02 | 消息收发架构：为你的App，加上实时通信功能

2019-08-30 袁武林

即时消息技术剖析与实战 进入课程 >

讲述：袁武林
时长 16:26　大小 11.29M

你好，我是袁武林。

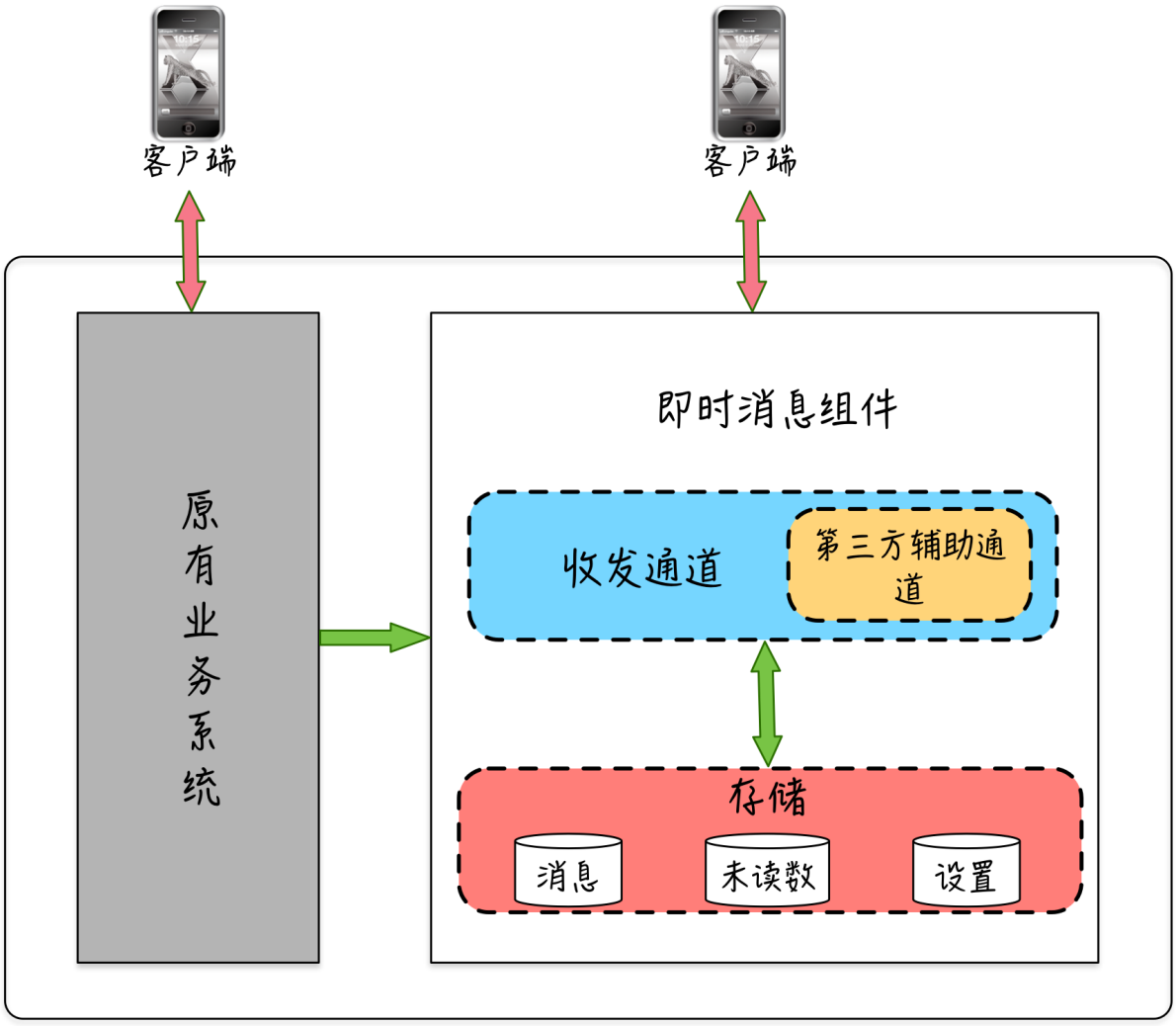
前一篇文章中，我们从使用者的直观角度和从业者的实现维度，了解一个 IM 系统都应该具备哪些要素。但实际上，从我的角度来看，我更倾向于把“IM”看做是一门可以融入到各种业务系统中，为业务系统提供“实时交互”能力的技术模块。

比如，极客时间想在它的 App 中增加一个互动模块，支持用户点对点的实时聊天功能。那么，我们就可以相应地通过一些 IM SDK 的方式，快速地把即时消息的技术引入到已有的业务系统中。

同样的，一个传统的视频网站如果想让自己的视频支持弹幕功能，也可以通过引入即时消息的技术，来让视频弹幕的参与者能实时、高效地和其他观看者进行各种互动。

所以，从某种程度上看，随着移动网络的快速发展以及资费的快速下降，即时消息技术也越来越多地被广泛应用到各种业务系统中，用于提升用户实时互动的能力。

那么，接下来，我们就一起从即时消息更细化的实现角度来看一看，给一个已有系统增加即时消息功能，大致上都有哪些具体工作。



如果为原有的业务系统增加实时消息模块，在不需要重建账号体系的前提下，整体上大概包括几块内容：

一般来说首先需要制定好消息内容和未读数的存储，另外需要建立比原业务系统更加高效实时的消息收发通道，当然也包括依托第三方辅助通道来提升消息到达率。

下面我们分别来看一下各部分大体需要做的工作都包括哪些。

消息存储

我们回想一下上一篇的内容，即时消息系统中，消息作为互动的载体，是必不可少的要素之一。

一般来说，大部分即时消息系统为了便于查看历史消息或者用于暂存离线消息，都需要对消息进行服务端存储，因此，我们先来看一看，这些互动过程产生的消息在服务端应该怎么存储或者暂存。

消息索引和消息内容

这里，我以点对点消息的存储为例，来讲解一下。

点对点消息的参与方有两个：消息发送方和消息接收方。收发双方的历史消息都是相互独立的。互相独立的意思就是：假设发送方删除了某一条消息，接收方仍然可以获取到这条消息。

所以，从库表的设计上分析，这里需要索引表中收发双方各自有一条自己的索引记录：一条是消息发送方的发件箱索引，另一条是消息接收方的收件箱索引。

由于收发双方看到的消息内容实际都是一致的，因此还需要一个独立的消息内容表。

消息内容表用于存储消息维度的一些基本信息，比如消息 ID、消息内容、消息类型、消息产生时间等。收发双方的两个索引表通过同一个消息 ID 和这个内容表关联。

这里假设张三给李四发送一条消息，消息存储在 MySQL，或者类似的关系型数据库中，那么上面涉及的两张表大致如下：

内容表

消息ID	消息内容	消息类型	消息产生时间
1001	你好	文本消息	2019-07-15 12:00:00

索引表

索引的用户UID	索引消息的另一方参与用户UID	是发件箱还是收件箱	消息ID
12345678（张三的UID）	87654321（李四的UID）	0	1001
87654321（李四的UID）	12345678（张三的UID）	1	1001

比如张三给李四发了一条“你好”的消息，那么这个动作会向内容表存储一条消息。这条消息内容是这样的：ID 为 1001，消息内容是“你好”，消息类型是文本消息，还有当时消息创建的时间。

并且，它同时会往索引表里存储两条记录。

一条是张三的索引：内容有会话对方的 UID（李四的 UID），是发件箱的索引（也就是 0），同时记录这条消息的内容表里的消息 ID 为 1001。

另一条是李四的索引：内容有会话对方的 UID（张三的 UID），是收件箱的索引（也就是 1），同样也同时记录这条消息的内容表里的消息 ID 为 1001。

联系人列表

有了消息和索引后，如上一篇中的描述，一般 IM 系统还需要一个最近联系人列表，来让互动双方快速查找需要聊天的对象，联系人列表一般还会携带两人最近一条聊天消息用于展示。

这里你需要理解的是，和消息索引表的存储逻辑相比，联系人列表在存储上有以下区别。

> **联系人列表只更新存储收发双方的最新一条消息，不存储两人所有的历史消息。**
>
> **消息索引表的使用场景一般用于查询收发双方的历史聊天记录，是聊天会话维度；而联系人表的使用场景用于查询某一个人最近的所有联系人，是用户全局维度。**

在库表的设计上，联系人列表的存储实际和消息索引表类似，只不过消息索引表在接收到消息时，大部分情况都是插入操作，而联系人列表很多时候是更新操作。

最近联系人表

索引的用户UID	索引消息的另一方参与用户UID	消息ID
12345678（张三的UID）	87654321 （李四的UID）	1001
87654321 （李四的UID）	12345678（张三的UID）	1001

还是刚才那个例子，张三给李四发完消息后，除了在内容表和索引表插入记录，还会更新各自的最近联系人表，这里需要分别更新张三的最近联系人表和李四的最近联系人表。

比如更新张三的最近联系人表，如果和李四之前没有聊天记录，那么新插入一条联系人记录。联系人的对方 UID 为李四的 UID，和这个联系人最新的一条消息 ID 是 1001。

如果张三和李四之前已经有过聊天记录，那么只需要更新张三和李四的最新的一条聊天消息 ID 为 1001，同样的办法再更新一次李四的联系人列表。

以上就是消息存储部分最重要的三个表，消息内容表、消息索引表、联系人列表。它们大致的存储结构，我们就设计好了。

消息收发通道

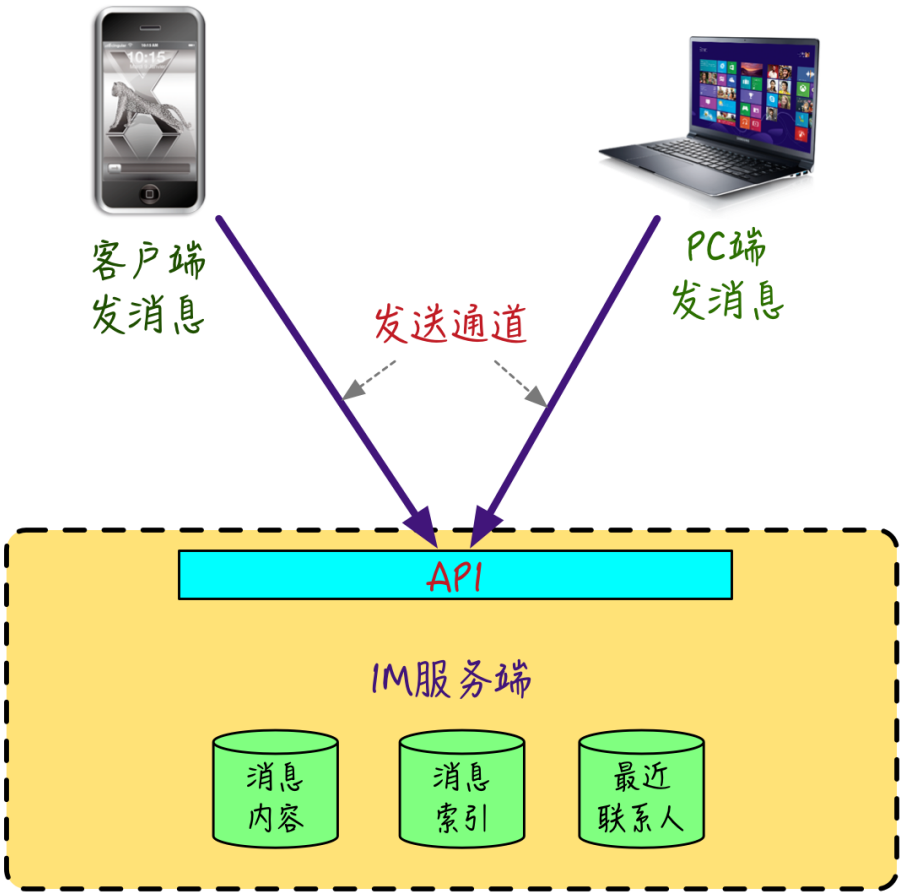
设计好消息的存储结构后，接下来，我们需要考虑的是：如何将消息发出去，以及怎么把消息投递给接收方。这里逻辑上涉及了两条通道：一条是消息发送通道，一条是消息接收通道。

发送方通过发送通道把消息从本地发送到 IM 服务端；IM 服务端通过接收通道把消息投递给接收方。

消息发送通道

发送通道的实现上有很多种方式，比如下面的两种。

1. IM 服务端提供一个 HTTP 协议的 API 接口，客户端需要发送消息时，调用这个接口把消息发给 IM 服务端。
2. 客户端和 IM 服务端维护一个 TCP 长连接，客户端有消息发送时，会以私有协议来封装这条要发送的消息，然后通过这个 TCP 长连接把消息发给 IM 服务端。

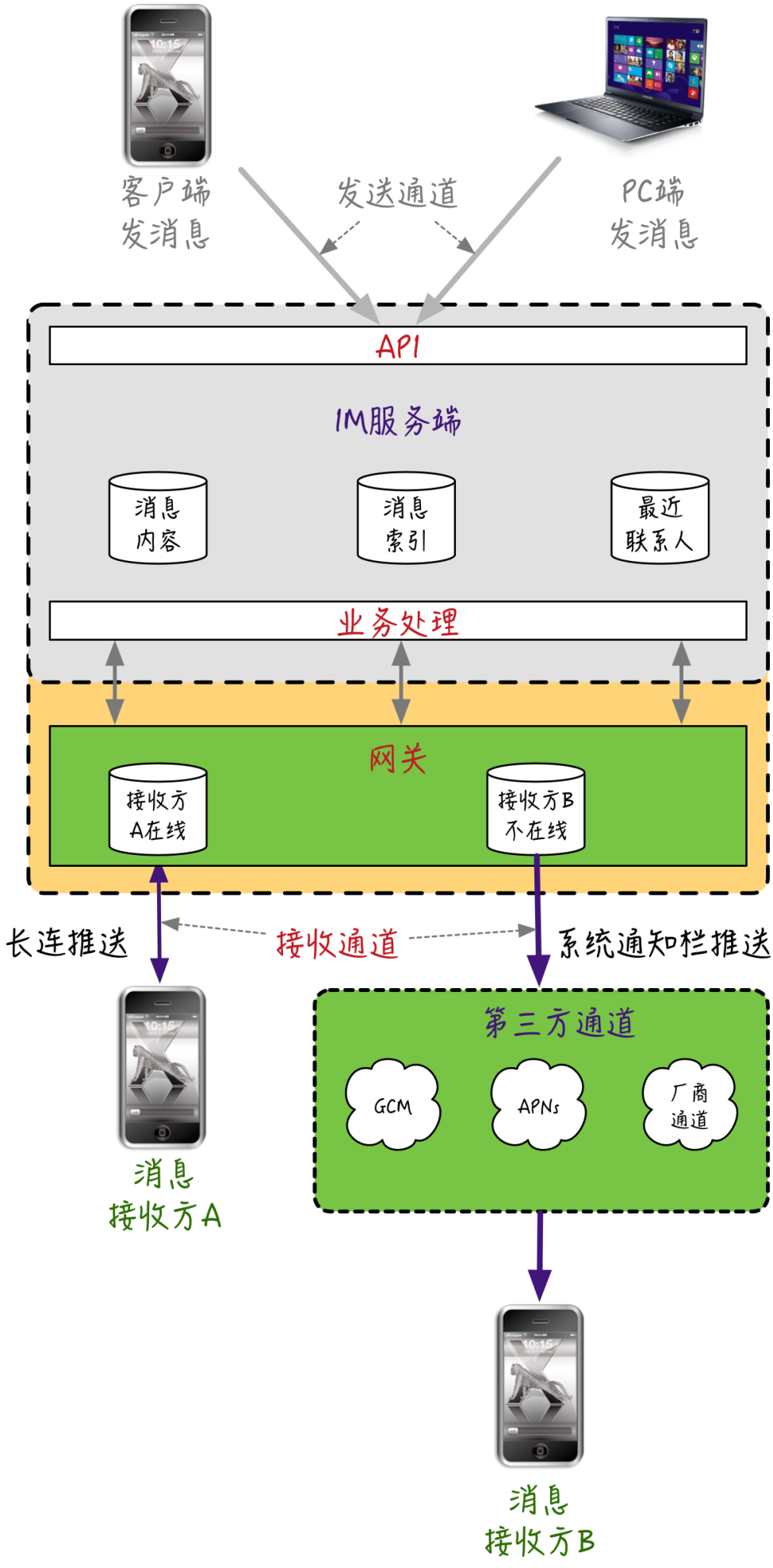


所以，发送通道的实现相对比较简单，**重点在于：IM 服务端提供消息发送的 API，发送方可以通过任意方式调用到这个 API，把消息发出去即可。**

消息接收通道

对于我们最常见的非 P2P 模式的 IM 系统来说，由于**有一条消息要投递给某个接收方**这个事件，接收方并没有办法能实时知道，只有 IM 服务端收到发送方发出的消息时能实时感知到，因此消息投递这个动作一般都是 IM 服务端触发的（这里，我们不去讨论由接收方通过轮询获取消息的模式）。

下面，我画了一张图来说明接收通道的业务逻辑，目前业界在消息接收通道的实现上较多采用的方式是下面这样的。


客户端
发消息
发送通道
PC端
发消息
API
IM服务端
消息
内容
消息
索引
最近
联系人
业务处理
网关
接收方
A在线
接收方B
不在线
长连推送
接收通道
系统通知栏推送
第三方通道
GCM
APNs
厂商
通道
消息
接收方A
消息
接收方B

解释一下这张图。

IM 服务端的网关服务和消息接收方设备之间维护一条 TCP 长连接（或者 Websocket 长连接），借助 TCP 的**全双工能力，也就是能够同时接收与发送数据的能力。**当有消息需要投递时，通过这条长连接实时把消息从 IM 服务端推送给接收方。

对于接收方不在线（比如网络不通、App 没打开等）的情况，还可以通过第三方手机操作系统级别的辅助通道，把这条消息通过手机通知栏的方式投递下去。

这里简单解释一下，常见的第三方操作系统级别的辅助通道。比如苹果手机的 APNs（Apple Push Notification Service）通道、Android 手机的 GCM 通道，还有各种具体手机厂商（如小米、华为等）提供的厂商通道。

这些通道由于是手机厂商来维护的，只要手机网络可通，因此可以在我们的 App 在没有打开的情况下，也能把消息实时推送下去。

当然，这些第三方操作系统级别的辅助通道也存在一些问题，因此大部分情况下也只是作为一个辅助手段来提升消息的实时触达的能力，这个在后续课程中，我会再详细说明。

因此，对于消息接收通道，重点在于需要在 IM 服务端和接收方之间，维护一个可靠的长连接，什么叫可靠的长连接呢，这里的可靠可以理解为下列两种情况。

1. IM 服务端和接收方能较为精确地感知这个长连接的可用性，当由于网络原因连接被中断时，能快速感知并进行重连等恢复性操作。
2. 可靠性的另一层含义是：通过这个长连接投递的消息不能出现丢失的情况，否则会比较影响用户体验。这个问题的解决会在后续第 3 篇的课程中来详细展开。

我在上面大概说明了一下，逻辑上消息收发通道各自的作用和一般的实现，当然这两条通道在实际的实现上，可以是各自独立存在的，也可以合并在一条通道中。

消息未读数

现在我们有了消息的收发通道和消息的存储，用户通过发送通道把消息发到 IM 服务端，IM 服务端对消息内容、收发双方的消息索引进行存储，同时更新双方的最近联系人的相关记录，然后 IM 服务端通过和消息接收方维护的接收通道，将消息实时推送给消息接收方。

如果消息接收方当前不在线，还可以通过第三方操作系统级别的辅助通道，来实时地将消息通过手机通知栏等方式推送给接收方。

整体上来看，一条消息从发送、存储、接收的生命之旅基本上比较完整了，但对于即时消息的场景来说，还有一个比较重要的功能，会对双方在互动积极性和互动频率上产生比较大的影响，这个就是消息的未读数提醒。

用过 QQ、微信的用户应该都有一个比较明显的感知，很多时候为了避免通知栏骚扰，会限制掉 App 在通知栏提醒权限，或者并没有注意到通知栏的提醒，这些情况都可能会让我们无法及时感知到“有人给我发了新的消息”这个事情。

那么作为一个重要的补救措施就是消息的未读提醒了。就我个人而言，很多时候是看到了 QQ 或者微信 App 的角标，上面显示的多少条未读消息，才打开 App，然后通过 App 里面具体某个联系人后面显示，和当前用户有多少条未读这个数字，来决定打开哪个联系人的聊天页进行查看。

上面通过未读提醒来查看消息的环节中涉及了两个概念：一个是我有多少条未读消息，另一个是我和某个联系人有多少条未读消息。

因此，我们在消息未读数的实现上，一般需要针对用户维度有一个总未读数的计数，针对某一个具体用户需要有一个会话维度的会话未读的计数。

那么，这两个消息未读数变更的场景是下面这样的：

1. 张三给李四发送一条消息，IM 服务端接收到这条消息后，给李四的总未读数增加 1，给李四和张三的会话未读也增加 1；
2. 李四看到有一条未读消息后，打开 App，查看和张三的聊天页，这时会执行未读变更，将李四和张三的会话未读减 1，将李四的总未读也减 1。

这个具体的未读数存储可以是在 IM 服务端（如 QQ、微博），也可以是在接收方的本地端上存储（微信），一般来说，需要支持“消息的多终端漫游”的应用需要在 IM 服务端进行未读存储，不需要支持“消息的多终端漫游”可以选择本地存储即可。

对于在 IM 服务端存储消息未读数的分布式场景，如何保证这两个未读数的一致性也是一个比较有意思的事情，这个问题我会留到第 6 篇来和你详细讨论。

小结

上面我们从一条消息“产生、存储、接收”的整个生命周期出发，较为系统地从实现的角度上对消息系统的几个关键部分进行了讲述。可以简单地总结为下面几点。

1. 消息的发送方通过发送通道来把消息提交到 IM 服务端。
2. IM 服务端接收到发送的消息后，会进行消息的存储以便于后续历史消息的查看，消息的存储从实现上可以分为：消息内容存储、消息索引存储、最近联系人列表存储。
3. IM 服务端接收到发送的消息后，还会针对接收方进行未读数的变更，以提醒用户查看未读的消息，消息未读数的实现上一般分为：用户维度的总未读和会话维度的会话未读。
4. IM 服务端进行完消息存储和未读变更后，会通过接收通道把消息推送给接收方，接收通道一般是通过 IM 服务端和消息接收方之间维护的长连接来实现，还会使用第三方操作系统级别的辅助通道，来提升“自建的长连接不可用“时，实时触达的能力。

最后，留给你两个思考题。

1. 消息存储中，内容表和索引表如果需要分库处理，应该按什么字段来哈希？ 索引表可以和内容表合并成一个表吗?

2. 能从索引表里获取到最近联系人所需要的信息，为什么还需要单独的联系人表呢?

你可以给我留言，我们一起讨论。感谢你的收听，我们下期再见。

上一篇　01 | 架构与特性：一个完整的IM系统是怎样的?

下一篇　03 | 轮询与长连接：如何解决消息的实时到达问题?

精选留言 (30)

写留言

王棕生
2019-08-30

1. 消息存储中，内容表和索引表如果需要分库处理，应该按什么字段来哈希？ 索引表可以和内容表合并成一个表吗?
答： 内容表应该按主键消息ID来哈希做分库分表处理，这样便于定位某一条具体的消息；索引表应该按索引的用户UID来哈希做分库分表处理，这样可以使得当前用户的所有联系人都落在一张表上，减少遍历所有表的麻烦。 ...
展开

作者回复:

23

Julien
2019-08-30

每节课的思考题可以在下一节课开始之前揭晓吗？谢谢。

11

Colin
2019-08-30

请教各位评论区的大神们：消息未读数更新时机大家都是怎么设计的？比如说：发一条消息，未读消息+1这个操作大家都是如何更新的？如果是在数据库更新，再同步到客户端，可能会出现消息到达了，未读数还没更新的现象，如果是在客户端+1，再保存到数据库，别人写个脚本就能把数据给改乱了，不安全。应该如何设计比较合理？

展开

1 2

挨踢菜鸟
2019-08-30

老师，请问websocket如何多实例部署，如何解决实例重启造成连接断开的问题

作者回复: 没太理解到您的意思哈，websocket网关只有是无状态的多实例部署没啥问题的呀；实例重启断开连接是肯定的，需要解决的是断开后客户端需要有重连机制以及如果尽量减少实例重启的概率。

4 2

Geek_9b1a66
2019-08-30

消息存储有什么推荐的数据库吗

展开

作者回复: 这个需要看具体的业务场景吧，比如考虑访问模型，数据量大小，读写的比例等等。在我们自己的场景里mysql和hbase，pika都有在使用。

1

小可
2019-08-30

1.由于索性表与内容表有关联，分库时两张表都应该按内容表id哈希，如果按用户id哈希，如果记录与内容不在一个库，获取消息时还要夸库查询，增加了系统复杂度，也会影响性能；

不能合并，一般IM系统都有群发消息功能，如果内容表合并到索引表，那内容数据冗余就太多了，从而占用存储空间...

展开

作者回复: 这里需要考虑一个问题哈：我们在查询两个人之间的历史消息的时候是用户维度的查询还是消息维度的查询？如果按消息id哈希，查询两个人之间的历史消息在只有uid的情况下该怎么查呢?

2 1

小小小、盘子

2019-08-30

老师，我之前用两个用户id，根据某种规则生成一个唯一会话id，然后外加一个发送人id，这样索引表和内容表用一个表就可以，这么做是否可行？有什么优缺点?

展开

作者回复: 单表单记录也是可行的，只是需要考虑清楚具体的使用场景：比如会话的某一方删除消息该怎么处理？如果业务需要类似邮箱的发件箱和收件箱这种设计是否方便索引查询?

2 1

2102

2019-09-02

索引表数据量太大

展开

asdf100

2019-09-02

对于内容表根据主键字段分库分表的话，如果用户查看最近一段的消息的时候，不是从多个地方分别获取数据再聚合了么，这样也有弊端的吧?

展开

作者回复: 是个好问题，内容表的获取在拿到消息id后可以采取并发获取的方式哈，但是一页一般消息id是有限的（比如20），但分库分表的表数量会很多，上千都很常见。

manymore13

2019-09-02

老师，我做前端的，问两个后台基础问题
1. 图中网关怎么理解？做什么用的？
2.你提的第一个问题中 应该按什么字段来哈希？
这个哈希是用来做什么的？是确定唯一值吗？内容表不是有消息ID吗？

展开

作者回复: 1. 网关就是设备连接到服务端的服务器。
2. 数据量太大的情况下，需要进行分库操作，这里的hash就是应该怎么来分库可以让我们的读写都高效。

iMARS
2019-09-02

按照消息ID取模进行负载分摊，如果按照用户ID会出现分布不均的问题。其次，不建议把消息内容和索引表合并。对于消息内容存储的需求和关系型数据库不一定相同，分离后，可考虑使用非关系型数据库管理，当然会有一些运维和数据关联查询上的需求。

展开

作者回复: 考虑下消息索引表的访问模式哈，对于索引表最终要求的是尽量能提升获取的效率，所以如果需要有索引表，应该是uid维度的查询会更高效一些。

云师兄
2019-09-02

以往的实践中，我们采用uid进行分表，将单个用户的所有消息记录收敛到一个表中，在查询单个用户纬度上消息避免跨表。收发双方之所以都需要采用索引，是因为可能会放到不同的分表中去，消息内容表独立，也是避免冗余存储，特别是群聊时候。常用联系人表相比消息表，一个是只记录最新记录，一个是所有记录。前者更想一个会话纬度，在客户端查询是否消息是最新，避免了查询消息表用户所有记录分组并排序。是否理解有偏差？

展开

作者回复: 👍

Leon

2019-09-02

之间面试被问到怎么设计存储群聊的聊天记录，保证聊天记录的有序性，没答出来，老师帮忙解答下

作者回复: 一般来说群聊消息的存储上主要是考虑存储冗余的问题，一个群的消息只需要存一份即可，每条消息id通过自增序号或者“时间维度相关”的发号器来生成，通过这个id排序即可。群聊相关的话题后续课程还会细讲哈

ly

2019-09-02

最近联系人记录的仅仅是最近有过会话的联系人，而单独的联系人是存储全部联系人。索引表和内容表应该可以合并，这就是一种消息冗余，至于分库的hash的字段，由于没有过这个经验，还真不明白，请老师指点！

展开

作者回复: 索引表和内容表是否可以合并还需要看：第一，一条消息发出后，需要在索引表记录几条记录，如果需要记录多条，那么分开的话会减少冗余，如果只需要记录一条那么是可以合并的，实现上可以根据具体数据的使用场景来确定，分开和合并都是支持的。
数据存储hash方式很多呀，比如说：如果索引和内容单独存，那么内容表可以按消息id哈希，索引表可以按消息所属人uid来hash。

一路向北

2019-09-01

索引表和内容表可以合并。将“是发件箱还是收件箱”字段去除，变成另一个字段，字段值为1,2,3.分别表示仅发方可见、仅收方可见、收发双方都可见。此时，两个UID字段的组合表明了谁是发方谁是收方。

成

2019-09-01

接入服务的高可用性如何保障，服务和服务之间如何高效相互通信，还希望后续多介绍下

作者回复: 有一篇课程专门讲接入服务的高可用，敬请期待。

东东
2019-09-01

老师，群消息是怎么存储的呢，群离线又是怎么拉取的呢，这个可以细化点吗？

作者回复: 后面课程会有讲到的哦

Geek_71a2ee
2019-08-31

老师，你好，请教两个问题，第一im系统里面的账号关联关系业内一般怎么设计，常用的是哪种？第第二个问题，接收方如何跟服务器保持一个长链接，能否给写个例子看看，如果有上亿的在线用户，那服务器压力岂不是很大，服务器，一台服务器也大约能链接多少在线用户？

展开

作者回复: 第一个问题没太看懂哈，账号关联关系具体是指啥？单台服务器如果只是挂连接能支持几百万上千万的，但实际线上由于不仅仅是连接的维护，还有消息的传输，所以单台一般不会挂这么多连接。

飞羽惊鸿
2019-08-30

老师，websocket如果进行分布式部署？分布式部署后服务器之间的通信采用何种方式比较合理？如果统一管理分布式服务下的所有服务

展开

作者回复: 只要是websocket网关是无状态的就可以随意分布式部署的呀，部署后理论上网关服务器不需要进行通信，只是网关服务器和业务服务器需要通信吧。

小胡子
2019-08-30

这样的消息结构，收到一条消息时的写操作是否过多？消息入库、更新联系人最后一条消息、更新未读数

作者回复: 这些步骤也不是一定的，对于不需要多终端消息完全同步的场景，很多操作就可以推给客户端，在端上来实现了。