Georgia Tech Research Institute
Problem. Solved.

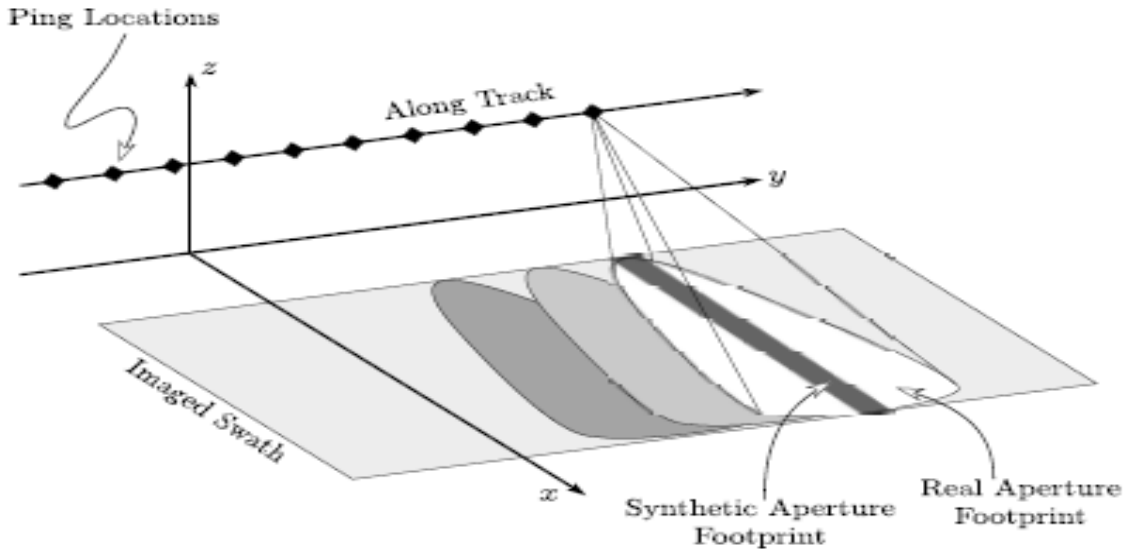
Using GPUs to Accelerate Synthetic Aperture Sonar Imaging via Backpropagation

GPU Technology Conference 2012
May 15, 2012

Thomas M. Benson, Daniel P. Campbell, Daniel A. Cook
thomas.benson@gtri.gatech.edu

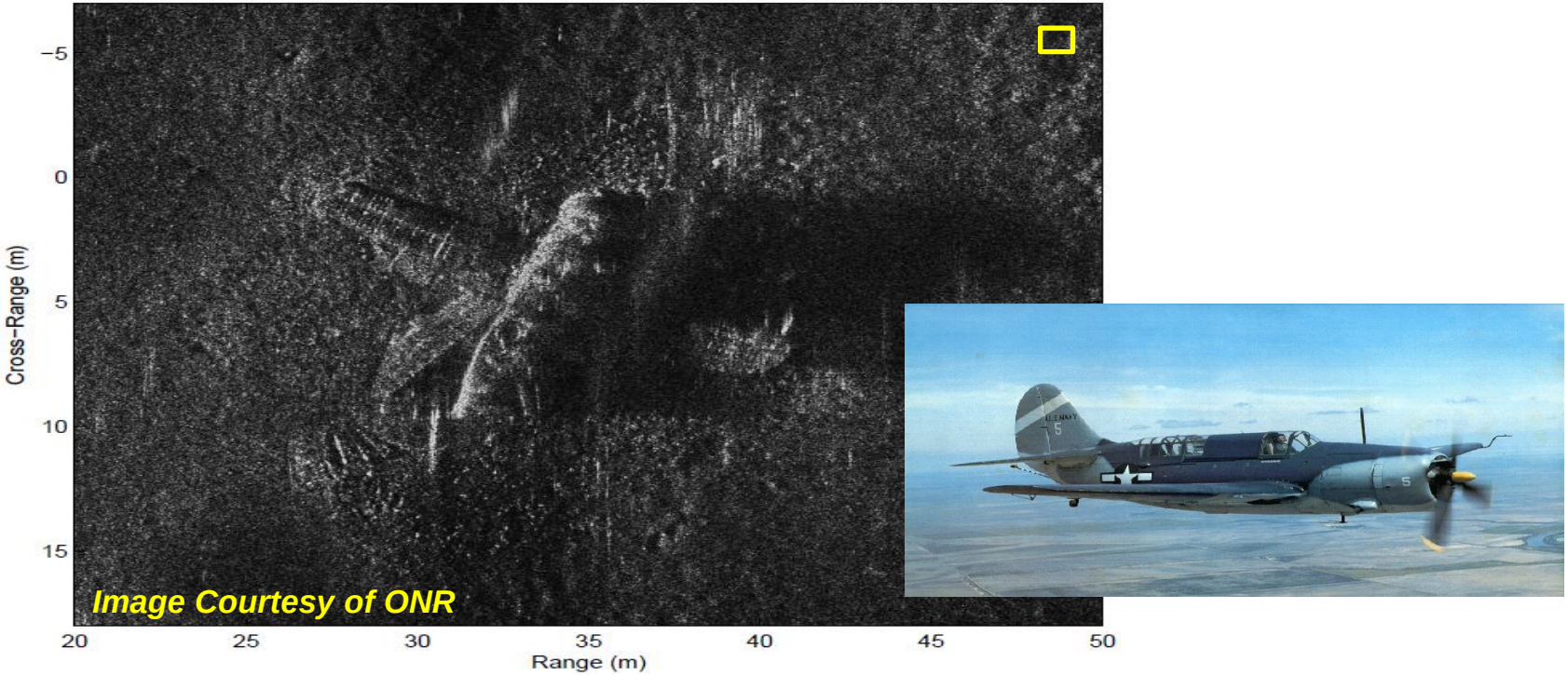
Synthetic Aperture Sonar (SAS)

- Active sonar: Emits acoustic pings/pulses, listens for echo returns
- Forms an image by combining multiple pings



SAS Sample Image

Curtiss SB2C (2.5 cm res.)



Backpropagation / Backprojection

$$f(\boldsymbol{x}) = \int_{\ell} \overline{s}_{\text{ideal}}(\tau_p, t) s(\tau_p, t)\ d\ell$$
$$= \int \delta(\tau_p, t - 2R/c) \overline{s}_{\text{ideal}}(\tau_p, t) s(\tau_p, t)\ d\tau_p dt$$

---

for all pings p and pixels v **do**
 Compute range R_{pv} from platform to pixel
 if pixel is within main beam **then**
  Compute platform motion during ping
  for all receiver channels r **do**
   Apply platform motion to update r location
   Compute range R_{pvr} from pixel to receiver channel
   Accumulate $X_{\text{interp}} \cdot \exp\left(-jk_u(R_{pv} + R_{pvr})\right)$ into v
  end for
 end if
end for

Advantages of Backpropagation

- **Fast, FFT-based image formation methods are limited:**
  - Constrained collection paths (e.g., linear or circular)
  - Image formed on planar surfaces
  - Both constraints problematic at sea
- **Backpropagation general enough to avoid such constraints, but at significantly higher computational cost**

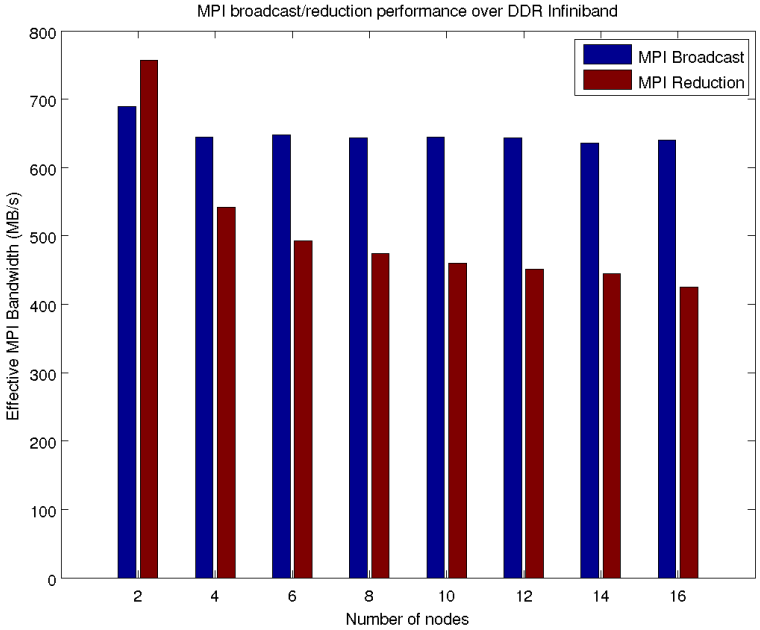
Synthetic Aperture Sonar/Radar

- **Typical SAS range/resolution: 100m/3cm**
- **Typical SAR range/resolution: 10km/0.3m**
- **Light travels 200,000 times faster than sound in seawater**
  - SAR can assume no intra-pulse motion; SAS cannot
  - SAS involves more computation per pulse, but collects data much more slowly

MPI Distribution

Image Domain Decomposition

- Broadcast raw data
- Form tile of image per node
- Gather tiles



Data Domain Decomposition

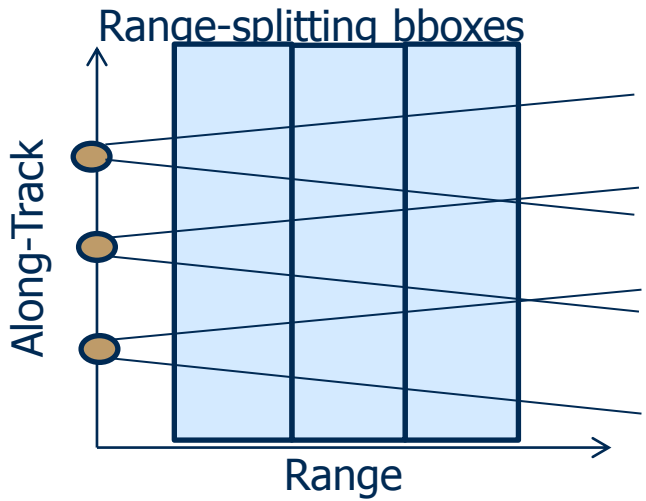
- Send data subsets
- Form fully-sized image with partial data per node
- Reduction on all images

- Broadcasts more efficient than reductions
- Broadcasts precede BP and can thus be hidden

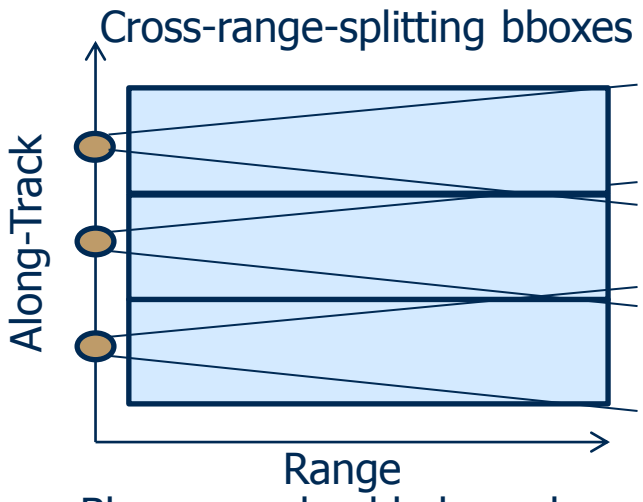
→ **We employ image domain decomposition**

Image Domain Decomposition – Load Balancing

- Per-GPU image bounding boxes (bboxes)
- Transmit beam is directed → spatially varying ping contributions per pixel

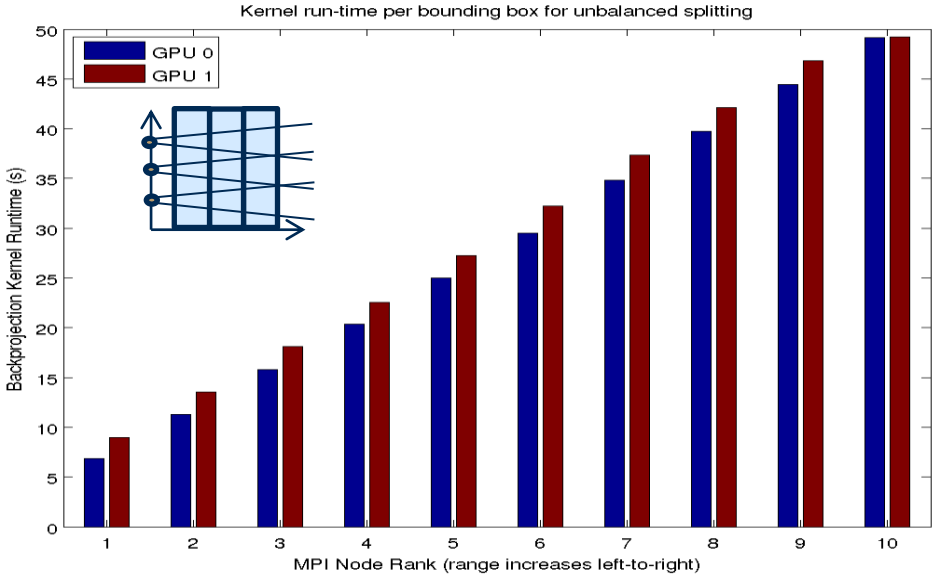


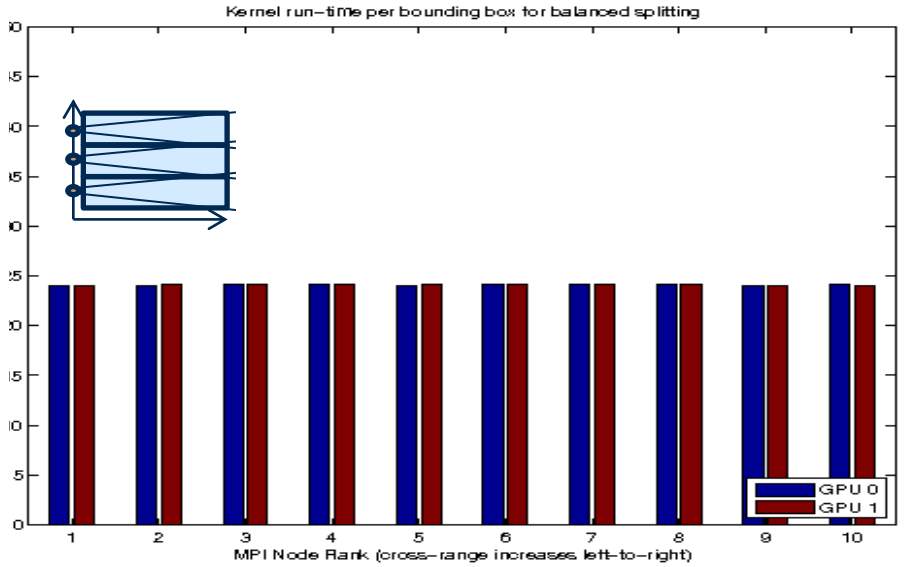
Bounding boxes at far range have higher workload



Bboxes are load balanced

Image Domain Decomposition – Load Balancing

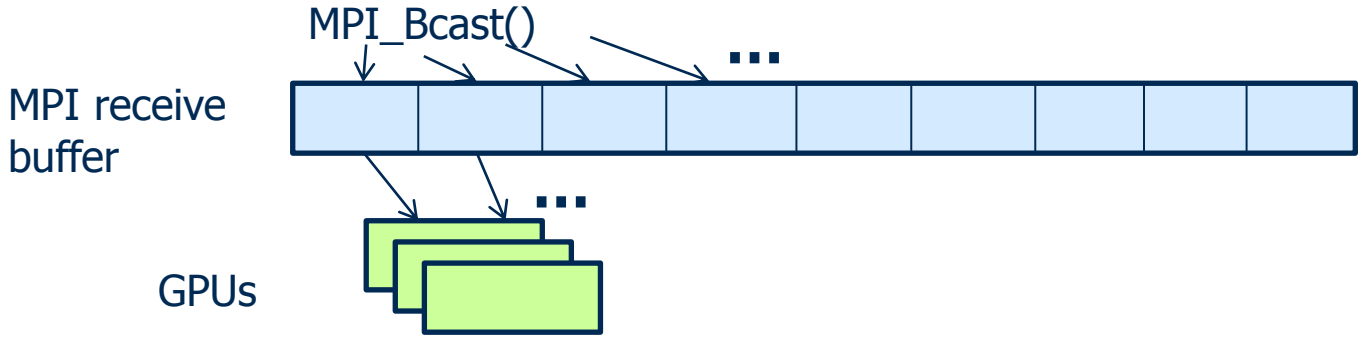




Bounding boxes splitting in cross-range effectively load balance up to at least 20 GPUs.

MPI Distribution

- Data starts on a single node
- Distributed to all nodes via MPI_Bcast()
  - Separate thread broadcasts 256MB at a time → provides asynchronous behavior, allows communication hiding
- Root node calls MPI_Gatherv() as soon as MPI_Bcast() completes → async. image data transfers



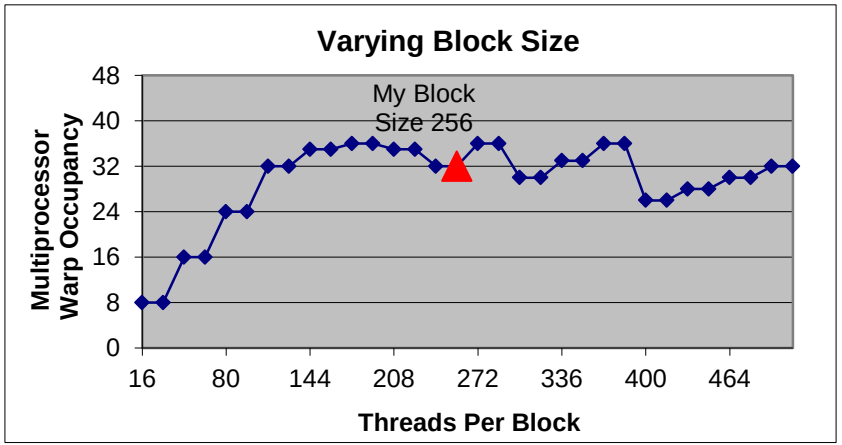
Handling Large Datasets

- **Input data set (simulated)**
  - 622 pings, 66 receive channels, 32K range bins
  - Complex input (8 bytes/elem) → ~ 10.5 GB input data
- **Output image is 32K x 32K**
  - Complex pixels → 8 GB total output (distributed)

- **We store the full input buffer to avoid blocking MPI**
  - This buffer is too large to pin/lock
- **Careful to minimize PCI-e transfers**
  - e.g., no image transfers between pulse chunks

Thread Block Mapping, Occupancy, Etc.

- One thread per pixel: each thread iterates through all available pings
- Thread block size is 16x16: 256 threads/block
- 28 registers/thread → 67% occupancy
- Shared memory usage < 1K/block (receiver positions)



Although occupancy can be increased from 67% to 75%, that offered no advantage in our testing.

CUDA Occupancy Calculator

Single-Node Optimization Considerations

- **Textures for clamped, interpolated sampling**
  - Unclear whether interpolation accuracy is sufficient
- **Intrinsics for transcendentals (__sin(), __cos())**
  - Used for rotations, matched filter
- **Careful separation of channel-invariant calculations**
  - This offered a 10x+ improvement over our previous version
- **Shared memory for (shared) receiver positions**
- **Memory caching policy dependent on texture usage**
  - Without textures, L2-only caching via –Xptxas –dlcm=cg
    - Reduces over-fetch of phase history data (4.8% speedup)
  - With textures, L1/L2 caching via –Xptxas –dlcm=ca (default)
    - 128B fetch / L1 useful for Tx positions (1.2% speedup)

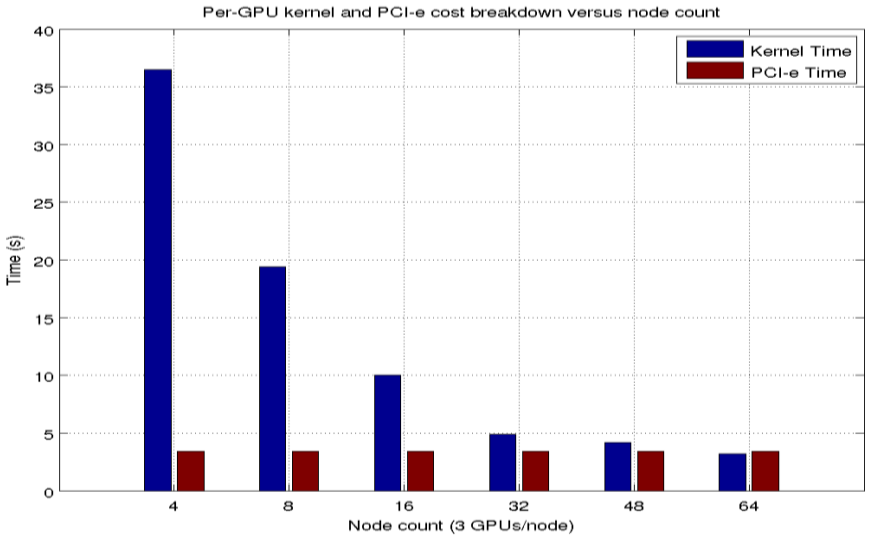
Keeneland: Deployment Platform

- **Keeneland Initial Deployment**
- **120 Nodes**
  - **3 Tesla M2070**
    - **Currently being upgraded to M2090**
  - **QDR Infiniband**
  - **2x 6-core Westmere**
- http://keeneland.gatech.edu/

This research used resources of the Keeneland Computing Facility at the Georgia Institute of Technology, which is supported by the National Science Foundation under Contract OCI-0910735.

Initial Results

- Initial runs on Keeneland to identify scaling bottlenecks
- Ignoring all other costs and comparing only kernel and PCI-e time reveals that PCI-e is eventually a bottleneck



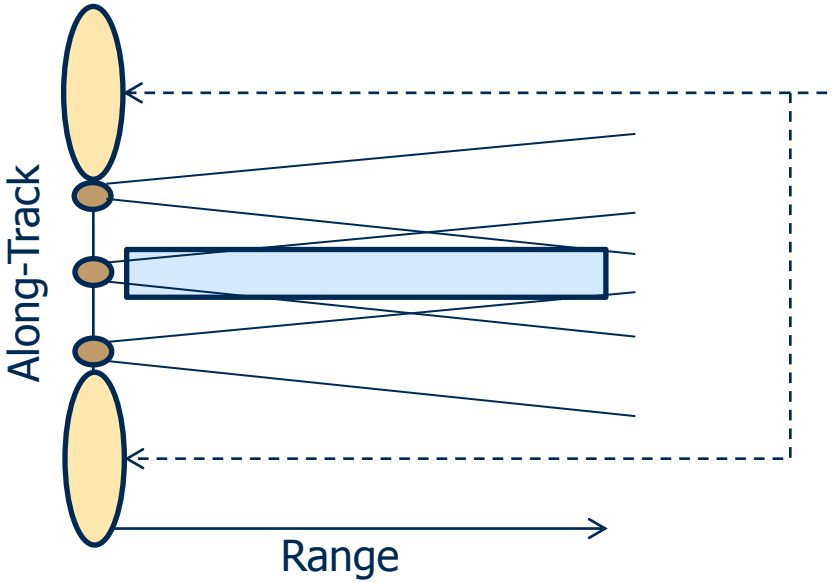
PCI-e time is constant (all data is transferred to the GPUs) and at 64 nodes exceeds the kernel time.

Fortunately, we can cull much of the data on the host and never transfer it to the GPU.

We later address bottlenecks imposed by MPI.

Host-Based Culling

- Many pings will not contribute to a given bounding box
  - Smaller bounding boxes → fewer contributing pings

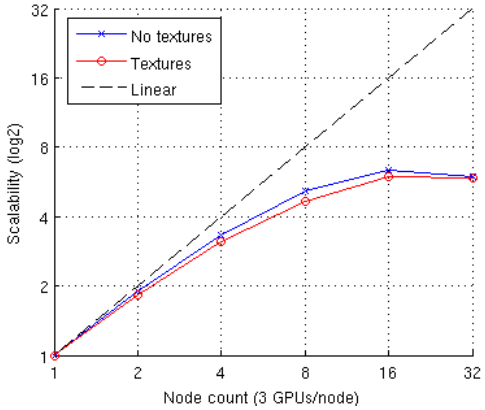


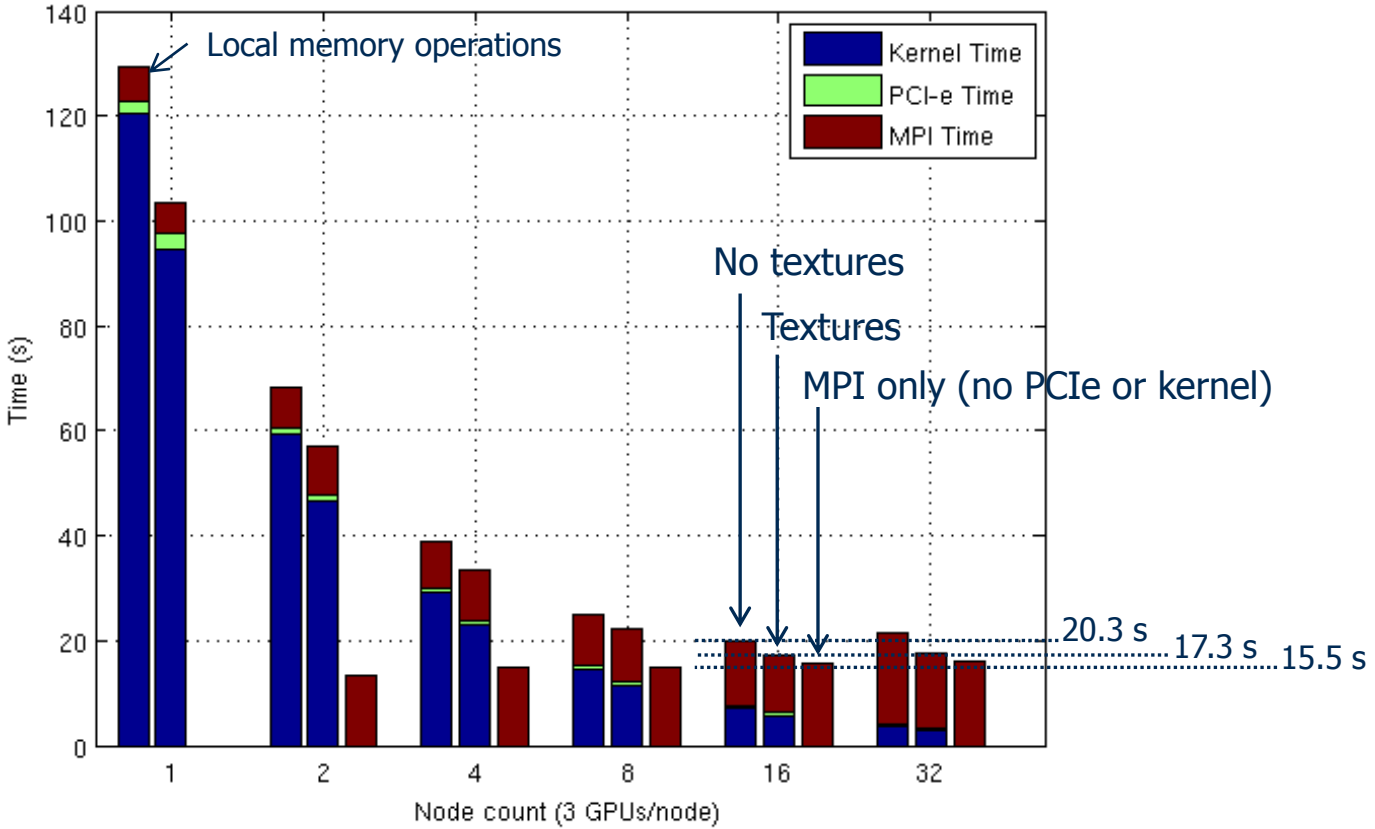
These regions can be culled on the host to avoid PCI-e transfers.

This is a per-pulse calculation that accounts for platform rotations (the array normal can vary by position, so all pulses need to be checked).

This check reduces PCI-e bandwidth as the GPU count increases (and bboxes become smaller), so helps scalability considerably.

Run-Time Results With Host-Based Culling





Summary / Conclusions

- MPI broadcast costs can be effectively hidden until total communication costs exceed computation costs
- Communication costs exceed computation at 16 nodes
- Image formation is faster than real-time (i.e., data collection), but would be processed in batch mode after a large collection
  - We can apply weak scaling to process multiple data sets concurrently
- Kernel optimizations and host culling shifted the node/GPU count with minimum cost from 48/144 to 16/48.
  - Host culling cuts PCI-e costs by over 5x at 16 nodes

Future Work

- Extend to larger (streaming) data sets
  - Circular MPI buffer
- Apply hybrid approach also utilizing CPUs for backprojection
  - Addresses the MPI bottleneck
- Investigate sending only required data to each node to reduce MPI communication time
  - At 16 nodes, MPI_Bcast() requires 8.4 seconds and MPI_Gatherv() requires 6.6 seconds, so bcast is quite efficient

Acknowledgements

This work supported in part by DARPA under contracts HR0011-10-C-0145. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Thank you!

Questions / Comments?