BitTorrent 协议及其相关技术

58121124 张博彦

2023 年 5 月 30 日

1 BitTorrent 协议出现的背景

在传统的场景下, 用户希望下载一个文件, 一般都会通过比如 HTTP/FTP 的方式从目标站点的服务器上下载, 服务器的带宽通常都是有限的, 当同时下载的用户过多时, 将超出服务器的带宽限制, 这时用户便会发现文件下载地很慢甚至是无法继续下载, 而 BitTorrent（简称 BT）协议便是为了解决这个问题, 该协议将文件分片, 每个终端用户下载文件分片, 在下载的同时也会互相分发自己已下载的文件分片给其它正在下载的用户, 从而将部分原本应从服务器拉取数据所造成的带宽压力分散给了终端用户, 所有正在同时下载同一文件的终端用户构成一个图结构, 互相点对点式地分发文件片段, 因此对于大文件、多人同时下载时有非常好的文件分发效率。

2 BitTorrent 的架构

BitTorrent 协议的实现离不开四个个极其重要的部分：Tracker 服务器，DHT 网络，.torrent 文件以及 Peer Protocol，以下对这四个个部分进行具体介绍。

2.1 Tracker 服务器

对于纯 BT 协议来说，每个 BT 网络中至少要有一台 Tracker 服务器（追踪服务器），tracker 主要基本工作有以下三个方面：

1. 记录种子信息
2. 记录节点信息

3. 计算并返回节点列表给 BT 客户端

而用户进行做种功能时，就会把种子信息记录到 tracker 服务器上。当其他用户用相关的 BT 软件打开种子后，BT 软件会对种子进行解析，主要得到种子的相关信息，包括：文件名、文件大小、tracker 地址等。然后相关的 BT 软件会向 tracker 地址发送请求报文，开始进行下载。BT 向 tracker 发送的是 Get 请求，请求的内容主要有以下九个方面：

1. info_hash: 种子文件 info 字段的 SHA1 值
2. peer_id: 节点标识，由 BT 客户端每次启动时随机生成
3. port: 节点端口，主要用于跟其他节点交互
4. uploaded: 总共上传的字节数
5. downloaded: 总共下载的字节数
6. left: 文件剩余的待下载字节数
7. numwant:BT 客户端期望得到的节点数
8. ip: 该信息并非必须请求的内容，因为 Tracker 可以得到请求的 IP 地址，不需要客户端直接上传
9. event:event 一共有四种形式：started、stopped、completed 以及空。当 BT 客户端开始种子下载时，第一个发起的请求为 started，在下载过程中，该值一直为空，直到下载完成后才发起 completed 请求。做种过程中，发送的 event 也为空。如果 BT 客户端停止做种或退出程序，则会发起 stopped 请求。

当 tracker 收到该请求后，进行以下几步处理：

1. 根据 info_hash 查找种子信息，如果 tracker 没有该种子的任何信息，tracker 服务器返回错误或返回 0 个种子数.
2. 如果 tracker 找到了种子信息，接下来就会去查找是否数据库中已存在该 peer_id 的节点。接下来根据 event 的值进行相关处理。
3. 如果 event 是 stopped，说明该节点已不可用，系统会删除 tracker 上关于该节点的记录信息。

4. 如果 event 是 completed，说明种子节点 +1，非种子-1。

5. 如果 event 是 started，说明这是种子第一次连接 tracker，tracker 需要记录该节点信息。此外如果 left=0，说明这是一个种子节点。

6. 如果 event 是空，则说明节点正在下载或上传，需要更新 tracker 服务器上该节点的信息。

7. 最后 tracker 从本地挑选出 numwant 个节点信息返回给 BT 客户端，实际返回的节点数不一定等于 numwant，但 tracker 会尽量达到这个数量。

2.2 DHT 网络

2.2.1 DHT 网络中节点的构成及通讯流程

DHT 网络中每一个节点有一个全局的唯一标识，称为 node ID（节点 id），节点 id 是随机从文件中的 160 位 hash 中随机抽取的。Distance metric（距离度量）用来比较两个节点 id 或者节点 id 与 hash 之间的距离。所有的节点保存一个路由表（该路由表对于 DHT 网络极其重要，之后会进行详细的叙述）以及它和 dht 网络中一小部分节点交流的信息。在自身节点的信息中，其他节点 id 离自己的节点 id 越近，保存在自己节点中的其它节点 id 的信息就越详细。所有的节点必须知道离自身较近的其它节点，而距离自己很远的节点仅需要有足够的握手信息即可。

在 Kad 算法中，距离度量是对两个 hash 值进行异或运算，并且把结果转换成无符号整数，使用公式表示即为 distance(A,B) = |A xor B|（xor 为异或运算符），该 distance 值越小，表示 A 与 B 的距离越近。

当一个节点想找到一个文件的相邻节点信息时，就使用距离算法把文件的 hash 字段和它自己路由表中的节点 id 进行比较，然后与距离最近的节点进行通信，向它们发送请求获取正在下载这个文件的 peer 节点列表的信息。如果它请求的节点保存有该文件的 peer 节点列表，则把 peer 节点列表返回给发送请求的节点。如果没有，则它必须返回自己路由表中离文件 hash 最近的节点列表给请求者。原始节点不断迭代的发送请求直到找到离目标文件 hash 更近的节点。请求搜索结束之后，下载节点把 peer 节点的信息保存在自己的路由表里面。

2.2.2 DHT 网络中路由表的结构

每一个节点都拥有一个路由表保存一些已知的通信好的节点。路由表中的节点通常用来作为起始节点，当其它节点向这个节点发送请求时，路由表中的这些节点就会被返回给发送请求的结点。

路由表覆盖从 0 到 2^{160} 的完整节点 ID 空间。路由表又被划分为 k 个 buckets(简称 K 桶), 每一个 bucket 包含一个子部分的节点 ID 空间。一个空的路由表只有一个 bucket，该 bucket 中的 ID 范围为 min=0 到 max=2^{160}。当一个节点 ID 为 n 的节点插入到表中时，它将被放到 ID 范围在 min< N < max 的 bucket 中。一个空的路由表只有一个 bucket 所以所有的节点都将被放到这个 bucket 中。每一个 bucket 最多只能保存 K 个节点。当一个 bucket 放满了节点之后，将不再允许新的节点加入，除非用户自身的节点 ID 在这个 bucket 的范围内。若出现这样的情况，则该 bucket 将被分裂为 2 个新的 buckets，每一个新 bucket 的范围都是原来旧 bucket 的一半。原来旧 bucket 中的节点将被重新分配到这两个新的 buckets 中。

当遇到有新节点但 buckets 装满了节点时，便会采用新的方式进行处理。若 buckets 装满了正常的节点时, 那么新的节点将被丢弃。但, 若 bucket 中存在一个或多个节点出现异常，不再能正常工作，那么路由表就用新的节点替换异常节点。如果 bucket 中有规定时间 (一般会设置为 120 秒) 内都没有活跃过的节点，则该节点便为异常节点，这时我们向最久没有联系的节点发送 ping 命令。如果被 ping 的节点给出了回复，那么我们向第二久没有联系的异常节点发送 ping，不断这样循环下去，直到有某一个节点没有给出 ping 的回复，或者当前 bucket 中的所有 nodes 都是再次变为正常节点。

2.3 .torrent 文件

.torrent 文件采用 bencode 编码格式，该格式有以下四种数据类型及其编码格式:

1. string: <length>:<string>。
2. interger: i<int>e
3. list:l[data1][data2][···]e
4. dictionary:d[key1][value1][key2][value2][···]e

整个.torrrent 文件为一个字典类型，包含以下个键:

1. info: 该 BT 种子的文件信息,类型为 dictionary,其中包含 piece length, name，files 等数据

2. announce:tracker 服务器的地址，类型为 string

3. announce-list: 可选 tracker 服务器地址，类型为 list

4. creation date: 文件创建时间，类型为 interger

2.4 Peer Protocol

Peer Protocol 是当前正在下载同一资源的对等结点 (peer) 之间进行数据传输使用的协议,BT 协议中的 Peer Protocol 基于 TCP 或 UTP, 在使用 BT 协议下载时, 所有正在下载同一资源的结点都是对等的, 结点之间相互建立的连接也是对等的。对等结点建立的连接的数据传输方向是双向的, 数据可以由任何一端发往另一端, 对等结点每接收到一个完整的片段（piece）之后, 接收方便计算该 piece 的 SHA1 哈希并与.torrent 文件中 info 字段中的相应分段做对比, 若哈希值相等, 则说明该分片传输成功, 此时这两个对等结点都拥有了资源文件关于该片段的数据, 在实际的传输过程中, 片段会进一步分为固定大小的片（slice）, 对等结点使用片作为最基本的数据传输单位, 接收到一个片段的所有片之后便计算 SHA1 哈希进行对比。

对等结点在建立完传输层连接之后, 便开始进行 Peer Protocol 的握手, 握手消息含有以下 5 个字段的数据:

1. pstrlen: 十进制格式, 长度为 19。

2. pstr: 该值为"BitTorrent protocol"。

3. reserved:BT 协议的保留字段, 用于以后的扩展, 一般将这该段字节的值全部设置为 0。

4. info_hash: 与请求 BT Tracker 时发送的 info_hash 参数值相同。

5. peer_id: 与请求 BT Tracker 时发送的 peer_id 参数值相同。

对等结点一方在传输层连接建立以后便发送握手信息, 另一方收到握手信息后也回复一个握手信息, 任何一方当收到非法的握手信息 (pstrlen 长度不为 19，pstr 不为 “BitTorrent protocol” 等情况)，则立即断开连接。

若校验没有问题, 则握手成功。接着对等双方开始进行数据传输, 数据的通用格式为 length prefix + type id + payload, 其中 length prefix 是消息的长度, length prefix 为 0 表示 Keep Alive 消息,BT 协议标准规定 Keep Alive 消息每两分钟发送一次, 接收端忽略该消息即可, 对于其它类型的消息, 其对应的 type id 如下:

1. type id = 0, 1, 2, 3: 分别为 choke、unchoke、interested, not interested 消息, 此四种消息没有都 payload,length prefix 都为 1。

2. type id = 4: have 消息,have 消息的 paylod 只包含一个整数, 该整数对应的是该终端刚刚接收完成并校验通过 SHA1 哈希的片段索引, 终端在接收到并校验完一个片段后, 就向它所知道的所有 peer 都发送 have 消息以宣示它拥有了这个片段, 其它终端接收到 have 消息之后便可以知道对方已经有该片段的文件数据了, 因而可以向其发送 request 消息来获取该片段。

3. type id = 5: bitfield 消息,bitfield 消息只作为对等结点进行通信时所发送的第一个消息, 即在握手完成之后, 其它类型消息发送之前。若文件没有分片, 则不发送该消息, 它的 payload 是一个字节序列, 逻辑上是一个 bitmap 结构, 指示当前该终端已下载的文件分片, 其中第一个字节的 8 位分别表示文件的前 8 个分片, 第二个字节的 8 位分别表示文件的第 9 至 16 个分片, 以此类推, 已下载的分片对应的位的值为 1, 否则为 0, 由于文件分片数不一定是 8 的整数倍, 所以最后一个分片可能有冗余的比特位, 对于这些冗余的比特位都设置为 0。

4. type id = 6: request 消息, 该消息用于一方向另一方请求文件数据, 其 payload 含有 3 个字段, 分别是 index,begin 和 length。其中 index 指示文件分片的索引,begin 指示 index 对应的片段内的字节索引, length 指定请求的长度,length 值一般取 2 的整数次幂, 现在所有的 BitTorrent 实现中,length 的值都取 16KB,BT 协议规定结点通过随机的顺序请求下载文件片段。

5. type id = 7: piece 消息, 该消息是对 request 消息的响应, 即返回对应的文件片段, 其 payload 与 request 消息大致相同, 仅其中的片段为 request 消息请求的片段。

6. type id = 8：cancel 消息,cancel 消息与 request 消息的 payload 字段完全相同, 但作用相反, 用于取消对应的下载请求。

3 BitTorrent 的工作流程

BitTorrent 协议把提供下载的文件虚拟分成大小相等的块，块大小必须为 2k 的整数次方，该操作并不会再硬盘上产生各个块文件，而仅仅是虚拟分块，并把每个块的索引信息和 Hash 验证码写入.torrent 文件中，作为被下载文件的“索引”。

用户下载时，BT 客户端首先解析.torrent 文件得到 Tracker 地址，然后连接 Tracker 服务器。Tracker 服务器回应下载者的请求，提供用户其他下载者（包括发布者）的 IP。同时，BT 客户端也可通过解析.torrent 文件得到节点路由表，然后连接路由表中的有效节点，由网络节点提供下载者其他下载者的 IP。之后用户连接其他下载者，根据.torrent 文件，两者分别告知对方自己已有的块，然后交换对方没有的数据。下载者每得到一个块，需要算出下载块的 Hash 验证码与.torrent 文件中的对比，如果一样则说明块正确，不一样则需要重新下载这个块。

4 总结

以上便是 BitTorrent 协议实现原理和过程的简单介绍。随着互联网的发展，BitTorrent 协议已经为数不胜数的用户所采纳，但同时传输文件时稳定性较差，盗版、违法内容传播风险等问题也日益明显，想要真正将 BT 下载管理好、使用好，这些问题的解决刻不容缓。