**后端编码规范总结**

**前言：**

本文用于总结基于现在项目后端框架构建的服务器在开发时候的注意事项或可能存在的问题，如有任何疑问与观点见解，欢迎交流。

1. **进程**

Erlang vm与其他语言的虚拟环境最大的不同就是erlang vm的运行是以一个个轻量级进程为单位（经测试，一个新建的进程约占2kb内存），我们项目也是基于这个基石搭建的，服务器在运行时会新建各种各样的进程如玩家进程，任务进程，物品进程，场景进程等等，无论使用神马进程去实现自己逻辑都好，必须遵守以下三个原则：

1.分工均匀：如果进程计算的压力过大，便会成为服务器运作的瓶颈，这时候你需要考虑如何为这个进程减压，我们现在的做法就是创建一个个的代理进程，将一份任务分配给一个个同样地位的进程去处理，从而提交工作效率，但代理进程在设计使用也需要有所注意，不然后患无穷（将在第3点说明）

2.数据一致：多进程的好处是可以并发工作，从某方面说，这也是个坏处，当多个进程操作处理同一份数据时，就会出现脏数据，这是非常危险的，因为脏数据不是异常也不是错误，erlang vm不会对这种问题做任何异常处理，那就意味着当发生数据错误，问题就很难定位。所以，在平时的开发中，我们应该注意将一些重要的数据（如玩家信息），托管给一个进程单独处理，这样就可以避免脏数据问题。Ps:面对多进程还有一种方法就是加锁，erlang 的api是有提供锁操作的，但不建议使用，因为加锁后，并发的逻辑就变成单进程流水，影响效率，再有，锁操作本身就不符合erlang设计的理念

3.代理进程：代理进程虽然能给逻辑处理带来效率的提升，但当逻辑设计先后顺序时，代理进程就带来不少隐患。因为代理进程的进程优先级是一样的，在多核的erlang vm中，就会出现后面的代理进程先进任务队列而前面的后进，这样就会出现逻辑错误，所以要求有先后顺序的逻辑千万不要随便使用代理进程的形式

1. **Ets**

Ets又称内存数据库，作用类似c++与java常用的memcache，ets为erlang系统的缓存带来便捷的同时,也存在风险，其中最大的风险就是并发问题，ets有2把锁， 一把保护meta table, 一把保护数据表，锁是系统的读写锁，这个开销是不容忽视的。所以，当多个进程同时操作一个ets时，就会出现严重的锁竞争，对后端的性能有很大影响，为避免该情况的发生，一般有两种做法：

1.ets分表：将一个ets按照逻辑需要分成多个，几个逻辑相同的进程操作同一个ets

2.使用进程字典

3.打开ets的并发开关：该方法能一定程度上提高ets表的读写效率，做法如下图



1. **原子**

在erlang 中，atom可以理解成常量，它可以包含任何字符，以小写字母开头，如果不是以小写字母开头或者是字母之外的符号，需要用单引号包括起来，比如abc,’AB’。atom的匹配速度非常快，但是由于erlang VM将所有模块的atom保存在一个全局的atom表中，它的最大限制是1048576，也就是说erlang的VM只能处理1048576个不同的atom，当atom的个数大于这个限制时，VM就会down掉。所以当你的模块中使用了list\_to\_atom函数生成动态的atom时就要非常的注意是否会产生大量的不同atom。

1. **编码规范**
2. 在逻辑循环分块时尽量使用尾递归，erlang对尾递归是有做优化的
3. 注意erlang的++操作，因为erlang的++操作实际就是将左边的列表复制到右边，所以保持左边的列表尽量小

3.尽量控制函数逻辑的单一性，简单的说，就是让每个函数只处理一件事，这个法则无论在erlang还是其他语言都适用

4.让你的函数名称为你注释，最好的注释就是没有注释，当同事看到你的函数名时就明白你的函数是干嘛用的 ，当然，当有时候函数的内涵太丰富时，请你加上你的注释，以让调用者明白，这个是干嘛用的