GPU Acceleration for Seismic Interpretation Algorithms

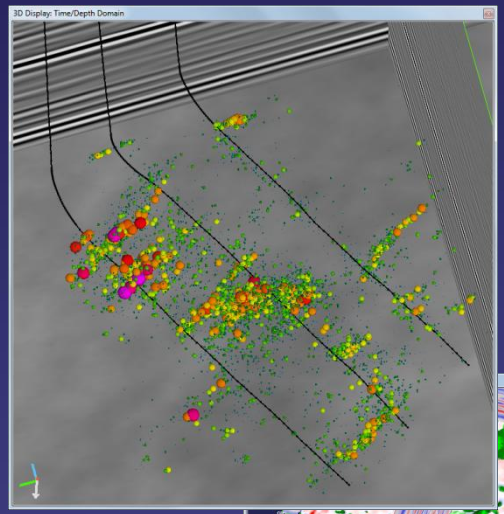
Jon Marbach, Ph.D.
Pate Motter
TerraSpark Geosciences

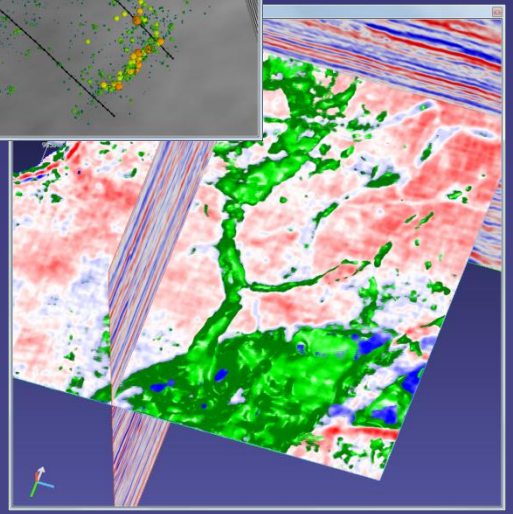
GPU Acceleration for Interpretation

- Talk overview
  - Who am I? (~1min)
  - Who is TerraSpark? (~1min)
  - 3 Key Interpretation Algorithms (~5min)
    - Horizon Orientation (Dip/Strike)
    - Volumetric Curvature
    - Automated Fault Extraction (Fault Enhancement)
  - GPU Acceleration of these algorithms (~5min)
  - Lessons Learned (~5min)

GPU Acceleration for Interpretation

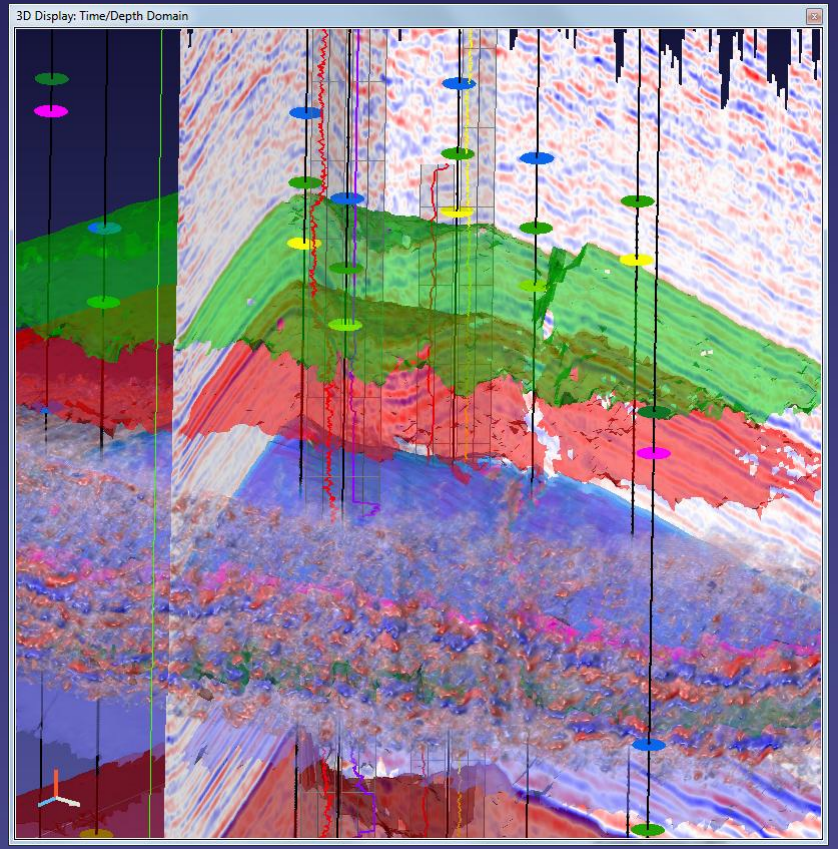
- Who am I?
  - Software Architect / Engineer, not a Geoscientist
    - Couldn't tell a Horst from a Graben
  - Background in graphics (with OpenGL)
    - I love GPUs!
  - An OpenCL user
    - Not a CUDA expert
  - Under pressure to get results quickly
    - I go for quick-wins
    - Might not know every trick in the book





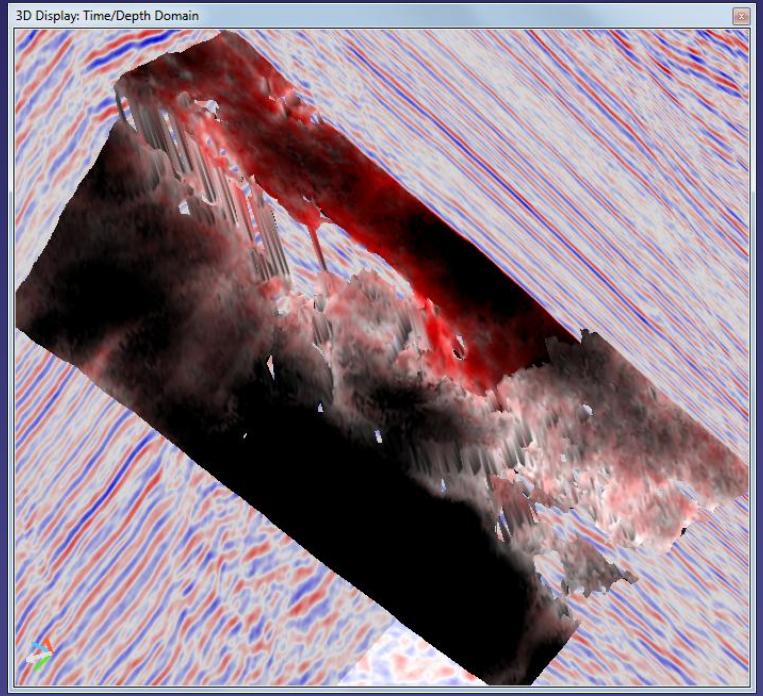
GPU Acceleration for Interpretation

- Who is TerraSpark?
  - Maker of the seismic interpretation package Insight Earth™
    - Workstation-class application
  - Specialize in interpretation of Faults, Salt, Stratigraphy, and Shale Plays
  - Time-saving tools for interpreters
  - “True 3D” interpretation approach
    - Compute-intensive attribute calculations
  - Visualization-oriented



GPU Acceleration for Interpretation

- Why GPU Acceleration?
  - 3D “Image” Data
  - Algorithms are image-processing inspired
  - Algorithms are data-parallel
  - Algorithms tend to “gather” many neighboring samples
  - Interpolation is fast
  - or maybe the real reason is…



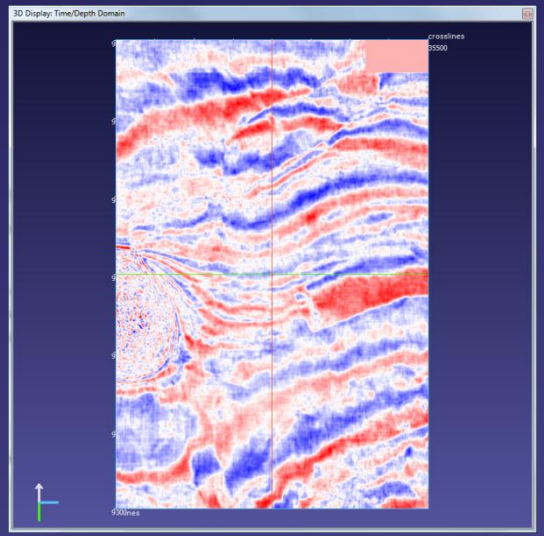
GPU Acceleration for Interpretation

- Why GPU Acceleration?

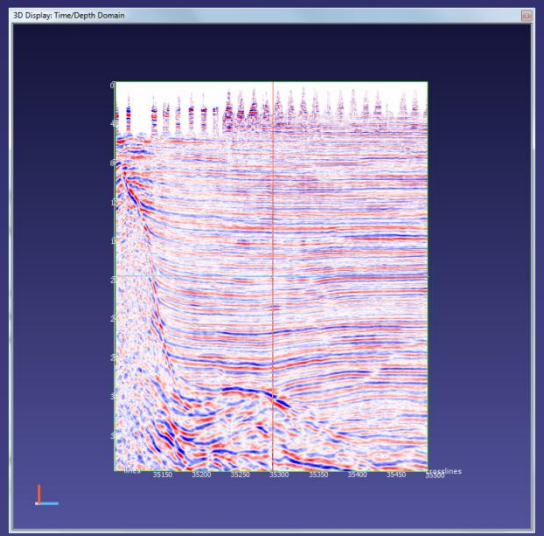
Source: http://graphjam.memebase.com/2008/10/13/song-chart-memes-time-perception-of-1-minute/

GPU Acceleration for Interpretation

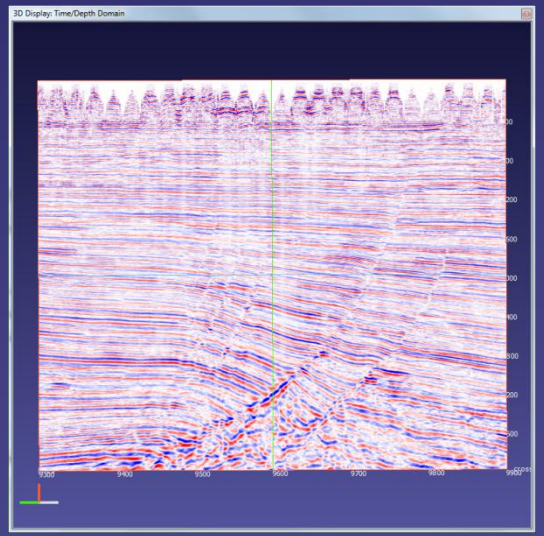
- 3D Seismic Dataset: Eugene Island 175 (“Half Dome”)
  - Offshore Gulf of Mexico – features a salt dome, complex faulting, many stratigraphic features



Time Slice



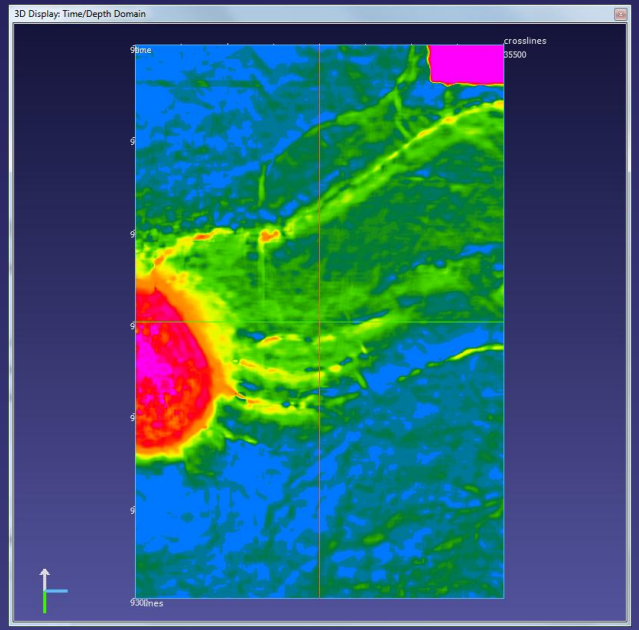
Inline



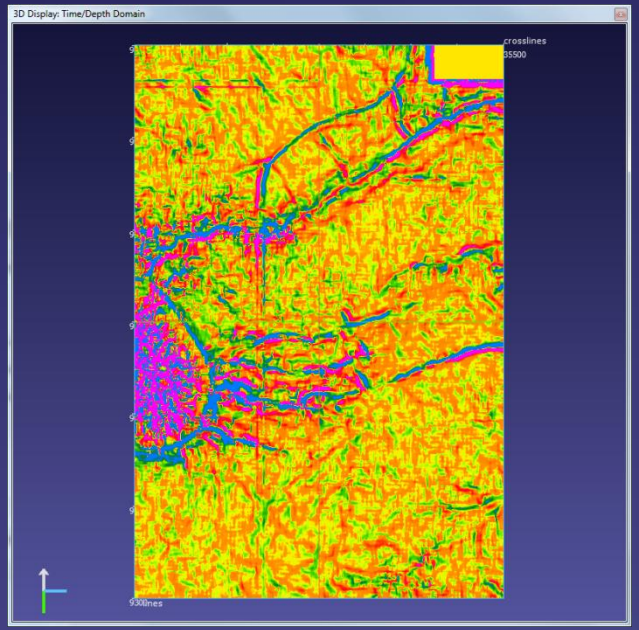
Crossline

GPU Acceleration for Interpretation

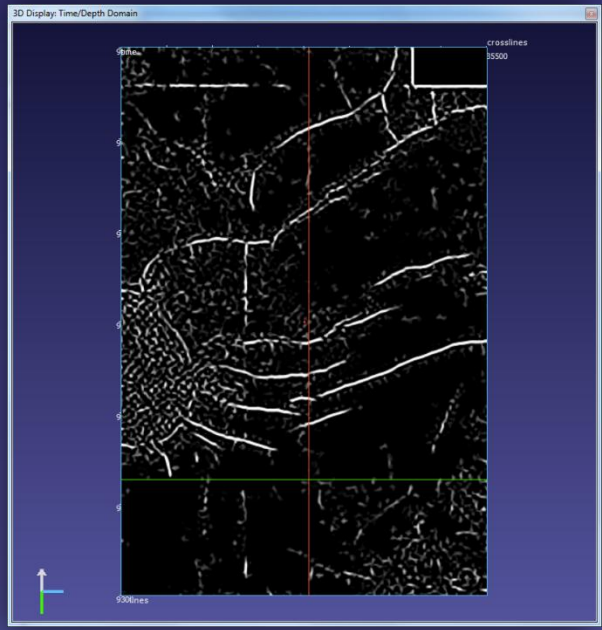
- 3 Key Interpretation Algorithms / Volumes



Horizon Orientation (Dip/Strike)



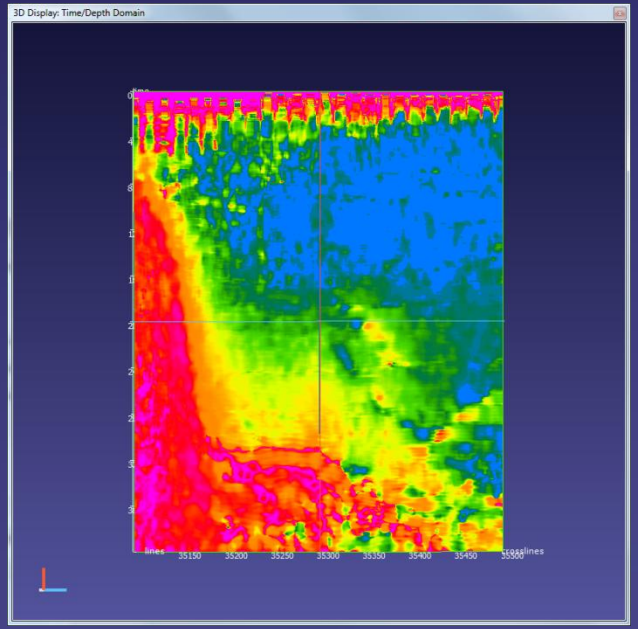
Curvature



Fault Enhance Volume

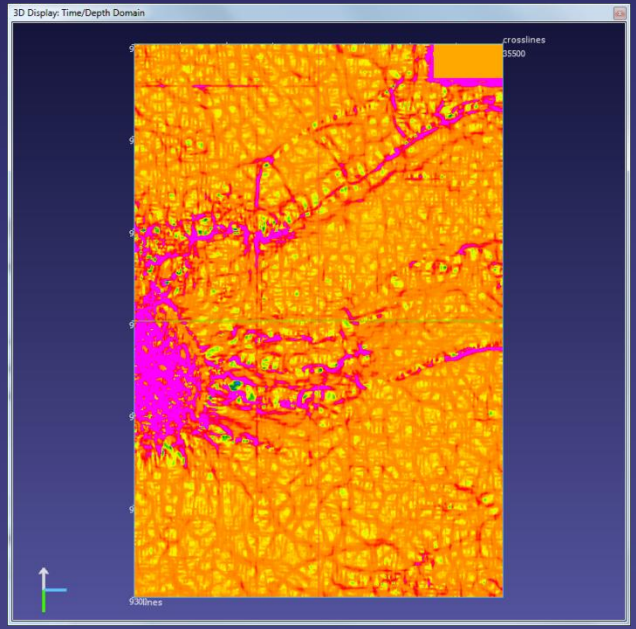
GPU Acceleration for Interpretation

- Horizon Orientation (Dip/Strike)
  - Why is it important?
    - Guides other “Structurally Oriented” attribute calculations
    - Indicates trends in the volume
  - Why is it compute-intensive?
    - Calculates gradient structure tensors
    - Relies on an eigensystem solver
    - Smooths tensors over a large area
  - Why accelerate it?
    - CPU runtimes on the order of tens of minutes
    - Creates workflow bottlenecks



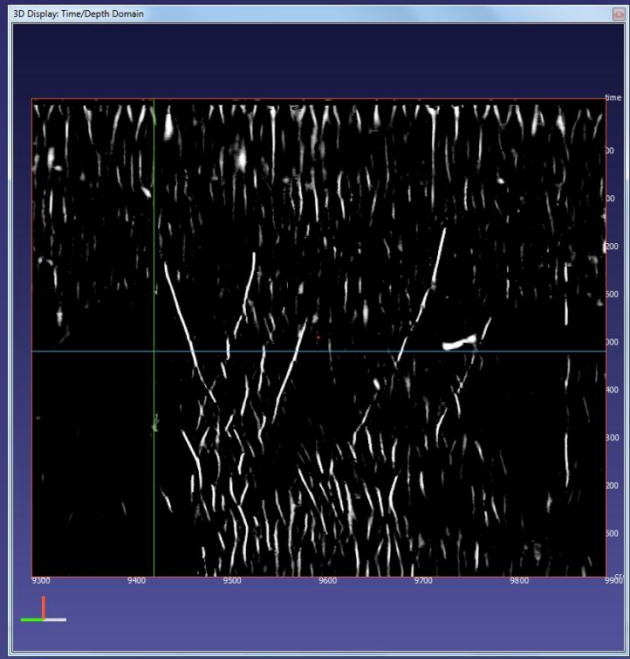
GPU Acceleration for Interpretation

- Volumetric Curvature
  - Why is it important?
    - Can reveal subtle stratigraphic features
    - Can reveal small-scale fracturing
  - Why is it compute-intensive?
    - Performs fine-grained radial sampling
    - Operates on vector data
    - Interpolates samples
  - Why accelerate it?
    - CPU runtimes on the order of tens of minutes
    - Increase turn-around time on parameter testing



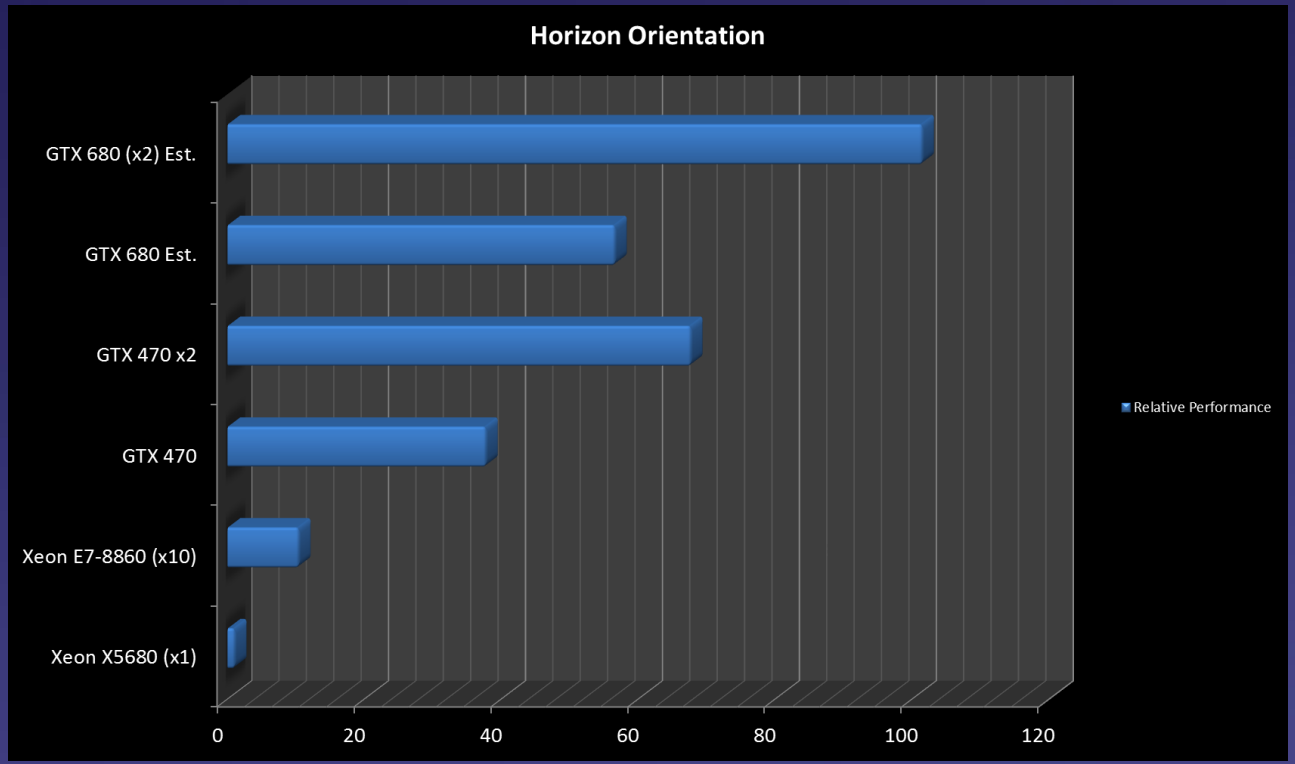
GPU Acceleration for Interpretation

- Fault Enhance (Automated Fault Extraction, aka “AFE”)
  - Why is it important?
    - Identifies and eases interpretation of potentially 1000s of faults
    - Huge savings of interpreter time
  - Why is it compute-intensive?
    - Samples large neighborhoods of voxels in 3D (a radial search)
    - Interpolates samples in 3D
  - Why accelerate it?
    - CPU runtimes on the order of hours or *days*!
    - Sensitive to parameterization, but iteration impractical



GPU Acceleration for Interpretation

- Horizon Orientation GPU Performance

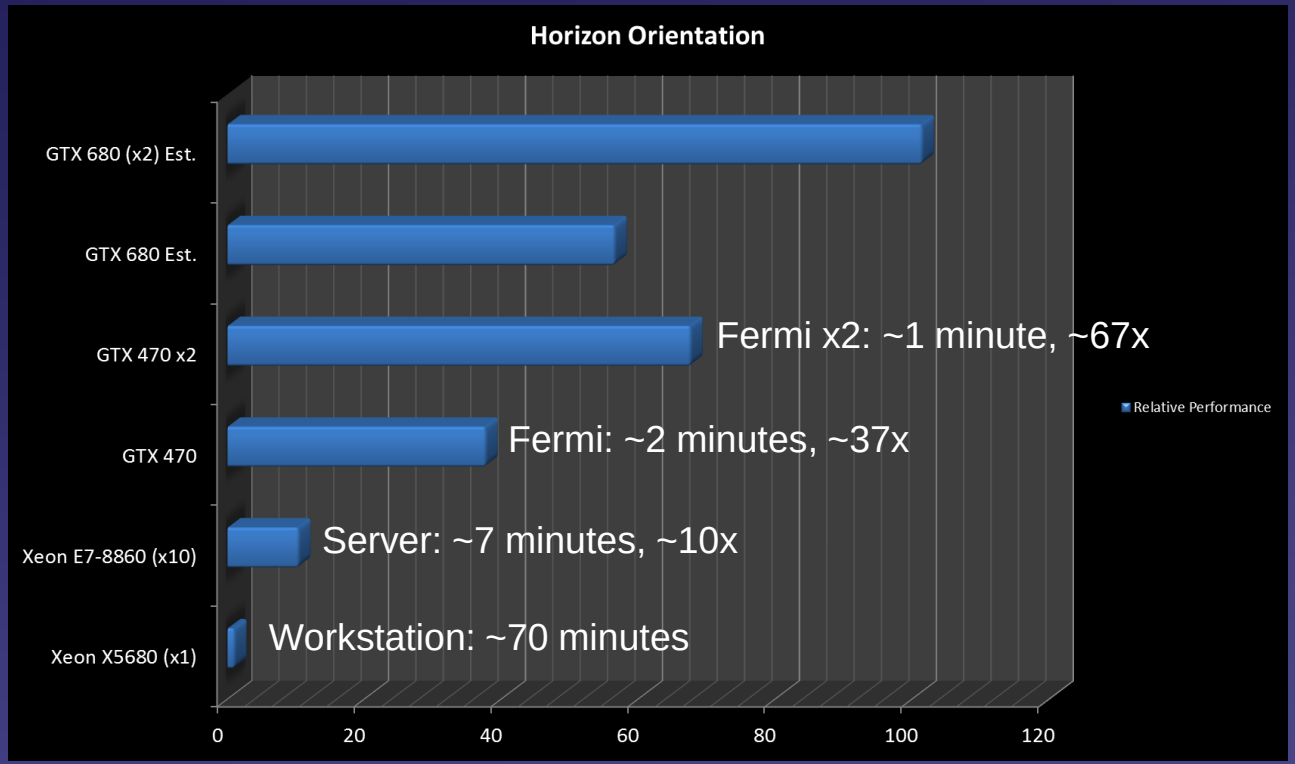


Base performance increase: ~250x (1 Fermi GPU vs 1 Intel Westmere core)

Dataset is 401x601x1000 (~240 Million samples)

GPU Acceleration for Interpretation

- Horizon Orientation GPU Performance



Base performance increase: ~250x (1 Fermi GPU vs 1 Intel Westmere core)

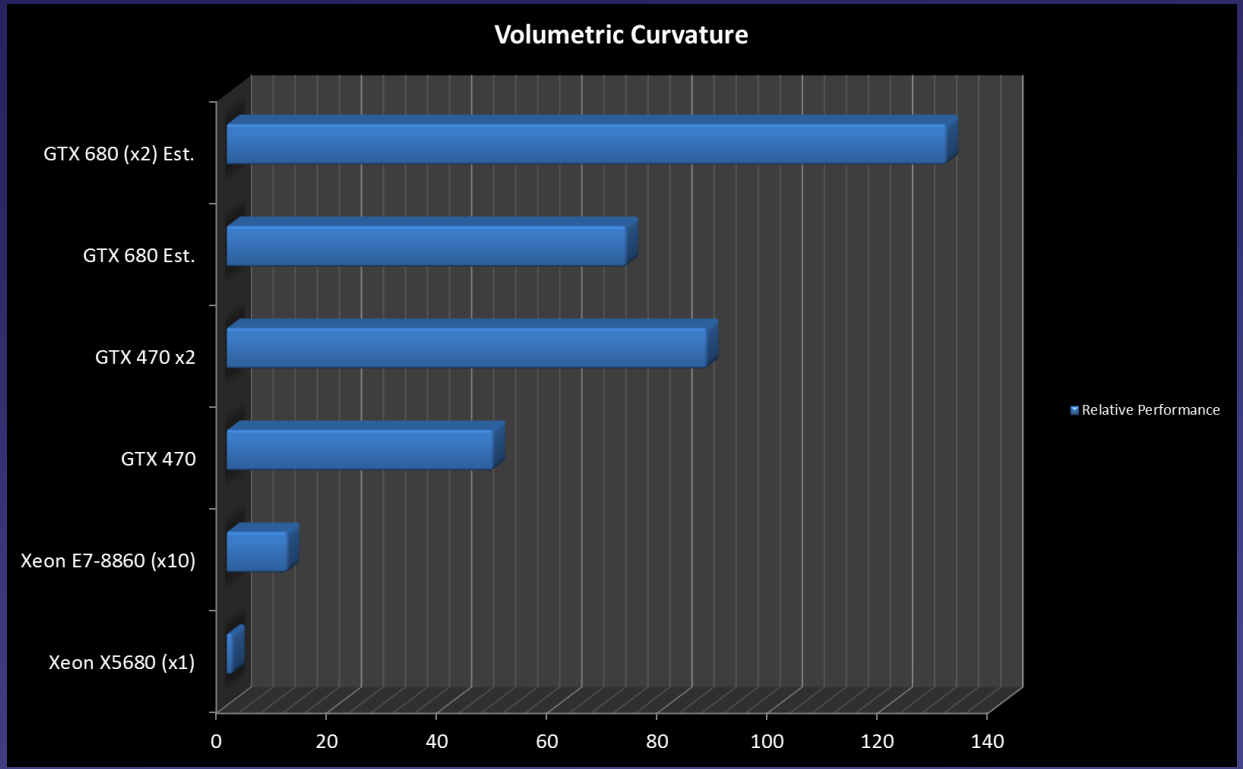
Dataset is 401x601x1000 (~240 Million samples)

GPU Acceleration for Interpretation

- Horizon Orientation Technical Challenges
  - Memory consumption
    - Requires a buffer that's 3*3*30*X*Y*4
    - Solution: Break down processing into smaller, overlapping regions
  - Precision
    - Single-precision Eigensystem solver was producing "measels"
    - Solution: increase precision of the solver to double
    - Negligible Performance impact (cost is hidden?)
  - Memory layout
    - CPU-style 3x3 tensor layout not compatible with coalesced memory reads
    - Solution: lay out all [0][0] elements contiguously, then all [0][1]…

GPU Acceleration for Interpretation

- Volumetric Curvature GPU Performance

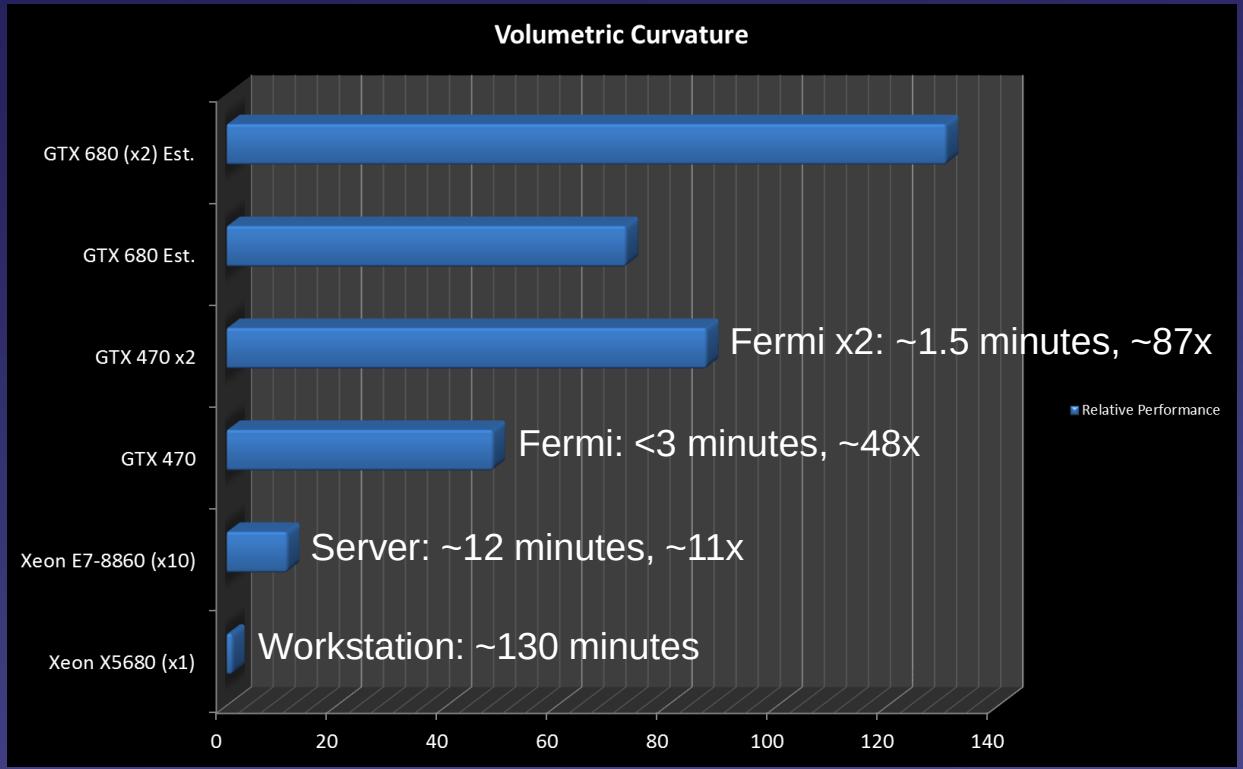


Base performance increase: ~300x (1 Fermi GPU vs 1 Intel Westmere core)

Dataset is 401x601x1000 (~240 Million samples)

GPU Acceleration for Interpretation

- Volumetric Curvature GPU Performance



Base performance increase: ~300x (1 Fermi GPU vs 1 Intel Westmere core)

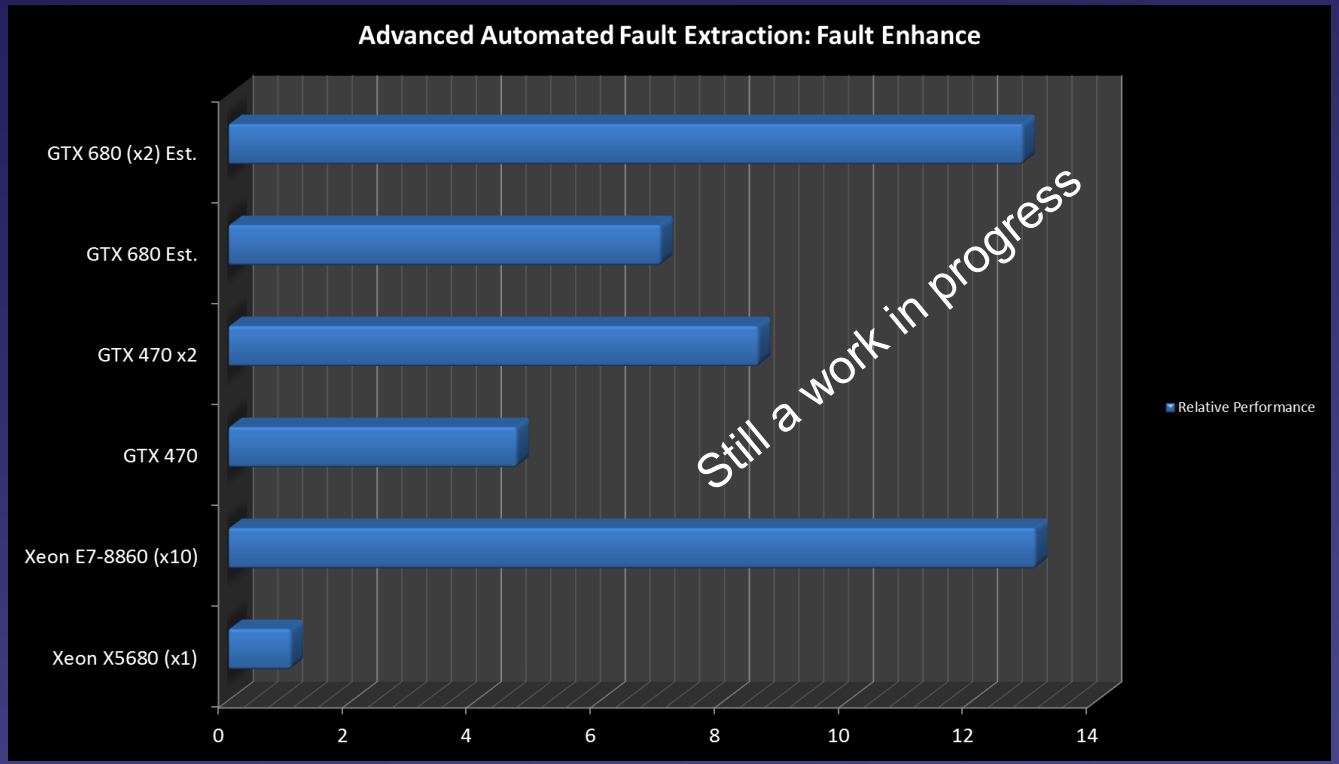
Dataset is 401x601x1000 (~240 Million samples)

GPU Acceleration for Interpretation

- Volumetric Curvature Technical Challenges
  - This one was pretty straightforward!
  - Problem can be handled with 2D textures / buffers
    - No significant memory consumption issues
  - Vector data + mask embedded in 8-bit RGBA texture
    - Very fast reads and interpolation
    - This is where we get the speedup
  - Have to be careful mixing normalized and non-normalized coords
  - Uses two medium-sized local float arrays
    - “Register pressure”? (TODO: Investigate impact?)
  - Difficult to get identical results from GPU (precision?)

GPU Acceleration for Interpretation

- Fault Enhance GPU Performance



Base performance increase: ~40x (1 Fermi GPU vs 1 Intel Westmere core)

Dataset is 401x601x1000 (~240 Million samples)

GPU Acceleration for Interpretation

- Fault Enhance Technical Challenges
  - Easily our most complex algorithm – 1000s of lines of code
    - 3D Textures required (decomposition required)
    - Many stages of calculation needing to be ported
      - Or develop hybrid CPU/GPU strategy?
    - Read-write memory dependencies
      - Can be split up by ping-ponging, but changes the results slightly
      - This is a “Is the CPU code really right?” moment
    - 30-40+ kernel arguments per kernel

GPU Acceleration for Interpretation

- Fault Enhance Technical Challenges (cont.)
  - Requires a median-finding algorithm
    - Solution: Use a brute-force iterative method
  - Includes a histogram-based rescaling
    - Solution: TODO / Just run on CPU
  - Uses two medium-sized local float arrays
    - Register pressure? TODO: Investigate impact
  - Performance not as good as other algorithms
    - Further optimization required
  - 3D search performed at each voxel – Watchdog timer issues
    - Solution: Conservative global work group size

GPU Acceleration for Interpretation

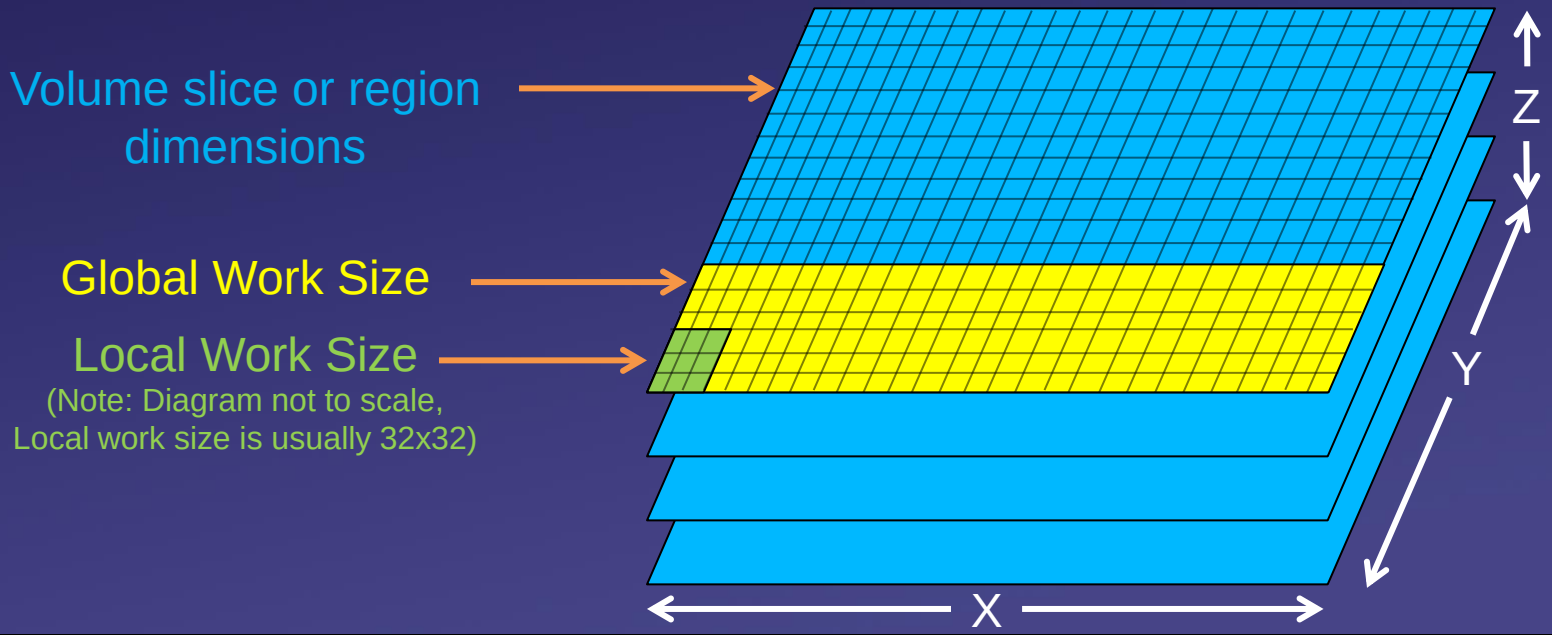
- General Lessons Learned
  - Which algorithms to tackle?
    - The “worst offenders” performance-wise
    - Image-processing based, especially if 3D
    - Ones that are fairly “mature” (not likely to change)
      - …because you’ll need to maintain 2 code paths now
    - But… Don’t take on the hardest one first
  - Who should work on this?
    - For best productivity someone with experience in
      - Graphics or CUDA/OpenCL; the algorithm to be ported; the system into which it fits

GPU Acceleration for Interpretation

- General Lessons Learned (cont.)
  - Strive for identical CPU vs GPU results
    - I sense that Geoscientists are skeptical of GPUs
    - Be ready to give an explanation for differences in results
    - Verify conformance as you go (read back data, dump, and diff)
  - Don’t settle for “OK” GPU performance
    - My first cut is never the fastest
    - Pay attention to NVIDIA’s recommendations
      - Coalesced memory reads
      - “NDRange optimizations” – how much work is sent to the GPU (see next)
    - Revisit optimization strategies (even failed ones) as you optimize

GPU Acceleration for Interpretation

- NDRange optimizations
  - How much work is sent to the GPU at a time

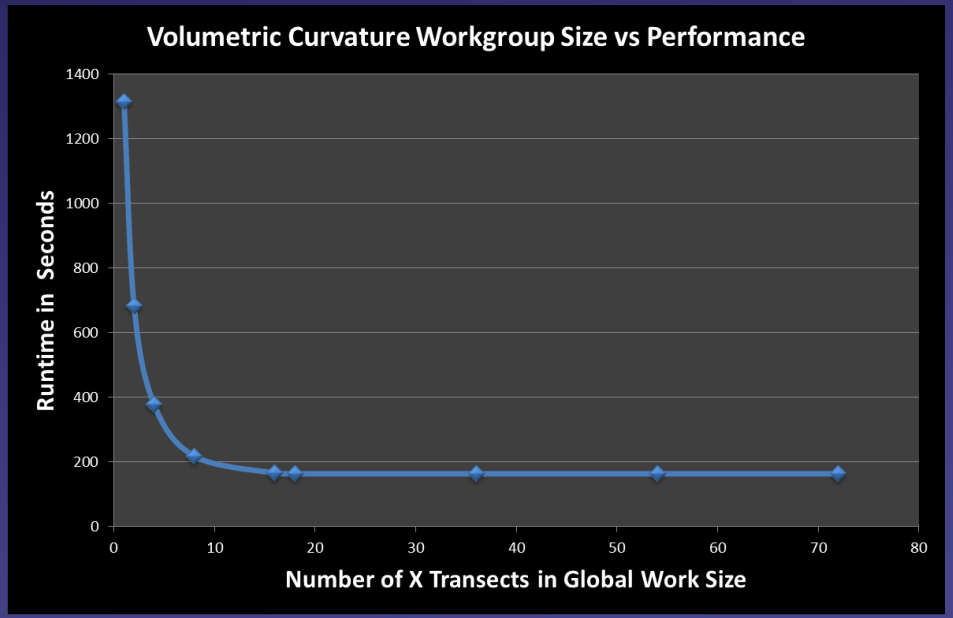


GPU Acceleration for Interpretation

- NDRange optimizations
  - How much work is sent to the GPU at a time
    - We generally issue kernels on *n* x-strips of the volume at a time
    - Offsets into the data handled by a kernel parameter
      - int j = get_global_id(1) + yOffset
      - (Is the kernel offset parameter supported yet?)
    - Must give the GPU enough threads for optimal performance but without running into watchdog (TDR) timeouts
    - *Note:* Definitely query CL_KERNEL_WORK_GROUP_SIZE

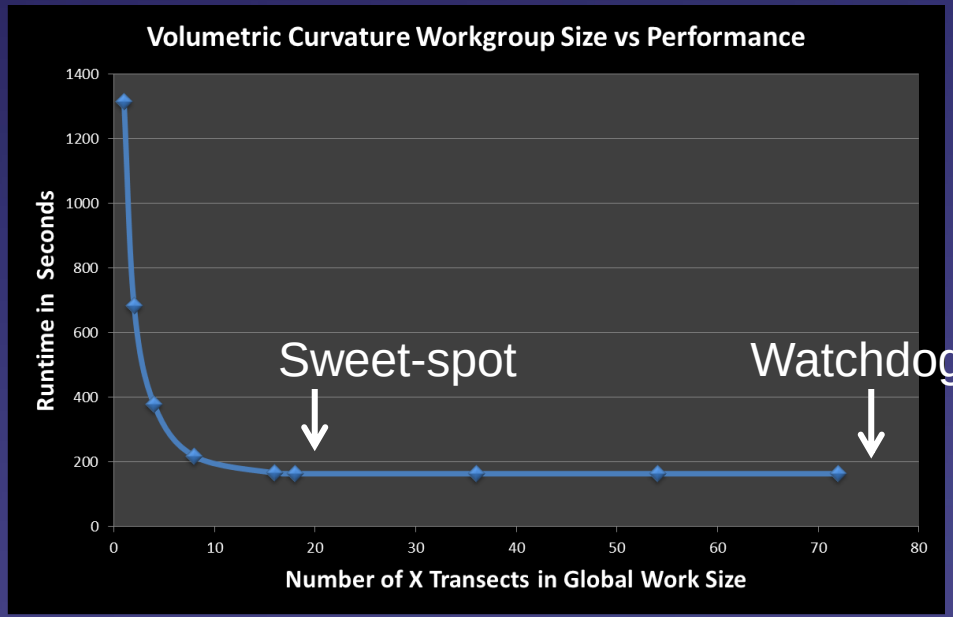
GPU Acceleration for Interpretation

- NDRange optimizations
  - How much work is sent to the GPU at a time



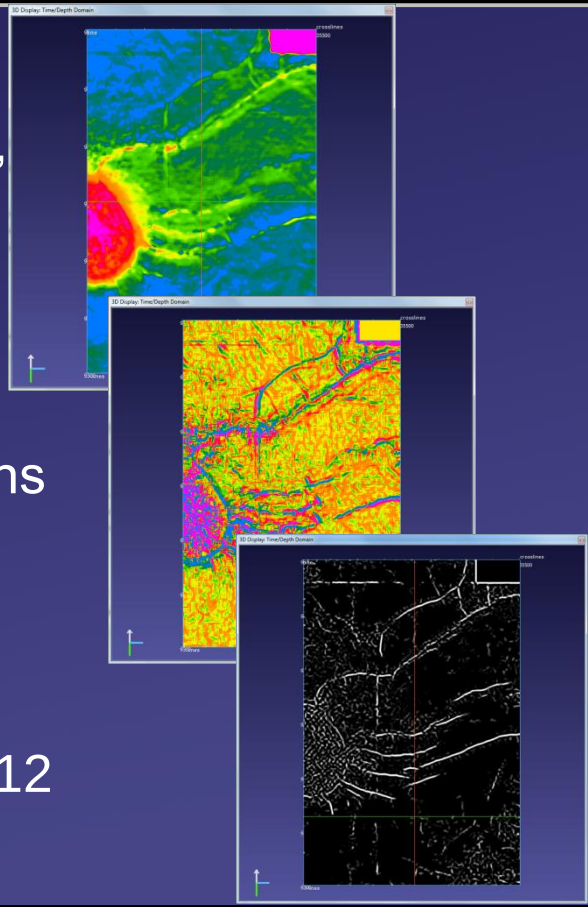
GPU Acceleration for Interpretation

- NDRange optimizations
  - How much work is sent to the GPU at a time



GPU Acceleration for Interpretation

- Conclusions
  - Seismic attribute calculations are a “no-brainer” for GPU acceleration
  - Performance gains are well worth the effort
  - GPU Acceleration brings server-class performance to seismic interpreters’ workstations
  - Small investment in hardware == big improvement in interpreter productivity (and happiness!?)
  - Available commercially from TerraSpark Q4 2012



GPU Acceleration for Seismic Interpretation Algorithms

Jon Marbach, Ph.D.
Pate Motter
TerraSpark Geosciences

Special Thanks to our research sponsors:
BHP, BP, Chevron, ConocoPhillips, Repsol, and Stone Energy