Data and ML model management with PyTorch and AWS



This white paper provides an overview of production scale implementation of a data analytics solutions, particularly focussing on the various technologies offered by the Amazon Web Services (AWS). The white paper also explores the PyTorch framework for running machine learning models at large scale harnessing the various resources provided by AWS.

Sudeep Kurian

**Table of Contents**

* Introduction 3
* Production Scale Environments 5
* Managed Services 7
* Storage Resources 8
* Compute Resources 10
* Running Machine Learning

at Production Scale 14

* Conclusion 21

**Introduction**

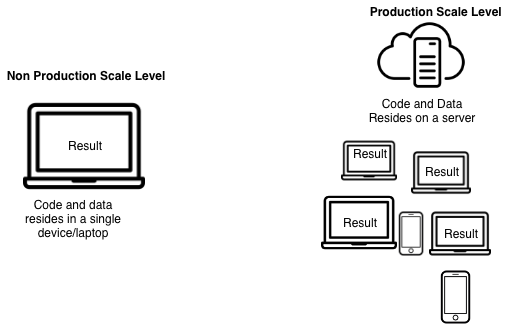
Analytics is a field encapsulating Data Science which looks at, processes, and gives inferences from the crude data gave to it. As for data inference, it very well may be isolated into Data Analytics and Business Analytics. While Business Analytics makes us think from a business perspective, Data Analytics requests that we give bits of knowledge absolutely from the data.

**Production Scale**

**Production** refers to software/systems that can control interactions with users and can also involve multiple users. It can integrate with other software and systems and can be run somewhere other than personal laptops. Further, these systems are automated and tested. They are validated and strictly controlled while being updated.

**Scale** on the other hand refers to the ability of scaling in terms of different aspects based on the requirement. It involves scaling size of data (up or down, based on requirement), scaling number of users, scaling computational abilities and scaling regulations and policies.

**Non-production scale analytics** refers to the model building and development stage that is often conducted on personal computers and laptops with not much scope for scaling. It is just like the code analysts write on their personal Jupyter notebooks or R consoles. Once a model or code is built successfully on a non-production scale level, a proper framework is required to scale it up to a production scale level.



**Machine Learning at production scale**

**Machine learning** is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.[1]

Machine learning is no longer a buzzword in the world of data analytics. Machine learning applications have become so ubiquitous that we often ignore the fact that these applications are intelligent codes which are being implemented at such a large scale.

Hence machine learning code is an important part of any analytics pipeline and in order to maintain this code at production scale level, a developer needs to implement a proper framework for optimal resource allocation.

The following diagram shows the process of training a basic machine learning model:



This is a non-production scale approach of training and testing a model. One might face issues while scaling this model in a large scale. Training a model requires all the data together. Meanwhile prediction doesn’t necessarily require a lot of data points together. Also, one might not need to train data repeatedly but it’s possible that we might need to predict something at a very fast rate (like 1000 times per second). Also, training the model requires a lot of time if the dataset is huge. Live data makes it even more difficult for updating the model at large scale.

In order to scale such a machine learning model in a data analytics pipeline one needs to ensure the following:

* Resources (Storage and computation) should be allocated according to the application for less operational cost
* Historic data needs to be updated in a fixed time interval and new models should be trained as data keeps updating
* While a new model is being trained, the end user should not face a stoppage of service
* Code should be implemented efficiently for saving time, as these resources might be charged according to time

This is what motivates us to implement a machine learning algorithm using services like AWS which provides us with such computing resources and facilitates scalability of code. Frameworks like PyTorch and TensorFlow ensure that the code we write for such purposes is time efficient and hence production scale ready.

**Production Scale Environments**

It is not possible to run one’s analytics code on a personal laptop as it is not possible to scale easily in terms of number of users, size of data,  scaling regulations or even computational abilities. Some of the key requirements for a production scale environment are listed below:

* **Need to run scalable data science:** Current scripts and models can easily run when the data behind the model grows multiple times. As data grows the storage allocated should also grow accordingly
* **Cost Effective:** Buying a new machine for every big request or requirement can be prohibitive. A cloud based environment would allow renting out a higher configuration for a few hours or days and thus have solution to the problem at a fraction of the cost
* **Collaboration:** Allows several users to work off the same data without creating copies on individual machines
* **Sharing:** Makes the code transferable across users, irrespective of libraries used in the code
* **Easier Deployment:** Larger ecosystem for machine learning system deployments
* **Flexibility:** Using servers or the cloud as the production environment allows for real-time scaling up or down based on requirement

Servers or cloud is often used as production scale environment. There are many cloud based services available in today’s market which include:

* Amazon Web Services (AWS)
* Microsoft Azure
* Google Cloud Platform
* IBM Cloud - IaaS

Services offered by AWS

AWS provides tools to compute, develop, store data, manage tools and maintain database. Further, it allows migration, networking and content delivery, media services, machine learning, analytics, customer engagement, business productivity, security, identity and compliance, mobile services, internet of things, application integration, AWS cost management and much more.

AWS makes a good choice for the production scale data environment for a variety of reasons. Storage, computation and security are the main AWS services that are used for production scale data analytics. Apart from this, AWS offers its services in a plethora of domains.

Following are a few reasons why AWS is chosen over other similar cloud based services:

1. **Plethora of options and services:** AWS offers around 100 cloud-based services for its enterprise customers, which is by far the largest via any other competitor.
2. **Security and Flexibility:** AWS gives you the functionality of Amazon Machine Images (AMIs), which enable us to spin-up clones in multiple regions and environments, making it hassle-free to setup the entire process again and again. It also has a robust, world-class infrastructure for the security of its data centres as well as over the internet. Moreover AWS offers a very secure framework for users and ensures that sensitive data resides safely in it’s environment
3. **Cost effective:** Since AWS shifted from its by-the-hour pricing to by-the-second pricing, its pricing came in line with Azure and GCP and became one of the most competitively priced cloud solutions in the industry. Moreover, AWS is also becoming a huge attraction for the smaller industries, since it does not ask for any price commitments, not even for a month. One has to pay only for the time it’s using the Amazon services and can decide to discontinue anytime.

Following are some of the domains where AWS offers its services:

* Compute resources
* Storage resources
* Database resources
* Migration
* Networking and content delivery
* Developer tools
* Analytics
* Machine learning tools
* AR/VR
* IoT applications
* Security Identity and Compliance
* Game Development

And many more

There is an increased user base who have migrated from traditional servers and are receiving services offered by Amazon, there is incredible user support for troubleshooting issues faced. A lot of users help each other on sites like StackOverflow and Quora to help each other on various issues they might face.

**Managed Services**

Service providers like AWS provide their users a multitude of options for scaling up to production level. For example in order to run a database instance a user has the following two alternatives in an AWS environment:

* **Non-Managed solution/ ”Roll your Own”:**
  1. Spin up EC2 (compute) instances
  2. Attach EBS(storage)
  3. Installing required software/ packages
  4. Run Database software (like Postgres, MySQL etc ) on EC2 instances
* **Managed service:** Spin up a Database “cluster” on a database services provider like AWS where packages and software are preinstalled

A managed services is referred to as the services provided by an information technology (IT) services provider that manages and assumes responsibility for providing a defined set of services to its clients. These services are less flexible and a little bit expensive as compared to setting up our own database cluster, which is a non-managed service.

An example of a managed service can be AWS, where it provides us with the entire Database cluster without any hassle of setting up the instances or the storage options by ourselves.

Advantages of using a managed service are:

* Low set up time – configuration or set up work is low
* Low maintenance requirements
* High levels of stability as offered by amazon
* Responsibility for active service lies with the service provider
* Built in methods of scalability

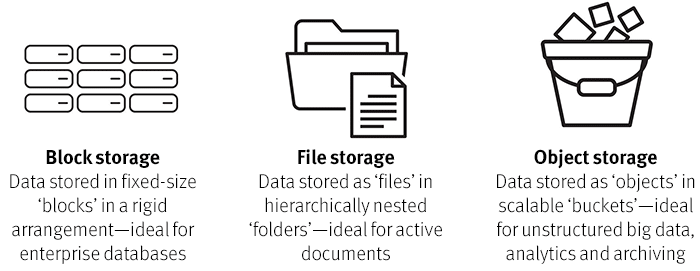
Some Disadvantages of using a managed service are:

* Increased costs as compared to “roll your own” solutions
* Reduced customization/flexibility
* Vendor Lock-in (It is difficult to transition from one vendor to another)

Storage Resources

In order to run a data analytics task on production scale, we need a few storage resources. There is a diversity of storage resources available which include:

* **Block Storage:** Block storage is data storage typically used in storage-area network (SAN) environments where data is stored in volumes, also referred to as blocks. Each block is assigned an arbitrary identifier by which it can be stored and retrieved, but no metadata providing further context. Database storage is a common use for block storage.[2]
* **File Storage:** File storage, also called *file-level* or *file-based storage*, stores data in a hierarchical structure. The data is saved in [files](https://whatis.techtarget.com/definition/file) and [folders](https://whatis.techtarget.com/definition/folder), and presented to both the system storing it and the system retrieving it in the same format.[3]
* **Object Storage:** Object storage, also called *object-based storage*, is an approach to addressing and manipulating data storage as discrete units, called *objects*. Objects are kept inside a single repository, and are not nested as files inside a folder inside other folders.[4]



Source: <https://www.emc.com/storage/elastic-cloud-storage/articles/what-is-object-storage-cloud-ecs.htm>

Most of the cloud services provide all the 3 types of storage options mentioned above, however, the most preferred one is ‘Object Storage’ due to its low-cost, simplicity, flexibility, and scalability.

Other than the cloud storage options, we also have what we call the ‘traditional, on-premise’ IT infrastructure data storage choices. A few of them include -

* **Memory:** It includes storage, such as cache, in-memory databases. It provides one of the fastest ways to access the data.
* **Databases:** Includes structured and unstructured data stored in traditional SQL Relational Databases and NoSQL Non-Relational Databases respectively.
* **Network attached storage (NAS):** NAS provides a file-level interface to storage which can be shared over many systems. It is a little slower as compared to other storage options
* **Direct-attached storage (DAS):** It refers to the local storage at each server. It provides incredible speed
* **Storage Area Network (SAN):** This is a block-based storage which has a high speed architecture, though lower than DAS, but increased durability for database storage.

AWS offers a multitude of storage resources with each one of them having unique combination of performance, durability, cost, and interface. A few of the choices are mentioned below -

* **Amazon Simple Storage Service (S3):** This is one of the scalable and common storage options provided by AWS. It provides high durability storage option for its users.

**Advantages:**

* Durable and easy to manage
* Easy scalability
* Less expensive
* Secure service with variety of encryption services available

**Disadvantages:**

* The organization is similar to file system, however, there exists only a flat hierarchy, unlike typical file systems
* It does not version the data, like GitHub or such software. This has to be done manually which might be a task if the data is huge and needs to be versioned and tracked.
* It allows everyone to access what is present in one object storage, it doesn’t put restrictions in the files in one object storage which might be a requirement.
* We can provide read/write access to individual buckets but it is difficult to enforce such standards on particular files within a bucket
* **Amazon Glacier:** Amazon Glacier is an extremely low-cost storage service that provides highly secure, durable, and flexible storage for data archiving and online backup. Amazon Glacier is a low-cost storage service designed to store data that is infrequently accessed and long-lived. Amazon Glacier retrieval jobs typically complete in 3 to 5 hours.

**Advantages:**

* Low cost
* Scalability and elasticity
* High durability and availability

**Disadvantages:**

* Rapidly Changing data might be better served with a different storage solution like Amazon EBS
* Data stored in Amazon Glacier is not available in real time

In the data analytics pipeline, **Amazon S3** can be used as an object storage option, which often lies right at the start of the pipeline. We store the input data in S3 and take it out for pre-processing and running it through the Machine Learning Model to get the output.

On the other side, **Amazon Relational Database Service (RDS)** makes it easy to store the data in a relational database in a cloud. This is often the data which is outputted after travelling the entire analytics pipeline and has to be used in some other User Interface, say on a web page. Hence, this oftentimes fits into the last part of the analytics pipeline to store the output data, just before rendering it on some user interface.

**Storage solutions for data analytics**

There is a huge variety of storage resources offered by AWS. Of those, a few solutions are highly relevant to data analytics and considered to be apt for carrying out complex data computation and analytics work. A few of them are mentioned below along with a situation as when to use such storage resources-

* **Amazon S3 :** Amazon’s S3 is an object storage system. It is really inexpensive and can be used to store unstructured data (images, videos etc). On data analytic application can be storing images for an image detection application.
* **Amazon Redshift:**  Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. Redshift is used for complicated queries. These queries are not run as frequently. Redshift can handle computationally expensive querying
* **Amazon Relational Database Service(RDS):**  It is a web service that provides the capabilities of various SQL vendors like MySQL,Oracle etc as a managed cloud based service. It is optimal for optimal for new application with structured data that requires sophisticated querying and joining capabilities. As compared to Redshift, Amazon RDS is meant to be the primary database

We would use RDS as opposed to Redshift in the following cases :

* If we want to run smaller queries as opposed to large queries.
* If we want to run these small queries at very high frequency rather than large queries at rather lesser frequencies.
* RDS is meant to be your primary database, i.e. your transactional database, which is also optimized to run not so computationally expensive queries, as opposed to Redshift, which does run computationally expensive analytics queries.
* For example, web applications that require smaller queries but at high frequency can be run on RDS as opposed to Redshift.

Compute Resources

Compute resources are resources that provide processing capabilities in the cloud. CPU and memory are collectively referred to as *compute resources*, or just *resources*.Compute resources are measurable quantities that can be requested, allocated, and consumed.

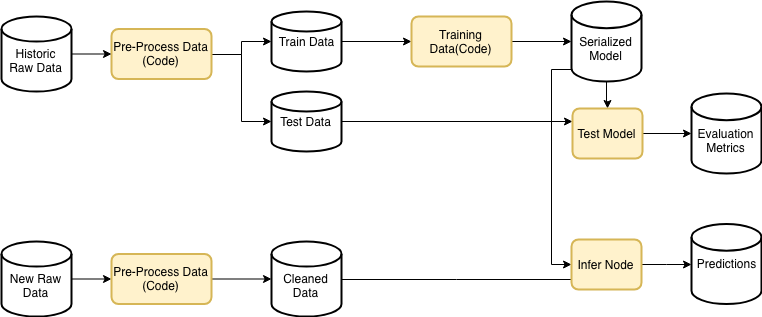
**Types of computing resources:**

* Virtual clusters
* Virtual resource pools
* Physical servers

In order to perform any data analytics task, we need processing and storage capabilities. These capabilities are provided by the compute resources we allocate for the task.  Hence it is essential to have compute resources for performing a data analytics task.

**Need for compute resources**

The following chart explains a basic data analytics pipeline at production scale:



Data analytics pipeline

In the above flow diagram, the Pre-Processing , training the model, testing our model and inference node all require some computational and processing resource. These parts of a data analysis pipeline require compute resources.

**Compute Resources offered by AWS**

AWS offers various choices for compute resources to develop, deploy, run and scale applications. Following are the options available:

**Amazon EC2:** Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. It should be used if the end purpose is to run any application, controlling and managing server or cluster level functions such as scaling and deployment.

**Advantages:**

* Amazon EC2 enables you to increase or decrease capacity within minutes. One can commission one, hundreds, or even thousands of server instances simultaneously.
* Complete control of your instances including root access and the ability to interact with them as you would any machine i.e. high flexibility
* Multiple geographic areas to run servers with a standard interface.
* Programmatic/API access to do everything.
* Multiple availability zones in each region for availability and capacity planning.
* Entire AWS ecosystem of services and support and community to build on

**Disadvantages:**

* As security is one of the main features so AWS limits some of its features which cannot be changed at all are-

o   **EC-2 classic-** Maximum of 500 per instance and each Security Group can have a maximum of 100 permissions

o   **EC2-VPC-** Up to 100 security groups per VPC(virtual private cloud)

* Instance types are rigid, must get entirely bigger instances even if just interested in more CPU or RAM
* Expensive at on-demand rates if elasticity is not needed (or expensive upfront payment if not using server for entire purchase length)
* VM performance can be highly variable due to size, placement, and other uses on the same physical hardware

**Amazon Lightsail:** Amazon Lightsail is an Amazon cloud service that offers bundles of cloud compute power and memory for new or less experienced cloud users. AWS designed the Lightsail service to make it simpler to understand and purchase rudimentary compute capacity. AWS also manages the infrastructure, which shares the same uptime and global regions and availability zones as EC2, and makes it available with a few mouse clicks.

**Advantages:**

* Billed a flat predictable rate for a plan each month
* User doesn’t have to worry about security groups or other settings.
* Easier for the user to follow  through the intuitive Lightsail console

**Disadvantages:**

* Not flexible
* More expensive price packages
* Fixed size memory allocation

**Amazon ECS:** Amazon Elastic Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure. It is used to run stateless or stateful applications packaged as Docker containers. Common use cases include, Web applications, batch jobs, docker workloads and micro applications.

**Advantages:**

* AWS manages the fault tolerance of the application.
* AWS manages cluster state and container deployment.
* High Flexibility (lets you choose resources and scale it to the most granular level)
* Relatively easy to switch to a different platform ( Easier vendor migration)
* Load balance -ECS helps you in distributing the traffic across all your containers
* Security-ECS permits the instances to have a minimal role and as well allows you to manage the task role and instance role separately

**Disadvantages:**

* A little complicated to set up

**AWS Lambda:** AWS Lambda is used to  run event-initiated, stateless applications that need quick response times. It allows one to run code without provisioning or managing servers.  It is a computing service that runs code in response to events and automatically manages the computing resources required by that code.

**Advantages:**

* Reduced cost of execution- only pay for the computing costs that your code makes use of
* Improved Application Resiliency
* Reduced risk of not having to rely upon a single machine to perform all the tasks of serving your app

**Disadvantages:**

* AWS Lambda functions are timeboxed, with a default timeout of three seconds (it is configurable up to five minutes). This means you need to spend more time orchestrating and organizing your functions, so that they can work in a distributed fashion on your data.
* One is not able to custom install packages or software on the running environment. If the code has extra OS needs, it may need to be adapted – or hosted with another service entirely

**Where do these fit in the data analytics pipeline?**

Compute resources like EC2 can be used for pre-processing our data. Machine Learning/Artificial Intelligence Model is like a function in a production scale analytics workflow/pipeline. It takes an input like some features for the model, which in-turn comes from an object store or some storage solution. In order to do this processing (run the function) we need some compute resources(like EC2) and some managed solutions which cater to our specific data analytics/machine learning needs (like AMIs, Sagemaker etc).

We can also use these compute resources to test our models. Different type of resources can fit best for different types of roles in the pipeline.

Compute resources managed by AWS include:

* Deep Learning AMI: An instance is generated with all of Amazon Managed Images (AMI)
* Sagemaker:
  1. Has AMIs under the hood
  2. It spins up an EC2 and opens up a Jupyter instance automatically
  3. It can spin up an instance of PyTorch thus reducing the time and effort spent in installations and optimizations

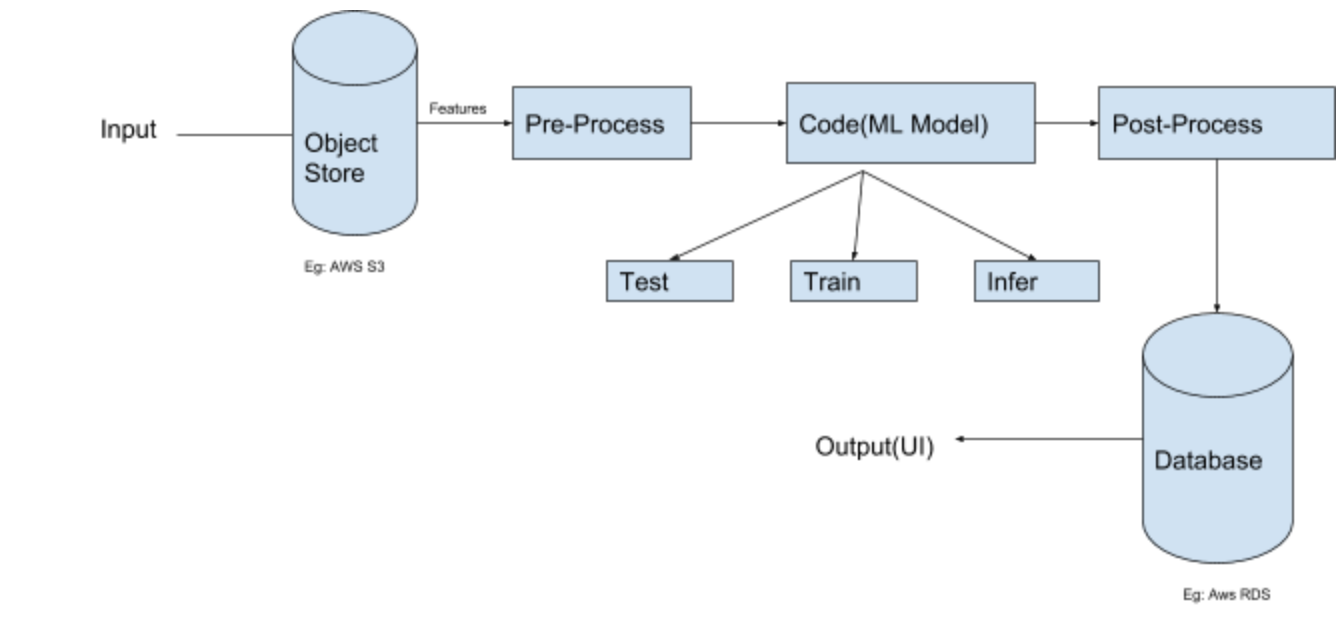
Deep learning AMIs are convenient when the task involves scaling up an existing model / code to production level.

Sagemaker, on the other hand would be a better option when a model needs to be built or modified as it provides access to instances of PyTorch and Jupyter notebook. Amazon Sagemaker only requires a docker container with the users preferred framework and libraries - such as Caffe2, PyTorch, Microsoft Cognitive Toolkit (CNTK), Chainer, or Torch and it will manage the underlying infrastructure to train models.



|  |  |
| --- | --- |
| Resource | Where does it fit the pipeline |
| EC2 | When you spin up your own pipeline by scratch, the ML codes are supposed to run on different EC2 instance |
| Amazon S3 object Storage | It is used to store structured/unstructured data. However, it’s not as organised as a database |
| AMI | An Amazon Machine Image (AMI) is a template that contains a software configuration. It uses |
| Amazon Sagemaker | When a model needs to be built or modified as it provides access to instances of PyTorch and Jupyter notebook. |
| Amazon Redshift | It’s a DBMS framework. It is used for structured data. It is used for databases that require computationally heavy and complicated querying. |
| Amazon RDS | RDS is also a DBMS framework and is generally meant to be the primary transactional database. It is generally used for data which requires frequent quick querying |

Running Machine Learning at Production Scale



The above flowchart depicts a basic machine learning workflow.

Different parts of the pipeline require different computational power. Having various pieces in the workflow offers flexibility. Generally, the training part is computationally heavy and is not performed as often as the prediction part. A compute instance that will process the training code should have a better hardware than the inference part.

We also want different parts of the code to work at different times. We don’t train the model as often as we use the serialized model to predict or infer. Having them in different machines allows us to choose when to use them and hence in term offers increased flexibility. Also, it removes the bottlenecking and makes the flow even more efficient.

**PyTorch**

PyTorch is an open-source deep learning platform which integrates Python and its popular libraries and packages to write neural network layers in Python, right from research purposes to production deployment.

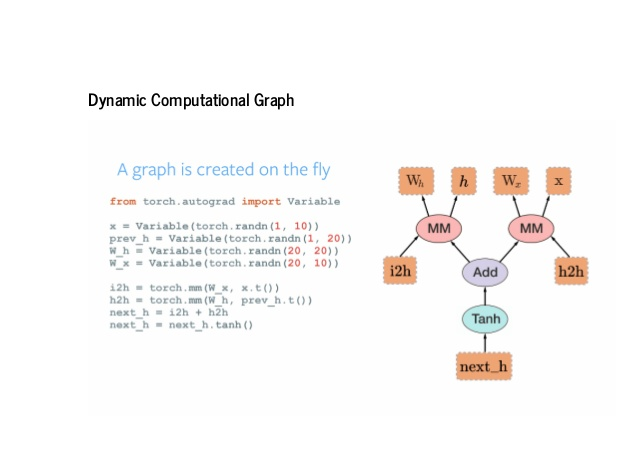


**Why PyTorch?**

Frameworks like scikit-learn are not considered scalable because their algorithms use a single thread and run computations sequentially. Hence large amount of computation will take a lot of time and hence is not considered a very scalable approach. However frameworks like PyTorch and TensorFlow are considered production ready and scalable because there is an underlying difference in how things run under the hood. An additional step is implemented before running the code. The interpreter reads the code and created an Interpretable Representation(IR). A Computational Graph is generated on the fly which tells the computer which operations can be run independent of each other which enables the possibility of running different computations parallelly and hence makes the process more scalable and production ready.

**Computational graph**

A computational graph tells the computer which operations can be run independent of each other and tells the most efficient way to run the code parallelly. It is useful for implementing scalable solution of doing computationally expensive calculations and operations on a large dataset.



Source: <https://www.slideshare.net/abdulmuneer/pytorch-for-tfdevelopers>

**Model Serialization**

**Serialization** basically means “to save”. Here, in this context we are saving the model and making an artifact of trained parameters of the model. This is because the Ideal Parameters can either come via a code file or from a data file. In the context of data storage, serialization is the process of translating data structures or object state into a format that can be stored or transmitted and reconstructed later. For instance, for a linear equation such as: y = 0.5\*x + 17.2

A serialized model would be like : { “int”: 17.2, “coeff”: 0.5 }

**Advantage:**

* Once serialized, in order to use the same model again, you do not need to train the model again which usually takes up a lot of time. You can use this serialized model to make predictions multiple number of times.
* Easy to use and can be customized
* Can be used as a mechanism to exchange objects between python libraries and third party libraries

**Disadvantage:**

* Should not be used for large sized data because it can cause significant overhead in terms of memory requirements

**How does PyTorch help in serialization?**

PyTorch offers a simple API which can either save all the weights of a model or pickle the entire class. Pickle is basically used for serializing and de-serializing a Python object structure. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea behind is that this character stream contains all the information necessary to reconstruct the object in another python script.

There are two main approaches for serializing and restoring a mode in PyTorch:

* The first saves and loads only the model parameter
* The second saves and loads the entire model. However in this case, the serialized data is bound to the specific classes and the exact directory structure used, so it can break in various ways when used in other projects, or after some serious refactors.

Running PyTorch on AWS

There are various methods to run PyTorch on AWS. One of the easiest ways to spin up a Jupyter notebook with PyTorch pre-loaded is by using the Amazon Sagemaker service. The steps to use Sagemaker to spin up a PyTorch instance are:

* Create an Amazon SageMaker Notebook instance
* Open the instance created (this will give you a Notebook interface)
* Create a new notebook and select ‘conda\_pytorch\_p36’ which will make Python 3 and PyTorch available in the Jupyter Notebook

Other ways to run PyTorch can include:

* Using an EC2 compute resource clubbed with an EBS storage and installing an OS and all the prerequisite software and packages (including PyTorch) manually
* Using an AMI with a preinstalled OS and manually installing required software sans packages manually

**Model artifact** is an artifact in the Unified Modelling Language (UML) is the specification of a physical piece of information that is used or produced by a software development process, or by deployment and operation of a system.

Examples of artifacts include model files, source files, scripts, and binary executable files, a table in a database system, a development deliverable, a word-processing document, or a mail message.

**ONNX**

The Open Neural Network Exchange (ONNX)​format is meant to provide a common way to represent the data used by neural networks. ONNX provides a definition of an extensible computational graph model as well as definitions of built-in operators and standard data types. Most frameworks have their own specific model format that will only work with models from other frameworks by way of a conversion tool. Ideally we want everyone to have specialised format that can be read by any program. ONNX allows models to be swapped freely between frameworks without the conversion process. A model trained on one framework can be used for inference by another framework.

Some benefits of ONNX:

* **Framework Interoperability**: Enabling interoperability makes it possible to get great ideas into production faster. ONNX enables models to be trained in one framework and transferred to another for inference. ONNX models are currently supported in Caffe2, Microsoft Cognitive Toolkit, MXNet, and PyTorch, and there are connectors for many other common frameworks and libraries.
* Hardware Optimizations: ONNX makes it easier for optimizations to reach more developers. Any tools exporting ONNX models can benefit ONNX-compatible runtimes and libraries designed to maximize performance on some of the best hardware in the industry.



**Managing model artifacts from PyTorch in AWS**

Amazon SageMaker provides an open-source container that makes writing a PyTorch script and running it in Amazon SageMaker easier. SageMaker PyTorch Container is an open source library for making the PyTorch framework run on Amazon SageMaker.

This repository also contains Docker files which install this library, PyTorch, and dependencies for building SageMaker PyTorch images.

The SageMaker team uses this repository to build its official PyTorch image. For end users, this repository is typically of interest if you need implementation details for the official image, or if you want to use it to build your own customized PyTorch image.

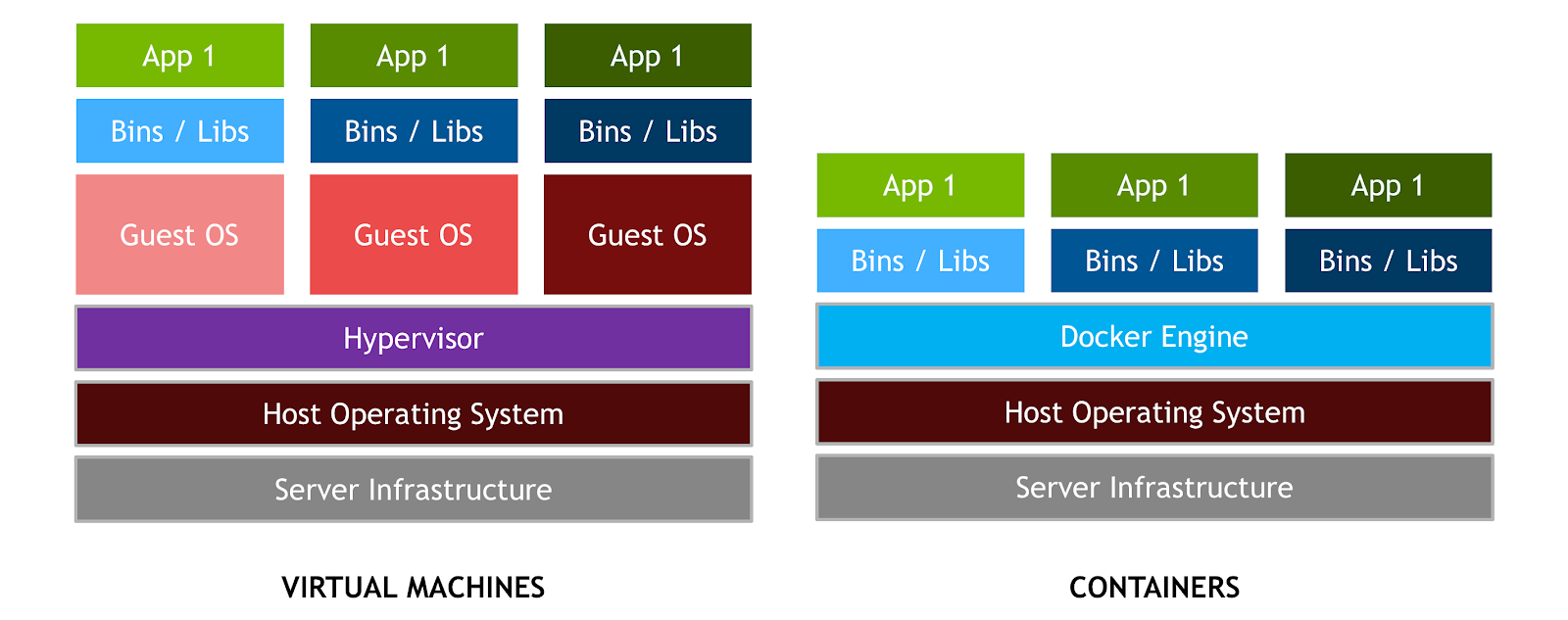
One can save models made in PyTorch in ONNX format and the same ONNX model can now be used in the supported frameworks like CAFFE2 and TensorFlow.

Deploying PyTorch and other Processing Stages with Docker

**Docker** is a computer program that performs operating-system-level virtualization, also known as "containerization"

Docker has been a popular choice for data processing as it shares the underlying resources of the machine and have images for a code that can run in any environment like windows, Linux etc. It acts essentially like a virtual machine but offers the following advantages over VM:

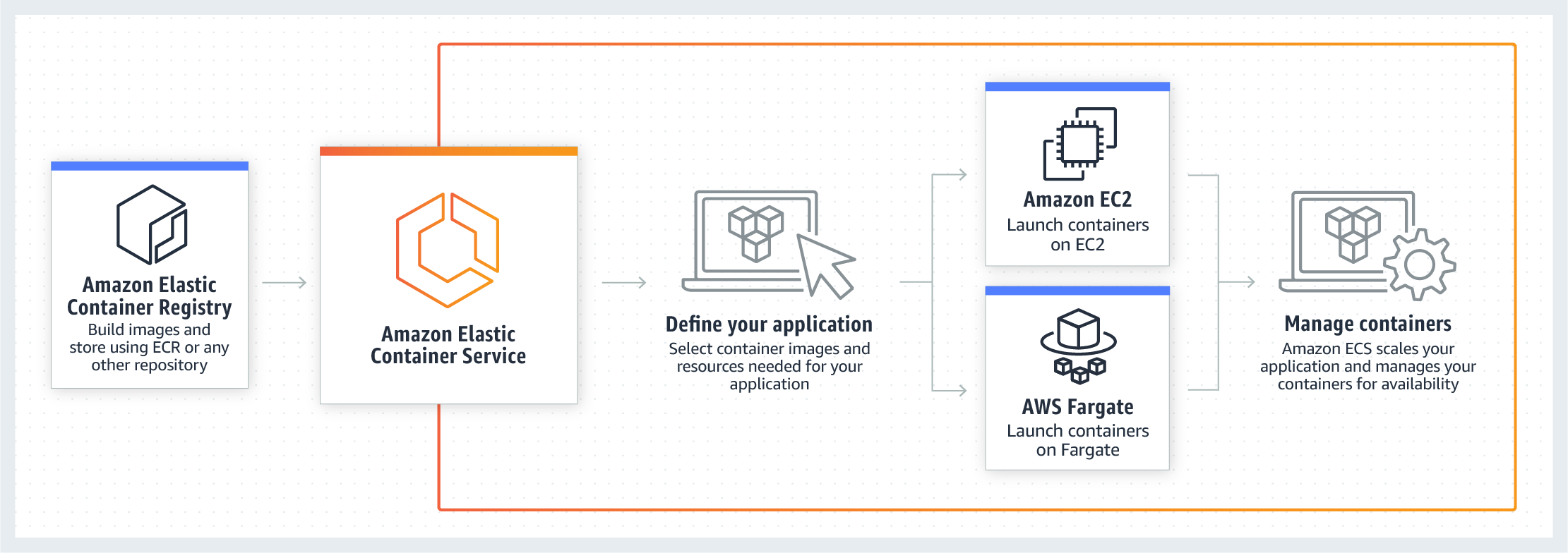
* Requires lesser storage as compared to VM as it eliminates the need of a Guest OS which takes up a lot of space
* It is quicker to run
* It’s quicker to spin up or spin down
* You don’t need to worry about installing software/packages
* Containerization breaks barriers for devOp teams to accelerate deployment times and frequency
* Containerization enables deployment and operational agility



Source: <https://devblogs.nvidia.com/wp-content/uploads/2016/06/VM_vs_Docker.png>

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high-performance [container](https://aws.amazon.com/containers/) orchestration service that supports [Docker](https://aws.amazon.com/docker/) containers and allows one to easily run and scale containerized applications on AWS. Amazon ECS eliminates the need to install and operate one’s own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.

With simple API calls, one can launch and stop Docker-enabled applications, query the complete state of your application, and access many familiar features.[5]



Source: <https://aws.amazon.com/ecs/>

Serving Serialized Models

After serializing a model we can utilize / serve the model of an **API layer** or go **for batch data processing.**

**Batch Processing**

In batch processing, the tasks are scheduled to work on an accumulated data set. Batch jobs can be stored up during working hours and then executed during the evening or whenever the computer is idle. Batch processing is particularly useful for operations that require the computer or a peripheral device for an extended period of time. An example of batch processing is the way that credit card companies process billing. The customer does not receive a bill for each separate credit card purchase but one monthly bill for all of that

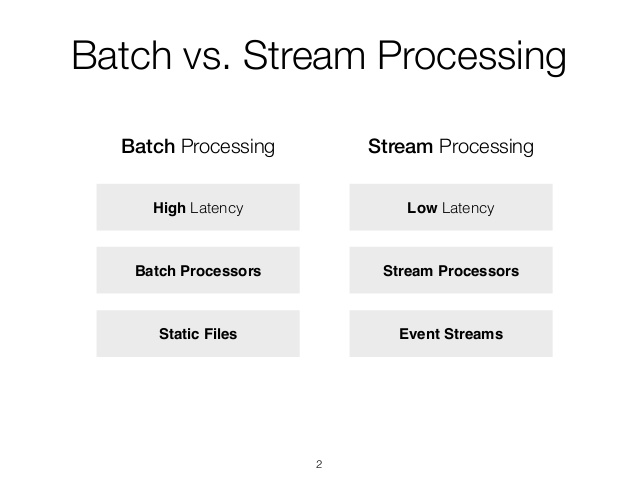
month



**API Layer Processing(Stream processing)**

API Layer is different from batch processing in a way that we have to make it available to the users at all times, after the final steps of pre-processing the data. This means, user gets the result in real time instead of waiting for the batch of job to complete.





**How to serve a model with a custom Docker container and MXNet**

First build a model and push it into an API layer to allow other users get access to the result in standardized way. A layer of code takes input, runs a model gives back the output. Still a batch component is used in sense of making updates in regular intervals.

The model is taken out of S3 object storage, and put into a system that takes in a request and the updated model is put back in S3, available to all clients.

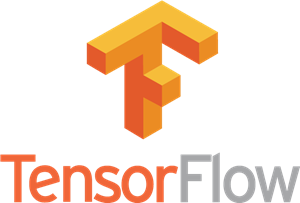
**Model Server for Apache MXNet (**MMS) is an open source component that is designed to simplify the task of deploying deep learning models for inference at production scale. With MMS, AWS contributes an open source engineering toolset for Apache MXNet which simplifies the process of deploying deep learning models.

Features:

* Support for the OpenAPI specifications, which enables easy integration and auto-generation of client code for popular stacks such as Java, JavaScript, C#, and more
* Tooling to package and export all model artifacts into a single “model archive” file that encapsulates everything required for serving an MXNet model.
* Automated setup of a serving stack, including HTTP inference endpoints
* Pre-configured Docker images, set up with NGINX, MXNet, and MMS, for scalable model serving
* Ability to customize every step in the inference execution pipeline, from model initialization, through pre-processing and inference, up to post-processing the model’s output.
* Real-time operational metrics to monitor the inference service and endpoints, covering latencies, resource utilization, and errors.

Some other alternatives to MXnet are :

* Tensorflow Serving
* Clipper
* Deep Detect



Conclusion

Running data analytics at production level is very different from running the same analytics on a non-production scale level. There are various challenges that one faces when the analytics tool is deployed for the end user. Fortunately, this issue is well recognized by some companies and there are a lot of tools and resources available to tackle such problems. AWS is an example of such a service offered by Amazon. With the tools for all kinds of users with different levels of experiences and different use-cases, AWS does a really good job to facilitate the scalability and deployment of a data analytics workflow and makes the task really easy with all the customer support they provide. Overall, the task of scaling an analytics framework has been really enhanced with all kinds of services available and one needs to stay updated in order to make the optimum use these resources and increase profitability in their organisation.

References:

[1] <https://www.sas.com/en_us/insights/analytics/machine-learning.html>

[2] <https://searchstorage.techtarget.com/definition/block-storage>

[3] <https://searchstorage.techtarget.com/definition/file-storage>

[4] <https://searchstorage.techtarget.com/definition/object-storage>

[5] <https://aws.amazon.com/ecs/>