The Role of Python in Cloud Computing (October 2014)

Author: Adam Lloyd, R00117318, DWEB2, Cloud computing with Python COMP8038

Abstract—**From humble beginnings, Python has become one of the most popular languages for a varied number of applications in the development industry. This paper will look at cloud technologies, frameworks, data analysis and other factors for Python's emergence as a market leader with a specific focus on its applications in the cloud**.

***Index Terms*— Python, Cloud, Computing.**

I. Introduction

IN THIS PAPER THE ROLE OF PYTHON IN CLOUD COMPUTING IS EXPLORED AND THE POINT AT WHICH PYTHON AND CLOUD COMPUTING INTERSECT WILL BE IDENTIFIED. THE REASON THAT PYTHON IS USED BY COMPANIES SUCH AS REDDIT AND GOOGLE FOR THEIR WEB APPLICATIONS IS EXPLORED BUT PRIOR TO THAT, THE REASON FOR PYTHON'S GROWING POPULARITY AS A PROGRAMMING LANGUAGE IN GENERAL MUST BE EXPLAINED. HOW DID GUIDO VAN ROSSUM'S "HOBBY" PROJECT, NAMED FOR HIS LOVE OF MONTY PYTHON'S FLYING CIRCUS [1], GROW UP TO RIVAL LANGUAGES SUCH AS JAVA AND C AND BE ADOPTED BY INDUSTRY GIANTS?

II. What is Python?

Before the applications of Python in the cloud are explore the benefits of the Python programming language as a whole must first be identified.

To quote the Python.org website - "Python is a programming language that lets you work quickly and integrate systems more effectively." [2] This statement will ultimately define the conclusions make in this paper and Python's growing popularity.

Python is a solid and powerful high level interpreted language that is dynamically typed with an emphasis on code readability via its indentation functionality. Using indentation as a feature rather than a convention makes Python code more readable and reduces the code needed to implement it[3]. Good practice would already be to indent the code so Python takes that and cuts out the middle man so to speak.

While it may not be as "fast" as Java or C++, the overheads are reduced with the minimal syntax reducing the amount of code developers have to write and Python programs can be up to 5 times smaller than an equivalent program written in Java [2] which makes the program easier to maintain and debug ultimately saving developers time and money.

Python is very portable and is available for all major operating systems. The same source code will run on a Microsoft machine and a Nokia Series 60 cell phone. [4]

Python is referred to as having a "batteries included" philosophy[25] meaning that it comes with a robust standard library which covers everything from asynchronous processing to .zip files and building web servers with 3 lines of code [4]
If there is something that its standard library cannot do then the Python Package Index (PYPI) [5] contains more than 50,000 packages at the time of writing which will cover nearly any need a developer may have. There are also many cloud specific libraries, some of which will be explored later.

If there is something that you need Python to do that it can't but another language can then you can "wrap" lower level code with SWIG(Simplified Wrapper and Interface Generator) and it will run as native Python code. [4]

Finally Python is and open source which means only good things for security, support due to the large community developing new features and writing documentation and perhaps most importantly, cost.

Once again, the old adage "time is money" is very much applicable especially in the IT sector and Python's features make it an excellent choice for those working on the cutting edge of software development. So much so that the benefits of Python are perceived to be so great to Google that they have adopted it as their sole programming language.

This paper was produced as an assignment for the Cloud-Computing with Python module in Cork Institute of Technology provided by Aisling O'Driscoll.
Author: Adam Lloyd, R00117318, Cork Institute of Technology. DWEB2 – Bsc (Hons) Web Development year 2.

III. Cloud computing

When developing for the cloud the infrastructural needs of an application will likely not be known for certain until the application is deployed. It will not be clear if the application is going to be the next Facebook or get as much traffic as your neighbours blog about his cat. There are many flexible cloud computing options that enable the developer to pay for only what the application needs and uses rather than laying out a physical infrastructure prior to this knowledge.

Three main services are available in the cloud, Software of a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The decision of which service to choose will depend on the needs of your application or service. This paper will examine the advantages and disadvantages of each of the services and where Python is used within them.

IV. Cloud technologies

A. SaaS

Starting at the highest level of abstraction from the physical metal of a deployment solution is SaaS which provides a point of access for customers, usually through web browsers, to manage applications running on servers. There is no need for management of software or hardware only a focus on the application itself. The disadvantage of this service is that there is little scope for customisation of the platform it is running on and the stability of the application is at the mercy of the stability of the PaaS it is running on.

Since SaaS is mostly used by end users and not for application development this paper will not explore the applications of it in detail, however it is notable that Python's libsaas library provides easier interface with SaaS APIs(Application Program Interface) [6] by providing methods to authenticate and construct URLs which pass back Python objects.

B. PaaS

Going down a level from SaaS into development territory is PaaS which provides an environment in which software can be developed and deployed quickly and easily without having to deal with the infrastructure that it is running on. The service provider will manage virtualisation, servers, storage and networking leaving developers free to focus on the application itself. With a service like Heroku for example, a web application can be deployed as easily as typing "heroku create" then pushing your code to a remote repository using a "git push" command [7].

PaaS services usually offer some form of free tier usage which makes them a very popular option for personal projects and for prototyping applications quickly before commercial deployment on an IaaS platform. That is not to say that PaaS is merely an amateur solution as a PaaS can be scaled up as needed to provide all that is needed for a more robust commercial web application.

Examples of PaaS providers that support Python include Heroku and Google App Engine.

C. IaaS

IaaS or "elastic computing" is another level down and is a service that provides a virtual or physical infrastructure including storage, networking and servers which can be scaled easily without the need to store and maintain the equipment. Compared to PaaS, IaaS users gain more control over their development platform and more scalability. Traditional service models operate on a pay as you go system where you only pay for what you need or use.

It might seem like choosing IaaS over PaaS(SaaS) is counter productive to making things easier and faster but for many projects the customisation provided by an IaaS solution is a necessity and luckily Python is here to help. With the Boto and Apache LibCloud libraries Python can be used to automate and manage most popular cloud services by integrating the cloud service APIs into Python scripts. The decision of which library to choose though depends again on the project at hand and the chosen infrastructure solutions.

Boto focuses on Amazon AWS by providing excellent support for the majority of Amazon's cloud services and having the documentation to back it up. [8] Apache LibCloud's focus is more of a jack of all trades in that it supports many different cloud service providers alongside Amazon AWS including Rackspace and OpenStack [9]. If you project is largely based around an Amazon AWS infrastructure then clearly Boto is the library of choice but if integration between multiple service providers is required then LibCloud is the prudent choice.

Whichever option is decided upon the both provide simple API method calls to quickly connect to and manage cloud instances so even when a project needs the extra control provided by an IaaS solution Python is there to make the process smoother.

Examples of IaaS providers include: Amazon Web Service (AWS), and Google Compute Engine (GCE).

V. Heroku vs Google App Engine vs Amazon EC2

Amazon EC2 and GAE/Heroku cannot be directly compared as market competitors as one is an IaaS and the others PaaS but the solutions they provide could easily be compared for the purposes of application deployment and development. It is with this that this paper will explore these three options as popular choices.

Both Heroku and Google App Engine (GAE) are quite similar PaaS solutions and provide users with a ready made environment to quickly deploy applications provided that their technology is supported by the PaaS. Amazon EC2, being an IaaS platform provides much more flexibility in that the platform can be customised to fit an application instead of developing an application to fit Heroku or GAE. Cost wise, Amazon EC2s free tier usage only applies for the first year after account creation and includes 750 hours of EC2 usage per month and 5GB of S3 storage [10].

Notable EC2 users include Pinterest and Netflix [10].

GAE provides everything that is needed to get an application off the group in their SDK and it is relatively simple to do so however, GAE runs on Google's own cloud servers however can only run modules which are written in pure Python so the aforementioned “wrapping” of other language code is not possible. GAE also requires that APIs be written exclusively for the AppEngine so moving applications away from GAE will require substantial code refactoring [11]. GAE uses Google Bigtable for databases which allows for good scalability but restricts data export and interactions [12]. GAE also provides access to many Google services such as account integration and image manipulation. GAEs free tier usage is based on usage and has limits including 1GB of traffic per day and 5GB of cloud storage [13].

Notable GAE users include BestBuy and CocaCola [13].

Heroku is hosted on Amazons EC2 servers. It does not restrict “wrapped” code the way GAE does so allows for more freedom with Python. It uses standard SQL databases which allow for much better exporting of data than GAEs Bigtable and has a lot of extensions to make deploying applications much easier, you can even push code to Heroku via GIT. Heroku's server is called a dyno and your first dyno is totally free with more dynos costing an hourly sum [7]. To store data when using Heroku an external service like Amazon's S3 is needed so additional costs can be incurred here. Heroku's pricing model is also simpler than GAE once you are beyond the free Tier requirements so that is something to consider when looking to scale your application.

Notable Heroku users include Toyota and Macy's[14].

In conclusion, EC2 offers less in the way off free tier usage then GAE and Heroku but offers more control over the platform the application will be running on at the cost of having to deploy it yourself and administer the infrastructure that it is running on. Heroku and GAE provide a much simpler deployment process but require the application to be designed somewhat specifically for their systems, more so with GAE.

For the deployment and development of small to medium applications the PaaS solutions appear the more prudent choice giving more time to focus on the development. Amazon's EC2 would be a good choice to deploy a larger application due to the relative ease of scaling the infrastructure and greater control of platform configuration.

Once again Python's strengths make it a perfect fit, making things easier, be in by automating management of EC2 instances with Boto or LibCloud or providing integration with the various PaaS services.

VI. Web Frameworks

Something to consider before looking at the Web frameworks that use Python is that Python itself is a framework [3] in that it provides a layer of abstraction from generic functions and syntactical obligations allowing the developer to focus on the application and not the low level details. For web development it goes a step further by installing with its own basic web server for testing.

When it comes to writing a web application, choosing something like PHP means your application will be written in PHP but with Python's varied web frameworks, while will writing Python, you can be abstracted further again using the API of the more full featured frameworks. If it is taken that Python is itself a framework in that it is a collection of modules and libraries that make development easier then the concept of a web framework is just that, a collection of packages or modules which allows developers to write and deploy applications faster and easier [15].

Frameworks can be "Full Stack" which include provide built in functionality for some commonly required functionality such as user authentication and handling sessions like Django, “Micro frameworks” such as Flask which are simple but highly extendible or they can be basic which just provide the application server to run the application (CherryPy) [15]. The choice of framework type is dictated by preference and requirements.

Python's role in these frameworks can differ in that some allow for more traditional Python code to be written (Flask) while others focus on out of the box functionality and even higher levels of abstraction using their own API and templating language to make development even easier once the it is learned [16]. Whatever option is chosen there are plenty of modules that can be added to implement any functionality you need and due to its open source nature, Python has numerous free libraries for functions like database interaction and templating systems(Jina2, Mako) to make the development process even smoother.

Given all these benefits it makes sense for a developer building a large web application to use Python with a web framework over traditional development methods such as PHP as the time to develop and application can be reduced significantly, there will be less code and it will be easier to maintain. Proof of this can be seen by looking at current trends in web development with giants such as Google, Reddit and Yahoo all using Python as their base programming language [17].

To compare all Python web frameworks would be a subject for a paper all of its own so this this paper will compare two popular frameworks. Django, the largest and most popular framework and Flask, a minimal micro framework.

VII. Django vs Flask

Django is the most popular framework at present due to it's robust out of the box functionality making minor development tasks easy and fast. However this comes at a cost in that learning Django's API and templating language has a steep learning curve [16] and difficulty arising if the developer wishes to modify one of the features as they are all so well integrated. Once of Djangos philosophies is DRY(Don't Repeat Yourself) which refers to the ability to modularise code and remove the need for it to be repeated across a web application [18] which further adds to its efficiency and practicality for use with large applications.

A micro framework like Flask on the other hand starts off very small and basic offering a development server and a templating system out of the box. If a function is required then it can easily be extended to add complexity as development progresses using libraries from the PyPi[5] repository. The need to learn a framework specific API is removed in favor of more traditional and cleaner Python code [19].

Ultimately Flask can do anything that Django can with a bit of work and a few extensions but for a large company developing large web applications Django is understandably the most popular choice as it saves time and money once the learning process is complete. For a developing quick prototypes or making a standalone application Flask may be the framework of choice [19].

Python's role in framework based web development lies, once again, in its strengths. The speed of development, the ease of code maintenance and the extendability provided by its many libraries all add value and same time. The ability to create modularise and reuse code removes redundancy and lowers overheads and Pythons in built web server makes it ideal for the purpose.

VIII. Big Data Analytics

Big data is the term used for extremely large and complex data sets and while Python has been a popular choice for scientific computing and data analysis for many years due to it's ease of use and powerful libraries (SciPy, Pandas) [2] most Big Data analysis has traditionally been done with Java or similar languages. An example of which is Apache Hadoop which provides fast, reliable data analysis and is able to handle large data sets so has become synonymous with big data analytics. Hadoop traditionally requires a very high level of Java programming to program which compounds the already daunting task of analysing such complex data.

A more flexible language like Python would make the task simpler and, using Python's powerful libraries and in built dictionary features it may even be more effective [21]. However, in order to simplify the task and use Python in place of Java, Jython [22], which allows Python source code to be compiled as Java to run on a Java Virtual Machine(JVM) was used. In keeping with Python's open source and well supported nature there are now many frameworks and libraries like dumbo and mrjob [23] to do the job as well providing more convenient options.

With cutting edge analysis methods being used for researching and predicting even pre-natal health issues [24] saving time is more important than ever and Python's powerful dictionaries, list manipulation and extensive external libraries make it a solid choice when faced with such daunting tasks.

IX. Conclusion

At the beginning of this paper it was quoted that "Python is a programming language that lets you work quickly and integrate systems more effectively" and the applications for this have been found to be numerous. In an everyday programming environment easily maintainable Python scripts can automate regular tasks saving time and allowing more focus on larger tasks. In the cloud Python is used to make interacting with a chosen infrastructure simpler with Boto and LibCloud, deployment of a web application easier with frameworks like Django and Flask, and analyse Big Data with libraries to integrate with Hadoop.

This paper then set out to identify the point where Python and the cloud intersect. It has been shown that the point where Python is chosen due to it's strengths is this point. When complex tasks and systems must be developed and integrated quickly with a focus more on the core of the project than the extraneous tasks surrounding its development Python is chosen to take advantage of is power.

Time is money and Python saves time. Anywhere the developer can be abstracted from the more mundane aspects of development Python fills in the gap. It is easy to learn and maintain making for more saved time in the future and the future is where Python's place is, gaining more traction in the industry as the years go on, becoming more and more supported and documented and making projects easier and more cost effective wherever it is applied.

X. References

[1] (Accessed 27 Oct 2014) History of Python. [Online]. http://www.python-course.eu/python3_history_and_philosophy.php

[2] (Accessed 23 Oct 2014) Python.org. [Online]. https://www.python.org/

[3] Mike Levin, "Python Programming Language Advantages," http://mikelev.in/2011/01/python-programming-language-advantages/, Accessed Oct 23 2014.

[4] Darryl K. Taft. (Accessed Oct 24 2014) eWeek. [Online] http://www.eweek.com/c/a/Cloud-Computing/15-Ways-Python-Is-a-Powerful-Force-on-the-Web-275427/

[5] (Accessed Oct 23 2014) Python.org. [Online]. https://pypi.python.org/pypi

[6] (Accessed Oct 27 2014) Libsaas. [Online]. http://libsaas.net/

[7] (Accessed Oct 25 2014) Heroku Dev Center. [Online]. heroku.com

[8] (Accessed Oct 24 2014) Boto. [Online]. http://boto.readthedocs.org

[9] (Accessed Oct 24 2014) Apache LibCloud. [Online]. https://libcloud.readthedocs.org/en/latest/supported_providers.html

[10] (Accessed Oct 23 2014) Amazon Web Services. [Online]. http://aws.amazon.com/free/

[11] (Accessed Oct 25 2014) ScriptRock. [Online]. http://www.scriptrock.com/articles/heroku-appengine

[12] (Accessed Oct 25 2014) Echo One. [Online]. http://rrees.me/2009/10/08/heroku-versus-gae-gaej/

[13] (Accessed Oct 24 2014) Google Cloud Platform. [Online]. https://cloud.google.com/pricing/

[14] (Accessed Oct 27 2014) Heroku. [Online]. https://www.heroku.com/customers

[15] (Accessed Oct 25 2014) wiki.Python.org. [Online]. https://wiki.python.org/moin/WebFrameworks

[16] (Accessed Oct 27 2014) InfoWorld - Pillars of Python. [Online]. http://www.infoworld.com/article/2622836/application-development/pillars-of-python--six-python-web-frameworks-compared.html?page=2

[17] (Accessed Oct 27 2014) Line of Thought. [Online]. http://lineofthought.com/tools/python

[18] (Accessed Oct 25 2014) Django. [Online]. https://docs.djangoproject.com/en/dev/misc/design-philosophies

[19] (Accessed Oct 24 2014) Six Feet Up. [Online]. ttp://www.sixfeetup.com/blog/4-python-web-frameworks-compared

[20] (Accessed Oct 27 2014) Hadoop. [Online]. http://hadoop.apache.org/

[21] (Accessed Oct 27 2014) Continuum. [Online]. http://continuum.io/python-for-big-data

[22] (Accessed Oct 27 2014) Sourceforge. [Online]. http://sourceforge.net/projects/jython/

[23] (Accessed Oct 27 2014) All Things Hadoop. [Online]. http://allthingshadoop.com/category/python/

[24] (Accessed Oct 26 2014) ClouDx-i. [Online]. http://www.cloudxi.eu/

[25] (Accessed Oct 26 2014) Python.org [Online] https://docs.python.org/2/tutorial/stdlib.htm