(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102333108 A
(43)申请公布日 2012.01.25

(21)申请号 201110066128.4

(22)申请日 2011.03.18

(71)申请人 北京神州数码思特奇信息技术股份有限公司
地址 100085 北京市海淀区上地九街 9 号数码科技广场二层

(72)发明人 王涛

(74)专利代理机构 北京轻创知识产权代理有限公司 11212
代理人 杨立

(51)Int.Cl.
H04L 29/08(2006.01)

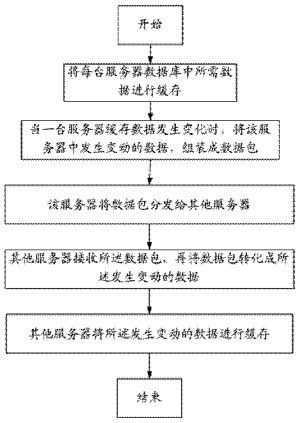
权利要求书 1 页 说明书 6 页 附图 2 页

(54)发明名称

分布式缓存同步系统及方法

(57)摘要

本发明涉及一种分布式系统缓存同步系统，它包括多台服务器，所述每台服务器包括：缓存单元，将数据库中所需数据进行缓存；监听单元，监听作为接收端的其他服务器的请求或作为发送端的服务器的操作；转化单元，将服务器中发生变动的缓存数据转化成数据包，或将数据包转化成发生变动的数据；传输单元，将所述数据包传输至其他服务器。本发明还涉及一种分布式系统缓存同步方法。降低对数据库访问次数；读取缓存的速度大大高于读取文件和数据库，大大提高读取速度。分布式的支持，可以使大型高并发系统采用缓存成为可能。



CN 102333108 A

1. 一种分布式系统缓存同步的方法，其特征在于：它包括以下步骤：

步骤 1：将每台服务器数据库中所需数据进行缓存；

步骤 2：当一台服务器缓存数据发生变化时，将该服务器中发生变动的数据，组装成数据包；

步骤 3：该服务器将数据包分发给其他服务器；

步骤 4：其他服务器接收所述数据包，再将数据包转化成所述发生变动的数据；

步骤 5：其他服务器将所述发生变动的数据进行缓存。

2. 根据权利要求 1 所述的分布式系统缓存同步的方法，其特征在于：所述步骤 1 包括以下子步骤：

步骤 1.1：在每台服务器上构造用于在内存中存储的域，即将要进行缓存的数据库数据按需求划分成值对象，以值对象的包名加类名组成缓存中的唯一域；

步骤 1.2：将数据库中的记录读出，并将其构造成实体类，且数据库中的每一条记录对应成一个实体类；

步骤 1.3：将实体类以 Map 的键值对方式存放入域中，其中 Key 为主键，Value 值为对应的实体类。

3. 根据权利要求 1 或 2 所述的分布式系统缓存同步的方法，其特征在于：所述步骤 2 中每台服务器配置了 Listener，当一台服务器缓存数据发生变化时，Listener 同时分发请求到其他服务器。

4. 根据权利要求 1 或 2 所述的分布式系统缓存同步的方法，其特征在于：所述步骤 2 中服务器通过反射机制将发生变动的数据转化成 Map 结构的数据包。

5. 根据权利要求 1 或 2 所述的分布式系统缓存同步的方法，其特征在于：所述步骤 3 为该服务器通过 xmlrpc 方式，将数据包分发给其他服务器。

6. 一种分布式系统缓存同步系统，其特征在于：它包括多台服务器，所述每台服务器包括：

缓存单元，将数据库中所需数据进行缓存；

监听单元，监听作为接收端的其他服务器的请求或作为发送端的服务器的操作；

转化单元，将服务器中发生变动的缓存数据转化成数据包，或将数据包转化成发生变动的数据；

传输单元，将所述数据包传输至其他服务器。

7. 根据权利要求 6 所述的一种分布式系统缓存同步系统，其特征在于：它还包括应急加载缓存接口，当某台服务器当机时，所述应急加载缓存接口重新加载缓存数据，同时保证与其他服务器的正常通信。

分布式缓存同步系统及方法

技术领域

[0001] 本发明涉及一种分布式缓存同步系统及方法。

背景技术

[0002] 随着电信业务的迅猛发展，为电信行业公司的 IT 系统带来巨大压力。数据读取效率问题就突出出来。如 BOSS（Business & Operation Support System，业务运营支撑系统）系统响应慢、系统不可用，对客户的满意度及应用中的问题无法及时获知和改进等。

[0003] 所以研发出可以高效配置，提高访问数据的速度，降低系统消耗的方法也就成了一种必然需要。

发明内容

[0004] 针对以上所述的数据读取的效率问题，本发明提供一种分布式缓存同步系统及方法。

[0005] 本发明解决上述技术问题的技术方案如下：一种分布式系统缓存同步的方法，它包括以下步骤：

步骤 1：将每台服务器数据库中所需数据进行缓存；

步骤 2：当一台服务器缓存数据发生变化时，将该服务器中发生变动的数据，组装成数据包；

步骤 3：该服务器将数据包分发给其他服务器；

步骤 4：其他服务器接收所述数据包，再将数据包转化成所述发生变动的数据；

步骤 5：其他服务器将所述发生变动的数据进行缓存。

[0006] 进一步的，步骤 1 包括以下子步骤：

步骤 1.1：在每台服务器上构造用于在内存中存储的域，即将要进行缓存的数据库数据按需求划分成值对象，以值对象的包名加类名组成缓存中的唯一域；

步骤 1.2：将数据库中的记录读出，并将其构造成实体类，且数据库中的每一条记录对应成一个实体类；

步骤 1.3：将实体类以 Map 的键值对方式存放入域中，其中 Key 为主键，Value 值为对应的实体类。

[0007] 进一步的，步骤 2 中每台服务器配置了 Listener（Listener 是 Servlet（Servlet 是一种服务器端的 Java 应用程序，具有独立于平台和协议的特性，可以生成动态的 Web 页面。）的监听器，它可以监听客户端的请求、服务端的操作等），当一台服务器缓存数据发生变化时，Listener 同时分发请求到其他服务器。

[0008] 进一步的，步骤 2 中服务器通过反射机制将发生变动的数据转化成 Map 结构的数据包。

[0009] 进一步的，步骤 3 为该服务器通过 xmlrpc 方式，将数据包分发给其他服务器。

[0010] 一种分布式系统缓存同步系统，它包括多台服务器，所述每台服务器包括：

缓存单元，将数据库中所需数据进行缓存；

监听单元，监听作为接收端的其他服务器的请求或作为发送端的服务器的操作；

转化单元，将服务器中发生变动的缓存数据转化成数据包，或将数据包转化成发生变动的数据；

传输单元，将所述数据包传输至其他服务器。

[0011] 进一步的，它还包括应急加载缓存接口，当某台服务器当机时，所述应急加载缓存接口重新加载缓存数据，同时保证与其他服务器的正常通信。

[0012] 本发明的有益效果是：

1. 降低对数据库访问次数，一次读取数据库存入缓存，可以接受无数次访问，接受数据库资源。

[0013] 2. 读取缓存的速度大大高于读取文件和数据库，大大提高读取速度。

[0014] 3. 分布式的支持，可以使大型高并发系统采用缓存成为可能。

附图说明

[0015] 图 1 为本发明分布式缓存同步方法的流程图；

图 2 为本发明分布式缓存同步系统的结构框图。

具体实施方式

[0016] 以下结合附图对本发明的原理和特征进行描述，所举实例只用于解释本发明，并非用于限定本发明的范围。

[0017] 如图 1 所示，一种分布式系统缓存同步的方法，它包括以下步骤：

步骤 1：将每台服务器数据库中所需数据进行缓存；

步骤 2：当一台服务器缓存数据发生变化时，将该服务器中发生变动的数据，组装成数据包；

步骤 3：该服务器将数据包分发给其他服务器；

步骤 4：其他服务器接收所述数据包，再将数据包转化成所述发生变动的数据；

步骤 5：其他服务器将所述发生变动的数据进行缓存。

[0018] 步骤 1 包括以下子步骤：

步骤 1.1：在每台服务器上构造用于在内存中存储的域，即将要进行缓存的数据库数据按需求划分成值对象，以值对象的包名加类名组成缓存中的唯一域；

步骤 1.2：将数据库中的记录读出，并将其构造成实体类，且数据库中的每一条记录对应成一个实体类；

步骤 1.3：将实体类以 Map 的键值对方式存放入域中，其中 Key 为主键，Value 值为对应的实体类。

[0019] 步骤 2 为每台服务器配置了 Listener，当一台服务器缓存数据发生变化时，Listener 同时分发请求到其他服务器，并且该服务器通过反射机制（反射机制，主要是指程序可以访问、检测和修改它本身状态或行为的一种能力）将发生变动的数据转化成 Map（Map，将键映射到值的对象）结构的数据包。

[0020] 步骤 3 为该服务器通过 xmlrpc（xmlrpc 是使用 http 协议做为传输协议的 rpc 机

制，使用 xml 文本的方式传输命令和数据）方式，将数据包分发给其他服务器。

[0021] 如图 2 所示，一种分布式系统缓存同步系统，它包括多台服务器 1，所述每台服务器 1 包括：

缓存单元 3，将数据库中所需数据进行缓存；

监听单元 6，监听作为接收端的其他服务器的请求或作为发送端的服务器的操作；

转化单元 4，将服务器中发生变动的缓存数据转化成数据包，或将数据包转化成发生变动的数据；

传输单元 2，将所述数据包传输至其他服务器。

[0022] 它还包括应急加载缓存接口 5，当某台服务器当机时，所述应急加载缓存接口 5 重新加载缓存数据，同时保证与其他服务器的正常通信。可以修正所有缓存数据同步问题。更能保证数据的准确性。

[0023] 每台服务器 1 都可以做发送端和接受端。这种方式就极大的加强的系统的安全稳定性，即当有某些服务器出现当机等问题时。不影响其他服务器 1 的正常运行。

[0024] 采用 java 语言编写的程序具体如下：

1. 以下是利用反射机制将 object（实体类）和 Map 互转的实现：

```
/*
 * 将 Map 转换成类
 */
public static void map2Object(Object obj, Map map)
      throws IllegalArgumentException, IllegalAccessException {
   Field[] fields = obj.getClass().getDeclaredFields();
   for (int j = 0; j < fields.length; j++) {
fields[j].setAccessible(true);// 设置这个变量不进行访问权限检查 在类里设置的变量可以为 private
      fields[j].set(obj, map.get(fields[j].getName()));
   }
}
/*
 * 将类转换成 Map
 */
public static void object2Map(Object obj, Map map)
      throws IllegalArgumentException, IllegalAccessException {
   Field[] fields = obj.getClass().getDeclaredFields();
   for (int j = 0; j < fields.length; j++) {
      fields[j].setAccessible(true);// 设置这个变量不进行访问权限检查在类里设置的变量可以为 private
      map.put(fields[j].getName(), fields[j].get(obj));
   }
}
```

```
/*
   根据类名得到实例
*/
public static Object getClassByName(String className) throws Exception
{
    Class c;
    Object o;
    c = Class.forName(className);
    o = (c.getClassLoader().loadClass(className)).newInstance();
    return o;
}
```

2. 以下是发送端遍历发送数据的实现：

```
/*
   入口 遍历发送给集群中每个成员。
[0025]    */
public boolean distributData(List list,String className){
   boolean flag = true;
   try{
if(isDistribute!=null && isDistribute.equals("on")){// 分布式缓存开关。
[0026]          convertDataforRpc(list,className);
        for (int i = 0; i < host.length; i++) {
logger.info(host[i] + ":" + port[i] + "/" + server[i]+ " are in updata.....");
          if (!sendData(host[i], port[i], server[i],dataList)) {
logger.error("IndexServerError error:"+ host[i]+ ":"+ port[i]+ "/"+ server[i]+ " faild !");
             flag = false;
          }
        }
   }
   }catch(Exception e){
     logger.error("error: in PromptDistributer.distributData()",e);
     return false;
   }
   return flag;
}
```

3 以下是接受数据的实现

```
/*
   接受 rpc 传过来的数据包 并解析 再同步缓存 sava
*/
public Boolean receiveUpdateData(List rpcData) {
```

```
        try {
          String className = (String)rpcData.get(0);
    logger.info("*****seccess received bossDataDistrabutData***classname="+className);
          Map rpcDataMap = new HashMap();
          List rpcRelultList = new ArrayList();
          // 更新缓存
          SychronizedBossInfo se = new SychronizedBossInfo();
          //**************************
          Object o = DistrabutTool.getClassByName(className);
    for (int i = 1; i < rpcData.size(); i++) {//从1开始。0的为是是className
              o = DistrabutTool.getClassByName(className);
              rpcDataMap = (HashMap) rpcData.get(i);
              DistrabutTool.map2Object(o,rpcDataMap);
              rpcRelultList.add(o);;
          }
          //**************************
      if(className.equals("com.sitech.kms.ehcache.pojo.UserInfo")){// 用户
              se.updataUserByCach(rpcRelultList);
      }else if(className.equals("com.sitech.kms.ehcache.pojo.Dept")){// 部门
              se.updataDeptByCach(rpcRelultList);
      }else if(className.equals("com.sitech.kms.ehcache.pojo.UserRole")){// 用户角色
              se.updataUserRoleByCach(rpcRelultList);
    }else if(className.equals("com.sitech.kms.ehcache.pojo.RoleMenu")){//角色菜单
              se.updataRoleMenuByCach(rpcRelultList);
      }else if(className.equals("com.sitech.kms.ehcache.pojo.RoleModule")){//角色 功 能 模块
              se.updataRoleModuleByCach(rpcRelultList);
          }
        } catch (Exception e) {
          e.printStackTrace();
    logger.error("IndexServerError in PromptDistributReceiver.receiveUpdateData",e);
          return new Boolean("false") ;
        }
        return new Boolean("true");
      }
```

现在行业内查询数据库的数据时，基本上都是直接用 sql 语句查询。每次查询都会对数据库造成压力和资源消耗。采用本发明后可以一次读取数据库后，无限次接受请求。并且本发明读取的是内存数据，所以速度比读取数据库快很多。

[0027] 适用于比较少更新表数据。一般要使用在比较少执行 write（将数据写入已打开的文件内）操作的表（包括 update（更新表中原有数据），insert（插入），delete（删除）

等）

本发明阐述了通过将多种数据类型存入缓存。以提高对数据访问的效率，适用于有数据访问时效率问题的一切系统。

[0028] 以上所述仅为本发明的较佳实施例，并不用以限制本发明，凡在本发明的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

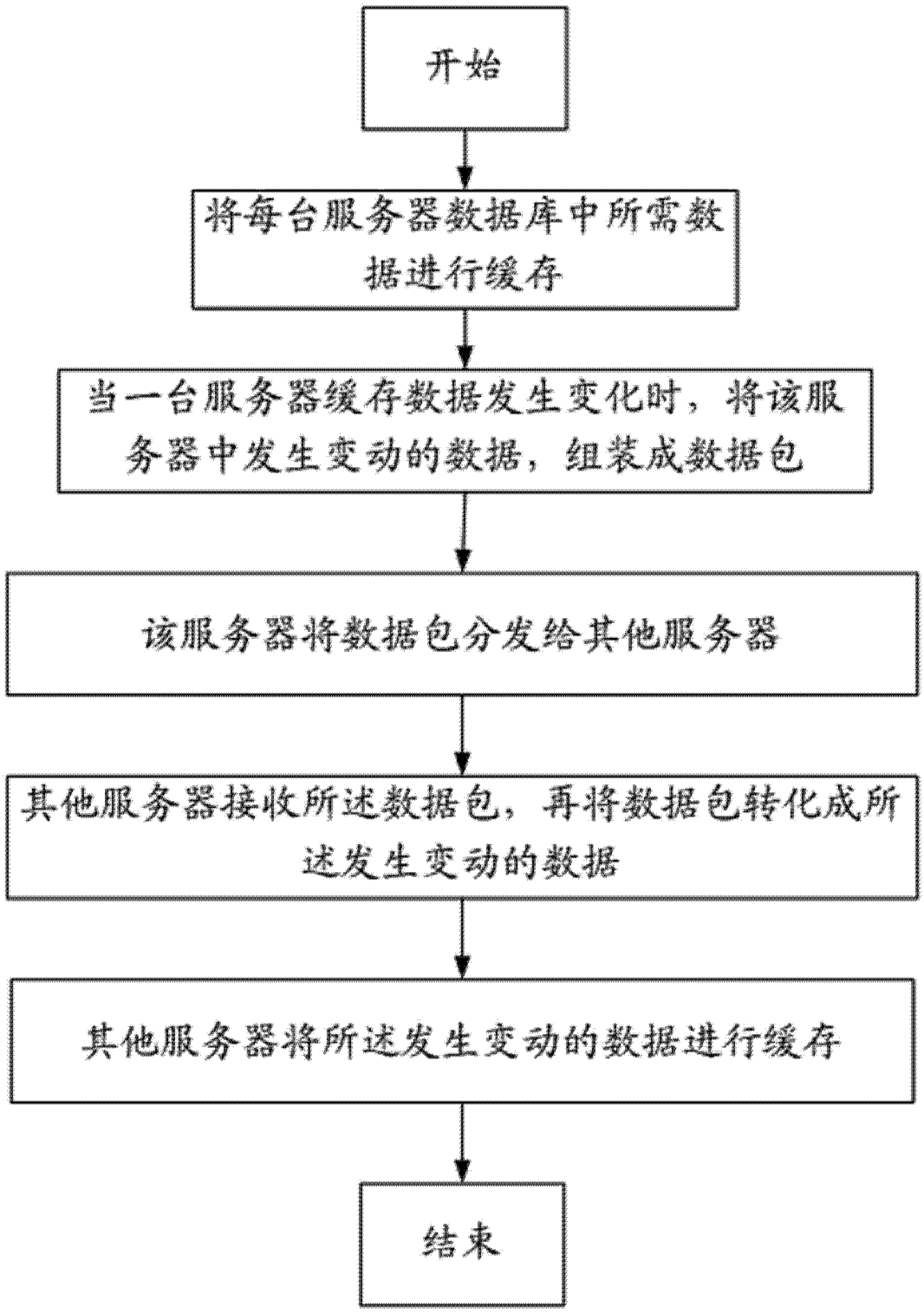

开始
将每台服务器数据库中所需数据进行缓存
当一台服务器缓存数据发生变化时，将该服务器中发生变动的数据，组装成数据包
该服务器将数据包分发给其他服务器
其他服务器接收所述数据包，再将数据包转化成所述发生变动的数据
其他服务器将所述发生变动的数据进行缓存
结束

图 1

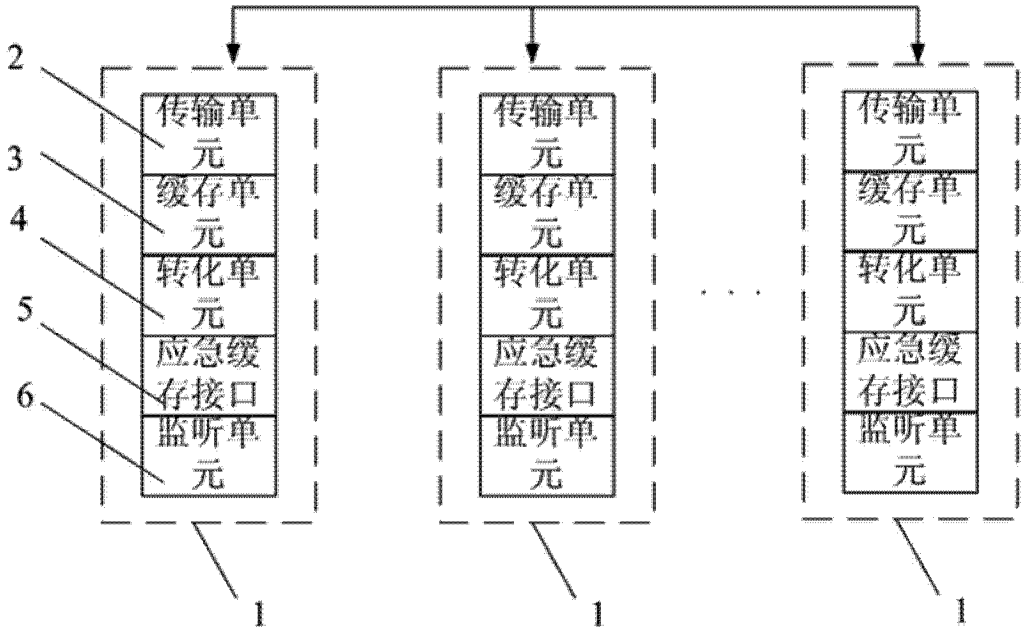


图 2