Plush: An Infrastructure for Managing and Visualizing Distributed Applications

Jeannie Albrecht, Ryan Braud, Darren Dao, John Jersin, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat
University of California, San Diego
{jalbrecht, rbraud, ctuttle, snoeren, vahdat} @ cs.ucsd.edu, {hdao, jjersin} @ ucsd.edu http://plush.ucsd.edu

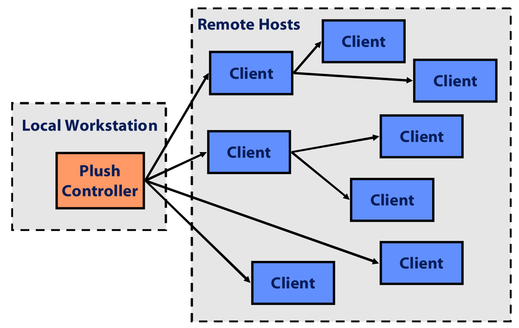
Goal: Provide an extensible application management system for large-scale distributed systems

Motivation

- **Problem:** How do we deploy, manage, and maintain distributed applications that simultaneously run on hundreds of heterogeneous physical machines?
- Existing approaches for finding resources and managing applications are cumbersome, manual, and error-prone
- Tools exist to address some issues, but utility is limited by lack of integration
- **Plush** provides a unified environment to support the distributed application design and deployment lifecycle on PlanetLab and in clusters

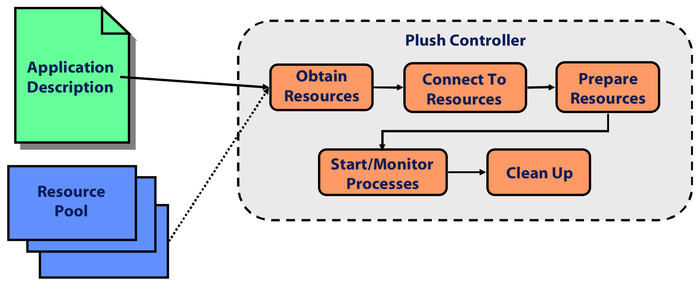
Architecture

- **Plush** consists of an application **controller** that communicates with **client** processes running on each of the available resources
- Application description specifies resources, software, program execution, synchronization requirements, and process monitoring details

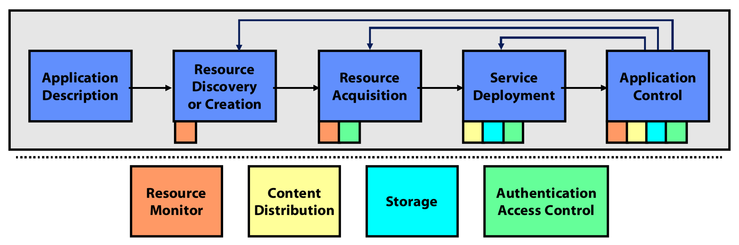


The Plush Controller

- Takes application description and pool of resources (optional) as input
- Uses a resource matcher to select/allocate resources based on user criteria
- Installs a set of user-defined software packages and application files
- Configures and starts processes, monitoring the running application and resources throughout execution
- Performs cleanup actions at the clients after execution completes



Pluggable Framework



- Typical distributed application lifecycle, including the 5 main lifecycle stages (blue boxes) and their key underlying components (other boxes)
- **Plush** allows users to plug in customized, environment-specific services for each phase in the lifecycle using simple XML-RPC interfaces
- Sample plug-in services:
  - SWORD for resource discovery and acquisition
  - Shirako or Usher to create Xen virtual machines on demand
  - Bullet/CoBlitz for efficient service deployment

Application Description

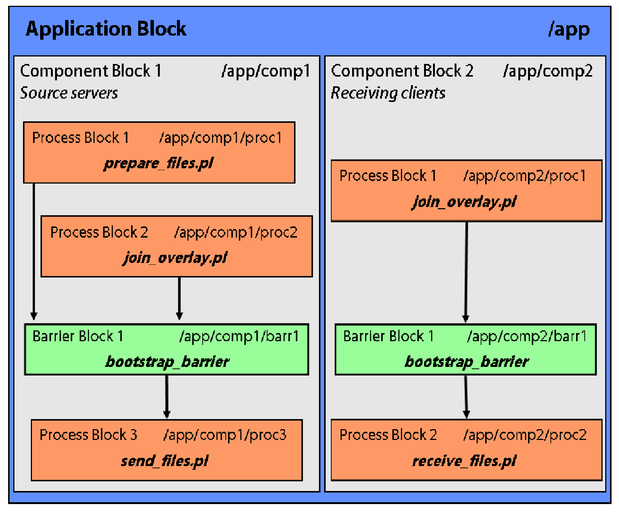
- Plush XML document that describes the distributed application
- Users generate the XML manually or using a GUI

```
<?xml version="1.0" encoding="utf-8"?>
<plush>
 <application name="simple_application">
  <software name="simple_soft" type="tar">
   <package name="software.tar" type="web">
    <path>http://plush.ucsd.edu/software.tar</path>
    <dest_path>software.tar</dest_path>
   </package>
  </software>
  <component name="cluster_1">
   <software name="simple_soft"/>
   <rspec>
    <num_hosts>10</num_hosts>
   </rspec>
   <resources>
    <resource type="planetlab" group="ucsd_plush"/>
   </resources>
  </component>
  <experiment name="simple_exp">
   <execution>
    <component_block name="comp_block1">
     <component name="cluster_1" />
     <process_block name="proc_block1">
      <process name="process1">
       <path>cat</path>
       <cmdline>
        <arg>software.txt</arg>
       </cmdline>
      </process>
     </process_block>
    </component_block>
   </execution>
  </experiment>
 </application>
</plush>
```

- Specify a name for *application*
- Describe *software* packages
- Define a group of *resources* using a *component*
- Describe resource specification using *rspec*
- Define which set of *resources* to use
- Combine a *process_block* with a previously defined *component* in a *component_block*
- Describe an execution using a *process_block* containing one or more *process* elements

Application Building Blocks

- Plush application descriptions are comprised of different types of blocks: **application_blocks**, **component_blocks**, **process_blocks**, and **barrier_blocks**
- This simple file distribution application consists of two groups of resources: source servers and receiving clients
- Servers prepare files for transfer, join the overlay network, and wait for clients to join the overlay before sending files
- Clients join the overlay and wait for servers to prepare files and join the overlay before receiving files
- Barriers synchronize processes and components



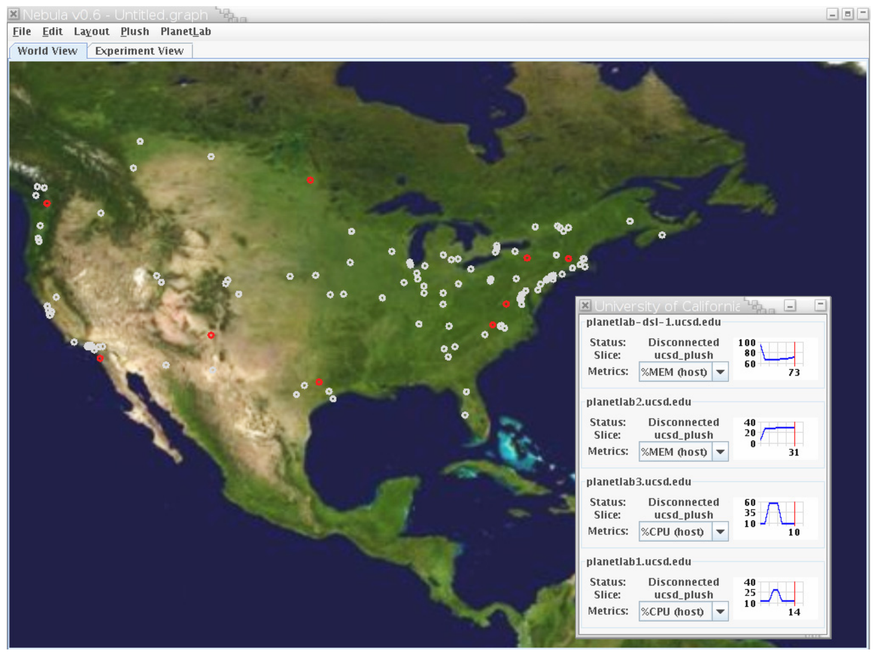
Using the Plush Terminal

- Plush users can interact with Plush via a shell-like terminal interface
- The table below shows some basic Plush terminal commands

Command	Description
load <filename>	Read an XML app description
connect <hostname>	Start and connect to a Plush client on a remote host
disconnect	Close all open client connections
info control	Print the controller's state information
run	Start executing the application
shell <quoted string>	Run "quoted string" as a shell command on all hosts

Visualizing Plush Applications on PlanetLab

- Nebula is a GUI for running applications with Plush
- Users can visualize Plush applications running on PlanetLab in real time
- Colored dots on the map below indicate available sites (grey), and running sites (red)



Research Contributions

- A high-level specification language for distributed computations that captures the requirements of a broad range of applications
- Extensibility to support a range of mechanisms for resource discovery and creation, resource acquisition, software installation, and application control
- A unified framework for distributed application design, deployment, and visualization