Mini Display Wall

Group-3

Group Members:

- Sandeep Parmar
- Madhur Bansal
- Paritosh Pankaj
- Tanmay Purohit

Course Instructors:

- Dr. Preeti Malakar
- Dr. Soumya Dutta

Problem Statement

- Build a Mini Display wall that can help in high resolution visualization of large scale data

- Implement and compare the two approaches:
  - Sending data in parallel to renderers
  - Sending rendered images to clients

Technical Method

For implementing the above two approaches, we used the following resources and libraries:

- Programming language: Python
- Libraries used: NumPy, VTK, OpenCV, Socket, Threading

Sending Simulation Data to Clients

- Data is streamed from to server to clients using TCP/IP sockets for visualization.

- Server:
  - Data is generated as a 2-D array at the server at every timestep
  - The 2-D array is partitioned into smaller parts
  - Each part is sent to a client based on the index assigned to the client

- Client:
  - Each client receives the 2-D data array at every timestep
  - The 2-D array is visualized on every client using VTK

Sending Images to Clients

- Rendered images are sent from the server to the clients for display. This approach is not real time.

- Server:
    - Data is generated and rendered using VTK at the server itself
    - The rendered images are partitioned and stored on the disc using OpenCV
    - A separate process sends images to clients based on the index

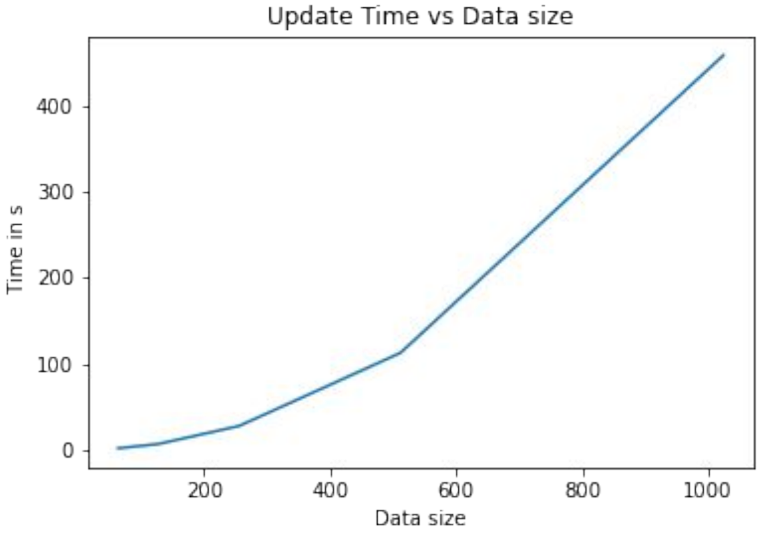
- Client:
    - Client receives the images and stores them on the disc
    - The images can be visualized at any point of time

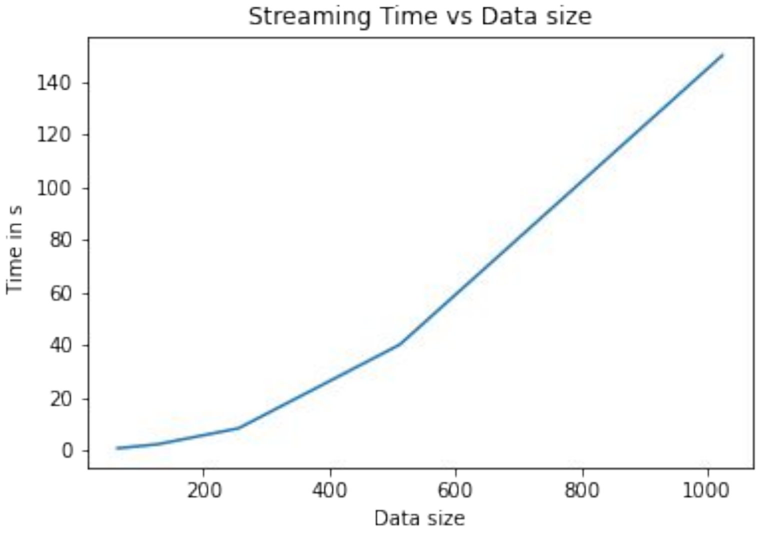
Performance Evaluation

Dimension		**64*64**	**128*128**	**256*256**	**512*512**	**1024*1024**
Timestep		100	100	100	100	100
Data streaming	**Updates (in s)**	1.673	6.68	27.63	112.62	458.36
	Sending (in s)	0.703	2.24	8.28	40.08	149.92
Image Transfer	**Updates and storing images (in s)**	19.61	28.98	57.95	152.81	NA
	Sending (in s)			108.98		

***Averaged over 5 runs.**

Performance Evaluation

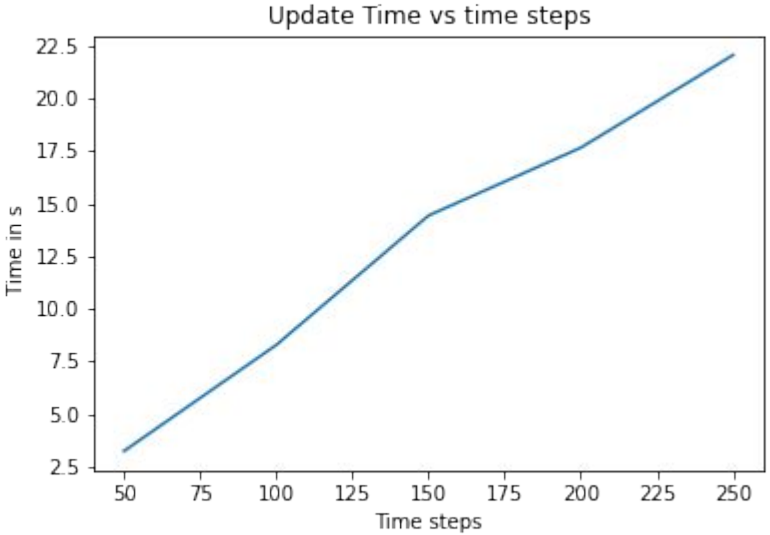




Performance Evaluation

Dimension		**256*256**	**256*256**	**256*256**	**256*256**	**256*256**
Timestep		50	100	150	200	250
Data streaming	**Updates (in s)**	13.85	27.56	41.43	55.42	69.13
	Sending (in s)	3.23	8.28	14.44	17.67	22.08

***Averaged over 5 runs.**


Update Time vs time steps
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Time in s
50
75
100
125
150
175
200
225
250
Time steps

Challenges Faced

- We first tried Visit (libsim approach) for implementation.
- In VTK C++ setup was difficult.
- Client was unable to receive whole data in one recv() call.
- Network traffic also impacting the streaming time.

Future Work

- Data generation on server side can be done parallely.
- Visualisation of array can be improved
- We can extend our problem statement to 3D-data.

Work Distribution

- Client-Server connection using TCP/IP Sockets
  - Madhur and Sandeep

- VTK array visualisation and Image generation
  - Tanmay and Paritosh

Thank You :)