Visualizing the Supernova Explosion of a 25-Solar-Mass Star and the Simultaneous Birth of a Neutron-Star

Joseph A. Insley[a,b], Nestor Alvarez[a,b], Hal Brynteson[a,b], Adam Burrows[c], Ina Murphy[a,b]

[a]*Argonne National Laboratory, USA*
[b]*Northern Illinois University, USA*
[c]*Princeton University, USA*

Abstract

This explanatory visualization shows the results of a state-of-the-art 3D simulation of supernova explosion and neutron-star birth. It is a rare instance where the full stellar evolution of an object, including the physics of the convection and the radiation, has been simulated in three dimensions. Among the highlights is the deep core that is shrinking after explosion due to neutrino cooling and deleptonization on its way to becoming a cold, compact neutron star. There is also evidence of inner proto-neutron star convection, perhaps the site of magnetic dynamo action that can turn a pulsar into a magnetar. An exterior view shows the blast wave, which cocoons the newly-birthed neutron star, moving at ~10,000 km/s. Additionally, a reusable pipeline was developed, which leverages state-of-the-art tools for scientific data analysis and visualization resulting in high-quality renderings.

Keywords:
scientific visualization, high performance computing

1. Introduction

Core-collapse supernovae dramatically announce the death of massive stars and the birth of neutron stars. During this violent process, a combination of high- density nuclear physics, multi-dimensional hydrodynamics, radiation transport, and neutrino physics determines whether and how the star explodes. Typically stars are evolved on computers using spherical symmetry and convection is approximated by so called mixing link theory. However, to fully capture the complexity of the behavior within these stars all the physics of the convection and the radiation with complicated equation of state are required to allow it to evolve in the full three dimensions. The multi-group, multi-dimensional radiation hydrodynamics (RHD) code Fornax has been used to accomplish this.

The specific simulation visualized here tracks the early evolution, just under a second after the core collapse, of supernova explosion and neutron-star birth of a 25-solar-mass star [1, 2]. It was calculated at the Argonne Leadership Computing Facility (ALCF) using Theta, an 11.7 petflop Intel® Xeon Phi™ processor-based supercomputer. It operated on a grid consisting of ~34 million grid points, and consumed ~60 million node hours. Each of the 885 time steps simulated generated ~1.7GB of data, for a total of ~1.6TB for the full run.

The addition of tracer particles, generated in the visualization and analysis process, helps to illustrate the behavior and nested structure of the evolving star. Figure 1 shows the path of these particles in the initial stages, demonstrating how material is falling in. A short time later, as seen in Figure 2, there is a radiative region at the central core, where the blue particles rise and then fall, never leaving this interior region. This is the neutron star at birth. This star will shrink and cool, and be left behind. There are at least 100 million such neutron stars in the galaxy. This is bounded by a convective layer, where the particle behavior is more erratic, due to the violent turbulence in this region. In figure 3 we see the exterior shock wave as it continues to expand. Interior to that are surfaces and tracers which map the composition of the material, which is different in different directions. The result is a very dynamic explosion, which is aspherical and asymmetrical. While the explosion is ejecting material outward in one direction, accretion of material is falling back in in another direction.

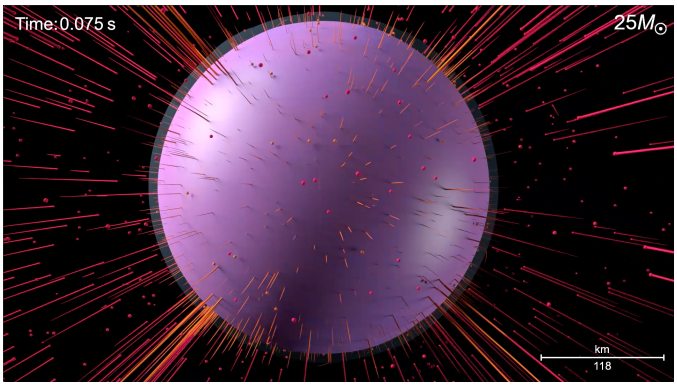

Figure 1: Particle path lines illustrate that matter is initially falling inward.

2. Visualization

One goal of this project was to produce an explanatory visualization for communicating the complex processes taking

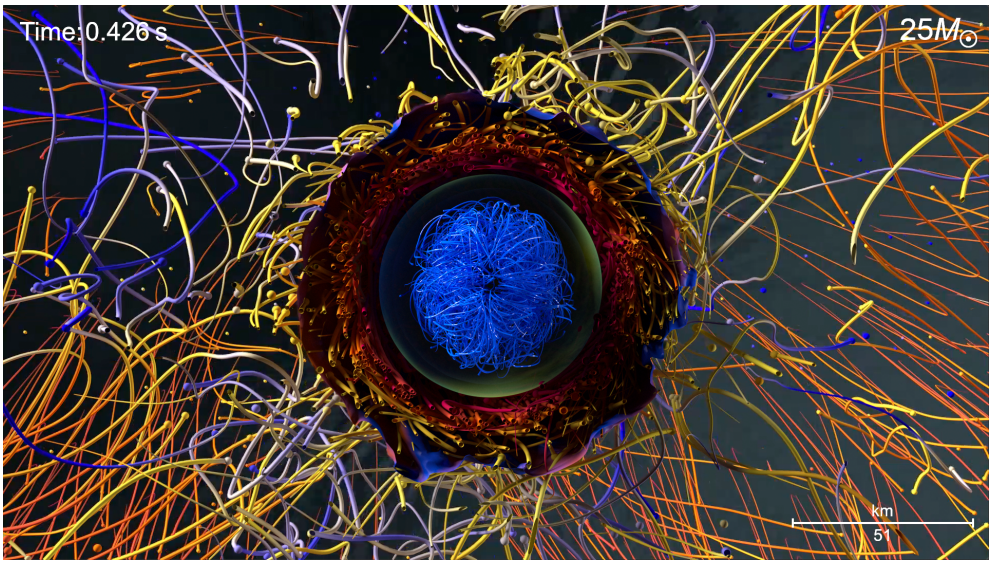

Figure 2: Blue particles raise and fall in the central radiative region that is the neutron star at birth. Exterior to this is a convective region, where turbulence causes erratic behavior of the particles.

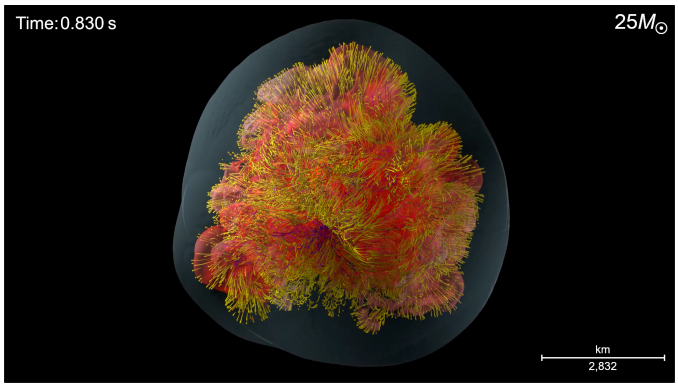

Figure 3: The outer surface shown here is the expanding shock wave. Interior to that are surfaces and tracers which map the composition of the material of the very dynamic explosion, which is aspherical and asymmetrical.

place during the early evolution of a neutron star after supernova explosion in a high-mass star. Additionally, we aimed to develop a reusable pipeline that leverages state-of-the-art tools for scientific data analysis and visualization resulting in high-quality renderings. Here we describe two major components of this workflow, which leverage Cooley, a visualization and analysis cluster at the ALCF.

2.1. Data Processing

The simulation data is first loaded into ParaView[3], where standard visualization algorithms can be applied, including calculation of isocontours of several different scalar fields. While there is much interest in plotting the pathlines of particles injected into the flow field of the simulation, these were not computed as a part of the original simulation. Thus we turn to ParaView for calculating these. However, a number of factors made this difficult to accomplish by simply applying ParaView's ParticleTracer filter.

Because the behaviour of the flow changes dramatically in different regions over time, it was important to continually inject new particles in several locations at every time step. However, tracing an increasing number of particles at each step eventually leads to unacceptable compute times at each step. This can be prevented by giving particles a time to live, i.e. limiting the number of steps that a particle is traced before being discarded. This restricts the number of active particles at any given time, keeping the compute time from growing uncontrollably.

Even after addressing this issue, another remains. In order for ParaView to trace all of the given particles at any step, it needs to know their current location. At any given time step, save for the first one, there will be particles that were injected at a previous step. Which means that it needs to go back to when those particles were injected to calculate the path they took to the current location. Consequently, ParaView needs to start at the first time step, and march through the whole time series in a single pass. Given the large number of particles, and time steps, long calculation times, and scheduling policies which limit the length of single job run times on Cooley, this is not possible.

To address this challenge we developed a series of custom ParaView scripts which can be run in batch mode across many nodes of the Cooley cluster in parallel. The first set of scripts injects particles at a given step N, and applies the ParticleTracer filter to trace their path over a fixed number of M steps. However, rather than telling ParaView about all of the time steps in the data set, the script makes a list of steps, starting at N and going through N + M. This way ParaView can start at step N, and does not need to go back to step 0 every time to trace forward. Multiple instances of this script are run across multiple nodes of Cooley in parallel. Eventually, one instance for each time step has been executed. Once completed, there will be M particle files for each time step of the simulation.

A second set of scripts is used to take the M particle files

for each time step and combine them into a single particle file for each step. Additional scripts are used to apply the TemporalParticlesToPathlines filter to generate pathlines for each of the particles. The vertices of the resulting pathlines are used to define splines, which smooth out the trails. A radius is also calculated based on particle age at each point along the trail, enabling the representation of the trail to be tapered, from larger at the head of the trail to smaller at the tail.

2.2. Advanced Rendering

Another part of the pipeline is used to improve the rendering quality of the individual frames of the animation. The integration of OSPRay[4], a ray tracing-based rendering engine for high-performance, high-fidelity visualization optimized for Intel Architecture CPUs, as a backend renderer for ParaView has greatly improved the realism possible with ParaView. At the same time, exposure of controls for features newly available in OSPRay is often lagging within ParaView .

Thus, after applying visualization algorithm filters in ParaView, instead of rendering the geometry that is produced directly in ParaView it is saved out to disk. A small number of these geometry files are then transferred to a local workstation where they can be loaded into Autodesk Maya [5], a 3D modeling and animation tool often used in the film industry. V-Ray[6] is an advanced 3D renderer, which leverages OSPRay under the covers. A V-Ray plugin for Maya provides a simple interface for applying V-Ray's advanced materials and lighting. Once applied, these settings are saved in a *vrscene* file. This vrscene file is then transferred back to Cooley, where it can be rendered in batch mode. An advantage of using the vrscene format is that the geometry data is stored in the vrscene as an external link to the data on disk, rather than encoded and stored directly in the vrscene. This allows us to move just a small number of files to the local workstation to set up the scene, and then swap out the linked file for different time steps when doing the final rendering on Cooley.

We use 40 nodes of Cooley to render different segments of the animation data in parallel. A combination of post-processing tools, including ImageMagick, ffmpeg, and the Adobe Creative Suite of tools, was used to combine the frames and add annotations, to produce the final animation.

2.3. Reusing the Pipeline

As previously stated, an important objective of this effort was to establish a pipeline that could be easily repeated. While the visualization presented here is for one specific simulation, a 25-solar-mass star, the research team will conduct a full suite of 3D simulations. As a byproduct of this investigation, libraries of supernova simulation data will be generated. Although a number of modifications will be required, the scripts and workflows developed here will serve as a recipe for generating similar visualizations for future simulations. For example, stars of differing masses may explode at different times, or expand faster or slower, or not at all. Placement for seed particles used in the path tracing will need to be adjusted accordingly. Similarly, camera placement and paths will need to be tailored to specific star behavior.

Acknowledgment

The authors wish to thank the ALCF Visualization and Analysis Team: Janet Knowles, Victor Mateevitsi, and Silvio Rizzi, along with Michael E. Papka, for their invaluable contributions. An award of computer time was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This simulation and visualization were produced using resources of the ALCF at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

References

[1] A. Burrows, D. Radice, D. Vartanyan, H. Nagakura, M. A. Skinner, J. C. Dolence, The overarching framework of core-collapse supernova explosions as revealed by 3D fornax simulations, Monthly Notices of the Royal Astronomical Society 491 (2) (2019) 2715–2735. arXiv:https://academic.oup.com/mnras/article-pdf/491/2/2715/31221715/stz3223.pdf, doi:10.1093/mnras/stz3223.
URL https://doi.org/10.1093/mnras/stz3223

[2] A. Burrows, D. Vartanyan, Core-collapse supernova explosion theory, Nature 589 (7840) (2021) 29–39.

[3] J. Ahrens, B. Geveci, C. Law, Paraview: An end user tool for large data visualization, in: C. Hansen, C. Johnson (Eds.), The Visualization Handbook, Morgan Kaufmann, 2005.

[4] I. Wald, G. Johnson, J. Amstutz, C. Brownlee, A. Knoll, J. Jeffers, J. Gunther, P. Navratil, Ospray - a cpu ray tracing framework for scientific visualization, IEEE Transactions on Visualization and Computer Graphics 23 (1) (2017) 931–940. doi:10.1109/TVCG.2016.2599041.
URL https://doi.org/10.1109/TVCG.2016.2599041

[5] Maya overview. (2022).
URL https://www.autodesk.com/products/maya/overview

[6] V-ray collection. (2022).
URL https://www.chaosgroup.com/vray/collection