Beowulf Analysis Symbolic INterface

A multi-user environment for parallel data analysis and visualization

Enrico Vesperini
Doug Jones
Dave Goldberg
Steve McMillan (PI)
and the BASIN team

Department of Physics
Drexel University

SciPy2007
August 16, 2007
Who we are...

Theoretical and observational astrophysicists working with large $(10^2-10^3 \text{ GB})$ datasets from simulations and observations:

- stellar dynamics, N-body simulations of dense stellar systems, globular star clusters;
- gravitational lensing, dark matter distribution
- observational extra-galactic astronomy, Sloan Digital Sky Survey-SDSS.
What we use:
Beowulf Clusters

- Beowulf clusters: attractive (low-cost) parallel systems
- Several packages exist for serial data analysis (e.g., in observational astronomy, IRAF, MIDAS).
- For parallel data analysis each group tends to (re-) develop its own set of tools:
  - Expertise in parallel tools/algorithms.
  - Large time investment.
What is it?

• Integrated suite of tools for parallel data analysis and visualization.

• Multi-user environment for interactive data analysis and visualization.
Why?

- Easy and transparent parallel data analysis.
- Avoid redundant development of functions commonly used in data analysis.
Where?
BASIN is freely (GPL) available at
http://www.physics.drexel.edu/BASIN
1) Start the computational engine on a remote parallel server.

2) Connect a local client to the computational engine.

3) Read file distribute data.

4) Parallel calculation of new attributes.

5) Visualize distributed data (using VisIt-LLNL).

6) Transfer data on the local machine and plot with one of the standard Python plotting packages (Matplotlib).
BASIN architecture

Beowulf

BASIN Parallel Data Analysis Engine
BASIN C++/Python kernel
IPython engines

BASIN data format/Viz data format translation

Graphics data interface

Parallel Visualization Engine

IPython1 Controller

Commands Data

Client 1 (BASIN and Visualization python interfaces)
BASIN Python GUI

Client 2 (BASIN and Visualization python interfaces)
BASIN Python GUI

Client 3 (BASIN and Visualization python interfaces)
BASIN Python GUI

Machine 1
Machine 2
Machine 3

BASIN Kernel
BASIN Client/GUI
IPython
Visualiz.
BASIN architecture

Beowulf

BASIN Parallel Data Analysis Engine
BASIN C++/Python kernel

IPython engines

Commands Data

IPython 1 Controller

Client 1
(BASIN and Visualization python interfaces)
BASIN Python GUI

Machine 1

Client 2
(BASIN and Visualization python interfaces)
BASIN Python GUI

Machine 2

Client 3
(BASIN and Visualization python interfaces)
BASIN Python GUI

Machine 3

BASIN data format/Viz data format translation

Parallel Visualization Engine

Graphics data interface

graphics command interface

Graphics display

BASIN Kernel

BASIN Client/GUI

IPython

Visualiz.
Data Analysis Engine

BASIN kernel: classes and functions for data distribution and parallel data operations.

(C++/MPI)

Objects visible to the user:

• **Region** defined by the data file read

• **Data**:
  - **Grid**
  - **List**

• **Attribute**
Data distribution and parallel operations

Attribute class

• Hide the complexity of data distribution, retrieval and parallel operations.
• Tools to ease the parallel data distribution and management on a distributed memory machine.
• Shared-memory view of data in a distributed memory machine.
• Arbitrary n-dimensional arrays of primitive datatypes or user-defined structures and objects.
• Parallel math (elementwise and reduction) and logical operations on distributed data.
• Each process can locate its own block of data.
Data distribution and parallel operations

Attribute class

• Hide the complexity of data distribution, retrieval and parallel operations.
• Tools to ease the parallel data distribution and management on a distributed memory machine.
• Shared-memory view of data in a distributed memory machine.
• Arbitrary n-dimensional arrays of primitive datatypes or user-defined structures and objects.
• Parallel math (elementwise and reduction) and logical operations on distributed data.
• Each process can locate its own block of data.
Data distribution and parallel operations

Attribute class

• Hide the complexity of data distribution, retrieval and parallel operations.
• Tools to ease the parallel data distribution and management on a distributed memory machine.
• Shared-memory view of data in a distributed memory machine.
• Arbitrary n-dimensional arrays of primitive datatypes or user-defined structures and objects.
• Parallel math (elementwise and reduction) and logical operations on distributed data.
• Each process can locate its own block of data.
Data distribution and parallel operations

Attribute class

- Hide the complexity of data distribution, retrieval and parallel operations.
- Tools to ease the parallel data distribution and management on a distributed memory machine.
- Shared-memory view of data in a distributed memory machine.
- Arbitrary n-dimensional arrays of primitive datatypes or user-defined structures and objects.
- Parallel math (elementwise and reduction) and logical operations on distributed data.
- Each process can locate its own block of data.
Data distribution and parallel operations

Attribute class

- Hide the complexity of data distribution, retrieval and parallel operations.
- Tools to ease the parallel data distribution and management on a distributed memory machine.
- Shared-memory view of data in a distributed memory machine.
- Arbitrary n-dimensional arrays of primitive datatypes or user-defined structures and objects.
- Parallel math (elementwise and reduction) and logical operations on distributed data.
- Each process can locate its own block of data.
Data distribution and parallel operations

Attribute class

- Hide the complexity of data distribution, retrieval and parallel operations.
- Tools to ease the parallel data distribution and management on a distributed memory machine.
- Shared-memory view of data in a distributed memory machine.
- Arbitrary n-dimensional arrays of primitive datatypes or user-defined structures and objects.
- Parallel math (elementwise and reduction) and logical operations on distributed data.
- Each process can locate its own block of data.
Data distribution and parallel operations

Attribute class

```python
reg0 = Region("/home/vesperin/starclust.dat")
m = reg0.get_attribute("mass")
x = reg0.get_attribute("x")
y = reg0.get_attribute("y")
z = reg0.get_attribute("z")
print m[10]
m[120] = 12.3
m[120] = 12.3
total_Mass = sum(m)
logm = log10(m)
Hm = where(m>5,m,0)
logr = log10(sqrt(x*x+y*y+z*z))
```
Center of Mass

- C/ MPI
- BASIN(C++/Python)

```c
...m, x...

MPI_Scatter(&m[0], n_loc, MPI_DOUBLE, &m_loc[0], n_loc, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Scatter(&m[0], n_loc, MPI_DOUBLE, &m_loc[0], n_loc, MPI_DOUBLE, 0, MPI_COMM_WORLD);

for(int i = 0; i < n_loc; i++)
{
    x_sum_loc += x_loc[i] * m_loc[i];
    m_sum_loc += m_loc[i];
}

MPI_Reduce(&x_sum_loc, &x_sum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Reduce(&m_sum_loc, &m_sum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

if(myRank == 0) {
    com[0] = x_sum / m_sum;
}

com=sum(m*x)/sum(m)
```
Data Analysis Engine

BASIN kernel: Scientific Packages

- **Cosmology**
- **Stellar dynamics**
- **Statistics**
- **FFT** (FFTW, http://www.fftw.org/)
- **Coordinate transformations**
BASIN Python User Interface and GUI

• BASIN Python interface created with Boost Python.
• A remote Python client can invoke BASIN commands to be executed by the Data Analysis Engine.
• Multiple distributed clients can connect to the same BASIN Data Analysis Engine and share the same data

( IPython/IPython1 F.Perez, B.Granger
http://ipython.scipy.org/moin/IPython1)
BASIN Python User Interface and GUI

• Connection to the remote server
• I/O
• Save/import session history
BASIN Python User Interface and GUI

Python Shell

```python
import subprocess
import bpy
from bpy import *

# Example:
reg0 = bpy.data.objects['Region0'].get_attribute('x')
```

Free Python Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Alias</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>Alias</td>
<td></td>
</tr>
<tr>
<td>reg0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log_basin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BASIN Data Structure

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Dimensions</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg0</td>
<td>Region</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Connected Parallel Mode: On, 2 nodes

Users

<table>
<thead>
<tr>
<th>Command</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg0</td>
<td>test</td>
</tr>
<tr>
<td>x</td>
<td>test</td>
</tr>
<tr>
<td>y</td>
<td>test</td>
</tr>
<tr>
<td>log=log10(x)</td>
<td>Enrico</td>
</tr>
</tbody>
</table>

Save Batch...

Clicking on a row will either select it (turning it white), or deselect it (turning it gray). Only those selected (white) commands will be saved to a batch file.
BASIN Python User Interface and GUI

- Visualization packages
BASIN Python User Interface and GUI

- Standard math functions
BASIN Python User Interface and GUI

• Scientific packages
BASIN Python User Interface and GUI

### BASIN Session data structure

<table>
<thead>
<tr>
<th>Data</th>
<th>Type</th>
<th>Dimensions</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>Region</td>
<td>list</td>
<td>None</td>
</tr>
<tr>
<td>Index</td>
<td>Attribute</td>
<td>2441.0</td>
<td>None</td>
</tr>
<tr>
<td>mass</td>
<td>Attribute</td>
<td>2441.0</td>
<td>None</td>
</tr>
<tr>
<td>w</td>
<td>Attribute</td>
<td>2441.0</td>
<td>None</td>
</tr>
<tr>
<td>y</td>
<td>Attribute</td>
<td>2441.0</td>
<td>None</td>
</tr>
<tr>
<td>z</td>
<td>Attribute</td>
<td>2441.0</td>
<td>None</td>
</tr>
</tbody>
</table>

### Reference to Attribute and “free” Attributes

### Python output

```python
import ipython.kernel.api as kernel
rc = kernel.RemoteController(('frinkiac.physics.drexel.edu', 10800))
rc.activate()

# Free attributes

def free_basin_import(* variables, **kwargs):
    free_basin_import(*variables, **kwargs)
```

### Server/local client switch

Python window

BASIN
BASIN architecture
Visualization

VisIt

developed at LLNL
http://www.llnl.gov/visit

• Visualization of large distributed datasets (structured and unstructured meshes)

• Parallel visualization engine

• Available in BASIN (in collaboration with Brad Whitlock at LLNL)
Visualization

Partiview

Developed at NCSA
http://virdir.ncsa.uiuc.edu/partiview/

- Visualization of 4D particle datasets
- BASIN/Partiview interface: work in progress (in collaboration with Stuart Levy, Matt Hall at NCSA)
Visualization

Gnuplot

• Simple plotting package based on the Gnuplot API.
• Only for development purposes.
• Available in BASIN
Visualization

Data transferred to the client machine: all the Python plotting packages (e.g. Matplotlib, Gnuplot, Chaco, etc.)
Summary

Goals:

Ease access to parallel data analysis
Avoid redundant development
Interactive and multi-user parallel data analysis
Summary

What we have:

Kernel for parallel data management and operations (C++/Python)

Scientific packages

Interface to a few existing visualization packages
Summary

What to look for next:

Increase science scope beyond astrophysics
Extend visualization options
Two-way communication with visualization packages
Improve ease of use and installation