

GPUUnit

Daniel Bagnell
Jason Economou
Rajkumar Jayachandran
Tim McJilton
Gabriel Schwartz
Andrew Sherman

Advisor: Prof. Jeremy Johnson
Stakeholders: Prof. Steve McMillan
Alfred Whitehead
The Leiden Observatory

Overview

Introduction

Purpose

- Purpose of GPUTest
- Target Audiences

Features and Design

User Interface

- Experiment Editor
- Cluster Control

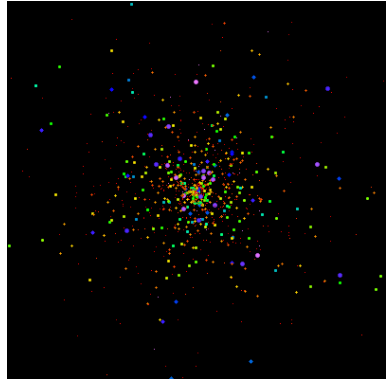
Testing

Impact

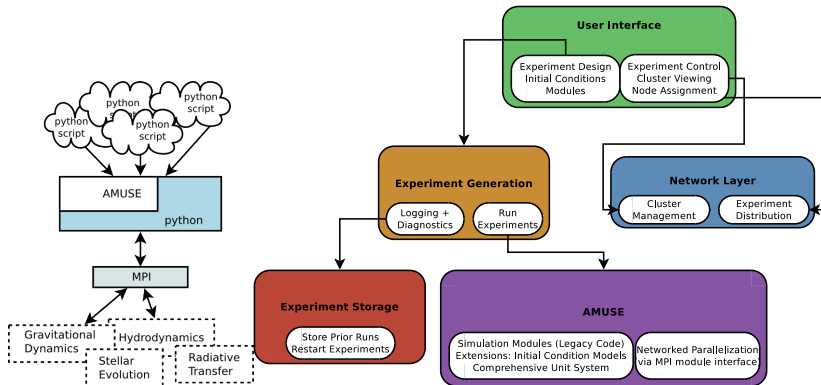
Demo

Motivation

- ▶ Astrophysics researchers need to simulate movement and evolution of star clusters and galaxies.
- ▶ Every star pulls on all of the others: $O(n^2)$ for the simplest algorithm.
- ▶ Stars evolve over time, mass and size changes.



Astrophysical Multipurpose Software Environment (AMUSE)



State of AMUSE

- ▶ Currently used by researchers to run large-scale simulations.
- ▶ Scripts, diagnostics, logging are all written by hand.
- ▶ AMUSE API/programming knowledge is required to create experiments.

```
modules, particles, convert_nbody = initialization(experiment)

#Create channels Between Particles and Modules
channels_to_module = []
channels_from_module = []

for module in modules:
    #channels_from_module.append(module.particles.new_channel_to(particles))
    #channels_to_module.append(particles.new_channel_to(module.particles))
    module.particles.copy_values_of_state_attributes_to(particles)

for diagnostic in experiment.diagnostics:
    diagnostic.convert_nbody = convert_nbody

while time <= tmax:
    #Evolve Modules
    for module in modules:
        module.evolve_model(time)

    if len(channels_to_module)-1: #Currently disabled
        #Synchronize Particles Across Channels
        for channel in channels_from_module:
            channel.copy()
        for channel in channels_to_module:
            channel.copy()

    for module in modules:
        module.particles.copy_values_of_state_attributes_to(particles)

    #Run Diagnostic Scripts
    for diagnostic in experiment.diagnostics:
        if diagnostic.should_update(time, modules):
```

Purpose of GPUnit

- ▶ Ease the use of AMUSE
- ▶ Create/Design/Modify experiments
- ▶ Select, configure, swap out modules and initial conditions
- ▶ Store and restore progress of running experiments.

Target Audiences

- ▶ Physics Students
- ▶ Observational Astrophysicists
- ▶ Theoretical Astrophysicists

Features

- ▶ Configurable experiments that can be saved and shared.
- ▶ Diagnostic tools that compute useful metrics.
- ▶ Storage of experiment state in case of crashes.
- ▶ Custom diagnostics and code generation.
- ▶ Provides a display of cluster usage to aid in scheduling.

Design

- ▶ Written in Python using the PyQt4 GUI toolkit.
- ▶ AMUSE is written in Python, streamlines interaction.
- ▶ C++ was considered as it supports Qt as well.
 - ▶ Communication w/AMUSE would be cumbersome.
 - ▶ AMUSE would be in a separate process.
- ▶ Designed APIs for diagnostics, logging and experiment persistence.
 - ▶ Users can create new diagnostics easily.
 - ▶ Experiments can be stored in a file structure, a remote DB etc...



Experiment Editor

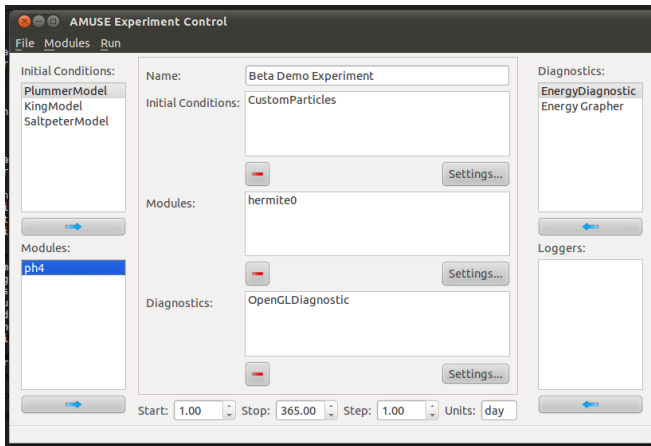


Figure: Experiment Editor

Cluster Control

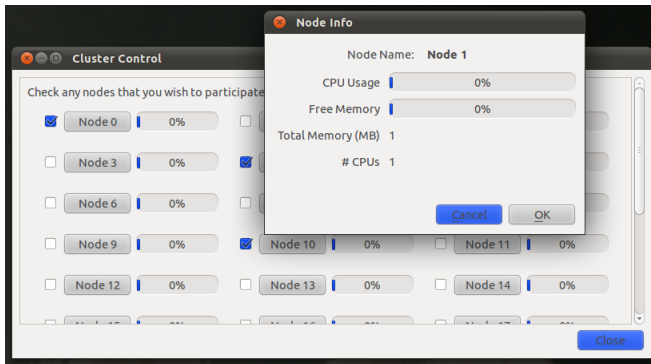


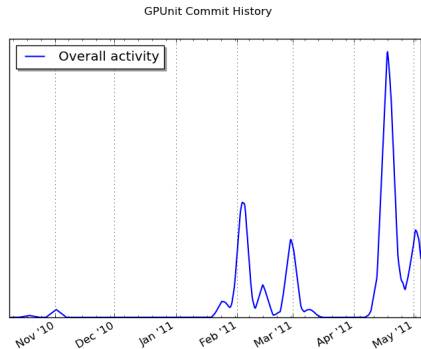
Figure: Cluster View

Tests

- ▶ Table of tests that pass.

Team Management

- ▶ Used Mercurial as our version control system.
 - ▶ Distributed, allows off-line commits.
- ▶ Team met weekly.
 - ▶ Once to plan work, once to code.
- ▶ Bi-weekly advisor meetings.



Potential Benefits

Demo

- ▶ Demonstration of a simulation.

Questions

- ▶ Questions?