DOE Office of Science INCITE Project: *Extreme-scale Simulation of Supernovae and Magnetars from Realistic Progenitors* 2019 Q1 Report

Principal Investigator:
Sean M. Couch
Michigan State University

Co-Investigators:
Andrew Christlieb (Michigan State University)
Evan O'Connor (Stockholm University)
Kuo-Chuan Pan (National Tsinghua University)
Luke Roberts (Michigan State University)
MacKenzie Warren (Michigan State University)

April 1, 2019

1 Project Usage

In 2019 Q1 we expended 3.2M core-hours on Mira out of our total 2019 allocation of 150M core-hours. This is substantially behind the linear usage curve. During Q1 we have focussed on getting new project personnel up to speed on the project, code tuning, and implementation of new capabilities in our core-collapse supernova (CCSN) application (see below). We are now running one of our primary simulation milestones in the Capability queue and will start a second Capability-scale simulation in the next week or two. Unlike in years past, when our simulations started small and grew to the Capability queue, we are starting 2019 already at that scale. Based on past usage experience, we do not anticipate any issues returning to the "ideal" usage rate.

2 Report on Project Milestones

Our milestones for Year 2, and corresponding progress, were:

1. Long time simulations of MHD CCSNe - These simulations have been restarted from simulations carried over from 2018 and are running in the Capability queue.

2. High-resolution simulation of MHD dynamos in the proto-neutron star - So far in Q1 we have analyzed simulations from 2018 that will serves as the initial conditions for this high-resolution simulation. We anticipate starting this simulation in the next week or two. Given the extreme resolution of this simulation, it will run in the Capability queue from the outset.

3. MHD simulation of CCSN progenitors - These simulations will be started in Q2. We are tuning our progenitor application to make better use of OpenMP threading.

4. CCSN simulation with 3D progenitors - The progenitor model for these simulations is now finished and ready to be used in CCSN simulations. We plan to run these simulations on Theta starting in Q2 or Q3.

5. Implement microphysics from TEAMS SciDAC collaboration and neutrino-electron scattering (NES) - the TEAMS microphysics package is not yet ready for production simulations. We have during Q1 finished an implementation of NES and are now testing it in 1D and 2D simulations. We anticipate being able to use this new capability in our planned 2019 CCSN simulations.

3 Project Productivity

3.1 Primary

Publications

- "Simulating Turbulence-aided Neutrino-driven Core-collapse Supernova Explosions in One Dimension", Couch, S. M., Warren, M. L., O'Connor, E. P. 2019, *arXiv e-prints*, arXiv:1902.01340

- "Features of Accretion Phase Gravitational Wave Emission from Two-dimensional Rotating Core-Collapse Supernovae", Pajkos, M. A., Couch, S. M., Pan, K., O'Connor, E. P. 2019, *arXiv e-prints*, arXiv:1901.09055

Presentations

- "Gravitational Waves from Core-collapse Supernovae," S.M. Couch, LIGO SN Group Seminar, March 2018

4 Center Feedback

Our catalyst, Adrian Pope, has been extremely helpful. He is now helping us tune our code for Theta.

5 Code Description and Characterization

`FLASH` is a highly capable, fully modular, extensible, community code that is widely used in astrophysics, cosmology, fluid dynamics, and plasma physics, and other fields. The capabilities of the FLASH code include adaptive mesh refinement (AMR), several self-gravity solvers, an advection-diffusion-reaction (ADR) flame model, an accurate and detailed treatment of nuclear burning, and a sophisticated two-moment neutrino transport scheme based on an explicit hyperbolic solver. The neutrino interactions are included through the open-source neutrino interaction library `NuLib`. We have enhanced the performance of the two-moment neutrino transport scheme significantly as well as upgraded the transport to now include full velocity and gravitational red-shift dependence in the evolution equations.

`FLASH` is written in modern Fortran, with some utility functions written in C, and a build system written in Python. It requires MPI library support, and either HDF5 or P-NetCDF for I/O. Additional mathematical software, such as `Hypre`, may be required to configure `FLASH` for particular simulations.

Algorithm classes used within `FLASH` include Sparse Linear Algebra solvers, FFT, active and passive particles, structured grids, and AMR.