第 35 卷 第 3 期 计 算 机 工 程 2009 年 2 月
Vol.35 No.3 Computer Engineering February 2009
·安全技术· 文章编号：1000—3428(2009)03—0189—03 文献标识码：A 中图分类号：TP309

基于龙芯 SIMD 技术的 AES 加解密优化

顾丽红，魏海蕊
(中国石油大学(华东)计算机与通信工程学院，东营 257061)

摘 要：高级加密标准 AES 是 Linux 系统中安全网络协议采用的主流的加解密算法。该文通过分析 AES 加解密算法，结合龙芯平台的体系结构特征，提出基于多媒体指令扩展(SIMD 技术)优化 AES 性能的方法。优化前后的安全文件传输协议 Sftp(AES 加解密)数据传输结果表明，龙芯 SIMD 技术优化 AES 算法减少了加解密时间，有效地提高了 Sftp 的网络传输速率。

关键词：龙芯；SIMD 技术；AES 加解密；Sftp 协议

Optimization of AES Encryption and Decryption Based on SIMD Technology for Loongson

GU Li-hong, WEI Hai-rui
(School of Computer and Communication Engineering, University of Petroleum (East China), Dongying 257061)

【Abstract】The advanced encryption standard is the mainstream encryption algorithm in the security network protocol used in Linux system. Through analysis of AES encryption algorithm, with the architecture characteristics of the Loongson platform, the method for performance optimization of AES based on the expansion of multimedia instructions (SIMD technology) is proposed. Before and after optimization, the results of data transmission using security file transfer protocol Sftp (AES encryption) shows that the Loongson SIMD optimization has reduced the time of encryption and decryption and has effectively improved the Sftp network transmission rate.

【Key words】Loongson; SIMD technology; AES encryption and decryption; Sftp protocol

1 概述

高级加密标准(Advanced Encryption Standard, AES)是一种可以用于保护电子数据的加密算法，它基于排列和置换运算，且为对称、迭代密钥分组。与公共密钥加密使用密钥对不同，对称密钥密码使用相同的密钥加密和解密数据，通过分组密码返回的加密数据位数与输入数据相同。迭代加密使用一个循环结构，在该循环中重复置换(permutations)和替换(substitutions)输入数据。数据加密标准 AES 可以使用 128 位、192 位和 256 位密钥，并且用 128 位分组加密和解密数据。AES 加密过程主要包括 3 个部分：圈数，轮变化和密钥扩展。AES 每一个圈变换由以下 3 个层组成：非线性层 SubByte，线行混合层 ShiftRows 和 MixColumns，密钥加层 AddRoundKey。不同的分组长度，其对应的轮变化次数是不同的，它直接取决于密钥长度，本文以 128 位密钥为例，其轮变化次数为 10。密钥扩展由以下 3 层组成：RotWord，S 盒变换 SubWord，轮常数表 Rcon[][1]。

为提高网络数据传输的安全性，Scp, Sftp 和 Smtp 等多数网络协议在数据传输过程中均采用 OpenSSL 加密传输。基于 OpenSSL(用于实现安全套接层协议(SSL v2/v3)和传输层安全协议(TLS v1)以及形成一个功效完整的通用加密库)的 libcrypto.so 库集成了 AES 算法的相应函数，其中加密函数 AES_encrypt()和解密函数 AES_decrypt()使用官方 AES 加解密算法进行数据加解密。本文以 Sftp 为例，针对龙芯平台与 X86 平台，从数据传输速率和 CPU 利用率进行量化对比，提出了结合龙芯 SIMD 技术优化 AES 加解密算法的方法。实验结果表明该方法有效地提高了龙芯平台的 CPU 利用率和传输速率，对提高基于 AES 算法的安全网络性能有指导意义。

2 高级加密算法 AES

2.1 密钥扩展 KeyExpansion 的构建

AES 加密和解密算法使用了一个由种子密钥字节数组生成的密钥调度表 *w*[*row*][1]。密钥扩展由以下 3 层组成：*w*[*row*]行位移置换操作 RotWord：*w*[]循环左移 8 位；S 盒置换操作 SubWord；轮常数表数组 Rcon[]异或(xor)操作。KeyExpansion 密钥扩展流程如图 1 所示。

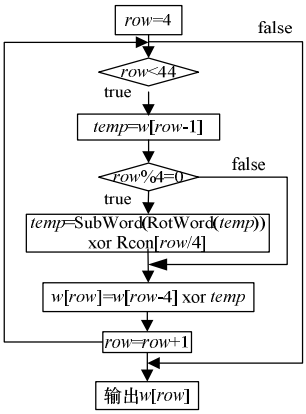

图 1 KeyExpansion 密钥扩展流程

基金项目：国家“863”计划基金资助项目“低成本先进计算机”(2006AA010201)

作者简介：顾丽红(1970－)，女，副教授、硕士，主研方向：嵌入式系统平台，分布式 Web 服务系统；魏海蕊，硕士研究生

收稿日期：2008-10-15 **E-mail：**weihr@lemote.com

密钥长度为 128 位，当 4≤*row*＜44 时，*w*[*row*-1]赋值给 *w*[*row*]；若行数 *row* 是 4 的倍数时，则执行 SubWord, RotWord 置换操作和 Rcon 异或操作；最后执行异或操作。

2.2 128 位密钥加密

执行加密传输时，首先拷贝 128 位输入数据到 ***state*** 4×4 Byte 矩阵中。AES 算法的主循环对 ***state*** 矩阵执行 4 个不同的迭代和置换操作，在规范中称为非线性层 SubByte 单字节代替操作；线行混合层 ShiftRows 置换操作和 MixColumns 列混合变换替换操作；密钥加层 AddRoundKey 轮密钥加层异或操作[2]。

SubBytes 例程是一个替代操作。如图 2 所示，SubBytes 例程将 ***state*** 矩阵中的每个字节替换成一个由 Sbox 对应的新字节。

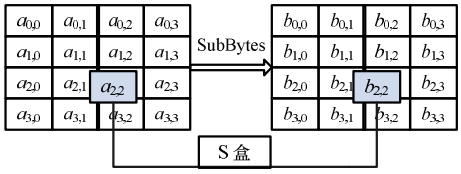

图 2 SubBytes 字节替换操作图例

ShiftRows 例程是行位移置换操作。如图 3 所示，ShiftRows 例程将 ***state*** 矩阵中的字节向左旋转。不同行执行不同位数的移位。

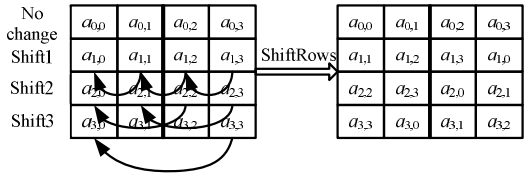

图 3 ShiftRows 行位移置换操作图例

MixColumns 列混合变换是一个替代操作。其用 ***state*** 字节列的值进行数字域加和域乘的结果代替每个字节(^代表异或操作，*代表域乘)。

$$state[a,b] = (0x02 * state[a,b]) \wedge (0x03 * state[(a+1)\%4,b]) \wedge (0x01 * state[(a+2)\%4,b]) \wedge (0x01 * state[(a+3)\%4,b])$$

AddRoundKey 轮密钥加层执行异或操作。其用密钥调度表对 ***state*** 矩阵实行逐字节异或操作，并用轮密钥表 *w*[*b*,*a*] 异或输入 *state*[*a*,*b*]。

$$state[a,b] = state[a,b] \wedge w[(round * 4) + b, a]$$

2.3 高级加密算法 AES 加密的复杂度分析

AES 算法中的 aes_core.c 分别使用 S 盒与 Si 盒构造了 4 个加密查询表格 *Te* 和 4 个解密查询表格 *Td*(*代表域乘操作):

$$Te0[x] = S[x] * [02,01,01,03]$$
$$Te1[x] = S[x] * [03,02,01,01]$$
$$Te2[x] = S[x] * [01,03,02,01]$$
$$Te3[x] = S[x] * [01,01,03,02]$$
$$Td0[x] = Si[x] * [0e,09,0d,0b]$$
$$Td1[x] = Si[x] * [0b,0e,09,0d]$$
$$Td2[x] = Si[x] * [0d,0b,0e,09]$$
$$Td3[x] = Si[x] * [09,0d,0b,0e]$$

并将最终的计算结果存放在 code segment 中的 *Te*, *Td* 中。因此，加解密执行时只需要进行查表和简单的异或操作就可以完成加解密的任务，避免了复杂的 MixColumns 列混合变换的域加和域乘运算。但算法在加解密过程中依然频繁执行移位、异或和内存访存操作。以加密为例:

$$t0 = Te0[s0 >> 24] \wedge Te1[(s1 >> 16) \& 0xff] \wedge Te2[(s2 >> 8) \& 0xff] \wedge Te3[s3 \& 0xff] \wedge rk[4]$$

其中，*t*0 为下次访问内存时的下标，将替换 *s*0；*rk*[4]为第 5 行密钥；^为异或操作；*t*1, *t*2, *t*3 同样处理替换 *s*1, *s*2, *s*3，然后循环执行操作 9 次。数据的计算量大，对内存数据的存取频繁，资源消耗大。

3 Sftp 的性能分析和优化必要性

目前，AES 是安全网络协议采用的通用加解密算法，应用广泛。Scp, Sftp, Smtp 等多数安全网络协议在数据传输过程中均采用 AES 加密传输。本节以 Sftp 为例，分析了 Sftp 的工作流程、执行时间分布得出，加解密耗时成为龙芯平台 Sftp 的性能瓶颈及优化的必要性。

3.1 系统的网络性能分析

为了评测龙芯 2 号网络性能，实验首先测试了龙芯 2 号与 X86 平台上系统在未开启 NFS, SMB/CIFS 等网络文件系统服务的数据传输吞吐量，确定网络性能的好坏与系统平台的无关性。龙芯 2 号参数为主频：666 MHz；内存：256 MB；内核：2.6.18.1ict；系统版本：Debian GNU/Linux 4.2。X86 参数为主频：2 660 MHz；内存：512 MB；内核：2.6.18-6-686；系统版本：Debian GNU/Linux 4.0。

为保证数据准确性，龙芯 2 号与 X86 平台的 FTP 服务端采用 Proftpd-1.3.1，客户端采用 Ncftp-3.2.0(相同配置的 X86 PC)。对于 FTP 文件传输，龙芯 2 号和 X86 平台的下载速度分别为 10.24 MB/s, 10.45 MB/s。这表明：龙芯 2 号的网络性能比较理想。

3.2 龙芯 2 号 Sftp 的性能分析

3.2.1 Sftp 原理及工作流程分析

Sftp(Secure file tranfer program)是 ssh 内含的协议，只要 sshd 服务器启动即可用，其本身不需要 FTP 服务器启动。Sftp 分为 2 部分，服务端的 Sftp-server 及 Sftp-client，此有别于 FTP over SSH2。

图 4 详细介绍了 Sftp 的工作流程：Sftp-client 向远程主机上传文件时，Sftp 首先创建 1 个子进程，这个子进程执行程序 ssh(sshd 为其 daemon 进程)。进程 Sftp 和 ssh 之间通过管道通信。Sftp 负责从本地主机读取文件，然后将文件数据写入管道；ssh 则从管道读取要传送的数据，然后通过网络套接字发送到服务器端。在发送之前，ssh 要使用 AES 加密算法 AES[3-4]对数据进行加密。Sftp 从远程主机下载文件时，首先龙芯 2 号服务器创建 Sftp-server 进程(Sftp 协议的服务端程序)。随后，ssh 被调用(ssh 进程并非被系统直接调用，而是作为 Sftp-server 的一个子程序从中调用)。ssh 进程和远程客户端建立连接，接收加密数据并解密后写入管道，Sftp-server 从管道读取文件写入龙芯 2 号和 X86 服务器的磁盘中。

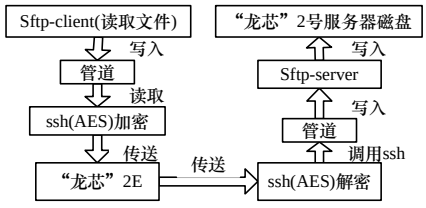

图 4 Sftp 的工作流程

3.2.2 龙芯 2 号 Sftp 执行时间分析

实验分析发现，服务器端 Sftp 在上传过程的时间分配上，ssh 进程占用了 90%多；下载过程中，ssh 进程占用的比例也高达 88%。故实验使用 Oprofile 的周期事件重点对龙芯 2 号

Sftp 数据传输时 ssh 进程函数进行采样分析。默认采样频率为 300 00 次/周期。

由表 1 测试数据分析可知：进程 ssh 有 60%的时间消耗在对加解密库 libcrypto.so 操作中，且加解密函数 AES_encrypt,AES_decrypt 占用系统资源比例最大。由此得出优化加解密函数成为提高加密数据传输性能的首选方向。

表 1　Sftp 下载时 ssh 进程函数执行时间分布情况

文件名	函数名	占用比例/(%)
libcrypto.so	AES_encrypt	32.27
	AES_cbc_encrypt	17.63
	md5_block_host_order	14.20
libc-2.3.6.so	memcpy	5.76
ssh	ssh	1.72
vmlinux	src_unaligned_dst_aligned	8.84
	both_aligned	2.81
	rtl8139_poll	2.04
	move_128bytes	1.30

3.3　龙芯处理器 SIMD 技术简介

龙芯 2 号处理器是 64 位、4 发射、乱序执行的高性能通用处理器，它兼容 MIPSⅢ指令集[5]。龙芯 2 号增强型处理器采用 SIMD(单指令多数据流)技术，通过扩展多媒体指令集实现了对多媒体应用的指令级支持。龙芯多媒体指令支持对 8 个字节、4 个 16 位数、2 个 32 位数或 1 个 64 位数(龙芯 2 号体系结构下，每个字是 32 位)的并行定点操作[6]。

SIMD 技术是一种典型的并行数据处理技术。处理器把输入的数据分解为多个较短的数据，然后由单指令并行操作，从而提高处理海量、可分解数据的能力。每条多媒体指令同时对几个数据单元进行操作，从而实现空间上的并行性[7]。龙芯 SIMD 技术适合于密集型数据处理的特点为提高 AES 加解密效率提供了可能。下节描述采用龙芯 SIMD 技术优化 AES 加解密算法的具体设计和实现。

4　AES 加解密算法

从上面的实验结果和分析可以得出，libcrypto.so 中的 AES 加密函数 AES_encrypt()、解密函数 AES_decrypt()处理输入数据时，其数据处理有以下几个突出特点：并行性运算操作程度高；运算性质大多是运算密集型的循环运算；运算数据量大，访存操作频繁。由于 AES 加解密算法本身已得到高效的优化，但数据加解密运算的 CPU 资源占用依旧太大，因此根据加解密过程的分析和龙芯 SIMD 技术的特点，在保证代码正确性的前提下，提高数据单元处理的并行性，消除或尽量减少各种处理器延迟。

因为解密是加密的逆过程，所以本文以加密过程进行说明。优化中采用 C 语言中嵌入汇编的方法。龙芯体系结构下，多媒体指令由作为 1 号协处理器的 FPU 执行，因此多媒体寄存器和通用寄存器之间进行数据传送需要使用 dmtc1, dmfc1 此类指令；而多媒体寄存器与内存之间的数据交换则要使用 ldc1, sdc1 此类指令。内存读取单元相对于其他执行单元是最耗时的。作为一种优化策略，把数据尽可能地存在寄存器中处理，处理完毕后一次写入内存是提高效率的重要手段。在 C 语言实现加解密算法 AES 中，4 字包数据 *s*0, *s*1, *s*2, *s*3(共 128 位，单字 32 位)放入一个 32 位数组中：*a*[0]=*s*0; *a*[1]=*s*2; *a*[2]=*s*1; *a*[3]=*s*3，可以一次读入 *s*0, *s*264 位多媒体寄存器，这样即可减少读取内存的次数，且对 *s*0, *s*2; *s*1, *s*3 进行分组，便于分组数据的高低位重置操作。

128 位包数据读入多媒体寄存器后，为减少指令的重复执行，把 4 字包数据分组：*s*0, *s*2；*s*1, *s*3，如图 5 所示。优化后的加密过程不再以字为单位逐个异或、置换、迭代操作，而是以双字为单位同时处理 2 组数据。由于通用寄存器不能施加于多媒体寄存器上，以往数据的传送需先将多媒体寄存器中的数据移到通用寄存器中，运算结束后再移回至多媒体寄存器中，但此类做法影响到龙芯 2 号处理器的执行效率。为避免多媒体寄存器与通用寄存器之间频繁的数据交换，使用龙芯 2 号可以直接施加于多媒体寄存器的操作指令来提高多媒体的性能。在优化中，首先从内存中读入双字保存在多媒体寄存器中组成一个 64 位数 *s*2*s*0，然后 pshufh 指令实现高低位重置，psrlw 指令实现包数据分别右移操作；同时，*s*0*s*2 则执行 psrlw, fand 包数据分别进行右移操作、逻辑与操作。然后对生成的 2 组 64 位数分别进行拆包，拆包得出的单个字节作为访问 *Te* 的下标。最后，从 *Te* 中访问得到的数据重新以 64 位为单位读入多媒体寄存器中执行异或操作。同样，*s*1, *s*3 的操作如上，经过 4 次相同的处理过程即完成了一个循环的地址对齐存取操作。生成的最后结果 *t*0, *t*2 和 *t*1, *t*3 又作为下次循环操作的始值。

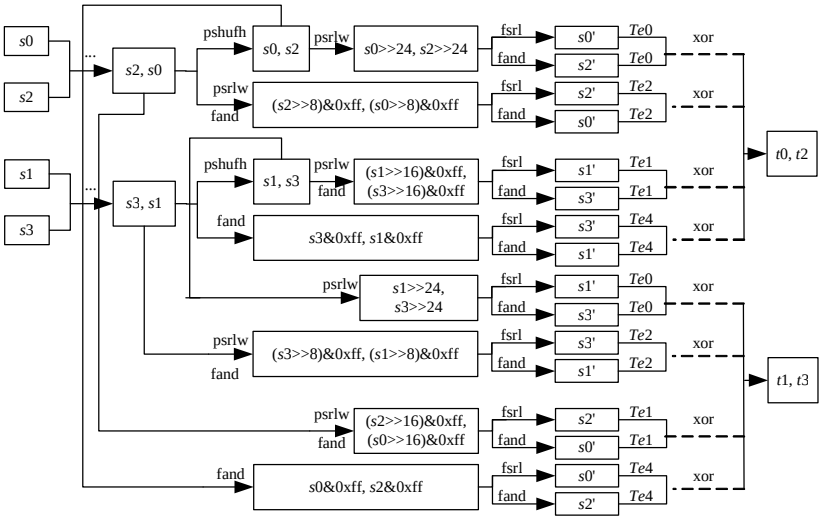

图 5　AES 加密优化的流程

优化 libcrypto.so 库后，测试龙芯 2 号 Sftp 数据传输时的吞吐量，图 6 为优化前后 Sftp 数据传输吞吐量对比结果。

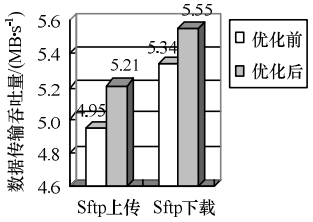

图 6　AES 算法优化前后龙芯 2 号 Sftp 吞吐量对比

从图 6 中可以看出，经过龙芯 2 号 SIMD 技术优化 AES 加解密函数后的 Sftp 传输效率有了比较明显的提高。但由于 AES 加解密计算在执行中频繁从内存存取数据，耗时严重，相比非加密 FTP 传输，在减少内存存取操作的优化上，龙芯仍有较大的提升空间。

(下转第 221 页)