IdP 2 Clustering

Failover, Higher Availability, and Load Balancing

SWITCH

Serving Swiss Universities

Chad La Joie
chad.lajoie@switch.ch

Terminology

- **Node** - an instance of the IdP
- **Cluster** - a collection of nodes acting as a single, logical IdP
  - The number of nodes in a cluster need not correlate to the number of physical machines in the cluster
- **Failover** - detection of a node failure and redirection of work to an operational node
  - Very easy to accomplish and rarely requires any change to service
- **High(er) Availability** - failover without loss of existing operational data (e.g. session)
  - In stateless systems failover == high availability, the IdP however is not stateless
- **Load Balancing** - intelligent distribution of incoming work amongst available nodes

Techniques: DNS Round Robin

- How it works
  - Give each node in the cluster the SAME DNS name of the logical IdP
  - As hosts resolve the name the DNS gives the next IP in the list
- Pros
  - Easy to setup
- Cons
  - Poor failover heuristics - clients usually cache DNS results for a period of time
  - Poor load balancing - clients can “clump up” on a single node, or a couple nodes
  - Nodes are constantly churning data in the replicated data pool
  - Can not run multiple instances of the service on a single node, under the same name, but on different ports
- The Shibboleth team **strongly** discourages this approach

Techniques: Hardware Load Balancer

- How it works
  - Hardware sits “in front” of the cluster and assumes the name of the logical IdP
  - Incoming requests are forwarded to the IdP cluster nodes
- Pros
  - Guaranteed failover, load-balancing, and high-availability heuristics
  - Possible increase of security of each IdP node
  - Hardware can be used for more than just the IdP
- Cons
  - More difficult to setup
  - Requires purchase of hardware
  - Add additional complexity to setup
- Shibboleth team recommends this approach

IdP Clustering

- Current documented method for clustering is through the use of Terracotta
  - Terracotta is a small client/server model that replicates heap bytes between JVMs
- 3 step setup
  - Configure your firewall (open ports: 9510, 9530)
  - Edit Terracotta config ($IDP_HOME/conf/tc-config.xml) to list each node in cluster
  - Install Terracotta
- Cluster Startup
  - Start Terracotta Servers
    - At least one server must be started before the nodes are started
  - Start IdP Nodes

Recommended Deployment Model

- Install and configure a hardware load balancer
- Install Terracotta on each IdP node
- Run a Terracotta server on each IdP node
  - This eliminates Terracotta as a single point of failure
  - If a Terracotta server goes down the remaining ones will elect a new master server