Interactive Remote Large-Scale Data Visualization via Prioritized Multi-resolution Streaming

GROUP - 8

Divyanshu - 241110023
Senthil Ganesh P - 241110089
Khushwant Kaswan - 241110035

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur (IITK)

Contents

Introduction

- ➢ Scientists working with petascale and future exascale supercomputers face significant challenges in visualizing and analyzing large-scale simulation data remotely.
- ➢ Scientists and supercomputing resources are located in geographically different location.
- ➢ The most significant barrier to effective distance visualization is the lack of network bandwidth.
- ➢ Large data sizes surpass display and network capabilities.

Introduction *contd.*

- ➢ Improving network architecture is important to handle large data transfers effectively.
- ➢ However, this research proposes an alternative approach to reduce the demand on the network by providing smarter visualization.

Approaches to Remote Visualization

- **Image-Based approach**
- **Representation-Based approach**
- **Multi-resolution representation-Based approach**

These methods are designed to visualize large-scale data generated by supercomputers for remote analysis.

Image-Based approach

Process Overview

- ➢ All raw data and simulations are stored and processed on the supercomputer.
- ➢ The supercomputer renders the data into image frames according to the user’s requests.
- ➢ The rendered images are transmitted over the network to the client’s device.
- ➢ The client receives the images and displays them sequentially.

Image-Based approach *contd.*

Advantage

- **Fixed Data Size :** Sends fixed-size images, which consume constant bandwidth.
- **Simpler for Clients :** No need for powerful local rendering hardware is required.

Disadvantage

- **Bandwidth and Latency Dependent :** The frame rate and interaction speed are limited by network capacity .Performance is highly dependent on network quality.
- **Slow Interactivity :** Interaction can be slow because each new view requires a new image to be rendered and sent.

Representation-Based approach

Process Overview

- ➢ The supercomputer performs initial data processing and prepares data representations based on the user's needs.
- ➢ The processed data representations are sent from the supercomputer to the client over the network.
- ➢ The client receives the data representations and perform local rendering.
- ➢ This rendering happens in real-time on the client machine, allowing users to interact directly with the data.

Representation-Based approach *contd.*

Advantage

➢ **Enhanced Interactivity :** It allows local rendering, improving the interactive experience.

Disadvantages

➢ **Initial Latency :** The first data representation can take longer to arrive, as it is larger than a simple image.

➢ **Memory Limitations:** The representation can be larger than the client side memory.

Image based vs Representation based approach

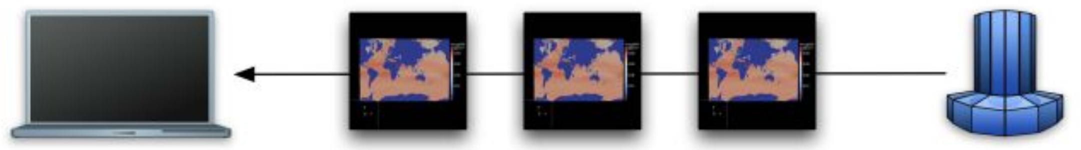
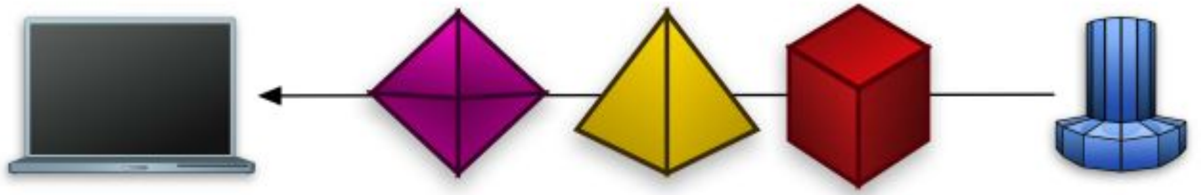


Image-Based approach



Representation-Based approach

Image-based sends images over time, while representation-based sends processed data over time.

Multi-resolution representation based approach

Process overview

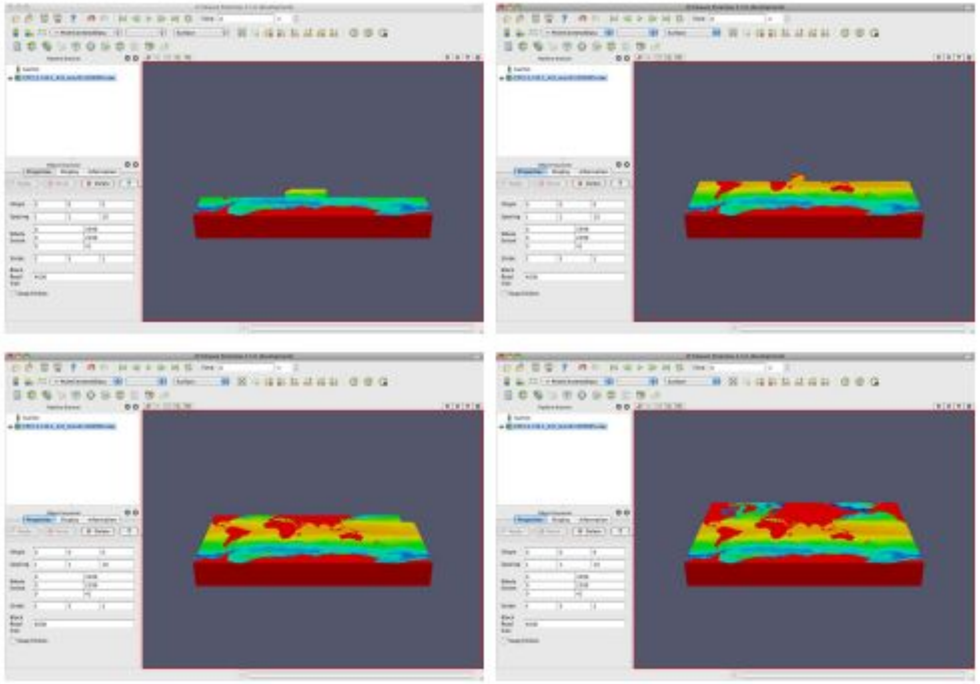
- **Data Processing :** Supercomputer processes the data set into multiple resolutions (low to high).
- **Chunk Transmission :** Send chunks of data in multi-resolution manner that matches the network speed and display size.
- **Local Rendering :** Client renders the received chunks, starting with low-resolution data for immediate visualization.
- **Resolution Enhancement :** Full resolution visualization is done over time by streaming high resolution representation. (Will see later)

Multi-resolution representation based approach *contd.*

Advantage

- ➢ **Efficient Bandwidth Use:** By matching data chunk size to network speed, the approach avoids overloading of the network.
- ➢ **Lowers Initial latency :** First frame latency is reduced by sending small sized data chunk.
- ➢ **Immediate visualization :** Immediate visualization is achieved through low resolution data chunk.
- ➢ **Accounting for client memory limitations :** Ensures that we do not exceed the memory limitations of the local computers that are performing the rendering

Previous and Related work



Prioritized streaming of surface coloring of the CFC concentration in simulated ocean data.

The blocks fill in from front to back. This is because the pieces here are prioritized by shortest distance from the camera

Process overview

- **Data Division :** The large data set is divided into smaller pieces or blocks.
- **Prioritization and Culling:** These chunks of data are streamed based on priority. Pieces that are less important to the final image are removed to save processing time.
- **Streaming Sequence:** Pieces closer to the camera (that have a higher priority) are streamed first, ensuring that the most relevant information is displayed early in the visualization process.
- **Purpose :** This architecture reduced I/O and processing time by only operating on pieces that contribute to the final result.

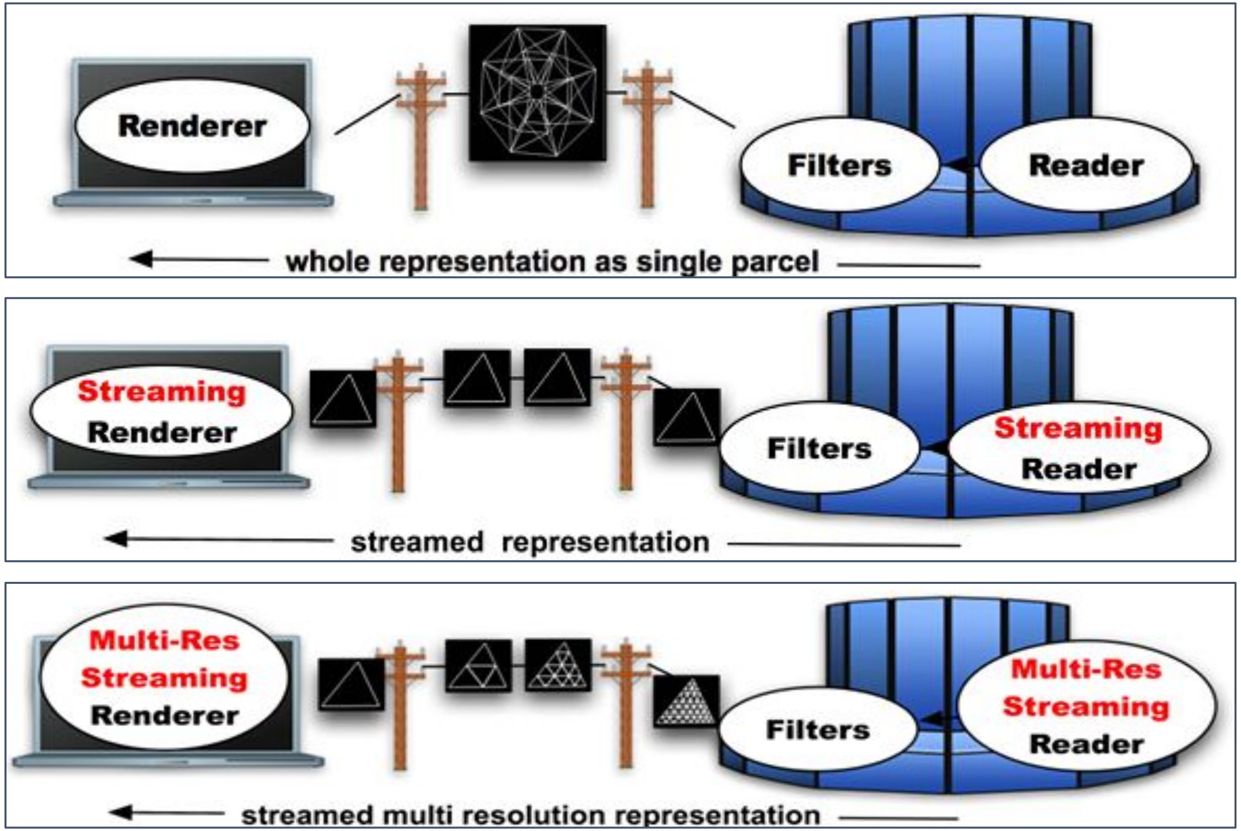
Methodology Overview

- Focus of the research- Prioritized, multi-resolution, representation based remote visualisation

- Method of prioritizing multi-resolution data (important low resolution data and subsequent refinement)

- Key additions- Server and client (handling multi-resolution data) apart from pipeline

Methodology Overview *contd.*


Renderer
Filters
Reader
whole representation as single parcel
Streaming Renderer
Filters
Streaming Reader
streamed representation
Multi-Res Streaming Renderer
Filters
Multi-Res Streaming Reader
streamed multi resolution representation

Methodology Overview *cont*.

- ➢ Role of prioritization in Multi-resolution streaming (Enhancement over time)
- ➢ Requirement of full resolution data and optimised network utilisation (area of interest)
- ➢ Enhancing resolution through spatial split and assignment of priority
- ➢ This Multi-resolution functionality operates in various visualization operations

Implementation

- ➢ Reader and filters reside on supercomputer side while renderer on client side connected through WAN
- ➢ Renderer requests data pieces that are incrementally served by supercomputer
- ➢ Multi-resolution visualisation added to the original streaming architecture (that runs on full resolution)

Multi-resolution Reader

- The problem with preprocessing the data for generating multi-resolution data - Time and space factor in rewriting the data.
- Uniqueness of our multi-resolution data reader - preserves the original data in highest resolution and creates additional files.
- Virtual tree structure with specific height and degree.

Multi-resolution Reader *contd.*

User selects

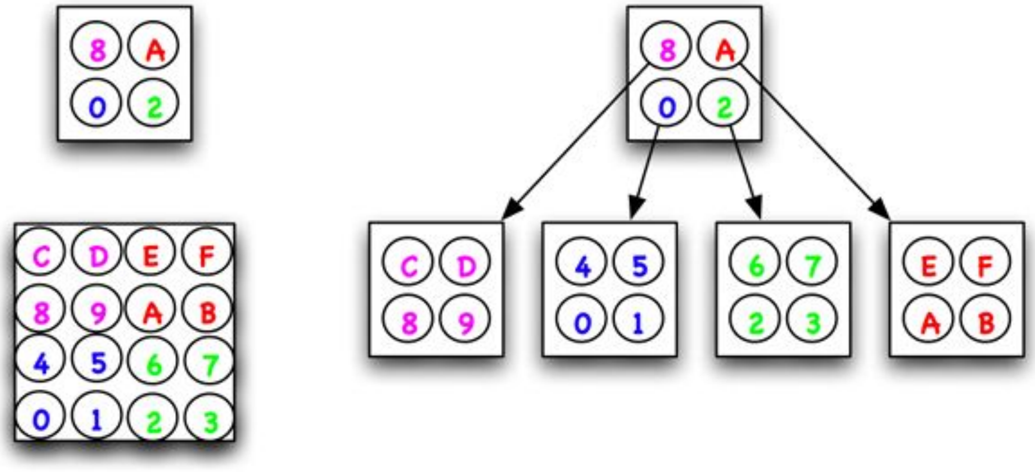
- ➢ *degree* where 2^{degree} is the no. of children for each parent
- ➢ *Stride* where the sampling at a level *k* is $stride^{degree*k}$
- ➢ *height* which is the no. of levels of resolution

Size of the additional files containing multi-resolution representations is given as:

$$\sum_{k=1}^{height-1} r^k = \frac{r^{height} - r}{r - 1}$$

$$r = \frac{1}{stride^{degree}}$$

Multi-resolution Reader *contd.*



A multi-resolution representation tree with *stride*=2, *degree*=2 and *height*=2

Multi-resolution Reader *contd.*

- Implication of the sum in deciding the total size of original file and sum of multi-resolution file
- Choosing *degree*, *stride* and *height* for the given data set
- Data transfer time = $D * r^k / b$ (assuming zero server processing time)
- Implication of height on lowest resolution data fraction size ($D * r^{height-1}$) and further on data piece size ($stride^{degree*height-1}$).
- Implication of Large values of height on latency and bandwidth

Multi-resolution Reader *contd.*

Pros of this method of generating Multi-resolution data:

➢ Reduced pre-processing of data (through striding) to achieve a linear time

➢ Time to generate Multi-resolution data is same as reading the data once

Cons of this method of generating Multi-resolution data:

➢ More aliasing error due to strided sampling (but correct data will be shown in our highest resolution)

Multi-resolution Renderer

- Runs on the client side and drives the piece processing requests
- Subsequent requests will refine the data based on priority
- Parent piece may be split into 2^{degree} pieces and requests are sent down the pipeline which is met by the reader (post filtering)
- This has the benefit of immediate image displayed on the client side through low resolution data (and progressively update visualisation over time)

Multi-resolution Renderer *contd.*

The rendering logic for multi-resolution renderer:

- ➢ Request and render the lowest resolution data
- ➢ Put that data in priority queue
- ➢ While priority queue is not empty
    - Dequeue top element
    - Split it into children
    - Calculate the priority of the children or determine if they can be culled
    - Request the remaining children in priority order and render them
    - Push the children into the priority queue
- ➢ The renderer will continue until there can be no more refinement (happens continuously when visual interaction takes place)

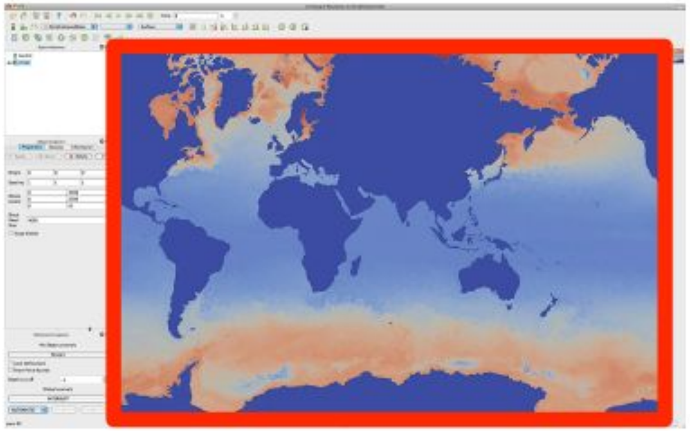
Pipeline Information Exchange

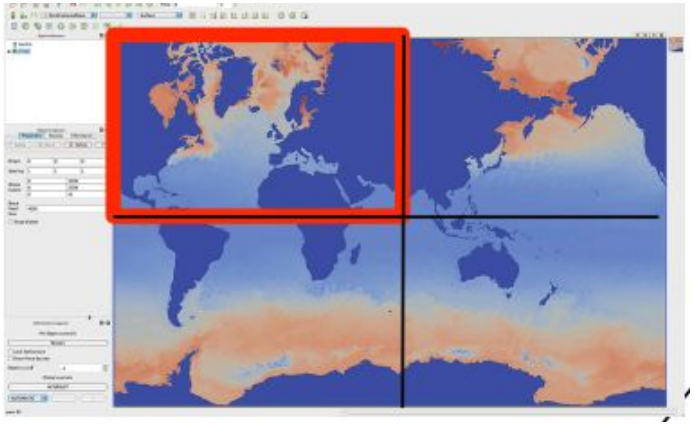
Modifications made to the ParaView visualization pipeline to support multi-resolution rendering :

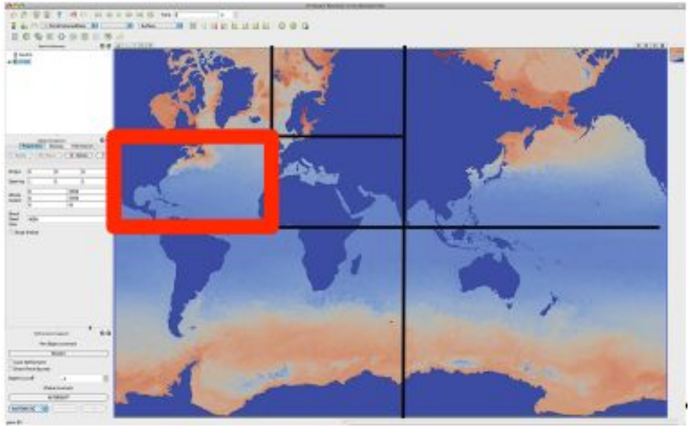
1. Meta-Information in the Pipeline
   a. Meta-information in the visualization pipeline is passed both upstream and downstream
   b. Used previous work's streaming architecture that allowed prioritization and culling by attaching metadata keys, such as the PRIORITY key.
2. New MetaData Keys for Multi-Resolution
   a. Two keys are introduced: REQUEST RESOLUTION and REQUEST UPDATE EXTENT
   b. During the image res. update, the renderer requests different data pieces based on priority and resolution levels needed.
3. Default Prioritization
   a. Prioritization is disabled by default as a conservative measure.
   b. Some filters may change the data in the pipeline, which may invalidate metadata. When this happens, prioritization might fail due to unreliable metadata.

Multi-resolution Prioritized Streaming (Full Extent)

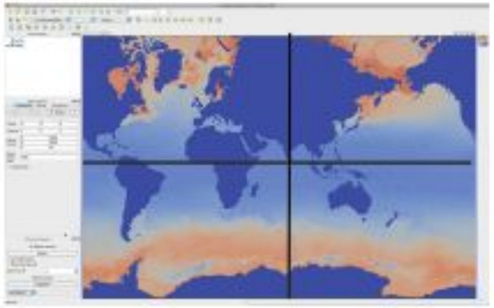
1. Send and render lowest resolution data
2. Virtually split into spatial pieces and prioritized by Pipeline Information Exchange
3. Progressively updates and refines the rendering of highest priority piece
4. Goto step 2 until the data is at the highest resolution

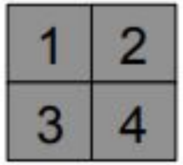


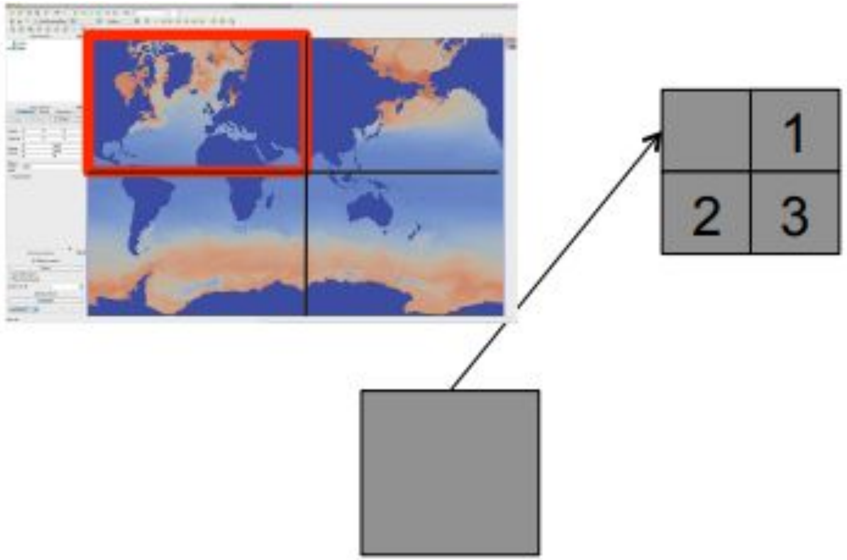




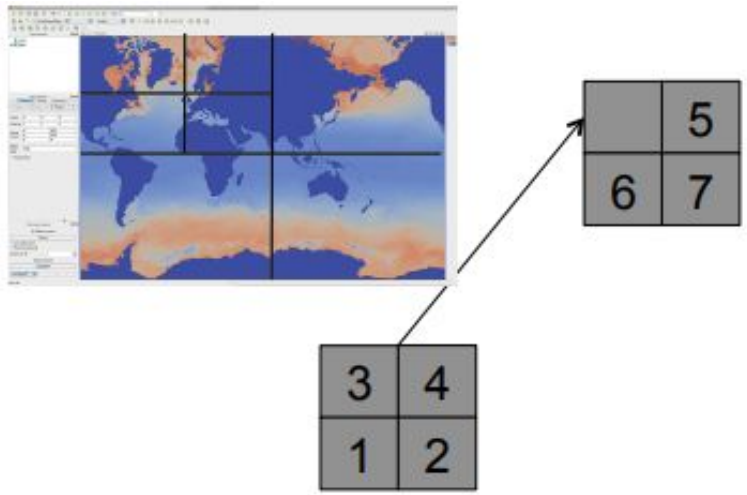
Multi-resolution Prioritized Streaming *contd.*

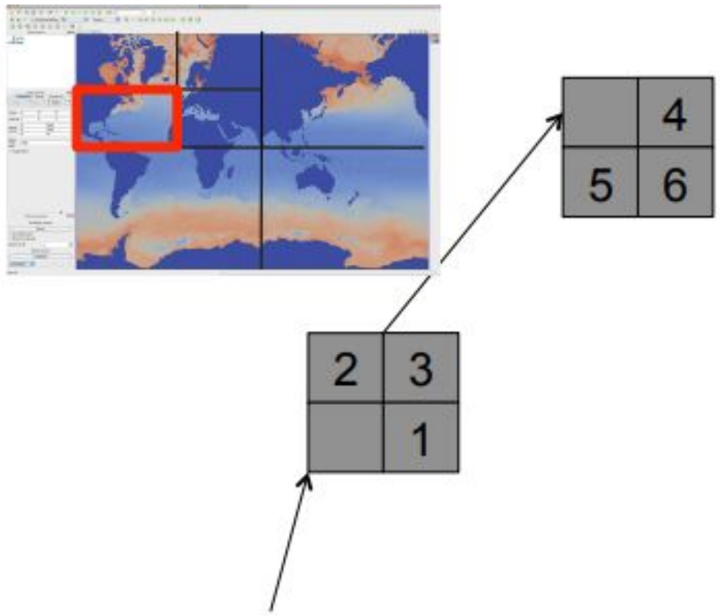




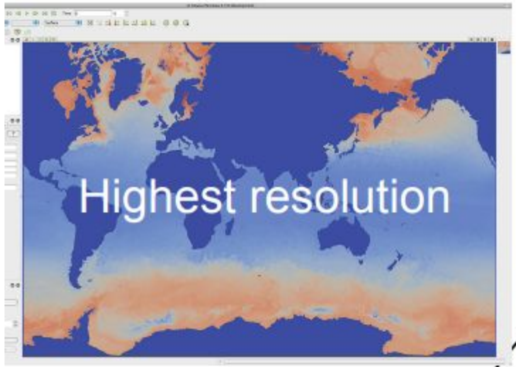


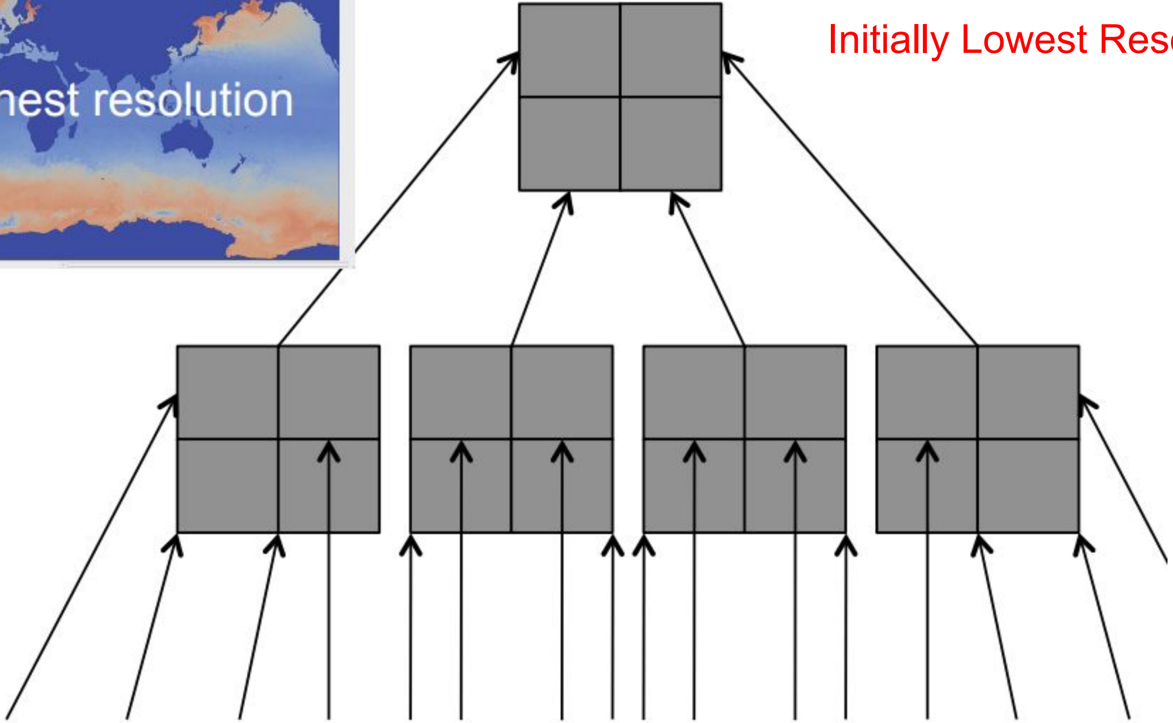
Multi-resolution Prioritized Streaming *contd.*





Multi-resolution Prioritized Streaming *contd.*





Remote LARGE Data

- LANL climate team uses 100 TeraFLOPs of Jaguar (ORNL) to run their Parallel Ocean Program (POP) simulations.
- The team is generating approximately 1.5 terabytes a day.
- At a 10 Mbps Link between LANL and ORNL this would take approximately 370 hours (over 15 days) to transfer.

Due to the size of the data and the remote location, a fast ,scalable remote visualization solution was needed.

Remote LARGE Data *contd.*

Used single field of float data from the POP data set.
This data set at full resolution is 3600×2400×42 at 1.35 GB.

For generating multi-resolution data, stride = 2, height = 5 and degree = 2

The original data is virtually broken and stored as :

- 1800 × 1200 × 42 : 64 virtual pieces of 346 MB total
- 900×600×42 : 16 virtual pieces of 86.5 MB total
- 450 × 300 × 42 : 4 virtual pieces of 22.6 MB total
- Lowest resolution is 225×150×42 : 1 piece at 5.4 MB.

The multi-resolution data representation on disk only takes 1/3 ∗ 1.35GB extra disk space.

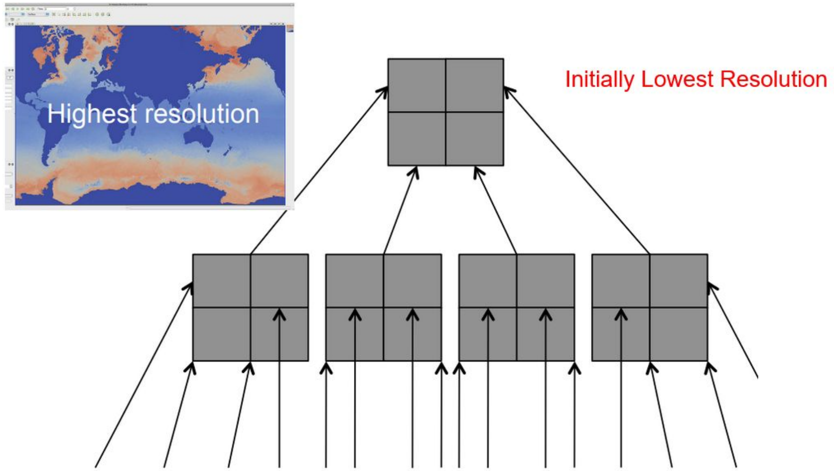
Remote LARGE Data *contd.*

h=5

The server processed the data and creates height − 1 resolution levels ie. 4 resolution levels.

For sake of convenience the resolution is represented as $n * n * 1$ where $n = 2^{h-1}$

- Lowest Resolution : 16x16x1 : 1 piece
- Next better resolution : 8x8x1 : 4 pieces
- 4x4x1 :16 pieces
- 2x2x1 : 64 pieces



Results

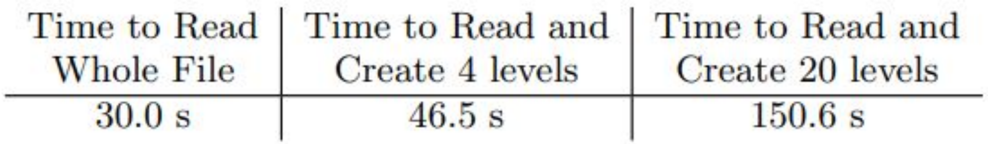
The tests were performed with a modified version of ParaView that implements multi-resolution streaming.

Specifications :

- Mac Pro 2 × 3GHz Quad Core Xeon
- 16GB of memory and Seagate storage drive (300MB/s peak disk bandwidth and 8 MB disk cache)
- Varied the tests between 1Mb to 1Gb bandwidth with a 100ms latency

Before every test the OS file cache was flushed, otherwise the data set would be cached in memory skewing the timings.

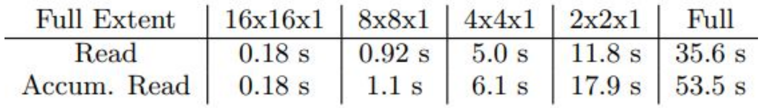
Results *contd.*



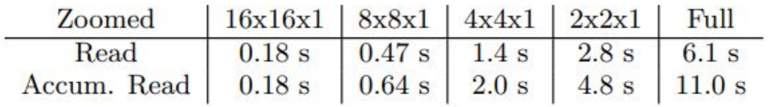
Time to Read Whole File	Time to Read and Create 4 levels	Time to Read and Create 20 levels
30.0 s	46.5 s	150.6 s

Observations :

- ➢ 20 levels is impractical because spatial chunks/read sizes become one float each, and sending one float at a time is not practical due to network latency
- ➢ Keeping the data as-is, but with additional multiresolution files, has not had a significant impact on our read times for processing.
- ➢ The time to read the full res. data set along with small spatial extents of multi resolution streaming pieces is 35.6 seconds.
- ➢ Zoomed reading : We can also get small spatial portions of main data very quickly as observed in the read times for zoomed extent.

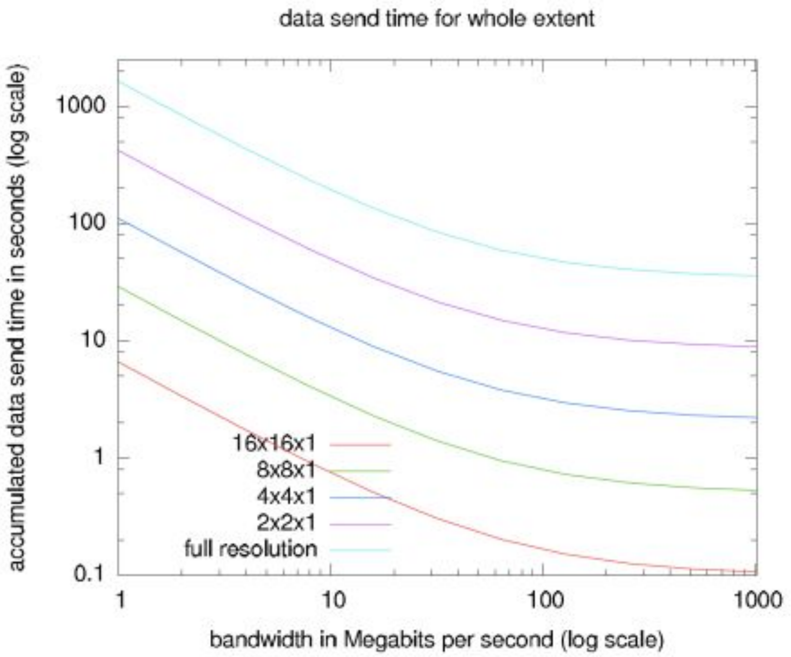


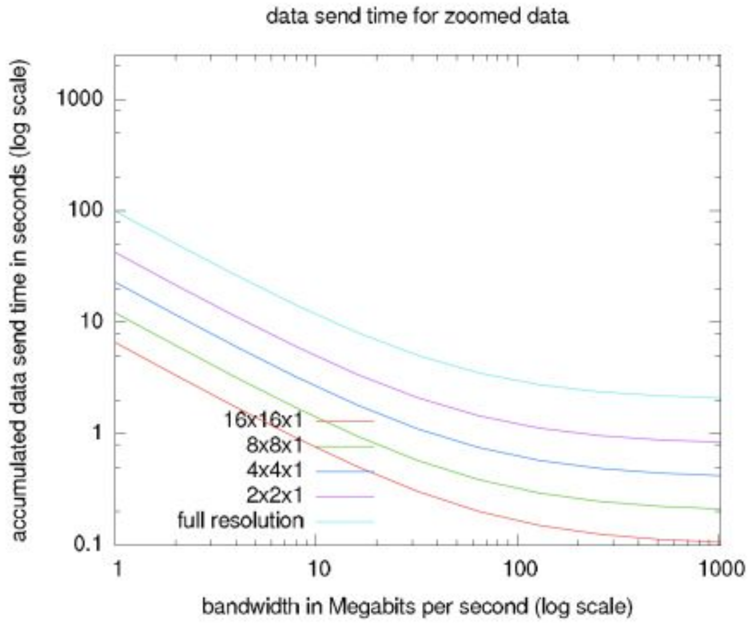
Full Extent	16x16x1	8x8x1	4x4x1	2x2x1	Full
Read	0.18 s	0.92 s	5.0 s	11.8 s	35.6 s
Accum. Read	0.18 s	1.1 s	6.1 s	17.9 s	53.5 s



Zoomed	16x16x1	8x8x1	4x4x1	2x2x1	Full
Read	0.18 s	0.47 s	1.4 s	2.8 s	6.1 s
Accum. Read	0.18 s	0.64 s	2.0 s	4.8 s	11.0 s

Results *contd.*



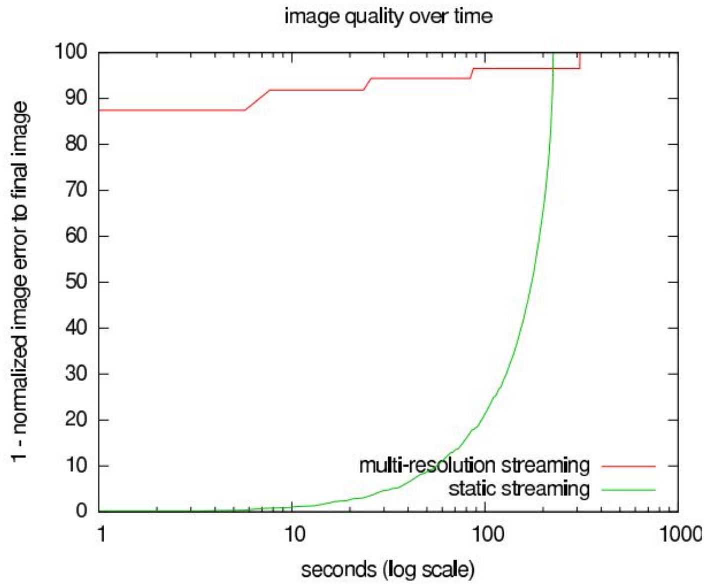


Results *contd.*

<u>Surface rendering of whole POP ocean dataset over 8 Mbps Network speed and 100ms latency</u>

Observations :
Immediate good quality image,
with increasing refinement over time
90% image accuracy in the first frame

In CIELUV color space, the distance between two points approximately tells how different the colors are in luminance, chroma, and hue.
(CIE)International Commission on Illumination



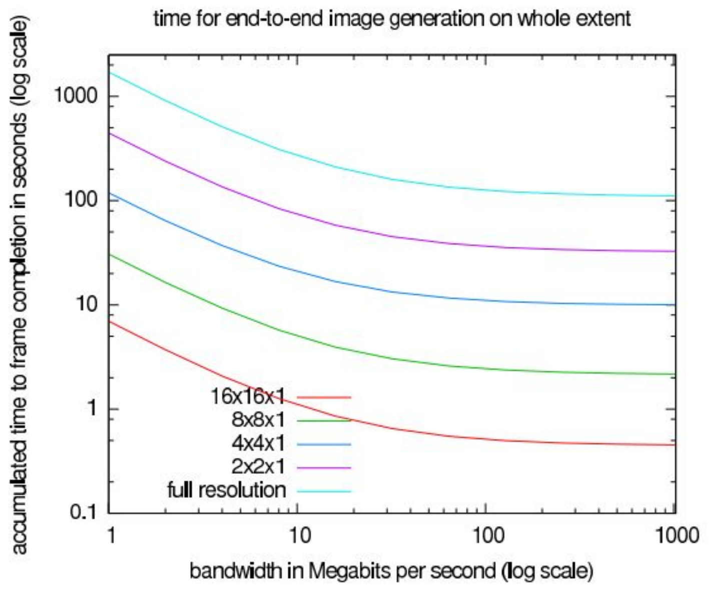
CIELUV color space distance plot from the final image

Results *contd.*

Rendering whole spatial extent of POP dataset (no prioritization)

Full Extent	16x16x1	8x8x1	4x4x1	2x2x1	Full
Render	0.03 s	0.10 s	0.38 s	1.4 s	5.6 s

After the representation is delivered, the user can interact with it continuously.
While the user is interacting with this data,
higher resolution data is continuously being streamed to the client, improving the rendering as the data is acquired.



*render time is taken from the instance of the first incremental render

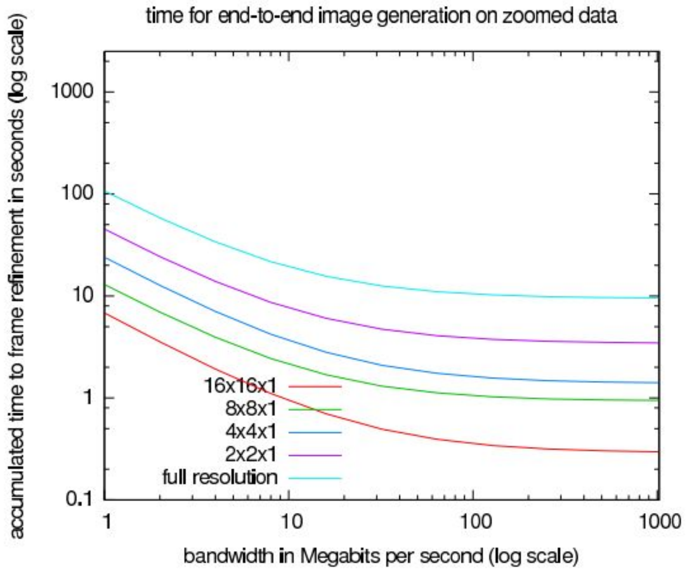
Results *contd.*

<u>Rendering zoomed in portion of POP dataset (with prioritization and culling)</u>

Zoomed	16x16x1	8x8x1	4x4x1	2x2x1	Full
Render	0.03 s	0.03 s	0.05 s	0.09 s	0.27 s

In a zoomed in portion, the data is refined according to the view frustum and streamed to the client which progressively updates the visualization. The rest of the data outside the view frustum is not touched and left at low resolution.

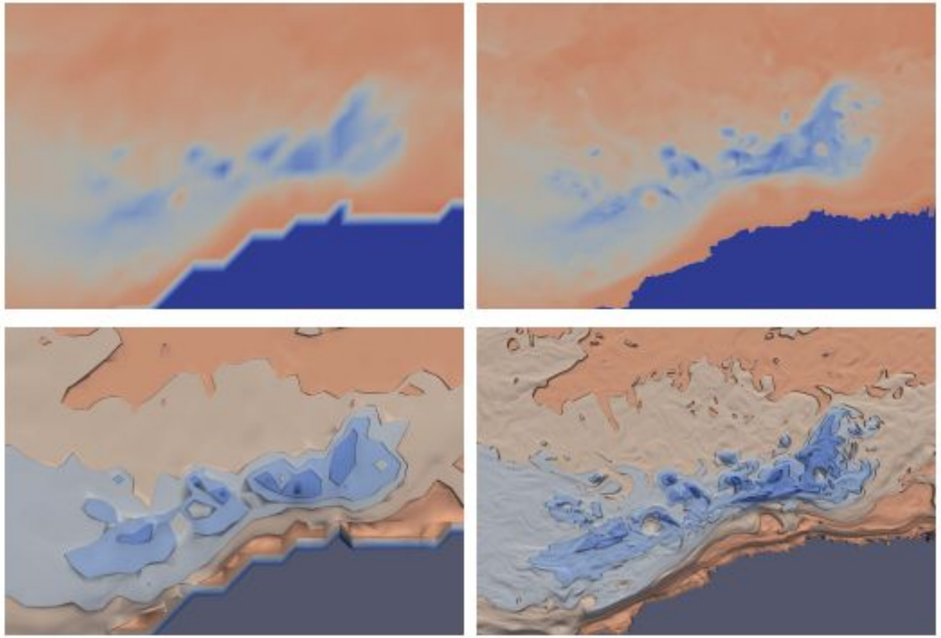
By prioritizing and only sending multiple resolution data for that prioritized view area, the time for its end-to-end completion is significantly reduced since many pieces will fall outside of view frustum.



Multi-resolution system visualizing a zoomed in portion of the POP data set with arbitrary filters and representations in ParaView

Surface rendering

Iso surfacing



Algorithms that are only local modifications or embarrassingly parallel will work as-is with this method.

An example filter that will not work as-is is the streamline filter, since it is a global algorithm. The filter would need to be updated to work in this multi-resolution system.

Thank You

Multi-resolution Distance Visualization System

- Overviews obtained quickly in low resolutions
- Increasing details quickly over time
- Zoomed details prioritized on demand
- Fast client side rendering
- Usable for large scale local visualization, too – possibly integrate into render server, as well (multi-resolution used on supercomputer)