Motivation

- Astrophysics researchers need to simulate movement and evolution of stars and galaxies.
  - Because they can't go out and play with real stars and galaxies.
- We need tools to make these simulations because coding a simulation is complex requiring extensive physics and programming knowledge to produce efficient code.
- Our project makes state-of-the-art astrophysics simulations accessible.
  - The end goal of the physicist is not to write simulations, it's to use them.
- In these simulations, every star can affect all of the others.
  - With as many as a few million bodies (galaxies, gas clouds etc...), computation times grow quickly.
  - Real research requires code that runs on powerful hardware to get results in reasonable time (on the order of a week sometimes).
- Complex software (AMUSE) exists to perform these computations efficiently.
  - Combines hardware-specific solutions to problems with a variety of physical problem domains, at varying degrees of speed/accuracy.

Target Audiences

Physics Student

- Minimal to no programming experience, minimal knowledge of astronomy.
- Our software will help them learn by performing simple experiments and observing results.

Observational Astrophysicists

- Not much programming experience, Good understanding of astronomy
- Our software will enable them to reproduce and analyze observed stellar phenomena

Theoretical Astrophysicists

- Significant programming experience, Good understanding of astronomy
- Theoretical astrophysicists may need to make many small parameter changes to long running experiments.
- Our software lets them make these customizations and update values without rewriting code.

Overview

Astrophysical Multipurpose Software Environment (AMUSE)

- Here is AMUSE's architecture setup.
- AMUSE uses a library called MPI to gather physics code written in many languages under one python interface.
  - Codes include gravity and stellar evolution to name a few.
- Also includes useful things like unit conversions and methods to manipulate large groups of stars.
- Our software provides a framework that builds on AMUSE to generate and run experiments.

State of AMUSE

- Partnership between Drexel and the Leiden Observatory in the Netherlands, sponsored by NOVA.
- NOVA = Netherlands Research School for Astronomy
- Mention large scale again
- Written by hand = hard to share
- Waste of work to replicate someone else's diagnostics to fit your exact circumstances.
- Code to the right is FORTRAN from AMUSE's community codebase.

Purpose of GPUnit

- Ease the creation, execution, and analysis of experiments with AMUSE
- Create experiments with minimal to no programming
- Repeatability
- Sharing Experiments
- API for results / diagnostics

Features

- Explain how features satisfy requirements.
- Configurable experiments -> less programming.
- Diagnostics -> common API for metrics
- Code is generated to run actual experiment -> advanced users can tweak it
- Storage of state -> repeat experiment if it crashes

Architecture

- The interface lets the user put the experiment together.
- The experiment generator lets advanced users customize details.
  - They can make small changes to how the experiment runs.
  - They can also add completely new code that does something our framework wouldn't normally do.
- The network layer gives the user a view of how the cluster is being used.
  - Uses multicast messages to discover nodes, tracks each node individually.
- We provide a storage API to save and share experiments.
- Each run of the experiment is saved separately, including
  - star positions and masses
  - diagnostic and logging output
  - settings for the components of that run of the experiment.
- All of this is built on top of AMUSE's features such as units and simulation modules.

Design

- Previous physics simulations were one-off scripts written for a specific problem, on specific hardware.
- AMUSE is the only package to provide a uniform interface to a variety of tools on a variety of hardware.
- We settled on Python because AMUSE is a Python library, interaction is streamlined.
- If we had used C++, AMUSE would run in a separate process, introduces unnecessary disconnect between our code and AMUSE.
- Challenges:
  - Figuring out how AMUSE works.
  - Making a useful tool that simplified experiment creation without taking away any of AMUSE's power/features.
  - Allow future developers to expand on this work:
    - Modular diagnostics w/API to do the work, experiment storage abstraction: allows for remote backup.

Tests

- Table of tests that pass.

Project Plan

- AMUSE codebase is large and complex (as we have mentioned)
- Before we could plan our project we needed to figure out how AMUSE worked.
- Learning continued throughout the project.

Team Management

- Bi-weekly team meetings helped get a lot of work done
- Able to code and discuss at the same time in person (useful)

Project Impact

- Researchers can discover important things much faster when they don't have to fuss with experiment boilerplate.
- Students can learn about what astrophysicists really do first-hand without going too deep into complicated issues.
- Our design is extensible and leaves room for more advanced features.
- New features can be added by anyone by writing code that follows the APIs in our design.

Demo

- Demonstration of a simulation.

Questions

- Questions?