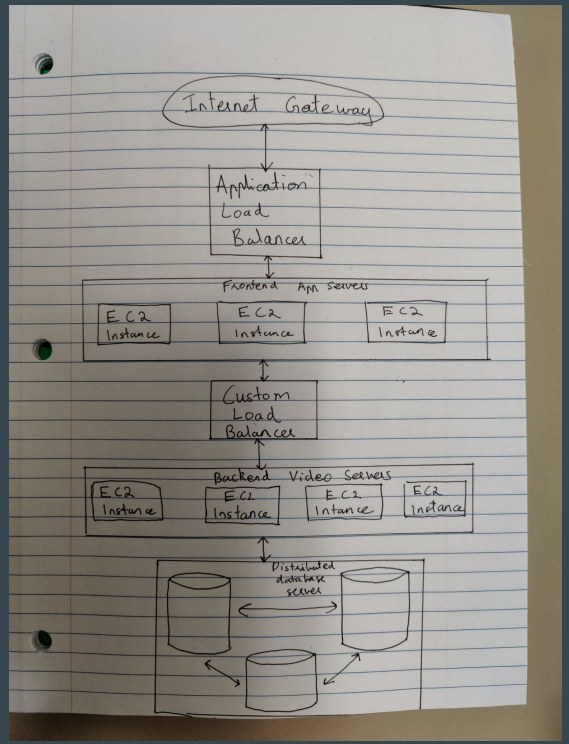
Use Case

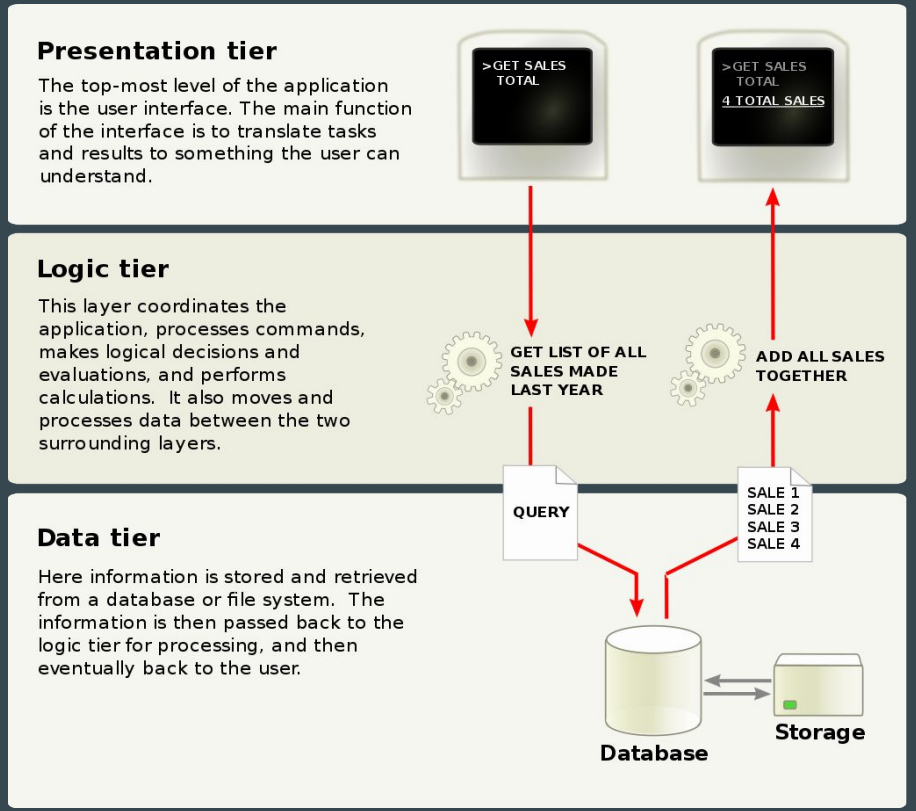
What is our project?



- AWS-deployed video streaming web application with multi-tiered architecture
- Testing and optimizing custom load balancer between frontend and backend servers
- Many LB models considered like CPU usage algorithms and machine learning

Multi-tiered Architecture

- In a web application, 3 common tiers are frontend, backend, and database
- Multi-tier benefits:
  - Modularity
  - Scalability
  - Availability
  - security
- The servers in each tier handle a specific role in the hierarchy



What is a Load Balancer?

- Load balancers (LB) are intermediary nodes between the tiers of a multi-tier web server architecture
- LBs consider a cross-tier task and choose a server from the target tier that is best suited at that time
- There are various LB models for answering the “which server?” question
  - many include using server details as parameters or choosing based on precedent performance

Use Case Requirements - Video Streaming App

- User will be presented with homepage where they can select a particular video they are trying to stream

- Streaming Requirements as follows (typical video streaming standards):
  - transmitted at 3.5 Mbps
  - play length of 9-10 minutes
  - lag length less than 15 seconds (2.5% lag ratio)
  - buffer fill (time it takes video to load) less than 2 seconds

Use Case Requirements - Load Balancers

- LBs will use metrics to choose the front-end and back-end server to use for various processes according to various models
  - e.g. processor utilization, mean response time

- The various LBs' performances will be benchmarked against RandomLB and GreedyLB algorithms
  - RLB will pick servers entirely at random (strong control)
  - GLB only uses shortest processing time to choose servers

Technical Challenges - Video Streaming App

- All tiers of the architecture should be relevant and useful
  - Creating robust front-end, back-end, and database structure

- Target tiers of each LB must be put under sufficient load
  - Warrants usage of LB to solve non-trivial problem of “which server?”
  - Ensures benchmark comparisons reveal useful patterns

- Video Streaming server requires end-user nodes
  - Simulating load creation for the server tiers
  - Tracking app performance from user perspective

Technical Challenges - Load Balancers

- LBs should be able to effectively and efficiently communicate between their respective server tiers

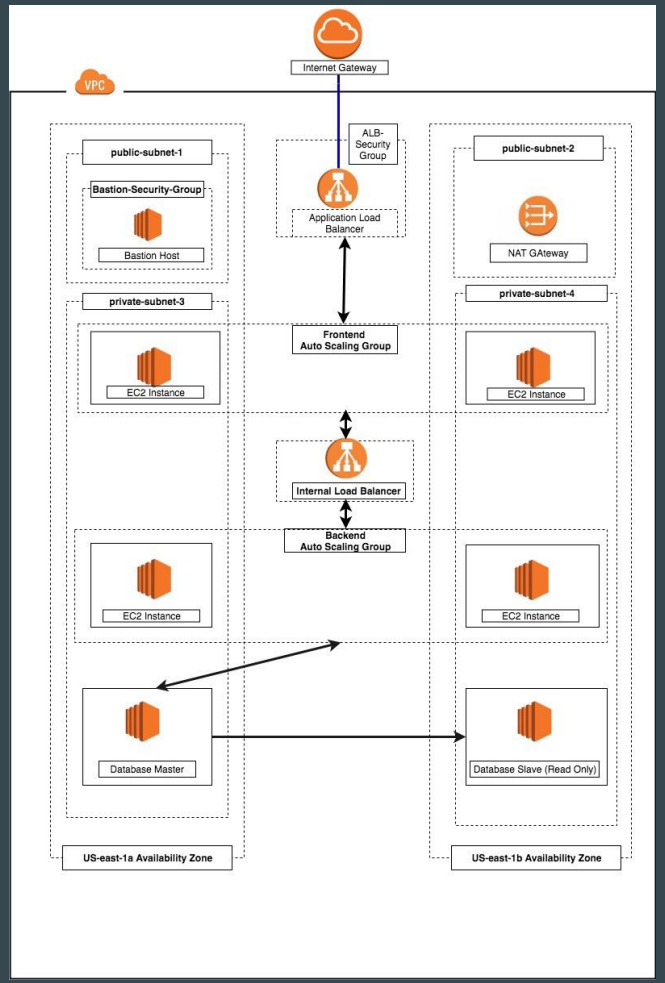
- LBs need access to parameters necessary for their respective models
  - CPU utilization across target server tier
  - I/O levels
  - Hardware specifications

- Need a monitoring system for analyzing each LB's respective performance
  - Should be independent of LB internal program for modularity and robustness

Solution Approaches - AWS

- AWS provides a comprehensive platform for building a multi-tier server architecture
  - Front-end, back-end, and database tiers built via a Virtual Private Cloud interface

- AWS LBs are not sufficiently configurable; we will replace with custom LBs
  - Functionally intermediary EC2 instances



Solution Approaches - End-User Simulation

- Begin with simple single user trial to test server connections

- Move on to multi target trials, we are considering running user simulation scripts on either our own devices, the ECE clusters, or additional AWS ec2 instances

- Would need a sufficient amount of users to put enough load on our servers for differences in load balancing algorithms to matter

Solution Approaches - Monitor Nodes

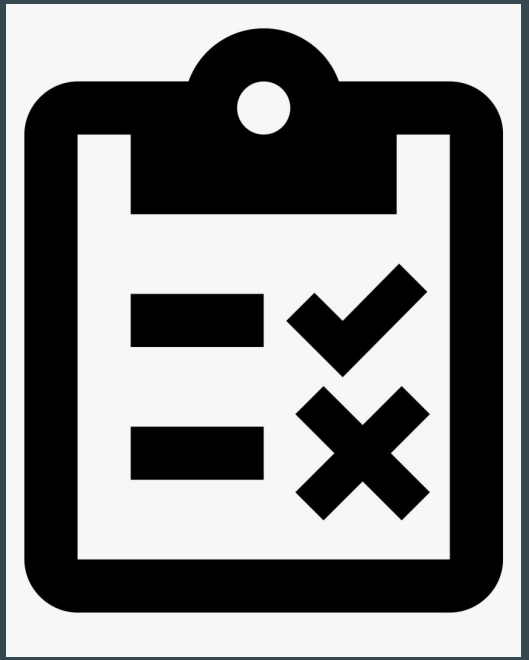
- Nodes that monitor LBs and overall system for performance analysis

- LB monitors are separate EC2 instances that get key metrics from LB I/O
  - modularity and 3rd-party verification
  - E.g. execution time, waiting time, and migration time

- Overall system monitoring can in the End-user nodes
  - Bit rate, lag ratio, etc.

How do we test our product?

- Many trials with users and requests sent over a specific time span (sample size)
- Trial groups have different load balancer implementations
- Simulate users from different geographic locations
- Average metrics for trial groups to compare LB implementations' performance



Schedule and Division of Labor

TASK TITLE	TASK OWNER	START DATE	DUE DATE	PERCENT COMPLETE	01/17 - 01/23	01/24 - 01/30	01/31 - 02/06	02/07 - 02/13	02/14 - 02/20	02/21 - 02/27	02/28 - 03/06	03/07 - 03/13
Project Conception and Initiation												
Team Formation	All	1/17/22	1/23/22	100%								
Project Conceptualization	All	1/24/22	1/30/22	100%								
Writing Abstract	All	1/24/22	1/30/22	100%								
Proposal Presentation Slides	All	1/31/22	2/6/22	100%								
Preparation for Proposal Presentation	Mitul	1/31/2022	2/6/22	100%								
Project Design and Prototyping												
Obtaining AWS Credits	Mitul	2/7/22	2/13/22	20%								
Set Up and Test AWS Connections	All	2/7/22	2/27/22	0%								
Design Presentation Slides	All	2/14/22	2/20/22	0%								
Preparation for Design Presentation	Jason	2/14/22	2/20/22									
Design Review Report	All	2/14/22	2/20/22	0%								
Project Building and Launch												
Video Web Application Frontend	Mitul	2/28/22	03/13/22	0%								
Video Web Application Backend	Jason	2/28/22	03/13/22	0%								
Video Web Application Database	Nakul	2/28/22	03/13/22	0%								
Build Basic Custom Load Balancer	Jason	3/14/22	3/27/22	0%								
Set Up Monitor Node	Nakul	3/14/22	3/27/22	0%								
Implement Random Load Balancing as Control	Mitul	3/28/22	04/03/22									
Test Single User Scenario	Mitul	3/28/22	04/03/22	0%								
Project Monitoring and Improvement												
Begin End User Simulation	Jason	04/04/22	4/10/22									
Implement Basic Machine Learning Algorithms	Mitul	04/04/22	4/10/22	0%								
Implement Traditional Algorithms	Nakul	04/04/22	4/10/22	0%								
Monitor and Compile Data From Monitor Node	All	04/04/22	4/24/22	0%								
Implement Advanced ML Algorithms	All	04/11/22	4/24/22									
Final Presentation Slides	All	04/18/22	4/24/22									
Preparation for Final Presentation	Nakul	04/18/22	05/01/22	0%								