UNIT - 4

Big Data Analytics

Outline

- Big Data analytics approaches
- Approaches for clustering big data
- Approaches for classification of big data
- Recommendation Systems

Big Data

- Big data is defined as **collections of data sets** whose **volume, velocity in terms of time variation, or variety** is so large that it is difficult to **store, manage, process and analyze the data** using traditional databases and data processing tools.
- **Characteristics of big data:**
  - **Volume**
    - Though there is no fixed threshold for the volume of data to be considered as big data, however, typically, the term **big data is used for massive scale data** that is difficult to store, manage and process using traditional databases and data processing architectures. The volumes of data generated by modern IT, industrial, healthcare and systems is growing exponentially driven by the lowering costs of data storage and processing architectures and the need to extract valuable insights from the data to improve business processes, efficiency and service to consumers.
  - **Velocity**
    - Velocity is another important characteristic of big data and the primary reason for exponential growth of data. Velocity of data refers to how fast the data is generated. Modern IT, industrial and other systems are generating data at increasingly higher speeds generating big data.
  - **Variety**
    - Variety refers to the forms of the data. Big data comes in different forms such as structured or unstructured data, including text data, image, audio, video and sensor data.

Clustering Big Data

- Clustering is the **process of grouping similar data items together** such that data items that are more similar to each other (with respect to some similarity criteria) than other data items are put in one cluster.
- Clustering big data is of much interest, and happens in applications such as:
    - Clustering social network data to find a group of similar users
    - Clustering electronic health record (EHR) data to find similar patients.
    - Clustering sensor data to group similar or related faults in a machine
    - Clustering market research data to group similar customers
    - Clustering clickstream data to group similar users
- Clustering is achieved by **clustering algorithms** that belong to a **broad category algorithms** called **unsupervised machine learning**.
- Unsupervised machine learning algorithms find the **patterns and hidden structure** in data for which no training data is available.

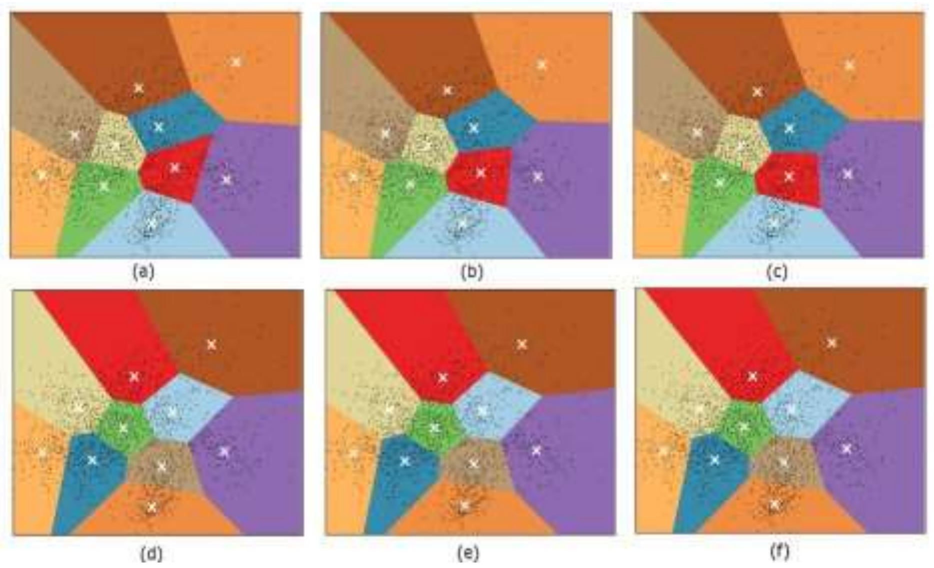
k-means Clustering

- k-means is a clustering algorithm that groups data items into k clusters, where k is user defined.
- Each cluster is defined by a centroid point.
- k-means clustering begins with a set of k centroid points which are either randomly chosen from the dataset or chosen using some initialization algorithm such as canopy clustering.
- The algorithm proceeds by finding the distance between each data point in the data set and the centroid points.
- Based on the distance measure, each data point is assigned to a cluster belonging to the closest centroid.
- In the next step the centroids are recomputed by taking the mean value of all the data points in a cluster.
- This process is repeated till the centroids no longer move more than a specified threshold.

k-means Clustering

k-means Clustering Algorithm

```
Start with k centroid points
while the centroids no longer move beyond a threshold or maximum number of iterations reached: for each
        point in the dataset:
                for each centroid:
                        find the distance between the point and the centroid
                        assign the point to the cluster belonging to the nearest centroid for each
                cluster:
                        recompute the centroid point by taking mean value of all points in the cluster
```



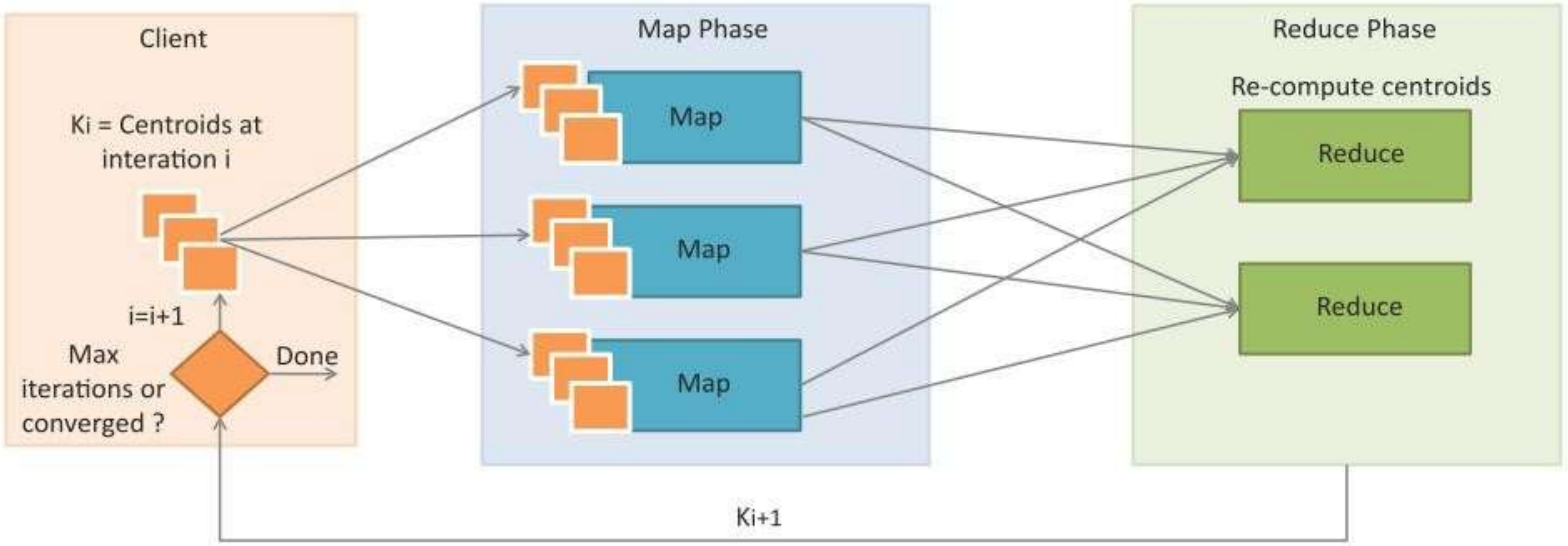
Example of clustering 300 points with k-means: (a) iteration 1, (b) iteration 2, (c) iteration 3, (d) iteration 5, (e) iteration 10, (f) iteration 100.

Clustering Documents with k-means

- Document clustering is the most commonly used application of k-means clustering algorithm.
- Document clustering problem occurs in many big data applications such as finding similar news articles, finding similar patients using electronic health records, etc.
- Before applying k-means algorithm for document clustering, the documents need to be vectorized. Since documents contain textual information, the process of vectorization is required for clustering documents.
- The process of generating document vectors involves several steps:
  - A dictionary of all words used in the tokenized records is generated. Each word in the dictionary has a dimension number assigned to it which is used to represent the dimension the word occupies in the document vector.
  - The number of occurrences or term frequency (TF) of each word is computed.
  - Inverse Document Frequency (IDF) for each word is computed. Document Frequency (DF) for a word is the number of documents (or records) in which the word occurs.
  - Weight for each word is computed. The term weight Wi is used in the document vector as the value for the dimension-i.
  - Similarity between documents is computed using a distance measure such as Euclidean distance measure.

k-means with MapReduce

- The data to be clustered is distributed on a distributed file system such as HDFS and split into blocks which are replicated across different nodes in the cluster.
- Clustering begins with an initial set of centroids. The client program controls the clustering process.
- In the Map phase, the distances between the data samples and centroids are calculated and each sample is assigned to the nearest centroid.
- In the Reduce phase, the centroids are recomputed using the mean of all the points in each cluster.
- The new centroids are then fed back to the client which checks whether convergence is reached or maximum number of iterations are completed.

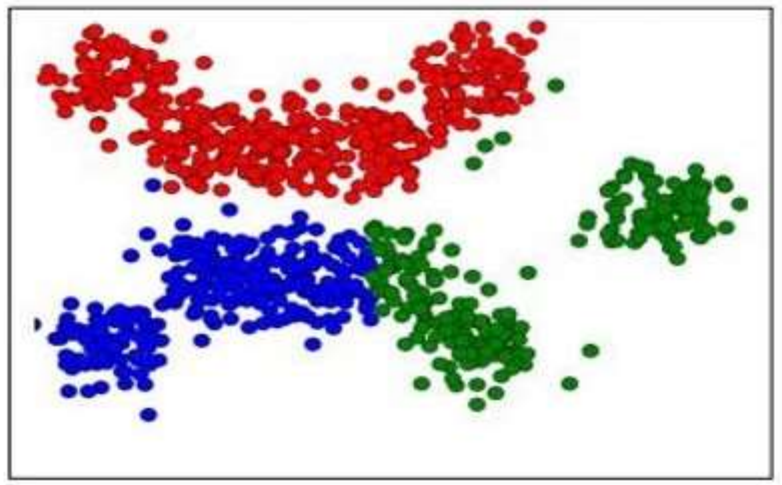


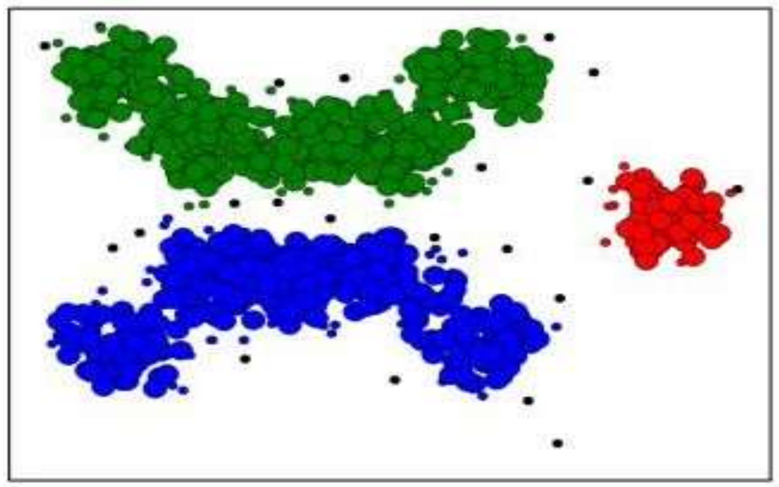
DBSCAN clustering

- DBSCAN is a density clustering algorithm that works on the notions of density reachability and density connectivity.
- **Density Reachability**
    - Is defined on the basis of *Eps*-neighborhood, where *Eps*-neighborhood means that for every point *p* in a cluster *C* there is a point *q* in *C* so that *p* is inside of the *Eps*-neighborhood of *q* and there are at least a minimum number (*MinPts*) of points in an *Eps*-neighborhood of that point.
    - A point *p* is called directly density-reachable from a point *q* if it is not farther away than a given distance (*Eps*) and if it is surrounded by at least a minimum number (*MinPts*) of points that may be considered to be part of a cluster.
- **Density Connectivity**
    - A point ***p*** is density connected to a point ***q*** if there is a point ***o*** such that both, ***p*** and ***q*** are density-reachable from ***o*** wrt. ***Eps*** and ***MinPts***.
- **A cluster, is then defined based on the following two properties:**
    - **Maximality:** For all point *p, q* if *p* belongs to cluster *C* and *q* is density-reachable from *p* (wrt. *Eps* and *MinPts*), then *q* also belongs to the cluster *C*.
    - **Connectivity:** For all point *p, q* in cluster *C, p* is density-connected to *q* (wrt. *Eps* and *MinPts*).

DBSCAN vs K-means

- DBSCAN can find irregular shaped clusters as seen from this example and can even find a cluster completely surrounded by a different cluster.
- DBSCAN considers some points as noise and does not assign them to any cluster.


(a) kmeans


(b) DBSCAN

Classification of Big Data

- Classification is the process of categorizing objects into predefined categories.
- Classification is achieved by classification algorithms that belong to a broad category of algorithms called supervised machine learning.
- Supervised learning involves inferring a model from a set of input data and known responses to the data (training data) and then using the inferred model to predict responses to new data.
- **Binary classification**
  - Binary classification involves categorizing the data into two categories. For example, classifying the sentiment of a news article into positive or negative, classifying the state of a machine into good or faulty, classifying the heath test into positive or negative, etc.
- **Multi-class classification**
  - Multi-class classification involves more than two classes into which the data is categorized. For example, gene expression classification problem involves multiple classes.
- **Document classification**
  - Document classification is a type of multi-class classification approach in which the data to the classified is in the form of text document. For classifying news articles into different categories such as politics, sports, etc.

Performance of Classification Algorithms

- **Precision:** Precision is the fraction of objects that are classified correctly.

$$Precision = \frac{TruePositive}{(TruePositive + FalsePositive)}$$

- **Recall:** Recall is the fraction of objects belonging to a category that are classified correctly.

$$Recall = \frac{TruePositive}{(TruePositive + FalseNegative)}$$

- **Accuracy:**

$$Accuracy = \frac{(TruePositive + TrueNegative)}{(TruePositive + TrueNegative + FalsePositive + FalseNegative)}$$

- **F1-score:** F1-score is a measure of accuracy that considers both precision and recall. F1-score is the harmonic means of precision and recall given as,

$$F1 - Score = \frac{2(Precision)(Recall)}{(Precision + Recall)}$$

Naive Bayes

- Naive Bayes is a probabilistic classification algorithm based on the Bayes theorem with a naive assumption about the independence of feature attributes. Given a class variable C and feature variables F1,...,Fn , the conditional probability (posterior) according to Bayes theorem is given as,

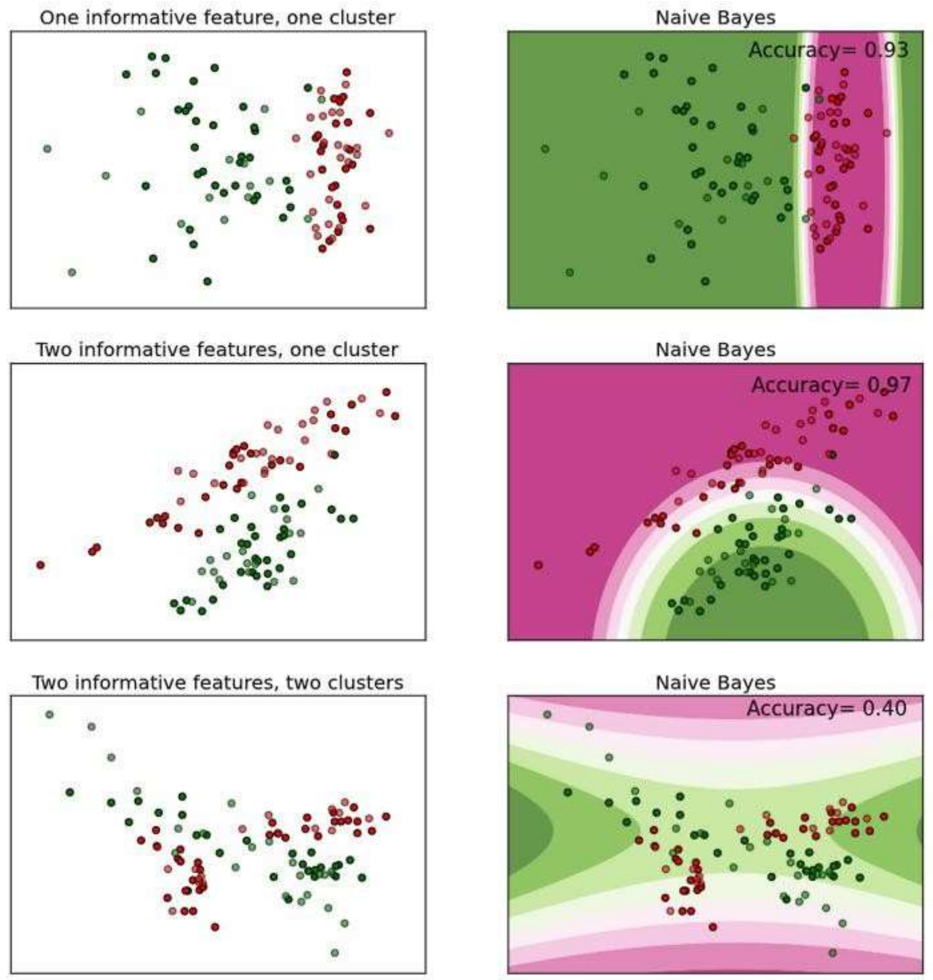
$$P(C|F_1,...,F_n) = \frac{P(F_1,...,F_n|C)P(C)}{P(F_1,...,F_n)}$$

- where, P(C|F1,...,Fn) is the posterior probability, P(F1,...,Fn |C) is the likelihood and P(C) is the prior probability and P(F1,...,Fn) is the evidence. Naive Bayes makes a naïve assumption about the independence every pair of features given as,

$$P(F_1,...,F_n|C) = \prod_{i=1}^{n} P(F_i|C)$$

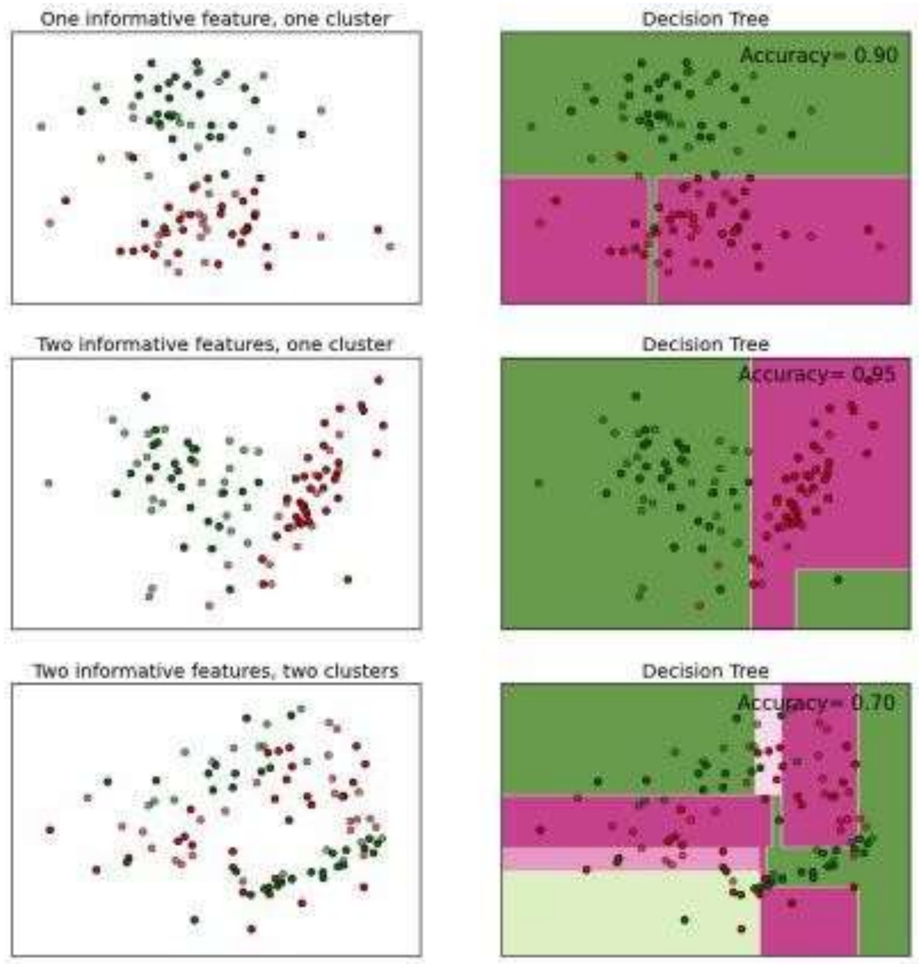
- Since the evidence P(F1,...,Fn) is constant for a given input and does not depend on the class variable C, only the numerator of the posterior probability is important for classification.
- With this simplification, classification can then be done as follows,

$$C = argmax_C P(C) \prod_{i=1}^{n} P(F_i|C)$$



Decision Trees

- Decision Trees are a supervised learning method that use a tree created from simple decision rules learned from the training data as a predictive model.
- The predictive model is in the form of a tree that can be used to predict the value of a target variable based on a several attribute variables.
- Each node in the tree corresponds to one attribute in the dataset on which the “split” is performed.
- Each leaf in a decision tree represents a value of the target variable.
- The learning process involves recursively splitting on the attributes until all the samples in the child node have the same value of the target variable or splitting further results in no further information gain.
- To select the best attribute for splitting at each stage, different metrics can be used.



Splitting Attributes in Decision Trees

To select the best attribute for splitting at each stage, different metrics can be used such as:

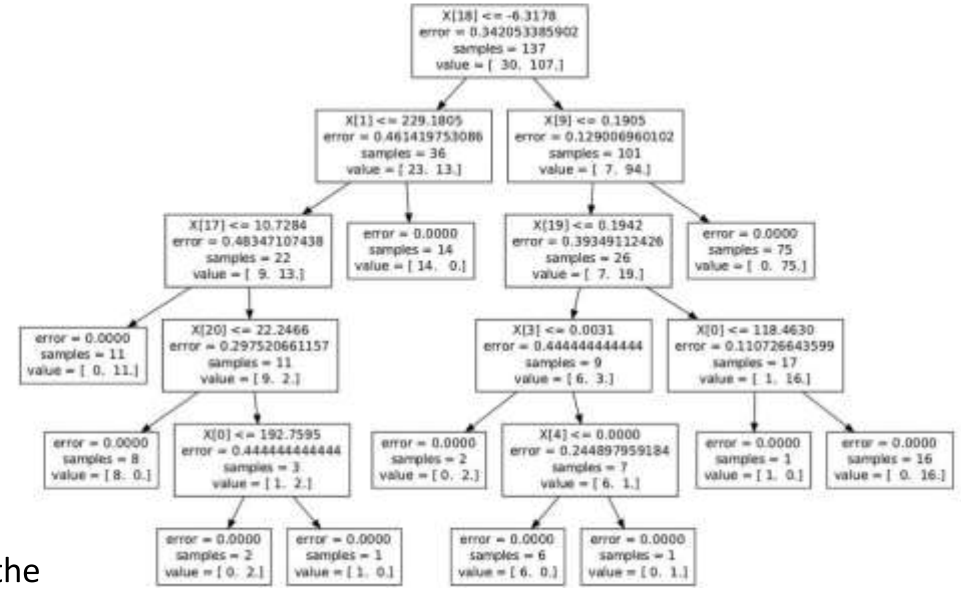
- Information Gain
    - Information content of a discrete random variable X with probability mass function (PMF), P(X), is defined as,

$$I(X) = -\log_2 P(X)$$

    - Information gain is defined based on the entropy of the random variable which is defined as,

$$H(X) = E[I(X)] = E[-\log_2 P(X)] = -\sum_i \log_2 P(x_i)$$

    - Entropy is a measure of uncertainty in a random variable and choosing the attribute with the highest information gain results in a split that reduces the uncertainty the most at that stage.
- Gini Coefficient
    - Gini coefficient measures the inequality, i.e. how often a randomly chosen sample that is labeled based on the distribution of labels, would be labeled incorrectly. Gini coefficient is defined as, $G(X) = 1 - \sum_i P(x_i)^2$



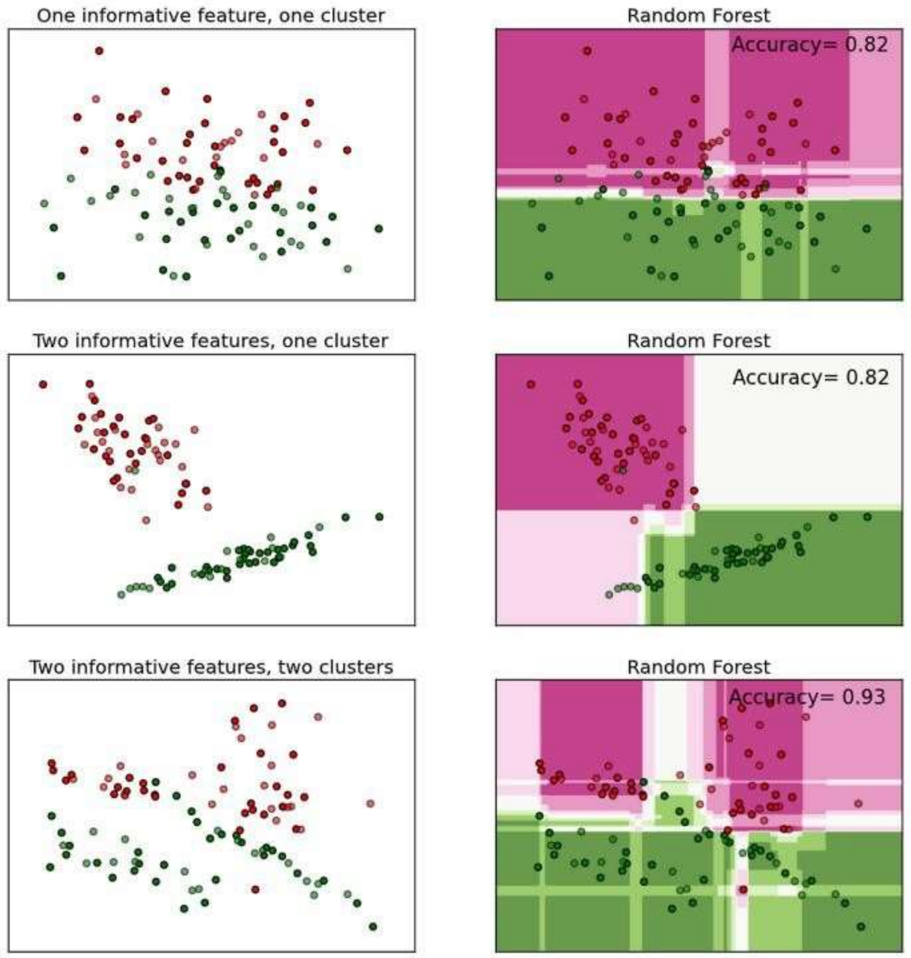
X[18] <= -6.3178 error = 0.342053385902 samples = 137 value = [30. 107.]
X[1] <= 229.1805 error = 0.461419753086 samples = 36 value = [23. 13.]
X[9] <= 0.1905 error = 0.129006960102 samples = 101 value = [7. 94.]
X[17] <= 10.7284 error = 0.48347107438 samples = 22 value = [9. 13.]
error = 0.0000 samples = 14 value = [14. 0.]
X[19] <= 0.1942 error = 0.39349112426 samples = 26 value = [7. 19.]
error = 0.0000 samples = 75 value = [0. 75.]
error = 0.0000 samples = 11 value = [0. 11.]
X[20] <= 22.2466 error = 0.297520661157 samples = 11 value = [9. 2.]
X[3] <= 0.0031 error = 0.444444444444 samples = 9 value = [6. 3.]
X[0] <= 118.4630 error = 0.110726643599 samples = 17 value = [1. 16.]
error = 0.0000 samples = 8 value = [8. 0.]
X[0] <= 192.7595 error = 0.444444444444 samples = 3 value = [1. 2.]
error = 0.0000 samples = 2 value = [0. 2.]
X[4] <= 0.0000 error = 0.244897959184 samples = 7 value = [6. 1.]
error = 0.0000 samples = 1 value = [1. 0.]
error = 0.0000 samples = 16 value = [0. 16.]
error = 0.0000 samples = 2 value = [0. 2.]
error = 0.0000 samples = 1 value = [1. 0.]
error = 0.0000 samples = 6 value = [6. 0.]
error = 0.0000 samples = 1 value = [0. 1.]

Decision Tree Algorithms

- There are different algorithms for building decisions trees, popular ones being ID3 and C4.5.
- **ID3:**
  - Attributes are discrete. If not, discretize the continuous attributes.
  - Calculate the entropy of every attribute using the dataset.
  - Choose the attribute with the highest information gain.
  - Create branches for each value of the selected attribute.
  - Repeat with the remaining attributes.
- The ID3 algorithm can be result in over-fitting to the training data and can be expensive to train especially for continuous attributes.
- C4.5
  - The C4.5 algorithm is an extension of the ID3 algorithm. C4.5 supports both discrete and continuous attributes.
  - To support continuous attributes, C4.5 finds thresholds for the continuous attributes and then splits based on the threshold values. C4.5 prevents over-fitting by pruning trees after they have been created.
  - Pruning involves removing or aggregating those branches which provide little discriminatory power.

Random Forest

- Random Forest is an ensemble learning method that is based on randomized decision trees.
- Random Forest trains a number decision trees and then takes the majority vote by using the mode of the class predicted by the individual trees.

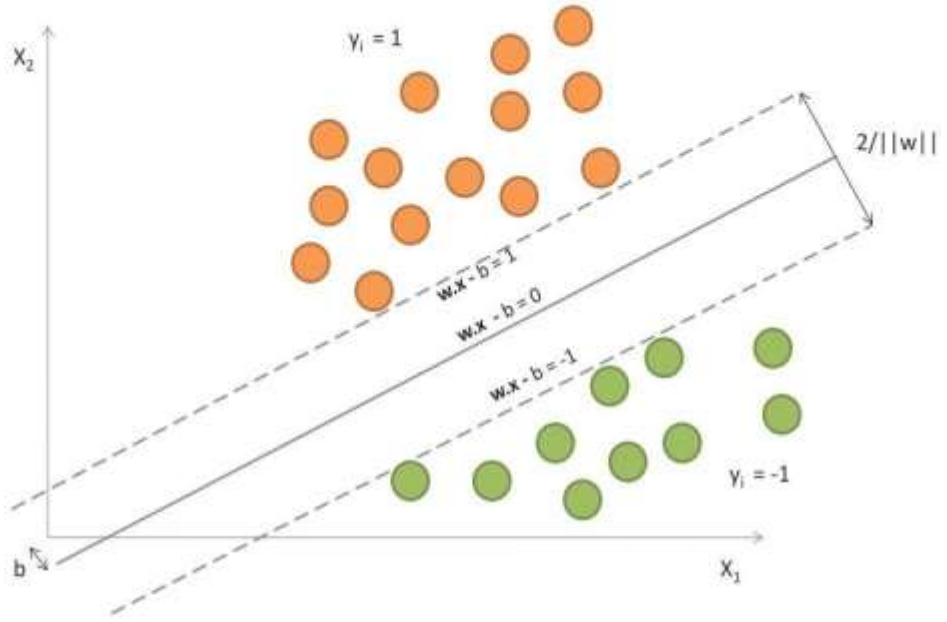


Breiman’s Algorithm

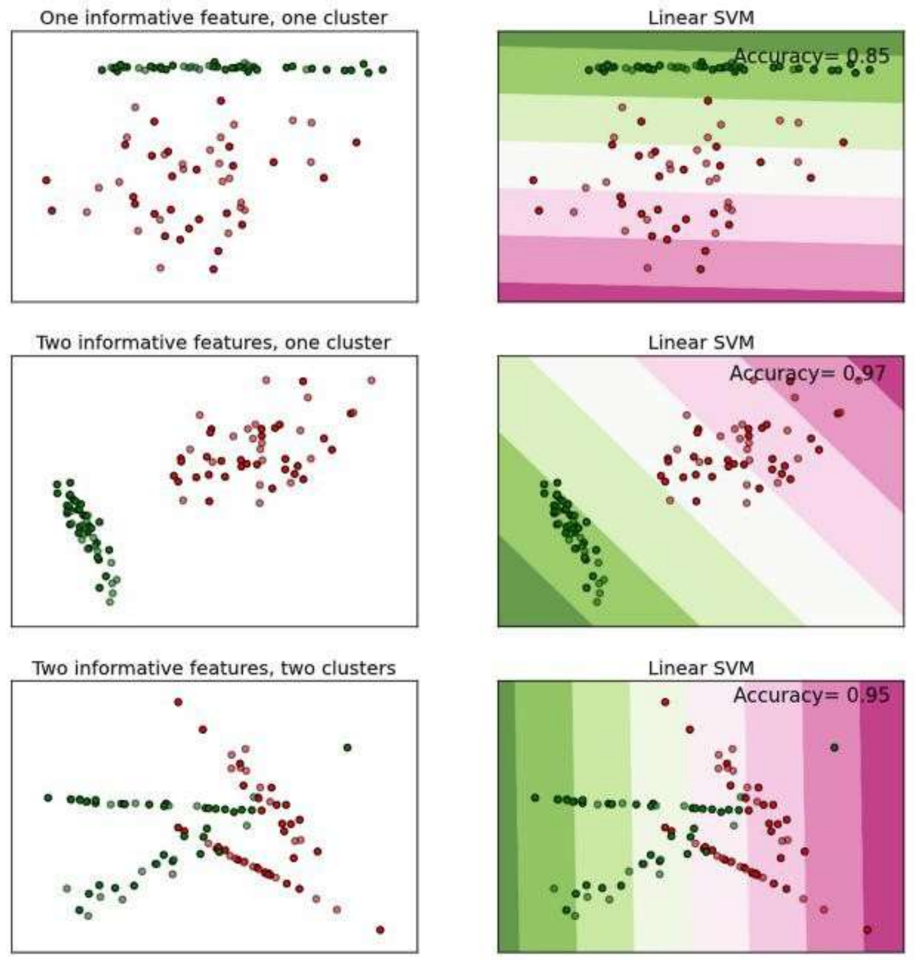
1. Draw a bootstrap sample (n times with replacement from the N samples in the training set) from the dataset
2. Train a decision tree
    - Until the tree is fully grown (maximum size)
    - Choose next leaf node
    - Select m attributes (m is much less than the total number of attributes M) at random.
    - Choose the best attribute and split as usual
3. Measure out-of-bag error
    - Use the rest of the samples (not selected in the bootstrap) to estimate the error of the tree, by predicting their classes.
4. Repeat steps 1-3 k times to generate k trees.
5. Make a prediction by majority vote among the k trees

Support Vector Machine

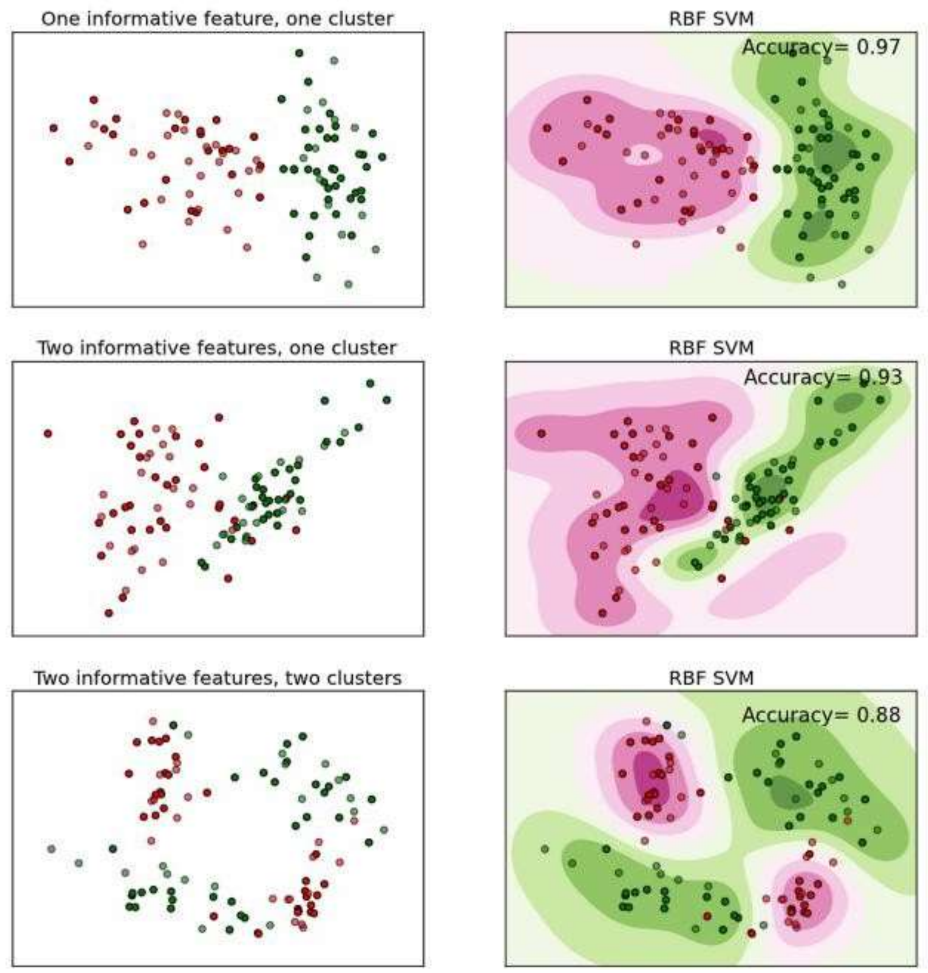
- Support Vector Machine (SVM) is a supervised machine learning approach used for classification and regression.
- The basic form is SVM is a binary classifier that classifies the data points into one of the two classes.
- SVM training involves determining the maximum margin hyperplane that separates the two classes.
- The maximum margin hyperplane is one which has the largest separation from the nearest training data point.
- Given a training data set (x_i ,y_i) where x_i is an n dimensional vector and y_i = 1 if x_i is in class 1 and y_i = -1 if x_i is in class 2.
- A standard SVM finds a hyperplane **w.x**-b = 0, which correctly separates the training data points and has a maximum margin which is the distance between the two hyperplanes **w.x**-b = 1 and **w.x**-b = -1



Support Vector Machine



Binary classification with Linear SVM



Binary classification with RBF SVM

Recommendation Systems

- Recommendation systems are an important part of modern cloud applications such as e-Commerce, social networks, content delivery networks, etc.
- Item-based or Content-based Recommendation
  - Provides recommendations to users (for items such as books, movies, songs, or restaurants) for unrated items based on the characteristics of the item.
- Collaborative Filtering
  - Provides recommendations based on the ratings given by the user and other users to similar items.

Multimedia Cloud

Outline

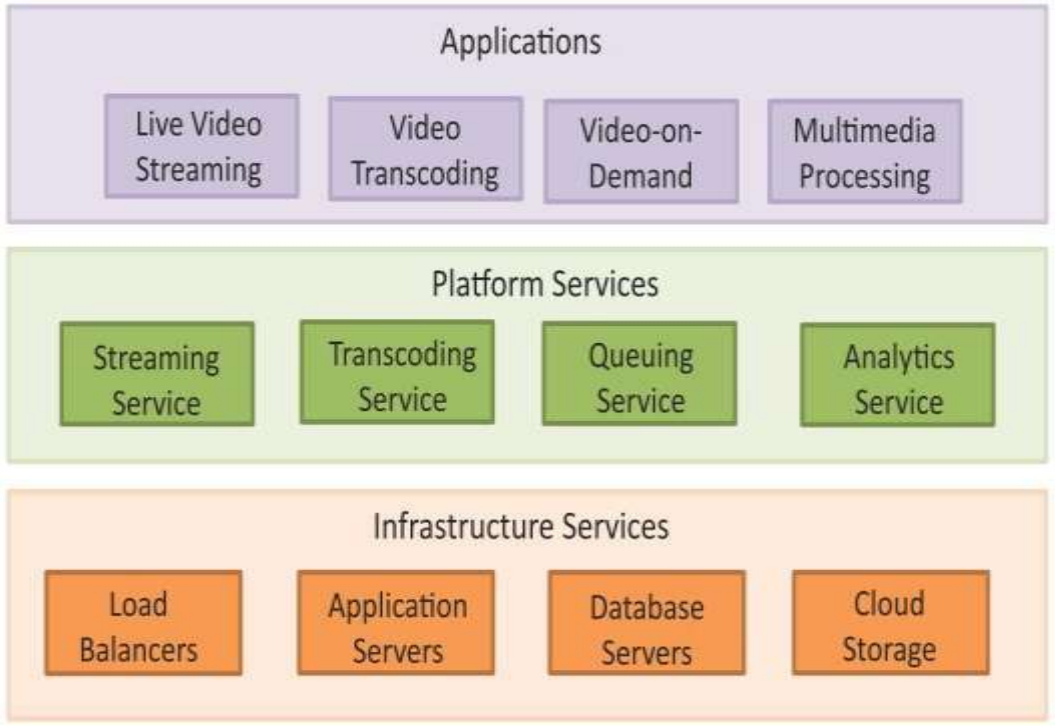
- Reference architecture for Multimedia Cloud
- Case study of a live video streaming cloud application
- Case study of a video transcoding cloud application

Design methodology for PaaS service model

- For applications that use the Platform-as-a-service (PaaS) cloud service model, the architecture and deployment design steps are not required since the platform takes care of the architecture and deployment.
- **Component Design**
  - In the component design step, the developers have to take into consideration the platform specific features.
- **Platform Specific Software**
  - Different PaaS offerings such as Google App Engine, Windows Azure Web Sites, etc., provide platform specific software development kits (SDKs) for developing cloud applications.
- **Sandbox Environments**
  - Applications designed for specific PaaS offerings run in sandbox environments and are allowed to perform only those actions that do not interfere with the performance of other applications.
- **Deployment & Scaling**
  - The deployment and scaling is handled by the platform while the developers focus on the application development using the platform-specific SDKs.
- **Portability**
  - Portability is a major constraint for PaaS based applications as it is difficult to move the

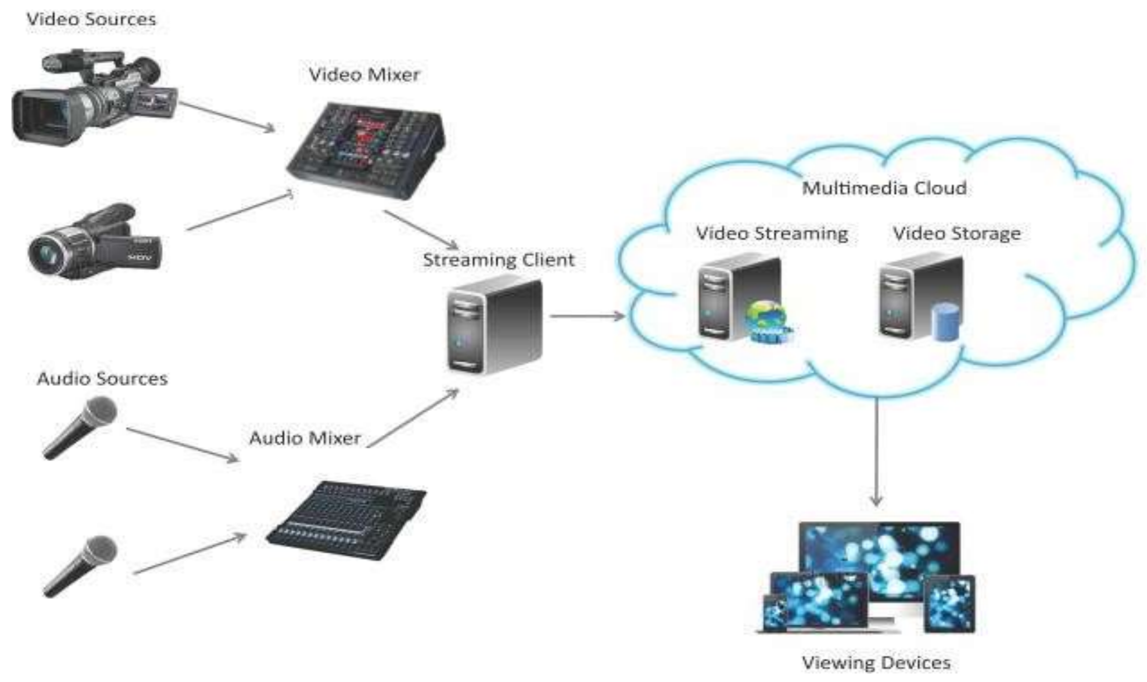
Multimedia Cloud Reference Architecture

- **Infrastructure Services**
  - In the Multimedia Cloud reference architecture, the first layer is the infrastructure services layer that includes computing and storage resources.
- **Platform Services**
  - On top of the infrastructure services layer is the platform services layer that includes frameworks and services for streaming and associated tasks such as transcoding and analytics that can be leveraged for rapid development of multimedia applications.
- **Applications**
  - The topmost layer is the applications such as live video streaming, video transcoding, video-on-demand, multimedia processing etc.
  - Cloud-based multimedia applications alleviates the burden of installing and maintaining multimedia applications locally on the multimedia consumption devices (desktops, tablets, smartphone, etc) and provide access to rich multimedia content.
- **Service Models**
  - A multimedia cloud can have various service models such as IaaS, PaaS and SaaS that offer infrastructure, platform or application services.



Multimedia Cloud - Live Video Streaming

- Workflow of a live video streaming application that uses multimedia cloud:
  - The video and audio feeds generated by a number cameras and microphones are mixed/multiplexed with video/audio mixers and then encoded by a client application which then sends the encoded feeds to the multimedia cloud.
  - On the cloud, streaming instances are created on-demand and the streams are then broadcast over the internet.
  - The streaming instances also record the event streams which are later moved to the cloud storage for video archiving.



Workflow for live video streaming using multimedia cloud

Streaming Protocols

- **RTMP Dynamic Streaming (Unicast)**
  - High-quality, low-latency media streaming with support for live and on-demand and full adaptive bitrate.
- **RTMPE (encrypted RTMP)**
  - Real-time encryption of RTMP.
- **RTMFP (multicast)**
  - IP multicast encrypted with support for both ASM or SSM multicast for multicast-enabled network.
- **RTMFP (P2P)**
  - P2P live video delivery between Flash Player clients.
- **RTMFP (multicast fusion)**
  - IP and P2P working together to support higher QoS within enterprise networks.
- **HTTP Dynamic Streaming (HDS)**
  - Enabling on-demand and live adaptive bitrate video streaming of standards-based MP4 media over regular HTTP connections.
- **Protected HTTP Dynamic Streaming (PHDS)**
  - Real-time encryption of HDS.
- **HTTP Live Streaming (HLS)**
  - HTTP streaming to iOS devices or devices that support the HLS format; optional encryption with AES128 encryption standard.

RTMP Streaming

- Real Time Messaging Protocol (RTMP) is a protocol for streaming audio, video and data over the Internet.
- The plain version of RTMP protocol works on top of TCP. RTMPS is a secure variation of RTMP that works over TLS/SSL.
- RTMP provides a bidirectional message multiplex service over a reliable stream transport, such as TCP.
- RTMP maintains persistent TCP connections that allow low-latency communication.
- RTMP is intended to carry parallel streams of video, audio, and data messages, with associated timing information, between a pair of communicating peers.
- Streams are split into fragments so that delivery of the streams smoothly.
- The size of the stream fragments is either fixed or negotiated dynamically between the client and server.
- Default fragment sizes used are 64-bytes for audio data, and 128 bytes for video data.
- RTMP implementations typically assign different priorities to different classes of messages, which can affect the order in which messages are enqueued to the underlying stream transport when transport capacity is constrained.

HTTP Live Streaming

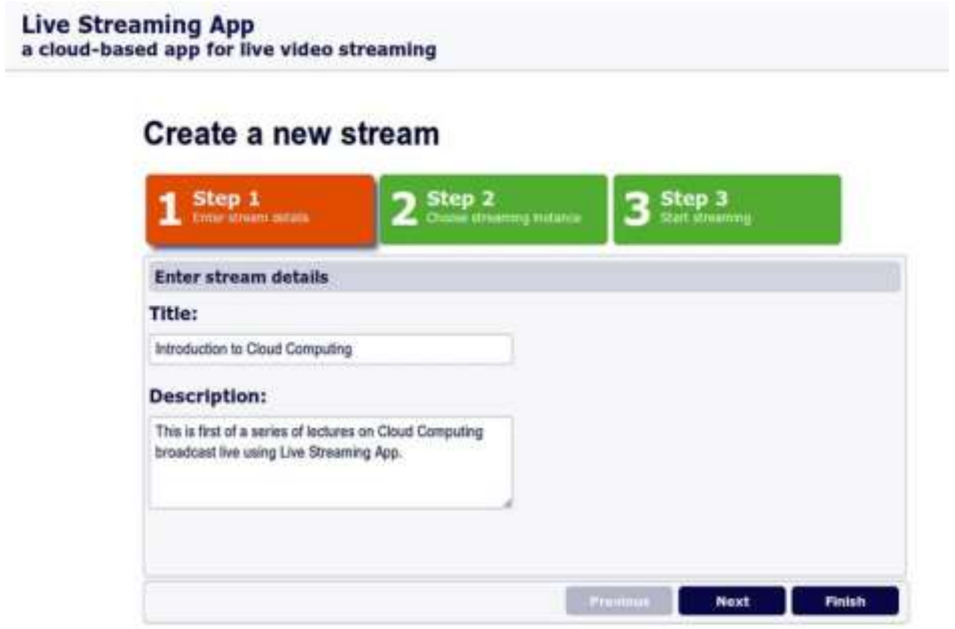
- HTTP Live Streaming (HLS) can dynamically adjust playback quality to match the available speed of wired or wireless networks.
- HLS supports multiple alternate streams at different bit rates, and the client software can switch streams intelligently as network bandwidth changes.
- HLS also provides for media encryption and user authentication over HTTPS, allowing publishers to protect their work.
- The protocol works by splitting the stream into small chunks which are specified in a playlist file.
- Playlist file is an ordered list of media URIs and informational tags.
- The URIs and their associated tags specify a series of media segments.
- To play the stream, the client first obtains the playlist file and then obtains and plays each media segment in the playlist.

HTTP Dynamic Streaming

- HTTP Dynamic Streaming (HDS) enables on-demand and live adaptive bitrate video delivery of standards-based MP4 media (H.264 or VPC) over regular HTTP connections.
- HDS combines HTTP (progressive download) and RTMP (streaming download) to provide the ability to deliver video content in a steaming manner over HTTP.
- HDS supports adaptive bitrate which allows HDS to detect the client’s bandwidth and computer resources and serve content fragments encoded at the most appropriate bitrate for the best viewing experience.
- HDS supports high-definition video up to 1080p, with bitrates from 700 kbps up to and beyond 6 Mbps, using either H.264 or VP6 video codecs, or AAC and MP3 audio codecs.
- HDS allows leveraging existing caching infrastructures, content delivery networks (CDNs) and standard HTTP server hardware to deliver on-demand and live content.

Live Video Steaming App – Case Study

- **Functionality**
  - Live video streaming application allows on-demand creation of video streaming instances in the cloud.
- **Development**
  - The live streaming application is created using the Django framework and uses Amazon EC2 cloud instances.
  - For video stream encoding and publishing, the Adobe Flash Media Live Encoder and Flash Media Server are used



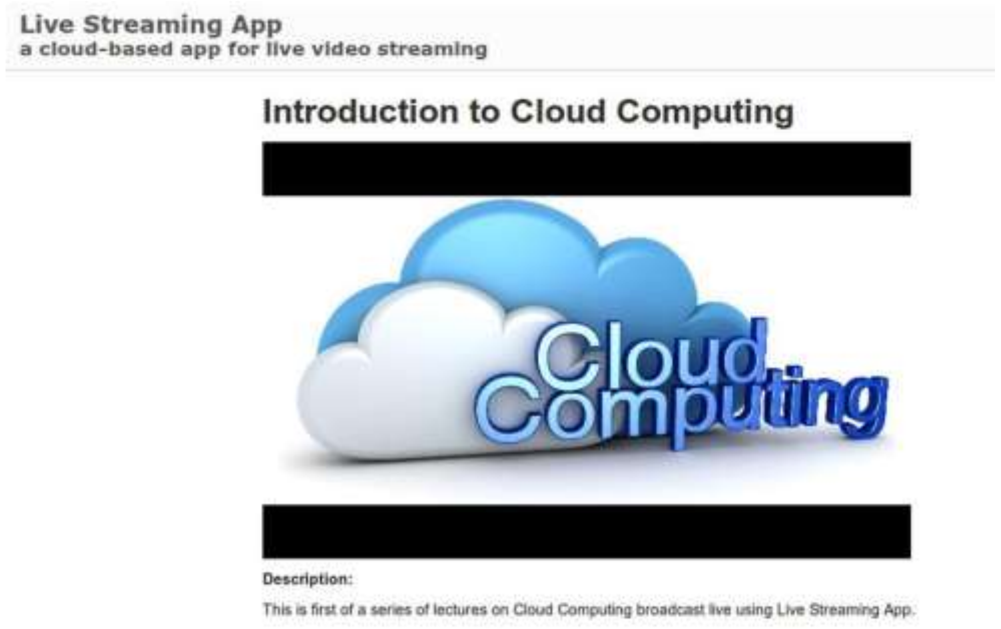
Live Streaming App
a cloud-based app for live video streaming

Create a new stream

Enter stream details

Title:
Introduction to Cloud Computing

Description:
This is first of a series of lectures on Cloud Computing broadcast live using Live Streaming App.



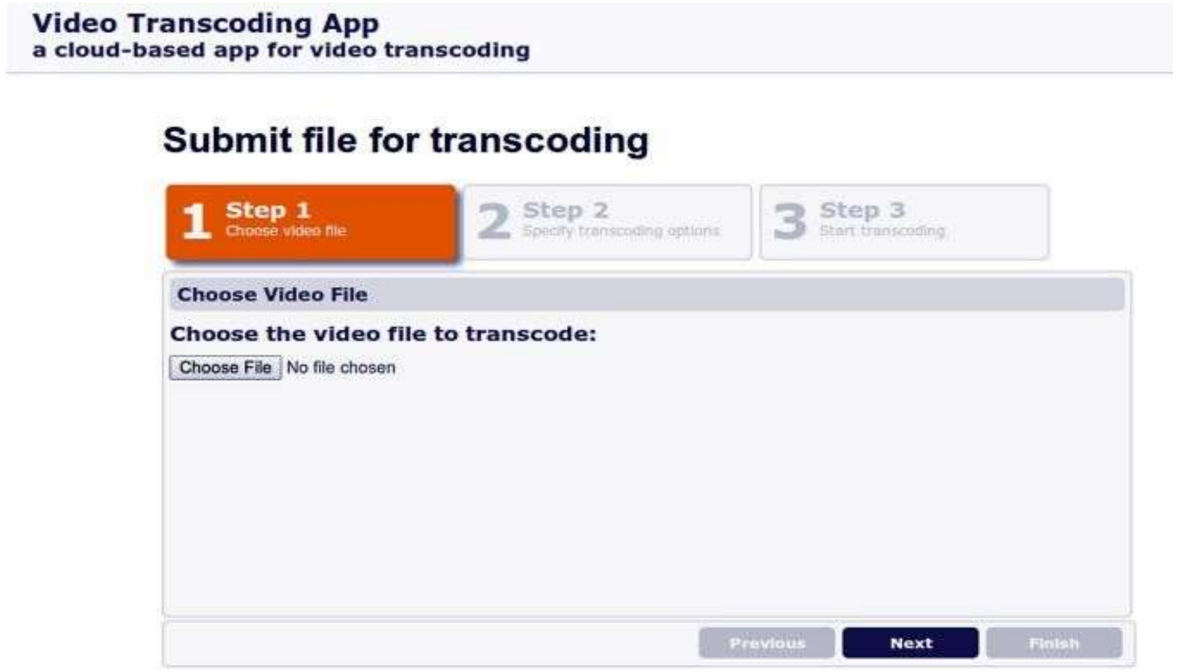
Live Streaming App
a cloud-based app for live video streaming

Introduction to Cloud Computing

Description:
This is first of a series of lectures on Cloud Computing broadcast live using Live Streaming App.

Video Transcoding App – Case Study

- **Functionality**
  - Video transcoding application is based on multimedia cloud.
  - The transcoding application allows users to upload video files and choose the conversion presets.
- **Development**
  - The application is built upon the Amazon Elastic Transcoder.
  - Elastic Transcoder is highly scalable, relatively easy to use service from Amazon that allows converting video files from their source format into versions that will playback on mobile devices like smartphones, tablets and PCs.


Video Transcoding App
a cloud-based app for video transcoding

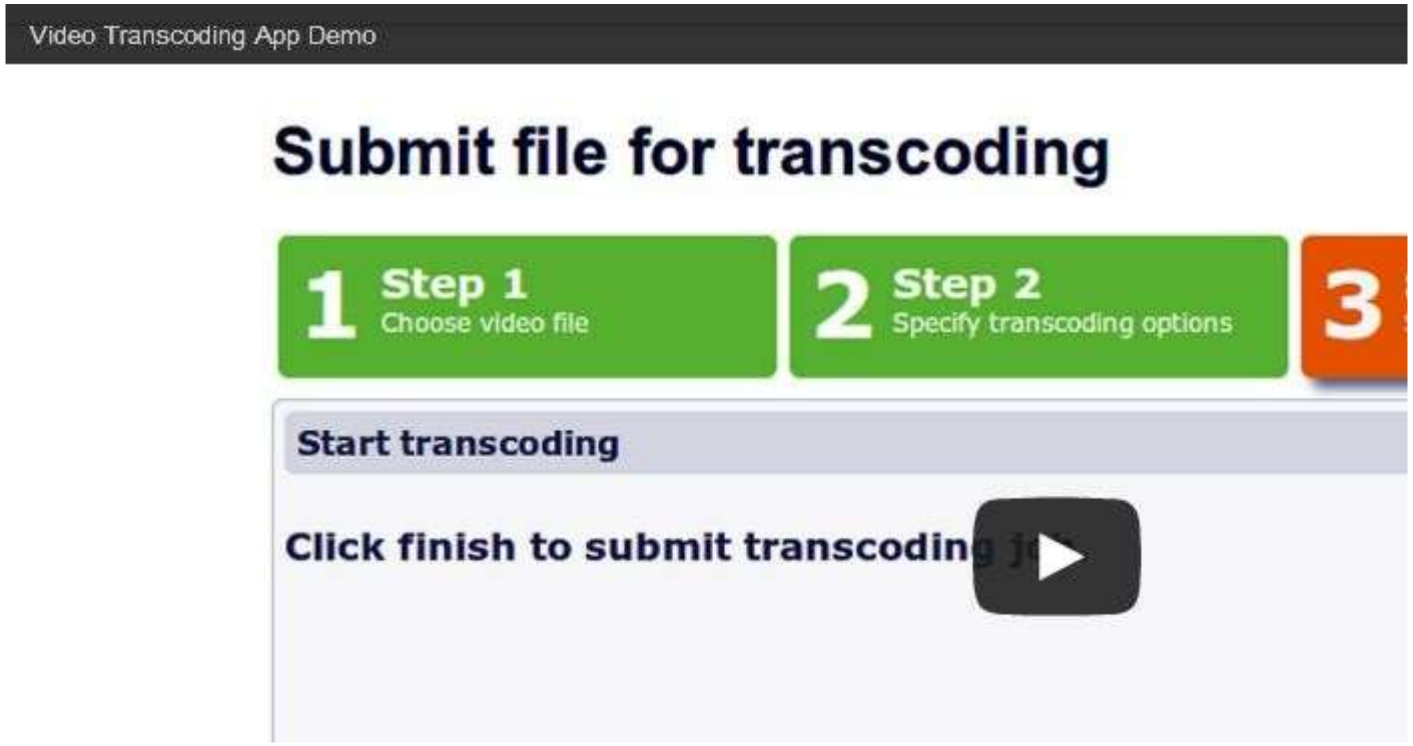
Submit file for transcoding

Choose Video File

Choose the video file to transcode:

No file chosen
1 Step 1 Choose video file
2 Step 2 Specify transcoding options
3 Step 3 Start transcoding
Choose File
Previous
Next
Finish

Video Transcoding App – Demo



Video Transcoding App Demo

Submit file for transcoding

1 Step 1
Choose video file

2 Step 2
Specify transcoding options

3

Start transcoding

Click finish to submit transcoding j

Cloud Application Benchmarking & Tuning

Outline

- Cloud application workload characteristics
- Performance metrics for cloud applications
- Cloud application testing
- Performance testing tools
- Load test and bottleneck detection case study

Benchmarking

- Benchmarking of cloud applications is important or the following reasons:
  - **Provisioning and capacity planning**
    - The process of provisioning and capacity planning for cloud applications involves determining the amount of computing, memory and network resources to provision for the application.
    - Benchmarking can help in comparing alternative deployment architectures and choosing the best and most cost effective deployment architecture that can meet the application performance requirements.
  - **Ensure proper utilization of resources**
    - Benchmarking can help in determining the utilization of computing, memory and network resources for applications and identify resources which are either under-utilized or over-provisioned and hence save deployments costs.
  - **Market readiness of applications**
    - Performance of an application depends on the characteristics of the workloads it experiences. Different types of workloads can dead to different performance for the same application.
    - To ensure the market readiness of an application it is important to model all types of workloads the application can experience and benchmark the application with such workloads.

Cloud Application Benchmarking - Steps

- **Trace Collection/Generation**
  - The first step in benchmarking cloud applications is to collect/generate traces of real application workloads. For generating a trace of workload, the application is instrumented to log information such as the requests submitted by the users, the time-stamps of the requests, etc.
- **Workload Modeling**
  - Workload modeling involves creation of mathematical models that can be used for generation of synthetic workloads.
- **Workload Specification**
  - Since the workload models of each class of cloud computing applications can have different workload attributes, a Workload Specification Language (WSL) is often used for specification of application workloads. WSL can provide a structured way for specifying the workload attributes that are critical to the performance of the applications. WSL can be used by synthetic workload generators for generating workloads with slightly varying the characteristics.
- **Synthetic Workload Generation**
  - Synthetic workloads are used for benchmarking cloud applications. An important requirement for a synthetic workload generator is that the generated workloads should be representative of the real workloads.

Synthetic Workload Generation Approaches

- **Empirical approach**
  - In this approach traces of applications are sampled and replayed to generate the synthetic workloads.
  - The empirical approach lacks flexibility as the real traces obtained from a particular system are used for workload generation which may not well represent the workloads on other systems with different configurations and load conditions.

- **Analytical approach**
  - Uses mathematical models to define the workload characteristics that are used by a synthetic workload generator.
  - Analytical approach is flexible and allows generation of workloads with different characteristics by varying the workload model attributes.
  - With the analytical approach it is possible to modify the workload model parameters one at a time and investigate the effect on application performance to measure the application sensitivity to different parameters.

User Emulation vs Aggregate Workloads

The commonly used techniques for workload generation are:

- **User Emulation**
  - Each user is emulated by a separate thread that mimics the actions of a user by alternating between making requests and lying idle.
  - The attributes for workload generation in the user emulation method include think time, request types, inter-request dependencies, for instance.
  - User emulation allows fine grained control over modeling the behavioral aspects of the users interacting with the system under test, however, it does not allow controlling the exact time instants at which the requests arrive the system.
- **Aggregate Workload Generation:**
  - Allows specifying the exact time instants at which the requests should arrive the system under test.
  - However, there is no notion of an individual user in aggregate workload generation, therefore, it is not possible to use this approach when dependencies between requests need to be satisfied.
  - Dependencies can be of two types inter-request and data dependencies.
  - An inter-request dependency exists when the current request depends on the previous request, whereas a data dependency exists when the current requests requires input data which is obtained from the response of the previous request.

Workload Characteristics

- **Session**
    - A set of successive requests submitted by a user constitute a session.
- **Inter-Session Interval**
    - Inter-session interval is the time interval between successive sessions.
- **Think Time**
    - In a session, a user submits a series of requests in succession. The time interval between two successive requests is called think time.
- **Session Length**
    - The number of requests submitted by a user in a session is called the session length.
- **Workload Mix**
    - Workload mix defines the transitions between different pages of an application and the proportion in which the pages are visited.

Application Performance Metrics

The most commonly used performance metrics for cloud applications are:

- **Response Time**
  - Response time is the time interval between the moment when the user submits a request to the application and the moment when the user receives a response.

- **Throughput**
  - Throughput is the number of requests that can be serviced per second.

Considerations for Benchmarking Methodology

- **Accuracy**
    - Accuracy of a benchmarking methodology is determined by how closely the generated synthetic workloads mimic the realistic workloads.
- **Ease of Use**
    - A good benchmarking methodology should be user friendly and should involve minimal hand coding effort for writing scripts for workload generation that take into account the dependencies between requests, workload attributes, for instance.
- **Flexibility**
    - A good benchmarking methodology should allow fine grained control over the workload attributes such as think time, inter-session interval, session length, workload mix, for instance, to perform sensitivity analysis.
    - Sensitivity analysis is performed by varying one workload characteristic at a time while keeping the others constant.
- **Wide Application Coverage**
    - A good benchmarking methodology is one that works for a wide range of applications and not tied to the application architecture or workload types.

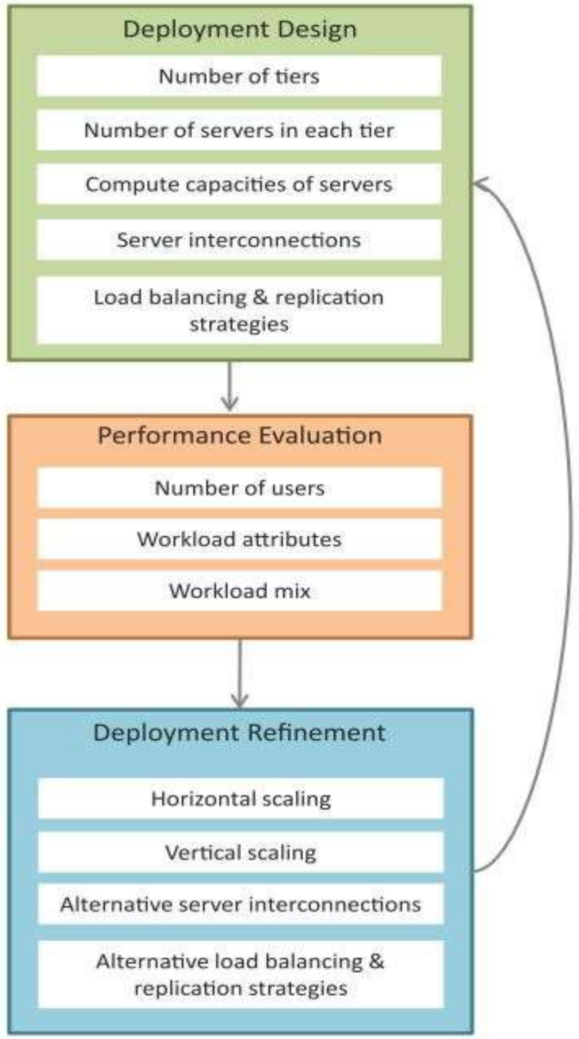
Types of Tests

- **Baseline Tests**
  - Baseline tests are done to collect the performance metrics data of the entire application or a component of the application.
  - The performance metrics data collected from baseline tests is used to compare various performance tuning changes which are subsequently made to the application or a component.
- **Load Tests**
  - Load tests evaluate the performance of the system with multiple users and workload levels that are encountered in the production phase.
  - The number of users and workload mix are usually specified in the load test configuration.
- **Stress Tests**
  - Stress tests load the application to a point where it breaks down.
  - These tests are done to determine how the application fails, the conditions in which the application fails and the metrics to monitor which can warn about impending failures under elevated workload levels.
- **Soak Tests**
  - Soak tests involve subjecting the application to a fixed workload level for long periods of time.
  - Soak tests help in determining the stability of the application under prolonged use and how the performance changes with time.

Deployment Prototyping

- Deployment prototyping can help in making deployment architecture design choices.
- By comparing performance of alternative deployment architectures, deployment prototyping can help in choosing the best and most cost effective deployment architecture that can meet the application performance requirements.

- Deployment design is an iterative process that involves the following steps:
- Deployment Design
  - Create the deployment with various tiers as specified in the deployment configuration and deploy the application.
- Performance Evaluation
  - Verify whether the application meets the performance requirements with the deployment.
- Deployment Refinement
  - Deployments are refined based on the performance evaluations. Various alternatives can exist in this step such as vertical scaling, horizontal scaling, for instance.

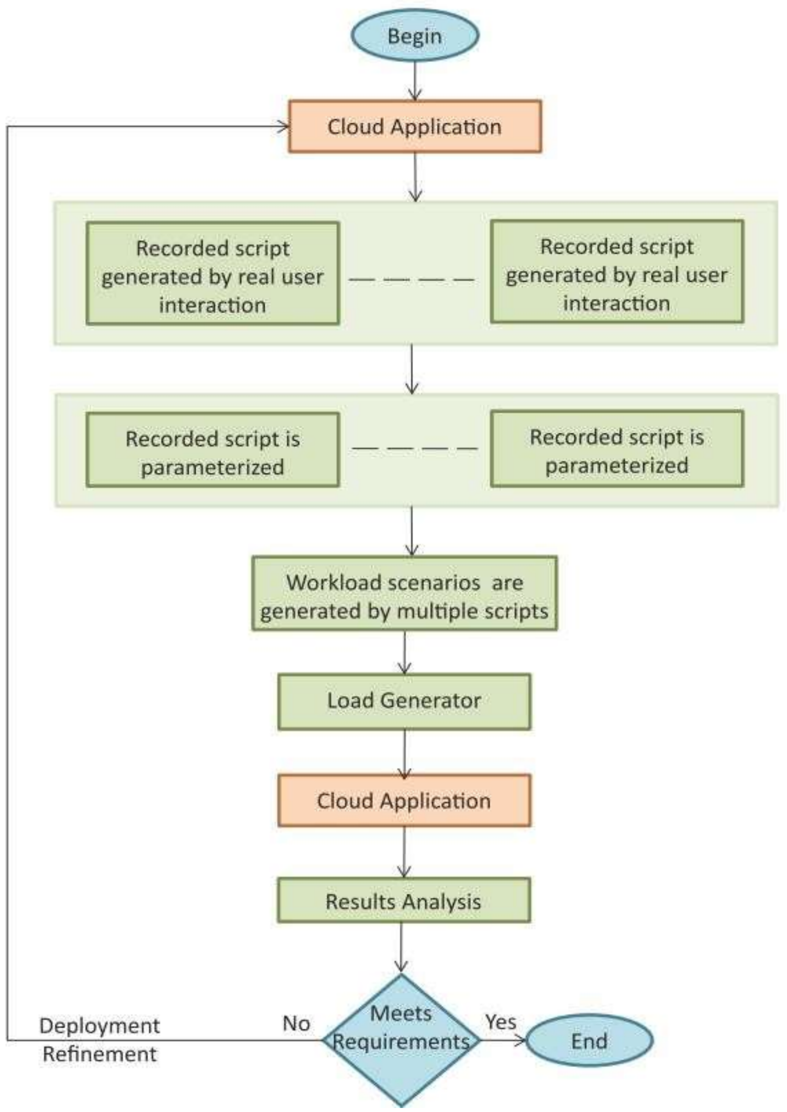


Deployment Design
Number of tiers
Number of servers in each tier
Compute capacities of servers
Server interconnections
Load balancing & replication strategies
Performance Evaluation
Number of users
Workload attributes
Workload mix
Deployment Refinement
Horizontal scaling
Vertical scaling
Alternative server interconnections
Alternative load balancing & replication strategies

Performance Evaluation Workflow

Semi-Automated Workflow (Traditional Approach)

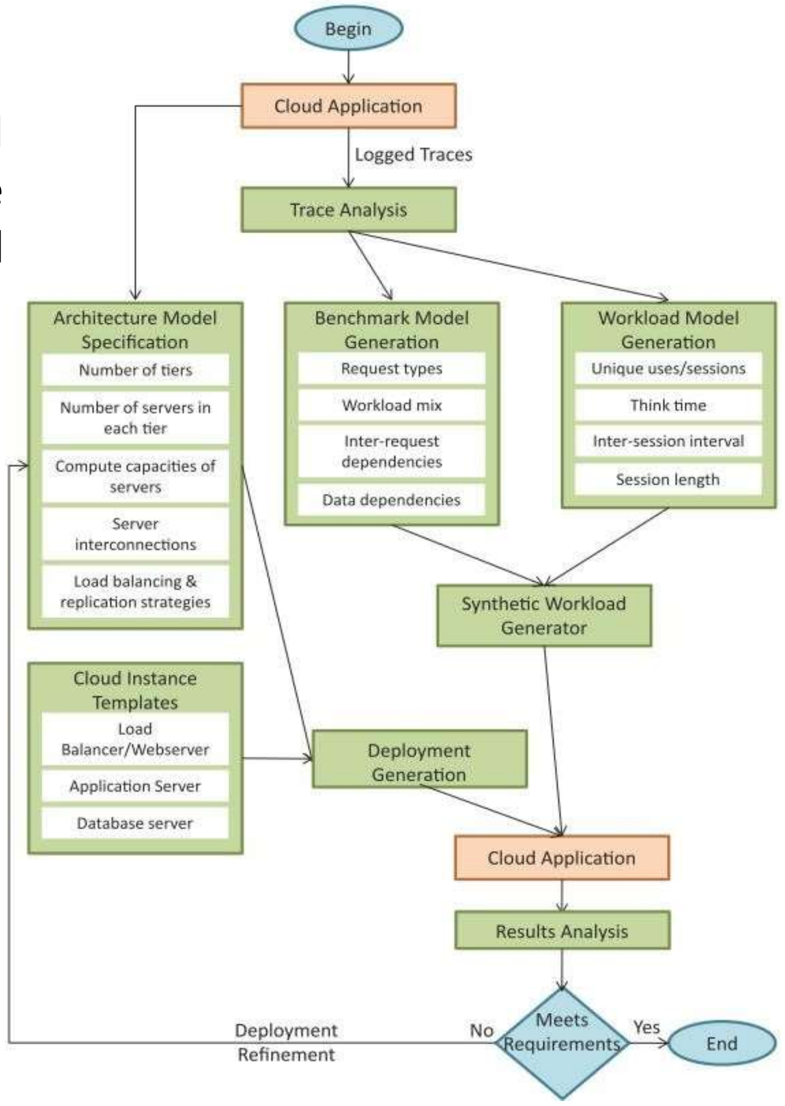
- In traditional approach to capture workload characteristics, a real user's interactions with a cloud application are first recorded as virtual user scripts.
- The recorded virtual user scripts then are parameterized to account for randomness in application and workload parameters.
- Multiple scripts have to be recorded to create different workload scenarios. This approach involves a lot of manual effort.
- To add new specifications for workload mix and new requests, new scripts need to be recorded and parameterized.
- Traditional approaches which are based on manually generating virtual user scripts by interacting with a cloud application, are not able to generate synthetic workloads which have the same characteristics as real workloads.
- Traditional approaches do now allow rapidly comparing various deployment architectures.


Begin
Cloud Application
Recorded script generated by real user interaction
Recorded script generated by real user interaction
Recorded script is parameterized
Recorded script is parameterized
Workload scenarios are generated by multiple scripts
Load Generator
Cloud Application
Results Analysis
Meets Requirements
No
Yes
End
Deployment Refinement

Performance Evaluation Workflow

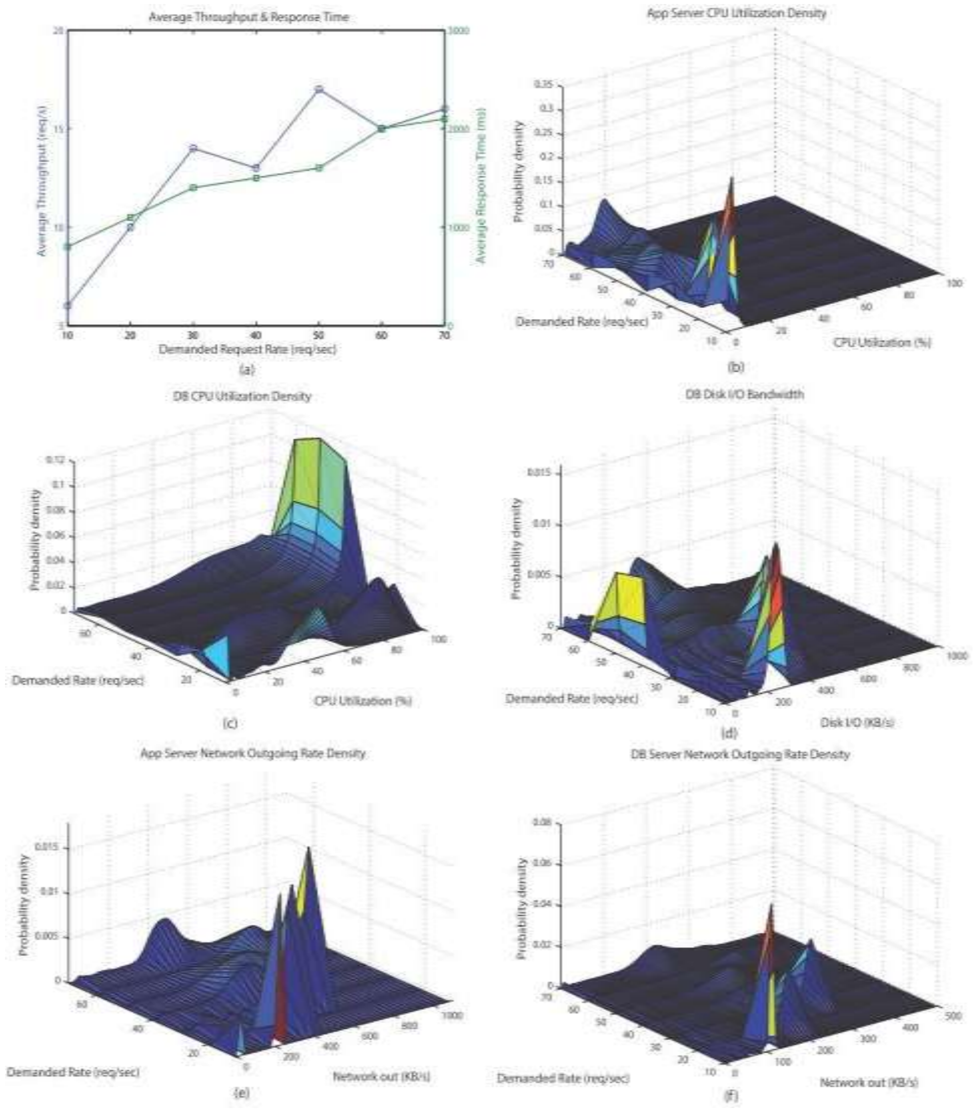
Fully-Automated Workflow (Modern Approach)

- In the automated approach real traces of a multi-tier application which are logged on web servers, application servers and database servers are analyzed to generate benchmark and workload models that capture the cloud application and workload characteristics.
- A statistical analysis of the user requests in the real traces is performed to identify the right distributions that can be used to model the workload model attributes.
- Real traces are analyzed to generate benchmark and workload models.
- Various workload scenarios can be created by changing the specifications of the workload model.
- Since real traces from a cloud application are used to capture workload and application characteristics into workload and benchmark models, the generated synthetic workloads have the same characteristics as real workloads.
- An architecture model captures the deployment configurations of multi-tier applications.



Benchmarking Case Study

- Fig (a) shows the average throughput and response time. The observed throughput increases as demanded request rate increases. As more number of requests are served per second by the application, the response time also increases. The observed throughput saturates beyond a demanded request rate of 50 req/sec.
- Fig (b) shows the CPU usage density of one of the application servers. This plot shows that the application server CPU is non-saturated resource.
- Fig (c) shows the database server CPU usage density. From this density plot we observe that the database CPU spends a large percentage of time at high utilization levels for demanded request rate more than 40 req/sec.
- Fig (d) shows the density plot of the database disk I/O bandwidth.
- Fig (e) shows the network out rate for one of the application servers
- Fig (f) shows the density plot of the network out rate for the database server. From this plot we observe a continuous saturation of the network out rate around 200 KB/s.
- Analysis
    - Throughput continuously increases as the demanded request rate increases from 10 to 40 req/sec. Beyond 40 req/sec demanded request rate, we observe that throughput saturates, which is due to the high CPU utilization density of the database server CPU. From the analysis of density plots of various system resources we observe that the database CPU is a system bottleneck.


Average Throughput & Response Time
Average Throughput (req/s)
Average Response Time (ms)
Demanded Request Rate (req/sec)
(a)
App Server CPU Utilization Density
Probability density
Demanded Rate (req/sec)
CPU Utilization (%)
(b)
DB CPU Utilization Density
Probability density
Demanded Rate (req/sec)
CPU Utilization (%)
(c)
DB Disk I/O Bandwidth
Probability density
Demanded Rate (req/sec)
Disk I/O (KB/s)
(d)
App Server Network Outgoing Rate Density
Probability density
Demanded Rate (req/sec)
Network out (KB/s)
(e)
DB Server Network Outgoing Rate Density
Probability density
Demanded Rate (req/sec)
Network out (KB/s)
(f)