1 Detailed Theoretical Analysis

Lemma 1.1 (Expected DAC for All-at-Once Fetching)**.** *If the predicted position lies in a page with a uniformly distributed offset, the expected number of I/Os with all-at-once strategy is*

$$\mathbb{E}[DAC] = 1 + \frac{2\varepsilon}{C_{\text{ipp}}}, \tag{1}$$

where C_{ipp} *represents the number of items per page.*

Proof. Denote the offset within the page of the predicted position by $s \sim U(0, C_{ipp} - 1)$. The I/O needs to fetch the page that contains the predicted position and additional pages required to cover the left and right of the $\pm\varepsilon$ window:

$$\mathbb{E}[\text{DAC}] = \frac{1}{C_{\text{ipp}}} \sum_{s=0}^{C_{\text{ipp}}-1} \left(1 + \left\lceil \tfrac{\varepsilon - s}{C_{\text{ipp}}} \right\rceil + \left\lceil \tfrac{\varepsilon - (C_{\text{ipp}} - 1 - s)}{C_{\text{ipp}}} \right\rceil \right). \tag{2}$$

Rewrite $\varepsilon = \lambda \cdot C_{\text{ipp}} + r$ for some $\lambda \in \mathbb{N}$ and $0 \leq r < C_{\text{ipp}}$. Then we have

$$\begin{aligned} \mathbb{E}[\text{DAC}] &= \sum_{s=0}^{C_{\text{ipp}}-1} \frac{1}{C_{ipp}} \Big(1 + (\lambda + \mathbb{1}_{\{s<r\}}) + (\lambda + \mathbb{1}_{\{s>C_{ipp}-1-r\}})\Big) \\ &= 1 + 2\lambda + \frac{2r}{C_{\text{ipp}}} = 1 + \frac{2\varepsilon}{C_{\text{ipp}}}. \end{aligned} \tag{3}$$

□

Lemma 1.2. *(Expected Cost For One-By-One Fetching). If the predicted position lies in a page with a uniformly distributed offset, the expected number of I/Os under the one-by-one strategy is*

$$\mathbb{E}[DAC] = 1 + \frac{\varepsilon}{C_{ipp}} \tag{4}$$

Proof. Suppose the predicted position is at $\hat{y}$ and the true position y is uniformly random over the entire search window $[\hat{y} - \varepsilon, \hat{y} + \varepsilon]$. Let X denote the distance between y and the lower bound $\hat{y} - \varepsilon$, i.e., $X \sim U(0, 2\varepsilon)$.

Let k denote a uniformly distributed offset of the lower bound within a page. The expected I/O cost becomes:

$$\mathbb{E}(\text{DAC}) = 1 + \frac{1}{2\varepsilon + 1} \sum_{x=0}^{2\varepsilon} \Big(\frac{1}{C_{\text{ipp}}} \sum_{k=0}^{C_{\text{ipp}}-1} \Big\lfloor \frac{k + x}{C_{\text{ipp}}} \Big\rfloor \Big). \tag{5}$$

Let $x = q \cdot C_{\text{ipp}} + r_1, 0 \leq r_1 < C_{\text{ipp}}$,

$$\frac{1}{C_{\text{ipp}}} \sum_{k=0}^{C_{\text{ipp}}-1} \Big\lfloor \frac{k + x}{C_{\text{ipp}}} \Big\rfloor = q + \frac{r_1}{C_{\text{ipp}}}. \tag{6}$$

Let $2\varepsilon = M \cdot C_{\text{ipp}} + r_2, 0 \leq r_2 < C_{\text{ipp}}$,

$$\begin{aligned} \sum_{x=0}^{2\varepsilon} q &= \sum_{m=0}^{M-1} m \cdot C_{\text{ipp}} \; + \; M \cdot (r_2 + 1) \\ &= \frac{C_{\text{ipp}} \cdot M \cdot (M-1)}{2} + M \cdot (r_2 + 1), \end{aligned} \tag{7}$$

$$\sum_{x=0}^{2\varepsilon} r_1 = M \cdot \frac{C_{\text{ipp}} \cdot (C_{\text{ipp}} - 1)}{2} + \frac{r_2 \cdot (r_2 + 1)}{2}. \tag{8}$$

Combining Equations (5) to (8), we obtain:

$$\begin{aligned} \mathbb{E}[\text{DAC}] &= 1 + \frac{1}{M \cdot C_{\text{ipp}} + r_2 + 1} \Big(\frac{(M \cdot C_{\text{ipp}} + r_2)(M \cdot C_{\text{ipp}} + r_2 + 1)}{2 \cdot C_{\text{ipp}}} \Big) \\ &= 1 + \frac{M \cdot C_{\text{ipp}} + r_2}{2 \cdot C_{\text{ipp}}} = 1 + \frac{\varepsilon}{C_{\text{ipp}}} \end{aligned} \tag{9}$$

□

Theorem 1.3 (Buffer Hit Rate for Sorted Queries)**.** *Let $\mathcal{K} = (k_1, \dots, k_{|\mathcal{K}|})$ be an array of keys. A query sequence $\mathcal{Q}$ is **sorted** w.r.t. $\mathcal{K}$ if it requests k_i before k_j for all $i < j$. For any such query sequence $\mathcal{Q}$, suppose the buffer capacity C satisfies*

$$C \geq 1 + \lceil 2\varepsilon / C_{\text{ipp}} \rceil,$$

where C_{ipp} is the number of items per page. Then the cache hit rate equals $h = \frac{R-N}{R}$.

Proof. Let $(p_1, \dots, p_R)$ be the page reference sequence generated by processing the m queries, and partition it by queries:

$$(p_1, \dots, p_R) = \pi_1 \| \pi_2 \| \cdots \| \pi_m,$$

where π_t is the subsequence of page IDs referenced while processing query t. For a learned-index-based engine, query t touches exactly the pages in its last-mile window $W_t = [L_t, H_t]$, with $|W_t| \leq 1 + \lceil \frac{2\varepsilon}{C_{\text{ipp}}} \rceil$. Let $\mathcal{P}$ be the set of distinct pages appearing in $(p_1, \dots, p_R)$, and $|\mathcal{P}| = N$. Since queries are sorted, the windows move monotonically, i.e., $L_{t+1} \geq L_t$. Hence, between two consecutive queries, only pages newly entering the window can miss: pages in $W_{t+1} \cap W_t$ are still resident and therefore hit, while pages in $W_{t+1} \setminus W_t$ may incur misses. Because $C \geq C_\delta \geq |W_t|$ for all t, the entire window W_t fits in cache during the processing of query t, so no page in W_t can be evicted before π_t finishes. By monotonicity of L_t, once a page is loaded it is either reused by subsequent overlapping windows (and thus hits) or never referenced again. Therefore, each distinct page in $\mathcal{P}$ incurs exactly one compulsory miss—on its first reference—and all later references are hits. The total number of misses is N, so the hit rate is

$$h = \frac{R - N}{R},$$

as claimed. □

Corollary 1.4 (Sorted Order Maximizes Hit Rate)**.** *Given a multiset of queries $\mathcal{Q}$, the cache hit rate is maximized when $\mathcal{Q}$ is executed in sorted order.*

Proof. For an arbitrary ordering σ, let $\mathcal{P}$ be the set of distinct pages referenced by the resulting trace, and $|P| = N$. Let $b_\sigma(p)$ denote the number of cache misses (i.e., disk I/Os) incurred by page $p \in \mathcal{P}$. Since each distinct page must be brought into cache at least once, we have $b_\sigma(p) \geq 1$ for all $p \in \mathcal{P}$. Therefore,

$$\text{misses}(\sigma) = \sum_{p \in \mathcal{P}} p_\sigma(q) \;\geq\; \sum_{p \in \mathcal{P}} 1 \;=\; N \;=\; \text{misses}(\text{sorted}). \tag{10}$$

Under sorted order, we have $b_{\text{sorted}}(q) = 1$ for all q and the total number of misses attains this lower bound. It follows that

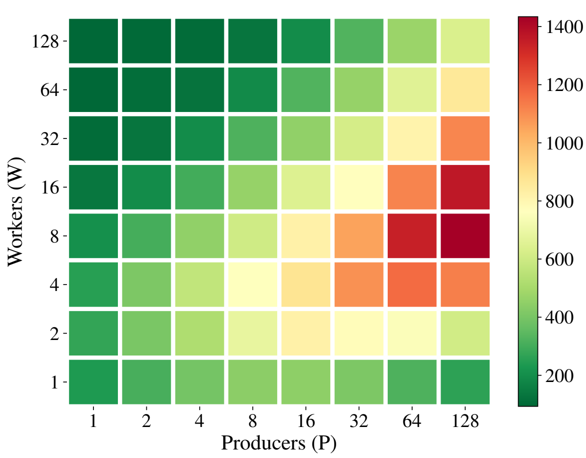
$$\text{hits}(\sigma) = n - \text{misses}(\sigma) \leq n - N,$$

so the hit rate is maximized under sorted order.

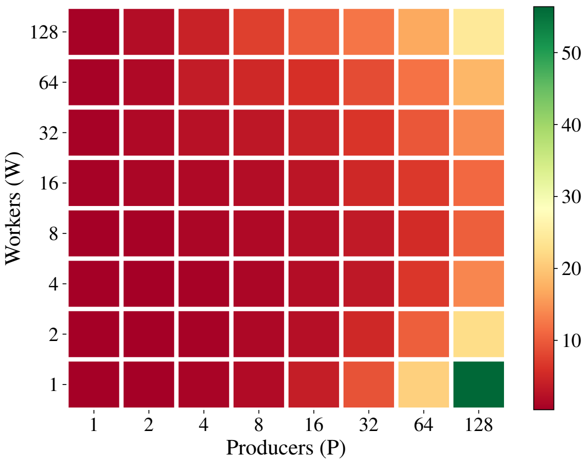
□

2 Extended Experimental Evaluation

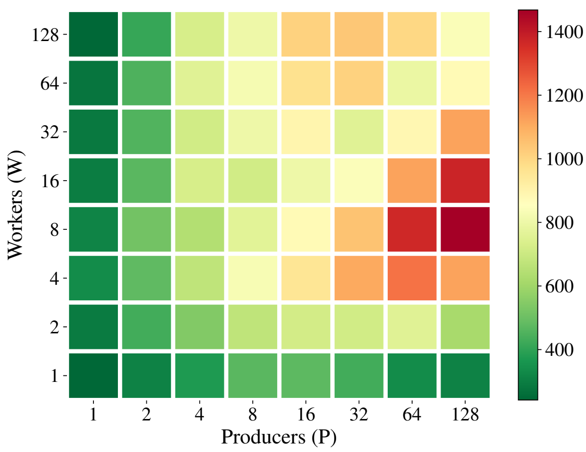
2.1 Worker Threads Configuration



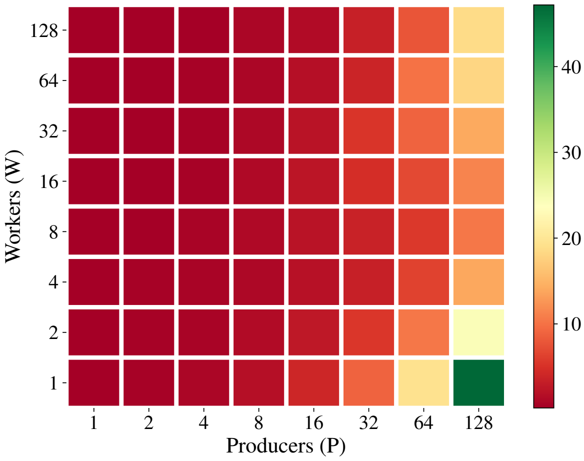
(a) Throughput (KOPS) without page buffer.



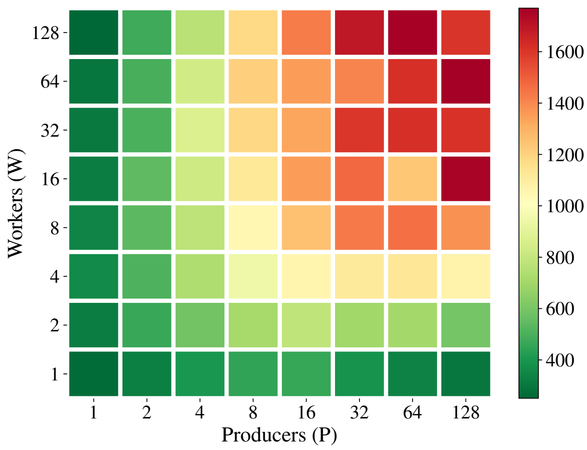
(b) Latency (ms) without page buffer.



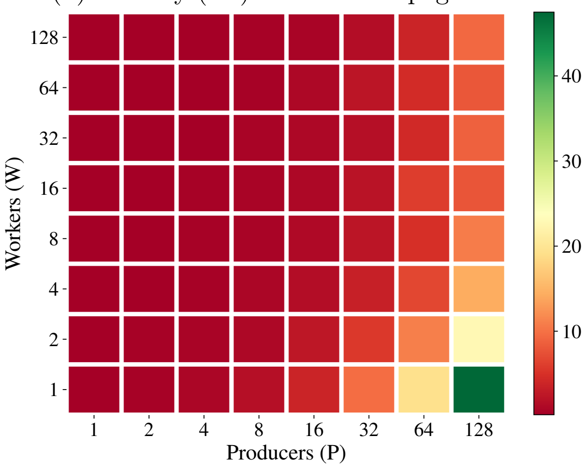
(c) Throughput (KOPS) with 100KB page buffer.



(d) Latency (ms) with 100KB page buffer.



(e) Throughput (KOPS) with 1MB page buffer.



(f) Latency (ms) with 1MB page buffer.

2.2 Performance of FALCON

2.3 Evaluation of CAM