Acta Informatica 14, 107–117 (1980)

Acta Informatica

Memory Fragmentation in Buddy Methods for Dynamic Storage Allocation

Allan G. Bromley

Basser Department of Computer Science,
University of Sydney, NSW 2006, Australia

Summary. Methods are presented for the estimation of bounds on the memory fragmentation in buddy methods of dynamic storage allocation. These bounds, when compared with the results of simulation studies, are found to be sufficiently precise to be of considerable practical utility.

Introduction

Buddy methods of dynamic storage allocation [1–6] have attracted considerable interest, principally because of the low overheads associated with block allocation and deallocation. As a class the buddy methods partition storage into a small number of allocatable block sizes. Requests for blocks of storage are satisfied by allocating a block of the smallest standard size sufficient to contain the block requested. As a consequence it is unnecessary to search lists of free blocks for one of a suitable size and the time overhead for block allocation is essentially constant, independent of the number of free or allocated blocks. Similarly, block deallocation is essentially a constant overhead as the partitioning scheme specifies the ways in which free blocks may be merged with adjacent (buddy) blocks to form a larger free block and again no search process is necessary.

Unfortunately buddy methods cannot fully utilise the available free storage. Since only a small set of block sizes are available for allocation some storage will, in general, be wasted as the requested block size does not fill the block allocated. This wastage, termed **internal fragmentation** is measured as the fraction of the total storage allocated that is so wasted. Additionally, the available storage may become excessively partitioned into small blocks so that a request for a larger block cannot be satisfied although the total free storage might be sufficient were it not so fragmented. This **external fragmentation** is measured (informally) as the fraction of the total storage that remains unallocated when first a request for a block of storage cannot be satisfied because the free storage, though sufficient in total, is fragmented. The **total fragmentation** is the proportion of the total storage which cannot be utilised for either of the above

0001-5903/80/0014/0107/$02.20

causes and is given, in obvious notation, by TOT = INT + EXT − INT.EXT. This total fragmentation is, in practice, the most important factor characterising the behaviour of a buddy system. It is the purpose of this paper to develop simple theoretical methods by which bounds on the total fragmentation can be estimated.

Four distinct buddy methods, differing in their memory partitioning schemes, will be considered in this paper. The simplest [4, 5] partitions the available storage into blocks which are always of size 2^k words for some k. Such a block can be split, if required, into two "buddy" blocks of size 2^{k-1} or, when free, can be merged with its unique buddy, also of size 2^k, into a single block of 2^{k+1} words. This **simple buddy** method is simple to implement. A second scheme, the **fibonacci buddy** method [3], partitions the storage into blocks with sizes proportional to successive fibonacci numbers f_k and splits such a block into a pair of unequal buddies with sizes proportional to f_{k-1} and f_{k-2}. The **weighted buddy** method [6] partitions storage into blocks of size 2^k or 3.2^k. Blocks of size 2^k may be split into buddies of size 3.2^{k-2} and 2^{k-2} whilst blocks of size 3.2^{k-2} may be split into 2^{k-1} and 2^{k-2}. Finally, the **variant buddy** method [1] is similar to the weighted buddy method except that adjacent blocks of size 2^k, which are not buddies but the fragments of two successive splits of a block of size 2^{k+2} first to $3.2^k + 2^k$ and finally to $2^{k+1} + 2^k + 2^k$, may be merged into a single block of size 2^{k+1}. A free block of size 2^k may thus be merged in several ways to form a larger free block and hence no longer has a unique buddy. Althgough the allocation and deallocation routines are more complex they still take a time which is independent of the number of free or allocated blocks and this method retains the desirable characteristics of the other buddy methods.

Simulation Studies

It is a simple matter to simulate the behaviour of the buddy methods as a basis for comparisons between the methods and with the theoretical methods described in this paper. The simulation closely follows those described previously [5, 6] and is repeated below.

Let TIME be initially zero and all of the memory initially be available.

P1. Advance TIME by 1.

P2. Free all blocks in the system that are scheduled to be released at the current value of TIME.

P3. Calculate two quantities: S, a random size, and T, a random "lifetime", based on some probability distributions.

P4. Reserve a new block of size S to be released at TIME + T. Return to step P1.

Two block size distributions were used in the simulations:

LIN. An integer chosen uniformly between 100 and 2,000.

INV. An integer chosen between 100 and 2,000 with a frequency inversely proportional to the size. This distribution provides a much greater frequency of requests for blocks of small size.

These distributions were changed slightly in the case of the fibonacci buddy method as shown in Table 1 to minimise differences between buddy methods

attributable to the "fit" of the upper and lower bounds to the available block sizes and thus to make the results more comparable. Since it is only our intention to compare simulation results with theoretical bounds, this adjustment is not important. The time distribution was a random integer chosen uniformly between 1 and 100 for both block size distributions.

Free blocks of each size were kept on independent queues; blocks were allocated from the head of the queue, deallocated blocks were returned to the tail of the queue. The data structure used to maintain the free lists of unallocated blocks is known [6] to have a small effect on the memory fragmentation but our theoretical models are not capable of representing this level of detail. In each allocated block one word was reserved for a usage indicator bit, a size indicator field, and any other indicator or pointer information required by the particular buddy method simulated.

The internal fragmentation was calculated when a steady state was reached. This was taken at TIME = 2,000 in all cases. The internal fragmentation is the ratio of the sum of the unused space in all allocated blocks (including the control word needed by the buddy methods) to the total of space in all allocated blocks.

Memory overflow occurs when a request for memory cannot be satisfied because only smaller blocks are available. No attempt is made to queue memory requests that cannot immediately be satisfied. The simulations thus most closely approximate a multiprecision arithmetic or symbolic algebra application rather than a multi-programming operating system or simulation language application where some parallel task might continue whilst one is held up waiting for storage to become available. In this simulation memory overflow was caused by ceasing to release blocks after TIME = 2,000 and the external fragmentation was taken as the ratio of the unallocated memory to the total memory size when overflow occurred.

Table 1 gives the results of the simulation studies. Each simulation was repeated 10 times and from these means and standard deviations for the internal, external, and total fragmentation were computed.

Internal Fragmentation

To predict the internal fragmentation we need to know the distribution of stored block sizes in the memory. Since we have assumed in our simulations that storage requests are not queued but are always satisfied immediately we can deduce the distribution of allocated block sizes from the distributions of requested block size and lifetime. In our simulations the lifetime is independent of the block size so at any moment the average distribution of stored block sizes is simply equal to the distribution of requested block sizes. The stored block size distributions are shown in Table 2 for the case of the variant buddy method.

The internal fragmentation is now readily predicted. For each possible requested block size the block size allocated is determined and hence the amount of storage that will be wasted including the control word in each allocated block used by the buddy method. A simple sum of these wasted

Table 1. Comparison of observations from simulation studies with predictions made using the methods in this paper

Model			Average internal fragmentation		Average external fragmentation				Total observed
Partitioning Scheme	Total memory	Request size range	observed	predicted	observed	lower bound	improved upper bound	upper bound	
Simple	131,072	100–2,000 (Lin)	25.4 ± 1.2	25.9	0.5 ± 0.4	0.0	0.0	0.0	25.7 ± 1.2
Buddy	131,072	100–2,000 (Inv)	28.1 ± 1.6	28.1	0.3 ± 0.3	0.0	0.0	0.0	28.3 ± 1.7
Fibonacci	126,336	100–1,620 (Lin)	20.1 ± 1.0	20.0	2.1 ± 1.3	0.5	2.3	27.5	21.8 ± 1.6
Buddy	126,336	100–1,620 (Inv)	21.5 ± 1.0	21.4	1.8 ± 0.8	0.8	2.0	27.0	23.0 ± 1.0
Weighted	131,072	100–2,000 (Lin)	14.6 ± 0.8	14.9	23.6 ± 4.0	21.3	28.3	41.7	34.8 ± 3.5
Buddy	131,072	100–2,000 (Inv)	15.7 ± 0.6	15.7	5.6 ± 2.2	2.2	6.4	43.0	20.4 ± 2.3
Variant	131,072	100–2,000 (Lin)	14.9 ± 0.9	14.9	7.5 ± 2.4	7.2	9.4	12.9	21.2 ± 2.0
Buddy	131,072	100–2,000 (Inv)	16.8 ± 0.5	15.7	1.9 ± 0.6	2.0	2.9	14.7	18.4 ± 0.8

Table 2. Sample block size distributions

Block size	Request frequency	
	LIN	INV
2,048	465	168
1,536	512	257
1,024	256	183
768	256	257
512	128	183
384	128	257
256	64	183
192	64	257
128	28	156
64	–	–

fragments, weighted by the assumed request distribution, divided by the average space allocated yields the internal fragmentation [3]. Table 1 shows the excellent agreement between predicted values of the internal fragmentation and the values observed in the simulation studies. It would, in principle, be a straightforward matter to determine higher moments of the internal fragmentation attributable to stochastic variations in the instantaneous distribution of allocated block sizes.

External Fragmentation

For the purpose of simulation studies it is usual to define the external fragmentation in the operational manner described above: allow the system to reach an equilibrium state at near full utilisation (by suitably adjusting the average life time of allocated blocks or the total storage available), then cease all deallocation of blocks (i.e., make the life time arbitrarily large) and, when first a block request cannot be satisfied, take the proportion of the total storage space that has not been allocated as the external fragmentation.

The rationale behind this operational definition is that the fragmentation may differ in some manner when there is no block deallocation but by starting from an equilibrium situation we minimise this effect (if any). Such a definition is scarcely satisfactory for theoretical use; it is more reasonable to suppose that the system is indeed in equilibrium at the point where external fragmentation is measured. In simulation studies we might be tempted to achieve this by increasing the life time of allocated blocks very slowly so that the system is always effectively in an equilibrium state. However, such an approach is unworkable in practice as the instantaneous number and distribution of allocated blocks may be arbitrarily rich in large blocks compared to the average request distribution. Hence, the more slowly we attempt to approach the final saturated equilibrium the more certain it is that overflow will occur with an atypical block distribution. This is of more than theoretical importance since the average number of allocated blocks is usually small (a few tens or hundreds) so the stochastic variations are significant.

Consequently it is difficult to find a definition of external fragmentation equally suitable for both theoretical and simulation studies. The issue has been side stepped in this paper. The theoretical analyses which follow suppose an equilibrium distribution for the stored block sizes and concern themselves only with counting the number of blocks of each size allocated and do not otherwise consider the structure imposed by the memory partitioning scheme. Only the distribution of allocated blocks is of significance, not (except for the purpose of estimating bounds) the sequence in which those blocks were allocated. The results obtained certainly justify this approach for a first order theory.

The simulation studies show that the external fragmentation varies considerably depending both on the partitioning scheme and on the request block size distribution. It is not difficult to see the cause of this. Consider the weighted buddy method and a request for a block of size 2^k. If the smallest suitable free block available is of size 2^{k+1} it will be partitioned twice to $3.2^{k-1}+2^{k-1}$, then $2^k+2^{k-1}+2^{k-1}$ and the first of these blocks used to satisfy the request. Unless the request size distribution favours very strongly requests for small blocks the two small blocks of size 2^{k-1} are unlikely both to be allocated before saturation occurs and will therefore contribute to the external fragmentation. The external fragmentation is then attributable to the tendency of the partitioning scheme to produce small fragment blocks as a side effect of allocating larger blocks but mitigated by any excess demand for small blocks over larger blocks. Informally this accounts well for the simulation results in Table 1.

There is a trade off apparent between internal and external fragmentation depending on the partitioning scheme used. Internal fragmentation is clearly minimised when the ratio of successive allocatable block sizes is near unity. However, this implies that the partitioning scheme always splits a block into two pieces, one of which is much smaller than the other and consequently the external fragmentation might be expected to be large. The variant buddy method attempts to compromise.

Lower Bound to Average External Fragmentation

As when predicting the internal fragmentation we shall assume a single distribution of stored block sizes which has been deduced from the requested block size and life time distributions. Bounds to the average external fragmentation can then be estimated by considering different sequences in which the stored blocks might have been allocated to make up the distribution of stored block sizes. If all larger blocks are requested before any smaller ones then there is the greatest possibility that small fragment blocks produced by the partitioning scheme when allocating the larger blocks can be utilised to satisfy later requests. This will give a lower bound to the average external fragmentation.

The computation is straightforward. Given the stored block size distribution multiply by a factor n, say, to give the number of blocks of each size allocated, on average, when saturation is reached. It is not necessary for either n or the number of blocks of any size allocated to be integers since only an extreme case is

considered as a bound to the actual numbers possible. The partitioning scheme indicates the number of blocks of the largest size available; allocate the number required by the stored block distribution and partition the remainder into smaller allocatable blocks. Repeat this process for successively smaller block sizes until at the smallest block size the remaining blocks cannot be further partitioned. At each block size the number of unallocated blocks must not be negative so an upper bound for the factor n is fixed. The smallest of these bounds is taken as the actual bound to n and the number of unallocated blocks of the smallest size may be computed and hence the external fragmentation. This method is essentially a counting operation which does not consider the structure of storage produced by the partitioning scheme. It is also a matter of indifference that in general the fragmentation would not consist entirely of blocks of the smallest possible size.

Consider, for example, the variant buddy method with the linear distribution of requests detailed in Table 1. A total storage of 131,072 words will yield 64 blocks of size 2,048 for which the request is proportional to $465n$ (all blocks from 1,536 to 2,000 inclusive since one control word in each block is assumed). The remainder after satisfying these results will be $64-465n$ blocks of size 2,048 which may be partitioned to yield the same number of blocks each of size 1,536 and 512. The request for blocks of size 1,536 is proportional to $512n$ so the remainder may be partitioned to yield $64-977n$ blocks each of size 1,024 and 512. However, each of the latter blocks will be adjacent to another block of size 512 produced in the first partitioning and may be merged to yield a total of $128-1{,}954n$ blocks of size 1,024 and leaving $512n$ blocks of size 512. This process may be continued with the results shown in Table 3.

Lower bound estimates computed in this manner are compared in Table 1 with the results of simulation studies. The measure of agreement, at least for the models considered, is remarkably good.

Table 3. Sample calculation of lower bound to average external fragmentation

Block size	Request frequency	Remainder	Bound on n
2,048	$465n$	$64-465n$	0.13763
1,536	$512n$	$64-977n$	0.06552
1,024	$256n$	$128-2{,}210n$	0.05792
768	$256n$	$128-2{,}466n$	0.05191*
512	$128n$	$256-4{,}548n$	0.05629
384	$128n$	$256-4{,}676n$	0.05475
256	$64n$	$512-9{,}160n$	0.05590
192	$64n$	$512-9{,}224n$	0.05551
128	$28n$	$1{,}024-18{,}348n$	0.05581
64	–	$2{,}048-36{,}632n$	–

$n = 0.05191$

$$\text{External fragmentation} = \frac{2{,}048 - 36{,}632n}{2{,}048} = 7.2\,\%$$

Upper Bound to Average External Fragmentation

By analogy with the preceding section the worst case, and an upper bound to the external fragmentation, is obtained when all smaller blocks are requested before any larger blocks, since we then have no opportunity for the small fragment blocks produced by the partitioning scheme when allocating larger blocks to be utilised to satisfy subsequent requests.

Again the computation is straightforward. Suppose the number of blocks of each allocatable size is given with an arbitrary multiplying factor n as before. An examination of the partitioning scheme will show the maximum number of blocks of each size which could be obtained. A simple division yields for each allocatable block size the proportion of the total storage which will be required to satisfy requests for blocks of that size. Since by allocating smaller blocks first there will be no interaction of these demands on the total storage, these proportions are summed to yield the total demand and n is determined by forcing this sum to unity. The external fragmentation is then determined as before.

Details of the calculation for the variant buddy method example previously discussed are given in Table 4.

Upper bound estimates computed in this manner are compared in Table 1 with the results of simulation studies. The upper bound estimates, though of theoretical interest, are generally too loose to be practical.

Table 4. Sample calculation of upper bound to average external fragmentation

Block size	Max. blocks obtainable	Request frequency	Proportion required
2,048	64	$465n$	$7.266n$
1,536	64	$512n$	$8.000n$
1,024	128	$256n$	$2.000n$
768	128	$256n$	$2.000n$
512	256	$128n$	$0.500n$
384	256	$128n$	$0.500n$
256	512	$64n$	$0.125n$
192	512	$64n$	$0.125n$
128	1,024	$28n$	$0.027n$
64	2,048	–	–
			$20.543n$

$n = 1/20.543 = 0.04868$

$$\text{External fragmentation} = \frac{2{,}048 - 36{,}632n}{2{,}048} = 12.9\,\%$$

Improved Upper Bound to Average External Fragmentation

Suppose now that requests for blocks of storage occur in a random sequence without bias to either larger or smaller blocks first. The average external

fragmentation will then only exceed the lower bound estimate by a small margin since requests for smaller blocks will, on average, be uniformly interspersed amongst requests for larger blocks and will simply consume the fragments formed when allocating the immediately preceding larger blocks. The only significant additional contribution to external fragmentation will come from those requests for larger blocks which follow the last request for a smaller block, since any fragments generated when allocating the larger blocks will not be consumed by smaller blocks and will contribute directly to the external fragmentation.

For each block size we consider the next smaller block size that can consume fragments generated in allocating the larger block size. To simplify the calculation the number of blocks of each size allocated is taken to be that given by the lower bound calculation. The number of blocks of larger size allocated after the last of the next smaller size that can consume their fragments can then be estimated. The fragments that could be generated in allocating these trailing larger blocks can then be calculated and are taken as external fragmentation additional to that given by the lower bound calculation.

Details of the calculation for the variant buddy method example previously discussed are given in Table 5. Partitioning the total storage into blocks of 2,048 words generates no smaller fragments so requests for blocks of this size, whether trailing or not, contribute nothing additional to the external fragmentation. For every block of size 1,536 words generated by the partitioning scheme one fragment of 512 words is produced. From the lower bound calculation we expect $m=26.58$ blocks of size 1,536 and $n=6.64$ of size 512 to be allocated. Thus $m/(n+1)=3.48$ blocks of size 1,536 may be expected to be allocated following the last of size 512 and to contribute the equivalent of 27.81 blocks of size 64 (the most convenient counting unit) to the total storage wasted.

Table 5. Sample calculation of improved upper bound to average external fragmentation

Block size	Request frequency	Trailing requests	Fragmentation (unit 64)	Wastage (unit 64)
2,048	24.136	–	–	–
1,536	26.575	3.477	8	27.813
1,024	13.288	–	–	–
768	13.288	3.075	4	12.298
512	6.644	–	–	–
384	6.644	2.709	2	5.417
256	3.322	–	–	–
192	3.322	–	–	–
128	1.453	–	–	–
64	–	–	–	–
				45.528

Additional external fragmentation = 45.528/2,048 = 2.2 %
Total external fragmentation = 9.4 %

This simple calculation may be expected to give an upper bound to the external fragmentation for several reasons:

(i) Only the next smaller block size is considered as consuming fragments from allocating larger block sizes. Still smaller blocks could consume some of these fragments and this effect, though small with the block size distributions we have used, would be to reduce the external fragmentation.

(ii) When partitioning storage to allocate a smaller block an unallocated block suitable for allocation to a larger following block may remain. In this case no fragments will be generated when allocating a trailing larger block. This possibility is avoided if the larger of two buddy blocks is always partitioned when a smaller block is to be allocated. Simulation studies [6] have shown that the effect of this is to increase the external fragmentation by about 3 %. Our calculation will therefore overestimate the external fragmentation by about this amount.

(iii) The requests prior to the trailing larger blocks will have a distribution richer in smaller blocks than that used in the lower bound analysis on which the present calculation is based. This higher proportion of smaller blocks will, in general, cause a better fit to the partitioning scheme than we have assumed so the total external fragmentation will be an overestimate.

(iv) Use of the lower bound calculation to estimate the number of blocks of each size allocated will lead to an overestimation since the additional external fragmentation is ignored. Reducing the number of blocks of each size will reduce the additional external fragmentation and again our calculation will lead to an overestimate.

The improved upper bounds to the average external fragmentation are compared in Table 1 with the results of the simulation studies. The improved upper bounds are found to correspond fairly well with the upper one standard deviation points found in the simulations, When taken together with the lower bound estimate they provide an estimate of the total memory fragmentation of considerable practical value.

Conclusion

The behaviour of a buddy system depends quite strongly on the storage partitioning scheme, the form of the request distribution, and the details of the fit of the end points and any peaks in the request distribution to the allocatable block sizes. Using the methods presented in this paper and a small calculator the memory fragmentation in any case of interest can be estimated in about 30 min and the estimates may be expected to be of considerable practical value. Certainly the evaluation of new buddy schemes should be greatly facilitated.

Any significant tightening of the estimates of external fragmentation will need a much better model than that used in the present paper since the differences between the present estimates and simulation studies is not much greater than the effects which simulation studies [6] show for minor variations in the

implementation such as whether free blocks are stacked or quened. The present model, since it neglects the dynamic nature of the allocation/deallocation process, makes no provision for considering such factors.

Acknowledgement. I wish to thank Rodney Parkin for his capable assistance in carrying out the simulation studies and the referees of previous papers in this area for their provocative criticisms.

References

1. Bromley AG (1976) An improved buddy method for dynamic storage allocation, Proc 7th Aust Comp Conf, 708-715
2. Hinds JA (1975) An algorithm for locating adjacent storage blocks in the buddy system, Comm ACM **18**: 221-222
3. Hirschberg DS (1973) A class of dynamic storage allocation algorithms. Comm ACM **16**: 615-618
4. Knowlton KC (1965) A fast storage allocator. Comm. ACM **8**:623-625
5. Knuth DE (1968) The art of computer programming, Vol. 1, Addison-Wesley, Reading, MA, p 435
6. Shen KK, Peterson JL (1974) A weighted buddy method for dynamic storage allocation, Comm ACM **17**: 558-562; and **18**: 202.

Received July 18, 1977; Revised March 21, 1978