Towards Direct Visualization on CPU and Xeon Phi

Aaron Knoll
SCI Institute, University of Utah

Intel HPC DevCon 2016

In collaboration with:
Ingo Wald, Jim Jeffers — Intel Corporation
Joe Insley, Silvio Rizzi, Mike Papka — Argonne National Laboratory

Utah IPCC/Intel Vis Center

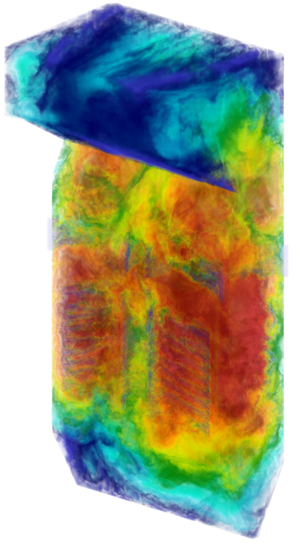
- **University of Utah, Salt Lake City**
  The “birthplace of computer graphics” —
  Evans and Sutherland, Catmull, Kajiya, Blinn, Phong…

- **Scientific Computing and Imaging Institute:**
  World leader in scientific visualization — *“graphics for science” and more.*

- **Intel centers at SCI: 6 faculty, 9 students**

  - **Intel Vis Center**
    PIs: Ingo Wald (Intel), Chris Johnson, Chuck Hansen
    - Large-scale vis and HPC technology on CPU/Phi hardware — especially **OSPRay**.

  - **IPCC for “Applied Visualization, Computing and Analysis”:**
    PIs: Aaron Knoll, Valerio Pascucci, Martin Berzins
    - Applying OSPRay to visualization and HPC production in practice (i.e., **Uintah**)
    - Visualization analysis research: IO, topology, multifield/multidimensional
    - Staging Intel resources for both the Vis Center and IPCC.



- External partners:

  - **Uintah:** DOE PSAAP II efficient coal boiler simulation (Phil Smith, Utah ICSE) and DOE INCITE computational awards (Martin Berzins)
    350M hours for 2016 — the largest single open-science computational effort in the nation.

  - **Nanoview collaboration with Argonne National Laboratory:**
    Support materials science users at Argonne National Laboratory, US Dept of Energy (DOE)
    Mike Papka (director of ALCF), Joe Insley (ALCF vis lead), Silvio Rizzi (ALCF vis staff)

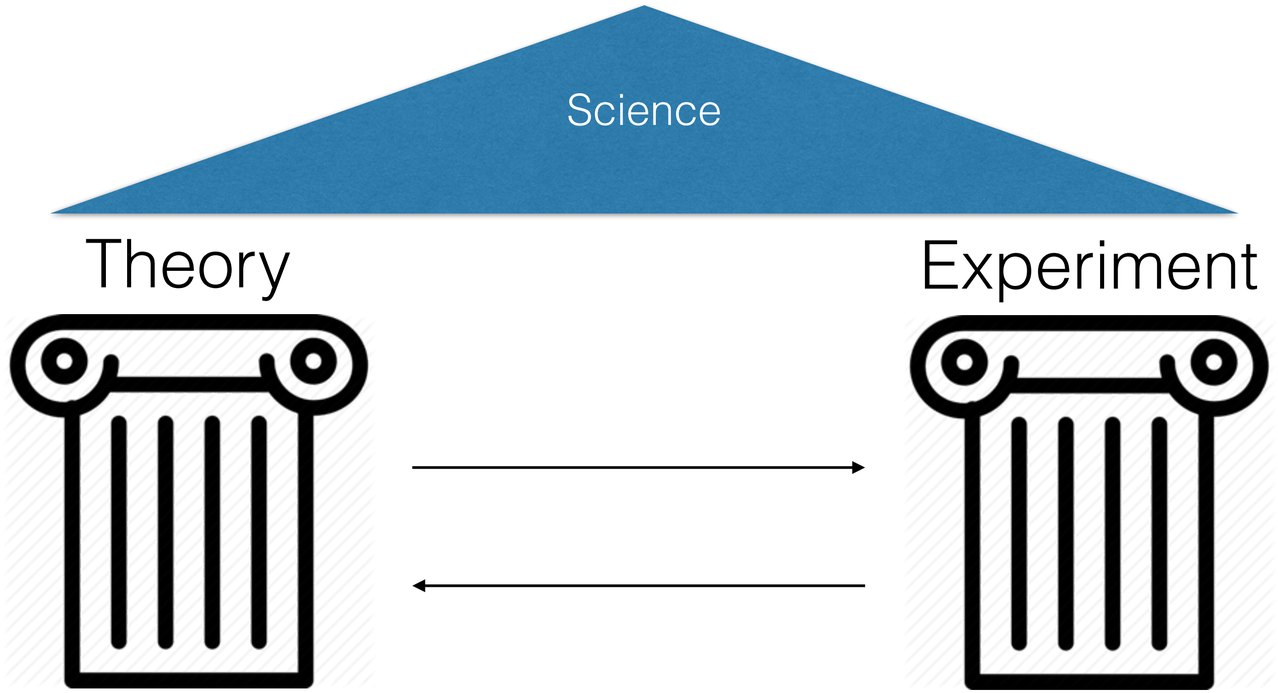
SCI
www.sci.utah.edu

Roadmap

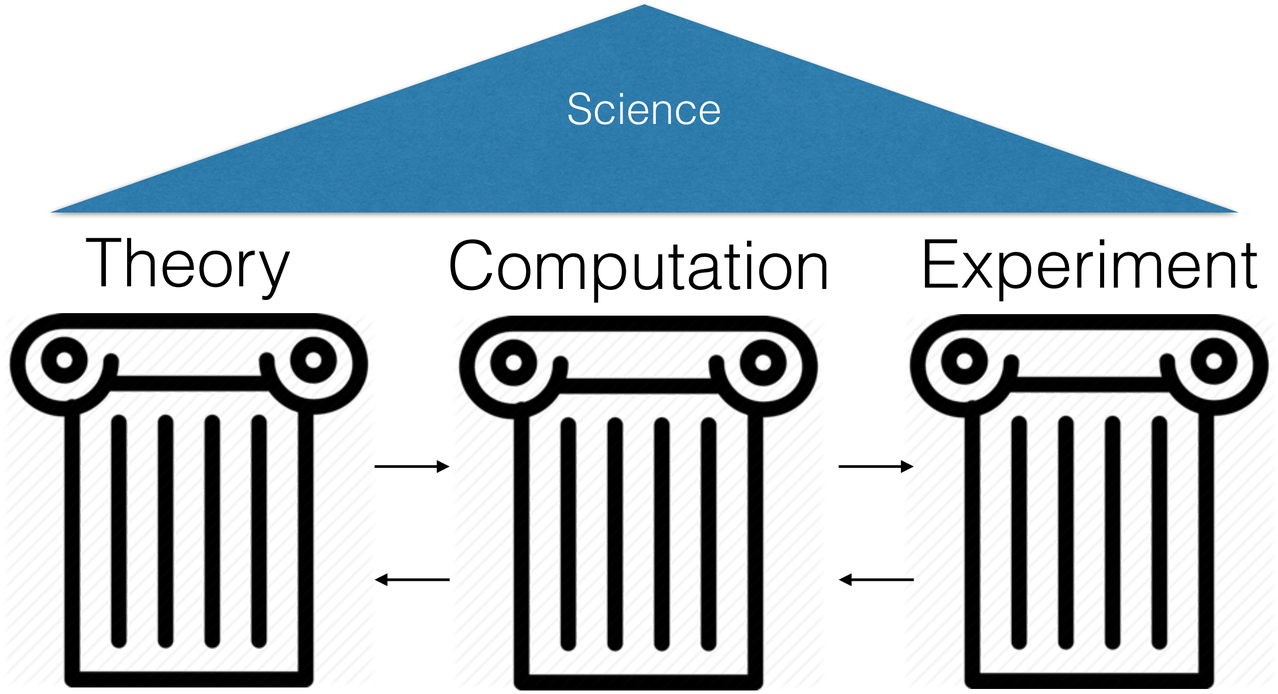
- Part I: The scientific visualization landscape today
  - Why vis?
  - Direct vs Indirect visualization
  - GPU direct visualization: Nanovol.
  - Where Nanovol failed…
- Part II: CPU-based Visualization
  - Again, why?
  - OSPRay
- Part III: OSPRay integration and related work
- Part IV: OSPRay and other CPU vis research
- Summary thoughts…

Part I: The scientific visualization landscape today

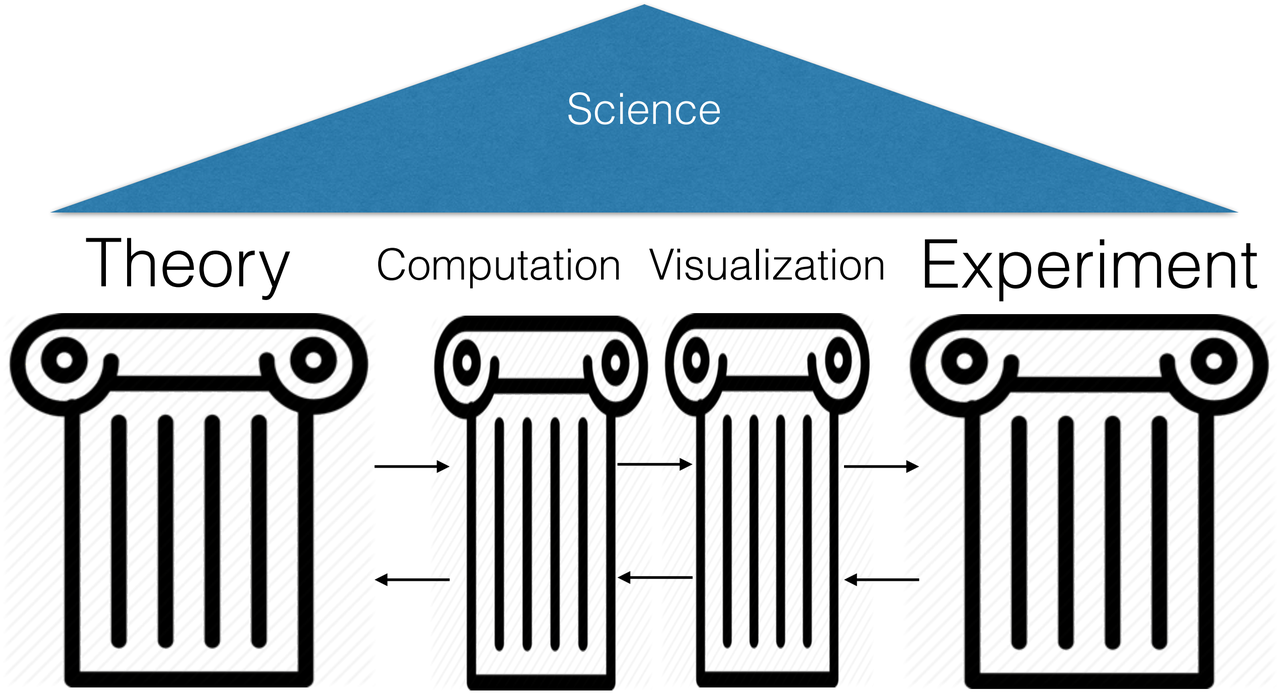
Pillars of the scientific method



Pillars of the scientific method

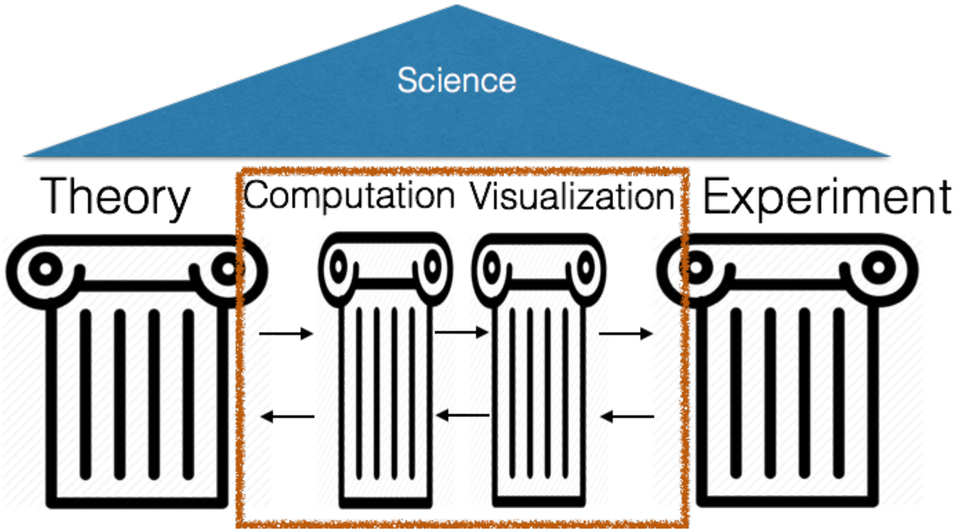


Pillars of the scientific method



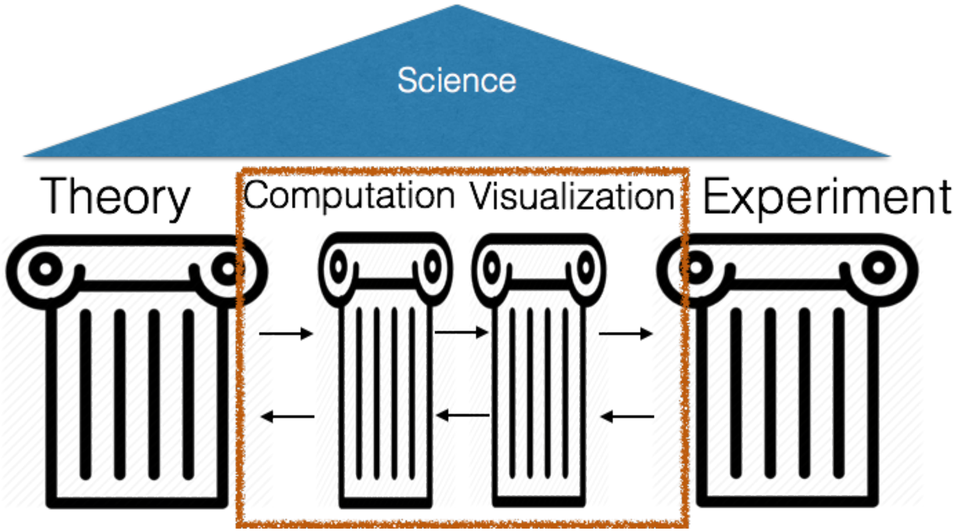
Why vis?

- If computing is the third pillar, visualization is the **fourth pillar of the scientific method.**
- Needed in:
  - Analysis
  - Debugging / Validation
  - Communication
- “Scientific vis” is often overlooked in its own community…
  - “Production tools are good enough”?
  - “Just use the same GPU graphics we use for games”?



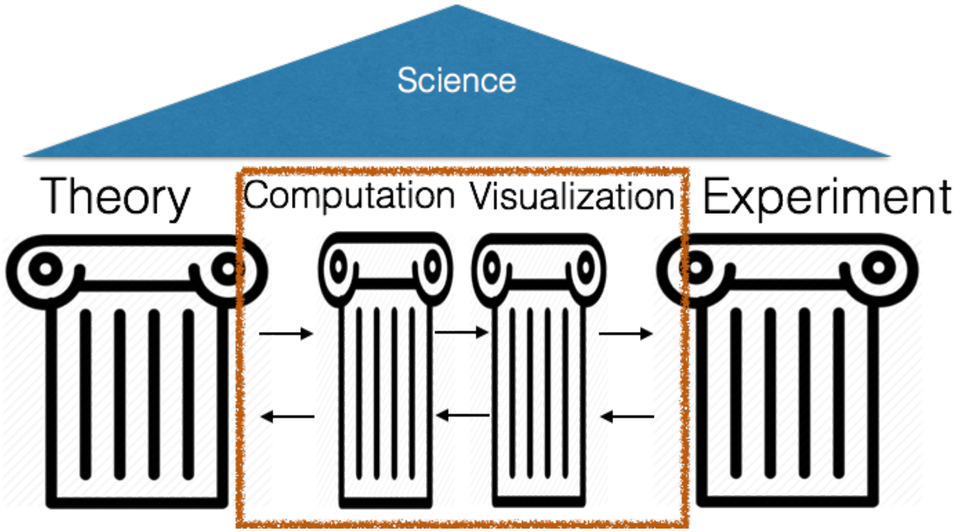
Why vis?

- If computing is the third pillar, visualization is the **fourth pillar of the scientific method.**
- Needed in:
  - Analysis
  - Debugging / Validation
  - Communication
- “Scientific vis” is often overlooked in its own community…
  - “Production tools are good enough”?
    Barely handle mid-gigascale data —
    2 orders of magnitude / 10 years
    behind simulation!
  - “Just use the same GPU graphics we use for games”?



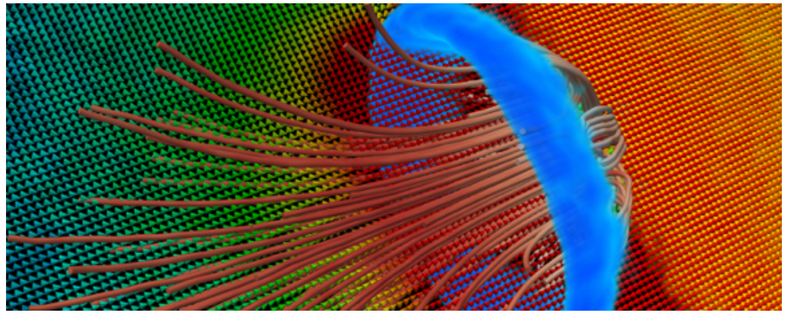
Why vis?

- If computing is the third pillar, visualization is the **fourth pillar of the scientific method.**
- Needed in:
  - Analysis
  - Debugging / Validation
  - Communication
- “Scientific vis” is often overlooked in its own community…
  - “Production tools are good enough”?
    Barely handle mid-gigascale data —
    2 orders of magnitude / 10 years
    behind simulation!
  - “Just use the same GPU graphics we use for games”?
    Rasterization is designed for millions of polygons, really fast.
    Vis should support billions—trillions of elements, a bit slower.



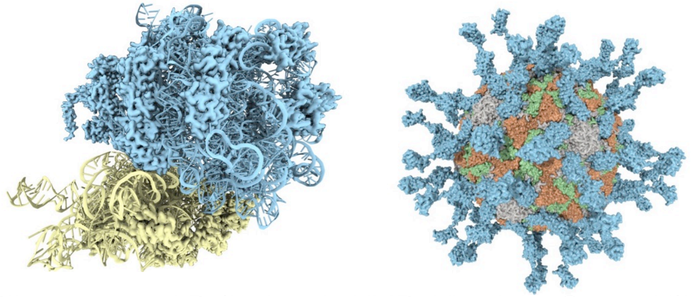
Visualization codes: general production, domain-specific, and research

- Scientific visualization
  (ParaView, VisIt, SCIRun, Ensight)



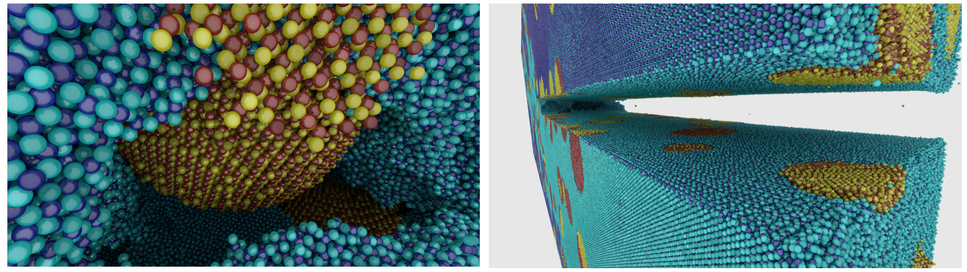
Silicon bubble MD simulation in ParaView, Ken-ichi Nomura, USC.
Vis: Joe Insley, ANL

- Molecular visualization
  (VMD, JMol, PyMol, Avogadro)



Ribosome and Poliovirus in VMD. Vis: John Stone, UIUC

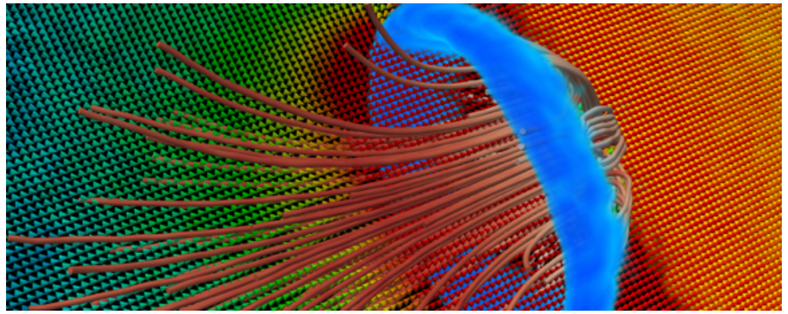
- Particle visualization
  (ospray/pkd, megamol)



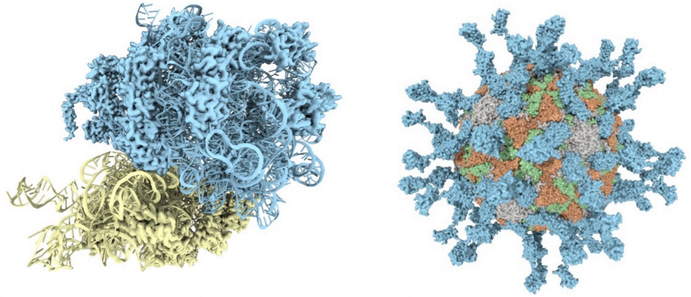
100M atom Al2O3 - SiC MD simulation in OSPRay/pkd,
Rajiv Kalia, USC. Vis: me.

Visualization codes: general production, domain-specific, and research

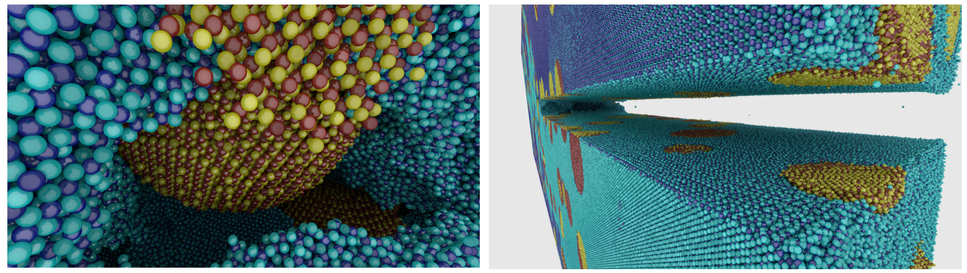
- Scientific visualization
  (ParaView, VisIt, SCIRun, Ensight)
- Molecular visualization
  (VMD, JMol, PyMol, Avogadro)
- Particle visualization
  (ospray/pkd, megamol)



Silicon bubble MD simulation in ParaView, Ken-ichi Nomura, USC.
Vis: Joe Insley, ANL



Ribosome and Poliovirus in VMD. Vis: John Stone, UIUC



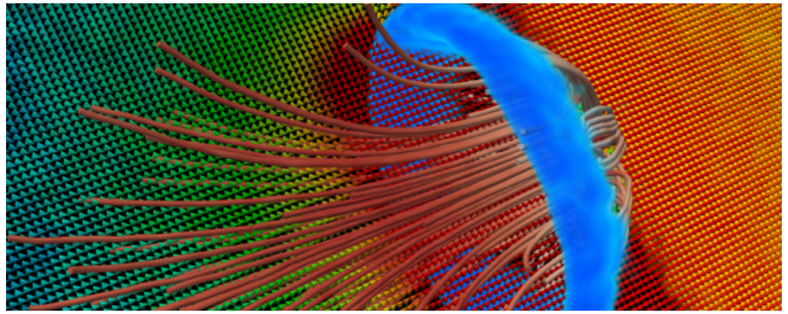
100M atom Al2O3 - SiC MD simulation in OSPRay/pkd,
Rajiv Kalia, USC. Vis: me.

general

special

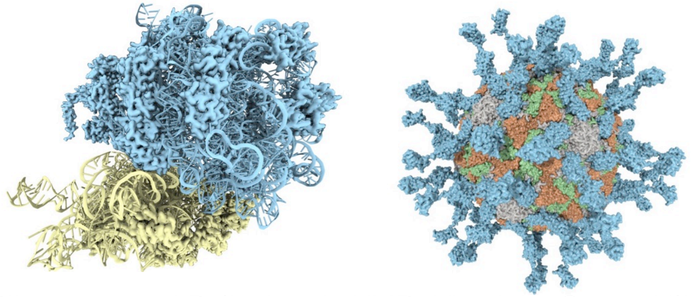
Visualization codes: general production, domain-specific, and research

- Scientific visualization
  (ParaView, VisIt, SCIRun, Ensight)



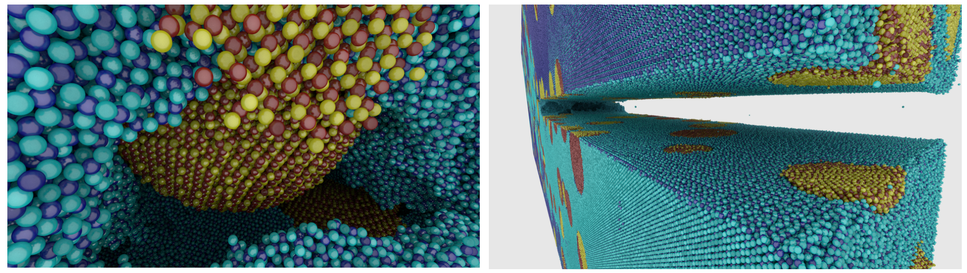
Silicon bubble MD simulation in ParaView, Ken-ichi Nomura, USC.
Vis: Joe Insley, ANL

- Molecular visualization
  (VMD, JMol, PyMol, Avogadro)



Ribosome and Poliovirus in VMD. Vis: John Stone, UIUC

- Particle visualization
  (ospray/pkd, megamol)



100M atom Al2O3 - SiC MD simulation in OSPRay/pkd,
Rajiv Kalia, USC. Vis: me.

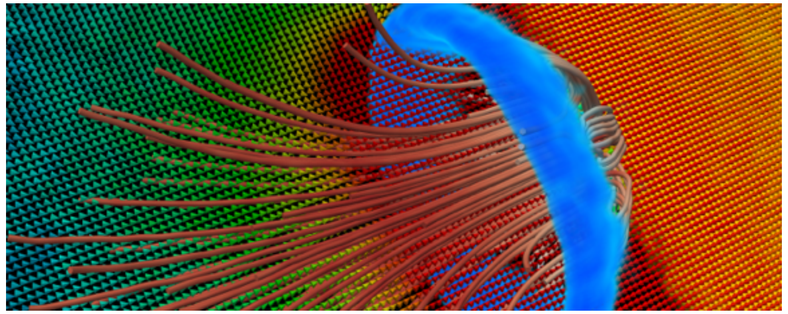
slow

fast

Visualization codes: general production, domain-specific, and research

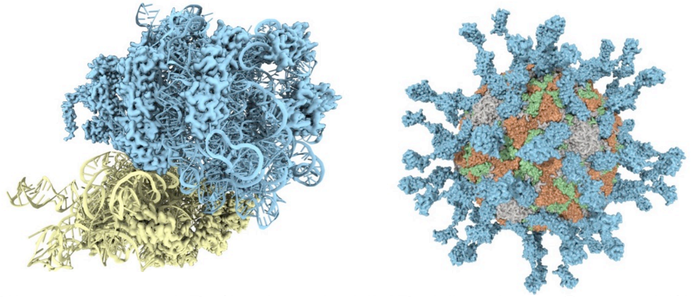
famous

- Scientific visualization
  (ParaView, VisIt, SCIRun, Ensight)



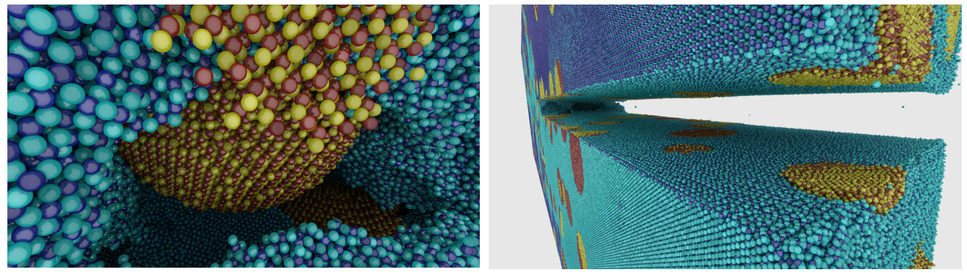
Silicon bubble MD simulation in ParaView, Ken-ichi Nomura, USC.
Vis: Joe Insley, ANL

- Molecular visualization
  (VMD, JMol, PyMol, Avogadro)



Ribosome and Poliovirus in VMD. Vis: John Stone, UIUC

- Particle visualization
  (ospray/pkd, megamol)

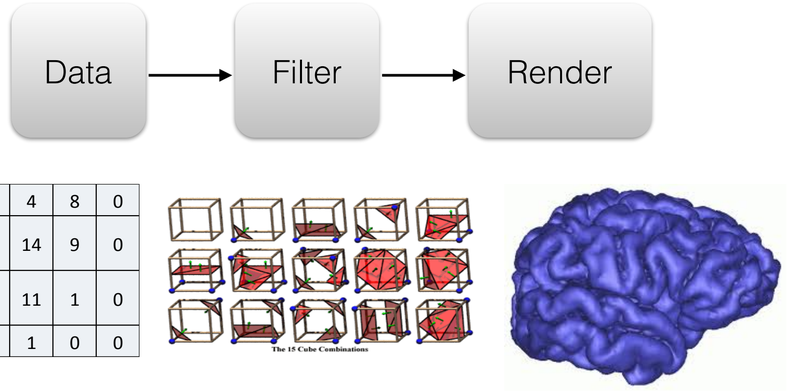


100M atom Al2O3 - SiC MD simulation in OSPRay/pkd,
Rajiv Kalia, USC. Vis: me.

obscure

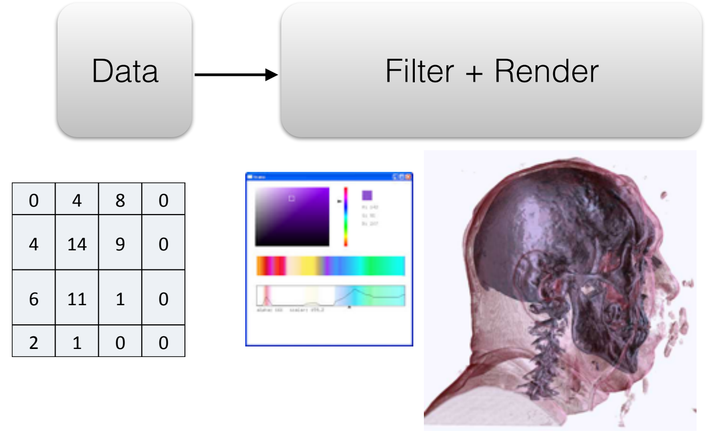
“Direct” vs “Indirect” visualization

Indirect



- ex: marching cubes, rasterization
- based on triangles
- large memory overhead
- heavy preprocess
- pipeline workflow
- good strong scaling (compute)

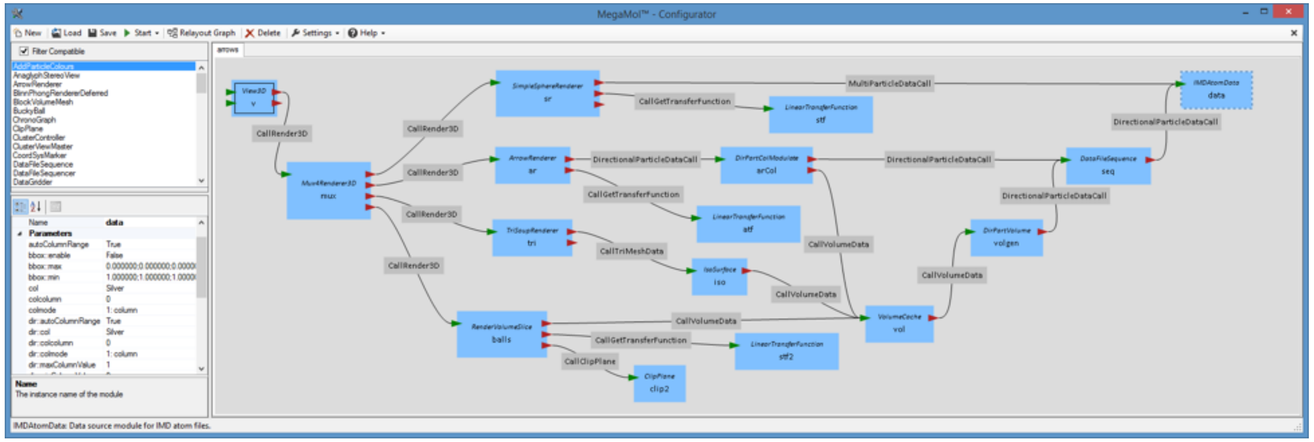
Direct

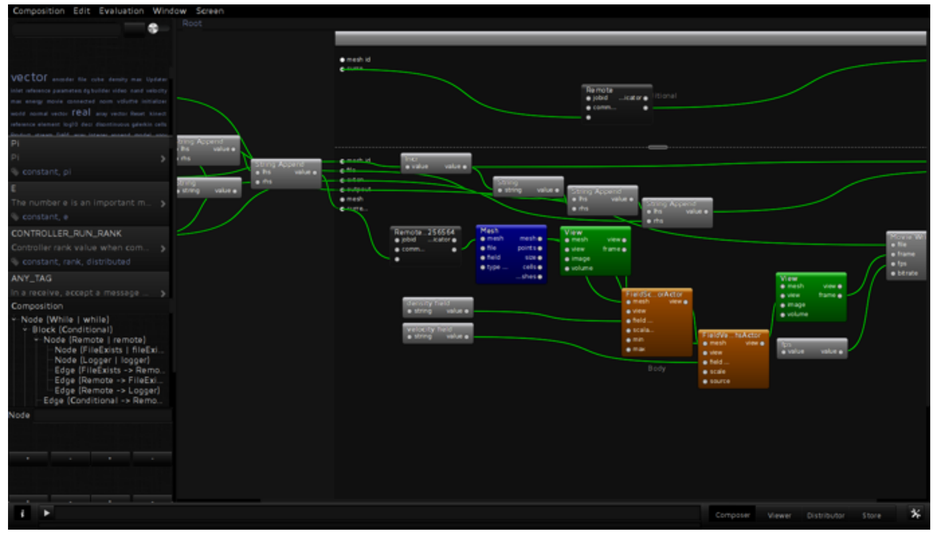


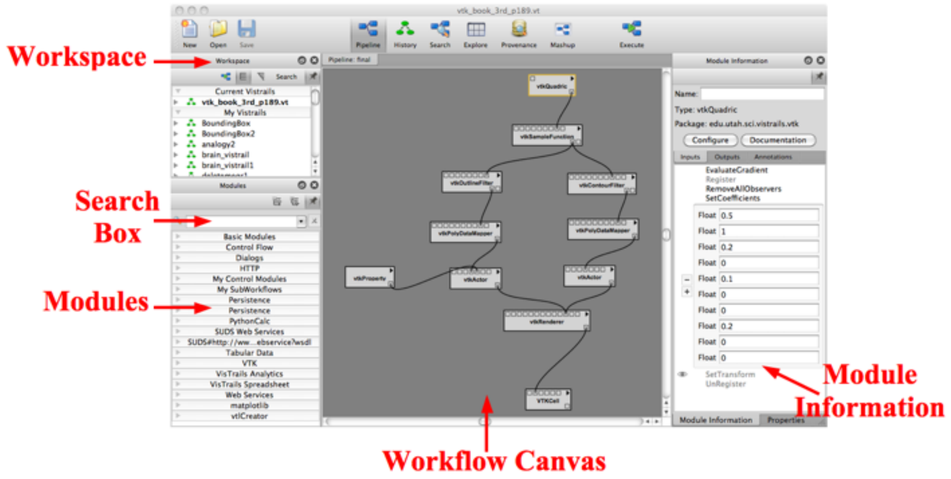
- ex: volume rendering, ray tracing
- based on volumes, glyphs
- low memory overhead
- little or no preprocess
- flat workflow
- good weak scaling (memory)

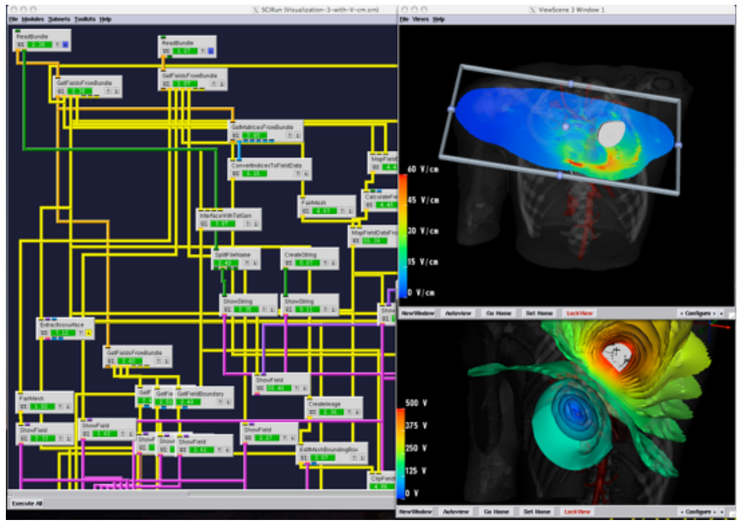
Problems with indirect visualization

1. The visualization pipeline is complex.









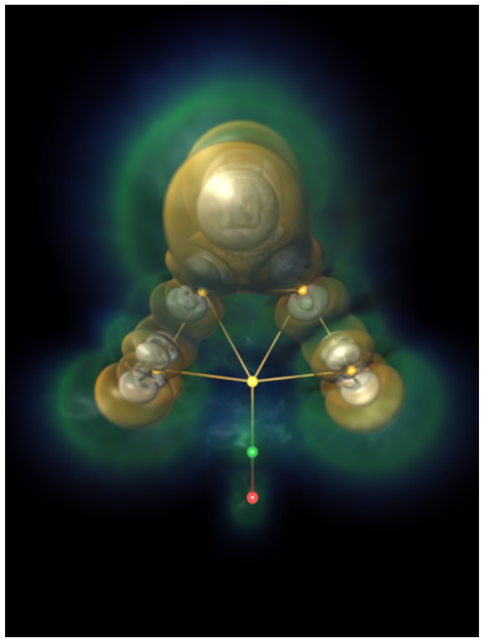
www.sci.utah.edu

2. Most visualization data are not triangles.

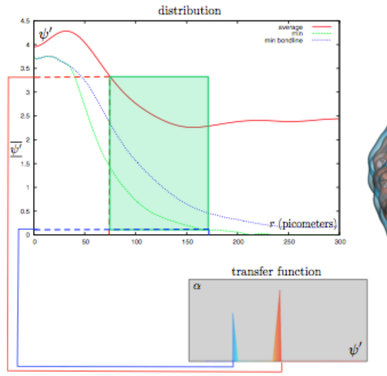
Re-envisioning scientific visualization

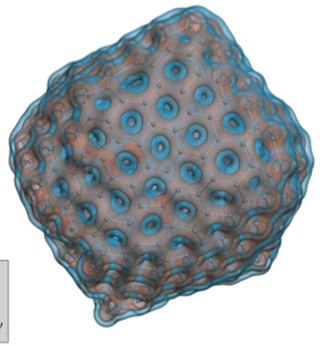
- Indirect methods and strong scaling solve IO challenges, but require resources
  - In situ and computational steering are useful, but will not fully replace storage for logistical reasons…
  - New memory/disk technologies (3DXPoint) are on the horizon
- Directions:
  - Move from **indirect** techniques to more **direct** techniques (OSPRay, vl3).
  - Leverage **large memory** for large time-varying and multifield vis problems (CPU and KNL).
  - Use appropriate **parallel data formats** to avoid distributed fileserver inefficiency (PIDX).
    - when disk == memory, writing to these formats becomes “in situ”.

Early “direct vis”: Nanovol on the GPU, 2010-2014



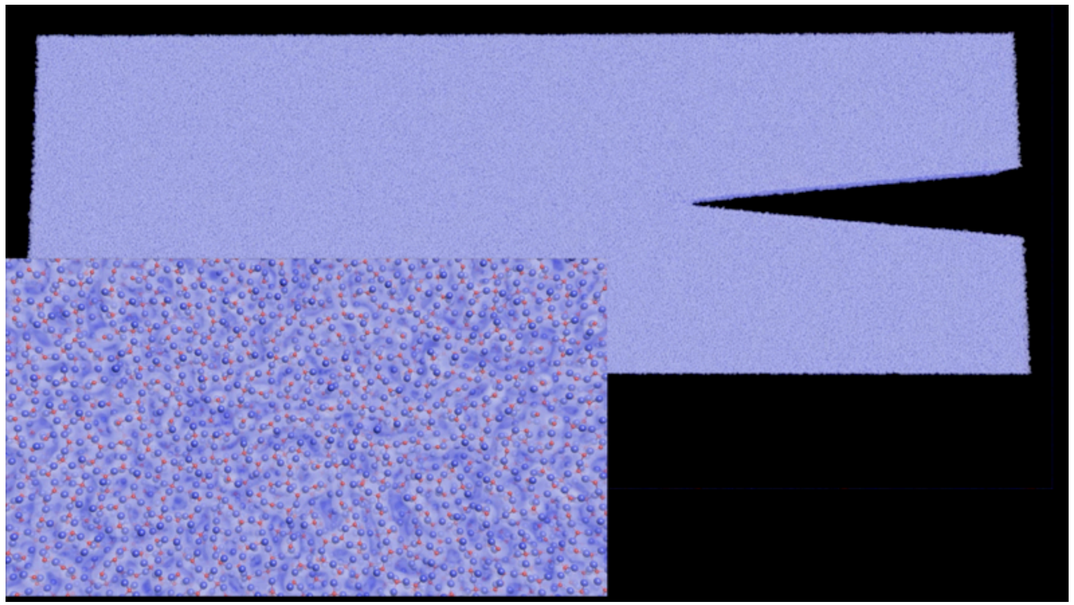
- Immediately visualize + analyze materials data with almost no preprocess pipeline
  - Used grid-based volume + glyph, ray casting on the GPU, view-dependent antialiasing and LOD
  - Volume rendering of molecular orbitals, approximate RBF volumes, volume analysis

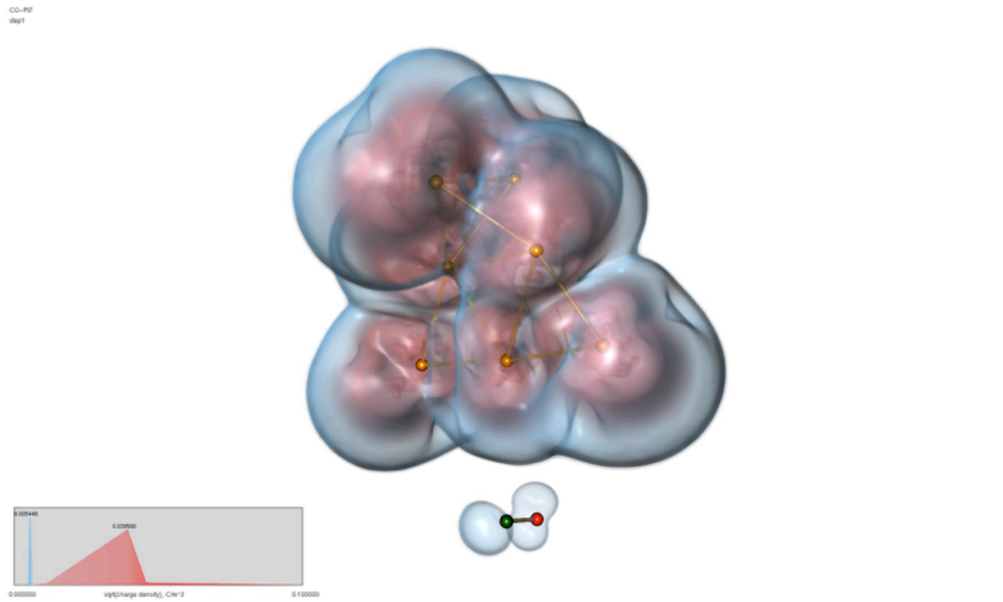


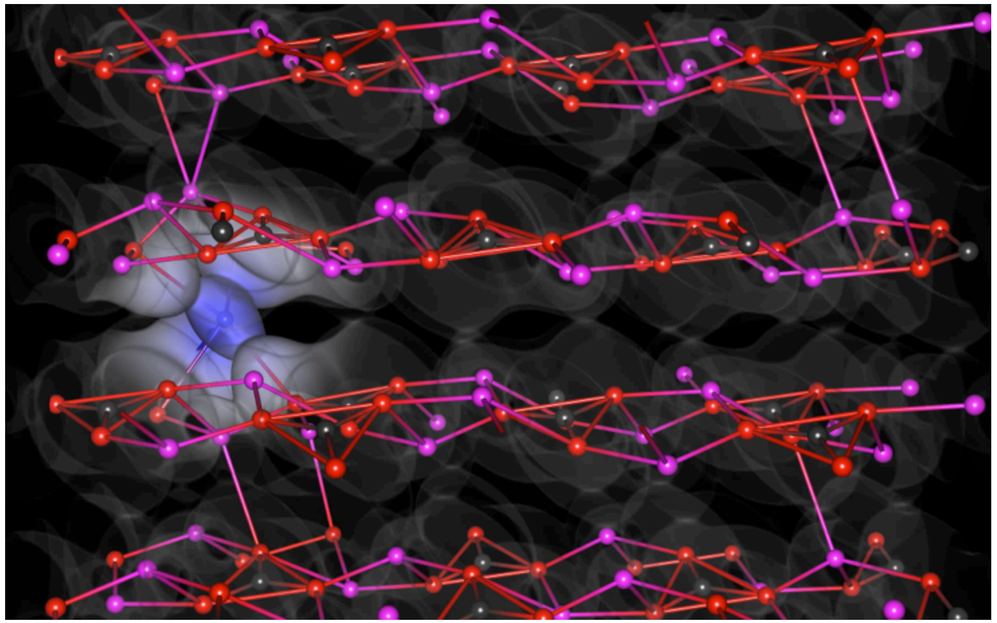


Khairi Reda, Aaron Knoll, Ken-ichi Nomura, Michael E. Papka, Andrew E. Johnson, and Jason Leigh.
Visualizing Large-Scale Atomistic Simulations in Ultra-Resolution Immersive Environments. Proc. IEEE LDAV, pp 59-65, 2013.

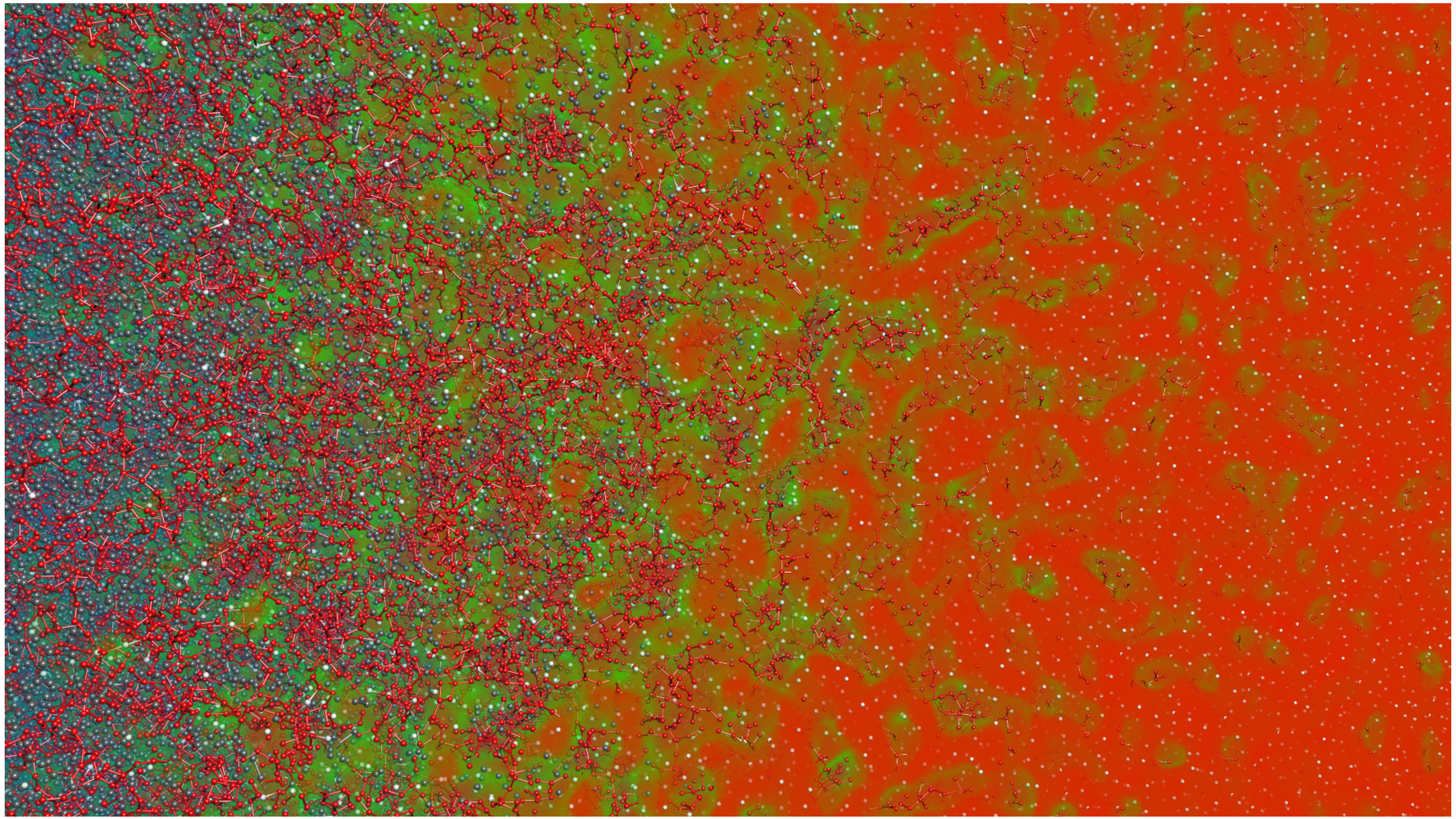
Production vis with Nanovol







Where Nanovol broke



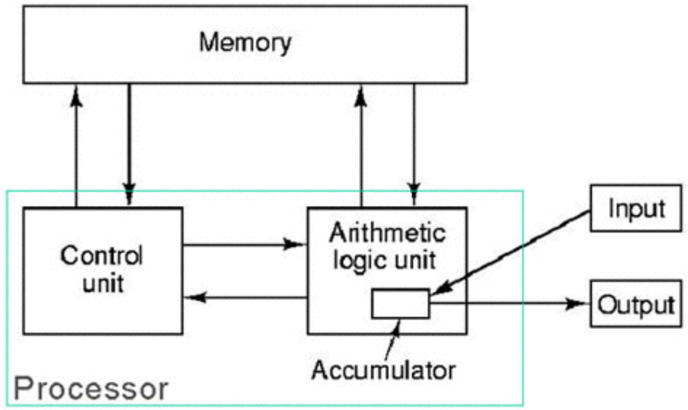
15M ANP3 aluminum oxidation dataset (~1 GB / timestep) — Ken-ichi Nomura, USC
Could only fit a 0.5 voxel-per-Angstrom volume in memory on a 680 GTX!
Coarse macrocell grid, lots of geometry, very slow performance (0.2 fps @ 1080p with sticks)

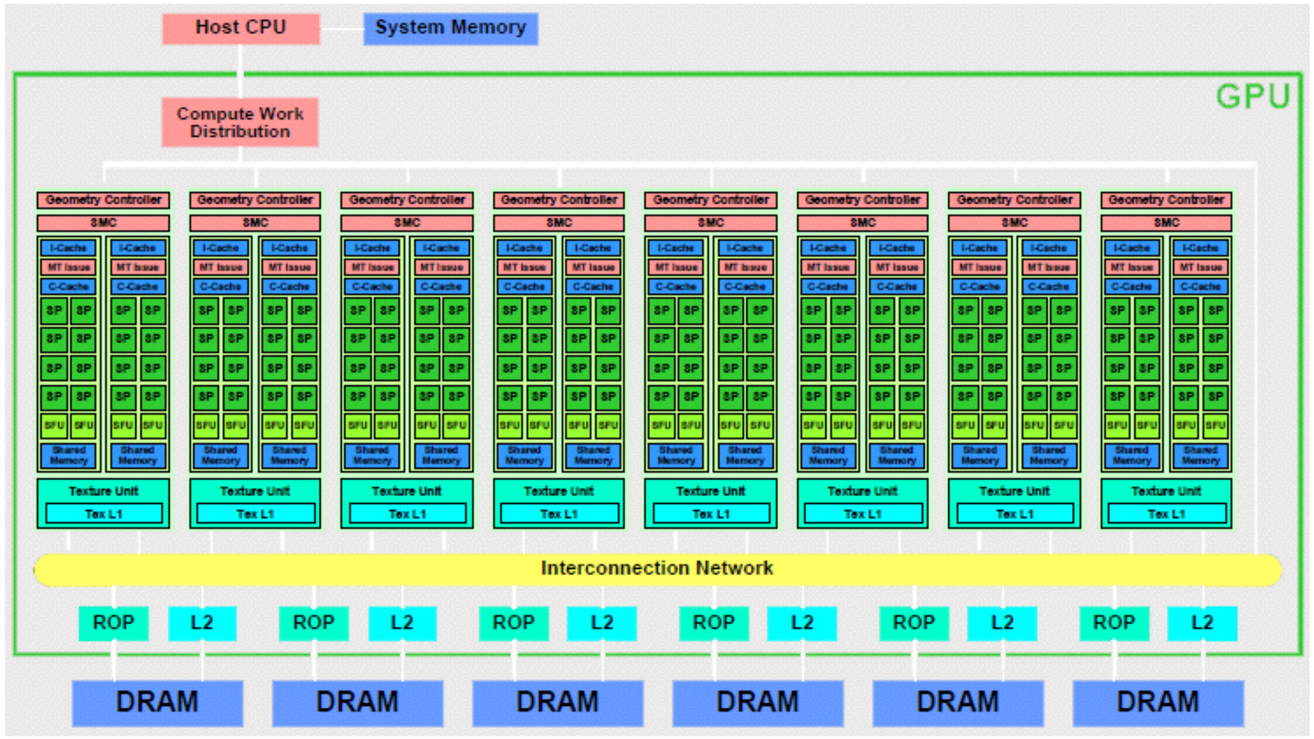
Where Nanovol broke

- Problems:
  - Mismatch between glyph and volume data resolution
  - **Slow PCI bus, lack of memory on GPU.**
- Possible solutions:
  - engineer out-of-core solutions for ball-and-stick, particle + volume data
  - use compression to squeeze data into GPU memory.
  - **Use CPUs.**

Part II: CPU-based Visualization

Why would anyone use a CPU for visualization?!!!!



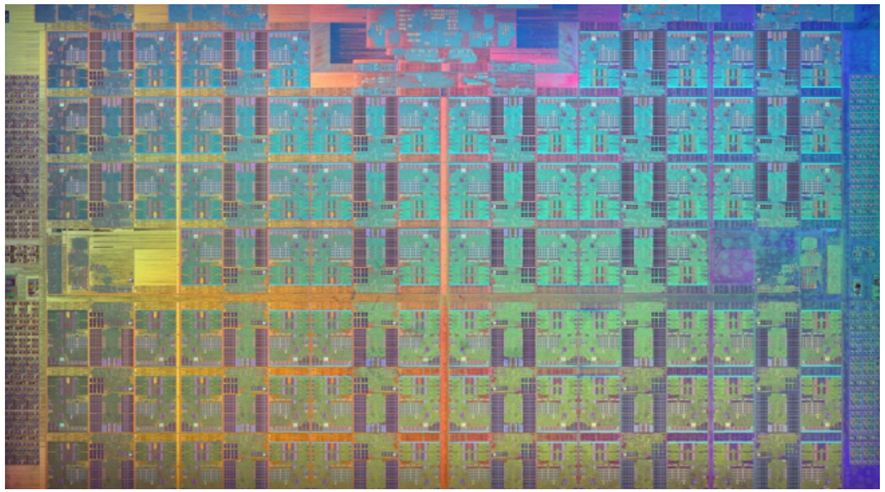


CPU
(e.g., Von Neumann 1945)

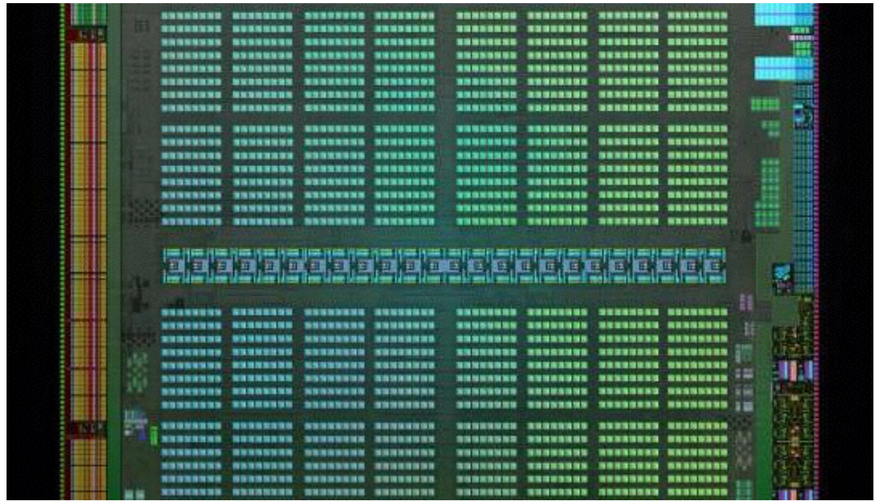
GPU
(e.g., NVIDIA G80, 2006)

Not to mention… vis is graphics, and GPUs are designed especially for graphics… right?

KNL vs Pascal

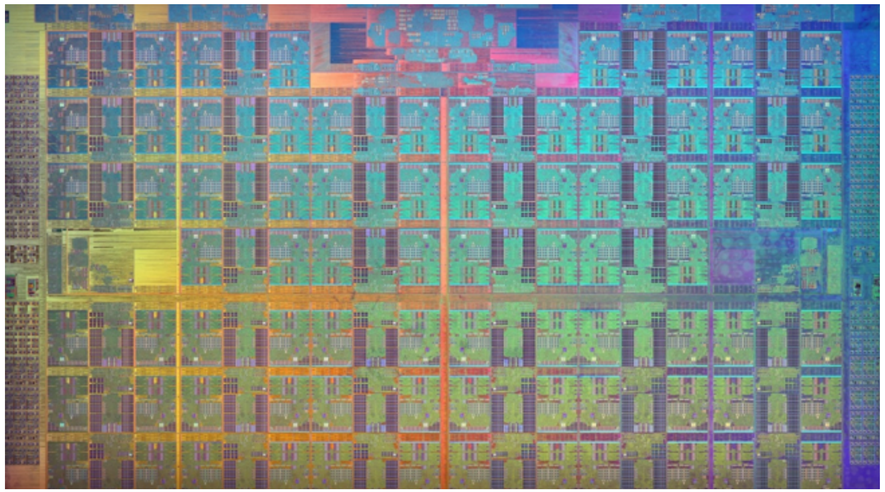


Intel Xeon Phi “KNL”
72 physical “cores”
Two 8-wide DP SIMD lanes / core
3 TF DP

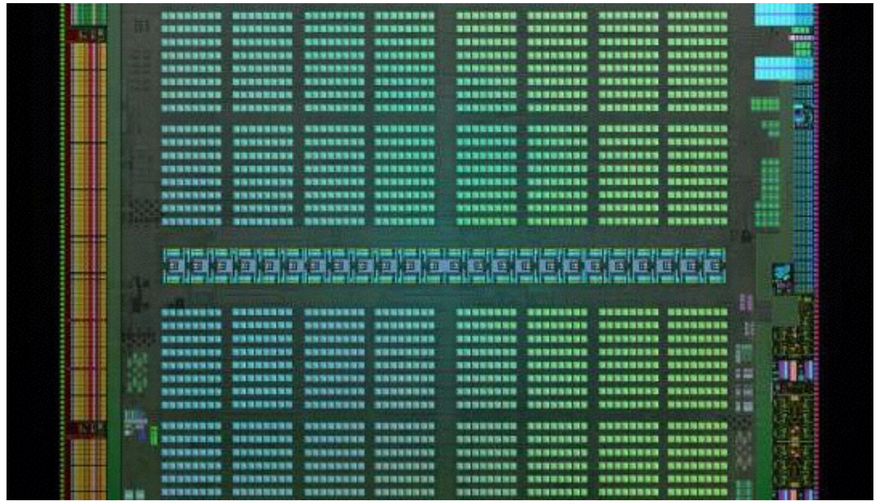


NVIDIA Tesla GP100
56 SM's
32 cores/SM (FP64)
5.3 TF DP

KNL vs Pascal



Intel Xeon Phi “KNL”
72 physical “cores”
Two 8-wide DP SIMD lanes / core
3 TF DP
Up to 384 GB DRAM ***



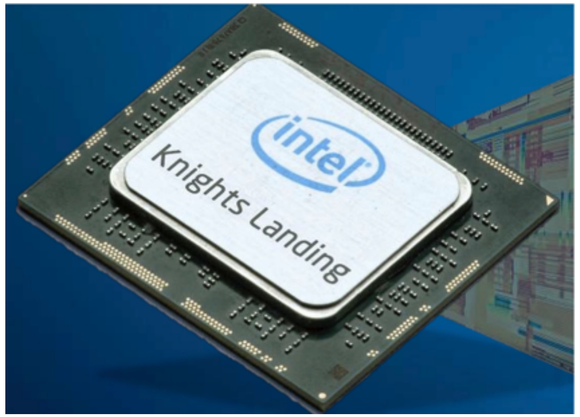
NVIDIA Tesla GP100
56 SM's
32 cores/SM (FP64)
5.3 TF DP
Up to 16 GB NVRAM ***

(***Actual RAM size and speed may vary. KNL has 16 GB on-package MCDRAM used as cache, or in other very confusing ways. Pascal has NVLINK, possibly fast RMA.)

KNL (Intel Xeon Phi) vs CPU (Intel Xeon)

- KNL is more “CPU like” than its Xeon Phi predecessor KNC was, but still more GPU-like than Xeons.
  - Not a “coprocessor” — a full CPU running its own OS
- >2x as energy efficient and ~2x the peak FLOPS of dual-socket Broadwell
- ~1/2 as fast (or worse) for unvectorized code.

KNL has 1/2 the performance of this 4-socket workstation… for ~1/12th the price.
(it’s a lot quieter, too!)



64-core 1.3 GHz Xeon Phi 7210 DAP, 96 GB RAM
Roughly $5K

72 core, 2.5 GHz 4-socket Haswell-EX E7-8890 v3, 3 TB RAM
Roughly $60K (?)

Why else use CPUs?

- **In situ visualization.**
  - IO is very slow
  - We can't throw away time steps forever.
  - We need to start doing vis at larger scales, and on the compute resource
    - ideally after some in situ analysis / filtering, but before data are archived to disk
  - 3D XPoint and parallel IO will help, but this problem isn't going away.
    We need to be able to render at the same scale that we are computing at.
- What if we do vis on the HPC resource itself?

top500.org

Top 10

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
6	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
7	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
9	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	**Hazel Hen** - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	
10	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834

top500.org

CPU-based

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
6	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
7	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
9	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	**Hazel Hen** - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	
10	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834

top500.org

GPU-based

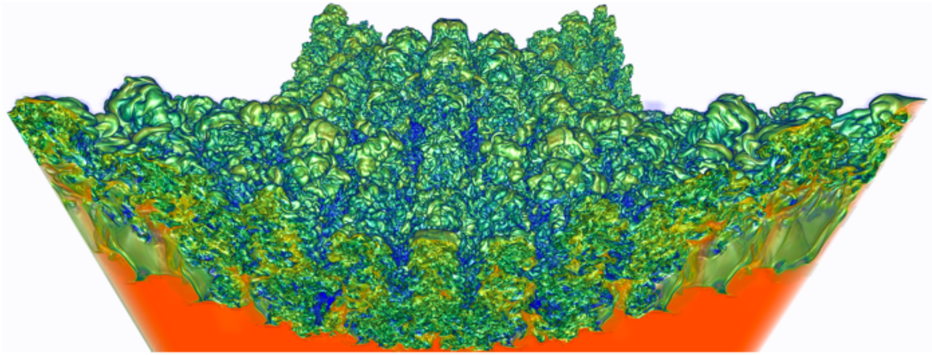
Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
6	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
7	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
9	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	**Hazel Hen** - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	
10	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834

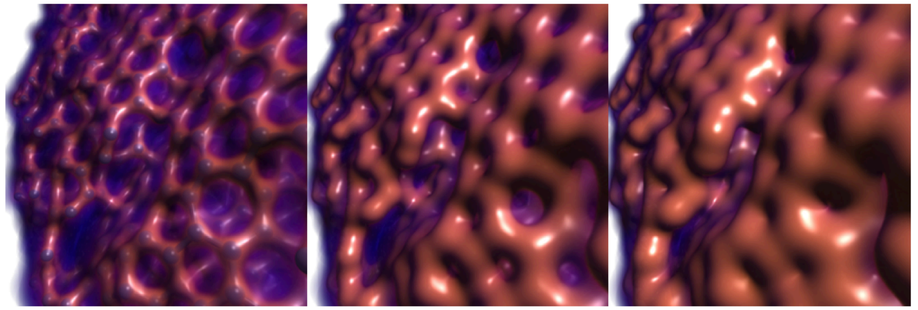
How do we build vis solutions for CPU?

- The right goals and algorithms (**research**), and usable software (**production**).

- Before 2013: comprehensive CPU rendering solutions did not yet exist
  - OpenRT, Manta, targeted *graphics* — had major shortcomings.

  Strong evidence CPU-based visualization was possible, and desirable:
  - Knoll et al. Pacific Vis 2011:
  Volume rendering an 8 GB dataset on 8-core CPU workstation faster than a 128-node GPU cluster
  - Wald et al. Siggraph 2014:
  Embree: *acceleration structure builds are no longer a major bottleneck.*



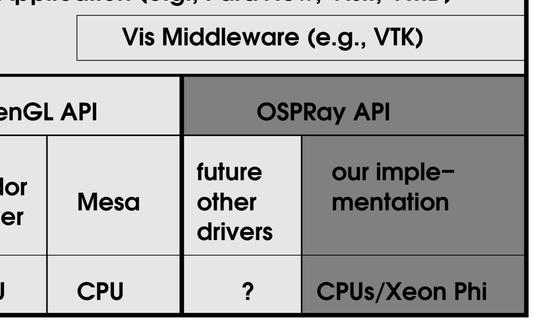


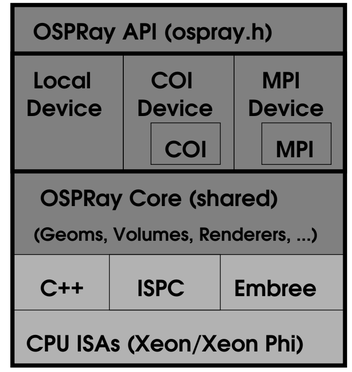
- 2013—2015: Experiments in IVL show that KNC Xeon Phi is competitive, and sometimes better than GPUs
  - Knoll et al. Eurovis 2014:
  RBF volume rendering shown to be 20x faster on KNC than on an NVIDIA K20 GPU

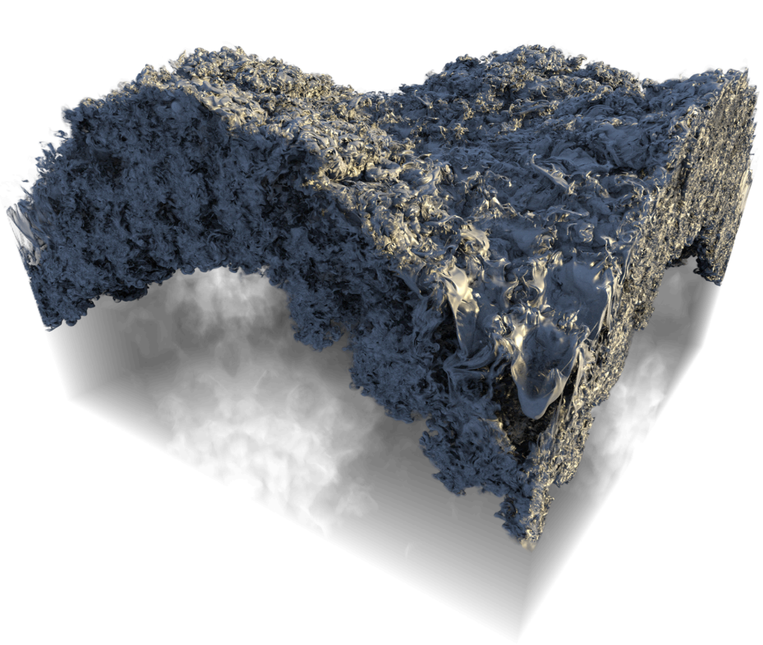
- 2015 —: OSPRay, production-ready CPU ray tracing for visualization.

OSPRay

- **Ray tracing system and API for visualization**
  - Frame buffers, cameras, scenes, data management
  - Polygonal surfaces, implicit surfaces (isosurfaces), glyphs, streamlines, volumes
  - Ray tracing (true AO, global illumination, hard shadows)
- Uses the Intel SPMD program compiler (ISPC) for fast vectorization from kernel code
  - "GLSL / CUDA for CPUs"
- Specifies an API for visualization
  - Similar to OpenGL (but simpler), with additional ray tracing and visualization semantics.
- Integrated into main "indirect" production vis packages (ParaView, VisIt, VMD)
- Open-source (BSD Clause 2 license) and free to use!
- Often almost as fast (or faster) than GPU approaches — **and (almost) never runs out of memory!**





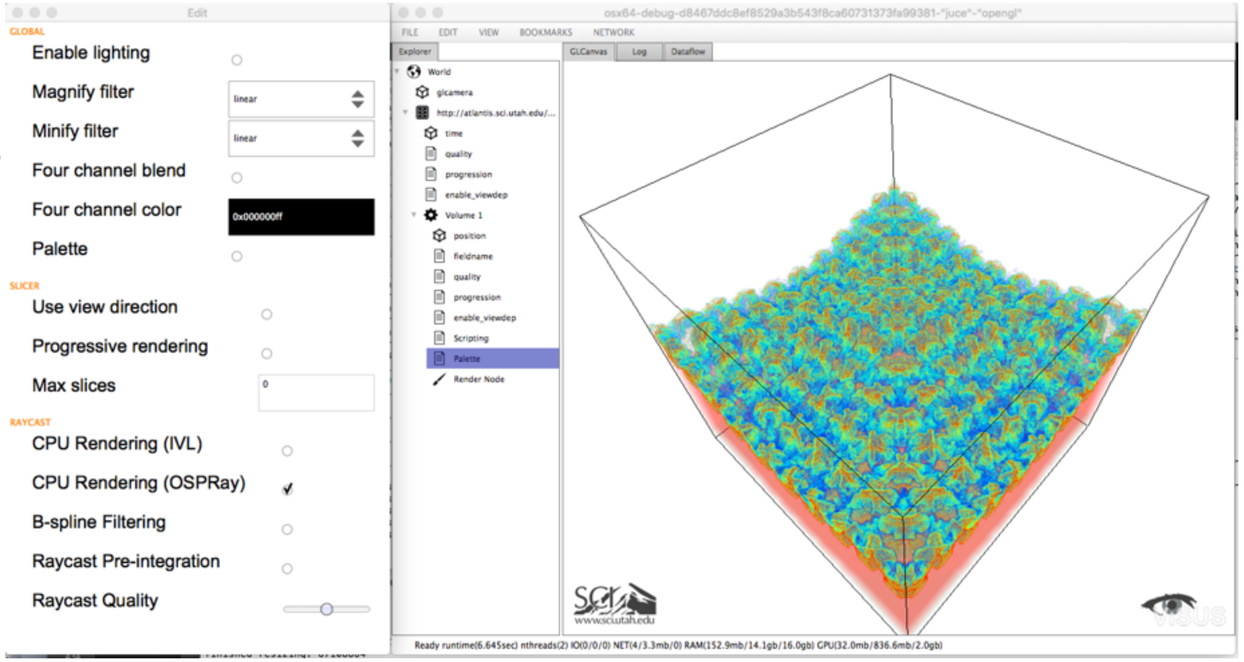


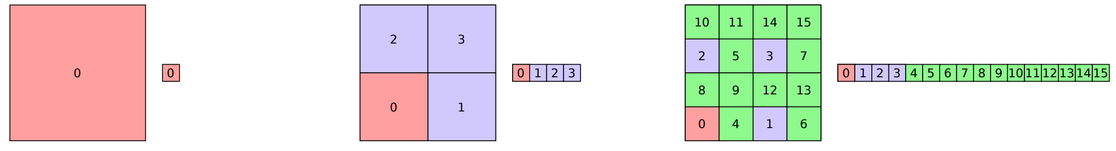
Ingo Wald, Gregory P Johnson, Jefferson Amstutz, Carson Brownlee, Aaron Knoll, James Jeffers, Paul Navratil.
OSPRay: A CPU Ray Tracing Framework for Scientific Visualization. IEEE Vis 2016 (accepted for publication).

Part III: OSPRay integration and related work

ViSUS and PIDX

- **ViSUS**: “Dynamic streams for visualization” — query/analyze scientific data at any resolution, from a remote disk resource.
  - Key technology, “**PIDX**”, a multi-resolution parallel disk storage format.
  - **“Cloud computing for scientific vis”**
- 2014-2015: testing using IVL(CVL)-based CPU volume renderer
- 2016: OSPRay backend
- OSPRay volume rendering is **~3x faster than IVL CPU backend** on 4-socket Xeon E7-8890 v3 (Broadwell)
- Challenges:
  - Rendering not really a bottleneck!
  - Resident data are typically small (16 MB!)
  - Intel gen-core (Iris Pro) works great! (“real GPUs”, OSPRay are overkill)
  - Future work: **combine IDX query with OSPRay block_bricked volume?**

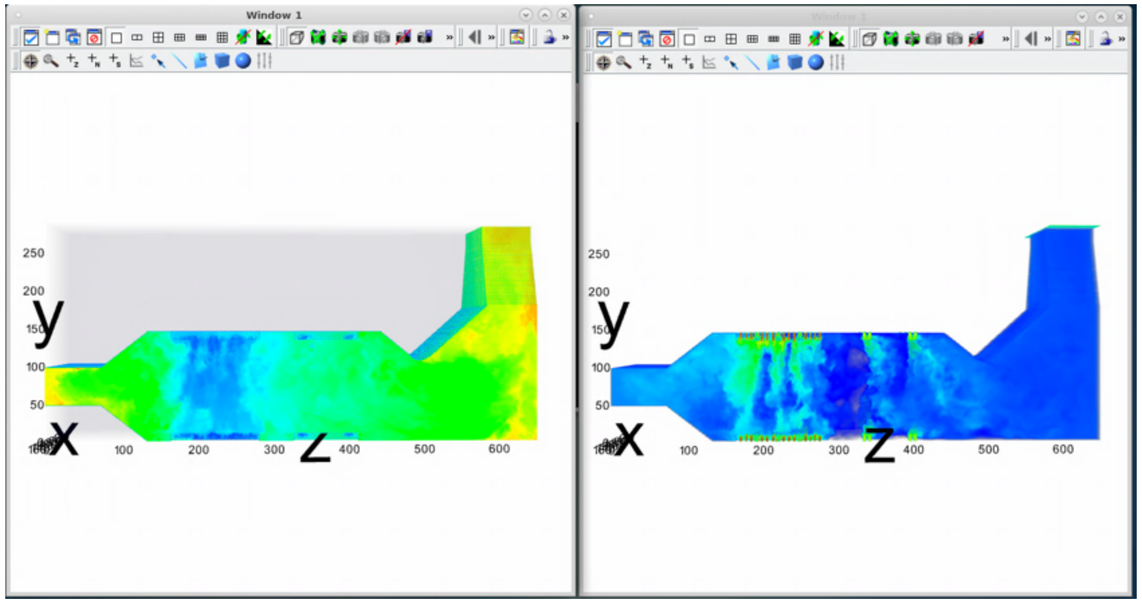


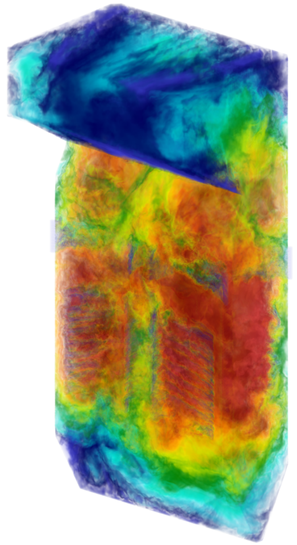


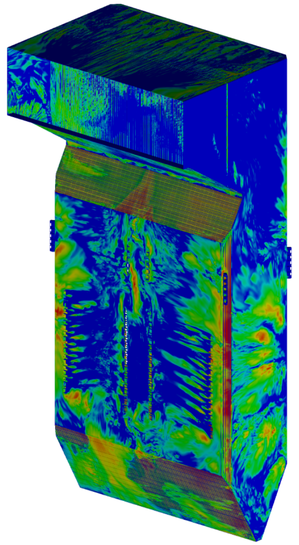
(a) RR_0 (b) RR_2 (c) RR_4

Sidharth Kumar,* John Edwards,* Peer-Timo Bremer,*‡ Aaron Knoll,* Cameron Christensen,* Venkatram Vishwanath,† Philip Carns,† John A. Schmidt,* Valerio Pascucci*·
Efficient I/O and storage of adaptive resolution data. Proc. Intl Conf for High Performance Computing, Networking, Storage and Analysis (Supercomputing 2014)

VisIt + PIDX + OSPRay + Uintah and OSPRay







" With PIDX I/O time came down from 50% of total simulation time to 7%, thus allowing us to dump more data more frequently and have a much better understanding of the actual science."

– Ben Isaac (PhD, PIDX user and Research Associate at Institute for Clean & Secure Energy)

69.3 Million Compute Hours

260,712 Cores

~200 Terabytes

- VisIt and PIDX: used in production for the Uintah coal boiler efficiency computations, 350M hour 2016 DOE INCITE award (PI, Martin Berzins)
- Leverage two branches of VisIt 2.10: **VisIt-OSPRay** (Alok Hota, Jian Huang, Hank Childs) and **VisIt+PIDX** (Steve Petruzza, Valerio Pascucci)
- Visualizations currently performed on Cooley (GPUs): **now possible on Theta KNL's!**

From Fall 2016 Uintah PSAAP TST meeting — Valerio Pascucci.

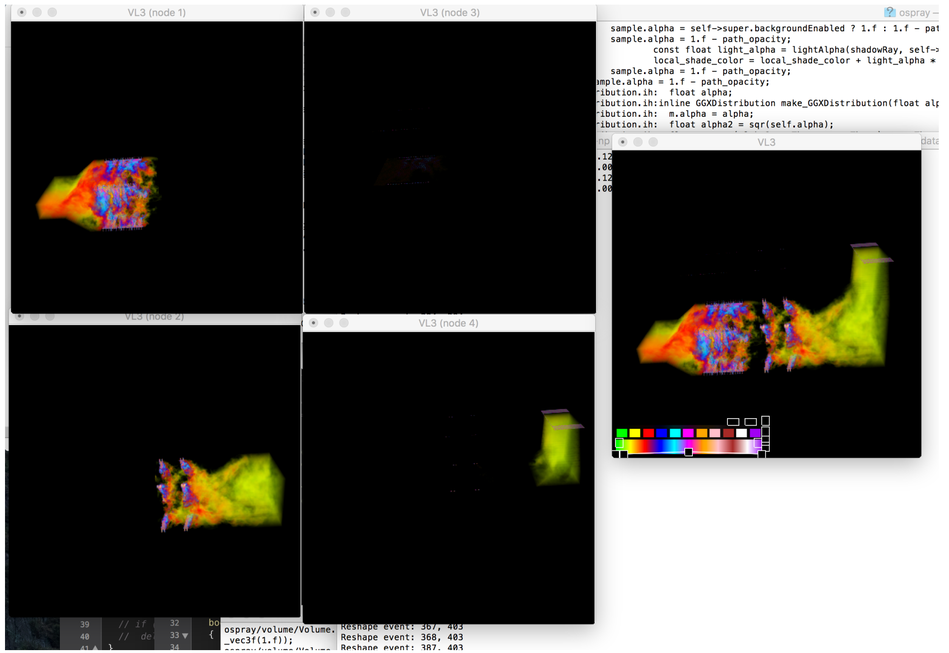
vl3

- Special-purpose, large-scale volume rendering API from Argonne National Laboratory
  - a data-parallel “direct visualization” framework
  - designed for large distributed data (particle, structured grid)
  - Rizzi et al. EGPGV 2015: 30 billion particle HACC data

- Originally for GPU clusters (GLSL, CUDA) — now for CPU/Phi using OSPRay.
  - Function on Theta KNL cluster and upcoming Aurora supercomputer

- OSPRay structured volume renderer backend using “scivis” renderer
  - similar to GLSL invocation

- OpenSWR used for CPU-based compositing
  - future: OpenMP-based implementation, similar to Grosset et al. EGPGV15



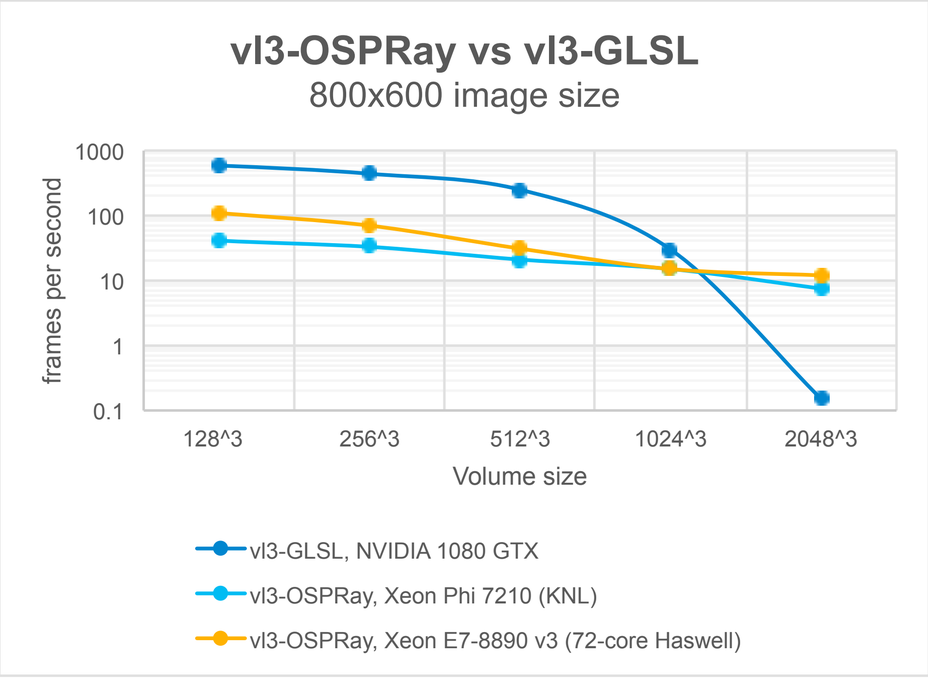

```
renderer = ospNewRenderer(”scivis");
volume = ospNewVolume("shared_structured_volume");
ospSetString(volume,"voxelType","float");
ospSetVec3i(volume,"dimensions", (const osp::vec3i&)dimensions);
OSPData data = ospNewData(nVoxels,OSP_FLOAT,fdata,OSP_DATA_SHARED_BUFFER);
ospSetData(volume, "voxelData",data);
ospSetVec3f(volume, "boundingBoxMin", bbox_min);
ospSetVec3f(volume, "boundingBoxMax", bbox_max);
…
ospRenderFrame(framebuffer, renderer, OSP_FB_COLOR);
```

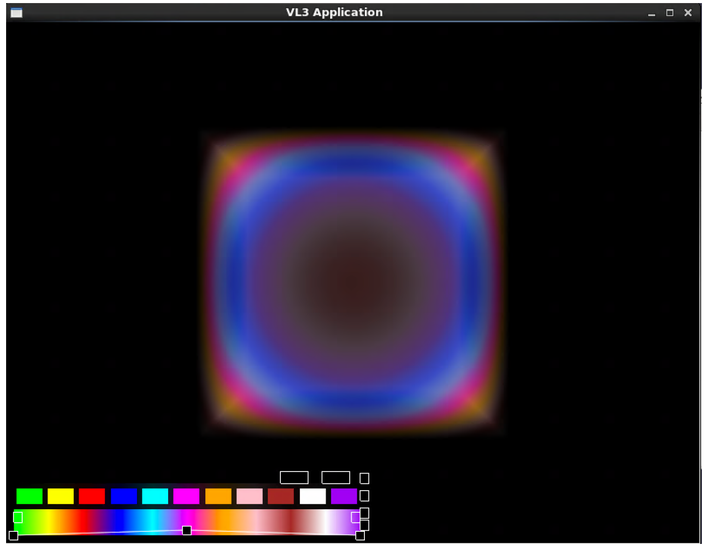
vl3-ospray KNL bakeoff

- Comparing vl3-ospray with vl3-GLSL backend, on KNL, 72-core Haswell, and GPU.
  - uses “raycast_volume_renderer”, early termination disabled for an “apples to apples” comparison
  - new “scivis" renderer uses more optimization

- **KNL does surprisingly well** — “sweet” spot” around 1k^3 — 2k^3 volume data.
  - “cache mode” with DRAM—MCDRAM works!

- Much slower than NVIDIA 1080 GTX GPU for small data, much faster than GPU when out-of-core.

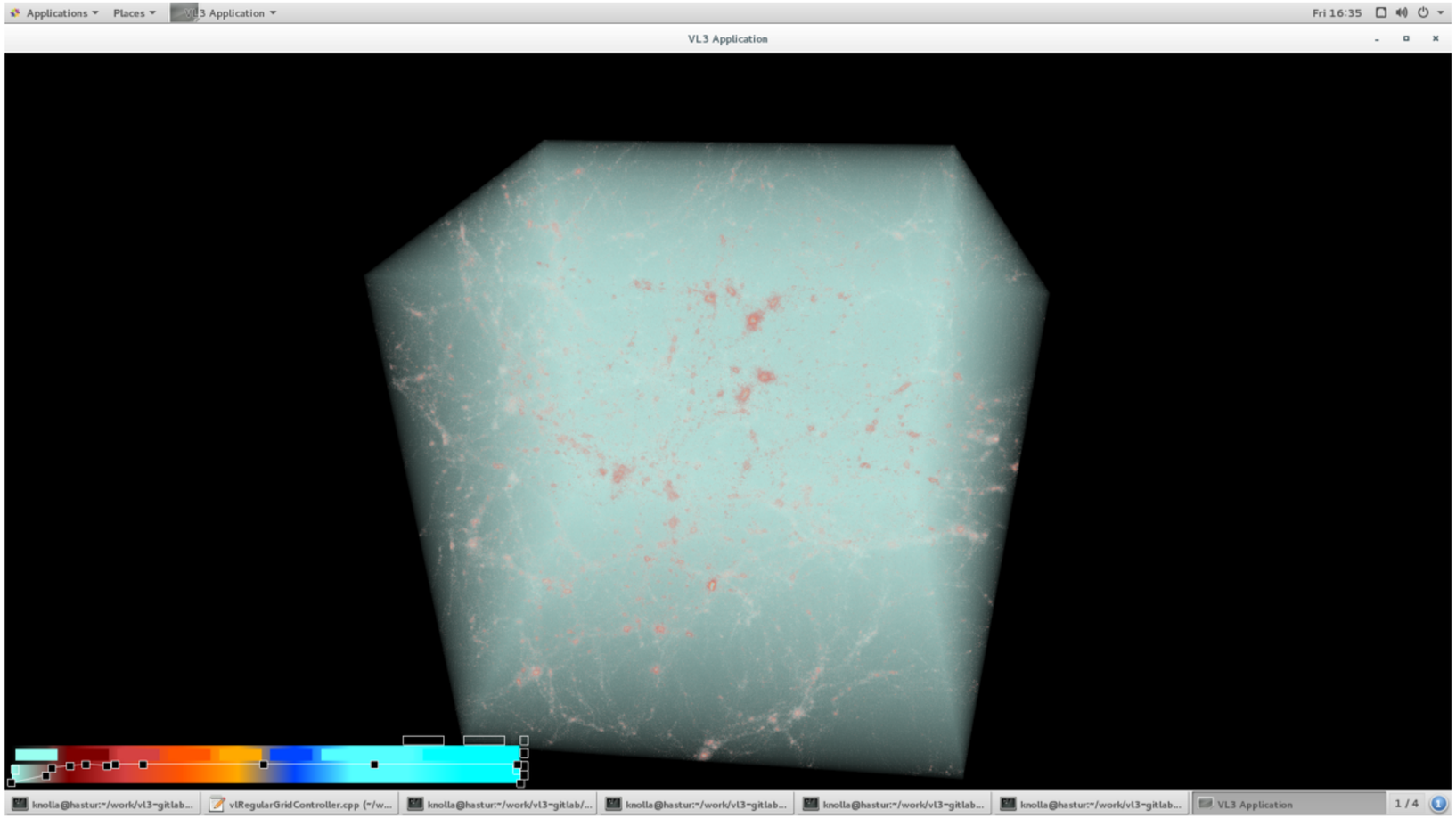




NVIDIA Geforce 1080 GTX 8 GB in a dual Xeon E5-2650 with 64 GB DRAM
Xeon E7-8890 v3 is a 72-core, 2.5 GHz 4-socket Brickland-EX platform with 3 TB DRAM
Xeon Phi 7210 is a 64-core 1.3 GHz KNL with 16 GB MCDRAM and 96 GB DRAM

vl3-ospray - with “scivis” renderer

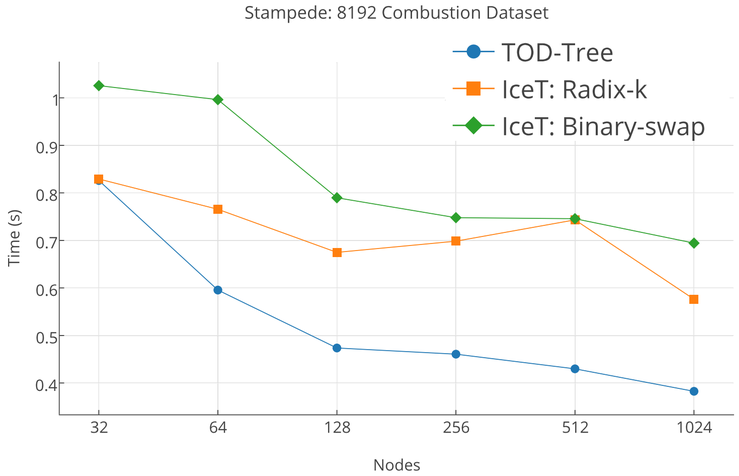
- Noisy volume data require higher sampling rate.
  - new in OSPRay v1.1.0 — adaptive volume rendering
  - On one node:
  32 GB HACC dark matter density volume, resampled from 500 M particles (74 GB), ~7–10 fps at 1080p on a KNL
  - What about data parallel at large scale?

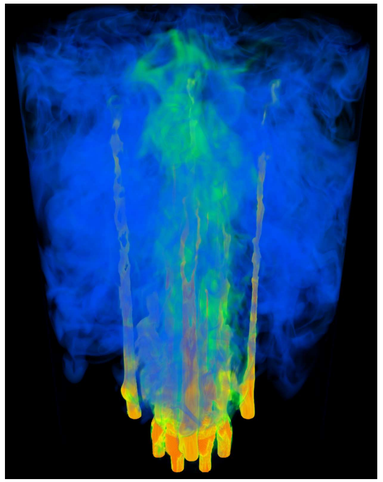


Part IV: OSPRay and CPU vis research

Faster parallel compositing on CPUs

- OSPRay data-parallel rendering is great
- Our results: consistently interactive compositing up to 4K resolution on 1K nodes, thanks to CPUs!
  - use a tree structure to decompose, and carefully overlap communication with computation
  - 2x faster than IceT using OpenGL
  - don’t bother sending frames across the PCI bus to the GPU!
- Variant of OpenMP SIMD compositing base being integrated in vl3.

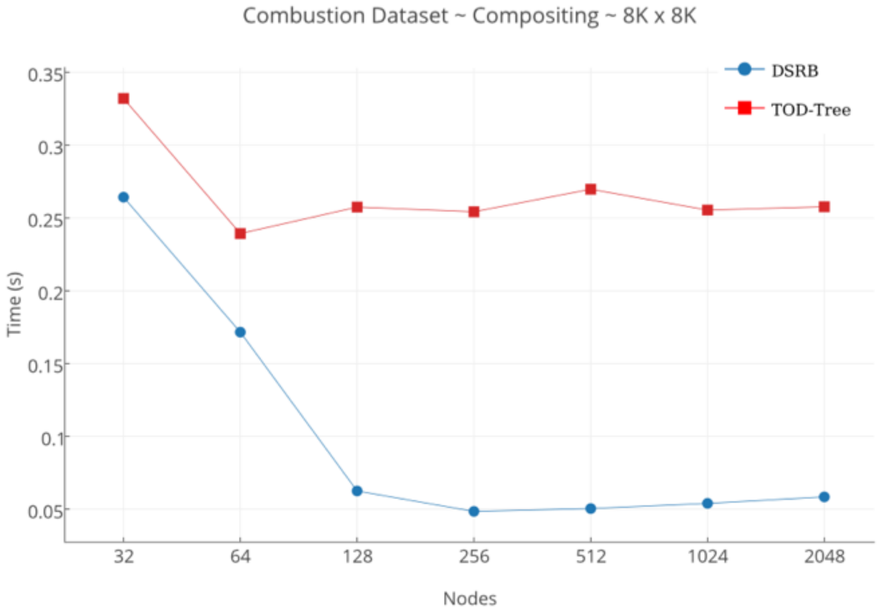


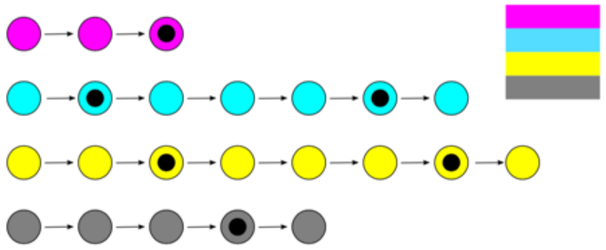


P. Grosset, M. Prasad, C. Christensen, A. Knoll, C.D. Hansen. “TOD-Tree: Task-Overlapped Direct send Tree Image Compositing for Hybrid MPI Parallelism”. Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV) 2015

Dynamically scheduled region-based compositing

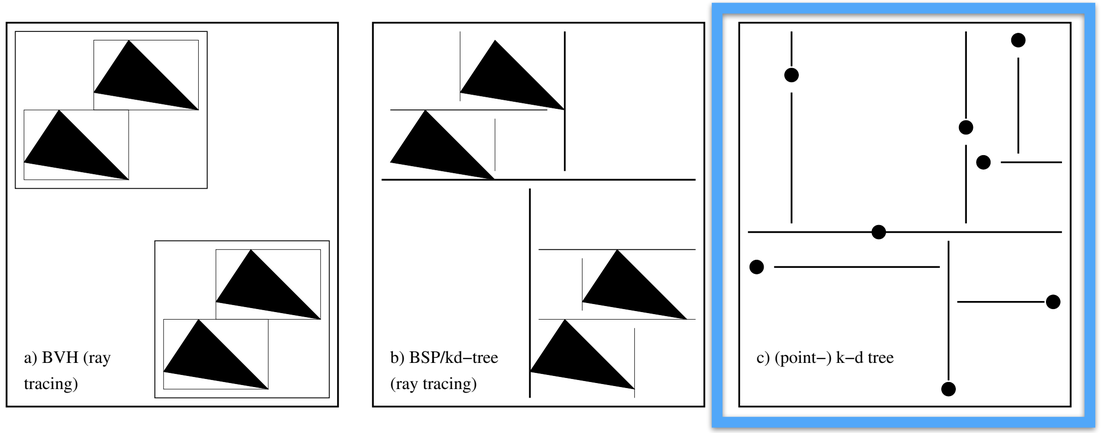
- Faster large-scale compositing for unbalanced visualization workloads
- Improves on previous TOD-Tree paper (EGPGV15) and GPUDirect extension (IEEE TVCG 2016)
  - scales up to 2K nodes on Edison
  - Simple OpenMP CPU compositing competitive with optimized GPU techniques!
- Similar approaches could be used into OSPRay distributed data API.





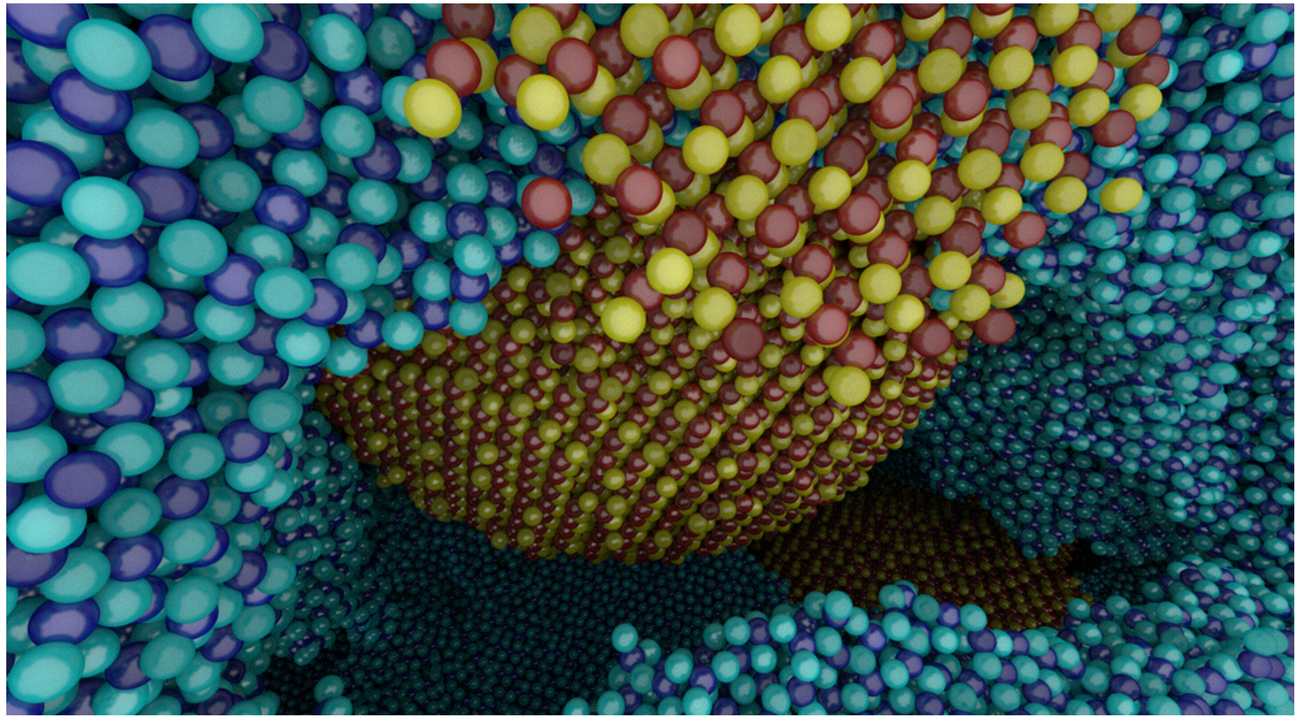
Pascal Grosset, Aaron Knoll, Chuck Hansen. “Dynamically Scheduled Region-based Compositing.” Eurographics Symposium on Parallel Graphics and Visualization 2016.

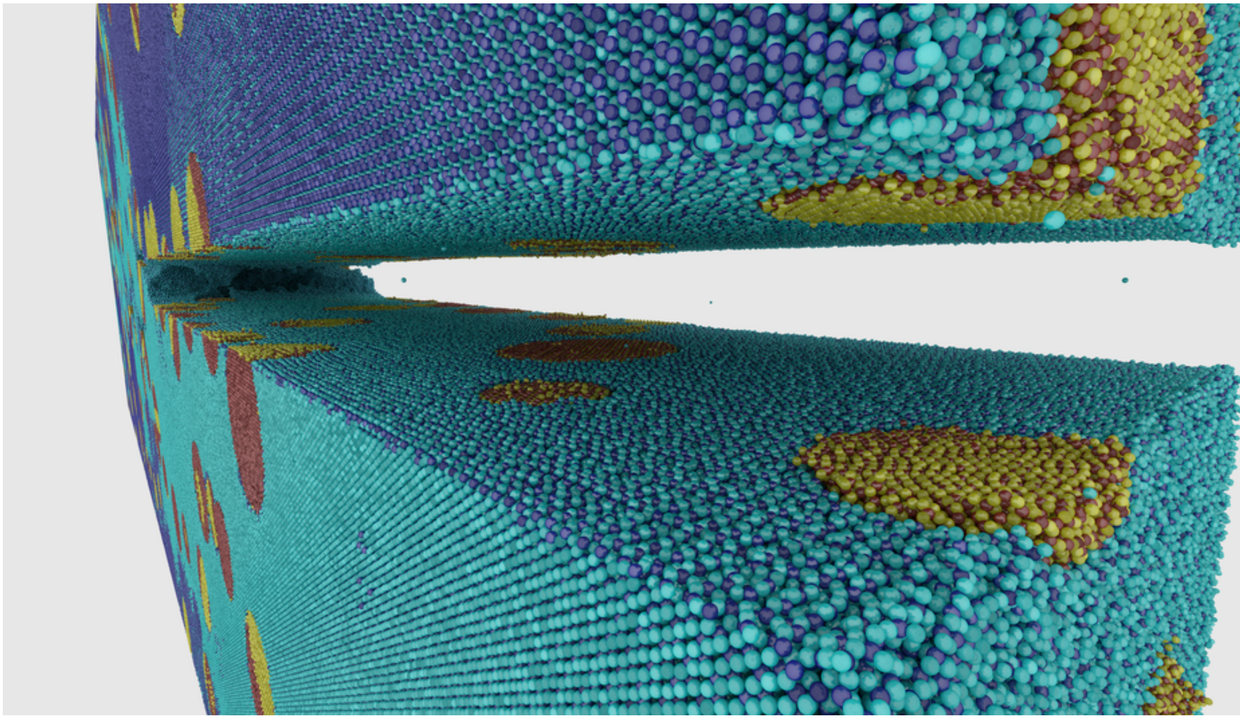
P-k-d Trees: low-footprint particle storage



- OSPRay uses the Embree BVH by default to accelerate particle data
  - fast to build and traverse with Embree, but with a ~4x memory overhead!
- A better solution: **the balanced k-d tree, or "point" k-d tree**
  - The data *are* the acceleration structure
  - zero (or little) memory overhead, fast to build
  - ~30% slower to render than state-of-the-art Embree BVH
- Implemented in both OSPRay and IVL (paper written on OSPRay implementation)

P-k-d trees for materials

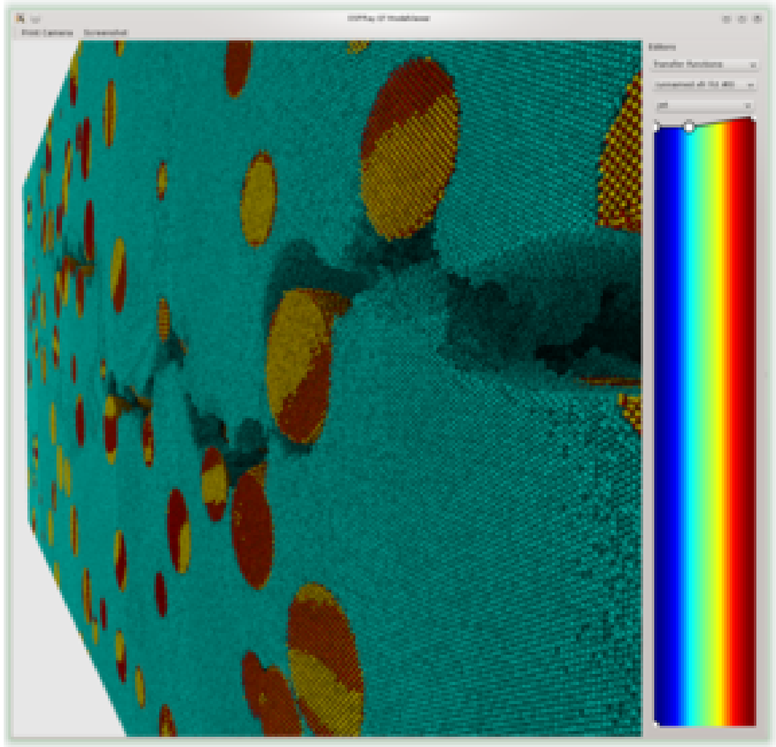


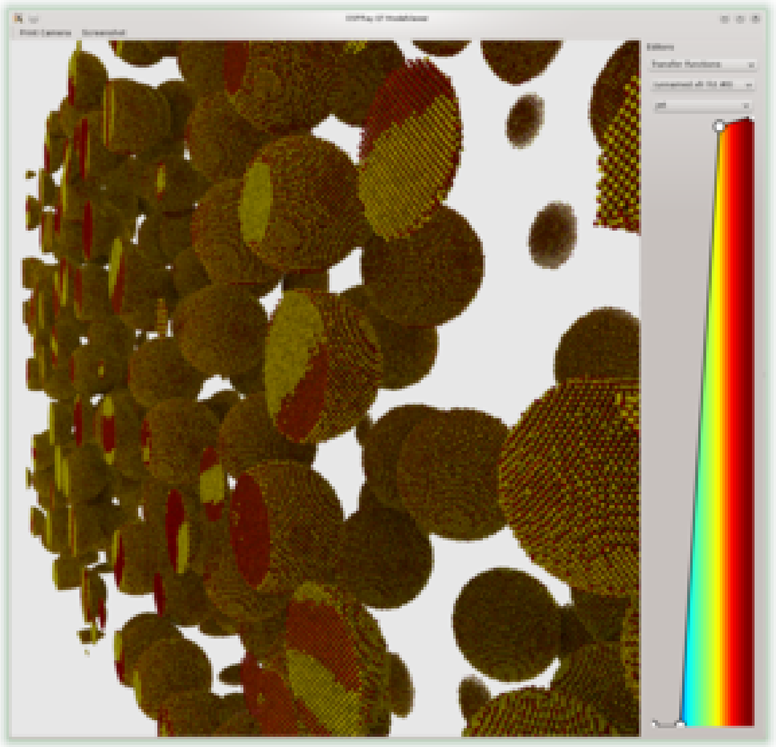


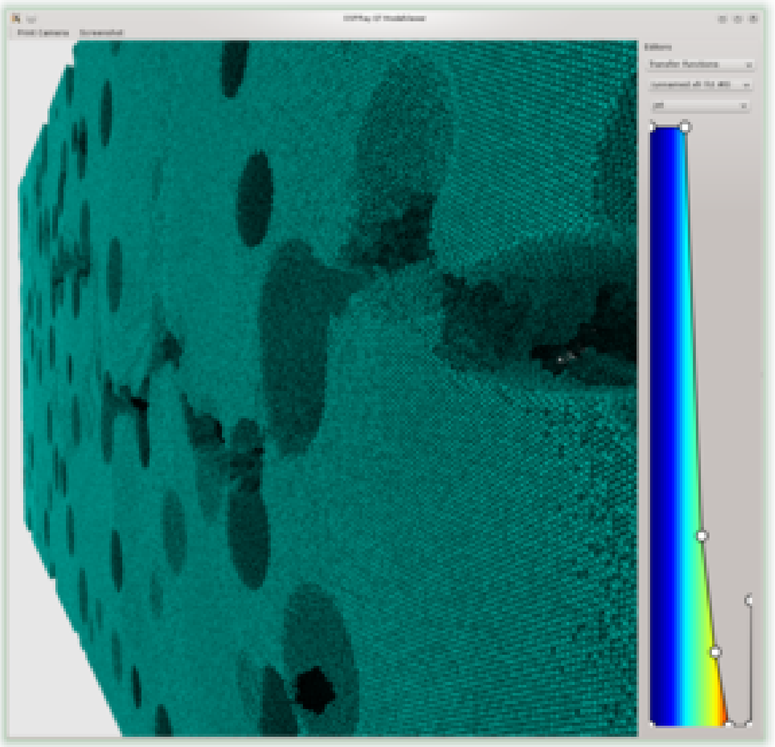
100M atom Al2-O3 SiC alumina-coated nanoparticle MD simulation (Aiichiro Nakano, Rajiv Kalia, USC)
Rendered in OSPRay with path tracing (1 spp with progressive rendering), 2–4 fps at 4K resolution
DOE INCITE allocation at Argonne National Laboratory, 2014

Ingo Wald, Aaron Knoll, Gregory P. Johnson, Will Usher, Valerio Pascucci and Michael E. Papka.
CPU Ray Tracing Large Particle Data with Balanced P-k-d Trees. IEEE Vis 2015

Dynamic filtering with P-k-d trees





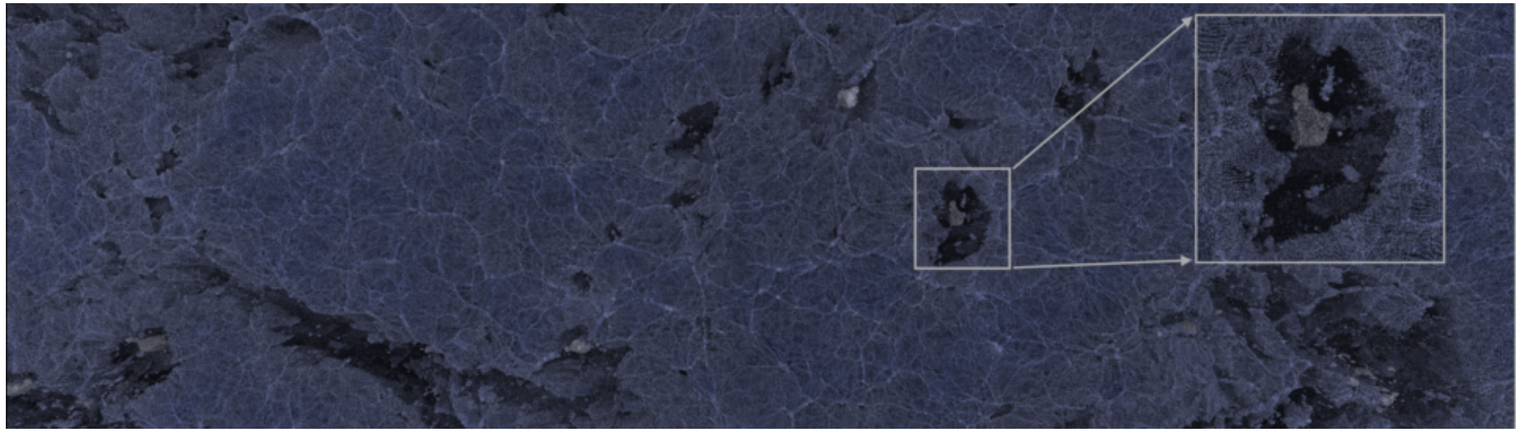


Ingo Wald, Aaron Knoll, Gregory P. Johnson, Will Usher, Valerio Pascucci and Michael E. Papka.
CPU Ray Tracing Large Particle Data with Balanced P-k-d Trees. IEEE Vis 2015

Two different ways to visualize the early universe, ~30 billion particles

1. Mostly opaque with ray tracing.

One 72-core CPU workstation, 3 TB shared memory, P-k-d trees

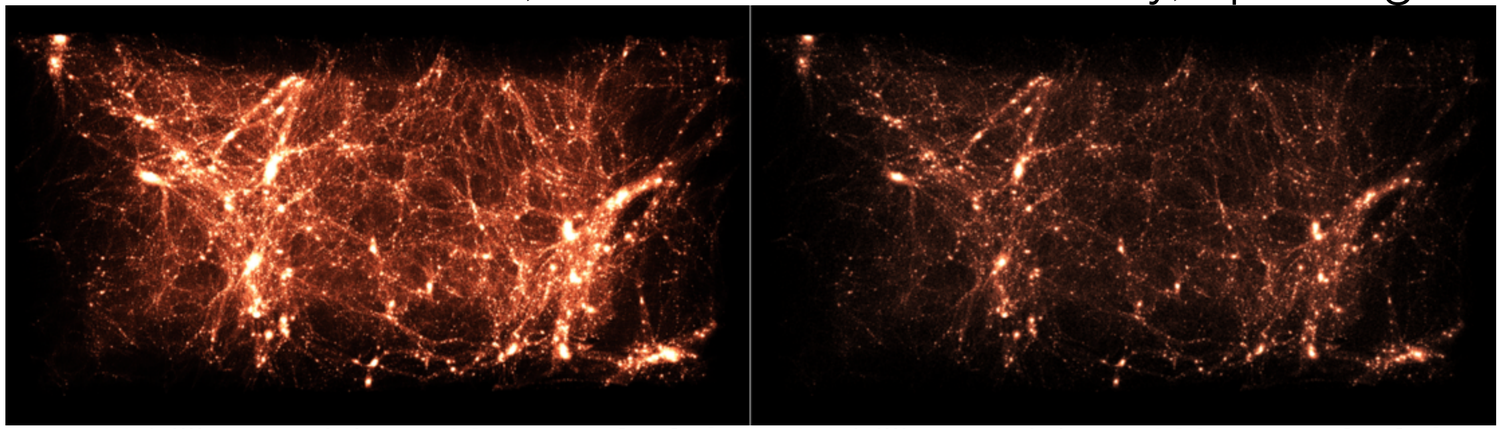


I Wald, A Knoll, G Johnson, W Usher, M E Papka, V Pascucci. “CPU Ray Tracing Large Particle Data with Balanced P-k-d Trees”, IEEE Vis 2015
30 billion particle (450 GB) subset of a PM3D simulation, ray traced with ambient occlusion

6 FPS (72-core 2.5 GHz Xeon E7-8890 v3) at 4096x1920 = **~50 megapixels/s** (MRays/s)

2. Mostly transparent with rasterization:

128-GPU cluster, 1 TB distributed memory, splatting



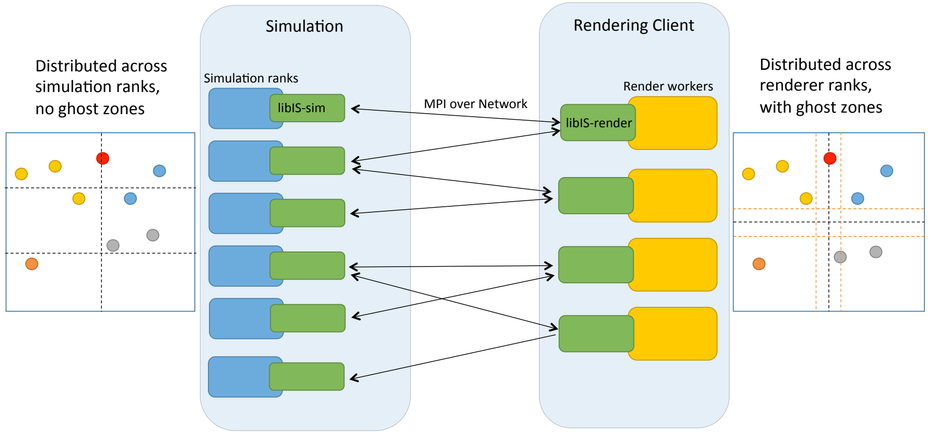
28 billion particles: **~20 megapixels/s** 2.8 billion particles: **~200 megapixels/s**

S. Rizzi, M. Hereld, J. Insley, M. Papka, V. Vishwanath. “Large-Scale Parallel Vis. of Particle-Based Simulations using Point Sprites and LOD”, EGPGV 2015. ~32 billion particle HACC dataset with LOD filtering.

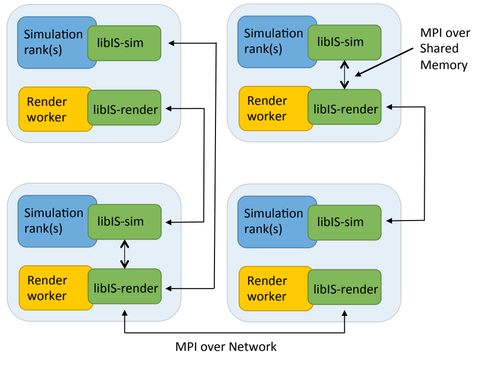
Ongoing research directions

In Situ Exploration with P-k-d trees

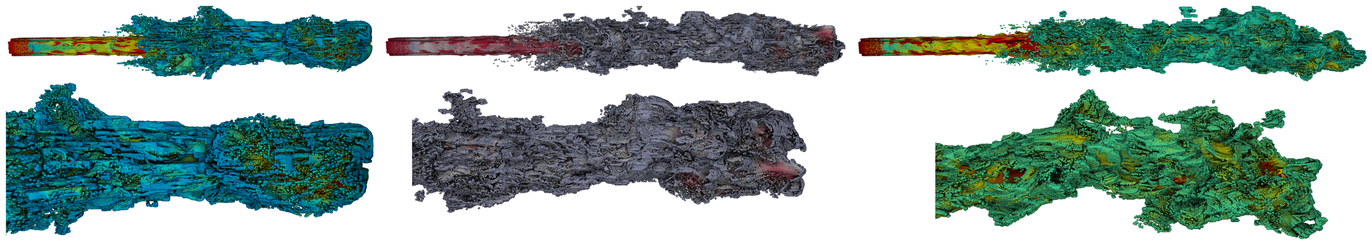
- Loosely-coupled system; simulation connects directly to OSPRay
  - CPU/Phi resources used for both compute and rendering
  - OSPRay client connects/disconnects at will
  - Low memory overhead compared to VTK-based approaches



a) Simulation and OSPRay on different resources



b) Simulation and OSPRay on shared resources



Will Usher, Ingo Wald, Aaron Knoll, Michael Papka, Valerio Pascucci

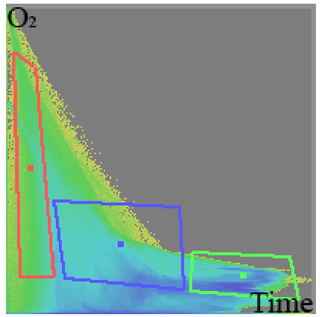
"In Situ Exploration of Particle Simulations with CPU Ray Tracing" Workshop on In Situ Visualization, ISC 2016, Supercomputing Frontiers and Innovations (submitted)

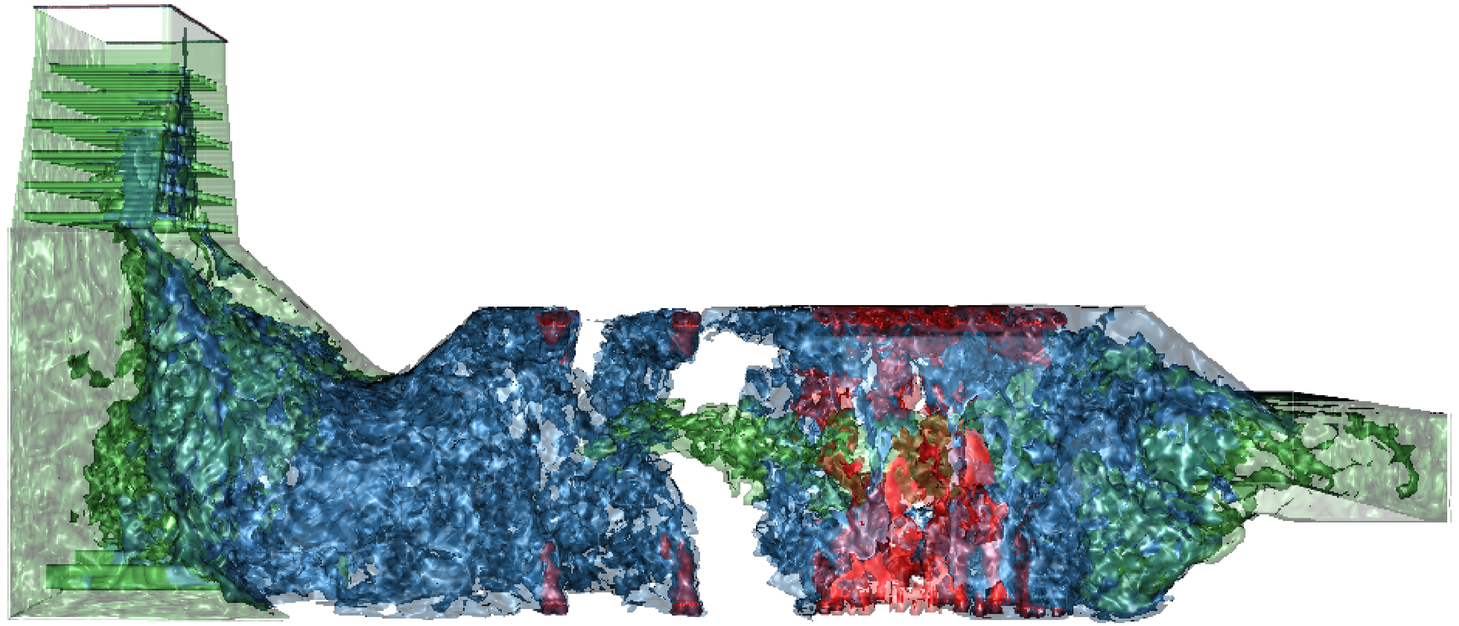
Evolving into the data-parallel API in core OSPRay…

Large multifield data

20 GB / timestep LiAlH2O DFT simulation, courtesy Aiichiro Nakano, University of Southern California
CPU volume rendering using IVL wrappers in Nanovol
Load and visualize all 780 multifields at once!
5K timesteps, 100 TB total

Fiber surfaces: classifying and summarizing multifields



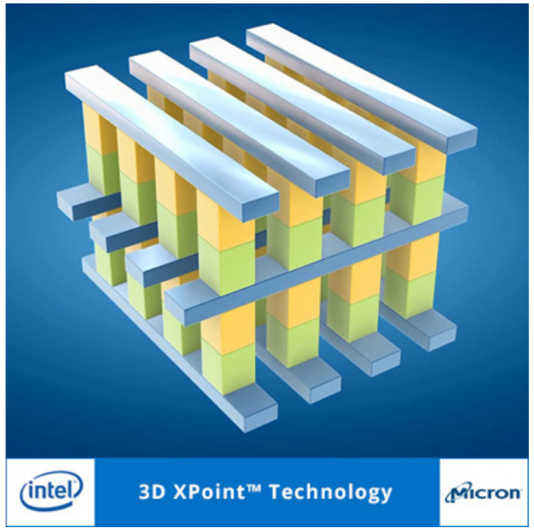


- Fiber surfaces: a multifield equivalent of isosurfaces (Carr et al. Eurovis 2015).
  - Allows a “clean division” of multifield data into interesting regions, based on a scatterplot.
  - Full Uintah BSF simulation: 130 fields! OSPRay and CPUs are needed for the memory!
- New theory needed to extend the technique beyond 2-field data

Kui Wu, Aaron Knoll, Ben Isaac, Hamish Carr, and Valerio Pascucci. Direct Multifield Volume Ray Casting of Fiber Surfaces. IEEE Visualization 2015.

DV: a data model for direct visualization

- Porting GPU code to OSPRay requires effort — and often supports just one type of data (i.e., structured volume)
- How can we simultaneously support multiple data types?
- DV: a data model simplifying GPU-OSPRay ports, designed for direct visualization.
  - data model is defined by the user, vis or simulation code as needed.
  - just-in-time compilation creates data structures, algorithms on demand
    - bypasses classes vs template issues in ISPC, pointer issues on GPU
  - merge data formats for parallel IO and visualization (leverage PIDX, 3DXPoint!)




```
dvCell cell;
cell.addField("voxels", DV_ELEMENTS, DV_FLOAT, 1, 64);
cell.addField("particles", DV_VERTICES, DV_FLOAT, 3, 16);
cell.writeBackend( "jit/_dvCell.h");

dvContainer container(DV_ARRAY, DV_GRID, 3);
container.writeBackend("jit/_dvContainer.h");
```

```
struct _dvCell{
  float voxels[64];     //elements
  vec3f particles[16];  //vertices
}

struct _dvContainer{
  static const int dimensionality = 3;
  ulong dimensions[dimensionality];
  _dvCell* cells;
}
```

Summary thoughts…

- Visualization will remain crucial as long as we are doing computing.

- **CPU-based ray tracing methods are key to achieving long-term scalability — and represent the "bleeding edge" of scientific visualization.**

  - With 3DXPoint, the line between "in situ" and "stored" data is blurry…

  - With Omnipath, the line between "in-memory" and "remote access" is blurry…

  - OSPRay opens up opportunities that did not exist before

    - extending the capabilities of existing "indirect" visualization systems (VMD, ParaView, VisIt)

    - new research in "direct" visualization (vl3, pkd trees)

    - multifield and time-varying visualization

Thank you!

Intel Parallel Computing Center Program
Argonne Leadership Computing Facility
DOE DE-AC02-06CH11357
NSF NSF CISE ACI-0904631

OSPRay team: Ingo Wald, Jim Jeffers, Carson Brownlee, Jeff Amstutz, Johannes Guenther
Intel: Mark West, Brian Napier, Lisa Smith, Joe Curley, Kent Li, Nathan Schulz
Argonne LCF: Mike Papka, Joe Insley, Silvio Rizzi, Ying Li
Argonne Materials Science Division: Kah Chun Lau, Larry Curtiss, Hakim Iddir, Lei Cheng
Argonne Center for Nanoscale Materials: Bin Liu, Maria Chan
Argonne Chemical Sciences and Engineering: Julius Jellinek, Aslihan Sumer

Utah students/staff: Will Usher, Qi Wu, Kui Wu, Pascal Grosset, Attila Gyulassy, Cameron Christensen, John Holmen
TACC: Paul Navratil, Greg Abram
Micron: Ed Caward, Janene Ellefson

VisIt-OSPRay: Hank Childs, Jian Huang, Alok Hota