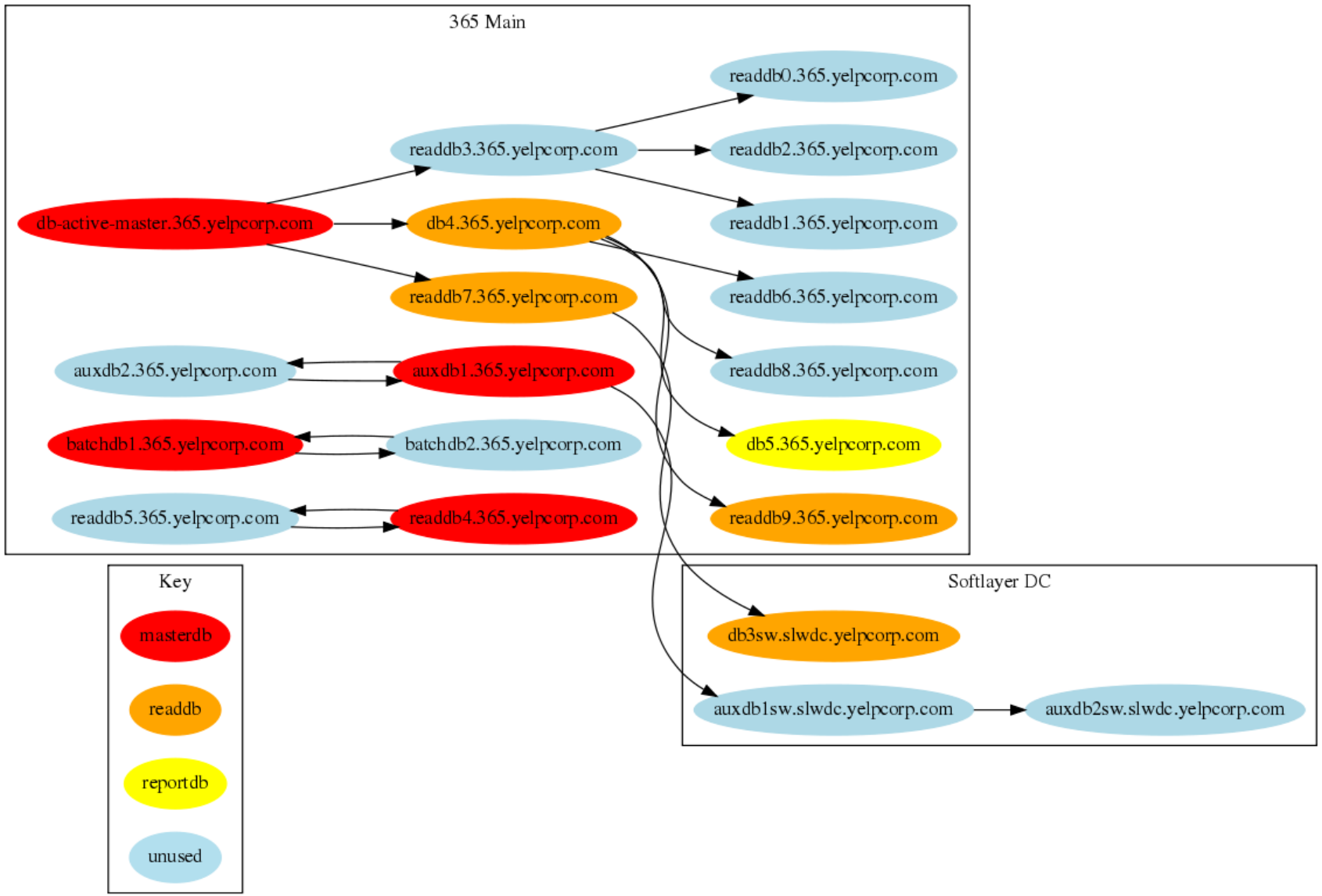
Life, Databases, and You

Adam Derewecki
given for EECS 340 @ CWRU

$ whoami

- Graduated BSE Comp Eng 2007
  - Research Assistant for Networking Dept
  - TA for ENGR131 (Fall '05, '06)
- IBM Software Engineer, 2008 - 2009
  - DB2 XML Storage and Runtime teams
- Yelp Backend Engineer, 2009 - Present

Yelp's Database Setup



- MySQL
- memcached for expensive, often made queries

Replicas and Code

- 1 master
  - all writes from production and batch machines
  - only does reads in a "dirty session"
- 2 direct replicas
  - 1 'master' replica
  - 1 backup replica
- "time warp" replica
- read, reporting databases
  - master replica's replicas used for read operations
  - read databases used for servlets (high response)
  - reporting databases used for batch processes (less strict responsiveness guarantees

Sharding across databases

- Everything "important" lives in main db
- Auxiliary db
  - metrics rollups
  - can be hit from webs
    - know that it will be slower
- Batch db
  - only used by batch machines
  - batch emailers

- Why shard?
  - Stuff that needs to be fast can stay fast
  - Non-essential failures isolated from main site
    - Batches less predictable than servlets

When you see your DBA running...

Get out of his/her way.

Why don't we use sexy key/value db

- We do!
  - Tokyo Tyrant for flowstate
  - memcached for query caching
- Problems with less mature technology for main data store
- Failure rates for Database systems
  - databases are some of the most mature software projects around today
  - DB2 code base ~35 years of maturity
- People writing these don't always have the actual scale problems they think they have

MySQL as a key/value store

```
CREATE TABLE entities (
    added_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    id BINARY(16) NOT NULL,
    updated TIMESTAMP NOT NULL,
    body MEDIUMBLOB,
    UNIQUE KEY (id),
    KEY (updated)
) ENGINE=InnoDB;

{   "id": "71f0c4d2291844cca2df6f486e96e37c",
    "user_id": "f48b0440ca0c4f66991c4d5f6a078eaf",
    "title": "kv stores are great lol",
    "link": "http://mysql.com",
    "published": 1235697046,
    "updated": 1235697046, }

CREATE TABLE index_user_id (
    user_id BINARY(16) NOT NULL,
    entity_id BINARY(16) NOT NULL UNIQUE,
    PRIMARY KEY (user_id, entity_id)
) ENGINE=InnoDB;
```

- http://bret.appspot.com/entry/how-friendfeed-uses-mysql

"this is not a jab at foursquare"

- MongoDB is a key/value store database
- most data foursquare stores is check-in history
- mongod servers sharded by user id
- on October 5th, experienced 11+ hours of downtime
  - shard got overloaded (load balance fail)
    - brought up another shard
  - new shard didn't fix load balance problem
    - ended up doing a 6 hour re-index
- my 2 cents
  - "For reasons that are not entirely clear to us right now, though, the addition of this shard caused the entire site to go down."
  - making drastic changes to the production database topology in times of crisis is never good
- http://blog.foursquare.com/2010/10/05/so-that-was-a-bummer/


I'm in ur
CompSci
eyes
pwning ur

SQL in Yelp's Code: Raw SQL

```
query_kwargs = {
    'flags_add': flags_add,
    'flags_remove': flags_remove,
    'id': id
}
cursor.execute(
    'UPDATE user_location
    SET flags = (flags & ~%(flags_remove)s) | %(flags_add)s
    WHERE id = %(id)s' % query_kwargs
)
```

- Why you shouldn't do this
  - Prone to typos
  - No centralized escaping
  - Python knows nothing about this query

SQL in Yelp's Code: SQLAlchemy

- SQLAlchemy Project
  - We've interviewed candidates who have used it

```
cols = (
    dbobjs.BusinessWeekly.c.title,
    dbobjs.BatchEmail.c.time_to_send
)
from_obj = dbobjs.BusinessWeekly.table.join(dbobjs.BatchEmail, onclause=(dbobjs.
BusinessWeekly.c.batch_email_id == dbobjs.BatchEmail.c.id))

weekly = EZA.select(cursor, dbobjs.BusinessWeekly.c.id == 1826, from_obj=from_obj, cols=cols)
```

- Pros
  - more Pythonic
  - valid SQL syntax generation guarantee
- Cons
  - Boilerplate

SQL in Yelp's Code: ORM

- Object-relational Mapping / Active Record Pattern
  - Martin Fowler, 2003
- More natural way of storing objects in the database
- Pythonic representation of MySQL data
  - create multiple objects in Python w/o touching db
  - flush in one batch
  - keeps a cache of data you have looked up
  - lazy loading of joins
    - only perform join if you access data that uses it
- Based on SQLAlchemy's Elixir
  - http://elixir.ematia.de/trac/wiki
  - similar to Django's ORM

SQL in Yelp's Code: ORM

```
class User(StandardModel):
    __tablename__ = "user"

    id = build_id_column('id', primary_key=True)
    first_name = Column(String(32))
    last_name = Column(String(32))
    flags = bitfield_property(
        CONFIRMED                   = 0x00000001,
        INACTIVE                    = 0x00000002,
    )
    time_created = build_time_column(default_now=True)
    primary_email_id = build_id_column('primary_email_id',
        foreign_key=ForeignKey("%s.id" % table_names.user_email))
    primary_email = build_related_column('primary_email', 'UserEmail',
        primaryjoin="User._primary_email_id==UserEmail._id",
        backref="user", lazy=False

user = session.query(User).filter_by(user=42).one()
print user.first_name

users = []
for first, last in [('Adam', 'Derewecki'), ('Burt', 'Reynolds')]:
users.append(User(first_name=first, last_name=last))
session.add_all(users)
session.flush() # writes to database
```

Making MySQL perform better

- Slow queries are the obvious problems
  - make sure you have appropriate indexes
  - try not to fetch more than you need
  - avoid subselects in MySQL
  - MySQL comes with a slow query log that records queries that take longer than a second
- Too many "non-slow" queries
  - ex: 2000 weeklies, separate query to load each
  - memcache comes in handy for frequent queries
  - RTT may seem negligible but isn't always

Monitoring MySQL

- Log queries from production
  - Full query + time spent
- Daily deadlock report sent to developers
- Watchdog report
  - all queries that take longer than 10s

Deadlock report entry

```
  server: db-active-master.365
      ts: 2010-10-05 11:37:04
  thread: 4458755
  txn_id: 16412074876
 txn_time: 0
    user: yelplive
 hostname: batch4b.365.yelpcorp
      ip: 10.18.4.14
      db: yelp
     tbl: user_badge
     idx: user_id
lock_type: RECORD
lock_mode: X
wait_hold: w
  victim: 0
   query: INSERT INTO user_badge (time_created, user_id, badge_key, badge_sub_key,
full_badge_key, flags) VALUES (UNIX_TIMESTAMP(), 4504334, 'tutorial', 'Rookie',
'tutorial::Rookie', 524290)
```

Speeding up Email

```
while True:
    email_segment = get_emails_to_send()
    for email in email_segment:
        msg_id = smtplib.send_email(email)
        database.update_flags_by_id(
            email.id, flags|=SENT
        )
        database.update_columns_by_id(
            email.id,
            send_time=time.time(),
            msg_id=msg_id
        )
```

Speeding up Email

```
while True:
    email_segment = get_emails_to_send()
    for email in email_segment:
        msg_id = smtplib.send_email(email)
        database.update_flags_by_id(
            email.id, flags|=SENT
        )
        database.update_columns_by_id(
            email.id,
            send_time=time.time(),
            msg_id=msg_id
        )
```

- Two queries for every email sent

Speeding up Email

```
while True:
    email_segment = get_emails_to_send()
    for email in email_segment:
        msg_id = smtplib.send_email(email)
        database.update_by_id(
            email.id,
            send_time=time.time(),
            msg_id=msg_id,
            flags=database.Email.set_bits(
                SENT=True
            )
        )
```

- Changed to one call -- half as much time spent in RTT
- Sped up sending by ~40%

Speeding up Email

- Can we do better? Hells yes.
- One query better than two, but # of queries still directly proportional to # of emails sent
- Goal: # of queries constant per loop iteration

```
CREATE TABLE `email_update` ; -- duplicate of `email` table
INSERT INTO `email_update` VALUES (email1, email2...) ;
UPDATE `email` e, `email_update` eu SET e.sent_time=eu.sent_time,
   e.flags=eu.flags, e.msgid=eu.msgid WHERE e.id=eu.id ;
DELETE FROM `email_update` ;
```

- 4 queries, regardless of segment size
- segment size only limited by max MySQL query length (1.5 MB or so by default)

Speeding up Email

- Why not use CREATE TEMPORARY TABLE?
  - temp tables exist only in memory
  - MySQL slaves can crash during routine workload in a replicated environment
    - you're boned.
  - http://www.xaprb.com/blog/2007/01/20/how-to-make-mysql-replication-reliable/
- MOAR SPEED
  - Still 4 round trips to the database for an atomic modification
  - Concatenate queries into 1 and send over the wire
    - now that's fast
- Added bonus
  - Instead of 12 mailers hitting the database many times/second, they hit with a big payload every ~12 seconds
  - Less time spent dealing with contention*

Speeding up Email: Take-aways

- Query time itself is important but obvious
- Roundtrip time is the silent killer
- Beefing up your MySQL server's hardware goes a long way
  - it turns out that what was taking a significant amount of time on our dev server wasn't a problem in production
  - Production MySQL server has 96 GB of RAM
    - simple queries have total time of *micro*seconds to complete

Anecdote: The Marquee

A moment ago

Elite '10
37
82

Jessica K.
Cincinnati, OH

Ludlow Bromley Yacht Club

Ludlow, KY

This place is honestly indescribable. I could tell you that they serve beer, make good food, and have docks for boaters to tie off and join the fun. I could tell you how they don't accept credit cards but have a conveniently-located ATM for everyone to use. I could tell you how every time I've been here, it's better than the last time I went.

Bottom line: go here, especially in the summer when you can appreciate the river, the cool refreshing breeze, and the great carefree attitude that can only come from a yacht club in warm weather.

10/6/2010 10:10 am

Elite '10
7
62

Kelly N.
Aurora, CO

Aurora Movie Tavern

Aurora, CO

I have gone back and forth on liking and hating The Movie Tavern. Sometimes I love it and sometimes I wish I had never gone. So here it is my pros and cons:

Pros:
I get to eat an actual meal and have a beer while I watch my movie. Sure the food is not outstanding but its good enough. The burgers are not bad and I think they just might have the best fried pickles I have ever had in my whole life! The prices are great. I can get a move ticket for under $10 and if I get there before 5pm its only $6 which ROCKS. Oh the movies are new movies too, this is not a discount old movie kinda place. The price of food is comparable to a Chilli's or Applebees. It also happens to be close to home which is nice

Cons: Pretty much the worse service you can ever receive. I have gone so many times I can't even count and maybe 2 times I have had OK service. One time I had to flag someone down THREE TIMES before I got a server to help me!! It really heats me up when I receive bad service especially since I am a good tipper! The other things is it is noisy throughout the movie. The servers do take orders and talk to customers through the movie so you have to just get over that if you are going to go. There is no getting around it.

I guess when all is said and done I will still go because I like my movies to not cost me an arm and a leg and the cold beer and friend pickles generally save the day when I get crap service.

I think that if Management made an effort to train the servers and organize the whole process a little better the whole experience would ROCK but for now it just falls a little short.

10/6/2010 10:10 am

Elite '10
131
285

Laurice M.
Los Angeles, CA

The Troubadour

West Hollywood, CA

Yeah. Still my fave venue EVER! Great sound, great views, great acts.

The next time I go here I wanna try the food. The chef was really sweet to ask people at the bar and/or lounge if they wanted anything before he closed up shop.

10/6/2010 10:10 am

Elite '10
66
109

Spa Velia

San Diego, CA

They put the Ahhh in Spaaa.....

The Final Slide

- Any of this sound interesting? We're hiring!
  - www.yelp.com/careers
  - 50 person engineering team
  - Open source
    - github.com/yelp
    - engineeringblog.yelp.com
  - Located in downtown San Francisco