RV College of Engineering

Go, change the world

Experiential Learning Phase -2

Topic : Global Connectivity Framework Analysis of Multi Node Servers and Load Balancing using Bellman Ford Algorithm

SUBJECT	Advanced Algorithms
COURSE CODE	21CS55B3

Team Introduction

USN	Name	Email Id
1RV21CS009	Abhishek Yadav	abhishekyadav.cs21@rvce.edu.in
1RV21CS046	Hardik Hiraman Pawar	hardikhpawar.cs21@rvce.edu.in
1RV21CS049	Harshit Dhoot	harshitdhoot.cs21@rvce.edu.in

Objectives

- **Latency and Load Analysis:** Utilize the Bellman-Ford algorithm to analyze and calculate latency and load metrics for each network node, providing a detailed understanding of data transmission delays and processing loads across the network.
- **Visualization and Reporting:** Develop a multi-bar chart visualization to present the calculated metrics in a visually appealing and informative way, enabling network administrators to easily interpret and analyze network performance data.
- **Optimization Recommendations:** Use the insights from the analysis to provide recommendations for optimizing network performance, such as identifying and addressing bottlenecks, optimizing resource allocation, and improving overall network efficiency.

Selection Sort:

- This algorithm iterates over the ratio list. When a ratio exceeds the threshold, it calculates the difference between the ratio and the threshold. It then finds the smallest ratio in the list and checks if adding the difference to it keeps it below the threshold. If so, it adjusts the ratios accordingly. This process continues until all ratios are within the threshold.

- The time complexity of this algorithm is O(n^2) due to the nested loops (outer loop runs n times, and the inner loop finds the minimum ratio each time).

- The space complexity is O(1) as it operates on the original list without additional storage.

Linear Sort:

- This algorithm also iterates over the ratio list but processes the ratios in two passes. The first pass accumulates the excess ratio (above the threshold) in a buffer, setting the ratios that exceed the threshold to the threshold value. The second pass redistributes the buffer to the ratios below the threshold, ensuring each ratio stays below the threshold.

- The time complexity of this algorithm is O(n) as it iterates over the ratio list twice.

- Like the selection sort, it has a space complexity of O(1) as it modifies the original list without using additional storage.

Algorithm Used

Bellman Ford:

- **Initialize distances:** Set the distance to the source node to 0 and all other nodes to a large number (infinity).
- **Relax edges repeatedly:** Repeat the following steps for a total of n-1 times, where n is the number of nodes: For each edge (u, v) in the graph, if the distance to node u plus the weight of the edge (u, v) is less than the current distance to node v, update the distance to node v to the new lower value.
- **Check for negative cycles:** After n-1 iterations, check for any edges (u, v) where the distance to node u plus the weight of the edge (u, v) is less than the current distance to node v. If such edges exist, the graph contains a negative cycle.
- **Output the shortest paths:** If there are no negative cycles, the final distances to each node represent the shortest path from the source node to all other nodes in the graph.

Methodology

Step 1 : Fetch User's Choice of Region:

- Prompt the user to choose the region for which they want to perform latency, load, and Bellman Ford calculations.

Step 2: Retrieve Nodes Data:

- Read node data from text files for the selected region.
- Retrieve the midpoint and region name for the selected region.

Step 3 : Latency Calculation:

- Compute latency for each node using the Euclidean formula based on the midpoint.
- Plot latency vs node graph.
- Plot latency vs node graph in ascending order.
- Divide nodes into 4 clusters (Quadrants 1, 2, 3, 4).
- Plot a bar chart for each cluster to check the density of nodes distribution.
- Convert latencies into latency ratios.
- Plot the latency ratio vs nodes graph.
- Distribute the latency ratios such that all ratios fall below a predetermined threshold using a distribution strategy (e.g., selection_sort or linear_sort).
- Plot the latency ratio vs nodes graph after normalization (distribution).

Methodology

Step 4 :Load Calculation:

- Get lower and upper limits for load for each node.
- Plot load vs nodes graph.
- Convert loads to load ratios.
- Plot the load ratio vs nodes graph.
- Distribute the load ratios such that all load ratios fall below a predetermined threshold using a distribution strategy (e.g., selection_sort or linear_sort).
- Plot the load ratio vs nodes graph after normalization (distribution).

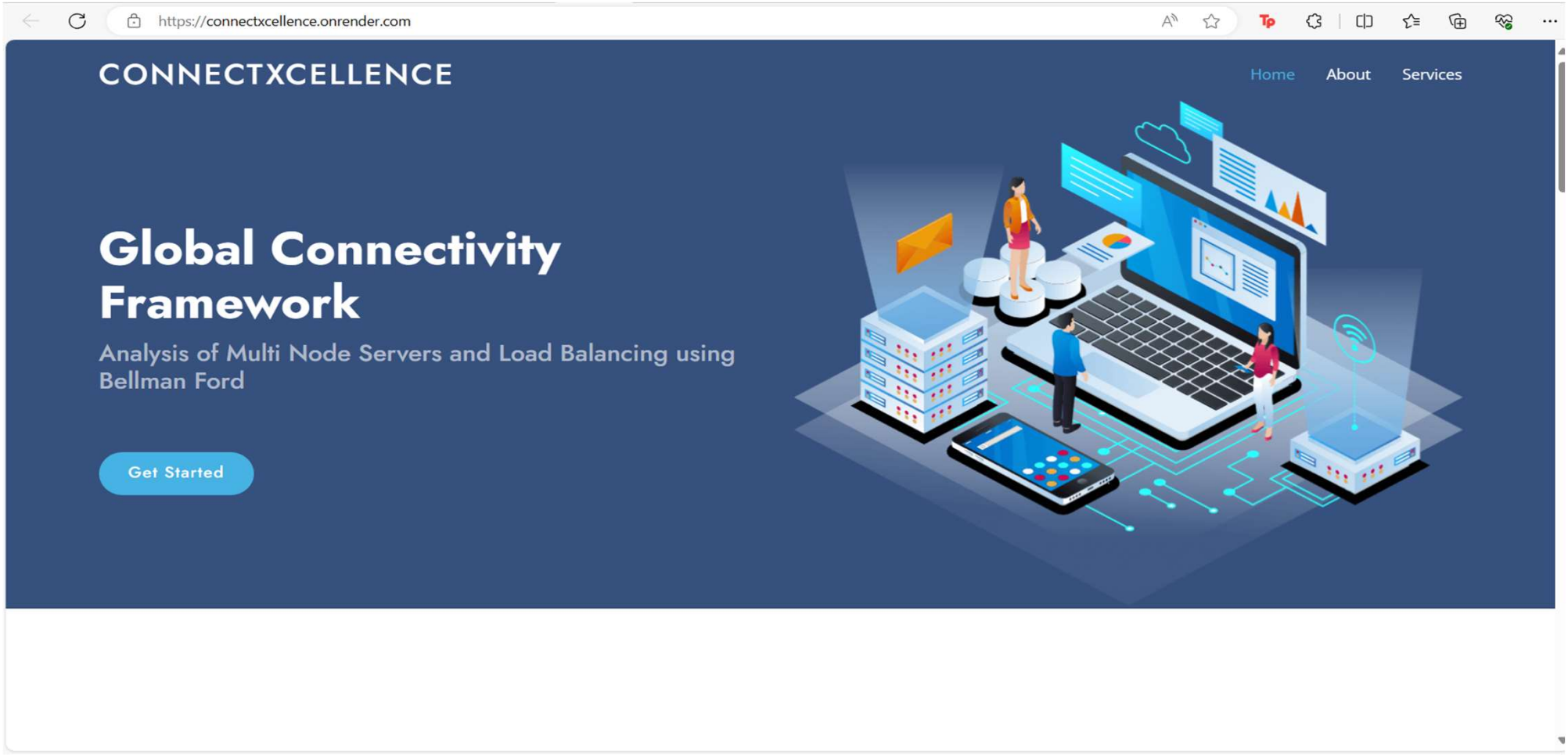
Step 5 : Multi-Bar Chart Analysis of Latency & Load:

- Create a multi-bar chart to analyze latency and load metrics for each node.

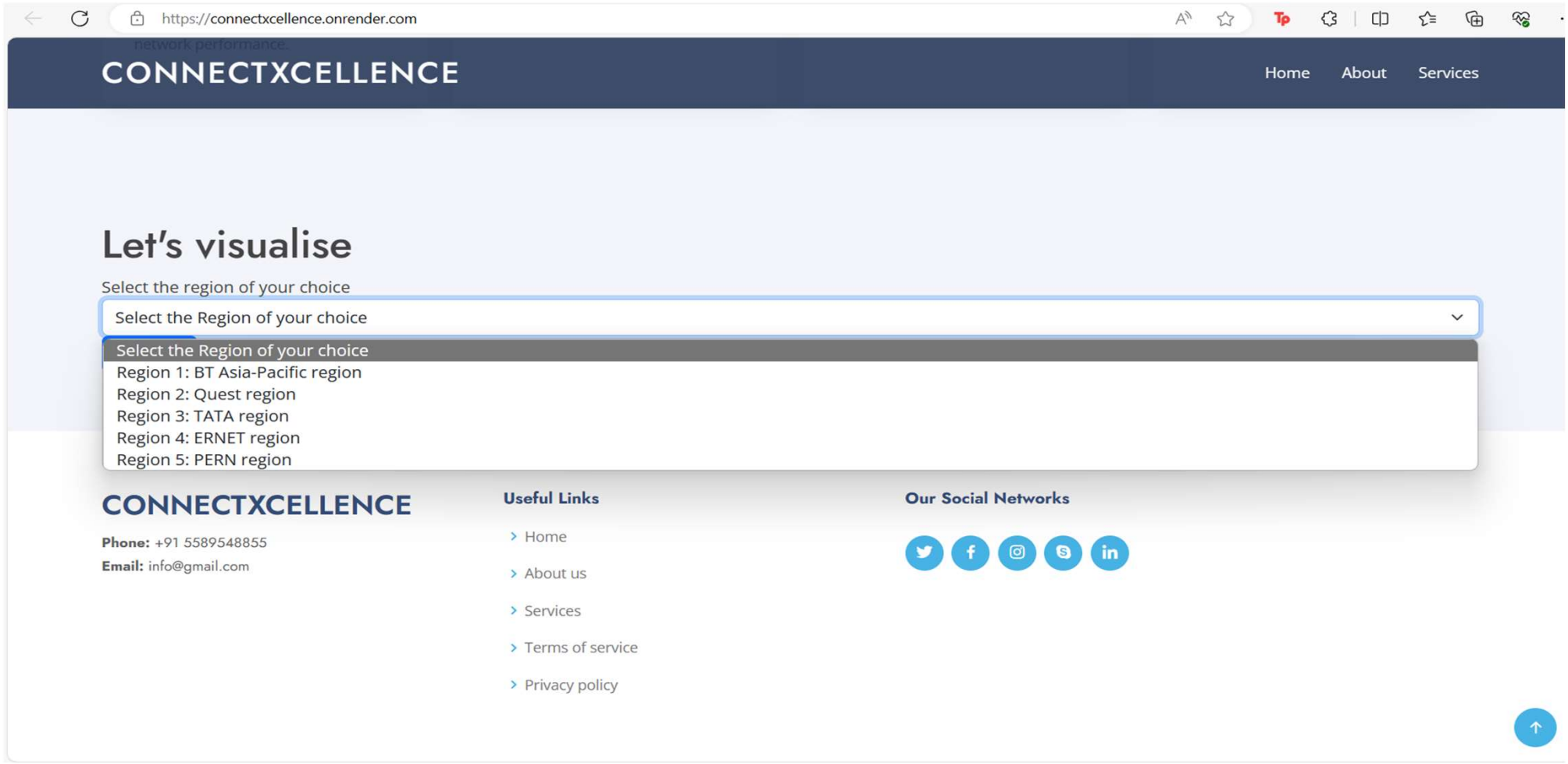
Step 6: Bellman Ford Algorithm:

- Ask the user for the source node.
- Create an adjacency list for all nodes.
- Simulate traffic congestion by setting half of the latencies from the source to neighbors as a large number.
- Apply the Bellman Ford Algorithm to calculate the shortest path.
- Store the results (old distance, new distance, shortest path) in a markdown file called "bellman.md".
- Plot the new network graph with nodes as cities and edges as the distances between two cities.

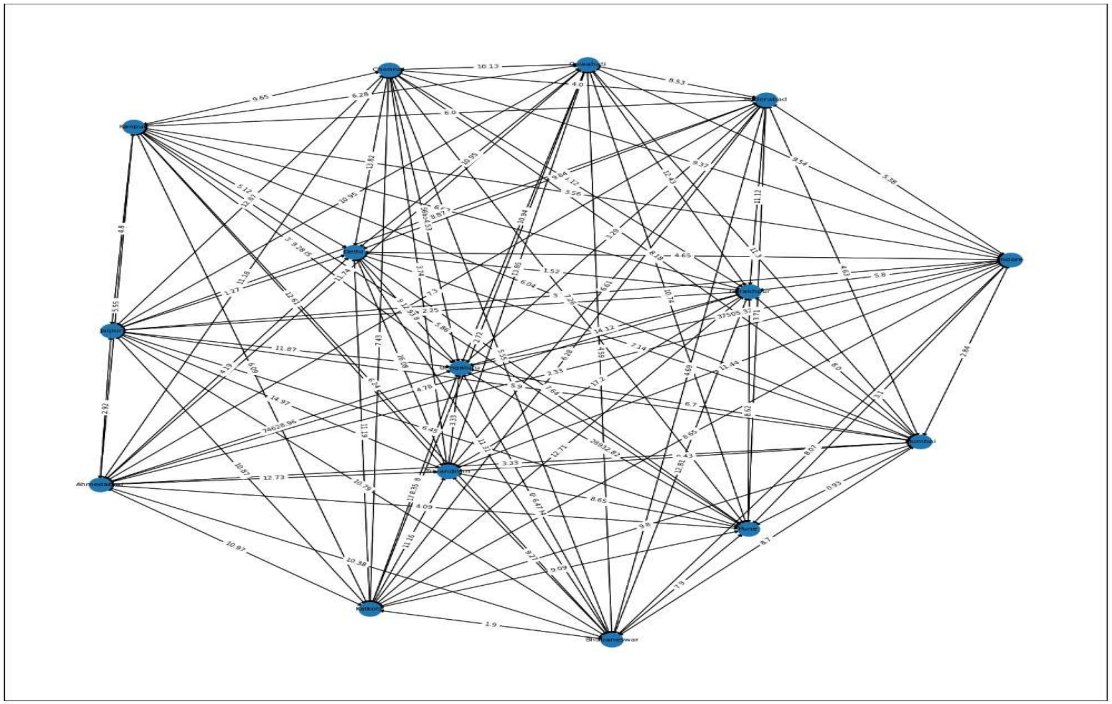
Snapshot


https://connectxcellence.onrender.com
CONNECTXCELLENCE
Home
About
Services
Global Connectivity Framework
Analysis of Multi Node Servers and Load Balancing using Bellman Ford
Get Started

Snapshot


https://connectxcellence.onrender.com
CONNECTXCELLENCE
Home
About
Services
Let's visualise
Select the region of your choice
Select the Region of your choice
Select the Region of your choice
Region 1: BT Asia-Pacific region
Region 2: Quest region
Region 3: TATA region
Region 4: ERNET region
Region 5: PERN region
CONNECTXCELLENCE
Phone: +91 5589548855
Email: info@gmail.com
Useful Links
Home
About us
Services
Terms of service
Privacy policy
Our Social Networks

Snapshot



Snapshot

Adjacency Lists

Pune Node

Neighbour Name	Indore	Trivandrum	Mumbai	Ahmedabad	Jaipur	Chennai	Bengaluru	Delhi	Guwahati	Gorakhpur	K
Latency (ms)	3.1	8.65	0.93	4.09	6.45	7.26	5.82	7.64	10.74	8.62	5

Indore Node

Neighbour Name	Pune	Trivandrum	Mumbai	Ahmedabad	Jaipur	Chennai	Bengaluru	Delhi	Guwahati	Gorakhpur	Ka
Latency (ms)	3.1	11.44	2.84	2.33	3.55	9.37	8.32	4.65	9.54	5.8	3.5

Snapshot

Adjacency List for Source Node: 'Bengaluru' after converting ~ 50% of latencies to a large number

Neighbour Name	Latency (ms)
Pune	91807.98
Indore	72633.91
Trivandrum	47774.83
Mumbai	12638.0
Ahmedabad	69125.07
Jaipur	11.87
Chennai	47316.95
Delhi	15395.96
Guwahati	27435.12
Gorakhpur	11267.87
Kanpur	9.28
Kalkota	70839.67
Bhubaneswar	10190.39

Shortest Path from Source Node to all Nodes

Node Name	Old Distance (ms)	New Distance (ms)	Path
Pune	91807.98	7.0	Hyderabad -> Pune
Indore	72633.91	8.67	Hyderabad -> Indore
Trivandrum	47774.83	9.9	Hyderabad -> Trivandrum
Mumbai	12638.0	7.92	Hyderabad -> Mumbai
Ahmedabad	69125.07	10.59	Hyderabad -> Ahmedabad
Jaipur	11.87	11.87	Jaipur
Chennai	47316.95	7.29	Hyderabad -> Chennai
Bengaluru	0	0	
Delhi	15395.96	13.13	Hyderabad -> Delhi
Guwahati	27435.12	11.82	Hyderabad -> Guwahati
Gorakhpur	11267.87	14.12	Jaipur -> Gorakhpur
Kanpur	9.28	9.28	Kanpur
Kalkota	70839.67	9.57	Hyderabad -> Kalkota
Bhubaneswar	10190.39	7.98	Hyderabad -> Bhubaneswar

Applications

Network Monitoring and Management: The application can be used for real-time monitoring of network performance, allowing administrators to identify and address issues promptly, ensuring smooth network operations.

Resource Allocation Optimization: By analyzing latency and load metrics, the application can help optimize resource allocation, ensuring that network resources are efficiently utilized and bottlenecks are minimized.

Network Planning and Design: The insights provided by the application can be used in the planning and design of network infrastructure, helping to create efficient and scalable networks.

Quality of Service (QoS) Improvement: By optimizing network performance, the application can improve the overall quality of service for users, ensuring a reliable and fast network connection.

Troubleshooting and Diagnostics: The application can be used for troubleshooting network issues, helping administrators identify and resolve problems quickly, minimizing downtime and disruption to network services.

Conclusion

- Utilizes Bellman-Ford algorithm for network performance analysis and optimization.
- Calculates latency and load metrics for each network node.
- Visualizes metrics in a multi-bar chart for easy interpretation.
- Provides insights for resource allocation and bottleneck identification.
- Empowers network administrators to make informed decisions for network optimization.
- Enhances capabilities for proactive network management.

Project Demo

THANK YOU