# LCRLTS 负载均衡方案

当定位区域比较多，或定位标签比较多的时候，会产生大量的计算需求，对 CPU 的要求会比较高，单台服务器忙不过来，使用多台服务器做负载均衡就是很自然的事了。

对于大规模的定位需求来说，我们可以分为两种：单片区大规模 和 多片区大规模。

**单片区大规模**是指有很多个定位区域，这些定位区域基本上是连成一片的。例如 50 个定位区域，这些区域之间是相连的。一个标签在这些区域之间移动，可能会有多个区域都计算出这个标签的坐标，系统需要进行判断应该输出哪个区域的坐标。

**多片区大规模**是指有很多个定位区域，但是这些区域分成很多片区，片区之间地理上不相连。例如 50 个定位区域分成 2 片区，有 20 个是一片区，另外 30 是另一片区；这两个片区之间相距 100 米以上。当标签在某个片区内移动的时候，只有这个片区内的区域会计算出坐标，另外一个片区内的区域不会知道这个标签的存在，系统在做区域切换的时候只需要考虑当前片区内的区域就可以了，另外一个片区的区域不用考虑。

# 1. 多片区方案

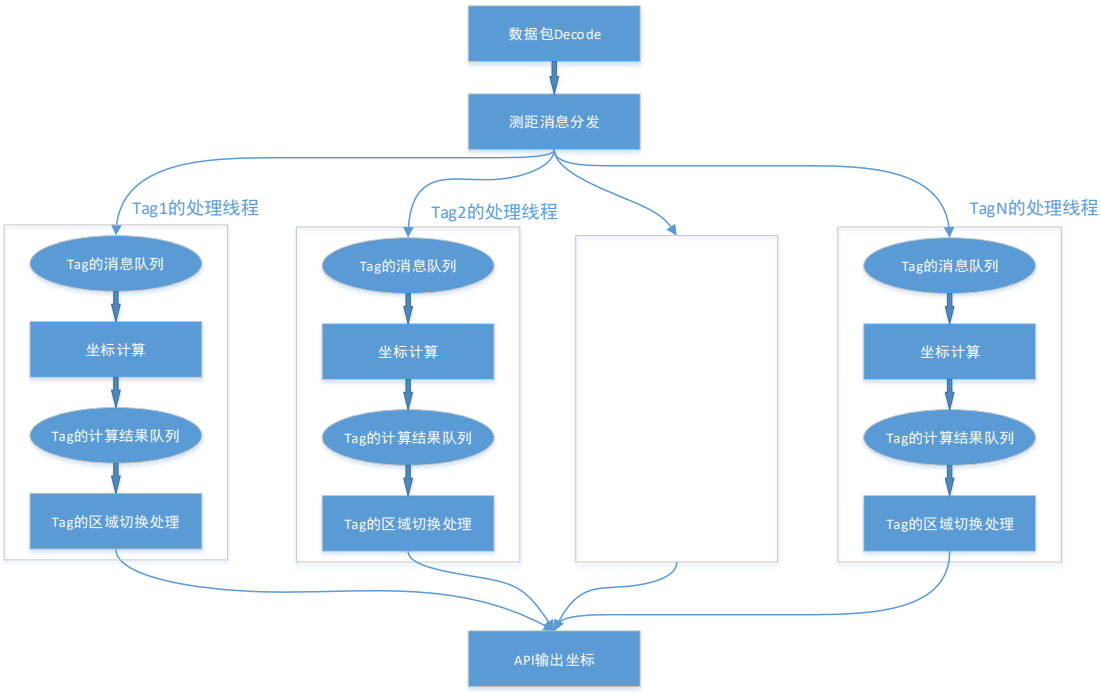
对于多片区的需求，最简单的办法就是建多个单独的定位系统，每个片区一个定位系统。最后把这些系统输出的坐标集成在一起输出到应用程序，或者应用程序分别从这些系统中读取坐标。
因为各个片区之间没有任何逻辑上的关联关系，所以可以使用部署多个定位系统这种简单的方式实现负载均衡。

如果多片区中的某些片区包括有很多的区域，或者有很多标签，我们还需要对这个片区的负载进行分割，具体见后续的单片区方案。

# 2. 单片区方案

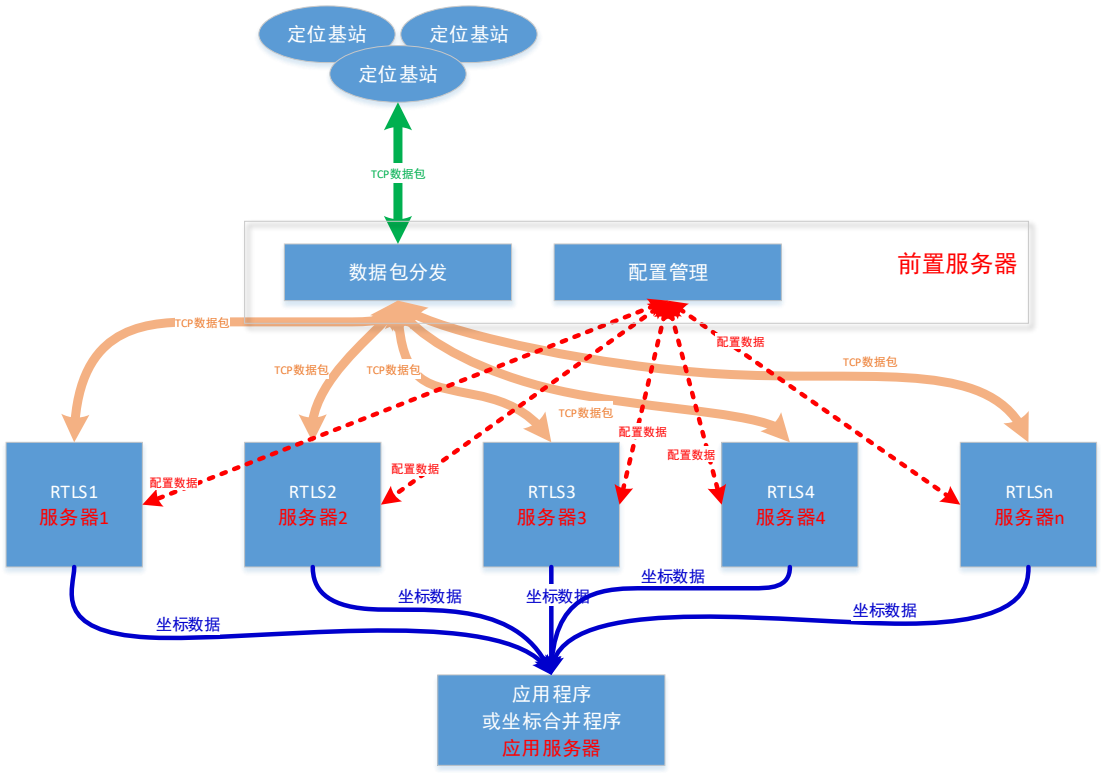
对于单片区的系统，情况比较复杂。
因为各个区域间的关系、标签与区域的关系，这些关系之间我们需要清理出最小原子流程，才可以把这些原子流程分散到多台服务器上去计算。

我们先看一下定位系统的内部处理流程




在定位引擎的所有处理环节中，最消耗 CPU 的环节是“坐标计算”和“Tag 区域切换处理”。**坐标计算**根据时间差数据解方程计算坐标，并且对坐标进行固定纠偏；**Tag 区域切换处理**从各区域计算出的标签坐标中，根据各种权重关系，选择一个坐标输出。

根据上面的处理流程图，可以看出按标签来分割负载是比较合理的方案，把各个“Tag 处理线程”拆分到不同的机器上去处理。



上图是负载均衡结构图。

**前置服务器：**

- 与基站交互，接收基站传来的各类 TCP 消息，并向基站发送心跳消息；
- 按规则对标签分组，不同组的标签的测距消息分发给不同的服务器；
- 把测距消息之外的其他各类消息分发给所有服务器；
- 从某台服务器读取定位引擎配置参数，分发给其他的服务器，保证各台服务器的配置相同。

**服务器 1~N：**

这些服务器运行标准版本的定位引擎。

- 与前置服务器交互，在它看来，前置服务器就像是很多定位基站一样。
- 从前置服务器接收配置参数，或发送配置参数给前置服务器。
- 计算标签坐标
- 对多区域的坐标进行投票，选出最合适的坐标输出

**应用服务器：**

接收各个定位引擎计算出的坐标。

可以在应用程序中直接与各个定位引擎连接，接收各个定位引擎计算出的坐标；

也可以单独写一个接口程序，与各个定位引擎连接，接收各个定位引擎计算出的坐标后，统一输出给应用程序。应用程序使用从这个接口程序读取坐标数据。