分布式缓存

原文地址为http://ehcache.sourceforge.net/documentation/distributed_caching.html
译者：中华风筝
联系方式：kite.china@gmail.com（也是GTalk帐号）

从 1.2 版本开始，Ehcache 可以使用分布式的缓存了。

分布式这个特性是以 plugin 的方式实现的。Ehcache 自带了一些默认的分布式缓存插件实现，这些插件可以满足大部分应用的需要。如果需要使用其他的插件那就需要自己开发了，开发者可以通过查看 distribution 包里的源代码及 JavaDoc 来实现它。

尽管不是必须的，在使用分布式缓存时理解一些ehcahce的设计思想也是有帮助的。这可以参看分布式缓存设计的页面。

以下的部分将展示如何让分布式插件同 ehcache 一起工作。

下面列出的是一些分布式缓存中比较重要的方面：

- 你如何知道集群环境中的其他缓存？
- 分布式传送的消息是什么形式？
- 什么情况需要进行复制？增加（Puts），更新（Updates）或是失效（Expiries）？
- 采用什么方式进行复制？同步还是异步方式？

为了安装分布式缓存，你需要配置一个 PeerProvider、一个 CacheManagerPeerListener，它们对于一个 CacheManager 来说是全局的。每个进行分布式操作的 cache 都要添加一个 cacheEventListener 来传送消息。

正确的元素类型

只有可序列化的元素可以进行复制。

一些操作，比如移除，只需要元素的键值而不用整个元素；在这样的操作中即使元素不是可序列化的但键值是可序列化的也可以被复制，

成员发现（Peer Discovery）

Ehcache 进行集群的时候有一个 cache 组的概念。每个 cache 都是其他 cache 的一个 peer，没有主 cache 的存在。刚才我们问了一个问题：你如何知道集群环境中的其他缓存？这个问题可以命名为成员发现（Peer Discovery）。

Ehcache 提供了两种机制用来进行成员发现，就像一辆汽车：手动档和自动档。

要使用一个内置的成员发现机制要在 ehcache 的配置文件中指定 cacheManagerPeerProviderFactory 元素的 class 属性为 net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory。

自动的成员发现

自动的发现方式用 TCP 广播机制来确定和维持一个广播组。它只需要一个简单的配置可以自动的在组中添加和移除成员。在集群中也不需要什么优化服务器的知识，这是默认推荐的。

成员每秒向群组发送一个“心跳”。如果一个成员 5 秒种都没有发出信号它将被群组移除。如果一个新的成员发送了一个“心跳”它将被添加进群组。

任何一个用这个配置安装了复制功能的 cache 都将被其他的成员发现并标识为可用状态。

要设置自动的成员发现，需要指定 ehcache 配置文件中 cacheManagerPeerProviderFactory 元素的 properties 属性，就像下面这样：

```
peerDiscovery=automatic multicastGroupAddress=multicast address | multicast host name multicastGroupPort=port timeToLive=0-255 (timeToLive 属性详见常见问题部分的描述)
```

示例

假设你在集群中有两台服务器。你希望同步 sampleCache1 和 sampleCache2。每台独立的服务器都要有这样的配置：

配置 server1 和 server2

```
<cacheManagerPeerProviderFactory
 class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
 properties="peerDiscovery=automatic, multicastGroupAddress=230.0.0.1,

 multicastGroupPort=4446, timeToLive=32"/>
```

手动进行成员发现

进行手动成员配置要知道每个监听器的 IP 地址和端口。成员不能在运行时动态地添加和移除。在技术上很难使用广播的情况下就可以手动成员发现，例如在集群的服务器之间有一个不能传送广播报文的路由器。你也可以用手动成员发现进行单向的数据复制，只让 server2 知道 server1 而 server1 不知道 server2。

配置手动成员发现，需要指定 ehcache 配置文件中 cacheManagerPeerProviderFactory 的 properties 属性，像下面这样：

```
peerDiscovery=manual rmiUrls=//server:port/cacheName, ...
```

rmiUrls 配置的是服务器 cache peers 的列表。注意不要重复配置。

示例

假设你在集群中有两台服务器。你要同步 sampleCache1 和 sampleCache2。下面是每个服务器需要的配置：

配置 server1

```
<cacheManagerPeerProviderFactory
 class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
 properties="peerDiscovery=manual,
 rmiUrls=//server2:40001/sampleCache11|//server2:40001/sampleCache12"/>
```

配置 server2

```
<cacheManagerPeerProviderFactory
 class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
 properties="peerDiscovery=manual,
 rmiUrls=//server1:40001/sampleCache11|//server1:40001/sampleCache12"/>
```

配置 CacheManagerPeerListener

每个 CacheManagerPeerListener 监听从成员们发向当前 CacheManager 的消息。

配置 CacheManagerPeerListener 需要指定一个 CacheManagerPeerListenerFactory，它以插件的机制实现，用来创建 CacheManagerPeerListener。

cacheManagerPeerListenerFactory 的属性有：

class – 一个完整的工厂类名。

properties – 只对这个工厂有意义的属性，使用逗吃分隔。

Ehcache 有一个内置的基于 RMI 的分布系统。它的监听器是 RMICacheManagerPeerListener，这个监听器可以用 RMICacheManagerPeerListenerFactory 来配置。

```
<cacheManagerPeerListenerFactory
class="net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory"
properties="hostName=localhost, port=40001,
socketTimeoutMillis=2000"/>
```

有效的属性是：

hostname (可选) – 运行监听器的服务器名称。标明了做为集群群组的成员的地址，同时也是你想要控制的从集群中接收消息的接口。

在 CacheManager 初始化的时候会检查 hostname 是否可用。

如果 hostName 不可用，CacheManager 将拒绝启动并抛出一个连接被拒绝的异常。

如果指定，hostname 将使用 InetAddress.getLocalHost().getHostAddress()来得到。

警告：不要将 localhost 配置为本地地址 127.0.0.1，因为它在网络中不可见将会导致不能从远程服务器接收信息从而不能复制。在同一台机器上有多个 CacheManager 的时候，你应该只用 localhost 来配置。

port – 监听器监听的端口。

socketTimeoutMillis (可选) – Socket 超时的时间。默认是 2000ms。

配置 CacheReplicators

每个要进行同步的 cache 都需要设置一个用来向 CacheManagerr 的成员复制消息的缓存事件监听器。这个工作要通过为每个 cache 的配置增加一个 cacheEventListenerFactory 元素来完成。

```
<!-- Sample cache named sampleCache2. -->
    <cache name="sampleCache2"
           maxElementsInMemory="10"
           eternal="false"
           timeToIdleSeconds="100"
           timeToLiveSeconds="100"
           overflowToDisk="false">
        <cacheEventListenerFactory
class="net.sf.ehcache.distribution.RMICacheReplicatorFactory"
                                   properties="replicateAsynchronously=true,
```

```
replicatePuts=true, replicateUpdates=true, replicateUpdatesViaCopy=false,
replicateRemovals=true "/>
</cache>
```

class – 使用 net.sf.ehcache.distribution.RMICacheReplicatorFactory

这个工厂支持以下属性：

replicatePuts=true | false – 当一个新元素增加到缓存中的时候是否要复制到其他的 peers. 默认是 true。

replicateUpdates=true | false – 当一个已经在缓存中存在的元素被覆盖时是否要进行复制。默认是 true。

replicateRemovals= true | false – 当元素移除的时候是否进行复制。默认是 true。

replicateAsynchronously=true | false – 复制方式是异步的（指定为 true 时）还是同步的（指定为 false 时）。默认是 true。

replicateUpdatesViaCopy=true | false – 当一个元素被拷贝到其他的 cache 中时是否进行复制（指定为 true 时为复制），默认是 true。

你可以使用 ehcache 的默认行为从而减少配置的工作量，默认的行为是以异步的方式复制每件事；你可以像下面的例子一样减少 RMICacheReplicatorFactory 的属性配置：

```
<!-- Sample cache named sampleCache4. All missing RMICacheReplicatorFactory properties
default to true -->
    <cache name="sampleCache4"
           maxElementsInMemory="10"
           eternal="true"
           overflowToDisk="false"
           memoryStoreEvictionPolicy="LFU">
        <cacheEventListenerFactory
class="net.sf.ehcache.distribution.RMICacheReplicatorFactory"/>
</cache>
```

常见的问题

Windows 上的 Tomcat

有一个Tomcat或者是JDK的bug，在tomcat启动时如果tomcat的安装路径中有空格的话，在启动时RMI监听器会失败。参见 http://archives.java.sun.com/cgi-bin/wa?A2=ind0205&L=rmi-users&P=797 和 http://www.ontotext.com/kim/doc/sys-doc/faq-howto-bugs/known-bugs.html。

由于在 Windows 上安装 Tomcat 默认是装在“Program Files”文件夹里的，所以这个问题经常发生。

广播阻断

自动的 peer discovery 与广播息息相关。广播可能被路由阻拦，像 Xen 和 VMWare 这种虚拟化的技术也可以阻拦广播。如果这些都打开了，你可能还在要将你的网卡的相关配置打开。

一个简单的办法可以告诉广播是否有效，那就是使用 ehcache remote debugger 来看“心跳”是否可用。

广播传播的不够远或是传得太远

你可以通过设置 badly misnamed time to live 来控制广播传播的距离。用广播 IP 协议时，timeToLive 的值指的是数据包可以传递的域或是范围。约定如下：

- 0 是限制在同一个服务器
- 1 是限制在同一个子网
- 32 是限制在同一个网站
- 64 是限制在同一个 region
- 128 是限制在同一个大洲
- 255 是不限制

译者按：上面这些资料翻译的不够准确，请读者自行寻找原文理解吧。

在 Java 实现中默认值是 1，也就是在同一个子网中传播。改变 timeToLive 属性可以限制或是扩展传播的范围。