**Hades通用计算服务器组件**

****

目录

[1. 核心功能 3](#_Toc486253539)

[1.1 脚本系统 3](#_Toc486253540)

[1.2 Remote Procedure Call 3](#_Toc486253541)

[1.3分布式服务器与单节点服务器 4](#_Toc486253542)

[1.4网络通讯 4](#_Toc486253543)

[1.5功能扩展 4](#_Toc486253544)

[1.6调试与跨平台 5](#_Toc486253545)

[2. 底层架构 5](#_Toc486253546)

[2.1服务体系 5](#_Toc486253547)

[2.2事件驱动 6](#_Toc486253548)

[2.3线程模型 6](#_Toc486253549)

[2.4池化与内存管理 8](#_Toc486253550)

[2.4插件体系 8](#_Toc486253551)

[3现有功能模块 9](#_Toc486253552)

[3.1 HTTP服务器 9](#_Toc486253553)

[3.2 高效数学库 10](#_Toc486253554)

[3.3 导航网格 10](#_Toc486253555)

[3.4 机器状态 10](#_Toc486253556)

[3.5 Torch 人工智能 10](#_Toc486253557)

[3.6 更多的网络资源 10](#_Toc486253558)

[4典型分布式架构 11](#_Toc486253559)

[4.1科学计算集群 11](#_Toc486253560)

[4.2 GS系统集群 12](#_Toc486253561)

[4.3开房间游戏服务器集群 13](#_Toc486253562)

# 核心功能

## 1.1 脚本系统

Hades通用计算服务器组件将底层多线程环境与上层业务逻辑隔离，使用Lua为依托的脚本系统为核心构架了一系列化繁为简的服务体系、RPC分布式、面向过程开发、热更新等核心功能。而且依托Lua为核心的支持RPC的服务体系确保了线程的安全性，极大降低了开发难度。而热更新功能带给开发人员更便捷的手段在线修复服务器Bug、调整数值。

## 1.2 Remote Procedure Call

Hades通用计算服务器组件支持跨不同节点的远程调用，且由于Lua脚本系统极高的灵活性，使得不同节点之间的远程调用代码及其简单，如下列代码为例：

for \_, node in pairs(self.clusternodes) do

node:SetLearningRate(currentlearningRate);

end

这段简单的for循环节点调用实际上是一个远程调用。Lua的灵活性使得该调用在底层层面转化为一个函数id与参数数据为核心的一条调用协议，并发送至远程服务器。而对上层使用者来说这句代码就是一行简单Lua函数调用，这种特性极大的降低了服务器集群系统的开发难度。

RPC体系的简便使得Hades框架犹如一块乐高积木，让每一位使用者用任意数量的积木来搭建属于自己的业务系统。而多节点分布式服务器开发如同在同一进程中开发程序一样简单。

## 1.3分布式服务器与单节点服务器

Hades通用计算服务器组件是架构无关的服务器组件，所以使用hades即可假设一套分布式体系服务器，也可以架设但节点服务器体系。Hades同时提供两套服务体系的内核插件，Remoteservicessystem和Localservicessytem供大家使用（当然使用Remoteservicessystem也可以做单节点服务器，但基于效率考虑我们推荐使用Localservicessystem）

## 1.4网络通讯

Hades通用计算服务器组件内核系统提供一套以多线程为基础的基于事件驱动的高效率TCP网络IO监听处理框架。同时以openssl为基础提供传输层安全的通讯协议，增强服务器数据的保密性以及安全性。

## 1.5功能扩展

Hades通用计算服务器组件内核插件为主体，功能插件位辅助的的系统框架简便的提供了C/C++层面的功能扩展，进阶开发者可以通过以插件形式使用C/C++层面内核功能为脚本层提供大量的功能扩展。例如Hades提供的Recast Navmesh就是一这种形式提供了寻路功能。

## 1.6调试与跨平台

Hades通用计算服务器组件同时支持大部分\*nix与non-\*nix系列服务器，且在windows 7、windows server2013、Ubuntu server 14.04 LTS、Ubuntu16.04 LTS系统上已有部署于检验。支持已ZeroBrane studio为主的调试环境。考虑到目前市面上存在较多的windows环境开发人员，跨平台也提供了更友善的开发环境，进一步降低开发成本。

# 底层架构

## 2.1服务体系

Hades服务器的业务处理的核心是围绕已服务体系为概念而设计的。在系统中每一个服务是独立的且被隔离的，服务之间只能以接口的形式相互调用传递信息（调用会被序列化为调用指令在多服务间传递），这样在服务器这种高并发环境中保证了服务数据的唯一性以及安全性，且服务本身是线程安全的，每一个单独的服务确保同一时刻只允许在一个线程上进行业务逻辑处理。使用这样的服务体系架构，使得应用层脚本开发人员在调用其它服务的时候，不知不觉的使得代码在多线程的环境下执行，这样提高了系统处理能力，合理的利用了线程，这一切又是都是对上层开发者透明的，开发高并发程序如同开发单线程程序一样简单。

## 2.2事件驱动

Hades使用基于libevent为基础构建以事件为驱动的网络数据响应模式。而libevent本身使用基于 [/dev/poll](http://download.oracle.com/docs/cd/E19253-01/816-5177/6mbbc4g9n/index.html), [kqueue](http://www.freebsd.org/cgi/man.cgi?query=kqueue&apropos=0&sektion=0&format=html), [event ports](http://developers.sun.com/solaris/articles/event_completion.html), [POSIX select](http://manpages.debian.net/cgi-bin/man.cgi?query=select), [Windows select()](http://msdn.microsoft.com/en-us/library/ms740141(v=vs.85).aspx), [poll](http://manpages.debian.net/cgi-bin/man.cgi?query=poll), [epoll](http://www.xmailserver.org/linux-patches/epoll.txt)为基础的通讯模型，提供完整的高可复用的网络通讯层。基于事件驱动这一特点，服务器本身也构架于该模式之下。对于上层应用可以选择性的使用传统loop模式（定时器系统提供定时服务）或者更为现代化的事件驱动为模型开发整体上层应用。如果使用事件为基础的网络消息响应模式，在理想状况下可以经可能的降低每一条数据的处理周期，降低游戏与服务器通讯的请求延迟。

## 2.3线程模型

Hades服务器内核系统目前拥有2套核心线程模型，第一套为mulitlibeventserversystem网络服务模块使用的网络通讯线程模型，第二套为服务体系核心线程模型。

其中网络线程模型使用一条监听线程和多条IO处理线程来将不同的连接请求平均的发送至不同的线程来进行业务上的数据读写。其流程模型如下图所示：

系统启动

端口监听

事件响应

退出事件

连接事件

分发请求

系统启动

请求响应

读写操作

连接操作

请求监听

退出

服务体系

…

而另外一套服务体系线程模型，为本通用计算服务器组件的核心线程模型，一切上层应用都建立在此模型之上。服务体系对线程利用的设计思想类似于目前多进程操作系统对多个进程管理的设计思想。其中每一个独立的服务，对于操作系统中的一个进程，这样类似的“进程隔离”手段保证了每一个服务的数据独立性与线程的安全性。而在线程的利用上，只有当服务接收到调用请求之后才会分配线程进行处理，且遇到任意异步调用（例如远程调用、异步服务）之时将该服务携程暂停，让出线程。这样设计，提高了线程的负载能力，从而提高服务器对业务的处理能力。线程模型图如下所示：

任务分发

调用请求

服务管理

服务A

服务A

服务C

服务B

服务D

服务C

空闲

空闲

## 2.4池化与内存管理

Hades在设计过程中通过两种手段保证内存的使用效率，降低内存碎片产生的数目。第一种方法是用一个全局的内存池来管理全部的内存分配。第二种方法使用一个线程安全的对象池来缓存和维护系统在运行过程中可能产生的动态对象。不夸张的说对象池甚至将只能之中的引用计数也通过池的形式来缓存起来。而上层的Lua脚本层中的全部对象是lua通过内存吃的方式申请与管理的，这样一来系统在达到饱和状态后基本不会再有新的内存的申请与释放，这样最大限度的减轻了内存碎片产生的可能性。

## 2.4插件体系

Hades通用计算服务器组件底层使用已内核插件为主体，功能插件位辅助的的系统框架，使用者可以根据自己的不同需求使用不同的插件系统来拼装节点功能。

例如目前系统存在两份不同的网络通讯层内核插件，分别是mulitlibeventserversystem与libeventserversystem这两份插件同时提供了TCP网络功能，不同的是第一个插件使用多线程框架适合运用在网络并发较高的环境，而另一个插件使用单线程网络框架，适合运用在网络并发较低的环境下，这两套插件在各自擅长环境都有优于对方的性能效率。

插件体系将全部内核系统以虚函数的形式抽象出来提供了每一个内核系统的独立性，同时也更简便的为进阶开发者提供了功能定制与功能扩展的便利性。例如开发者如果有特殊需要，可以自己开发一套以C#为基础的脚本系统来实现脚本系统内核接口，这样就可以在不需要修改任何其他系统的代码的情况下，平滑的替换为C#脚本体系。

目前Hades内核系统分别是：文件系统、配置表系统、日志系统、内存管理系统、网络服务系统、网络连接系统、脚本系统、序列化系统、服务系统、线程系统、定时器系统。

# 3现有功能模块

## 3.1 HTTP服务器

HTTP/HTTPS服务器功能模块提供一些基础的针对HTTP协议请求响应的处理功能，该服务模块会将请求转发至指定服务进行并非处理。

## 3.2 高效数学库

提供一套以C为基础的游戏常有的数学库相关计算，例如矩阵计算、向量计算、四元数计算、椎体计算、平面计算、射线计算等常用的游戏相关的数学工具。

## 3.3 导航网格

导航网格是以RecastNavgation为基础开发的地图寻路工具，同时提供UE4导出插件以及Unity导出插件。

## 3.4 机器状态

机器状态提供采集计算机内存、IO、CPU、等硬件使用情况的功能。

## 3.5 Torch 人工智能

系统还提供基于Torch的人工智能分布式计算框架，可以在该框架下快速搭建，人工智能相关模型与应用。

## 3.6 更多的网络资源

Hades使用Lua作为脚本层，而Lua社区有大量的开源方案进行各种不同业务处理，例如XML解析，Json解析等等，这些功能大部分都可以直接运用在Hades服务器中，大大提高了开发效率，降低了项目组重复造轮子的烦恼。

# 4典型分布式架构

## 4.1科学计算集群

随着人工智能的兴起与流行，人工智能运用在多方面越加成熟。但是随着人工智能的发展，需要人工智能进行计算的数据也越来越复杂越来越庞大，训练时间越来越漫长，因此分布式计算成为大型人工智能系统必备的核心系统。而使用Hades为基础的分布式训练系统很好的运用了Hades对分布式开发的友好性，很容易开发出如下图所示的分布式人工智能系统。

标本调度

发送训练状态

中心服务器

同步参数、数据调度

分布式文件系统

TensorBoard数据监控

监控训练状况

计算节点

计算节点

计算节点

…

计算节点

读取标本

使用这套分布式架构，计算节点可以按照计算复杂量来伸缩。而中心节点使用参数同步的方法，尽量降低同步对性能的损耗。

## 4.2 GS系统集群

GS系统是引擎部与客户服务部门开发的一套针对潜在VIP客户提供一对一专属客服支持，用户引导的客户服务体系系统。该系统由三部分组成，分别是客户服务后台系统，IM聊天系统，客户端系统。该系统的系统架构图如下：

链接至系统接受查询

聊天服务器

客服系统后台

客服请求处理服务器

…

客户端查询

客户端查询

Nginx

链接服务器

链接服务器

客户端

客户端

客户端

http请求查询是否为vip

在这个结构先，客户端启动后首先通过https请求向服务器查询自己是否为需要服务的对象，如果是，那么客户端将主动链接到链接服务器，来提对接服后台的数据查询和聊天功能。这种结构可以通过单独的伸缩查询服务器和链接服务器来适应不同的项目需求，日均活跃用户更大的游戏选增更多的客户端查询服务器，而需要进行链接维护用户大的选择增加更多的链接服务器。

## 4.3开房间游戏服务器集群

Hades通用计算服务器组件是一套拥有上下文环境的分布式服务器框架，这种特性使得Hades同样适用于游戏服务器的开发，下图为一款以开房间为主的分布式游戏服务器：

数据库集群

Gate服务器

…

房间服务器

房间服务器

客户端

客户端

客户端

在这样的结构之下Gate服务器提供更高的网络IO性能，而业务逻辑的计算迁移至房间服务器，有效地拆分了对服务器不同负载类型的平衡。