基于ICE服务的开发模式详解

张明义

2017/05/02 创建

# 目的

Ice提供了基本的网络通信框架、对象、服务概念，有力的支持我们完成：

1. 网络通信的位置透明(只需要固定Registry地址，其他所有地址均透明)
2. 跨平台、跨语言通信(Slice中间语言)
3. Slice定义推动我们深入思考系统中的概念，强化概念抽象过程。
4. 业务多点分布式部署，负载均衡。

但是，Ice并没有提供给我们分布式下的数据一致性框架，也就是说一旦业务在多点部署，服务内部的缓存将导致严重的数据一致性问题，最终，分布式必须变为无状态服务。这样，服务化就完全没有了任何意义，和网站没有任何区别，最终压力又将转向数据库。

为了利用服务内自建的缓存，目前的服务框架基本Paxos在Ice基础上开发了一致性框架，完成了资源到业务对象的严格映射，目前的机制保障了：

1. 围绕着一种资源的所有操作一定发生在一个服务进程内部。
2. 如果服务宕机，分为两种情况：
   1. 短时间（5秒）内服务重启，所有之前定位在本服务的资源依然会在本服务执行。
   2. 超过一定时间（5秒）后，如果服务仍未启动，在资源业务请求时会重新定向到新的节点，并且保障：原节点重启后，已经转移的资源不会重新定位到原节点。

这样，开发人员可以放心的使用本地缓存，并可以确保，这里缓存的数据，相同业务的其他节点一定不会修改，基于此，业务流程中对数据的读写可以进行大量优化，充分利用带有一致性缓存的服务，将显著降低数据库压力，这也是系统设计中的关键点。

# 基本概念

## Ice概念

* Slice

中间语言，Ice提供了工具将其生成Cpp、C#等多种语言接口、声明文件。

Slice业务接口定义的过程就是开发人员在做系统设计的过程。

* IceGrid

Ice在基本的通信框架基础上提供的网格计算基础设施，其实现了业务对象位置透明的特性，其主要包括两个组件：IceGridRegistry和IceGridNode，这两个组件以windows服务形式存在，并不需要我们做任何开发相关的工作，仅仅部署上即可。

* + IceGridRegistry

起到了名字服务的作用，所有我们的业务对象的访问地址都会自动注册到此服务内，为所有对象访问者提供位置查询接口。

* + IceGridNode

通常，每台机器上都部署一个IceGridNode，代表着网格中的一个节点，其会与IceBox进行交互，完成IceBox的启动、停止等等所有本机业务服务的部署管理工作。

* IceBox

Ice为我们实现的一个业务service的运行容器，以进程的形式体现，在IceGrid架构下，其启动、停止由IceGridNode完成，脱离IceGrid，我们也可以调试启动它，给他相应的配置文件就可以使其加载我们自己的业务service DLL。

一个IceBox会包含一到多个业务service。

* Service（业务服务）

业务object的容器，每个服务内部可以包含一到多个对象，一般而言，业务关联紧密的对象会放到一个Service内，service由DLL形式体现，会被IceBox加载。

* Object（业务对象）

Slice接口的继承、实现者，这里实现了业务逻辑。

* 部署描述XML文件

XML文件，描述了整个网格内都有哪些icegridnode、每个node下都有哪些icebox以及其下所有service、object。

所有业务逻辑要用到的自定义配置也在其内部进行定义，服务及对象有能力从中读取到自己的配置。

## Lx概念

* ObjectInvoker(业务对象访问器)

为了做到分布式一致性，现有框架下抛弃了Ice自身的位置透明机制，采用基于Paxos选举的位置定位机制，因此，在Ice基础上封装了对象访问器概念，内部完成选举、缓存、过期等等机制，但对业务实现透明。

## 概念关系



一般情况下，每台机器部署一个IceGridNode，每个IceGridNode管理本机内部的多个IceBox，每个IceBox内部根据系统设计结果决定一到多个Service(Dll)及每个Service内的对象集合。

# 安装部署包目录说明

“LXGrid”作为部署目录的统一名称，在其下有如下子目录：

|  |  |  |
| --- | --- | --- |
| **目录** | **说明** | **开发环境需要** |
| Config | 配置文件都在这里，开发机需要修改其中node.cfg的内容，需要修改两项：一是node自身的ip地址，二是registry地址. | 是 |
| CrashDump | 崩溃dump文件存放目录，c++专用。 | 是 |
| Data | Icegridnode产生的数据都会放在这里，这个目录是在上述node.cfg中指定的，一般不需要修改。 | 是 |
| IcePatchServer | Patch服务专用目录。 | 否 |
| IceRunTime | Ice运行时库 | 是 |
| Log | 日志输出目录 | 是 |
| Plugin | Registry插件目录，插件完成代理有效性判定。 | 否 |
| Store | Paxos、持久化缓存文件目录。 | 否 |
| Tool | 一些常用工具，如log查看器、depends、远程调试器等。 | 否 |

# 调试环境安装与部署

开发过程中，开发人员可以进行两种部署模式:

* 一是在开发机部署完整的网格，这样，整个网格的管理由开发人员自己完成，包括icegridregistry、icegridnode的部署、xml描述文件的维护等。

优点是能够了解整个网格的部署过程，但过程过于繁琐，更新、维护成本过高。

* 二是，本机仅作为测试网格的一个icegridnode，直接使用测试环境的icegridregistry作为名字服务。

这里先介绍方法二，准备材料：

1. 安装部署包。
2. 环境变量修改批处理文件。

**步骤0**：向网格管理员索要node序号及网格registry地址、网格管理员修改XML，增加服务描述。

**步骤1**: 设置环境变量、修改本地icegridnode配置、注册icegridnode服务。

* 修改部署包中的env.bat，环境变量包括:

Path中加入IceRunTime目录。

NEW\_FOUNDATION\_ROOT 对应本地的TFS://FoundationNew目录。

NEW\_ICE\_ROOT 对应本地的 TFS://FoundationNew/FoundationICE目录。

修改完成后，执行env.bat即可。

* Icegridnode的配置文件在上述部署目录的config目录下，一般命名为：node{name}.cfg

// 修改此ip为本地ip（非回路ip）

IceGrid.Node.Endpoints=tcp -h 192.168.2.65

// 修改此LXGrid{GridName}.Node{序号}, 网格是服务所在的网格，可以是Main或App，序号是网格管理员分配的。

IceGrid.Node.Name=LXGridMain.Node2

// 改为本地部署目录的Log目录的绝对路径

Ice.LogFile=D:/Develop/LXGrid/Log/LXGridMain.Node2.log

// 改为本地部署目录的Data目录的绝对路径

IceGrid.Node.Data=D:/Develop/LXGrid/Data/Node2

// 修改为服务所在的网格及网格的registry地址，registry地址向网格管理员。

Ice.Default.Locator=LXGridMain/Locator:tcp -h 192.168.2.21 -p 4061

* 注册icegridnode服务

cmd-> 输入 iceserviceinstall icegridnode node{name}.cfg -> 回车

**步骤2**: 将发布目录下的config文件拷贝到工程debug目录下，并修改其中dll路径变为当前路径。

config文件目录：在部署目录的Data\Node0\servers\{服务名} \config目录下

将其拷贝到当前服务工程debug目录下，并修改其内容，示例如下：

原内容：

…

IceBox.Service.LXServiceClusterNode=D:/LXGrid/Data/Node0/servers/LXZone0/distrib/LXSystem/LXServiceClusterNode:create --Ice.Config='D:/LXGrid/Data/Node0/servers/LXZone0/config/config\_LXServiceClusterNode'

IceBox.Service.LXServiceZone=D:/LXGrid/Data/Node0/servers/LXZone0/distrib/LXZone/LXServiceZone:create --Ice.Config='D:/LXGrid/Data/Node0/servers/LXZone0/config/config\_LXServiceZone'

…

将删除红色内容删除即可。

**步骤3**: visual studio F5调试启动并进行调用测试(假设基本服务代码已经开发完成，开发步骤参考下面《开发过程》章节)。

因为引入了网格的概念，整个开发环境的建立还是相对比较复杂，这里描述下网格管理员与服务开发人员的配合流程：



# 开发过程

未完，待续。

* 基于服务的开发的特点是：

业务代码会运行在一个**分布式**环境下。

有本地缓存可以利用。

因此，整个Ice服务开发过程会着重强调系统设计，这包括：

系统概念、分布式一致性考虑设计直接影响着业务object的划分角度与粒度。

Slice编写

Service类、object类实现

组件使用场景描述：

DynamicLocker：使得分布式乐观锁退化为进程内锁，同时，避免数据集锁，实现了单一资源锁。

ObjectCacher：缓存类，可设置总的过期时间

PerformanceCounter:精确到微秒的计时器，是个做性能分析不错的工具。

自定义业务配置