Synnefo: A Complete Cloud Stack over Ganeti

VANGELIS KOUKIS, CONSTANTINOS VENETSANOPOULOS, AND NECTARIOS KOZIRIS

Vangelis Koukis is the technical lead of the ~okeanos project at the Greek Research and Technology Network (GRNET). His research interests include large-scale computation in the cloud, high-performance cluster interconnects, and shared block-level storage. Koukis has a Ph.D. in Electrical and Computer Engineering from the National Technical University of Athens. vkoukis@grnet.gr

Constantinos Venetsanopoulos is a cloud engineer at the Greek Research and Technology Network. His research interests include distributed storage in virtualized environments and large-scale virtualization management. Venetsanopoulos has a diploma in Electrical and Computer Engineering from the National Technical University of Athens. cven@grnet.gr

Nectarios Koziris is a Professor in the Computing Systems Laboratory at the National Technical University of Athens. His research interests include parallel architectures, interaction between compilers, OSes and architectures, OS virtualization, large-scale computer and storage systems, cloud infrastructures, distributed systems and algorithms, and distributed data management. Koziris has a Ph.D. in Electrical and Computer Engineering from the National Technical University of Athens. nkoziris@cslab.ece.ntua.gr

Synnefo is a complete open source cloud stack that provides Compute, Network, Image, Volume and Storage services, similar to the ones offered by AWS. Synnefo manages multiple Ganeti [2] clusters at the backend for the handling of low-level VM operations. Essentially, it provides the necessary layer around Ganeti to implement the functionality of a complete cloud stack. This approach enforces clear separation between the cluster management layer and the cloud layer, a distinction that is central to Synnefo's design. This separation allows for easier upgrades without impacting VM stability, improves scalability, and simplifies administration. To boost third-party compatibility, Synnefo exposes the OpenStack APIs to users. We have developed two stand-alone clients for its APIs: a rich Web UI and a command-line client.

In this article, we describe Synnefo's overall architecture, its interaction with Ganeti, and the benefits of decoupling the cloud from the cluster layer. We focus on Synnefo's handling of files, images, and VM volumes in an integrated way and discuss advantages when choosing Synnefo to deliver a private or public cloud. We conclude with our experiences from running a real-world production deployment on Synnefo.

Layers

Before describing Synnefo itself in more detail, we will talk about the five distinct layers we recognize in building a complete cloud stack, from the lowest level, closest to the machine, to the highest level, closest to the user:

The *VM-hypervisor* layer is a single VM as created by the hypervisor. The *node* layer represents a number of VMs running on a single physical host. The software on this layer manages the hypervisor on a single physical node and the storage and network visible by the node and sets them up accordingly for each VM. The *cluster* layer is responsible for managing a number of physical nodes, with similar hardware configuration, all managed as a group. The software on this layer coordinates the addition/removal of physical nodes, allows for balanced allocation of virtual resources, and handles live VM migration. The *cloud* layer manages a number of clusters and also brings the user into the picture. The software on this layer handles authentication, resource sharing, ACLs, tokens, accounting, and billing. It also implements one or more APIs and decides how to forward user requests to potentially multiple clusters underneath. The *API* layer is not an actual software layer but rather is the API specification that should be used by the clients of the cloud platform. Finally, at the highest level, we have the *UI* layer that speaks to the platform's APIs.

Building a cloud stack is a difficult engineering problem because it spans many distinct domains. The task is complicated because it involves two distinct mindsets that meet at the cloud↔cluster boundary. On one side is traditional cluster management: low-level virtualization and OS concepts, processes, synchronization, locking, scheduling, block storage management, network switches/routers, and knowledge that there is physical hardware involved, which fails frequently. On the other side lies the fast-paced world of Web-based development, Web services, rich UIs, HTTP APIs, REST, JSON, and XML.

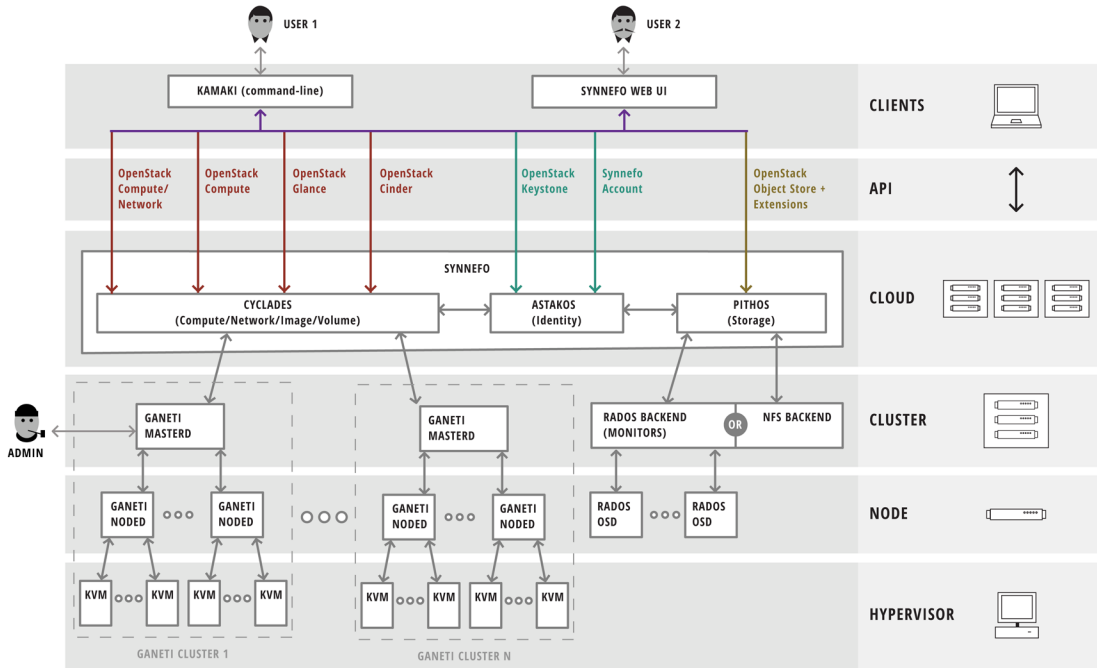

Figure 1: An overview of the Synnefo architecture including all layers

These two sides are served by people with different mindsets and different skill sets. We argue it also is most efficient to be served by different software, keeping a clear separation between the cloud and the cluster layers. Synnefo sits at the cloud layer. We wear a different hat when implementing Synnefo at the cloud layer than when implementing components at the cluster layer that integrate it with Ganeti, or when contributing to Ganeti itself.

Overall Architecture

An overview of the Synnefo stack is shown in Figure 1. Synnefo has three main components:

- Astakos is the common Identity/Account management service across Synnefo.
- Pithos is the File/Object Storage service.
- Cyclades is the Compute/Network/Image and Volume service.

Table 1 provides an explanation for the names we used.

These components are implemented in Python using the Django framework. Each service exposes the associated OpenStack APIs to end users. The service scales out on a number of workers, uses its own private DB to hold cloud-level data, and issues requests to the cluster layer, as necessary.

Synnefo has a number of smaller components that plug into Ganeti to integrate it into a Synnefo deployment.

In the following, we describe the functionality of each main component.

Astakos (Identity)

Astakos is the Identity management component, which provides a common user base to the rest of Synnefo. Astakos handles user creation, user groups, resource accounting, quotas, and projects, and it issues authentication tokens used across the infrastructure. Astakos supports multiple authentication methods: local username/password pairs; LDAP/Active Directory; SAML 2.0 (Shibboleth) federated logins; and login with third-party credentials, including Google, Twitter, and LinkedIn. Users can add

Synnefo	Greek for "cloud," which seemed good for a cloud platform.
~okeanos	Greek for "ocean," an abundant resource pool for life on Earth.
Astakos	Greek for "lobster," a crustacean with big claws and a hard exoskeleton.
Pithos	Ancient Greek name for storage vessels, e.g., for oil or grains.
Cyclades	The main island group in the Aegean Sea.
Kamaki	Greek for "harpoon"; if VMs are fish in the ocean, a harpoon may come handy.
Archipelago	Greek for "a cluster of islands," which seemed good for a distributed storage system.

Table 1: The story behind the names of Synnefo and its components. Many of the names follow a sea theme, as Synnefo's origins are in the ~okeanos service.

multiple login methods to a single account, according to configured policy.

Astakos keeps track of resource usage across Synnefo, enforces quotas, and implements a common user dashboard. Quota handling is resource-type agnostic: resources (e.g., VMs, public IPs, GBs of storage, or disk space) are defined by each Synnefo component independently, then imported into Astakos for accounting and presentation.

Astakos runs at the cloud layer and exposes the OpenStack Keystone API for authentication, along with the Synnefo Account API for quota, user group, and project management.

Pithos (Object/File Storage)

Pithos is the Object/File Storage component of Synnefo. Users upload files on Pithos using either the Web UI, the command-line client, or native syncing clients. Pithos is a thin layer mapping user-files to content-addressable blocks that are then stored on a storage backend. Files are split in blocks of fixed size, which are hashed independently to create a unique identifier for each block, so each file is represented by a sequence of block names (a *hashmap*). This way, Pithos provides deduplication of file data; blocks shared among files are only stored once. The current implementation uses 4 MB blocks hashed with SHA256. Content-based addressing also enables efficient two-way file syncing that can be used by all Pithos clients (e.g., the "kamaki" command-line client or the native Windows/Mac OS clients). Whenever someone wants to upload an updated version of a file, the client hashes all blocks of the file and then requests the server to create a new version for this block sequence. The server will return an error reply with a list of the missing blocks. The client may then upload each block one by one, and retry file creation. Similarly, whenever a file has been changed on the server, the client can ask for its list of blocks and only download the modified ones.

Pithos runs at the cloud layer and exposes the OpenStack Object Storage API to the outside world, with custom extensions for syncing. Any client speaking to OpenStack Swift can also be used to store objects in a Pithos deployment. The process of mapping user files to hashed objects is independent from the actual storage backend, which is selectable by the administrator using pluggable drivers. Currently, Pithos has drivers for two storage backends: files on a shared file system (e.g., NFS, Lustre, or GPFS) or objects on a Ceph/RADOS [3] cluster. Whatever the storage backend, it is responsible for storing objects reliably, without any connection to the cloud APIs or to the hashing operations.

Cyclades (Compute/Network/Image/Volume)

Cyclades is the Synnefo component that implements the Compute, Network, Image, and Volume services. Cyclades exposes the associated OpenStack REST APIs: OpenStack Compute, Network, Glance, and, soon, Cinder. Cyclades is the part that manages multiple Ganeti clusters at the backend. Cyclades issues commands to a Ganeti cluster using Ganeti's Remote API (RAPI). The administrator can expand the infrastructure dynamically by adding new Ganeti clusters to reach datacenter scale. Cyclades knows nothing about low-level VM management operations, e.g., handling of VM creations, migrations among physical nodes, and handling of node downtimes; the design and implementation of the end-user API is orthogonal to VM handling at the backend.

We strive to keep the implementation of Cyclades independent of Ganeti code. We write around Ganeti, and add no Synnefo-specific code inside it. Whenever the mechanism inside Ganeti does not suffice, we extend it independently from Synnefo, and contribute patches to the official upstream for review and eventual inclusion in the project.

There are two distinct, asynchronous paths in the interaction between Synnefo and Ganeti. The *effect* path is activated in response to a user request; Cyclades issues VM control commands to Ganeti over RAPI. The *update* path is triggered whenever the state of a VM changes, due to Synnefo- or administrator-initiated actions happening at the Ganeti level. In the update path, we exploit Ganeti's hook mechanism to produce notifications to the rest of the Synnefo infrastructure over a message queue.

Tying It All Together

Synnefo's greatest strength lies in the integrated way it handles its three basic storage entities: Files, named pieces of user data; Images, the static templates from which live VM instances are initialized; and Volumes, the block storage devices, the virtual disks on which live VMs operate. In this section, we describe the duality between Files and Images (an Image is a file on Pithos that has specific metadata), and the duality between Images and Volumes (a Volume is a live VM disk that originates from an Image).

Images as Files on Pithos

Synnefo uses Pithos to store both system and user-provided Images in the same way it stores all other files. Because Images of the same OS share many identical blocks, deduplication comes in handy. Assume a user has created a "golden" VM Image on her own computer, and has customized it to her liking. When she is ready to deploy it, she uploads it as a file to Pithos, registers it as an Image with Cyclades, then spawns new VMs from it. When she needs to update her Image, she just repeats the process. Every upload uses the Pithos syncing protocol, which means the client will only need to upload the blocks changed since the previous time. Pithos features a file-sharing mechanism, which applies to Image files too: users can attach custom ACLs to them, share them with other users or closed groups, or make them public.

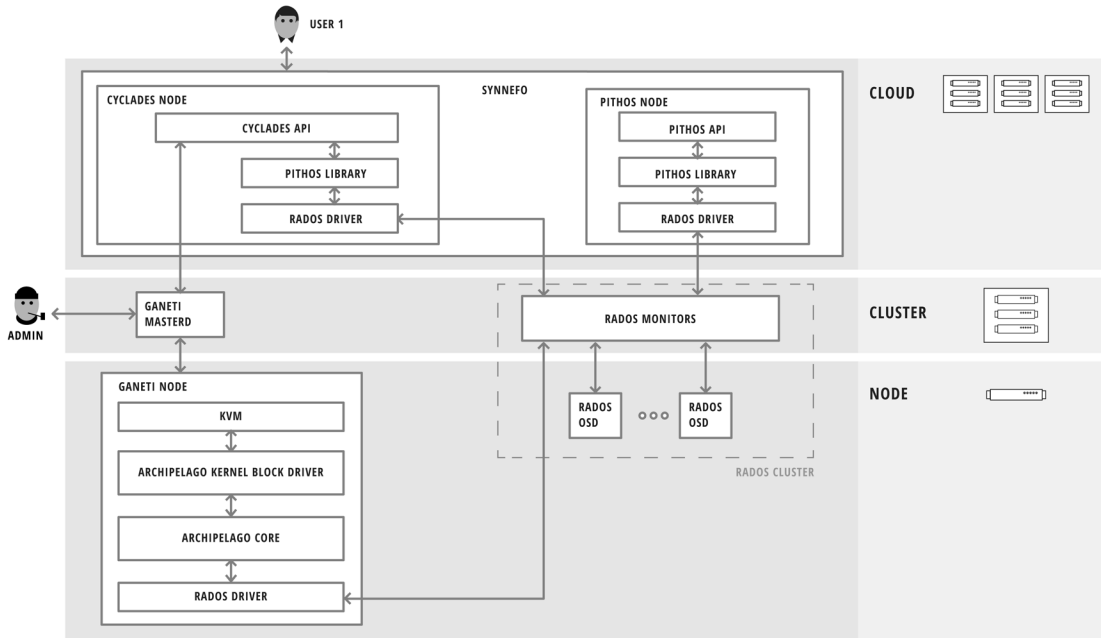

Figure 2: Integrated storage for Images and Volumes with Archipelago

Image Deployment Inside Ganeti

To support the secure deployment of user-provided, untrusted images with Ganeti, we have developed a Ganeti OS definition called *snf-image*. Image deployment entails two steps: (1) Volume initialization—the Image is fetched from backend storage and copied to the block device of the newly created instance, and (2) optional Image customization. Customization includes resizing the root file system, changing passwords for root or other users, file injection (e.g., for SSH keys), and setting a custom hostname. All Image customization is done inside a helper VM, in isolation from the physical host, enhancing robustness and security.

For Volume initialization, snf-image can fetch Image data from a number of storage backends. Volume initialization can use a shared file system (e.g., NFS), perform an HTTP or FTP download, or, in the Synnefo case, contact a Pithos storage backend directly. snf-image can deploy most major Linux distributions (Debian, Ubuntu, CentOS, Fedora, Arch, openSUSE, Gentoo), Windows Server 2008R2 and 2012, as well as FreeBSD.

Archipelago: Integrated Handling of Volumes and Images

Synnefo supports all different storage options ("disk templates") offered by Ganeti to back the virtual disks used by VMs ("Volumes"). Each storage backend has different redundancy and performance characteristics; Synnefo brings the choice of storage backend all the way up to the user, who can select based on the intended usage of the VM.

The Ganeti-provided disk templates are good options for long-running, persistent VMs (e.g., a departmental file server running on the cloud); however, they are not a good fit when the usage scenario needs thin VM provisioning: for example, when the user wants to spin up a large number of short-lived, identical VMs (e.g., from a custom golden Image), run a parallel program for a few hours or days, then shut them down. In this case, the time and space overhead of copying Image data to all Volumes is significant.

Archipelago is a block storage layer developed with Synnefo, which integrates VM Images with Volumes. Archipelago enables thin creation of Volumes as copy-on-write clones of Images, with zero data movement, as well as making snapshots of a Volume at a later time to create VM Images. Archipelago plugs into Ganeti and acts as one of its disk templates. Cyclades then uses Archipelago for fast provisioning of VMs from Images stored on Pithos, with minimal overhead. To implement clones and snapshots, Archipelago keeps track of VM block allocation in maps, initialized from Pithos files (hashmaps). Maps are stored along with actual data blocks. Archipelago can use various storage backends to store data, similarly to Pithos. Archipelago has pluggable drivers, currently for file system-backed block storage, or Ceph/RADOS, so clone and snapshot functionality is independent of the underlying backend. Figure 2 shows how Archipelago is integrated into a Synnefo deployment. In such a scenario, Archipelago shares its storage backend with Pithos. This enables a workflow as follows: a user uploads the contents of an Image as a file on Pithos, with efficient syncing, registers it as an Image

with Cyclades, then spawns a large number of thinly provisioned VMs from this Image. Because Archipelago shares the storage backend with Pithos, it creates one new volume per VM without copying the data. The actual 4 MB blocks of data that make up the Image remain as blocks in the storage backend, after being uploaded to Pithos by the user. Archipelago will create one map per VM, with all maps referencing the original Pithos blocks for the Image. Whenever a VM modifies data on its volume, Archipelago allocates a new block for it and updates the map for its volume accordingly.

Synnefo Advantages

The decoupled design of Synnefo brings the following advantages:

- Synnefo combines the stability of Ganeti with the self-service provisioning of clouds. This allows it to run workloads that do not fit the standard model of a volatile cloud, such as long-running servers in fault-tolerant, persistent VMs. Archipelago-backed storage covers the need for fast provisioning of short-lived, computationally intensive worker VMs.
- In a Synnefo deployment, Synnefo and Ganeti follow distinct upgrade schedules, with software upgrades rolled out gradually, without affecting all of the stack at once.
- The Ganeti clusters are self-contained. The administrator has complete control (e.g., to add/remove physical nodes or migrate VMs to different nodes via the Ganeti side path) without Synnefo knowing about it. Synnefo is automatically notified and updates user-visible state whenever necessary. For example, a VM migration happening at Ganeti level is transparent to Synnefo, whereas a VM shutdown by the admin will propagate up to the user.
- The system scales dynamically and linearly by adding new Ganeti clusters into an existing installation. Heterogeneity across clusters allows Synnefo to provide services with different characteristics and levels of QoS (e.g., virtual-to-physical CPU ratio).
- Two-level allocation policy for VMs with different criteria: at the cloud layer, Synnefo selects a Ganeti cluster according to high-level criteria (e.g., QoS); at the cluster layer, Ganeti selects a physical node based on lower-level criteria (e.g., free RAM on node).
- There is no single database housing all VM configuration data. Low-level state is handled separately in each Ganeti cluster. Physical nodes have no access to the Cyclades database at the cloud layer. This minimizes the possible impact of a hypervisor breakout and simplifies hardening of DB security.
- Out–of-the-box integration with different storage backend technologies, including File, LVM, DRBD, NAS, or Archipelago on commodity hardware.

Running in Production

Synnefo has been running in production since 2011, powering GRNET's ~okeanos [1] public cloud service. Synnefo's development team has grown to more than 15 people in the past three years. As of this writing, ~okeanos runs more than 5,000 active VMs, for more than 3,500 users. Users have launched more than 100,000 VMs and more than 20,000 virtual networks.

Using Synnefo in production has enabled:

- Rolling software and hardware upgrades across all nodes. We have done numerous hardware and software upgrades (kernel, Ganeti, Synnefo), many requiring physical node reboots, without user-visible VM interruption.
- Moving the whole service to a different datacenter, with cross-datacenter live VM migrations, from Intel to AMD machines, without the users noticing.
- On-the-fly syncing of NFS-backed Pithos blocks to RADOS-backed storage, and integration with Archipelago for thin VM provisioning.
- Scaling from a few physical hosts to multiple racks with dynamic addition of Ganeti backends.
- Overcoming limitations of the networking hardware regarding number of VLANs. Ganeti provides for pluggable networking scripts, which we exploit to run thousands of virtual LANs over a single physical VLAN with MAC-level filtering, in a custom configuration. We have also tested VXLAN-based network encapsulation, again with no code modifications to Ganeti or Synnefo.
- Preserving the ability to live migrate while upgrading across incompatible KVM versions, by maintaining the virtual hardware configuration independently.

Synnefo is open source. Source code, distribution packages, documentation, many screenshots and videos, as well as a test deployment open to all can be found at http://www.synnefo.org.

References

[1] Vangelis Koukis, Constantinos Venetsanopoulos, and Nectarios Koziris, "~okeanos: Building a Cloud, Cluster by Cluster," *IEEE Internet Computing*, vol. 17, no. 13, May-June 2013, pp. 67-71.

[2] Guido Trotter and Tom Limoncelli, "Ganeti: Cluster Virtualization Manager," USENIX *;login:*, vol. 38, no. 3, 2013.

[3] Sage A. Weil, Andrew W. Leung, Scott A. Brandt, and Carlos Maltzahn, "RADOS: A Scalable, Reliable Storage Service for Petabyte-Scale Storage Clusters," *Proceedings of the 2nd International Workshop on Petascale Data Storage*, PDSW '07, held in conjunction with Supercomputing '07 (ACM, 2007), pp. 35-44.