

嵌入式系统新型动态内存管理机制的研究

王 欣^{1,2}, 邱 昕¹, 慕福奇¹, 冷永清¹

(1 中国科学院 微电子研究所, 北京 100029; 2 中国科学院大学, 北京 100049)

摘 要: 动态内存管理是影响嵌入式系统性能的重要因素之一, 其必须满足 3 个性能需求: 平均分配时间短, 最差情况下分配时间可预计, 内存碎片少. 利用嵌入式应用整个运行周期内的统计数据, 提出一种基于不同内存块生命期的动态内存管理机制(Smart Memory allocator)的改进算法, 在不同内存区采用不同的分配, 释放和合并策略. 最后将改进的分配器移植到 ZYNQ7000 进行试验测试, 测试结果表明本方法拥有更快的内存分配释放速度, 并且减少了内存碎片率.

关键词: 生命期; 内存管理; 内存碎片; ZYNQ7000

中图分类号: TP368.1

文献标识码: A

文章编号: 1000-7180(2017)08-0066-04

Study of New Dynamic Memory Management Mechanism in Real-time Embedded System

WANG Xin^{1,2}, QIU Xin¹, MU Fu-qi¹, LENG Yong-qing¹

(1 Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China;

2 University of Chinese Academy of Sciences, Beijing 10049, China)

Abstract: Dynamic Memory Management Mechanism is one of the most important factors which affecting the performance of embedded systems. There are three performance requirements must be met: average allocation time is short, worst-case allocation time is predictable, and memory fragmentation is few. Then conducting the experimental test on the ZYNQ7000 platform. At last, the results show that this method improved the speed of memory allocation and release, with reducing memory fragmentation rate.

Key words: life-spans; memory management; memory fragment; ZYNQ7000

1 引言

随着智能硬件、物联网行业的迅猛发展, 嵌入式系统在各个领域都得到了广泛的应用, 动态内存管理算法作为决定系统性能的重要因素之一, 也成为了嵌入式领域一个重要的研究课题.

近年来, 研究人员提出了多种动态内存管理算法, 大概分类如下^[1-2]: (1) 顺序查找算法; (2) Buddy 算法(伙伴算法)^[3]; (3) 位图匹配算法, 使用位图来标记堆的使用情况; (4) TLSF 匹配算法^[4].

这些常用算法都有各自的优缺点和适用场景,

但都没有忽略动态内存分配与释放的统计数据信息, 在动态内存管理中利用统计数据来提高内存管理性能正成为的一个重要趋势^[5].

基于此点, Ramakrishna^[6]提出了 smart DM allocator(以下简称 Smart 分配器), 该分配器将内存堆划分为 2 个区域, 通过基于过去数据的启发式算法来预测某个大小内存块的生命期长短, 根据预测结果在相应的内存区进行分配, 其中生命期短的内存块即为频繁申请释放的内存块. 但是, 该分配器在内存释放时对长短生命期的内存块都使用了立即合并的策略, 从而增加了短生命期内存块被频繁分割

收稿日期: 2016-11-23; 修回日期: 2016-12-28

基金项目: 中科院国防科技创新基金(CXJJ-16M124); 国家自然科学基金青年科学基金(61501455); 北京市自然科学基金面上项目(4162068)

合并的概率,影响了内存分配效率。

本文在该分配器的基础上,根据不同生命期的内存块申请释放行为的不同,改进了空闲内存块的合并策略,并在 Xilinx 的 ZYNQ7000 系列平台上进行了测试和验证,证明了改进后的分配器具有更好的实时性和更低的碎片率。

2 动态内存分配器的性能需求

嵌入式系统的动态内存分配器有 3 个主要的性能需求:平均分配时间短,最差情况下分配时间可预计,内存碎片少。

为了满足上述需求,Smart 分配器采用了类似 TLSF 算法的位图和离散空闲链表的数据结构,算法复杂度为 $O(1)$,该算法下内存平均分配时间短,最差情况下分配时间也可以预计。

而内存碎片的产生往往是因为动态内存分配算法不能有效利用空闲内存,过多的内存碎片有时会耗尽有限的内存资源,使得任务不能得到实时执行。关于内存碎片,Masmano^[4]等给出了如下的结论:

- (1)最佳匹配可以产生较小的碎片;
- (2)内存块被释放后应该立即合并产生大的空闲内存块;
- (3)应该尽量重用会被频繁申请释放的内存块。

所以 Smart 分配器采用最佳匹配策略,同时考虑到短生命期的内存块使用时间较短,且释放后又被申请的概率比较大,重用该类型内存块可以减小内存碎片,所以 Smart 分配器根据内存块大小和统计数据来预测不同大小内存块的生命期,并对短生命期的内存块采取不分割的策略减少分配时间,本文改进后的分配器又采用短生命期内存块不合并的策略,减小内存块被频繁分割合并的概率。

3 Smart 分配器的改进与实现

改进后的分配器根据生命期将内存划分为两个区域,对短生命期的内存块采取不分割和不合并的策略,对长生命期的内存块采取精确分割和立即合并的策略,从而减小内存碎片。同时,采取位图和离散空闲列表保证分配时间的快速和最差情况下分配时间可预计。

3.1 内存块生命期

Smart 分配器是通过基于过去行为的启发式算法来预测内存块的生命期,根据生命期来采取不同的策略。生命期表示内存块被应用占用的时间,可以通过内存块大小与对应分配次数相结合来预测。

所以此分配器通过统计对应大小内存块的分配次数和释放次数来预测内存块的生命期,当分配次数与释放次数之差小于阈值时,就认为是短生命期,则对应大小的内存块就会在短生命期内内存区进行分配。

3.2 分配器数据结构

Smart 分配器类似于 TLSF 算法,采用二级位图来管理。如图 1 所示,定义了两套两级链表结构,分别对应长短生命期的内存块,第一级链表将内存区划按照 $2^k (5 \leq k \leq 31)$ 划分,二级在第一级的基础上再线性划分,并通过位图表示对应链表是否为空。

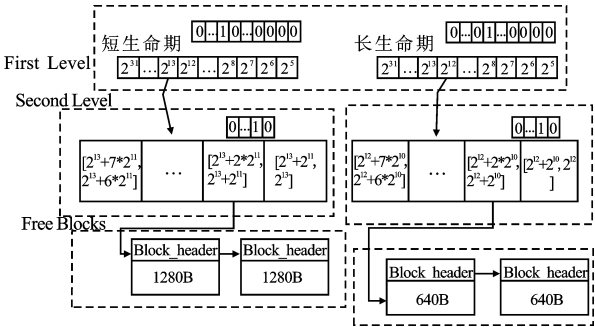


图 1 Smart 分配器的数据结构

在内存分配和释放过程中,会根据收集的统计数据改变预测位图(PredMask),该变量被用来预测内存块的生命期。比如计算出一级索引为 N ,则如果 PredMask 的第 N 位为 1,则为短生命期,反之为长生命期。如图 2 为根据统计信息更新 PredMask 的流程。

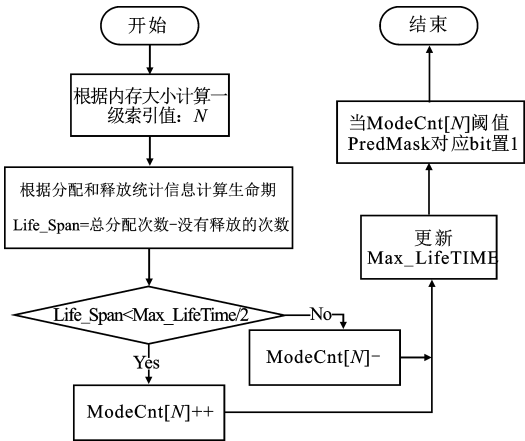


图 2 根据统计更新 PredMask 过程

3.3 分配器实现

由数据结构可知,该分配器最重要的参数是:一级索引(First_level_index)、二级索引(Second_level_index)、位图、预测位图(PredMask)。

当申请内存块大小为 Size 时,索引参数可由式(1)、(2) 算出:

$$N = \log_2 \text{Size} \tag{1}$$

$$M \geq \frac{\text{Size} - 2^N}{2^{N-L}} \tag{2}$$

式中, N 代表一级索引; M 代表二级索引; 2^L 代表二级线性分割的个数.

当嵌入式应用申请内存块时,利用索引值与 PredMask 来预测出内存块的生命期,如果为短生命期,则将分割标志置为不可分割,对应的位图找到对应的空闲链表进行内存分配,如果空闲链表非空且有合适大小的内存块,则立即分配;如果没有合适的内存块或者链表为空,则二级索引或者一级索引向上取值,直到有合适的空闲内存块,当一级索引值变化时,则分割标志变为可分割;如果为长生命期,则一直为可分割,分配过程如图 3 所示.

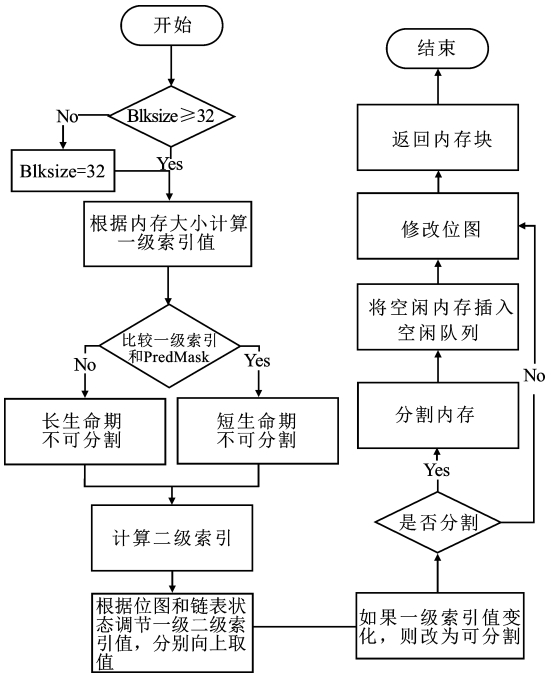


图 3 内存分配过程

释放时,如图 4 所示,读取内存块头的长短生命期内存块标志,如果是短生命期内存块,则采取不合并策略,找到相应的空闲链表插入;如果为长生命期链表,则采取立即合并策略,然后再插入相应空闲内存链表.

4 移植与测试

4.1 分配器移植

本文将该分配器移植到了 ZYNQ7000 开发平台上,其中 ZYNQ7000 平台^[7]是 Xilinx 公司推出的一款集成 ARM Cortex-A9 双核处理器和 FPGA 可编程逻辑资源的 SoC. 测试只在 arm-A9 单核上运行,且关闭了 MMU,主频为 666 MHz, L1 L/D cache 均为 32 K, DDR 512M, 软件环境则为 FreeRTOS^[8]. 因为内存分配算法的性能要求主要在分配速度和内存碎片率上,所以测试主要集中在这两个指标.

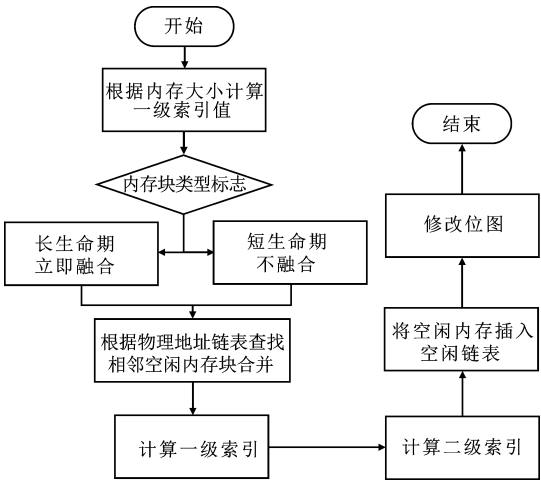


图 4 内存释放过程

辑资源的 SoC. 测试只在 arm-A9 单核上运行,且关闭了 MMU,主频为 666 MHz, L1 L/D cache 均为 32 K, DDR 512M, 软件环境则为 FreeRTOS^[8]. 因为内存分配算法的性能要求主要在分配速度和内存碎片率上,所以测试主要集中在这两个指标.

4.2 性能分析测试

参考文献[9]设计测试程序,将用户请求的内存大小分为 4 个区间(单位是 byte),分别为[0, 256]、[256, 512]、[512, 1 024]、[1 024, 2 048],在每个区间范围内产生随机值,并频繁分配和释放随机值大小的内存,每组测试 100 次,并测量内存分配和释放时间,取平均值.

在上述测试程序中,参考文献[10],统计在每次分配时请求的内存块大小 m ,实际分配的内存块大小 n ,则内部碎片率为 $(n-m)/m$. 当 $m=n$ 时,代表内部碎片为 0; $(n-m)/m$ 越大,代表内存碎片越多.

4.3 测试结果分析

测试结果如表 1~3 所示.

表 1 内存分配时间测试结果 μs

算法	最小值	最大值	平均值
Smart	1.407	1.755	1.628
改进后 Smart	0.948	1.161	1.032

表 2 内存释放时间测试结果 μs

算法	最小值	最大值	平均值
Smart	1.020	1.026	1.023
改进后 Smart	0.582	0.591	0.587

表 3 碎片率测试结果 %

算法	最小值	最大值	平均值
Smart	4.7	9.0	7.2
改进后 Smart	4.1	8.0	6.1

因为在每个区间都是进行 100 次的分配释放,所以相当于是对短生命期内存块的操作,由于改进后的 Smart 分配器在释放时没有进行立即合并,所以在后续的分配中减小了因没有合适大小内存块再去查找分割的概率,而且不进行立即合并,在释放时不需要查找相邻空闲内存块,只进行插入操作,所以分配和释放时间都要小于未改进前,与测试结果一致。

对于碎片率,因为改进后的分配器减少了分割次数并且保留了被频繁使用的内存块,所以在申请该类内存块时,该类内存块大小对应的空闲链表非空的概率增大,从而减小了被分配内存块大小与请求大小的差值,即内部碎片率减小,与测试结果一致。

5 结束语

本文介绍并改进了 Smart 分配器,改进后的分配器根据统计数据预测内存块生命期,并针对不同生命期内存块的不同特点来决定分配和合并策略,从而提高内存的利用率,提高了短生命期内存块被重复使用的概率,并通过实验测试证明了改进后的 Smart 分配器拥有很好的实时性和更低的碎片率,可以很好地满足资源有限并对实时性要求很高的嵌入式系统对动态内存分配器的需求,增强了嵌入式系统的健壮性。

参考文献:

- [1] 池元武. 嵌入式实时操作系统动态内存管理优化方案的研究[D]. 上海:上海交通大学, 2011.
- [2] Diwase D, Shah S, Diwase T, et al. Survey report on memory allocation strategies for real time operating system in context with embedded devices[EB/OL]. [2016-08-08]. <https://www.researchgate.net/publication/265806496>.

- [3] 姜艳, 曾学文, 孙鹏, 等. 实时嵌入式多媒体系统模糊阈值合并内存管理算法[J]. 西安电子科技大学学报(自然科学版), 2012, 39(5):174-180.
- [4] Masmano M, Ripoll I, Crespo A, et al. TLSF: a new dynamic memory allocator for real-time systems[J]. 2004, 16:79-88.
- [5] Ramakrishna M, Kim J, Lee W, et al. Smart dynamic memory allocator for embedded systems[C]// International Symposium on Computer and Information Sciences. IEEE, 2008:1-6.
- [6] Lee J, Yi J. Improving memory efficiency of dynamic memory allocators for real-time embedded systems[J]. Etri Journal, 2011, 33(2):230-239.
- [7] Crockett L H. The Zynq book : embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC [M]. Strathclyde Academic Media, 2014.
- [8] 杨继兰. 嵌入式实时操作系统 FreeRTOS 在 x86 上的移植[D]. 西安:西安电子科技大学, 2011.
- [9] 刘林, 朱青, 何昭晖. FreeRTOS 内存管理方案的分析与改进[J]. 计算机工程与应用, 2016, 52(13):76-80.
- [10] Zorn B, Grunwald D. Evaluating models of memory allocation[J]. Acm Transactions on Modeling & Computer Simulation, 1992, 4(1):107-131.

作者简介:

王欣 男, (1992-), 硕士研究生. 研究方向为嵌入式系统. E-mail: wangxin3@ime.ac.cn.

邱昕 男, (1980-), 博士, 副研究员. 研究方向为无线通信.

慕福奇 男, (1957-), 研究员. 研究方向为无线通信.

冷永清 男, (1981-), 博士, 助理研究员. 研究方向为微波固态功率技术、高效率功率放大器、微波发射机和接收机.