第 5 章

持久层 Redis 数据库设计

Redis（Remote Dictionary Server），即远程字典服务，Redis 是一个 key-value 型的 NoSQL 数据库。

Redis 的官网地址为 redis.io，从 2010 年 3 月 15 日起，Redis 的开发工作由 VMware 主持，Redis 的开发由 Pivotal 赞助。

5.1 Redis 功能介绍

与 Memcached 相比，Redis 支持存储的 value 类型更多，包括 string（字符串）、list（链表）、set（集合）、zset（sorted set 有序集合）和 hash（哈希类型）。这些数据类型都支持 push/pop、add/remove 及取交集、并集和差集或更丰富的操作，而且这些操作都是原子性的。在此基础上，Redis 支持各种不同方式的排序。与 Memcached 一样，为了保证效率，数据都是缓存在内存中。区别是 Redis 会周期性地把更新数据写入磁盘或写入追加的记录文件，并且在此基础上实现了 Master-Slave（主从）同步。

Redis 是一个高性能的 key-value 数据库，在很大程度补偿了 Memcached 这类 key/value 存储的不足，在部分场合可以对关系数据库起到很好的补充作用。Redis 提供了 Java、C/C++、C#、PHP、JavaScript、Perl、Object-C、Python、Ruby、Erlang 等客户端，使用非常方便。

Redis 支持主从同步，数据可以从主服务器向任意数量的从服务器上同步，这使得 Redis 可以执行单层树复制。存盘可以有意无意地对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，都可以订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有好处。

Redis 不是比较成熟的 Memcached 或者 MySQL 的替代品，是对于大型互联网应用在架构上很好的补充。现在越来越多的互联网平台纷纷在做基于 Redis 的架构改造，如京东、淘宝、当当、唯品会、美团、新浪微博等。

下面简单公布一下源于新浪微博的 Redis 平台实际应用数据，供参考。

- 每天 5000 亿次读写操作。
- 超过 18TB 内存。

- 500 多个服务器。

5.2 Redis 应用场景

Redis 的应用场景基于 Redis 自身的以下特点。

- Redis 是内存数据库，读写速度非常快，因此 Redis 适合做 Cache（缓存），用于提高数据访问速度，减少关系数据库的压力。
- 关系数据库的 Connection 连接数非常有限（单机在几千以内），而 Redis 的单机并发连接可以达到几十万，因此适合在高并发环境缓解关系数据库的压力。
- Redis 中没有事务（Transaction）控制，因此关键的交易数据读写仍然需要关系数据库，非关键数据的读写可以转到 Redis 中。
- Redis 数据之间没有关联关系，数据结构简单，数据拓展容易，因此 Redis 适合做大集群部署，数据承载量非常庞大。
- Redis 中的数据类型简单，无法做结构化查询（SQL），因此不适合有复杂关系的数据读写。

Redis 的实际应用非常广泛，下面简单列举几个常见的应用场景。

- 与关系数据库配合，做高速缓存。
- 全局计数器（如文章的阅读量、微博点赞数）。
- 利用 zset 类型存储排行榜数据。
- 利用 list 自然排序存储最新 n 个数据（新浪/Twitter 用户消息列表）。
- HTTP Session 数据存储。
- 分布式锁应用，防止超卖。
- 高并发环境全局 ID 获取（如生成唯一的订单编号）。
- 限流（int 类型的 incr 方法，限制访问次数）。
- 利用 hash 结构做购物车存储。
- 消息队列（list 提供了两个阻塞的弹出操作：blpop/brpop）。
- 防止缓存穿透和雪崩故障。

5.3 Redis 下载与安装

Redis 目标安装环境 CentOS7，安装步骤如下。

（1）下载 Redis 安装包。

从 Redis 官网下载 redis-5.0.10.tar.gz，使用 xftp 工具上传到/usr/local 目录下（如图 5-1 所示）。

（2）进入虚拟机 local 目录，解压 Redis 安装包。

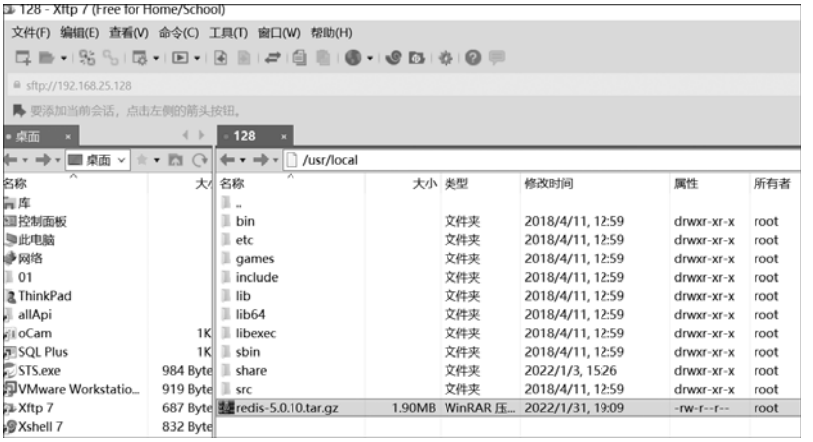

图 5-1　上传 Redis 安装包

```
# cd  /usr/local
# tar  -xzvf  redis-5.0.10.tar.gz      //把压缩包解压
# mv  redis-5.0.10 redis               //修改解压后的目录名
# rm  redis-5.0.10.tar.gz              //删除压缩包
```

（3）安装 GCC 编译环境。

```
# yum install -y gcc-c++
# gcc  -v                             //查看安装后的 GCC 版本
```

（4）使用 GCC 编译 Redis。

```
# cd  /usr/local/redis                //进入安装目录
# make                                //编译
# make                                //再次编译确认
```

（5）安装 Redis。

```
# make install
```

（6）检查 Redis 安装信息。

进入 cd /usr/local/bin 目录，若显示如图 5-2 所示信息，表示 Redis 安装成功。

```
[root@localhost redis]#  cd /usr/local/bin
[root@localhost bin]# ls
redis-benchmark  redis-check-aof  redis-check-rdb  redis-cli  redis-sentinel  redis-server
[root@localhost bin]#
```

图 5-2　Redis 安装信息

（7）修改 Redis 配置文件。

进入 Redis 的安装目录 cd /usr/local/redis，修改 redis.conf 的配置信息（如图 5-3 所示）。

```
[root@localhost redis]# cd /usr/local/redis
[root@localhost redis]# ls
00-RELEASENOTES  CONTRIBUTING  deps     Makefile   README.md   runtest          runtest-moduleapi  sentinel.conf  tests
BUGS             COPYING       INSTALL  MANIFESTO  redis.conf  runtest-cluster  runtest-sentinel   src            utils
[root@localhost redis]#
```

图 5-3　Redis 配置文件

修改 Redis 启动模式如下。

```
#vim redis.conf
     daemonize  yes        //作为后台服务启动
```

```
protected-mode no
#bind 127.0.0.1      //允许外部使用 IP 访问，注释掉该行
cat redis.conf       //检查修改结果
```

（8）启动 Redis 服务。

直接执行下面的命令行，redis-server 启动时需要访问 redis.conf 配置文件。

```
# /usr/local/bin/redis-server /usr/local/redis/redis.conf
# ps -ef | grep redis  //redis 启动后，检查 redis 进程（如图 5-4 所示）
```

```
[root@localhost redis]#  /usr/local/bin/redis-server /usr/local/redis/redis.conf
13777:C 31 Jan 2022 19:59:42.309 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
13777:C 31 Jan 2022 19:59:42.309 # Redis version=5.0.10, bits=64, commit=00000000, modified=0, pid=13777, just started
13777:C 31 Jan 2022 19:59:42.309 # Configuration loaded
[root@localhost redis]# ps -ef | grep redis
root      13778      1  0 19:59 ?        00:00:00 /usr/local/bin/redis-server *:6379
root      13797   9282  0 20:01 pts/0    00:00:00 grep --color=auto redis
[root@localhost redis]#
```

图 5-4　启动 Redis 服务器

（9）客户端连接 Redis 服务器（如图 5-5 所示）。

```
# redis-cli -p 6379                          //本地连接方式
# redis-cli -h 192.168.25.128  -p 6379       //远程连接方式
# ping                                       //测试回应 PING
# keys *
# exit                                       //客户端退出
```

```
[root@localhost redis]# redis-cli -p 6379
127.0.0.1:6379> ping
PONG
127.0.0.1:6379>
[root@localhost redis]# redis-cli -h 192.168.25.128  -p 6379
192.168.25.128:6379> ping
PONG
192.168.25.128:6379> keys *
(empty list or set)
192.168.25.128:6379>
```

图 5-5　客户端连接 Redis 服务器

（10）使用 Jedis 访问测试。

```
# service firewalld status        //查看防火墙状态
# service firewalld stop          //关闭所有端口的防火墙
```

5.4　案例：当当书城 Redis 实战

当当书城功能需求和物理表设计参见 4.8 节。本节在 4.8 节的基础上，引入 Redis 数据库对当当书城进行性能优化和功能完善。

5.4.1　Jedis 连接 Redis 服务器

Jedis 是连接 Redis 服务器最常用的客户端，在 Java 环境导入包 jedis-2.6.1.jar 和 commons-pool2-2.4.2.jar，即可使用 Jedis 客户端连接 Redis 服务器。

参考代码类 RedisUtil，使用连接池方式访问 Redis，核心代码如下。

```
public class RedisUtil {
     private static String ADDR = "192.168.25.128";    //Redis 服务器 IP
     private static int PORT = 6379;                   //Redis 的端口号
```

```
    private static JedisPool jedisPool = null;
    /**         * 初始化 Redis 连接池         */
    static {
        try {
            JedisPoolConfig config = new JedisPoolConfig();
            config.setMaxTotal(80000);          //可用连接实例的最大数量
            config.setMaxIdle(200);
            config.setMaxWaitMillis(10000);     //单位毫秒，默认值为-1，表示永不超时
            config.setTestOnBorrow(true);       //通过提前测试，确保连接可用
            jedisPool = new JedisPool(config, ADDR, PORT);
        } catch (Exception e) {
            Log.logger.error(e.getMessage(), e);
        }
    }
    /**
     *  获取 Jedis 实例
     * @return
     */
    public synchronized static Jedis getJedis() throws Exception {
        Jedis jedis = null;
        if (jedisPool != null) {
            jedis = jedisPool.getResource();
        }
        return jedis;
}
/**         * 释放 Jedis 资源         */
    public static void close(final Jedis jedis) {
        if (jedis != null) {
            jedisPool.returnResource(jedis);
        }
    }
}
```

5.4.2　图书缓存和排序

参见 4.8 节的当当书城主页（如图 4-38 所示），主页显示的图书都是后台设置的推荐图书（如图 4-40 所示的表设计）。传统的实现方法是从 MySQL 数据库中直接提取主页推荐图书，但是当当书城用户量庞大，频繁的主页访问会给 MySQL 数据库带来巨大的压力。

如果把图书数据缓存到 Redis 数据库中，主页推荐图书从 Redis 中提取，就会大大缓解 MySQL 数据库的压力。操作步骤如下。

（1）定义监听器，在系统启动时装载所有图书数据。

```
@WebListener
public class LoadListener implements ServletContextListener {
    private BookBiz bkBiz;
    /**     *    监听器访问 Spring 容器，加载所有图书       */
    public void contextInitialized(ServletContextEvent sce)  {
        ApplicationContext ctx = WebApplicationContextUtils
                .getWebApplicationContext(sce.getServletContext());
        bkBiz = ctx.getBean(BookBiz.class);
        try {
            Jedis jedis = RedisUtil.getJedis();
            loadAllBooks(jedis);
```

```
                setMainSortBooks(jedis);
                RedisUtil.close(jedis);
                Log.logger.info("数据库连接 OK，所有图书已经加载到 Redis 中");
                bkBiz.logPath();
            } catch (Exception e) {
                Log.logger.error(e.getMessage(),e);
            }
        }
        /**     * 把所有的图书信息都装载到 Redis 数据库中，存储类型为 hash     */
        private void loadAllBooks(Jedis jedis) throws Exception{
            List<Book> books = bkBiz.getAllBooks();      //从数据库中读取所有图书
            Map<String, String> map = new HashMap<String, String>();
            for(Book bk : books){
                String bkString=JsonUtils.objectToJson(bk); //每本书生成一个 json 串
                map.put(bk.getIsbn(), bkString);
            }
            jedis.hmset("allBookList",map);              //把所有图书，存于 hash 中
        }
        /**
         * 主页推荐图书需要按照显示顺序号进行排序，因此存储类型为 zset
         *  注意：zset 中只存储排序号和 ISBN，图像详情从 hash 中提取
         */
        private void setMainSortBooks(Jedis jedis)throws Exception{
            List<MBook> books = bkBiz.getAllMainBook();
            for(MBook bk : books){
                int  dno = bk.getDno();
                jedis.zadd("bkMainZset", dno,bk.getIsbn());
            }
        }
}
```

（2）所有图书数据，在 Redis 中按照 hash 结构存储，主键是 ISBN，值是 json 串。从 MySQL 数据库中提取的图书实体对象，需要转换成 json 串。实体与 json 串的互转方案，本书推荐使用 jackson 来实现，这也是 Spring 框架优先推荐的 json 转换方案。

```
public class JsonUtil {
    private static final ObjectMapper MAPPER = new ObjectMapper();
    public static String objectToJson(Object data) throws Exception{
        String string = MAPPER.writeValueAsString(data);
        return string;
    }
    public static <T> T jsonToObject(String jsonData, Class<T>beanType) throws
        Exception{
        T t = MAPPER.readValue(jsonData, beanType);
        return t;
    }
    public static <T> List<T> jsonToList(String jsonData,Class<T>beanType) throws
        Exception{
        JavaType JavaType = MAPPER.getTypeFactory().constructParametricType
                        (List.class, beanType);
        List<T> list = MAPPER.readValue(jsonData, JavaType);
        return list;
    }
}
```

（3）持久层使用 Mybatis 从 MySQL 中提取主页图书，装载到 Redis 中。

```
<select id="getAllMainBook" resultType="MBook">
```

```
     select b.isbn,b.bname,b.price,b.pic,m.dno,m.rtime
          from tbook b,TBookMain m where b.isbn=m.isbn order by m.dno desc
</select>
```

（4）用户访问书城主页时，从 Redis 中提取主页推荐图书信息。

```
@Controller
public class BookAction {
     @Autowired
     private BookRedis bkRedis;
     @RequestMapping("/main")
     public String getMainBook(Model model) throws Exception {
          List<Book> books = bkRedis.getAllMainBook(); //按顺序提取主页图书
          model.addAttribute("books",books);
          return "/jsp/main.jsp";
     }
}
```

（5）从 zset 中提取所有主页图书的顺序，然后从 hash 中提取图书详情。

```
@Service
public class BookRedis {
     /**     *  从 Redis 中读取所有主页推荐的图书        */
     public List<Book> getAllMainBook() throws Exception{
          List<Book> bkList = new ArrayList<Book>();
          Jedis jedis = RedisUtil.getJedis();
          Set<String> isbns = jedis.zrevrange("bkMainZset", 0, -1);
          for(String isbn : isbns){
              String bkstring = jedis.hget("allBookList", isbn);
              Book bk = JsonUtil.jsonToObject(bkstring, Book.class);
              bkList.add(bk);
          }
          RedisUtil.close(jedis);
          return bkList;
     }
}
```

5.4.3　统计图书访问次数

用户在当当书城，通过搜索或主页推荐找到图书后，单击进入图书详情页（如图 5-6 所示）。图书详情信息从 Redis 中提取，性能远高于访问 MySQL。同时图书的每一次访问，都会被记录下来，这样热点图书排行就很容易被统计出来了（如图 5-7 所示）。

图书访问次数统计的实现步骤如下。

（1）普通用户浏览图书时，统计访问次数。

```
@Controller
public class BookAction {
     @Autowired
     private BookRedis bkRedis;
     @RequestMapping("/bookInfo")
     public String getBookInfo(@RequestParam String isbn,Modelmodel) throws
          Exception{
          Book book = bkRedis.getBookInfo(isbn);   //从 Redis 中读取图书详情
          model.addAttribute("bk", book);
          bkRedis.addBookPageView(isbn);           //统计访问次数
          return "/jsp/BookDetail.jsp";
```

```
        }
}
```

图 5-6　图书详情浏览

图 5-7　图书访问次数统计

（2）BookRedis 逻辑类中的 getBookInfo()从 Redis 中读取图书详情。

```
public Book getBookInfo(String isbn) throws Exception {
     Jedis jedis = RedisUtil.getJedis();
     String bkstring = jedis.hget("allBookList", isbn);
     try {
          return JsonUtil.jsonToObject(bkstring, Book.class);
     } finally {
           RedisUtil.close(jedis);
     }
}
```

（3）BookRedis 逻辑类中的 addBookPageView()方法，表示每次浏览图书时访问次数加 1。

```
public void addBookPageView(String isbn) throws Exception{
     Jedis jedis = RedisUtil.getJedis();
```

```
        jedis.zincrby("bkpv", 1, isbn);       //zset 中存储访问次数，每次给 score 加 1
        RedisUtil.close(jedis);
    }
```

（4）管理员登录后，进入页面访问次数统计页。

```
    @GetMapping("/back/bkPageView")
    public String getBookPageView(Model model) throws Exception{
        List<Book> books = bkRedis.getBookPageView();
        model.addAttribute("bkpv", books);
        return "/back/bookPageView.jsp";
    }
```

（5）根据 zset 中的图书访问次数统计，显示排名在前 100 名的热门图书。

```
    public List<Book> getBookPageView() throws Exception{
        List<Book> books = new ArrayList<Book>();
        Jedis jedis = RedisUtil.getJedis();
        Set<Tuple> bkpv = jedis.zrevrangeWithScores("bkpv", 0, 100);
        for(Tuple tuple : bkpv){
            String bkJson = jedis.hget("allBookList", tuple.getElement());

            Book book = JsonUtil.jsonToObject(bkJson, Book.class);
            book.setPageview((int)tuple.getScore());
            books.add(book);
        }
        RedisUtil.close(jedis);
        return  books;
    }
```

（6）视图层显示图书访问排名列表。

```
    <table border="0" width=60% align="center">
        <tr>
            <td>书号 ISBN</td>
            <td>图书名称</td>
            <td>价格
            <td>访问次数</td>
        </tr>
        <c:forEach var="bk" items="${bkpv}">
        <tr>
            <td>${bk.isbn}</td>
            <td>${bk.bname}</td>
            <td>${bk.price}
            <td>${bk.pageview}</td>
        </tr>
        </c:forEach>
    </table>
```

5.4.4　图书评论

用户浏览器图书时，可以查看评论信息，已登录的用户还可以评论点赞。购买了图书的用户可以发表评论，如图 5-8 所示。

图书评论信息是非关键的业务数据（无须事务控制），因此既可以放在 MySQL 数据库中，也可以放到 Redis 中。

本节讲解基于 Redis 的图书评论代码实现方案，操作步骤如下。

图 5-8　图书评论与评论点赞

（1）从控制器 BookAction 中进入图书评论页。

```
@GetMapping("/pingLun")
public String pingLun(String isbn,Model model) throws Exception{
     List<BookComment> bcList = bkRedis.getBookComments(isbn);
     model.addAttribute("bcList", bcList);
     Book bk = bkRedis.getBookInfo(isbn);
     model.addAttribute("bk", bk);
     return "/jsp/bookComment.jsp";
}
```

（2）从 Redis 中提取已有的当前图书评论信息。在 Redis 中，一本书有多条评论，每本书的所有评论信息使用 list 结构存储，list 的命名规则为“pl-isbn 值”。

```
public List<BookComment> getBookComments(String isbn) throws Exception{
     List<BookComment> comms = new ArrayList<BookComment>();
     Jedis jedis = RedisUtil.getJedis();
     List<String>  bcList =  jedis.lrange("pl-"+isbn, 0, -1);
     for(String bc : bcList){
     BookComment bcJson = JsonUtil.jsonToObject(bc, BookComment.class);
          comms.add(bcJson);
     }
     RedisUtil.close(jedis);
     return comms;
}
```

（3）已购买图书的用户，可以提交评论信息。

```
@PostMapping("/pingLun")
public String pingLun(String bkComm,String isbn,Model model,@SessionAttribute
                    User user) throws Exception {
     bkRedis.addBookComment(isbn, user.getUname(), bkComm);
     List<BookComment> bcList = bkRedis.getBookComments(isbn);
     model.addAttribute("bcList", bcList);
     Book bk = bkRedis.getBookInfo(isbn);
     model.addAttribute("bk", bk);
     return "/jsp/bookComment.jsp";
}
```