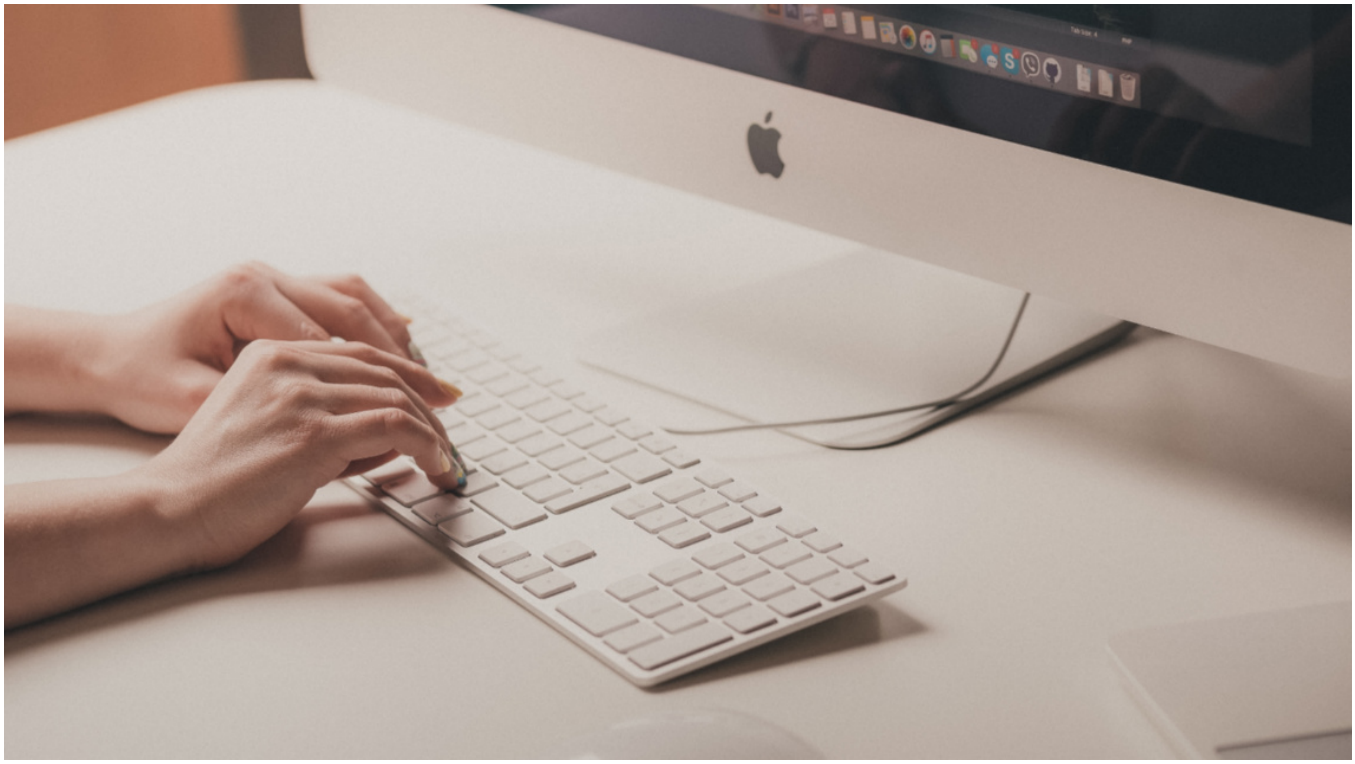
13 | 架构设计流程：详细方案设计

2018-05-26 李运华



13 | 架构设计流程：详细方案设计
朗读人：黄洲君　08′58″ | 4.11M

完成备选方案的设计和选择后，我们终于可以长出一口气，因为整个架构设计最难的一步已经完成了，但整体方案尚未完成，架构师还需继续努力。接下来我们需要再接再励，将最终确定的备选方案进行细化，使得备选方案变成一个可以落地的设计方案。所以今天我来讲讲架构设计流程第 4 步：详细方案设计。

架构设计第 4 步：详细方案设计

简单来说，详细方案设计就是将方案涉及的关键技术细节给确定下来。

- 假如我们确定使用 Elasticsearch 来做全文搜索，那么就需要确定 Elasticsearch 的索引是按照业务划分，还是一个大索引就可以了；副本数量是 2 个、3 个还是 4 个，集群节点数量是 3 个还是 6 个等。
- 假如我们确定使用 MySQL 分库分表，那么就需要确定哪些表要分库分表，按照什么维度来分库分表，分库分表后联合查询怎么处理等。
- 假如我们确定引入 Nginx 来做负载均衡，那么 Nginx 的主备怎么做，Nginx 的负载均衡策略用哪个（权重分配？轮询？ip_hash？）等。

可以看到，详细设计方案里面其实也有一些技术点和备选方案类似。例如，Nginx 的负载均衡策略，备选有轮询、权重分配、ip_hash、fair、url_hash 五个，具体选哪个呢？看起来和备选方案阶段面临的问题类似，但实际上这里的技术方案选择是**很轻量级的**，我们无须像备选方案阶段那样操作，而只需要简单根据这些技术的适用场景选择就可以了。

例如，Nginx 的负载均衡策略，简单按照下面的规则选择就可以了。

- 轮询（默认）

每个请求按时间顺序逐一分配到不同的后端服务器，后端服务器分配的请求数基本一致，如果后端服务器“down 掉”，能自动剔除。

- 加权轮询

根据权重来进行轮询，权重高的服务器分配的请求更多，主要适应于后端服务器性能不均的情况，如新老服务器混用。

- ip_hash

每个请求按访问 IP 的 hash 结果分配，这样每个访客固定访问一个后端服务器，主要用于解决 session 的问题，如购物车类的应用。

- fair

按后端服务器的响应时间来分配请求，响应时间短的优先分配，能够最大化地平衡各后端服务器的压力，可以适用于后端服务器性能不均衡的情况，也可以防止某台后端服务器性能不足的情况下还继续接收同样多的请求从而造成雪崩效应。

- url_hash

按访问 URL 的 hash 结果来分配请求，每个 URL 定向到同一个后端服务器，适用于后端服务器能够将 URL 的响应结果缓存的情况。

这几个策略的适用场景区别还是比较明显的，根据我们的业务需要，挑选一个合适的即可。例如，比如一个电商架构，由于和 session 比较强相关，因此如果用 Nginx 来做集群负载均衡，那么选择 ip_hash 策略是比较合适的。

详细设计方案阶段可能遇到的一种极端情况就是在详细设计阶段发现备选方案不可行，一般情况下主要的原因是备选方案设计时遗漏了某个关键技术点或者关键的质量属性。例如，我曾经参与过一个项目，在备选方案阶段确定是可行的，但在详细方案设计阶段，发现由于细节点太多，方案非常庞大，整个项目可能要开发长达 1 年时间，最后只得废弃原来的备选方案，重新调整项

目目标、计划和方案。这个项目的主要失误就是在备选方案评估时忽略了开发周期这个质量属性。

幸运的是，这种情况可以通过下面方式有效地避免：

- **架构师不但要进行备选方案设计和选型，还需要对备选方案的关键细节有较深入的理解。**例如，架构师选择了 Elasticsearch 作为全文搜索解决方案，前提必须是架构师自己对 Elasticsearch 的设计原理有深入的理解，比如索引、副本、集群等技术点；而不能道听途说 Elasticsearch 很牛，所以选择它，更不能成为把“细节我们不讨论”这句话挂在嘴边的“PPT 架构师”。
- **通过分步骤、分阶段、分系统等方式，尽量降低方案复杂度**，方案本身的复杂度越高，某个细节推翻整个方案的可能性就越高，适当降低复杂性，可以减少这种风险。
- 如果方案本身就很复杂，那就采取**设计团队**的方式来进行设计，博采众长，汇集大家的智慧和经验，防止只有 1~2 个架构师可能出现的思维盲点或者经验盲区。

详细方案设计实战

虽然我们上期在“前浪微博”消息队列的架构设计挑选了备选方案 2 作为最终方案，但备选方案设计阶段的方案粒度还比较粗，无法真正指导开发人员进行后续的设计和开发，因此需要在备选方案的基础上进一步细化。

下面我列出一些备选方案 2 典型的需要细化的点供参考，有兴趣的同学可以自己尝试细化更多的设计点。

1. 细化设计点 1：数据库表如何设计？

- 数据库设计两类表，一类是日志表，用于消息写入时快速存储到 MySQL 中；另一类是消息表，每个消息队列一张表。
- 业务系统发布消息时，首先写入到日志表，日志表写入成功就代表消息写入成功；后台线程再从日志表中读取消息写入记录，将消息内容写入到消息表中。
- 业务系统读取消息时，从消息表中读取。
- 日志表表名为 MQ_LOG，包含的字段：日志 ID、发布者信息、发布时间、队列名称、消息内容。
- 消息表表名就是队列名称，包含的字段：消息 ID（递增生成）、消息内容、消息发布时间、消息发布者。
- 日志表需要及时清除已经写入消息表的日志数据，消息表最多保存 30 天的消息数据。

2. 细化设计点 2：数据如何复制？

直接采用 MySQL 主从复制即可，只复制消息存储表，不复制日志表。

3. 细化设计点 3：主备服务器如何倒换？

采用 ZooKeeper 来做主备决策，主备服务器都连接到 ZooKeeper 建立自己的节点，主服务器的路径规则为“/MQ/server/ 分区编号 /master”，备机为“/MQ/server/ 分区编号 /slave”，节点类型为 EPHEMERAL。

备机监听主机的节点消息，当发现主服务器节点断连后，备服务器修改自己的状态，对外提供消息读取服务。

4. 细化设计点 4：业务服务器如何写入消息？

- 消息队列系统设计两个角色：生产者和消费者，每个角色都有唯一的名称。
- 消息队列系统提供 SDK 供各业务系统调用，SDK 从配置中读取所有消息队列系统的服务器信息，SDK 采取轮询算法发起消息写入请求给主服务器。如果某个主服务器无响应或者返回错误，SDK 将发起请求发送到下一台服务器。

5. 细化设计点 5：业务服务器如何读取消息？

- 消息队列系统提供 SDK 供各业务系统调用，SDK 从配置中读取所有消息队列系统的服务器信息，轮流向所有服务器发起消息读取请求。
- 消息队列服务器需要记录每个消费者的消费状态，即当前消费者已经读取到了哪条消息，当收到消息读取请求时，返回下一条未被读取的消息给消费者。

6. 细化设计点 6：业务服务器和消息队列服务器之间的通信协议如何设计？

考虑到消息队列系统后续可能会对接多种不同编程语言编写的系统，为了提升兼容性，传输协议用 TCP，数据格式为 ProtocolBuffer。

当然还有更多设计细节就不再一一列举，因此这还不是一个完整的设计方案，我希望可以通过这些具体实例来说明细化方案具体如何去做。

小结

今天我为你讲了架构设计流程的第四个步骤：详细方案设计，并且基于模拟的“前浪微博”消息队列系统，给出了具体的详细设计示例，希望对你有所帮助。这个示例并不完整，有兴趣的同学可以自己再详细思考一下还有哪些细节可以继续完善。

这就是今天的全部内容，留一道思考题给你吧，你见过“PPT 架构师”么？他们一般都具备什么特点？

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）

版权归极客邦科技所有，未经许可不得转载

精选留言

正是那朵玫瑰 38

可以完善的细节：

1、发送端和消费端如何寻址

利用zookeeper做注册中心，把broker的地址注册到zk上，发送端和消费端只要配置注册中心的地址即可获取集群所以broker地址，当有broker下线时，发送端和消费端能及时更新broker地址。

2、发送端消息重试

当发送消息发生网络异常时（不包括超时异常），可以重新选择下一台broker来重试发送，重试策略可以自定义。

3、消息消费采用pull还是push?

考虑push模式会更复杂，故放弃，采用pull模式，消费端主动去拉，为了达到与push模式相同的低延迟效果，可以采用长轮询的方式，消费端轮询拉取消息费，当有消费可消费时，返回消息，如果没有可消费的消息，挂起当前线程，直到超时或者有可消费的消息为止。

4、消息重复问题

消息中间件不解决消息重复的问题，有业务系统自己根据业务的唯一id去重。

5、顺序消息

发送端在发生顺序消息时，只发送到相同broker的相同队列，消费端消费时，顺序消息只能由同一个消费端消息。

6、定时消息
发送端指定消息延时多长时间消费，broker端定时扫描定时消息，达到延时时间的消息加入到消费队列。
7、事务消息
发送端分两步，先预发送消息，broker端只记录消息为预发送状态，再执行本地事务，然后再根据本地事务的成功或者失败发送确认消息（回滚还是提交），这步如果发生异常，broker启动定时任务，把未确认的消息发送给发送端回查事务状态（需要发送端提供回查接口）。

目前就想到这么多。

2018-05-28

作者回复

厉害，基本上重点都涵盖了

2018-05-28

稳健的少年 17

PPT架构师以脱离一线时间较久的领导居多吧。很多技术他们没有真正使用过，只是从各种途径得知很强大，其他公司（BAT）都在用，于是就在PPT中写入这种技术。缺点就是很容易做出貌似可行，深究其细节全是坑的设计。

2018-05-26

作者回复

BAT三个字是设计捷径，但也很多坑

2018-05-27

ant 10

很有幸我们现在的架构师就是PPT架构师，我觉得他的优点就是懂了很多的概念，能说话到，可以忽悠住老板。缺点也很明显，就是他知道的都不是很深，比如曾经我们的搜索引擎原型，他并不能说出ES和solr的优缺点(当然我也不知道，平时用solr多点)，最后我们选了ES，他给的原因就是朋友说的ES比solr好，后面搜索这里就交给我来搞了。我们是互联网项目，在重构的项目的时候他选择了jpa，这就导致变化需求的时候，查询这块比较麻烦，不灵活。其实就像前面说的，每个技术存在就是合理的，只是每个有每个技术的使用点，架构师应该对常见的技术栈原理非常清楚，知道什么时候应该使用什么技术。
我理解的PPT架构师的特点就是知识点多，知道概念，能虎住老板，缺点就是技术栈不咋滴，特别是细节上。我觉得架构师应该帮助员工成长，而不是遇到问题就说这个问题我没遇到过，你上网搜索下解决方案。
初次留言，欢迎板砖

2018-05-26

作者回复

架构师确实需要技术面宽，但别只知道技术名词，基本使用，关键原理，优缺点都需要了解

2018-05-27

蜗牛 4

就一些新的技术引入，架构师需要做哪些技术验证，或者研究到什么深度以后，才认为该技术适合呢?

2018-05-26

作者回复

基本原理，优点缺点，关键设计点，架构师至少要安装过，编写demo体验过，确定选型后，要进行性能和可用性测试例如es的索性设计就是关键设计点

2018-05-26

bluefantasy 3

请教华仔：您说的”日志表是尾部追加，性能高”，这个日志表是用myisam存储引擎吗？我刚刚查了文档innodb只有系统表空间和日志文件是序列化写呀，普通表文件都是随机写。 请指教。

2018-05-28

作者回复

这个日志表是我们自己设计的，不管是innodb还是myisam都可以，我说的尾部追加是指我们只会往日志表插入数据，类似kafka的文件尾部追加一样

2018-05-29

Jolie Liang 3

对于PPT架构师，总结有以下几个特点：

1）整体思路照搬

2）不能准确定位业务需要解决的痛点

3）对所需技术的原理理解不深

4）执行到后期，返工及补坑的工作比较多

.....

2018-05-26

作者回复

PPT可以造火箭，实际实现只能造鞭炮

2018-05-27

燃烧的阿布 3

业务系统发布消息时，首先写入到日志表，日志表写入成功就代表消息写入成功；后台线程再从日志表中读取消息写入记录，将消息内容写入到消息表中。

业务系统读取消息时，从消息表中读取。

这里的设计是不是冗余了？？

2018-05-26

作者回复

日志表是尾部追加，性能高

2018-05-27

空档滑行 3

真见过PPT架构师，1.自己基本不写代码，2.对某一项技术比较精通所以架构设计时尽量往这上面靠，比如对mysql很熟悉所以设计出的系统大量用到mysql 存储过程，3.设计出来的架构完全不考虑老系统兼容或者迁移难度

2018-05-26

作者回复

架构师手里有一把锤子，然后所有的问题都是钉子😂

2018-05-27

李志博 2

认识一个工作10多年的架构师，天天只会说代码命名和注释的问题，从来没谈过什么高大上的架构，也从来没见他写过代码，有一次我来他来帮我看一个问题，他装没听见，走了……

2018-05-31

作者回复

你们公司还要架构师么😂😂

2018-06-01

我的名字叫浩仔、 2

我们的架构师连PPT都懒得做。

2018-05-30

crazyone 2

华哥，"日志表是尾部追加，性能高"，这个具体实施细节能否讲解下。

2018-05-29

作者回复

其实高性能的很多存储方案都是这样设计的，MySQL有Binlog，HBase有HLog，道理都是相通的。

在这个备选方案中，我们设计一个日志表，假设名称叫MQ_LOG，包含ID, time, queueName, message, publisher等几个字段，每次收到消息发布请求时先写这个表，每次都是表尾部追加。

如果不用自己设计的日志表，mysql的binlog也类似尾部追加，性能也不错，缺点就是没法自己灵活实现各种刷盘机制了。

2018-05-29

Fenlon 1

一楼事物消息 还有种方案就是本地事物解决方案。 消息和业务一块存在本地，只要保证本地消息能被发到broker就好了

2018-06-15

Sylar. 1

我也没有明白作者说的日志表尾部追加性能好，具体是什么操作，还望赐教，多谢了！

2018-05-28

dilei 1

看来ppt架构师还是很多的啊，应该推荐他们来看看老师的文章o(^o^)o

2018-05-28

Garwen 1

老师，您讲的详细设计示例特别好，我非常想看到一些比较完整的详细设计文稿，不知可否有推荐。

2018-05-28

作者回复

可以自己尝试，我没有这几个备选方案的详细文档，内部的方案设计和这个方案不同，而且资料也不能泄露，望谅

2018-05-28

itperson 1

这种架构师 不了解技术细节 只了解大体使用哪种技术 技术有哪些限制或者坑了解的不透彻 这样在细化阶段会产生很多问题 这种架构师忽悠领导还凑活 落地时大多就会出状况 由此我想问 如何快速提升架构硬实力 尤其是当下技术发展迅速 人的精力有限 如何更好的进行技术了解与选型

2018-05-27

卡莫拉内西 1

ppt架构师共同的特点是:
1. 方案设计大多从网上照抄
2.回避别人提出的技术细节问题
3.口头禅: 别人都是这么用的
4.只会众多技术名称的单词拼写，而且还经常在交谈中时不时的说出一两个

2018-05-27

作者回复

细节不讨论，哈哈哈😂😂

2018-05-27

Hook 1

请老师指导下：
同一个队列的消息数据是不是被分散到了不同的分区上？此时假如要考虑消息顺序性的话，是不是就不满足了？

2018-05-27

作者回复

是的，保证消息顺序很复杂

2018-05-27

公众号：歪脖贰点零 1

架构师对个人的知识点、知识面、知识体系要求很高，也就是技术宽度与深度的问题。

平常说打杂也不为过，什么都要懂，但不一定都得上手做，要能指引他人按着蓝图去实施，当然碰到棘手问题，还是要深入进去解决。

架构师是体现了一个团队的技术强度，PPT架构师有时候还是需要做一做，但不能只停留在PPT里，落地实战同样不能逊色。

2018-05-27

作者回复

我的习惯是最好安装运行，然后写demo体验一下，单纯看文档理解还是不够

2018-05-27

行用 1

现在很多架构师只关注主流程，大方向，根本不关注细节。导致做出来的东西用户体验不好

2018-05-27

作者回复

两者需要兼顾，把握平衡

2018-05-27