计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(12):222−226 [doi: 10.15888/j.cnki.csa.006693]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Zookeeper 的分布式 ICE 中间件研究①

冯战胜, 张　激, 彭　宏, 刘颖娜

(华东计算技术研究所, 上海 201808)
通讯作者: 冯战胜, E-mail: fovercome@live.com

摘　要: 分布式 ICE 中间件是一款高效的 RPC 框架, 它支持跨语言开发, 其中的 IceGrid 服务是整个框架的核心, 主要的作用是给客户端提供位置管理, 远程管理, 负载均衡和服务注册等服务. 在 ICE 构建的分布式平台中, 一旦 IceGrid 服务出现问题, 整个网络通信就会出现问题. IceGrid 本身支持主从模式的高可用, 但是当主节点不可用时, 从节点不会自动升级为主节点, 必须在从节点进行手动配置重启, 这严重影响了 IceGrid 的效率. Zookeeper 是开放源码的分布式应用程序协调服务, 它是一个为分布式应用提供一致性服务的软件, 本文提出一种基于 Zookeeper 的 IceGrid 服务的高可用改进模型, 当主节点不可访问时, 由 Zookeeper 在从节点中进行选举协调, 选中的从节点自动升级为主节点.

关键词: Zookeeper; 分布式 ICE 中间件; 高可用; 主从切换; 配置管理

引用格式: 冯战胜,张激,彭宏,刘颖娜.基于 Zookeeper 的分布式 ICE 中间件研究.计算机系统应用,2018,27(12):222–226. http://www.c-s-a.org.cn/1003-3254/6693.html

Research on Distributed ICE Middleware Based on Zookeeper

FENG Zhan-Sheng, ZHANG Ji, PENG Hong, LIU Ying-Na

(East China Institute of Computing Technology, Shanghai 201808, China)

Abstract: Distributed ICE middleware is an efficient RPC framework that supports cross language development, and the IceGrid service is the core of the whole framework. The main function is to provide services such as location management, remote management, load balancing and service registration for clients. In the distributed platform constructed by ICE, once the IceGrid service is in trouble, the whole network communication will have problems. IceGrid itself supports the high availability of the master slave mode, but when the master is not available, slave will not automatically upgrade to the master, and it must be restarted manually, which seriously affects the efficiency of the IceGrid. Zookeeper is an open source distributed application coordination service. It is a software that provides conformance services for distributed applications. This paper presents a highly available and improved model for IceGrid services based on Zookeeper. When the master is not accessible, Zookeeper selects a new master from one of slaves automatically.

Key words: Zookeeper; distributed ICE middleware; high availability; master-slave switching; configuration management

在互联网大行其道的今天, 各种分布式系统已经司空见惯. 搜索引擎、电商网站、微博、微信、O2O 平台等, 凡是涉及到大规模用户、高并发访问的, 无一不是分布式[1]. RPC 作为分布式架构的核心, 一旦出现问题, 整个分布式网络就会出现瘫痪, RPC 框架必须做到高可用才能保证分布式网络的稳定. ICE 中间件作为一款跨语言开发的 RPC 框架, 因其高效的性能和完整的服务, 被惠普等大型企业所采用.

① 收稿时间: 2018-05-29; 修改时间: 2018-06-19; 采用时间: 2018-06-28; csa 在线出版时间: 2018-12-03

Zookeeper 是开放源码的分布式应用程序协调服务, 许多分布式软件都采用 Zookeeper 作为集群[2]管理, Zookeeper 本身也支持高可用, 随着软件版本的更新, Zookeeper 的高效与稳定很适合作为分布式环境下的高可用服务[3].

1 Zookeeper 技术简介

Zookeeper 是 Hadoop 的正式子项目, 是一个高效可靠的开源的分布式协调服务框架, 由知名互联网公司雅虎创建, 是 Google 公司的分布式服务框架 Chubby 的开源实现. 在分布式环境下, Zookeeper 可以保证顺序一致性, 原子性, 可靠性和实时性.

Zookeeper 本质上是一个分布式的小文件存储系统, 具有下述应用特性:

(1) 简单. Zookeeper 允许通过分布式程序共享方式, 组织成类似标准文件系统层级命名空间协调的分布式程序. 这个命名空间包含类似文件和目录的数据寄存器, 不像典型的用于存储的文件系统. Zookeeper 的数据保存在内存中, 可实现高吞吐量和低延迟访问.

(2) 冗余. Zookeeper 本身和其自身协调的程序一样, 设计为冗余的, 在拥有许多主机集合的主机组上运行. 组成 Zookeeper 服务群的所有服务器都已知对方, 只要所有服务器中的主服务器可用, Zookeeper 服务就是可用的. 客户端链接的一个 Zookeeper 服务器通过发送请求、获取响应、监听事件、发送心跳维持一个 TCP(传输控制协议) 链接, 如果链接中断, 客户端将连接到另外一个服务器

(3) 有序. Zookeeper 用一个反映所有 Zookeeper 事务顺序的数字, 标记每一个更新. 以后的操作可以使用这个顺序数字标记.

(4) 快速. Zookeeper 在读取占主要地位的负载环境下, 在读取操作比写入操作更频繁的情况下, 读取速度非常快.

(5) 丰富的 API(应用程序编程接口). Zookeeper 为开发人员提供了一套丰富的 API, 减轻了开发人员编写通用协议的负担[4].

1.1 Zookeeper 的配置管理

程序在启动运行中总是需要配置文件, 如果程序分散部署在多台机器上, 要逐个改变所在机器的配置文件就变得困难. 现在可以把这些配置全部放到 Zookeeper 上去, 保存在 Zookeeper 的某个目录节点中, 然后所有相关应用程序对这个目录节点进行监听, 一旦配置信息发生变化, 每个应用程序就会收到 Zookeeper 的通知, 然后从 Zookeeper 获取新的配置信息应用到系统中[5]. 基本过程如图 1.

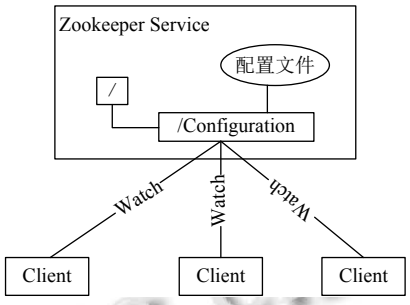

图 1 Zookeeper 的配置管理

1.2 Zookeeper 的集群管理

所有机器约定在父目录 father 下创建临时目录节点, 然后 Zookeeper 监听父目录节点的子节点的消息变化. 一旦有机器出现问题, 该机器与 Zookeeper 的连接就会断开, 其所创建的临时目录节点就会被删除, 所有的其他机器都会收到通知.

当有新机器加入, 就会在 father 父目录下创建临时节点, Zookeeper 会把这个消息发送给父目录下的所有机器, 所有机器就会收到新机器加入的通知. 如图 2.

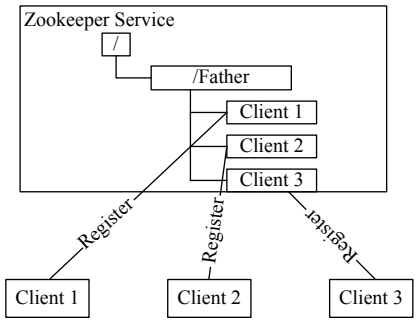

图 2 Zookeeper 的集群管理

1.3 Zookeeper 的选举机制

Zookeeper 的角色主要有 3 类: 领导者 (leader)、学习者 (learner) 和客户 (client), 三者关系如图 3 所示. 领导者负责处理写请求和系统状态更新请求; 学习者负责接收领导者发送的请求并对请求进行投票, 以及客户端的读写请求; 客户端负责通过 TCP 链接向学习者发送请求和获取响应. 学习者又可细分为跟随者 (follower) 和观察者 (observer), 两者的区别为: 观察者只同步领导者的状态, 但不参与领导者失效时的竞选和投票[6].

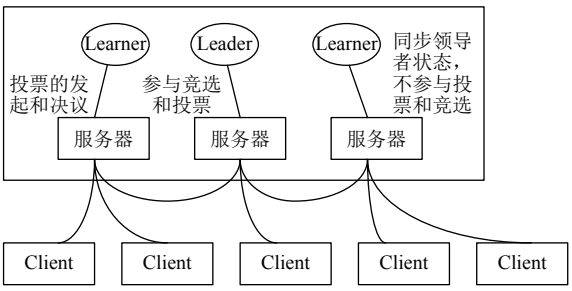

图 3 Zookeeper 的角色关系

当领导者失去响应时, 进入恢复模式 (竞选), 某服务器推荐自己作为领导者, 将消息发送给集群, 等待其它服务器接收该举荐信息. 若同意, 向领导者服务器发送同意消息; 若不同意, 重复自荐流程, 直到获得 N/2+1 票数后, 才被认定为领导者, 整个竞选流程结束. 具体流程如下:

首先, 某台服务器发起选举线程, 选举线程向所有服务器发起一次询问. 当收到回复, 选举线程验证 zxid (Zookeeper transaction ID), 判断是否为自己发起的询问, 并获 取对方 id, 同时保存到此次选举投票记录表中. 当收到所有服务器回复后, 选举线程计算并选出 zxid 值最大的服务器, 将该服务器的信息, 设置为被推荐的领导者. 被推荐的领导者发起选举, 向其它服务器发送 PROPOSAL 消息, 其它服务器接收该消息后, 进行投票; 若同意, 则发送 ACK 消息. 此时, 被推荐的领导者判断是否有超过 *N*/2+1 的服务器同意; 若超过, 则被推荐的领导者获胜; 否则, 重复流程, 直到领导者被选出.

1.4 Watcher 机制

客户端在所关心的目录节点上注册 Watcher, 当被监听的节点状态发生变化时 (即数据改变、被删除、子目录节点增加或删除时), 会触发 WatchedEvent 事件, 并发送给对其进行监听的客户端. Watcher 机制可以保证整个集群系统中各个服务器之间的通信和协调, 实现对数据变更的实时处理. Zookeeper 所提供的基本操作包括: create(创建节点)、delete(删除节点)、setData(设置节点数据)、getData(获取节点数据)、exists(查看节点是否存在)、以及 getChildren(获取子节点) 等. 其中, exists、getChildren、getData 操作可以在节点上注册 Watcher; 而 create, delete, setData 操作可以触发设置在节点上的 Watcher[7].

在该机制中, 服务端存储事件的信息, 客户端存储事件的信息和 Watcher 的执行逻辑. 当在某节点上注册 Watcher 事件时, 客户端线程向服务器注册 Watcher; 同时, 在 Watcher 对象管理器中存储 Watcher 对象以及对应的节点路径. 当某节点触发 Watcher 事件时, 服务器向对应的客户端发送通知 (包含事件类型和节点路径), 该客户端线程从 Watcher 对象管理器中根据映射关系取出对应的 Watcher 对象, 执行回调函数. 其中, 回调函数包含 WatchEvent 类型的参数, 该类型参数 (WatchEvent) 封装事件的通知状态 (keeperState)、事件类型 (EventType) 和节点路径 (path), 方便回调函数对事件进行处理.

2 ICE 中间件简介

ICE 是一种新的面向对象的中间件技术. 从根本上说, ICE 为构建面向对象的客户－服务器应用提供了工具、API 和库支持[8]. 可用于替代像 CORBA 或 COM/DCOM/COM+这样的中间件, 为构建面向对的分布式客户服务器应用提供了平台支持. ICE 应用适合在异种环境中使用: 客户和服务器可以用不同的编程语言编写, 可以运行在不同的操作系统和机器架构上, 并且可以使用多种网络技术进行通信. 在最新的 3.7 版本中, 支持 c++, c#, .Net, java, javascript, objective-c, python, ruby 等编程语言, 操作系统支持 windows, linux 和 mac os, 支持最新的 docker 部署, 并且无论部署环境如何, 这些应用的源码都是可移植的.

ICE 定义了自己的规范语言 Slice, 它用于使对象接口与其实现相分离的基础性抽象机制. 客户和服务器可以用不同的编程语言编写, 可以运行在不同的操作系统和机器架构上, 并且可以使用多 种网络协议进行高效通信.

ICE 客户与服务器内部的逻辑结构如图 4 所示, ICE 核心为远地通信提供了客户端和服务器端运行时支持, ICE 核心是作为客户和服务器可与之链接的库提供的, 客户 ICE 核心、服务器 ICE 核心和对象适配来自于库代码, 代理和骨架由 Slice 定义生成. 客户应用和服务器应用来自于程序员编写的上层应用代码[9].

3 IceGrid 服务架构分析

IceGrid 是 ICE 的核心服务, 提供了若干基本的服务, 可以简化分布式应用的开发和部署[9]. IceGrid 主要由 registry 和 node 组成, registry 主要用来为客户端提供定位服务, 并且管理部署的 node 节点, node 用来装载服务, 根据 registry 的指示对服务进行部署与管理.

定位服务: 保证服务节点的位置透明性. 当客户端需要访问某一个服务时, 不需要了解服务的具体位置,

直接访问 registry, 由 registry 将服务的具体位置告诉客户端, 做到服务端的位置透明.

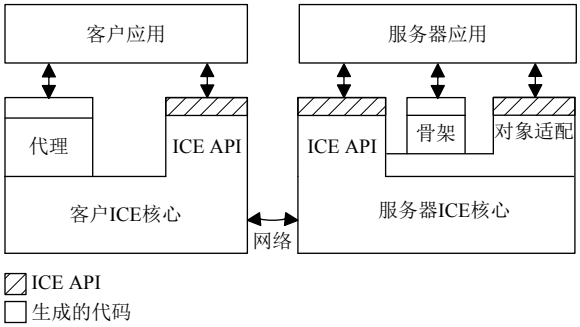

图 4 客户与服务器的结构

服务管理: 管理服务的注册, 启动与关闭. 所有的服务都需要向 registry 进行注册, 当服务需要被访问时, 由 registry 通知相应的 node 对服务进行启动与关闭的管理, 服务的负载均衡也由 registry 进行控制, 整个网络服务的部署也由 registry 进行管理. registry 是整个网络的枢纽.

4 对 IceGrid 提出的改进

IceGrid 作为整个 ICE 网络服务的骨架, 起着至关重要的作用, 所有的分布式网络服务都通过 IceGrid 进行访问, 当整个网络比较庞大时, 单个 registry 已经不能满足网络的负载需求, 当 registry 出现问题时, 整个网络就会瘫痪. 所以 registry 本身支持主从模式的冗余.

首先启动 master. Master 的配置文件与 slave 的配置文件有所不同, master 中必须指定 registry 的名字为 master, 以和其他 slave 进行区分.

远程将部署文件 discriptor 发送给 master. 当 master 收到部署文件后, 会等待 node 节点和 slave 的注册.

启动所有的 slave. 当 slave 启动后, 会主动向 master 进行注册, 注册的过程中, master 会将部署文件发送给所有的 slave, 在 slave 与 master 之间, 实现了一个发布订阅服务, 当 master 对部署文件进行更新时, 会通知所有的 slave 进行部署文件同步.

启动所有的 node 节点. 当 node 节点启动后, 会首先向 master 进行注册, 随后 master 将相应的部署配置文件发给 node, 并将所有 slave 的列表也发给 node, node 通过部署文件启动相应的服务. 然后向所有的 slave 进行注册.

当服务部署完成后, master 和 slave 都可以给客户端提供服务, 都可以对 node 进行管理, 但是只能通过 master 对部署文件进行更改, slave 对部署文件是只读模式.

4.1 registry 主从模式存在的问题

(1) 主从切换不能自动进行

当 master 出现问题时, slave 不能主动进行选举切换到 master, 如果 master 出现问题, slave 虽然可以继续提供服务, 但是部署文件不能进行更新, 一旦需要更新部署文件, 就必须更改某一个 slave 的配置为 master, 使用新的配置重新启动这个 slave, 当整个网络十分庞大复杂时, 会增加运维的成本.

(2) registry 要实现的功能太多

registry 不仅要提供位置服务, 服务管理, 还需要实现发布订阅服务对部署文件进行同步管理. 当部署文件有更新时, master 要通知所有的 slave, 并且将部署文件同步到所有的 slave 中, 当某个 slave 因为网络问题断开连接重新与 master 建立连接时, master 要重新与 slave 同步数据库文件. 当网络负载很大时, registry 要做的事情太多, 将这么多功能放在一起, 很容易造成网络故障, 进而加剧网络环境的进一步恶化.

4.2 通过 Zookeeper 对 registry 主从模式的改进

Zookeeper 作为一款开源多年的分布式协调服务框架, 支持高可用, 本身非常可靠. 通过利用 Zookeeper 的配置管理和集群管理功能, 可以解决 registry 中遇到的问题.

整体的 Zookeeper 目录结构图如图 5, 其中根节点为“/IceGrid”, 下一级包含 2 个目录节点: “/Configuration”, “/Registrys”.

(1) 利用集群管理解决主从切换问题

在 Zookeeper 中, 首先建立持久化目录节点 (PERSISTENT)“/registrys”, 称为父目录, 用于集中化管理所有的 registry 服务器, 每一个 registry 均被创建为父目录下的临时目录节点 (EPHEMERAL), 称为子目录, 并在父目录下注册 Watcher. 当某一个 registry 宕机或因网络问题离开集群时, 其会话 (session) 过期, 对应的临时目录节点被自动删除. 根据 Watcher 机制, 当被监听的父目录节点状态发生变化时, 即父目录节点下的子目录节点被删除, 就会触发 WatchedEvent 事件, 关注该父目录节点的所有 registry 均被通知. 若失去响应的 registry 恰为 master 时, 根据选举机制, Zookeeper 服务器进入恢复模式 (竞选), 触发选举流程, 选出新的 master. 同理, 当父目录节点的子目录节点增加时, 关注该父目录节点的所有 registry 均被通知, 且根据选举机制, Zookeeper 服务器进入广播模式 (同步), 该新加入的 registry 与 master 进行状态同步.

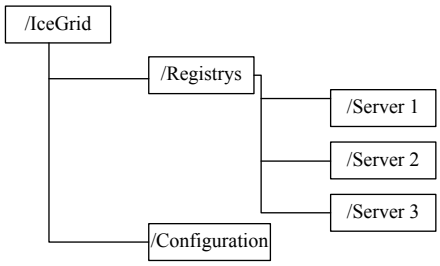

图 5 Zookeeper 目录结构图

因此, 利用 Watcher 机制和选举机制, 可解决 IceGrid 中 registry 的主从切换问题. 当 master 出现问题, 由 Zookeeper 选出新的 master, 当有新的 registry 加入时, 由 Zookeeper 通知 master 与新的 registry 进行状态同步.

(2) 利用配置管理解决部署文件同步问题

在 IceGrid 中, 由于 registry 集群中的各个服务节点, 均采用同一套部署文件; 更新部署文件只能通过 master 进行, 随后 master 需要通知所有的 slave 进行部署文件同步, 同步期间, 所有的 slava 需要与 master 进行读写操作.

依据 Zookeeper 节点特有的数据存储能力, 以及 Watcher 机制, 可以实现部署文件信息的统一管理. 将整个 registry 的部署信息存储在持久化目录节点 (PERSISTENT)“/Configuration”上, 集群中所有 registry 作为 Zookeeper 客户端 , 在该节点上注册 Watcher. 当该节点上的数据发生更改 (即部署信息被管理员修改), 触发设置在该节点上的 Watcher. 所有关注该节点的客户端均被通知, 并根据收到的通知更新自身部署信息, 实现与 master 部署信息的同步. 这样, 每当部署信息被修改, 不需要 master 对 slave 进行通知, 由 Zookeeper 的 watcher 机制进行通知, 部署文件的同步也由 Zookeeper 进行同步, master 只需要专注于提供其他服务就好, 这样也不需要 master 与 slave 进行频繁的读写操作, 降低 master 的读写压力. 所有的 I/O 读写全部由 Zookeeper 集群提供.

5 测试结果

通过在 3 台服务器上部署 3 个 IceRegistry 服务, 分别取名为 r1, r2, r3 这 3 台服务器都向 Zookeeper 集群进行注册, 在首次注册中, r2 为主节点, r1 和 r3 为从节点, 期间断开主节点 r2, 然后通过 Zookeeper 的配置管理对部署文件进行更新, 发现部署信息仍能在余下的 r1 和 r3 中进行同步, 通过 Zookeeper 的 shell 命令查看节点信息, 发现 r1 被选为了新的主节点, IceGrid 部署的服务更新成功.

然后重新开启 r2, 再通过 shell 对 Zookeeper 的文件系统进行查看, r2 又重新向 Zookeeper 进行注册, 并且部署文件也自动同步到了 r2 的数据库中.

通过以上的集群搭建与测试, 说明 IceRegistry 的主从节点可以自动切换, 不再需要人工干预. 并且不再需要主节点来同步部署信息或者发现新的从节点, 这些责任全部由 Zookeeper 集群来做.

6 结论与展望

通过将 Zookeeper 加入到 IceGrid 框架中, 利用 Zookeeper 的集群管理可以解决 registry 不能自动主从切换的问题, 利用 Zookeeper 的配置管理可以解决 registry 之间的部署文件同步问题, 改进后的 IceGrid 将主从切换和部署文件同步等功能剥离到了 Zookeeper 中, 这样不仅解决了 IceGrid 之前存在的主从切换的缺陷问题, 也减轻了 registry 需要承担的任务, 可以更可靠的为 client 提供定位服务.

本文就 ICE 的主从切换方面进行了探索, 提出了一种基于 Zookeeper 的解决方案, 随着中间件技术的不断进步, 还有更多优秀的中间件加入到分布式应用当中来. 因此基于中间件的架构模式会随着广泛应用而不断成熟, 中间件技术会更加适应复杂的网络环境和繁杂的应用场景.

参考文献

1 Tanenbaum AS, Van Steen M. 分布式系统原理与范型. 杨剑锋, 常晓波, 李敏, 译. 北京: 清华大学出版社, 2004.
2 CSDN 博客. 分布式与集群的区别. https://blog.csdn.net/bluishglc/article/details/5483162, 2010-04-13.
3 Bernstein PA. Middleware: A model for distributed system services. Communications of the ACM, 1996, 39(2): 86–98. [doi: 10.1145/230798.230809]
4 黄毅斐. 基于 ZooKeeper 的分布式同步框架设计与实现[硕士学位论文]. 杭州: 浙江大学, 2012.
5 苟丽美, 张锋叶, 林国华. 基于 Zookeeper 的 GIS 集群实现. 计算机工程与设计, 2017, 38(9): 2573–2579.
6 CSDN 博客. Zookeeper 的功能以及工作原理. https://blog.csdn.net/xqb_756148978/article/details/52259381, [2016-08-20].
7 ZooKeepcr 官网. http://zookeeper.apache.org/doc.
8 Henning M. A new approach to object-oriented middleware. IEEE Internet Computing, 2004, 8(1): 66–75.
9 ZeroC, Lnc. Ice 3.6.4 documentation. https://zeroc.com/download/Ice/3.6/Ice-3.6.4.pdf, 2017.