Salsa: a parallel, interactive, particle-based analysis tool

Thomas Quinn, Laxmikant Kale, Filippo Gioachin, Orion Lawlor, Graeme Lufkin, Gregory Stinson
University of Illinois at Urbana-Champaign, University of Washington

Motivation
- Astronomical simulations/observations generate huge amount of data
- These data cannot be loaded into a single machine memory
- Even if they can be loaded, the interaction with the user can be too slow and inflexible
- Need for parallel analyzer tools which are able to:
  - Scale well to large number of processors
  - Provide flexibility to the user

Salsa Overview
- Built on top of Charm++ runtime system
- Client running on the user machine
- Server running on parallel platform
- Server offers:
  - Flexibility of dynamically modifying the data structures (Attribute framework)
  - Mechanism to upload code to be executed on the server (High level scripting)

Large datasets
- Planet Formation
- Universe Forming
- Simulation of the Universe (Evrard et al. 2002)

Virtual Observatory
- SDSS (1 million galaxies, hundreds of million of stars each)
- LSST (20 TB of images every night)
- Database available on the Internet

Server control flow
- Initialize structures
- Wait for a client to connect, and listen for the following events:
  - Listing/choosing of a simulation
  - Coloring requests
  - Box definitions (for zooming)
  - Group creation/modification
  - Attribute creation/modification
  - Python scripting code execution

Attribute framework
- A parallel structure for maintaining and updating all the particles
- Implemented as a Charm++ “array”, an indexed collection of objects distributed among the processors
- Particles are the basic entities
  - Grouped into families
  - Containing a list of attributes
- Operations can be performed either on (groups of) particles or attributes

Actions for particles
- Load a dataset
- Assign colors
- Generate an image
- Create groups
- Perform collective operations on a group like:
  - Compute center of mass
  - Compute total mass
- Select particles for Python “iterative” mode

Actions for attributes
- Create new attributes
- Delete attributes
- Modify an attribute for a group of particles
- Selectively return/modify values for Python low level interface

Large datasets
- 1,000,000 particles
- 80,000,000 particles
- 1,000,000,000 particles (snapshot size: 12GB)
- ??? TB, PB

Images as rendered by Salsa

A Galaxy colored by family (2.9M particles):
- dark matter in gray
- gas in red
- stars in yellow

Grouping/Zooming
- Boxing allows to define a box for the images to be displayed, which result in an effective zoom in

Coloring
- It is possible to color the particles using any of the attribute they have. Either present in the loaded data, or dynamically added
- A subset of the types present may be chosen
- A coloring based on the density...
- ...and one based on the potential

Grouping
- It is possible to select a group of particles by an interval of an attribute, and apply coloring or scripting to this group
- Selected group...
- ...and colored with density

Rotating
- The data can be visualized using any projection plane in the 3D space
- With the plane rotating, the server provides new images to render in real time

Planet Formation
Universe Forming
Simulation of the Universe
Virtual Observatory

SDSS (1 million galaxies, hundreds of million of stars each)
LSST (20 TB of images every night)
Database available on the Internet

Images as rendered by Salsa

A Galaxy colored by family (2.9M particles):
- dark matter in gray
- gas in red
- stars in yellow

Grouping/Zooming
- Boxing allows to define a box for the images to be displayed, which result in an effective zoom in

Coloring
- It is possible to color the particles using any of the attribute they have. Either present in the loaded data, or dynamically added
- A subset of the types present may be chosen
- A coloring based on the density...
- ...and one based on the potential

Grouping
- It is possible to select a group of particles by an interval of an attribute, and apply coloring or scripting to this group
- Selected group...
- ...and colored with density

Rotating
- The data can be visualized using any projection plane in the 3D space
- With the plane rotating, the server provides new images to render in real time

Salsa: a parallel, interactive, particle-based analysis tool
Thomas Quinn, Laxmikant Kale, Filippo Gioachin, Orion Lawlor, Graeme Lufkin, Gregory Stinson
University of Illinois at Urbana-Champaign, University of Washington
Salsa: a parallel, interactive, particle-based analysis tool
Thomas Quinn, Laxmikant Kale, Filippo Gioachin, Orion Lawlor, Graeme Lukkin, Gregory Stinson
University of Illinois at Urbana-Champaign, University of Washington

Charm++ Architecture

- Software engineering
  - Number of virtual processors can be independently controlled
- Message driven execution
  - Computation performed upon receipt of a message
  - Predictability
  - Automatic out-of-core execution
  - Asynchronous reductions
- Dynamic mapping
  - Heterogeneous clusters
  - Vacate, adjust to speed, share
  - Automatic checkpointing/restarting
  - Automatic dynamic load balancing
  - Change set of processors used
- Communication optimization

CCS – Converse Client-Server Protocol

- Protocol similar to http
  - Available with every Charm++ program
  - Full parallelism
  - Multi-interpreters concurrently running
  - Persistence of information across calls
  - Orthogonality of usage modes
  - Orthogonality of objects
  - Each Charm++ object can export an interface to Python
  - No need for recompile and reload
  - User customization

Benefits

- Software engineering
  - Number of virtual processors can be independently controlled
  - Separate VPs for different modules
- Message driven execution
  - Computation performed upon receipt of a message
  - Predictability
  - Automatic out-of-core execution
  - Asynchronous reductions
- Dynamic mapping
  - Heterogeneous clusters
  - Vacate, adjust to speed, share
  - Automatic checkpointing/restarting
  - Automatic dynamic load balancing
  - Change set of processors used
- Communication optimization

LiveViz

- Uses CCS functionality
- Upon request, every object creates a piece of image
- The image is combined and sent back to the client
- Scales well with number of processors

CCS – Converse Client-Server Protocol

- Protocol similar to http
  - Available with every Charm++ program
  - Full parallelism
  - Multi-interpreters concurrently running
  - Persistence of information across calls
  - Orthogonality of usage modes
  - Orthogonality of objects
  - Each Charm++ object can export an interface to Python
  - No need for recompile and reload
  - User customization

Future work

- Flexibility on parameter marshalling
- Integration of other high level scripting languages, c++ as next
- Enabling all features available from the interface in the client
- More refined and accurate group finding
- Building data structures over the particles
- Integration with active simulation

High level language scripting

Features

- Use Once
- Run Once
- Prints release
- Low level
- Server function

Usage modes

- Persistent
- Iterative
- Prints retain
- High level

Temporal messages exchanged

- Client
- Server
- Parallel program
- Send request
- Execute the request
- Combine results

Client window

- Access the method “countParticles” exposed by the object
- Print the string into the client
- The output of the execution
- The output of the execution

The “ck” module

- Allows Python to call suspendable methods
- Charm++ can start a parallel task
- When the results are ready, Python will be resumed with the correct return value

The “charm” module

- MyArray exposes Python functionality through CCS
- array [1D, python] MyArray {
  entry [python] void mymethod(int handle);
}

The “charm” module

- Implements methods to create and update an iterator over particles
  - buildIterator
  - nextIteratorUpdate
- Script executed for every particle selected by the iterator

The “charm” module

- Allows Python to call suspendable methods
- Charm++ can start a parallel task
- When the results are ready, Python will be resumed with the correct return value

The “charm” module

- MyArray exposes Python functionality through CCS
- array [1D, python] MyArray {
  entry [python] void mymethod(int handle);
}

Copy code as server

Future work

- Flexibility on parameter marshalling
- Integration of other high level scripting languages, c++ as next
- Enabling all features available from the interface in the client
- More refined and accurate group finding
- Building data structures over the particles
- Integration with active simulation

Salsa: a parallel, interactive, particle-based analysis tool
Thomas Quinn, Laxmikant Kale, Filippo Gioachin, Orion Lawlor, Graeme Lukkin, Gregory Stinson
University of Illinois at Urbana-Champaign, University of Washington

CCS – Converse Client-Server Protocol

- Protocol similar to http
  - Available with every Charm++ program
  - Full parallelism
  - Multi-interpreters concurrently running
  - Persistence of information across calls
  - Orthogonality of usage modes
  - Orthogonality of objects
  - Each Charm++ object can export an interface to Python
  - No need for recompile and reload
  - User customization

Virtual Object

User View: object carrying the job

Real processor

Software engineering

- Number of virtual processors can be independently controlled
- Separate VPs for different modules

Message driven execution

- Computation performed upon receipt of a message
- Predictability
- Automatic out-of-core execution
- Asynchronous reductions

Dynamic mapping

- Heterogeneous clusters
- Vacate, adjust to speed, share
- Automatic checkpointing/restarting
- Automatic dynamic load balancing
- Change set of processors used
- Communication optimization

LiveViz

- Uses CCS functionality
- Upon request, every object creates a piece of image
- The image is combined and sent back to the client
- Scales well with number of processors

CCS – Converse Client-Server Protocol

- Protocol similar to http
  - Available with every Charm++ program
  - Full parallelism
  - Multi-interpreters concurrently running
  - Persistence of information across calls
  - Orthogonality of usage modes
  - Orthogonality of objects
  - Each Charm++ object can export an interface to Python
  - No need for recompile and reload
  - User customization

Benefits

- Software engineering
  - Number of virtual processors can be independently controlled
  - Separate VPs for different modules
- Message driven execution
  - Computation performed upon receipt of a message
  - Predictability
  - Automatic out-of-core execution
  - Asynchronous reductions
- Dynamic mapping
  - Heterogeneous clusters
  - Vacate, adjust to speed, share
  - Automatic checkpointing/restarting
  - Automatic dynamic load balancing
  - Change set of processors used
  - Communication optimization

LiveViz

- Uses CCS functionality
- Upon request, every object creates a piece of image
- The image is combined and sent back to the client
- Scales well with number of processors

CCS – Converse Client-Server Protocol

- Protocol similar to http
  - Available with every Charm++ program
  - Full parallelism
  - Multi-interpreters concurrently running
  - Persistence of information across calls
  - Orthogonality of usage modes
  - Orthogonality of objects
  - Each Charm++ object can export an interface to Python
  - No need for recompile and reload
  - User customization

Future work

- Flexibility on parameter marshalling
- Integration of other high level scripting languages, c++ as next
- Enabling all features available from the interface in the client
- More refined and accurate group finding
- Building data structures over the particles
- Integration with active simulation

http://charm.cs.uiuc.edu

http://nchilada.astro.washington.edu