Programming concepts you probably won't learn in the Case CS curriculum

Adam Derewecki
given October 6th for 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

- derwiki on Hacker News, github, Twitter
- tech blog @ derwiki.tumblr.com

Adam D.'s Reviews | San Fran

www.yelp.com/user_details?userid=S0yLkzB9QHC3Q4BpfM_iCw

Yelp | Yelp Mail | Google Docs | Calendar | delicious | tickets | IA Tickets | compete.com | derwiki.pb | Other Bookmarks

Now in the UK!

Friends' Activity 100 | Logged in as Adam D. | My Account | Log Out

yelp
Real people. Real reviews.®

Search for (e.g. taco, cheap dinner, Max's)

Near (Address, City, State or Zip)
cleveland, oh

Search

Welcome | About Me | Write a Review | Find Reviews | Invite Friends | Messaging | Talk | Events | Member Search

Profile Home | Lists | Reviews | Quick Tips | Compliments | Friends | Following | Bookmarks | Events

Adam "Ad-rock \m/" D.'s Profile [Edit nickname]

derwiki.yelp.com Edit »

"I write reviews waiting for Bullbot to finish."

'10 elite

195 Friends
144 Reviews
16 Firsts
19 Quick Tips
7 Fans
61 Local Photos
1 Event Submitted
7 Lists

Add More Photos

Rating Distribution

5 stars 49
4 stars 67
3 stars 23

You Have 1 Unfinished Review Show

Recent Reviews 144 Reviews

Search Reviews

Filter by: Location Category | Sort by: Date

Alamere Falls
Category: Active Life

Pt. Reyes National Seashore
Bolinas, CA 94924

9/29/2010 1 photo

The Alamere Falls trail starts at the Palomarin Trailhead and continues along the coast for a few miles of breathtaking coastal views, and then detours inland a little bit to go by Bass and Pelican Lakes. Bass Lake had a bunch of people chilling on rafts or just swimming around in general. There's even a rope swing! Past Bass Lake is Pelican Lake, which is so picture perfect I thought I was looking at a PGA golf game for the Sega Genesis (I'm sure more recent golf games have been made but have no first hand experience).

As you near the coast for the final time (maybe a mile out) there's a sign for the "caution unmaintained" Alamere Falls trail. It's not as bad as it sounds, but toward the end you're definitely climbing down some crumbly rocks -- I'd use caution if you're taking kids with you. After that, you're treated to three waterfalls of various sizes, with the final one being the famous waterfall the flows DIRECTLY INTO THE OCEAN. Ever see that before? I hadn't.

195 Friends

New Reviews!
770 | 800 | Elite '10
LiLy D.

1428 | 700 | Elite '10
Lizzie G.

New Reviews!
273 | 82 | Elite '10
Cara L.

New Reviews!
297 | 149 | Elite '10
Elodie F.

New Reviews!
598 | 452 | Elite '10
Kelly S.

New Reviews!
4055 | 925 | Elite '10
ruggy j.

More »

43 Compliments

Technologies used at Yelp

- Apache + mod_python
- Python 2.5.4
  - inhouse framework, similar to Django
- MySQL 5.0
- git 1.7
- JavaScript, Prototype
- Ubuntu servers
- iPhone, Android, Pre, and Blackberry apps
- Gearman distributed work queue
- AWS Elastic Map Reduce (EMR)

Neat stats about us

- 13,000,000 reviews
- 38,000,000 unique monthly visitors
- 237,000,000 page views/month
- 5,100,000,000 page views over the last 5 years
- ~100 servers in production environment
- 50 engineers
  - Consumer, biz, internal apps teams
  - Infrastructure
  - Search, Spam, Data Mining, Ad Delivery
  - Release engineering
  - Systems
  - Mobile

What Internal Applications does at Yelp

- Four fulltime, one previous-intern-turned-part-time
- Community Manager tools
  - ~40 CMs in cities around the world
  - write and maintain tools to help them do their jobs
    - Weekly Yelp newsletter workflow
    - Yelp Elite Squad email tool
    - newsletter metrics
- User Operations tools
  - "the Queue"
- Yelp Deals
  - Deals email editor
- Salesforce data refreshing
- Email infrastructure

How we `git`-r-done at Yelp

- What is git?
  - Distributed Version Control System
    - "similar" to CVS, SVN, Mercurial, etc
  - Developed by Linus Torvalds to manage the Linux source tree
- Why use git?
  - Fast.
  - Offline commits
  - Ability to pull/rebase commits from remote repo
  - Forks: git clone git://myserver/repo.git
  - Graph based history, not linear

Subversion: The Dark Years

- Pushmaster would let people know they can check in
- Race to svn up && svn merge
  - Need the latest checkout to commit
  - Had to wait on developers to commit before pushing to stage
- Commits that get removed (because of test failures) need to be 'reverse-committed' instead of fully removed
- SLOW.
- Wasn't scaling well as number of developers increased

git: "The New Way"

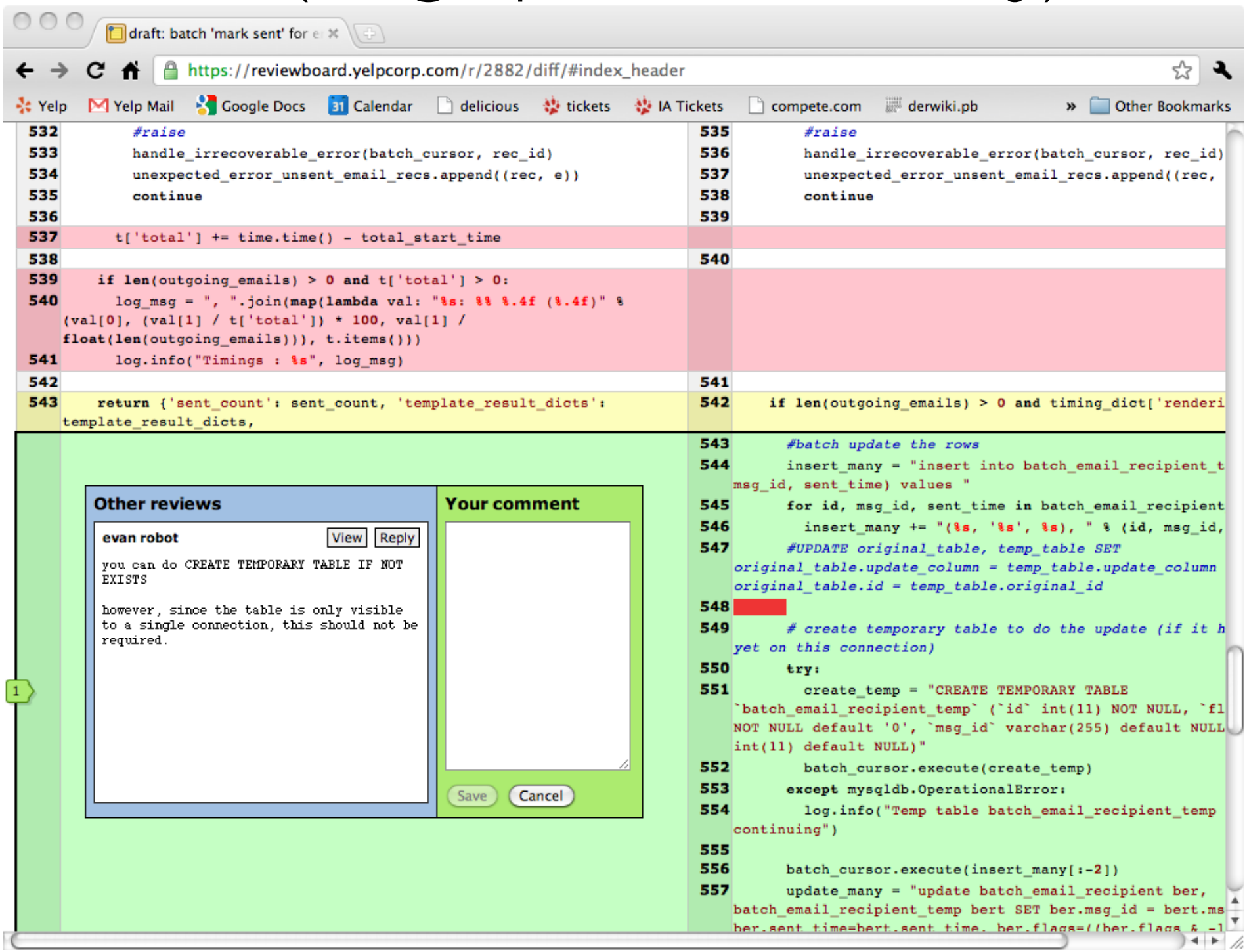
- When a changeset is ready to go out, 'release branch' is committed
  - local branch could be many commits, release is one
- Pushmaster creates new 'deploy' branch off stable-master
- Pushmaster pulls in all release branches
- Testing is done, bad commits are rebased out if necessary
- deploy is pushed into production
- once push is deemed stable, it is 'certified'
  - deploy branch is rebased onto master, tagged stable
  - master now has the same history as deploy
  - `git push deploy :origin` to cleanup the deploy branch

Pushmaster App: http://github.com/Yelp/PushmasterApp

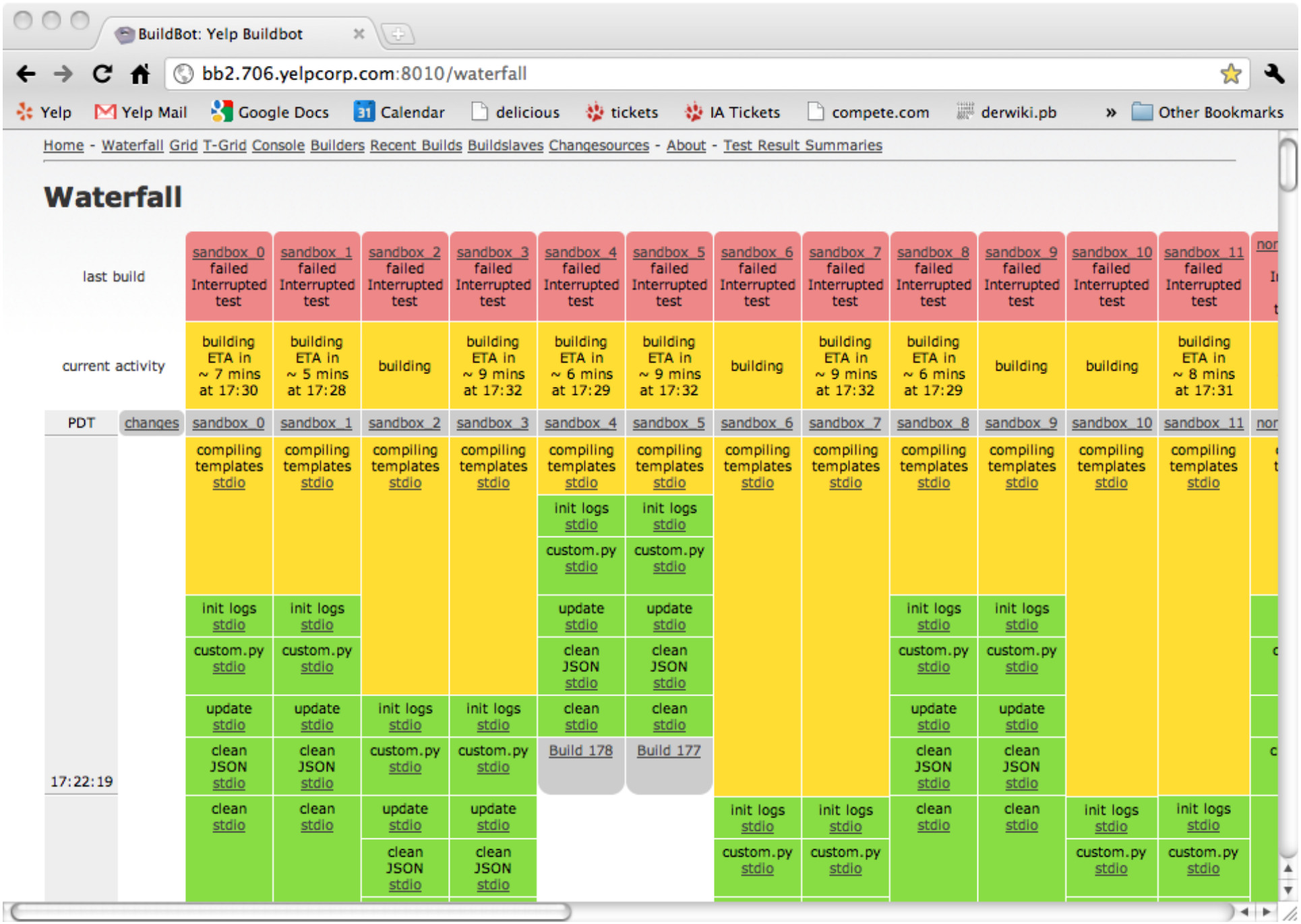
From idea to production

- Create branch
- Code!
- Add tests to verify functionality
- Put out for code review
- Iterate on code review comments until thumbs up all around
- Submit release branch to Pushmaster app
- Pushmaster pushes your changeset (and others) to staged environment + kicks off BuildBot
- Interactive testing on stage, automated BuildBot testing
- Make tweaks, pull bad changesets, repeat until looks good
- Pushmaster pushes deploy branch (with your changeset) into production
- Pushmaster certifies and rebased `deploy` onto `master`

ReviewBoard (free @ http://www.reviewboard.org/)



BuildBot (free @ http://buildbot.net/)



Distributing Work

- Long running jobs
  - MapReduce batches
  - Reports that take longer than servlet-rendering time
- Daemon processes
  - batch email senders
  - scraper detection daemon
- Some jobs take on the order of seconds, others can take almost an entire day
- git repository for fcron files
  - script to deploy them to all of our batch machines
- Problems
  - Batch machine dies, you're boned

Distributing Work: Gearman

- distributed workers, centralized queue
- created by LiveJournal, currently implemented in C
- language independent (take that Erlang!)
- (at least) one gearmand server
  - accepts incoming jobs to queue
  - maintains knowledge of all workers registered to a task
  - assigns jobs to workers as they become available for work
- job producers (clients)
  - create job on a 'channel' with args necessary to do the work -- i.e. 'email_renderer', {'start_id':300, 'end_id':399}
  - come from web machines, batch machines, other workers
- many workers can listen to the same channel
  - from different machines!

Distributing Work: Gearman

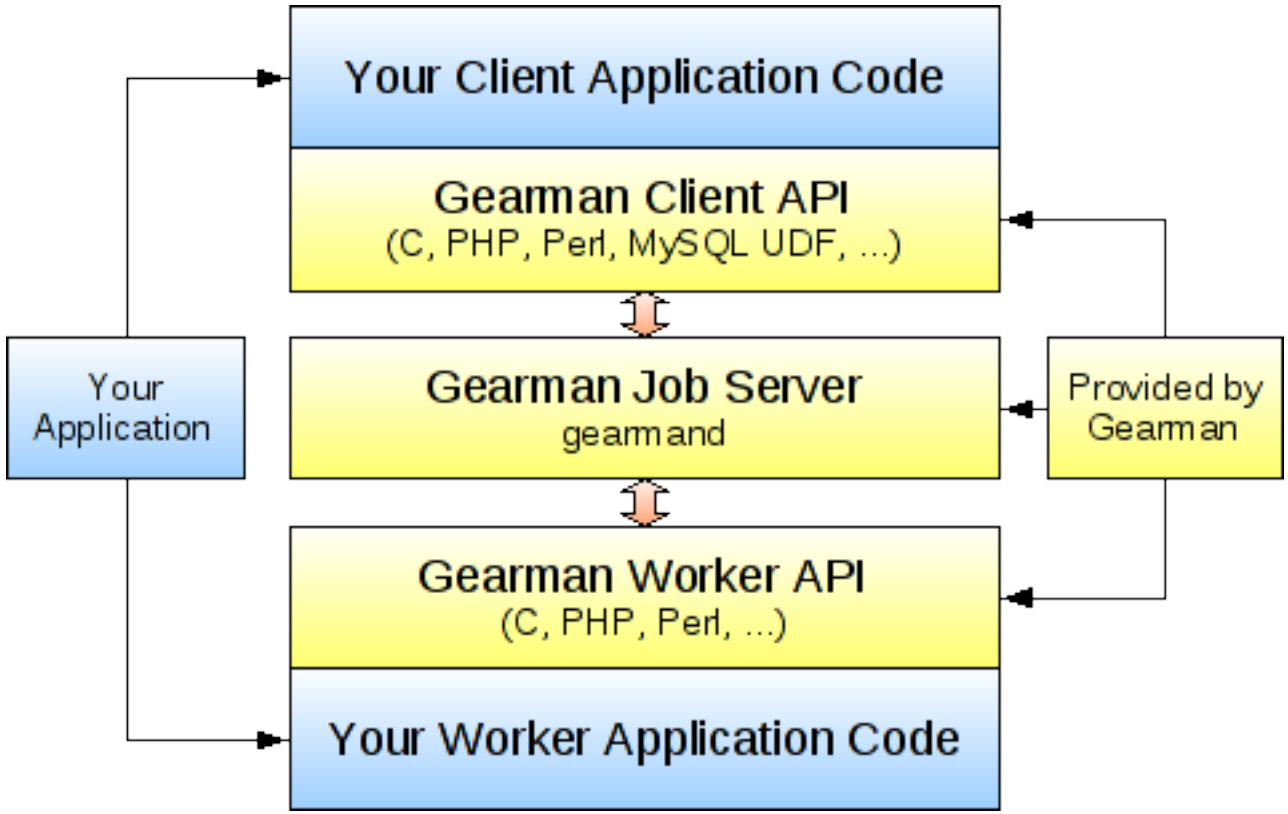


image borrowed from http://en.wikipedia.org/wiki/Gearman

Distributing Work: Gearman Benefits

- very cheap scaling across different nodes
  - "gearman-enabling" a function is a few line change
  - adds a little overhead per task, but most jobs are long running enough to outweigh this
- very cheap asynchronous tasks from servlets
  - good for expensive calls where you just need a guarantee that the job gets done but don't need the result of that call
    - set user MRU location
- fault tolerance
  - put workers on all machines -- if one batch machine goes down (OOM or something), the others will try to handle the workload

Creating a Gearman Task

- Called by in servlet execution

```
def set_mru_by_user_id(self, user_id, location):
    """Set a user's most recently used location
    Args:
    enc_user_id -- the user to update
    location -- dict, a location dictionary as returned from the Geocoder
    """
    try:
        keyword_arguments = { 'user_id':user_id, 'location':location }
        gearman_client = yelp.core.connections.get_gearman_client()
        gearman_client.submit_job(
            config.gearman_tasks.user_recent_location_SET_MRU_TASK,
            keyword_arguments, background=True # set async
        )
    except Exception, e:
        log.exception("Failure queuing gearman task for user_location")
```

Consuming a Gearman Task

```
import config.gearman_tasks
from batch.gearman_workers.gearman_worker_daemon import YelpGearmanWorkerDaemon, task_listener
from logic import user_recent_location, txn

class UserRecentLocationWorker(YelpGearmanWorkerDaemon):
    """Worker listening to user_recent_location set_mru requests"""

    @task_listener(config.gearman_tasks.user_recent_location_SET_MRU_TASK)
    def user_recent_location_mru_update(self, keyword_arguments):
        with txn.begin(self.conns.aux) as cursor:
            self.logic.UserRecentLocation._set_mru_by_user_id(cursor,
                keyword_arguments['user_id'],
                keyword_arguments['location']
            )

if __name__ == "__main__":
    batch = UserRecentLocationWorker()
    batch.start()
```

More Complicated Workflows

- What if you have many long running tasks that need to be broken up themselves?
- master/slave configuration
  1. submit a job for the long running master task
  2. master task does set up work, then creates many slave tasks and submits them
  3. master task waits until all slave tasks have finished
  4. master task does final step work & finishes job

A Yelpy Example

- About 40 cities receive a Weekly Yelp
- Cities have anywhere from 10,000 to 240,000 recipients

1. Create a weekly_generation_master job for each city with weekly_id and time_to_send parameters
2. 1 of 2 dedicated weekly_generation_master workers will start job
    - pulls list of recipients, chops into segments
    - create a weekly_generation_slave job for each segment with weekly_id and recipient user_ids
3. 15 weekly_generation_slave workers concurrently render segments
4. When all slave workers for a master have finished, master does final steps and queues weekly to send

Other Gearman Stuff

- A job is guaranteed to finish, but not on the first try
  - even if there are seemingly no failures, the job can be re-done
  - plan around it!
    - make sure there's a lightweight way check and see if the task you're about to start has already been done
    - context specific checks (have these recipients already been rendered)
    - lock table with unique task IDs
- Yelp has adopted the python-gearman API
  - http://github.com/yelp/python-gearman

Elastic Map Reduce in a Nutshell

- What is Map Reduce?
  - Massive parallelization of independent problems
  - Map step
    - Split your Big Problem into many smaller problems, distribute them to each of your 'mappers'
  - Reduce step
    - Once a small problem has been solved, the result gets sent to the reducer where it is combined with other small problem results
    - Reducers keep running until all result sets have been combined

```
python batch/loggrep.py -v
    --min-date=2010-10-05 --max-date=2010-10-05
    --pattern='/weekly.*' --pattern='/biz/yelp-san-francisco.*'
    --ec2-instance-type=c1.xlarge --num-ec2-instances=5
```

Elastic Map Reduce in a Nutshell

```
class MRGrep(mr_job.MRJob):
    def process_log_line(self, line):
        try:
            log_dict = ApacheLogDict(line, as_datetime=True)
        except ApacheParseError:
            return

        line_to_match = line if self.options.grep_full_line else log_dict['page']
        if not any([re.search(pattern, line_to_match) for pattern in self.options.pattern]):
            return
        # similar to "GROUP BY" in SQL
        key_dict = yelpy.filter_by_keys(log_dict, self.options.unique_log_fields.split(','))
        if 'time' in key_dict:
            time_var = key_dict.pop('time')
            key_dict['time'] = time_var.strftime(self.options.time_format)

        return (key_dict, 1)

    def mapper(self, _, line):
        key_value = self.process_log_line(line)
        if key_value is not None:
            yield key_value

    def reducer(self, key, values):
        yield key, sum(values)
```

Running EMR Job

2010-10-06 22:47:27,697 - batch.loggrep: INFO startup: command=`batch/loggrep.py -v --min-date=2010-10-05 --max-date=2010-10-05 --pattern=/weekly.* --pattern=/biz/yelp-san-francisco.* --ec2-instance-type=c1.xlarge --num-ec2-instances=5`, version=name = 'main 1'
2010-10-06 22:47:27,701 - batch.loggrep: INFO Example of formatted date string: 2010-10-06
2010-10-06 22:47:27,702 - batch.loggrep: INFO Matching any of the following regular expressions:
2010-10-06 22:47:27,703 - batch.loggrep: INFO * /weekly.*
2010-10-06 22:47:27,703 - batch.loggrep: INFO * /biz/yelp-san-francisco.*
2010-10-06 22:47:27,704 - batch.internalapps.weekly_metrics: INFO ['page', 'time']
2010-10-06 22:47:27,704 - batch.loggrep: INFO instantiating batch.mr_grep.MRGrep with args: --pattern /weekly.* --pattern /biz/yelp-san-francisco.* --unique-log-fields page,time --time-format %Y-%m-%d --runner emr --ec2-instance-type c1.xlarge --num-ec2-instances 5 s3://yelp-scribe-logs/logs/access/2010/10/05/*.gz
2010-10-06 22:47:27,710 - mrjob.emr : INFO looking for configs in /nail/home/derwiki/.mrjob
2010-10-06 22:47:27,711 - mrjob.emr : INFO looking for configs in /nail/pg/derwiki/loc/mrjob.conf
2010-10-06 22:47:27,712 - mrjob.emr : INFO found configs at /nail/pg/derwiki/loc/mrjob.conf
2010-10-06 22:47:28,205 - mrjob.runner: INFO creating tmp directory /scratch/derwiki/mr_grep.derwiki.20101006.224727.743542
2010-10-06 22:47:28,206 - mrjob.runner: INFO writing wrapper script to /scratch/derwiki/mr_grep.derwiki.20101006.224727.743542/wrapper.py
2010-10-06 22:47:28,213 - mrjob.emr : INFO writing master bootstrap script to /scratch/derwiki/mr_grep.derwiki.20101006.224727.743542/b.py
2010-10-06 22:47:28,224 - mrjob.emr : INFO Copying non-input files into s3://yelp-emr-dev/tmp/mr_grep.derwiki.20101006.224727.743542/files/
2010-10-06 22:47:40,556 - mrjob.emr : INFO Waiting 5.0s for S3 eventual consistency
2010-10-06 22:47:45,550 - mrjob.emr : INFO Creating Elastic MapReduce job flow
2010-10-06 22:47:45,840 - mrjob.emr : INFO Job flow created with ID: j-F1PE2B03DAU4
2010-10-06 22:48:15,917 - mrjob.emr : INFO Job launched 30.1s ago, status STARTING
2010-10-06 22:48:46,057 - mrjob.emr : INFO Job launched 60.2s ago, status STARTING
2010-10-06 22:49:16,119 - mrjob.emr : INFO Job launched 90.3s ago, status STARTING

Hackathons

- Every 6 months
- Self organized teams
- 2 days to work on *anything*
- Anything?
  - KegMate

Work Environment

- Career page doesn't lie -- nerf dart warzone
- At least 3 kegerators
  - 1 is connected to an iPad (KegMate)
- Learning Groups at least twice a month
  - software transactional memory
  - Fortran and slide rules
  - non-blocking I/O
  - Burning Man
  - photography
- Darwin!

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