SDVis and In-Situ Visualization on TACC’s Stampede-KNL

Paul A. Navrátil, Ph.D.
Manager – Scalable Visualization Technologies
pnav@tacc.utexas.edu
High-Fidelity Visualization Natively on Xeon and Xeon Phi
OUTLINE

- Stampede Architecture
  - Stampede – Sandy Bridge
  - Stampede - KNL
  - Stampede 2 – KNL + Skylake

- Software-Defined Visualization Stack
  - VNC
  - OpenSWR
  - OSPRay

- Path to In-Situ
  - ParaView Catalyst
  - VisIt Libsim
  - Direct renderer integration
STAMPEDE ARCHITECTURE
Stampede Sandy Bridge

- Mellanox FDR Interconnect
- 6400 compute nodes, each with:
  - 2x Intel Xeon E5-2680 "Sandy Bridge"
  - 1x Intel Xeon Phi SE10P
  - 32 GB RAM / 8 GB Phi RAM
- 16 Large Memory nodes, each with:
  - 4x Intel Xeon E5-4650 "Sandy Bridge"
  - 2x NVIDIA Quadro 2000 "Fermi"
  - 1 TB RAM
- 128 GPU nodes, each with:
  - 2x Intel Xeon E5-2680
  - 1x Intel Xeon Phi SE10P
  - 1x NVIDIA Tesla K20 "Kepler"
  - 32 GB RAM
STAMPEDE KNL

- Deployed in 2016 as planned upgrade to Stampede
- First KNL-based system in Top500
- Intel OmniPath interconnect
- 508 nodes, each with:
  - 1x Intel Xeon Phi 7250
  - 96 GB RAM + 16 GB MCDRAM

Notes:
- Shared $WORK and $SCRATCH Separate $HOME directories
- Separate Login Node login-knl1.stampede.tacc.utexas.edu
- Login is Intel Xeon E5-2695 “Haswell”
- Compile on compute node or use “–xMIC-AVX512” on login
- “normal” and “development” queues are Cache-Quadrant
- Other MCDRAM configs available by queue name
STAMPEDE 2 (COMING 2017)

- ~18 PF Dell Intel Xeon + Intel Xeon Phi system
- Combine KNL + Skylake + OmniPath + 3D XPoint
- Phase 1: Spring 2017
  - Stampede KNL + 4200 new KNL nodes + new filesystem
  - 60% of Stampede Sandy Bridge to remain operational during this phase
- Phase 2: Fall 2017
  - 1736 Intel Skylake nodes
- Phase 3: Spring 2018
  - Add 3D XPoint memory to subset of nodes
KEY ARCHITECTURAL TAKE-AWAY

- Current and near-future cyberinfrastructure will use processors with many cores
- Each core contains wide vector units: use them for max utilization (e.g., AVX512)

- Fortunately the Software-Defined Visualization stack is optimized for such processors!
- Use your preferred rendering method independent of the underlying hardware
  - Performant rasterization
  - Performant ray tracing
  - Visualization and analysis on the simulation machine
### SOFTWARE-DEFINED VISUALIZATION – WHY?

<table>
<thead>
<tr>
<th>File Size</th>
<th>100 Gbps</th>
<th>10 Gbps</th>
<th>1 Gbps</th>
<th>300 Mbps</th>
<th>54 Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB</td>
<td>&lt; 1 sec</td>
<td>1 sec</td>
<td>10 sec</td>
<td>35 sec</td>
<td>2.5 min</td>
</tr>
<tr>
<td>1 TB</td>
<td>~100 sec</td>
<td>~17 min</td>
<td>~3 hours</td>
<td>~10 hours</td>
<td>~43 hours</td>
</tr>
<tr>
<td>1 PB</td>
<td>~1 day</td>
<td>~12 days</td>
<td>~121 days</td>
<td>&gt;1 year</td>
<td>~5 years</td>
</tr>
</tbody>
</table>

Increasingly Difficult to Move Data from Simulation Machine
SOFTWARE-DEFINED VISUALIZATION
SOFTWARE-DEFINED VISUALIZATION – WHY?
SOFTWARE-DEFINED VISUALIZATION – WHY?
SOFTWARE-DEFINED VISUALIZATION – WHY?
SOFTWARE-DEFINED VISUALIZATION – WHY?
SOFTWARE-DEFINED VISUALIZATION STACK

- OpenSWR Software Rasterizer
  - openswr.org
  - Performant rasterization for Xeon and Xeon Phi
  - Thread-parallel vector processing (previous parallel Mesa3D only has threaded fragments)
  - Support for wide vector instruction sets, particularly AVX2 (and soon AVX512)
  - Integrated into Mesa3D 12.0 as optional driver (mesa3d.org)

- Best Uses
  - Lines
  - Graphs
  - User Interfaces
SOFTWARE-DEFINED VISUALIZATION STACK

- OSPRay Ray Tracer
  - ospray.org
  - Performant ray tracing for Xeon and Xeon Phi incorporating Embree kernels
  - Thread- and wide-vector parallel using Intel ISPC (including AVX512 support)
  - Parallel rendering support via distributed framebuffer

- Best Uses
  - Photorealistic rendering
  - Realistic lighting
  - Realistic material effects
  - Large geometry
  - Implicit geometry (e.g., molecular "ball and stick" models)
SOFTWARE-DEFINED VISUALIZATION STACK

- GraviT Scheduling Framework
  - tacc.github.io/GraviT/
  - Large-scale, data-distributed ray tracing (uses OSPRay for rendering engine target)
  - Parallel rendering support via distributed ray scheduling

- Best Uses
  - Large distributed data
  - Data outside of renderer control
  - Incoherent ray-intensive sampling (e.g., global illumination approximations)
OSPRay Test Suite – Sample Images
OSPRay Test Suite – MCDRAM Performance Results
ParaView Test Suite – ManySpheres
ParaView manyspheres Benchmark @ 400s

Likely VNC limited
ParaView manyspheres Benchmark @ 1024s

- Likely VNC limited
Definitely VNC limited!
FIU Core Sample – Sample Image
ParaView fiu Benchmark @ 400s

Frames per Second (log)

- **Likely VNC limited**

- **streamlines**
- **streamlines + contour**

- mesa
- gpu
- swr
- ospray KNL
- ospray Xeon
ParaView fiu Benchmark @ 1024s

Frames per Second (log)

- mesa
- gpu
- swr
- ospray KNL
- ospray Xeon

Likely VNC limited
Likely hitting VNC desktop limits!
Path to In-Situ Visualization
**Why In-Situ Visualization?**

- Processors (like KNL) enabling larger, more detailed simulations
- File system technologies not scaling at same rate (if at all….)
- Touching disk is expensive:
  - During simulation: time checkpointing is (often) not time computing
  - During analysis: loading the data is (often) the overwhelming majority of runtime
- In-situ capabilities overcome this data bottleneck
  - Render directly from resident simulation data
  - Tightly coupled vis opens doors for online analysis, computational steering, etc
CURRENT IN-SITU OPTIONS

- Simulation developer
  - Implement visualization API (ParaView Catalyst, VisIt libsim, VTK)
  - Implement data framework (ADIOS, etc)
  - Implement direct rendering calls (OSPRay API, etc)
- Simulation user
  - Hope the developers do one of the above 😊
  - Do one of the above yourself 😎
  - Hope technology keeps post-hoc analysis viable (3D XPoint NVRAM might help)
**In-Situ Visualization APIs**

- ParaView Catalyst (and Cinema)  
  (www.paraview.org/in-situ/)
- VisIt Libsim  
  (www.visitusers.org/index.php?title=Libsim_Batch)
- Direct VTK integration  
  (www.vtk.org)
- Visualization ops already implemented
- Need coordination b/t teams to ensure simulation and vis performance

Image courtesy of Kitware Inc.
IN-SITU-COMPATIBLE DATA FRAMEWORKS

- Damaris – https://hal.inria.fr/hal-00859603/en
- SCIRun – http://www.sci.utah.edu/cibc-software/scirun.html

- (Possibly) more invasive implementation effort
- (Possibly) broader benefits beyond visualization (framework now controls data)
- Requires engagement from simulation team to ensure performance and accuracy
**IN-SITU DIRECT RENDERING**

- Render directly within simulation using API (e.g., OSPRay, OpenGL, etc)
- Must implement visualization operations within simulation code
  - Lightest weight, lowest overhead
  - Requires visualization algorithm knowledge for efficient implementation
- Locks in particular rendering and visualization modes
IN-SITU FUTURE?

Useful perspectives at ISAV – http://conferences.computer.org/isav/2016/
TACC/Kitware IPCC – Unimpeded In Situ Visualization on Intel Xeon and Intel Xeon Phi

- Optimize ParaView Catalyst for current and near-future Intel architectures
  - KNL, Skylake, Omnipath, 3D XPoint NVRAM
  - Use Stampede-KNL as testbed to target TACC Stampede 2, NERSC Cori, LANL Trinity

- Focus on data and rendering paths for OpenSWR and OSPRay
  - Parallelize VTK data processing filters
  - Catalyst integration with simulation
  - Targeted algorithm improvements
    - Increase processor and memory utilization
THANK YOU!

pnav@tacc.utexas.edu