Deploying ML models in Production

Part 2: Model Serving Patterns and Infrastructure

Ramin Toosi

ML Infrastructure

- **On Prem**
  - Train and deploy on your own hardware infrastructure
  - Manually procure hardware GPUs, CPUs etc
  - Profitable for large companies running ML projects for longer time
- **On Cloud**
  - Train and deploy on cloud choosing from several service providers
    - Amazon Web Services, Google Cloud Platform, Microsoft Azure, etc

Model Serving

- On Prem
  - Can use open source, pre-built servers
    - TF-Serving, KF-Serving, NVidia and more...
- On Cloud
  - Create VMs and use open source pre-built servers
  - Use the provided ML workflow

Model Servers

- Simplify the task of deploying machine learning models at scale.
- Can handle scaling, performance, some model lifecycle management etc.,

Model Servers

TensorFlow Serving

PyTorch

TorchServe

Kubeflow

KF Serving

NVIDIA

TRITON INFERENCE SERVER

TensorFlow Serving

TensorFlow

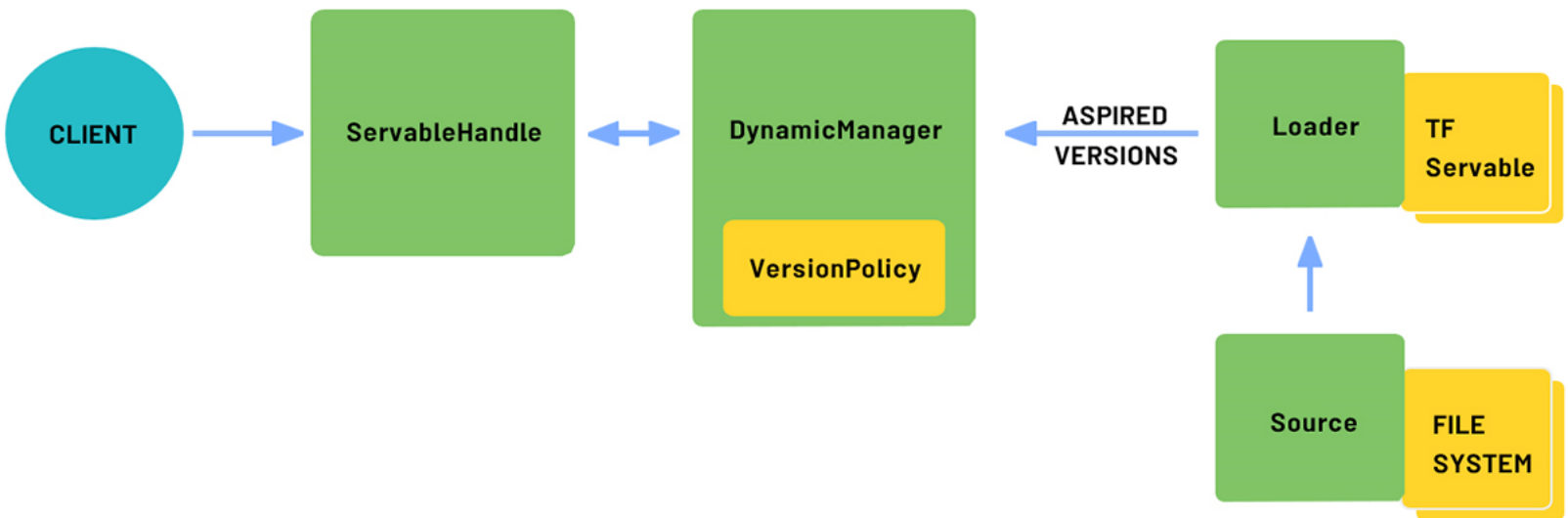
Supports many servables

TF Models	Non TF Models	Word Embeddings	Vocabularies	Feature Transformations

- Out of the box integration with TensorFlow Models
- Batch and Real-time
- Multi-Model Serving Inference
- Exposes gRPC and REST endpoints

TensorFlow Serving Architecture

- **Legend:**
  - **Blue:** Caller-specific code
  - **Green:** TF Serving core classes / APIs
  - **Yellow:** Pluggable implementations



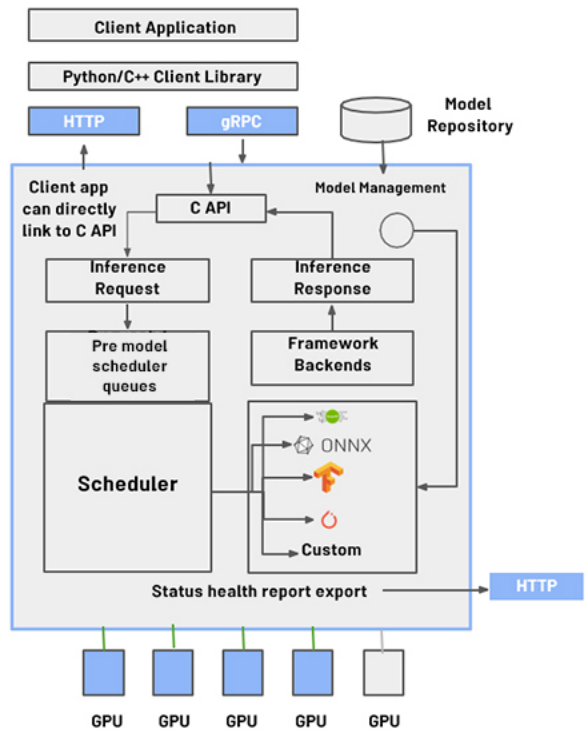
NVIDIA Triton Inference Server

NVIDIA
TRITON INFERENCE SERVER

- Simplifies deployment of AI models at scale in production.
- Open source inference serving software
- Deploy trained models from any framework:
  - TensorFlow, TensorRT, PyTorch, ONNX Runtime, or a custom framework
- Models can be stored on:
  - Local storage, AWS S3, GCP, Any CPU-GPU Architecture (cloud, data centre or edge)
- HTTP REST or gRPC endpoints are supported.

Architecture

- **Triton Inference Server Architecture supports:**
  - Single GPU for multiple models from same or different frameworks
  - Multi-GPU for same model
    - Can run instances of model on multiple GPUs for increased inference performance.
- **Supports model ensembles.**

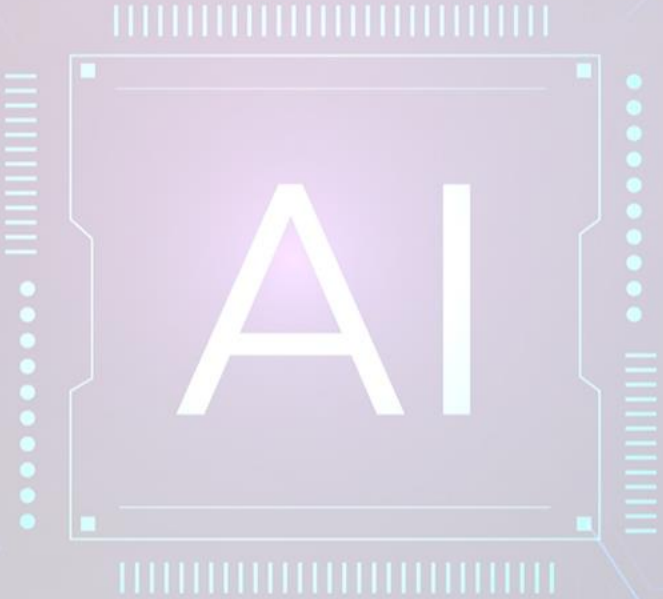


Torch Serve

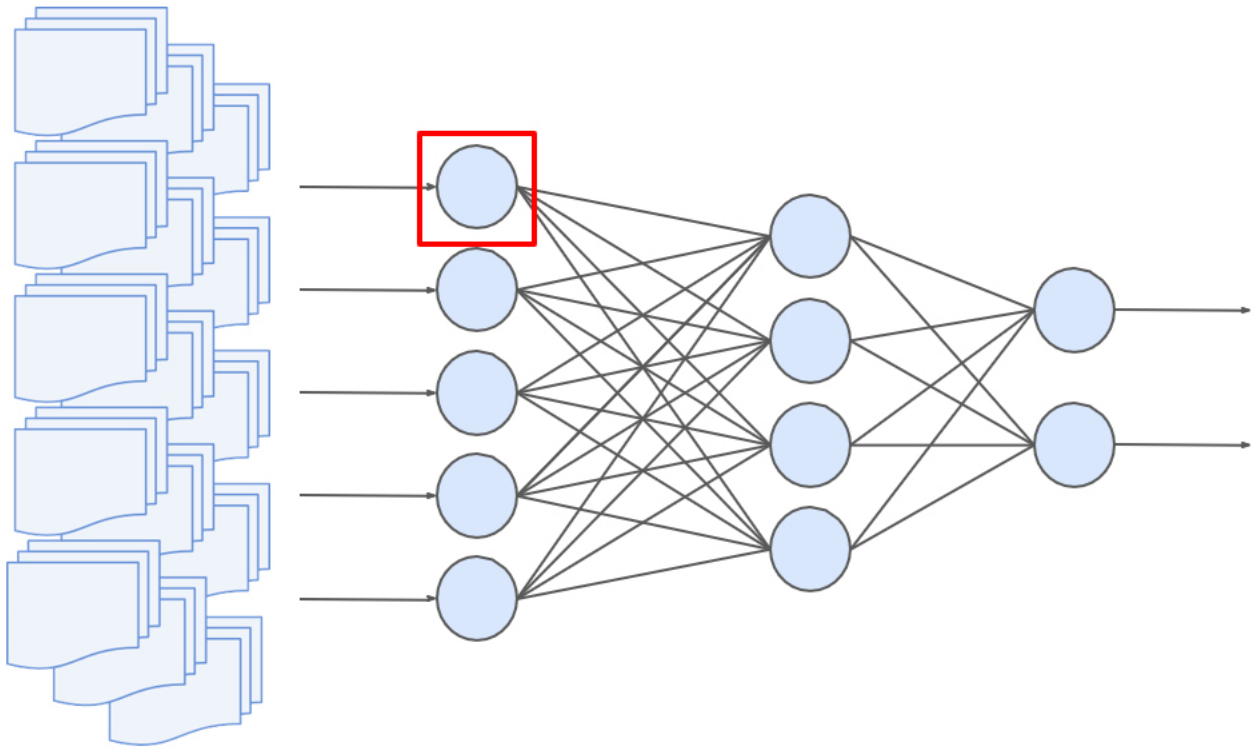
- Model serving framework for PyTorch models.
- Initiative from AWS and Facebook
- Batch and Real-time Inference
- Multi-Model Serving
- Supports REST Endpoints
- Monitor Detail Logs and Customized Metrics
- Default handlers for Image Classification
- Object Detection, Image Segmentation, Text Classification
- A/B Testing

TorchServe

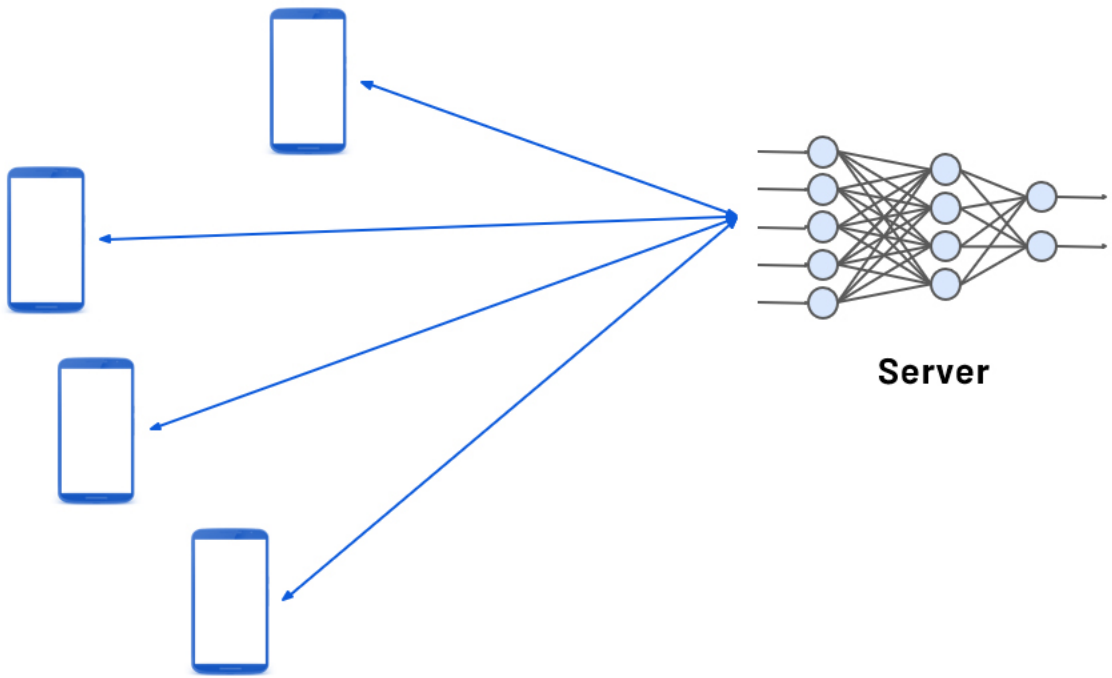
Scaling



Why is Scaling Important?



Why is Scaling Important?



vertical scaling

- Increased Power
- Upgrading
- More RAM
- Faster Storage
- Adding or upgrading GPU/TPU

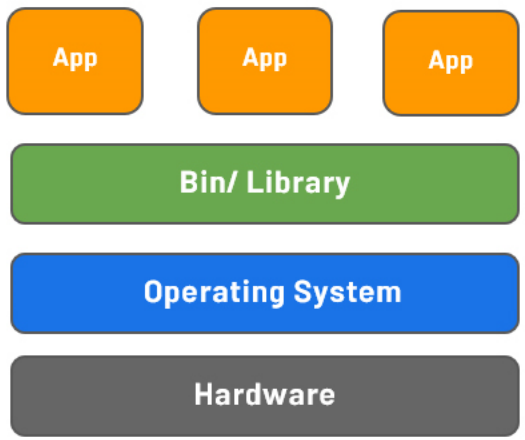
horizontal scaling

- More CPUs/GPUs instead of bigger ones
- Scale up as needed
- Scale back down to minimums

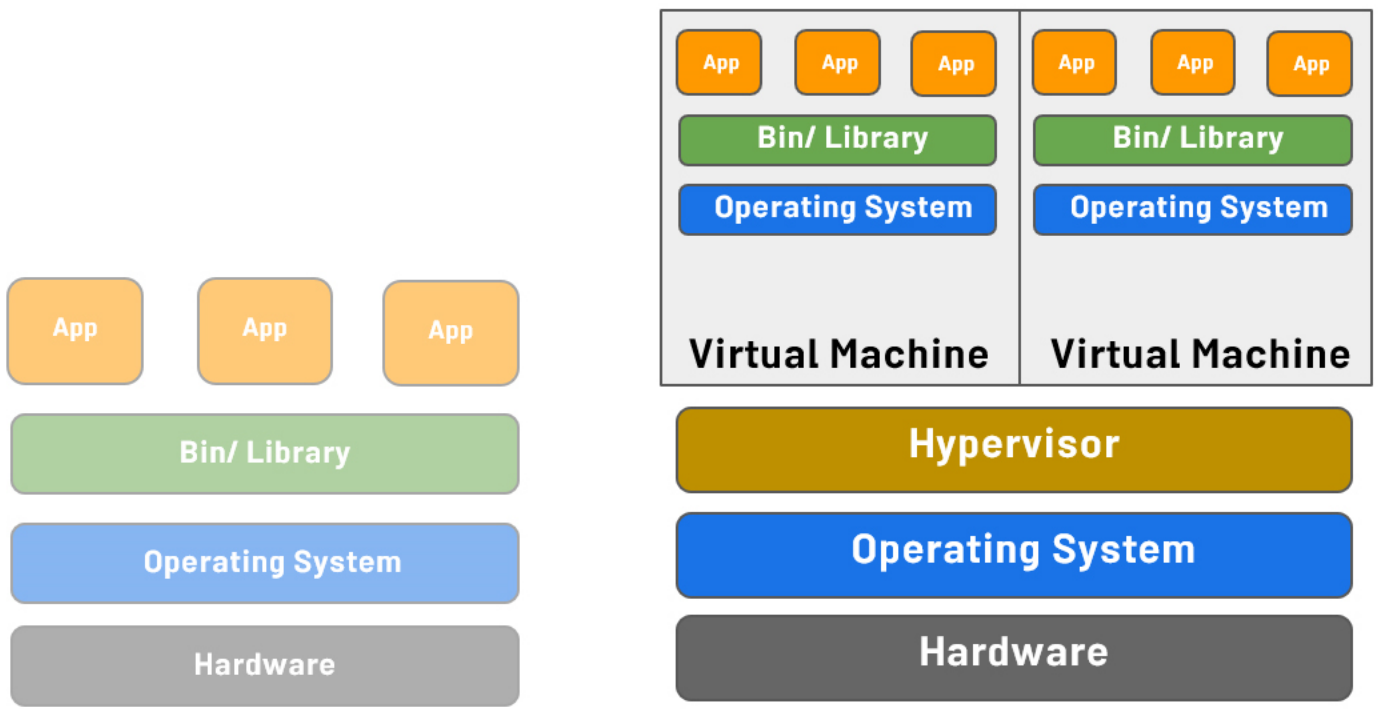
Why Horizontal Over Vertical Scaling

- **Benefit of elasticity**
  - Shrink or grow no of nodes based on load, throughput, latency requirements.
- **Application never goes offline**
  - No need for taking existing servers offline for scaling
- **No limit on hardware capacity**
  - Add more nodes any time at increased cost

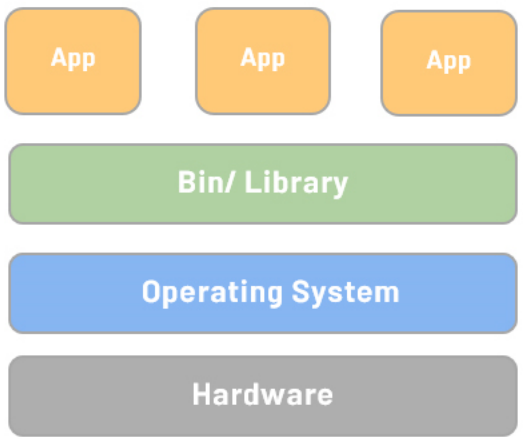
Typical System Architecture

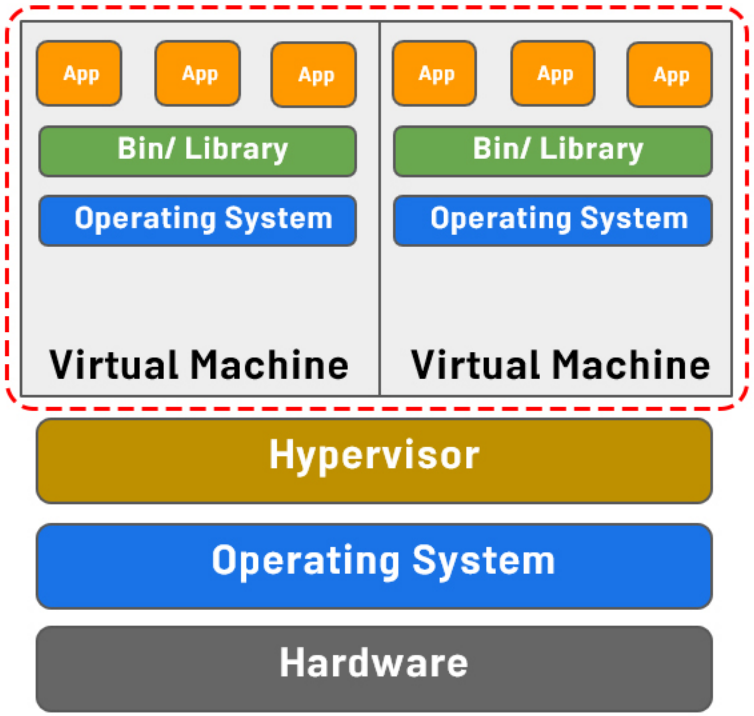


Virtual Machine (VM) Architecture



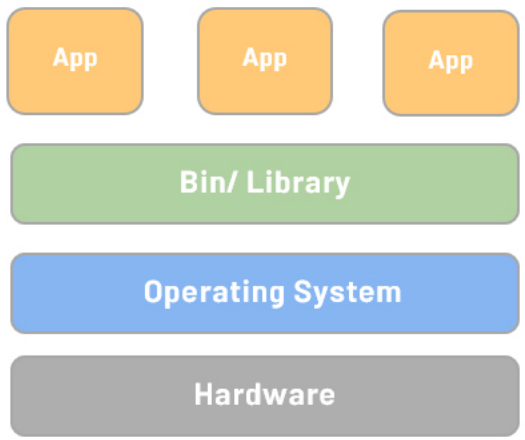
VM Management

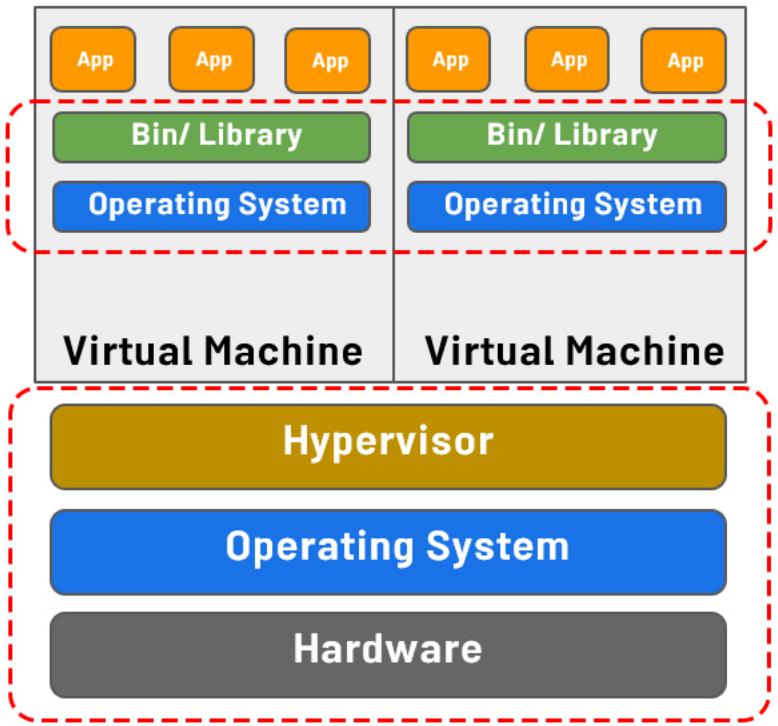




Hypervisor and Scaling

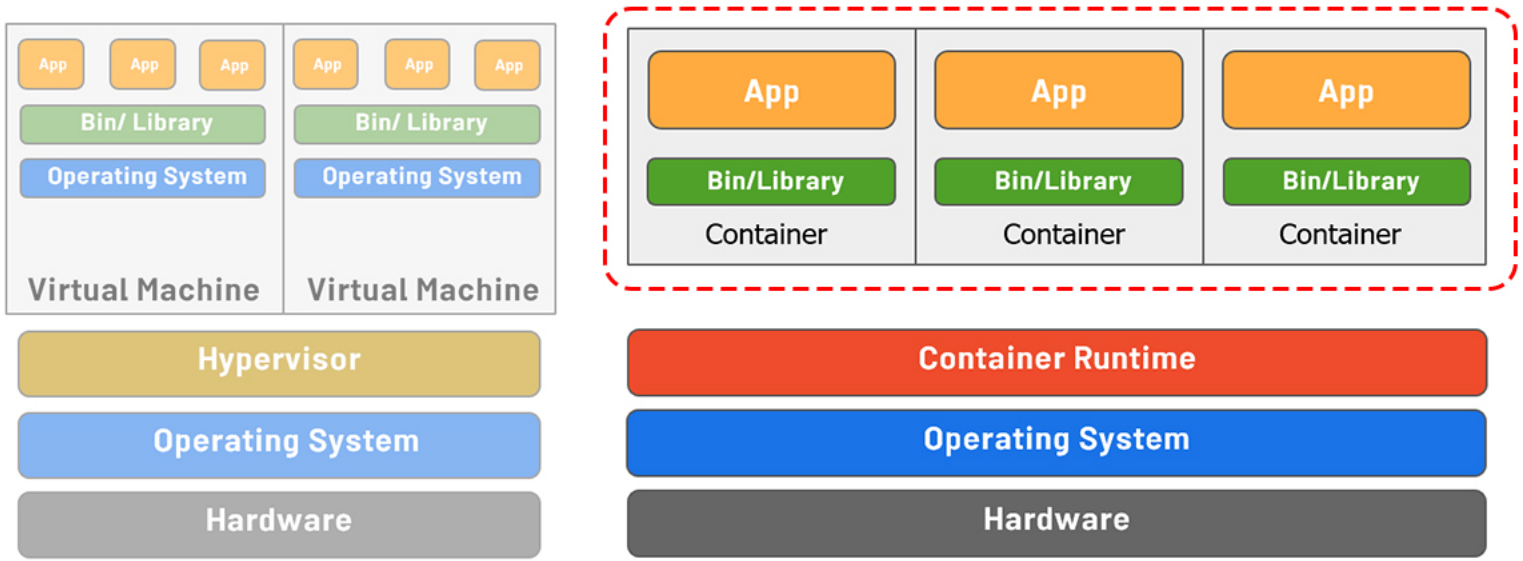
VM limitation





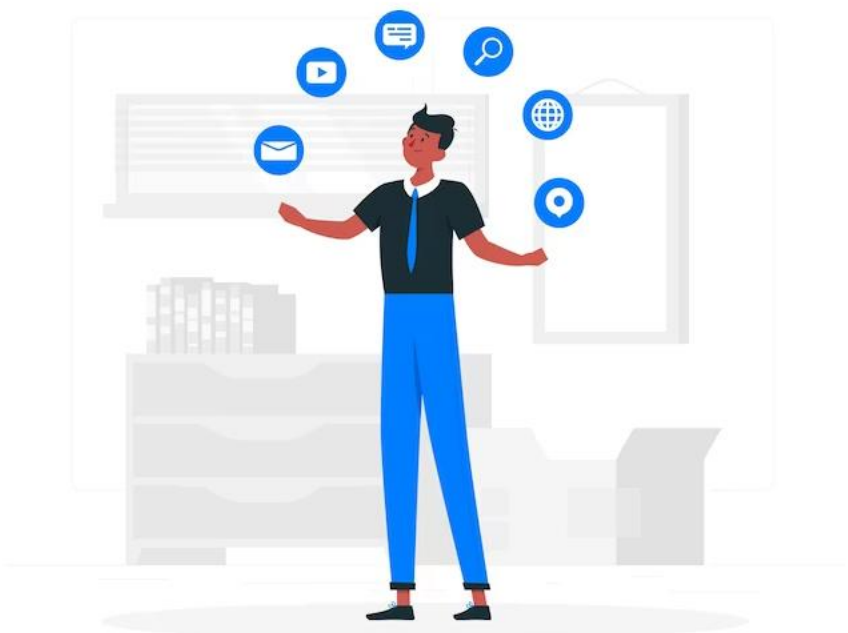
Building Containers

Container Advantages



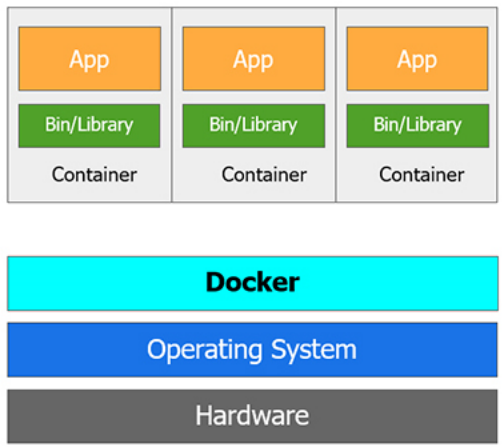
Containers Advantages

- Less OS requirements - more apps!
- Abstraction
- Easy deployment based on container runtime



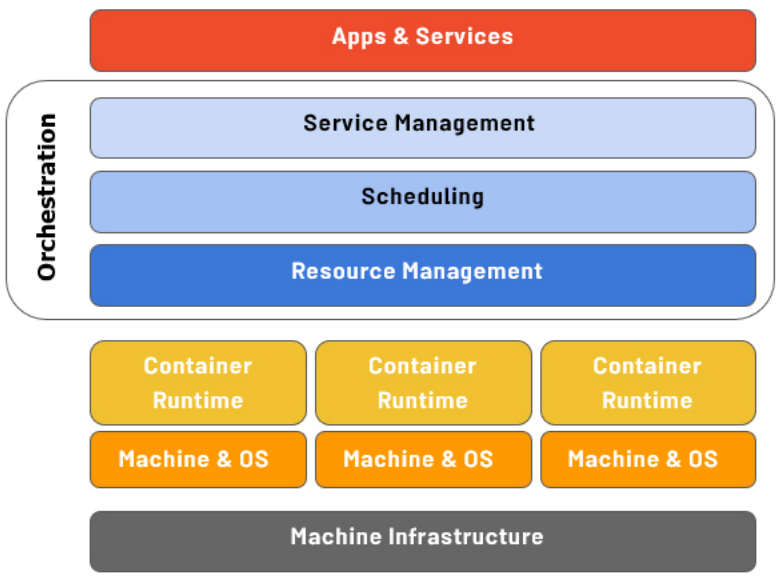
Docker: Container Runtime

- Open source container technology
- Most popular container runtime
- Started as container technology for Linux
- Available for Windows applications as well.
- Can be used in data centres, personal machines or public cloud.
- Docker partners with major cloud services for containerization.



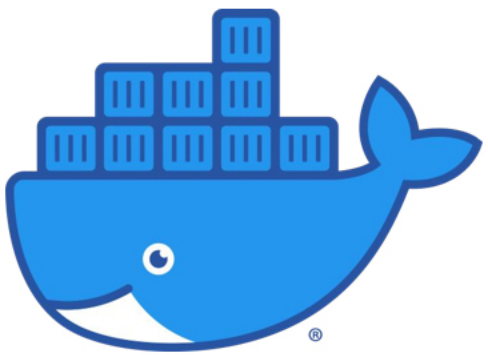
Enter Container Orchestration

- Manages life cycle of containers in production environments
- Manages scaling of containers
- Ensures reliability of containers
- Distributes resources between containers
- Monitors health of containers



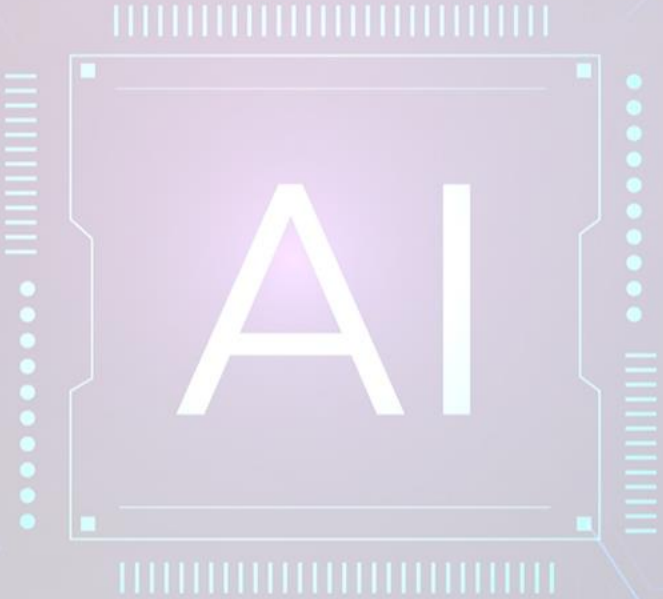
Popular Container Orchestration Tools

Kubernetes



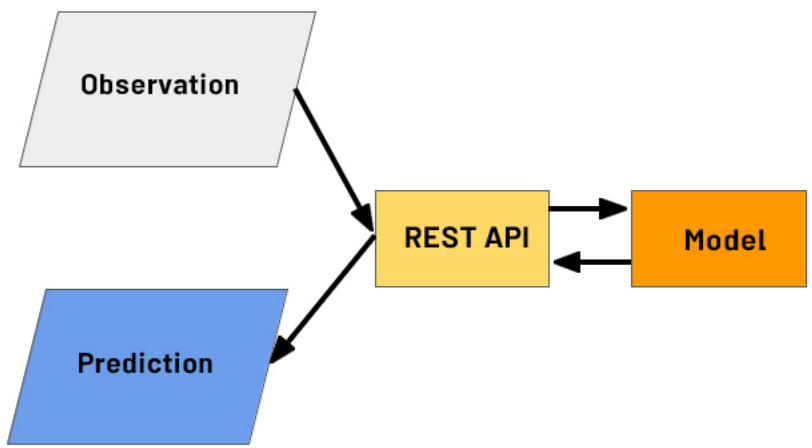
Docker Swarm

Online Inference



Online Inference

- Process of generating machine learning predictions in real time upon request.
- Predictions are generated on a single observation of data at runtime.
- Can be generated at any time of the day on demand



Optimising ML Inference

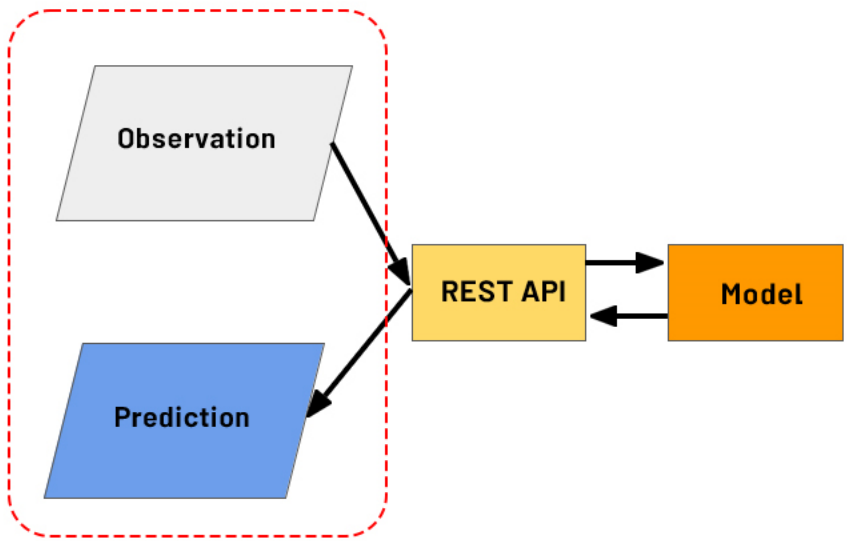
Metrics to Optimize

- Latency
- Throughput
- Cost

Optimising ML Inference

Metrics to Optimize

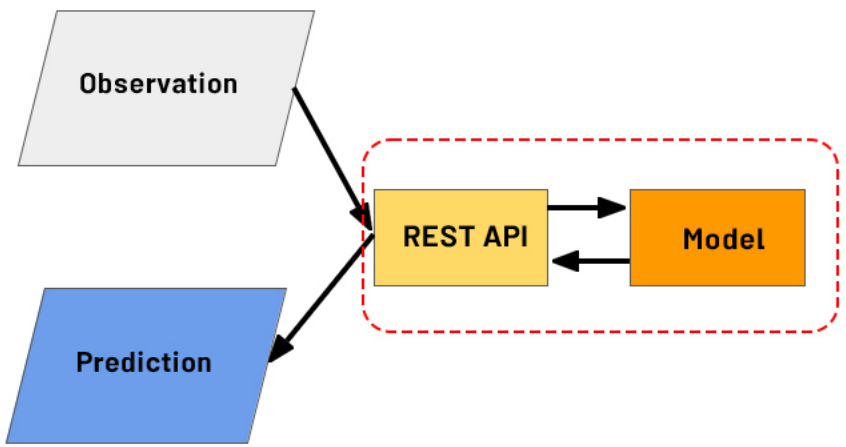
- Latency
- Throughput
- Cost



Optimising ML Inference

Metrics to Optimize

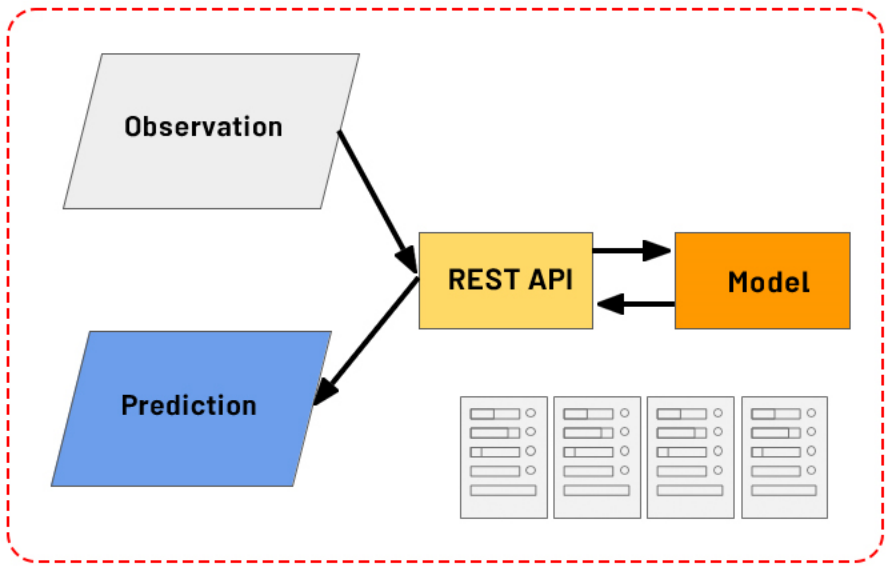
- Latency
- Throughput
- Cost



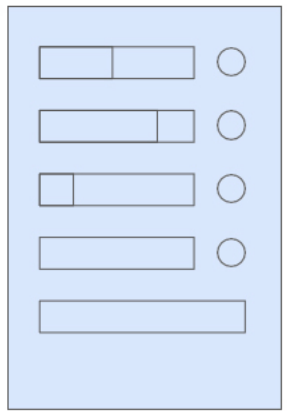
Optimising ML Inference

Metrics to Optimize

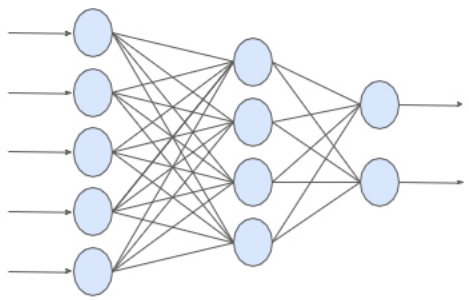
- Latency
- Throughput
- Cost



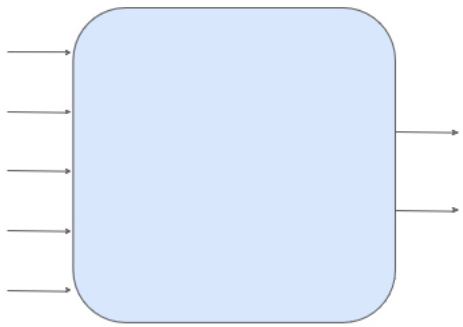
Inference Optimization



Infrastructure



Model Architecture



Model Compilaion

Preprocessing Operations Needed Before Inference

Data Cleansing

- Correcting invalid values in incoming data

Feature Tuning

- Normalization
- Clipping outliers
- Imputing Missing Values

Feature Construction

- Combine inputs
- Feature crossing
- Polynomial expansion

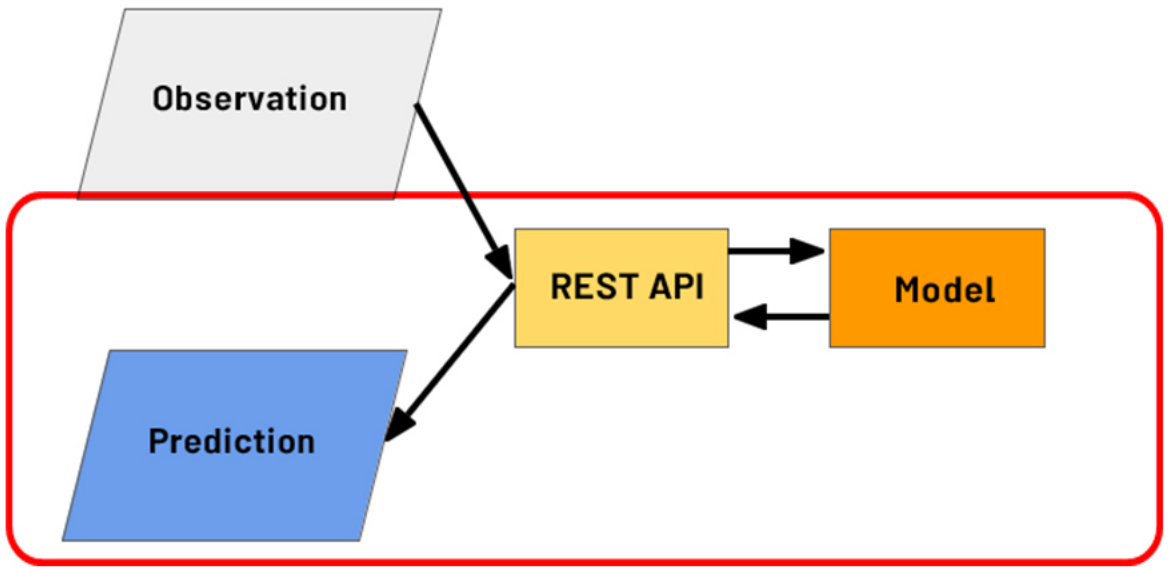
Representation Transformation

- Change data format for the model
- One-hot encoding
- Vectorization

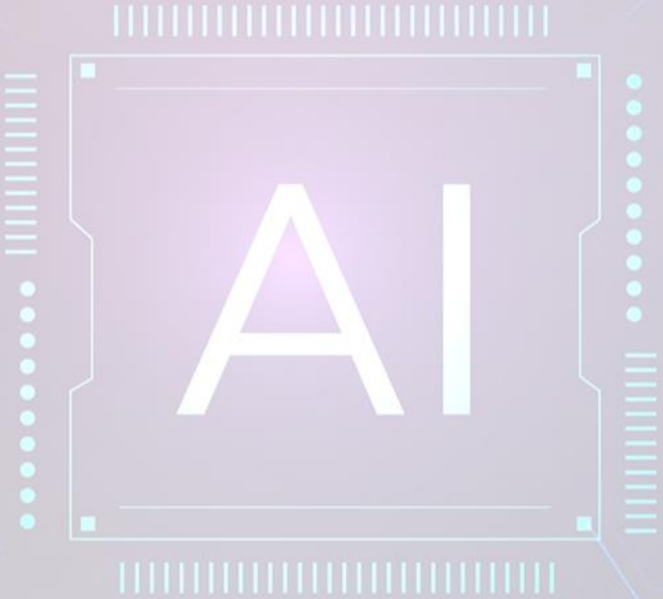
Feature Selection

- Apply same feature selection done during training on incoming data and features fetched from cache

Processing After Obtaining Predictions

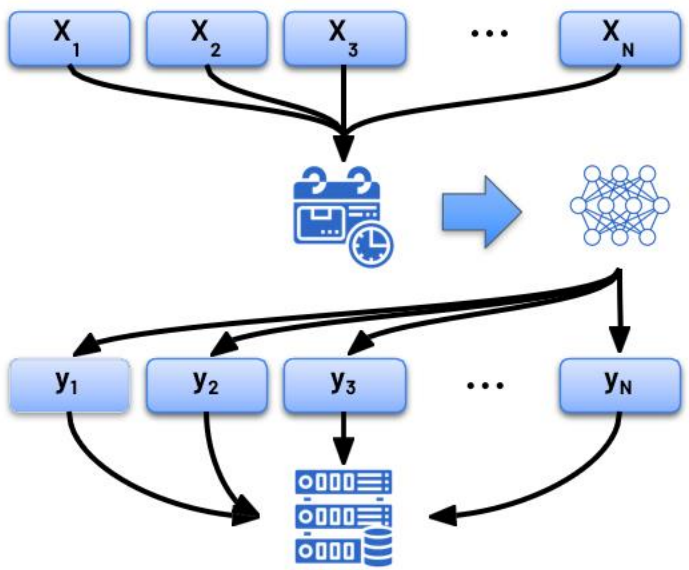


Batch Inference



Batch Inference

- Generating predictions on batch of a observations
- Batch jobs are often generated on some recurring schedule
- Predictions are stored and made available to developers or end users



Advantages of Batch Inference

- Complex machine learning models for improved accuracy

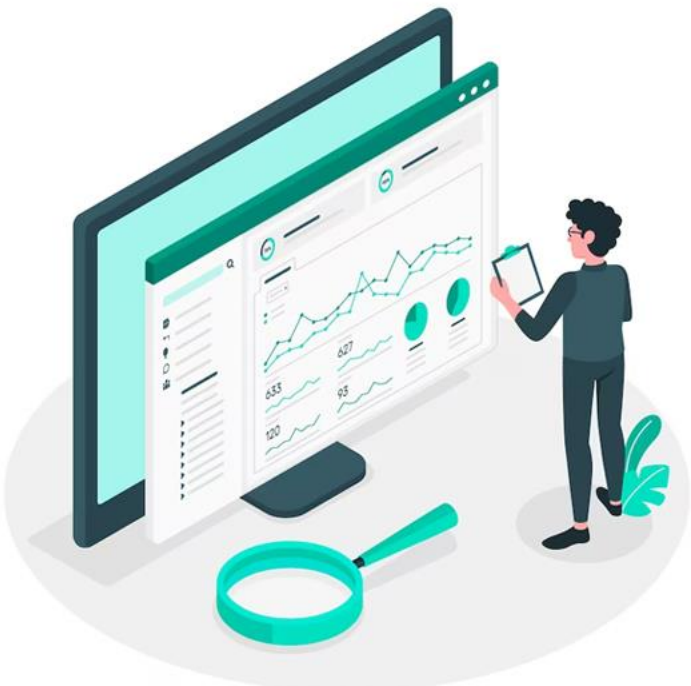
- Caching not required

  - Reduced cost of ML system

  - Longer data retrieval
  - Not a problem as predictions are not in real time

Limitations of Batch Inference

- Long Update Latency
- Predictions based on older data
  - Recommendations from same age bracket
  - Recommendations from same geolocation



Limitations of Batch Inference

Important Metrics to Optimize

- Most important metric to optimize while performing batch predictions:

Throughput

- Prediction service should be able to handle large volumes of inferences at a time.
- Predictions need not be available immediately.
- Latency can be compromised.

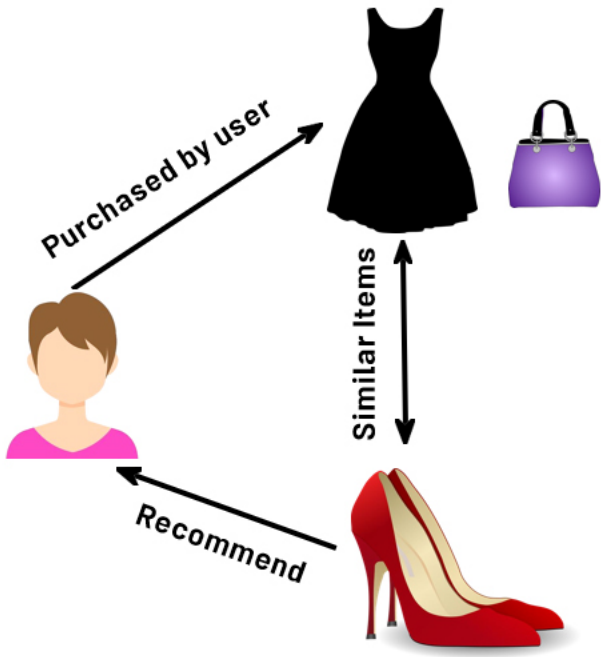
Limitations of Batch Inference

How to Increase Throughput?

- Use hardware accelerators like GPU's, TPU's.
- Increase number of servers/workers
  - Load several instances of model on multiple workers to increase throughput

Use Case - Product Recommendations

- E-commerce sites: new recommendations on a recurring schedule
- Cache these for easy retrieval
- Enables use of more predictors to train more complex models.
  - Helps personalization to a greater degree, but with delayed data



Use Case - Sentiment Analysis

- User sentiment based on customer reviews
- No need for realtime prediction for this problem
- CNN, RNN, or LSTM all work for this problem
- These models are more complex, but more accurate for the problem
- More cost effective to use them with batch prediction

My experience so far has been fantastic

POSITIVE

The product is ok I guess

NEUTRAL

Your support team is pathetic

NEGATIVE

Use Case - Demand Forecasting

- Estimate the demand for products for inventory and ordering optimization
- Predict future based on historical data (time series)
- Many models available as this is a batch predictions problem

Data Processing - Batch and Streaming

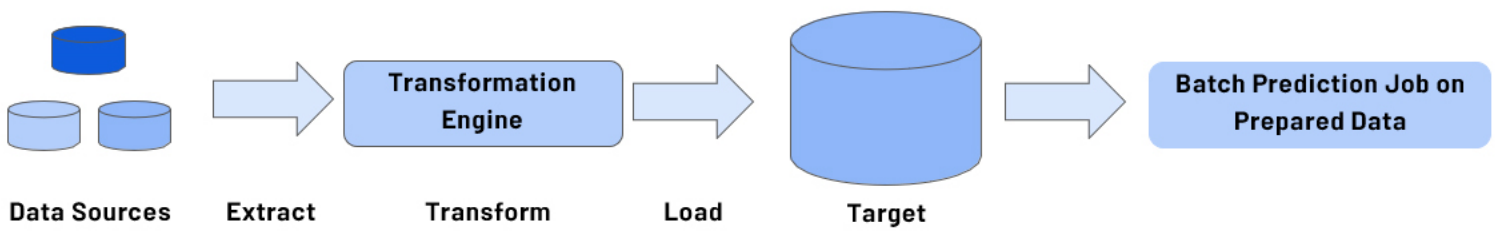
- Data can be of different types based on the source.
- Batch Data
  - Batch processing can be done on data available in huge volumes in data lakes, from csv fies, log files etc.,
- Streaming Data
  - Real-time streaming data, like data from sensors.

ETL on Data

- **Before data is used for making batch predictions:**
  - It has to be extracted from multiple sources like log files, streaming sources, APIs, apps etc.
  - Transformed
  - Loaded into a database for prediction
- **This is done using ETL Pipelines**

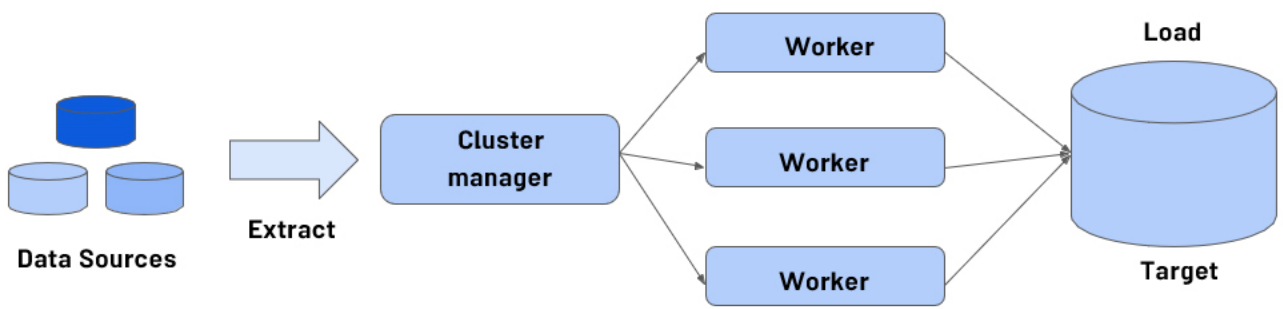
ETL Pipelines

- Set of processes for
  - extracting data from data sources
  - Transforming data
  - Loading into an output destination like data warehouse
- From there data can be consumed for training or making predictions using ML models



Distributed Processing

- ETL can be performed huge volumes of data in distributed manner.
- Data is split into chunks and parallely processed by multiple workers.
- The results of the ETL workflow are stored in a database.
- Results in lower latency and higher throughput of data processing.



ETL Pipeline components Batch Processing