·体系结构与软件技术· 文章编号：1000-3428(2018)05-0047-06 文献标志码：A 中图分类号：TP316

基于多核 ARM 体系结构的基础函数优化方法

贺爱香[1,2]，顾乃杰[1]，苏俊杰[1]

（1. 中国科学技术大学 计算机科学与技术学院，合肥 230027；2. 安徽新华学院 信息工程学院，合肥 230088）

摘 要： 为充分利用嵌入式多核 ARM 微处理器体积小、功耗低、成本低、性能高的优点，以此提高程序响应速度，研究 ARM 体系结构及基于该体系结构基础函数的优化问题。基于 ARM Cortex-A72 平台，结合 ARM v8 体系结构特点，对 Bionic 库中字符串和内存处理函数进行分析。实验结果表明，采用整字处理、循环展开、特殊指令等技术进行程序级优化后，Bionic 库中常用基础函数的性能在 ARM Cortex-A72 平台上均有不同程度的提升。

关键词： 多核 ARM；体系结构；整字处理；循环展开；特殊指令

中文引用格式： 贺爱香，顾乃杰，苏俊杰. 基于多核 ARM 体系结构的基础函数优化方法[J]. 计算机工程，2018，44(5)：47-52，59.

英文引用格式： HE Aixiang，GU Naijie，SU Junjie. Optimization Method of Basis Function Based on Multi-core ARM Architecture[J]. Computer Engineering，2018，44(5)：47-52，59.

Optimization Method of Basis Function Based on Multi-core ARM Architecture

HE Aixiang[1,2]，GU Naijie[1]，SU Junjie[1]

（1. School of Computer Science and Technology，University of Science and Technology of China，Hefei 230027，China；
2. School of Information Engineering，Anhui Xinhua University，Hefei 230088，China）

【Abstract】 In order to make full use of the advantages of embedded multi-core ARM microprocessor，such as small size，low power consumption，low cost and high performance to improve the system speed，this paper studies the architecture of ARM and the optimization problem of basic function based on the architecture. It analyzes the string and memory processing functions in Bionic library based on the structural characteristics of ARM v8 on the ARM Cortex-A72 platform. Experimental results show that after program level optimization of whole word processing，loop unrolling and special instruction optimization techniques，the performance of commonly used base functions in the Bionic library is improved to varying degrees on the ARM Cortex-A72 platform.

【Key words】 multi-core ARM；architecture；whole word processing；loop unrolling；special instruction

DOI： 10.19678/j.issn.1000-3428.0047957

0 概述

随着嵌入式系统的发展，多核 ARM 微处理器[1]因其体积小、功耗低、成本低、性能高等特点很快占领了移动通信市场领域。Android[2]是运行于 ARM 架构处理器之上的一款开源操作系统软件，据市场研究公司 Gartner 2016 年上半年最新统计数据显示，Android 市场占有率为近九成[3]。Bionic[4]是 Android 系统中以 C 语言构建的基础函数库，为系统提供系统接口和扩展功能函数。在处理器的性能得到提高的情况下，程序充分利用处理器的性能，不但可以减少计算资源的浪费，还可以提高程序的响应速度，因此对 Bionic 库基础函数进行优化对提升 ARM 平台性能与用户体验有很大作用[5]。针对 ARM 体系结构进行优化是非常重要的工作，很多学者和研究人员利用各种优化方法积极探索该问题。为了高效合理地使用寄存器和存储器，提高程序的执行效率，文献[6]对 ARM 体系结构提出了减小汉明距离、使用位操作指令、正确选择流程控制语句和压缩代码密度等优化方法。文献[7]基于 ARM 嵌入式系统充分利用软硬件资源，提出了循环展开、避免使用除法、利用条件执行等 C 程序优化方法。文献[8]使用 128 位访存指令、循环展开等方法，结合龙芯 3A 处理器将 BLAS 库性能提高了 6 倍。文献[9]对 Bionic 库中的热点函数进行了汇编优化，使得 Android 系统整体性能得到了提升。

基金项目： 安徽省自然科学基金(1408085MKL06)；安徽省教育厅自然科学基金重点项目(KJ2015A300)；安徽省教育厅项目(2016mooc197)。

作者简介： 贺爱香(1978—)，女，讲师、硕士，主研方向为体系结构优化、数据挖掘；顾乃杰，教授、博士生导师；苏俊杰，博士研究生。

收稿日期： 2017-07-13 **修回日期：** 2017-10-14 **E-mail：** heaixiang_1010@126.com

基于以上研究,本文在对 Bionic 库中字符串和内存处理函数进行分析的同时,结合多核 ARMv8 体系结构特征[10],提出不同的优化方案,包括整字处理[11]、循环展开[12]、特殊指令[13]等优化方法,使得 Bionic 库中常用基础函数的性能在 ARM Cortex-A72 平台上有不同程度的提升。

1 背景知识

1.1 ARM 体系结构

ARM 是一类微处理器的统称[14],基于 ARM 内核的芯片统称为 ARM 芯片。ARM 体系结构采用精简指令集,具有定长指令、大量的寄存器、独特的装载/保存(Load/Store)等特点[15]。ARM 公司从 1991 年推出 ARM1 处理器到现在,其体系结构已经从 ARMv1 发展到现在的 ARMv8,每一个体系架构版本都定义了一套指令集和相应的功能框架,并且每个结构体系向后兼容,图 1 为 ARMv5 至 ARMv8 的架构比较[16]。基于 ARMv8 的体系结构产品不仅主导了移动通信领域,而且在无人机控制系统[17]、汽车导航[18]、智能家居[19]等智能硬件控制系统中应用广泛,本文是基于 ARMv8 最新架构的研究。

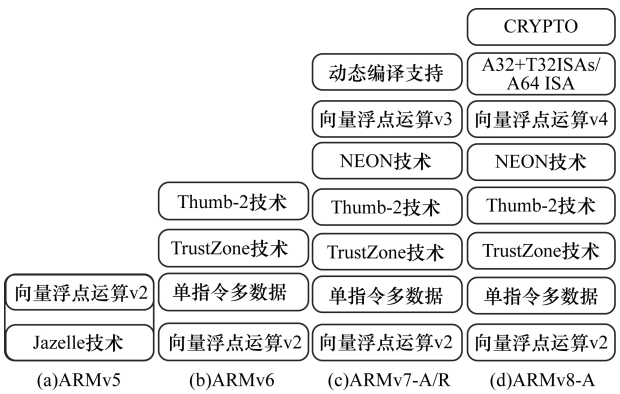

图 1 ARMv5 至 ARMv8 架构比较

ARMv8 提供一种更加清晰的架构,同时考虑到将来的发展趋势采用一种全新的架构来实现,是目前最新的架构[20]。ARMv8-A 系列面向尖端的基于虚拟内存的操作系统和用户应用。ARMv8 的架构继承以往 ARMv7 与之前处理器技术的基础,除了支持现有的 16 bit/32 bit 的 Thumb2 指令集外,也向前兼容现有的 A32(ARM 32 bit)指令集和 NEON 指令集,并扩充了现有的 A32(ARM 32 bit)和 T32(Thumb2 32 bit)指令集。ARMv8 采用基于 64 bit 的 AArch64 架构,新增了 A64(ARM 64 bit)指令集,解决了 ARMv7 架构遗留的虚拟地址问题,定义了 AArch64 和 AArch32 两套运行环境(称作 Execution state),分别执行 64 bit 和 32 bit 指令集,软件可以在需要的时候,切换 Execution state。AArch64 架构使用新的概念(exception level),重新解释了 processor mode、privilege level 等概念。在 ARMv7 安全扩展的基础上,新增了 CRYPTO(加密)模块,支持安全相关的应用需求[21]。在 ARMv7 虚拟化扩展的基础上,提供完整的虚拟化框架,从硬件上支持虚拟化[22]。其架构框图如图 2 所示。

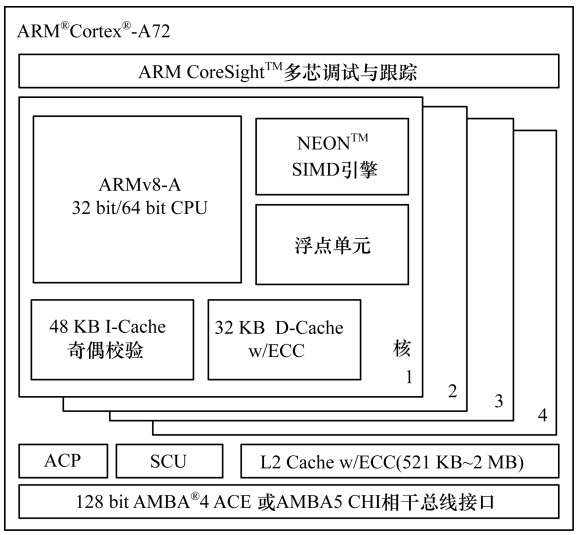

图 2 ARM Cortex-A72 处理器架构

1.2 NEON 技术

现代计算机 2 个大主流体系结构为 X86 和 ARM,为了高效利用硬件资源,均支持单指令多数据流(Single Instruction Multiple Datastream,SIMD)技术[23]。NEON 是 ARM Cortex-A 系列处理器新增的一个运算部件,使得每次运算可达到 128 位,扩展了 SIMD,能更快速地处理多媒体数据,使用户体验更流畅[24]。NEON 具有 128 位宽的运算部件,可减少对内存的访问,从而提高数据处理速度。

NEON 指令可批处理 SIMD;通过对齐和非对齐数据访问,寄存器被视为同一数据类型的元素的矢量;数据类型可为 8 位、16 位、32 位、64 位有/无符号整型、32 位单精度和 64 位双精度浮点型;指令在所有通道中执行同一操作。如图 3 所示,显示了 UADD16 Q0,Q1,Q2 指令操作,实现了对寄存器 Q1 和 Q2 中 8 路 16 bit 的并行相加,并将最后结果保存到 R0 中。

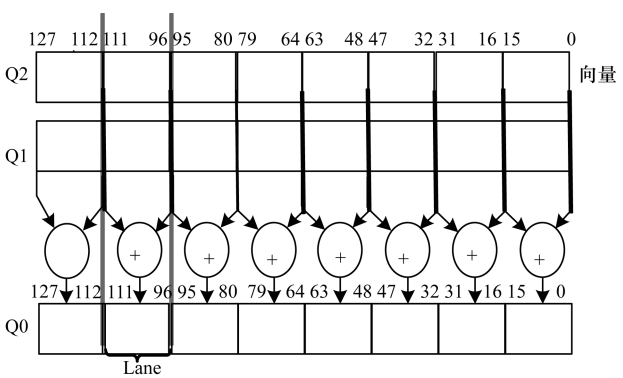

图 3 8 路 16 bit 整数加法运算

1.3　Bionic

Bionic 是 Google 公司专为 Android 系统开发的基础函数库,提供 C/C + + 标准库功能。GNU/Linux 以及其他类 Unix 系统的基础函数库最常用的就是 Glibc。与 Glibc 相比,Bionic 是轻量级 C 库,大小仅为 200 KB,是 Glibc 体积的一半,这意味着 Bionic 占用低内存[25]。

Bionic 库中包含字符串与内存处理函数、数学计算函数、排序与查找、字符串加密等基础工具型函数。本文通过大量实验,着重优化 Bionic 库中的 memset、strcmp、strncmp、strcpy、strncpy、memcpy、memmove、strlen、strcat、strdup 等函数。

2　函数优化

针对多核 ARMv8 体系结构,考虑使用整字处理(合并字节)、特殊指令、循环展开、指令调度等方法进行优化。本节将探讨如何使用这些优化技术,来实现 Bionic 库中热点函数的优化。

2.1　函数介绍

在 Bionic 库函数中,字符串是通过一段连续的内存来表示,所以字符串的操作与内存操作基本一致。两者的区别在于字符串是用空字符'\0'来表示结尾的,而内存块结尾由其给出大小决定。根据函数的不同功能,可以将字符串和内存处理函数分为以下 7 类,并选取代表性的热点函数进行优化。

1)求字符串长度的函数有 strlen、strnlen。

2)内存拷贝与赋值函数有 memcpy、menpcpy、memmove、memccpy、memset 等。

3)字符串拷贝与连接函数有 strcpy、strncpy、strdup、strndup、stpcpy、stpncpy、strcat、strncat 等。

4)字符串与内存比较函数有 memcmp、strcmp、strncmp 等。

5)字符串与内存查找函数有 memchr、memrchr、strchr、strrchr、strstr、strcasestr、memmem、strspn 等。

6)字符串分割函数有 strtok 等。

7)字符串变换函数有 strfry 等。

2.2　整字处理

整字处理是一种简单的合并操作,即将多个字节合并在一起进行处理。通过分析字符串与内存处理函数得出,大部分函数的行为都是从 1 个或 2 个内存地址开始读取内存并进行相应的操作。改变原始单个字节的遍历方式,合并字节,以 4 Byte 或 8 Byte或者 16 Byte 为单位进行处理。

而在 32 位的 ARM Cortex-A7 平台上可以 4 Byte合并处理。如测量字符串长度 strlen 函数,原始函数需要从内存逐字节取数到寄存器进行计算,使用整字处理可以将 4 Byte 看成一个单元计算。即先按整字判断,当指针已经是 4 Byte 对齐的位置时,每次读 4 个字节一起比较,直到最后一次 4 Byte 的判断,其中必有一个字节为'\0'。所以这个串尾的判断是关键。

对于所有整字节优化处理时的串尾判断,都可以考虑使用 DEF(X)判断其中是否含'\0'。如果没有,继续循环读取,如果发现了读进来的字符有'\0',则将字节数合并或按单字节比较找到第一个出现'\0'的位置,然后计算完毕并且返回。对于 4 Byte、32 位的情况,可定义 DEF(X)为(((X)-0X01010101)&~(X)&0X80808080),其中,X 代表 4 Byte 合并的数据,0X01010101 和 0X80808080 为在 4 Byte 情况下的 2 个常量,每个字节的最高位和最低位分别设置为 1。如果 DEF(X)不为 0,则说明 X 中含有一个值为 0 的字节。在 Bionic 库中的求字符串长度函数、内存拷贝与赋值函数、字符串拷贝与连接函数、字符串与内存比较函数均可采用该方法进行优化。以 strlen 函数为代表的优化流程设计如图 4 所示。

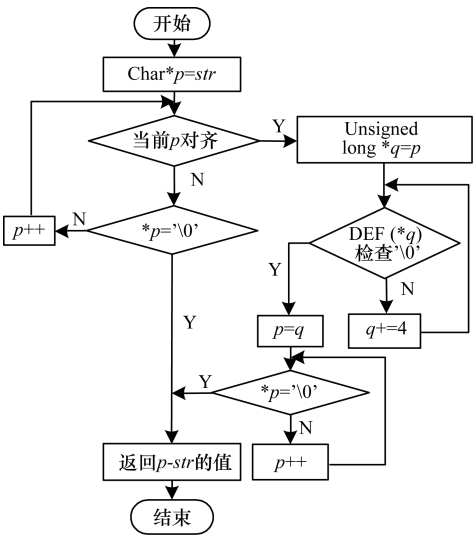

图 4　strlen 函数的优化算法流程

2.3　循环展开

循环展开是一种通过复制循环体中的指令,达到扩大循环体、减少循环重复次数的优化方法。通过减少循环变量的计算、比较次数和跳转指令的执行次数来提高代码性能。循环变量的计算和比较都需要一定的执行时间,并且在 ARM 上跳转指令会打断流水线的正常运转,是执行周期较长的指令。一般情况下对循环体较短的循环进行展开可以有效地减少循环的各种开销,提升程序代码效率。

循环展开除了能降低循环开销以外,更重要的作用在于展开后可以对变量进行重命名以减少数据相关和对基本块进行指令调度。如图 5 所示,图 5(a)是原来的循环,对其直接展开 4 次可得到图 5(b);图 5(b)中的变量 *S* 会引起数据相关,依次对每一条语句中的 *S* 变量进行重命名得到图 5(c),

有效地减小了数据相关;在 ARM 体系结构中,所有的操作都需要在寄存器中完成,所以图 5(c)中的每条语句事实上需要 2 条指令完成,先是将数组中的值取到寄存器中,然后进行累加运算,而这两步本身也存在写后读相关,所以增加中间变量并对其进行指令调度得到图 5(d),取数指令和运算指令之间存在一定的时间间隔,有效地减少了数据相关带来的延迟。

```
for(i=0;i<N;i++)
{
S+=a[i];
}
```
(a)未展开

```
for(i=0;i<N;i+=4)
{
S+=a[i];
S+=a[i+1];
S+=a[i+2];
S+=a[i+3];
}
```
(b)4次展开

```
for(i=0;i<N;i+=4)
{
r1+=a[i];
r2+=a[i+1];
r3+=a[i+2];
r4+=a[i+3];
}
S=S1+S2+S3+S4;
```
(c)展开后变量重命名

```
for(i=0;i<N;i+=4)
{
r1+=a[i];
r2+=a[i+1];
r3+=a[i+2];
r4+=a[i+3];
S1+=r1;
S2+=r2;
S3+=r3;
S4+=r4;
}
S=S1+S2+S3+S4;
```
(d)展开后指令调度

图 5 循环展开示意图

循环展开作为程序优化技术已非常成熟,常用的编译器都会支持这个功能。其直接作用是减少循环控制指令的开销,多层循环的外层循环展开还可以减少访存操作次数。但其缺点就是会导致代码膨胀,增加 Cache 的压力、寄存器使用的冲突、边界处理代码和控制开销。由于 ARM Cortex-A72 的一级 Cache 容量高达 64 KB,二级 Cache 容量为 2 MB,一般热点代码都比较短,因此代码膨胀对其影响不大。在特定平台上需要解决的关键问题是循环展开因子的选择,一般的准则是循环内定点和浮点变量的数量都不要超过寄存器的总量。

令 ARM 处理寄存器数为 R,循环展开 k 次后循环体所需要的寄存器数目为 $P(k)$,则为了避免寄存器冲突,最优循环展开因子为:

$$\max\{k\},P(k)\leqslant R,k\in Z \tag{1}$$

ARM Cortex-A72 处理器有 31 个 64 位长的通用寄存器,即 $R=31$。以图 5 代码为例,当 4 次展开时并且采用变量重命名和指令调度,循环体每次迭代时需要 8 个寄存器,展开 k 次则需要 $P(k)=2k$ 个寄存器,由式(1)可知 $2k\leqslant 31$,则 $k\leqslant 15$,计算得到循环展开因子取 15 为最佳因子。但在实际应用中,为了保证寄存器不冲突,对于不同平台,最优的展开因子需要通过实验搜索比 k 略小的值来获得。

Binoic 库中的内存拷贝与赋值函数 memcpy、memset,字符串拷贝与连接函数 strncpy、strncat,字符串与内存比较函数 strcmp、memcmp,函数的实现循环体短导致指令跳转的次数多,流水中断次数多,指令流水线的加速优势得不到充分利用。采用循环展开的方法可以减少指令跳转的次数,从而提升流水性能。

对任意循环都可采用循环展开以提升性能,但循环展开也会导致代码膨胀、增加寄存器压力等副作用。所以,针对不同循环问题如何选取循环展开因子,需结合体系结构实验,使程序性能最优。memset 函数是访问密集型函数之一,以 memset 函数为代表利用循环展开并以 4 为因子直接展开的优化流程设计如图 6 所示。

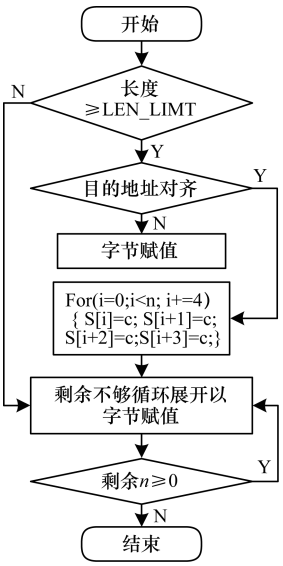

图 6 memset 函数的优化算法流程

2.4 特殊指令

使用特殊指令是通过使用目标机器指令集中一些特殊的指令来提高程序的性能。ARM Cortex-A7 采用了先进的 NEON 指令集技术,拥有多条 128 位访存指令。对数据的加载和存储分别使用 vld1q 和 vst1q 指令一次可以将 128 位数取到寄存器中,使用这些指令可以极大地提高取数和存储数的效率。在 NEON 中对齐与非对齐都使用 vld1q 和 vst1q 指令,但非对齐处理的周期要长。利用该指令分别对字符串和内存处理函数进行改写从而优化 Bionic 库。本文以 memcpy 函数为代表进行优化设计,流程图如图 7 所示,memcpy 函数优化算法考虑 dst 地址与 scr 地址对齐的情况,分为 dst 与 scr 地址 16 Byte 对齐、dst 与 scr 地址非对齐,但偏移值相等、dst 与 scr 地址非对齐,且偏移值不等 3 种情况。

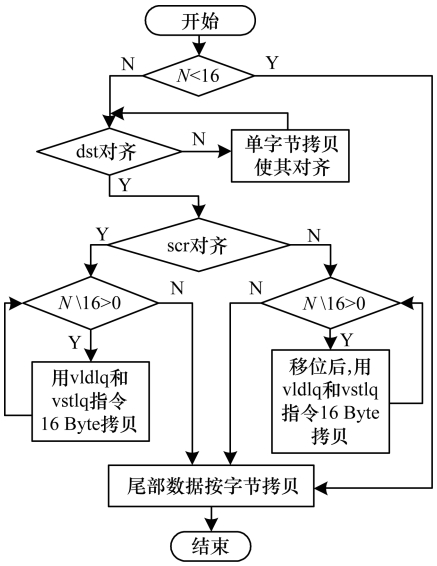

图 7　strcpy 函数的优化算法流程

3　实验结果分析

本文中实验采用 ARM Cortex-A7 双核处理器 Android 平台,取 64 B 到 512 KB 倍增的数据规模(size)进行测试,并对每个 size 运行十万次求平均,记录了原函数和优化后函数的运行时间,再计算出性能提升比。

在 Bionic 库中的字符串和内存处理函数均可采用整字处理方法优化,以 strlen 函数为代表采用整字处理,优化结果如表 1 所示。其在不同的数据规模下平均性能提升了 37.96% 。

表 1　strlen 函数优化性能比较

size	my_strlen/μs	strlen/μs	性能提升比例/%
64 Byte	86	86	0.00
128 Byte	146	164	10.98
256 Byte	234	354	33.90
512 Byte	498	889	43.98
1 KB	897	1 678	46.54
2 KB	1 965	3 576	45.05
4 KB	3 879	6 879	43.61
8 KB	7 543	13 995	46.10
16 KB	15 678	28 234	44.47
32 KB	31 018	57 456	46.01
64 KB	65 012	117 540	44.69
128KB	132 678	233 764	43.24
512 KB	261 013	473 785	44.91

在多核 ARM Cortex-A72 处理器平台中,通过以 memset 函数为代表采用循环展开进行优化实验,在不同数据规模的情况下,分别将循环展开因子取 1、2、3、4、5、6、7、8、9、10、11、12、16 进行实验,计算平均性能比较结果如图 8 所示。展开因子在 5 以内,同时数据在 L1-Cache 中时,访存速度快,循环体的代码执行周期很少,这时性能提升比例几乎可以取得线性提升。memset 函数展开时虽不增加对寄存器的消耗,展开因子超过 5 时,继续展开效果不明显,直到展开因子为 8 时性能达到最优,这是由于循环展开的副作用开始产生,导致整体性能开始波动,继续展开会引起其他不可控因素而使得性能下降。当展开因子取 8 时平均性能提升比为 83.08% ,如表 2 所示。

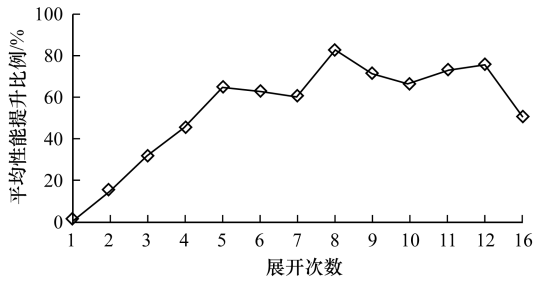

图 8　循环展开效果

表 2　memset 函数优化性能比较

size	my_memset1/μs	memset/μs	性能提升比例/%
64 Byte	40	121	66.94
128 Byte	73	214	65.89
256 Byte	80	458	82.53
512 Byte	132	885	85.08
1 KB	251	1 831	86.29
2 KB	488	3 540	86.21
4 KB	1 038	7 141	85.46
8 KB	1 984	14 405	86.23
16 KB	3 815	28 503	86.62
32 KB	7 446	57 190	86.98
64 KB	14 801	113 678	86.98
128 KB	28 480	228 057	87.51
512 KB	57 961	455 292	87.27

以 strcpy 函数为代表利用特殊指令对 Bionic 库中所有字符串和内存函数进行优化,并考虑对齐、非对齐偏移相等和非对齐偏移不等 3 种情况,优化结果分别如表 3 ~ 表 5 所示。strcpy 函数源地址和目标地址均对齐时平均性能提升 60.44% ,非对齐偏移量相等时平均性能提升 39.93% ,非对齐偏移量不等时平均性能提升 43.75% 。

表 3 strcpy 函数对齐优化性能比较

size	my_strcpy/μs	strcpy/μs	性能提升比例/%
64 Byte	86	205	58.05
128 Byte	176	379	53.56
256 Byte	299	789	62.10
512 Byte	631	1 630	61.29
1 KB	1 187	2 974	60.09
2 KB	2 472	6 500	61.97
4 KB	4 834	11 890	59.34
8 KB	8 909	23 093	61.42
16 KB	16 608	43 891	62.16
32 KB	33 772	89 785	62.39
64 KB	80 235	204 540	60.77
128 KB	142 930	366446	61.00
512 KB	306 273	798485	61.64

表 4 strcpy 函数非对齐偏移量相等优化性能比较

size	my_strcpy/μs	strcpy/μs	性能提升比例/%
64 Byte	109	118	7.63
128 Byte	219	252	13.10
256 Byte	325	527	38.33
512 Byte	854	1 465	41.71
1 KB	1 645	2 978	44.76
2 KB	3 198	5 987	46.58
4 KB	5 938	11 345	47.66
8 KB	10 923	21 389	48.93
16 KB	20 897	40 569	48.49
32 KB	50 602	96 534	47.58
64 KB	102 334	196 783	48.00
128 KB	255 647	455 697	43.90
512 KB	456 745	793 426	42.43

表 5 strcpy 函数非对齐偏移量不等优化性能比较

size	my_strcpy/μs	strcpy/μs	性能提升比例/%
64 Byte	102	136	25.00
128 Byte	198	267	25.84
256 Byte	356	598	40.47
512 Byte	735	1 287	42.89
1 KB	1 534	2 894	46.99
2 KB	3 113	5 698	45.37
4 KB	7 002	13 734	49.02
8 KB	10 873	22 086	50.77
16 KB	24 543	48 965	49.88
32 KB	44 567	89 058	49.96
64 KB	103 564	195 653	47.07
128 KB	254 643	489 753	48.01
512 KB	407 645	775 644	47.44

4 结束语

本文围绕 Android 系统底层 Bionic 库字符串和内存处理函数,结合 ARM Cortex-A72 处理器及其体系结构和指令集特点,通过使用整字处理、循环展开、NEON 访存特殊指令等优化方法进行了程序级优化设计。实验结果表明,Bionic 库中的字符串和内存处理函数在不同数据规模下平均性能均有提升,并通过大量实验找出了适合于该平台的 memset 函数最优循环展开因子为 8。

下一步将进一步分析 ARM Cortex-A72 处理器性能及特点,深入研究软件流水[26]对循环展开的影响,找出每个函数的最优展开因子以及影响多层循环展开性能的主要因素。融合整字处理、循环展开、指令调度、数据预取[27]等优化方法,对 Bionic 库中字符串和内存处理函数、数学计算、查找与排序、加密等函数进行优化,实现高性能的 Bionic 库。

参考文献

[1] 赵利军,王震宇,王奕森,等. 基于 ARMv8 架构 gadget 自动搜索框架[J]. 计算机应用与软件,2016,33(5):307-311,316.

[2] 雒海东. 基于 ARM 和 Android 的智能物联网药箱的设计[J]. 电子设计工程,2017,25(5):146-149.

[3] August. 谷歌 Android 夺回 IOS 部分市场份额[EB/OL]. [2017-05-18]. http://mobile.yesky.com/463/104559463.html.

[4] 冯璐霞,李春江,黄亚斌. 面向 ARM64 架构多核微处理器的模板计算性能优化研究[J]. 计算机工程与科学,2017,39(5):829-833.

[5] 叶炳发. Android 操作系统移植及关键技术[D]. 广州:暨南大学,2010.

[6] 谢 川,贺玲玲. 基于 ARM 处理器的软件优化设计[J]. 微计算机信息,2009,25(4):164-166.

[7] 金 丽,包志华,陈海进. 基于 ARM 嵌入式系统的 C 程序优化设计方法[J]. 南通大学学报(自然科学版),2006,5(3):61-64.

[8] 顾乃杰,李 凯,陈国良. 基于龙芯 2F 体系结构的 BLAS 库优化[J]. 中国科学技术大学学报,2008,38(7):854-859.

[9] 曹 越. 面向 Android 系统库文件访存的汇编优化策略[J]. 测控技术,2016,35(1):113-117,126.

[10] 匿名. ARM 授权博通 ARMv7 与 ARMv8 架构[J]. 单片机与嵌入式系统应用,2013,13(3):38.

[11] 罗红兵,张晓霞,王 伟,等. 科学计算应用程序单核指令级优化研究[J]. 计算机研究与发展,2014,51(6):1263-1269.

[12] 何颂颂,顾乃杰,朱海涛,等. 面向龙芯 3A 体系结构的 BLAS 库优化[J]. 小型微型计算机系统,2012,33(3):571-575.

[13] MISHRA A,AGARWAL M,RAJU K S. Hardware and software performance of image processing applications on reconfigurable systems[C]//Proceedings of Annual IEEE India Conference. Washington D. C.,USA:IEEE Press,2015:1-5.

(下转第 59 页)

[5] 王　鹃,胡　威,张雨菡,等. 基于 Docker 的可信容器[J]. 武汉大学学报(理学版),2017,63(2):15-25.
[6] BERNSTEIN D. Containers and cloud: from LXC to docker to Kubernetes [J]. IEEE Cloud Computing, 2014,1(3):81-84.
[7] 杨　鹏,马志程,彭　博,等. 集成 Docker 容器的 OpenStack 云平台性能研究[J]. 计算机工程,2017,43(8):26-31.
[8] 齐　勇. 基于 docker 的网络服务质量控制器的设计与实现[D]. 济南:山东大学,2015.
[9] Docker container networking [EB/OL]. [2017-05-10]. https://docs.docker.com/ engine/userguide/networking/.
[10] EDELMAN J. Docker networking [EB/OL]. [2017-04-25]. http://www.dasblinkenlichten.com/docker-networki ng-101-hostmode/.
[11] XIE Bin, SUN Guanyi, GUO Ma. Docker based overlay network performance evaluation in large scale streaming system[C]//Proceedings of 2016 IEEE Conference on Advanced Information Management, Communicates, Electronic and Automation Control. Washington D. C., USA:IEEE Press,2016:366-369.
[12] 龚　正,吴治辉,叶伙荣,等. Kubernetes 权威指南[M]. 北京:电子工业出版社,2016.
[13] Google. Calico documentation [EB/OL]. [2017-05-10]. http://docs.projectcalico.org/v2.0/introduction/.
[14] 冯明振. 基于 macvlan 的 Docker 容器网络系统的设计与实现[D]. 杭州:浙江大学,2016.
[15] 王亚玲,李春阳,崔　蔚,等. 基于 Docker 的 PaaS 平台建设[J]. 计算机系统应用,2016,25(3):72-77.
[16] DUSIA A, YANG Y, TAUFER M. Network quality of service in Docker containers[C]//Proceedings of 2015 IEEE International Conference on Cluster Computing. Washington D. C., USA:IEEE Press,2015:527-528.
[17] 何震苇,严丽云,李慧云. 基于开源 PaaS 技术的互联网业务平台自动部署方案[J]. 电信科学,2015,31(10):172-179.
[18] JARAMILLO D, NGUYEN D V, SMART R. Leveraging microservices architecture by using Docker technology [J]. IEEE SoutheastCon,2016,16(5):1-5.
[19] CALINCIUC A, SPOIALA C C, TURCU C O, et al. OpenStack and Docker: building a high-performance IaaS platform for interactive social media applications[C]// Proceedings of 2016 International Conference on Development and Application Systems. Washington D. C., USA: IEEE Press,2016:287-290.
[20] Google. Network policies[EB/OL]. [2017-05-10]. http://Kubernetes.kansea.com/docs/user-guide/networkpolicies.
[21] MEDEL V, RANA O, BAÑARESJ Á, et al. Adaptive application scheduling under interference in Kubernetes[C]// Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing. Washington D. C., USA: IEEE Press,2016:426-427.

编辑　吴云芳

(上接第 52 页)

[14] 周　余,都思丹. ARM11 MPCore 性能分析与优化研究[J]. 南京大学学报(自然科学版),2009,45(1):5-10.
[15] NAKASHIMA S, NAITO T. Program optimization method, program optimization program, and program optimization apparatus: US 14/799, 625 [P]. 2015-07-15.
[16] 张俊卫,王　晶,张伟功,等. 基于大数据的高能效数据中心服务器研究[J]. 计算机工程,2017,43(8):74-81.
[17] 辛海洋,彭　飞. 一种基于 ARM + FPGA 的无人机地面测控系统设计与实现[J]. 电子世界,2017(15):129.
[18] 刘永刚. 基于 ARM 的汽车导航系统设计[J]. 山东工业技术,2016(8):282.
[19] 任　瑾,龙小丽,张晓亚. 基于 ARM 和 ZigBee 技术的智能家居系统的设计[J]. 自动化应用,2017(8):49-51,54.
[20] 杜　琦,姜　浩,李　宽,等. 面向 ARMv8 64 位多核处理器 QTRSM 的实现[J]. 计算机工程与科学,2017,39(3):451-457.
[21] 刘　婧,王天成,王　健,等. 基于指令模板的通用处理器约束随机指令生成方法[J]. 计算机工程,2015,41(10):309-313.
[22] HOFMANN J, FEY D, RIEDMANN M, et al. Performance analysis of the Kahan-enhanced scalar product on current multi-corecore and many-core processors[EB/OL]. [2016-03-09]. https://arxiv.org/abs/1505.02586.
[23] 董亚卓,常　歌. 面向 FPGA 设计的类 C 语言及其关键技术研究[J]. 网络安全技术与应用,2017(8):50-52.
[24] 迟利华,刘　杰. 基于安腾微处理器的程序性能优化与分析[J]. 计算机工程与科学,2011,33(9):42-47.
[25] 余超君. 基于 CK810 的 Android 系统移植研究[D]. 杭州:浙江大学,2014.
[26] 张仁高,郑启龙,王向前,等. 基于依赖环问题的改进软流水框架[J]. 计算机工程与应用,2017,53(17):65-69.
[27] 张　琦,陈玉荣,李建国,等. 基于多核系统的视频特征提取程序并行化及性能优化方法[J]. 中国科学院研究生院学报,2011,28(4):531-547.

编辑　顾逸斐