文章编号：2095-2163(2023)08-0155-03 中图分类号：TP314 文献标志码：A

基于申威架构的 Redis 适配安装及性能验证

王 丰，鲍正刚
（中电科申泰信息科技有限公司，江苏 无锡 214000）

摘 要：Redis 作为内存数据缓存技术，当前已应用至各大网络服务中。申威处理器作为国产通用高性能处理器，具有高性能、低功耗且完全自主可控，在民用市场使用中生态存在缺失，同时生态应用完善不足。本文以 Redis-5.0.3 为例主要介绍 Redis 基于申威 SW3231 处理器下的编译安装应用及 Redis 服务在申威处理器下的性能验证。

关键词：申威架构；Redis；编译安装；存删性能

Redis adaptation installation and performance verification based on Shenwei architecture

WANG Feng, BAO Zhenggang
(China Electronics Technology Shentai Information Technology Co., Ltd., Wuxi Jiangsu 214000, China)

【Abstract】 As an in-memory data caching technology, Redis has been applied to all major network services. Shenwei processor, which is a domestic general-purpose high-performance processor, has high performance, low power consumption and is completely autonomous and controllable. There is a lack of ecology in civil market use, and at the same time, the ecological application is not perfect. Taking Redis-5.0.3 as an example, this paper mainly introduces the compilation and installation application of Redis based on Shenwei SW3231 processor and the performance verification of Redis service under Shenwei processor.

【Key words】 Shenwei processor; Redis; compile and install; save / delete performance

0 引 言

Redis 作为内存数据库得到广泛应用[1]，主要是安装在基于 x86 或 ARM 架构处理器的服务器上使用，并提供数据存储服务。

近年来，国产处理器的研发已取得可观成就。申威架构的处理器作为纯国产指令集架构处理器，现已应用于服务器和 PC 终端，其中的申威 SW3231 采用的是“申威 64”自主指令系统[2]，考虑到处理器为国产指令架构，故而软件生态受到较大限制。目前，随着国产申威处理器在民用服务中的推广使用，为实现国产申威架构生态发展，在申威架构上编译安装 Redis 内存数据库，同时也要验证在申威处理器服务器上运行性能。

1 编译安装

1.1 安装准备

在对 Redis 源码编译安装前，需要设置好其安装或编译所需的基础环境，设置时主要涉及到：gcc、gcc-c++、pcre-devel、zlib zlib-devel、openssl openssl-devel、tcl8.6.1。

综合前文论述中，gcc、gcc-c++、pcre-devel、zlib zlib-devel、openssl openssl-devel 可通过 yum 指令进行在线直接安装，安装命令示例为：yum install openssl openssl-devel。

准备待编译安装的 Redis 源码包：redis-5.0.3.tar、tcl8.6.1-src.tar 可在官网下载源码包或者 git 资源库中下载源码，tcl8.6.1 为 Redis 安装资源依赖，部分操作系统中已预安装、不需要重复安装，未安装的则要进行编译安装。

1.2 编译安装

(1)解压 redis-5.0.3.tar.gz。此处用到的指令为：

```
#tar -zxvf redis-5.0.3.tar.gz
```

(2)进入 scr 目录并编译。此处用到的指令为：

```
# cd redis-5.0.3/src
```

作者简介：王 丰(1987-)，男，工程师，主要研究方向：国产处理器申威架构处理器生态适配；鲍正刚(1996-)，男，工程师，主要研究方向：国产处理器申威架构处理器生态适配。

收稿日期：2022-09-07

(3)执行编译。此处用到的指令为:

make

编译过程中可能出现报错,报错时具体如图 1 所示。

```
In file included from adlist.c:34:
zmalloc.h:50:10: fatal error: jemalloc/jemalloc.h: No such file or directory
 #include <jemalloc/jemalloc.h>
          ^~~~~~~~~~~~~~~~~~~~~
compilation terminated.
make[1]: *** [Makefile:228: adlist.o] Error 1
make[1]: Leaving directory '/root/redis-4.0.9/src'
```

图 1 编译报错

Fig. 1 Compilation error

(4)重新编译。此处用到的指令为:

make MALLOC=libc

(5)编译通过后执行测试。此处用到的指令为:

make test

编译测试如出现图 2 显示的报错,就要对 tcl8.6.1-src.tar.gz 进行解压、及编译安装,此后再重新执行对 Redis 的编译安装。

```
[root@localhost src]# make test
You need tcl 8.5 or newer in order to run the Redis test
make: *** [test] Error 1
```

图 2 缺 tcl 报错

Fig. 2 Missing tcl error

(6)编译安装。此处用到的指令为:

make install PREFIX=/usr/local/redis

编译测试通过后即可执行编译安装,本文指定安装路径为:/usr/local/redis,编译安装后安装文件将位于该目录之下。

1.3 配置

(1)配置文件。从解压的源码安装包中找到配置文件 reds.conf,拷贝该文件至 redis 的安装路径的 bin 目录下:# cp redis.conf /usr/local/redis/bin/。

(2)配置。安装完成后进行配置,主要配置包含有连接端口、是否后台运行、数据存放地址、log 日志存放位置以及接入密码等。

其中,端口一般采用默认端口,后台启动需要将 daemonize no 改为 daemonize yes;数据存放地址默认存放在安装目录 bin 文件夹下,通过修改配置文件中 dir 后面的路径来修改存放地址,本文存放地址修改为:/home/data/redis-data 目录下;log 日志配置需要在 logfile 配置路径和 log 写入的文件,本文配置为:/usr/local/redis/logs/redis-server.log;系统默认设置是不需要密码登录,如需要启用接入密码则是把配置文件中# requirepass foobared 的注释去掉(即去掉#)[3],然后设置自己的密码。

配置文件修改完成后即可启动 Redis 服务,服务开启后在后台运行,运行过程中日志可在配置文件中配置的地址下查看对应的运行日志。

2 性能验证

完成编译适配后,通过统计写入和删除不同大小数据包所耗时间,对 Redis 数据缓存服务在申威架构处理器上应用的性能进行验证分析[4]。对此拟展开阐释分述如下。

(1)存储性能。存储性能验证:编写 Java 测试程序,调用 Redis 服务,将数据写入。程序运行在多核 x86 系统中,测试程序将同时开启 100 个线程,每个线程单独写入 10 000 条数据。

采集所有线程的运行时间,对所得结果进行平均,即可得到每个线程写入数据的耗时,耗时结果见表 1,耗时曲线如图 3 所示。

表 1 10 000 条数据写入平均耗时表

Tab. 1 Table of average time for writing 10 000 pieces of data

数据长度	2	4	8	16	32	64	128	256	512	1k	2k	3k	4k
单线程平均耗时/(10 000 条·ms^{-1})	14 799	15 028	15 201	14 684	15 270	16 395	23 900	39 976	74 146	142 159	277 281	409 686	542 890

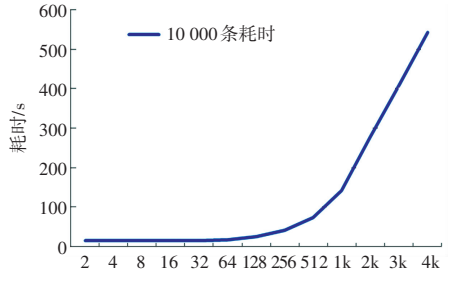

图 3 10 000 条数据写入平均耗时曲线图

Fig. 3 Average time-consuming curve of writing 10 000 pieces of data

(2)删除性能。删除性能验证时,同样采用 Java 程序进行调用 Redis 服务数据的删除,程序也是运行于 x86 操作系统中,同时开启 100 个线程,每个线程删除 10 000 条数据,采集到所有线程所耗时间,并得到平均数据。

至此,研究得到的每个线程删除数据平时耗时结果见表 2,耗时曲线如图 4 所示。

本节点对 Redis 数据缓存服务在申威架构(SW3231)处理器进行数据存储和数据删除性能的

验证，数据在大于 512 字节时，不论写入、还是删除，在数据处理中的用时出现明显增加，但总体上已能满足民用市场业务需求。

表 2　10 000 条数据删除平均耗时表

Tab. 2　Table of average time for deleting 10 000 pieces of data

数据长度	2	4	8	16	32	64	128	256	512	1k	2k	3k	4k
单线程平均耗时/(10 000 条 · ms^{-1})	14 277	14 725	14 468	14 163	14 715	14 624	14 352	14 328	15 862	21 425	35 848	47 318	54 523

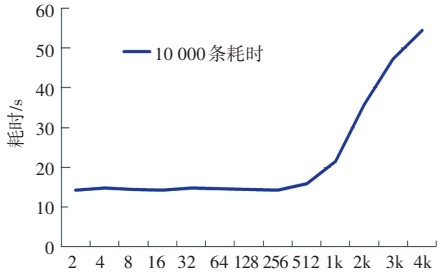

图 4　10 000 条数据删除平均耗时曲线图

Fig. 4　Average time-consuming curve of deleting 10 000 pieces of data

由上述验证可见，申威架构处理器已经在民用服务生态有了较大的跨越，性能使用基本上已然满足了现有民用市场的基本需求，完全可以实现申威架构处理器作为 Redis 数据缓存业务服务器来提供业务服务。

3　结束语

综上所述，Redis 在目前的申威架构处理器下已经可以实现编译适配，并且可作为独立数据缓存服务器来提供业务服务；而在本次研究中，还对Redis 数据缓存服务在申威架构处理器运行过程中的性能做了验证，验证结果表明申威架构处理器对 Redis 数据缓存服务的支撑完全满足当前大部分网络业务需求；自主可控的申威架构处理器在数据缓存功能上已经能够达到市场民用业务需求[5]，虽然其运行耗时在较大量级数据的存储和删除过程中出现了明显增大，但 Redis 服务作为数据缓存技术，在业务需求中通常都以小数据为主。同时，申威架构处理也在不断地进行升级完善，完全国产化的申威处理器已经能够满足民用业务上的需求。

参考文献

[1] 梅玉娜，冯东，李展，等. 基于 Redis 的网络大学平台性能优化研究[J]. 电力信息与通信技术，2016(12)：112-116.
[2] 郭兴，罗凯，刘彦飞. 一种基于国产申威 SW3231 处理器的高速存储设备的设计[J]. 电子质量，2021(12)：52-57.
[3] 赵学作. Redis 数据库的安装与配置[J]. 网络安全和信息化，2020(03)：108-110.
[4] 张恒，赵荣彩，董本松. 基于申威众核处理器的 MD5 解密算法优化[J]. 计算机与现代化，2022(02)：13-18.
[5] 施光源. 基于国产申威处理器的自主可控产品实践之路[J]. 网络空间安全，2018(07)：78-81.

(上接第 154 页)

[16] ZHANG Xuan, LIANG Xun, ZHIYULI A, et al. At-lstm: An attention-based LSTM model for financial time series prediction [J]. IOP Conference Series: Materials Science and Engineering, 2019, 569(5): 1-8.
[17] SONG Youwei, WANG Jiahui, LIANG Zhiwei, et al. Utilizing BERT intermediate layers for aspect based sentiment analysis and natural language inference[EB/OL]. (2022-02-12)[2022-06-09]. https://arxiv.org/abs/2002.04815.
[18] MA Dehong, LI Sujian, ZHANG Xiaodong, et al. Interactive attention networks for aspect-level sentiment classification[C]// Proceedings of the 26th International Joint Conference on Artificial Intelligence. Melbourne, Australia: AAAI, 2017: 4068-4074.