**IceGrid应用**

**配置手册**

**V 2.1**

中诚信资讯科技有限公司

目录

[1. 概述 3](#_Toc294013311)

[1.1 配置目标 3](#_Toc294013312)

[1.2 实验环境 3](#_Toc294013313)

[1.3 局限 3](#_Toc294013314)

[2. 配置过程 3](#_Toc294013315)

[2.1 服务器端配置 3](#_Toc294013316)

[2.1.1 主注册服务配置 5](#_Toc294013317)

[2.1.2 从注册服务配置 7](#_Toc294013318)

[2.1.3 应用部署配置 10](#_Toc294013319)

[2.1.4 节点配置 13](#_Toc294013320)

[2.2 客户端配置 14](#_Toc294013321)

[3. 结果验证 14](#_Toc294013322)

[3.1 程序方式 14](#_Toc294013323)

[3.2 工具方式 14](#_Toc294013324)

[4. 高级应用配置 20](#_Toc294013325)

[4.1 集成IceBox 20](#_Toc294013326)

[4.1.1 IceBox服务程序编写 20](#_Toc294013327)

[4.1.2 IceGrid集成IceBox服务 21](#_Toc294013328)

[4.1.3 测试验证 25](#_Toc294013329)

[4.2 集成IcePatch2 27](#_Toc294013330)

# 概述

## 1.1 配置目标

本文档是描述Ice中间件中的IceGrid服务的应用配置，通过使用IceGrid服务来实现：

1. 服务器端服务分布式部署。
2. 服务器端服务按需激活。
3. 服务器端服务多节点负载均衡。
4. 注册服务主/从热备（Master/Slaves）
5. 集成IceBox服务

## 1.2 实验环境

1. 硬件：hp服务器，3台
2. 操作环境：Red Hat 5
3. 服务器程序：ServerApp.jar
4. 说明：实际应用中，服务器节点可任意扩充、操作系统可被更换、服务器程序可用实际项目的服务程序替换，本文档所描述的配置方式具有通用性，适用但不局限于当前实验环境。

## 1.3 局限

本文档不详细描述IceGrid服务的运行机制和实现原理，不详细介绍服务器端和客户端程序的实现，主要描述IceGrid服务应用的配置步骤、主要配置项及验证配置结果等。

# 配置过程

## 2.1 服务器端配置

**配置步骤：**

1. 创建主注册服务（Master）的配置文件config\_master.grid，文件名称可以任意
2. 创建从注册服务（Slave）的配置文件 config\_slave.grid， 文件名称可以任意
3. 创建各节点服务的配置文件config.node，文件名称可以任意
4. 创建分布式应用配置文件app.xml，文件名称可以任意，但格式最好定义成xml
5. 运行Ice提供的工具，启动我们的分布式应用，主要有如下两个工具：icegridnode和icegridadmin。详细启动过程如下：
6. icegridnode --Ice.Config=config\_master.grid 启动主注册服务
7. icegridnode --Ice.Config=config\_slave.grid 启动从注册服务
8. icegridadmin --Ice.Config= config\_master.grid -e "application add app.xml" 部署分布式服务

icegridadmin --Ice.Config= config\_master.grid -e "application update app.xml" 重新部署分布式服务

1. icegridnode --Ice.Config=config.node 将各节点注册到注册服务的注册表中

**配置文件清单：**

假设有n个节点（n > 0）, 其中从注册服务有x个，(x > 0)

config\_master.grid ---------- 主注册服务配置文件 --------- 1份

config\_slave.grid ----------- 从注册服务配置文件 --------- x份

config.node ----------------- 节点配置文件 -------------- n份

app.xml ---------------------- 部署配置文件 -------------- 1份

通常情况下，由于注册服务占用资源很少，所以一般都会和一个节点集成在一起，并且可以和节点服务在一个进程中运行。因此，如果假设服务部署到n个服务器，通常情况下配置文件清单如下：

config\_master.grid -- 主注册服务配置文件 -- 1份 -- 主注册服务信息+节点信息

config\_slave.grid --- 从注册服务配置文件 -- x份 -- 从注册服务信息+节点信息

config.node --------- 节点配置文件 ---- n-1-x份 -- 节点信息

app.xml ------------- 部署配置文件 -------- 1份 -- 部署信息

其中app.xml要和config\_master.grid放在一台服务器上，下面的各章节将详细介绍各配置文件。

### 2.1.1 主注册服务配置

**config\_master.grid的内容：**

#

# The IceGrid Instance Name

#

IceGrid.InstanceName=IceGridRDDataSource # 1

#

# The IceGrid locator proxy.

#

Ice.Default.Locator=IceGridRDDataSource/Locator:default -h 10.0.5.201 -p 12000:default -h 10.0.5.202 -p 12000 #2

#

# IceGrid registry configuration.

#

IceGrid.Registry.Client.Endpoints=default -p 12000 #3

IceGrid.Registry.Server.Endpoints=default #4

IceGrid.Registry.Internal.Endpoints=default #5

IceGrid.Registry.Data=master #6

IceGrid.Registry.PermissionsVerifier=IceGridRDDataSource/NullPermissionsVerifier #7

IceGrid.Registry.AdminPermissionsVerifier=IceGridRDDataSource/NullPermissionsVerifier#8

IceGrid.Registry.SSLPermissionsVerifier=IceGridRDDataSource/NullSSLPermissionsVerifier#9

IceGrid.Registry.AdminSSLPermissionsVerifier=IceGridRDDataSource/NullSSLPermissionsVerifier #10

#

# IceGrid SQL configuration if using SQL database.

#

#Ice.Plugin.DB=IceGridSqlDB:createSqlDB #11

#IceGrid.SQL.DatabaseType=QSQLITE #12

#IceGrid.SQL.DatabaseName=register/Registry.db #13

#

#

#Ice Error and Standard output Set

#

#Ice.StdErr=master/stderr.txt #14

#Ice.StdOut= master /stdout.txt #15

#

#Trace Registry properties

#

Ice.ProgramName=Master #16

IceGrid.Registry.Trace.Node=3 #17

IceGrid.Registry.Trace.Replica=3 #18

#

# IceGrid node configuration.

#

IceGrid.Node.Name=node\_1 #19

IceGrid.Node.Endpoints=default #20

IceGrid.Node.Data=node\_1 #21

IceGrid.Node.CollocateRegistry=1 #22

#IceGrid.Node.Output=node\_1 #23

#IceGrid.Node.RedirectErrToOut=1 #24

# Trace properties.

#

IceGrid.Node.Trace.Activator=1 #25

#IceGrid.Node.Trace.Adapter=2 #26

#IceGrid.Node.Trace.Server=3 #27

#

# Dummy username and password for icegridadmin.

#

IceGridAdmin.Username=mygrid #28

IceGridAdmin.Password=mygrid #29

**配置项说明：**

# 1 为这个应用实例指定一个唯一的标识

# 2 注册服务的端点信息(主注册服务和所有的从注册服务)，节点注册时要用到

# 3 客户端访问注册服务器的端点信息

# 4 服务访问注册服务器的端点信息，通常是default

# 5 内部访问端点信息，通常是default，节点用这个端口和注册服务通信

# 6 注册服务的数据目录的路径

# 7 设定防火墙安全代理，从而控制客户端访问注册表时可用的权限

# 8 设定防火墙安全代理，从而控制注册表管理者可用的权限

# 9 设定SSL安全代理，从而设定客户端访问注册表时的SSL安全访问机制

# 10 设定SSL安全代理，从而设定注册表管理者的SSL安全访问机制

# 11 指定Ice对象序列化的机制，如果不设置，默认用Freeze机制

# 12 指定使用数据库的类型

# 13 指定使用数据库的名称

# 14 指定标准错误输出文件

# 15 指定标准输出文件

# 16 指定主注册服务的名称

# 17 指定主注册服务跟踪节点信息的级别（0~3），默认为0

# 18 指定主/从热备注册服务的跟踪级别（0~3），默认为0

# 19 定义节点的名称，必须唯一

# 20 节点被访问的端口信息，注册服务使用这个端点和节点通信，通常设为default

# 21 节点的数据目录的路径

# 22 定义节点是否和注册服务并置在一起，设为1时并置，设为0时不并置

# 23 节点标准输出信息重定向蹈的目录路径，会自动生成输出文件

# 24 节点上的服务程序的标准错误重定向到标准输出

# 25 激活器跟踪级别，通常有0，1，2，3级，默认是0

# 26 对象适配器跟踪级别，通常有0，1，2，3级，默认是0

# 27 服务跟踪级别，通常有0，1，2，3级，默认是0

# 28 IceGrid管理器登录该应用的用户名

# 29 IceGrid管理器登录该应用的密码

未涉及的属性还有一些，如果需要请参考官方文档。

### 2.1.2 从注册服务配置

**config\_slave.grid的内容：**

#

# The IceGrid locator proxy.

#

Ice.Default.Locator=IceGridRDDataSource/Locator:default -h 10.0.2.241 -p 12000:default -h 10.0.2.242 -p 12000 #1

#

# IceGrid registry configuration.

#

IceGrid.Registry.Client.Endpoints=default -p 12000 #2

IceGrid.Registry.Server.Endpoints=default #3

IceGrid.Registry.Internal.Endpoints=default #4

IceGrid.Registry.Data=slave\_1 #5

IceGrid.Registry.ReplicaName=slave\_1 #6

IceGrid.Registry.PermissionsVerifier=IceGridRDDataSource/NullPermissionsVerifier #7

IceGrid.Registry.AdminPermissionsVerifier=IceGridRDDataSource/NullPermissionsVerifier#8

IceGrid.Registry.SSLPermissionsVerifier=IceGridRDDataSource/NullSSLPermissionsVerifier#9

IceGrid.Registry.AdminSSLPermissionsVerifier=IceGridRDDataSource/NullSSLPermissionsVerifier #10

#

# IceGrid SQL configuration if using SQL database.

#

#Ice.Plugin.DB=IceGridSqlDB:createSqlDB #11

#IceGrid.SQL.DatabaseType=QSQLITE #12

#IceGrid.SQL.DatabaseName=register/Registry.db #13

#

#

#Ice Error and Standard output Set

#

#Ice.StdErr=slave\_1/stderr.txt #14

#Ice.StdOut=slave\_1 /stdout.txt #15

#

#Trace Registry properties

#

Ice.ProgramName=Slave\_1 #16

IceGrid.Registry.Trace.Node=3 #17

IceGrid.Registry.Trace.Replica=3 #18

#

# IceGrid node configuration.

#

IceGrid.Node.Name=node\_2 #19

IceGrid.Node.Endpoints=default #20

IceGrid.Node.Data=node\_2 #21

IceGrid.Node.CollocateRegistry=1 #22

#IceGrid.Node.Output=node\_2 #23

#IceGrid.Node.RedirectErrToOut=1 #24

# Trace properties.

#

IceGrid.Node.Trace.Activator=1 #25

#IceGrid.Node.Trace.Adapter=2 #26

#IceGrid.Node.Trace.Server=3 #27

#

# Dummy username and password for icegridadmin.

#

IceGridAdmin.Username=mygrid #28

IceGridAdmin.Password=mygrid #29

**配置项说明：**

其实这个文件和主注册配置文件基本一样，差别只有一点：

1. *没有指定应用实例名，因为在主注册服务中已经有了定义*
2. *多了第6行，IceGrid.Registry.ReplicaName=slave\_1，指定从注册服务的名称*

其它的基本就没有差别了，大部分属性项在config\_master.grid里面都有定义，为了方便阅读，下面也将用到的各项给出说明：

# 1 注册服务的端点信息(主注册服务和所有的从注册服务)，节点注册时要用到

# 2 客户端访问注册服务器的端点信息

# 3 服务访问注册服务器的端点信息，通常是default

#4 内部访问端点信息，通常是default，节点用这个端口和注册服务通信

# 5 注册服务的数据目录的路径

# 6 指定从注册服务的名称

# 7 设定防火墙安全代理，从而控制客户端访问注册表时可用的权限

#8 设定防火墙安全代理，从而控制注册表管理者可用的权限

# 9 设定SSL安全代理，从而设定客户端访问注册表时的SSL安全访问机制

#10 设定SSL安全代理，从而设定注册表管理者的SSL安全访问机制

# 11 指定Ice对象序列化的机制，如果不设置，默认用Freeze机制

# 12 指定使用数据库的类型

# 13 指定使用数据库的名称

# 14 指定标准错误输出文件

# 15 指定标准输出文件

# 16 指定从注册服务运行时程序名称

# 17 指定从注册服务跟踪节点信息的级别（0~3），默认为0

# 18 指定主/从热备注册服务的跟踪级别（0~3），默认为0

# 19 定义节点的名称，必须唯一

# 20 节点被访问的端口信息，注册服务使用这个端点和节点通信，通常设为default

# 21 节点的数据目录的路径

# 22 定义节点是否和注册服务并置在一起，设为1时并置，设为0时不并置

# 23 节点标准输出信息重定向蹈的目录路径，会自动生成输出文件

# 24 节点上的服务程序的标准错误重定向到标准输出

# 25 激活器跟踪级别，通常有0，1，2，3级，默认是0

# 26 对象适配器跟踪级别，通常有0，1，2，3级，默认是0

# 27 服务跟踪级别，通常有0，1，2，3级，默认是0

# 28 IceGrid管理器登录该应用的用户名

# 29 IceGrid管理器登录该应用的密码

### 2.1.3 应用部署配置

**app.xml配置文件内容：**

1<icegrid>

2 <application name=*"RTDSSystem"*>

3 <server-template id=*"RTDSSystemServer"*>

4 <parameter name=*"index"*/>

5 <server id=*"RTDSSystemServer-${index}"* exe=*"java"* activation=*"on-demand"*>

6 <adapter name=*"RTDataSysytem"* endpoints=*"tcp"* replica-group=*"ReplicatedRTDataSysytemAdp"*/>

7 <option>-jar</option>

8 <option>ServerApp.jar</option>

9 </server>

10 </server-template>

11

12 <replica-group id=*"ReplicatedRTDataSysytemAdp"*>

13 <load-balancing type=*"round-robin"*/>

14 <object identity=*"RTDataSource"* type=*"::RTDataSystem::RTDataSource"*/>

15 </replica-group>

16

17 <node name=*"node\_1"*>

18 <server-instance template=*"RTDSSystemServer"* index=*"1"*/>

19 <server-instance template=*"RTDSSystemServer"* index=*"11"*/>

20 <server-instance template=*"RTDSSystemServer"* index=*"111"*/>

21 </node>

22 <node name=*"node\_2"*>

23 <server-instance template=*"RTDSSystemServer"* index=*"2"*/>

24 <!--server-instance template="RTDSSystemServer" index="22"/-->

25 <!--server-instance template="RTDSSystemServer" index="222"/-->

26 </node>

27 <node name=*"node\_3"*>

28 <server-instance template=*"RTDSSystemServer"* index=*"3"*/>

29 <!--server-instance template="RTDSSystemServer" index="33"/-->

30 <!--server-instance template="RTDSSystemServer" index="333"/-->

31 </node>

32 </application>

33</icegrid>

**配置文件结构分析：**

IceGrid里，部署是一个在注册服务中表述一个应用（Application）的过程，而部署配置文件就是来描述这些配置信息的文件，这个配置文件是用xml标记性语言来描述的。通常一个部署应该包含如下信息：

1. 应用标签（application）,name属性定义这个应用的名字
2. 服务（server）, 一个逻辑上的服务器，能够通过exe命令而启动的一个服务程序。activation属性，是设置服务的启动方式，on-demand是最常用的方式，另外还有always等启动方式；option标签是exe执行命令命令行的参数；
3. 适配器（adpter），定义服务器端的适配器。

name属性唯一标志这个适配器；

endpoints属性指定端点信息；

replica-group属性标示该适配器是个可复制组集群，并指定这个可复制组的名称；

register-process属性定义了是否这个节点是否可以被icegrid关闭；

1. 节点（node），它应该代表了一个物理上的节点。

name属性指定节点的名字，并且是唯一的。

1. 可复制组（replica-group），一组对象适配器的集合。

id属性唯一标识一个可复制组；

load-balancing子项中type属性指定负载均衡策略，icegrid提供了四种负载均衡策略： Random (随机方式)

Adaptive(适配方式)

Round Robin(最近最少使用)

Ordered（顺序方式）

object子项定义适配器绑定的服务对象信息。其中identity属性指定对象的标识，type属性指定了对象的层次结构类型。这两个属性都可以唯一的标识一个服务对象。

1. 服务模板（server-temple），服务模板是对服务的一个抽象，避免了重复定义。这样，在节点中描述服务时只需要实例化它的服务模板就可以了。

id属性唯一标识一个服务模板；

parameter子项定义服务模板的参数，可包含多个，主要实例化服务时用；

server子项就是上面2中的服务定义；

另外还有一些特殊的服务模板，比如：icebox服务模板，它的定义和通用的服务模板的定义不太一样。

**解析app.xml文件：**

通过对配置文件结构的分析，来解析一下app.xml。

第1行，标识这是一个icegrid的配置文件；

第2行，标识应用的名称为*RTDSSystem，*这个名称是唯一的；

第3~10行，定义了一个服务模板*RTDSSystemServer，*并有一个参数index；

其中5~9定义了这个模板包含的服务定义，第6行是这个服务包含的对象适配器

的定义；

第12~15行，是对可复制组的定义，包括服务对象的定义和负载均衡策略；

第17~21行，是对节点node\_1的定义，指定了节点的名称，包含的服务（3个服务）；

第22~26行，是对节点node\_2的定义

第27~31行，是对节点node\_3的定义

最后两行是闭合标签，至此一个icegrid的分布式部署配置文件就完成了。

**部署配置文件的扩展：**

app.xml中对服务模板、适配器、服务对象等的配置都是一个，事实上这些可以在文件中定义多个，比如可以有多个服务模板，一个服务里可以有多个适配器，可以有多个可复制组，一个节点里可以有多个不同类型的服务等。

另外，app.xml可以包含其它的xml。

### 2.1.4 节点配置

**config.grid文件的内容：**

#

# The IceGrid locator proxy.

#

Ice.Default.Locator=IceGridRDDataSource/Locator:default -h 10.0.2.241 -p 12000:default -h 10.0.2.242 -p 12000 #1

#

# IceGrid node configuration.

#

IceGrid.Node.Name=node\_2 #2

IceGrid.Node.Endpoints=default #3

IceGrid.Node.Data=node\_2 #4

IceGrid.Node.Output=node\_2 #5

IceGrid.Node.RedirectErrToOut=1 #6

# Trace properties.

#

IceGrid.Node.Trace.Activator=1 #7

#IceGrid.Node.Trace.Adapter=2 #8

#IceGrid.Node.Trace.Server=3 #9

**配置项说明：**

事实上，这个文件里面的配置项，在config\_slave.grid中都有描述，但这里也列出来，方便阅读。

#1 注册服务的端点信息(主注册服务和所有的从注册服务)，节点注册时要用到

#2 定义节点的名称，必须唯一

#3 节点被访问的端口信息，注册服务使用这个端点和节点通信，通常设为default

#4 节点的数据目录的路径

#5 节点标准输出信息重定向的目录路径，会自动生成输出文件

#6 节点上的服务程序的标准错误重定向到标准输出

#7 激活器跟踪级别，通常有0，1，2，3级，默认是0

#8 对象适配器跟踪级别，通常有0，1，2，3级，默认是0

#9 服务跟踪级别，通常有0，1，2，3级，默认是0

## 客户端配置

客户端的配置很简单，和分布式相关的配置就一项，添加如下：

#

# The IceGrid locator proxy.

#

Ice.Default.Locator=IceGridRDDataSource/Locator:default -h 10.0.2.241 -p 12000:default -h 10.0.2.242 -p 12000 #注册服务的端点信息(主注册服务和所有的从注册服务)，用于定位

# 结果验证

## 3.1 程序方式

1. 启动服务器

1. icegridnode --Ice.Config=config\_master.grid 启动主注册服务和节点1
2. icegridnode --Ice.Config=config\_slave.grid 启动从注册服务和节点2
3. icegridadmin --Ice.Config=config\_master.grid -e "application add app.xml" 部署分布式服务
4. icegridnode --Ice.Config=config.node 启动节点3

2. 启动客户端，进行多次远程调用，根据执行情况就可以判断服务器端是否配置成功。

## 3.2 工具方式

用Ice官方提供的可视化管理工具IceGridGUI.jar来验证和管理icegrid的部署。

1. 打开dos窗口，在命令行下进入C:\Program Files\ZeroC\Ice-3.4.1\bin目录下，然后运行“java –jar IceGridGUI.jar”，弹出IceGrid Admin的主界面，如下图所示：

1.TIF

图3.2.1主界面

1. 点击按钮File—>login，弹出登录对话框，输入用户名、密码，Ice Instance Name和ice Registry endpoint，点击“OK”按钮，就可以进入该部署的管理界面2.TIF

图3.2.2 File菜单

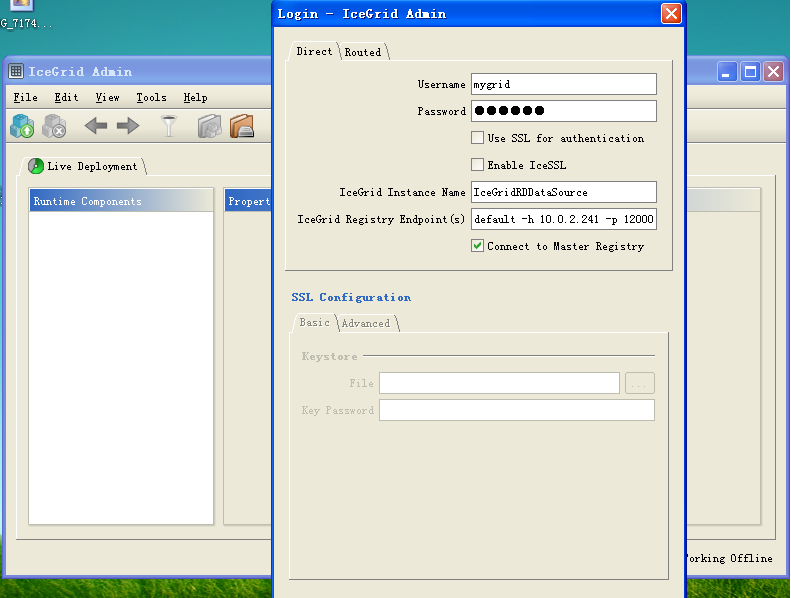


图3.2.3 登陆窗口

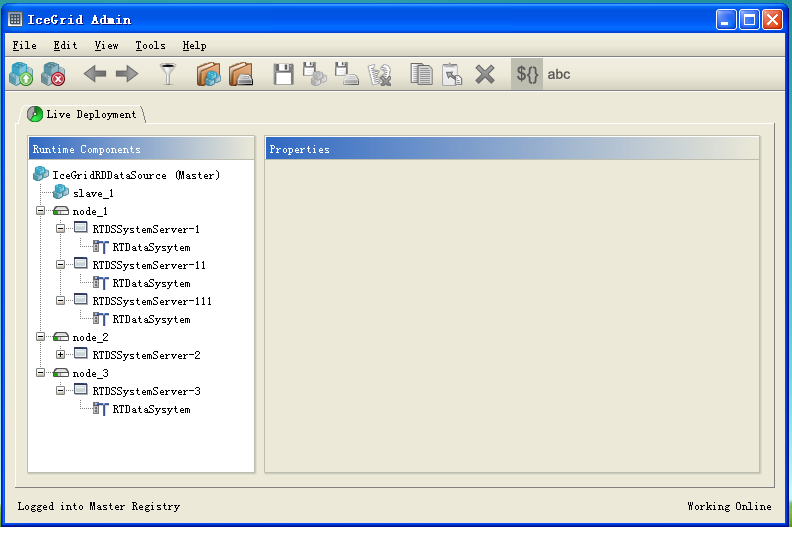


图3.2.4 实时管理（服务已加载，但未激活）

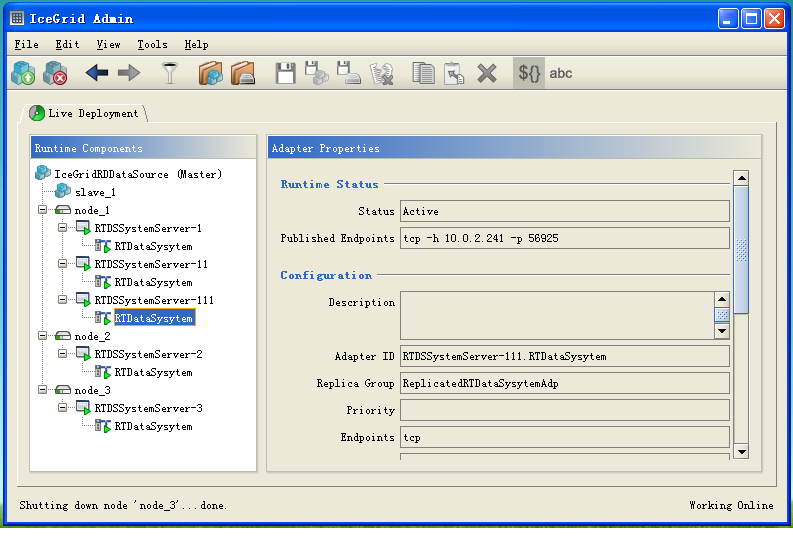


图3.2.5 实时管理（服务已激活）

1. 由图3.2.4和3.2.5可以看到，该应用部署的名字是IceGridRDDataSource，它进行了主/从注册服务的热备（Master/slave\_1）,当Master崩溃后，slave\_1就会接收客户端的请求；同时该应用有三个节点（node\_1,node\_2,node\_3）,每个节点都分布了同一类服务，通过负载均衡的方式共同完成客户端的请求，其中node\_1部署了三个，node\_2和node\_3分别部署了一个。
2. 注册服务（Master/Slave）可以通过该管理工具被关闭，在其上面点击右键，选取弹出的菜单项“shutdown”。只要存在一个活动状态的注册服务，整个应用依然能够提供服务。由于该工具登陆时默认使用的是主注册服务（Master），所以，当关闭Master后管理界面将被关闭，如果想再次进入管理界面，需要重新登陆并选择slave注册服务器，因为主注册服务已经被关闭。

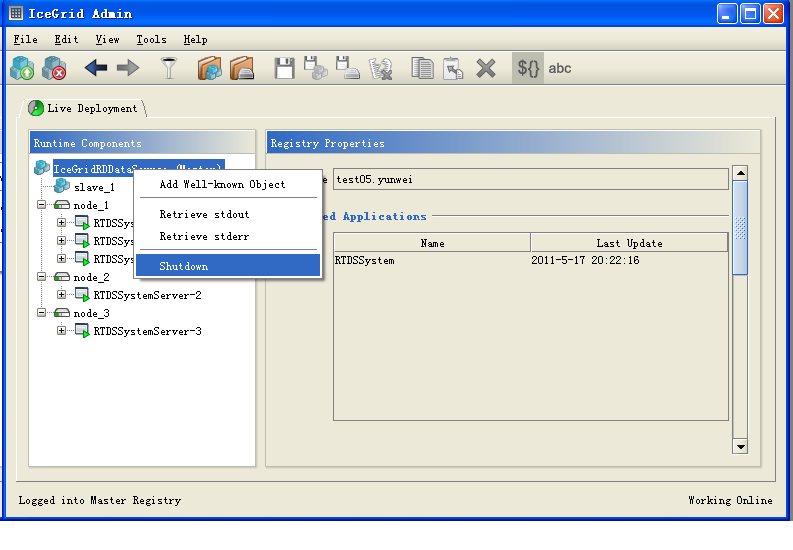


图3.2.6 关闭Master

1. 同注册服务器一样，各个节点也是可以用同样的方式被关闭，同样由于主注册服务和节点1的应用服务集成在一起，当节点1被关闭时，主注册服务也将结束。如果希望再次进入管理界面，需要用从注册服务（slave）重新登陆。

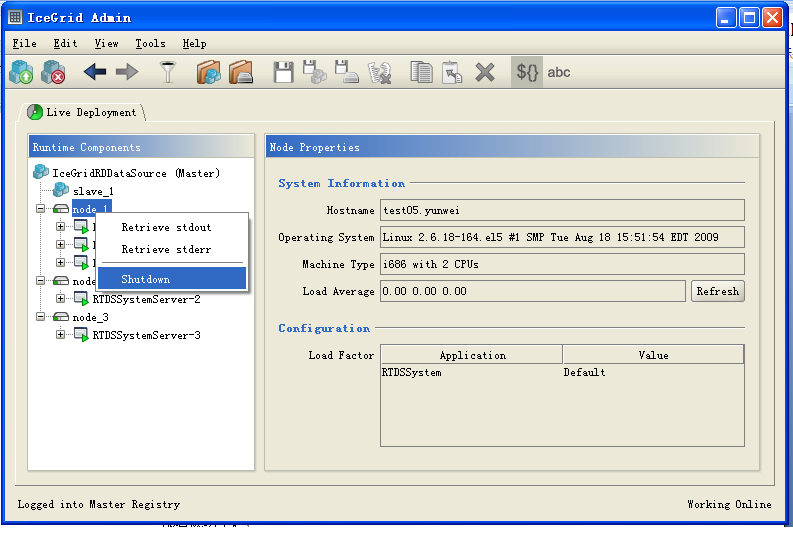


图3.2.7 关闭节点

1. 每个节点上的服务可以被启动、停止，也可以被设置为有效或失效，还有一些其它的功能菜单，这里不再详述，这些操作通过右键菜单都可以方便的进行。

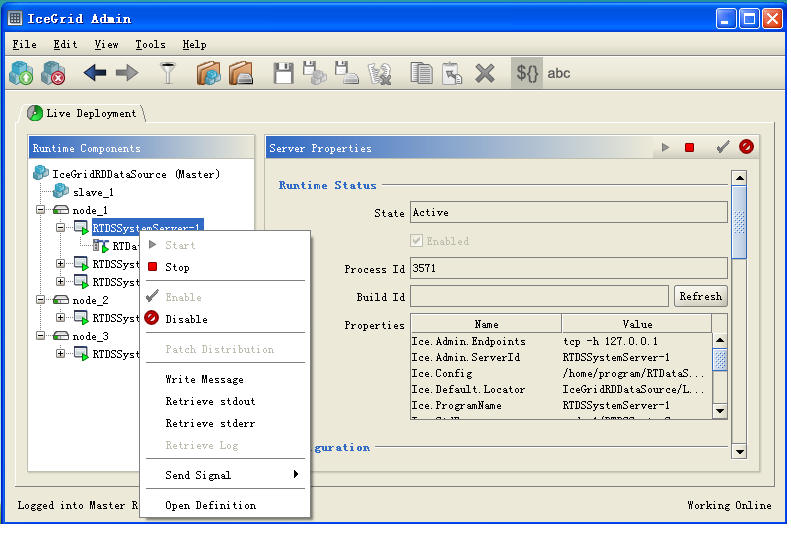


图3.2.8 关闭节点

1. 如果我们能看到部署的各个注册服务、节点、各个应用服务，并能操纵正常，就说明部署成功了。

# 高级应用配置

## 4.1 集成IceBox

在文档《IceBox开发和配置》（当前是1.0版）中，介绍了一个IceBox服务程序的开发方法和单独应用中配置和管理的过程。在实际的应用中，IceBox服务通常集成到IceGrid中，并通过IceGrid进行激活和部署。

本章节中IceBox服务是集成在IceGrid中，并通过IceGrid进行部署，所以IceBox服务的配置信息不再同《IceBox开发和配置》中一样在config.icebox中描述，而是直接配置在部署文件app.xml中。那也就是说，IceGrid集成IceBox服务，只需要在app.xml文件中添加Icebox服务相关的配置信息就可以了。事实上，有关Ice所有的配置信息（除IceGrid自身的配置信息），都可以添加到app.xml中，并通过icegrid部署后生效。

下面各节详细描述IceBox服务的集成过程。

### 4.1.1 IceBox服务程序编写

请参考文档《IceBox开发和配置》，这里不再详述。由于IceBox服务相关的配置信息都放在了app.xml中，并且服务是通过IceGrid按需激活的，因此这里程序代码略有调整。下面列出IceBox服务的实现代码：

文件名：ServerService.java

**import** main.java.DataSource;

**import** IceBox.Service;

**public** **class** ServerService **implements** IceBox.Service {

/\*\*

\* **@param** name 配置文件中的service名称

\* **@param** communicator对象,由IceBox.ServiceManager负责创建和销毁。

\* 可能同时被其他服务共享使用（由配置文件决定），object Adapter的名

\* 称必须是唯一的；

\* **@param** args 配置文件中的参数列表

\* **@Override**

\*\*/

**public** **void** start(String name, Ice.Communicator communicator,

String[] args){

//创建objectAdapter，名称有配置文件决定

Adapter = communicator.createObjectAdapter(

"RTDataSystem-"+name);

//创建servant

String RTDataSourceIdentity = communicator.getProperties().

getProperty("RTDataSource.Identity");

DataSource objDataSrc = **new** DataSource("dataSource");

Adapter.add(objDataSrc,

communicator.stringToIdentity(RTDataSourceIdentity));

Adapter.activate();

}

/\*\*

\*

\* **@param** args

\* **@Override**

\*

\*\*/

**public** **void** stop()

{

Adapter.destroy();

}

**private** Ice.ObjectAdapter Adapter;

}

### 4.1.2 IceGrid集成IceBox服务

IceGrid集成IceBox只和部署文件（app.xml）有关，IceBox服务（service）的粒度和普通的server是一样的，因此IceBox service的部署和普通的server非常类似，它同样有模板、服务（service）和实例化的概念，可以将IceBox service理解为一个特殊的server。

为了能更清楚的描述这个集成配置的过程，在IceGrid配置的基础上，添加IceBox服务。具体目标如下：

1. 集成ServerService服务（service），并且ServerService服务（service）使用的服务对象和之前server的服务对象使用同一个（type--::RTDataSystem::RTDataSource）
2. 在节点1（node\_1）上添加IceBox服务功能（IceBox-Node1），这个IceBox服务包含了5个ServerService服务；同样的在节点2（node\_2）上也添加一个IceBox服务功能（IceBox-Node2），也包含了5个ServerService服务
3. 这些IceBox服务中分布的多个服务（service）和之前已经存在的服务（server）一起通过IceGrid实现负载均衡

为了实现上述的功能，需要添加IceBox服务的相关配置，首先看一下此时app.xml的变化，变化和添加部分用浅灰阴影标出。

**app.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<icegrid>

<application name="**RTDSSystem**">

<server-template id="**RTDSSystemServer**">

<parameter name="**index**"/>

<server id="**RTDSSystemServer-${index}**" exe="**java**" activation="**on-demand**">

<adapter name="**RTDataSysytem**" endpoints="**tcp**"

replica-group="**RTDataSystemGroup**"/>

<option>**-jar**</option>

<option>**ServerApp.jar**</option>

</server>

</server-template>

<!— begin服务模板定义-->

1 <service-template id="**RTDSystemService**">

2 <parameter name="**name**"/>

3 <service name="**${name}**" entry="**ServerService**">

4 <description>A simple service named after ${name}</description>

5 <properties>

6 <property name="**RTDataSource.Identity**" value="**RTDataSource**"/>

7 </properties>

8 <adapter name="**RTDataSystem-${name}**" endpoints="**tcp**"

id="**RTDataSystem-${name}**" replica-group="**RTDataSystemGroup**"

server-lifetime="**false**"/>

11 </service>

12 </service-template>

<!-- end服务模板定义-->

<replica-group id="**RTDataSystemGroup**">

<load-balancing type="**round-robin**"/>

<!--load-balancing type="ordered" /-->

<!--load-balancing type="adaptive" /-->

<!--load-balancing type="random" n-replicas="0"/-->

<object identity="**RTDataSource**" type="**::RTDataSystem::RTDataSource**"/>

</replica-group>

<node name="**node\_1**">

<server-instance template="**RTDSSystemServer**" index="**1**"/>

<server-instance template="**RTDSSystemServer**" index="**11**"/>

<server-instance template="**RTDSSystemServer**" index="**111**"/>

<!— begin IceBox服务配置 IceBox-Node1-->

1 <icebox id="**IceBox-Node1**" activation="**on-demand**" exe="java">

2 <description>**A sample IceBox server IceBox-Node1**</description>

3 <option>**IceBox.Server**</option>

4 <properties>

5 <property name="**IceBox.InstanceName**" value="**${server}**"/>

6 <property name="**Ice.Admin.Endpoints**" value="**tcp -h 10.0.2.241**"/>

7 <property name="**IceBox.Trace.ServiceObserver**" value="**1**"/>

8 </properties>

9 <service-instance template="**RTDSystemService**" name="**one**"/>

10 <service-instance template="**RTDSystemService**" name="**two**"/>

11 <service-instance template="**RTDSystemService**" name="**three**"/>

12 <service-instance template="**RTDSystemService**" name="**four**"/>

13 <service-instance template="**RTDSystemService**" name="**five**"/>

14 </icebox>

<!— end IceBox服务配置 IceBox-Node1-->

</node>

<node name="**node\_2**">

<server-instance template="**RTDSSystemServer**" index="**2**"/>

<server-instance template="**RTDSSystemServer**" index="**22**"/>

<server-instance template="**RTDSSystemServer**" index="**222**"/>

<!— begin IceBox服务配置 IceBox-Node2-->

1 <icebox id="**IceBox-Node2**" activation="**on-demand**" exe="**java**">

2 <description>**A sample IceBox server IceBox-Node2**</description>

3 <option>**IceBox.Server**</option>

4 <properties>

5 <property name="**IceBox.InstanceName**" value="**${server}**"/>

6 <property name="**Ice.Admin.Endpoints**" value="**tcp -h 10.0.2.242**"/>

7 <property name="**IceBox.Trace.ServiceObserver**" value="**1**"/>

8 </properties>

9 <service-instance template="**RTDSystemService**" name="**2-one"**/>

10 <service-instance template="**RTDSystemService**" name="**2-two**"/>

11 <service-instance template="**RTDSystemService**" name="**2-three**"/>

12 <service-instance template="**RTDSystemService**" name="**2-four**"/>

13 <service-instance template="**RTDSystemService**" name="**2-five**"/>

14 </icebox>

<!— begin IceBox服务配置 IceBox-Node2-->

</node>

<node name="**node\_3**">

<server-instance template="**RTDSSystemServer**" index="**3**"/>

</node>

</application>

</icegrid>

app.xml中增加的IceBox服务相关的配置部分如下：

* **服务摸板(Service Template)：**

可以对比一下server template的定义，两者基本上没有什么区别，最大的不同是

Server template中server是指定一个可执行的程序，而service中指定的是动态加载

的组件入口。以下解释上述配置中的服务模板的定义：

第1行指定定义模板的id，唯一标志一个服务模板，第12是闭合标签；

第2行定义了一个参数name，默认值是“name”；

第3~11行定义了模板中使用的服务（service），并在该service中指定了名称、

入口、描述信息、配置属性，定义了一个对象适配器；

第4行，是该服务的描述信息；

第5~7行，是属性定义列表，这里定义了一个属性RTDataSource.Identity，并

指定其值为RTDataSource；

第8行，定义了一个对象适配器，指定了其name、endpoints、id、replica-group

等属性信息，这个基本上和server中adapter的定义没有什么区别

*以上内容就是service模板的定义。*

* **IceBox服务(IceBox-Node1)：**

icebox服务的定义被包含在分布的服务器节点中，主要包括三部分的信息：

1. IceBox服务的启动配置信息
2. IceBox的属性配置信息
3. Service服务实例化列表

下面解释这块内容：

第1~3行，指定了IceBox服务的名称，启动方式，启动执行程序等

第4~8行，指定了IceBox服务的属性配置列表，这里定义了IceBox服务的实

例名称、管理器访问端点以及service被跟踪的级别

第9~13行，实例化了5个service服务

至此，一个包含了5个serverservice服务的IceBox服务被集成在node1中。

* **IceBox服务(IceBox-Node2)**

同IceBox服务(IceBox-Node1)中描述，只是具体value有所不同，这里不再解释。

### 4.1.3 测试验证

验证方式同第3章，这里不再赘述。部署完成后，就可以通过IceGridGUI.jar程序来进行管理，基本情况如下图所示：

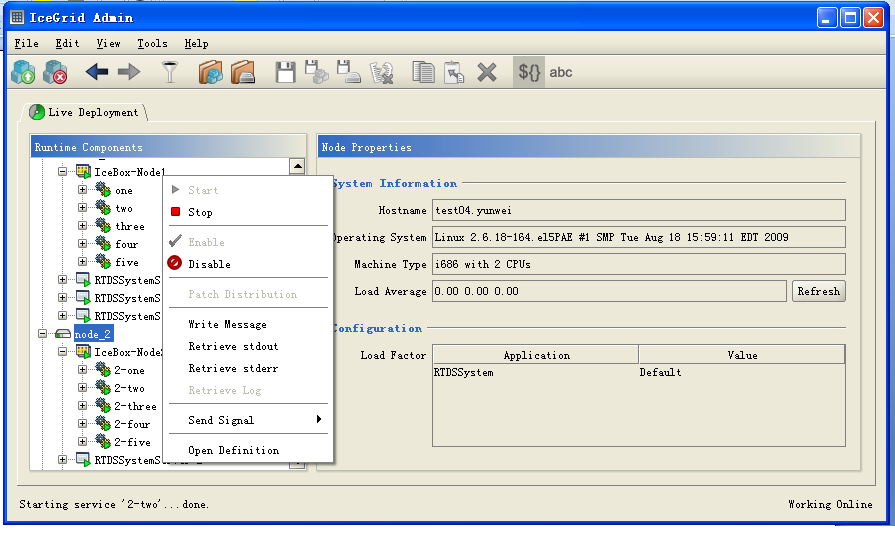


图4.1.3.1 IceGridGUI工具管理IceBox服务

运行测试程序（jsp）进行远程调用，打开浏览器，在地址栏中输入：<http://10.0.2.243:8080/testApp/>，按回车键。如果能得到结果，就说明服务器端已经正常工作了，如下图所示：

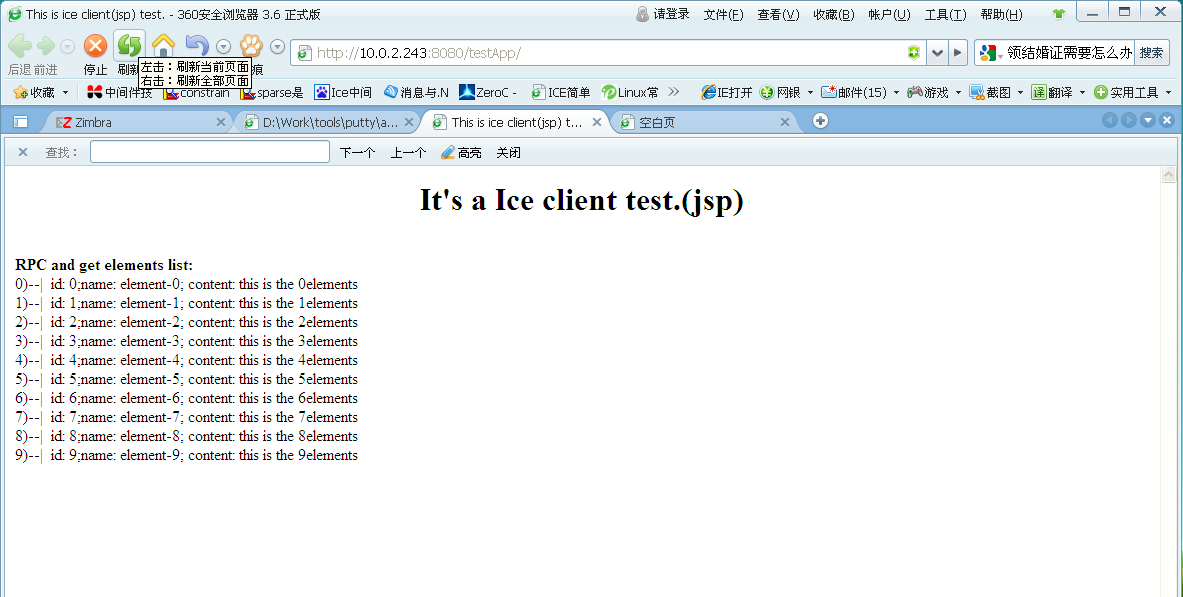


图4.1.3.2 jsp客户端远程调用

查看服务器端IceBox服务中service执行的情况，以节点1为例（对应的服务器ip：

10.0.2.241），监测service运行的情况：

tail –f /home/program/RTDataSystem/node\_1/IceBox-Node1.out

多刷新几次浏览器，此时IceGrid就会通过负载均衡，将客户端的一些请求转移过来，

如下图所示：

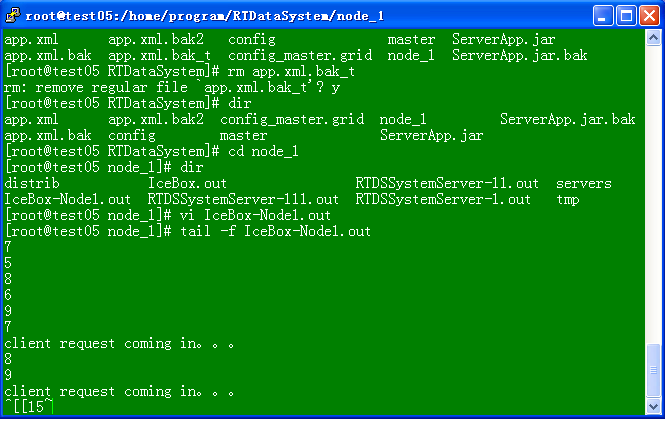


图4.1.3.3 服务器端service响应客户端请求

## 4.2 集成IcePatch2

下个版本补充