系统实现的总体构想

一、系统架构图

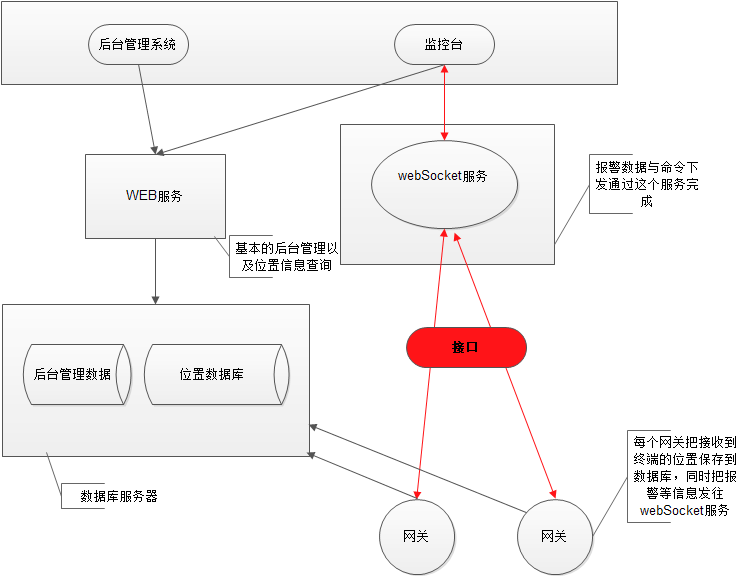


图1: 系统基本结构

二、网关

1. 位置信息记录： 把从终端收到的位置信息保存到历史数据库，同时更新车辆的最新位置信息。

［历史数据，要分多个数据库，目前先按照月份进行分库，每个月有一个数据库，其中的位置信息表记录该月所有车辆的位置信息］

［最新数据，可以采用内存数据库进行实时更新，web服务器可以定期把内存数据库的数据更新到mysql数据库中，暂时也可以考虑直接更新mysql数据库表，将来由性能问题再考虑上述策略］

1. 向webSocket服务器发送报警信息（注意，这里面要有一个报警信息的过滤，多条重复的报警信息要进行过滤，同样报警信息仅发给webSocket一条）
2. 当网关连接到webSocket服务器时，要把与该网关连接的在线终端列表发给webSocket服务器［终端上线接口］，并且每次有新的终端设备连接（或者断开）网关，网关都要发送一条“终端上线消息（或者终端下线消息）”给WebSocket服务器。
3. 接收来自webSocket服务器的下发命令。
4. 每次终端连接上网关之后，网关都要先查询终端的参数信息与属性信息，然后更新数据库。
5. webSocket服务器（数据分发服务）：
6. 记录每个网关负责连接的终端信息。与网关建立长连接。
7. 记录每个监控平台关心的车辆列表。与监控台（浏览器）建立长连接。

1. 网关把终端的报警信息发送给webSocket服务器，webSocket服务器把报警信息推给监控台。
2. 来自浏览器（监控台）的下发指令也通过这个渠道最终下发到终端。
3. web服务器

1. 不与网关打交道，所以web服务器只是处理常规的管理任务。

1. 位置信息的实时显示，为了保证报警信息的实时性，监控台的位置信息直接从位置数据库依靠轮询的方式进行检索。（一般位置信息30秒或者1分钟刷新一次即可）。
2. 其他对实时性没有要求的，直接通过web服务器来满足，信息的交换依靠数据库的信息即可。
3. 数据库的特别说明

考虑到位置信息的数据量，采用位置信息进行按月分库策略。每个月创建一个新的数据库。比如： gpsdb201301 ， gpsdb201302等等。每个数据库中放置位置信息的历史数据。

主体数据库存放整个系统的数据信息表。

最新位置数据库——可以考虑采用内存数据库的策略。暂时先直接更新最新位置数据表，将来有性能要求，再处理。

1. 位置信息

实时位置信息采用轮询的方式从web服务器上进行获取。为了减少数据量的传递，可以考虑几种策略：

1. 根据地图显示区域，每次浏览器从服务端取数据，传递一个地图的区域信息，只有满足下列条件的车辆位置信息才会送到客户端：
2. 在该区域
3. 在线车辆
4. 属于登陆用户监控的车辆
5. 对于叠加到一起的车辆，只发送一个车辆的位置信息（这个是高级操作了，意思就是在地图上几个或者几百个车辆的位置符合一定条件时，在地图上就用一个标注进行显示的情况）
6. 对于与前一个位置点不发生变化的（比如停车情况，直发送一次）——这个需要根据最新位置信息与历史位置进行判断。这种情况如果开销比较大，可以不考虑。
7. 位置的实时性显示包括地图显示与列表更新，所以列表的排序要根据发生变化的车辆进行实时更新。就是有变化的排在前面。
8. 位置信息的具体文字描述信息可以考虑不存储到数据库，数据库字段就是其经纬度等信息。位置信息的具体位置描述，需要高德地图的API根据经纬度查询获得，所以客户端得到位置经纬度，需要调用高德地图的apI进行显示。
9. 报警信息
10. 报警信息通过网关直接发送到webSocket服务器，为了减少相同报警信息发送给webSOcket服务器，网关需要进行过滤。
11. 报警信息通过webSocket服务器，发送给浏览器，浏览器对报警字段进行解析，显示报警信息。
12. 报警都有一个报警开始时间以及一个解警报时间。如果网关只发送报警开始时的信息，那么解除警报的信息如何获取，是不是网关在做报警过滤的逻辑时，也需要把解除警报的那条信息发给webSocket。 ？？？？