**Distributed Application Management Using Plush**
Jeannie Albrecht, Ryan Braud, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat
University of California, San Diego
(jalbrecht, rbraud, ctuttle, snoeren, vahdat) @ cs.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 and allocate resources based on user's 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.

---

### 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 (and partial barriers) are used to provide synchronization across processes and components.

---

### 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/Arkose for efficient service deployment

---

### 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.

---

### Visualizing Plush Applications on PlanetLab

- Plush has integrated with Nebula to provide a GUI for running applications.

- Users can visualize Plush applications running on PlanetLab in real time.

- Colored dots on the map below indicate available hosts (red), running hosts (blue), and selected hosts (green).

---

### Application Description

- Plush XML document that describes the distributed application.

- Users generate the XML manually or using a GUI.

- Specify a name for the application.

- Describe software packages.

- Define a group of resources using a component.

- Describe resource specification using rspec.

- Define which set of resources to use.

- Combines a process_block with a previously defined component in the component_block.

- Describe the execution using a process_block which contains one or more process elements.

---

### Research Contributions

- A high-level specification language for distributed computations that captures the requirements of a broad range of applications.

- Extensibility from the ground up 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.