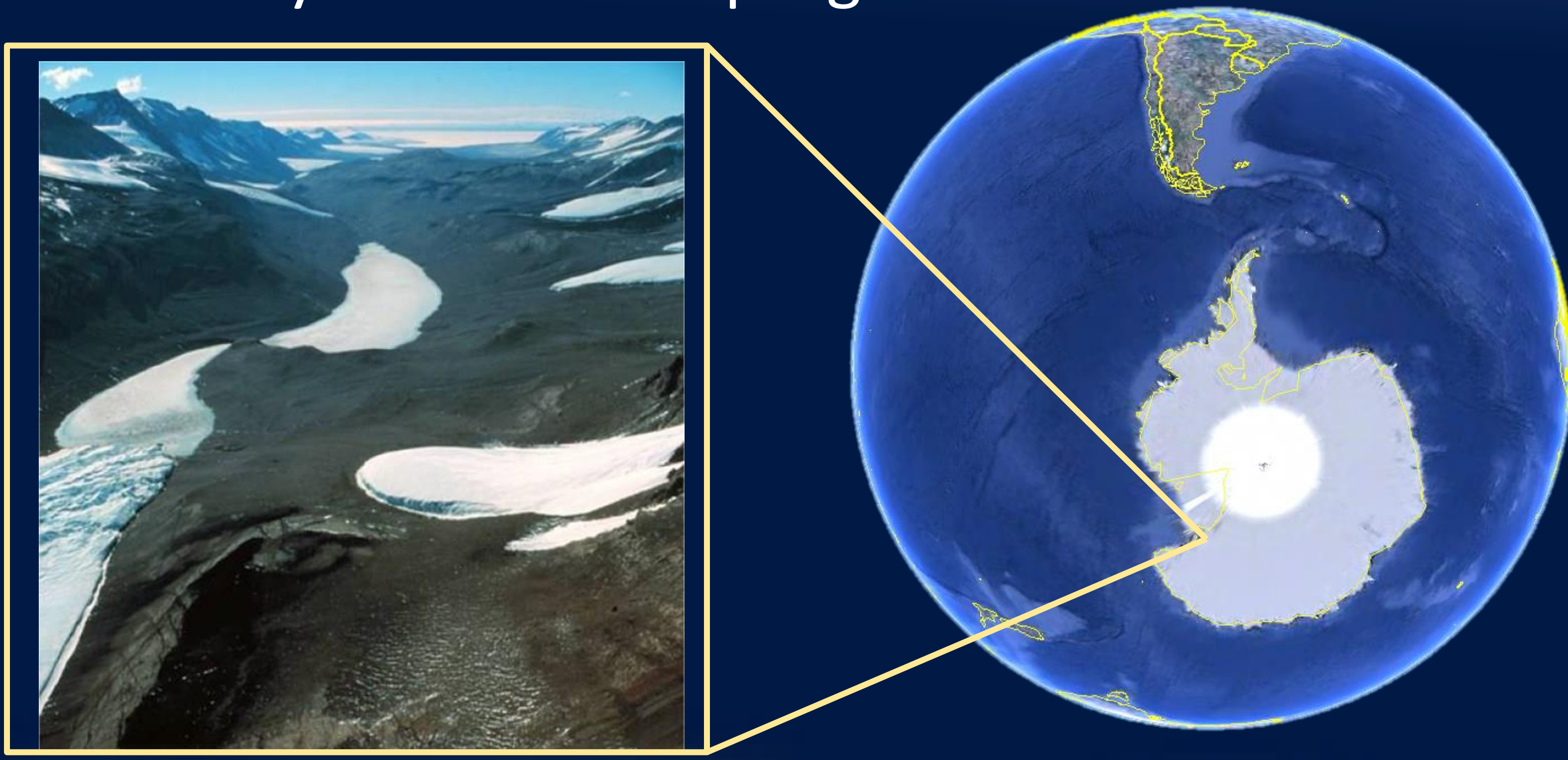
Parallel Sonar Beam Tracing

CS566 Final Project

Alessandro Febretti, Homam Abu Saleem

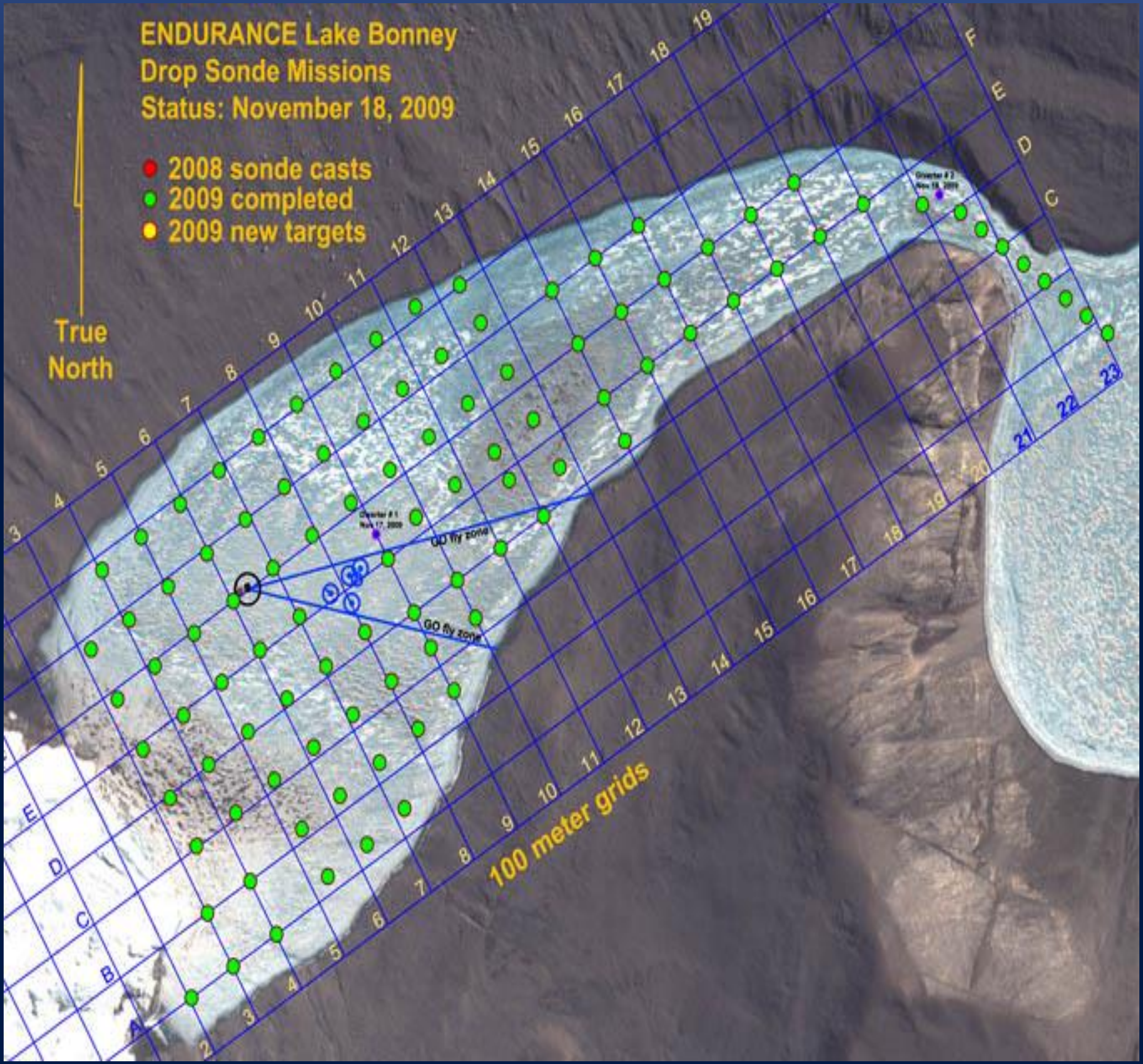
ENDURANCE

- **Environmentally Non - Disturbing Under - ice Robotic ANtarctic Explorer**
- Funded by NASA ASTEP program

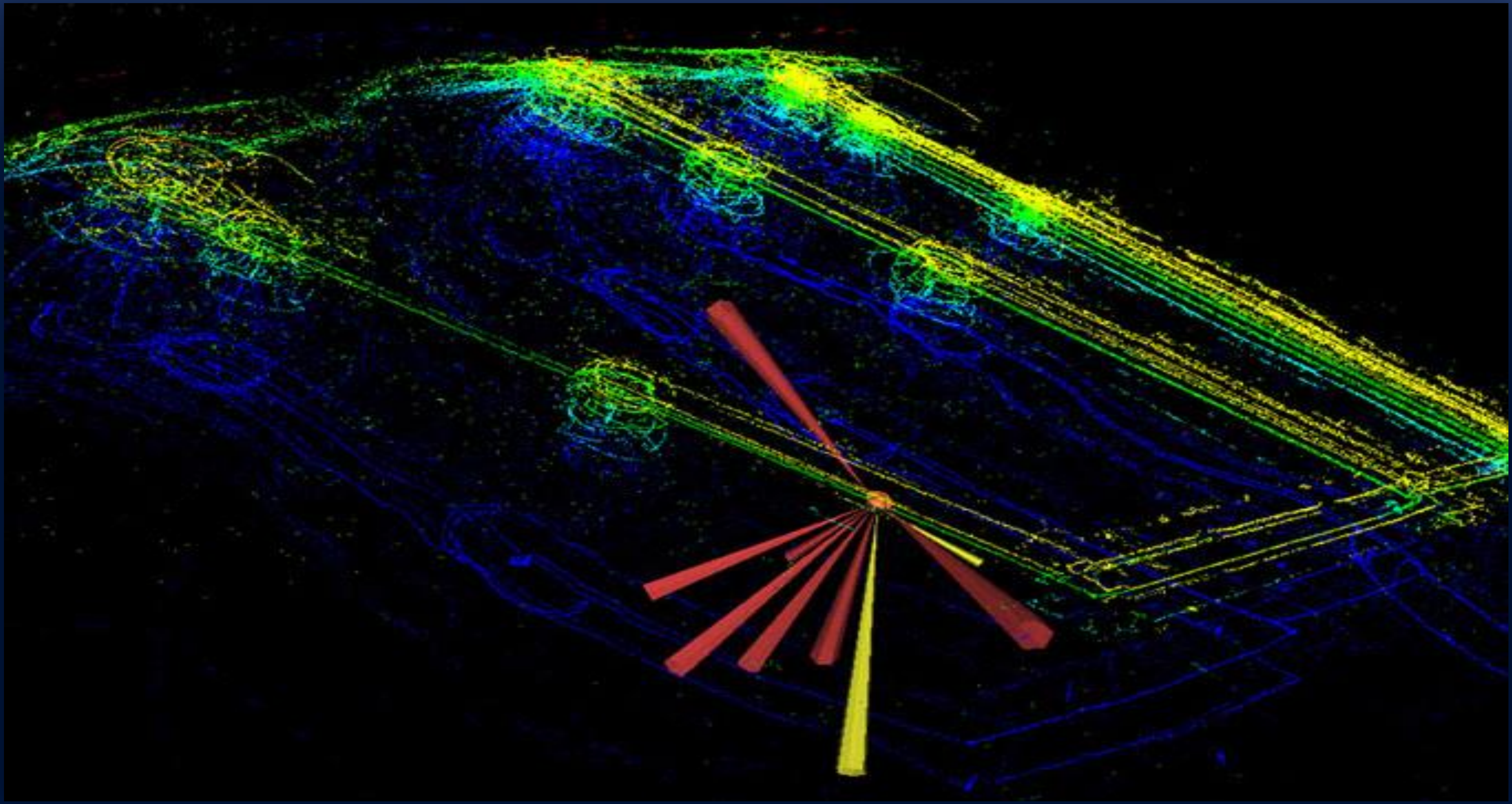



StoneAerospa
com

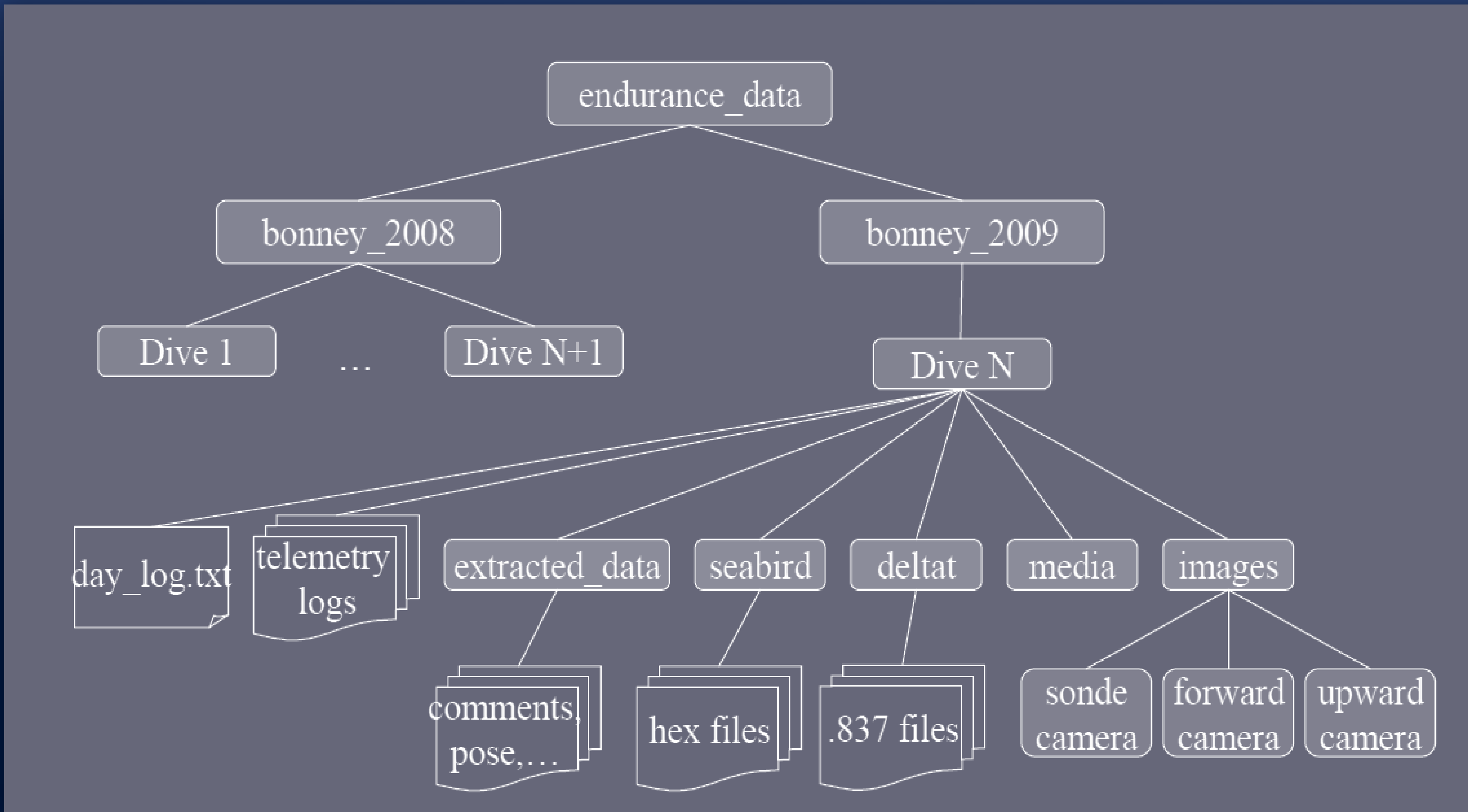
Data collection: Sonde



Data collection: Sonar



Dataset structure



Objectives

- Create a new, high resolution lake bathymetry
  - Compute precise lake volume
  - Simulate lake interaction with the McMurdo Valley System
- Generate a 3D model of the lake/glacier interface

Challenges

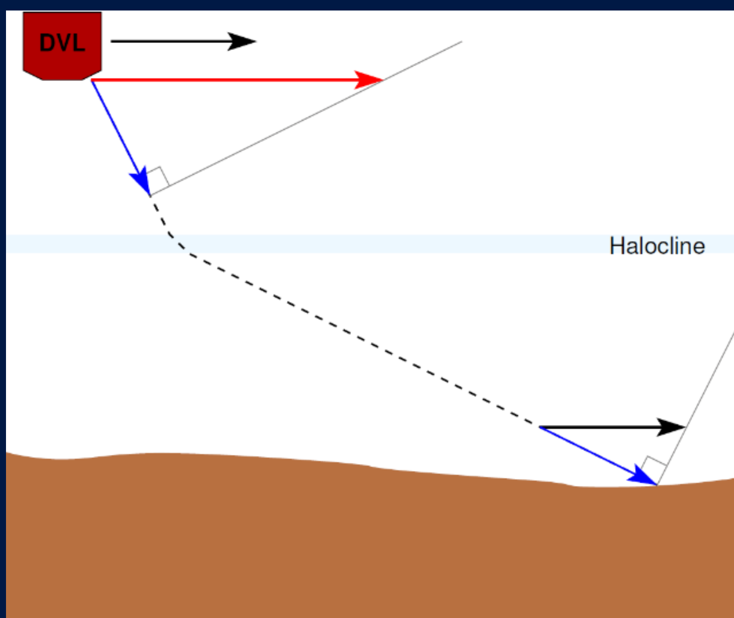
- Sonar data is very imprecise
  - Multipath reflections
  - Water turbidity
  - Navigation / attitude errors
  - Water level variations
  - Lake Chemistry

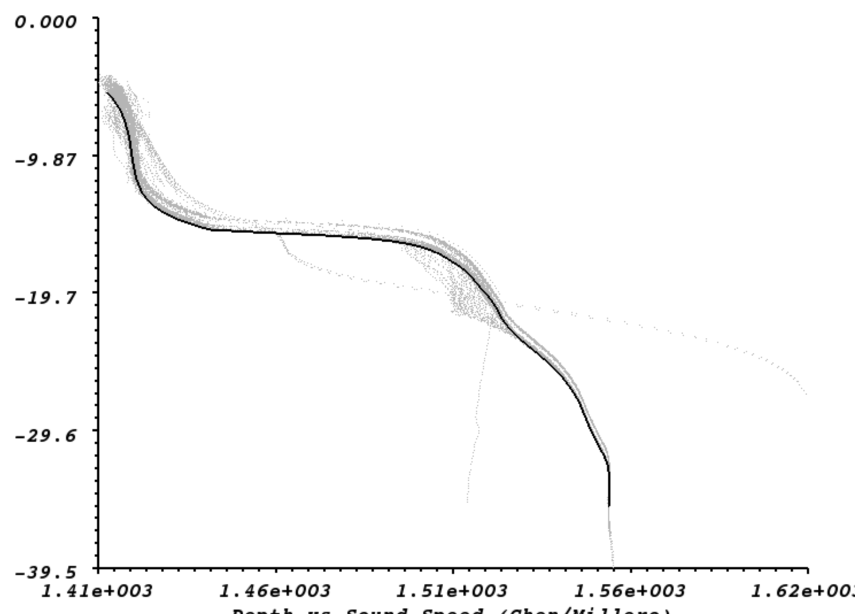
How Sound Travels Through Water

- Basic measurement: return time of sound pressure wave
  - Converted to distance using estimated speed of sound
- Speed of sound in sea water: ~1500m/s
  - estimate changes with temperature, salinity, pressure.
  - Sound speed influences beam paths
    - Refraction (Snell's Law)

How Sound Travels Through Water

- Different sound velocity models (i.e. DelGrosso, Chen-Millero)
  - Tested in ocean water only

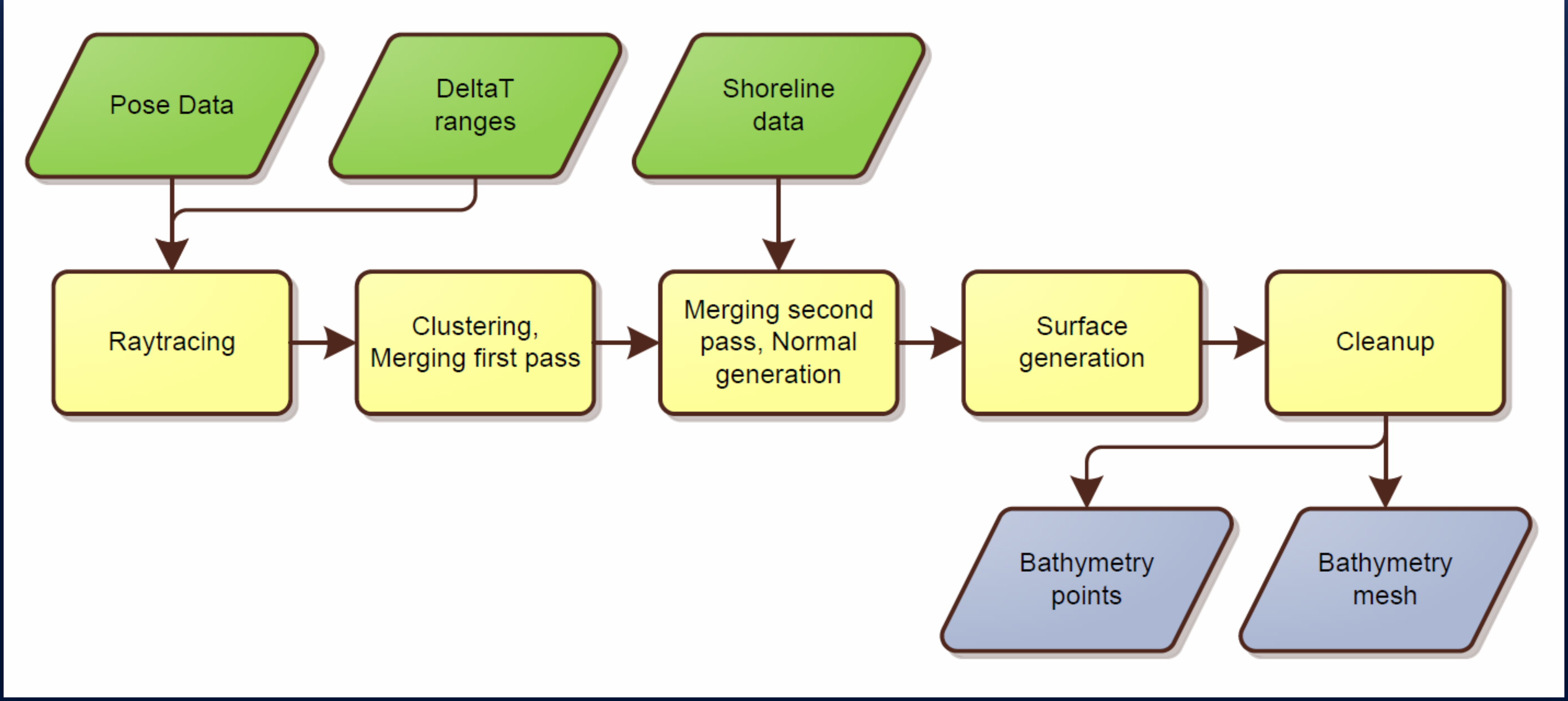




Requirements

- To address objective researchers need to
  - Tweak SVM and beam tracing parameters
  - Apply different navigation corrections
  - Change noise filtering levels
- Big parameter space
  - Need fast tweak-process-visualize pipeline
    - Point cloud is enough
    - Surface reconstruction can be done offline (ex. using Poisson reconstruction)

Pipeline


Pose Data
DeltaT ranges
Shoreline data
Raytracing
Clustering, Merging first pass
Merging second pass, Normal generation
Surface generation
Cleanup
Bathymetry points
Bathymetry mesh

dttools

- Toolkit used for ENDURANCE sonar processing
  - dtrt
    - beam tracer, adapted from MBSystem
  - dtmerge
    - merge, cleanup, normal estimation
  - dtpoisson
    - surface reconstruction
  - dtsample
    - dataset query tool
- We add mpidtrt, mpidtmerge

Ray-tracing Steps

- Load the configuration files
- Load DeltaT files
- Load Position files
- Merge DeltaT points and Position points
- Trim off some noise points in the DeltaT points
- Calculate distance using sound velocity property, orientation, and position configurations for DeltaTpoints / P
- Generate 3D coordinates using the distance, position and orientation configurations
- Write the coordinates to the output file

Parallel Ray-tracing Steps

- Load the configuration files
- Load the entire DeltaT file to get the data set size
- Load the entire Position file because all of the points are needed for each point in DeltaT
- Each process merges (total DeltaT points / P) and Position points
- Last process does the remainder of DeltaTpoints / P since division will leave a remainder < P
- Trim off some noise points in the DeltaT points
- Calculate distance using sound velocity property, orientation, and position configurations for DeltaTpoints / P
- Generate 3D coordinates using the distance, position and orientation configurations DeltaTpoints / P
- Each process writes the coordinates to a separate output file
- Process 0 merges all of the separate output files into one and deletes the separate output file

Merging DeltaT and Position

- Each DeltaT ping has a time stamp
- Each Position data has a time stamp
- Search the time stamp for the positions and pick the closes one to the time stamp of DeltaT ping
- The position data will be used as position data for the DeltaT ping

Removing Noise

- Each ping has multiple sonar signals sent and receive the reflection
- If a received signal is not within a set filter angle, disregard it
- If a received signal has time that is less than a certain threshold, we disregard it

Parallel Ray-tracing Notes

- At least one process has to load the entire DeltaT and Position files to get the size of data since it is unknown, disadvantage
- The writing part has to be sequential, because when multiple processes write to the same file, there will concurrency problem and some process overwrites another process output
- The sound velocity was calculated by scientists and provided along with the data. It was calculated according to the water profile for each region

Ray-trace Input/Output Samples

- DeltaT: 1259565829.523583 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 39.360 39.520 0.000 39.360 39.200 39.040 39.040 38.880 38.720 38.560 38.400
  38.080 38.080 38.240 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 34.080 34.080 0.000 34.080 33.920 33.920 33.920 33.760 33.760 33.760 33.600 33.600 33.600 33.440
  33.440 33.440 33.280 33.120 33.120 33.120 33.120 32.960 32.960 32.800 32.800 32.800 32.800 32.640 0.000
  32.640 32.480 32.480 32.320 32.320 32.160 32.160 32.160 32.000 32.000 32.000 32.000 32.160 0.000 32.160
  32.000 32.000 32.000 31.840 31.840 31.840 31.680 31.680 31.680 31.520 31.520 31.520 31.520 31.360 31.360
  31.360 31.200 31.200 31.200 31.360 31.360 31.520 31.520 31.680 31.680 31.680 31.680 31.680 31.680 31.520
  31.520 31.520 31.520 31.520 31.520 31.360 31.360 31.040 31.040 32.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 35.520 35.840 35.840
  36.000 36.000 36.640 36.640 36.640 36.640 36.480 36.480 36.480 36.480 36.480 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 45.440 45.600 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

- Position: 1259565829.483745 0.001 0.003 5.548 -3.172 -1.014 5.378

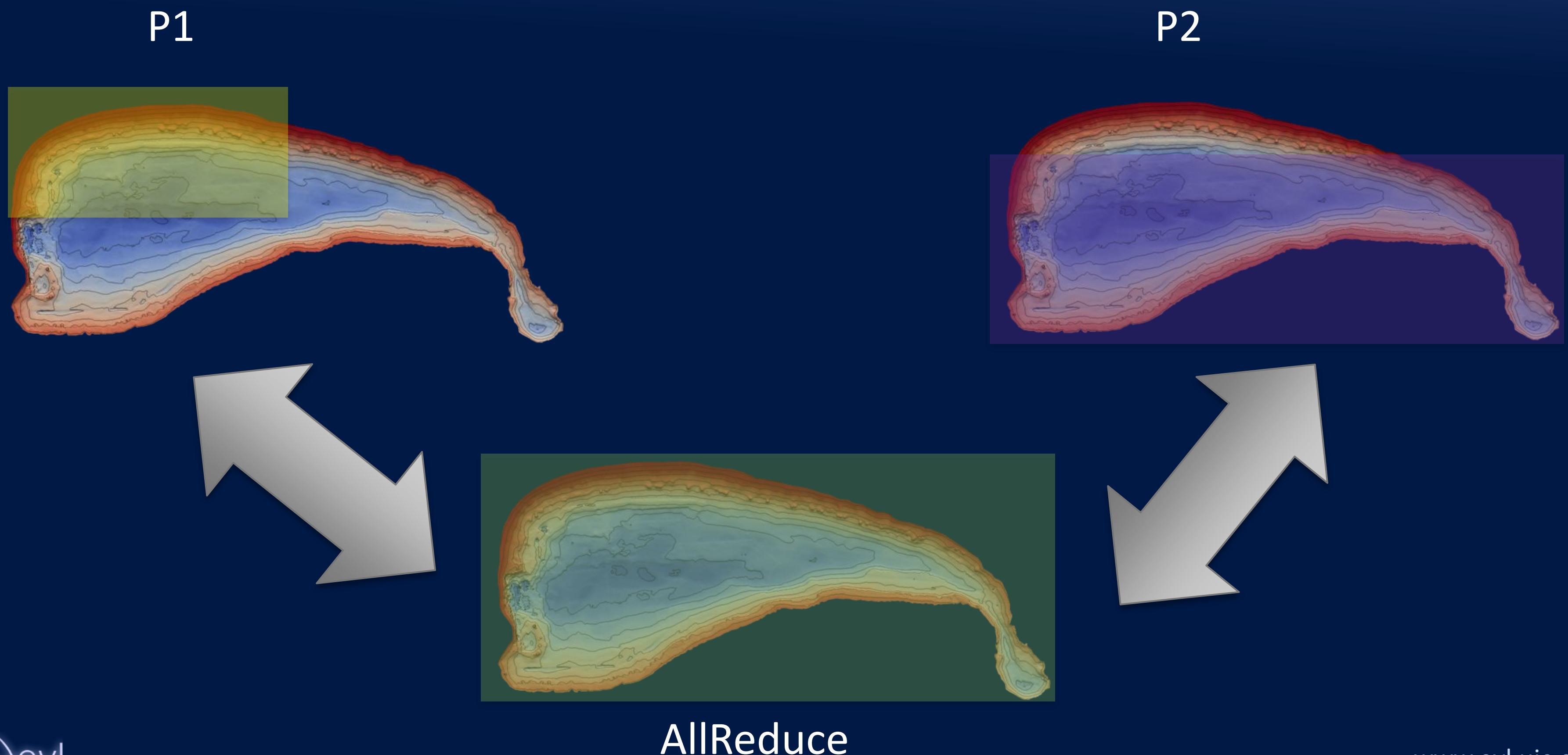
- Output: 0.054576, 0.113037, 62.693869

mpidtmerge: steps

- compute bounds
- merge point set into local grid
- distribute bins
- compute final points
- write points

compute bounds

- Each node works on subset of points
- MPI_AllReduce to compute global min/max

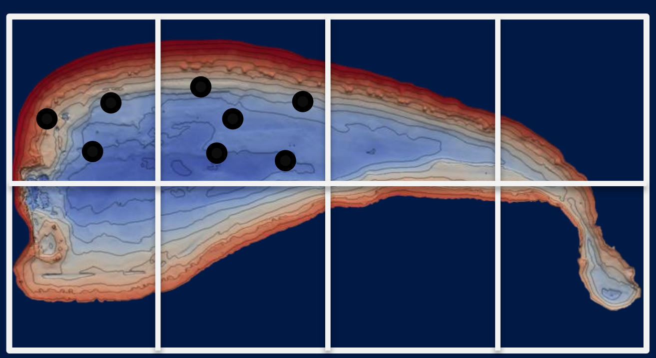


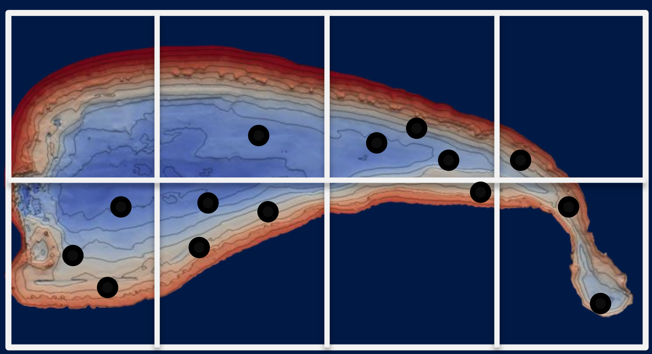
evl

www.evl.uic.edu

merge point set into local grid

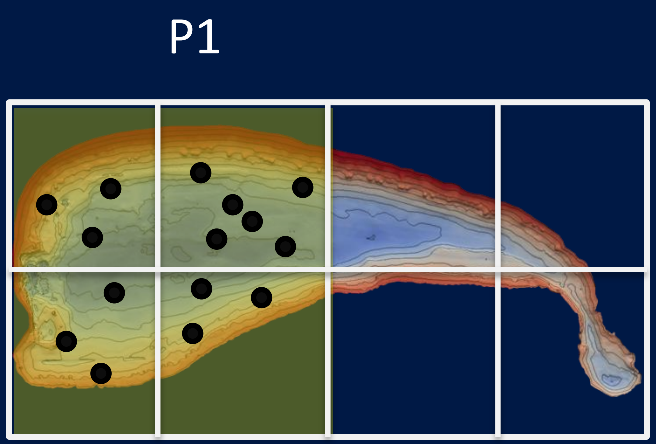
- Each node works on same subset of points again
- global min/max used to create binning grid
  - use octree lookup to quickly find target bin for points
- Accumulate position, normal, num. points for each bin

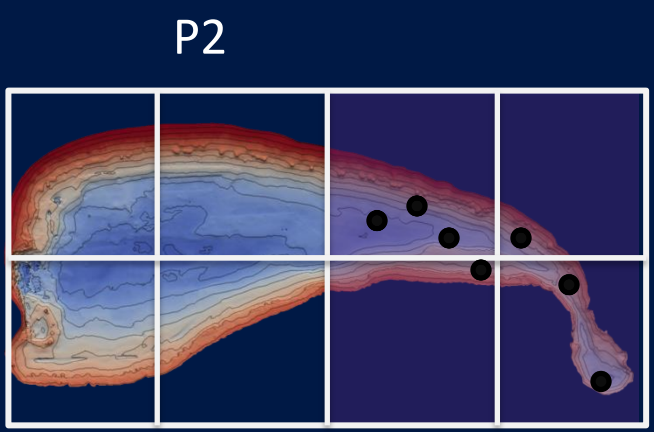




distribute bins

- Each node is assigned a subset of bins
- bins are exchanged and merged
- each node now has global accumulated positions, normal, point counts for bins it owns

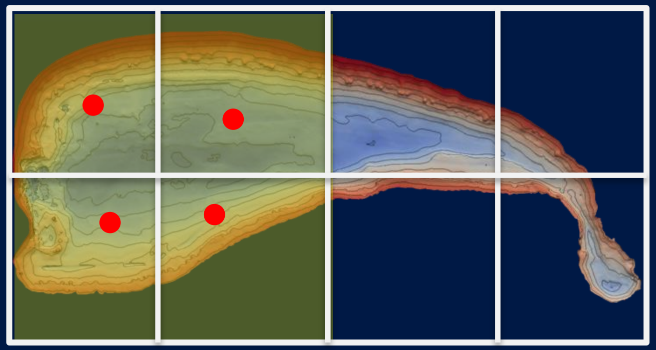




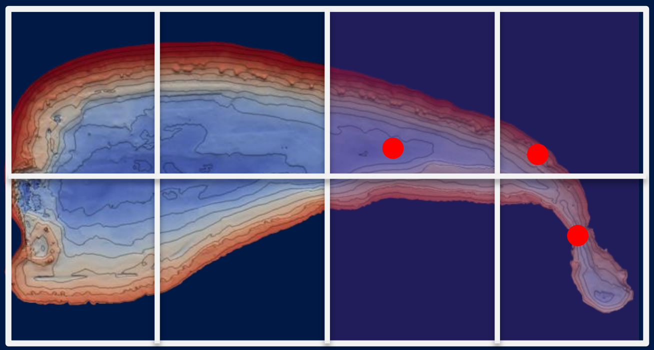
compute final points

- Each node computes final points, discards bins under threshold.
- each node now has part of final solution
- Normal estimation optionally happens here

P1

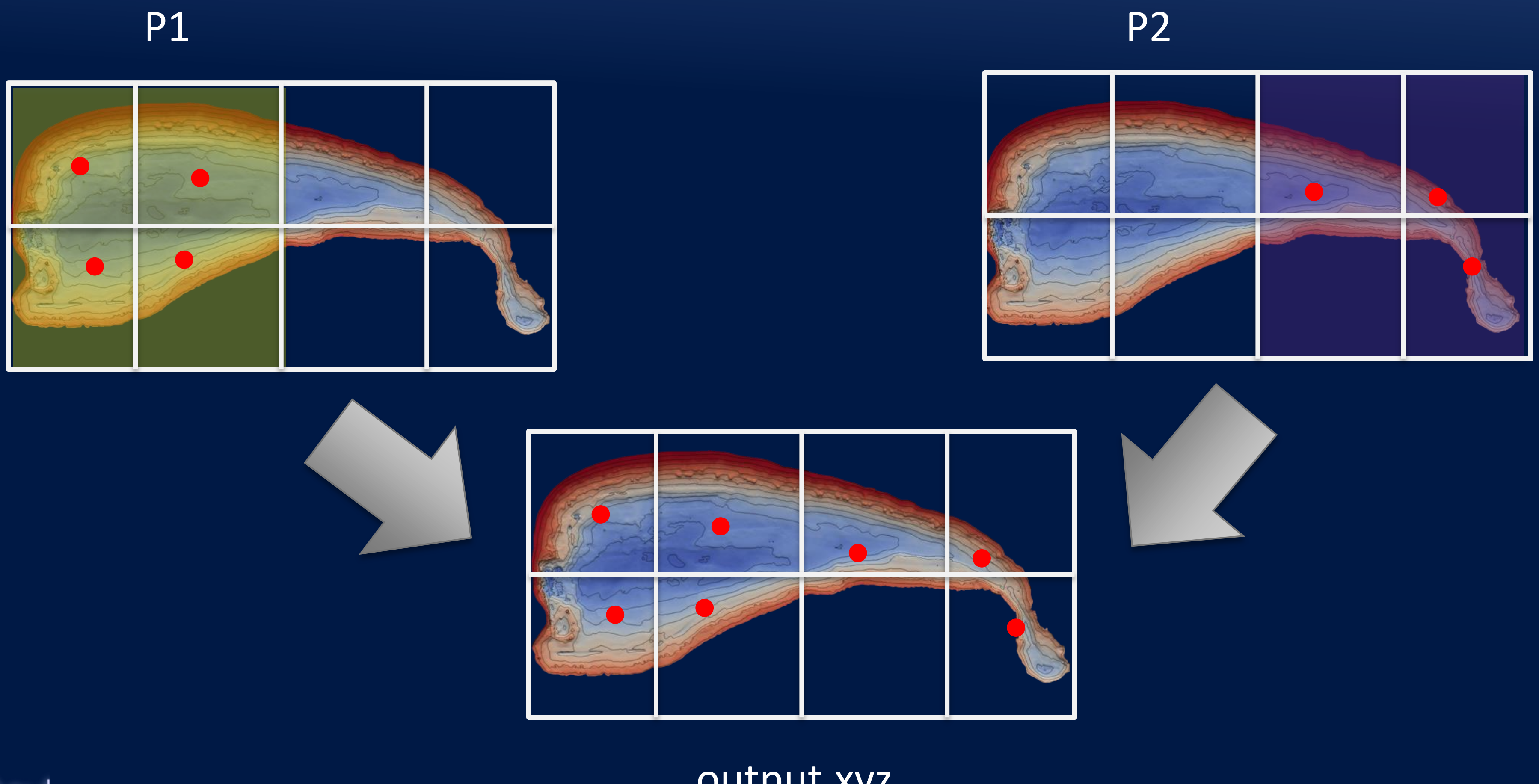


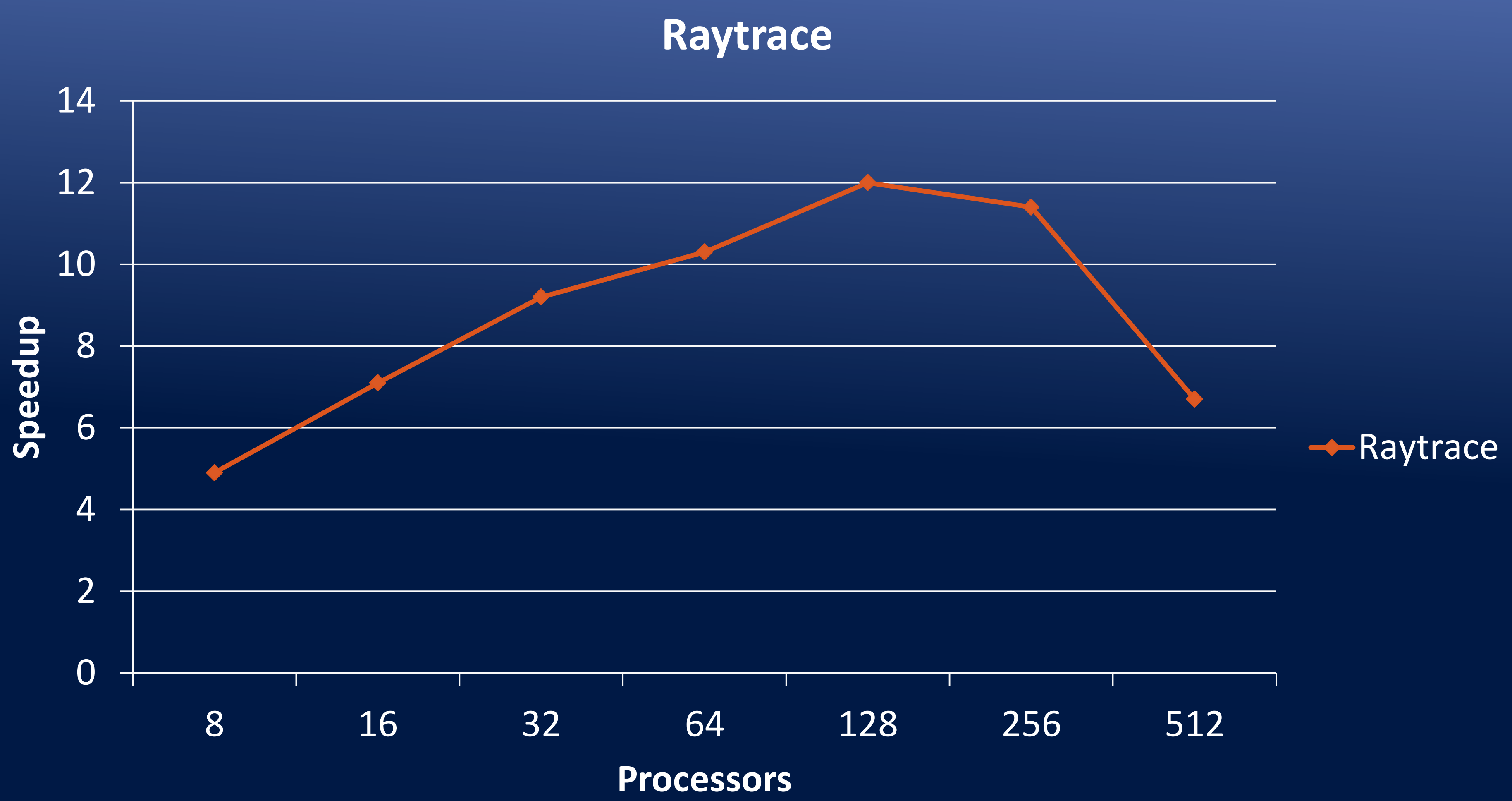
P2



write output

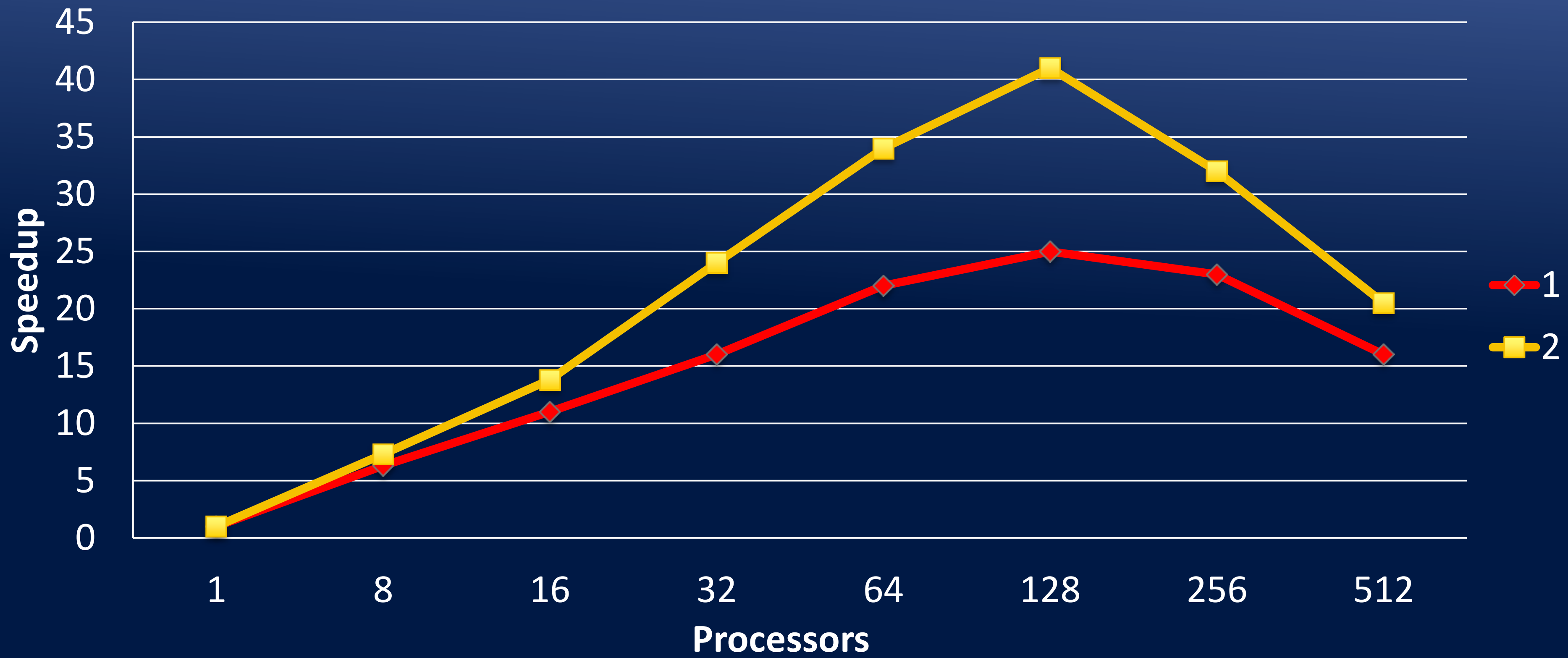
- Each node appends its part of solution to output file
- Sequential. First node overwrites. Other nodes append.





- Run on EVL Lyra cluster (36 machines, 16 cores)
- Best speedup (P=128): 128x
- Sequential time: ~45.5 minutes. Best parallel time: ~3.9 minutes
- This speedup is for processing each ping of the data set, most of the time we do no need to process each ping, we process something like 1 ping out of 10 pings, so this time would be much less than the above

Merge Speedup for 1m and 2m voxel size

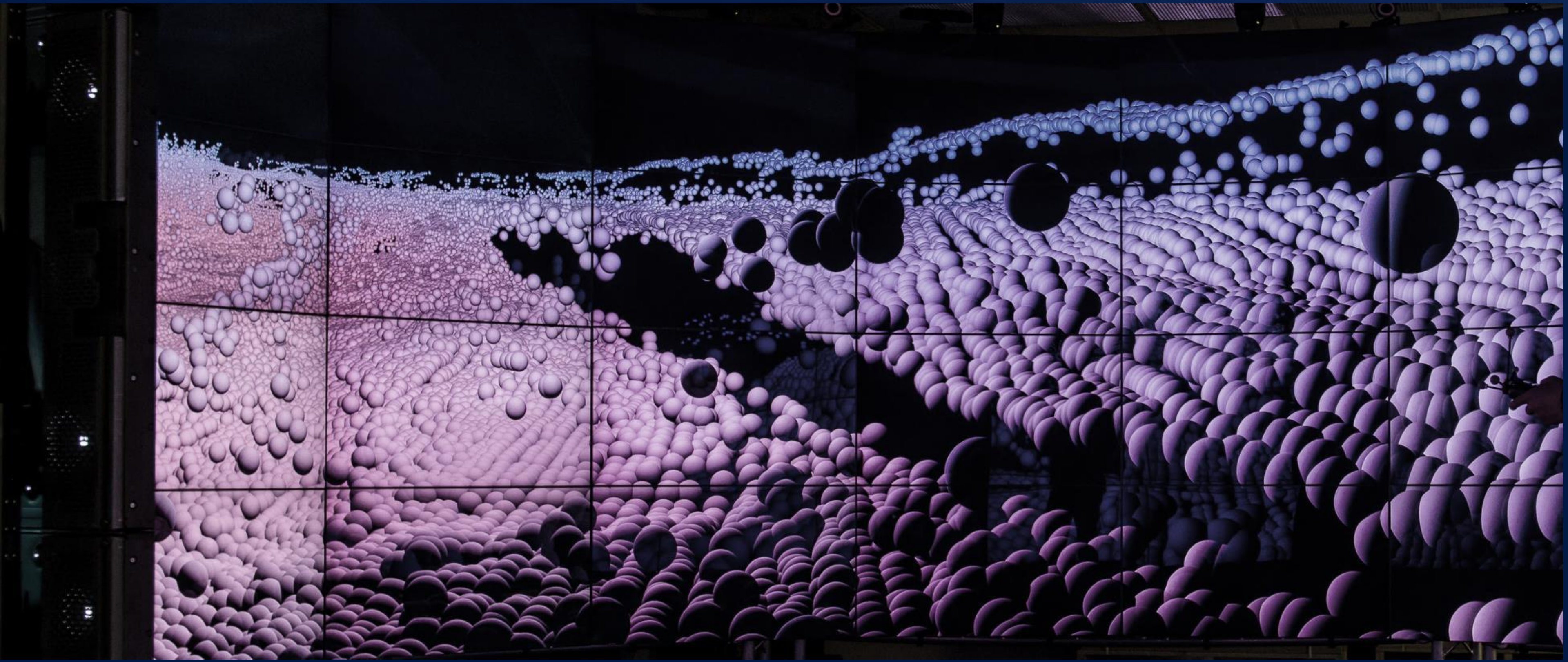


- Run on EVL Lyra cluster (36 machines, 16 cores)
- Best speedup (P=128): 25-40x
- Sequential time: ~3.5 minutes. Best parallel time: ~5 seconds
- Entire dataset (~200Mil points) can be reprocessed almost in real time

Visualization

Visualization

- xyz data copied to GPU memory as-is
- Geometry shader generates user facing quads from points
- Fragment shader rasterizes sphere within quad
- Scales to a few mil points
- Future work: LOD / spatial optimizations



Future work

- Import full, unmerged data into visualization
- Fast query of visualized data
- Mark regions as noisy, feed back to merge step, reprocess on-the-fly
- Integrate with other data sources
  - Chemistry readings
  - AUV position
  - Map