Aneka: Cloud Application Platform

Dr. Prasenjit Chanak
Assistant Professor

Department of Computer Science and Engineering
Indian Institute of Technology (BHU), Varanasi-221005

Resource Reservation

- Aneka’s Resource Reservation supports the execution of distributed applications and allows for reserving resources for exclusive use by specific applications
- Resource reservation is built out of two different kinds of services: Resource Reservation and the Allocation Service
- Resource Reservation keeps track of all the reserved time slots in the Aneka Cloud and provides a unified view 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
- Applications that need to complete within a given deadline can make a reservation request for a specific number of nodes in a given timeframe.
- If it is possible to satisfy the request, the Reservation Service will return a reservation identifier as proof of the resource booking

Resource Reservation

- Aneka allows for different implementations of the service, which mostly vary in the protocol that is used to reserve resources or the parameters that can be specified while making a reservation request
- **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:** Constitutes a very thin implementation that allows a resource broker to reserve nodes in Aneka Clouds and control the logic with which these nodes are reserved. This implementation is useful in integration scenarios in which Aneka operates in an intercloud environment

Resource Reservation

- Resource reservation is fundamental to ensuring the quality of service that is negotiated for applications
- It allows Aneka to have a predictable environment in which applications can complete within the deadline or not be executed at all
- The assumptions made by the reservation service for accepting reservation requests are based on the static allocation of such requests to the existing physical (or virtual) infrastructure available at the time of the requests and by taking into account the current and future load
- This solution is sensitive to node failures that could make Aneka unable to fulfill the service-level agreement (SLA) made with users

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.
- 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
- Aneka does not provide a centralized scheduling engine, but each programming model features its own scheduling service that needs to work in synergy with the existing services of the middle- ware

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
- As happens for the scheduling services, each programming model has its own requirements, but it is possible to identify some common operations that apply across all the range of supported models:
    - 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

- Application Services constitute the runtime support of the programming model in the Aneka Cloud. Currently there are several supported models:
- **Task Model:** This model provides the support for the independent "bag of tasks" applications and many computing tasks. In this model, an 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
- **MapReduce Model:** This is an implementation of MapReduce as proposed by Google on top of Aneka
- **Parameter Sweep Model:** This model is a specialization of the Task Model for applications that can be described by a template task whose instances are created by generating different combinations of parameters, which identify a specific point into the domain of interest