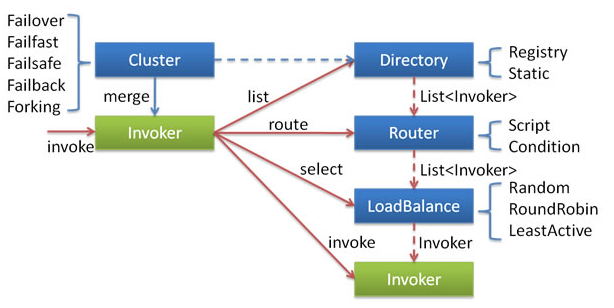
# **集群容错**：

在集群调用失败时，Dubbo 提供了多种容错方案，缺省为 failover 重试。

各节点关系：

* 这里的Invoker是Provider的一个可调用Service的抽象，Invoker封装了Provider地址及Service接口信息
* Directory代表多个Invoker，可以把它看成List<Invoker>，但与List不同的是，它的值可能是动态变化的，比如注册中心推送变更
* Cluster将Directory中的多个Invoker伪装成一个Invoker，对上层透明，伪装过程包含了容错逻辑，调用失败后，重试另一个
* Router负责从多个Invoker中按路由规则选出子集，比如读写分离，应用隔离等
* LoadBalance负责从多个Invoker中选出具体的一个用于本次调用，选的过程包含了负载均衡算法，调用失败后，需要重选

## **集群容错模式**

### **Failover Cluster(**默认配置**)**

失败自动切换，当出现失败，重试其它服务器。通常用于读操作，但重试会带来更长延迟。可通过retries="2"来设置重试次数(不含第一次)。

### **Failfast Cluster**

快速失败，只发起一次调用，失败立即报错。通常用于非幂等性的写操作，比如新增记录。

### **Failsafe Cluster**

失败安全，出现异常时，直接忽略。通常用于写入审计日志等操作。

### **Failback Cluster**

失败自动恢复，后台记录失败请求，定时重发。通常用于消息通知操作。

### **Forking Cluster**

并行调用多个服务器，只要一个成功即返回。通常用于实时性要求较高的读操作，但需要浪费更多服务资源。可通过forks="2"来设置最大并行数。

### **Broadcast Cluster**

广播调用所有提供者，逐个调用，任意一台报错则报错。通常用于通知所有提供者更新缓存或日志等本地资源信息。

除dubbo提供的策略外，也可以自行扩展集群容错策略。

# **负载均衡(****LoadBalance)：**

## **负载均衡策略**

### **Random LoadBalance**

* **随机**，按权重设置随机概率。
* 在一个截面上碰撞的概率高，但调用量越大分布越均匀，而且按概率使用权重后也比较均匀，有利于动态调整提供者权重。

### **RoundRobin LoadBalance**

* **轮询**，按公约后的权重设置轮询比率。
* 存在慢的提供者累积请求的问题，比如：第二台机器很慢，但没挂，当请求调到第二台时就卡在那，久而久之，所有请求都卡在调到第二台上。

### **LeastActive LoadBalance**

* **最少活跃调用数**，相同活跃数的随机，活跃数指调用前后计数差。
* 使慢的提供者收到更少请求，因为越慢的提供者的调用前后计数差会越大。

### **ConsistentHash LoadBalance**

* **一致性 Hash**，相同参数的请求总是发到同一提供者。
* 当某一台提供者挂时，原本发往该提供者的请求，基于虚拟节点，平摊到其它提供者，不会引起剧烈变动。
* 算法参见：http://en.wikipedia.org/wiki/Consistent\_hashing
* 缺省只对第一个参数 Hash，如果要修改，请配置 <dubbo:parameter key="hash.arguments" value="0,1" />
* 缺省用 160 份虚拟节点，如果要修改，请配置 <dubbo:parameter key="hash.nodes" value="320" />

都可以通过**@Service**、**@Reference**注解参数配置

**只订阅不注册：**

为方便开发测试，经常会在线下共用一个所有服务可用的注册中心，这时，如果一个正在开发中的服务提供者注册，可能会影响消费者不能正常运行。

可以让服务提供者开发方，只订阅服务(开发的服务可能依赖其它服务)，而不注册正在开发的服务，通过直连测试正在开发的服务。

<dubbo:registry address="10.20.153.10:9090?register=false" />

