The Eucalyptus Open-source Cloud-computing System

Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov

Jeff Bezanson, MIT (6.897 Spring 2011)

Eucalyptus is an open source cloud-computing system (and now an open source company). This means it is a piece of data center software that handles the process of provisioning hardware resources in response to requests from users. Eucalyptus provides infrastructure-as-a-service (IaaS), so you get to upload disk images and access the hardware at a low level by running a VM.

Requests are made using a web service API; the client requests a URL or POSTs an XML document with request parameters such as hardware architecture, memory and storage capacity, available connectivity, and geographic location. The server returns an XML document describing the results (e.g. names of the machines). The Eucalyptus authors chose to implement Amazons EC2 API, so the two services are compatible. There are some other cloud APIs, such as vCloud from VMWare. They seem to be fairly similar in functionality, but of course have different names for everything and different XML schema.

The authors of the paper have a background in grid computing, so they highlight what they see as the differences between grid and cloud. The cloud is designed to support a large number of users, while grid computing was meant to run very large jobs for a small number of users. Cloud computing involves selecting a particular provider and running in their datacenter(s), while grid computing involved a federation of multiple organizations (e.g. universities). It is interesting to consider what we gain and lose in each of these models.

Eucalyptus must handle five basic tasks to do its job: (1) scheduling VM instances, (2) storing disk images and user data, (3) constructing virtual networks, (4) defining and executing service-level agreements (SLAs), and (5) providing user and administrative interfaces. These jobs are split among four components, each its own web service:

- Node Controller: executes VMs and monitors resources on one node. Supports runInstance and terminateInstance calls.
- Cluster Controller: manages groups of nodes, sets up virtual network. Supports runInstances, describeInstances, etc. Collects resource information from node controllers and tells them to run VMs as appropriate, reports back to the cloud controller about how many instances it was able to schedule.
- Storage Controller: key-value store, S3 semantics, lock-free
- Cloud Controller: user and administrator UIs, API and web interface

Administrators can choose whether to treat Eucalyptus-managed VMs like other machines on their network, or to let Eucalyptus set up an isolated virtual network for each set of VMs. Users can declare networks and attach VMs to them. All VMs get private IP addresses, and users can request public IPs for VMs as needed. NAT is used to get to other nodes. The Cloud Controller acts as a router between virtual networks, using iptables to implement user-requested traffic restrictions.

The Cloud Controller tracks the systems resource state to estimate whether user SLAs can be satisfied. On certain events, such as timer expirations, requests are evaluated. The platform might then reject requests or make changes to accommodate them. SLAs can also express requests like using a particular cluster, or using the emptiest cluster. The paper doesnt have a lot of detail on this.

Questions to ponder: Is there a cloud API war? Now that EC2 has multiple implementations, are we likely to see it emerge as a de-facto standard? Is scheduling at the granularity of VMs the final answer on resource sharing?