简单写篇 ZNet 的服务器架构原则

如果从 ZServer4D 转到 ZNet，基本上 100%会被 C4 体系搞头晕。因为 C4 很偏架构，同时，那些代码非常高级，理解 C4 不可以像 ZS 那样通过阅读代码完成。

首先，我们要明确一下 C4 的设计思路

C4 底层设计

- C4 走的是 p2pVM+双通道程序模型，任何 C4 模块，永远是双通道
- C4 没有阻塞支持机制，所有的数据收发，连接，均是异步触发事件的程序模型

C4 的中层设计

讲解中层设计之前，要讲一下 ZNet 的三种应用通讯模型库

- Z.Net.DoubleTunnelIO.pas，该框架是 ZS 早期带验证机制数据库的双通道框架
- Z.Net.DoubleTunnelIO.NoAuth.pas，不带验证双通道框架
- Z.Net.DoubleTunnelIO.VirtualAuth.pas，接口化验证双通道框架

这三种通讯模型，内置支持了文件，消息队列，数据库队列，并且千锤百炼
ZS+ZNet+C4 所有的双通道服务，都在这三种通讯模型上继承而来
C4 所有的高级服务器，都是基于以上三种通讯模型继承使用

C4 的高层设计

C4 解决了服务器技术的复用问题，C4 从设计角度将服务器常用一体化设计，直接拆解成了部件，例如我们平时经常用到的用户身份验证数据库，在 C4 里面是一个独立部件，使用时将它组合起来即可。

使用 C4 框架开发服务器，底层的开发基本上是一次性的，同时，底层的部件非常生僻，都是大家平时很难见到的框架设计。当开发大型服务器系统时，这些服务器部件，都可以独立实例存在，就像 SaaS 中的节点，将他们组织起来，加以调度，即可应用，这将是非常省事的做法。试想一下，即时通讯系统服用，文件系统复用。

C4 的精华都在于服务器部件，尤其是对高并发，高速，大流量网络的支持，让这些部件的设计显得非常生僻：一大堆 Cache 机制，各种数据结构，用内置脚本的调试方式。

值得一提的地方：所有的 C4 服务器部件，支持库以 Z.Net.C4_*.pas 开头，表示这是正规 C4 部件，需要使用 C4 的组网机制来驱动服务。如果支持库以 Z.Net.C4.VM_***.pas 开头，表示这个 C4 部件支持独立服务器工作，既创建一个实例服务器也就完成了，没有 C4 组网机制，这些 VM 开头的独立服务器均能 100%支持高并发和大流量。**

by.qq600585
2020-10-15