文件时，设置了 Block 副本数量为 3。请问，当有程序需要读取该 Block 的文件数据时，HDFS 的处理逻辑是如何确定应该访问哪个副本的？

说明：
此部分是 HDFS 相关内容，非本学期教学内容。

6. 同学们对 HBase 中 ColumnFamily 的作用不是很理解，请你告诉同学们，将数据表分解为 ColumnFamily 的优点是什么？

说明：
此部分是 HBase 相关内容，非本学期教学内容。