PLANETLAB

Dynamic Slice Creation

The PlanetLab Architecture Team

Editor: Larry Peterson
Princeton University

Editor: Amin Vahdat
Duke University

PDN-02-005
October 2002

Status: Ongoing Draft.

Dynamic Slice Creation

The PlanetLab Architecture Team

December 17, 2002

1 Introduction

Slices are a central abstraction in PlanetLab. This document establishes the vocabulary needed to talk about slices, and defines a high-level architecture for how slices are dynamically created on behalf of services. This is a working document that we expect to eventually result in a set of interfaces.

2 Terminology

Service: A set of distributed and cooperating programs delivering some higher-level functionality to end-users, other services, or PlanetLab itself. Services that provide functionality needed by PlanetLab are called *infrastructure services*. The individual program component running on a single node is sometimes called a *capsule*. Examples services include peer-to-peer storage systems, content distribution networks, routing overlays, and resource monitoring (the latter is an example of an infrastructure service). Each service runs in a *slice* of PlanetLab.

Slice: A horizontal cut of global PlanetLab resources allocated to a given *service*. A slice encompasses some amount of processing, memory, storage, and network resources across a set of individual PlanetLab nodes distributed over the network. A slice is more than just the sum of the distributed resources, however. It is more accurate to view a slice as a network of *virtual machines*, with a set of local resources bound to each virtual machine.

Virtual Machine: The environment where a capsule that implements some aspect of a service runs. Each virtual machine runs on a single node and is allowed to consume some fraction of that node's resources. We refer to the set of local resources allocated to a virtual machine as a *sliver* of that node. In addition to being bound to a sliver, a virtual machine also defines the programming interface (execution environment) to which capsules are written. Multiple virtual machines run on each PlanetLab node, where a virtual machine monitor (VMM) arbitrates the node's resources among them.

Principal: A user authorized to execute services on PlanetLab. Each principal has some amount of resource privileges, that is, the right to request (and consume) some minimum portion of the global PlanetLab resources. In some cases, there might be a one-to-one relationship between principals and services. In other cases, a principal might correspond to a PI at a site, that in turn divides its share of resources hierarchically among multiple services. Similarly, multiple principals might collaborate to temporarily fund a service.

Ticket: A credential issued by a node or an agent trusted by a node. A ticket specifies resource amounts (i.e., token bucket specifications for cycle and link bandwidth, and limits on memory and disk usage), a node, and a time frame. The bearer of a ticket can redeem it at the node for a *lease* on the resources over the given time frame, subject to the node's admission control policy. It is possible to create "best effort" tickets that can be redeemed for a fair share of the surplus resources on the node. Some tickets may expire if not redeemed within a short time after they are issued.

3 Operational Overview

There are three pieces of information needed to decide whether to create a slice (admit a service). One is a statement of the service's requirements. This information resides with the services. The second is an accurate picture of the current state of each node (e.g., it's available capacity). This information is most accurately known at individual nodes. The third is knowledge about how many resources the service has a right to consume. This is a policy statement, which has both a local and a global component. The local component allows each site to express that some fraction of its local resources are allocated to services that are important to users at the site. The global component of the policy allows PlanetLab as a whole to ensure that infrastructure services, along with other "valued" services, receive sufficient slices, and that all other services receive a fair share.

Five system components are involved in dynamically creating slices and launching services within them.

Admission Control: A privileged component running on each node, and implemented as part of the VMM. It takes a set of tickets as input, and determines if the tickets can be redeemed based on its knowledge of the locally available resources. The ticket set is signed by an authority that the node trusts to enforce the global allocation policy. If the request can be satisfied, the specified resources are reserved, a virtual machine is created and bound to those resources, and a lease is returned. This lease is later used by the service manager to launch a capsule within the virtual machine. (Note that creating a virtual machine involves additional steps that are unrelated to admission control, for example, installing a set of authorized keys needed to launch the service within the VM.)

Resource Monitor: An infrastructure service running on each node. It monitors resource availability on the node and periodically reports the results to one or more resource brokers. Although it is possible that there will eventually be many resource monitors, planet-lab.org provides a default monitoring service. Resource monitors depend on an interface exported by the virtual machine that allows them to record certain facts about the state of the node; e.g., the current load.

Agent: Collects resource availability information from a set of resource monitors or other agents, and issues tickets that can be used to acquire resources on the monitored nodes. An agent can respond to two kinds of queries. First, it can *advertise* the tickets it is holding that meet certain criteria. Second, it can *grant* tickets themselves to a requester. Although there may eventually be a hierarchy of agents, a default agent is initially provided by planet-lab.org.

Resource Broker: May run as a service in PlanetLab, or exist outside of PlanetLab. A resource broker responds to queries from service managers trying to discover a slice to run in. Each query describes the resources needed by the service, and specifies the principal on whose behalf the request is being made. Given a query, the broker first contacts one or more agents to discover what tickets it is possible to obtain. It then matches this information with the service's resource requirements to produce a slice specification, contacts the relevant agents to obtain the tickets, and returns them to the service manager.

Service Manager: One per service, generally running outside of PlanetLab. It contacts a resource broker to discover a slice and obtain the tickets needed to instantiate it. It then submits the tickets to the admission control mechanism on each node to create a network of virtual machines. Should virtual machine creation succeed on each node, the service manager then launches the service, that is, loads and starts capsules in each virtual machine. Admission control returns a lease on the slice. The manager must periodically renew this lease.

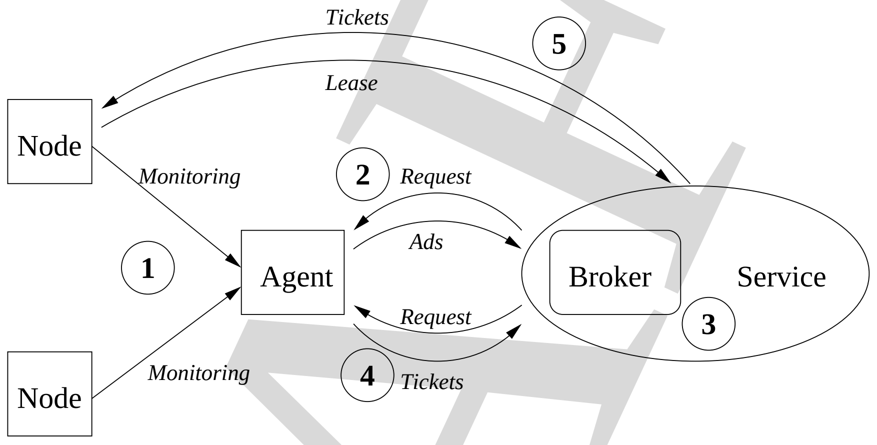

Figure 1: Acquiring a Slice

Figure 1 shows one picture of how these components might interact. Step 1 shows resource monitoring information flowing from a set of nodes to their trusted agent. At step 2, a resource broker running within the service manager requests a description of tickets held by the agent, and the agent responds with a set of advertisements. At step 3, the broker combines the advertisements with the known service requirements to produce a slice specification. The broker requests the tickets for the slice from the agent at step 4 and the agent replies with the tickets. Finally, at step 5 the service presents the tickets to the node on which they are good; the node instantiates a virtual machine bound to the resources and returns a lease to this VM to the service.

Note that this is a simple scenario. Our architecture allows for agent hierarchies in which information is aggregated upward toward "meta-agents", with each agent advertising information to other agents based on arbitrary policies. It is also possible that a node might advertise information about some fraction of its resources to one agent (e.g., an agent that promises to solicit services that the node wants to run) and some fraction of its resources to another agent (e.g., an agent that gives tickets to experimental services). Additionally, in our example the broker is part of the service manager, but in principle, multiple independent brokers can compete by providing specialized slice-matching algorithms.

4 Discussion

The architecture just described provides a great deal of latitude. This section discusses some of the issues that we anticipate as the architecture is reduced to a working implementation.

4.1 Admission Control

We envision a node administrator dividing its resources into two pools: those set aside for services selected by the local site (these are typically services that are of value to users at the site), and those set aside for all other services (these are typically consumed by experimental serivces). The latter resources are most likely advertised through agents as described above. It is not specified how the former resources are made available to the services; for instance, the node could run a local agent that grants tickets directly to a preferred service's resource broker rather than advertising them.

The final decision to honor a ticket is always made by the node's admission controller based on current resource usage and local policies. Refusing tickets leads to coordination problems between the nodes that have the resources and the services that want to use them. However, the PlanetLab architecture allows latitude in how responsibility is shared between agents, brokers, and admission control mechanisms. It should be possible to exploit this flexibility in order to make ticket refusal an uncommon event.

Tickets may be refused because the agent issued them based on stale information from the resource monitor. One way to keep the agent and admission control components in sync is for the agent to confirm that the allocation actually took place when it next hears from the resource monitor running on that node. Another approach is for the agent to interject itself between the service manager and the admission control mechanism. That is, the agent itself can invoke the admission control mechanism on each node, and return the resulting VM lease. A third approach would be for the node to delegate its admission control decision to the agent by unconditionally accepting all requests approved by the agent.

A node may also refuse tickets because honoring them would violate some local or global policy. A service manager can ensure that its request complies with global policy by having its ticket set signed by an authority that enforces this policy (e.g., planet-lab.org). On the other hand, the node's agent can implement the node's local policies by not advertising or issuing tickets to certain brokers. For example, the agent could decide not to advertise tickets for a particular node to a broker representing service X, because it knows that the node will not allow X to run there. The key is that ultimate authority over policy resides with the local node, but this information can be propagated up through a hierarchy of agents to make ticket refusal infrequent.

4.2 Resource Monitor/Agent

The interface between a resource monitor and its trusted agent is private. However, at a minimum the reports generated by the resource monitor running on node should include the following three vectors:

Maximum Capacity: A resource vector specifying the maximum capacity of the node. The link bandwidth element is set by the local administrator (see above); the other elements correspond to the actual capacity of the machine.

Free Capacity: A resource vector specifying the fraction of the global pool that is not currently allocated.

Current Load: A utilization vector specifying the measured load of the system. Unlike the previous two vectors, this one reports the measured state of the node, and so includes the impact of best effort services.

The agent for the node collects these vectors from its resource monitor. The agent then generates tickets based on the node's free capacity, or in the case of best effort tickets, on the current load. Since these tickets are based on observations that could be stale, a ticket must always be regarded as a strong hint rather than an entitlement to resources on the node.

Agents can implement policies by choosing the manner in which it advertises resources and issues tickets. As part of the price of participating in PlanetLab, the node's agent is required to issue some number of tickets to PlanetLab's global agent. PlanetLab then distributes these tickets to brokers in accordance with its own global policy.

4.3 Resource Broker

The primary job of a resource broker is to obtain information about resource availability from agents and to match it with the slice specification provided by a service manager. There may be many brokers, each providing different algorithms for selecting slices. An example of a simple matching algorithm may be to choose the first N nodes that can provide a minimum amount of resources; a sophisticated broker may do matching based on complex specifications such as “three machines at sites A, B, C interconnected by at least 10Mbps and with at least 256MB of memory each”. The more general the request, the more likely it is to be satisfied by the resource broker. How service managers specify their requirements, and how resource brokers satisfy them, is an active research area for the PlanetLab community.

4.4 Service Managers

Service managers face two issues due to the decoupling of slice selection, slice creation, and launching a service in PlanetLab. First, slice selection is done at the global level by a resource broker, while slice creation occurs at the local level when the service manager redeems tickets for a VM instantiation. This means that it is up to the service manager to decide how to deal with partial success in creating the set of virtual machines needed to implement the slice. Second, slices are typically created by a service manager that then launches a service within the slice. However, it is also possible that some other entity creates the slice (e.g., it happens at boot time according to some local policy) and the service later launches itself in the slice. The latter case requires that the creator of the slice give the service a lease for the slice through some out-of-band mechanism.

Contributors

This document is edited by Larry Peterson and Amin Vahdat, with contributions from Tom Anderson, Andy Bavier, Mic Bowman, Brent Chun, Timothy Roscoe, Mike Wawrzoniak, and John Wroclawski. Comments should be sent to

planetlab-slices@lists.sourceforge.net.