Coven
a Framework for High Performance Problem Solving Environments

Nathan A. DeBardeleben
Walter B. Ligon III
Sourabh Pandit
Dan C. Stanzione Jr.

Parallel Architecture Research Laboratory
Clemson University

Problem Solving Environments in HPC

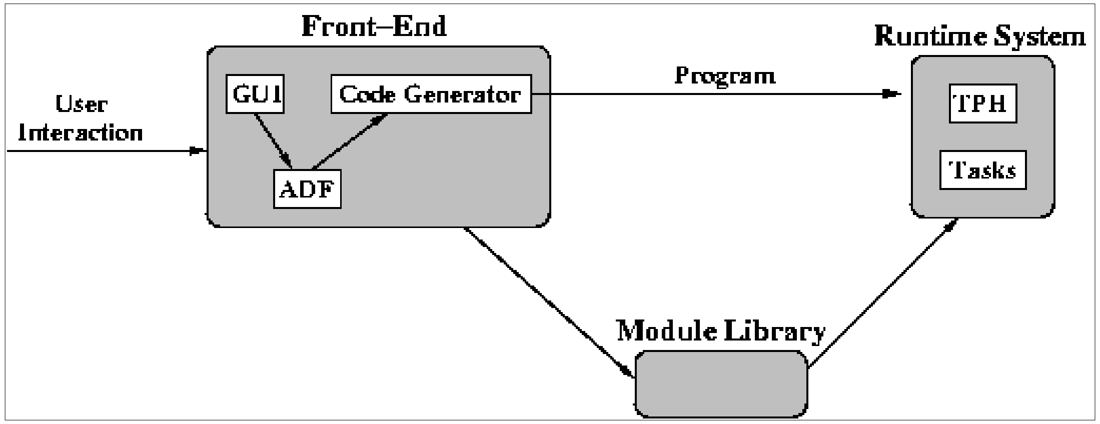
- PSEs are integral parts of modern HPC
- Help manage complexity of modern scientific computing
- Good PSE hides many details of the system, application, or both
- Good PSE flexible enough to solve the problem yet powerful enough to provide reasonably high performance

PSE Construction

- Some good examples of PSEs for HPCs, but specific to an application domain
- Little work has been done in creating a reusable framework for PSE construction
- Two important characteristics of a good PSE framework:
  - Flexibility
    - The ability to support a wide range of computational models that various domains may require
  - Abstraction
    - Carefully hide the details of both the underlying computer system and the problem domain, where appropriate

Coven

- Framework for building PSEs for parallel computers
- Three main components: runtime system, front-end, and module library



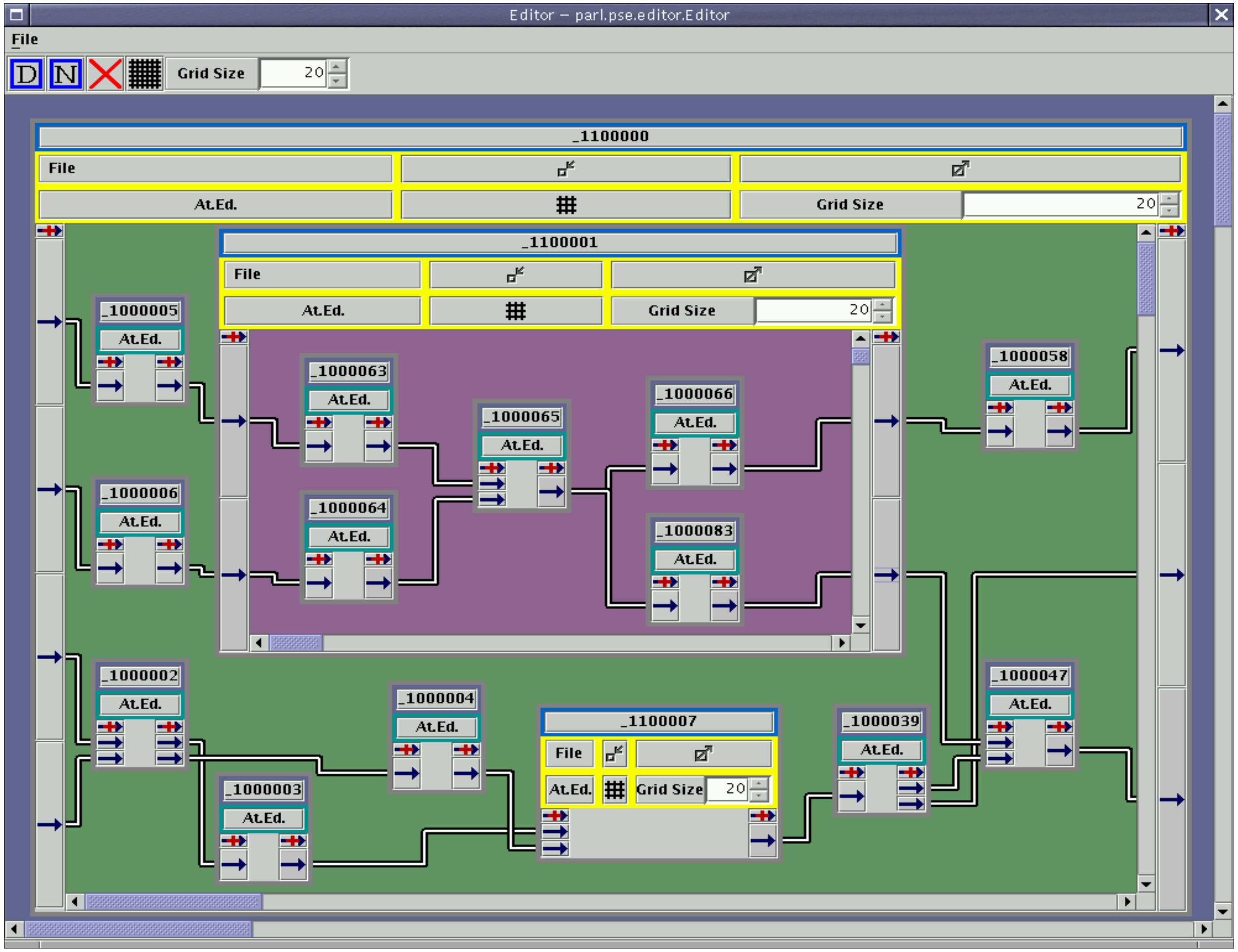
Runtime Driver

- Multi-threaded parallel runtime system
- Targets Beowulf clusters
- Uses a runtime generated data structure (TPH) to manage partitioning data sets among cluster nodes
- Executes applications capable of supporting most parallel programming constructs

Agent Based Front End

- Allows multiple custom interfaces to be constructed
- Stores information about the specification, implementation, and performance of the application in an attributed graph format
- Facilitates ways for agents to provide suitable abstractions for a particular class of user

GUI Screenshot



Module Library

- Collaborative repository
- Many pre-defined modules
- Users can add their own modules
- Holds both system and application modules
- Transfers modules from the front-end (GUI) machine to the back-end (parallel) machine

Tagged Partition Handle

- Internal data structure within Coven
- Hold buffers of data and provide means for creating, accessing, modifying, and destroying these buffers
- Handles all inputs to and outputs from modules
- Since TPHs pass from machine to machine, module programmers describe the data to access through a tag

Tagged Partition Handle

- Parallel task
  - Module
    - Piece of code which operates on some data
  - Tagged Partition Handle
    - Data structure containing related data
- Goal is to schedule execution of modules and TPHs to perform tasks in parallel
- Coven runtime driver provides means for this which allows overlapping of I/O, computation, and communication

Modules

- Reside in the module library on front-end machine
- Transferred to parallel computer upon execution
- Two classes of modules:
  - Application modules
  - System modules

Application Modules

- Written by an application designer
- Examples:
  - Compute resultant of vector multiplication
  - Compute partial force between two bodies
  - Calculate lat / long of a buffer of grid points
  - Update a temperature matrix based on values of neighboring cells

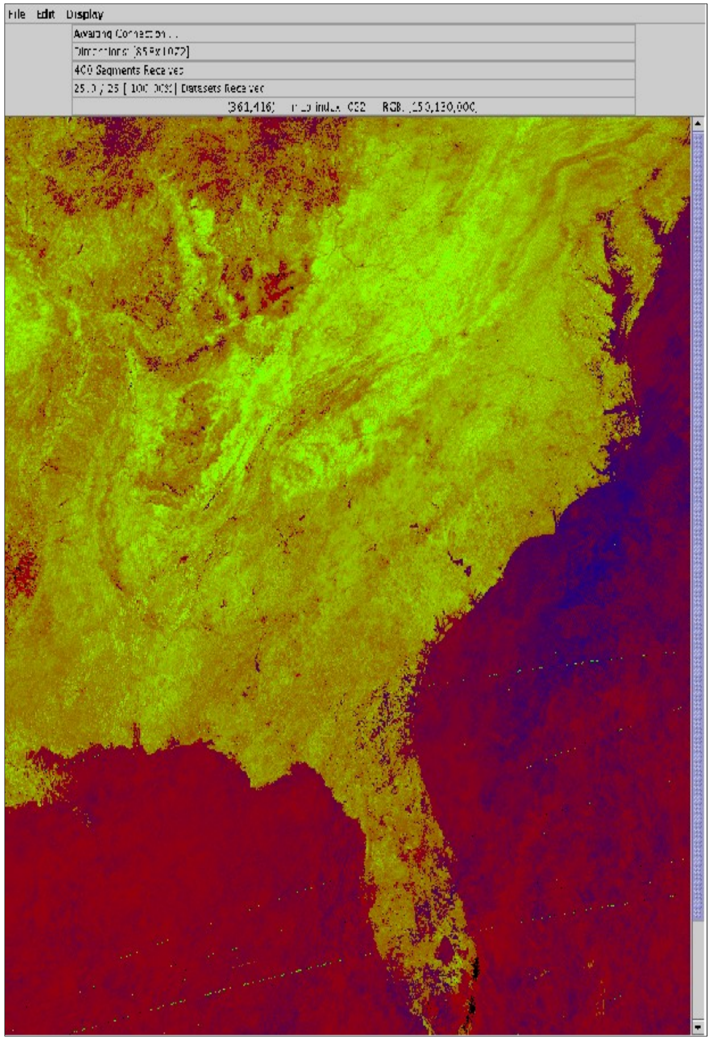
System Modules

- Written by someone familiar with parallel computing, load balancing, etc.
- Allows for steering of computation
- Examples:
  - Perform parallel communication such as shifting data to neighbor in a parallel stage (such as with MPI)
  - Partition data
  - Create TPHs
  - Consume TPHs

Prototype PSEs

- CERSe
  - Remotely sensed satellite data
  - Legacy NASA remote sensing code
- Medea
  - N-Body simulations
- Still in development
  - Molecular dynamics
  - CFD / Heat transfer

CERSe



Awaiting Connection . . .

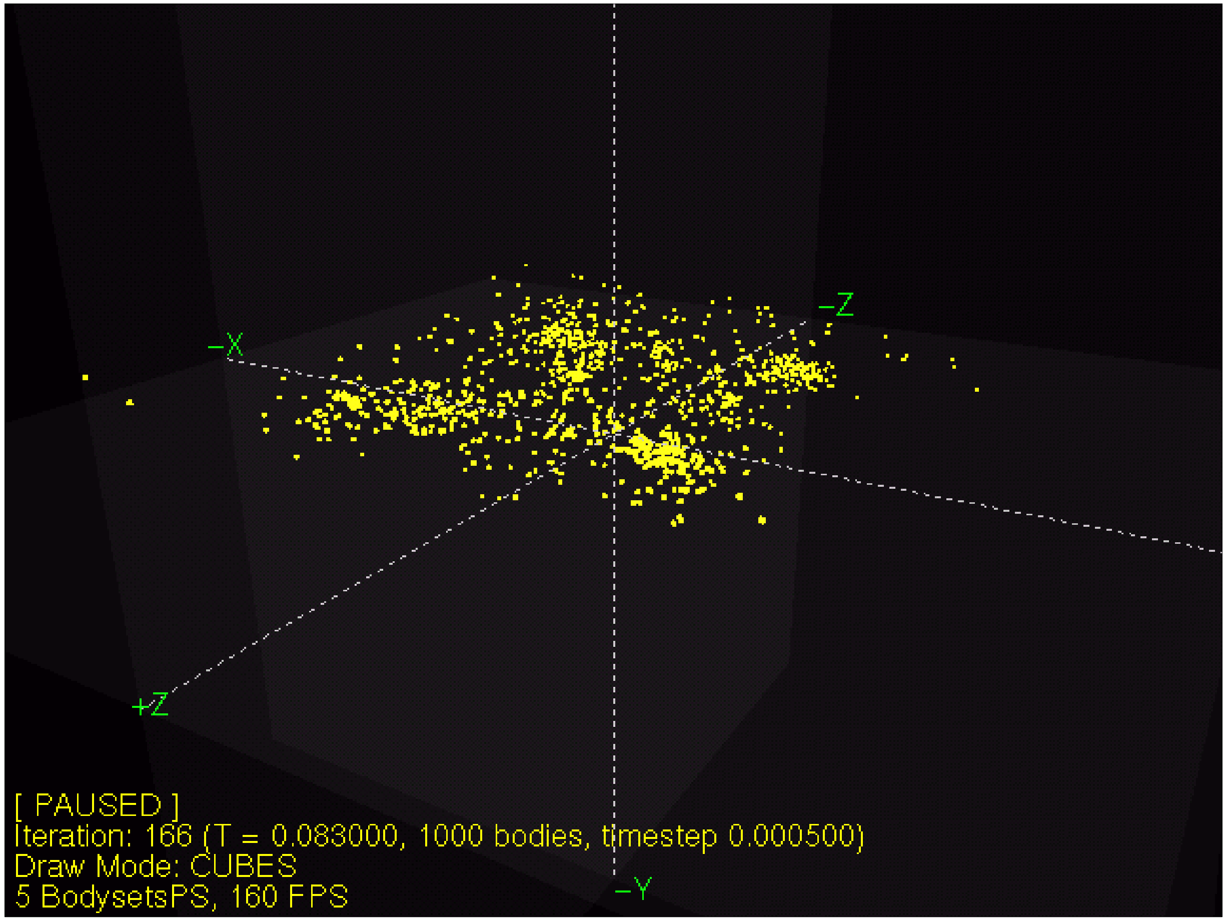
Dimensions: [859x1072]

400 Segments Received

25.0 / 25 [100.00%] Datasets Received

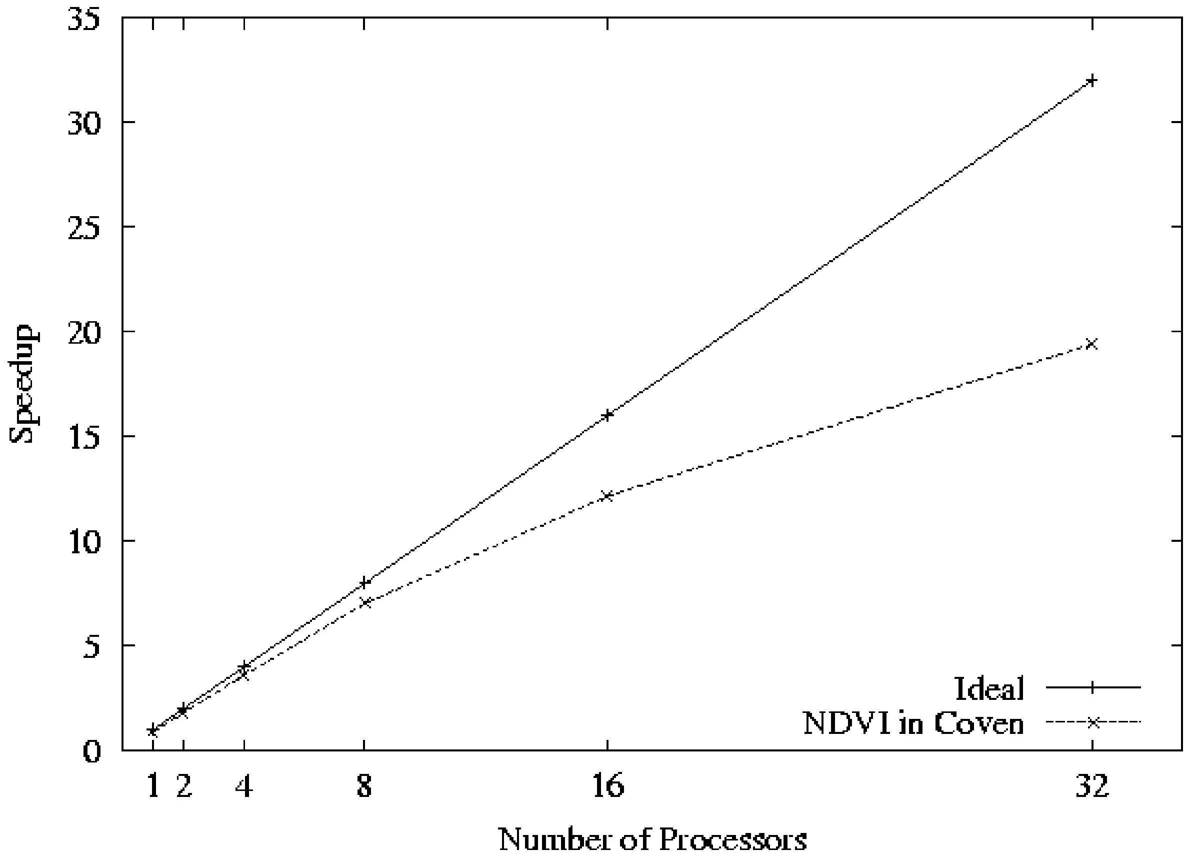
(361,416) [illegible] index 032 RGB (150,130,000)

Medea

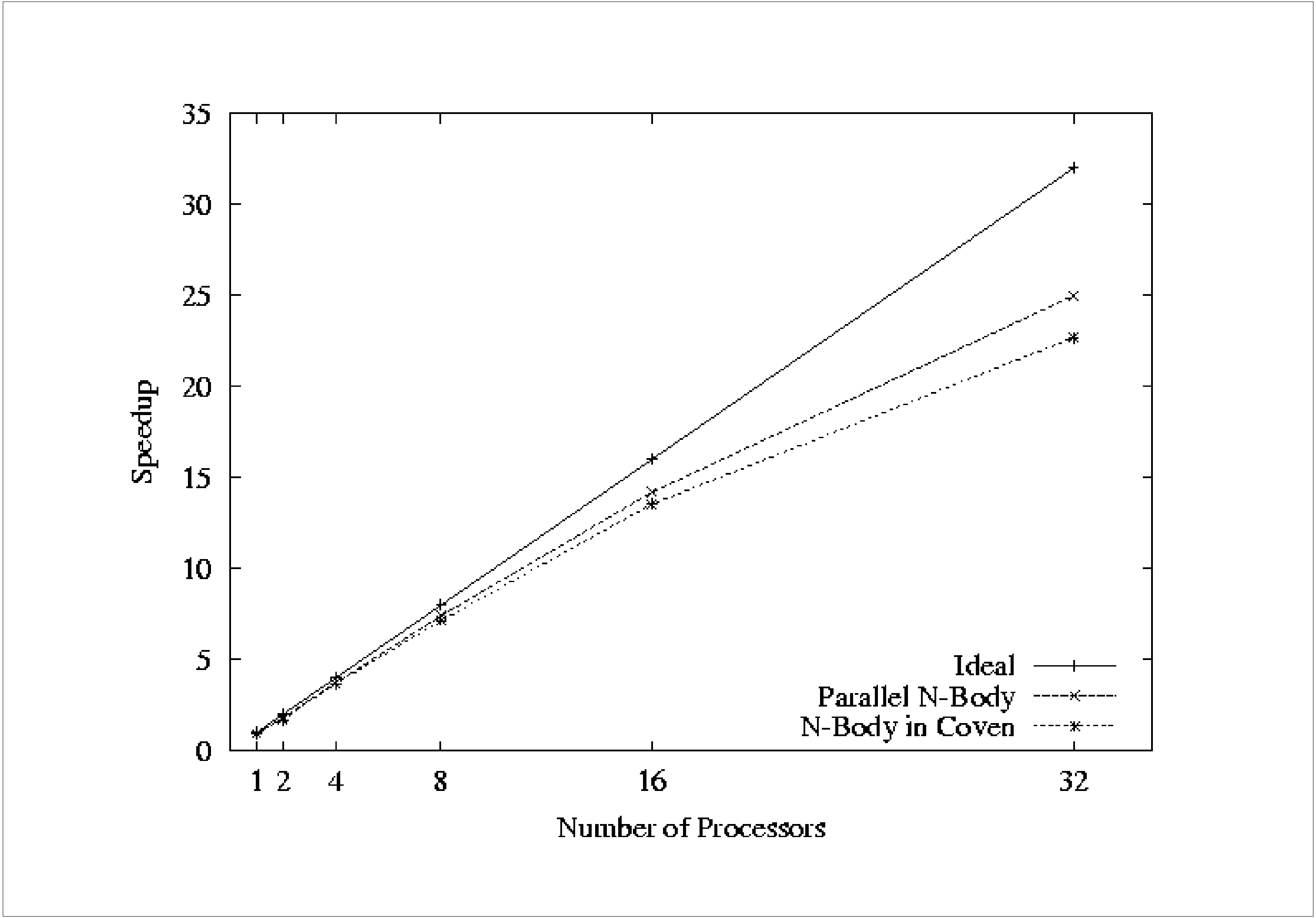


```
[ PAUSED ]
Iteration: 166 (T = 0.083000, 1000 bodies, timestep 0.000500)
Draw Mode: CUBES
5 BodysetsPS, 160 FPS
```

NDVI Performance



N-Body Performance



Conclusions and Future Work

- Presented a customizable framework for the creation of PSEs for HPCs
- Prototype PSEs have been demonstrated
- Applications built using these PSEs have achieved promising performance
- Coven can speed up the PSE construction process
- Create additional prototype PSEs to evaluate the flexibility of the framework
- Study performance tuning with the framework

Acknowledgements

- This work was supported in part by:
  - NASA grant NAG5-8605
  - ERC Program of the National Science Foundation under Award Number EEC-9731680
- The Parallel Architecture Research Laboratory
  - http://www.parl.clemson.edu

CLEMSON UNIVERSITY

PARL Parallel Architecture Research Laboratory

CLEMSON UNIVERSITY

PARL

Parallel Architecture Research Laboratory