# Template

Mali Akmanalp, Sophie Hilgard, Andrew Ross

February 26, 2017

## 1 Base Case

Assume $n$ items in total DB
$NC$ items that fit in cache
$NM$ items that fit in memtable
$R$ ratio between layers of LSM tree such that
$L1 = R * NM$
$L2 = R^2 * NM \ldots$

We can solve for $j$ the total number of layers required to store all the data:

$$NM * \frac{1 - R^j}{1 - R} = n$$

$$j = \frac{log(1 - n * \frac{1-R}{NM})}{logR}$$

The average cost of a write remains the same as for the basic LSM tree case:

$$\log_R \frac{n}{NM}$$

The average cost of a read, we consider probabilistically over all possible locations of the read item, assuming a random distribution of reads:
Probability that read is in memtable = $\frac{NM}{n} = p(mt)$
Probability that read is in cache = $\frac{NC}{n} = p(cache)$
Probability that read is in L1 but not in cache = $\frac{NM*R - \frac{NM*R}{NM*\frac{1-(R^j-1)}{1-R}} * NC}{n} = p(L1)$
Where the numerator is the number of items that are in the first layer

$$NM * R$$

minus the proportion of items from that layer that are probabilistically in the cache already

$$\frac{NM * R}{NM * \frac{1-(R^j-1)}{1-R}} * NC$$

Where here the $R^j - 1$ comes from the fact that items already in memtable (L0) are not allowed to occupy the cache.

Expected cost of read = $p(mt) * 0 + p(cache) + 0 + \sum_{i=1}^{j} p(Li) * i$

## 2 Skewed Reads

Now consider the case for skewed reads, where we say $d_{hf}$ ($d_{lf}$) percent of the data receives $r_{hf}$ ($r_{lf}$) percent of the reads (where $d_{hf} + d_{lf} = 1$ and $r_{hf} + r_{lf} = 1$). On average, we can assume that the cache contains $r_{hf} * NC$ items from $d_{hf} * n$ and $r_{lf} * NC$ items from $d_{lf} * n$. Then the expected cost of a read is dependent on whether the data item being read is in $d_{hf} * n$ or $d_{lf} * n$ as the probability of a cache hit varies.
For data in $d_{hf} * n$,

Probability that read is in memtable = $\frac{NM * d_{hf}}{d_{hf} * n} = p(mt)$
Probability that read is in cache = $\frac{r_{hf} * NC}{d_{hf} * n} = p(cache_{hf})$
Probability that read is in L1 but not in cache = $\frac{NM * R * d_{hf} - \frac{NM * R}{NM * \frac{1-(R^j-1)}{1-R}} * r_{hf} * NC}{d_{hf} * n} = p(L1_{hf})$
Expected cost of read on item in $d_{hf}$: $E[C_{hf}] = p(mt) * 0 + p(cache_{hf}) + 0 + \sum_{i=1}^{j} p(Li_{hf}) * i$

Concretely, consider where we have 3 levels and 800 total items with a cache of size 10 and a ratio of 2 (for L0=100, L1 = 200, L2 = 400 items), with $d_{hf} = .2$ and $d_{lf} = .8$ and $r_{hf} = .8$ and $r_{lf} = .2$. Then the cache on average contains 8 items from $d_{hf} * n$ and 2 items from $d_{lf} * n$. If we execute a read on one of the 200 items in $d_{hf}$, then, there is a $\frac{8}{200}$ chance that that item is in the cache. If we execute a read on one of the $200 * \frac{1}{4} = 50$ items of $d_{hf} * n$ in L1, we expect that $\frac{2}{6} * 8$ of those items would have actually been found already in cache, as this level contains $\frac{2}{6}$ of all of the items not in the memtable. Then the probability that a read is found in L1 is the proportion of the $d_{hf} * n = 160$ items that will reside in L1 but not in the cache, which is $\frac{40 - \frac{2}{6} * 8}{160}$.

The expected cost of a read on an item in $d_{lf}$ can be enumerated analogously, and we combine the expectation of reads in $d_{hf}$ and $d_{lf}$ as:
Expected cost of read = $r_{hf} * E[C_{hf}] + r_{lf} * E[C_{lf}]$

We can also define $r_{lf}$ in terms of $r_{hf}$ as $r_{hf} - 1$ and $d_{lf}$ in terms of $d_{hf}$ as $d_{hf} - 1$ . (Doing this will make the effect that moving these parameters in one direction or the other has more obvious in the total overall formula.)

## 3 Bloom Filters

For a Bloom filter of $k$ bits with $h$ independent hash functions $h_1, h_2, ...h_h$, the probability that a given bit is still set to 0 after inserting $n$ keys is

$$(1 - \frac{1}{k})^{n*h}$$

Then the probability of a false positive is

$$(1 - (1 - \frac{1}{k})^{n*h})^h \approx (1 - e^{-hn/k})^h$$

We can minimize this over $h$ to find the optimal number of hash functions, which is $h = \ln(2) * \frac{k}{n}$. Assuming that this is the number of hash functions $h$ we will use, the probability of a false positive

as a function of the number of bits is then

$$(1 - e^{-\ln(2)*k/n*n/k})^{\ln(2)*\frac{k}{n}} = (\frac{1}{2})^{\ln(2)*\frac{k}{n}} \approx (.6185)^{\frac{k}{n}}$$

For an item in any any level $L_i$ of the LSM tree with $i \geq 2$ we can reduce the expected cost of accessing that item from $i$ by the number of Bloom filter negatives at any level $j < i$.
Then the expected cost of accessing an item at $L_i = \sum_{j=1}^{i} p(fp_j) * 1 + 1$
Where $p(fp_j)$ is the probability of a false positive for that key at level $j$ and 1 is the cost of actually accessing the item at level $i$ assuming fence pointers that lead us to the correct page.

# 4 Variable Cache Size