Cache Oblivious Median Finding.

1. Conceptually partition the array into $N/5$ 5-tuples.

2. Compute the median of each 5-tuple by two parallel scans. Takes $\Theta(N/B + 1)$ memory transfers, assuming that $M \geq 2B$.

3. Recursively compute the median $m$ of these medians (i.e. a recursive call on a problem of size $N/5$).

4. Partition the array into the elements $\leq m$ and the elements $> m$ by doing three parallel scans, one reading the array, and two others writing the partitioned arrays. This takes $\Theta(N/B + 1)$ memory transfers assuming that $M \geq 3B$.

5. Count the lengths of these two subarrays and recurse into the appropriate half.

Recurrence for running time (see e.g. CLRS):

$$T(N) = T(1/5N) + T(7/10N) + O(N)$$

Recurrence for number of memory transfers:

$$T(N) = T(1/5N) + T(7/10N) + O(N/B + 1)$$

What’s the base case? First try: $T(O(1)) = O(1)$. Then there are $N^c$ leaves in the recursion tree, where $c \approx 0.8397803$ and each leaf incurs a constant number of memory transfers. So $T(N)$ is at least $\Omega(N^c)$, which is larger than $O(N/B + 1)$ when $N$ is larger than $B$ but smaller than $BN^c$.

Second try: $T(O(B)) = O(1)$, because once the problem fits into $O(1)$ blocks, all five steps incur only a constant number of memory transfers. Then there are only $(N/B)^c$ leaves in the recursion tree, which cost only $O((N/B)^c) = o(N/B)$ memory transfers. Thus the cost per level decreases geometrically from the root, so the total cost is the cost of the root: $O(N/B + 1)$.

---

$^1$c is the solution to $(1/5)^c + (7/10)^c = 1$ which arises from plugging $L(N) = N^c$ into the recurrence for the number $L(N)$ of leaves: $L(N) = L(N/5) + L(7N/10), L(1) = 1$. 

1