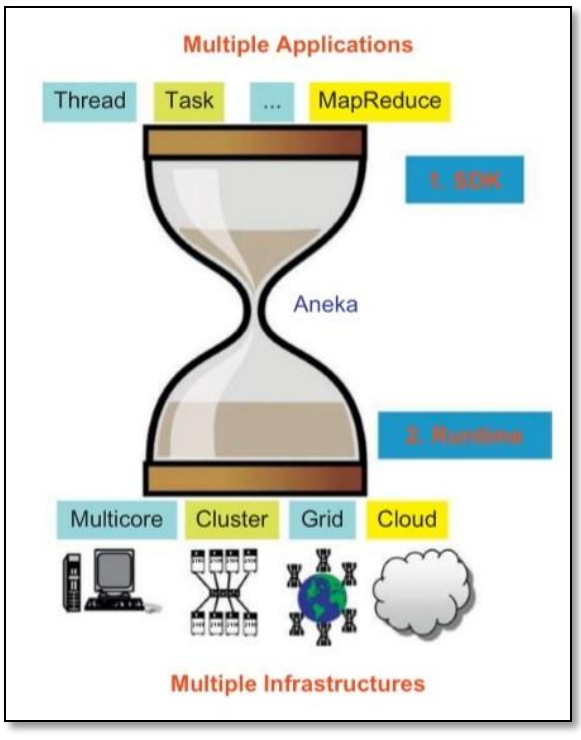
ANEKA: CLOUD APPLICATION PLATFORM

Aneka is Manjrasoft Pty. Ltd.'s solution for developing, deploying, and managing cloud applications. Aneka consists of a scalable cloud middleware that can be deployed on top of heterogeneous computing resources. It offers an extensible collection of services coordinating the execution of applications, helping administrators monitor the status of the cloud, and providing integration with existing cloud technologies. One of Aneka's key advantages is its extensible set of application programming interfaces (APIs) associated with different types of programming models - such as **Task, Thread, and MapReduce -** used for developing distributed applications, integrating new capabilities into the cloud, and supporting different types of cloud deployment models: public, private, and hybrid. These features differentiate Aneka from infrastructure management software and characterize it as a platform for developing, deploying, and managing execution of applications on various types of clouds.



Aneka is a software platform for developing cloud computing applications. Aneka is a pure **PaaS** solution for cloud computing. Aneka is a cloud middleware product that can be deployed on a heterogeneous set of resources: a network of computers, a multicore server, datacenters, virtual cloud infrastructures, or a mixture of these. The framework provides both middleware for managing and scaling distributed applications and an extensible set of APIs for developing them.

Framework Overview

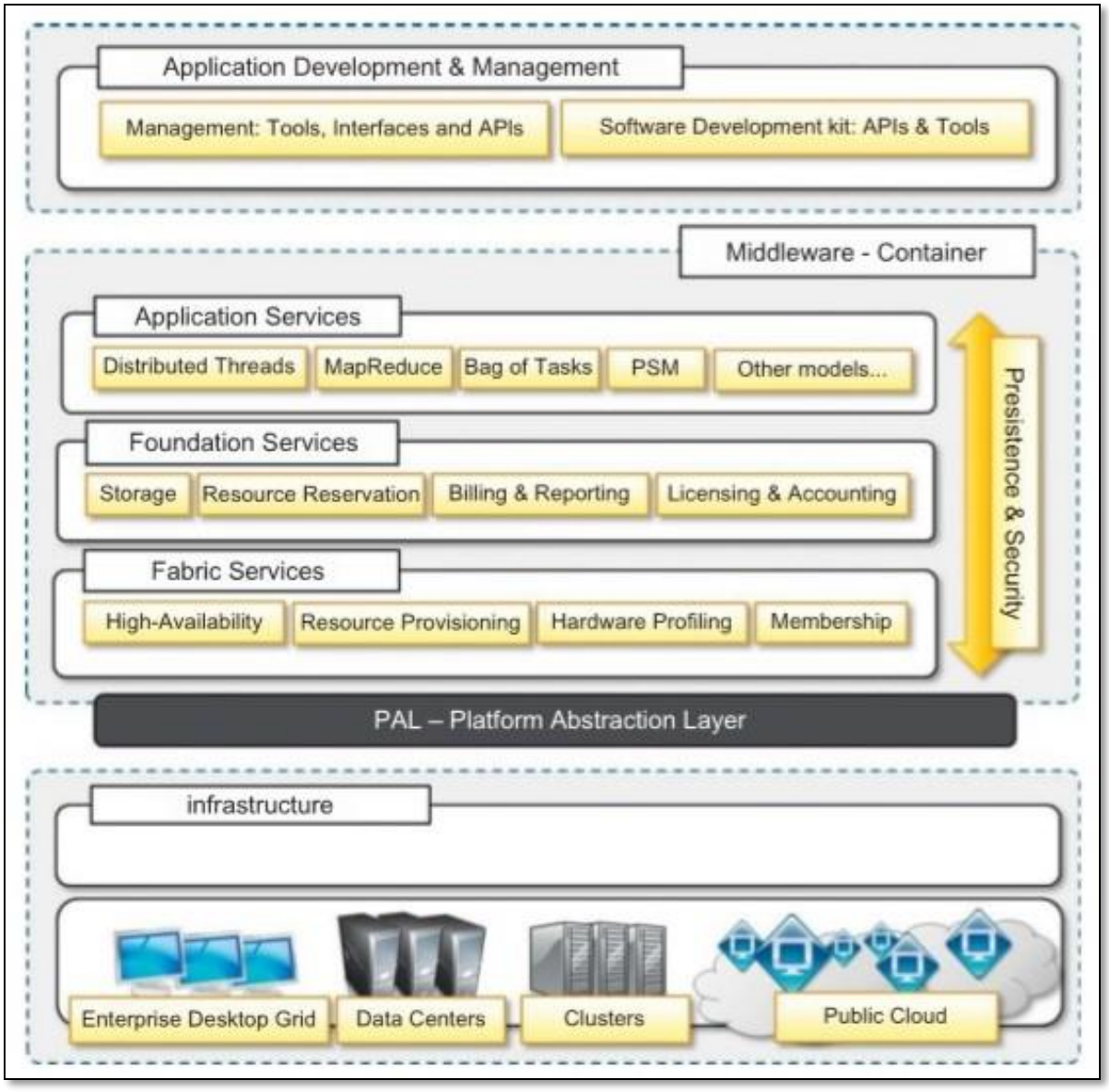


Figure provides a complete overview of the components of the Aneka framework. The core infrastructure of the system provides a uniform layer that allows the framework to be deployed over different platforms and operating systems. The physical and virtual resources representing the bare metal of the cloud are managed by the Aneka container, which is installed on each node and constitutes the basic building block of the middleware. A collection of interconnected containers constitutes the Aneka Cloud: a single domain in which services are made available to users and developers.

The container features three different classes of services: Fabric Services, Foundation Services, and Execution Services. These take care of infrastructure management, supporting services for the Aneka Cloud, and application management and execution, respectively.

These services are made available to developers and administrators by means of the application management and development layer, which includes interfaces and APIs for developing cloud applications and the management tools and interfaces for controlling Aneka Clouds.

Aneka implements a service-oriented architecture (SOA), and services are the fundamental components of an Aneka Cloud. Services operate at container level and, except for the platform abstraction layer, they provide developers, users, and administrators with all features offered by the framework. Services also constitute the extension and customization point of Aneka Clouds: The infrastructure allows for the integration of new services or replacement of the existing ones with a different implementation.

The framework includes the basic services for infrastructure and node management, application execution, accounting, and system monitoring; existing services can be extended and new features can be added to the cloud by dynamically plugging new ones into the container. Such extensible and flexible infrastructure enables Aneka Clouds to support different programming and execution models for applications. A programming model represents a collection of abstractions that developers can use to express distributed applications; the runtime support for a programming model is constituted by a collection of execution and foundation services interacting together to carry out application execution. Thus, the implementation of a new model requires the development of the specific programming abstractions used by application developers and the services, providing runtime support for them. Programming models are just one aspect of application management and execution. Within an Aneka Cloud environment, there are different aspects involved in providing a scalable and elastic infrastructure and distributed runtime for applications. These services involve:

- **Elasticity and Scaling:** By means of the dynamic provisioning service, Aneka supports dynamically upsizing and downsizing of the infrastructure available for applications.
- **Runtime Management:** The runtime machinery is responsible for keeping the infrastructure up and running and serves as a hosting environment for services. It is primarily represented by the container and a collection of services that manage service membership and lookup, infrastructure maintenance, and profiling.
- **Resource Management:** Aneka is an elastic infrastructure in which resources are added and removed dynamically

according to application needs and user requirements. To provide QoS-based execution, the system not only allows dynamic provisioning but also provides capabilities for reserving nodes for exclusive use by specific applications.

- **Application Management:** A specific subset of services is devoted to managing applications. These services include scheduling, execution, monitoring, and storage management.
- **User Management:** Aneka is a multitenant distributed environment in which multiple applications, potentially belonging to different users, are executed. The framework provides an extensible user system via which it is possible to define users, groups, and permissions. The services devoted to user management build up the security infrastructure of the system and constitute a fundamental element for accounting management.
- **QoS / SLA Management & Billing:** Within a cloud environment, application execution is metered and billed. Aneka provides a collection of services that coordinate together to take into account the usage of resources by each application and to bill the owning user accordingly.

ANATOMY OF THE ANEKA CONTAINER

The Aneka container constitutes the building blocks of Aneka Clouds and represents the runtime machinery available to services and applications. The container, the unit of deployment in Aneka Clouds, is a lightweight software layer designed to host services and interact with the underlying operating system and hardware. The main role of the container is to provide a lightweight environment in which to deploy services and some basic capabilities such as communication channels through which it interacts with other nodes in the Aneka Cloud. Almost all operations performed within Aneka are carried out by the services managed by the container. The services installed in the Aneka container can be classified into three major categories:

- Fabric Services
- Foundation Services
- Application Services

The services stack resides on top of the Platform Abstraction Layer (PAL), representing the interface to the underlying operating system and hardware. It provides a uniform view of the software and hardware environment in which the container is running. Persistence and security traverse all the services stack to provide a secure and reliable infrastructure.

The Platform Abstraction Layer

The Platform Abstraction Layer (PAL) addresses this heterogeneity and provides the container with a uniform interface for accessing the relevant hardware and operating system information, thus allowing the rest of the container to run unmodified on any supported platform. The PAL is responsible for detecting the

supported hosting environment and providing the corresponding implementation to interact with it to support the activity of the container. The PAL provides the following features:

- Uniform and platform-independent implementation interface for accessing the hosting platform
- Uniform access to extended and additional properties of the hosting platform
- Uniform and platform-independent access to remote nodes
- Uniform and platform-independent management interfaces

The PAL is a small layer of software that comprises a detection engine, which automatically configures the container at boot time, with the platform-specific component to access the above information and an implementation of the abstraction layer for the Windows, Linux, and Mac OS X operating systems.

The collectible data that are exposed by the PAL are the following:

- Number of cores, frequency, and CPU usage
- Memory size and usage
- Aggregate available disk space
- Network addresses and devices attached to the node

Fabric Services

Fabric Services define the lowest level of the software stack representing the Aneka Container. They provide access to the resource-provisioning subsystem and to the monitoring facilities implemented in Aneka. Resource-provisioning services are in charge of dynamically providing new nodes on demand by relying on virtualization technologies, while monitoring services allow for hardware profiling and implement a basic monitoring infrastructure

that can be used by all the services installed in the container.

Foundation Services

Fabric Services are fundamental services of the Aneka Cloud and define the basic infrastructure management features of the system. Foundation Services are related to the logical management of the distributed system built on top of the infrastructure and provide supporting services for the execution of distributed applications. All the supported programming models can integrate with and leverage these services to provide advanced and comprehensive application management. These services cover:

- Storage management for applications
- Accounting, billing, and resource pricing
- Resource reservation

Foundation Services provide a uniform approach to managing distributed applications and allow developers to concentrate only on the logic that distinguishes a specific programming model from the others. Together with the Fabric Services, Foundation Services constitute the core of the Aneka middleware. These services are mostly consumed by the execution services and Management Consoles. External applications can leverage the exposed capabilities for providing advanced application management.

Application Services

Application Services manage the execution of applications and constitute a layer that differentiates according to the specific programming model used for developing distributed applications on top of Aneka. The types and the number of services that compose

this layer for each of the programming models may vary according to the specific needs or features of the selected model. It is possible to identify two major types of activities that are common across all the supported models: Scheduling and Execution. Aneka defines a reference model for implementing the runtime support for programming models that abstracts these two activities in corresponding services: Scheduling Service and the Execution Service. Moreover, it also defines base implementations that can be extended in order to integrate new models.

BUILDING ANEKA CLOUDS

Aneka is primarily a platform for developing distributed applications for clouds. As a software platform it requires infrastructure on which to be deployed; this infrastructure needs to be managed. Infrastructure management tools are specifically designed for this task, and building clouds is one of the primary tasks of administrators. Aneka supports various deployment models for public, private, and hybrid clouds.

PRIVATE CLOUD DEPLOYMENT MODE

A private deployment mode is mostly constituted by local physical resources and infrastructure management software providing access to a local pool of nodes, which might be virtualized. In this scenario Aneka Clouds are created by harnessing a heterogeneous pool of resources such has desktop machines, clusters, or workstations.

These resources can be partitioned into different groups, and Aneka can be configured to leverage these resources according to application needs. Moreover, leveraging the Resource Provisioning Service, it is possible to integrate virtual nodes provisioned from a local resource pool managed by systems such as XenServer, Eucalyptus, and OpenStack.

Figure 1 shows a common deployment for a private Aneka Cloud. This deployment is acceptable for a scenario in which the workload of the system is predictable and a local virtual machine manager can easily address excess capacity demand. Most of the Aneka nodes are constituted of physical nodes with a long lifetime and a static configuration and generally do not need to be reconfigured often. The different nature of the machines harnessed in a private

environment allows for specific policies on resource management and usage that can be accomplished by means of the Reservation Service.

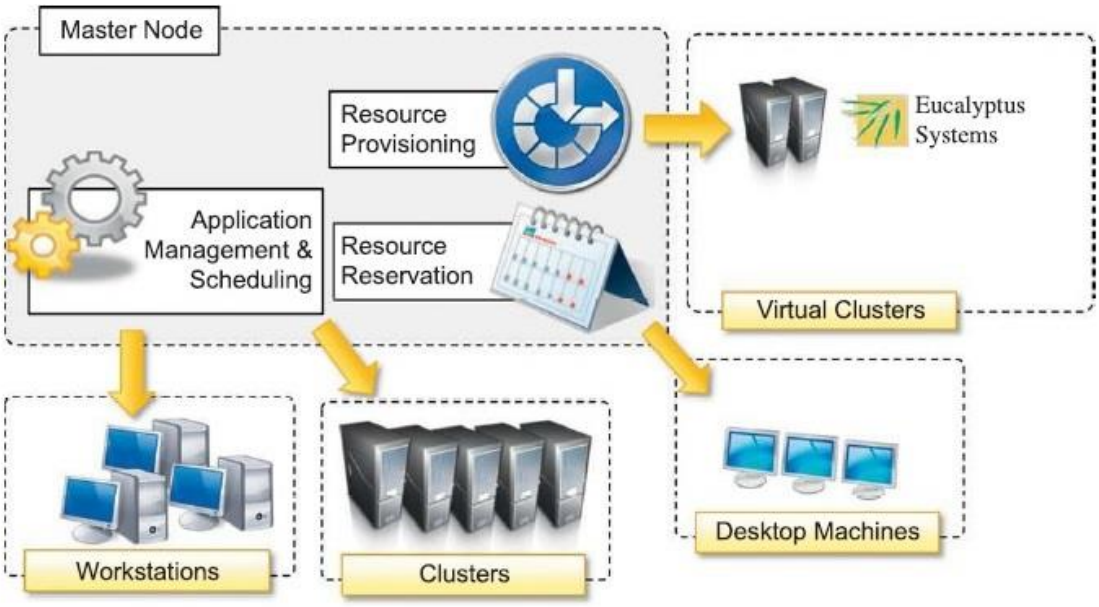


Figure 1: Private Cloud Deployment Mode

For example, desktop machines that are used during the day for office automation can be exploited outside the standard working hours to execute distributed applications. Workstation clusters might have some specific legacy software that is required for supporting the execution of applications and should be preferred for the execution of applications with special requirements.

PUBLIC CLOUD DEPLOYMENT MODE

Public Cloud deployment mode features the installation of Aneka master and worker nodes over a completely virtualized infrastructure that is hosted on the infrastructure of one or more resource providers such as Amazon EC2 or GoGrid. In this case it is possible to have a static deployment where the nodes are provisioned beforehand and used as though they were real

machines. This deployment merely replicates a classic Aneka installation on a physical infrastructure without any dynamic provisioning capability. More interesting is the use of the elastic features of IaaS providers and the creation of a Cloud that is completely dynamic. **Figure 2** provides an overview of this scenario.

The deployment is generally contained within the infrastructure boundaries of a single IaaS provider. The reasons for this are to minimize the data transfer between different providers, which is generally priced at a higher cost, and to have better network performance. In this scenario it is possible to deploy an Aneka Cloud composed of only one node and to completely leverage dynamic provisioning to elastically scale the infrastructure on demand. A fundamental role is played by the Resource Provisioning Service, which can be configured with different images and templates to instantiate. Other important services that have to be included in the master node are the Accounting and Reporting Services. These provide details about resource utilization by users and applications and are fundamental in a multitenant Cloud where users are billed according to their consumption of Cloud capabilities.

Dynamic instances provisioned on demand will mostly be configured as worker nodes, and, in the specific case of Amazon EC2, different images featuring a different hardware setup can be made available to instantiate worker containers. Applications with specific requirements for computing capacity or memory can provide additional information to the scheduler that will trigger the appropriate provisioning request. Application execution is not the only use of dynamic instances; any service requiring elastic scaling can leverage dynamic provisioning. Another example is the Storage

Service. In multitenant Clouds, multiple applications can leverage the support for storage; in this scenario it is then possible to introduce bottlenecks or simply reach the quota limits allocated for storage on the node. Dynamic provisioning can easily solve this issue as it does for increasing the computing capability of an Aneka Cloud.

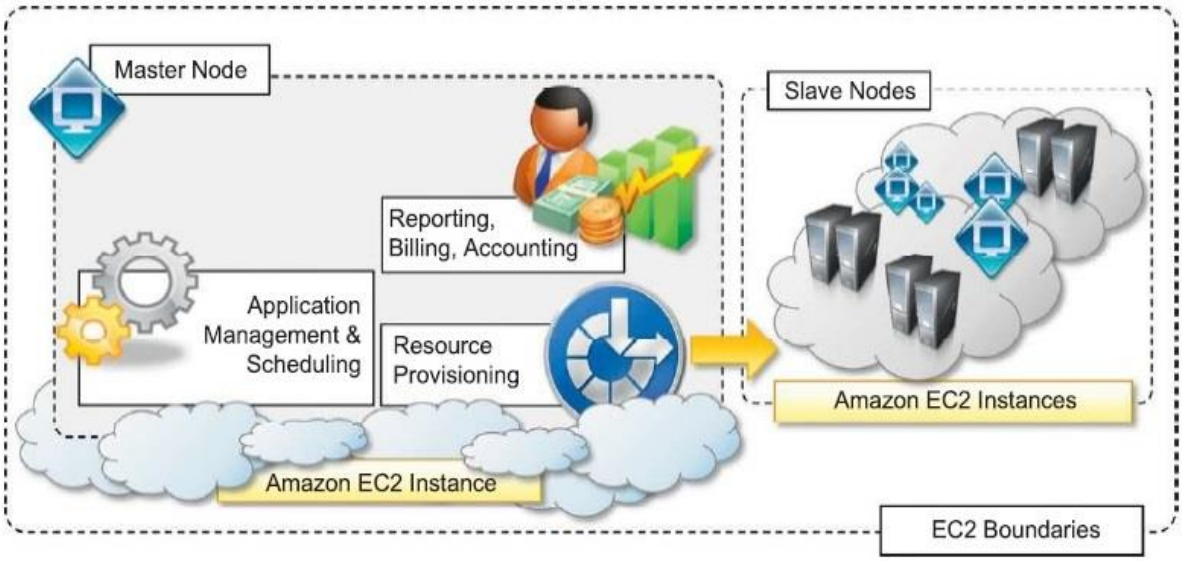


Figure 2: Public Cloud Deployment Mode

Deployments using different providers are unlikely to happen because of the data transfer costs among providers, but they might be a possible scenario for federated Aneka Clouds. In this scenario resources can be shared or leased among providers under specific agreements and more convenient prices. In this case the specific policies installed in the Resource Provisioning Service can discriminate among different resource providers, mapping different IaaS providers to provide the best solution to a provisioning request.

HYBRID CLOUD DEPLOYMENT MODE

The hybrid deployment model constitutes the most common deployment of Aneka. In many cases, there is an existing computing infrastructure that can be leveraged to address the

computing needs of applications. This infrastructure will constitute the static deployment of Aneka that can be elastically scaled on demand when additional resources are required. An overview of this deployment is presented in **Figure 3**.

This scenario constitutes the most complete deployment for Aneka that is able to leverage all the capabilities of the framework:

- Dynamic Resource Provisioning
- Resource Reservation
- Workload Partitioning
- Accounting, Monitoring, and Reporting

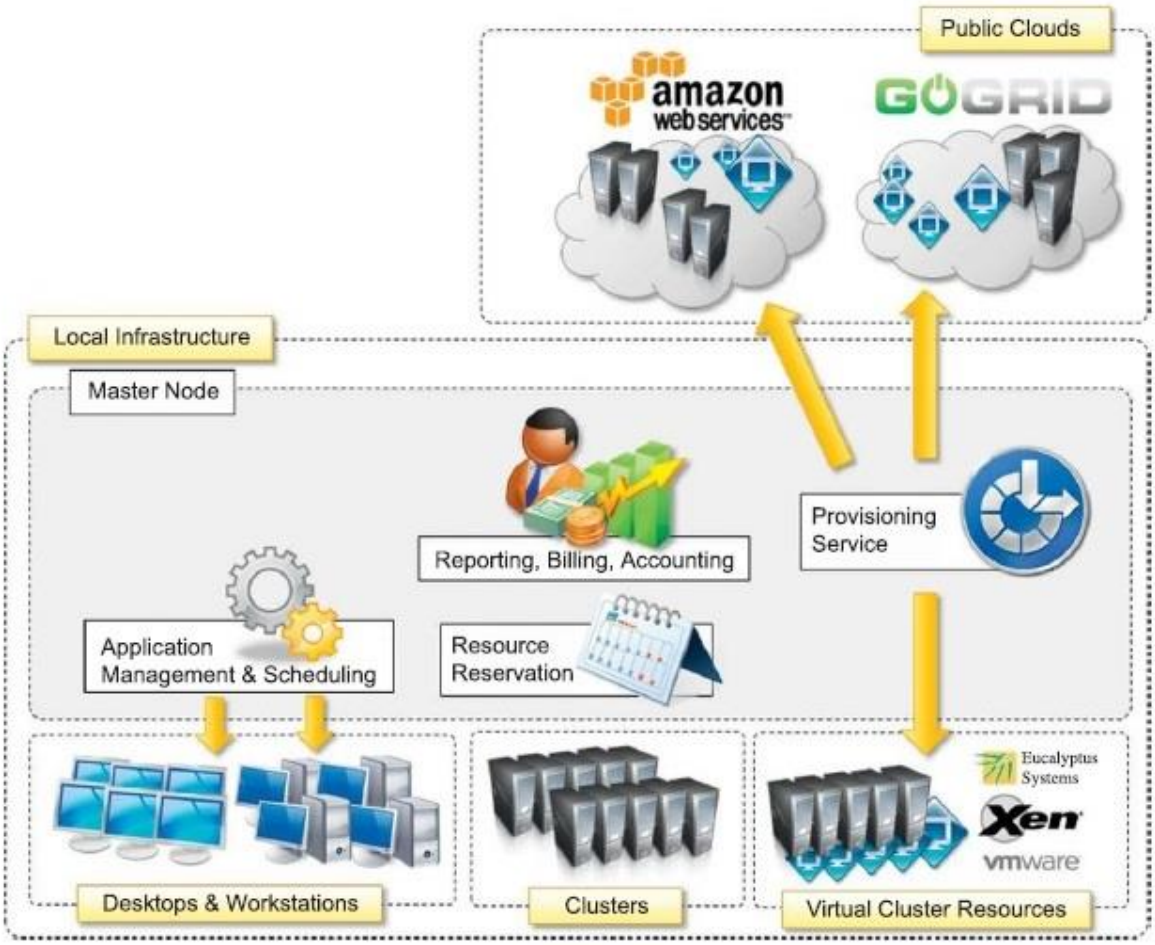


Figure 3: Hybrid Cloud Deployment Mode

Moreover, if the local premises offer some virtual machine management capabilities, it is possible to provide a very efficient

use of resources, thus minimizing the expenditure for application execution.

In a hybrid scenario, heterogeneous resources can be used for different purposes. As we discussed in the case of a private cloud deployment, desktop machines can be reserved for low priority workload outside the common working hours. The majority of the applications will be executed on workstations and clusters, which are the nodes that are constantly connected to the Aneka Cloud. Any additional computing capability demand can be primarily addressed by the local virtualization facilities, and if more computing power is required, it is possible to leverage external IaaS providers.

Different from the Aneka Public Cloud deployment is the case in which it makes more sense to leverage a variety of resource providers to provision virtual resources. Since part of the infrastructure is local, a cost in data transfer to the external IaaS infrastructure cannot be avoided. It is then important to select the most suitable option to address application needs. The Resource Provisioning

Service implemented in Aneka exposes the capability of leveraging several resource pools at the same time and configuring specific policies to select the most appropriate pool for satisfying a provisioning request. These features simplify the development of custom policies that can better serve the needs of a specific hybrid deployment.

CLOUD PROGRAMMING AND MANAGEMENT

Aneka's primary purpose is to provide a scalable middleware product in which to execute distributed applications. Application development and management constitute the two major features that are exposed to developers and system administrators. Aneka provides developers with a comprehensive and extensible set of APIs and administrators with powerful and intuitive management tools. The APIs for development are mostly concentrated in the Aneka SDK; management tools are exposed through the Management Console.

1) Aneka SDK: Aneka provides APIs for developing applications on top of existing programming models, implementing new programming models, and developing new services to integrate into the Aneka Cloud. The development of applications mostly focuses on the use of existing features and leveraging the services of the middleware, while the implementation of new programming models or new services enriches the features of Aneka. The SDK provides support for both programming models and services by means of the **Application Model and the Service Model.** The former covers the development of applications and new programming models; the latter defines the general infrastructure for service development.

a) **Application Model:** The Application Model represents the minimum set of APIs that is common to all the programming models for representing and programming distributed applications on top of Aneka. This model is further specialized according to the needs and the particular features of each of the programming models.

b) **Service Model:** The Aneka Service Model defines the basic

requirements to implement a service that can be hosted in an Aneka Cloud. The container defines the runtime environment in which services are hosted. Each service that is hosted in the container must be compliant with the **IService** interface, which exposes the following methods and properties:

- Name and status
- Control operations such as Start, Stop, Pause, and Continue methods
- Message handling by means of the HandleMessage method

Management Tools

Aneka is a pure PaaS implementation and requires virtual or physical hardware to be deployed. This layer also includes capabilities for managing services and applications running in the Aneka Cloud.

a) **Infrastructure Management:** Aneka leverages virtual and physical hardware in order to deploy Aneka Clouds. Virtual hardware is generally managed by means of the Resource Provisioning Service, which acquires resources on demand according to the need of applications, while physical hardware is directly managed by the Administrative Console by leveraging the Aneka management API of the PAL. The management features are mostly concerned with the provisioning of physical hardware and the remote installation of Aneka on the hardware.

b) **Platform Management:** provides the basic layer on top of which Aneka Clouds are deployed. The creation of Clouds is orchestrated by deploying a collection of services on the physical infrastructure that allows the installation and the management of containers. A collection of connected containers defines the

platform on top of which applications are executed. The features available for platform management are mostly concerned with the logical organization and structure of Aneka Clouds. It is possible to partition the available hardware into several Clouds variably configured for different purposes. Services implement the core features of Aneka Clouds and the management layer exposes operations for some of them, such as Cloud monitoring, resource provisioning and reservation, user management, and application profiling.

c) Application Management: Applications identify the user contribution to the Cloud. The management APIs provide administrators with monitoring and profiling features that help them track the usage of resources and relate them to users and applications. This is an important feature in a cloud computing scenario in which users are billed for their resource usage. Aneka exposes capabilities for giving summary and detailed information about application execution and resource utilization. All these features are made accessible through the Aneka Cloud Management Studio, which constitutes the main Administrative Console for the Cloud.

DATA-INTENSIVE COMPUTING - MAPREDUCE PROGRAMMING

Data-intensive computing focuses on aa class of applications that deal with a large amount of data. Several application fields, ranging from computational science to social networking, produce large volumes of data that need to be efficiently stored, made accessible, indexed, and analyzed.

These tasks become challenging as the quantity of information accumulates and increases over time at higher rates. Distributed computing is definitely of help in addressing these challenges by providing more scalable and efficient storage architectures and a better performance in terms of data computation and processing.

MapReduce, which is a popular programming model for creating data-intensive applications and their deployment on clouds.

What is data-intensive computing?

Data-intensive computing is concerned with production, manipulation, and analysis of large-scale data in the range of hundreds of megabytes (MB) to petabytes (PB) and beyond. The term **dataset** is commonly used to identify a collection of information elements that is relevant to one or more applications. Datasets are often maintained in repositories, which are infrastructures supporting the storage, retrieval, and indexing of large amounts of information. To facilitate the classification and search, relevant bits of information, called **metadata,** are attached to datasets.

Data-intensive computations occur in many application domains. Computational science is one of the most popular ones. People

conducting scientific simulations and experiments are often keen to produce, analyze, and process huge volumes of data. Hundreds of gigabytes of data are produced every second by telescopes mapping the sky; the collection of images of the sky easily reaches the scale of petabytes over a year. Bioinformatics applications mine databases that may end up containing terabytes of data. Earthquake simulators process a massive amount of data, which is produced as a result of recording the vibrations of the Earth across the entire globe.

Besides scientific computing, several IT industry sectors require support for data-intensive computations. Customer data for any telecom company would easily be in the range of 10—100 terabytes. This volume of information is not only processed to generate billing statements, but it is also mined to identify scenarios, trends, and patterns that help these companies provide better service.

Characterizing Data-Intensive Computations

Data-intensive applications not only deal with huge volumes of data but, very often, also exhibit compute-intensive properties. Data-intensive applications handle datasets on the scale of multiple terabytes and petabytes. Datasets are commonly persisted in several formats and distributed across different locations.

TECHNOLOGIES FOR DATA-INTENSIVE COMPUTING

Data-intensive computing concerns the development of applications that are mainly focused on processing large quantities of data. Therefore, storage systems and programming models constitute a natural classification of the technologies supporting data-intensive computing.

DISTRIBUTED FILE SYSTEMS AND STORAGE CLOUDS

Distributed file systems constitute the primary support for data management. Mostly these file systems constitute the data storage support for large computing clusters, supercomputers, massively parallel architectures, and lately, storage/computing clouds.

Lustre: The Lustre file system is a massively parallel distributed file system that covers the needs of a small workgroup of clusters to a large-scale computing cluster. The file system is used by several of the Top 500 supercomputing systems. Lustre is designed to provide access to petabytes (PBs) of storage to serve thousands of clients with an I/O throughput of hundreds of gigabytes per second (GB/s).

IBM General Parallel File System (GPFS): GPFS is the high-performance distributed file system developed by IBM that provides support for the RS/6000 supercomputer and Linux computing clusters. GPFS is a multiplatform distributed file system built over several years of academic research and provides advanced recovery mechanisms.

Google File System (GFS): GFS is the storage infrastructure that supports the execution of distributed applications in Google's

computing cloud. The system has been designed to be a fault-tolerant, highly available, distributed file system built on commodity hardware and standard Linux operating systems.

Sector: Sector is the storage cloud that supports the execution of data-intensive applications defined according to the Sphere framework. It is a user space file system that can be deployed on commodity hardware across a wide-area network.

Amazon Simple Storage Service (S3): Amazon S3 is the online storage service provided by Amazon. Even though its internal details are not revealed, the system is claimed to support high availability, reliability, scalability, infinite storage, and low latency at commodity cost. The system offers a flat storage space organized into buckets, which are attached to an Amazon Web Services (AWS) account.

MapReduce PROGRAMMING MODEL

MapReduce is a programming platform Google introduced for processing large quantities of data. It expresses the computational logic of an application in two simple functions: map and reduce. Data transfer and management are completely handled by the distributed storage infrastructure (i.e., the Google File System), which is in charge of providing access to data, replicating files, and eventually moving them where needed.

Developers are provided with an interface that presents data at a higher level: as a collection of key- value pairs. The computation of **MapReduce** applications is then organized into a workflow of map and reduce operations that is entirely controlled by the runtime system; developers need only specify how the map and reduce functions operate on the key-value pairs.

The **MapReduce** model is expressed in the form of the two functions, which are defined as follows:

- map (k1, v1) —› list(k2, v2)
- reduce (k2, list(v2)) —› list(v2)

The map function reads a key-value pair and produces a list of key-value pairs of different types. The reduce function reads a pair composed of a key and a list of values and produces a list of values of the same type.