Supplementary Material
Optimal Web-scale Tiering as a Flow Problem

A Practical Issues

A.1 Deferred and Approximate Updates

Assume that we updated z_d at iteration n and we revisit it at iteration n'. This means that z_d at iteration n' is given by applying gradients of $f_\lambda(z_d)$ repeatedly and by moving η in the negative gradient direction. We may compute the aggregate result of all steps by simply adding up the steplengths for each segment, rescaled by the slope λ_j. Denote by

$$s(n) := \sum_{j=1}^{n} \eta_j \text{ and let } \delta(n', n) := s(n') - s(n) \tag{10}$$

the aggregate steps lengths from time n to time n'. Note that λ_t^{-1} is the aggregate steplength required to cross the interval $[t-1, t]$. Algorithm 2 carries out the deferred updates by moving step by step down the slope of f_λ. This is required for invoking the gradient computation and update step of Algorithm 1.

While precomputing the steplength is a significant computational improvement, storing (10) is substantial: a billion steps translate into 4GB of data. This can be remedied by an integral approximation

$$\delta(n', n) = \sum_{j=n+1}^{n'} \eta_j = \sum_{j=n+1}^{n'} \tfrac{1}{\sqrt{j+n_0}} \approx 2\left[\sqrt{n'+n_0} - \sqrt{n+n_0}\right]$$

which becomes increasingly accurate for large $|n'-n|$. It allows us to obtain values for $\delta(n', n')$ in constant time without any storage.

A.2 Data Reduction and Max/Sum Heuristics

The amount of data used in the optimization problem can be reduced significantly by eliminating documents and queries which are definitely assigned to particular tiers.

Consider the case of only two tiers (we only have λ_1): any query occurring more frequently v_q than λ_1 will automatically ensure that the associated pages are cached. Consequently we may remove this query from the dataset, assign all related pages to the first tier $x_d = 0$ and *remove* them from all remaining queries. Secondly, any document d for which $\sum_{q \in Q_d} v_q$ is displayed less than λ_1 will definitely *not* be in the cache. Consequently all queries using d will by default fail and can be removed from the dataset. Note that this thresholding procedure can be repeated with the remaining (so far undetermined) documents and queries.

An analogous reasoning applies to multiple tiers: for any query q with weight $v_q \geq \lambda_t$ we know that all $d \in D_q$ will definitely be stored in tier t or lower — the subgradients with respect to z_d are at least v_q at this tier. Any document which, accumulated over all queries $q \in Q_d$ is not requested more than λ_t times cannot be displayed at t or higher. An appealing side-effect of this data reduction is that the gradients of the remaining functions l_q cover a much smaller dynamic range. This accelerates convergence [11] since optimization progress inversely depends on the gradient range.

Furthermore, both $s_d := \sum_{q \in Q_d} v_q$ and $m_d := \max_{q \in Q_d} v_q$ are good tiering heuristics in their own right. If we had only one page per query the optimal solution would be to sort according to s_d. On the other hand, for large $|D_q|$ ordering documents according to m_d proves near optimal as we see

on both synthetic and real data. This suggests a very simple heuristic for obtaining near-optimal tiering, namely to sort based on m_d. Empirically we found that a good initialization for the page variables z_d to be $-(0.9 \log m_d + 0.1 \log s_d)$ scaled and shifted to fit the $[1, k]$ range, which helps convergence. But if we use Algorithm 3 for extra computational advantage, the constant initialization $z_d = k$ already works efficiently.

B Extensions

We describe three types of extensions on our proposed tiering approach: beyond hit and miss, smoothing and robustness. We will discuss those in turn.

B.1 Beyond Hit and Miss

So far we only discussed a rather primitive model of penalties per query, namely that we would incur a penalty $v_q p_t$ for not serving a query at level t. The motivation for this simplification was twofold — we were interested in finding the optimal tier arrangement for a given set of pages to be retrieved per query and moreover, we did not distinguish between the value of different pages or the possibility of retrieving only a partial set of results per query. In the following we show that considerably more sophisticated score functions still lead to integral solutions.

Lemma 10 *Denote by $\mathcal{S}$ a collection of sets, and by $\lambda_{St}, \gamma_{St} \geq 0$ and $\eta_S \in \mathbb{R}$ weighting coefficients. Then, the optimization problem obtained by replacing $\sum_q v_q \max_{d:(d,q)\in G} z_d$ with*

$$\sum_{S \in \mathcal{S}} \max_{d \in S} \left[\eta_S z_d + \sum_t \lambda_{St} \max(0, t - z_d) + \gamma_{St} \max(0, z_d - t) \right]$$

has an integral solution. ∎

B.2 Smoothing

The approach we discussed so far works well whenever the number of queries significantly exceeds the number of pages in the cache. While the query stream of search engines is obviously tremendous, the above assumption is no longer satisfied when optimizing over hundreds of billions of pages (this would require nearly a Trillion queries to obtain good statistics in the tails).

Assume that each document d comes with a set of features ϕ_d, e.g. its relevance in the Hubs and Spokes model, or alternatively PageRank [9, 12], the indegrees/outdegrees of a page, the likelihood that it is spam, or other content-related information. In this case, one would expect that such information ought to be valuable in deciding at which tier to store a page. We can take advantage of this by modeling $z_d = \langle \phi_d, w \rangle$ for a suitable parameter vector w and a page-feature vector ϕ_d. The resulting optimization problem is convex in w and we can use the same algorithm we used for z_d to optimize over w. Focusing only ϕ_d exclusively, though, is ineffective since it ignores the fact that certain pages simply happen to be popular whereas others simply happen not to be popular at all despite meaningful features ϕ_d. Replacing ϕ_d by $(\phi_d, \nu_d e_d)$, where e_d is the unit vector for document d and ν_d is an indicator variable which characterizes an a-priori estimate of the importance of a page, allows us to have a page-specific weight for common pages whereas for infrequent pages we simply smooth over the prior coefficients.

B.3 Robustness

So far we assumed that v_q is exactly observed. This can be extended to allow for deviations in v by means of robust optimization. The following minimax problem remains convex, hence it is accessible to efficient solution:

$$\underset{z}{\text{minimize}} \underset{\epsilon \in \mathcal{E}}{\text{maximize}} \sum_q \left[(v_q + \epsilon_q) \max_{d \in D_q} z_d \right] + \sum_d f_\lambda(z_d) \tag{11}$$

Here $\epsilon \in \mathcal{E}$ denotes an admissible perturbation of query values, and may be any ℓ_p balls ($0 < p < \infty$) around v, thus including the case of sparse perturbation when $p < 1$.

C Experiments on Synthetic Data

The purpose of experiments on synthetic data is to obtain a small enough dataset which allows us to compare both heuristics, the online solver, and the (much slower) LP solution *exactly*. We generated a random bipartite query-page graph using 150 queries and 150 pages. Each query vertex has a degree of 3, and value $v_q := 10(2+q)^{-0.8}$ mimicking a power law distribution of real data.

We experimented with a 2-tier system by varying the relative size of the prime (cache) tier. We evaluate system performance in *session miss*: for each session q, a miss occurs if any one of the associated pages is not found in cache, incurring v_q misses for that session. The experimental results are summarized in Figure 4. Our proposed method (OPT-tier) outperforms baselines by a significant margin.

To assess the convergence properties of our online algorithm, we compare the quality of the solutions given by linear program (Section 3.1) and online algorithm (Section 3.4). From Figure 4, shows that the online solver (ONL OPT-tier) converges to the solution of linear programming (LP OPT-tier) within few passes over the data. Note that the LP solver is computationally costly, thus unsuitable for problems even at the scale of 1000.

We examine the same synthetic data set for a 3-tier assignment problem. Here we can vary i.e. the relative sizes of the prime tier and the second tier. We report the relative improvement of our tiering algorithm as ratios of (generalized) session misses in Figure 5. As before, our method consistently outperforms the max heristic and, especially the sum heuristic. We observe that the size of the prime tier affects relative improvement more than the size of the second tier.

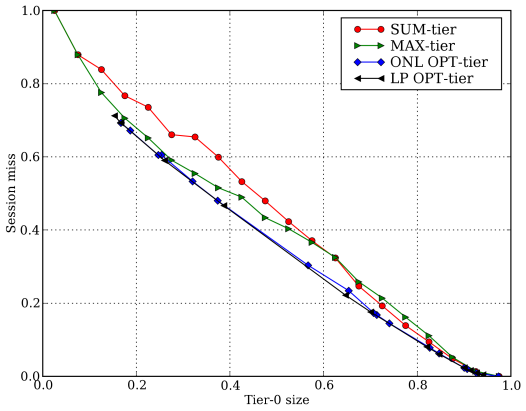

Figure 4: Session miss rate performance on the 150 queries-150 documents with 3 docs/query dataset. The caching performance was rescaled to yield a miss rate of 1 for a cache size of 2.5% for sessions. Our proposed method (OPT-tier) outperforms baselines by a significant margin in the total cache miss rate. The online solver (ONL OPT-tier) converges to the LP solution (LP OPT-tier).

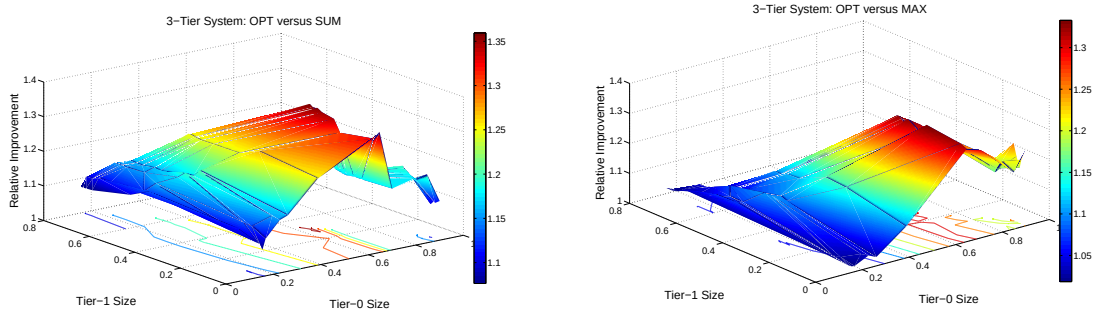

Figure 5: Cache performance for a set of 3 tiers. Our method consistently outperforms the baselines for all choices of both tiers. The difference is most pronounced for large tier sizes where interactions between pages matter most.

D Proofs

Lemma 1 *Assume that $C_k \geq |D| > C_{k-1}$. Then there exists an optimal solution of (2) such that $\sum_d \{z_d \leq t\} = C_t$ for all $1 \leq t < k$.*

Proof Assume that z^*, v^* is the optimal solution. Note that the objective function only depends on v^* directly. If the capacity constraint is not met with equality we may decrease the tiers of an arbitrary set of pages until the constraints are met. Since this only relaxes the constraints on v^* further while not increasing the objective function, the solution is still optimal. ■

Lemma 2 *The solutions of (2) and (4) are equivalent.*

Proof Firstly, the variable sets (z, u) and (x, y) are equivalent (we have an explicit bijection). The same applies to the constraints between them — eq. (4b) implies that the retrieval tier for query q needs to be at least as high as that of the highest page. Finally, the objective function sums over all tier levels from 2 to k such that a document found at tier t will contribute via $p_2 + \ldots + p_t$. Hence equality holds. ■

Lemma 3 *For any choice of λ with $\lambda_t \geq 0$ the linear program (5) has an integral solution, i.e. there exists some x^*, y^* satisfying $x^*_{dt}, y^*_{qt} \in \{0; 1\}$ which minimize (5). Moreover, for $\bar{C}_t = \sum_d x^*_{dt}$ the solution (x^*, y^*) also solves (4).*

Proof We first show that (5) has an integral solution for all choices of λ. This holds since constraints are totally unimodular: the constraint matrix has only one 1 and one -1 entry per row. Integrality follows [8].

By construction, for the choice of $\bar{C}_t = \sum_d x^*_{dt}$ the condition (4c) is met with equality, hence the integral solution of (5) is also the solution of a linear program arising from a relaxation of the integer linear program (4) to a linear program. However, since the relaxation has an integral solution it follows that (x^*, y^*) are also optimal for (4). ■

Lemma 4 *Denote by $L^*(\lambda)$ the value of (5) at the solution of (5) and let $L(\lambda) := L^*(\lambda) + \sum_t \bar{C}_t \lambda_t$. Hence $L(\lambda)$ is concave in λ and moreover, $L(\lambda)$ is maximized for a choice of λ where the solution of (5) satisfies the constraints of (4).*

Proof Subtracting $\sum_t \bar{C}_t \lambda_t$ from the objective of (5) yields a reduced Lagrange function which enforces the constraint $\sum_d x_{dt} \geq \bar{C}_t$. As such, it is concave in λ and at its maximum the capacity constraint is satisfied. ■

Lemma 5 *We may scale p_t and λ_t together by constants $\beta_t > 0$, such that $p'_t/p_t = \beta_t = \lambda'_t/\lambda_t$. The resulting solution of this new problem (6) with (p', λ') is unchanged.*

Proof We introduce Lagrange multipliers γ_{dt} due to constraints of the form $\sum_{t=1}^{k-2} \gamma_{dt}(x_{dt} - x_{d,t+1})$, which can be rewritten as $\sum_{t=1}^{k-1} \alpha_{dt} x_{dt}$. At optimality we know for a given (p, λ) that the gradient of (6) needs to match the Lagrange multipliers (α_{dt}). Denote by x^* and α^* the solution of the optimization problem and the corresponding Lagrange multipliers. Rescaling λ and p as per assumption we see that by rescaling α the optimality conditions still hold. Hence x^* must also solve (5) for (p', λ'). ■

Lemma 6 *Assume that $\bar{C}_t$ is monotonically decreasing and that $p_t = 1$ for $t \geq 1$. Then any choice of λ satisfying the capacity constraints is monotonically non-increasing.*

Proof If $\lambda_t = \lambda_{t+1}$ we arrive at a solution where $x_{dt} = x_{d,t+1}$ since in this case the functions concerning both variables are identical. Moreover, choosing $\lambda_{t+1} > \lambda_t$ can only lead to an increase in $x_{d,t+1}$. However, since $x_{dt} \geq x_{d,t+1}$ by constraint, this means that for any $\lambda_{t+1} \geq \lambda_t$ we have $x_{d,t+1} = x_{dt}$.

Then we may choose $\lambda'_t = \lambda'_{t+1} = \frac{\lambda_t + \lambda_{t+1}}{2}$ and obtain the same solution with a nonincreasing sequence of λ_t: it has the same value of the objective function and moreover the joint subgradients are identical since terms in λ_t and λ_{t+1} are added. A recursive averaging procedure generates a nonincreasing sequence of equivalent values for λ which completes the proof. ■

Lemma 7 *The solution of (8) is equivalent to that of (5).*

Proof

(A) (8) is convex, has a unique minimum value.

(B) There is an injective mapping from any set of variables in (8) to the thermometer code of (5) with the property that the values of the objective function coincide in this case. From this it follows that the minimum of (8) cannot exceed the minimum of (5).

(C) For an integral set of variables in (5) there is an injective map to (8) such that, again, the objective functions coincide. From this it follows that the minimum of (5) cannot exceed the minimum of (8).

Combination of (B) and (C) proves the claim. ■

Lemma 10 *Denote by $\mathcal{S}$ a collection of sets, and by $\lambda_{St}, \gamma_{St} \geq 0$ and $\eta_S \in \mathbb{R}$ weighting coefficients. Then, the optimization problem obtained by replacing $\sum_q v_q \max_{d:(d,q)\in G} z_d$ with*

$$\sum_{S\in\mathcal{S}} \max_{d\in S} \left[\eta_S z_d + \sum_t \lambda_{St} \max(0, t - z_d) + \gamma_{St} \max(0, z_d - t) \right]$$

has an integral solution.

Proof [sketch only] We treat each $S \in \mathcal{S}$ as if it were a query of its own with documents $d \in S$ associated with it. Within each set S note that the score function is piecewise linear with discontinuities occurring only at integers. Hence we may use the same thermometer code decomposition as discussed in Section 2.3 to rewrite the problem in terms of $[0, 1]$ valued variables with totally unimodular constraints. The overall problem has an integral solution. ■