22 | 答疑篇：分布式体系架构与分布式计算相关问题

2019-11-13 聂鹏程

分布式技术原理与算法解析 进入课程 >

讲述：聂鹏程
时长 13:25　大小 12.30M

你好，我是聂鹏程。今天，我来继续带你打卡分布式核心技术。

到目前为止，“分布式技术原理与算法解析”专栏已经更新 21 篇文章了，我已经为你介绍了分布式技术四纵四横知识体系中的三横，即“布式资源管理”“分布式计算技术”和“分布式通信”，以及四纵中的“分布式协同”和“分布式调度”。

在这里，我首先要感谢你们坚持学习每一篇文章，以及对每一道思考题的积极思考与讨论，并且还在此基础上对类似问题进行了扩展。

比如，@1024、@每天晒白牙、@游弋云端、@Jackey 和 @Dale 等同学，对双主问题展开了激烈的讨论；再比如，@xj_zh、@mt11912、@小白啊、@随心而至等同学，对

Master 如何判断 Slave 是否存活的问题进行了讨论，特别是 @小白啊还专门查询了 Kubernetes 的方法，在留言区进行了回复。

这样的同学还有很多，我就不再一一点名了。今天，我就针对前面文章涉及的与思考题有关的留言，做一次进一步地梳理与分析，以帮助你夯实前面所学的知识点。

留言涉及的问题有很多，但我经过进一步地分析和总结后，发现大家特别感兴趣和有疑惑的思考题主要分为两类：

分布式体系架构中，如何判断节点存活的问题；

分布式计算技术中，离线计算、批量计算、实时计算和流式计算的区别。

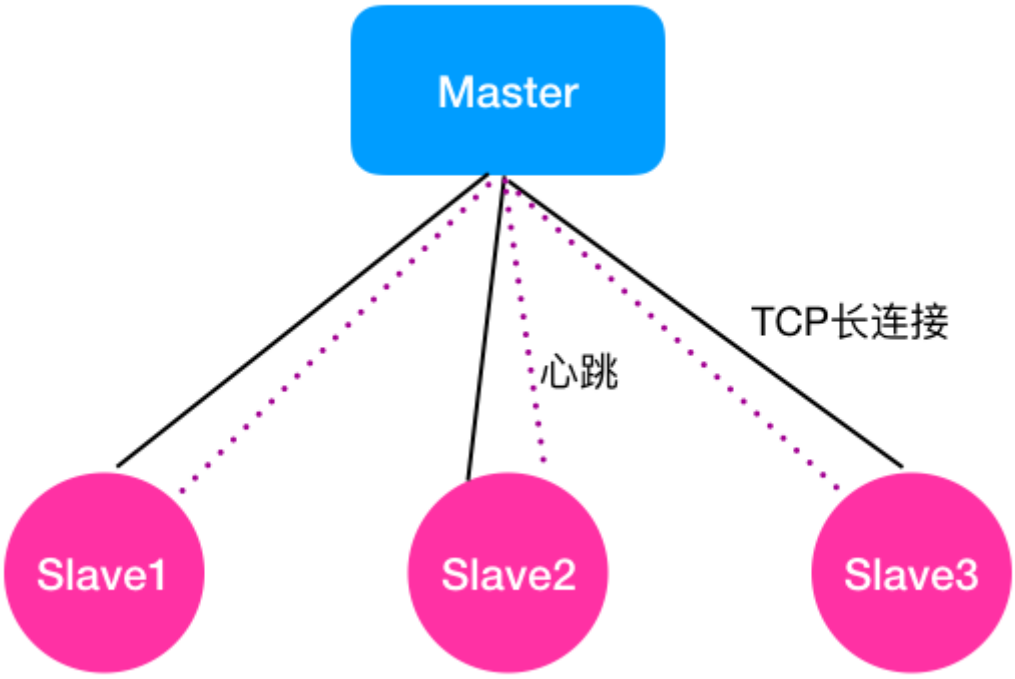
今天，我主要就对这两类思考题进行一下分析和讲解。

分布式体系架构相关问题

在第 9 篇文章“分布式体系结构之集中式结构：一人在上，万人在下”中，我给你留了一个思考题：**在集中式架构中，Master 如何判断 Slave 是否存活呢**？

首先，我先和你说说 Slave 故障的两类情况：一种是 Slave 进程退出，另一种是 Slave 所在服务器宕机或重启了。你可能会说，这两种情况的判断方法，难道还不一致吗？别着急，且听我慢慢道来。

如下图所示，假设 Master 节点与 3 个 Slave 节点相连。请注意，我在图中，Master 与 Slave 之间画了两条线，实线旁写的是 TCP 长连接，虚线旁写的是心跳。因为 Master 与 Slave 之间的监控关系是固定的，因此我用了两种机制协同来判断 Slave 是否存活。



其中，**TCP 长连接就是针对 Slave 进程退出，但是 Slave 所在服务器未故障的情况。**这种方式是借助 TCP 长连接的工作原理进行判断的。TCP 长连接中，TCP 会对对端的 Socket 进行检测，当发现对端 Socket 不可用时，比如不能发出探测包或探测包未收到响应，会返回 -1 的状态，表示连接断开。所以，这种方式可以快速检测到 Slave 进程的退出。

对于 Slave 所在服务器故障的情况，由于服务器宕机或重启，那么系统环境等均不工作了，这种情况 TCP 长连接也无法进行探测了，也就是说 TCP 长连接方法在这种场景下无法判断节点是否故障。

对于这种场景，现有的软件架构中，基本都采用了**心跳方式**。其核心策略是，Master 按照周期性（比如每隔 1s）的方式给 Slave 发送心跳包，正常情况下 Slave 收到 Master 发送的心跳包后，会立即回复一个心跳包，告知 Master 自己还活着。当某个 Slave（比如 Slave1）所在服务器故障后，由于 Slave 无法接收到 Master 的心跳包，也就无法回复了。

因此，Master 也无法接收到这个 Slave（比如 Slave1）的回复信息。通常情况下，**系统会设置一个阈值（一般设置为与心跳周期一致），若超过这个阈值还未收到 Slave 节点的回复，Master 就会标记自己与该 Slave 心跳超时。**

其中，设置阈值的目的是，解决 Slave 故障情况下，Master 一直收不到心跳信息而阻塞在那里等待心跳回复的问题。一般连续 k 次 Master 与 Slave 的心跳超时，Master 就会判断

该 Slave 故障了。其中，设置连续 k 次的目的是，降低因为系统做垃圾回收或网络延迟导致误判的概率。

这里的 k，主要是根据业务场景进行设置的。如果 k 设置得太小，容易导致故障误判率过高，因为系统在做垃圾回收或系统进程正在占用资源时，会阻塞心跳，导致心跳包无法及时回复而超时，从而被误判。如果 k 设置得太大，会导致故障发现的时间过长，因为故障发现时间 =k* 心跳发送周期。

接下来，我们继续延展下这个问题吧。

追问 1：非集中式架构中，如何判断节点是否存活？

集中式架构中，采用了 TCP 连接和心跳协同判断节点是否存活，那么非集中式架构中是否也是这样判断的呢？

其实，**在非集中式架构与集中式架构中，判断节点是否存活的原理有所不同。**因为，非集中式架构中节点之间是对等的，没有 Master 与 Slave 之分。如果每个节点间都建立 TCP 长连接，假设集群中有 n 个节点，那么每个节点均需要与其他 n-1 个节点建长连接，这将导致每个节点的资源占用都会非常多。因此，非集中式架构是采用心跳的方式进行判断的。

这里你可能会问，如果像集中式架构那样，每个节点与其他 n-1 个节点都发送心跳的话，整个集群中同一时间心跳消息为 n*(n-1)，消息量也特别大，甚至会导致网络风暴，应该怎么办。

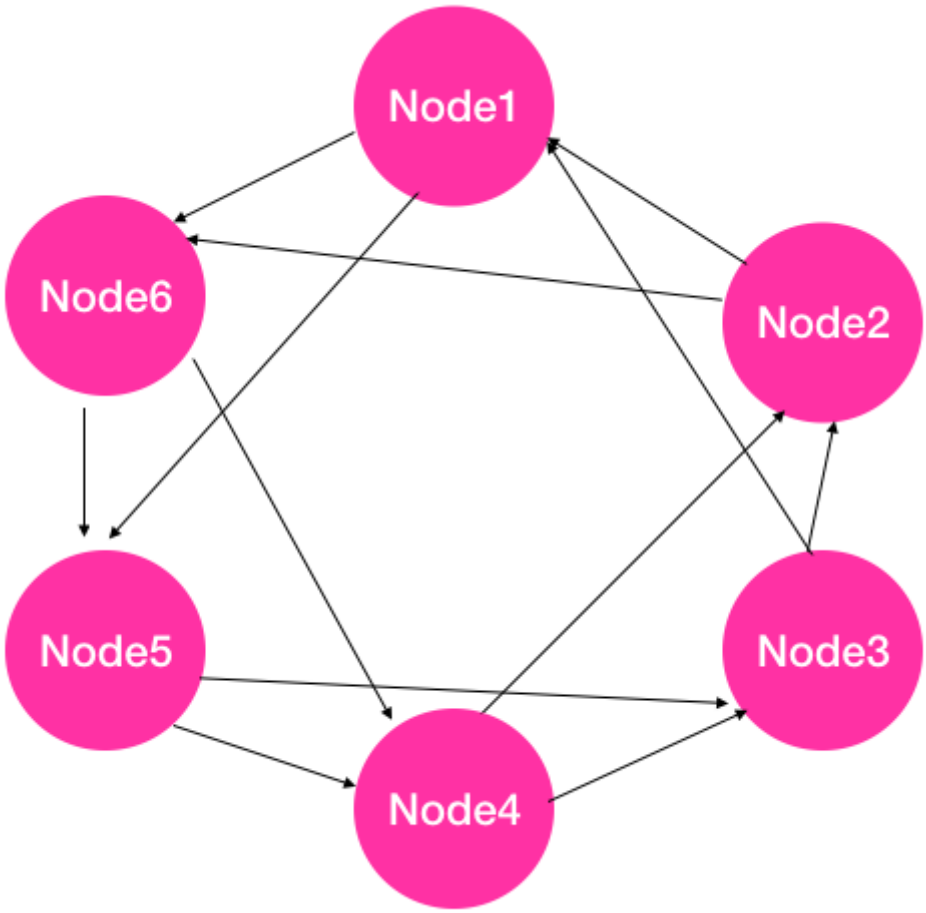
其实，与集中式架构中的心跳包不同，非集中式架构中采用的心跳方式的核心思想是，**每个节点被 b（1≤b<n）个节点监控，以减少心跳信息量。**

接下来，**我们以 Akka 的原理为例，先来看看 b 的取值原则吧**。

> 如果用户不设置 b 的值，那么 b 默认取值的原则是：若集群中节点总数 n 小于 6，b=n-1；若 n 大于等于 6，b=5。
>
> 若用户设置 b 的值，则 b 以用户设置的值为准。

接下来，我们再看看 **Akka 集群中具体是如何通过心跳方式判断节点是否存活的**。

1. Akka 中集群组建完成后，每个节点拥有整个集群中的节点列表。
2. 每个节点根据集群节点列表，计算哈希值（比如根据节点 ID 计算一个哈希值），然后基于哈希值，将所有节点组成一个哈希环（比如，从小到大的顺序），如下图所示。由于每个节点上的计算方法一致，因此虽然每个节点独立计算，但每个节点上维护的哈希环是一致的。
3. 根据哈希环，针对每个节点逆时针或顺时针方向选择 b（图中设置 b=2）个临近节点作为监控节点，比如图中 Node 2 和 Node3 监控 Node1，Node 3 和 Node4 监控 Node2，以此类推。由于每个节点被 b 个节点监控，反过来也可以说，在这个环上每个节点监控 b 个节点，因此具体的实现方式是每个节点按照逆时针或顺时针方向选择 b 个节点进行监控。
4. 当某个节点发现自己监控的节点心跳超时时（比如 Node 2 发现 Node1 心跳超时），则标记该节点不可达（Node2 标记 Node1 不可达），并将该信息通过 Gossip 协议传播给集群中的其他节点。
5. 如果某个节点被标记为不可达之后（比如 Node1 不可达），若不将该节点踢出集群，那么 Node2 和 Node3 仍然会给 Node1 发送心跳，若后面 Node2 又发现 Node1 心跳可达时，则重新将 Node1 更新为可达状态，然后同步给集群中其他节点。



这里的**判断心跳超时机制，可采用集中式方法中的连续 k 次心跳超时的方法进行判断，也可以通过历史心跳信息进行预测。**具体的预测方法，我将在第 31 篇文章“分布式高可用之故障恢复：知错能改，善莫大焉”中做进一步讲解。

追加 2：一个集群为什么会存在双主的场景呢？

上面，我提到判断节点存活的方法主要是通过心跳的方式。如果是因为网络连接断开，那么节点之间就会被误判为对方故障了。在主备场景下，通常会出现双主的情况。这也就是第 4 篇文章“分布式选举：国不可一日无君”的课后思考题答案了。

在主备场景下，正常情况下，主节点提供服务，备节点是对主节点的数据、状态进行备份，以确保主故障后备升主后业务可以正常运行。主备节点之间通常会通过心跳的方式进行检测，目的是监控主节点是否故障，若故障则备升主，保证业务运行。

想象一下，如果主备节点之间的网络连接断开了，那么主节点与备节点之间心跳均不可达，因此主节点会认为备节点故障，此时主节点会继续提供服务，而备节点会认为主节点故障，备升主。所以，集群中就出现了双主的场景。

好了，以上就是关于分布式体系结构中如何判断节点是否存活的相关问题了，相信你对这几个问题有了比较深刻的理解。接下来，我们再看看分布计算技术的相关问题吧。

分布计算技术相关问题

在分布式计算技术中，我们经常会听到离线计算、批量计算、实时计算和流式计算这四个概念，也常常会弄混。那么，**离线计算和批量计算，实时计算和流式计算到底是什么呢？离线计算和批量计算、实时计算和流式计算分别是等价的吗？**

接下来，就请你带着问题，随我一起进入下文。

首先，我们来看一下**离线计算。**通常我们提到的离线计算，主要的应用场景是对时延要求不敏感、计算量大、需要计算很长时间（比如需要数天、数周甚至数月）的场景，比如大数据分析、复杂的 AI 模型训练（比如神经网络）等。

这种场景如果采用在线计算或实时计算的话，通常会存在数据量不够或大量计算影响正在运行的业务等问题，因此往往会采用离线计算的方式。

离线计算方式的核心思想是，先采集数据，并将这些数据存储起来，待数据达到一定量或规模时再进行计算，然后将计算结果（比如离线训练的模型）应用到实际业务场景中。

其次，我们看一下**批量计算。**批量计算通常是指，将原始数据集划分为多个数据子集，然后每个任务负责处理一个数据子集，多个任务并发执行，以加快整个数据的处理。比如，我在第 15 篇文章“分布式计算模式之 MR: 一门同流合污的艺术”中，讲 MR 计算模式时提到，MapReduce 中的 Map 其实就属于批量计算，Map 计算的结果会通过 Reduce 进行汇总。

接下来，我们再看一下**实时计算。**实时计算其实是和离线计算相对应的，离线计算对时延要求不敏感，相反，实时计算对时延的要求比较敏感。这种模式需要短时间执行完成并输出结果，比如秒级、分钟级，也就是说强调时效，通常用于秒杀、抢购等场景。实时计算由于时延要求低，因此计算量通常不大、数据量也不会太多，所计算的数据往往是 K、M 级别的。

最后，我们在看看**流式计算。**我在第 16 篇文章“分布式计算模式之 Stream: 一门背锅的艺术”中，与你讲述了流式计算。流计算强调的是实时性，数据一旦产生就会被立即处理，当一条数据被处理完成后，会立刻通过网络传输到下一个节点，由下一个节点继续处理。这种模式通常用于商业场景中每天的报表统计、持续多天的促销活动效果分析等。

为了便于你理解与记忆，我将这四种计算模式的特点，汇总为了一张表格，如下所示。

	离线计算	批量计算	实时计算	流式计算
执行时延	时延不敏感，计算耗时很长（比如需要数天、数周甚至数月）	时延不敏感，计算时间较长，小时级	时延敏感，计算时间通常为秒级、分钟级	时延敏感，计算时间较短，通常是分钟级
数据规模	大规模数据	较大规模数据	小规模数据	小规模流式数据
分类维度	计算时延	计算方式	计算时延	计算方式
应用场景	大数据分析、复杂的AI模型训练（比如神经网络）	分布式排序、倒序索引构建等场景	秒杀、抢购等场景	每天的报表统计、持续多天的促销活动效果分析等场景

通过对这四种计算模式的讲解，相信你已经发现了，离线计算和批量计算对任务执行的时延不是特别敏感，而实时计算和流式计算对任务执行的时延敏感。但，离线计算和实时计算是

从计算时延的维度进行分类的，而批量计算和流式计算是从计算方式的维度进行分类的，因此我们**不能将离线计算和批量计算直接等同，也不能将实时计算和流式计算直接等同。**

总结

我把前面 21 篇文章中，大家针对思考题的讨论和困惑，筛选出了分布式系统架构中如何判断节点是否存活，以及四种分布式计算模式的异同，做了进一步展开，梳理成了今天的这篇答疑文章。

如果还有哪些思考题或者留言问题，还没来得及扩展的话，你可以留言给我，后续我会再找机会进行解答。最后，我要和你说的是，和我一起打卡分布式核心技术，一起遇见更优秀的自己吧。

篇幅所限，留言区见。

我是聂鹏程，感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎你把这篇文章分享给更多的朋友一起阅读。我们下期再会！

上一篇　21 | 分布式通信之消息队列：货物目取

下一篇　特别放送 | 那些你不能错过的分布式系统论文

精选留言 (8)

写留言

simon
2019-11-14

对于 Slave 所在服务器故障的情况，由于服务器宕机或重启，那么系统环境等均不工作了，这种情况 TCP 长连接也无法进行探测了，也就是说 TCP 长连接方法在这种场景下无法判断节点是否故障。
为什么？？？难道重启或岩机，还能tcp长连接？反过来，心跳也能检查进程退出吧，因为如果进程退出也不会有心跳吧

展开

1

Jackey
2019-11-13

关于检测存活还是有些疑问。长连接为什么不能检测机器故障呢？服务器宕机时长连接也会断开的吧。
另外非集中式的下线条件是过半数以上机器标记为不可达才会认为机器下线对吗？下线后监控的顺序会发生改变吗？

展开

5　1

tt
2019-11-13

判断节点是否存活的方法中，基于长链接的方法是利用了TCP层本身的机制；而基于心跳的方式是基于应用层自己的方法去实现。

如果用OSI模型来描述，就是前者是四层的存活检测，后者是七层的存活检测。

展开

1

Dylan
2019-11-16

那假设出现双主的场景，一般是通过什么有效策略去解决呢，或者有没有好的办法尽量去避免这种情况

张先生
2019-11-15

我以为就我看不到心跳检测，原来大家都有疑问。为什么TCP断开就判断是程序挂了，心跳就是服务器挂了？？不明白

展开

随心而至
2019-11-13

非集中式的心跳机制好多是基于Gossip协议做的，比如consul，redis。

安排
2019-11-13

tcp长连接和心跳那块儿不是很理解，我觉得探测slave进程故障才应该用心跳，探测机器故障长连接是可以的。
slave进程如果还活着但是死锁卡住了，其实长连接是探测不到的，内核协议栈会正常回复探测报文。slave进程如果异常退出，但是操作系统没死，这时socket关闭时有机会发fin，master会收到fin（不考虑网络丢失），所以可以很快知道slave进程死了。如果slave所...

展开

1

忆水寒
2019-11-13

收获满满

展开