动态二进制翻译器 QEMU 中冗余指令消除技术研究

宋 强 陈香兰 陈华平
(中国科学技术大学计算机科学与技术学院 安徽 合肥 230026)

摘 要 计算机体系结构的不断发展,使得代码迁移工作变得尤为重要,在这种背景下,二进制翻译技术应运而生。二进制翻译技术使得在已有的体系结构下生成的可执行文件自动迁移到新的体系结构中成为可能。以龙芯 2F 处理器为硬件平台,研究二进制翻译器 QEMU 中冗余指令的删除优化技术,使用代码活性分析方法来降低代码膨胀度,提高执行效率。该优化技术带来的优化效果超过其自身开销,具有实际优化价值。

关键词 二进制翻译 冗余指令消除 动态优化 虚拟机

中图分类号 TP314 **文献标识码** A

OPTIMIZATION TECHNIQUE OF REDUNDANT INSTRUCTIONS ELIMINATION IN DYNAMIC BINARY TRANSLATOR QEMU

Song Qiang Chen Xianglan Chen Huaping
(*School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, Anhui, China*)

Abstract The continuous development of computer architecture makes code migration become particularly important. In this context, the binary translation technology comes into being. Binary translation technology makes it possible that the executable file generated on an existing architecture is automatically migrated to a new architecture. In this paper, the authors present methods for redundant instructions elimination to reduce code expansion and improve the efficiency on Godson 2F CPU as the hardware platform. It is such a technique that its optimization achievements exceed its cost, thus it achieves practical optimization values.

Keywords Binary translator Redundant instructions elimination Dynamic optimization Virtual machine

0 引 言

随着新体系结构的研究和开发,代码迁移问题更加凸显出来。为了使新的体系结构得到推广流行,必须要对其提供相应的软件支持;但由于新体系结构尚未流行,又导致很少有软件厂商愿意为其开发应用软件。在这样的大背景下,以二进制翻译为核心的虚拟机技术获得了广泛的关注与研究。目前,无论是 PC 市场还是服务器市场都基本上被 X86 体系结构的处理器所占据,RISC 架构的处理器要想获得广泛的应用,首先就要实现对 X86 的兼容,二进制翻译技术使 X86 兼容的实现变为可能。

二进制翻译是将源体系结构上的二进制程序转化为目标体系结构上的二进制程序的过程。当只拥有源体系结构上的二进制程序而没有源代码时,使用二进制翻译技术可以实现快速的迁移,当然这种迁移会以损失部分性能为代价。虚拟机就是以二进制翻译为核心技术来实现的,主要可分为两类:进程虚拟机和系统虚拟机。进程虚拟机实现了为不同于用户主机的体系结构而开发的应用程序,可以按照和主机系统上的所有其他程序相同的方式安装使用以及交互,它为用户应用程序提供了虚拟的应用程序接口 ABI(Application Binary Interface)环境;系统虚拟机实现了为不同于用户主机的体系结构而开发的操作系统可以安装并运行在用户主机上,并可以在其中安装运行针对这种操作系统的应用程序,它提供了一个完整的系统环境,包括中央处理器以及周边设备,在此环境中属于多个用户的许多进程可以并存。

QEMU 系统是目前较为先进的多源多目标的二进制翻译系统,而且它同时支持进程级虚拟和系统级虚拟两种工作模式,具有高速、跨平台、开源、易移植等优点[1]。现有的一些研究多是进程级虚拟机优化,并以 QEMU-v0.9 为基础,该版本的 QEMU 尚未引入 TCG(Tiny Code Generator)技术,而是将源二进制代码翻译为中间代码,再根据中间代码拷贝已编译好的二进制代码,第二步并不是严格的动态翻译。龙芯 2F 是一款 64 位、四发射、乱序执行的 RISC 处理器,实现 MIPS III 指令集[2]。本文是在龙芯 2F 上利用 QEMU-v0.12(该版本引入了 TCG 技术,它将中间 TCG 代码翻译为目标代码时也是完全的动态翻译)作为将 X86 指令翻译到 MIPS 指令的系统级二进制翻译器运行 Windows XP 操作系统,针对龙芯 2F 的特点进行移植,并对 QEMU 翻译过程中产生的冗余指令进行删除优化研究[3]。由于目标是对全系统模拟进行优化,而系统中运行的程序多种多样,导致现有的热路径分析等统计代码块执行情况的方法无法使用,这类利用程序的统计流和控制流进行优化的方法更适合于进程级虚拟机的优化,对于全系统模拟则需要更通用更细粒度的优化方法。

收稿日期:2011 - 08 - 29。国家重大专项"核心电子器件、高端通用芯片及基础软件产品"(2009ZX01028-002-003-005)。宋强,硕士,主研领域:二进制翻译及优化。陈香兰,博士。陈华平,教授。

1 QEMU 翻译系统简介与移植

QEMU 系统可以实现将 arm(同时支持 arm7 与 arm9)、X86、MIPS 下的 ELF 格式的可执行文件翻译到中间代码,然后再翻译到目标指令集体系结构 ISA(Instruction Set Architecture)上执行,我们将其移植到龙芯平台会使用到将 X86 指令翻译成 MIPS 指令的部分。

1.1 QEMU 翻译系统结构框架

QEMU 系统由控制核心、前端解码器、中端分析优化器、后端翻译器等几个模块组成,如图 1 所示。其中控制核心负责整个系统的流程,前端解码器负责将源平台二进制代码(X86 代码)翻译成中间代码 TCG,中端分析优化器利用代码的活性分析去除 TCG 代码中部分冗余指令,后端翻译器负责将 TCG 代码翻译成目标平台的二进制代码(MIPS 代码),以在宿主计算机(龙芯 2F 处理器)上运行。

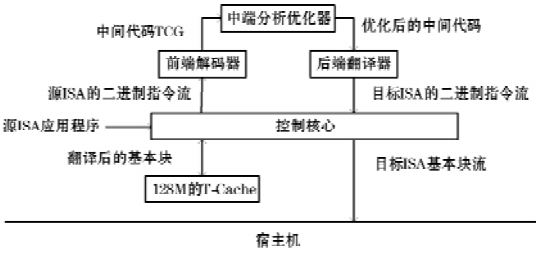

图 1 QEMU 系统结构框架

1.2 QEMU 的移植

龙芯 2F 处理器使用了三个独立的 Cache:一级指令 Cache、一级数据 Cache 以及二级混合 Cache,其中除一级指令 Cache 外都采用写回策略。执行一个基本块(一段以跳转指令结束的顺序代码)的步骤如下:首先将翻译后的基本块加载到二级 Cache,然后再将二级 Cache 中的数据加载到一级指令 Cache,最后执行一级指令 Cache 中的基本块。由于龙芯 2F 中的一级指令 Cache 没有采用写回策略,那么当我们在翻译以外的环节修改了翻译后的基本块内容时,就必须手动刷新 Cache,否则会导致一级指令 Cache 中的数据与主存不一致。因此将该版本的 QEMU 移植到龙芯 2F 上的关键就是:当发生翻译外的环节修改了基本块内容的情况时,加入刷新 Cache 的代码。

2 利用活性分析技术去除 TCG 中的冗余指令

活性分析技术最早应用于编译器,编译器前端通过大量的临时变量将源代码翻译为中间表示,接着分析中间代码来确定哪些临时变量同时被使用,如果一个变量中保存的值将会被使用,那么就说该变量是活跃的。将该技术引入到二进制翻译的优化环节,可以消除一些冗余的死代码,降低代码膨胀率,从而提高执行效率[5]。QEMU 系统的中端优化器正是根据 TCG 代码的特点,利用活性分析技术删除了大量冗余代码。

2.1 活性分析背景

活性分析的关键是确定变量活性信息,为确定变量活性首先需要明确以下概念:

pred[n]:节点 n 的所有前驱节点集合;

succ[n]:节点 n 的所有后继节点集合;

def[n]:通过节点 n 进行赋值的变量的集合,称为变量定值集合;

use[n]:节点 n 在赋值号右边出现的变量的集合,称为变量引用集合;

活性信息:如果有一条直接路径从变量所在的边指向该变量的使用而不通过任何定义,那么该变量在这条边上是活跃的。如果一个变量在一个节点的任何一条流入边上均活跃,则变量在该节点处处于 live-in;如果变量在任何一条流出边上均活跃,则变量在该节点处处于 live-out。

有了上述概念,我们就可以根据节点的 def 和 use 集合,使用以下方式计算活性信息:

(1) 如果一个变量属于 use[n],那么它在节点 n 处是 live-in 的,即如果一条语句使用了一个变量,那么该变量在进入这条语句时就是活跃的;

(2) 如果一个变量在节点 n 处是 live-in 的,那么在属于 pred[n]的所有节点中则是 live-out 的;

(3) 如果一个变量在节点 n 处是 live-out 的,而且不在 def[n]内,那么该变量在 n 处是 live-in 的,即如果变量 a 的值在语句 n 的结尾被使用而且 n 没有为其赋值,那么 a 值在进入 n 的入口时就是活跃的。

上述三点的变量集合等式如下:

$$in[n] = use[n] \cup (out[n] - def[n])$$

$$out[n] = \bigcup_{s \in succ[n]} in[s]$$

2.2 TCG 代码冗余消除

QEMU 系统由源基本块生成的 TCG 中间代码中的 temps 变量具有如下特点:在且仅在该基本块内有效,即本基本块最后一条语句的 out 集合为空。这样,有了初始条件,只要逆向遍历 TCG 基本块,并根据上节的两个等式,有如下结论:

针对第 i 条语句,若 def[i]中的所有元素均不属于 out[i],即存在下述关系:

$$def[i] \cap out[i] = \Phi$$

则这条指令属于冗余指令,将其修改为空指令,在后端翻译器处理过程中直接跳过翻译。

3 后端代码冗余的发现与消除优化

QEMU 系统生成的后端 MIPS 代码的冗余是影响代码膨胀率的最直接因素,原 QEMU 系统没有针对 MIPS 代码的冗余进行任何消除优化。

3.1 后端代码冗余发现

对 QEMU 启动 Windows XP 过程中生成的日志文件进行分析,发现其中含有大量的类似表 1 所示的代码对。

表 1 匹配模式

模式 1	模式 2
move rd1,rs1 opc rd2,rs2,imm	move rd1, rs1 opc rd2,rs2,rt2

第一条指令即是 MIPS 的汇编指令 ADDU rd, rs, 0;第二条指令中的 opc 可为 ADDIU,ANDI,ORI,SLL,SRA,SUBU,ADDU,SRL,XORI。

针对这种类型的模式,当满足条件:rd1 == rd2 并且 rd1 == rs2 时,可以在将 TCG 代码翻译为 MIPS 代码的过程中删除第一

条指令并将第二条指令改为 opc rd1, rs1, imm/rt,这样就消除了一条冗余指令。

3.2 冗余 MOVE 指令删除算法

为消除后端代码中冗余的 move 指令,需要首先声明两个变量 g_rs、g_rd 用来保存 move 指令中的源与目标,以及一个标志 g_moveFlag 用来标识上一条指令是否为 move 指令。算法流程如下:

```
if (g_moveFlag && rd == g_rd && rs == g_rs)
    删除上一条 move 指令
    将 rs 设置为 g_rd
生成当前指令并写入缓存
if 当前指令是 move 指令
    设置 g_moveFlag 为真,即相对于下一条指令的上一条指令为 move 指令
    将当前 move 指令的 rd,rs 分别保存到 g_rd, g_rs 中
else
    设置 g_moveFlag 为假
```

3.3 冗余 MOVE 指令判定的改进算法

将活性分析引入寻找冗余 move 指令的过程中可以进一步消除冗余指令,这样我们将着眼于整个代码块而不是代码块中相邻的两条指令。该算法在遇到 move 指令时不会将其写入后端的 MIPS 代码缓存中,而是将其添加到一个与该基本块相关的链表中,我们在翻译后面的代码时,会出现下面几种情况:

(1) 当前指令是对链表中存在的 move 指令的目标操作数的引用操作,则将当前指令的对应操作数修改为对应的 move 指令的源操作数,写入 MIPS 缓存;

(2) 当前指令是对链表中存在的 move 指令的目标操作数的定值操作,则将当前指令写入 MIPS 缓存,并将对应的 move 指令从链表中去除;

(3) 当前指令是对链表中存在的 move 指令的源操作数的定值操作,则将链表中的 move 指令写入到 MIPS 缓存,再写入当前指令,最后将对应的 move 指令从链表中去除;

(4) 当前指令是跳转指令,则将链表中的所有 move 指令写入 MIPS 缓存,再写入当前指令,最后清空链表。

表 2 是截取 QEMU 日志文件中的一段目标代码,其中原始程序段中共有三条 move 指令,如果使用上一节的简单删除算法,只有第一条 move 指令与其后面的 addi 指令符合匹配规则,可将其去除。但如果使用本节提到的改进算法,则可以去除第一条和第二条 move 指令,表 3 阐述了应用改进算法的翻译过程。

表 2　使用冗余删除的效果

原始指令片段	4.2 的删除算法	4.3 的改进算法

move s2, s1
andi s2, s2, 0xffff
lw s3, 12(s0)
move s4, s2
move s5, s3
subu s3, s4, s3
addu s4, s3, s5
	andi s2, s1, 0xffff
	lw s3, 12(s0)
	move s4, s2
	move s5, s3
	subu s3, s4, s3
	addu s4, s3, s5
		andi s2, s1, 0xffff
		lw s3, 12(s0)
		move s5, s3
		subu s3, s2, s3
		addu s4, s3, s5

表 3　应用改进算法的翻译过程

指令序列	应用改进算法的翻译过程	翻译后的链表状态
move s2, s1	move 指令,加入链表	move s2, s1
andi s2, s2, 0xffff	满足情况 1:对链表中的 move s2, s1 的目的操作数 s2 引用,用 s1 替换　满足情况 2:对链表中的 move s2, s1 的目的操作数 s2 定值,将该 move 指令从链表中去除	空
lw s3, 12(s0)	原样翻译	空
move s4, s2	move 指令,加入链表	move s4, s2
move s5, s3	move 指令,加入链表	move s4, s2 move s5, s3
subu s3, s4, s3	满足情况 1:对链表中的 move s4, s2 的目的操作数 s4 引用,将 s4 替换为 s2 满足情况 3:对链表中的 move s5, s3 的源操作数定值,将该 move 指令写入目标代码缓存并从链表中去除	move s4, s2
addu s4, s3, s5	满足情况 2:对链表中的 move s4, s2 的目的操作数 s4 定值,将该 move 指令从链表中去除	空

4 测试结果

由于现有的龙芯盒子是 2F 处理器,主频为 800Hz 左右,内存 512MB,在 QEMU 上运行的 Windows XP 系统上不便于测试,所以采用使用 QEMU 虚拟一个 Linux 内核,并在其上运行 nbench 测试集进行测试的手段,实验结果如表 4 所示。

表 4　实验结果

nbench 中的测试用例	优化前/每秒迭代次数	简单删除算法/每秒迭代次数	改进算法/每秒迭代次数
NUMERIC SORT	37.985	38.2664	39.556
STRING SORT	1.404	1.40318	1.40636
BITFIELD	9.8858E+6	9.93082E+6	1.01982E+7
FP EMULATION	2.21858	2.24856	2.2779
FOURIER	630.85	638.642	632.75
ASSIGNMENT	0.462054	0.470014	0.480616
IDEA	83.5004	83.2284	86.7698
HUFFMAN	26.5648	26.6588	27.4786
NUEURAL NET	0.273988	0.279004	0.275588
LU DECOMPOSITION	11.9642	11.965	12.034

(下转第 98 页)

图 4 公交车辆实时监控(监控单车)

图 5 公交车辆实时监控(单条线路)

图 6 公交车辆历史轨迹回放

上述图中小箭头图片表示在当前地图范围内的公交车辆,箭头方向代表车辆运行的方向,箭头颜色代表车辆状态,蓝色表示正常,红色表示车辆处于失去联络,绿色表示车辆处于脱线。箭头旁边的数字表示公交车辆的车号。车辆轨迹回放中的红色点表示时间段内小车的轨迹点。

5 结 语

利用 ArcGIS Server 实现的公交车辆监控系统,能够实现对公交车辆的实时准确定位与监控,辅助调度,使车辆管理可以科学有序,提高了公交公司的运营效率和效益,同时也提高了乘客的满意度。系统在国内一些城市投入了使用,取得了良好的社会和经济效益,具有非常广阔的前景。

参 考 文 献

[1] 刘志强. 智能公交调度监控系统的研究与开发[D]. 山东:山东大学,2005.

[2] ESRI. ArcGIS Server:ESRI's Enterprise GIS Application Server[M]. USA:esri,2004.

[3] ESRI. ArcGIS Server Concepts and Terminology[EB/OL]. http://edn. esri. com.

[4] 贺海靖,王东浩,吕斌,等. AJAX 结合 WebGIS 在监控系统中的应用[J]. 自动化技术和应用,2008,27(4):133 - 134.

[5] 沈百玲. ArcGISServer 体系结构和开发简介[M]. 北京:ESRI 中国有限公司,2004.

[6] 薛蕾,李林,王农鹤. 基于 ArcGIS Server 和 AJAX 的 WebGIS 系统开发[J]. 农业网络信息,2008(7):24 - 26.

[7] 姚真凯. 瓦片地图技术在水利 WebGIS 系统中的开发研究与应用[J]. 浙江水利科技,2009(6):40 - 42.

[8] 李春立,曾致远,徐学军. 基于 ArcGIS Engine 的车辆监控系统[J]. 计算机工程,2006(24):257 - 259.

(上接第 69 页)

根据表 3 求得简单删除算法与改进算法的加速比,如图 2 所示。

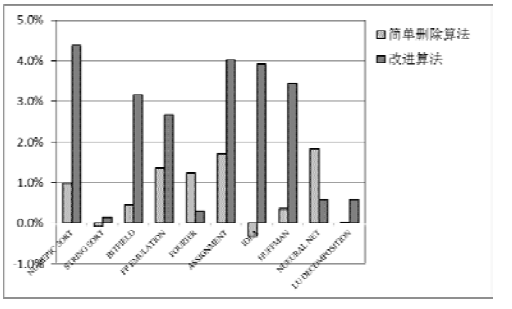

图 2 加速比

由图 2 可以看到对于生成的 MIPS 代码进行冗余 move 指令消除优化普遍可以提高 2% 到 5% 的性能。总体上,对于检测冗余 move 指令所耗费的时间低于消除冗余后所带来的性能提升,具有实际的优化价值。

5 结 语

由于动态二进制翻译器 QEMU 采用二次翻译策略,即先将源二进制代码翻译为中间 TCG 代码,再将 TCG 代码翻译为目标二进制代码,而这两次翻译都会产生一些冗余指令,从而增大了代码膨胀度,QEMU 本身对 TCG 代码中的冗余指令进行了消除优化,却没有处理目标代码中的冗余。本文针对龙芯 2F 处理器的特点对 QEMU 进行了移植,并将代码活性分析技术引入到对目标代码的冗余指令消除优化中,从而降低代码膨胀度,提高效率。

参 考 文 献

[1] Bellard F, Usenix. QEMU, a fast and portable dynamic translator[C]//USENIX Association Proceedings of the FREENIX/Open Source Track,2005:41 - 46.

[2] 蔡嵩松,刘淇,王剑,等. 基于龙芯处理器的二进制翻译器优化[J]. 计算机工程,2009,35(7).

[3] 王丽一,文延华. 动态二进制翻译中的冗余 load 删除优化技术[J]. 计算机应用与软件,2008,25(6).

[4] Yabin. An Optimization Approach for QEMU[C]//2009 1st International Conference on Information Science and Engineering (ICISE 2009), 2009:129 - 132.

[5] Probst M, Krall A, Scholz B. Register liveness analysis for optimizing dynamic binary translation[C]//Ninth Working Conference on Reverse Engineering, Proceedings, 2002:35 - 44.

[6] 史辉辉,管海兵,梁阿磊. 动态二进制翻译中热路径优化的软件实现[J]. 计算机工程,2007,33(23).