Optimal Probabilistic Cache Stampede Prevention

Andrea Vattani
Goodreads, Amazon Inc.
188 Spear St.
San Francisco, California
avattani@goodreads.com

Flavio Chierichetti[*]
Dipartimento di Informatica
Sapienza University
Rome, Italy
flavio@di.uniroma1.it

Keegan Lowenstein
Bugsnag Inc.
30 7th St.
San Francisco, California
keegan@bugsnag.com

ABSTRACT

When a frequently-accessed cache item expires, multiple requests to that item can trigger a cache miss and start regenerating that same item at the same time. This phenomenon, known as cache stampede, severely limits the performance of databases and web servers. A natural countermeasure to this issue is to let the processes that perform such requests to randomly ask for a regeneration before the expiration time of the item. In this paper we give optimal algorithms for performing such probabilistic early expirations. Our algorithms are theoretically optimal and have much better performances than other solutions used in real-world applications.

1. INTRODUCTION

The cache stampede problem (also called dog-piling, cache miss storm, or cache choking) is a situation that occurs when a popular cache item expires, leading to multiple requests seeing a cache miss and regenerating that same item at the same time.

This issue is a consequence of the typical pattern used for dealing with a cache miss (see Figure 1): a cache item request checks whether the item is cached, and regenerates the item if it is not present. The flaw with this approach is that many different requests may see that cache miss at the same time and regenerate the item in cache through a potentially expensive computation. The number of requests seeing the cache miss (and therefore contributing to the stampede) depends not only on the request rate, but also on the time needed to recompute the item. For example, if the cache item is accessed 10 times per second, and the recomputation of the item takes 3 seconds, then 30 requests will recompute the item.

Having several re-computations of the same cache item, apart from being wasteful, often leads to overloading of the system or slow-down of the database, which is also why that particular item was cached in the first place. A cache stampede is often referred to as a cascading failure because several concurrent re-computations will increase the time of each individual recomputation by bogging down the system, in turn causing even more requests to be part of the stampede.

Due to the widespread nature and the threats associated with this problem, multiple approaches have been proposed to mitigate cache stampedes.

- *External re-computation*: Rather than regenerating the cache item by the requests themselves upon cache expiration, have a separate background process to periodically regenerate the item. This solution prevents cache stampedes all together, but it is often discarded because of the burden of maintaining an external process (need to enforce availability of this daemon process, need for monitoring, code separation/repetition, etc.) This becomes even more daunting when dealing with multiple cache items, as it involves keeping track of which items to periodically re-compute and when — in fact, the background process would even regenerate cache items that were never requested (which can be, in some settings, a waste of processing time).

- *Locking*: Upon a cache miss, a request attempts to acquire a lock for that cache key, and regenerates the item only if it acquires it. Depending on how the lock mechanism is implemented (atomically or not), this approach may mitigate cache stampedes or completely prevent them. One issue with this approach is that all requests that would have been part of a stampede (apart from the one acquiring the lock) have no cache item to return. There are a few options: have the client handle the absence of the item properly; have the requests not acquiring the lock wait for the item to be regenerated; or, keep a stale item in the cache to be used while the new value is generated. Apart from this issue, this approach requires one extra write for the locking mechanism (doubling the number of write operations), tuning a time-to-live for the lock itself (high enough to recompute the item, but less than the re-computation frequency), and implementing a locking mechanism. Finally, this approach is not fault-tolerant: if the request acquiring the lock fails while re-generating the item, no item or a stale item will be served until the lock expires and a new lock is acquired.

[*]Supported in part by a Google Focused Research Award, and by the MIUR-PRIN National Project "ARS TechnoMedia" (Algorithmics for Social Technological Networks).

Proceedings of the VLDB Endowment, Vol. 8, No. 8
Copyright 2015 VLDB Endowment 2150-8097/15/04.

- *Probabilistic early expiration*: Each individual request may regenerate the item in cache before its expiration by making an independent probabilistic decision. The probability of performing an early expiration increases as the request time gets closer to the expiration of the item. This approach is depicted in Figure 2: as the figure shows, each request essentially pretends to be sometime in the future and if at that time the cache is expired then it regenerates the item. Here, the leap in the future depends on the probabilistic choice. Since the probabilistic decision is made independently by each request, the effect of the stampede is mitigated as less requests will expire at the same time. The most amenable feature of this approach is its simplicity. The drawbacks are the choice of the probabilistic decision (i.e. how to pick the distribution $\mathcal{D}$ in Figure 2); and the lack of guarantees for its effectiveness, in terms of assessing both the stampede reduction and when early expirations happen (i.e. how much earlier than the desired expiration). For example, if the item in cache contains hourly statistics, it is important not to regenerate these stats too much earlier than the hour mark.

In this paper we show how probabilistic early expiration can be made extremely effective. In particular, we present a simple instantiation of the probabilistic decision with the exponential function $\text{Exp}(\lambda)$, which we demonstrate to be optimal. A fundamental property we show is that the parameter λ needs *not* to depend on the rate of requests in order to effectively reduce stampedes. This property makes our solution very attractive in that, in its simplest form, it requires no parameter tuning to perform effectively. This implementation, which we call XFetch (for eXponential fetch), is detailed in Section 5. A higher request rate for a fixed λ, while not affecting the stampede size, will cause earlier expirations of the item in cache, but we show that this dependency is very moderate.

The general problem of efficiently keeping a cache of frequently accessed (but dynamically-changing) results has been studied in many guises and settings, both from an applied and a theoretical point of view, e.g.: Web (e.g., dynamic web pages and search results [3, 4]), networking (e.g., routers' look-up tables [14]), databases (e.g., [2]). The cache stampede problem strains many software systems [1, 15], specifically those based on decentralized cache value recomputations (e.g., distributed web servers responding to web requests); these systems are usually backed by caches with primitive get/set operations, and which do not provide locking mechanisms, or protection against stampedes (a notable example is the widely-used distributed caching system Memcached [8]). A number of systems use a probabilistic early expiration strategy to avoid stampedes. Notably, a probabilistic early expiration strategy is used in Perl's unified Cache Handling Interface (CHI) [11] — a module which is part of many web-applications (e.g., Drupal [5, 10]).

CHI uses the uniform distribution to implement early expiration — as documented in [11], by allowing programmers to set the length of the interval on which the uniform distribution will be defined. In this paper, we will show that the uniform distribution is far from being optimal, while the exponential distribution is much more efficient (on all axes) and close to optimality.

```
function Fetch($key, ttl$)
    $value \leftarrow$ CacheRead($key$)
    if !$value$ then
        $value \leftarrow$ RecomputeValue()
        CacheWrite($key$, $value$, $ttl$)
    end
    return $value$
end
```

Figure 1: Typical pattern for retrieving an item in cache with a time-to-live. Here *ttl* is the time-to-live of the cache item: after CacheWrite$(key, value, ttl)$ is called, the *key* will be present in the cache for *ttl* units of time after which it will expire. The call to RecomputeValue() is typically expensive.

The rest of the paper is organized as follows. We start off with describing a framework to model stampedes in Section 2. Our results are then detailed in Section 3. The main part of our analysis is in Section 4. Section 5 contains some implementation notes, and Section 6 contains the results of our experiments. The appendix contains the proofs missing from the main body.

All the logarithms in this paper are assumed to be natural (that is, $\log e = 1$, where e is the Napier's constant).

2. MODEL

In this section we define a general framework to model the effect of stampedes and the efficacy of probabilistic early expirations.

Without loss of generality, we restrict our attention to an arbitrary item in cache whose expiration time we assume to be τ. We assume that the recomputation of the item takes one unit of time[1]. We will use some of the basic terminology of queueing theory, and of probability theory — the concepts that we will use can be found in many text-books (e.g., [6, 9]).

2.1 Process rate

To capture the notion of process rate, we consider a (possible infinite) sequence of processes accessing the item in cache at non-increasing times $\{s_i\}_i$. Given a certain $n > 0$ representing the *rate* of processes, we model the inter-arrival times $(s_i - s_{i-1})$ as independent samples from a non-negative distribution with expectation $\frac{1}{n}$, so that in average n processes will access the item in a single unit of time.

Definition 1 (Process arrival) *Let $\mathcal{I}$ be an arbitrary non-negative distribution with expectation 1 and standard deviation $\sigma_{\mathcal{I}}$. The process arrival is defined by inter-arrival times drawn independently from $\frac{\mathcal{I}}{n}$. Hence, inter-arrivals have mean $\frac{1}{n}$ and standard deviation $\frac{\sigma_{\mathcal{I}}}{n}$.*

As a notable example, the well-known Poisson point process [13] of inter-arrival mean $\frac{1}{n} = \frac{\sigma_{\mathcal{I}}}{n}$ satisfies $\sigma_{\mathcal{I}} = 1$ (that is, mean and standard deviation are equal).

It is important to notice that since the recomputation of the item takes one unit of time and the process arrival is such that n processes access the item in a unit of time, we have stampedes of n processes (in average) if no form of stampede prevention is implemented.

[1]This is without loss of generality as we can simply scale the process rate to allow for a different recomputation time. See Section 5 for a more detailed discussion.

```
function FETCH($key, ttl$; $\mathcal{D}$)
    $value, expiry \leftarrow$ CACHEREAD($key$)
    $gap \sim \mathcal{D}$
    if !$value$ or TIME() + $gap \geq expiry$ then
        $value \leftarrow$ RECOMPUTEVALUE()
        CACHEWRITE($key$, $value$, $ttl$)
    end
    return $value$
end
```

Figure 2: Probabilistic early expiration of a cache item. Here, $\mathcal{D}$ is a probability distribution with non-negative support (i.e. $gap \geq 0$). The variable *expiry* represents the time at which the *key* expires from the cache.

2.2 Probabilistic early expirations

We now proceed with formalizing *probabilistic early expirations.* As previously discussed, this approach attempts to mitigate stampedes by having the processes possibly regenerate the item in cache before it expires, where this decision is taken probabilistically and independently by each process. As depicted in Figure 2, this probabilistic choice can be interpreted as each process pretending to be sent sometime in the future and checking if at that time the item would be expired. It is evident that the crux of this approach lies in choosing a distribution $\mathcal{D}$ to stochastically decide the *time gap* each process employs.

Definition 2 (Gap distribution) *A gap distribution $\mathcal{D}$ is any distribution defined over $t \in \mathbb{R}^{\geq 0}$.*

For example, the Cache Handling Interface (CHI) for Perl, sets $\mathcal{D}$ to the uniform distribution over the interval $[0, \xi]$, where ξ is a user-specified parameter to tune the tradeoff between stampede prevention and how early expirations can happen.

For a given gap distribution $\mathcal{D}$, we have that a process accessing the item in cache (with expiration τ) at time s, will regenerate it in either of the two following cases:

(i) *Early expiration*: When $s < \tau$ and $s + Y \geq \tau$, where Y is sampled from $\mathcal{D}$.

(ii) *Regular expiration*: When $s \geq \tau$. The cache item is expired at this point so the process needs to refresh it.

Observe that as the cache access time s gets closer to τ, the probability $\Pr_{Y\sim D}(s + Y > \tau)$ of an early expiration increases. For example, in the case of CHI, this probability increases linearly as s approaches τ.

2.3 Effectiveness

We finally proceed with defining how effective a gap distribution $\mathcal{D}$ is. On one hand, early expirations may reduce large stampedes since only a fraction of the processes accessing the item will stampede on an early expiration; at the same time, we also do not want to expire the item too much earlier than the desired expiration time. These are the two quantities that we care about.

Definition 3 (Stampede size) *Fix an inter-arrival distribution $\mathcal{I}$, a process rate n and a gap distribution $\mathcal{D}$. Let $Z = Z(\mathcal{I}, n, \mathcal{D})$ be the random variable denoting the first time a process regenerates the item. The* stampede size *$S_{\mathcal{I},n,\mathcal{D}}$ is the number of processes re-generating the item in the time interval $[Z, Z + 1)$.*

Observe that if $Z \geq \tau$ (that is, no process regenerates the item before it expires) then we run into a *regular expiration* causing a stampede of roughly n processes during the time interval $[Z, Z+1)$. Analogously, if $Z = \tau - \epsilon$, with $0 < \epsilon < 1$, then there will be a stampede of roughly $(1 - \epsilon) \cdot n$ processes in the time interval $[\tau, \tau + (1 - \epsilon))$.

Definition 4 (Early expiration gap) *Fix an inter-arrival distribution $\mathcal{I}$, a process rate n and a gap distribution $\mathcal{D}$. Let $Z = Z(\mathcal{I}, n, \mathcal{D})$ be the random variable denoting the first time a process regenerates the item. The* early expiration gap *$T_{\mathcal{I},n,\mathcal{D}} = \max\{\tau - Z, 0\}$ is how much earlier than the regular expiration time the early expiration occurred (or zero if no early expiration occurs).*

A low early expiration gap is particularly important in applications where the cache item contains some periodic (typically hourly or daily) statistics.

Note that both stampede size and early expiration gap are random variables that depend on the randomness of the process distribution $\mathcal{I}$ and of the gap distribution $\mathcal{D}$.

Intuitively, if the gap distribution $\mathcal{D}$ allows for very early expirations, then it should be more effective in reducing large stampedes. On the other hand, if it only allows expirations close to the desired expiration of the item, then stampedes are more likely to be large. How effective a gap distribution $\mathcal{D}$ is a combination of these two criteria.

Definition 5 (Effectiveness) *Fix an inter-arrival distribution $\mathcal{I}$. Then we say that a gap distribution $\mathcal{D}$ is (s, γ)-effective if*

1. $E[S] \leq (1 + o_n(1))s$
2. $E[T] \leq (1 + o_n(1))\gamma$

(In the definition above the "little-o" notation $o_n(1)$ hides factors going to 0 as n grows[2].)

Consider a scenario where early expirations are never done. This scenario is obtained by a gap distribution $\mathcal{D}_0$ that assigns probability one at $t = 0$ and zero otherwise. Then we have that $\mathcal{D}_0$ is $(n, 0)$-effective, as the early expiration gap is always zero but the n processes (in expectation) accessing the cache between time τ and $\tau + 1$ will all regenerate the cache item.

The core problem addressed in this paper is whether we can find a distribution $\mathcal{D}$ which gives the best of both worlds. That is, a distribution $\mathcal{D}$ which can substantially reduce the size of the stampedes, while keeping the early expiration gap low.

3. OUR RESULTS

We first consider the *uniform distribution* $\mathcal{D} = U(0, \xi)$, used for instance by the Perl Cache Handling Interface (CHI). While, by tuning ξ, this distribution is able to reduce stampedes by increasing the early expiration gap, we have that this trade-off has a linear dependence in ξ. In particular, we can show the following result.

[2]By definition, $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Theorem 6 (Uniform) *The uniform distribution $U(0,\xi)$ is no better than $(\frac{n}{2\xi},\xi)$-effective.*

This means that if we want to reduce the stampede size to $\sqrt{n}$, then we need to allow for expirations as early as $\sqrt{n}/2$. In practice this may not been good enough: for example, suppose we have a frequently-accessed cache item that gets recomputed once a day and whose recomputation takes one minute; then if we have $10,000$ processes accessing the item in a minute, we would need to allow expirations as early as 50 minutes to reduce the stampede size to 100 processes.

We then propose to use an *exponential distribution* of parameter λ whose implementation, XFETCH, is extremely simple (see Section 5). We are able to show that the exponential distribution is able to drastically reduce the size of a stampede while keeping expirations close to the desired expiration time.

Theorem 7 (Exponential) *The exponential distribution* $\mathrm{Exp}(\lambda)$ *is* $\left((e^{\lambda}-1)(\frac{1}{\lambda}+\frac{1}{e}),\frac{1}{\lambda}\log n\right)$*-effective.*

For example, for $\lambda=1$, we obtain early expiration gap of size $\log n$ with stampedes of size $e-\frac{1}{e}$. Going back to our example above, an exponential distribution with $\lambda=1$ would reduce the stampede size to merely $e-\frac{1}{e}\approx 2.4$ processes without having expirations earlier than $\log_e(10,000)\approx 9.2$ minutes.

It is natural to ask whether there exists a distribution that would guarantee a constant early expiration gap (rather than $\log n$) while keeping the stampede low to a constant. We show that this is not possible and, in fact, that the exponential distribution is optimal in this respect.

Theorem 8 (Optimality) *Consider any distribution $\mathcal{D}$ that is independent of n. If, for each n, $\mathcal{D}$ has early expiration gap at most $\frac{1}{\lambda}\log n$, then it has expected stampede size at least $e^{\Omega(\lambda)}$.*

For concreteness, observe that Theorem 8 implies that the Exponential distribution is optimal in the full range of λ — e.g., if we want the early expiration gap to be at most $\log\log n$, then the expected stampede size has to be at least $e^{\Omega(\log n/\log\log n)}$, and the Exponential distribution matches this bound. Observe that the above optimality theorem holds if $\mathcal{D}$ is independent of the process rate n — that is, if the algorithm cannot make any guess on the process rate. In Section 7, we will see that if we have an approximate knowledge of n, then we can make the early expiration gap smaller than $O(\log n)$ while keeping the expected stampede size to a constant.

We then evaluate the performance of XFETCH on real-world and synthetic datasets. Our experiments show that our approach out-performs current methods from all angles even when $\lambda=1$. In addition, the experimental results show that our approach is very robust to bursts.

4. ANALYSIS

In this section we assume that the item in cache we consider has expiration time τ. We start with a couple of definitions that will turn out useful for the analysis.

Definition 9 (y-early process) *For $y\geq 0$, we say that a process is y-*early *if it arrives at time $\tau-y$. In other words, a y-early process is a process that accesses the item in cache y units of time before its expiration time.*

Definition 10 (Early expiration probability) *Fix a gap distribution $\mathcal{D}$. We define $f_{\mathcal{D}}(y)$ as the probability that a y-early process performs an early expiration when sampling from $\mathcal{D}$.*

$$f_{\mathcal{D}}(y)=\Pr_{Y\sim\mathcal{D}}(Y>y)=1-\Pr_{Y\sim\mathcal{D}}(Y\leq y).$$

4.1 Uniform distribution

In this section we instantiate the gap distribution $\mathcal{D}$ with the uniform distribution $U(0,\xi)$; this will serve both as a warm-up for some of the techniques that we will use in the paper, as well as a proof of Theorem 6. By definition of $U(0,\xi)$, we have that

$$f_{U(0,\xi)}(y)=\begin{cases}1-\frac{y}{\xi}, & \text{for } 0\leq y\leq\xi\\ 0, & \text{for } y>\xi\end{cases}$$

Observe how the probability of a process performing an early expiration increases *linearly* as the time approaches the item expiration.

We will show that the uniform distribution fails to achieve good efficiency even for the simple case where the process inter-arrival times are all equal to $\frac{1}{n}$, that is when the process inter-arrivals distribution is such that $\sigma_{\mathcal{I}}=0$.

The following lemma shows that the early expiration gap tends to ξ as n grows.

Lemma 11 (Early expiration gap for $U(0,\xi)$) *Let $T=T_{U(0,\xi)}$ be the early expiration gap. For any $a=a(n)>0$, we have that*

$$\Pr\left(T\leq\left(1-\frac{a}{n}\right)\xi\right)\leq\frac{e^{-(a^2\xi/n-1)}}{1-\frac{a}{n}}$$

PROOF. Consider any $0\leq y\leq\xi$. In order to have early expiration gap $T\leq y$, it must be that no x-early process with $y\leq x\leq\xi$ performs an early expiration. The probability that a x-early process does not perform expiration is $1-f_{U(0,\xi)}(x)$. Since the cadence of the processes is exactly $\frac{1}{n}$, we can write

$$\begin{aligned}\Pr(T\leq y)&\leq\prod_{i=0}^{n(\xi-y)}\left(1-f_{U(0,\xi)}\left(\xi-\frac{i}{n}\right)\right)\\&=\prod_{i=0}^{n(\xi-y)}\left(1-\frac{i}{n\xi}\right)\\&=\exp\left(\sum_{i=0}^{n(\xi-y)}\log\left(1-\frac{i}{n\xi}\right)\right)\\&\leq\exp\left(\int_{z=0}^{n(\xi-y)-1}\log\left(1-\frac{z}{n\xi}\right)dz\right),\end{aligned}$$

where the last step holds since $\log\left(1-\frac{i}{n\xi}\right)$ is decreasing in i.

By solving the integral, we get

$$\begin{aligned}\Pr(T\leq y)&\leq\exp\left(-(ny+1)\log\frac{y}{\xi}+(ny+1)-n\xi\right)\\&=\left(\frac{y}{\xi}\right)^{-ny-1}e^{-n(\xi-y)+1}\end{aligned}$$

By substituting $y = (1 - \frac{a}{n})\xi$, we can conclude

$$\begin{aligned}\Pr\left(T \le \left(1 - \frac{a}{n}\right)\xi\right) &\le \left(1 - \frac{a}{n}\right)^{-n\xi(1-\frac{a}{n})-1} e^{-a\xi+1} \\ &\le \frac{e^{\frac{a}{n}n\xi(1-\frac{a}{n})}e^{-a\xi+1}}{1-\frac{a}{n}} \\ &= \frac{e^{-(a^2\xi/n-1)}}{1-\frac{a}{n}},\end{aligned}$$

where the second inequality holds since $1 - x \le e^{-x}$ for any $0 \le x \le 1$. □

Applying the lemma above with $a = a(n) = \sqrt{\frac{1+\log n}{n}}$, we can conclude that the early expiration gap is at least $(1 - \sqrt{\frac{\log n}{n}})\xi = (1 - o(1))\xi$ with probability at least $1 - O(\frac{1}{n})$. This also implies that $E[T] \ge (1 - o(1))\xi$.

To establish Theorem 6, it is left to show that the stampede size decreases linearly with ξ, which we do in the following lemma.

Lemma 12 (Stampede size for $\boldsymbol{U(0,\xi)}$) *Let $S = S_{U(0,\xi)}$ be the stampede size. Then,*

$$E[S] \ge \frac{n}{2\xi}$$

PROOF. Since $f_{U(0,\xi)}(y)$ increases as y decreases and the early expiration gap cannot be more than ξ, we have that the size of the stampede starting exactly ξ units of time before the expiration is a lower bound on the size of any stampede. Hence,

$$E[S] \ge \sum_{i=1}^{n} f_{U(0,\xi)}\left(\xi - \frac{i}{n}\right) = \sum_{i=1}^{n} \frac{i}{n\xi} = \frac{n+1}{2\xi}.$$

□

4.2 Exponential distribution

In this section we instantiate the gap distribution $\mathcal{D}$ with the exponential distribution $\text{Exp}(\lambda)$. By Definition 10 and the fact that $\Pr(Y \le y) = 1 - e^{-\lambda y}$ for $Y \sim \text{Exp}(\lambda)$, we have that the probability that a y-early process regenerates the item in cache is

$$f_{\text{Exp}(\lambda)}(y) = 1 - \Pr_{Y\sim\text{Exp}(\lambda)}(Y \le y) = e^{-\lambda y}.$$

Since we are dealing with the general class of inter-arrival distributions as per Definition 1, we are going to use Chebyshev's lemma to bound the number of processes present in a specific interval.

Lemma 13 (Chebyshev [6, 9]) *Let X be a random variable with finite expected value μ and finite non-zero variance σ^2. Then for any real number $k > 0$,*

$$\Pr(|X - \mu| \ge k) \le \frac{\sigma^2}{k^2}.$$

The following lemma uses Chebyshev's bound to establish that with high probability the number of processes present in any interval of unitary length is tightly concentrated around its expectation n. (The proofs of all claims in this section are deferred to the Appendix.)

Lemma 14 (Concentration) *Let the processes be distributed according to the inter-arrival distribution from Definition 1. Consider any interval of unit length and let N be the number of processes in the interval. For any real number $\delta > 0$, we have*

$$\Pr(N \ge (1+\delta)n) \le \frac{(1+\delta)\sigma_{\mathcal{I}}^2}{\delta^2 n}.$$

For any real number $0 < \delta < 1$, we have

$$\Pr(N \le (1-\delta)n) \le \frac{(1-\delta)\sigma_{\mathcal{I}}^2}{\delta^2 n} \le \frac{\sigma_{\mathcal{I}}^2}{\delta^2 n}.$$

The following lemma shows that the early expiration gap will not exceed $\nu = \frac{1+\epsilon}{\lambda}\log n$ with high probability. The crucial observation in the proof is that each of the (roughly) n processes in the interval right before this threshold performs an early expiration with probability at most $f_{\text{Exp}(\lambda)}(\nu) = \frac{1}{n^{1+\epsilon}}$, so the probability that at least one of them will regenerate the item is at most $\frac{1}{n^\epsilon}$.

Lemma 15 (Early expiration gap for Exp($\boldsymbol{\lambda}$)) *Let $T = T_{\text{Exp}(\lambda)}$ be the early expiration gap. Then for any $\epsilon > 0$, $\delta > 0$,*

$$\Pr\left(T > \frac{1+\epsilon}{\lambda}\log n\right) \le \frac{1+\delta}{n^\epsilon(1-e^{-\lambda})} + O\left(\frac{\sigma_{\mathcal{I}}^2}{\delta^2 n}\right)$$

The analysis of the stampede size is more subtle. It essentially establishes that the probability of an expiration as early as $i + \frac{1}{\lambda}\log n$ is roughly $e^{-e^{\lambda i}/(e^\lambda - 1)}$ and entails stampedes of size $e^{\lambda i}$. The expected stampede size is then $\sum e^{-e^{\lambda i}/(e^\lambda-1)}e^{\lambda i} \approx (e^\lambda - 1)(\frac{1}{\lambda} + \frac{1}{e})$.

Lemma 16 (Stampede size for Exp($\boldsymbol{\lambda}$)) *Let $S = S_{\text{Exp}(\lambda)}$ be the stampede size. Then, for any $\delta > 0$,*

$$E[S] \le \left(1 + O\left(\frac{\sigma_{\mathcal{I}}^2 \log n}{\delta^2 n}\right)\right) \cdot \frac{e^\lambda - 1}{1-\delta}\left(\frac{1}{\lambda} + \frac{1}{e}\right).$$

Theorem 7 follows by having ϵ slowly approach zero (e.g. $\epsilon = \frac{\log\log n}{\log n}$) in Lemma 15 (implying an expected early expiration gap of at most $(1 + o(1))\frac{1}{\lambda}\log n$); and having δ approach zero (e.g. $\delta = \frac{1}{\log n}$) in Lemma 16.

4.3 Optimality

We will use a Poisson point process as model for inter-arrival distribution [13]. That is, in the notation of Definition 1, we have $\mathcal{I} = \text{Exp}(1)$ with $\sigma_{\mathcal{I}} = 1$. For a process rate n, we then have that the number X of processes in any unit interval is distributed like a Poisson random variable with parameter n: $\Pr(X = k) = \frac{n^k e^{-n}}{k!}$.

Consider any gap distribution $\mathcal{D}$ independent of n. By Definition 10, $f_{\mathcal{D}}(y)$ is the probability that a y-early process performs an early expiration according to the distribution $\mathcal{D}$. Assuming $f_{\mathcal{D}}(y)$ is integrable[3], let $p_i = \int_i^{i+1} f_{\mathcal{D}}(y)\,dy$, for each integer $i \ge 0$ — that is, p_i is the probability that some y-early process, where y is chosen uniformly at random in $(i, i+1]$, performs an early expiration. We will show that p_i cannot be much larger than what it is with the exponential distribution, unless the early expiration gap is larger than $\log n$.

[3]Observe that $f_{\mathcal{D}}(y)$ is integrable in Riemann terms if it is monotone, or has finitely many discontinuities; $f_{\mathcal{D}}(y)$ is integrable in Lebesgue terms if it has at most countably many discontinuities.

```
function XFETCH($key, ttl$; $\beta = 1$)
    $value, \Delta, expiry \leftarrow$ CACHEREAD($key$)
    if !$value$ or TIME() $- \Delta\beta\log($RAND$()) \geq expiry$ then
        $start \leftarrow$ TIME()
        $value \leftarrow$ RECOMPUTEVALUE()
        $\Delta \leftarrow$ TIME() $-$ $start$
        CACHEWRITE($key$, $(value, \Delta)$, $ttl$)
    end
    return $value$
end
```

Figure 3: Simple implementation of cache stampede prevention with exponential gap distribution $\mathcal{D} = \text{Exp}(\frac{1}{\beta})$. The parameter β defaults to 1 and already provides effective prevention against cache stampedes. It can be increased for even better guarantees against stampedes, if earlier expirations are not a concern.

Lemma 17 *Fix any $\epsilon > 0$. Suppose that, for each $n \geq 2$, the early expiration gap $T_{\mathcal{D}}$ satisfies $E[T_{\mathcal{D}}] \leq \epsilon \log(n-1)$. Then $p_i \leq e^{-\left(1-e^{-1/3}\right)\frac{i}{\epsilon}}$ for each integer $i \geq 0$.*

We will now use Lemma 17 (which upper bounds the probability that a process, whose arrival is chosen uniformly at random in a window of size 1, performs an early expiration) to prove Lemma 18 — which, then, directly entails Theorem 8, our main lower bound.

Lemma 18 *Let $\epsilon > 0$ be small enough. If, for each $n \geq 2$, the early expiration gap $T_{\mathcal{D}}$ satisfies $E[T_{\mathcal{D}}] \leq \epsilon \log(n-1)$ then the expected stampede is at least $e^{\Omega(1/\epsilon)}$.*

5. IMPLEMENTATION NOTES

In this section we present an explicit implementation of our cache stampede prevention approach, XFETCH (taking its name by our use of the eXponential function).

The discussion thus far assumed (without loss of generality) that the recomputation of the item in cache takes one unit of time. This allowed our analysis to have the gaps sampled from Exp(λ) to be independent of a particular time unit. In practice though, the gaps we sample from Exp(λ) needs to be scaled by the recomputation time. Figure 3 shows how this time Δ can be recorded upon recomputation of the item and stored as part of the cache value. The scaled gap $-\Delta\beta\log(\text{RAND}())$ corresponds to sampling from $\mathcal{D} = \text{Exp}(\frac{1}{\beta})$ and scaling by a factor Δ.

The pseudo-code assumes that the cache server is able to provide the expiration time (*expiry*) of the item upon a cache read of the corresponding key. If this is not the case, this expiration can be stored also as part of the cache value, which will then become $(value, \Delta, \text{TIME}() + ttl)$. Also, to avoid accumulating the gaps given by the early expirations, we can simply adjust the $ttl \leftarrow ttl + (expiry - \text{TIME}())$ right before the cache write.

6. EXPERIMENTS

In this section we describe experimental results based on the results discussed thus far.

For our experiments we use a real dataset that consists of a week of requests for a popular cache item used in the `www.goodreads.com` website. In particular the cache item in

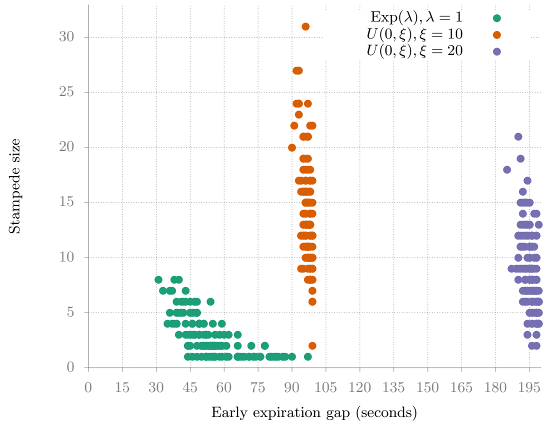

Figure 4: Scatterplot – regeneration time is 10s.

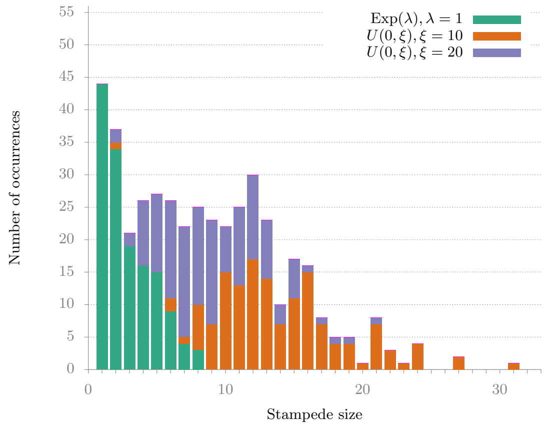

Figure 5: Stacked histogram of distribution of stampede sizes – regeneration time is 10s.

consideration is a *hourly* statistic for the most popular tags used in all of the user-created quotes. The recomputation of this cache item takes about 10 seconds. The inter-arrival time of the requests in this dataset is about 0.07s with a standard deviation of 0.25s. In the terminology of Definition 1, this means that $\sigma_{\mathcal{I}} \approx (0.25/0.07) \approx 3.6$. Since the time to regenerate the item is about 10 seconds, we have that the process rate is $n \approx 140$ in average.

Figure 4 shows a scatter plot where each data point for a specific distribution corresponds to the regeneration of the cache item with respect to a specific hour-long interval during the week of data. Specifically, consider a data point (x, y) of a specific distribution $\mathcal{D}$ and say it corresponds to a specific hour-long interval ending at time τ. Then this data point signifies that out of all processes in that interval, the first one to perform an early expiration did so x seconds before τ, and caused a stampede of size y (that is, $y - 1$ more processes performed an early expiration during the 10-second window starting at time $\tau - x$).

The exponential gap distribution Exp(λ) with $\lambda = 1$ clearly outperforms the uniform distribution $U(0, \xi)$ with $\xi = 10$, both in terms of stampede size and early expiration gap. Even when allowing the uniform distribution to perform ex-

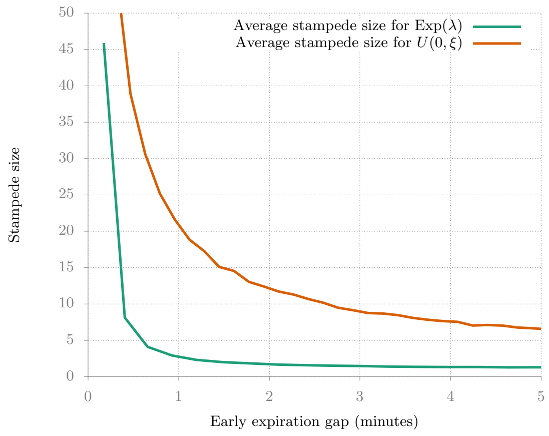

Figure 6: Stampede size as a function of the early expiration gap – regeneration time is 10s.

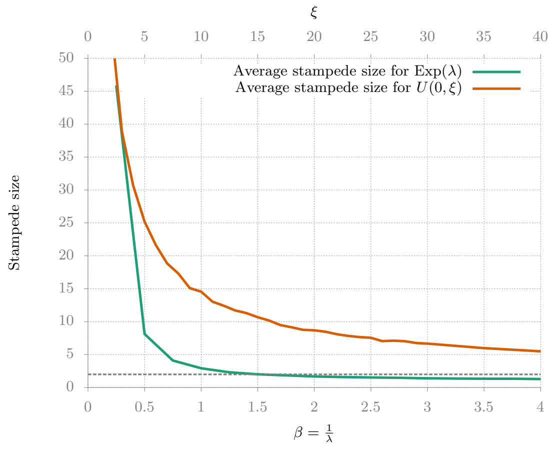

Figure 7: Stampede size as a function of the distribution parameter – regeneration time is 10s.

pirations twice as early, by setting $\xi = 20$ (which has been suggested to be a good choice of parameter for the uniform distribution in CHI [11,12]), we still get stampedes of much larger size than the exponential distribution. This fact is made more clear in Figure 5 where we show the distribution of stampede sizes. For the exponential distribution, most stampedes have size 1 (i.e., no stampede) or 2, and no stampede is larger than 8. On the hand, the uniform distribution shows average stampede size closer to 10 with occasional dangerous stampedes of size 20 or more.

Figure 6 shows the average stampede size as a function of the average early expiration gap where the average is taken over all the hour-long intervals with 100 trials per interval. The different values are obtained by varying the distribution parameter. It is striking how stampedes of size less than 10 (respectively, less than 5) are achieved with expirations that are less than 20 seconds (respectively, less than 40 seconds) early, especially considering that regenerating the item in cache takes 10 seconds. Analogously, Figure 7 shows the average stampede size as a function of the distribution parameter. The dashed horizontal line is at $y = 2$, and shows that increasing $\beta = \frac{1}{\lambda}$ to 1.5 already drops the

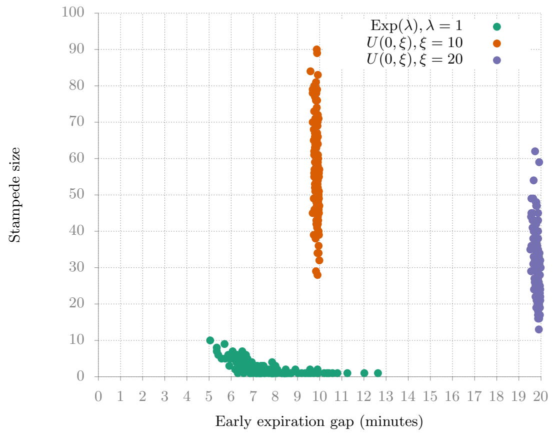

Figure 8: Scatterplot – regeneration time is 1 minute

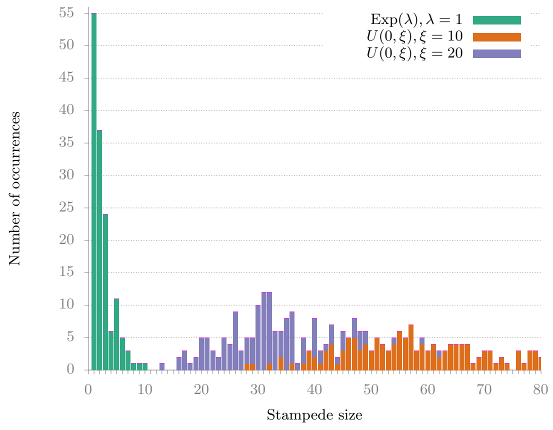

Figure 9: Stacked histogram of distribution of stampede sizes – regeneration time is 1 minute

average stampede size to less than 2.

Using the same dataset, we now move to the scenario where the time to regenerate the item is 1 minute. Note that this could also be viewed as increasing the process rate. Indeed, in this case we obtain a process rate of $n \approx 840$ in average, which could potentially lead to dreadful stampedes. Figures 8-9 show that the exponential function Exp(λ) with $\lambda = 1$ is still as effective as before in preventing stampedes, with no stampede larger than 10. On the other hand, a higher rate heavily penalizes the uniform distribution which exhibits alarming stampedes of size over 80 and 50, for $\xi = 10$ and $\xi = 20$, respectively. Figures 10-11 complete the picture by showing average stampede size under 5 with early expiration gap less than 5 minutes, still when using Exp(λ) with $\lambda = 1$.

6.1 Bursts

In this section we demonstrate the robustness of our approach to sudden bursts of requests. We generate a synthetic sequence of requests using the following model of bursts [7]: each time interval is either in a "low" or a "high" state, and from an interval to the next we change state with probability p. In the low (resp. high) intervals, processes are generated

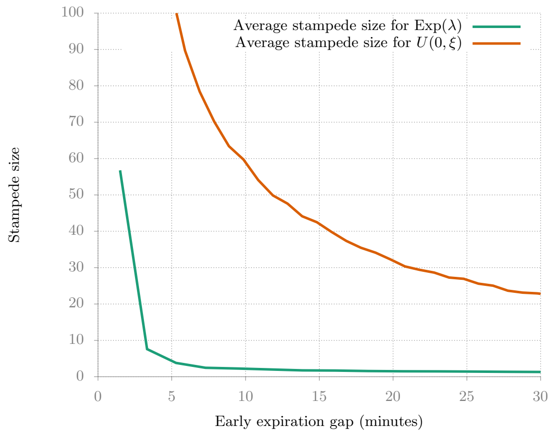

Figure 10: Stampede size as function of the early expiration gap – regeneration takes 1 minute.

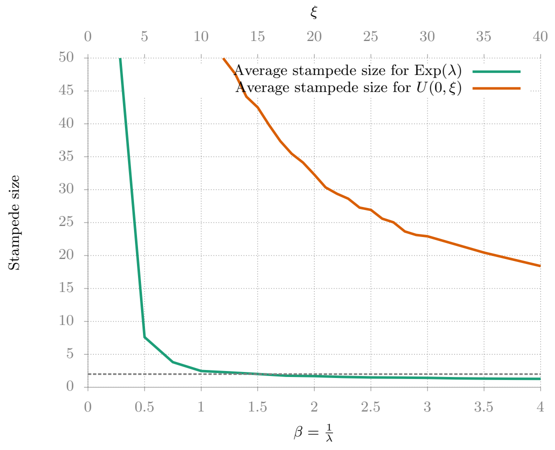

Figure 11: Stampede size as function of distribution parameter – regeneration takes 1 minute.

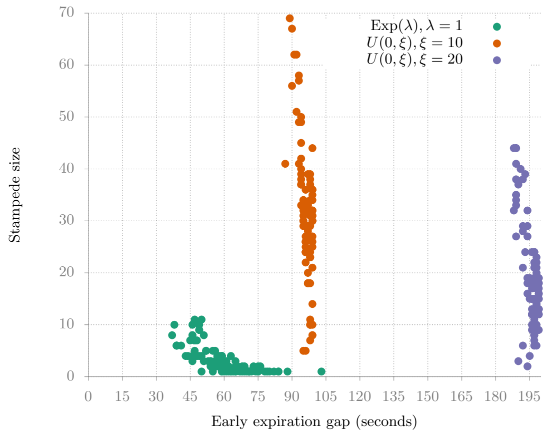

Figure 12: Scatterplot for bursty data

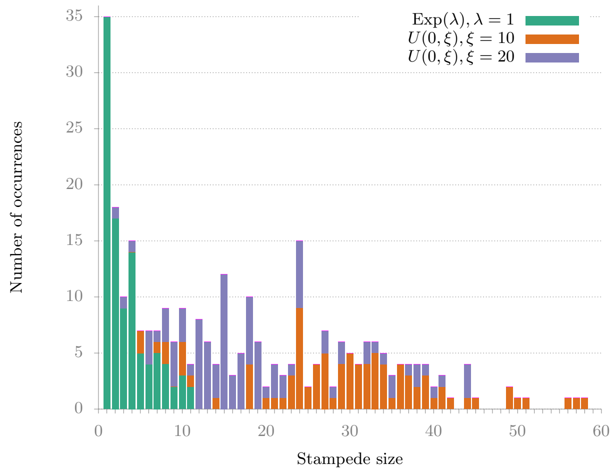

Figure 13: Stacked histogram of distribution of stampede sizes for bursty data.

with a Poisson point process of rate n_{low} (resp. n_{high}). For our experiments, we test our approach against aggressive bursts by setting $n_{\text{low}} = 50, n_{high} = 500$ and $p = 0.1$. Our intervals are of length 10 seconds, which we also use as the time to regenerate the cache item.

Figures 12-13 show how the exponential function is practically immune to sudden bursts, especially when comparing this behavior with that of Figures 4-5 (where $n \approx 140$). On the other hand, the comparison show that the uniform distribution suffers of higher fluctuations in stampede size, with peaks around 70 and 45 for $\xi = 10$ and $\xi = 20$, respectively).

7. EXTENSIONS: KNOWN RATE

In this section we propose algorithms that work under the assumption that the process rate is approximately known. This knowledge of n will allow us to beat the theoretical lower bound proved in Theorem 8, which deals with algorithms that are oblivious of n.

We will start by showing that, if we have a constant multiplicative approximation of n, then we can decrease the early expiration gap to $O(\log\log n)$ while keeping the expected stampede to a constant independent of n.

We will then move on to show that, with a very precise knowledge of n, the early expiration gap can be lowered to $O(\log^\star n)$ — that is, to the *iterated natural logarithm* of n: the number of times one can apply the $\log(\cdot)$ function, starting from n, and until reaching a non-positive value[4]. The iterated logarithm grows extremely slowly — e.g., to get $\log^\star n \geq 3$, it is necessary that $n > 3814279$, and to get $\log^\star n \geq 4$, n has to be larger than $10^{1656520}$.

This other algorithm is then, in theory, much more effective than the other when the process rate n is known in advance; unfortunately, though, the process rate is unstable in real applications. Thus, we believe that the iterated-logarithm algorithm is not going to be effective in practice. In terms of our goal of understanding the limits of what a cache-stampede prevention (early expiration) strategy can achieve, though, it is quite useful to describe and analyze this algorithm.

We now describe our approximately-known rate algorithm that gives a $O(\log\log n)$ expected early expiration gap. Es-

[4]The iterated natural logarithm $\log^\star n$ equals 0 if $n \leq 1$, and equals $1 + \log^\star(\log n)$ otherwise.

sentially, we will modify the (exponential) gap distribution of Theorem 7 to get a smaller expiration gap, assuming some knowledge of n. Suppose that $\hat{n}$ is our guess of n. Then, the distribution $\mathcal{D} = \mathcal{D}_{\hat{n}}$ will be defined as follows:

$$\mathcal{D} = \begin{cases} \text{Exp}(\lambda) & \text{with probability } \frac{\log \hat{n}}{\hat{n}} \\ 0 & \text{with probability } 1 - \frac{\log \hat{n}}{\hat{n}}. \end{cases}$$

Observe that, for $y > 0$, whenever a y-early process samples $\mathcal{D} = 0$, then it can be disregarded from an early expiration point of view as it will not trigger a recomputation. Intuitively, then, our gap distribution $\mathcal{D}$ reduces the process rate from n to $O(\log n)$. Our analysis of the exponential gap distribution can then be used to bound the performance of $\mathcal{D}$.

Lemma 19 *Suppose $\frac{n}{\alpha} \le \hat{n} \le \alpha n$, for some $\alpha \ge 1$. If $\alpha < e^{o(\log \log n)}$, then the distribution $\mathcal{D}_{\hat{n}}$ is*

$$\left(\left(e^{\lambda} - 1\right) \left(\frac{1}{\lambda} + \frac{1}{e} \right), \frac{1}{\lambda} \log \log n \right) \text{-effective.}$$

Finally, we state our Lemma about the best strategy we know of for the case of known process rates.

Lemma 20 *There exists a known-rate strategy with $O(1)$ expected stampede size, and $O(\log^{\star} n)$ expiration gap.*

8. CONCLUSIONS

In this paper we presented XFETCH, an effective approach against cache stampedes based on probabilistic early expirations. Our approach is extremely simple to implement and requires no parameter tuning. Using an analysis based on general stochastic request distributions, we show that our approach is immune to high frequency of requests in terms of reducing stampedes, and that the relationship with how early the expirations are performed is optimal. Experimental results on real-world and synthetic datasets demonstrate how our approach out-performs current methods and also show its robustness to bursts of requests.

APPENDIX

PROOF OF LEMMA 14. If $N \ge (1+\delta)n$, it must be that $\sum_{i=1}^{(1+\delta)n} X_i \le 1$, where X_i are i.i.d. and distributed according to the inter-arrival distribution, that is have mean $\frac{1}{n}$ and standard deviation $\frac{\sigma_{\mathcal{I}}}{n}$. If $X = \sum_{i=1}^{(1+\delta)n} X_i$, then we have $\mu_X = E[X] = (1+\delta)$ and $\sigma_X^2 = Var(X) \le \frac{(1+\delta)\sigma_{\mathcal{I}}^2}{n}$. Using Lemma 13,

$$\begin{aligned} \Pr(N \ge (1+\delta)n) &\le \Pr(X \le 1) = \Pr(\mu_X - X \ge \mu_X - 1) \\ &\le \frac{(1+\delta)\sigma_{\mathcal{I}}^2}{n(\mu_X - 1)^2} = \frac{(1+\delta)\sigma_{\mathcal{I}}^2}{\delta^2 n}. \end{aligned}$$

The other case is analogous. □

PROOF OF LEMMA 15. Fix any $\epsilon = \epsilon(n) > 0$, and let $\nu = \frac{1+\epsilon}{\lambda} \log n$. For any $i \ge 0$, let E_i be the event that $T \in (\nu + i, \nu + i + 1]$. Then, by a union bound we have $\Pr(T > \nu) \le \sum_{i \ge 0} \Pr(E_i)$.

Set $\ell_i = e^{\lambda i/2}$, and let N_i denote the number of y-early processes with $\nu + i \le y \le \nu + i + 1$. Fix any $\delta > 0$, and let $B_{i,j}$ be the event $\frac{N_i}{(1+\delta)n} \in [j, j+1)$.

$$\Pr(E_i) \le \sum_{j=0}^{\ell_i} \Pr(E_i | B_{i,j}) \Pr(B_{i,j}) + \Pr\left(\frac{N_i}{(1+\delta)n} \ge (\ell_i + 1) \right).$$

We can bound $\Pr(E_i | B_{i,j}) < (j+1)(1+\delta)ne^{-\lambda(\nu+i)} = (j+1)(1+\delta)e^{-\lambda i}n^{-\epsilon}$. For $1 \le j \le \ell_i$, this is at most $O(\frac{\ell_i e^{-\lambda i}}{n^{\epsilon}}) = O(\frac{e^{-\lambda i/2}}{n^{\epsilon}})$. For $1 \le j \le \ell_i$, we can apply Lemma 14 to obtain $\Pr(B_{i,j}) \le \Pr\left(\frac{N_i}{(1+\delta)n} \ge 1\right) \le \frac{(1+\delta)\sigma_{\mathcal{I}}^2}{\delta^2 n} = O(\sigma_{\mathcal{I}}^2 \delta^{-2} n^{-1})$. Finally, for $\Pr\left(\frac{N_i}{(1+\delta)n} \ge (\ell_i + 1)\right)$, Lemma 14 yields that this probability is $O(\sigma_{\mathcal{I}}^2 n^{-1} \ell_i^{-1}) = O(\sigma_{\mathcal{I}}^2 n^{-1} e^{-\lambda i/2})$. Combining these observations,

$$\begin{aligned} \Pr(E_i) &\le \frac{(1+\delta)}{n^{\epsilon}} e^{-\lambda i} + O(\sigma_{\mathcal{I}}^2 \delta^{-2} n^{-(1+\epsilon)} e^{-\frac{\lambda i}{2}}) + O(\sigma_{\mathcal{I}}^2 n^{-1} e^{-\frac{\lambda i}{2}}) \\ &= \frac{(1+\delta)}{n^{\epsilon}} e^{-\lambda i} + O(\sigma_{\mathcal{I}}^2 \delta^{-2} n^{-1}) e^{-\lambda i/2}. \end{aligned}$$

Finally, the closed formula for geometric series yields

$$\Pr(T > \nu) \le \sum_{i \ge 0} \Pr(E_i) = \frac{1+\delta}{n^{\epsilon}(1 - e^{-\lambda})} + O\left(\frac{\sigma_{\mathcal{I}}^2}{\delta^2 n} \right).$$

□

Claim 21 *Let $T = T_{\text{Exp}(\lambda)}$ be the early expiration gap and $\nu = \frac{1}{\lambda} \log n$. Then, for any integer $-\nu \le i \le \nu$ and real $0 < \delta < 1$, we have*

$$\Pr\left(T \in (\nu - i - 1, \nu - i)\right) \le \sum_{m=0}^{2\nu} \left(\frac{2\nu\sigma_{\mathcal{I}}^2}{\delta^2 n} \right)^m e^{-(1-\delta)\frac{e^{\lambda(i-m)} - \frac{1}{n}}{e^{\lambda} - 1}}.$$

PROOF. In order to have $T \in (\nu - i - 1, \nu - i)$, it must be that no y-early process with $y \ge \nu - i$ performs an early expiration. Then, if A_j is the event that no y-early process with $\nu - (j+1) \le y \le \nu - j$ performs an early expiration, we can write

$$\Pr(T \in (\nu - i - 1, \nu - i)) \le \prod_{j=-\nu}^{i-1} \Pr(A_j).$$

Consider any $0 < \delta < 1$. To bound $\Pr(A_j)$, we use Lemma 14 to get a lower bound of $(1-\delta)n$ on the number N_j of y-early processes with $\nu - (j+1) \le y \le \nu - j$. We then use the fact that for each of these $(1-\delta)n$ processes the probability of performing an early expiration is at least $e^{-\lambda(\nu - j)}$.

$$\begin{aligned} \Pr(A_j) &= \Pr(A_j | N_j \ge (1-\delta)n) \Pr(N_j \ge (1-\delta)n) \\ &\quad + \Pr(A_j | N_j \le (1-\delta)n) \Pr(N_j \le (1-\delta)n) \\ &\le \Pr(A_j | N_j \ge (1-\delta)n) + \frac{\sigma_{\mathcal{I}}^2}{\delta^2 n} \\ &\le \left(1 - e^{-\lambda(\nu - j)}\right)^{(1-\delta)n} + \frac{\sigma_{\mathcal{I}}^2}{\delta^2 n} \\ &= \left(1 - \frac{e^{\lambda j}}{n}\right)^{(1-\delta)n} + \frac{\sigma_{\mathcal{I}}^2}{\delta^2 n} \\ &\le e^{-(1-\delta)e^{\lambda j}} + \frac{\sigma_{\mathcal{I}}^2}{\delta^2 n}, \end{aligned}$$

where we used the fact that $e^{\lambda\nu} = \frac{1}{n}$ and the fact that $1 - x \le e^{-x}$. The above implies that

$$\begin{aligned}
&\Pr(T \in (\nu - i - 1, \nu - i)) \\
&\quad \le \prod_{j=-\nu}^{i-1} \left(e^{-(1-\delta)e^{\lambda j}} + \frac{\sigma_{\mathcal{I}}^2}{\delta^2 n} \right) \\
&\quad \le \sum_{m=0}^{\nu+i} \binom{\nu+i}{m} \left(\prod_{j=-\nu}^{i-m-1} e^{-(1-\delta)e^{\lambda j}} + \left(\frac{\sigma_{\mathcal{I}}^2}{\delta^2 n} \right)^m \right) \\
&\quad \le \sum_{m=0}^{\nu+i} \left(\frac{(\nu+i)\sigma_{\mathcal{I}}^2}{\delta^2 n} \right)^m \prod_{j=-\nu}^{i-m-1} e^{-(1-\delta)e^{\lambda j}} \\
&\quad \le \sum_{m=0}^{2\nu} \left(\frac{2\nu\sigma_{\mathcal{I}}^2}{\delta^2 n} \right)^m \prod_{j=-\nu}^{i-m-1} e^{-(1-\delta)e^{\lambda j}}
\end{aligned}$$

To conclude the claim, we use the close formula for geometric sums and the fact that $e^{\lambda\nu} = n^{-1}$.

$$\begin{aligned}
\prod_{j=-\nu}^{i-m-1} e^{-(1-\delta)e^{\lambda j}} &= \exp\left(-(1-\delta) \sum_{j=-\nu}^{i-m-1} e^{\lambda j} \right) \\
&= \exp\left(-(1-\delta) \frac{e^{\lambda(i-m)} - e^{\lambda\nu}}{e^{\lambda} - 1} \right).
\end{aligned}$$

□

Proof of Lemma 16. Let $\nu = \frac{1}{\lambda}\log n$. We can write

$$\begin{aligned}
&E[S] = E[S|T \ge 2\nu] \Pr(T \ge 2\nu) + \\
&\sum_{i=-\nu}^{\nu} E[S|T \in (\nu - i - 1, \nu - i)] \Pr(T \in (\nu - i - 1, \nu - i)).
\end{aligned}$$

The probability that a y-early process with $y \ge 2\nu$ updates the cache is at most $f_{\mathrm{Exp}(\lambda)}(2\nu) = e^{-2\lambda\nu}$. Therefore, conditioning on an early expiration gap T greater than 2ν, we have that $E[S|T \ge 2\nu] \le ne^{-2\lambda\nu} = 1/n$. Hence, the first term is $O(n^{-1})$.

We now consider the summation in the above expression for $E[S]$. By Claim 21 and the fact that $E[S|T \in (\nu - i - 1, \nu - i)] \le ne^{-\lambda(\nu-i)} = e^{\lambda i}$, we have

$$\begin{aligned}
&\sum_{i=-\nu}^{\nu} E[S|T \in (\nu - i - 1, \nu - i)] \Pr(T \in (\nu - i - 1, \nu - i)) \\
&\le \sum_{i=-\nu}^{\nu} e^{\lambda i} \sum_{m=0}^{2\nu} \left(\frac{2\nu\sigma_{\mathcal{I}}^2}{\delta^2 n} \right)^m e^{-(1-\delta)\frac{e^{\lambda(i-m)} - \frac{1}{n}}{e^{\lambda}-1}} \\
&= e^{\frac{1}{n}} \sum_{m=0}^{2\nu} \left(\frac{2\nu\sigma_{\mathcal{I}}^2}{\delta^2 n} \right)^m e^{\lambda m} \sum_{i=-\nu}^{\nu} e^{\lambda(i-m)} e^{-(1-\delta)\frac{e^{\lambda(i-m)}}{e^{\lambda}-1}} \\
&\le e^{\frac{1}{n}} \sum_{m=0}^{2\nu} \left(\frac{2\nu\sigma_{\mathcal{I}}^2}{\delta^2 n} \right)^m e^{\lambda m} \sum_{i=-\infty}^{\infty} e^{\lambda i} e^{-(1-\delta)\frac{e^{\lambda i}}{e^{\lambda}-1}} \\
&\le \left(1 + O\left(\frac{\nu\sigma_{\mathcal{I}}^2}{\delta^2 n} \right) \right) \cdot \sum_{i=-\infty}^{\infty} e^{\lambda i} e^{-(1-\delta)\frac{e^{\lambda i}}{e^{\lambda}-1}}.
\end{aligned}$$

We now conclude the proof by showing a bound on the series above. Let $f(x) = e^{\lambda x} e^{-ce^{\lambda x}}$ and $x^* = \frac{1}{\lambda}\log\frac{1}{c}$, where $c = \frac{1-\delta}{e^{\lambda}-1}$. We have that $f(x)$ is increasing for $x < x^*$, and decreasing for $x > x^*$. Therefore, using the fact that $F(x) = \int f(x)dx = -\frac{1}{c\lambda}e^{-ce^{\lambda x}}$, we can approximate the series as follows:

$$\begin{aligned}
\sum_{i=-\infty}^{\infty} f(i) &= \sum_{i=-\infty}^{x^*-1} f(i) + f(x^*) + \sum_{i=x^*+1}^{\infty} f(i) \\
&\le \int_{-\infty}^{x^*} f(x)dx + f(x^*) + \int_{x^*}^{\infty} f(x)dx \\
&\le F(x^*) - F(-\infty) + f(x^*) + F(\infty) - F(x^*) \\
&= \frac{1}{c\lambda} + \frac{1}{ce} \\
&= \frac{e^{\lambda} - 1}{1 - \delta} \left(\frac{1}{\lambda} + \frac{1}{e} \right).
\end{aligned}$$

□

Proof of Lemma 17. Suppose the contrary, that is, suppose that there exists an i for which $p_i > e^{-\left(1-e^{-1/3}\right)\frac{i}{\epsilon}}$. Fix $n = \left\lceil e^{\left(1-e^{-1/3}\right)\frac{i}{\epsilon}} \right\rceil$, and observe that

$$\log(n-1) < \left(1 - e^{-1/3}\right)\frac{i}{\epsilon},$$

which implies $i > \frac{1}{1-e^{-1/3}} \cdot \epsilon \cdot \log(n-1)$.

By a standard Chernoff-like bound (for Poissonian variables), we have that the probability that less than $\frac{n}{2}$ processes end up in the interval $[i, i+1]$ is at most:

$$2e^{-n}\frac{n^{n/2}}{(n/2)!} = \Theta\left((2/e)^{n/2} \cdot n^{-1/2} \right).$$

The exponential distribution of the Poisson point process is such that, if we condition on k process showing up in that interval, the distribution of each of those k processes is uniform at random in the interval.

Therefore, under the conditioning that at least $\frac{n}{2}$ processes end up in the interval, we have that the probability that no y-early process with $i \le y \le i+1$ performs an early expiration is at most:

$$(1-p_i)^{n/2} \le \left(1 - e^{-\left(1-e^{-1/3}\right)\frac{i}{\epsilon}} \right)^{n/2} \le e^{-1/2},$$

since $(1-x)^{\lceil 1/x \rceil} \le e^{-1}$, for each $x \in (0,1]$.

Therefore, the expected early expiration gap is at least

$$\left(1 - e^{-1/2} - \Theta\left((2/e)^{n/2} \cdot n^{-1/2} \right) \right) \cdot i > \epsilon \log(n-1),$$

a contradiction. □

Proof of Lemma 18. Assume $n \le \left\lfloor e^{-1+\left(1-e^{-1/3}\right)/\epsilon} \right\rfloor$. By the union bound, we have that the probability that the early expiration gap is more than 1 is at most:

$$\begin{aligned}
\Pr(T_{\mathcal{D}} > 1) &\le n \sum_{i=1}^{\infty} p_i \le n \sum_{i=1}^{\infty} e^{-\left(1-e^{-1/3}\right)\frac{i}{\epsilon}} \\
&= n \cdot \frac{e^{-\left(1-e^{-1/3}\right)/\epsilon}}{1 - e^{-\left(1-e^{-1/3}\right)/\epsilon}} \\
&\le n \cdot \frac{e^{-\left(1-e^{-1/3}\right)/\epsilon}}{1 - e^{-1}} \le \frac{1}{e-1}
\end{aligned}$$

Suppose that $p_0 > e^{-\left(1-e^{-1/3}\right)/(2\epsilon)}$, and fix

$$n = \left\lfloor e^{-1+\left(1-e^{-1/3}\right)/\epsilon} \right\rfloor.$$

Again, we have that the probability that the probability that less than $n/2$ processes end up in an interval of length 1 is at most $\Theta\left((2/e)^{n/2} \cdot n^{-1/2}\right)$. The expected stampede can then be lower bounded by:

$$\begin{aligned}\Pr(T_{\mathcal{D}} \le 1)p_0\frac{n}{3} &\ge \frac{e-2}{3e-3}e^{-\left(1-e^{-1/3}\right)/(2\epsilon)}\left\lfloor e^{-1+\left(1-e^{-1/3}\right)/\epsilon}\right\rfloor \\ &\ge \frac{e-2}{3e-3}e^{-\left(1-e^{-1/3}\right)/(2\epsilon)}e^{-1+\left(1-e^{-1/3}\right)/\epsilon} - 1 \\ &= \frac{e-2}{3e^2-3e}e^{\left(1-e^{-1/3}\right)/(2\epsilon)} - 1 \\ &\ge e^{\Omega(1/\epsilon)}.\end{aligned}$$

Suppose, instead, that $p_0 < e^{-\left(1-e^{-1/3}\right)/(2\epsilon)}$. Fix $n = \left\lfloor e^{-1+\left(1-e^{-1/3}\right)/(2\epsilon)}/3\right\rfloor$. The probability that more than $2n$ processes end up in an interval of length 1 can be shown to be (using Poissonian tail bounds) at most

$$2e^{-n}\frac{n^{2n}}{(2n)!} = \Theta\left((e/4)^n \cdot n^{-1/2}\right).$$

We then have that:

$$\Pr(T_{\mathcal{D}} > 0) \le 3n \cdot p_0 + 3n\sum_{i=1}^{\infty} p_i \le \frac{1}{e} + \frac{1}{e-1} < \frac{19}{20}.$$

If $T_{\mathcal{D}} = 0$, that is no process performs an early expiration, then we run into a regular expiration with a stampede of size n. Therefore the expected stampede is at least

$$\frac{1}{20}\cdot n \ge e^{\Omega(1/\epsilon)}.$$

The proof is concluded. □

Proof of Lemma 19. Recall that if the process arrival distribution is $\mathcal{I}$, then the distribution of the interval between two consecutive processes is $\mathcal{I}/n$. Without loss of generality, say that the expected value of $\mathcal{I}$ is 1 and that $\sigma_{\mathcal{I}}$ is its standard deviation. Let $p = \frac{\log \hat{n}}{\hat{n}}$. Consider a second setting where the rate is $p \cdot n$, with the exponential gap distribution $\mathrm{Exp}(\lambda)$, and with a new process arrival distribution $\mathcal{I}'$. To sample from $\mathcal{I}'$, we procede as follows: first, we flip a coin with head probability p until we get heads; let k be the number of coin flips (observe that k is distributed like the geometric distribution $\mathrm{Geom}(p)$ with parameter p); then, sample k i.i.d. variables $X_1, \ldots, X_k \sim \mathrm{Exp}(\lambda)$ — the sample of $\mathcal{I}'$ will be equal to $p \cdot (X_1 + X_2 + \cdots X_k)$.

A simple coupling shows that the original process (having rate n, inter-arrival distribution $\mathcal{I}$, and gap distribution $\mathcal{D}$) is equivalent to the new process (having rate $p \cdot n$, inter-arrival distribution $\mathcal{I}'$, and gap distribution $\mathrm{Exp}(\lambda)$). Therefore, after having computed the expectation and the variance of $\mathcal{I}'$, we can apply Theorem 7 to get our claim. We have

$$\begin{aligned}\mu_{\mathcal{I}'} &= \mathrm{E}[\mathcal{I}'] = p \cdot \mathrm{E}[\mathrm{Geom}(p)] \cdot \mathrm{E}[\mathrm{Exp}(\lambda)] = \frac{1}{\lambda}, \\ \mathrm{Var}[\mathcal{I}'] &= p^2 \cdot \big(\mathrm{E}[\mathrm{Geom}(p)] \cdot \mathrm{Var}[\mathrm{Exp}(\lambda)] \\ &\qquad + \mathrm{E}[\mathrm{Exp}(\lambda)]^2 \cdot \mathrm{Var}[\mathrm{Geom}(p)]\big) \\ &= \frac{p}{\lambda^2} + \frac{1-p}{\lambda^2} = \lambda^{-2}.\end{aligned}$$

Therefore, $\sigma_{\mathcal{I}'} = \lambda^{-1}$. We can then apply Lemma 15 to get the early expiration gap is at most $\frac{\log(pn)}{\lambda}$ with high probability. Observe that

$$\frac{1}{\alpha}\log(n/\alpha) \le pn \le \alpha\log(\alpha n).$$

Thus, the early expiration gap is at most $\frac{\log\log n + \log\alpha}{\lambda}$ with high probability. By $\alpha < e^{o(\log\log n)}$, we get that the expiration gap is at most $(1 + o(1)) \cdot \frac{\log\log n}{\lambda}$.

We can then use Lemma 16, to conclude that the expected stampede is at most:

$$\begin{aligned}&\left(1 + O\left(\frac{\sigma_{\mathcal{I}'}^2\log(pn)}{\delta^2 pn}\right)\right)\cdot\frac{e^\lambda - 1}{1-\delta}\left(\frac{1}{\lambda} + \frac{1}{e}\right) \\ &= \left(1 + O\left(\alpha\frac{\lambda^{-2}\log(\alpha\log(\alpha n))}{\delta^2\log(n/\alpha)}\right)\right)\cdot\frac{e^\lambda - 1}{1-\delta}\left(\frac{1}{\lambda} + \frac{1}{e}\right).\end{aligned}$$

By $\alpha < e^{o(\log\log n)}$, if we let δ shrink to 0, we get that the expected stampede is at most

$$(1 + o(1))\cdot\frac{e^\lambda - 1}{1-\delta}\left(\frac{1}{\lambda} + \frac{1}{e}\right).$$

□

Proof of Lemma 20. We assume that the rate is known. Let $n_k = 1$, and, for $i \ge 1$, $n_{i-1} = e^{n_i}$.

Fix some $k \ge 1$, and let the rate n be equal to $n = n_0$. Observe that $k = \log^\star n$.

Now, given a time $-t$, fix:

$$p_t = \begin{cases} \frac{n_{\lfloor t/2\rfloor}}{n} & \text{if } \lfloor t\rfloor < 2k+1, \lfloor t\rfloor \text{ even} \\ 0 & \text{otherwise}\end{cases}$$

That is, for a time in $(-\infty, -2k-1]$, the probability is 0; for a time in $(-2k-1, -2k]$, the probability is n^{-1}; for a time in $(-2k, -2k+1]$, the probability is 0; for a time in $(-2k+1, -2k+2]$, it is en^{-1}; for a time in $(-2k+2, -2k+3]$, it is 0; for a time in $(-2k+3, -2k+4]$, it is $e^e n^{-1}$, …, for a time in $(-2, 1]$ it is 1, and for a time in $(-1, 0]$ it is 0.

If the process passes time $-2i-1$, $i = 0, \ldots, k$, without having ever refreshed then, in expectation, it will make n_i refreshes before reaching time $-2i+1$. The probability P_i of making no refresh in the interval $(-2i-1, -2i+1]$, if the process has not refreshed before time $-2i-1$, is equal to:

$$P_i = (1-p_i)^n = \left(1 - \frac{n_i}{n}\right)^n \to e^{-n_i} = n_{i-1}^{-1}.$$

Observe that $P_{i+1}\cdot p_i \cdot n = P_{i+1}\cdot n_i = 1$. The expected number of refreshes in the interval $(-i-1, -i]$ is then equal to:

$$P_k \cdot P_{k-1}\cdots P_{i+1}\cdot p_i \cdot n = \prod_{j=i+2}^{k} P_j.$$

The algorithm will refresh with probability 1 in the interval $(-2, -1]$. Therefore the expected stampede is upper bounded by:

$$2 + \sum_{i=1}^{k-2}\prod_{j=i+2}^{k} P_j < 3.$$

The early expiration gap, on the other hand, is at most equal to $2k+1$, with $k = \log^\star n$, since the probability of a refresh is 0 if $t \le -2k-1$. □

A. REFERENCES

[1] J. Allspaw and J. Robbins. *Web Operations: Keeping the Data On Time*. O'Reilly Series. O'Reilly Media, Incorporated, 2010.

[2] J. Anton, L. Jacobs, X. Liu, J. Parker, Z. Zeng, and T. Zhong. Web caching for database applications with oracle web cache. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 594–599. ACM, 2002.

[3] X. Bai and F. P. Junqueira. Online result cache invalidation for real-time web search. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 641–650, New York, NY, USA, 2012. ACM.

[4] J. Challenger, A. Iyengar, and P. Dantzig. A scalable system for consistently caching dynamic web data. In *INFOCOM '99*, volume 1, pages 294–303 vol.1, Mar 1999.

[5] Drupal Community. `https://www.drupal.org/`. Accessed December 30, 2014.

[6] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 2012.

[7] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 91–101, 2002.

[8] Memcached. `http://memcached.org/`. Accessed December 30, 2014.

[9] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[10] J. Sheltren, N. Newton, and N. Catchpole. *High Performance Drupal: Fast and Scalable Designs*. O'Reilly Media, 2013.

[11] J. Swartz. Chi-0.58. `http://search.cpan.org/~haarg/CHI-0.58/lib/CHI.pm`. Accessed December 30, 2014.

[12] J. Swartz. Problems and solutions for typical perl cache usage. `http://www.openswartz.com/2008/02/`. Accessed December 30, 2014.

[13] H. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 2010.

[14] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable high speed ip routing lookups. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '97, pages 25–36, New York, NY, USA, 1997. ACM.

[15] P. Zaitsev. Percona mysql performance blog. `http://www.percona.com/blog/2010/09/10/cache-miss-storm/`. Accessed December 30, 2014.