Rate-Limiting at Scale

SANS AppSec Las Vegas 2012

Nick Galbreath @ngalbreath nickg@etsy.com

Who is Etsy?

- “Marketplace for Small Creative Businesses”
- Alexa says #51 for USA traffic
- > $500MM transaction volume last year

What's a Rate Limit?

Maximum number of events per (brief) period per user after which the resource is denied.

e.g. "no more than 2 logins per minute"

Why?

Whoops

- Without rate limits on credit card authorizations your site becomes a card skimmer site.
- Using a website is much easier than going to the gas station pump or other anonymous card reader

Whoops

- Rate limits needed for anything that gets reviewed by humans such as customer service requests.
- CRMs are typically bad at dealing with spammy stuff

Whoops

- Robots / Crawlers Gone Wild (not always an intended DDoS)
  - 20,000 items in shopping cart
    - spam attack!
- Can crush sites very quickly, at almost no cost. Especially when crawl generates load or writes to the database

Do Rate Limits Stop all Fraud? No, but...

- Eliminates false positives and punks
- Allows you to focus on more sophisticated attacks
- Protects against damaging bursts of activity (malicious or not)

Rate Limits are needed on anything that depends on an external resource

This is almost everything!

Implementation

Continuous Rate Limits

- Store user identifier, event-type, timestamp
- Allows easy rate-limits for multiple ranges
- Allows easy cross-event limits
- Easy to implement in SQL

Continuous RL Schema

ngalbreath — emacs — 42×10

```
CREATE TABLE events (
   userid BIGINT NOT NULL,
   eventid    INT NOT NULL,
   created TIMESTAMP NOT NULL,
# OR created BIGINT NOT NULL,
#    for microseconds
   PRIMARY KEY (userid,eventid,createdL)
);
-uuu:---F1  schema.sql      All (3,15)
```

Check if your database timestamps store microseconds or not. You want ‘em.

Continuous RL Queries

ngalbreath — emacs — 42×10

```
SELECT COUNT(*) FROM events
 WHERE userid=? AND eventid=?
   AND created >= NOW() - ?period?;

INSERT INTO events VALUES(?,?,NOW());

DELETE FROM events
  WHERE created > NOW() - ?maxperiod?
-uuu:---F1  queries.sql      All (5,0)
```

At Scale Pain

- At scale, this is really painful for databases to handle.
  - Constant index churn
  - Use in-memory database (or run off ramdisk) if trying this out

Quantized Rate Limits

- Stores a count in a time-window or bucket.

- Map current time to a bucket
- (int) (NOW()/period) e.g. NOW()/3600 is gives the hour bucket.

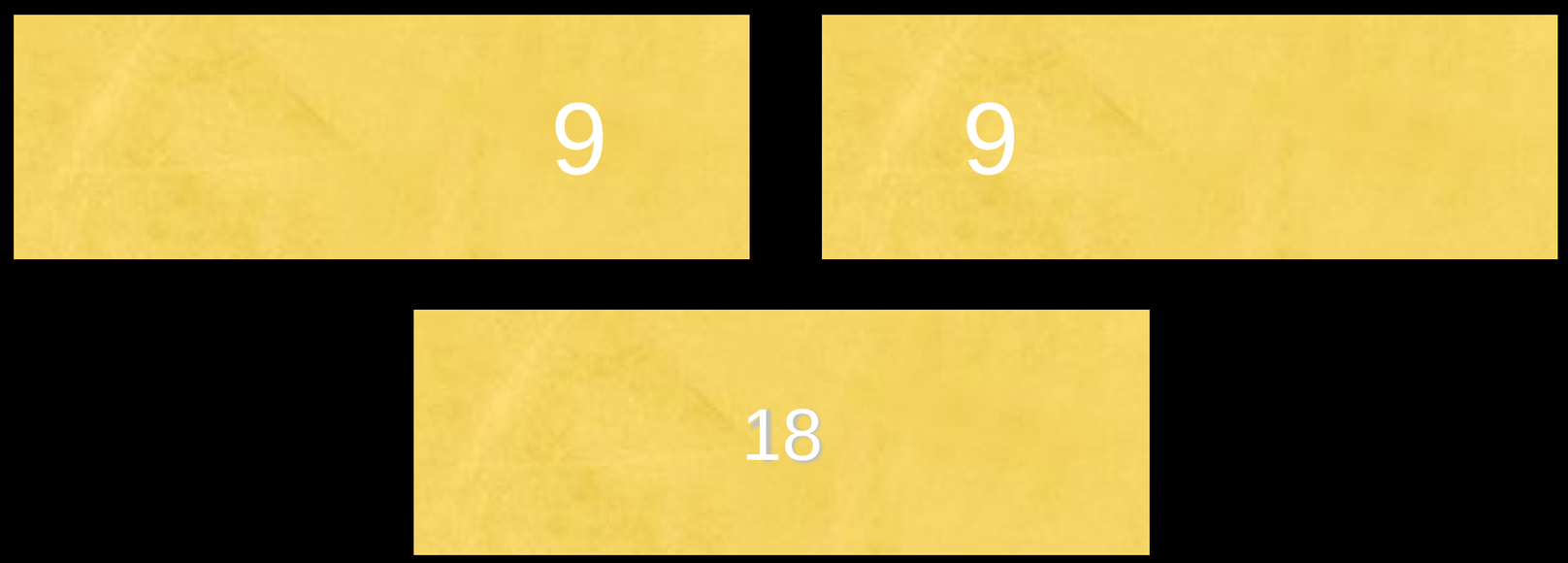
```
mysql> select floor(NOW() / 3600) as bucket;
+------------+
| bucket     |
+------------+
| 5589007547 |
+------------+
```

Direct Lookup

- Everything is primary key lookup.
  userid-event-period-bucketid
  60min*:* “nickg-login-3600-5589007547”
  10min: “nickg-login-600-33534045284”
- Multiple time-frames require multiple buckets, which means multiple inserting/checking.

Quantized RL Accuracy

- Not exact. If you set *N per Period*, quantized rate-limits may go as high as:
  (n-1)x2 per Period.
- *e.g. 10 per minute --> 18 per minute*



Rate-Limits at Scale

- We traded exact accuracy and flexibility for scaling.
- Implementation using Memcache or Redis (and perhaps SQL)
  set nickg-login-60-212331231 += 1
- Well known sharding techniques
- Auto-expiration of old buckets
- Each set/get takes 1/10 or less of millisecond. Almost invisible.

Usage

Please write unit tests!

- Easy to get wrong, and consequences can be unpleasant
- Edge cases and race conditions
  - memcache doesn’t have a “insert or increment” operation. Need to do multiple steps and check error conditions.

Rollout

- Once in production start with guestimates on rate limits
- If rate limit is triggered, take no action and only log/graph
- Does volume match expectations?
- Wash, Rinse, Repeat until tuned appropriately

So a user hit a rate limit. Now what?

- Do you let them know? (visible indicator)
- Do you start CAPTCHA-ing?
- Do you black hole it? (silent)

Also keep logging and graphing. You'll need these to debug when things go awry

Other Topics

Anonymous Identifiers

- hash of (IP + appropriate HTTP headers)
- order of headers matters
  different browsers order them differently
- Spoofed user agents don’t always get the order right
- Can use first 8 bytes of hash and convert to 64-bit integer (and make negative)

Laddering

- Use Laddering to do rate limits at different time scales for the same event.
  - Set a short period and high rate, to prevent bursts
  - Set a longer period with lower rate, to prevent slow crawls robots.

Ladder *longer* periods to have a *smaller* rate

Negative example:

2 per Minute (~0.033 events per sec)

or 2x60 = 120 per Hour

so laddering with

300 per Hour (~ 0.083 events per sec)
does nothing, but

100 per Hour (~ 0.028) is good.

And finally

- Almost every action on Etsy has ladder rate-limit
- We learn the hard way what is *not* limited
- Virtually no performance impact at scale.
- Should we open source the driver?

Etsy

Nick Galbreath nickg@etsy.com
@ngalbreath
SANS AppSec Las Vegas 2012