**Push Or Pull**

无论是消息系统，还是配置管理中心，甚至存储系统，你都要面临这样一个选择，push模型 or pull模型?是服务端主动给客户端推送数据，还是客户端去服务器拉数据，一张图表对比如下：

|  |  |  |
| --- | --- | --- |
|  | **push模型** | **pull模型** |
| 描述 | 服务端主动发送数据给客户端 | 客户端主动从服务端拉取数据，通常客户端会定时拉取 |
| 实时性 | 较好，收到数据后可立即发送给客户端 | 一般，取决于pull的间隔时间 |
| 服务端状态 | 需要保存push状态，哪些客户端已经发送成功，哪些发送失败 | 服务端无状态 |
| 客户端状态 | 无需额外保存状态 | 需保存当前拉取的信息的状态，以便在故障或者重启的时候恢复 |
| 状态保存 | 集中式，集中在服务端 | 分布式，分散在各个客户端 |
| 负载均衡 | 服务端统一处理和控制 | 客户端之间做分配，需要协调机制，如使用zookeeper |
| 其他 | 服务端需要做流量控制，无法最大化客户端的处理能力。  其次，在客户端故障情况下，无效的push对服务端有一定负载。 | 客户端的请求可能很多无效或者没有数据可供传输，浪费带宽和服务器处理能力 |
| 缺点方案 | 服务器端的状态存储是个难点，可以将这些状态转移到DB或者key-value存储，来减轻server压力。 | 针对实时性的问题，可以将push加入进来，push小数据的通知信息，让客户端再来主动pull。  针对无效请求的问题，可以设置逐渐延长间隔时间的策略，以及合理设计协议尽量缩小请求数据包来节省带宽。 |

在面对大量甚至海量客户端的时候，使用push模型，保存大量的状态信息是个沉重的负担，加上复制N份数据分发的压力，也会使得实时性这唯一的优点也被放小。使用pull模型，通过将客户端状态保存在客户端，大大减轻了服务器端压力，通过客户端自身做流量控制也更容易，更能发挥客户端的处理能力，但是需要面对如何在这些客户端之间做协调的难题。