Large Data Visualization using Shared Distributed Resources

Huadong Liu, **Micah Beck**, Jian Huang, Terry Moore
Department of Computer Science
University of Tennessee
Knoxville, TN

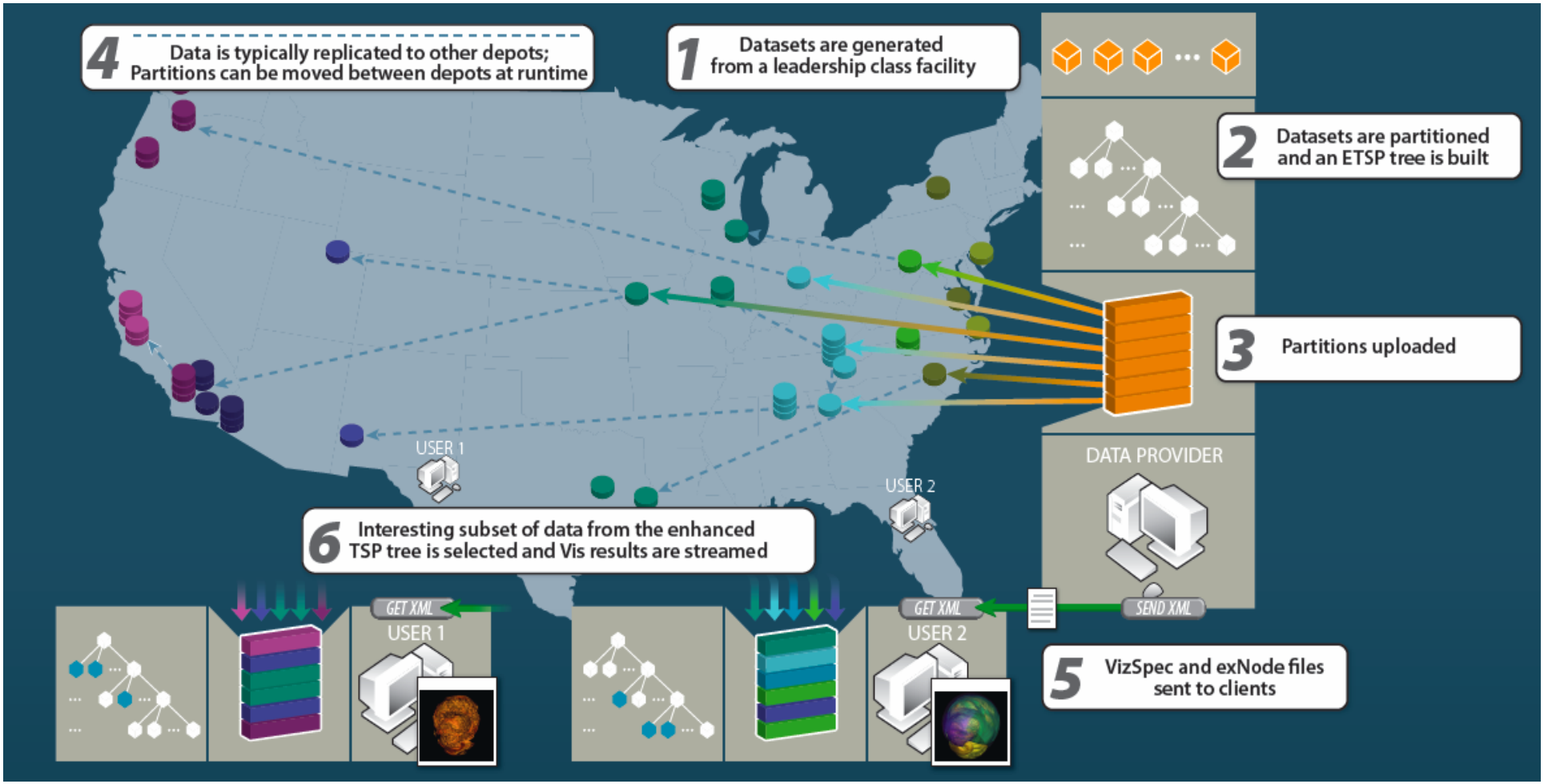
Background

- To use large-scale shared resources for cutting edge computation jobs is a great idea
  - Communication: “the Network”
  - Storage and Computation: “The Grid”?
- To implement this vision for production use, several high-level services are needed. For example:
  - Resource discovery and management (reservation)
  - Data transfer
  - Scheduling (dynamic monitoring, adaptive control)
- Our model is the Network, not the Comp/Data Center
  - Logistical Networking infrastructure and architecture
  - Internet Backplane Protocol, exNode, LoRS
- NGS Funding: PIs Beck, Dongarra, Huang, Plank

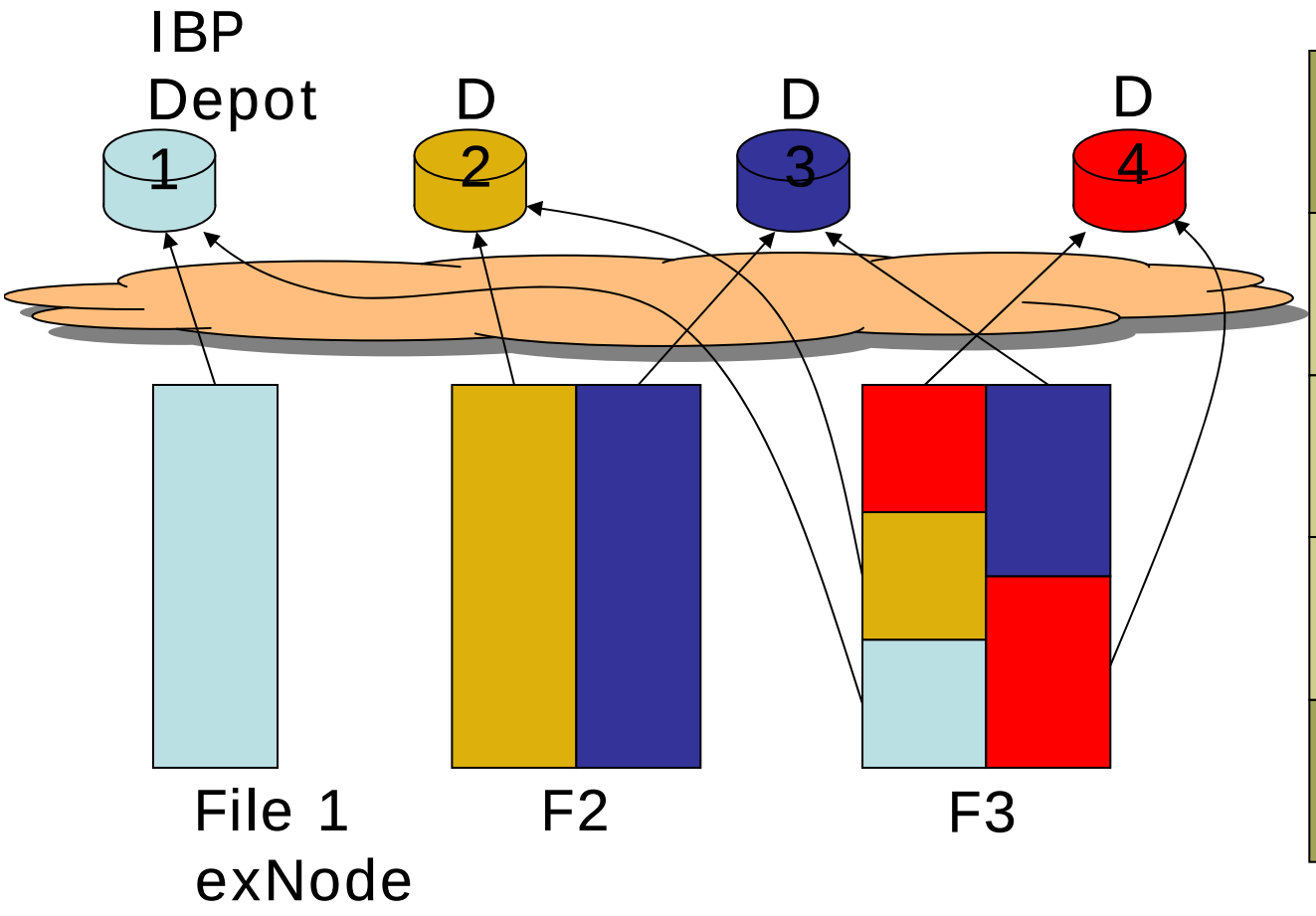
Distributed Visualization

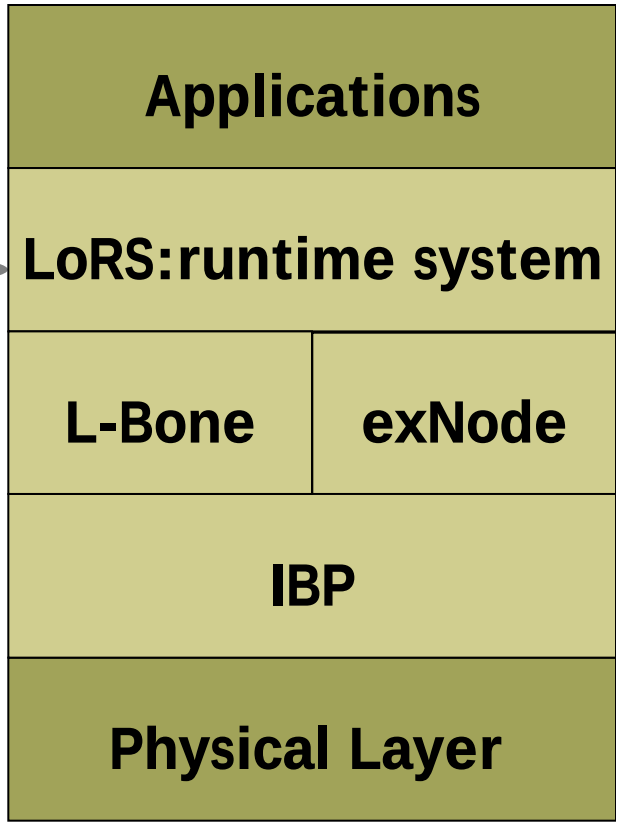
- Our work focuses on large data visualization:
  - Useful when available on-demand
  - Useful when can be shared in an executable form
  - Use as many processors as available (beyond clusters?)
  - Available in a widespread manner
  - Data intensive
- Non-standard definition of Distributed Viz
  - We aim to support geographically distributed users
  - The infrastructure does not need to be centralized
  - Our comp/storage nodes are independent network nodes

Distributed Visualization



Data Replication: IBP & exNodes





Visualization Operations

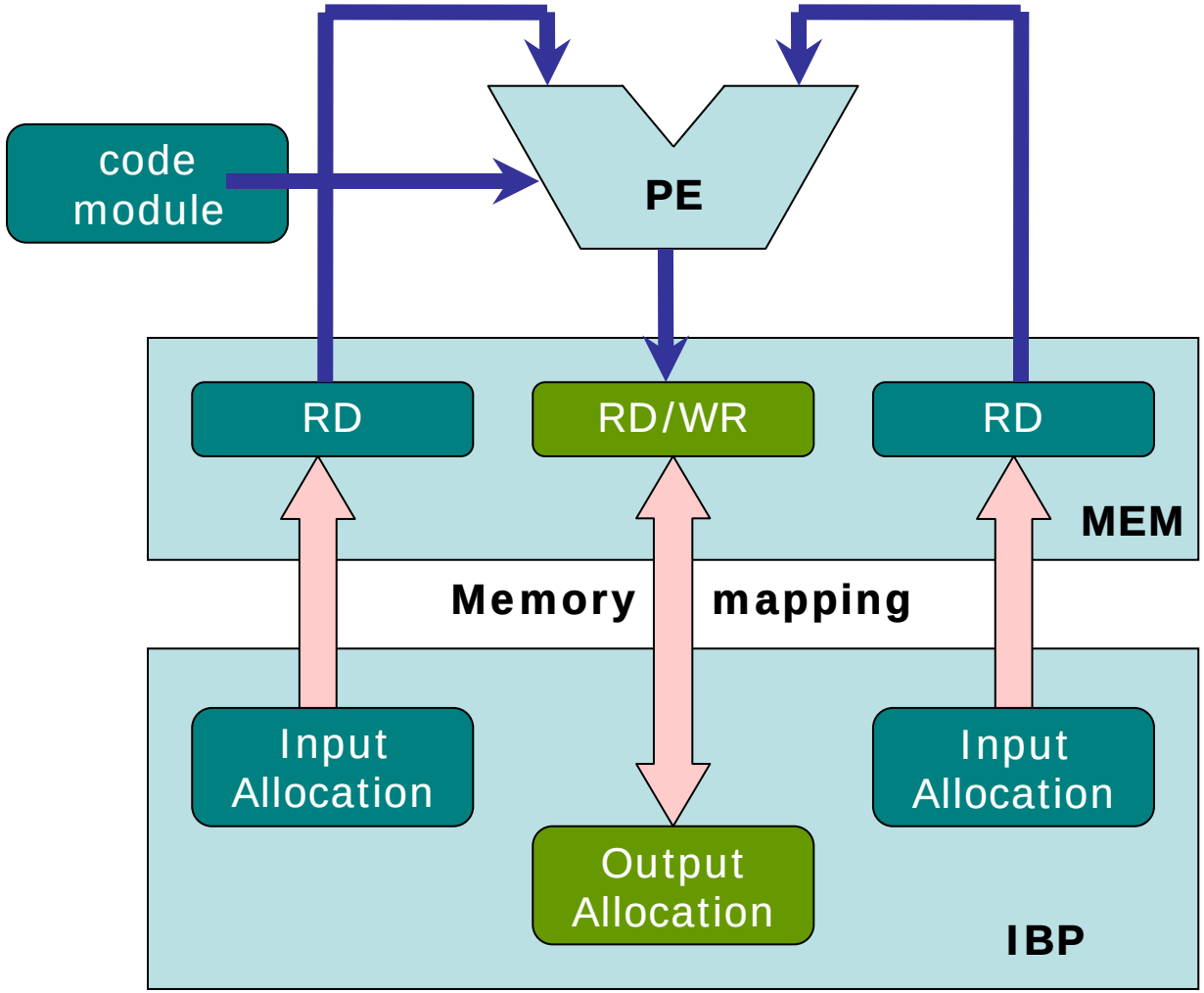
- We constructed a set of basic visualization operations as a highly portable library:
  - the Visualization Cookbook Library (vcblib)
  - includes major visualization algorithms like software volume rendering, iso-surfacing and flow visualization
  - builds and runs on Unix, Linux, Windows and Mac OS.

- vcblib provides a reliable and portable building block to deploy visualization operations to the wide area.

Executing vcblib ops on NFU

- NFU (Network Functional Unit) is a generic, best effort computation service
  - Maximum memory size
  - Limited duration of execution
  - Weak semantics
- Strong services must be constructed on top (I.e. the scheduler of the parallel visualization algorithm)

Network Functional Unit

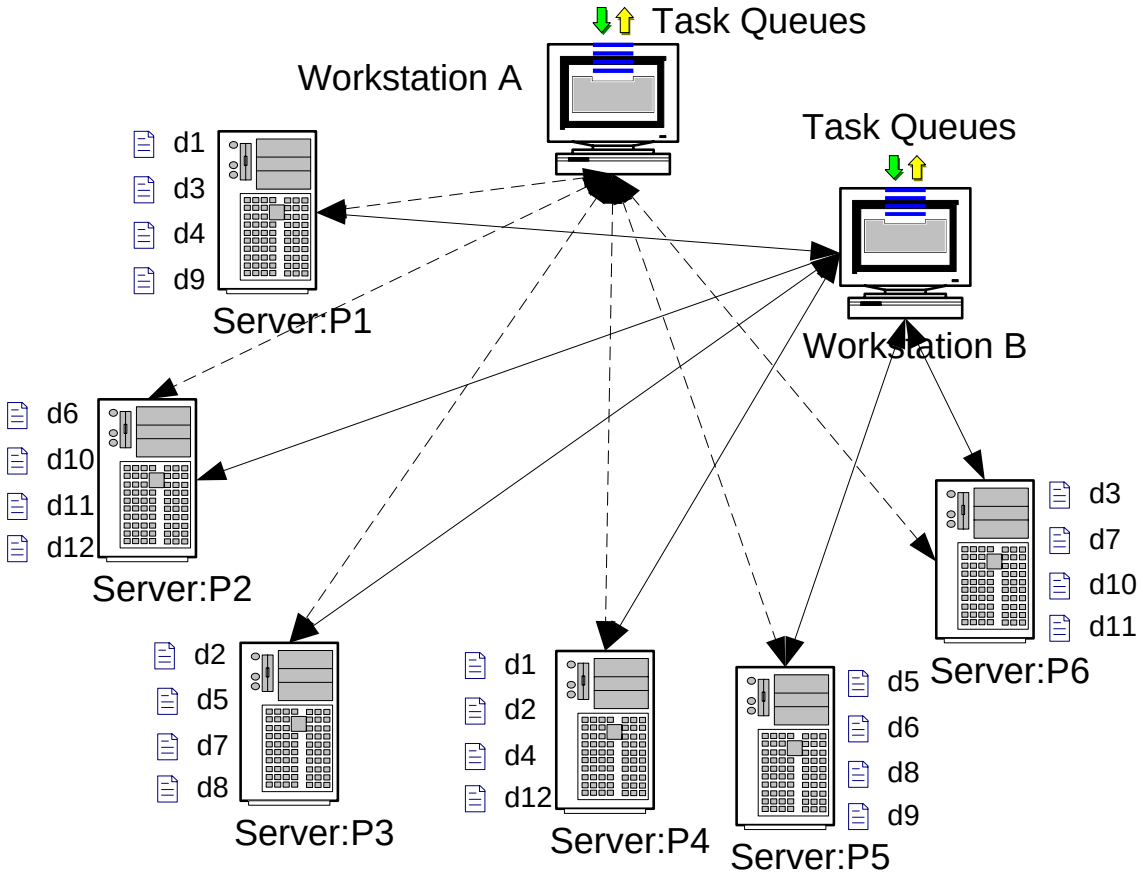


NFU is novel due to:

1. weakened semantic and
2. control of security-sensitive operations.
3. Supports both local and mobile code modules

Scheduling

Huadong Liu, University of Tennessee



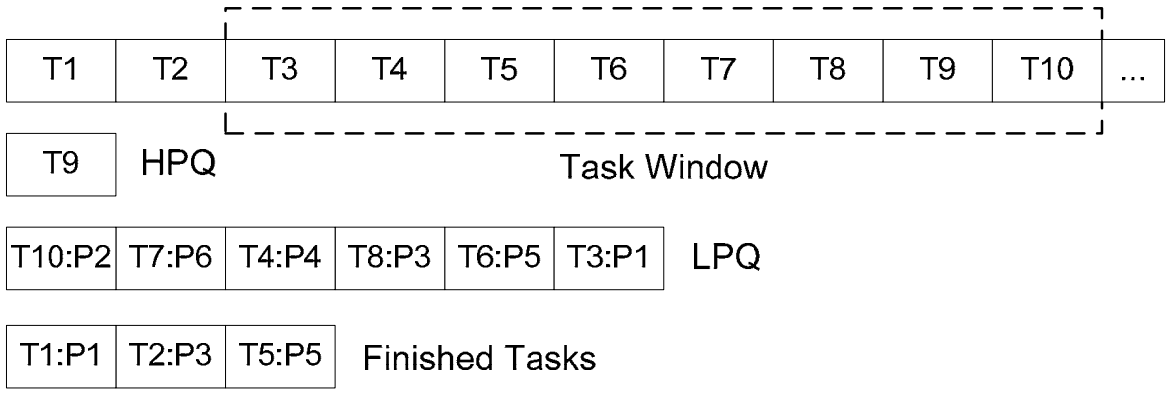
- Depots: {*P1*,*P2*,…,*Pm*}
  Pi described by bw *bi* & computational power *ci*
- Partitioned dataset {*d1*,*d2*,…, *dn*}, k-way replication
- Vis only needs one copy of each *dj*
- (Optional) DM tasks
  Mij replicates *dj* on *Pi*

Key Challenge:

Resource performance varies over time !!!

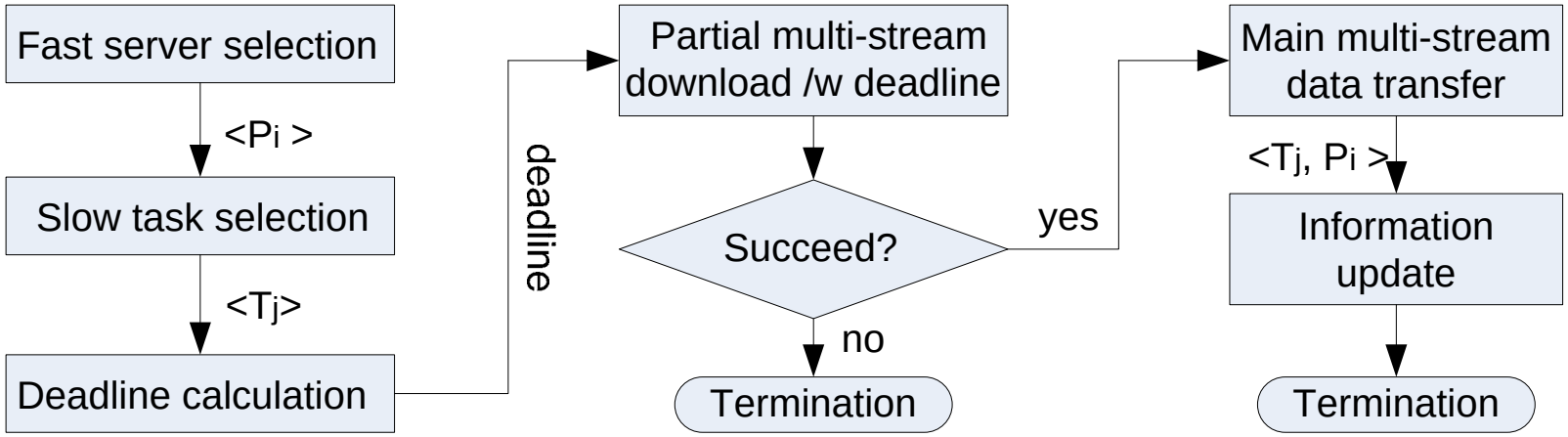
Scheduling

- Depots are ranked by number of volume partitions processed so far
- High vs. Low priority queues (HPQ vs. LPQ) of tasks
  - HPQ: tasks-to-be-assigned, keyed by shortest potential processing time
  - LPQ: tasks-already-assigned, keyed by longest potential wait time



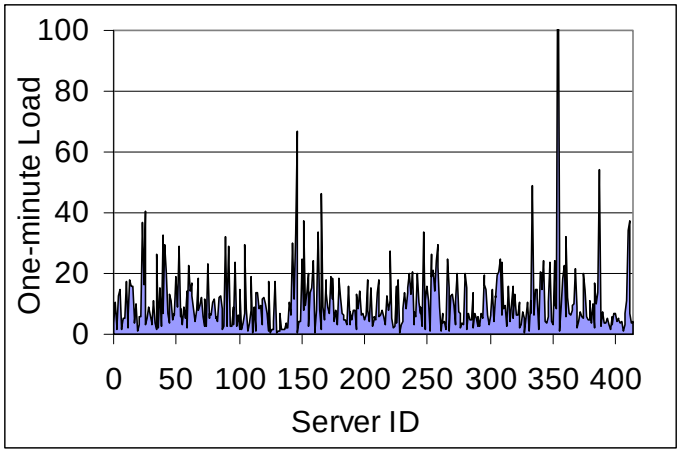
Dynamic Data Movement

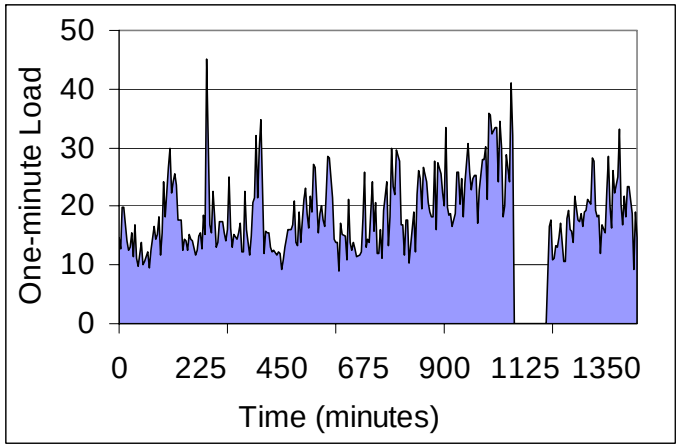
- Some data partitions are just “unlucky” to be on slow or heavily loaded servers
- After fast depots are done with local tasks, can dynamically “steal” some slow “partitions”



Results: the depots

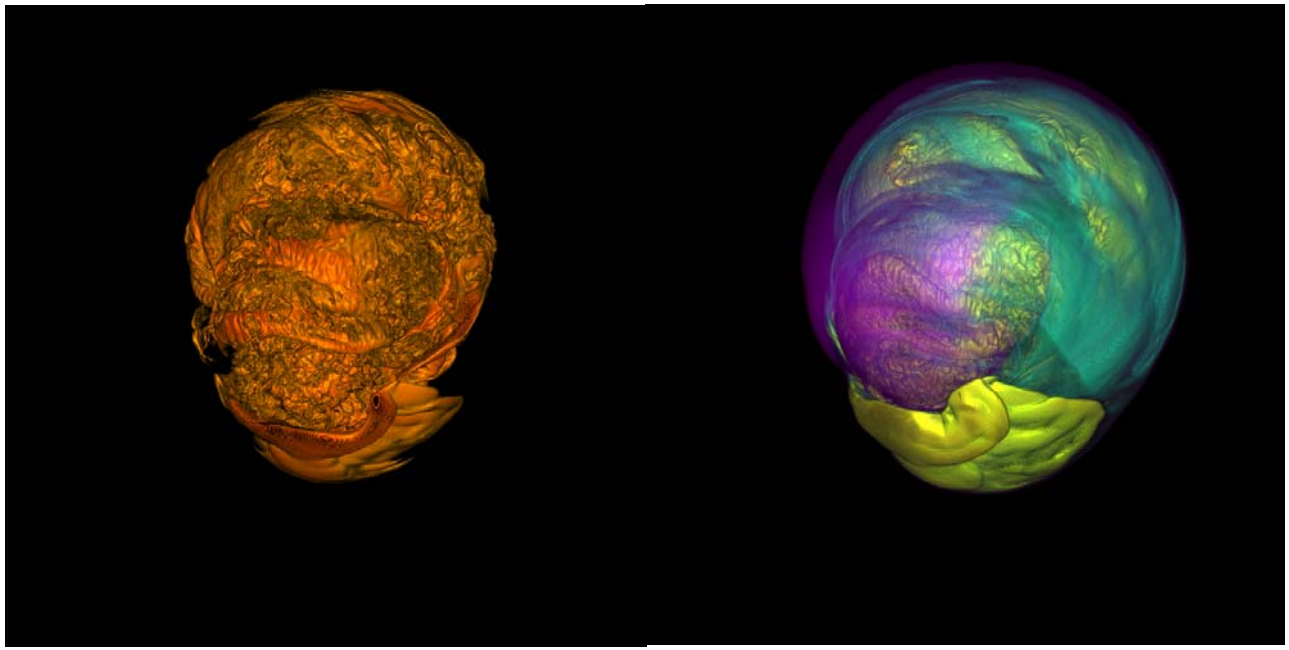
- Most of our test depots are Planet-Lab nodes
- The machines workload varies much from one to one
- The workload is also highly time varying





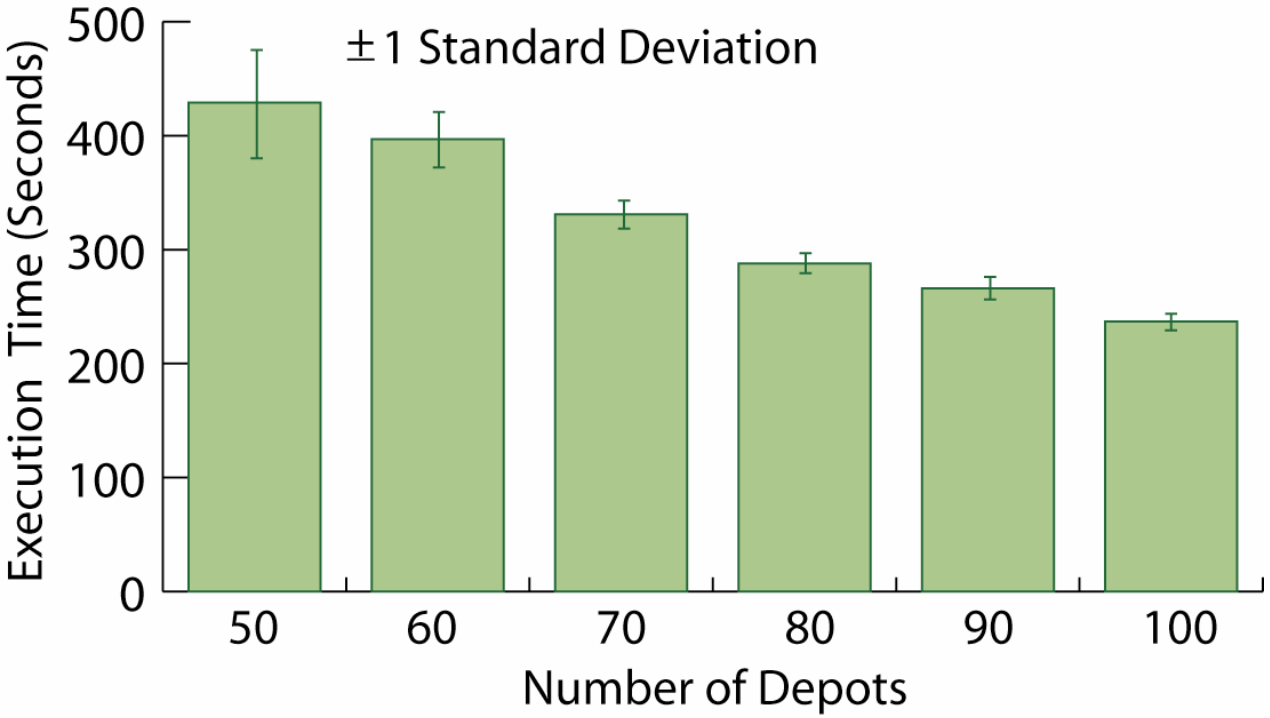
Results: the data

- Test data: 30 timestep of Tera-scale Supernova Initiative, 75GB in total
  - Provided by Tony Mezzacappa (ORNL) and John Blondin (ORNL) under the auspices of DOE SciDAC TSI project



Results: the performance

- 800x800 image resolution, 0.5 step size in ray-casting, per-fragment classification and Phong shading
- With 100 depots, the average rendering time: 237 sec



To the User

- You program your visualization by editing an XML file
  - ASCII file, 3KB in size
  - A template is provided

```
<viewing_parameters>
        <lighting num_lights=2> ... <lighting>
        <viewport_transform> ... <viewport_transform>
        <data_specific> ... <data_specific> ...
<viewing_parameters>
<raycasting>
        <scale> ... </scale>
        <color_scheme> ... </color_scheme>
        <interval_range> ... </interval_range>
</raycasting> ...
```

Let’s go to the video…