

# Building Scalable Apps using Google App Engine

[bit.ly/scalableapps](http://bit.ly/scalableapps)

PyCon India 2010

Pranav Prakash



# Agenda

- Scalability
- Introduction to Google App Engine
- Hello World
- App Engine DataStore
- Cache
- Scalability Revisited
- Case Study
- Q-n-A



# Scalability

- Scaling Up
- Scaling Out



# Google App Engine

- Develop, Deploy web apps on Google's Infrastructure
- Application Environment
- Sandbox Environment
- Runtime Environment
  - Python
  - Java
- DataStore
- Google Accounts
- Services
- Cron Jobs and Task Queues



# Hello World :)

```
from google.appengine.ext import webapp
from google.appengine.ext.webapp.util import run_wsgi_app

class MainPage(webapp.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello, webapp World!')

application = webapp.WSGIApplication(
    [('/', MainPage)],
    debug=True)

def main():
    run_wsgi_app(application)

if __name__ == "__main__":
    main()
```



# app.yaml

```
application: helloworld  
version: 1  
runtime: python  
api_version: 1
```

```
handlers:
```

```
- url: /*  
  script: helloworld.py
```



# App Engine Datastore

- Scalable data storage for your applications
- Write once, read many
- Stores data entities with properties, organised by application defined Kinds
- Queries on entities of same kind
- Filter and sort order on property values and keys
- Pre-Indexing



# Entities and Models

- Datastore Entity = key + set(attributes)
- Models describe the kind of data an app uses

```
from google.appengine.ext import db  
import datetime
```

```
class Person(db.Model):  
    name = db.StringProperty(required=True)  
    birthdate = db.DateProperty()  
    height = db.IntegerProperty()  
    is_admin = db.BooleanProperty(default=False)
```

```
ted = Person(key_name='person_ted',  
             name='Ted',  
             birthdate=datetime.datetime(1986,12,04),  
             height = 185)  
db.put(ted)
```





# Entity Groups, Ancestors and Path

- Entities residing in same part of the distributed network
- Transactions
- Parent Child relationship
- Path and Key uniqueness



# Fetching Entities

- GQL
- GqlQuery
- Query
- db.get()

```
all_teds = Person.gql("WHERE name = :1", "Ted").fetch(100)
```

```
all_teds_1 = GqlQuery("SELECT * FROM Person WHERE name = :1",  
"Ted").fetch(100)
```

```
all_teds = Person.all().filter("name = ", "Ted").fetch(100)
```

```
specific_ted = Person.get_by_key_name('person_ted')
```

```
specific_teds_key = db.Key('Person', 'person_ted')  
specific_ted_1 = db.get(specific_ted_key)
```



# Cache

- Memcache
- App Caching



# Scalability Revisited

- Read and Write Infrequently
- Keys, Key Names, ID
- Batch Reads and Writes
- Small Entity Groups
- Sharding
- Memcache



# Case Study

- Social apps and games from Oxylabs





[bit.ly/scalableapps](http://bit.ly/scalableapps)

pranny@gmail.com

