Cloud programming and management

Aneka SDK

- 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

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
- Each distributed application running on top of Aneka is an instance of the ApplicationBase , M class, where M identifies the specific type of application manager used to control the application
- Application classes constitute the developers' view of a distributed application on Aneka Clouds, whereas application managers are internal components that interact with Aneka Clouds in order to monitor and control the execution of the application
- Aneka further specializes applications into two main categories: (1) applications whose tasks are generated by the user and (2) applications whose tasks are generated by the runtime infrastructure.

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

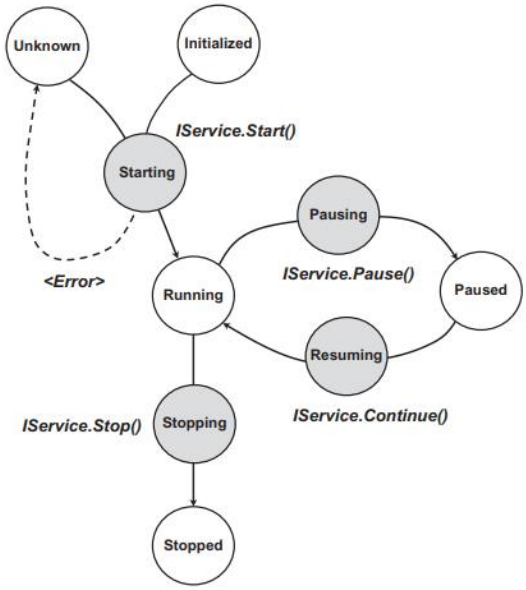


FIGURE 5.9
Service life cycle.

- The diagram above is the reference life cycle of each service instance in the Aneka container

Management tools

Infrastructure management

- The management features are mostly concerned with the provisioning of physical hardware and the remote installation of Aneka on the hardware.

Platform management

- 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

Application management

- 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

Data Intensive Computing: Map reduce Programming

- Data intensive computing is a 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 icreases over time at higher rates

- Here the distributed computing comes to rescue by providing more scalable and efficient storage architectures and a better performance in terms of data computation and processing
- The use of distributed computing is not straight forward, there are several challenges in form of data representation, efficient algorithm and scalable infrastructure are need to be rectified

What is Data Intensive Computing

- Data intensive computing deals 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 infrastructure supporting the storage,retrieval and indexing of large amount of information
- Inorder to facilitate the classification and the search of relevant bits of information, metadata are attached to datasets

Data Intensive Computing Application domains

- Computational Science
- Bioinformatics applications
- Earthquakes simulators
- Social media
- Several It industry
- Customer data

Characterizing Data Intensive Computations

- Data intensive applications not only deal with huge volumes of data but, very often, also exhibit compute intensive properties
- It handles datasets the scale of multiple terabytes and petabytes
- Datasets are commonly persisted in several formats and distributed across different locations
- Data processed in multistep nalytical pipelines, which includes transformation and fusion stages
- They also need efficient mechanisms for data management , filtering and fusion, and efficient querying and distribution

Challenges ahead

- Scalable algorithms that can search and process massive datasets
- New metadata management technologies that can scale to handle complex, heterogeneous, and distributed data sources
- Advances in high performance computing platform aimed at providing a better support for accessing in memory multi-terabyte structures
- High performance, high-relaible, petascale distributed file systems
- Data signature generation technique for data reduction and rapid processing
- New approaches to software mobility for delivering algorithms able to move the computation where the data is located
- Specialized hybrid interconnection architectures providing a better support for filtering mutigigabyte data streams coming from high speed networks and scientific intruments

- Flexible and high performance software integration techniques facilitating the combination of software modules running on different platforms to quickly form analytical pipelines

Historical Perspectives

Data Grids

- A data grid provides services helping users to discover, transfer, and manipulate large datasets stored in distributed repositories and also create and manage copies of them
- Data Grids offer two main functionalities: high performance and reliable file transfer for moving large amounts of data ; and scalable replica discovery and management mechanisms for an easy access to distributed datasets
- Data grids mostly provide storage and dataset management facilities as support of scientific experiments that produce huge volumes of data
- Datagrids have their own characteristics and introduce new challenges
    - Massive datasets: scale of gigabytes, terabytes and beyond
    - Shared data collections: includes distributed collections of data
    - Unified namespace: Every data element has single logical name to locate data collections and resources
    - Access restrictions: involve both coarse grained(large system) and fine grained(small system) access control
- E.g.,The LHC Grid, Bioinformatics Research Network(BIRN) etc.

Data Clouds and Big Data

- Big Data applies to datasets whose size is beyond the ability of commonly used software tools to capture, manage and process the data within a tolerable elapsed time
- Cloud Technologies support data intensive computing by
    - Providing a large amount of compute instances on demand
    - Providing a storage system optimized for keeping large blobs of data on distributed systems
    - Providing frameworks and programming APIsoptimized for the processing and management of large amount of data

Databases and Data-Intensive Computing

- Distributed databases are a collection of data stored at different sites of a computer network
- A distributed database can be created by splitting and scattering the data of an existing database over different sites, or by federating together multiple existing databases
- Each site providing services for the execution of local applications ,but also participates in the execution of a global application

Technologies for Data Intensive Computing

- Data-intensive computing concerns the development of applications that are mainly focused on processing large quantities of data.
- Traditional database management systems cannot be used because the data is unstructured.
- Due to *i) popularity of Big Data ii) importance of data analytics in business iii)unstructured nature of data and iv)evolution of cloud computing* which provides high computation power, some new database and data management techniques are indeed.
- Distributed file systems constitute the primary support for the management of data.

High performance distributed file systems

Lustre:

- A massively parallel distributed file system
- 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 including the one rated as the most powerful supercomputer in the June 2012 list.
- Provide access to petabytes (PBs) of storage, to serve thousands of clients with an IO throughput of hundreds of gigabytes per second (GB/s).
- Implements a robust failover strategy and recovery mechanism

- **IBM General Parallel File System (GPFS)**
  - high performance distributed file system developed by IBM providing support for RS/6000 supercomputer and Linux computing clusters

- Multiplatform distributed file system
- Provides advanced recovery mechanism
- Able to support petabytes of storage

High performance distributed file systems

Google File System (GFS):

- Is the storage infrastructure supporting the execution of distributed applications in the Google's computing Cloud
- Designed to be a fault tolerant, high available, distributed file system built on commodity hardware and standard Linux operating systems
- It has been designed with the following assumptions:
  - The system is built on top of commodity hardware that often fails.
  - The system stores a modest number of large files, multi-GB files are common and should be treated efficiently, small files must be supported but there is no need to optimize for that.
  - The workloads primarily consist of two kinds of reads: large streaming reads and small random reads.

- The workloads also have many large , sequential writes that append data to files
- High sustained bandwidth is more important than low latency

High performance distributed file systems

- **Amazon Simple Storage Service (S3):**
  - provided by Amazon.
  - The system offers a flat storage space organized into buckets, which are attached to an AWS (Amazon Web Services) account.
  - Each bucket can store multiple objects, each of them identified by a unique key.
  - Objects are identified by unique URLs and exposed through the HTTP protocol, thus allowing a very simple get-put semantics
  - It is also possible to define authenticated URLs, which provide public access to anyone for a limited (and configurable) period of time.

- E.g. Bitorrent

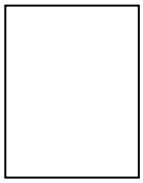
Not Only SQL (NoSQL) Systems

- The term NoSQL used to identify a relational database, which did not use a SQL interface to manipulate and query data, but relied on a set of UNIX shell scripts and commands to operate on text files containing the actual data.
- On many redefining through years, now the term NoSQL is a big umbrella encompassing all the storage and database management systems that differ in some way from the relational model
- This often implies the use of tables without fixed schemas to accommodate a larger range of data types or avoiding joins to increase the performance and scale horizontally
- Broad Classification of NoSQL :
  - *Document stores* (Apache Jackrabbit, Apache CouchDB)
  - *Tabular stores* (Google BigTable, Hadoop HBase, Hypertable).
  - Tuple Stores
  - Object databases

Apache CouchDB and MongoDB

- Document stores
- Both of them provide a schema-less store where the primary objects are documents organized into a collection of key-value fields. The value of each field can be of type string, integer, float, date, or an array of values.
- The databases use a RESTful interface and represents data in JSON format.
- allow querying and indexing data by using the MapReduce programming model
- expose Javascript as a base language for data querying and manipulation rather than SQL, and support large files as documents.
- the two systems support data replication and high-availability.
  - The two systems support data replication and high availability
  - CouchDB ensures ACID properties on data

Amazon Dynamo



- distributed key-value store offered by amazon
- an incrementally scalable and highly available storage system.
- build an infrastructure serving 10 million of requests per day across 1000 of servers.
- provides a simplified interface based on a *get/put* semantics, where objects are stored and retrieved with a unique identifier (key).
- an *eventually consistent* model, which means that in the long term all the users will see the same data

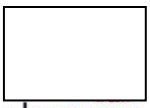
Google Bigtable

- distributed storage system designed to scale up to petabytes of data across thousands of servers
- Bigtable provides storage support for several Google applications
- The key design goals of Bigtable are wide applicability, scalability, high performance, and high availability.
- Bigtable organizes the data storage in tables whose rows are distributed over the distributed Google file system, Columns can be grouped in column family
- Client applications are provided with very simple data access model that allow them to address data at column level
- Each column value is stored in multiple versions that can be automatically time-stamped by Bigtable

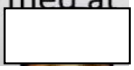
Hadoop HBase

- HBase is the distributed database supporting the storage needs of t Hadoop distributed programming platform.
- Inspired by Google Bigtable
- main goal is to offer real time read/write operations for tables with billions of rows and millions of columns by clustering hardware
- Internal structure is almost same as that of BigTable
- the entire system is backed by the Hadoop Distributed File System (HDFS), which mimics the structure and the services of GFS

Apache Cassandra

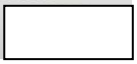


- Initially developed by Facebook. Provides storage support for several very large web applications such as *Facebook* itself, *Digg*, and *Twitter*
- Inspired from Amazon Dynamo which follows a fully distributed design and Google Bigtable from which inherits the "Column Family" concept.
- The system is designed to avoid a single point of failure and offer a highly reliable service
- based on the concept of table that is implemented as a distributed multi-dimensional map indexed by a key
- The APIs provided by the system to access and manipulate the data are very simple: insertion, retrieval, and deletion. The insertion is performed at row level, while retrieval and deletion can operate at column level.



The MapReduce Programming model

- MapReduce is a programming platform introduced by Google for processing large quantities of data.
- It expresses the computation logic of an application into two simple functions: *map* and *reduce*.
- Data transfer and management is completely handled by the distributed storage infrastructure (i.e. the Google File System), which is in charge of providing access to data, replicate files, and eventually move them where needed.
- developers have only to specify how the map and reduce functions operate on the key value pairs



Working of Map-Reduce Model

- 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 by a key and a list of values and produces a list of values of the same type.
  - *map (k1,v1) → list(k2,v2)*
  - *reduce(k2,list(v2)) → list(v2)*
- Explanation: Suppose Flipkart want to calculate the sale of 1 year in India city wise. So it may terabytes of data and usual program may run out of memory to calculate it. In map reducing work is going to be distributed. The mapper distributed on multiple machine get a small chunks of data, say for a month or even less. So each mapper work on affordable data and produce intermediate result. Next phase is shuffling and sorting, in which this intermediate result is sorted and directed to individual reducer. In this case each distributed reducer get a small data say sales data of Delhi or individual city from all intermediate data. Then reducer sum up all da produce final output.

MapReduce computation

- So, two major stages in the terms of MapReduce computations are:
  - **Analysis.** This phase operates directly to the data input file and corresponds to the operation of the map task. Moreover, the computation at this stage are widely parallel since map tasks are executed without any sequencing or ordering.
  - **Aggregation.** This phase operates on the intermediate results and it is characterized by operations which are aimed at aggregating, summing, and or elaborating the data obtained at the previous stage to present it into its final form. This is the task performed by the reduce function.

Aneka MapReduce Programming

- Aneka provides an implementation of the MapReduce abstractions by following the reference model introduced by Google and implemented by Hadoop.
- The *MapReduce Programming Model* defines the abstractions and the runtime support for developing MapReduce applications on top of Aneka.
- A Mapper and Reducer class that are extended from the AnekaMapReduce APIs. The runtime support is constituted by three main elements:
  - *MapReduce Scheduling Service*, which plays the role of the master process in the Google and Hadoop implementation.
  - *MapReduce Execution Service*, which plays the role of the worker pro the Google and Hadoop implementation.
  - A specialized distributed file system that is used to move data files.

Aneka MapReduce Infrastructure