Design Overview

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

Advisor: Prof. Jeremy Johnson
Stakeholders: Prof. Steve McMillan
Alfred Whitehead

February 21, 2011

Components

- Interface
- Scripts
- Diagnostics + Logging
- Networking

Experiment

- Encapsulates everything required to reproduce or restart the simulation
- The abstract requirements of an experiment include:
  - Persistence (some storage mechanism like a DB or directory structure)
  - Logging/Diagnostics:
    - Which ones are used
    - Where/how is their output stored
  - Parameters:
    - Modules
    - Initial Conditions
    - Particles

Code Generation

- Experiments specify things like initial conditions, modules etc...
- These are used as input to generate scripts
- The scripts are generated using the code generator module (currently part of the interface).
- Different modules may require different code to run them, though this is not the case now.
- Code generation allows the end user (scientist) to tweak the generated experiment to suit their needs.

Interface

- Frontend for creating an experiment.
- Defines initial conditions etc...
- Specify where to run the experiment.
- Connects to network implementation using IPC
- Controls distribution of modules using MPI (mpirun, threaded module evolution)

Diagnostics + Logging

- Diagnostics and logging implemented using Python
- Diagnostics/logging modules can do anything the user needs so long as it fits the given interface
- Diagnostics have access to the entire system state.
- Logging has access to the experiment information:
  - Which timestep was run.
  - Time between timesteps.
  - Types of modules loaded.

Networking

- Tracks which experiments are running on which nodes
- Shows node usage and availability
- Discovery performed using multicast, no need to specify a list of nodes