

741 Appendix

742 A Hindsight Optimal Policy for TEL

743 *Proof.* Fix a feasible \mathbf{x} , the optimal \mathbf{u} is given by

$$u_{i,t} = \left(\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - x_{i,t} - \xi \right)^+,$$

744 as otherwise we have $u_{i,t}$ infeasible or can be improved. Thus we can focus on cache decisions \mathbf{x} .

745 We first show that the optimal solution to the optimization problem (I) must satisfies $x_{i,t} \leq$

746 $\left(\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi \right)^+$ for every $i \in [N], t \in \mathcal{A}_i$. Suppose not, i.e., the optimal \mathbf{x}' to

747 (I) satisfies $x'_{i,t} > \left(\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi \right)^+$ for some $i \in [N], t \in \mathcal{A}_i$. Then we have

$$\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - x'_{i,t} - \xi < 0, u'_{i,t} = 0.$$

748 In this case, setting $x_{i,t} = \left(\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi \right)^+$ will not affect the objective value while

749 releasing cache capacity that can be directed to other pages, which contradicts the optimality of \mathbf{x}' .

750 Thus the optimal solution satisfies

$$\begin{aligned} \sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} u_{i,t} &= \sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} \left(\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - x_{i,t} - \xi \right)^+ \\ &= \sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} \mathbb{1} \left\{ \sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi \geq 0 \right\} \cdot \left(\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - x_{i,t} - \xi \right) \\ &\quad + \sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} \mathbb{1} \left\{ \sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi < 0 \right\} \cdot 0 \\ &= \sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} \mathbb{1} \left\{ \sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi \geq 0 \right\} \left(\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi \right) - \sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} x_{i,t} \end{aligned}$$

751 where the last equality holds as $x_{i,t} = 0$ for the case with $\sum_{j=1}^t q_{i,j} + \sum_{j=1}^{t-1} a_{i,j} - \xi < 0$.

752 Therefore, minimizing $\sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} u_{i,t}$ is equivalent to maximizing $\sum_{i \in [N]} \sum_{t \in \mathcal{A}_i} x_{i,t}$ as the

753 first term is a constant. \square

754 B Proof of Theorem 2

755 *Proof.* Let's fix an arbitrary sample path of user prompt traces $\{q_{i,t}\}$ and model response traces
756 $\{a_{i,t}\}$. Let's assume the optional caching setting, as the proof for forced caching setting is similar and
757 in fact simpler, as the caching policy don't consider the arriving conversation when making eviction
758 decisions.

759 Suppose conversation θ arrives and the three policies start with the same cache states. For convenience
760 write $L_\theta = L_\theta + Q + A$, and $\lambda_\theta = \lambda_{\text{turn}}$ denote the length of conversation θ after service and the
761 belief turn rate updated upon its arrival.

762 For conversations with length $L_i \leq \xi - \bar{Q}$ (we assume $\xi > \bar{Q}$),

- 763 • Threshold LRU caches zero tokens by definition of the threshold,

- Tail-Optimized LRU also caches zero because $L_i + \bar{Q} - \xi \leq 0$,
- LRU caches at least zero tokens.

By the next time such conversations arrive, the length of the chat history plus the user prompt is no longer than $L_i + \bar{Q} \leq \xi$, thus the costs (i.e., number of uncached tokens) under all three policies are all zero.

For conversations with length $L_i \geq \xi - \bar{Q}$,

- Threshold LRU first stores the entire history (L_i tokens) then, if necessary, evicts according to LRU;
- Tail-Optimized LRU only stores $L_i + \bar{Q} - \xi$ and then, if necessary, also evicts according to LRU;

Hence, before the common LRU-eviction stage begins, Tail-Optimized LRU has already evicted more tokens than Threshold LRU. As the three policies start with the same cache state, the additional number of tokens to evicted according to LRU are ordered as

$$\text{LRU} \geq \text{ThLRU} \geq \text{TLRU}.$$

Let X_i^{LRU} denote the number of blocks cached under LRU for conversation i .

- If $X_i^{\text{LRU}} \geq L_i + \bar{Q} - \xi$, then we must have

$$X_i^{\text{ThLRU}} \geq X_i^{\text{LRU}} \geq L_i + \bar{Q} - \xi = X_i^{\text{TLRU}},$$

and such an inequality holds until the next time conversation i arrives again, as the three policies use the same priority order at the LRU eviction phase, and the extra tokens evicted by Tail-Optimized LRU before using LRU is always no less than the other two. The costs under all three policies are all zero.

- If $X_i^{\text{TLRU}} < L_i + \bar{Q} - \xi$, then we must have

$$X_i^{\text{TLRU}} \geq X_i^{\text{ThLRU}} \geq X_i^{\text{LRU}},$$

due to the relative ordering of number to evicted according to LRU, and such an inequality holds until the next time conversation i arrives again as well. Thus, the costs under all three policies ordered as

$$(L_i + Q_i - X_i^{\text{TLRU}} - \xi)^+ \leq (L_i + Q_i - X_i^{\text{ThLRU}} - \xi)^+ \leq (L_i + Q_i - X_i^{\text{LRU}} - \xi)^+.$$

Because the comparison was carried out on an arbitrary sample path, the same ordering holds almost surely, which completes the proof. \square

C Proof of Theorem 3

Proof. Given system state λ, L, X , let θ denote the index of the conversation that is the k^{th} arrival with user prompt length Q and model response length A . The finite-horizon value-to-go function is

$$V_k(\lambda, L, X, \theta, Q, A) = \underbrace{(L_\theta + Q - X_\theta - \xi)^+}_{\text{number of uncached blocks above threshold}} + \min_{\mathbf{X}' \in \mathcal{X}(\mathbf{X}, \theta, L_\theta + Q + A)} \mathbb{E}_{\tau, \theta', Q', A'} \left[V_{k+1} \left(\underbrace{\Phi(\lambda, \theta, \tau)}_{\text{belief arrival state transition}}, \underbrace{\Psi(L, \theta, Q + A)}_{\text{conversation length transition}}, \mathbf{X}', \theta', Q', A' \right) \right]$$

with $V_{M+1}(\cdot) = 0$, where the feasible caching decision space is

$$\mathcal{X}(\mathbf{X}, \theta, L) = \{\mathbf{Y} \in \mathbb{N}^{\dim(\mathbf{X}, \theta)} : \sum_i Y_i \leq C, 0 \leq Y_\theta \leq L, 0 \leq Y_i \leq X_i, i \neq \theta\}$$

793 with $\dim(\mathbf{X}, \theta) = \dim(\mathbf{X}) + \mathbb{1}\{\theta > \dim(\mathbf{X})\}$, here $\dim(\mathbf{X})$ denotes the dimension of vector \mathbf{X} ,
 794 and the dimension expands when a new conversation arrives; $\Phi(\boldsymbol{\lambda}, \theta, \tau)$ update the belief turn rate of
 795 conversation θ to $\bar{\lambda}_i$, then discount belief turn rates of all conversations by $\exp(-\mu\tau)$; $\Psi(\mathbf{L}, \theta, Q + A)$
 796 updates the conversation length vector. Specifically, we increase the dimension of \mathbf{L} if necessary (i.e.,
 797 when θ represents a new conversation), and add $Q + A$ to its θ^{th} entry.

798 Let θ denote the index of the conversation that is the k^{th} arrival, with user prompt length Q and
 799 model response length A . Define the belief arrival rate vector as $\boldsymbol{\lambda}$, the conversation length vector as
 800 \mathbf{L} , and the cached token length vector as \mathbf{X} , the cost-to-go function is given by:

$$V_k(\boldsymbol{\lambda}, \mathbf{L}, \mathbf{X}, \theta, Q, A) = (L_\theta + Q - X_\theta - \xi)^+ \\ + \min_{\mathbf{X}' \in \mathcal{X}(\mathbf{X}, \theta, L_\theta + Q + A)} \mathbb{E}_{\tau, \theta', Q', A'} [V_{k+1}(\Phi(\boldsymbol{\lambda}, \theta, \tau), \Psi(\mathbf{L}, \theta, Q + A), \mathbf{X}', \theta', Q', A')]$$

801 with $V_{M+1}(\cdot) = 0$. Let's rewrite $\Phi(\boldsymbol{\lambda}, \theta, \tau) = \Phi(\Gamma(\boldsymbol{\lambda}, \theta), \tau)$ with

- 802 • $\Gamma(\boldsymbol{\lambda}, \theta)$ updates the return rate vector upon the arrival of conversation θ . Specifically, this
 803 operator changes the belief arrival rate of conversation θ to $\bar{\lambda}_i$.
- 804 • $\Phi(\boldsymbol{\lambda}, \tau)$ discount all return rates by $\exp(-\mu\tau)$.

805 The proof is by using induction and argue that if we choose a different caching state than \mathbf{X}^{TLRU} , the
 806 cost will be higher. At last arrival M , $V_M(\boldsymbol{\lambda}, \mathbf{L}, \mathbf{X}, \theta, Q_\theta, A_\theta) = (L_\theta + Q_\theta - X_\theta - \xi)^+$ and any
 807 caching policy is optimal.

808 Suppose this holds for the $k^{th} + 1$ arrival. We proceed to show that the result holds for the k^{th} arrival.
 809 To simplify the notation, we define

$$J_k(\boldsymbol{\lambda}, \mathbf{L}, \mathbf{X}) = \mathbb{E}_{\tau, \theta, Q_\theta, A_\theta} [V_k(\Phi(\boldsymbol{\lambda}, \tau), \mathbf{L}, \mathbf{X}, \theta, Q_\theta, A_\theta)], \tilde{\boldsymbol{\lambda}} = \Gamma(\boldsymbol{\lambda}, \theta), \tilde{\mathbf{L}} = \Psi(\mathbf{L}, \theta, Q_\theta + A_\theta),$$

810 Then we need to prove

$$J_{k+1}(\tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{L}}, \mathbf{X}^{\text{ETLRU}}) \leq J_{k+1}(\tilde{\boldsymbol{\lambda}}, \tilde{\mathbf{L}}, \mathbf{X}')$$

811 To see this, by definition of state transition, suppose the inter-arrival time is τ , the discounted arrival
 812 rates are

$$\tilde{\lambda}_i \cdot \exp(-\mu\tau) \text{ with } \tilde{\lambda}_\theta = \bar{\lambda}_\theta.$$

813 From these expressions, we can conclude that regardless of value of τ , the return rates at next arrival
 814 maintain the same relative ordering as in $\boldsymbol{\lambda}$ for conversations. Note that due to heterogeneous turn
 815 rates across conversations, conversation θ that arrived at the k^{th} arrival may not have the highest
 816 turn rate. Without loss of generality, let's assume $\tau = 0$ and let $\tilde{p}_j = \tilde{\lambda}_j / (\lambda_{\text{conv}} + \sum_i \tilde{\lambda}_i)$ denote the
 817 probability that conversation j returns at the next arrival, and $\tilde{p}_{\dim(\mathbf{X}(1))+1} = \lambda_{\text{conv}} / (\lambda_{\text{conv}} + \sum_i \tilde{\lambda}_i)$
 818 denote the probability that a new conversation starts at the next arrival.

819 We proceed to show this holds for any number of tokens evicted by treating each token eviction
 820 separately. Among all conversations that have arrived so far and have at least one cached token, list
 821 their indices as $i(1), i(2), \dots$ in ascending order of the ranking criterion score

$$\tilde{\lambda}_i \mathbb{P}(\tilde{L}_i + Q_i - \xi \geq X_i),$$

822 Insert conversation θ that just arrives into this ordered list according to its own score

$$\bar{\lambda}_\theta \mathbb{P}(Q_\theta - \xi \geq 0)$$

Let $\mathbf{X}(1)$ denote the cache state that evicts a token from conversation $i(1)$, and $\mathbf{X}(k)$ denote the cache state that evicts a token from conversation $i(k)$ with $k > 1$.

$$\begin{aligned}
& J_{k+1}(\tilde{\lambda}, \tilde{L}, \mathbf{X}(1)) \\
&= \sum_{i \in \dim(\mathbf{X}(1))+1} \tilde{p}_i \mathbb{E}_{Q_i} [(\tilde{L}_i + Q_i - X_i(1) - \xi)^+] \\
&\quad + \tilde{p}_{i(1)} \mathbb{E}_{Q_{i(1)}, A_{i(1)}} \left[\min_{\mathbf{X}'(1) \in \mathcal{X}(\mathbf{X}(1), i(1), \tilde{L}_{i(1)} + Q_{i(1)} + A_{i(1)})} J_{k+2}(\Gamma(\tilde{\lambda}, i(1)), \Psi(\tilde{L}, i(1), Q_{i(1)} + A_{i(1)}), \mathbf{X}'(1)) \right] \\
&\quad + \tilde{p}_{i(k)} \mathbb{E}_{Q_{i(k)}, A_{i(k)}} \left[\min_{\mathbf{X}'(1) \in \mathcal{X}(\mathbf{X}(1), i(k), \tilde{L}_{i(k)} + Q_{i(k)} + A_{i(k)})} J_{k+2}(\Gamma(\tilde{\lambda}, i(k)), \Psi(\tilde{L}, i(k), Q_{i(k)} + A_{i(k)}), \mathbf{X}'(1)) \right] \\
&\quad + \sum_{i \neq i(1), i(k)} \mathbb{E}_{Q_i, A_i} \left[\min_{\mathbf{X}'(1) \in \mathcal{X}(\mathbf{X}(1), i, \tilde{L}_i + Q_i + A_i)} J_{k+2}(\Gamma(\tilde{\lambda}, i), \Psi(\tilde{L}, i, Q_i + A_i), \mathbf{X}'(1)) \right] \\
&\leq \sum_{i \in \dim(\mathbf{X}(1))+1} \tilde{p}_i \mathbb{E}_{Q_i} [(\tilde{L}_i + Q_i - X_i(k) - \xi)^+] \\
&\quad + \tilde{p}_{i(1)} \mathbb{E}_{Q_{i(1)}, A_{i(1)}} \left[\min_{\mathbf{X}'(k) \in \mathcal{X}(\mathbf{X}(k), i(1), \tilde{L}_{i(1)} + Q_{i(1)} + A_{i(1)})} J_{k+2}(\Gamma(\tilde{\lambda}, i(1)), \Psi(\tilde{L}, i(1), Q_{i(1)} + A_{i(1)}), \mathbf{X}'(k)) \right] \\
&\quad + \tilde{p}_{i(k)} \mathbb{E}_{Q_{i(k)}, A_{i(k)}} \left[\min_{\mathbf{X}'(k) \in \mathcal{X}(\mathbf{X}(k), i(k), \tilde{L}_{i(k)} + Q_{i(k)} + A_{i(k)})} J_{k+2}(\Gamma(\tilde{\lambda}, i(k)), \Psi(\tilde{L}, i(k), Q_{i(k)} + A_{i(k)}), \mathbf{X}'(k)) \right] \\
&\quad + \sum_{i \neq i(1), i(k)} \mathbb{E}_{Q_i, A_i} \left[\min_{\mathbf{X}'(k) \in \mathcal{X}(\mathbf{X}(k), i, \tilde{L}_i + Q_i + A_i)} J_{k+2}(\Gamma(\tilde{\lambda}, i), \Psi(\tilde{L}, i, Q_i + A_i), \mathbf{X}'(k)) \right] \\
&= J_{k+1}(\tilde{\lambda}, \tilde{L}, \mathbf{X}(k)),
\end{aligned}$$

where the inequality holds as

- by Definition 1, $\mathbf{X}(1)$ is the optimal solution while $\mathbf{X}(k)$ is a feasible solution, and \tilde{p}_i are proportional to $\tilde{\lambda}_i$, thus

$$\sum_{i \in \dim(\mathbf{X}(1))+1} \tilde{p}_i \mathbb{E}_{Q_i} [(\tilde{L}_i + Q_i - X_i(1) - \xi)^+] \leq \sum_{i \in \dim(\mathbf{X}(1))} \tilde{p}_i \mathbb{E}_{Q_i} [(\tilde{L}_i + Q_i - X_i(k) - \xi)^+]$$

and the expected cost incurred when a new conversation arrives (with probability $\tilde{p}_{\dim(\mathbf{X}(1))+1}$) is $\mathbb{E}_Q[(Q - \xi)^+]$ for both caching state, thus

$$\sum_{i \in \dim(\mathbf{X}(1))+1} \tilde{p}_i \mathbb{E}_{Q_i} [(\tilde{L}_i + Q_i - X_i(1) - \xi)^+] \leq \sum_{i \in \dim(\mathbf{X}(1))+1} \tilde{p}_i \mathbb{E}_{Q_i} [(\tilde{L}_i + Q_i - X_i(k) - \xi)^+]$$

- by induction hypothesis, the optimal $\mathbf{X}'(1)^*$ and $\mathbf{X}'(k)^*$ are given by the optimization problem (6). Fix a user prompt length Q_i and a model response length A_i and we compare the cost-to-do under $\mathbf{X}(1)$ and $\mathbf{X}(k)$.

- if conversation $i(1)$ arrives next, then one need to evict one more token from $\mathbf{X}(1)$ than from $\mathbf{X}(k)$. Suppose the extra token evicted from $\mathbf{X}(1)$ is from conversation $i(k)$, then $\mathbf{X}'(1)^* = \mathbf{X}'(k)^*$. If not, then this means the extra token evicted is from another conversation with better ranking criterion, thus we have

$$\begin{aligned}
& J_{k+2}(\Gamma(\tilde{\lambda}, i(1)), \Psi(\tilde{L}, i(1), Q_{i(1)} + A_{i(1)}), \mathbf{X}'(1)^*) \\
& \leq J_{k+2}(\Gamma(\tilde{\lambda}, i(1)), \Psi(\tilde{L}, i(1), Q_{i(1)} + A_{i(1)}), \mathbf{X}'(k)^*)
\end{aligned}$$

by the induction hypothesis.

- if conversation $i(k)$ arrives next, then one need to evict one more token from $\mathbf{X}(k)$ than from $\mathbf{X}(1)$. In the optional caching model, the extra token evicted form $\mathbf{X}(k)$

840 must be from conversation $i(1)$ by the definition of the ranking of conversations, thus
 841 $\mathbf{X}'(1)^* = \mathbf{X}'(k)^*$.

$$\begin{aligned} & J_{k+2}(\Gamma(\tilde{\lambda}, i(k)), \Psi(\tilde{L}, i(k), Q_{i(k)} + A_{i(k)}), \mathbf{X}'(k)^*) \\ &= J_{k+2}(\Gamma(\tilde{\lambda}, i(k)), \Psi(\tilde{L}, i(k), Q_{i(k)} + A_{i(k)}), \mathbf{X}'(k)^*) \end{aligned}$$

842 – if conversation other than $i(1), i(k)$ arrives next, then $\mathbf{X}'(k)^*$ and $\mathbf{X}'(1)^*$ need to evict
 843 the same number of tokens. If $\mathbf{X}'(k)$ evicts at least one token from conversation $i(k)$,
 844 then $\mathbf{X}'(1)^* = \mathbf{X}'(k)^*$. If not, then this means $\mathbf{X}'(1)^*$ evicts one token from another
 845 conversation with better ranking criterion, thus we have

$$J_{k+2}(\Gamma(\tilde{\lambda}, i), \Psi(\tilde{L}, i, A_i), \mathbf{X}'(1)^*) \leq J_{k+2}(\Gamma(\tilde{\lambda}, i), \Psi(\tilde{L}, i, A_i), \mathbf{X}'(k)^*).$$

846 Therefore, by induction, the result holds for all $k \geq 1$. □

847 **Greedy Implementation.** The optimization problem (6) need not be solved explicitly as a
 848 token-by-token greedy procedure suffices. At a high-level, the algorithm ranks each token by
 849 arrival rates weighted by its counterfactual cost, i.e., the cost increase when we evict this token.

$$\mathbb{P}(L_i + Q_i - \xi \geq X_i) = \mathbb{E}[(L_i + Q_i - \xi - (X_i - 1))^+] - \mathbb{E}[(L_i + Q_i - \xi - X_i)^+]$$

850 i.e., the difference in expected cost if we further evict one token when we have X_i tokens in cache.

Algorithm 2: Expected-Tail-Optimized LRU Policy

Input: Number of conversations N , cache sizes $\{X_i\}$, current lengths $\{L_i\}$, belief turn rates $\{\lambda_i\}$, distribution of length of user prompt $\{Q_i\}$, threshold ξ , arriving conversation θ , arriving user prompt length Q , arriving model response length A , tokens to evict n

Output: Updated cache sizes $\{X_i\}$

evicted $\leftarrow 0$

$L_\theta \leftarrow L_\theta + Q + A$ // Update system state for arriving conversation θ

$X_\theta \leftarrow L_\theta$

$\lambda_\theta \leftarrow \bar{\lambda}_\theta$

for each $i \in E$ **do**

Compute $v_i \leftarrow \lambda_i \cdot \mathbb{P}(L_i + Q_i - \xi \geq X_i)$ // Ranking criterion

while evicted $< n$ **do**

Find $j = \arg \min_{i \in [N], X_i \geq 1} v_i$ // Conversation with minimum value

$X_j \leftarrow X_j - 1$ // Evict one token

evicted \leftarrow evicted $+ 1$

$v_j \leftarrow \lambda_j \cdot \mathbb{P}(L_j + Q_j - \xi \geq X_j)$ // Update ranking criterion

return $\{X_i\}$.

851 In the implementation of the algorithm, one can use min-heap to process which token to evict using
 852 the ranking criterion. The computational complexity of the algorithm is given by $\mathcal{O}(|E| + n \log |E|)$,
 853 where $|E|$ is number of conversations with non-zero cached tokens and n is the number of tokens one
 854 needs to evict.

855 We show that Algorithm 2 indeed returns a cache state that is an optimal solution to the optimization
 856 problem (6).

857 **Lemma 1.** Algorithm 2 returns an optimal solution to the optimization problem (6).

858 *Proof.* We prove by contradiction. Note that it is possible for the algorithm to return multiple optimal
 859 solutions, and it is also possible for the optimization problem (6) to have multiple optimal solutions.
 860 Suppose not, then the two set of solutions do not intersect. Let \mathbf{X}^* denote one optimal solution. Then
 861 there must exist two conversations i, j such that $X_j^* \geq 1$ and

$$\lambda_i \mathbb{P}(L_i + Q_i - \xi \geq X_i^* + 1) > \lambda_j \mathbb{P}(L_j + A_j - \xi \geq X_j^*),$$

862 Then we can construct another solution \mathbf{X}' such that $X'_k = X_k^*$ for $k \neq i, j$, and $X'_i = X_i^* + 1$, $X'_j =$
863 $X_j^* - 1$. Then the difference between the objective values of \mathbf{X}' and \mathbf{X}^* is given by

$$\begin{aligned} \text{OBJ}(\mathbf{X}') - \text{OBJ}(\mathbf{X}^*) &= \lambda_i \mathbb{E}[(L_i + Q_i - \xi - (X_i^* + 1))^+] + \lambda_j \mathbb{E}[(L_j + Q_j - \xi - (X_j^* - 1))^+] \\ &\quad - (\lambda_i \mathbb{E}[(L_i + Q_i - \xi - X_i^*)^+] + \lambda_j \mathbb{E}[(L_j + Q_j - \xi - X_j^*)^+]) \\ &= \lambda_j \mathbb{P}(L_j + A_j - \xi \geq X_j) - \lambda_i \mathbb{P}(L_i + Q_i - \xi \geq X_i + 1) \\ &< 0, \end{aligned}$$

864 which contradicts the optimality of \mathbf{X}^* . \square

865 D Discussion on Forced Caching

866 **Implementation of Tail-Optimized LRU.** To implement forced caching, especially at GPU level,
867 the server needs to decide which block to evict as serving the turn. The server may not know the total
868 number of cache blocks to evict due to the uncertainty in the model response length, nevertheless the
869 server can repeatedly call our algorithm to evict more tokens if needed.

870 **Hindsight optimal policy.** To model forced caching, we replace optional caching constraint (4) with

$$x_{i,t+1} = \sum_{j=1}^t (q_{i,j} + a_{i,j}), \forall i \in [N], t \in \mathcal{A}_i \text{ and } t < T, \quad (9)$$

871 i.e., when a turn arrives, the server is required to cache its whole chat history including newly
872 generated response. Theorem 1 continues to hold under forced caching.

873 **No-Worse-Than-LRU Guarantee** Theorem 2 continues to hold under forced caching.

874 **Expected-Tail-Optimized LRU.** To model forced caching, we replace feasible caching decision
875 space under optional caching with

$$\mathcal{X}_{\mathcal{F}}(\mathbf{X}, \theta, L) = \{\mathbf{Y} \in \mathbb{N}^{\dim(\mathbf{X}, \theta)} : \sum_i Y_i \leq C, Y_\theta = L, 0 \leq Y_i \leq X_i, i \neq \theta\}.$$

876 Theorem 3 continues to hold under forced caching, i.e., Expected-Tail-Optimized LRU remains to be
877 optimal, if

- 878 • every future prompt (if it arrives) has a known, fixed length $Q \geq 0$. Here this fixed length
879 can be heterogeneous across conversations and across turns. Crucially, the decision-maker
880 still does not know if any given conversation will return; they only know that should it
881 return, its next-turn question length will be Q . In this case, Expected-Tail-Optimized LRU
882 is reduced to a deterministic version as stated in Algorithm 1.
- 883 • conversations have homogeneous turn rates λ_{turn} .

884 This fixed-prompt-length assumption holds when prompts are pre-specified. When $Q = 0$ after the
885 first turn and responses also have zero length, our model reduces to classic paging with unit page size.

886 E Additional Figures

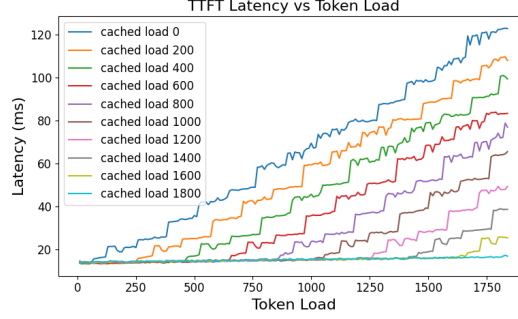


Figure 3: Experimental results demonstrating the linear fit. We used Vicuna-7B with vLLM’s prefix-caching enabled on a Colab A100 GPU. The plot shows TTFT latency as a function of total prompt length and cached-prefix size.

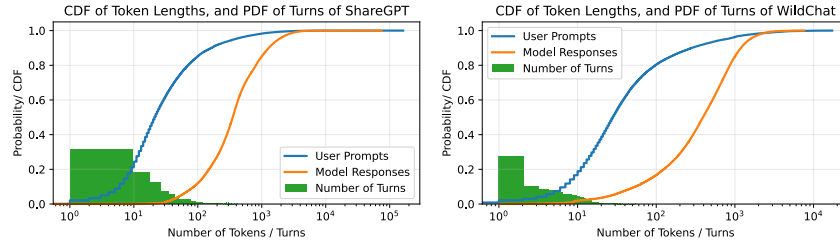


Figure 4: Distributions of turns and tokens of ShareGPT [Contributors, 2025] and WildChat [Zhao et al., 2024] datasets.

F Results on ShareGPT with Synthetic Timestamps

ShareGPT [Contributors, 2025] does not include timestamps of each request, thus we generate them with the stochastic model described in Section 4. Specifically, for each conversation, we draw exponential inter-arrival times with rate $\lambda_{\text{conv}} = 1$, then for each conversation, we generate inter-arrival times between each turn within a conversation using Exponential distribution with rate $\lambda_{\text{turn}} = 3$. The average prompt length in ShareGPT is approximately 100 tokens (we thus set $\hat{Q} = 100$ in implementation), with an average of 3.5 turns per conversation.

Tables 4–5 show that Tail-Optimized LRU still beats both LRU and Threshold-LRU: it trims P90 by up to 10%, and P95 by up to 7%. The smaller improvement compared to the ones observed in WildChat (Tables 1–2) stem from the already-high base latencies under LRU (with capacity $C = 1000$ under LRU, medium, P90, P95, P99 tail latencies are roughly 209 ms, 1415 ms, 2447 ms, 3649 ms), thus percentage improvements shrink.

Table 4: Relative latency improvement of T-LRU over LRU with various ξ (ShareGPT)

Capacity	$\xi = 50\text{ms}$		$\xi = 100\text{ms}$		$\xi = 200\text{ms}$		$\xi = 300\text{ms}$		$\xi = 500\text{ms}$	
	p90	p95	p90	p95	p90	p95	p90	p95	p90	p95
1000	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.9%	0.6%	0.9%	0.9%
2000	0.7%	0.0%	0.7%	0.2%	0.9%	1.5%	1.4%	2.0%	2.7%	2.3%
4000	0.6%	0.6%	0.7%	1.8%	2.0%	2.5%	2.5%	3.0%	4.2%	3.5%
6000	0.7%	1.3%	2.1%	2.9%	4.9%	3.6%	8.1%	4.2%	10.0%	4.7%
8000	1.5%	0.9%	2.6%	1.8%	3.5%	2.5%	4.8%	3.6%	9.6%	5.0%
10000	0.9%	0.7%	3.6%	1.7%	4.3%	2.9%	5.1%	3.6%	9.0%	6.9%

Using a 200 ms SLO, T-LRU cuts the share of requests above the budget by 2–8% across capacities (Table 6). Improvements again peak when ξ is near the desired percentile; extremely high ξ trades those mid-tail wins for heavier protection of the extreme tail, echoing the WildChat pattern.

Lastly, in spite of the extra foresight, End-Aware T-LRU and Length-Aware T-LRU show only marginal gains over T-LRU, and all three policies perform very closely to Tail-Optimized Belady, the

Table 5: Relative latency improvement of T-LRU over Threshold-LRU with various ξ (ShareGPT)

	$\xi = 50\text{ms}$		$\xi = 100\text{ms}$		$\xi = 200\text{ms}$		$\xi = 300\text{ms}$		$\xi = 500\text{ms}$	
Capacity	p90	p95	p90	p95	p90	p95	p90	p95	p90	p95
1000	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.9%	0.6%	0.9%	0.9%
2000	0.7%	0.0%	0.7%	0.2%	0.9%	1.5%	1.4%	2.0%	2.7%	2.3%
4000	0.6%	0.6%	0.7%	1.8%	2.0%	2.5%	2.5%	3.0%	4.2%	3.5%
6000	0.7%	0.8%	2.1%	2.4%	4.9%	3.1%	8.1%	3.7%	10.0%	4.2%
8000	0.8%	0.3%	2.0%	1.2%	2.9%	2.0%	4.1%	3.0%	9.0%	4.5%
10000	0.9%	0.6%	3.6%	1.6%	4.3%	2.8%	5.1%	3.5%	9.0%	6.8%

Table 6: Relative improvement of T-LRU: % reduction in requests with latency > 200ms (ShareGPT)

Capacity	$\xi = 50\text{ms}$		$\xi = 100\text{ms}$		$\xi = 150\text{ms}$		$\xi = 200\text{ms}$		$\xi = 500\text{ms}$	
	LRU	Thre-LRU	LRU	Thre-LRU	LRU	Thre-LRU	LRU	Thre-LRU	LRU	Thre-LRU
1000	0.7%	0.0%	1.7%	1.0%	2.3%	1.6%	2.3%	1.6%	-3.5%	-4.2%
2000	1.1%	1.0%	1.9%	1.8%	2.6%	2.5%	3.1%	3.0%	-8.6%	-8.7%
4000	1.8%	1.3%	3.9%	3.4%	4.7%	4.2%	4.8%	4.3%	-15.6%	-16.2%
6000	1.9%	1.1%	4.7%	3.9%	6.0%	5.2%	4.7%	3.9%	-26.2%	-27.2%
8000	1.2%	0.9%	4.3%	4.0%	4.9%	4.6%	2.9%	2.6%	-39.3%	-39.7%
10000	2.2%	1.8%	6.5%	6.1%	7.9%	7.6%	3.3%	3.0%	-50.7%	-51.2%

904 optimal hindsight policy that minimizes the Tail Excess Latency. This is exactly what our stochastic
 905 model predicts: under Poisson arrivals, LRU’s recency order is already a near-perfect proxy for
 906 “furthest in the future”, the rule the hindsight policy uses for eviction, so extra foresight offers
 907 diminishing returns.

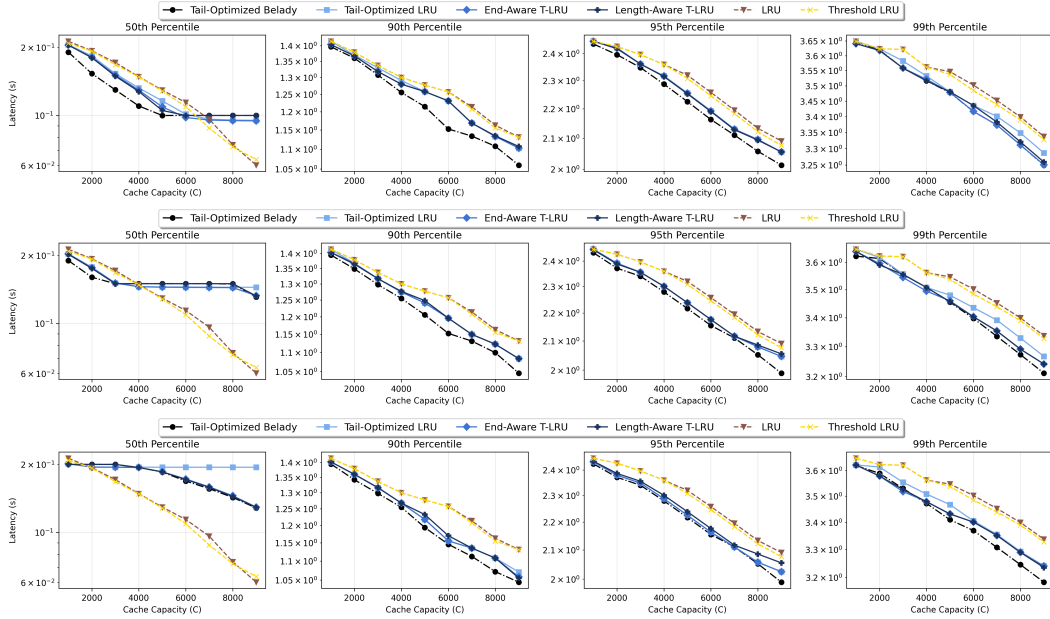


Figure 5: Latency results for various settings (threshold latency $\xi = 100, 200, 300$ ms) from top to bottom panels (ShareGPT)