# Large-Scale Parallel Visualization of Particle Datasets using Point Sprites

Silvio Rizzi*, Mark Hereld*, Joseph Insley*, Michael E. Papka*†, Thomas Uram*, and Venkatram Vishwanath*
*Argonne National Laboratory
†Northern Illinois University
Contact: {srizzi, hereld, insley, papka, turam, venkat}@anl.gov

***Abstract*—Massive particle-based datasets can be efficiently rendered using point sprites. Among other advantages, the method is efficient in its use of memory, preserves the dynamic range of the simulations, and provides excellent image quality. Preliminary results of large-scale parallel rendering of particle data sets using point sprites are presented in this poster. Performance and scalability are evaluated on a GPU-based computer cluster using datasets of up to $3200^3$ particles at resolutions of up to 6144x3072 pixels.**

## I. Motivation

Recent particle-based large-scale simulations are producing vast amounts of data, demanding analysis and visualization algorithms to keep up with their increasing requirements. Mapping particles into regular grids for volume rendering is a possible solution, but the resulting 3D grids may be sparse and make inefficient use of memory. In addition, the mapping of particle positions to fixed grid points results in losses of dynamic range. On the other hand, direct point rendering methods use particle coordinates, avoiding memory sparsity and loss of dynamic range. In addition, hardware-accelerated point rendering is an eminently parallel task and should provide excellent performance and scalability for large-scale datasets.

Among a number of possible direct particle rendering methods, we will be focusing on point sprite rendering. In this poster, we present preliminary results of a parallel implementation of point sprite rendering for large-scale datasets, evaluating its strong and weak scalability using up to 32 billion ($3200^3$) particles on 128 GPUs, at resolutions of up to 6144x3072 pixels.

## II. Related Work

An out-of-core method for rendering massive point clouds using multi-way kd-trees is presented in [1]. Parallel rendering is implemented using the Equalizer framework. OpenGL points are used as rendering primitives on up to six parallel rendering nodes and data sizes scale up to 368 million points. Similarly, in [2], a multi-resolution hierarchy with level-of-detail selection is presented. Particles are rendered as points and expanded using the geometry shader to a screen-aligned face. Data consisting of $2160^3$ particles from the Millennium Simulation Project are used and experiments are run on a single GPU.

A method for high-quality particle rendering with point sprites on GPUs is presented in [3]. Results are shown for up to 2.8 million particles on a single GPU. In [4], CUDA kernels are incorporated into Splotch for parallel rendering. Results are shown for 400 million particles of a GADGET simulation on a single GPU. Scalability is analyzed for up to 220 million particles.

A Paraview visualization of a one billion particle simulation is presented in [5], as well as a test analysis on a simulation using $256^3$ particles. Statistical sampling techniques to reduce the size of large particle datasets for in-situ analysis and visualization are discussed in [6]. This implementation was tested in Paraview with a $2048^3$ particle run. Another example of Paraview used for visualization of cosmological data is presented in [7], with two million SPH particles and two million dark matter particles per time-step from a GADGET simulation. The particle data is interpolated into a $128^3$ element regular grid and imported into Paraview for visualization as isosurfaces.

As previously mentioned, the preliminary results presented in this poster scale up to full datasets of $3200^3$ particles. To the best of our knowledge, there are no other approaches in the literature reporting scalability to datasets of that magnitude.

## III. Implementation

This work has been implemented as part of vl3, a parallel GPU-based framework for large-scale visualization developed at Argonne National Laboratory and the Computation Institute of the University of Chicago. Its parallel rendering and sort-last compositing architecture are described in [8].

Point sprites is a well-known technique to visualize particle systems. Our implementation was inspired by the N-body simulation demo [9], part of the Nvidia CUDA GPU Computing SDK. As part of the vl3 framework, our code reads the particle data from a parallel filesystem, with every MPI rank taking care of all particles in its assigned subregion of the simulation space. Vertex Buffer Objects (VBOs) are allocated on GPU memory and the particle information is copied into them. In every rendering cycle, the parallel rendering processes receive the position and orientation of the camera and render their particle data as point sprites, generating the resulting image in a Frame Buffer Object (FBO), whose contents are then read back and sent over the network to the corresponding compositing nodes. The vl3 classes to support GLSL shaders are used to implement fragment and vertex shaders that modify the size of the sprites according to their distance to the camera and apply a Gaussian texture to them. Alpha blending is enabled and set to accumulate color values on the color buffer.

OpenGL can assign individual RGB color values to each particle, allowing us to explore velocity fields, or coloring particles by their individual ID numbers to track their evolution. It is possible to encode additional information in the fourth coordinate component, such as point size, that can be taken into account by the vertex shader to draw particles of different sizes. In addition, shaders can apply different textures to point sprites. For example, particles belonging to halos can use a distinctive texture and color to help identifying them. Finally, multiple time steps can be loaded into GPU memory for interactive spatio-temporal exploration of the dataset.

## IV. Evaluation

Our experimental testbed consists of Tukey, a GPU cluster at Argonne National Laboratory built with multi-core X86 computers. There are 96 compute nodes with 64 GB RAM and two Nvidia Tesla M2070 GPUs per node. The nodes are connected via a QDR Infiniband interconnect and the MPI implementation is Mvapich2 from Ohio State University (OSU).

**Image quality**: Examples of the resulting image quality are shown in the poster. In contrast with other methods, such as ray tracing volume rendering, images obtained with point sprite rendering allow to clearly distinguish the mass density field, as well as visualize individual particles in zoom-in regions.

**Weak scaling**: The performance of our parallel implementation is evaluated as the number of GPUs is increased, while preserving the average number of particles rendered per GPU. In such a way, we scale from 2 GPUs rendering about 500 million particles to 128 GPUs rendering a full HACC dataset consisting of $3200^3$ particles (about 32 billion particles.) A camera view was selected so that all particles contribute to the final image. In doing so, the rendering load should be similar for all GPUs. Compositing network communication dominates the total time as the number of GPUs increases, confirming a fact already pointed out in [5] and previous experiments in [8]. However, in this poster we focus on the rendering time for point sprites, demonstrating that this rendering component is highly parallelizable and scales nicely as we keep adding more particles until reaching the full size of the dataset at 128 GPUs.

**Strong scaling**: In this experiment we use another $1024^3$ particle dataset from a HACC simulation. Fitting the entire particle dataset in GPU memory requires at least eight GPUs, therefore our tests scale from 8 GPUs (rendering about 130 million particles each) to 128 GPUs (about 8 million particles each.) The problem size is preserved while the number of GPUs is increased, therefore the full $1024^3$ particle dataset is rendered in all cases. As in the previous case, a proper camera view was selected so that the entire dataset is visible and all particles are rendered.

## V. Conclusion and Future Work

In this poster we propose parallel hardware-accelerated point sprite rendering as an alternative to other commonly used methods for rendering large-scale particle datasets. We have initially focused on rendering performance, scaling to a full dataset of $3200^3$ particles. The current implementation has shown excellent weak scalability on up to 128 GPUs in terms of its rendering time. It has also shown good strong scalability and rendering performance.

Future work will concentrate on interactivity and scaling to even larger particle datasets. A combination of Level-Of-Detail (LOD), out-of-core, and other methods will likely be needed to scale to the largest datasets, currently consisting of $10240^3$ particles. A sort-first strategy to mitigate compositing communication time along with load balancing schemes will also be investigated.

## Acknowledgment

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357 including the Scientific Discovery through Advanced Computing (SciDAC) Institute for Scalable Data Management, Analysis, and Visualization. This research has been funded in part and used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. Salman Habib, Katrin Heitmann, and the HACC team at Argonne National Laboratory are gratefully acknowledged for providing datasets and the Generic IO (GIO) library used in this work.

## References

[1] P. Goswami, F. Erol, R. Mukhi, R. Pajarola, and E. Gobbetti, "An efficient multi-resolution framework for high quality interactive rendering of massive point clouds using multi-way kd-trees," *The Visual Computer*, vol. 29, no. 1, pp. 69–83, 2013.

[2] R. Fraedrich, J. Schneider, and R. Westermann, "Exploring the millennium run-scalable rendering of large-scale cosmological datasets," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 6, pp. 1251–1258, 2009.

[3] C. P. Gribble, A. J. Stephens, J. E. Guilkey, and S. G. Parker, "Visualizing particle-based simulation datasets on the desktop," in *British HCI 2006 Workshop on Combining Visualization and Interaction to Facilitate Scientific Exploration and Discovery*, 2006, pp. 1–8.

[4] M. Rivi, C. Gheller, T. Dykes, M. Krokos, and K. Dolag, "Gpu accelerated particle visualization with splotch," *Astronomy and Computing*, vol. 5, no. 0, pp. 9 – 18, 2014.

[5] J. Woodring, K. Heitmann, J. Ahrens, P. Fasel, C.-H. Hsu, S. Habib, and A. Pope, "Analyzing and visualizing cosmological simulations with paraview," *The Astrophysical Journal Supplement Series*, vol. 195, no. 1, p. 11, 2011.

[6] J. Woodring, J. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann, "In-situ sampling of a large-scale particle simulation for interactive visualization and analysis," in *Computer Graphics Forum*, vol. 30, no. 3. Wiley Online Library, 2011, pp. 1151–1160.

[7] P. A. Navrátil, J. L. Johnson, and V. Bromm, "Visualization of cosmological particle-based datasets," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 6, pp. 1712–1718, 2007.

[8] S. Rizzi, M. Hereld, J. Insley, M. E. Papka, T. Uram, and V. Vishwanath, "Performance modeling of vl3 volume rendering on gpu-based clusters," in *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2014, pp. 65–72.

[9] L. Nyland, M. Harris, and J. Prins, "Fast n-body simulation with cuda," *GPU gems*, vol. 3, no. 1, pp. 677–696, 2007.