BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Deployment Patterns

Pravin Y Pawar

Adapted from “Machine Lerning Engineering”
By Andriy Burkov

Model Deployment Patterns

- Once the model has been built and thoroughly tested, it can be deployed
  - means to make it available for accepting queries generated by the users of the production system

- Once the production system accepts the query, the latter is transformed into a feature vector
  - The feature vector is then sent to the model as input for scoring
  - The result of the scoring then is returned to the user

- A trained model can be deployed in various ways
  - can be deployed on a server, or on a user's device
  - can be deployed for all users at once, or to a small fraction of users

- A model can be deployed following several patterns:
  - statically, as a part of an installable software package,
  - dynamically on the user's device,
  - dynamically on a server, or
  - via model streaming

Static Deployment

- Very similar to traditional software deployment
  - prepare an installable binary of the entire software
  - model is packaged as a resource available at the runtime

- Depending on the operating system and the runtime environment
  - Objects of both the model and the feature extractor can be packaged as a
    - part of a dynamic-link library (DLL on Windows),
    - Shared Objects (*.so files on Linux),
    - or be serialized and saved in the standard resource location for virtual machine-based systems,
    - such as Java and .Net.

Static Deployment(2)

Pros and Cons

- Many advantages:
  - software has direct access to the model, so the execution time is fast for the user
  - user data doesn't have to be uploaded to the server at the time of prediction
    - saves time and preserves privacy
  - model can be called when the user is offline
  - software vendor doesn't have to care about keeping the model operational
    - becomes the user's responsibility

- Several drawbacks:
  - The separation of concerns between the machine learning code and the application code isn't always obvious
    - makes it harder to upgrade the model without also having to upgrade the entire application
  - If the model has certain computational requirements for scoring (such as access to an accelerator or a GPU)
    - may add complexity and confusion as to where the static deployment can or cannot be used

Dynamic Deployment on User’s Device

- Similar to a static deployment, in the sense the user runs a part of the system as a software application on their device
- Difference is that in dynamic deployment, the model is not part of the binary code of the application
- Achieves better separation of concerns!
  - Pushing model updates is done without updating the whole application running on the user’s device
- Dynamic deployment can be achieved in several ways:
  - by deploying model parameters,
  - by deploying a serialized object, and
  - by deploying to the browser

Deployment of Model Parameters

- In this deployment scenario, the model file only contains the learned parameters
  - user's device has installed a runtime environment for the model
- Some machine learning packages, like TensorFlow,
  - have a lightweight version that can run on mobile devices
- Alternatively, frameworks such as Apple's Core ML allow running models on Apple devices
  - created using popular packages, including scikit-learn, Keras, and XGBoost

Deployment of a Serialized Object

- Model file is a serialized object that the application would deserialize

- The advantage is that don’t need to have a runtime environment for model on the user’s device
  - all needed dependencies will be deserialized with the object of the model

- An evident drawback is that an update might be quite “heavy,”
  - which is a problem if your software system has millions of users

Deploying to Browser

- Most modern devices have access to a browser, either desktop or mobile

- Some machine learning frameworks, such as TensorFlow.js,
  - have versions that allow to train and run a model in a browser, by using JavaScript as a runtime

- Even possible to train a TensorFlow model in Python,
  - then deploy it to, and run it in the browser's JavaScript runtime environment
  - if a GPU (graphics processing unit) is available on the client's device, Tensorflow.js can leverage it

Dynamic Deployment on User's Device(2)

Advantages and Drawbacks

- Advantages
  - Calls to the model will be fast for the user
  - Reduces the impact on the organization's servers, as most computations are performed on the user's device

- If the model is deployed to the browser,
  - advantage is organization's infrastructure only needs to serve a web page that includes the model's parameters
  - A downside is bandwidth cost and application startup time might increase.
    - users must download the model's parameters each time they start the web application
    - as opposed to doing it only once when they install an application

Dynamic Deployment on User's Device(3)

Advantages and Drawbacks

- Monitoring
    - Deploying to a user's device makes it difficult to monitor the model performance

- Model updates
    - A serialized object can be quite voluminous
    - Some users may be offline during the update, or even turn off all future updates
    - may end up with different users using very different model versions
    - becomes difficult to upgrade the server-side part of the application

- Third-party analyses
    - Deploying models on the user's device means that the model easily becomes available for third-party analyses
    - may try to reverse-engineer the model to reproduce its behavior
    - may search for weaknesses by providing various inputs and observing the output
    - may adapt their data so the model predicts what they want

Dynamic Deployment on a Server

- Because of the complications with other approaches, and problems with performance monitoring,
- Most frequent deployment pattern is to place the model on a server (or servers),
  - make it available as
    - a Representational State Transfer application programming interface (REST API) in the form of a web service,
    - Google’s Remote Procedure Call (gRPC) service

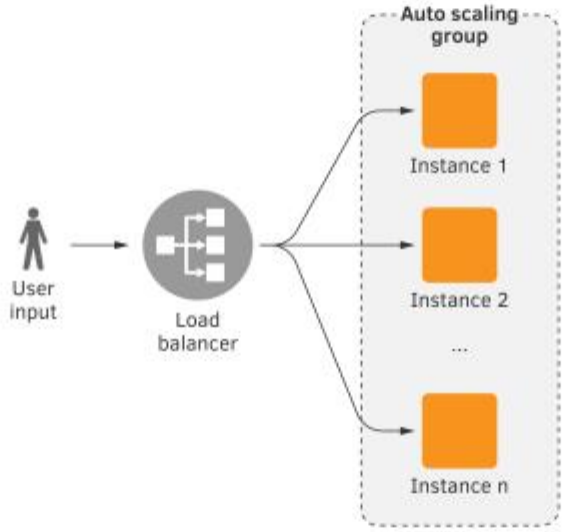
- Four common practices
  - Deployment on a Virtual Machine
  - Deployment in a Container
  - Serverless Deployment
  - Model Streaming

Deployment on a Virtual Machine(VM)

- In a typical web service architecture deployed in a cloud environment
  - predictions are served in response to canonically-formatted HTTP requests

- A web service running on a virtual machine
  - receives a user request containing the input data,
  - calls the machine learning system on that input data
  - then transforms the output of the machine learning system into the output JSON or XML string

- To cope with high load, several identical VMs are running in parallel
  - A load balancer dispatches the incoming requests to a specific virtual machine
  - VMs can be added and closed manually, or be a part of an autoscaling group that launches
  - VMs can be terminated virtual machines based on their usage



: Deploying a machine learning model as a web service on a virtual machine.

Deployment on a Virtual Machine(VM)2

- In Python, a REST API web service is usually implemented
  - using a web application framework such as Flask or FastAPI
- TensorFlow, a popular framework used to train deep models,
  - comes with TensorFlow Serving, a built-in gRPC service
- Advantage: Architecture of the software system is conceptually simple: a typical web or gRPC service
- Downsides
  - Need to maintain servers (physical or virtual)
  - If virtualization is used, there is an additional computational overhead due to virtualization and running multiple OS
  - Network latency, which can be a serious issue, depending on how fast you need to process scoring results
  - has a relatively higher cost, compared to deployment in a container, or a serverless deployment

Deployment in a Container

- A more modern alternative to a virtual-machine-based deployment
  - considered more resource-efficient and flexible than with virtual machines

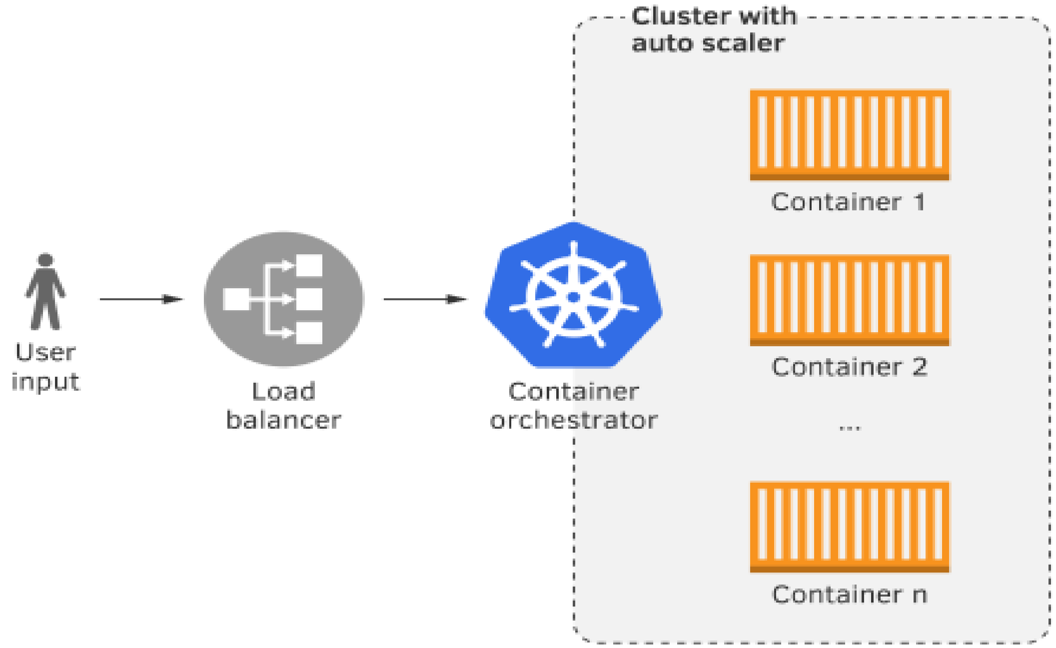
- A container is similar to a virtual machine
  - in the sense that it is also an isolated runtime environment with its own filesystem, CPU, memory, and process space
- The main difference, however, is that all containers are running on the same virtual or physical machine
  - share the operating system, while each virtual machine runs its own instance of the operating system

- Deployment Process
  - The machine learning system and the web service are installed inside a container
    - Usually, a container is a Docker container, but there are alternatives
  - Then a container-orchestration system is used to run the containers on a cluster of physical or virtual servers
    - A typical choice of a container-orchestration system for running on-premises or in a cloud platform, is Kubernetes
    - Some cloud platforms provide both their own container-orchestration engine, such as AWS Fargate and Google Kubernetes Engine

Deployment in a Container(2)

Organization

- Virtual or physical machines are organized into a cluster,
  - whose resources are managed by the container orchestrator
- New virtual or physical machines can be manually added to the cluster, or closed
- If your software is deployed in a cloud environment,
  - a cluster autoscaler can launch (and add to the cluster) or terminate virtual machines
  - based on the usage of the cluster.



Deploying a model as a web service in a container running on a cluster.

Deployment in a Container(3)

- Advantage
  - More resource-efficient as compared to the deployment on a virtual machine
  - Allows the possibility to automatically scale with scoring requests
  - Allows us to scale-to-zero- reduced down to zero replicas when idle and brought back up if there is a request to serve
    - the resource consumption is low compared to always running services
    - leads to less power consumption and saves cost of cloud resources

- Drawback
  - Containerized deployment is generally seen as more complicated, and requires expertise

Serverless Deployment

- Several cloud services providers, including Amazon, Google, and Microsoft, offers serverless computing
  - Lambda-functions on Amazon Web Services, and Functions on Microsoft Azure and Google Cloud Platform

- The serverless deployment consists of preparing a zip archive
  - with all the code needed to run the machine learning system (model, feature extractor, and scoring code)
  - must contain a file with a specific name that contains a specific function
  - is uploaded to the cloud platform and registered under a unique name

- The cloud platform provides an API to submit inputs to the serverless function
  - specifies its name, provides the payload, and yields the outputs

- The cloud platform takes care of
  - deploying the code and the model on an adequate computational resource,
  - executing the code,
  - routing the output back to the client

Serverless Deployment(2)

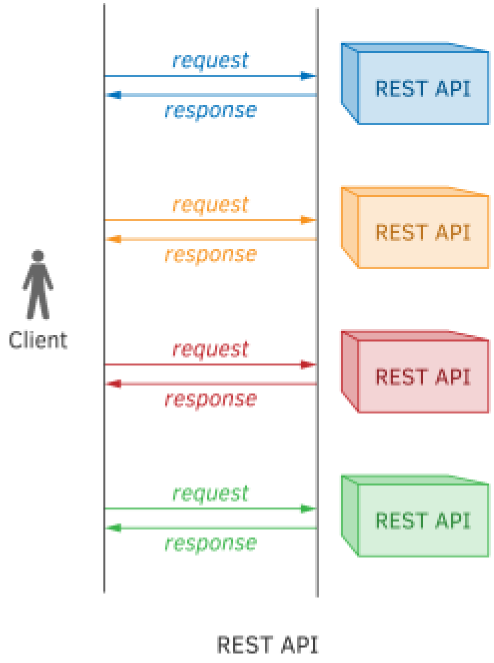
Advantages and Limitations

- Advantages to relying on serverless deployment
  - don’t have to provision resources such as servers or virtual machines
  - don’t have to install dependencies, maintain, or upgrade the system
  - highly scalable and can easily and effortlessly support thousands of requests per second
  - support both synchronous and asynchronous modes of operation
  - cost-efficient: only pay for compute-time
  - simplifies canary deployment, or canarying
  - Rollbacks are also very simple in the serverless deployment because it is easy to switch back to the previous version of the function

- Limitations
  - Restrictions by the cloud service provider
    - the function’s execution time, zip file size, and amount of RAM available on the runtime
  - Zip file size limit can be a challenge - A typical model requires multiple heavyweight dependencies
  - Unavailability of GPU access can be a significant limitation for deploying deep models

Model Serving – REST API Revisited

- In complex systems, there can be many models applied to the same input
  - a model can input a prediction from another model
- For example, the input may be a news article
  - One model can predict the topic of the article
  - Another model can extract named entities
  - Third model can generate a summarization of the article, and so on
- According to the REST API deployment pattern, need one REST API per model
  - client would call one API by sending a news article as a part of the request - get the topic as response
  - client calls another API by sending a news article, and gets the named entities as response; etc.



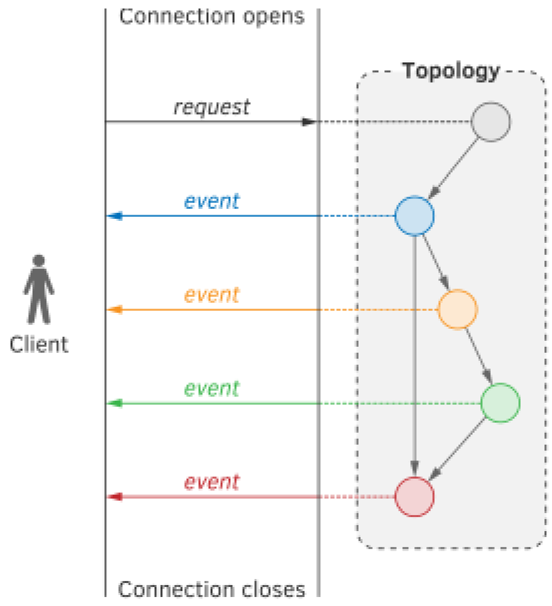
Model Streaming

Can be seen as an inverse to the REST API

- Streaming works differently
  - All models, as well as the code needed to run them, are registered within a stream-processing engine (SPE)
  - Apache Storm, Apache Spark, and Apache Flink
  - Or, they are packaged as an application based on a stream-processing library (SPL),
  - such as Apache Samza, Apache Kafka Streams, and Akka Streams

- Based on notion of data processing topology
  - Input data flows in as an infinite stream of data elements sent by the client
  - Following a predefined topology, each data element in the stream undergoes a transformation in the nodes of the topology
  - Transformed, the flow continues to other nodes.

- In a stream-processing application, nodes transform their input in some way,
  - then either,
    - send the output to other nodes, or
    - send the output to the client, or
    - persist the output to the database or a filesystem.



streaming

Thank You!

In our next session: