Your easy move to serverless computing and radically simplified data processing

Dr. Gil Vernik, IBM Research
About myself

- Gil Vernik
- IBM Research from 2010
- PhD in mathematics. Post-doc in Germany
- Architect, 25+ years of development experience
- Active in open source


Twitter: @vernikgil

https://www.linkedin.com/in/gil-vernik-1a50a316/
Agenda

- **What** problem we solve
- **Why** serverless computing
- **How** to make an easy move to serverless
- Use cases
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 825184.
The motivation..
Simulations

- **Alice** is working in the risk management department at the bank.
- She needs to evaluate a new contract.
- She decided to run a **Monte-Carlo simulation** to evaluate the contract.
- There is need about 100,000,000 calculations to get a better estimation.
The challenge

How and where to scale the code of Monte Carlo simulations?

Business logic
Data processing

• **Maria** needs to run face detection using TensorFlow over millions of images. The process requires raw images to be pre-processed before used by TensorFlow

• Maria wrote a code and tested it on a single image

• Now she needs to execute the same code at massive scale, with parallelism, on terabytes of data stored in object storage
The challenge

How to scale the code to run in parallel on terabytes of data without become a systems expert in scaling the code and learn storage semantics?
Mid summary

• How and where to **scale** the code?
• How to process massive data sets without become a storage expert?
• How to **scale** certain flows from the existing applications without major disruption to the existing system?
VMs, containers and the rest

• Naïve solution to scale an application - provision high resourced virtual machines and run your application there
  • Complicated, Time consuming, Expensive

• Recent trend is to leverage container platforms
  • Containers have better granularity comparing to VMs, better resource allocations, and so on.
  • Docker containers became popular, yet many challenges how to ”containerize” existing code or applications
  • Comparing VMs and containers is beyond the scope of this talk… 🙃

• Leverage Function as a Service platforms
# main() will be invoked when you Run This Action.
#
# @param Cloud Functions actions accept a single parameter,
# which must be a JSON object.
# @return which must be a JSON object.
# It will be the output of this action.
# 
# import sys

def main(dict):
    if 'name' in dict:
        name = dict['name']
    else:
        name = 'Strata NY'
    greeting = 'Hello ' + name + '!
    print(greeting)
    return {'greeting':greeting}
Function as a Service

- Unit of computation is a function
- Function is a short lived task
- Smart activation, event driven, etc.
- Usually stateless
- Transparent auto-scaling
- Pay only for what you use
- No administration
- All other aspects of the execution are delegated to the Cloud Provider
Are there still challenges?

- How to integrate FaaS into existing applications and frameworks without major disruption?
- Users need to be familiar with API of storage and FaaS platform
- How to control and coordinate invocations
- How to scale the input and generate output
Why is it still “complicated” to move workflows to the Cloud? User need to be familiar with cloud provider API, use deployments tools, write code according to cloud provider spec and so on.

Can FaaS be used for broad scope of flows? (RISELab at UC Berkley, 2017)

**Occupy the Cloud: Distributed Computing for the 99%,**
(ERIC JONAS, QIFAN PU, SHIVARAM VENKATARAMAN, ION STOICA, BENJAMIN RECHT, 2017)

PyWren - an open source framework released
Push to the cloud with PyWren

- Serverless for more use cases (not just event based or “Glue” for services)
- Push to the Cloud experience
- Designed to scale Python application at massive scale
Cloud Button Toolkit

- **PyWren-IBM** (aka CloudButton Toolkit) is a novel Python framework extending PyWren
- 600+ commits to PyWren-IBM on top of PyWren
- Being developed as part of CloudButton project
- Led by IBM Research Haifa

Open source [https://github.com/pywren/pywren-ibm-cloud](https://github.com/pywren/pywren-ibm-cloud)
PyWren-IBM example

data = [1,2,3,4]

def my_map_function(x):
    return x+7

import pywren_ibm_cloud as cbutton

cb = cbutton.ibm_cf_executor()

cb.map(my_map_function, data)

print (cb.get_result())

[8,9,10,11]
PyWren-IBM example

data = "cos://mybucket/year=2019/"

```python
def my_map_function(obj, boto3_client):
    # business logic
    return obj.name
```

```python
import pywren_ibm_cloud as cbutton
cb = cbutton.ibm_cf_executor()
cb.map(my_map_function, data))
```

```python
print (cb.get_result())
```

[d1.csv, d2.csv, d3.csv,...]
Unique differentiations of PyWren-IBM

- Pluggable implementation for FaaS platforms
  - IBM Cloud Functions, Apache OpenWhisk, OpenShift by Red Hat, Kubernetes
  - Supports Docker containers
- Seamless integration with Python notebooks
- Advanced input data partitioner
  - Data discovery to process large amounts of data stored in IBM Cloud Object storage, chunking of CSV files, supports user provided partition logic
- Unique functionalities
  - Map-Reduce, monitoring, retry, in-memory queues, authentication token reuse, pluggable storage backends, and many more..
What PyWren-IBM good for

- Batch processing, UDF, ETL, HPC and Monte Carlo simulations
- **Embarrassingly parallel** workload or problems - often the case where there is little or no dependency or need for communication between parallel tasks
- Subset of map-reduce flows
What PyWren-IBM requires?

Function as a Service platform

- IBM Cloud Functions, Apache OpenWhisk
- OpenShift, Kubernetes, etc.

Storage accessed from Function as a Service platform through S3 API

- IBM Cloud Object Storage
- Red Hat Ceph
PyWren-IBM and HPC
What is HPC?

- High Performance Computing
- Mostly used to solve advanced problems that may be simulations, analysis, research problems, etc.
- Does HPC well defined? – depends whom you ask
  - Super computers or highly parallel processing or both?
  - MPI (Message Passing Interface) for communication or there is only need to exchange results between simulations?
  - Data locality or “fast “access to the data?
  - Super fast? “fast” enough? Or good enough?
HPC and “super” computers

- Dedicated HPC super computers
- Designed to be super fast
- Calculations usually rely on Message Passing Interface (MPI)

**Pros**: HPC super computers

**Cons**: HPC super computers
HPC and VMs

- No need to buy expensive machines
- Frameworks to run HPC flows over VMs
- Flows usually depend on MPI, data locality
- Recent academic interest

**Pros**: Virtual Machines

**Cons**: Virtual Machines
HPC and Containers

- Good granularity, parallelism, resource allocation, etc.
- Research papers, frameworks
- Singularity / Docker containers
- **Pros: containers**
- **Cons: many focuses how to move entire application into containers, which usually require to re-design applications**
HPC and FaaS with PyWren-IBM

- FaaS is a perfect platform to scale code and applications
- Many FaaS platforms allows users to use Docker containers
- Code can contain any dependencies
- PyWren-IBM is natural fit for many HPC flows
- **Pros**: the easy move to serverless
- **Try it yourself**...
Use cases and demos..

PyWren-IBM framework
https://github.com/pywren/pywren-ibm-cloud

IBM Cloud
Object Storage

IBM Cloud
Functions
Monte Carlo and PyWren-IBM

Monte Carlo methods are a broad class of computational algorithms - evaluate the risk and uncertainty, investments in projects, popular methods in finance

PyWren is natural fit to scale Monte Carlo computations across FaaS platform

User need to write business logic and PyWren does the rest
Stock price prediction

- A mathematical approach for stock price modelling. More accurate for modelling prices over longer periods of time.
- We run Monte Carlo stock prediction over IBM Cloud Functions with PyWren-IBM.
- With PyWren-IBM total code is ~40 lines. Without PyWren-IBM running the same code requires 100s of additional lines of code.

<table>
<thead>
<tr>
<th>Number of forecasts</th>
<th>Local run (1CPU, 4 cores)</th>
<th>IBM CF</th>
<th>Total number of CF invocations</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000</td>
<td>10,000 seconds</td>
<td>~70 seconds</td>
<td>1000</td>
</tr>
</tbody>
</table>

- We run 1000 concurrent invocations, each consuming 1024MB of memory.
- Each invocation predicted a forecast of 1080 days and used 100 random samples per prediction. Totally we did 108,000,000 calculations.
Protein Folding

• Proteins are biological polymers that carry out most of the cell’s day-to-day functions.
• Protein structure leads to protein function
• Proteins are made from a linear chain of amino acids and folded into variety of 3-D shapes
• Protein folding is a complex process that is not yet completely understood
Replica exchange

- Monte Carlo simulations are popular methods to predict protein folding
- **ProtoMol** is a specially designed framework for molecular dynamics
  - [http://protomol.sourceforge.net](http://protomol.sourceforge.net)
- A highly parallel replica exchange molecular dynamics (REMD) method used to exchange Monte Carlo processes for efficient sampling
  - A series of tasks (replicas) are run in parallel at various temperatures
  - From time to time, the configurations of neighboring tasks are exchanged
- Various HPC frameworks allow running Protein Folding
  - Depends on MPI
  - VMs or dedicated HPC machines
Protein folding with PyWren-IBM

Each invocation runs ProtoMol library to run Monte Carlo simulations.

Our experiment – 99 jobs
- Each job executes many IBM CF invocations
- Each invocation runs 100 Monte Carlo steps
- Each step running 10000 Molecular Dynamic steps
- REMD exchange the results of the completed job which used as an input to the following job
- Our approach doesn’t use MPI
PyWren-IBM for batch data processing
PyWren-IBM for data processing

Face recognition experiment with PyWren-IBM over IBM Cloud

- Align faces using open source from 1000 images stored in IBM cloud object storage
- Given python code that know how to extract face from a single image
- Run from any Python notebook
Processing images without PyWren-IBM

Business Logic

- Loop over all images
- Close to 100 lines of “boiler plate” code to find the images, read and write the objects, etc.
- Data scientist needs to be familiar with s3 API
- Execution time approximately 36 minutes!
import logging
import os
import sys
import time
import shutil
import cv2
from openface.align_dlib import AlignDlib

logger = logging.getLogger(__name__)
temp_dir = '/tmp'

def preprocess_image(bucket, key, data_stream, storage_handler):
    ""
    Detect face, align and crop :param input_path. Write output to :paramoutput_path
    :param bucket: COS bucket
    :param key: COS key (object name) - may contain delimiters
    :param storage_handler: can be used to read / write data from / into COS
    ""

crop_dim = 180

# print("Process bucket {} key ".format(bucket, key))

sys.stdout.write(".
")

# key of the form /subdir1 '/../subdirN/file_name
key_components = key.split('/

file_name = key_components[len(key_components)-1]
input_path = temp_dir + '/' + file_name

if not os.path.exists(temp_dir + '/' + 'output'):
    os.makedirs(temp_dir + '/' + 'output')
output_path = temp_dir + '/' + 'output/' + file_name

with open(input_path, 'wb') as localfile:
    shutil.copyfileobj(data_stream, localfile)

exists = os.path.isfile(temp_dir + '/' + 'shape_predictor_68_face_landmarks')

if exists:
    pass;
else:
    res = storage_handler.get_object(bucket, 'lfw/model/shape_predictor_68_face_landmarks.dat', stream=True)

    with open(temp_dir + '/' + 'shape_predictor_68_face_landmarks', 'wb') as localfile:
        shutil.copyfileobj(res, localfile)

align_dlib = AlignDlib(temp_dir + '/' + 'shape_predictor_68_face_landmarks')

image = _process_image(input_path, crop_dim, align_dlib)

if image is not None:
    # print('Writing processed file: {}'.format(output_path))

    cv2.imwrite(output_path, image)
    f = open(output_path, "rb")
    processed_image_path = os.path.join('output', key)
    storage_handler.put_object(bucket, processed_image_path, f)
    os.remove(output_path)
else:
    pass;

    # print("Skipping filename: {}".format(input_path))
    os.remove(input_path)

def _process_image(filename, crop_dim, align_dlib):
    image = None
    aligned_image = None
    image = _buffer_image(filename)
    if image is not None:
        aligned_image = _align_image(image, crop_dim, align_dlib)
    else:
        raise IOError('Error buffering image: {}'.format(filename))
    return aligned_image

def _buffer_image(filename):
    logger.debug('Reading image: {}'.format(filename))

    image = cv2.imread(filename, )
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return image

def _align_image(image, crop_dim, align_dlib):
    bb = align_dlib.getLargestFaceBoundingBox(image)
    aligned = align_dlib.align(crop_dim, image, bb, landmarkIndices=AlignDlib.INNER_EYES_AND_BOTTOM_LIP)
    if aligned is not None:
        aligned = cv2.cvtColor(aligned, cv2.COLOR_BGR2RGB)
    return aligned

Business Logic

Boiler plate

• Under 3 lines of “boiler plate”!
• Data scientist does not need to use s3 API!
• Execution time is 35s
• 35 seconds as compared to 36 minutes!
Satellite batch data processing

- Horizontally Scalable
- Unified Imagery Ingestion Pipeline
- Efficient Output Query on Both Raster Data and Vector Data

Input

Application Code or Docker Image
Raw Satellite Imagery

IBM Cloud Functions With PyWren-IBM

Output

Processed Raster Data
Processed Raster Data Meta Data
Processed Vector Data
PyWren-IBM for spatial metabolomics
ALEXANDROV TEAM AT EMBL HEIDELBERG
SPATIAL METABOLOMICS ACROSS SCALES

ORGANISM

1 m
Protsyuk et al, Nature Protocols 2018
Bouslimani et al, PNAS 2017

TISSUES

1 cm
Palmer et al, Nature Methods 2017
Alexandrov et al, BioRxiv 2019

MICROBIAL PLATES

SINGLE CELLS

1 μm
Rappez et al, BioRxiv 2019

Interdisciplinary
SINGLE-CELL BIOLOGY
-OMICS
COMPUTATIONAL

Applications
INFLAMMATION
IMMUNITY
CANCER

Methods Dev
MOLECULAR
IMAGE ANALYSIS
ML, AI, BIG DATA
METASPACE use case

- Alexandrov team at EMBL develops novel computational biology tools to reveal the spatial organization of metabolic processes [https://www.embl.de/research/units/scb/alexandrov/](https://www.embl.de/research/units/scb/alexandrov/)

- "Imaging mass spectrometry" application
  - Uses Apache Spark and deployed across VMs in the cloud

Customer uploads medical image → Customer choose molecular databases → Data pre-processing and segmentation → Molecular scan of datasets with dedicated algorithms → Generation output images with results
Metabolomics with PyWren-IBM

Metabolomics application with PyWren-IBM
• We use PyWren-IBM to provide a prototype that deploys Metabolite annotation engine as a serverless actions in the IBM Cloud
  • https://github.com/metaspace2020/pywren-annotation-pipeline

Benefit of PyWren-IBM
• Better control of data partitions
• Speed of deployment, no need VMs
• Elasticity and automatic scale
• And many more...
Behind the scenes

- **Dataset Input**: up to 50GB binary file
- **Molecular Databases**: up to 100M molecular strings

Results

Molecular annotation engine

Image processing

Metabolite annotation engine deployed by PyWren-IBM
A whole-body section of a mouse model showing localization of glutamate. Glutamate is a well-known neurotransmitter abundant in the brain. It however is linked to cancer where it supports proliferation and growth of cancer cells. Both facts are supported by the detected localization, obtained using METASPACE. Data provided by Genentech.
Summary

PyWren-IBM is a novel framework with advanced capabilities to run user code in the cloud.

We demonstrated the benefit of PyWren-IBM for HPC, molecular biology and batch data pre-processing.

For more use case and examples visit our project page: https://github.com/pywren/pywren-ibm-cloud

All is open source

Thank you

gilv@il.ibm.com