Nordic Journal of Computing

APPROXIMATE AVERAGE STORAGE UTILIZATION OF BUCKET METHODS WITH ARBITRARY FANOUT

CHUAN-HENG ANG
Department of Information Systems and Computer Science
National University of Singapore

HANAN SAMET*
Computer Science Department and
Institute of Advanced Computer Studies and
Center for Automation Research
University of Maryland
College Park, MD 20742

Abstract. The approximate average storage utilization of bucket methods with fanout of n, assuming a uniform distribution of data, is shown to depend only on the fanout and not on any other factors such as the directory structure. It obeys the formula $(\ln n)/(n-1)$. The analysis makes use of a generalization of previously published methods for $n = 2$ and $n = 4$ and its predictions match these results. The formula is applicable to methods that are based on digital searching (e.g., tries) or balancing rather than comparison-based methods. The formula is also used to detect an erroneous statement about the average storage utilization of a ternary system in [12].

Key words: data structures, analysis of algorithms, bucket methods, average storage utilization, B-tree, Grid file, EXHASH, EXCELL, quadtries, ternary system.

CR Classification: D.4.3, E.1, E.2, F.2.2, H.2.2

1. Introduction

In many applications, the volume of data is so high that it is stored in secondary storage such as disks where it is organized using various data structures for efficient processing. Since the data (termed *records*) is usually retrieved in chunks such as pages or blocks rather than individually, it is natural to organize these blocks (usually termed *buckets*) so that similarly-valued records are stored together. Methods that make use of buckets in organizing the records are called *bucket methods*. The techniques that we discuss are applicable to records corresponding to points in one, two, or more dimensions (i.e., multiattribute data). In order to simplify the presentation, we usually refer to the data (i.e., the set of records) as *points* since this makes

* This work was supported in part by the National Science Foundation under Grant IRI–9216970 and the Department of Energy under Contract DEFG0295ER25237.

Received March 1996. Revised September 1996.

the discussion of the analysis more meaningful although it is independent of the dimensionality of the data.

Each record is stored in a certain order in a designated bucket with capacity m. When a bucket is full, it is split into several buckets so that the records in the full bucket together with the additional record that caused the bucket to split can be redistributed according to a given rule into new buckets. There are many bucket methods in use. They differ, in part, on the type of records for which they are designed, the nature of the redistribution strategy, and on the number of buckets n into which a full bucket is split, termed the *fanout*. Some may use a directory, or even several levels of directories, to speed up bucket access.

If we regard each bucket as a node in a tree, then n is just the out degree of the tree, and the corresponding bucket method is termed a *general bucket method* of fanout n. If we only use the fanouts of the trees to classify the bucket methods, then many of them will be found to belong to the same class. For example, the B-tree [3], the Grid File [12], EXHASH [4], and EXCELL [16] are considered to be instances of the general bucket method of fanout 2, with 0.69 average storage utilization, even though they are designed to deal with data of different dimensionality. On the other hand, the PR quadtree (or more correctly the quadtrie) [13, 15] is a general bucket method of fanout 4. with 0.46 average storage utilization.

In this paper we are interested in the average storage utilization of the various bucket methods. The storage utilization u of bucket method R that uses J buckets of capacity m to provide a total capacity of $a = Jm$ slots for storing v records has a value of v/a where u ranges between 0 and 1. The more full is each bucket (i.e. the closer to m is the number of records that each bucket contains), the smaller is the value of J, and hence the larger is the value of u. Thus the greater the value of u, the more efficient is R in terms of storage utilization. In particular, we show that U, the average of u, under appropriate data distribution models, only depends on the fanout value n (i.e., how many buckets are needed when a full bucket is split) thereby enabling us to ignore the implementation details of the different bucket methods. It should be clear that regardless of the bucket method that we use, it will often be the case that U oscillates as the number of points increases since it is impossible for all buckets to always be full.

The analysis of the average storage utilization has been carried out by a number of researchers for different bucket methods (e.g., EXHASH [4], B-trees [9, 17], EXCELL [16]). Most assume that the data is drawn from a uniform distribution. However, their analysis was for individual bucket methods rather than a general one. Our contribution is a unification of their results in a manner that is independent of the fanout and the directory structure. It is important to note that our analysis assumes that if a record is in a particular bucket, then it is equally likely to be in any of the children of the bucket in the case of a split. This assumption usually leads to a balanced structure and is thus useful for data structures based on digital methods (e.g., tries). or on the use of balancing (e.g., AVL trees [1], balanced B-trees [17], etc.). Such structures are the subject of our analysis. In contrast, in the case of comparison-based methods (e.g., binary search trees [8], point quadtrees [6], etc.), the order in which the data is encountered is important. Their analyses complement our results. In particular, they are based on the use of generating functions (e.g., [7]) and, of course, yield different results. We do not discuss them further here.

The rest of this paper is organized as follows. We first derive in Section 2 the average storage utilization for arbitrary n by generalizing the method of [4]. This method is quite complex but is done to show that we can only obtain an asymptotic result (i.e., as the number of records gets very large) which oscillates about its mean. Next, in Section 3, we obtain an approximate average storage utilization, which is the same as the mean derived in Section 2 but in a much simpler manner, by generalizing the method of [9]. Section 4 verifies the results obtained in Sections 2 and 3 and shows how a previously published result for a ternary system is discovered to be erroneous. Section 5 concludes our presentation by mentioning a simple application of the formula.

2. Derivation of Average Storage Utilization

In this section we perform a detailed mathematical analysis of the average storage utilization of a general bucket method of fanout n. The analysis is a generalization from $n = 2$ to arbitrary values of n (thereby being applicable to any other bucketing methods based on digital methods and balancing) of the analysis performed by Fagin, *et al.* [4] for EXHASH which is based on an analysis by Knuth [8].

Let T be a given tree structure of a general bucket method of fanout n with v points (i.e., records) with node (i.e., bucket) capacity m. Let I be the number of internal nodes, J the number of leaf nodes (i.e., buckets), and S be the total number of nodes of T. We have $S = I + J$. Since $I = (S-1)/n$, we have $I = S - J = nI + 1 - J$. So $J = (n-1)I + 1$. By taking the average over all T containing v random points, we have $E[J] = (n-1) \cdot E[I] + 1$.

As explained in [5, page 159], we can assume that the random points falling within a node at depth j follow a Poisson distribution such that the probability for a node to contain k points is

$$\frac{e^{-v_j} v_j^k}{k!} \quad \text{where} \quad v_j = \frac{v}{n^j}.$$

The probability for a node to be internal is

$$p_j = \sum_{k=m+1}^{\infty} \frac{e^{-v_j} v_j^k}{k!},$$

which is just the probability for the node to contain more than m points. Strictly speaking, the summation should begin with $k = m + 1$ and only go up to $k = v$ instead of $k = \infty$. In the following, in order to simplify our presentation, we always omit the range of an integral and the range of the index of a summation as long as the omission does not cause any confusion. Since there are n^j possible nodes at depth j, and letting p_j be the probability that each possible node is actually an internal node, the average number of internal nodes at depth j is $n^j p_j$. We find that

$$E[I] = \sum_{j=0}^{\infty} n^j p_j = \sum_j n^j \sum_k e^{-v_j} \frac{v_j^k}{k!} = \sum_j \sum_k n^j e^{-v_j} \frac{v_j^k}{k!}.$$

Since $v_j = v/n^j$, we can rewrite the term within the double summation as $a_k b_k$ where

$$a_k = e^{-v_j} \frac{v_j^{k-1}}{(k-1)!} \quad \text{and} \quad b_k = v/k.$$

Since all a_k's are positive, $\sum_{k=m+1}^{\infty} a_k$ converges to a value which is less than $\sum_{k=1}^{\infty} a_k = 1$. Since b_k is decreasing and $\lim_{k\to\infty} b_k = 0$, $\sum_k a_k b_k$ converges [14, theorem 3.42 page 70]. Moreover, it converges absolutely. The order of the summation can be interchanged [14, theorem 3.45 page 71]. Thus we have

$$E[I] = \sum_k \sum_j n^j e^{-v_j} \frac{v_j^k}{k!} = \sum_k \frac{v^k}{k!} \sum_j n^{j(1-k)} e^{\frac{-v}{n^j}}.$$

From [8, page 132] we know that

$$e^{-x} = \frac{1}{2\pi i} \int_{\frac{1}{2}-\infty i}^{\frac{1}{2}+\infty i} \frac{\Gamma(w)}{x^w} dw$$

where the integration is over the line $1/2 + ti$, and t is a real number. By changing the variable z to $w - 1$, we have

$$e^{-x} = \frac{1}{2\pi i} \int_{-\frac{1}{2}-\infty i}^{-\frac{1}{2}+\infty i} \frac{\Gamma(z+1)}{x^{z+1}} dz.$$

Using $z!$ for $\Gamma(z+1)$, we have,

$$\begin{aligned} E[I] &= \sum_k \frac{v^k}{k!} \sum_j n^{j(1-k)} \frac{1}{2\pi i} \int_{-\frac{1}{2}-\infty i}^{-\frac{1}{2}+\infty i} z!/(\frac{v}{n^j})^{z+1} dz \\ &= \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} \int_{-\frac{1}{2}-\infty i}^{-\frac{1}{2}+\infty i} \frac{z!}{v^{z+1}} \sum_j (n^{z+2-k})^j dz \end{aligned}$$

Since $k \geq m+1 \geq 2$ for non-leaf nodes and a point on the line of integration can be expressed as $z = -1/2 + ti$ where t is a real number, we have $\mathrm{Re}(z + 2 - k) = -1/2 + 2 - k < 0$ and hence $|n^{z+2-k}| = |n^{3/2-k}| \cdot |n^{ti}| = |n^{3/2-k}| < 1$.

Since $\sum_j |n^{z+2-k}|^j$ converges, $\sum_j (n^{z+2-k})^j$ also converges [14, theorem 3.45 page 71]. Therefore,

$$\begin{aligned} E[I] &= \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} \int_{-\frac{1}{2}-\infty i}^{-\frac{1}{2}+\infty i} \frac{z!}{v^{z+1}} \frac{1}{1-n^{z+2-k}} dz \\ &= \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} \int_{-\frac{1}{2}-\infty i}^{-\frac{1}{2}+\infty i} \phi(z) dz \quad \text{where} \quad \phi(z) = \frac{z!}{v^{z+1}} \frac{1}{1-n^{z+2-k}}. \end{aligned}$$

The integrals of complex variables can sometimes be solved by finding the poles of the integrand, superimposing a closed curve over the path of integration, and applying the Cauchy residue theorem [14]. The poles of a function $\phi(z)$ are defined to be those z which satisfy $\phi(z) = \infty$ or equivalently, $1/\phi(z) = 0$. A pole z is *simple* if it is a simple root of $1/\phi(z) = 0$. In the above integration, the poles of the integrand are those obtained from $z!$ which are $z = -1, -2, \cdots$ that will force $z!$ to become ∞ and those roots of $1 - n^{z+2-k} = 0$ which are $z = k - 2 + 2\pi i h \log_n e$, where h is an integer. All these poles are simple.

Letting L_1 be the line joining the points $-\frac{1}{2} - si$ to $-\frac{1}{2} + si$, we have

$$E[I] = \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} G \quad \text{where} \quad G = \int_{-\frac{1}{2}-\infty i}^{-\frac{1}{2}+\infty i} \phi(z) dz = \lim_{s\to\infty} \int_{L_1} \phi(z) dz$$

Let $L_2 = [k-3/2-\epsilon-si, k-3/2-\epsilon+si]$, $L_3 = [-1/2-si, k-3/2-\epsilon-si]$, $L_4 = [k-3/2-\epsilon+si, -1/2+si]$. Then $C = -L_1 \cup L_3 \cup L_2 \cup L_4$ is a closed curve. We assume this curve to be extended to infinity implicitly in the following derivation and drop $\lim_{s\to\infty}$ from now on. Now $\int_C = \int_{-L_1} + \int_{L_2} + \int_{L_3} + \int_{L_4}$. By the Schwarz reflection principle, the function value of $\phi(z)$ on L_3 is identical to that on L_4. Therefore, $\int_{L_3} = -\int_{L_4}$. Thus $G = \int_{L_1} = -\int_C + \int_{L_2}$, and we have

$$E[I] = \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} G = B + \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} \int_{L_2} \phi(z) dz \quad \text{where}$$

$$B = \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} \cdot \left(- \int_C \phi(z) dz \right).$$

Notice that all the poles $\{-1, -2, \cdots\}$ are outside the region bounded by C whereas the poles $k - 2 + 2\pi i h \log_n e$ are within the region when C is sufficiently large.. According to the Cauchy residue theorem, the integral of a complex function $\phi(z)$ over a closed curve is $2\pi i$ times the sum of the residues of the function at the poles. The residue of $\phi(z)$ at a pole which is outside the closed curve is zero. If $\phi(z)$ is written as $g(z)/f(z)$, then the residue of $\phi(z)$ at a simple pole w within a closed curve is $g(w)/f'(w)$. Since the poles $\{-1, -2, \cdots\}$ lie outside C, the residues at these points are zeroes. If we rewrite the integrand $\phi(z)$ as $g(z)/f(z)$ where $g(z) = z!/v^{z+1}$

and $f(z) = 1 - n^{z+2-k}$, then we have $f'(z) = -n^{z+2-k}\log_e n$. For each pole $w = k - 2 + 2\pi i h \log_n e$, and therefore

$$f'(w) = -(\log_e n)n^{2\pi i h \log_n e} = -\log_e n (n^{\log_n e})^{h2\pi i} = -\log_e n.$$

We have

$$\begin{aligned} -\int_C \phi(z)dz &= -2\pi i \sum_h \frac{1}{-\log_e n} \frac{(k-2+2\pi i h \log_n e)!}{v^{k-1+2\pi i h \log_n e}} \\ &= 2\pi i \sum_h \log_n e (k-2+2\pi i h \log_n e)! v^{-k+1-2\pi i h \log_n e}. \end{aligned}$$

We can now solve for B.

$$\begin{aligned} B &= \sum_k \frac{v^k}{k!} \frac{1}{2\pi i} (-\int_C \phi(z)dz) \\ &= \sum_k \frac{v^k}{k!} \sum_h \log_n e (k-2+2\pi i h \log_n e)! v^{-k+1-2\pi i h \log_n e} \\ &= \sum_k \sum_h \log_n e (k-2+2\pi i h \log_n e)! \frac{v}{k!} e^{-2\pi i h \log_n v} \\ &= v \sum_h e^{-2\pi i h \log_n v} \log_n e \sum_k \frac{(k-2+2\pi i h \log_n e)!}{k!} \\ &= v \sum_h e^{-2\pi i h \log_n v} (\log_n e) A \quad \text{where} \quad A = \sum_{k=m+1} \frac{(k-2+2\pi i h \log_n e)!}{k!}. \end{aligned}$$

As noted before, the summation index k starts from $m+1$ for an internal node. Now

$$\begin{aligned} A &= \sum_{k=0} \frac{(k+m-1+2\pi i h \log_n e)!}{(k+m+1)!} \\ &= \frac{(m-1+2\pi i h \log_n e)!}{(m+1)!} F(m+2\pi i h \log_n e, 1; m+2; 1) \end{aligned}$$

where $F(a,b;c;x) = 1 + \frac{a \cdot b}{1 \cdot c}x + \frac{a \cdot (a+1) \cdot b \cdot (b+1)}{1 \cdot 2 \cdot c \cdot (c+1)}x^2 + \cdots$ is a hypergeometric series. When $x = 1$, $F(a,b;c;1) = \frac{\Gamma(c)\Gamma(c-a-b)}{\Gamma(c-a)\Gamma(c-b)}$.

In addition, when $b = 1$, $F(a,1;c;1) = \frac{\Gamma(c)\Gamma(c-a-1)}{\Gamma(c-a)\Gamma(c-1)} = \frac{c-1}{c-a-1}$.

Therefore, we have

$$\begin{aligned} A &= \frac{(m-1+2\pi i h \log_n e)!}{(1-2\pi i h \log_n e) \cdot m!} \quad \text{and} \\ B &= v \sum_h e^{-2\pi i h \log_n v} \log_n e \frac{(m-1+2\pi i h \log_n e)!}{(1-2\pi i h \log_n e) \cdot m!} \\ &= \frac{v}{m} \sum_h c_{m,h} e^{-2\pi i h \log_n v} = \frac{v}{m} \Phi_m(\log_n v) \end{aligned}$$

where

$$c_{m,h} = \frac{\log_n e}{(1 - 2\pi i h \log_n e)} \frac{(m-1+2\pi i h \log_n e)!}{(m-1)!} \quad \text{and}$$

$$\Phi_m(x) = \sum_h c_{m,h} e^{-2\pi i h x}.$$

We still need to solve $\int_{L_2} \phi(z)dz$ in order to evaluate $E[I]$. From [4], we have

$$\frac{1}{2\pi i}\int_{L_2} \phi(z)dz = O\left(\frac{(k-1-\epsilon)!}{(v^{k-\frac{1}{2}-\epsilon}(\frac{1}{2}-\epsilon))}\right) \quad \text{and}$$

$$\sum_k \frac{v^k}{k!} O\left(\frac{(k-1-\epsilon)!}{v^{k-\frac{1}{2}-\epsilon}\cdot(\frac{1}{2}-\epsilon)}\right) = O\left(\frac{v^{\frac{1}{2}+\epsilon}}{\epsilon\cdot(\frac{1}{2}-\epsilon)}\right) = O(v^{\frac{1}{2}}\log v)$$

when $\epsilon = 1/\log v$. Therefore, we now have

$$E[I] = B + O(v^{\frac{1}{2}}\log v) = \frac{v}{m}\Phi_m(\log_n v) + O(v^{\frac{1}{2}}\log v) \quad \text{and}$$

$$E[J] = 1 + (n-1)E[I] = 1 + (n-1)(\frac{v}{m}\Phi_m(\log_n v) + O(v^{\frac{1}{2}}\log v))$$

When m is fixed, recall from Section 1 that the storage utilization u is v/Jm. Thus, the average storage utilization U is

$$U = \frac{v}{mE[J]} = \frac{1}{\frac{m}{v} + (n-1)\Phi_m(\log_n v) + O(v^{-\frac{1}{2}}\log v)}$$

When $v \to \infty$, we have $\frac{m}{v} \to 0$ and $v^{-\frac{1}{2}}\log v \to 0$. In this case, $\lim_{v\to\infty} U$ does not exist since Φ_m is oscillating. The oscillation is caused by the fact that as the number of points v increases, it is impossible for all the buckets to always be full thereby causing a variation in the storage utilization u and the average storage utilization U as they are computed in terms of the number of avaliable slots for storing points. The average of $\Phi_m(x)$ is

$$c_{m,0} = \frac{\log_n e(m-1)!}{1\cdot(m-1)!} = \log_n e.$$

Therefore, the mean value of U is

$$\frac{1}{(n-1)\log_n e} = \frac{\log_e n}{n-1}.$$

3. A Simpler Derivation

The rationale behind the long, and somewhat tedious, process of deriving the average storage utilization U of a general bucket method is to enable us to realize that U can only be represented by $\Phi_m(\log_n v)$ asymptotically. There are other terms of smaller order which contribute to U and diminish as $v \to \infty$. Most importantly, the average storage utilization does not have a limit as $v \to \infty$. Instead, it always oscillates around the mean $c_{m,0}$.

If we are only interested in finding the mean value of U, the average storage utilization, then we can adapt the result described in [9] to the case of a general bucket method. Assume that in a tree T of a general bucket method, each full node with m points is split into n nodes each containing m/n points. Strictly speaking, there should be $m+1$ points in this case, but in order to simplify matters we use m in our derivation. The storage utilization u of T with v points and k leaf nodes is $u = v/(km)$. By taking the average, we have

$$U = E[u] = \frac{v}{m}E[\frac{1}{k}].$$

The minimum value of k is v/m, and arises when all the nodes are full. The maximum value of k is nv/m and arises just after all the nodes have been produced by splitting their parent nodes. Assume that the number of nodes follows a uniform distribution in the interval $[v/m, nv/m]$. The corresponding density is thus

$$\frac{1}{\frac{nv}{m} - \frac{v}{m}}dk = \frac{m}{(n-1)v}dk,$$

and we have

$$\begin{aligned} E[\frac{1}{k}] &= \int_{v/m}^{nv/m} \frac{1}{k}\frac{m}{(n-1)v}dk \\ &= \frac{m}{(n-1)v}\left[\ln k\right]_{v/m}^{nv/m} \\ &= \frac{m}{(n-1)v}(\ln\frac{nv}{m} - \ln\frac{v}{m}) \\ &= \frac{m}{(n-1)v}\ln n. \end{aligned}$$

Substituting into U yields the same result as we obtained in Section 2 for the mean of U — that is

$$U = \frac{v}{m}\cdot\frac{m}{(n-1)v}\ln n = \frac{\ln n}{n-1}.$$

The result obtained in this section is an approximation as we have assumed that the distribution of the leaf nodes is uniform in its range of values rather than the distribution of the points. Also, we have assumed that the distribution of the leaf nodes is continuous where, in fact, the number of leaf nodes can only take on integer values. Thus we have shown that the approximate value of U derived in this section is the same as the mean of U derived in Section 2. In the rest of this paper we refer to both of these values as the *approximate average storage utilization.*

4. Empirical Results

In order to evaluate the accuracy of our formula for the approximate average storage utilization, we compare the values it yields with known values of some widely used bucket methods. When $n = 2$, $U = 0.693$. This is close to 0.67 which is reported as the average storage utilization of the B-tree [8], the Grid file [12], and EXCELL [16]. When $n = 4$, $U = 0.462$. This is close to 0.47 which is reported as the average utilization of a PR quadtree [16].

When $n = 3$, the general bucket method is termed a *ternary system*. Its average storage utilization has been observed to be 0.39 [12] which is quite different from our predicted analytic value of $(\ln 3)/2 = .549$. No formal derivation for the 0.39 value was given in [12]. As we show below, we believe that our predicted value of .549 is more plausible and that the value of 0.39 is far too low for a ternary system.

Consider a general bucket method of fanout 2. Although its minimum storage utilization is 0.5, when a binary tree which consists of only one full bucket is split, the approximate average storage utilization is 0.693 – that is, an increase of 0.193 over the minimum value. Similarly, when the fanout of the general bucket method is 4, its minimum storage utilization is 0.25, while its approximate average storage utilization is 0.462 – that is, an increase of 0.212 over the minimum value. Therefore, we would expect that the approximate average storage utilization of a general bucket method of fanout 3 to have an increment somewhere around $(0.193 + 0.212)/2 = 0.202$ in excess of its minimum storage utilization which is 0.333. In other words, the average storage utilization of a ternary system is expected to be about 0.535, which is an approximation of 0.549, our predicted analytic result.

As further evidence for our doubts about the correctness of the result reported previously [12] for the ternary system, we performed some empirical tests. In particular, we wrote a program to create a specified number of trees with a given fanout, each of which contains a designated number of random points, and calculated the average storage utilization of these trees. When the average storage utilizations are plotted against the number of points stored in the trees, the curves were found to oscillate for all the general bucket methods regardless of the fanout values. Figure 4.1 shows the results for a ternary system which confirm our doubts about the value reported in [12].

From the analysis in Section 2, we know that the average storage utilization of a general bucket method is basically a periodic function in $\log_n v$. Thus, taking the average of the maximum and the minimum of the values from Figure 4.1 produces a more satisfactory result in estimating the average storage utilization than that obtained by simply taking the average of all the values in Figure 4.1. Moreover, to minimize the contribution of the term $O(v^{-1/2} \log v)$ in the formula for the average storage utilization, larger values of v should be used whenever possible. Referring to Figure 4.1, we see that for the ternary system, the maximum is around 0.576, the minimum is 0.525, and the average is 0.551, which is close to the predicted analytic

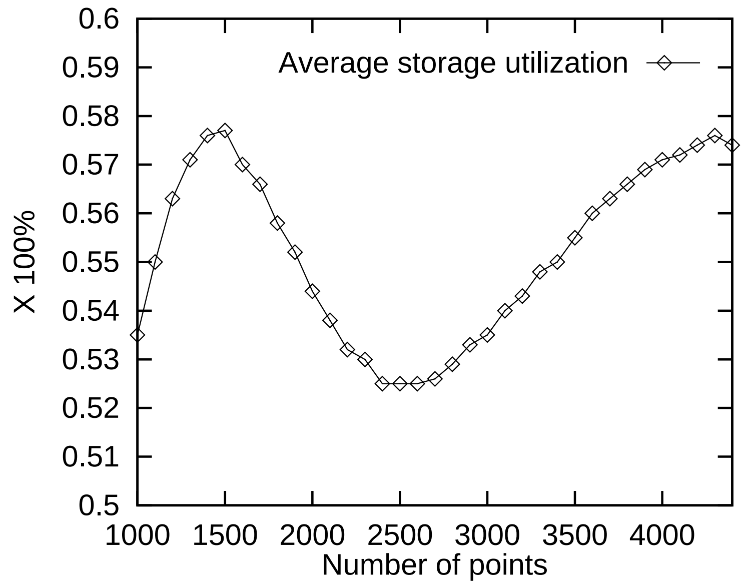

Fig. 4.1: Average storage utilization of a ternary system.

Fanout	Empirical value	Predicted value
2	0.695	0.693
3	0.551	0.549
4	0.468	0.462
5	0.415	0.402
6	0.376	0.358

Fig. 4.2: Average storage utilizations of general bucket methods.

value of 0.549. Using such an estimating technique, Figure 4.2 tabulates the empirical values of the average storage utilizations for the general bucket methods of fanout 2 to 6. The figure also tabulates the predicted analytic values for comparison. From the figure, we see that the empirical and predicted analytic values are in close agreement.

5. Conclusion

We have shown two ways to derive a general formula used to calculate the approximate average storage utilization of a general bucket method. As a result, there is no need to perform the tedious derivation of the average storage utilization of a specific bucket method in the future. The formula is simple and easy to use.

An immediate application of our result is to estimate of the storage requirements of a bucket method. In particular, If a bucket method with node capacity m is used to store v points, then the expected number of leaf nodes (or buckets) $E[J]$ that is required can be calculated by using the formula

$$E[J] = \frac{v}{mU} \approx \frac{(n-1)v}{(\log_n e)m} = (n-1)\log_e n \frac{v}{m}.$$

For verification purposes, it has been reported in [2] that for a PR quadtree with $m = 1$ and $v = 1000$, the average number of leaf nodes created is 2153.4 which is very close to the estimated value $E[J]$ of 2164.

Another consequence of our analysis is that we are now able to explain the phasing phenomenon [10, 11] (i.e., the oscillation of the plot of the average storage utilization of a bucket method). Since $U = 1/\Phi_m(\log_n v)$ asymptotically and Φ_m is periodic in $\log_n v$, we have that U is also periodic in $\log_n v$ and its curve will fluctuate like a sine function around the average $c_{m,0}$.

6. Acknowledgements

We have benefitted greatly from discussions with Michael B. Dillencourt.

References

[1] Adel'son-Vel'skiĭ, G. M. and Landis, E. M. 1962. An algorithm for the organization of information. *Doklady Akademii Nauk SSSR 146*, 263–266.

[2] Ang, C. 1989. *Applications and analysis of hierarchical data structures*. PhD thesis, University of Maryland, College Park, MD.

[3] Bayer, R. and McCreight, E. M. 1972. Organization and maintenance of large ordered indices. *Acta Informatica 1*, 3, 173–189.

[4] Fagin, R., Nievergelt, J., Pippenger, N., and Strong, H. R. 1979. Extendible hashing - a fast access method for dynamic files. *ACM Transactions on Database Systems 4*, 3 (September), 315–344.

[5] Feller, W. 1970. *An Introduction to Probability Theory and its Applications*, third edition. Volume 1. John Wiley and Sons, New York.

[6] Finkel, R.A. and Bentley, J.L. 1974. Quad trees: a data structure for retrieval on composite keys. *Acta Informatica 4*, 1, 1–9.

[7] Hoshi, M. and Flajolet, P. 1992. Page usage in a quadtree index. *BIT 32*, 3, 384–402.

[8] Knuth, D. E. 1973. *The Art of Computer Programming: Sorting and Searching*. Volume 3. Addison-Wesley, Reading, MA.

[9] Leung, C. H. C. 1984. Approximate storage utilization of B-trees: a simple derivation and generalizations. *Information Processing Letters 19*, 12 (November), 199–201.

[10] Nelson, R. C. and Samet, H. 1986. A population analysis of quadtrees with variable node size. Tech. report, Computer Science TR 1740, College Park, MD.

[11] Nelson, R. C. and Samet, H. 1987. A population analysis for hierarchical data structures. In *Proceedings of the SIGMOD Conference*, 270–277.

[12] Nievergelt, J., Hinterberger, H., and Sevcik, K. C. 1984. The grid file: an adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems 9*, 1 (March), 38–71.

[13] Orenstein, J. A. 1982. Multidimensional tries used for associative searching. *Information Processing Letters 14*, 4 (June), 150–157.

[14] Rudin, W. 1966. *Real and Complex Analysis*. McGraw-Hill, New York.
[15] Samet, H. 1990. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA.
[16] Tamminen, M. 1982. Performance analysis of cell based geometric file organizations. *Computer Graphics, Vision, and Image Processing 24*, 2 (November), 168–181.
[17] Yao, A. C. 1978. On random 2-3 trees. *Acta Informatica 9*, 2, 159–168.