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

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)

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

Flexibility

- 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

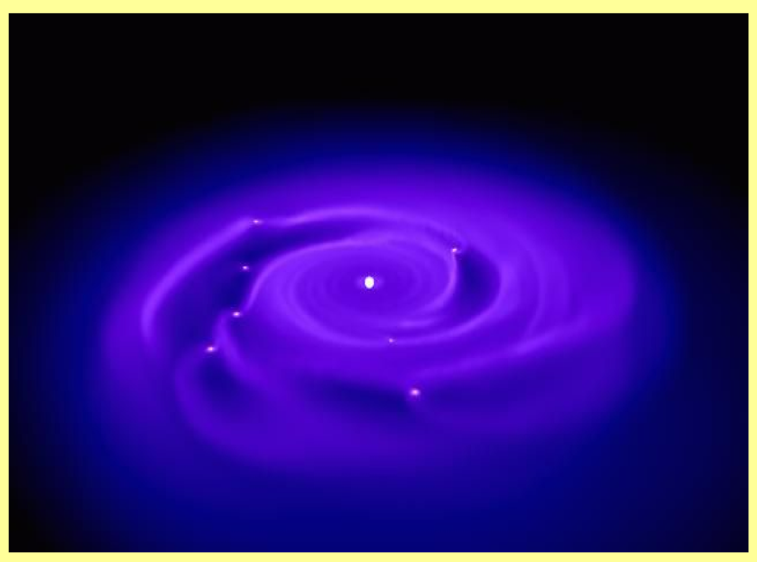
- Load a dataset
- Assign colors
- Generate an image
- Create groups
- Perform collective operations on a group like:
  - Compute center of mass
  - Compute total mass
- Select particles for Python "iterative" mode

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

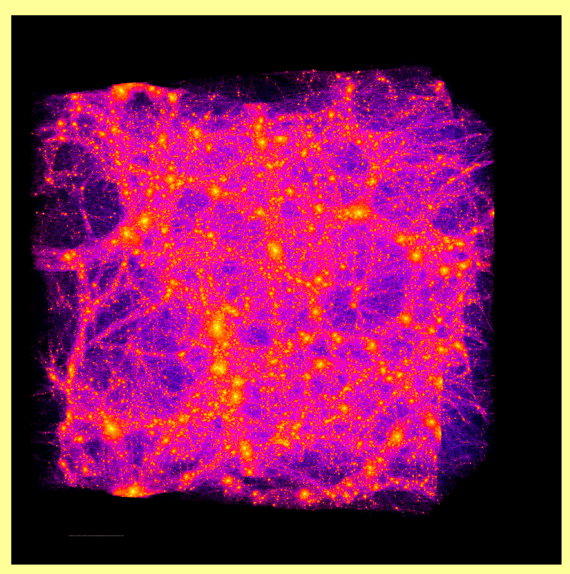
Large datasets

Planet Formation



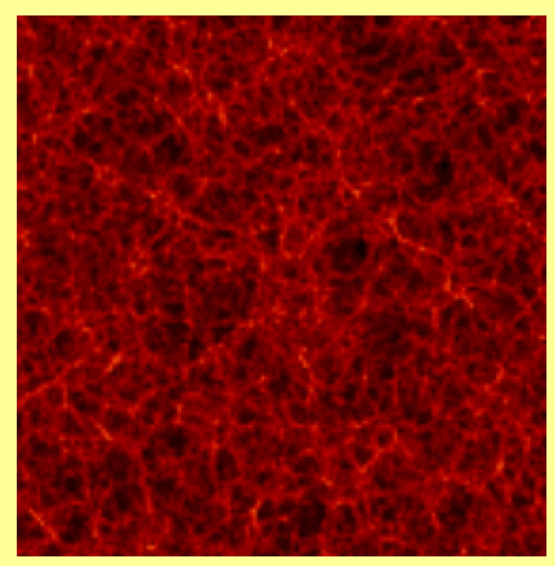
1,000,000 particles

Universe Forming



80,000,000 particles

Simulation of the Universe (Evrard et al. 2002)



1,000,000,000 particles (snapshot size: 12GB)

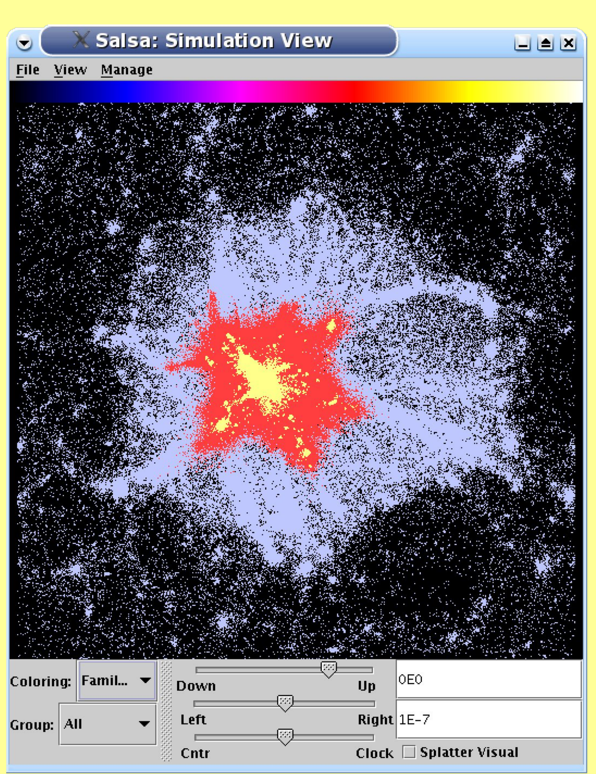
Virtual Observatory

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

??? TB, PB

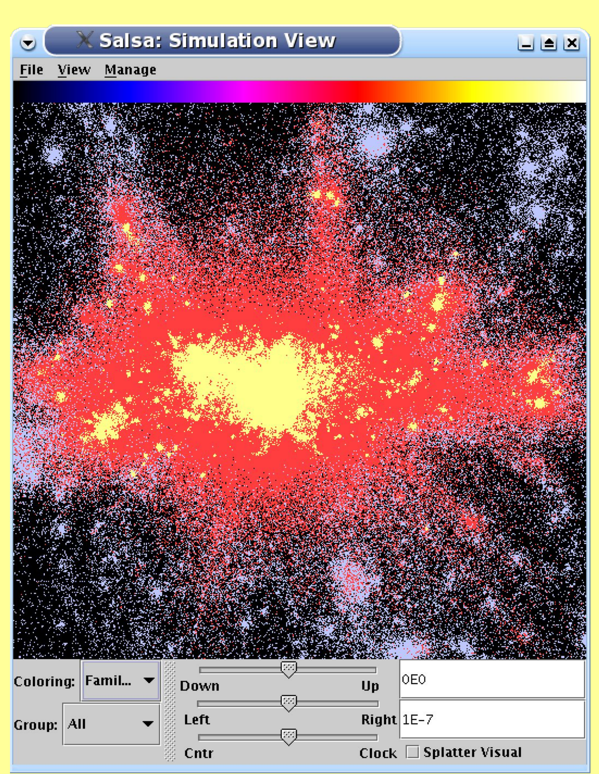
Images as rendered by Salsa

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



Grouping/Zooming

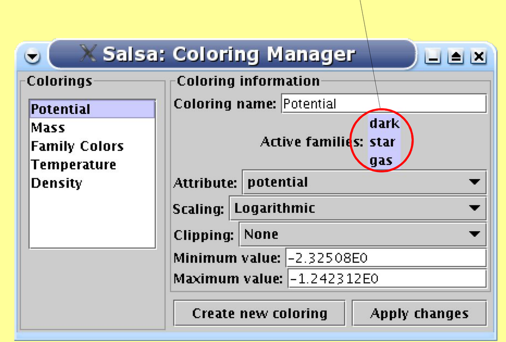
Boxing allows to define a box for the images to be displayed, which result in an effective zoom in

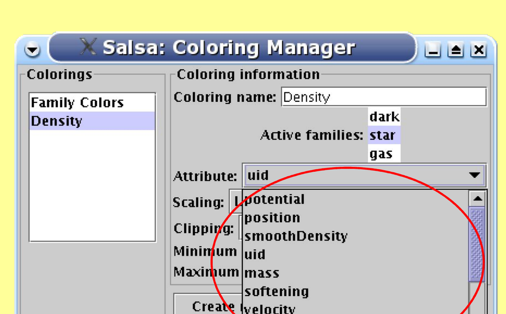


Coloring

It is possible to color the particles using any of the attribute they have. Either present in the loaded data, or dynamically added

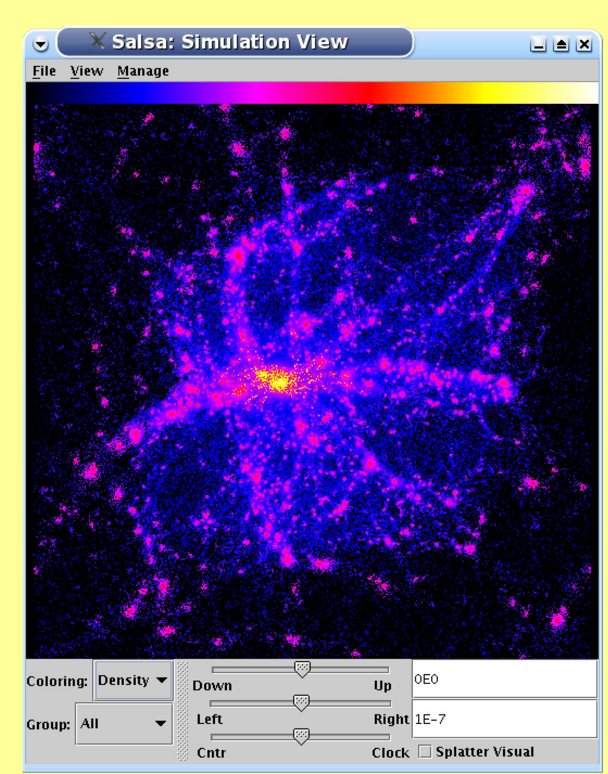
A subset of the types present may be choosen



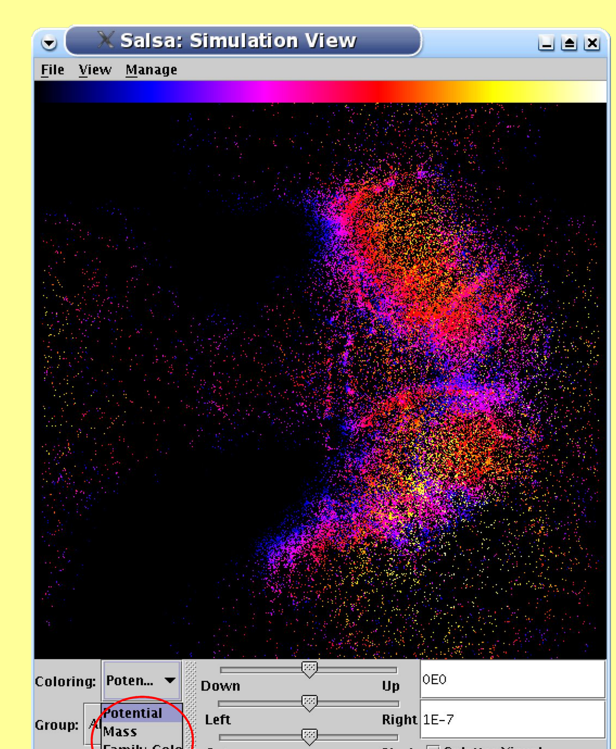


List of all attributes available

A coloring based on the density...



...and one based on the potential

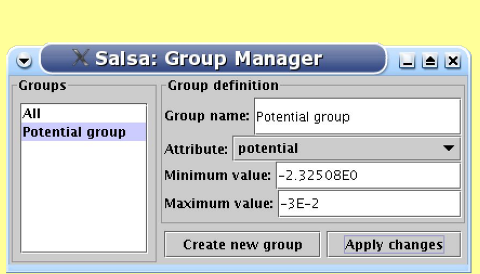


Colorings available

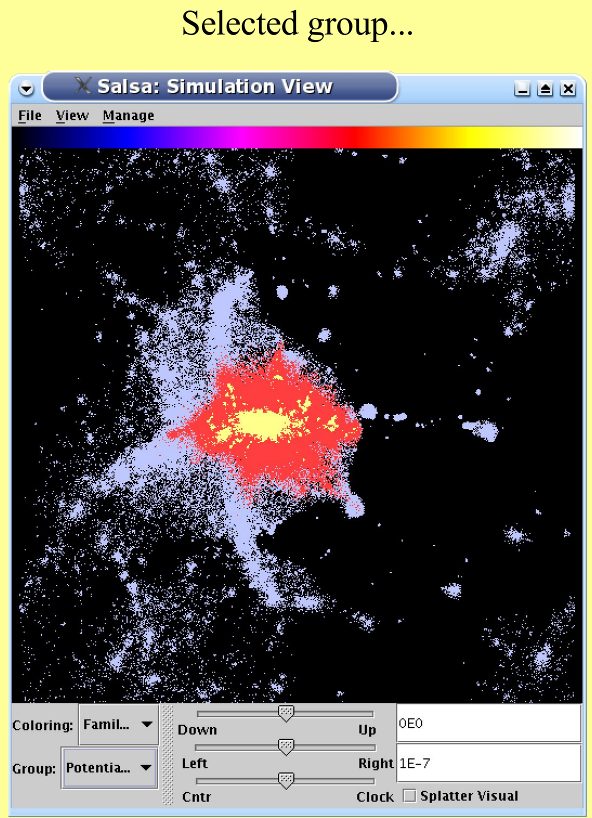
Grouping

It is possible to select a group of particles by an interval of an attribute, and apply coloring or scripting to this group.

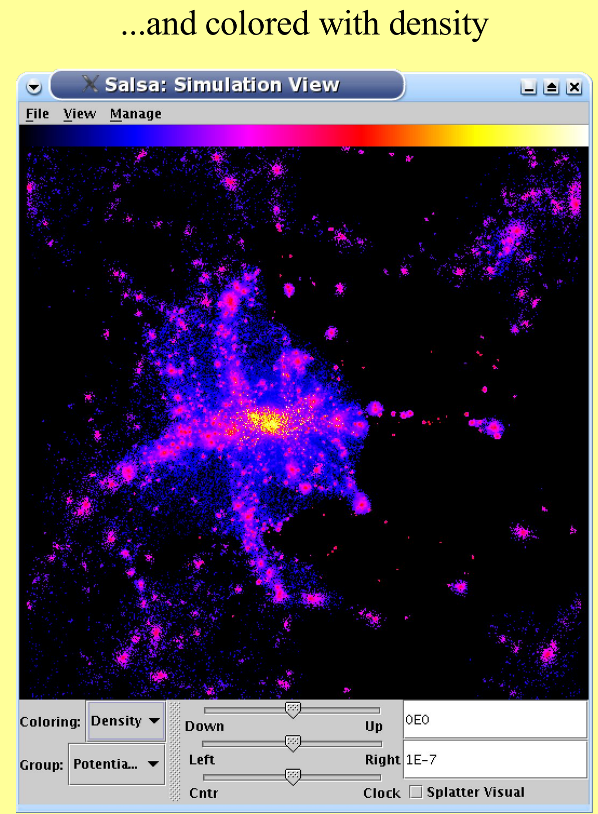
Window to select particles in a window of potential, and its following coloring with its density attribute



Selected group...

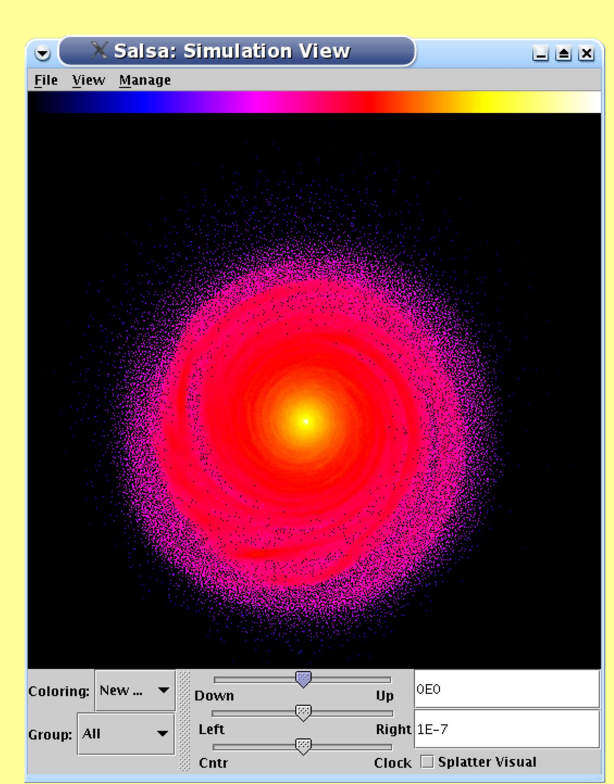


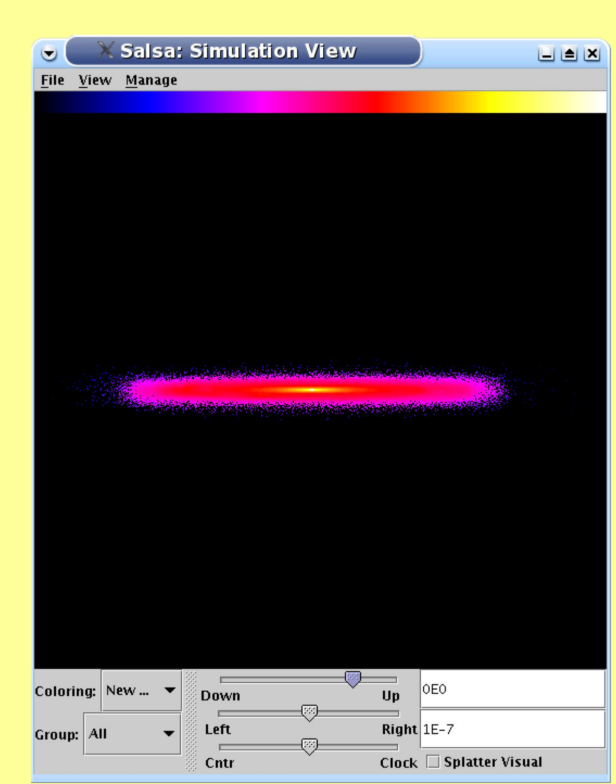
...and colored with density



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.

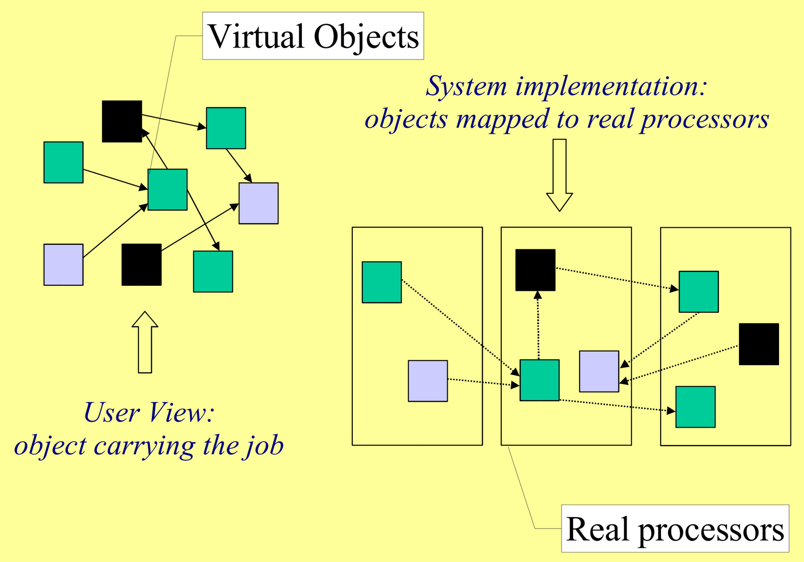




Salsa: a parallel, interactive, particle-based analysis tool

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

Charm++ Architecture

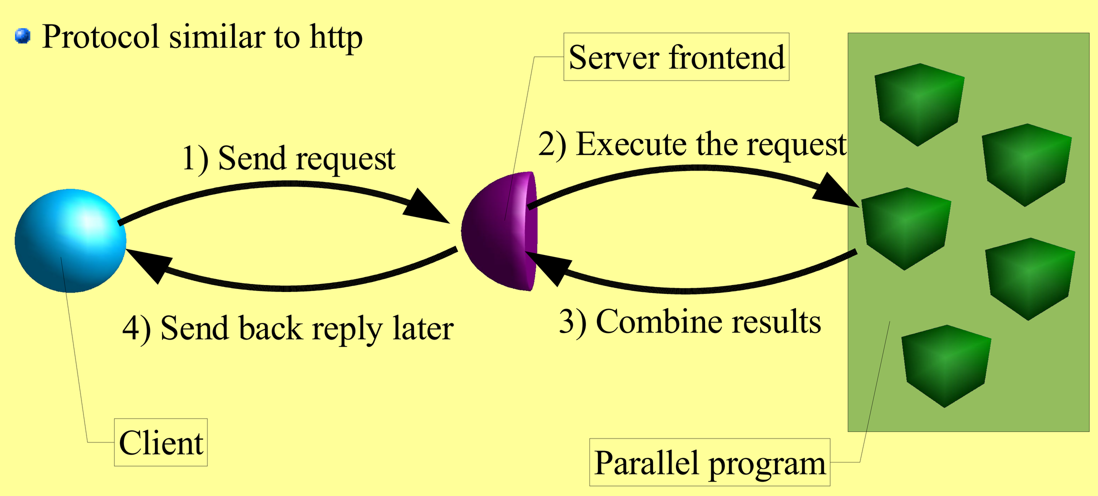


Benefits

- 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
  - Adaptive overlap of communication
  - 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

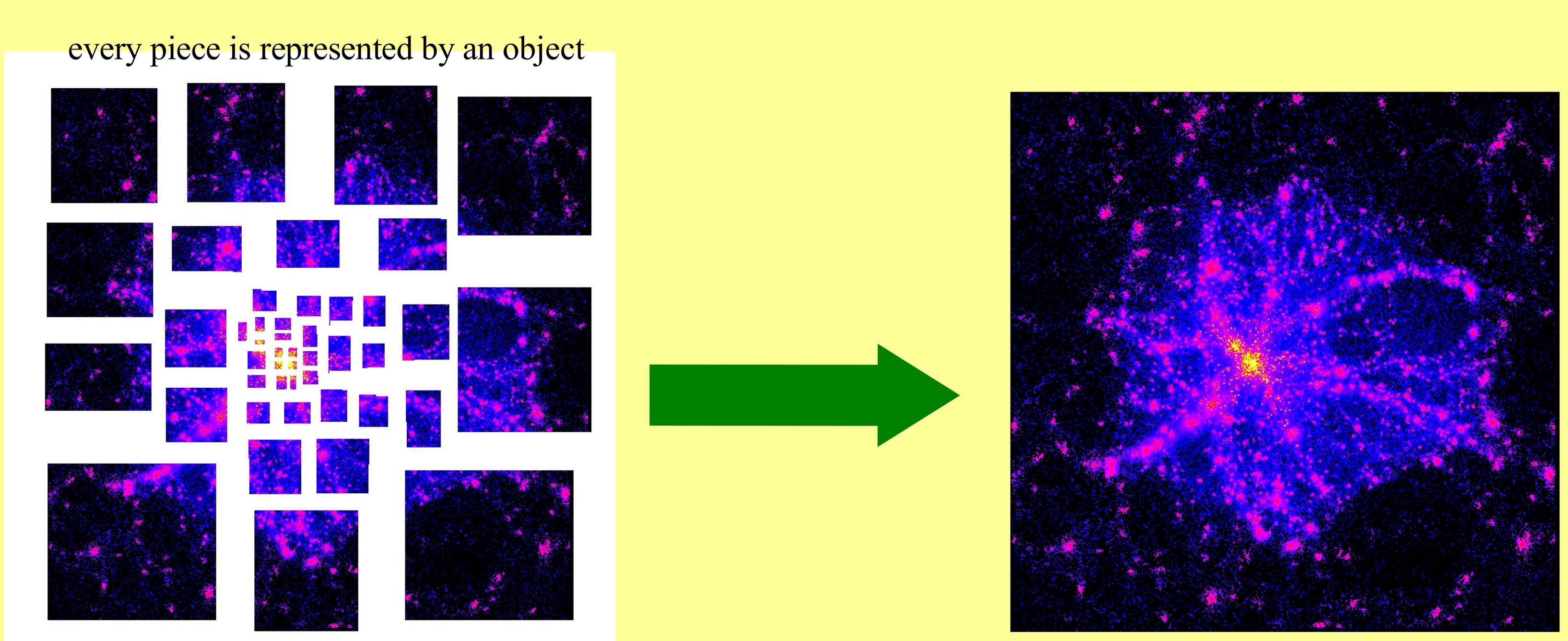
CCS – Converse Client-Server Protocol

- Protocol similar to http



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

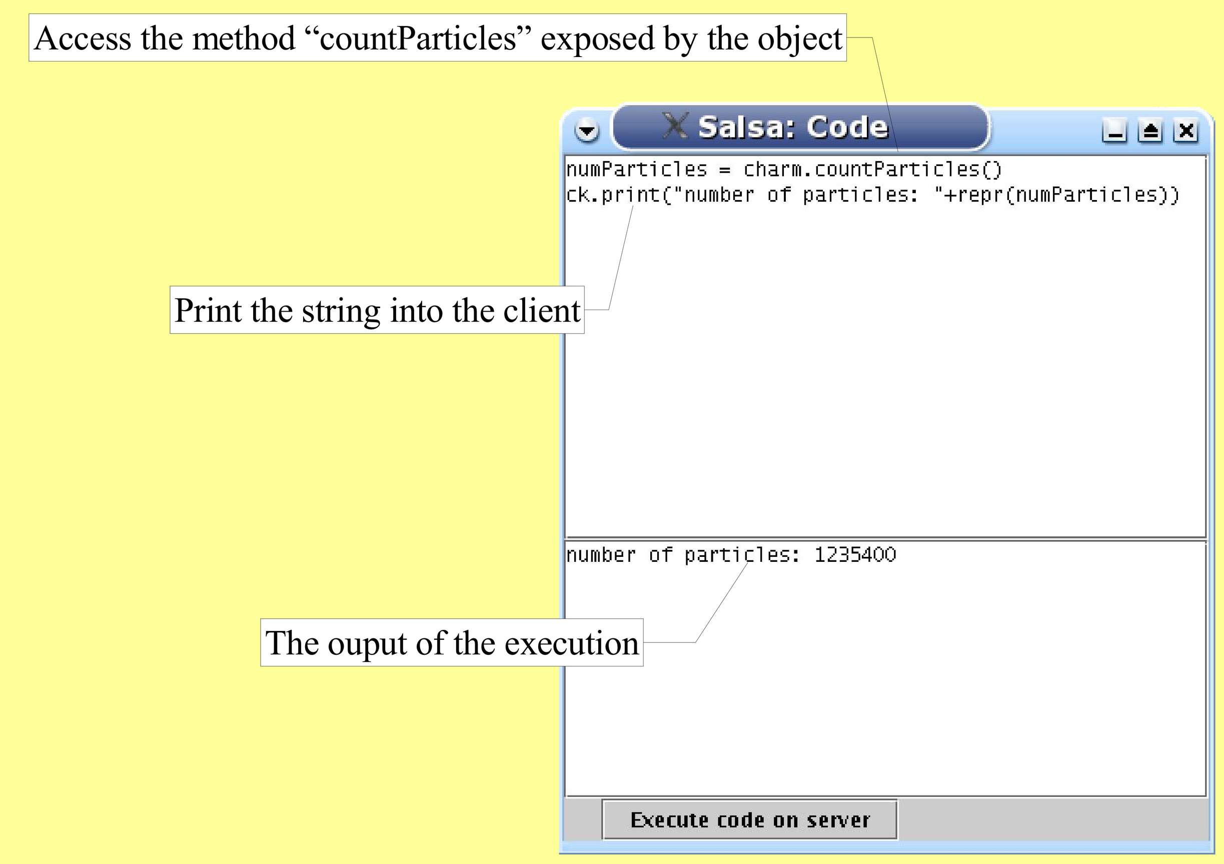


High level language scripting

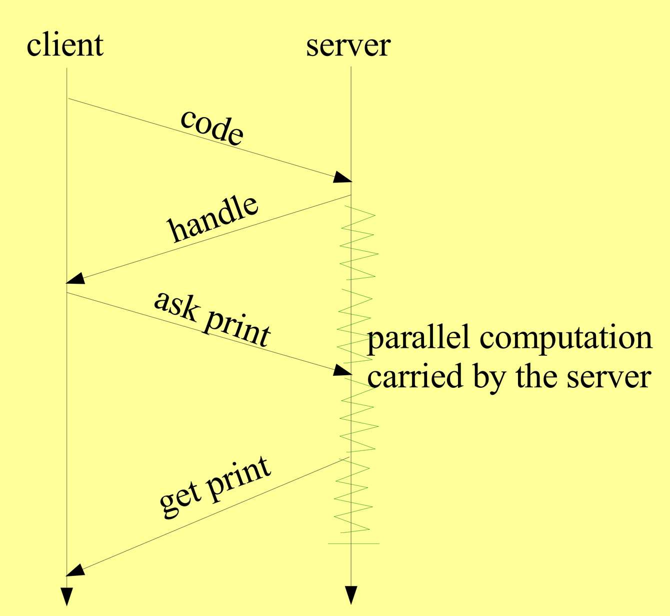
Features

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

Client window



Temporal messages exchanged



Usage modes

- Use Once
- Persistent
- Run Once
- Iterative
- Prints release
- Prints retain
- Low level
- High level

Iterative:

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

The "ck" module

- Directly access to a single particle
  - read(where)
  - write(where, what)
- Basic system utilities
- Prints forwarding

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

MyArray exposes Python functionality through CCS

```
array [1D, python] MyArray {
   entry ...
   entry [python] void mymethod(int handle);
}
```

mymethod is accessible through the "charm" module

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

PARALLEL PROGRAMMING LABORATORY
http://charm.cs.uiuc.edu

N-chilada
http://nchilada.astro.washington.edu