Buffer Pool Data Structure

The traditional model, as mentioned in the book *The Design of the UNIX Operating System* by *Maurice J. Bach*, describes the Buffer Pool as a Hashed Buffer Array which links to Buffer Queues. The concept of this Hashed Buffer Array came into picture to uniformly distribute buffers into the Buffer Pool so as to reduce the size of each buffer queue to optimize the searching process.

Let's say we distribute (uniformly) the Buffer Pool into k buffers, and we have a total of n buffers. Now, for each operation on the pool, we have to search for a specific buffer, which will take $O\left(\lceil \frac{n}{k} \rceil\right)$ time for each iteration as only linear search is supported by these Queues.

Though all other operations work in constant time, that is, $O(1)$, but still the most costly operation is the search operation, and it is done at each iteration of the *getblk* algorithm. So the complexity of each iteration in the *getblk* algorithm, with the above stated Buffer Pool data structure, is $O\left(\lceil \frac{n}{k} \rceil\right)$.

Now, let's say that it takes m *getblk* iterations for returning the appropriate buffer. We'll find the lower and upper bound on m.

The lower bound is pretty straight forward, it can be easily seen that in the best case, $m = 1$.

But if we try to find the upper bound on m, we get to know that in the worst case, it can take up to n iterations (consider the case when all buffers in the free list are marked as *delayed_write*).

$$\therefore 1 \leq m \leq n$$

Total time taken by the *getblk* algorithm is $O\left(m\lceil \frac{n}{k} \rceil\right)$.

Considering that all cases occur uniformly at random, then the expected value of m will be the mean of all possible values. This can be expressed mathematically as:

$$E[m] \;=\; \frac{1}{n}\sum_{i=1}^{n} i \;=\; \frac{1}{n} \times \frac{n(n+1)}{2} \;=\; \frac{n+1}{2}$$

$\therefore$ On an average, we can say that $m = \frac{n+1}{2}, \quad \implies \quad m = O(n)$.

Therefore, total time of the *getblk* algorithm can be bounded by the set of polynomials $O\left(\frac{n^2}{k}\right)$. As k is a constant, we can say that average time is bounded by $O(n^2)$.

What we did to do it better?

We have eliminated the concept of the Hashed Buffer Array linked with Linear Queues and placed a Balanced Binary Search Tree (BST) as the Buffer Pool itself to avoid linear searches, which apparently was the costliest operation of the previously used Data Structure. We have used `std::set` from `C++ Standard Template Library (STL)` with an appropriate `Custom Comparator` for our buffer objects. It is a Red Black Tree (Balanced BST) which keeps unique elements and keeps them sorted at all times. All operations work in $O(\log_2 n)$ time, where n is the size of the container (Buffer Pool).

The pace at which modern computers are developing, memory is becoming cheaper, and we have an ease to use enormously more memory than what we used to 20 years back. Considering this fact, the number of buffers on a system is growing at a very rapid pace. The previously discussed $O(n^2)$ approach will not suffice for a large value of n.

Now, let us compare our model's performance with the previously discussed one. The previous model took $O(n^2)$ time for a single call on average, considering uniformity in randomness of the buffers demanded. We will provide the exact same conditions to the same algorithm but with our modified Data Structure.

For a single *getblk* call, it now takes $O(m \log_2 n)$ time in total.

For $m = O(n)$, as described in the previous section, our modified *getblk* algorithm will work in $O(n \log_2 n)$ time for a single call.

We'll compare the performance of both the Algorithms.

n	$O(n^2)$	$O(n \log_2 n)$
10^4	0.104 s	0.001 s
10^5	7.318 s	0.001 s
10^6	125.622 s	0.001 s

Moreover, we've also studied from `Linux` source code that it also maintains it's buffer cache in the form of a non - linear data structure (Radix Tree). So, in our opinion (and according to the mathematical analysis), this is a more practical model, than the one given in the before mentioned book.