ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.12, No.30, October 2016
E-mail:kfyj@dnzs.net.cn
http://www.dnzs.net.cn
Tel:+86-551-65690963 65690964

基于multimap映射的动态内存分配算法探究

曹海涛,余永红

(安徽财经大学 管理科学与工程学院,安徽 蚌埠 233030)

摘要:对多种不同的动态内存分配算法的特点与优劣进行对比、分析,在兼顾效率和内存碎片率指标的要求下,提出了基于multimap映射的动态内存分配算法。该算法以内存块的大小作为键,内存块的地址信息作为值,以键值对的形式存储内存块的地址,并在内存块实体的首部与尾部添加标识信息。为检验算法效果,设计了多组数据对新算法和现有经典内存管理算法效率进行比较,实验结果表明新算法在降低时间开销,保留较大连续空间,减少内存碎片等方面具有较明显的改善。

关键词:动态内存分配;内存碎片;边界标识法;multimap

中图分类号:TP311 文献标识码:A 文章编号:1009-3044(2016)30-0222-02

DOI:10.14004/j.cnki.ckt.2016.4054

Algorithm of Dynamic Memory Allocation Base on Multimap

CAO Hai-tao, YU Yong-hong

(.School of Management and Engineering, Anhui University of Finance & Economic, Bengbu 233030, China)

Abstract: Under the consideration of efficiency and memory fragments indictors, a variety of dynamic memory allocation algorithms were compared and analyzed, and a new dynamic memory allocation based on multimap was proposed. The algorithm put the size of the memory block as the key and the address of the memory block as the value, used the key-value pair to store the address of memory block, and added the boundary tag at both the head and tail of memory block. In order to verify the performance of new algorithm, multiple sets of experiment data were tested on new algorithm and existing classic algorithms of memory allocation.The experiment results show that the new algorithm has obviously improved on time cost, keeping large contiguous space and memory fragments reduction.

Keywords: dynamic memory allocation; memory fragmentation; the boundary tag method; multimap

动态内存分配使用灵活,广泛地应用于实际开发中。在程序的运行中,更是频繁使用动态的内存分配方式,因此对动态内存分配的算法效率有着较高要求。目前动态内存分配算法主要的两类为,基于顺序搜索的动态内存分配算法[1,2,3,4]与基于索引搜索动态内存分配算法[1-2,5-7]。

首次适应算法[1-4]。将可用内存块安置在空闲链表中,分配时总是从空闲链表首部开始查询,并将首个满足的内存块分配出去,该算法有效的保留了高地址端的大空闲区。缺点:在空闲链表的首部(物理空间的首部)容易产生大量的小碎片,且增加了下一次的查询次数。

循环首次适应法[1-2]。与首次适应法相似,循环首次适应法可以将产生的小碎片均匀的分布,有效地缩短了下次查询的时间,但是仍然没有有效地解决存储空间碎片化的问题,而且会导致缺少大的空闲分区。

最佳适应法[1-2,4]。最佳适应法可以有效保留较大的内存块,以满足用户对较大连续存储空间的需求,但是在内存中会产生更小的碎片。

最差适应法[1-2]。将空闲链表中的内存块按照大小顺序,逆序排列(由大至小)。可以有效地减少碎片的产生,但是内存分区会缺乏较大的连续存储空间。

快速适应法[1-2]。将空闲块按照大小进行分类,同时建立一张管理索引表,此算法在查询内存块时效率较高O(1),主要缺点在于合并内存块时,要遍历所有的内存块时间复杂度为O(n)。

伙伴系统[1-2,5-7]。已分配或未分配的内存区大小均为2的k次幂,其时间性能比快速适应算法差,但是由于采用了索引搜索算法,比顺序搜索算法好。而其空间性能,由于对空闲分区进行合并,提高了内存空间的可使用率,故优于快速适应法,比顺序搜索法略差。

哈希算法[1]。哈希算法就是利用哈希快速查找的优点,哈希算法具有较高的查找效率,在实际动态内存分配中使用广泛,但是在合并内存时需要逐个查找可用内存块并检查是否为可用的邻区内存。因此,实际开发中亟待需求高效的内存分配与合并算法。基于顺序搜索的动态内存分配算法的特点是管理简单,但效率不高。基于索引搜索的动态内存分配算法的特点是,分配效率较高,但回收效率较低。

收稿日期:2016-08-20

基金项目:基于Android平台的教室使用情况监测软件(项目编号:201510378570)。

作者简介:曹海涛(1995—),男,安徽宿州人,本科,主要研究方向:程序设计;余永红(1967—),男,江西南昌人,教授,博士,主要研究方向:数据库理论与安全,程序设计。

两类不同的分配算法在不同的实际应用中有着各自的优势，但是在大多数的应用开发中普遍存在大内存块不易保留，内存碎片化，合并内存块时间开销大等问题[8-9]。multimap映射算法就是指将内存块的大小作为multimap的键，对应的值会记录内存块的地址信息以供分配[10-11]。该算法可以较好地解决上述问题，可以满足不同应用的需求。

1 改进的内存分配算法

基于multimap映射的动态内存分配算法采用C++的STL中提供的multimap关联容器，允许多个相同的键存在，键与值为一一对应关系。算法以内存块的大小作为键，内存块的地址信息作为值，以键值对的形式存储内存块的地址，并在内存块实体的首部与尾部添加标识信息。

1.1 内存分配算法

算法使用STL中multimap关联容器来存储Mcb（自定义的内存控制块）的地址信息。multimap提供了一种可以有重复键值的STL map类型。multimap<int, Mcb*>作为为容器的类型，以内存块的大小作为键，便于查询内存块。当用户申请内存时，通过multimap提供的方法去查询可供用户使用的内存块。使用lower_bound(k),upper_bound(k)可以得到一组指向键为k的元素的迭代器（在有多个相同键值对的情况下，分别指向元素的首端和尾端），如不存在则两个迭代器均指向首个大于键的值元素[11]。

为了避免产生过小的碎片，定义一个常量MIN_SIZE。如果切割内存块后会产生小于MIN_SIZE的碎片，则不再经行切割，直接分配供用户使用。

1.2 内存释放、合并算法

内存块在使用后，为了避免产生内存大量的碎片。需要在释放内存块时，尽可能地将物理空间连续的内存块合并到一起。在合并内存块时，需要遍历空闲内存以查找左邻块、右邻块，时间复杂度为O(n)[12-14]。为提高合并效率，本次算法采用边界标识法[2]。在内存块的首部和尾部添加一些信息（首部添加Mcb类型，尾部添加unsigned类型记录此内存块的大小）。添加边界标识的内存块信息分布，如下图1所示：

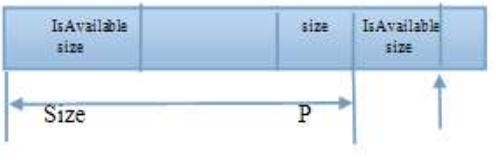

图1 边界标识内存块

图1中的指针p，为返回给用户的可用内存块首地址。IsAvailable用于标识内存块的可用状态，size的大小为包括Mcb与后边界标志在内的大小。通过边界标识法可使合并内存块的复杂度为O(1)。

算法步骤：

在合并内存块时会查找左邻块、右邻块，当释放指针p的内存区域时会发生以下步骤：

（1）将指针P回退到内存块的首部，得到Mcb *mcb（使其指向内存控制块的首部）。

（2）检查当前指针是否在堆区域，指针不可越界。

（3）将mcb->availeble = 1变为可用状态。

（4）将指针p继续回退sizeof(unsigned)个字节，读取左邻区的长度。再将指针p回退size-sizeof(unsigned)字节，经过强制类型转换，读取Mcb并判断是否为空闲状态，若是则将其从空闲multimap记录中删除，与左邻块合并后，并将左邻块Mcb中的size改为size+mcb->size，将合并后内存块的尾部标识改为size+mcb->size，将指向向左合并后的内存块，重复执行步骤（4）；否则直接跳至步骤（5）。

（5）将指针p向右偏移mcb->size个字节，将指针强制类型转换为Mcb，读取available。若右邻块为空闲块，则将其从空闲multimap记录中删除，并与p当前指向的内存块合并，将mcb->size更新为mcb->size+size，右邻块的尾部标志的值改为size+mcb->size，重复执行（5）步骤；否则跳至步骤（6）。

（6）将合并后的空闲块添加到multimap中。添加至multimap中的内存块可能是与左邻块合并后的内存块，与右邻块合并后的内存块，或者是未合并的内存块。

边界标识法使得在合并内存时，可以通过偏移指针直接获取物理相邻的内存可用信息，所以可使合并内存块的复杂度为O(1)。

2 实验对比分析

数据准备：为尽可能的体现每种分配算法的特征（如，体现循环首次适应的特征，应当至少将内存块的分配循环一次），本次数据元素的选取，将参考实际应用中使用内存块大小的概率，产生若干数据元素以组成数据集合。本次实验环境为：Windows8操作系统，开发工具为vc++6.0。

在实际使用中，申请的内存块大小近似服从正态分布。本次模拟中，可供分配的总空间大小为10000个字节。参考实际使用情况，设定正态分布中期望与方差，可以得到符合实际且较为合理的数据集合[15-16]。

假设数据元素 $x \sim N(500, 32400)$，$\mu = 500$，$\sigma = 180$

$P\{0 < x \leq 200\} = \Phi((200-500)/180) - \Phi(-500/180) = 0.0457$

$P\{200 < x \leq 400\} = \Phi((400-500)/180) - \Phi((200-500)/180) = 0.2427$

$P\{400 < x \leq 600\} = \Phi((600-500)/180) - \Phi((400-500)/180) = 0.4176$

$P\{0 < x \leq 200\} = P\{800 < x \leq 1000\} = 0.0457$

$P\{200 < x \leq 400\} = P\{600 < x \leq 800\} = 0.2427$

按照概率随机产生23个样本，作为申请内存的指定大小，并从中随机选取6个元素用于释放内存[16]。通过计算可以得到不同大小的内存块使用的概率，以及取样数量，如下表1所示。

表1 内存块取样表

尺寸范围	P(在此区间的概率)	N1(样本个数理论值)	N2(样本个数实际值)
(0,200]	0.0457	1.052	2
(200, 400]	0.2427	5.586	6
(400, 600]	0.4176	9.69	9
(600, 800]	0.2427	5.586	5
(800, 1000]	0.0457	1.052	1

通过取样得到的数据集合为：

Malloc {72, 111, 387, 712, 381, 414, 710, 528, 205, 604, 496, 304, 345, 840, 695, 588, 523, 433, 464, 621, 494, 509, 558}

Free{712, 528, 710, 496, 840, 621}

执行顺序：

分配内存时，按照Malloc集合中元素的顺序进行分配。在为528, 604, 345, 523, 621, 494大小的内存块分配内存后，按顺序分别释放Free集合中的元素。

对比不同分配算法下的实验结果，比较不同尺寸空闲块数量，以及空闲块最值可以得到图2、图3：

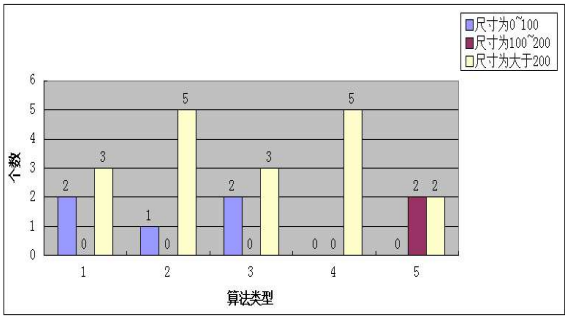

图2　不同尺寸空闲块数量的比较

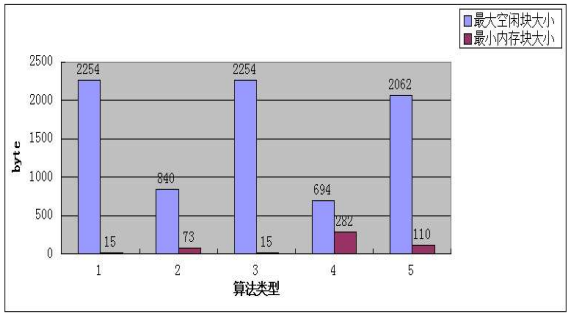

图3　空闲块的最值比较

注：P{0 < x ≤ 100} = 0.0132，此区间内存块被使用的概率很小，所以将大小在(0, 100]的内存块视为内存碎片。

编号1, 2, 3, 4, 5分别代表者首次适应法，循环首次适应法，最佳适应法，最差适应法，基于multimap映射算法。

通过图2可以看出，在不同尺寸空闲块数量的比较上，首次适应法与最佳适应法具有相同的结果。循环首次适应法与最差适应法产生大于200个字节的内存块数量较多，首次适应法、循环首次适应法、最佳适应法均产生了不同数量的内存碎片。multimap映射法没有产生内存碎片，且大于200个字节的内存块数目不多。

通过图3可以看出，首次适应法、最佳适应法在空闲块的最值比较上具有相同的结果，保留了较大的连续空间，同

时产生了难以利用的内存碎片。循环首次适应法、最差适应法未能保留较大连续空间，且循环首次适应法产生了内存碎片，但碎片尺寸大于首次适应法与最佳适应法所产生的碎片。multimap映射法保留了较大的连续空间，且未产生内存碎片。

由图2、图3的分析表明，multimap映射法在保留较大连续空间、减少内存碎片方面具有一定的优势。

不同的分配算法选取的数据结构可能存在一定的差异，通过分析不同算法所采用的数据结构以及具体操作的时间开销，并参考图2、图3的实验结果。可以对不同内存分配算法的效率进行比较，从而得到具有更低时间开销的分配算法。整理结果可得如下表2：

表2 不同内存分配算法的效率比较

算法类型	碎片数量	碎片情况	分配内存时间复杂度	回收内存时复杂度
首次适应法	2	集中在低地址端	O(n)	O(n)
循环首次适应法	1	存在于高地址端	O(n)	O(n)
最佳适应法	2	集中在低地址端	O(n)	O(n)
最差适应法	0	无	O(1)	O(n)
基于multimap映射算法	0	无	O(log n)	O(1)

由表2可以看出，在分配内存上最差适应法与multimap映射法具有较低的时间复杂度，分别为O(1)、O(log n)。在回收内存上multimap映射法的时间复杂度为O(1),优势较为明显。经综合比较可以发现multimap映射法在降低时间开销、保留较大连续空间、减少内存碎片方面具有较为明显的优势。

3结束语

对几种动态内存分配算法进行分析与对比，可以看出选取不同的数据结构会对效率产生显著地影响。multimap映射算法通过红黑树的组织存储形式，提高了内存块的分配与回收的效率，通过边界标识法极大地改善了内存块合并的效率。虽然边界标识法是以消耗一定空间的方式来提升速度，但是当消耗的空间所占有效空间的比例较小时，是可以接受的。总体上，multimap算法较好地解决了大内存块不易保留，易产生碎片，合并内存时间开销大等问题。

参考文献：

[1]汤小丹,梁红兵,哲凤屏,汤子瀛.计算机操作系统(第四版)[M].西安:西安电子科技大学出版社,2014.

[2]严蔚敏.数据结构C语言版[M].北京:清华大学出版社,2007.

[3]Weinstock C.Dynamic Storage Allocation Techniques[D].Carnegie-Mellon University, Pittsburgh,1976.

[4]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].上海交通大学软件学院, 2011.

[5]Peterson J L,Norman T A.Buddy Systems[J].Communications of the ACM, 2007, 20(6):421-431.

[6]姜立波.Linux内存管理分析与研究[D].电子科技大学,2011:46-58.

[7]王振江.提高堆数据局部性的动态池分配技术[J].计算机学报,2011(4):665-675.

[8]Stephenson C J. Fast Fits: New Methods for dynamic storage allocation[C]. Proceedings of the 9th Symposium on Operating Systems, ACM, 1983.

[9]McKusick M.K,Karels M J.Efficient Kernel Memory Allocation on Shared-Memory Multiprocessors. Proceedings of the Winter USENIX Technical Conference,1993.