PyCon2013 Schedule & Notes Documentation

Release 1.0

Andrew Schoen

March 16, 2013

CONTENTS

Mark Ransom's live notes from PyCon US 2013.

CHAPTER

ONE

THURSDAY

1.1 Tutorial: Managing Python App Performance

Presenter: Graham Dumpleton, New Relic

Track: Sponsor Tutorials

Description:

> There's no longer any question of whether dynamic languages are appropriate for both large and small scale apps–you use the languages and tools that get the job done. Python suits environments where rapid change is the norm.

1.1.1 Introduction to New Relic

New Relic uses monkey patching to inject their performance tracking code. If you're using gunicorn or wsgi, it's a simple magic wrapper script.

They're not doing full on profiling of the application because there is too much overhead, so instead they use wrap the wsgi entry point and monitor that.

They also have instrumentation for monitoring:

- Middlewares
- Database layer
- Templating

They will pull out the view handlers and use those as the transaction names.

It's not an exhausting stack trace, they try to strip out all sensitive information, but they do give you a lot of the same information that you would see in a django stack trace page.

1.1.2 Troubleshooting Common Application Problems

http://newrelic-python-kata.herokuapp.com/

It takes a few minutes for transactions to start showing up on their web interface.

Once the transactions start showing up in your transaction list, click into the transaction allows you to see a summary of the view, and you can drill down into the actual SQL that's being called.

Kata 1

In the example, we see that on the employee_list view, we're getting 50 employees from the database, and for each employee we're making another request to the BioData and Payroll tables.

We can drill into the template to see that we're making a seperate request for each employee's bio and payroll info.

Kata 2

A view is loading very slowly, we can drill into the transaction to see that it is the template rendering that is taking all the time, so we are able to drill in and see that we're doing filtering in the template that should be happening in the view.

Kata 3

New Relic will capture your exceptions instead of emailing them to you. This way you can set threshholds so that if your application starts throwing a lot of errors you will be notified.

In this kata, we're using an view that caltulates factorials so that we can pass in invalid values and see that some requests are throwing exceptions that we can see the stack trace for them.

Kata 4

Making a request to a weather api to pull the weather for 10 different cities, and it's taking a while to load. We can see in the transaction breakdown that it is the API call that is taking so much time. We can implement caching of weather data to speed up page load times.

1.2 Tutorial: Integrating Selenium and Sauce Labs into your Continuous Integration Build

Presenter: Jason Carr, Sauce Labs

Track: Sponsor Tutorials

Description:

CI can be hard. Making Selenium a part of your CI infrastructure is even harder. This tutorial will cover the end-to-end process of adding Selenium infrastructure to your CI build using Sauce Labs, and integrations with Jenkins and Travis CI.

1.2.1 Selenium WebDriver

WebDriver lets you send your script to a WebDriver server to execute. The script gets translated into commands to drive the browser, which then returns a json response.

Sometimes you have to wait for the internet, but Selenium tests are run as procedural scripts, so you will often have to put in explicit waits for things to load.

Selenium in the Cloud

It's possible to run your tests in the cloud on SauceLabs infrastructure, you just specify desired_capabilities to the different features you want:

- browserName - Firefox
- platform - Linux
- etc

Security Through Purity They completely destroy a VM when you are done running your tests, so your VM never gets used again, and you never get a VM that was used by someone else.

Selenium is just a library, it's not only for testing, it's just for controlling a browser from a script.

Quick Intro to Selenium

Some simple code examples:

```
>>> from selenium import webdriver
>>> driver = webdriver.Chrome()
>>> driver.get('http://www.google.com')
driver.title
>>> driver.title
u'Google'
>>> driver.find_element_by_name('q').send_keys('pycon')
>>> driver.title
u'pycon - Google Search'
```

Cloud Meets Local Deve

It's possible to run tests on the SauceLabs cloud that are hitting your local dev machine, using a built in vpn solution.

You can watch the tests running through a browser based VPN session directly to the VM.

1.2.2 Tools To Use

- **Jenkins Server**
  - Github plugin
  - InjectEnv plugin
  - Sauce OnDemand plugin
- virtuaenv/pip/distribute
- Selenium client library
- Sauce Connect
- Sauce Labs Account
- sauceclient library

Basic Steps

- Install Jenkins
- Install Sauce OnDemand plugin
- **Configure Sauce OnDeminad plugin in Jenkins**
  - Enter your api username and api key
  - Test your connection (button in the config page)
- Configure the git[hub] plugin
- Configure the github webhook configuration so that your tests get run on every push

It's possible to get unique ids for accessing test runs without passing credentials around, which is good on a large team where you don't want everyone to have to log in to see test run output.

Dynamic Test Classes

https://github.com/santiycr/cssify/blob/master/tests/test_cssify_web.py

CHAPTER

TWO

FRIDAY

2.1 Messaging at Scale at Instagram

Presenter: Rick Branson

Track: II

Description:

As activity accelerated from just a few thousand activities per day to hundreds of millions, Instagram needed a reliable, scalable messaging infrastructure to distribute work and messages. In this talk, I'll jump from a crash course in the abstract concepts of queueing into the implementation details & hard-earned know-how from experience building massive-scale Python-based systems.

> https://us.pycon.org/2013/schedule/presentation/106/

Trying to get photos for friends of friends is expensive.

You should try to get out of the request ASAP.

Justin Beiber effect: Hundreds of thousands of followers.

2.1.1 Gearman & Python

- Persistence is slow
- Ran out of memory
- Pulling based, so workers pull
- Should have used Redis

2.1.2 Celery

- Fast
- Great Python support
- celeryd to keep things going

2.1.3 RabbitMQ

- Not as fast as Redis, but reasonably fast
- Mirrored broker nodes
- Over previsioned so they can burst up to higher capacity

2.1.4 Alerting

They use Sensu (Ruby)

2.1.5 Graphing

Graphite and statsd

2.1.6 Brokers

They use a round robin boker approach

2.1.7 Performance

- They push about 4,000 tasks per second
- ~25,000 app threads publishing tasks

2.1.8 Why They Chose Celery

They cna get new engineers up to speed quickly

2.1.9 Scaling Out

- Back in the day Celey only supported 1 broker host
- They created kombu-multibroker

2.1.10 Gevent

Only some of their tasks run on gevent, some are on multiprocessing mode. Celeryd_multi allows running tasks in different worker modes.

They us Gevent for anything network bound, and anything that needs network bound functionality and local actions they split it up with callbacks.

2.1.11 Problems

- Slow tasks monopolize workers
- Running higher concurrency is inefficient
- Lower batch size is also inefficient

They isolated their feed delivery, because anything that you don't want to get backed up by slow tasks should be on its own worker.

They have three concurrenc levels

- Fast
- Feed (important)
- Default

They start new tasks out in default and then promote them to Fast as they prove themselves to be fast.

2.1.12 Failures

It's impossible to determine whether a task has died or is just really slow, so it's important that tasks be idempotent so thta you can retry.

You need acknowledgements for when tasks finish successfully.

They only pass self-contained, non-opaque data as arguments to tasks.

Tasks should execute within a few seconds, otherwise restarts take a long time and they gum up the works. They use a soft time limit of 20 seconds, and a hard time limit of 30 seconds.

2.1.13 Future

- They'd like to get better
- Utilize results storage and other celery features they aren't using now
- Single cluster for control queues, becuas they're breaking all the management

tools for Celery right now * Eliminate their multi-broker shim (kombu-multibroker) now that celery

supports multiple brokers

2.2 This Old Video Site: How PBS streams video

Presenter: Edgar Roman

Track: II

Description:

Overview of how the Public Broadcasting Service streams video online. Learn how PBS uses python and other services to provide video streaming online. Talk will discuss lessons learned, explanation of video formats, and experiences with mobile device support. Talk will include recommendations for others to easily adopt similar practices to quickly host their own online video site.

https://us.pycon.org/2013/schedule/presentation/133/

2.2.1 Goals

PBS wants to be as accessible as possible, which means supporting as many devices as possible.

H.264 is widely supported, but it lives under a legal cloud.

They want to play ads and have flexibility to support social sharing.

2.2.2 Delivery

You can use HTTP, but they use RTMP at PBS.

In 2009 the standard was 400kbps, but now it's more like 800kbps to 1.2mpbs

2.2.3 Targets Devices

- **Apple's iOS**
  - No Flash
  - Http Live Streaming (HLS)
  - Auto bitrate adjust possible
  - Any CDN will do
  - Built-in player
- **Android**
  - So many variants that it's difficult to determine what the device will support
  - Later versions support HLS
  - End up using MP4 baseline

2.2.4 HTML5 Video Tag

Great for supported devices, but doesn't support the rich environment features that they need, especially advertising and captioning.

There are some really good frameworks:

- VideoJS
- MediaElement
- JWPlayer

2.2.5 Transcoding

Local:

- ffmpeg
- x264 Handbrake

Online:

- Zencoder

- Encoding.com

They create 16 streams for each video, but they start wth 5 Mbps

2.2.6 Stealing

PBS offers free streaming lonline to there is less motivation to steal.

Beware of the DRM Graveyard

2.3 Making DISQUS Realtime

Presenter: Adam Hitchcock

Track: II

Description:

What does it take to add realtime functionality to a truly web scale app. The result is the DISQUS realtime system, a highly concurrent system for allowing web clients to subscribe to arbitrary events in the DISQUS infrastructure.

https://us.pycon.org/2013/schedule/presentation/46/

2.3.1 DISQUS

People are more engaged when things happen in real time.

They get a lot of traffic, over a billion hits per month.

They call it RealERtime.

They use:

- Thoonk
- Redis
- pub sub
- nginx

2.3.2 Real Time Progression

Old system was memcache pulling every 5 seconds.

New system is redis pub/sub and a Flask proxy cluster.

They quickly ran out of CPU on the Flask servers, so they moved things to a backend server to handle formatting.

This was better, but the Flask servers were still growing too quickly.

Replaced Flask servers and HA Proxy servers with an Nginx pub endpoint.

2.3.3 Thoonk

The Thoonk queue takes post_save and post_delete hooks, sits on tops of Redis.

Provides job semantics (what's claimed, what's not)

2.3.4 Gevent

Gevent is the best thing ever, it lets you process things really fast.

Works with pipelines and mixins to compartmentalize logic.

Data pipelines using mixings (each mixin manipulates the data and passes it on).

2.3.5 Nginx is Great

Replaced webservers and redis pub/sub

> http://wiki.nginx.org/HttpPushStreamModule

EventSource Lets the browser handle the async calls instead of making javascript do it.

2.3.6 Testing

Darktime - Use existing network to load test, pull out a certain percentage of your user base at the new code.

Darkesttime - Testing a single instance with a ton of traffic.

Measure everything - Especially when numers don't match up, even if it's hardin a distributed system, and try to express things as +1 and -1 if you can.

When the pope was announced they saw over 6TB of traffic that day.

2.3.7 Lessons

- Do hard work early
- End-to-end acks are good, but expensive
- redis/nginx pubsub is effectively free
- Greenlets (gevent) are free too

2.3.8 Also

Check out HttpPushStreamModule!!!

2.4 Twisted Logic: Endpoints and Why You Shouldn't Be Scared of Twisted

Presenter: Ashwini Oruganti

Track: III

Description:

This talk will be a survey of my learning experience adding new endpoint APIs to Twisted, an event-driven networking engine (as a Google Summer of Code project), with a special focus on the analysis of some of the horror stories that surround Twisted. Right from the asynchronous I/O model to Deferreds: if it scares you, we'll figure a way out and see what the makers of Twisted say when confronted.

https://us.pycon.org/2013/schedule/presentation/40/

2.4.1 Endpoints

Standardized APIs for connecting clients and listening for requests.

Example:

```
endpoint = TCP4ServerEndpoint(reactor, 8007)
endpoint.listen(Factory())
```

It's Just Code

2.4.2 The Moral

- Don't get flustered
- Don't overthink
- Forget it's Twisted

Read the code -> Solve the Problem

Callbacks can be confusing, and difficult to debug.

It's very complicated, it's even in the name (Twisted).

Part of the problem is that the idea of the Twisted framework is foreign, it's different than the other frameworks that we use (django, etc). But you're using Twisted because it has something that the other alternatives don't have, so even if the ideas around the framework are different than what you're used to, you know why you need it, so you have a frame of reference, you just need to lean on that.

2.5 Visualizing Github, Part I: Data to Information

Presenter: Dana Bauer, Idan Gazit

Track: III

Description:

A treasure trove of data is captured daily by Github. What stories can that data tell us about how we think, work, and interact? How would one go about finding and telling those stories? This two-part talk is a soup-to-nuts tour of practical data visualization with Python and web technologies, covering both the extraction and display of data in illumination of a familiar dataset.

https://us.pycon.org/2013/schedule/presentation/112/

2.5.1 Data and Mapping

Github focuses a lot on charts and graphs because so much of the data that they have focuses on activity over time.

They didn't even know what data they had.

Ben Fry - Visualizing Data

It's important to work with a team of people that have a mix of skills, no steps in the process should be completed in isolation.

- Acquire
- Prase
- Filter
- Mine
- Represent

2.5.2 Acquiring the Data

- 3.4 million users
- 5.7 million repos

They pulled down the top 200 repos of the top 10 languages, put it in the cloud on Amazon EC2, and started cloning all of the repos.

2.5.3 Working With The Data

Now they wanted to start working with their data.

They worked within iPython Notebook, but if they got disconnected it was still a problem becuase they couldn't see where they were currently at.

Not all of the information is in the git repos, such as users and github specific information.

They had to hit Github's API, but rate limiting was still an issue.

The data was changing constantly, so there was some distortion in the snapshot of the data.

2.6 Python Profiling

Presenter: Amjith Ramanujam

Track: II

Description:

This talk will give a tour of different profiling techniques available for Python applications. We'll cover specific modules in Python for doing function profiling and line level profiling. We'll show the short comings of such mechanisms in production and discuss how to do sampled profiling of specific functions. We'll finish with statistical profilers that use thread stack interrogation.

https://us.pycon.org/2013/schedule/presentation/86/

2.6.1 cProfile

Python - Batteries Included. It comes with its own profiler, **cProfile**:

```
python -m cProfile my_scripy.py
```

Sometimes you get a ton of output, so you should save the output to a file:

```
python -m cProfile -o file.prof my_script.py
```

RunSnakeRun is a GUI interface for analyzing profile files created with cProfile.

A Catch

Using the profiler adds overhead to your code, and it runs slower, so you don't want to run the profiler in production.

It's slow

So you could only profile critical functions by using a decorator to start the profiler at the beginning of the function and turn it off at the end.

Most likely your critical functions are the ones you wnat to be fast, so you don't want them being profiled all the time.

2.6.2 Statistical Profiler

Kind of like an Overly Attached Girlfriend.

Interrupted - "Hey" Inquired - "Where are you? What are you doing?" Collate - "You don't love me anymore!"

This all has overhead.

Inquire: Stack frame of every thread, every 100 ms.:

```
>>> import sys, traceback
>>> frames = sys._current_frames()
>>> stack = traceback.extract_stack(frames)
```

StatProf Uses unix signals and the CLI to profile stack traces.

Plop Uses unix signals with a callback.

2.6.3 X-Ray Sessions

Was secret, but now it's in beta. What it does is, you can pick a specific page and you want as much information as possible. You can run an x-ray session on that page and they will collect the first 100 traces (transactions) that run on that page.

Normally you only get targeted instrumentation, but this will run the profiling on the entire trace, so you can come look at the profile and see exactly what's going on. They also provide you with histograms of those 100 requests, so you can see exactly what percentage of those requests fell within a secific range.

2.7 Transforming Code into Beautiful, Idiomatic Python

Presenter: Raymond Hettinger

Track: VI

Description:

Learn to take better advantage of Python's best features and improve existing code through a series of code transformations, "When you see this, do that instead."

https://us.pycon.org/2013/schedule/presentation/126/

Examples:

```
for x in range(len(colors)):
    print colors[x]

# Should be
```

```
for x in colors:
    print colors


n = min(len(names), len(colors))
for i in range(n):
    print names[i], '-->', colors[i]

# Should be
for name, color in zip(names, colors):
    print name, '-->', color

# But zip is slow, so you should use izip:
for name, color in izip(names, colors):
    print name, '-->', color
```

Awesome components:

- zip/izip
- defaultdict
- Chain
- deque
- localcontext
- with open() as f:
- with ignored(SpecificException): do something that could throw an exception

CHAPTER

THREE

SATURDAY

3.1 Who's there? - Home Automation with Arduino/RaspberryPi

Presenter: Rupa Dachere

Track: II

Description:

Have you ever found yourself obsessively checking UPS or FedEx tracking site to see if your package finally got delivered at your doorstep? Or wondered when your contractor/gardener showed up to do their job? Come join me to learn how to build your own gadget to notify you when your package or contractor shows up at your doorstep!

https://us.pycon.org/2013/schedule/presentation/75/

3.1.1 Who's At The Door?

Wanted to be able to see who was at the door.

Arduino with a proximity sensor -> USB -> Raspberry Pi with a webcam -> take a picture and upload to web -> SMS message using Twillio with a link to the picture.

3.2 Mobile Application Testing with Python and Selenium

Presenter: Jason Carr

Track: IV

Description:

Selenium has grown to be a mature platform on the desktop, but with 'mobile now' being the mantra for so many companies, can we use Selenium to effectively test mobile apps? What about Native apps? This talk will cover using Python to test mobile web applications with Selenium, as well as an in depth overview of the future of Selenium to test Native iOS and Android applications.

https://us.pycon.org/2013/schedule/presentation/79/

3.2.1 Selenium is Just a Library

It's a tool, like anything else, and it has its place in your toolbelt.

Selenium RC is the old bustedness. It doesn't support some mobile browsers, things like that.

Selenium WebDriver is the new hotness, aka Selenium 2. It supports mobile browsers.

It is pruposed that the low level web driver functionality will become a standard and will be built directly into browsers in the future.

3.2.2 Using Selenium on a Mobile Browser

Every mobile browser is a webdriver, because it's not running on your computer that is running the actual Selenium script:

```
from selenium import webdriver

desired_capabilities = {}
desired_capabilities['browser'] = 'android'
desired_capabilities['platform'] = 'linux'
desired_capabilities['version'] = '4.0'
```

iOS

iOS Webdriver is gimped, won't even let you quit, clear cache, etc, and it requires Xcode.

Android

- Requires the Android SDK
- An APK is created that you run in the emulator
- Setup is hard and annoying

Setup

- Boot up emulator
- Connect to adb server
- Install the .apk file
- Forward tcp:8080 to tcp:8080 on the phone
- Run your Selenium test

3.2.3 Limitations

WebViews - A window to the internet provided by the OS that you can put in your app, they aren't maintained the same way as full fledged browsers. Second class citizens.

They are hard to accurately test, and it's hard to test on hardware devices.

Native App Testing

*iOS**

Currently the only option is UI Automation

- Javascript only
- Limited command line control

- No interaction with test

Android

Not much better, UI Automator

- Java tests, compiled and pushed as jars
- No interaction
- No test interoperability

Alternativess

- Require recompiling apps to add code
- Varias APIs
- Mixed community and limited help
- Forced language implementation, so you can't use the tools you want

This is all BAD

3.2.4 Enter Appium

- No recompilation of app required, so what you test against is what you can

deploy to the app store * Use Selenium API * All methods are first class citizens, they wrap the test frameworks that the companies provide to you * Any language and OS, supported by Selenium * Open source

Wraps UI Automation and UI Instruments

Because it uses the native testing framework (and wraps it), it supports all of the browser and device functionality that the native testing framework exposes.

iOS

Your Script -> Appium Server -> iOS UI Instruments -> Test App

Android

Your Script -> Appium Server -> Android UI Automator -> Test App

3.2.5 What's Next

They not only wrapped the native UI Automation functionality, but they also wrapped the Mobile Webkit implementation, because they want it to work on real devices. We want ito test W3C compliant browsers whenever possible.

- Drive a real browser
- Actual rendering
- Standards compliant

3.3 Designers + Developers: Collaborating on your Python project

Presenter: Julia Elman, Mark Lavin

Track: II

Description:

Working in teams is an important part of what we do as developers & designers. Whether it's desktop applications or mobile sites, we work together to create successful end products. But how do we work together in different environments? What is the best work-flow for a mix of skill sets?

We'll be talking about our various methods & work-flows that we found successful in working collaboratively.

https://us.pycon.org/2013/schedule/presentation/56/

3.3.1 You've Got an Idea

Sometimes you get an idea for a project, you start mulling it over and eventually you decide to work on it and try to make something. You might start recruiting people, and you're likely going to recruit from your circle of friends, which means that they are probably a lot like you.

A Designer

Maybe you should try to recruit a designer? A stock UI framework (Twitter Bootstrap, for example) will only get you so far.

You need to bring in people who compliment your skillset, because if you bring in more people like you, than the will help you get things done faster, but maybe not better.

Beware Too Many Cooks

Keep your team compact, be selective, but seek for diversity.

3.3.2 Comfort Zones

Be careful not to go off into your own world and do your own thing, then try to smash the whole thing together at the end. Don't work in a silo. Be willing to get outside of your comfert zone and expertise. Other people need your feedback, even if you'r enot an expert in their line of work, because you can have ideas that they might not think about.

Designers

It can be scary for designers (or anyone, really) to try to work with the terminal, it can be very intimidating. But designers should just dive in.

One of the best parts of collaboration is the knowledge exchange. Developers should be patient and help the designers, answer questions, and direct them to good tools. It will make your life easier to have your team up to speed.

Developers

We have to be willing to give up some control. Many developers fall into two categories:

- They don't have the confidence to say what is good design and what isn't
- They don't care enough to pay attention

You can and should know what good design is. Pay attention to things that work intuitively, make a mental note of them, bring that to the table.

Developers should work on being better designers, think about how to make your software more intuitive for the users.

3.3.3 Automation

Use your expertise to remove the cruft so that you can focus on the idea. Automate the menial tasks.

- Setting up dev environment
- Deployments
- Builds
- Testing

Let designers and developers focus on the details of the idea.

3.3.4 Documentation

Document everything you can't automate. Tailor those documents to every type of person that needs them. If you want people to get involved in your awesome idea, you should document what you have so far.

3.3.5 Bottom Line

In good collaboration you are stretched beyond your skillset and you learn something.

Also, in a good collaboration, failure becomes an option, because the act of the collaboration becomes the real positive outcome. The success of the projcet itself matters less because you have improved yourself.

It's ok to fail in a good collaboration, as long as you don't fail at collaborating.

3.3.6 Example

Wanted in page editing, using the power of the Django templating language, and wanted the developer to be able to put in sane defaults for when there isn't anything in the database.

Inventing on Principle by Bret Victor - You should be able to see what you're changing, while you're changing. vimeo.com/36579366

Developers start thinking about the technical challenges right away, they start thinking about what they know they can do, and become sceptical.

They started talking about it to people, who shared additional ideas.

Django Scribbler - http://www.github.com/caktus/django-scribbler/

3.3.7 Closing Thoughts

People can contribute a lot more to a project than just git commits.

CHAPTER

FOUR

SUNDAY