UNIT V

Aneka: Cloud Application Platform

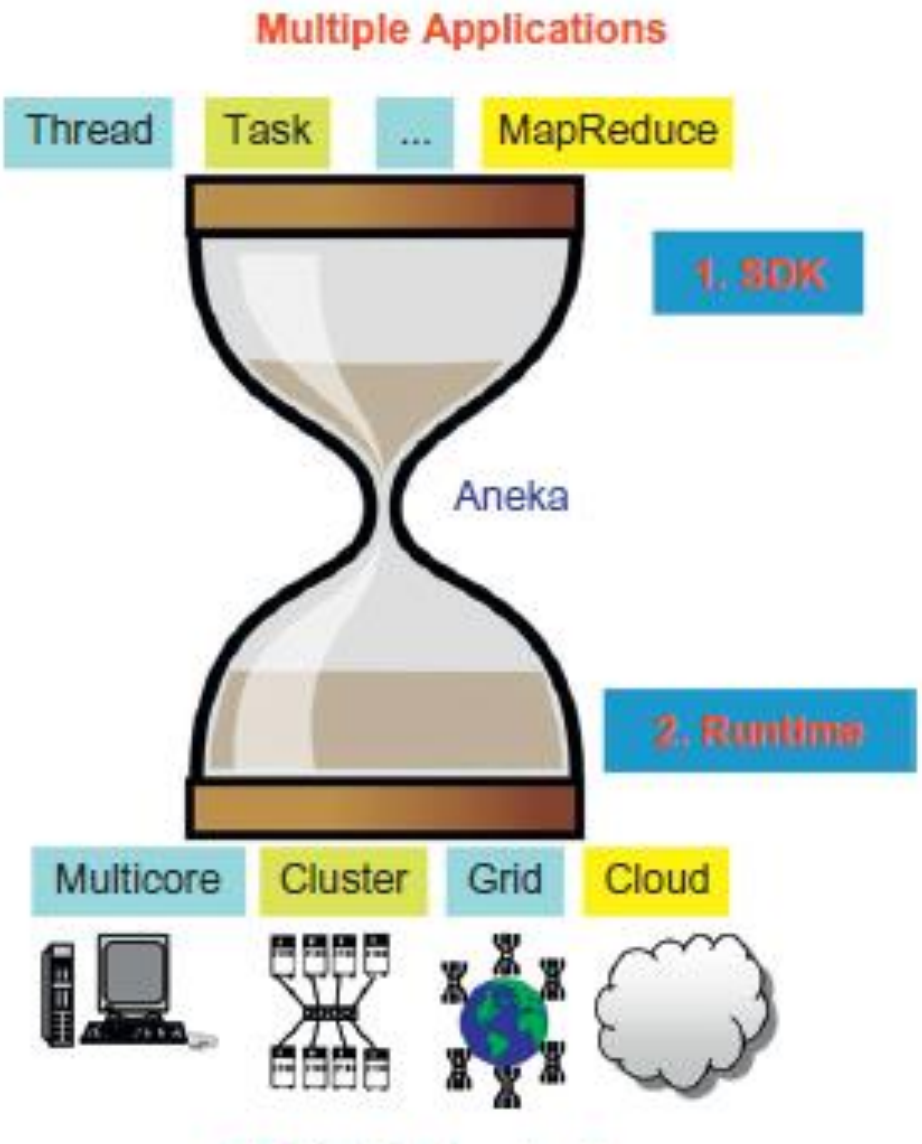
Syllabus

- Framework Overview
- Anatomy of the Aneka Container
  - From the Ground Up: Platform Abstraction Layer
  - Fabric Services
  - Foundation Services
  - Application Services

Introduction

- 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 (see below Figure).
- 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's capabilities at a glance



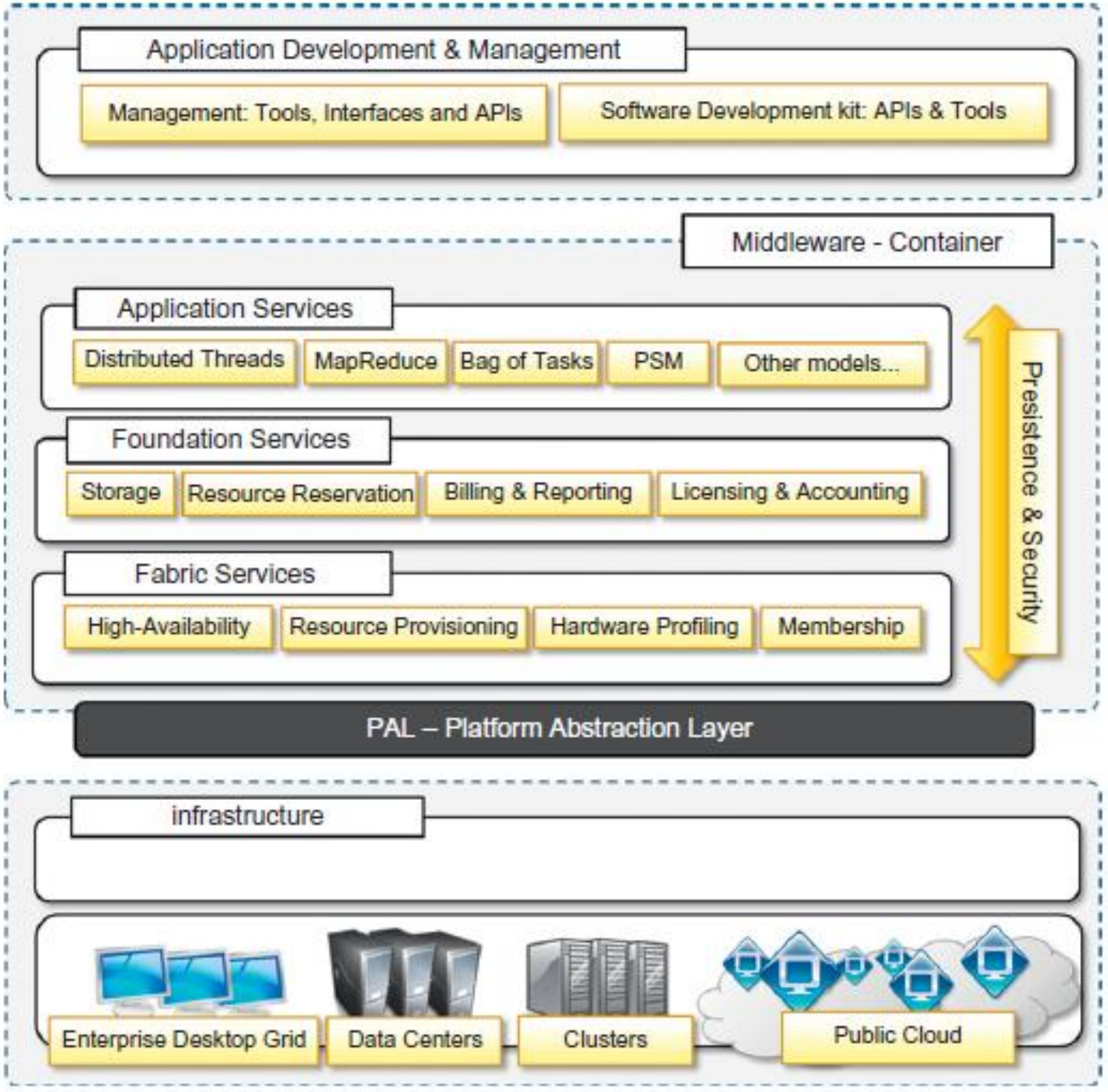
Framework Overview

- Aneka is a software platform for developing cloud computing applications.
- It allows harnessing of disparate computing resources and managing them into a unique virtual domain—the Aneka Cloud—in which applications are executed.
- According to the Cloud Computing Reference, 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

- Below 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.

Aneka framework overview


Application Development & Management
Management: Tools, Interfaces and APIs
Software Development kit: APIs & Tools
Middleware - Container
Application Services
Distributed Threads
MapReduce
Bag of Tasks
PSM
Other models...
Foundation Services
Storage
Resource Reservation
Billing & Reporting
Licensing & Accounting
Fabric Services
High-Availability
Resource Provisioning
Hardware Profiling
Membership
Presistence & Security
PAL – Platform Abstraction Layer
infrastructure
Enterprise Desktop Grid
Data Centers
Clusters
Public Cloud

Framework Overview

- A collection of interconnected containers constitute 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**.

Framework Overview

- Various Services offered by Aneka Cloud Platform are:
  - **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 run time machinery is responsible for keeping the infrastructure up and running and serves as a hosting environment for services.
  - **Resource management**: Aneka is an elastic infrastructure in which resources are added and removed dynamically according to application needs and user requirements.

Framework Overview

- **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 multi tenant 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.
- **QoS/SLA management and 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 is the unit of deployment in Aneka Clouds, and it is a lightweight software layer designed to host services and interact with the underlying operating system and hardware.
- The services installed in the Aneka container can be classified into three major categories:
  - Fabric Services
  - Foundation Services
  - Application Services

Anatomy of the Aneka container

- 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.

From the ground up: The Platform Abstraction Layer

- The core infrastructure of the system is based on the .NET technology and allows the Aneka container to be portable over different platforms and operating systems.
- The Common Language Infrastructure (CLI), defines a common runtime environment and application model for executing programs but does not provide any interface to access the hardware or to collect performance data from the hosting operating system.
- In a cloud environment each operating system has a different file system organization and stores that information differently.

From the ground up: The Platform Abstraction Layer

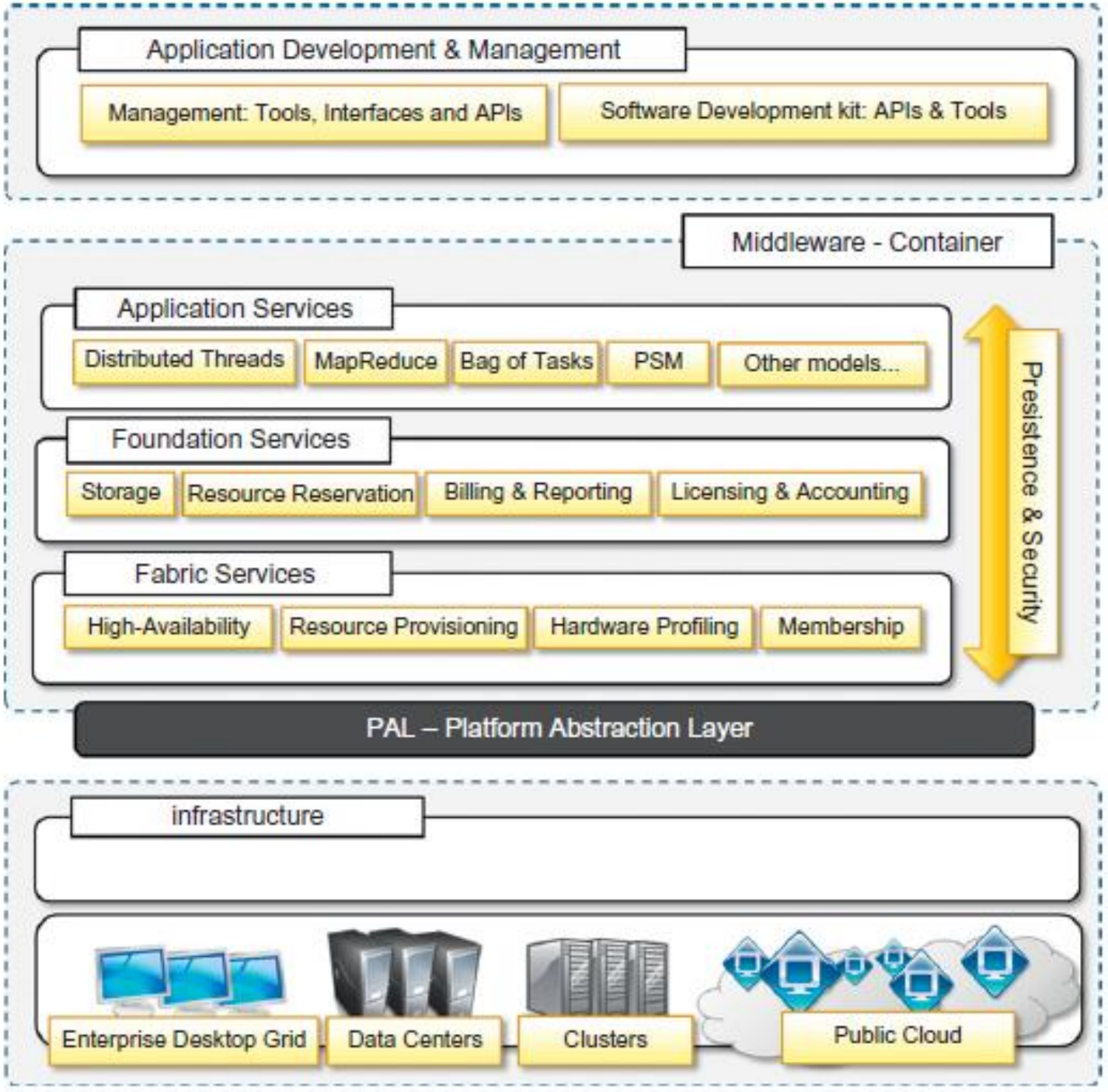
- The Platform Abstraction Layer (PAL) addresses this heterogeneity and provides the container with a uniform interface for accessing the relevant hardware and operating system information, thus allowing the rest of the container to run unmodified on any supported platform.
- The PAL is responsible for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

From the ground up: The Platform Abstraction Layer

- 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

From the ground up: The Platform Abstraction Layer

- 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


Application Development & Management
Management: Tools, Interfaces and APIs
Software Development kit: APIs & Tools
Middleware - Container
Application Services
Distributed Threads
MapReduce
Bag of Tasks
PSM
Other models...
Foundation Services
Storage
Resource Reservation
Billing & Reporting
Licensing & Accounting
Fabric Services
High-Availability
Resource Provisioning
Hardware Profiling
Membership
Presistence & Security
PAL – Platform Abstraction Layer
infrastructure
Enterprise Desktop Grid
Data Centers
Clusters
Public Cloud

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.
- **Monitoring services** allow for hardware profiling and implement a basic monitoring infrastructure that can be used by all the services installed in the container.
- The two services of Fabric class are:
  - Profiling and monitoring
  - Resource management

Profiling and monitoring

- Profiling and monitoring services are mostly exposed through the **Heartbeat, Monitoring, and Reporting Services**.
- The **first** makes available the information that is collected through the PAL; the **other two** implement a generic infrastructure for monitoring the activity of any service in the Aneka Cloud.

Profiling and monitoring

- The **Heartbeat Service** periodically collects the dynamic performance information about the node and publishes this information to the membership service in the Aneka Cloud.
- It collects basic information about memory, disk space, CPU, and operating system.
- These data are collected by the index node of the Cloud, and makes them available for reservations and scheduling services that optimizes them for heterogeneous infrastructure.
- A specific component, called **Node Resolver**, is in charge of collecting these data and making them available to the Heartbeat Service.

Profiling and monitoring

- The set of built-in services for monitoring and profiling is completed by a generic monitoring infrastructure, which allows any custom service to report its activity.
- This infrastructure is composed of the Reporting and Monitoring Services.
- The **Reporting Service** manages the store for monitored data and makes them accessible to other services or external applications for analysis purposes.
- On each node, an instance of the **Monitoring Service** acts as a gateway to the **Reporting Service** and forwards to it all the monitored data that has been collected on the node.
- Any service that wants to publish monitoring data can leverage the local monitoring service without knowing the details of the entire infrastructure.

Profiling and monitoring

- Currently several built-in services provide information through this channel:
  - The **Membership Catalogue** tracks the performance information of nodes.
  - The **Execution Service** monitors several time intervals for the execution of jobs.
  - The **Scheduling Service** tracks the state transitions of jobs.
  - The **Storage Service** monitors and makes available information about data transfer, such as upload and download times, filenames, and sizes.
  - The **Resource Provisioning Service** tracks the provisioning and lifetime information of virtual nodes.

Resource management

- Aneka provides a collection of services that are incharge of managing resources.
- These are the **Index Service (or Membership Catalogue) and Resource Provisioning Service**.
- The ***Membership Catalogue*** is Aneka's fundamental component for resource management; it keeps track of the basic node information for all the nodes that are connected or disconnected.
- The Membership Catalogue implements the basic services of a directory service, allowing the search for services using attributes such as names and nodes.

Resource management

- The **resource provisioning** infrastructure built into Aneka is mainly concentrated in the Resource Provisioning Service, which includes all the operations that are needed for provisioning virtual instances.
- The implementation of the service is based on the idea of resource pools.
- A resource pool abstracts the interaction with a specific IaaS provider by exposing a common interface so that all the pools can be managed uniformly.

Foundation Services

- 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 model scan 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

- 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.

Storage management

- Aneka offers **two** different facilities for storage management:
- A **centralized file storage**, which is mostly used for the execution of **compute- intensive applications**.
- A **distributed file system**, which is more suitable for the execution of **data-intensive applications**.
- As the requirements for the two types of applications are rather different. **Compute-intensive applications** mostly require powerful processors and do not have high demands in terms of storage, which in many cases is used to store small files that are easily transferred from one node to another. Here, a **centralized storage** node is sufficient to store data.
- **Centralized storage** is implemented through and managed by Aneka's Storage Service.

Storage management

- The protocols for implementing centralized storage management are supported by a concept of **File channel**. It consists of a **File Channel controller** and **File channel handler**.
- The **file channel controller** constitutes the server component of the channel, where files are stored and made available; the **file channel handler** represents the client component, which is used by user applications or other components of the system to upload, download, or browse files.
- **Data-intensive applications** are characterized by large data files (gigabytes or terabytes, petabytes) and here processing power required by tasks is not more.
- Here instead of centralized data storage a distributed file system is used for storing data by using all the nodes belonging to the cloud.
- Data intensive applications are implemented by means of a **distributed file system**. **Google File system** is best example for distributed file systems.

Storage management

- Typical characteristics of Google File system are:
  - Files are huge by traditional standards (multi-gigabytes).
  - Files are modified by appending new data rather than rewriting existing data.
  - There are two kinds of major workloads: large streaming reads and small random reads.
  - It is more important to have a sustained bandwidth than a low latency.

Accounting, billing, and resource pricing

- **Accounting Services** keep track of the status of applications in the Aneka Cloud.
- The information collected for accounting is primarily related to infrastructure usage and application execution.
- Billing is another important feature of accounting.
- Aneka is a multitenant cloud programming platform in which the execution of applications can involve provisioning additional resources from commercial IaaS providers.

Accounting, billing, and resource pricing

- Aneka Billing Service provides detailed information about each user's usage of resources, with the associated costs.
- Resource pricing is associated with the price fixed for different types of resources/nodes that are provided for the subscribers. Powerful resources are priced high and less featured resources are priced low.
- Two internal services used by accounting and billing are **Accounting service** and **Reporting Service**.

Resource reservation

- Resource Reservation supports the execution of distributed applications and allows for reserving resources for exclusive use by specific applications.
- **Two** types of services are used to build resource reservation:
  - **The Resource Reservation**
  - **The Allocation Service**
- **The Resource Reservation** keeps track of all the reserved time slots in the Aneka Cloud and provides a unified view of the system. (provides overview of the system)
- **The Allocation Service** is installed on each node that features execution services and manages the database of information regarding the allocated slots on the local node.

Resource reservation

- Different Reservation Service Implementations supported by Aneka Cloud are:
  - **Basic Reservation:** Features the basic capability to reserve execution slots on nodes and implements the alternate offers protocol, which provides alternative options in case the initial reservation requests cannot be satisfied.
  - **Libra Reservation:** Represents a variation of the previous implementation that features the ability to price nodes differently according to their hardware capabilities.
  - **Relay Reservation:** This implementation is useful in integration scenarios in which Aneka operates in an inter cloud environment.

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: the **Scheduling Service** and the **Execution Service**.

Scheduling

- **Scheduling Services** are in charge of planning the execution of distributed applications on top of Aneka and governing the allocation of jobs composing an application to nodes.
- They also constitute the integration point with several other **Foundation** and **Fabric Services**, such as the **Resource Provisioning Service**, the **Reservation Service**, the **Accounting Service**, and the **Reporting Service**.

Scheduling

- Common tasks that are performed by the scheduling component are the following:
  - Job to node mapping
  - Rescheduling of failed jobs
  - Job status monitoring
  - Application status monitoring

Execution

- **Execution Services** control the execution of single jobs that compose applications.
- They are in charge of setting up the runtime environment hosting the execution of jobs.
- Some of the common operations that apply across all the range of supported models are:
  - Unpacking the jobs received from the scheduler
  - Retrieval of input files required for job execution
  - Sandboxed execution of jobs
  - Submission of output files at the end of execution
  - Execution failure management (i.e., capturing sufficient contextual information useful to identify the nature of the failure)
  - Performance monitoring
  - Packing jobs and sending them back to the scheduler

Execution

- The various Programming Models Supported by Execution Services of Aneka Cloud are:
  - **Task Model** - This model provides the support for the independent "bag of tasks" applications and many computing tasks. In this model application is modeled as a collection of tasks that are independent from each other and whose execution can be sequenced in any order.
  - **Thread Model** - This model provides an extension to the classical multithreaded programming to a distributed infrastructure and uses the abstraction of Thread to wrap a method that is executed remotely.
  - **Map Reduce Model** - This is an implementation of Map Reduce as proposed by Google on top of Aneka.
  - **Parameter Sweep Model** - This model is a specialized form of Task Model for applications that can be described by a template task whose instances are created by generating different combinations of parameters.