Beanstalk 协议

千石（淘宝网）
qianshi@taobao.com

约定

本文是对 beanstalk 协议的翻译，原文见这里。 如果在熟悉协议之前想先了解一下 beanstalk 的基本概念和系统特性，可以参考我之前写的一篇 blog。

由于翻译总是会和原文有偏差，所以对一些核心的概念词汇将不予翻译，读者可以仁者见仁，智者见智。这些词汇包括：

beanstalk beanstalkd producer consumer tube job

有些词汇本文给出了翻译，但只作为参考，在在这里注明：

reserve 获取 release 释放 bury 休眠

touch 触碰 peek 窥视 kick 踢

所有的请求和响应都以下面的格式标记：

request ***response***

对于和统计相关的三个命令：stats-job, stats-tube, stats 本文未给出相应的翻译，有想了解的读者请参考原文。

篇幅虽短，但文中难免会有错误，发现了请发送到我的邮箱（qianshi@taobao.com），或者在微博（@淘宝千石）上给我留言，谢谢。

协议说明

Beanstalk 协议使用 ASCII 编码，运行在 TCP 协议之上。客户端负责主动建立连接，发送命令和数据，等待响应以及关闭连接。对于每个连接，服务端以接收请求的顺序串行地处理命令，并按相同的顺序发送响应。协议中所有的数字都是十进制并且非负的（除非有明确说明）。

协议中的名字都是 ASCII 字符串。名字可以包含字母（A-Z 和 a-z）、数字（0-9）、连字符（"-"）、加（"+"）、斜线（"/"）、分号（";"）、点（"."）、美元符号（"$"）、下划线（"_"）和括号（“(” 和")"），但是不能以连字符开头。名字以空格或换行结束。每个名字至少要有一个字符的长度。

协议中包含两类数据：文本行和非结构化的块数据。文本行用来传输客户端的命令和服务端的响应。块数据用来传输 job 体和统计信息。每个 job 体是一个不透明的字节序列。服务端从不检查或修改 job 体，直接按其原有的格式返回。job 体的解释是有客户端来进行的。

当不再需要使用服务端时，客户端可以选择发送 ”quit” 命令或直接关闭 TCP 连接。然而，beanstalkd 可以很好处理和维护大量打开的连接，所以，客户端应该尽可能包括连接的打开和重用。这同样也可以省去建立 TCP 连接的开销。

如果客户端违背了协议（比如发送错误格式的请求或者不存在的命令）或者服务端出错，服务端将返回下面信息的一种：

- ***“OUT_OF_MEMORY\r\n”*** 表示服务端不能为 job 分配足够的内存。客户端应该稍后重试。
- ***“INTERNAL_ERROR\r\n”*** 表示服务端的 bug。这种情况不应该发生。如果发生了，可以报给 http://groups.google.com/group/beanstalk-talk。
- ***"BAD_FORMAT\r\n"*** 表示客户端发送了一个错误格式的命令行。可能呢导致这种错误发生的情况有：行没有以\r\n 结尾、在需要整数的地方出现了非数字字符、参数的数目错误、或者其他格式的命令行错误。
- ***"UNKNOWN_COMMAND\r\n"*** 表示客户端发送了一个服务端不认识的命令。

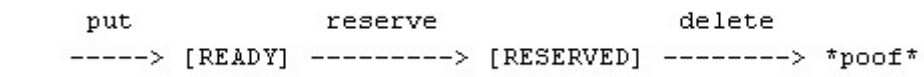
这些错误响应不会在本文以下各节的命令列出，但都隐含在所有命令的说明里。客户端在发送命令之后都要准备好接收错误响应。

最后，如果服务端发生了严重的错误不能继续服务，服务端将会主动关闭连接。

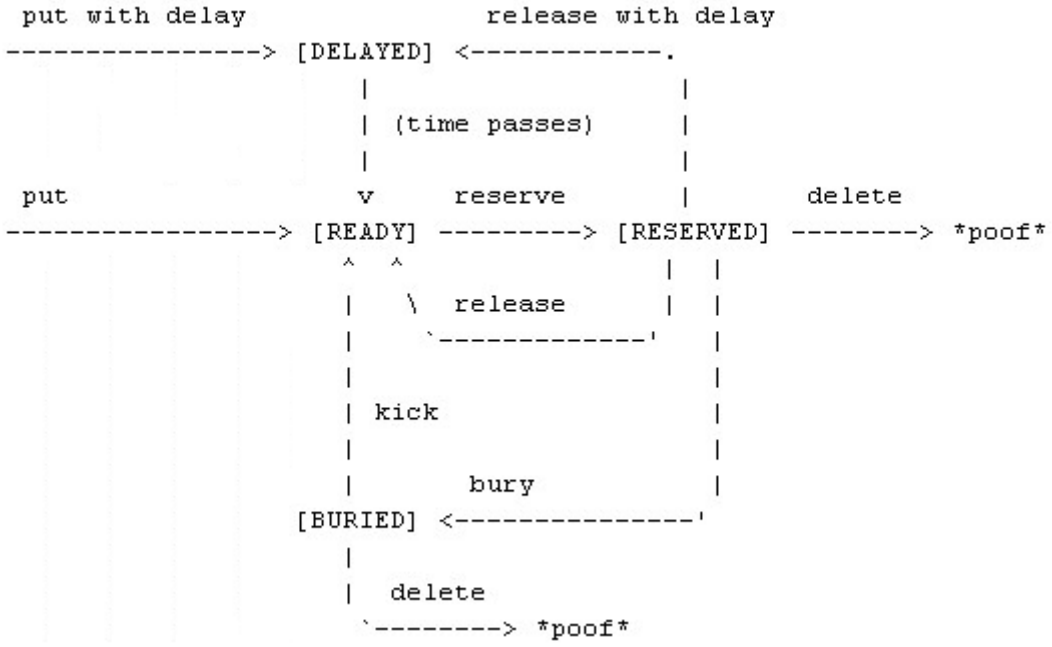
job 生命周期

Beanstalk 中的 job 由客户端发送的 ”put” 命令创建。在整个的生命周期中，一个 job 可以经历 “ready”, “reserved”, “delayed” 和 ”buried” 四种状态。”put”命令之后，一个 job 通常处于 “ready” 状态。它将在 ready 队列等待 worker 执行 “reserve” 命令。如果这个 job 是 ready 队列的下一个，它将由 worker 来处理。这个 worker 将执行这个 job，当 worker 完成了该 job，可以发送 “delete” 命令来删除 job。

下图是一个简单的 job 生命周期：



下面的图片说明了更复杂的情况：



系统可以有一个或多个 tube。每个 tube 包含一个 ready 队列和一个 delay 队列。一个 job 的生命周期只能在一个 tube 中。Consumer 可以发送 “watch” 命令来关注一个 tube，也可以发送 “ignore”命令来忽略一个 tube。被关注的 tube 将会出现在 consumer 的 “watch list”里面。客户端获取的 job 可能来自其 “watch list” 中任意一个 tube。

当客户端刚建立连接，其 “watch list” 中只有初始设置的 “default” tube。如果客户端提交 job 前没有发送 “use”命令，job 将被存储在 “default” tube 中。

Tube 是在被引用到时按需创建的。如果一个 tube 空了（也就是说不包含任何 ready, delayed 或者 buried job）并且没有客户端引用，该 tube 将被删除。

Producer 命令

put

“put” 命令用来向队列中插入一个 job，包括一个命令行和一个 job 体：

put <pri> <delay> <ttr> <bytes>\r\n
<data>\r\n

“put” 命令将向客户端现在正在使用的 tube 中插入一个 job（”use” 命令见下文）。

- <pri> 是一个小于 2**32 的整数。小优先级数值的 job 将会排在大优先级数值的 job 前面执行。最高优先级是 0，最低优先级是 4,294,967,295。
- <delay> 是一个整形数，表示将 job 放入 ready 队列需要等待的秒数。
- <ttr> --time to run—是一个整形数，表示允许一个 worker 执行该 job 的秒数。这个时间将从一个 worker 获取一个 job 开始计算。如果该 worker 没能在<ttr> 秒内删除、释放或休眠该 job，这个 job 就会超时，服务端会主动释放该 job。最小 ttr 为 1。如果客户端设置了 0，服务端会默认将其增加到 1。
- <bytes> 是一个整形数，表示 job 体的大小，不包括结尾的”\r\n”。这个值不能大于 max-job-size（默认值为 2**16）。
- <data> 及 job 体，是一个长度为<byetes> 的字符序列。

在发送命令行和 job 体之后，客户端等待如下的响应：

- ***"INSERTED <id>\r\n"*** 表示成功。<id> 是新 job 的数字编号。
- ***"BURIED <id>\r\n"*** 表示服务端没有足够的内存将 job 添加到优先级队列中。<id> 是新 job 的数字编号。
- ***"EXPECTED_CRLF\r\n"*** 表示该 job 必须以 CR-LF 对，及 ”\r\n” 结尾。这两个字节在客户端的 put 命令中将不会被计算到 job 大小中。
- ***"JOB_TOO_BIG\r\n"*** 表示客户端发送一个超过 max-job-size 字节的 job 体。
- ***"DRAINING\r\n"*** 表示服务端已进入”drain mode”，将不会再接收新的 job。客户端应该超时其他的服务端或者断开连接稍后重试。

use

“use” 命令是为 producer 设计的。put 命令会将 job 放入由该命令指定的 tube 中。如果没有执行 use 命令，job 将会被放入”default” tube。

use <tube>\r\n

tube 是一个不超过 200 字节的名字。通过名字来指定使用哪个 tube。如果该 tube 不存在，就会被自动创建。

唯一的响应是：

USING <tube>\r\n

<tube>是现在正在使用的 tube 的名称。

Worker 命令

reserve

一个进程可以通过”reserve”, “delete”, “release”和”bury”命令来从队列里面消耗 job。”reserve”命令如下：

reserve\r\n

或者，可以指定一个超时时间：

reserve-with-timeout <seconds>\r\n

beanstalkd 将一直等待发送响应，直到一个 job 可用。当一个 job 被客户端获取，客户端必须在 TTR 时间内完成该 job，否则 job 就会超时。当该 job 超时，服务端会将该 job 重新放回 ready 队列。TTR 和当前剩余时间都可以在 stats-job 命令的响应中找到。

如果超时设置为 0，服务端将立刻返回一个响应或者 TIME_OUT。如果超时是一个大于 0 的值，客户端将会在获取 job 请求中阻塞，直到一个 job 可用或者超时。

对于一个已经被获取 job 的 TTR，最后一秒是作为安全边际被服务端保留的，在此期间，客户端将无法试图去等待别的 job。如果客户端在安全边际期间发送获取 job 命令，或者安全边际到来时还在等待获取命令的完成，服务端将返回：

DEADLINE_SOON\r\n

这给了客户端一个在服务端自动发布该 job 之前，删除或者重新发布已经获取 job 的机会。

TIMED_OUT\r\n

当指定了一个非负的超时时间值，并且在超时时或没有一个 job 可用，服务端将返回 TIMED_OUT。

否则，这条命令唯一成功的响应形式是一个文本行后跟一个 job 体。

RESERVED <id> <bytes>\r\n

<data>\r\n

- <id> job 的 id，在这个 beanstalkd 的实例中，代表该 job 的唯一整数。
- <bytes> 表示 job 体大小的整数，不包括结尾的 ”\r\n”。

- ✧ <data> 表示消息体，一个长度为 <bytes> 的字节序列。这是对之前 put 命令 job 体原始字节的逐字拷贝。

delete

delete 命令从服务端完全删除一个 job。当客户端已经成功执行完一个 job 时，该 job 一般会被使用。客户端可以删除一个已经被获取的 job，可用的 job，以及被休眠的 job。delete 命令如下：

delete <id>\r\n

<id> 表示要删除的 job id。

客户端将等待如下的响应行：

- ✧ ***"DELETED\r\n"*** 表示已经成功删除。
- ✧ ***"NOT_FOUND\r\n"*** 表示该 job 不存在，或者该 job 没有被客户端获取，已经不在 ready 或 buried 状态。这种情况发生这客户端发送 delete 命令之前，该 job 已经超时。

release

release 命令将一个已经被获取的 job 重新放回 ready 队列（并将 job 状态置为 “ready”），让该 job 可以被其他客户端执行。这个命令经常在 job 因为短暂的错误而失败时使用。格式如下：

release <id> <pri> <delay>\r\n

- ✧ <id> 表示要 release 的 job id。
- ✧ <pri> 表示给该 job 分配的新的优先级。
- ✧ <delay> 表示在该 job 被放入 ready 队列之前需要等待的秒数。在此期间，job 的状态将是”delayed”。

客户端期待的响应如下：

- ✧ ***"RELEASED\r\n"*** 表示成功
- ✧ ***"BURIED\r\n"*** 表示服务端没有足够的内存将 job 添加到优先级队列中。
- ✧ ***"NOT_FOUND\r\n"*** 表示该 job 不存在或没有被客户端获取。

bury

bury 命令将一个 job 的状态置为”buried”。Buried job 被放在一个 FIFO 的链表中，在客户端调用 kick 命令之前，这些 job 将不会被服务端处理。

bury 命令格式如下：

bury <id> <pri>\r\n

- ✧ <id> 表示该 job 的 id
- ✧ <pri> 表示分配给该 job 新的优先级

有两种可能的响应：

- ✧ ***"BURIED\r\n"*** 表示成功
- ✧ ***"NOT_FOUND\r\n"*** 表示该 job 不存在或没有被客户端获取。

touch

touch 命令允许一个 worker 请求在一个 job 获取更多执行的时间。这对于那些需要长时间完成的 job 是非常有用的，但同时也可能利用 TTR 的优势将一个 job 从一个无法完成工作的 worker 处移走。一个 worker 可以通过该命令来告诉服务端它还在执行该 job（比如：在收到 DEADLINE_SOON 是可以发生给命令）。

touch 命令格式如下：

touch <id>\r\n

✧ <id> 表示当前连接获取 job 的 id

有两种可能的响应：

✧ ***" TOUCHED\r\n"*** 表示成功

✧ ***"NOT_FOUND\r\n"*** 表示该 job 不存在或没有被客户端获取。

watch

watch 命令将一个 tube 名称加入当前连接的 watch list。reserve 命令将从 watch list 里面的任意一个 tube 中获取 job。对于每个新建的连接，watch list 中只有一个”default” tube。

watch <tube>\r\n

✧ <tube> 是一个不超过 200 字节的名称。它指定将一个 tube 加入到 watch list。如果该 tube 不存在，将会被及时创建。

响应是：

WATCHING <count>\r\n

✧ <count> 表示当前 watch list 中的 tube 数目。

ignore

ignore 命令是为 consumer 设计的。ignore 命令将一个 tube 从当前连接的 watch list 中移除。

ignore <tube>\r\n

响应如下：

✧ ***"WATCHING <count>\r\n"*** 表示成功。<count>表示当前 watch list 中的 tube 数目。

✧ ***"NOT_IGNORED\r\n"*** 表示客户端正在试图 ignore watch list 中的最后一个 tube。

其他命令

peek

peek 命令可以让客户端检查系统中的 job。这个命令有 4 个变种。除第一个命令之外，其他 3 个命令都只能作用在当前使用的 tube 上面。

"peek <id>\r\n" 返回编号为<id>的 job。

"peek-ready\r\n" 返回当前 tube 下一个 ready 的 job。

"peek-delayed\r\n" 返回当前 tube 剩余 delay 时间最短的 job。

"peek-buried\r\n" 返回当前 tube buried list 中下一个 job。

有两种可能的响应，其中一个是单行：

"NOT_FOUND\r\n" 表示请求的 job 不存在，或者没有 job 在当前请求的状态队列中。

还有一个是一行跟一个块数据，表示命令执行成功：

FOUND <id> <bytes>\r\n

<data>\r\n

- <id> 表示 job 的 id。
- <bytes> 表示 job 体大小的整数，不包括结尾的 ”\r\n”。
- <data> 表示消息体，一个长度为 <bytes> 的字节序列。

kick

kick 命令只能针对当前正在使用的 tube 执行。它将 buried 或者 delayed 状态的 job 移动到 ready 队列。命令格式如下：

kick <bound>\r\n

- <bound> 表示每次 kick job 的上限，服务端将最多 kick <bound>个 job。

响应格式如下：

KICKED <count>\r\n

- <count> 表示本次 kick 操作作用 job 的数目。

list-tubes

list-tubes 命令返回已经存在的所有 tube 的列表。格式如下：

list-tubes\r\n

响应如下：

OK <bytes>\r\n

<data>\r\n

- <bytes> 是<data>的字节大小。
- <data>是一个长度为<bytes> 的字节序列。它是一个包含所有 tube 名称的 YAML 文件。

list-tube-used

list-tube-used 命令返回客户当前正在使用的 tube。格式如下：

list-tube-used\r\n

响应如下：

USING <tube>\r\n

✧ <tube> 是正在使用的 tube 名称。

list-tubes-watched

list-tubes-watched 命令返回客户端当前正在关注的 tube 名称列表。格式如下：

list-tubes-watched\r\n

响应如下：

OK <bytes>\r\n

<data>\r\n

✧ <bytes> 是<data>的字节大小。

✧ <data>是一个长度为<bytes> 的字节序列。它是一个包含被关注 tube 名称的 YAML 文件。

pause-tube

pause-tube 命令用来在给定时间内暂停从 tube 获取 job。格式如下：

pause-tube <tube-name> <delay>\r\n

✧ <tube> 表示要暂停的 tube。

✧ <delay> 表示在可以从 tube 获取 job 之前需要等待的秒数。

有两种可能的响应：

"PAUSED\r\n" 表示成功。

"NOT_FOUND\r\n" 表示该 tube 不存在。

quit

quit 命令用来关闭当前连接。格式如下：

quit\r\n