Service Discovery with Apache Curator

Curator的介绍

Curator就是Zookeeper的一个客户端工具（不知道Zookeeper的同学可以到http://www.ibm.com/developerworks/cn/opensource/os-cn-zookeeper/学习下），封装ZooKeeper client与ZooKeeper server之间的连接处理以及zookeeper的常用操作，提供ZooKeeper各种应用场景(recipe, 比如共享锁服务, 集群领导选举机制)的抽象封装。当然还有他看起来非常舒服的Fluent风格的API。 Curator主要从以下几个方面降低了zk使用的复杂性:

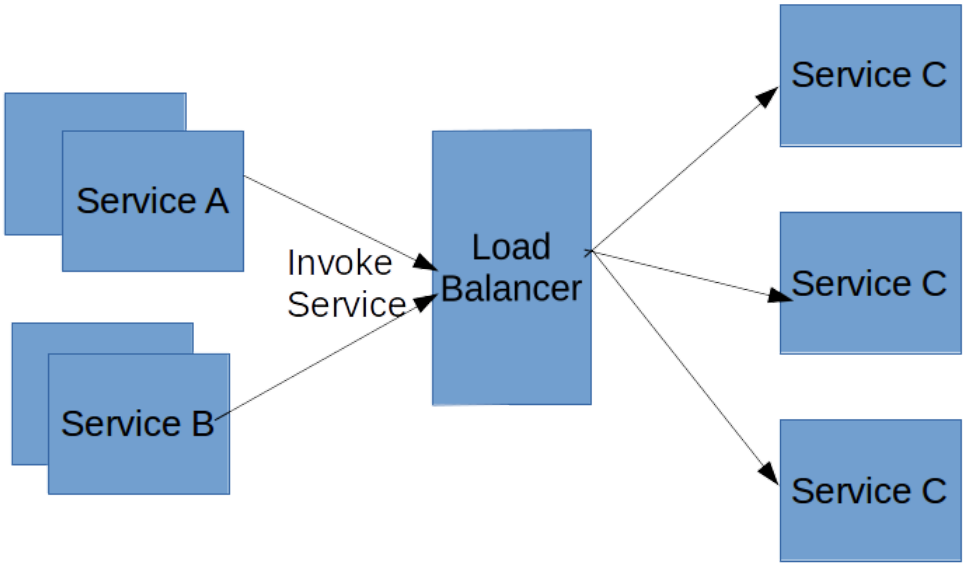
- 重试机制:提供可插拔的重试机制, 它将给捕获所有可恢复的异常配置一个重试策略, 并且内部也提供了几种标准的重试策略(比如指数补偿).
- 连接状态监控: Curator初始化之后会一直的对zk连接进行监听, 一旦发现连接状态发生变化, 将作出相应的处理.
- zk客户端实例管理:Curator对zk客户端到server集群连接进行管理. 并在需要的情况, 重建zk实例, 保证与zk集群的可靠连接
- 各种使用场景支持:Curator实现zk支持的大部分使用场景支持(甚至包括zk自身不支持的场景), 这些实现都遵循了zk的最佳实践, 并考虑了各种极端情况.

Curator通过以上的处理, 让用户专注于自身的业务本身, 而无需花费更多的精力在zk本身.这里我们介绍的是**Curator**的**Service Discovery**模块

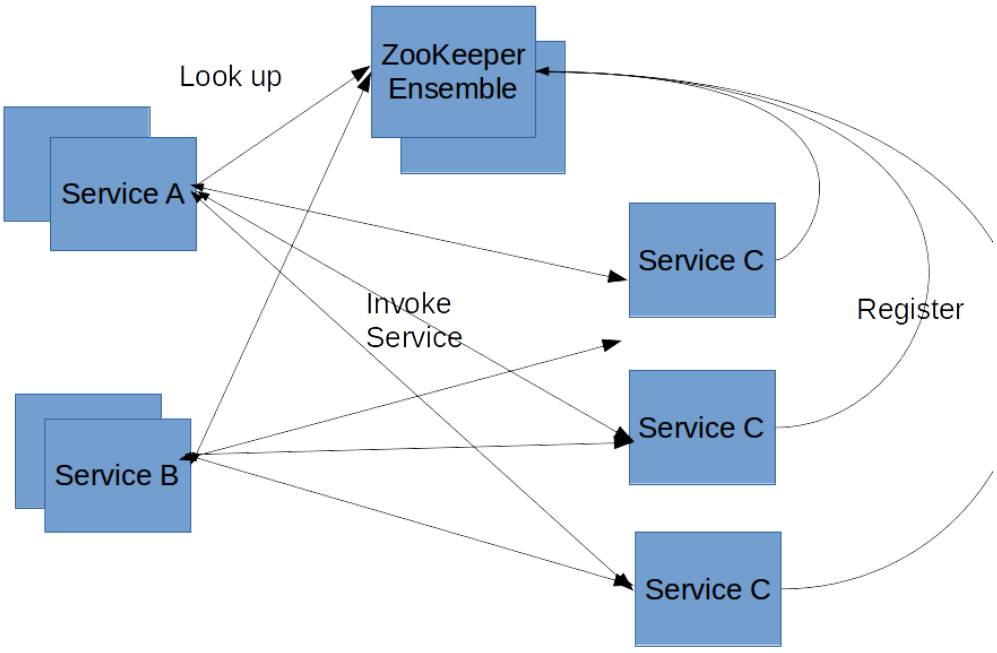
Service Discovery

我们通常在调用服务的时候，需要知道服务的地址，端口，或者其他一些信息，通常情况下，我们是把他们写到程序里面，但是随着服务越来越多，维护起来也越来越费劲，更重要的是，由于地址都是在程序中配置的，我们根本不知道远程的服务是否可用，当我们增加或者删除服务，我们又需要到配置文件中配置么？ 这时候，Zookeeper帮大忙了，我们可以把我们的服务注册到Zookeeper中，创建一个临时节点（当连接断开之后，节点将被删除），存放我们的服务信息（url，ip，port等信息），把这些临时节点都存放在以serviceName命名的节点下面，这样我们要获取某个服务的地址，只需要到Zookeeper中找到这个path，然后就可以读取到里面存放的服务信息，这时候我们就可以根据这些信息调用我们的服务。这样，通过Zookeeper我们就做到了动态的添加和删除服务，做到了一旦一个服务时效，就会自动从Zookeeper中移除，基本上Curator中的Service Discovery就是做的这点事。

下面我们用两张图片来比较一下，一般情况下的服务调用，和使用 Dynamic Service Registry 的区别



使用zookeeper做服务注册之后:



关于Apache curator的service discovery的一些介绍可以参考官方文档：http://curator.apache.org/curator-x-discovery/index.html

Service Discovery 的使用

一般而言，分为 Service Registry 和 Service Discovery，对应服务端和客户端。也就是由服务提供者，讲自身的信息注册到Zookeeper，然后，客户端通过到Zookeeper中查找服务信息，然后根据信息就行调用（见上图）。说了这么多，上代码了。

首先我们定义个payload，我们这一在里面存储一些服务信息。这个信息会被保存在Zookeeper，这里只是举个例子，你还可以写入更多你想要的信息。

```
package discovery;

import org.codehaus.jackson.map.annotate.JsonRootName;



/**
 * Created by hupeng on 2014/9/16.
 */
@JsonRootName("details")
public class InstanceDetails {

    private String id;

    private String listenAddress;

    private int listenPort;

    private String interfaceName;

    public InstanceDetails(String id, String listenAddress, int listenPort,String interfaceName) {
        this.id = id;
        this.listenAddress = listenAddress;
        this.listenPort = listenPort;
        this.interfaceName = interfaceName;
    }

    public InstanceDetails() {
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
```

```
    public String getListenAddress() {
        return listenAddress;
    }

    public void setListenAddress(String listenAddress) {
        this.listenAddress = listenAddress;
    }

    public int getListenPort() {
        return listenPort;
    }

    public void setListenPort(int listenPort) {
        this.listenPort = listenPort;
    }

    public String getInterfaceName() {
        return interfaceName;
    }

    public void setInterfaceName(String interfaceName) {
        this.interfaceName = interfaceName;
    }

    @Override
    public String toString() {
        return "InstanceDetails{" +
                "id='" + id + '\'' +
                ", listenAddress='" + listenAddress + '\'' +
                ", listenPort=" + listenPort +
                ", interfaceName='" + interfaceName + '\'' +
                '}';
    }
}
```

我们先写服务注册，也就是服务端那边做的事情。

```
package discovery;

import org.apache.curator.framework.CuratorFramework;
import org.apache.curator.x.discovery.ServiceDiscovery;
import org.apache.curator.x.discovery.ServiceDiscoveryBuilder;
import org.apache.curator.x.discovery.ServiceInstance;
import org.apache.curator.x.discovery.details.JsonInstanceSerializer;

import java.io.IOException;
/**
 * Created by hupeng on 2014/9/16.
 */
public class ServiceRegistrar{

    private ServiceDiscovery<InstanceDetails> serviceDiscovery;
    private final CuratorFramework client;


    public ServiceRegistrar(CuratorFramework client,String basePath) throws Exception {
        this.client = client;
        JsonInstanceSerializer<InstanceDetails> serializer = new
JsonInstanceSerializer<InstanceDetails>(InstanceDetails.class);
        serviceDiscovery = ServiceDiscoveryBuilder.builder(InstanceDetails.class)
                .client(client)
                .serializer(serializer)
                .basePath(basePath)
                .build();
        serviceDiscovery.start();
    }

    public void registerService(ServiceInstance<InstanceDetails> serviceInstance) throws Exception {
        serviceDiscovery.registerService(serviceInstance);
    }

    public void unregisterService(ServiceInstance<InstanceDetails> serviceInstance) throws Exception
{
        serviceDiscovery.unregisterService(serviceInstance);

    }
```

```
    public void updateService(ServiceInstance<InstanceDetails> serviceInstance) throws Exception {
        serviceDiscovery.updateService(serviceInstance);

    }

    public void close() throws IOException {
        serviceDiscovery.close();
    }
}
```

一般情况下，会在我们服务启动的时候就将服务信息注册，比如我们是web项目的话可以写一个Servlet Listener进行注册，这里为了方便，写一个Main方法进行测试，如果我们把我们的信息存储在payload中的话，UriSpec是可以不定义的。

```
package discovery;

import org.apache.curator.framework.CuratorFramework;
import org.apache.curator.framework.CuratorFrameworkFactory;
import org.apache.curator.retry.ExponentialBackoffRetry;
import org.apache.curator.x.discovery.ServiceInstance;
import org.apache.curator.x.discovery.UriSpec;


import java.util.UUID;

/**
 * User: hupeng
 * Date: 14-9-16
 * Time: 下午8:05
 */
public class ServerApp {

    public static void main(String[] args) throws Exception {
        CuratorFramework client = CuratorFrameworkFactory.newClient("127.0.0.1:2181", new
ExponentialBackoffRetry(1000, 3));
        client.start();
        ServiceRegistrar serviceRegistrar = new ServiceRegistrar(client,"services");
        ServiceInstance<InstanceDetails> instance1 = ServiceInstance.<InstanceDetails>builder()
                .name("service1")
                .port(12345)
                .address("192.168.1.100")    //address不写的话，会取本地ip
                .payload(new
InstanceDetails(UUID.randomUUID().toString(),"192.168.1.100",12345,"Test.Service1"))
                .uriSpec(new UriSpec("{scheme}://{address}:{port}"))
                .build();
        ServiceInstance<InstanceDetails> instance2 = ServiceInstance.<InstanceDetails>builder()
                .name("service2")
                .port(12345)
                .address("192.168.1.100")
                .payload(new
InstanceDetails(UUID.randomUUID().toString(),"192.168.1.100",12345,"Test.Service2"))
                .uriSpec(new UriSpec("{scheme}://{address}:{port}"))
                .build();
        serviceRegistrar.registerService(instance1);
        serviceRegistrar.registerService(instance2);


        Thread.sleep(Integer.MAX_VALUE);
    }
}
```

再来写Service discovery

```
package discovery;

import com.google.common.collect.Lists;
import com.google.common.collect.Maps;
import org.apache.curator.framework.CuratorFramework;
import org.apache.curator.utils.CloseableUtils;
import org.apache.curator.x.discovery.ServiceDiscovery;
import org.apache.curator.x.discovery.ServiceDiscoveryBuilder;
import org.apache.curator.x.discovery.ServiceInstance;
import org.apache.curator.x.discovery.ServiceProvider;
import org.apache.curator.x.discovery.details.JsonInstanceSerializer;
```

```
import org.apache.curator.x.discovery.strategies.RandomStrategy;

import java.io.Closeable;
import java.io.IOException;
import java.util.List;
import java.util.Map;
/**
 * Created by hupeng on 2014/9/16.
 */
public class ServiceDiscoverer {
    private ServiceDiscovery<InstanceDetails> serviceDiscovery;
    private Map<String, ServiceProvider<InstanceDetails>> providers = Maps.newHashMap();
    private List<Closeable> closeableList = Lists.newArrayList();
    private Object lock = new Object();


    public ServiceDiscoverer(CuratorFramework client ,String basePath) throws Exception {
        JsonInstanceSerializer<InstanceDetails> serializer = new
JsonInstanceSerializer<InstanceDetails>(InstanceDetails.class);
        serviceDiscovery = ServiceDiscoveryBuilder.builder(InstanceDetails.class)
                .client(client)
                .basePath(basePath)
                .serializer(serializer)
                .build();

        serviceDiscovery.start();
    }


    public ServiceInstance<InstanceDetails> getInstanceByName(String serviceName) throws Exception {
        ServiceProvider<InstanceDetails> provider = providers.get(serviceName);
        if (provider == null) {
            synchronized (lock) {
                provider = providers.get(serviceName);
                if (provider == null) {
                    provider = serviceDiscovery.serviceProviderBuilder().
                            serviceName(serviceName).
                            providerStrategy(new RandomStrategy<InstanceDetails>())
                            .build();
                    provider.start();
                    closeableList.add(provider);
                    providers.put(serviceName, provider);
                }
            }
        }


        return provider.getInstance();
    }


    public synchronized void close(){
       for (Closeable closeable : closeableList) {
           CloseableUtils.closeQuietly(closeable);
       }
    }


}
```

客户端测试程序：

```
package discovery;

import org.apache.curator.framework.CuratorFramework;
import org.apache.curator.framework.CuratorFrameworkFactory;
import org.apache.curator.retry.ExponentialBackoffRetry;
import org.apache.curator.utils.CloseableUtils;
import org.apache.curator.x.discovery.ServiceInstance;

/**
 * User: hupeng
 * Date: 14-9-16
 * Time: 下午8:16
 */
public class ClientApp {
```

```
    public static void main(String[] args) throws Exception {
        CuratorFramework client = CuratorFrameworkFactory.newClient("127.0.0.1:2181", new
ExponentialBackoffRetry(1000, 3));
        client.start();
        ServiceDiscoverer serviceDiscoverer = new ServiceDiscoverer(client,"services");

        ServiceInstance<InstanceDetails> instance1 =
serviceDiscoverer.getInstanceByName("service1");

        System.out.println(instance1.buildUriSpec());
        System.out.println(instance1.getPayload());

        ServiceInstance<InstanceDetails> instance2 =
serviceDiscoverer.getInstanceByName("service1");

        System.out.println(instance2.buildUriSpec());
        System.out.println(instance2.getPayload());

        serviceDiscoverer.close();
        CloseableUtils.closeQuietly(client);
    }
}
```

好了，代码就到这里，如果有什么问题的话，请指正。