GPUnit

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

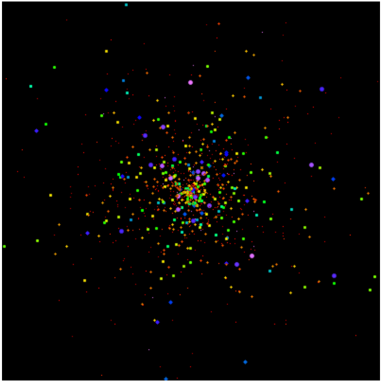
Advisor: Prof. Jeremy Johnson
Stakeholders: Prof. Steve McMillan
Alfred Whitehead
The Leiden Observatory

May 6, 2011

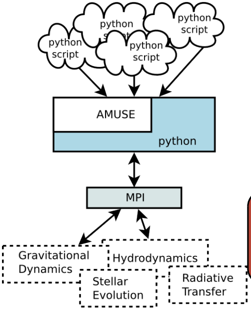
Overview

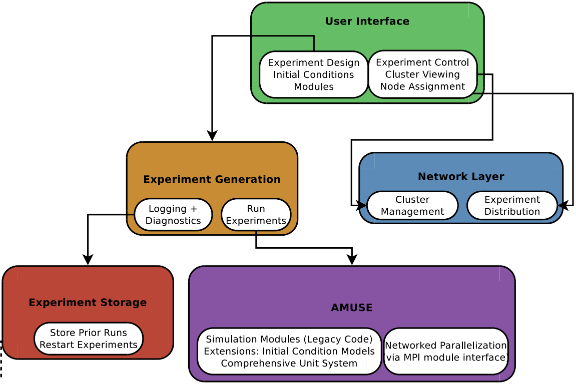
Motivation

- Astrophysics researchers need to simulate movement and evolution of star clusters and galaxies.
- Every star pulls on all of the others: $O\left(n^2\right)$ 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:
```

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

$$
\begin{aligned}
&\ddot{X} + \frac{C}{M}\dot{X} + \omega_n^2\Big[1 - \frac{f(\Phi)}{2}(\Delta k_1 + \Delta k_2 \cos 2\Phi)\Big]X \\
&\quad - \frac{\omega_n^2 f(\Phi)\Delta k_2 \sin 2\Phi}{2}\Big(Y_m - \frac{P}{K}\Big) \\
&\quad = \varepsilon(\Omega + \dot{\varphi})^2 \cos(\Phi + \delta) + \varepsilon\ddot{\varphi}\sin(\Phi + \delta), \\
&\ddot{Y}_m + \frac{C}{M}\dot{Y}_m - \frac{\omega_n^2 f(\Phi)\Delta k_2 \sin 2\Phi}{2}X \\
&\quad + \omega_n^2\Big[1 - \frac{f(\Phi)}{2}(\Delta k_1 - \Delta k_2 \cos 2\Phi)\Big]Y_m \\
&\quad = \varepsilon(\Omega + \dot{\varphi})^2 \sin(\Phi + \delta) - \varepsilon\ddot{\varphi}\cos(\Phi + \delta) \\
&\quad - \frac{Pf(\Phi)}{2M}(\Delta k_1 - \Delta k_2 \cos 2\Phi), \\
&\ddot{\theta} + \frac{K_t + K_c}{I_0}\theta - \frac{K_t}{I_0}\varphi = -\frac{C_t + C_c}{I_0}\dot{\theta} + \frac{C_t}{I_0}\dot{\varphi}, \\
&\ddot{\varphi} + \frac{C_t}{I}\dot{\varphi} - \frac{C_t}{I}\dot{\theta} + \frac{K_t}{I}\varphi - \frac{K_t}{I}\theta \\
&\quad = \frac{P\,\varepsilon f(\Phi)}{2I}(\Delta k_1 \cos(\Phi + \delta) - \Delta k_2 \cos(\Phi - \delta)) \\
&\quad + \frac{P^2}{2KI}\Big[\frac{1}{2}\frac{\partial f(\Phi)}{\partial \Phi}(\Delta k_1 - \Delta k_2 \cos 2\Phi) \\
&\qquad + f(\Phi)\Delta k_2 \sin 2\Phi\Big] + \Gamma_c,
\end{aligned}
$$

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

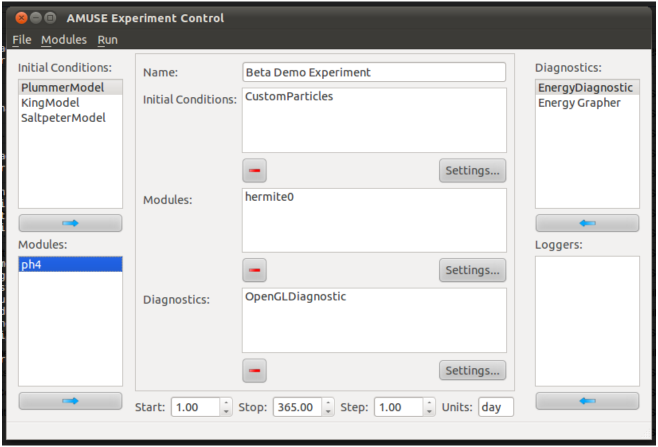


Figure: Experiment Editor

Cluster Control

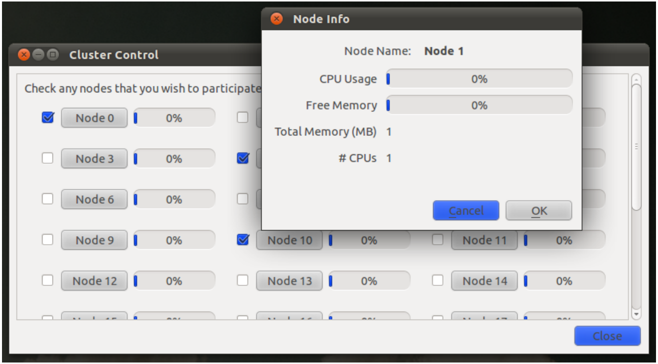


Figure: Cluster View

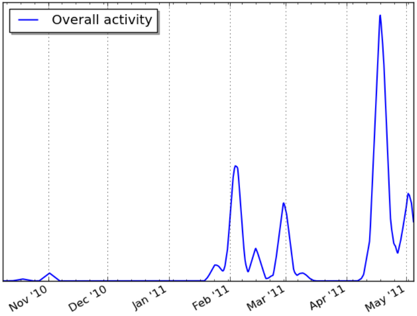
Tests

- 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.

GPUnit Commit History



Potential Benefits

Demo

- Demonstration of a simulation.

Questions

- Questions?