ZooKeeper

ZooKeeper

A highly-available service for coordinating processes of distributed applications.

- Developed at Yahoo! Research
- Started as sub-project of Hadoop, now a top-level Apache project
- Development is driven by application needs

ZooKeeper

- Zookeeper provides a flexible coordination infrastructure for distributed environment. ZooKeeper framework supports many of the today's best industrial applications.
- Aims to provide a simple and high performance kernel for building more complex client
- Wait free
- FIFO
- No lock
- Pipeline architecture

Zookeeper - Overview

- ZooKeeper is a distributed co-ordination service to manage large set of hosts. Co-ordinating and managing a service in a distributed environment is a complicated process. ZooKeeper solves this issue with its simple architecture and API. ZooKeeper allows developers to focus on core application logic without worrying about the distributed nature of the application.

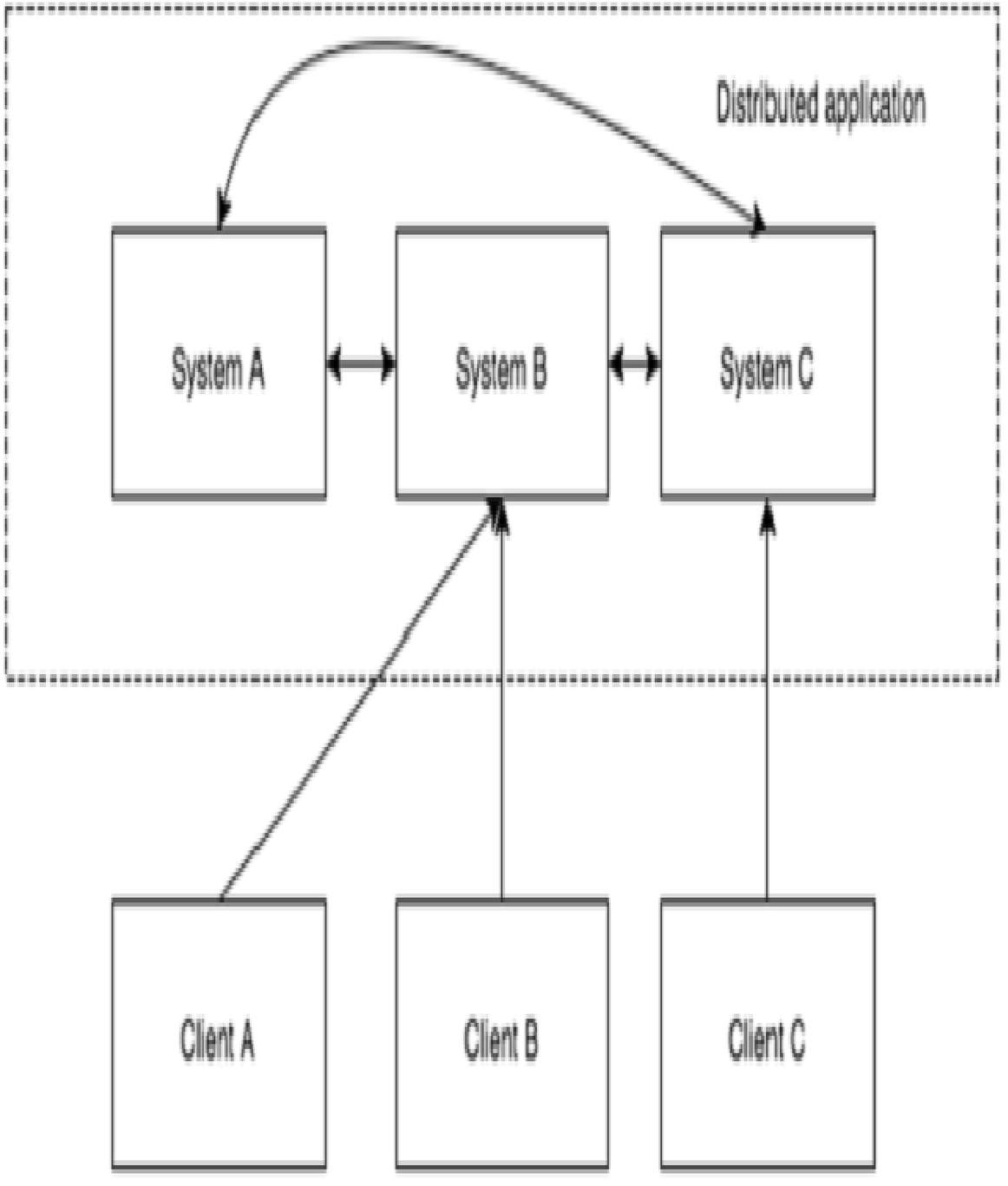
- The ZooKeeper framework was originally built at “Yahoo!” for accessing their applications in an easy and robust manner. Later, Apache ZooKeeper became a standard for organized service used by Hadoop, HBase, and other distributed frameworks. For example, Apache HBase uses ZooKeeper to track the status of distributed data.

Distributed Application

- A distributed application can run on multiple systems in a network at a given time (simultaneously) by coordinating among themselves to complete a particular task in a fast and efficient manner. Normally, complex and time-consuming tasks, which will take hours to complete by a non-distributed application (running in a single system) can be done in minutes by a distributed application by using computing capabilities of all the system involved.

- The time to complete the task can be further reduced by configuring the distributed application to run on more systems. A group of systems in which a distributed application is running is called a **Cluster** and each machine running in a cluster is called a **Node**.

- A distributed application has two parts, **Server** and **Client** application. Server applications are actually distributed and have a common interface so that clients can connect to any server in the cluster and get the same result. Client applications are the tools to interact with a distributed application.



- Benefits of Distributed Applications
  - **Reliability** − Failure of a single or a few systems does not make the whole system to fail.
  - **Scalability** − Performance can be increased as and when needed by adding more machines with minor change in the configuration of the application with no downtime.
  - **Transparency** − Hides the complexity of the system and shows itself as a single entity / application.
- Challenges of Distributed Applications
  - **Race condition** − Two or more machines trying to perform a particular task, which actually needs to be done only by a single machine at any given time. For example, shared resources should only be modified by a single machine at any given time.
  - **Deadlock** − Two or more operations waiting for each other to complete indefinitely.
  - **Inconsistency** − Partial failure of data

What is Apache ZooKeeper Meant For?

Apache ZooKeeper is a service used by a cluster (group of nodes) to coordinate between themselves and maintain shared data with robust synchronization techniques. ZooKeeper is itself a distributed application providing services for writing a distributed application.

The common services provided by ZooKeeper are as follows −

- **Naming service** − Identifying the nodes in a cluster by name. It is similar to DNS, but for nodes.
- **Configuration management** − Latest and up-to-date configuration information of the system for a joining node.
- **Cluster management** − Joining / leaving of a node in a cluster and node status at real time.
- **Leader election** − Electing a node as leader for coordination purpose.
- **Locking and synchronization service** − Locking the data while modifying it. This mechanism helps you in automatic fail recovery while connecting other distributed applications like Apache HBase.
- **Highly reliable data registry** − Availability of data even when one or a few nodes are down.

Distributed applications offer a lot of benefits, but they throw a few complex and hard-to-crack challenges as well. ZooKeeper framework provides a complete mechanism to overcome all the challenges. Race condition and deadlock are handled using **fail-safe synchronization approach**. Another main drawback is inconsistency of data, which ZooKeeper resolves with **atomicity**.

Benefits of ZooKeeper

–

- **Simple distributed coordination process**
- **Synchronization** − Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.
- **Ordered Messages**
- **Serialization** − Encode the data according to specific rules. Ensure your application runs consistently. This approach can be used in MapReduce to coordinate queue to execute running threads.
- **Reliability**
- **Atomicity** − Data transfer either succeed or fail completely, but no transaction is partial.

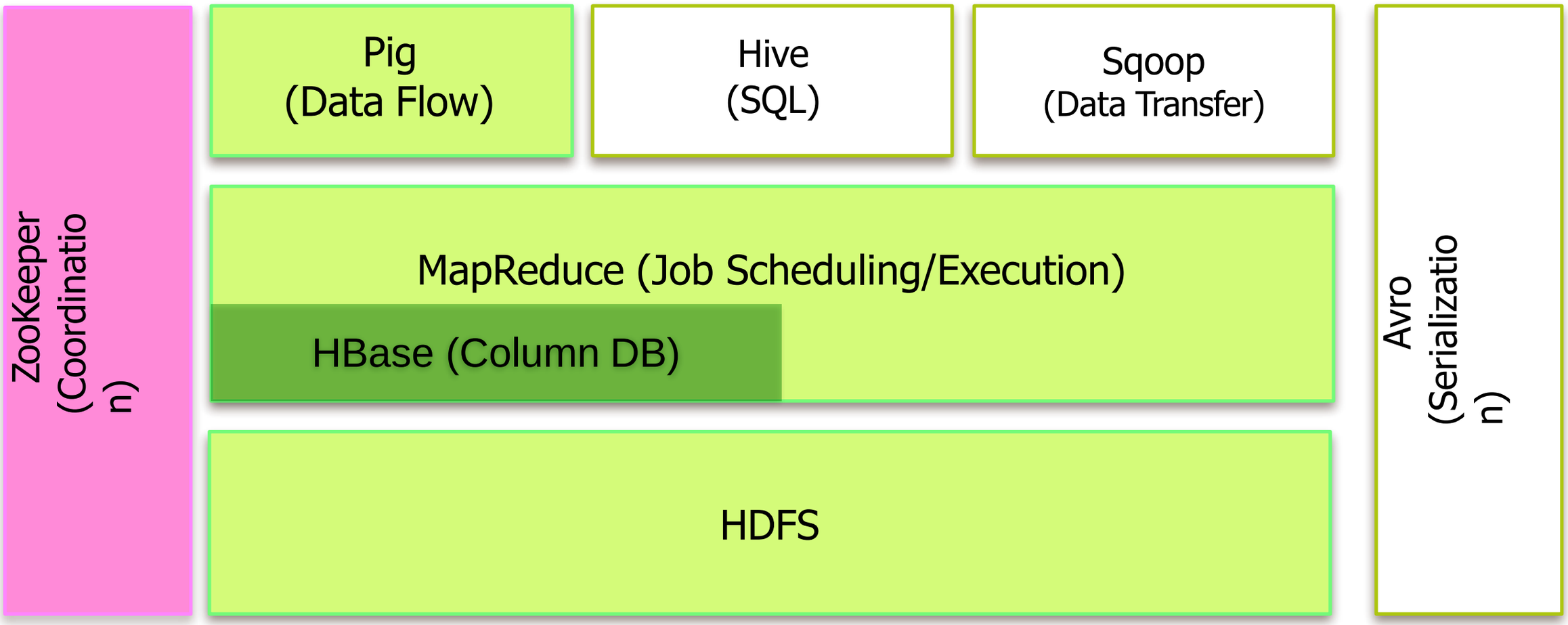
What is coordination?

Group membership
Leader election
Dynamic Configuration
Status monitoring

Coordination

Proper coordination is not easy.

ZooKeeper in the Hadoop ecosystem



Motivation

- In the past: a **single** program running on a **single** computer with a **single** CPU
- Today: applications consist of **independent** programs running on a **changing** set of computers
- Difficulty: **coordination** of those independent programs
- Developers have to deal with **coordination logic** and **application logic** at the same time

ZooKeeper: **designed** to relieve developers from writing coordination logic code.

Zookeeper - Installation

Before installing ZooKeeper, make sure your system is running on any of the following operating systems −

- **Any of Linux OS** − Supports development and deployment. It is preferred for demo applications.
- **Windows OS** − Supports only development.
- **Mac OS** − Supports only development.

ZooKeeper server is created in Java and it runs on JVM. You need to use JDK 6 or greater.

Now, follow the steps given below to install ZooKeeper framework on your machine.

- Step 2.1: Download ZooKeeper

  To install ZooKeeper framework on your machine, visit the following link and download the latest version of ZooKeeper. http://zookeeper.apache.org/releases.html
  As of now, the latest version of ZooKeeper is 3.4.6 (ZooKeeper-3.4.6.tar.gz).

- Step 2.2: Extract the tar file

  Extract the tar file using the following commands −

  ```
  $ cd opt/ $ tar -zxf zookeeper-3.4.6.tar.gz $ cd zookeeper-3.4.6 $ mkdir data
  ```

- Step 2.3: Create configuration file

  Open the configuration file named **conf/zoo.cfg** using the command **vi conf/zoo.cfg** and all the following parameters to set as starting point.

  ```
  $ vi conf/zoo.cfg tickTime = 2000 dataDir = /path/to/zookeeper/data clientPort = 2181 initLimit = 5 syncLimit = 2
  ```

  Once the configuration file has been saved successfully, return to the terminal again. You can now start the zookeeper server.

- Step 2.4: Start ZooKeeper server

  Execute the following command −

  ```
  $ bin/zkServer.sh start
  ```

  After executing this command, you will get a response as follows −

  ```
  $ JMX enabled by default $ Using config: /Users/../zookeeper-
  3.4.6/bin/../conf/zoo.cfg $ Starting zookeeper ... STARTED
  ```

- Step 2.5: Start CLI

  Type the following command −

  ```
  $ bin/zkCli.sh
  ```

  After typing the above command, you will be connected to the ZooKeeper server and you should get the following response.

  ```
  Connecting to localhost:2181 ..............  ..............  ..............
  Welcome to ZooKeeper! ..............  ..............  WATCHER:: WatchedEvent
  state:SyncConnected type: None path:null [zk: localhost:2181(CONNECTED) 0]
  ```

  Stop ZooKeeper Server

  After connecting the server and performing all the operations, you can stop the zookeeper server by using the following command.

  ```
  $ bin/zkServer.sh stop
  ```